Exakat Documentation

Release 1

Damien Seguy

Jun 24, 2021

Contents

Contents:

CHAPTER 1

Introduction

This is the documentation of the Exakat engine, version 2.2.2 (Build 1238), on Thu, 17 Jun 2021 15:56:34 +0000.

1.1 What is Exakat ?

Exakat is a tool for analyzing, reporting and assessing PHP code source efficiently and systematically. Exakat processes PHP 5.2 to 7.4 and 8.0 code, as well as reporting on security, performance, code quality, migration.

Exakat reads the code, builds an AST and several dependency graphs, then indexes all of it in a graph database. From there, exakat runs analysis, collecting potential errors and descriptive information about the code. Finally, exakat produces reports, both for humans and machines.

1.2 Use Cases

1.2.1 Code quality

Exakat detects hundreds of issues in PHP code : dead code, incompatible calls, undefined calls, illogical expressions, etc. Exakat is built for PHP, and cover common mistakes.

1.2.2 PHP version migration

Every PHP middle version is a migration by itself : based on the manual and common practices, exakat find both backward incompatibilities, that prevent migration, and new features, that makes code modern.

Exakat review code for minor version, and spot bug fixes that may impact the code.

1.2.3 Framework code quality

Common best practices and recommendations for specific plat-forms like Wordpress, CakePHP or Zend Framework are covered.

1.2.4 PHP configurations

Exakat detects several specialized analyzes, for Web security : making the code more secure online; PHP performances : allowing faster execution.

1.2.5 Security, performances, testability

Exakat has several specialized analyzes, for Web security : making the code more secure online; PHP performances : allowing faster execution; Testability : targeting the common pitfalls that makes code less testable.

1.2.6 Feature inventories

When auditing code, it is important to have a global view. Exakat collects all PHP features (magic functions, any operator, special functions or patterns) and represents them in one report, giving auditors a full view.

Exakat inventories all literals for later review, helping with the magic number syndrome and any data refactoring.

1.3 Exakat compared to others

1.3.1 Code sniffer

Automated coding standard violation detection for PHP review the code for syntax layout. Exakat is not a coding standard detection tool, as it focuses on bug finding, rather than coding layout.

While checking for coding standard, some bugs may be detected, and when checking for bugs, some coding standards may be found too.

Using AST, dependency graphs and knowledge databases, Exakat reviews the code, checks its potential usage and mis-usage. Exakat doesn't take any presentation nor comments into accounts : only functions, variables and their effects.

1.3.2 Phan, PHPstan, PHP

PHP code quality checks, based on type compatibility, and structure definitions. Exakat shares AST style analysis but it goes a bit further by including common mistakes and actual PHP features detections.

1.3.3 PHP7mar, PHP7cc

Code review for PHP 5 to migrate to PHP 7. Exakat covers every middle version from PHP 5.3 to PHP 7.3.

1.3.4 PHP-ci, Jenkins, Grumphp

Continuous integration and code quality management check the code by running code quality tools and collecting all the reported informations. Exakat is a good companion for those tools.

Exakat provides machine readable format reports, such as json, xml, text that may be consumed by CI. Exakat provides also human readable format, such as HTML, for interactive review of the reports, and a longer usage life span.

1.4 Platforms

Exakat is an Open Source tool. The code is available on Github.com/exakat/exakat, as Docker image and Vagrant file. It is also available as a phar download.

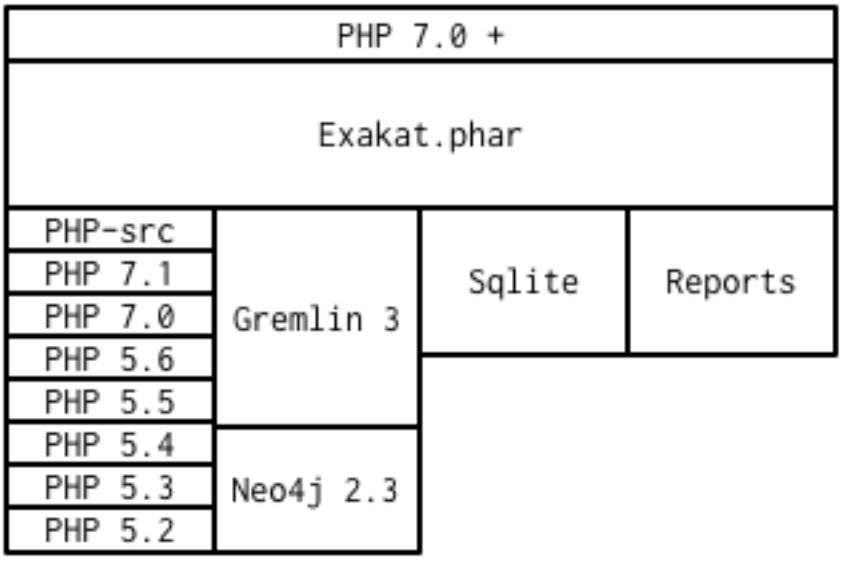
Exakat cloud is a SaaS platform, offering exakat audits on code, anytime, at reduced cost.

Exakat SAS is a Service company, providing consulting and training services around automated analysis and code quality for PHP.

1.5 Architecture

Exakat relies on PHP to lint and tokenize the target code; a graph database to process the AST and the tokens; a SQLITE 3 database to store the results and produce the various reports.

Exakat itself runs on PHP 7.2, with a short selection of extensions. It is tested with PHP 7.0 and 7.3.



Source code is imported into exakat using VCS client, like git, SVN, mercurial, tar, zip, bz2 or even symlink. Only reading access is actually required : the code is never modified in any way.

At least one version of PHP have to be used, and it may be the same running Exakat. Only one version is used for analysis and it may be different from the running PHP version. For example, exakat may run with PHP 7.2 but audit code with PHP 5.6. Extra versions of PHP are used to provide compilations reports. PHP middle versions may be configured separately. Minor versions are not important, except for edge cases.

The gremlin server is used to query the source code. Once analyzes are all finished, the results are dumped into a SQLITE database and the graph may be removed. Reports are build from the SQLITE database.

CHAPTER 2

Release Note

Here is the release note of exakat.

Version 2.2.2 (Si, coming up)

- **Architecture**
  -
- **Report**
  -
- **Analysis**
  -
- **Tokenizer**
  -

Version 2.2.1 (Chen, 2020-11-20)

- **Architecture**
  - Export : WIP of exporting PHP code from graph
  - New directives : rules_version_max, rules_version_min, ignore_rules and ignore_namespace
- **Report**
  - Sarif : Fixed line number that may be null or less
  - Ambassador : Fixed visibility report
- **Analysis**
  - New analysis : check for match as a keyword
  - New analysis : replace static variable by static properties
  - New analysis : warn about usage of get_object_vars()
  - New analysis : report global and static variables that are declared multiple times

  - Updated analysis : extended Used Classes to abstract classes
  - Updated analysis : wrong number of argument now supports $this()
  - Updated analysis : parse_str last argument doesn't apply anymore in PHP 8
  - Updated analysis : useless argument now omits parameter with default value
  - Checked unit tests : 3797 / 3800 test pass (99% pass)
- **Tokenizer**
  - Fixed race condition with phpdocs
  - Refactored static and global variables definitions (avoid double definitions)
  - Fixed detection of [] inside a list()
  - Fixed detection of alternative syntax for switch
  - Added use property to usenamespace too (for grouping)

Version 2.2.0 (Mao, 2020-10-15)

- **Architecture**
  - Extended Export command to produce PHP scripts from the graph database
  - Added more typehints
  - Added new command 'onefile'
  - Sped up database restart with id reset
  - Updated list of functions for several extensions. Started adding methods, class constants..
- **Report**
  - Ambassador : updated popularities
  - Ambassador : added missing PHP 8.0 ruleset
- **Analysis**
  - New analysis : report arguments and properties whose name clashes with the typehint
  - New analysis : report long preparation before throw command
  - New analysis : missing __isset() method
  - New analysis : suggest array_keys() for array_search in loops
  - New analysis : array_map() complains with values by reference
  - New analysis : report final private properties
  - New analysis : report misnamed constant/variable
  - New analysis : check for attribute configuration (PHP 8.0)
  - New analysis : suggest dropping variable in catch clause
  - New analysis : report resources that should not be tested with is_resource (PHP 8.0)
  - New analysis : check for named arguments and variadic
  - Updated analysis : wrong number of argument now supports $this()
  - Updated analysis : redefined private property uses OVERWRITE
  - Updated analysis : refactored UndefinedFunctions for speed

  - Updated analysis : array_map() complains with values by reference
  - Updated analysis : removed false positives on properties in strings
  - Updated analysis : unsupported types with operators skips cast values
  - Updated analysis : cancelled parameters are also for array_map/array_walk
  - Updated analysis : variable variable skips variables inside strings
  - Updated analysis : removed functions are not reported when in if/then with function_exists()
  - Updated analysis : wrong optional parameter fixed false positive with …
  - Updated analysis : extended list of removed directives, functions and constants
  - Removed analysis : RealVariables
  - Checked unit tests : 3761 / 3772 test pass (99% pass)
- **Tokenizer**
  - Added Void to empty default/case
  - Bitoperation added to isRead
  - Fixed list[] in a Foreach
  - Fixed token T_OPEN_DOLLAR_CURLY_BRACKET

Version 2.1.9 (Yin, 2020-10-01)

- **Architecture**
  - Removed old and unused commands
  - Modernized usage of docker as phpexec
  - New directive php_extensions to managed list of ext
- **Report**
  - Ambassador : removed 3 gremlins from typehint stats, added scalar types
  - New Migration80 report, dedicated to PHP 8.0 migrations
  - New Stubs.ini report, dedicated to exakat extensions production
- **Analysis**
  - New analysis : report arguments which are not nullable because of constants.
  - New analysis : could use stringable interface
  - New analysis : suggest explode()'s third argument when applicable
  - New analysis : suggest PHP 8.0 promoted properties
  - New analysis : report arrays with negative index, and auto-indexing
  - New analysis : report unsupported types with operators
  - New analysis : report usage of track_errors directive (PHP 8.0)
  - New analysis : report useless types on __get/__set
  - New analysis : count the number of use expressions in a file
  - New analysis : Avoid modifying typed arguments
  - New analysis : Report Assumptions in the code

  - New analysis : array_fill() usage with objects
  - New analysis : mismatch between parameter name and type
  - Updated analysis : magic methods definitions also find usage for __invoke()
  - Updated analysis : noscream operator usage may have exceptions
  - Updated analysis : identical methods and identical closures
  - Updated data : list of exceptions and their emitters

- **Tokenizer**
  - Upgraded detection of extensions' structures, beyond functions

Version 2.1.8 (Chou, 2020-09-18)

- **Architecture**
  - added '–' options, and kept the '-' options, for migration purposes. (–format and -format are both available)
  - Added support for PHP 8 attributes in dump.sqlite
  - Added 'precision' to rule docs.
  - Moved all but one data collection from Dump -collect to Dump/ analysis.
- **Report**
  - New report : SARIF
  - Typehint suggestion report : Tick classes when they are fully covered
  - Weekly report : fix donuts display.
  - Stubsjson : Added support for PHP attributes
  - Stubs : Added support for PHP attributes
- **Analysis**
  - New ruleset : CI-Checks
  - New analysis : 'Multiple declare(strict_types = 1)'
  - New analysis : 'No more (unset) in PHP 8'
  - New analysis : Cancel methods in parent : when methods should not have been abstracted in parent class.
  - New analysis : '$php_errormsg is removed in PHP 8'
  - New analysis : 'Mismatch Parameter Name' checks parameter names between inherited methods for consistency
  - Upgraded analysis : 'Useless Arguments' is accelerated
  - Upgraded analysis : 'Don't use Void' weeded out false positives
  - Upgraded analysis : 'Wrong type for native calls' weeded out false positives
  - Upgraded analysis : 'Non static methods called statically' was refactored for PHP 8.0 support
  - Upgraded analysis : 'PHP Keywords' includes 'match'
  - Upgraded analysis : 'Useless instruction' reports '$a ?? null' as useless.
  - Upgraded analysis : 'Uncaught exceptions' is extended to local variables

  - Upgraded analysis : 'Foreach favorites' also covers the keys
  - Upgraded analysis : 'Should Preprocess' skips expressions with constants
  - Upgraded analysis : 'Compare Hashes' has more functions covered
  - Removed analysis : 'Normal Properties' : no need anymore.

- **Tokenizer**
  - Moved isPhp attribute to Task/Load plugin
  - Created isExt attribute to Task/Load plugin

Version 2.1.7 (zi, 2020-09-07)

- **Architecture**
  - Refactored loading class, to keep query load at optimal size for Gremlin
  - GC during load to free memory
  - More typehints
  - Move several collections to Dump/ ruleset
- **Report**
  - Upgraded Typesuggestion report with report on closures and arrow functions
  - Added Arrowfunctions in inventories
  - Added collection of arguments and details for closures and arrowfunctions
- **Analysis**
  - New analysis : Could Be In Parent : suggest methods that should be defined in a parent
  - New analysis : Don't pollute namespace
  - New analysis : report insufficient return typehints
  - Upgraded analysis : 'Method signature must be compatible' now PHP 8.0 compatible
  - Upgraded analysis : 'Wrong type with native function' fixes false positives
  - Upgraded analysis : 'Same condition' added coverage for || conditions
  - Upgraded analysis : 'Missing returntype' extended to class typehints
  - Upgraded analysis : 'Should Use This' also covers special functions like get_class_called()
  - Upgraded analysis : 'No concat in loop' skips nested loops
  - Upgraded analysis : 'Always false' covers typehint usage
  - Upgraded analysis : 'NoChoice' doesn't report large expressions
  - Upgraded analysis : 'Dont mix PlusPlus' skip () and =
  - Upgraded analysis : 'Fallthrough' don't report final cases without break
  - Checked unit tests : 3663 / 3630 test pass (99% pass)
- **Tokenizer**
  - Removed 'root' property
  - Upgraded to new Attributes #[] in detection and normalisation
  - Fixed constant detection within instanceof

  - Created RETURN and RETURNED for Arrowfunctions (there is no return otherwise)
  - Parent method also calls children methods when those are not defined there
  - Support for multiple attributes in one syntax

Version 2.1.6 (Night Patrol Deity, 2020-08-28)

- **Architecture**
  - More typehints coverage
  - Various speed-up
  - Lighter logging with gremlin
  - Fixed installation path
- **Report**
  - Upgraded Typesuggestion report
  - Upgraded Stubs and Stubsjson
- **Analysis**
  - New analysis : report PHP 8.0 unknown parameters
  - New analysis : overwritten methods with different argument counts
  - New analysis : Warn of iconv and TRANSLIT for portability
  - New analysis : Warn of glob and {} for portability
  - Upgraded analysis : 'Useless check' covers new situations.
  - Upgraded analysis : 'Abstract away' now covers new calls.
  - Upgraded analysis : 'Must return Typehint' skips Void.
  - Upgraded analysis : 'Missing new' with less false positives
  - Checked unit tests : 3559 / 3630 test pass (98% pass)
- **Tokenizer**
  - Support for Virtualmethod and imports from traits
  - Refactored Usenamespace atom
  - Fixed calculations of fullnspath for static::class
  - Fixed detection of null/true/false in new()
  - Added support for T_BAD_CHARACTER

Version 2.1.5 (Day Patrol Deity, 2020-08-04)

- **Architecture**
  - Fixed comment size estimation by 1 for T_COMMENT
  - Added more typehints to code
- **Report**
  - Typehint suggestions : added ticks to fully typed methods
  - Emissary : Extract more information from dump.sqlite, instead of datastore.sqlite
  - Ambassador : Added a list of parameters, defined in the application

  - Ambassador : Added a list of fossilised methods
  - Stubs : Added check around PHP native functions and CIT
  - StubsJson : Added property for PHP native structures

- **Analysis**
  - New analysis : Report insufficient initialisation for array_merge() collector variable
  - New analysis : Report useless triple equals
  - New analysis : Don't compare typed boolean return values
  - New analysis : Report wrong type used with PHP functions
  - New analysis : Suggest abstracting away some PHP native functions
  - New analysis : Report try block that are too large
  - New analysis : Report variables potentially undefined in catch clause
  - New analysis : Report swapped arguments in methods overwriting
  - Upgraded analysis : InvalidPackFormat speed up
  - Upgraded analysis : Added parameter to Security/ShouldUsePreparedStatement to choose the preparing method
  - Upgraded analysis : Added parameter to Security/HardcodedPasswords to choose the name of properties/index
  - Upgraded analysis : PHP 8.0 new scalar typehint, stringable interface

- **Tokenizer**
  - Added support for named parameters (PHP 8.0)
  - Trimmed some properties from atoms
  - Removed non-existent atom mentions
  - Added support for Attributes (WIP)
  - Added support for ?->
  - Added support for new T_*_NAME tokens

Version 2.1.4 (Marshal of Heavenly Blessing, 2020-07-23)

- **Architecture**
  - Added time of last commit in audit results
  - Added more typehints
  - Upgraded PHP native method description with typehints (WIP)

- **Report**
  - Typehint suggestion report
  - New toplogies : call order,
  - Ambassador : new statistics for typehint usage

- **Analysis**
  - New analysis : Report double assignation of objects
  - New analysis : Typehints/CouldBe*, which makes suggestions for typehints

  - New analysis : Checks for argument type when typehint is present in custom methods
  - Upgraded analysis : Too Many Finds may be configured for threshold and prefix/suffix
  - Upgraded analysis : Typehints stats were extended to properties and multiple typehints
  - Upgraded analysis : Global outside Loop is extended to static variable too
  - Upgraded analysis : ErrorMessages also detect local variable contents
  - Upgraded analysis : Speed up for NullBoolean, Interfaces IsNotImplemented, InvalidPackFormat, arrayIndex, noWeakCrypto
  - Checked unit tests : 3532 / 3496 test pass (99% pass)

- **Tokenizer**
  - Removed 'aliased' property in atoms
  - Fixed spotting of PHP native constants, when in Define() structure
  - Fixed loading of false values
  - Added support for the trailing comma in closure's use expression
  - more handling of phpdocs
  - Null is now reused when it is a default value, as a typehint.
  - Logical was split in two : Logical and Bitoperation
  - Added support for match() {} expression
  - Fixed boolean calculations during Load
  - Removed auto-referencing in DEFAULT calculations

Version 2.1.3 (Marshal of the Heavenly Canopy, 2020-07-02)

- **Architecture**
  - Removed all usage of datastore in Reports, and only rely on dump.
  - ignore_rules is now case insensitive
  - Moved some of the loading to a separate gremlin call to reduce the size of node load.
  - Fixed the branch option with Git calls.
  - Storing trait's use expresion's options.
- **Report**
  - Ambassador ; New inventory : PHP protocol used (php, phar, glob://...)
  - Stubs and StubsJson, have been tested extensively
- **Analysis**
  - New analysis : report double assignations of the same object ($a = $b = new C)
  - New analysis : report cyclic references
  - Upgraded analysis : Used Constants edge situations
  - Upgraded analysis : No real comparison : extended analysis to constants
  - Upgraded analysis : extended detection of dynamic method calls to call_user_func*
  - Upgraded analysis : paths are detected with new functions

  - Checked unit tests : 3490 / 3520 test pass (99% pass)

- **Tokenizer**
  - More phpdoc support (from code to report)
  - Added isPHP to absolute FQN notations

Version 2.1.2 (Mountain Deity, 2020-06-25)

- **Architecture**
  - Removed files task from initproject.
  - Added ignore_rule directive, to ignore specific rules while running a specific report
  - More documentation (in particular, modifications section)
  - Exakat avoids to return twice the same results (file and line)
  - Sped up some analysis, and added a time limit per analysis
  - Removed double linking for static variables
- **Report**
  - New reports ; Stubs and StubsJson, which produce the stubs of the audited code (PHP and JSON format) (WIP)
  - New report ; Typehint suggestion (WIP)
  - Ambassador ; offers the configuration for all the rules that spotted issues in the current audit, for reuse in other codes
  - Collect the number of property per class
- **Analysis**
  - New analysis : Report methods that are too much indented on average
  - New analysis : Report possible confusion between a class and an alias
  - New analysis : Report variables that are static and global at the same time
  - New analysis : Report statement with long blocks
  - New analysis : Report phpdoc's deprecated methods and function calls
  - Upgraded analysis : Dereferencing levels now include () and =
  - Upgraded analysis : Unused Methods now skips classes that calls themselves dynamically
  - Upgraded analysis : No Need Get_class() was refactored
  - Upgraded analysis : Avoid Optional Properties was refactored
  - Upgraded analysis : Variable inconsistent Usage was extended with more reach
  - Upgraded analysis : Indirect Injections was upgraded with better reach with variables
  - Upgraded analysis : Direct Injections was upgraded with include
  - Upgraded analysis : PHP 8.0 new scalar typehint, stringable interface
  - Upgraded analysis : Mismatch Type and default now avoids undefined constants
  - Upgraded analysis : Wrong Optional Parameter is upgraded for PHP 8.0
  - Upgraded analysis : Indentation level was refactored
  - Checked unit tests : 3480 / 3510 test pass (99% pass)

- **Tokenizer**
  - Upgraded detection of PHP native constants, when they are in absolute notation
  - Dump task stores use expressions' options, plus minor fixes
  - Added support for Attributes (PHP 8.0)
  - Added support for Union types (PHP 8.0)
  - AtomIs step (WITH_VARIABLE) was extended with local variables
  - DEFAULT doesn't point anymore on auto-updated values
  - Extended support for phpdoc in the code
  - Added support for promoted properties (PHP 8.0)

Version 2.1.1 (Earth Deity, 2020-06-01)

- **Architecture**
  - Using timeLimit() to prevent Gremlin from running too deep in the rabbit hole
  - Added Neo4j Graphson V3 Graph driver
  - Moved 'Dump' rules to a specific Ruleset for easier administration
  - Propagated the upgrade to PHP 8.0 union types to three more rules
  - Fixed access to the list of ignored files
  - Added support for explicit stub files
  - Fixed multiple calls to Dump (better reentrant)
- **Report**
  - New report : Meters, which holds measures for the audited code.
  - Ambassador : inventory of OpenSSL ciphers
- **Analysis**
  - New analysis : Report unused traits
  - New analysis : Report chmod 777 system calls
  - New analysis : Check for keylength when generated by PHP
  - New analysis : Report methods with prefix/suffix and expected typehint
  - New analysis : Mark classes when they call dynamically their own methods
  - New analysis : Check for constants hidden in variable names ${X} != $X;
  - New analysis : Throw will be an expression in PHP 8.0
  - Upgraded analysis : Dangling operator now checks for loops too
  - Upgraded analysis : 'Variables used once' now skips variable definitions
  - Upgraded analysis : 'Access Private' takes into account dynamic classes
  - Upgraded analysis : 'Could Centralize' now uses a custom threshold. Default is 8 usage of an expression to centralize.
  - Upgraded analysis : 'Return true/false' checks that they are alone in the blocks
  - Upgraded analysis : 'Unreachable code' checks on constants values before reporting the next expression

  - Upgraded analysis : 'Magic methods' are case insensitive
  - Upgraded analysis : 'No Hardcoded passwords' has new functions that require a password
  - Upgraded analysis : 'Unused methods' are omitted for dynamically called methods and overwritten methods
  - Upgraded analysis : Insufficient Property Typehint also works for untyped properties
  - Upgraded analysis : PHP 8.0 new scalar typehint, stringable interface
  - Checked unit tests : 3383 / 3444 test pass (98% pass)
- **Tokenizer**
  - Arguments with null as default values, automatically are nullable
  - Intval is also an integer for logical operations
  - Default Values now omits recursives assignations
  - Fixed fullnspath for PHP short tags
  - Added link between new command and constructor of anonymous classes.

Version 2.1.0 (City God, 2020-05-13)

- **Architecture**
  - results stored in HashResults are now testable
  - Moved all query methods to Query/DSL namespace, from Analyzer class
- **Report**
  - New report : ClassReview, with focus on classes structures
  - New report : Typechecks, with focus on type hint usage
  - Ambassador : Added typehint stats section
  - Ambassador : fixed display of classes name in classes tree
  - Ambassador : some missing sections have been rehabilitated
- **Analysis**
  - New analysis : Trailing comma in signature (PHP 8.0)
  - New analysis : Hidden nullable types
  - New analysis : Not implemented abstract methods
  - New analysis : Report confusion between variables and arguments with arrow functions
  - Upgraded analysis : No literal for reference was extended
  - Upgraded analysis : Add zero is extended to constants
  - Upgraded analysis : This is for classes is now valid with arrow functions
  - Upgraded analysis : Useless arguments takes also into account constants
  - Upgraded analysis : Wrong Type With Call supports variadic arguments
  - Upgraded analysis : Extension constants now support fully qualified names
  - Upgraded analysis : Bad Typehint relay is compatible with union types
  - Upgraded analysis : Multiple Identical Cases now handles constants too

  - Checked unit tests : 3437 / 3477 test pass (99% pass)

- **Tokenizer**
  - Restored 'List' atom
  - Interface methods are now 'abstract' by default
  - Added 'array' typehint for variadic arguments
  - Distinguish between argument and local variable in fn functions
  - Removed nullable property
  - propagate calls now propagates closures and arrow functions
  - Added support for union types (PHP 8.0)
  - Check all error messages from php, not just the first ones

Version 2.0.9 (Jialan, 2020-04-30)

- **Architecture**
  - Added option in TU for analysis that won't fill the result table.
  - Reduced the number of duplicate links in the graph
  - Upgraded tokens for PHP 8.0.
- **Analysis**
  - New analysis : Don't collect void
  - New analysis : Wrongly inited properties
  - New analysis : Not inited properties
  - Upgraded analysis : PHP 8.0 removed functions
  - Upgraded analysis : Useless instructions also include global/static variables
  - Upgraded analysis : Bad Relay Function now works with return types and property types
  - Upgraded analysis : 'Scalar or object properties' are upgraded with static calls
  - Removed analysis : Classes and Arrays IsRead and IsModified. Use properties now.
  - Checked unit tests : 3347 / 3420 test pass (97% pass)
- **Tokenizer**
  - Fixed edge case for xor, with intval
  - Refactored multiple calculation for cast values
  - Added support for links between constants and use expressions
  - Linked classes with calls, when using use expression

Version 2.0.8 (Ao Run, 2020-04-20)

- **Architecture**
  - Added new information in dump.sqlite, to make report autonomous
- **Analysis**
  - Upgraded analysis : Paths are also recognized with constants, and more functions
  - Upgraded analysis : Should Use single Quotes

  - Checked unit tests : 3328 / 3398 test pass (97% pass)

- **Tokenizer**
  - Fixed detection of PHP constants

Version 2.0.7 (Ao Shun, 2020-04-14)

- **Architecture**
  - Adopted strict_types
  - Removed ctype1 attribute
  - Moved linting into separate processes
  - Refactored analysis to export to dump via SQL
  - Added 'None' ruleset to Dump task
- **Report**
  - Ambassador : Added Constant's order report
  - None : Added support for No report
- **Analysis**
  - Upgraded analysis : Undefined class constants
  - Upgraded analysis : Undefined global constants
  - Upgraded analysis : Undefined property
  - Checked unit tests : 3347 / 3420 test pass (97% pass)
- **Tokenizer**
  - Support PHP 8.0's tokens
  - Added support for multiple typehint in the engine
  - Fixed edge case for boolean type casting

Version 2.0.6 (Ao Qin, 2020-03-04)

- **Architecture**
  - Refactored analysis types for first UT
  - Moving to PHP 7.4 by default
- **Report**
  - Rector : added more coverage
  - All : better display of typed properties
- **Analysis**
  - New analysis : Semantic names of arguments
  - New analysis : !$a == $b
  - New prototype : possibles interfaces
  - Upgraded analysis : Overwritten literals now skips .=
  - Upgraded analysis : Scalar or object handles return type
  - Checked unit tests : 3322 / 3420 test pass (97% pass)

Version 2.0.5 (Ao Guang, 2019-11-25)

- **Architecture**
  - Fixed access to severity and timetofix from compiled extension
- **Report**
  - Ambassador : Fixed links to documentation
- **Analysis**
  - Upgraded analysis : Mismatched Type and Default now omit undefined constants
  - Checked unit tests : 3366 / 3402 test pass (99% pass)

Version 2.0.4 (Army Defeating Star of Heaven's Gate, 2019-11-18)

- **Architecture**
  - Reducing Analyzer's class method count
  - Moving more collections to Dump/ and Complete/
- **Report**
  - Rector : added more coverage
  - Ambassador : Skiped analysis are now reported, not with -1
  - Ambassador : Foreach favorites's graph is displayed
  - Ambassador : Visibility suggestion has full method names
- **Analysis**
  - Upgraded analysis : Don't Mix ++ now skips $a[$b++]
  - Upgraded analysis : Type hint stats skips some return values
  - Checked unit tests : 3365 / 3401 test pass (99% pass)

Version 2.0.3 (Military Star of the North Pole, 2019-11-11)

- **Architecture**
  - Added check on xdebug presence (nesting limit)
  - Moving more collections to Dump/
- **Analysis**
  - New analysis : Nullable typehint requires a test on NULL
  - New analysis : Typehint that requires too much
  - Upgraded analysis : Printf check on arguments works with '.'
  - Upgraded analysis : No magic for arrays skips __get()
  - Upgraded analysis : Const recommended, but not when methods are used
  - Upgraded analysis : Written only variables handles compact()
  - Upgraded analysis : Callbacks need returns, but not for spl_autoload_register()
  - Upgraded analysis : Extended analysis to Concatenation an Heredoc for Email
  - Upgraded analysis : Disconnected classes handles case sensitivity
  - Checked unit tests : 3371 / 3397 test pass (99% pass)

Version 2.0.2 (Danyuan Star of Honesty and Chasity, 2019-11-04)

- **Architecture**
  - Adding more typehint
  - Created new class to build Dot files
  - Cleaned double examples
  - Dump handles multiple definitions for constants, class, trait, functions.
- **Report**
  - Added new Topology report
  - Added new Type hint topology sort
  - Stubs : added class constant visibility
- **Analysis**
  - New analysis : Report argument whose name clashes with typehint
  - New analysis : Report properties that are insufficiently typed
  - Moved 'Inclusions' to Dump/
  - Added steps to find original and relayed arguments
- **Tokenizer**
  - Fixed paralellisation bug in Load

Version 2.0.1 (Military Star of the North Pole, 2019-10-28)

- **Architecture**
  - Added more return type
  - Centralized reading for ini or json
- **Report**
  - Ambassador: fixed Foreach favorites
  - Ambassador: added sort to number of parameter list
  - Checked unit tests : 3345 / 3377 test pass (99% pass)
- **Analysis**
  - Upgraded xmlwriter to json

Version 2.0.0 (Civil Star of Mystery and Darkness, 2019-10-21)

- **Architecture**
  - Manual file/line fixes
  - More simplifcations in load step
- **Report**
  - Ambassador : fixed performance display
  - Ambassador : report list of shell commands
  - Typehint4all : first report
  - Perfile : fixed sorting

- **Analysis**
  - New analysis : Report possible typehint for bool, int, string, array. WIP
  - Upgraded analysis : common alternatives are extended to switch and elsif
  - Upgraded analysis : xmlreader description includes class constants, properties and methods.
  - Upgraded analysis : callback needs return, is extended to php native functions
  - Checked unit tests : 3345 / 3377 test pass (99% pass)

Version 1.9.9 (Lasting Prosperity Star of True Man, 2019-10-14)

- **Architecture**
  - Documentation review
- **Report**
  - New reports : Stubs, Rector
  - Typehint stats
  - Stubs takes into account use expression
  - Added Concrete5 and Typo3 as vendors
- **Analysis**
  - New analysis : checks on is_a third argument
  - New analysis : Invalid mbstring encodings
  - New analysis : Weird Index in arrays
  - New analysis : Avoid FILTER_SANITIZE_MAGIC_QUOTES
  - New analysis : Don't forget third argument
  - New analysis : Hard to update methods
  - New analysis : Merge two ifthen into one
  - New analysis : Report wrong type with calls
  - New analysis : Check case for namespaces
  - Updated analysis : Undefined interfaces now includes interfaces extensions
  - Updated analysis : Report more wrong types with return type
  - Updated analysis : Register globals also applied to class
  - Updated analysis : Could Use Try covers more new, functions and static calls
  - Updated analysis : Useless Cast also reports (string) array (always Array)
  - Checked unit tests : 3343 / 3366 test pass (99% pass)
- **Tokenizer**
  - Create default values for foreach
  - Load captures empty files, and omit them
  - Create default values also handles ??=

Version 1.9.8 (Giant Gate Star of Dark Essence, 2019-10-07)

- **Architecture**

  - Upgraded dump command to handle multiple -P
  - .yaml configuration handles multiple reports
  - Started journey to strict_types
  - Code cleaning
- **Report**
  - Ambassador : Fixed report of Flexible Docs
  - Ambassador : trimmed delimiters in inventories
  - Inventory : Foreach, with key values
- **Analysis**
  - New analysis : Wrong case for functions
  - New analysis : Parameter Hiding
  - New analysis : Report usage of Traversable
  - Updated analysis : Undeclared properties skips undefined properties
  - Updated analysis : Useless Interface, modernized query
  - Updated analysis : String Holding Variables now skips default, const, sprintf
  - Updated analysis : Binaries are not confused with hex
  - Updated analysis : Extended 'Insufficient typehint' to abstract classes
  - Checked unit tests : 3324 / 3343 test pass (99% pass)
- **Tokenizer**
  - Fixed handling of large powers
  - Added more escaping when storing to SQLITE

Version 1.9.7 (Greedy Wolf Star of Sunlight, 2019-09-30)

- **Architecture**
  - Added support for analysis reporting missing values in a reference list
  - Fixe batch dumping of results
- **Report**
  - Ambassador : new inventory : dereferencing levels
- **Analysis**
  - New analysis : Use PHP Native URL parsing functions
  - New analysis : Maximum dereferencing level
  - New analysis : Use case value in a switch : it was already tested
  - Updated analysis : No class as typehint accepts abstract classes
  - Updated analysis : Create Magic Property reachs out to traits
  - Updated analysis : Security also reports usage of unserialize()
  - Updated analysis : Mistmatched default argument also covers methods
  - Updated analysis : Never used parameter also covers methods

  - Updated analysis : Unused global also cover static variables
  - Updated analysis : Duplicate strings threshold is not 15, not 5.
  - Checked unit tests : 3289 / 3319 test pass (99% pass)
- **Tokenizer**
  - RETURNTYPE, TYPEHINT, and DEFAUT are not always on, with Void atom, or better.
  - DEFAULT value targets end-values, skips ??, ?:, () and =.
  - Exceptions now reports errors in the Query, not where it is thrown

Version 1.9.6 (Star of Birth, 2019-09-23)

- **Architecture**
  - Moved new elements to Complete/
  - Moved new elements to Dump/
  - Initial configuration of project now includes analysis parameters with default
  - Added descriptions to Rulesets
  - New command Config : displays current configuration for reuse and editing
  - Upgraded Doctor : support for docker-php, in-code
- **Report**
  - Ambassador : removed {} on magic property inventory
  - Ambassador : new inventory of network protocols used (udp://, ssh2://...)
- **Analysis**
  - New analysis : avoid mb_string inside loops
  - New analysis : avoid SSLvx and TLSv1.0
  - New analysis : report duplicate literal in the code, with parameter
  - New analysis : warn about null property
  - New coverage : calls to __call and __callStatic
  - Updated coverage : expressions with parenthesis
  - Updated coverage : default values are now targeting the final value in multiple assignations.
  - Updated analysis : Strange Variable name skips Staticdefinition and its default value
  - Updated analysis : Useless instructions are upgrade with pure functions
  - Updated analysis : Extended Closure2string with Arrowfunctions
  - Updated analysis : Extended 'Could be local variable' to traits
  - Updated analysis : Unused Global also covers static variables
  - Checked unit tests : 3279 / 3304 test pass (99% pass)
- **Tokenizer**
  - Updated tokens for PHP 7.4

Version 1.9.5 (Star of Adversity, 2019-09-16)

- **Architecture**

  - Added count property to Analysis node, stepstone for Diff analysis
  - Added support for 'optional' step
  - Added support for 'interfaces' as typehint for remote definitions
  - Removed more true/false values
  - Fixed strtolower with mb_strtolower in Dump
- **Report**
  - Added several PHP error messages
  - Ambassador : added inventory of magic properties
  - Ambassador : added inventory of typehints for methods (WIP)
  - Added support for function/closure/argument arguments
  - Added support for function/closure/argument arguments
- **Analysis**
  - New analysis : No literal value as referenced argument
  - New analysis : use array_slice or array_splice
  - New analysis : Useless typechecks with Typehint
  - New analysis : Report non-implemented interfaces
  - New analysis : Incompatible Signatures with Self (PHP 7.4+)
  - New analysis : Report wrong expectations from interfaces
  - Upgraded analysis : Excluded __construct and __destruct from Magic Methods
  - Upgraded analysis : Concat and Addition : Now also for bitshift
  - Upgraded analysis : Incompatible Signatures with Self (PHP 7.3)
  - Upgraded analysis : Elseif and Sequences are omitted in Level analysis
- **Tokenizer**
  - Upgraded support for magic properties

Version 1.9.4 (Star of Benefit, 2019-09-09)

- **Architecture**
  - Dump avoid storing multiple definition for the same class
  - Added more native return definitions
  - Adding UT for Complete/
  - Dump inventories are being moved to analysis class
  - Moving more Themes => rulesets
- **Report**
  - Ambassador : Fixed several internal links
  - Ambassador : Displays the levels of nesting in the code
  - Ambassador : Upgraded compatibility report with PHP 7.4
  - New report : Stubs

- **Analysis**
  - New analysis : PHP 7.4 New Directives
  - New analysis : Too many dimensions with array
  - New analysis : Check concat and coalesce precedence
  - New analysis : Adopt explode() third argument
  - New analysis : Ternary and useless assignation
  - New analysis : Nested ternary without parenthesis
  - New analysis : Spread operator with arrays
  - New analysis : Max level of indentation
  - New analysis : Use Arrowfunctions
  - Upgraded analysis : Clone with non object handles containers
  - Upgraded analysis : Calling non-static methods statically
  - Upgraded analysis : Unresolved Instanceof
  - Upgraded analysis : Array_merge and variadic, extended to isset
  - Checked unit tests : 3234 / 3259 test pass (99% pass)
- **Tokenizer**
  - Last element of list() is not omitted anymore

Version 1.9.3 (Star of Longevity, 2019-09-02)

- **Architecture**
  - Created new Complete category, with data complement for analysis
  - Refactored constant propagation
  - Made code compatible with PHP 7.4
  - Rename project_themas to project_rulesets
  - Added support of -p with .exakat.yaml
- **Report**
  - Ambassador : reworked presentation for visibility suggestions
- **Analysis**
  - New analysis : report covariance and contravariance for compatibility
  - New analysis : no spread operator for hash values
  - New analysis : self-closing tags are omitted by strip_tags
  - New analysis : report Openssl_random_pseudo_byte second argument usage
  - New analysis : CURLPIPE_HTTP1 is obsolete
  - New analysis : removed PHP 7.4 directives
  - New analysis : do not use ... with array_merge without checks
  - Updated analysis : added crc32c as hash algorithm
  - Removed analysis : Removed Curly Arrays (double take)

  - Checked unit tests : 3219 / 3240 test pass (99% pass)

- **Tokenizer**
  - Extended OVERWRITE to Interfaces
  - Extended support for class_alias()

Version 1.9.2 (Star of Prosperity, 2019-08-26)

- **Architecture**
  - Introduced a new set of analysis : Complete
  - Cleaned code for PHP 7.4 usage
  - Refactored Query to skip impossible Gremlin calls
  - Now using Project for project names
- **Report**
  - New report : classes dependencies (HTML version)
  - New report : files dependencies (HTML and DOT version)
  - Ambassador : datas -> data
- **Analysis**
  - New analysis : {} are deprecated in PHP 7.4
  - New analysis : Don't use ENT_IGNORE
  - New analysis : fn is a PHP 7.4 keyword
  - Updated analysis : Functions/UseConstantAsArguments covers also password_hash()
  - Updated analysis : printf arguments now handles positional formatters
  - Checked unit tests : 3172 / 3199 test pass (99% pass)
- **Tokenizer**
  - Fixed precedence for left associativity

Version 1.9.1 (Star of Life, 2019-08-19)

- **Architecture**
  - Fixed zip as code source
- **Report**
  - Ambassador : Fixed issues list for Favorites
  - Owasp : switched dashboards
- **Analysis**
  - Updated analysis : Loop Calling got one extra check
  - Checked unit tests : 3148 / 3187 test pass (99% pass)

Version 1.9.0 (Ming Wenzhang of Jiayin, 2019-07-29)

- **Architecture**
  - Added missing configuration file for tinkergraph 3.4
  - Upgraded support for running exakat with PHP 7.4

- **Analysis**
  - New analysis : array_key_exists() now report object usage
  - New analysis : report mb_strrpos 4th argument
  - New analysis : Reflection export are deprecated
  - New analysis : Report classes without parents but with 'parent'
  - New analysis : Don't use scalar as arrays
  - New analysis : Report use of PHP 7.4 serialize method
  - Updated analysis : Multiple Identical Keys checks for undefined keys first
  - Updated analysis : Dont be too manual : extended to catch clauses
  - Updated analysis : setcookie detection anchors the keyword at the beginning of the string
  - Updated analysis : Failed Substr comparison now works with constants
  - Updated analysis : Added support for continue 2 and 3
  - Checked unit tests : 3147 / 3186 test pass (99% pass)
- **Tokenizer**
  - Added support for __serialize and __unserialize
  - Added support for numeric literal separator
  - Skip entirely unparsable files

Version 1.8.9 (Meng Feiqing of Jiachen, 2019-07-22)

- **Architecture**
  - Check on graphdb configuration : default to nogremlin
  - Added support for baseline for project and report
  - Moved more doc to ruleset
  - Check on .git folder for update
  - Added -version option for upgrade command
  - Doctor honors .exakat.yml file
- **Analysis**
  - New analysis : Report useless type of checks
  - New analysis : Disconnected classes
  - New analysis : Avoid using mb_detect_encoding()
  - New analysis : Check that source and blind variables are different in foreach
  - New analysis : ~ or ! favorite
  - Updated analysis : Is Zero omits multiplications
  - Updated analysis : Used Private Property is upgraded
  - Updated analysis : Multiple Identical Keys : refactored
  - Updated analysis : Undefined variables now skips extract, include, eval
  - Checked unit tests : 3147 / 3166 test pass (99% pass)

- **Tokenizer**
  - Refactored support for Foreach : each blind variable is in VALUE
  - Upgraded precedence for ! (not)
  - Propagate constants with assignations
  - Fixed link to $this inside heredoc and co
  - Fixed an edgecase where Static method call was confused with Newcall

Version 1.8.8 (Wei Yuqing of Jiawu, 2019-07-15)

- **Architecture**
  - Modernized tinkergraph support
  - When pcntl is available, stubs are produced in a child process
  - Removed duplicated methods
  - Exported sequences to helpers
  - More UT libraries are supported
  - Federated BUSYTIMEOUT in constant
- **Report**
  - Ambassador and all dependend reports were refactored : menu is configurable with Yaml
  - Emissary is the upcoming configurable report.
- **Analysis**
  - New step : Load data from code
  - New analysis : Variables used for setting aside value temporarily
  - New analysis : Use PHP array_* functions, instead of loops
  - Updated analysis : Unused methods now skips methods from PHP native interfaces (Arrayaccess)
  - Updated analysis : No class for typehint is now omitting PHP and extensions classes
  - Updated analysis : Switch to Switch applies to comparisons now
  - Updated analysis : Close namingg was sped up significantly
  - Updated analysis : array_column() suggestion was refined
  - Updated analysis : Htmlentities parameters also support some parenthesis usage
  - Updated analysis : Constant Scalar Expression only target specified expressions
  - Updated analysis : Static Properties skip Virtual properties
  - Checked unit tests : 3131 / 3155 test pass (99% pass)
- **Tokenizer**
  - Refactored support for Exit and Die
  - Added raw support for phpdoc

Version 1.8.7 (Hu Wenchang of Jiashen, 2019-07-08)

- **Architecture**
  - Added bugs fixes up to 7.3.7

  - New factory method for the graph

- **Analysis**
  - New analysis : Backward compatible check on generators (can't return)
  - New analysis : Report wrong return typehint
  - New analysis : Use DateTimeImmutable
  - New concept : Methods that throw errors
  - Updated analysis : Recursive functions disambiguate methods
  - Updated analysis : Refactored property/variable confusion
  - Updated analysis : Could typehint checks on type validations
  - Updated analysis : Variable used once check for abstract methods
  - Updated analysis : Array_merge in loops omits file_put_contents()
  - Updated analysis : Simple Regex covers all special sequences, and unicode sequences
  - Checked unit tests : 3131 / 3142 test pass (99% pass)
- **Tokenizer**
  - Differentiated support for self and static in calls
  - Moved Symfony support to its extension
  - Reworked loading to make it parallels.

Version 1.8.6 (Wei Yuqing of Jiawu, 2019-07-01)

- **Architecture**
  - Added support for Tinkegraph 3.4
  - Extended support for Dev
  - Renamed Themes to Ruleset (WIP)
  - Split several long running queries into smaller chunks
  - Cached files to memory, write them once only
  - Optimized sides queries : omitting them when possible
  - Added count of issues in Analyse node
  - Optimized loading by grouping by inV
  - More coverage for Arrowfunction
- **Report**
  - Dump : collect PHP cyclomatic complexity
- **Analysis**
  - New analysis : Dependant abstract classes
  - New analysis : Don't use Null or Boolean as an array
  - New analysis : Infinite recursion
  - Updated analysis : Raised levels
  - Updated analysis : Method signature must be compatible

  - Updated analysis : Access Private in Trait is OK
  - Updated analysis : Recursive function
  - Checked unit tests : 3099 / 3105 test pass (99% pass)
- **Tokenizer**
  - Upgraded support for 'Modules'

Version 1.8.5 (Zhan Zijiang of Jiaxu, 2019-06-24)

- **Architecture**
  - Fixed several bugs in the online documentation
  - Started removing analysis, replacing with analysis
  - Fixed path in docker PHP usage.
- **Report**
  - Ambassador : Export full INI and YAML config to replicate audit
- **Analysis**
  - New analysis : Unused class constants
  - New analysis : Could Use available Trait
  - New analysis : literal that Could Be Constant
  - Updated analysis : Access Private in Trait is OK
  - Updated analysis : multiple identical argument is extended to closures, methods
  - Updated analysis : ext/rdkafka
  - Updated analysis : No Hardcoded Hash is accelerated
  - Updated analysis : Extended printf() check to constants
  - Updated analysis : Optimized 'redefined method'
  - Updated analysis : Memoize Magic Call
  - Updated analysis : set_locale requires constants
  - Checked unit tests : 3099 / 3105 test pass (99% pass)
- **Tokenizer**
  - Added missing isModified to Foreach keys
  - Class Method Definition handles old style constructor
  - strict_types don't yield a block
  - Added typed values for magic constants
  - Refactored new -> constructor link for Self, Static, parent
  - Added missing arguments count to Newcall

Version 1.8.4 (Wang Wenqing of Jiazi, 2019-06-17)

- **Architecture**
  - Added support for PHP in docker images for compilation tests
  - First prototype for Gremlin in a specific docker image

- **Report**
  - Ambassador : restored original URL
  - Replaced 'Complexity' => 'Time To Fix'
  - Replaced 'Receipt' => Ruleset
- **Analysis**
  - New analysis : regex with arrays
  - New analysis : Complex property names
  - New analysis : array_key_exists speed up
  - New analysis : curl_version forbidden argument
  - New analysis : PHP 7.4 new functions, classes and constants
  - Fixed analysis : Long Variable
  - Updated analysis : printf() format check extended to constants
  - Updated analysis : Written only variables is extended to static and global
  - Updated analysis : refactored 'Make default'
  - Updated analysis : 'Wrong number of arguments' is extended to methods
  - Updated analysis : 'Use coalesce' checks for
  - Updated analysis : Refactored 'Nested ifthen' to have a parameter
  - Updated analysis : Extended 'Class Usage' to return typehint
  - Updated analysis : Sped up 'Used Classes'
  - Checked unit tests : 2993 / 3071 test pass (97% pass)
- **Tokenizer**
  - Upgraded handling of declare with strict_types
  - Support for magic properties across classes and traits
  - Added support for parent with properties
  - Properties are handled with static and normal at the same time
  - Fixed virtualproperties with static keyword (self and parent are ok)
  - Added argument count for 'new A', without parenthesis
  - Restored old break behavior for PHP 5 and older.

Version 1.8.3 (Jade Man of Yang, 2019-06-10)

- **Architecture**
  - Extension docs show version numbers
  - Manual uses internal links
- **Report**
  - New report : SARB
  - Updated report : Ambassador list number of arguments in natural order
- **Analysis**

  - New analysis : from substr() to trim()
  - New analysis : suggest making magic property a concrete one (2 ways)
  - New analysis : no array auto-append
  - Updated analysis : 'Scalar or object property' refactored
  - Updated analysis : 'Multiple identical keys' get a new check on intval, broadened to constants
  - Updated analysis : 'Indirect injection' accelerated
  - Updated analysis : 'Could be class constant' accelerated
  - Updated analysis : 'Never used property' refactored
  - Updated analysis : 'Modern empty' modernized and broadened
  - Updated analysis : 'Useless check' skips isset/empty as they may be useful
  - Updated analysis : 'Identical methoods' skips abstract methods
  - Updated analysis : 'No Count Zero' also uses sizeof(), skips switch()
  - Checked unit tests : 2993 / 3071 test pass (97% pass)

- **Tokenizer**
  - Upgraded local definitions for properties to Load phase
  - Handle static keyword in closures
  - Moved 'Real' to 'Float'
  - Created 'Scalartypehint' atom
  - Fixed intval, boolval for true and false

Version 1.8.2 (Zhao Ziyu of Dingchou, 2019-06-03)

- **Architecture**
  - Refactored 'Update' command, to VCS
  - Collect missing definitions counts
  - Report handles a list of analysis names
- **Analysis**
  - New analysis : No Need To Get_Class
  - New analysis : Report identical inherited methods
  - New analysis : Function returning -1 in case of error
  - Updated analysis : TypeHint must be returned, doesn't apply to abstract methods or interface methods
  - Updated analysis : 'Could Use Interface' also checks for static and visibility
  - Updated analysis : 'Concat empty' skips variables
  - Checked unit tests : 3024 / 3048 test pass (99% pass)
- **Tokenizer**
  - Created 'virtual' properties, for limiting children agglomerations
  - Fixed normalized code for use traits
  - Added DEFAULT to all variable definitions

  - Connect strings to class definitions
  - Handle variable in 'compact', when they are static

Version 1.8.1 (Zhang Wentong of Dinghai, 2019-05-27)

- **Architecture**
  - Fixed Symlink destination
  - Added collecting classes children, traits and interfaces counts
  - Added support for constants and functions in modules
  - Added missing functions in data
- **Report**
  - New report : exakatYaml, which help configuring exakat
  - New report : Yaml
  - New report : Top10
  - Updated report : Json, text and xml get 'fullcode'
- **Analysis**
  - Updated analysis : Should use self is extended to parent classes
  - Updated analysis : Should use prepared statement now skips some SQL queries
  - Checked unit tests : 3024 / 3048 test pass (99% pass)

Version 1.8.0 (Zang Wengong of Dingyou, 2019-05-20)

- **Architecture**
  - Added missing native PHP functions
  - Restored anchor for ignore_dirs[] configuration
  - Removed more MAX_LOOPING usage
- **Report**
  - Ambassador : removed { & @ } artefacts from globals
- **Analysis**
  - New analysis : Function returning -1 in case of error
  - New analysis : Report PHP 7.4 unpacking inside array
  - New analysis : Report PHP 7.4 new functions and fn
  - New analysis : Useless arguments
  - New analysis : Addition and concatenation precedence for PHP 7.4
  - New analysis : report concatenation of empty strings
  - New analysis : casting has precedence over ternary
  - New analysis : report already used traits
  - New analysis : report missing traits in use expression
  - Updated analysis : isset on whole arrays : extended analysis to Phpvariables
  - Updated analysis : SQLITE3 requires single quotes

  - Updated analysis : Dir then slash : extended to constants
  - Updated analysis : Variable Strange Name extended to strange types
  - Updated analysis : Possible interface's analysis is sped up
  - Checked unit tests : 3021 / 3045 test pass (99% pass)
- **Tokenizer**
  - Fixed fullcode of Usetrait
  - Extended method definitions to traits
  - Extended fluent interface detection to parents
  - Fixed dump for visibility change
  - Handle method aliases in use expression (as)
  - Better noDelimiter for double quotes strings

Version 1.7.9 (Shi Shutong of Dingwei, 2019-05-13)

- **Architecture**
  - Upgraded list of functions by extension : openssl, math, hrtime
  - Added global atom to track all globals
  - Rewrote several Dump queries with DSL
  - Added support for Notice in Phpexec
  - Added support for .exakat.ini and .exakat.yaml
  - Added support for arrow functions : fn =>
  - Added support for spread operator in arrays [...[1,2,3]]
- **Report**
  - Inventories : added 'inclusions' and 'global variables'
  - Ambassador : added global variables
- **Analysis**
  - New analysis : support for ext/ffi, uuid
  - Updated analysis : Nested Ternary handles parenthesis
  - Updated analysis : Static loops is extended to references and arrays
  - Updated analysis : Recursive function is extended to Magic methods and Closures
  - Checked unit tests : 3014 / 3019 test pass (99% pass)
- **Tokenizer**
  - Moved 'is_in_ignored_dir' to a property
  - Cleaned getFullnspath() call in Load
  - Fixed latent bug on Function fullnspath
  - Heredoc and Nowdoc are reported as constant if needed
  - Isset() is not read
  - Ignore PHP notices when linting

  - Globals are now centralised across a repository
  - Extended definitions for Virtualproperties
  - Removed double DEFINITION link with new

Version 1.7.8 (Cui Juqing of Dingyi, 2019-05-06)

- **Architecture**
  - renamed test.php to ut.php in tests
  - reorganized destinations folders
  - organized exakat for 'inside code' audit
- **Analysis**
  - New analysis : support for libsvm
  - Updated analysis : Multiple unset() handles unset() at the beginning of the scope
  - Updated analysis : undefined static class now accounts for PHP and module classes
  - Checked unit tests : 2961 / 2995 test pass (99% pass)
- **Tokenizer**
  - Extended class usage to static::class.
  - refactored 2 analysis for speed : double instruction and double assignations
  - fixed recent bug where Project token is twice.

Version 1.7.7 (Sima Qing of Dingmao, 2019-04-29)

- **Architecture**
  - Upgraded to gremlin-php 3.1.1
  - Moved autoload into its own namespace
  - Started extending themes to modules
  - Skip external libraries when unit testing
  - Dump got one more query moved to DSL
  - Fixed build for overwritten methods, extended to magic methods
  - Load tokens by batch (5000+ tokens), not by file.
- **Analysis**
  - New analysis : Security : integer conversion
  - New analysis : implode() with one argument
  - Updated analysis : Invalid Regex handles \ more precisely
  - Updated analysis : delimiter detection was checked for all of them
  - Checked unit tests : 2947 / 2983 test pass (99% pass)
- **Tokenizer**
  - Upgraded Fallback detection for functions

Version 1.7.6 (Jade Maiden of Yin, 2019-04-22)

- **Architecture**

  - Refactored Class definition with return typehint
  - Added configuration for including development extensions.
  - Extended LoadFinal typehint hunting
- **Report**
  - Phpcsfixer : new report
  - Ambassador : report usage of overridden PHP functions
  - Ambassador : new favorite : variable name in catch clause
- **Analysis**
  - New analysis : array_merge and ellipsis should use coalesce
  - New analysis : Report overridden PHP native functions
  - New analysis : Merge all unset() into one
  - Updated analysis : Added missing constant for curl, pgsql, openssl
  - Updated analysis : Variadic are not variable arguments
  - Updated analysis : Useless Reference argument extended to foreach()
  - Updated analysis : Use Constant also covers pi()
  - Updated analysis : Inclusion Wrong Case handles dirname with 2nd argument
  - Updated analysis : Useless Argument : handles some edge cases with arrays
  - Checked unit tests : 2947 / 2975 test pass (99% pass)
- **Tokenizer**
  - Upgraded handling of isRead and isModified attributes
  - Changed variadic argument counts in method declarations
  - Fixed original value in 'Sign'

Version 1.7.5 (Xue King Zhuanlun, 2019-04-15)

- **Architecture**
  - Cleaned unused variables
- **Report**
  - Ambassador : bugfixes report version 7.3, dropped 5.6 and 5.5
- **Analysis**
  - Updated analysis : Already interface : extended to interface parents
  - Updated analysis : Else if to elseif : extended to one-liners
  - Updated analysis : No reference for ternary was extended
  - Updated analysis : Implements is for interface
  - Updated analysis : Refactored Is a Magic Property
  - Updated analysis : Refactored Conditional structures for constants
  - Checked unit tests : 2926 / 2950 test pass (99% pass)
- **Tokenizer**

  - Link properties to magicmethod
  - Deduplicated virtual properties
  - Added isRead and IsModified properties. Omitting the corresponding analysis.

Version 1.7.4 (Lu King Pingdeng, 2019-04-08)

- **Architecture**
  - reports, themes may be specified multiple times
  - 'project' command also work on themes and report from command line
  - Added htmlpurifier in auto-ignored libraries
  - Counting definitions, omitting Virtualproperties
  - Automatically detect identical files
- **Report**
  - Inventories are grouped by values, sorted by count
- **Analysis**
  - Updated analysis : This is for class : extended analysis to self and parent
  - Updated analysis : Undefined Classes
  - Updated analysis : Refactored Defined Parent MP
  - Updated analysis : Redefined PHP function is restricted to global scope
  - Updated analysis : Could Use Alias also covers functions, constants.
  - Updated analysis : Refined SQL detection
  - Fixed step : goToAllParentsTrait missed some of the parent
  - Checked unit tests : 2916 / 2944 test pass (99% pass)
- **Tokenizer**
  - Removed impossible implementations of traits
  - Fixed functioncalls' 'absolute' property
  - Refined parent's definitions
  - Trait also sports virtualproperties
  - Virtualproperties now respect visibilities
  - Distinguish Variables from Staticpropertynames
  - Added missing DEFINITION for Use (namespaces)

Version 1.7.3 (Huang, King Dushi, 2019-04-01)

- **Architecture**
  - New command 'show' that display project creation command
  - Refactored UT detection mechanism
- **Report**
  - Ambassador : report identical files in the code
  - Ambassador : global variable inventory is now grouped by name

- **Analysis**
  - Updated analysis : PPPDeclaration style : handles Virtualproperties
  - Updated analysis : Closure2string : extended analysis
  - Updated analysis : Non-Ascii variable skips { }, & and @
  - Updated analysis : Could Be Static exclude abstract methods
  - Updated analysis : MismatchedTypehint : handles methodcalls and class hierarchy
  - Updated analysis : Could Use Try : refined analysis to avoid literals
  - Updated analysis : Hidden use, handles Virtualproperty
  - Updated analysis : Classes, wrong case, handles FQN
  - Checked unit tests : 2846 / 2926 test pass (97% pass)
- **Tokenizer**
  - Moved creation of Virtualproperty early, to catch more situations
  - Virtualproperty mimic Propertydefinition
  - Added extra check when roaming the classes tree
  - Handles Sign constant values correctly

Version 1.7.2 (Dong King Taishan, 2019-03-25)

- **Architecture**
  - Restored the external library checker
  - Added support for extension's CIT (Symfony, Drupal)
- **Report**
  - Ambassador : added Suggestions theme to docs.
  - Perfile : New report, text, per file
- **Analysis**
  - New analysis : Report potential 'unsupported operand type'
  - New analysis : Check for existence with __call() and __callstatic
  - Updated analysis : Wrong number of arguments (methods) upgraded
  - Updated analysis : Could Be Static ignores empty methods, constants methods
  - Updated analysis : Added Variable to possibly useless expression
  - Updated analysis : Constant names are detected based on available noDelimiter
  - Updated analysis : Abstract classes may have no abstract methods
  - Checked unit tests : 2889 / 2912 test pass (99% pass)
- **Tokenizer**
  - Added link between __clone and clone
  - Now handling functions and constants when ignored
  - Fixed dynamic constants in collector

Version 1.7.1 (Bi King Biancheng, 2019-03-18)

- **Report**
  - Ambassador : report lines that concentrate lots of issues
- **Analysis**
  - Extended GoToAllImplements to extended interfaces
  - Updated analysis : NoScream usage, with authorized functioncall list like fopen
  - Updated analysis : HiddenUse with support for virtual properties
  - Checked unit tests : 2867 / 2900 test pass (99% pass)
- **Tokenizer**
  - Added support for 'Virtualproperties'
  - Harmonized file escaping feature

Version 1.7.0 (Bao King Yama, 2019-03-11)

- **Architecture**
  - Added auto-documenting 'ignored' cit to weed out obvious false positive
- **Report**
  - Made Diplomat the default report
  - Added History report : it stores metrics from audit to audit
- **Analysis**
  - New analysis : Identify self transforming variables ($x = foo($x))
  - New analysis : Report unclonable variables
  - Updated analysis : Undefined Classes, Interfaces and Trait now omit 'ignored' cit from folders
  - Updated analysis : Inconsistent usage is refactored for properties
  - Updated analysis : Useless expression, with clone new x
  - Updated analysis : Only Variable For Reference accepts $this, $_GET
  - Updated analysis : Lost References was modernized
  - Checked unit tests : 2854 / 2884 test pass (99% pass)
- **Tokenizer**
  - Refactored support for Staticmethod (in a trait's use)
  - Added definitions for trait's use

Version 1.6.9 (Lu King Wuguan, 2019-03-04)

- **Architecture**
  - Optimized Dump when navigating the links to the File Atom
  - Refactored LoadFinal into separate classes
  - Upgraded to Tinkergraph 3.3.5
  - Added options to cleandb to stop and start gremlin from exakat
  - Skip the task if no analysis has to run
- **Analysis**

  - New analysis : Report inconsistent usage of properties or variables
  - New analysis : Typehinted return must return
  - Updated analysis : Variables used once handles closure (use) correctly
  - Updated analysis : Is Zero was refactored partially (WIP)
  - Updated analysis : Bad Typehint relay got a fix
  - Updated analysis : Function Subscripting is only suggested for one usage
  - Updated analysis : Lost References was modernized
  - Checked unit tests : 2854 / 2881 test pass (99% pass)
- **Tokenizer**
  - Added definition for injected properties
  - Fixed sack() for subqueries
  - $this is not a classic variable
  - Removed double DEFINITION links
  - Fixed edge case with define() at the end of a script

Version 1.6.8 (Yu King Songdi, 2019-02-25)

- **Architecture**
  - Added support for PHP 8.0
  - Fixed Constant FNP
  - Advance progressbar when ignoring files
- **Report**
  - Ambassador : report usage of factories
  - Collect stats about Foreach usage
- **Analysis**
  - New analysis : Report violation of law of Demeter
  - New analysis : Report removed constants and functions in PHP 8.0
  - Updated analysis : Refactored Nullable Typehint
  - Checked unit tests : 2851 / 2872 test pass (99% pass)
- **Tokenizer**
  - Fixed edge case for Logical with strings
  - Reduced max level of looping in GoToAllParents
  - Distinguish $$ and ${$

Version 1.6.7 (Li King Chujiang, 2019-02-18)

- **Architecture**
  - Documentation covers more PHP functions
  - Added some missing PHP functions
  - Fixed destination folder for extensions

- **Report**
  - Ambassador : limited size of default values in visibility report.
  - Ambassador : reporting class depth
  - Ambassador : reporting dynamically created constants
  - Diplomat : leanner, meaner version of Ambassador
  - New category : Top 10 classic mistakes
- **Analysis**
  - New analysis : Report when relayed typehint are not the sames
  - Updated analysis : Regex now handles local variables and constants
  - Updated analysis : Variables Used Once now covers closures and use
  - Checked unit tests : 2846 / 2867 test pass (99% pass)
- **Tokenizer**
  - Defineconstant may be constant
  - Fixed handling of Nullable for typehint
  - Started preparing for Gremlin 3.4.0 : WIP

Version 1.6.6 (Jiang King Qinguang , 2019-02-11)

- **Architecture**
  - Removed FetchContext() from DSL
  - Added options to follow constants from atomIs.
- **Report**
  - Now dumps magic methods
- **Analysis**
  - New analysis : Report insufficient interfaces in typehint
  - Updated analysis : Class constant now ignore empty classes
  - Checked unit tests : 2837 / 2858 test pass (99% pass)
- **Tokenizer**
  - Moved 'Define' to its own atom
  - Upgraded Logical to hanlde Strings as PHP
  - Fixed T_POWER => T_POW
  - Refactored calculation for globalpath
  - Fixed edgecase with endswitch;

Version 1.6.5 (Mahagate, 2019-02-04)

- **Architecture**
  - Added CVS as an external service
  - Graph GSNeo4j export variable for shell access. putenv is not sufficient
  - Dump : report class name, not its code

  - Extended listAllThemes to extensions
  - Fixed bug in extension loader with phar
- **Report**
  - Ambassador : restored file dependencies tree
  - Ambassador : fixed altered directive filename
  - Ambassador : added direct link to docs
- **Analysis**
  - New analysis : arrays that are initialized with strings
  - New analysis : Avoid Lone variables as conditions
  - New analysis : Added support for weakref and pcov
  - Updated analysis : extended regex to arrays in preg_* calls
  - Updated analysis : Implicit globals now also marks the variable in global space
  - Updated analysis : Add Zero, Multiply by One also cover 2 * $x = 1;
  - Updated analysis : Could Use Interface now takes into account PHP interfaces, and classes first level.
  - Updated analysis : Relay Functions now omits calls to parent's __construct and __destruct
  - Checked unit tests : 2830 / 2852 test pass (99% pass)

Version 1.6.4 (Parasamgate, 2019-01-28)

- **Architecture**
  - Added support for CVS as a VCS
  - Upgraded support for tar as a VCS
  - Added support modification counts by files
  - Added first tracking for closures
  - Upgraded Tinkergraph driver
- **Report**
  - Added Atoms in the documentations
  - Extra protection for Class Changes
- **Analysis**
  - Updated analysis : Use-arguments are now counted as arguments
  - Updated analysis : Max Argument check was refactored
  - Updated analysis : IsModified now takes into account extensions
  - Updated analysis : Should Use This now exclude empty methods
  - Updated analysis : undefined classes now support PHP 7.4 typed properties
  - Updated analysis : added missing scalar PHP types
  - Updated analysis : uncaught exceptions now cover parents
  - Updated analysis : refactored incompatibility checks for methods
  - Checked unit tests : 2824 / 2841 test pass (99% pass)

- **Tokenizer**
  - Refactored alternative ending, removed extra VOID
  - Upgraded contexts and their nesting
  - Added extra checks on variables names
  - Added support for ??= (PHP 7.4)

Version 1.6.3 (Paragate, 2019-01-21)

- **Architecture**
  - Better presentation for exakat extensions
  - Added build.xml for Jenkins
  - Fixed copyright years
- **Report**
  - Ambassador : fixed class name for Phpcompilation
- **Analysis**
  - New analysis : assign and compare at the same time
  - Updated analysis : uncaught exceptions now cover parents
  - Updated analysis : strpos too much is extended to strrpos and strripos
  - Updated analysis : Refactored Indirect injections for more refined reports
  - Updated analysis : Empty Block doesn't omit Ifthen anymore
  - Updated analysis : Implemented methods are public mistook interface methods
  - Updated analysis : Object Reference omits arguments that are wholly assigned
  - Checked unit tests : 2808 / 2826 test pass (99% pass)
- **Tokenizer**
  - Added support for PHP 7.4 typed properties (needs PHP 7.4-dev)

Version 1.6.2 (Silver Headed Gate, 2019-01-14)

- **Architecture**
  - Fixed infinite loop when an option missed a value
  - Produce phpversion in config.ini, but leave it commented
- **Report**
  - Ambassador : colored syntax for visibility report
  - Ambassador : inventory reports now display number of usages
- **Analysis**
  - Updated analysis : Added support for PHP 7.2.14
  - Updated analysis : Avoid Using Class handles
  - Updated analysis : Unused Functions works with multiple identical functions
  - Checked unit tests : 2795 / 2817 test pass (99% pass)
- **Tokenizer**

  - Fixed bug that mixed T_OR and T_XOR
  - Fixed bug that missed intval for Power
  - Handles multiple definitions of functions
  - Removed one Void too many with closing tag

Version 1.6.1 (Golden Light Gate, 2019-01-07)

- **Architecture**
  - Upgraded documentation for Extensions
  - Upgraded processing of files, specially with special chars
  - Project stops when no token are found
  - Storing hash for each files. RFU.
- **Report**
  - Ambassador : added support for class constant's changes
  - Ambassador : added classSize report
  - Ambassador : 'New issues' now takes line difference into account
  - Themes are better dumped
- **Analysis**
  - New analysis : array_key_exists() is faster in PHP 7.4
  - New analysis : partial report from preg_match()
  - Updated analysis : Avoid Using Class handles
  - Updated analysis : Class Usage uses class_alias()
  - Updated analysis : Empty traits
  - Updated analysis : Unused arguments now skips __set()
  - Updated analysis : Path strings
  - Updated analysis : Missing include handles more concatenations
  - Checked unit tests : 2792 / 2812 test pass (99% pass)
- **Tokenizer**
  - Fixed precedence for identical operators
  - Fixed bug with ?> inside switch

Version 1.6.0 (VirupakSa, 2018-12-31)

- **Architecture**
  - VCS are not tested when they are not used
- **Analysis**
  - Updated analysis : Php Reserved names ignores variable variables
  - Updated analysis : Array not using a constant, with Heredoc
  - Updated analysis : Long arguments
  - Updated analysis : Empty With Expression ignores simple assignations

  - Refactored analysis : Callback needs returns
  - Refactored analysis : No Return used
  - Checked unit tests : 2780 / 2805 test pass (99% pass)
- **Tokenizer**
  - Fixed regression with Yield and =>
  - Fixed edge case "$a[-0x00]"

Version 1.5.9 (Dhrtarastra, 2018-12-24)

- **Architecture**
  - Use PHP in project config for default PHP version
  - cleandb uses -p
  - Moved projects/.exakat to projects/<-p>/.exakat folders
  - Using $config and not more hardcoded tinkergraph
  - Extra check on doctor
- **Report**
  - Ambassador : extra check for 'previous' report
- **Analysis**
  - Upgraded analysis : Empty With Expression skip a few false positive
  - Checked unit tests : 2770 / 2795 test pass (99% pass)
- **Tokenizer**
  - Fixed edgecase for methods named 'class'
  - Fixed class name in Project

Version 1.5.8 (Virudhaka, 2018-12-17)

- **Architecture**
  - Handles themas provided by extensions
  - Added busyTimeout for dump.sqlite
  - Reduced size of thema tables
  - Docs handle parameter dynamically
  - Added 'update' for extensions
- **Report**
  - Ambassador : added a 'Path' inventory, with file paths
- **Analysis**
  - New analysis : Closures that are identical
  - Upgraded analysis : Url and SQL detection, case sensitivity
  - Upgraded analysis : Could Use array_fill_keys
  - Upgraded analysis : Undefined functions doesn't miss functions inside classes, handles interfaces
  - Upgraded analysis : Empty Functions better handles return;

  - Upgraded analysis : Long Argument may be configured
  - Upgraded analysis : Fixed bug with empty include path
  - Checked unit tests : 2770 / 2795 test pass (99% pass)
- **Tokenizer**
  - Added FNP to strings
  - First link between method and definition with typehint
  - Support for class_alias
  - Fixed edge case with use ?>
  - Fixed variable in string behavior for $this and $php variables

Version 1.5.7 (Vaisravana, 2018-12-10)

- **Architecture**
  - Extended Dump to support aliased methods
  - Support for SQLITE in extensions
  - Moved each framework to extensions
  - Added Laravel extension
- **Documentation**
  - First version for the Extension chapter
  - Fixed mysterious ‘ in the docs
- **Report**
  - Ambassador : added a ‘New issues’ section, with new analysis
  - Ambassador : added trait matrix
  - Ambassador : fixed an infinite loop when trait include themselves in cycles
  - Added more message count to several reports
- **Analysis**
  - New analysis : method could be static
  - New analysis : multiple inclusion of traits
  - New analysis : avoid self using traits
  - New analysis : ext/wasm and ext/async
  - Upgraded analysis : No Hardcoded Hash, skip hexadecimal numbers
  - Upgraded analysis : Defined properties extends to traits
  - Upgraded analysis : PSS outside a class, when PSS are in strings
  - Upgraded analysis : Access private works with methods (not just static)
  - Checked unit tests : 2772 / 2785 test pass (99% pass)
- **Tokenizer**
  - Fixed bug in Dump, when nothing to clean
  - Fixed edge bug on Callable detection

  - Extended support for self, static and parent, in typehint and new
  - Fixed precedence of yield and yield from
  - Fixed handling of throw at the end of a script
  - Added support to solve conflict on traits

Version 1.5.6 (Jingang, 2018-12-03)

- **Architecture**
  - Moved all framework to extensions. WIP.
  - Code cleaning
  - Refactored the analysis dependency sorting
  - Now display progress bar for files
  - Fixed configuration for directories and files
- **Report**
  - Fixed FileDependecy and DependencyWheel, to actually count messages
- **Analysis**
  - Added a lot more new method descriptions for PHP native classes
  - New analysis : suggestion simplification for !isset($a) || !isset($a[1])
  - New analysis : Useless Trait alias
  - New analysis : report usage of ext/sdl
  - Upgraded analysis : Refactored IsZero, to handle assignations and parenthesis
  - Upgraded analysis : pack format is better checked
  - Checked unit tests : 2759 / 2771 test pass (99% pass)
- **Tokenizer**
  - Fixed a missing fullnspath for origin in Use for Traits
  - Handles simple aliases for traits methods
  - Fixed mishandling of variables inside strings
  - Fixed support of negative numbers inside strings
  - Fixed bug with yield inside an array
  - Fixed strange case with define and integers as constant names

Version 1.5.5 (Ratnadhvaja, 2018-11-25)

- **Architecture**
  - Initial version of Exakat extensions
  - Moved processing of 2-tokens files to Load
  - Speed up CSV creations
  - Upgrades are read from https, no http
  - Moved loading's sqlite to memory for speed gain
  - Doctor now auto-create test folder

- **Report**
  - New report : Php city. See your PHP code as a city
  - Ambassador : Appinfo() now reports keywords used as method or property
  - Fixed reported names of properties
- **Analysis**
  - New analysis : checks some HTTP headers for security
  - New analysis : Use _file() functions, not file_get_contents()
  - New analysis : Optimize looks for fgetcsv()
  - Upgraded analysis : Several refactored analysis
  - Checked unit tests : 3083 / 3096 test pass (99% pass)
- **Tokenizer**
  - Fixed encoding error in loading, for clone types.

Version 1.5.4 (Mahakasyapa, 2018-11-19)

- **Architecture**
  - Added error message for memory limit
  - Added GC to Project action
  - Migrated Melis to extension
  - Dumping data is now done en masse
  - Analysers now handle side-queries
  - Clear message in case of memory limit
  - Doctor doesn't stop at missing helpers
  - VCS leak less errors
  - Added support for 7z
  - Extended validation for themas
  - Restored Tinkergraph driver
  - Upgrade logs with extra reports
- **Analysis**
  - New analysis : Report problems with class constant visibilities
  - New analysis : Avoid self, parent and static in interfaces
  - Upgraded analysis : Variable reuse now skips empty arrays
  - Checked unit tests : 3077 / 3090 test pass (99% pass)
- **Tokenizer**
  - Fixed bug where variable was mistaken for a string inside strings

Version 1.5.3 (Ananda, 2018-11-12)

- **Architecture**
  - Extended results to methods, traits

  - Added support for PHP 7.2.12
  - 'master' is not used anymore as default branch
  - Fixed creation of initial config/exakat.ini
  - Fixed handling badly written exakat.ini or PHP binary paths
- **Report**
  - Ambassador : report classes that could be final or abstract
- **Analysis**
  - New analysis : Property Used Once : now includes redefined functions
  - New analysis : iterator_to_array() should use yield with keys or array_merge()
  - New analysis : Don't loop on yield : use yield from
  - Upgraded analysis : Dependant trait now include parent-traits
  - Checked unit tests : 3080 / 3093 test pass (99% pass)
- **Tokenizer**
  - Changed handling of variable that are both global AND local
  - Disambiguated variables and properties
  - Extended OVERWRITE to constants and methods

Version 1.5.2 (Master Puti, 2018-11-05)

- **Report**
  - Fixed storage of themes in dump.sqlite
  - Ambassador : report nothing when there are no trait, interface or class in the tree.
- **Analysis**
  - New analysis : idn_to_ascii() will get new default
  - New analysis : support for decimal extension
  - New analysis : support for psr extension
  - Upgraded analysis : Extended support to PHP native exceptions
  - Upgraded analysis : Could use typecast now handles intval() second param
  - Upgraded analysis : Variable strange names avoids properties
  - Checked unit tests : 3058 / 3085 test pass (99% pass)
- **Tokenizer**
  - Upgraded support for arrays inside strings (string/constant distinction)
  - Added DEFINITION for constant() and defined()
  - Fixed value of line for some placeholder definition

Version 1.5.1 (Eighteen Arhats, 2018-10-29)

- **Analysis**
  - New analysis : could use basename() second args
  - Upgraded analysis : Variables strange names do not report ...

    - Checked unit tests : 3061 / 3079 test pass (99% pass)
- **Tokenizer**
    - Moved TRAILING as a property
    - Moved NULLABLE as a property
    - Sync ALIAS with AS
    - Fixed link between Use expression when using an alias

Version 1.5.0 (Pilanpo Bodhisattva, 2018-10-22)

- **Architecture**
    - Fixed ” in the examples of the manual
    - Upgraded stability with new history testing
- **Report**
    - Ambassador : now report interface and trait hierarchy
    - Ambassador : new format inventory for pack and printf
    - Dump : Fixed list of traits
- **Analysis**
    - New analysis : Could Use Try, for native calls that may produce an exception
    - New analysis : idn_to_ascii() will get new default
    - Upgraded analysis : Undefined variables exclude $this
    - Upgraded analysis : Variables used once avoid properties
    - Upgraded analysis : ext/json : JsonException
    - Upgraded analysis : added new PHP 7.3 constants (curl, pgsql, mbstring, standard)
    - Upgraded analysis : scalar or object property now ignore NULL as default
    - Refactored analysis : UsedProtectedMethod
    - Checked unit tests : 3059 / 3071 test pass (99% pass)
- **Tokenizer**
    - Handles NaN and INF when the literals reach them
    - Static constant may be variable if object is variable
    - Removed superfluous linking for static calls.

Version 1.4.9 (Lingji Bodhisattva, 2018-10-15)

- **Architecture**
    - Extended documentation with phpVersion, time to fix and severity
    - Upgraded bufixes to PHP 7.2.11
    - Added more tests on arguments in the DSL
    - Removed double definitions for class constants
    - Initial support for extension folder
- **Report**

  - Collect the number of local variables, per method

- **Analysis**
  - New analysis : report accessing properties the wrong way
  - New analysis : suggest named patterns
  - New analysis : check Pack() arguments
  - New analysis : Return in generators, for PHP 7.0 +
  - New analysis : Repeated interfaces
  - New analysis : Static properties shouldn't use references until PHP 7.3
  - New analysis : Don't read and write in the same expression
  - Upgraded analysis : is interface methods, extended to magic methods
  - Upgraded analysis : empty regex
  - Upgraded analysis : never used properties
  - Upgraded analysis : logical operators in letters
  - Upgraded analysis : could use interface, extended with PHP native interfaces
  - Upgraded analysis : Is Zero, better handling of mixed expressions
  - Refactored analysis : Empty functions
  - Refactored analysis : Used Private Methods
  - Checked unit tests : 3036 / 3055 test pass (99% pass)
- **Tokenizer**
  - Added DEFINITION between new and __construct
  - Added support for className::class()
  - Added better support for dynamic method calls
  - Added better support for dynamic property calls
  - Removed some usage of TokenIs

Version 1.4.8 (Ksitigarbha, 2018-10-08)

- **Architecture**
  - Adding more validation at DSL step level : stricter check on args, speed gain
  - Cleaning more analysis from MAX_LOOPING variable
  - Better protection for file names
  - Removed static properties from DSL
- **Analysis**
  - New analysis : Don't use __clone before PHP 7.0
  - New analysis : Watch out for filter_input as a data source
  - Upgraded analysis : Method Used Below refactored for speed
  - Upgraded analysis : Undefined class constants now takes into account interfaces
  - Removed anaysis : Relaxed Heredoc was double with Flexible Heredoc

  - Checked unit tests : 3016 / 3033 test pass (99% pass)

- **Tokenizer**
  - Build links between methodcall and method in a class
  - Added links between method and its overwritten version in child
  - Fixed fallback for functions
  - Fixed linked between traits and their definition
  - Removed variable definition for Parametername
  - Simplified double usage between return and pushExpression()

Version 1.4.7 (Maitreya, 2018-10-01)

- **Architecture**
  - Added 'Suggestions' section to documentation, for many rules
  - WIP : removing usage of MAX_LOOPING in analysis
  - Added a lot of new external services
  - Added documentation for creating a new analysis
- **Analysis**
  - Upgraded analysis : No interface was dropped in PHP 7.2
  - Upgraded analysis : IsAMagicProperty extended to parents
  - Removed anaysis : Relaxed Heredoc was double with Flexible Heredoc
  - Checked unit tests : 3017 / 3029 test pass (99% pass)
- **Tokenizer**
  - Linking variable in closure's use to its local variable
  - Removed some unused atoms from GraphElements

Version 1.4.6 (Dipankara, 2018-09-24)

- **Architecture**
  - Various code refactorisations
  - Migration to PHPUnit 7.3.5
  - Fixed filenames case
  - Better handling of VCS
  - More validations for project names
  - More docs
- **Report**
  - Ambassador/Weekly : fixed ' in analyser titles
- **Analysis**
  - Upgraded analysis : Fopen mode accepts 'r+b'
  - Upgraded analysis : Unused Traits
  - Upgraded analysis : Undefined Variables

  - Checked unit tests : 3020 / 3033 test pass (99% pass)

- **Tokenizer**
  - New analysis : report literal used with reference
  - Added support for boolval to Keyvalue
  - Fixed support for boolval to Arraylist
  - Added DEFINITION to static methods
  - Added Variabledefinition for local variables
  - Fixed bug in Not

Version 1.4.5 (Guanyin Bodhisattva, 2018-09-17)

- **Architecture**
  - Removed times() for until() in Dumps
- **Report**
  - Manual : added folders tree
- **Analysis**
  - New analysis : Add Default To Parameter
  - Upgraded analysis : Avoid reporting PHP function as classes
  - Upgraded analysis : More empty Functions than just foo() { }
  - Upgraded analysis : Wrong Number of argument now takes into account variadic
  - Upgraded analysis : Should Use Constant now encompasses () and ?: structures
  - Upgraded analysis : This Is Not An Array now takes ArrayObject/SimpleXmlElement into account
  - Checked unit tests : 3009 / 3020 test pass (99% pass)
- **Tokenizer**
  - Fixed 'constant' status with Arrayliteral
  - Fixed bug where strings are build close to the end of the script

Version 1.4.4 (White Dragon Horse, 2018-09-10)

- **Architecture**
  - Doctor reports the set of tokens used
  - Lots of docs checks
- **Report**
  - Ambassador / Phpconfiguration : report disable_functions and disable_classes
  - Finished Weekly report
- **Analysis**
  - New analysis : report ext/seaslog
  - Upgraded analysis : Incompatible signatures
  - Fixed DSL : analysisIs
  - Checked unit tests : 3000 / 3010 test pass (99% pass)

- **Tokenizer**
  - Closure are now processed with runplugin
  - Removed depencencies to usedClasses
  - Fixed detections of Closure at the end of a script

Version 1.4.3 (Sha Wujing, 2018-09-03)

- **Architecture**
  - No error if missing svn
  - Extended 'First' thema
  - Now reporting PHP native CIT, constants and functions
- **Report**
  - Ambassador : php.ini suggestions includes disable_functions
- **Analysis**
  - New analysis : report typecasting for json_decode
  - New analysis : report classes that could be final
  - New analysis : simplify closure into callback
  - New analysis : report inconsistent elseif conditions
  - Upgraded analysis : Reduced false positive on Type/Default mismatch
  - Upgraded analysis : Drop Else After Return uses elsif
  - Upgraded analysis : Unused Private Property (rare)
  - Checked unit tests : 2990 / 3004 test pass (99% pass)
- **Tokenizer**
  - Removed extra Void after function definitions
  - Fixed fullnspath with define()

Version 1.4.2 (Zhu Bajie, 2018-08-27)

- **Architecture**
  - Fixed leftover bugs in the new DSL language
  - Adopter Query in LoadFinal (first test)
  - Extended support for clone type 1
- **Report**
  - New Report : Weekly report
- **Analysis**
  - New analysis : report forgotten conflict in traits
  - New analysis : undefined insteadof
  - New analysis : undefined variable
  - New analysis : report classes that must call parent::__construct
  - Upgraded analysis : Inexistant Compact variable

  - Upgraded analysis : Test class was refactored
  - Checked unit tests : 2975 / 2989 test pass (99% pass)

- **Tokenizer**
  - New atom : Staticmethod, for Insteadof (replacing 'Staticconstant')
  - Added DEFINITION link for array('class', 'method') structure

Version 1.4.1 (Tang Sanzang, 2018-08-20)

- **Architecture**
  - Spined off Query for Gremlin, with Exakat DSL.
  - Centralized 'methods' property in Analysis class
  - Extended MAX_LOOPING usage
- **Analysis**
  - Added new thema : Class Review
  - Upgraded analysis : Defined Parent MP (less queries)
  - Upgraded analysis : Less false positives
  - Added support for PHP 7.2.9
  - Checked unit tests : 2965 / 2980 test pass (99% pass).
- **Tokenizer**
  - Fixed Edge case with Ternary and Boolean
  - Added Staticpropertyname to distinguish from variables
  - Added support for remote definitions to methods
  - Removed global path for CIT (no fallback)

Version 1.4.0 (Sun Wu Kong, 2018-08-13)

- **Architecture**
  - Chunked result inserts for Dump
  - More support for PHP 7.4
- **Report**
  - Ambassador : added new Appinfo for relaxed Heredoc, trailing comma…
- **Analysis**
  - New analysis : class can be abstract
  - New analysis : trailing comma
  - New analysis : relaxed heredoc
  - New analysis : removed functions in PHP 7.3
  - New analysis : continue versus break
  - Upgraded analysis : Hardcoded passwords is extended to objects
  - Checked unit tests : 2964 / 2979 test pass (99% pass).
- **Tokenizer**

  - Measure definitions stats for classes.
  - Added support for relaxed heredoc
  - Added support for closure as a return value
  - Refactored support for Ternary and Labels

Version 1.3.9 (Du Ruhui, 2018-08-06)

- **Architecture**
  - Added support for PHP 7.4
  - 'Copy' won't update anymore
- **Report**
  - Ambassador : fixed repeated 'compatibility' menu entry
- **Analysis**
  - New analysis : avoid __CLASS__ and get_called_class().
  - New analysis : prepare for (real) deprecation
  - New analysis : const / define preference
  - New analysis : define case sensitivity preference
  - New analysis : avoid defining assert() in namespaces
  - Removed analysis : Variables/Arguments
  - Checked unit tests : 2957 / 2971 test pass (99% pass).
- **Tokenizer**
  - Removed Noscream - AT atom
  - Added definition for class constants
  - Fixed bug : can't apply ~ to false
  - Extended DEFINITION support to closure's use and references

Version 1.3.8 (Fang Xuanling, 2018-07-30)

- **Architecture**
  - 'Copy' won't update code anymore.
- **Analysis**
  - Upgraded analysis : 'should use operator' only applies to constant chr() call
  - Upgraded analysis : Useless Instructions is faster
  - Checked unit tests : 2948 / 2962 test pass (99% pass).
- **Tokenizer**
  - Added support for variable definitions in methods

Version 1.3.7 (unnamed demon, 2018-07-16)

- **Architecture**
  - Fixed handling of multiple updates
- **Report**

  - More documentations

- **Analysis**
  - New analysis : report usage of callback to process array
  - New analysis : report usage of case insensitive constants
  - Upgraded analysis : Hardcoded passwords is extended to objects
  - Upgraded analysis : Go To Key Directly handles comparisons
  - Added support for PHP 7.0.20
  - Checked unit tests : 2948 / 2962 test pass (99% pass).

Version 1.3.6 (Zhang Gongjin, 2018-07-16)

- **Architecture**
  - Added support for Rar archives
  - Removed call to gremlin server at 'status' time
- **Analysis**
  - New analysis : support for msgpack extension
  - New analysis : support for lzf extension
  - Upgraded analysis : added missing function names in several extensions
  - Checked unit tests : 2941 / 2955 test pass (99% pass).

Version 1.3.5 (Gao Shilian, 2018-07-09)

- **Architecture**
  - Removed 4 unused exceptions
  - Extracted Query from Analysis
- **Report**
  - Reports : centralized all doc reading
  - Reports : doc reading now parses sections (avoid overlap)
  - Ambassador : Added exakat version and build to dashboard.
  - Ambassador : Added Class Tree (All class hierarchies)
- **Analysis**
  - Fixed bug with 'last' and '2last'
  - New analysis : Report undefined::class
  - New analysis : Report returned assignations as useless
  - New analysis : Split scalar typehint by versions
  - Upgraded analysis : Extended Reuse Variable to instantiations
  - Upgraded analysis : Masking parenthesis are only for referenced arguments
  - Upgraded analysis : Wrong case doesn't apply to parent/static/self
  - Upgraded analysis : Locally Unused Properties are extended to traits
  - Upgraded analysis : Should Preprocess is extended to concatenations

  - Upgraded analysis : Array_key_fill exclude variables by default
  - Upgraded analysis : Ambiguous static reports the whole property definition
  - Checked unit tests : 2919 / 2944 test pass (99% pass).
- **Tokenizer**
  - Added missing constants
  - Fixed support for goto true;
  - Fixed edge case for nested ternaries and boolean
  - Moved Goto and Label to Name Atom

Version 1.3.4 (Cheng Yaojin, 2018-07-02)

- **Architecture**
  - Added check when unarchiving tar.gz and tar.bz
  - Added check for neo4j installation, (error grabing)
  - Moved Upgrade to tmp folder
- **Analysis**
  - Parameters are actually defined in the class
  - New analysis : ambiguous visibilities of properties
  - New analysis : report usage of PHP 7.1+ hash algorithm
  - New analysis : csprng (random_bytes and random_int)
  - New analysis : ext/libeio
  - New analysis : report incompatible signatures for methods
  - Upgraded analysis : Unused Private Methods handles fluent interfaces
  - Upgraded analysis : Defined Parent keyword
  - Upgraded analysis : Recursion
  - Refactored codeIs/codeIsNot
  - Checked unit tests : 2908 / 2923 test pass (99% pass).
- **Tokenizer**
  - Added support for 'parent' definitions
  - Fixed element counts in concatenation
  - Fixed operator priority in Strval
  - Upgraded handling of undefined constants to string

Version 1.3.3 (Ma Sanbao, 2018-06-25)

- **Architecture**
  - Better handling of fallback to global for functions
  - Weekly code clean
  - Refactored several analysis for speed
- **Report**

  - Ambassador : fixed regression in the dashboard
  - Fixed edge case with properties
- **Analysis**
  - New analysis : closure that can be static
  - Upgraded analysis : empty function doesn't count static or global
  - Upgraded analysis : reported globals include $GLOBALS also
  - Checked unit tests : 2881 / 2911 test pass (98% pass).
- **Tokenizer**
  - Moved collection of functioncall to LoadFinal
  - Added collection of interfaces and newcall
  - Moved Declare to its own token
  - Moved Property definitions to its own token

Version 1.3.2 (Duan Zhixian, coming up)

- **Architecture**
  - Reading stats from store, not graph.
  - Git now fails silently if login is requested at clone / pull
- **Report**
  - New analysis : == or === favorites
  - New analysis : > or < favorites
  - Upgraded analysis : written only variables is now faster
  - Upgraded analysis : PHP reserved words has now 2 parameters
  - Removed analysis : Type/Integer, Real, Closures.
  - Checked unit tests : 2901 / 2914 test pass (99% pass).
- **Tokenizer**
  - Static, PPP, Final and Abstract are now properties
  - Fixed regex in several rules
  - Added support for code clone detection (WIP)

Version 1.3.1 (Liu Hongji, 2018-06-03)

- **Architecture**
  - Cleaned code of unused classes and ;
  - Fixed connexion script to the database
  - Fixed check of php.log folder
- **Report**
  - Ambassador : display correct compilation state
- **Analysis**
  - Upgraded analysis : used constant is also applied to defined()

  - Upgraded analysis : used protected methods is case insensitive
  - Upgraded analysis : Empty class omits extended classes
  - Upgraded analysis : More sequences to SimplePreg
  - Upgraded analysis : Throwable is not 'unthrown' anymore
  - Removed analysis : Static CPM
  - Checked unit tests : 2901 / 2914 test pass (99% pass).
- **Tokenizer**
  - Upgraded support for ::class

Version 1.3.0 (Xue Rengui, 2018-06-03)

- **Architecture**
  - Added support for Tinkergraph 3.3.3
  - Handles situations where exakat has no database
  - Check for PHP version at bootstrap
- **Report**
  - Ambassador : Updated PHP recommendation report with PHP 7.3
  - All : Variables don't sport ... nor & anymore
- **Analysis**
  - New analysis : Single Use Variable
  - New analysis : Should Use Operator
  - New analysis : Check JSON production
  - New analysis : Report visibility usage with constants
  - Upgraded analysis : used constant is also applied to defined()
  - Upgraded analysis : used protected methods is case insensitive
  - Upgraded analysis : used directives handle function version
  - Upgraded analysis : added lcg_value for better rand
  - Upgraded analysis : Use Nullable extended to methods, closures.
  - Upgraded analysis : Fixed support for '_' native function
  - Checked unit tests : 2895 / 2907 test pass (99% pass).

Version 1.2.9 (Wang Gui, 2018-05-28)

- **Architecture**
  - Removed query cache from gremlin
  - Added pre-query check to prevent queries that have no chance of result
- **Report**
  - Ambassador : first 50% of documentation fix : double quotes are not well displayed
  - Ambassador : Results are ordered by files, then by lines
- **Analysis**

  - New analysis : Flexible Heredoc syntax
  - New analysis : Non-compatible methods
  - New analysis : Use the Blind Var
  - New analysis : Inexistant Compact
  - New analysis : Typehint / default value mismatch
  - Upgraded analysis : strict_types are not recognized as undefined constant
  - Upgraded analysis : More new methods for PHP 7.3
  - Upgraded analysis : Dependant traits
  - Upgraded analysis : Strpos comparison
  - Upgraded analysis : Method Must Return
  - Checked unit tests : 2885 / 2889 test pass (99% pass).
- **Tokenizer**
  - Interface may have const, not traits (Loading)
  - Added support for static call to methods

Version 1.2.8 (Xu Jingzong, 2018-05-21)

- **Architecture**
  - Implemented a cache for speed boost.
  - Refactored files finding method
  - Git VCS always submit a user when cloning (using exakat by default)
  - Moved custom themes from themas.ini to themes.ini
- **Report**
  - Ambassador : fixed naming the audit
  - Ambassador : added 'Dead code' section
  - Doctor : split themes display (default/customs)
- **Analysis**
  - New analysis : Report what should be done in SQL
  - New analysis : Typehinted reference
  - New analysis : Strpos doing too much work
  - New analysis : Can't instantiate class
  - Upgraded analysis : Don't echo error
  - Upgraded analysis : PPP Declaration style
  - Upgraded analysis : Useless abstract class
  - Upgraded analysis : Buried assignation doesn't report declare anymore
  - Upgraded analysis : Abstract methods are not reported as unused
  - Upgraded analysis : relaxed version constraint for all Extensions/*
  - Checked unit tests : 2852 / 2856 test pass (99% pass).

- **Tokenizer**
  - Fixed handling of short_open_tags
  - Fixed edge case with %

Version 1.2.7 (Li Yuanji, 2018-05-14)

- **Architecture**
  - Extended status command to all VCS
  - Added support for customized themes
  - Added Upgrading section, List of parametrized analysis, revamped summary
  - Simplified handling of commandline options
  - Removed usage of JSON for 'doctor'
- **Report**
  - A lot more documentation, examples, links.
  - Optimized type downloader
  - Added report themes pre-requisites
- **Analysis**
  - New analysis : ext/cmark
  - Upgraded analysis : too many children is configurable
  - Upgraded analysis : error_reporting 0 and -1 are not reported as issues.
  - Checked unit tests : 2835 / 2839 test pass (99% pass).
- **Tokenizer**
  - Fixed bug where constant self referenced.
  - Moved Identifiers to Names
  - Added first definitions for members.

Version 1.2.6 (Li Jiancheng, 2018-05-07)

- **Architecture**
  - Moved more classes to helpers
  - Removed constants for Tokens
  - Upgraded to Robo 1.2.3
- **Report**
  - Added support for custom themas for reports.
- **Analysis**
  - New analysis : zookeeper
  - New analysis : Report missing parenthesis
  - New analysis : Report invalid interval checks
  - New analysis : Suggest array_unique when possible
  - New analysis : Report when callback needs a return

  - New analysis : Reduce the number of if
  - Updated Exception list, up to PHP 7.3
  - Upgraded analysis : Printf Arguments
  - Upgraded analysis : Count On Null
  - Upgraded analysis : Regex on Collector
  - Upgraded analysis : File Inclusion wrong case handles parenthesis
  - Upgraded analysis : Make globals a property
  - Upgraded analysis : Invalid regex
  - Checked unit tests : 2814 / 2818 test pass (99% pass).
- **Tokenizer**
  - Added definition links for staticmethodcalls.
  - Added boolean and int values to __DIR__ and co.
  - Removed several static properties
  - Fixed precedence of instanceof
  - Added support for Null val

Version 1.2.5 (Li Yuan, 2018-04-30)

- **Architecture**
  - Added command 'config' to configure project from commandline
  - Made Exakat reentrant
  - Moved Configuration creation to external file
  - Upgraded status when audit isn't run yet
- **Analysis**
  - New analysis : Regex on Collector
  - Upgraded analysis : Only Variable with reference argument
  - Upgraded analysis : File Inclusion Wrong Case
  - Upgraded analysis : Invalid Regex
  - Added support for PHP 7.2.5, 7.1.17 and 7.0.30
  - Checked unit tests : 2802 / 2809 test pass (99% pass).
- **Tokenizer**
  - Fixed various bugs with constant scalar expression

Version 1.2.4 (Li Chunfeng, 2018-04-23)

- **Architecture**
  - Now fail with explicit message for memory running out
- **Report**
  - Ambassador : Updated 'confusing variables' report
- **Analysis**

  - Upgraded analysis : Could be short assignment
  - Upgraded analysis : Could be static
  - Upgraded analysis : Fail Substr Comparison (handles constants)
  - Checked unit tests : 2796 / 2801 test pass (99% pass).
- **Tokenizer**
  - Added propagation of constants when value can be processed
  - Introduced 'Parameter' token, to differentiate with Variable
  - Fixed syntax highlighting
  - Fixed a bug with negative bitshift

Version 1.2.3 (Yuan Tiangang, 2018-04-16)

- **Architecture**
  - New append for logs
- **Report**
  - New report : Manual.
  - Ambassador : Rewrote the export of 'confusing variables'
- **Analysis**
  - New analysis : report strtr bad usage
  - New analysis : don't unset properties
  - Upgraded analysis : Invalid Regex
  - Upgraded analysis : Property Could Be Local
  - Upgraded analysis : No Hardcoded path
  - Upgraded analysis : echo/print preferences also report printf
  - Removed analysis : Close Naming (now done at Report level)
  - Checked unit tests : 2770 / 2786 test pass (99% pass).
- **Tokenizer**
  - Removed double definition for functioncalls

Version 1.2.2 (Yin Kaishan, 2018-04-09)

- **Architecture**
  - Cleaned doctor so it works even without requirements
  - Fixed special chars with git URL
- **Report**
  - Ambassador : new inventory with classes changes in heritage
  - Ambassador : new inventory of large expressions
  - Upgraded report : Defined Exceptions are cleaned of doubles
- **Analysis**
  - New analysis : report Redefined Private Properties

  - New analysis : report substr() usage with strlen
  - Upgraded analysis for Inclusion Wrong Case filenames
  - Upgraded analysis : Cast To Boolean is extended to True/False
  - Upgraded analysis : Omit negative lengths
  - Upgraded analysis : interface search also include parameter counts
  - Upgraded analysis : Failed Substr Comparison handles special chars
  - Upgraded analysis : Identical consecutive omits arrays
  - Checked unit tests : 2757 / 2775 test pass (99% pass).

Version 1.2.1 (Fu Yi, 2018-04-02)

- **Architecture**
  - Fixed generation of analysis logs
  - Fixed doctor, which wouldn't diagnostic the absence of needed extensions
- **Report**
  - More real-life examples in docs
- **Analysis**
  - New favorites : property declaration unique or multiples ?
  - New analysis : $a = +$b;
  - New analysis for Melis : Regex check and Route constraints
  - Upgraded analysis : Constant used below
  - Checked unit tests : 2760 / 2766 test pass (99% pass).
- **Tokenizer**
  - Fixed counts in property declarations
  - Fixed final new lines in heredoc/nowdoc

Version 1.2.0 (Xiao Yu, 2018-03-26)

- **Architecture**
  - Upgraded concurrency with analysis
  - Replaced $_SERVER['_'] by PHP_BINARY
  - Removed old code (> 1.0.0)
  - Adopted 'stable' version for progressbar
  - Fixed loading with Bazaar
  - Added support for Parametrized analysis
  - Better initial configuration with doctor
- **Report**
  - Ambassador : upgraded analysis settings table
- **Analysis**
  - New analysis : Report Private functions for Wordpress

  - New analysis : Suggest simplifying chr(123);
  - New analysis : Too many native calls
  - Updated analysis : fallthrough are not reported with die
  - New Theme : Random
  - Collecting more stats for classes.
  - Checked unit tests : 2758 / 2741 test pass (99% pass).
- **Tokenizer**
  - Upgraded support for Heredoc

Version 1.1.9 (Qin Qiong, 2018-03-19)

- **Architecture**
  - Better documentation for reports
  - Adding Real Code examples to documentation
  - Refactored Config reading
  - Moved more VCS information to its own class
- **Report**
  - Upgraded report : Ambassador reports the number of parameters in methods
  - New report : favorites (spin-off from Ambassador)
  - Upgraded report : Inventories also covers Dateformat, Regex, Sql, Url, Email, Unicode Blocks.
- **Analysis**
  - New analysis : too many parameters
  - New analysis : report mass creation of arrays
  - Checked unit tests : 2755 / 2738 test pass (99% pass).

Version 1.1.8 (Yuchi Gong, 2018-03-12)

- **Architecture**
  - Reduced cache when running analysis
  - Fixed order of analysis
- **Report**
  - Ambassador : fixed faceted search problems
  - Codacy : added codacy-style report
- **Analysis**
  - New analysis : support for IBM db2, leveldb
  - New analysis : should use count's second argument
  - Upgraded analysis : Randomly sorted arrays
  - Checked unit tests : 2749 / 2731 test pass (99% pass).
- **Tokenizer**
  - Fixed edge case where die is an argument

  - Fixed edge case where Yield returns a array

Version 1.1.7 (Xu Maogong, 2018-03-05)

- **Architecture**
  - Removed most static in Analysis
- **Report**
  - New format : All, that produces all reports
  - Ambassador : new report estimates fitting PHP version
  - Ambassador : report enable_dl in configuration
- **Analysis**
  - New analysis : report dynamic library loading
  - New analysis : suggest array_fill_keys()
  - New analysis : PHP 7.3 optional last argument
  - New analysis : added support for xxtea, opencensus, varnish, uopz
  - Upgraded BugFixes report to PHP 7.2.3
  - Updated analysis : ext/cairo has new functions
  - Updated analysis : PHP 7.3 new functions
  - Removed analysis : NullCoalesce (double)
  - Checked unit tests : 2743 / 2731 test pass (99% pass).
- **Tokenizer**
  - Moved 'constant' to plugins
  - Fixed bug when updating with HG

Version 1.1.6 (Wei Zheng, 2018-02-26)

- **Architecture**
  - Created 'First', a recipe of initial analysis
  - Prepared installation for compose
- **Report**
  - Restored 'INLINE' results
  - New reports : Stats
  - Collect PHP native function cool
- **Analysis**
  - New analysis : report suggest compact instead of array
  - New analysis : list with references (PHP 7.3+)
  - New analysis : report situation where check is done on non-cast value
  - New analysis : foreach($array as $o -> $v) as error prone
  - Handle cases where PHP regex are not compilable anyway
  - Checked unit tests : 2732 / 2722 test pass (99% pass).

- **Tokenizer**
  - Propagate constant concatenation values.
  - Fixed calculation of intval
  - Refactored Configuration readers
  - Fixed bug when calculating __METHOD__

Version 1.1.5 (Li Shimin, 2018-02-19)

- **Architecture**
  - Refactored all reports
  - Removed outdated Devoops report
- **Report**
  - Upgraded BugFixes report to PHP 7.2.2
  - Ambassador : generates a list of confusing variables
  - New report : OWASP
- **Analysis**
  - New analysis : Use Math
  - New analysis : Extensions ext/hrtime
  - New analysis : Possible Infinite Loops
  - Upgraded analysis : addZero, Multiply by one supports new situations
  - Upgraded analysis : added microtime, uniqid .. to better rand.
  - Checked unit tests : 2719 / 2724 test pass (99% pass).
- **Tokenizer**
  - Fixed check on script compilation that was too strict.
  - Fixed internal assert()
  - Exported VCS to separate classes
  - Refactored load with 3 separate plugins : intval, noDelimiter, booval

Version 1.1.4 (The Great White Turle, 2018-02-12)

- **Architecture**
  - Build concatenation values in scalar constante expression.
  - Upgraded export of file dependencies values
- **Report**
  - Ambassador : fixed duration of audit.
  - Composer : provides a full list of depend extensions
- **Analysis**
  - New analysis : Report useless catch
  - New analysis : suggest using array_search / array_keys instead of foreach
  - New analysis : double array_flip is slow

  - New analysis : Suggest using cached values
  - New analysis : Functions that fallback to global namespace
  - Upgraded analysis : Encoded letters supports leading 0 in unicode codepoint
  - Upgraded analysis : Variable strange names now report 3 identical consecutive letters
  - Upgraded analysis : Upgraded support to __dir__
  - Checked unit tests : 2716 / 2711 test pass (99% pass).
- **Tokenizer**
  - Fixed definitions link for functions

Version 1.1.3 (The fairy Su'e, 2018-02-05)

- **Report**
  - Fixed Ambassador : the favorites weren't displayed.
- **Analysis**
  - New analysis : Report useless references
  - New analysis : Melis configuration : Undefined configuration array
  - New analysis : Melis configuration : make string.
  - Upgraded analysis : Parent first
  - Checked unit tests : 2700 / 2695 test pass (99% pass).
- **Tokenizer**
  - Better handling of Labels.
  - Fixed edge case where class and constants where mistaken one for the other

Version 1.1.2 (Jade Rabbit Spirit, 2018-01-29)

- **Architecture**
  - Upgraded docs to tinkergraph 3.2.7
- **Analysis**
  - New analysis : Report missing included files
  - New analysis : ZF3 : No Echo Outside a View.
  - New analysis : Local Global variable : report variable that looks global but are not
  - Upgraded analysis : Directive names are check with case sensitive analysis
  - Checked unit tests : 2687 / 2693 test pass (99% pass).
- **Tokenizer**
  - Magic Constant hold their actual value
  - Fixed Fullnspath for constants (case sensitive)
  - Fixed edge case with exit and die
  - Fixed edge case with exit and die and -1

Version 1.1.1 (Wood Xie of Dipper, 2018-01-22)

- **Architecture**

  - Fixed path when calling exakat from outside its install folder
  - First analysis for Melis Framework
  - Optimized dictionary collection
- **Report**
  - Ambassador : upgraded graph for class sizes
- **Analysis**
  - New analysis : report case problems with includes
  - New analysis : Melis framework
  - New analysis : inventory of view properties for Zend Framework
  - New analysis : report view files for Zend Framework
  - Upgraded analysis : + is accepted as regex delimiter
  - Upgraded analysis : same condition searches inside blocks
  - Checked unit tests : 2665 / 2671 test pass (99% pass).
- **Tokenizer**
  - Magic constants __DIR__ and __FILE__ get their actual value in noDelimiter
  - Created Eval atom
  - Removed 'Name' token for echo, print, die, exit.
  - Upgraded handling of constant names inside strings
  - Removed a bug when storing dictionary.

Version 1.1.0 (Wood Dragon of Horn, 2018-01-15)

- **Architecture**
  - Replaced 'code' property with a dictionary
- **Tokenizer**
  - Introduced 'Magicmethod' for Magic methods in class
  - Fixed a bug when ' is in file path
  - Fixed a bug when several raw HTML are in a PHP script.

Version 1.0.11 (Wood Dragon of Well, 2018-01-08)

- **Architecture**
  - Added assertion for property name.
- **Report**
  - Ambassador : Added report of classes's size.
  - Fixed missing audit end's time.
- **Analysis**
  - New analysis : Sqlite3 doesn't escape "
  - Upgraded analysis : Strange names also report qqqq sequences in variable names
  - Checked unit tests : 2617 / 2657 test pass (99% pass).

- **Tokenizer**
  - Fixed fullnspath handling for constants (case insensitive for the constant name)

Version 1.0.10 (Wood Wolf of Legs, 2018-01-01)

- **Architecture**
  - Fixed Sqlite3 escaping error : use ‘, not “
- **Report**
  -
- **Analysis**
  - Upgraded analysis : ? is possible as delimiter
  - Analysis works better with nested structures
  - Checked unit tests : 2601 / 2649 test pass (99% pass).
- **Tokenizer**
  - First plugin for Load Task.
  - Upgraded support for define-d constant.
  - Introduced Phpvariable
  - Fixed scoping with array index.

Version 1.0.9 (King of Dust Protection, 2017-12-25)

- **Report**
  - Ambassador : list complex expressions.
  - Dump : added function inventory
  - Dump : added begin and end line for structures.
- **Analysis**
  - New analysis : report reference error with Ternary operator
  - New analysis : report Undefined classes in Wordpress.
  - Upgraded analysis : preg option E, tighter regex.
- **Tokenizer**
  - Better handling of long path name. TBC.
  - Introduced Parent, Static, Self, Exit, Echo, Print.

Version 1.0.8 (King of Heat Protection, 2017-12-18)

- **Architecture**
  - Doctor reports memory_limit and JAVA_OPTIONS/JAVA_HOME
  - Made database restart more portable
  - Added spell checking on docs
- **Report**
  - Ambassador : Regex inventory added
  - Ambassador : Largest expressions reported

- **Analysis**
  - New analysis : report identical operands on both sides of operator
  - New analysis : report potentially mistaken concatenation in array
  - New analysis : report mistaken scalar typehint
  - New analysis : report undefined classes by symfony version
  - New analysis : report undefined classes by wordpress version
  - Upgraded analysis : Interfaces are also reported from return typehint
  - Upgraded analysis : Mistaken concatenation got rid of various false-positives
  - Checked unit tests : 2601 / 2633 test pass (99% pass).
- **Tokenizer**
  - Isset, Empty, Phpvariables now have their own atom.
  - Fixed edge case with $ token
  - Fixed Constant fqn building
  - UTF-8 protection for propertyname

Version 1.0.7 (King of Heat Protection, 2017-12-11)

- **Architecture**
  - Added /var to default omitted folders
- **Analysis**
  - New analysis : should use array_filter.
  - New analysis : ext/igbinary
  - Checked unit tests : 2533 / 2599 test pass (97% pass).
- **Tokenizer**
  - Fixed

Version 1.0.6 (Fuli, 2017-12-04)

- **Architecture**
  - Refactored description
  - Moved PHPsyntax to a function
- **Analysis**
  - New analysis : Never used parameter.
  - New analysis : always use named boolean parameters
  - Upgraded analysis : unused arguments
  - Checked unit tests : 2573 / 2585 test pass (99% pass).
- **Tokenizer**
  - Added new token : This for $this
  - Updated loader to handle PHP 7.3 functioncall syntax (final ,)
  - Turned Markcallable into an independant analysis

Version 1.0.5 (King of Cold Protection, 2017-11-27)

- **Architecture**
  - Configured Exakat for Tinkergraph 3.3. Still unfinished.
  - Documentation now has an external link to extensions.
- **Report**
  - Ambassador : added more inventories : URL SQL, email, GET index, MD5, Mime
- **Analysis**
  - New analysis : parent first
  - New analysis : Report uncommon Environment Vars
  - New analysis : Report invalid Regex
  - New analysis : Report contatenation in Zend DB
  - Fixed analysis : Deprecated Functions
  - Fixed analysis : Unknown PCRE2 option
  - Upgraded analysis : hardcoded password
  - Upgraded analysis : array_merge in loops
  - Upgraded analysis : substr() first. Handle following expressions
  - Refactored analysis : Used Functions
  - Refactored analysis : Add Zero
  - Checked unit tests : 2573 / 2585 test pass (99% pass).
- **Tokenizer**
  - Fixed a bug that linked functions and definitions

Version 1.0.4 (Boxiang Demon, 2017-11-20)

- **Architecture**
  - PhpExec, get only path to binary.
  - Cleaned docs of double links
  - Cleaned code
- **Report**
  - Added libsodium, Argon2 to Crypto; DL() usage to PHP.
  - Compatibility report only focuses on backward incompatibilities.
  - New recipes will cover 'suggestions for better code'. Coming up.
- **Analysis**
  - New analysis : " string is better than ' (sorry. . .)
  - New analysis : PHP 7.3's PCRE 2
  - New analysis : report missing 'new' in front of class name.
  - New analysis : use is_object instead of is_resource for ext/hash
  - New analysis : report non-countable calls

  - New analysis : report DL usage in Appinfo
  - New analysis : slice first, then map arrays.
  - New analysis : Avoid 5th argument in PHP 7.2 for set_error_handler
  - New analysis : avoid null with get_class()
  - New analysis : suggest using list() with foreach instead of arrays
  - New analysis : avoid using $this as argument in constructor
  - New analysis : Report usage of ext/vips
  - New inventory : GPC variables
  - Updated analysis : Use Class Operator doesn't report methods names anymore
  - Updated analysis : Long argument size is raised to 60 chars
  - Updated analysis : ignore when missing break is in last case
  - Updated analysis : Use This ignores 'self'.
  - Updated analysis : Randomly sorted Arrays ignores arrays of 3 or less.
  - Updated analysis : ext/mcrypt gets its constants
  - Updated analysis : more strange names being used in code
  - Updated analysis : more PHP 7.2 removed functions
  - Checked unit tests : 2563 / 2572 test pass (99% pass).

- **Tokenizer**
  - Reduced duplicated that may lead to loading error.

Version 1.0.3 (Baize Demon, 2017-11-13)

- **Architecture**
  - Fixed driver Tinkergraph, which was not setting the right ids.
  - Doctor now reports $JAVA_OPTIONS, in case one need to allocate more memory
  - Doctor now reports token limit
  - Moved config.ini creation to first phase of init.
  - Fixed collect of error when init with git.
  - Upgraded driver gremlin-php to 3.0.2
- **Report**
  - Ambassador : Now reports the namespaces as a tree.
  - New analysis : report members that are static and not.
  - Updated analyzis : normal method called statically.
- **Analysis**
  - Added support for Drupal, FuelPHP and Phalcon.

Version 1.0.2 (Suanni Demon, 2017-11-06)

- **Architecture**
  - Better report of error messages from VCS.

  - Updated support for Vagrant

- **Report**
  - Ambassador : Fixed display for 'Callback'

- **Analysis**
  - New analysis : substr() first, then replace.
  - New analysis : report double prepare (WP).
  - New analysis : avoir the +1 month trap
  - New analysis : check for printf() options
  - New analysis : check for placeholder in prepare (WP)
  - New analysis : avoid direct injection into prepare (WP)
  - New analysis : performance recommendation for switch.
  - New analysis : merge if/if into if/then/else
  - Checked unit tests : 2500 / 2536 test pass (99% pass).

Version 1.0.1 (Xueshi Demon, 2017-10-30)

- **Architecture**
  - Created Result class for Graphdb results
  - Docker image is updated with version 1.0.1
  - Vagrant files are updated with version 1.0.1
  - Preparing support for Gremlin 3.3.0

- **Report**
  - Added support for PHP 7.1.11 and 7.0.25

- **Analysis**
  - New analysis : could be else (for consecutive opposite if/then)
  - Checked unit tests : 2517 / 2527 test pass (99% pass).

Version 1.0.0 (Roushi Demon, 2017-10-23)

- **Architecture**
  - Tested on Gremlin 3.2.6. Checked Gremlin 3.3.0, but it needs more work.
  - Upgraded doctor for installation and report.
  - Upgraded docs to set gremlin-server as default install.

- **Report**
  - Added support for Clang-style report.
  - Ambassador : fixed link to exception Tree.
  - Inventories : Date format,
  - Audit names are reported in every Ambassador-style report.

- **Analysis**
  - Upgraded PHP directive list.

  - Functions In For loop : prevent issue if the function uses a loop variable.
  - Useless instruction : do not report return $i++ if $i is reference
  - Useless instruction : Avoir reporting properties when they are magic
  - New analysis : mark properties to be magic.
  - Upgraded list of PHP logins, to report hard coded passwords.
  - Upgraded close naming : variables that differ with 1 chars are reported.
  - Added assert(false...) to list of branching syntax.
  - Checked unit tests : 2515 / 2525 test pass (99% pass).

Version 0.12.16 (Tawny Lion Demon, 2017-10-16)

- **Report**
  - Beta version for Drill Instructor
  - Upgraded Inventories report with Sessions, Cookies, Incoming variables
- **Analysis**
  - New analysis : Expression too complex.
  - New analysis : Session Handler must implements SessionUpdateTimestampHandlerInterface
  - New analysis : is Zero : additions that negate some terms
  - New analysis : unconditional loops
  - Upgraded Zend Framework review with latest versions (feed, http, eventmanager...)
  - Upgraded 'Strange names' with new typos
  - Upgraded 'Logical to in_array' to handle separated comparisons
  - Checked unit tests : 2505 / 2515 test pass (99% pass).
- **Tokenizer**
  - Fixed bug with Sign in Additions.

Version 0.12.15 (Nine Headed Lion, 2017-10-09)

- **Architecture**
  - Server : now supports stop, start and restart.
  - Every audit gets a random name, for easy differentiation
  - Added support for PHP 7.3
- **Report**
  - Ambassador : list of analysis that report nothing : Good job!
  - Slim report : fixed build
- **Analysis**
  - New analysis : file upload names vulnerability check
  - New analysis : variable that may hold different types of date
  - New analysis : always anchor regex
  - Checked unit tests : 2475 / 2480 test pass (99% pass).

Version 0.12.14 (Grand Saint of Nine Spirits, 2017-10-02)

- **Architecture**
  - Support UTF-8 on Gremlin Server (other encoding are not)
  - Better display of vcs updates
- **Report**
  - Ambassador : added Security and Performances
  - Ambassador : Upgraded exception presentation
- **Analysis**
  - New analysis : report fallthrough in switch
  - New analysis : inventory regex
  - Added support for PHP 7.1.10 and 7.0.24

Version 0.12.13 (King of the Southern Hill, 2017-09-25)

- **Architecture**
  - Code cleaning
- **Report**
  - Ambassador : changed display of the audit
- **Analysis**
  - Refactored several analysis

Version 0.12.12 (Ruler of the Kingdom of Miefa, 2017-09-18)

- **Report**
  - Ambassador : fixed collect of interfaces and trait names
- **Analysis**
  - New analysis : ext/Parle
  - New analysis : help optimize pathinfo() usage
  - New analysis : catch array_values() usage with list and pathinfo()
  - Updated analysis : Don't show error messages with catch->getMessage();
  - Updated analysis : No concat in loop handles $x = $c . $x;
  - Checked unit tests : 2456 / 2461 test pass (99% pass).
- **Tokenizer**
  - Added support for ', " and > in file names. Still missing support for
  - Restaured fallback to global constants.
  - Fixed special case : <?php ++$x ?>

Version 0.12.11 (Half-Guanyin, 2017-09-11)

- **Architecture**
  - Added support options for branches and tags
  - Added support for config in server mode

- **Report**
  - Fixed methods dump for interfaces.
- **Analysis**
  - Added all analysis to report could be private/protected for
- **Tokenizer**
  - Fixed handling of '<' char in paths

Version 0.12.10 (Golden Nosed Albino Rat Spirit, 2017-09-04)

- **Architecture**
  - Upgraded server version with config alteration features.
  - New generated config-cache
- **Report**
  - Fixed property names in Visibility report
- **Analysis**
  - Arrays/IsModified : arrays are not modified unless in a (unset)
- **Tokenizer**
  - Fixed 'constant' for functioncalls
  - Introduced 'Name' for Identifier without a fullnspath
  - Added support for branches and tags in init
  - Fixed edge case with $o->$$b

Version 0.12.9 (Lady Earth Flow, 2017-08-28)

- **Architecture**
  - Creates config.cache, with cached calculated configs. Remove to update.
- **Report**
  - GraphQL : Upgraded GraphQL report, with relationships.
- **Analysis**
  - New analysis : suggest moving for() to foreach()
  - New analysis : shell_exec/exec/*backtick* favorite
  - Update analysis : Abstract Static is for PHP 7.0-
- **Tokenizer**
  - Removed Arguments and ARGUMENTS.
  - Finished 'factory' from Config.
  - Better handling of long path names.

Version 0.12.8 (ruler of the Kingdom of Biqiu, 2017-08-21)

- **Analysis**
  - New analysis : use foreach, not for()
  - New analysis : ext/fam, ext/rdkafka

- **Tokenizer**
  - Fixed edge case where pathnames are too long on OSX.

Version 0.12.7 (Old Man of the South Pole, 2017-08-14)

- **Architecture**
  - Fixed project_vcs when none is used.
- **Analysis**
  - Better documentation for in_array replacements and array_unique()
  - Added support for PHP 7.1.8 and 7.0.22

Version 0.12.6 (White Faced Vixen Spirit, 2017-08-07)

- **Analysis**
  - New analysis : no negative for strings before 7.1
  - New analysis : use in_array instead of ||
  - Updated analysis : preg_quote has no delimiter
- **Tokenizer**
  - Fixed bug in handling real value for negative numbers

Version 0.12.5 (White Deer Spirit, 2017-07-31)

- **Architecture**
  - Removed config singleton
- **Report**
  - New report : simpletables (HTML)
- **Analysis**
  - New analysis : report optional parameters
  - New analysis : report concat inside a loop
  - Updated analysis : Could Be Class Constant, when no visibility is provided.

Version 0.12.4 (peacock Mahamayuri, 2017-07-24)

- **Architecture**
  - Optimized performances for large projects (over 2M tokens)
  - Support Neo4j as a driver for Tinkgerpop
- **Report**
  - Now covering all PHP 7.2 features
- **Analysis**
  - New analysis : Extension xattr
  - New analysis : report 'object' as a class name
  - New analysis : No Array for magic property
  - New analysis : suggest reducing code for isset
  - New favorite : and / &&

  - Updated analysis : fetch correct delimiter, even if escaped.
  - Extended coverage for several analysis
  - Removed several nested-subqueries (bad for performances)
- **Tokenizer**
  - Tinkergraph/Neo4j : reworked loading data from disk.
  - Added protection for $ in filename

Version 0.12.3 (Golden Winged Great Peng, 2017-07-17)

- **Architecture**
  - Prepared options for several back servers : Tinkergraph, Gremlin-Server/Neo4j, Janusgraph
- **Report**
  - New report : Marmelab (GraphQL server)
- **Analysis**
  - New analysis : Report when a property is used as object or scalar
  - New analysis : Mismatched Typehint
  - New analysis : Mismatched Default values
  - Upgraded analysis :
  - Fixed a gremlin bug in noAtomInside
- **Tokenizer**
  - Added support for trailing comma in group use (PHP 7.2)
  - Fixed building of constants' values

Version 0.12.2 (Samantabhadra, 2017-07-10)

- **Architecture**
  - Added support for Tinkergraph as graph backend
- **Report**
  - Ambassador : reports callback/closures, all 3 declares (ticks, encoding, strict_types)
  - Ambassador : reports strict_types as favorite
  - PlantUML : upgraded report
- **Analysis**
  - New analysis : Mismatched ternary branches
  - New analysis : mkdir, by default, uses 777.
  - New analysis : ext/lapack
  - Upgraded analysis : option E for preg_match, refined results
  - Checked unit tests : 2337 / 2366 test pass (99% pass).
- **Tokenizer**
  - Added support for Instanceof and GROUPUSE with Nsname

Version 0.12.1 (Yellow Toothed Elephant, 2017-07-03)

- **Architecture**
  - Refactored structures extractions in dump
- **Report**
  - New report : PlantUML
  - Ambassador : Appinfo now reports how popular is a feature
- **Analysis**
  - New analysis : Const / Define() favorite for constants
  - New analysis : do not return in finally
  - Upgraded analysis : Add Zero was refactored
- **Tokenizer**
  - Prepared list of tokens and relations

Version 0.12.0 (Manjusri, 2017-06-26)

- **Architecture**
  - Added support for Janusgraph (Gremlin 3)
  - Refactored dump's data collection for speed.bb
- **Report**
  - Added support for Wordpress and Joomla as Frameworks
- **Analysis**
  - New analysis : Avoid Optional properties
  - New analysis : Multiple declarations of functions
  - New analysis : Non breakable spaces in names
  - New analysis : Favorite Heredoc delimiter
  - New analysis : ext/swoole
- **Tokenizer**
  - Modified several nodes/links names, for compatibility purposes

Version 0.11.8 (Xiaozuanfeng, 2017-06-19)

- **Architecture**
  - Starte working on JanusGraph to add to Neo4j/Gremlin3
- **Report**
  - Ambassador : reports Strings encoding and Unicode-block (when available)
  - Ambassador : reports framework founds (first 6, more as we go).
  - Ambassador : reports how frequently an analysis yield results to compare with current situation
- **Analysis**
  - New analysis : Classes where declaration order differs from : use, const, properties and methods.
  - New analysis : Could use interface (but implements is missing)
  - New analysis : Cant Inherit Abstract Method (PHP 7.2 upgrade)

  - New analysis : use session_start() options
  - Updated analysis : Dynamica method calls cover {} too
  - Checked unit tests : 2305 / 2305 test pass (100% pass).
- **Tokenizer**
  - Checked code on early PHP 7.2 version

Version 0.11.7 (Long Armed Ape Monkey, 2017-06-12)

- **Report**
  - Ambassador : report detected patterns (2 firsts)
  - None report : for when dump is sufficient
- **Analysis**
  - New analysis : could factor functioncalls
  - New analysis : PSR-* usage
  - New analysis : support for Judy and Gender extensions
  - Added thema for Compatibility PHP 7.3
  - Added thema for Dependency Injection
- **Tokenizer**
  - Fixed edge case where classes starting with 'namespace' where mistakenly processed
  - Removed Block from CIT

Version 0.11.6 (Red Bottomed Horse Monkey, 2017-06-05)

- **Architecture**
  - Removed singleton to Config. WIP
- **Report**
  - Ambassador : reports usage of PSR 3,6,7,11,13,16.
  - UML : report now protects file names
- **Analysis**
  - New analysis : Ext stats
  - New analysis : report mixed concatenation / interpolation strings
  - Updated analysis : htmlentities actually uses combinaison, not alternatives,
  - Updated analysis : Close Tag consistency ignores __HALT_COMPILER files

Version 0.11.5 (Intelligent Stone Monkey, 2017-05-30)

- **Report**
  - Ambassador : fixed visibility suggestion
  - New report : Dependency wheel
- **Analysis**
  - New analysis : avoid typehinting with classes
  - New analysis : implemented methods must be public

  - New analysis : no reference on left of assignement
  - New analysis : Could typehint with instanceof
  - Updated analysis : Useless parenthesis cover clone, yield, yield from.
  - Updated analysis : Make One Call also reports nested calls
- **Tokenizer**
  - Split functions and closures,
  - Split classes and anonymous classes
  - Split variable with definitions (Property, Static and Global)
  - File count is always reported (even 0)

Version 0.11.4 (Six Eared Macaque, 2017-05-22)

- **Architecture**
  - Results : returns now multiple results at once
- **Report**
  - New report : codeflower
  - Ambassador : report usage of Debug functions, browscap
  - Ambassador : omits 0 in donuts
  - Ambassador : faceted search for compatiblity
- **Analysis**
  - New analysis : report functions whose return is not used
  - New analysis : only variable can be passed by reference
  - Added limits to all in-depth searches
  - Checked unit tests : 2216 / 2216 test pass (100% pass).
- **Tokenizer**
  - Fixed edge case, where return is finished by a close tag
  - Split Variables into Variables, Objects and Arrays.

Version 0.11.3 (Sun Deity of Mao, 2017-05-15)

- **Architecture**
  - Speed up batch processing for lists of analysis
  - Split data collection from the initial dump.
- **Report**
  - Ambassador : Upgraded presentation of issues, and internals links.
- **Analysis**
  - New analysis : Sphinx extension
  - New analysis : GRPC extension
  - New analysis : reports arrays that are randomly sorted.
  - New analysis : report multiple catch clauses

  - Updated analysis : direct injections include all SERVER_* values
  - Upgrade for PHP 7.1.15 and 7.0.19
- **Tokenizer**
  - Split Functioncall into Functioncall, MethocallCall and Newcall.
  - Added support for 'namespace' in any full name.

Version 0.11.2 (Scorpion Demon, 2017-05-08)

- **Architecture**
  - Code cleaning, and more stability
- **Analysis**
  - New analysis : Report preference between != and <>
  - New analysis : report empty regex and wrong delimiters
  - Added protection for $ in RegexDelimiters

Version 0.11.1 (Ruler of Women's Country, 2017-05-01)

- **Architecture**
  - Fixed handling for large list of data in gremlin queries
  - Handles static in anonymous classes correctly
- **Report**
  - Reports handle traits like class.
- **Analysis**
  - New analysis : ends arrays with , or not (favorite)
  - New analysis : suspicious comparison
  - New analysis : strange spaces in strings
- **Tokenizer**
  - Arrays are now Arrayliteral, split from Functioncall

Version 0.11.0 (Immortal Ruyi, 2017-04-24)

- **Architecture**
  - Removed prepared statements from loops in dump
  - made Gremlin cache compatible with 32bits platforms
- **Report**
  - Ambassador : first work on upgrading visibilities for properties.
- **Analysis**
  - New analysis : could use str_repeat()
  - New analysis : Crc32() Might Be Negative
  - Update analysis : Queries in loop reports cubrid and sqlsrv, prepared statements.
  - Update analysis : type mismatch for indices works on constants too.
  - Update analysis : Loop calling covers less ground

- **Tokenizer**
  - Split function and method entities for differentiated processing

Version 0.10.9 (Single Horned Rhinoceros King, 2017-04-17)

- **Architecture**
  - File extensions are processed before include/ignore dirs.
  - Reduced number of DEFINITION links, leading to less processing.
  - Added several assertion() in the code
  - Added assertions report in doctor (better leave them out with phar)
- **Report**
  - Added support for PHP 7.0.18 and 7.1.4
  - Ambassador : better layout for favorites
  - Zend Framework : 8 new components supported
  - Zend Framework : now supports zendframework/zendframework too
  - Zend Framework : report unused components
- **Analysis**
  - New analysis : report nested Use expressions
  - New analysis : report repeated regex (to be federated)
  - New analysis : report code that output directly to std
  - Updated analysis : Should use this now omits overwritten methods
  - New analysis : report overwritten methods
  - Upgraded analysis : 2123 / 2123 test pass (100% pass)

Version 0.10.8 (King of Spiritual Touch, 2017-04-10)

- **Report**
  - Slim report : list of routes used.
- **Analysis**
  - New analysis : report Group Use Declaration (PHP 7.0+)
  - Zend Framework : 30 components are now covered.
  - Slim : No echo in route callable and Inventory of routes.
  - PHP : list of new PHP 7.2 functions.
- **Tokenizer**
  - Sped up loading time by 10%.
  - Added support for PHP6 binary string : $a = u'b';

Version 0.10.7 (Immortal of Antelope Power, 2017-04-03)

- **Report**
  - Ambassador : fixed composer report.
  - Added report for Composer (beta phase)

  - Added report for Slim framework.
- **Analysis**
  - Added support for Slim versions.
  - Added 10 new components for Zend Framework 3
- **Tokenizer**
  - Fixed support for $ in file names.

Version 0.10.6 (Immortal of Elk Power, 2017-03-27)

- **Architecture**
  - Major speed up of loading and analysis
  - Fixed themes configuration.
- **Report**
  - Ambassador : report cookies usage, infinite and NAN usage
  - Zend Framework : Report incompatibilites component/version for ZF3
- **Analysis**
  - Upgraded analysis : 1941 / 1941 test pass (100.00% pass)
  - New analysis : Zend Framework 3 Deprecated
  - New analysis : Zend cache, view, db.
  - New analysis : Report missing type tests.
  - New analysis : suggest setcookie() with safe arguments
  - New analysis : Do not cast to Int
  - New analysis : CakePHP classes compatibilities from 2.5 to 3.3
  - Upgraded analysis : instanceof doesn't report traits anymore
  - Upgraded analysis : mb_ereg has options in the 4th arguments
  - Upgraded analysis : more strange names
- **Tokenizer**
  - Reviewed most of the load processing.
  - Reduced the number of 'fullnspath' properties.

Version 0.10.5 (Immortal of Tiger Power, 2017-03-13)

- **Architecture**
  - Collect graph size in dump.sqlite
  - Collect memory usage in dump.sqlite
  - Now uses the calling PHP version to run all parts of exakat (no config)
  - Doctor report the ran gremlin version.
- **Report**
  - Ported the Zend Framework report to ambassador
  - Added regex delimiter in favorites.

  - Ambassador : syntax coloring

- **Analysis**
  - New analysis : could be typehinted 'callable'
  - New analysis : encoded letters in strings for security
  - New analysis : report arguments that may be callable
  - New analysis : report strangely named variables
  - New analysis : report strangely named constants
  - New analysis : too many FindsBy*() methods
  - Updated analysis : Useless Instructions doesn't report array_merge(_recursive) with one argument
  - Updated analysis : array_replace handles ...
  - Updated analysis : 7.2 deprecation with assert()
  - Generalized usage of commons for CIT
  - Added first 4 set of analysis for Zend Framework 3
  - Added support for dynamic new $a[i];
- **Tokenizer**
  - Fixed fullnspath with new on functioncall
  - Reduced the number of fullnspath loaded
  - Added support for 's'() as functioncall
  - Fixed case where file names has ' ' in it

Version 0.10.4 (Dragon King of the West Sea, 2017-03-06)

- **Architecture**
  - Ignore some classic files by default (README, LICENSE...)
- **Report**
  - Ambassador : protection of HTML values
  - PHPcompilation : fixed export to stdout
- **Analysis**
  - New analysis : report useless else branches
  - New analysis : should regenerate session Id, for PHP and Zend Framework
  - Added support for Extension Data structures (ext/ds)
  - Upgraded analysis : Hardcoded Hash
  - Speed up analysis for extensions
- **Tokenizer**
  - Fixed edge case where a constant was used inside a ternary operator
  - Fixed processing of labels

Version 0.10.3 (Dragon King of the Jing River, 2017-02-27)

- **Architecture**

  - Added URL glossary to Manual.
  - Extended CS ruleset
  - Use exakat/exakat as user/login for git.
  - New helper to rename analysis
  - Project command now accept -P/-T to run one analysis/Thema directly
- **Report**
  - New report style : Codesniffer
- **Analysis**
  - New analysis : suggest usage for array_column()
  - New analysis : __DIR__ must be concatenated with a string starting with '/'
  - New analysis : report usage of parent, self and static outside a class/trait
  - New analysis : report properties used only in one method
  - New analysis : report properties used only once at all
  - New analysis : multiple aliases per class
  - Updated analysis : Fopen() mode support 'e' option (7.1.2 +)
  - Updated analysis : Make One Call covers str_replace, substr_replace, preg_replace*
  - Updated analysis : Unused arguments : now ignores arguments from interface or parent
- **Tokenizer**
  - Removed double DEFINITION link. Faster loading, less processing.
  - Fixed an edge case when function name is boolean or null.
  - Cleaned atom and tokens names
  - Fixed edge case when object is instantiated in a ternary

Version 0.10.2 (Water Lizard Dragon, 2017-02-20)

- **Architecture**
  -
- **Report**
  - Text format now understand -T, -P to extract only some of the results.
  - Fixed dump of extends.
- **Analysis**
  - Added support for PHP 7.1.2 and PHP 7.0.16
  - New analysis : report forgotten 'throw' keyword.
  - New analysis : report class / function confusing name
  - Added support for libsodium
  - Upgraded PHP Relaxed Keyword : Ignore properties.
  - Upgraded analysis : 1824 / 1826 test pass (99.9% pass)
- **Tokenizer**

  - Fixed a bug that mistakes native PHP classes for functions
  - Fixed rare situation with grouped const/function.

Version 0.10.1 (King of Wuji Kingdom, 2017-02-13)

- **Architecture**
  - Report SVN revision when updating or not.
  - Default reports are in config.
  - Configure now supports include_dirs, to include files.
  - Project name is now noted in datastore.
  - Inventories is a default themas; PHP Compatibility < 5.6 are not default anymore.
- **Documentation**
  - Fixed outgoing links
  - Better coverage of PHP functions
- **Report**
  - Added 'Inventories' report : reports all names and literals
  - Ambassador : Added list of included files, Yield From and classes stats
- **Analysis**
  - New Analysis : Strange Names For Methods (Classes/StrangeName)
  - New Analysis : SQL queries (Type/Sql)
  - New Analysis : Avoid Non Wordpress Globals (Wordpress/AvoidOtherGlobals)
  - Upgraded analysis : Should be single quote, escape sequences refined.
  - Upgraded analysis : Should Preprocess now support determinist PHP functions
  - Upgraded analysis : 1817 / 1824 test pass (99.6% pass)
- **Tokenizer**
  - Fixed LOC counting.
  - Fixed edge case when closure is directly use as argument
  - Fixed double inventories for Use's Definitions

Version 0.10.0 (Azure Lion, 2017-02-06)

- **Architecture**
  - Replacement of booleans with constants (WIP)
  - Removed PHPloc (merged features into load)
  - Added coding standard for Code Sniffer (ruleset.xml)
  - PHP version used default to running script version
  - Now reading Token Constants from the binaries
  - Doctor reports project configuration if -p is used
- **Report**
  -

- **Analysis**
  - New Analysis : No Boolean As Default
  - New Analysis : Raised Access Level
  - New Analysis : Recommend Wpdb->prepare when variables are in query
  - Directive suggestion now include error_log
  - Upgraded analysis : UselessParenthesis also checks Typehint
  - Upgraded analysis : 1804 / 1811 test pass (99.6% pass)
- **Tokenizer**
  - Reinforced detection of parsable PHP script
  - Fixed Files command : it now cleans data before running
  - Removed warning about memory
  - Index creation made lighter

Version 0.9.9 (Pilanpo Bodhisattva, 2017/01/30)

- **Architecture**
  - Moving true/false to constants
- **Report**
  - Ambassador : Added 'Compilation' and Version compatibility reports.
  - Prepared collection of dependencies in dump
- **Analysis**
  - New Thema : Compatibility PHP 7.2
  - New analysis : Deprecated Features of PHP 7.2
  - New analysis : Removed Function for PHP 7.2
  - New preference : New Line Style
  - Upgraded analysis : 1781 / 1802 test pass (98.9% pass)

Version 0.9.8 (Multiple Eyed Creature, 2017-01-23)

- **Architecture**
  - Moved 'Truthy/Falsy' as 'boolean' characteristics
  - Updated Gremlin3 interface to handle Groovy maps
  - Added default name when creating project
- **Report**
  - Added checks on merged table at Dump stage
  - Added support for PHP 7.1.1 and 7.0.15
- **Analysis**
  - New analysis : variables assigned twice or more
  - New preference : new x() / new x;
  - Upgraded analysis : 1785 / 1794 test pass (99.5% pass)

  - Fixed Interface usage : missing interfaces extends interfaces
  - Added extra check for Functioncalls
- **Tokenizer**
  - Added support for instanceof + several names

Version 0.9.7 (Hundred Eyed Demon Lord, 2017-01-16)

- **Architecture**
  - Fixed constant names for tokens in Load
  - Changed duplication check to dedup(). Cleaned analysis for duplicates.
  - Speed but for large projects. Work in Progress.
  - Reduced usage of static properties
  - Better detection of PHP scripts during project
- **Report**
  - Fixed generation of inventories when no target is provided
- **Analysis**
  - New analysis : Could Be Protected Property (not a public)
  - New analysis : avoid large literal arrays in local variables.
  - New analysis : report long arguments.
  - Removed analysis : Structures/EchoArguments (double with Echo With Concat)
- **Tokenizer**
  - Fixed list of constants for PHP 7.1

Version 0.9.6 (Spider Demons, 2017-01-09)

- **Architecture**
  - Added support for report/analysis theme list in config (exakat and project)
  - Better cleaning of projects
  - Doctor : Initialisation with themes/reports; Reports executable being used.
  - Added a log for gremlin Queries
  - Rebuild the server command
  - Added 'catalog' command
- **Report**
  - Split Phpconfiguration into eponymous and Phpcompilation
- **Analysis**
  - New analysis : avoid Glob, use scandir without sorting.
  - New analysis : always configure ext/sqlite3 FetchRow()
  - New analysis : no string with append
  - Removed analysis : Structures/ForeachSourcesNotVariable
  - Upgraded Analysis 'Should Import Functions'

  - Upgraded analysis : 1764 / 1773 test pass (99.5% pass).

- **Tokenizer**
  - Added 'aliased' property to nodes.

Version 0.9.5 (Immortal Ziyang, 2017-01-04)

- **Architecture**
  - Better check of PHP version
- **Report**
  - Ambassador : report analysis settings
  - PHP Compilations : supports all extensions
  - New report : Inventories
- **Analysis**
  - New analysis : Don't Use Fallback to Global space
  - New analysis : MongoDB (ext/mongo version 3)
  - New analysis : zbarcode
  - Bug : Fixed intval for octals in Arrays/MultipleIdenticalKeys
  - Removed analysis : Php/InconsistantClosingTag (double)
- **Tokenizer**
  - Ranking arguments, not functioncall

Version 0.9.4 (Lady of Jinsheng Palace, 2016-12-19)

- **Architecture**
  - Rewrote the concurrence check (removed needs for ext/sem)
  - Results are never double anymore
  - Upgraded gremlin calls, to handle n
  - Dump cleans the previous values before dumping
  - Excluded namespaces classes when searching for external libraries
- **Report**
  - Ambassador : extension usage, inventories, global lists, stats, PHP Compilation directives
  - Covers more compilation directives (Not finished)
- **Analysis**
  - New analysis : Final by Ocramius
  - Upgraded : Comparison with == : added curl_exec
  - Upgraded : isset with constant (mistake on properties as arrays)
  - Upgraded : Avoid using now uses full NS path
  - Upgraded : Useless instructions handles for() correctly
  - Upgraded : Recursive, IsGenerator and Loop Calling includes yield from
  - Upgraded analysis : 1741 / 1750 test pass (99.5% pass).

Version 0.9.3 (Purple-Gold Bells, 2016-12-12)

- **Architecture**
  - Lots of cleaned code
  - Harmonized data for extensions
  - Stop 'project' if no code is available
  - Now using stub in phar.
- **Report**
  - Added directives, bugfixes, external services and
  - Added support for PHP 7.0.14 and 5.6.29
- **Analysis**
  - New analysis : Wordpress, recommend prepare()
  - More favorite reports : final ?> and unset()/(unset)
  - Reduced number of double reports for many analysis
  - Update : Fixed analysis with $THIS
  - Upgrade : report useless casting of comparisons
  - Update : Should use this takes into account parent

Version 0.9.2 (Golden Haired Hou, 2016-12-05)

- **Architecture**
  - First version of Exakat for docker (beta)
  - Added a waiting loop in cleandb
  - Docs include a list of new analysis per version
- **Report**
  - Added 2 first inventories, Appinfo() in Ambassador
  - Favorites now reports global/$GLOBALS
  - Restore composer.lock report
  - Upgraded uselessReturn for the final return.
- **Analysis**
  - New analysis for Newt, Nsapi,
  - New analysis : __ in methods names
  - New analysis : Too many local variables
  - New analysis : Avoid array_push()
  - Upgraded ext/apache coverage

Version 0.9.1 (Sai Tai Sui, 2016-11-28)

- **Architecture**
  - Docker supported in exakat/config.ini for PHP binaries.
  - Added exakatSince in analysis documentation

  - Added some missing tokens in anonymize command
- **Report**
  - Added several new analysis for PHP 7.1
- **Analysis**
  - new analysis : find methods that could return Void
  - new analysis : find malformed octal sequence in strings
  - new analysis : spot rethrown exception
  - new analysis : reach the last element
  - new analysis : find undefined Zend Framework classes (2.0 to 3.0)
  - Upgraded analysis : 1706 / 1714 test pass (99.5% pass).
- **Tokenizer**
  - Fixed handling references (some were missing)
  - Fixed handling of ellipsis (some were missing)

Version 0.9.0 (Python Demon, 2016-11-21)

- **Architecture**
  - Project now include 'Preference' analysis
  - Dump is now incremental (-u option), and doesn't need to be run in paralell
  - Added new hashAnalysis table, to handle generic results from analysis.
  - Added project name in the graph.
  - New command 'status' to report the current status of exakat
- **Report**
  - Ambassador includes 'Preferences' section and new menu system
  - Upgraded progressbar to display project processing
- **Analysis**
  - New analysis : Early Bail Out (with if/then)
  - New analysis : PHP 7.1 backward incompatibilities with microseconds
  - New analysis : Wordpress : recommend using WP api, not PHP.
  - Upgraded 'Constant condition' to include do..while()
  - Upgraded 'Useless Abstract' to include methodless classes
  - Upgraded analysis : 1687 / 1697 test pass (99% pass).
- **Tokenizer**
  - Added 'Array' to list of determinist functions (more constants are spotted)
  - Fixed 'Name' for Array Short Syntax.
  - Fixed variadic support

Version 0.8.9 (Yellow Brows Great King, 2016-11-14)

- **Architecture**

  - Fixed and document -tgz and -zip option of init
  - Removed progress folder
  - Made MagicNumber a parallel task in Project.
  - Turned some die into assertion()
  - .phar doesn't report any PHP errors.
  - Checked compilation with PHP 5.3->7.2
- **Report**
  - Removed Faceted report
  - Added Bugfixes for PHP 7.0.13, 5.6.28 and PHP 7.2
  - Added 'One variable string' to Radwell report
- **Analysis**
  - New analysis : Object Calisthenics #1, #4
  - New analysis : check that properties are all set at constructor time.
  - New analysis : spot useless checks
  - Updated UndefinedParentMP to take PHP ext classes into account
  - Upgraded 'array_merge in loops' with file_put_contents
  - Upgraded 'useless parenthesis' with math operations
  - Upgraded analysis : 1666 / 1682 test pass (99% pass).
  - Added debug Query method to analysis
- **Tokenizer**
  - Fixed Files to compile first, then count tokens
  - Find Ext Lib handle UT classes better
  - Added limit to 'code' before loading into database. There is a 2M limit.
  - Fixed edge case with nested foreach()
  - Fixed segmentation fault when getting tokens from a script with wrong encoding

Version 0.8.8 (Apricot Immortal, 2016-11-07)

- **Architecture**
  - Added concurency test to avoid running several instance at the same time
  - Report error when it happens with git clone
  - Added UT classes to external libraries
  - Dump is now hidden until finished.
  - Better detection of java and composer (Thanks Julien)
- **Report**
  - New report : Radwell
  - New report : PhpConfiguration helping with configure and php.ini
  - Ambassador : Fixed dashboard values

- **Analysis**
  - New analysis : time() vs strtotime('now')
  - New analysis : useless casting
  - New analysis : No Isset() with Empty()
  - New analysis : don't echo errors
  - New analysis : ext/rar
  - New analysis : use Class::class when possible
  - Added array_key_exists() to slow functions list.
  - Upgraded UpperCaseKeywords to handle partial uppercase
  - Added reported directives for ext/filter
  - Upgraded 'Variables used once' to exclude $this and arguments
  - Upgraded Unreachable Code with break/continue;
  - Multiple Identical Keys now handles null, boolean, real.
  - Upgraded analysis : 1652 / 1668 test pass (99% pass).
- **Tokenizer**
  - Now spots true, false, null as Boolean and Null
  - Removed 'xargs too many arguments' error on Linux

Version 0.8.7 (Naked Demon, 2016-10-31)

- **Architecture**
  - Upgraded Boolean and Integer to report results without storing them in graph
- **Analysis**
  - New analysis : modernizable empty() calls
  - New analysis : recommend Positive conditions
  - New analysis : drop else after return
  - Upgraded analysis : unreacheable code handles if/then with returns.
  - Added tests for Boolean and Null
  - More not Hashes dict.
  - Upgraded analysis : 1637 / 1650 test pass (99% pass).
- **Tokenizer**
  - Fixed line number of <?=
  - Fixed token on arguments

Version 0.8.6 (Fuyun Sou, 2016-10-24)

- **Architecture**
  - New command to ping a queue
  - More documentation
- **Report**

  - Ambassador report sped up multiple times
  - Text, Json and XML all report only analysis (not the dependencies)
- **Analysis**
  - New analysis : suggest ternary instead of Ifthen
  - New analysis : check for returned value usage
  - Added support for PHP 7.0.12 and 5.6.27
  - Added more bugs fixing from extensions
  - Fixed analysis for Zend Framework 1
  - Ignore $this in variable used once
  - Fixed report with unlimited arguments functions
  - Overwritten literals : Ignore assignations in for()
  - Upgraded old PHP 5.* analysis to Gremlin 3
  - Upgraded analysis : 1639 / 1645 test pass (99% pass).
- **Tokenizer**
  - Fixed precedence between require and .
  - Better fullcode for <?=

Version 0.8.5 (Naked Demon, 2016-10-17)

- **Architecture**
  - Moved all classes under Exakat folder for clean hierarchy
- **Report**
  - Ambassador : restored line number in display
- **Analysis**
  - New analysis, check for substr() comparisons with literals
  - New analysis, suggest boolean cast, instead of Ternary.
  - New analysis, spot 3 levels of if/then
  - Upgraded 'hardcoded password', for kadm5 and hash_* functions
  - Upgraded 'external libs', with Zend Framework
  - Upgraded analysis : 1625 / 1638 test pass (99% pass).

Version 0.8.4 (Lingkongzi, 2016-10-10)

- **Architecture**
  - Moved Tasks into ExkatTasks
  - Fixed findExternalLibs
- **Report**
  - Ambassador report got good annex, fixed settings and faceted search
  - Omit clearPHP if not present in docs
- **Analysis**

  - New analysis : detect multiple identical traits/interface in CIT
  - New analysis : suggest creating aliases to reduce code
  - New analysis : spot aliases that may be reused again
  - New analysis : hidden use, that are not at the beginning of the code
  - Upgraded analysis : 1607 / 1618 test pass (99% pass).
  - More documentations to many analysis
  - HasMagicProperty report all magic methods
  - Upgraded 'Useless Parenthesis' with more situations
  - Upgraded 'Unchecked resources' with 2 more situations
  - Fixed several analysis when using Boolean and Null as a class
  - Fixed analysisIsNot with arrays
  - Removed include-like from undefined functions
  - Arrays/AmbiguousKeys : Extended to arrays calls
- **Tokenizer**
  - Fixed edge case with return ?>
  - Fixed path for reporting

Version 0.8.3 (Guzhi Gong, 2016-10-03)

- **Architecture**
  - Created temp folder .exakat in projects_dir
  - Removed mentions of float, only using Real
  - Moved Config to ExakatConfig
  - More examples in docs
- **Report**
  - Added settings and files to Ambassador
- **Analysis**
  - New analysis for dependant Traits
  - Added new Theme 'Cakephp' with 6 analysis for migration
  - New values for Not-a-hash
  - Unresolved Catch now takes Throwable into account
- **Tokenizer**
  - Fixed edge case where return is used inside if/then without { } nor value.
  - Fixed 'code' and 'token' for ?: and ()

Version 0.8.2 (Jinjie Shiba Gong, 2016-09-26)

- **Architecture**
  - More examples in docs
  - Fixed 'file' in results

- **Report**
  - Added more media for Ambassador
- **Analysis**
  - New analysis for count/strlen compared to 0
  - Upgraded analysis : 1563 / 1579 test pass (99% pass).
  - Backported all 4 Wordpress analysis (wpdb, nonce usage)
  - Added new Wordpress analysis : variable escaping in templates
- **Tokenizer**
  - Fixed <?= so it is handled like echo

Version 0.8.1 (Babo'erben, 2016-09-19)

- **Architecture**
  - Added main Try/Catch
- **Report**
  - Added 'Ambassador' report.
- **Analysis**
  - Upgraded analysis : 1540 / 1561 test pass (99% pass).
  - More documentation (examples, glossary)
  - Added a list of stopwords for No Hardcoded Hash
  - Upgraded analysis 'No Hardcoded Path' with protocols and glob with wildcards
  - Upgraded analysis 'No Hardcoded Hash' with stopwords
  - Added new Analysis for portability : spot common Linux files
  - Added new Analysis : use system temp dir, not hardcoded one
  - New analysis that spot unused protected methods
  - Added Time-to-fix and severity to all analysis
- **Tokenizer**
  - Fixed edge case with if/then and try/catch
  - Synchronized constants in Tokens/Consts*.php
  - Added support for PHP 7.2

Version 0.8.0 (Benbo'erba, 2016-09-12)

- **Architecture**
  - More examples in the docs
  - Better find root in export
- **Report**
  - Prepared code for new report style
- **Analysis**
  - New analysis : no throw in __destruct

  - New analysis : spot empty blocks in control structures
  - Update : Check parse_str and mb_parse_str()
  - Upgraded analysis : 1524 / 1540 test pass (99% pass).
- **Tokenizer**
  - Fixed representation of [] and [index] with static properties

Version 0.7.10 (Nine Headed Bug, 2016-09-05)

- **Architecture**
  - Added optional dependency to mbstring in Doctor
  -
- **Analysis**
  - Added analysis for PHP 7.1 features
  - Upgraded analysis : 1377 / 1510 test pass (91% pass).
- **Tokenizer**
  - Removed parasit 'void' added in sequences.
  - Raised export max depth to 15.
  - Fixed FQN for new without parenthesis
  - Fixed support for PHP 5.5/5.6.
  - Added support for iterable
  - Checked support for extensions and ignore dirs

Version 0.7.9 (Wansheng Princess, 2016-08-29)

- **Architecture**
  - Added several features at Loading time : mark global variables in $GLOBALS, fallback FQN in functions, link constant to definitions.
- **Analysis**
  - Added analysis for impossible comparisons (count($a) < or >= 0)
  - Added analysis for PHP 7.1 : removed directives, added functions
  - Upgraded analysis : 1485 / 1522 test pass (97.5% pass).
- **Tokenizer**
  - Fixed edge case with <?= $v;
  - Fixed priorities between include and .
  - Better support of trait in classes

Version 0.7.8 (Wansheng Dragon King, 2016-08-22)

- **Architecture**
  - Prepared databases for PHP 7.2
- **Analysis**
  - Reports that preg_match results are not checked

  - Report List short syntax usage.
  - Upgraded analysis : 1224 / 1493 test pass.

- **Tokenizer**
  -

Version 0.7.7 (Water Repelling Golden Crystal Beast, 2016-08-17)

- **Analysis**
  - Upgraded Bug database to handle PHP 7.0.10, 5.6.24 and 5.5.38

Version 0.7.5 (Jade Faced Princess, 2016-07-19)

- **Architecture**
  - Added 'anonymize' command, that anonymize files and projects
- **Analysis**
  - new analysis : recommend preg_replace_callback_array() when there are several call to preg_replace_callback_array()
  - Upgraded analysis : 1103 / 1464 test pass.
- **Tokenizer**
  - Lots of fixes for stability : tested on 28M tokens

Version 0.7.4 (Great Sage Who Pacifies Heaven, 2016-07-12)

- **Architecture**
  - Entirely rewrote the 'Tokenizer' part
  - Upgraded database schema
- **Analysis**
  - Upgraded analysis : 1027 / 1461 test pass.
- **Tokenizer**
  - Entirely rewrote the 'Tokenizer' part
  - 1851 UT pass correctly (extra 51)

Version 0.6.7 (Red boy, 2016-05-30)

- **Report**
  - Added List With Keys in Appinfo()
  - Added by-reference functions mention
  - Now reporting good visibility/static for __callstatic
  - Added bug info for PHP 7.0.7, 5.5.36, 5.6.21
- **Analysis**
  - New : recommend instanceof over is_object()
  - Fixed several ignored limitations, due to case : $phpversion
- **Tokenizer**
  - Fixed 'originclass' in namespaced use

Version 0.6.6 (Princess Iron Fan, 2016-05-23)

- **Report**
  - New report, suggest disable_functions directive value.
  - Added support for memcached directives
- **Analysis**
  - New analysis : spot throw without new
  - New analysis : suggest adding 2nd parameter to unserialize in PHP 7.0+
  - New analysis : spot successive if/then with the same condition
  - Added support for zendoptimizer and suhosin extensions
  - PHP7 indirect expression : added support for {} in properties
- **Tokenizer**
  - Raised cycle count, to speed up building AST for large projects

Version 0.6.5 (Great Sage Who Pacifies Heaven, 2016-05-16)

- **Analysis**
  - New analysis : spot globals that may be turned into property
  - New analysis : check that ZF1 classes are well located
  - Upgraded 'dangling foreach reference' to support key=>value
  - Better support for PHP 7 indirect expression
  - More directives for xdebug
  - Eval Without Try is PHP 7 only
  - No Choice analysis is now case insensitive
- **Tokenizer**
  - Added support for keys in list() (PHP 7.1)
  - Added support for constant visibility (PHP 7.2)
  - Added support for Multi catch : catch(A|B $e) (PHP 7.1)
  - Fixed bug with + and instanceof
  - Fixed precedence between :: and ??

Version 0.6.4 (Bull Demon King, 2016-05-09)

- **Architecture**
  - Externalized the list of recognized libraries to Json
  - Added 'Wordpress' and 'Coding convention' as Recipes
- **Report**
  - Initial report for Zend Framework. Still prototyping.
- **Analysis**
  - Accelerated analysis for Implicit GLobals variables
  - New analyze : Indirect Injections (Security)

  - New analyze : Should Use Coalesce (code upgrade)
  - New analyze : Suggest dirname(__FILE__) => __DIR__
  - Added 'str_rot13' as unsafe 'crypto'
  - Properties without default can't be redefined
  - Added Yield and Yield From as structures without parenthesis needs
  - Double Assignation, unless 2nd call is a functioncall (less false positives)

Version 0.6.3 (Jade Faced Princess, 2016-05-02)

- **Architecture**
  - Removed several useless pieces of code (self analysis)
  - Added documentation for Wordpress Recipes
  - Lengthened Cycle for tokenizer
- **Report**
  - Added bugfixes for PHP 7.0.6, 5.6.21, 5.5.35.
  - Now reporting token counts per files
- **Analysis**
  - New analysis : Spot variable that holds $_GET, $_POST, $_REQUEST or $_COOKIE values (internal)
  - New analysis : Report variables that are overwritten by themselves
  - New analysis : Report useless switch (empty, 1 case only)
  - Upgraded NoChoice to handle larger sequences
  - Upgraded Useless Global to handle global $x / $GLOBALS['x'] situations
  - New analysis : Wordpress Recipe : Unverified Nonce, Best Usage for $wpdb
  - New analysis : Void for PHP 7.1
- **Tokenizer**
  - Fixed but with Typehint
  - Added phppowerpoint class in external libraries

Version 0.6.2 (Long Armed Ape Monkey, 2016-04-25)

- **Architecture**
  - Fixed phar detection (based on ext/phar)
  - Cleaned code with myself
- **Report**
  - New report format : clustergrammer
- **Analysis**
  - New analysis : same conditions in If / Then
  - New analysis : spot dead code in catch expressions
  - Static loops now exclude methods usage

  - Indirect variable expression are stricter
  - preg_* Option e has better support for delimiters
  - Upgraded Direct Injection in case of concatenation
  - Detect Ellipsis when counting arguments
  - Could use short assignation : avoid $a += $a + 3;
- **Tokenizer**
  - Sped up Typehint detection
  - No indexing for T_STRING in properties
  - Reduced errors from token_get_all()

Version 0.6.1 (Red Bottomed Horse Monkey, 2016-04-18)

- **Architecture**
  - Prepared to support PHP 7.1
  - Fixed bug in user / passwords when initing the project
  - Better support for ::class when searching for libraries
- **Analysis**
  - UselessParenthesis : spot nested parenthesis
  - Spot exceptions that are thrown but uncaught by the current code
  - Support for ext/lua,
  - New : Check catch order in try/catch
  - Better identification of Composer classes, based on composer.json
  - Now spot interfaces in use declarations (less undefined interfaces)
- **Tokenizer**
  - Added support for PHP 7.1
  - key => value in list() calls
  - visibility for constants in Classes and Interfaces
  - Accelerated up Typehint support

Version 0.6.0 (Intelligent Stone Monkey, 2016-04-11)

- **Architecture**
  - Fixed a bug in Find external libraries
  - Applied fixed based on new analysis audit
  - Fixed a bug that prevented results to be prepared for report (Thanks Philippe G.)
- **Report**
  - Now reports reason for excluding a file from analysis
- **Analysis**
  - New analysis : Logical Mistake (first version),
  - New analysis : Iffectations (code restoration)

  - New analysis : Common alternatives
  - New analysis : No Choice (No alternatives)
  - New analysis : Random_* Without Try (security risk)
  - New analysis : Unknown PCRE options
  - New analysis : Identical conditions
  - New analysis : Hardcoded hashes
  - Upgrade List with appends with variable name
  - Upgrade /e option detection
  - Fixed detection of unused use, with long namespaces.
  - Added finfo to ext/finfo
  - Finds exceptions that are reserved for later throwing
  - Exclude anonymous classes from Already Defined Interface
- **Tokenizer**
  - Extended cycle number to speed up tokenizer.
  - Better escaping of file names

Version 0.5.9 (Six Eared Macaque, 2016-04-04)

- **Architecture**
  - One progressbar per Recipe during project analysis
  - report's documentation
  - Upgraded 'External Lib' to ignore Composer folders.
  - Fixed a bug about interpreting tokens
  - Dump collects classes, interfaces, traits definitions
  - Now storing project name in database for future use
  - Removed PHP configuration modifications (error_reporting, display_errors)
- **Report**
  - Added 'Uml' report : hierarchy report
  - Now reports Pear Usage
  - Upgraded Bugfix database for 7.0.5, 5.6.20 and 5.5.34
  - Report Yield (from) usage
  - New external configuration files : bazar, github, docker, openshift
- **Analysis**
  - Added detection for undefined classes in ZF (1.8 to 1.12)
  - New : report undefined Traits
  - Added support for parent/grandparent when checking argument numbers
  - Added support for V8js
- **Tokenizer**

  - Fixed bug in fullnspath for use within trait or class
  - It is possible to reach a property on an array append
  - Fixed AST between PHP 5 and 7 for globals
  - Simplified ++ analysis

Version 0.5.8 (Sun Deity of Mao, 2016-03-28)

- **Architecture**
  - Moved to self::, instead of static::.
  - First UT for command line
  - Sped up phploc. Prepare code for finite states, in Tasks.
  - Prepare for Gremlin3 (moved gremlin calls to class)
  - Reduced shell_exec usage
- **Report**
  - Fixed display bugs in Devoops report
  - Removed double analysis
  - 'Wrong number of arguments' now supports constructors
- **Analysis**
  - Upgraded 'No Hardcoded IP' to handle constants, spot domains
  - Added support for TokyoTyrant
  - New analysis : spot simple regex, and suggest strpos
  - Excluded "$a[b]" from undefined constants
- **Tokenizer**
  - Fixed bug with nested call to echo.
  - Fixed bug where concatenation ends on a 'AS' keyword
  - Added support of Constants in Foreach
  - Fixed multiple bugs in Grouped Use
  - Support for function as 'class' in static calls
  - Comparison accepts powers
  - Added support for empty array short syntax in sequence
  - Support constant with visibility
  - Parenthesis may be the base for Arrays

Version 0.5.7 (Scorpion Demon, 2016-03-21)

- **Architecture**
  - Added support for folders in UT, for tests that requires several files
  - Improved compatibility with PHPunit
  - Moving gremlin_query() to Gremlin2 class
  - Doctor also reports for phar

  - Improved adaptation to PHP and Exakt in server mode
  - Autoload shouldn't die
  - Fixed case when calling Phpexec
  - Upgraded status presentation in server mode
- **Report**
  - More details for Global Variable list
- **Analysis**
  - Now spotting class when it is inside a string
  - Check for $this outside a trait/class
  - Check for ternary/concatenation precedence
  - Spot classes that attempt to extend final
  - Spot set_exception_handler() that may need rework
  - Refined array_merge analysis, in case of nested loops
- **Tokenizer**
  - Yield [from] may be inside an array
  - Refactored for/foreach tokens
  - Added support for a 'Project' node

Version 0.5.6 (Ruler of Women's Country, 2016-03-14)

- **Architecture**
  - Fixed some backward compatibility with PHP 5.4
  - Started revamping 'Status' command
  - Centralized all tokenizations to PhpExec class
  - Removed usage of __DIR__ and __FILE__
- **Analysis**
  - Spot usage of empty() that can't work on PHP 5.4
  - Suggest using random_int instead of rand
  - Upgraded 'No Array_merge in loops' with array_merge_recursive
  - Added support for scalar type hint in Undefined Classes
  - New analysis : Better rand()
- **Tokenizer**
  - Instanceof has lower precedence than comparison

Version 0.5.5 (Immortal Ruyi, 2016-03-07)

- **Architecture**
  - Added default values for all neo4j_* configs
- **Report**
  - Added support for bugfixes in 7.0.4, 5.6.19 and 5.5.33

  - Added support for bugfixes in 7.1.0-dev

- **Analysis**
  - Added support for Typehint in Undeclared Classes
  - Extended 'Multiple Classes in One File' to interfaces and traits
  - Added analysis for truthy and falsy
  - Spot interfaces implemented by parents (Thanks PHP Inspect)
  - Report usage for unsafe Curl options
- **Tokenizer**
  - Fixed emptyString inside a Heredoc
  - Fixed bug where Sign has lower priority than Power

Version 0.5.4 (Nezha, 2016-02-29)

- **Architecture**
  - Removed some shell_exec() to help with portability
  - Clean command now rebuilds an empty datastore
  - Check the availability of php binaries before using
  - Produce report in a hidden folder, then push it
- **Report**
  - Report the list of bug fixes that apply to code
- **Analysis**
  - Help using preg_match_all options
- **Tokenizer**
  - Fixed a bug with reference and instanceof

Version 0.5.3 (Li Jing, 2016-02-22)

- **Architecture**
  - More UT
  - Supports symlinks for neo4j's folder
  - Supports symlinks for 'code' folder in projects
  - Added upgrade command to check for exakat's available versions and upgrade
- **Analysis**
  - Spot CLI scripts
  - Undefined Interfaces avoids self, parent, static
  - Fixed bug in spotting undefined Interface
  - Variable Used Once in a method are not arguments
  - Added support for all structures in Double Assignation

Version 0.5.2 (Single Horned Rhinoceros King, 2016-02-15)

- **Analysis**

  - Fixed functioncall detection with 'empty'
  - Refined 'Buried assignation' analysis
  - Fixed a bug when using definitions (class, trait, interface, functions...)
  - Better support for case-insensitive constants
  -

- **Tokenizer**
  - Fixed bug in use statement
  - Now spots PHP code in files without extension
  - Upgraded support for grouped Use statement
  - namespace may be a valid nsname part
  - Fixed bracket reports in do... while

Version 0.5.1 (King of Spiritual Touch, 2016-02-08)

- **Architecture**
  - Added test in UT to skip incompilable sources
  - Stabilized tokenizer's UT (partial)
- **Report**
  - HTML protection in Devoops format
  - No display of negative stats
  - Added support for directives : wincache, xcache, apc, opcache
  - Added support for eaccelerator and openssl
- **Analysis**
  - New analysis : Spot unknown PHP directive names
  - Fixed Constants/MultipleDefinedConstants
  - Better detection of functioncalls (with List)
  - Better spotting of ini_set arguments
  - Unreachable code now finds die and exit
  - ObjectReference won't report references on scalar types
  - Revamped 'pregOptionE' analysis
  - Cleaned code with too many arguments
  - Removed useless print
  - Better report of eval() usage
  - Revamped 'Dynamic code' report
  - Fixed bug in Case/Default that are empty
  - Avoided sequences of sequences in Case/Default
  - Fixed Detection of classes' usage with extension
- **Tokenizer**

  - Fixed bracket detection on While and DoWhile
  - Detect void in DoWhile
  - Removed useless T_DIE token
  - Fixed fullcode processing for anonymous classes

Version 0.5.0 (Immortal of Antelope Power, 2016-02-01)

- **Architecture**
  - Added support for HTTP API, through 'server' command.
- **Analysis**
  - Fopen modes checked
  - Redefined default, in class's properties
- **Tokenizer**
  - Fixed situation where echo and print used parenthesis (they don't)
  - Fixed rare but with instanceof and concatenation
  - Fixed support of integers in Gremlin
  - Fixed bug in addslashes and and $ protection order
  - Made Assignations more robust (no un-processed tokens)
  - Reduced the number of shell_exec usage => speed up
  - Finished support for relaxed keyword support in classes (PHP 7)

Version 0.4.6 (Immortal of Elk Power, 2016-01-25)

- **Architecture**
  - New installation script with Vagrant and Ansible (Thanks Alexis!)
  - Updated documentation
  - Added a command to remove a project
- **Report**
  - Devoops reports has case-insensitive menu sort
- **Analysis**
  - Spot redefined properties, classes and methods.
  - Spot properties that may be turned private
  - Fixed special case in Wrong Number Of Arguments
  - Fixed 'OnePage' analysis
- **Tokenizer**
  - Finished support for relaxed keywords in classes
  - Sped up tokenizer by keeping counts of tokens in datastore
  - Fixed detection of CakePHP
  - Fixed special case with Labels
  - Fixed rare case with die() within ternary operator

Version 0.4.5 (Immortal of Tiger Power, 2016-01-18)

- **Architecture**
  - Upgraded documentation
  - Default command is 'help'
- **Report**
  - Better version for FacetedJson report
- **Analysis**
  - New analysis that spots wrong type of argument in PHP internal functions
  - Fixed Isset With Constant for PHP 7
  - Fixed a bug that limited query size during analysis (good for bigger projects)
  - Include variadic (...) to Variable Argument Number
- **Tokenizer**
  - Fixed a bug that blocked tokenizer when a analyzed script generated parse errors.
  - Added support for bazar, svn.
  - Fixed a bug in Nsnames at Loading time.

Version 0.4.4 (Crown Prince Mo'ang, 2016-01-11)

- **Architecture**
  - Reviewed OnePage analysis
  - Dump as now an option to select Recipes
  - Dump forces line to be integer
  - Added a task to update a project's code (git only ATM)
- **Report**
  - Better check when opening database for report (more to come)
  - FacetedJson (and Json) report ignore non-unicode lines
  - Added 'search' box to facetedJson
- **Analysis**
  - Switch To Switch suggestions
  - Unused arguments patch for arguments used in methods
  - Unused properties doesn't mistake function static variable
- **Tokenizer**
  - All Nsnames are now build at Loading time
  - Constants may be calld 'const'
  - More relaxed syntax for methods (exit, include, eval...)
  - Foreach may use coalesce
  - Fixed an edge case with Closures in functioncall

Version 0.4.3 (Tuolong, 2015-01-04)

- **Architecture**
  - Copyright year bump
  - Doctor reports memory_limit and php version consistency
  - Switched to rmDirRecursive
- **Report**
  - Removed old style reporting system
- **Analysis**
  - Fixed fileupload and filesystem directives reports
  - Added report of Environment variable usage
  - Added iconv_set_encoding to the list of directive usage
  - Extension analyzes now takes into account namespaces and traits
  - Analysiss all have severity and time to fix
- **Tokenizer**
  -

Version 0.4.2 (Red Boy, 2015-12-22)

- **Architecture**
  - Published documentation on http://exakat.readthedocs.org
  - First version of the faceted report (-format Faceted)
- **Report**
  - First version of the faceted report (-format Faceted)
  - Fixed Dump that actually finishes after some time
- **Analysis**
  - Spot unused arguments
  - Fixed notInInterface() filter
  - Upgraded HtmlEntitiesCall

Version 0.4.1 (Azure Lion, 2015-12-14)

- **Architecture**
  - Rebuild the report system, for speed and versatility.
- **Report**
  - Available format : JSON, Sqlite, XML, Text and HTML (Devoops).
  - Rules are now part of the documentation.
- **Analysis**
  - Upgraded 'Buried assignations'
  - Locally Unused also spots properties without visibility (but with definition)
  - Could be class constant, if the property is used at least once
  - Better detection of files that are Definitions only (fix at Namespace calls)

  - ++ is now correctly reported as isRead and isWritten in Arguments
  - Closure's use($x) are now reported in both context (calling and called)
  - Removed usage of 'back' method, that is blocking at high token counts
- **Tokenizer**
  - Fixed support for {} and {$ } inside strings
  - Fixed bug with Typehint, that prevented compilation
  - Fixed several (rare) edge cases with Sign and Staticproperties.
  - Fixed detection of closing tags

Version 0.4.0 (Lion Lynx Demon, 2015-12-07)

- **Architecture**
  - Made PHP 7.0 the default (moved to 0.4.0)
  - Ran unit tests on PHPunit 5.1
  - Added a background tasks to build report. Will allow for progressive report.
- **Report**
  - Rewrote the report from scratch. Should be finished next iteration.
  - New report is working for XML and Text report.
- **Analysis**
  - Added support for ext/pecl_http
  - Added several classic folders as ignored by default (change this in config.ini)
  - Create a check for functioncall (and not methods)
  - Spots join('', file())
  - Safely ignoring some dynamic calls in undefined functions (Thanks Marc Delisle)
  - Removed ArrayAppend from double assignation
- **Tokenizer**
  - Fixed a bug when class was auto-referenced.
  - Fixed detecting Static properties when they are also arrays.
  - Fixed fatal errors for mal-formed octals

Version 0.3.12 (Nine Tailed Vixen, 2015-11-30)

- **Architecture**
  - ProgressBar is now displayed during Analyze phase.
- **Report**
  - Report list of error messages used in the library
- **Analysis**
  - Omit eval with hardcoded strings
  - Exclude some index from _SERVER from the report (they are safe)
  - Exclude php://* files as hard coded path

  - Report usage of timestamp to calculate duration
  - Spots unused traits
  - Fixed support for big integers
- **Tokenizer**
  - First support for relaxed keywords in classes. More to come.
  - Checked UT on PHP 7 (Soon to become default version)
  - Fixed version detection in Tokenizer
  - Fixed fullnspath in Use expression;

Version 0.3.11 (Hu A'qi, 2015-11-16)

- **Architecture**
  - Report external services files that may be in the repository
- **Report**
  - Report nested dirname calls (may be changed in PHP 7)
- **Analysis**
  - Better spotting of static loops
  - Don't confuse $globals and $GLOBALS
- **Tokenizer**
  - Rewrote support for As in classes.
  - Fixed arguments that were indexed as Void
  - Trimmed code

Version 0.3.10 (Silver Horned King, 2015-11-09)

- **Architecture**
  - Centralized call to cypher.
- **Report**
  - Sped up several analyzes
- **Analysis**
  - Fixed naming bug with reflexion
  - Support class name in arrays, short syntax
  - Report Relay Functions
  - More PHP 7 incompatibilities reports
- **Tokenizer**
  - Support for 7.1 compilation (dev only)
  - Added cakephp to external libraries
  - Fixed parsing bug with static (as property definition)
  - Fixed 'count' in sequences from Function
  - Rewrote Argument detection (when there is no parenthesis)

Version 0.3.9 (Golden Horned King, 2015-11-02 up)

- **Architecture**
  - Cleaned code with Exakat
- **Analysis**
  - Refined report about double assignation
  - Fixed argument counting in Function Definition
  - Better support of array in Locally Used Properties
  - Updated Composer database
- **Tokenizer**
  - Fixed a bug that ignored Blocks
  - Fixed a rare bug with echo and the following arguments

Version 0.3.8 (Baihuaxiu, 2015-10-26)

- **Architecture**
  - Cleaned too many display (they go to log now), leaving commandline empty (or -v)
  - A lot more PHP 7 incompatibilities spotted
- **Report**
  - Added the list of global variables in the projects (if any)
  - Fixed reports for PHP 5.2 (they were ignored)
- **Analysis**
  - Better handling of composer in unresolved classes
  - Spot setlocale with string (PHP 7)
  - Spot string unpacking (PHP 7)
  - Upgraded static method call, to avoid classes of the same family
  - Report eval without try/catch
  - Report preg_replace with /e
  - Fixed report for empty list()
  - Spot hexadecimal in strings
  - Report usort (and co) as incompatibilities between PHP 7 and 5
- **Tokenizer**
  - Fixed edge case with Sign and namespaced function
  - Added xajax, adodb and gacl as common library
  - Fixed arguments in short array syntax
  - Fixed case where [3] was spotted inside a string

Version 0.3.7 (Yellow Robe Demon, 2015-10-19)

- **Architecture**
  - Added and reviewed many UT. More stability.

- **Report**
  - Fixed the report of the actual version of PHP being used.
  - Non-run analysis are not marked with a stethoscope
  - Report now report closures and not the containing method
  - Removed some dashboard that would generate empty links
- **Analysis**
  - Better spot of blocks inside Alternative syntax
  - Speed up method spotting
  - Fixed properties which were mistaken with deep definitions
- **Tokenizer**
  - Fixed fullcode for Typehint
  - Removed Ppp and moved it to Visibility

Version 0.3.6 (White Bone Demon, 2015-10-12)

- **Architecture**
  - Large speed up at Parsing stage, for large projects
  - Added git informations in Doctor
- **Tokenizer**
  - Changed processing for Arguments.
  - Support for more PHP 7 features, including Use Grouping,
  - Fixed support for ~
  - Simplified ::class handling

Version 0.3.5 (Mingyue, 2015-10-06)

- **Architecture**
  - Reported usage of array constants, improving backward compatibility
  - Checked running on PHP 7
- **Report**
  - Added Definition annex
  - Fixed 'version incompatible' report that was mistaken with 'no result'
  - List all directives being modified in the code
  - List more directives that should be set for production.
- **Analysis**
  - Reworked the Themes about compatibility.
  - Added many tests for PHP 7.0 compatibility
  - Sped up UsedMethod analysis
  - Added support for PHP 7 feature : Unicode Escape Sequences, New functions/classes/interfaces, Removed Functions,

- **Tokenizer**
  - Changed processing for Empty PHP code
  - Support Variable Indirection for both PHP 5 and 7 (depends on exec version)
  - Avoid ignoring all code when finding External Libraries
  - Fixed edge cases with declare() when it is conditional.
  - Support for PHP 7's f()()()

Version 0.3.4 (Qingfeng, 2015-09-28 up)

- **Architecture**
  - Added token_limit configuration to avoid running too large project (default is 1 000 000)
  - Several new tools for internal consistency check.
  - Removed support for neo-contrib's gremlin plugin
- **Report**
  - Report libraries that were found and ignored
- **Analysis**
  - Sped up queries that required previous analysis or multiples atoms
  - Spot global keywords inside loops (perf)
  - Better spotting of Composer classes
  - Report double assignations
- **Tokenizer**
  - Added support for Anonymous classes (PHP 7)
  - Fixed namespace manipulations (They weren't lower case)
  - Mark constants as fail back globals or local to the namespace
  - Support Null Coalesce operator (PHP 7)
  - Fixed rare case for empty strings and noDelimiter

Version 0.3.3 (Immortal Zhenyuan, 2015-09-21)

- **Architecture**
  - Removed some shell stderr that leaked to the main script
- **Report**
  - Added the list of used analysis
  - favicon is now used in the report (Devoops)
  - Fixed count report for Else
  - Fixed directive reports for trader, bcmath and ldap.
- **Analysis**
  - Rebuild the composer database
  - Fixed htmlentities analyze
  - Spot usage of 'substr($s, $p, +/- 1)' and recommend '$s[$p]'

- **Tokenizer**
  - Fixed Multiplication with instantiation

Version 0.3.2 (Tiger Vanguard, 2015-09-14)

- **Report**
  - Added link back from analysis to its themes.
- **Analysis**
  - Useless Returns are now Trait compatible
  - Optimized Composer validation
  - Removed IsKnownVendor analyze (replaced by Composer)
  - Spot inconsistent concatenations ("$a b".$c)
- **Tokenizer**
  - Fixed situation where forgotten white spaces didn't have a file
  - Removed DELETE and S_STRING index
  - Fixed compatibility with Debian (shell commands)
  - Added UT for and / && precedence versus =
  - Fixed identification of empty instructions (Functions / Closure have different behaviors)

Version 0.3.1 (Yellow Wind Demon, 2015-09-03)

- **Architecture**
  - Removed usage of Everyman dependencies
  - Added support for Neo4j Authentication
  - Added a JobQueue
  - Cleaned code with exakat itself
- **Report**
  - Added Dump to SQLITE format for custom manipulations of the results
  - Added new collection of rules for Calesthenics (dev)
  - Updated composer database
  - Now reporting found Composer.
- **Analysis**
  - Fixed Compilation spotting
- **Tokenizer**
  - Fixed an edge case with Sign, when used in a concatenation

Version 0.3.0 (Lingxuzi, 2015-Aug-25)

- **Architecture**
  - Moved to Thinkaurelius's gremlin plug-in, Neo4j 2.2.4 and Java 8.
- **Report**
  - Added a view by File

  - Added sorting for results (by file and by analyze)

- **Analysis**
  - Spot functions whose results should be checked before they are used
  - Spot breaks/continue out of a loop
  - Exports all the results in a dump.sqlite file

- **Tokenizer**
  - Fixed a minor bug with ::class (messed up the {} counts)
  - removed dependency to Everyman's Neo4j classes.
  - Added a step that removes big and identifiable libraries in PHP (such as tcpdf, jpgraph, etc..)

Version 0.2.5 (Scholar in a White Robe, 2015-Aug-17)

- **Report**
  - List the files that are ignored in the annex

- **Analysis**
  - Updated Knowledge Database for memcache, aliases, zlib, standard
  - Added more directives to Review
  - Added support for xhprof

- **Tokenizer**
  - Fixed bug with Else (Not-alternative)
  - Fixed Sequence creation with If-Then
  - Yield may be assigned
  - Removed one Tokenizer's operation (filterOut2)
  - Fixed priorities with Concatenation, Multiplication, Additions
  - Process Echo and Print separately
  - Automatically removes common bundled libraries to reduce app size

Version 0.2.4 (Black Wind Demon, 2015-06-22)

- **Analysis**
  - Rebuild the composer database
  - Lots of new extensions supported : ev, libevent, event, php-ast, wikidiff2, proctitle, inotify, ibase, amqp, geoip, output buffering,
  - Report errors when non-variables are returned by reference
  - Marked more analyzes for PHP 7
  - Fixed Unpreprocess structures with split
  - Upgraded spotting for useless parenthesis
  - Added a check ++$i vs $i++;
  - Exclude abstract methods from Variables Used Once
  - Added new directives

  - Also check for ASP Tags

- **Tokenizer**
  - Fixed the fullpath for functions when they are not defined in the code
  - Upgraded support for Return Type (PHP 7.0+)
  - error_reporting with -1 is OK
  - Fixed a precedence problem with & and &&
  - Refactored Ifthen token to support return type
  - Added a kill command when cleaning Database

Version 0.2.3 (Techu Shi, 2015-06-22)

- **Analysis**
  - Report usage of Return Typehint, and Scalar Typehint
  - Report usage of classes that used to return null on new
  - Report useless abstract classes
- **Tokenizer**
  - Upgraded 'init' command, to handle various VCS
  - Added support for Return Typehint

Version 0.2.2 (Xiong Shangjun, 2015-06-16)

- **Analysis**
  - Now spots short assignations
  - More UselessInstructions spotted
  - Ignore Unset as modified values in loops
- **Tokenizer**
  - Added support for PHP7 new tokens (T_SPACESHIP, T_COALESCE, T_YIELD_FROM)
  - Split loading into more .csv files for lighter and more robust queries
  - Better support for arrays [1,2,3] as functioncall (just like array())
  - Process tokens by batches of 800
  - Clean vertex at each queries, not Sequence

Version 0.2.1 (General Yin, 2015-06-02)

- **Analysis**
  - sizeOf may have 2 arguments
  - 2 clearPHP link added in documentation
- **Tokenizer**
  - Fixed bug with Bitshift and Addition
  - Fixed bug with Sequence when merging sequences
  - Fixed bug with String and Addition
  - Fixed Visibility in Use instruction

  - Foreach accepts Constants as Source
  - Fixed special case for nested IfThen

Version 0.2.0 (Demon of Confusion, 2015-05-15)

- First version

CHAPTER 3

with a Bare metal installation

Here are 2 tutorials to run Exakat on your code. You may install exakat with the *projects* folder, and centralize multiple audits in one place, or run exakat in-code, right from the source code. You may also run exakat on a host machine (aka, bare-metal), or as a docker container.

- Bare metal install
- with projects folder
- within the code

All tutorials follow the same steps :

- Project initialisation
- Audit run
- Reports access

3.1 Bare metal install, with projects folder

3.1.1 Installation

Refer to the *Installation* section in the ADMINISTRATOR GUIDE to install Exakat.

3.1.2 Initialization

First, fetch the code to be audited. This has to be done once. Later, the code may be updated.

```
php exakat.phar init -p sculpin -R https://github.com/sculpin/sculpin
```

This command inits the project in the 'projects' folder, with the name 'sculpin', then clone the code with the provided repository. By default, the cloning is done by git.

Exakat requires a copy of the code to run an audit. When accessing via VCS, such as git, mercurial, svn, etc., read-only access is sufficient and recommended. Exakat doesn't write anything in the code, nor stage, commit or push.

More information on options in the _Commands.

3.1.3 Execution

After initialization, you may run an audit :

```
php exakat.phar project -p sculpin
```

This command runs the whole auditing cycle : code loading, code audits and report building. It is ready to work with the initial configuration. The configuration may be adapted later.

Once the run is finished, the reports are place in the folder *projects/sculpin/*. For example, a HTML version is available in *projects/sculpin/report/index.html*. Simply open the 'projects/sculpin/report/index.html' file in a browser.

3.1.4 More reports

Once the 'project' command has been fully run, you may run the 'report' command to create different reports. Usually, 'Diplomat' has the most complete report, and other focused reports are available.

It is possible to create the remaining reports, once an audit has been finished. Here is an example of a Uml report.

```
php exakat.phar report -p sculpin -format Uml -file uml
```

This export the current project in UML format. The file is called 'uml.dot' : dot is added by exakat, as the report has to be opened by graphviz compatible software.

The full list of available reports are in the *Reports* section.

Once it is finished, the reports are in the folder *projects/sculpin/* under different names.

3.1.5 New run

After adding some modifications in the code, commit them in the repository. Then, run :

```
php exakat.phar update  -p sculpin
php exakat.phar project -p sculpin
```

This command updates the repository to the last modification, then runs the whole audit again. If the code is not using a VCS repository, then the update command has no effect on the code. You should update the code manually, by replacing it with a newer version.

Once the audit is finished, the reports are in the same folders as previously : *projects/sculpin/report* (HTML version).

The reports replace any previous report. To keep a report of a previous version, move it away from the current location, or give it another name.

3.2 Bare metal install, within the code

This tutorial runs exakat from the source code repository.

3.2.1 Installation

Refer to the *Installation* section in the ADMINISTRATOR GUIDE to install Exakat.

3.2.2 Initialization

Go to the directory that contains the source code.

Create a configuration file called *.exakat.yml*, with the following content :

```
project: "name"
```

This is the minimum configuration for that file. It is sufficient for this tutorial, and we will produce more reports later. You will read more about _Configuration in the dedicated section.

3.2.3 Execution

After creating the configuration file above, an audit may be run :

```
exakat project
```

This command runs the whole cycle : code loading, code audits and report building. It works without initial configuration.

Once it is finished, the reports are in the current folder. Simply open the 'report/index.html' file in a browser.

3.2.4 More reports

When running exakat inside code, audits must be configured before the run of the audit.

Edit the .exakat.yml file, and update the file with the following lines :

```
project: "name"
project_reports:
  - Uml
  - Plantuml
  - Ambassador
```

Then, run the audit as explained in the previous section.

This configuration produces 3 reports : "Ambassador", which is the default report, "Uml", available in the 'uml.dot' file, and "Plantuml", that may be opened with plantuml.

The full list of available reports are in the 'Command' section.

3.2.5 New run

After some modifications in the code, run again exakat with the same command than the first time. Since the audit is run within the code source, no update operation is needed.

Check the *config.ini* file before running the audit, to check if all the reports you want are configured.

```
exakat project
```

CHAPTER 4

with a Docker installation

Here are 2 tutorials to run Exakat on your code. You may install exakat with the projects folder, and centralize your audits in one place, or run exakat in-code, right from the source code. You may also run exakat with a bare-metal installation, or as a docker container.

- Docker container
- with projects folder
- within the code

All four tutorials offer the same steps : + Project initialisation + Audit run + Reports access

4.1 Docker container, with projects folder

This tutorial runs exakat audits, when source code are organized in the *projects* folder. Any folder will do, since exakat is now hosted in the docker image.

4.1.1 Initialization

Go to the directory that contains the 'projects' folder.

Init the project with the following command :

```
docker run -it --rm -v /Users/famille/Desktop/analyzeG3/projects:/usr/src/exakat/
↪projects exakat/exakat:latest exakat init -p sculpin -R https://github.com/sculpin/
↪sculpin -git
```

This will create a 'projects/sculpin' folder, with various documents and folder. The most important folder being 'code', where the code of the project is fetched, an cached. See _Commands for more details about the *init* command.

4.1.2 Execution

After creating the project, an audit may be run from the same directory:

```
docker run -it --rm -v /Users/famille/Desktop/analyzeG3/projects:/usr/src/exakat/
↪projects exakat/exakat:dev exakat project -p sculpin
```

This command runs the whole cycle : code loading, code audits and report building.

Once it is finished, the report is available in the *projects/sculpin/report/* folder. Open *projects/sculpin/report/index.htmll* with a browser.

4.1.3 More reports

When running exakat with the projects folder, reports may be configured before the run of the audit, in the config.ini file, or in command line, or extracted after the run.

After a first audit, use the *report* command. Here is an example with the *Uml* report.

```
docker run -it --rm -v /Users/famille/Desktop/analyzeG3/projects:/usr/src/exakat/
↪projects exakat/exakat:dev exakat report -p sculpin -format Uml
```

Reports may only be build if the analysis they depend on, were already processed.

In command line, use the *-format* option, multiple times if necessary.

```
docker run -it --rm -v /Users/famille/Desktop/analyzeG3/projects:/usr/src/exakat/
↪projects exakat/exakat:dev exakat project -p sculpin -format Uml
```

In config.ini, edit the *projects/sculpin/report/config.ini* file, and add the following lines :

```
project_reports[] = 'Uml';
project_reports[] = 'Plantuml';
project_reports[] = 'Ambassador';
```

Then, run the audit as explained in the previous section.

The full list of available reports are in the _Reports section.

4.1.4 New run

After adding some modifications to the code and committing them, you need to update the code before running it again : otherwise, it will run on the previous version of the code.

```
docker run -it --rm -v /Users/famille/Desktop/analyzeG3/projects:/usr/src/exakat/
↪projects exakat/exakat:dev exakat update -p sculpin
docker run -it --rm -v /Users/famille/Desktop/analyzeG3/projects:/usr/src/exakat/
↪projects exakat/exakat:dev exakat project -p sculpin
```

4.2 Docker container, within the code folder

This tutorial runs exakat audits from the source code repository, with a docker container.

4.2.1 Installation

Refer to the _Installation section to install Exakat on docker.

4.2.2 Initialization

Go to the directory that contains the source code.

Create a configuration file called *.exakat.yml*, with the following content :

```
project: "name"
```

This is the minimum configuration for that file. You may read more about _Configuration in the dedicated section.

4.2.3 Execution

After creating the configuration file, an audit may be run from the same directory:

```
docker run -it --rm -v $(`pwd`):/src exakat/exakat:latest exakat project
```

This command runs the whole cycle : code loading, code audits and report building. It works without initial configuration.

Once it is finished, the report is displayed on the standard output (aka, the screen).

4.2.4 More reports

When running exakat inside code, reports must be configured before the run of the audit : they will be build immediately.

Edit the .exakat.yml file, and add the following lines :

```
project: "name"
project_reports:
  - Uml
  - Plantuml
  - Ambassador
```

Then, run the audit as explained in the previous section.

This configuration produces 3 reports : “Ambassador”, which is the default report, “Uml”, available in the ‘uml.dot’ file, and “Plantuml”, that may be opened with plantuml.

The full list of available reports are in the _Reports section.

4.2.5 New run

After adding some modifications to the code, run again exakat with the same command than the first time. Since the audit is run within the code source, no explicit update operation is needed.

Check the *.exakat.yml* file before running the audit, to check if all the reports you want are configured.

```
docker run -it --rm -w /src -v $(pwd):/src --entrypoint "/usr/src/exakat/exakat.phar"␣
↪exakat/exakat:latest project
```

CHAPTER 5

Overview

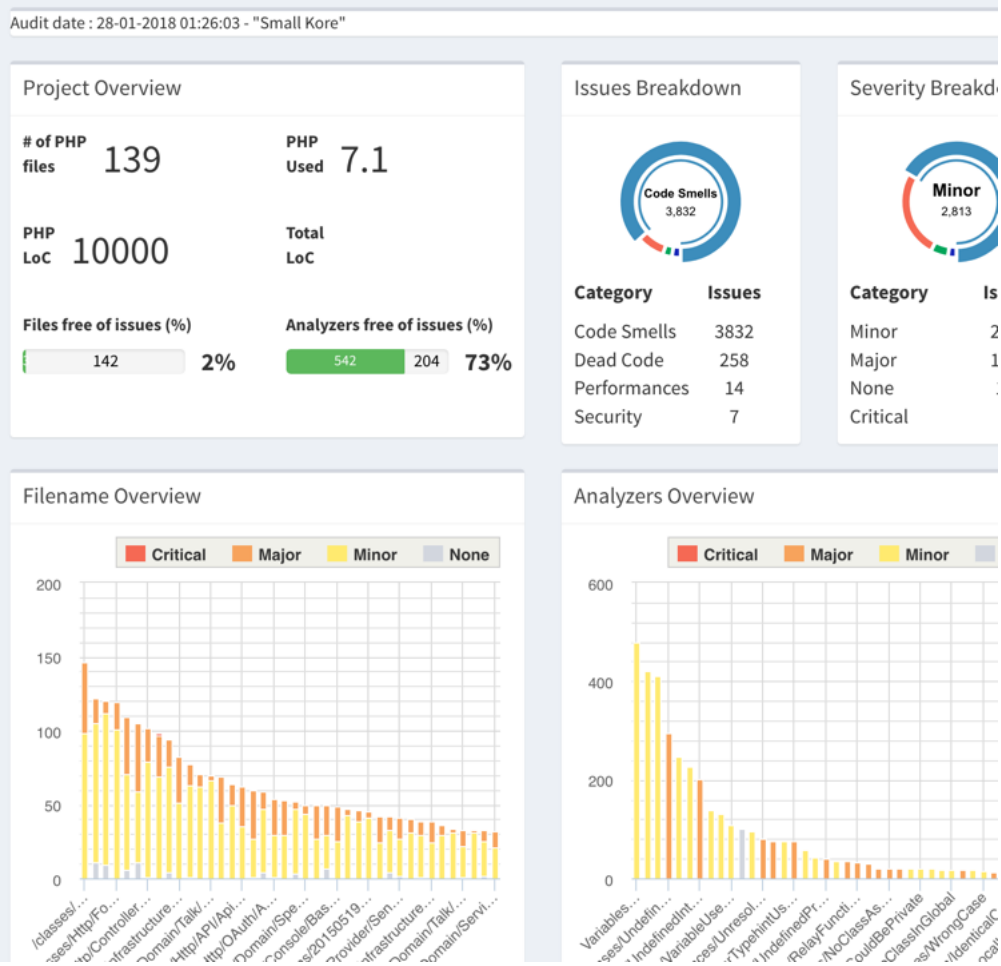
5.1 Summary

- *1371 analyzers*
- *Compatible with PHP 5.2 to 8.0*
- *Migration guide from 5.2 to 8.0*
- *Modernize your code*
- *Detect code smells or bugs that impact the code*
- *appinfo(): the list of PHP features*
- *List of significant PHP directives*
- *Framework and application support*
- *Hierarchy Diagrams*
- *Code visualizations*

5.2 1371 analyzers

There are currently 1371 different analyzers that check the PHP code to report code smells. Analyzers are inspired by PHP manual, migration documents, community good practices, computer science or simple logic.

Some of them track rare occurrences, and some are frequent. Some track careless mistakes and some are highly complex situations. In any case, exakat has your back, and will warn you.



5.3 Compatible with PHP 5.2 to 8.0

The Exakat engine audits code with PHP versions that range from PHP 5.2 to PHP 8.0-dev.

The Exakat engine itself runs on PHP 7.x+ and is regularly checked on those versions. It is possible to run Exakat on 7.2 and audit a code with PHP 5.6.

5.4 Migration guide from 5.2 to 8.0

Every middle version of PHP comes with its migration guide from the manual, and from community's feedback. Incompatibilities are included as analyzers in Exakat, and report everything they can find that may prevent you from moving to the newer version.

Although they won't catch it all, they do reduce the amount of unexpected surprises by a lot.

Version	Name	7.3	7.2	7.1	7.0	5.6	5.5	5.4	!
	Compilation		⚠	⚠	⚠	⚠	⚠		
5.4+	Methodcall On New	✓	✓	✓	✓	✓	✓	✓	⚠
5.5+	Cant Use Return Value In Write Context	✓	✓	✓	✓	✓	✓	⚠	⚠
5.5+	::class	✓	✓	✓	✓	✓	✓	⚠	⚠
5.5+	Empty With Expression	✓	✓	✓	✓	✓	✓	⚠	⚠
5.6+	Constant Scalar Expressions	✓	✓	✓	✓	✓	⚠	⚠	⚠
7.0-	Abstract Static Methods	⚠	⚠	⚠	⚠	✓	✓	✓	✓
7.0-	Null On New	⚠	⚠	⚠	⚠	✓	✓	✓	✓
7.0-	ext/apc	⚠	⚠	⚠	⚠	✓	✓	✓	✓
7.0-	ext/mysql	⚠	⚠	⚠	⚠	✓	✓	✓	✓
7.0-	Reserved Keywords In PHP 7	⚠	⚠	⚠	⚠	✓	✓	✓	✓
7.0+	Parenthesis As Parameter	✓	✓	✓	✓	⚠	⚠	⚠	⚠
7.1-	New Functions In PHP 7.1	⚠	⚠	⚠	✓	✓	✓	✓	✓
7.2-	PHP 7.2 Deprecations	⚠	⚠	✓	✓	✓	✓	✓	✓
7.2-	New Functions In PHP 7.2	⚠	⚠	✓	✓	✓	✓	✓	✓
7.2-	PHP 7.2 Removed Functions	⚠	⚠	✓	✓	✓	✓	✓	✓
5.4+	Binary Glossary								
5.5+	Const With Array								
5.5+	Use password_hash()								

5.5 Modernize your code

Migrations are too often considered over when incompatibilities are removed. In fact, the best is still to come : using the new features. Or, using the new features from previous versions, that were forgotten. Exakat dedicates a whole category of suggestions to modern PHP features that should be used now.

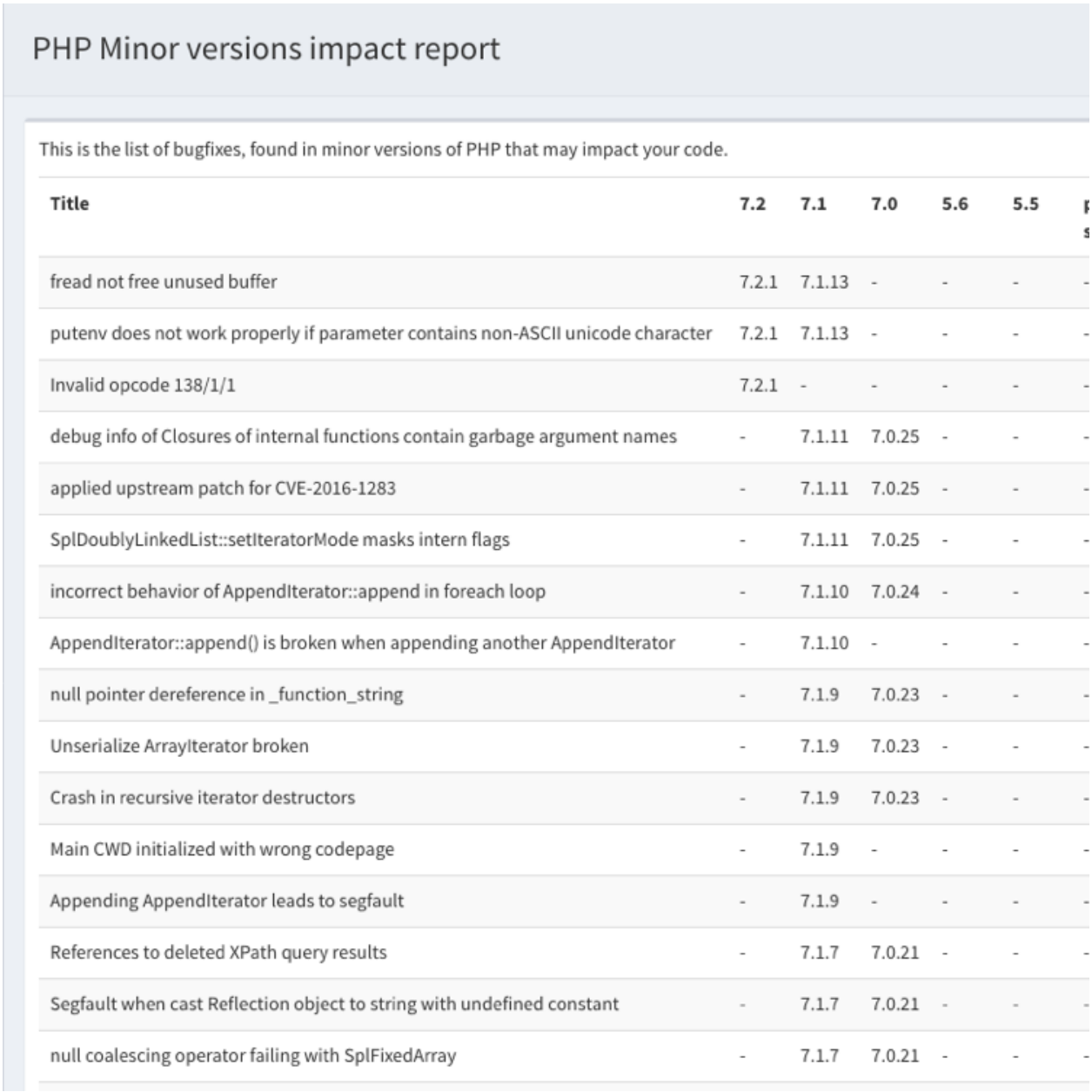
Visibility recommendations

Name	Value	None (public)	Public	Protected	Private	Consta
class AuthUser						
STATUS_DEL	-1	★		★	★	
STATUS_NORMAL	1	★		★	★	
IS_SUPER_NO	0	★				
IS_SUPER_YES	1	★				
public static function tableName( ) { /**/ }		★	★			
public function rules( ) { /**/ }		★	★	★		
public function attributeLabels( ) { /**/ }		★				
public static function findByUsername($username) { /**/ }		★	★	★		
public function validatePassword($password) { /**/ }		★				
public function setPassword($password) { /**/ }		★				
public static function findIdentity($id) { /**/ }		★	★	★		
public static function findIdentityByAccessToken($token) { /**/ }		★	★	★		
public function getId( ) { /**/ }		★				

5.6 Detect code smells or bugs that impact the code

Every minor version of PHP comes with bug fixes and modifications at the function level. Some special situations are better handled, and that may have impact in your code. Every modified function, class, trait or interface that is also found in your code is reported here, giving a good overview of the impact of every minor version.

Safe bet : keep up to date!



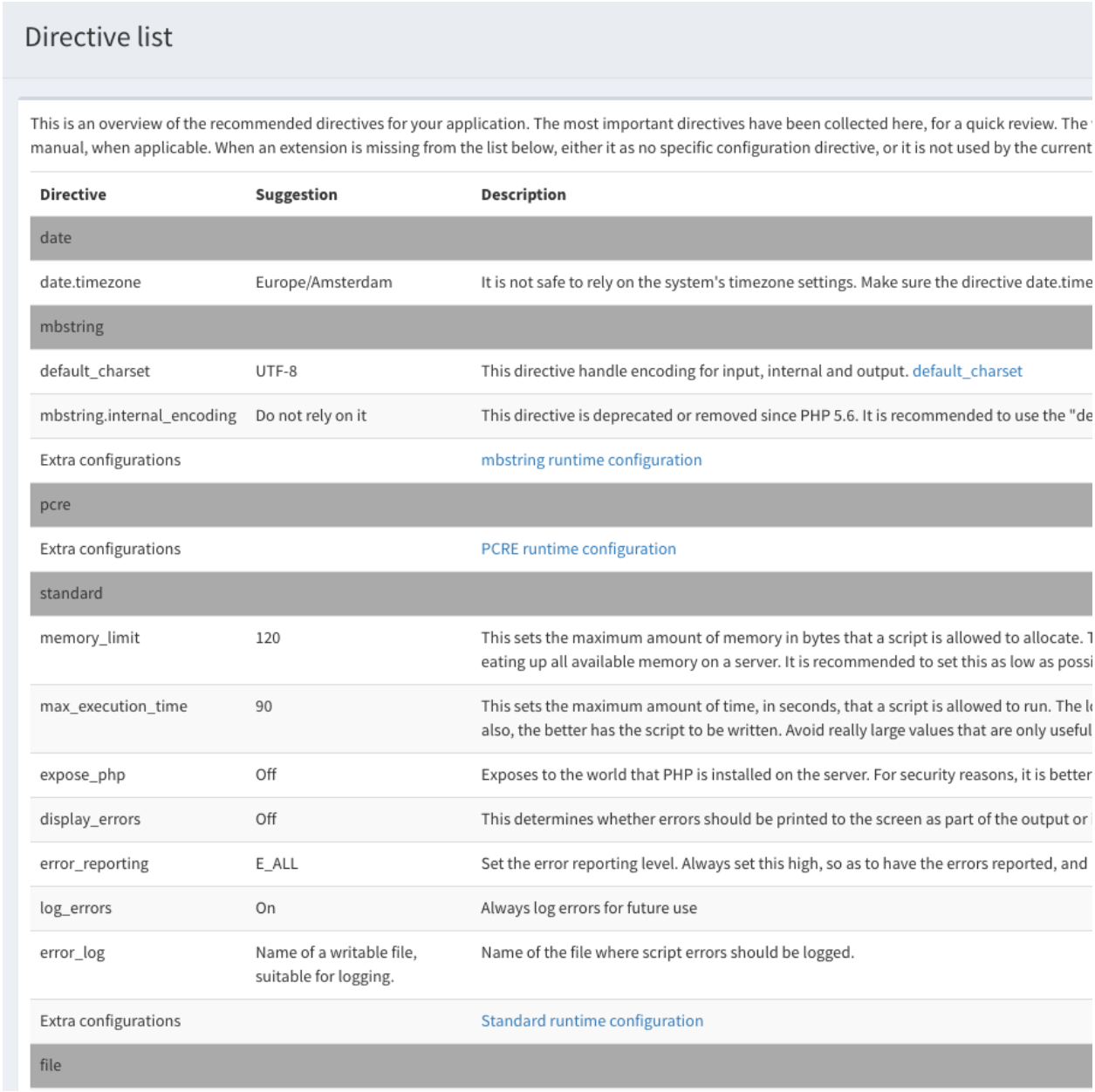
PHP Minor versions impact report

This is the list of bugfixes, found in minor versions of PHP that may impact your code.

Title	7.2	7.1	7.0	5.6	5.5	p s
fread not free unused buffer	7.2.1	7.1.13	-	-	-	-
putenv does not work properly if parameter contains non-ASCII unicode character	7.2.1	7.1.13	-	-	-	-
Invalid opcode 138/1/1	7.2.1	-	-	-	-	-
debug info of Closures of internal functions contain garbage argument names	-	7.1.11	7.0.25	-	-	-
applied upstream patch for CVE-2016-1283	-	7.1.11	7.0.25	-	-	-
SplDoublyLinkedList::setIteratorMode masks intern flags	-	7.1.11	7.0.25	-	-	-
incorrect behavior of AppendIterator::append in foreach loop	-	7.1.10	7.0.24	-	-	-
AppendIterator::append() is broken when appending another AppendIterator	-	7.1.10	-	-	-	-
null pointer dereference in _function_string	-	7.1.9	7.0.23	-	-	-
Unserialize ArrayIterator broken	-	7.1.9	7.0.23	-	-	-
Crash in recursive iterator destructors	-	7.1.9	7.0.23	-	-	-
Main CWD initialized with wrong codepage	-	7.1.9	-	-	-	-
Appending AppendIterator leads to segfault	-	7.1.9	-	-	-	-
References to deleted XPath query results	-	7.1.7	7.0.21	-	-	-
Segfault when cast Reflection object to string with undefined constant	-	7.1.7	7.0.21	-	-	-
null coalescing operator failing with SplFixedArray	-	7.1.7	7.0.21	-	-	-

5.7 appinfo(): the list of PHP features

Do you know the PHP features that your application rely upon ? Recursivité, reflexion, backticks or anonymous classes ? Exakat collect all those features, and sum them up in one nice table, so you know all of it.



Directive list

This is an overview of the recommended directives for your application. The most important directives have been collected here, for a quick review. The manual, when applicable. When an extension is missing from the list below, either it as no specific configuration directive, or it is not used by the current

Directive	Suggestion	Description
date		
date.timezone	Europe/Amsterdam	It is not safe to rely on the system's timezone settings. Make sure the directive date.time
mbstring		
default_charset	UTF-8	This directive handle encoding for input, internal and output. default_charset
mbstring.internal_encoding	Do not rely on it	This directive is deprecated or removed since PHP 5.6. It is recommended to use the "de
Extra configurations		mbstring runtime configuration
pcre		
Extra configurations		PCRE runtime configuration
standard		
memory_limit	120	This sets the maximum amount of memory in bytes that a script is allowed to allocate. T eating up all available memory on a server. It is recommended to set this as low as possi
max_execution_time	90	This sets the maximum amount of time, in seconds, that a script is allowed to run. The l also, the better has the script to be written. Avoid really large values that are only useful
expose_php	Off	Exposes to the world that PHP is installed on the server. For security reasons, it is better
display_errors	Off	This determines whether errors should be printed to the screen as part of the output or
error_reporting	E_ALL	Set the error reporting level. Always set this high, so as to have the errors reported, and
log_errors	On	Always log errors for future use
error_log	Name of a writable file, suitable for logging.	Name of the file where script errors should be logged.
Extra configurations		Standard runtime configuration
file		

5.8 List of significant PHP directives

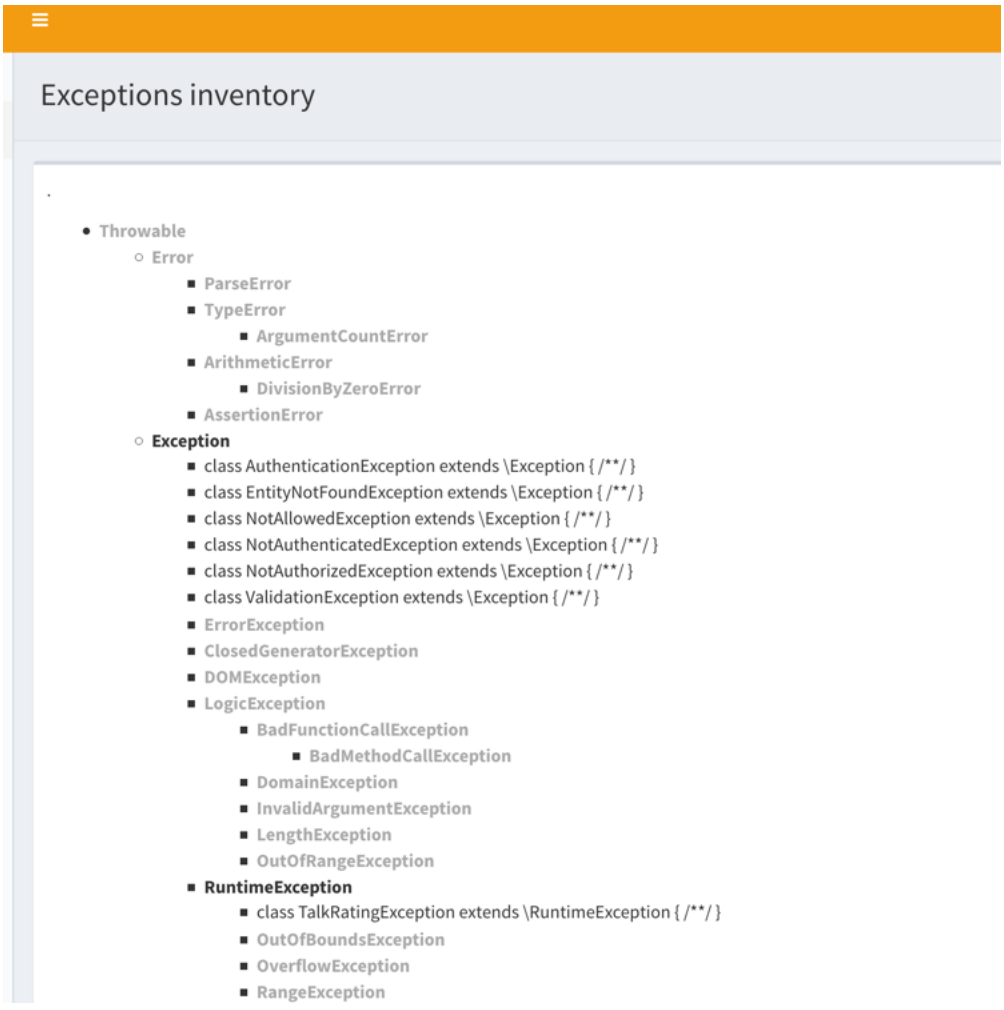
Exakat recommends which PHP directives to check while preparing your code for production. If 'memory_limit' is an ever green, may be 'post_max_size' (linked to file_upload), or assertions shouldn't be forgotten. Based on feature and extension usage, it also list the most important directives, and leads you to the full manual list, in case you want to fine tune it to the max. Use it as a reminder.

5.9 Framework and application support

Exakat provides support for framework and application specific rules. Supported frameworks includes Cakephp, Codeigniter, Drupal, Laravel, Melis, Slim, Symfony, Wordpress and Zend Framework

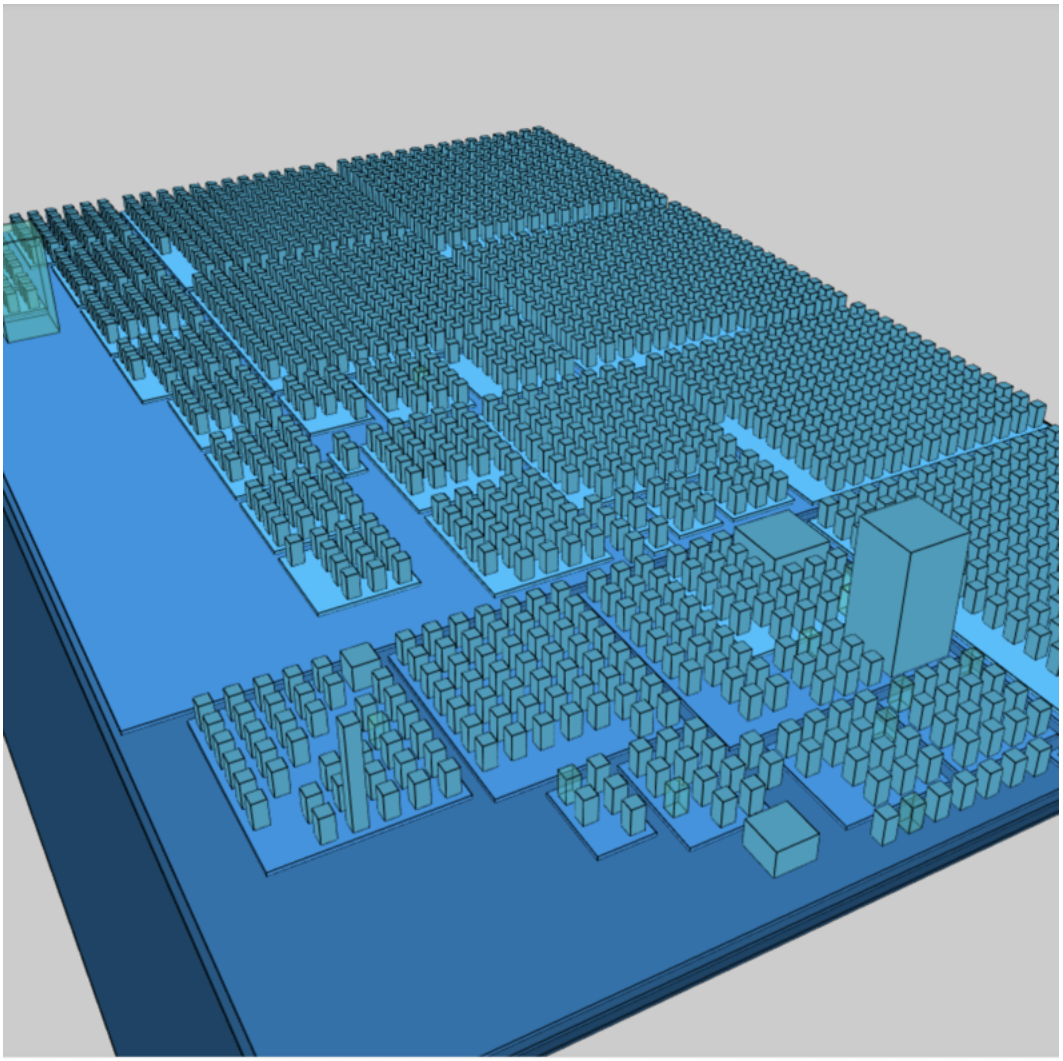
5.10 Hierarchy Diagrams

Exakat documents the code automatically with several diagrams, such as : * UML class diagramm, based on inheritance (classes), usage (traits) and implementations (interfaces), grouped by namespaces. * The Exceptions tree * The traits tree and the trait matrix



5.11 Code visualizations

Exakat documents the code automatically with several diagrams, such as : a full UML class diagramm, based on inheritance (classes), usage (traits) and implementations (interfaces), grouped by namespaces.



CHAPTER 6

PHP Version

6.1 Compatible with PHP 5.2 to 8.0-dev

The Exakat engine audits code with PHP versions that range from PHP 5.2 to PHP 8.0-dev.

The Exakat engine itself runs on PHP 7.x+ and is regularly checked on those versions. It is possible to run Exakat on 7.2 and audit a code with PHP 5.6.

CHAPTER 7

Library & Framework Support

7.1 Summary

- Supported Rulesets
- Supported Reports
- Supported PHP Extensions
- Applications
- Recognized Libraries
- New analyzers
- External services
- PHP Error messages
- Exakat Changelog

7.2 External Library Support

Libraries that are popular, large and often included in repositories are identified early in the analysis process, and ignored. This prevents Exakat to analysis some code foreign to the current repository : it prevents false positives from this code, and make the analysis much lighter. The whole process is entirely automatic.

Those libraries, or even some of the, may be included again in the analysis by commenting the ignored_dir[] line, in the projects/<project>/config.ini file.

- ADOdb
- atoum
- BBQ
- CakePHP

- CI xmlRPC
- CPDF
- Codeception
- DomPDF
- FPDF
- phpGACL
- gettext Reader
- jpGraph
- HTML2PDF
- HTML Purifier
- http_class
- IDNA convert
- lessc
- magpieRSS
- MarkDown Parser
- Markdown
- mpdf
- oauthToken
- passwordHash
- pChart
- pclZip
- Propel
- phpExecl
- phpMailer
- PHPSpec
- PHPUnit
- qrCode
- Services_JSON
- sfYaml
- SimplePie
- SimpleTest
- swift
- Smarty
- Symfony Unit Test
- tcpdf
- text_diff

- text highlighter
- tfpdf
- Typo3TestingFramework
- UTF8
- Xajax
- Yii
- Zend Framework

7.3 External Services Support

List of external services whose configuration files has been commited in the code.

- Apache - .htaccess, htaccess.txt
- Apple - .DS_Store
- appveyor - appveyor.yml, .appveyor.yml
- ant - build.xml
- apigen - apigen.yml, apigen.neon
- arcunit - .arcunit
- artisan - artisan
- atoum - .bootstrap.atoum.php, .atoum.php, .atoum.bootstrap.php
- arcanist - .arclint, .arcconfig
- bazaar - .bzr
- babeljs - .babel.rc, .babel.js, .babelrc
- behat - behat.yml.dist, behat.yml
- box2 - box.json, box.json.dist
- bower - bower.json, .bowerrc
- circleCI - circle.yml, .circleci
- codacy - .codacy.json
- codeception - codeception.yml, codeception.dist.yml
- codecov - .codecov.yml, codecov.yml
- codeclimate - .codeclimate.yml
- composer - composer.json, composer.lock, vendor
- couscous - couscous.yml
- Code Sniffer - .php_cs, .php_cs.dist, .phpcs.xml, php_cs.dist, phpcs.xml, phpcs.xml.dist
- coveralls - .coveralls.yml
- crowdin - crowdin.yml
- cvs - CVS

- docker - .dockerignore, .docker, docker-compose.yml, Dockerfile
- dotenv - .env.dist, .env, .env.example
- drone - .dockerignore, .docker
- drupalci - drupalci.yml
- drush - drush.services.yml
- editorconfig - .editorconfig
- eslint - .eslintrc, .eslintignore, eslintrc.js, .eslintrc.js, .eslintrc.json
- Exakat - .exakat.yaml, .exakat.yml, .exakat.ini
- flintci - .flintci.yml
- git - .git, .gitignore, .gitattributes, .gitmodules, .mailmap, .githooks
- github - .github
- gitlab - .gitlab-ci.yml
- gulpfile - gulpfile.js
- grumphp - grumphp.yml.dist, grumphp.yml
- gush - .gush.yml
- gruntjs - Gruntfile.js
- humbug - humbug.json.dist, humbug.json
- infection - infection.yml, .infection.yml, infection.json.dist
- insight - .sensiolabs.yml
- jetbrains - .idea
- jshint - .jshintrc, .jshintignore
- mercurial - .hg, .hgtags, .hgignore, .hgeol
- mkdocs - mkdocs.yml
- npm - package.json, .npmignore, .npmrc, package-lock.json
- openshift - .openshift
- phan - .phan
- pharcc - .pharcc.yml
- phalcon - .phalcon
- phpbench - phpbench.json
- phpci - phpci.yml
- Phpdocumentor - .phpdoc.xml, phpdoc.dist.xml
- phpdox - phpdox.xml.dist, phpdox.xml
- phinx - phinx.yml
- phpformatter - .formatter.yml
- phpmetrics - .phpmetrics.yml.dist
- phpsa - .phpsa.yml

- phpspec - phpspec.yml, .phpspec, phpspec.yml.dist
- phpstan - phpstan.neon, .phpstan.neon, phpstan.neon.dist
- phpswitch - .phpswitch.yml
- PHPUnit - phpunit.xml.dist, phpunit.xml
- prettier - .prettierrc, .prettierignore
- psalm - psalm.xml
- puppet - .puppet
- rmt - .rmt.yml
- robo - RoboFile.php
- scrutinizer - .scrutinizer.yml
- semantic versioning - .semver
- SPIP - paquet.xml
- stickler - .stickler.yml
- storyplayer - storyplayer.json.dist
- styleci - .styleci.yml
- stylelint - .stylelintrc
- sublimelinter - .csslintrc
- svn - svn.revision, .svn, .svnignore
- transifex - .tx
- Robots.txt - robots.txt
- travis - .travis.yml, .env.travis, .travis, .travis.php.ini, .travis.coverage.sh, .travis.ini
- varci - .varci, .varci.yml
- Vagrant - Vagrantfile
- visualstudio - .vscode
- webpack - webpack.mix.js, webpack.config.js
- yarn - yarn.lock
- Zend_Tool - zfproject.xml

7.4 Supported PHP Extensions

PHP extensions are used to check for structures usage (classes, interfaces, etc.), to identify dependencies and directives.

PHP extensions are described with the list of structures they define : functions, classes, constants, traits, variables, interfaces, namespaces, and directives.

- ext/amqp
- ext/apache
- ext/apc
- ext/apcu

- ext/array
- ext/php-ast
- ext/async
- ext/bcmath
- ext/bzip2
- ext/cairo
- ext/calendar
- ext/cmark
- ext/com
- ext/crypto
- ext/csprng
- ext/ctype
- ext/curl
- ext/cyrus
- ext/date
- ext/db2
- ext/dba
- ext/decimal
- ext/dio
- ext/dom
- ext/ds
- ext/eaccelerator
- ext/eio
- ext/enchant
- ext/ereg
- ext/ev
- ext/event
- ext/exif
- ext/expect
- ext/fam
- ext/fann
- ext/fdf
- ext/ffi
- ext/ffmpeg
- ext/file
- ext/fileinfo

- ext/filter
- ext/fpm
- ext/ftp
- ext/gd
- ext/gearman
- ext/gender
- ext/geoip
- ext/gettext
- ext/gmagick
- ext/gmp
- ext/gnupgp
- ext/grpc
- ext/hash
- ext/hrtime
- ext/pecl_http
- ext/ibase
- ext/iconv
- ext/igbinary
- ext/iis
- ext/imagick
- ext/imap
- ext/info
- ext/inotify
- ext/intl
- ext/json
- ext/judy
- ext/kdm5
- ext/lapack
- ext/ldap
- ext/leveldb
- ext/libevent
- ext/libsodium
- ext/libxml
- ext/lua
- ext/lzf
- ext/mail

- ext/mailparse
- ext/math
- ext/mbstring
- ext/mcrypt
- ext/memcache
- ext/memcached
- ext/mhash
- ext/ming
- ext/mongo
- ext/mongodb
- ext/msgpack
- ext/mssql
- ext/mysql
- ext/mysqli
- ext/ncurses
- ext/newt
- ext/nsapi
- ext/ob
- ext/oci8
- ext/odbc
- ext/opcache
- ext/opencensus
- ext/openssl
- ext/parle
- ext/parsekit
- ext/password
- ext/pcntl
- ext/pcov
- ext/pcre
- ext/pdo
- ext/pgsql
- ext/phalcon
- ext/phar
- ext/posix
- ext/proctitle
- ext/pspell

- ext/psr
- ext/rar
- ext/rdkafka
- ext/readline
- ext/recode
- ext/redis
- ext/reflection
- ext/runkit
- ext/sdl
- ext/seaslog
- ext/sem
- ext/session
- ext/shmop
- ext/simplexml
- ext/snmp
- ext/soap
- ext/sockets
- ext/sphinx
- ext/spl
- ext/sqlite
- ext/sqlite3
- ext/sqlsrv
- ext/ssh2
- ext/standard
- ext/stats
- String
- ext/suhosin
- ext/svm
- ext/swoole
- ext/tidy
- ext/tokenizer
- ext/tokyotyrant
- ext/trader
- ext/uopz
- ext/uuid
- ext/v8js

- ext/varnish
- ext/vips
- ext/wasm
- ext/wddx
- ext/weakref
- ext/wikidiff2
- ext/wincache
- ext/xattr
- ext/xcache
- ext/xdebug
- ext/xdiff
- ext/xhprof
- ext/xml
- ext/xmlreader
- ext/xmlrpc
- ext/xmlwriter
- ext/xsl
- ext/xxtea
- ext/yaml
- ext/yis
- ext/zbarcode
- ext/zend_monitor
- ext/zip
- ext/zlib
- ext/0mq
- ext/zookeeper

CHAPTER 8

Configuration

8.1 Summary

8.2 Common Behavior

8.2.1 General Philosophy

Exakat tries to avoid configuration as much as possible, so as to focus on working out of the box, rather than spend time on pre-requisite.

As such, it probably does more work, but that may be dismissed later, at reading time.

More configuration options appear with the evolution of the engine.

8.2.2 Precedence

The exakat engine read directives from three places :

1. The command line options
2. The .exakat.ini file at the root of the code
3. The config.ini file in the project directory

4. The exakat.ini file in the config directory
5. The default values in the code

The precedence of the directives is the same as the list above : command line options always have highest priority, config.ini files are in second, when command line are not available, and finally, the default values are read in the code.

Some of the directives are only available in the config.ini files.

8.2.3 Common Options

All options are the same, whatever the command provided to exakat. -f always means files, and -q always means quick.

Any option that a command doesn't understand is ignored.

Any option that is not recognized is ignored and reported (with visibility).

8.3 Project Configuration

Project configuration are were the project specific configuration are stored. For example, the project name, the ignored directories or its external libraries are kept. Configurations only affect one project and not the others.

Project configuration file are called 'config.ini'. They are located, one per project, in the 'projects/<project name>/config.ini' file.

8.3.1 Available Options

Here are the currently available options in Exakat's project configuration file : projects/<project name>/config.ini

Option	Description
phpversion	Version with which to run the analyze. It may be one of : 7.3, 7.2, 7.1, 7.0, 5.6, 5.5, 5.4, 5.3, 5.2. Default is 7.2 or the CLI version used to init the project. 5.* versions are available, but are less tested. 7.3 is actually the current dev version.
include_dirs	This is the list of files and dir to include in the project's directory. It is chrooted in the project's folder. Values provided with a starting / are used as a path prefix. Values without / are used as a substring, anywhere in the path. include_dirs are added AFTER ignore_dirs, so as to partially ignore a folder, such as the vendor folder from composer.
ignore_dirs	This is the list of files and dir to ignore in the project's directory. It is chrooted in the project's folder. Values provided with a starting / are used as a path prefix. Values without / are used as a substring, anywhere in the path.
ignore_dirs	This is the list of files and dir to ignore in the project's directory. It is chrooted in the project's folder. Values provided with a starting / are used as a path prefix. Values without / are used as a substring, anywhere in the path.
file_extensions	This is the list of file extensions that is considered as PHP scripts. All others are ignored. All files bearing those extensions are subject to check, though they are scanned first for PHP tags before being analyzed. The extensions are comma separated, without dot. The default are : php, php3, inc, tpl, phtml, tmpl, phps, ctp
project_name	This is the project name, as it appears at the top left in the Ambassador report.
project_url	This is the repository URL for the project. It is used to get the source for the project.
project_vcs	This is the VCS used to fetch the project source.
project_description	This is the description of the project.
project_packagist	This is the packagist name for the code, when the code is fetched with composer.

8.4 In-code Configuration

In-code configuration is a configuration file that sits at the root of the code. When exakat finds it, it uses it for in-code auditing.

The file is *.exakat.yaml*, and is a valid YAML file. *.exakat.yml* is also valid, but not recommended.

In case the file is found but not valid, Exakat reverts to default values.

Unrecognized values are ignored.

8.4.1 Exakat in-code example

```
project: exakat
project_name: exakat
project_rulesets:
- my_ruleset
- Security
project_report:
- Ambassador
file_extensions: php,php3,phtml
include_dirs:
  - /
ignore_dirs:
  - /tests
  - /vendor
  - /docs
```

(continues on next page)

(continued from previous page)

```
  - /media
ignore_rules:
  - Structures/AddZero
rulesets:
  my_ruleset:
      - Structures/AddZero
      - Structures/MultiplyByOne
```

8.4.2 Exakat in-code skeleton

Copy-paste this YAML code into a file called *.exakat.yaml*, located at the root of your repository.

```
file_extensions: php,php3,phtml
project: <project short name>
project_name: <project name, as displayed in reports>
project_rulesets:
- <list of rulesets to apply>
- Analysis
file_extensions: php,php3,phtml
project_report:
- <list of reports to build>
- Ambassador
include_dirs:
  - /
ignore_rules:
  -
ignore_dirs:
  - /tests
  - /vendor
  - /docs
  - /media
```

8.4.3 Available Options

Here are the currently available options in Exakat's project configuration file : projects/<project name>/config.ini

Option	Description
include_dirs[]	This is the list of files and dir to include in the project's directory. It is chrooted in the project's folder. Values provided with a starting / are used as a path prefix. Values without / are used as a substring, anywhere in the path. include_dirs are added AFTER ignore_dirs, so as to partially ignore a folder, such as the vendor folder from composer.
ignore_dirs[]	This is the list of files and dir to ignore in the project's directory. It is chrooted in the project's folder. Values provided with a starting / are used as a path prefix. Values without / are used as a substring, anywhere in the path.
ignore_rules[]	The rules mentioned in this list are ignored when running the audit. Rules are ignored after loading the rulesets configuration : as such, it is possible to ignore rules inside a ruleset, without ignoring the whole ruleset. The rules in this list are Exakat's short name : ignore_rules[] = "Structures/AddZero"
file_extensions	This is the list of file extensions that is considered as PHP scripts. All others are ignored. All files bearing those extensions are subject to check, though they are scanned first for PHP tags before being analyzed. The extensions are comma separated, without dot. The default are : php, php3, inc, tpl, phtml, tmpl, phps, ctp
project_name	This is the project name, as it appears at the top left in the Ambassador report.
project_url	This is the repository URL for the project. It is used to get the source for the project.
project_vcs	This is the VCS used to fetch the project source.
project_description	This is the description of the project.
project_packagist	This is the packagist name for the code, when the code is fetched with composer.

8.5 Commandline Configuration

Commandline configurations are detailled with each command, in the _Commands section.

8.6 Specific analysis configurations

Some analyzer may be configured individually. Those parameters are then specific to one analyzer, and it only affects their behavior.

Analyzers may be configured in the *project/*/config.ini*; they may also be configured globally in the *config/exakat.ini* file.

Array() / [ ] Consistence

- array_ratio : 10
  - Percentage of arrays in one of the syntaxes, to trigger the other syntax as a violation.

Too Many Array Dimensions

- maxDimensions : 3
  - Number of valid dimensions in an array.

Custom Class Usage

- forbiddenClasses :
  - List of classes to be avoided

Cancel Common Method

- cancelThreshold : 75

  - Minimal number of cancelled methods to suggest the cancellation of the parent.

Could Be Parent Method

- minChildren : 4
  - Minimal number of children using this method.

Fossilized Method

- fossilizationThreshold : 6
  - Minimal number of overwriting methods to consider a method difficult to update.

Immutable Signature

- maxOverwrite : 8
  - Minimal number of method overwrite to consider that any refactor on the method signature is now hard.

Make Magic Concrete

- magicMemberUsage : 1
  - Minimal number of magic member usage across the code, to trigger a concrete property.

Too Many Children

- childrenClassCount : 15
  - Threshold for too many children classes for one class.

Too Many Dereferencing

- tooManyDereferencing : 7
  - Maximum number of dereferencing.

Too Many Finds

- minimumFinds : 5
  - Minimal number of prefixed methods to report.
- findPrefix : find
  - list of prefix to use when detecting the 'find'. Comma-separated list, case insensitive.
- findSuffix :
  - list of fix to use when detecting the 'find'. Comma-separated list, case insensitive.

Too Many Injections

- injectionsCount : 5
  - Threshold for too many injected parameters for one class.

Large Try Block

- tryBlockMaxSize : 5
  - Maximal number of expressions in the try block.

Long Preparation For Throw

- preparationLineCount : 8
  - Minimal number of lines before the throw.

Missing Include

- constant_or_variable_name : 100
  - Literal value to be used when including files. For example, by configuring 'Files_MissingInclude["HOME_DIR"] = "/tmp/myDir/";', then 'include HOME_DIR . "my_class.php"; will be actually be used as '/tmp/myDir/my_class.php'. Constants must be configured with their correct case. Variable must be configured with their initial '$'. Configure any number of variable and constant names.

Could Make A Function

- centralizeThreshold : 8
  - Minimal number of calls of the function with one common argument.

Hardcoded Passwords

- passwordsKeys : password_keys.json
  - List of array index and property names that shall be checked for potential secret key storages.

Prefix And Suffixes With Typehint

- prefixedType : prefixedType['is'] = 'bool';

prefixedType['has'] = 'bool'; prefixedType['set'] = 'void'; prefixedType['list'] = 'array';

- List of prefixes and their expected returntype
- suffixedType : prefixedType['list'] = 'bool';

prefixedType['int'] = 'int'; prefixedType['string'] = 'string'; prefixedType['name'] = 'string'; prefixedType['description'] = 'string'; prefixedType['id'] = 'int'; prefixedType['uuid'] = 'Uuid';

- List of suffixes and their expected returntype

Too Many Local Variables

- tooManyLocalVariableThreshold : 15
  - Minimal number of variables in one function or method to report.

Too Many Parameters

- parametersCount : 8
  - Minimal number of parameters to report.

Too Much Indented

- indentationAverage : 1
  - Minimal average of indentation in a method to report. Default is 1.0, which means that the method is on average at one level of indentation or more.
- minimumSize : 3
  - Minimal number of expressions in a method to apply this analysis.

Useless Argument

- maxUsageCount : 30
  - Maximum count of function usage. Use this to limit the amount of processed arguments.

Abstract Away

- abstractableCalls :

  - Functions that shouldn't be called directly, unless in a method.
- abstractableClasses :
  - Classes that shouldn't be instantiated directly, unless in a method.

Memoize MagicCall

- minMagicCallsToGet : 2
  - Minimal number of calls of a magic property to make it worth locally caching.

PHP Keywords As Names

- reservedNames :
  - Other reserved names : all in a string, comma separated.
- allowedNames :
  - PHP reserved names that can be used in the code. All in a string, comma separated.

Too Many Native Calls

- nativeCallCounts : 3
  - Number of native calls found inside another call.

Keep Files Access Restricted

- filePrivileges : 0777
  - List of forbidden file modes (comma separated).

Should Use Prepared Statement

- queryMethod : query_methods.json
  - Methods that call a query.

Too Complex Expression

- complexExpressionThreshold : 30
  - Minimal number of operators in one expression to report.

Long Arguments

- codeTooLong : 100
  - Minimum size of a functioncall or a methodcall to be considered too long.

Too Long A Block

- longBlock : 200
  - Size of a block for it to be too long. A block is commanded by a for, foreach, while, do...while, if/then else structure.

Max Level Of Nesting

- maxLevel : 4
  - Maximum level of nesting for control flow structures in one scope.

Nested Ifthen

- nestedIfthen : 3
  - Maximal number of acceptable nesting of if-then structures

@ Operator

- authorizedFunctions : noscream_functions.json
  - Functions that are authorized to sports a @.

Duplicate Literal

- minDuplicate : 15
  - Minimal number of duplication before the literal is reported.

Selector

- selector :
  - A selector expression to identify atoms in the code.

Variables With Long Names

- variableLength : 20
  - Minimum size of a long variable name, including the initial $.

8.7 Check Install

Once the prerequisite are installed, it is advised to run to check if all is found :

php exakat.phar doctor

After this run, you may edit 'config/config.ini' to change some of the default values. Most of the time, the default values will be OK for a quick start.

CHAPTER 9

Scoping analysis

9.1 Summary

- *scoping files*
- *scoping rules*
- *scoping reports*

9.2 Scoping files

ignore_dirs and include_dirs are the option used to select files within a folder. Here are some tips to choose

- From the full list of files, ignore_dirs[] is applied, then include_dirs is applied. The remaining list is processed.
- ignore one file : *ignore_dirs[] = "/path/to/file.php"*
- ignore one dir : *ignore_dirs[] = "/path/to/dir/"*
- ignore siblings but include one dir : *ignore_dirs[] = "/path/to/parent/"; include_dirs[] = "/path/to/parent/dir/"*
- ignore every name containing 'test' : *ignore_dirs[] = "test";*
- only include one dir (and exclude the rest): *include_dirs[] = "/path/to/dir/";*
- omitting include_dirs defaults to *"include_dirs[] = ""*
- omitting ignore_dirs defaults to *"ignore_dirs[] = ""*
- including or ignoring files multiple times only has effect once

include_dirs has priority over the *config.cache* configuration file. If a folder has been marked for exclusion in the *config.cache* file, it may be forced to be included by configuring its value with include_dirs in the *config.ini* file.

9.3 Scoping rules

to be completed

9.4 Scoping reports

Exakat builds a list of analysis to run, based on two directives : *project_reports* and *projects_themes*. Both are list of rulesets. Unknown rulesets are omitted.

project_reports makes sure you can extract those reports, while *projects_themes* allow you to build reports a la carte later, and avoid running the whole audit again.

9.4.1 Required rulesets

First, analysis are very numerous, and it is very tedious to sort them by hand. Exakat only handles 'themes' which are groups of analysis. There are several list of rulesets available by default, and it is possible to customize those lists.

When using the *projects_themes* directive, you can configure which rulesets must be processed by exakat, each time a 'project' command is run. Those rulesets are always run.

9.4.2 Report-needed rulesets

Reports are build based on results found during the auditing phase. Some reports, like 'Ambassador' or 'Drillinstructor' needs the results of specific rulesets. Others, like 'Text' or 'Json' build reports at the last moment.

As such, exakat uses the project_reports directive to collect the list of necessary rulesets, and add them to the *projects_themes* results.

9.4.3 Late reports

It is possible de extract a report, even if the configuration has not been explicitly set for it.

For example, it is possible to build the Owasp report after telling exakat to build a 'Ambassador' report, as Ambassador includes all the analysis needed for Owasp. On the other hand, the contrary is not true : one can't get the Ambassador report after running exakat for the Owasp report, as Owasp only covers the security rulesets, and Ambassador requires other rulesets.

9.4.4 Recommendations

- The 'Ambassador' report has all the classic rulesets, it's the most comprehensive choice.
- To collect everything possible, use the ruleset 'All'. It's also the longest-running ruleset of all.
- To get one report, simply configure project_report with that report.
- You may configure several rulesets, like 'Security', 'Suggestions', 'CompatibilityPHP73', and later extract independant results with the 'Text' or 'Json' format.
- If you just want one compulsory report and two optional reports (total of three), simply configure all of them with project_report. It's better to produce extra reports, than run again a whole audit to collect missing informations.
- It is possible to configure customized rulesets, and use them in project_rulesets

- Excluding one analyzer is not supported. Use custom rulesets to build a new one instead.

9.4.5 Example

```
project_reports[] = 'Drillinstructor';
project_reports[] = 'Owasp';

project_themes[] = 'Security';
project_themes[] = 'Suggestions';
```

With that configuration, the Drillinstructor and the Owasp report are created automatically when running 'project'. Use the following command to get the specific rulesets ;

```
php exakat.phar report -p <project> -format Text -T Security -v
```

9.5 Predefined config files

INI configuration for built-in rulesets. Copy them in config/themes.ini, and make your owns.

27 rulesets detailled here :

9.5.1 Analyze

[Analyze]
analyzer[] = "Arrays/AmbiguousKeys";
analyzer[] = "Arrays/MultipleIdenticalKeys";
analyzer[] = "Arrays/NoSpreadForHash";
analyzer[] = "Arrays/NonConstantArray";
analyzer[] = "Arrays/NullBoolean";
analyzer[] = "Arrays/RandomlySortedLiterals";
analyzer[] = "Arrays/TooManyDimensions";
analyzer[] = "Attributes/ModifyImmutable";
analyzer[] = "Classes/AbstractOrImplements";
analyzer[] = "Classes/AbstractStatic";
analyzer[] = "Classes/AccessPrivate";
analyzer[] = "Classes/AccessProtected";
analyzer[] = "Classes/AmbiguousStatic";
analyzer[] = "Classes/AmbiguousVisibilities";
analyzer[] = "Classes/AvoidOptionArrays";
analyzer[] = "Classes/AvoidOptionalProperties";
analyzer[] = "Classes/CantExtendFinal";
analyzer[] = "Classes/CantInstantiateClass";
analyzer[] = "Classes/CheckOnCallUsage";
analyzer[] = "Classes/CitSameName";
analyzer[] = "Classes/CloneWithNonObject";
analyzer[] = "Classes/CouldBeAbstractClass";
analyzer[] = "Classes/CouldBeFinal";

analyzer[] = “Classes/CouldBeStatic”;
analyzer[] = “Classes/CouldBeStringable”;
analyzer[] = “Classes/CyclicReferences”;
analyzer[] = “Classes/DependantAbstractClass”;
analyzer[] = “Classes/DifferentArgumentCounts”;
analyzer[] = “Classes/DirectCallToMagicMethod”;
analyzer[] = “Classes/DontSendThisInConstructor”;
analyzer[] = “Classes/DontUnsetProperties”;
analyzer[] = “Classes/EmptyClass”;
analyzer[] = “Classes/FinalByOcramius”;
analyzer[] = “Classes/HiddenNullable”;
analyzer[] = “Classes/ImplementIsForInterface”;
analyzer[] = “Classes/ImplementedMethodsArePublic”;
analyzer[] = “Classes/IncompatibleSignature”;
analyzer[] = “Classes/IncompatibleSignature74”;
analyzer[] = “Classes/InstantiatingAbstractClass”;
analyzer[] = “Classes/MakeDefault”;
analyzer[] = “Classes/MakeGlobalAProperty”;
analyzer[] = “Classes/MethodSignatureMustBeCompatible”;
analyzer[] = “Classes/MismatchProperties”;
analyzer[] = “Classes/MissingAbstractMethod”;
analyzer[] = “Classes/MultipleDeclarations”;
analyzer[] = “Classes/MultipleTraitOrInterface”;
analyzer[] = “Classes/NoMagicWithArray”;
analyzer[] = “Classes/NoPSSOutsideClass”;
analyzer[] = “Classes/NoParent”;
analyzer[] = “Classes/NoPublicAccess”;
analyzer[] = “Classes/NoSelfReferencingConstant”;
analyzer[] = “Classes/NonNullableSetters”;
analyzer[] = “Classes/NonPpp”;
analyzer[] = “Classes/NonStaticMethodsCalledStatic”;
analyzer[] = “Classes/OldStyleConstructor”;
analyzer[] = “Classes/OldStyleVar”;
analyzer[] = “Classes/ParentFirst”;
analyzer[] = “Classes/PropertyCouldBeLocal”;
analyzer[] = “Classes/PropertyNeverUsed”;
analyzer[] = “Classes/PropertyUsedInOneMethodOnly”;
analyzer[] = “Classes/PssWithoutClass”;
analyzer[] = “Classes/RedefinedConstants”;
analyzer[] = “Classes/RedefinedDefault”;
analyzer[] = “Classes/RedefinedPrivateProperty”;
analyzer[] = “Classes/ScalarOrObjectProperty”;
analyzer[] = “Classes/ShouldUseSelf”;
analyzer[] = “Classes/ShouldUseThis”;
analyzer[] = “Classes/StaticContainsThis”;
analyzer[] = “Classes/StaticMethodsCalledFromObject”;
analyzer[] = “Classes/SwappedArguments”;

```
analyzer[] = "Classes/ThisIsForClasses";
analyzer[] = "Classes/ThisIsNotAnArray";
analyzer[] = "Classes/ThisIsNotForStatic";
analyzer[] = "Classes/ThrowInDestruct";
analyzer[] = "Classes/TooManyDereferencing";
analyzer[] = "Classes/TooManyFinds";
analyzer[] = "Classes/TooManyInjections";
analyzer[] = "Classes/UndeclaredStaticProperty";
analyzer[] = "Classes/UndefinedClasses";
analyzer[] = "Classes/UndefinedConstants";
analyzer[] = "Classes/UndefinedParentMP";
analyzer[] = "Classes/UndefinedProperty";
analyzer[] = "Classes/UndefinedStaticMP";
analyzer[] = "Classes/UndefinedStaticclass";
analyzer[] = "Classes/UnresolvedClasses";
analyzer[] = "Classes/UnresolvedInstanceof";
analyzer[] = "Classes/UnusedClass";
analyzer[] = "Classes/UnusedConstant";
analyzer[] = "Classes/UseClassOperator";
analyzer[] = "Classes/UseInstanceof";
analyzer[] = "Classes/UsedOnceProperty";
analyzer[] = "Classes/UselessAbstract";
analyzer[] = "Classes/UselessConstructor";
analyzer[] = "Classes/UselessFinal";
analyzer[] = "Classes/UsingThisOutsideAClass";
analyzer[] = "Classes/WeakType";
analyzer[] = "Classes/WrongName";
analyzer[] = "Classes/WrongTypedPropertyInit";
analyzer[] = "Constants/BadConstantnames";
analyzer[] = "Constants/ConstRecommended";
analyzer[] = "Constants/ConstantStrangeNames";
analyzer[] = "Constants/CreatedOutsideItsNamespace";
analyzer[] = "Constants/InvalidName";
analyzer[] = "Constants/MultipleConstantDefinition";
analyzer[] = "Constants/StrangeName";
analyzer[] = "Constants/UndefinedConstants";
analyzer[] = "Exceptions/CantThrow";
analyzer[] = "Exceptions/CatchUndefinedVariable";
analyzer[] = "Exceptions/ForgottenThrown";
analyzer[] = "Exceptions/OverwriteException";
analyzer[] = "Exceptions/ThrowFunctioncall";
analyzer[] = "Exceptions/UncaughtExceptions";
analyzer[] = "Exceptions/Unthrown";
analyzer[] = "Exceptions/UselessCatch";
analyzer[] = "Files/InclusionWrongCase";
analyzer[] = "Files/MissingInclude";
analyzer[] = "Functions/AliasesUsage";
```

analyzer[] = “Functions/AvoidBooleanArgument”;
analyzer[] = “Functions/CallbackNeedsReturn”;
analyzer[] = “Functions/CancelledParameter”;
analyzer[] = “Functions/CannotUseStaticForClosure”;
analyzer[] = “Functions/CouldCentralize”;
analyzer[] = “Functions/DeepDefinitions”;
analyzer[] = “Functions/DontUseVoid”;
analyzer[] = “Functions/EmptyFunction”;
analyzer[] = “Functions/FnArgumentVariableConfusion”;
analyzer[] = “Functions/HardcodedPasswords”;
analyzer[] = “Functions/InsufficientTypehint”;
analyzer[] = “Functions/MismatchParameterAndType”;
analyzer[] = “Functions/MismatchParameterName”;
analyzer[] = “Functions/MismatchTypeAndDefault”;
analyzer[] = “Functions/MismatchedDefaultArguments”;
analyzer[] = “Functions/MismatchedTypehint”;
analyzer[] = “Functions/ModifyTypedParameter”;
analyzer[] = “Functions/MustReturn”;
analyzer[] = “Functions/NeverUsedParameter”;
analyzer[] = “Functions/NoBooleanAsDefault”;
analyzer[] = “Functions/NoLiteralForReference”;
analyzer[] = “Functions/NoReturnUsed”;
analyzer[] = “Functions/OnlyVariableForReference”;
analyzer[] = “Functions/OnlyVariablePassedByReference”;
analyzer[] = “Functions/RedeclaredPhpFunction”;
analyzer[] = “Functions/RelayFunction”;
analyzer[] = “Functions/ShouldUseConstants”;
analyzer[] = “Functions/ShouldYieldWithKey”;
analyzer[] = “Functions/TooManyLocalVariables”;
analyzer[] = “Functions/TypehintMustBeReturned”;
analyzer[] = “Functions/TypehintedReferences”;
analyzer[] = “Functions/UndefinedFunctions”;
analyzer[] = “Functions/UnknownParameterName”;
analyzer[] = “Functions/UnusedArguments”;
analyzer[] = “Functions/UnusedInheritedVariable”;
analyzer[] = “Functions/UnusedReturnedValue”;
analyzer[] = “Functions/UseConstantAsArguments”;
analyzer[] = “Functions/UselessReferenceArgument”;
analyzer[] = “Functions/UselessReturn”;
analyzer[] = “Functions/UsesDefaultArguments”;
analyzer[] = “Functions/UsingDeprecated”;
analyzer[] = “Functions/WithoutReturn”;
analyzer[] = “Functions/WrongArgumentType”;
analyzer[] = “Functions/WrongNumberOfArguments”;
analyzer[] = “Functions/WrongOptionalParameter”;
analyzer[] = “Functions/WrongReturnedType”;
analyzer[] = “Functions/WrongTypeWithCall”;

analyzer[] = “Functions/funcGetArgModified”;
analyzer[] = “Interfaces/AlreadyParentsInterface”;
analyzer[] = “Interfaces/CantImplementTraversable”;
analyzer[] = “Interfaces/ConcreteVisibility”;
analyzer[] = “Interfaces/CouldUseInterface”;
analyzer[] = “Interfaces/EmptyInterface”;
analyzer[] = “Interfaces/IsNotImplemented”;
analyzer[] = “Interfaces/NoGaranteeForPropertyConstant”;
analyzer[] = “Interfaces/RepeatedInterface”;
analyzer[] = “Interfaces/UndefinedInterfaces”;
analyzer[] = “Interfaces/UselessInterfaces”;
analyzer[] = “Namespaces/ConstantFullyQualified”;
analyzer[] = “Namespaces/EmptyNamespace”;
analyzer[] = “Namespaces/HiddenUse”;
analyzer[] = “Namespaces/MultipleAliasDefinitionPerFile”;
analyzer[] = “Namespaces/MultipleAliasDefinitions”;
analyzer[] = “Namespaces/ShouldMakeAlias”;
analyzer[] = “Namespaces/UnresolvedUse”;
analyzer[] = “Namespaces/UseWithFullyQualifiedNS”;
analyzer[] = “Performances/ArrayMergeInLoops”;
analyzer[] = “Performances/LogicalToInArray”;
analyzer[] = “Performances/MemoizeMagicCall”;
analyzer[] = “Performances/PrePostIncrement”;
analyzer[] = “Performances/StrposTooMuch”;
analyzer[] = “Performances/UseArraySlice”;
analyzer[] = “Php/ArrayKeyExistsWithObjects”;
analyzer[] = “Php/AssertFunctionIsReserved”;
analyzer[] = “Php/AssignAnd”;
analyzer[] = “Php/Assumptions”;
analyzer[] = “Php/AvoidMbDectectEncoding”;
analyzer[] = “Php/BetterRand”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/Crc32MightBeNegative”;
analyzer[] = “Php/Deprecated”;
analyzer[] = “Php/DontPolluteGlobalSpace”;
analyzer[] = “Php/EmptyList”;
analyzer[] = “Php/FopenMode”;
analyzer[] = “Php/ForeachObject”;
analyzer[] = “Php/HashAlgos”;
analyzer[] = “Php/Incompilable”;
analyzer[] = “Php/InternalParameterType”;
analyzer[] = “Php/IsAWithString”;
analyzer[] = “Php/IsnullVsEqualNull”;
analyzer[] = “Php/LogicalInLetters”;
analyzer[] = “Php/MissingMagicIsset”;
analyzer[] = “Php/MissingSubpattern”;
analyzer[] = “Php/MultipleDeclareStrict”;

```
analyzer[] = "Php/MustCallParentConstructor";
analyzer[] = "Php/NoClassInGlobal";
analyzer[] = "Php/NoReferenceForTernary";
analyzer[] = "Php/OnlyVariableForReference";
analyzer[] = "Php/PathinfoReturns";
analyzer[] = "Php/ReservedNames";
analyzer[] = "Php/ScalarAreNotArrays";
analyzer[] = "Php/ShortOpenTagRequired";
analyzer[] = "Php/ShouldUseCoalesce";
analyzer[] = "Php/StrtrArguments";
analyzer[] = "Php/TooManyNativeCalls";
analyzer[] = "Php/UnknownPcre2Option";
analyzer[] = "Php/UseObjectApi";
analyzer[] = "Php/UsePathinfo";
analyzer[] = "Php/UseSetCookie";
analyzer[] = "Php/UseStdclass";
analyzer[] = "Php/WrongAttributeConfiguration";
analyzer[] = "Php/WrongTypeForNativeFunction";
analyzer[] = "Php/oldAutoloadUsage";
analyzer[] = "Security/DontEchoError";
analyzer[] = "Security/ShouldUsePreparedStatement";
analyzer[] = "Structures/AddZero";
analyzer[] = "Structures/AlteringForeachWithoutReference";
analyzer[] = "Structures/AlternativeConsistenceByFile";
analyzer[] = "Structures/AlwaysFalse";
analyzer[] = "Structures/ArrayFillWithObjects";
analyzer[] = "Structures/ArrayMapPassesByValue";
analyzer[] = "Structures/ArrayMergeAndVariadic";
analyzer[] = "Structures/ArrayMergeArrayArray";
analyzer[] = "Structures/AssigneAndCompare";
analyzer[] = "Structures/AutoUnsetForeach";
analyzer[] = "Structures/BailOutEarly";
analyzer[] = "Structures/BooleanStrictComparison";
analyzer[] = "Structures/BreakOutsideLoop";
analyzer[] = "Structures/BuriedAssignation";
analyzer[] = "Structures/CastToBoolean";
analyzer[] = "Structures/CastingTernary";
analyzer[] = "Structures/CatchShadowsVariable";
analyzer[] = "Structures/CheckAllTypes";
analyzer[] = "Structures/CheckJson";
analyzer[] = "Structures/CoalesceAndConcat";
analyzer[] = "Structures/CommonAlternatives";
analyzer[] = "Structures/ComparedComparison";
analyzer[] = "Structures/ConcatEmpty";
analyzer[] = "Structures/ContinueIsForLoop";
analyzer[] = "Structures/CouldBeElse";
analyzer[] = "Structures/CouldBeStatic";
```

analyzer[] = “Structures/CouldUseDir”;
analyzer[] = “Structures/CouldUseShortAssignation”;
analyzer[] = “Structures/CouldUseStrrepeat”;
analyzer[] = “Structures/DanglingArrayReferences”;
analyzer[] = “Structures/DirThenSlash”;
analyzer[] = “Structures/DontChangeBlindKey”;
analyzer[] = “Structures/DontMixPlusPlus”;
analyzer[] = “Structures/DontReadAndWriteInOneExpression”;
analyzer[] = “Structures/DoubleAssignation”;
analyzer[] = “Structures/DoubleInstruction”;
analyzer[] = “Structures/DoubleObjectAssignation”;
analyzer[] = “Structures/DropElseAfterReturn”;
analyzer[] = “Structures/EchoWithConcat”;
analyzer[] = “Structures/ElseIfElseif”;
analyzer[] = “Structures/EmptyBlocks”;
analyzer[] = “Structures/EmptyLines”;
analyzer[] = “Structures/EmptyTryCatch”;
analyzer[] = “Structures/ErrorReportingWithInteger”;
analyzer[] = “Structures/EvalUsage”;
analyzer[] = “Structures/EvalWithoutTry”;
analyzer[] = “Structures/ExitUsage”;
analyzer[] = “Structures/FailingSubstrComparison”;
analyzer[] = “Structures/ForeachReferenceIsNotModified”;
analyzer[] = “Structures/ForeachSourceValue”;
analyzer[] = “Structures/ForgottenWhiteSpace”;
analyzer[] = “Structures/GlobalUsage”;
analyzer[] = “Structures/Htmlentitiescall”;
analyzer[] = “Structures/IdenticalConditions”;
analyzer[] = “Structures/IdenticalConsecutive”;
analyzer[] = “Structures/IdenticalOnBothSides”;
analyzer[] = “Structures/IfWithSameConditions”;
analyzer[] = “Structures/Iffectation”;
analyzer[] = “Structures/ImpliedIf”;
analyzer[] = “Structures/ImplodeArgsOrder”;
analyzer[] = “Structures/InconsistentElseif”;
analyzer[] = “Structures/IndicesAreIntOrString”;
analyzer[] = “Structures/InfiniteRecursion”;
analyzer[] = “Structures/InvalidPackFormat”;
analyzer[] = “Structures/InvalidRegex”;
analyzer[] = “Structures/IsZero”;
analyzer[] = “Structures/ListOmissions”;
analyzer[] = “Structures/LogicalMistakes”;
analyzer[] = “Structures/LoneBlock”;
analyzer[] = “Structures/LongArguments”;
analyzer[] = “Structures/MaxLevelOfIdentation”;
analyzer[] = “Structures/MbstringThirdArg”;
analyzer[] = “Structures/MbstringUnknownEncoding”;

analyzer[] = “Structures/MergeIfThen”;
analyzer[] = “Structures/MismatchedTernary”;
analyzer[] = “Structures/MissingCases”;
analyzer[] = “Structures/MissingNew”;
analyzer[] = “Structures/MissingParenthesis”;
analyzer[] = “Structures/MixedConcatInterpolation”;
analyzer[] = “Structures/ModernEmpty”;
analyzer[] = “Structures/MultipleDefinedCase”;
analyzer[] = “Structures/MultipleTypeVariable”;
analyzer[] = “Structures/MultiplyByOne”;
analyzer[] = “Structures/NegativePow”;
analyzer[] = “Structures/NestedIfthen”;
analyzer[] = “Structures/NestedTernary”;
analyzer[] = “Structures/NeverNegative”;
analyzer[] = “Structures/NextMonthTrap”;
analyzer[] = “Structures/NoAppendOnSource”;
analyzer[] = “Structures/NoChangeIncomingVariables”;
analyzer[] = “Structures/NoChoice”;
analyzer[] = “Structures/NoDirectUsage”;
analyzer[] = “Structures/NoEmptyRegex”;
analyzer[] = “Structures/NoGetClassNull”;
analyzer[] = “Structures/NoHardcodedHash”;
analyzer[] = “Structures/NoHardcodedIp”;
analyzer[] = “Structures/NoHardcodedPath”;
analyzer[] = “Structures/NoHardcodedPort”;
analyzer[] = “Structures/NoIssetWithEmpty”;
analyzer[] = “Structures/NoNeedForElse”;
analyzer[] = “Structures/NoNeedForTriple”;
analyzer[] = “Structures/NoParenthesisForLanguageConstruct”;
analyzer[] = “Structures/NoReferenceOnLeft”;
analyzer[] = “Structures/NoSubstrOne”;
analyzer[] = “Structures/NoVariableIsACondition”;
analyzer[] = “Structures/Noscream”;
analyzer[] = “Structures/NotEqual”;
analyzer[] = “Structures/NotNot”;
analyzer[] = “Structures/ObjectReferences”;
analyzer[] = “Structures/OnceUsage”;
analyzer[] = “Structures/OneLineTwoInstructions”;
analyzer[] = “Structures/OnlyVariableReturnedByReference”;
analyzer[] = “Structures/OrDie”;
analyzer[] = “Structures/PossibleInfiniteLoop”;
analyzer[] = “Structures/PrintAndDie”;
analyzer[] = “Structures/PrintWithoutParenthesis”;
analyzer[] = “Structures/PrintfArguments”;
analyzer[] = “Structures/QueriesInLoop”;
analyzer[] = “Structures/RepeatedPrint”;
analyzer[] = “Structures/RepeatedRegex”;

analyzer[] = “Structures/ResultMayBeMissing”;
analyzer[] = “Structures/ReturnTrueFalse”;
analyzer[] = “Structures/SameConditions”;
analyzer[] = “Structures/ShouldChainException”;
analyzer[] = “Structures/ShouldMakeTernary”;
analyzer[] = “Structures/ShouldPreprocess”;
analyzer[] = “Structures/ShouldUseExplodeArgs”;
analyzer[] = “Structures/StaticLoop”;
analyzer[] = “Structures/StripTagsSkipsClosedTag”;
analyzer[] = “Structures/StrposCompare”;
analyzer[] = “Structures/SuspiciousComparison”;
analyzer[] = “Structures/SwitchToSwitch”;
analyzer[] = “Structures/SwitchWithoutDefault”;
analyzer[] = “Structures/TernaryInConcat”;
analyzer[] = “Structures/TestThenCast”;
analyzer[] = “Structures/ThrowsAndAssign”;
analyzer[] = “Structures/TimestampDifference”;
analyzer[] = “Structures/UncheckedResources”;
analyzer[] = “Structures/UnconditionLoopBreak”;
analyzer[] = “Structures/UnknownPregOption”;
analyzer[] = “Structures/Unpreprocessed”;
analyzer[] = “Structures/UnsetInForeach”;
analyzer[] = “Structures/UnsupportedTypesWithOperators”;
analyzer[] = “Structures/UnusedGlobal”;
analyzer[] = “Structures/UseConstant”;
analyzer[] = “Structures/UseInstanceof”;
analyzer[] = “Structures/UsePositiveCondition”;
analyzer[] = “Structures/UseSystemTmp”;
analyzer[] = “Structures/UselessBrackets”;
analyzer[] = “Structures/UselessCasting”;
analyzer[] = “Structures/UselessCheck”;
analyzer[] = “Structures/UselessGlobal”;
analyzer[] = “Structures/UselessInstruction”;
analyzer[] = “Structures/UselessParenthesis”;
analyzer[] = “Structures/UselessSwitch”;
analyzer[] = “Structures/UselessUnset”;
analyzer[] = “Structures/VardumpUsage”;
analyzer[] = “Structures/WhileListEach”;
analyzer[] = “Structures/WrongRange”;
analyzer[] = “Structures/pregOptionE”;
analyzer[] = “Structures/toStringThrowsException”;
analyzer[] = “Traits/AlreadyParentsTrait”;
analyzer[] = “Traits/DependantTrait”;
analyzer[] = “Traits/EmptyTrait”;
analyzer[] = “Traits/MethodCollisionTraits”;
analyzer[] = “Traits/TraitNotFound”;
analyzer[] = “Traits/UndefinedInsteadof”;

analyzer[] = “Traits/UndefinedTrait”;
analyzer[] = “Traits/UselessAlias”;
analyzer[] = “Type/NoRealComparison”;
analyzer[] = “Type/OneVariableStrings”;
analyzer[] = “Type/ShouldTypecast”;
analyzer[] = “Type/SilentlyCastInteger”;
analyzer[] = “Type/StringHoldAVariable”;
analyzer[] = “Type/StringWithStrangeSpace”;
analyzer[] = “Typehints/MissingReturntype”;
analyzer[] = “Variables/AssignedTwiceOrMore”;
analyzer[] = “Variables/ConstantTypo”;
analyzer[] = “Variables/LostReferences”;
analyzer[] = “Variables/OverwrittenLiterals”;
analyzer[] = “Variables/StrangeName”;
analyzer[] = “Variables/UndefinedConstantName”;
analyzer[] = “Variables/UndefinedVariable”;
analyzer[] = “Variables/VariableNonascii”;
analyzer[] = “Variables/VariableUsedOnce”;
analyzer[] = “Variables/VariableUsedOnceByContext”;
analyzer[] = “Variables/WrittenOnlyVariable”;

9.5.2 Attributes

[Attributes]
analyzer[] = “Attributes/ModifyImmutable”;
analyzer[] = “Functions/KillsApp”;
analyzer[] = “Functions/UsingDeprecated”;

9.5.3 CE

[CE]
analyzer[] = “Arrays/ArrayNSUsage”;
analyzer[] = “Arrays/Arrayindex”;
analyzer[] = “Arrays/Multidimensional”;
analyzer[] = “Arrays/NegativeStart”;
analyzer[] = “Arrays/Phparrayindex”;
analyzer[] = “Arrays/WithCallback”;
analyzer[] = “Classes/Abstractclass”;
analyzer[] = “Classes/Abstractmethods”;
analyzer[] = “Classes/Anonymous”;
analyzer[] = “Classes/ClassAliasUsage”;
analyzer[] = “Classes/Classnames”;
analyzer[] = “Classes/CloningUsage”;
analyzer[] = “Classes/ConstantDefinition”;
analyzer[] = “Classes/DynamicClass”;
analyzer[] = “Classes/DynamicConstantCall”;

```
analyzer[] = "Classes/DynamicMethodCall";
analyzer[] = "Classes/DynamicNew";
analyzer[] = "Classes/DynamicPropertyCall";
analyzer[] = "Classes/FinalPrivate";
analyzer[] = "Classes/ImmutableSignature";
analyzer[] = "Classes/MagicMethod";
analyzer[] = "Classes/MultipleClassesInFile";
analyzer[] = "Classes/NonPpp";
analyzer[] = "Classes/OldStyleConstructor";
analyzer[] = "Classes/OverwrittenConst";
analyzer[] = "Classes/RedefinedMethods";
analyzer[] = "Classes/StaticMethods";
analyzer[] = "Classes/StaticMethodsCalledFromObject";
analyzer[] = "Classes/StaticProperties";
analyzer[] = "Classes/TestClass";
analyzer[] = "Classes/VariableClasses";
analyzer[] = "Complete/OverwrittenProperties";
analyzer[] = "Complete/SetParentDefinition";
analyzer[] = "Composer/Autoload";
analyzer[] = "Composer/IsComposerNsname";
analyzer[] = "Composer/UseComposer";
analyzer[] = "Composer/UseComposerLock";
analyzer[] = "Constants/CaseInsensitiveConstants";
analyzer[] = "Constants/ConditionedConstants";
analyzer[] = "Constants/ConstantUsage";
analyzer[] = "Constants/Constantnames";
analyzer[] = "Constants/DynamicCreation";
analyzer[] = "Constants/IsExtConstant";
analyzer[] = "Constants/MagicConstantUsage";
analyzer[] = "Constants/PhpConstantUsage";
analyzer[] = "Constants/VariableConstant";
analyzer[] = "Dump/CallOrder";
analyzer[] = "Dump/CollectAtomCounts";
analyzer[] = "Dump/CollectClassChanges";
analyzer[] = "Dump/CollectClassChildren";
analyzer[] = "Dump/CollectClassConstantCounts";
analyzer[] = "Dump/CollectClassDepth";
analyzer[] = "Dump/CollectClassInterfaceCounts";
analyzer[] = "Dump/CollectClassTraitsCounts";
analyzer[] = "Dump/CollectClassesDependencies";
analyzer[] = "Dump/CollectDefinitionsStats";
analyzer[] = "Dump/CollectFilesDependencies";
analyzer[] = "Dump/CollectForeachFavorite";
analyzer[] = "Dump/CollectGlobalVariables";
analyzer[] = "Dump/CollectLiterals";
analyzer[] = "Dump/CollectLocalVariableCounts";
analyzer[] = "Dump/CollectMbstringEncodings";
```

```
analyzer[] = "Dump/CollectMethodCounts";
analyzer[] = "Dump/CollectNativeCallsPerExpressions";
analyzer[] = "Dump/CollectParameterCounts";
analyzer[] = "Dump/CollectParameterNames";
analyzer[] = "Dump/CollectPhpStructures";
analyzer[] = "Dump/CollectPropertyCounts";
analyzer[] = "Dump/CollectReadability";
analyzer[] = "Dump/CollectUseCounts";
analyzer[] = "Dump/CollectVariables";
analyzer[] = "Dump/ConstantOrder";
analyzer[] = "Dump/CyclomaticComplexity";
analyzer[] = "Dump/DereferencingLevels";
analyzer[] = "Dump/EnvironnementVariables";
analyzer[] = "Dump/FossilizedMethods";
analyzer[] = "Dump/Inclusions";
analyzer[] = "Dump/IndentationLevels";
analyzer[] = "Dump/NewOrder";
analyzer[] = "Dump/ParameterArgumentsLinks";
analyzer[] = "Dump/TypehintingStats";
analyzer[] = "Dump/Typehintorder";
analyzer[] = "Exceptions/DefinedExceptions";
analyzer[] = "Exceptions/MultipleCatch";
analyzer[] = "Exceptions/ThrownExceptions";
analyzer[] = "Extensions/Extamqp";
analyzer[] = "Extensions/Extapache";
analyzer[] = "Extensions/Extapc";
analyzer[] = "Extensions/Extapcu";
analyzer[] = "Extensions/Extarray";
analyzer[] = "Extensions/Extast";
analyzer[] = "Extensions/Extasync";
analyzer[] = "Extensions/Extbcmath";
analyzer[] = "Extensions/Extbzip2";
analyzer[] = "Extensions/Extcairo";
analyzer[] = "Extensions/Extcalendar";
analyzer[] = "Extensions/Extcmark";
analyzer[] = "Extensions/Extcom";
analyzer[] = "Extensions/Extcrypto";
analyzer[] = "Extensions/Extcsprng";
analyzer[] = "Extensions/Extctype";
analyzer[] = "Extensions/Extcurl";
analyzer[] = "Extensions/Extcyrus";
analyzer[] = "Extensions/Extdate";
analyzer[] = "Extensions/Extdb2";
analyzer[] = "Extensions/Extdba";
analyzer[] = "Extensions/Extdecimal";
analyzer[] = "Extensions/Extdio";
analyzer[] = "Extensions/Extdom";
```

analyzer[] = “Extensions/Extds”;
analyzer[] = “Extensions/Exteaccelerator”;
analyzer[] = “Extensions/Exteio”;
analyzer[] = “Extensions/Extenchant”;
analyzer[] = “Extensions/Extereg”;
analyzer[] = “Extensions/Extev”;
analyzer[] = “Extensions/Extevent”;
analyzer[] = “Extensions/Extexif”;
analyzer[] = “Extensions/Extexpect”;
analyzer[] = “Extensions/Extfam”;
analyzer[] = “Extensions/Extfann”;
analyzer[] = “Extensions/Extfdf”;
analyzer[] = “Extensions/Extffi”;
analyzer[] = “Extensions/Extffmpeg”;
analyzer[] = “Extensions/Extfile”;
analyzer[] = “Extensions/Extfileinfo”;
analyzer[] = “Extensions/Extfilter”;
analyzer[] = “Extensions/Extfpm”;
analyzer[] = “Extensions/Extftp”;
analyzer[] = “Extensions/Extgd”;
analyzer[] = “Extensions/Extgearman”;
analyzer[] = “Extensions/Extgender”;
analyzer[] = “Extensions/Extgeoip”;
analyzer[] = “Extensions/Extgettext”;
analyzer[] = “Extensions/Extgmagick”;
analyzer[] = “Extensions/Extgmp”;
analyzer[] = “Extensions/Extgnupg”;
analyzer[] = “Extensions/Extgrpc”;
analyzer[] = “Extensions/Exthash”;
analyzer[] = “Extensions/Exthrtime”;
analyzer[] = “Extensions/Exthttp”;
analyzer[] = “Extensions/Extibase”;
analyzer[] = “Extensions/Exticonv”;
analyzer[] = “Extensions/Extigbinary”;
analyzer[] = “Extensions/Extiis”;
analyzer[] = “Extensions/Extimagick”;
analyzer[] = “Extensions/Extimap”;
analyzer[] = “Extensions/Extinfo”;
analyzer[] = “Extensions/Extinotify”;
analyzer[] = “Extensions/Extintl”;
analyzer[] = “Extensions/Extjson”;
analyzer[] = “Extensions/Extjudy”;
analyzer[] = “Extensions/Extkdm5”;
analyzer[] = “Extensions/Extlapack”;
analyzer[] = “Extensions/Extldap”;
analyzer[] = “Extensions/Extleveldb”;
analyzer[] = “Extensions/Extlibevent”;

```
analyzer[] = "Extensions/Extlibsodium";
analyzer[] = "Extensions/Extlibxml";
analyzer[] = "Extensions/Extlua";
analyzer[] = "Extensions/Extlzf";
analyzer[] = "Extensions/Extmail";
analyzer[] = "Extensions/Extmailparse";
analyzer[] = "Extensions/Extmath";
analyzer[] = "Extensions/Extmbstring";
analyzer[] = "Extensions/Extmcrypt";
analyzer[] = "Extensions/Extmemcache";
analyzer[] = "Extensions/Extmemcached";
analyzer[] = "Extensions/Extmhash";
analyzer[] = "Extensions/Extming";
analyzer[] = "Extensions/Extmongo";
analyzer[] = "Extensions/Extmongodb";
analyzer[] = "Extensions/Extmsgpack";
analyzer[] = "Extensions/Extmssql";
analyzer[] = "Extensions/Extmysql";
analyzer[] = "Extensions/Extmysqli";
analyzer[] = "Extensions/Extncurses";
analyzer[] = "Extensions/Extnewt";
analyzer[] = "Extensions/Extnsapi";
analyzer[] = "Extensions/Extob";
analyzer[] = "Extensions/Extoci8";
analyzer[] = "Extensions/Extodbc";
analyzer[] = "Extensions/Extopcache";
analyzer[] = "Extensions/Extopencensus";
analyzer[] = "Extensions/Extopenssl";
analyzer[] = "Extensions/Extparle";
analyzer[] = "Extensions/Extparsekit";
analyzer[] = "Extensions/Extpassword";
analyzer[] = "Extensions/Extpcntl";
analyzer[] = "Extensions/Extpcov";
analyzer[] = "Extensions/Extpcre";
analyzer[] = "Extensions/Extpdo";
analyzer[] = "Extensions/Extpgsql";
analyzer[] = "Extensions/Extphalcon";
analyzer[] = "Extensions/Extphar";
analyzer[] = "Extensions/Extposix";
analyzer[] = "Extensions/Extproctitle";
analyzer[] = "Extensions/Extpspell";
analyzer[] = "Extensions/Extpsr";
analyzer[] = "Extensions/Extrar";
analyzer[] = "Extensions/Extrdkafka";
analyzer[] = "Extensions/Extreadline";
analyzer[] = "Extensions/Extrecode";
analyzer[] = "Extensions/Extredis";
```

analyzer[] = “Extensions/Extreflection”;
analyzer[] = “Extensions/Extrunkit”;
analyzer[] = “Extensions/Extsdl”;
analyzer[] = “Extensions/Extseaslog”;
analyzer[] = “Extensions/Extsem”;
analyzer[] = “Extensions/Extsession”;
analyzer[] = “Extensions/Extshmop”;
analyzer[] = “Extensions/Extsimplexml”;
analyzer[] = “Extensions/Extsnmp”;
analyzer[] = “Extensions/Extsoap”;
analyzer[] = “Extensions/Extsockets”;
analyzer[] = “Extensions/Extsphinx”;
analyzer[] = “Extensions/Extspl”;
analyzer[] = “Extensions/Extsqlite”;
analyzer[] = “Extensions/Extsqlite3”;
analyzer[] = “Extensions/Extsqlsrv”;
analyzer[] = “Extensions/Extssh2”;
analyzer[] = “Extensions/Extstandard”;
analyzer[] = “Extensions/Extstats”;
analyzer[] = “Extensions/Extstring”;
analyzer[] = “Extensions/Extsuhosin”;
analyzer[] = “Extensions/Extsvm”;
analyzer[] = “Extensions/Extswoole”;
analyzer[] = “Extensions/Exttidy”;
analyzer[] = “Extensions/Exttokenizer”;
analyzer[] = “Extensions/Exttokyotyrant”;
analyzer[] = “Extensions/Exttrader”;
analyzer[] = “Extensions/Extuopz”;
analyzer[] = “Extensions/Extuuid”;
analyzer[] = “Extensions/Extv8js”;
analyzer[] = “Extensions/Extvarnish”;
analyzer[] = “Extensions/Extvips”;
analyzer[] = “Extensions/Extwasm”;
analyzer[] = “Extensions/Extwddx”;
analyzer[] = “Extensions/Extweakref”;
analyzer[] = “Extensions/Extwikidiff2”;
analyzer[] = “Extensions/Extwincache”;
analyzer[] = “Extensions/Extxattr”;
analyzer[] = “Extensions/Extxcache”;
analyzer[] = “Extensions/Extxdebug”;
analyzer[] = “Extensions/Extxdiff”;
analyzer[] = “Extensions/Extxhprof”;
analyzer[] = “Extensions/Extxml”;
analyzer[] = “Extensions/Extxmlreader”;
analyzer[] = “Extensions/Extxmlrpc”;
analyzer[] = “Extensions/Extxmlwriter”;
analyzer[] = “Extensions/Extxsl”;

analyzer[] = “Extensions/Extxxtea”;
analyzer[] = “Extensions/Extyaml”;
analyzer[] = “Extensions/Extyis”;
analyzer[] = “Extensions/Extzbarcode”;
analyzer[] = “Extensions/Extzendmonitor”;
analyzer[] = “Extensions/Extzip”;
analyzer[] = “Extensions/Extzlib”;
analyzer[] = “Extensions/Extzmq”;
analyzer[] = “Extensions/Extzookeeper”;
analyzer[] = “Files/IsCliScript”;
analyzer[] = “Files/NotDefinitionsOnly”;
analyzer[] = “Functions/Closures”;
analyzer[] = “Functions/ConditionedFunctions”;
analyzer[] = “Functions/DeepDefinitions”;
analyzer[] = “Functions/Dynamiccall”;
analyzer[] = “Functions/FallbackFunction”;
analyzer[] = “Functions/Functionnames”;
analyzer[] = “Functions/FunctionsUsingReference”;
analyzer[] = “Functions/IsExtFunction”;
analyzer[] = “Functions/IsGenerator”;
analyzer[] = “Functions/MarkCallable”;
analyzer[] = “Functions/MismatchParameterName”;
analyzer[] = “Functions/MultipleDeclarations”;
analyzer[] = “Functions/NullableWithConstant”;
analyzer[] = “Functions/Recursive”;
analyzer[] = “Functions/RedeclaredPhpFunction”;
analyzer[] = “Functions/Typehints”;
analyzer[] = “Functions/UnbindingClosures”;
analyzer[] = “Functions/UseArrowFunctions”;
analyzer[] = “Functions/VariableArguments”;
analyzer[] = “Functions/WrongOptionalParameter”;
analyzer[] = “Interfaces/Interfacenames”;
analyzer[] = “Interfaces/IsExtInterface”;
analyzer[] = “Namespaces/Alias”;
analyzer[] = “Namespaces/NamespaceUsage”;
analyzer[] = “Namespaces/Namespacesnames”;
analyzer[] = “Patterns/CourrierAntiPattern”;
analyzer[] = “Patterns/DependencyInjection”;
analyzer[] = “Patterns/Factory”;
analyzer[] = “Php/AlternativeSyntax”;
analyzer[] = “Php/Argon2Usage”;
analyzer[] = “Php/ArrayKeyExistsWithObjects”;
analyzer[] = “Php/AssertionUsage”;
analyzer[] = “Php/AutoloadUsage”;
analyzer[] = “Php/CastUnsetUsage”;
analyzer[] = “Php/CastingUsage”;
analyzer[] = “Php/Coalesce”;

```
analyzer[] = "Php/ConcatAndAddition";
analyzer[] = "Php/CryptoUsage";
analyzer[] = "Php/DeclareEncoding";
analyzer[] = "Php/DeclareStrict";
analyzer[] = "Php/DeclareStrictType";
analyzer[] = "Php/DeclareTicks";
analyzer[] = "Php/DetectCurrentClass";
analyzer[] = "Php/DirectivesUsage";
analyzer[] = "Php/DlUsage";
analyzer[] = "Php/EchoTagUsage";
analyzer[] = "Php/EllipsisUsage";
analyzer[] = "Php/ErrorLogUsage";
analyzer[] = "Php/FilterToAddSlashes";
analyzer[] = "Php/Gotonames";
analyzer[] = "Php/GroupUseDeclaration";
analyzer[] = "Php/Haltcompiler";
analyzer[] = "Php/HashAlgos74";
analyzer[] = "Php/IdnUts46";
analyzer[] = "Php/Incompilable";
analyzer[] = "Php/IntegerSeparatorUsage";
analyzer[] = "Php/IsINF";
analyzer[] = "Php/IsNAN";
analyzer[] = "Php/Labelnames";
analyzer[] = "Php/ListShortSyntax";
analyzer[] = "Php/ListWithKeys";
analyzer[] = "Php/MiddleVersion";
analyzer[] = "Php/NestedTernaryWithoutParenthesis";
analyzer[] = "Php/NoMoreCurlyArrays";
analyzer[] = "Php/OveriddenFunction";
analyzer[] = "Php/PearUsage";
analyzer[] = "Php/Php74Deprecation";
analyzer[] = "Php/Php74NewClasses";
analyzer[] = "Php/Php74NewConstants";
analyzer[] = "Php/Php74NewFunctions";
analyzer[] = "Php/Php74RemovedDirective";
analyzer[] = "Php/Php74RemovedFunctions";
analyzer[] = "Php/Php74ReservedKeyword";
analyzer[] = "Php/Php74mbstrrpos3rdArg";
analyzer[] = "Php/Php7RelaxedKeyword";
analyzer[] = "Php/Php80NamedParameterVariadic";
analyzer[] = "Php/Php80NewFunctions";
analyzer[] = "Php/Php80OnlyTypeHints";
analyzer[] = "Php/Php80RemovedConstant";
analyzer[] = "Php/Php80RemovedDirective";
analyzer[] = "Php/Php80RemovedFunctions";
analyzer[] = "Php/Php80RemovesResources";
analyzer[] = "Php/Php80UnionTypehint";
```

analyzer[] = “Php/Php80VariableSyntax”;
analyzer[] = “Php/PhpErrorMsgUsage”;
analyzer[] = “Php/RawPostDataUsage”;
analyzer[] = “Php/ReflectionExportIsDeprecated”;
analyzer[] = “Php/ReturnTypehintUsage”;
analyzer[] = “Php/ScalarAreNotArrays”;
analyzer[] = “Php/ScalarTypehintUsage”;
analyzer[] = “Php/SignatureTrailingComma”;
analyzer[] = “Php/SpreadOperatorForArray”;
analyzer[] = “Php/SuperGlobalUsage”;
analyzer[] = “Php/ThrowUsage”;
analyzer[] = “Php/ThrowWasAnExpression”;
analyzer[] = “Php/TrailingComma”;
analyzer[] = “Php/TriggerErrorUsage”;
analyzer[] = “Php/TryCatchUsage”;
analyzer[] = “Php/TryMultipleCatch”;
analyzer[] = “Php/TypedPropertyUsage”;
analyzer[] = “Php/UseAttributes”;
analyzer[] = “Php/UseBrowscap”;
analyzer[] = “Php/UseCli”;
analyzer[] = “Php/UseContravariance”;
analyzer[] = “Php/UseCookies”;
analyzer[] = “Php/UseCovariance”;
analyzer[] = “Php/UseMatch”;
analyzer[] = “Php/UseNullSafeOperator”;
analyzer[] = “Php/UseNullableType”;
analyzer[] = “Php/UseTrailingUseComma”;
analyzer[] = “Php/UseWeb”;
analyzer[] = “Php/UsesEnv”;
analyzer[] = “Php/YieldFromUsage”;
analyzer[] = “Php/YieldUsage”;
analyzer[] = “Psr/Psr11Usage”;
analyzer[] = “Psr/Psr13Usage”;
analyzer[] = “Psr/Psr16Usage”;
analyzer[] = “Psr/Psr3Usage”;
analyzer[] = “Psr/Psr6Usage”;
analyzer[] = “Psr/Psr7Usage”;
analyzer[] = “Security/CantDisableClass”;
analyzer[] = “Security/CantDisableFunction”;
analyzer[] = “Structures/AddZero”;
analyzer[] = “Structures/ArrayMapPassesByValue”;
analyzer[] = “Structures/ComplexExpression”;
analyzer[] = “Structures/ConstDefineFavorite”;
analyzer[] = “Structures/ConstantScalarExpression”;
analyzer[] = “Structures/CurlVersionNow”;
analyzer[] = “Structures/DereferencingAS”;
analyzer[] = “Structures/DontReadAndWriteInOneExpression”;

```
analyzer[] = "Structures/DynamicCalls";
analyzer[] = "Structures/DynamicCode";
analyzer[] = "Structures/ElseUsage";
analyzer[] = "Structures/ErrorMessages";
analyzer[] = "Structures/ErrorReportingWithInteger";
analyzer[] = "Structures/EvalUsage";
analyzer[] = "Structures/ExitUsage";
analyzer[] = "Structures/FileUploadUsage";
analyzer[] = "Structures/FileUsage";
analyzer[] = "Structures/ForgottenWhiteSpace";
analyzer[] = "Structures/FunctionSubscripting";
analyzer[] = "Structures/GlobalInGlobal";
analyzer[] = "Structures/GlobalUsage";
analyzer[] = "Structures/IncludeUsage";
analyzer[] = "Structures/MailUsage";
analyzer[] = "Structures/MultipleCatch";
analyzer[] = "Structures/NestedLoops";
analyzer[] = "Structures/NoDirectAccess";
analyzer[] = "Structures/NonBreakableSpaceInNames";
analyzer[] = "Structures/Noscream";
analyzer[] = "Structures/NotNot";
analyzer[] = "Structures/OnceUsage";
analyzer[] = "Structures/OpensslRandomPseudoByteSecondArg";
analyzer[] = "Structures/ResourcesUsage";
analyzer[] = "Structures/ShellUsage";
analyzer[] = "Structures/ShortTags";
analyzer[] = "Structures/TryFinally";
analyzer[] = "Structures/UnsupportedTypesWithOperators";
analyzer[] = "Structures/UseDebug";
analyzer[] = "Traits/IsExtTrait";
analyzer[] = "Traits/Php";
analyzer[] = "Traits/TraitUsage";
analyzer[] = "Traits/Traitnames";
analyzer[] = "Type/ArrayIndex";
analyzer[] = "Type/Binary";
analyzer[] = "Type/Email";
analyzer[] = "Type/GPCIndex";
analyzer[] = "Type/Heredoc";
analyzer[] = "Type/Hexadecimal";
analyzer[] = "Type/Md5String";
analyzer[] = "Type/Nowdoc";
analyzer[] = "Type/Octal";
analyzer[] = "Type/Pack";
analyzer[] = "Type/Path";
analyzer[] = "Type/Printf";
analyzer[] = "Type/Protocols";
analyzer[] = "Type/Regex";
```

analyzer[] = “Type/Shellcommands”;
analyzer[] = “Type/Sql”;
analyzer[] = “Type/Url”;
analyzer[] = “Variables/References”;
analyzer[] = “Variables/StaticVariables”;
analyzer[] = “Variables/UncommonEnvVar”;
analyzer[] = “Variables/VariableLong”;
analyzer[] = “Variables/VariableVariables”;
analyzer[] = “Vendors/Codeigniter”;
analyzer[] = “Vendors/Concrete5”;
analyzer[] = “Vendors/Drupal”;
analyzer[] = “Vendors/Ez”;
analyzer[] = “Vendors/Fuel”;
analyzer[] = “Vendors/Joomla”;
analyzer[] = “Vendors/Laravel”;
analyzer[] = “Vendors/Phalcon”;
analyzer[] = “Vendors/Symfony”;
analyzer[] = “Vendors/Typo3”;
analyzer[] = “Vendors/Wordpress”;
analyzer[] = “Vendors/Yii”;

9.5.4 CI-checks

[CI-checks]
analyzer[] = “Arrays/MultipleIdenticalKeys”;
analyzer[] = “Classes/CheckOnCallUsage”;
analyzer[] = “Classes/DirectCallToMagicMethod”;
analyzer[] = “Classes/DontUnsetProperties”;
analyzer[] = “Classes/MultipleDeclarations”;
analyzer[] = “Classes/MultipleTraitOrInterface”;
analyzer[] = “Classes/NoMagicWithArray”;
analyzer[] = “Classes/NoParent”;
analyzer[] = “Classes/NonPpp”;
analyzer[] = “Classes/NonStaticMethodsCalledStatic”;
analyzer[] = “Classes/RedefinedConstants”;
analyzer[] = “Classes/RedefinedDefault”;
analyzer[] = “Classes/StaticContainsThis”;
analyzer[] = “Classes/StaticMethodsCalledFromObject”;
analyzer[] = “Classes/ThrowInDestruct”;
analyzer[] = “Classes/UndeclaredStaticProperty”;
analyzer[] = “Classes/UndefinedConstants”;
analyzer[] = “Classes/UndefinedProperty”;
analyzer[] = “Classes/UndefinedStaticclass”;
analyzer[] = “Classes/UseClassOperator”;
analyzer[] = “Classes/UseInstanceof”;
analyzer[] = “Classes/UselessFinal”;

```
analyzer[] = "Classes/WrongTypedPropertyInit";
analyzer[] = "Constants/ConstRecommended";
analyzer[] = "Constants/ConstantStrangeNames";
analyzer[] = "Constants/MultipleConstantDefinition";
analyzer[] = "Constants/UndefinedConstants";
analyzer[] = "Exceptions/OverwriteException";
analyzer[] = "Exceptions/ThrowFunctioncall";
analyzer[] = "Exceptions/UselessCatch";
analyzer[] = "Functions/AliasesUsage";
analyzer[] = "Functions/CallbackNeedsReturn";
analyzer[] = "Functions/MustReturn";
analyzer[] = "Functions/NoLiteralForReference";
analyzer[] = "Functions/RedeclaredPhpFunction";
analyzer[] = "Functions/ShouldYieldWithKey";
analyzer[] = "Functions/TypehintMustBeReturned";
analyzer[] = "Functions/TypehintedReferences";
analyzer[] = "Functions/UndefinedFunctions";
analyzer[] = "Functions/UnknownParameterName";
analyzer[] = "Functions/UnusedInheritedVariable";
analyzer[] = "Functions/UseConstantAsArguments";
analyzer[] = "Functions/UsesDefaultArguments";
analyzer[] = "Functions/WrongNumberOfArguments";
analyzer[] = "Functions/WrongOptionalParameter";
analyzer[] = "Functions/WrongReturnedType";
analyzer[] = "Functions/WrongTypeWithCall";
analyzer[] = "Interfaces/CantImplementTraversable";
analyzer[] = "Interfaces/IsNotImplemented";
analyzer[] = "Interfaces/UndefinedInterfaces";
analyzer[] = "Namespaces/EmptyNamespace";
analyzer[] = "Namespaces/HiddenUse";
analyzer[] = "Namespaces/MultipleAliasDefinitionPerFile";
analyzer[] = "Namespaces/MultipleAliasDefinitions";
analyzer[] = "Namespaces/ShouldMakeAlias";
analyzer[] = "Performances/ArrayMergeInLoops";
analyzer[] = "Performances/PrePostIncrement";
analyzer[] = "Performances/StrposTooMuch";
analyzer[] = "Performances/UseArraySlice";
analyzer[] = "Php/AssignAnd";
analyzer[] = "Php/BetterRand";
analyzer[] = "Php/ConcatAndAddition";
analyzer[] = "Php/Deprecated";
analyzer[] = "Php/FopenMode";
analyzer[] = "Php/InternalParameterType";
analyzer[] = "Php/IsAWithString";
analyzer[] = "Php/IsnullVsEqualNull";
analyzer[] = "Php/LogicalInLetters";
analyzer[] = "Php/MissingSubpattern";
```

```
analyzer[] = "Php/NoClassInGlobal";
analyzer[] = "Php/NoReferenceForTernary";
analyzer[] = "Php/ScalarAreNotArrays";
analyzer[] = "Php/ShouldUseCoalesce";
analyzer[] = "Php/StrtrArguments";
analyzer[] = "Php/UseObjectApi";
analyzer[] = "Php/UsePathinfo";
analyzer[] = "Php/WrongTypeForNativeFunction";
analyzer[] = "Security/DontEchoError";
analyzer[] = "Security/ShouldUsePreparedStatement";
analyzer[] = "Structures/AddZero";
analyzer[] = "Structures/AlteringForeachWithoutReference";
analyzer[] = "Structures/AssigneAndCompare";
analyzer[] = "Structures/AutoUnsetForeach";
analyzer[] = "Structures/BooleanStrictComparison";
analyzer[] = "Structures/CastingTernary";
analyzer[] = "Structures/CheckJson";
analyzer[] = "Structures/CoalesceAndConcat";
analyzer[] = "Structures/CouldUseDir";
analyzer[] = "Structures/CouldUseShortAssignation";
analyzer[] = "Structures/CouldUseStrrepeat";
analyzer[] = "Structures/DanglingArrayReferences";
analyzer[] = "Structures/DirThenSlash";
analyzer[] = "Structures/DropElseAfterReturn";
analyzer[] = "Structures/ElseIfElseif";
analyzer[] = "Structures/EmptyBlocks";
analyzer[] = "Structures/ErrorReportingWithInteger";
analyzer[] = "Structures/EvalWithoutTry";
analyzer[] = "Structures/ExitUsage";
analyzer[] = "Structures/FailingSubstrComparison";
analyzer[] = "Structures/ForeachReferenceIsNotModified";
analyzer[] = "Structures/ForgottenWhiteSpace";
analyzer[] = "Structures/Htmlentitiescall";
analyzer[] = "Structures/IdenticalConditions";
analyzer[] = "Structures/IdenticalOnBothSides";
analyzer[] = "Structures/IfWithSameConditions";
analyzer[] = "Structures/ImpliedIf";
analyzer[] = "Structures/ImplodeArgsOrder";
analyzer[] = "Structures/IndicesAreIntOrString";
analyzer[] = "Structures/InvalidPackFormat";
analyzer[] = "Structures/InvalidRegex";
analyzer[] = "Structures/IsZero";
analyzer[] = "Structures/ListOmissions";
analyzer[] = "Structures/LogicalMistakes";
analyzer[] = "Structures/LoneBlock";
analyzer[] = "Structures/MbstringThirdArg";
analyzer[] = "Structures/MbstringUnknownEncoding";
```

```
analyzer[] = "Structures/MergeIfThen";
analyzer[] = "Structures/MissingParenthesis";
analyzer[] = "Structures/MultipleDefinedCase";
analyzer[] = "Structures/MultiplyByOne";
analyzer[] = "Structures/NegativePow";
analyzer[] = "Structures/NestedTernary";
analyzer[] = "Structures/NeverNegative";
analyzer[] = "Structures/NextMonthTrap";
analyzer[] = "Structures/NoChoice";
analyzer[] = "Structures/NoEmptyRegex";
analyzer[] = "Structures/NoIssetWithEmpty";
analyzer[] = "Structures/NoParenthesisForLanguageConstruct";
analyzer[] = "Structures/NoReferenceOnLeft";
analyzer[] = "Structures/NoSubstrOne";
analyzer[] = "Structures/Noscream";
analyzer[] = "Structures/NotEqual";
analyzer[] = "Structures/NotNot";
analyzer[] = "Structures/ObjectReferences";
analyzer[] = "Structures/OrDie";
analyzer[] = "Structures/PrintAndDie";
analyzer[] = "Structures/PrintWithoutParenthesis";
analyzer[] = "Structures/PrintfArguments";
analyzer[] = "Structures/RepeatedPrint";
analyzer[] = "Structures/RepeatedRegex";
analyzer[] = "Structures/ResultMayBeMissing";
analyzer[] = "Structures/ReturnTrueFalse";
analyzer[] = "Structures/SameConditions";
analyzer[] = "Structures/ShouldChainException";
analyzer[] = "Structures/ShouldMakeTernary";
analyzer[] = "Structures/ShouldUseExplodeArgs";
analyzer[] = "Structures/StripTagsSkipsClosedTag";
analyzer[] = "Structures/StrposCompare";
analyzer[] = "Structures/SwitchWithoutDefault";
analyzer[] = "Structures/TernaryInConcat";
analyzer[] = "Structures/ThrowsAndAssign";
analyzer[] = "Structures/TimestampDifference";
analyzer[] = "Structures/UncheckedResources";
analyzer[] = "Structures/UnconditionLoopBreak";
analyzer[] = "Structures/UseConstant";
analyzer[] = "Structures/UseInstanceof";
analyzer[] = "Structures/UseSystemTmp";
analyzer[] = "Structures/UselessBrackets";
analyzer[] = "Structures/UselessCasting";
analyzer[] = "Structures/UselessCheck";
analyzer[] = "Structures/UselessInstruction";
analyzer[] = "Structures/UselessParenthesis";
analyzer[] = "Structures/UselessUnset";
```

analyzer[] = “Structures/VardumpUsage”;
analyzer[] = “Structures/WhileListEach”;
analyzer[] = “Structures/pregOptionE”;
analyzer[] = “Traits/UndefinedInsteadof”;
analyzer[] = “Traits/UndefinedTrait”;
analyzer[] = “Traits/UselessAlias”;
analyzer[] = “Type/NoRealComparison”;
analyzer[] = “Type/OneVariableStrings”;
analyzer[] = “Type/ShouldTypecast”;
analyzer[] = “Type/SilentlyCastInteger”;
analyzer[] = “Type/StringWithStrangeSpace”;
analyzer[] = “Typehints/MissingReturntype”;
analyzer[] = “Variables/UndefinedVariable”;

9.5.5 ClassReview

[ClassReview]
analyzer[] = “Classes/AvoidOptionArrays”;
analyzer[] = “Classes/CancelCommonMethod”;
analyzer[] = “Classes/ConstantClass”;
analyzer[] = “Classes/CouldBeAbstractClass”;
analyzer[] = “Classes/CouldBeClassConstant”;
analyzer[] = “Classes/CouldBeFinal”;
analyzer[] = “Classes/CouldBeParentMethod”;
analyzer[] = “Classes/CouldBePrivate”;
analyzer[] = “Classes/CouldBePrivateConstante”;
analyzer[] = “Classes/CouldBePrivateMethod”;
analyzer[] = “Classes/CouldBeProtectedConstant”;
analyzer[] = “Classes/CouldBeProtectedMethod”;
analyzer[] = “Classes/CouldBeProtectedProperty”;
analyzer[] = “Classes/CouldBeStatic”;
analyzer[] = “Classes/CyclicReferences”;
analyzer[] = “Classes/DependantAbstractClass”;
analyzer[] = “Classes/DifferentArgumentCounts”;
analyzer[] = “Classes/DisconnectedClasses”;
analyzer[] = “Classes/FinalPrivate”;
analyzer[] = “Classes/Finalclass”;
analyzer[] = “Classes/Finalmethod”;
analyzer[] = “Classes/FossilizedMethod”;
analyzer[] = “Classes/HiddenNullable”;
analyzer[] = “Classes/InsufficientPropertyTypehint”;
analyzer[] = “Classes/MismatchProperties”;
analyzer[] = “Classes/MissingAbstractMethod”;
analyzer[] = “Classes/MutualExtension”;
analyzer[] = “Classes/NoParent”;
analyzer[] = “Classes/NoSelfReferencingConstant”;

analyzer[] = “Classes/NonNullableSetters”;
analyzer[] = “Classes/PropertyCouldBeLocal”;
analyzer[] = “Classes/RaisedAccessLevel”;
analyzer[] = “Classes/RedefinedProperty”;
analyzer[] = “Classes/ShouldUseSelf”;
analyzer[] = “Classes/UndeclaredStaticProperty”;
analyzer[] = “Classes/UninitedProperty”;
analyzer[] = “Classes/UnreachableConstant”;
analyzer[] = “Classes/UnusedConstant”;
analyzer[] = “Classes/UselessTypehint”;
analyzer[] = “Classes/WrongTypedPropertyInit”;
analyzer[] = “Functions/ExceedingTypehint”;
analyzer[] = “Functions/ModifyTypedParameter”;
analyzer[] = “Functions/NullableWithoutCheck”;
analyzer[] = “Functions/WrongReturnedType”;
analyzer[] = “Interfaces/AvoidSelfInInterface”;
analyzer[] = “Interfaces/IsNotImplemented”;
analyzer[] = “Interfaces/NoGaranteeForPropertyConstant”;
analyzer[] = “Interfaces/UselessInterfaces”;
analyzer[] = “Performances/MemoizeMagicCall”;
analyzer[] = “Php/MissingMagicIsset”;
analyzer[] = “Structures/CouldBeStatic”;
analyzer[] = “Structures/DoubleObjectAssignation”;
analyzer[] = “Traits/SelfUsingTrait”;
analyzer[] = “Traits/UnusedClassTrait”;
analyzer[] = “Variables/NoStaticVarInMethod”;

9.5.6 Coding conventions

[Coding conventions]
analyzer[] = “”;

9.5.7 CompatibilityPHP53

[CompatibilityPHP53]
analyzer[] = “Arrays/ArrayNSUsage”;
analyzer[] = “Arrays/MixedKeys”;
analyzer[] = “Classes/Anonymous”;
analyzer[] = “Classes/CantInheritAbstractMethod”;
analyzer[] = “Classes/ChildRemoveTypehint”;
analyzer[] = “Classes/ConstVisibilityUsage”;
analyzer[] = “Classes/IntegerAsProperty”;
analyzer[] = “Classes/NonStaticMethodsCalledStatic”;
analyzer[] = “Classes/NullOnNew”;
analyzer[] = “Exceptions/MultipleCatch”;
analyzer[] = “Extensions/Extdba”;

```
analyzer[] = “Extensions/Extfdf”;
analyzer[] = “Extensions/Extming”;
analyzer[] = “Functions/GeneratorCannotReturn”;
analyzer[] = “Functions/MultipleSameArguments”;
analyzer[] = “Namespaces/UseFunctionsConstants”;
analyzer[] = “Php/CantUseReturnValueInWriteContext”;
analyzer[] = “Php/CaseForPSS”;
analyzer[] = “Php/ClassConstWithArray”;
analyzer[] = “Php/ClosureThisSupport”;
analyzer[] = “Php/CoalesceEqual”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/ConstWithArray”;
analyzer[] = “Php/DefineWithArray”;
analyzer[] = “Php/DirectCallToClone”;
analyzer[] = “Php/EllipsisUsage”;
analyzer[] = “Php/ExponentUsage”;
analyzer[] = “Php/FlexibleHeredoc”;
analyzer[] = “Php/GroupUseDeclaration”;
analyzer[] = “Php/GroupUseTrailingComma”;
analyzer[] = “Php/HashAlgos53”;
analyzer[] = “Php/HashAlgos71”;
analyzer[] = “Php/ListShortSyntax”;
analyzer[] = “Php/ListWithKeys”;
analyzer[] = “Php/ListWithReference”;
analyzer[] = “Php/MethodCallOnNew”;
analyzer[] = “Php/NoListWithString”;
analyzer[] = “Php/NoReferenceForStaticProperty”;
analyzer[] = “Php/NoReturnForGenerator”;
analyzer[] = “Php/NoStringWithAppend”;
analyzer[] = “Php/NoSubstrMinusOne”;
analyzer[] = “Php/PHP70scalartypehints”;
analyzer[] = “Php/PHP71scalartypehints”;
analyzer[] = “Php/PHP72scalartypehints”;
analyzer[] = “Php/PHP73LastEmptyArgument”;
analyzer[] = “Php/ParenthesisAsParameter”;
analyzer[] = “Php/Php54NewFunctions”;
analyzer[] = “Php/Php55NewFunctions”;
analyzer[] = “Php/Php56NewFunctions”;
analyzer[] = “Php/Php70NewClasses”;
analyzer[] = “Php/Php70NewFunctions”;
analyzer[] = “Php/Php70NewInterfaces”;
analyzer[] = “Php/Php71NewClasses”;
analyzer[] = “Php/Php72NewClasses”;
analyzer[] = “Php/Php73NewFunctions”;
analyzer[] = “Php/Php7RelaxedKeyword”;
analyzer[] = “Php/StaticclassUsage”;
analyzer[] = “Php/TrailingComma”;
```

analyzer[] = “Php/TypedPropertyUsage”;
analyzer[] = “Php/UnicodeEscapePartial”;
analyzer[] = “Php/UnicodeEscapeSyntax”;
analyzer[] = “Php/UnpackingInsideArrays”;
analyzer[] = “Php/UseNullableType”;
analyzer[] = “Php/debugInfoUsage”;
analyzer[] = “Structures/Break0”;
analyzer[] = “Structures/ConstantScalarExpression”;
analyzer[] = “Structures/ContinueIsForLoop”;
analyzer[] = “Structures/DereferencingAS”;
analyzer[] = “Structures/ForeachWithList”;
analyzer[] = “Structures/FunctionSubscripting”;
analyzer[] = “Structures/IssetWithConstant”;
analyzer[] = “Structures/NoGetClassNull”;
analyzer[] = “Structures/PHP7Dirname”;
analyzer[] = “Structures/SwitchWithMultipleDefault”;
analyzer[] = “Structures/VariableGlobal”;
analyzer[] = “Type/Binary”;
analyzer[] = “Type/MalformedOctal”;
analyzer[] = “Variables/Php5IndirectExpression”;
analyzer[] = “Variables/Php7IndirectExpression”;

9.5.8 CompatibilityPHP54

[CompatibilityPHP54]
analyzer[] = “Arrays/MixedKeys”;
analyzer[] = “Classes/Anonymous”;
analyzer[] = “Classes/CantInheritAbstractMethod”;
analyzer[] = “Classes/ChildRemoveTypehint”;
analyzer[] = “Classes/ConstVisibilityUsage”;
analyzer[] = “Classes/IntegerAsProperty”;
analyzer[] = “Classes/NonStaticMethodsCalledStatic”;
analyzer[] = “Classes/NullOnNew”;
analyzer[] = “Exceptions/MultipleCatch”;
analyzer[] = “Extensions/Extmhash”;
analyzer[] = “Functions/GeneratorCannotReturn”;
analyzer[] = “Functions/MultipleSameArguments”;
analyzer[] = “Namespaces/UseFunctionsConstants”;
analyzer[] = “Php/CantUseReturnValueInWriteContext”;
analyzer[] = “Php/CaseForPSS”;
analyzer[] = “Php/ClassConstWithArray”;
analyzer[] = “Php/CoalesceEqual”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/ConstWithArray”;
analyzer[] = “Php/DefineWithArray”;
analyzer[] = “Php/DirectCallToClone”;

```
analyzer[] = “Php/EllipsisUsage”;
analyzer[] = “Php/ExponentUsage”;
analyzer[] = “Php/FlexibleHeredoc”;
analyzer[] = “Php/GroupUseDeclaration”;
analyzer[] = “Php/GroupUseTrailingComma”;
analyzer[] = “Php/HashAlgos53”;
analyzer[] = “Php/HashAlgos54”;
analyzer[] = “Php/HashAlgos71”;
analyzer[] = “Php/ListShortSyntax”;
analyzer[] = “Php/ListWithKeys”;
analyzer[] = “Php/ListWithReference”;
analyzer[] = “Php/NoListWithString”;
analyzer[] = “Php/NoReferenceForStaticProperty”;
analyzer[] = “Php/NoReturnForGenerator”;
analyzer[] = “Php/NoStringWithAppend”;
analyzer[] = “Php/NoSubstrMinusOne”;
analyzer[] = “Php/PHP70scalartypehints”;
analyzer[] = “Php/PHP71scalartypehints”;
analyzer[] = “Php/PHP72scalartypehints”;
analyzer[] = “Php/PHP73LastEmptyArgument”;
analyzer[] = “Php/ParenthesisAsParameter”;
analyzer[] = “Php/Php54RemovedFunctions”;
analyzer[] = “Php/Php55NewFunctions”;
analyzer[] = “Php/Php56NewFunctions”;
analyzer[] = “Php/Php70NewClasses”;
analyzer[] = “Php/Php70NewFunctions”;
analyzer[] = “Php/Php70NewInterfaces”;
analyzer[] = “Php/Php71NewClasses”;
analyzer[] = “Php/Php72NewClasses”;
analyzer[] = “Php/Php73NewFunctions”;
analyzer[] = “Php/Php7RelaxedKeyword”;
analyzer[] = “Php/StaticclassUsage”;
analyzer[] = “Php/TrailingComma”;
analyzer[] = “Php/TypedPropertyUsage”;
analyzer[] = “Php/UnicodeEscapePartial”;
analyzer[] = “Php/UnicodeEscapeSyntax”;
analyzer[] = “Php/UnpackingInsideArrays”;
analyzer[] = “Php/UseNullableType”;
analyzer[] = “Php/debugInfoUsage”;
analyzer[] = “Structures/BreakNonInteger”;
analyzer[] = “Structures/CalltimePassByReference”;
analyzer[] = “Structures/ConstantScalarExpression”;
analyzer[] = “Structures/ContinueIsForLoop”;
analyzer[] = “Structures/CryptWithoutSalt”;
analyzer[] = “Structures/DereferencingAS”;
analyzer[] = “Structures/ForeachWithList”;
analyzer[] = “Structures/IssetWithConstant”;
```

analyzer[] = “Structures/NoGetClassNull”;
analyzer[] = “Structures/PHP7Dirname”;
analyzer[] = “Structures/SwitchWithMultipleDefault”;
analyzer[] = “Structures/VariableGlobal”;
analyzer[] = “Type/MalformedOctal”;
analyzer[] = “Variables/Php5IndirectExpression”;
analyzer[] = “Variables/Php7IndirectExpression”;

9.5.9 CompatibilityPHP55

[CompatibilityPHP55]
analyzer[] = “Classes/Anonymous”;
analyzer[] = “Classes/CantInheritAbstractMethod”;
analyzer[] = “Classes/ChildRemoveTypehint”;
analyzer[] = “Classes/ConstVisibilityUsage”;
analyzer[] = “Classes/IntegerAsProperty”;
analyzer[] = “Classes/NonStaticMethodsCalledStatic”;
analyzer[] = “Classes/NullOnNew”;
analyzer[] = “Exceptions/MultipleCatch”;
analyzer[] = “Extensions/Extapc”;
analyzer[] = “Extensions/Extmysql”;
analyzer[] = “Functions/GeneratorCannotReturn”;
analyzer[] = “Functions/MultipleSameArguments”;
analyzer[] = “Namespaces/UseFunctionsConstants”;
analyzer[] = “Php/ClassConstWithArray”;
analyzer[] = “Php/CoalesceEqual”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/ConstWithArray”;
analyzer[] = “Php/DefineWithArray”;
analyzer[] = “Php/DirectCallToClone”;
analyzer[] = “Php/EllipsisUsage”;
analyzer[] = “Php/ExponentUsage”;
analyzer[] = “Php/FlexibleHeredoc”;
analyzer[] = “Php/GroupUseDeclaration”;
analyzer[] = “Php/GroupUseTrailingComma”;
analyzer[] = “Php/HashAlgos53”;
analyzer[] = “Php/HashAlgos54”;
analyzer[] = “Php/HashAlgos71”;
analyzer[] = “Php/ListShortSyntax”;
analyzer[] = “Php/ListWithKeys”;
analyzer[] = “Php/ListWithReference”;
analyzer[] = “Php/NoListWithString”;
analyzer[] = “Php/NoReferenceForStaticProperty”;
analyzer[] = “Php/NoReturnForGenerator”;
analyzer[] = “Php/NoStringWithAppend”;
analyzer[] = “Php/NoSubstrMinusOne”;

```
analyzer[] = "Php/PHP70scalartypehints";
analyzer[] = "Php/PHP71scalartypehints";
analyzer[] = "Php/PHP72scalartypehints";
analyzer[] = "Php/PHP73LastEmptyArgument";
analyzer[] = "Php/ParenthesisAsParameter";
analyzer[] = "Php/Password55";
analyzer[] = "Php/Php55RemovedFunctions";
analyzer[] = "Php/Php56NewFunctions";
analyzer[] = "Php/Php70NewClasses";
analyzer[] = "Php/Php70NewFunctions";
analyzer[] = "Php/Php70NewInterfaces";
analyzer[] = "Php/Php71NewClasses";
analyzer[] = "Php/Php72NewClasses";
analyzer[] = "Php/Php73NewFunctions";
analyzer[] = "Php/Php7RelaxedKeyword";
analyzer[] = "Php/TrailingComma";
analyzer[] = "Php/TypedPropertyUsage";
analyzer[] = "Php/UnicodeEscapePartial";
analyzer[] = "Php/UnicodeEscapeSyntax";
analyzer[] = "Php/UnpackingInsideArrays";
analyzer[] = "Php/UseNullableType";
analyzer[] = "Php/debugInfoUsage";
analyzer[] = "Structures/ConstantScalarExpression";
analyzer[] = "Structures/ContinueIsForLoop";
analyzer[] = "Structures/IssetWithConstant";
analyzer[] = "Structures/NoGetClassNull";
analyzer[] = "Structures/PHP7Dirname";
analyzer[] = "Structures/SwitchWithMultipleDefault";
analyzer[] = "Structures/VariableGlobal";
analyzer[] = "Type/MalformedOctal";
analyzer[] = "Variables/Php5IndirectExpression";
analyzer[] = "Variables/Php7IndirectExpression";
```

9.5.10 CompatibilityPHP56

```
[CompatibilityPHP56]
    analyzer[] = "Classes/Anonymous";
    analyzer[] = "Classes/CantInheritAbstractMethod";
    analyzer[] = "Classes/ChildRemoveTypehint";
    analyzer[] = "Classes/ConstVisibilityUsage";
    analyzer[] = "Classes/IntegerAsProperty";
    analyzer[] = "Classes/NonStaticMethodsCalledStatic";
    analyzer[] = "Classes/NullOnNew";
    analyzer[] = "Exceptions/MultipleCatch";
    analyzer[] = "Functions/GeneratorCannotReturn";
    analyzer[] = "Functions/MultipleSameArguments";
```

```
analyzer[] = "Php/CoalesceEqual";
analyzer[] = "Php/ConcatAndAddition";
analyzer[] = "Php/DefineWithArray";
analyzer[] = "Php/DirectCallToClone";
analyzer[] = "Php/FlexibleHeredoc";
analyzer[] = "Php/GroupUseDeclaration";
analyzer[] = "Php/GroupUseTrailingComma";
analyzer[] = "Php/HashAlgos53";
analyzer[] = "Php/HashAlgos54";
analyzer[] = "Php/HashAlgos71";
analyzer[] = "Php/ListShortSyntax";
analyzer[] = "Php/ListWithKeys";
analyzer[] = "Php/ListWithReference";
analyzer[] = "Php/NoListWithString";
analyzer[] = "Php/NoReferenceForStaticProperty";
analyzer[] = "Php/NoReturnForGenerator";
analyzer[] = "Php/NoStringWithAppend";
analyzer[] = "Php/NoSubstrMinusOne";
analyzer[] = "Php/PHP70scalartypehints";
analyzer[] = "Php/PHP71scalartypehints";
analyzer[] = "Php/PHP72scalartypehints";
analyzer[] = "Php/PHP73LastEmptyArgument";
analyzer[] = "Php/ParenthesisAsParameter";
analyzer[] = "Php/Php70NewClasses";
analyzer[] = "Php/Php70NewFunctions";
analyzer[] = "Php/Php70NewInterfaces";
analyzer[] = "Php/Php71NewClasses";
analyzer[] = "Php/Php72NewClasses";
analyzer[] = "Php/Php73NewFunctions";
analyzer[] = "Php/Php7RelaxedKeyword";
analyzer[] = "Php/Php80OnlyTypeHints";
analyzer[] = "Php/RawPostDataUsage";
analyzer[] = "Php/TrailingComma";
analyzer[] = "Php/TypedPropertyUsage";
analyzer[] = "Php/UnicodeEscapePartial";
analyzer[] = "Php/UnicodeEscapeSyntax";
analyzer[] = "Php/UnpackingInsideArrays";
analyzer[] = "Php/UseNullableType";
analyzer[] = "Structures/ContinueIsForLoop";
analyzer[] = "Structures/IssetWithConstant";
analyzer[] = "Structures/NoGetClassNull";
analyzer[] = "Structures/PHP7Dirname";
analyzer[] = "Structures/SwitchWithMultipleDefault";
analyzer[] = "Structures/VariableGlobal";
analyzer[] = "Type/MalformedOctal";
analyzer[] = "Variables/Php5IndirectExpression";
analyzer[] = "Variables/Php7IndirectExpression";
```

9.5.11 CompatibilityPHP70

[CompatibilityPHP70]
analyzer[] = “Classes/CantInheritAbstractMethod”;
analyzer[] = “Classes/ChildRemoveTypehint”;
analyzer[] = “Classes/ConstVisibilityUsage”;
analyzer[] = “Classes/IntegerAsProperty”;
analyzer[] = “Classes/toStringPss”;
analyzer[] = “Exceptions/MultipleCatch”;
analyzer[] = “Extensions/Extereg”;
analyzer[] = “Functions/funcGetArgModified”;
analyzer[] = “Php/CoalesceEqual”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/EmptyList”;
analyzer[] = “Php/FlexibleHeredoc”;
analyzer[] = “Php/ForeachDontChangePointer”;
analyzer[] = “Php/GlobalWithoutSimpleVariable”;
analyzer[] = “Php/GroupUseTrailingComma”;
analyzer[] = “Php/HashAlgos53”;
analyzer[] = “Php/HashAlgos54”;
analyzer[] = “Php/HashAlgos71”;
analyzer[] = “Php/ListShortSyntax”;
analyzer[] = “Php/ListWithAppends”;
analyzer[] = “Php/ListWithKeys”;
analyzer[] = “Php/ListWithReference”;
analyzer[] = “Php/NoReferenceForStaticProperty”;
analyzer[] = “Php/NoSubstrMinusOne”;
analyzer[] = “Php/PHP71scalartypehints”;
analyzer[] = “Php/PHP72scalartypehints”;
analyzer[] = “Php/PHP73LastEmptyArgument”;
analyzer[] = “Php/Php70RemovedDirective”;
analyzer[] = “Php/Php70RemovedFunctions”;
analyzer[] = “Php/Php71NewClasses”;
analyzer[] = “Php/Php72NewClasses”;
analyzer[] = “Php/Php73NewFunctions”;
analyzer[] = “Php/Php80OnlyTypeHints”;
analyzer[] = “Php/Php80UnionTypehint”;
analyzer[] = “Php/ReservedKeywords7”;
analyzer[] = “Php/SetExceptionHandlerPHP7”;
analyzer[] = “Php/TrailingComma”;
analyzer[] = “Php/TypedPropertyUsage”;
analyzer[] = “Php/UnpackingInsideArrays”;
analyzer[] = “Php/UseNullableType”;
analyzer[] = “Php/UsortSorting”;
analyzer[] = “Structures/BreakOutsideLoop”;
analyzer[] = “Structures/ContinueIsForLoop”;
analyzer[] = “Structures/McryptcreateivWithoutOption”;

analyzer[] = “Structures/NoGetClassNull”;
analyzer[] = “Structures/SetlocaleNeedsConstants”;
analyzer[] = “Structures/pregOptionE”;
analyzer[] = “Type/HexadecimalString”;
analyzer[] = “Variables/Php7IndirectExpression”;

9.5.12 CompatibilityPHP71

[CompatibilityPHP71]
analyzer[] = “Arrays/StringInitialization”;
analyzer[] = “Classes/CantInheritAbstractMethod”;
analyzer[] = “Classes/ChildRemoveTypehint”;
analyzer[] = “Classes/IntegerAsProperty”;
analyzer[] = “Classes/UsingThisOutsideAClass”;
analyzer[] = “Extensions/Extmcrypt”;
analyzer[] = “Php/BetterRand”;
analyzer[] = “Php/CoalesceEqual”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/FlexibleHeredoc”;
analyzer[] = “Php/GroupUseTrailingComma”;
analyzer[] = “Php/HashAlgos53”;
analyzer[] = “Php/HashAlgos54”;
analyzer[] = “Php/ListWithReference”;
analyzer[] = “Php/NoReferenceForStaticProperty”;
analyzer[] = “Php/PHP72scalartypehints”;
analyzer[] = “Php/PHP73LastEmptyArgument”;
analyzer[] = “Php/Php70RemovedDirective”;
analyzer[] = “Php/Php70RemovedFunctions”;
analyzer[] = “Php/Php71NewFunctions”;
analyzer[] = “Php/Php71RemovedDirective”;
analyzer[] = “Php/Php71microseconds”;
analyzer[] = “Php/Php72NewClasses”;
analyzer[] = “Php/Php73NewFunctions”;
analyzer[] = “Php/Php80OnlyTypeHints”;
analyzer[] = “Php/Php80UnionTypehint”;
analyzer[] = “Php/SignatureTrailingComma”;
analyzer[] = “Php/TrailingComma”;
analyzer[] = “Php/TypedPropertyUsage”;
analyzer[] = “Php/UnpackingInsideArrays”;
analyzer[] = “Structures/ContinueIsForLoop”;
analyzer[] = “Structures/NoGetClassNull”;
analyzer[] = “Structures/NoSubstrOne”;
analyzer[] = “Structures/pregOptionE”;
analyzer[] = “Type/HexadecimalString”;
analyzer[] = “Type/OctalInString”;

9.5.13 CompatibilityPHP72

[CompatibilityPHP72]
analyzer[] = “Constants/UndefinedConstants”;
analyzer[] = “Php/AvoidSetErrorHandlerContextArg”;
analyzer[] = “Php/CoalesceEqual”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/FlexibleHeredoc”;
analyzer[] = “Php/HashAlgos53”;
analyzer[] = “Php/HashAlgos54”;
analyzer[] = “Php/HashUsesObjects”;
analyzer[] = “Php/ListWithReference”;
analyzer[] = “Php/NoReferenceForStaticProperty”;
analyzer[] = “Php/PHP73LastEmptyArgument”;
analyzer[] = “Php/Php72Deprecation”;
analyzer[] = “Php/Php72NewClasses”;
analyzer[] = “Php/Php72NewConstants”;
analyzer[] = “Php/Php72NewFunctions”;
analyzer[] = “Php/Php72ObjectKeyword”;
analyzer[] = “Php/Php72RemovedFunctions”;
analyzer[] = “Php/Php73NewFunctions”;
analyzer[] = “Php/Php80OnlyTypeHints”;
analyzer[] = “Php/Php80UnionTypehint”;
analyzer[] = “Php/SignatureTrailingComma”;
analyzer[] = “Php/ThrowWasAnExpression”;
analyzer[] = “Php/TrailingComma”;
analyzer[] = “Php/TypedPropertyUsage”;
analyzer[] = “Php/UnpackingInsideArrays”;
analyzer[] = “Structures/CanCountNonCountable”;
analyzer[] = “Structures/ContinueIsForLoop”;
analyzer[] = “Structures/NoGetClassNull”;
analyzer[] = “Structures/pregOptionE”;

9.5.14 CompatibilityPHP73

[CompatibilityPHP73]
analyzer[] = “Constants/CaseInsensitiveConstants”;
analyzer[] = “Php/AssertFunctionIsReserved”;
analyzer[] = “Php/CoalesceEqual”;
analyzer[] = “Php/CompactInexistant”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/IntegerSeparatorUsage”;
analyzer[] = “Php/Php73NewFunctions”;
analyzer[] = “Php/Php73RemovedFunctions”;
analyzer[] = “Php/Php74NewDirective”;
analyzer[] = “Php/Php80OnlyTypeHints”;
analyzer[] = “Php/Php80UnionTypehint”;

analyzer[] = “Php/SignatureTrailingComma”;
analyzer[] = “Php/ThrowWasAnExpression”;
analyzer[] = “Php/TypedPropertyUsage”;
analyzer[] = “Php/UnknownPcre2Option”;
analyzer[] = “Php/UnpackingInsideArrays”;
analyzer[] = “Structures/ContinueIsForLoop”;
analyzer[] = “Structures/DontReadAndWriteInOneExpression”;

9.5.15 CompatibilityPHP74

[CompatibilityPHP74]
analyzer[] = “Functions/UnbindingClosures”;
analyzer[] = “Php/ArrayKeyExistsWithObjects”;
analyzer[] = “Php/AvoidGetobjectVars”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/DetectCurrentClass”;
analyzer[] = “Php/FilterToAddSlashes”;
analyzer[] = “Php/HashAlgos74”;
analyzer[] = “Php/IdnUts46”;
analyzer[] = “Php/NestedTernaryWithoutParenthesis”;
analyzer[] = “Php/NoMoreCurlyArrays”;
analyzer[] = “Php/Php74Deprecation”;
analyzer[] = “Php/Php74NewClasses”;
analyzer[] = “Php/Php74NewConstants”;
analyzer[] = “Php/Php74NewFunctions”;
analyzer[] = “Php/Php74RemovedDirective”;
analyzer[] = “Php/Php74RemovedFunctions”;
analyzer[] = “Php/Php74ReservedKeyword”;
analyzer[] = “Php/Php74mbstrrpos3rdArg”;
analyzer[] = “Php/Php80NewFunctions”;
analyzer[] = “Php/Php80OnlyTypeHints”;
analyzer[] = “Php/Php80UnionTypehint”;
analyzer[] = “Php/Php80VariableSyntax”;
analyzer[] = “Php/ReflectionExportIsDeprecated”;
analyzer[] = “Php/ScalarAreNotArrays”;
analyzer[] = “Php/SignatureTrailingComma”;
analyzer[] = “Php/ThrowWasAnExpression”;
analyzer[] = “Php/UseMatch”;
analyzer[] = “Structures/CurlVersionNow”;
analyzer[] = “Structures/DontReadAndWriteInOneExpression”;
analyzer[] = “Structures/OpensslRandomPseudoByteSecondArg”;

9.5.16 CompatibilityPHP80

[CompatibilityPHP80]
analyzer[] = “Arrays/NegativeStart”;

analyzer[] = “Classes/FinalPrivate”;
analyzer[] = “Classes/OldStyleConstructor”;
analyzer[] = “Functions/MismatchParameterName”;
analyzer[] = “Functions/NullableWithConstant”;
analyzer[] = “Functions/WrongOptionalParameter”;
analyzer[] = “Php/AvoidGetobjectVars”;
analyzer[] = “Php/CastUnsetUsage”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/Php80NamedParameterVariadic”;
analyzer[] = “Php/Php80RemovedConstant”;
analyzer[] = “Php/Php80RemovedDirective”;
analyzer[] = “Php/Php80RemovedFunctions”;
analyzer[] = “Php/Php80RemovesResources”;
analyzer[] = “Php/PhpErrorMsgUsage”;
analyzer[] = “Php/ReservedMatchKeyword”;
analyzer[] = “Structures/ArrayMapPassesByValue”;
analyzer[] = “Structures/UnsupportedTypesWithOperators”;

9.5.17 CompatibilityPHP81

[CompatibilityPHP81]
analyzer[] = “”;

9.5.18 Dead code

[Dead code]
analyzer[] = “Classes/CantExtendFinal”;
analyzer[] = “Classes/LocallyUnusedProperty”;
analyzer[] = “Classes/UnresolvedCatch”;
analyzer[] = “Classes/UnresolvedInstanceof”;
analyzer[] = “Classes/UnusedClass”;
analyzer[] = “Classes/UnusedMethods”;
analyzer[] = “Classes/UnusedPrivateMethod”;
analyzer[] = “Classes/UnusedPrivateProperty”;
analyzer[] = “Classes/UnusedProtectedMethods”;
analyzer[] = “Constants/UnusedConstants”;
analyzer[] = “Exceptions/AlreadyCaught”;
analyzer[] = “Exceptions/CaughtButNotThrown”;
analyzer[] = “Exceptions/Rethrown”;
analyzer[] = “Exceptions/Unthrown”;
analyzer[] = “Functions/UnusedFunctions”;
analyzer[] = “Functions/UnusedInheritedVariable”;
analyzer[] = “Functions/UnusedReturnedValue”;
analyzer[] = “Functions/UselessTypeCheck”;
analyzer[] = “Interfaces/UnusedInterfaces”;
analyzer[] = “Namespaces/EmptyNamespace”;

analyzer[] = “Namespaces/UnusedUse”;
analyzer[] = “Structures/EmptyLines”;
analyzer[] = “Structures/UnreachableCode”;
analyzer[] = “Structures/UnsetInForeach”;
analyzer[] = “Structures/UnusedLabel”;
analyzer[] = “Traits/SelfUsingTrait”;

9.5.19 LintButWontExec

[LintButWontExec]
analyzer[] = “Classes/AbstractOrImplements”;
analyzer[] = “Classes/CloneWithNonObject”;
analyzer[] = “Classes/CouldBeStringable”;
analyzer[] = “Classes/Finalclass”;
analyzer[] = “Classes/Finalmethod”;
analyzer[] = “Classes/IncompatibleSignature”;
analyzer[] = “Classes/MethodSignatureMustBeCompatible”;
analyzer[] = “Classes/MismatchProperties”;
analyzer[] = “Classes/MutualExtension”;
analyzer[] = “Classes/NoMagicWithArray”;
analyzer[] = “Classes/NoPSSOutsideClass”;
analyzer[] = “Classes/NoSelfReferencingConstant”;
analyzer[] = “Classes/RaisedAccessLevel”;
analyzer[] = “Classes/UsingThisOutsideAClass”;
analyzer[] = “Classes/WrongTypedPropertyInit”;
analyzer[] = “Exceptions/CantThrow”;
analyzer[] = “Functions/MismatchTypeAndDefault”;
analyzer[] = “Functions/MustReturn”;
analyzer[] = “Functions/OnlyVariableForReference”;
analyzer[] = “Functions/TypehintMustBeReturned”;
analyzer[] = “Interfaces/CantImplementTraversable”;
analyzer[] = “Interfaces/ConcreteVisibility”;
analyzer[] = “Interfaces/IsNotImplemented”;
analyzer[] = “Interfaces/RepeatedInterface”;
analyzer[] = “Php/OnlyVariableForReference”;
analyzer[] = “Traits/MethodCollisionTraits”;
analyzer[] = “Traits/TraitNotFound”;
analyzer[] = “Traits/UndefinedInsteadof”;
analyzer[] = “Traits/UndefinedTrait”;
analyzer[] = “Traits/UselessAlias”;

9.5.20 Performances

[Performances]
analyzer[] = “Arrays/GettingLastElement”;
analyzer[] = “Arrays/SliceFirst”;

analyzer[] = “Classes/MakeMagicConcrete”;
analyzer[] = “Classes/UseClassOperator”;
analyzer[] = “Functions/Closure2String”;
analyzer[] = “Performances/ArrayKeyExistsSpeedup”;
analyzer[] = “Performances/ArrayMergeInLoops”;
analyzer[] = “Performances/Autoappend”;
analyzer[] = “Performances/AvoidArrayPush”;
analyzer[] = “Performances/CacheVariableOutsideLoop”;
analyzer[] = “Performances/CsvInLoops”;
analyzer[] = “Performances/DoInBase”;
analyzer[] = “Performances/DoubleArrayFlip”;
analyzer[] = “Performances/FetchOneRowFormat”;
analyzer[] = “Performances/IssetWholeArray”;
analyzer[] = “Performances/JoinFile”;
analyzer[] = “Performances/MakeOneCall”;
analyzer[] = “Performances/MbStringInLoop”;
analyzer[] = “Performances/NoConcatInLoop”;
analyzer[] = “Performances/NoGlob”;
analyzer[] = “Performances/NotCountNull”;
analyzer[] = “Performances/OptimizeExplode”;
analyzer[] = “Performances/PHP7EncapsedStrings”;
analyzer[] = “Performances/Php74ArrayKeyExists”;
analyzer[] = “Performances/PrePostIncrement”;
analyzer[] = “Performances/RegexOnArrays”;
analyzer[] = “Performances/RegexOnCollector”;
analyzer[] = “Performances/SimpleSwitch”;
analyzer[] = “Performances/SlowFunctions”;
analyzer[] = “Performances/SubstrFirst”;
analyzer[] = “Performances/UseBlindVar”;
analyzer[] = “Performances/timeVsstrtotime”;
analyzer[] = “Php/ShouldUseArrayColumn”;
analyzer[] = “Php/ShouldUseFunction”;
analyzer[] = “Php/UsePathinfoArgs”;
analyzer[] = “Structures/CouldUseShortAssignation”;
analyzer[] = “Structures/EchoWithConcat”;
analyzer[] = “Structures/EvalUsage”;
analyzer[] = “Structures/ForWithFunctioncall”;
analyzer[] = “Structures/GlobalOutsideLoop”;
analyzer[] = “Structures/NoArrayUnique”;
analyzer[] = “Structures/NoAssignationInFunction”;
analyzer[] = “Structures/NoSubstrOne”;
analyzer[] = “Structures/Noscream”;
analyzer[] = “Structures/SimplePreg”;
analyzer[] = “Structures/WhileListEach”;

9.5.21 Rector

[Rector]
analyzer[] = “Php/IsAWithString”;
analyzer[] = “Structures/ElseIfElseif”;
analyzer[] = “Structures/ShouldPreprocess”;

9.5.22 Security

[Security]
analyzer[] = “Functions/HardcodedPasswords”;
analyzer[] = “Php/BetterRand”;
analyzer[] = “Security/AnchorRegex”;
analyzer[] = “Security/AvoidThoseCrypto”;
analyzer[] = “Security/CompareHash”;
analyzer[] = “Security/ConfigureExtract”;
analyzer[] = “Security/CryptoKeyLength”;
analyzer[] = “Security/CurlOptions”;
analyzer[] = “Security/DirectInjection”;
analyzer[] = “Security/DontEchoError”;
analyzer[] = “Security/DynamicDl”;
analyzer[] = “Security/EncodedLetters”;
analyzer[] = “Security/FilterInputSource”;
analyzer[] = “Security/IndirectInjection”;
analyzer[] = “Security/IntegerConversion”;
analyzer[] = “Security/KeepFilesRestricted”;
analyzer[] = “Security/MinusOneOnError”;
analyzer[] = “Security/MkdirDefault”;
analyzer[] = “Security/MoveUploadedFile”;
analyzer[] = “Security/NoEntIgnore”;
analyzer[] = “Security/NoNetForXmlLoad”;
analyzer[] = “Security/NoSleep”;
analyzer[] = “Security/NoWeakSSLCrypto”;
analyzer[] = “Security/RegisterGlobals”;
analyzer[] = “Security/SafeHttpHeaders”;
analyzer[] = “Security/SessionLazyWrite”;
analyzer[] = “Security/SetCookieArgs”;
analyzer[] = “Security/ShouldUsePreparedStatement”;
analyzer[] = “Security/ShouldUseSessionRegenerateId”;
analyzer[] = “Security/Sqlite3RequiresSingleQuotes”;
analyzer[] = “Security/UnserializeSecondArg”;
analyzer[] = “Security/UploadFilenameInjection”;
analyzer[] = “Security/parseUrlWithoutParameters”;
analyzer[] = “Structures/EvalUsage”;
analyzer[] = “Structures/EvalWithoutTry”;
analyzer[] = “Structures/Fallthrough”;
analyzer[] = “Structures/NoHardcodedHash”;

analyzer[] = “Structures/NoHardcodedIp”;
analyzer[] = “Structures/NoHardcodedPort”;
analyzer[] = “Structures/NoReturnInFinally”;
analyzer[] = “Structures/PhpinfoUsage”;
analyzer[] = “Structures/RandomWithoutTry”;
analyzer[] = “Structures/VardumpUsage”;
analyzer[] = “Structures/pregOptionE”;

9.5.23 Semantics

[Semantics]
analyzer[] = “Arrays/WeirdIndex”;
analyzer[] = “Functions/FnArgumentVariableConfusion”;
analyzer[] = “Functions/MismatchParameterAndType”;
analyzer[] = “Functions/OneLetterFunctions”;
analyzer[] = “Functions/ParameterHiding”;
analyzer[] = “Functions/PrefixToType”;
analyzer[] = “Functions/SemanticTyping”;
analyzer[] = “Functions/WrongTypehintedName”;
analyzer[] = “Php/ClassFunctionConfusion”;
analyzer[] = “Structures/PropertyVariableConfusion”;
analyzer[] = “Type/DuplicateLiteral”;
analyzer[] = “Type/SimilarIntegers”;
analyzer[] = “Variables/VariableOneLetter”;

9.5.24 Suggestions

[Suggestions]
analyzer[] = “Arrays/RandomlySortedLiterals”;
analyzer[] = “Arrays/ShouldPreprocess”;
analyzer[] = “Arrays/SliceFirst”;
analyzer[] = “Classes/CancelCommonMethod”;
analyzer[] = “Classes/ParentFirst”;
analyzer[] = “Classes/ShouldDeepClone”;
analyzer[] = “Classes/ShouldHaveDestructor”;
analyzer[] = “Classes/ShouldUseSelf”;
analyzer[] = “Classes/TooManyChildren”;
analyzer[] = “Classes/UnitializedProperties”;
analyzer[] = “Classes/UselessTypehint”;
analyzer[] = “Constants/CouldBeConstant”;
analyzer[] = “Exceptions/CouldUseTry”;
analyzer[] = “Exceptions/LargeTryBlock”;
analyzer[] = “Exceptions/LongPreparation”;
analyzer[] = “Exceptions/OverwriteException”;
analyzer[] = “Exceptions/UnusedExceptionVariable”;
analyzer[] = “Functions/AddDefaultValue”;

```
analyzer[] = "Functions/Closure2String";
analyzer[] = "Functions/CouldBeStaticClosure";
analyzer[] = "Functions/CouldCentralize";
analyzer[] = "Functions/NeverUsedParameter";
analyzer[] = "Functions/NoReturnUsed";
analyzer[] = "Functions/TooManyParameters";
analyzer[] = "Functions/TooMuchIndented";
analyzer[] = "Functions/UselessDefault";
analyzer[] = "Interfaces/AlreadyParentsInterface";
analyzer[] = "Interfaces/UnusedInterfaces";
analyzer[] = "Namespaces/AliasConfusion";
analyzer[] = "Namespaces/CouldUseAlias";
analyzer[] = "Patterns/AbstractAway";
analyzer[] = "Performances/ArrayKeyExistsSpeedup";
analyzer[] = "Performances/IssetWholeArray";
analyzer[] = "Performances/SubstrFirst";
analyzer[] = "Php/AvoidReal";
analyzer[] = "Php/CompactInexistant";
analyzer[] = "Php/CouldUseIsCountable";
analyzer[] = "Php/CouldUsePromotedProperties";
analyzer[] = "Php/DetectCurrentClass";
analyzer[] = "Php/ImplodeOneArg";
analyzer[] = "Php/IssetMultipleArgs";
analyzer[] = "Php/LogicalInLetters";
analyzer[] = "Php/NewExponent";
analyzer[] = "Php/PregMatchAllFlag";
analyzer[] = "Php/ReturnWithParenthesis";
analyzer[] = "Php/ShouldPreprocess";
analyzer[] = "Php/ShouldUseArrayColumn";
analyzer[] = "Php/ShouldUseArrayFilter";
analyzer[] = "Php/ShouldUseCoalesce";
analyzer[] = "Php/UseDateTimeImmutable";
analyzer[] = "Php/UseGetDebugType";
analyzer[] = "Php/UseSessionStartOptions";
analyzer[] = "Php/UseStrContains";
analyzer[] = "Structures/ArraySearchMultipleKeys";
analyzer[] = "Structures/BasenameSuffix";
analyzer[] = "Structures/BooleanStrictComparison";
analyzer[] = "Structures/CouldUseArrayFillKeys";
analyzer[] = "Structures/CouldUseArrayUnique";
analyzer[] = "Structures/CouldUseCompact";
analyzer[] = "Structures/CouldUseDir";
analyzer[] = "Structures/CouldUseMatch";
analyzer[] = "Structures/DeclareStaticOnce";
analyzer[] = "Structures/DirectlyUseFile";
analyzer[] = "Structures/DontCompareTypedBoolean";
analyzer[] = "Structures/DontLoopOnYield";
```

analyzer[] = “Structures/DropElseAfterReturn”;
analyzer[] = “Structures/EchoWithConcat”;
analyzer[] = “Structures/EmptyWithExpression”;
analyzer[] = “Structures/FunctionPreSubscripting”;
analyzer[] = “Structures/JsonWithOption”;
analyzer[] = “Structures/ListOmissions”;
analyzer[] = “Structures/LongBlock”;
analyzer[] = “Structures/MismatchedTernary”;
analyzer[] = “Structures/MultipleUnset”;
analyzer[] = “Structures/NamedRegex”;
analyzer[] = “Structures/NoNeedGetClass”;
analyzer[] = “Structures/NoParenthesisForLanguageConstruct”;
analyzer[] = “Structures/NoSubstrOne”;
analyzer[] = “Structures/OneIfIsSufficient”;
analyzer[] = “Structures/PHP7Dirname”;
analyzer[] = “Structures/PossibleIncrement”;
analyzer[] = “Structures/RepeatedPrint”;
analyzer[] = “Structures/ReuseVariable”;
analyzer[] = “Structures/SGVariablesConfusion”;
analyzer[] = “Structures/SetAside”;
analyzer[] = “Structures/ShouldUseForeach”;
analyzer[] = “Structures/ShouldUseMath”;
analyzer[] = “Structures/ShouldUseOperator”;
analyzer[] = “Structures/SubstrLastArg”;
analyzer[] = “Structures/SubstrToTrim”;
analyzer[] = “Structures/UnreachableCode”;
analyzer[] = “Structures/UseArrayFunctions”;
analyzer[] = “Structures/UseCaseValue”;
analyzer[] = “Structures/UseCountRecursive”;
analyzer[] = “Structures/UseListWithForeach”;
analyzer[] = “Structures/UseUrlQueryFunctions”;
analyzer[] = “Structures/WhileListEach”;
analyzer[] = “Traits/MultipleUsage”;
analyzer[] = “Variables/ComplexDynamicNames”;
analyzer[] = “Variables/NoStaticVarInMethod”;

9.5.25 Top10

[Top10]
analyzer[] = “Classes/DontUnsetProperties”;
analyzer[] = “Classes/UnitializedProperties”;
analyzer[] = “Classes/UnresolvedInstanceof”;
analyzer[] = “Constants/ConstRecommended”;
analyzer[] = “Functions/ShouldYieldWithKey”;
analyzer[] = “Performances/ArrayMergeInLoops”;
analyzer[] = “Performances/CsvInLoops”;

analyzer[] = “Performances/NoConcatInLoop”;
analyzer[] = “Performances/SubstrFirst”;
analyzer[] = “Php/AvoidReal”;
analyzer[] = “Php/ConcatAndAddition”;
analyzer[] = “Php/LetterCharsLogicalFavorite”;
analyzer[] = “Php/LogicalInLetters”;
analyzer[] = “Php/MissingSubpattern”;
analyzer[] = “Structures/CouldUseStrrepeat”;
analyzer[] = “Structures/DanglingArrayReferences”;
analyzer[] = “Structures/FailingSubstrComparison”;
analyzer[] = “Structures/ForWithFunctioncall”;
analyzer[] = “Structures/NextMonthTrap”;
analyzer[] = “Structures/NoChoice”;
analyzer[] = “Structures/NoSubstrOne”;
analyzer[] = “Structures/ObjectReferences”;
analyzer[] = “Structures/QueriesInLoop”;
analyzer[] = “Structures/RepeatedPrint”;
analyzer[] = “Structures/StrposCompare”;
analyzer[] = “Structures/UseListWithForeach”;
analyzer[] = “Type/NoRealComparison”;
analyzer[] = “Variables/VariableUsedOnce”;

9.5.26 Typechecks

[Typechecks]
analyzer[] = “Classes/ChildRemoveTypehint”;
analyzer[] = “Classes/FossilizedMethod”;
analyzer[] = “Functions/BadTypehintRelay”;
analyzer[] = “Functions/InsufficientTypehint”;
analyzer[] = “Functions/MismatchTypeAndDefault”;
analyzer[] = “Functions/MismatchedDefaultArguments”;
analyzer[] = “Functions/MismatchedTypehint”;
analyzer[] = “Functions/MissingTypehint”;
analyzer[] = “Functions/NoClassAsTypehint”;
analyzer[] = “Functions/ShouldBeTypehinted”;
analyzer[] = “Functions/WrongArgumentType”;
analyzer[] = “Functions/WrongTypeWithCall”;
analyzer[] = “Interfaces/UselessInterfaces”;
analyzer[] = “Php/NotScalarType”;
analyzer[] = “Typehints/CouldBeCallable”;
analyzer[] = “Typehints/CouldBeFloat”;
analyzer[] = “Typehints/CouldBeInt”;
analyzer[] = “Typehints/CouldBeIterable”;
analyzer[] = “Typehints/CouldBeNull”;
analyzer[] = “Typehints/CouldBeParent”;
analyzer[] = “Typehints/CouldBeSelf”;

analyzer[] = “Typehints/CouldBeString”;
analyzer[] = “Typehints/CouldBeVoid”;

9.5.27 php-cs-fixable

[php-cs-fixable]
analyzer[] = “Classes/DontUnsetProperties”;
analyzer[] = “Php/ImplodeOneArg”;
analyzer[] = “Php/IsnullVsEqualNull”;
analyzer[] = “Php/IssetMultipleArgs”;
analyzer[] = “Php/LogicalInLetters”;
analyzer[] = “Php/NewExponent”;
analyzer[] = “Structures/CouldUseDir”;
analyzer[] = “Structures/ElseIfElseif”;
analyzer[] = “Structures/MultipleUnset”;
analyzer[] = “Structures/PHP7Dirname”;
analyzer[] = “Structures/UseConstant”;

CHAPTER 10

Rule

10.1 Rules

Exakat provides unique 1371 rules to detect BUGS, CODE SMELLS, SECURITY OR QUALITY ISSUES in your PHP code.

Each rule is documented with : * a PHP version : The version of PHP to wich the rule apply * Short Name or Identifier : The Id of the rule necessary in all configuration files * Code example : The illustrative way to explain the issue detected by the rule and the targeted example of the remediated code * Time to Fix : a estimated duration to remediate the code * Severity : the impact level of the issue generated by the rule * Exakat Since : The version of Exakat Engine after which the rule is applicable

Note: The detail of Rules is available in our *REFERENCE GUIDE*.

10.2 Rulesets

A Ruleset is configurable with the -T option, when running exakat in command line. For example :

```
php exakat.phar analyze -p <project> -T <Security>
```

Note: The detail of Rulesets is available in our *REFERENCE GUIDE*.

CHAPTER 11

Report

11.1 Configuring a report before the audit

By default, Exakat builds the 'Ambassador' report for any project. If you want another report, or want to ignore the build of Ambassador, configure it before running the audit.

To do so, open the *projects/<project>/config.ini* file, and mention the list of report like that :

```
project_reports[] = 'Owasp';
project_reports[] = 'Weekly';
```

By configuring the reports before the audit, Exakat processes only the needed analysis, and produces all the reports for each audit.

11.2 Generating a report after the audit

If you have run an audit, but wants to extract another report for a piece of code, you can use the following command :

php exakat.phar report -p <project> -format <format> -file <filename>

Where <format> is one of the format listed in the following section, and <filename> is the target file.

Note that some format requires some specific audits to be run : they will fail if those results are not available. Then, run the audit again, and mention the desired audit in the configuration.

11.3 Common behavior

Default format is Text. Each report has a default filename, that may be configured with the -file option. Each report adds a file extension to the provided filename.

A special value for -file is 'stdout'. Some formats may be output to stdout, such as Text or Json. Not all format are accepting that value : some format, like Ambassador or Sqlite, may only be written to directories.

Each report is stored in its <project> folder, under the requested name.

Reports may be generated at any time, during execution of the analysis (partial results) or later, even if another audit is running.

CHAPTER 12

Cobbler

12.1 What are cobblers

Cobblers mend PHP code. They apply a transformation to it.

Cobblers are a complement to code analysis : the analysis spot code to be fixed, the cobbler mends the code. Later, the analysis doesn't find those issues anymore.

12.2 Cobbler command

To run a cobbler, use the *cobble* command.

```
php exakat cobble -p <project> <write-options> -P <Cobbler/Name>
```

The <project> parameter is the project on which the cobbler is run. It must have been *init*-ed with Exakat.

<Cobbler/Name> is the name of the cobbler to run. The list of available cobblers are in the documentation.

<write-options> configure the destination of the updated code. The available options are :

- –branch <branch> : the modified code is written in a new branch, called <branch>. The branch may be configured for each cobbler.
- –inplace : the analyzed code is replaced by the modified code. This cannot be reverted
- -f <filename> : the modified code is written in the <filename> file. Only one file is written.
- -d <dirname> : the modified codes are written in the <directory> folder. Files are written with the original name and path from the root of the repository.
- default behavior : –branch Exakat/Cobbler/Name.

12.3 Analysis and Cobblers

The analysis come first, and then the cobbler. The analysis reads the code, assess the situation and report patterns in the code that should be fixed. Then, the results from the analysis are given to the Cobbler, as a starting point. The cobbler applies various modifications in the code, and then, produce a new code. That code is now free of issues that the analysis found.

12.4 One analysis, one cobbler

For example, Performances/PrePostIncrement is the analysis that reports post-increment that should be converted into pre-increments. This is the base analysis for the Structure/PostToPre cobbler. This cobbler updates the code and turns `$a++` into `++$a`, and `$b--` into `--$b`. The resulting code is then stored into a new VCS branch, so that it may be reviewed before PR.

Cobblers are often created to apply one of the possible fixes related to one analysis. For example, Performances/PrePostIncrement might be fixed by turning the Post increment into a pre-increment, but it may also be replaced by a constant, instead of a literal.

```
<?php

$a++;

// Speed up the code with pre-increment
// ++$a;

// Make the ++ operation configurable
// const C = 1;
// $a = $a + C;

?>
```

It is not possible to apply the two cobblers at the same time, since they do not pursue the same goals. One is a performance improvement, the other one make the code configurable.

12.5 One analysis, multiple cobblers

When one analysis produces results that may be fixed with multiple cobbler, apply the following strategy : + Run the different cobblers, and write the results in different branches + Do a PR with each branch, and cherry pick the transformations

12.6 Multiple analysis, one cobbler

It is possible to apply the same cobbler to the results of multiple analysis : for example, the Structures/RemoveCode may be applied simultaneously to the analysis *Structures/UselessExpressions* and *Classes/UnusedClasses*. Both analysis spot unused code, that may well be removed.

12.7 Cobbler configuration

Cobblers take the following configuration directives :

- Source analysis : the analysis which should be resolved by the cobbler. One or more analysis may be provided. Default values are provided, and available in the documentation.
- Branch name : the branch used in the current VCS, to store the mended code.
- Specific configuration : some cobblers accept customs configuration. They are detailled in the documentation of the cobbler.

12.8 INI configuration example:

12.9 Cobbler tutorial

12.10 Pre-requisite

We assume that Exakat has been *install*-ed, and that an exakat project is already inited.

The way to run a cobbler is to call the *cobble* command. In this example, exakat removes the noscream @ operator, based on the *Structures/NoScream* analysis, and store the results in the *target-branch* for the *project name*.

```
> php exakat init -p phulp -R <URL> -git
> php exakat cobble -p <project name> -b <target_branch> -P Structures/RemoveNoScream
```

CHAPTER 13

Rules

13.1 Introduction

Exakat provides unique 1371 rules to detect BUGS, CODE SMELLS, SECURITY OR QUALITY ISSUES in your PHP code.

Each rule is documented with code example to allow you to remediate your code. If you want to automate remediation, ours cobblers can are there to fix the issues in your code for your.

13.2 List of Rules

13.2.1 Ambiguous Array Index

Indexes should not be defined with different types than int or string.

Array indices only accept integers and strings, so any other type of literal is reported. In fact, `null` is turned into an empty string, booleans are turned into an integer, and real numbers are truncated (not rounded).

```
<?php

$x = [ 1   => 1,
       '1' => 2,
       1.0 => 3,
       true => 4];
// $x only contains one element : 1 => 4

// Still wrong, immediate typecast to 1
$x[1.0]  = 5;
$x[true] = 6;

?>
```

They are indeed distinct, but may lead to confusion.

See also array.

Suggestions

- Only use string or integer as key for an array.
- Use transtyping operator (string) and (int) to make sure of the type

Specs

Short name	Arrays/AmbiguousKeys
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*PrestaShop*, *Mautic*

13.2.2 Array() / [ ] Consistence

array() or [ ] is the favorite.

array() and [ ] have the same functional use.

The analyzed code has less than 10% of one of them : for consistency reasons, it is recommended to make them all the same.

It happens that array() or [] are used depending on coding style and files. One file may be consistently using array(), while the others are all using [].

```
<?php

$a = array(1, 2);
$b = array(array(3, 4), array(5, 6));
$c = array(array(array(7, 8), array(9, 10)), array(11, 12), array(13, 14)));

// be consistent
$d = [1, 3];
?>
```

The only drawback to use [] over array() is backward incompatibility.

Suggestions

- Use one syntax consistently.

Name	Default	Type	Description
array_ratio	10	integer	Percentage of arrays in one of the syntaxes, to trigger the other syntax as a violation.

Specs

Short name	Arrays/ArrayBracketConsistence
Rulesets	none
Exakt since	0.8.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.3 Array Index

List of all indexes used in arrays.

```
<?php

// Index
$x['index'] = 1;

// in array creation
$a = array('index2' => 1);
$a2 = ['index3' => 2];

?>
```

Specs

Short name	Arrays/Arrayindex
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	Very high

13.2.4 Short Syntax For Arrays

Arrays written with the new short syntax.

PHP 5.4 introduced the new short syntax, with square brackets. The previous syntax, based on the array() keyword is still available.

```
<?php

// All PHP versions array
$a = array(1, 2, 3);

// PHP 5.4+ arrays
$a = [1, 2, 3];

?>
```

See also Array.

Specs

Short name	Arrays/ArrayNSUsage
Rulesets	*CE*, *CompatibilityPHP53*
Exakt since	0.8.4
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.5 Empty Final Element

The array() construct allows for the empty last element.

By putting an element on each line, and adding the final comma, it is possible to reduce the size of the diff when comparing code with the previous version.

```
<?php

// Array definition with final empty element
$array = [1,
          2,
          3,
          ];

// This array definition has only one line of diff with the previous array : the line␣
↪with '4,'
$array = [1,
          2,
          3,
          4,
          ];

// This array definition is totally different from the first array :
$array = [1, 2, 3, 4];

?>
```

See also Array, Zend Framework Coding Standard and How clean is your code? How clean are your diffs?.

Specs

Short name	Arrays/EmptyFinal
Rulesets	none
Exakt since	0.11.0
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.6 Empty Slots In Arrays

PHP tolerates the last element of an array to be empty.

```
<?php
    $a = array( 1, 2, 3, );
    $b =       [ 4, 5, ];
?>
```

Specs

Short name	Arrays/EmptySlots
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.7 Getting Last Element

Getting the last element of an array relies on array_key_last().

array_key_last() was added in PHP 7.3. Before that,

```
<?php

$array = ['a' => 1, 'b' => 2, 'c' => 3];

// Best solutions, by far
$last = $array[array_key_last($array)];

// Best solutions, just as fast as each other
$last = $array[count($array) - 1];
$last = end($array);

// Bad solutions

// popping, but restoring the value.
$last = array_pop($array);
$array[] = $last;

// array_unshift would be even worse

// reversing array
$last = array_reverse($array)[0];

// slicing the array
$last = array_slice($array, -1)[0]',
$last = current(array_slice($array, -1));
);

?>
```

Suggestions

- Use PHP native function : array_key_last(), when using PHP 7.4 and later
- Use PHP native function : array_pop()
- Organise the code to put the last element in the first position (array_unshift() instead of append operator [])

Specs

Short name	Arrays/GettingLastElement
Rulesets	*Performances*
Exakt since	0.9.0
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Thelia*

13.2.8 Mass Creation Of Arrays

Literal creation of an array, by assigning a lot of index.

```
<?php

$row['name'] = $name;
$row['last'] = $last;
$row['address'] = $address;

?>
```

Specs

Short name	Arrays/MassCreation
Rulesets	none
Exakt since	1.1.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.9 Mistaken Concatenation

A unexpected structure is built for initialization. It may be a typo that creates an unwanted expression.

```
<?php

// This 'cd' is unexpected. Isn't it 'c', 'd' ?
$array = array('a', 'b', 'c'. 'd');
```

(continues on next page)

(continued from previous page)

```
$array = array('a', 'b', 'c', 'd');

// This 4.5 is unexpected. Isn't it 4, 5 ?
$array = array(1, 2, 3, 4.5);
$array = array(1, 2, 3, 4, 5);

?>
```

Specs

Short name	Arrays/MistakenConcatenation
Rulesets	none
Exakt since	1.0.8
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.10 Mixed Keys Arrays

Avoid mixing constants and literals in array keys.

When defining default values in arrays, it is recommended to avoid mixing constants and literals, as PHP may mistake them and overwrite the previous with the latter.

Either switch to a newer version of PHP (5.5 or newer), or make sure the resulting array hold the expected data. If not, reorder the definitions.

```
<?php

const ONE = 1;

$a = [ 1   => 2,
       ONE => 3];

?>
```

Suggestions

- Use only literals or constants when building the array

Specs

Short name	Arrays/MixedKeys
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*
Exakt since	0.8.4
Php Version	5.6+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.11 Multidimensional Arrays

Simply, arrays of arrays.

```
<?php
    $x[1][2] = $x[2][3][4];

?>
```

See also Type array and Using Multidimensional Arrays in PHP.

Specs

Short name	Arrays/Multidimensional
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.12 Multiple Index Definition

Indexes that are defined multiple times in the same array.

```
<?php
    // Multiple identical keys
    $x = array(1 => 2,
               2 => 3,
               1 => 3);

    // Multiple identical keys (sneaky version)
    $x = array(1 => 2,
               1.1 => 3,
               true => 4);

    // Multiple identical keys (automated version)
    $x = array(1 => 2,
               3,         // This will be index 2
               2 => 4);  // this index is overwritten
?>
```

They are indeed overwriting each other. This is most probably a typo.

Suggestions

- Review your code and check that arrays only have keys defined once.
- Review carefully your code and check indirect values, like constants, static constants.

Specs

Short name	Arrays/MultipleIdenticalKeys
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Magento*, *MediaWiki*

13.2.13 Negative Start Index In Array

> Negative starting index in arrays changed in PHP 8.0. Until then, they were ignored, and automatic index started always at 0. Since PHP 8.0, the next index is calculated.

The behavior will break code that relies on automatic index in arrays, when a negative index is used for a starter.

```
<?php

$x = [-5 => 2];
$x[] = 3;

print_r($x);

/*
PHP 7.4 and older
Array
(
    [-5] => 2
    [0] => 3
)
*/

/*
PHP 8.0 and more recent
Array
(
    [-5] => 2
    [-4] => 3
)
*/

?>
```

See also PHP RFC: Arrays starting with a negative index.

Suggestions

- Explicitely create the index, instead of using the automatic indexing
- Add an explicit index of 0 in the initial array, to set the automatic process in the right track
- Avoid using specified index in array, conjointly with automatic indexing.

Specs

Short name	Arrays/NegativeStart
Rulesets	*CE*, *CompatibilityPHP80*
Exakt since	2.1.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.14 Non-constant Index In Array

Undefined constants revert as strings in Arrays. They are also called `barewords`.

In `$array[index]`, PHP cannot find index as a constant, but, as a default behavior, turns it into the string `index`.

This default behavior raise concerns when a corresponding constant is defined, either using define() or the const keyword (outside a class). The definition of the index constant will modify the behavior of the index, as it will now use the constant definition, and not the 'index' string.

```
<?php

// assign 1 to the element index in $array
// index will fallback to string
$array[index] = 1;
//PHP Notice:  Use of undefined constant index - assumed 'index'

echo $array[index];       // display 1 and the above error
echo "$array[index]";     // display 1
echo "$array['index']";   // Syntax error


define('index', 2);

 // now 1 to the element 2 in $array
 $array[index] = 1;

?>
```

It is recommended to make index a real string (with ' or "), or to define the corresponding constant to avoid any future surprise.

Note that PHP 7.2 removes the support for this feature.

See also PHP RFC: Deprecate and Remove Bareword (Unquoted) Strings and Syntax.

Suggestions

- Declare the constant to give it an actual value
- Turn the constant name into a string

Specs

Short name	Arrays/NonConstantArray
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Dolibarr*, *Zencart*

13.2.15 No Spread For Hash

The spread operator `...` only works on integer-indexed arrays.

```
<?php

// This is valid, as ``-33`` is cast to integer by PHP automagically
var_dump(...[1,-33 => 2, 3]);

// This is not valid
var_dump(...[1,C => 2, 3]);

?>
```

See also Variable-length argument lists.

Suggestions

- Add a call to array_values() instead of the hash

Specs

Short name	Arrays/NoSpreadForHash
Rulesets	*Analyze*
Exakt since	1.9.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.16 Null Or Boolean Arrays

Null and booleans are valid PHP array base. Yet, they only produces `null` values. They also did not emits any warning until PHP 7.4.

This analysis has been upgraded to cover int and float types too.

```
<?php

// outputs NULL
var_dump(null[0]);

const MY_CONSTANT = true;
// outputs NULL
var_dump(MY_CONSTANT[10]);

?>
```

See also Null and True.

Suggestions

- Avoid using the array syntax on null and boolean
- Avoid using null and boolean on constant that are expecting arrays

Specs

Short name	Arrays/NullBoolean
Rulesets	*Analyze*
Exakt since	1.8.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.17 PHP Arrays Index

List of indexes used when manipulating PHP arrays in the code.

```
<?php

// HTTP_HOST is a PHP array index.
$ip = 'http'.$_SERVER['HTTP_HOST'].'/'.$row['path'];

//'path' is not a PHP index

?>
```

Specs

Short name	Arrays/Phparrayindex
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.18 Randomly Sorted Arrays

Those literal arrays are written in several places, but their items are in various orders.

This may reduce the reading and proofing of the arrays, and induce confusion. The random order may also be a residue of development : both arrays started with different values, but they grew overtime to handle the same items. The way they were written lead to the current order.

Unless order is important, it is recommended to always use the same order when defining literal arrays. This makes it easier to match different part of the code by recognizing one of its literal.

```
<?php

// an array
$set = [1,3,5,9,10];

function foo() {
    // an array, with the same values but different order, in a different context
    $list = [1,3,5,10,9,];
}

// an array, with the same order than the initial one
$inits = [1,3,5,9,10];

?>
```

Suggestions

- Match the sorting order of the arrays. Choose any of them.
- Configure a constant and use it as a replacement for those arrays.
- Leave the arrays intact : the order may be important.
- For hash arrays, consider turning the array in a class.

Specs

Short name	Arrays/RandomlySortedLiterals
Rulesets	*Analyze*, *Suggestions*
Exakt since	0.11.2
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Contao*, *Vanilla*

13.2.19 Preprocess Arrays

Using long list of assignations for initializing arrays is significantly slower than the declaring them as an array.

```
<?php

// Slow way
$a = []; // also with $a = array();
$a[1] = 2;
$a[2] = 3;
$a[3] = 5;
$a[4] = 7;
$a[5] = 11;

// Faster way
$a = [1 => 2,
      2 => 3,
      3 => 5,
      4 => 7,
      5 => 11];

// Even faster way if indexing is implicit
$a = [2, 3, 5, 7, 11];

?>
```

If the array has to be completed rather than created, it is also faster to use += when there are more than ten elements to add.

```
<?php

// Slow way
$a = []; // also with $a = array();
$a[1] = 2;
$a[2] = 3;
$a[3] = 5;
// some expressions to get $seven and $eleven
$a[4] = $seven;
$a[5] = $eleven;

// Faster way
$a = [1 => 2,
```

(continues on next page)

(continued from previous page)

```
      2 => 3,
      3 => 5];
// some expressions to get $seven and $eleven
$a += [4 => $seven,
       5 => $eleven];

// Even faster way if indexing is implicit
$a = [2, 3, 5];
// some expressions to get $seven and $eleven
$a += [$seven, $eleven];

?>
```

Suggestions

- Preprocess the code so PHP doesn't do it. Keep the detailed version into comments.

Specs

Short name	Arrays/ShouldPreprocess
Rulesets	*Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.20 Slice Arrays First

Always start by reducing an array before applying some transformation on it. The shorter array will be processed faster.

```
<?php

// fast version
$a = array_map('foo', array_slice($array, 2, 5));

// slower version
$a = array_slice(array_map('foo', $array), 2, 5);
?>
```

The gain produced here is greater with longer arrays, or greater reductions. They may also be used in loops. This is a micro-optimisation when used on short arrays.

Suggestions

- Use the array transforming function on the result of the array shortening function.

Specs

Short name	Arrays/SliceFirst
Rulesets	*Performances*, *Suggestions*
Exakt since	1.0.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*WordPress*

13.2.21 String Initialization

It used to be possible to initialize a variable with an string, and use it as an array. It is not the case anymore in PHP 7.1.

```
<?php

// Initialize arrays with array()
$a = array();
$a[3] = 4;

// Don't start with a string
$a = '';
$a[3] = 4;
print $a;

// Don't start with a string
if (is_numeric($a)) {
    $a[] = $a;
}

?>
```

See also PHP 7.1 no longer converts string to arrays the first time a value is assigned with square bracket notation.

Suggestions

- Always initialize arrays with an empty array(), not a string.

Specs

Short name	Arrays/StringInitialization
Rulesets	*CompatibilityPHP71*
Exakt since	1.6.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.22 Too Many Array Dimensions

When arrays a getting to many nesting.

```
<?php

$a           = array();   // level 1;
$a[1]        = array();   // level 2
$a[1][2]     = array();   // level 3 : still valid by default
$a[1][2][3] = array();   // level 4

?>
```

PHP has no limit, and accepts any number of nesting levels. Yet, this is usually very memory hungry.

Suggestions

-

Name	Default	Type	Description
maxDimensions	3	integer	Number of valid dimensions in an array.

Specs

Short name	Arrays/TooManyDimensions
Rulesets	*Analyze*
Exakt since	1.9.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.23 Weird Array Index

Array index that looks weird. Arrays index may be string or integer, but some strings looks weird.

In particular, strings that include terminal white spaces, often leads to missed values.

```
<?php

$array = ['a ' => 1, 'b' => 2, 'c' => 3];

// Later in the code

//Notice: Undefined index: a in /Users/famille/Desktop/analyzeG3/test.php on line 8
echo $array['a'];

//Notice: Undefined index: b  in /Users/famille/Desktop/analyzeG3/test.php on line 10
// Note that the space is visible, but easy to miss
echo $array['b '];

// all fine here
```

(continues on next page)

(continued from previous page)

```
echo $array['c'];

?>
```

Although this is rare error, and often easy to spot, it is also very hard to find when it strikes.

Suggestions

- Remove white spaces when using strings as array index.

Specs

Short name	Arrays/WeirdIndex
Rulesets	*Semantics*
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.24 Handle Arrays With Callback

Use functions like array_map().

```
<?php

// Handles arrays with callback
$uppercase = array_map('strtoupper', $source);

// Handles arrays with foreach
foreach($source as &$s) {
    $s = uppercase($s);
}

?>
```

See also array_map.

Specs

Short name	Arrays/WithCallback
Rulesets	*CE*
Exakt since	1.3.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.25 Modify Immutable

A class, marked as immutable, is being modified.

This attribute is supported as a PHPdoc comment, '@immutable, and as a PHP 8.0 attribute.

```
<?php

/** @Immutable */
#[Immutable]
class x {
    public $x = 1, $y, $z;
}

$x = new X;
// $x->x is modified, while it should not
$x->x = 2 + $x->z;

// $x->z is read only, as expected

?>
```

See also phpstorm-stubs/meta/attributes/Immutable.php and PhpStorm 2020.3 EAP #4: Custom PHP 8 Attributes.

Suggestions

- Removed the modification
- Clone the immutable object

Specs

Short name	Attributes/ModifyImmutable
Rulesets	*Analyze*, *Attributes*
Exakt since	2.2.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.26 Abstract Class Usage

List of all abstract classes being used.

```
<?php

abstract class foo {
    function foobar();
}

class bar extends foo {
    // extended method
    function foobar() {
```

(continues on next page)

(continued from previous page)

```
        // doSomething()
    }

    // extra method
    function barbar() {
        // doSomething()
    }
}
?>
```

See also Classes abstraction.

Specs

Short name	Classes/Abstractclass
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.27 Abstract Methods Usage

List of all abstract methods being used.

```
<?php

// abstract class
abstract class foo {
    // abstract method
    function foobar();
}

class bar extends foo {
    // extended abstract method
    function foobar() {
        // doSomething()
    }

    // extra method
    function barbar() {
        // doSomething()
    }
}
?>
```

See also Classes abstraction.

Specs

Short name	Classes/Abstractmethods
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.28 Abstract Or Implements

A class must implements all abstract methods of it parent, or be abstract too.

While PHP lints this code, it won't execute it and stop with a Fatal Error : `Class BA contains 1 abstract method and must therefore be declared abstract or implement the remaining methods (A\:\:aFoo).`

```
<?php

abstract class Foo {
    abstract function FooBar();
}

// This is in another file : php -l would detect it right away

class FooFoo extends Foo {
    // The method is not defined.
    // The class must be abstract, just like Foo
}

?>
```

See also Class Abstraction.

Suggestions

- Implements all the abstract methods of the class
- Make the class abstract

Specs

Short name	Classes/AbstractOrImplements
Rulesets	*Analyze*, *LintButWontExec*
Exakt since	1.3.3
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Zurmo*

13.2.29 Abstract Static Methods

Methods cannot be both abstract and static. Static methods belong to a class, and will not be overridden by the child class. For normal methods, PHP will start at the object level, then go up the hierarchy to find the method. With static, it is necessary to mention the name, or use Late Static Binding, with self or static. Hence, it is useless to have an abstract static method : it should be a static method.

A child class is able to declare a method with the same name than a static method in the parent, but those two methods will stay independent.

This is not the case anymore in PHP 7.0+.

```
<?php

abstract class foo {
    // This is not possible
    static abstract function bar() ;
}

?>
```

See also Why does PHP 5.2+ disallow abstract 'static class methods? <https://stackoverflow.com/questions/999066/why-does-php-5-2-disallow-abstract-static-class-methods>'_.

Suggestions

- Remove abstract keyword from the method
- Remove static keyword from the method
- Remove the method

Specs

Short name	Classes/AbstractStatic
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	7.0-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.30 Accessing Private

List of calls to private properties/methods that will compile but yield some fatal error upon execution.

```
<?php

class a {
    private $a;
}

class b extends a {
```

(continues on next page)

(continued from previous page)

```
    function c() {
        $this->a;
    }
}

?>
```

Specs

Short name	Classes/AccessPrivate
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.31 Access Protected Structures

It is not allowed to access protected properties or methods from outside the class or its relatives.

```
<?php

class foo {
    protected $bar = 1;
}

$foo = new Foo();
$foo->bar = 2;

?>
```

See also Visibility and Understanding The Concept Of Visibility In Object Oriented PHP.

Suggestions

- Change 'protected' to 'public' to relax the constraint
- Add a getter method to reach the target value
- Remove the access to the protected value and find it another way

Specs

Short name	Classes/AccessProtected
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.32 Ambiguous Static

Methods or properties with the same name, are defined static in one class, and not static in another. This is error prone, as it requires a good knowledge of the code to make it static or not.

Try to keep the methods simple and unique. Consider renaming the methods and properties to distinguish them easily. A method and a static method have probably different responsibilities.

```
<?php

class a {
    function mixedStaticMethod() {}
}

class b {
    static function mixedStaticMethod() {}
}

/... a lot more code later .../

$c->mixedStaticMethod();
// or
$c::mixedStaticMethod();

?>
```

Specs

Short name	Classes/AmbiguousStatic
Rulesets	*Analyze*
Exakt since	1.0.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.33 Ambiguous Visibilities

The properties have the same name, but have different visibilities, across different classes.

While it is legit to have a property with the same name in different classes, it may easily lead to confusion. As soon as the context is need to understand if the property is accessible or not, the readability suffers.

It is recommended to handle the same properties in the same way across classes, even when the classes are not related.

```
<?php

class person {
    public $name;
    private $address;
}

class gangster {
    private $name;
    public $nickname;
    private $address;
}

$someone = Human::load(123);
echo 'Hello, '.$someone->name;

?>
```

Suggestions

- Sync visibilities for both properties, in the different classes
- Use different names for properties with different usages

Specs

Short name	Classes/AmbiguousVisibilities
Rulesets	*Analyze*
Exakt since	1.3.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Typo3*

13.2.34 Anonymous Classes

Anonymous classes.

```
<?php

// Anonymous class, available since PHP 7.0
$object = new class { function __construct() { echo __METHOD__; } };

?>
```

Specs

Short name	Classes/Anonymous
Rulesets	*CE*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	7.0+
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.35 Avoid Optional Properties

Avoid optional properties, to prevent littering the code with existence checks.

When a property has to be checked once for existence, it is safer to check it each time. This leads to a decrease in readability and a lot of checks added to the code.

Either make sure the property is set with an actual object rather than with null, or use a null object. A null object offers the same interface than the expected object, but does nothing. It allows calling its methods, without running into a Fatal error, nor testing it.

```
<?php

// Example is courtesy 'The Coding Machine' : it has been adapted from its original␣
↪form. See link below.

class MyMailer {
    private $logger;

    public function __construct(LoggerInterface $logger = null) {
        $this->logger = $logger;
    }

    private function sendMail(Mail $mail) {
        // Since $this->logger may be null, it must be tested anytime it is used.
        if ($this->logger) {
            $this->logger->info('Mail successfully sent.');
        }
    }
}

?>
```

See also Avoid optional services as much as possible, The Null Object Pattern – Polymorphism in Domain Models, and Practical PHP Refactoring: Introduce Null Object.

Suggestions

- Use a null object to fill any missing value
- Make sure the property is set at constructor time

Specs

Short name	Classes/AvoidOptionalProperties
Rulesets	*Analyze*
Exakt since	0.12.0
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
Examples	*ChurchCRM*, *Dolibarr*

13.2.36 Avoid option arrays in constructors

Avoid option arrays in constructors. Use one parameter per injected element.

```
<?php

class Foo {
    // Distinct arguments, all typehinted if possible
    function __constructor(A $a, B $b, C $c, D $d) {
        $this->a = $a;
        $this->b = $b;
        $this->c = $c;
        $this->d = $d;
    }
}

class Bar {
    // One argument, spread over several properties
    function __constructor(array $options) {
        $this->a = $options['a'];
        $this->b = $options['b'];
        $this->c = $options['c'];
        $this->d = $options['d'];
    }
}

?>
```

See also Avoid option arrays in constructors and PHP RFC: Named Arguments (Type-safe and documented options).

Suggestions

- Spread the options in the argument list, one argument each
- Use a configuration class, that hold all the elements with clear names, instead of an array
- Use named parameters to pass and document the arguments

Specs

Short name	Classes/AvoidOptionArrays
Rulesets	*Analyze*, *ClassReview*
Exakt since	1.7.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.37 Custom Class Usage

List of usage of custom classes throughout the code.

Name	Default	Type	Description
forbiddenClasses		ini_hash	List of classes to be avoided

Specs

Short name	Classes/AvoidUsing
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.38 Cancel Common Method

A parent method's is too little used in children.

The parent class has a method, which is customised in children classes, though most of the time, those are empty : hence, cancelled.

```
<?php

class x {
    abstract function foo();
    abstract function bar();
}

class y1 extends x {
    function foo() { doSomething(); }
    function bar() { doSomething(); };
}

class y2 extends x {
    // foo is cancelled : it must be written, but has no use.
    function foo() {  }
    function bar() { doSomething(); };
```

(continues on next page)

(continued from previous page)

```
}

?>
```

A threshold of `cancelThreshold` % of the children methods have to be cancelled to report the parent class. By default, it is 75 (or 3 out of 4).

Suggestions

- Drop the common method, and the cancelled methods in the children
- Fill the children's methods with actual code

Name	Default	Type	Description
cancelThreshold	75	integer	Minimal number of cancelled methods to suggest the cancellation of the parent.

Specs

Short name	Classes/CancelCommonMethod
Rulesets	*ClassReview*, *Suggestions*
Exakt since	2.1.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.39 Can't Extend Final

It is not possible to extend final classes.

Since PHP fails with a fatal error, this means that the extending class is probably not used in the rest of the code. Check for dead code.

```
<?php
    // File Foo
    final class foo {
        public final function bar() {
            // doSomething
        }
    }
?>
```

In a separate file :

```
<?php
    // File Bar
    class bar extends foo {
```

(continues on next page)

(continued from previous page)

```
    }
?>
```

See also Final Keyword.

Suggestions

- Remove the final keyword
- Remove the extending class

Specs

Short name	Classes/CantExtendFinal
Rulesets	*Analyze*, *Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Critical
Time To Fix	Instant (5 mins)
Precision	Medium

13.2.40 Cant Inherit Abstract Method

Inheriting abstract methods was made available in PHP 7.2. In previous versions, it emitted a fatal error.

```
<?php

abstract class A           { abstract function bar(stdClass $x);   }
abstract class B extends A { abstract function bar($x): stdClass; }

//   Fatal error: Can't inherit abstract function A::bar()
?>
```

See also PHP RFC: Allow abstract function override.

Suggestions

- Avoid inheriting abstract methods for compatibility beyond 7.2 (and older)

Specs

Short name	Classes/CantInheritAbstractMethod
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*
Exakt since	0.11.8
Php Version	7.2+
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.41 Cant Instantiate Class

When constructor is not public, it is not possible to instantiate such a class. Either this is a conception choice, or there are factories to handle that. Either way, it is not possible to call new on such class.

PHP reports an error similar to this one : 'Call to private Y::__construct() from invalid context'.

```
<?php

//This is the way to go
$x = X::factory();

//This is not possible
$x = new X();

class X {
    //This is also the case with proctected __construct
    private function __construct() {}

    static public function factory() {
        return new X();
    }
}

?>
```

See also In a PHP5 class, when does a private constructor get called?, Named Constructors in PHP and PHP Constructor Best Practices And The Prototype Pattern.

Specs

Short name	Classes/CantInstantiateClass
Rulesets	*Analyze*
Exakt since	1.2.8
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High
Examples	*WordPress*

13.2.42 Check On __Call Usage

When using the magic methods __call() and __staticcall(), make sure the method exists before calling it.

If the method doesn't exists, then the same method will be called again, leading to the same failure. Finally, it will crash PHP.

```
<?php

class safeCall {
    function __class($name, $args) {
        // unsafe call, no checks
        if (method_exists($this, $name)) {
            $this->$name(...$args);
        }
    }
}

class unsafeCall {
    function __class($name, $args) {
        // unsafe call, no checks
        $this->$name(...$args);
    }
}

?>
```

See also Method overloading and ‘‘Magical PHP: __call <https://www.garfieldtech.com/index.php/blog/magical-php-call>‘_.

Suggestions

- Add a call to method_exists() before using any method name
- Relay the call to another object that doesn't handle __call() or __callStatic()

Specs

Short name	Classes/CheckOnCallUsage
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.7.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.43 Child Class Removes Typehint

PHP 7.2 introduced the ability to remove a typehint when overloading a method. This is not valid code for older versions.

```
<?php

class foo {
    function foobar(foo $a) {}
}

class bar extends foo {
    function foobar($a) {}
}

?>
```

Specs

Short name	Classes/ChildRemoveTypehint
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *Typechecks*
Exakt since	0.12.4
Php Version	7.2+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.44 Class, Interface Or Trait With Identical Names

The following names are used at the same time for classes, interfaces or traits. For example,

```
<?php
    class a     { /* some definitions */ }
```

(continues on next page)

(continued from previous page)

```
    interface a { /* some definitions */ }
    trait a     { /* some definitions */ }
?>
```

Even if they are in different namespaces, identical names makes classes easy to confuse. This is often solved by using alias at import time : this leads to more confusion, as a class suddenly changes its name.

Internally, PHP use the same list for all classes, interfaces and traits. As such, it is not allowed to have both a trait and a class with the same name.

In PHP 4, and PHP 5 before namespaces, it was not possible to have classes with the same name. They were simply included after a check.

Suggestions

- Use distinct names for every class, trait and interface.
- Keep eponymous classes, traits and interfaces in distinct files, for definition but also for usage. When this happens, rename one of them.

Specs

Short name	Classes/CitSameName
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*shopware*, *NextCloud*

13.2.45 Usage Of class_alias()

`class_alias` creates dynamically an alias for classes.

```
<?php

class foo { }

class_alias('foo', 'bar');

$a = new foo;
$b = new bar;

// the objects are the same
var_dump($a == $b, $a === $b);
var_dump($a instanceof $b);

// the classes are the same
var_dump($a instanceof foo);
var_dump($a instanceof bar);
```

(continues on next page)

(continued from previous page)

```
var_dump($b instanceof foo);
var_dump($b instanceof bar);

?>
```

See also class_alias.

Specs

Short name	Classes/ClassAliasUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.46 Classes Names

List of all classes, as defined in the application.

```
<?php

// foo is in the list
class foo {}

// Anonymous classes are not in the list
$o = class { function foo(){} }

?>
```

Specs

Short name	Classes/Classnames
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.47 Class Usage

List of classes being used.

```
<?php
```

(continues on next page)

(continued from previous page)

```
// Class may be used in a use expression
use MyClass as MyAliasedClass;

// class may be aliased with class_alias
class_alias('MyOtherAliasedClass', 'MyClass');

// Class may be instanciated
$o = new MyClass();

// Class may be used with instanceof
var_dump($o instanceof \MyClass);

// Class may be used in static calls
MyClass::aConstant;
echo MyClass::$aProperty;
echo MyClass::aMethod( $o );

// Class may be extended
class MyOtherClass {

}

class MyClass extends MyOtherClass {
    const aConstant = 1;

    public static $aProperty = 2;

    // also used as a typehint
    public static function aMethod(MyClass $object) {
        return __METHOD__;
    }
}

?>
```

Specs

Short name	Classes/ClassUsage
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.48 Clone With Non-Object

The `clone` keyword must be used on variables, properties or results from a function or method call. `clone` cannot be used with constants or literals.

```php
<?php

class x { }
$x = new x();

// Valid clone
$y = clone $x;

// Invalid clone
$y = clone x;

?>
```

Cloning a non-object lint but won't execute.

See also Object cloning.

Suggestions

- Only clone containers (like variables, properties...)
- Add typehint to injected properties, so they are checked as objects.

Specs

Short name	Classes/CloneWithNonObject
Rulesets	*Analyze*, *LintButWontExec*
Exakt since	1.7.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.49 Clone Usage

List of all clone situations.

```php
<?php
    $dateTime = new DateTime();
    echo (clone $dateTime)->format('Y');
?>
```

See also Object cloning.

Specs

Short name	Classes/CloningUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.50 Constant Class

A class or an interface only made up of constants. Constants usually have to be used in conjunction of some behavior (methods, class...) and never alone.

```
<?php

class ConstantClass {
    const KBIT = 1000;
    const MBIT = self::KBIT * 1000;
    const GBIT = self::MBIT * 1000;
    const PBIT = self::GBIT * 1000;
}

?>
```

As such, they should be PHP constants (build with define or const), or included in a class with other methods and properties.

See also PHP Classes containing only constants.

Suggestions

- Make the class an interface
- Make the class an abstract class, to avoid its instantiation

Specs

Short name	Classes/ConstantClass
Rulesets	*ClassReview*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.51 Constant Definition

List of class constants being defined.

```php
<?php

// traditional way of making constants
define('aConstant', 1);

// modern way of making constants
const anotherConstant = 2;

class foo {
    // Not a constant, a class constant.
    const aClassConstant = 3;
}

?>
```

See also PHP Constants.

Specs

Short name	Classes/ConstantDefinition
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.52 Constant Used Below

Mark class constants that are used in children classes.

```php
<?php

class foo {
    // This constant is used in children
    protected PROTECTEDPROPERTY = 1;

    // This constant is not used in children
    protected LOCALPROTECTEDPROPERTY = 1;

    private function foobar() {
        // PROTECTEDPROPERTY is used here, but defined in parent
        echo self::LOCALPROTECTEDPROPERTY;
    }
}

class foofoo extends foo {
    private function bar() {
        // protectedProperty is used here, but defined in parent
        print self::PROTECTEDPROPERTY;
    }
}
```

(continues on next page)

(continued from previous page)

```
?>
```

This analysis marks constants at their definition, not the current class, nor the (grand-)'parent <https://www.php.net/manual/en/language.oop5.paamayim-nekudotayim.php>'_.

Specs

Short name	Classes/ConstantUsedBelow
Rulesets	none
Exakt since	0.12.10
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.53 Constructors

Mark methods as constructors.

```
<?php

class x {
    // Normal constructor
    function __construct() {}
}

class y {
    // Old style constructor, obsolete since PHP 7.1
    function y() {}
}

class z {
    // Normal constructor
    function __construct() {}

    // Old style constructor, but with lower priority
    function z() {}
}

?>
```

See also Constructors and Destructors.

Specs

Short name	Classes/Constructor
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.54 Const Visibility Usage

Visibility for class constant controls the accessibility to class constant.

A public constant may be used anywhere in the code; a protected constant usage is restricted to the class and its relatives; a private constant is restricted to itself.

This feature was introduced in PHP 7.1. It is recommended to use explicit visibility, and, whenever possible, make the visibility private.

```
<?php

class x {
    public const a = 1;
    protected const b = 2;
    private const c = 3;
    const d = 4;
}

interface i {
    public const a = 1;
      const d = 4;
}

?>
```

See also Class Constants and PHP RFC: Support Class Constant Visibility.

Suggestions

- Add constant visibility, at least 'public'.

Specs

Short name	Classes/ConstVisibilityUsage
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*
Exakt since	1.3.0
Php Version	7.1+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.55 Could Be Abstract Class

An abstract class is never instantiated, and has children class that are. As such, a 'parent' class that is never instantiated by itself, but has its own children instantiated could be marked as abstract.

That will prevent new code to try to instantiate it.

```
<?php

// Example code would actually be split over multiple files.


// That class could be abstract
class motherClass {}

// Those classes shouldn't be abstract
class firstChildren extends motherClass {}
class secondChildren extends motherClass {}
class thirdChildren extends motherClass {}

new firstChildren();
new secondChildren();
new thirdChildren();

//Not a single : new motherClass()

?>
```

See also Class Abstraction Abstract classes and methods.

Suggestions

- Make this class an abstract class

Specs

Short name	Classes/CouldBeAbstractClass
Rulesets	*Analyze*, *ClassReview*
Exakt since	1.3.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Edusoho*, *shopware*

13.2.56 Could Be Class Constant

When a property is defined and read, but never modified, it may be a constant.

```
<?php

class foo {
    // $this->bar is never modified.
    private $bar = 1;

    // $this->foofoo is modified, at least once
    private $foofoo = 2;

    function method($a) {
        $this->foofoo = $this->bar + $a + $this->foofoo;

        return $this->foofoo;
    }

}

?>
```

Starting with PHP 5.6, even array() may be defined as constants.

Specs

Short name	Classes/CouldBeClassConstant
Rulesets	*ClassReview*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.57 Class Could Be Final

Any class that has no extension should be `final` by default.

As stated by `Matthias Noback : If a class is not marked final, it has at least one subclass.`

Prevent your classes from being subclassed by making them `final`. Sometimes, classes are not meant or thought to be derivable.

```
<?php

class x {}             // This class is extended
class y extends x {}   // This class is extended
class z extends y {}   // This class is not extended

final class z2 extends y {}  // This class is not extended

?>
```

See also Negative architecture, and assumptions about code.

Suggestions

- Make the class final
- Extends the class

Specs

Short name	Classes/CouldBeFinal
Rulesets	*Analyze*, *ClassReview*
Exakt since	1.4.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.58 Could Be Parent Method

A method is defined in several children, but not in a the parent class. It may be worth checking if this method doesn't belong the parent class, as an abstraction.

```
<?php

// The parent class
class x { }

// The children class
class y1 extends x {
    // foo is common to y1 and y2, so it shall be also a method in x
    function foo() {}
    // fooY1 is specific to y1
    function fooY1() {}
}

class y2 extends x {
```

(continues on next page)

(continued from previous page)

```
    function foo() {}
    // fooY2 is specific to y1
    function fooY2() {}
}

?>
```

Only the name of the method is used is for gathering purposes. If the code has grown organically, the signature (default values, typehint, argument names) may have followed different path, and will require a refactorisation.

Suggestions

- Create an abstract method in the parent
- Create an concrete method in the parent, and move default behavior there by removing it in children classes

Name	Default	Type	Description
minChildren	4	integer	Minimal number of children using this method.

Specs

Short name	Classes/CouldBeParentMethod
Rulesets	*ClassReview*
Exakt since	2.1.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.59 Property Could Be Private Property

The following properties are never used outside their class of definition Given the analyzed code, they could be set as private.

```
<?php

class foo {
    public $couldBePrivate = 1;
    public $cantdBePrivate = 1;

    function bar() {
        // couldBePrivate is used internally.
        $this->couldBePrivate = 3;
    }
}

class foo2 extends foo {
    function bar2() {
        // cantdBePrivate is used in a child class.
        $this->cantdBePrivate = 3;
```

(continues on next page)

(continued from previous page)

```
    }
}

//$couldBePrivate is not used outside
$foo = new foo();

//$cantdBePrivate is used outside the class
$foo->cantdBePrivate = 2;

?>
```

Note that dynamic properties (such as $x->$y) are not taken into account.

Suggestions

- Remove the unused property
- Use the private property
- Change the visibility to allow access the property from other part of the code

Specs

Short name	Classes/CouldBePrivate
Rulesets	*ClassReview*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.60 Could Be Private Class Constant

Class constant may use `private` visibility.

Since PHP 7.1, constants may also have a public/protected/private visibility. This restrict their usage to anywhere, class and children or class.

As a general rule, it is recommended to make constant `private` by default, and to relax this restriction as needed. PHP makes them public by default.

```
<?php

class foo {
    // pre-7.1 style
    const PRE_71_CONSTANT = 1;

    // post-7.1 style
    private const PRIVATE_CONSTANT = 2;
    public const PUBLIC_CONSTANT = 3;

    function bar() {
```

(continues on next page)

(continued from previous page)

```
        // PRIVATE CONSTANT may only be used in its class
        echo self::PRIVATE_CONSTANT;
    }
}

// Other constants may be used anywhere
function x($a = foo::PUBLIC_CONSTANT) {
    echo $a.' '.foo:PRE_71_CONSTANT;
}

?>
```

Constant shall stay `public` when the code has to be compatible with PHP 7.0 and older.

They also have to be public in the case of component : some of those constants have to be used by external actors, in order to configure the component.

See also Class Constants.

Specs

Short name	Classes/CouldBePrivateConstante
Rulesets	*ClassReview*
Exakt since	0.12.10
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Phinx*

13.2.61 Method Could Be Private Method

The following methods are never used outside their class of definition. Given the analyzed code, they could be set as private.

```
<?php

class foo {
    public function couldBePrivate() {}
    public function cantdBePrivate() {}

    function bar() {
        // couldBePrivate is used internally.
        $this->couldBePrivate();
    }
}

class foo2 extends foo {
    function bar2() {
        // cantdBePrivate is used in a child class.
        $this->cantdBePrivate();
    }
}
```

(continues on next page)

(continued from previous page)

```
//couldBePrivate() is not used outside
$foo = new foo();

//cantdBePrivate is used outside the class
$foo->cantdBePrivate();

?>
```

Note that dynamic properties (such as $x->$y) are not taken into account.

Specs

Short name	Classes/CouldBePrivateMethod
Rulesets	*ClassReview*
Exakt since	0.12.11
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.62 Could Be Protected Class Constant

Class constant may use 'protected' visibility.

Since PHP 7.1, constants may also have a public/protected/private visibility. This restrict their usage to anywhere, class and children or class.

As a general rule, it is recommended to make constant 'private' by default, and to relax this restriction as needed. PHP makes them public by default.

```
<?php

class foo {
    // pre-7.1 style
    const PRE_71_CONSTANT = 1;

    // post-7.1 style
    protected const PROTECTED_CONSTANT = 2;
    public const PUBLIC_CONSTANT = 3;
}

class foo2 extends foo {
    function bar() {
        // PROTECTED_CONSTANT may only be used in its class or its children
        echo self::PROTECTED_CONSTANT;
    }
}

class foo3 extends foo {
    function bar() {
        // PROTECTED_CONSTANT may only be used in its class or any of its children
        echo self::PROTECTED_CONSTANT;
```

(continues on next page)

(continued from previous page)

```
    }
}

// Other constants may be used anywhere
function x($a = foo::PUBLIC_CONSTANT) {
    echo $a.' '.foo:PRE_71_CONSTANT;
}

?>
```

Specs

Short name	Classes/CouldBeProtectedConstant
Rulesets	*ClassReview*
Exakt since	0.12.11
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.63 Could Be Protected Method

Those methods are declared public, but are never used publicly. They may be made protected.

```
<?php

class foo {
    // Public, and used publicly
    public publicMethod() {}

    // Public, but never used outside the class or its children
    public protectedMethod() {}

    private function bar() {
        $this->protectedMethod();
    }
}

$foo = new Foo();
$foo->publicMethod();

?>
```

These properties may even be made private.

Specs

Short name	Classes/CouldBeProtectedMethod
Rulesets	*ClassReview*
Exakt since	0.12.11
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.64 Could Be Protected Property

Those properties are declared public, but are never used publicly. They may be made protected.

```
<?php

class foo {
    // Public, and used publicly
    public $publicProperty;
    // Public, but never used outside the class or its children
    public $protectedProperty;

    function bar() {
        $this->protectedProperty = 1;
    }
}

$foo = new Foo();
$foo->publicProperty = 3;

?>
```

This property may even be made private.

Specs

Short name	Classes/CouldBeProtectedProperty
Rulesets	*ClassReview*
Exakt since	0.9.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.65 Method Could Be Static

A method that doesn't make any usage of $this could be turned into a static method.

While static methods are usually harder to handle, recognizing the static status is a first step before turning the method into a standalone function.

```php
<?php

class foo {
    static $property = 1;

    // legit static method
    static function staticMethod() {
        return self::$property;
    }

    // This is not using $this, and could be static
    function nonStaticMethod() {
        return self::$property;
    }

    // This is not using $this nor self, could be a standalone function
    function nonStaticMethod() {
        return self::$property;
    }
}

?>
```

Suggestions

- Make the method static
- Make the method a standalone function
- Make use of $this in the method : may be it was forgotten.

Specs

Short name	Classes/CouldBeStatic
Rulesets	*Analyze*, *ClassReview*
Exakt since	1.5.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*FuelCMS*, *ExpressionEngine*

13.2.66 Could Be Stringable

Stringable is an interface that mark classes as string-castable. It is introduced in PHP 8.0.

Classes that defined a __toString() magic method may be turned into a string when the typehint, argument, return or property, requires it. This is not the case when strict_types is activated. Yet, until PHP 8.0, there was nothing to identify a class as such.

```php
<?php
```

(continues on next page)

(continued from previous page)

```
// This class may implement Stringable
class x {
    function __tostring() {
        return 'asd';
    }
}

echo (new x);

?>
```

See also PHP RFC: Add Stringable interface.

Suggestions

-

Specs

Short name	Classes/CouldBeStringable
Rulesets	*Analyze*, *LintButWontExec*
Exakt since	2.1.9
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.67 Cyclic References

Avoid cyclic references.

Cyclic references happen when an object points to another object, which reciprocate. This is particularly possible with classes, when the child class has to keep a reference to the parent class.

```
<?php

class a {
    private $p = null;

    function foo() {
        $this->p = new b();
        // the current class is stored in the child class
        $this->p->m($this);
    }
}

class b {
    private $pb = null;

    function n($a) {
        // the current class keeps a link to its parent
```

(continues on next page)

(continued from previous page)

```
        $this->pb = $a;
    }
}
?>
```

Cyclic references, or circular references, are memory intensive : only the garbage collector can understand when they may be flushed from memory, which is a costly operation. On the other hand, in an acyclic reference code, the reference counter will know immediately know that an object is free or not.

See also About circular references in PHP and A Journey to find a memory leak.

Suggestions

- Use a different object when calling the child objects.
- Refactor your code to avoid the cyclic reference.

Specs

Short name	Classes/CyclicReferences
Rulesets	*Analyze*, *ClassReview*
Exakt since	2.1.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.68 Defined Class Constants

Connect class constants with their definition when it can find it. This includes class constants, one level of parent (extended) or interfaces (implemented).

```
<?php

class X {
    const Y = 2;

    function foo() {
        // This is defined on the line above
        echo self::Y;

        // This is not defined in the current code
        echo X::X;
    }
}

?>
```

Specs

Short name	Classes/DefinedConstants
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.69 Defined Parent MP

Check static calls with 'parent'.

```
<?php

class foo {
    protected function parentDefined() {}
    protected function unusedParentMethod() {}

    // visibility is checked too
    protected function unusuableParentMethod() {}
}

class bar extends foo {

    private function someMethod() {
        // reported
        parent::parentDefined();

        // not reported, as method is unreachable in parent
        parent::unusuableParentMethod();

        // not reported, as method is undefined in parent
        parent::parentUndefined();

    }

    protected function parentDefined2() {}
}

?>
```

Specs

Short name	Classes/DefinedParentMP
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.70 Defined Properties

List of properties that are explicitly defined in the class, its parents or traits.

```
<?php

class foo {
    // property definition
    private bar = 2;
}

?>
```

See also Properties.

Specs

Short name	Classes/DefinedProperty
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.71 Defined static:: Or self::

List of all defined static and self properties and methods.

```
<?php

class x {
    static public function definedStatic() {}
    private definedStatic = 1;

    public function method() {
        self::definedStatic();
        self::undefinedStatic();

        static::definedStatic;
        static::undefinedStatic;
    }
}

?>
```

Specs

Short name	Classes/DefinedStaticMP
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.72 Law of Demeter

The law of Demeter specifies a number of constraints to apply to methodcalls from within an method, so as to keep dependencies to a minimum.

```
<?php

class x {
    function foo($arg) {
        $this->foo();     // calling oneself is OK
        $this->x->bar(); // calling one's property is OK
        $arg->bar2();     // calling arg's methods is OK

        $local = new y();
        $z = $y->bar3();       // calling a local variable is OK

        $z->bar4();       // calling a method on a previous result is wrong
    }
}

?>
```

See also Do your objects talk to strangers? and Law of Demeter.

Suggestions

-

Specs

Short name	Classes/DemeterLaw
Rulesets	none
Exakt since	1.6.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.73 Dependant Abstract Classes

> Abstract classes should be autonomous. It is recommended to avoid depending on methods, constant or properties that should be made available in inheriting classes, without explicitly abstracting them.

The following abstract classes make usage of constant, methods and properties, static or not, that are not defined in the class. This means the inheriting classes must provide those constants, methods and properties, but there is no way to enforce this.

This may also lead to dead code : when the abstract class is removed, the host class have unused properties and methods.

```
<?php

// autonomous abstract class : all it needs is within the class
abstract class c {
    private $p = 0;

    function foo() {
        return ++$this->p;
    }
}

// dependant abstract class : the inheriting classes needs to provide some properties␣
↪or methods
abstract class c2 {
    function foo() {
        // $p must be provided by the extending class
        return ++$this->p;
    }
}

class c3 extends c2 {
    private $p = 0;
}
?>
```

See also Traits/DependantTrait.

Suggestions

- Make the class only use its own resources
- Split the class in autonomous classes
- Add local property definitions to make the class independent

Specs

Short name	Classes/DependantAbstractClass
Rulesets	*Analyze*, *ClassReview*
Exakt since	1.8.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.74 Different Argument Counts

Two methods with the same name shall have the same number of compulsory argument. PHP accepts different number of arguments between two methods, if the extra arguments have default values. Basically, they shall be called interchangeably with the same number of arguments.

The number of compulsory arguments is often mistaken for the same number of arguments. When this is the case, it leads to confusion between the two signatures. It will also create more difficulties when refactoring the signature.

While this code is legit, it is recommended to check if the two signatures could be synchronized, and reduce future surprises.

```
<?php

class x {
    function foo($a ) {}
}

class y extends x {
    // This method is compatible with the above, its signature is different
    function foo($a, $b = 1) {}
}

?>
```

Suggestions

- Extract the extra arguments into other methods
- Remove the extra arguments
- Add the extra arguments to all the signatures

Specs

Short name	Classes/DifferentArgumentCounts
Rulesets	*Analyze*, *ClassReview*
Exakt since	2.1.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.75 No Direct Call To Magic Method

PHP features magic methods, which are methods related to operators.

Magic methods, such as __get(), related to =, or __clone(), related to `clone`, are supposed to be used in an object environment, and not with direct call.

It is recommended to use the magic method with its intended usage, and not to call it directly. For example, typecast to `string` instead of calling the __toString() method.

```
<?php
// Write
  print $x->a;
// instead of
  print $x->__get('a');

class Foo {
    private $b = secret;

    public function __toString() {
        return strtoupper($this->b);
    }
}

$bar = new Foo();
echo (string) $bar;

?>
```

Accessing those methods in a static way is also discouraged.

See also Magic Methods and Magical PHP: '__call <https://www.garfieldtech.com/blog/magical-php-call>'_.

Specs

Short name	Classes/DirectCallToMagicMethod
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.76 Disconnected Classes

> One class is extending the other, but they do not use any features from one another. Basically, those two classes are using extends, but they are completely independent and may be separated.

When using the 'extends' keyword, the newly created classes are now acting together and making one. This should be visible in calls from one class to the other, or simply by property usage : they can't live without each other.

On the other hand, two completely independent classes that are merged, although they should be kept separated.

```
<?php

class A {
    private $pa = 1;

    function fooA() {
        $this->pa = 2;
    }
}

// class B and Class A are totally independent
```

(continues on next page)

(continued from previous page)

```
class B extends A {
    private $pb = 1;

    function fooB() {
        $this->pb = 2;
    }
}


// class C makes use of class A : it is dependent on the parent class
class C extends A {
    private $pc = 1;

    function fooB() {
        $this->pc = 2 + $this->fooA();
    }
}
?>
```

Suggestions

- Remove the extension
- Make actual usage of the classes, at least from one of them

Specs

Short name	Classes/DisconnectedClasses
Rulesets	*ClassReview*
Exakt since	1.8.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*WordPress*

13.2.77 Don't Send $this In Constructor

Don't use `$this` as an argument while in the __construct(). Until the constructor is finished, the object is not finished, and may be in an unstable state. Providing it to another code may lead to error.

This is true when the receiving structure puts the incoming object immediately to work, and don't store it for later use.

```
<?php

// $this is only provided when Foo is constructed
class Foo {
    private $bar = null;
    private $data = array();

    static public function build($data) {
```

(continues on next page)

(continued from previous page)

```
        $foo = new Foo($data);
        // Can't build in one call. Must make it separate.
        $foo->finalize();
    }

    private function __construct($data) {
        // $this is provided too early
        $this->data = $data;
    }

    function finalize() {
        $this->bar = new Bar($this);
    }
}

// $this is provided too early, leading to error in Bar
class Foo2 extends Foo {
    private $bar = null;
    private $data = array();

    function __construct($data) {
        // $this is provided too early
        $this->bar = new Bar($this);
        $this->data = $data;
    }
}

class Bar {
    function __construct(Foo $foo) {
        // the cache is now initialized with a wrong
        $this->cache = $foo->getIt();
    }
}

?>
```

See also Don't pass this out of a constructor.

Suggestions

- Finish the constructor first, then call an external object.
- Sending $this should be made accessible in a separate method, so external objects may call it.
- Sending the current may be the responsibility of the method creating the object.

Specs

Short name	Classes/DontSendThisInConstructor
Rulesets	*Analyze*
Exakt since	1.0.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Woocommerce*, *Contao*

13.2.78 Don't Unset Properties

Avoid unsetting properties. They would go undefined, and raise more warnings.

When getting rid of a property, assign it to null. This keeps the property in the object, yet allows existence check without errors.

```
<?php

class Foo {
    public $a = 1;
}

$a = new Foo();

var_dump((array) $a) ;
// la propriété est reportée, et null
// ['a' => null]

unset($a->a);

var_dump((array) $a) ;
//Empty []

// Check if a property exists
var_dump($a->b === null);

// Same result as above, but with a warning
var_dump($a->c === null);

?>
```

This analysis works on properties and static properties. It also reports magic properties being unset.

Thanks for Benoit Burnichon for the original idea.

Suggestions

- Never unset properties : set it to null or its default value instead
- Make the property an array, and set/unset its index

Specs

Short name	Classes/DontUnsetProperties
Rulesets	*Analyze*, *CI-checks*, *Top10*, *php-cs-fixable*
Exakt since	1.2.3
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Vanilla*, *Typo3*

13.2.79 Dynamic Classes

Dynamic calls of classes.

```
<?php

class x {
    static function staticMethod() {}
}

$class = 'x';
$class::staticMethod();

?>
```

Specs

Short name	Classes/DynamicClass
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.80 Dynamic Class Constant

Dynamic calls to class constants.

Constant may be dynamically called with the constant() function.

```
<?php
    // Dynamic access to 'E_ALL'
    echo constant('E_ALL');

    interface i {
        const MY_CONSTANT  = 1;
    }
```

(continues on next page)

(continued from previous page)

```
    // Dynamic access to 'E_ALL'
    echo constant('i::MY_CONSTANT');

?>
```

Specs

Short name	Classes/DynamicConstantCall
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.81 Dynamic Methodcall

Dynamic calls to class methods.

```
<?php

class x {
    static public function foo() {}
           public function bar() {}
}

$staticmethod = 'foo';
// dynamic static method call to x::foo()
x::$staticmethod();

$method = 'bar';
// dynamic method call to bar()
$object = new x();
$object->$method();

?>
```

Specs

Short name	Classes/DynamicMethodCall
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.82 Dynamic New

Dynamic instantiation of classes.

```
<?php
  $object = new $classname()
?>
```

.

Specs

Short name	Classes/DynamicNew
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.83 Dynamic Property

Dynamic access to class property.

```
<?php

class x {
    static public $foo = 1;
           public $bar = 2;
}

$staticproperty = 'foo';
// dynamic static property call to x::$foo
echo x::${$staticproperty};

$property = 'bar';
// dynamic property call to bar()
$object = new x();
$object->$property = 4;

?>
```

Specs

Short name	Classes/DynamicPropertyCall
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.84 Dynamic Self Calls

A class that calls itself dynamically. This may be property or methods.

Calling itself dynamically happens when a class is configured to call various properties (container) or methods.

```
<?php

class x {
    function foo() {
        $f = 'goo';
        return $this->$f();
    }

    function goo() {
        return rand(1, 10);
    }
}
?>
```

This rule is mostly useful internally, to side some special situations.

Specs

Short name	Classes/DynamicSelfCalls
Rulesets	none
Exakt since	2.1.1
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.85 Empty Classes

Classes that do no define anything at all. This is probably dead code.

Classes that are directly derived from an exception are omitted.

```
<?php

//Empty class
class foo extends bar {}

//Not an empty class
class foo2 extends bar {
    const FOO = 2;
}

//Not an empty class, as derived from Exception
class barException extends \Exception {}

?>
```

Suggestions

- Remove an empty class :it is probably dead code.
- Add some code to the class to make it concrete.

Specs

Short name	Classes/EmptyClass
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*WordPress*

13.2.86 Class Should Be Final By Ocramius

'Make your classes always final, if they implement an interface, and no other public methods are defined'.

When a class should be final, as explained by `Ocramius` (`Marco Pivetta`).

```
<?php

interface i1 {
    function i1() ;
}

// Class should final, as its public methods are in an interface
class finalClass implements i1 {
    // public interface
    function i1 () {}

    // private method
    private function a1 () {}
}

?>
```

See also When to declare classes final.

Specs

Short name	Classes/FinalByOcramius
Rulesets	*Analyze*
Exakt since	0.9.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.87 Final Class Usage

List of all final classes being used.

final may be applied to classes and methods.

```
<?php
class BaseClass {
   public function test() {
       echo 'BaseClass::test() called'.PHP_EOL;
   }

   final public function moreTesting() {
       echo 'BaseClass::moreTesting() called'.PHP_EOL;
   }
}

class ChildClass extends BaseClass {
   public function moreTesting() {
       echo 'ChildClass::moreTesting() called'.PHP_EOL;
   }
}
// Results in Fatal error: Cannot override final method BaseClass::moreTesting()
?>
```

See also Final Keyword.

Specs

Short name	Classes/Finalclass
Rulesets	*ClassReview*, *LintButWontExec*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.88 Final Methods Usage

List of all final methods being used.

final may be applied to classes and methods.

```
<?php
class BaseClass {
   public function test() {
       echo 'BaseClass::test() called'.PHP_EOL;
   }

   final public function moreTesting() {
       echo 'BaseClass::moreTesting() called'.PHP_EOL;
   }
}
```

(continues on next page)

(continued from previous page)

```
class ChildClass extends BaseClass {
   public function moreTesting() {
       echo 'ChildClass::moreTesting() called'.PHP_EOL;
   }
}
// Results in Fatal error: Cannot override final method BaseClass::moreTesting()
?>
```

See also Final Keyword.

Specs

Short name	Classes/Finalmethod
Rulesets	*ClassReview*, *LintButWontExec*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.89 Final Private Methods

PHP's private methods cannot be overwritten, as they are dedicated to the current class. That way, the `final` keyword is useless.

PHP 8.0 warns when it finds such a method.

```
<?php

class foo {
    // Final and private both prevent child classes to overwrite the method
    final private function bar() {}

    // Final and protected (or public) keep this method available, but not␣
↪overwritable
    final protected function bar() {}
}

?>
```

See also Final Keyword.

Suggestions

- Remove the final keyword
- Relax visibility

Specs

Short name	Classes/FinalPrivate
Rulesets	*CE*, *ClassReview*, *CompatibilityPHP80*
Exakt since	2.2.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.90 Fossilized Method

A method is fossilized when it is overwritten so often that changing a default value, a return type or an argument type is getting difficult.

This happens when a class is extended. When a method is overwritten once, it may be easy to update the signature in two places. The more methods are overwriting a parent method, the more difficult it is to update it.

This analysis counts the number of times a method is overwritten, and report any method that is ovrewritten more than 6 times. This threshold may be configured.

```
<?php

class x1 {
    // foo1() is never overwritten. It is easy to update.
    function foo1() {}

    // foo7() is overwritten seven times. It is hard to update.
    function foo7() {}
}

// classes x2 to x7, all overwrite foo7();
// Only x2 is presente here.
class x2 extends x1 {
    function foo7() {}
}

?>
```

Name	Default	Type	Description
fossilizationThreshold	6	integer	Minimal number of overwriting methods to consider a method difficult to update.

Specs

Short name	Classes/FossilizedMethod
Rulesets	*ClassReview*, *Typechecks*
Exakt since	2.0.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.91 Class Has Fluent Interface

Mark a class as such when it contains at least one fluent method. A fluent method is a method that returns $this, for chaining.

```
<?php

class foo {
    private $count = 0;

    function a() {
        ++$this->count;
        return $this;
    }

    function b() {
        $this->count += 2;
        return $this;
    }

    function c() {
        return $this->count;
    }
}

$bar = new foo();
print $bar->a()
          ->b()
          ->c();

// display 3 (1 + 2).

?>
```

See also **The basics of Fluent interfaces in PHP** **and** Fluent interface are evil

Specs

Short name	Classes/HasFluentInterface
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.92 Has Magic Property

The class has defined one of the magic methods.

The magic methods are : __call(), __callStatic(), __get(), __set(), __isset(), __unset(), __sleep(), __wakeup(), __toString(), __invoke(), __set_state(), __clone() and __debugInfo().

__construct() and __destruct() are omitted here.

```
<?php

class WithMagic {
    // some more methods, const or properties

    public function __get() {
        // doSomething();
    }
}

?>
```

See also Property overloading.

Specs

Short name	Classes/HasMagicProperty
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.93 Hidden Nullable

Argument with default value of null are nullable. Even when the `null` typehint (PHP 8.0), or the `?` operator are not used, setting the default value to null is allowed, and makes the argument nullable.

This doesn't happen with properties : they must be defined with the nullable type to accept a ‘‘null‘‘value as default value.

This doesn't happen with constant, which can't be typehinted.

```
<?php

// explicit nullable parameter $s
function bar(?string $s = null) {

// implicit nullable parameter $s
function foo(string $s = null) {
    echo $s ?? 'NULL-value';
}

// both display NULL-value
foo();
foo(null);

?>
```

See also Nullable types and Type declaration.

Suggestions

- Change the default value to a compatible literal : for example, `string $s = ''`
- Add the explicit `?` nullable operator, or ‘‘null‘‘with PHP 8.0
- Remove the default value

Specs

Short name	Classes/HiddenNullable
Rulesets	*Analyze*, *ClassReview*
Exakt since	2.1.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.94 Identical Methods

When the parent class and the child class have the same method, the child might drop it. This reduces code duplication.

Duplicate code in methods is often the results of code evolution, where a method was copied with the hierarchy, but the original wasn't removed.

This doesn't apply to *private* methods, which are reserved for one class.

```
<?php

class a {
    public function foo() {
        return rand(0, 100);
    }
}
```

(continues on next page)

(continued from previous page)

```
class b extends a {
    public function foo() {
        return rand(0, 100);
    }
}

?>
```

Suggestions

- Drop the method from the parent class, in particular if only one child uses the method.
- Drop the method from the child class, in particular if there are several children class
- Use an abstract method, and make sure every child has its own implementation
- Modify one of the methods so they are different

Specs

Short name	Classes/IdenticalMethods
Rulesets	none
Exakt since	1.8.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.95 Immutable Signature

Overwrites makes refactoring a method signature difficult. PHP enforces compatible signature, by checking if arguments have the same type, reference and default values.

In PHP 7.3, typehint had to be the same, or dropped. In PHP 7.4, typehint may be contravariant (arguments), or covariant (returntype).

This analysis may be configured with `maxOverwrite`. By default, a minimum of 8 overwritten methods is considered difficult to update.

```
<?php

// Changing any of the four foo() method signature will trigger a PHP warning
class a {
    function foo($a) {}
}

class ab1 extends a {
    // four foo() methods have to be refactored at the same time!
    function foo($ab1) {}
}
```

(continues on next page)

(continued from previous page)

```
class ab2 extends a {
    function foo($ab2) {}
}

class ab3 extends ab1 {
    function foo($abc1) {}
}

?>
```

When refactoring a method, all the related methodcall may have to be updated too. Adding a type, a default value, or a new argument with default value won't affect the calls, but only the definitions. Otherwise, calls will also have to be updated.

IDE may help with signature refactoring, such as Refactoring code.

See also Covariance and contravariance (computer science), extends.

Name	Default	Type	Description
maxOverwrite	8	integer	Minimal number of method overwrite to consider that any refactor on the method signature is now hard.

Specs

Short name	Classes/ImmutableSignature
Rulesets	*CE*
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.96 Implemented Methods Are Public

Class methods that are defined in an interface must be public. They cannot be either private, nor protected.

This error is not reported by lint, but is reported at execution time.

```
<?php

interface i {
    function foo();
}

class X {
    // This method is defined in the interface : it must be public
    protected function foo() {}

    // other methods may be private
    private function bar() {}
}
```

(continues on next page)

(continued from previous page)

```
?>
```

See also Interfaces and Interfaces - the next level of abstraction.

Suggestions

- Make the implemented method public

Specs

Short name	Classes/ImplementedMethodsArePublic
Rulesets	*Analyze*
Exakt since	0.11.5
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.97 Implement Is For Interface

With class heritage, implements should be used for interfaces, and extends with classes.

PHP defers the implements check until execution : the code in example does lint, but won,t run.

```
<?php

class x {
    function foo() {}
}

interface y {
    function foo();
}

// Use implements with an interface
class z implements y {}

// Implements is for an interface, not a class
class z implements x {}

?>
```

Suggestions

- Create an interface from the class, and use it with the implements keyword

Specs

Short name	Classes/ImplementIsForInterface
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.98 Incompatible Signature Methods

Methods should have the same signature when being overwritten.

The same signatures means the children class must have : + the same name + the same visibility or less restrictive + the same typehint or removed + the same default value or removed + a reference like its parent

This problem emits a fatal error, for abstract methods, or a warning error, for normal methods. Yet, it is difficult to lint, because classes are often stored in different files. As such, PHP do lint each file independently, as unknown parent classes are not checked if not present. Yet, when executing the code, PHP lint the actual code and may encounter a fatal error.

```
<?php

class a {
    public function foo($a = 1) {}
}

class ab extends a {
    // foo is overloaded and now includes a default value for $a
    public function foo($a) {}
}

?>
```

See also Object Inheritance.

Suggestions

- Make signatures compatible again

Specs

Short name	Classes/IncompatibleSignature
Rulesets	*Analyze*, *LintButWontExec*
Exakt since	1.3.3
Php Version	7.4-
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High
Examples	*SuiteCrm*

13.2.99 Incompatible Signature Methods With Covariance

Methods should have the compatible signature when being overwritten.

The same signatures means the children class must have : + the same name + the same visibility or less restrictive + the same contravariant typehint or removed + the same covariant return typehint or removed + the same default value or removed + a reference like its parent

This problem emits a fatal error, for abstract methods, or a warning error, for normal methods. Yet, it is difficult to lint, because classes are often stored in different files. As such, PHP do lint each file independently, as unknown parent classes are not checked if not present. Yet, when executing the code, PHP lint the actual code and may encounter a fatal error.

```
<?php

class a {
    public function foo($a = 1) {}
}

class ab extends a {
    // foo is overloaded and now includes a default value for $a
    public function foo($a) {}
}

?>
```

See also **Object Inheritance,** PHP RFC: Covariant Returns and Contravariant Parameters and *Incompatible Signature Methods*.

Suggestions

- Make signatures compatible again

Specs

Short name	Classes/IncompatibleSignature74
Rulesets	*Analyze*
Exakt since	1.3.3
Php Version	7.4+
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High
Examples	*SuiteCrm*

13.2.100 Instantiating Abstract Class

PHP cannot instantiate an abstract class.

The classes are actually abstract classes, and should be derived into a concrete class to be instantiated.

```
<?php

abstract class Foo {
```

(continues on next page)

(continued from previous page)

```
    protected $a;
}

class Bar extends Foo {
    protected $b;
}

// instantiating a concrete class.
new Bar();

// instantiating an abstract class.
// In real life, this is not possible also because the definition and the␣
↪instantiation are in the same file
new Foo();

?>
```

See also Class Abstraction.

Specs

Short name	Classes/InstantiatingAbstractClass
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.101 Insufficient Property Typehint

The typehint used for a class property doesn't cover all it usage.

The typehint is insufficient when a undefined method is called, or if members are access while the typehint is an interface.

```
<?php

class A {
    function a1() {}
}

// PHP 7.4 and more recent
class B {
    private A $a = null;

    function b2() {
        // this method is available in A
        $this->a->a1();
        // this method is NOT available in A
        $this->a->a2();
    }
}
```

(continues on next page)

(continued from previous page)

```
// Supported by all PHP versions
class C {
    private $a = null;

    function __construct(A $a) {
        $this->a = $a;
    }

    function b2() {
        // this method is available in A
        $this->a->a1();
        // this method is NOT available in A
        $this->a->a2();
    }
}

?>
```

This analysis relies on typehinted properties, as introduced in PHP 7.4. It also relies on typehinted assignations at construct time : the typehint of the assigned argument will be used as the property typehint. Getters and setters are not considered here.

Suggestions

- Change the typehint to match the actual usage of the object in the class.

Specs

Short name	Classes/InsufficientPropertyTypehint
Rulesets	*ClassReview*
Exakt since	2.0.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.102 Integer As Property

It is backward incompatible to use integers are property names. This feature was introduced in PHP 7.2.

If the code must be compatible with previous versions, avoid casting arrays to object.

```
<?php

// array to object
$arr = [0 => 1];
$obj = (object) $arr;
var_dump(
    $obj,
    $obj->{'0'}, // PHP 7.2+ accessible
```

(continues on next page)

(continued from previous page)

```
    $obj->{0} // PHP 7.2+ accessible

    $obj->{'b'}, // always been accessible
);
?>
```

See also PHP RFC: Convert numeric keys in object/array casts.

Specs

Short name	Classes/IntegerAsProperty
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*
Exakt since	1.0.4
Php Version	7.2+
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.103 Is A PHP Magic Property

Mark properties usage when they are actually a magic call.

```
<?php

class magicProperty {
    public $b;

    function __get($name) {
        // do something with the value
    }

    function foo() {
        $this->a;
        $this->b;
    }
}

?>
```

See also Magic Methods.

Specs

Short name	Classes/IsaMagicProperty
Rulesets	none
Exakt since	0.12.17
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.104 Is An Extension Class

Those classes belongs to a PHP Extensions.

```
<?php

// This is a native PHP class
$o = new Stdclass();

// This is not a native PHP class
$o = new Elephpant();

?>
```

Specs

Short name	Classes/IsExtClass
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.105 Is Interface Method

Mark a method as part of an interface that the current class implements.

```
<?php

interface i {
    function i20();
}

class x implements i {
    // This is an interface method
    function i20() {}

    // This is not an interface method
    function x20() {}
```

(continues on next page)

(continued from previous page)

```
}

?>
```

Specs

Short name	Classes/IsInterfaceMethod
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.106 Is Not Class Family

Mark a static method call as inside the family of classes. Children are not considered here.

```
<?php

class a {
    function familyMethod() {}
}

classs b {
    function foo() {
        self::familyMethod(); // This is a call to a family method
        b::notAFamilyMethod(); // This is a call to a method of a class outside the␣
↪family
    }
}
?>
```

Specs

Short name	Classes/IsNotFamily
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.107 Is Upper Family

Does the static call is made within the current hierarchy of class, or, is it made in the class, in the children or outside.

This applies to static methodcalls, property accesses and class constants.

```
<?php

class AAA            { function inAAA() {} }    // upper family : grand-parent
class AA extends AAA { function inAA()  {} }    // upper family : parent
class A  extends AA  { function inA()   {} }    // current family
class B  extends A   { function inB()   {} }    // lower family
class C              { function inC()   {} }    // outside family

?>
```

Specs

Short name	Classes/IsUpperFamily
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.108 Locally Unused Property

Those properties are defined in a class, and this class doesn't have any method that makes use of them.

While this is syntactically correct, it is unusual that defined resources are used in a child class. It may be worth moving the definition to another class, or to move accessing methods to the class.

```
<?php

class foo {
    public $unused, $used;// property $unused is never used in this class

    function bar() {
        $this->used++; // property $used is used in this method
    }
}

class foofoo extends foo {
    function bar() {
        $this->unused++; // property $unused is used in this method, but defined in␣
↪the parent class
    }
}

?>
```

Suggestions

- Move the property definition to the child classes
- Move some of the child method, using the property, to the parent class

Specs

Short name	Classes/LocallyUnusedProperty
Rulesets	*Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.109 Locally Used Property

Properties that are used in the class where they are defined.

```
<?php

class foo {
    public $unused, $used;// property $unused is never used in this class

    function bar() {
        $this->used++; // property $used is used in this method
    }
}

$foo = new Foo();
$foo->unused = 'here'; // property $unused is used outside the class definition
?>
```

Specs

Short name	Classes/LocallyUsedProperty
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.110 Magic Methods

List of PHP magic methods being used. The magic methods are

__call(), __callStatic(), __get(), __set(), __isset(), __unset(), __sleep(), __wakeup(), __toString(), __invoke(), __set_state(), __clone() and __debugInfo().

__construct and __destruct are omitted here, as they are routinely used to create and destroy objects.

```
<?php

class foo{
```

(continues on next page)

(continued from previous page)

```
    // PHP Magic method, called when cloning an object.
    function __clone() {}
}
?>
```

See also Magic Method.

Specs

Short name	Classes/MagicMethod
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.111 Magic Properties

List of magic properties used in the code

Suggestions

-

Specs

Short name	Classes/MagicProperties
Rulesets	none
Exakt since	1.9.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.112 Assign Default To Properties

Properties may be assigned default values at declaration time. Such values may be later modified, if needed.

```
<?php

class foo {
    private $propertyWithDefault = 1;
    private $propertyWithoutDefault;
    private $propertyThatCantHaveDefault;
```

(continues on next page)

(continued from previous page)

```
    public function __construct() {
        // Skip this extra line, and give the default value above
        $this->propertyWithoutDefault = 1;

        // Static expressions are available to set up simple computation at␣
↪definition time.
        $this->propertyWithoutDefault = OtherClass::CONSTANT + 1;

        // Arrays, just like scalars, may be set at definition time
        $this->propertyWithoutDefault = [1,2,3];

        // Objects or resources can't be made default. That is OK.
        $this->propertyThatCantHaveDefault = fopen('/path/to/file.txt');
        $this->propertyThatCantHaveDefault = new Fileinfo();
    }
}

?>
```

Default values will save some instructions in the constructor, and makes the value obvious in the code.

Suggestions

- Add a default value whenever possible. This is easy for scalars, and array()

Specs

Short name	Classes/MakeDefault
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	use-properties-default-values
Examples	*LiveZilla*, *phpMyAdmin*

13.2.113 Make Global A Property

Calling global (or $GLOBALS) in methods is slower and less testable than setting the global to a property, and using this property.

Using properties is slightly faster than calling global or $GLOBALS, though the gain is not important.

Setting the property in the constructor (or in a factory), makes the class easier to test, as there is now a single point of configuration.

```
<?php

// Wrong way
class fooBad {
```

(continues on next page)

(continued from previous page)

```
    function x() {
        global $a;
        $a->do();
        // Or $GLOBALS['a']->do();
    }
}

class fooGood {
    private $bar = null;

    function __construct() {
        global $bar;
        $this->bar = $bar;
        // Even better, do this via arguments
    }

    function x() {
        $this->a->do();
    }
}

?>
```

Suggestions

- Avoid using global variables, and use properties instead
- Remove the usage of these global variables

Specs

Short name	Classes/MakeGlobalAProperty
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.114 Make Magic Concrete

Speed up execution by replacing magic calls by concrete properties.

Magic properties are managed dynamically, with `__get``and ``__set`. They replace property access by a methodcall, and they are much slower than the first.

When a property name is getting used more often, it is worth creating a concrete property, and skip the method call. The threshold for 'magicMemberUsage' is 1, by default.

```
<?php

class x {
```

(continues on next page)

(continued from previous page)

```
    private $values = array('a' => 1,
                            'b' => 2);

    function __get($name) {
        return $this->values[$name] ?? '';
    }
}

$x = new x();
// Access to 'a' is repeated in the code, at least 'magicMemberUsage' time (cf␣
↪configuration below)
echo $x->a;

?>
```

See also Performances/MemoizeMagicCall.

Suggestions

- Make frequently used properties concrete; keep the highly dynamic as magic

Name	Default	Type	Description
magicMemberUsage	1	integer	Minimal number of magic member usage across the code, to trigger a concrete property.

Specs

Short name	Classes/MakeMagicConcrete
Rulesets	*Performances*
Exakt since	1.8.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.115 Method Is Overwritten

This marks an method that is overwritten in a child class.

```
<?php

class A {
    function intactMethodA() {}          // Not overwritten in any children
    function overwrittenMethodInAA() {} // overwritten in AA
}

class AA extends A {
    function intactMethodAA() {}         // Not overwritten, because no extends
    function overwrittenMethodInAA() {} // Not overwritten, because no extends
```

(continues on next page)

(continued from previous page)

```
}

?>
```

Specs

Short name	Classes/MethodIsOverwritten
Rulesets	none
Exakt since	0.10.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.116 Method Signature Must Be Compatible

Make sure methods signature are compatible.

PHP generates the infamous Fatal error at execution : `Declaration of FooParent\:\:Bar() must be compatible with FooChildren\:\:Bar()`

```
<?php

class x {
    function xa() {}
}

class xxx extends xx {
    function xa($a) {}
}

?>
```

Suggestions

- Fix the child class method() signature.
- Fix the parent class method() signature, after checking that it won't affect the other children.

Specs

Short name	Classes/MethodSignatureMustBeCompatible
Rulesets	*Analyze*, *LintButWontExec*
Exakt since	1.2.9
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.117 Method Used Below

Mark methods that are used in children classes.

```
<?php

class foo {
    // This method is used in children
    protected function protectedMethod() {}

    // This method is not used in children
    protected function localProtectedMethod() {}

    private function foobar() {
        // protectedMethod is used here, but defined in parent
        $this->localProtectedMethod();
    }
}

class foofoo extends foo {
    private function bar() {
        // protectedMethod is used here, but defined in parent
        $this->protectedMethod();
    }
}

?>
```

This doesn't mark the current class, nor the (grand-)'parent <https://www.php.net/manual/en/language.oop5.paamayim-nekudotayim.php>'_ ones.

Specs

Short name	Classes/MethodUsedBelow
Rulesets	none
Exakt since	0.12.11
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.118 Mismatch Properties Typehints

Properties must match within the same family.

When a property is declared both in a parent class, and a child class, they must have the same type. The same type includes a possible null value.

This doesn't apply to private properties, which are only visible locally.

```
<?php

// property $p is declared as an object of type a
class x {
```

(continues on next page)

(continued from previous page)

```
    protected A $p;
}

// property $p is declared again, this time without a type
class a extends x {
    protected $p;
}
?>
```

This code will lint, but not execute.

Suggestions

- Remove some of the property declarations, and only keep it in the highest ranking parent
- Match the typehints of the property declarations
- Make the properties private
- Remove the child class (or the parent class)

Specs

Short name	Classes/MismatchProperties
Rulesets	*Analyze*, *ClassReview*, *LintButWontExec*
Exakt since	2.1.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.119 Missing Abstract Method

Abstract methods must have a non-abstract version for the class to be complete. A class that is missing one abstract definition cannot be instantiated.

```
<?php

// This is a valid definition
class b extends a {
    function foo() {}
    function bar() {}
}

// This compiles, but will emit a fatal error if instantiated
class c extends a {
    function bar() {}
}

// This illustration lint but doesn't run.
// moving this class at the beginning of the code will make lint fail
abstract class a {
```

(continues on next page)

(continued from previous page)

```
    abstract function foo() ;
}

?>
```

See also Classes Abstraction.

Suggestions

- Implement the missing methods
- Remove the partially implemented class
- Mark the partially implemented class abstract

Specs

Short name	Classes/MissingAbstractMethod
Rulesets	*Analyze*, *ClassReview*
Exakt since	2.1.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.120 Multiple Classes In One File

It is regarded as a bad practice to store several classes in the same file. This is usually done to make life of __autoload() easier.

It is often unexpected to find class `foo` in the `bar.php` file. This is also the case for interfaces and traits.

```
<?php

// three classes in the same file
class foo {}
class bar {}
class foobar{}

?>
```

One good reason to have multiple classes in one file is to reduce include time by providing everything into one nice include.

See also Is it a bad practice to have multiple classes in the same file?.

Suggestions

- Split the file into smaller files, one for each class

Specs

Short name	Classes/MultipleClassesInFile
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.121 Multiple Class Declarations

It is possible to declare several times the same class in the code. PHP will not mention it until execution time, since declarations may be conditional.

```
<?php

$a = 1;

// Conditional declaration
if ($a == 1) {
    class foo {
        function method() { echo 'class 1';}
    }
} else {
    class foo {
        function method() { echo 'class 2';}
    }
}

(new foo())->method();
?>
```

It is recommended to avoid declaring several times the same class in the code. The best practice is to separate them with namespaces, they are for here for that purpose. In case those two classes are to be used interchangeably, the best is to use an abstract class or an interface.

Suggestions

- Store classes with different names in different namespaces
- Change the name of the classes and give them a common interface to allow from common behavior

Specs

Short name	Classes/MultipleDeclarations
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.122 Multiple Property Declaration On One Line

Multiple properties are defined on the same line. They could be defined independantly, on separate expressions.

Keeping properties separate helps documenting and refactoring them independantly.

```
<?php

// multiple definition on one expression
class point {
    private $x, $y, $z;

    // more code
}

// one line, one definition
class point2 {
    private $x;

    private $y;

    private $z;

    // more code
}

?>
```

Suggestions

- Split the definitions to one by line

Specs

Short name	Classes/MultiplePropertyDeclarationOnOneLine
Rulesets	none
Exakt since	2.2.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.123 Multiple Identical Trait Or Interface

There is no need to use the same trait, or implements the same interface more than once.

Up to PHP 7.1 (at least), this doesn't raise any warning. Traits are only imported once, and interfaces may be implemented as many times as wanted.

```
<?php
```

(continues on next page)

(continued from previous page)

```
class foo {
    use t3,t3,t3;
}

class bar implements i,i,i {

}

?>
```

Suggestions

- Remove the duplicate trait or interfaces

Specs

Short name	Classes/MultipleTraitOrInterface
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.124 Classes Mutually Extending Each Other

Those classes are extending each other, creating an extension loop. PHP will yield a fatal error at running time, even if it is compiling the code.

```
<?php

// This code is lintable but won't run
class Foo extends Bar { }
class Bar extends Foo { }

// The loop may be quite large
class Foo extends Bar { }
class Bar extends Bar2 { }
class Bar2 extends Foo { }

?>
```

Specs

Short name	Classes/MutualExtension
Rulesets	*ClassReview*, *LintButWontExec*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.125 New On Functioncall Or Identifier

Object instantiation with new works with or without arguments. Both are valid in PHP.

The analyzed code has less than 10% of one of the two forms : for consistency reasons, it is recommended to make them all the same.

```
<?php

$a = new stdClass();

// Parenthesis are used when arguments are compulsory
$mysql = new MySQLI($host, $user, $pass);

// Parenthesis are omitted when no arguments are available
// That also makes the instantiation look different
$b = new stdClass;

?>
```

Specs

Short name	Classes/NewOnFunctioncallOrIdentifier
Rulesets	none
Exakt since	0.9.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.126 No Magic Method With Array

Magic method `__set()` doesn't work for array syntax.

When overloading properties, they can only be used for scalar values, excluding arrays. Under the hood, PHP uses `__get()` to reach for the name of the property, and doesn't recognize the following index as an array. It yields an error : Indirect modification of overloaded property.

```
<?php
```

(continues on next page)

(continued from previous page)

```
class c {
    private $a;
    private $o = array();

    function __get($name) {
        return $this->o[$name];
    }

    function foo() {
        // property b doesn't exists
        $this->b['a'] = 3;

        print_r($this);
    }

    // This method has no impact on the issue
    function __set($name, $value) {
        $this->o[$name] = $value;
    }
}

$c = new c();
$c->foo();

?>
```

It is possible to use the array syntax with a magic property : by making the `__get` returns an array, the syntax will actually extract the expected item in the array.

This is not reported by linting.

In this analysis, only properties that are found to be magic are reported. For example, using the b property outside the class scope is not reported, as it would yield too many false-positives.

See also Overload.

Suggestions

- Use a distinct method to append a new value to that property
- Assign the whole array, and not just one of its elements

Specs

Short name	Classes/NoMagicWithArray
Rulesets	*Analyze*, *CI-checks*, *LintButWontExec*
Exakt since	0.12.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	Medium

13.2.127 Non Nullable Getters

A getter needs to be nullable when a property is injected.

In particular, if the injection happens with a separate method, there is a time where the object is not consistent, and the property holds a default non-object value.

```
<?php

class Consistent {
    private $db = null;

    function __construct(Db $db) {
        $this->db = $db;
        // Object is immediately consistent
    }

    // Db might be null
    function getDb() {
        return $this->db;
    }
}

class Inconsistent {
    private $db = null;

    function __construct() {
        // No initialisation
    }

    // This might be called on time, or not
    // This typehint cannot be nullable, nor use null as default
    function setDb(DB $db) {
        return $this->db;
    }

    // Db might be null
    function getDb() {
        return $this->db;
    }
}
?>
```

Suggestions

- Remove the nullable option and the tests on `null`.

Specs

Short name	Classes/NonNullableSetters
Rulesets	*Analyze*, *ClassReview*
Exakt since	1.9.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.128 Forgotten Visibility

Some classes elements (property, method, constant) are missing their explicit visibility.

By default, it is public. It should at least be mentioned as public, or may be reviewed as protected or private.

Class constants support also visibility since PHP 7.1.

final, static and abstract are not counted as visibility. Only public, private and protected. The PHP 4 var keyword is counted as undefined.

Traits, classes and interfaces are checked.

```
<?php

// Explicit visibility
class X {
    protected sconst NO_VISIBILITY_CONST = 1; // For PHP 7.2 and later

    private $noVisibilityProperty = 2;

    public function Method() {}
}

// Missing visibility
class X {
    const NO_VISIBILITY_CONST = 1; // For PHP 7.2 and later

    var $noVisibilityProperty = 2; // Only with var

    function NoVisibilityForMethod() {}
}

?>
```

See also Visibility and Understanding The Concept Of Visibility In Object Oriented PHP.

Suggestions

- Always add explicit visibility to methods and constants in a class
- Always add explicit visibility to properties in a class, after PHP 7.4

Specs

Short name	Classes/NonPpp
Rulesets	*Analyze*, *CE*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	always-have-visibility
Examples	*FuelCMS*, *LiveZilla*

13.2.129 Non Static Methods Called In A Static

Static methods have to be declared as such (using the static keyword). Then, one may call them without instantiating the object.

PHP 7.0, and more recent versions, yield a deprecated error : `Non-`static <https://www.php.net/manual/en/language.oop5.static.php>`_ method A\:\:B() should not be called statically`.

PHP 5 and older doesn't check that a method is static or not : at any point, the code may call one method statically.

```
<?php
    class x {
        static public function sm( ) { echo __METHOD__.\n; }
        public public sm( ) { echo __METHOD__.\n; }
    }

    x::sm( ); // echo x::sm

    // Dynamic call
    ['x', 'sm']();
    [\x::class, 'sm']();

    $s = 'x::sm';
    $s();

?>
```

It is a bad idea to call non-static method statically. Such method may make use of special variable $this, which will be undefined. PHP will not check those calls at compile time, nor at running time.

It is recommended to update this situation : make the method actually static, or use it only in object context.

Note that this analysis reports all static method call made on a non-static method, even within the same class or class hierarchy. PHP silently accepts static call to any in-family method.

```
<?php
    class x {
        public function foo( ) { self::bar() }
        public function bar( ) { echo __METHOD__.\n; }
    }
?>
```

See also Static Keyword <https://www.php.net/manual/en/language.oop5.'static.php>'_.

Suggestions

- Call the method the correct way
- Define the method as static

Specs

Short name	Classes/NonStaticMethodsCalledStatic
Rulesets	*Analyze*, *CI-checks*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Medium
Examples	*Dolphin*, *Magento*

13.2.130 Class Without Parent

Classes should not refer to `parent` when it is not extending another class.

In PHP 7.4, it is a Deprecated warning. In PHP 7.3, it was a Fatal error, when the code was finally executed.

```
<?php

class x {
    function foo() {
        parent::foo();
    }
}
?>
```

Suggestions

- Update the class and make it extends another class
- Change the parent mention with a fully qualified name
- Remove the call to the parent altogether

Specs

Short name	Classes/NoParent
Rulesets	*Analyze*, *CI-checks*, *ClassReview*
Exakt since	1.9.0
Php Version	7.4-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.131 self, parent, static Outside Class

self, parent and static should be called inside a class or trait. PHP lint won't report those situations.

self, parent and static may be used in a trait : their actual value will be only known at execution time, when the trait is used.

```
<?php
// In the examples, self, parent and static may be used interchangeably

// This raises a Fatal error
//Fatal error: Uncaught Error: Cannot access static:: when no class scope is active
new static();

// static calls
echo self::CONSTANTE;
echo self::$property;
echo self::method();

// as a type hint
function foo(static $x) {
    doSomething();
}

// as a instanceof
if ($x instanceof static) {
    doSomething();
}

?>
```

Such syntax problem is only revealed at execution time : PHP raises a Fatal error.

The origin of the problem is usually a method that was moved outside a class, at least temporarily.

See also Scope Resolution Operator (::).

Specs

Short name	Classes/NoPSSOutsideClass
Rulesets	*Analyze*, *LintButWontExec*
Exakt since	0.10.3
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.132 No Public Access

The properties below are declared with public access, but are never used publicly. They can be made protected or private.

```
<?php

class foo {
    public $bar = 1;             // Public, and used in public space
    public $neverInPublic = 3;   // Public, but never used in outside the class

    function bar() {
        $neverInPublic++;
    }
}

$x = new foo();
$x->bar = 3;
$x->bar();

?>
```

Specs

Short name	Classes/NoPublicAccess
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.133 Normal Methods

Spot normal Methods.

```
<?php

class foo{
    // Normal method
```

(continues on next page)

(continued from previous page)

```
    private function bar() {}

    // Static method
    private static function barbar() {}
}

?>
```

Specs

Short name	Classes/NormalMethods
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.134 No Self Referencing Constant

It is not possible to use a constant to define itself in a class. It yields a fatal error at runtime.

The PHP error reads : `Cannot declare `self <https://www.php.net/manual/en/language.oop5.paamayim-nekudotayim.php>`_-referencing constant '`self <https://www.php.net/manual/en/language.oop5.paamayim-nekudotayim.php>`_\:\:C2'`. Unlike PHP which is self-referencing, self referencing variables can't have a value : just don't use that.

```
<?php
    class a {
        const C1 = 1;          // fully defined constant
        const C2 = self::C2;   // self referencing constant
        const C3 = a::C3 + 2; // self referencing constant
    }
?>
```

The code may access an already declared constant with self or with its class name.

```
<?php
    class a {
        const C1 = 1;
        const C2 = a::C1;
    }
?>
```

This error is not detected by linting. It is only detected at instantiation time : if the class is not used, it won't appear.

Suggestions

- Give a literal value to this constant
- Give a constant value to this constant : other class constants or constant are allowed here.

Specs

Short name	Classes/NoSelfReferencingConstant
Rulesets	*Analyze*, *ClassReview*, *LintButWontExec*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.135 Null On New

Until PHP 7, some classes instantiation could yield null, instead of throwing an exception.

After issuing a 'new' with those classes, it was important to check if the returned object were null or not. No exception were thrown.

```
<?php

// Example extracted from the wiki below
$mf = new MessageFormatter('en_US', '{this was made intentionally incorrect}');
if ($mf === null) {
    echo 'Surprise!';
}

?>
```

This inconsistency has been cleaned in PHP 7 : see See Internal Constructor Behavior

See also PHP RFC: Constructor behaviour of internal classes.

Suggestions

- Remove the check on null after a new instantiation

Specs

Short name	Classes/NullOnNew
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	7.0-
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.136 Old Style Constructor

PHP classes used to have the method bearing the same name as the class acts as the constructor. That was PHP 4, and early PHP 5.

The manual issues a warning about this syntax : `Old style constructors are DEPRECATED in PHP 7.0, and will be removed in a future version. You should always use `__construct() <https://www.php.net/manual/en/language.oop5.decon.php>`_ in new code.`

```
<?php

namespace {
    // Global namespace is important
    class foo {
        function foo() {
            // This acts as the old-style constructor, and is reported by PHP
        }
    }

    class bar {
        function __construct() { }
        function bar() {
            // This doesn't act as constructor, as bar has a __construct() method
        }
    }
}

namespace Foo\Bar{
    class foo {
        function foo() {
            // This doesn't act as constructor, as bar is not in the global namespace
        }
    }
}

?>
```

This is no more the case in PHP 5, which relies on `__construct()` to do so. Having this old style constructor may bring in confusion, unless you are also supporting old time PHP 4.

Note that classes with methods bearing the class name, but inside a namespace are not following this convention, as this is not breaking backward compatibility. Those are excluded from the analyze.

See also Constructors and Destructors.

Suggestions

- Remove old style constructor and make it `__construct()`
- Remove old libraries and use a modern component

Specs

Short name	Classes/OldStyleConstructor
Rulesets	*Analyze*, *CE*, *CompatibilityPHP80*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-php4-class-syntax

13.2.137 Var Keyword

Var was used in PHP 4 to mark properties as public. Nowadays, new keywords are available : public, protected, private. Var is equivalent to public.

It is recommended to avoid using var, and explicitly use the new keywords.

```
<?php

class foo {
    public $bar = 1;
    // Avoid var
    //var $bar = 1;
}

?>
```

See also Visibility.

Suggestions

- It is recommended to avoid using var, and explicitly use the new keywords : private, protected, public

Specs

Short name	Classes/OldStyleVar
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-php4-class-syntax
Examples	*xataface*

13.2.138 One Object Operator Per Line

Avoid using more than one operator -> per line, to prevent information overload.

```
<?php

// Spread operators on multiple lines
$object->firstMethodCall()
       ->property
       ->secondMethodCall();

// This is not readable
$object->firstMethodCall()->property->secondMethodCall();

// This is OK, as objects are different.
$a2->b2($c2->d2, $e2->f2);

?>
```

Specs

Short name	Classes/OneObjectOperatorPerLine
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.139 Only Static Methods

Marks a class that has only static methods.

Specs

Short name	Classes/OnlyStaticMethods
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.140 Order Of Declaration

The order used to declare members and methods has a great impact on readability and maintenance. However, practices varies greatly. As usual, being consistent is the most important and useful.

The suggested order is the following : traits, constants, properties, methods. Optional characteristics, like final, static... are not specified. Special methods names are not specified.

```
<?php

class x {
    use traits;

    const CONSTANTS = 1;
    const CONSTANTS2 = 1;
    const CONSTANTS3 = 1;

    private $property = 2;
    private $property2 = 2;
    private $property3 = 2;

    public function foo() {}
    public function foo2() {}
    public function foo3() {}
    public function foo4() {}
}

?>
```

Suggestions

- Always declare class elements (traits, constants, properties, methods) in the same order.

Specs

Short name	Classes/OrderOfDeclaration
Rulesets	none
Exakt since	0.11.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Medium

13.2.141 Overwritten Class Const

Those class constants are overwritten in a parent class. This may lead to confusion, as the value of the constant may change depending on the way it is called.

```
<?php

class foo {
    const C = 1;
}

class bar extends foo {
    const C = 2;

    function x() {
        // depending on the access to C, value is different.
```

(continues on next page)

(continued from previous page)

```
        print self::C.' '.static::C.' '.parent::C;
    }
}

?>
```

Specs

Short name	Classes/OverwrittenConst
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.142 Parent First

When calling parent constructor, always put it first in the `__construct` method. It ensures the parent is correctly build before the child start using values.

```
<?php

class father {
    protected $name = null;

    function __construct() {
        $this->name = init();
    }
}

class goodSon {
    function __construct() {
        // parent is build immediately,
        parent::__construct();
        echo my name is.$this->name;
    }
}

class badSon {
    function __construct() {
        // This will fail.
        echo my name is.$this->name;

        // parent is build later,
        parent::__construct();
    }
}

?>
```

This analysis doesn't apply to Exceptions.

Suggestions

- Use `parent\:\:__construct` as the first call in the constructor.

Specs

Short name	Classes/ParentFirst
Rulesets	*Analyze*, *Suggestions*
Exakt since	1.0.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*shopware*, *PrestaShop*

13.2.143 Properties Declaration Consistence

Properties may be declared all at once, or one by one.

The analyzed code has less than 10% of one of them : for consistency reasons, it is recommended to make them all the same.

It happens that choosing unique declarations or multiple depends on coding style and files.

```
<?php

class x {
    // Some declarations are made by batch
    private $a1 = 1,
            $a2 = 2;
    public $c = 1, $c2 = 2, $c4 = 3;

    // Most declarations are made one by one
    protected $b = 1;
    protected $b1 = 1;
    protected $b2 = 1;
    protected $b3 = 1;
    protected $b4 = 1;
    protected $b5 = 1;
    protected $b6 = 1;
    protected $b7 = 1;
    protected $b8 = 1;
    protected $b9 = 1;
    protected $b10 = 1;
    protected $b11 = 1;
    protected $b12 = 1;
    protected $b13 = 1;
    protected $b14 = 1;
    protected $b15 = 1;
    protected $b16 = 1;
    protected $b17 = 1;
    protected $b18 = 1;
    protected $b19 = 1;
```

(continues on next page)

(continued from previous page)

```
}
?>
```

See also Properties.

Specs

Short name	Classes/PPPDeclarationStyle
Rulesets	none
Exakt since	1.2.1
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.144 Property Could Be Local

A property only used in one method may be turned into a local variable.

Public an protected properties are omitted here : they may be modified somewhere else, in the code. This analysis may be upgraded to support those properties, when tracking of such properties becomes available.

Classes where only one non-magic method is available are omitted.

Traits with private properties are processed the same way.

```
<?php

class x {
    private $foo = 1;

    // Magic method, and constructor in particular, are omitted.
    function __construct($foo) {
        $this->foo = $foo;
    }

    function bar() {
        $this->foo++;

        return $this->foo;
    }

    function barbar() {}
}

?>
```

Suggestions

- Remove the property and make it an argument in the method
- Use that property elsewhere

Specs

Short name	Classes/PropertyCouldBeLocal
Rulesets	*Analyze*, *ClassReview*
Exakt since	1.1.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Mautic*, *Typo3*

13.2.145 Property Names

Variables are used in property definitions, when they are located in a class.

```
<?php

static $x; // not a property, a static variable

class foo {
    static $x; // now, this is a static property
    public $y, $z = 1; // normal properties

    public function bar() {
        static $x; // again, a static variable
    }
}

?>
```

See also Properties.

Specs

Short name	Classes/PropertyDefinition
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.146 Never Used Properties

Properties that are never used. They are defined in a class or a trait, but they never actually used.

Properties are considered used when they are used locally, in the same class as their definition, or in a parent class : a parent class is always included with the current class.

On the other hand, properties which are defined in a class, but only used in children classes is considered unused, since children may also avoid using it.

```php
<?php

class foo {
    public $usedProperty = 1;

    // Never used anywhere
    public $unusedProperty = 2;

    function bar() {
        // Used internally
        ++$this->usedProperty;
    }
}

class foo2  extends foo {
    function bar2() {
        // Used in child class
        ++$this->usedProperty;
    }
}

// Used externally
++$this->usedProperty;

?>
```

Suggestions

- Drop unused properties
- Change the name of the unused properties
- Move the properties to children classes
- Find usage for unused properties

Specs

Short name	Classes/PropertyNeverUsed
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*WordPress*

13.2.147 Property Used Above

Property used in the parent classes. If the definition of the property is in the child class, then the parent should not know about it and make usage of it.

It may also be used in the current class, or its children, though this is not reported by this analyzer.

```
<?php

class A {
    public function foo() {
        $this->pb++;
    }
}

class B extends A {
    protected $pb = 0;        // property     used above
    protected $pb2 = 0;       // property NOT used above
}

?>
```

See also **'Classes/PropertyUsedBelow'_**.

Suggestions

- Move the definition of the property to the upper class
- Move the usage of the property to the lower class

Specs

Short name	Classes/PropertyUsedAbove
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Medium

13.2.148 Property Used Below

Mark properties that are used in children classes.

```
<?php

class foo {
    // This property is used in children
    protected protectedProperty = 1;

    // This property is not used in children
    protected localProtectedProperty = 1;

    private function foobar() {
        // protectedProperty is used here, but defined in parent
        $this->localProtectedProperty = 3;
    }
}
```

(continues on next page)

(continued from previous page)

```
class foofoo extends foo {
    private function bar() {
        // protectedProperty is used here, but defined in parent
        $this->protectedProperty = 3;
    }
}

?>
```

This doesn't mark the current class, nor the (grand-)'parent <https://www.php.net/manual/en/language.oop5.paamayim-nekudotayim.php>'_ ones.

Specs

Short name	Classes/PropertyUsedBelow
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Medium

13.2.149 Property Used In One Method Only

> Properties should be used in several methods. When a property is used in only one method, this should have be of another shape.

Properties used in one method only may be used several times, and read only. This may be a class constant. Such properties are meant to be overwritten by an extending class, and that's possible with class constants.

Properties that read and written may be converted into a variable, static to the method. This way, they are kept close to the method, and do not pollute the object's properties.

```
<?php

class foo {
    private $once = 1;
    const ONCE = 1;
    private $counter = 0;

    function bar() {
        // $this->once is never used anywhere else.
        someFunction($this->once);
        someFunction(self::ONCE);   // Make clear that it is a
    }

    function bar2() {
        static $localCounter = 0;
        $this->counter++;

        // $this->once is only used here, for distinguising calls to someFunction2
        if ($this->counter > 10) { // $this->counter is used only in bar2, but it may␣
↪be used several times
```

(continues on next page)

(continued from previous page)

```
            return false;
        }
        someFunction2($this->counter);

        // $localCounter keeps track for all the calls
        if ($localCounter > 10) {
            return false;
        }
        someFunction2($localCounter);
    }
}

?>
```

Note : properties used only once are not returned by this analysis. They are omitted, and are available in the analysis *Used Once Property*.

Suggestions

- Drop the property, and inline the value
- Drop the property, and make the property a local variable
- Use the property in another method

Specs

Short name	Classes/PropertyUsedInOneMethodOnly
Rulesets	*Analyze*
Exakt since	0.10.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Contao*

13.2.150 Internally Used Properties

Properties that are used internally.

```
<?php

class x {
    public $internallyUsedProperty = 1;
    public $externallyUsedProperty = 1;
    public $alsoExternallyUsedProperty = 1;

    function foo() {
        $this->internallyUsedProperty = 2;
    }
}
```

(continues on next page)

(continued from previous page)

```
class y extends x {
    function bar() {
        $this->externallyUsedProperty = 3;
    }
}

$X = new x();
$X->alsoExternallyUsedProperty = 3;

?>
```

Specs

Short name	Classes/PropertyUsedInternally
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.151 Parent, Static Or Self Outside Class

Parent, static and self keywords must be used within a class or a trait. They make no sens outside a class or trait scope, as self and static refers to the current class and parent refers to one of parent above.

PHP 7.0 and later detect their usage at compile time, and emits a fatal error.

```
<?php

class x {
    const Y = 1;

    function foo() {
        // self is \x
        echo self::Y;
    }
}

const Z = 1;
// This lint but won't anymore
echo self::Z;

?>
```

Static may be used in a function or a closure, but not globally.

Specs

Short name	Classes/PssWithoutClass
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.152 Raised Access Level

A property's visibility may be lowered, but not raised.

This error may be detected when the classes are all in the same file : then, PHP reports the problem. However, when the classes are separated in different files, as it is customary, PHP won't check this at linting time, yielding a fatal error at execution time.

First file.

```
<?php

class Foo {
    public $publicProperty;
    protected $protectedProperty;
    private $privateProperty;
}
?>
```

Second file.

```
<?php

class Bar extends Foo {
    private $publicProperty;
    private $protectedProperty;
    private $privateProperty;   // This one is OK
}
?>
```

See also Visibility and Understanding the concept of visibility in object oriented php.

Suggestions

- Lower the visibility in the child class
- Raise the visibility in the parent class

Specs

Short name	Classes/RaisedAccessLevel
Rulesets	*ClassReview*, *LintButWontExec*
Exakt since	0.10.0
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.153 Redefined Class Constants

Redefined class constants.

Class constants may be redefined, though it is prone to errors when using them, as it is now crucial to use the right class name to access the right value.

```
<?php

class a {
    const A = 1;
}

class b extends a {
    const A = 2;
}

class c extends c { }

echo a::A, ' ', b::A, ' ', c::A;
// 1 2 2

?>
```

It is recommended to use distinct names.

Specs

Short name	Classes/RedefinedConstants
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.154 Redefined Default

Classes allows properties to be set with a default value. When those properties get, unconditionally, another value at constructor time, then one of the default value are useless. One of those definition should go : it is better to define properties outside the constructor.

```
<?php

class foo {
    public $redefined = 1;

    public function __construct( ) {
        $this->redefined = 2;
    }
}

?>
```

Suggestions

- Move the default assignation to the property definition
- Drop the reassignation in the constructor

Specs

Short name	Classes/RedefinedDefault
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Piwigo*

13.2.155 Redefined Methods

> Redefined methods are overwritten methods. Those methods are defined in different classes that are part of the same classes hierarchy.

Protected and public redefined methods replace each other. Private methods are kept separated, and depends on the caller to be distinguished.

```
<?php

class foo {
    function method() {
        return 1;
    }
}

class bar extends foo {
    function method() {
        return 2;
    }
}
?>
```

See also Object Inheritance.

Specs

Short name	Classes/RedefinedMethods
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.156 Redefined Private Property

Private properties are local to their defined class. PHP doesn't forbid the re-declaration of a private property in a child class.

However, having two or more properties with the same name, in the class hierarchy tends to be error prone.

```
<?php

class A {
    private $isReady = true;
}

class B {
    private $isReady = false;
}

?>
```

Specs

Short name	Classes/RedefinedPrivateProperty
Rulesets	*Analyze*
Exakt since	1.2.3
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Zurmo*

13.2.157 Redefined Property

Property redefined in a parent class.

Using heritage, it is possible to define several times the same property, at different levels of the hierarchy.

```
<?php

class foo {
    protected $aProperty = 1;
```

(continues on next page)

(continued from previous page)

```
}

class bar extends foo {
    // This property is redefined in the parent class, leading to potential confusion
    protected $aProperty = 1;
}

?>
```

When this is the case, it is difficult to understand which class will actually handle the property.

In the case of a private property, the different instances will stay distinct. In the case of protected or public properties, they will all share the same value.

It is recommended to avoid redefining the same property in a hierarchy.

Specs

Short name	Classes/RedefinedProperty
Rulesets	*ClassReview*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.158 Not Same Name As File

The class, interface or trait in this file as a different name, case included, than the file name.

In the following example, the file name is `Foo.php`. .. code-block:: php

<?php

// normal host of this file class Foo {

// some code

}

// case-typo this file class foo {

// some code

}

// strangely stored class class foo {

// some code

}

// This is valid name, but there is also a Foo class, and other classe in this file. interface Foo { }

?>

Specs

Short name	Classes/SameNameAsFile
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.159 Scalar Or Object Property

Property shouldn't use both object and scalar syntaxes. When a property may be an object, it is recommended to implement the Null Object pattern : instead of checking if the property is scalar, make it always object.

```
<?php

class x {
    public $display = 'echo';

    function foo($string) {
        if (is_string($this->display)) {
            echo $this->string;
        } elseif ($this->display instanceof myDisplayInterface) {
            $display->display();
        } else {
            print Error when displaying\n;
        }
    }
}

interface myDisplayInterface {
    public function display($string); // does the display in its own way
}

class nullDisplay implements myDisplayInterface {
    // implements myDisplayInterface but does nothing
    public function display($string) {}
}

class x2 {
    public $display = null;

    public function __construct() {
        $this->display = new nullDisplay();
    }

    function foo($string) {
        // Keep the check, as $display is public, and may get wrong values
        if ($this->display instanceof myDisplayInterface) {
            $display->display();
        } else {
            print Error when displaying\n;
```

(continues on next page)

(continued from previous page)

```
        }
    }
}

// Simple class for echo
class echoDisplay implements myDisplayInterface {
    // implements myDisplayInterface but does nothing
    public function display($string) {
        echo $string;
    }
}

?>
```

See also Null Object Pattern. and The Null Object Pattern.

Suggestions

- Only use one type of syntax with your properties.

Specs

Short name	Classes/ScalarOrObjectProperty
Rulesets	*Analyze*
Exakt since	0.12.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*SugarCrm*

13.2.160 Should Deep Clone

> By default, PHP makes a shallow clone. It only clone the scalars, and keep the reference to any object already referenced. This means that the cloned object and its original share any object they hold as property.

This is where the magic method __clone() comes into play. It is called, when defined, at clone time, so that the cloned object may clone all the needed sub-objects.

It is recommended to use the __clone() method whenever the objects hold objects.

```
<?php

class a {
    public $b = null;

    function __construct() {
        $this->b =  new Stdclass();
        $this->b->c = 1;
    }
```

(continues on next page)

(continued from previous page)

```
}

class ab extends a {
    function __clone() {
        $this->b = clone $this->b;
    }
}

// class A is shallow clone, so $a->b is not cloned
$a = new a();
$b = clone $a;
$a->b->c = 3;
echo $b->b->c;
// displays 3

// class Ab is deep clone, so $a->b is cloned
$a = new ab();
$b = clone $a;
$a->b->c = 3;
echo $b->b->c;
// displays 1

?>
```

See also PHP Clone and Shallow vs Deep Copying and Cloning objects.

Suggestions

-

Specs

Short name	Classes/ShouldDeepClone
Rulesets	*Suggestions*
Exakt since	1.7.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.161 Should Have Destructor

PHP destructors are called when the object has to be destroyed. By default, PHP calls recursively the destructor on internal objects, until everything is unset.

Unsetting objects and resources explicitly in the destructor is a good practice to reduce the amount of memory in use. It helps PHP resource counter to keep the numbers low, and easier to clean. This is a major advantage for long running scripts.

```
<?php
```

(continues on next page)

(continued from previous page)

```
class x {
    function __construct() {
        $this->p = new y();
    }

    function __destruct() {
        print __METHOD__.PHP_EOL;
        unset($this->p);
    }
}

class y {
    function __construct() {
        print __METHOD__.PHP_EOL;
        $this->p = new y();
    }

    function __destruct() {
        print __METHOD__.PHP_EOL;
        unset($this->p);
    }
}

$a = (new x);
sleep(1);

// This increment the resource counter by one for the property.
$p = $a->p;
unset($a);
sleep(3);

print 'end'.PHP_EOL;
// Y destructor is only called here, as the object still exists in $p.

?>
```

See also Destructor, and Php Destructors.

Suggestions

- Add a destruct method to the class to help clean at destruction time.

Specs

Short name	Classes/ShouldHaveDestructor
Rulesets	*Suggestions*
Exakt since	1.5.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.162 Could Use self

`self` keyword refers to the current class, or any of its parents. Using it is just as fast as the full class name, it is as readable and it is will not be changed upon class or namespace change.

It is also routinely used in traits : there, `self` represents the class in which the trait is used, or the trait itself.

```
<?php

class x {
    const FOO = 1;

    public function bar() {
        return self::FOO;
// same as return x::FOO;
    }
}

?>
```

See also Scope Resolution Operator (::).

Suggestions

- replace the explicit name with self

Specs

Short name	Classes/ShouldUseSelf
Rulesets	*Analyze*, *ClassReview*, *Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*WordPress*, *LiveZilla*

13.2.163 Should Use Local Class

Methods should use the defining class, or be functions.

Methods should use `$this` with another method or a property, or call `parent\:\:`. Static methods should call another static method, or a static property. Methods which are overwritten by a child class are omitted : the parent class act as a default value for the children class, and this is correct.

```
<?php

class foo {
    public function __construct() {
        // This method should do something locally, or be removed.
    }
}
```

(continues on next page)

(continued from previous page)

```
class bar extends foo {
    private $a = 1;

    public function __construct() {
        // Calling parent:: is sufficient
        parent::__construct();
    }

    public function barbar() {
        // This is acting on the local object
        $this->a++;
    }

    public function barfoo($b) {
        // This has no action on the local object. It could be a function or a␣
↪closure where needed
        return 3 + $b;
    }
}

?>
```

Note that a method using a class constant is not considered as using the local class, for this analyzer.

Suggestions

- Make this method a function
- Actually use $this, or any related attributes of the class

Specs

Short name	Classes/ShouldUseThis
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
ClearPHP	not-a-method

13.2.164 Static Methods Can't Contain $this

Static methods are also called `class methods` : they may be called even if the class has no instantiated object. Thus, the local variable `$this` won't exist, PHP will set it to NULL as usual.

```
<?php

class foo {
    // Static method may access other static methods, or property, or none.
    static function staticBar() {
```

(continues on next page)

(continued from previous page)

```
        // This is not possible in a static method
        return self::otherStaticBar() . static::$staticProperty;
    }

    static function bar() {
        // This is not possible in a static method
        return $this->property;
    }
}

?>
```

Either this is not a static method, which is fixed by removing the `static` keyword, or replace all $this mention by static properties `Class\:\:$property`.

See also Static Keyword <https://www.php.net/manual/en/language.oop5.‘static.php>‘_

Suggestions

- Remove any $this usage
- Turn any $this usage into a static call : $this->foo() => self::foo()

Specs

Short name	Classes/StaticContainsThis
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-static-this
Examples	*xataface*, *SugarCrm*

13.2.165 Static Methods

List of all static methods.

```
<?php

class foo {
    static public function staticMethod() {

    }

    public function notStaticMethod() {

    }

    private function method() {
```

(continues on next page)

(continued from previous page)

```
        // This is not a property
        new static();
    }
}

?>
```

Specs

Short name	Classes/StaticMethods
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.166 Static Methods Called From Object

> Static methods may be called without instantiating an object. As such, they never interact with the special variable '$this', as they do not depend on object existence.

Besides this, static methods are normal methods that may be called directly from object context, to perform some utility task.

To maintain code readability, it is recommended to call static method in a static way, rather than within object context.

```
<?php
    class x {
        static function y( ) {}
    }

    $z = new x( );

    $z->y( ); // Readability : no one knows it is a static call
    x::y( );  // Readability : here we know
?>
```

Suggestions

- Switch to static method syntax
- Remove the static option from the method

Specs

Short name	Classes/StaticMethodsCalledFromObject
Rulesets	*Analyze*, *CE*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.167 Static Properties

List of all static properties.

```
<?php

class foo {
    static public $staticProperty = 1;
           public $notStaticProperty = 2;

    private function method() {
        // This is not a property
        new static();
    }
}

function bar() {
    // This is not a static property
    static $staticVariable;

    //....
}

?>
```

Specs

Short name	Classes/StaticProperties
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.168 Strange Names For Methods

Those methods should have another name.

Ever wondered why the `__constructor` is never called? Or the `__consturct` ?

Those errors most often originate from typos, or quick fixes that where not fully tested. Other times, they were badly chosen, or ran into PHP's own reserved keywords.

```
<?php

class foo {
    // The real constructor
    function __construct() {}

    // The fake constructor
    function __constructor() {}

    // The 'typo'ed' constructor
    function __consturct() {}

    // This doesn't clone
    function clone() {}
}

?>
```

Suggestions

- Use the proper name
- Remove the method, when it is not used and tests still pass.

Specs

Short name	Classes/StrangeName
Rulesets	none
Exakt since	0.10.1
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.169 Swapped Arguments

Overwritten methods must be compatible, but argument names is not part of that compatibility.

Methods with the same name, in two classes of the same hierarchy, must be compatible for typehint, default value, reference. The name of the argument is not taken into account when checking such compatibility, at least until PHP 7.4.

```
<?php

class x {
    function foo($a, $b) {}

    function bar($a, $b) {}
}
```

(continues on next page)

(continued from previous page)

```
class y extends x {
    // foo is compatible (identical) with the above class
    function foo($a, $b) {}

    // bar is compatible with the above class, yet, the argument might not receive␣
↪what they expect.
    function bar($b, $a) {}
}

?>
```

This analysis reports argument lists that differs in ordering. This analysis doesn't report argument lists that also differs in argument names.

Suggestions

- Make sure the names of the argument are in the same order in all classes and interfaces

Specs

Short name	Classes/SwappedArguments
Rulesets	*Analyze*
Exakt since	2.1.5
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.170 Test Class

Those are test classes, based on popular UT frameworks.

Specs

Short name	Classes/TestClass
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.171 $this Belongs To Classes Or Traits

$this variable represents the current object, inside a class or trait scope.

It is a pseudo-variable, and should be used within class's or trait's methods and not outside. It should also not be used in static methods.

PHP 7.1 is stricter and check for $this at several situations. Some are found by static analysis, some are dynamic analysis.

```
<?php

// as an argument
function foo($this) {
    // Using global
    global $this;
    // Using static (not a property)
    static $this;

    // Can't unset it
    unset($this);

    try {
        // inside a foreach
        foreach($a as $this) {  }
        foreach($a as $this => $b) {  }
        foreach($a as $b => $this) {  }
    } catch (Exception $this) {
        // inside a catch
    }

    // with Variable Variable
    $a = this;
    $$a = 42;
}

class foo {
    function bar() {
        // Using references
        $a =& $this;
        $a = 42;

        // Using extract(), parse_str() or similar functions
        extract([this => 42]);  // throw new Error(Cannot re-assign $this)
        var_dump($this);
    }

    static function __call($name, $args) {
        // Using __call
        var_dump($this); // prints object(C)#1 (0) {}, php-7.0 printed NULL
        $this->test();   // prints ops
    }

}
?>
```

Suggestions

- Do not use *$this* as a variable name, except for the current object, in a class, trait or closure.

Specs

Short name	Classes/ThisIsForClasses
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*OpenEMR*

13.2.172 $this Is Not An Array

`$this` variable represents the current object and it is not an array.

This is unless the class (or its parents) has the `ArrayAccess` interface, or extends `ArrayObject` or `SimpleXMLElement`.

```
<?php

// $this is an array
class Foo extends ArrayAccess {
    function bar() {
        ++$this[3];
    }
}

// $this is not an array
class Foo2 {
    function bar() {
        ++$this[3];
    }
}

?>
```

See also ArrayAccess, ArrayObject and The Basics.

Suggestions

- Extends `ArrayObject`, or a class that extends it, to use `$this` as an array too.
- Implements `ArrayAccess` to use `$this` as an array too.
- Use a property in the current class to store the data, instead of $this directly.

Specs

Short name	Classes/ThisIsNotAnArray
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.173 $this Is Not For Static Methods

Static methods shouldn't use $this variable.

$this variable represents an object, the current object. It is not compatible with a static method, which may operate without any object.

While executing a static method, $this is actually set to NULL.

```
<?php

class foo {
    static $staticProperty = 1;

    // Static methods should use static properties
    static public function count() {
        return self::$staticProperty++;
    }

    // Static methods can't use $this
    static public function bar() {
        return $this->a;    // No $this usage in a static method
    }
}

?>
```

See also Static Keyword <https://www.php.net/manual/en/language.oop5.‘static.php>‘_.

Suggestions

- Remove the static keyword on the method, and update all calls to this method to use $this
- Remove the usage of $this in the method, replacing it with static properties
- Make $this an argument (and change its name) : then, make the method a function

Specs

Short name	Classes/ThisIsNotForStatic
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-static-this

13.2.174 Throw In Destruct

> According to the manual, `Attempting to throw an exception from a destructor (called in the time of script termination) causes a fatal error.`

The destructor may be called during the lifespan of the script, but it is not certain. If the exception is thrown later, the script may end up with a fatal error.

Thus, it is recommended to avoid throwing exceptions within the `__destruct` method of a class.

```
<?php

// No exception thrown
class Bar {
    function __construct() {
        throw new Exception('__construct');
    }

    function __destruct() {
        $this->cleanObject();
    }
}

// Potential crash
class Foo {
    function __destruct() {
        throw new Exception('__destruct');
    }
}

?>
```

See also Constructors and Destructors.

Suggestions

- Remove any exception thrown from a destructor

Specs

Short name	Classes/ThrowInDestruct
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.175 Too Many Children

Classes that have more than 15 children. It is worth checking if they cannot be refactored in anyway.

The threshold of 15 children can be configured. There is no technical limitation of the number of children and grand-children for a class.

The analysis doesn't work recursively : only direct generations are counted. Only children that can be found in the code are counted.

```
<?php

// parent class
// calling it grandparent to avoid confusion with 'parent'
class grandparent {}


class children1 extends grandparent {}
class children2 extends grandparent {}
class children3 extends grandparent {}
class children4 extends grandparent {}
class children5 extends grandparent {}
class children6 extends grandparent {}
class children7 extends grandparent {}
class children8 extends grandparent {}
class children9 extends grandparent {}
class children11 extends grandparent {}
class children12 extends grandparent {}
class children13 extends grandparent {}
class children14 extends grandparent {}
class children15 extends grandparent {}
class children16 extends grandparent {}
class children17 extends grandparent {}
class children18 extends grandparent {}
class children19 extends grandparent {}

?>
```

See also Why is subclassing too much bad (and hence why should we use prototypes to do away with it)?.

Suggestions

- Split the original class into more specialised classes

Name	Default	Type	Description
childrenClassCount	15	integer	Threshold for too many children classes for one class.

Specs

Short name	Classes/TooManyChildren
Rulesets	*Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Typo3*, *Woocommerce*

13.2.176 Too Many Dereferencing

Linking too many properties and methods, one to the other.

This analysis counts both static calls and normal call; methods, properties and constants. It also takes into account arrays along the way.

The default limit of chaining methods and properties is set to 7 by default.

```
<?php

// 9 chained calls.
$main->getA()->getB()->getC()->getD()->getE()->getF()->getG()->getH()->getI()->
↪property;

?>
```

Too many chained methods is harder to read.

Suggestions

-

Name	Default	Type	Description
tooManyDereferencing	7	integer	Maximum number of dereferencing.

Specs

Short name	Classes/TooManyDereferencing
Rulesets	*Analyze*
Exakt since	1.9.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.177 Too Many Finds

Too many methods called 'find*' in this class. It is may be time to consider the Specification pattern.

```
<?php

// quite a fishy interface
interface UserInterface {
    public function findByEmail($email);
    public function findByUsername($username);
    public function findByFirstName($firstname);
    public function findByLastName($lastname);
    public function findByName($name);
    public function findById($id);

    public function insert($user);
    public function update($user);
}

?>
```

See also On Taming Repository Classes in Doctrine , On Taming Repository Classes in Doctrine… Among other things., specifications.

Name	Default	Type	Description
minimumFinds	5	integer	Minimal number of prefixed methods to report.
findPrefix	find	string	list of prefix to use when detecting the 'find'. Comma-separated list, case insensitive.
findSuffix		string	list of fix to use when detecting the 'find'. Comma-separated list, case insensitive.

Specs

Short name	Classes/TooManyFinds
Rulesets	*Analyze*
Exakt since	0.10.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.178 Too Many Injections

When a class is constructed with more than four dependencies, it should be split into smaller classes.

```
<?php

// This class relies on 5 other instances.
// It is probably doing too much.
class Foo {
```

(continues on next page)

(continued from previous page)

```
    public function __construct(
            A $a,
            B $b,
            C $c,
            D $d
            E $e ) {
        $this->a = $a;
        $this->b = $b;
        $this->d = $d;
        $this->d = $d;
        $this->e = $e;
    }
}

?>
```

See also Dependency Injection Smells.

Suggestions

- Split the class into smaller classes. Try to do less in that class.

Name	Default	Type	Description
injectionsCount	5	integer	Threshold for too many injected parameters for one class.

Specs

Short name	Classes/TooManyInjections
Rulesets	*Analyze*
Exakt since	0.11.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*NextCloud*, *Thelia*

13.2.179 Magic Visibility

The class magic methods must have public visibility and cannot be static.

```
<?php

class foo{
    // magic method must bt public and non-static
    public static function __clone($name) {    }

    // magic method can't be private
    private function __get($name) {    }

    // magic method can't be protected
```

(continues on next page)

(continued from previous page)

```
    private function __set($name, $value) {     }

    // magic method can't be static
    public static function __isset($name) {     }
}

?>
```

See also Magic methods.

Specs

Short name	Classes/toStringPss
Rulesets	*CompatibilityPHP70*
Exakt since	0.8.4
Php Version	5.4-
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.180 DI Cyclic Dependencies

When injecting dependencies, classes that mutually depend on each other is a code smell.

Dependency injection should be organized as an acyclic tree-like structure

```
<?php

// Classes A and B depends on each other.
class A {
    protected $b;

    public function __construct(B $b) {
        $this->b = $b;
    }
}

class B {
    public $a;

    protected function setA(A $a) {
        $this->a = $a;
    }
}
?>
```

See also Dependency Injection Smells.

Specs

Short name	Classes/TypehintCyclicDependencies
Rulesets	none
Exakt since	0.11.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.181 Wrong Access Style to Property

> Use the right syntax when reaching for a property. Static properties use the `\:\:` operator, and non-static properties use `->`.

Mistaking one of the other raise two different reactions from PHP : `Access to undeclared `static <https://www.php.net/manual/en/language.oop5.static.php>`_ property` is a fatal error, while `PHP Notice: Accessing `static <https://www.php.net/manual/en/language.oop5.static.php>`_ property aa\:\:$a as non `static <https://www.php.net/manual/en/language.oop5.static.php>`_` is a notice.

```
<?php

class a {
    static public $a = 1;

    function foo() {
        echo self::$a; // right
        echo $this->a; // WRONG
    }
}

class b {
    public $b = 1;

    function foo() {
        echo $this->$b;  // right
        echo b::$b;      // WRONG
    }
}

?>
```

This analysis reports both static properties with a -> access, and non-static properties with a *::* access.

See also Static Keyword <https://www.php.net/manual/en/language.oop5.‘static.php>‘_.

Suggestions

- Match the property call with the definition
- Make the property static

Specs

Short name	Classes/UndeclaredStaticProperty
Rulesets	*Analyze*, *CI-checks*, *ClassReview*
Exakt since	1.4.9
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High
Examples	*HuMo-Gen*

13.2.182 Undefined Classes

Those classes are used in the code, but there are no definition for them.

This may happens under normal conditions, if the application makes use of an unsupported extension, that defines extra classes; or if some external libraries, such as PEAR, are not provided during the analysis.

```
<?php

// FPDF is a classic PDF class, that is usually omitted by Exakat.
$o = new FPDF();

// Exakat reports undefined classes in instanceof
// PHP ignores them
if ($o instanceof SomeClass) {
    // doSomething();
}

// Classes may be used in typehint too
function foo(TypeHintClass $x) {
    // doSomething();
}

?>
```

This analysis also checks in attributes.

Suggestions

- Fix the typo in the class name
- Add a missing 'use' expression
- Create the missing class

Specs

Short name	Classes/UndefinedClasses
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	Medium

13.2.183 Undefined Class Constants

Class constants that are used, but never defined. This should yield a fatal error upon execution, but no feedback at compile level.

```
<?php

class foo {
    const A = 1;
    define('B', 2);
}

// here, C is not defined in the code and is reported
echo foo::A.foo::B.foo::C;

?>
```

Suggestions

- Fix the name of the constant
- Add the constant to the current class or one of its parent
- Update the constant's visibility

Specs

Short name	Classes/UndefinedConstants
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.184 Undefined Parent

List of properties and methods that are accessed using `parent` keyword but are not defined in the parent classes.

This may compile but, eventually yields a fatal error during execution.

```
<?php

class theParent {
    // No bar() method
    // private bar() method is not accessible to theChild
}

class theChild extends theParent {
    function foo() {
        // bar is defined in theChild, but not theParent
        parent::bar();
    }

    function bar() {

    }
}

?>
```

Note that if the parent is defined using `extends someClass` but `someClass` is not available in the tested code, it will not be reported : it may be in composer, another dependency, or just missing.

See also parent <https://www.php.net/manual/en/keyword.‘parent.php>‘_.

Suggestions

- Remove the usage of the found method
- Add a definition for the method in the appropriate parent
- Fix the name of the method, and replace it with a valid definition
- Change ‘parent’ with ‘self’ if the method is eventually defined in the current class
- Change ‘parent’ with another object, if the method has been defined in another class
- Add the ‘extends’ keyword to the class, to actually have a parent class

Specs

Short name	Classes/UndefinedParentMP
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.185 Undefined Properties

List of properties that are not explicitly defined in the class, its parents or traits.

```
<?php

class foo {
    // property definition
    private bar = 2;

    function foofoo() {
        // $this->bar is defined in the class
        // $this->barbar is NOT defined in the class
        return $this->bar + $this->barbar;
    }
}

?>
```

It is possible to spot unidentified properties by using the PHP's magic methods `__get` and `__set`. Even if the class doesn't use magic methods, any call to an undefined property will be directed to those methods, and they can be used as a canary, warning that the code is missing a definition.

```
<?php

trait NoUnefinedProperties {
     function __get($name) {
             assert(false, "Attempt to read the $name property, on the class ".__
↪CLASS__;
     }

     function __set($name, $value) {
             assert(false, "Attempt to read the $name property, on the class ".__
↪CLASS__;
     }
}

?>
```

See also Properties.

Suggestions

- Add an explicit property definition, and give it `null` as a default value : this way, it behaves the same as undefined.
- Rename the property to one that exists already.

Specs

Short name	Classes/UndefinedProperty
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-undefined-properties
Examples	*WordPress*, *MediaWiki*

13.2.186 Undefined ::class

`\:\:class` doesn't check if a corresponding class exists.

`\:\:class` must be checked with a call to class_exists(). Otherwise, it may lead to a `Class 'foo' not found` or even silent dead code : this happens also with Catch and instanceof commands with undefined classes. PHP doesn't raise an error in that case.

```
<?php

class foo() {}

// prints foo
echo foo::class;

// prints bar though bar doesn't exist.
echo bar::class;

?>
```

See also Class Constants.

Specs

Short name	Classes/UndefinedStaticclass
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.3.5
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.187 Undefined static:: Or self::

self and static refer to the current class, or one of its parent. The property or the method may be undefined.

```
<?php
```

(continues on next page)

(continued from previous page)

```
class x {
    static public function definedStatic() {}
    private definedStatic = 1;

    public function method() {
        self::definedStatic();
        self::undefinedStatic();

        static::definedStatic;
        static::undefinedStatic;
    }
}

?>
```

See also Late ‘Static Bindings <https://www.php.net/manual/en/language.oop5.late-static-bindings.php>‘_.

Suggestions

- Define the missing method or property
- Remove usage of that undefined method or property
- Fix name to call an actual local structure

Specs

Short name	Classes/UndefinedStaticMP
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*xataface*, *SugarCrm*

13.2.188 Uninitialized Property

Uninitilized properties are not fully bootstrapped at the end of the constructor.

Properties may be inited at definition time, along with their visibility and type. Some types are not inited at definition time, as any object, so they should be inited during constructor. At the end of the former, all properties shall have a legit value, and be ready for usage.

```
<?php

class x {
    private $foo = null;
    private $uninited;

    function __construct($arg) {
        $this->foo = $args;
```

(continues on next page)

(continued from previous page)

```
        // $this->uninited is not inited, nor at definition, nor in constructor
        // it will hold null at the beginning of the next method call
    }
}

?>
```

Suggestions

- Remove the property, and move it to another class
- Add an initialisation for this property

Specs

Short name	Classes/UninitedProperty
Rulesets	*ClassReview*
Exakt since	2.0.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.189 Unitialized Properties

Properties that are not initialized in the constructor, nor at definition.

```
<?php

class X {
    private $i1 = 1, $i2;
    protected $u1, $u2;

    function __construct() {
        $this->i2 = 1 + $this->u2;
    }

    function m() {
        echo $this->i1, $this->i2, $this->u1, $this->u2;
    }
}
?>
```

With the above class, when m() is accessed right after instantiation, there will be a missing property. Using default values at property definition, or setting default values in the constructor ensures that the created object is consistent.

Suggestions

- Add an explicit initialization for each property.

Specs

Short name	Classes/UnitializedProperties
Rulesets	*Suggestions*, *Top10*
Exakt since	0.8.9
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*SPIP*

13.2.190 Unreachable Class Constant

Class constants may be unreachable due to visibility configuration.

Since PHP 7.1, class constants support visibility. Their usage may be restricted to the current class, or `private`, to classes that extends or are extended by the current class, or `protected`. They may also be `public`, just like it was before.

```
<?php

class Foo{
    private const PRIVATE = 1;
            const PUBLIC = 3;
}

// PHP 7.1- and older
echo Foo::PUBLIC;

// This is not accessible
echo Foo::PRIVATE;

?>
```

See also Class Constant and PHP RFC: Support Class Constant Visibility.

Suggestions

- Make the class constant protected, when the call to the constant is inside a related class.
- Create another constant, that may be accessible
- Make the class constant public

Specs

Short name	Classes/UnreachableConstant
Rulesets	*ClassReview*
Exakt since	1.5.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.191 Unresolved Catch

Catch clauses do not check for Exception existence.

Catch clauses check that the emitted expression is of the requested Class, but if that class doesn't exist in the code, the catch clause is always false. This is dead code.

```
<?php

try {
    // doSomething()
} catch (TypoedExxeption $e) { // Do not exist Exception
    // Fix this exception
} catch (Stdclass $e) {         // Exists, but is not an exception
    // Fix this exception
} catch (Exception $e) {         // Actual and effective catch
    // Fix this exception
}
?>
```

Suggestions

- Fix the name of the exception
- Remove the catch clause
- Add a use expression with a valid name
- Create/import the missing exception

Specs

Short name	Classes/UnresolvedCatch
Rulesets	*Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-unresolved-catch

13.2.192 Unresolved Classes

The following classes are instantiated in the code, but their definition couldn't be found.

```
<?php

class Foo extends Bar {
    private function foobar() {
        // here, parent is not resolved, as Bar is not defined in the code.
        return parent::$prop;
    }
}
```

(continues on next page)

(continued from previous page)

```
?>
```

Suggestions

- Check for namespaces and aliases and make sure they are correctly configured.

Specs

Short name	Classes/UnresolvedClasses
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.193 Unresolved Instanceof

The instanceof operator doesn't confirm if the compared class exists.

It checks if an variable is of a specific class. However, if the referenced class doesn't exist, because of a bug, a missed inclusion or a typo, the operator always fails, without a warning.

```
<?php

namespace X {
    class C {}

    // This is OK, as C is defined in X
    if ($o instanceof C) { }

    // This is not OK, as C is not defined in global
    // instanceof respects namespaces and use expressions
    if ($o instanceof \C) { }

    // This is not OK, as undefinedClass
    if ($o instanceof undefinedClass) { }

    // This is not OK, as $class is now a full namespace. It actually refers to \c,␣
↪which doesn't exist
    $class = 'C';
    if ($o instanceof $class) { }
}
?>
```

Make sure the following classes are well defined.

See also Instanceof.

Suggestions

- Remove the call to instanceof and all its dependencies.
- Fix the class name and use a class existing in the project.

Specs

Short name	Classes/UnresolvedInstanceof
Rulesets	*Analyze*, *Dead code*, *Top10*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-unresolved-instanceof
Examples	*WordPress*

13.2.194 Unused Classes

The following classes are never explicitly used in the code.

Note that this may be valid in case the current code is a library or framework, since it defines classes that are used by other (unprovided) codes. Also, this analyzer may find classes that are, in fact, dynamically loaded.

```
<?php

class unusedClasss {}
class usedClass {}

$y = new usedClass();

?>
```

Suggestions

- Remove unused classes
- Make use of unused classes
- Fix class name

Specs

Short name	Classes/UnusedClass
Rulesets	*Analyze*, *Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.195 Unused Class Constant

The class constant is unused. Consider removing it.

```
<?php

class foo {
    public const UNUSED = 1; // No mention in the code

    private const USED = 2;  // used constant

    function bar() {
        echo self::USED;
    }
}

?>
```

Suggestions

- Remove the class constant
- Use the class constant

Specs

Short name	Classes/UnusedConstant
Rulesets	*Analyze*, *ClassReview*
Exakt since	1.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.196 Unused Methods

Those methods are never called.

They are probably dead code, unless they are called dynamically.

This analysis omits methods which are in a class that makes dynamical self calls : `$this->$m()`. That way, any method may be called.

This analysis omits methods which are overwritten by a child class. That way, they are considered to provide a default behavior.

```
<?php

class foo {
    public function used() {
        $this->used();
    }
```

(continues on next page)

(continued from previous page)

```
    public function unused() {
        $this->used();
    }
}

class bar extends foo {
    public function some() {
        $this->used();
    }
}

$a = new foo();
$a->used();

?>
```

See also Dead Code: Unused Method.

Suggestions

- Make use of the method
- Remove the method
- Move the method to another class

Specs

Short name	Classes/UnusedMethods
Rulesets	*Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.197 Unused Private Methods

Private methods that are not used are dead code.

Private methods are reserved for the defining class. Thus, they must be used with the current class, with `$this` or `self\:\:`.

Protected methods, in a standalone class, are also included.

```
<?php

class Foo {
    // Those methods are used
    private function method() {}
    private static function staticMethod() {}
```

(continues on next page)

(continued from previous page)

```
    // Those methods are not used
    private function unusedMethod() {}
    private static function staticUnusedMethod() {}

    public function bar() {
        self::staticMethod();
        $this->method();
    }
}

?>
```

This analysis skips classes that makes self dynamic calls, such as `$this->$method()`.

Suggestions

- Remove the private method, as it is unused
- Add a call to this private method
- Change method visibility to make it available to other classes

Specs

Short name	Classes/UnusedPrivateMethod
Rulesets	*Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.198 Unused Private Properties

Unused static properties should be removed.

Unused private properties are dead code. They are usually leftovers of development or refactorisation : they used to have a mission, but are now left.

Being private, those properties are only accessible to the current class or trait. As such, validating the

```
<?php

class foo {
    // This is a used property (see bar method)
    private $used = 1;

    // This is an unused property
    private $unused = 2;

    function bar($a) {
        $this->used += $a;
```

(continues on next page)

(continued from previous page)

```
        return $this->used;
    }
}

?>
```

Suggestions

- Remove the property altogether
- Check if the property wasn't forgotten in the rest of the class
- Check if the property is correctly named
- Change the visibility to protected or public : may be a visibility refactoring was too harsh

Specs

Short name	Classes/UnusedPrivateProperty
Rulesets	*Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*OpenEMR*, *phpadsnew*

13.2.199 Unused Protected Methods

The following protected methods are unused in children class. As such, they may be considered for being private.

Methods reported by this analysis are not used by children, yet they are protected.

```
<?php

class Foo {
    // This method is not used
    protected function unusedBar() {}
    protected function usedInFoo() {}
    protected function usedInFooFoo() {}

    public function bar2() {
        // some code
        $this->usedInFoo();
    }
}

class FooFoo extends Foo {
    protected function bar() {}
```

(continues on next page)

(continued from previous page)

```
    public function bar2() {
        // some code
        $this->usedInFooFoo();
    }
}

class someOtherClass {
    protected function bar() {
        // This is not related to foo.
        $this->unusedbar();
    }
}

?>
```

No usage of those methods were found.

This analysis is impacted by dynamic method calls.

Suggestions

- Make use of the protected method in the code
- Remove the method

Specs

Short name	Classes/UnusedProtectedMethods
Rulesets	*Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.200 Use ::Class Operator

Use `\:\:class` to hardcode class names, instead of strings.

This is actually faster than strings, which are parsed at execution time, while `\:\:class` is compiled, making it faster to execute.

`\:\:class` operator is also able to handle use expressions, including aliases and local namespace. The code is easier to maintain. For example, the target class's namespace may be renamed, without changing the `\:\:class`, while the string must be updated.

`\:\:class` operator works with `self` and ‘‘static‘‘keywords.

```
<?php

namespace foo\bar;

use foo\bar\X as B;
```

(continues on next page)

(continued from previous page)

```
class X {}

$className = '\foo\bar\X';

$className = foo\bar\X::class;

$className = B\X;

$object = new $className;

?>
```

This is not possible when building the name of the class with concatenation.

This is a micro-optimization. This also helps static analysis, as it gives more information at compile time to analyse.

See also ::class.

Suggestions

- Replace strings by the ::class operator whenever possible

Specs

Short name	Classes/UseClassOperator
Rulesets	*Analyze*, *CI-checks*, *Performances*
Exakt since	0.8.7
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	Medium
Examples	*Typo3*

13.2.201 Used Classes

The following classes are used in the code.

Classes may be use when they are instantiated, or with static calls

```
<?php

class unusedClasss { const X = 1;}
class usedClass {}

$y = new usedClass(usedClass::X);

?>
```

Specs

Short name	Classes/UsedClass
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.202 Used Methods

Those methods are used in the code. This analysis is mostly useful for its contrary.

```
<?php

class foo {
    public function used() {
        $this->used();
    }

    // No usage of 'unused', as method call, in or out of the definition class.
    public function unused() {
        $this->used();
    }
}

class bar extends foo {
    public function some() {
        $this->used();
    }
}

$a = new foo();
$a->used();

?>
```

Specs

Short name	Classes/UsedMethods
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.203 Used Once Property

Property used once in their defining class.

Properties used in one method only may be used several times, and read only. This may be a class constant. Such properties are meant to be overwritten by an extending class, and that's possible with class constants.

Setting properties with default values is a good way to avoid littering the code with literal values, and provide a single point of update (by extension, or by hardcoding) for all those situations. A constant is definitely better suited for this task.

```
<?php

class foo {
    private $defaultCols = '*';
    cont DEFAULT_COLUMNS = '*';

    // $this->defaultCols holds a default value. Should be a constant.
    function bar($table, $cols) {
        // This is necessary to activate usage of default values
        if (empty($cols)) {
            $cols = $this->defaultCols;
        }
        $res = $this->query('SELECT '.$cols.' FROM '.$table);
        // ....
    }

    // Upgraded version of bar, with default values
    function bar2($table, $cols = self::DEFAULT_COLUMNS) {
        $res = $this->query('SELECT '.$cols.' FROM '.$table);
        // .....
    }
}

?>
```

Suggestions

- Remove the property, as it is probably not unused
- Add another usage of the property where it is useful

Specs

Short name	Classes/UsedOnceProperty
Rulesets	*Analyze*
Exakt since	0.10.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.204 Used Private Methods

List of all private methods that are used.

Protected methods, in a standalone class, are also included.

```php
<?php

class Foo {
    // Those methods are used
    private function method() {}
    private static function staticMethod() {}

    // Those methods are not used
    private function unusedMethod() {}
    private static function staticUnusedMethod() {}

    public function bar() {
        self::staticMethod();
        $this->method();
    }
}

?>
```

Specs

Short name	Classes/UsedPrivateMethod
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.205 Used Static Properties

List of all static properties that are used.

A private property is used when it is defined and read. A private property that is only written is not used. A property that is only read is used, as it may have a default value, or act as NULL.

```php
<?php

class foo {
    // This is a used property (see bar method)
    private $used = 1;

    function bar($a) {
        $this->used += $a;

        return $this->used;
    }
}

?>
```

Specs

Short name	Classes/UsedPrivateProperty
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.206 Used Protected Method

Marks methods being used in the current class or its children classes.

```
<?php

class foo {
    // This is reported
    protected usedByChildren() {}

    // This is not reported
    protected notUsedByChildren() {}
}

class bar extends foo {
    // The parent method is not overloaded, though it may be
    protected someMethod() {
        // The parent method is called
        $this->usedByChildren();
    }

}

?>
```

See also Visibility.

Specs

Short name	Classes/UsedProtectedMethod
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.207 Use Instanceof

The `instanceof` operator is a more precise alternative to `is_object()`. It is also faster.

instanceof checks for an variable to be of a class or its parents or the interfaces it implements. Once `instanceof` has been used, the actual attributes available (properties, constants, methods) are known, unlike with `is_object()`.

Last, `instanceof` may be upgraded to Typehint, by moving it to the method signature.

```
<?php

class Foo {

    // Don't use is_object
    public function bar($o) {
        if (!is_object($o)) { return false; } // Classic argument check
        return $o->method();
    }

    // use instanceof
    public function bar($o) {
        if ($o instanceof myClass) {  // Now, we know which methods are available
            return $o->method();
        }

        return false; } // Default behavior
    }

    // use of typehinting
    // in case $o is not of the right type, exception is raised automatically
    public function bar(myClass $o) {
        return $o->method();
    }
}

?>
```

`instanceof` and `is_object()` may not be always interchangeable. Consider using isset() on a known property for a simple check on objects. You may also consider is_string(), is_integer() or is_scalar(), in particular instead of !`is_object() <https://www.php.net/is_object>`_.

The `instanceof` operator is also faster than the `is_object()` functioncall.

See also Type Operators and is_object.

Suggestions

- Use instanceof and remove is_object()
- Create a high level interface to check a whole family of classes, instead of testing them individually
- Use typehint when possible
- Avoid mixing scalar types and objects in the same variable

Specs

Short name	Classes/UseInstanceof
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*TeamPass*, *Zencart*

13.2.208 Useless Abstract Class

> Those classes are marked 'abstract' and they are never extended. This way, they won't be instantiated nor used.

Abstract classes that have only static methods are omitted here : one usage of such classes are Utilities classes, which only offer static methods.

```
<?php

// Never extended class : this is useless
abstract class foo {}

// Extended class
abstract class bar {
    public function barbar() {}
}

class bar2 extends bar {}

// Utility class : omitted here
abstract class bar {
    public static function barbar() {}
}

?>
```

Suggestions

- Drop the abstract keyword
- Actually add an abstract keyword

Specs

Short name	Classes/UselessAbstract
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.209 Useless Constructor

Class constructor that have empty bodies are useless. They may be removed.

```
<?php

class X {
    function __construct() {
        // Do nothing
    }
}

class Y extends X {
    // Useful constructor, as it prevents usage of the parent
    function __construct() {
        // Do nothing
    }
}

?>
```

Specs

Short name	Classes/UselessConstructor
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.210 Useless Final

When a class is declared final, all of its methods are also final by default.

There is no need to declare them individually final.

```
<?php

    final class foo {
```

(continues on next page)

(continued from previous page)

```
        // Useless final, as the whole class is final
        final function method() { }
    }

    class bar {
        // Useful final, as the whole class is not final
        final function method() { }
    }

?>
```

See also Final Keyword, and When to declare final.

Specs

Short name	Classes/UselessFinal
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-useless-final

13.2.211 Useless Typehint

__get and __set magic methods won't use any typehint. The name of the magic property is always cast to string.

__call()

```
<?php

class x {
    // typehint is set and ignored
    function __set(float $name, string $value) {
        $this->$name = $value;
    }

    // typehint is set and ignored
    function __get(integer $name) {
        $this->$name = $value;
    }

    // typehint is checked by PHP 8.0 linting
    // typehint is enforced by PHP 7.x
    function __call(integer $name) {
        $this->$name = $value;
    }
}

$o = new x;
```

(continues on next page)

(continued from previous page)

```
$b = array();
// Property will be called 'Array'
$o->{$b} = 2;

// type of $m is check at calling time. It must be string.
$o->{$m}();

?>
```

See also __set.

Suggestions

- Use *string* for the *$name* parameter
- Use no typehint for the *$name* parameter

Specs

Short name	Classes/UselessTypehint
Rulesets	*ClassReview*, *Suggestions*
Exakt since	2.1.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.212 Use This

Those methods should be using $this, or a static method or property.

A method that doesn't use any local data may be considered for a move : may be this doesn't belong here.

The following functioncalls have been added, as access to the current class, without using *$this* or *self* : + get_class( ) + get_called_class( ) + get_object_vars( ) + get_parent_class( ) + get_class_vars( ) + get_class_methods( )

```
<?php

class dog {
    private $name = 'Rex';

    // This method is related to the current object and class
    public function attaboy() {
        return Fetch, $this->name, Fetch\n;
    }

    // Not using any class related data : Does this belong here?
    public function addition($a, $b) {
        return $a + $b;
    }
}
?>
```

See also The Basics.

Specs

Short name	Classes/UseThis
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.213 Using $this Outside A Class

`$this` is a special variable, that should only be used in a class context.

Until PHP 7.1, `$this` may be used as an argument in a function or a method, a global, a static : while this is legit, it sounds confusing enough to avoid it.

```
<?php

function foo($this) {
    echo $this;
}

// A closure can be bound to an object at later time. It is valid usage.
$closure = function ($x) {
    echo $this->foo($x);
}

?>
```

Starting with PHP 7.1, the PHP engine check thoroughly that `$this` is used in an appropriate manner, and raise fatal errors in case it isn't.

Yet, it is possible to find `$this` outside a class : if the file is included inside a class, then `$this` will be recognized and validated. If the file is included outside a class context, it will yield a fatal error : `Using `$this <https://www.php.net/manual/en/language.oop5.basic.php>`_ when not in object context.`

See also Closure::bind and The Basics.

Specs

Short name	Classes/UsingThisOutsideAClass
Rulesets	*Analyze*, *CompatibilityPHP71*, *LintButWontExec*
Exakt since	0.8.4
Php Version	7.0-
Severity	Critical
Time To Fix	Instant (5 mins)
Precision	High

13.2.214 Dynamically Called Classes

Indicates if a class is called dynamically.

```
<?php

// This class is called dynamically
class X {
    const CONSTANTE = 1;
}

$classe = 'X';

$x = new $classe();

echo $x::CONSTANTE;

?>
```

Specs

Short name	Classes/VariableClasses
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.215 Weak Typing

The test on a variable is not enough. The variable is checked for null, then used as an object or an array.

```
<?php

if ($a !== null) {
    echo $a->b;
}

?>
```

See also From assumptions to assertions.

Suggestions

- Use instanceof when checking for objects
- Use is_array() when checking for arrays. Also consider is_string(), is_int(), etc.
- Use typehint when the variable is an argument

Specs

Short name	Classes/WeakType
Rulesets	*Analyze*
Exakt since	1.2.8
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*TeamPass*

13.2.216 Wrong Class Name Case

The spotted classes are used with a different case than their definition. While PHP accepts this, it makes the code harder to read.

It may also be a violation of coding conventions.

```
<?php

// This use statement has wrong case for origin.
use Foo as X;

// Definition of the class
class foo {}

// Those instantiations have wrong case
new FOO();
new X();

?>
```

See also PHP class name constant case sensitivity and PSR-11.

Suggestions

- Match the defined class name with the called name

Specs

Short name	Classes/WrongCase
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*WordPress*

13.2.217 Illegal Name For Method

PHP has reserved usage of methods starting with __ for magic methods. It is recommended to avoid using this prefix, to prevent confusions.

```
<?php

class foo{
    // Constructor
    function __construct() {}

    // Constructor's typo
    function __constructor() {}

    // Illegal function name, even as private
    private function __bar() {}
}

?>
```

See also Magic Methods.

Suggestions

- Avoid method names starting with a double underscore : __
- Use method visibilities to ensure that methods are only available to the current class or its children

Specs

Short name	Classes/WrongName
Rulesets	*Analyze*
Exakt since	0.9.2
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
Examples	*PrestaShop*, *Magento*

13.2.218 Wrong Typed Property Default

Property is typed with an incompatible default value type.

Init type might be a new instance, the return of a method call or an interface compatible object.

```
<?php

class x {
    private A $property;
    private B $incompatible;

    function __construct() {
        // This is compatible
```

(continues on next page)

(continued from previous page)

```
        $this->property = new A();

        // This is incompatible : new B() expected
        $this->incompatible = new C();

    }
}

?>
```

PHP compiles such code, but won't execute it, as it detects the incompatibility.

Suggestions

- Remove the type hint of the property
- Fix the initialization call
- Use an interface for typehint

Specs

Short name	Classes/WrongTypedPropertyInit
Rulesets	*Analyze*, *CI-checks*, *ClassReview*, *LintButWontExec*
Exakt since	2.0.9
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.219 Create Compact Variables

This command creates Variable definitions, based on usage of 'compact'.

```
<?php

function foo() {
    $a = 1;
    return compact('a');
}
?>
```

This only works when compact() is used with literal values, or with constants. Dynamic values are not reported.

Suggestions

-

Specs

Short name	Complete/CreateCompactVariables
Rulesets	none
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.220 Create Default Values

This commands adds a link between variables, property definitions and any assignation to this container.

Variables have no definition expression in PHP. Exakat holds their definition with the *Variabledefinition* node.

Properties have definitions, and non-compulsory default values. This command creates multiple DEFINITION link for them.

DEFAULT is convenient in the case of *null* value, which will be assigned an object at execution time.

```
<?php

function foo() {
    // local Variabledefinition links to this expression
    $a = 1;
}

class x {
    // 1 is a default value
    private $p = 1;

    function __construct() {
        // 2 is also a default value for this.
        // This default value is different from the above as it is a part of an␣
↪assignation
        $this->p = 2;
    }
}

?>
```

Short assignations, such as += are not considered default value. It needs to be a full assignation

Specs

Short name	Complete/CreateDefaultValues
Rulesets	none
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.221 Complete/CreateForeachDefault

Suggestions

-

Specs

Short name	Complete/CreateForeachDefault
Rulesets	none
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.222 Create Magic Property

This command creates a link DEFINITION between a __get and __set calls, and its equivalent magic method.

```
<?php

class x {
    function foo() {
        // This is linked to __set
        $this->a = 1;

        // This is linked to __get
        return $this->b;
    }

    function __get($name) {
        return 1;
    }

    function __set($name, $value) {
        // Store the value
    }
}

?>
```

This command may not detect all possible link for the __get and __set call. It may be missing information about the nature of the object.

Specs

Short name	Complete/CreateMagicProperty
Rulesets	none
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.223 Extended Typehints

Produces all the definition links between typehints (arguments, return types, properties) and the definitions that are valid with the typehint.

```
<?php

function foo(A $A) {}

// This is the raw definition of the above typehint
interface A {}

// This is valid definition of the above typehint
class X implements A {}
// This is valid definition of the above typehint
class Y extends X {}

// This is not related to the typehint
class Z {}

?>
```

Suggestions

-

Specs

Short name	Complete/ExtendedTypehints
Rulesets	none
Exakt since	2.1.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.224 Follow Closure Definition

This command adds DEFINITION link between closure definitions and their usage.

Local usage of the closure, in the same scope, are detected. Relayed closure, when they are transmitted to another method for usage, is detected, for one level.

```
<?php

function foo() {
    $closure = function () {};
    // Local usage
    echo $closure();
}

function bar(Closure $x) {
    // relayed usage
    echo $x();
}


?>
```

Suggestions

-

Specs

Short name	Complete/FollowClosureDefinition
Rulesets	none
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.225 Make Class Constant Definition

This command adds DEFINITION link between class constant definitions and their usage.

```
<?php

class x {
    public const A = 1;
}

// Link to the constant definition
echo x::A;

// Cannot find the original class
echo $x::A;

?>
```

Suggestions

-

Specs

Short name	Complete/MakeClassConstantDefinition
Rulesets	none
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.226 Make Class Method Definition

This command links a method call to its method definition.

```
<?php

class x {
    function foo() {
        // This links to the bar() method
        return $this->bar();
    }

    function bar() {
        // This links to the link() method
        return $this->bar();
    }
}

?>
```

This command may not detect all possible link for the methods. It may be missing information about the nature of the object.

This command may also produce multiple definitions link, when the definition are ambiguous.

Specs

Short name	Complete/MakeClassMethodDefinition
Rulesets	none
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.227 Make Functioncall With Reference

Mark parameters as `isModified` if the functioncall uses reference.

This works on PHP native functions and custom functions.

This doesn't work on dynamic calls.

Suggestions

-

Specs

Short name	Complete/MakeFunctioncallWithReference
Rulesets	none
Exakt since	1.9.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.228 Overwritten Constant

This command adds OVERWRITE link between class constant definitions.

```
<?php

class x {
    protected const A = 1;
}

class y extends x {
    protected const A = 1;
}

?>
```

The *A* constant will be linked between classes x and y, with an OVERWRITE link.

Specs

Short name	Complete/OverwrittenConstants
Rulesets	none
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.229 Overwritten Methods

This command adds OVERWRITE link between methods definitions of classes.

```
<?php

class x {
    protected function foo() {}
}

class y extends x {
    protected function foo() {}
}

?>
```

The *foo* method will be linked between classes x and y, with an OVERWRITE link.

Specs

Short name	Complete/OverwrittenMethods
Rulesets	none
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.230 Overwritten Properties

This command adds OVERWRITE link between property definitions of classes.

```
<?php

class x {
    protected $p = 1;
}

class y extends x {
    protected $p = 1;
}

?>
```

The *$p* property will be linked between classes x and y, with an OVERWRITE link.

Specs

Short name	Complete/OverwrittenProperties
Rulesets	*CE*
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.231 Complete/PhpExtStubPropertyMethod

Provides isExt property to method call and properties access, based on typehints and local instantiation.

Specs

Short name	Complete/PhpExtStubPropertyMethod
Rulesets	none
Exakt since	2.1.9
Php Version	7.4-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.232 Complete/PhpNativeReference

Suggestions

-

Specs

Short name	Complete/PhpNativeReference
Rulesets	none
Exakt since	1.9.1
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.233 Propagate Calls

Update the graph, by linking a call to its definition. A call may be a function call, a closure call, a method call, a static methodcall.

Note that the definition is not always available, and the linking may fail. This is the case for PHP native functions, for dynamically build names, or omitted libraries.

Specs

Short name	Complete/PropagateCalls
Rulesets	none
Exakt since	1.9.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.234 Propagate Constants

This command calculates constant expression values, and set them in the graph.

```
<?php

const A = 1;
const B = A + 2;

?>
```

After running this command, B has `intval` of 3.

This command propagate `const` constants, class constants and define() constants, when possible.

Specs

Short name	Complete/PropagateConstants
Rulesets	none
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.235 SetA rray Class Definition

Link arrays() with their class / method definition.

PHP accepts an array structure such as `[class, method]`, or `[$object, method]` as a valid method callback. This analysis build such relations, whenever they are static.

```
<?php

class x {
    public function foo() {}
}

// designate the foo method in the x class
$f = [\x, 'foo'];
```

(continues on next page)

(continued from previous page)

```
array_

?>
```

Suggestions

-

Specs

Short name	Complete/SetArrayClassDefinition
Rulesets	none
Exakt since	1.9.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.236 Set Class_Alias Definition

Links `new` calls to the concrete class when class_alias() was used to create the name. The link is `DEFINITION`.

class_alias() are detected at loading time, and are used unconditionally.

This means that the fully qualified name of the `new` call and the instantiated class may be different : without the alias, the fully qualified name is the current fullcode, or its use's origin, while with class_alias(), it is an arbitrary name.

```
<?php

class x {
    public function foo() {}
}

class_alias('x', 'y');

//y exists, as an alias of x.
$y = new y;

?>
```

Suggestions

-

Specs

Short name	Complete/SetClassAliasDefinition
Rulesets	none
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.237 Set Class Method Remote Definition

Links method to the method definition. The link is `DEFINITION`.

Static method calls and normal method calls are both solved with this rule. Parent classes and trait are also searched for the right method.

```
<?php

class x {
    public function __construct() {}
    public function foo() {}
}

// This links to __construct method
$a = new x;

// This links to foo() method
$a->foo();

?>
```

Suggestions

-

Specs

Short name	Complete/SetClassMethodRemoteDefinition
Rulesets	none
Exakt since	1.9.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.238 Set Class Property Definition With Typehint

Links method call to its definition, thanks to property typehinting. The link is `DEFINITION`.

```php
<?php

class x {
    public x $p = null;

    public function bar() {
        return $this;
    }
}

$x = new x;

// $x->p is of 'x' class
$x->p->bar();

?>
```

Suggestions

-

Specs

Short name	Complete/SetClassPropertyDefinitionWithTypehint
Rulesets	none
Exakt since	1.9.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.239 Set Class Remote Definition With Global

Links method call to its definition, thanks to the global definition. The link is `DEFINITION`.

```php
<?php

class x {
    public function bar() {    }
}

global $a;
$a = new X;

function foo() {
    global $a;

    // This links to class x, method bar(), thanks to global.
    return $a->bar();
}

?>
```

Suggestions

-

Specs

Short name	Complete/SetClassRemoteDefinitionWithGlobal
Rulesets	none
Exakt since	1.9.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.240 Complete/SetClassRemoteDefinitionWithInjection

Suggestions

-

Specs

Short name	Complete/SetClassRemoteDefinitionWithInjection
Rulesets	none
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.241 Set Class Remote Definition With Local New

Links method calls and properties to its definition, thanks to the local new. The link is `DEFINITION`.

```
<?php

class x {
    public function bar() {     }
}

function foo() {
    $a = new x;

    // This links to class x, method bar(), thanks to the local new.
    return $a->bar();
}

?>
```

Suggestions

-

Specs

Short name	Complete/SetClassRemoteDefinitionWithLocalNew
Rulesets	none
Exakt since	1.9.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.242 Complete/SetClassRemoteDefinitionWithParenthesis

Links method call to its definition, thanks to the new in parenthesis. The link is `DEFINITION`.

```
<?php

class x {
    public function bar() {     }
}

function foo() {
    // This links to class x, method bar(), thanks to the new.
    return (new x)->bar();
}

?>
```

Suggestions

-

Specs

Short name	Complete/SetClassRemoteDefinitionWithParenthesis
Rulesets	none
Exakt since	1.9.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.243 Set Class Remote Definition With Return Typehint

Links method call to its definition, thanks to the typed return. The link is `DEFINITION`.

```php
<?php

class x {
    public function bar() {    }
}

function foo() {
    $a = bar();
    // This links to class x, method bar(), thanks to the new.
    return $a->bar();
}

function bar() : x {
    return new x;
}

?>
```

Suggestions

-

Specs

Short name	Complete/SetClassRemoteDefinitionWithReturnTypehint
Rulesets	none
Exakt since	1.9.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.244 Complete/SetClassRemoteDefinitionWithTypehint

Links method call to its definition, thanks to the typed argument. The link is `DEFINITION`.

```php
<?php

class x {
    public function bar() {    }
}

function foo(x $a) {
    // This links to class x, method bar(), thanks to the typehint.
    return $a->bar();
}

?>
```

Suggestions

-

Specs

Short name	Complete/SetClassRemoteDefinitionWithTypehint
Rulesets	none
Exakt since	1.9.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.245 Set Clone Link

This command creates a link DEFINITION between a clone call, and its equivalent magic method.

```
<?php

class x {
    // Store an object
    private $a;

    function foo() {
        // This clone is linked to the magic method below
        return clone $this;
    }

    function __clone() {
        $this->a = clone $this->a;
    }
}

// This is not linked to any __clone method, by lack of information
clone $x;

?>
```

This command may not detect all possible link for the clone. It may be missing information about the nature of the clone object.

Specs

Short name	Complete/SetCloneLink
Rulesets	none
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.246 Set Parent Definition

This command creates a DEFINITION link between *parent* keyword and the actual parent class.

```
<?php

class x {
    const A = 1;
}

class y extends x {
    function foo() {
        // 'parent' needs a DEFFINITION link to the class x
        return parent::A;
    }
}

?>
```

Specs

Short name	Complete/SetParentDefinition
Rulesets	*CE*
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.247 Set String Method Definition

Links a string with a static method call to its definition. The link is `DEFINITION`.

```
<?php
    class B {
        static public function C() {}
    }

    $a = 'B::C';
?>
```

Suggestions

-

Specs

Short name	Complete/SetStringMethodDefinition
Rulesets	none
Exakt since	1.9.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.248 Solve Trait Methods

This command adds DEFINITION link between trait's method definitions and their usage in classes.

```
<?php

trait t {
    function foo() {

    }
}

class x {
    use t { t::foo as foo2; };

    function bar() {
        // Link to foo() in trait t
        $this->foo();
        // Link to foo() in trait t, thanks to 'as'
        $this->foo2();
    }
}

?>
```

Suggestions

-

Specs

Short name	Complete/SolveTraitMethods
Rulesets	none
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.249 Composer's autoload

Is this code using the autoload from Composer.

Specs

Short name	Composer/Autoload
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.250 Is Composer Class

Mark a class as part of Composer's library.

Specs

Short name	Composer/IsComposerClass
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.251 Is Composer Interface

Mark interfaces as Composer interfaces.

Specs

Short name	Composer/IsComposerInterface
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.252 Composer Namespace

Mark this namespace as a Composer namespace.

When the namespace is found in the composer database, it is marked as such.

```
<?php

namespace Monolog;

use Monolog\Processor\WebProcessor;
use Monolog\Handler\TestHandler;

class MyLogger extends WebProcessor {
    /**/
}

?>
```

See also Packagist.

Specs

Short name	Composer/IsComposerNsname
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.253 Composer Usage

Mark the usage of composer, mostly by having a `composer.json` file.

Specs

Short name	Composer/UseComposer
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.254 Use Composer Lock

Reports if `composer.lock` was committed to the archive.

Specs

Short name	Composer/UseComposerLock
Rulesets	*CE*
Exakt since	0.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.255 Bad Constants Names

PHP's manual recommends that developer do not use constants with the convention __NAME__. Those are reserved for PHP future use.

For example, __TRAIT__ recently appeared in PHP, as a magic constant. In the future, other may appear.

```
<?php

const __MY_APP_CONST__ = 1;

const __MY_APP_CONST__ = 1;

define('__MY_OTHER_APP_CONST__', 2);

?>
```

The analyzer will report any constant which name is ___.*.___, or even _.*_ (only one underscore).

See also Constants.

Suggestions

- Avoid using names that doesn't comply with PHP's convention

Specs

Short name	Constants/BadConstantnames
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*PrestaShop*, *Zencart*

13.2.256 Case Insensitive Constants

PHP constants may be case insensitive, when defined with define() and the third argument.

This feature is deprecated since PHP 7.3 and will be removed in PHP 8.0.

```php
<?php

// case sensitive
define('A', 1);

// case insensitive
define('B', 1, true);

echo A;
// This is not possible
//echo a;

// both possible
echo B;
echo b;

?>
```

See also define.

Specs

Short name	Constants/CaseInsensitiveConstants
Rulesets	*CE*, *CompatibilityPHP73*
Exakt since	1.3.9
Php Version	All
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	High

13.2.257 Conditioned Constants

Indicates if a constant will be defined only if a condition is met.

```php
<?php

if (time() > 1519629617) {
    define('MY_CONST', false);
} else {
    define('MY_CONST', time() - 1519629617);
}

?>
```

Specs

Short name	Constants/ConditionedConstants
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.258 Constants Names

List of PHP constants being defined.

```
<?php

// with const
const X = 1;

// with define()
define ('Y', 2);

?>
```

Specs

Short name	Constants/Constantnames
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.259 Constants With Strange Names

List of constants being defined with names that are incompatible with PHP standards.

```
<?php

// Define a valid PHP constant
define('ABC', 1);
const ABCD = 2;

// Define an invalid PHP constant
define('ABC!', 1);
echo defined('ABC!') ? constant('ABC!') : 'Undefined';

// Const doesn't allow illegal names
```

(continues on next page)

(continued from previous page)

```
?>
```

See also PHP Constants.

Suggestions

- Rename constants to be valid constants
- Adopt a naming conversion scheme, to translate names from an incompatible source to PHP's standard (and back).

Specs

Short name	Constants/ConstantStrangeNames
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.260 Constants Usage

List of constants being used.

```
<?php

const MY_CONST = 'Hello';

// PHP_EOL (native PHP Constant)
// MY_CONST (custom constant)
echo PHP_EOL . MY_CONST;

?>
```

Specs

Short name	Constants/ConstantUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.261 Const Or Define Preference

`Const` and define() have almost the same functional use : they create constants.

The analyzed code has less than 10% of one of them : for consistency reasons, it is recommended to make constant definition consistent.

It is recommended to use `const` for global constants, as this keyword is processed at compile time, while define() is executed.

Note that define() used to allow the creation of case-insensitive constants, but this is deprecated since PHP 7.3 and will be removed in PHP 8.0.

```
<?php

    define('A1', 1);
    define('A2', 1);
    define('A3', 1);
    define('A4', 1);
    define('A5', 1);
    define('A6', 1);
    define('A7', 1);
    define('A8', 1);
    define('A9', 1);
    define('A10',1);

    const B = 3;

?>
```

See also Constant definition and Define.

Specs

Short name	Constants/ConstDefinePreference
Rulesets	none
Exakt since	1.3.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.262 Use const

The const keyword may be used to define constant, just like the define() function.

When defining a constant, it is recommended to use 'const' when the features of the constant are not dynamical (name or value are known at compile time). This way, constant will be defined at compile time, and not at execution time.

```
<?php
  //Do
  const A = 1;
  // Don't
  define('A', 1);
```

(continues on next page)

(continued from previous page)

```
?>
```

define() function is useful when the constant is not known at compile time, or when case sensitivity is necessary.

```
<?php
  // Read $a in database or config file
  define('A', $a);

  // Read $a in database or config file
  define('B', 1, true);
  echo b;
?>
```

See also Syntax.

Suggestions

- Use const instead of define()

Specs

Short name	Constants/ConstRecommended
Rulesets	*Analyze*, *CI-checks*, *Top10*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*phpMyAdmin*, *Piwigo*

13.2.263 Could Be Constant

Literals may be replaced by an existing constant.

Constants makes the code easier to read, as they may bear a meaningful name. They also hide implementation values, with a readable name, such as `const READABLE= true;`. Later, upgrading constant values is easier than scouring the code with a new literal.

Not all literal can be replaced by a constant values : sometimes, literal may have the same literal value, but different meanings. Check with your application semantics before changing any literal with a constant.

```
<?php

const A = 'abc';
define('B', 'ab');

class foo {
    const X = 'abcd';
}
```

(continues on next page)

(continued from previous page)

```
// Could be replaced by B;
$a = 'ab';

// Could be replaced by A;
$a = 'abc';

// Could be replaced by foo::X;
$a = 'abcd';

?>
```

This analysis currently doesn't support arrays.

This analysis also skips very common values, such as boolean, `0` and `1`. This prevents too many false positive.

Suggestions

- Turn the literal into an existing constant

Specs

Short name	Constants/CouldBeConstant
Rulesets	*Suggestions*
Exakt since	1.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.264 Constants Created Outside Its Namespace

Constants Created Outside Its Namespace.

Using the define() function, it is possible to create constant outside their namespace, but using the fully qualified namespace.

```
<?php

namespace A\B {
    // define A\B\C as 1
    define('C', 1);
}

namespace D\E {
    // define A\B\C as 1, while outside the A\B namespace
    define('A\B\C', 1);
}

?>
```

However, this makes the code confusing and difficult to debug. It is recommended to move the constant definition to its namespace.

Specs

Short name	Constants/CreatedOutsideItsNamespace
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.265 Custom Constant Usage

Using constants that were not defined in PHP extensions or PHP itself.

```
<?php

// display MY_CONSTANT : MY_CONSTANT is a user constant.
echo MY_CONSTANT;

// display PHP version : PHP_VERSION is a native PHP constant.
echo PHP_VERSION;

// MY_CONSTANT definition.
const MY_CONSTANT;

?>
```

See also PHP Constants.

Specs

Short name	Constants/CustomConstantUsage
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.266 Constant Case Preference

Define() creates constants which are case sensitive or not.

The analyzed code has less than 10% of one of them : for consistency reasons, it is recommended to make constant sentivity definition consistent.

Note that define() used to allow the creation of case-sensitive constants, but this is deprecated since PHP 7.3 and will be removed in PHP 8.0.

```
<?php

    define('A1', 1);
    define('A2', 1);
    define('A3', 1);
    define('A4', 1);
    define('A5', 1);
    define('A6', 1);
    define('A7', 1);
    define('A8', 1);
    define('A9', 1);
    define('A10',1);

    define('A10',1, true);

?>
```

See also Constant definition.

Specs

Short name	Constants/DefineInsensitivePreference
Rulesets	none
Exakt since	1.3.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.267 Constant Dynamic Creation

> Registering constant with dynamic values. Dynamic values include values read in external sources (files, databases, remote API, ...), random sources (time, rand(), ...)

Dynamic constants are not possible with the `const` keyword, though static constant expression allows for a good range of combinations, including conditions.

```
<?php

$a = range(0, 4);
foreach($array as $i) {
    define(A$i, $i);
    define(N$i, true);
}

define(C, 5);

?>
```

Specs

Short name	Constants/DynamicCreation
Rulesets	*CE*
Exakt since	1.6.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.268 True False Inconsistant Case

TRUE or true or True is the favorite.

Usually, PHP projects choose between ALL CAPS True/False, or all lowercase True/False. Sometimes, the project will have no recommendations.

When your project use a vast majority of one of the convention, then the analyzer will report all remaining inconsistently cased constant.

```
<?php

$a1 = true;
$a2 = true;
$a3 = true;
$a4 = true;
$a5 = true;
$a6 = true;
$a7 = true;
$a8 = true;
$a9 = true;
$a10 = true;

// This convention is inconsistence with the rest
$b1 = TRUE;
?>
```

Specs

Short name	Constants/InconsistantCase
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.269 Invalid Constant Name

There is a naming convention for PHP constants names.

According to PHP's manual, constant names, ' A valid constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.'.

Constant, must follow this regex : `/[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*/`.

In particular when defined using define() function, no error is produced. When using `const`, on the other hand, the

```
<?php

define('+3', 1); // wrong constant!

echo constant('+3'); // invalid constant access

?>
```

See also Constants.

Suggestions

- Change constant name

Specs

Short name	Constants/InvalidName
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*OpenEMR*

13.2.270 Is An Extension Constant

Mark a constant if it belongs to a known extension.

```
<?php

// JSON_HEX_AMP is a constant from ext/json
echo json_encode($object, JSON_HEX_AMP);

// JSON_HEX_AMP is a constant from ext/json
echo json_encode($object, JSON_HOAX_AMP);

?>
```

See also Supported PHP Extensions.

Specs

Short name	Constants/IsExtConstant
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.271 Is Global Constant

Mark a constant that may fallback to a global const definition, even though it is in a namespace.

This analysis skips PHP and ext's functions, namespaced constants.

```
<?php

namespace X {

    const PHP_VERSION = 1;

    // Local constant
    echo PHP_VERSION;

    // This constant fallsback to \E_ALL, unless DNS_NS is defined in this namespace
    echo E_ALL;

    // This constant is always \DNS_NS
    echo \DNS_NS;

    // This is a Notice
    echo UNDEFINED_CONSTANT;
}

?>
```

See also $GLOBALS and Variable scope.

Specs

Short name	Constants/IsGlobalConstant
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.272 Is PHP Constant

Mark a constant if it is a PHP constant.

```
<?php

// This is an PHP constant
$a = HTML_ENTITIES;

// This is an PHP function
$a = CMS_ORDER;

?>
```

Specs

Short name	Constants/IsPhpConstant
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.273 Magic Constant Usage

There are eight magical constants that change depending on where they are used. For example, the value of __LINE__ depends on the line that it's used on in your script. These special constants are case-insensitive.

- __LINE__
- __FILE__
- __DIR__
- __FUNCTION__
- __CLASS__
- __TRAIT__
- __METHOD__
- __NAMESPACE__

```
<?php

echo 'This code is in file '__FILE__.', line '.__LINE__;

?>
```

See also Magic Constants.

Specs

Short name	Constants/MagicConstantUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.274 Multiple Constant Definition

Some constants are defined several times in your code. This will lead to a fatal error, if they are defined during the same execution.

Multiple definitions may happens at bootstrap, when the application code is collecting information about the current environment. It may also happen at inclusion time, which one set of constant being loaded, while other definition are not, avoiding conflict. Both are false positive.

```
<?php

// OS is defined twice.
if (PHP_OS == 'Windows') {
    define('OS', 'Win');
} else {
    define('OS', 'Other');
}

?>
```

Suggestions

- Move the constants to a class, and include the right class based on control flow.
- Give different names to the constants, and keep the condition close to utilisation.
- Move the constants to an external configuration file : it will be easier to identify that those constants may change.

Specs

Short name	Constants/MultipleConstantDefinition
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Dolibarr*, *OpenConf*

13.2.275 PHP Constant Usage

List of PHP constants being used.

```
<?php

const MY_CONST = 'Hello';

// PHP_EOL (native PHP Constant)
// MY_CONST (custom constant, not reported)
echo PHP_EOL . MY_CONST;

?>
```

See also Predefined Constants.

Specs

Short name	Constants/PhpConstantUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.276 Strange Name For Constants

Those constants looks like a typo from other names.

```
<?php

// This code looks OK : DIRECTORY_SEPARATOR is a native PHP constant
$path = $path . DIRECTORY_SEPARATOR . $file;

// Strange name DIRECOTRY_SEPARATOR
$path = $path . DIRECOTRY_SEPARATOR . $file;

?>
```

Suggestions

- Fix any typo in the spelling of the constants
- Tell us about common misspelling so we can upgrade this analysis

Specs

Short name	Constants/StrangeName
Rulesets	*Analyze*
Exakt since	0.10.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.277 Undefined Constants

Constants definition can't be located.

Those constants are not defined in the code, and will raise errors, or use the fallback mechanism of being treated like a string.

```
<?php

const A = 1;
define('B', 2);

// here, C is not defined in the code and is reported
echo A.B.C;

?>
```

It is recommended to define them all, or to avoid using them.

See also Constants.

Suggestions

- Define the constant
- Fix the name of the constant
- Fix the namespace of the constant (FQN or use)
- Remove the usage of the constant

Specs

Short name	Constants/UndefinedConstants
Rulesets	*Analyze*, *CI-checks*, *CompatibilityPHP72*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.278 Unused Constants

Those constants are defined in the code but never used. Defining unused constants slow down the application, as they are executed and stored in PHP hashtables.

```
<?php

// const-defined constant
const USED_CONSTANT   = 0;
const UNUSED_CONSTANT = 1 + USED_CONSTANT;

// define-defined constant
define('ANOTHER_UNUSED_CONSTANT', 3);

?>
```

It is recommended to comment them out, and only define them when it is necessary.

Suggestions

- Make use of the constant
- Remove the constant

Specs

Short name	Constants/UnusedConstants
Rulesets	*Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.279 Variable Constants

Variable constants are actually constants whose value is accessed via the function constant(). Otherwise, there is no way to dynamically access a constant (aka, when the developer has the name of the constant as a incoming parameter, and it requires the value of it).

```
<?php

const A = 'constant_value';

$constant_name = 'A';

$variableConstant = constant($constant_name);

?>
```

See also constant().

Specs

Short name	Constants/VariableConstant
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.280 Call Order

This is a representation of the code. Each node is a function or method, and each link a is call from a method to another.

The only link is the possible call from a method to the other. All control flow is omitted, including conditional calls and loops.

```
<?php

    function foo() {
        bar();
        foobar();
    }

    function bar() {
        foobar();
    }

    function foobar() {

    }
?>
```

From the above script, the resulting network will display 'foo() -> bar(), foo() -> foobar(), bar() -> foobar()' calls.

Suggestions

-

Specs

Short name	Dump/CallOrder
Rulesets	*CE*
Exakt since	2.1.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.281 Collect Atom Counts

Specs

Short name	Dump/CollectAtomCounts
Rulesets	*CE*
Exakt since	2.1.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.282 Collect Block Size

Collect block size for for, foreach, while, do…while, ifthen.

Suggestions

-

Specs

Short name	Dump/CollectBlockSize
Rulesets	none
Exakt since	2.2.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.283 Dump/CollectClassChanges

Suggestions

-

Specs

Short name	Dump/CollectClassChanges
Rulesets	*CE*
Exakt since	2.1.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.284 Collect Class Children Count

Count the number of class children for each class.

```
<?php

// 2 children
class a {}

// 1 children
class b extends a {}

// no children
class c extends b {}

// no children
class d extends a {}
?>
```

Specs

Short name	Dump/CollectClassChildren
Rulesets	*CE*
Exakt since	2.0.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.285 Collect Class Constant Counts

This analysis collects the number of class constants per class or interface.

The count applies to classes, anonymous classes and interfaces. They are considered distinct one from another.

```
<?php

class foo {
    // 3 constant
    const A =1, B =2;
}

interface bar {
    // 3 properties
    const A=1, B=2, C=3;
}

?>
```

Suggestions

-

Specs

Short name	Dump/CollectClassConstantCounts
Rulesets	*CE*
Exakt since	2.1.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.286 Collect Class Depth

Count the number of level of extends for classes.

```
<?php

class a {}

class b extends a {}

class c extends b {}

class d extends a {}
?>
```

Specs

Short name	Dump/CollectClassDepth
Rulesets	*CE*
Exakt since	2.0.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.287 Collect Classes Dependencies

Collect Classes Dependencies

Specs

Short name	Dump/CollectClassesDependencies
Rulesets	*CE*
Exakt since	2.1.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.288 Collect Class Interface Counts

Collect the number of interfaces implemented per class.

```
<?php

// This class implements 3 interfaces
class x implements i, j, k {
    // Some code
}

?>
```

Suggestions

-

Specs

Short name	Dump/CollectClassInterfaceCounts
Rulesets	*CE*
Exakt since	2.0.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.289 Collect Class Traits Counts

Counts the number of trait used in a class.

```
<?php

// Use no traits
class x {}

// Use one trait
class y {
    use TraitT;
}

?>
```

Suggestions

-

Specs

Short name	Dump/CollectClassTraitsCounts
Rulesets	*CE*
Exakt since	2.1.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.290 Dump/CollectDefinitionsStats

Suggestions

-

Specs

Short name	Dump/CollectDefinitionsStats
Rulesets	*CE*
Exakt since	2.1.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.291 Collect Files Dependencies

Collect all dependencies between files, based on definitions and usage.

For example, file *A.php*, which defines de class *A*, is a dependence to a file *B.php*, which makes a call to a method from *A*, or use *A* as a typehint, etc..

Specs

Short name	Dump/CollectFilesDependencies
Rulesets	*CE*
Exakt since	2.1.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.292 Foreach() Favorite

Collect the name used in foreach() loops. Then, sorts them in order of popularity.

Specs

Short name	Dump/CollectForeachFavorite
Rulesets	*CE*
Exakt since	1.9.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.293 Dump/CollectGlobalVariables

Suggestions

-

Specs

Short name	Dump/CollectGlobalVariables
Rulesets	*CE*
Exakt since	2.1.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.294 Collect Literals

Collects all literals in the application, for inventory purposes.

Specs

Short name	Dump/CollectLiterals
Rulesets	*CE*
Exakt since	1.9.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.295 Collect Local Variable Counts

This analysis collects the number of local variables used in a method or a function.

The count applies to functions, methods, closures and arrow functions.

Arguments and global variables are not counted. Static variables are.

```php
<?php

function foo($arg) {
    global $w;

    // This is a local variable
    $x = rand(1, 2);

    return $x + $arg + $w;
}

?>
```

Suggestions

-

Specs

Short name	Dump/CollectLocalVariableCounts
Rulesets	*CE*
Exakt since	2.1.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.296 Collect Mbstring Encodings

This analysis collects the encoding names, used by ext/mb functions.

```php
<?php

mb_stotolower('PHP', 'iso-8859-1');

mb_stotolower('PHP', 'iso-8859-1');

?>
```

Suggestions

-

Specs

Short name	Dump/CollectMbstringEncodings
Rulesets	*CE*
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.297 Collect Method Counts

This analysis collects the number of methods per class, trait or interface.

The count applies to classes, anonymous classes, traits and interfaces. They are considered distinct one from another.

```
<?php

class foo {
    // 2 methods
    function __construct() {}
    function foo() {}
}

interface bar {
    // 1 method
    function a() ;
}

class barbar {
    // 3 methods
    function __construct() {}
    function foo() {}
    function a() {}
}

?>
```

Suggestions

-

Specs

Short name	Dump/CollectMethodCounts
Rulesets	*CE*
Exakt since	2.1.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.298 Collect Native Calls Per Expressions

Computes the number of PHP native call per expression.

Suggestions

-

Specs

Short name	Dump/CollectNativeCallsPerExpressions
Rulesets	*CE*
Exakt since	2.1.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.299 Collect Parameter Counts

This analysis collects the number of parameter per method.

The count applies to functions, methods, closures and arrow functions.

```
<?php

// parameter count on function : 1
function foo($a) { }

// parameter count on closure : 2
function ($b, $c = 2) {}

// parameter count on method : 0 (none)
class x {
    function moo() { }
}
?>
```

Suggestions

-

Specs

Short name	Dump/CollectParameterCounts
Rulesets	*CE*
Exakt since	1.9.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.300 Collect Parameter Names

This analysis collects the names of all parameters. It also counts the number of occurrences of each name.

The names are collected from functions, methods, closures and arrow functions. Compulsory and optional parameters are all processed.

```
<?php

// parameter $a
function foo($a) { }

// parameter $b, $c
function ($b, $c = 2) {}

// parameters in interfaces are counted too.
// Here, $a will be counted with the one above.
interfaces x {
    function moo($a);
}
?>
```

Specs

Short name	Dump/CollectParameterNames
Rulesets	*CE*
Exakt since	2.1.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.301 Collect Php Structures

Collect Php Structures

Specs

Short name	Dump/CollectPhpStructures
Rulesets	*CE*
Exakt since	2.1.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.302 Collect Property Counts

This analysis collects the number of properties per class or trait.

The count applies to classes, anonymous classes and traits. They are considered distinct one from another.

Properties may be `static` or not. Visibility, default values and typehints are omitted.

```
<?php

class foo {
    // 3 properties
    private $p1, $p2, $p3;
}

trait foo {
    // 3 properties
    protected $p1;
    public $p2 = 1, $p3;
}

?>
```

Suggestions

-

Specs

Short name	Dump/CollectPropertyCounts
Rulesets	*CE*
Exakt since	2.1.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.303 Collect Readability

Measure readability for methods, functions and closures, then report them.

Specs

Short name	Dump/CollectReadability
Rulesets	*CE*
Exakt since	2.1.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.304 Collect Use Counts

Count the number of use expression in a file. This count 4 uses.

```
<?php

use A as B;
use F\C, F\D, F\E;

?>
```

Specs

Short name	Dump/CollectUseCounts
Rulesets	*CE*
Exakt since	2.1.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.305 Collect Variables

Collect all variables from the code. Their type is mentionned, as variable, object or array, depending on their usage.

Suggestions

-

Specs

Short name	Dump/CollectVariables
Rulesets	*CE*
Exakt since	2.1.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.306 Constant Order

Order of dependency of constants.

Constants, either global or class, may be built using static expression. In turn, this means that constants have now a build order. For example :

```
<?php

// A is an independant global constant
const A = 1;
// B is an dependant global constant : it is built with A
const B = A + 1;

class x {
    // x::C is an dependant class constant : it is built with A
    const C = A + 3;
}

?>
```

The code above leads to the following order : `A` - `B`, `C`. `A` can be built without constraints, while `B` and `C` must be build when `A` is available. Note that `B` and `C` are both dependant on `A`, but are not dependant on each other.

The resulting tree displays the different relationship between the constants.

Note : `define``constants are not considered here. Only ``const` constants, global or class.

Specs

Short name	Dump/ConstantOrder
Rulesets	*CE*
Exakt since	2.0.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.307 Cyclomatic Complexity

Calculate cyclomatic complexity for each methods, function, and closures.

Suggestions

-

Specs

Short name	Dump/CyclomaticComplexity
Rulesets	*CE*
Exakt since	1.9.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.308 Dump/DereferencingLevels

Suggestions

-

Specs

Short name	Dump/DereferencingLevels
Rulesets	*CE*
Exakt since	1.9.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.309 Environment Variable Usage

Collects all environment variables in the application, for inventory purposes.

```
<?php

$implicit_global = 1;
global $explicit_global;

function foo() {
    $local_variable = 2;
}

?>
```

See also Variable scope.

Specs

Short name	Dump/EnvironnementVariables
Rulesets	*CE*
Exakt since	1.9.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.310 Dump/FossilizedMethods

Suggestions

-

Specs

Short name	Dump/FossilizedMethods
Rulesets	*CE*
Exakt since	2.1.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.311 Dump/Inclusions

Suggestions

-

Specs

Short name	Dump/Inclusions
Rulesets	*CE*
Exakt since	2.0.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.312 Indentation Levels

Collect all level of nesting for methods and functions.

Suggestions

-

Specs

Short name	Dump/IndentationLevels
Rulesets	*CE*
Exakt since	1.9.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.313 New Order

Order in which new calls are done.

```
<?php

class x {}

// class Y has precedence over class X, as it needs to be called first to get to X
class y {
    function foo() {
        return new x();
    }
}


?>
```

Specs

Short name	Dump/NewOrder
Rulesets	*CE*
Exakt since	2.0.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.314 Links Between Parameter And Argument

Collect various stats about arguments and parameter usage.

A parameter is one slot in the method definition. An argument is a slot in the method call. Both are linked by the method and their respective position in the argument list.

- Total number of argument usage, linked to a parameter : this excludes arguments from external libraries and native PHP functions. For reference.
- Number of identical parameter : cases where argument and parameter have the same name.
- Number of different parameter : cases where argument and parameter have the different name.
- Number of expression argument : cases where argument is an expression
- Number of constant argument : cases where the argument is a constant

```
<?php

function foo($a, $b) {
    // some code
}

// $a is the same as the parameter
// $c is different from the paramter $b
foo($a, $c);

const C = 1;

// Foo is called with a constant (1rst argument)
// Foo is called with a expression (2nd argument)
foo(C, 1+3);

?>
```

Specs

Short name	Dump/ParameterArgumentsLinks
Rulesets	*CE*
Exakt since	2.0.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.315 Typehinting Stats

This module collects statistics about typehinting usage.

Suggestions

-

Specs

Short name	Dump/TypehintingStats
Rulesets	*CE*
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.316 Typehint Order

Topological order, based on typehints.

Each function, method that use typehint is a link between a type of data and another one. The argument typehint acts as a filter, and the returned type hint is the next step.

```
<?php

// This library imposes the following order : A -> B -> C
function foo(A $a) : B { }
function bar(B $b) : C { }

?>
```

Specs

Short name	Dump/Typehintorder
Rulesets	*CE*
Exakt since	2.0.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.317 Exception Order

When catching exception, the most specialized exceptions must be in the early catch, and the most general exceptions must be in the later catch. Otherwise, the general catches intercept the exception, and the more specialized will not be read.

```
<?php

class A extends \Exception {}
class B extends A {}

try {
    throw new A();
}
catch(A $a1) { }
```

(continues on next page)

(continued from previous page)

```
catch(B $b2 ) {
    // Never reached, as previous Catch is catching the early worm
}

?>
```

Suggestions

- Remove one of the catch clause

Specs

Short name	Exceptions/AlreadyCaught
Rulesets	*Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Woocommerce*

13.2.318 Can't Throw Throwable

Classes extending `Throwable` can't be thrown. The same applies to interfaces.

Although this code lints, PHP throws a Fatal error when executing or including it : `Class fooThrowable cannot implement interface `Throwable <https://www.php.net/manual/en/class.throwable.php>`_, extend Exception or Error instead.`

```
<?php

// This is the way to go
class fooException extends \Exception { }

// This is not possible and a lot of work
class fooThrowable implements \throwable { }

?>
```

See also Throwable, Exception and Error.

Suggestions

- Extends the Exception class
- Extends the Error class

Specs

Short name	Exceptions/CantThrow
Rulesets	*Analyze*, *LintButWontExec*
Exakt since	1.3.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.319 Caught Variable

Suggestions

- Make all caught constant consistent, and avoid using them for something else

Specs

Short name	Exceptions/CatchE
Rulesets	none
Exakt since	1.7.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.320 Catch Undefined Variable

Always initialize variable before the try block, when they are used in a catch block. If the exception is raised before the variable is defined, the catch block may have to handle an undefined variable, leading to more chaos.

```
<?php

$a = 1;
try {
    mayThrowAnException();
    $b = 2;
} catch (\Exception $e) {
    // $a is already defined, as it was done before the try block
    // $b may not be defined, as it was initialized after the exception-throwing␣
↪expression
    echo $a + $b;
}

?>
```

Suggestions

- Always define the variable used in the catch clause, before the try block.

Specs

Short name	Exceptions/CatchUndefinedVariable
Rulesets	*Analyze*
Exakt since	2.1.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.321 Undefined Caught Exceptions

Those are exceptions that are caught in the code, but are not defined in the application.

They may be externally defined, such as in core PHP, extensions or libraries. Make sure those exceptions are useful to your application : otherwise, they are dead code.

```
<?php

try {
    library_function($some, $args);

} catch (LibraryException $e) {
    // This exception is not defined, and probably belongs to Library
    print Library failed\n;

} catch (OtherLibraryException $e) {
    // This exception is not defined, and probably do not belongs to this code
    print Library failed\n;

} catch (\Exception $e) {
    // This exception is a PHP standard exception
    print Something went wrong, but not at Libary level\n;
}

?>
```

Suggestions

- Remove the catch clause, as it is dead code
- Make sure the exception is thrown by the underlying code

Specs

Short name	Exceptions/CaughtButNotThrown
Rulesets	*Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.322 Caught Exceptions

Exceptions used in catch clause.

```
<?php

try {
    foo();
} catch (MyException $e) {
    fixException();
} finally {
    clean();
}

?>
```

See also Exceptions.

Specs

Short name	Exceptions/CaughtExceptions
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.323 Could Use Try

Some commands may raise exceptions. It is recommended to use the try/catch block to intercept those exceptions, and process them.

- / : `DivisionByZeroError`
- % : `DivisionByZeroError`
- intdiv() : `DivisionByZeroError`
- << : `ArithmeticError`
- >> : `ArithmeticError`

- `Phar\:\:mungserver : PharException`
- `Phar\:\:webphar : PharException`

See also Predefined Exceptions, PharException.

Suggestions

- Add a try/catch clause around those commands
- Add a check on the values used with those operator : for example, check a dividend is not 0, or a bitshift is not negative

Specs

Short name	Exceptions/CouldUseTry
Rulesets	*Suggestions*
Exakt since	1.5.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Mautic*

13.2.324 Defined Exceptions

This is the list of defined exceptions.

```
<?php

class myException extends \Exception {}

// A defined exception
throw new myException();

// not a defined exception : it is already defined.
throw new \RuntimeException();

?>
```

See also Exceptions.

Specs

Short name	Exceptions/DefinedExceptions
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.325 Forgotten Thrown

An exception is instantiated, but not thrown.

```
<?php

class MyException extends \Exception { }

if ($error !== false) {
    // This looks like 'throw' was omitted
    new MyException();
}

?>
```

Suggestions

- Remove the throw expression
- Add the new to the throw expression

Specs

Short name	Exceptions/ForgottenThrown
Rulesets	*Analyze*
Exakt since	0.10.2
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.326 PHP Exception

Mark an exception as a native exception. They may come from PHP standard distribution or an extension.

```
<?php

// From the native set
$a = new LogicException('Logic error');
throw $a;

// From an extension
throw new ZookeeperException('Zookeeper error');

?>
```

See also Exceptions.

Specs

Short name	Exceptions/IsPhpException
Rulesets	none
Exakt since	1.5.2
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.327 Large Try Block

Try block should enclosing only the expression that may emit an exception.

When writing large blocks of code in a try, it becomes difficult to understand where the expression is coming from. Large blocks may also lead to catch multiples exceptions, with a long list of catch clause.

In particular, the catch clause will resume the execution without knowing where the try was interrupted : there are no indication of achievement, even partial. In fact, catching an exception signals a very dirty situation.

```
<?php

// try is one expression only
try {
    $database->query($query);
} catch (DatabaseException $e) {
    // process exception
}

// Too many expressions around the one that may actually emit the exception
try {
    $SQL = build_query($arguments);
    $database = new Database($dsn);
    $database->setOption($options);
    $statement = $database->prepareQuery($SQL);
    $result = $statement->query($query);
} catch (DatabaseException $e) {
    // process exception
}

?>
```

This analysis reports try blocks that are 5 lines or more. This threshold may be configured with the directive `tryBlockMaxSize`. Catch clause, and finally are not considered here.

Suggestions

- Reduce the amount of code in the block, by moving it before and after

Name	Default	Type	Description
tryBlockMaxSize	5	integer	Maximal number of expressions in the try block.

Specs

Short name	Exceptions/LargeTryBlock
Rulesets	*Suggestions*
Exakt since	2.1.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.328 Long Preparation For Throw

When throwing an exception, move the preparing code in the exception. This will keep the `throw` call simple.

```
<?php

// Examples extracted from Alain Schlesser's blog
public function render( $view ): string {

   if ( ! $this->views->has( $view ) ) {
      switch ( gettype( $view ) ) {
         case 'object':
            $view = get_class( $view );
         case 'string':
            $message = sprintf(
               'The requested View %s does not exist.',
               $view
            );
            break;
         default:
            $message = sprintf(
               'An unknown View type of %s was requested.',
               $view
            );
      }

      throw new ViewWasNotFound( $message );
   }

   echo $this->views->get( $view )
             ->render();
}

?>
```

See also Structuring PHP Exceptions session and Best practices for handling exceptional behavior.

Suggestions

- Move the preparation into the Exception to keep the throw simple

Name	Default	Type	Description
preparationLineCount	8	integer	Minimal number of lines before the throw.

Specs

Short name	Exceptions/LongPreparation
Rulesets	*Suggestions*
Exakt since	2.2.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Unknown

13.2.329 Multiple Exceptions Catch()

It is possible to have several distinct exceptions class caught by the same catch, preventing code repetition.

This is a new feature since PHP 7.1.

```
<?php

// PHP 7.1 and more recent
try {
    throw new someException();
} catch (Single $s) {
    doSomething();
} catch (oneType | anotherType $s) {
    processIdentically();
} finally {

}

// PHP 7.0 and older
try {
    throw new someException();
} catch (Single $s) {
    doSomething();
} catch (oneType $s) {
    processIdentically();
} catch (anotherType $s) {
    processIdentically();
} finally {

}

?>
```

This is a backward incompatible feature of PHP 7.1.

Specs

Short name	Exceptions/MultipleCatch
Rulesets	*CE*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.330 Overwritten Exceptions

In catch blocks, it is good practice to avoid overwriting the incoming exception, as information about the exception will be lost.

```
<?php

try {
    doSomething();
} catch (SomeException $e) {
    // $e is overwritten
    $e = new anotherException($e->getMessage());
    throw $e;
} catch (SomeOtherException $e) {
    // $e is chained with the next exception
    $e = new Exception($e->getMessage(), 0, $e);
    throw $e;
}

?>
```

Suggestions

- Use another variable name to create new values inside the catch
- Use anonymous catch clause (no variable caught) in PHP 8.0, to make this explicit

Specs

Short name	Exceptions/OverwriteException
Rulesets	*Analyze*, *CI-checks*, *Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.331 Rethrown Exceptions

Throwing a caught exception is usually useless and dead code.

When exceptions are caught, they should be processed or transformed, but not rethrown as is.

Those issues often happen when a catch structure was positioned for debug purposes, but lost its usage later.

```
<?php

try {
    doSomething();
} catch (Exception $e) {
    throw $e;
}

?>
```

See also What are the best practices for catching and re-throwing exceptions? and Exception chaining.

Suggestions

- Log the message of the exception for later usage.
- Remove the try/catch and let the rest of the application handle this exception.
- Chain the exception, by throwing a new exception, including the caught exception.

Specs

Short name	Exceptions/Rethrown
Rulesets	*Dead code*
Exakt since	0.9.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*PrestaShop*

13.2.332 Throw Functioncall

The `throw` keyword expects to use an exception. Calling a function to prepare that exception before throwing it is possible, but forgetting the new keyword is also possible.

```
<?php

// Forgotten new
throw \RuntimeException('error!');

// Code is OK, function returns an exception
throw getException(ERROR_TYPE, 'error!');

function getException(ERROR_TYPE, $message) {
```

(continues on next page)

(continued from previous page)

```
    return new \RuntimeException($messsage);
}

?>
```

When the `new` keyword is forgotten, then the class constructor is used as a function name, and now exception is emitted, but an `Undefined function` fatal error is emitted.

See also Exceptions.

Suggestions

- Add the new operator to the call
- Make sure the function is really a functioncall, not a class name
- Use return typehints for functions, so that Exception may be detected

Specs

Short name	Exceptions/ThrowFunctioncall
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	Medium
Examples	*SugarCrm*, *Zurmo*

13.2.333 Thrown Exceptions

Usage of throw keyword.

```
<?php

throw new MyException('Error happened');

?>
```

See also Exceptions.

Specs

Short name	Exceptions/ThrownExceptions
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.334 Uncaught Exceptions

The following exceptions are thrown in the code, but are never caught.

```
<?php

// This exception is throw, but not caught. It will lead to a fatal error.
if ($message = check_for_error()) {
    throw new My\Exception($message);
}

// This exception is throw, and caught.
try {
    if ($message = check_for_error()) {
        throw new My\Exception($message);
    }
} catch (\Exception $e) {
    doSomething();
}

?>
```

Either they will lead to a Fatal Error, or they have to be caught by an including application. This is a valid behavior for libraries, but is not for a final application.

See also Structuring PHP Exceptions.

Suggestions

- Catch all the exceptions you throw

Specs

Short name	Exceptions/UncaughtExceptions
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.335 Unthrown Exception

These are exceptions that are defined in the code but never thrown.

```
<?php

//This exception is defined but never used in the code.
class myUnusedException extends \Exception {}

//This exception is defined and used in the code.
class myUsedException extends \Exception {}
```

(continues on next page)

(continued from previous page)

```
throw new myUsedException('I was called');

?>
```

See also Exceptions.

Specs

Short name	Exceptions/Unthrown
Rulesets	*Analyze*, *Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-unthrown-exceptions

13.2.336 Unused Exception Variable

The variable from a catch clause is not used. It is expected to be used, either by chaining the exception, or logging the message.

In PHP 8.0, this variable may be omitted.

```
<?php

try{
    doSomething();
} catch (A $a) {
    // $a is caught, but not used here
} catch (B $b) {
    // $b is caught, and used
    log($b->getMessage());
} catch (C) {
    // Caught and ignored (PHP 8.0 +)
}

?>
```

Suggestions

- Drop the variable in the clause expression (PHP 8.0 and more recent)
- Chain the exception
- Log the exception message

Specs

Short name	Exceptions/UnusedExceptionVariable
Rulesets	*Suggestions*
Exakt since	2.2.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.337 Useless Catch

Catch clause should handle the exception with some work.

Among the task of a catch clause : log the exception, clean any mess that was introduced, fail graciously.

```
<?php

function foo($a) {
    try {
        $b = doSomething($a);
    } catch (Throwable $e) {
        // No log of the exception : no one knows it happened.

        // return immediately ?
        return false;
    }

    $b->complete();

    return $b;
}

?>
```

See also Exceptions and Best practices for PHP exception handling.

Suggestions

- Add a log call to the catch block
- Handle correctly the exception

Specs

Short name	Exceptions/UselessCatch
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.1.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Zurmo*, *PrestaShop*

13.2.338 ext/amqp

Extension `amqp`.

PHP AMQP Binding Library. This is an interface with the RabbitMQ AMQP client library. It is a C-language AMQP client library for use with v2.0+ of the RabbitMQ broker.

```
<?php
$cnn = new AMQPConnection();
$cnn->connect();
echo 'Used channels: ', $cnn->getUsedChannels(), PHP_EOL;
$ch = new AMQPChannel($cnn);
echo 'Used channels: ', $cnn->getUsedChannels(), PHP_EOL;
$ch = new AMQPChannel($cnn);
echo 'Used channels: ', $cnn->getUsedChannels(), PHP_EOL;
$ch = null;
echo 'Used channels: ', $cnn->getUsedChannels(), PHP_EOL;
?>
```

See also PHP AMQP Binding Library.

Specs

Short name	Extensions/Extamqp
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.339 ext/apache

Extension Apache.

These functions are only available when running PHP as an Apache module.

```
<?php
  $ret = apache_getenv(SERVER_ADDR);
```

(continues on next page)

(continued from previous page)

```
  echo $ret;
?>
```

See also Extension Apache and Apache server.

Specs

Short name	Extensions/Extapache
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.340 ext/apc

Extension Alternative PHP Cache.

The Alternative PHP Cache (APC) is a free and open opcode cache for PHP. Its goal is to provide a free, open, and robust framework for caching and optimizing PHP intermediate code.

This extension is considered unmaintained and dead.

```
<?php
   $bar = 'BAR';
   apc_add('foo', $bar);
   var_dump(apc_fetch('foo'));
   echo PHP_EOL;

   $bar = 'NEVER GETS SET';
   apc_add('foo', $bar);
   var_dump(apc_fetch('foo'));
   echo PHP_EOL;
?>
```

See also Alternative PHP Cache.

Specs

Short name	Extensions/Extapc
Rulesets	*CE*, *CompatibilityPHP55*
Exakt since	0.8.4
Php Version	7.0-
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.341 ext/apcu

Extension APCU.

APCu is APC stripped of opcode caching. The Alternative PHP Cache (APC) is a free and open opcode cache for PHP. Its goal is to provide a free, open, and robust framework for caching and optimizing PHP intermediate code.

```
<?php
$bar = 'BAR';
apcu_add('foo', $bar);
var_dump(apcu_fetch('foo'));
echo \n;
$bar = 'NEVER GETS SET';
apcu_add('foo', $bar);
var_dump(apcu_fetch('foo'));
echo \n;
?>
```

See also APCU, ext/apcu and krakjoe/apcu.

Specs

Short name	Extensions/Extapcu
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.342 ext/array

Core functions processing arrays.

These functions manipulate arrays in various ways. Arrays are essential for storing, managing, and operating on sets of variables.

This is not a real extension : it is a documentation section, that helps classifying the functions.

```
<?php
function odd($var)
{
    // returns whether the input integer is odd
    return($var & 1);
}

function even($var)
{
    // returns whether the input integer is even
    return(!($var & 1));
}

$array1 = array('a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5);
$array2 = array(6, 7, 8, 9, 10, 11, 12);
```

(continues on next page)

(continued from previous page)

```
echo 'Odd :'.PHP_EOL;
print_r(array_filter($array1, 'odd'));
echo 'Even:'.PHP_EOL;
print_r(array_filter($array2, 'even'));
?>
```

See also Arrays.

Specs

Short name	Extensions/Extarray
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.343 ext/php-ast

PHP-AST extension (PHP 7.0 +).

```
<?php

$code = <<<'EOC'
<?php
$var = 42;
EOC;

var_dump(ast\parse_code($code, $version=50));

?>
```

See also ext/ast and Extension exposing PHP 7 abstract syntax tree.

Specs

Short name	Extensions/Extast
Rulesets	*CE*
Exakt since	0.8.4
Php Version	7.0+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.344 ext/async

Concurrent Task Extension for PHP.

This extension provides concurrent Zend VM executions using native C fibers in PHP.

```
<?php
namespace Concurrent;
register_shutdown_function(function () {
    echo "===> Shutdown function(s) execute here.\n";
});
$work = function (string $title): void {
    var_dump($title);
};
Task::await(Task::async(function () use ($work) {
    $defer = new Deferred();

    Task::await(Task::async($work, 'A'));
    Task::await(Task::async($work, 'B'));

    Task::async(function () {
        $defer = new Deferred();

        Task::async(function () use ($defer) {
            (new Timer(1000))->awaitTimeout();

            $defer->resolve('H :)');
        });

        var_dump(Task::await($defer->awaitable()));
    });

    Task::async(function () use ($defer) {
        var_dump(Task::await($defer->awaitable()));
    });

    $timer = new Timer(500);

    Task::async(function () use ($timer, $defer, $work) {
        $timer->awaitTimeout();

        $defer->resolve('F');

        Task::async($work, 'G');
    });

    var_dump('ROOT TASK DONE');
}));
Task::async($work, 'C');
Task::async(function () use ($work) {
    (new Timer(0))->awaitTimeout();

    Task::async($work, 'E');
});
Task::async(function ($v) {
    var_dump(Task::await($v));
}, Deferred::value('D'));
var_dump('=> END OF MAIN SCRIPT');

?>
```

See also ext-async repository.

Specs

Short name	Extensions/Extasync
Rulesets	*CE*
Exakt since	1.5.6
Php Version	7.3+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.345 ext/bcmath

Extension BC Math.

For arbitrary precision mathematics PHP offers the Binary Calculator which supports numbers of any size and precision up to `2147483647-1` (or `0x7FFFFFFF-1`) decimals, represented as strings.

```
<?php

echo bcpow('2', '123');
//10633823966279326983230456482242756608

echo 2**123;
//1.0633823966279E+37
?>
```

See also BC Math Functions.

Specs

Short name	Extensions/Extbcmath
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.346 ext/bzip2

Extension ext/bzip2.

Bzip2 Functions for PHP.

```
<?php

$file = '/tmp/foo.bz2';
$bz = bzopen($file, 'r') or die('Couldn\'t open $file for reading');

bzclose($bz);
```

(continues on next page)

(continued from previous page)

```
?>
```

See also Bzip2 Functions.

Specs

Short name	Extensions/Extbzip2
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.347 ext/cairo

Extension ext/cairo.

Cairo is a native PHP extension to create and modify graphics using the Cairo Graphics Library.

```
<?php
// Example from https://github.com/gtkforphp/cairo/blob/master/examples/big-line.php
$width = 100;
$height = 100;
$sur = new CairoPSSurface(temp.ps, $width, $height);

$con = new CairoContext($sur);
$con->setSourceRgb(0,0,1);
$con->moveTo(50,50);
$con->lineTo(50000,50000);
$con->stroke();
$con->setSourceRgb(0,1,0);
$con->moveTo(50,50);
$con->lineTo(-50000,50000);
$con->stroke();
$con->setSourceRgb(1,0,0);
$con->moveTo(50,50);
$con->lineTo(50000,-50000);
$con->stroke();
$con->setSourceRgb(1,1,0);
$con->moveTo(50,50);
$con->lineTo(-50000,-50000);
$con->stroke();

$sur->writeToPng(dirname(__FILE__)   . /big-line-php.png);
?>
```

See also cairo, gtkforphp/cairo.

Specs

Short name	Extensions/Extcairo
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.348 ext/calendar

Extension ext/calendar.

The calendar extension presents a series of functions to simplify converting between different calendar formats.

```
<?php
$number = cal_days_in_month(CAL_GREGORIAN, 8, 2003); // 31
echo "There were {$number} days in August 2003";
?>
```

See also Calendar Functions.

Specs

Short name	Extensions/Extcalendar
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.349 ext/cmark

Extension Cmark, for Common Mark.

cmark provides access to the reference implementation of CommonMark, a rationalized version of Markdown syntax with a specification.

```
<?php
$text = new CommonMark\Node\Text;
$text->literal = 'Hello World';
$document = new CommonMark\Node\Document;
$document->appendChild(
     (new CommonMark\Node\Paragraph)
             ->appendChild($text));
echo CommonMark\Render\HTML($document);
?>
```

See also Cmark and ext/cmark.

Specs

Short name	Extensions/Extcmark
Rulesets	*CE*
Exakt since	1.2.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.350 ext/com

Extension COM and `.Net` (Windows).

COM is an acronym for 'Component Object Model'; it is an object orientated layer (and associated services) on top of DCE RPC (an open standard) and defines a common calling convention that enables code written in any language to call and interoperate with code written in any other language (provided those languages are COM aware).

```
<?php
$domainObject = new COM(WinNT://Domain);
foreach ($domainObject as $obj) {
   echo $obj->Name . <br />;
}
?>
```

See also COM and .Net (Windows).

Specs

Short name	Extensions/Extcom
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.351 ext/crypto

Extension ext/crypto (PECL).

Objective PHP binding of OpenSSL Crypto library.

```
<?php
use Crypto\Cipher;
use Crypto\AlgorihtmException;
$algorithm = 'aes-256-cbc';
if (!Cipher::hasAlgorithm($algorithm)) {
     die('Algorithm $algorithm not found' . PHP_EOL);
}
try {
```

(continues on next page)

(continued from previous page)

```
    $cipher = new Cipher($algorithm);
    // Algorithm method for retrieving algorithm
    echo 'Algorithm: ' . $cipher->getAlgorithmName() . PHP_EOL;
    // Params
    $key_len = $cipher->getKeyLength();
    $iv_len = $cipher->getIVLength();

    echo 'Key length: ' . $key_len . PHP_EOL;
    echo 'IV length: '  . $iv_len . PHP_EOL;
    echo 'Block size: ' . $cipher->getBlockSize() . PHP_EOL;
    // This is just for this example. You should never use such key and IV!
    $key = str_repeat('x', $key_len);
    $iv = str_repeat('i', $iv_len);
    // Test data
    $data1 = 'Test';
    $data2 = 'Data';
    $data = $data1 . $data2;
    // Simple encryption
    $sim_ct = $cipher->encrypt($data, $key, $iv);

    // init/update/finish encryption
    $cipher->encryptInit($key, $iv);
    $iuf_ct  = $cipher->encryptUpdate($data1);
    $iuf_ct .= $cipher->encryptUpdate($data2);
    $iuf_ct .= $cipher->encryptFinish();
    // Raw data output (used base64 format for printing)
    echo 'Ciphertext (sim): ' . base64_encode($sim_ct) . PHP_EOL;
    echo 'Ciphertext (iuf): ' . base64_encode($iuf_ct) . PHP_EOL;
    // $iuf_out == $sim_out
    $ct = $sim_ct;
    // Another way how to create a new cipher object (using the same algorithm and␣
↪mode)
    $cipher = Cipher::aes(Cipher::MODE_CBC, 256);
    // Simple decryption
    $sim_text = $cipher->decrypt($ct, $key, $iv);

    // init/update/finish decryption
    $cipher->decryptInit($key, $iv);
    $iuf_text = $cipher->decryptUpdate($ct);
    $iuf_text .= $cipher->decryptFinish();
    // Raw data output ($iuf_out == $sim_out)
    echo 'Text (sim): ' . $sim_text . PHP_EOL;
    echo 'Text (iuf): ' . $iuf_text . PHP_EOL;
}
catch (AlgorithmException $e) {
    echo $e->getMessage() . PHP_EOL;
}

?>
```

See also pecl crypto and php-crypto.

Specs

Short name	Extensions/Extcrypto
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.352 ext/csprng

CSPRNG Functions : cryptographically secure pseudo-random number generator.

The CSPRNG API provides an easy and reliable way to generate crypto-strong random integers and bytes for use within cryptographic contexts.

```
<?php
$bytes = random_bytes(5);
var_dump(bin2hex($bytes));

//string(10) 385e33f741
?>
```

See also CSPRNG and Cryptographically secure pseudorandom number generator.

Specs

Short name	Extensions/Extcsprng
Rulesets	*CE*
Exakt since	1.3.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.353 ext/ctype

Extension ext/ctype.

Ext/ctype checks whether a character or string falls into a certain character class according to the current locale.

```
<?php
$strings = array('AbCd1zyZ9', 'foo!#$bar');
foreach ($strings as $testcase) {
    if (ctype_alnum($testcase)) {
        echo "The string $testcase consists of all letters or digits.\n";
    } else {
        echo "The string $testcase does not consist of all letters or digits.\n";
    }
```

(continues on next page)

(continued from previous page)

```
}
?>
```

See also Ctype funtions.

Specs

Short name	Extensions/Extctype
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.354 ext/curl

Extension curl.

PHP supports libcurl, a library created by Daniel Stenberg. It allows the connection and communication to many different types of servers with many different types of protocols.

```
<?php

$ch = curl_init("http://www.example.com/");
$fp = fopen("example_homepage.txt", "w");

curl_setopt($ch, CURLOPT_FILE, $fp);
curl_setopt($ch, CURLOPT_HEADER, 0);

curl_exec($ch);
curl_close($ch);
fclose($fp);
?>
```

See also Curl for PHP and curl.

Specs

Short name	Extensions/Extcurl
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.355 ext/cyrus

Extension ext/cyrus.

The Cyrus IMAP server is electronic mail server software developed by Carnegie Mellon University.

```
<?php

$connexion = cyrus_connect ('localhost');

?>
```

See also Cyrus and Cyrus IMAP server.

Specs

Short name	Extensions/Extcyrus
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.356 ext/date

Extension ext/date.

These functions allows the manipulation of date and time from the server where the PHP scripts are running.

```
<?php
$dt = new DateTime('2015-11-01 00:00:00', new DateTimeZone('America/New_York'));
echo 'Start: ', $dt->format('Y-m-d H:i:s P'), PHP_EOL;
$dt->add(new DateInterval('PT3H'));
echo 'End:   ', $dt->format('Y-m-d H:i:s P'), PHP_EOL;
?>
```

See also Date and Time.

Specs

Short name	Extensions/Extdate
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.357 ext/db2

Extension for IBM DB2, Cloudscape and Apache Derby.

This extension gives access to IBM DB2 Universal Database, IBM Cloudscape, and Apache Derby databases using the DB2 Call Level Interface (DB2 CLI).

```
<?php
$conn = db2_connect($database, $user, $password);

if ($conn) {
    $stmt = db2_exec($conn, 'SELECT count(*) FROM animals');
    $res = db2_fetch_array( $stmt );
    echo $res[0] . PHP_EOL;

    // Turn AUTOCOMMIT off
    db2_autocommit($conn, DB2_AUTOCOMMIT_OFF);

    // Delete all rows from ANIMALS
    db2_exec($conn, 'DELETE FROM animals');

    $stmt = db2_exec($conn, 'SELECT count(*) FROM animals');
    $res = db2_fetch_array( $stmt );
    echo $res[0] . PHP_EOL;

    // Roll back the DELETE statement
    db2_rollback( $conn );

    $stmt = db2_exec( $conn, 'SELECT count(*) FROM animals' );
    $res = db2_fetch_array( $stmt );
    echo $res[0] . PHP_EOL;
    db2_close($conn);
}
?>
```

See also IBM Db2.

Specs

Short name	Extensions/Extdb2
Rulesets	*CE*
Exakt since	1.1.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.358 ext/dba

Extension ext/dba.

These functions build the foundation for accessing Berkeley DB style databases.

```
<?php

$id = dba_open('/tmp/test.db', 'n', 'db2');

if (!$id) {
    echo 'dba_open failed'.PHP_EOL;
    exit;
}

dba_replace('key', 'This is an example!', $id);

if (dba_exists('key', $id)) {
    echo dba_fetch('key', $id);
    dba_delete('key', $id);
}

dba_close($id);
?>
```

See also Database (dbm-style) Abstraction Layer.

Specs

Short name	Extensions/Extdba
Rulesets	*CE*, *CompatibilityPHP53*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.359 ext/decimal

Extension php-decimal, by `Rudi Theunissen`.

This library provides a PHP extension that adds support for correctly-rounded, arbitrary-precision decimal floating point arithmetic. Applications that rely on accurate numbers (ie. money, measurements, or mathematics) can use Decimal instead of float or string to represent numerical values.

```
<?php

use Decimal\Decimal;

$op1 = new Decimal(0.1, 4);
$op2 = 0.123456789;

print_r($op1 + $op2);


use Decimal\Decimal;

/**
 * @param int $n The factorial to calculate, ie. $n!
```

(continues on next page)

(continued from previous page)

```
 * @param int $p The precision to calculate the factorial to.
 *
 * @return Decimal
 */
function factorial(int $n, int $p = Decimal::DEFAULT_PRECISION): Decimal
{
    return $n < 2 ? new Decimal($n, $p) : $n * factorial($n - 1, $p);
}

echo factorial(10000, 32);

?>
```

See also PHP Decimal and libmpdec.

Specs

Short name	Extensions/Extdecimal
Rulesets	*CE*
Exakt since	1.5.2
Php Version	7.0+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.360 ext/dio

Extension DIO : Direct Input Output.

PHP supports the direct io functions as described in the Posix Standard (Section 6) for performing I/O functions at a lower level than the C-Language stream I/O functions

```
<?php

$fd = dio_open('/dev/ttyS0', O_RDWR | O_NOCTTY | O_NONBLOCK);

dio_close($fd);
?>
```

See also DIO.

Specs

Short name	Extensions/Extdio
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.361 ext/dom

Extension Document Object Model.

The DOM extension allows the manipulation of XML documents through the DOM API with PHP.

```
<?php

$dom = new DOMDocument('1.0', 'utf-8');

$element = $dom->createElement('test', 'This is the root element!');

// We insert the new element as root (child of the document)
$dom->appendChild($element);

echo $dom->saveXML();
?>
```

See also Document Object Model.

Specs

Short name	Extensions/Extdom
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.362 ext/ds

Extension Data Structures : Data structures.

See also : Efficient data structures for PHP 7.

```
<?php

$vector = new \Ds\Vector();

$vector->push('a');
$vector->push('b', 'c');

$vector[] = 'd';

print_r($vector);

?>
```

Specs

Short name	Extensions/Extds
Rulesets	*CE*
Exakt since	0.10.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.363 ext/eaccelerator

Extension Eaccelerator.

eAccelerator is a free open-source PHP accelerator & optimizer.

See also Eaccelerator and eaccelerator/eaccelerato.

Specs

Short name	Extensions/Exteaccelerator
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.364 ext/eio

Extension EIO.

This is a PHP extension wrapping functions of the libeio library written by Marc Lehmann.

Libeio is a an asynchronous I/O library. Features basically include asynchronous versions of POSIX API(read, write, open, close, stat, unlink, fdatasync, mknod, readdir etc.); sendfile (native on Solaris, Linux, HP-UX, FreeBSD); readahead. libeio itself emulates the system calls, if they are not available on specific(UNIX-like) platform.

```
<?php
$str      = str_repeat('1', 20);
$filename = '/tmp/tmp_file' .uniqid();
@unlink($filename);
touch($filename);
eio_open($filename, EIO_O_RDWR, NULL, EIO_PRI_DEFAULT, function($filename, $fd) use (
↪$str) {
    eio_write($fd, $str, strlen($str), 0, null, function($fd, $written) use ($str,
↪$filename) {
            var_dump([
                    'written'  => $written,
                    'strlen'   => strlen($str),
                    'filesize' => filesize($filename),
                    'count'    => substr_count(file_get_contents($filename), '1')
```

(continues on next page)

(continued from previous page)

```
                        ]);
    }, $fd);
}, $filename);
eio_event_loop();
?>
```

See also libeio, PHP extension for libeio.

Specs

Short name	Extensions/Exteio
Rulesets	*CE*
Exakt since	1.3.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.365 ext/enchant

Extension Enchant.

Enchant is the PHP binding for the Enchant spelling library. Enchant steps in to provide uniformity and conformity on top of all spelling libraries, and implement certain features that may be lacking in any individual provider library.

```
<?php
$tag = 'en_US';
$r = enchant_broker_init();
$bprovides = enchant_broker_describe($r);
echo 'Current broker provides the following backend(s):'.PHP_EOL;
print_r($bprovides);

$dicts = enchant_broker_list_dicts($r);
print_r($dicts);
if (enchant_broker_dict_exists($r,$tag)) {
    $d = enchant_broker_request_dict($r, $tag);
    $dprovides = enchant_dict_describe($d);
    echo 'dictionary $tag provides:'.PHP_EOL;
    $wordcorrect = enchant_dict_check($d, 'soong');
    print_r($dprovides);
    if (!$wordcorrect) {
        $suggs = enchant_dict_suggest($d, 'soong');
        echo 'Suggestions for "soong":';
        print_r($suggs);
    }
    enchant_broker_free_dict($d);
} else {
}
enchant_broker_free($r);
?>
```

See also Enchant spelling library and Enchant.

Specs

Short name	Extensions/Extenchant
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.366 ext/ereg

Extension ext/ereg.

```
<?php
if (ereg ('([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})', $date, $regs)) {
    echo $regs[3].'.'.$regs[2].'.'.$regs[1];
} else {
    echo 'Invalid date format: '.$date;
}
?>
```

See also Ereg.

Specs

Short name	Extensions/Extereg
Rulesets	*CE*, *CompatibilityPHP70*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.367 ext/ev

Extension ev.

ext/ev is a high performance full-featured event loop written in C.

```
<?php
// Create and start timer firing after 2 seconds
$w1 = new EvTimer(2, 0, function () {
    echo '2 seconds elapsed'.PHP_EOL;
});

// Create and launch timer firing after 2 seconds repeating each second
// until we manually stop it
$w2 = new EvTimer(2, 1, function ($w) {
    echo 'is called every second, is launched after 2 seconds'.PHP_EOL;
    echo 'iteration = ', Ev::iteration(), PHP_EOL;
```

(continues on next page)

(continued from previous page)

```
    // Stop the watcher after 5 iterations
    Ev::iteration() == 5 and $w->stop();
    // Stop the watcher if further calls cause more than 10 iterations
    Ev::iteration() >= 10 and $w->stop();
});

// Create stopped timer. It will be inactive until we start it ourselves
$w_stopped = EvTimer::createStopped(10, 5, function($w) {
    echo 'Callback of a timer created as stopped'.PHP_EOL;

    // Stop the watcher after 2 iterations
    Ev::iteration() >= 2 and $w->stop();
});

// Loop until Ev::stop() is called or all of watchers stop
Ev::run();

// Start and look if it works
$w_stopped->start();
echo 'Run single iteration'.PHP_EOL;
Ev::run(Ev::RUN_ONCE);

echo 'Restart the second watcher and try to handle the same events, but don\'t block'.
↪PHP_EOL;
$w2->again();
Ev::run(Ev::RUN_NOWAIT);

$w = new EvTimer(10, 0, function() {});
echo 'Running a blocking loop'.PHP_EOL;
Ev::run();
echo 'END'.PHP_EOL;
?>
```

See also Ev and libev.

Specs

Short name	Extensions/Extev
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.368 ext/event

Extension event.

This is an extension to efficiently schedule I/O, time and signal based events using the best I/O notification mechanism available for specific platform. This is a port of libevent to the PHP infrastructure.

```php
<?php
// Read callback
function readcb($bev, $base) {
    //$input = $bev->input; //$bev->getInput();

    //$pos = $input->search('TTP');
    $pos = $bev->input->search('TTP');

    while (($n = $bev->input->remove($buf, 1024)) > 0) {
        echo $buf;
    }
}

// Event callback
function eventcb($bev, $events, $base) {
    if ($events & EventBufferEvent::CONNECTED) {
        echo 'Connected.';
    } elseif ($events & (EventBufferEvent::ERROR | EventBufferEvent::EOF)) {
        if ($events & EventBufferEvent::ERROR) {
            echo 'DNS error: ', $bev->getDnsErrorString(), PHP_EOL;
        }

        echo 'Closing'.PHP_EOL;
        $base->exit();
        exit('Done'.PHP_EOL);
    }
}

if ($argc != 3) {
    echo <<<EOS
Trivial HTTP 0.x client
Syntax: php {$argv[0]} [hostname] [resource]
Example: php {$argv[0]} www.google.com /

EOS;
    exit();
}

$base = new EventBase();

$dns_base = new EventDnsBase($base, TRUE); // We'll use async DNS resolving
if (!$dns_base) {
    exit('Failed to init DNS Base'.PHP_EOL);
}

$bev = new EventBufferEvent($base, /* use internal socket */ NULL,
    EventBufferEvent::OPT_CLOSE_ON_FREE | EventBufferEvent::OPT_DEFER_CALLBACKS,
    'readcb', /* writecb */ NULL, 'eventcb'
);
if (!$bev) {
    exit('Failed creating bufferevent socket'.PHP_EOL);
}

//$bev->setCallbacks('readcb', /* writecb */ NULL, 'eventcb', $base);
$bev->enable(Event::READ | Event::WRITE);

$output = $bev->output; //$bev->getOutput();
```

(continues on next page)

(continued from previous page)

```
if (!$output->add(
    'GET '.$argv[2].' HTTP/1.0'."\r\n".
    'Host: '.$argv[1]."\r\n".
    'Connection: Close'.\r\n\r\n
)) {
    exit('Failed adding request to output buffer\n');
}

if (!$bev->connectHost($dns_base, $argv[1], 80, EventUtil::AF_UNSPEC)) {
    exit('Can\'t connect to host '.$argv[1].PHP_EOL);
}

$base->dispatch();
?>
```

See also Event and libevent.

Specs

Short name	Extensions/Extevent
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.369 ext/exif

Extension EXIF : Exchangeable image file format.

The EXIF extension manipulates image meta data.

```
<?php
echo 'test1.jpg:<br />';
$exif = exif_read_data('tests/test1.jpg', 'IFD0');
echo $exif===false ? 'No header data found.<br />' : 'Image contains headers<br />';

$exif = exif_read_data('tests/test2.jpg', 0, true);
echo 'test2.jpg:<br />';
foreach ($exif as $key => $section) {
    foreach ($section as $name => $val) {
        echo $key.$name.': '.$val.'<br />';
    }
}
?>
```

See also Exchangeable image information.

Specs

Short name	Extensions/Extexif
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.370 ext/expect

Extension Expect.

This extension allows to interact with processes through `PTY`. You may consider using the `expect://` wrapper with the filesystem functions which provide a simpler and more intuitive interface.

```
<?php
ini_set('expect.loguser', 'Off');

$stream = fopen('expect://ssh root@remotehost uptime', 'r');

$cases = array (
    array (0 => 'password:', 1 => PASSWORD)
);

switch (expect_expectl ($stream, $cases)) {
    case PASSWORD:
        fwrite ($stream, 'password'.PHP_EOL);
        break;

    default:
        die ('Error was occurred while connecting to the remote host!'.PHP_EOL);
}

while ($line = fgets($stream)) {
      print $line;
}
fclose ($stream);
?>
```

See also expect.

Specs

Short name	Extensions/Extexpect
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.371 ext/fam

File Alteration Monitor extension.

FAM monitors files and directories, notifying interested applications of changes.

ext/FAM is not available for Windows

```
<?php

$fam = fam_open('myApplication');
fam_monitor_directory($fam, '/tmp');
fam_close($fam);

?>
```

See also File Alteration Monitor.

Specs

Short name	Extensions/Extfam
Rulesets	*CE*
Exakt since	0.12.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.372 ext/fann

Extension `FANN` : Fast Artificial Neural Network.

PHP binding for `FANN` library which implements multi-layer artificial neural networks with support for both fully connected and sparsely connected networks.

```
<?php
$num_input = 2;
$num_output = 1;
$num_layers = 3;
$num_neurons_hidden = 3;
$desired_error = 0.001;
$max_epochs = 500000;
$epochs_between_reports = 1000;

$ann = fann_create_standard($num_layers, $num_input, $num_neurons_hidden, $num_
↪output);

if ($ann) {
    fann_set_activation_function_hidden($ann, FANN_SIGMOID_SYMMETRIC);
    fann_set_activation_function_output($ann, FANN_SIGMOID_SYMMETRIC);

    $filename = dirname(__FILE__) . '/xor.data';
    if (fann_train_on_file($ann, $filename, $max_epochs, $epochs_between_reports,
↪$desired_error))
```

(continues on next page)

(continued from previous page)

```
        fann_save($ann, dirname(__FILE__) . '/xor_float.net');

    fann_destroy($ann);
}
?>
```

See also extension FANN, PHP-ML, Rubix ML, and lib FANN.

Specs

Short name	Extensions/Extfann
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.373 ext/fdf

Extension ext/fdf.

Forms Data Format (FDF) is a format for handling forms within PDF documents.

```
<?php
$outfdf = fdf_create();
fdf_set_value($outfdf, 'volume', $volume, 0);

fdf_set_file($outfdf, 'http:/testfdf/resultlabel.pdf');
fdf_save($outfdf, 'outtest.fdf');
fdf_close($outfdf);
Header('Content-type: application/vnd.fdf');
$fp = fopen('outtest.fdf', 'r');
fpassthru($fp);
unlink('outtest.fdf');
?>
```

See also Form Data Format.

Specs

Short name	Extensions/Extfdf
Rulesets	*CE*, *CompatibilityPHP53*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.374 ext/ffi

Extension `FFI` : Foreign Function Interface .

This extension allows the loading of shared libraries (.DLL or .so), calling of C functions and accessing of C data structures in pure PHP, without having to have deep knowledge of the Zend extension API, and without having to learn a third "intermediate" language. The public API is implemented as a single class FFI with several static methods (some of them may be called dynamically), and overloaded object methods, which perform the actual interaction with C data.

```
<?php
//Example : Calling a function from shared library
// create FFI object, loading libc and exporting function printf()
$ffi = FFI::cdef(
    "int printf(const char *format, ...);", // this is a regular C declaration
    "libc.so.6");
// call C's printf()
$ffi->printf("Hello %s!\n", "world");
?>
```

See also Foreign Function Interface <https://www.php.net/manual/en/book.'ffi.php>'_, and ext/'ffi <https://github.com/dstogov/php-ffi>'_ and A PHP Compiler, aka The 'FFI Rabbit Hole <https://blog.ircmaxell.com/2019/04/compilers-ffi.html>'_.

Suggestions

-

Specs

Short name	Extensions/Extffi
Rulesets	*CE*
Exakt since	1.7.9
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.375 ext/ffmpeg

Extension `ffmpeg` for PHP.

`ffmpeg-php` is an extension for PHP that adds an easy to use, object-oriented API for accessing and retrieving information from video and audio files.

```
<?php

$movie = new ffmpeg_movie($path_to_media, $persistent);
echo 'The movie lasts '.$movie->getDuration().' seconds';

?>
```

See also ffmpeg-php and FFMPEG.

Specs

Short name	Extensions/Extffmpeg
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.376 ext/file

Filesystem functions from standard.

Extension that handle access to file on the file system.

```
<?php
$row = 1;
if (($handle = fopen('test.csv', 'r')) !== FALSE) {
    while (($data = fgetcsv($handle, 1000, ',')) !== FALSE) {
        $num = count($data);
        echo '<p> $num fields in line $row: <br /></p>'.PHP_EOL;
        $row++;
        for ($c=0; $c < $num; $c++) {
            echo $data[$c] . '<br />'.PHP_EOL;
        }
    }
    fclose($handle);
}
?>
```

See also filesystem.

Specs

Short name	Extensions/Extfile
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.377 ext/fileinfo

Extension ext/fileinfo.

This module guesses the content type and encoding of a file by looking for certain magic byte sequences at specific positions within the file.

```
<?php
$finfo = finfo_open(FILEINFO_MIME_TYPE); // return mime type ala mimetype extension
foreach (glob('*') as $filename) {
    echo finfo_file($finfo, $filename) . PHP_EOL;
}
finfo_close($finfo);
?>
```

See also Filinfo.

Specs

Short name	Extensions/Extfileinfo
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.378 ext/filter

Extension filter.

This extension filters data by either validating or sanitizing it.

```
<?php
$email_a = 'joe@example.com';
$email_b = 'bogus';

if (filter_var($email_a, FILTER_VALIDATE_EMAIL)) {
    echo 'This ($email_a) email address is considered valid.'.PHP_EOL;
}
if (filter_var($email_b, FILTER_VALIDATE_EMAIL)) {
    echo 'This ($email_b) email address is considered valid.'.PHP_EOL;
} else {
    echo 'This ($email_b) email address is considered invalid.'.PHP_EOL;
}
?>
```

See also Data filtering.

Specs

Short name	Extensions/Extfilter
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.379 ext/fpm

Extension FPM, FastCGI Process Manager.

FPM (FastCGI Process Manager) is an alternative PHP FastCGI implementation with some additional features (mostly) useful for heavy-loaded sites. .. code-block:: php

<?php echo $text; fastcgi_finish_request( );

?>

See also FastCGI Process Manager.

Specs

Short name	Extensions/Extfpm
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.380 ext/ftp

Extension FTP.

The functions in this extension implement client access to files servers speaking the File Transfer Protocol (FTP) as defined in RFC 959.

```
<?php
// set up basic connection
$conn_id = ftp_connect($ftp_server);

// login with username and password
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// check connection
if ((!$conn_id) || (!$login_result)) {
    echo 'FTP connection has failed!';
    echo 'Attempted to connect to $ftp_server for user $ftp_user_name';
    exit;
} else {
    echo 'Connected to $ftp_server, for user $ftp_user_name';
}

// upload the file
$upload = ftp_put($conn_id, $destination_file, $source_file, FTP_BINARY);

// check upload status
if (!$upload) {
    echo 'FTP upload has failed!';
} else {
    echo 'Uploaded $source_file to $ftp_server as $destination_file';
}
```

(continues on next page)

(continued from previous page)

```
// close the FTP stream
ftp_close($conn_id);
?>
```

See also FTP.

Specs

Short name	Extensions/Extftp
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.381 ext/gd

Extension GD for PHP.

This extension allows PHP to create and manipulate image files in a variety of different image formats, including GIF, PNG, JPEG, WBMP, and XPM.

```
<?php

header("Content-type: image/png");
$string = $_GET['text'];
$im     = imagecreatefrompng("images/button1.png");
$orange = imagecolorallocate($im, 220, 210, 60);
$px     = (imagesx($im) - 7.5 * strlen($string)) / 2;
imagestring($im, 3, $px, 9, $string, $orange);
imagepng($im);
imagedestroy($im);

?>
```

See also Image Processing and GD.

Specs

Short name	Extensions/Extgd
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.382 ext/gearman

Extension Gearman.

Gearman is a generic application framework for farming out work to multiple machines or processes.

```
<?php

# Create our client object.
$gmclient= new GearmanClient();

# Add default server (localhost).
$gmclient->addServer();

echo 'Sending job'.PHP_EOL;

# Send reverse job
do
{
  $result = $gmclient->doNormal('reverse', 'Hello!');

  # Check for various return packets and errors.
  switch($gmclient->returnCode())
  {
    case GEARMAN_WORK_DATA:
      echo 'Data: '.$result . PHP_EOL;;
      break;
    case GEARMAN_WORK_STATUS:
      list($numerator, $denominator)= $gmclient->doStatus();
      echo 'Status: '.$numerator.'/'.$denominator.' complete'. PHP_EOL;
      break;
    case GEARMAN_WORK_FAIL:
      echo 'Failed\n';
      exit;
    case GEARMAN_SUCCESS:
      echo 'Success: $result\n';
      break;
    default:
      echo 'RET: ' . $gmclient->returnCode() . PHP_EOL;
      exit;
  }
}
while($gmclient->returnCode() != GEARMAN_SUCCESS);

?>
```

See also Gearman on PHP and Gearman.

Specs

Short name	Extensions/Extgearman
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.383 ext/gender

Gender extension.

The Gender PHP extension is a port of the gender.c program originally written by Joerg Michael. Its main purpose is to find out the gender of firstnames, based on a database of over 40000 firstnames from 54 countries.

```
<?php

namespace Gender;

$gender = new Gender;


$name = 'Milene';
$country = Gender::FRANCE;

$result = $gender->get($name, $country);

$data = $gender->country($country);

switch($result) {
    case Gender::IS_FEMALE:
        printf('The name %s is female in %s\n', $name, $data['country']);
    break;


    case Gender::IS_MOSTLY_FEMALE:
        printf('The name %s is mostly female in %s\n', $name, $data['country']);
    break;


    case Gender::IS_MALE:
        printf('The name %s is male in %s\n', $name, $data['country']);
    break;


    case Gender::IS_MOSTLY_MALE:
        printf('The name %s is mostly male in %s\n', $name, $data['country']);
    break;


    case Gender::IS_UNISEX_NAME:
        printf('The name %s is unisex in %s\n', $name, $data['country']);
    break;
```

(continues on next page)

(continued from previous page)

```
    case Gender::IS_A_COUPLE:
        printf('The name %s is both male and female in %s\n', $name, $data['country
↪']);
    break;


    case Gender::NAME_NOT_FOUND:
        printf('The name %s was not found for %s\n', $name, $data['country']);
    break;


    case Gender::ERROR_IN_NAME:
        echo 'There is an error in the given name!'.PHP_EOL;
    break;

    default:
        echo 'An error occurred!'.PHP_EOL;
    break;

}

?>
```

See also ext/gender manual and genderReader.

Specs

Short name	Extensions/Extgender
Rulesets	*CE*
Exakt since	0.11.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.384 ext/geoip

Extension geoip for PHP.

The GeoIP extension allows the localisation of an IP address.

```
<?php
$org = geoip_org_by_name('www.example.com');
if ($org) {
    echo 'This host IP is allocated to: ' . $org;
}
?>
```

See also GeoIP.

Specs

Short name	Extensions/Extgeoip
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.385 ext/gettext

Extension Gettext.

The gettext functions implement an NLS (Native Language Support) API which can be used to internationalize your PHP applications.

```
<?php
// Set language to German
putenv('LC_ALL=de_DE');
setlocale(LC_ALL, 'de_DE');

// Specify location of translation tables
bindtextdomain('myPHPApp', './locale');

// Choose domain
textdomain('myPHPApp');

// Translation is looking for in ./locale/de_DE/LC_MESSAGES/myPHPApp.mo now

// Print a test message
echo gettext('Welcome to My PHP Application');

// Or use the alias _() for gettext()
echo _('Have a nice day');
?>
```

See also Gettext and ext/gettext

Specs

Short name	Extensions/Extgettext
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.386 ext/gmagick

Extension gmagick.

Gmagick is a php extension to create, modify and obtain meta information of images using the GraphicsMagick API.

```
<?php
//Instantiate a new Gmagick object
$image = new Gmagick('example.jpg');

//Make thumbnail from image loaded. 0 for either axes preserves aspect ratio
$image->thumbnailImage(100, 0);

//Create a border around the image, then simulate how the image will look like as an␣
↪oil painting
//Note the chaining of mutator methods which is supported in gmagick
$image->borderImage(yellow, 8, 8)->oilPaintImage(0.3);

//Write the current image at the current state to a file
$image->write('example_thumbnail.jpg');
?>
```

See also PHP gmagick and gmagick.

Specs

Short name	Extensions/Extgmagick
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.387 ext/gmp

Extension ext/gmp.

These functions allow for arbitrary-length integers to be worked with using the GNU MP library.

```
<?php
$pow1 = gmp_pow('2', 131);
echo gmp_strval($pow1) . PHP_EOL;
$pow2 = gmp_pow('0', 0);
echo gmp_strval($pow2) . PHP_EOL;
$pow3 = gmp_pow('2', -1); // Negative exp, generates warning
echo gmp_strval($pow3) . PHP_EOL;
?>
```

See also GMP and GNU MP library.

Specs

Short name	Extensions/Extgmp
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.388 ext/gnupgp

Extension GnuPG.

This module allows you to interact with gnupg.

```
<?php
// init gnupg
$res = gnupg_init();
// not really needed. Clearsign is default
gnupg_setsignmode($res,GNUPG_SIG_MODE_CLEAR);
// add key with passphrase 'test' for signing
gnupg_addsignkey($res,"8660281B6051D071D94B5B230549F9DC851566DC","test");
// sign
$signed = gnupg_sign($res,"just a test");
echo $signed;
?>
```

See also Gnupg Function for PHP <http://www.php.net/manual/en/book.‘gnupg.php>‘_ and GnuPG <https://www.‘gnupg.org/>‘_.

Specs

Short name	Extensions/Extgnupg
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.389 ext/grpc

Extension for GRPC : A high performance, open-source universal RPC framework.

```
<?php

//https://github.com/grpc/grpc/blob/master/examples/php/greeter_client.php

require dirname(__FILE__).'/vendor/autoload.php';
```

(continues on next page)

(continued from previous page)

```
// The following includes are needed when using protobuf 3.1.0
// and will suppress warnings when using protobuf 3.2.0+
@include_once dirname(__FILE__).'/helloworld.pb.php';
@include_once dirname(__FILE__).'/helloworld_grpc_pb.php';
function greet($name)
{
    $client = new Helloworld\GreeterClient('localhost:50051', [
        'credentials' => Grpc\ChannelCredentials::createInsecure(),
    ]);
    $request = new Helloworld\HelloRequest();
    $request->setName($name);
    list($reply, $status) = $client->SayHello($request)->wait();
    $message = $reply->getMessage();
    return $message;
}
$name = !empty($argv[1]) ? $argv[1] : 'world';
echo greet($name).\n;

?>
```

See also GRPC and GRPC on PECL.

Specs

Short name	Extensions/Extgrpc
Rulesets	*CE*
Exakt since	0.11.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.390 ext/hash

Extension for HASH Message Digest Framework.

Message Digest (hash) engine. Allows direct or incremental processing of arbitrary length messages using a variety of hashing algorithms.

```
<?php
/* Create a file to calculate hash of */
file_put_contents('example.txt', 'The quick brown fox jumped over the lazy dog.');

echo hash_file('md5', 'example.txt');
?>
```

See also HASH Message Digest Framework.

Specs

Short name	Extensions/Exthash
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.391 ext/hrtime

High resolution timing Extension.

The HRTime extension implements a high resolution *StopWatch* class. It uses the best possible API on different platforms which brings resolution up to nanoseconds. It also makes possible to implement a custom stopwatch using low level ticks delivered by the underlaying system.

```
<?php

$c = new HRTime\StopWatch;

$c->start();
/* measure this code block execution */
for ($i = 0; $i < 1024*1024; $i++);
$c->stop();
$elapsed0 = $c->getLastElapsedTime(HRTime\Unit::NANOSECOND);

/* measurement is not running here*/
for ($i = 0; $i < 1024*1024; $i++);

$c->start();
/* measure this code block execution */
for ($i = 0; $i < 1024*1024; $i++);
$c->stop();
$elapsed1 = $c->getLastElapsedTime(HRTime\Unit::NANOSECOND);

$elapsed_total = $c->getElapsedTime(HRTime\Unit::NANOSECOND);

?>
```

See also ext/hrtime manual.

Specs

Short name	Extensions/Exthrtime
Rulesets	*CE*
Exakt since	1.1.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.392 ext/pecl_http

Extension HTTP.

This HTTP extension aims to provide a convenient and powerful set of functionalities for one of PHP major applications.

It eases handling of HTTP URL, headers and messages, provides means for negotiation of a client's preferred content type, language and charset, as well as a convenient way to send any arbitrary data with caching and resuming capabilities.

It provides powerful request functionality with support for parallel requests.

```
<?php

$client = new http\Client;
$client->setSslOptions(array("verifypeer" => true));
$client->addSslOptions(array("verifyhost" => 2));

$client->enqueue($req = new http\Client\Request("GET", "https://twitter.com/"));
$client->send();
$ti = (array) $client->getTransferInfo($req);
var_dump($ti);

?>
```

See also ext-http and pecl_http.

Specs

Short name	Extensions/Exthttp
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.393 ext/ibase

Extensions `Interbase` and `Firebird`.

`Firebird` is a relational database offering many ISO SQL-2003 features that runs on Linux, Windows, and a variety of Unix platforms.

```
<?php

$host = 'localhost:/path/to/your.gdb';

$dbh = ibase_connect($host, $username, $password);
$stmt = 'SELECT * FROM tblname';

$sth = ibase_query($dbh, $stmt) or die(ibase_errmsg());

?>
```

See also Firebase / Interbase and Firebird.

Specs

Short name	Extensions/Extibase
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.394 ext/iconv

Extension ext/iconv.

With this module, you can turn a string represented by a local character set into the one represented by another character set, which may be the Unicode character set.

```
<?php
$text = "This is the Euro symbol '€'.";

echo 'Original : ', $text, PHP_EOL;
echo 'TRANSLIT : ', iconv("UTF-8", "ISO-8859-1//TRANSLIT", $text), PHP_EOL;
echo 'IGNORE   : ', iconv("UTF-8", "ISO-8859-1//IGNORE", $text), PHP_EOL;
echo 'Plain    : ', iconv("UTF-8", "ISO-8859-1", $text), PHP_EOL;

?>
```

See also Iconv, and libiconv.

Specs

Short name	Extensions/Exticonv
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.395 ext/igbinary

Extension igbinary.

igbinary is a drop in replacement for the standard php serializer. Instead of time and space consuming textual representation, igbinary stores php data structures in compact binary form.

```
<?php
     $serialized = igbinary_serialize($variable);
     $unserialized = igbinary_unserialize($serialized);
?>
```

See also igbinary.

Specs

Short name	Extensions/Extigbinary
Rulesets	*CE*
Exakt since	1.0.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.396 ext/iis

Extension IIS Administration.

It provides functions to administrate Microsoft Internet Information Server (IIS).

```
<?php
  $path = iis_get_server_by_path('/path/to/root/folder/')
?>
```

This extension is available for Windows only.

See also IIS Administration.

Specs

Short name	Extensions/Extiis
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.397 ext/imagick

Extension Imagick for PHP.

Imagick is a native php extension to create and modify images using the ImageMagick API.

```
<?php

header('Content-type: image/jpeg');

$image = new Imagick('image.jpg');

// If 0 is provided as a width or height parameter,
// aspect ratio is maintained
$image->thumbnailImage(100, 0);

echo $image;

?>
```

See also Imagick for PHP and Imagick.

Specs

Short name	Extensions/Extimagick
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.398 ext/imap

Extension ext/imap.

This extension operate with the IMAP protocol, as well as the NNTP, POP3 and local mailbox access methods.

```
<?php
$mbox = imap_open('{imap.example.org}', 'username', 'password', OP_HALFOPEN)
      or die('can't connect: ' . imap_last_error());

$list = imap_list($mbox, '{imap.example.org}', '*');
if (is_array($list)) {
    foreach ($list as $val) {
        echo imap_utf7_decode($val) . PHP_EOL;
    }
} else {
    echo 'imap_list failed: ' . imap_last_error() . PHP_EOL;
}

imap_close($mbox);
?>
```

See also IMAP.

Specs

Short name	Extensions/Extimap
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.399 ext/info

PHP Options and Information.

These functions enable you to get a lot of information about PHP itself, e.g. runtime configuration, loaded extensions, version and much more.

```
<?php
/*
Our php.ini contains the following settings:

display_errors = On
register_globals = Off
post_max_size = 8M
*/

echo 'display_errors = ' . ini_get('display_errors') . "\n";
echo 'register_globals = ' . ini_get('register_globals') . "\n";
echo 'post_max_size = ' . ini_get('post_max_size') . "\n";
echo 'post_max_size+1 = ' . (ini_get('post_max_size')+1) . "\n";
echo 'post_max_size in bytes = ' . return_bytes(ini_get('post_max_size'));

function return_bytes($val) {
    $val = trim($val);
    $last = strtolower($val[strlen($val)-1]);
    switch($last) {
        // The 'G' modifier is available since PHP 5.1.0
        case 'g':
            $val *= 1024;
        case 'm':
            $val *= 1024;
        case 'k':
            $val *= 1024;
    }

    return $val;
}

?>
```

See also PHP Options And Information.

Specs

Short name	Extensions/Extinfo
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.400 ext/inotify

Extension inotify.

The Inotify extension gives access to the Linux kernel subsystem that acts to extend filesystems to notice changes to the filesystem, and report those changes to applications.

```
<?php
// Open an inotify instance
$fd = inotify_init();

// Watch __FILE__ for metadata changes (e.g. mtime)
$watch_descriptor = inotify_add_watch($fd, __FILE__, IN_ATTRIB);

// generate an event
touch(__FILE__);

// Read events
$events = inotify_read($fd);
print_r($events);

// The following methods allows to use inotify functions without blocking on inotify_
↪read():

// - Using stream_select() on $fd:
$read = array($fd);
$write = null;
$except = null;
stream_select($read,$write,$except,0);

// - Using stream_set_blocking() on $fd
stream_set_blocking($fd, 0);
inotify_read($fd); // Does no block, and return false if no events are pending

// - Using inotify_queue_len() to check if event queue is not empty
$queue_len = inotify_queue_len($fd); // If > 0, inotify_read() will not block

// Stop watching __FILE__ for metadata changes
inotify_rm_watch($fd, $watch_descriptor);

// Close the inotify instance
// This may have closed all watches if this was not already done
fclose($fd);

?>
```

See also ext/inotify manual and inotify.

Specs

Short name	Extensions/Extinotify
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.401 ext/intl

Extension international.

Internationalization extension (further is referred as Intl) is a wrapper for ICU library, enabling PHP programmers to perform various locale-aware operations including but not limited to formatting, transliteration, encoding conversion, calendar operations, UCA-conformant collation, locating text boundaries and working with locale identifiers, timezones and graphemes.

```
<?php
$coll = new Collator('en_US');
$al   = $coll->getLocale(Locale::ACTUAL_LOCALE);
echo Actual locale: $al\n;

$formatter = new NumberFormatter('en_US', NumberFormatter::DECIMAL);
echo $formatter->format(1234567);
?>
```

See also Internationalization Functions.

Specs

Short name	Extensions/Extintl
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.402 ext/json

Extension JSON.

This extension implements the JavaScript Object Notation (JSON) data-interchange format. PHP implements a superset of JSON as specified in the original RFC 7159.

```
<?php
$arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);

echo json_encode($arr);
?>
```

See also JavaScript Object Notation and JSON.

Specs

Short name	Extensions/Extjson
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.403 ext/judy

The Judy extension.

PHP Judy is a PECL extension for the Judy C library implementing dynamic sparse arrays.

```
<?php
$judy = new Judy(Judy::BITSET);
if ($judy->getType() === judy_type($judy) &&
    $judy->getType() === Judy::BITSET) {
    echo 'Judy BITSET type OK'.PHP_EOL;
} else {
    echo 'Judy BITSET type check fail'.PHP_EOL;
}
unset($judy);
?>
```

See also php-judy.

Specs

Short name	Extensions/Extjudy
Rulesets	*CE*
Exakt since	0.11.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.404 ext/kdm5

Extension kdm5 : Kerberos V .

These package allows you to access Kerberos V administration servers. You can create, modify, and delete Kerberos V principals and policies.

```
<?php
    // Extracted from the PHP Manual
  $handle = kadm5_init_with_password(afs-1, GONICUS.LOCAL, admin/admin, password);

  print <h1>get_principals</h1>\n;
  $principals = kadm5_get_principals($handle);
  for( $i=0; $i<count($principals); $i++)
      print $principals[$i]<br>\n;

  print <h1>get_policies</h1>\n;
  $policies = kadm5_get_policies($handle);
  for( $i=0; $i<count($policies); $i++)
      print $policies[$i]<br>\n;

  print <h1>get_principal burbach@GONICUS.LOCAL</h1>\n;

  $options = kadm5_get_principal($handle, burbach@GONICUS.LOCAL );
  $keys = array_keys($options);
  for( $i=0; $i<count($keys); $i++) {
    $value = $options[$keys[$i]];
    print $keys[$i]: $value<br>\n;
  }

  $options = array(KADM5_PRINC_EXPIRE_TIME => 0);
  kadm5_modify_principal($handle, burbach@GONICUS.LOCAL, $options);

  kadm5_destroy($handle);
?>
```

See also Kerberos V and Kerberos: The Network Authentication Protocol.

Specs

Short name	Extensions/Extkdm5
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.405 ext/lapack

Extension Lapack. LAPACK provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems.

```
<?php

$a = array(
    array( 1.44,  -7.84,  -4.39,   4.53),
```

(continues on next page)

(continued from previous page)

```
    array(-9.96,  -0.28,  -3.24,   3.83),
    array(-7.55,   3.24,   6.27,  -6.64),
    array( 8.34,   8.09,   5.28,   2.06),
    array( 7.08,   2.52,   0.74,  -2.47),
    array(-5.45,  -5.70,  -1.19,   4.70),
);

$b = array(
    array( 8.58,   9.35),
    array( 8.26,  -4.43),
    array( 8.48,  -0.70),
    array(-5.28,  -0.26),
    array( 5.72,  -7.36),
    array( 8.93,  -2.52),
);

$result = Lapack::leastSquaresByFactorisation($a, $b);
?>
```

See also Lapack <https://www.php.net/manual/en/book.‘lapack.php>‘_ and php-‘lapack <https://github.com/ianbarber/php-lapack>‘_.

Specs

Short name	Extensions/Extlapack
Rulesets	*CE*
Exakt since	0.12.2
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.406 ext/ldap

Extension ext/ldap.

LDAP is the Lightweight Directory Access Protocol, and is a protocol used to access ‘Directory Servers’. The Directory is a special kind of database that holds information in a tree structure.

```
<?php
// basic sequence with LDAP is connect, bind, search, interpret search
// result, close connection

echo '<h3>LDAP query test</h3>';
echo 'Connecting ...';
$ds=ldap_connect('localhost');  // must be a valid LDAP server!
echo 'connect result is ' . $ds . '<br />';

if ($ds) {
    echo 'Binding ...';
    $r=ldap_bind($ds);     // this is an 'anonymous' bind, typically
                           // read-only access
```

(continues on next page)

(continued from previous page)

```
    echo 'Bind result is ' . $r . '<br />';

    echo 'Searching for (sn=S*) ...';
    // Search surname entry
    $sr=ldap_search($ds, 'o=My Company, c=US', 'sn=S*');
    echo 'Search result is ' . $sr . '<br />';

    echo 'Number of entries returned is ' . ldap_count_entries($ds, $sr) . '<br />';

    echo 'Getting entries ...<p>';
    $info = ldap_get_entries($ds, $sr);
    echo 'Data for ' . $info['count'] . ' items returned:<p>';

    for ($i=0; $i<$info['count']; $i++) {
        echo 'dn is: ' . $info[$i]['dn'] . '<br />';
        echo 'first cn entry is: ' . $info[$i]['cn'][0] . '<br />';
        echo 'first email entry is: ' . $info[$i]['mail'][0] . '<br /><hr />';
    }

    echo 'Closing connection';
    ldap_close($ds);

} else {
    echo '<h4>Unable to connect to LDAP server</h4>';
}
?>
```

See also Lightweight ‘Directory Access Protocol <https://www.php.net/manual/en/book.ldap.php>‘_.

Specs

Short name	Extensions/Extldap
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.407 ext/leveldb

PHP Binding for LevelDB.

LevelDB is a fast key-value storage library written at Google that provides an ordered mapping from string keys to string values.

```
<?php

$db = new LevelDB($leveldb_path);

$batch = new LevelDBWriteBatch();
$batch->set('batch_foo', 'batch_bar');
$batch->put('batch_foo2', 'batch_bar2');
```

(continues on next page)

(continued from previous page)

```
$batch->delete('batch_foo');

$db->write($batch);

$batch->clear();
$batch->delete('batch_foo2');
$batch->set('batch_foo', 'batch again');

?>
```

See also ext/leveldb on Github and Leveldb.

Specs

Short name	Extensions/Extleveldb
Rulesets	*CE*
Exakt since	1.1.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.408 ext/libevent

Extension libevent.

Libevent is a library that provides a mechanism to execute a callback function when a specific event occurs on a file descriptor or after a timeout has been reached.

```
<?php

function print_line($fd, $events, $arg)
{
    static $max_requests = 0;

    $max_requests++;

    if ($max_requests == 10) {
        // exit loop after 10 writes
        event_base_loopexit($arg[1]);
    }

    // print the line
    echo  fgets($fd);
}

// create base and event
$base = event_base_new();
$event = event_new();

$fd = STDIN;

// set event flags
```

(continues on next page)

(continued from previous page)

```
event_set($event, $fd, EV_READ | EV_PERSIST, 'print_line', array($event, $base));
// set event base
event_base_set($event, $base);

// enable event
event_add($event);
// start event loop
event_base_loop($base);

?>
```

See also libevent and Libevent ext.

Specs

Short name	Extensions/Extlibevent
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.409 ext/libsodium

Extension for libsodium : in PECL until PHP 7.2, and in core ever since.

The Sodium crypto library (libsodium) is a modern, easy-to-use software library for encryption, decryption, signatures, password hashing and more.

Sodium supports a variety of compilers and operating systems, including Windows (with MinGW or Visual Studio, x86 and x64), iOS and Android.

The design choices emphasize security, and “magic constants” have clear rationales.

```
<?php
// Example from the docs : https://paragonie.com/book/pecl-libsodium/read/06-hashing.
↪md#crypto-generichash

// Fast, unkeyed hash function.
// Can be used as a secure replacement for MD5
$h = \Sodium\crypto_generichash('msg');

// Fast, keyed hash function.
// The key can be of any length between \Sodium\CRYPTO_GENERICHASH_KEYBYTES_MIN
// and \Sodium\CRYPTO_GENERICHASH_KEYBYTES_MAX, in bytes.
// \Sodium\CRYPTO_GENERICHASH_KEYBYTES is the recommended length.
$h = \Sodium\crypto_generichash('msg', $key);

// Fast, keyed hash function, with user-chosen output length, in bytes.
// Output length can be between \Sodium\CRYPTO_GENERICHASH_BYTES_MIN and
// \Sodium\CRYPTO_GENERICHASH_BYTES_MAX.
// \Sodium\CRYPTO_GENERICHASH_BYTES is the default length.
```

(continues on next page)

(continued from previous page)

```
$h = \Sodium\crypto_generichash('msg', $key, 64);

?>
```

See also PHP extension for libsodium and Using Libsodium in PHP Projects.

Specs

Short name	Extensions/Extlibsodium
Rulesets	*CE*
Exakt since	0.10.2
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.410 ext/libxml

Extension libxml.

These functions/constants are available as of PHP 5.1.0, and the following core extensions rely on this libxml extension: DOM, libxml, SimpleXML, SOAP, WDDX, XSL, XML, XMLReader, XMLRPC and XMLWriter.

```
<?php

// $xmlstr is a string, containing a XML document.

$doc = simplexml_load_string($xmlstr);
$xml = explode(PHP_EOL, $xmlstr);

if ($doc === false) {
    $errors = libxml_get_errors();

    foreach ($errors as $error) {
        echo display_xml_error($error, $xml);
    }

    libxml_clear_errors();
}


function display_xml_error($error, $xml)
{
    $return  = $xml[$error->line - 1] . PHP_EOL;
    $return .= str_repeat('-', $error->column) . '^'.PHP_EOL;

    switch ($error->level) {
        case LIBXML_ERR_WARNING:
            $return .= 'Warning ',$error->code.': ';
            break;
         case LIBXML_ERR_ERROR:
            $return .= 'Error '.$error->code.': ';
            break;
```

(continues on next page)

(continued from previous page)

```
        case LIBXML_ERR_FATAL:
            $return .= 'Fatal Error '.$error->code.': ';
            break;
    }

    $return .= trim($error->message) .
               PHP_EOL.'  Line: '.$error->line .
               PHP_EOL.'  Column: '.$error->column;

    if ($error->file) {
        $return .= "\n  File: $error->file";
    }

    return $return.PHP_EOL.PHP_EOL.'--------------------------------------------'.PHP_
↪EOL.PHP_EOL;
}

?>
```

See also libxml.

Specs

Short name	Extensions/Extlibxml
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.411 ext/lua

Extension Lua.

'Lua is a powerful, fast, light-weight, embeddable scripting language.' This extension embeds the lua interpreter and offers an OO-API to lua variables and functions.

```
<?php
$lua = new Lua();
$lua->eval(<<<CODE
    print(2);
CODE
);
?>
```

See also ext/lua manual and LUA

Specs

Short name	Extensions/Extlua
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.412 ext/lzf

Extension LZF.

LZF is a very fast compression algorithm, ideal for saving space with only slight speed cost. It can be optimized for speed or space at the time of compilation.

```
<?php
$compressed = lzf_compress(This is test of LZF extension);

echo base64_encode($compressed);
?>
```

See also lzf and liblzf.

Specs

Short name	Extensions/Extlzf
Rulesets	*CE*
Exakt since	1.3.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.413 ext/mail

Extension for mail.

The mail() function allows you to send mail.

```
<?php
// The message
$message = "Line 1\r\nLine 2\r\nLine 3";

// In case any of our lines are larger than 70 characters, we should use wordwrap()
$message = wordwrap($message, 70, "\r\n");

// Send
mail('caffeinated@example.com', 'My Subject', $message);
?>
```

See also Mail related functions.

Specs

Short name	Extensions/Extmail
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.414 ext/mailparse

Extension mailparse.

Mailparse is an extension for parsing and working with email messages. It can deal with RFC 822 (MIME) and RFC 2045 (MIME) compliant messages.

```
<?php

$mail = mailparse_msg_create();
mailparse_msg_parse($mail, $mailInString);
$parts = mailparse_msg_get_structure($mail);

foreach($parts as $part) {
    $section = mailparse_msg_get_part($mail, $part);
    $info = mailparse_msg_get_part_data($section);
}

?>
```

See also Mailparse.

Specs

Short name	Extensions/Extmailparse
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.415 ext/math

Core functions that provides math standard functions.

This is not a real extension : it is a documentation section, that helps sorting the functions.

```
<?php
echo decbin(12) . PHP_EOL;
echo decbin(26);
?>
```

See also Mathematical Functions.

Specs

Short name	Extensions/Extmath
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.416 ext/mbstring

Extension `ext/mbstring`.

`mbstring` provides multibyte specific string functions that help you deal with multibyte encodings in PHP.

```
<?php
/* Convert internal character encoding to SJIS */
$str = mb_convert_encoding($str, "SJIS");

/* Convert EUC-JP to UTF-7 */
$str = mb_convert_encoding($str, "UTF-7", "EUC-JP");

/* Auto detect encoding from JIS, eucjp-win, sjis-win, then convert str to UCS-2LE */
$str = mb_convert_encoding($str, "UCS-2LE", "JIS, eucjp-win, sjis-win");

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
$str = mb_convert_encoding($str, "EUC-JP", "auto");
?>
```

See also Mbstring.

Specs

Short name	Extensions/Extmbstring
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.417 ext/mcrypt

Extension for mcrypt.

This extension has been deprecated as of PHP 7.1.0 and moved to PECL as of PHP 7.2.0.

This is an interface to the mcrypt library, which supports a wide variety of block algorithms such as DES, TripleDES, Blowfish (default), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 and GOST in CBC, OFB, CFB and ECB cipher modes. Additionally, it supports RC6 and IDEA which are considered 'non-free'. CFB/OFB are 8bit by default.

```
<?php
    # --- ENCRYPTION ---

    # the key should be random binary, use scrypt, bcrypt or PBKDF2 to
    # convert a string into a key
    # key is specified using hexadecimal
    $key = pack('H*',
↪'bcb04b7e103a0cd8b54763051cef08bc55abe029fdebae5e1d417e2ffb2a00a3');

    # show key size use either 16, 24 or 32 byte keys for AES-128, 192
    # and 256 respectively
    $key_size =  strlen($key);
    echo 'Key size: ' . $key_size . PHP_EOL;

    $plaintext = 'This string was AES-256 / CBC / ZeroBytePadding encrypted.';

    # create a random IV to use with CBC encoding
    $iv_size = mcrypt_get_iv_size(MCRYPT_RIJNDAEL_128, MCRYPT_MODE_CBC);
    $iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);

    # creates a cipher text compatible with AES (Rijndael block size = 128)
    # to keep the text confidential
    # only suitable for encoded input that never ends with value 00h
    # (because of default zero padding)
    $ciphertext = mcrypt_encrypt(MCRYPT_RIJNDAEL_128, $key,
                                 $plaintext, MCRYPT_MODE_CBC, $iv);

    # prepend the IV for it to be available for decryption
    $ciphertext = $iv . $ciphertext;

    # encode the resulting cipher text so it can be represented by a string
    $ciphertext_base64 = base64_encode($ciphertext);

    echo  $ciphertext_base64 . PHP_EOL;

    # === WARNING ===

    # Resulting cipher text has no integrity or authenticity added
    # and is not protected against padding oracle attacks.

    # --- DECRYPTION ---

    $ciphertext_dec = base64_decode($ciphertext_base64);

    # retrieves the IV, iv_size should be created using mcrypt_get_iv_size()
    $iv_dec = substr($ciphertext_dec, 0, $iv_size);
```

(continues on next page)

(continued from previous page)

```
    # retrieves the cipher text (everything except the $iv_size in the front)
    $ciphertext_dec = substr($ciphertext_dec, $iv_size);

    # may remove 00h valued characters from end of plain text
    $plaintext_dec = mcrypt_decrypt(MCRYPT_RIJNDAEL_128, $key,
                                    $ciphertext_dec, MCRYPT_MODE_CBC, $iv_dec);

    echo  $plaintext_dec . PHP_EOL;
?>
```

See also extension mcrypt and mcrypt.

Specs

Short name	Extensions/Extmcrypt
Rulesets	*CE*, *CompatibilityPHP71*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.418 ext/memcache

Extension Memcache.

Memcache module provides handy procedural and object oriented interface to memcached, highly effective caching daemon, which was especially designed to decrease database load in dynamic web applications.

```
<?php

$memcache = new Memcache;
$memcache->connect('localhost', 11211) or die ('Could not connect');

$version = $memcache->getVersion();
echo 'Server\'s version: '.$version.'<br/>';

$tmp_object = new stdClass;
$tmp_object->str_attr = 'test';
$tmp_object->int_attr = 123;

$memcache->set('key', $tmp_object, false, 10) or die ('Failed to save data at the␣
↪server');
echo 'Store data in the cache (data will expire in 10 seconds)<br/>';

$get_result = $memcache->get('key');
echo 'Data from the cache:<br/>';

var_dump($get_result);

?>
```

See also Memcache on PHP and memcached.

Specs

Short name	Extensions/Extmemcache
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.419 ext/memcached

Extension ext-memcached.

This extension uses the libmemcached library to provide an API for communicating with memcached servers. It also provides a session handler (*memcached*).

```
<?php
$m = new Memcached();
$m->addServer('localhost', 11211);

$m->set('foo', 100);
var_dump($m->get('foo'));
?>
```

See also ext/memcached manual and memcached.

Suggestions

-

Specs

Short name	Extensions/Extmemcached
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.420 ext/mhash

Extension mhash (obsolete since PHP 5.3.0).

This extension provides functions, intended to work with mhash.

```
<?php
$input = 'what do ya want for nothing?';
$hash = mhash(MHASH_MD5, $input);
echo 'The hash is ' . bin2hex($hash) . '<br />'.PHP_EOL;
$hash = mhash(MHASH_MD5, $input, 'Jefe');
echo 'The hmac is ' . bin2hex($hash) . '<br />'.PHP_EOL;
?>
```

See also Extension mhash.

Specs

Short name	Extensions/Extmhash
Rulesets	*CE*, *CompatibilityPHP54*
Exakt since	0.9.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.421 ext/ming

Extension ext/ming, to create swf files with PHP.

Ming is an open-source (LGPL) library which allows you to create SWF ('Flash') format movies.

```
<?php
  $s = new SWFShape();
  $f = $s->addFill(0xff, 0, 0);
  $s->setRightFill($f);

  $s->movePenTo(-500, -500);
  $s->drawLineTo(500, -500);
  $s->drawLineTo(500, 500);
  $s->drawLineTo(-500, 500);
  $s->drawLineTo(-500, -500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $i->setDepth(1);
  $p->nextFrame();

  for ($n=0; $n<5; ++$n) {
    $i->rotate(-15);
    $p->nextFrame();
  }

  $m = new SWFMovie();
  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(6000, 4000);

  $i = $m->add($p);
  $i->setDepth(1);
```

(continues on next page)

(continued from previous page)

```
  $i->moveTo(-500,2000);
  $i->setName('box');

  $m->add(new SWFAction('/box.x += 3;'));
  $m->nextFrame();
  $m->add(new SWFAction('gotoFrame(0); play();'));
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

See also Ming (flash) and Ming.

Specs

Short name	Extensions/Extming
Rulesets	*CE*, *CompatibilityPHP53*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.422 ext/mongo

Extension MongoDB driver (legacy).

```
<?php

// connect
$m = new MongoClient();

// select a database
$db = $m->comedy;

// select a collection (analogous to a relational database\'s table)
$collection = $db->cartoons;

// add a record
$document = array( 'title' => 'Calvin and Hobbes', 'author' => 'Bill Watterson' );
$collection->insert($document);

// add another record, with a different 'shape'
$document = array( 'title' => 'XKCD', 'online' => true );
$collection->insert($document);

// find everything in the collection
$cursor = $collection->find();

// iterate through the results
foreach ($cursor as $document) {
```

(continues on next page)

(continued from previous page)

```
    echo $document['title'] . PHP_EOL;
}

?>
```

Note : this is not the MongoDB driver. This is the legacy extension.

See also ext/mongo manual and MongdDb.

Specs

Short name	Extensions/Extmongo
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.423 ext/mongodb

Extension MongoDb.

Do not mistake with extension Mongo, the previous version.

Mongodb driver supports both PHP and HHVM and is developed atop the libmongoc and libbson libraries.

```
<?php
require 'vendor/autoload.php'; // include Composer's autoloader

$client = new MongoDB\Client(mongodb://localhost:27017);
$collection = $client->demo->beers;

$result = $collection->insertOne( [ 'name' => 'Hinterland', 'brewery' => 'BrewDog' ]␣
↪);

echo Inserted with Object ID {$result->getInsertedId()};
?>
```

See also MongoDB driver.

Specs

Short name	Extensions/Extmongodb
Rulesets	*CE*
Exakt since	0.9.5
Php Version	7.0+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.424 ext/msgpack

Extension msgPack.

This extension provide API for communicating with MessagePack serialization.

```
<?php

    $serialized = msgpack_serialize(array('a' => true, 'b' => 4));
    $unserialized = msgpack_unserialize($serialized);

?>
```

See also msgpack for PHP and MessagePack.

Specs

Short name	Extensions/Extmsgpack
Rulesets	*CE*
Exakt since	1.3.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.425 ext/mssql

Extension MSSQL, Microsoft SQL Server.

These functions allow you to access MS SQL Server database.

```
<?php
// Connect to MSSQL
$link = mssql_connect('KALLESPC\SQLEXPRESS', 'sa', 'phpfi');

if (!$link || !mssql_select_db('php', $link)) {
    die('Unable to connect or select database!');
}

// Do a simple query, select the version of
// MSSQL and print it.
$version = mssql_query('SELECT @@VERSION');
$row = mssql_fetch_array($version);

echo $row[0];

// Clean up
mssql_free_result($version);
?>
```

See also Microsoft SQL Server and Microsoft PHP Driver for SQL Server.

Specs

Short name	Extensions/Extmssql
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.426 ext/mysql

Extension for MySQL (Original MySQL API).

This extension is deprecated as of PHP 5.5.0, and has been removed as of PHP 7.0.0. Instead, either the mysqli or PDO_MySQL extension should be used. See also the MySQL API Overview for further help while choosing a MySQL API. .. code-block:: php <?php $result = mysql_query('SELECT * WHERE 1=1'); if (!$result) { die('Invalid query: ' . mysql_error()); } ?> See also Original MySQL API and MySQL.

Specs

Short name	Extensions/Extmysql
Rulesets	*CE*, *CompatibilityPHP55*
Exakt since	0.8.4
Php Version	7.0-
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.427 ext/mysqli

Extension mysqli for MySQL.

The mysqli extension allows you to access the functionality provided by MySQL 4.1 and above.

```
<?php
$mysqli = new mysqli('localhost', 'my_user', 'my_password', 'world');

/* check connection */
if (mysqli_connect_errno()) {
    printf('Connect failed: %s\n', mysqli_connect_error());
    exit();
}

$city = 'Amersfoort';

/* create a prepared statement */
if ($stmt = $mysqli->prepare('SELECT District FROM City WHERE Name=?')) {

    /* bind parameters for markers */
    $stmt->bind_param('s', $city);
```

(continues on next page)

(continued from previous page)

```
    /* execute query */
    $stmt->execute();

    /* bind result variables */
    $stmt->bind_result($district);

    /* fetch value */
    $stmt->fetch();

    printf('%s is in district %s\n', $city, $district);

    /* close statement */
    $stmt->close();
}

/* close connection */
$mysqli->close();
?>
```

See also MySQL Improved Extension <https://www.php.net/manual/en/book.'mysqli.php>'_ and MySQL.

Specs

Short name	Extensions/Extmysqli
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.428 ext/ncurses

Extension ncurses (CLI).

ncurses (new curses) is a free software emulation of curses in System V Rel 4.0 (and above).

```
<?php
ncurses_init();
ncurses_start_color();
ncurses_init_pair(1, NCURSES_COLOR_GREEN, NCURSES_COLOR_BLACK);
ncurses_init_pair(2, NCURSES_COLOR_RED,   NCURSES_COLOR_BLACK);
ncurses_init_pair(3, NCURSES_COLOR_WHITE, NCURSES_COLOR_BLACK);
ncurses_color_set(1);
ncurses_addstr('OK   ');
ncurses_color_set(3);
ncurses_addstr('Success!'.PHP_EOL);
ncurses_color_set(2);
ncurses_addstr('FAIL ');
ncurses_color_set(3);
ncurses_addstr('Success!'.PHP_EOL);
?>
```

See also Ncurses Terminal Screen Control and Ncurses.

Specs

Short name	Extensions/Extncurses
Rulesets	*CE*
Exakt since	0.9.2
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.429 ext/newt

Newt PHP CLI extension.

This is a PHP language extension for RedHat Newt library, a terminal-based window and widget library for writing applications with user friendly interface.

```
<?php
newt_init ();
newt_cls ();

newt_draw_root_text (0, 0, Test Mode Setup Utility 1.12);
newt_push_help_line (null);

newt_get_screen_size ($rows, $cols);

newt_open_window ($rows/2-17, $cols/2-10, 34, 17, Choose a Tool);

$form = newt_form ();

$list = newt_listbox (3, 2, 10);

foreach (array (
    Authentication configuration,
    Firewall configuration,
    Mouse configuration,
    Network configuration,
    Printer configuration,
    System services) as $l_item)
{
    newt_listbox_add_entry ($list, $l_item, $l_item);
}

$b1 = newt_button (5, 12, Run Tool);
$b2 = newt_button (21, 12, Quit);

newt_form_add_component ($form, $list);
newt_form_add_components ($form, array($b1, $b2));

newt_refresh ();
newt_run_form ($form);
```

(continues on next page)

(continued from previous page)

```
newt_pop_window ();
newt_pop_help_line ();
newt_finished ();
newt_form_destroy ($form);
?>
```

See also Newt.

Specs

Short name	Extensions/Extnewt
Rulesets	*CE*
Exakt since	0.9.2
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.430 ext/nsapi

NSAPI specific functions calls.

These functions are only available when running PHP as a NSAPI module in Netscape/iPlanet/Sun webservers.

```
<?php

// This scripts depends on ext/nsapi
if (ini_get('nsapi.read_timeout') < 60) {
    doSomething();
}

?>
```

See also Sun, iPlanet and Netscape servers on Sun Solaris.

Specs

Short name	Extensions/Extnsapi
Rulesets	*CE*
Exakt since	0.9.2
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.431 ext/ob

Extension Output Buffering Control.

The Output Control functions allow you to control when output is sent from the script.

```
<?php

ob_start();
echo Hello\n;

setcookie(cookiename, cookiedata);

ob_end_flush();

?>
```

See also Output Buffering Control.

Specs

Short name	Extensions/Extob
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.432 ext/oci8

Extension ext/oci8.

OCI8 gives access Oracle Database 12c, 11g, 10g, 9i and 8i.

```
<?php

$conn = oci_connect('hr', 'welcome', 'localhost/XE');
if (!$conn) {
    $e = oci_error();
    trigger_error(htmlentities($e['message'], ENT_QUOTES), E_USER_ERROR);
}

// Prepare the statement
$stid = oci_parse($conn, 'SELECT * FROM departments');
if (!$stid) {
    $e = oci_error($conn);
    trigger_error(htmlentities($e['message'], ENT_QUOTES), E_USER_ERROR);
}

// Perform the logic of the query
$r = oci_execute($stid);
if (!$r) {
    $e = oci_error($stid);
    trigger_error(htmlentities($e['message'], ENT_QUOTES), E_USER_ERROR);
}

// Fetch the results of the query
```

(continues on next page)

(continued from previous page)

```
print '<table border='1'>' . PHP_EOL;
while ($row = oci_fetch_array($stid, OCI_ASSOC+OCI_RETURN_NULLS)) {
    print '<tr>' . PHP_EOL;
    foreach ($row as $item) {
        print '    <td>' . ($item !== null ? htmlentities($item, ENT_QUOTES) : ' 
↪') . '</td>' . PHP_EOL;
    }
    print '</tr>' . PHP_EOL;
}
print '</table>' . PHP_EOL;

oci_free_statement($stid);
oci_close($conn);

?>
```

See also Oracle OCI8 and Oracle.

Specs

Short name	Extensions/Extoci8
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.433 ext/odbc

Extension ODBC.

In addition to normal ODBC support, the Unified ODBC functions in PHP allow you to access several databases that have borrowed the semantics of the ODBC API to implement their own API. Instead of maintaining multiple database drivers that were all nearly identical, these drivers have been unified into a single set of ODBC functions.

```
<?php
$a = 1;
$b = 2;
$c = 3;
$stmt    = odbc_prepare($conn, 'CALL myproc(?,?,?)');
$success = odbc_execute($stmt, array($a, $b, $c));
?>
```

See also ODBC (Unified), Unixodbc and IODBC.

Specs

Short name	Extensions/Extodbc
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.434 ext/opcache

Extension opcache.

OPcache improves PHP performance by storing precompiled script bytecode in shared memory, thereby removing the need for PHP to load and parse scripts on each request.

```
<?php

echo opcache_compile_file('/var/www/index.php');

print_r(opcache_get_status());

?>
```

See also OPcache functions.

Specs

Short name	Extensions/Extopcache
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.435 ext/opencensus

Extension PHP for OpenCensus :

A stats collection and distributed tracing framework.

```
<?php
opencensus_trace_begin('root', ['spanId' => '1234']);
opencensus_trace_add_annotation('foo');
opencensus_trace_begin('inner', []);
opencensus_trace_add_annotation('asdf', ['spanId' => '1234']);
opencensus_trace_add_annotation('abc');
opencensus_trace_finish();
opencensus_trace_finish();
```

(continues on next page)

(continued from previous page)

```
$traces = opencensus_trace_list();
echo Number of traces: . count($traces) . \n;
$span = $traces[0];
print_r($span->timeEvents());
$span2 = $traces[1];
print_r($span2->timeEvents());
?>
```

See also opencensus.

Specs

Short name	Extensions/Extopencensus
Rulesets	*CE*
Exakt since	1.1.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.436 ext/openssl

Extension Openssl.

This extension binds functions of `OpenSSL` library for symmetric and asymmetric encryption and decryption, `PBKDF2`, `PKCS7`, `PKCS12`, `X509` and other cryptographic operations. In addition to that it provides implementation of `TLS` streams.

```
<?php
// $data and $signature are assumed to contain the data and the signature

// fetch public key from certificate and ready it
$pubkeyid = openssl_pkey_get_public("file://src/openssl-0.9.6/demos/sign/cert.pem");

// state whether signature is okay or not
$ok = openssl_verify($data, $signature, $pubkeyid);
if ($ok == 1) {
    echo "good";
} elseif ($ok == 0) {
    echo "bad";
} else {
    echo "ugly, error checking signature";
}
// free the key from memory
openssl_free_key($pubkeyid);
?>
```

See also ext/OpenSSL and OpenSSL.

Specs

Short name	Extensions/Extopenssl
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.437 ext/parle

Extension Parser and Lexer.

The parle extension provides lexing and parsing facilities. The implementation is based on » Ben Hanson's libraries and requires a » C++14 capable compiler.

```
<?php

use Parle\{Token, Lexer, LexerException};

/* name => id */
$token = array(
        'EOI' => 0,
        'COMMA' => 1,
        'CRLF' => 2,
        'DECIMAL' => 3,
);
/* id => name */
$token_rev = array_flip($token);

$lex = new Lexer;
$lex->push("[\x2c]", $token['COMMA']);
$lex->push("[\r][\n]", $token['CRLF']);
$lex->push("[\d]+", $token['DECIMAL']);
$lex->build();

$in = 0,1,2\r\n3,42,5\r\n6,77,8\r\n;

$lex->consume($in);

do {
        $lex->advance();
        $tok = $lex->getToken();

        if (Token::UNKNOWN == $tok->id) {
                throw new LexerException('Unknown token "'.$tok->value.'" at offset '.
↪$tok->offset.'.');
        }

        echo 'TOKEN: ', $token_rev[$tok->id], PHP_EOL;
} while (Token::EOI != $tok->id);

?>
```

See also Parsing and Lexing.

Specs

Short name	Extensions/Extparle
Rulesets	*CE*
Exakt since	0.12.12
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.438 ext/parsekit

Extension Parsekit.

These functions allow runtime analysis of opcodes compiled from PHP scripts.

```
<?php
var_dump(parsekit_compile_file('hello_world.php', $errors, PARSEKIT_SIMPLE));
?>
```

See also Parsekit.

Specs

Short name	Extensions/Extparsekit
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.439 ext/password

Extension password.

The password hashing API provides an easy to use wrapper around crypt() and some other password hashing algorithms, to make it easy to create and manage passwords in a secure manner.

```
<?php
// See the password_hash() example to see where this came from.
$hash = '\$2y\$07$BCryptRequires22Chrcte/VlQH0piJtjXl.0t1XkA8pw9dMXTpOq';

if (password_verify('rasmuslerdorf', $hash)) {
    echo 'Password is valid!';
} else {
    echo 'Invalid password.';
}
?>
```

See also Password Hashing and crypt man page.

Specs

Short name	Extensions/Extpassword
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.440 ext/pcntl

Extension for process control.

Process Control support in PHP implements the Unix style of process creation, program execution, signal handling and process termination. Process Control should not be enabled within a web server environment and unexpected results may happen if any Process Control functions are used within a web server environment.

```
<?php
declare(ticks=1);

$pid = pcntl_fork();
if ($pid == -1) {
     die('could not fork');
} else if ($pid) {
     exit(); // we are the parent
} else {
     // we are the child
}

// detatch from the controlling terminal
if (posix_setsid() == -1) {
    die('could not detach from terminal');
}

// setup signal handlers
pcntl_signal(SIGTERM, 'sig_handler');
pcntl_signal(SIGHUP, 'sig_handler');

// loop forever performing tasks
while (1) {

    // do something interesting here

}

function sig_handler($signo)
{

     switch ($signo) {
         case SIGTERM:
             // handle shutdown tasks
             exit;
             break;
         case SIGHUP:
```

(continues on next page)

(continued from previous page)

```
                // handle restart tasks
                break;
            default:
                // handle all other signals
        }

}

?>
```

See also Process Control.

Specs

Short name	Extensions/Extpcntl
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.441 ext/pcov

CodeCoverage compatible driver for PHP

A self contained CodeCoverage compatible driver for PHP7

```
<?php
\pcov\start();
$d = [];
for ($i = 0; $i < 10; $i++) {
     $d[] = $i * 42;
}
\pcov\stop();
var_dump(\pcov\collect());
?>
```

See also PCOV.

Specs

Short name	Extensions/Extpcov
Rulesets	*CE*
Exakt since	1.6.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.442 ext/pcre

Extension ext/pcre. PCRE stands for Perl Compatible Regular Expression. It is a standard PHP extension.

```
<?php

$zip_code = $_GET['zip'];

// Canadian Zip code H2M 3J1
$zip_ca = '/^([a-zA-Z]\d[a-zA-Z])\ {0,1}(\d[a-zA-Z]\d)$/';

// French Zip code  75017
$zip_fr = '/^\d{5}$/';

// Chinese Zip code  590615
$zip_cn = '/^\d{6}$/';

var_dump(preg_match($_GET['zip']));

?>
```

See also Regular Expressions (Perl-Compatible).

Specs

Short name	Extensions/Extpcre
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.443 ext/pdo

Generic extension PDO.

The PHP Data Objects (PDO) extension defines a lightweight, consistent interface for accessing databases in PHP.

```
<?php
/* Execute a prepared statement by passing an array of values */
$sql = 'SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour';
$sth = $dbh->prepare($sql, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
$sth->execute(array(':calories' => 150, ':colour' => 'red'));
$red = $sth->fetchAll();
$sth->execute(array(':calories' => 175, ':colour' => 'yellow'));
$yellow = $sth->fetchAll();
?>
```

See also PHP Data Object.

Specs

Short name	Extensions/Extpdo
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.444 ext/pgsql

Extension PostGreSQL.

PostgreSQL is an open source descendant of this original Berkeley code. It provides SQL92/SQL99 language support, transactions, referential integrity, stored procedures and type extensibility.

```
<?php
// Connect to a database named 'mary'
$dbconn = pg_connect('dbname=mary');

// Prepare a query for execution
$result = pg_prepare($dbconn, 'my_query', 'SELECT * FROM shops WHERE name = \$1');

// Execute the prepared query.  Note that it is not necessary to escape
// the string 'Joe's Widgets' in any way
$result = pg_execute($dbconn, 'my_query', array('Joe\'s Widgets'));

// Execute the same prepared query, this time with a different parameter
$result = pg_execute($dbconn, 'my_query', array('Clothes Clothes Clothes'));

?>
```

See also PostgreSQL and PostgreSQL: The world's most advanced open source database.

Specs

Short name	Extensions/Extpgsql
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.445 ext/phalcon

Extension Phalcon : High Performance PHP Framework.

Phalcon's autoload examples from the docs : Tutorial 1: Let's learn by example

```
<?php

use Phalcon\Loader;

// ...

$loader = new Loader();

$loader->registerDirs(
    [
        ../app/controllers/,
        ../app/models/,
    ]
);

$loader->register();

?>
```

See also PhalconPHP.

Specs

Short name	Extensions/Extphalcon
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.446 ext/phar

Extension phar.

The phar extension provides a way to put entire PHP applications into a single file called a `phar` (PHP Archive) for easy distribution and installation.

```
<?php
try {
    $p = new Phar('/path/to/my.phar', 0, 'my.phar');
    $p['myfile.txt'] = 'hi';
    $file = $p['myfile.txt'];
    var_dump($file->isCompressed(Phar::BZ2));
    $p['myfile.txt']->compress(Phar::BZ2);
    var_dump($file->isCompressed(Phar::BZ2));
} catch (Exception $e) {
    echo 'Create/modify operations on my.phar failed: ', $e;
}
?>
```

See also phar.

Specs

Short name	Extensions/Extphar
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.447 ext/posix

Extension POSIX.

Ext/posix contains an interface to those functions defined in the IEEE 1003.1 (POSIX.1) standards document which are not accessible through other means.

```
<?php
posix_kill(999459,SIGKILL);
echo 'Your error returned was '.posix_get_last_error(); //Your error was ___
?>
```

See also 1003.1-2008 - IEEE Standard for Information Technology - Portable Operating System Interface (POSIX(R)).

Specs

Short name	Extensions/Extposix
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.448 ext/proctitle

Extension proctitle.

This extension allows changing the current process', and thread, name on Linux and *BSD systems. This is useful when using pcntl_fork() to identify running processes in process list

```
<?php
setproctitle('myscript');
?>
```

See also proctitle.

Specs

Short name	Extensions/Extproctitle
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.449 ext/pspell

Extension pspell.

These functions allow you to check the spelling of a word and offer suggestions.

```
<?php
$pspell_link = pspell_new('en');

if (pspell_check($pspell_link, 'testt')) {
    echo 'This is a valid spelling';
} else {
    echo 'Sorry, wrong spelling';
}
?>
```

See also Pspell and pspell.

Specs

Short name	Extensions/Extpspell
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.450 ext/psr

Extension PSR : PHP Standards Recommendations.

This PHP extension provides the interfaces from the PSR standards as established by the PHP-FIG group. You can use interfaces provided by this extension in another extension easily - see this example.

Currently supported PSR :

- PSR-3 - *psr/http-message*
- PSR-11 - *psr/container*
- PSR-13 - *psr/link*

- PSR-15 - *psr/http-server*
- PSR-16 - *psr/simple-cache*
- PSR-17 - *psr/http-factory*

```
<?php
// Example from the tests, for Cache (PSR-6)
use Psr\Cache\CacheException;
class MyCacheException extends Exception implements CacheException {}
$ex = new MyCacheException('test');
var_dump($ex instanceof CacheException);
var_dump($ex instanceof Exception);
try {
    throw $ex;
} catch( CacheException $e ) {
    var_dump($e->getMessage());
}
?>
```

See also php-psr and PHP-FIG.

Specs

Short name	Extensions/Extpsr
Rulesets	*CE*
Exakt since	1.5.2
Php Version	7.0+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.451 ext/rar

Extension RAR.

Rar is a powerful and effective archiver created by Eugene Roshal. This extension gives you possibility to read Rar archives but doesn't support writing Rar archives, because this is not supported by the UnRar library and is directly prohibited by its license.

```
<?php

$arch = RarArchive::open(example.rar);
if ($arch === FALSE)
    die(Cannot open example.rar);

$entries = $arch->getEntries();
if ($entries === FALSE)
    die(Cannot retrieve entries);


?>
```

See also Rar archiving and rarlabs.

Specs

Short name	Extensions/Extrar
Rulesets	*CE*
Exakt since	0.8.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.452 ext/rdkafka

Extension for RDkafka.

PHP-rdkafka is a thin librdkafka binding providing a working PHP 5 / PHP 7 Kafka 0.8 / 0.9 / 0.10 client.

```
<?php

$rk = new RdKafka\Producer();
$rk->setLogLevel(LOG_DEBUG);
$rk->addBrokers(10.0.0.1,10.0.0.2);

?>
```

See also Kafka client for PHP.

Specs

Short name	Extensions/Extrdkafka
Rulesets	*CE*
Exakt since	0.12.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.453 ext/readline

Extension readline.

The readline functions implement an interface to the GNU Readline library. These are functions that provide editable command lines.

```
<?php
//get 3 commands from user
for ($i=0; $i < 3; $i++) {
        $line = readline("Command: ");
        readline_add_history($line);
}

//dump history
```

(continues on next page)

(continued from previous page)

```
print_r(readline_list_history());

//dump variables
print_r(readline_info());
?>
```

See also ext/readline and The GNU Readline Library.

Specs

Short name	Extensions/Extreadline
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.454 ext/recode

Extension GNU Recode.

This module contains an interface to the GNU Recode library. The GNU Recode library converts files between various coded character sets and surface encodings.

```
<?php
echo recode_string('us..flat', 'The following character has a diacritical mark: á');
?>
```

This extension is not available on Windows.

See also ext/recode and Recode.

Specs

Short name	Extensions/Extrecode
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.455 ext/redis

Extension ext/redis.

The phpredis extension provides an API for communicating with the Redis key-value store.

```
<?php

$redis = new Redis();
$redis->connect('127.0.0.1', 6379);

$redis->setOption(Redis::OPT_SERIALIZER, Redis::SERIALIZER_NONE);     // don't␣
↪serialize data
$redis->setOption(Redis::OPT_SERIALIZER, Redis::SERIALIZER_PHP);      // use built-in␣
↪serialize/unserialize
$redis->setOption(Redis::OPT_SERIALIZER, Redis::SERIALIZER_IGBINARY);         // use␣
↪igBinary serialize/unserialize

$redis->setOption(Redis::OPT_PREFIX, 'myAppName:');  // use custom prefix on all keys

/* Options for the SCAN family of commands, indicating whether to abstract
   empty results from the user.  If set to SCAN_NORETRY (the default), phpredis
   will just issue one SCAN command at a time, sometimes returning an empty
   array of results.  If set to SCAN_RETRY, phpredis will retry the scan command
   until keys come back OR Redis returns an iterator of zero
*/
$redis->setOption(Redis::OPT_SCAN, Redis::SCAN_NORETRY);
$redis->setOption(Redis::OPT_SCAN, Redis::SCAN_RETRY);
?>
```

See also A PHP extension for ‘Redis <https://github.com/phpredis/phpredis/>‘_ and Redis.

Specs

Short name	Extensions/Extredis
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.456 ext/reflection

Extension Reflection.

PHP comes with a complete reflection API that adds the ability to reverse-engineer classes, interfaces, functions, methods and extensions. Additionally, the reflection API offers ways to retrieve doc comments for functions, classes and methods.

```
<?php
/**
 * A simple counter
 *
 * @return    int
 */
function counter1()
{
    static $c = 0;
```

(continues on next page)

(continued from previous page)

```
    return ++$c;
}

/**
 * Another simple counter
 *
 * @return    int
 */
$counter2 = function()
{
    static $d = 0;
    return ++$d;

};

function dumpReflectionFunction($func)
{
    // Print out basic information
    printf(
        PHP_EOL.'===> The %s function '%s''.PHP_EOL.
        '     declared in %s'.PHP_EOL.
        '     lines %d to %d'.PHP_EOL,
        $func->isInternal() ? 'internal' : 'user-defined',
        $func->getName(),
        $func->getFileName(),
        $func->getStartLine(),
        $func->getEndline()
    );

    // Print documentation comment
    printf('---> Documentation:'.PHP_EOL.' %s',PHP_EOL, var_export($func->
↪getDocComment(), 1));

    // Print static variables if existant
    if ($statics = $func->getStaticVariables())
    {
        printf('---> Static variables: %s',PHP_EOL, var_export($statics, 1));
    }
}

// Create an instance of the ReflectionFunction class
dumpReflectionFunction(new ReflectionFunction('counter1'));
dumpReflectionFunction(new ReflectionFunction($counter2));
?>
```

See also Reflection.

Specs

Short name	Extensions/Extreflection
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.457 ext/runkit

Extension Runkit.

The runkit extension provides means to modify constants, user-defined functions, and user-defined classes. It also provides for custom superglobal variables and embeddable sub-interpreters via sandboxing.

```
<?php
class Example {
    function foo() {
        echo 'foo!'.PHP_EOL;
    }
}

// create an Example object
$e = new Example();

// Add a new public method
runkit_method_add(
    'Example',
    'add',
    '$num1, $num2',
    'return $num1 + $num2;',
    RUNKIT_ACC_PUBLIC
);

// add 12 + 4
echo $e->add(12, 4);
?>
```

See also runkit.

Specs

Short name	Extensions/Extrunkit
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.458 ext/sdl

Extensions ext/sdl.

Simple DirectMedia Layer (SDL) is a cross-platform software development library designed to provide a hardware abstraction layer for computer multimedia hardware components.

```
<?php
/**
 * Example of how to change screen properties such as title, icon or state using the␣
↪PHP-SDL extension.
 *
 * @author Santiago Lizardo <santiagolizardo@php.net>
 */
require 'common.php';
SDL_Init( SDL_INIT_VIDEO );
$screen = SDL_SetVideoMode( 640, 480, 16, SDL_HWSURFACE );
if( null == $screen )
{
     fprintf( STDERR, 'Error: %s' . PHP_EOL, SDL_GetError() );
}
for( $i = 3; $i > 0; $i-- )
{
     SDL_WM_SetCaption( Switching to fullscreen mode in $i seconds..., null );
     SDL_Delay( 1000 );
}
SDL_WM_ToggleFullscreen( $screen );
SDL_Delay( 3000 );
SDL_WM_ToggleFullscreen( $screen );
SDL_WM_SetCaption( Back from fullscreen mode. Quitting in 2 seconds..., null );
SDL_Delay( 2000 );
SDL_FreeSurface( $screen );
SDL_Quit();

?>
```

See also phpsdl, Simple DirectMedia Layer and About SDL.

Suggestions

-

Specs

Short name	Extensions/Extsdl
Rulesets	*CE*
Exakt since	1.5.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.459 ext/seaslog

Extension Seaslog.

An effective,fast,stable log extension for PHP.

```
<?php
$basePath_1 = SeasLog::getBasePath();

SeasLog::setBasePath('/log/base_test');
$basePath_2 = SeasLog::getBasePath();

var_dump($basePath_1,$basePath_2);

/*
string(19) /log/seaslog-ciogao
string(14) /log/base_test
*/

$lastLogger_1 = SeasLog::getLastLogger();

SeasLog::setLogger('testModule/app1');
$lastLogger_2 = SeasLog::getLastLogger();

var_dump($lastLogger_1,$lastLogger_2);
/*
string(7) default
string(15) testModule/app1
*/
?>
```

See also ext/SeasLog on Github, and SeasLog.

Specs

Short name	Extensions/Extseaslog
Rulesets	*CE*
Exakt since	1.4.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.460 ext/sem

Extension Semaphore, Shared Memory and IPC.

This module provides wrappers for the System V IPC family of functions. It includes semaphores, shared memory and inter-process messaging (IPC).

```
<?php

$key          = ftok(__FILE__,'a');
$semaphore    = sem_get($key);
```

(continues on next page)

(continued from previous page)

```
sem_acquire($semaphore);
sem_release($semaphore);
sem_remove($semaphore);

?>
```

See also Semaphore, Shared Memory and IPC.

Specs

Short name	Extensions/Extsem
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.461 ext/session

Extension ext/session.

Session support in PHP consists of a way to preserve certain data across subsequent accesses.

```
<?php
session_start();
if (!isset($_SESSION['count'])) {
  $_SESSION['count'] = 0;
} else {
  $_SESSION['count']++;
}
?>
```

See also Session.

Specs

Short name	Extensions/Extsession
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.462 ext/shmop

Extension ext/shmop.

Shmop is an easy to use set of functions that allows PHP to read, write, create and delete Unix shared memory segments.

```
<?php
// Create a temporary file and return its path
$tmp = tempnam('/tmp', 'PHP');

// Get the file token key
$key = ftok($tmp, 'a');

// Attach the SHM resource, notice the cast afterwards
$id = shm_attach($key);

if ($id === false) {
    die('Unable to create the shared memory segment');
}

// Cast to integer, since prior to PHP 5.3.0 the resource id
// is returned which can be exposed when casting a resource
// to an integer
$id = (integer) $id;
?>
```

See also Semaphore, Shared Memory and IPC.

Specs

Short name	Extensions/Extshmop
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.463 ext/simplexml

Extension `SimpleXML`.

The `SimpleXML` extension provides a very simple and easily usable toolset to convert XML to an object that can be processed with normal property selectors and array iterators.

```
<?php

$xml = <<<'XML'
<?xml version='1.0' standalone='yes' ? >
<movies>
 <movie>
  <title>PHP: Behind the Parser</title>
  <characters>
   <character>
    <name>Ms. Coder</name>
    <actor>Onlivia Actora</actor>
   </character>
```

(continues on next page)

(continued from previous page)

```
   <character>
    <name>Mr. Coder</name>
    <actor>El Act&#211;r</actor>
   </character>
  </characters>
  <plot>
   So, this language. It's like, a programming language. Or is it a
   scripting language? All is revealed in this thrilling horror spoof
   of a documentary.
  </plot>
  <great-lines>
   <line>PHP solves all my web problems</line>
  </great-lines>
  <rating type="thumbs">7</rating>
  <rating type="stars">5</rating>
 </movie>
</movies>
XML;

$movies = new SimpleXMLElement($xml);

echo $movies->movie[0]->plot;
?>
```

See also SimpleXML.

Specs

Short name	Extensions/Extsimplexml
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.464 ext/snmp

Extension SNMP.

The SNMP extension provides a very simple and easily usable toolset for managing remote devices via the Simple Network Management Protocol.

```
<?php
$nameOfSecondInterface = snmp3_get('localhost', 'james', 'authPriv', 'SHA', 'secret007
↪', 'AES', 'secret007', 'IF-MIB::ifName.2');
?>
```

See also Net ‘SNMP <http://www.net-snmp.org/>‘_ and SNMP.

Specs

Short name	Extensions/Extsnmp
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.465 ext/soap

Extension SOAP.

The SOAP extension can be used to write SOAP Servers and Clients. It supports subsets of » SOAP 1.1, » SOAP 1.2 and » WSDL 1.1 specifications.

```
<?php

$client = new SoapClient("some.wsdl");

$client = new SoapClient("some.wsdl", array('soap_version'   => SOAP_1_2));

$client = new SoapClient("some.wsdl", array('login'          => "some_name",
                                            'password'       => "some_password"));

?>
```

See also SOAP and SOAP specifications.

Specs

Short name	Extensions/Extsoap
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.466 ext/sockets

Extension socket.

The socket extension implements a low-level interface to the socket communication functions based on the popular BSD sockets, providing the possibility to act as a socket server as well as a client.

```
<?php

//Example #2 Socket example: Simple TCP/IP client
//From the PHP manual
```

(continues on next page)

(continued from previous page)

```
error_reporting(E_ALL);

echo "<h2>TCP/IP Connection</h2>\n";

/* Get the port for the WWW service. */
$service_port = getservbyname('www', 'tcp');

/* Get the IP address for the target host. */
$address = gethostbyname('www.example.com');

/* Create a TCP/IP socket. */
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
if ($socket === false) {
    echo 'socket_create() failed: reason: ' . socket_strerror(socket_last_error()) .␣
↪PHP_EOL;
} else {
    echo 'OK.'.PHP_EOL;
}

echo 'Attempting to connect to '$address' on port '$service_port'...';
$result = socket_connect($socket, $address, $service_port);
if ($result === false) {
    echo 'socket_connect() failed.\nReason: ($result) ' . socket_strerror(socket_last_
↪error($socket)) . '\n';
} else {
    echo 'OK.'.PHP_EOL;
}

$in = ""HEAD / HTTP/1.1\r\n"";
$in .= ""Host: www.example.com\r\n"";
$in .= ""Connection: Close\r\n\r\n"";
$out = '';

echo 'Sending HTTP HEAD request...';
socket_write($socket, $in, strlen($in));
echo "OK.\n";

echo 'Reading response:\n\n';
while ($out = socket_read($socket, 2048)) {
    echo $out;
}

echo 'Closing socket...';
socket_close($socket);
echo 'OK.\n\n';
?>
```

See also Sockets.

Specs

Short name	Extensions/Extsockets
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.467 ext/sphinx

Extension for the Sphinx search server.

This extension provides bindings for Sphinx search client library.

```
<?php

$s = new SphinxClient;
$s->setServer(localhost, 6712);
$s->setMatchMode(SPH_MATCH_ANY);
$s->setMaxQueryTime(3);

$result = $s->query(test);

var_dump($result);

?>
```

See also Sphinx Client and Sphinx Search.

Specs

Short name	Extensions/Extsphinx
Rulesets	*CE*
Exakt since	0.11.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.468 ext/spl

SPL extension.

The Standard PHP Library (SPL) is a collection of interfaces and classes that are meant to solve common problems.

```
<?php

// Example with FilesystemIterator
$files = new FilesystemIterator('/path/to/dir');
```

(continues on next page)

(continued from previous page)

```
foreach($files as $file) {
    echo $file->getFilename() . PHP_EOL;
}

?>
```

See also Standard PHP Library (SPL).

Specs

Short name	Extensions/Extspl
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.469 ext/sqlite

Extension Sqlite 2.

Support for SQLite version 2 databases. The support for this version of Sqlite is not maintained anymore. It is recommended to use `SQLite3`.

```
<?php

if ($db = sqlite_open('mysqlitedb', 0666, $sqliteerror)) {
    sqlite_query($db, 'CREATE TABLE foo (bar varchar(10))');
    sqlite_query($db, 'INSERT INTO foo VALUES ("fnord")');
    $result = sqlite_query($db, 'select bar from foo');
    var_dump(sqlite_fetch_array($result));
} else {
    die($sqliteerror);
}

?>
```

See also ext/sqlite and SQLite.

Specs

Short name	Extensions/Extsqlite
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.470 ext/sqlite3

Extension Sqlite3.

Support for SQLite version 3 databases.

```
<?php
$db = new SQLite3('mysqlitedb.db');

$results = $db->query('SELECT bar FROM foo');
while ($row = $results->fetchArray()) {
    var_dump($row);
}
?>
```

See also ext/sqlite3 and Sqlite.

Specs

Short name	Extensions/Extsqlite3
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.471 ext/sqlsrv

Extension for Microsoft SQL Server Driver.

The SQLSRV extension allows you to access Microsoft SQL Server and SQL Azure databases when running PHP on Windows.

```
<?php
$serverName = 'serverName\sqlexpress';
$connectionInfo = array( 'Database'=>'dbName', 'UID'=>'username', 'PWD'=>'password' );
$conn = sqlsrv_connect( $serverName, $connectionInfo);
if( $conn === false ) {
     die( print_r( sqlsrv_errors(), true));
}

$sql = 'INSERT INTO Table_1 (id, data) VALUES (?, ?)';
$params = array(1, 'some data');

$stmt = sqlsrv_query( $conn, $sql, $params);
if( $stmt === false ) {
     die( print_r( sqlsrv_errors(), true));
}
?>
```

See also Microsoft SQL Server Driver and PHP Driver for SQL Server Support for LocalDB.

Specs

Short name	Extensions/Extsqlsrv
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.472 ext/ssh2

Extension ext/ssh2.

```
<?php
/* Notify the user if the server terminates the connection */
function my_ssh_disconnect($reason, $message, $language) {
  printf("Server disconnected with reason code [%d] and message: %s\n",
         $reason, $message);
}

$methods = array(
  'kex' => 'diffie-hellman-group1-sha1',
  'client_to_server' => array(
    'crypt' => '3des-cbc',
    'comp' => 'none'),
  'server_to_client' => array(
    'crypt' => 'aes256-cbc,aes192-cbc,aes128-cbc',
    'comp' => 'none'));

$callbacks = array('disconnect' => 'my_ssh_disconnect');

$connection = ssh2_connect('shell.example.com', 22, $methods, $callbacks);
if (!$connection) die('Connection failed');
?>
```

See also SSH2 functions and ext/ssh2 on PECL.

Specs

Short name	Extensions/Extssh2
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.473 ext/standard

Standards PHP functions.

This is not a real PHP extension : it covers the core functions.

```
<?php
/*
Our php.ini contains the following settings:

display_errors = On
register_globals = Off
post_max_size = 8M
*/

echo 'display_errors = ' . ini_get('display_errors') . PHP_EOL;
echo 'register_globals = ' . ini_get('register_globals') . PHP_EOL;
echo 'post_max_size = ' . ini_get('post_max_size') . PHP_EOL;
echo 'post_max_size+1 = ' . (ini_get('post_max_size')+1) . PHP_EOL;
echo 'post_max_size in bytes = ' . return_bytes(ini_get('post_max_size'));

function return_bytes($val) {
    $val = trim($val);
    $last = strtolower($val[strlen($val)-1]);
    switch($last) {
        // The 'G' modifier is available since PHP 5.1.0
        case 'g':
            $val *= 1024;
        case 'm':
            $val *= 1024;
        case 'k':
            $val *= 1024;
    }

    return $val;
}

?>
```

See also PHP Options/Info Functions.

Specs

Short name	Extensions/Extstandard
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.474 ext/stats

Statistics extension.

This extension contains few dozens of functions useful for statistical computations. It is a wrapper around 2 scientific libraries, namely DCDFLIB (Library of C routines for Cumulative Distributions Functions, Inverses, and Other parameters) by B. Brown & J. Lavato and RANDLIB by Barry Brown, James Lavato & Kathy Russell.

```
<?php

$x = [ 15, 16, 8, 6, 15, 12, 12, 18, 12, 20, 12, 14, ];
$y = [ 17.24, 15, 14.91, 4.5, 18, 6.29, 19.23, 18.69, 7.21, 42.06, 7.5, 8,];

sprintf(%2.9f, stats_covariance($a_1, $a_2));

?>
```

See also Statistics and ext/stats.

Specs

Short name	Extensions/Extstats
Rulesets	*CE*
Exakt since	0.11.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.475 String

Strings in PHP. Strings are part of the core of PHP, and are not a separate extension.

```
<?php
$str = Mary Had A Little Lamb and She LOVED It So;
$str = strtolower($str);

echo $str; // Prints mary had a little lamb and she loved it so
?>
```

See also String functions.

Specs

Short name	Extensions/Extstring
Rulesets	*CE*
Exakt since	0.9.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.476 ext/suhosin

Suhosin extension.

Suhosin (pronounced 'su-ho-shin') is an advanced protection system for PHP installations. It was designed to protect servers and users from known and unknown flaws in PHP applications and the PHP core.

```
<?php

// sha256 is a ext/suhosin specific function
$sha256 = sha256($string);

?>
```

See also Suhosin.org

Specs

Short name	Extensions/Extsuhosin
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.477 ext/svm

Extension `SVM`.

`SVM` is in interface with the `libsvm`, from . ‘‘libsvm‘‘is a library for Support Vector Machines, a classification tool for machine learning.

```
<?php
   $data = array(
       array(-1, 1 => 0.43, 3 => 0.12, 9284 => 0.2),
       array(1, 1 => 0.22, 5 => 0.01, 94 => 0.11),
   );

   $svm = new SVM();
   $model = $svm->train($data);

   $data = array(1 => 0.43, 3 => 0.12, 9284 => 0.2);
   $result = $model->predict($data);
   var_dump($result);
   $model->save('model.svm');
?>
```

See also SVM <http://www.php.net/‘svm>‘_, LIBSVM – A Library for Support Vector Machines, ext/‘svm <https://pecl.php.net/package/svm>‘_ and ianbarber/php-‘svm <https://github.com/ianbarber/php-svm>‘_.

Specs

Short name	Extensions/Extsvm
Rulesets	*CE*
Exakt since	1.7.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.478 ext/swoole

Swoole : Production-Grade Async programming Framework for PHP.

Swoole is an event-driven asynchronous & concurrent networking communication framework with high performance written only in C for PHP.

```
<?php
for($i = 0; $i < 100; $i++) {
    Swoole\Coroutine::create(function() use ($i) {
        $redis = new Swoole\Coroutine\Redis();
        $res = $redis->connect('127.0.0.1', 6379);
        $ret = $redis->incr('coroutine');
        $redis->close();
        if ($i == 50) {
            Swoole\Coroutine::create(function() use ($i) {
                $redis = new Swoole\Coroutine\Redis();
                $res = $redis->connect('127.0.0.1', 6379);
                $ret = $redis->set('coroutine_i', 50);
                $redis->close();
            });
        }
    });
}

?>
```

See also Swoole and Swoole src.

Specs

Short name	Extensions/Extswoole
Rulesets	*CE*
Exakt since	0.12.0
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.479 ext/tidy

Extension Tidy.

Tidy is a binding for the Tidy HTML clean and repair utility which allows you to not only clean and otherwise manipulate HTML documents, but also traverse the document tree.

```
<?php
ob_start();
?>
```

<html>a html document</html> .. code-block:: php

<?php $html = ob_get_clean();

// Specify configuration $config = array(

'indent' => true, 'output-xhtml' => true, 'wrap' => 200);

// Tidy $tidy = new tidy; $tidy->parseString($html, $config, 'utf8'); $tidy->cleanRepair();

// Output echo $tidy; ?>

See also Tidy <https://www.php.net/manual/en/book.'tidy.php>'_ and HTML-'tidy <http://www.html-tidy.org/>'_.

Specs

Short name	Extensions/Exttidy
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.480 ext/tokenizer

Extension Tokenizer.

The Tokenizer functions provide an interface to the PHP tokenizer embedded in the Zend Engine.

```
<?php
/*
* T_ML_COMMENT does not exist in PHP 5.
* The following three lines define it in order to
* preserve backwards compatibility.
*
* The next two lines define the PHP 5 only T_DOC_COMMENT,
* which we will mask as T_ML_COMMENT for PHP 4.
*/
if (!defined('T_ML_COMMENT')) {
   define('T_ML_COMMENT', T_COMMENT);
} else {
   define('T_DOC_COMMENT', T_ML_COMMENT);
}
```

(continues on next page)

(continued from previous page)

```
$source = file_get_contents('example.php');
$tokens = token_get_all($source);

foreach ($tokens as $token) {
   if (is_string($token)) {
       // simple 1-character token
       echo $token;
   } else {
       // token array
       list($id, $text) = $token;

       switch ($id) {
           case T_COMMENT:
           case T_ML_COMMENT: // we\'ve defined this
           case T_DOC_COMMENT: // and this
               // no action on comments
               break;

           default:
               // anything else -> output 'as is'
               echo $text;
               break;
       }
   }
}
?>
```

See also tokenizer.

Specs

Short name	Extensions/Exttokenizer
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.481 ext/tokyotyrant

Extension for Tokyo Tyrant.

tokyo_tyrant extension provides a wrapper for Tokyo Tyrant client libraries.

```
<?php
$tt = new TokyoTyrant("localhost");
$tt->put("key", "value");
echo $tt->get("key");
?>
```

See also tokyo_tyrant and Tokyo cabinet.

Specs

Short name	Extensions/Exttokyotyrant
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.482 ext/trader

Extension trader.

The trader extension is a free open source stock library based on TA-Lib. It's dedicated to trading software developers requiring to perform technical analysis of financial market data.

```
<?php

// get_data() reads the data from a source
var_dump(trader_avgprice(
     get_data(open, $data0),
     get_data(high, $data0),
     get_data(low, $data0),
     get_data(close, $data0)
));

?>
```

See also trader (PECL), 'TA-lib <http://www.ta-lib.org/>'_ and ext/trader.

Specs

Short name	Extensions/Exttrader
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.483 ext/uopz

Extension UOPZ : User Operations for Zend.

The uopz extension is focused on providing utilities to aid with unit testing PHP code.

It supports the following activities: Intercepting function execution, Intercepting object creation, Hooking into function execution, Manipulation of function statics, Manipulation of function flags, Redefinition of constants, Deletion of constants, Runtime creation of functions and methods,

```php
<?php
// The example is extracted from the UOPZ extension test suite : tests/001.phpt
class Foo {
     public function bar(int $arg) : int {
              return $arg;
     }
}
var_dump(uopz_set_return(Foo::class, 'bar', true));
$foo = new Foo();
var_dump($foo->bar(1));
uopz_set_return(Foo::class, 'bar', function(int $arg) : int {
     return $arg * 2;
}, true);
var_dump($foo->bar(2));
try {
     uopz_set_return(Foo::class, 'nope', 1);
} catch(Throwable $t) {
     var_dump($t->getMessage());
}
class Bar extends Foo {}
try {
     uopz_set_return(Bar::class, 'bar', null);
} catch (Throwable $t) {
     var_dump($t->getMessage());
}

     uopz_set_something(Bar::class, 'bar', null);

?>
```

See also ext/uopz and User Operations for Zend.

Specs

Short name	Extensions/Extuopz
Rulesets	*CE*
Exakt since	1.1.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.484 ext/uuid

> Extension `UUID`. A universally unique identifier (UUID) is a 128-bit number used to identify information in computer systems.

An interface to the libuuid system library. The libuuid library is used to generate unique identifiers for objects that may be accessible beyond the local system. The Linux implementation was created to uniquely identify ext2 filesystems created by a machine. This library generates UUIDs compatible with those created by the Open Software Foundation (OSF) Distributed Computing Environment (DCE) utility uuidgen.

```
<?php
    // example from the test suitee of the extension.

    // check basic format of generated UUIDs
    $uuid = uuid_create();
    if (preg_match("/[[:xdigit:]]{8}-[[:xdigit:]]{4}-[[:xdigit:]]{4}-[[:xdigit:]]{4}-
↪[[:xdigit:]]{12}/", $uuid)) {
            echo "basic format ok\n";
    } else {
            echo "basic UUID format check failed, generated UUID was $uuid\n";
    }

?>
```

See also libuuid <https://linux.‘die.net/man/3/libuuid>‘_ and ext/uuid.

Specs

Short name	Extensions/Extuuid
Rulesets	*CE*
Exakt since	1.7.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.485 ext/v8js

Extension v8js.

This extension embeds the V8 Javascript Engine into PHP.

```
<?php

$v8 = new V8Js();

/* basic.js */
$JS = <<< EOT
len = print('Hello' + ' ' + 'World!' + '\n');
len;
EOT;

try {
  var_dump($v8->executeString($JS, 'basic.js'));
} catch (V8JsException $e) {
  var_dump($e);
}

?>
```

See also V8 Javascript Engine Integration <https://www.php.net/manual/en/book.‘v8js.php>‘_, V8 Javascript Engine for PHP <https://github.com/phpv8/‘v8js>‘_ and pecl ‘v8js <https://pecl.php.net/package/v8js>‘_.

Specs

Short name	Extensions/Extv8js
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.486 ext/varnish

Extension PHP for varnish.

Varnish Cache is an open source, state of the art web application accelerator. The extension makes it possible to interact with a running varnish instance through TCP socket or shared memory.

```
<?php
    $args = array(
        VARNISH_CONFIG_HOST => '::1',
        VARNISH_CONFIG_PORT => 6082,
        VARNISH_CONFIG_SECRET => '5174826b-8595-4958-aa7a-0609632ad7ca',
        VARNISH_CONFIG_TIMEOUT => 300,
    );
    $va = new VarnishAdmin($args);
?>
```

See also ext/varnish and pecl/Varnish.

Specs

Short name	Extensions/Extvarnish
Rulesets	*CE*
Exakt since	1.1.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.487 ext/vips

Extension VIPS.

The VIPS image processing system is a very fast, multi-threaded image processing library with low memory needs.

```
<?php
     dl('vips.' . PHP_SHLIB_SUFFIX);
     $x = vips_image_new_from_file($argv[1])[out];
     vips_image_write_to_file($x, $argv[2]);
?>
```

See also php-vips-ext, libvips and libvips adapter for PHP Imagine.

Specs

Short name	Extensions/Extvips
Rulesets	*CE*
Exakt since	1.0.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.488 ext/wasm

Extension WASM.

The goal of the project is to be able to run WebAssembly binaries from PHP directly. So much fun coming!

From the php-ext-wasm documentation :

```
<?php

//There is a toy program in examples/simple.rs, written in Rust (or any other␣
↪language that compiles to WASM):
// Stored in file __DIR__ . '/simple.wasm'
/*
#[no_mangle]
pub extern C fn sum(x: i32, y: i32) -> i32 {
    x + y
}
*/

$instance = new WASM\Instance(__DIR__ . '/simple.wasm');

var_dump(
    $instance->sum(5, 37) // 42!
);

?>
```

See also php-ext-wasm.

Specs

Short name	Extensions/Extwasm
Rulesets	*CE*
Exakt since	1.5.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.489 ext/wddx

Extension WDDX.

The Web Distributed Data Exchange, or WDDX, is a free, open XML-based technology that allows Web applications created with any platform to easily exchange data with one another over the Web.

```
<?php
  echo wddx_serialize_value("PHP to WDDX packet example", "PHP packet");
?>
```

See also Wddx on PHP and WDDX.

Specs

Short name	Extensions/Extwddx
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.490 ext/weakref

Weak References for PHP.

Weak references provide a non-intrusive gateway to ephemeral objects. Unlike normal (strong) references, weak references do not prevent the garbage collector from freeing that object. For this reason, an object may be destroyed even though a weak reference to that object still exists. In such conditions, the weak reference seamlessly becomes invalid.

```
<?php
class MyClass {
    public function __destruct() {
        echo Destroying object!\n;
    }
}

$o1 = new MyClass;

$r1 = new WeakRef($o1);

if ($r1->valid()) {
    echo Object still exists!\n;
    var_dump($r1->get());
} else {
    echo Object is dead!\n;
}

unset($o1);

if ($r1->valid()) {
    echo Object still exists!\n;
```

(continues on next page)

(continued from previous page)

```
    var_dump($r1->get());
} else {
    echo Object is dead!\n;
}
?>
```

See also Weak references <https://www.php.net/manual/en/book.‘weakref.php>‘_ **and** PECL extension that implements weak references and weak maps in PHP <https://github.com/colder/php-‘weakref>‘_.

Specs

Short name	Extensions/Extweakref
Rulesets	*CE*
Exakt since	1.6.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.491 ext/wikidiff2

Extension wikidiff2.

Wikidiff2 is a PHP and HHVM module that provides the external diff engine for MediaWiki.

```
<?php
$x = <<<EOT
foo bar
baz
quux
bang
EOT;

$y = <<<EOT
foo test
baz
test
bang
EOT;

print wikidiff2_inline_diff( $x, $y, 2 );
?>
```

See also wikidiff2 and wikidiff2 (C ext).

Specs

Short name	Extensions/Extwikidiff2
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.492 ext/wincache

Extension Wincache.

The Wincache extension for PHP is a PHP accelerator that is used to increase the speed of PHP applications running on Windows and Windows Server.

```
<?php
$fp = fopen('/tmp/lock.txt', 'r+');
if (wincache_lock("lock_txt_lock")) { // do an exclusive lock
    ftruncate($fp, 0); // truncate file
    fwrite($fp, 'Write something here\n');
    wincache_unlock("lock_txt_lock"); // release the lock
} else {
    echo 'Couldn't get the lock!';
}
fclose($fp);
?>
```

See also WinCache Homepage.

Specs

Short name	Extensions/Extwincache
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.493 ext/xattr

Extensions xattr.

The xattr extension allows for the manipulation of extended attributes on a filesystem.

```
<?php
$file = 'my_favourite_song.wav';
xattr_set($file, 'Artist', 'Someone');
xattr_set($file, 'My ranking', 'Good');
```

(continues on next page)

(continued from previous page)

```
xattr_set($file, 'Listen count', '34');

/* ... other code ... */

printf('You\'ve played this song %d times', xattr_get($file, 'Listen count'));
?>
```

See also xattr and Extended attributres.

Specs

Short name	Extensions/Extxattr
Rulesets	*CE*
Exakt since	0.12.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.494 ext/xcache

Extension Xcache.

XCache is a open-source opcode cacher, which means that it accelerates the performance of PHP on servers.

```
<?php
if (!xcache_isset(count)) {
  xcache_set(count, load_count_from_mysql());
}
?>
```

This guest book has been visited .. code-block:: php

<?php echo $count = xcache_inc(count); ?>

times.

```
<?php
// save every 100 hits
if (($count % 100) == 0) {
  save_count_to_mysql($count);
}
?>
```

See also xcache.

Specs

Short name	Extensions/Extxcache
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.495 ext/xdebug

Xdebug extension.

The Xdebug is a extension PHP which provides debugging and profiling capabilities.

```
<?php
class Strings
{
    static function fix_string($a)
    {
        echo
            xdebug_call_class().
            "::".
            xdebug_call_function().
            " is called at ".
            xdebug_call_file().
            ":".
            xdebug_call_line();
    }
}

$ret = Strings::fix_string( 'Derick' );
?>
```

See also Xdebug.

Specs

Short name	Extensions/Extxdebug
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.496 ext/xdiff

Extension xdiff.

xdiff extension enables you to create and apply patch files containing differences between different revisions of files.

```
<?php
$old_version = 'my_script-1.0.php';
$patch = 'my_script.patch';

$errors = xdiff_file_patch($old_version, $patch, 'my_script-1.1.php');
if (is_string($errors)) {
   echo 'Rejects:'.PHP_EOL;
   echo $errors;
}

?>
```

See also xdiff and libxdiff.

Specs

Short name	Extensions/Extxdiff
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.497 ext/xhprof

Extension xhprof.

XHProf is a light-weight hierarchical and instrumentation based profiler.

```
<?php
xhprof_enable(XHPROF_FLAGS_CPU + XHPROF_FLAGS_MEMORY);

for ($i = 0; $i <= 1000; $i++) {
    $a = $i * $i;
}

$xhprof_data = xhprof_disable();

$XHPROF_ROOT = '/tools/xhprof/';
include_once $XHPROF_ROOT . '/xhprof_lib/utils/xhprof_lib.php';
include_once $XHPROF_ROOT . '/xhprof_lib/utils/xhprof_runs.php';

$xhprof_runs = new XHProfRuns_Default();
$run_id = $xhprof_runs->save_run($xhprof_data, 'xhprof_testing');

echo 'http://localhost/xhprof/xhprof_html/index.php?run={$run_id}&source=xhprof_
↪testing'.PHP_EOL;

?>
```

See also XHprof Documentation.

Specs

Short name	Extensions/Extxhprof
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.498 ext/xml

Extension xml (Parser).

This PHP extension implements support for James Clark's expat in PHP. This toolkit lets you parse, but not validate, XML documents.

```
<?php
$file = data.xml;
$depth = array();

function startElement($parser, $name, $attrs)
{
    global $depth;

    if (!isset($depth[$parser])) {
        $depth[$parser] = 0;
    }

    for ($i = 0; $i < $depth[$parser]; $i++) {
        echo ;
    }
    echo $name\n;
    $depth[$parser]++;
}

function endElement($parser, $name)
{
    global $depth;
    $depth[$parser]--;
}

$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, startElement, endElement);
if (!($fp = fopen($file, r))) {
    die(could not open XML input);
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf(XML error: %s at line %d,
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
```

(continues on next page)

(continued from previous page)

```
xml_parser_free($xml_parser);
?>
```

See also XML Parser.

Specs

Short name	Extensions/Extxml
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.499 ext/xmlreader

Extension XMLReader.

The XMLReader extension is an XML Pull parser. The reader acts as a cursor going forward on the document stream and stopping at each node on the way.

```
<?php

    $xmlreader = new XMLReader();
    $xmlreader->xml("<xml><div>Content</div></xml>");
    $xmlreader->read();
    $xmlreader->read();
    $xmlreader->readString();

?>
```

See also xmlreader.

Specs

Short name	Extensions/Extxmlreader
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.500 ext/xmlrpc

Extension ext/xmlrpc.

This extension can be used to write XML-RPC servers and clients.

```
<?php
$request = xmlrpc_encode_request('method', array(1, 2, 3));
$context = stream_context_create(array('http' => array(
    'method' => 'POST',
    'header' => 'Content-Type: text/xml',
    'content' => $request
)));
$file = file_get_contents('http://www.example.com/xmlrpc', false, $context);
$response = xmlrpc_decode($file);
if ($response && xmlrpc_is_fault($response)) {
    trigger_error('xmlrpc: '.$response['faultString'].' ('.$response['faultCode']));
} else {
    print_r($response);
}
?>
```

See also XML-RPC.

Specs

Short name	Extensions/Extxmlrpc
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.501 ext/xmlwriter

Extension ext/xmlwriter.

The XMLWriter extension wraps the libxml xmlWriter API inside PHP.

```
<?php
$xw = xmlwriter_open_memory();
xmlwriter_set_indent($xw, TRUE);
xmlwriter_start_document($xw, NULL, 'UTF-8');
xmlwriter_start_element($xw, 'root');
xmlwriter_write_attribute_ns($xw, 'prefix', '', 'http://www.php.net/uri');
xmlwriter_start_element($xw, 'elem1');
xmlwriter_write_attribute($xw, 'attr1', 'first');
xmlwriter_end_element($xw);
xmlwriter_full_end_element($xw);
xmlwriter_end_document($xw);
$output = xmlwriter_flush($xw, true);
print $output;
// write attribute_ns without start_element first
$xw = xmlwriter_open_memory();
var_dump(xmlwriter_write_attribute_ns($xw, 'prefix', 'id', 'http://www.php.net/uri',
↪'elem1'));
print xmlwriter_output_memory($xw);
?>
```

See also XMLWriter and Module xmlwriter from libxml2.

Specs

Short name	Extensions/Extxmlwriter
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.502 ext/xsl

Extension XSL.

The XSL extension implements the XSL standard, performing XSLT transformations using the libxslt library.

```
<?php

// Example from the PHP manual

$xmldoc = new DOMDocument();
$xsldoc = new DOMDocument();
$xsl = new XSLTProcessor();

$xmldoc->loadXML('fruits.xml');
$xsldoc->loadXML('fruits.xsl');

libxml_use_internal_errors(true);
$result = $xsl->importStyleSheet($xsldoc);
if (!$result) {
    foreach (libxml_get_errors() as $error) {
        echo "Libxml error: {$error->message}\n";
    }
}
libxml_use_internal_errors(false);

if ($result) {
    echo $xsl->transformToXML($xmldoc);
}

?>
```

See also XSL extension;

Specs

Short name	Extensions/Extxsl
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.503 ext/xxtea

Extension xxtea : XXTEA encryption algorithm extension for PHP.

XXTEA is a fast and secure encryption algorithm. This is a XXTEA extension for PHP. It is different from the original XXTEA encryption algorithm. It encrypts and decrypts string instead of uint32 array, and the key is also string.

```
<?php
// Example is extracted from the xxtea repository on github : tests/xxtea.phpt

$str = 'Hello World! ';
$key = '1234567890';
$base64 = 'D4t0rVXUDl3bnWdERhqJmFIanfn/6zAxAY9jD6n9MSMQNoD8TOS4rHHcGuE=';
$encrypt_data = xxtea_encrypt($str, $key);
$decrypt_data = xxtea_decrypt($encrypt_data, $key);
if ($str == $decrypt_data && base64_encode($encrypt_data) == $base64) {
    echo 'success!';
} else {
    echo base64_encode($encrypt_data);
    echo 'fail!';
}
?>
```

See also PECL ext/xxtea and ext/xxtea on Github.

Specs

Short name	Extensions/Extxxtea
Rulesets	*CE*
Exakt since	1.1.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.504 ext/yaml

Extension YAML.

This extension implements the YAML Ain't Markup Language (YAML) data serialization standard. Parsing and emitting are handled by the LibYAML library.

```php
<?php
$addr = array(
    'given' => 'Chris',
    'family'=> 'Dumars',
    'address'=> array(
        'lines'=> '458 Walkman Dr.
        Suite #292',
        'city'=> 'Royal Oak',
        'state'=> 'MI',
        'postal'=> 48046,
      ),
  );
$invoice = array (
    'invoice'=> 34843,
    'date'=> '2001-01-23',
    'bill-to'=> $addr,
    'ship-to'=> $addr,
    'product'=> array(
        array(
            'sku'=> 'BL394D',
            'quantity'=> 4,
            'description'=> 'Basketball',
            'price'=> 450,
          ),
        array(
            'sku'=> 'BL4438H',
            'quantity'=> 1,
            'description'=> 'Super Hoop',
            'price'=> 2392,
          ),
      ),
    'tax'=> 251.42,
    'total'=> 4443.52,
    'comments'=> 'Late afternoon is best. Backup contact is Nancy Billsmer @ 338-4338.
↪',
    );

// generate a YAML representation of the invoice
$yaml = yaml_emit($invoice);
var_dump($yaml);

// convert the YAML back into a PHP variable
$parsed = yaml_parse($yaml);

// check that roundtrip conversion produced an equivalent structure
var_dump($parsed == $invoice);
?>
```

See also YAML.

Specs

Short name	Extensions/Extyaml
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.505 ext/yis

Yellow Pages extensions (NIS).

NIS (formerly called Yellow Pages) allows network management of important administrative files (e.g. the password file).

```
<?php
$entry = yp_first($domain, 'passwd.byname');

$key = key($entry);
$value = $entry[$key];

echo 'First entry in this map has key ' . $key . ' and value ' . $value;
?>
```

See also The Linux NIS(YP)/NYS/NIS+ HOWTO and YP/NIS.

Specs

Short name	Extensions/Extyis
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.506 ext/zbarcode

Extension Zbarcode.

PHP extension for reading barcodes.

```
<?php
/* Create new image object */
$image = new ZBarCodeImage('test.jpg');

/* Create a barcode scanner */
$scanner = new ZBarCodeScanner();

```

(continues on next page)

(continued from previous page)

```
/* Scan the image */
$barcode = $scanner->scan($image);

/* Loop through possible barcodes */
if (!empty($barcode)) {
     foreach ($barcode as $code) {
             printf('Found type %s barcode with data %s\n', $code['type'], $code['data
↪']);
     }
}
?>
```

See also php-zbarcode.

Specs

Short name	Extensions/Extzbarcode
Rulesets	*CE*
Exakt since	0.9.5
Php Version	7.0+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.507 ext/zend_monitor

Extension `zend_monitor`.

See also Zend Monitor - PHP API.

Specs

Short name	Extensions/Extzendmonitor
Rulesets	*CE*
Exakt since	1.7.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.508 ext/zip

Extension ext/zip.

This extension enables you to transparently read or write ZIP compressed archives and the files inside them.

```
<?php

$zip = new ZipArchive();
```

(continues on next page)

(continued from previous page)

```
$filename = './test112.zip';

if ($zip->open($filename, ZipArchive::CREATE)!==TRUE) {
    exit('cannot open <$filename>');
}

$zip->addFromString('testfilephp.txt' . time(), '#1 This is a test string added as␣
↪testfilephp.txt.'.PHP_EOL);
$zip->addFromString('testfilephp2.txt' . time(), '#2 This is a test string added as␣
↪testfilephp2.txt.'.PHP_EOL);
$zip->addFile($thisdir . '/too.php','/testfromfile.php');
echo 'numfiles: ' . $zip->numFiles . PHP_EOL;
echo 'status:' . $zip->status . PHP_EOL;
$zip->close();
?>
```

See also Zip.

Specs

Short name	Extensions/Extzip
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.509 ext/zlib

Extension ext/zlib.

```
<?php

$filename = tempnam('/tmp', 'zlibtest') . '.gz';
echo "<html>\n<head></head>\n<body>\n<pre>\n";
$s = "Only a test, test, test, test, test, test, test, test!\n";

// open file for writing with maximum compression
$zp = gzopen($filename, 'w9');

// write string to file
gzwrite($zp, $s);

// close file
gzclose($zp);

?>
```

See also Zlib.

Specs

Short name	Extensions/Extzlib
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.510 ext/0mq

Extension ext/zmq for `0mq`.

`ØMQ is a software library that lets you quickly design and implement a fast message-based application.`

```
<?php

// Example from https://github.com/kuying/ZeroMQ/blob/
↪d80dcc3dc1c14a343ca90bbd656b98fd55366548/zguide/examples/PHP/msgqueue.php
    /*
     *  Simple message queuing broker
     *  Same as request-reply broker but using QUEUE device
     * @author Ian Barber <ian(dot)barber(at)gmail(dot)com>
     */
    $context = new ZMQContext();
    //  Socket facing clients
    $frontend = $context->getSocket(ZMQ::SOCKET_ROUTER);
    $frontend->bind(tcp://*:5559);
    //  Socket facing services
    $backend = $context->getSocket(ZMQ::SOCKET_DEALER);
    $backend->bind(tcp://*:5560);
    //  Start built-in device
    new ZMQDevice($frontend, $backend);

?>
```

See also ZeroMQ and ZMQ.

Specs

Short name	Extensions/Extzmq
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.511 ext/zookeeper

Extension for Apache Zookeeper.

ZooKeeper is an Apache project that enables centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.

```
<?php
$zookeeper = new Zookeeper('locahost:2181');
$path = '/path/to/node';
$value = 'nodevalue';
$zookeeper->set($path, $value);

$r = $zookeeper->get($path);
if ($r)
  echo $r;
else
  echo 'ERR';
?>
```

See also ext/zookeeper, Install 'Zookeeper PHP Extension <https://blog.programster.org/install-zookeeper-php-extension>'_ Zookeeper and .

Specs

Short name	Extensions/Extzookeeper
Rulesets	*CE*
Exakt since	1.2.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.512 Definitions Only

File is definition only.

Definition-only files only include structure definitions : class, functions, traits, interfaces, constants, global, declare(), use and include().

Some functioncalls are also considered definition only, as they configure PHP, but don't process data : * ini_set() * error_reporting * register_shutdown_function() * set_session_handler() * set_error_handler() * spl_autoload_register()

File A : .. code-block:: php

    <?php

    // This file has only definitions function foo() {}

    define('a', 1);

    class bar {}

    ?>

File B : .. code-block:: php

<?php

// This file has only definitions function foo() {}

define('a', 1);

class bar {}

?>

Specs

Short name	Files/DefinitionsOnly
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.513 Global Code Only

This file has only global code.

Specs

Short name	Files/GlobalCodeOnly
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.514 Inclusion Wrong Case

Inclusion should follow exactly the case of included files and path. This prevents the infamous case-sensitive filesystem bug, where files are correctly included in a case-insensitive system, and failed to be when moved to production.

```
<?php

// There must exist a path called path/to and a file library.php with this case
include path/to/library.php;

// Error on the case, while the file does exist
include path/to/LIBRARY.php;

// Error on the case, on the PATH
include path/TO/library.php;
```

(continues on next page)

(continued from previous page)

```
?>
```

See also include_once, about case sensitivity and inclusions.

Suggestions

- Make the inclusion string identical to the file name.
- Change the name of the file to reflect the actual inclusion. This is the best way when a naming convention has been set up for the project, and the file doesn't adhere to it. Remember to change all other inclusion.

Specs

Short name	Files/InclusionWrongCase
Rulesets	*Analyze*
Exakt since	1.1.1
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.515 Is CLI Script

Mark a file as a CLI script, based on the usage of #!.

Specs

Short name	Files/IsCliScript
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.516 File Is Component

Check that a file only contains definition elements, such as traits, interfaces, classes, constants, global variables, use or inclusions.

Such a file is a component, that may be included in some other code and used. By itself, it doesn't run.

Specs

Short name	Files/IsComponent
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.517 Missing Include

The included files doesn't exists in the repository. The inclusions target a files that doesn't exist.

The analysis works with every type of inclusion : include(), require(), include_once() and require_once(). It also works with parenthesis when used as parameter delimiter.

The analysis doesn't take into account `include_path`. This may yield false positives.

```
<?php

include 'non_existent.php';

// variables are not resolved. This won't be reported.
require ($path.'non_existent.php');

?>
```

Missing included files may lead to a fatal error, a warning or other error later in the execution.

Name	Default	Type	Description
constant_or_variable_name	100	string	Literal value to be used when including files. For example, by configuring Files_MissingInclude["HOME_DIR"] = "/tmp/myDir/";', then 'include HOME_DIR . "my_class.php"; will be actually be used as '/tmp/myDir/my_class.php'. Constants must be configured with their correct case. Variable must be configured with their initial '$'. Configure any number of variable and constant names.

Specs

Short name	Files/MissingInclude
Rulesets	*Analyze*
Exakt since	1.1.2
Php Version	All
Severity	Critical
Time To Fix	Instant (5 mins)
Precision	High

13.2.518 Not Definitions Only

Files should only include definitions (class, functions, traits, interfaces, constants), or global instructions, but not both.

```
<?php
// This whole script is a file

// It contains definitions and global code
class foo {
    static public $foo = null;
}
//This can be a singleton creation
foo::$foo = new foo();

trait t {}

class bar {}

?>
```

Within this context, globals, use, and namespaces instructions are not considered a warning.

Specs

Short name	Files/NotDefinitionsOnly
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.519 External Config Files

List services being used in this code repository, based on configuration files that are committed.

For example, a .git folder is an artefact of a GIT repository.

Specs

Short name	Files/Services
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.520 Add Default Value

Parameter in methods definition may receive a default value. This allows the called method to set a value when the parameter is omitted.

```
<?php

function foo($i) {
    if (!is_integer($i)) {
        $i = 0;
    }
}

?>
```

See also Function arguments.

Suggestions

- Add a default value for parameters

Specs

Short name	Functions/AddDefaultValue
Rulesets	*Suggestions*
Exakt since	1.4.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Zurmo*, *Typo3*

13.2.521 Aliases Usage

PHP manual recommends to avoid function aliases.

Some functions have several names, and both may be used the same way. However, one of the names is the main name, and the others are aliases. Aliases may be removed or change or dropped in the future. Even if this is not forecast, it is good practice to use the main name, instead of the aliases.

```
<?php

// official way to count an array
$n = count($array);

// official way to count an array
$n = sizeof($array);

?>
```

Aliases are compiled in PHP, and do not provide any performances over the normal function.

Aliases are more likely to be removed later, but they have been around for a long time.

See documentation : List of function aliases.

Suggestions

- Always use PHP recommended functions

Specs

Short name	Functions/AliasesUsage
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-aliases
Examples	*Cleverstyle*, *phpMyAdmin*

13.2.522 Use Named Boolean In Argument Definition

Boolean in argument definitions is confusing.

It is recommended to use explicit constant names, instead. They are more readable. They also allow for easy replacement when the code evolve and has to replace those booleans by strings. This works even also with classes, and class constants.

```
<?php

function flipImage($im, $horizontal = NO_HORIZONTAL_FLIP, $vertical = NO_VERTICAL_
↪FLIP) { }

// with constants
const HORIZONTAL_FLIP = true;
const NO_HORIZONTAL_FLIP = true;
const VERTICAL_FLIP = true;
const NO_VERTICAL_FLIP = true;

rotateImage($im, HORIZONTAL_FLIP, NO_VERTICAL_FLIP);


// without constants
function flipImage($im, $horizontal = false, $vertical = false) { }

rotateImage($im, true, false);

?>
```

See also Flag Argument, to avoid boolean altogether.

Specs

Short name	Functions/AvoidBooleanArgument
Rulesets	*Analyze*
Exakt since	1.0.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*phpMyAdmin*, *Cleverstyle*

13.2.523 Bad Typehint Relay

A bad typehint relay happens where a type hinted argument is relayed to a parameter with another typehint. This will lead to a Fatal error, and stop the code. This is possibly a piece of dead code.

```
<?php

// the $i argument is relayed to bar, which is expecting a string.
function foo(int $i) : string {
    return bar($i);
}

// the return value for the bar function is not compatible with the one from foo;
function bar(string $s) : int {
    return (int) $string + 1;
}

?>
```

It is recommended to harmonize the typehints, so the two methods are compatible.

Suggestions

- Harmonize the typehint so they match one with the other.
- Remove dead code
- Apply type casting before calling the next function, or return value

Specs

Short name	Functions/BadTypehintRelay
Rulesets	*Typechecks*
Exakt since	1.6.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.524 Callback Function Needs Return

When used with array_map() functions, the callback must return something. This return may be in the form of a `return` statement, a global variable or a parameter with a reference. All those solutions extract information from the callback.

```
<?php

// This filters each element
$filtered = array_filter($array, function ($x) {return $x == 2; });

// This return void for every element
$filtered = array_filter($array, function ($x) {return ; });

// costly array_sum()
$sum = 0;
$filtered = array_filter($array, function ($x) use (&$sum) {$sum += $x; });

// costly array_sum()
global $sum = 0;
$filtered = array_filter($array, function () {global $sum; $sum += $x; });

// register_shutown_function() doesn't require any return
register_shutown_function(my_shutdown);

?>
```

The following functions are omitted, as they don't require the return :

- forward_static_call_array()
- forward_static_call()
- register_shutdown_function()
- register_tick_function()

See also array_map.

Suggestions

- Add an explicit return to the callback
- Use *null* to unset elements in an array without destroying the index

Specs

Short name	Functions/CallbackNeedsReturn
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.2.6
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Contao*, *Phpdocumentor*

13.2.525 Cancelled Parameter

When a parameter is hardcoded, and cannot be changed from the calling expression. The argument is in the signature, but it is later hardcoded to a literal value : thus, it is not usable, from the caller point of view.

Reference argument are omitted here, as the value change, however hardcoded, may have an impact on the calling code.

```
<?php

function foo($a, $b) {
    // $b is cancelled, and cannot be changed.
    $b = 3;

    // $a is the only parameter here
    return $a + $b;
}

function bar($a, $b) {
    // $b is actually processed
    $c = $b;
    $c = process($c);

    $b = $c;

    // $a is the only parameter here
    return $a + $b;
}

?>
```

Suggestions

- Remove the parameter in the method signature

Specs

Short name	Functions/CancelledParameter
Rulesets	*Analyze*
Exakt since	2.2.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.526 Cannot Use Static For Closure

The reported closures and arrow functions cannot use the static keyword.

Closures that makes use of the $this pseudo-variable cannot use the static keyword, at it prevents the import of the $this context in the closure. It will fail at execution.

Closures that makes use of the bindTo() method, to change the context of execution, also cannot use the static keyword. Even if $this is not used in the closure, the static keyword prevents the call to bindTo().

```
<?php

class x {
    function foo() {
        // Not possible, $this is now undefined in the body of the closure
        static function () { return $this->a;};
    }

    function foo2() {
        // Not possible, $this is now undefined in the body of the arrow function
        static fn () => $this->a;
    }

    function foo3() {
        // Not possible, the closure gets a new context before being called.
        $a = static fn () => $ba;
        $this->foo4($a);
    }

    function foo4($c) {
        $c->bindTo($this);
        $c();
    }

}
?>
```

See also Static anonymous functions <https://www.php.net/manual/en/functions.anonymous.php#functions.anonymous-functions.‘static>‘_.

Suggestions

- Remove the static keyword
- Remove the call to bindTo() method
- Remove the usage of the $this variable

Specs

Short name	Functions/CannotUseStaticForClosure
Rulesets	*Analyze*
Exakt since	2.2.2
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.527 Cant Use Function

Those functions only contains an error or an exception. As such, they are a warning that such function or method shouldn't be used.

```
<?php

function obsoleteFoo() {
    throw new exception('Don\'t use obsoleteFoo() but rather the new version of foo().
↪');
}
?>
```

Suggestions

-

Specs

Short name	Functions/CantUse
Rulesets	none
Exakt since	1.8.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.528 Closure Could Be A Callback

Closure or arrowfunction could be simplified to a callback. Callbacks are strings or arrays.

A simple closure that only returns arguments relayed to another function or method, could be reduced to a simpler expression. They

Closure may be simplified with a string, for functioncall, with an array for methodcalls and static methodcalls.

Performances : simplifying a closure tends to reduce the call time by 50%.

```
<?php

// Simple and faster call to strtoupper
$filtered = array_map('strtoupper', $array);

// Here the closure doesn't add any feature over strtoupper
$filtered = array_map(function ($x) { return strtoupper($x);}, $array);

// Methodcall example : no fix
$filtered = array_map(function ($x) { return $x->strtoupper() ;}, $array);

// Methodcall example  : replace with array($y, 'strtoupper')
$filtered = array_map(function ($x) use ($y) { return $y->strtoupper($x) ;}, $array);

// Static methodcall example
```

(continues on next page)

(continued from previous page)

```
$filtered = array_map(function ($x) { return $x::strtoupper() ;}, $array);

// Static methodcall example   : replace with array('A', 'strtoupper')
$filtered = array_map(function ($x) { return A::strtoupper($x) ;}, $array);

?>
```

See also Closure class and Callbacks / Callables.

Suggestions

- Replace the closure by a string, with the name of the called function
- Replace the closure by an array, with the name of the called method and the object as first element

Specs

Short name	Functions/Closure2String
Rulesets	*Performances*, *Suggestions*
Exakt since	1.4.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Tine20*, *NextCloud*

13.2.529 Closures Glossary

List of all the closures in the code.

```
<?php

// A closure is also a unnamed function
$closure = function ($arg) { return 'A'.strtolower($arg); }

?>
```

See also The 'Closure Class <https://www.php.net/manual/en/class.closure.php>'_.

Specs

Short name	Functions/Closures
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.530 Conditioned Function

Indicates if a function is defined only if a condition is met.

<?php

// This is a conditioned function. // it only exists if the PHP binary doesn't have it already. if (!function_exists('join')) {

**function join($glue, $array) {** return implode($glue, $array);

}

}

Specs

Short name	Functions/ConditionedFunctions
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.531 Could Be Typehinted Callable

Those arguments may use the callable Typehint.

'callable' is a PHP keyword that represents callback functions. Those may be used in dynamic function call, like $function(); or as callback functions, like with array_map();

callable may be a string representing a function name or a static call (including ::), an array with two elements, (a class or object, and a method), or a closure.

When arguments are used to call a function, but are not marked with 'callable', they are reported by this analysis.

```
<?php

function foo(callable $callable) {
    // very simple callback
    return $callable();
}

function foo2($array, $callable) {
    // very simple callback
    return array_map($array, $callable);
}

?>
```

See also Callback / callable.

Suggestions

- Add the typehint callable

- Use the function is_callable() inside the method if 'callable' is too strong.

Specs

Short name	Functions/CouldBeCallable
Rulesets	none
Exakt since	0.10.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Magento*, *PrestaShop*

13.2.532 Could Be Static Closure

Closure and arrow functions may be static, and prevent the import of `$this`.

By preventing the useless import of `$this`, you avoid useless work.

This also has the added value to prevent the usage of `$this` from the closure. This is a good security practice.

```
<?php

class Foo
{
    function __construct()
    {

        // Not possible to use $this
        $func = static function() {
            var_dump($this);
        };
        $func();

        // Normal import of $this
        $closure = function() {
            var_dump($this);
        };
    }
};
new Foo();

?>
```

This is a micro-optimisation. Apply it in case of intensive usage.

See also Anonymous functions, GeneratedHydrator and Static anonymous functions <https://www.php.net/manual/en/functions.anonymous.php#functions.anonymous-functions.'static>'_.

Suggestions

- Add the static keyword to the closure.
- Make actual usage of $this in the closure.

Specs

Short name	Functions/CouldBeStaticClosure
Rulesets	*Suggestions*
Exakt since	1.3.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Piwigo*

13.2.533 Could Make A Function

> When a function is called across the code with the same arguments often enough, it should be turned into a local API.

This approach is similar to turning literals into constants : it centralize the value, it helps refactoring by updating it. It also makes the code more readable. Moreover, it often highlight common grounds between remote code locations.

The analysis looks for functions calls, and checks the arguments. When the calls occurs more than 4 times, it is reported.

```
<?php

// str_replace is used to clean '&' from strings.
// It should be upgraded to a central function
function foo($arg ) {
    $arg = str_replace('&', '', $arg);
    // do something with $arg
}

class y {
    function bar($database ) {
        $value = $database->queryName();
        $value = str_replace('&', '', $value);
        // $value = removeAmpersand($value);
        // do something with $arg2
    }
}

// helper function
function removeAmpersand($string) {
    return str_replace('&', '', $string);
}

?>
```

See also Don't repeat yourself (DRY).

Suggestions

- Create a constant for common pieces of data
- Create a function based on context-free repeated elements

- Create a class based on repeated elements with dependent values

Name	Default	Type	Description
centralizeThreshold	8	integer	Minimal number of calls of the function with one common argument.

Specs

Short name	Functions/CouldCentralize
Rulesets	*Analyze*, *Suggestions*
Exakt since	0.11.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.534 Could Typehint

Arguments that are tested with instanceof gain from making it a Typehint.

```
<?php

function foo($a, $b) {
    // $a is tested for B with instanceof.
    if (!$a instanceof B) {
        return;
    }

    // More code
}

function foo(B $a, $b) {
    // May omit the initial test

    // More code
}

?>
```

Suggestions

- Add the typehint, remove the test on the type

Specs

Short name	Functions/CouldTypehint
Rulesets	none
Exakt since	0.11.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.535 Could Type With Array

That argument may be typed with `array`.

```
<?php

// $a is used with a function which requires an int.
function foo($a) {
    return array_keys($a);
}

?>
```

See also Type declarations.

Suggestions

- Add the `array` typehint to the function.

Specs

Short name	Functions/CouldTypeWithArray
Rulesets	none
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.536 Could Type With Boolean

That argument may be typed with `bool`.

```
<?php

// $a is used with a function which requires a boolean.
function foo($code, $a) {
    return var_dump($code, $a);
}
```

(continues on next page)

(continued from previous page)

```
?>
```

See also Type declarations.

Suggestions

- Add the `bool` typehint to the function.

Specs

Short name	Functions/CouldTypeWithBool
Rulesets	none
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.537 Could Type With Int

That argument may be typed with `int`.

```
<?php

// $a is used with a function which requires an int.
function foo($a) {
    return chr($a);
}

?>
```

See also Type declarations.

Suggestions

- Add the `int` typehint to the function.

Specs

Short name	Functions/CouldTypeWithInt
Rulesets	none
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.538 Could Type With Iterable

Suggest using `iterable` typehint for arguments.

`iterable` represents both `array` and objects that implements `Iterator` interface. Both types are coerced, and usable here.

```
<?php

// $s may be both an array or an iterator
function foo($s) : int {
    $t = 0;
    foreach($s as $v) {
        $t += (int) $v;
    }

    return $t;
}

?>
```

See also Iterables.

Suggestions

- Add the iterable type

Specs

Short name	Functions/CouldTypeWithIterable
Rulesets	none
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.539 Could Type With String

That argument may be typed with `string`.

```
<?php

// $a is used with a function which requires a string.
function foo($a) {
    return strtolower($a);
}

?>
```

See also Type declarations.

Suggestions

- Add the `string` typehint to the function.

Specs

Short name	Functions/CouldTypeWithString
Rulesets	none
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.540 Deep Definitions

Structures, such as functions, classes, interfaces, traits, etc. may be defined anywhere in the code, including inside functions. This is legit code for PHP.

Since the availability of autoload, with spl_register_autoload(), there is no need for that kind of code. Structures should be defined, and accessible to the autoloading. Inclusions and deep definitions should be avoided, as they compel code to load some definitions, while autoloading will only load them if needed.

```
<?php

class X {
    function init() {
        // myFunction is defined when and only if X::init() is called.
        if (!function_exists('myFunction'){
            function myFunction($a) {
                return $a + 1;
            }
        })
    }
}

?>
```

Functions are excluded from autoload, but shall be gathered in libraries, and not hidden inside other code.

Constants definitions are tolerated inside functions : they may be used for avoiding repeat, or noting the usage of such function.

Definitions inside a if/then statement, that include PHP version check are accepted here.

See also Autoloading Classes.

Suggestions

- Move function definitions to the global space : outside structures, and method.

Specs

Short name	Functions/DeepDefinitions
Rulesets	*Analyze*, *CE*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Dolphin*

13.2.541 Dont Collect Void

When a method returns void, there is no need to collect the result. The collected value will actually be `null`.

```
<?php

function foo() : void {
    // doSomething()
}

// This is useless
$result = foo();

// This is useless. It looks like this is a left over from code refactoring
echo foo();

?>
```

Suggestions

- Move the call to the function to its own expression with a semi-colon.
- Remove assignation of the result of such calls.

Specs

Short name	Functions/DontUseVoid
Rulesets	*Analyze*
Exakt since	2.0.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.542 Dynamic Function Call

Mark a functioncall made with a variable name.

```php
<?php

// function definition
function foo() {}

// function name is in a variable, as a string.
$var = 'foo';

// dynamic call of a function
$var();

call_user_func($var);

?>
```

Specs

Short name	Functions/Dynamiccall
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.543 Function With Dynamic Code

Mark a method, function, closure, arrow function that includes dynamic code.

Dynamic code is based on usage of include() and co, extract() and eval().

```php
<?php

// Function with dynamic code
function foo($x) {
    include $x;
    return $y;
}

// Static coe Function
function foo($x) {
    return $y + $x;
}

?>
```

This is a support rule, to help omits some special cases in other rules.

Specs

Short name	Functions/DynamicCode
Rulesets	none
Exakt since	2.1.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.544 Empty Function

Function or method whose body is empty.

Such functions or methods are rarely useful. As a bare minimum, the function should return some useful value, even if constant.

A method is considered empty when it contains nothing, or contains expressions without impact.

```
<?php

// classic empty function
function emptyFunction() {}

class bar {
    // classic empty method
    function emptyMethod() {}

    // classic empty function
    function emptyMethodWithParent() {}
}

class barbar extends bar {
    // NOT an empty method : it overwrites the parent method
    function emptyMethodWithParent() {}
}

?>
```

Methods which overwrite another methods are omitted. Methods which are the concrete version of an abstract method are considered.

Suggestions

- Fill the function with actual code
- Remove any usage of the function, then remove the function

Specs

Short name	Functions/EmptyFunction
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Contao*

13.2.545 Exceeding Typehint

The typehint is not fully used in the method. Some of the defined methods in the typehint are unused. A tighter typehint could be used, to avoid method pollution.

```
<?php

interface i {
    function i1();
    function i2();
}

interface j {
    function j1();
    function j2();
}

function foo(i $a, j $b) {
    // the i typehint is totally used
    $a->i1();
    $a->i2();

    // the i typehint is not totally used : j2() is not used.
    $b->j1();
}

?>
```

Tight typehint prevents the argument from doing too much. They also require more maintenance : creation of dedicated interfaces, method management to keep all typehint tight.

See also Functions/InsufficientTypehint.

Suggestions

- Keep the typehint tight, do not inject more than needed.

Specs

Short name	Functions/ExceedingTypehint
Rulesets	*ClassReview*
Exakt since	2.0.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.546 Fallback Function

A function that is called with its name alone, and whose definition is in the global scope.

```
<?php

namespace {
    // global definition
    function foo() {}
}

namespace Bar {
    // local definition
    function foo2() {}

    foo(); // definition is in the global namespace
    foo2(); // definition is in the Bar namespace
}

?>
```

See also Using namespaces: fallback to global function/constant.

Specs

Short name	Functions/FallbackFunction
Rulesets	*CE*
Exakt since	1.1.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.547 Fn Argument Variable Confusion

Avoid using local variables as arrow function arguments.

When a local variable name is used as an argument's name in an arrow function, the local variable from the original scope is not imported. They are now two distinct variables.

When the local variable is not listed as argument, it is then imported in the arrow function.

```
<?php

function foo() {
    $locale = 1;

    // Actually ignores the argument, and returns the local variable ``$locale``.
    $fn2 = fn ($argument) => $locale;

    // Seems similar to above, but returns the incoming argument
    $fn2 = fn ($locale) => $locale;
}

?>
```

See also Arrow functions.

Suggestions

- Change the name of the local variable
- Change the name of the argument

Specs

Short name	Functions/FnArgumentVariableConfusion
Rulesets	*Analyze*, *Semantics*
Exakt since	2.1.0
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.548 func_get_arg() Modified

func_get_arg() and func_get_args() used to report the calling value of the argument until PHP 7. Since PHP 7, it is reporting the value of the argument at calling time, which may have been modified by a previous instruction.

```
<?php

function x($a) {
    $a++;
    print func_get_arg(0);
}

x(0);
?>
```

This code will display 1 in PHP 7, and 0 in PHP 5.

Specs

Short name	Functions/funcGetArgModified
Rulesets	*Analyze*, *CompatibilityPHP70*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.549 Function Called With Other Case Than Defined

Functions and methods are defined with a specific case. Often, this is done on purpose,
either to distinguish the method from others, such as PHP natives functions, or to follow a naming convention.
PHP functions are case insensitive, which leads to situations like :

```
<?php
  function myUtility($arg) {
    /* some code here */
  }

   myutility($var);
?>
```

It is recommended to use the same casing in the function call and the function definition.

Suggestions

- Use the same case for the function and its call.

Specs

Short name	Functions/FunctionCalledWithOtherCase
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.550 Functions Glossary

List of all the defined functions in the code.

```
<?php

// A function
function aFunction() {}
```

(continues on next page)

(continued from previous page)

```
// Closures (not reported)
$closure = function ($arg) {  }

// Methods
class foo {
    function aMethod() {}
}

?>
```

Specs

Short name	Functions/Functionnames
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.551 Functions Using Reference

Functions and methods using references in their signature.

```
<?php

function usingReferences( &$a) {}

class foo {
    public function methodUsingReferences($b, &$c = 1) {}
}
?>
```

Specs

Short name	Functions/FunctionsUsingReference
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.552 Generator Cannot Return

Generators could not use return and yield at the same time. In PHP 7.0, generator can now use both of them.

```php
<?php

// This is not allowed until PHP 7.0
function foo() {
    yield 1;
    return 'b';
}

?>
```

Suggestions

- Remove the return

Specs

Short name	Functions/GeneratorCannotReturn
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	1.8.7
Php Version	7.0+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.553 Hardcoded Passwords

Hardcoded passwords in the code.

Hardcoding passwords is a bad idea. Not only it make the code difficult to change, but it is an information leak. It is better to hide this kind of information out of the code.

```php
<?php

$ftp_server = '300.1.2.3';   // yes, this doesn't exists, it's an example
$conn_id = ftp_connect($ftp_server);

// login with username and password
$login_result = ftp_login($conn_id, 'login', 'password');

?>
```

See also 10 GitHub Security Best Practices and Git How-To: Remove Your Password from a Repository.

Suggestions

- Remove all passwords from the code. Also, check for history if you are using a VCS.

Name	Default	Type	Description
passwordsKeys	password_keys.json	data	List of array index and property names that shall be checked for potential secret key storages.

Specs

Short name	Functions/HardcodedPasswords
Rulesets	*Analyze*, *Security*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
ClearPHP	no-hardcoded-credential

13.2.554 Method Has Fluent Interface

Mark a method when it only returns $this.

Fluent interfaces allows for chaining methods calls. This implies that $this is always returned, so that the next method call is done on the same object.

```
<?php

$object = new foo();
$object->this()
       ->is()
       ->a()
       ->fluent()
       ->interface();

class foo {
    function this() {
        // doSomething
        return $this;
    }

    function is() {
        // doSomethingElse
        return $this;
    }

    /// Etc. for a(), fluent(), interface()...
}

?>
```

See also Fluent Interfaces in PHP and Fluent Interfaces are Evil.

Specs

Short name	Functions/HasFluentInterface
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.555 Method Has No Fluent Interface

Mark a method as such when it contains at least one return that doesn't return $this.

Specs

Short name	Functions/HasNotFluentInterface
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.556 Insufficient Typehint

An argument is typehinted, but it actually calls methods that are not listed in the interface.

Classes may be implementing more methods than the one that are listed in the interface they also implements. This means that filtering objects with a typehint, but calling other methods will be solved at execution time : if the method is available, it will be used; if it is not, a fatal error is reported.

```
<?php

class x implements i {
    function methodI() {}
    function notInI() {}
}

interface i {
    function methodI();
}

function foo(i $x) {
    $x->methodI(); // this call is valid
    $x->notInI();  // this call is not garanteed
}
?>
```

Inspired by discussion with Brandon Savage.

Suggestions

- Extend the interface with the missing called methods
- Change the body of the function to use only the methods that are available in the interface
- Change the used objects so they don't depend on extra methods

Specs

Short name	Functions/InsufficientTypehint
Rulesets	*Analyze*, *Typechecks*
Exakt since	1.6.6
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.557 Is An Extension Function

This is an extension function.

```
<?php

// range is a native PHP function. It is always available
$array = range(0, 100);

// json_encode is an extension function : it requires that PHP was compile with ext/
↪json
echo json_encode($array);

?>
```

Almost every PHP extension defines extra functions. Nowadays, they are prefixed, like `mysqli_connect`, `ldap_close`, or `zlib_decode`. Sometimes, they are even in a namespace. Refer to the extension itself to learn more about its functions usage.

Specs

Short name	Functions/IsExtFunction
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.558 Is Generator

Mark as such functions or methods that are using yield and yield from.

```
<?php

function generator() {
    yield from generator2();

    return 3;
}

function generator2() {
    yield 1;
    yield 2;
}

?>
```

Specs

Short name	Functions/IsGenerator
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.559 Functioncall Is Global

Marks functioncall when they are global and not located in another function, class or trait (namespaces are OK).

Specs

Short name	Functions/IsGlobal
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.560 Exit-like Methods

Those methods terminate the execution.

They are detected when they do call exit() <https://www.php.net/‘exit>‘_ or die() <https://www.php.net/‘die>‘_. They may also be identified with the PHP 8.0 *#[NoReturn]* attribute, or the PHPDOC `@noreturn` (case insensitive).

If they are called, they will stop the application. They are a user-land equivalent of exit or die.

```php
<?php

// This function anytime the code has finished its processing.
function finish() {
    global $html;

    echo $html;
    die();
}

?>
```

See also PhpStorm 2020.3 EAP #4: Custom PHP 8 Attributes.

Specs

Short name	Functions/KillsApp
Rulesets	*Attributes*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Medium

13.2.561 Functions In Loop Calls

The following functions call each-other in a loop fashion : A -> B -> A.

When those functions have no other interaction, the code is useless and should be dropped.

```php
<?php

function foo1($a) {
    if ($a < 1000) {
        return foo2($a + 1);
    }
    return $a;
}

function foo2($a) {
    if ($a < 1000) {
        return foo1($a + 1);
    }
    return $a;
}

// if foo1 nor foo2 are called, then this is dead code.
// if foo1 or foo2 are called, this recursive call should be investigated.

?>
```

Loops of size 2, 3 and 4 function are supported by this analyzer.

Suggestions

- Drop all the functions in the loop, as they are dead code

Specs

Short name	Functions/LoopCalling
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.562 Mark Callable

Create an attribute that guess what are the called function or methods, when possible.

Specs

Short name	Functions/MarkCallable
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.563 Mismatched Default Arguments

Arguments are relayed from one method to the other, and the arguments have different default values.

Although it is possible to have different default values, it is worth checking why this is actually the case.

```
<?php

function foo($a = null, $b = array() ) {
    // foo method calls directly bar.
    // When argument are provided, it's OK
    // When argument are omited, the default value is not the same as the next method
    bar($a, $b);
}

function bar($c = 1, $d = array() ) {

}

?>
```

Suggestions

- Synchronize default values to avoid surprises
- Drop some of the default values

Specs

Short name	Functions/MismatchedDefaultArguments
Rulesets	*Analyze*, *Typechecks*
Exakt since	0.12.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*SPIP*

13.2.564 Mismatched Typehint

Relayed arguments don't have the same typehint.

Typehint acts as a filter method. When an object is checked with a first class, and then checked again with a second distinct class, the whole process is always false : $a can't be of two different classes at the same time.

```
<?php

// Foo() calls bar()
function foo(A $a, B $b) {
    bar($a, $b);
}

// $a is of A typehint in both methods, but
// $b is of B then BB typehing
function bar(A $a, BB $b) {

}

?>
```

Note : This analysis currently doesn't check generalisation of classes : for example, when B is a child of BB, it is still reported as a mismatch.

Suggestions

- Ensure that the default value match the expected typehint.

Specs

Short name	Functions/MismatchedTypehint
Rulesets	*Analyze*, *Typechecks*
Exakt since	0.12.3
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*WordPress*

13.2.565 Mismatch Parameter And Type

When the name of the parameter contradicts the type of the parameter.

This is mostly semantics, so it will affect the coder and the auditor of the code. PHP is immune to those errors.

```
<?php

// There is a discrepancy between the typehint and the name of the variable
function foo(int $string) { }

// The parameter name is practising coding convention typehints
function bar(int $int) { }

?>
```

Suggestions

- Synch the name of the parameter and the typehint.

Specs

Short name	Functions/MismatchParameterAndType
Rulesets	*Analyze*, *Semantics*
Exakt since	2.1.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.566 Mismatch Parameter Name

Parameter name change in overwritten method. This may lead to errors when using PHP 8.0 named arguments.

PHP use the name of the parameter in the method whose code is executed. When the name change between the method and the overwritten method, the consistency is broken.

```
<?php

class x {
    function getValue($name) {}
}

class y extends x {
    // consistent with the method above
    function getValue($name) {}
}

class z extends x {
    // inconsistent with the method above
    function getValue($label) {}
}

?>
```

Here is another example, in early PHP 8.0 (courtesy of Carnage).

```
<?php

interface Pager
{
    public function fetch($page = 0, ...$categories);
}

class DbPager implements Pager
{
    public function fetch($seite = 0, ...$kategorien)
    {
        var_dump($kategorien);
    }
}

$dbPager = new DbPager();
$dbPager->fetch(page: 1, categories: 2);

?>
```

Suggestions

- Make sure all the names are the same, between methods

Specs

Short name	Functions/MismatchParameterName
Rulesets	*Analyze*, *CE*, *CompatibilityPHP80*
Exakt since	2.1.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.567 Mismatch Type And Default

The argument typehint and its default value don't match.

The code may lint and load, and even work when the arguments are provided. Though, PHP won't eventually execute it.

Most of the mismatch problems are caught by PHP at linting time. It displays the following error message : 'Argument 1 passed to foo() must be of the type integer, string given'.

The default value may be a constant (normal or class constant) : as such, PHP might find its value only at execution time, from another include. As such, PHP doesn't report anything about the situation at compile time.

The default value may also be a constant scalar expression : since PHP 7, some of the simple operators such as +, -, , %, '* <https://www.php.net/manual/en/language.operators.arithmetic.php>'_, etc. are available to build default values. Among them, the ternary operator and Coalesce. Again, those expression may be only evaluated at execution time, when the value of the constants are known.

```
<?php

// bad definition : the string is actually an integer
const STRING = 3;

function foo(string $s = STRING) {
    echo $s;
}

// works without problem
foo('string');

// Fatal error at compile time
foo();

// Fail only at execution time (missing D), and when default is needed
function foo2(string $s = D ? null : array()) {
    echo $s;
}

?>
```

PHP reports typehint and default mismatch at compilation time, unless there is a static expression that can't be resolved within the compiled file : then it is checked only at runtime, leading to a Fatal error.

See also Type declarations.

Suggestions

- Match the typehint with the default value
- Do not rely on PHP type juggling to change the type on the fly

Specs

Short name	Functions/MismatchTypeAndDefault
Rulesets	*Analyze*, *LintButWontExec*, *Typechecks*
Exakt since	1.2.9
Php Version	All
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	Medium

13.2.568 Missing Typehint

No typehint was found for this parameter, or as a return type for the function.

void is considered a specified typehint, and is not reported here.

```
<?php

function foo($no_typehint) : void {}

function no_return_type() {}

?>
```

See also Type Declaration.

Suggestions

-

Specs

Short name	Functions/MissingTypehint
Rulesets	*Typechecks*
Exakt since	2.0.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.569 Modified Typed Parameter

Reports modified parameters, which have a non-scalar typehint. Such variables should not be changed within the body of the method. Unlike typed properties, which always hold the expected type, typed parameters are only guaranteed type at the beginning of the method block.

```
<?php

class x {
```

(continues on next page)

(continued from previous page)

```
    function foo(Y $y) {
        // $y is type Y

        // A cast version of $y is stored into $yAsString. $y is untouched.
        $yAsString = (string) $y;

        // $y is of type 'int', now.
        $y = 1;

        // Some more code

        // display the string version.
        echo $yAsString;
        // so, Y $y is now raising an error
        echo $y->name;
    }
}

?>
```

This problem doesn't apply to scalar types : by default, PHP pass scalar parameters by value, not by reference. Class types are always passed by reference.

This problem is similar to **'Classes/DontUnsetProperties'_** : the static specification of the property may be unset, leading to confusing 'undefined property', while the class hold the property definition.

Suggestions

- Use different variable names when converting a parameter to a different type.
- Only use methods and properties calls on a typed parameter.

Specs

Short name	Functions/ModifyTypedParameter
Rulesets	*Analyze*, *ClassReview*
Exakt since	2.1.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.570 Multiple Functions Declarations

Some functions are declared multiple times in the code.

PHP accepts multiple definitions for the same functions, as long as they are not in the same file (linting error), or not included simultaneously during the execution.

This creates to several situations in which the same functions are defined multiple times : the function may be compatible with various PHP version, but their implementation may not. Or the function is part of a larger library, and sometimes only need without the rest of the library.

It is recommended to avoid having several functions with the same name in one repository. Turn those functions into methods and load them when needed.

```
<?php

namespace a {
    function foo() {}
}

// Other file
namespace a {
    function foo() {}
    function bar() {}
}


?>
```

Specs

Short name	Functions/MultipleDeclarations
Rulesets	*CE*
Exakt since	0.12.0
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.571 Multiple Identical Closure

Several closures are defined with the same code.

It may be interesting to check if a named function could be defined from them.

```
<?php

// the first squares, with closure
$squares= array_map(function ($a) {return $a * $a; }, range(0, 10) );

// later, in another file...
// another identical closure
$squaring = function ($x) { return $x * $x; };
foo($x, $squaring);

?>
```

This analysis also reports functions and methods that look like the closures : they may be considered for switch.

Suggestions

- Create a function with the body of those closures, and replace the closures by the function's name.

Specs

Short name	Functions/MultipleIdenticalClosure
Rulesets	none
Exakt since	1.5.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Medium

13.2.572 Multiple Returns

Functions and methods that have multiple return statement.

This makes it difficult to maintain : since the function may be short-circuited early, some later instruction may be omitted.

Ideally, guard clauses, which check if arguments are valid or not at the beginning of the method are the only exception to this rule.

```
<?php

function foo() {
    // This is a guard clause, that checks arguments.
    if ($a < 0) {
        return false;
    }

    $b = 0;
    for($i = 0; $i < $a; $i++) {
        $b += bar($i);
    }

    return $b;
}
?>
```

Currently, the engine doesn't spot guard clauses.

See also Single Function 'Exit Point <http://wiki.c2.com/?SingleFunctionExitPoint>'_.

Specs

Short name	Functions/MultipleReturn
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.573 Multiple Definition Of The Same Argument

A method's signature is holding twice (or more) the same argument. For example, function x ($a, $a) { ... }.

This is accepted as is by PHP 5, and the last parameter's value will be assigned to the variable. PHP 7.0 and more recent has dropped this feature, and reports a fatal error when linting the code.

```
<?php
  function x ($a, $a) { print $a; };
  x(1,2); => display 2

    // special case with a closure :
  function ($a) use ($a) { print $a; };
  x(1,2); => display 2

?>
```

However, this is not common programming practise : all arguments should be named differently.

See also Prepare for PHP 7 error messages (part 3).

Suggestions

- Give different names to different parameters

Specs

Short name	Functions/MultipleSameArguments
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	7.0-
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	all-unique-arguments

13.2.574 Must Return Methods

The following methods are expected to return a value that will be used later. Without return, they are useless.

Methods that must return are : __get(), __isset(), __sleep(), __toString(), __set_state(), __invoke(), __debugInfo(). Methods that may not return, but are often expected to : __call(), __callStatic().

```
<?php

class foo {
    public function __isset($a) {
        // returning something useful
        return isset($this->$var[$a]);
    }
```

(continues on next page)

(continued from previous page)

```
    public function __get($a) {
        $this->$a++;
        // not returning...
    }

    public function __call($name, $args) {
        $this->$name(...$args);
        // not returning anything, but that's OK
    }

}
?>
```

Suggestions

- Add a return expression, with a valid data type
- Remove the return typehint

Specs

Short name	Functions/MustReturn
Rulesets	*Analyze*, *CI-checks*, *LintButWontExec*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.575 Never Used Parameter

When a parameter is never used at calltime, it may be turned into a local variable.

It seems that the parameter was set up initially, but never found its practical usage. It is never mentioned, and always fall back on its default value.

Parameter without a default value are reported by PHP, and are usually always filled.

```
<?php

// $b may be turned into a local var, it is unused
function foo($a, $b = 1) {
    return $a + $b;
}

// whenever foo is called, the 2nd arg is not mentionned
foo($a);
foo(3);
foo('a');
foo($c);

?>
```

Suggestions

- Drop the unused argument in the method definition
- Actually use the argument when calling the method
- Drop the default value, and check warnings that mention usage of this parameter

Specs

Short name	Functions/NeverUsedParameter
Rulesets	*Analyze*, *Suggestions*
Exakt since	1.0.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Piwigo*

13.2.576 No Boolean As Default

Default values should always be set up with a constant name.

Class constants and constants improve readability when calling the methods or comparing values and statuses.

```
<?php

const CASE_INSENSITIVE = true;
const CASE_SENSITIVE = false;

function foo($case_insensitive = true) {
    // doSomething()
}

// Readable code
foo(CASE_INSENSITIVE);
foo(CASE_SENSITIVE);


// unreadable code  : is true case insensitive or case sensitive ?
foo(true);
foo(false);

?>
```

See also FlagArgument and Clean code: The curse of a boolean parameter.

Suggestions

- Use constants or class constants to give value to a boolean literal
- When constants have been defined, use them when calling the code
- Split the method into two methods, one for each case

Specs

Short name	Functions/NoBooleanAsDefault
Rulesets	*Analyze*
Exakt since	0.10.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*OpenConf*

13.2.577 No Class As Typehint

Avoid using classes as typehint : always use interfaces. This way, different classes, or versions of classes may be passed as argument. The typehint is not linked to an implementation, but to signatures.

A class is needed when the object is with properties : interfaces do not allow the specifications of properties.

```
<?php

class X {
    public $p = 1;

    function foo() {}
}

interface i {
    function foo();
}

// X is a class : any update in the code requires changing / subclassing X or the␣
↪rest of the code.
function bar(X $x) {
    $x->foo();
}

// I is an interface : X may implements this interface without refactoring and pass
// later, newer versions of X may get another name, but still implement I, and still␣
↪pass
function bar2(I $x) {
    $x->foo();
}

function bar3(I $x) {
    echo $x->p;
}

?>
```

See also Type hinting for interfaces.

Suggestions

- Create an interface with the important methods, and use that interface

- Create an abstract class, when public properties are also needed

Specs

Short name	Functions/NoClassAsTypehint
Rulesets	*Typechecks*
Exakt since	0.11.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Vanilla*, *phpMyAdmin*

13.2.578 No Literal For Reference

Method arguments and return values may be by reference. Then, they need to be a valid variable.

Objects are always passed by reference, so there is no need to explicitly declare it.

Expressions, including ternary operator, produce value, and can't be used by reference directly. This is also the case for expression that include one or more reference.

```
<?php

// variables, properties, static properties, array items are all possible
$a = 1;
foo($a);

//This is not possible, as a literal can't be a reference
foo(1);

function foo(&$int) { return $int; }


// This is not a valid reference
function &bar() { return 2; }
function &bar2() { return 2 + $r; }

?>
```

Wrongly passing a value as a reference leads to a PHP Notice.

See also References.

Suggestions

- Remove the reference in the method signature (argument or return value)
- Make the argument an object, by using a typehint (non-scalar)
- Put the value into a variable prior to call (or return) the method

Specs

Short name	Functions/NoLiteralForReference
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.9.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.579 No Return Used

The return value of the following functions are never used. The return argument may be dropped from the code, as it is dead code.

This analysis supports functions and static methods, when a definition may be found. It doesn't support method calls.

```
<?php

function foo($a = 1;) { return 1; }
foo();
foo();
foo();
foo();
foo();
foo();

// This function doesn't return anything.
function foo2() { }

// The following function are used in an expression, thus the return is important
function foo3() {  return 1;}
function foo4() {  return 1;}
function foo5() {  return 1;}

foo3() + 1;
$a = foo4();
foo(foo5());

?>
```

Suggestions

- Remove the return statement in the function
- Actually use the value returned by the method, for test or combination with other values

Specs

Short name	Functions/NoReturnUsed
Rulesets	*Analyze*, *Suggestions*
Exakt since	0.11.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*SPIP*, *LiveZilla*

13.2.580 Nullable With Constant

Arguments are automatically nullable with a literal null. They used to also be nullable with a constant null, before PHP 8.0.

```
<?php

// Extracted from https://github.com/php/php-src/blob/master/UPGRADING

// Replace
function test(int $arg = CONST_RESOLVING_TO_NULL) {}
// With
function test(?int $arg = CONST_RESOLVING_TO_NULL) {}
// Or
function test(int $arg = null) {}

?>
```

Suggestions

- Use the valid syntax

Specs

Short name	Functions/NullableWithConstant
Rulesets	*CE*, *CompatibilityPHP80*
Exakt since	2.1.9
Php Version	8.0-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.581 Nullable Without Check

Nullable typehinted argument should be checked before usage.

```php
<?php

// This will emit a fatal error when $a = null
function foo(?A $a) {
    return $a->m();
}

// This is stable
function foo(?A $a) {
    if ($a === null) {
        return 42;
    } else {
        return $a->m();
    }
}

?>
```

Suggestions

-

Specs

Short name	Functions/NullableWithoutCheck
Rulesets	*ClassReview*
Exakt since	2.0.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.582 One Letter Functions

One letter functions seems to be really short for a meaningful name. This may happens for very high usage functions, so as to keep code short, but such functions should be rare.

```php
<?php

// Always use a meaningful name
function addition($a, $b) {
    return $a + $b;
}

// One letter functions are rarely meaningful
function f($a, $b) {
    return $a + $b;
}

?>
```

Suggestions

- Use full names for functions
- Remove the function name altogether : use a closure

Specs

Short name	Functions/OneLetterFunctions
Rulesets	*Semantics*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*ThinkPHP*, *Cleverstyle*

13.2.583 Only Variable For Reference

When a method is requesting an argument to be a reference, it cannot be called with a literal value.

The call must be made with a variable, or any assimilated data container : array, property or static property.

```
<?php

// This is not possible
foo(1,2);

// This is working
foo($a, $b);

function foo($a, &$b) {}

?>
```

Note that PHP may detect this error at linting time, if the method is defined after being called : at that point, PHP will only check the problem during execution. This is definitely the case for methods, compared to functions or static methods.

See also Passing arguments by reference.

Suggestions

- Put the literal value in a variable before calling the method.
- Omit the arguments, when it won't be used.

Specs

Short name	Functions/OnlyVariableForReference
Rulesets	*Analyze*, *LintButWontExec*
Exakt since	1.4.6
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.584 Only Variable Passed By Reference

> When an argument is expected by reference, it is compulsory to provide a container. A container may be a variable, an array, a property or a static property.

This may be linted by PHP, when the function definition is in the same file as the function usage. This is silently linted if definition and usage are separated, if the call is dynamical or made as a method.

```
<?php

function foo(&$bar) { /**/ }

function &bar() { /**/ }

// This is not possible : strtolower() returns a value
foo(strtolower($string));

// This is valid : bar() returns a reference
foo(bar($string));

?>
```

This analysis currently covers functioncalls and static methodcalls, but omits methodcalls.

Suggestions

- Store the previous result in a variable, and then call the function.

Specs

Short name	Functions/OnlyVariablePassedByReference
Rulesets	*Analyze*
Exakt since	0.11.3
Php Version	All
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Dolphin*, *PhpIPAM*

13.2.585 Optional Parameter

An optional parameter is a method argument that has both a typehint and a default value.

Such argument is optional, as it may be omitted. When this is the case, the code has to differentiate between the default behavior or the actual usage. It is recommended to avoid providing a default value, and use a null object.

```
<?php

class foo {
    function methodWithOptionalArgument(bar $x = null) {
        if ($x === null) {
            // default behavior
        } else {
            // normal behavior
        }
    }

    function methodWithCompulsoryArgument(bar $x) {
        // normal behavior
        // $x is always a bar.
    }
}
?>
```

Specs

Short name	Functions/OptionalParameter
Rulesets	none
Exakt since	0.12.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.586 Parameter Hiding

When a parameter is set to another variable, and never used.

While this is a legit syntax, parameter hiding tends to make the code confusing. The parameter itself seems to be unused, while some extra variable appears.

Keep this code simple by removing the hiding parameter.

```
<?php

function substract($a, $b) {
    // $b is given to $c;
    $c = $b;

    $c is used, but $b would be the same
    return $a - $c;
}
```

(continues on next page)

(continued from previous page)

```
?>
```

Suggestions

- Remove the hiding parameter

Specs

Short name	Functions/ParameterHiding
Rulesets	*Semantics*
Exakt since	1.9.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.587 Prefix And Suffixes With Typehint

> This analysis checks the relationship between methods prefixes and suffixes, with their corresponding return typehint.

For example, a method with the signature `function isACustomer() {}` should return a boolean. That boolean can then be read when calling the method : `if ($user->isACustomer()) {}`.

There are multiple such conventions that may be applied. For example, `has*` should return a boolean, `set*` should return nothing (a.k.a `void`), and `get*` shall return any kind of type.

```
<?php

class x  {
    // Easy to read convention
    function isAUser() : bool {}

    // shall return a boolean
    function isACustomer() {}

    // shall return a string, based on suffix 'name => string'
    function getName() {}

    // shall return a string, based on suffix 'name => string'
    function getUsername() {}

    // shall return \Uuid, based on prefix 'uuid => \Uuid'
    function getUuid() {}

    // shall return anything, based on no prefix nor suffix
    function getBirthday() {}

}

?>
```

There are 2 parameters for this analysis. It is recommended to customize them to get an better results, related to the naming conventions used in the code.

`prefixedType` is used for prefix in method names, which is the beginning of the name. `suffixedType` is used for suffixes : the ending part of the name. Matching is case insensitive.

The prefix is configured as the index of the map, while the related type is configured as the value of the map.

`prefixToType['is'] = 'bool';` will be use as `is*` shall use the `bool` typehint.

Multiple typehints may be used at the same time. PHP supports multiple types since PHP 8.0, and Exakat will support them with any PHP version. Specify multiple types by separating them with comma. Any typehint not found in this list will be reported, including `null`.

PHP scalar types are available : `string`, `int`, `void`, etc. Explicit types, based on classes or interfaces, must use the fully qualified name, not the short name. `suffixToType['uuid'] = '\Uuid';` will be use as `*uuid` shall use the `\Uuid` typehint.

When multiple rules applies, only one is reported.

Suggestions

-

Name	Default	Type	Description
prefixedType	prefixedType['is'] = 'bool'; prefixedType['has'] = 'bool'; prefixedType['set'] = 'void'; prefixedType['list'] = 'array';	ini_hash	list of prefixes and their expected return-type
suffixedType	prefixedType['list'] = 'bool'; prefixedType['int'] = 'int'; prefixedType['string'] = 'string'; prefixedType['name'] = 'string'; prefixedType['description'] = 'string'; prefixedType['id'] = 'int'; prefixedType['uuid'] = 'Uuid';	ini_hash	list of suffixes and their expected return-type

Specs

Short name	Functions/PrefixToType
Rulesets	*Semantics*
Exakt since	2.1.1
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.588 Real Functions

Real functions, not methods.

Function keywords, that are not in a class, trait, interface, nor a closure.

```
<?php

// a real Function
```

(continues on next page)

(continued from previous page)

```
function realFunction () {}

// Those are not real functions
function ($closure) { }

class foo {
    function isAClassMethod() {}
}

interface fooi {
    function isAnInterfaceMethod() {}
}

trait foot {
    function isATraitMethod() {}
}
?>
```

Specs

Short name	Functions/RealFunctions
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.589 Recursive Functions

Recursive functions are functions that calls itself.

```
<?php

// a recursive function ; it calls itself
function factorial($n) {
    if ($n == 1) { return 1; }

    return factorial($n - 1) * $n;
}
?>
```

Methods are not handled here.

Specs

Short name	Functions/Recursive
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.590 Redeclared PHP Functions

Function that bear the same name as a PHP function, and that are declared.

This is useful when managing backward compatibility, like emulating an old function, or preparing for newer PHP versions, like emulating new upcoming function.

```
<?php

if (version_compare(PHP_VERSION, 7.0) > 0) {
    function split($separator, $string) {
        return explode($separator, $string);
    }
}

print_r( split(' ', '2 3'));

?>
```

Suggestions

- Check if it is still worth emulating that function

Specs

Short name	Functions/RedeclaredPhpFunction
Rulesets	*Analyze*, *CE*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.591 Relay Function

Relay function only delegate workload to another one.

Relay functions and methods are delegating the actual work to another function or method. They do not have any impact on the results, besides exposing another name for the same feature.

```
<?php

function myStrtolower($string) {
    return \strtolower($string);
}

?>
```

Relay functions are typical of transition API, where an old API have to be preserved until it is fully migrated. Then, they may be removed, so as to reduce confusion, and simplify the API.

Suggestions

- Remove relay function, call directly the final function
- Remove the target function, and move the code here
- Add more logic to that function, like conditions or cache

Specs

Short name	Functions/RelayFunction
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*TeamPass*, *SPIP*

13.2.592 Semantic Typing

Arguments names are only useful inside the method's body. They are not actual type.

```
<?php

// arguments should be a string and an array
function foo($array, $str) {
    // more code
    return $boolean;
}

// typehint is actually checking the values
function bar(iterable $closure) : bool {
    // more code
    return true;
}

?>
```

Suggestions

- Use a typehint to make sure the argument is of the expected type.

Specs

Short name	Functions/SemanticTyping
Rulesets	*Semantics*
Exakt since	2.0.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.593 Argument Should Be Typehinted

When a method expects objects as argument, those arguments should be typehinted. This way, it provides early warning that a wrong object is being sent to the method.

The analyzer will detect situations where a class, or the keywords 'array' or 'callable'.

```
<?php

// What are the possible classes that have a 'foo' method?
function foo($bar) {
    return $bar->foo();
}

?>
```

Closure arguments are omitted.

See also Type declarations.

Suggestions

- Add the typehint to the function arguments

Specs

Short name	Functions/ShouldBeTypehinted
Rulesets	*Typechecks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
ClearPHP	always-typehint
Examples	*Dolphin*, *Mautic*

13.2.594 Should Use Constants

The following functions have related constants that should be used as arguments, instead of scalar literals, such as integers or strings.

```
<?php

// The file is read and new lines are ignored.
$lines = file('file.txt', FILE_IGNORE_NEW_LINES)

// What is this doing, with 2 ?
$lines = file('file.txt', 2);

?>
```

See also Bitmask Constant Arguments in PHP.

Suggestions

- Use PHP native constants whenever possible, for better readability.

Specs

Short name	Functions/ShouldUseConstants
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Tine20*

13.2.595 Should Yield With Key

iterator_to_array() will overwrite generated values with the same key.

PHP generators are based on the `yield` keyword. They also delegate some generating to other methods, with `yield from`.

When delegating, `yield from` uses the keys that are generated with `yield`, and otherwise, it uses auto-generated index, starting with 0.

The trap is that each `yield from` reset the index generation and start again with 0. Coupled with iterator_to_array(), this means that the final generated array may lack some values, while a foreach() loop would yield all of them.

```
<?php

function g1() : Generator {
     for ( $i = 0; $i < 4; $i++ ) { yield $i; }
}

function g2() : Generator {
     for ( $i = 5; $i < 10; $i++ ) { yield $i; }
```

(continues on next page)

(continued from previous page)

```
}

function aggregator() : Generator {
     yield from g1();
     yield from g2();
}

print_r(iterator_to_array());

/*
Array
(
    [0] => 6
    [1] => 7
    [2] => 8
    [3] => 9
    [4] => 4  // Note that 4 and 5 still appears
    [5] => 5  // They are not overwritten by the second yield
)
*/


foreach ( aggregator() as $i ) {
     print $i.PHP_EOL;
}

/*
0  // Foreach has no overlap and yield it all.
1
2
3
4
5
6
7
8
9
*/

?>
```

Thanks to Holger Woltersdorf for pointing this.

See also Generator syntax and Yielding values with keys.

Suggestions

- Use iterator_to_array() on each generator separately, and use array_merge() to merge all the arrays.
- Always yield with distinct keys
- Avoid iterator_to_array() and use foreach()

Specs

Short name	Functions/ShouldYieldWithKey
Rulesets	*Analyze*, *CI-checks*, *Top10*
Exakt since	1.5.2
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.596 Too Many Local Variables

> Too many local variables were found in the methods. When over 15 variables are found in such a method, a violation is reported.

Local variables exclude globals (imported with global) and arguments. Local variable include static variables.

When too many variables are used in a function, it is a code smells. The function is trying to do too much and needs extra space for juggling. Beyond 15 variables, it becomes difficult to keep track of their name and usage, leading to confusion, overwriting or hijacking.

```
<?php

// This function is OK : 3 vars are arguments, 3 others are globals.
function a20a3g3($a1, $a2, $a3) {
    global $a4, $a5, $a6;

    $a1  = 1;
    $a2  = 2;
    $a3  = 3 ;
    $a4  = 4 ;
    $a5  = 5 ;
    $a6  = 6 ;
    $a7  = 7 ;
    $a8  = 8 ;
    $a9  = 9 ;
    $a10 = 10;
    $a11  = 11;
    $a12  = 12;
    $a13  = 13 ;
    $a14  = 14 ;
    $a15  = 15 ;
    $a16  = 16 ;
    $a17  = 17 ;
    $a18  = 18 ;
    $a19  = 19 ;
    $a20 = 20;

}

// This function has too many variables
function a20() {

    $a1  = 1;
    $a2  = 2;
```

(continues on next page)

(continued from previous page)

```
    $a3  = 3 ;
    $a4  = 4 ;
    $a5  = 5 ;
    $a6  = 6 ;
    $a7  = 7 ;
    $a8  = 8 ;
    $a9  = 9 ;
    $a10 = 10;
    $a11  = 11;
    $a12  = 12;
    $a13  = 13 ;
    $a14  = 14 ;
    $a15  = 15 ;
    $a16  = 16 ;
    $a17  = 17 ;
    $a18  = 18 ;
    $a19  = 19 ;
    $a20 = 20;

}

?>
```

Suggestions

- Remove some of the variables, and inline them
- Break the big function into smaller ones
- Find repeated code and make it a separate function

Name	Default	Type	Description
tooManyLocalVariableThreshold	15	integer	Minimal number of variables in one function or method to report.

Specs

Short name	Functions/TooManyLocalVariables
Rulesets	*Analyze*
Exakt since	0.9.2
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*HuMo-Gen*

13.2.597 Too Many Parameters

Method has too many parameters. Exakat has a default parameter count which may be configured.

A method that needs more than 8 parameters is trying to do too much : it should be reviewed and split into smaller methods.

```
<?php

// This methods has too many parameters.
function alertSomeone($name, $email, $title, $message, $attachements, $signature,
↪$bcc, $cc, $extra_headers) {
    /* too much code here */
}

?>
```

See also How many parameters is too many ? and Too Many Parameters.

Suggestions

- Reduce the number of parameters to a lower level
- Break the function into smaller functions
- Turn the function into a class

Name	Default	Type	Description
parametersCount	8	integer	Minimal number of parameters to report.

Specs

Short name	Functions/TooManyParameters
Rulesets	*Suggestions*
Exakt since	1.1.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*WordPress*, *ChurchCRM*

13.2.598 Too Much Indented

Reports methods that are using more than one level of indentation on average.

Indentations levels are counted for each for, foreach, if…then, while, do..while, try..catch..finally structure met. Compulsory expressions, such as conditions, are not counted in the total. Levels of indentation start at 0 (no indentation needed)

This analysis targets methods which are build around large conditions : the actual useful code is nested inside the branches of the if/then/else (for example).

The default threshold `indentationAverage` of 1 is a good start for spotting large methods with big conditional code, and will leave smaller methods, even when they only contain one if/then. Larger methods shall be refactored in smaller size.

The parameter `minimumSize` set aside methods which are too small for refactoring.

```php
<?php

// average 0
function foo0() {
    $a = rand(1,2);
    $a *= 3;

    return $a;
}

// average 0.66 = (0 + 1 + 1) / 3
function foo0_66() {
    // if () is at level 0
    if ($a == 2) { // condition is not counted
        $a = 1;    // level 1
    } else {
        $a = 2;    // level 1
    }
}

// average 1 = (0 + 2 + 1 + 1) / 4
function foo1() {
    // if () is at level 0
    if ($a == 2) {
        // if () is at level 1
        if ($a == 2) {
            $a = 1; // level 2
        }
        $a = 1; // level 1
    } else {
        $a = 2; // level 1
    }
}

?>
```

This analysis is distinct from Structures/MaxLevelOfIdentation, which only reports the highest level of indentation. This one reports how one method is build around one big

See also Structures/MaxLevelOfIdentation.

Suggestions

- Refactor the method to reduce the highest level of indentation
- Refactor the method move some of the code to external methods.

Name	Default	Type	Description
indentationAverage	1	real	Minimal average of indentation in a method to report. Default is 1.0, which means that the method is on average at one level of indentation or more.
minimumSize	3	real	Minimal number of expressions in a method to apply this analysis.

Specs

Short name	Functions/TooMuchIndented
Rulesets	*Suggestions*
Exakt since	2.1.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.599 Typehinted References

> Typehinted arguments have no need for references. Since they are only an object, they are already a reference.

In fact, adding the & on the argument definition may lead to error like `Only variables should be passed by reference`.

This applies to the `object` type hint, but not the the others, such as `int` or `bool`.

```
<?php
    // a class
    class X {
        public $a = 3;
    }

    // typehinted reference
    //function foo(object &$x) works too
    function foo(X &$x) {
        $x->a = 1;

        return $x;
    }

    // Send an object
    $y = foo(new X);

    // This prints 1;
    print $y->a;
?>
```

See also Passing by reference and Objects and references.

Suggestions

- Remove reference for typehinted arguments, unless the typehint is a scalar typehint.

Specs

Short name	Functions/TypehintedReferences
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.2.8
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.600 Typehint Must Be Returned

When using a typehint for a method, it is compulsory to use a at least one return in the method's body. This is true for nullable typehint too : `return` alone won't be sufficient.

```
<?php

// The function returns a value (here, correct object)
function foo() : Bar { return new Bar(); }

// The function should at least, return a value
function foo() : Bar { }

// The function should at least, return a value : Null or an object. Void, here, is␣
↪not acceptable.
function foo() : ?Bar { return; }

?>
```

PHP lint this, but won't execute it.

This analysis doesn't check if the returned value is compatible with the returned typehint. Only its presence is checked.

See also Return Type Declaration and Type hint in PHP function parameters and return values.

Suggestions

- Add a return with a valid value

Specs

Short name	Functions/TypehintMustBeReturned
Rulesets	*Analyze*, *CI-checks*, *LintButWontExec*
Exakt since	1.6.9
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.601 Typehints

List of all the types (classes or scalar) used in Typehinting.

```
<?php

// here, array, myObject and string are all typehints.
function foo(array $array, myObject $x, string $string) {

}

?>
```

See also Type declarations.

Specs

Short name	Functions/Typehints
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.602 Unbinding Closures

Never drop `$this`, once a closure was created in a non-static method.

From the PHP wiki : Currently it is possible to unbind the $this variable from a closure that originally had one by using $closure->bindTo(null). Due to the removal of static calls to non-static methods in PHP 8, we now have a guarantee that $this always exists inside non-static methods. We would like to have a similar guarantee that $this always exists for non-static closures declared inside non-static methods. Otherwise, we will end up imposing an unnecessary performance penalty either on $this accesses in general, or $this accesses inside such closures.

```
<?php

class x {
    private $a = 3;

    function foo() {
        return function () { echo $this->a; };
    }
}

$closure = (new x)->foo();

// $this was expected, and it is not anymore
$closure->bindTo(null);

$closure->bindTo(new x);

?>
```

Calling bindTo() with a valid object is still valid.

See also Unbinding '$this from non-static closures <https://wiki.php.net/rfc/deprecations_php_7_4#unbinding_this_from_non-static_closures>'_.

Suggestions

- Create a static closure, which doesn't rely on $this at all
- Remove the call to bindTo(null).

Specs

Short name	Functions/UnbindingClosures
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.9.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.603 Undefined Functions

Some functions are called, but not defined in the code. This means that the functions are probably defined in a missing library, or in an extension. If not, this will yield a Fatal error at execution.

```
<?php

// Undefined function
foo($a);

// valid function, as it belongs to the ext/yaml extension
$parsed = yaml_parse($yaml);

// This function is not defined in the a\b\c namespace, nor in the global namespace
a\b\c\foo();

?>
```

See also Functions.

Suggestions

- Fix the name of the function in the code
- Remove the functioncall in the code
- Define the function for the code to call it
- Include the correct library in the code source

Specs

Short name	Functions/UndefinedFunctions
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.604 Unknown Parameter Name

The name of the parameter doesn't belong to the method signature. Named arguments was introduced in PHP 8.0.

```
<?php

// All good
foo(a:1, b:2, c:3);
foo(...['a':1, 'b':2, 'c':3]);

// A is not a parameter name, it should be a : names are case sensitive
foo(A:1, b:2, c:3);
foo(...['A':1, 'b':2, 'c':3]);

function foo($a, $b, $c) {}
?>
```

See also Named Arguments.

Suggestions

- Fix the name of the parameter and use a valid one
- Remove the parameter name, and revert to positional notation

Specs

Short name	Functions/UnknownParameterName
Rulesets	*Analyze*, *CI-checks*
Exakt since	2.1.6
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.605 Unset Arguments

There is no need to unset arguments. Those values will be freed at the end of the function anyhow.

```
<?php

function foo($a, $b) {
    $b = $a * 2;
    // This is useless. $a will be freed at the end of the function.
    unset($a);
}

?>
```

Specs

Short name	Functions/UnsetOnArguments
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.606 Unused Arguments

Those arguments are not used in the method or function.

Unused arguments should be removed in functions : they are just dead code.

Unused argument may have to stay in methods, as the signature is actually defined in the parent class.

```
<?php

// $unused is in the signature, but not used.
function foo($unused, $b, $c) {
    return $b + $c;
}
?>
```

Suggestions

- Drop the argument from the signature
- Actually use that argument in the body of the method

Specs

Short name	Functions/UnusedArguments
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*ThinkPHP*, *phpMyAdmin*

13.2.607 Unused Functions

The functions below are unused. They look like dead code.

Recursive functions, level 1, are detected : they are only reported when a call from outside the function is made. Recursive functions calls of higher level (A calls B calls A) are not handled.

```
<?php

function used() {}
// The 'unused' function is defined but never called
function unused() {}

// The 'used' function is called at least once
used();

?>
```

Suggestions

- Use the function in the code
- Remove the functions from the code

Specs

Short name	Functions/UnusedFunctions
Rulesets	*Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Woocommerce*, *Piwigo*

13.2.608 Unused Inherited Variable In Closure

Some closures forgot to make usage of inherited variables.

Closure have two separate set of incoming variables : the arguments (between parenthesis) and the inherited variables, in the 'use' clause. Inherited variables are extracted from the local environment at creation time, and keep their value until execution.

The reported closures are requesting some local variables, but do not make any usage of them. They may be considered as dead code.

```
<?php

// In this closure, $y is forgotten, but $u is used.
$a = function ($y) use ($u) { return $u; };

// In this closure, $u is forgotten
$a = function ($y, $z) use ($u) { return $u; };

?>
```

See also Anonymous functions.

Suggestions

- Remove the unused inherited variable
- Make us of the unused inherited variable

Specs

Short name	Functions/UnusedInheritedVariable
Rulesets	*Analyze*, *CI-checks*, *Dead code*
Exakt since	1.0.11
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*shopware*, *Mautic*

13.2.609 Unused Returned Value

The function called returns a value, which is ignored.

Usually, this is a sign of dead code, or a missed check on the results of the functioncall. At times, it may be a valid call if the function has voluntarily no return value.

It is recommended to add a check on the return value, or remove the call.

```
<?php

// simplest form
function foo() {
    return 1;
}

foo();
```

(continues on next page)

(continued from previous page)

```
// In case of multiple return, any one that returns something means that return value␣
↪is meaningful
function bar() {
    if (rand(0, 1)) {
        return 1;
    } else {
        return ;
    }
}

bar();

?>
```

Note that this analysis ignores functions that return void (same meaning that PHP 7.1 : return ; or no return in the function body).

Specs

Short name	Functions/UnusedReturnedValue
Rulesets	*Analyze*, *Dead code*
Exakt since	0.8.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.610 Use Arrow Functions

Arrow functions are closures that require less code to write.

Arrow functions were introduced in PHP 7.4. They added the reserved keyword `fn`. s

```
<?php

array_map(fn(A $b): int => $b->c, $array);

function array_values_from_keys($arr, $keys) {
    return array_map(fn($x) => $arr[$x], $keys);
}
?>
```

See also RFC : Arrow functions and Arrow functions in PHP.

Specs

Short name	Functions/UseArrowFunctions
Rulesets	*CE*
Exakt since	1.9.4
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.611 Use Constant As Arguments

Some methods and functions are defined to be used with constants as arguments. Those constants are made to be meaningful and readable, keeping the code maintenable. It is recommended to use such constants as soon as they are documented.

```
<?php

// Turn off all error reporting
// 0 and -1 are accepted
error_reporting(0);

// Report simple running errors
error_reporting(E_ERROR | E_WARNING | E_PARSE);

// The first argument can be one of INPUT_GET, INPUT_POST, INPUT_COOKIE, INPUT_SERVER,
↪ or INPUT_ENV.
$search_html = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_SPECIAL_CHARS);

// sort accepts one of SORT_REGULAR, SORT_NUMERIC, SORT_STRING, SORT_LOCALE_STRING,␣
↪SORT_NATURAL
// SORT_FLAG_CASE may be added, and combined with SORT_STRING or SORT_NATURAL
sort($fruits);

?>
```

Here is the list of function that use a unique PHP constant as argument :

- array_change_key_case()
- array_multisort()
- array_unique()
- count()
- dns_get_record()
- easter_days()
- extract()
- filter_input()
- filter_var()
- fseek()
- get_html_translation_table()

- gmp_div_q()
- gmp_div_qr()
- gmp_div_r()
- html_entity_decode()
- htmlspecialchars_decode()
- http_build_query()
- http_parse_cookie()
- http_parse_params()
- http_redirect()
- http_support()
- parse_ini_file()
- parse_ini_string()
- parse_url()
- pathinfo()
- pg_select()
- posix_access()
- round()
- scandir()
- socket_read()
- str_pad()
- trigger_error()

Here is the list of functions that use a combination of PHP native functions as argument.

- arsort()
- asort()
- error_reporting()
- filter_input()
- filter_var()
- get_html_translation_table()
- htmlentities()
- htmlspecialchars()
- http_build_url()
- jdtojewish()
- krsort()
- ksort()
- pg_result_status()
- phpcredits()

- phpinfo()
- preg_grep()
- preg_match()
- preg_split()
- rsort()
- runkit_import()
- sort()
- stream_socket_client()
- stream_socket_server()

Suggestions

- Use PHP native constants, whenever possible, instead of meaningless literals.

Specs

Short name	Functions/UseConstantAsArguments
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Tikiwiki*, *shopware*

13.2.612 Used Functions

The functions below are used in the code.

A function is used in the code when it is called literally, or as a string callback.

```
<?php

function used() {}
// The 'unused' function is defined but never called
function unused() {}

// The 'used' function is called at least once
used();

// The 'used' function is called as a callback
array_filter($array, 'used');

?>
```

Specs

Short name	Functions/UsedFunctions
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.613 Useless Argument

> The argument is always used with the same value. This value could be hard coded in the method, and save one argument slot.

There is no indication that this argument will be used with other values. It may be a development artifact, that survived without cleaning.

```
<?php

// All foo2 arguments are used with different values
function foo2($a, $b) {}
foo2(1, 2);
foo2(2, 2);
foo2(3, 3);

// The second argument of foo is always used with 2
function foo($a, $b) {}
foo(1, 2);
foo(2, 2);
foo(3, 2);

?>
```

Methods with less than 3 calls are not considered here, to avoid reporting methods used once. Also, arguments with a default value are omitted.

The chances of useless arguments decrease with the number of usage. The parameter *maxUsageCount* prevents highly called methods (more than the parameter value) to be processed.

Suggestions

- Remove the argument and hard code its value inside the method
- Add the value as default in the method signature, and drop it from the calls
- Add calls to the method, with more varied arguments

Name	Default	Type	Description
maxUsageCount	30	integer	Maximum count of function usage. Use this to limit the amount of processed arguments.

Specs

Short name	Functions/UselessArgument
Rulesets	none
Exakt since	1.8.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.614 Useless Default Argument

One of the argument has a default value, and this default value is never used. Every time the method is called, the argument is provided explicitly, rendering the default value actually useless.

```
<?php

function goo($a, $b = 3) {
    // do something here
}

// foo is called 3 times, and sometimes, $b is not provided.
goo(1,2);
goo(1,2);
goo(1);


function foo($a, $b = 3) {
    // do something here
}

// foo is called 3 times, and $b is always provided.
foo(1,2);
foo(1,2);
foo(1,2);
?>
```

Suggestions

- Remove the default value
- Remove the explicit argument in the function call, when it is equal to the default value

Specs

Short name	Functions/UselessDefault
Rulesets	*Suggestions*
Exakt since	1.7.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.615 Useless Referenced Argument

The argument has a reference, but is only used for reading.

This is probably a development artefact that was forgotten. It is better to remove it.

This analysis also applies to foreach() loops, that declare the blind variable as reference, then use the variable as an object, accessing properties and methods. When a variable contains an object, there is no need to declare a reference : it is a reference automatically.

```
<?php

function foo($a, &$b, &$c) {
    // $c is passed by reference, but only read. The reference is useless.
    $b = $c + $a;
    // The reference is useful for $b
}

foreach ($array as &$element) {
    $element->method();
}

?>
```

See also Objects and references.

Suggestions

- Remove the useless & from the argument
- Make an actual use of the argument before the end of the method

Specs

Short name	Functions/UselessReferenceArgument
Rulesets	*Analyze*
Exakt since	1.1.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Woocommerce*, *Magento*

13.2.616 Useless Return

The spotted functions or methods have a return statement, but this statement is useless. This is the case for constructor and destructors, whose return value are ignored or inaccessible.

When return is void, and the last element in a function, it is also useless.

```
<?php

class foo {
```

(continues on next page)

(continued from previous page)

```
    function __construct() {
        // return is not used by PHP
        return 2;
    }
}

function bar(&$a) {
    $a++;
    // The last return, when empty, is useless
    return;
}

?>
```

Suggestions

- Remove the return expression. Keep any other calculation.

Specs

Short name	Functions/UselessReturn
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	Very high
Examples	*ThinkPHP*, *Vanilla*

13.2.617 Useless Type Check

With typehint, some checks on the arguments are now handled by the type system.

In particular, a type hinted argument can't be null, unless it is explicitly nullable, or has a `null` value as default.

```
<?php

// The test on null is useless, it will never happen
function foo(A $a) {
    if (is_null($a)) {
        // do something
    }
}

// Either nullable ? is too much, either the default null is
function barbar(?A $a = null) {
}

// The test on null is useful, the default value null allows it
function bar(A $a = null) {
    if ($a === null) {
```

(continues on next page)

(continued from previous page)

```
        // do something
    }
}


?>
```

See also Type Declarations.

Suggestions

- Remove the nullable typehint
- Remove the null default value
- Remove tests on null

Specs

Short name	Functions/UselessTypeCheck
Rulesets	*Dead code*
Exakt since	1.8.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.618 Uses Default Values

Default values are provided to methods so as to make it convenient to use. However, with new versions, those values may change. For example, in PHP 5.4, htmlentities() switched from `Latin1` to `UTF-8` default encoding.

```
<?php

$string = Eu não sou o pão;

echo htmlentities($string);

// PHP 5.3 : Eu n&Atilde;&pound;o sou o p&Atilde;&pound;o
// PHP 5.4 : Eu n&atilde;o sou o p&atilde;o

// Stable across versions
echo htmlentities($string, 'UTF8');

?>
```

As much as possible, it is recommended to use explicit values in those methods, so as to prevent from being surprise at a future PHP evolution.

This analyzer tend to report a lot of false positives, including usage of count(). Count() indeed has a second argument for recursive counts, and a default value. This may be ignored safely.

Suggestions

- Mention all arguments, as much as possible

Specs

Short name	Functions/UsesDefaultArguments
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.619 Using Deprecated Method

A call to a deprecated method has been spotted. A method is deprecated when it bears a `@deprecated` parameter in its typehint definition.

Deprecated methods which are not called are not reported.

```
<?php

// not deprecated method
not_deprecated();

// deprecated methods
deprecated();
$object = new X();
$object->deprecatedToo();

/**
 * @deprecated since version 2.0.0
 */
function deprecated() {}

// PHP 8.0 attribute for deprecation
class X {
    #[ Deprecated]
    function deprecatedToo() {}
}

function not_deprecated() {}

?>
```

See also @deprecated.

Suggestions

- Replace the deprecated call with a stable call
- Remove the deprecated attribute from the method definition

- Remove the deprecated call

Specs

Short name	Functions/UsingDeprecated
Rulesets	*Analyze*, *Attributes*
Exakt since	2.1.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.620 Has Variable Arguments

Indicates if this function or method accept an arbitrary number of arguments, based on func_get_args(), func_get_arg() and func_num_args() usage.

```
<?php

// Fixed number of arguments
function fixedNumberOfArguments($a, $b) {
    if (func_num_args() > 2) {
        $c = func_get_args();
        array_shift($c); // $a
        array_shift($c); // $b
    }
    // do something
}

// Fixed number of arguments
function fixedNumberOfArguments($a, $b, $c = 1) {}

?>
```

Specs

Short name	Functions/VariableArguments
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.621 Methods Without Return

List of all the function, closures, methods that have no explicit return.

Functions that hold the `void` return type are omitted.

```php
<?php

// With return null : Explicitly not returning
function withExplicitReturn($a = 1) {
    $a++;
    return null;
}

// Without indication
function withoutExplicitReturn($a = 1) {
    $a++;
}

// With return type void : Explicitly not returning
function withExplicitReturnType($a = 1) : void {
    $a++;
}

?>
```

See also return.

Suggestions

- Add the returntype 'void' to make this explicit behavior

Specs

Short name	Functions/WithoutReturn
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.622 Wrong Argument Type

Checks that the type of the argument is consistent with the type of the called method.

```php
<?php

function foo(int $a) { }

//valid call, with an integer
foo(1);

//invalid call, with a string
foo('asd');

?>
```

This analysis is valid with PHP 8.0.

Suggestions

- Always use a valid type when calling methods.

Specs

Short name	Functions/WrongArgumentType
Rulesets	*Analyze*, *Typechecks*
Exakt since	2.1.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.623 Wrong Function Name Case

The spotted functions are used with a different case than their definition. While PHP accepts this, it makes the code harder to read.

It may also be a violation of coding conventions.

```
<?php

// Definition of the class
function foo () {}

// Those calls have wrong case
FOO();
\Foo();

// This is valid
foo();

?>
```

See also PHP class name constant case sensitivity and PSR-11.

Suggestions

- Match the defined functioncall with the called name

Specs

Short name	Functions/WrongCase
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.624 Wrong Number Of Arguments

Those functioncalls are made with too many or too few arguments.

When the number arguments is wrong for native functions, PHP emits a warning. When the number arguments is too small for custom functions, PHP raises an exception. When the number arguments is too high for custom functions, PHP ignores the arguments. Such arguments should be handled with the variadic operator, or with func_get_args() family of functions.

```
<?php

echo strtoupper('This function is', 'ignoring arguments');
//Warning: strtoupper() expects exactly 1 parameter, 2 given in Command line code on␣
↪line 1

echo strtoupper();
//Warning: strtoupper() expects exactly 1 parameter, 0 given in Command line code on␣
↪line 1

function foo($argument) {}
echo foo();
//Fatal error: Uncaught ArgumentCountError: Too few arguments to function foo(), 0␣
↪passed in /Users/famille/Desktop/analyzeG3/test.php on line 10 and exactly 1␣
↪expected in /Users/famille/Desktop/analyzeG3/test.php:3

echo foo('This function is', 'ignoring arguments');

?>
```

It is recommended to check the signature of the methods, and fix the arguments.

Suggestions

- Add more arguments to fill the list of compulsory arguments
- Remove arguments to fit the list of compulsory arguments
- Use another method or class

Specs

Short name	Functions/WrongNumberOfArguments
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-missing-argument.md
Examples	*xataface*

13.2.625 Wrong Number Of Arguments In Methods

Those methods are called with a wrong number of arguments : too many or too few. Check the signature.

```
<?php

class Foo {
    private function Bar($a, $b) {
        return $a + $b;
    }

    public function foobar() {
        $this->Bar(1);

        // Good amount
        $this->Bar(1, 2);

        // Too Many
        $this->Bar(1, 2, 3);
    }
}

?>
```

Methods with a variable number of argument, either using ellipsis or func_get_args() are ignored.

PHP emits an error at runtime, when arguments are not enough : ''. PHP doesn't emit an error when too many arguments are provided.

Suggestions

- Adapt the call to use one of the right number of arguments : this means dropping the extra ones, or adding the missing ones
- Adapt the signature of the method, and use a default value

Specs

Short name	Functions/WrongNumberOfArgumentsMethods
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.626 Wrong Optional Parameter

Wrong placement of optional parameters.

PHP parameters are optional when they defined with a default value, like this :

```
<?php
    function x($arg = 1) {
        // PHP code here
    }
?>
```

When a function have both compulsory and optional parameters, the compulsory ones should appear first, and the optional should appear last :

```
<?php
    function x($arg, $arg2 = 2) {
        // PHP code here
    }
?>
```

PHP solves this problem at runtime, assign values in the same other, but will miss some of the default values and emits warnings.

It is better to put all the optional parameters at the end of the method's signature.

Optional parameter wrongly placed are now a Notice in PHP 8.0. The only previous case that is allowed in PHP 8.0 and also in this analysis, is when the `null` value is used as default for typed arguments.

See also Function arguments.

Suggestions

- Give default values to all but first parameters. Null is a good default value, as PHP will use it if not told otherwise.
- Remove default values to all but last parameters. That is probably a weak solution.
- Change the order of the values, so default-valued parameters are at the end. This will probably have impact on the rest of the code, as the API is changing.

Specs

Short name	Functions/WrongOptionalParameter
Rulesets	*Analyze*, *CE*, *CI-checks*, *CompatibilityPHP80*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*FuelCMS*, *Vanilla*

13.2.627 Wrong Type Returned

The returned value is not compatible with the specified return type.

```
<?php

// classic error
function bar() : int {
    return 'A';
}

// classic static error
const B = 2;
function bar() : string {
```

(continues on next page)

(continued from previous page)

```
    return B;
}

// undecideable error
function bar($c) : string {
    return $c;
}

// PHP lint this, but won't execute it
function foo() : void {
    // No return at all
}

?>
```

See also Returning values and Void Return Type.

Suggestions

- Match the return type with the return value
- Remove the return expression altogether
- Add a typecast to the returning expression

Specs

Short name	Functions/WrongReturnedType
Rulesets	*Analyze*, *CI-checks*, *ClassReview*
Exakt since	1.8.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.628 Wrong Typehinted Name

The parameter name doesn't reflect the typehint used.

There are no restriction on parameter names, except its uniqueness in the signature. Yet, using a scalar typehint as the name for another typehinted value is just misleading.

```
<?php

function foo(string $array,
             int $int) {
    // doSomething()
}

function bar(array $strings) {
    // doSomething()
}
```

(continues on next page)

(continued from previous page)

```
?>
```

This analysis relies on exact names : calling an array a list of `strings` is OK with this analysis.

This analysis relies on a few variations of names : `bool` and `boolean`, `int` and `integer`.

Suggestions

- Rename the parameter

Specs

Short name	Functions/WrongTypehintedName
Rulesets	*Semantics*
Exakt since	2.0.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.629 Wrong Type With Call

This analysis checks that a call to a method uses the right literal values' types.

Currently, this analysis doesn't take into account `strict_types = 1`.

```
<?php

function foo(string $a) {

}

// wrong type used
foo(1);

// wrong type used
foo("1");

?>
```

Suggestions

- Use the right type with all literals

Specs

Short name	Functions/WrongTypeWithCall
Rulesets	*Analyze*, *CI-checks*, *Typechecks*
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.630 Already Parents Interface

The same interface is implemented by a class and one of its children.

That way, the child doesn't need to implement the interface, nor define its methods to be an instance of the interface.

```
<?php

interface i {
    function i();
}

class A implements i {
    function i() {
        return __METHOD__;
    }
}

// This implements is useless.
class AB extends A implements i {
    // No definition for function i()
}

// Implements i is understated
class AB extends A {
    // redefinition of the i method
    function i() {
        return __METHOD__.' ';
    }
}

$x = new AB;
var_dump($x instanceof i);
// true

$x = new AC;
var_dump($x instanceof i);
// true

?>
```

Suggestions

- Keep the implements call in the class that do implements the methods. Remove it from the children classes.

Specs

Short name	Interfaces/AlreadyParentsInterface
Rulesets	*Analyze*, *Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	Very high
Examples	*WordPress*, *Thelia*

13.2.631 Avoid Self In Interface

Self and Parent are tricky when used in an interface.

`self` refers to the current interface or its extended parents : as long as the constant is defined in the interface family, this is valid. On the other hand, when `self` refers to the current class, the resolution of names will happen at execution time, leading to confusing results.

`parent` has the same behavior than `self`, except that it doesn't accept to be used inside an interface, as it will yield an error. This is one of those error that lint but won't execute in certain conditions.

`Static` can't be used in an interface, as it needs to be resolved at call time anyway.

```
<?php

interface i extends ii {
    // This 'self' is valid : it refers to the interface i
    public const I = self::I2 + 2;

    // This 'self' is also valid, as it refers to interface ii, which is a part of␣
↪interface i
    public const I2 = self::IP + 4;

    // This makes interface i dependant on the host class
    public const I3 = parent::A;
}

?>
```

See also Scope Resolution Operator (::).

Suggestions

- Use a fully qualified namespace instead of self
- Use a locally defined constant, so self is a valid reference

Specs

Short name	Interfaces/AvoidSelfInInterface
Rulesets	*ClassReview*
Exakt since	1.5.4
Php Version	All
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	High

13.2.632 Cant Implement Traversable

It is not possible to implement the `Traversable``interface. The alternative is to implement ``Iterator` or `IteratorAggregate`.

`Traversable` may be useful when used with `instanceof`.

```
<?php

// This lints, but doesn't run
class x implements Traversable {

}

if( $argument instanceof Traversable ) {
    // doSomething
}

?>
```

See also Traversable, Iterator and IteratorAggregate..

Suggestions

- Implement Iterator or IteratorAggregate

Specs

Short name	Interfaces/CantImplementTraversable
Rulesets	*Analyze*, *CI-checks*, *LintButWontExec*
Exakt since	1.9.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.633 Concrete Visibility

Methods that implements an interface in a class must be public.

PHP does lint this, unless the interface and the class are in the same file. At execution, it stops immediately with a Fatal error : 'Access level to c::iPrivate() must be public (as in class i) ';

```
<?php

interface i {
    function iPrivate() ;
    function iProtected() ;
    function iPublic() ;
}

class c implements i {
    // Methods that implements an interface in a class must be public.
    private function iPrivate() {}
    protected function iProtected() {}
    public function iPublic() {}
}

?>
```

See also Interfaces.

Suggestions

- Always set interface methods to public.

Specs

Short name	Interfaces/ConcreteVisibility
Rulesets	*Analyze*, *LintButWontExec*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.634 Forgotten Interface

The following classes have been found implementing an interface's methods, though it doesn't explicitly implements this interface. This may have been forgotten.

```
<?php

interface i {
    function i();
}

// i is not implemented and declared
class foo {
    function i() {}
    function j() {}
}
```

(continues on next page)

(continued from previous page)

```
// i is implemented and declared
class foo implements i {
    function i() {}
    function j() {}
}

?>
```

See also *Could Use Trait*.

Suggestions

- Mention interfaces explicitly whenever possible

Specs

Short name	Interfaces/CouldUseInterface
Rulesets	*Analyze*
Exakt since	0.11.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.635 Empty Interfaces

Empty interfaces are a code smell. Interfaces should contains at least a method or a constant, and not be totally empty.

```
<?php

// an empty interface
interface empty {}

// an normal interface
interface normal {
    public function i() ;
}

// a constants interface
interface constantsOnly {
    const FOO = 1;
}

?>
```

See also Empty interfaces are bad practice and Blog : Are empty interfaces code smell?.

Suggestions

- Remove the interface
- Add some methods or constants to the interface

Specs

Short name	Interfaces/EmptyInterface
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.636 Interface Methods

List the names of the methods in an interface.

```
<?php

interface i {
    // This is an interface method name
    function foo() ;
}

?>
```

Specs

Short name	Interfaces/InterfaceMethod
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.637 Interfaces Glossary

List of all the defined interfaces in the code.

```
<?php

// interfaceName is reported
interface interfaceName {
    function interfaceMethod() ;
}
?>
```

Specs

Short name	Interfaces/Interfacenames
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.638 Interfaces Usage

List of used interfaces.

Interfaces are used when mentioned in a class or another interface, with implements keyword; they are used in instanceof expression, in typehints and class constant.

```
<?php

// interface definition
interface i {
    const I = 2;
}

// interface extension
interface i2 extends i {}

// interface implementation
class foo implements i {}

$foo = new foo();

var_dump($foo instanceof i);

function bar( i $arg) { }
bar($foo);

// in class constant
echo i::I;

?>
```

Specs

Short name	Interfaces/InterfaceUsage
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.639 Is An Extension Interface

This is an interface defined in a PHP C extension.

```
<?php

// MyInterface is not recognized as an extension interface
function foo ( MyInterface $a) {
    // \ArrayAccess is recognized as a native PHP extension
    if ($a instanceof \ArrayAccess) {
        // doSomething()
    }
}

?>
```

Specs

Short name	Interfaces/IsExtInterface
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.640 Interfaces Is Not Implemented

Classes that implements interfaces, must implements each of the interface's methods.

```
<?php

class x implements i {
    // This method implements the foo method from the i interface
    function foo() {}

    // The method bar is missing, yet is requested by interface i
    function foo() {}
}

interface i {
    function foo();
    function bar();
}

?>
```

This problem tends to occur in code that splits interfaces and classes by file. This means that PHP's linting will skip the definitions and not find the problem. At execution time, the definitions will be checked, and a Fatal error will occur.

This situation usually detects code that was forgotten during a refactorisation of the interface or the class and its sibblings.

See also Interfaces.

Suggestions

- Implements all the methods from the interfaces
- Remove the class
- Make the class abstract
- Make the missing methods abstract

Specs

Short name	Interfaces/IsNotImplemented
Rulesets	*Analyze*, *CI-checks*, *ClassReview*, *LintButWontExec*
Exakt since	1.9.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.641 Interfaces Don't Ensure Properties

When using an interface as a typehint, properties are not enforced, nor available.

An interface is a template for a class, which specify the minimum amount of methods and constants. Properties are never defined in an interface, and should not be relied upon.

```
<?php

interface i {
    function m () ;
}

class x implements i {
    public $p = 1;

    function m() {
        return $this->p;
    }
}

function foo(i $i, x $x) {
    // this is invalid, as $p is not defined in i, so it may be not available
    echo $i->p;

    // this is valid, as $p is defined in $x
    echo $x->p;
}

?>
```

Suggestions

- Use classes for typehint when properties are accessed
- Only use methods and constants which are available in the interface

Specs

Short name	Interfaces/NoGaranteeForPropertyConstant
Rulesets	*Analyze*, *ClassReview*
Exakt since	1.9.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.642 PHP Interfaces

List of PHP interfaces being used in the code.

```
<?php

// Countable is a PHP native interface
class Enumeration extends Countable {
    function count() { return 1; }
}

?>
```

Specs

Short name	Interfaces/Php
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.643 Possible Interfaces

This analyzer lists classese that may be a base to create interfaces.

Currently, only classes with more than 1 method are used, and interfaces are considered when at least 2 methods are common.

Signature and method options are not taken into account.

```
<?php

class a {
    function m1 () {}
    function m2 () {}
    function m3 () {}
}

class b {
    function m1 () {}
    function m2 () {}
    function m4 () {}
}

// This class has not enough shared methods with other classes
class c {
    function m1 () {}
    function m4 () {}
    function m5 () {}
}

?>
```

Suggestions

- Add those interfaces, and use the *implements* keyword in the mentionned classes.

Specs

Short name	Interfaces/PossibleInterfaces
Rulesets	none
Exakt since	2.0.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.644 Repeated Interface

A class should implements only once an interface. An interface can only extends once another interface. In both cases, parent classes or interfaces must be checked.

PHP accepts multiple times the same interface in the `implements` clause. In fact, it doesn't do anything beyond the first implement.

```
<?php

use i as j;

interface i {}

// Multiple ways to reference an interface
```

(continues on next page)

(continued from previous page)

```
class foo implements i, \i, j {}

// This applies to interfaces too
interface bar extends i, \i, j {}

?>
```

This code may compile, but won't execute.

See also Object Interfaces and The Basics.

Suggestions

- Remove the interface usage at the lowest class or interface

Specs

Short name	Interfaces/RepeatedInterface
Rulesets	*Analyze*, *LintButWontExec*
Exakt since	1.4.9
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.645 Undefined Interfaces

Some typehints or `instanceof` that are relying on undefined interfaces or classes. They will always return false. Any condition based upon them are dead code.

```
<?php

class var implements undefinedInterface {
    // If undefinedInterface is undefined, this code lints but doesn't run
}

if ($o instanceof undefinedInterface) {
    // This is silent dead code
}

function foo(undefinedInterface $a) {
    // This is dead code
    // it will probably be discovered at execution
}

?>
```

See also Object interfaces, Type declarations, and Instanceof.

Suggestions

- Implement the missing interfaces
- Remove the code governed by the missing interface : the whole method if it is an typehint, the whole if/then if it is a condition.

Specs

Short name	Interfaces/UndefinedInterfaces
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*xataface*

13.2.646 Unused Interfaces

Those interfaces are defined and never used. They should be removed, as they are dead code.

Interfaces may be use as parent for other interfaces, as typehint (argument, return and property), in instance of.

```
<?php

interface used {}
interface unused {}

// Used by implementation
class c implements used {}

// Used by extension
interface j implements used {}

$x = new c;

// Used in a instanceof
var_dump($x instanceof used);

// Used in a typehint
function foo(Used $x) {}

?>
```

Suggestions

- Remove the interface
- Actually use the interface

Specs

Short name	Interfaces/UnusedInterfaces
Rulesets	*Dead code*, *Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Tine20*

13.2.647 Used Interfaces

Interfaces used in the code.

```
<?php

interface used {}

// Used by implementation
class c implements used {}

// Used by extension
interface j implements used {}

$x = new c;

// Used in a instanceof
var_dump($x instanceof used);

// Used in a typehint
function foo(Used $x) {}

?>
```

Specs

Short name	Interfaces/UsedInterfaces
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.648 Useless Interfaces

The interfaces below are defined and are implemented by some classes.

However, they are never used to enforce an object's class in the code, using instanceof or in a typehint. As they are currently used, those interfaces may be removed without change in behavior.

```
<?php
    // only defined interface but never enforced
    interface i {};
    class c implements i {}
?>
```

Interfaces should be used in Typehint or with the instanceof operator.

```
<?php
    interface i {};

    function foo(i $arg) {
        // Now, $arg is always an 'i'
    }

    function bar($arg) {
        if (!($arg instanceof i)) {
            // Now, $arg is always an 'i'
        }
    }
?>
```

Suggestions

- Use the interface with instanceof, or a typehint
- Drop the interface altogether : both definition and implements keyword

Specs

Short name	Interfaces/UselessInterfaces
Rulesets	*Analyze*, *ClassReview*, *Typechecks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-useless-interfaces
Examples	*Woocommerce*

13.2.649 Modules/IncomingData

Suggestions

-

Specs

Short name	Modules/IncomingData
Rulesets	none
Exakt since	1.8.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.650 Modules/NativeReplacement

Suggestions

-

Specs

Short name	Modules/NativeReplacement
Rulesets	none
Exakt since	1.8.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.651 Aliases

List of all aliases used, to alias namespaces.

```
<?php

// This is an alias
use stdClass as aClass;

// This is not an alias : it is not explicit
use stdClass;

trait t {
    // This is not an alias, it's a trait usage
    use otherTrait;
}

?>
```

See also Using namespaces: Aliasing/Importing.

Specs

Short name	Namespaces/Alias
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.652 Possible Alias Confusion

An alias is used for a class that doesn't belong to the current namespace, while there is such a class. This also applies to traits and interfaces.

When no alias is used, PHP will search for a class in the local space. Since classes, traits and interfaces are usually stored one per file, it is a valid syntax to create an alias, even if this alias name is the name of a class in the same namespace.

Yet, with an alias refering to a remote class, while a local one is available, it is possible to generate confusion.

```
<?php

// This should be in a separate file, but has been merged here, for display purposes.
namespace A {
    //an alias from a namespace called C
    use C\A as C_A;

    //an alias from a namespace called C, which will superseed the local A\B class␣
↪(see below)
    use C\D as B;
}

namespace A {
    // There is a class B in the A namespace
    class B {}
}

?>
```

Suggestions

- Avoid using existing classes names for alias
- Use a coding convention to distinguish alias from names

Specs

Short name	Namespaces/AliasConfusion
Rulesets	*Suggestions*
Exakt since	2.1.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.653 Fully Qualified Constants

Constants defined with their namespace.

When defining constants with define() function, it is possible to include the actual namespace :

```
<?php

define('a\b\c', 1);

?>
```

However, the name should be fully qualified without the initial . Here, abc constant will never be accessible as a namespace constant, though it will be accessible via the constant() function.

Also, the namespace will be absolute, and not a relative namespace of the current one.

Suggestions

- Drop the initial when creating constants with define() : for example, use trim($x, ‘’), which removes anti-slashes before and after.

Specs

Short name	Namespaces/ConstantFullyQualified
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.654 Could Use Alias

This long name may be reduced by using an available alias.

This applies to classes (as full name or prefix), and to constants and functions.

```
<?php

use a\b\c;
use function a\b\c\foo;
use const a\b\c\D;

// This may be reduced with the above alias to c\d()
new a\b\c\d();

// This may be reduced to c\d\e\f
new a\b\c\d\e\f();

// This may be reduced to c()
new a\b\c();

// This may be reduced to D
echo a\b\c\D;

// This may be reduced to D
a\b\c\foo();

// This can't be reduced : it is an absolute name
\a\b\c\foo();

// This can't be reduced : it is no an alias nor a prefix
a\b\d\foo();

?>
```

Suggestions

- Use all your aliases so as to make the code shorter and more readable
- Add new aliases for missing path
- Make class names absolute and drop the aliases

Specs

Short name	Namespaces/CouldUseAlias
Rulesets	*Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.655 Empty Namespace

Declaring a namespace in the code and not using it for structure declarations or global instructions is useless.

Using simple style :

```
<?php

namespace Y;

class foo {}


namespace X;
// This is useless

?>
```

Using bracket-style syntax :

```
<?php

namespace X {
    // This is useless
}

namespace Y {

    class foo {}

}

?>
```

Suggestions

- Remove the namespace

Specs

Short name	Namespaces/EmptyNamespace
Rulesets	*Analyze*, *CI-checks*, *Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-empty-namespace

13.2.656 Global Import

Mark a Use statement that is importing a global class in the current file.

```
<?php

namespace Foo {
    // This is a global import
```

(continues on next page)

(continued from previous page)

```
    use Stdclass;
}
?>
```

Specs

Short name	Namespaces/GlobalImport
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.657 Hidden Use Expression

The use expression for namespaces should always be at the beginning of the namespace block.

It is where everyone expect them, and it is less confusing than having them at various levels.

```
<?php

// This is visible
use A;

class B {}

// This is hidden
use C as D;

class E extends D {
    use traitT; // This is a use for a trait

    function foo() {
        // This is a use for a closure
        return function ($a) use ($b) {}
    }
}

?>
```

Suggestions

- Group all uses together, at the beginning of the namespace or class

Specs

Short name	Namespaces/HiddenUse
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Tikiwiki*, *OpenEMR*

13.2.658 Multiple Alias Definitions Per File

Avoid aliasing the same name with different aliases. This leads to confusion.

```
<?php

// first occurrence
use name\space\ClasseName;

// when this happens, several other uses are mentionned

// name\space\ClasseName has now two names
use name\space\ClasseName as anotherName;

?>
```

See also Namespaces/MultipleAliasDefinition.

Specs

Short name	Namespaces/MultipleAliasDefinitionPerFile
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.10.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.659 Multiple Alias Definitions

Some aliases are representing different classes across the repository. This leads to potential confusion.

Across an application, it is recommended to use the same namespace for one alias. Failing to do this lead to the same keyword to represent different values in different files, with different behavior. Those are hard to find bugs.

```
<?php

namespace A {
    use d\d; // aka D
```

(continues on next page)

(continued from previous page)

```
}

// Those are usually in different files, rather than just different namespaces.

namespace B {
    use b\c as D; // also D. This could be named something else
}

?>
```

Suggestions

- Give more specific names to classes
- Use an alias 'use AB ac BC' to give locally another name

Specs

Short name	Namespaces/MultipleAliasDefinitions
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*ChurchCRM*, *Phinx*

13.2.660 Namespaces Glossary

List of all the defined namespaces in the code, using the namespace keyword.

```
<?php

// One reported namespace
namespace one\name\space {}

// This global namespace is reported, as it is explicit
namespace { }

?>
```

Global namespaces are mentioned when they are explicitly used.

Specs

Short name	Namespaces/Namespacesnames
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.661 Namespaces

Inventory of all namespaces.

Specs

Short name	Namespaces/NamespaceUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.662 Should Make Alias

Long names should be aliased.

Aliased names are easy to read at the beginning of the script; they may be changed at one point, and update the whole code at the same time. Finally, short names makes the rest of the code readable.

```
<?php

namespace x\y\z;

use a\b\c\d\e\f\g as Object;

// long name, difficult to read, prone to change.
new a\b\c\d\e\f\g();

// long name, difficult to read, prone to silent dead code if namespace change.
if ($o instanceof a\b\c\d\e\f\g) {

}

// short names Easy to update all at once.
new Object();
if ($o instanceof Object) {

}
```

(continues on next page)

(continued from previous page)

```
?>
```

Specs

Short name	Namespaces/ShouldMakeAlias
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.663 Unresolved Use

The following use instructions cannot be resolved to a known class, interface, trait, constant or function. They should be dropped or fixed.

A known class, interface, trait, constant or function is defined in PHP (standard), an extension, a stub or the current code.

```
<?php

namespace A {
    // class B is defined
    class B {}
    // class C is not defined
}

namespace X/Y {

    use A/B;  // This use is valid
    use A/C;  // This use point to nothing.

    new B();
    new C();
}

?>
```

Use expression are options for the current namespace.

See also Using namespaces: Aliasing/Importing.

Suggestions

- Remove the use expression
- Fix the use expression

Specs

Short name	Namespaces/UnresolvedUse
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
ClearPHP	no-unresolved-use

13.2.664 Unused Use

Unused use statements. They may be removed, as they clutter the code and slows PHP by forcing it to search in this list for nothing.

```
<?php

use A as B; // Used in a new call.
use Unused; // Never used. May be removed

$a = new B();

?>
```

Suggestions

- Remove the unused use

Specs

Short name	Namespaces/UnusedUse
Rulesets	*Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	Very high
ClearPHP	no-useless-use

13.2.665 Used Use

List of use statements. Those use are made to import namespaces structures, not to include traits.

```
<?php

namespace A {
    class b {}
```

(continues on next page)

(continued from previous page)

```
}

namespace B {
    use A\B as B;

    new B();
}

?>
```

See also Using namespaces: Aliasing/Importing.

Specs

Short name	Namespaces/UsedUse
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.666 Use Const And Functions

Since PHP 5.6 it is possible to import specific functions or constants from other namespaces.

```
<?php

namespace A {
    const X = 1;
    function foo() { echo __FUNCTION__; }
}

namespace My{
    use function A\foo;
    use constant A\X;

    echo foo(X);
}

?>
```

See also Using namespaces: Aliasing/Importing.

Specs

Short name	Namespaces/UseFunctionsConstants
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Exakt since	0.8.4
Php Version	5.6+
Severity	Major
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.667 Use With Fully Qualified Name

Use statement doesn't require a fully qualified name.

PHP manual recommends not to use fully qualified name (starting with) when using the 'use' statement : they are "the leading backslash is unnecessary and not recommended, as import names must be fully qualified, and are not processed relative to the current namespace".

```
<?php

// Recommended way to write a use statement.
use  A\B\C\D as E;

// No need to use the initial \
use \A\B\C\D as F;

?>
```

Suggestions

- Remove the initial in use expressions.

Specs

Short name	Namespaces/UseWithFullyQualifiedNS
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.668 Wrong Case Namespaces

Namespaces are case-insentives.

```
<?php

// Namespaces should share the same case
```

(continues on next page)

(continued from previous page)

```
namespace X {}

namespace x {}

?>
```

Suggestions

- Synchronize all names

Specs

Short name	Namespaces/WrongCase
Rulesets	none
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.669 Abstract Away

Avoid using PHP native functions that produce data direcly in the code. For example, date() or random_int(). They should be abstracted away in a method, that will be replaced later for testing purposes, or even debugging.

To abstract such calls, place them in a method, and add an interface to this method. Then, create and use those objects.

```
<?php

// abstracted away date
$today = new MyDate();
echo 'Date : '.$today->date('r');

// hard coded date of today : it changes all the time.
echo 'Date : '.date('r');

interface MyCalendar{
    function date($format) : string ;
}

class MyDate implements MyCalendar {
    function date($format) : string { return date('r'); }
}

// Valid implementation, reserved for testing purpose
// This prevents from waiting 4 years for a test.
class MyDateForTest implements MyCalendar {
    function date($format) : string { return date('r', strtotime('2016-02-29 12:00:00
↪')); }
}
```

(continues on next page)

(continued from previous page)

```
?>
```

This analysis targets two API for abstraction : time and random values. Time and date related functions may be replaced by Carbon, Clock, Chronos. Random values may be replaced with RandomLib or a custome interface.

See also Being in control of time in PHP and How to test non-deterministic code.

Suggestions

- Abstract away the calls to native PHP functions, and upgrade the unit tests

Name	Default	Type	Description
abstractableCalls		ini_hash	Functions that shouldn't be called directly, unless in a method.
abstractableClasses		ini_hash	Classes that shouldn't be instantiated directly, unless in a method.

Specs

Short name	Patterns/AbstractAway
Rulesets	*Suggestions*
Exakt since	2.1.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.670 Courier Anti-Pattern

> The courier anti-pattern is the storage of a dependency by a class, in order to create an instance that requires this dependency.

The class itself doesn't actually need this dependency, but has a dependency to a class that requires it.

```
<?php

// The foo class requires bar
class Foo {
    public function __construct(Bar $b) {
    }
}

// Class A doesn't depends on Bar, but depends on Foo
// Class A never uses Bar, but only uses Foo.
class A {
    private $courier;

    public function __construct(Bar $courier) {
        $this->courier = $courier;
    }

    public function Afoo() {
```

(continues on next page)

(continued from previous page)

```
        $b = new Foo($this->courier);
    }

}

?>
```

The alternative here is to inject Foo instead of Bar.

See also Courier Anti-pattern.

Specs

Short name	Patterns/CourrierAntiPattern
Rulesets	*CE*
Exakt since	0.11.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.671 Dependency Injection

A dependency injection is a typehinted argument, that is stored in a property by the constructor.

```
<?php

// Classic dependency injection
class foo {
    private $bar;

    public function __construct(Bar $bar) {
        $this->bar = $bar;
    }

    public function doSomething($args) {
        return $this->bar->barbar($args);
    }
}

// Without typehint, this is not a dependency injection
class foo {
    private $bar;

    public function __construct($bar) {
        $this->bar = $bar;
    }
}

?>
```

See also Understanding Dependency Injection.

Specs

Short name	Patterns/DependencyInjection
Rulesets	*CE*
Exakt since	0.11.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.672 An OOP Factory

A method or function that implements a factory. A factory is a class that handles the creation of an object, based on parameters. The factory hides the logic that leads to the creation of the object.

```
<?php
    class AutomobileFactory {
        public static function create($make, $model) {
            $className = \\Automaker\\Brand\\$make;
            return new $className($model);
        }
    }

    // The factory is able to build any car, based on their
    $fuego = AutomobileFactory::create('Renault', 'Fuego');

    print_r($fuego->getMakeAndModel()); // outputs Renault Fuego
?>
```

See also Factory (object-oriented programming) and Factory.

Suggestions

-

Specs

Short name	Patterns/Factory
Rulesets	*CE*
Exakt since	1.6.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.673 array_key_exists() Speedup

isset() used to be the fastest, but array_key_exists() is. Since PHP 7.4, array_key_exists() has its own opcode, leading to better features and speed.

isset() is faster for all non-empty values, but is limited when the value is NULL or empty : then, array_key_exists() has the good features.

This change makes `array_key_exists() <https://www.php.net/array_key_exists>`_ actually faster than `isset() <https://www.www.php.net/isset>`_ by ~25% (tested with GCC 8, -O3, march=native, mtune=native)..

```
<?php

$foo = [123 => 456];

// This is sufficient and efficient since PHP 7.4
if (array_search_key($foo[123])) {
    // do something
}

// taking advantages of performances for PHP 7.4 and older
if (isset($foo[123]) || array_search_key($foo[123])) {
    // do something
}

?>
```

See also Implement ZEND_ARRAY_KEY_EXISTS opcode to speed up 'array_key_exists() <https://github.com/php/php-src/pull/3360>'_.

Suggestions

- Remove the logical test and the isset() call

Specs

Short name	Performances/ArrayKeyExistsSpeedup
Rulesets	*Performances*, *Suggestions*
Exakt since	1.6.1
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.674 No array_merge() In Loops

array_merge() is memory intensive : every call will duplicate the arguments in memory, before merging them.

To handle arrays that may be quite big, it is recommended to avoid using array_merge() in a loop. Instead, one should use array_merge() with as many arguments as possible, making the merge a on time call.

```
<?php

// A large multidimensional array
$source = ['a' => ['a', 'b', /*...*/],
           'b' => ['b', 'c', 'd', /*...*/],
```

(continues on next page)

(continued from previous page)

```
            /*...*/
            ];

// Faster way
$b = array();
foreach($source as $key => $values) {
    //Collect in an array
    $b[] = $values;
}

// One call to array_merge
$b = call_user_func_array('array_merge', $b);
// or with variadic
$b = call_user_func('array_merge', ..$b);

// Fastest way (with above example, without checking nor data pulling)
$b = call_user_func_array('array_merge', array_values($source))
// or
$b = call_user_func('array_merge', ...array_values($source))

// Slow way to merge it all
$b = array();
foreach($source as $key => $values) {
    $b = array_merge($b, $values);
}

?>
```

Note that array_merge_recursive() and file_put_contents() are affected and reported the same way.

Suggestions

- Store all intermediate arrays in a temporary variable, and use array_merge() once, with ellipsis or call_user_func_array().

Specs

Short name	Performances/ArrayMergeInLoops
Rulesets	*Analyze*, *CI-checks*, *Performances*, *Top10*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-array_merge-in-loop
Examples	*Tine20*

13.2.675 Autoappend

Appending a variable to itself leads to enormous usage of memory.

```
<?php

// Always append a value to a distinct variable
foreach($a as $b) {
    $c[] = $b;
}

// This copies the array to itself, and double the size each loop
foreach($a as $b) {
    $c[] = $c;
}
?>
```

Suggestions

- Change the variable in the append, on the left
- Change the variable in the append, on the right

Specs

Short name	Performances/Autoappend
Rulesets	*Performances*
Exakt since	1.8.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.676 Avoid array_push()

array_push() is slower than the [] operator.

This is also true if the [] operator is called several times, while array_push() may be called only once. And using count after the push is also faster than collecting array_push() return value.

```
<?php

$a = [1,2,3];
// Fast version
$a[] = 4;

$a[] = 5;
$a[] = 6;
$a[] = 7;
$count = count($a);

// Slow version
array_push($a, 4);
$count = array_push($a, 5,6,7);

// Multiple version :
```

(continues on next page)

(continued from previous page)

```
$a[] = 1;
$a[] = 2;
$a[] = 3;
array_push($a, 1, 2, 3);


?>
```

This is a micro-optimisation.

Suggestions

- Use the [] operator

Specs

Short name	Performances/AvoidArrayPush
Rulesets	*Performances*
Exakt since	0.9.1
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.677 Cache Variable Outside Loop

Avoid recalculating constant values inside the loop.

Do the calculation once, outside the loops, and then reuse the value each time.

One of the classic example if doing `count($array)` in a `for` loop : since the source is constant during the loop, the result of count() is always the same.

```
<?php

$path = '/some/path';
$fullpath = realpath("$path/more/dirs/");
foreach($files as $file) {
    // Only moving parts are used in the loop
    copy($file, $fullpath.$file);
}

$path = '/some/path';
foreach($files as $file) {
    // $fullpath is calculated each loop
    $fullpath = realpath("$path/more/dirs/");
    copy($file, $fullpath.$file);
}

?>
```

Depending on the load of the called method, this may increase the speed of the loop from little to enormously.

Suggestions

- Avoid using blind variables outside loops.
- Store blind variables in local variables or properties for later reuse.

Specs

Short name	Performances/CacheVariableOutsideLoop
Rulesets	*Performances*
Exakt since	1.2.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.678 fputcsv() In Loops

fputcsv() is slow when called on each row. It actually flushes the data to the disk each time, and that results in a inefficient dump to the disk, each call.

To speed up this process, it is recommended to dump the csv to memory first, then dump the memory to the disk, in larger chunks. Since fputcsv() works only on stream, it is necessary to use a memory stream.

```
<?php

// Speedy yet memory intensive version
$f = fopen('php://memory', 'w+');
foreach($data_source as $row) {
    // You may configure fputcsv as usual
    fputcsv($f, $row);
}
rewind($f); // Important
$fp = fopen('final.csv', 'w+');
fputs($fp, stream_get_contents($f));
fclose($fp);
fclose($f);

// Slower version
$fp = fopen('final.csv', 'w+');
foreach($data_source as $row) {
    // You may configure fputcsv as usual
    fputcsv($fp, $row);
}
fclose($fp);
?>
```

The speed improvement is significant on small rows, while it may be less significant on larger rows : with more data in the rows, the file buffer may fill up more efficiently. On small rows, the speed gain is up to 7 times.

Suggestions

- Use fputcsv() on a memory stream, and flush it on the disk once

Specs

Short name	Performances/CsvInLoops
Rulesets	*Performances*, *Top10*
Exakt since	1.5.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.679 Do In Base

Use SQL expression to compute aggregates.

```
<?php

// Efficient way
$res = $db->query('SELECT sum(e) AS sumE FROM table WHERE condition');

// The sum is already done
$row = $res->fetchArray();
$c += $row['sumE'];

// Slow way
$res = $db->query('SELECT e FROM table WHERE condition');

// This aggregates the column e in a slow way
while($row = $res->fetchArray()) {
    $c += $row['e'];
}

?>
```

Suggestions

- Rework the query to move the calculations in the database

Specs

Short name	Performances/DoInBase
Rulesets	*Performances*
Exakt since	1.2.8
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.680 Double array_flip()

Avoid double array_flip() to gain speed. While array_flip() alone is usually useful, a double call to array_flip() is made to make values and keys unique.

```
<?php

// without array_flip
function foo($array, $value) {
    $key = array_search($array, $value);

    if ($key !== false) {
        unset($array[$key]);
    }

    return $array;
}

// double array_flip
// array_flip() usage means that $array's values are all unique
function foo($array, $value) {
    $flipped = array_flip($value);
    unset($flipped[$value]);
    return array_flip($flipped);
}

?>
```

Suggestions

- use array_unique() or array_count_values
- use array_flip() once, and let PHP garbage collect it later
- Keep the original values in a separate variable

Specs

Short name	Performances/DoubleArrayFlip
Rulesets	*Performances*
Exakt since	1.1.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*NextCloud*

13.2.681 Fetch One Row Format

When reading results with ext/Sqlite3, it is recommended to explicitly request SQLITE3_NUM or SQLITE3_ASSOC, while avoiding the default value and SQLITE3_BOTH.

```php
<?php

$res = $database->query($query);

// Fastest version, but less readable
$row = $res->fetchArray(\SQLITE3_NUM);
// Almost the fastest version, and more readable
$row = $res->fetchArray(\SQLITE3_ASSOC);

// Default version. Quite slow
$row = $res->fetchArray();

// Worse case
$row = $res->fetchArray(\SQLITE3_BOTH);

?>
```

This is a micro-optimisation. The difference may be visible with 200k rows fetches, and measurable with 10k.

Suggestions

- Specify the result format when reading rows from a Sqlite3 database

Specs

Short name	Performances/FetchOneRowFormat
Rulesets	*Performances*
Exakt since	0.9.6
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.682 Isset() On The Whole Array

Isset() works quietly on a whole array. There is no need to test all previous index before testing for the target index.

```php
<?php

// Straight to the point
if (isset($a[1]['source'])) {
    // Do something with $a[1]['source']
}

// Doing too much work
if (isset($a) && isset($a[1]) && isset($a[1]['source'])) {
    // Do something with $a[1]['source']
}

?>
```

There is a gain in readability, by avoiding long and hard to read logical expression, and reducing them in one simple isset call.

There is a gain in performances by using one call to isset, instead of several, but it is a micro-optimization.

See also Isset <http://www.php.net/‘isset>‘_.

Suggestions

- Remove all unnecessary calls to isset()

Specs

Short name	Performances/IssetWholeArray
Rulesets	*Performances*, *Suggestions*
Exakt since	1.5.6
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Tine20*, *ExpressionEngine*

13.2.683 Joining file()

Use file() to read lines separately.

Applying join(‘’,) or implode(‘’,) to the result of file() provides the same results than using file_get_contents(), but at a higher cost of memory and processing.

If the delimiter is not ‘’, then implode() and file() are a better solution than file_get_contents() and str_replace() or nl2br().

```
<?php

// memory intensive
$content = file_get_contents('path/to/file.txt');

// memory and CPU intensive
$content = join('', file('path/to/file.txt'));

// Consider reading the data line by line and processing it along the way,
// to save memory
$fp = fopen('path/to/file.txt', 'r');
while($line = fget($fp)) {
    // process a line
}
fclose($fp);

?>
```

Always use file_get_contents() to get the content of a file as a string. Consider using readfile() to echo the content directly to the output.

See also file_get_contents and file.

Suggestions

- Use file_get_contents() instead of implode(file()) to read the whole file at once.
- Use readfile() to echo the content to stdout at once.
- Use fopen() to read the lines one by one, generator style.

Specs

Short name	Performances/JoinFile
Rulesets	*Performances*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*WordPress*, *SPIP*

13.2.684 Logical To in_array

Multiples exclusive comparisons may be replaced by in_array().

in_array() makes the alternatives more readable, especially when the number of alternatives is large. In fact, the list of alternative may even be set in a variable, and centralized for easier management.

Even two 'or' comparisons are slower than using a in_array() call. More calls are even slower than just two. This is a micro-optimisation : speed gain is low, and marginal. Code centralisation is a more significant advantage.

```
<?php

// Set the list of alternative in a variable, property or constant.
$valid_values = array(1, 2, 3, 4);
if (in_array($a, $valid_values) ) {
    // doSomething()
}

if ($a == 1 || $a == 2 || $a == 3 || $a == 4) {
    // doSomething()
}

// in_array also works with strict comparisons
if (in_array($a, $valid_values, true) ) {
    // doSomething()
}

if ($a === 1 || $a === 2 || $a === 3 || $a === 4) {
    // doSomething()
}

?>
```

See also in_array().

Suggestions

- Replace the list of comparisons with a in_array() call on an array filled with the various values
- Replace the list of comparisons with a isset() call on a hash whose keys are the various values

Specs

Short name	Performances/LogicalToInArray
Rulesets	*Analyze*
Exakt since	0.12.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high
Examples	*Zencart*

13.2.685 Make One Call With Array

Avoid calling the same function several times by batching the calls with arrays.

Calling the same function to chain modifications tends to be slower than calling the same function with all the transformations at the same time. Some PHP functions accept scalars or arrays, and using the later is more efficient.

```
<?php

$string = 'abcdef';

//str_replace() accepts arrays as arguments
$string = str_replace( ['a', 'b', 'c'],
                       ['A', 'B', 'C'],
                       $string);

// Too many calls to str_replace
$string = str_replace( 'a', 'A', $string);
$string = str_replace( 'b', 'B', $string);
$string = str_replace( 'c', 'C', $string);

// Too many nested calls to str_replace
$string = str_replace( 'a', 'A', str_replace( 'b', 'B', str_replace( 'c', 'C',
↪$string)));

?>
```

Potential replacements :

Function	Replacement
str_replace() str_ireplace() substr_replace() preg_replace() preg_replace_callback()	str_replace() str_replace() substr_replace() preg_replace() preg_replace_callback_array()

```
<?php
$subject = 'Aaaaaa Bbb';
```

(continues on next page)

(continued from previous page)

```
//preg_replace_callback_array() is better than multiple preg_replace_callback :
preg_replace_callback_array(
    [
        '~[a]+~i' => function ($match) {
            echo strlen($match[0]), ' matches for a found', PHP_EOL;
        },
        '~[b]+~i' => function ($match) {
            echo strlen($match[0]), ' matches for b found', PHP_EOL;
        }
    ],
    $subject
);

$result = preg_replace_callback('~[a]+~i', function ($match) {
            echo strlen($match[0]), ' matches for a found', PHP_EOL;
        }, $subject);

$result = preg_replace_callback('~[b]+~i', function ($match) {
            echo strlen($match[0]), ' matches for b found', PHP_EOL;
        }, $subject);

//str_replace() accepts arrays as arguments
$string = str_replace( ['a', 'b', 'c'],
                       ['A', 'B', 'C'],
                       $string);

// Too many calls to str_replace
$string = str_replace( 'a', 'A');
$string = str_replace( 'b', 'B');
$string = str_replace( 'c', 'C');

?>
```

Suggestions

- use str_replace() with arrays as arguments.
- use preg_replace() with arrays as arguments.
- use preg_replace_callback() for merging multiple complex calls.

Specs

Short name	Performances/MakeOneCall
Rulesets	*Performances*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*HuMo-Gen*, *Edusoho*

13.2.686 No mb_substr In Loop

Do not use loops on mb_substr().

mb_substr() always starts at the beginning of the string ot search for the nth char, and recalculate everything. This means that the first iterations are as fast as substr() (for comparison), while the longer the string, the slower mb_substr().

The recommendation is to use preg_split() with the *u* option, to split the string into an array. This save multiple recalculations.

```
<?php

// Split the string by characters
$array = preg_split('//u', $string, -1, PREG_SPLIT_NO_EMPTY);
foreach($array as $c) {
    doSomething($c);
}

// Slow version
$nb = mb_strlen($mb);
for($i = 0; $i < $nb; ++$i) {
    // Fetch a character
    $c = mb_substr($string, $i, 1);
    doSomething($c);
}

?>
```

See also Optimization: How I made my PHP code run 100 times faster and How to iterate UTF-8 string in PHP?.

Suggestions

- Use preg_split() and loop on its results.

Specs

Short name	Performances/MbStringInLoop
Rulesets	*Performances*
Exakt since	1.9.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.687 Memoize MagicCall

Cache calls to magic methods in local variable. Local cache is faster than calling again the magic method as soon as the second call, provided that the value hasn't changed.

`__get` is slower, as it turns a simple member access into a full method call.

```php
<?php

class x {
    private $values = array();

    function __get($name) {
        return $this->values[$name];
    }
    // more code to set values to this class
}

function foo(x $b) {
    $a = $b->a;
    $c = $b->c;

    $d = $c;      // using local cache, no new access to $b->__get($name)
    $e = $b->a;  // Second access to $b->a, through __get
}

function bar(x $b) {
    $a = $b->a;
    $c = $b->c;

    $b->bar2(); // this changes $b->a and $b->c, but we don't see it

    $d = $b->c;
    $e = $b->a;  // Second access to $b->a, through __get
}

?>
```

The caching is not possible if the processing of the object changes the value of the property.

See also __get performance questions with PHP, *Make Magic Concrete* and Benchmarking magic.

Suggestions

- Cache the value in a local variable, and reuse that variable
- Make the property concrete in the class, so as to avoid __get() altogether

Name	Default	Type	Description
minMagicCallsToGet	2	integer	Minimal number of calls of a magic property to make it worth locally caching.

Specs

Short name	Performances/MemoizeMagicCall
Rulesets	*Analyze*, *ClassReview*
Exakt since	1.8.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.688 Avoid Concat In Loop

> Concatenations inside a loop generate a lot of temporary variables. They are accumulated and tend to raise the memory usage, leading to slower performances.

It is recommended to store the values in an array, and then use implode() on that array to make the concatenation at once. The effect is positive when the source array has at least 50 elements.

```
<?php

// Concatenation in one operation
$tmp = array();
foreach(data_source() as $data) {
    $tmp[] = $data;
}
$final = implode('', $tmp);

// Concatenation in many operations
foreach(data_source() as $data) {
    $final .= $data;
}

?>
```

The same doesn't apply to addition and multiplication, with array_sum() and array_multiply(), as those operations work on the current memory allocation, and don't need to allocate new memory at each step.

See also PHP 7 performance improvements (3/5): Encapsed strings optimization.

Suggestions

- Collect all pieces in an array, then implode() the array in one call.

Specs

Short name	Performances/NoConcatInLoop
Rulesets	*Performances*, *Top10*
Exakt since	0.12.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
Examples	*SuiteCrm*, *ThinkPHP*

13.2.689 Avoid glob() Usage

glob() and scandir() sorts results by default. When that kind of sorting is not needed, save some time by requesting `NOSORT` with those functions.

Besides, whenever possible, use scandir() instead of glob().

```
<?php

// Scandir without sorting is the fastest.
scandir('docs/', SCANDIR_SORT_NONE);

// Scandir sorts files by default. Same as above, but with sorting
scandir('docs/');

// glob sorts files by default. Same as below, but no sorting
glob('docs/*', GLOB_NOSORT);

// glob sorts files by default. This is the slowest version
glob('docs/*');

?>
```

Using opendir() and a while loop may be even faster.

This analysis skips scandir() and glob() if they are expliciely configured with flags (aka, sorting is explicitly needed).

glob() accepts wildchar, such as `*`, that may not easily replaced with scandir() or opendir().

See also Putting glob to the test, How to list files recursively in a directory with PHP iterators and glob://.

Suggestions

- Use FilesystemIterator, DirectoryIterator classes.
- Use `RegexIterator` to filter any unwanted results from `FilesystemIterator`.
- Use `glob` protocol for files : $it = new DirectoryIterator('glob://path/to/examples/*.php');

Specs

Short name	Performances/NoGlob
Rulesets	*Performances*
Exakt since	0.9.6
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Phinx*, *NextCloud*

13.2.690 No Count With 0

Comparing count(), strlen() or mb_strlen() to 0 is a waste of resources. There are three distinct situations.

When comparing count() with 0, with ===, ==, !==, !=, it is more efficient to use empty(). empty() is a language construct that checks if a value is present, while count() actually load the number of element.

```
<?php

// Checking if an array is empty
if (count($array) == 0) {
    // doSomething();
}
// This may be replaced with
if (empty($array)) {
    // doSomething();
}

?>
```

When comparing count() strictly with 0 and >, it is more efficient to use `!(empty( ))`

```
<?php

// Checking if an array is empty
if (count($array) > 0) {
    // doSomething();
}
// This may be replaced with
if (!empty($array)) {
    // doSomething();
}

Of course comparing count() with negative values, or with >= is useless.

<?php

// Checking if an array is empty
if (count($array) < 0) {
    // This never happens
    // doSomething();
}

?>
```

Comparing count(), strlen() or mb_strlen() with other values than 0 cannot be replaced with a comparison with empty().

Note that this is a micro-optimisation : since PHP keeps track of the number of elements in arrays (or number of chars in strings), the total computing time of both operations is often lower than a ms. However, both functions tends to be heavily used, and may even be used inside loops.

See also count, strlen and mb_strlen.

Suggestions

- Use empty() on the data
- Compare the variable with a default value, such as an empty array

Specs

Short name	Performances/NotCountNull
Rulesets	*Performances*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Contao*, *WordPress*

13.2.691 Optimize Explode()

Limit explode() results at call time. explode() returns a string, after breaking it into smaller strings, with a delimiter.

By default, explode() breaks the whole string into smaller strings, and returns the array. When not all the elements of the returned array are necessary, using the third argument of explode() speeds up the process, by removing unnecessary work.

```
<?php

$string = '1,2,3,4,5,';

// explode() returns 2 elements, which are then assigned to the list() call.
list($a, $b) = explode(',', $string, 2);

// explode() returns 6 elements, only two of which are then assigned to the list()␣
↪call. The rest are discarded.
list($a, $b) = explode(',', $string, 2);

// it is not possible to skip the first elements, but it is possible to skip the last␣
↪ones.
echo explode(',', $string, 2)[1];

// This protects PHP, in case $string ends up with a lot of commas
$string = foo(); // usually '1,2' but not known
list($a, $b) = explode(',', $string, 2);
?>
```

Limiting explode() has no effect when the operation is already exact : it simply prevents explode() to cut more than needed if the argument is unexpectedly large.

This optimisation applies to split(), preg_split() and mb_split(), too.

This is a micro optimisation, unless the exploded string is large.

Suggestions

- Add a limit to explode() call

Specs

Short name	Performances/OptimizeExplode
Rulesets	*Performances*
Exakt since	2.1.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.692 Always Use Function With array_key_exists()

> array_key_exists() has been granted a special VM opcode, and is much faster. This applies to PHP 7.4 and more recent.

It requires that array_key_exists() is statically resolved, either with an initial \, or a `use function` expression. This doesn't affect the global namespace.

```
<?php

namespace my/name/space;

// do not forget the 'function' keyword, or it will apply to classes.
use function array_key_exists as foo; // the alias is not necessary, and may be␣
↪omitted.

// array_key_exists is aliased to foo :
$c = foo($a, $b);

// This call requires a fallback to global, and will be slow.
$c = array_key_exists($a, $b);

?>
```

This analysis is related to Php/ShouldUseFunction, and is a special case, that only concerns array_key_exists().

See also Add array_key_exists to the list of specialy compiled functions.

Suggestions

- Use the *use* command for arrray_key_exists(), at the beginning of the script
- Use an initial before array_key_exists()

- Remove the namespace

Specs

Short name	Performances/Php74ArrayKeyExists
Rulesets	*Performances*
Exakt since	1.8.4
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.693 Use PHP7 Encapsed Strings

PHP 7 has optimized the handling of double-quoted strings. In particular, double-quoted strings are much less memory hungry than classic concatenations.

PHP allocates memory at the end of the double-quoted string, making only one call to the allocator. On the other hand, concatenations are allocated each time they include dynamic content, leading to higher memory consumption.

```
<?php

$bar = 'bar';

/* PHP 7 optimized this */
$a = "foo and $bar";

/* This is PHP 5 code (aka, don't use it) */
$a = 'foo and ' . $bar;

// Constants can't be used with double quotes
$a = 'foo and ' . __DIR__;
$a = foo and __DIR__; // __DIR__ is not interpolated

?>
```

Concatenations are still needed with constants, static constants, magic constants, functions, static properties or static methods.

See also PHP 7 performance improvements (3/5): Encapsed strings optimization.

Specs

Short name	Performances/PHP7EncapsedStrings
Rulesets	*Performances*
Exakt since	1.0.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.694 Pre-increment

When possible, use the pre-increment operator (`++$i` or `--$i`) instead of the post-increment operator (`$i++` or `$i--`).

The latter needs an extra memory allocation that costs about 10% of performances.

```
<?php

// ++$i should be preferred over $i++, as current value is not important
for($i = 0; $i <10; ++$i) {
    // do Something
}

// ++$b and $b++ have different impact here, since $a will collect $b + 1 or $b,␣
↪respectively.
$a = $b++;

?>
```

This is a micro-optimisation. However, its usage is so widespread, including within loops, that it may eventually have an significant impact on execution time. As such, it is recommended to adopt this rule, and only consider changing legacy code as they are refactored for other reasons.

Suggestions

- Use the pre increment when the new value is not reused.

Specs

Short name	Performances/PrePostIncrement
Rulesets	*Analyze*, *CI-checks*, *Performances*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high
Examples	*ExpressionEngine*, *Traq*

13.2.695 Regex On Arrays

Avoid using a loop with arrays of regex or values. There are several PHP function which work directly on arrays, and much faster.

preg_grep() is able to extract all matching strings from an array, or non-matching strings. This usually saves a loop over the strings.

preg_filter() is able to extract all strings from an array, matching at least one regex in an array. This usually saves a double loop over the strings and the regex. The trick here is to provide '$0' as replacement, leading preg_filter() to replace the found string by itself.

Finally, preg_replace_callback() an preg_replace_callback_array() are also able to apply an array of regex to an array of strings, and then, apply callbacks to the found values.

```php
<?php

$regexs = ['/ab+c/', '/abd+/', '/abe+/'];
$strings = ['/abbbbc/', '/abd/', '/abeee/'];

// Directly extract all strings that match one regex
foreach($regexs as $regex) {
    $results[] = preg_grep($regex, $strings);
}

// extract all matching regex, by string
foreach($strings as $string) {
    $results[] = preg_filter($regexs, array_fill(0, count($regexs), '\$0'), $string);
}

// very slow way to get all the strings that match a regex
foreach($regexs as $regex) {
    foreach($strings as $string) {
        if (preg_match($regex, $string)) {
            $results[] = $string;
        }
    }
}

?>
```

See also preg_filter.

Suggestions

- Apply preg_match() to an array of string or regex, via preg_filter() or preg_grep().
- Apply preg_match() to an array of string or regex, via preg_replace_callback() or preg_replace_callback_array().

Specs

Short name	Performances/RegexOnArrays
Rulesets	*Performances*
Exakt since	1.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.696 Processing Collector

When accumulating data in a variable, within a loop, it is slow to apply repeatedly a function to the variable.

The example below illustrate the problem : `$collector` is build with element from `$array`. `$collector` actually gets larger and larger, slowing the in_array() call each time.

It is better to apply the preg_replace() to `$a`, a short variable, and then, add `$a` to the collector.

```php
<?php

// Fast way
$collector = '';
foreach($array as $a){
    $a = preg_replace('/__(.*?)__/', '<b>\$1</b>', $a);
    $collector .= $a;
}

// Slow way
$collector = '';
foreach($array as $a){
    $collector .= $a;
    $collector = preg_replace('/__(.*?)__/', '<b>\$1</b>', $collector);
}

?>
```

Suggestions

- Avoid applying the checks on the whole data, rather on the diff only.

Specs

Short name	Performances/RegexOnCollector
Rulesets	*Performances*
Exakt since	1.2.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.697 Simple Switch

Switches are faster when relying only on integers or strings.

Since PHP 7.2, simple switches that use only strings or integers are optimized. The gain is as great as the switch is big.

```php
<?php

// Optimized switch.
switch($b) {
    case "a":
        break;
    case "b":
        break;
    case "c":
        break;
    case "d":
        break;
    default :
```

(continues on next page)

(continued from previous page)

```
        break;
}

// Unoptimized switch.
// Try moving the foo() call in the default, to keep the rest of the switch optimized.
switch($c) {
    case "a":
        break;
    case foo($b):
        break;
    case "c":
        break;
    case "d":
        break;
    default :
        break;
}

?>
```

See also PHP 7.2's "switch" optimisations.

Suggestions

- Split the switch between literal and dynamic cases
- Remove the dynamic cases from the switch

Specs

Short name	Performances/SimpleSwitch
Rulesets	*Performances*
Exakt since	1.0.1
Php Version	7.2+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.698 Slow Functions

Avoid using those slow native PHP functions, and replace them with alternatives.

```
<?php

$array = source();

// Slow extraction of distinct values
$array = array_unique($array);

// Much faster extraction of distinct values
$array = array_keys(array_count_values($array));
```

(continues on next page)

(continued from previous page)

```
?>
```

Slow Function	Faster
array_diff() array_intersect() array_key_exists() array_map() array_search() array_udiff() array_uintersect() array_unshift() array_walk() in_array() preg_replace() strstr() uasort() uksort() usort() array_unique()	foreach() foreach() isset() and array_key_exists() foreach() array_flip() and isset() Use another way Use another way Use another way foreach() isset() strpos() strpos() Use another way Use another way Use another way array_keys() and array_count_values()

array_unique() has been accelerated in PHP 7.2 and may be used directly from this version on : Optimize ‘array_unique() <https://github.com/php/php-src/commit/6c2c7a023da4223e41fea0225c51a417fc8eb10d>‘_.

array_key_exists() has been accelerated in PHP 7.4 and may be used directly from this version on : Implement ZEND_ARRAY_KEY_EXISTS opcode to speed up ‘array_key_exists() <https://github.com/php/php-src/pull/3360>‘_.

Suggestions

- Replace the slow function with a faster version
- Remove the usage of the slow function

Specs

Short name	Performances/SlowFunctions
Rulesets	*Performances*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	avoid-those-slow-functions
Examples	*ChurchCRM*, *SuiteCrm*

13.2.699 strpos() Too Much

> strpos() covers the whole string before reporting 0. If the expected string is expected be at the beginning, or a fixed place, it is more stable to use substr() for comparison.

The longer the haystack (the searched string), the more efficient is that trick. The string has to be 10k or more to have impact, unless it is in a loop.

```
<?php

// This always reads the same amount of string
if (substr($html, 0, 6) === '<html>') {

}
```

(continues on next page)

(continued from previous page)

```
// When searching for a single character, checking with a known position ($string[
↪$position]) is even faster
if ($html[0] === '<') {

}

// With strpos(), the best way is to search for something that exist, and use absence␣
↪as worst case scenario
if (strpos($html, '<html>') > 0) {

} else {
    //
}

// When the search fails, the whole string has been read
if (strpos($html, '<html>') === 0) {

}

?>
```

This applies to stripos() too.

Suggestions

- Check for presence, and not for absence
- use substr() and compare the extracted string
- For single chars, try using the position in the string

Specs

Short name	Performances/StrposTooMuch
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.2.8
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*WordPress*

13.2.700 Substring First

Always start by reducing a string before applying some transformation on it. The shorter string will be processed faster.

```
<?php

// fast version
$result = strtolower(substr($string, $offset, $length));

```

(continues on next page)

(continued from previous page)

```
// slower version
$result = substr(strtolower($string), $offset, $length);
?>
```

The gain produced here is greater with longer strings, or greater reductions. They may also be used in loops. This is a micro-optimisation when used on short strings and single string reductions.

This works with any reduction function instead of substr(), like trim(), iconv(), etc.

Suggestions

- Always reduce the string first, then apply some transformation

Specs

Short name	Performances/SubstrFirst
Rulesets	*Performances*, *Suggestions*, *Top10*
Exakt since	1.0.1
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*SPIP*, *PrestaShop*

13.2.701 time() Vs strtotime()

time() is actually faster than strtotime() with 'now' key string.

```
<?php

// Faster version
$a = time();

// Slower version
$b = strtotime('now');

?>
```

This is a micro-optimisation. Relative gain is real, but small unless the function is used many times.

Suggestions

- Replace strtotime() with time(). Do not change strtotime() with other value than 'now'.

Specs

Short name	Performances/timeVsstrtotime
Rulesets	*Performances*
Exakt since	0.8.7
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Woocommerce*

13.2.702 Use array_slice()

Array_slice is de equivalent of substr() for arrays.

array_splice() is also possible, to remove a portion of array inside the array, not at the ends. This has no equivalent for strings.

```
<?php

$array = range(0, 9);

// Extract the 5 first elements
print_r(array_slice($array, 0, 5));

// Extract the 4 last elements
print_r(array_slice($array, -4));

// Extract the 2 central elements : 4 and 5
print_r(array_splice($array, 4, 2));

// slow way to remove the last elementst of an array
for($i = 0; $i < 4) {
    array_pop($array);
}

?>
```

See also array_slice and array_splice.

Suggestions

-

Specs

Short name	Performances/UseArraySlice
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.9.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.703 Use The Blind Var

When in a loop, it is faster to rely on the blind var, rather than the original source.

When the key is referenced in the foreach loop, it is faster to use the available container to access a value for reading.

Note that it is also faster to use the value with a reference to handle the writings.

```
<?php

// Reaching $source[$key] via $value is faster
foreach($source as $key => $value) {
    $coordinates = array('x' => $value[0],
                         'y' => $value[1]);
}

// Reaching $source[$key] via $source is slow
foreach($source as $key => $value) {
    $coordinates = array('x' => $source[$key][0],
                         'y' => $source[$key][1]);
}

?>
```

Suggestions

- Use the blind var

Specs

Short name	Performances/UseBlindVar
Rulesets	*Performances*
Exakt since	1.2.9
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.704 PHP Alternative Syntax

Identify the usage of alternative syntax in the code, for If then, Switch, While, For and Foreach.

```
<?php

// Normal syntax
if ($a == 1) {
    print $a;
}

// Alternative syntax : identical to the previous one.
if ($a == 1) :
    print $a;
endif;

?>
```

See also Alternative syntax.

Specs

Short name	Php/AlternativeSyntax
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.705 Argon2 Usage

Argon2 is an optionally compiled password hashing API.

Argon2 has been added to the password hashing API in PHP 7.2.

It is not available in older version. It also requires PHP to be compiled with the –with-password-argon2 option.

```
<?php

// Hashing a password with argon2
$hash = password_hash('password', PASSWORD_ARGON2I, ['memory_cost' => 1<<17,
                                                     'time_cost'   => PASSWORD_ARGON2_
↪DEFAULT_TIME_COST,
                                                     'threads'     => PASSWORD_ARGON2_
↪DEFAULT_THREADS]);

?>
```

See also Argon2 Password Hash.

Specs

Short name	Php/Argon2Usage
Rulesets	*CE*
Exakt since	1.0.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.706 array_key_exists() Works On Arrays

array_key_exists() requires arrays as second argument. Until PHP 7.4, objects were also allowed, yet it is now deprecated.

```
<?php

// Valid way to check for key
$array = ['a' => 1];
var_dump(array_key_exists('a', $array))


// Deprecated since PHP 7.4
$object = new Stdclass();
$object->a = 1;
var_dump(array_key_exists('a', $object))

?>
```

See also array_key_exists() with objects, and array_key_exists, and.

Suggestions

- Use the (array) cast to turn the object into an array
- Use the native PHP function proprety_exists() or isset() on the property to check them.

Specs

Short name	Php/ArrayKeyExistsWithObjects
Rulesets	*Analyze*, *CE*, *CompatibilityPHP74*
Exakt since	1.9.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.707 Assert Function Is Reserved

Avoid defining an `assert` function in namespaces.

While they work fine when the assertions are active (`zend.assertions=1`), calls to unqualified `assert` are optimized away when assertions are not active.

Since PHP 7.3, a fatal error is emitted : `Defining a custom `assert() <https://www.php.net/assert>`_ function is deprecated, as the function has special semantics.`

```
<?php
//      Run this with zend.assertions=1 and
// Then run this with zend.assertions=0

namespace Test {
    function assert() {
        global $foo;

        $foo = true;
    }
}

namespace Test {
    assert();

    var_dump(isset($foo));
}

?>
```

See also assert and User-defined assert function is optimized away with zend.assertions=-1.

Suggestions

- Rename the custom function with another name

Specs

Short name	Php/AssertFunctionIsReserved
Rulesets	*Analyze*, *CompatibilityPHP73*
Exakt since	1.3.9
Php Version	All
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	High

13.2.708 Assertions

Usage of assertions, to add checks within PHP code.

Assertions should be used as a debugging feature only. You may use them for sanity-checks that test for conditions that should always be TRUE and that indicate some programming errors if not or to check for the presence of certain features like extension functions or certain system limits and features.

```
<?php

function foo($string) {
```

(continues on next page)

(continued from previous page)

```
    assert(!empty($string), 'An empty string was provided!');

    echo '['.$string.']';
}

?>
```

See also assert.

Specs

Short name	Php/AssertionUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.709 Assign With And Precedence

The lettered logical operators yield to assignation. It may collect less information than expected.

It is recommended to use the &&, ^ and || operators, instead of and, or and xor, to prevent confusion.

```
<?php

// The expected behavior is
// The following are equivalent
 $a =  $b  && $c;
 $a = ($b && $c);

// The unexpected behavior is
// The following are equivalent
 $a = $b  and $c;
($a = $b) and $c;

?>
```

See also Operator Precedence.

Suggestions

- Always use symbol && rather than letter and
- To be safe, add parenthesis to enforce priorities

Specs

Short name	Php/AssignAnd
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.12.4
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	Very high
Examples	*xataface*

13.2.710 Assumptions

Assumptions in the code, that leads to possible bugs.

Some conditions may be very weak, and lead to errors. For example, the code below checks that the variable *$a* is not null, then uses it as an array. There is no relationship between 'not null' and 'being an array', so this is an assumption.

```
<?php

// Assumption : if $a is not null, then it is an array. This is not always the case.
function foo($a) {
    if ($a !== null) {
        echo $a['name'];
    }
}

// Assumption : if $a is not null, then it is an array. Here, the typehint will␣
↪ensure that it is the case.
// Although, a more readable test is is_array()
function foo(?array $a) {
    if ($a !== null) {
        echo $a['name'];
    }
}

?>
```

See also From assumptions to assertions.

Suggestions

- Make the context of the code more explicit

Specs

Short name	Php/Assumptions
Rulesets	*Analyze*
Exakt since	2.1.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.711 Autoloading

Usage of the autoloading feature of PHP.

```
<?php

spl_autoload_register('my_autoloader');

// Old way to autoload. Deprecated in PHP 7.2
function __autoload($class ) {}

?>
```

Defining the __autoload() function is obsolete since PHP 7.2.

See also __autoload.

Specs

Short name	Php/AutoloadUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.712 Avoid get_object_vars()

get_object_vars() changes behavior between PHP 7.3 and 7.4. It also behaves different within and outside a class.

```
<?php

// Code from Doug Bierer
$obj = new ArrayObject(['A' => 1,'B' => 2,'C' => 3]);
var_dump($obj->getArrayCopy());
var_dump(get_object_vars($obj));

?>
```

See also get_object_vars script on 3V4L and The Strange Case of 'ArrayObject <https://phptraining.net/articles/strange_case_of_array_object>'_.

Suggestions

- Use ArrayObject and getArrayCopy() method

Specs

Short name	Php/AvoidGetobjectVars
Rulesets	*CompatibilityPHP74*, *CompatibilityPHP80*
Exakt since	2.2.1
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.713 Avoid mb_dectect_encoding()

mb_dectect_encoding() is bad at guessing encoding.

For example, UTF-8 and ISO-8859-1 share some common characters : when a string is build with them it is impossible to differentiate the actual encoding.

```
<?php

$encoding = mb_encoding_detect($_GET['name']);

?>
```

See also mb_encoding_detect, PHP vs. The Developer: Encoding Character Sets, DPC2019: Of representation and interpretation: A unified theory - Arnout Boks.

Suggestions

- Store and transmit the data format

Specs

Short name	Php/AvoidMbDectectEncoding
Rulesets	*Analyze*
Exakt since	1.8.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.714 Avoid Real

PHP has two float data type : real and double. `real` is rarely used, and might be deprecated in PHP 7.4.

To prepare code, avoid using is_real() and the `(real)` typecast.

```
<?php

// safe way to check for float
if (!is_float($a)) {
    $a = (float) $a;
}

// Avoid doing that
if (!is_real($a)) {
    $a = (real) $a;
}

?>
```

See also PHP RFC: Deprecations for PHP 7.4.

Suggestions

- Replace is_real() by is_float()
- Replace (real) by (float)

Specs

Short name	Php/AvoidReal
Rulesets	*Suggestions*, *Top10*
Exakt since	1.3.9
Php Version	8.0-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.715 Avoid set_error_handler $context Argument

Avoid configuring set_error_handler() with a method that accepts 5 arguments. The last argument, `$errcontext`, is deprecated since PHP 7.2, and will be removed later.

```
<?php

// setting error_handler with an incorrect closure
set_error_handler(function($errno, $errstr, $errfile, $errline) {});

// setting error_handler with an incorrect closure
set_error_handler(function($errno, $errstr, $errfile, $errline, $errcontext) {});

?>
```

See also set_error_handler();

Suggestions

- Remove the 6th argument of registered handlers.

Specs

Short name	Php/AvoidSetErrorHandlerContextArg
Rulesets	*CompatibilityPHP72*
Exakt since	1.0.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
Examples	*shopware*, *Vanilla*

13.2.716 Use random_int()

rand() and mt_rand() should be replaced with random_int().

At worse, rand() should be replaced with mt_rand(), which is a drop-in replacement and srand() by mt_srand().

random_int() replaces rand(), and has no seeding function like srand().

Other sources of entropy that should be replaced by random_int() : microtime(), uniqid(), time(). Those a often combined with hashing functions and mixed with other sources of entropy, such as a salt.

```
<?php

// Avoid using this
$random = rand(0, 10);

// Drop-in replacement
$random = mt_rand(0, 10);

// Even better but different :
// valid with PHP 7.0+
try {
    $random = random_int(0, 10);
} catch (\Exception $e) {
    // process case of not enoug random values
}

// This is also a source of entropy, based on srand()
// random_int() is a drop-in replacement here
$a = sha256(uniqid());

?>
```

Since PHP 7, random_int() along with random_bytes(), provides cryptographically secure pseudo-random bytes, which are good to be used when security is involved. openssl_random_pseudo_bytes() may be used when the `OpenSSL` extension is available.

See also CSPRNG and OpenSSL.

Suggestions

- Use random_bytes() and randon_int(). At least, use them as a base for random data, and then add extra prefix and suffix, and a hash call on top.

Specs

Short name	Php/BetterRand
Rulesets	*Analyze*, *CI-checks*, *CompatibilityPHP71*, *Security*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	Very high
Examples	*Thelia*, *FuelCMS*

13.2.717 Cant Use Return Value In Write Context

empty() used to work only on data containers, such as variables. Until PHP 5.5, it was not possible to use directly expressions, such as functioncalls, inside an empty() function call : they were met with a 'Can't use function return value in write context' fatal error.

```
<?php

function foo($boolean) {
    return $boolean;
}

// Valid since PHP 5.5
echo empty(foo(true)) : 'true' : 'false';

?>
```

This also applies to methodcalls, static or not.

See also Cant Use Return Value In Write Context.

Specs

Short name	Php/CantUseReturnValueInWriteContext
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*
Exakt since	0.8.4
Php Version	5.5+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.718 Use Lower Case For Parent, Static And Self

The special parent, static and self keywords needed to be lowercase to be usable. This was fixed in PHP 5.5; otherwise, they would yield a 'PHP Fatal error: Class 'PARENT' not found'.

parent, static and self are traditionally written in lowercase only. Mixed case and Upper case are both valid, though.

```
<?php

class foo {
    const aConstante = 233;

    function method() {
        // Wrong case, error with PHP 5.4.* and older
        echo SELF::aConstante;

        // Always right.
        echo self::aConstante;
    }
}

?>
```

Until PHP 5.5, non-lowercase version of those keywords are generating a bug.

Suggestions

- Upgrade to PHP 5.6 or more recent

Specs

Short name	Php/CaseForPSS
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*
Exakt since	0.8.4
Php Version	5.5-
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.719 Cast Usage

List of all cast usage.

PHP does not require (or support) explicit type definition in variable declaration; a variable's type is determined by the context in which the variable is used.

```
<?php

if (is_int($_GET['x'])) {
    $number = (int) $_GET['x'];
} else {
    error_display('a wrong value was provided for "x"');
}
```

(continues on next page)

(continued from previous page)

```
?>
```

Until PHP 7.2, a `(unset)` operator was available. It had the same role as `unset()` as a function.

See also Type Juggling and unset.

Specs

Short name	Php/CastingUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.720 Cast Unset Usage

Usage of the *(unset)* cast operator. It is removed in PHP 8.0, and was deprecated since PHP 7.2.0.

```
<?php

$a = 1;
(unset) $a;

// functioncall is OK
unset($a);

?>
```

See also Unset casting.

Suggestions

- Replace *(unset)* with a call to unset().
- Remove the unset call altogether.

Specs

Short name	Php/CastUnsetUsage
Rulesets	*CE*, *CompatibilityPHP80*
Exakt since	2.1.8
Php Version	8.0-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.721 Class Const With Array

Constant defined with const keyword may be arrays but only stating with PHP 5.6. Define never accept arrays : it only accepts scalar values.

Specs

Short name	Php/ClassConstWithArray
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Exakt since	0.8.4
Php Version	5.5+
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	High

13.2.722 Class Function Confusion

Avoid classes and functions bearing the same name.

When functions and classes bear the same name, calling them may be confusing. This may also lead to forgotten 'new' keyword.

```
<?php

class foo {}

function foo() {}

// Forgetting the 'new' operator is easy
$object = new foo();
$object = foo();

?>
```

Suggestions

- Use a naming convention to distinguish functions and classes
- Rename the class or the function (or both)
- Use an alias with a *use* expression

Specs

Short name	Php/ClassFunctionConfusion
Rulesets	*Semantics*
Exakt since	0.10.2
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.723 Close Tags

PHP manual recommends that script should be left open, without the final closing ?>. This way, one will avoid the infamous bug 'Header already sent', associated with left-over spaces, that are lying after this closing tag.

Specs

Short name	Php/CloseTags
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	leave-last-closing-out

13.2.724 Close Tags Consistency

PHP scripts may omit the final closing tag.

This is a convention, used to avoid the infamous 'headers already sent' error message, that appears when a script with extra invisible spaces is included before actually emitting the headers.

The PHP manual recommends : If a file is pure PHP code, it is preferable to omit the PHP closing tag at the end of the file.. (See PHP Tags):

```
.. code-block:: php

   <?php

   class foo {

   }
```

Specs

Short name	Php/CloseTagsConsistency
Rulesets	none
Exakt since	0.9.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.725 Closure May Use $this

$this is automatically accessible to closures.

When closures were introduced in PHP, they couldn't use the $this variable, making is cumbersome to access local properties when the closure was created within an object.

```
<?php

// Invalid code in PHP 5.4 and less
class Test
{
    public function testing()
    {
        return function() {
            var_dump($this);
        };
    }
}

$object = new Test;
$function = $object->testing();
$function();

?>
```

This is not the case anymore since PHP 5.4.

See also Anonymous functions.

Specs

Short name	Php/ClosureThisSupport
Rulesets	*CompatibilityPHP53*
Exakt since	0.8.4
Php Version	5.4-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.726 Coalesce

Usage of coalesce operator, in PHP since PHP 5.3.

Note that the coalesce operator is a special case of the ternary operator.

```
<?php

// Coalesce operator, since PHP 5.3
$a = $b ?: 'default value';

// Equivalent to $a = $b ? $b : 'default value';

?>
```

See also Ternary Operator.

Specs

Short name	Php/Coalesce
Rulesets	*CE*
Exakt since	0.8.4
Php Version	7.0+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.727 Coalesce Equal

Usage of coalesce assignement operator. The operator is available in PHP since PHP 7.4.

```
<?php

// Coalesce operator, since PHP 5.3
$a ??= 'default value';

// Equivalent to the line above
$a = $a ?? 'default value';

?>
```

See also Ternary Operator.

Suggestions

- Use the short assignement syntax

Specs

Short name	Php/CoalesceEqual
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*
Exakt since	2.0.4
Php Version	7.4+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.728 Compact Inexistant Variable

Compact() doesn't warn when it tries to work on an inexistant variable. It just ignores the variable.

This behavior changed in PHP 7.3, and compact() now emits a warning when the compacted variable doesn't exist.

```
<?php

function foo($b = 2) {
    $a = 1;
    // $c doesn't exists, and is not compacted.
    return compact('a', 'b', 'c');
}
?>
```

For performances reasons, this analysis only works inside methods and functions.

See also compact and PHP RFC: Make compact function reports undefined passed variables.

Suggestions

- Fix the name of variable in the compact() argument list
- Remove the name of variable in the compact() argument list

Specs

Short name	Php/CompactInexistant
Rulesets	*CompatibilityPHP73*, *Suggestions*
Exakt since	1.2.9
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.729 Concat And Addition

Precedence between addition and concatenation has changed. In PHP 7.4, addition has precedence, and before, addition and concatenation had the same precedence.

From the RFC : `Currently the precedence of '.', '+' and '-' operators are equal. Any combination of these operators are simply evaluated left-to-right.`

This is counter-intuitive though: you rarely want to add or subtract concatenated strings which in general are not numbers. However, given PHP's capability of seamlessly converting an integer to a string, concatenation of these values is desired.''

```
<?php
// Extracted from the RFC
echo sum: . $a + $b;

// current behavior: evaluated left-to-right
echo (sum: . $a) + $b;
```

(continues on next page)

(continued from previous page)

```
// desired behavior: addition and subtraction have a higher precendence
echo sum : . ($a + $b);

?>
```

This analysis reports any addition and concatenation that are mixed, without parenthesis. Addition also means substraction here, aka using + or -.

The same applies to bitshift operations, << and >>. There is no RFC for this change.

See also Change the precedence of the concatenation operator.

Suggestions

- Add parenthesis around the addition to ensure its expected priority
- Move the addition outside the concatenation

Specs

Short name	Php/ConcatAndAddition
Rulesets	*Analyze*, *CE*, *CI-checks*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP74*, *CompatibilityPHP80*, *Top10*
Exakt since	1.8.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.730 Const With Array

The const keyword supports array. This feature was added in PHP 5.6.

The array must be filled with other constants. It may also be build using the '+' operator.

```
<?php

const PRIMES = [2, 3, 5, 7];

class X {
```

(continues on next page)

(continued from previous page)

```
    const TWENTY_THREE = 23;
    const MORE_PRIMES = PRIMES + [11, 13, 17, 19];
    const EVEN_MORE_PRIMES = self::MORE_PRIMES + [self::TWENTY_THREE];
}

?>
```

See also Class Constants and Constants Syntax.

Specs

Short name	Php/ConstWithArray
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Exakt since	0.8.4
Php Version	5.5+
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.731 Cookies Variables

Cookies names, used across the application.

```
<?php

if (isset($_COOKIE['myCookie'])) {
    // Usual method for reading and setting cookies
    $_COOKIE['myCookie']++;
}

// Usual method for writing cookies
setcookie('myCookie', $value);

?>
```

See also setcookie.

Specs

Short name	Php/CookiesVariables
Rulesets	none
Exakt since	0.12.16
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.732 Use is_countable

is_countable() checks if a variables holds a value that can be counted. It is recommended to use it before calling count().

is_countable() accepts arrays and object whose class implements countable.

```
<?php

function foo($arg) {
    if (!is_countable($arg)) {
        // $arg cannot be passed to count()
        return 0
    }
    return count($arg);
}

function bar($arg) {
    if (!is_array($arg) and !$x instanceof \Countable) {
        // $arg cannot be passed to count()
        return 0
    }

    return count($arg);
}

?>
```

See also PHP RFC: is_countable.

Suggestions

- Use is_countable()
- Create a compatibility function that replaces is_countable() until the code is ready for PHP 7.3

Specs

Short name	Php/CouldUseIsCountable
Rulesets	*Suggestions*
Exakt since	1.3.8
Php Version	7.3+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.733 Could Use Promoted Properties

Promoted properties reduce PHP code at __construct() time. This feature is available in PHP 8.0.

```
<?php

class x {
```

(continues on next page)

(continued from previous page)

```
    function __construct($a, $b) {
        // $a argument may be promoted to property $c
        $this->c = $a;

        // $b argument cannot be upgraded to property, as it is updated.
        // Move the addition to the new call, or keep the syntax below
        $this->d = $b + 2;
    }
}

?>
```

See also PHP 8: Constructor property promotion and PHP RFC: Constructor Property Promotion.

Suggestions

- Update the constructor syntax, and remove the property specification.

Specs

Short name	Php/CouldUsePromotedProperties
Rulesets	*Suggestions*
Exakt since	2.1.9
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.734 Crc32() Might Be Negative

crc32‘() <https://www.php.net/>‘_ may return a negative number, on 32 bits platforms.

According to the manual : Because PHP’s integer type is signed many `CRC32` checksums will result in negative integers on 32 bits platforms. On 64 bits installations, all crc32‘() <https://www.php.net/>‘_ results will be positive integers though.

```
<?php

// display the checksum with %u, to make it unsigned
echo sprintf('%u', crc32($str));

// turn the checksum into an unsigned hexadecimal
echo dechex(crc32($str));

// avoid concatenating crc32 to a string, as it may be negative on 32bits platforms
echo 'prefix'.crc32($str);

?>
```

See also crc32‘() <https://www.php.net/crc32>‘_.

Specs

Short name	Php/Crc32MightBeNegative
Rulesets	*Analyze*
Exakt since	0.11.0
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.735 Crypto Usage

Usage of cryptography and hashes functions.

The functions listed are the native PHP functions, and do not belong to a specific extension, like `OpenSSL`, `mcrypt` or `mhash`.

Cryptography and hashes are mainly used for storing sensitive data, such as passwords, or to verify authenticity of data. They may also be used for name-randomization with cache.

```
<?php

if (md5($_POST['password']) === $row['password_hash']) {
    user_login($user);
} else {
    error('Wrong password');
}
?>
```

See also Cryptography Extensions.

Specs

Short name	Php/CryptoUsage
Rulesets	*CE*
Exakt since	1.0.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.736 Date Formats

Inventory of date formats used in the code.

Date format are detected with

```
<?php

$time = time();
// This is a formated date
```

(continues on next page)

(continued from previous page)

```
echo date('r', $time);

?>
```

See also Date and Time.

Suggestions

-

Specs

Short name	Php/DateFormats
Rulesets	none
Exakt since	0.12.16
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.737 __debugInfo() Usage

The magic method __debugInfo() provides a custom way to dump an object.

It has been introduced in PHP 5.6. In the previous versions of PHP, this method is ignored and won't be called when debugging.

```
<?php

// PHP 5.6 or later
class foo {
    private $bar = 1;
    private $reallyHidden = 2;

    function __debugInfo() {
        return ['bar' => $this->bar,
                'reallyHidden' => 'Secret'];
    }
}

$f = new Foo();
var_dump($f);

/* Displays :
object(foo)#1 (2) {
  [bar]=>
  int(1)
  [reallyHidden]=>
  string(6) Secret
}
*/
```

(continues on next page)

(continued from previous page)

```
?>
```

See also Magic methods.

Specs

Short name	Php/debugInfoUsage
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Exakt since	0.8.4
Php Version	5.6+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Dolibarr*

13.2.738 Encoding Usage

Usage of declare(encoding =);.

```
<?php

// Setting encoding for the file;
    declare(encoding = 'UTF-8');

?>
```

See also declare.

Specs

Short name	Php/DeclareEncoding
Rulesets	*CE*
Exakt since	0.12.1
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.739 strict_types Preference

`strict_types` is a PHP mode where typehint are enforced strictly or weakly. By default, it is weak typing, allowing backward compatibility with previous versions.

This analysis reports if `strict_types` are used systematically or not. `strict_types` affects the calling file, not the definition file.

```
<?php

// define strict_types
declare(strict_types = 1);

foo(1);

?>
```

See also Strict typing.

Specs

Short name	Php/DeclareStrict
Rulesets	*CE*
Exakt since	0.12.2
Php Version	7.0+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.740 Declare strict_types Usage

> Usage of `strict_types`. By default, PHP attempts to change the original type to match the type specified by the type-declaration. With an explicit `strict_types` declaration, PHP ensures that the incoming argument has the exact type.

`strict_types` were introduced in PHP 7.0.

```
<?php

// Setting strict_types;
    declare(strict_types = 1);

    function foo(int $i) {
        echo $i;
    }

    // Always valid : displays 1
    foo(1);
    // with strict types, this emits an error
    // without strict types, this displays 1
    foo(1.7);

?>
```

See also declare.

Specs

Short name	Php/DeclareStrictType
Rulesets	*CE*
Exakt since	0.12.1
Php Version	7.0+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.741 Ticks Usage

Usage of declare(ticks =);.

```
<?php

// Setting ticks value
    declare(ticks = 'UTF-8');

?>
```

See also declare.

Specs

Short name	Php/DeclareTicks
Rulesets	*CE*
Exakt since	0.12.1
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.742 Define With Array

PHP 7.0 has the ability to define an array as a constant, using the define() native call. This was not possible until that version, only with the const keyword.

```
<?php

//Defining an array as a constant
define('MY_PRIMES', [2, 3, 5, 7, 11]);

?>
```

Specs

Short name	Php/DefineWithArray
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	7.0+
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	High

13.2.743 Deprecated PHP Functions

The following functions are deprecated. It is recommended to stop using them now and replace them with a durable equivalent.

Note that these functions may be still usable : they generate warning that help tracking their usage in the log. To eradicate their usage, watch the logs, and update any deprecated warning. This way, the code won't be stuck when the function is actually removed from PHP.

```
<?php

// This is the current function
list($day, $month, $year) = explode('/', '08/06/1995');

// This is deprecated
list($day, $month, $year) = split('/', '08/06/1995');

?>
```

Suggestions

- Replace those deprecated with modern syntax
- Stop using deprecated syntax

Specs

Short name	Php/Deprecated
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-deprecated
Examples	*Dolphin*

13.2.744 Detect Current Class

Detecting the current class should be done with *self::class* or *static::class* operator.

__CLASS__ may be replaced by `self\:\:class`. get_called_class() may be replaced by `static\:\:class`.

__CLASS__ and get_called_class() are set to be deprecated in PHP 7.4.

```
<?php

class X {
    function foo() {
        echo __CLASS__.\n;           // X
        echo self::class.\n;         // X

        echo get_called_class().\n;  // Y
        echo static::class.\n;       // Y
    }
}

class Y extends X {}

$y = new Y();
$y->foo();

?>
```

See also PHP RFC: Deprecations for PHP 7.4.

Suggestions

- Use the self::class operator to detect the current class name, instead of __CLASS__ and get_class().
- Use the static::class operator to detect the current called class name, instead of get_called_class().

Specs

Short name	Php/DetectCurrentClass
Rulesets	*CE*, *CompatibilityPHP74*, *Suggestions*
Exakt since	1.3.8
Php Version	8.0-
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.745 Direct Call To __clone()

Direct call to magic method __clone() was forbidden. It is allowed since PHP 7.0.

From the RFC : `Doing calls like $obj->`__clone(<https://www.php.net/manual/en/language.oop5.magic.php>`_ ) is now allowed. This was the only magic method that had a compile-time check preventing some calls to it, which doesn't make sense. If we allow all other magic methods to be called, there's no reason to forbid this one.`

```
<?php
```

(continues on next page)

(continued from previous page)

```
    class Foo {
        function __clone() {}
    }

    $a = new Foo;
    $a->__clone();
?>
```

See also Directly calling '__clone is allowed <https://wiki.php.net/rfc/abstract_syntax_tree#directly_calling_clone_is_allowed>'_.

Suggestions

- Use the clone operator to call the __clone magic method

Specs

Short name	Php/DirectCallToClone
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	1.4.8
Php Version	7.0+
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	High

13.2.746 Unknown Directive Name

Unknown directives names used in the code.

The following list has directive mentioned in the code, that are not known from PHP or any extension. If this is due to a mistake, the directive must be fixed to be actually useful.

```
<?php

// non-existing directive
$reporting_error = ini_get('reporting_error');
$error_reporting = ini_get('error_reproting'); // Note the inversion
if (ini_set('dump_globals')) {
    // doSomething()
}

// Correct directives
$error_reporting = ini_get('reporting_error');
if (ini_set('xdebug.dump_globals')) {
    // doSomething()
}

?>
```

Specs

Short name	Php/DirectiveName
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.747 Directives Usage

List of the directives mentioned in the code.

```
<?php

//accessing the configuration to change it
ini_set('timelimit', -1);

//accessing the configuration to check it
ini_get('safe_mode');

?>
```

Specs

Short name	Php/DirectivesUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.748 Dl() Usage

Dynamically load PHP extensions with dl().

```
<?php

    // dynamically loading ext/vips
     dl('vips.' . PHP_SHLIB_SUFFIX);

?>
```

See also dl.

Specs

Short name	Php/DlUsage
Rulesets	*CE*
Exakt since	1.0.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.749 Don't Pollute Global Space

Avoid creating definitions in the global name space.

The global namespace is the default namespace, where all functions, classes, constants, traits and interfaces live. The global namespace is also known as the root namespace.

In particular, PHP native classes usually live in that namespace. By creating functions in that namespace, the code may encounter naming conflict, when the PHP group decides to use a name that the code also uses. This already happened in PHP version 5.1.1, where a `Date` native class was introduced, and had to be disabled in the following minor version.

Nowadays, conflicts appear between components, which claim the same name.

See also Using namespaces: fallback to global function/constant.

Suggestions

- Create a namespace for your code, and store your definition there.

Specs

Short name	Php/DontPolluteGlobalSpace
Rulesets	*Analyze*
Exakt since	2.1.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.750 <?= Usage

Usage of the <?= tag, that echo's directly the following content.

<?= $variable; ?>

Specs

Short name	Php/EchoTagUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.751 Ellipsis Usage

Usage of the ellipsis keyword. The keyword is three dots : ... . It is also named variadic or splat operator.

It may be in function definitions, either in functioncalls.

... allows for packing or unpacking arguments into an array.

```
<?php

$args = [1, 2, 3];
foo(...$args);
// Identical to foo(1,2,3);

function bar(...$a) {
    // Identical to : $a = func_get_args();
}
?>
```

See also PHP RFC: Syntax for variadic functions, PHP 5.6 and the Splat Operator, and Variable-length argument lists.

Specs

Short name	Php/EllipsisUsage
Rulesets	*CE*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Exakt since	0.8.4
Php Version	5.6+
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.752 Empty List

Empty list() are not allowed anymore in PHP 7. There must be at least one variable in the list call.

```
<?php

//Not accepted since PHP 7.0
list() = array(1,2,3);

//Still valid PHP code
```

(continues on next page)

(continued from previous page)

```
list(,$x) = array(1,2,3);

?>
```

Suggestions

- Remove empty list() calls

Specs

Short name	Php/EmptyList
Rulesets	*Analyze*, *CompatibilityPHP70*
Exakt since	0.8.4
Php Version	7.0-
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.753 Error_Log() Usage

Usage of error_log() function. This leads to checking the configuration of `error_log` in the PHP configuration directives.

```
<?php

error_log(logging message\n);

?>
```

Specs

Short name	Php/ErrorLogUsage
Rulesets	*CE*
Exakt since	0.10.0
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.754 Exponent Usage

Usage of the ** operator or **=, to make exponents.

```
<?php

$eight = 2 ** 3;
```

(continues on next page)

(continued from previous page)

```
$sixteen = 4;
$sixteen \*\*\= 2;

?>
```

See also Arithmetic Operators.

Suggestions

- Use the operator when no literal negative number is in the expression

Specs

Short name	Php/ExponentUsage
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Exakt since	0.8.4
Php Version	5.6+
Severity	Major
Time To Fix	Instant (5 mins)
Precision	Very high

13.2.755 Php/FailingAnalysis

Specs

Short name	Php/FailingAnalysis
Rulesets	none
Exakt since	1.2.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.756 Filter To add_slashes()

`FILTER_SANITIZE_MAGIC_QUOTES` is deprecated. In PHP 7.4, it should be replaced with addslashes()

According to the migration RDFC : 'Magic quotes were deprecated all the way back in PHP 5.3 and later removed in PHP 5.4. The filter extension implements a sanitization filter that mimics this behavior of magic_quotes by calling addslashes() on the input in question.'

```
<?php

// Deprecated way to filter input
$var = filter_input($input, FILTER_SANITIZE_MAGIC_QUOTES);

```

(continues on next page)

(continued from previous page)

```
// Alternative way to filter input
$var = addslashes($input);

?>
```

addslashes() used to filter data while building SQL queries, to prevent injections. Nowadays, prepared queries are a better option.

See also Deprecations for PHP 7.4.

Suggestions

- Replace `FILTER_SANITIZE_MAGIC_QUOTES` with addslashes()
- Replace `FILTER_SANITIZE_MAGIC_QUOTES` with an adapted escaping system

Specs

Short name	Php/FilterToAddSlashes
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.757 Flexible Heredoc

Flexible syntax for Heredoc.

The new flexible syntax for heredoc and nowdoc enable the closing marker to be indented, and remove the new line requirement after the closing marker.

It was introduced in PHP 7.3.

```
<?php

// PHP 7.3 and newer
foo($a = <<<END

    flexible syntax
    with extra indentation

    END);

// All PHP versions
$a = <<<END

    Normal syntax

END;
```

(continues on next page)

(continued from previous page)

```
?>
```

This syntax is backward incompatible : once adopted in the code, previous versions won't compile it.

See also Heredoc and Flexible Heredoc and Nowdoc Syntaxes.

Specs

Short name	Php/FlexibleHeredoc
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*
Exakt since	1.2.9
Php Version	7.3+
Severity	Critical
Time To Fix	Instant (5 mins)
Precision	High

13.2.758 Wrong fopen() Mode

Wrong file opening for fopen().

fopen() has a few modes, as described in the documentation : 'r', 'r+', for reading; 'w', 'w+' for writing; 'a', 'a+' for appending; 'x', 'x+' for modifying; 'c', 'c+' for writing and locking, 't' for text files and windows only. An optional 'b' may be used to make the fopen() call more portable and binary safe. Another optional 't' may be used to make the fopen() call process automatically text input : this one should be avoided.

```
<?php

// open the file for reading, in binary mode
$fp = fopen('/tmp/php.txt', 'rb');

// New option e in PHP 7.0.16 and 7.1.2 (beware of compatibility)
$fp = fopen('/tmp/php.txt', 'rbe');

// Unknown option x
$fp = fopen('/tmp/php.txt', 'rbx');

?>
```

Any other values are not understood by PHP.

Suggestions

- Check the docs, choose the right opening mode.

Specs

Short name	Php/FopenMode
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Tikiwiki*, *HuMo-Gen*

13.2.759 Foreach Don't Change Pointer

foreach loops use their own internal cursor.

A foreach loop won't change the internal pointer of the array, as it works on a copy of the source. Hence, applying array pointer's functions such as current() or next() to the source array won't have the same behavior in PHP 5 than PHP 7.

This only applies when a foreach() by reference is used.

```
<?php

$numbers = range(1, 10);
next($numbers);
foreach($numbers as &$number){
    print $number;
    print current($numbers).\n; // Always
}

?>
```

See also foreach no longer changes the internal array pointer and foreach.

Suggestions

- Do not change the pointer on the source array while in the loop

Specs

Short name	Php/ForeachDontChangePointer
Rulesets	*CompatibilityPHP70*
Exakt since	0.8.4
Php Version	7.0-
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.760 Foreach On Object

Foreach on object looks like a typo. This is particularly true when both object and member are variables.

Foreach on an object member is a legit PHP syntax, though it is very rare : blind variables rarely have to be securing in an object to be processed.

```
<?php

// Looks suspicious
foreach($array as $o -> $b) {
    doSomething();
}

// This is the real thing
foreach($array as $o => $b) {
    doSomething();
}

?>
```

Specs

Short name	Php/ForeachObject
Rulesets	*Analyze*
Exakt since	1.1.6
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.761 $GLOBALS Or global

Usually, PHP projects make a choice between the global keyword, and the $GLOBALS variable. Sometimes, the project has no recommendations.

When your project use a vast majority of one of the convention, then the analyzer will report all remaining inconsistently cased constant.

```
<?php

global $a, $b, $c, $d, $e, $f, $g, $h, $i, $j, $k, $l, $m;

// This access is inconsistent with the previous usage
$GLOBALS['a'] = 2;

?>
```

Specs

Short name	Php/GlobalsVsGlobal
Rulesets	none
Exakt since	0.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.762 Simple Global Variable

The global keyword should only be used with simple variables. Since PHP 7, it cannot be used with complex or dynamic structures.

```
<?php

// Forbidden in PHP 7
global $normalGlobal;

// Forbidden in PHP 7
global $$variable->global ;

// Tolerated in PHP 7
global ${$variable->global};

?>
```

Suggestions

- Add curly braces so the syntax is compatibles PHP 5 and PHP 7
- Remove this syntax, and access the variable through another way : argument, array, property.

Specs

Short name	Php/GlobalWithoutSimpleVariable
Rulesets	*CompatibilityPHP70*
Exakt since	0.8.4
Php Version	7.0-
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.763 Goto Names

List of all goto labels used in the code.

```
<?php

GOTO_NAME_1:

// reports the usage of GOTO_NAME_1
goto GOTO_NAME_1;

UNUSED_GOTO_NAME_1:

?>
```

Specs

Short name	Php/Gotonames
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high
ClearPHP	no-goto

13.2.764 Group Use Declaration

The group use declaration is used in the code.

```
<?php

// Adapted from the RFC documentation
// Pre PHP 7 code
use some\name_space\ClassA;
use some\name_space\ClassB;
use some\name_space\ClassC as C;

use function some\name_space\fn_a;
use function some\name_space\fn_b;
use function some\name_space\fn_c;

use const some\name_space\ConstA;
use const some\name_space\ConstB;
use const some\name_space\ConstC;

// PHP 7+ code
use some\name_space\{ClassA, ClassB, ClassC as C};
use function some\name_space\{fn_a, fn_b, fn_c};
use const some\name_space\{ConstA, ConstB, ConstC};

?>
```

See also Group Use Declaration RFC and Using namespaces: Aliasing/Importing.

Specs

Short name	Php/GroupUseDeclaration
Rulesets	*CE*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.10.7
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.765 Group Use Trailing Comma

The usage of a final empty slot in array() was allowed with use statements. This works in PHP 7.2 and more recent.

Although this empty instruction is ignored at execution, this allows for clean presentation of code, and short diff when committing in a VCS.

```
<?php

// Valid in PHP 7.2 and more recent.
use a\b\{c,
         d,
         e,
         f,
        };

// This won't compile in 7.1 and older.

?>
```

See also Trailing Commas In List Syntax and Revisit trailing commas in function arguments.

Specs

Short name	Php/GroupUseTrailingComma
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*
Exakt since	0.12.3
Php Version	7.2+
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.766 __halt_compiler

__halt_compiler() usage.

```
<?php

// open this file
$fp = fopen(__FILE__, 'r');

// seek file pointer to data
fseek($fp, __COMPILER_HALT_OFFSET__);

// and output it
var_dump(stream_get_contents($fp));

// the end of the script execution
__halt_compiler(); the installation data (eg. tar, gz, PHP, etc.)

?>
```

Specs

Short name	Php/Haltcompiler
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.767 Hash Algorithms

There is a long but limited list of hashing algorithm available to PHP. The one found doesn't seem to be existing.

```
<?php

// This hash has existed in PHP. Check with hash_algos() if it is available on your␣
↪system.
echo hash('ripmed160', 'The quick brown fox jumped over the lazy dog.');

// This hash doesn't exist
echo hash('ripemd160', 'The quick brown fox jumped over the lazy dog.');

?>
```

See also hash_algos.

Suggestions

- Use a hash algorithm that is available on several PHP versions
- Fix the name of the hash algorithm

Specs

Short name	Php/HashAlgos
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.768 Hash Algorithms Incompatible With PHP 5.3

List of hash algorithms incompatible with PHP 5.3.

```
<?php

// Compatible only with 5.3 and more recent
echo hash('md2', 'The quick brown fox jumped over the lazy dog.');

// Always compatible
echo hash('ripemd320', 'The quick brown fox jumped over the lazy dog.');

?>
```

See also hash_algos.

Specs

Short name	Php/HashAlgos53
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.769 Hash Algorithms Incompatible With PHP 5.4/5.5

List of hash algorithms incompatible with PHP 5.4 and 5.5.

```
<?php

// Compatible only with 5.4 and more recent
echo hash('fnv132', 'The quick brown fox jumped over the lazy dog.');
```

(continues on next page)

(continued from previous page)

```
// Always compatible
echo hash('ripemd320', 'The quick brown fox jumped over the lazy dog.');

?>
```

See also hash_algos.

Specs

Short name	Php/HashAlgos54
Rulesets	*CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*
Exakt since	0.8.4
Php Version	5.4-
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.770 Hash Algorithms Incompatible With PHP 7.1-

List of hash algorithms incompatible with PHP 7.1 and more recent. At the moment of writing, this is compatible up to 7.3.

The hash algorithms were introduced in PHP 7.1.

```
<?php

// Compatible only with 7.1 and more recent
echo hash('sha512/224', 'The quick brown fox jumped over the lazy dog.');

// Always compatible
echo hash('ripemd320', 'The quick brown fox jumped over the lazy dog.');

?>
```

See also hash_algos.

Specs

Short name	Php/HashAlgos71
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*
Exakt since	1.3.4
Php Version	7.1-
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.771 Hash Algorithms Incompatible With PHP 7.4-

List of hash algorithms incompatible with PHP 7.3 and older recent. At the moment of writing, this is compatible up to 7.4s.

The hash algorithms were introduced in PHP 7.4s.

```
<?php

// Compatible only with 7.1 and more recent
echo hash('crc32cs', 'The quick brown fox jumped over the lazy dog.');

// Always compatible
echo hash('ripemd320', 'The quick brown fox jumped over the lazy dog.');

?>
```

See also hash_algos.

Specs

Short name	Php/HashAlgos74
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.3.4
Php Version	7.4-
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.772 Hash Will Use Objects

The ext/hash extension used resources, and is being upgraded to use resources.

```
<?php

// Post 7.2 code
    $hash = hash_init('sha256');
    if (!is_object($hash)) {
        trigger_error('error');
    }
    hash_update($hash, $message);

// Pre-7.2 code
    $hash = hash_init('md5');
    if (!is_resource($hash)) {
        trigger_error('error');
    }
    hash_update($hash, $message);

?>
```

See also Move ext/hash from resources to objects.

Specs

Short name	Php/HashUsesObjects
Rulesets	*CompatibilityPHP72*
Exakt since	1.0.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.773 idn_to_ascii() New Default

The default parameter value of idn_to_ascii() and idn_to_utf8‘() <https://www.php.net/>‘_ is now INTL_IDNA_VARIANT_UTS46 instead of the deprecated INTL_IDNA_VARIANT_2003.

```
<?php

echo idn_to_ascii('täst.de');

?>
```

See also idn_to_ascii, idn_to_utf8 and Unicode IDNA Compatibility Processing.

Suggestions

- Explicitely add the second parameter to the idn_to_ascii() and idn_to_utf8() functions.

Specs

Short name	Php/IdnUts46
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.5.0
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.774 Implode One Arg

implode() may be called with one arg. It is recommended to avoid it.

Using two arguments makes it less surprising to new comers, and consistent with explode() syntax.

```
<?php

$array = range('a', 'c');

// empty string is the glue
print implode('', $array);

// only the array : PHP uses the empty string as glue.
// Avoid this
print implode($array);

?>
```

See also implode.

Suggestions

- Add an empty string as first argument

Specs

Short name	Php/ImplodeOneArg
Rulesets	*Suggestions*, *php-cs-fixable*
Exakt since	1.7.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.775 Incoming Values

The names of the variables that are passed via the superglobals.

```
<?php

$x = $_GET['y']; // y is the incoming variable

?>
```

Specs

Short name	Php/IncomingValues
Rulesets	none
Exakt since	1.7.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.776 Incoming Variables

Incoming names, used across the application.

Incoming variables are first-level index in `$_POST`, `$_GET`, `$_COOKIE`, `$_REQUEST` and `$_FILE`;

`$_SESSION` and `$_ENV` are not reported as incoming data, as they are not supposed to be manipulated by normal user.

Dynamic names are not reported too.

```
<?php

$name = $_GET['name'];
$cookie = $_COOKIE['cookie'];

// 'archive' is the incoming variable, not 'file_name'
$file_name = $_FILE['archive']['file_name'];

// This is not reported, because it is from $_ENV.
$db_pass = $_ENV['DB_PASS'];

// This is not reported, because it is dynamic
$x = 'userId';
$userId = $_GET[$x];

?>
```

Specs

Short name	Php/IncomingVariables
Rulesets	none
Exakt since	0.12.16
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.777 Incompilable Files

Files that cannot be compiled, and, as such, be run by PHP. Scripts are linted against various versions of PHP.

This is usually undesirable, as all code must compile before being executed. It may be that such files are not compilable because they are not yet ready for an upcoming PHP version.

```
<?php

// Can't compile this : Print only accepts one argument
print $a, $b, $c;

?>
```

Code that is not compilable with older PHP versions means that the code is breaking backward compatibility : good or bad is project decision.

When the code is used as a template for PHP code generation, for example at installation time, it is recommended to use a distinct file extension, so as to distinguish them from actual PHP code.

Suggestions

- If this file is a template for PHP code, change the extension to something else than .php
- Fix the syntax so it works with various versions of PHP

Specs

Short name	Php/Incompilable
Rulesets	*Analyze*, *CE*
Exakt since	0.8.4
Php Version	All
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	High
ClearPHP	no-incompilable
Examples	*xataface*

13.2.778 Numeric Literal Separator

Integer and floats may be written with internal underscores. This way, it is possible to separate large number into smaller groups, and make them more readable.

Numeric Literal Separators were introduced in PHP 7.4 and are not backward compatible.

```
<?php
$a = 1_000_000_000;    // A billion
$a = 1000000000;       // A billion too...

$b = 107_925_284.88; // 6 light minute to kilometers = 107925284.88 kilometers
$b = 107925284.88;   // Same as above
?>
```

See also PHP RFC: Numeric Literal Separator.

Suggestions

-

Specs

Short name	Php/IntegerSeparatorUsage
Rulesets	*CE*, *CompatibilityPHP73*
Exakt since	1.9.0
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.779 Wrong Parameter Type

The expected parameter is not of the correct type. Check PHP documentation to know which is the right format to be used.

```
<?php

// substr() shouldn't work on integers.
// the first argument is first converted to string, and it is 123456.
echo substr(123456, 0, 4); // display 1234

// substr() shouldn't work on boolean
// the first argument is first converted to string, and it is 1, and not t
echo substr(true, 0, 1); // displays 1

// substr() works correctly on strings.
echo substr(123456, 0, 4);

?>
```

Specs

Short name	Php/InternalParameterType
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Zencart*

13.2.780 Is_A() With String

When using is_a() with a string as first argument, the third argument is compulsory.

```
<?php

// is_a() works with string as first argument, when the third argument is 'true'
if (is_s('A', 'B', true)) {}

// is_a() works with object as first argument
if (is_s(new A, 'A')) {}
?>
```

See also is_a().

Suggestions

- Add the third argument, and set it to true
- Use an object as a first argument

Specs

Short name	Php/IsAWithString
Rulesets	*Analyze*, *CI-checks*, *Rector*
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.781 Manipulates INF

This code handles INF situations. INF represents the infinity, when used in a float context. It happens when a calculation returns a number that is much larger than the maximum allowed float (not integer), or a number that is not a Division by 0.

```
<?php

// pow returns INF, as it is equivalent to 1 / 0 ^ 2
$a = pow(0,-2); //

// exp returns an actual value, but won't be able to represent it as a float
$a = exp(PHP_INT_MAX);

// 0 ^ -1 is like 1 / 0 but returns INF.
$a = pow(0, -1);

var_dump(is_infinite($a));

// This yields a Division by zero exception
$a = 1 / 0;

?>
```

See also Math predefined constants.

Specs

Short name	Php/IsINF
Rulesets	*CE*
Exakt since	0.10.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.782 Manipulates NaN

This code handles `Not-a-Number` situations. `Not-a-Number`, also called `NaN`, happens when a calculation can't return an actual float.

```
<?php

// acos returns a float, unless it is not possible.
$a = acos(8);

var_dump(is_nan($a));

?>
```

See also Floats.

Specs

Short name	Php/IsNAN
Rulesets	*CE*
Exakt since	0.10.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.783 Use === null

It is faster to use === null instead of is_null().

```
<?php

// Operator === is fast
if ($a === null) {

}

// Function call is slow
if (is_null($a)) {

}


?>
```

Suggestions

- Use === comparison

Specs

Short name	Php/IsnullVsEqualNull
Rulesets	*Analyze*, *CI-checks*, *php-cs-fixable*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	avoid-those-slow-functions

13.2.784 Isset Multiple Arguments

isset() may be used with multiple arguments and acts as a AND.

```
<?php

// isset without and
if (isset($a, $b, $c)) {
    // doSomething()
}

// isset with and
if (isset($a) && isset($b) && isset($c)) {
    // doSomething()
}

?>
```

See also Isset <http://www.php.net/‘isset>‘_.

Suggestions

- Merge all isset() calls into one

Specs

Short name	Php/IssetMultipleArgs
Rulesets	*Suggestions*, *php-cs-fixable*
Exakt since	0.12.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*ThinkPHP*, *LiveZilla*

13.2.785 Labels

List of all labels used in the code.

```
<?php

// A is label.
goto A:

A:

// A label may be used by several gotos.
goto A:

?>
```

Specs

Short name	Php/Labelnames
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.786 Logical Operators Favorite

PHP has two sets of logical operators : letters (and, or, xor) and chars (&&, ||, ^).

The analyzed code has less than 10% of one of the two sets : for consistency reasons, it is recommended to make them all the same.

Warning : the two sets of operators have different precedence levels. Using and or && is not exactly the same, especially and not only, when assigning the results to a variable.

```
<?php

$a1 = $b and $c;
$a1 = $b and $c;
$a1 = $b and $c;
$a1 = $b or $c;
$a1 = $b OR $c;
$a1 = $b and $c;
$a1 = $b and $c;
$a1 = $b and $c;
$a1 = $b or $c;
$a1 = $b OR $c;
$a1 = $b ^ $c;

?>
```

Using and or && are also the target of other analysis.

See also Logical Operators and Operators Precedence.

Suggestions

- Pick a favorite, and enforce it

Specs

Short name	Php/LetterCharsLogicalFavorite
Rulesets	*Top10*
Exakt since	0.12.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.787 List Short Syntax

Usage of short syntax version of list().

```
<?php

// PHP 7.1 short list syntax
// PHP 7.1 may also use key => value structures with list
[$a, $b, $c] = ['2', 3, '4'];

// PHP 7.0 list syntax
list($a, $b, $c) = ['2', 3, '4'];

?>
```

Specs

Short name	Php/ListShortSyntax
Rulesets	*CE*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*
Exakt since	0.8.4
Php Version	7.1+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.788 List With Appends

List() behavior has changed in PHP 7.0 and it has impact on the indexing when list is used with the [] operator.

```
<?php

$x = array();
list($x[], $x[], $x[]) = [1, 2, 3];

print_r($x);

?>
```

In PHP 7.0, results are ::

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
)
```

In PHP 5.6, results are ::

```
Array
(
    [0] => 3
    [1] => 2
    [2] => 1
)
```

Suggestions

- Refactor code to avoid using append in a list() call

Specs

Short name	Php/ListWithAppends
Rulesets	*CompatibilityPHP70*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.789 List With Keys

Setting keys when using list() is a PHP 7.1 feature.

```
<?php

// PHP 7.1 and later only
list('a' => $a, 'b' => $b) = ['b' => 1, 'c' => 2, 'a' => 3];

?>
```

Specs

Short name	Php/ListWithKeys
Rulesets	*CE*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*
Exakt since	0.8.4
Php Version	7.1+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.790 List With Reference

Support for references in list calls is not backward compatible with older versions of PHP. The support was introduced in PHP 7.3.

```
<?php

$array = [1,2,3];

[$c, &$d, $e] = $a;

$d++;
$c++;
print_r($array);
/*
displays
Array
(
    [0] => 1  // Not a reference to $c, unchanged
    [1] => 3  // Reference from $d
    [2] => 3
)
*/
?>
```

See also list() Reference Assignment.

Suggestions

- Avoid using references in list for backward compatibility

Specs

Short name	Php/ListWithReference
Rule-sets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*
Exakt since	1.1.6
Php Version	7.3+
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.791 Logical Should Use Symbolic Operators

Logical operators come in two flavors : and / &&, || / or, ^ / xor. However, they are not exchangeable, as && and and have different precedence.

```php
<?php

// Avoid lettered operator, as they have lower priority than expected
$a = $b and $c;
// $a === 3 because equivalent to ($a = $b) and $c;

// safe way to write the above :
$a = ($b and $c);

$a = $b && $c;
// $a === 1

?>
```

It is recommended to use the symbol operators, rather than the letter ones.

See also Logical Operators.

Suggestions

- Change the letter operators to the symbol one : and => &&, or => ||, xor => ^. Review the new expressions as processing order may have changed.
- Add parenthesis to make sure that the order is the expected one

Specs

Short name	Php/LogicalInLetters
Rulesets	*Analyze*, *CI-checks*, *Suggestions*, *Top10*, *php-cs-fixable*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-letter-logical
Examples	*Cleverstyle*, *OpenConf*

13.2.792 Methodcall On New

It is possible to call a method right at object instantiation.

This syntax was added in PHP 5.4+. Before, this was not possible : the object had to be stored in a variable first.

```php
<?php

// Data is collected
$data = data_source();

// Data is saved, but won't be reused from this databaseRow object. It may be ignored.
$result = (new databaseRow($data))->save();

// The actual result of the save() is collected and tested.
if ($result !== true) {
```

(continues on next page)

(continued from previous page)

```
    processSaveError($data);
}

?>
```

This syntax is interesting when the object is not reused, and may be discarded

Specs

Short name	Php/MethodCallOnNew
Rulesets	*CompatibilityPHP53*
Exakt since	0.8.4
Php Version	5.4+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.793 PHP Bugfixes

This is the list of features, used in the code, that also received a bug fix in recent PHP versions.

Specs

Short name	Php/MiddleVersion
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.794 Missing __isset() Method

When using empty() on magic properties, the magic method __isset() must be implemented.

```
<?php

class foo {
    function __get($name) { return 'foo'; }
    // No __isset method
}

// Return TRUE, until __isset() exists
var_dump(
   empty((new foo)->bar);
);

?>
```

See also When empty is not empty.

Suggestions

- Implement __isset() method when using empty on magic properties

Specs

Short name	Php/MissingMagicIsset
Rulesets	*Analyze*, *ClassReview*
Exakt since	2.2.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.795 Possible Missing Subpattern

When capturing subpatterns are the last ones in a regex, PHP doesn't fill their spot in the resulting array. This leads to a possible missing index in the result array.

```
<?php

// displays a partial array, from 0 to 1
preg_match('/(a)(b)?/', 'adc', $r);
print_r($r);
/*
Array
(
    [0] => a
    [1] => a
)
*/

// displays a full array, from 0 to 2
preg_match('/(a)(b)?/', 'abc', $r);
print_r($r);

/*
Array
(
    [0] => ab
    [1] => a
    [2] => b
)
*/

// double 'b' when it is found
print preg_replace(',^a(b)?,', './\$1\$1', 'abc'); // prints ./abbc
print preg_replace(',^a(b)?,', './\$1\$1', 'adc'); // prints ./dc

?>
```

?>

The same applies to preg_replace() : the pattern may match the string, but no value is available is the corresponding sub-pattern.

In PHP 7.4, a new option was added : PREG_UNMATCHED_AS_NULL, which always provides a value for the subpatterns.

See also Bug #50887 preg_match , last optional sub-patterns ignored when empty and Bug #73948 Preg_match_all should return NULLs on trailing optional capture groups..

Suggestions

- Add an always capturing subpatterns after the last ?
- Move the ? inside the parenthesis, so the parenthesis is always on, but the content may be empty
- Add a test on the last index of the resulting array, to ensure it is available when needed
- Use the PREG_UNMATCHED_AS_NULL option (PHP 7.4+)

Specs

Short name	Php/MissingSubpattern
Rulesets	*Analyze*, *CI-checks*, *Top10*
Exakt since	1.6.1
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*phpMyAdmin*, *SPIP*

13.2.796 Multiple Declaration Of Strict_types

> At least two declare() commands are declaring *strict_types* in one file. Only one is sufficient, and should be the first expression in the file.

Indeed, any *strict_types* set to 1 will have the final word. Setting *strict_types* to 0 will not revert the configuration, wherever is this call made.

```
<?php
declare(strict_types=1);
declare(strict_types=1);

// rest of the code

?>
```

See also Declare.

Suggestions

- Just remove all but one of them.

Specs

Short name	Php/MultipleDeclareStrict
Rulesets	*Analyze*
Exakt since	2.1.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.797 Must Call Parent Constructor

Some PHP native classes require a call to parent::__construct() to be stable.

As of PHP 7.3, two classes currently need that call : SplTempFileObject and SplFileObject.

The error is only emitted if the class is instantiated, and a parent class is called.

```
<?php

class mySplFileObject extends \SplFileObject {
    public function __construct()     {
        // Forgottent call to parent::__construct()
    }
}

(new mySplFileObject())->passthru();
?>
```

See also Why, php? WHY???.

Suggestions

- Add a call to the parent's constructor
- Remove the extension of the parent class

Specs

Short name	Php/MustCallParentConstructor
Rulesets	*Analyze*
Exakt since	1.4.1
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.798 Nested Ternary Without Parenthesis

It is not allowed to nest ternary operator within itself, without parenthesis. This has been implemented in PHP 7.4.

The reason behind this feature is to keep the code expressive. See the Warning message for more explanations

```
<?php

$a ? 1 : ($b ? 2 : 3);

// Still valid, as not ambiguous
$a ? $b ? 1 : 2 : 3;

// Produces a warning
//Unparenthesized `a ? b : c ? d : e` is deprecated. Use either `(a ? b : c) ? d : e`␣
↪or `a ? b : (c ? d : e)`
$a ? 1 : $b ? 2 : 3;

?>
```

See also PHP RFC: Deprecate left-associative ternary operator.

Suggestions

- Add parenthesis to nested ternary calls

Specs

Short name	Php/NestedTernaryWithoutParenthesis
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.9.4
Php Version	7.4-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.799 ** For Exponent

The operator ** calculates exponents, also known as power.

Use it instead of the slower function pow(). This operator was introduced in PHP 5.6.

```
<?php
    $cube = pow(2, 3); // 8

    $cubeInPHP56 = 2 ** 3; // 8
?>
```

Be aware the the '-' operator has lower priority than the ** operator : this leads to the following confusing result.

```
<?php
    echo -3 ** 2;
    // displays -9, instead of 9
?>
```

This is due to the parser that processes separately – and the following number. Since ** has priority, the power operation happens first.

Being an operator, ** is faster than pow(). This is a microoptimisation.

See also Arithmetic Operators.

Suggestions

- Use the ** operator
- For powers of 2, use the bitshift operators
- For literal powers of 2, consider using the `0xFFFFFFFF` syntax.

Specs

Short name	Php/NewExponent
Rulesets	*Suggestions*, *php-cs-fixable*
Exakt since	0.8.4
Php Version	5.6+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high
Examples	*Traq*, *TeamPass*

13.2.800 No Class In Global

Avoid defining structures in Global namespace. Always prefer using a namespace. This will come handy later, either when publishing the code, or when importing a library, or even if PHP reclaims that name.

```
<?php

// Code prepared for later
namespace Foo {
    class Bar {}
}

// Code that may conflict with other names.
namespace {
    class Bar {}
}

?>
```

Suggestions

- Use a specific namespace for your classes

Specs

Short name	Php/NoClassInGlobal
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.10.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Dolphin*

13.2.801 No List With String

list() can't be used anymore to access particular offset in a string. This should be done with substr() or $string[$offset] syntax.

```
<?php

$x = 'abc';
list($a, $b, $c) = $x;

//list($a, $b, $c) = 'abc'; Never works

print $c;
// PHP 5.6- displays 'c'
// PHP 7.0+ displays nothing

?>
```

See also PHP 7.0 Backward incompatible changes : list() can no longer unpack string variables .

Suggestions

- Use str_split() to break a string into bytes
- Use substr() or $string[$offset] syntax to access specific bytes in the string

Specs

Short name	Php/NoListWithString
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	7.0-
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.802 No More Curly Arrays

Only use square brackets to access array elements. The usage of curly brackets for array access is deprecated since PHP 7.4.

```
<?php

$array = [1,2,3];

// always valid
echo $array[1];

// deprecated in PHP 7.4
echo $array{1};

?>
```

See also Deprecate curly brace syntax and Deprecate curly brace syntax for accessing array elements and string offsets.

Suggestions

- Always use square brackets to access particular index in an array

Specs

Short name	Php/NoMoreCurlyArrays
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.9.2
Php Version	8.0-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.803 No Reference For Static Property

Static properties used to behave independently when set to a reference value. This was fixed in PHP 7.3.

According to the PHP 7.3 changelog : `In PHP, `static <https://www.php.net/manual/en/language.oop5.static.php>`_ properties are shared between inheriting classes, unless the `static <https://www.php.net/manual/en/language.oop5.static.php>`_ property is explicitly overridden in a child class. However, due to an implementation artifact it was possible to separate the `static <https://www.php.net/manual/en/language.oop5.static.php>`_ properties by assigning a reference. This loophole has been fixed..

```
<?php

        class Test {
            public static $x = 0;
        }
        class Test2 extends Test { }
```

(continues on next page)

(continued from previous page)

```
        Test2::$x = &$x;
        $x = 1;

        var_dump(Test::$x, Test2::$x);
        // Previously: int(0), int(1)
        // Now: int(1), int(1)

?>
```

See also PHP 7.3 UPGRADE NOTES.

Specs

Short name	Php/NoReferenceForStaticProperty
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*
Exakt since	1.4.9
Php Version	7.3-
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.804 No Reference For Ternary

> The ternary operator and the null coalescing operator are both expressions that only return values, and not a variable.

This means that any provided reference will be turned into its value. While this is usually invisible, it will raise a warning when a reference is expected. This is the case with methods returning a reference.

A PHP notice is generated when using a ternary operator or the null coalesce operator : `Only variable references should be returned by reference`. The notice is also emitted when returning objects.

This applies to methods, functions and closures.

```
<?php

// This works
function &foo($a, $b) {
    if ($a === 1) {
        return $b;
    } else {
        return $a;
    }
}

// This raises a warning, as the operator returns a value
```

(continues on next page)

(continued from previous page)

```
function &foo($a, $b) { return $a === 1 ? $b : $a; }

?>
```

See also Null Coalescing Operator, Ternary Operator.

Suggestions

- Drop the reference at assignation time
- Drop the reference in the argument definition
- Drop the reference in the function return definition

Specs

Short name	Php/NoReferenceForTernary
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.0.8
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
Examples	*phpadsnew*

13.2.805 No Return For Generator

Return is not allowed in generator. In PHP versions older than 5.6 and older, they yield a fatal Error.

```
<?php

function generatorWithReturn() {
    yield 1;
    return 2;
}

?>
```

See also Generators overview.

Specs

Short name	Php/NoReturnForGenerator
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	1.4.9
Php Version	7.0+
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.806 No String With Append

PHP 7 doesn't allow the usage of [] with strings. [] is an array-only operator.

```
<?php

$string = 'abc';

// Not possible in PHP 7
$string[] = 'd';

?>
```

This was possible in PHP 5, but is now forbidden in PHP 7.

Suggestions

- Use the concatenation operator `.` to append strings.
- Use the concatenation short assignement `.=` to append strings.

Specs

Short name	Php/NoStringWithAppend
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	7.0+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.807 No Substr Minus One

Negative index were introduced in PHP 7.1. This syntax is not compatible with PHP 7.0 and older.

```
<?php
$string = 'abc';

echo $string[-1]; // c

echo $string[1]; // a

?>
```

See also Generalize support of negative string offsets.

Suggestions

- Use the -1 index in a string, instead of a call to substr()

Specs

Short name	Php/NoSubstrMinusOne
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*
Exakt since	0.12.5
Php Version	7.1+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.808 Not A Scalar Type

`int` is the actual PHP scalar type, not `integer`.

PHP 7 introduced several scalar types, in particular `int`, `bool` and `float`. Those three types are easily mistaken with `integer`, `boolean`, `real` and `double`.

Unless those classes actually exists, PHP emits some strange error messages.

```
<?php

// This expects a scalar of type 'integer'
function foo(int $i) {}

// This expects a object of class 'integer'
function abr(integer $i) {}

?>
```

Thanks to `Benoit Viguier` for the original idea for this analysis.

See also Type declarations.

Suggestions

- Do not use `int` as a class name, an interface name or a trait name.

Specs

Short name	Php/NotScalarType
Rulesets	*Typechecks*
Exakt since	1.0.7
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	Very high

13.2.809 Old Style __autoload()

Avoid __autoload(), only use spl_register_autoload().

__autoload() is deprecated since PHP 7.2 and possibly removed in later versions. spl_register_autoload() was introduced in PHP 5.1.0.

__autoload() may only be declared once, and cannot be modified later. This creates potential conflicts between libraries that try to set up their own autoloading schema.

On the other hand, spl_register_autoload() allows registering and de-registering multiple autoloading functions or methods.

```
<?php

// Modern autoloading.
function myAutoload($class){}
spl_register_autoload('myAutoload');

// Old style autoloading.
function __autoload($class){}

?>
```

Do not use the old __autoload() function, but rather the new spl_register_autoload() function.

See also Autoloading Classe.

Suggestions

- Move to spl_register_autoload()
- Remove usage of the old __autoload() function
- Modernize usage of old libraries

Specs

Short name	Php/oldAutoloadUsage
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	use-smart-autoload
Examples	*Piwigo*

13.2.810 Only Container For Reference

When a PHP function requires an argument to be a reference, it cannot be called with a literal value.

The call must be made with a variable, or any assimilated data container : array, property or static property.

```
<?php

// This is not possible
preg_match('/a/', $string, []);

// This is working
preg_match('/a/', $string, $r);

?>
```

See also Passing arguments by reference.

Suggestions

- Put the literal value in a variable before calling the function.
- Omit the arguments, when it won't be used.

Specs

Short name	Php/OnlyVariableForReference
Rulesets	*Analyze*, *LintButWontExec*
Exakt since	2.2.0
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.811 PHP Overridden Function

It is possible to declare and use PHP native function in a namespace.

Within the declaration namespace, it is easy to confuse the local version and the global version, unless the function has been prefixed with \.

```
<?php

namespace A {
    use function A\dirname as split;

    function dirname($a, $b) { return __FUNCTION__; }

    echo dirname('/a/b/c');
    echo split('a', 'b');

    echo \dirname('/a/b/c');
}

?>
```

When a piece of code use overridden function, any newcomer may be confused by the usage of classic PHP native function in surprising situations.

It is recommended to avoid redeclare PHP native function in namespaces.

Suggestions

- Change the name of the function, in its declaration and usage.

Specs

Short name	Php/OveriddenFunction
Rulesets	*CE*
Exakt since	1.7.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.812 Parenthesis As Parameter

Using parenthesis around parameters used to silent some internal check. This is not the case anymore in PHP 7, and should be fixed by removing the parenthesis and making the value a real reference.

```
<?php

// PHP 7 sees through parenthesis
$d = foo(1, 2, $c);

// Avoid parenthesis in arguments
$d = foo(1, 2, ($c));

?>
```

Suggestions

- Remove the parenthesis when they are only encapsulating an argument

Specs

Short name	Php/ParenthesisAsParameter
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.813 Use password_hash()

password_hash() and password_check() are a better choice to replace the use of crypt() to check password.

PHP 5.5 introduced these functions.

```
<?php

$password = 'rasmuslerdorf';
$hash = '\$2y\$10$YCFsG6elYca568hBi2pZ0.3LDL5wjgxct1N8w/oLR/jfHsiQwCqTS';

// The cost parameter can change over time as hardware improves
$options = array('cost' => 11);

// Verify stored hash against plain-text password
if (password_verify($password, $hash)) {
    // Check if a newer hashing algorithm is available
    // or the cost has changed
    if (password_needs_rehash($hash, PASSWORD_DEFAULT, $options)) {
        // If so, create a new hash, and replace the old one
        $newHash = password_hash($password, PASSWORD_DEFAULT, $options);
    }

    // Log user in
}
?>
```

See also Password hashing.

Specs

Short name	Php/Password55
Rulesets	*CompatibilityPHP55*
Exakt since	0.8.4
Php Version	5.5+
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.814 Pathinfo() Returns May Vary

pathinfo() function returns an array whose content may vary. It is recommended to collect the values after check, rather than directly.

```
<?php

$file = '/a/b/.c';
//$extension may be missing, leading to empty $filename and filename in $extension
list( $dirname, $basename, $extension, $filename ) = array_values( pathinfo($file) );

//Use PHP 7.1 list() syntax to assign correctly the values, and skip array_values()
//This emits a warning in case of missing index
['dirname'   => $dirname,
 'basename'  => $basename,
 'extension' => $extension,
 'filename'  => $filename ] = pathinfo($file);

//This works without warning
$details = pathinfo($file);
```

(continues on next page)

(continued from previous page)

```
$dirname   = $details['dirname'] ?? getpwd();
$basename  = $details['basename'] ?? '';
$extension = $details['extension'] ?? '';
$filename  = $details['filename'] ?? '';

?>
```

The same applies to parse_url(), which returns an array with various index.

Suggestions

- Add a check on the return value of pathinfo() before using it.

Specs

Short name	Php/PathinfoReturns
Rulesets	*Analyze*
Exakt since	0.12.11
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*NextCloud*

13.2.815 Pear Usage

Pear Usage : list of Pear packages in use.

```
<?php
    require_once('MDB2.php');
    $dsn = 'mysql://user:pass@host';
    $mdb2 = &MDB2::factory($dsn);
    $mdb2->setFetchMode(MDB2_FETCHMODE_ASSOC);
?>
```

See also PEAR.

Specs

Short name	Php/PearUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.816 New Functions In PHP 5.4

PHP introduced new functions in PHP 5.4. If there are defined functions with such names, there will be a conflict when upgrading. It is advised to change those functions' name.

Specs

Short name	Php/Php54NewFunctions
Rulesets	*CompatibilityPHP53*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.817 Functions Removed In PHP 5.4

Those functions were removed in PHP 5.4.

```
<?php

// Deprecated as of PHP 5.4.0
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$db_list = mysql_list_dbs($link);

while ($row = mysql_fetch_object($db_list)) {
     echo $row->Database . "\n";
}

?>
```

See also Deprecated features in PHP 5.4.x.

Specs

Short name	Php/Php54RemovedFunctions
Rulesets	*CompatibilityPHP54*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.818 New Functions In PHP 5.5

PHP introduced new functions in PHP 5.5. If you have already defined functions with such names, you will get a conflict when trying to upgrade. It is advised to change those functions' name.

Specs

Short name	Php/Php55NewFunctions
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.819 Functions Removed In PHP 5.5

Those functions were removed in PHP 5.5.

- php_logo_guid()
- php_egg_logo_guid()
- php_real_logo_guid()
- zend_logo_guid()
- mcrypt_cbc()
- mcrypt_cfb()
- mcrypt_ecb()
- mcrypt_ofb()

```
<?php

echo '<img src="' . $_SERVER['PHP_SELF'] .
     '?=' . php_logo_guid() . '" alt="PHP Logo !" />';

?>
```

See also Deprecated features in PHP 5.5.x.

Suggestions

- Stop using those functions

Specs

Short name	Php/Php55RemovedFunctions
Rulesets	*CompatibilityPHP55*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.820 New Functions In PHP 5.6

PHP introduced new functions in PHP 5.6. If you have already defined functions with such names, you will get a conflict when trying to upgrade. It is advised to change those functions' name.

Specs

Short name	Php/Php56NewFunctions
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.821 PHP 7.0 New Classes

Those classes are now declared natively in PHP 7.0 and should not be declared in custom code.

There are 8 new classes :

- `Error`
- `ParseError`
- `TypeError`
- `ArithmeticError`
- `DivisionByZeroError`
- `ClosedGeneratorException`
- `ReflectionGenerator`
- `ReflectionType`
- `AssertionError`

```
<?php

namespace {
    // Global namespace
    class Error {
        // Move to a namespace
        // or, remove this class
    }
}

namespace B {
    class Error {
        // This is OK : in a namespace
    }
}

?>
```

See also New Classes and Interfaces.

Specs

Short name	Php/Php70NewClasses
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.822 New Functions In PHP 7.0

The following functions are now native functions in PHP 7.0. It is advised to change them before moving to this new version.

- get_resources()
- gc_mem_caches()
- preg_replace_callback_array()
- posix_setrlimit()
- random_bytes()
- random_int()
- intdiv()
- error_clear_last()

Specs

Short name	Php/Php70NewFunctions
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.823 PHP 7.0 New Interfaces

The following interfaces are introduced in PHP 7.0. They shouldn't be defined in custom code.

Specs

Short name	Php/Php70NewInterfaces
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.824 PHP 7.0 Removed Directives

List of directives that are removed in PHP 7.0.

Specs

Short name	Php/Php70RemovedDirective
Rulesets	*CompatibilityPHP70*, *CompatibilityPHP71*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.825 PHP 7.0 Removed Functions

The following PHP native functions were removed in PHP 7.0.

- ereg()
- ereg_replace()
- eregi()
- eregi_replace()
- split()
- spliti()
- sql_regcase()
- magic_quotes_runtime()
- set_magic_quotes_runtime()
- call_user_method()
- call_user_method_array()
- set_socket_blocking()
- mcrypt_ecb()
- mcrypt_cbc()

- mcrypt_cfb()
- mcrypt_ofb()
- datefmt_set_timezone_id()
- imagepsbbox()
- imagepsencodefont()
- imagepsextendfont()
- imagepsfreefont()
- imagepsloadfont()
- imagepsslantfont()
- imagepstext()

This analysis skips redefined PHP functions : when a replacement for a removed PHP function was created, with condition on the PHP version, then its usage is considered valid.

See also PHP 7.0 Removed Functions.

Suggestions

- Replace the old functions with modern functions
- Remove the usage of the old functions
- Create an alternative function by wiring the old name to a new feature

Specs

Short name	Php/Php70RemovedFunctions
Rulesets	*CompatibilityPHP70*, *CompatibilityPHP71*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.826 PHP 7.0 Scalar Typehints

New scalar typehints were introduced : `bool`, `int`, `float`, `string`.

They cannot be used before PHP 7.0, and will be confused with classes or interfaces.

```
<?php

function foo(string $name) {
    print Hello $name;
}

foo(Damien);
// display 'Hello Damien'
```

(continues on next page)

(continued from previous page)

```
foo(33);
// displays an error

?>
```

See also Scalar type declarations, and PHP 7 SCALAR TYPE DECLARATIONS.

Specs

Short name	Php/PHP70scalartypehints
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	1.3.5
Php Version	7.0+
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.827 PHP 7.1 Microseconds

PHP supports microseconds in `DateTime` class and date_create() function. This was introduced in PHP 7.1.

In previous PHP versions, those dates only used seconds, leading to lazy comparisons :

```
<?php

$now = date_create();
usleep(10);               // wait for 0.001 ms
var_dump($now == date_create());

?>
```

This code displays true in PHP 7.0 and older, (unless the code was run too close from the next second). In PHP 7.1, this is always false.

This is also true with `DateTime` :

```
<?php

$now = new DateTime();
usleep(10);               // wait for 0.001 ms
var_dump((new DateTime())->format('u') == $now->format('u'));

?>
```

This evolution impacts mostly exact comparisons (== and ===). Non-equality (!= and !==) will probably be always true, and should be reviewed.

See also Backward incompatible changes.

Specs

Short name	Php/Php71microseconds
Rulesets	*CompatibilityPHP71*
Exakt since	0.8.9
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.828 Php 7.1 New Class

New classes, introduced in PHP 7.1. If classes where created with the same name, in current code, they have to be moved in a namespace, or removed from code to migrate safely to PHP 7.1.

The new class is : ReflectionClassConstant. The other class is 'Void' : this is forbidden as a class name, as Void is used for return type hint.

```
<?php

class ReflectionClassConstant {
    // Move to a namespace, do not leave in global
    // or, remove this class
}

?>
```

Specs

Short name	Php/Php71NewClasses
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.829 New Functions In PHP 7.1

The following functions are now native functions in PHP 7.1. It is advised to change them before moving to this new version.

- curl_share_strerror()
- curl_multi_errno()

- curl_share_errno()
- mb_ord()
- mb_chr()
- mb_scrub()
- is_iterable()

Specs

Short name	Php/Php71NewFunctions
Rulesets	*CompatibilityPHP71*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.830 PHP 7.1 Removed Directives

List of directives that are removed in PHP 7.1.

Specs

Short name	Php/Php71RemovedDirective
Rulesets	*CompatibilityPHP71*
Exakt since	0.8.4
Php Version	7.1+
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.831 PHP 7.1 Scalar Typehints

A new scalar typehint was introduced : iterable.

It can't be used before PHP 7.1, and will be confused with classes or interfaces.

```
<?php

function foo(iterable $iterable) {
    foreach ($iterable as $value) {
        echo $value.PHP_EOL;
    }
}

foo(range(1,20));
// works with array
```

(continues on next page)

(continued from previous page)

```
foo(new ArrayIterator([1, 2, 3]));
// works with an iterator

foo((function () { yield 1; })() );
// works with a generator

?>
```

See also iterable pseudo-type, and The iterable Pseudo-Type.

Specs

Short name	Php/PHP71scalartypehints
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*
Exakt since	1.3.5
Php Version	7.1+
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.832 PHP 7.2 Deprecations

Several functions are deprecated in PHP 7.2.

- parse_str() with no second argument
- assert() on strings
- Usage of gmp_random(), create_function(), each()
- Usage of (unset)
- Usage of `$php_errormsg`
- directive `mbstring.func_overload` (not supported yet)

Deprecated functions and extensions are reported in a separate analysis.

See also Deprecations for PHP 7.2.

Suggestions

- Remove the deprecated functions, and replace them with a new feature
- Use a replacement function to emulate this old behavior

Specs

Short name	Php/Php72Deprecation
Rulesets	*CompatibilityPHP72*
Exakt since	0.9.9
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.833 Php 7.2 New Class

New classes, introduced in PHP 7.2. If classes where created with the same name, in current code, they have to be moved in a namespace, or removed from code to migrate safely to PHP 7.2.

The new class is : HashContext.

```
<?php

namespace {
    // Global namespace
    class HashContext {
        // Move to a namespace
        // or, remove this class
    }
}

namespace B {
    class HashContext {
        // This is OK : in a namespace
    }
}

?>
```

Specs

Short name	Php/Php72NewClasses
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*
Exakt since	1.0.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.834 New Constants In PHP 7.2

The following constants are now native in PHP 7.2. It is advised to avoid using such names for constant before moving to this new version.

- PHP_OS_FAMILY
- PHP_FLOAT_DIG
- PHP_FLOAT_EPSILON
- PHP_FLOAT_MAX
- PHP_FLOAT_MIN
- SQLITE3_DETERMINISTIC
- CURLSSLOPT_NO_REVOKE
- CURLOPT_DEFAULT_PROTOCOL
- CURLOPT_STREAM_WEIGHT
- CURLMOPT_PUSHFUNCTION
- CURL_PUSH_OK
- CURL_PUSH_DENY
- CURL_HTTP_VERSION_2TLS
- CURLOPT_TFTP_NO_OPTIONS
- CURL_HTTP_VERSION_2_PRIOR_KNOWLEDGE
- CURLOPT_CONNECT_TO
- CURLOPT_TCP_FASTOPEN
- DNS_CAA

See also New global constants in 7.2.

Specs

Short name	Php/Php72NewConstants
Rulesets	*CompatibilityPHP72*
Exakt since	0.10.7
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.835 New Functions In PHP 7.2

The following functions are now native functions in PHP 7.2. It is advised to change custom functions that are currently created, and using those names, before moving to this new version.

- mb_ord()
- mb_chr()

- mb_scrub()
- stream_isatty()
- proc_nice() (Windows only)

Suggestions

- Move custom functions with the same name to a new namespace
- Change the name of any custom functions with the same name
- Add a condition to the functions definition to avoid conflict

Specs

Short name	Php/Php72NewFunctions
Rulesets	*CompatibilityPHP72*
Exakt since	0.10.7
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.836 PHP 7.2 Object Keyword

'object' is a PHP keyword. It can't be used for class, interface or trait name.

This is the case since PHP 7.2.

```
<?php

// Valid until PHP 7.2
class object {}

// Altough it is really weird anyway...

?>
```

See also List of Keywords.

Specs

Short name	Php/Php72ObjectKeyword
Rulesets	*CompatibilityPHP72*
Exakt since	0.12.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.837 PHP 7.2 Removed Functions

The following PHP native functions were removed in PHP 7.2.

- png2wbmp()
- jpeg2wbmp()
- create_function()
- gmp_random()
- each()

This analysis skips redefined PHP functions : when a replacement for a removed PHP function was created, with condition on the PHP version, then its usage is considered valid.

See also Deprecated features in PHP 7.2.x.

Specs

Short name	Php/Php72RemovedFunctions
Rulesets	*CompatibilityPHP72*
Exakt since	0.9.9
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.838 PHP 7.2 Scalar Typehints

A new scalar typehint was introduced : object.

It can't be used before PHP 7.2, and will be confused with classes or interfaces.

```
<?php

function test(object $obj) : object
{
    return new SplQueue();
}

test(new StdClass());

?>
```

See also New object type, and PHP 7.2 and Object Typehint.

Specs

Short name	Php/PHP72scalartypehints
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*
Exakt since	1.3.5
Php Version	7.2+
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.839 PHP 7.3 Last Empty Argument

PHP allows the last element of any functioncall to be empty. The argument is then not send.

This was introduced in PHP 7.3, and is not backward compatible.

The last empty line is easier on the VCS, allowing clearer text diffs.

```
<?php

function foo($a, $b) {
    print_r(func_get_args());
}


foo(1,
    2,
    );

foo(1);


?>
```

See also Allow a trailing comma in function calls and Trailing commas.

Specs

Short name	Php/PHP73LastEmptyArgument
Rule-sets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*
Exakt since	1.1.7
Php Version	7.3+
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.840 New Functions In PHP 7.3

New functions are added to new PHP version.

The following functions are now native functions in PHP 7.3. It is compulsory to rename any custom function that was created in older versions. One alternative is to move the function to a custom namespace, and update the `use` list at the beginning of the script.

- net_get_interfaces
- gmp_binomial
- gmp_lcm
- gmp_perfect_power
- gmp_kronecker
- openssl_pkey_derive
- is_countable
- ldap_exop_refresh

Note : At the moment of writing, all links to the manual are not working.

Specs

Short name	Php/Php73NewFunctions
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*
Exakt since	0.10.7
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.841 PHP 7.3 Removed Functions

The following PHP native functions were removed in PHP 7.3.

- image2wbmp()

This analysis skips redefined PHP functions : when a replacement for a removed PHP function was created, with condition on the PHP version, then its usage is considered valid.

See also PHP 7.3 Removed Functions.

Specs

Short name	Php/Php73RemovedFunctions
Rulesets	*CompatibilityPHP73*
Exakt since	1.4.0
Php Version	All
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	High

13.2.842 PHP 7.4 Constant Deprecation

One constant is deprecated in PHP 7.4.

- CURLPIPE_HTTP1

See also Deprecations for PHP 7.2.

Suggestions

- Use CURLPIPE_MULTIPLEX or CURLPIPE_NOTHING

Specs

Short name	Php/Php74Deprecation
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.9.3
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.843 mb_strrpos() Third Argument

Passing the encoding as 3rd parameter to mb_strrpos() is deprecated. Instead pass a 0 offset, and encoding as 4th parameter.

```
<?php

// Finds the position of the last occurrence of of a string in a string, starting at␣
↪position 10
$extract = mb_strrpos($haystack, $needle, 10, 'utf8');

// This is the old behavior. Here, the offset will be 0, by default
$extract = mb_strrpos($haystack, $needle, 'utf8');
?>
```

See also mb_strrpos().

Suggestions

-

Specs

Short name	Php/Php74mbstrrpos3rdArg
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.8.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.844 Php 7.4 New Class

New classes, introduced in PHP 7.4. If classes where created with the same name, in current code, they have to be moved in a namespace, or removed from code to migrate safely to PHP 7.4.

The new classes are :

- `ReflectionReference`
- `WeakReference`

```
<?php

namespace {
    // Global namespace
    class WeakReference {
        // Move to a namespace
        // or, remove this class
    }
}

namespace B {
    class WeakReference {
        // This is OK : in a namespace
    }
}

?>
```

Suggestions

- Move the current classes with the same names into a distinct domain name

Specs

Short name	Php/Php74NewClasses
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.0.4
Php Version	7.4-
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.845 New Constants In PHP 7.4

The following constants are now native in PHP 7.4. It is advised to avoid using such names for constant before moving to this new version.

- MB_ONIGURUMA_VERSION
- SO_LABEL
- SO_PEERLABEL
- SO_LISTENQLIMIT
- SO_LISTENQLEN
- SO_USER_COOKIE
- PHP_WINDOWS_EVENT_CTRL_C
- PHP_WINDOWS_EVENT_CTRL_BREAK
- TIDY_TAG_ARTICLE
- TIDY_TAG_ASIDE

- TIDY_TAG_AUDIO
- TIDY_TAG_BDI
- TIDY_TAG_CANVAS
- TIDY_TAG_COMMAND
- TIDY_TAG_DATALIST
- TIDY_TAG_DETAILS
- TIDY_TAG_DIALOG
- TIDY_TAG_FIGCAPTION
- TIDY_TAG_FIGURE
- TIDY_TAG_FOOTER
- TIDY_TAG_HEADER
- TIDY_TAG_HGROUP
- TIDY_TAG_MAIN
- TIDY_TAG_MARK
- TIDY_TAG_MENUITEM
- TIDY_TAG_METER
- TIDY_TAG_NAV
- TIDY_TAG_OUTPUT
- TIDY_TAG_PROGRESS
- TIDY_TAG_SECTION
- TIDY_TAG_SOURCE
- TIDY_TAG_SUMMARY
- TIDY_TAG_TEMPLATE
- TIDY_TAG_TIME
- TIDY_TAG_TRACK
- TIDY_TAG_VIDEO
- STREAM_CRYPTO_METHOD_TLSv1_3_CLIENT
- STREAM_CRYPTO_METHOD_TLSv1_3_SERVER
- STREAM_CRYPTO_PROTO_TLSv1_3
- T_COALESCE_EQUAL
- T_FN

See also New global constants in 7.4.

Suggestions

- Move the constants to a new namespace
- Remove the old constants
- Rename the old constants

Specs

Short name	Php/Php74NewConstants
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.846 PHP 74 New Directives

List of directives that are new in PHP 7.4.

- `zend.exception_ignore_args` : From the php.ini : `Allows to include or exclude arguments from stack traces generated for exceptions. Default: Off`
- `opcache.preload_user`

See RFC Preload.

Suggestions

- Do not use those directives with PHP before version 7.4

Specs

Short name	Php/Php74NewDirective
Rulesets	*CompatibilityPHP73*
Exakt since	1.9.4
Php Version	7.4-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.847 New Functions In PHP 7.4

New functions are added to new PHP version.

The following functions are now native functions in PHP 7.3. It is compulsory to rename any custom function that was created in older versions. One alternative is to move the function to a custom namespace, and update the `use` list at the beginning of the script.

- mb_str_split
- password_algos

Note : At the moment of writing, all links to the manual are not working.

Specs

Short name	Php/Php74NewFunctions
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.8.0
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.848 PHP 7.4 Removed Directives

List of directives that are removed in PHP 7.4.

- allow_url_include

See Deprecation allow_url_include.

Suggestions

- Stop using this directive

Specs

Short name	Php/Php74RemovedDirective
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.9.3
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.849 PHP 7.4 Removed Functions

The following PHP native functions were deprecated in PHP 7.4.

- hebrevc()
- convert_cyr_string()
- ezmlm_hash()
- money_format()
- restore_include_path()
- get_magic_quotes_gpc()

- get_magic_quotes_runtime()

This analysis skips redefined PHP functions : when a replacement for a removed PHP function was created, with condition on the PHP version, then its usage is considered valid.

See also PHP 7.4 Removed Functions and PHP 7.4 Deprecations : Introduction.

Suggestions

-

Specs

Short name	Php/Php74RemovedFunctions
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.9.0
Php Version	All
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.850 PHP 7.4 Reserved Keyword

`fn` is a new PHP keyword. In PHP 7.4, it is used to build the arrow functions. When used at an illegal position, `fn` generates a Fatal error at compile time.

As a key word, `fn` is not allowed as constant name, function name, class name or inside namespaces.

```
<?php

// PHP 7.4 usage of fn
function array_values_from_keys($arr, $keys) {
    return array_map(fn($x) => $arr[$x], $keys);
}

// PHP 7.3 usage of fn
const fn = 1;

function fn() {}

class x {
    // This is valid in PHP 7.3 and 7.4
    function fn() {}
}

?>
```

`fn` is fine for method names. It may also be used for constants with define(), and constant() but it is not recommended.

See also PHP RFC: Arrow Functions.

Suggestions

-

Specs

Short name	Php/Php74ReservedKeyword
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.9.2
Php Version	7.4-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.851 Php7 Relaxed Keyword

Most of the traditional PHP keywords may be used inside classes, trait or interfaces.

```
<?php

// Compatible with PHP 7.0 +
class foo {
    // as is a PHP 5 keyword
    public function as() {

    }
}

?>
```

This was not the case in PHP 5, and will yield parse errors.

See also Loosening Reserved Word Restrictions.

Specs

Short name	Php/Php7RelaxedKeyword
Rulesets	*CE*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	7.0+
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.852 PHP 80 Named Parameter Variadic

Named parameter with variadic have been renamed from 0 to 'parameter name' in PHP 8.0

```
<?php

function foo($a, ...$b) {
    print_r($b);
}

foo(3, 4);
```

(continues on next page)

(continued from previous page)

```
foo(3, b: 4);               // PHP 8 only
foo(...[2, b=> [3, 4]]); // PHP 8 only

?>
```

In PHP 7.0, with positional argument only, the content of $b is in an array, index 0. This is also true with PHP 8.0.

In PHP 8.0, with named arguments, the content of $b is in an array, index 'b';

Since the behavior of the variadic depends on the calling syntax (with or without named parameter), the receiving must ensure the correct reception, and handle both cases.

Suggestions

- Apply array_values() to the variadic arguments.

Specs

Short name	Php/Php80NamedParameterVariadic
Rulesets	*CE*, *CompatibilityPHP80*
Exakt since	2.2.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.853 New Functions In PHP 8.0

New functions are added to new PHP version.

The following functions are now native functions in PHP 7.3. It is compulsory to rename any custom function that was created in older versions. One alternative is to move the function to a custom namespace, and update the `use` list at the beginning of the script.

- str_contains
- fdiv
- preg_last_error_msg

Note : At the moment of writing, all links to the manual are not working.

Specs

Short name	Php/Php80NewFunctions
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	2.0.8
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.854 Php 8.0 Only TypeHints

Two scalar typehints are introduced in version 8. They are `false` and `null`. In PHP 7.0, both those values could not be used as a class or interface name, to avoid confusion with the actual booleans, nor `null` value.

`false` represents a false boolean, and nothing else. It is more restrictive than a boolean, which accepts true too. `null` is an alternative syntax to `?` : it allows the type to be `null`.

Both the above typehints are to be used in cunjunction with other types : they can't be used alone.

```
<?php

// function accepts an A object, or null.
function foo(A|null $x) {}

// same as above
function foo2(A|null $x) {}

// returns an object of class B, or false
function bar($x) : false|B {}

?>
```

See also PHP RFC: Union Types 2.0.

Suggestions

-

Specs

Short name	Php/Php80OnlyTypeHints
Rulesets	*CE*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP74*
Exakt since	2.0.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.855 PHP 8.0 Removed Constants

The following PHP native constants were removed in PHP 8.0.

- INTL_IDNA_VARIANT_2003 (See Deprecate and remove 'INTL_IDNA_VARIANT_2003 <https://wiki.php.net/rfc/deprecate-and-remove-intl_idna_variant_2003>'_)

Suggestions

- Remove usage of INTL_IDNA_VARIANT_2003 and use

Specs

Short name	Php/Php80RemovedConstant
Rulesets	*CE*, *CompatibilityPHP80*
Exakt since	1.6.8
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.856 PHP 8.0 Removed Directives

List of directives that are removed in PHP 8.0.

In PHP 8.0, *track_errors* was removed.

You can detect valid directives with ini_get(). This native function will return false, when the directive doesn't exist, while actual directive values will be returned as a string.

Suggestions

- Remove usage of *track_errors*.

Specs

Short name	Php/Php80RemovedDirective
Rulesets	*CE*, *CompatibilityPHP80*
Exakt since	2.1.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.857 PHP 8.0 Removed Functions

The following PHP native functions were removed in PHP 8.0.

- image2wbmp()
- png2wbmp()
- jpeg2wbmp()
- ldap_sort()
- hebrevc()
- convert_cyr_string()

- ezmlm_hash()
- money_format()
- get_magic_quotes_gpc()
- get_magic_quotes_gpc_runtime()
- create_function()
- each()
- read_exif_data()
- gmp_random()
- fgetss()
- restore_include_path()
- gzgetss()

Specs

Short name	Php/Php80RemovedFunctions
Rulesets	*CE*, *CompatibilityPHP80*
Exakt since	1.6.8
Php Version	8.0-
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.858 PHP Resources Turned Into Objects

> Multiple PHP native functions now return objects, not resources. Any check on those values with is_resource() is now going to fail.

The affected functions are the following :

- curl_init()
- curl_multi_init()
- curl_share_init()
- deflate_init()
- enchant_broker_init()
- enchant_broker_request_dict()
- enchant_broker_request_pwl_dict()
- inflate_init()
- msg_get_queue()
- openssl_csr_new()
- openssl_csr_sign()
- openssl_pkey_new()

- openssl_x509_read()
- sem_get()
- shm_attach()
- shmop_open()
- socket_accept()
- socket_addrinfo_bind()
- socket_addrinfo_connect()
- socket_create_listen()
- socket_create()
- socket_import_stream()
- socket_wsaprotocol_info_import()
- xml_parser_create_ns()
- xml_parser_create()

See also UPGRADING PHP 8.0.

Suggestions

- Change the condition from is_resource() to instanceof

Specs

Short name	Php/Php80RemovesResources
Rulesets	*CE*, *CompatibilityPHP80*
Exakt since	2.2.0
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.859 Union Typehint

Union typehints allows the specification of several typehint for the same argument or return value. This is a PHP 8.0 new feature.

Several typehints are specified at the same place as a single one. The different values are separated by a pipe character `|`, like for exceptions

```
<?php

// Example from the RFC https://wiki.php.net/rfc/union_types_v2
class Number {
    private int|float $number;

    public function setNumber(int|float $number): void {
```

(continues on next page)

(continued from previous page)

```
        $this->number = $number;
    }

    public function getNumber(): int|float {
        return $this->number;
    }
}
?>
```

Union types are not compatible with PHP 7 and older.

See also PHP RFC: Union Types 2.0.

Suggestions

-

Specs

Short name	Php/Php80UnionTypehint
Rulesets	*CE*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP74*
Exakt since	2.0.9
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.860 Php 8.0 Variable Syntax Tweaks

> Several variable syntaxes are added in version 8.0. They extends the PHP 7.0 syntax updates, and fix a number of edges cases.

In particular, `new``and ``instanceof` now support a way to inline the expression, rather than use a temporary variable.

Magic constants are now accessible with array notation, just like another constant. It is also possible to use method calls : although this is Syntacticly correct for PHP, this won't be executed, as the left operand is a string, and not an object.

```
<?php

 // array name is dynamically build
 echo foo$bar[0];
 // static method
 foo$bar::baz();
 // static property
```

(continues on next page)

(continued from previous page)

```
 foo$bar::$baz;

 // Syntactly correct, but not executable
 foo$bar->baz();

 // expressions with instanceof and new
    $object = new (class_.$name);
    $x instanceof (class_$name);

    // PHP 7.0 style
    $className = class_.$name;
    $object = new $className;

?>
```

See also PHP RFC: Variable Syntax Tweaks and scalar_objects in PHP.

Specs

Short name	Php/Php80VariableSyntax
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	2.0.8
Php Version	8.0+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.861 $php_errormsg Usage

$php_errormsg is removed since PHP 8.0. $php_errormsg tracks the last error message, with the directive *track_errors*. All was removed in PHP 8.0, and shall be replaced with error_get_last().

```
<?php

function foo() {
    global $php_errormsg;

    echo 'Last error: '.$php_errormsg;

    echo 'Also, last error: '.error_get_last();
}

?>
```

Suggestions

- Use error_get_last() instead.

Specs

Short name	Php/PhpErrorMsgUsage
Rulesets	*CE*, *CompatibilityPHP80*
Exakt since	2.1.8
Php Version	8.0-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.862 preg_match_all() Flag

preg_match_all() has an option to configure the structure of the results : it is either by capturing parenthesis (by default), or by result sets.

The second option is the most interesting when the following foreach() loop has to manipulate several captured strings at the same time. No need to use an index in the first array and use it in the other arrays.

```
<?php
$string = 'abababb';

// default behavior
preg_match_all('/(a)(b)/', $string, $r);
$found = '';
foreach($r[1] as $id => $s) {
    $found .= $s.$r[2][$id];
}

// better behavior
preg_match_all('/(a)(b)/', $string, $r, PREG_SET_ORDER);
$found = '';
foreach($r as $s) {
    $found .= $s[1].$s[2];
}

?>
```

The second syntax is easier to read and may be marginally faster to execute (preg_match_all() and foreach()).

Suggestions

- Use flags to adapt the results of preg_match_all() to your code, not the contrary.

Specs

Short name	Php/PregMatchAllFlag
Rulesets	*Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*FuelCMS*

13.2.863 Displays Text

Function calls that displays something to the output.

```
<?php

// Displays de the content of $a
print $a;

// Displays de the content of $b
print_r($b);

// Returns de the content of $b, no display.
$c = var_export($b, true);

?>
```

Specs

Short name	Php/Prints
Rulesets	none
Exakt since	0.10.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.864 $HTTP_RAW_POST_DATA Usage

`$HTTP_RAW_POST_DATA` is deprecated, and should be replaced by `php://input`.

`$HTTP_RAW_POST_DATA` is deprecated since PHP 5.6.

It is possible to prepare code to this lack of feature by setting `always_populate_raw_post_data` to -1.

```
<?php

// PHP 5.5 and older
$postdata = $HTTP_RAW_POST_DATA;
```

(continues on next page)

(continued from previous page)

```
// PHP 5.6 and more recent
$postdata = file_get_contents(php://input);

?>
```

See also $HTTP_RAW_POST_DATA variable.

Suggestions

- Use php://input with fopen() instead.

Specs

Short name	Php/RawPostDataUsage
Rulesets	*CE*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.865 Reflection Export() Is Deprecated

export() method in Reflection classes is now deprecated. It is obsolete since PHP 7.4 and will disappear in PHP 8.0.

The Reflector interface, which is implemented by all reflection classes, specifies two methods: __toString() and export().

```
<?php

ReflectionFunction::export('foo');
// same as
echo new ReflectionFunction('foo'), \n;

$str = ReflectionFunction::export('foo', true);
// same as
$str = (string) new ReflectionFunction('foo');

?>
```

See also Reflection export() methods and Reflection.

Suggestions

- Cast the object to string
- Remove the call to export()

Specs

Short name	Php/ReflectionExportIsDeprecated
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.9.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.866 Reserved Keywords In PHP 7

PHP reserved names for class/trait/interface. They won't be available anymore in user space starting with PHP 7.

For example, string, float, false, true, null, resource,‘… <https://www.php.net/manual/en/functions.arguments.php#functions.variable-arg-list>‘_ are not acceptable as class name.

```
<?php

// This doesn't compile in PHP 7.0 and more recent
class null { }

?>
```

See also List of other reserved words.

Suggestions

- Avoid using PHP reserved keywords

Specs

Short name	Php/ReservedKeywords7
Rulesets	*CompatibilityPHP70*
Exakt since	0.8.4
Php Version	7.0-
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.867 Reserved Match Keyword

Match is a new instruction in PHP 8.0. For that, it becomes a reserved keyword, and cannot be used in various situations.

```
<?php

// Match as a standalone keyword
```

(continues on next page)

(continued from previous page)

```
use X as Match;

// No more use as a typehint
function foo(match $a ) : match {}
$a instanceof match;

// No use as method name or function name
matCH(a, 4) ;
$method->MAtch();
$static::MATch();

// Match in a Fully qualified name is OK
b\match ;

// Match a property name or a class constant is OK
$match->maTCH;
C::MATCH;

?>
```

See also Match expression V2.

Suggestions

- Change the name from Match to something else.

Specs

Short name	Php/ReservedMatchKeyword
Rulesets	*CompatibilityPHP80*
Exakt since	2.2.1
Php Version	8.0-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Unknown

13.2.868 PHP Keywords As Names

PHP has a set of reserved keywords. It is recommended not to use those keywords for names structures.

PHP does check that a number of structures, such as classes, methods, interfaces... can't be named or called using one of the keywords. However, in a few other situations, no check are enforced. Using keywords in such situation is confusing.

```
<?php

// This keyword is reserved since PHP 7.2
class object {
    // _POST is used by PHP for the $_POST variable
    // This methods name is probably confusing,
    // and may attract more than its share of attention
```

(continues on next page)

(continued from previous page)

```
    function _POST() {

    }
}

?>
```

See also List of Keywords, Predefined Classes, Predefined Constants, List of other reserved words and Predefined Variables.

Suggestions

- Rename the structure
- Choose another naming convention to avoid conflict and rename the current structures

Name	Default	Type	Description
reservedNames		string	Other reserved names : all in a string, comma separated.
allowedNames		string	PHP reserved names that can be used in the code. All in a string, comma separated.

Specs

Short name	Php/ReservedNames
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*ChurchCRM*, *xataface*

13.2.869 Return Typehint Usage

Spot usage of return typehint. It is a PHP 7.0 feature.

Return typehint were introduced in PHP 7.0, and are backward incompatible with PHP 5.x.

```
<?php

function foo($a) : stdClass {
    return new \stdClass();
}

?>
```

See also RFC: Return Type Declarations and Return Type Declarations.

Specs

Short name	Php/ReturnTypehintUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	7.0+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.870 Return With Parenthesis

return statement doesn't need parenthesis. PHP tolerates them with return statement, but it is recommended not to use them.

From the PHP Manual : 'Note: Note that since return is a language construct and not a function, the parentheses surrounding its argument are not required and their use is discouraged.'.

```
<?php

function foo() {
    $a = rand(0, 10);

    // No need for parenthesis
    return $a;

    // Parenthesis are useless here
    return ($a);

    // Parenthesis are useful here: they are needed by the multplication.
    return ($a + 1) * 3;
}

?>
```

See also PHP return(value); vs return value; and return.

Suggestions

- Remove the parenthesis

Specs

Short name	Php/ReturnWithParenthesis
Rulesets	*Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.871 Safe Phpvariables

Mark the safe PHP variables.

PHP superglobales are usually filled with external data that should be filtered. However, some values may be considered safe, as they are under the control of the developer.

`$_GET`, `$_POST`, `$_FILES`, `$_REQUEST`, `$_COOKIES` are all considered unsafe. Their level of validation is checked in other analysis.

`$_SERVER` is partially safe. It is valid for the following values : `DOCUMENT_ROOT`, `REQUEST_TIME`, `REQUEST_TIME_FLOAT`, `SCRIPT_NAME`, `SERVER_ADMIN`, `_`.

```
<?php

// DOCUMENT_ROOT is a safe variable
echo $_SERVER['DOCUMENT_ROOT'];

// $_SERVER's PHP_SELF MUST be validated before usage
echo $_SERVER['PHP_SELF'];

// $_GET MUST be validated before usage
echo $_GET['_'];

?>
```

See also Predefined Variables.

Specs

Short name	Php/SafePhpvars
Rulesets	none
Exakt since	2.1.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.872 Scalar Are Not Arrays

It is wrong to use a scalar as an array, a Warning is emitted. PHP 7.4 emits a Warning in such situations.

```
<?php

// Here, $x may be null, and in that case, the echo will fail.
function foo(?A $x) {
    echo $x[2];
}

?>
```

Typehinted argument with a scalar are reported by this analysis. Also, nullable arguments, both with typehint and return type hint.

See also E_WARNING for invalid container read array-access.

Suggestions

- Update type hints to avoid scalar values
- Remove the array syntax in the code using the results
- Cast to string type, so the array notation is accessible

Specs

Short name	Php/ScalarAreNotArrays
Rulesets	*Analyze*, *CE*, *CI-checks*, *CompatibilityPHP74*
Exakt since	1.9.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.873 Scalar Typehint Usage

Spot usage of scalar type hint : `int`, `float`, `boolean` and `string`.

Scalar typehint are PHP 7.0 and more recent. Some, like `object`, is 7.2.

Scalar typehint were not supported in PHP 5 and older. Then, the typehint is treated as a class name.

```
<?php

function withScalarTypehint(string $x) {}

function withoutScalarTypehint(someClass $x) {}

?>
```

See also PHP RFC: Scalar Type Hints and Type declarations.

Specs

Short name	Php/ScalarTypehintUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	7.0+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.874 Serialize Magic Method

Classes that defines __serialize() and __unserialize() are using Serialize Magic.

Serialize magic methods were introduced in PHP 7.4, and are not effective before.

```
<?php

class x {
    function __serialize() {}
    function __unserialize() {}
}

?>
```

See also New custom object serialization mechanism.

Suggestions

-

Specs

Short name	Php/SerializeMagic
Rulesets	none
Exakt since	1.9.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.875 Session Variables

Sessions names, used across the application.

```
<?php

if (isset($_SESSION['mySessionVariable'])) {
    $_SESSION['mySessionVariable']['counter']++;
} else {
    $_SESSION['mySessionVariable'] = array('counter'  => 1,
                                           'creation' => time());
}

?>
```

See also Sessions.

Specs

Short name	Php/SessionVariables
Rulesets	none
Exakt since	0.12.16
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.876 set_exception_handler() Warning

The set_exception_handler() callable function has to be adapted to PHP 7 : `Exception` is not the right typehint, it is now `Throwable`.

When in doubt about backward compatibility, just drop the typehint. Otherwise, use `Throwable`.

```
<?php

// PHP 5.6- typehint
class foo { function bar(\Exception $e) {} }

// PHP 7+ typehint
class foo { function bar(Throwable $e) {} }

// PHP 5 and PHP 7 compatible typehint (note : there is none)
class foo { function bar($e) {} }

set_exception_handler(foo);

?>
```

Specs

Short name	Php/SetExceptionHandlerPHP7
Rulesets	*CompatibilityPHP70*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.877 PHP Handlers Usage

PHP has a number of handlers that may be replaced by customized code : session, shutdown, error, exception. They are noted here.

The example is adapted from the PHP documentation of set_error_handler().

```php
<?php
// error handler function
function myErrorHandler($errno, $errstr, $errfile, $errline)
{
    if (!(error_reporting() & $errno)) {
        // This error code is not included in error_reporting, so let it fall
        // through to the standard PHP error handler
        return false;
    }

    switch ($errno) {
    case E_USER_ERROR:
        echo '<b>My ERROR</b> [$errno] $errstr<br />'.PHP_EOL;
        echo '  Fatal error on line '.$errline.' in file .'$errfile;
        echo ', PHP ' . PHP_VERSION . ' (' . PHP_OS . ')<br />'.PHP_EOL;
        echo 'Aborting...<br />'.PHP_EOL;
        exit(1);
        break;

    case E_USER_WARNING:
        echo '<b>My WARNING</b> ['.$errno.'] '.$errstr.'<br />'.PHP_EOL;
        break;

    case E_USER_NOTICE:
        echo '<b>My NOTICE</b> ['.$errno.'] '.$errstr.'<br />'.PHP_EOL;
        break;

    default:
        echo 'Unknown error type: ['.$errno.'] $errstr<br />'.PHP_EOL;
        break;
    }

    /* Don't execute PHP internal error handler */
    return true;
}


// set to the user defined error handler
$old_error_handler = set_error_handler(myErrorHandler);

?>
```

See also set_error_handler.

Specs

Short name	Php/SetHandlers
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.878 Shell Favorite

PHP has several syntax to make system calls. shell_exec(), exec() and back-ticks, ‘ are the common ones.

It was found that one of those three is actually being used over 90% of the time. The remaining cases should be uniformed, so has to make this code consistent.

```
<?php

// back-ticks &#96; are only used once.
&#96;back-tick&#96;;

shell_exec('exec1');
shell_exec('exec2');
shell_exec('exec3');
shell_exec('exec4');
shell_exec('exec5');
shell_exec('exec6');
shell_exec('exec7');
shell_exec('exec8');
shell_exec('exec9');
shell_exec('exec10');
shell_exec('exec11');
shell_exec('exec12');

?>
```

See also Execution Operators and shell_exec().

Specs

Short name	Php/ShellFavorite
Rulesets	none
Exakt since	0.12.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.879 Short Open Tags

Usage of short open tags is discouraged. The following files were found to be impacted by the short open tag directive at compilation time. They must be reviewed to ensure no <? tags are found in the code.

Specs

Short name	Php/ShortOpenTagRequired
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.880 Should Preprocess Chr()

Replace literal chr() calls with their escape sequence.

chr() is a functioncall, that cannot be cached. It is only resolved at execution time. On the other hand, literal values are preprocessed by PHP and may be cached.

```
<?php

// This is easier on PHP
$a = "\120\110\120\040 is great!";

// This is slow
$a = chr(80), chr(72), chr(80), chr(32), ' is great!';

// This would be the best with this example, but it is not always possible
$a = 'PHP is great!';

?>
```

This is a micro-optimisation.

See also Escape sequences.

Suggestions

- Use PHP string sequences, and skip chr() at execution time

Specs

Short name	Php/ShouldPreprocess
Rulesets	*Suggestions*
Exakt since	1.1.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*phpadsnew*

13.2.881 Should Use array_column()

Avoid writing a whole slow loop, and use the native array_column().

array_column() is a native PHP function, that extract a property or a index from a array of object, or a multidimensional array. This prevents the usage of foreach to collect those values.

```
<?php

$a = array(array('b' => 1),
           array('b' => 2, 'c' => 3),
           array(          'c' => 4)); // b doesn't always exists

$bColumn = array_column($a, 'b');

// Slow and cumbersome code
$bColumn = array();
foreach($a as $k => $v) {
    if (isset($v['b'])) {
        $bColumn[] = $v['b'];
    }
}

?>
```

array_column() is faster than foreach() (with or without the isset() test) with 3 elements or more, and it is significantly faster beyond 5 elements. Memory consumption is the same.

See also [blog] ‘array_column() <https://benramsey.com/projects/array-column/>‘_.

Suggestions

- Use array_column(), instead of a foreach()

Specs

Short name	Php/ShouldUseArrayColumn
Rulesets	*Performances*, *Suggestions*
Exakt since	0.10.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.882 Should Use array_filter()

Should use array_filter().

array_filter() is a native PHP function, that extract elements from an array, based on a closure or a function. Using array_filter() shortens your code, and allows for reusing the filtering logic across the application, instead of hard coding it every time.

```
<?php

$a = range(0, 10); // integers from 0 to 10

// Extracts odd numbers
$odds = array_filter($a, function($x) { return $x % 2; });
$odds = array_filter($a, 'odd');

// Slow and cumbersome code for extracting odd numbers
$odds = array();
foreach($a as $v) {
    if ($a % 2) { // same filter than the closure above, or the odd function below
        $bColumn[] = $v;
    }
}

function foo($x) {
    return $x % 2;
}

?>
```

array_filter() is faster than foreach() (with or without the isset() test) with 3 elements or more, and it is significantly faster beyond 5 elements. Memory consumption is the same.

See also array_filter.

Suggestions

- Use array_filter()

Specs

Short name	Php/ShouldUseArrayFilter
Rulesets	*Suggestions*
Exakt since	1.0.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*xataface*, *shopware*

13.2.883 Should Use Coalesce

PHP 7 introduced the ?? operator, that replaces longer structures to set default values when a variable is not set.

```
<?php

// Fetches the request parameter user and results in 'nobody' if it doesn't exist
$username = $_GET['user'] ?? 'nobody';
// equivalent to: $username = isset($_GET['user']) ? $_GET['user'] : 'nobody';
```

(continues on next page)

(continued from previous page)

```
// Calls a hypothetical model-getting function, and uses the provided default if it␣
↪fails
$model = Model::get($id) ?? $default_model;
// equivalent to: if (($model = Model::get($id)) === NULL) { $model = $default_model;␣
↪}

?>
```

Sample extracted from PHP docs Isset Ternary.

See also New in PHP 7: null coalesce operator.

Suggestions

- Replace the long syntax with the short one

Specs

Short name	Php/ShouldUseCoalesce
Rulesets	*Analyze*, *CI-checks*, *Suggestions*
Exakt since	0.8.4
Php Version	7.0+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*ChurchCRM*, *Cleverstyle*

13.2.884 Should Use Function

Functioncalls that fall back to global scope should be using 'use function' or be fully namespaced.

PHP searches for functions in the local namespaces, and in case it fails, makes the same search in the global scope. Anytime a native function is referenced this way, the search (and fail) happens. This slows down the scripts.

The speed bump range from 2 to 8 %, depending on the availability of functions in the local scope. The overall bump is about 1 µs per functioncall, which makes it a micro optimisation until a lot of function calls are made.

Based on one of Marco Pivetta tweet.

```
<?php

namespace X {
    use function strtolower as strtolower_aliased;

    // PHP searches for strtolower in X, fails, then falls back to global scope,␣
↪succeeds.
    $a = strtolower($b);

    // PHP searches for strtolower in global scope, succeeds.
    $a = \strtolower($b);
```

(continues on next page)

(continued from previous page)

```
    // PHP searches for strtolower_aliased in global scope, succeeds.
    $a = \strtolower_aliased($b);
}

?>
```

This analysis is a related to Performances/Php74ArrayKeyExists, and is a more general version.

See also blog post.

Suggestions

- Use the *use* command for arrray_key_exists(), at the beginning of the script
- Use an initial before array_key_exists()
- Remove the namespace

Specs

Short name	Php/ShouldUseFunction
Rulesets	*Performances*
Exakt since	0.9.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.885 Signature Trailing Comma

Trailing comma in method signature. This feature was added in PHP 8.0.

Allowing the trailing comma makes it possible to reduce the size of VCS's diff, when adding , removing a parameter.

```
<?php

// Example from the RFC
class Uri {
    private function __construct(
        ?string $scheme,
        ?string $user,
        ?string $pass,
        ?string $host,
        ?int $port,
        string $path,
        ?string $query,
        ?string $fragment // <-- ARGH!
    ) {
        ...
    }
}
?>
```

See also PHP RFC: Allow trailing comma in parameter list.

Suggestions

-

Specs

Short name	Php/SignatureTrailingComma
Rulesets	*CE*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP74*
Exakt since	2.1.0
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.886 Spread Operator For Array

The variadic operator may be used with arrays. This has been introduced in PHP 7.4.

list() is not allowed to use this operator, as list() expected variables, not values.

```
<?php

$array = [1, 2, 3];
$extended_array = [...$array, 4, 5, 6];

// invalid syntax
[...$a] = [1,2,3];

?>
```

See also Spread Operator in Array Expression.

Suggestions

-

Specs

Short name	Php/SpreadOperatorForArray
Rulesets	*CE*
Exakt since	1.9.4
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.887 ::class

PHP has a special class constant to hold the name of the class : `class` keyword. It represents the class name that is used in the left part of the operator.

Using `\:\:class` is safer than relying on a string. It does adapt if the class's name or its namespace is changed'. It is also faster, though it is a micro-optimisation.

It is introduced in PHP 5.5.

```
<?php

use A\B\C as UsedName;

class foo {
    public function bar( ) {
        echo ClassName::class;
        echo UsedName::class;
    }
}

$f = new Foo( );
$f->bar( );
// displays ClassName
// displays A\B\C

?>
```

Be aware that `\:\:class` is a replacement for __CLASS__ magic constant.

See also Class Constant.

Suggestions

- Use ::class whenever possible. That exclude any dynamic call.

Specs

Short name	Php/StaticclassUsage
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*
Exakt since	0.8.4
Php Version	5.5+
Severity	Major
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.888 Strtr Arguments

Strtr() replaces characters by others in a string. When using strings, strtr() replaces characters as long as they have a replacement. All others are ignored.

In particular, strtr() works on strings of the same size, and cannot be used to remove chars.

```php
<?php

$string = 'abcde';
echo strtr($string, 'abc', 'AB');
echo strtr($string, 'ab', 'ABC');
// displays ABcde
// c is ignored each time

// strtr can't remove a char
echo strtr($string, 'a', '');
// displays a

?>
```

See also strtr.

Suggestions

- Check the call to strtr() and make sure the arguments are of the same size
- Replace strtr() with str_replace(), which works with strings and array, not chars
- Replace strtr() with preg_match(), which works with patterns and not chars

Specs

Short name	Php/StrtrArguments
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.2.3
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
Examples	*SuiteCrm*

13.2.889 Super Global Usage

Spot usage of Super global variables, such as $_GET, $_POST or $_REQUEST.

```php
<?php

echo htmlspecialchars($_GET['name'], UTF-8);

?>
```

See also Superglobals.

Specs

Short name	Php/SuperGlobalUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.890 Throw

List of thrown exceptions.

```
<?php
if ($divisor === 0) {
    // Throw native exception
    throw new DivisionByZeroError("Shouldn't divide by one");
}

if ($divisor === 1) {
    // Throw custom exception
    throw new DontDivideByOneException("Shouldn't divide by one");
}
?>
```

See also Exceptions.

Specs

Short name	Php/ThrowUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.891 Throw Was An Expression

Throw used to be an expression. In PHP 7.0, there were some location where one couldn't use a throw : this was the case for arrow functions, which expect one expression as function's body.

Using throw as an instruction makes the code incompatible with PHP 7 version and older.

```
<?php

// Valid in PHP 8.0 and more recent
$fn = fn($a) => throw new Exception($a);

?>
```

See also Throw Expression and Exceptions.

Suggestions

-

Specs

Short name	Php/ThrowWasAnExpression
Rulesets	*CE*, *CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP74*
Exakt since	2.1.1
Php Version	8.0+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.892 Too Many Native Calls

Avoid stuffing too many PHP native call inside another functioncall.

For readability reasons, or, more often, for edge case handling, it is recommended to avoid nesting too many PHP native calls.

This analysis reports any situation where more than 3 PHP native calls are nested.

```
<?php

// Too many nested functions
$cleanArray = array_unique(array_keys(array_count_values(array_column($source, 'x
↪'))));

// Avoid warning when source is empty
$extract = array_column($source, 'x');
if (empty($extract)) {
    $cleanArray = array();
} else {
    $cleanArray = array_unique(array_keys(array_count_values($extract)));
}

// This is not readable, although it is short.
// It may easily get out of hand.
echo chr(80), chr(72), chr(80), chr(32), ' is great!';

?>
```

Name	Default	Type	Description
nativeCallCounts	3	integer	Number of native calls found inside another call.

Specs

Short name	Php/TooManyNativeCalls
Rulesets	*Analyze*
Exakt since	1.1.10
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*SPIP*

13.2.893 Trailing Comma In Calls

The last argument may be left empty.

This feature was introduced in PHP 7.3.

```
<?php

// VCS friendly call
// PHP 7.3 and more recent
foo(1,
    2,
    3,
   );

// backward compatible call
// All PHP versions
foo(1,
    2,
    3
   );

?>
```

See also PHP RFC: Allow a trailing comma in function calls.

Specs

Short name	Php/TrailingComma
Rulesets	*CE*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*
Exakt since	1.4.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.894 Trigger Errors

List of situations where user errors are triggered.

PHP errors are triggered with trigger_error().

```
<?php
if ($divisor == 0) {
    trigger_error('Cannot divide by zero', E_USER_ERROR);
}
?>
```

See also trigger_error.

Specs

Short name	Php/TriggerErrorUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.895 Caught Expressions

List of caught exceptions.

```
<?php

// This analyzer reports MyException and Exception
try {
    doSomething();
} catch (MyException $e) {
    fixIt();
} catch (\Exception $e) {
    fixIt();
}

?>
```

Specs

Short name	Php/TryCatchUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.896 Try With Multiple Catch

Try may be used with multiple catch clauses.

```
<?php

try {
    OneCatch();
} catch (FirstException $e) {

}

try {
    TwoCatches();
} catch (FirstException $e) {
} catch (SecondException $e) {
}

?>
```

See also Exceptions.

Specs

Short name	Php/TryMultipleCatch
Rulesets	*CE*
Exakt since	0.11.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.897 Typed Property Usage

Traditionally, PHP properties aren't typed. Since PHP 7.4, it is possible to type properties, just like arguments.

```
<?php

class User {
    public int $id;
    public string $name;

    public function __construct(int $id, string $name) {
        $this->id = $id;
        $this->name = $name;
    }
}
?>
```

See also Typed Properties 2.0.

Suggestions

-

Specs

Short name	Php/TypedPropertyUsage
Rulesets	*CE*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*
Exakt since	1.6.2
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.898 Unicode Escape Partial

PHP 7 introduces a new escape sequence for strings : u{hex}. It is backward incompatible with previous PHP versions for two reasons :

PHP 7 will recognize en replace those sequences, while PHP 5 keep them intact. PHP 7 will halt on partial Unicode Sequences, as it tries to understand them, but may fail.

```
<?php

echo \u{1F418}\n;
// PHP 5 displays the same string
// PHP 7 displays : an elephant

echo \u{NOT A UNICODE CODEPOINT}\n;
// PHP 5 displays the same string
// PHP 7 emits a fatal error

?>
```

Is is recommended to check all those strings, and make sure they will behave correctly in PHP 7.

Specs

Short name	Php/UnicodeEscapePartial
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	7.0-
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.899 Unicode Escape Syntax

Usage of the Unicode Escape syntax, with the `\u{xxxxx}` format, available since PHP 7.0.

```
<?php

// Produce an elephant icon in PHP 7.0+
echo \u{1F418};

// Produce the raw sequence in PHP 5.0
echo \u{1F418};

?>
```

See also PHP RFC: Unicode Codepoint Escape Syntax, Code point and Unicode.

Specs

Short name	Php/UnicodeEscapeSyntax
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	7.0+
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.900 Unknown Pcre2 Option

`PCRE2` supports different options, compared to `PCRE1`. `PCRE2` was adopted with PHP 7.3.

The `S` modifier : it used to tell PCRE to spend more time studying the regex, so as to be faster at execution. This is now the default behavior, and may be dropped from the regex.

The `X` modifier : `X` is still existing with `PCRE2`, though it is now the default for `PCRE2`, and not for PHP as time of writing. In particular, `Any backslash in a pattern that is followed by a letter that has no special meaning causes an error, thus reserving these combinations for future expansion. ``. It is recommended to avoid using useless sequence \s in regex to get ready for that change. All the following letters ``gijkmoqyFIJMOTY` . Note that `clLpPuU` are valid `PRCE` sequences, and are probably failing for other reasons.

```
<?php

// \y has no meaning. With X option, this leads to a regex compilation error, and a␣
↪failed test.
preg_match('/ye\y/', $string);
preg_match('/ye\y/X', $string);

?>
```

See also Pattern Modifiers and PHP RFC: PCRE2 migration.

Specs

Short name	Php/UnknownPcre2Option
Rulesets	*Analyze*, *CompatibilityPHP73*
Exakt since	1.0.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.901 Unpacking Inside Arrays

> The variadic operator is now available inside arrays. Until PHP 7.4, it is not possible to use the variadic operator, or ... inside arrays.

The workaround is to use array_merge(), after checking that arrays are not empty.

```
<?php

$a = ['a', 'b', 'c'];
$b = ['d', 'e', 'f'];

// PHP 7.4
$c = [...$a, ...$b];

// PHP 7.3 and older
$c = array_merge($a, $b);

?>
```

See also Spread Operator in Array Expression and PHP 5.6 and the Splat Operator .

Suggestions

- Replace array_merge() with ....

Specs

Short name	Php/UnpackingInsideArrays
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*
Exakt since	1.8.0
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.902 Unset() Or (unset)

Unset() and (unset) have the same functional use.

The analyzed code has less than 10% of one of them : for consistency reasons, it is recommended to make them all the same.

It happens that unset() or (unset) are used depending on coding style and files. One file may be consistently using unset(), while the others are all using (unset).

```
<?php

// be consistent
(unset) $a1;
(unset) $a2;
(unset) $a3;
(unset) $a4;
(unset) $a5;
(unset) $a6;
(unset) $a7;
(unset) $a8;
(unset) $a9;
(unset) $a10;
(unset) $a11;
(unset) $a12;

unset($b);
?>
```

Specs

Short name	Php/UnsetOrCast
Rulesets	none
Exakt since	0.9.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.903 Unusual Case For PHP Functions

Usually, PHP functions are written all in lower case.

```
<?php

// All uppercases PHP functions
ECHO STRTOLOWER('This String');

?>
```

Specs

Short name	Php/UpperCaseFunction
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.904 Non-lowercase Keywords

The usual convention is to write PHP keywords (like `as`, `foreach`, `switch`, `case`, `break`, etc.) all in lowercase.

```
<?php

// usual PHP convention
foreach($array as $element) {
    echo $element;
}

// unusual PHP conventions
Foreach($array AS $element) {
    eCHo $element;
}

?>
```

PHP understands them in lowercase, UPPERCASE or WilD Case, so there is nothing compulsory here. Although, it will look strange to many.

Some keywords are missing from this analysis : `extends`, `implements`, `as`. This is due to the internal engine, which doesn't keep track of them in its AST representation.

Suggestions

- Use lowercase only PHP keywords, except for constants such as __CLASS__.

Specs

Short name	Php/UpperCaseKeyword
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.905 Use PHP Attributes

PHP 8.0 attributes. WIP.

```
<?php

@@foo(4)
class x {

}

?>
```

See also PHP RFC: Shorter Attribute Syntax, Attributes Amendements and Shorter Attribute Syntax Change.

.

Specs

Short name	Php/UseAttributes
Rulesets	*CE*
Exakt since	2.1.6
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.906 Use Browscap

Browscap is a browser database, accessible via get_browser().

Browscap is the 'Browser Capabilities Project'.

```
<?php
echo $_SERVER['HTTP_USER_AGENT'] . \n\n;

$browser = get_browser(null, true);
print_r($browser);
?>
```

See also browscap.

Specs

Short name	Php/UseBrowscap
Rulesets	*CE*
Exakt since	0.11.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.907 Use Cli

Signal the usage of code in CLI mode, through the usage of *$argv* and *$argc* variables.

```
<?php

// Characteristics of CLI usage
getopt(abcd);

?>
```

Specs

Short name	Php/UseCli
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.908 Use Contravariance

Contravariance is compatible argument typehint. A child class may accept an object of a parent class of the argument type of its parent's method.

Since a children class may accept a parent class of the argynebt type, the evolution is in opposite order.

Contravariance is a PHP 7.4 feature. Contravariance is distinct from return type covariance.

```
<?php
class X {
  function m(Y $z): X {}
}

// m is overwriting the parent's method.
// The return type is different.
// The return type is compatible, as Y is also a sub-class of X.
class Y extends X {
  function m(X $z): Y {}
}

?>
```

See also Covariant Returns and Contravariant Parameters **and** *Php/UseCovariance*.

Suggestions

-

Specs

Short name	Php/UseContravariance
Rulesets	*CE*
Exakt since	1.9.3
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.909 Use Cookies

This code source uses cookies.

Cookie usage is spotted with the usage of setcookie(), setawcookie() and header() with the 'Set-Cookie' header.

```
<?php

     header('Set-Cookie: '.$name.'='.$value.'; EXPIRES'.$date.';');

    // From the PHP Manual :
    setcookie('TestCookie3', $value, time()+3600, '/~rasmus/', 'example.com', 1);

?>
```

See also : Cookies.

Specs

Short name	Php/UseCookies
Rulesets	*CE*
Exakt since	0.10.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.910 Use Covariance

> Covariance is compatible return typehint. A child class may return an object of a child class of the return type of its parent's method.

Since a children class may return a children class of the return type, the evolution is in the same order.

Covariance is a PHP 7.4 feature. Covariance is distinct from argument contravariance.

```
<?php
class X {
  function m(Y $z): X {}
}

// m is overwriting the parent's method.
// The return type is different.
// The return type is compatible, as Y is also a sub-class of X.
class Y extends X {
  function m(X $z): Y {}
}

?>
```

See also **Covariant Returns and Contravariant Parameters** **and** *Php/UseContravariance*.

Suggestions

-

Specs

Short name	Php/UseCovariance
Rulesets	*CE*
Exakt since	1.9.3
Php Version	7.4+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.911 Use DateTimeImmutable Class

The DateTimeImmutable <https://www.php.net/'DateTimeImmutable>'_ class is the immutable version of the Datetime class.

While DateTime <https://www.php.net/'DateTime>'_ may be modified 'in situ', `DateTimeImmutable` cannot be modified. Any modification to such an object will return a new and distinct object. This avoid interferences that are hard to track.

```
<?php
// Example extracted from Derick Rethans' article (link below)

function formatNextMondayFromNow( DateTime $dt )
{
        return $dt->modify( 'next monday' )->format( 'Y-m-d' );
}

$d = new DateTime();                            //2014-02-17
echo formatNextMondayFromNow( $d ), \n;
echo $d->format( 'Y-m-d' ), \n;              //2014-02-17
?>
```

See also What's all this 'immutable date' stuff, anyway?, DateTimeImmutable, The 'DateTime <https://www.php.net/'DateTime>'_ class <https://www.php.net/manual/en/class.datetime.php>'_ and The 'DateTimeImmutable <https://www.php.net/'DateTimeImmutable>'_ class <https://www.php.net/manual/en/class.datetimeimmutable.php>'_.

Suggestions

- Always use DateTimeImmutable when manipulating dates.

Specs

Short name	Php/UseDateTimeImmutable
Rulesets	*Suggestions*
Exakt since	1.8.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.912 Use get_debug_type()

get_debug_type() returns the given type of a variable. It was introduced in PHP 8.0.

```
<?php
  // From the RFC
  throw new TypeError('Expected ' . Foo::class . ' got ' . (is_object($bar) ? get_
↪class($bar) : gettype($bar)));

  // Becomes
  throw new TypeError('Expected ' . Foo::class . ' got ' . get_debug_type($bar));
```

(continues on next page)

(continued from previous page)

```
?>
```

See also PHP RFC: get_debug_type.

Suggestions

- Replace the ternary with a call to get_debug_type()

Specs

Short name	Php/UseGetDebugType
Rulesets	*Suggestions*
Exakt since	2.1.9
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.913 Uses PHP 8 Match()

Use the match() syntax.

```
<?php


$A = match($a) {
    'a' => 'A',
    'b' => 'B',
    default => 'd',
};

?>
```

See also match Match expression.

Specs

Short name	Php/UseMatch
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	2.1.4
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.914 Use Nullable Type

The code uses nullable type, available since PHP 7.1.

Nullable Types are an option to type hint : they allow the passing value to be null, or another type.

According to the authors of the feature : 'It is common in many programming languages including PHP to allow a variable to be of some type or null. This null often indicates an error or lack of something to return.'

```
<?php

function foo(?string $a = 'abc') : ?string {
    return $a.b;
}

?>
```

See also Type declarations and PHP RFC: Nullable Types.

Specs

Short name	Php/UseNullableType
Rulesets	*CE*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*
Exakt since	0.8.4
Php Version	7.1+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.915 Use NullSafe Operator

The nullsafe operator ?-> is an alternative to the object operator ->. It silently fails the whole expression if a null is used for object.

```
<?php

$o = null;

// PHP 8.0 Failsafe : $r = null;
$r = $o->method();

// PHP 7.4- : Call to a member function method() on null
$r = $o->method();

?>
```

See also PHP RFC: Nullsafe operator.

Specs

Short name	Php/UseNullSafeOperator
Rulesets	*CE*
Exakt since	2.1.6
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.916 Use PHP Object API

OOP API is the modern version of the PHP API.

When PHP offers the alternative between procedural and OOP api for the same features, it is recommended to use the OOP API.

Often, this least to more compact code, as methods are shorter, and there is no need to bring the resource around. Lots of new extensions are directly written in OOP form too.

OOP / procedural alternatives are available for mysqli <https://www.php.net/manual/en/book.‘mysqli.php>‘_, tidy <https://www.php.net/manual/en/book.‘tidy.php>‘_, cairo, finfo, and some others.

```
<?php
/// OOP version
$mysqli = new mysqli(localhost, my_user, my_password, world);

/* check connection */
if ($mysqli->connect_errno) {
    printf(Connect failed: %s\n, $mysqli->connect_error);
    exit();
}

/* Create table doesn't return a resultset */
if ($mysqli->query(CREATE TEMPORARY TABLE myCity LIKE City) === TRUE) {
    printf(Table myCity successfully created.\n);
}

/* Select queries return a resultset */
if ($result = $mysqli->query(SELECT Name FROM City LIMIT 10)) {
    printf(Select returned %d rows.\n, $result->num_rows);

    /* free result set */
    $result->close();
}
?>
```

```
<?php
/// Procedural version
$link = mysqli_connect(localhost, my_user, my_password, world);

/* check connection */
if (mysqli_connect_errno()) {
    printf(Connect failed: %s\n, mysqli_connect_error());
    exit();
```

(continues on next page)

(continued from previous page)

```
}

/* Create table doesn't return a resultset */
if (mysqli_query($link, CREATE TEMPORARY TABLE myCity LIKE City) === TRUE) {
    printf(Table myCity successfully created.\n);
}

?>
```

Suggestions

- Use the object API

Specs

Short name	Php/UseObjectApi
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
ClearPHP	use-object-api
Examples	*WordPress*, *PrestaShop*

13.2.917 Use Pathinfo

Use pathinfo() function instead of string manipulations.

pathinfo() is more efficient and readable and string functions.

```
<?php

$filename = '/path/to/file.php';

// With pathinfo();
$details = pathinfo($filename);
print $details['extension'];  // also capture php

// With string functions (other solutions possible)
$ext = substr($filename, - strpos(strreverse($filename), '.')); // Capture php

?>
```

When the path contains UTF-8 characters, pathinfo() may strip them. There, string functions are necessary.

Suggestions

- Use pathinfo() and its second argument

Specs

Short name	Php/UsePathinfo
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*SuiteCrm*

13.2.918 Use pathinfo() Arguments

pathinfo() has a second argument to select only useful data.

It is twice faster to get only one element from pathinfo() than get the four of them, and use only one.

This analysis reports pathinfo() usage, without second argument, where only one or two indices are used, after the call.

```
<?php

// This could use only PATHINFO_BASENAME
function foo_db() {
    $a = pathinfo($file2);
    return $a['basename'];
}

// This could be 2 calls, with PATHINFO_BASENAME and PATHINFO_DIRNAME.
function foo_de() {
    $a = pathinfo($file3);
    return $a['dirname'].'/'.$a['basename'];
}

// This is OK : 3 calls to pathinfo() is slower than array access.
function foo_deb() {
    $a = pathinfo($file4);
    return  $a['dirname'].'/'.$a['filename'].'.'.$a['extension'];
}

?>
```

Depending on the situation, the functions dirname() and basename() may also be used. They are even faster, when only fetching those data.

See also list.

Suggestions

- Use PHP native function pathinfo() and its arguments

Specs

Short name	Php/UsePathinfoArgs
Rulesets	*Performances*
Exakt since	0.12.12
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Zend-Config*, *ThinkPHP*

13.2.919 Uses Environment

Spot usage of $_ENV and getenv() putenv() functions that will fetch data from the environment.

```
<?php

// Take some configuration from the environment
$secret_key = getenv('secret_key');

?>
```

Specs

Short name	Php/UsesEnv
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.920 Use session_start() Options

It is possible to set the session's option at session_start() call, skipping the usage of session_option().

This way, session's options are set in one call, saving several hits.

This is available since PHP 7.0. It is recommended to set those values in the `php.ini` file, whenever possible.

```
<?php

// PHP 7.0
session_start(['session.name' => 'mySession',
               'session.cookie_httponly' => 1,
               'session.gc_maxlifetime' => 60 * 60);

// PHP 5.6- old way
ini_set ('session.name', 'mySession');
ini_set(session.cookie_httponly, 1);
```

(continues on next page)

(continued from previous page)

```
ini_set('session.gc_maxlifetime', 60 * 60);
session_start();

?>
```

Suggestions

- Use session_start() with array arguments

Specs

Short name	Php/UseSessionStartOptions
Rulesets	*Suggestions*
Exakt since	0.11.8
Php Version	7.0+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*WordPress*

13.2.921 Should Use SetCookie()

Use setcookie() or setrawcookie(). Avoid using header() to do so, as the PHP native functions are more convenient and easier to spot during a refactoring.

setcookie() applies some encoding internally, for the value of the cookie and the date of expiration. Rarely, this encoding has to be skipped : then, use setrawencoding().

Both functions help by giving a checklist of important attributes to be used with the cookie.

```
<?php

// same as below
setcookie(myCookie, 'chocolate', time()+3600, /, , true, true);

// same as above. Slots for path and domain are omitted, but should be used whenever␣
↪possible
header('Set-Cookie: myCookie=chocolate; Expires='.date('r', (time()+3600)).'; Secure;␣
↪HttpOnly');

?>
```

See also Set-Cookie, setcookie.

Suggestions

- Use setcookie() function, instead of header()
- Use setcookie() function, instead of header()

Specs

Short name	Php/UseSetCookie
Rulesets	*Analyze*
Exakt since	0.10.6
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.922 Avoid Using stdClass

`stdClass` is the default class for PHP. It is instantiated when PHP needs to return a object, but no class is specifically available.

It is recommended to avoid instantiating this class, nor use it is any way.

```
<?php

$json = '{a:1,b:2,c:3}';
$object = json_decode($json);
// $object is a stdClass, as returned by json_decode

// Fast building of $o
$a = [];
$a['a'] = 1;
$a['b'] = 2;
$a['c'] = 3;
json_encode( (object) $a);

// Slow building of $o
$o = new stdClass();
$o->a = 1;
$o->b = 2;
$o->c = 3;
json_encode($o);

?>
```

If you need a `stdClass` object, it is faster to build it as an array, then cast it, than instantiate `stdClass`. This is a micro-optimisation.

Suggestions

- Create a custom class to handle the properties

Specs

Short name	Php/UseStdclass
Rulesets	*Analyze*
Exakt since	0.9.1
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.923 Use str_contains()

str_contains() checks if a string is within another one. It replaces a call to strpos() with a comparison.

```
<?php

if (str_contains(abc, a)) { doSomething(); }

// strpos is used only for detection.
if (strpos(abc, a) !== false) { doSomething(); }

// strpos returns a position,
$pos = strpos(abca, a, 3);
if ($pos > 3) { doSomething();

?>
```

Note that this function is case sensitive : it cannot replace stripos().

Note that this function is single-byte only : it cannot replace mb_strpos().

This analysis omits calls to strpos() that are saved to a variable. strpos() is actually returning the position of the found string in the haystack, which may be reused later.

See also PHP RFC: str_contains.

Suggestions

- Switch to str_contains()

Specs

Short name	Php/UseStrContains
Rulesets	*Suggestions*
Exakt since	2.2.0
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.924 Use Closure Trailing Comma

Use a trailing comma in the closure's use list. This feature was added in PHP 8.0.

```
<?php

// PHP 8.0 valid syntax
$f = function foo() use ($a, ) { };

// always valid syntax for closure
$f = function foo() use ($a ) { };

?>
```

See also Trailing Comma In 'Closure Use List <https://wiki.php.net/rfc/trailing_comma_in_closure_use_list>'_.

Specs

Short name	Php/UseTrailingUseComma
Rulesets	*CE*
Exakt since	2.1.6
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.925 Use Web

The code is used in web environment.

The web usage is identified through the usage of the superglobals.

```
<?php

// Accessing $_GET is possible when PHP is used in a web server.
$x = filter_validate($_GET['x'], FILTER_EMAIL);

?>
```

Specs

Short name	Php/UseWeb
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.926 Usort Sorting In PHP 7.0

Usort(), uksort() and uasort() behavior has changed in PHP 7. Values that are equals (based on the user-provided method) may be sorted differently than in PHP 5.

If this sorting is important, it is advised to add extra comparison in the user-function and avoid returning 0 (thus, depending on default implementation).

```
<?php

$a = [ 2, 4, 3, 6];

function noSort($a) { return $a > 5; }

usort($a, 'noSort');
print_r($a);

?>
```

In PHP 5, the results is ::

```
Array
(
    [0] => 3
    [1] => 4
    [2] => 2
    [3] => 6
)
```

in PHP 7, the result is ::

```
Array
(
    [0] => 2
    [1] => 4
    [2] => 3
    [3] => 6
)
```

Suggestions

- Make sure the sorting function doesn't generate any ex-aequos.

Specs

Short name	Php/UsortSorting
Rulesets	*CompatibilityPHP70*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	Medium

13.2.927 Wrong Attribute Configuration

A class is attributed to the wrong PHP structure.

```
<?php
#[Attribute(Attribute::TARGET_CLASS)]
class ClassAttribute { }

// Wrong
#[ClassAttribute]
function foo () {}

// OK
#[ClassAttribute]
class y  {}

?>
```

Suggestions

- Remove the attribute from the wrongly attributed structure
- Extend the configuration of the attribute with Attribute::TARGET_*

Specs

Short name	Php/WrongAttributeConfiguration
Rulesets	*Analyze*
Exakt since	2.2.0
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.928 Wrong Type For Native PHP Function

This analysis reports calls to a PHP native function with a wrongly typed value.

```
<?php

// valid calls
echo exp(1);
echo exp(2.5);

// invalid calls
echo exp(1);
echo exp(array(2.5));

// valid call, but invalid math
// -1 is not a valid value for log(), but -1 is a valid type (int) : it is not␣
↪reported by this analysis.
echo log(-1);
?>
```

Suggestions

- Set the code to the valid type, when calling a PHP native function

Specs

Short name	Php/WrongTypeForNativeFunction
Rulesets	*Analyze*, *CI-checks*
Exakt since	2.1.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.929 Yield From Usage

Usage of generator delegation, with `yield from` keyword.

In PHP 7, generator delegation allows you to yield values from another `Generator`, `Traversable` object, or array by using the `yield from`.

`Yield from` was introduced in PHP 7.1, and is backward incompatible.

```
<?php

// Yield delegation
function foo() {
    yield from bar();
}

function bar() {
    yield 1;
}

?>
```

See also Generator Syntax and Understanding PHP Generators.

Specs

Short name	Php/YieldFromUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	7.0+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.930 Yield Usage

Usage of generators, with yield keyword.

Yield was introduced in PHP 5.5, and is backward incompatible.

```
<?php

function prime() {
    $primes = [2, 3, 5, 7, 11, 13, 17, 19];
    foreach($primes as $prime) {
        yield $prime;
    }
}

?>
```

See also Generator Syntax, Deal with Memory Gently using Yield in PHP and Understanding PHP Generators.

Specs

Short name	Php/YieldUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	5.5+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.931 Fopen Binary Mode

Use explicit `b` when opening files.

fopen() supports a `b` option in the second parameter, to make sure the read is binary. This is the recommended way when writing portable applications, between Linux and Windows.

```
<?php

// This opens file with binary reads on every OS
$fp = fopen('path/to/file.doc', 'wb');

// This may not open files with binary mode on Windows
$fp = fopen('path/to/file.doc', 'w');

?>
```

Also, Windows PHP does support a `t` option, that translates automatically line endings to the right value. As this is Windows only, this should be avoided for portability reasons.

See also fopen.

Specs

Short name	Portability/FopenMode
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.932 GLOB_BRACE Usage

GLOB_BRACE is not always available. This is the case on Solaris OS, and on Alpine OS, used for Docker.

```
<?php

// glob uses GLOB_BRACE
$abcFiles = glob($path.'/{a,b,c}*', GLOB_BRACE);

// avoiding usage of GLOB_BRACE
$abcFiles = array_merge(glob($path.'/a*'),
                        glob($path.'/b*'),
                        glob($path.'/c*'),
                       );

?>
```

It is possible to check the support for GLOB_BRACE by checking the presence of the constant.

See also Alpine Linux and GLOB_BRACE breaks Sulu on Alpine Linux.

Suggestions

- Create as many glob() calls at there are alternative in the braces

Specs

Short name	Portability/GlobBraceUsage
Rulesets	none
Exakt since	2.1.6
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.933 Iconv With Translit

The transliteration feature of iconv() depends on the underlying system to support it.

```
<?php

$string = iconv('utf-8', 'utf-8//TRANSLIT', $source);

?>
```

See also iconv().

Suggestions

- Use an OS that supports TRANSLIT with iconv
- Remove the usage of TRANSLIT

Specs

Short name	Portability/IconvTranslit
Rulesets	none
Exakt since	2.1.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.934 Linux Only Files

List of files that are only found on Linux style systems. They are making the application depend on the system.

```
<?php

// Really non-portable system check
$os = shell_exec(cat /proc/version);
echo You are using $os\n;

?>
```

Specs

Short name	Portability/LinuxOnlyFiles
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.935 PSR-11 Usage

PSR-11 describes a common interface for dependency injection containers.

It is supported by an set of interfaces, that one may use in the code.

```
<?php

namespace MyNamespace;

// MyContainerInterface implements the PSR-7 ServerRequestInterface.
// MyContainerInterface is more of a black hole than a real Container.
class MyContainerInterface implements \Psr\Container\ContainerInterface {
    public function get($id) {}
    public function has($id) {}
}

?>
```

See also PSR-11 : Dependency injection container.

Specs

Short name	Psr/Psr11Usage
Rulesets	*CE*
Exakt since	0.11.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.936 PSR-13 Usage

PSR-13 describes a common interface for dependency injection containers.

It is supported by an set of interfaces, that one may use in the code.

```
<?php

namespace MyNamespace;

// MyLink implements the PSR-13 LinkInterface.
// MyLink is more of a black hole than a real Container.
class MyLink implements LinkInterface {
    public function getHref() {}
    public function isTemplated() {}
    public function getRels() {}
    public function getAttributes() {}
}

?>
```

See also PSR-13 : Link definition interface.

Specs

Short name	Psr/Psr13Usage
Rulesets	*CE*
Exakt since	0.11.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.937 PSR-16 Usage

PSR-16 describes a simple yet extensible interface for a cache item and a cache driver. It is supported by an set of interfaces, that one may use in the code.

```
<?php

namespace My\SimpleCache;

// MyCache implements the PSR-16 Simple cache.
// MyCache is more of a black hole than a real cache.
class MyCache implements Psr\SimpleCache\CacheInterface {
    public function get($key, $default = null) {}
    public function set($key, $value, $ttl = null) {}
    public function delete($key) {}
    public function clear() {}
    public function getMultiple($keys, $default = null) {}
    public function setMultiple($values, $ttl = null) {}
    public function deleteMultiple($keys) {}
    public function has($key) {}
}

?>
```

See also PSR-16 : Common Interface for Caching Libraries.

Specs

Short name	Psr/Psr16Usage
Rulesets	*CE*
Exakt since	0.11.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.938 PSR-3 Usage

PSR-3 describes a common interface for logging libraries.

It is supported by an set of interfaces, that one may use in the code.

```php
<?php

namespace MyNamespace;

// MyLog implements the PSR-3 LoggerInterface.
// MyLog is more of a black hole than a real Log.
namespace ;

class MyLog implements \Psr\Log\LoggerInterface {
    public function emergency($message, array $context = array()) {}
    public function alert($message, array $context = array()) {}
    public function critical($message, array $context = array()) {}
    public function error($message, array $context = array()) {}
    public function warning($message, array $context = array()) {}
    public function notice($message, array $context = array()) {}
    public function info($message, array $context = array()) {}
    public function debug($message, array $context = array()) {}
    public function log($level, $message, array $context = array()) {}
}

?>
```

See also PSR-3 : Logger Interface.

Specs

Short name	Psr/Psr3Usage
Rulesets	*CE*
Exakt since	0.11.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.939 PSR-6 Usage

PSR-6 is the cache standard for PHP.

The goal of PSR-6 is to allow developers to create cache-aware libraries that can be integrated into existing frameworks and systems without the need for custom development.

It is supported by an set of interfaces, that one may use in the code.

```php
<?php

namespace MyNamespace;

// MyCacheItem implements the PSR-7 CacheItemInterface.
// This MyCacheItem is more of a black hole than a real CacheItem.
class MyCacheItem implements \Psr\Cache\CacheItemInterface {
    public function getKey() {}
    public function get() {}
    public function isHit() {}
    public function set($value) {}
```

(continues on next page)

(continued from previous page)

```
    public function expiresAt($expiration) {}
    public function expiresAfter($time) {}
}

?>
```

See also PSR-6 : Caching.

Specs

Short name	Psr/Psr6Usage
Rulesets	*CE*
Exakt since	0.11.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.940 PSR-7 Usage

> PSR-7 describes common interfaces for representing HTTP messages as described in RFC 7230 and RFC 7231, and URIs for use with HTTP messages as described in RFC 3986.

It is supported by an set of interfaces, that one may use in the code.

```
<?php

namespace MyNamespace;

// MyServerRequest implements the PSR-7 ServerRequestInterface.
// MyServerRequest is more of a black hole than a real Server.
class MyServerRequest extends  \Psr\Http\Message\ServerRequestInterface  {
    public function getServerParams() {}
    public function getCookieParams() {}
    public function withCookieParams(array $cookies) {}
    public function getQueryParams() {}
    public function withQueryParams(array $query) {}
    public function getUploadedFiles() {}
    public function withUploadedFiles(array $uploadedFiles) {}
    public function getParsedBody() {}
    public function withParsedBody($data) {}
    public function getAttributes() {}
    public function getAttribute($name, $default = null) {}
    public function withAttribute($name, $value) {}
    public function withoutAttribute($name) {}
}

?>
```

See also PSR-7 : HTTP message interfaces.

Specs

Short name	Psr/Psr7Usage
Rulesets	*CE*
Exakt since	0.11.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.941 Always Anchor Regex

Unanchored regex finds the requested pattern, and leaves room for malicious content.

Without ^ and $, the regex searches for any pattern that satisfies the criteria, leaving any unused part of the string available for arbitrary content. It is recommended to use both anchor

```
<?php

$birthday = getSomeDate($_GET);

// Permissive version : $birthday = '1970-01-01<script>xss();</script>';
if (!preg_match('/\d{4}-\d{2}-\d{2}/', $birthday) {
    error('Wrong data format for your birthday!');
}

// Restrictive version : $birthday = '1970-01-01';
if (!preg_match('/^\d{4}-\d{2}-\d{2}$/', $birthday) {
    error('Wrong data format for your birthday!');
}

echo 'Your birthday is on '.$birthday;

?>
```

Note that $ may be a line ending, still leaving room after it for injection.

```
<?php

$birthday = '1970-01-01'.PHP_EOL.'<script>xss();</script>';

?>
```

This analysis reports false positive when the regex is used to search a pattern in a much larger string. Check if this rule doesn't apply, though.

See also CWE-625: Permissive Regular Expression.

Suggestions

- Add an anchor to the beginning and ending of the string

Specs

Short name	Security/AnchorRegex
Rulesets	*Security*
Exakt since	0.12.15
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.942 Avoid Those Hash Functions

The following cryptography algorithms are considered insecure, and should be replaced with new and more performant algorithms.

`MD2`, `MD4`, `MD5`, `SHA0`, `SHA1`, `CRC`, `DES`, `3DES`, `RC2`, `RC4`.

When possible, avoid using them, may it be as PHP functions, or hashing function configurations (mcrypt, hash...).

```
<?php

// Weak cryptographic algorithm
echo md5('The quick brown fox jumped over the lazy dog.');

// Weak cryptographic algorthim, used with a modern PHP extension (easier to update)
echo hash('md5', 'The quick brown fox jumped over the lazy dog.');

// Strong cryptographic algorthim, used with a modern PHP extension
echo hash('sha156', 'The quick brown fox jumped over the lazy dog.');

?>
```

Weak cryptography is commonly used for hashing values when caching them. In such cases, security is not a primary concern. However, it may later become such, when hackers get access to the cache folders, or if the cached identifier is published. As a preventive protection, it is recommended to always use a secure hashing function.

See also Secure Hash Algorithms.

Suggestions

- Keep the current crypto, and add a call to a stronger one.
- Change the crypto for a more modern one and update the related databases

Specs

Short name	Security/AvoidThoseCrypto
Rulesets	*Security*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.943 Can't Disable Class

This is the list of potentially dangerous PHP class being used in the code, such as 'Phar <https://www.php.net/Phar>'_.

```
<?php

// This script uses ftp_connect(), therefore, this function shouldn't be disabled.
$phar = new Phar();

?>
```

This analysis is the base for suggesting values for the `disable_classes` directive.

Specs

Short name	Security/CantDisableClass
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.944 Can't Disable Function

This is the list of potentially dangerous PHP functions being used in the code, such as exec() or fsockopen().

eval() is not reported here, as it is not a PHP function, but a language construct : it can't be disabled.

```
<?php

// This script uses ftp_connect(), therefore, this function shouldn't be disabled.
$ftp = ftp_connect($host, 21);

// This script doesn't use imap_open(), therefore, this function may be disabled.

?>
```

This analysis is the base for suggesting values for the `disable_functions` directive.

Specs

Short name	Security/CantDisableFunction
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.945 Compare Hash

When comparing hash values, it is important to use the strict comparison : hash_equals(), === or !==.

In a number of situations, the hash value will start with `0e`, and PHP will understand that the comparison involves integers : it will then convert the strings into numbers, and it may end up converting them to 0.

Here is an example :

```
<?php

// The two following passwords hashes matches, while they are not the same.
$hashed_password = 0e462097431906509000000000000;
if (hash('md5','240610708',false) == $hashed_password) {
  print 'Matched.'.PHP_EOL;
}

// hash returns a string, that is mistaken with 0 by PHP
// The strength of the hashing algorithm is not a problem
if (hash('ripemd160','20583002034',false) == '0') {
  print 'Matched.'.PHP_EOL;
}

if (hash('md5','240610708',false) !== $hashed_password) {
  print 'NOT Matched.'.PHP_EOL;
}

// Display true
var_dump(md5('240610708') == md5('QNKCDZO') );

?>
```

You may also use password_hash() and password_verify() : they work together without integer conversion problems, and they can't be confused with a number.

See also Magic Hashes What is the best way to compare hashed strings? (PHP) and md5('240610708') == md5('QNKCDZO').

Suggestions

- Use dedicated functions for hash comparisons
- Use identity operators (===), and not equality operators (==) to compare hashes
- Compare hashes in the database (or external system), where such confusion is not possible

Specs

Short name	Security/CompareHash
Rulesets	*Security*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	strict-comparisons
Examples	*Traq*, *LiveZilla*

13.2.946 Configure Extract

The extract() function overwrites local variables when left unconfigured.

Extract imports variables from an array into the local scope. In case of a conflict, that is when a local variable already exists, it overwrites the previous variable.

In fact, extract() may be configured to handle the situation differently : it may skip the conflicting variable, prefix it, prefix it only if it exists, only import overwriting variables... It may also import them as references to the original values.

This analysis reports extract() when it is not configured explicitly. If overwriting is the intended objective, it is not reported.

```
<?php

// ignore overwriting variables
extract($array, EXTR_SKIP);

// prefix all variables explicitly variables with 'php_'
extract($array, EXTR_PREFIX_ALL, 'php_');

// overwrites explicitly variables
extract($array, EXTR_OVERWRITE);

// overwrites implicitely variables : do we really want that?
extract($array, EXTR_OVERWRITE);

?>
```

Always avoid using extract() on untrusted sources, such as `$_GET`, `$_POST`, `$_FILES`, or even databases records.

See also extract.

Suggestions

- Always use the second argument of extract(), and avoid using `EXTR_OVERWRITE`

Specs

Short name	Security/ConfigureExtract
Rulesets	*Security*
Exakt since	1.2.9
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Zurmo*, *Dolibarr*

13.2.947 Check Crypto Key Length

Each cryptography algorithm requires a reasonable length. Make sure an up-to-date length is used.

This rule use the following recommendations :

- 'OPENSSL_KEYTYPE_RSA <https://www.php.net/OPENSSL_KEYTYPE_RSA>'_' => 3072
- 'OPENSSL_KEYTYPE_DSA <https://www.php.net/OPENSSL_KEYTYPE_DSA>'_' => 2048
- 'OPENSSL_KEYTYPE_DH <https://www.php.net/OPENSSL_KEYTYPE_DH>'_' => 2048
- 'OPENSSL_KEYTYPE_EC <https://www.php.net/OPENSSL_KEYTYPE_EC>'_' => 512

The values above are used with the openssl PHP extension.

```
<?php

// Extracted from the documentation

// Generates a new and strong key
$private_key = openssl_pkey_new(array(
    private_key_type => OPENSSL_KEYTYPE_EC,
    private_key_bits => 1024,
));

// Generates a new and weak key
$private_key = openssl_pkey_new(array(
    private_key_type => OPENSSL_KEYTYPE_EC,
    private_key_bits => 256,
));

?>
```

See also The Definitive 2019 Guide to Cryptographic Key Sizes and Algorithm Recommendations and Cryptographic Key Length Recommendation.

Suggestions

- Lengthen the cryptographic key

Specs

Short name	Security/CryptoKeyLength
Rulesets	*Security*
Exakt since	2.1.1
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.948 Safe Curl Options

> It is advised to always use `CURLOPT_SSL_VERIFYPEER` and `CURLOPT_SSL_VERIFYHOST` when requesting a SSL connection.

With those tests, the certificate is verified, and if it isn't valid, the connection fails : this is a safe behavior.

```
<?php
$ch = curl_init();

curl_setopt($ch, CURLOPT_URL, https://www.php.net/);

// To be safe, always set this to true
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, true);

curl_exec($ch);
curl_close($ch);
?>
```

See also Don't turn off 'CURLOPT_SSL_VERIFYPEER, fix your PHP configuration <https://www.saotn.org/dont-turn-off-curlopt_ssl_verifypeer-fix-php-configuration/>'_, Certainty: Automated CACert.pem Management for PHP Software and Server-Side HTTPS Requests.

Suggestions

- Always use CURLOPT_SSL_VERIFYPEER and HTTPS for communication with other servers

Specs

Short name	Security/CurlOptions
Rulesets	*Security*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*OpenConf*

13.2.949 Direct Injection

The following code act directly upon PHP incoming variables like `$_GET` and `$_POST`. This makes those snippets very unsafe.

```
<?php

// Direct injection
echo Hello.$_GET['user']., welcome.;

// less direct injection
foo($_GET['user']);
function foo($user) {
    echo Hello.$user., welcome.;
}

?>
```

See also Cross-Site Scripting (XSS)

Suggestions

- Validate input : make sure the incoming data are what you expect from them.
- Escape output : prepare outgoing data for the next system to use.

Specs

Short name	Security/DirectInjection
Rulesets	*Security*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.950 Don't Echo Error

It is recommended to avoid displaying error messages directly to the browser.

PHP's uses the `display_errors` directive to control display of errors to the browser. This must be kept to `off` when in production.

```
<?php

// Inside a 'or' test
mysql_connect('localhost', $user, $pass) or die(mysql_error());

// Inside a if test
$result = pg_query( $db, $query );
if( !$result )
{
     echo Erreur SQL: . pg_error();
```

(continues on next page)

(continued from previous page)

```
    exit;
}

// Changing PHP configuration
ini_set('display_errors', 1);
// This is also a security error : 'false' means actually true.
ini_set('display_errors', 'false');

?>
```

Error messages should be logged, but not displayed.

See also Error reporting and List of php.ini directives.

Suggestions

- Remove any echo, print, printf() call built with error messages from an exception, or external source.

Specs

Short name	Security/DontEchoError
Rulesets	*Analyze*, *CI-checks*, *Security*
Exakt since	0.8.7
Php Version	All
Severity	Critical
Time To Fix	Instant (5 mins)
Precision	High
Examples	*ChurchCRM*, *Phpdocumentor*

13.2.951 Dynamic Library Loading

Loading a variable dynamically requires a lot of care in the preparation of the library name.

In case of injection in the variable, the dynamic loading of a library gives a lot of power to an intruder.

```
<?php

    // dynamically loading a library
     dl($library. PHP_SHLIB_SUFFIX);

    // dynamically loading ext/vips
     dl('vips.' . PHP_SHLIB_SUFFIX);

    // static loading ext/vips (unix only)
     dl('vips.so');

?>
```

See also dl.

Suggestions

- Use a switch structure, to make the dl() calls static.
- Avoid using dl() and make the needed extension always available in PHP binary.

Specs

Short name	Security/DynamicDl
Rulesets	*Security*
Exakt since	1.1.7
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.952 Encoded Simple Letters

Some simple letters are written in escape sequence.

Usually, escape sequences are made to encode unusual characters. Using escape sequences for simple characters, like letters or numbers is suspicious.

This analysis also detects Unicode codepoint with superfluous leading zeros.

```
<?php

// This escape sequence makes eval hard to spot
$a = ev\1011;
$a('php_info();');

// With a PHP 7.0 unicode code point sequence
$a = ev\u{000041}l;
$a('php_info();');

// With a PHP 5.0+ hexadecimal sequence
$a = ev\x411;
$a('php_info();');

?>
```

Suggestions

- Make all simple letter appear clearly
- Add comments about why this code is encoded

Specs

Short name	Security/EncodedLetters
Rulesets	*Security*
Exakt since	0.10.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Zurmo*

13.2.953 filter_input() As A Source

The filter_input() and filter_input_array() functions access directly to `$_GET`. They represent a source for external data just like `$_GET`, `$_POST`, etc.

The main feature of filter_input() is that it is already filtered. The main drawback is that `FILTER_FLAG_NONE` is the `none` filter, and that default configuration is *FILTER_UNSAFE_RAW*.

The filter extension keeps access to the incoming data, even after the super globals, such as `$_GET`, are unset.

```
<?php

// Removing $_GET
$_GET = [];

// with the default : FILTER_UNSAFE_RAW, this means XSS
echo filter_input(INPUT_GET, 'i');

// Same as above :
echo filter_var(_GET, 'i');

?>
```

Thanks to Frederic Bouchery for reporting this special case.

See also Data filtering.

Suggestions

- Use the classic $_GET, $_POST super globals, which are easier to audit.
- Use your framework's parameter access.

Specs

Short name	Security/FilterInputSource
Rulesets	*Security*
Exakt since	1.4.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.954 GPRC Aliases

The following variables are holding the content of $_GET, $_POST, $_REQUEST or $_COOKIE. They shouldn't be trusted, just like their original variables.

Specs

Short name	Security/GPRAliases
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.955 Indirect Injection

Look for injections through indirect usage for GPRC values ($_GET, $_POST, $_REQUEST, $_COOKIE).

```
<?php

$a = $_GET['a'];
echo $a;

function foo($b) {
    echo $b;
}
foo($_POST['c']);

?>
```

Suggestions

- Always validate incoming values before using them.

Specs

Short name	Security/IndirectInjection
Rulesets	*Security*
Exakt since	0.8.4
Php Version	All
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	High

13.2.956 Integer Conversion

Comparing incoming variables to integer may lead to injection.

When comparing a variable to an integer, PHP applies type juggling, and transform the variable in an integer too. When the value converts smoothly to an integer, this means the validation may pass and yet, the value may carry an injection.

```
<?php

// This is safe :
if ($_GET['x'] === 2) {
    echo $_GET['x'];
}

// Using (int) for validation and for display
if ((int) $_GET['x'] === 2) {
    echo (int) $_GET['x'];
}

// This is an injection
// '2 <script>' == 2, then echo will make the injection
if ($_GET['x'] == 2) {
    echo $_GET['x'];
}

// This is unsafe, as $_GET['x']  is tested as an integer, but echo'ed raw
if ((int) $_GET['x'] === 2) {
    echo $_GET['x'];
}

?>
```

This analysis spots situations where an incoming value is compared to an integer. The usage of the validated value is not analyzed further.

See also Type Juggling Authentication Bypass Vulnerability in CMS Made Simple, PHP STRING COMPARISON VULNERABILITIES and PHP Magic Tricks: Type Juggling.

Suggestions

- Add the typecasting to all read access to the incoming variable

Specs

Short name	Security/IntegerConversion
Rulesets	*Security*
Exakt since	1.7.7
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.957 Keep Files Access Restricted

Avoid using 0777 as file or directory mode. In particular, setting a file or a directory to 0777 (or universal read-write-execute) may lead to security vulnerabilities, as anything on the server may read, write and even execute

File mode may be changed using the chmod() function, or at directory creation, with mkdir().

```
<?php

file_put_contents($file, $content);

// this file is accessible to the current user, and to his group, for reading and␣
↪writing.
chmod($file, 0550);

// this file is accessible to everyone
chmod($file, 0777);

?>
```

By default, this analysis report universal access (0777). It is possible to make this analysis more restrictive, by providing more forbidden modes in the `filePrivileges` parameter. For example : `511,510,489`. Only use a decimal representation.

See also Security/MkdirDefault and Least Privilege Violation.

Suggestions

- Set the file mode to a level of restriction as low as possible.

Name	Default	Type	Description
filePrivileges	0777	string	List of forbidden file modes (comma separated).

Specs

Short name	Security/KeepFilesRestricted
Rulesets	*Security*
Exakt since	2.1.1
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.958 Minus One On Error

Some PHP native functions return -1 on error. They also return 1 in case of success, and 0 in case of failure. This leads to confusions.

In case the native function is used as a condition without explicit comparison, PHP type cast the return value to a boolean. In this case, -1 and 1 are both converted to true, and the condition applies. This means that an error situation is mistaken for a successful event.

```php
<?php

// Proper check of the return value
if (openssl_verify($data, $signature, $public) === 1) {
    $this->loginAsUser($user);
}

// if this call fails, it returns -1, and is confused with true
if (openssl_verify($data, $signature, $public)) {
    $this->loginAsUser($user);
}
?>
```

This analysis searches for if/then structures, ternary operators inside while() / do... 'while() <https://www.php.net/manual/en/control-structures.while.php>'_ loops.

See also Can you spot the vulnerability? (openssl_verify) and Incorrect Signature Verification.

Suggestions

- Compare explicitly the return value to 1

Specs

Short name	Security/MinusOneOnError
Rulesets	*Security*
Exakt since	1.8.0
Php Version	All
Severity	Critical
Time To Fix	Instant (5 mins)
Precision	High

13.2.959 Mkdir Default

mkdir() gives universal access to created folders, by default. It is recommended to gives limited set of rights (0755, 0700), or to explicitly set the rights to 0777.

```php
<?php

// By default, this dir is 777
mkdir('/path/to/dir');

// Explicitely, this is wanted. It may also be audited easily
mkdir('/path/to/dir', 0777);

// This dir is limited to the current user.
mkdir('/path/to/dir', 0700);

?>
```

See also Why 777 Folder Permissions are a Security Risk.

Suggestions

- Always use the lowest possible privileges on folders
- Don't use the PHP default : at least, make it explicit that the 'universal' rights are voluntary

Specs

Short name	Security/MkdirDefault
Rulesets	*Security*
Exakt since	0.12.2
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Mautic*, *OpenEMR*

13.2.960 move_uploaded_file Instead Of copy

Always use move_uploaded_file() with uploaded files. Avoid using copy or rename with uploaded file.

move_uploaded_file() checks to ensure that the file designated by filename is a valid upload file (meaning that it was uploaded via PHP's HTTP POST upload mechanism).

```
<?php

    // $a->file was filled with $_FILES at some point
    move_uploaded_file($a->file['tmp_name'], $target);

    // $a->file was filled with $_FILES at some point
    rename($a->file['tmp_name'], $target);

?>
```

See also move_uploaded_file and Uploading Files with PHP.

Suggestions

- Always use move_uploaded_file()
- Extract the needed information from the file, and leave it for PHP to remove without storage

Specs

Short name	Security/MoveUploadedFile
Rulesets	*Security*
Exakt since	1.3.2
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.961 No ENT_IGNORE

Certain characters have special significance in HTML, and should be represented by HTML entities if they are to preserve their meanings.

ENT_IGNORE is a configuration option for htmlspecialchars(), that ignore any needed character replacement. This mean the raw input will now be processed by PHP, or a target browser.

It is recommended to use the other configuration options : `ENT_COMPAT`, `ENT_QUOTES`, `ENT_NOQUOTES`, `ENT_SUBSTITUTE`, `ENT_DISALLOWED`, `ENT_HTML401`, `ENT_XML1`, `ENT_XHTML` or `ENT_HTML5`.

```
<?php

// This produces a valid HTML tag
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_IGNORE);
echo $new; // &lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;

// This produces a valid string, without any HTML special value
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
echo $new; // &lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;

?>
```

See also htmlspecialchars and Deletion of Code Points.

Suggestions

- Use of the the other options

Specs

Short name	Security/NoEntIgnore
Rulesets	*Security*
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.962 No Net For Xml Load

Simplexml and ext/DOM load all external entities from the web, by default. This is dangerous, in particular when loading unknown XML code.

Look at this XML code below : it is valid. It defines an entity `xxe`, that is filled with a file, read on the system and base64 encoded.:

```
&lt;!DOCTYPE replace [&lt;!ENTITY xxe SYSTEM "php://filter/convert.base64-encode/
↪resource=index.php"&gt; ]&gt;
<replace>&xxe;</replace>
```

This file could be processed with the following code : note, you can replace 'index.php' in the above entity by any valid filepath.

```
<?php
    $dom = new DOMDocument();
    $dom->loadXML($xml, LIBXML_NOENT | LIBXML_DTDLOAD);
    $info = simplexml_import_dom($dom);

    print base64_decode($info[0]);
?>
```

Here, PHP tries to load the XML file, finds the entity, then solves the entity by encoding a file called `index.php`. The source code of the file is not used as data in the XML file.

At that point, the example illustrates how a XXE works : by using the XML engine to load external resources, and preprocessing the XML code. in fact, there is only one change to make this XML code arbitrarily injected ::

```
&lt;!DOCTYPE replace [&lt;!ENTITY writer SYSTEM https://www.example.com/entities.dtd&
↪gt; ]&gt;
<replace>&xxe;</replace>
```

With the above example, the XML code is static (as, it never changes), but the 'xxe' definitions are loaded from a remove website, and are completely under the attacker control.

See also XML External Entity, XML External Entity (XXE) Processing and Detecting and exploiting XXE in SAML Interfaces.

Suggestions

- Strip out any entity when using external XML
- Forbid any network to the XML engine, by configuring the XML engine without network access

Specs

Short name	Security/NoNetForXmlLoad
Rulesets	*Security*
Exakt since	1.0.11
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.963 Avoid sleep()/usleep()

sleep() and usleep() help saturate the web server.

Pausing the script for a specific amount of time means that the Web server is also making all related resources sleep, such as database, sockets, session, etc. This may used to set up a DOS on the server.

```
<?php

$begin = microtime(true);
checkLogin($user, $password);
$end   = microtime(true);

```

(continues on next page)

(continued from previous page)

```
// Making all login checks looks the same
usleep(1000000 - ($end - $begin) * 1000000);

// Any hit on this page now uses 1 second, no matter if load is high or not
// Is it now possible to saturate the webserver in 1 s ?

?>
```

As much as possible, avoid delaying the end of the script.

sleep() and usleep() have less impact in commandline (`CLI`).

Suggestions

- Add a deadline of usage in the session, and wait past this deadline to start serving again. Until then, abort immediately.
- Use element in the GUI to delay or slow usage.

Specs

Short name	Security/NoSleep
Rulesets	*Security*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.964 No Weak SSL Crypto

When enabling PHP's stream SSL, it is important to use a safe protocol.

All the SSL protocols (1.0, 2.0, 3.0), and TLS (1.0 are unsafe. The best is to use the most recent TLS, version 1.2.

stream_socket_enable_crypto() and curl_setopt() are checked.

```
<?php

// This socket will use SSL v2, which
$socket = 'sslv2://www.example.com';
$fp = fsockopen($socket, 80, $errno, $errstr, 30);

?>
```

Using the TLS transport protocol of PHP will choose the version by itself.

See also Insecure Transportation Security Protocol Supported (TLS 1.0), The 2018 Guide to Building Secure PHP Software and Internet Domain: TCP, UDP, SSL, and TLS.

Suggestions

- Use TLS transport, with version 1.2

Specs

Short name	Security/NoWeakSSLCrypto
Rulesets	*Security*
Exakt since	1.9.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.965 parse_str() Warning

The parse_str() function parses a query string and assigns the resulting variables to the local scope. This may create a unexpected number of variables, and even overwrite the existing one.

```
<?php
  function foo( ) {
    global $a;

    echo $a;
  }

  parse_str('a=1'); // No second parameter
  foo( );
  // displays 1
?>
```

Always use an empty variable a second parameter to parse_str(), so as to collect the incoming values, and then, filter them in that array.

Suggestions

- Use the second parameter when calling parse_url();
- Change to PHP 8.0 version, which made the second argument compulsory

Specs

Short name	Security/parseUrlWithoutParameters
Rulesets	*Security*
Exakt since	0.8.4
Php Version	8.0-
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
ClearPHP	know-your-variables

13.2.966 Register Globals

`register_globals` was a PHP directive that dumped all incoming variables from GET, POST, COOKIE and FILES as global variables in the called scripts.

This lead to security failures, as the variables were often used but not filtered.

Though it is less often found in more recent code, `register_globals` is sometimes needed in legacy code, that haven't made the move to eradicate this style of coding. Backward compatible pieces of code that mimic the `register_globals` features usually create even greater security risks by being run after scripts startup. At that point, some important variables are already set, and may be overwritten by the incoming call, creating confusion in the script.

Mimicking `register_globals` is achieved with variables variables, extract(), parse_str() and import_request_variables() (Up to PHP 5.4).

```
<?php

// Security warning ! This overwrites existing variables.
extract($_POST);

// Security warning ! This overwrites existing variables.
foreach($_REQUEST as $var => $value) {
    $$var = $value;
}

?>
```

Suggestions

- Avoid reimplementing register_globals
- Use a container to store and access commonly used values

Specs

Short name	Security/RegisterGlobals
Rulesets	*Security*
Exakt since	0.8.4
Php Version	All
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	High
Examples	*TeamPass*, *XOOPS*

13.2.967 Safe HTTP Headers

Avoid configuring HTTP headers with lax restriction from within PHP.

There are a lot of HTTP headers those days, targeting various vulnerabilities. To ensure backward compatibility, those headers have a default mode that is lax and permissive. It is recommended to avoid using those from within the code.

```
<?php

//Good configuration, limiting access to origin
header('Access-Control-Allow-Origin: https://www.exakat.io');

//Configuration is present, but doesn't restrict anything : any external site is a␣
↪potential source
header('Access-Control-Allow-Origin: *');

?>
```

See also Hardening Your HTTP Security Headers, How To Secure Your Web App With HTTP Headers and Security-Headers.

Suggestions

- Remove usage of those headers

Specs

Short name	Security/SafeHttpHeaders
Rulesets	*Security*
Exakt since	1.5.5
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.968 Sensitive Argument

Spot the argument that are sensitive for security. The functioncalls that are hosting a sensitive argument are called a sink.

```
<?php

// first argument $query is a sensitive argument
mysqli_query($query);

?>
```

Specs

Short name	Security/SensitiveArgument
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.969 Session Lazy Write

Classes that implements SessionHandlerInterface must also implements SessionUpdateTimestampHandlerInterface.

The two extra methods are used to help lazy loading : the first actually checks if a sessionId is available, and the seconds updates the time of last usage of the session data in the session storage.

This was spotted by `Nicolas Grekas`, and fixed in Symfony [HttpFoundation] Make sessions secure and lazy #24523.

```
<?php

interface SessionUpdateTimestampHandlerInterface {
    // returns a boolean to indicate that valid data is available for this sessionId,␣
↪or not.
    function validateId($sessionId);

    //called to change the last time of usage for the session data.
    //It may be a file's touch or full write, or a simple update on the database
    function updateTimestamp($sessionId, $sessionData);
}

?>
```

See also Sessions: Improve original RFC about lazy_write and the Sessions.

Suggestions

- Implements the SessionUpdateTimestampHandlerInterface interface

Specs

Short name	Security/SessionLazyWrite
Rulesets	*Security*
Exakt since	0.12.15
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.970 Set Cookie Safe Arguments

The last five arguments of setcookie() and setrawcookie() are for security. Use them anytime you can.

```
setcookie ( string $name [, string $value = [, int $expire = 0 [, string $path
= [, string $domain = [, bool $secure = false [, bool $httponly = false ]]]]]]
)
```

The `$expire` argument sets the date of expiration of the cookie. It is recommended to make it as low as possible, to reduce its chances to be captured. Sometimes, low expiration date may be several days (for preferences), and other times, low expiration date means a few minutes.

The `$path` argument limits the transmission of the cookie to URL whose path matches the one mentioned here. By default, it is `'/'`, which means the whole server. If a cookie usage is limited to a part of the application, use it here.

The `$domain` argument limits the transmission of the cookie to URL whose domain matches the one mentioned here. By default, it is `''`, which means any server on the internet. At worse, you may use `mydomain.com` to cover your whole domain, or better, refine it with the actual subdomain of usage.

The `$secure` argument limits the transmission of the cookie over HTTP (by default) or HTTPS. The second is better, as the transmission of the cookie is crypted. In case HTTPS is still at the planned stage, use '$_SERVER[HTTPS]'. This environment variable is false on HTTP, and true on HTTPS.

The `$httponly` argument limits the access of the cookie to JavaScript. It is only transmitted to the browser, and retransmitted. This helps reducing XSS and CSRF attacks, though it is disputed.

The `$samesite` argument limits the sending of the cookie to the domain that initiated the request. It is by default `Lax` but should be upgraded to `Strict` whenever possible. This feature is available as PHP 7.3.

```
<?php

//admin cookie, available only on https://admin.my-domain.com/system/, for the next␣
↪minute, and not readable by javascript
setcookie(admin, $login, time()+60, /system/, admin.my-domain.com, $_SERVER['HTTPS'],␣
↪1);

//login cookie, available until the browser is closed, over http or https
setcookie(login, $login);

//removing the login cookie : Those situations are omitted by the analysis
setcookie(login, '');

?>
```

See also setcookie and 'SameSite' cookie attribute.

Suggestions

- Use all the argument when setting cookies with PHP functions

Specs

Short name	Security/SetCookieArgs
Rulesets	*Security*
Exakt since	0.10.6
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.971 Should Use Prepared Statement

Modern databases provides support for prepared statement : it separates the query from the processed data and raise significantly the security.

Building queries with concatenations is not recommended, though not always avoidable. When possible, use prepared statements.

```
<?php
/* Execute a prepared statement by passing an array of values */

$sql = 'SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour';
$sth = $conn->prepare($sql, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
$sth->execute(array(':calories' => 150, ':colour' => 'red'));
$red = $sth->fetchAll();
?>
```

Same code, without preparation :

```
<?php

    $sql = 'SELECT name, color, calories FROM fruit WHERE calories < '.$conn-
↪quote(150).' AND colour = '.$conn->quotes('red').' ORDER BY name';
    $sth = $conn->query($sql) as $row);
}
?>
```

See also Prepared Statements <https://www.php.net/manual/en/'mysqli.quickstart.prepared-statements.php>'_, PHP 'MySQLi Prepared Statements Tutorial to Prevent SQL Injection <https://websitebeaver.com/prepared-statements-in-php-mysqli-to-prevent-sql-injection>'_, The Best Way to Perform 'MYSQLI Prepared Statements in PHP <https://developer.hyvor.com/php/prepared-statements>'_.

Suggestions

- Use an ORM
- Use an Active Record library
- Change the query to hard code it and make it not injectable

Name	Default	Type	Description
queryMethod	query_methods.json	data	Methods that call a query.

Specs

Short name	Security/ShouldUsePreparedStatement
Rulesets	*Analyze*, *CI-checks*, *Security*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Dolibarr*

13.2.972 Should Use session_regenerateid()

session_regenerateid() should be used when sessions are used.

When using sessions, a session ID is assigned to the user. It is a random number, used to connect the user and its data on the server. Actually, anyone with the session ID may have access to the data. This is why those session ID are so long and complex.

A good approach to protect the session ID is to reduce its lifespan : the shorter the time of use, the better. While changing the session ID at every hit on the page may no be possible, a more reasonable approach is to change the session id when an important action is about to take place. What important means is left to the application to decide.

Based on this philosophy, a code source that uses ZendSession but never uses ZendSession::regenerateId() has to be updated.

```
<?php

    session_start();

    $id = (int) $_SESSION['id'];
    // no usage of session_regenerateid() anywhere triggers the analysis

    // basic regeneration every 20 hits on the page.
    if (++$_SESSION['count'] > 20) {
        session_regenerateid();
    }

?>
```

See session_regenerateid() and PHP Security Guide: Sessions.

Suggestions

- Add session_regenerateid() call before any important operation on the application

Specs

Short name	Security/ShouldUseSessionRegenerateId
Rulesets	*Security*
Exakt since	0.10.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.973 Sqlite3 Requires Single Quotes

The escapeString() method from `SQLite3` doesn't escape `"`, but only `'`.

```
<?php

// OK. escapeString is OK with '
$query = "SELECT * FROM table WHERE col = '".$sqlite->escapeString($x)."'";
```

(continues on next page)

(continued from previous page)

```
// This is vulnerable to " in $x
$query = 'SELECT * FROM table WHERE col = "'.$sqlite->escapeString($x).'"';

?>
```

To properly handle quotes and NUL characters, use bindParam() instead.

Quote from the PHP manual comments : `The reason this function doesn't escape double quotes is because double quotes are used with names (the equivalent of backticks in MySQL), as in table or column names, while single quotes are used for values.`

See also SQLite3::escapeString.

Suggestions

- Use prepared statements whenever possible
- Switch the query to use single quote

Specs

Short name	Security/Sqlite3RequiresSingleQuotes
Rulesets	*Security*
Exakt since	1.0.10
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.974 Super Globals Contagion

Basic tainting system.

Specs

Short name	Security/SuperGlobalContagion
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.975 Unserialize Second Arg

Since PHP 7, unserialize() function has a second argument that limits the classes that may be unserialized. In case of a breach, this is limiting the classes accessible from unserialize().

One way to exploit unserialize, is to make PHP unserialized the data to an available class, may be one that may be auto-loaded.

```
<?php

// safe unserialization : only the expected class will be extracted
$serialized = 'O:7:dbClass:0:{}';
$var = unserialize($serialized, ['dbClass']);
$var->connect();

// unsafe unserialization : $var may be of any type that was in the serialized string
// although, here, this is working well.
$serialized = 'O:7:dbClass:0:{}';
$var = unserialize($serialized);
$var->connect();

// unsafe unserialization : $var is not of the expected type.
// and, here, this will lead to disaster.
$serialized = 'O:10:debugClass:0:{}';
$var = unserialize($serialized);
$var->connect();

?>
```

See also unserialize(), Securely Implementing (De)Serialization in PHP, and Remote code execution via PHP [Unserialize].

Suggestions

- Add a list of class as second argument of any call to unserialize(). This is valid for PHP 7.0 and later.

Specs

Short name	Security/UnserializeSecondArg
Rulesets	*Security*
Exakt since	0.8.4
Php Version	7.0+
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Piwigo*, *LiveZilla*

13.2.976 Upload Filename Injection

When receiving a file via Upload, it is recommended to store it under a self-generated name. Any storage that uses the original filename, or even a part of it may be vulnerable to injections.

```
<?php

// Security error ! the $_FILES['upload']['filename'] is provided by the sender.
// 'a.<script>alert(\'a\')</script>'; may lead to a HTML injection.
$extension = substr( strrchr($_FILES['upload']['name'], '.') ,1);
```

(continues on next page)

(continued from previous page)

```
if (!in_array($extension, array('gif', 'jpeg', 'jpg')) {
    // process error
    continue;
}
// Md5 provides a name without special characters
$name = md5($_FILES['upload']['filename']);
if(@move_uploaded_file($_FILES['upload']['tmp_name'], '/var/no-www/upload/'.$name.'.'.
↪$extension)) {
    safeStoring($name.'.'.$extension, $_FILES['upload']['filename']);
}

// Security error ! the $_FILES['upload']['filename'] is provided by the sender.
if(@move_uploaded_file($_FILES['upload']['tmp_name'], $_FILES['upload']['filename']))
↪{
    safeStoring($_FILES['upload']['filename']);
}

// Security error ! the $_FILES['upload']['filename'] is provided by the sender.
// 'a.<script>alert('a')</script>'; may lead to a HTML injection.
$extension = substr( strrchr($_FILES['upload']['name'], '.') ,1);
$name = md5($_FILES['upload']['filename']);
if(@move_uploaded_file($_FILES['upload']['tmp_name'], $name.'.'.$extension)) {
    safeStoring($name.'.'.$extension, $_FILES['upload']['filename']);
}

?>
```

It is highly recommended to validate any incoming file, generate a name for it, and store the result in a folder outside the web folder. Also, avoid accepting PHP scripts, if possible.

See also [CVE-2017-6090], CWE-616: Incomplete Identification of Uploaded File Variables, Why File Upload Forms are a Major Security Threat.

Suggestions

- Validate uploaded filenames
- Rename files upon storage, and keep the original name in a database

Specs

Short name	Security/UploadFilenameInjection
Rulesets	*Security*
Exakt since	0.12.14
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.977 Adding Zero

Adding 0 is useless, as 0 is the neutral element for addition. Besides, when one of the argument is an integer, PHP triggers a cast to integer.

It is recommended to make the cast explicit with `(int)`.

```
<?php

// Explicit cast
$a = (int) foo();

// Useless addition
$a = foo() + 0;
$a = 0 + foo();

// Also works with minus
$b = 0 - $c; // drop the 0, but keep the minus
$b = $c - 0; // drop the 0 and the minus

$a += 0;
$a -= 0;

?>
```

Adding zero is also reported when the zero is a defined constants.

If it is used to type cast a value to integer, then casting with `(int)` is clearer.

Suggestions

- Remove the +/- 0, may be the whole assignation
- Use an explicit type casting operator (int)

Specs

Short name	Structures/AddZero
Rulesets	*Analyze*, *CE*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-useless-math
Examples	*Thelia*, *OpenEMR*

13.2.978 Altering Foreach Without Reference

Foreach() loop that should use a reference.

When using a foreach loop that modifies the original source, it is recommended to use referenced variables, rather than access the original value with $source[$index].

Using references is then must faster, and easier to read.

```
<?php

// Using references in foreach
foreach($source as $key => &$value) {
    $value = newValue($value, $key);
}

// Avoid foreach : use array_map
$source = array_walk($source, 'newValue');
    // Here, $key MUST be the second argument or newValue

// Slow version to update the array
foreach($source as $key => &$value) {
    $source[$key] = newValue($value, $key);
}
?>
```

You may also use array_walk() or array_map() (when $key is not used) to avoid the use of foreach.

See also foreach.

Suggestions

- Add the reference on the modified blind variable, and avoid accessing the source array

Specs

Short name	Structures/AlteringForeachWithoutReference
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	use-reference-to-alter-in-foreach
Examples	*Contao*, *WordPress*

13.2.979 Alternative Syntax Consistence

PHP allows for two syntax : the alternative syntax, and the classic syntax.

The classic syntax is almost always used. When used, the alternative syntax is used in templates.

This analysis reports files that are using both syntax at the same time. This is confusing.

```
<?php

// Mixing both syntax is confusing.
foreach($array as $item) :
    if ($item > 1) {
        print $item elements\n;
    } else {
        print $item element\n;
```

(continues on next page)

(continued from previous page)

```
    }
endforeach;

?>
```

Specs

Short name	Structures/AlternativeConsistenceByFile
Rulesets	*Analyze*
Exakt since	0.11.2
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.980 Comparison Is Always True

Based on the incoming types of arguments, the comparison never change.

```
<?php

function foo(array $a) {
    // This will always fail
    if ($a === 1) {

    } elseif (is_int($a)) {

    }

    // This will always succeed
    if ($a !== null) {

    } elseif (is_null($a)) {

    }
}

?>
```

Suggestions

- Remove the constant condition and its corresponding blocks
- Make the constant condition variable

Specs

Short name	Structures/AlwaysFalse
Rulesets	*Analyze*
Exakt since	1.9.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.981 Array_Fill() With Objects

array_fill() fills an array with identical objects, not copies nor clones. This means that all the filled objects are a reference to the same object. Changing one of them will change any of them.

Make sure this is the intended effect in the code.

```
<?php

$x = new StdClass();
$array = array_fill(0, 10, $x);

$array[3]->y = Set in object #3;

// displays Set in object #3;
echo $array[5]->y;

?>
```

This applies to array_pad() too. It doesn't apply to array_fill_keys(), as objects will be cast to a string before usage in this case.

Suggestions

- Use a loop to fill in the array with cloned() objects.

Specs

Short name	Structures/ArrayFillWithObjects
Rulesets	*Analyze*
Exakt since	2.1.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.982 Array_Map() Passes By Value

array_map() requires the callback to receive elements by value. Unlink array_walk(), which accepts by value or by reference, depending on the action taken.

PHP 8.0 and more recent emits a Warning

```
<?php
// Example, courtery of Juliette Reinders Folmer
function trimNewlines(&$line, $key) {
    $line = str_replace(array(\n, \r), '', $line);
}

$original = [
    text\n\n,
    text\n\r
];

$array = $original;
array_walk($array, 'trimNewlines');

var_dump($array);

array_map('trimNewlines', $original, [0, 1]);

?>
```

See also array_map.

Suggestions

- Make the callback first argument a reference

Specs

Short name	Structures/ArrayMapPassesByValue
Rulesets	*Analyze*, *CE*, *CompatibilityPHP80*
Exakt since	2.2.0
Php Version	7.4-
Severity	Major
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.983 array_merge() And Variadic

Always check value in variadic before using it with array_merge() and array_merge_recursive().

Before PHP 7.4, array_merge() and array_merge_recursive() would complain when no argument was provided. As such, using the spread operator ... on an empty array() would yield no argument, and an error.

```
<?php

//
$b = array_merge(...$x);



?>
```

Suggestions

- Add a check to the spread variable to ensure it is not empty
- Append an empty array to to the spread variable to ensure it is not empty

Specs

Short name	Structures/ArrayMergeAndVariadic
Rulesets	*Analyze*
Exakt since	1.9.2
Php Version	7.4-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.984 Array_merge Needs Array Of Arrays

> When collecting data to feed array_merge(), use an array of array as default value. `array(`array()) <https://www.php.net/array>`_` is the neutral value for array_merge();

This analysis also reports when the used types are not an array : array_merge() does not accept scalar values, but only arrays.

```
<?php

// safe default value
$a = array(array());

// when $list is empty, it is
foreach($list as $l) {
    $a[] = $l;
}
$b = array_merge($a);

?>
```

Since PHP 7.4, it is possible to call array_merge() without an argument : this means the default value may an empty array. This array shall not contain scalar values.

See also array_merge.

Suggestions

- Use `array(array())` or `[[]]` as default value for array_merge()
- Remove any non-array value from the values in the default array

Specs

Short name	Structures/ArrayMergeArrayArray
Rulesets	*Analyze*
Exakt since	2.1.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.985 Searching For Multiple Keys

array_search() and array_keys() find keys in an array. array_search() returns the first key that match a value, while array_keys() returns all the keys that match a value.

array_search() and array_keys() both accepts a final parameter to set a strict search or not.

```
<?php

$array = array(0,1,2,3,4,3);

// $id = 3
$id = array_search($array, 3);

// $ids = [3, 5];
$ids = array_keys($array, 3);

?>
```

Suggestions

- Use array_keys() to find multiple keys in an array
- Use array_keys() to find a unique key in an array

Specs

Short name	Structures/ArraySearchMultipleKeys
Rulesets	*Suggestions*
Exakt since	2.2.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Unknown

13.2.986 Assign And Compare

Assignation has a lower precedence than comparison. As such, the assignation always happens after the comparison. This leads to the comparison being stored in the variable, and not the value being compared.

```
<?php

if ($id = strpos($string, $needle) !== false) {
    // $id now contains a boolean (true or false), but not the position of the
↪$needle.
}

// probably valid comparison, as $found will end up being a boolean
if ($found = strpos($string, $needle) === false) {
    doSomething();
}

// always valid comparison, with parenthesis
if (($id = strpos($string, $needle)) !== false) {
    // $id now contains a boolean (true or false), but not the position of the
↪$needle.
}

// Being a lone instruction, this is always valid : there is no double usage with if␣
↪condition
$isFound = strpos($string, $needle) !== false;


?>
```

See also Operator Precedence.

Suggestions

- Use parenthesis
- Separate assignation and comparison
- Drop assignation or comparison

Specs

Short name	Structures/AssigneAndCompare
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.6.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.987 Assigned In One Branch

Report variables that are assigned in one branch, and not in the other.

```
<?php

if ($condition) {
    // $assigned_in_this_branch is assigned in only one of the branches
```

(continues on next page)

(continued from previous page)

```
    $assigned_in_this_branch = 1;
    $also_assigned = 1;
} else {
    // $also_assigned is assigned in the two branches
    $also_assigned = 1;
}

?>
```

Specs

Short name	Structures/AssignedInOneBranch
Rulesets	none
Exakt since	1.0.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.988 Same Variable Foreach

A foreach which uses its own source as a blind variable is actually broken.

Actually, PHP makes a copy of the source before it starts the loop. As such, the same variable may be used for both source and blind value.

Of course, this is very confusing, to see the same variables used in very different ways.

The source will also be destroyed immediately after the blind variable has been turned into a reference.

```
<?php

$array = range(0, 10);
foreach($array as $array) {
    print $array.PHP_EOL;
}

print_r($array); // display number from 0 to 10.

$array = range(0, 10);
foreach($array as &$array) {
    print $array.PHP_EOL;
}

print_r($array); // display 10

?>
```

Specs

Short name	Structures/AutoUnsetForeach
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.0.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.989 Bail Out Early

When using conditions, it is recommended to quit in the current context, and avoid else clause altogether.

The main benefit is to make clear the method applies a condition, and stop immediately when it is not satisfied. The main sequence is then focused on the actual code.

This works with the `break`, `continue`, `throw` and `goto` keywords too, depending on situations.

```
<?php

// Bailing out early, low level of indentation
function foo1($a) {
    if ($a > 0) {
        return false;
    }

    $a++;
    return $a;
}

// Works with continue too
foreach($array as $a => $b) {
    if ($a > 0) {
        continue false;
    }

    $a++;
    return $a;
}

// No need for else
function foo2($a) {
    if ($a > 0) {
        return false;
    } else {
        $a++;
    }

    return $a;
}

// No need for else : return goes into then.
function foo3($a) {
    if ($a < 0) {
```

(continues on next page)

(continued from previous page)

```
        $a++;
    } else {
        return false;
    }

    return $a;
}

// Make a return early, and make the condition visible.
function foo3($a) {
    if ($a < 0) {
        $a++;
        methodcall();
        functioncall();
    }
}

?>
```

See also Avoid nesting too deeply and return early (part 1) and Avoid nesting too deeply and return early (part 2).

Suggestions

- Detect errors, and then, return as soon as possible.
- When a if... then branches are unbalanced, test for the small branch, finish it with return. Then keep the other branch as the main code.

Specs

Short name	Structures/BailOutEarly
Rulesets	*Analyze*
Exakt since	0.8.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*OpenEMR*, *opencfp*

13.2.990 Use Basename Suffix

basename() will remove extension when it is provided as argument. The second argument will be removed from the name of the file.

```
<?php

$path = 'phar:///path/to/file.php';

// Don't forget the .
$filename = basename($path, '.php');

```

(continues on next page)

(continued from previous page)

```
// Too much work for this
$filename = substr(basename($path), 0, -4);

?>
```

Using basename() instead of substr() or else, makes the intention clear.

See also basename.

Suggestions

- Use basename(), remove more complex code based on substr() or str_replace()

Specs

Short name	Structures/BasenameSuffix
Rulesets	*Suggestions*
Exakt since	1.5.1
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*NextCloud*, *Dolibarr*

13.2.991 Strict Comparison With Booleans

Strict comparisons prevent from mistaking an error with a false.

Boolean values may be easily mistaken with other values, especially when the function may return integer or boolean as a normal course of action.

It is encouraged to use strict comparison === or !== when booleans are involved in a comparison.

```
<?php

// distinguish between : $b isn't in $a, and, $b is at the beginning of $a
if (strpos($a, $b) === 0) {
    doSomething();
}

// DOES NOT distinguish between : $b isn't in $a, and, $b is at the beginning of $a
if (strpos($a, $b)) {
    doSomething();
}

// will NOT mistake 1 and true
$a = array(0, 1, 2, true);
if (in_array($a, true, true)) {
    doSomething();
}

// will mistake 1 and true
```

(continues on next page)

(continued from previous page)

```
$a = array(0, 1, 2, true);
if (in_array($a, true)) {
    doSomething();
}

?>
```

switch() structures always uses == comparisons.

Native function in_array() has a third parameter to make it use strict comparisons.

Suggestions

- Use strict comparison whenever possible

Specs

Short name	Structures/BooleanStrictComparison
Rulesets	*Analyze*, *CI-checks*, *Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Phinx*, *Typo3*

13.2.992 Bracketless Blocks

PHP allows one liners as for(), foreach(), while(), do/while() loops, or as then/else expressions.

It is generally considered a bad practice, as readability is lower and there are non-negligible risk of excluding from the loop the next instruction.

```
<?php

// Legit one liner
foreach(range('a', 'z') as $letter) ++$letterCount;

// More readable version, even for a one liner.
foreach(range('a', 'z') as $letter) {
    ++$letterCount;
}

?>
```

switch() cannot be without bracket.

Specs

Short name	Structures/Bracketless
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.993 Break With 0

Cannot break 0, as this makes no sense. Break 1 is the minimum, and is the default value.

```
<?php
    // Can't break 0. Must be 1 or more, depending on the level of nesting.
    for($i = 0; $i < 10; $i++) {
        break 0;
    }

    for($i = 0; $i < 10; $i++) {
        for($j = 0; $j < 10; $j++) {
            break 2;
        }
    }

?>
```

Suggestions

- Remove 0, or the break

Specs

Short name	Structures/Break0
Rulesets	*CompatibilityPHP53*
Exakt since	0.8.4
Php Version	5.4-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.994 Break With Non Integer

When using a break, the argument of the operator must be a positive non-null integer literal or be omitted.

Other values were acceptable in PHP 5.3 and previous version, but this is now reported as an error.

```
<?php
    // Can't break $a, even if it contains an integer.
    $a = 1;
    for($i = 0; $i < 10; $i++) {
        break $a;
    }

    // can't break on float
    for($i = 0; $i < 10; $i++) {
        for($j = 0; $j < 10; $j++) {
            break 2.2;
        }
    }

?>
```

Specs

Short name	Structures/BreakNonInteger
Rulesets	*CompatibilityPHP54*
Exakt since	0.8.4
Php Version	5.4-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.995 Break Outside Loop

Starting with PHP 7, break or continue that are outside a loop (for, foreach(), do… 'while() <https://www.php.net/manual/en/control-structures.while.php>'_, while()) or a switch() statement won't compile anymore.

It is not possible anymore to include a piece of code inside a loop that will then break.

```
<?php

    // outside a loop : This won't compile
    break 1;

    foreach($array as $a) {
        break 1; // Compile OK

        break 2; // This won't compile, as this break is in one loop, and not 2
    }

    foreach($array as $a) {
        foreach($array2 as $a2) {
            break 2; // OK in PHP 5 and 7
        }
    }
?>
```

Specs

Short name	Structures/BreakOutsideLoop
Rulesets	*Analyze*, *CompatibilityPHP70*
Exakt since	0.8.4
Php Version	7.0-
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.996 Buried Assignation

Those assignations are buried in the code, and placed in unexpected situations.

They are difficult to spot, and may be confusing. It is advised to place them in a more visible place.

```
<?php

// $b may be assigned before processing $a
$a = $c && ($b = 2);

// Display property p immeiately, but also, keeps the object for later
echo ($o = new x)->p;

// legit syntax, but the double assignation is not obvious.
for($i = 2, $j = 3; $j < 10; $j++) {

}
?>
```

Suggestions

- Extract the assignation and set it on its own line, prior to the current expression.
- Check if the local variable is necessary

Specs

Short name	Structures/BuriedAssignation
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*XOOPS*, *Mautic*

13.2.997 Calltime Pass By Reference

PHP doesn't allow when a value is turned into a reference at functioncall, since PHP 5.4.

Either the function use a reference in its signature, either the reference won't pass.

```
<?php

function foo($name) {
    $arg = ucfirst(strtolower($name));
    echo 'Hello '.$arg;
}

$a = 'name';
foo(&$a);

?>
```

Suggestions

- Make the signature of the called method accept references
- Remove the reference from the method call
- Use an object instead of a scalar

Specs

Short name	Structures/CalltimePassByReference
Rulesets	*CompatibilityPHP54*
Exakt since	0.8.4
Php Version	5.4-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.998 Can't Count Non-Countable

Count() emits an error when it tries to count scalars or objects what don't implement Countable interface.

```
<?php

// Normal usage
$a = array(1,2,3,4);
echo count($a).items\n;

// Error emiting usage
$a = '1234';
echo count($a).chars\n;

// Error emiting usage
echo count($unsetVar).elements\n;

?>
```

See also Warn when counting non-countable types.

Suggestions

- Add a check before using count such as a type check

Specs

Short name	Structures/CanCountNonCountable
Rulesets	*CompatibilityPHP72*
Exakt since	1.0.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.999 Casting Ternary

Type casting has a precedence over ternary operator, and is applied first. When this happens, the condition is cast, although it is often useless as PHP will do it if needed.

This applies to the ternary operator, the coalesce operator ?: and the null-coalesce operator ??.

```
<?php
    $a = (string) $b ? 3 : 4;
    $a = (string) $b ?: 4;
    $a = (string) $b ?? 4;
?>
```

The last example generates first an error *Undefined variable: b*, since $b is first cast to a string. The result is then an empty string, which leads to an empty string to be stored into $a. Multiple errors cascade.

See also Operators Precedence.

Suggestions

- Add parenthesis around the ternary operator
- Skip the casting
- Cast in another expression

Specs

Short name	Structures/CastingTernary
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.8.0
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1000 Cast To Boolean

This expression may be reduced by casting to boolean type.

```
<?php

$variable = $condition == 'met' ? 1 : 0;
// Same as
$variable = (bool) $condition == 'met';

$variable = $condition == 'met' ? 0 : 1;
// Same as (Note the condition inversion)
$variable = (bool) $condition != 'met';
// also, with an indentical condition
$variable = !(bool) $condition == 'met';

// This also works with straight booleans expressions
$variable = $condition == 'met' ? true : false;
// Same as
$variable = $condition == 'met';

?>
```

Suggestions

- Remove the old expression and use `(bool)` operator instead
- Change the target values from true/false, or 0/1 to non-binary values, like strings or integers beyond 0 and 1.
- Complete the current branches with other commands

Specs

Short name	Structures/CastToBoolean
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*MediaWiki*, *Dolibarr*

13.2.1001 Catch Overwrite Variable

The try/catch structure uses some variables that are also in use in this scope. In case of a caught exception, the exception will be put in the catch variable, and overwrite the current value, loosing some data.

```
<?php

// variables and caught exceptions are distinct
$argument = 1;
try {
```

(continues on next page)

(continued from previous page)

```
    methodThatMayRaiseException($argument);
} (Exception $e) {
    // here, $e has been changed to an exception.
}

// variables and caught exceptions are overlapping
$e = 1;
try {
    methodThatMayRaiseException();
} (Exception $e) {
    // here, $e has been changed to an exception.
}

?>
```

It is recommended to use another name for these catch variables.

Suggestions

- Use a standard : only use $e (or else) to catch exceptions. Avoid using them for anything else, parameter, property or local variable.
- Change the variable, and keep the caught exception

Specs

Short name	Structures/CatchShadowsVariable
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-catch-overwrite
Examples	*PhpIPAM*, *SuiteCrm*

13.2.1002 Check All Types

When checking for time, avoid using else. Mention explicitly all tested type, and raise an exception when reaching else.

PHP has a short list of scalar types : null, boolean, integer, real, strings, object, resource and array. When a variable is not holding one the the type, then it may be of any other type.

Most of the time, when using a simple is_string() / else test, this is relying on the conception of the code. By construction, the arguments may be one of two types : array or string.

What happens often is that in case of failure in the code (database not working, another class not checking its results), a third type is pushed to the structure, and it ends up breaking the execution.

The safe way is to check the various types all the time, and use the default case (here, the else) to throw exception() or test an assertion and handle the special case.

```php
<?php

// hasty version
if (is_array($argument)) {
    $out = $argument;
} else {
    // Here, $argument is NOT an array. What if it is an object ? or a NULL ?
    $out = array($argument);
}

// Safe type checking : do not assume that 'not an array' means that it is the other␣
↪expected type.
if (is_array($argument)) {
    $out = $argument;
} elseif (is_string($argument)) {
    $out = array($argument);
} else {
    assert(false, '$argument is not an array nor a string, as expected!');
}

?>
```

Using is_callable(), is_iterable() with this structure is fine : when variable is callable or not, while a variable is an integer or else.

Using a type test without else is also accepted here. This is a special treatment for this test, and all others are ignored. This aspect may vary depending on situations and projects.

Suggestions

- Include a default case to handle all unknown situations
- Include and process explicit types as much as possible

Specs

Short name	Structures/CheckAllTypes
Rulesets	*Analyze*
Exakt since	0.10.6
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Zend-Config*, *Vanilla*

13.2.1003 Check JSON

Check errors whenever JSON is encoded or decoded.

In particular, `NULL` is a valid decoded JSON response. If you want to avoid mistaking NULL for an error, it is recommended to call `json_last_error`.

```
<?php

$encoded = json_encode($incoming);
// Unless JSON must contains some non-null data, this mistakes NULL and error
if(json_last_error() != JSON_ERROR_NONE) {
    die('Error when encoding JSON');
}

$decoded = json_decode($incoming);
// Unless JSON must contains some non-null data, this mistakes NULL and error
if($decoded === null) {
    die('ERROR');
}

?>
```

See also Option to make json_encode and json_decode throw exceptions on errors, json_last_error.

Suggestions

- Always check after JSON operation : encoding or decoding.
- Add a call to json_last_error()
- Configure operations to throw an exception upon error (`JSON_THROW_ON_ERROR`), and catch it.

Specs

Short name	Structures/CheckJson
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.3.0
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Woocommerce*

13.2.1004 Coalesce And Concat

The concatenation operator dot has precedence over the coalesce operator ??.

```
<?php

// Parenthesis are the right solution when in doubt
echo a . ($b ?? 'd') . $e;

// 'a' . $b is evaluated first, leading ot a useless ?? operator
'a' . $b ?? $c;

// 'd' . 'e' is evaluated first, leading to $b OR 'de'.
echo $b ?? 'd' . 'e';

?>
```

Suggestions

- Add parenthesis around ?? operator to avoid misbehavior
- Do not use dot and ?? together in the same expression

Specs

Short name	Structures/CoalesceAndConcat
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.9.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1005 Common Alternatives

In the following conditional structures, expressions were found that are common to both 'then' and 'else'. It may be interesting, though not always possible, to put them both out of the conditional, and reduce line count.

```
<?php
if ($c == 5) {
    $b = strtolower($b[2]);
    $a++;
} else {
    $b = strtolower($b[2]);
    $b++;
}
?>
```

may be rewritten in :

```
<?php

$b = strtolower($b[2]);
if ($c == 5) {
    $a++;
} else {
    $b++;
}

?>
```

Suggestions

- Collect common expressions, and move them before of after the if/then expression.
- Move a prefix and suffixes to a third-party method

Specs

Short name	Structures/CommonAlternatives
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Dolibarr*, *NextCloud*

13.2.1006 Compared But Not Assigned Strings

Those strings are compared to variables in the code, but those values are never assigned.

```
<?php

$a = 'b';

// Depending on the origin of $b, is this possible?
if ($b === 'c') {

}

?>
```

Specs

Short name	Structures/ComparedButNotAssignedStrings
Rulesets	none
Exakt since	1.3.2
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1007 Compared Comparison

Usually, comparison are sufficient, and it is rare to have to compare the result of comparison. Check if this two-stage comparison is really needed.

```
<?php

if ($a === strpos($string, $needle) > 2) {}

// the expression above apply precedence :
// it is equivalent to :
if (($a === strpos($string, $needle)) > 2) {}

?>
```

See also Operators Precedence.

Specs

Short name	Structures/ComparedComparison
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1008 Strict Or Relaxed Comparison

PHP has two comparison styles : strict and relaxed.

The analyzed code has less than 10% of one of them : for consistency reasons, it is recommended to make them all the same.

It is recommended to always use the strict comparison by default, and use the relaxed in case of specific situations.

```
<?php

// This compares $strict both in terms of value and type
if ($strict === 3) {

} elseif ($strict == 3) {
    // This compares $strict after an possible type casting.
    // '3', 3.0 or 3 would all be possible solutions.
}

?>
```

See also Comparison Operators.

Specs

Short name	Structures/ComparisonFavorite
Rulesets	none
Exakt since	1.3.2
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1009 Too Complex Expression

Long expressions should be broken in small chunks, to limit complexity.

Really long expressions tends to be error prone : either by typo, or by missing details. They are even harder to review, once the initially build of the expression is gone.

As a general rule, it is recommended to keep expressions short. The analysis include any expression that is more than 15 tokens large : variable and operators counts as one, properties, arrays count as two. Parenthesis are also counted.

PHP has no specific limit to expression size, so long expression are legal and valid. It is possible that the business logic requires a complex equation.

```
<?php

// Why not calculate wordwrap size separatedly ?
$a = explode("\n", wordwrap($this->message, floor($this->width / imagefontwidth($this-
↪>fontsize)), "\n"));

// Longer but easier to read
$width = floor($this->width / imagefontwidth($this->fontsize)), "\n");
$a = explode("\n", wordwrap($this->message, $width);

// Here, some string building, including error management with @, is making the data␣
↪quite complex.
fwrite($fp, 'HEAD ' . @$url['path'] . @$url['query'] . ' HTTP/1.0' . "\r\n" . 'Host:
↪' . @$url['host'] . "\r\n\r\n")

// Better validation of data.
$http_header = 'HEAD ';
if (isset($url['path'])) {
    $http_header .= $url['path'];
}
if (isset($url['query'])) {
    $http_header .= $url['query'];
}

$http_header .=  "\r\n";
if (isset($url['host'])) {
    $http_header .= 'Host: ' . $url['host'] . "\r\n\r\n";
}

fwrite($fp, $http_header);

?>
```

Suggestions

- Reduce complexity by breaking the expressions into smaller ones

Name	Default	Type	Description
complexExpressionThreshold	30	integer	Minimal number of operators in one expression to report.

Specs

Short name	Structures/ComplexExpression
Rulesets	*CE*
Exakt since	0.12.16
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1010 Concat Empty String

Using a concatenation to make a value a string should be replaced with a type cast.

Type cast to a string is done with `(string)` operator. There is also the function strval(), although it is less recommended.

```
<?php

$a = 3;

// explicite way to cast a value
$b = (string) $a; // $b is a string with the content 3

// Wrong way to cast a value
$c = $a . ''; // $c is a string with the content 3
$c = '' . $a; // $c is a string with the content 3
$a .= '';     // $a is a string with the content 3

// Wrong way to cast a value
$c = $a . '' . $b; // This is not reported. The empty string is useless, but not␣
↪meant to type cast

?>
```

See also Type Casting and PHP Type Casting.

Suggestions

- Avoid concatenating with empty strings
- Use (string) operator to cast to string
- Remove any concatenated empty string

Specs

Short name	Structures/ConcatEmpty
Rulesets	*Analyze*
Exakt since	1.8.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1011 Concatenation Interpolation Consistence

Concatenations are done with the . operator or by interpolation inside a string.

Interpolation is a clean way to write concatenation, though it gets messy with long dereferences or with constants. Concatenations are longer to write.

The analyzed code has less than 10% of one of them : for consistency reasons, it is recommended to make them all the same.

```
<?php

// be consistent
$a = $b $c;
$d = $b $e;
$e = $b $e;
$d = $b $f;
$f = $b $z;
$h = $b $e;
$y = $b $e;
$d = $b $x;
$j = $b $c;
$d = $b $g;
$d = $b $h;

// Be consistent, always use the same.
$z = $w.' '.$e;

?>
```

Specs

Short name	Structures/ConcatenationInterpolationFavorite
Rulesets	none
Exakt since	0.11.6
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1012 Conditional Structures

Structures that are defined, but only executed conditionally.

```
<?php

if (!function_exists('array_column')) {
    function array_column($a) {
        // some PHP
    }
}

if (!class_exists('foo')) {
    class foo {

    }
}

?>
```

Specs

Short name	Structures/ConditionalStructures
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1013 Constant Comparison

Constant to the left or right is a favorite.

Comparisons are commutative : they may be $a == B or B == $a. The analyzed code show less than 10% of one of the two : for consistency reasons, it is recommended to make them all the same.

Putting the constant on the left is also called 'Yoda Comparison', as it mimics the famous characters style of speech. It prevents errors like 'B = $a' where the comparison is turned into an assignation.

The natural way is to put the constant on the right. It is often less surprising.

Every comparison operator is used when finding the favorite.

```
<?php

//
if ($a === B) { doSomething(); }
if ($c > D) { doSomething(); }
if ($e !== G) { doSomething(); }
do { doSomething(); } while ($f === B);
while ($a === B) { doSomething(); }

// be consistent
```

(continues on next page)

(continued from previous page)

```
if (B === $a) {}

// Compari
if (B <= $a) {}

?>
```

Specs

Short name	Structures/ConstantComparisonConsistance
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1014 Constant Conditions

If/then structures have constant condition.

The condition doesn't change during execution, and the following blocks are always executed or not. This may also lead to an infinite or a null loop.

When this is the case, the condition may be removed, and dead code may be removed.

```
<?php

// static if
if (0.8) {
    $a = $x;
} else {
    $a = $y;
}

// static while
while (1) {
    $a = $x;
}

// static do..while
do {
    $a = $x;
} while ('b'. 'c');

// constant for() : No increment
for ($i = 0; $i < 10; ) {
    $a = $x;
}

// constant for() : No final check
for ( $i = 0; ; ++$i) {
    $a = $x;
```

(continues on next page)

(continued from previous page)

```
}


// static ternary
$a = TRUE ? $x : $y;

?>
```

It is advised to remove them, or to make them depend on configuration.

Specs

Short name	Structures/ConstantConditions
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1015 Constant Scalar Expressions

> Define constant with the result of static expressions. This means that constants may be defined with the const keyword, with the help of various operators but without any functioncalls.

This feature was introduced in PHP 5.6. It also supports array(), and expressions in arrays.

Those expressions (using simple operators) may only manipulate other constants, and all values must be known at compile time.

```
<?php

// simple definition
const A = 1;

// constant scalar expression
const B = A * 3;

// constant scalar expression
const C = [A ** 3, '3' => B];

?>
```

See also Constant Scalar Expressions.

Specs

Short name	Structures/ConstantScalarExpression
Rulesets	*CE*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*
Exakt since	0.8.4
Php Version	5.6+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1016 Const Or Define

`const` and `define()` have the same functional use : create constants.

The analyzed code has less than 10% of one of them : for consistency reasons, it is recommended to make them all the same.

They are almost interchangeable, though not totally : `define()` allows the creation of case-insensitive constants, while `Const` won't.

```
<?php

// be consistent
const A1  = 1 ;
const A2  = 2 ;
const A3  = 3 ;
const A4  = 4 ;
const A5  = 5 ;
const A6  = 6 ;
const A7  = 7 ;
const A8  = 8 ;
const A9  = 9 ;
const A10 = 10;
const A11 = 11;

define('A12', 12); // Be consistent, always use the same.

?>
```

See also define and const.

Specs

Short name	Structures/ConstDefineFavorite
Rulesets	*CE*
Exakt since	0.12.1
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1017 Continue Is For Loop

break and continue are very similar in PHP : they both break out of loop or switch. Yet, continue should be reserved for loops.

Since PHP 7.3, the execution will emit a warning when finding a continue inside a switch inside a loop : '"continue" targeting switch is equivalent to "break". Did you mean to use "continue 2"?'

```
<?php

while ($foo) {
    switch ($bar) {
        case 'baz':
            continue; // In PHP: Behaves like 'break;'
                      // In C:   Behaves like 'continue 2;'
    }
}

?>
```

See also Deprecate and remove 'continue targeting switch <https://wiki.php.net/rfc/continue_on_switch_deprecation>'_.

Suggestions

- Replace break by continue

Specs

Short name	Structures/ContinueIsForLoop
Rulesets	*Analyze*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*
Exakt since	1.3.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*XOOPS*

13.2.1018 Could Be Else

Merge opposition conditions into one if/then structure.

When two if/then structures follow each other, using a condition and its opposite, they may be merged into one.

```
<?php

// Short version
if ($a == 1) {
    $b = 2;
} else {
    $b = 1;
}

// Long version
if ($a == 1) {
    $b = 2;
}

if ($a != 1) {
    $b = 3;
}

?>
```

Suggestions

- Merge the two conditions into one structure
- Check if the second condition is still applicable

Specs

Short name	Structures/CouldBeElse
Rulesets	*Analyze*
Exakt since	1.0.1
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*SugarCrm*, *OpenEMR*

13.2.1019 Could Be Static

This global is only used in one function or method. It may be called 'static', instead of global. This allows you to keep the value between call to the function, but will not be accessible outside this function.

```
<?php
function foo( ) {
    static $variableIsReservedForX; // only accessible within foo( ), even between␣
↪calls.
    global $variableIsGlobal;       //      accessible everywhere in the application
}
?>
```

Specs

Short name	Structures/CouldBeStatic
Rulesets	*Analyze*, *ClassReview*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Dolphin*, *Contao*

13.2.1020 Could Use array_fill_keys

> array_fill_keys() is a native PHP function that creates an array from keys. It gets the list of keys, and a constant value to assign to each keys.

This is twice faster than doing the same with a loop.

Note that is possible to use an object as initializing value : every element of the final array will be pointing to the same value. And, also, using an object as initializing value means that the same object will be used for each key : the object will not be cloned for each value.

```
<?php

$array = range('a', 'z');

// Fast way to build the array
$b = array_fill_keys($a, 0);

// Fast way to build the array, but every element will be the same object
$b = array_fill_keys($a, new Stdclass());

// Slow way to build the array
foreach($array as $a) {
    $b[$a] = 0;
}

// Setting everything to null, slowly
$array = array_map(function() {}, $array);

?>
```

See also array_fill_keys.

Suggestions

- Use array_fill_keys()

Specs

Short name	Structures/CouldUseArrayFillKeys
Rulesets	*Suggestions*
Exakt since	1.1.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*ChurchCRM*, *PhpIPAM*

13.2.1021 Could Use array_unique

Use array_unique() to collect unique elements from an array.

Always try to use native PHP functions, instead of rebuilding them with custom PHP code.

```
<?php

    $unique = array();
    foreach ($array as $b) {
        if (!in_array($b, $unique)) {
            /*  May be more code */
            $unique[] = $b;
        }
    }
?>
```

See also array_unique.

Suggestions

- Turn the foreach() and its condition into a call to array_unique()
- Extract the condition from the foreach() and add a separate call to array_unique()

Specs

Short name	Structures/CouldUseArrayUnique
Rulesets	*Suggestions*
Exakt since	1.2.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Dolibarr*, *OpenEMR*

13.2.1022 Could Use Compact

Compact() turns a group of variables into an array. It may be used to simplify expressions.

```php
<?php

$a = 1;
$b = 2;

// Compact call
$array = compact('a', 'b');

$array === [1, 2];

// Detailing all the keys and their value
$array = ['a' => $a, 'b' => $b];

?>
```

Note that compact accepts any string, and any undefined variable is not set, without a warning.

See also compact.

Suggestions

- Replace the array() call with a compact() call.

Specs

Short name	Structures/CouldUseCompact
Rulesets	*Suggestions*
Exakt since	1.1.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*WordPress*

13.2.1023 Could Use __DIR__

Use __DIR__ constant to access the current file's parent directory.

Avoid using dirname() on __FILE__.

```php
<?php

// Better way
$fp = fopen(__DIR__.'/myfile.txt', 'r');

// compatible, but slow way
$fp = fopen(dirname(__FILE__).'/myfile.txt', 'r');

// Since PHP 5.3
assert(dirname(__FILE__) == __DIR__);

?>
```

__DIR__ has been introduced in PHP 5.3.0.

See also Magic Constants.

Suggestions

- Use __DIR__ instead of `dirname(__FILE__);`

Specs

Short name	Structures/CouldUseDir
Rulesets	*Analyze*, *CI-checks*, *Suggestions*, *php-cs-fixable*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Woocommerce*, *Piwigo*

13.2.1024 Could Use Match

The switch() syntax use may be replaced by a match() call.

The simplest case for such refactoring is when each of the switch's case (including default), assign one value to the same variable. See this below :

```
<?php
    switch($a) {
        case 1:
            $b = '1';
            break;
        case 2:
            $b = '3';
            break;
        default:
            $b = '0';
            break;
    }

    /*
    $b = match($a) {
        1 => '1',
        2 => '3',
        default => '0'
    };
    */
?>
```

Match() was introduced in PHP 8. It is not valid with older PHP versions.

See also Match().

Suggestions

- Replace switch() with match()

Specs

Short name	Structures/CouldUseMatch
Rulesets	*Suggestions*
Exakt since	2.2.2
Php Version	8.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Unknown

13.2.1025 Could Use Short Assignation

Use short assignment operator, to speed up code, and keep syntax clear.

Some operators, like * or +, have a compact and fast 'do-and-assign' version. They looks like a compacted version for = and the operator. This syntax is good for readability, and saves some memory in the process.

Depending on the operator, not all permutations of arguments are possible.

Addition and short assignation of addition have a different set of features when applied to arrays. Do not exchange one another in that case.

```
<?php

$a = 10 + $a;
$a += 10;

$b = $b - 1;
$b -= 1;

$c = $c * 2;
$c *= 2;

$d = $d / 3;
$d /= 3;

$e = $e % 4;
$e %= 4;

$f = $f | 5;
$f |= 5;

$g = $g & 6;
$g &= 6;

$h = $h ^ 7;
$h ^= 7;

$i = $i >> 8;
$i >>= 8;
```

(continues on next page)

(continued from previous page)

```
$j = $j << 9;
$j <<= 9;

// PHP 7.4 and more recent
$l = $l ?? 'value';
$l ??= 'value';

?>
```

Short operators are faster than the extended version, though it is a micro-optimization.

See also Assignation Operators.

Suggestions

- Change the expression to use the short assignation

Specs

Short name	Structures/CouldUseShortAssignation
Rulesets	*Analyze*, *CI-checks*, *Performances*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	use-short-assignations
Examples	*ChurchCRM*, *Thelia*

13.2.1026 Could Use str_repeat()

Use str_repeat() or str_pad() instead of making a loop.

Making a loop to repeat the same concatenation is actually much longer than using str_repeat(). As soon as the loop repeats more than twice, str_repeat() is much faster. With arrays of 30, the difference is significant, though the whole operation is short by itself.

```
<?php

// This adds 7 'e' to $x
$x .= str_repeat('e', 7);

// This is the same as above,
for($a = 3; $a < 10; ++$a) {
    $x .= 'e';
}

// here, $default must contains 7 elements to be equivalent to the previous code
foreach($default as $c) {
    $x .= 'e';
```

(continues on next page)

(continued from previous page)

```
}

?>
```

Suggestions

- Use strrepeat() whenever possible

Specs

Short name	Structures/CouldUseStrrepeat
Rulesets	*Analyze*, *CI-checks*, *Top10*
Exakt since	0.11.0
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high
Examples	*Zencart*

13.2.1027 crypt() Without Salt

PHP requires a salt when calling crypt(). 5.5 and previous versions didn't require it. Salt is a simple string, that is usually only known by the application.

According to the manual : The salt parameter is optional. However, crypt() creates a weak hash without the salt. PHP 5.6 or later raise an E_NOTICE error without it. Make sure to specify a strong enough salt for better security.

```
<?php
// Set the password
$password = 'mypassword';

// salted crypt usage (always valid)
$hash = crypt($password, '123salt');

// Get the hash, letting the salt be automatically generated
// This generates a notice after PHP 5.6
$hash = crypt($password);

?>
```

See also crypt.

Suggestions

- Always provide the second argument

Specs

Short name	Structures/CryptWithoutSalt
Rulesets	*CompatibilityPHP54*
Exakt since	0.8.4
Php Version	5.6-
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1028 curl_version() Has No Argument

curl_version() used to accept `CURLVERSION_NOW` as argument. Since PHP 7.4, it is a function without arguments.

```
<?php

// Compatible syntax
$details = curl_version(CURLVERSION_NOW);

// New PHP 7.4 syntax
$details = curl_version();

?>
```

See also curl_version.

Suggestions

- Drop all arguments from curl_version() calls.

Specs

Short name	Structures/CurlVersionNow
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1029 Dangling Array References

Always unset a referenced-variable used in a loop.

It is highly recommended to unset blind variables when they are set up as references after a loop.

```php
<?php

$array = array(1,2,3,4);

foreach($array as &$a) {
    $a += 1;
}
// This only unset the reference, not the value
unset($a);


// Dangling array problem
foreach($array as &$a) {
    $a += 1;
}
//$array === array(3,4,5,6);

// This does nothing (apparently)
// $a is already a reference, even if it doesn't show here.
foreach($array as $a) {}
//$array === array(3,4,5,5);

?>
```

When omitting this step, the next loop that will also require this variable will deal with garbage values, and produce unexpected results.

See also : No Dangling Reference, PHP foreach pass-by-reference: Do it right, or better not at all, How does PHP 'foreach' actually work?, References and foreach.

Suggestions

- Avoid using the reference altogether : sometimes, the reference is not needed.
- Add unset() right after the loop, to avoid reusing the reference.

Specs

Short name	Structures/DanglingArrayReferences
Rulesets	*Analyze*, *CI-checks*, *Top10*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-dangling-reference
Examples	*Typo3*, *SugarCrm*

13.2.1030 Declare Static Once

Global and static variables should be declared only once in a method. It is also recommended to configure it at the beginning of the method. This could be refined by defining the variable at the last common moment, though it lacks readability.

```
<?php

function foo() {
    if (rand(0, 1)) {
        static $x;

        ++$x;
    } else {
        static $x;

        --$x;
    }
}

?>
```

Defining static or global methods late is a micro-optimisation.

Suggestions

- Remove duplicate static and global calls
- Move the static and global calls to the beginning of the method
- Refactor the static and global variable to properties

Specs

Short name	Structures/DeclareStaticOnce
Rulesets	*Suggestions*
Exakt since	2.2.1
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.1031 Dereferencing String And Arrays

PHP allows the direct dereferencing of strings and arrays.

This was added in PHP 5.5. There is no need anymore for an intermediate variable between a string and array (or any expression generating such value) and accessing an index.

```
<?php
$x = array(4,5,6);
$y = $x[2] ; // is 6

May be replaced by
$y = array(4,5,6)[2];
$y = [4,5,6][2];
?>
```

Specs

Short name	Structures/DereferencingAS
Rulesets	*CE*, *CompatibilityPHP53*, *CompatibilityPHP54*
Exakt since	0.8.4
Php Version	5.3-
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1032 Die Exit Consistence

Die and Exit have the same functional use.

The analyzed code has less than 10% of one of them : for consistency reasons, it is recommended to make them all the same.

It happens that die or exit are used depending on coding style and files. One file may be consistently using exit, while the others are all using exit.

```
<?php

// be consistent
switch ($a) {
    case 1 :
        exit;
    case 2 :
        exit;
    case 3 :
        exit;
    case 4 :
        exit;
    case 5 :
        exit;
    case 6 :
        exit;
    case 7 :
        exit;
    case 8 :
        exit;
    case 9 :
        exit;
    case 10 :
        exit;
    default :
        die();    // Be consistent, always use the same.
}

?>
```

Using die or exit is also the target of other analysis.

Specs

Short name	Structures/DieExitConsistance
Rulesets	none
Exakt since	0.8.9
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1033 Difference Consistence

There are two operators to check a difference : <> and !=.

The analyzed code has less than 10% of one of them : for consistency reasons, it is recommended to make them all the same.

It happens that != and <> are used depending on coding style and files. One file may be consistently using <>, while the others are all using !=.

```
<?php

// Both != and <> are used in the code
// When one of them is used less than 10%, it is reported as a consistence issue.
if ($a != $b) {

} elseif ($c <> $d) {

}

?>
```

<> and != are the two only comparison operators that are identical.

See also Comparison Operators.

Specs

Short name	Structures/DifferencePreference
Rulesets	none
Exakt since	0.11.1
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1034 Directly Use File

Some PHP functions have a close cousin that work directly on files : use them. This is faster and less code to write.

- md5‘() <https://www.php.net/>‘_ => md5_file()

- highlight_string() => highlight_file(), show_source()
- parsekit_compile_string() => parsekit_compile_file()
- parse_ini_string() => parse_ini_file()
- sha1‘() <https://www.php.net/>‘_ => sha1_file()
- simplexml_load_string() => simplexml_load_file()
- yaml_parse() => yaml_parse_file()
- hash() => hash_file()
- hash_hmac() => hash_mac_file()
- hash_update() => hash_update_file()
- recode() => recode_file()
- recode_string() => recode_file()

```
<?php

// Good way
$file_hash = hash_file('sha512', 'example.txt');

// Slow way
$file_hash = hash('sha512', file_get_contents('example.txt'));

?>
```

See also hash_file.

Suggestions

- Use the _file() version of those functions

Specs

Short name	Structures/DirectlyUseFile
Rulesets	*Suggestions*
Exakt since	1.5.5
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1035 __DIR__ Then Slash

__DIR__ must be concatenated with a string starting with /.

The magic constant __DIR__ holds the name of the current directory, without final /. When it is used to build path, then the following path fragment must start with /. Otherwise, two directories names will be merged together.

```
<?php

// __DIR__ = /a/b/c
// $filePath = /a/b/c/g.php

// /a/b/c/d/e/f.txt : correct path
echo __DIR__.'/d/e/f.txt';
echo dirname($filePath).'/d/e/f.txt';

// /a/b/cd/e/f.txt : most probably incorrect path
echo __DIR__.'d/e/f.txt';
echo dirname($filePath).'d/e/f.txt';

?>
```

Suggestions

- Add a check on __DIR__, as it may be '/' when run at the root of the server
- Add a '/' at the beginning of the path after __DIR__.
- Add a call to realpath() or file_exists(), before accessing the file.

Specs

Short name	Structures/DirThenSlash
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.10.3
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	Very high
Examples	*Traq*

13.2.1036 Don't Be Too Manual

Adapt the examples from the PHP manual to your code. Don't reuse directly the same names in your code : be more specific about what to expect in those variables.

```
<?php

// Search for phone numbers in a text
preg_match_all('/((\d{3})-(\d{3})-(\d{4}))/', $string, $phoneNumber);

// Search for phone numbers in a text
preg_match_all('/(\d{3})-(\d{3})-(\d{4})/', $string, $matches);

?>
```

Suggestions

- Use precise name with your variables

Specs

Short name	Structures/DontBeTooManual
Rulesets	none
Exakt since	1.6.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1037 Dont Change The Blind Var

> When using a foreach(), the blind variables hold a copy of the original value. It is confusing to modify them, as it seems that the original value may be changed.

When actually changing the original value, use the reference in the foreach definition to make it obvious, and save the final reassignation.

When the value has to be prepared before usage, then save the filtered value in a separate variable. This makes the clean value obvious, and preserve the original value for a future usage.

```
<?php

// $bar is duplicated and kept
$foo = [1, 2, 3];
foreach($foo as $bar) {
    // $bar is updated but its original value is kept
    $nextBar = $bar + 1;
    print $bar . ' => ' . ($nextBar) . PHP_EOL;
    foobar($nextBar);
}

// $bar is updated and lost
$foo = [1, 2, 3];
foreach($foo as $bar) {
    // $bar is updated but its final value is lost
    print $bar . ' => ' . (++$bar) . PHP_EOL;
    // Now that $bar is reused, it is easy to confuse its value
    foobar($bar);
}

// $bar is updated and kept
$foo = [1, 2, 3];
foreach($foo as &$bar) {
    // $bar is updated and keept
    print $bar . ' => ' . (++$bar) . PHP_EOL;
    foobar($bar);
}

?>
```

Specs

Short name	Structures/DontChangeBlindKey
Rulesets	*Analyze*
Exakt since	0.8.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1038 Dont Compare Typed Boolean

There is no need to compare explicitly a function call to a boolean, when the definition has a boolean return typehint.

The analysis checks for equality and identity comparisons. It doesn't check for the not operator usage.

```
<?php

// Sufficient check
if (foo()) {
    doSomething();
}

// Superfluous check
if (foo() === true) {
    doSomething();
}

function foo() : bool {}

?>
```

Suggestions

- Simplify the code and make it short

Specs

Short name	Structures/DontCompareTypedBoolean
Rulesets	*Suggestions*
Exakt since	2.1.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1039 Don't Loop On Yield

Use `yield from`, instead of looping on a generator with `yield`.

`yield from` delegate the yielding to another generator, and keep calling that generator until it is finished. It also works with implicit generator datastructure, like arrays.

```
<?php

function generator() {
    for($i = 0; $i < 10; ++$i) {
        yield $i;
    }
}

function delegatingGenerator() {
    yield from generator();
}

// Too much code here
function generator2() {
    foreach(generator() as $g) {
        yield $g;
    }
}

?>
```

There is a performance gain when delegating, over looping manually on the generator. You may even consider writing the loop to store all values in an array, then `yield from` the array.

See also Generator delegation via yield from.

Suggestions

- Use *yield from* instead of the whole foreach() loop

Specs

Short name	Structures/DontLoopOnYield
Rulesets	*Suggestions*
Exakt since	1.5.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Dolibarr*, *Tikiwiki*

13.2.1040 Dont Mix ++

++ operators, pre and post, have two distinct behaviors, and should be used separately.

When mixed in a larger expression, they are difficult to read, and may lead to unwanted behaviors.

```
<?php

    // Clear and defined behavior
```

(continues on next page)

(continued from previous page)

```
    $i++;
    $a[$i] = $i;

    // The index is also incremented, as it is used AFTP the incrementation
    // With $i = 2; $a is array(3 => 3)
    $a[$i] = ++$i;

    // $i is actually modified twice
    $i = --$i + 1;
?>
```

See also EXP30-C. Do not depend on the order of evaluation for side effects.

Suggestions

- Extract the increment from the expression, and put it on a separate line.

Specs

Short name	Structures/DontMixPlusPlus
Rulesets	*Analyze*
Exakt since	1.3.2
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Contao*, *Typo3*

13.2.1041 Don't Read And Write In One Expression

Avoid giving value and using it at the same time, in one expression. This is an undefined behavior of PHP, and may change without warning.

One of those changes happens between PHP 7.2 and 7.3 :

```
<?php

$arr = [1];
$ref =& $arr[0];
var_dump($arr[0] + ($arr[0] = 2));
// PHP 7.2: int(4)
// PHP 7.3: int(3)

?>
```

See also UPGRADING 7.3.

Suggestions

- Split the expression in two separate expressions

Specs

Short name	Structures/DontReadAndWriteInOneExpression
Rulesets	*Analyze*, *CE*, *CompatibilityPHP73*, *CompatibilityPHP74*
Exakt since	1.4.9
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.1042 Double Assignation

This happens when a container (variable, property, array index) is assigned with values twice in a row. One of them is probably a debug instruction, that was forgotten.

```
<?php

// Normal assignation
$a = 1;

// Double assignation
$b = 2;
$b = 3;

?>
```

Specs

Short name	Structures/DoubleAssignation
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1043 Double Instructions

Twice the same call in a row. This is worth a check.

```
<?php

// repetition of the same command, with the same effect each time.
$a = array_merge($b, $c);
$a = array_merge($b, $c);

// false positive : commands are identical, but the effect is compounded
$a = array_merge($a, $c);
$a = array_merge($a, $c);
```

(continues on next page)

(continued from previous page)

```
?>
```

Suggestions

- Remove double work
- Avoid repetition by using loops, variadic or quantifiers *(dirname($path, 2))*

Specs

Short name	Structures/DoubleInstruction
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1044 Double Object Assignation

Make sure that assigning the same object to two variables is the intended purpose.

```
<?php

// $x and $y are the same object, as they both hold a reference to the same object.
// This means that changing $x, will also change $y.
$x = $y = new Z();

// $a and $b are distinct values, by default
$a = $b = 1;

?>
```

Suggestions

- Split the double assignation to two distinct instantiations
- Split the double assignation to two distinct lines

Specs

Short name	Structures/DoubleObjectAssignation
Rulesets	*Analyze*, *ClassReview*
Exakt since	2.1.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1045 Drop Else After Return

Avoid else clause when the then clause returns, but not the else. And vice-versa.

This way, the else block disappears, and is now the main sequence of the function.

This is also true if else has a return, and then not. When doing so, don't forget to reverse the condition.

```
<?php

// drop the else
if ($a) {
    return $a;
} else {
    doSomething();
}

// drop the then
if ($b) {
    doSomething();
} else {
    return $a;
}

// return in else and then
if ($a3) {
    return $a;
} else {
    $b = doSomething();
    return $b;
}

?>
```

Suggestions

- Remove the else clause and move its code to the main part of the method

Specs

Short name	Structures/DropElseAfterReturn
Rulesets	*Analyze*, *CI-checks*, *Suggestions*
Exakt since	0.8.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1046 Duplicate Calls

Duplicate calls within the same context. They should be called once, and then stashed in a variable for reuse. This saves a lot of time.

```php
<?php

function foo($name) {
    // The name decoration on the string is done twice. Once should be cached in a␣
↪variable.
    echo Hello,.ucfirst(strtolower($name)).<br />;

    $query = 'Insert into visitors values (.ucfirst(strtolower($name)).)';
    $res = $db->query($query);
}

?>
```

See also Constants and Userland naming Guide.

Specs

Short name	Structures/DuplicateCalls
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
ClearPHP	no-duplicated-code

13.2.1047 Dynamic Calls

List of dynamic calls. They will probably need to be review manually.

```php
<?php

$a = 'b';

// Dynamic call of a constant
echo constant($a);

// Dynamic variables
$$a = 2;
echo $b;

// Dynamic call of a function
$a('b');

// Dynamic call of a method
$object->$a('b');

// Dynamic call of a static method
A::$a('b');

?>
```

See also Variable functions.

Specs

Short name	Structures/DynamicCalls
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1048 Dynamic Code

List of instructions that were left during analysis, as they rely on dynamic data.

```
<?php

// Dynamic call to 'method';
$name = 'method';
$object->$name();

// Hard coded call to 'method';
$object->method();

?>
```

Any further analysis will need to start from here.

See also Variable functions.

Specs

Short name	Structures/DynamicCode
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1049 Echo Or Print

Echo and print have the same functional use. <?= and printf() are also considered in this analysis.

There seems to be a choice that is not enforced : one form is dominant, (> 90%) while the others are rare.

The analyzed code has less than 10% of one of the three : for consistency reasons, it is recommended to make them all the same.

It happens that print, echo or <?= are used depending on coding style and files. One file may be consistently using print, while the others are all using echo.

```
<?php

echo 'a';
echo 'b';
echo 'c';
echo 'd';
echo 'e';
echo 'f';
echo 'g';
echo 'h';
echo 'i';
echo 'j';
echo 'k';

// This should probably be written 'echo';
print 'l';

?>
```

Specs

Short name	Structures/EchoPrintConsistance
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1050 Echo With Concat

> Optimize your `echo`'s by not concatenating at `echo` time, but serving all argument separated. This will save PHP a memory copy.

If values, literals and variables, are small enough, this won't have visible impact. Otherwise, this is less work and less memory waste.

```
<?php
  echo $a, ' b ', $c;
?>
```

instead of

```
<?php
  echo  $a . ' b ' . $c;
  echo $a b $c;
?>
```

Suggestions

- Turn the concatenation into a list of argument, by replacing the dots by commas.

Specs

Short name	Structures/EchoWithConcat
Rulesets	*Analyze*, *Performances*, *Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-unnecessary-string-concatenation
Examples	*Phpdocumentor*, *TeamPass*

13.2.1051 Else If Versus Elseif

Always use elseif instead of else and if.

"The keyword elseif SHOULD be used instead of else if so that all control keywords look like single words". Quoted from the PHP-FIG documentation

```
<?php

// Using elseif
if ($a == 1) { doSomething(); }
elseif ($a == 2) { doSomethingElseIf(); }
else { doSomethingElse(); }

// Using else if
if ($a == 1) { doSomething(); }
else if ($a == 2) { doSomethingElseIf(); }
else { doSomethingElse(); }

// Using else if, no {}
if ($a == 1)  doSomething();
else if ($a == 2) doSomethingElseIf();
else  doSomethingElse();

?>
```

See also elseif/else if.

Suggestions

- Merge else and if into elseif
- Turn the else expression into a block, and have more than the second if in this block
- Turn the if / else if / else into a switch structure

Specs

Short name	Structures/ElseIfElseif
Rulesets	*Analyze*, *CI-checks*, *Rector*, *php-cs-fixable*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*TeamPass*, *Phpdocumentor*

13.2.1052 Else Usage

Else should be avoided by various means. For example, defaulting values before, or short-circuiting the method as soon as the condition is not met.

```
<?php

// $a is always set
$a = 'default';
if ($condition) {
    $a = foo($condition);
}

// Don't use else for default : set default before
if ($condition) {
    $a = foo($condition);
} else {
    $a = 'default';
}

// Use then to exit
if ( ! $condition) {
    return;
}
$a = foo($condition);

// don't use else to return
if ($condition) {
    $a = foo($condition);
} else {
    return;
}

?>
```

See also Avoid Else, Return Early and Why does clean code forbid else expression.

Specs

Short name	Structures/ElseUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1053 Empty Blocks

Full empty block, part of a control structures.

It is recommended to remove those blocks, so as to reduce confusion in the code.

```
<?php

foreach($foo as $bar) ; // This block seems erroneous
    $foobar++;

if ($a === $b) {
    doSomething();
} else {
    // Empty block. Remove this
}

// Blocks containing only empty expressions are also detected
for($i = 0; $i < 10; $i++) {
    ;
}

// Although namespaces are not control structures, they are reported here
namespace A;
namespace B;

?>
```

Suggestions

- Fill the block with a command
- Fill the block with a comment that explain the situation
- Remove the block and its commanding operator

Specs

Short name	Structures/EmptyBlocks
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Cleverstyle*, *PhpIPAM*

13.2.1054 Empty Instructions

Empty instructions are part of the code that have no instructions.

This may be trailing semi-colon or empty blocks for if-then structures.

Comments that explains the reason of the situation are not taken into account.

```
<?php
    $condition = 3;;;;
    if ($condition) { }
?>
```

Suggestions

- Remove the empty lines
- Fill the empty lines

Specs

Short name	Structures/EmptyLines
Rulesets	*Analyze*, *Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Zurmo*, *ThinkPHP*

13.2.1055 Empty Try Catch

The code does try, then catch errors but do no act upon the error.

```
<?php

try {
    doSomething();
} catch (Throwable $e) {
```

(continues on next page)

(continued from previous page)

```
    // ignore this
}

?>
```

At worst, the error should be logged, so as to measure the actual usage of the catch expression.

`catch( Exception $e)` (PHP 5) or `catch(`Throwable <https://www.php.net/manual/en/class.throwable.php>`_ $e)` with empty catch block should be banned. They ignore any error and proceed as if nothing happened. At worst, the event should be logged for future analysis.

See also Empty Catch Clause.

Suggestions

- Add some logging in the catch
- Add a comment to mention why the catch is empty
- Change the exception, chain it and throw again

Specs

Short name	Structures/EmptyTryCatch
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*LiveZilla*, *Mautic*

13.2.1056 Empty With Expression

empty() doesn't accept expressions until PHP 5.5. Until then, it is necessary to store the result of the expression in a variable and then, test it with empty().

```
<?php

// PHP 5.5+ empty() usage
if (empty(strtolower($b . $c))) {
    doSomethingWithoutA();
}

// Compatible empty() usage
$a = strtolower($b . $c);
if (empty($a)) {
    doSomethingWithoutA();
}

?>
```

See also empty.

Suggestions

- Use the compatible syntax, and store the result in a local variable before testing it with empty

Specs

Short name	Structures/EmptyWithExpression
Rulesets	*Suggestions*
Exakt since	0.8.4
Php Version	5.5+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*HuMo-Gen*

13.2.1057 Error Messages

Error message when an error is reported in the code. Those messages will be read by whoever is triggering the error, and it has to be helpful.

It is a good exercise to read the messages out of context, and try to understand what is about.

```
<?php

// Not so helpful messages
die('Here be monsters');
exit('An error happened');
throw new Exception('Exception thrown at runtime');

?>
```

Error messages are spotted via die, exit or throw.

Specs

Short name	Structures/ErrorMessages
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1058 error_reporting() With Integers

Using named constants with error_reporting is strongly encouraged to ensure compatibility for future versions. As error levels are added, the range of integers increases, so older integer-based error levels will not always behave as expected. (Adapted from the documentation).

```
<?php

// This is ready for PHP next version
error_reporting(E_ALL & ~E_DEPRECATED & ~E_STRICT & ~E_NOTICE & ~E_WARNING);

// This is not ready for PHP next version
error_reporting(2047);

// -1 and 0 are omitted, as they will be valid even is constants changes.
error_reporting(-1);
error_reporting(0);

?>
```

See also directive error_reporting and error_reporting.

Suggestions

- Always use the constant combination when configuring error_reporting or any PHP native function

Specs

Short name	Structures/ErrorReportingWithInteger
Rulesets	*Analyze*, *CE*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*SugarCrm*

13.2.1059 Eval() Usage

Using eval() is evil.

Using eval() is bad for performances (compilation time), for caches (it won't be compiled), and for security (if it includes external data).

```
<?php
    // Avoid using incoming data to build the eval() expression : any filtering error␣
↪leads to PHP injection
    $mathExpression = $_GET['mathExpression'];
    $mathExpression = preg_replace('#[^0-9+\-*/\(/)]#is', '', $mathExpression); //␣
↪expecting 1+2
    $literalCode = '$a = '.$mathExpression.';';
    eval($literalCode);
    echo $a;

    // If the code code given to eval() is known at compile time, it is best to put␣
↪it inline
    $literalCode = 'phpinfo();';
    eval($literalCode);
```

(continues on next page)

(continued from previous page)

```
?>
```

Most of the time, it is possible to replace the code by some standard PHP, like variable variable for accessing a variable for which you have the name. At worse, including a pregenerated file is faster and cacheable.

There are several situations where eval() is actually the only solution :

For PHP 7.0 and later, it is important to put eval() in a try..catch expression.

See also eval and The Land Where PHP Uses 'eval() <https://www.exakat.io/land-where-php-uses-eval/>'_.

Suggestions

- Use a dynamic feature of PHP to replace the dynamic code
- Store the code on the disk, and use include
- Replace create_function() with a closure!

Specs

Short name	Structures/EvalUsage
Rulesets	*Analyze*, *CE*, *Performances*, *Security*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-eval
Examples	*XOOPS*, *Mautic*

13.2.1060 eval() Without Try

`eval()` emits a `ParseError` exception with PHP 7 and later. Catching this exception is the recommended way to handle errors when using the `eval()` function.

```
<?php

$code = 'This is no PHP code.';

//PHP 5 style
eval($code);
// Ends up with a Fatal error, at execution time

//PHP 7 style
try {
    eval($code);
} catch (ParseError $e) {
    cleanUpAfterEval();
}

?>
```

Note that it will catch situations where `eval()` is provided with code that can't be used, but it will not catch security problems. Avoid using `eval()` with incoming data.

Suggestions

- Always add a try/catch block around eval() call

Specs

Short name	Structures/EvalWithoutTry
Rulesets	*Analyze*, *CI-checks*, *Security*
Exakt since	0.8.4
Php Version	7.0+
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High
Examples	*FuelCMS*, *ExpressionEngine*

13.2.1061 Exit() Usage

Using exit or die() <https://www.php.net/'die>'_ in the code makes the code untestable (it will break unit tests). Moreover, if there is no reason or string to display, it may take a long time to spot where the application is stuck.

```
<?php

// Throw an exception, that may be caught somewhere
throw new \Exception('error');

// Dying with error message.
die('error');

function foo() {
    //exiting the function but not dying
    if (somethingWrong()) {
        return true;
    }
}
?>
```

Try exiting the function/class with return, or throw exception that may be caught later in the code.

Suggestions

- Avoid exit and die. Let the script finish.
- Throw an exception and let it be handled before finishing

Specs

Short name	Structures/ExitUsage
Rulesets	*Analyze*, *CE*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-exit
Examples	*Traq*, *ThinkPHP*

13.2.1062 Failed Substr Comparison

The extracted string must be of the size of the compared string.

This is also true for negative lengths.

```
<?php

// Possible comparison
if (substr($a, 0, 3) === 'abc') { }
if (substr($b, 4, 3) === 'abc') { }

// Always failing
if (substr($a, 0, 3) === 'ab') { }
if (substr($a, 3, -3) === 'ab') { }

// Omitted in this analysis
if (substr($a, 0, 3) !== 'ab') { }

?>
```

Suggestions

- Fix the string
- Fix the length of the string
- Put the string in a constant, and use strlen() or mb_strlen()

Specs

Short name	Structures/FailingSubstrComparison
Rulesets	*Analyze*, *CI-checks*, *Top10*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Zurmo*, *MediaWiki*

13.2.1063 Switch Fallthrough

A switch with fallthrough is prone to errors.

A fallthrough happens when a case or default clause in a switch statement is not finished by a break (or equivalent); CWE report this as a security concern, unless well documented.

A fallthrough may be used as a feature. Then, it is indistinguishable from an error.

When the case block is empty, this analysis doesn't report it : the case is then used as an alias.

```
<?php
switch($variable) {
    case 1 :   // case 1 is not reported, as it actually shares the same body as case␣
↪33
    case 33 :
        break ;
    case 2 :
        break ;
    default:
        ++$a;
    case 4 :
        break ;
}
?>
```

This analysis doesn't take into account comments about the fallthrough.

See also CWE-484: Omitted 'Break Statement in Switch <https://cwe.mitre.org/data/definitions/484.html>'_ and Rule: no-switch-case-fall-through.

Suggestions

- Make separate code for each case. Always use break at the end of a case or default.

Specs

Short name	Structures/Fallthrough
Rulesets	*Security*
Exakt since	0.12.14
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1064 File Uploads

This code makes usage of file upload features of PHP.

Upload file feature is detected through the usage of specific functions :

```
<?php
$uploaddir = '/var/www/uploads/';
$uploadfile = $uploaddir . basename($_FILES['userfile']['name']);
```

(continues on next page)

(continued from previous page)

```
echo '<pre>';
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
    echo 'File is valid, and was successfully uploaded.'.PHP_EOL;
} else {
    echo 'Possible file upload attack!'.PHP_EOL;
}

echo 'Here is some more debugging info:';
print_r($_FILES);

print '</pre>';

?>
```

See also Handling file uploads.

Specs

Short name	Structures/FileUploadUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1065 File Usage

The application makes usage of files on the system (read, write, delete, etc.).

Files usage is based on the usage of file functions.

```
<?php
    $fp = fopen('/tmp/file.txt', 'w+');
    // ....
?>
```

See also filesystem.

Specs

Short name	Structures/FileUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1066 Foreach Needs Reference Array

When using foreach with a reference as value, the source must be a referenced array, which is a variable (or array or property or static property).

When the array is the result of an expression, the array is not kept in memory after the foreach loop, and any change made with & are lost.

This will do nothing

```
<?php
    foreach(array(1,2,3) as &$value) {
        $value *= 2;
    }
?>
```

This will have an actual effect

```
<?php
    $array = array(1,2,3);
    foreach($array as &$value) {
        $value *= 2;
    }
?>
```

Specs

Short name	Structures/ForeachNeedReferencedSource
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1067 Foreach Reference Is Not Modified

Foreach statement may loop using a reference, especially when the loop has to change values of the array it is looping on.

In the spotted loop, reference are used but never modified. They may be removed.

```
<?php

$letters = range('a', 'z');

// $letter is not used here
foreach($letters as &$letter) {
    $alphabet .= $letter;
}

// $letter is actually used here
foreach($letters as &$letter) {
    $letter = strtoupper($letter);
```

(continues on next page)

(continued from previous page)

```
}

?>
```

Suggestions

- Remove the reference from the foreach
- Actually modify the content of the reference

Specs

Short name	Structures/ForeachReferenceIsNotModified
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Dolibarr*, *Vanilla*

13.2.1068 Overwritten Source And Value

In a foreach(), it is best to keep source and values distinct. Otherwise, they overwrite each other.

Since PHP 7.0, PHP makes a copy of the original source, then works on it. This makes possible to use the same name for the source and the values.

```
<?php

// displays 0-1-2-3-3
$array = range(0, 3);
foreach($array as $array) {
    print $array . '-';
}
print_r($array);


/* displays 0-1-2-3-Array
(
    [0] => 0
    [1] => 1
    [2] => 2
    [3] => 3
)
*/
$array = range(0, 3);
foreach($array as $v) {
    print $v . '-';
}
print_r($array);
```

(continues on next page)

(continued from previous page)

```
?>
```

When the source is used as the value, the elements in the array are successively assigned to itself. After the loop, the original array has been replaced by its last element.

The same applies to the index, or to any variable in a list() structure, used in a foreach().

Suggestions

- Keep the source, the index and the values distinct

Specs

Short name	Structures/ForeachSourceValue
Rulesets	*Analyze*
Exakt since	1.8.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*ChurchCRM*, *ExpressionEngine*

13.2.1069 Foreach With list()

Foreach loops have the ability to use list() (or []) as blind variables. This syntax assign directly array elements to various variables.

PHP 5.5 introduced the usage of list in foreach() loops. Until PHP 7.1, it was not possible to use non-numerical arrays as list() wouldn't support string-indexed arrays.

```
<?php
    // PHP 5.5 and later, with numerically-indexed arrays
    foreach($array as list($a, $b)) {
        // do something
    }


    // PHP 7.1 and later, with arrays
    foreach($array as list('col1' => $a, 'col3' => $b)) { // 'col2 is ignored'
        // do something
    }
?>
```

Previously, it was compulsory to extract() the data from the blind array :

```
<?php
    foreach($array as $c) {
        list($a, $b) = $c;
        // do something
    }
?>
```

See also The list function & practical uses of array destructuring in PHP and Array destructuring in PHP.

Specs

Short name	Structures/ForeachWithList
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*
Exakt since	0.8.4
Php Version	5.5+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.1070 Forgotten Whitespace

Forgotten whitespaces only bring misery to the code.

White spaces have been left at either end of a file : before the PHP opening tag, or after the closing tag.

Usually, such whitespaces are forgotten, and may end up summoning the infamous 'headers already sent' error. It is better to remove them.

```
<?php
    // This script has no forgotten whitespace, not at the beginning
    function foo() {}

    // This script has no forgotten whitespace, not at the end
?>
```

See also How to fix Headers already sent error in PHP.

Suggestions

- Remove all whitespaces before and after a script. This doesn't apply to template, which may need to use those spaces.
- Remove the final tag, to prevent any whitespace to be forgotten at the end of the file. This doesn't apply to the opening PHP tag, which is always necessary.

Specs

Short name	Structures/ForgottenWhiteSpace
Rulesets	*Analyze*, *CE*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1071 For Using Functioncall

It is recommended to avoid functioncall in the for() statement.

```
<?php

// Fastest way
$nb = count($array);
for($i = 0; $i < $nb; ++$i) {
    doSomething($i);
}

// Same as above, but slow
for($i = 0; $i < count($array); ++$i) {
    doSomething($i);
}

// Same as above, but slow
foreach($portions as &$portion) {
    // here, array_sum() doesn't depends on the $grade. It should be out of the loop
    $portion = $portion / array_sum($portions);
}

$total = array_sum($portion);
foreach($portion as &$portion) {
    $portion = $portion / $total;
}

?>
```

This is true with any kind of functioncall that returns the same value throughout the loop.

Suggestions

- Call the function once, before the loop

Specs

Short name	Structures/ForWithFunctioncall
Rulesets	*Performances*, *Top10*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
ClearPHP	no-functioncall-in-loop

13.2.1072 Function Subscripting, Old Style

Since PHP 5.4, it is now possible use function results as an array, and access directly its element :

```
<?php

function foo() {
    return array(1 => 'a', 'b', 'c');
}

echo foo()[1]; // displays 'a';

// Function subscripting, the old way
function foo() {
    return array(1 => 'a', 'b', 'c');
}

$x = foo();
echo $x[1]; // displays 'a';

?>
```

Suggestions

- Skip the local variable and directly use the return value from the function

Specs

Short name	Structures/FunctionPreSubscripting
Rulesets	*Suggestions*
Exakt since	0.8.4
Php Version	5.4+
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*OpenConf*

13.2.1073 Function Subscripting

It is possible to use the result of a methodcall directly as an array, without storing the result in a temporary variable.

This works, given that the method actually returns an array.

This syntax was not possible until PHP 5.4. Until then, it was compulsory to store the result in a variable first. Although this is now superfluous, it has been a standard syntax in PHP, and is still being used.

```
<?php

function foo() {
    return array(1 => 'a', 'b', 'c');
}

echo foo()[1]; // displays 'a';

// Function subscripting, the old way
```

(continues on next page)

(continued from previous page)

```
function foo() {
    return array(1 => 'a', 'b', 'c');
}

$x = foo();
echo $x[1]; // displays 'a';

?>
```

Storing the result in a variable is still useful if the result is actually used more than once.

Specs

Short name	Structures/FunctionSubscripting
Rulesets	*CE*, *CompatibilityPHP53*
Exakt since	0.8.4
Php Version	5.4+
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1074 Global In Global

List of global variables. There are the global variables, defined with the global keyword, and the implicit global variables, defined in the global scope.

```
<?php
    global $explicitGlobal; // in global namespace

    $implicitGlobal = 1; // in global namespace, variables are automatically global

    function foo() {
        global $explicitGlobalInFoo; // in functions, globals must be declared with␣
↪global
    }
?>
```

See also Variable Scope.

Specs

Short name	Structures/GlobalInGlobal
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1075 Global Inside Loop

The global keyword must be used out of loops. Otherwise, it is evaluated each loop, slowing the whole process.

```
<?php

// Here, global is used once
global $total;
foreach($a as $b) {
    $total += $b;
}

// Global is called each time : this is slow.
foreach($a as $b) {
    global $total;
    $total += $b;
}
?>
```

Suggestions

- Move the global keyword outside the loop

Specs

Short name	Structures/GlobalOutsideLoop
Rulesets	*Performances*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1076 Global Usage

List usage of globals variables, with global keywords or direct access to $GLOBALS.

```
<?php
$a = 1; /* global scope */

function test()
{
    echo $a; /* reference to local scope variable */
}

test();

?>
```

It is recommended to avoid using global variables, at it makes it very difficult to track changes in values across the whole application.

See also Variable scope.

Specs

Short name	Structures/GlobalUsage
Rulesets	*Analyze*, *CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
ClearPHP	no-global

13.2.1077 Find Key Directly

No need for a foreach() to search for a key.

PHP offers two solutions : array_search() and array_keys(). Array_search() finds the first key that fits a value, and array_keys returns all the keys.

```
<?php

$array = ['a', 'b', 'c', 'd', 'e'];

print array_search($array, 'c');
// print 2 => 'c';

print_r(array_keys($array, 'c'));
// print 2 => 'c';

?>
```

See also array_search and array_keys.

Specs

Short name	Structures/GoToKeyDirectly
Rulesets	none
Exakt since	1.1.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.1078 Comparisons Orientation

Maths has two comparisons styles : > or <. Both may include equality : <= and >=.

The analyzed code has less than 10% of one of them : for consistency reasons, it is recommended to make them all the same.

It is recommended to always use the same comparison style.

```
<?php

// Always compare in the same direction
if ($a > $c) {

} elseif ($c > $b) {

} else {
    // equality case
}

// Alterning comparison style lead to harder to read code
if ($b > 3) {

} elseif ($b < 3) {

}

?>
```

See also Comparison Operators.

Specs

Short name	Structures/GtOrLtFavorite
Rulesets	none
Exakt since	1.3.2
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1079 Heredoc Delimiter

Heredoc and Nowdoc expressions may use a variety of delimiters.

There seems to be a standard delimiter in the code, and some exceptions : one or several forms are dominant (> 90%), while the others are rare.

The analyzed code has less than 10% of the rare delimiters. For consistency reasons, it is recommended to make them all the same.

Generally, one or two delimiters are used, with generic value. It is recommended to use a humanly readable delimiter : SQL, HTML, XML, GREMLIN, etc. This helps readability in the code.

```
<?php

echo <<<SQL
SELECT * FROM table1;
SQL;

echo <<<SQL
```

(continues on next page)

(continued from previous page)

```
SELECT * FROM table2;
SQL;

echo <<<SQL
SELECT * FROM table3;
SQL;

echo <<<SQL
SELECT * FROM table4;
SQL;

echo <<<SQL
SELECT * FROM table5;
SQL;

echo <<<SQL
SELECT * FROM table11;
SQL;

echo <<<SQL
SELECT * FROM table12;
SQL;

echo <<<SQL
SELECT * FROM table13;
SQL;

// Nowdoc
echo <<<'SQL'
SELECT * FROM table14;
SQL;

echo <<<SQL
SELECT * FROM table15;
SQL;


echo <<<HEREDOC
SELECT * FROM table215;
HEREDOC;

?>
```

Specs

Short name	Structures/HeredocDelimiterFavorite
Rulesets	none
Exakt since	0.12.0
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1080 Htmlentities Calls

htmlentities() and htmlspecialchars() are used to prevent injecting special characters in HTML code. As a bare minimum, they take a string and encode it for HTML.

The second argument of the functions is the type of protection. The protection may apply to quotes or not, to HTML 4 or 5, etc. It is highly recommended to set it explicitly.

The third argument of the functions is the encoding of the string. In PHP 5.3, it is `ISO-8859-1`, in 5.4, was `UTF-8`, and in 5.6, it is now default_charset, a `php.ini` configuration that has the default value of `UTF-8`. It is highly recommended to set this argument too, to avoid distortions from the configuration.

```
<?php
$str = 'A quote is <b>bold</b>';

// Outputs, without depending on the php.ini: A &#039;quote&#039; is &lt;b&gt;bold&lt;
↪/b&gt;
echo htmlentities($str, ENT_QUOTES, 'UTF-8');

// Outputs, while depending on the php.ini: A quote is &lt;b&gt;bold&lt;/b&gt;
echo htmlentities($str);

?>
```

Also, note that arguments 2 and 3 are constants and string, respectively, and should be issued from the list of values available in the manual. Other values than those will make PHP use the default values.

See also htmlentities and htmlspecialchars.

Suggestions

- Always use the third argument with htmlentities()

Specs

Short name	Structures/Htmlentitiescall
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.1081 Identical Conditions

These logical expressions contain members that are identical.

This means those expressions may be simplified.

```
<?php

// twice $a
if ($a || $b || $c || $a) {  }
```

(continues on next page)

(continued from previous page)

```
// Hiding in parenthesis is bad
if ((($a) ^ ($a))) {}

// expressions may be large
if ($a === 1 && 1 === $a) {}

?>
```

Suggestions

- Merge the two structures into one unique test
- Add extra expressions between the two structures
- Nest the structures, to show that different attempts are made

Specs

Short name	Structures/IdenticalConditions
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High
Examples	*WordPress*, *Dolibarr*

13.2.1082 Identical Consecutive Expression

Identical consecutive expressions are worth being checked.

They may be a copy/paste with unmodified content. When the content has to be duplicated, it is recommended to avoid executing the expression again, and just access the cached result.

```
<?php

$current  = $array[$i];
$next     = $array[$i + 1];
$nextnext = $array[$i + 1]; // OOps, nextnext is wrong.

// Initialization
$previous = foo($array[1]); // previous is initialized with the first value on purpose
$next     = foo($array[1]); // the second call to foo() with the same arguments␣
↪should be avoided
// the above can be rewritten as :
$next     = $previous; // save the processing.

for($i = 1; $i < 200; ++$i) {
    $next = doSomething();
}
?>
```

Specs

Short name	Structures/IdenticalConsecutive
Rulesets	*Analyze*
Exakt since	1.0.8
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1083 Identical On Both Sides

Operands should be different when comparing or making a logical combination. Of course, the value each operand holds may be identical. When the same operand appears on both sides of the expression, the result is know before execution.

```
<?php

// Trying to confirm consistency
if ($login == $login) {
    doSomething();
}

// Works with every operators
if ($object->login( ) !== $object->login()) {
    doSomething();
}

if ($sum >= $sum) {
    doSomething();
}

//
if ($mask && $mask) {
    doSomething();
}

if ($mask || $mask) {
    doSomething();
}

?>
```

Suggestions

- Remove one of the alternative, and remove the logical link
- Modify one of the alternative, and make it different from the other

Specs

Short name	Structures/IdenticalOnBothSides
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.0.8
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*phpMyAdmin*, *HuMo-Gen*

13.2.1084 Iffectations

Affectations that appears in a condition.

Iffectations are a way to do both a test and an affectations. They may also be typos, such as if ($x = 3) { ... }, leading to a constant condition.

```
<?php

// an iffectation : assignation in a If condition
if($connexion = mysql_connect($host, $user, $pass)) {
    $res = mysql_query($connexion, $query);
}

// Iffectation may happen in while too.
while($row = mysql_fetch($res)) {
    $store[] = $row;
}

?>
```

Suggestions

- Move the assignation inside the loop, and make an existence test in the condition.
- Move the assignation before the if/then, make an existence test in the condition.

Specs

Short name	Structures/Iffectation
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.1085 If With Same Conditions

Successive If / then structures that have the same condition may be either merged or have one of the condition changed.

```
<?php

if ($a == 1) {
    doSomething();
}

if ($a == 1) {
    doSomethingElse();
}

// May be replaced by
if ($a == 1) {
    doSomething();
    doSomethingElse();
}

?>
```

Note that if the values used in the condition have been modified in the first if/then structure, the two distinct conditions may be needed.

```
<?php

// May not be merged
if ($a == 1) {
    // Check that this is really the situation
    $a = checkSomething();
}

if ($a == 1) {
    doSomethingElse();
}

?>
```

Suggestions

- Merge the two conditions so the condition is used once.
- Change one of the condition, so they are different
- Make it obvious that the first condition is a try, preparing the normal conditions.

Specs

Short name	Structures/IfWithSameConditions
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*phpMyAdmin*, *Phpdocumentor*

13.2.1086 Implicit Global

Global variables, that are used in local scope with global keyword, but are not declared as global in the global scope. They may be mistaken with distinct values, while, in PHP, variables in the global scope are truly global.

```
<?php

// This is implicitely global
$implicitGlobal = 1;

global $explicitGlobal;
$explicitGlobal = 2;

foo();
echo $explicitFunctionGlobal;

function foo() {
    // This global is needed, but not the one in the global space
    global $implicitGlobal, $explicitGlobal, $explicitFunctionGlobal;

    // This won't be a global, as it must be 'global' in a function scope
    $notImplicitGlobal = 3;
    $explicitFunctionGlobal = 3;
}

?>
```

Specs

Short name	Structures/ImplicitGlobal
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1087 Implied If

It is confusing to emulate if/then with boolean operators.

It is possible to emulate a if/then structure by using the operators 'and' and 'or'. Since optimizations will be applied to them : when the left operand of 'and' is false, the right one is not executed, as its result is useless; when the left operand of 'or' is true, the right one is not executed, as its result is useless;

However, such structures are confusing. It is easy to misread them as conditions, and ignore an important logic step.

```
<?php

// Either connect, or die
mysql_connect('localhost', $user, $pass) or die();

// Defines a constant if not found.
defined('SOME_CONSTANT') and define('SOME_CONSTANT', 1);

// Defines a default value if provided is empty-ish
// Warning : this is
$user = $_GET['user'] || 'anonymous';

?>
```

It is recommended to use a real 'if then' structures, to make the condition readable.

Specs

Short name	Structures/ImpliedIf
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-implied-if

13.2.1088 Implode() Arguments Order

implode() accepted two signatures, but is only recommending one. Both types orders of string then array, and array then string have been possible until PHP 7.4.

In PHP 7.4, the order array then string is deprecated, and emits a warning. It will be removed in PHP 8.0.

```
<?php

$glue = ',';
$pieces = range(0, 4);

// documented argument order
$s = implode($glue, $pieces);

// Pre 7.4 argument order
$s = implode($pieces, $glue);
```

(continues on next page)

(continued from previous page)

```
// both produces 0,1,2,3,4

?>
```

See also implode().

Suggestions

- Always use the array as the second argument

Specs

Short name	Structures/ImplodeArgsOrder
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.9.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1089 Inclusions

List of all inclusions. Inclusions are made with include(), include_once(), require() and require_once().

```
<?php

include 'library.php';

// display is a function defined in 'library.php';
display('Message');

?>
```

Specs

Short name	Structures/IncludeUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1090 Inconsistent Concatenation

Concatenations happens within a string or using the dot operator. Using both is an inconsistent way of writing concatenations.

Switching methods of concatenation, sometimes in the same expression, is error prone. The reader gets confused, and may miss important information.

```
<?php

    //Concatenation
  $consistent = $a . 'b'. $c;

    //Interpolation
  $consistentToo = "{$a}b$c";

    // Concatenation and interpolation
  $inconsistent = $a . "b$c";

    // Concatenation and interpolation too
  $consistentThree = <<<CONSISTENT
{$a}b$c
CONSISTENT;

    // Concatenation and interpolation collisions
  $collision = theClass::CONSTANTE . "b{$c}".number_format($t, 2).' $CAD'.\n;

?>
```

There are some situations where using concatenation are compulsory : when calling a constant, or a function, or make use of the escape sequence. Those are ignored in this analysis.

Specs

Short name	Structures/InconsistentConcatenation
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*FuelCMS*

13.2.1091 Inconsistent Elseif

Chaining if/elseif requires a consistent string of conditions. The conditions are executed one after the other, and the conditions shouldn't overlap.

This analysis reports chains of elseif that don't share a common variable (or array, or property, etc..). As such, testing different conditions are consistent.

```
<?php

// $a is always common, so situations are mutually exclusive
```

(continues on next page)

(continued from previous page)

```
if ($a === 1) {
    doSomething();
} else if ($a > 1) {
    doSomethingElse();
} else {
    doSomethingDefault();
}

// $a is always common, so situations are mutually exclusive
// although, it may be worth checking the consistency here
if ($a->b === 1) {
    doSomething();
} else if ($a->c > 1) {
    doSomethingElse();
} else {
    doSomethingDefault();
}

// if $a === 1, then $c doesn't matter?
// This happens, but then logic doesn't appear in the code.
if ($a === 1) {
    doSomething();
} else if ($c > 1) {
    doSomethingElse();
} else {
    doSomethingDefault();
}

?>
```

Specs

Short name	Structures/InconsistentElseif
Rulesets	*Analyze*
Exakt since	1.4.3
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.1092 Indices Are Int Or String

Indices in an array notation such as `$array['indice']` may only be integers or string.

Boolean, Null or float will be converted to their integer or string equivalent.

```
<?php
    $a = [true => 1,
          1.0  => 2,
          1.2  => 3,
          1    => 4,
          '1'  => 5,
```

(continues on next page)

(continued from previous page)

```
          0.8  => 6,
          0x1  => 7,
          01   => 8,

          null  => 1,
          ''    => 2,

          false => 1,
          0     => 2,

          '0.8' => 3,
          '01'  => 4,
          '2a'  => 5
          ];

    print_r($a);

/*
The above displays
Array
(
    [1] => 8
    [0] => 2
    [] => 2
    [0.8] => 3
    [01] => 4
    [2a] => 5
)
*/
?>
```

Decimal numbers are rounded to the closest integer; Null is transtyped to '' (empty string); true is 1 and false is 0; Integers in strings are transtyped, while partial numbers or decimals are not analyzed in strings.

As a general rule of thumb, only use integers or strings that don't look like integers.

This analyzer may find constant definitions, when available.

Note also that PHP detects integer inside strings, and silently turn them into integers. Partial and octal numbers are not transformed.

```
<?php
    $a = [1      => 1,
          '2'    => 2,
          '011'  => 9, // octal number
          '11d'  => 11, // partial number
          ];

    var_dump($a);

/*
The above displays
array(4) {
  [1]=>
  int(1)
  [2]=>
  int(2)
```

(continues on next page)

(continued from previous page)

```
  [011]=>
  int(9)
  [11d]=>
  int(11)
}*/
?>
```

See also Arrays syntax.

Suggestions

- Do not use any type but string or integer
- Force typecast the keys when building an array

Specs

Short name	Structures/IndicesAreIntOrString
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Zencart*, *Mautic*

13.2.1093 Infinite Recursion

A method is calling itself, with unchanged arguments. This will probably repeat indefinitely.

This applies to recursive functions without any condition. This also applies to function which inject the incoming arguments, without modifications.

```
<?php

function foo($a, $b) {
    if ($a > 10) {
        return;
    }
    foo($a, $b);
}

function foo2($a, $b) {
    ++$a;   // $a is modified
    if ($a > 10) {
        return;
    }
    foo2($a, $b);
}

?>
```

Suggestions

- Modify arguments before injecting them again in the same method
- Use different values when calling the same method

Specs

Short name	Structures/InfiniteRecursion
Rulesets	*Analyze*
Exakt since	1.8.6
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1094 Invalid Pack Format

Some characters are invalid in a pack() format string.

pack() and unpack() accept the following format specifiers : `aAhHcCsSnviIlLNVqQJPfgGdeExXZ`.

unpack() also accepts a name after the format specifier and an optional quantifier.

All other situations is not a valid, and produces a warning : `pack(): Type t: unknown format code`

```
<?php
    $binarydata = pack(nvc*, 0x1234, 0x5678, 65, 66);

    // the first unsigned short is stored as 'first'. The next matches are names with␣
↪numbers.
    $res = unpack('nfirst/vc*', $binarydata);
?>
```

Check pack() documentation for format specifiers that were introduced in various PHP version, namely 7.0, 7.1 and 7.2.

See also pack and unpack.

Suggestions

- Fix the packing format with correct values

Specs

Short name	Structures/InvalidPackFormat
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.4.9
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1095 Invalid Regex

The PCRE regex doesn't compile. It isn't a valid regex.

Several reasons may lead to this situation : syntax error, Unknown modifier, missing parenthesis or reference.

```
<?php

// valid regex
preg_match('/[abc]/', $string);

// invalid regex (missing terminating ] for character class
preg_match('/[abc/', $string);

?>
```

Regex are check with the Exakat version of PHP.

Dynamic regex are only checked for simple values. Dynamic values may eventually generate a compilation error.

Suggestions

- Fix the regex before running it

Specs

Short name	Structures/InvalidRegex
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.0.5
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*SugarCrm*

13.2.1096 isset() With Constant

Until PHP 7, it was possible to use arrays as constants, but it was not possible to test them with isset.

```
<?php
const X = [1,2,3];

if (isset(X[4])) {}
?>
```

This would yield an error : `Cannot use `isset() <https://www.www.php.net/isset>`_ on the result of an expression (you can use "null !== expression" instead).` This is a backward incompatibility.

Specs

Short name	Structures/IssetWithConstant
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	7.0+
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.1097 Is Actually Zero

This addition actually may be simplified because one term is actually negated by another.

This kind of error happens when the expression is very large : the more terms are included, the more chances are that some auto-annihilation happens.

This error may also be a simple typo : for example, calculating the difference between two consecutive terms.

```
<?php

// This is quite obvious
$a = 2 - 2;

// This is obvious too. This may be a typo-ed difference between two consecutive␣
↪terms.
// Could have been $c = $fx[3][4] - $fx[3][3] or $c = $fx[3][5] - $fx[3][4];
$c = $fx[3][4] - $fx[3][4];

// This is less obvious
$a = $b[3] - $c + $d->foo(1,2,3) + $c + $b[3];

?>
```

Suggestions

- Clean the code and remove the null sum
- Fix one of the variable : this expression needs another variable here
- When adding differences, calculate the difference in a temporary variable first.

Specs

Short name	Structures/IsZero
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.12.15
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Dolibarr*, *SuiteCrm*

13.2.1098 Use json_decode() Options

json_decode() returns objects by default, unless the second argument is set to TRUE or JSON_OBJECT_AS_ARRAY. Then, it returns arrays.

Avoid casting the returned value from json_decode(), and use the second argument to directly set the correct type.

```
<?php

$json = '{a:b}';

// Good syntax
$array = json_decode($json, JSON_OBJECT_AS_ARRAY);

// GoToo much work
$array = (array) json_decode($json);

?>
```

Note that all objects will be turned into arrays, recursively. If you're expecting an array of objects, don't use the JSON_OBJECT_AS_ARRAY constant, and change your JSON code.

Note that JSON_OBJECT_AS_ARRAY is the only constant : there is no defined constant to explicitly ask for an object as returned value.

See also json_decode. .. index::

json

Suggestions

- Use the correct second argument of json_decode() : JSON_OBJECT_AS_ARRAY

Specs

Short name	Structures/JsonWithOption
Rulesets	*Suggestions*
Exakt since	1.4.3
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1099 list() May Omit Variables

Simply omit any unused variable in a list() call.

list() is the only PHP function that accepts to have omitted arguments. If the following code makes no usage of a listed variable, just omit it.

```
<?php
    // No need for '2', so no assignation
    list ($a, , $b) = array(1, 2, 3);

```

(continues on next page)

(continued from previous page)

```
    // works with PHP 7.1 short syntax
    [$a, , $b] = array(1, 2, 3);

    // No need for '2', so no assignation
    list ($a, $c, $b) = array(1, 2, 3);
?>
```

See also list.

Suggestions

- Remove the unused variables from the list call
- When the ignored values are at the beginning or the end of the array, array_slice() may be used to shorten the array.

Specs

Short name	Structures/ListOmissions
Rulesets	*Analyze*, *CI-checks*, *Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high
Examples	*OpenConf*, *FuelCMS*

13.2.1100 Logical Mistakes

Avoid logical mistakes within long expressions.

Sometimes, the logic is not what it seems. It is important to check the actual impact of every part of the logical expression. Do not hesitate to make a table with all possible cases. If those cases are too numerous, it may be time to rethink the whole expression.

```
<?php

// Always true
if ($a != 1 || $a != 2) { }

// $a == 1 is useless
if ($a == 1 || $a != 2) {}

// Always false
if ($a == 1 && $a == 2) {}

// $a != 2 is useless
if ($a == 1 && $a != 2) {}

?>
```

Based on article from `Andrey Karpov` Logical Expressions in C/C++. Mistakes Made by Professionals

Suggestions

- Change the expressions for them to have a real meaning

Specs

Short name	Structures/LogicalMistakes
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Dolibarr*, *Cleverstyle*

13.2.1101 Lone Blocks

Any grouped code without a commanding structure is useless.

Blocks are compulsory when defining a structure, such as a class or a function. They are most often used with flow control instructions, like if then or switch.

Blocks are also valid syntax that group several instructions together, though they have no effect at all, except confuse the reader. Most often, it is a ruin from a previous flow control instruction, whose condition was removed or commented. They should be removed.

```
<?php

    // Lone block
    //foreach($a as $b)
    {
        $b++;
    }
?>
```

Suggestions

- Remove the useless curly brackets

Specs

Short name	Structures/LoneBlock
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*ThinkPHP*, *Tine20*

13.2.1102 Long Arguments

Long arguments should be put in variable, to preserve readability.

When literal arguments are too long, they break the hosting structure by moving the next argument too far on the right. Whenever possible, long arguments should be set in a local variable to keep the readability.

```
<?php

// Now the call to foo() is easier to read.
$reallyBigNumber = <<<BIGNUMBER
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
BIGNUMBER
foo($reallyBigNumber, 2, '12345678901234567890123456789012345678901234567890');

// where are the next arguments ?
foo(
↪'12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567
↪', 2, '12345678901234567890123456789012345678901234567890');

// This is still difficult to read
foo(<<<BIGNUMBER
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
BIGNUMBER
, 2, '12345678901234567890123456789012345678901234567890');

?>
```

Literal strings and heredoc strings, including variables, that are over 50 chars longs are reported here.

Suggestions

- Put the long arguments in a separate variable, and use the variable in the second expression, reducing its total length

Name	Default	Type	Description
codeTooLong	100	integer	Minimum size of a functioncall or a methodcall to be considered too long.

Specs

Short name	Structures/LongArguments
Rulesets	*Analyze*
Exakt since	0.9.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Cleverstyle*, *Contao*

13.2.1103 Too Long A Block

The loop is operating on a block that is too long.

This analysis is applied to loops (for, foreach, while, do..while) and if/then/else/elseif structures.

Then length of a block is managed with the `longBlock` parameter. By default, it is 200 lines, from beginning to the end. Comments are taken into account.

```
<?php

$i = 0;
do {
    // 200 lines of PHP code

    ++$i;
} while($i < 100);

?>
```

Suggestions

- Move the code of the block to an method or a function
- Move part of the code of the block to methods or functions
- Extract repeated patterns and use them

Name	Default	Type	Description
longBlock	200	integer	Size of a block for it to be too long. A block is commanded by a for, foreach, while, do... while, if/then else structure.

Specs

Short name	Structures/LongBlock
Rulesets	*Suggestions*
Exakt since	2.1.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1104 Mail Usage

Report usage of mail from PHP.

The analysis is based on mail() function and various classes used to send mail.

```
<?php
// The message
$message = Line 1\r\nLine 2\r\nLine 3;

// In case any of our lines are larger than 70 characters, we should use wordwrap()
$message = wordwrap($message, 70, \r\n);

```

(continues on next page)

(continued from previous page)

```
// Send
mail('caffeinated@example.com', 'My Subject', $message);
?>
```

See also mail.

Specs

Short name	Structures/MailUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1105 Max Level Of Nesting

Avoid nesting structures too deep, as it hurts readability.

Nesting structures are : if/then, switch, for, foreach, while, do... while. Ternary operator, try/catch are not considered a nesting structures.

Closures, and more generally, functions definitions are counted separatedly.

This analysis checks for 4 levels of nesting, by default. This may be changed by configuration.

```
<?php

// 5 levels of indentation
function foo() {
    if (1) {
        if (2) {
            if (3) {
                if (4) {
                    if (5) {
                        51;
                    } else {
                        5;
                    }
                } else {
                    4;
                }
            } else {
                3;
            }
        } else {
            2;
        }
    } else {
        1;
    }
}
```

(continues on next page)

(continued from previous page)

```
// 2 levels of indentation
function foo() {
    if (1) {
        if (2) {
            // 3 levels of indentation
            return function () {
                if (3) {
                    if (4) {
                        if (5) {
                            51;
                        } else {
                            5;
                        }
                    } else {
                        4;
                    }
                } else {
                    3;
                }
            }
        } else {
            2;
        }
    } else {
        1;
    }
}


?>
```

Suggestions

- Refactor code to avoid nesting
- Export some nested blocks to an external method or function

Name	Default	Type	Description
maxLevel	4	integer	Maximum level of nesting for control flow structures in one scope.

Specs

Short name	Structures/MaxLevelOfIdentation
Rulesets	*Analyze*
Exakt since	1.9.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1106 Mbstring Third Arg

Some mbstring functions use the third argument for offset, not for encoding.

Those are the following functions :

- mb_strrichr()
- mb_stripos()
- mb_strrpos()
- mb_strstr()
- mb_stristr()
- mb_strpos()
- mb_strripos()
- mb_strrchr()
- mb_strrichr()
- mb_substr()

```
<?php

// Display BC
echo mb_substr('ABC', 1 , 2, 'UTF8');

// Yields Warning: mb_substr() expects parameter 3 to be int, string given
// Display 0 (aka, substring from 0, for length (int) 'UTF8' => 0)
echo mb_substr('ABC', 1 ,'UTF8');

?>
```

See also mb_substr() manual pages.

Suggestions

- Add a third argument
- Use the default encoding (aka, omit both third AND fourth argument)

Specs

Short name	Structures/MbstringThirdArg
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1107 Mbstring Unknown Encoding

The encoding used is not known to the ext/mbstring extension.

This analysis takes in charge all `mbstring` encoding and aliases. The full list of supported mbstring encoding is available with mb_list_encodings(). Each encoding alias is available with mb_encoding_aliases().

```
<?php

// Invalid encoding
$str = mb_strtolower($str, 'utf_8');

// Valid encoding
$str = mb_strtolower($str, 'utf8');
$str = mb_strtolower($str, 'UTF8');
$str = mb_strtolower($str, 'UTF-8');

?>
```

See also ext/mbstring.

Suggestions

- Use a valid mbstring encoding

Specs

Short name	Structures/MbstringUnknownEncoding
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1108 mcrypt_create_iv() With Default Values

Avoid using mcrypt_create_iv() default values.

mcrypt_create_iv() used to have `MCRYPT_DEV_RANDOM` as default values, and in PHP 5.6, it now uses `MCRYPT_DEV_URANDOM`.

```
<?php
    $size = mcrypt_get_iv_size(MCRYPT_CAST_256, MCRYPT_MODE_CFB);
    // mcrypt_create_iv is missing the second argument
    $iv = mcrypt_create_iv($size);

// Identical to the line below
//    $iv = mcrypt_create_iv($size, MCRYPT_DEV_RANDOM);

?>
```

If the code doesn't have a second argument, it relies on the default value. It is recommended to set explicitly the value, so has to avoid problems while migrating.

See also mcrypt_create_iv().

Specs

Short name	Structures/McryptcreateivWithoutOption
Rulesets	*CompatibilityPHP70*
Exakt since	0.8.4
Php Version	5.6-
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1109 Merge If Then

Two successive if/then into one, by merging the two conditions.

```
<?php

// two merge conditions
if ($a == 1 && $b == 2) {
    // doSomething()
}

// two distinct conditions
// two nesting
if ($a == 1) {
    if ($b == 2) {
        // doSomething()
    }
}

?>
```

Suggestions

- Merge the two structures into one

Specs

Short name	Structures/MergeIfThen
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1110 Mismatched Ternary Alternatives

A ternary operator should yield the same type on both branches.

Ternary operator applies a condition, and yield two different results. Those results will then be processed by code that expects the same types. It is recommended to match the types on both branches of the ternary operator.

```
<?php

// $object may end up in a very unstable state
$object = ($type == 'Type') ? new $type() : null;

//same result are provided by both alternative, though process is very different
$result = ($type == 'Addition') ? $a + $b : $a * $b;

//Currently, this is omitted
$a = 1;
$result = empty($condition) ? $a : 'default value';
$result = empty($condition) ? $a : getDefaultValue();

?>
```

Suggestions

- Use compatible data type in both branch of the alternative
- Turn the ternary into a if/then, with different processing

Specs

Short name	Structures/MismatchedTernary
Rulesets	*Analyze*, *Suggestions*
Exakt since	0.12.1
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*phpadsnew*, *OpenEMR*

13.2.1111 Missing Cases In Switch

It seems that some cases are missing in this switch structure.

When comparing two different switch() structures, it appears that some cases are missing in one of them. The set of cases are almost identical, but one of the values are missing.

Switch() structures using strings as literals are compared in this analysis. When the discrepancy between two lists is below 25%, both switches are reported.

```
<?php

// This switch operates on a, b, c, d and default
switch($a) {
```

(continues on next page)

(continued from previous page)

```
    case 'a': doSomethingA(); break 1;
    case 'b': doSomethingB(); break 1;
    case 'c': doSomethingC(); break 1;
    case 'd': doSomethingD(); break 1;
    default: doNothing();
}

// This switch operates on a, b, d and default
switch($o->p) {
    case 'a': doSomethingA(); break 1;
    case 'b': doSomethingB(); break 1;

    case 'd': doSomethingD(); break 1;
    default: doNothing();
}

?>
```

In the example, one may argue that the 'c' case is actually handled by the 'default' case. Otherwise, business logic may request that omission.

Suggestions

- Add the missing cases
- Add comments to mention that missing cases are processed in the default case

Specs

Short name	Structures/MissingCases
Rulesets	*Analyze*
Exakt since	0.10.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Tikiwiki*

13.2.1.1112 Maybe Missing New

This functioncall looks like a class instantiation that is missing the new keyword.

Any function definition was found for that function, but a class with that name was. New is probably missing.

```
<?php

// Functioncall
$a = foo();

// Class definition
class foo {}
// Function definition
```

(continues on next page)

(continued from previous page)

```
function foo {}


// Functioncall
$a = BAR;

// Function definition
class bar {}
// Constant definition
const BAR = 1;


?>
```

Suggestions

- Add the new
- Rename the class to distinguish it from the function
- Rename the function to distinguish it from the class

Specs

Short name	Structures/MissingNew
Rulesets	*Analyze*
Exakt since	1.0.4
Php Version	All
Severity	Critical
Time To Fix	Instant (5 mins)
Precision	Medium

13.2.1113 Missing Parenthesis

Add parenthesis to those expression to prevent bugs.

```
<?php

// Missing some parenthesis!!
if (!$a instanceof Stdclass) {
    print Not\n;
} else {
    print Is\n;
}

// Could this addition be actually
$c = -$a + $b;

// This one ?
$c = -($a + $b);

?>
```

See also Operators Precedence.

Specs

Short name	Structures/MissingParenthesis
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.2.6
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.1114 Mixed Concat And Interpolation

Mixed usage of concatenation and string interpolation is error prone. It is harder to read, and leads to overlooking the concatenation or the interpolation.

```
<?php

// Concatenation string
$a = $b . 'c' . $d;

// Interpolation strings
$a = {$b}c{$d};    // regular form
$a = {$b}c$d;      // irregular form

// Mixed Concatenation and Interpolation string
$a = {$b}c . $d;
$a = $b . c$d;
$a = $b . c{$d};

// Mixed Concatenation and Interpolation string with constant
$a = {$b}c . CONSTANT;

?>
```

Fixing this issue has no impact on the output. It makes code less error prone.

There are some situations where using concatenation are compulsory : when using a constant, calling a function, running a complex expression or make use of the escape sequence. You may also consider pushing the storing of such expression in a local variable.

Suggestions

- Only use one type of variable usage : either interpolation, or concatenation

Specs

Short name	Structures/MixedConcatInterpolation
Rulesets	*Analyze*
Exakt since	0.11.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*SuiteCrm*, *Edusoho*

13.2.1115 Modernize Empty With Expression

empty() accepts expressions as argument. This feature was added in PHP 5.5.

There is no need to store the expression in a variable before testing, unless it is reused later.

```
<?php

// PHP 5.5+ empty() usage
if (empty(foo($b . $c))) {
    doSomethingWithoutA();
}

// Compatible empty() usage
$a = foo($b . $c);
if (empty($a)) {
    doSomethingWithoutA();
}

// $a2 is reused, storage is legit
$a2 = strtolower($b . $c);
if (empty($a2)) {
    doSomething();
} else {
    echo $a2;
}

?>
```

See also empty() and empty() supports arbitrary expressions.

Suggestions

- Avoid the temporary variable, and use directly empty()

Specs

Short name	Structures/ModernEmpty
Rulesets	*Analyze*
Exakt since	0.8.6
Php Version	5.5+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1116 Multiple Catch

Indicates if a try structure have several catch statement.

```
<?php

// This try has several catch
try {
    doSomething();
} catch (RuntimeException $e) {
    processRuntimeException();
} catch (OtherException $e) {
    processOtherException();
}

?>
```

Specs

Short name	Structures/MultipleCatch
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1117 Multiples Identical Case

Some cases are defined multiple times, but only one will be processed. Check the list of cases, and remove the extra one.

Exakat tries to find the value of the case as much as possible, and ignore any dynamic cases (using variables).

```
<?php

const A = 1;

case ($x) {
    case 1 :
```

(continues on next page)

(continued from previous page)

```
        break;
    case true:    // This is a duplicate of the previous
        break;
    case 1 + 0:   // This is a duplicate of the previous
        break;
    case 1.0 :    // This is a duplicate of the previous
        break;
    case A :      // The A constant is actually 1
        break;
    case $y  :    // This is not reported.
        break;
    default:

}
?>
```

Suggestions

- Remove the double case
- Change the case to another and rightful value

Specs

Short name	Structures/MultipleDefinedCase
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-duplicate-case
Examples	*SugarCrm*, *ExpressionEngine*

13.2.1118 Multiple Type Variable

Avoid using the same variable with different types of data.

It is recommended to use different names for differently typed data, while processing them. This prevents errors where one believe the variable holds the former type, while it has already been cast to the later.

Incrementing variables, with math operations or concatenation, is OK : the content changes, but not the type. And casting the variable without storing it in itself is OK.

```
<?php

// $x is an array
$x = range('a', 'z');
// $x is now a string
$x = join('', $x);
$c = count($x); // $x is not an array anymore
```

(continues on next page)

(continued from previous page)

```
// $letters is an array
$letters = range('a', 'z');
// $alphabet is a string
$alphabet = join('', $letters);

// Here, $letters is cast by PHP, but the variable is changed.
if ($letters) {
    $count = count($letters); // $letters is still an array
}

?>
```

Suggestions

- Use a class that accepts one type of argument, and exports another type of argument.
- Use different variable for each type of data format : $rows (for array), $list (for implode('', $rows))
- Pass the final result as argument to another method, avoiding the temporary variable

Specs

Short name	Structures/MultipleTypeVariable
Rulesets	*Analyze*
Exakt since	0.12.15
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Typo3*, *Vanilla*

13.2.1119 Multiple Unset()

Unset() accepts multiple arguments, unsetting them one after each other. It is more efficient to call unset() once, than multiple times.

```
<?php

// One call to unset only
unset($a, $b, $c, $d);

// Too many calls to unset
unset($a);
unset($b);
unset($c);
unset($d);

?>
```

See also unset.

Suggestions

- Merge all unset into one call

Specs

Short name	Structures/MultipleUnset
Rulesets	*Suggestions*, *php-cs-fixable*
Exakt since	1.7.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1120 Multiply By One

Multiplying by 1 is a fancy type cast.

If it is used to type cast a value to number, then casting (int) or (float) is clearer. This behavior may change with PHP 7.1, which has unified the behavior of all hidden casts.

```
<?php

// Still the same value than $m, but now cast to integer or float
$m = $m * 1;

// Still the same value than $m, but now cast to integer or float
$n *= 1;

// make typecasting clear, and merge it with the producing call.
$n = (int) $n;

?>
```

See also Type Juggling

Suggestions

- Typecast to (int) or (float) for better readability
- Skip useless math operation altogether

Specs

Short name	Structures/MultiplyByOne
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-useless-math
Examples	*SugarCrm*, *Edusoho*

13.2.1121 Named Regex

Captured subpatterns may be named, for easier reference.

From the manual : It is possible to name a subpattern using the syntax `(?P<name>pattern)`. This subpattern will then be indexed in the matches array by its normal numeric position and also by name. PHP 5.2.2 introduced two alternative syntaxes `(?<name>pattern)` and `(?'name'pattern)`.

Naming subpatterns makes it easier to know what is read from the results of the subpattern : for example, `$r['name']` has more meaning than `$r[1]`.

Named subpatterns may also be shifted in the regex without impact on the resulting array.

```
<?php

$x = 'abc';
preg_match_all('/(?<name>a)/', $x, $r);
print_r($r[1]);
print_r($r['name']);

preg_match("/(?<name>a)(?'sub'b)/", $x, $s);
print $s[2];
print $s['sub'];

?>
```

See also Subpatterns.

Suggestions

- Use named regex, and stop using integer-named subpatterns

Specs

Short name	Structures/NamedRegex
Rulesets	*Suggestions*
Exakt since	1.4.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Phinx*, *shopware*

13.2.1122 Negative Power

The power operator ** has higher precedence than the sign operators + and -.

This means that -2 ** 2 == -4. It is in fact, -(2 ** 2).

When using negative power, it is clearer to add parenthesis or to use the pow() function, which has no such ambiguity :

```
<?php

// -2 to the power of 2 (a square)
pow(-2, 2) == 4;

// minus 2 to the power of 2 (a negative square)
-2 ** 2 == -(2 ** 2) == 4;

?>
```

Suggestions

- Avoid negative number, as operands of **
- Use parenthesis with negative numbers and **

Specs

Short name	Structures/NegativePow
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.1123 Nested Ifthen

Three levels of ifthen is too much. The method should be split into smaller functions.

```php
<?php

function foo($a, $b) {
    if ($a == 1) {
        // Second level, possibly too much already
        if ($b == 2) {

        }
    }
}

function bar($a, $b, $c) {
    if ($a == 1) {
        // Second level.
        if ($b == 2) {
            // Third level level.
            if ($c == 3) {
                // Too much
            }
        }
    }
}

?>
```

Name	Default	Type	Description
nestedIfthen	3	integer	Maximal number of acceptable nesting of if-then structures

Specs

Short name	Structures/NestedIfthen
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*LiveZilla*, *MediaWiki*

13.2.1124 Nested Loops

Nested loops happens when a loop (while, do..while, for, foreach), is used inside another loop.

```php
<?php

// Nested loops
foreach($array as $a) {
    foreach ($letters as $b) {
        // This is performed count($array) * count($letters) times.
        doSomething();
    }
}
```

(continues on next page)

(continued from previous page)

```
?>
```

Such structure tends to require a lot of processing, as the size of both loops have to be multiplied to estimate the actual payload. They should be avoided as much as possible. This may no be always possible, though.

Nested loops are worth a check for performances reasons, as they will process a lot of times the same instructions.

Specs

Short name	Structures/NestedLoops
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1125 Nested Ternary

Ternary operators should not be nested too deep.

They are a convenient instruction to apply some condition, and avoid a if() structure. It works best when it is simple, like in a one liner.

However, ternary operators tends to make the syntax very difficult to read when they are nested. It is then recommended to use an if() structure, and make the whole code readable.

```
<?php

// Simple ternary expression
echo ($a == 1 ? $b : $c) ;

// Nested ternary expressions
echo ($a === 1 ? $d === 2 ? $b : $d : $d === 3 ? $e : $c) ;
echo ($a === 1 ? $d === 2 ? $f ===4 ? $g : $h : $d : $d === 3 ? $e : $i === 5 ? $j :
↪$k) ;

//Previous expressions, written as a if / Then expression
if ($a === 1) {
    if ($d === 2) {
        echo $b;
    } else {
        echo $d;
    }
} else {
    if ($d === 3) {
        echo $e;
    } else {
        echo $c;
    }
}
```

(continues on next page)

(continued from previous page)

```
if ($a === 1) {
    if ($d === 2) {
        if ($f === 4) {
            echo $g;
        } else {
            echo $h;
        }
    } else {
        echo $d;
    }
} else {
    if ($d === 3) {
        echo $e;
    } else {
        if ($i === 5) {
            echo $j;
        } else {
            echo $k;
        }
    }
}

?>
```

See also Nested Ternaries are Great.

Suggestions

- Replace ternaries by if/then structures.
- Replace ternaries by a functioncall : this provides more readability, offset the actual code, and gives room for making it different.

Specs

Short name	Structures/NestedTernary
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-nested-ternary
Examples	*SPIP*, *Zencart*

13.2.1126 Always Positive Comparison

Some PHP native functions, such as count(), strlen(), or abs() only returns positive or null values.

When comparing them to 0, the following expressions are always true and should be avoided.

```
<?php

$a = [1, 2, 3];

var_dump(count($a) >= 0);
var_dump(count($a) < 0);

?>
```

Suggestions

- Compare count() to non-zero values
- Use empty()

Specs

Short name	Structures/NeverNegative
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Magento*

13.2.1127 New Line Style

New lines may be written with the sequence n or with the constant PHP_EOL.

When one of those alternatives is used over 90% of the time, it is considered as standard : the remaining are reported.

n are only located when used alone, in n (including the double quotes). When n is used inside a double-quoted string, its replacement with PHP_EOL would be cumbersome : as such, they are ignored by this analyzer.

```
<?php

// This may be repeated over 10 times
$a = PHP is a great language\n;
$a .= \n;

// This only appears once in the code : this line is reported.
$b = $a.PHP_EOL.$c;

?>
```

Specs

Short name	Structures/NewLineStyle
Rulesets	none
Exakt since	0.9.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1128 Next Month Trap

Avoid using +1 month with strtotime().

strtotime() calculates the next month by incrementing the month number. For day number that do not exist from one month to the next, strtotime() fixes them by setting them in the next-next month.

This happens to January, March, May, July, August and October. January is also vulnerable for 29 (not every year), 30 and 31.

Avoid using '+1 month', and rely on 'first day of next month' or 'last day of next month' to extract the next month's name.

```
<?php

// Base date is October 31 => 10/31
// +1 month adds +1 to 10 => 11/31
// Since November 31rst doesn't exists, it is corrected to 12/01.
echo date('F', strtotime('+1 month',mktime(0,0,0,$i,31,2017))).PHP_EOL;

// Base date is October 31 => 10/31
echo date('F', strtotime('first day of next month',mktime(0,0,0,$i,31,2017))).PHP_EOL;

?>
```

See also It is the 31st again.

Suggestions

- Review strtotime() usage for month additions
- Use datetime() and other classes, not PHP native functions
- Use a external library, like carbon, to handle dates

Specs

Short name	Structures/NextMonthTrap
Rulesets	*Analyze*, *CI-checks*, *Top10*
Exakt since	1.0.1
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Contao*, *Edusoho*

13.2.1129 No Append On Source

Do not append new elements to an array in a foreach loop. Since PHP 7.0, the array is still used as a source, and will be augmented, and used again.

```
<?php

// Relying on the initial copy
$a = [1];
$initial = $a;
foreach($initial as $v) {
    $a[] = $v + 1;
}

// Keep new results aside
$a = [1];
$tmp = [];
foreach($a as $v) {
    $tmp[] = $v + 1;
}
$a = array_merge($a, $tmp);
unset($tmp);

// Example, courtesy of Frederic Bouchery
// This is an infinite loop
$a = [1];
foreach($a as $v) {
    $a[] = $v + 1;
}

?>
```

Thanks to Frederic Bouchery for the reminder.

See also foreach and What will this code return? #PHP.

Suggestions

- Use a copy of the source, to avoid modifying it during the loop
- Store the new values in a separate storage

Specs

Short name	Structures/NoAppendOnSource
Rulesets	*Analyze*
Exakt since	1.8.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1130 Avoid array_unique()

The native function array_unique() is much slower than using other alternatives, such as array_count_values(), array_flip()/array_keys(), or even a foreach() loops.

```
<?php

// using array_unique()
$uniques = array_unique($someValues);

// When values are strings or integers
$uniques = array_keys(array_count_values($someValues));
$uniques = array_flip(array_flip($someValues))

//even some loops are faster.
$uniques = [];
foreach($someValues as $s) {
    if (!in_array($uniques, $s)) {
        $uniques[] $s;
    }
}

?>
```

See also array_unique.

Suggestions

- Upgrade to PHP 7.2
- Use an alternative way to make values unique in an array, using array_count_values(), for example.

Specs

Short name	Structures/NoArrayUnique
Rulesets	*Performances*
Exakt since	0.8.4
Php Version	7.2-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1131 Avoid Large Array Assignation

Avoid setting large arrays to local variables. This is done every time the function is called.

There are different ways to avoid this : inject the array, build the array once. Using an constant or even a global variable is faster.

The effect on small arrays (less than 10 elements) is not significant. Arrays with 10 elements or more are reported here. The effect is also more important on functions that are called often, or within loops.

```
<?php

// with constants, for functions
const ARRAY = array(1,2,3,4,5,6,7,8,9,10,11);
function foo() {
    $array = ARRAY;
    //more code
}

// with class constants, for methods
class x {
    const ARRAY = array(1,2,3,4,5,6,7,8,9,10,11);
    function foo() {
        $array = self::ARRAY;
        //more code
    }
}

// with properties, for methods
class x {
    private $array = array(1,2,3,4,5,6,7,8,9,10,11);

    function foo() {
        $array = $this->array;
        //more code
    }
}

// injection, leveraging default values
function foo($array = array(1,2,3,4,5,6,7,8,9,10,11)) {
    //more code
}

// local cache with static
function foo() {
    static $array;
    if ($array === null) {
        $array = array(1,2,3,4,5,6,7,8,9,10,11);
    }

    //more code
}

// Avoid creating the same array all the time in a function
class x {
    function foo() {
        // assign to non local variable is OK.
        // Here, to a property, though it may be better in a __construct or as␣
↪default values
```

(continues on next page)

(continued from previous page)

```
        $this->s = array(1,2,3,4,5,6,7,8,9,10,11);

        // This is wasting resources, as it is done each time.
        $array = array(1,2,3,4,5,6,7,8,9,10,11);
    }
}

?>
```

Suggestions

- Make the literal a global constant or a class constant
- Make the literal an argument, so it is injected
- Make the literal an property, with it as default value

Specs

Short name	Structures/NoAssignationInFunction
Rulesets	*Performances*
Exakt since	0.9.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1132 Don't Change Incomings

PHP hands over a lot of information using special variables like $_GET, $_POST, etc... Modifying those variables and those values inside variables means that the original content is lost, while it will still look like raw data, and, as such, will be untrustworthy.

```
<?php

// filtering and keeping the incoming value.
$_DATA'id'] = (int) $_GET['id'];

// filtering and changing the incoming value.
$_GET['id'] = strtolower($_GET['id']);

?>
```

It is recommended to put the modified values in another variable, and keep the original one intact.

Specs

Short name	Structures/NoChangeIncomingVariables
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1133 No Choice

A conditional structure is being used, but both alternatives are the same, leading to the illusion of choice.

Either the condition is useless, and may be removed, or the alternatives need to be distinguished.

```
<?php

if ($condition == 2) {
    doSomething();
} else {
    doSomething();
}

$condition == 2 ?     doSomething() :     doSomething();

?>
```

Suggestions

- Remove the conditional, and call the expression directly
- Replace one of the alternative with a distinct call
- Remove the whole conditional : it may end up being useless

Specs

Short name	Structures/NoChoice
Rulesets	*Analyze*, *CI-checks*, *Top10*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
Examples	*NextCloud*, *Zencart*

13.2.1134 No Direct Access

This expression protects files against direct access. It will kill the process if it realizes this is not supposed to be directly accessed.

Those expressions are used in applications and framework, to prevent direct access to definition files.

```
<?php

  // CONSTANT_EXEC is defined in the main file of the application
  defined('CONSTANT_EXEC') or die('Access not allowed'); : Constant used!

?>
```

Specs

Short name	Structures/NoDirectAccess
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1135 No Direct Usage

The results of the following functions shouldn't be used directly, but checked first.

For example, glob() returns an array, unless some error happens, in which case it returns a boolean (false). In such case, however rare it is, plugging glob() directly in a foreach() loops will yield errors.

```
<?php
    // Used without check :
    foreach(glob('.') as $file) { /* do Something */ }.

    // Used without check :
    $files = glob('.');
    if (!is_array($files)) {
        foreach($files as $file) { /* do Something */ }.
    }
?>
```

Suggestions

- Check the return of the function before using it, in particular for false, or array().

Specs

Short name	Structures/NoDirectUsage
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Edusoho*, *XOOPS*

13.2.1136 No Empty Regex

PHP regex don't accept empty regex, nor regex with alphanumeric delimiter.

Most of those errors happen at execution time, when the regex is build dynamically, but still may end empty. At compile time, such error are made when the code is not tested before commit.

```
<?php

// No empty regex
preg_match('', $string, $r);

// Delimiter must be non-alphanumerical
preg_replace('1abc1', $string, $r);

// Delimiter must be non-alphanumerical
preg_replace('1'.$regex.'1', $string, $r);

?>
```

See also PCRE and Delimiters.

Suggestions

- Fix the regex by adding regex delimiters

Specs

Short name	Structures/NoEmptyRegex
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.11.1
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Tikiwiki*

13.2.1137 No get_class() With Null

It is not possible to pass explicitly null to get_class() to get the current's class name. Since PHP 7.2, one must call get_class() without arguments to achieve that result.

```
<?php

class A {
  public function f() {
    // Gets the classname
    $classname = get_class();

    // Gets the classname and a warning
    $classname = get_class(null);
  }
}

$a = new A();
$a->f('get_class');

?>
```

Specs

Short name	Structures/NoGetClassNull
Rule-sets	*Analyze*, *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*
Exakt since	1.0.4
Php Version	All
Sever-ity	Major
Time To Fix	Instant (5 mins)
Preci-sion	High

13.2.1138 No Hardcoded Hash

Hash should never be hardcoded.

Hashes may be MD5, SHA1, SHA512, Bcrypt or any other. Such values must be easily changed, for security reasons, and the source code is not the safest place to hide it.

```
<?php

    // Those strings may be sha512 hashes.
    // it is recomemdned to check if they are static or should be put into␣
↪configuration
    $init512 = array( // initial values for SHA512
        '6a09e667f3bcc908', 'bb67ae8584caa73b', '3c6ef372fe94f82b', 'a54ff53a5f1d36f1
↪',
```

(continues on next page)

(continued from previous page)

```
    );

    // strings which are obvious conversion are ignored
    $decimal = intval('87878877', 12);
?>
```

See also Salted Password Hashing - Doing it Right and Hash-Buster.

Suggestions

- Put any hardcoded hash in a configuration file, a database or a environment variable. An external source.

Specs

Short name	Structures/NoHardcodedHash
Rulesets	*Analyze*, *Security*
Exakt since	0.8.4
Php Version	All
Severity	Critical
Time To Fix	Slow (1 hour)
Precision	High
Examples	*shopware*, *SugarCrm*

13.2.1139 No Hardcoded Ip

Do not leave hard coded IP in your code.

It is recommended to move such configuration in external files or databases, for each update. This may also come handy when testing.

```
<?php

// This IPv4 is hardcoded.
$ip = '183.207.224.50';
// This IPv6 is hardcoded.
$ip = '2001:0db8:85a3:0000:0000:8a2e:0370:7334';

// This looks like an IP
$thisIsNotAnIP = '213.187.99.50';
$thisIsNotAnIP = '2133:1387:9393:5330';

?>
```

`127.0.0.1`, `\:\:1` and `\:\:0` are omitted, and not considered as a violation.

See also Use of Hardcoded IPv4 Addresses and Never hard code sensitive information.

Suggestions

- Move the hardcoded IP to an external source : environment variable, configuration file, database.
- Remove the hardcoded IP and ask for it at execution.

- Use a literal value for default messages in form.

Specs

Short name	Structures/NoHardcodedIp
Rulesets	*Analyze*, *Security*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*OpenEMR*, *NextCloud*

13.2.1140 No Hardcoded Path

> It is not recommended to use hardcoded literals when designating files. Full paths are usually tied to one file system organization. As soon as the organisation changes or must be adapted to any external constraint, the path is not valid anymore.

Either use __FILE__ and __DIR__ to make the path relative to the current file; use a `DOC_ROOT` as a configuration constant that will allow the moving of the script to another folder; finally functions like sys_get_temp_dir() produce a viable temporary folder.

Relative paths are relative to the current execution directory, and not the current file. This means they may differ depending on the location of the start of the application, and are sensitive to chdir() and chroot() usage.

```
<?php

    // This depends on the current executed script
    file_get_contents('token.txt');

    // Exotic protocols are ignored
    file_get_contents('jackalope://file.txt');

    // Some protocols are ignored : http, https, ftp, ssh2, php (with memory)
    file_get_contents('http://www.php.net/');
    file_get_contents('php://memory/');

    // glob() with special chars * and ? are not reported
    glob('./*/foo/bar?.txt');
    // glob() without special chars * and ? are reported
    glob('/foo/bar/');

?>
```

Suggestions

- Add __DIR__ before the path to make it relative to the current file
- Add a configured prefix before the path to point to any file in the system
- Use sys_get_temp_dir() for temporary data

- Use `include_path` argument function, such as fie_get_contents(), to have the file located in configurable directories.

Specs

Short name	Structures/NoHardcodedPath
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
ClearPHP	no-hardcoded-path
Examples	*Tine20*, *Thelia*

13.2.1141 No Hardcoded Port

When connecting to a remove server, port is an important information. It is recommended to make this configurable (with constant or configuration), to as to be able to change this value without changing the code.

```
<?php

    // Both configurable IP and hostname
    $connection = ssh2_connect($_ENV['SSH_HOST'], $_ENV['SSH_PORT'], $methods,
↪$callbacks);

    // Both hardcoded IP and hostname
    $connection = ssh2_connect('shell.example.com', 22, $methods, $callbacks);

    if (!$connection) die('Connection failed');
?>
```

Suggestions

- Move the port to a configuration file, an environment variable

Specs

Short name	Structures/NoHardcodedPort
Rulesets	*Analyze*, *Security*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*WordPress*

13.2.1142 No isset() With empty()

empty() actually does the job of isset() too.

From the manual : `No warning is generated if the variable does not exist. That means `empty() <https://www.php.net/empty>`_ is essentially the concise equivalent to !`isset(<https://www.www.php.net/isset>`_$var) || $var == false`. The main difference is that isset() only works with variables, while empty() works with other structures, such as constants.

```
<?php


// Enough validation
if (!empty($a)) {
    doSomething();
}

// Too many tests
if (isset($a) && !empty($a)) {
    doSomething();
}

?>
```

See also Isset <http://www.php.net/‘isset>‘_ and empty.

Suggestions

- Only use isset(), just drop the empty()
- Only use empty(), just drop the empty()
- Use a null value, so the variable is always set

Specs

Short name	Structures/NoIssetWithEmpty
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.7
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*XOOPS*

13.2.1143 Non Breakable Space In Names

PHP allows non-breakable spaces in structures names, such as class, interfaces, traits, and variables.

This may be a nice trick to make names more readable outside code context, like long-named methods for tests.

```
<?php

class class with non breakable spaces {}

class ClassWithoutNonBreakableSpaces {}

?>
```

See also the original post by Matthieu Napoli : Using non-breakable spaces in test method names and PHP Variable Names.

Specs

Short name	Structures/NonBreakableSpaceInNames
Rulesets	*CE*
Exakt since	0.12.0
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1144 No Need For Else

Else is not needed when the Then ends with a break. A break may be the following keywords : break, continue, return, goto. Any of these send the execution somewhere in the code. The else block is then executed as the main sequence, only if the condition fails.

```
<?php

function foo() {
    // Else may be in the main sequence.
    if ($a1) {
        return $a1;
    } else {
        $a++;
    }

    // Same as above, but negate the condition : if (!$a2) { return $a2; }
    if ($a2) {
        $a++;
    } else {
        return $a2;
    }

    // This is OK
    if ($a3) {
        return;
    }

    // This has no break
    if ($a4) {
        $a++;
    } else {
```

(continues on next page)

(continued from previous page)

```
        $b++;
    }

    // This has no else
    if ($a5) {
        $a++;
    }
}
?>
```

See also Object Calisthenics, rule # 2.

Suggestions

- Remove else block, but keep the code

Specs

Short name	Structures/NoNeedForElse
Rulesets	*Analyze*
Exakt since	0.10.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Thelia*, *ThinkPHP*

13.2.1145 No Need For Triple Equal

There is no need for the identity comparison when the methods returns the proper type.

```
<?php

// foo() returns a string.
if ('a' === foo()) {
    // doSomething()
}


function foo() : string {
    return 'a';
}

?>
```

Suggestions

-

Specs

Short name	Structures/NoNeedForTriple
Rulesets	*Analyze*
Exakt since	2.1.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1146 No Need For get_class()

There is no need to call get_class() to build a static call. The argument of get_class() may be used directly.

```
<?php

//
$a->b::$c

// This is too much code
get_class($a->b)::$c

?>
```

See also Scope Resolution Operator (::).

Suggestions

- Use get_called_class(), which may carry different class names
- Use self, static or parent keywords, if you are already in the current class
- Use the argument of get_class() directly

Specs

Short name	Structures/NoNeedGetClass
Rulesets	*Suggestions*
Exakt since	1.8.1
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1147 No Parenthesis For Language Construct

Some PHP language constructs, such are `include`, `require`, `include_once`, `require_once`, `print`, `echo` don't need parenthesis. They accept parenthesis, but it is may lead to strange situations.

```
<?php

// This is an attempt to load 'foo.inc', or kill the script
include('foo.inc') or die();
// in fact, this is read by PHP as : include 1
// include  'foo.inc' or die();

?>
```

It it better to avoid using parenthesis with `echo`, `print`, `return`, `throw`, `yield`, `yield from`, `include`, `require`, `include_once`, `require_once`.

See also include.

Suggestions

- Remove parenthesis

Specs

Short name	Structures/NoParenthesisForLanguageConstruct
Rulesets	*Analyze*, *CI-checks*, *Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-parenthesis-for-language-construct
Examples	*Phpdocumentor*, *phpMyAdmin*

13.2.1148 No Reference On Left Side

Do not use references as the right element in an assignation.

```
<?php

$b = 2;
$c = 3;

$a = &$b + $c;
// $a === 2 === $b;

$a = $b + $c;
// $a === 5

?>
```

This is the case for most situations : addition, multiplication, bitshift, logical, power, concatenation. Note that PHP won't compile the code if the operator is a short operator (+=, .=, etc.), nor if the & is on the right side of the operator.

Specs

Short name	Structures/NoReferenceOnLeft
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.11.5
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.1149 No Return Or Throw In Finally

Avoid using return and throw in a finally block. Both command will interrupt the processing of the try catch block, and any exception that was emitted will not be processed. This leads to unprocessed exceptions, leaving the application in an unstable state.

Note that PHP prevents the usage of goto, break and continue within the finally block at linting phase. This is categorized as a Security problem.

```
<?php
function foo() {
        try {
            // Exception is thrown here
            throw new \Exception();
        } catch (Exception $e) {
            // This is executed AFTER finally
            return 'Exception';
        } finally {
            // This is executed BEFORE catch
            return 'Finally';
        }
    }
}

// Displays 'Finally'. No exception
echo foo();

function bar() {
        try {
            // Exception is thrown here
            throw new \Exception();
        } catch (Exception $e) {
            // Process the exception.
            return 'Exception';
        } finally {
            // clean the current situation
            // Keep running the current function
        }
        return 'Finally';
    }
}

// Displays 'Exception', with processed Exception
echo bar();
```

(continues on next page)

(continued from previous page)

```
?>
```

See also Return Inside Finally Block.

Suggestions

- Move the return right after the try/catch/finally call

Specs

Short name	Structures/NoReturnInFinally
Rulesets	*Security*
Exakt since	0.12.1
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1150 @ Operator

@ is the 'no scream' operator : it suppresses error output.

```
<?php

// Set x with incoming value, or else null.
$x = @$_GET['x'];

?>
```

This operator is actually very slow : it will process the error all the way up, and finally decide not to display it. It is often faster to check the conditions first, then run the method without `@`.

You may also set display_error to 0 in the `php.ini` : this will avoid user's error display, but will keep the error in the PHP logs, for later processing.

The only situation where `@` is useful is when a native PHP function displays errors messages when error happens and there is no way to check it from the code.

This is the case with fopen(), stream_socket_server(), token_get_all().

See also Error Control Operators and Five reasons why the shut-op operator should be avoided.

Suggestions

- Remove the @ operator by default

Name	Default	Type	Description
authorizedFunctions	noscream_functions.json	data	Functions that are authorized to sports a @.

Specs

Short name	Structures/Noscream
Rulesets	*Analyze*, *CE*, *CI-checks*, *Performances*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high
ClearPHP	no-noscream
Examples	*Phinx*, *PhpIPAM*

13.2.1151 Avoid Substr() One

Use array notation `$string[$position]` to reach a single byte in a string.

There are two ways to access a byte in a string : substr() and `$v[$pos]`.

The second style is more readable. It may be up to four times faster, though it is a micro-optimization. It is recommended to use it.

PHP 7.1 also introduces the support of negative offsets as string index : negative offset are also reported.

```
<?php

$string = 'abcde';

echo substr($string, $pos, 1);
echo $string[$pos];

echo mb_substr($string, $pos, 1);

// when $pos = 1
// displays bbb
// when $pos = 2
// displays ??

?>
```

Beware that substr() and `$v[$pos]` are similar, while mb_substr() is not. The first function works on bytes, while the latter works on characters.

Suggestions

- Replace substr() with the array notations for strings.
- Replace substr() with a call to mb_substr().

Specs

Short name	Structures/NoSubstrOne
Rulesets	*Analyze*, *CI-checks*, *CompatibilityPHP71*, *Performances*, *Suggestions*, *Top10*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*ChurchCRM*, *LiveZilla*

13.2.1152 Not Equal Is Not !==

Not and Equal operators, used separately, don't amount to the different operator `!==`.

`!$a == $b` first turns `$a``into the opposite boolean, then compares this boolean value to ``$b`. On the other hand, `$a !== $b` compares the two variables for type and value, and returns a boolean.

```
<?php

if ($string != 'abc') {
    // doSomething()
}

// Here, string will be an boolean, leading
if (!$string == 'abc') {
    // doSomething()
}

// operator priority may be confusing
if (!$object instanceof OneClass) {
    // doSomething()
}
?>
```

Note that the `instanceof` operator may be use with this syntax, due to operator precedence.

See also Operator Precedence.

Suggestions

- Use the != or !==
- Use parenthesis

Specs

Short name	Structures/NotEqual
Rulesets	*Analyze*, *CI-checks*
Exakt since	2.0.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1153 Not Not

Double not makes a boolean, not a `true`.

This is a wrong casting to boolean. PHP supports `(boolean)` to do the same, faster and cleaner.

```
<?php
    // Explicit code
    $b = (boolean) $x;
    $b = (bool) $x;

    // Wrong type casting
    $b = !!$x;

?>
```

See also Logical Operators and Type Juggling.

Suggestions

- Use `(bool)` casting operator for that
- Don't typecast, and let PHP handle it. This works in situations where the boolean is immediately used.

Specs

Short name	Structures/NotNot
Rulesets	*Analyze*, *CE*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-implied-cast
Examples	*Cleverstyle*, *Tine20*

13.2.1154 Not Or Tilde

There are two NOT operator in PHP : `!` and `~`. The first is a logical operator, and returns a boolean. The second is a bit-wise operator, and flips each bit.

Although they are distinct operations, there are situations where they provide the same results. In particular, when processing booleans.

Yet, ! and ~ are not the same. ~ has a higher priority, and will not yield to `instanceof`, while ! does.

The analyzed code has less than 10% of one of them : for consistency reasons, it is recommended to make them all the same.

```
<?php

// be consistent
if (!$condition) {
    doSomething();
}

if (~$condition) {
    doSomething();
}

?>
```

See also Bitwise Operators, Logical Operators and Operators Precedences .

Suggestions

- Be consistent

Specs

Short name	Structures/NotOrNot
Rulesets	none
Exakt since	1.8.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1155 Variable Is Not A Condition

> Avoid using a lone variable as a condition. It is recommended to use a comparative value, or one of the filtering function, such as isset(), empty().

Using the raw variable as a condition blurs the difference between an undefined variable and an empty value. By using an explicit comparison or validation function, it is easier to understand what the variable stands for.

```
<?php

if (isset($error)) {
    echo 'Found one error : '.$error!;
}

//
if ($errors) {
```

(continues on next page)

(continued from previous page)

```
    print count($errors).' errors found : '.join('', $errors).PHP_EOL;
    echo 'Not found';
}

?>
```

Thanks to the PMB team for the inspiration.

Suggestions

- Make the validation explicit, by using a comparison operator, or one of the validation function.

Specs

Short name	Structures/NoVariableIsACondition
Rulesets	*Analyze*
Exakt since	1.6.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1156 Objects Don't Need References

There is no need to create references for objects, as those are always passed by reference when used as arguments.

Note that when the argument is assigned another value, including another object, then the reference is needed : PHP forgets about reference when they are replaced.

```
<?php

    $object = new stdClass();
    $object->name = 'a';

    foo($object);
    print $object->name; // Name is 'b'

    // No need to make $o a reference
    function foo(&$o) {
        $o->name = 'b';
    }


    // $o is assigned inside the function : it must be called with a &, or the object␣
↪won't make it out of the foo3 scope
    function foo3(&$o) {
        $o = new stdClass;
    }

    $array = array($object);
```

(continues on next page)

(continued from previous page)

```
    foreach($array as &$o) { // No need to make this a reference
        $o->name = 'c';
    }

?>
```

See also Passing by reference.

Suggestions

- Remove the reference
- Assign the argument with a new value

Specs

Short name	Structures/ObjectReferences
Rulesets	*Analyze*, *CI-checks*, *Top10*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-references-on-objects
Examples	*Zencart*, *XOOPS*

13.2.1157 include_once() Usage

include_once() and require_once() functions should be avoided for performances reasons.

```
<?php

// Including a library.
include 'lib/helpers.inc';

// Including a library, and avoiding double inclusion
include_once 'lib/helpers.inc';

?>
```

Try using autoload for loading classes, or use include() or require() and make it possible to include several times the same file without errors.

Suggestions

- Avoid using include_once() whenever possible
- Use autoload() to load classes, and avoid loading them with include

Specs

Short name	Structures/OnceUsage
Rulesets	*Analyze*, *CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*XOOPS*, *Tikiwiki*

13.2.1158 One Dot Or Object Operator Per Line

Rule #4 of Object Calisthenics : Only one -> or . per line.

```
<?php

// Those should be on different lines for readability
$a->foo()->bar()->getFinal();

$a->foo()
  ->bar()
  ->getFinal();

// Those should be on different lines for readability
$concatenation = 'a' . 'b' . $c . 'd';

$concatenation = 'a' .
                 'b' .
                 $c .
                 'd';

?>
```

This analysis will also catch the following cases :

```
<?php
    // set of multiples (concatenations or properties or methodcalls)
    foo('a' . 'b', 'c'. 'd');
    foo('a' . 'b', $c->d);

?>
```

When kept, simple, this rule has some edge cases which are left to the reader.

```
<?php

$a = 'a' . 'b'
     .
     'c' . 'd';
$c = $f->g('e' . 'f');

$e = A::B::D;

?>
```

Specs

Short name	Structures/OneDotOrObjectOperatorPerLine
Rulesets	none
Exakt since	0.8.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1159 One Expression Brackets Consistency

Brackets around one-line expressions are not consistent.

PHP makes bracket optional when a control structure pilot only one expression. Both are semantically identical.

This analysis reports code that uses brackets while the vast majority of other expressions uses none. Or the contrary.

```
<?php

// One expression with brackets
for($i = 0; $i < 10; $i++) { $c++; }

// One expression without bracket
for($i2 = 0; $i2 < 10; $i2++)  $c++;

?>
```

Another analysis, [Structures/Bracketless], reports the absence of brackets as an error.

Specs

Short name	Structures/OneExpressionBracketsConsistency
Rulesets	none
Exakt since	0.9.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1160 One If Is Sufficient

Nested conditions may be written another way, and reduce the amount of code.

Nested conditions are equivalent to a `&&` condition. As such, they may be switched. When one of the condition has no explicit else, then it is lighter to write it as the first condition. This way, it is written once, and not repeated.

```
<?php

// Less conditions are written here.
     if($b == 2) {
```

(continues on next page)

(continued from previous page)

```
        if($a == 1) {
             ++$c;
     }
        else {
             ++$d;
     }
    }

// ($b == 2) is double here
    if($a == 1) {
     if($b == 2) {
             ++$c;
     }
    }
    else {
     if($b == 2) {
             ++$d;
     }
    }
?>
```

Suggestions

- Switch the if... then conditions, to reduce the amount of conditions to read.

Specs

Short name	Structures/OneIfIsSufficient
Rulesets	*Suggestions*
Exakt since	1.2.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Tikiwiki*

13.2.1161 More Than One Level Of Indentation

According to PHP Object Calisthenics, one level of indentation is sufficient.

It helps to abide by the Single Responsibility rule and increase reuse.

```
<?php

class foo {
    function multipleLevels($array) {
        $return = array();
        foreach($array as $b) {

            // This is a second level of indentation
            if ($this->check($b)) { continue; }
```

(continues on next page)

(continued from previous page)

```
            $return[] = $b;
        }
        return $return;
    }

    function oneLevel($array) {
        $return = array_filter($array, array($this, 'check'));
        return $return;
    }

}

?>
```

Specs

Short name	Structures/OneLevelOfIndentation
Rulesets	none
Exakt since	0.8.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1162 Several Instructions On The Same Line

Usually, instructions do not share their line : one instruction, one line.

This is good for readability, and help at understanding the code. This is especially important when fast-reading the code to find some special situation, where such double-meaning line way have an impact.

```
<?php

switch ($x) {
    // Is it a fallthrough or not ?
    case 1:
        doSomething(); break;

    // Easily spotted break.
    case 1:
        doSomethingElse();
        break;

    default :
        doDefault();
        break;
}

?>
```

See also Object Calisthenics, rule # 5.

Suggestions

- Add new lines, so that one expression is on one line

Specs

Short name	Structures/OneLineTwoInstructions
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Piwigo*, *Tine20*

13.2.1163 Only Variable Returned By Reference

Function can't return literals by reference.

When a function returns a reference, it is only possible to return variables, properties or static properties.

Anything else, like literals or static expressions, yield a warning at execution time.

```
<?php

// Can't return a literal number
function &foo() {
    return 3 + rand();
}

// bar must return values that are stored in a
function &bar() {
    $a = 3 + rand();
    return $a;
}

?>
```

Specs

Short name	Structures/OnlyVariableReturnedByReference
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1164 openssl_random_pseudo_byte() Second Argument

openssl_random_pseudo_byte() uses exceptions to signal an error. Since PHP 7.4, there is no need to use the second argument.

On the other hand, it is important to catch the exception that openssl_random_pseudo_byte() may emit.

```
<?php
    // PHP 7.4 way to check on random number generation
    try {
        $bytes = openssl_random_pseudo_bytes($i);
    } catch(\Exception $e) {
        die(Error while loading random number);
    }

    // Old way to check on random number generation
    $bytes = openssl_random_pseudo_bytes($i, $cstrong);
    if ($cstrong === false) {
        die(Error while loading random number);
    }
?>
```

See also openssl_random_pseudo_byte and PHP RFC: Improve ‘openssl_random_pseudo_bytes() <https://wiki.php.net/rfc/improve-openssl-random-pseudo-bytes>‘_.

Suggestions

- Skip the second argument, add a try/catch around the call to openssl_random_pseudo_bytes()

Specs

Short name	Structures/OpensslRandomPseudoByteSecondArg
Rulesets	*CE*, *CompatibilityPHP74*
Exakt since	1.9.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1165 Or Die

Classic old style failed error management.

```
<?php

// In case the connexion fails, this kills the current script
mysql_connect('localhost', $user, $pass) or die();

?>
```

Interrupting a script will leave the application with a blank page, will make your life miserable for testing. Just don’t do that.

See also pg_last_error or PDO::exec.

Suggestions

- Throw an exception
- Trigger an error with trigger_error()
- Use your own error mechanism

Specs

Short name	Structures/OrDie
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-implied-if
Examples	*Tine20*, *OpenConf*

13.2.1166 PHP7 Dirname

With PHP 7, dirname() has a second argument that represents the number of parent folder to follow. This prevent us from using nested dirname() calls to reach an grand-parent direct.

```
<?php
$path = '/a/b/c/d/e/f';

// PHP 7 syntax
$threeFoldersUp = dirname($path, 3);

// PHP 5 syntax
$threeFoldersUp = dirname(dirname(dirname($path)));

?>
```

See also dirname.

Suggestions

- Use dirname()'s second argument

Specs

Short name	Structures/PHP7Dirname
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *Suggestions*, *php-cs-fixable*
Exakt since	0.8.4
Php Version	7.0+
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*OpenConf*, *MediaWiki*

13.2.1167 Phpinfo

phpinfo() is a great function to learn about the current configuration of the server.

```
<?php

if (DEBUG) {
    phpinfo();
}

?>
```

If left in the production code, it may lead to a critical leak, as any attacker gaining access to this data will know a lot about the server configuration.

It is advised to never leave that kind of instruction in a production code.

phpinfo() may be necessary to access some specific configuration of the server : for example, `Apache` module list are only available via phpinfo(), and apache_get(), when they are loaded.

Suggestions

- Remove all usage of phpinfo()
- Add one or more constant to fine-tune the phpinfo(), and limit the amount of displayed information
- Replace phpinfo() with a more adapted method : get_loaded_extensions() to access the list of loaded extensions

Specs

Short name	Structures/PhpinfoUsage
Rulesets	*Security*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Dolphin*

13.2.1168 No Plus One

Incrementing a variable should be done with the ++ or – operators. Any other way, may be avoided.

```
<?php

// Best way to increment
++$x; --$y;

// Second best way to increment, if the current value is needed :
echo $x++, $y--;

// Good but slow
$x += 1;
$x -= -1;

$y += -1;
$y -= 1;

// even slower
$x = $x + 1;
$y = $y - 1;

?>
```

This is a micro optimisation.

Specs

Short name	Structures/PlusEgalOne
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1169 Possible Increment

This expression looks like a typo : a missing + would change the behavior.

The same pattern is not reported with -, as it is legit expression. + sign is usually understated, rather than explicit.

```
<?php

// could it be a ++$b ?
$a = +$b;

?>
```

See also Incrementing/Decrementing Operators and Arithmetic Operators.

Suggestions

- Drop the whole assignation
- Complete the addition with another value : $a = 1 + $b
- Make this a ++ operator : ++$b
- Make this a negative operator : -$b
- Make the casting explicit : (int) $b

Specs

Short name	Structures/PossibleIncrement
Rulesets	*Suggestions*
Exakt since	1.2.1
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Zurmo*, *MediaWiki*

13.2.1170 Possible Infinite Loop

Loops on files that can't be open results in infinite loop.

fgets(), and functions like fgetss(), fgetcsv(), fread(), return false when they finish reading, or can't access the file.

In case the file is not accessible, comparing the result of the reading to something that is falsy, leads to a permanent valid condition. The execution will only finish when the `max_execution_time` is reached.

```
<?php

$file = fopen('/path/to/file.txt', 'r');
// when fopen() fails, the next loops is infinite
// fgets() will always return false, and while will always be true.
while($line = fgets($file) != 'a') {
    doSomething();
}

?>
```

It is recommended to check the file resources when they are opened, and always use === or !== to compare readings. feof() is also a reliable function here.

Specs

Short name	Structures/PossibleInfiniteLoop
Rulesets	*Analyze*
Exakt since	1.1.5
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.1171 preg_replace With Option e

> preg_replace() supported the /e option until PHP 7.0. It allowed the use of eval()'ed expression as replacement. This has been dropped in PHP 7.0, for security reasons.

preg_replace() with /e option may be replaced with preg_replace_callback() and a closure, or preg_replace_callback_array() and an array of closures.

```
<?php

// preg_replace with /e
$string = 'abcde';

// PHP 5.6 and older usage of /e
$replaced = preg_replace('/c/e', 'strtoupper(\$0)', $string);

// PHP 7.0 and more recent
// With one replacement
$replaced = preg_replace_callback('/c/', function ($x) { return strtoupper($x[0]); },
↪$string);

// With several replacements, preventing multiple calls to preg_replace_callback
$replaced = preg_replace_callback_array(array('/c/' => function ($x) { return␣
↪strtoupper($x[0]); },
                                              '/[a-b]/' => function ($x) { return␣
↪strtolower($x[0]); }), $string);
?>
```

Suggestions

- Replace call to preg_replace() and /e with preg_replace_callback() or preg_replace_callback_array()

Specs

Short name	Structures/pregOptionE
Rulesets	*Analyze*, *CI-checks*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *Security*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Edusoho*

13.2.1172 Print And Die

Die() <https://www.php.net/'die>'_ also prints.

When stopping a script with die() <https://www.php.net/'die>'_, it is possible to provide a message as first argument, that will be displayed at execution. There is no need to make a specific call to print or echo.

```
<?php

//  die may do both print and die.
echo 'Error message';
die();

//  exit may do both print and die.
print 'Error message';
exit;

//  exit cannot print integers only : they will be used as status report to the␣
↪system.
print 'Error message';
exit 1;

?>
```

Specs

Short name	Structures/PrintAndDie
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1173 Printf Number Of Arguments

The number of arguments provided to printf() or vprintf() doesn't match the format string.

Extra arguments are ignored, and are dead code as such. Missing arguments are reported with a warning, and nothing is displayed.

Omitted arguments produce an error.

```
<?php

// not enough
printf(' a %s ', $a1);
// OK
printf(' a %s ', $a1, $a2);
// too many
printf(' a %s ', $a1, $a2, $a3);

// not enough
sprintf(' a %s ', $a1);
// OK
\sprintf(' a %s ', $a1, $a2);
// too many
sprintf(' a %s ', $a1, $a2, $a3);

?>
```

See also printf and sprintf.

Suggestions

- Sync the number of argument with the format command

Specs

Short name	Structures/PrintfArguments
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.0.1
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	Medium
Examples	*PhpIPAM*

13.2.1174 Avoid Parenthesis

Avoid Parenthesis for language construct. Languages constructs are a few PHP native elements, that looks like functions but are not.

Among other distinction, those elements cannot be directly used as variable function call, and they may be used with or without parenthesis.

```
<?php

// normal usage of include
include 'file.php';

// This looks like a function and is not
include('file2.php');
```

(continues on next page)

(continued from previous page)

```
?>
```

The usage of parenthesis actually give some feeling of comfort, it won't prevent PHP from combining those argument with any later operators, leading to unexpected results.

Even if most of the time, usage of parenthesis is legit, it is recommended to avoid them.

Specs

Short name	Structures/PrintWithoutParenthesis
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1175 Property Variable Confusion

Within a class, there is both a property and variables bearing the same name.

```
<?php
class Object {
    private $x;

    function SetData( ) {
        $this->x = $x + 2;
    }
}
?>
```

The property and the variable may easily be confused one for another and lead to a bug.

Sometimes, when the property is going to be replaced by the incoming argument, or data based on that argument, this naming schema is made on purpose, indicating that the current argument will eventually end up in the property. When the argument has the same name as the property, no warning is reported.

Suggestions

- Use different names for the properties and variables
- Adopt and apply a naming convention for variables and properties.

Specs

Short name	Structures/PropertyVariableConfusion
Rulesets	*Semantics*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*PhpIPAM*

13.2.1176 Queries In Loops

Avoid querying databases in a loop.

Querying an external database in a loop usually leads to performances problems. This is also called the 'n + 1 problem'.

This problem applies also to prepared statement : when such statement are called in a loop, they are slower than one-time large queries.

It is recommended to reduce the number of queries by making one query, and dispatching the results afterwards. This is true with SQL databases, graph queries, LDAP queries, etc.

```
<?php

// Typical N = 1 problem : there will be as many queries as there are elements in
↪$array
$ids = array(1,2,3,5,6,10);

$db = new SQLite3('mysqlitedb.db');

// all the IDS are merged into the query at once
$results = $db->query('SELECT bar FROM foo WHERE id  in ('.implode(',', $id).')');
while ($row = $results->fetchArray()) {
    var_dump($row);
}


// Typical N = 1 problem : there will be as many queries as there are elements in
↪$array
$ids = array(1,2,3,5,6,10);

$db = new SQLite3('mysqlitedb.db');

foreach($ids as $id) {
    $results = $db->query('SELECT bar FROM foo WHERE id = '.$id);
    while ($row = $results->fetchArray()) {
        var_dump($row);
    }
}

?>
```

This optimisation is not always possible : for example, some SQL queries may not be prepared, like `DROP TABLE` or `DESC`. `UPDATE` commands often update one row at a time, and grouping such queries may be counter-productive or unsafe.

Suggestions

- Batch calls by using WHERE clauses and applying the same operation to all similar data
- Use native commands to avoid double query : REPLACE instead of SELECT-(UPDATE/INSERT), or UPSERT, for example

Specs

Short name	Structures/QueriesInLoop
Rulesets	*Analyze*, *Top10*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
Examples	*TeamPass*, *OpenEMR*

13.2.1177 Random Without Try

random_int() and random_bytes() require a try/catch structure around them.

random_int() and random_bytes() emit Exceptions if they meet a problem. This way, failure can't be mistaken with returning an empty value, which leads to lower security.

```
<?php

try {
    $salt = random_bytes($length);
} catch (TypeError $e) {
    // Error while reading the provided parameter
} catch (Exception $e) {
    // Insufficient random data generated
} catch (Error $e) {
    // Error with the provided parameter : <= 0
}

?>
```

Since PHP 7.4, openssl_random_pseudo_bytes() has adopted the same behavior. It is included in this analysis : check your PHP version for actual application.

Suggestions

- Add a try/catch structure around calls to random_int() and random_bytes().

Specs

Short name	Structures/RandomWithoutTry
Rulesets	*Security*
Exakt since	0.8.4
Php Version	7.0+
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.1178 Regex Delimiter

PCRE regular expressions may use a variety of delimiters.

There seems to be a standard delimiter in the code, and some exceptions : one or several forms are dominant (> 90%), while the others are rare.

The analyzed code has less than 10% of the rare delimiters. For consistency reasons, it is recommended to make them all the same.

Generally, one or two delimiters are used, depending on the expected special chars in the scanned strings : for example, / tends to be avoided when parsing HTML.

Regex are literals, or partial literals, used in preg_match(), preg_match_all(), preg_replace(), preg_replace_callback(), preg_replace_callback_array().

```
<?php

echo 'a';
echo 'b';
echo 'c';
echo 'd';
echo 'e';
echo 'f';
echo 'g';
echo 'h';
echo 'i';
echo 'j';
echo 'k';

// This should probably be written 'echo';
print 'l';

?>
```

See also Ideal regex delimiters in PHP.

Specs

Short name	Structures/RegexDelimiter
Rulesets	none
Exakt since	0.10.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1179 Repeated print()

Always merge several print or echo in one call.

It is recommended to use echo with multiple arguments, or a concatenation with print, instead of multiple calls to print echo, when outputting several blob of text.

```
<?php

//Write :
  echo 'a', $b, 'c';
  print 'a' . $b . 'c';

//Don't write :
  print 'a';
  print $b;
  print 'c';
?>
```

Suggestions

- Merge all prints into one echo call, separating arguments by commas.
- Collect all values in one variable, and do only one call to print or echo.

Specs

Short name	Structures/RepeatedPrint
Rulesets	*Analyze*, *CI-checks*, *Suggestions*, *Top10*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-repeated-print
Examples	*Edusoho*, *HuMo-Gen*

13.2.1180 Repeated Regex

Repeated regex should be centralized.

When a regex is repeatedly used in the code, it is getting harder to update.

```
<?php

// Regex used several times, at least twice.
preg_match('/^abc_|^square$/i', $_GET['x']);

//.......

preg_match('/^abc_|^square$/i', $row['name']);

// This regex is dynamically built, so it is not reported.
preg_match('/^circle|^'.$x.'$/i', $string);

// This regex is used once, so it is not reported.
preg_match('/^circle|^square$/i', $string);

?>
```

Regex that are repeated at least once (aka, used twice or more) are reported. Regex that are dynamically build are not reported.

Suggestions

- Create a central library of regex
- Use the regex inventory to spot other regex that are close, and should be identical.

Specs

Short name	Structures/RepeatedRegex
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.10.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Vanilla*, *Tikiwiki*

13.2.1181 Resources Usage

List of situations that are creating resources.

```
<?php
    // This functioncall creates a resource to use
    $fp = fopen('/tmp/file.txt', 'r');

    if (!is_resource($fp)){
        thrown new RuntimeException('Could not open file.txt');
    }
?>
```

Specs

Short name	Structures/ResourcesUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1182 Results May Be Missing

preg_match() may return empty values, if the search fails. It is important to check for the existence of results before assigning them to another variable, or using it.

```
<?php
    preg_match('/PHP ([0-9\.]+) /', $res, $r);
    $s = $r[1];
    // $s may end up null if preg_match fails.
?>
```

Specs

Short name	Structures/ResultMayBeMissing
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1183 Return True False

These conditional expressions return true/false, depending on the condition. This may be simplified by dropping the control structure altogether.

```
<?php

if (version_compare($a, $b) >= 0) {
    return true;
} else {
    return false;
}

?>
```

This may be simplified with :

```
<?php

return version_compare($a, $b) >= 0;

?>
```

This may be applied to assignations and ternary operators too.

```
<?php

if (version_compare($a, $b) >= 0) {
    $a = true;
} else {
    $a = false;
}

$a = version_compare($a, $b) >= 0 ? false : true;

?>
```

Suggestions

- Return directly the comparison, without using the if/then structure
- Cast the value to (boolean) and use it instead of the ternary

Specs

Short name	Structures/ReturnTrueFalse
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Mautic*, *FuelCMS*

13.2.1184 Return void

Return returns null as default value. It is recommended to mention explicitly 'null' or find a meaningful return such as a boolean or a default value instead.

```
<?php

function foo(&$a) {
    ++$a;
    // No explicit return : it returns void
}

function bar(&$a) {
    ++$a;
```

(continues on next page)

(continued from previous page)

```
    // Explicit return : it returns null
    return null
}

?>
```

See also Void functions.

Specs

Short name	Structures/ReturnVoid
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1185 Reuse Variable

A variable is already holding the content that is calculated multiple times over.

It is recommended to use the cached value. This saves some computation, in particular when used in a loop, and speeds up the process.

```
<?php

function foo($a) {
    $b = strtolower($a);

    // strtolower($a) is already calculated in $b. Just reuse the value.
    if (strtolower($a) === 'c') {
        doSomething();
    }
}

?>
```

Suggestions

- Reuse the already created variable

Specs

Short name	Structures/ReuseVariable
Rulesets	*Suggestions*
Exakt since	1.1.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Medium

13.2.1186 Same Conditions In Condition

At least two consecutive if/then structures use identical conditions. The latter will probably be ignored.

This analysis returns false positive when there are attempt to fix a situation, or to call an alternative solution.

Conditions that are shared between if structures, but inside a logical OR expression are also detected.

```
<?php

if ($a == 1) { doSomething(); }
elseif ($b == 1) { doSomething(); }
elseif ($c == 1) { doSomething(); }
elseif ($a == 1) { doSomething(); }
else {}

// Also works on if then else if chains
if ($a == 1) { doSomething(); }
else if ($b == 1) { doSomething(); }
else if ($c == 1) { doSomething(); }
else if ($a == 1) { doSomething(); }
else {}

// Also works on if then else if chains
// Here, $a is common and sufficient in both conditions
if ($a || $b) { doSomething(); }
elseif ($a || $c) { doSomethingElse(); }

// This sort of situation generate false postive.
$config = load_config_from_commandline();
if (empty($config)) {
    $config = load_config_from_file();
    if (empty($config)) {
        $config = load_default_config();
    }
}

?>
```

Suggestions

- Merge the two conditions into one
- Make the two conditions different

Specs

Short name	Structures/SameConditions
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High
Examples	*TeamPass*, *Typo3*

13.2.1187 Sequences In For

For() instructions allows several instructions in each of its parameters. Then, the instruction separator is comma ‘,’, not semi-colon, which is used for separating the 3 arguments.

```
<?php
   for ($a = 0, $b = 0; $a < 10, $b < 20; $a++, $b += 3) {
    // For loop
   }
?>
```

This loop will simultaneously increment $a and $b. It will stop only when the last of the central sequence reach a value of false : here, when $b reach 20 and $a will be 6.

This structure is often unknown, and makes the for instruction quite difficult to read. It is also easy to oversee the multiples instructions, and omit one of them. It is recommended not to use it.

Specs

Short name	Structures/SequenceInFor
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1188 Set Aside Code

Setting aside code should be made into a method.

Setting aside code happens when one variable or member is stored locally, to be temporarily replaced by another value. Once the new value has been processed, the original value is reverted.

The temporary change of the value makes the code hard to read.

It is a good example of a piece of code that could be moved to a separate method or function. Using the temporary value as a parameter makes the change visible, and avoid local pollution.

```php
<?php

// Setting aside database
class cache extends Storage {
    private $database = null;

    function __construct($database) {
        $this->database = $database;
    }

    function foo($values) {
        // handling storage with sqlite3
        $secondary = new cache(new Sqlite3(':memory:'));
        $secondary->store($values);

        $this->store($values);      // handling storage with injection
    }
}

// Setting aside database to cache data in two distinct backend
class cache extends Storage {
    private $database = null;

    function __construct(\Pdo $database) {
        $this->database = $database;
    }

    function foo($values) {
        // $this->database is set aside for secondary configuration
        $side = $this->database;
        $this->database = new Sqlite3(':memory:');
        $this->store($values);      // handling storage with sqlite3
        $this->database = $side;
        // $this->database is restored
        $this->store($values);      // handling storage with injection
    }
}

?>
```

Suggestions

- Extract the code that run with the temporary value to a separate method.

Specs

Short name	Structures/SetAside
Rulesets	*Suggestions*
Exakt since	1.8.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1189 Setlocale() Uses Constants

setlocale() don't use strings but constants.

The first argument of setlocale() must be one of the valid constants, `LC_ALL`, `LC_COLLATE`, `LC_CTYPE`, `LC_MONETARY`, `LC_NUMERIC`, `LC_TIME`, `LC_MESSAGES <https://www.php.net/LC_MESSAGES>`_.

```
<?php

// Use constantes for setlocale first argument
setlocale(LC_ALL, 'nl_NL');
setlocale(\LC_ALL, 'nl_NL');

// Don't use string for setlocale first argument
setlocale('LC_ALL', 'nl_NL');
setlocale('LC_'.'ALL', 'nl_NL');

?>
```

The PHP 5 usage of strings (same name as above, enclosed in ' or ") is not legit anymore in PHP 7 and later.

See also setlocale.

Suggestions

- Use setlocale() constants

Specs

Short name	Structures/SetlocaleNeedsConstants
Rulesets	*CompatibilityPHP70*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.1190 Static Global Variables Confusion

PHP can't have variable that are both static and global variable. While the syntax is legit, the variables will be alternatively global or static.

It is recommended to avoid using the same name for a global variable and a static variable.

```
<?php

function foo() {
    $a = 1; // $a is a local variable

    global $a; // $a is now a global variable

    static $a; // $a is not w static variable
```

(continues on next page)

(continued from previous page)

```
}

?>
```

Suggestions

- Avoid using static variables
- Avoid using global variables
- Avoid using the same name for static and global variables

Specs

Short name	Structures/SGVariablesConfusion
Rulesets	*Suggestions*
Exakt since	2.1.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1191 Shell Usage

List of shell calls to system.

```
<?php
    // Using backtick operator
    $a = `ls -hla`;

    // Using one of PHP native or extension functions
    $a = shell_exec('ls -hla');
    $b = \pcntl_exec('/path/to/command');

?>
```

See also shell_exec and Execution Operators.

Specs

Short name	Structures/ShellUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1192 Using Short Tags

The code makes use of short tags. Short tags are the following : `<?` . A full scripts looks like that : `<? /* php code */ ?>`.

It is recommended to not use short tags, and use standard PHP tags. This makes PHP code compatible with XML standards. Short tags used to be popular, but have lost it.

See also PHP Tags.

Specs

Short name	Structures/ShortTags
Rulesets	*CE*
Exakt since	0.8.4
Php Version	7.0-
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-short-tags

13.2.1193 Should Chain Exception

Chain exception to provide more context.

When catching an exception and rethrowing another one, it is recommended to chain the exception : this means providing the original exception, so that the final recipient has a chance to track the origin of the problem. This doesn't change the thrown message, but provides more information.

Note : Chaining requires PHP > 5.3.0.

```
<?php
    try {
        throw new Exception('Exception 1', 1);
    } catch (\Exception $e) {
        throw new Exception('Exception 2', 2, $e);
        // Chaining here.

    }
?>
```

See also Exception::‘__construct <https://www.php.net/manual/en/exception.construct.php>‘_ and What are the best practices for catching and re-throwing exceptions?.

Suggestions

- Add the incoming exception to the newly thrown exception

Specs

Short name	Structures/ShouldChainException
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Magento*, *Tine20*

13.2.1194 Should Make Ternary

Ternary operators are the best when assigning values to a variable.

This way, they are less verbose, compatible with assignation and easier to read.

```
<?php
    // verbose if then structure
    if ($a == 3) {
        $b = 2;
    } else {
        $b = 3;
    }

    // compact ternary call
    $b = ($a == 3) ? 2 : 3;

    // verbose if then structure
    // Works with short assignations and simple expressions
    if ($a != 3) {
        $b += 2 - $a * 4;
    } else {
        $b += 3;
    }

    // compact ternary call
    $b += ($a != 3) ? 2 - $a * 4 : 3;

?>
```

See also Ternary Operator and Shorthand comparisons in PHP.

Specs

Short name	Structures/ShouldMakeTernary
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.5
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*ChurchCRM*

13.2.1195 Preprocessable

The following expression are made of literals or already known values : they may be fully calculated before running PHP.

```
<?php

// Building an array from a string
$name = 'PHP'.' '.'7.2';

// Building an array from a string
$list = explode(',', 'a,b,c,d,e,f');

// Calculating a power
$kbytes = $bytes / pow(2, 10);

// This will never change
$name = ucfirst(strtolower('PARIS'));

?>
```

By doing so, this will reduce the amount of work of PHP.

Suggestions

- Do the work yourself, instead of giving it to PHP

Specs

Short name	Structures/ShouldPreprocess
Rulesets	*Analyze*, *Rector*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*phpadsnew*

13.2.1196 Should Use Explode Args

explode() has a third argument, which limits the amount of exploded elements. With it, it is possible to collect only the first elements, or drop the last ones.

```
<?php

$exploded = explode(DELIMITER, $string);

// use explode(DELIMITER, $string, -1);
array_pop($exploded);

// use explode(DELIMITER, $string, -2);
$c = array_slice($exploded, 0, -2);
```

(continues on next page)

(continued from previous page)

```
// with explode()'s third argument :
list($a, $b) = explode(DELIMITER, $string, 2);

// with list() omitted arguments
list($a, $b, ) = explode(DELIMITER, $string);

?>
```

See also explode.

Suggestions

-

Specs

Short name	Structures/ShouldUseExplodeArgs
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.9.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1197 Should Use Foreach

Use foreach instead of for when traversing an array.

Foreach() is the modern loop : it maps automatically every element of the array to a blind variable, and loop over it. This is faster and safer.

```
<?php

// Foreach version
foreach($array as $element) {
    doSomething($element);
}

// The above case may even be upgraded with array_map and a callback,
// for the simplest one of them
$array = array_map('doSomething', $array);

// For version (one of various alternatives)
for($i = 0; $i < count($array); $i++) {
    $element = $array[$i];
    doSomething($element);
}

// Based on array_pop or array_shift()
while($value = array_pop($array)) {
```

(continues on next page)

(continued from previous page)

```
    doSomething($array);
}

?>
```

See also foreach and 5 Ways To Loop Through An Array In PHP.

Suggestions

- Move for() loops to foreach(), whenever they apply to a finite list of elements

Specs

Short name	Structures/ShouldUseForeach
Rulesets	*Suggestions*
Exakt since	0.12.7
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*ExpressionEngine*, *Woocommerce*

13.2.1198 Should Use Math

Use math operators to make the operation readable.

```
<?php

// Adding one to self
$a *= 2;
// same as above
$a += $a;

// Squaring oneself
$a \*\*\= 2;
// same as above
$a *= $a;

// Removing oneself
$a = 0;
// same as above
$a -= $a;

// Dividing oneself
$a = 1;
// same as above
$a /= $a;

// Divisition remainer
$a = 0;
// same as above
```

(continues on next page)

(continued from previous page)

```
$a %= $a;

?>
```

See also Mathematical Functions.

Suggestions

- Use explicit math assignation

Specs

Short name	Structures/ShouldUseMath
Rulesets	*Suggestions*
Exakt since	1.1.5
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*OpenEMR*

13.2.1199 Should Use Operator

Some functions duplicate the feature of an operator. When in doubt, it is better to use the operator.

Beware, some edge cases may apply. In particular, backward compatibility may prevent usage of newer features.

- array_push() is equivalent to []
- is_object() is equivalent to instanceof
- function_get_arg() and function_get_args() is equivalent to ellipsis : ...
- chr() is equivalent to string escape sequences, such as `\n`, `\x69`, `u{04699}`
- call_user_func() is equivalent to `$functionName(arguments)`, `$object->$method(`... <https://www.php.net/manual/en/functions.arguments.php#functions.variable-arg-list>`_$arguments)`
- is_null() is equivalent to `=== null`
- php_version() is equivalent to `PHP_VERSION` (the constant)
- is_array(), is_int(), is_object(), etc. is equivalent to a scalar typehint

Suggestions

- Use [] instead of array_push()
- Use instanceof instead of is_object()
- Use ... instead of function_get_arg() and function_get_args()
- Use escape sequences instead of chr()

- Use dynamic function call instead of call_user_func()
- Use === null instead of is_null()
- Use PHP_VERSION instead of php_version()
- Use typehint instead of is_int(), is_string(), is_bool(), etc.

Specs

Short name	Structures/ShouldUseOperator
Rulesets	*Suggestions*
Exakt since	1.3.0
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Zencart*, *SugarCrm*

13.2.1200 Simplify Regex

Avoid using regex when the searched string or the replacement are simple enough.

PRCE regex are a powerful way to search inside strings, but they also come at the price of performance. When the query is simple enough, try using strpos() or stripos() instead.

```
<?php

// simple preg calls
if (preg_match('/a/', $string))  {}
if (preg_match('/b/i', $string)) {} // case insensitive

// light replacements
if( strpos('a', $string)) {}
if( stripos('b', $string)) {}       // case insensitive

?>
```

Suggestions

- Use str_replace(), strtr() or even strpos()

Specs

Short name	Structures/SimplePreg
Rulesets	*Performances*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Zurmo*, *OpenConf*

13.2.1201 Static Loop

Static loop may be preprocessed.

It looks like the following loops are static : the same code is executed each time, without taking into account loop variables.

```
<?php

// Static loop
$total = 0;
for($i = 0; $i < 10; $i++) {
    $total += $i;
}

// The above loop may be replaced by (with some math help)
$total = 10 * (10  + 1) / 2;

// Non-Static loop (the loop depends on the size of the array)
$n = count($array);
for($i = 0; $i < $n; $i++) {
    $total += $i;
}

?>
```

It is possible to create loops that don't use any blind variables, though this is fairly rare. In particular, calling a method may update an internal pointer, like next() or `SimpleXMLIterator\:\:`next() <https://www.php.net/next>`_.

It is recommended to turn a static loop into an expression that avoid the loop. For example, replacing the sum of all integers by the `function $n * ($n + 1) / 2`, or using array_sum().

This analysis doesn't detect usage of variables with `compact`.

Suggestions

- Precalculate the result of that loop and removes it altogether
- Check that the loop is not missing a blind variable usage
- Replace the usage of a loop with a native PHP call : for example, with str_repeat(). Although the loop is still here, it usually reflects better the intend.

Specs

Short name	Structures/StaticLoop
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1202 strip_tags Skips Closed Tag

strip_tags() skips non-self closing tags. This means that tags such as `<br />` will be ignored from the 2nd argument of the function.

```
<?php

$input = 'a<br />';

// Displays 'a' and clean the tag
echo strip_tags($input, '<br>');

// Displays 'a<br />' and skips the allowed tag
echo strip_tags($input, '<br/>');

?>
```

See also strip_tags.

Suggestions

- Do not use self-closing tags in the 2nd parameter

Specs

Short name	Structures/StripTagsSkipsClosedTag
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.9.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1203 Strpos()-like Comparison

The result of that function may be mistaken with an error.

strpos(), along with several PHP native functions, returns a string position, starting at 0, or false, in case of failure.

```
<?php

// This is the best comparison
if (strpos($string, 'a') === false) { }

// This is OK, as 2 won't be mistaken with false
if (strpos($string, 'a') == 2) { }

// strpos is one of the 26 functions that may behave this way
if (preg_match($regex, $string)) { }

// This works like above, catching the value for later reuse
if ($a = strpos($string, 'a')) { }
```

(continues on next page)

(continued from previous page)

```
// This misses the case where 'a' is the first char of the string
if (strpos($string, 'a')) { }

// This misses the case where 'a' is the first char of the string, just like above
if (strpos($string, 'a') == 0) { }

?>
```

It is recommended to check the result of strpos() with === or !==, so as to avoid confusing 0 and false.

This analyzer list all the strpos()-like functions that are directly compared with == or !=. preg_match(), when its first argument is a literal, is omitted : this function only returns NULL in case of regex error.

The full list is the following :

- array_search()
- collator_compare()
- collator_get_sort_key()
- current()
- fgetc()
- file_get_contents()
- file_put_contents()
- fread()
- iconv_strpos()
- iconv_strrpos()
- imagecolorallocate()
- imagecolorallocatealpha()
- mb_strlen()
- next()
- pcntl_getpriority()
- preg_match()
- prev()
- readdir()
- stripos()
- strpos()
- strripos()
- strrpos()
- strtok()
- curl_exec()

In PHP 8.0, str_contains() will do the expected job of strpos(), with less confusion.

See also strpos not working correctly.

Suggestions

- Use identity comparisons, for 0 values : === instead of ==, etc.
- Compare with other exact values than 0 : strpos() == 2
- Use str_contains()

Specs

Short name	Structures/StrposCompare
Rulesets	*Analyze*, *CI-checks*, *Top10*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	strict-comparisons
Examples	*Piwigo*, *Thelia*

13.2.1204 Drop Substr Last Arg

Substr() works till the end of the string when the last argument is omitted. There is no need to calculate string size to make this work.

```
<?php

$string = 'abcdef';

// Extract the end of the string
$cde = substr($string, 2);

// Too much work
$cde = substr($string, 2, strlen($string));

?>
```

See also substr.

Suggestions

- Use negative length
- Omit the last argument to get the string till its end

Specs

Short name	Structures/SubstrLastArg
Rulesets	*Suggestions*
Exakt since	1.2.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*SuiteCrm*, *Tine20*

13.2.1205 Substr To Trim

When removing the first or the last character of a string, trim() does a more readable job.

trim(), ltrim() and rtrim() accept a string as second argument. Those will all be removed from the endings of the string.

```
<?php

$a = '$drop the dollar';
$b = substr($a, 1); // drop the first char
$b = ltrim($a, '$'); // remove the initial '$'s


$b = substr($a, 1);     // replace with ltrim()

$b = substr($a, 0, -1); // replace with rtrim()

$b = substr($a, 1, -1); // replace with trim()

?>
```

trim() will remove all occurrences of the requested char(). This may remove a loop with substr(), or remove more than is needed.

trim() doesn't work with multi-bytes strings, but so does substr(). For that, use mb_substr(), as there isn't any mb_trim function (yet).

See also trim, ltrim, rtrim.

Suggestions

- Replace substr() with trim(), ltrim() or rtrim().

Specs

Short name	Structures/SubstrToTrim
Rulesets	*Suggestions*
Exakt since	1.8.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1206 Suspicious Comparison

The comparison seems to be misplaced.

A comparison happens in the last argument, while the actual function expect another type : this may be the case of a badly placed parenthesis.

```
<?php

// trim expect a string, a boolean is given.
if (trim($str === '')){

}

// Just move the first closing parenthesis to give back its actual meaning
if (trim($str) === ''){

}

?>
```

Original idea by Vladimir Reznichenko.

Suggestions

- Remove the comparison altogether
- Move the comparison to its right place : that, or more the parenthesis.
- This may be what is intended : just leave it.

Specs

Short name	Structures/SuspiciousComparison
Rulesets	*Analyze*
Exakt since	0.11.0
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
Examples	*PhpIPAM*, *ExpressionEngine*

13.2.1207 Switch To Switch

The following structures are based on if / elseif / else. Since they have more than three conditions (not withstanding the final else), it is recommended to use the switch structure, so as to make this more readable.

On the other hand, switch() structures with less than 3 elements should be expressed as a if / else structure.

Note that if condition that uses strict typing (=== or !==) can't be converted to switch() as the latter only performs == or != comparisons.

```php
<?php

if ($a == 1) {

} elseif ($a == 2) {

} elseif ($a == 3) {

} elseif ($a == 4) {

} else {

}

// Better way to write long if/else lists
switch ($a) {
    case 1 :
        doSomething(1);
        break 1;

    case 2 :
        doSomething(2);
        break 1;

    case 3 :
        doSomething(3);
        break 1;

    case 4 :
        doSomething(4);
        break 1;

    default :
        doSomething();
        break 1;
}

?>
```

Note that simple switch statement, which compare a variable to a literal are optimised in PHP 7.2 and more recent. This gives a nice performance boost, and keep code readable.

See also PHP 7.2's switch optimisations and Is Your Code Readable By Humans? Cognitive Complexity Tells You.

Suggestions

- Use a switch statement, rather than a long string of if/else
- Use a match() statement, rather than a long string of if/else (PHP 8.0 +)

Specs

Short name	Structures/SwitchToSwitch
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Thelia*, *XOOPS*

13.2.1208 Switch With Too Many Default

Switch statements should only hold one default, not more. Check the code and remove the extra default.

PHP 7.0 won't compile a script that allows for several default cases.

Multiple default happens often with large switch().

```
<?php

switch($a) {
    case 1 :
        break;
    default :
        break;
    case 2 :
        break;
    default :  // This default is never reached
        break;
}

?>
```

Suggestions

- Remove the useless default : it may be the first, or the last. In case of ambiguity, keep the first, as it is the one being used at the moment.

Specs

Short name	Structures/SwitchWithMultipleDefault
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	7.0-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1209 Switch Without Default

Always use a default statement in switch().

Switch statements hold a number of 'case' that cover all known situations, and a 'default' one which is executed when all other options are exhausted.

```
<?php

// Missing default
switch($format) {
    case 'gif' :
        processGif();
        break 1;

    case 'jpeg' :
        processJpeg();
        break 1;

    case 'bmp' :
        throw new UnsupportedFormat($format);
}
// In case $format is not known, then switch is ignored and no processing happens,␣
↪leading to preparation errors


// switch with default
switch($format) {
    case 'text' :
        processText();
        break 1;

    case 'jpeg' :
        processJpeg();
        break 1;

    case 'rtf' :
        throw new UnsupportedFormat($format);

    default :
        throw new UnknownFileFormat($format);
}
// In case $format is not known, an exception is thrown for processing

?>
```

Most of the time, switch() do need a default case, so as to catch the odd situation where the 'value is not what it was expected'. This is a good place to catch unexpected values, to set a default behavior.

Suggestions

- Add a default case

Specs

Short name	Structures/SwitchWithoutDefault
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-switch-without-default
Examples	*Zencart*, *Traq*

13.2.1210 Ternary In Concat

Ternary and coalesce operator have higher priority than dot '.' for concatenation. This means that :

```
<?php
  // print B0CE as expected
  print 'B'.$b.'C'. ($b > 1 ? 'D') : 'E';

  // print E, instead of B0CE
  print 'B'.$b.'C'. $b > 1 ? 'D' : 'E';

  print 'B'.$b.'C'. $b > 1 ? 'D' : 'E';
?>
```

prints actually 'E', instead of the awaited 'B0CE'.

To be safe, always add parenthesis when using ternary operator with concatenation.

See also Operator Precedence.

Suggestions

- Use parenthesis
- Avoid ternaries and coalesce operators inside a string

Specs

Short name	Structures/TernaryInConcat
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High
Examples	*TeamPass*

13.2.1211 Test Then Cast

A test is run on the value, but the cast value is later used.

The cast may introduce a distortion to the value, and still lead to the unwanted situation. For example, comparing to 0, then later casting to an int. The comparison to 0 is done without casting, and as such, 0.1 is different from 0. Yet, (int) 0.1 is actually 0, leading to a Division by 0 error.

```
<?php

// Here. $x may be different from 0, but (int) $x may be 0
$x = 0.1;

if ($x != 0) {
    $y = 4 / (int) $x;
}

// Safe solution : check the cast value.
if ( (int) $x != 0) {
    $y = 4 / (int) $x;
}

?>
```

Suggestions

- Test with the cast value

Specs

Short name	Structures/TestThenCast
Rulesets	*Analyze*
Exakt since	1.1.6
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Dolphin*, *SuiteCrm*

13.2.1212 Throws An Assignement

It is possible to throw an exception, and, in the same time, assign this exception to a variable.

However, the variable will never be used, as the exception is thrown, and any following code is not executed, unless the exception is caught in the same scope.

```
<?php

    // $e is useful, though not by much
    $e = new() Exception();
    throw $e;
```

(continues on next page)

(continued from previous page)

```
    // $e is useless
    throw $e = new Exception();

?>
```

Suggestions

- Drop the assignation

Specs

Short name	Structures/ThrowsAndAssign
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1213 Timestamp Difference

`time()` and `microtime()` shouldn't be used to calculate duration.

`time()` and `microtime()` are subject to variations, depending on system clock variations, such as daylight saving time difference (every spring and fall, one hour variation), or leap seconds, happening on `June, 30th` or `December 31th`, as announced by IERS.

```
<?php

// Calculating tomorow, same hour, the wrong way
// tomorrow is not always in 86400s, especially in countries with daylight saving
$tomorrow = time()  + 86400;

// Good way to calculate tomorrow
$datetime = new DateTime('tomorrow');

?>
```

When the difference may be rounded to a larger time unit (rounding the difference to days, or several hours), the variation may be ignored safely.

When the difference is very small, it requires a better way to measure time difference, such as *Ticks <https://www.php.net/manual/en/control-structures.declare.php#control-structures.declare.ticks>'_*, *'ext/hrtime <https://www.php.net/manual/en/book.hrtime.php>'_, or including a check on the actual time zone (''ini_get()'* with 'date.timezone').

See also PHP 'DateTime <https://www.php.net/'DateTime>'_ difference – it's a trap! <http://blog.codebusters.pl/en/php-datetime-difference-trap/>'_ and PHP Daylight savings bug?.

Suggestions

- For small time intervals, use hrtime() functions
- For larger time intervals, use add() method with `DateTime`

Specs

Short name	Structures/TimestampDifference
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Zurmo*, *shopware*

13.2.1214 __toString() Throws Exception

Magical method __toString() can't throw exceptions.

In fact, __toString() may not let an exception pass. If it throw an exception, but must catch it. If an underlying method throws an exception, it must be caught.

```
<?php

class myString {
    private $string = null;

    public function __construct($string) {
        $this->string = $string;
    }

    public function __toString() {
        // Do not throw exceptions in __toString
        if (!is_string($this->string)) {
            throw new Exception("$this->string is not a string!!");
        }

        return $this->string;
    }
}

?>
```

A fatal error is displayed, when an exception is not intercepted in the __toString() function.

::

PHP Fatal error: Method myString::__toString() must not throw an exception, caught Exception: 'Exception message' in `file.php`

See also __toString().

Suggestions

- Remove any usage of exception from __toString() magic method

Specs

Short name	Structures/toStringThrowsException
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	7.4-
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1215 Try With Finally

Indicates if a try use a finally statement.

```
<?php

try {
    $a = doSomething();
} catch (Throwable $e) {
    // Fix the problem
} finally {
    // remove $a anyway
    unset($a);
}

?>
```

See also Exceptions, to learn about catching an exception.

Specs

Short name	Structures/TryFinally
Rulesets	*CE*
Exakt since	0.8.4
Php Version	5.5+
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.1216 Unchecked Resources

Resources are created, but never checked before being used. This is not safe.

Always check that resources are correctly created before using them.

```php
<?php

// always check that the resource is created correctly
$fp = fopen($d,'r');
if ($fp === false) {
    throw new Exception('File not found');
}
$firstLine = fread($fp);

// This directory is not checked : the path may not exist and return false
$uncheckedDir = opendir($pathToDir);
while(readdir($uncheckedDir)) {
    // do something()
}

// This file is not checked : the path may not exist or be unreadable and return false
$fp = fopen($pathToFile);
while($line = freads($fp)) {
    $text .= $line;
}

// unsafe one-liner : using bzclose on an unchecked resource
bzclose(bzopen('file'));

?>
```

See also resources.

Suggestions

- Add a check between the resource acquisition and its usage

Specs

Short name	Structures/UncheckedResources
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High
ClearPHP	no-unchecked-resources

13.2.1217 Unconditional Break In Loop

An unconditional break in a loop creates dead code. Since the break is directly in the body of the loop, it is always executed, creating a strange loop that can only run once.

Here, break may also be a return, a goto or a continue. They all branch out of the loop. Such statement are valid, but should be moderated with a condition.

```php
<?php

// return in loop should be in
function summAll($array) {
    $sum = 0;

    foreach($array as $a) {
        // Stop at the first error
        if (is_string($a)) {
            return $sum;
        }
        $sum += $a;
    }

    return $sum;
}

// foreach loop used to collect first element in array
function getFirst($array) {
    foreach($array as $a) {
        return $a;
    }
}

?>
```

Suggestions

- Remove the loop and call the content of the loop once.

Specs

Short name	Structures/UnconditionLoopBreak
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.12.16
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*LiveZilla*, *MediaWiki*

13.2.1218 Unkown Regex Options

Regex support in PHP accepts the following list of options : `eimsuxADJSUX`.

All other letter used as option are not supported : depending on the situation, they may be ignored or raise an error.

```php
<?php

// all options are available
if (preg_match('/\d+/isA', $string, $results)) { }
```

(continues on next page)

(continued from previous page)

```
// p and h are not regex options, p is double
if (preg_match('/\d+/php', $string, $results)) { }

?>
```

See also Pattern Modifiers

Suggestions

- Remove the unknown options
- Replace the option with a valid one
- Fix any syntax typo in the regex

Specs

Short name	Structures/UnknownPregOption
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.1219 Unpreprocessed Values

Preprocessing values is the preparation of values before PHP executes the code.

There is no macro language in PHP, that prepares the code before compilation, bringing some comfort and short syntax. Most of the time, one uses PHP itself to preprocess data.

For example :

```
<?php
    $days_en = 'monday,tuesday,wednesday,thursday,friday,saturday,sunday';
    $days_zh = ',,,,,,';

    $days = explode(',', $lang === 'en' ? $days_en : $days_zh);
?>
```

could be written

```
<?php
    if ($lang === 'en') {
        $days = ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday',
↪'sunday'];
    } else {
        $days = ['', '', '', '', '', '', ''];
    }
?>
```

and avoid preprocessing the string into an array first.

Preprocessing could be done anytime the script includes all the needed values to process the expression.

Suggestions

- Preprocess the values and hardcode them in PHP. Do not use PHP to calculate something at the last moment.
- Use already processed values, or cache to avoid calculating the value each hit.
- Create a class that export the data in the right format for every situation, including the developer's comfort.

Specs

Short name	Structures/Unpreprocessed
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	always-preprocess
Examples	*Zurmo*, *Piwigo*

13.2.1220 Unreachable Code

Code may be unreachable, because other instructions prevent its reaching.

For example, it be located after throw, return, exit() <https://www.php.net/‘exit>‘_, die() <https://www.php.net/‘die>‘_, goto, break or continue : this way, it cannot be reached, as the previous instruction will divert the engine to another part of the code.

```
<?php

function foo() {
    $a++;
    return $a;
    $b++;       // $b++ can't be reached;
}

function bar() {
    if ($a) {
        return $a;
    } else {
        return $b;
    }
    $b++;       // $b++ can't be reached;
}

foreach($a as $b) {
    $c += $b;
    if ($c > 10) {
        continue 1;
    } else {
```

(continues on next page)

(continued from previous page)

```
        $c--;
        continue;
    }
    $d += $e;    // this can't be reached
}

$a = 1;
goto B;
class foo {}    // Definitions are accessible, but not functioncalls
B:
echo $a;

?>
```

This is dead code, that may be removed.

Suggestions

- Remove the unreachable code
- Remove the blocking expression, and let the code execute

Specs

Short name	Structures/UnreachableCode
Rulesets	*Dead code*, *Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-dead-code

13.2.1221 Unset In Foreach

Unset applied to the variables of a `foreach` loop are useless. Those variables are copies and not the actual value. Even if the value is a reference, unsetting it has no effect on the original array : the only effect may be indirect, on elements inside an array, or on properties inside an object.

```
<?php

// When unset is useless
$array = [1, 2, 3];
foreach($array as $a) {
    unset($a);
}

print_r($array); // still [1, 2, 3]

foreach($array as $b => &$a) {
    unset($a);
```

(continues on next page)

(continued from previous page)

```
}

print_r($array); // still [1, 2, 3]

// When unset is useful
$array = [ [ 'c' => 1] ]; // Array in array
foreach($array as &$a) {
    unset(&$a['c']);
}

print_r($array); // now [ ['c' => null] ]

?>
```

See also foreach.

Suggestions

- Drop the unset

Specs

Short name	Structures/UnsetInForeach
Rulesets	*Analyze*, *Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1222 Unsupported Types With Operators

Arrays, resources and objects are generally not accepted with unary and binary operators.

The operators are +, -, *, /, **, %, <<, >>, &, |, ^, ~, ++ and –.

```
<?php

var_dump([] % [42]);
// int(0) in PHP 7.x
// TypeError in PHP 8.0 +

// Also impossible usage : index are string or int
$a = [];
$b = $c[$a];

?>
```

In PHP 8.0, the rules have been made stricter and more consistent.

The only valid operator is +, combined with arrays in both operands. Other situation will throw *TypeError*.

See also Stricter type checks for arithmetic/bitwise operators and TypeError.

Suggestions

- Do not use those values with those operators
- Use a condition to skip this awkward situation
- Add an extra step to turn this value into a valid type

Specs

Short name	Structures/UnsupportedTypesWithOperators
Rulesets	*Analyze*, *CE*, *CompatibilityPHP80*
Exakt since	2.1.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.1223 Unused Global

A global keyword is used in a method, yet the variable is not actually used. This makes PHP import values for nothing, or may create interference

```
<?php
    function foo() {
        global bar;

        return 1;
    }
?>
```

Suggestions

- Remove the global declaration
- Remove the global variable altogether

Specs

Short name	Structures/UnusedGlobal
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Dolphin*

13.2.1224 Unused Label

Some labels have been defined in the code, but they are not used. They may be removed as they are dead code.

```
<?php

$a = 0;
A:

    ++$a;

    // A loop. A: is used
    if ($a < 10) { goto A; }

// B is never called explicitely. This is useless.
B:

?>
```

There is no analysis for undefined goto call, as PHP checks that goto has a destination label at compile time :

See also Goto.

Suggestions

- Remove the unused label
- Add a goto call to this label
- Check for spelling mistakes

Specs

Short name	Structures/UnusedLabel
Rulesets	*Dead code*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1225 Use Array Functions

There are a lot of native PHP functions for arrays. It is often faster to take advantage of them than write a loop.

- array_push() : use array_merge()
- array_slice() : use array_chunk()
- index access : use array_column()
- append *[]*: use array_merge()
- addition : use array_sum()

- multiplication : use array_product()
- concatenation : use implode()
- ifthen : use array_filter()

```
<?php

$all = implode('-', $s).'-';

// same as above
$all = '';
foreach($array as $s) {
    $all .= $s . '-';
}

?>
```

See also Array Functions and *No array_merge() In Loops*.

Suggestions

- Remove the loop and use a native PHP function
- Add more expressions to the loop : batching multiple operations in one loop makes it more interesting than running separates loops.

Specs

Short name	Structures/UseArrayFunctions
Rulesets	*Suggestions*
Exakt since	1.8.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1226 Use Case Value

When switch() has branched to the right case, the value of the switched variable is know : it is the case.

This doesn't work with complex expression cases, nor with default.

```
<?php

switch($a) {
    case 'a' :
        // $a == 'a';
        echo $a;
        break;

    case 'b' :
        // $a == 'b';
        echo 'b';
```

(continues on next page)

(continued from previous page)

```
        break;
}

?>
```

Suggestions

- Use the literal value in the case, to avoid unnecessary computation.

Specs

Short name	Structures/UseCaseValue
Rulesets	*Suggestions*
Exakt since	1.9.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1227 Use Constant

The following functioncall have a constant equivalent, that is faster to use than calling the functions.

This applies to the following functions :

- pi() : replace with *M_PI*
- phpversion() : replace with *PHP_VERSION*
- php_sapi_name() : replace with *PHP_SAPI_NAME*

```
<?php

// recommended way
echo PHP_VERSION;

// slow version
echo php_version();

?>
```

See also PHP why 'pi() and M_PI <https://stackoverflow.com/questions/42021176/php-why-pi-and-m-pi>'_.

Suggestions

- Use the constant version, not the function.

Specs

Short name	Structures/UseConstant
Rulesets	*Analyze*, *CI-checks*, *php-cs-fixable*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1228 Use Count Recursive

The code could use the recursive version of count.

The second argument of count, when set to `COUNT_RECURSIVE`, count recursively the elements. It also counts the elements themselves.

```
<?php

$array = array( array(1,2,3), array(4,5,6));

print (count($array, COUNT_RECURSIVE) - count($array, COUNT_NORMAL));

$count = 0;
foreach($array as $a) {
    $count += count($a);
}
print $count;

?>
```

See also count.

Suggestions

- Drop the loop and use the 2nd argument of count()

Specs

Short name	Structures/UseCountRecursive
Rulesets	*Suggestions*
Exakt since	1.1.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*WordPress*, *PrestaShop*

13.2.1229 Use Debug

The code source includes calls to debug functions.

The following debug functions and libraries are reported :

- Aronduby Dump
- Cakephp Debug Toolbar
- Kint
- Krumo
- Nette tracy
- php-debugbar
- PHP native functions : print_r(), var_dump(), debug_backtrace(), debug_print_backtrace(), debug_zval_dump()
- Symfony debug
- Wordpress debug
- Xdebug
- Zend debug

```
<?php

// Example with Zend Debug
Zend\Debug\Debug::dump($var, $label = null, $echo = true);

?>
```

Specs

Short name	Structures/UseDebug
Rulesets	*CE*
Exakt since	0.11.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1230 Avoid get_class()

`get_class()` should be replaced with the `instanceof` operator to check the class of an object.

`get_class()` only compares the full namespace name of the object's class, while `instanceof` actually resolves the name, using the local namespace and aliases.

```
<?php

    use Stdclass as baseClass;

    function foo($arg) {
        // Slow and prone to namespace errors
```

(continues on next page)

(continued from previous page)

```
        if (get_class($arg) === 'Stdclass') {
            // doSomething()
        }
    }

    function bar($arg) {
        // Faster, and uses aliases.
        if ($arg instanceof baseClass) {
            // doSomething()
        }
    }
?>
```

See also get_class and Instanceof.

Specs

Short name	Structures/UseInstanceof
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1231 Useless Brackets

Standalone brackets have no use. Brackets are used to delimit a block of code, and are used by control statements. They may also be used to protect variables in strings.

Standalone brackets may be a left over of an old instruction, or a misunderstanding of the alternative syntax.

```
<?php

// The following brackets are useless : they are a leftover from an older instruction
// if (DEBUG)
{
    $a = 1;
}

// Here, the extra brackets are useless
for($a = 2; $a < 5; $a++) : {
    $b++;
} endfor;

?>
```

Suggestions

- Remove the brackets
- Restore the flow-control operation that was there and removed

- Move the block into a method or function, and call it

Specs

Short name	Structures/UselessBrackets
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*ChurchCRM*, *Piwigo*

13.2.1232 Useless Type Casting

There is no need to overcast returned values.

```
<?php

// trim always returns a string : cast is useless
$a = (string) trim($b);

// strpos doesn't always returns an integer : cast is useful
$a = (boolean) strpos($b, $c);

// comparison don't need casting, nor parenthesis
$c = (bool) ($b > 2);

?>
```

See also Type juggling.

Suggestions

- Remove the type cast

Specs

Short name	Structures/UselessCasting
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.7
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	Very high
Examples	*FuelCMS*, *ThinkPHP*

13.2.1233 Useless Check

There is no need to check the size of an array content before using foreach. Foreach() applies a test on the source, and skips the loop if no element is found.

```
<?php

// Checking for type is good.
if (is_array($array)) {
    foreach($array as $a) {
        doSomething($a);
    }
}

// Foreach on empty arrays doesn't start. Checking is useless
if (!empty($array)) {
    foreach($array as $a) {
        doSomething($a);
    }
}

?>
```

This analysis checks for conditions with sizeof() and count(). Conditions with isset() and empty() are omitted : they also check for the variable existence, and thus, offer extra coverage.

See also foreach.

Suggestions

- Drop the condition and the check
- Turn the condition into isset(), empty() and is_array()

Specs

Short name	Structures/UselessCheck
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.9
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Magento*, *Phinx*

13.2.1234 Useless Global

Global are useless in two cases. First, on super-globals, which are always globals, like $_GET; secondly, on variables that are not used.

```
<?php

// $_POST is already a global : it is in fact a global everywhere
```

(continues on next page)

(continued from previous page)

```
global $_POST;

// $unused is useless
function foo() {
    global $used, $unused;

    ++$used;
}

?>
```

Also, PHP has superglobals, a special team of variables that are always available, whatever the context. They are : $GLOBALS, $_SERVER, $_GET, $_POST, $_FILES, $_COOKIE, $_SESSION, $_REQUEST and $_ENV.

Suggestions

- Drop the global expression

Specs

Short name	Structures/UselessGlobal
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*Zencart*, *HuMo-Gen*

13.2.1235 Useless Instructions

Those instructions are useless, or contains useless parts.

For example, an addition whose result is not stored in a variable, or immediately used, does nothing : it is actually performed, and the result is lost. Just plain lost. In fact, PHP might detect it, and optimize it away.

Here the useless instructions that are spotted :

```
<?php

// Concatenating with an empty string is useless.
$string = 'This part '.$is.' useful but '.$not.'';

// This is a typo, that PHP turns into a constant, then a string, then nothing.
continue;

// Empty string in a concatenation
$a = 'abc' . '';

// Returning expression, whose result is not used (additions, comparisons, properties,
↪ closures, new without =, ...)
1 + 2;
```

(continues on next page)

(continued from previous page)

```
// Returning post-incrementation
function foo($a) {
    return $a++;
}

// array_replace() with only one argument
$replaced = array_replace($array);
// array_replace() is OK with ...
$replaced = array_replace(...$array);

// @ operator on source array, in foreach, or when assigning literals
$array = @array(1,2,3);

// Multiple comparisons in a for loop : only the last is actually used.
for($i = 0; $j = 0; $j < 10, $i < 20; ++$j, ++$i) {
    print $i.' '.$j.PHP_EOL;
}

// Counting the keys and counting the array is the same.
$c = count(array_keys($array))

//array_keys already provides an array with only unique values, as they were keys in␣
↪a previous array
$d = array_unique(array_keys($file['messages']))

// No need for assignation inside the ternary operator
$closeQuote = $openQuote[3] === "'" ? substr($openQuote, 4, -2) : $closeQuote =␣
↪substr($openQuote, 3);

?>
```

Suggestions

- Remove the extra semi-colon
- Remove the useless instruction
- Assign this expression to a variable and make use of it

Specs

Short name	Structures/UselessInstruction
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-useless-instruction

13.2.1236 Useless Parenthesis

Situations where parenthesis are not necessary, and may be removed.

Parenthesis group several elements together, and allows for a more readable expression. They are used with logical and mathematical expressions. They are necessary when the precedence of the operators are not the intended execution order : for example, when an addition must be performed before the multiplication.

Sometimes, the parenthesis provide the same execution order than the default order : they are deemed useless.

```
<?php

    if ( ($condition) ) {}
    while( ($condition) ) {}
    do $a++; while ( ($condition) );

    switch ( ($a) ) {}
    $y = (1);
    ($y) == (1);

    f(($x));

    // = has precedence over ==
    ($a = $b) == $c;

    ($a++);

    // No need for parenthesis in default values
    function foo($c = ( 1 + 2) ) {}
?>
```

See also Operators Precedence.

Suggestions

- Remove useless parenthesis, unless they are important for readability.

Specs

Short name	Structures/UselessParenthesis
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Mautic*, *Woocommerce*

13.2.1237 Useless Switch

This switch has only one case. It may very well be replaced by a ifthen structure.

```
<?php
switch($a) {
    case 1:
        doSomething();
        break;
}

// Same as

if ($a == 1) {
    doSomething();
}
?>
```

Suggestions

- Turn the switch into a if/then for better readability
- Add other cases to the switch, making it adapted to the situation

Specs

Short name	Structures/UselessSwitch
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Phpdocumentor*, *Dolphin*

13.2.1238 Useless Unset

There are situations where trying to remove a variable is actually useless.

PHP ignores any command that tries to unset a global variable, a static variable, or a blind variable from a foreach loop.

This is different from the garbage collector, which is run on its own schedule. It is also different from an explicit unset, aimed at freeing memory early : those are useful.

```
<?php

function foo($a) {
    // unsetting arguments is useless
    unset($a);

    global $b;
    // unsetting global variable has no effect
    unset($b);

    static $c;
```

(continues on next page)

(continued from previous page)

```
    // unsetting static variable has no effect
    unset($c);

    foreach($d as &$e){
        // unsetting a blind variable is useless
        (unset) $e;
    }
    // Unsetting a blind variable AFTER the loop is good.
    unset($e);
}

?>
```

See also unset.

Suggestions

- Remove the unset
- Set the variable to null : the effect is the same on memory, but the variable keeps its existence.
- Omit unsetting variables, and wait for the end of the scope. That way, PHP free memory en mass.

Specs

Short name	Structures/UselessUnset
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-useless-unset
Examples	*Tine20*, *Typo3*

13.2.1239 Use List With Foreach

Foreach() structures accepts list() as blind key. If the loop-value is an array with a fixed structure, it is possible to extract the values directly into variables with explicit names.

```
<?php

// Short way to assign variables
// Works on PHP 7.1, where list() accepts keys.
foreach($names as list('first' => $first, 'last' => $last)) {
    doSomething($first, $last);
}

// Short way to assign variables
// Works on all PHP versions with numerically indexed arrays.
foreach($names as list($first, $last)) {
```

(continues on next page)

(continued from previous page)

```
    doSomething($first, $last);
}

// Long way to assign variables
foreach($names as $name) {
    $first = $name['first'];
    $last = $name['last'];

    doSomething($first, $last);
}

?>
```

See also list and foreach.

Suggestions

- Use the list keyword (or the short syntax), and simplify the array calls in the loop.

Specs

Short name	Structures/UseListWithForeach
Rulesets	*Suggestions*, *Top10*
Exakt since	1.0.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high
Examples	*MediaWiki*

13.2.1240 Use Positive Condition

Whenever possible, use a positive condition.

Positive conditions are easier to understand, and lead to less understanding problems. Negative conditions are not reported when else is not present.

```
<?php

// This is a positive condition
if ($a == 'b') {
    doSomething();
} else {
    doSomethingElse();
}

if (!empty($a)) {
    doSomething();
} else {
    doSomethingElse();
}
```

(continues on next page)

(continued from previous page)

```
// This is a negative condition
if ($a == 'b') {
    doSomethingElse();
} else {
    doSomething();
}

// No need to force $a == 'b' with empty else
if ($a != 'b') {
    doSomethingElse();
}


?>
```

Suggestions

- Invert the code in the if branches, and the condition

Specs

Short name	Structures/UsePositiveCondition
Rulesets	*Analyze*
Exakt since	0.8.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*SPIP*, *ExpressionEngine*

13.2.1241 Use System Tmp

It is recommended to avoid hardcoding the temporary file. It is better to rely on the system's temporary folder, which is accessible with sys_get_temp_dir().

```
<?php

// Where the tmp is :
file_put_contents(sys_get_temp_dir().'/tempFile.txt', $content);


// Avoid hard-coding tmp folder :
// On Linux-like systems
file_put_contents('/tmp/tempFile.txt', $content);

// On Windows systems
file_put_contents('C:\WINDOWS\TEMP\tempFile.txt', $content);

?>
```

See also PHP: When is /tmp not /tmp?.

Suggestions

- Do not hardcode the temporary file, use the system's

Specs

Short name	Structures/UseSystemTmp
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.1242 Use Url Query Functions

PHP features several functions dedicated to processing URL's query string.

- parse_str()
- parse_url()
- http_build_query()

Those functions include extra checks : for example, http_build_query() adds urlencode() call on the values, and allow for choosing the separator and the Query string format.

```
<?php
$data = array(
    'foo' => 'bar',
    'baz' => 'boom',
    'cow' => 'milk',
    'php' => 'hypertext processor'
);

// safe and efficient way to build a query string
echo http_build_query($data, '', '&') . PHP_EOL;

// slow way to produce a query string
foreach($data as $name => &$value) {
    $value = $name.'='.$value;
}
echo implode('&', $data) . PHP_EOL;

?>
```

Suggestions

-

Specs

Short name	Structures/UseUrlQueryFunctions
Rulesets	*Suggestions*
Exakt since	1.9.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1243 var_dump()... Usage

var_dump(), print_r() or var_export() should not be left in any production code. They are debugging functions.

```
<?php

if ($error) {
    // Debugging usage of var_dump
    // And major security problem
    var_dump($query);

    // This is OK : the $query is logged, and not displayed
    $this->log(print_r($query, true));
}

?>
```

They may be tolerated during development time, but must be removed so as not to have any chance to be run in production.

Suggestions

- Remove usage of var_dump(), print_r(), var_export() without 2nd argument, and other debug functions.
- Push all logging to an external file, instead of the browser.

Specs

Short name	Structures/VardumpUsage
Rulesets	*Analyze*, *CI-checks*, *Security*
Exakt since	0.8.4
Php Version	All
Severity	Critical
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-debug-code
Examples	*Tine20*, *Piwigo*

13.2.1244 Variable Global

Variable global such are valid in PHP 5.6, but no in PHP 7.0. They should be replaced with ${$foo->bar}.

```
<?php

// Forbidden in PHP 7
global $normalGlobal;

// Forbidden in PHP 7
global $$variable->global ;

// Tolerated in PHP 7
global ${$variable->global};

?>
```

Specs

Short name	Structures/VariableGlobal
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.3
Php Version	7.0-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1245 Variable May Be Non-Global

Static and global keywords should be used as early as possible in a method.

Performance wise, it is better to call `global` or `static` only before using the variable.

Human-wise, it is recommended to put `global` or `static` at the beginning of the method, for better readability.

```
<?php

function foo() {
    // $a is not global yet. It is a local variable
    $a = 1;
    // Same for static variables
    $s = 5;

    // Now $a is global
    global $a;
    $a = 3;

    // Now $s is static
    static $s;
    $s = 55;
}

?>
```

See also Using ‘static variables <https://www.php.net/manual/en/language.variables.scope.php#language.variables.scope.‘static <https://www.php.net/manual/en/language.oop5.static.php>‘_>‘_ and The global keyword.

Suggestions

- Use static and global at the beginning of the method
- Move static and global to the first usage of the variable
- Remove any access to the variable before static and global

Specs

Short name	Structures/VariableMayBeNonGlobal
Rulesets	none
Exakt since	1.5.3
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1246 While(List() = Each())

This code structure is quite old : it should be replace by the more modern and efficient foreach.

This structure is deprecated since PHP 7.2. It may disappear in the future.

```
<?php

    while(list($key, $value) = each($array)) {
        doSomethingWith($key) and $value();
    }

    foreach($array as $key => $value) {
        doSomethingWith($key) and $value();
    }
?>
```

See also PHP RFC: Deprecations for PHP 7.2 : ‘Each() <https://wiki.php.net/rfc/deprecations_php_7_2#each>‘_.

Suggestions

- Change this loop with foreach
- Change this loop with an array_* function with a callback

Specs

Short name	Structures/WhileListEach
Rulesets	*Analyze*, *CI-checks*, *Performances*, *Suggestions*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*OpenEMR*, *Dolphin*

13.2.1247 Wrong Range Check

The interval check should use && and not ||.

```
<?php

//interval correctly checked a is between 2 and 999
if ($a > 1 && $a < 1000) {}

//interval incorrectly checked : a is 2 or more ($a < 1000 is never checked)
if ($a > 1 || $a < 1000) {}

?>
```

Suggestions

- Make the interval easy to read and understand
- Check the truth table for the logical operation

Specs

Short name	Structures/WrongRange
Rulesets	*Analyze*
Exakt since	1.2.5
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Dolibarr*, *WordPress*

13.2.1248 Yoda Comparison

Yoda comparison is a way to write conditions which places literal values on the left side.

```
<?php
  if (1 == $a) {
    // Then condition
```

(continues on next page)

(continued from previous page)

```
  }
?>
```

The objective is to avoid mistaking a comparison to an assignation. If the comparison operator is mistaken, but the literal is on the left, then an error will be triggered, instead of a silent bug.

```
<?php
    // error in comparison!
    if ($a = 1) {
        // Then condition
    }
?>
```

See also Yoda Conditions, Yoda Conditions: To Yoda or Not to Yoda.

Specs

Short name	Structures/YodaComparison
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1249 Already Parents Trait

Trait is already used a parent's class or trait. There is no use to include it a second time.

```
<?php

trait ta {
    use tb;
}

trait t1 {
    use ta;
    use tb; // also used by ta
}

class b {
    use t1; // also required by class c
    use ta; // also required by trait t1
}

class c extends b {
    use t1;
}

?>
```

See also Traits.

Suggestions

- Eliminate one of the trait request

Specs

Short name	Traits/AlreadyParentsTrait
Rulesets	*Analyze*
Exakt since	1.8.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1250 Could Use Trait

The following classes have been found implementing all of a trait's methods : it could use this trait, and remove duplicated code.

```
<?php

trait t {
    function t1() {}
    function t2() {}
    function t3() {}
}

// t1, t2, t3 method could be dropped, and replaced with 'use t'
class foo1 {
    function t1() {}
    function t2() {}
    function t3() {}

    function j() {}
}

// foo2 is just the same as foo1
class foo2 {
    use t;

    function j() {}
}

?>
```

The comparison between the class methods' and the trait's methods are based on token. They may yield some false-positives.

See also Interfaces/CouldUseInterface.

Suggestions

- Use trait, and remove duplicated code

Specs

Short name	Traits/CouldUseTrait
Rulesets	none
Exakt since	1.8.5
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1251 Dependant Trait

> Traits should be autonomous. It is recommended to avoid depending on methods or properties that should be in the using class.

The following traits make usage of methods and properties, static or not, that are not defined in the trait. This means the host class must provide those methods and properties, but there is no way to enforce this.

This may also lead to dead code : when the trait is removed, the host class have unused properties and methods.

```
<?php

// autonomous trait : all it needs is within the trait
trait t {
    private $p = 0;

    function foo() {
        return ++$this->p;
    }
}

// dependant trait : the host class needs to provide some properties or methods
trait t2 {
    function foo() {
        return ++$this->p;
    }
}

class x {
    use t2;

    private $p = 0;
}
?>
```

See also Classes/DependantAbstractClass.

Suggestions

- Add local property definitions to make the trait independent
- Make the trait only use its own resources
- Split the trait in autonomous traits

Specs

Short name	Traits/DependantTrait
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Zencart*

13.2.1252 Empty Traits

List of all empty trait defined in the code.

```
<?php

// empty trait
trait t { }

// Another empty trait
trait t2 {
    use t;
}

?>
```

Such traits may be reserved for future use. They may also be forgotten, and dead code.

Suggestions

- Add some code to the trait
- Remove the trait

Specs

Short name	Traits/EmptyTrait
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1253 Is Extension Trait

Indicates if this trait is defined in an extension or not.

Specs

Short name	Traits/IsExtTrait
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1254 Locally Used Property In Trait

Properties that are used in the class where they are defined.

```
<?php

trait foo {
    public $unused, $used;// property $unused is never used in this class

    function bar() {
        $this->used++; // property $used is used in this method
    }
}

class X {
    use foo;
}

$foo = new X();
$foo->unused = 'here'; // property $unused is used outside the trait definition
?>
```

Specs

Short name	Traits/LocallyUsedProperty
Rulesets	none
Exakt since	1.3.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1255 Method Collision Traits

Two or more traits are included in the same class, and they have methods collisions.

Those collisions should be solved with a `use` expression. When they are not, PHP stops execution with a fatal error : `Trait method M has not been applied, because there are collisions with other trait methods on C`.

```
<?php

trait A {
    public function A() {}
    public function M() {}
}

trait B {
    public function B() {}
    public function M() {}
}

class C {
    use  A, B;
}

class D {
    use  A, B{
        B::M insteadof A;
    };
}

?>
```

The code above lints, but doesn't execute.

See also Traits.

Specs

Short name	Traits/MethodCollisionTraits
Rulesets	*Analyze*, *LintButWontExec*
Exakt since	1.4.2
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.1256 Multiple Usage Of Same Trait

The same trait is used several times. One trait usage is sufficient.

```
<?php

// C is used twice, and could be dropped from B
trait A { use B, C;}
trait B { use C;}

?>
```

PHP doesn't raise any error when traits are included multiple times.

See also Traits.

Suggestions

- Remove any multiple traits from use expressions
- Review the class tree, and remove any trait mentioned multiple times

Specs

Short name	Traits/MultipleUsage
Rulesets	*Suggestions*
Exakt since	1.5.7
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*NextCloud*

13.2.1257 Redefined PHP Traits

List of all traits that bears name of a PHP trait. Although, at the moment, there are no PHP trait defined.

Specs

Short name	Traits/Php
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1258 Self Using Trait

Trait uses itself : this is unnecessary. Traits may use themselves, or be used by other traits, that are using the initial trait itself.

PHP handles the situation quietly, by ignoring all extra use of the same trait, keeping only one valid version.

```
<?php

// empty, but valid
trait a {}

// obvious self usage
trait b { use b; }

// less obvious self usage
trait c { use d, e, f, g, h, c; }

// level 2 self usage
```

(continues on next page)

(continued from previous page)

```
trait i { use j; }
trait j { use i; }

?>
```

See also Traits.

Suggestions

- Remove the extra usage of the trait.

Specs

Short name	Traits/SelfUsingTrait
Rulesets	*ClassReview*, *Dead code*
Exakt since	1.5.7
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1259 Trait Methods

List the names of the methods in a trait.

```
<?php

trait t {
    private $property = 1;

    // This is an interface method name
    function foo() {
        // This is not a trait method
        return function($a) { return $a + 1; }
    }
}

?>
```

Specs

Short name	Traits/TraitMethod
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1260 Trait Names

List all the traits names in the code.

```
<?php

// This trait is called 't'
trait t {}

?>
```

See also Traits.

Specs

Short name	Traits/Traitnames
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1261 Trait Not Found

A unknown trait is mentioned in the use expression.

The used traits all exist, but in the configuration block, some unmentioned trait is called.

Be aware that the traits used in any configuration block may originate in any use expression. PHP will check the configuration block at instantiation only, and after compiling : at that moment, it will know all the used traits across the class.

```
<?php
class x  {
    // c is not a used trait
    use a, b { c::d insteadof e;}

    // e is a used trait, even if is not in the use above.
    use e;
}
?>
```

See also Traits.

Suggestions

- Switch the name of the trait to an existing and used trait
- Drop the expression that rely on the non-existent trait

Specs

Short name	Traits/TraitNotFound
Rulesets	*Analyze*, *LintButWontExec*
Exakt since	1.7.9
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1262 Traits Usage

Usage of traits in the code.

```
<?php

trait t {
    function t() {
        echo 'I\'m in t';
    }
}

class foo {
    use t;
}

$x = new foo();
$x->t();

?>
```

See also Traits.

Specs

Short name	Traits/TraitUsage
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1263 Undefined Insteadof

`Insteadof` tries to replace a method with another, but it doesn't exists. This happens when the replacing class is refactored, and some of its definition are dropped.

`Insteadof` may replace a non-existing method with an existing one, but not the contrary.

```
<?php

trait A {
    function C (){}
}

trait B {
    function C (){}
}

class Talker {
    use A, B {
        B::C insteadof A;
        B::D insteadof A;
    }
}

new Talker();
?>
```

This error is not linted : it only appears at execution time.

See also Traits.

Suggestions

- Remove the insteadof expression
- Fix the original method and replace it with an existing method

Specs

Short name	Traits/UndefinedInsteadof
Rulesets	*Analyze*, *CI-checks*, *LintButWontExec*
Exakt since	1.4.2
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.1264 Undefined Trait

Those are undefined, traits .

When the using class or trait is instantiated, PHP emits a a fatal error.

```
<?php

use Composer/Component/someTrait as externalTrait;

trait t {
    function foo() {}
}
```

(continues on next page)

(continued from previous page)

```
// This class uses trait that are all known
class hasOnlyDefinedTrait {
    use t, externalTrait;
}

// This class uses trait that are unknown
class hasUndefinedTrait {
    use unknownTrait, t, externalTrait;
}
?>
```

Trait which are referenced in a *use* expression are omitted: they are considered part of code that is probably outside the current code, either omitted or in external component.

Suggestions

- Define the missing trait
- Remove usage of the missing trait

Specs

Short name	Traits/UndefinedTrait
Rulesets	*Analyze*, *CI-checks*, *LintButWontExec*
Exakt since	0.8.4
Php Version	All
Severity	Critical
Time To Fix	Quick (30 mins)
Precision	High

13.2.1265 Unused Trait In Class

A trait has been summoned in a class, but is not used. Traits may be used as a copy/paste of code, bringing a batch of methods and properties to a class. In the current case, the imported trait is never called. As such, it may be removed.

Currently, the analysis covers only traits that are used in the class where they are imported. Also, the properties are not covered yet.

```
<?php

trait t {
    function foo() { return 1;}
}

// this class imports and uses the trait
class UsingTrait {
    use t;

    function bar() {
```

(continues on next page)

(continued from previous page)

```
        return $this->foo() + 1;
    }
}

// this class imports but doesn't uses the trait
class UsingTrait {
    use t;

    function bar() {
        return 1;
    }
}

?>
```

There are some sneaky situations, where a trait falls into decay : for example, creating a method in the importing class, with the name of a trait class, will exclude the trait method, as the class method has priority. Other precedence rules may lead to the same effect.

See also Traits.

Suggestions

- Remove the trait from the class
- Actually use the trait, at least in the importing class
- Use conflict resolution to make the trait accessible

Specs

Short name	Traits/UnusedClassTrait
Rulesets	*ClassReview*
Exakt since	2.1.1
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1266 Unused Traits

Those traits are not used in a class or another trait. They may be dead code.

```
<?php

// unused trait
trait unusedTrait { /**/ }

// used trait
trait tUsedInTrait { /**/ }

trait tUsedInClass {
```

(continues on next page)

(continued from previous page)

```
    use tUsedInTrait;
    /**/
    }

class foo {
    use tUsedInClass;
}
?>
```

Specs

Short name	Traits/UnusedTrait
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1267 Used Trait

Mark a trait as being used by a class.

```
<?php

// One used trait
trait usedTrait {}

// One unused trait
trait unusedTrait {}

class foo {
    use usedTrait;
}

?>
```

Specs

Short name	Traits/UsedTrait
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1268 Useless Alias

It is not possible to declare an alias of a method with the same name.

PHP reports that `Trait method f has not been applied, because there are collisions with other trait methods on x`, which is a way to say that the alias will be in conflict with the method name.

When the method is the only one bearing a name, and being imported, there is no need to alias it. When the method is imported in several traits, the keyword `insteadof` is available to solve the conflict.

```
<?php

trait t {
    function h() {}
}

class x {
    use t {
        // This is possible
        t::f as g;

        // This is not possible, as the alias is in conflict with itself
        // alias are case insensitive
        t::f as f;
    }
}

?>
```

This code lints but doesn't execute.

See also Conflict resolution.

Suggestions

- Remove the alias
- Fix the alias or the origin method name
- Switch to insteadof, and avoid as keyword

Specs

Short name	Traits/UselessAlias
Rulesets	*Analyze*, *CI-checks*, *LintButWontExec*
Exakt since	1.5.6
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.1269 Type Array Index

All literal index used in the code.

```
<?php

// index is an index. it is read
$array['index'] = 1;

// another_index and second_level are read
$array[] = $array['another_index']['second_level'];

// variables index are not reported
$array[$variable] = 1;

?>
```

Specs

Short name	Type/ArrayIndex
Rulesets	*CE*
Exakt since	1.0.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1270 Binary Glossary

List of all the integer values using the binary format.

```
<?php

$a = 0b10;
$b = 0B0101;

?>
```

Specs

Short name	Type/Binary
Rulesets	*CE*, *CompatibilityPHP53*
Exakt since	0.8.4
Php Version	5.4+
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1271 All strings

Strings and heredocs in one place.

```
<?php

$string = 'string';

$query = <<<SQL
Heredoc
SQL;

?>
```

Specs

Short name	Type/CharString
Rulesets	none
Exakt since	0.10.1
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1272 Continents

List of all the continents mentioned in the code.

Specs

Short name	Type/Continents
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1273 Duplicate Literal

Report literals that are repeated across the code. The minimum replication is 5, and is configurable with `maxDuplicate`.

Repeated literals should be considered a prime candidate for constants.

Integer, reals and strings are considered here. Boolean, Null and Arrays are omitted. 0, 1, 2, 10 and the empty string are all omitted, as too common.

```
<?php
    // array index are omitted
    $x[3] = 'b';

    // constanst are omitted
```

(continues on next page)

(continued from previous page)

```
    const X = 11;
    define('Y', 'string')

    // 0, 1, 2, 10 are omitted
    $x = 0;

?>
```

Suggestions

- Create a constant and use it in place of the literal
- Create a class constant and use it in place of the literal

Name	Default	Type	Description
minDuplicate	15	integer	Minimal number of duplication before the literal is reported.

Specs

Short name	Type/DuplicateLiteral
Rulesets	*Semantics*
Exakt since	1.9.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1274 Email Addresses

List of all the email addresses that were found in the code.

Emails are detected with regex : `[_A-Za-z0-9-]+(\\.[_A-Za-z0-9-]+)*`@ <https://www.php.net/manual/en/language.operators.errorcontrol.php>`_[A-Za-z0-9]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})`

```
<?php

$email = 'contact@exakat.io';

?>
```

Specs

Short name	Type/Email
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1275 Incoming Variable Index Inventory

This collects all the index used in incoming variables : $_GET, $_POST, $_REQUEST, $_COOKIE.

```
<?php

// x is collected
echo $_GET['x'];

// y is collected, but no z.
echo $_POST['y']['z'];

// a is not collected
echo $_ENV['s'];

?>
```

Specs

Short name	Type/GPCIndex
Rulesets	*CE*
Exakt since	1.0.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1276 Heredoc Delimiter Glossary

List of all the delimiters used to build a Heredoc string.

In the example below, `EOD` is the delimiter.

```
<?php

$a = <<<EOD
heredoc
EOD;

?>
```

See also Heredoc.

Specs

Short name	Type/Heredoc
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1277 Hexadecimal Glossary

List of all the integer values using the hexadecimal format.

```
<?php

$hexadecimal = 0x10;

$anotherHexadecimal =0XAF;

?>
```

See also Integer Syntax.

Specs

Short name	Type/Hexadecimal
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1278 Hexadecimal In String

Mark strings that may be confused with hexadecimal.

Until PHP 7.0, PHP recognizes hexadecimal numbers inside strings, and converts them accordingly.

PHP 7.0 and until 7.1, converts the string to 0, silently.

PHP 7.1 and later, emits a 'A non-numeric value encountered' warning, and convert the string to 0.

```
<?php
    $a = '0x0030';
    print $a + 1;
    // Print 49
```

(continues on next page)

(continued from previous page)

```
    $c = '0x0030zyc';
    print $c + 1;
    // Print 49

    $b = 'b0x0030';
    print $b + 1;
    // Print 0
?>
```

Specs

Short name	Type/HexadecimalString
Rulesets	*CompatibilityPHP70*, *CompatibilityPHP71*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.1279 Http Headers

List of HTTP headers use in the code.

```
<?php

header('Location: http://www.example.com/');

// Parseable headers are also reported
header('Location: http://www.example.com/');

// UnParseable headers are not reported
header('GarbagexxxxXXXXxxxGarbagexxxxXXXXxxx');
header($header);

?>
```

Those headers are mostly used with header() function to send to browser.

See also List of HTTP header fields.

Specs

Short name	Type/HttpHeader
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1280 HTTP Status Code

List of all the HTTP status codes mentioned in the code.

```
<?php

http_response_code(418);

header('HTTP/1.1 418 I\'m a teapot');

?>
```

See also List of HTTP status codes.

Specs

Short name	Type/HttpStatus
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1281 Malformed Octal

Those numbers starts with a 0, so they are using the PHP octal convention. Therefore, one can't use 8 or 9 figures in those numbers, as they don't belong to the octal base. The resulting number will be truncated at the first erroneous figure. For example, 090 is actually 0, and 02689 is actually 22.

```
<?php

// A long way to write 0 in PHP 5
$a = 0890;

// A fatal error since PHP 7

?>
```

Also, note that very large octal, usually with more than 21 figures, will be turned into a real number and undergo a reduction in precision.

See also Integers.

Specs

Short name	Type/MalformedOctal
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	7.0-
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1282 Md5 Strings

List of all the MD5 values hard coded in the application.

MD5 values are detected as hexadecimal strings, of length 32. No attempt at recognizing the origin value is made, so any such strings, including dummy '11111111111111111111111111111111' are reported.

```
<?php
    // 32
   $a = '0cc175b9c0f1b6a831c399e269771111';

?>
```

See also MD5.

Specs

Short name	Type/Md5String
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1283 Mime Types

List of Mime Types that are mentioned in the code.

```
<?php

$mimeType = 'multipart/form-data';
$mimeType = 'image/jpeg';
$mimeType = 'application/zip';

header('Content-Type: '.$mimeType);

?>
```

See also Media Type and MIME.

Specs

Short name	Type/MimeType
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1284 No Real Comparison

Avoid comparing decimal numbers with ==, ===, !==, !=. Real numbers have an error margin which is random, and makes it very difficult to match even if the compared value is a literal.

PHP uses an internal representation in base 2 : any number difficult to represent with this base (like 0.1 or 0.7) will have a margin of error.

```
<?php

$a = 1/7;
$b = 2.0;

// 7 * $a is a real, not an integer
var_dump( 7 * $a === 1);

// rounding error leads to wrong comparison
var_dump( (0.1 + 0.7) * 10 == 8);
// although
var_dump( (0.1 + 0.7) * 10);
// displays 8

// precision formula to use with reals. Adapt 0.0001 to your precision needs
var_dump( abs(((0.1 + 0.7) * 10) - 8) < 0.0001);

?>
```

Use precision formulas with abs() to approximate values with a given precision, or avoid reals altogether.

See also Floating point numbers.

Suggestions

- Cast the values to integer before comparing
- Compute the difference, and keep it below a threshold
- Use the gmp or the bc extension to handle high precision numbers
- Change the 'precision' directive of PHP : ini_set('precision', 30) to make number larger
- Multiply by a power of ten, before casting to integer for the comparison
- Use floor(), ceil() or round() to compare the numbers, with a specific precision

Specs

Short name	Type/NoRealComparison
Rulesets	*Analyze*, *CI-checks*, *Top10*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
ClearPHP	no-real-comparison
Examples	*Magento*, *SPIP*

13.2.1285 Nowdoc Delimiter Glossary

List of all the delimiters used to build a Nowdoc string.

```
<?php
$nowdoc = <<<'EOD'

EOD;

?>
```

See also Nowdoc and Heredoc.

Specs

Short name	Type/Nowdoc
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1286 Octal Glossary

List of all the integer values using the octal format : an integer starting with an initial 0.

```
<?php

  $a = 1234; // decimal number
  $a = 0123; // octal number (equivalent to 83 decimal)

  // silently valid for PHP 5.x
  $a = 01283; // octal number (equivalent to 10 decimal)

?>
```

Putting an initial 0 is often innocuous, but in PHP, 0755 and 755 are not the same. The second is actually 1363 in octal, and will not provide the expected privileges.

See also Integers.

Specs

Short name	Type/Octal
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1287 Invalid Octal In String

Any octal sequence inside a string can't be go 377. Those will be a fatal error at parsing time.

The check is applied to the string, starting with PHP 7.1. In PHP 7.0 and older, those sequences were silently adapted (modulo/% 400).

```
<?php

// A valid octal in a PHP string
echo \100; // @

// Emit a warning in PHP 7.1
//Octal escape sequence overflow \500 is greater than \377
echo \500; // @

// Silent conversion
echo \478; // 8

?>
```

See also Integers.

Suggestions

- Use a double slash to avoid the sequence to be an octal sequence
- Use a function call, such as decoct() to convert larger number to octal notation

Specs

Short name	Type/OctalInString
Rulesets	*CompatibilityPHP71*
Exakt since	0.9.1
Php Version	7.1-
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1288 One Variable String

These strings only contains one variable or property or array.

```
<?php

$a = 0;
$b = "$a"; // This is a one-variable string

// Better way to write the above
$b = (string) $a;

// Alternatives :
$b2 = "$a[1]"; // This is a one-variable string
$b3 = "$a->b"; // This is a one-variable string
$c = "d";
$d = "D";
$b4 = "{$$c}";
$b5 = "{$a->foo()}";

?>
```

When the goal is to convert a variable to a string, it is recommended to use the type casting (string) operator : it is then clearer to understand the conversion. It is also marginally faster, though very little.

See also Strings and Type Juggling.

Suggestions

- Drop the surrounding string, keep the variable (or property. . .)
- Include in the string any concatenation that comes unconditionaly after or before
- Convert the variable to a string with the (type) operator

Specs

Short name	Type/OneVariableStrings
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*Tikiwiki*, *NextCloud*

13.2.1289 OpenSSL Ciphers Used

List of all the OpenSSL ciphers used in the code.

It is important to always use valid cipher modes for SSL. In case of non-existent cipher, the crypting or decrypting will not happen. Ciphers are marked as weak after their security is breached, and shall be removed from OpenSSL, and later, from PHP.

By reviewing this inventory, it is possible to detect forgotten ciphers, and fix them.

The full list of available ciphers for the PHP installation is available with the function openssl_get_cipher_methods().

```
<?php
// PHP documentation example, for PHP 7.1 and more recent
//$key should have been previously generated in a cryptographically safe way, like␣
↪openssl_random_pseudo_bytes
$plaintext = message to be encrypted;
$cipher = aes-128-gcm;
if (in_array($cipher, openssl_get_cipher_methods()))
{
    $ivlen = openssl_cipher_iv_length($cipher);
    $iv = openssl_random_pseudo_bytes($ivlen);
    $ciphertext = openssl_encrypt($plaintext, $cipher, $key, $options=0, $iv, $tag);
    //store $cipher, $iv, and $tag for decryption later
    $original_plaintext = openssl_decrypt($ciphertext, $cipher, $key, $options=0, $iv,
↪ $tag);
    echo $original_plaintext.\n;
}
?>
```

See also openssl_encrypt() and OpennSSL [PHP manual].

Suggestions

-

Specs

Short name	Type/OpensslCipher
Rulesets	none
Exakt since	2.1.1
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1290 Pack Format Inventory

All format used in the code with pack() and unpack().

```
<?php

$binarydata = "\x04\x00\xa0\x00";
$array = unpack("cn", $binarydata);
$initial = pack("cn", ...$array);

?>
```

Specs

Short name	Type/Pack
Rulesets	*CE*
Exakt since	1.5.0
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1291 Path lists

List of all paths that were found in the code.

Path are identified with this regex : `^(.*/)([^/]*)\.\w+$`. In particular, the directory delimiter is `/` : Windows delimiter `\` are not detected.

```
<?php

// the first argument is recognized as an URL
fopen('/tmp/my/file.txt', 'r+');

// the string argument  is recognized as an URL
$source = 'https://www.other-example.com/';

?>
```

URL are ignored when the protocol is present in the literal : `http://www.example.com` is not mistaken with a file.

See also Dir predefined constants and Supported Protocols and Wrappers.

Specs

Short name	Type/Path
Rulesets	*CE*
Exakt since	1.5.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1292 Perl Regex

List of all the Perl Regex (Pcre-style).

```
<?php

preg_match('/[abc]/', $haystack);

preg_replace('#[0-9A-Z]+#is', $y, $z);
```

(continues on next page)

(continued from previous page)

```
?>
```

Regex are spotted when they are literals : dynamically built regex, (including /$x/) are not reported.

Specs

Short name	Type/Pcre
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1293 Internet Ports

List of all the Internet ports mentioned in the code.

Ports are recognized based on a internal database of port. They are found in Integers.

```
<?php

// 21 is the default port for FTP
$ftp = ftp_connect($host, 21, $timeout = 90);

?>
```

See also List of TCP and UDP port numbers.

Specs

Short name	Type/Ports
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1294 Printf Format Inventory

All format used in the code with printf(), vprintf(), sprintf(), scanf() and fscanf().

```
<?php

// Display a number with 2 digits
echo printf("%'.2d\n", 123);
```

(continues on next page)

(continued from previous page)

```
?>
```

Specs

Short name	Type/Printf
Rulesets	*CE*
Exakt since	1.5.0
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1295 Protocol lists

List of all protocols that were found in the code.

From the manual : PHP comes with many built-in wrappers for various URL-style protocols for use with the filesystem functions such as fopen(), copy(), file_exists() and filesize().

```
<?php
// Example from the PHP manual, with the glob:// wrapper

// Loop over all *.php files in ext/spl/examples/ directory
// and print the filename and its size
$it = new DirectoryIterator(glob://ext/spl/examples/*.php);
foreach($it as $f) {
    printf(%s: %.1FK\n, $f->getFilename(), $f->getSize()/1024);
}
?>
```

See also Supported Protocols and Wrappers.

Specs

Short name	Type/Protocols
Rulesets	*CE*
Exakt since	2.1.3
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1296 Regex Inventory

All regex used in the code. PHP has the PCRE extension that handles all regex : preg_match(), preg_replace(), etc.

```
<?php

// PCRE regex used with preg_match
preg_match('/[abc]+/', $string);

// Mbstring regex, in the arabic range
if(mb_ereg('[\x{0600}-\x{06FF}]', $text))

?>
```

mbstring regexes are also collected. Pre-PHP 7.0 POSIX regex are not listed.

See also ‘ext/mbstring <http://www.php.net/manual/en/book.mbstring.php> ‘_ and ‘ext/pcre <http://www.php.net/manual/en/book.pcre.php> ‘_.

Specs

Short name	Type/Regex
Rulesets	*CE*
Exakt since	0.12.14
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1297 PHP Sapi

List of PHP SAPI mentioned in the code. When those SAPI are mentioned in strings, they are usually checked to take advantage of special characteristics. Check the code for portability.

```
<?php

require __DIR__.'/phpdbg.php';

$Phpdbg = new phpdbg();

?>
```

Specs

Short name	Type/Sapi
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1298 Shell commands

Shell commands, called from PHP.

Shell commands are detected with the italic quotes, and using shell_exec(), system(), exec() and proc_open().

```
<?php

// Shell command in a shell_exec() call
shell_exec('ls -1');

// Shell command with backtick operator
`ls -1 $path`;

?>
```

Specs

Short name	Type/Shellcommands
Rulesets	*CE*
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1299 Should Be Single Quote

Use single quote for simple strings.

Static content inside a string, that has no single quotes nor escape sequence (such as n or t), should be using single quote delimiter, instead of double quote.

```
<?php

$a = abc;

// This one is using a special sequence
$b = cde\n;

// This one is using two special sequences
$b = \x03\u{1F418};

?>
```

If you have too many of them, don't loose your time switching them all. If you have a few of them, it may be good for consistence.

Specs

Short name	Type/ShouldBeSingleQuote
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
ClearPHP	no-double-quote

13.2.1300 Should Typecast

When typecasting, it is better to use the casting operator, such as `(int)` or `(bool)`.

Functions such as intval() or settype() are always slower.

```
<?php

// Fast version
$int = (int) $X;

// Slow version
$int = intval($X);

// Convert to base 8 : can't use (int) for that
$int = intval($X, 8);


?>
```

This is a micro-optimisation, although such conversion may be use multiple time, leading to a larger performance increase.

Note that intval() may also be used to convert an integer to another base.

Suggestions

- Use a typecast, instead of a functioncall.

Specs

Short name	Type/ShouldTypecast
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*xataface*, *OpenConf*

13.2.1301 Silently Cast Integer

Those are integer literals that are cast to a float when running PHP. They are too big for the current PHP version, and PHP resorts to cast them into a float, which has a much larger capacity but a lower precision.

Compare your literals to `PHP_MAX_INT` (typically `9223372036854775807`) and `PHP_MIN_INT` (typically `-9223372036854775808`). This applies to binary (`0b10101...`), octal (`0123123...`) and hexadecimal (`0xfffff...`) too.

```
<?php

echo 0b10101011010101101010110101010110101010110101010110101010110101010110101111;
//6173123008118052203
echo 0b10101011010101101010110101010110101010110101010110101010110101010110101111;
//1.2346246016236E+19

echo 012312312312312312312312;
//1498121094048818771
echo 012312312312312312312312;
//1.1984968752391E+19

echo 0x12309812311230;
//5119979279159856
echo 0x12309812311230fed;
//2.0971435127439E+19

echo 9223372036854775807; //PHP_MAX_INT
//9223372036854775807
echo 9223372036854775808;
9.2233720368548E+18

?>
```

See also Integer overflow.

Suggestions

- Make sure hexadecimal numbers have the right number of digits : generally, it is 15, but it may depends on your PHP version.

Specs

Short name	Type/SilentlyCastInteger
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*MediaWiki*

13.2.1302 Similar Integers

This analysis reports all integer values that are expressed in different format.

```
<?php

// Three ways to write 10 (more available)
$a = 10;
$b = 012;
$x = 0xA;

// 7 is expressed in one way only
$d = 7;
$d = 7;

// Four ways to write 11 (more available)
$a = 11;
$b = 013;
$x = 0xB;
$x = -+-11;

// Expressions are not counted

?>
```

Suggestions

-

Specs

Short name	Type/SimilarIntegers
Rulesets	*Semantics*
Exakt since	1.9.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1303 Special Integers

Short and incomplete list of integers that may hold special values.

```
<?php

// 86400 is the number of seconds in a day
$day = 86400;

// 1400 is the number of minutes in a day
$day = 1440;

?>
```

The list includes powers of 2, duration in various units, factorial, ASCII codes and years.

Specs

Short name	Type/SpecialIntegers
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1304 SQL queries

SQL queries, detected in literal strings.

SQL queries are detected with keywords, inside literals or concatenations.

```
<?php

// SQL in a string
$query = 'SELECT name FROM users WHERE id = 1';

// SQL in a concatenation
$query = 'SELECT name FROM '.$table_users.' WHERE id = 1';

// SQL in a Heredoc
$query = <<<SQL
SELECT name FROM $table_users WHERE id = 1
SQL;

?>
```

Specs

Short name	Type/Sql
Rulesets	*CE*
Exakt since	0.10.1
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1305 String May Hold A Variable

Those strings looks like holding a variable.

Single quotes and Nowdoc syntax may include $ signs that are treated as literals, and not replaced with a variable value.

However, there are some potential variables in those strings, making it possible for an error : the variable was forgotten and will be published as such. It is worth checking the content and make sure those strings are not variables.

```
<?php

$a = 2;

// Explicit variable, but literal effect is needed
echo '$a is '.$a;

// One of the variable has been forgotten
echo '$a is $a';

// $CAD is not a variable, rather a currency unit
$total = 12;
echo $total.' $CAD';

// $CAD is not a variable, rather a currency unit
$total = 12;

// Here, $total has been forgotten
echo <<<'TEXT'
$total $CAD
TEXT;

?>
```

Specs

Short name	Type/StringHoldAVariable
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High

13.2.1306 Interpolation

The following strings contain variables that are will be replaced. However, the following characters are ambiguous, and may lead to confusion.

```
<?php

class b {
    public $b = 'c';
    function __toString() { return __CLASS__; }
}
$x = array(1 => new B());

// -> after the $x[1] looks like a 2nd dereferencing, but it is not.
print $x[1]->b;
// displays : b->b
```

(continues on next page)

(continued from previous page)

```
print {$x[1]->b};
// displays : c

?>
```

It is advised to add curly brackets around those structures to make them non-ambiguous.

See also Double quoted.

Specs

Short name	Type/StringInterpolation
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1307 Strings With Strange Space

An invisible space may be mistaken for a normal space.

However, PHP does straight comparisons, and may fail at recognizing. This analysis reports when it finds such strange spaces inside strings.

PHP doesn't mistake space and tables for whitespace when tokenizing the code.

This analysis doesn't report Unicode Codepoint Notation : those are visible in the code.

```
<?php

// PHP 7 notation,
$a = \u{3000};
$b = ;

// Displays false
var_dump($a === $b);

?>
```

See also Unicode spaces, and disallow irregular whitespace (no-irregular-whitespace).

Suggestions

- Replace the odd spaces with a normal space
- If unsecable spaces are important for presentation, add them at the templating level.

Specs

Short name	Type/StringWithStrangeSpace
Rulesets	*Analyze*, *CI-checks*
Exakt since	0.11.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*OpenEMR*, *Thelia*

13.2.1308 Internet Domains

List all internet domain used

Suggestions

-

Specs

Short name	Type/UdpDomains
Rulesets	none
Exakt since	1.9.6
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1309 Unicode Blocks

List of the Unicode blocks used in string literals.

This is the kind of characters that can be found in the applications strings.

```
<?php

$a = zoo;

$b = ; // Telugu character
$b = \u{0C12}; Same as above

$b = ; // Chinese Mandarin character
$b = \u{4EBA}; Same as above

?>
```

Note that Exakat only analyze PHP scripts : any translation available in a `.po` or external resource is not parsed and will not show.

See also Unicode block.

Specs

Short name	Type/UnicodeBlock
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1310 URL List

List of all the URL addresses that were found in the code.

```
<?php

// the first argument is recognized as an URL
ftp_connect('http://www.example.com/', $port, $timeout);

// the string argument  is recognized as an URL
$source = 'https://www.other-example.com/';

?>
```

See also Uniform Resource Identifier.

Suggestions

-

Specs

Short name	Type/Url
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1311 Could Be Array Typehint

This rule spots arguments, properties or return values that may be labeled with the `array` scalar typehint.

```
<?php

// $arg is used as an array in this function, so it may be typed : array
functions foo($arg) {

```

(continues on next page)

(continued from previous page)

```
    // the returned value is always an array, so this function might be typed as :␣
↪array
    return array($arg[3]);
}

?>
```

See also Type declarations.

Suggestions

- Add *array* typehint to the code.

Specs

Short name	Typehints/CouldBeArray
Rulesets	none
Exakt since	2.1.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1312 Could Be Boolean

Mark arguments and return types that can be set to boolean.

```
<?php

// Accept a boolean as input
function foo($b) {
    // Returns a boolean
    return $b === true;
}

?>
```

Suggestions

- Add *bool* typehint to the code.

Specs

Short name	Typehints/CouldBeBoolean
Rulesets	none
Exakt since	2.1.2
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1313 Could Be Callable

Mark arguments and return types that can be set to `callable`.

The analysis also reports properties that could be 'callable', although PHP doesn't allow that configuration.

```
<?php

// Accept a callable as input
function foo($b) {
    // Returns value as return
    return $b();
}

?>
```

See also Callbacks / callables.

Suggestions

- Add *callable* typehint to arguments or returntypes.

Specs

Short name	Typehints/CouldBeCallable
Rulesets	*Typechecks*
Exakt since	2.1.2
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1314 Could Be CIT

Mark arguments and return types that can be set to a class, interface definition.

```
<?php

// Accept an object as input
```

(continues on next page)

(continued from previous page)

```
function foo($b) {
    // Returns new object
    return new ($b->classname);
}

?>
```

Suggestions

- Add the class or interface typehint to the code.

Specs

Short name	Typehints/CouldBeCIT
Rulesets	none
Exakt since	2.1.2
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1315 Could Be Float

Mark arguments, properties and return types that can be set to `float`.

```
<?php

// Accept an int as input
function foo($b) {
    // Returns a float (cubic root of $b);
    return pow($b, 1 / 3);
}

?>
```

Suggestions

- Add *float* typehint to the code.

Specs

Short name	Typehints/CouldBeFloat
Rulesets	*Typechecks*
Exakt since	2.1.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1316 Could Be Generator

Return value may be typed `generator`.

```
<?php

// Yield makes foo() a generator
function foo() {
    yield 1;
    // Returns an int
    return $b + 8;
}

?>
```

Suggestions

- Add *Generator* typehint to the method.

Specs

Short name	Typehints/CouldBeGenerator
Rulesets	none
Exakt since	2.2.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1317 Could Be Integer

Mark arguments, properties and return types that can be set to `int`.

```
<?php

// Accept an int as input
function foo($b) {
    // Returns an int
    return $b + 8;
}

?>
```

Suggestions

- Add *int* typehint to the code.

Specs

Short name	Typehints/CouldBeInt
Rulesets	*Typechecks*
Exakt since	2.1.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1318 Could Be Iterable

Mark arguments, properties and return types that can be set to `iterable`.

```
<?php

// Accept an array or a traversable Object as input
function foo($b) {
    foreach($b as $c) {

    }

    // Returns an array
    return [$b];
}

?>
```

Suggestions

- Add *iterable* typehint to the code (PHP 8.0+).

Specs

Short name	Typehints/CouldBeIterable
Rulesets	*Typechecks*
Exakt since	2.1.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1319 Could Be Null

Mark arguments and return types that can be null.

```
<?php

// Accept null as input, when used as third argument of file_get_contents
```

(continues on next page)

(continued from previous page)

```
function foo($b) {
    $s = file_get_contents(URL, false, $b);

    // Returns a string
    return shell_exec($s);
}

?>
```

Suggestions

- Add *null* typehint to the code (PHP 8.0+).
- Add *?* typehint to the code.

Specs

Short name	Typehints/CouldBeNull
Rulesets	*Typechecks*
Exakt since	2.1.2
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1320 Could Be Parent

Mark arguments, return types and properties that can be set to `parent`.

This analysis works when typehints have already been configured.

```
<?php

class x extends w {
    // Accept a w object as input
    function foo(w $b) : w {
        // Returns a w object
        return $b;
    }
}

?>
```

Suggestions

- Add *parent* typehint to the code.
- Add the literal class/type typehint to the code.

Specs

Short name	Typehints/CouldBeParent
Rulesets	*Typechecks*
Exakt since	2.1.2
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1321 Could Be Self

Mark arguments, return types and properties that can be set to `self`. This applies only to methods.

This analysis works when typehints have already been configured.

```
<?php

class x {
    // Accept a x object as input
    function foo(x $b) : x {
        // Returns a x object
        return $b;
    }
}

?>
```

Suggestions

- Add *self* typehint to the code.
- Add the literal class/type typehint to the code.

Specs

Short name	Typehints/CouldBeSelf
Rulesets	*Typechecks*
Exakt since	2.1.2
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1322 Could Be String

Mark arguments and return types that can be set to string.

```php
<?php

// Accept a string as input
function foo($a) {
    // Returns a string
    return $a . 'string';
}

?>
```

Suggestions

- Choose the string typehint, and add it to the code.

Specs

Short name	Typehints/CouldBeString
Rulesets	*Typechecks*
Exakt since	2.1.2
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1323 Could Be Void

Mark return types that can be set to void.

```php
<?php

// No return, this should be void.
function foo() {
    ++$a; // Not useful
}

?>
```

All abstract methods (in classes or in interfaces) are omitted here.

Suggestions

- Add the void typehint to the code.

Specs

Short name	Typehints/CouldBeVoid
Rulesets	*Typechecks*
Exakt since	2.1.2
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1324 Could Not Type

Mark arguments, return types and properties that could not be typed.

Arguments, return types and properties that have no explicit typehint, and that could yield no guess from the following analysis, are deemed unable to receive a type.

- Typehints/CouldBeCIT
- Typehints/CouldBeString
- Typehints/CouldBeArray
- Typehints/CouldBeBoolean
- Typehints/CouldBeVoid
- Typehints/CouldBeCallable

`mixed` typehint, which acts as the universal typehint, is not processed here.

There are situation which cannot be typed, and legit : the example below is an illustration. `array_fill` is a native PHP example, where the second argument may be of any type. `__get``and ``__set` are also notoriously difficult to type, given the broad usage of arguments.

```
<?php

// Accepts any input, and returns any input
// This may be used, but not typed.
function foo($b) {
    return $b;
}

?>
```

Specs

Short name	Typehints/CouldNotType
Rulesets	none
Exakt since	2.1.2
Php Version	7.4-
Severity	Major
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.1325 Missing Some Returntype

The specified typehints are not compatible with the returned values.

The code of the method may return other types, which are not specified and will lead to a PHP fatal error. It is the case for insufficient typehints, when a typehint is missing, or inconsistent typehints, when the method returns varied types.

```
<?php

// correct return typehint
function fooSN() : ?string  {
    return shell_exec('ls -hla');
}

// insufficient return typehint
// shell_exec() may return null or string. Here, only string in specified for fooS,␣
↪and that may lead to a Fatal error
function fooS() : string  {
    return shell_exec('ls -hla');
}

// inconsistent return typehint
function bar() : int {
    return rand(0, 10) ? 1 : b;
}

?>
```

The analysis reports a method when it finds other return types than the one expected. In the case of multiple typehints, as for the last example, the PHP code may require an upgrade to PHP 8.0.

Suggestions

- Update the typehint to accept more types
- Update the code of the method to fit the expected returntype

Specs

Short name	Typehints/MissingReturntype
Rulesets	*Analyze*, *CI-checks*
Exakt since	2.1.7
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.1326 Selector

Makes a selection of atoms in the code, based on the selector parameter.

Name	Default	Type	Description
selector		string	A selector expression to identify atoms in the code.

Specs

Short name	Utils/Selector
Rulesets	none
Exakt since	2.2.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Unknown

13.2.1327 Assigned Twice

The same variable is assigned twice in the same function.

While this is possible and quite common, it is also a good practice to avoid changing a value from one literal to another. It is far better to assign the new value to

Incremental changes to a variables are not reported here.

```
<?php

function foo() {
    // incremental changes of $a;
    $a = 'a';
    $a++;
    $a = uppercase($a);

    $b = 1;
    $c = bar($b);
    // B changed its purpose. Why not call it $d?
    $b = array(1,2,3);

    // This is some forgotten debug
    $e = $config->getSomeList();
    $e = array('OneElement');
}

?>
```

Specs

Short name	Variables/AssignedTwiceOrMore
Rulesets	*Analyze*
Exakt since	0.9.8
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1328 Blind Variables

Variables that are used in foreach or for structure, for their managing the loop itself.

```
<?php
    foreach($array as $key => $value) {
        // $key and $value are blind values
    }

?>
```

Specs

Short name	Variables/Blind
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1329 Confusing Names

The following variables's name are very close and may lead to confusion.

Variables are 3 letters long (at least). Variables names build with an extra `s` are omitted. Variables may be scattered across the code, or close to each other.

Variables which differ only by case, or by punctuation or by numbers are reported here.

```
<?php

    // Variable names with one letter difference
    $fWScale = 1;
    $fHScale = 1;
    $fScale = 2;

    $oFrame = 3;
    $iFrame = new Foo();

    $v2_norm = array();
    $v1_norm = 'string';

    $exept11 = 1;
    $exept10 = 2;
    $exept8 = 3;

    // Variables that differ by punctation
    $locale = 'fr';
    $_locate = 'en';

    // Variables that differ by numbers
    $x11 = 'a';
    $x12 = 'b';
```

(continues on next page)

(continued from previous page)

```
    // Variables that differ by numbers
    $songMP3 = 'a';
    $Songmp3 = 'b';

    // This even looks like a typo
    $privileges  = 1;
    $privilieges = true;

    // This is not reported : Adding extra s is tolerated.
    $rows[] = $row;

?>
```

See also How to pick bad function and variable names.

Specs

Short name	Variables/CloseNaming
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1330 Complex Dynamic Names

Avoid using expressions as names for variables or methods.

There are no place for checks or flow control, leading to any rogue value to be used as is. Besides, the expression is often overlooked, and not expected there : this makes the code less readable.

It is recommended to build the name in a separate variable, apply the usual checks for existence and validity, and then use the name.

```
<?php

$a = new foo();

// Code is more readable
$name = strolower($string);
if (!property_exists($a, $name)) {
    throw new missingPropertyexception($name);
}
echo $a->$name;

// This is not check
echo $a->{strtolower($string)};

?>
```

This analysis only accept simple containers, such as variables, properties.

See also Dynamically Access PHP Object Properties with '$this <https://drupalize.me/blog/201508/dynamically-access-php-object-properties>'_.

Suggestions

- Extract the expression from the variable syntax, and make it a separate variable.

Specs

Short name	Variables/ComplexDynamicNames
Rulesets	*Suggestions*
Exakt since	1.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1331 Constant Typo Looks Like A Variable

A constant bears the same name as a variable. This might be a typo.

When the constant doesn't exist, PHP 8.0 yields a Fatal Error and stops execution. PHP 7.4 turns the undefined constant into its string equivalent.

```
<?php

// Get an object or null
$object = foo();

// PHP 8.0 stops here, with a Fatal Error
// PHP 7.4 makes this a string, and the condition is always true
if (!empty(object)) {
    // In PHP 7.4, this is not protected by the condition, and may yield an error.
    $object->doSomething();
}

?>
```

This analysis is case sensitive.

Suggestions

- Add a $ sign to the constant
- Use a different name for the variable, or the constant

Specs

Short name	Variables/ConstantTypo
Rulesets	*Analyze*
Exakt since	2.2.0
Php Version	8.0-
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Medium

13.2.1332 Globals

Global variables.

```
<?php

// global via global keyword
global $a, $b;

// global via $GLOBALS variable
$GLOBALS['c'] = 1;

?>
```

Specs

Short name	Variables/Globals
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1333 Inconsistent Variable Usage

Those variables are used in various and inconsistent ways. It is difficult to understand if they are an array, an object or a scalar variable.

```
<?php

// $a is an array, then $b is a string.
$a = ['a', 'b', 'c'];
$b = implode('-', $a);

// $a is an array, then it is a string.
$a = ['a', 'b', 'c'];
$a = implode('-', $a);

?>
```

Suggestions

- Keep one type for each variable. This keeps the code readable.
- Give different names to variables with different types.

Specs

Short name	Variables/InconsistentUsage
Rulesets	none
Exakt since	1.6.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*WordPress*

13.2.1334 Interface Arguments

Variables that are arguments in an interface.

```
<?php

interface i {
    function interfaceMethod($interfaceArgument) ;
}

class foo extends i {
    // Save function as above, but the variable is not reported
    function interfaceMethod($notAnInterfaceArgument) {}
}

?>
```

Specs

Short name	Variables/InterfaceArguments
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1335 Local Globals

A global variable is used locally in a method.

Either the global keyword has been forgotten, or the local variable should be renamed in a less ambiguous manner.

Having both a global and a local variable with the same name is legit. PHP keeps the contexts separated, and it processes them independently.

However, in the mind of the coder, it is easy to mistake the local variable $x and the global variable $x. May they be given different meaning, and this is an error-prone situation.

It is recommended to keep the global variables's name distinct from the local variables.

```
<?php

// This is actualy a global variable
$variable = 1;
$globalVariable = 2;

function foo() {
    global $globalVariable2;

    $variable = 4;
    $localVariable = 3;

    // This always displays 423, instead of 123
    echo $variable .' ' . $globalVariable . ' ' . $localVariable;
}

?>
```

Specs

Short name	Variables/LocalGlobals
Rulesets	none
Exakt since	1.1.2
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1336 Lost References

Either avoid references, or propagate them correctly.

When assigning a referenced variable with another reference, the initial reference is lost, while the intend was to transfer the content.

```
<?php

function foo(&$lostReference, &$keptReference)
{
    $c = 'c';

    // $lostReference was a reference, but now, it is another
    $lostReference =& $c;
    // $keptReference was a reference : now it contains the actual value
```

(continues on next page)

(continued from previous page)

```
    $keptReference = $c;
}

$bar = 'bar';
$bar2 = 'bar';
foo($bar, $bar2);

//displays bar c, instead of bar bar
print $bar. ' '.$bar2;

?>
```

Do not reassign a reference with another reference. Assign new content to the reference to change its value.

Suggestions

- Always assign new value to an referenced argument, and don't reassign a new reference

Specs

Short name	Variables/LostReferences
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High
Examples	*WordPress*

13.2.1337 No Static Variable In A Method

Refactor static variables into (static) properties.

Inside a class, it is recommended to use the class properties, static or not, to hold values between calls to the method. Inside a function, or a closure, no such container is available, so static variables may be useful. Although, a refactoring to a class is also recommended here.

Properties have clear definitions, and are less suprising than static variables.

```
<?php

class barbar {
    function foo() {
        static $counter = 0;

        // count the number of calls of this method
        return ++$counter;
    }
}

class bar {
    static $counter = 0;
```

(continues on next page)

(continued from previous page)

```
    function foo() {
        // count the number of calls of this method
        return ++self::$counter;
    }
}

?>
```

The static variable is easier to refactor as a static property. It is also possible to refactor it as a property, although it may impact the behavior of the previous code, or require extra work.

Suggestions

- Refactor the variable into a static property
- Refactor the variable into a property and dependency injection

Specs

Short name	Variables/NoStaticVarInMethod
Rulesets	*ClassReview*, *Suggestions*
Exakt since	2.2.1
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	Very high

13.2.1338 Overwriting Variable

Replacing the content of a variable by something different is prone to errors. For example, it is not obvious if the $text variable is plain text or HTML text.

```
<?php

// Confusing
$text = htmlentities($text);

// Better
$textHTML = htmlentities($text);

?>
```

Besides, it is possible that the source is needed later, for extra processing.

Note that accumulators, like += .= or [] etc., that are meant to collect lots of values with consistent type are OK.

Specs

Short name	Variables/Overwriting
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Quick (30 mins)
Precision	High

13.2.1339 Overwritten Literals

The same variable is assigned a literal twice. It is possible that one of the assignation is too much.

This analysis doesn't take into account the distance between two assignations : it may report false positives when the variable is actually used for several purposes, and, as such, assigned twice with different values.

```
<?php

function foo() {
    // Two assignations in a short sequence : one is too many.
    $a = 1;
    $a = 2;

    for($i = 0; $i < 10; $i++) {
        $a += $i;
    }
    $b = $a;

    // New assignation. $a is now used as an array.
    $a = array(0);
}

?>
```

Specs

Short name	Variables/OverwrittenLiterals
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Major
Time To Fix	Instant (5 mins)
Precision	High

13.2.1340 PHP5 Indirect Variable Expression

Indirect variable expressions changes between PHP 5 an 7.

The following structures are evaluated differently in PHP 5 and 7. It is recommended to review them or switch to a less ambiguous syntax.

```
<?php

// PHP 7
$foo = 'bar';
$bar['bar']['baz'] = 'foobarbarbaz';
echo $$foo['bar']['baz'];
echo ($$foo)['bar']['baz'];

// PHP 5
$foo['bar']['baz'] = 'bar';
$bar = 'foobarbazbar';
echo $$foo['bar']['baz'];
echo ${$foo['bar']['baz']};

?>
```

See Backward incompatible changes PHP 7.0

Suggestions

- Avoid using complex expressions, mixing `$$\`, `[0]` and `->` in the same expression
- Add curly braces {} to ensure that the precedence is the same between PHP 5 and 7. For example, `$$v` becomes `${$v}`

Specs

Short name	Variables/Php5IndirectExpression
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*
Exakt since	0.8.4
Php Version	7.0-
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.1341 Php 7 Indirect Expression

Those are variable indirect expressions that are interpreted differently in PHP 5 and PHP 7.

You should check them so they don't behave strangely.

```
<?php

// Ambiguous expression :
$b = $$foo['bar']['baz'];
echo $b;

$foo = array('bar' => array('baz' => 'bat'));
$bat = 'PHP 5.6';

// In PHP 5, the expression above means :
$b = $\{$foo['bar']['baz']};
```

(continues on next page)

(continued from previous page)

```
$b = 'PHP 5.6';

$foo = 'a';
$a = array('bar' => array('baz' => 'bat'));

// In PHP 7, the expression above means :
$b = ($$foo)['bar']['baz'];
$b = 'bat';

?>
```

See also Changes to variable handling.

Suggestions

- Avoid using complex expressions, mixing $$, [0] and -> in the same expression
- Add curly braces {} to ensure that the precedence is the same between PHP 5 and 7. For example, `$$v` becomes `${$v}`

Specs

Short name	Variables/Php7IndirectExpression
Rulesets	*CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*
Exakt since	0.8.4
Php Version	7.0+
Severity	Major
Time To Fix	Slow (1 hour)
Precision	High

13.2.1342 Real Variables

Inventory of real variables. Global, Static and property declarations are skipped here.

```
<?php

$realVariable = 1;

class foo {
    private $property;         // not a variable

    private function bar() {
        global $global;        // not a variable
        static $static;        // not a variable

    }
```

(continues on next page)

(continued from previous page)

```
}

?>
```

This is a refined version of a search on `T_VARIABLE` token.

Specs

Short name	Variables/RealVariables
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1343 Variable References

Variables that are references.

```
<?php

$a = '1'; // not a reference
$b = &$a; // a reference

?>
```

See also References.

Specs

Short name	Variables/References
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1344 Self-Transforming Variables

Variables that are assigned to themselves after transformation.

```
<?php

$s = strtolower($s);

// filtering one element AND dropping all that not 1
```

(continues on next page)

(continued from previous page)

```
$a = array_filter('foo', $a[1]);

$o->m = foo($o->m);

?>
```

Suggestions

- Try to use new variables to hold new values.

Specs

Short name	Variables/SelfTransform
Rulesets	none
Exakt since	1.7.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1345 Static Variables

In PHP, variables may be static. They will survive after the function execution end, and will be available at the next function run. They are distinct from globals, which are available application wide, and from static properties, which are tied to a class of objects.

```
<?php

function foo() {
    // static variable
    static $count = 0;

    echo ++$count;
}

class bar {
    // This is not a static variable :
    // it is a static property
    static $property = 1;
}

?>
```

Specs

Short name	Variables/StaticVariables
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1346 Strange Name For Variables

Variables with strange names. They might be a typo, or bear strange patterns.

Any variable with three identical letter in a row are considered as strange. 2 letters in a row is classic, and while three letters may happen, it is rare enough.

A list of classic typo is also used to find such variables.

This analysis is case-sensitive.

```
<?php

class foo {
    function bar() {
        // Strange name $tihs
        return $tihs;
    }

    function barbar() {
        // variables with blocks of 3 times the same character are reported
        // Based on Alexandre Joly's tweet
        $aaa = $bab + $www;
    }
}

?>
```

See also #QuandLeDevALaFleme.

Suggestions

- Fix the name of the variable
- Rename the variable to something better
- Drop the variable

Specs

Short name	Variables/StrangeName
Rulesets	*Analyze*
Exakt since	0.10.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*FuelCMS*, *PhpIPAM*

13.2.1347 Environment Variables

Environment variables are used to interact with the hosting system.

They often provides configuration parameter that are set by the host of the application to be used. That way, information is not hardcoded in the application, and may be changed at production.

```
<?php

//ENVIRONMENT set the production context
if (getenv('ENVIRONMENT') === 'Production') {
    $sshKey = getenv('HOST_KEY');
} elseif (getenv('ENVIRONMENT') === 'Developper') {
    $sshKey = 'NO KEY';
} else {
    header('No website here.');
    die();
}

?>
```

See also $_ENV.

Specs

Short name	Variables/UncommonEnvVar
Rulesets	*CE*
Exakt since	1.0.5
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Medium

13.2.1348 Undefined Constant Name

When using the `` syntax for variable, the name used must be a defined constant. It is not a simple string, like 'x', it is an actual constant name.

Interestingly, it is possible to use a qualified name within ‘‘, full or partial. PHP will lint such code, and will collect the value of the constant immediately. Since there is no fallback mechanism for fully qualified names, this ends with a Fatal error.

```
<?php

const x = a;
$a = Hello;

// Display 'Hello'  -> $a -> Hello
echo ;

// Yield a PHP Warning
// Use of undefined constant y - assumed 'y' (this will throw an Error in a future␣
↪version of PHP)
echo ;

// Yield a PHP Fatal error as PHP first checks that the constant exists
//Undefined constant 'y'
echo ;
?>
```

Suggestions

- Define the constant
- Turn the dynamic syntax into a normal variable syntax
- Use a fully qualified name (at least one) to turn this syntax into a Fatal error when the constant is not found. This doesn't fix the problem, but may make it more obvious during the diagnostic.

Specs

Short name	Variables/UndefinedConstantName
Rulesets	*Analyze*
Exakt since	2.1.1
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1349 Undefined Variable

Variable that is used before any creation.

It is recommended to use a default value for every variable used. When not specified, the default value is set to `NULL` by PHP.

```
<?php

// Adapted from the PHP manual
$var = 'Bob';
$Var = 'Joe';
```

(continues on next page)

(continued from previous page)

```
// The following line may emit a warning : Undefined variable: $undefined
echo $var, $Var, $undefined;       // outputs Bob, Joe,


?>
```

Variable may be created in various ways : assignation, arguments, foreach blind variables, static and global variables.

This analysis doesn't handle dynamic variables, such as `$$x`. It also doesn't handle variables outside a method or function.

See also Variable basics.

Suggestions

- Remove the expression that is using the undefined variable
- Fix the variable name
- Define the variable by assigning a value to it, before using it

Specs

Short name	Variables/UndefinedVariable
Rulesets	*Analyze*, *CI-checks*
Exakt since	1.4.2
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1350 Single Use Variables

Variables that are written, then read. Only used once.

Single-use variables may be trimmed down, and the initial expression may be used instead.

Single-use variables may improve readability, when the final expression grows too much with the extra expression.

```
<?php

function foo($d) {
    $a = 1;      // $a is used twice
    $b = $a + 2;  // $b is used once
    $c = $a + $b + $d; // $c is also used once
    // $d is an argument, so it's OK.

    retrun $c;
}

?>
```

Suggestions

- Merge the two expressions into one larger
- Make a second use of the variable

Specs

Short name	Variables/UniqueUsage
Rulesets	none
Exakt since	1.3.0
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High

13.2.1351 Variables With Long Names

VariablesLong collect all variables with more than 20 characters longs.

```
<?php

// Quite a long variable name
$There_is nothing_wrong_with_long_variable_names_They_tend_to_be_rare_and_that_make_
↪them_noteworthy = 1;

?>
```

There is nothing wrong with long variable names. They tend to be rare, and that make them noteworthy.

See also Basics.

Name	Default	Type	Description
variableLength	20	integer	Minimum size of a long variable name, including the initial $.

Specs

Short name	Variables/VariableLong
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1352 Non Ascii Variables

PHP allows certain characters in variable names. The variable name must only include letters, figures, underscores and ASCII characters from 128 to 255.

In practice, letters outside the scope of `a-zA-Z0-9` are rare, and require more care when editing the code or passing it from OS to OS.

```
<?php

class  {
    // An actual working class in PHP.
    public function __construct() {
        echo __CLASS__;
    }
}

$ = new ();

?>
```

See also Variables.

Suggestions

- Make sure those special chars have actual meaning.

Specs

Short name	Variables/VariableNonascii
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
Examples	*Magento*

13.2.1353 Variables With One Letter Names

> Variables with one letter name are the shortest name for variables. They also bear very little meaning : what does contain the variable `$w` ?

Some one-letter variables have meaning : `$x` and `$y` for coordinates, `$i`, `$j`, `$k` for blind variables. Others tend to be an easy way to give a name to a variable, without thinking too hard a good name.

```
<?php

// $a is reported as a one-letter variable
$a = 0;

// $i is considered a false positive.
for($i = 0; $i < 10; ++$i) {
    $a += doSomething($i);
}

?>
```

See also Using single characters for variable names in loops/exceptions and Single Letter Variable Names Still Considered Harmful.

Suggestions

- Make the variable more meaningful, with full words

Specs

Short name	Variables/VariableOneLetter
Rulesets	*Semantics*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	Very high

13.2.1354 PHP Variables

This is the list of PHP predefined variables that are used in the application.

```
<?php

// Reading an incoming email, with sanitation
$email = filter_var($_GET['email'], FILTER_SANITIZE_EMAIL);

?>
```

The web variables (`$\_GET`, `$\_COOKIE`, `$\_FILES`) are quite commonly used, though sometimes replaced by some special accessors. Others are rarely used.

Specs

Short name	Variables/VariablePhp
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1355 All Uppercase Variables

Usually, global variables are all in uppercase, so as to differentiate them easily. Though, this is not always the case, with examples like $argc, $argv or $http_response_header.

When using custom variables, try to use lowercase `$variables`, `$camelCase`, `$sturdyCase` or `$snake_case`.

```
<?php

// PHP super global, also identified by the initial _
$localVariable = $_POST;

// PHP globals
$localVariable = $GLOBALS['HTTPS'];

?>
```

See also Predefined Variables.

Specs

Short name	Variables/VariableUppercase
Rulesets	none
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1356 Used Once Variables

This is the list of used once variables.

```
<?php

// The variables below never appear again in the code
$writtenOnce = 1;

foo($readOnce);

?>
```

Such variables are useless. Variables must be used at least twice : once for writing, once for reading, at least. It is recommended to remove them.

In special situations, variables may be used once :

- PHP predefined variables, as they are already initialized. They are omitted in this analyze.
- Interface function's arguments, since the function has no body; They are omitted in this analyze.
- Dynamically created variables ($$x, ${$this->y} or also using extract), as they are runtime values and can't be determined at static code time. They are reported for manual review.
- Dynamically included files will provide in-scope extra variables.

This rule counts variables at the application level, and not at a method scope level.

Suggestions

- Remove the variable

- Fix the name of variable
- Use the variable a second time, at least

Specs

Short name	Variables/VariableUsedOnce
Rulesets	*Analyze*, *Top10*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Instant (5 mins)
Precision	High
Examples	*shopware*, *Vanilla*

13.2.1357 Used Once Variables (In Scope)

This is the list of used once variables, scope by scope. Those variables are used once in a function, a method, a class or a namespace. In any case, this means the variable is read or written, while it should be used at least twice.

```
<?php

function foo() {
    // The variables below never appear twice, inside foo()
    $writtenOnce = 1;

    foo($readOnce);
    // They do appear again in other functions, or in global space.
}

function bar() {
    $writtenOnce = 1;
    foo($readOnce);
}

?>
```

Suggestions

- Remove the variable
- Fix the name of variable
- Use the variable a second time in the current scope, at least

Specs

Short name	Variables/VariableUsedOnceByContext
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Quick (30 mins)
Precision	High
Examples	*shopware*

13.2.1358 Variable Variables

A variable variable takes the value of a variable and treats that as the name of a variable.

PHP has the ability to dynamically use a variable.

```
<?php

// Normal variable
$a = 'b';
$b = 'c';

// Variable variable
$d = $$b;

// Variable variable in string
$d = "$\{$b\}";

?>
```

They are also called 'dynamic variable'.

See also Variable variables.

Specs

Short name	Variables/VariableVariables
Rulesets	*CE*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1359 Written Only Variables

Those variables are being written, but never read. This way, they are useless and should be removed, or read at some point.

```php
<?php

// $a is used multiple times, but never read
$a = 'a';
$a .= 'b';

$b = 3;
//$b is actually read once
$a .= $b + 3;

?>
```

Suggestions

- Check that variables are written AND read in each context
- Remove variables that are only read
- Use the variable that are only read

Specs

Short name	Variables/WrittenOnlyVariable
Rulesets	*Analyze*
Exakt since	0.8.4
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High
ClearPHP	no-unused-variable
Examples	*Dolibarr*, *SuiteCrm*

13.2.1360 Codeigniter usage

This analysis reports usage of the Codeigniter framework.

```php
<?php

// A code igniter controller
class Blog extends CI_Controller {

        public function index()
        {
                echo 'Hello World!';
        }
}

?>
```

See also Codeigniter.

Specs

Short name	Vendors/Codeigniter
Rulesets	*CE*
Exakt since	0.11.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1361 Concrete usage

This analysis reports usage of the Concrete 5 framework.

```
<?php
namespace Application\Controller\PageType;

use Concrete\Core\Page\Controller\PageTypeController;

class BlogEntry extends PageTypeController
{

    public function view()
    {
    }
}
?>
```

See also Concrete 5.

Specs

Short name	Vendors/Concrete5
Rulesets	*CE*
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1362 Drupal Usage

This analysis reports usage of the Drupal CMS. The report is based on the usage of Drupal namespace.

```
<?php

namespace Drupal\example\Controller;

use Drupal\Core\Controller\ControllerBase;

```

(continues on next page)

(continued from previous page)

```
/**
 * An example controller.
 */
class ExampleController extends ControllerBase {

  /**
   * {@inheritdoc}
   */
  public function content() {
    $build = array(
      '#type' => 'markup',
      '#markup' => t('Hello World!'),
    );
    return $build;
  }

}

?>
```

See also Drupal.

Specs

Short name	Vendors/Drupal
Rulesets	*CE*
Exakt since	1.0.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1363 Ez cms usage

This analysis reports usage of the Ez cms.

```
<?php
namespace My\Bundle\With\Controller;

use eZ\Bundle\EzPublishCoreBundle\Controller;
use Symfony\Component\HttpFoundation\Request;

class DemoController extends Controller {
    public function demoCreateContentAction(Request $request) {
        //
    }
}

?>
```

See also Ez.

Specs

Short name	Vendors/Ez
Rulesets	*CE*
Exakt since	0.11.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1364 FuelPHP Usage

This analysis reports usage of the Fuel PHP Framework. The report is based on the usage of Fuel namespace.

```
<?php
// file located in APPPATH/classes/presenter.php
class Presenter extends \Fuel\Core\Presenter
{
    // namespace prefix
    protected static $ns_prefix = 'Presenter\';
}
?>
```

See also FuelPHP.

Specs

Short name	Vendors/Fuel
Rulesets	*CE*
Exakt since	1.0.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1365 Joomla usage

This analysis reports usage of the Joomla CMS.

```
<?php

// no direct access
defined('_JEXEC') or die('Restricted access');

jimport('joomla.application.component.controller');
JLoader::import('KBIntegrator', JPATH_PLUGINS . DS . 'kbi');

class MyController extends JController {
     function display($message) {
```

(continues on next page)

(continued from previous page)

```
        echo $message;
    }
}

?>
```

See also Joomla.

Specs

Short name	Vendors/Joomla
Rulesets	*CE*
Exakt since	0.11.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1366 Laravel usage

This analysis reports usage of the Laravel framework.

```
<?php

namespace App\Http\Controllers;

use App\User;
use App\Http\Controllers\Controller;

class UserController extends Controller
{
    /**
     * Show the profile for the given user.
     *
     * @param  int  $id
     * @return Response
     */
    public function show($id)
    {
        return view('user.profile', ['user' => User::findOrFail($id)]);
    }
}

?>
```

See also Laravel.

Specs

Short name	Vendors/Laravel
Rulesets	*CE*
Exakt since	0.11.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1367 Phalcon Usage

This analysis reports usage of the Phalcon Framework. The report is based on the usage of Phalcon namespace, which may be provided by PHP code inclusion or the PHP extension.

```
<?php

use Phalcon\Mvc\Application;

// Register autoloaders

// Register services

// Handle the request
$application = new Application($di);

try {
    $response = $application->handle();

    $response->send();
} catch (\Exception $e) {
    echo 'Exception: ', $e->getMessage();
}

?>
```

See also Phalcon.

Specs

Short name	Vendors/Phalcon
Rulesets	*CE*
Exakt since	1.0.3
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1368 Symfony usage

This analysis reports usage of the Symfony framework.

```php
<?php

// src/AppBundle/Controller/LuckyController.php
namespace AppBundle\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Component\HttpFoundation\Response;

class LuckyController
{
    /**
     * @Route(/lucky/number)
     */
    public function numberAction()
    {
        $number = mt_rand(0, 100);

        return new Response(
            '<html><body>Lucky number: '.$number.'</body></html>'
        );
    }
}

?>
```

See also Symfony.

Specs

Short name	Vendors/Symfony
Rulesets	*CE*
Exakt since	0.11.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1369 Typo 3 usage

This analysis reports usage of the Typo 3 CMS.

```php
<?php

namespace MyVendor\SjrOffers\Controller;

use TYPO3\CMS\Extbase\Mvc\Controller\ActionController;

class OfferController extends ActionController
{
   // Action methods will be following here
}
?>
```

See also Typo3.

Specs

Short name	Vendors/Typo3
Rulesets	*CE*
Exakt since	1.9.9
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1370 Wordpress usage

This analysis reports usage of the Wordpress platform.

```
<?php

//Usage of the WP_http class from Wordpress
$rags = array(
   'x' => '1',
   'y' => '2'
);
$url = 'http://www.example.com/';
$request = new WP_Http();
$result = $request->request( $url, array( 'method' => 'POST', 'body' => $body) );

?>
```

See also Wordpress.

Specs

Short name	Vendors/Wordpress
Rulesets	*CE*
Exakt since	0.11.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.2.1371 Yii usage

This analysis reports usage of the Yii framework.

```
<?php

// A Yii controller
class SiteController extends CController
{
    public function actionIndex()
    {
```

(continues on next page)

(continued from previous page)

```
        // ...
    }

    public function actionContact()
    {
        // ...
    }
}

?>
```

See also Yii.

Specs

Short name	Vendors/Yii
Rulesets	*CE*
Exakt since	0.11.8
Php Version	All
Severity	Minor
Time To Fix	Slow (1 hour)
Precision	High

13.3 Directory by Exakat version

List of analyzers, by version of introduction, newest to oldest. In parenthesis, the first element is the analyzer name, used with 'analyze -P' command, and the seconds, if any, are the ruleset, used with the -T option. Rulesets are separated by commas, as the same analysis may be used in several rulesets.

- 2.2.2
  - *Cannot Use Static For Closure*
  - *Could Be Generator*
  - *Could Use Match*
  - *Multiple Property Declaration On One Line*
  - *Selector*
- 2.2.1
  - *Avoid get_object_vars()*
  - *Declare Static Once*
  - *No Static Variable In A Method*
  - *Reserved Match Keyword*
- 2.2.0
  - *Array_Map() Passes By Value*
  - *Cancelled Parameter*

  - *Collect Block Size*
  - *Constant Typo Looks Like A Variable*
  - *Final Private Methods*
  - *Long Preparation For Throw*
  - *Missing __isset() Method*
  - *Modify Immutable*
  - *Only Container For Reference*
  - *PHP 80 Named Parameter Variadic*
  - *PHP Resources Turned Into Objects*
  - *Searching For Multiple Keys*
  - *Unused Exception Variable*
  - *Use str_contains()*
  - *Wrong Attribute Configuration*
- 2.1.9
  - *Array_Fill() With Objects*
  - *Assumptions*
  - *Collect Use Counts*
  - *Complete/PhpExtStubPropertyMethod*
  - *Could Be Stringable*
  - *Could Use Promoted Properties*
  - *Modified Typed Parameter*
  - *Negative Start Index In Array*
  - *Nullable With Constant*
  - *Optimize Explode()*
  - *PHP 8.0 Removed Directives*
  - *Unsupported Types With Operators*
  - *Use get_debug_type()*
  - *Useless Typehint*
- 2.1.8
  - *$php_errormsg Usage*
  - *Cancel Common Method*
  - *Cast Unset Usage*
  - *Collect Atom Counts*
  - *Collect Classes Dependencies*
  - *Collect Files Dependencies*
  - *Collect Php Structures*

- 2.0.2
  - *Dump/Inclusions*
  - *Insufficient Property Typehint*
  - *New Order*
  - *Nullable Without Check*
  - *Typehint Order*
  - *Wrong Typehinted Name*
- 1.9.9
  - *Collect Mbstring Encodings*
  - *Complete/CreateForeachDefault*
  - *Concrete usage*
  - *Could Type With Array*
  - *Could Type With Boolean*
  - *Could Type With Int*
  - *Could Type With Iterable*
  - *Could Type With String*
  - *Filter To add_slashes()*
  - *Immutable Signature*
  - *Is_A() With String*
  - *Mbstring Third Arg*
  - *Mbstring Unknown Encoding*
  - *Merge If Then*
  - *Shell commands*
  - *Typehinting Stats*
  - *Typo 3 usage*
  - *Weird Array Index*
  - *Wrong Case Namespaces*
  - *Wrong Type With Call*
- 1.9.8
  - *Cant Implement Traversable*
  - *Parameter Hiding*
  - *Propagate Calls*
- 1.9.7
  - *Foreach() Favorite*
  - *Make Functioncall With Reference*
  - *Too Many Dereferencing*

  - *Use Url Query Functions*
- 1.9.6
  - *Collect Parameter Counts*
  - *Dump/DereferencingLevels*
  - *Duplicate Literal*
  - *Internet Domains*
  - *No Weak SSL Crypto*
  - *No mb_substr In Loop*
  - *Non Nullable Getters*
  - *Use Case Value*
- 1.9.5
  - *Collect Literals*
  - *Environment Variable Usage*
  - *Interfaces Don't Ensure Properties*
  - *Interfaces Is Not Implemented*
  - *Magic Properties*
  - *No Literal For Reference*
  - *Use array_slice()*
- 1.9.4
  - *Coalesce And Concat*
  - *Comparison Is Always True*
  - *Cyclomatic Complexity*
  - *Nested Ternary Without Parenthesis*
  - *PHP 74 New Directives*
  - *Should Use Explode Args*
  - *Spread Operator For Array*
  - *Too Many Array Dimensions*
  - *Use Arrow Functions*
- 1.9.3
  - *Complete/SetClassRemoteDefinitionWithParenthesis*
  - *Complete/SetClassRemoteDefinitionWithTypehint*
  - *Indentation Levels*
  - *Max Level Of Nesting*
  - *No Spread For Hash*
  - *PHP 7.4 Constant Deprecation*
  - *PHP 7.4 Removed Directives*

  - *Numeric Literal Separator*
  - *PHP 7.4 Removed Functions*
  - *Reflection Export() Is Deprecated*
  - *Scalar Are Not Arrays*
  - *Serialize Magic Method*
  - *Similar Integers*
  - *Unbinding Closures*
  - *array_key_exists() Works On Arrays*
- 1.8.9
  - *Avoid mb_dectect_encoding()*
  - *Disconnected Classes*
  - *Not Or Tilde*
  - *Overwritten Source And Value*
  - *Useless Type Check*
  - *mb_strrpos() Third Argument*
- 1.8.8
  - *Set Aside Code*
  - *Use Array Functions*
- 1.8.7
  - *Cant Use Function*
  - *Generator Cannot Return*
  - *Use DateTimeImmutable Class*
  - *Wrong Type Returned*
- 1.8.6
  - *Dependant Abstract Classes*
  - *Infinite Recursion*
  - *Modules/IncomingData*
  - *Modules/NativeReplacement*
  - *Null Or Boolean Arrays*
- 1.8.5
  - *Could Use Trait*
- 1.8.4
  - *Always Use Function With array_key_exists()*
  - *Complex Dynamic Names*
  - *Could Be Constant*
  - *New Constants In PHP 7.4*

  - *Regex On Arrays*
  - *Unused Class Constant*
  - *curl_version() Has No Argument*
- 1.8.3
  - *Autoappend*
  - *Make Magic Concrete*
  - *Memoize MagicCall*
  - *Substr To Trim*
- 1.8.2
  - *Identical Methods*
  - *No Append On Source*
- 1.8.1
  - *No Need For get_class()*
- 1.8.0
  - *Already Parents Trait*
  - *Casting Ternary*
  - *Concat And Addition*
  - *Concat Empty String*
  - *Minus One On Error*
  - *New Functions In PHP 7.4*
  - *Unpacking Inside Arrays*
  - *Useless Argument*
- 1.7.9
  - *Avoid option arrays in constructors*
  - *Trait Not Found*
  - *Useless Default Argument*
  - *ext/ffi*
  - *ext/uuid*
  - *ext/zend_monitor*
- 1.7.8
  - *ext/svm*
- 1.7.7
  - *Implode One Arg*
  - *Incoming Values*
  - *Integer Conversion*
- 1.7.6

- 1.5.8
  - *Multiple Identical Closure*
  - *Path lists*
- 1.5.7
  - *Method Could Be Static*
  - *Multiple Usage Of Same Trait*
  - *Self Using Trait*
  - *ext/wasm*
- 1.5.6
  - *Isset() On The Whole Array*
  - *Useless Alias*
  - *ext/async*
  - *ext/sdl*
- 1.5.5
  - *Directly Use File*
  - *Safe HTTP Headers*
  - *fputcsv() In Loops*
- 1.5.4
  - *Avoid Self In Interface*
  - *Should Have Destructor*
  - *Unreachable Class Constant*
- 1.5.3
  - *Don't Loop On Yield*
  - *Variable May Be Non-Global*
- 1.5.2
  - *PHP Exception*
  - *Should Yield With Key*
  - *ext/decimal*
  - *ext/psr*
- 1.5.1
  - *Use Basename Suffix*
- 1.5.0
  - *Could Use Try*
  - *Pack Format Inventory*
  - *Printf Format Inventory*
  - *idn_to_ascii() New Default*

- 1.4.9
  - *Don't Read And Write In One Expression*
  - *Invalid Pack Format*
  - *Named Regex*
  - *No Reference For Static Property*
  - *No Return For Generator*
  - *Repeated Interface*
  - *Wrong Access Style to Property*
- 1.4.8
  - *Direct Call To __clone()*
  - *filter_input() As A Source*
- 1.4.6
  - *Only Variable For Reference*
- 1.4.5
  - *Add Default Value*
- 1.4.4
  - *ext/seaslog*
- 1.4.3
  - *Class Could Be Final*
  - *Closure Could Be A Callback*
  - *Inconsistent Elseif*
  - *Use json_decode() Options*
- 1.4.2
  - *Method Collision Traits*
  - *Undefined Insteadof*
  - *Undefined Variable*
- 1.4.1
  - *Must Call Parent Constructor*
- 1.4.0
  - *PHP 7.3 Removed Functions*
  - *Trailing Comma In Calls*
- 1.3.9
  - *Assert Function Is Reserved*
  - *Avoid Real*
  - *Case Insensitive Constants*
  - *Const Or Define Preference*

- 1.2.1
  - *Possible Increment*
  - *Properties Declaration Consistence*
- 1.1.10
  - *Too Many Native Calls*
- 1.1.9
  - *Should Preprocess Chr()*
  - *Too Many Parameters*
- 1.1.8
  - *Mass Creation Of Arrays*
  - *ext/db2*
- 1.1.7
  - *Could Use array_fill_keys*
  - *Dynamic Library Loading*
  - *PHP 7.3 Last Empty Argument*
  - *Property Could Be Local*
  - *Use Count Recursive*
  - *ext/leveldb*
  - *ext/opencensus*
  - *ext/uopz*
  - *ext/varnish*
  - *ext/xxtea*
- 1.1.6
  - *Could Use Compact*
  - *Foreach On Object*
  - *List With Reference*
  - *Test Then Cast*
- 1.1.5
  - *Possible Infinite Loop*
  - *Should Use Math*
  - *ext/hrtime*
- 1.1.4
  - *Double array_flip()*
  - *Fallback Function*
  - *Find Key Directly*
  - *Reuse Variable*

  - *Useless Catch*
- 1.1.3
  - *Useless Referenced Argument*
- 1.1.2
  - *Local Globals*
  - *Missing Include*
- 1.1.1
  - *Inclusion Wrong Case*
- 1.0.11
  - *No Net For Xml Load*
  - *Unused Inherited Variable In Closure*
- 1.0.10
  - *Sqlite3 Requires Single Quotes*
- 1.0.8
  - *Identical Consecutive Expression*
  - *Identical On Both Sides*
  - *Mistaken Concatenation*
  - *No Reference For Ternary*
- 1.0.7
  - *Not A Scalar Type*
  - *Should Use array_filter()*
- 1.0.6
  - *Never Used Parameter*
  - *Use Named Boolean In Argument Definition*
  - *ext/igbinary*
- 1.0.5
  - *Assigned In One Branch*
  - *Environment Variables*
  - *Invalid Regex*
  - *Parent First*
  - *Same Variable Foreach*
- 1.0.4
  - *Argon2 Usage*
  - *Avoid set_error_handler $context Argument*
  - *Can't Count Non-Countable*
  - *Crypto Usage*

  - *Dl() Usage*
  - *Don't Send $this In Constructor*
  - *Hash Will Use Objects*
  - *Incoming Variable Index Inventory*
  - *Integer As Property*
  - *Maybe Missing New*
  - *No get_class() With Null*
  - *Php 7.2 New Class*
  - *Php 7.4 New Class*
  - *Slice Arrays First*
  - *Type Array Index*
  - *Unknown Pcre2 Option*
  - *Use List With Foreach*
  - *Use PHP7 Encapsed Strings*
  - *ext/vips*
- 1.0.3
  - *Ambiguous Static*
  - *Drupal Usage*
  - *FuelPHP Usage*
  - *Phalcon Usage*
- 1.0.1
  - *Could Be Else*
  - *Next Month Trap*
  - *Printf Number Of Arguments*
  - *Simple Switch*
  - *Substring First*
- 0.12.17
  - *Is A PHP Magic Property*
- 0.12.16
  - *Cookies Variables*
  - *Date Formats*
  - *Incoming Variables*
  - *Session Variables*
  - *Too Complex Expression*
  - *Unconditional Break In Loop*
- 0.12.15

  - *Always Anchor Regex*
  - *Is Actually Zero*
  - *Multiple Type Variable*
  - *Session Lazy Write*
- 0.12.14
  - *Regex Inventory*
  - *Switch Fallthrough*
  - *Upload Filename Injection*
- 0.12.12
  - *Use pathinfo() Arguments*
  - *ext/parle*
- 0.12.11
  - *Could Be Protected Class Constant*
  - *Could Be Protected Method*
  - *Method Could Be Private Method*
  - *Method Used Below*
  - *Pathinfo() Returns May Vary*
- 0.12.10
  - *Constant Used Below*
  - *Could Be Private Class Constant*
- 0.12.9
  - *Shell Favorite*
- 0.12.8
  - *ext/fam*
  - *ext/rdkafka*
- 0.12.7
  - *Should Use Foreach*
- 0.12.5
  - *Logical To in_array*
  - *No Substr Minus One*
- 0.12.4
  - *Assign With And Precedence*
  - *Avoid Concat In Loop*
  - *Child Class Removes Typehint*
  - *Isset Multiple Arguments*
  - *Logical Operators Favorite*

- 0.11.7
  - *Forgotten Interface*
  - *Order Of Declaration*
- 0.11.6
  - *Concatenation Interpolation Consistence*
  - *Could Make A Function*
  - *Courier Anti-Pattern*
  - *DI Cyclic Dependencies*
  - *Dependency Injection*
  - *PSR-13 Usage*
  - *PSR-16 Usage*
  - *PSR-3 Usage*
  - *PSR-6 Usage*
  - *PSR-7 Usage*
  - *Too Many Injections*
  - *ext/gender*
  - *ext/judy*
- 0.11.5
  - *Could Typehint*
  - *Implemented Methods Are Public*
  - *Mixed Concat And Interpolation*
  - *No Reference On Left Side*
  - *PSR-11 Usage*
  - *ext/stats*
- 0.11.4
  - *No Class As Typehint*
  - *Use Browscap*
  - *Use Debug*
- 0.11.3
  - *No Return Used*
  - *Only Variable Passed By Reference*
  - *Try With Multiple Catch*
  - *ext/grpc*
  - *ext/sphinx*
- 0.11.2
  - *Alternative Syntax Consistence*

- 0.9.5
  - *One Expression Brackets Consistency*
  - *Should Use Function*
  - *ext/mongodb*
  - *ext/zbarcode*
- 0.9.4
  - *Class Should Be Final By Ocramius*
  - *String*
  - *ext/mhash*
- 0.9.3
  - *Close Tags Consistency*
  - *Unset() Or (unset)*
- 0.9.2
  - *$GLOBALS Or global*
  - *Illegal Name For Method*
  - *Too Many Local Variables*
  - *Use Composer Lock*
  - *ext/ncurses*
  - *ext/newt*
  - *ext/nsapi*
- 0.9.1
  - *Avoid Using stdClass*
  - *Avoid array_push()*
  - *Invalid Octal In String*
- 0.9.0
  - *Getting Last Element*
  - *Rethrown Exceptions*
- 0.8.9
  - *Array() / [ ] Consistence*
  - *Bail Out Early*
  - *Die Exit Consistence*
  - *Dont Change The Blind Var*
  - *More Than One Level Of Indentation*
  - *One Dot Or Object Operator Per Line*
  - *PHP 7.1 Microseconds*
  - *Unitialized Properties*

  - *Useless Check*
- 0.8.7
  - *Don't Echo Error*
  - *No isset() With empty()*
  - *Use ::Class Operator*
  - *Useless Type Casting*
  - *ext/rar*
  - *time() Vs strtotime()*
- 0.8.6
  - *Drop Else After Return*
  - *Modernize Empty With Expression*
  - *Use Positive Condition*
- 0.8.5
  - *Should Make Ternary*
  - *Unused Returned Value*
- 0.8.4
  - *$HTTP_RAW_POST_DATA Usage*
  - *$this Belongs To Classes Or Traits*
  - *$this Is Not An Array*
  - *$this Is Not For Static Methods*
  - *** For Exponent*
  - *<?= Usage*
  - *@ Operator*
  - *Abstract Class Usage*
  - *Abstract Methods Usage*
  - *Abstract Static Methods*
  - *Access Protected Structures*
  - *Accessing Private*
  - *Adding Zero*
  - *Aliases*
  - *Aliases Usage*
  - *All Uppercase Variables*
  - *Already Parents Interface*
  - *Altering Foreach Without Reference*
  - *Always Positive Comparison*
  - *Ambiguous Array Index*

- *Anonymous Classes*
- *Argument Should Be Typehinted*
- *Array Index*
- *Assertions*
- *Assign Default To Properties*
- *Autoloading*
- *Avoid Parenthesis*
- *Avoid Substr() One*
- *Avoid Those Hash Functions*
- *Avoid array_unique()*
- *Avoid get_class()*
- *Avoid sleep()/usleep()*
- *Bad Constants Names*
- *Binary Glossary*
- *Blind Variables*
- *Bracketless Blocks*
- *Break Outside Loop*
- *Break With 0*
- *Break With Non Integer*
- *Buried Assignation*
- *Calltime Pass By Reference*
- *Can't Disable Class*
- *Can't Disable Function*
- *Can't Extend Final*
- *Cant Use Return Value In Write Context*
- *Cast To Boolean*
- *Cast Usage*
- *Catch Overwrite Variable*
- *Caught Exceptions*
- *Caught Expressions*
- *Class Const With Array*
- *Class Has Fluent Interface*
- *Class Usage*
- *Class, Interface Or Trait With Identical Names*
- *Classes Mutually Extending Each Other*
- *Classes Names*

- *Clone Usage*
- *Close Tags*
- *Closure May Use $this*
- *Closures Glossary*
- *Coalesce*
- *Common Alternatives*
- *Compare Hash*
- *Compared Comparison*
- *Composer Namespace*
- *Composer Usage*
- *Composer's autoload*
- *Concrete Visibility*
- *Conditional Structures*
- *Conditioned Constants*
- *Conditioned Function*
- *Confusing Names*
- *Const With Array*
- *Constant Class*
- *Constant Comparison*
- *Constant Conditions*
- *Constant Definition*
- *Constant Scalar Expressions*
- *Constants Created Outside Its Namespace*
- *Constants Names*
- *Constants Usage*
- *Constants With Strange Names*
- *Constructors*
- *Continents*
- *Could Be Class Constant*
- *Could Be Static*
- *Could Use Alias*
- *Could Use Short Assignation*
- *Could Use __DIR__*
- *Could Use self*
- *Custom Class Usage*
- *Custom Constant Usage*

- *Dangling Array References*
- *Deep Definitions*
- *Define With Array*
- *Defined Class Constants*
- *Defined Exceptions*
- *Defined Parent MP*
- *Defined Properties*
- *Defined static:: Or self::*
- *Definitions Only*
- *Dependant Trait*
- *Deprecated PHP Functions*
- *Dereferencing String And Arrays*
- *Direct Injection*
- *Directives Usage*
- *Don't Change Incomings*
- *Double Assignation*
- *Double Instructions*
- *Duplicate Calls*
- *Dynamic Calls*
- *Dynamic Class Constant*
- *Dynamic Classes*
- *Dynamic Code*
- *Dynamic Function Call*
- *Dynamic Methodcall*
- *Dynamic New*
- *Dynamic Property*
- *Dynamically Called Classes*
- *Echo Or Print*
- *Echo With Concat*
- *Ellipsis Usage*
- *Else If Versus Elseif*
- *Else Usage*
- *Email Addresses*
- *Empty Blocks*
- *Empty Classes*
- *Empty Function*

- *Empty Instructions*
- *Empty Interfaces*
- *Empty List*
- *Empty Namespace*
- *Empty Slots In Arrays*
- *Empty Traits*
- *Empty Try Catch*
- *Empty With Expression*
- *Error Messages*
- *Eval() Usage*
- *Exception Order*
- *Exit() Usage*
- *Exit-like Methods*
- *Exponent Usage*
- *External Config Files*
- *Failed Substr Comparison*
- *File Is Component*
- *File Uploads*
- *File Usage*
- *Final Class Usage*
- *Final Methods Usage*
- *Fopen Binary Mode*
- *For Using Functioncall*
- *Foreach Don't Change Pointer*
- *Foreach Needs Reference Array*
- *Foreach Reference Is Not Modified*
- *Foreach With list()*
- *Forgotten Visibility*
- *Forgotten Whitespace*
- *Fully Qualified Constants*
- *Function Called With Other Case Than Defined*
- *Function Subscripting*
- *Function Subscripting, Old Style*
- *Functioncall Is Global*
- *Functions Glossary*
- *Functions In Loop Calls*

- *Functions Removed In PHP 5.4*
- *Functions Removed In PHP 5.5*
- *Functions Using Reference*
- *GPRC Aliases*
- *Global Code Only*
- *Global Import*
- *Global In Global*
- *Global Inside Loop*
- *Global Usage*
- *Globals*
- *Goto Names*
- *HTTP Status Code*
- *Hardcoded Passwords*
- *Has Magic Property*
- *Has Variable Arguments*
- *Hash Algorithms*
- *Hash Algorithms Incompatible With PHP 5.3*
- *Hash Algorithms Incompatible With PHP 5.4/5.5*
- *Heredoc Delimiter Glossary*
- *Hexadecimal Glossary*
- *Hexadecimal In String*
- *Hidden Use Expression*
- *Htmlentities Calls*
- *Http Headers*
- *Identical Conditions*
- *If With Same Conditions*
- *Iffectations*
- *Implement Is For Interface*
- *Implicit Global*
- *Implied If*
- *Inclusions*
- *Incompilable Files*
- *Inconsistent Concatenation*
- *Indices Are Int Or String*
- *Indirect Injection*
- *Instantiating Abstract Class*

- *Interface Arguments*
- *Interface Methods*
- *Interfaces Glossary*
- *Interfaces Usage*
- *Internally Used Properties*
- *Internet Ports*
- *Interpolation*
- *Invalid Constant Name*
- *Is An Extension Class*
- *Is An Extension Constant*
- *Is An Extension Function*
- *Is An Extension Interface*
- *Is CLI Script*
- *Is Composer Class*
- *Is Composer Interface*
- *Is Extension Trait*
- *Is Generator*
- *Is Global Constant*
- *Is Interface Method*
- *Is Not Class Family*
- *Is PHP Constant*
- *Is Upper Family*
- *Joining file()*
- *Labels*
- *Linux Only Files*
- *List Short Syntax*
- *List With Appends*
- *List With Keys*
- *Locally Unused Property*
- *Locally Used Property*
- *Logical Mistakes*
- *Logical Should Use Symbolic Operators*
- *Lone Blocks*
- *Lost References*
- *Magic Constant Usage*
- *Magic Methods*

- *Magic Visibility*
- *Mail Usage*
- *Make Global A Property*
- *Make One Call With Array*
- *Malformed Octal*
- *Mark Callable*
- *Md5 Strings*
- *Method Has Fluent Interface*
- *Method Has No Fluent Interface*
- *Methodcall On New*
- *Methods Without Return*
- *Mime Types*
- *Mixed Keys Arrays*
- *Multidimensional Arrays*
- *Multiple Alias Definitions*
- *Multiple Catch*
- *Multiple Class Declarations*
- *Multiple Classes In One File*
- *Multiple Constant Definition*
- *Multiple Definition Of The Same Argument*
- *Multiple Exceptions Catch()*
- *Multiple Identical Trait Or Interface*
- *Multiple Index Definition*
- *Multiple Returns*
- *Multiples Identical Case*
- *Multiply By One*
- *Must Return Methods*
- *Namespaces*
- *Namespaces Glossary*
- *Negative Power*
- *Nested Ifthen*
- *Nested Loops*
- *Nested Ternary*
- *Never Used Properties*
- *New Functions In PHP 5.4*
- *New Functions In PHP 5.5*

- *New Functions In PHP 5.6*
- *New Functions In PHP 7.0*
- *New Functions In PHP 7.1*
- *No Choice*
- *No Count With 0*
- *No Direct Access*
- *No Direct Call To Magic Method*
- *No Direct Usage*
- *No Hardcoded Hash*
- *No Hardcoded Ip*
- *No Hardcoded Path*
- *No Hardcoded Port*
- *No List With String*
- *No Parenthesis For Language Construct*
- *No Plus One*
- *No Public Access*
- *No Real Comparison*
- *No Self Referencing Constant*
- *No String With Append*
- *No array_merge() In Loops*
- *Non Ascii Variables*
- *Non Static Methods Called In A Static*
- *Non-constant Index In Array*
- *Non-lowercase Keywords*
- *Normal Methods*
- *Not Definitions Only*
- *Not Not*
- *Not Same Name As File*
- *Nowdoc Delimiter Glossary*
- *Null On New*
- *Objects Don't Need References*
- *Octal Glossary*
- *Old Style Constructor*
- *Old Style __autoload()*
- *One Letter Functions*
- *One Object Operator Per Line*

- *One Variable String*
- *Only Static Methods*
- *Only Variable Returned By Reference*
- *Or Die*
- *Overwriting Variable*
- *Overwritten Class Const*
- *Overwritten Exceptions*
- *Overwritten Literals*
- *PHP 7.0 New Classes*
- *PHP 7.0 New Interfaces*
- *PHP 7.0 Removed Directives*
- *PHP 7.0 Removed Functions*
- *PHP 7.1 Removed Directives*
- *PHP Alternative Syntax*
- *PHP Arrays Index*
- *PHP Bugfixes*
- *PHP Constant Usage*
- *PHP Handlers Usage*
- *PHP Interfaces*
- *PHP Keywords As Names*
- *PHP Sapi*
- *PHP Variables*
- *PHP5 Indirect Variable Expression*
- *PHP7 Dirname*
- *Parent, Static Or Self Outside Class*
- *Parenthesis As Parameter*
- *Pear Usage*
- *Perl Regex*
- *Php 7 Indirect Expression*
- *Php 7.1 New Class*
- *Php7 Relaxed Keyword*
- *Phpinfo*
- *Pre-increment*
- *Preprocess Arrays*
- *Preprocessable*
- *Print And Die*

- *Property Could Be Private Property*
- *Property Names*
- *Property Used Above*
- *Property Used Below*
- *Property Variable Confusion*
- *Queries In Loops*
- *Random Without Try*
- *Real Functions*
- *Real Variables*
- *Recursive Functions*
- *Redeclared PHP Functions*
- *Redefined Class Constants*
- *Redefined Default*
- *Redefined Methods*
- *Redefined PHP Traits*
- *Redefined Property*
- *Register Globals*
- *Relay Function*
- *Repeated print()*
- *Reserved Keywords In PHP 7*
- *Resources Usage*
- *Results May Be Missing*
- *Return True False*
- *Return Typehint Usage*
- *Return With Parenthesis*
- *Return void*
- *Safe Curl Options*
- *Same Conditions In Condition*
- *Scalar Typehint Usage*
- *Sensitive Argument*
- *Sequences In For*
- *Setlocale() Uses Constants*
- *Several Instructions On The Same Line*
- *Shell Usage*
- *Short Open Tags*
- *Short Syntax For Arrays*

- *Should Be Single Quote*
- *Should Chain Exception*
- *Should Make Alias*
- *Should Typecast*
- *Should Use Coalesce*
- *Should Use Constants*
- *Should Use Local Class*
- *Should Use Prepared Statement*
- *Silently Cast Integer*
- *Simple Global Variable*
- *Simplify Regex*
- *Slow Functions*
- *Special Integers*
- *Static Loop*
- *Static Methods*
- *Static Methods Called From Object*
- *Static Methods Can't Contain $this*
- *Static Properties*
- *Static Variables*
- *Strict Comparison With Booleans*
- *String May Hold A Variable*
- *Strpos()-like Comparison*
- *Super Global Usage*
- *Super Globals Contagion*
- *Switch To Switch*
- *Switch With Too Many Default*
- *Switch Without Default*
- *Ternary In Concat*
- *Test Class*
- *Throw*
- *Throw Functioncall*
- *Throw In Destruct*
- *Thrown Exceptions*
- *Throws An Assignement*
- *Timestamp Difference*
- *Too Many Children*

- *Trait Methods*
- *Trait Names*
- *Traits Usage*
- *Trigger Errors*
- *True False Inconsistant Case*
- *Try With Finally*
- *Typehints*
- *URL List*
- *Uncaught Exceptions*
- *Unchecked Resources*
- *Undefined Caught Exceptions*
- *Undefined Class Constants*
- *Undefined Classes*
- *Undefined Constants*
- *Undefined Functions*
- *Undefined Interfaces*
- *Undefined Parent*
- *Undefined Properties*
- *Undefined Trait*
- *Undefined static:: Or self::*
- *Unicode Blocks*
- *Unicode Escape Partial*
- *Unicode Escape Syntax*
- *Unknown Directive Name*
- *Unkown Regex Options*
- *Unpreprocessed Values*
- *Unreachable Code*
- *Unresolved Catch*
- *Unresolved Classes*
- *Unresolved Instanceof*
- *Unresolved Use*
- *Unserialize Second Arg*
- *Unset Arguments*
- *Unset In Foreach*
- *Unthrown Exception*
- *Unused Arguments*

- *Unused Classes*
- *Unused Constants*
- *Unused Functions*
- *Unused Global*
- *Unused Interfaces*
- *Unused Label*
- *Unused Methods*
- *Unused Private Methods*
- *Unused Private Properties*
- *Unused Protected Methods*
- *Unused Traits*
- *Unused Use*
- *Unusual Case For PHP Functions*
- *Usage Of class_alias()*
- *Use === null*
- *Use Cli*
- *Use Const And Functions*
- *Use Constant*
- *Use Constant As Arguments*
- *Use Instanceof*
- *Use Lower Case For Parent, Static And Self*
- *Use Nullable Type*
- *Use PHP Object API*
- *Use Pathinfo*
- *Use System Tmp*
- *Use This*
- *Use Web*
- *Use With Fully Qualified Name*
- *Use const*
- *Use password_hash()*
- *Use random_int()*
- *Used Classes*
- *Used Functions*
- *Used Interfaces*
- *Used Methods*
- *Used Once Variables (In Scope)*

- *Used Once Variables*
- *Used Private Methods*
- *Used Protected Method*
- *Used Static Properties*
- *Used Trait*
- *Used Use*
- *Useless Abstract Class*
- *Useless Brackets*
- *Useless Constructor*
- *Useless Final*
- *Useless Global*
- *Useless Instructions*
- *Useless Interfaces*
- *Useless Parenthesis*
- *Useless Return*
- *Useless Switch*
- *Useless Unset*
- *Uses Default Values*
- *Uses Environment*
- *Using $this Outside A Class*
- *Using Short Tags*
- *Usort Sorting In PHP 7.0*
- *Var Keyword*
- *Variable Constants*
- *Variable References*
- *Variable Variables*
- *Variables With Long Names*
- *Variables With One Letter Names*
- *While(List() = Each())*
- *Written Only Variables*
- *Wrong Class Name Case*
- *Wrong Function Name Case*
- *Wrong Number Of Arguments*
- *Wrong Number Of Arguments In Methods*
- *Wrong Optional Parameter*
- *Wrong Parameter Type*

- *Wrong fopen() Mode*
- *Yield From Usage*
- *Yield Usage*
- *Yoda Comparison*
- *::class*
- *__debugInfo() Usage*
- *__halt_compiler*
- *__toString() Throws Exception*
- *crypt() Without Salt*
- *error_reporting() With Integers*
- *eval() Without Try*
- *ext/0mq*
- *ext/amqp*
- *ext/apache*
- *ext/apc*
- *ext/apcu*
- *ext/array*
- *ext/bcmath*
- *ext/bzip2*
- *ext/cairo*
- *ext/calendar*
- *ext/com*
- *ext/crypto*
- *ext/ctype*
- *ext/curl*
- *ext/cyrus*
- *ext/date*
- *ext/dba*
- *ext/dio*
- *ext/dom*
- *ext/eaccelerator*
- *ext/enchant*
- *ext/ereg*
- *ext/ev*
- *ext/event*
- *ext/exif*

- *ext/expect*
- *ext/fann*
- *ext/fdf*
- *ext/ffmpeg*
- *ext/file*
- *ext/fileinfo*
- *ext/filter*
- *ext/fpm*
- *ext/ftp*
- *ext/gd*
- *ext/gearman*
- *ext/geoip*
- *ext/gettext*
- *ext/gmagick*
- *ext/gmp*
- *ext/gnupgp*
- *ext/hash*
- *ext/ibase*
- *ext/iconv*
- *ext/iis*
- *ext/imagick*
- *ext/imap*
- *ext/info*
- *ext/inotify*
- *ext/intl*
- *ext/json*
- *ext/kdm5*
- *ext/ldap*
- *ext/libevent*
- *ext/libxml*
- *ext/lua*
- *ext/mail*
- *ext/mailparse*
- *ext/math*
- *ext/mbstring*
- *ext/mcrypt*

- *ext/memcache*
- *ext/memcached*
- *ext/ming*
- *ext/mongo*
- *ext/mssql*
- *ext/mysql*
- *ext/mysqli*
- *ext/ob*
- *ext/oci8*
- *ext/odbc*
- *ext/opcache*
- *ext/openssl*
- *ext/parsekit*
- *ext/password*
- *ext/pcntl*
- *ext/pcre*
- *ext/pdo*
- *ext/pecl_http*
- *ext/pgsql*
- *ext/phalcon*
- *ext/phar*
- *ext/php-ast*
- *ext/posix*
- *ext/proctitle*
- *ext/pspell*
- *ext/readline*
- *ext/recode*
- *ext/redis*
- *ext/reflection*
- *ext/runkit*
- *ext/sem*
- *ext/session*
- *ext/shmop*
- *ext/simplexml*
- *ext/snmp*
- *ext/soap*

- *ext/sockets*
- *ext/spl*
- *ext/sqlite*
- *ext/sqlite3*
- *ext/sqlsrv*
- *ext/ssh2*
- *ext/standard*
- *ext/suhosin*
- *ext/tidy*
- *ext/tokenizer*
- *ext/tokyotyrant*
- *ext/trader*
- *ext/v8js*
- *ext/wddx*
- *ext/wikidiff2*
- *ext/wincache*
- *ext/xcache*
- *ext/xdebug*
- *ext/xdiff*
- *ext/xhprof*
- *ext/xml*
- *ext/xmlreader*
- *ext/xmlrpc*
- *ext/xmlwriter*
- *ext/xsl*
- *ext/yaml*
- *ext/yis*
- *ext/zip*
- *ext/zlib*
- *func_get_arg() Modified*
- *include_once() Usage*
- *isset() With Constant*
- *list() May Omit Variables*
- *mcrypt_create_iv() With Default Values*
- *parse_str() Warning*
- *preg_match_all() Flag*

  - *preg_replace With Option e*
  - *set_exception_handler() Warning*
  - *var_dump()... Usage*
- 0.8.3
  - *Variable Global*

13.4 Directory by PHP Function

- ‘ ‘
  - ‘ xmlwriter_open_memory()‘
    - *ext/xmlwriter*
- *$*
  - *$HTTP_RAW_POST_DATA*
    - *$HTTP_RAW_POST_DATA Usage*
  - *$_ENV*
    - *Incoming Variables*
    - *Uses Environment*
    - *No Hardcoded Port*
    - *Useless Global*
    - *Incoming Variable Index Inventory*
  - *$_GET*
    - *ext/gd*
    - *ext/pcre*
    - *Avoid mb_dectect_encoding()*
    - *Cast Usage*
    - *Incoming Values*
    - *Incoming Variables*
    - *Safe Phpvariables*
    - *Should Use Coalesce*
    - *Super Global Usage*
    - *Use Web*
    - *Always Anchor Regex*
    - *Direct Injection*
    - *filter_input() As A Source*
    - *GPRC Aliases*
    - *Indirect Injection*

        - *Integer Conversion*
        - *Eval() Usage*
        - *Implied If*
        - *Don't Change Incomings*
        - *Repeated Regex*
        - *Useless Global*
        - *Incoming Variable Index Inventory*
        - *PHP Variables*
    - *$_POST*
        - *Crypto Usage*
        - *PHP Keywords As Names*
        - *Super Global Usage*
        - *GPRC Aliases*
        - *Indirect Injection*
        - *Register Globals*
        - *Don't Change Incomings*
        - *Useless Global*
        - *Incoming Variable Index Inventory*
        - *All Uppercase Variables*
    - *$_REQUEST*
        - *Super Global Usage*
        - *GPRC Aliases*
        - *Indirect Injection*
        - *Register Globals*
        - *Useless Global*
        - *Incoming Variable Index Inventory*
    - *$this*
        - *Accessing Private*
        - *Avoid Optional Properties*
        - *Avoid option arrays in constructors*
        - *Check On __Call Usage*
        - *Could Be Class Constant*
        - *Property Could Be Private Property*
        - *Method Could Be Private Method*
        - *Could Be Protected Method*
        - *Could Be Protected Property*

* *Method Could Be Static*
* *Cyclic References*
* *Law of Demeter*
* *Dependant Abstract Classes*
* *No Direct Call To Magic Method*
* *Disconnected Classes*
* *Don't Send $this In Constructor*
* *Dynamic Self Calls*
* *Class Has Fluent Interface*
* *Insufficient Property Typehint*
* *Is A PHP Magic Property*
* *Locally Unused Property*
* *Locally Used Property*
* *Assign Default To Properties*
* *Make Global A Property*
* *Make Magic Concrete*
* *Method Used Below*
* *No Magic Method With Array*
* *Non Nullable Getters*
* *Non Static Methods Called In A Static*
* *Parent First*
* *Property Could Be Local*
* *Never Used Properties*
* *Property Used Above*
* *Property Used Below*
* *Property Used In One Method Only*
* *Internally Used Properties*
* *Redefined Default*
* *Scalar Or Object Property*
* *Should Deep Clone*
* *Should Have Destructor*
* *Should Use Local Class*
* *Static Methods Can't Contain $this*
* *Static Methods Called From Object*
* *$this Belongs To Classes Or Traits*
* *$this Is Not An Array*

* *$this Is Not For Static Methods*
* *Throw In Destruct*
* *Too Many Injections*
* *DI Cyclic Dependencies*
* *Wrong Access Style to Property*
* *Undefined Properties*
* *Uninitialized Property*
* *Unitialized Properties*
* *Unused Methods*
* *Unused Private Methods*
* *Unused Private Properties*
* *Unused Protected Methods*
* *Used Methods*
* *Used Once Property*
* *Used Private Methods*
* *Used Static Properties*
* *Used Protected Method*
* *Useless Typehint*
* *Use This*
* *Using $this Outside A Class*
* *Wrong Typed Property Default*
* *Create Default Values*
* *Create Magic Property*
* *Make Class Method Definition*
* *Set Class Property Definition With Typehint*
* *Set Clone Link*
* *Solve Trait Methods*
* *Long Preparation For Throw*
* *Cannot Use Static For Closure*
* *Could Be Static Closure*
* *Method Has Fluent Interface*
* *Method Has No Fluent Interface*
* *Must Return Methods*
* *Unbinding Closures*
* *Wrong Number Of Arguments In Methods*
* *Interfaces Don't Ensure Properties*

        * *Courier Anti-Pattern*
        * *Dependency Injection*
        * *Memoize MagicCall*
        * *Closure May Use $this*
        * *Could Use Promoted Properties*
        * *__debugInfo() Usage*
        * *Union Typehint*
        * *Typed Property Usage*
        * *Minus One On Error*
        * *Too Complex Expression*
        * *Avoid Large Array Assignation*
        * *More Than One Level Of Indentation*
        * *Property Variable Confusion*
        * *Set Aside Code*
        * *__toString() Throws Exception*
        * *var_dump()... Usage*
        * *Dependant Trait*
        * *Locally Used Property In Trait*
        * *Unused Trait In Class*
        * *Complex Dynamic Names*
        * *Used Once Variables*
- *(*
    - *()*
        * *Could Be Parent Method*
        * *Property Could Be Private Property*
        * *Method Could Be Private Method*
        * *Defined Parent MP*
        * *Law of Demeter*
        * *Class Should Be Final By Ocramius*
        * *Fossilized Method*
        * *Insufficient Property Typehint*
        * *Is Interface Method*
        * *Non Static Methods Called In A Static*
        * *Order Of Declaration*
        * *Never Used Properties*
        * *Property Used In One Method Only*

    - *Unused Protected Methods*
    - *Useless Typehint*
    - *Follow Closure Definition*
    - *Solve Trait Methods*
    - *ext/async*
    - *ext/ev*
    - *ext/newt*
    - *ext/reflection*
    - *Callback Function Needs Return*
    - *Cannot Use Static For Closure*
    - *Exceeding Typehint*
    - *Fallback Function*
    - *Is Generator*
    - *No Literal For Reference*
    - *No Return Used*
    - *Real Functions*
    - *Should Yield With Key*
    - *Too Many Local Variables*
    - *Too Much Indented*
    - *Unbinding Closures*
    - *Wrong Function Name Case*
    - *Interfaces Don't Ensure Properties*
    - *Possible Interfaces*
    - *Memoize MagicCall*
    - *Crc32() Might Be Negative*
    - *idn_to_ascii() New Default*
    - *PHP 7.1 Scalar Typehints*
    - *Wrong Attribute Configuration*
    - *Directly Use File*
    - *Don't Loop On Yield*
    - *Max Level Of Nesting*
    - *Could Use Trait*
    - *Undefined Insteadof*
    - *PHP5 Indirect Variable Expression*
    - *Non Ascii Variables*
- *

- **
  - *Modify Immutable*
  - *Composer Namespace*
  - *ext/bcmath*
  - *ext/decimal*
  - *ext/reflection*
  - *ext/sdl*
  - *Mismatch Type And Default*
  - *Only Variable Passed By Reference*
  - *Using Deprecated Method*
  - *Exponent Usage*
  - *** For Exponent*
  - *Constant Scalar Expressions*
  - *Negative Power*
  - *Unused Traits*
  - *Drupal Usage*
  - *Laravel usage*
  - *Symfony usage*
- .
  - ...
    - *No Spread For Hash*
    - *Ambiguous Static*
    - *Check On __Call Usage*
    - *Static Properties*
    - *Used Once Property*
    - *Constant Dynamic Creation*
    - *ext/ffi*
    - *ext/ldap*
    - *ext/phalcon*
    - *ext/sockets*
    - *ext/xattr*
    - *Method Has Fluent Interface*
    - *Mismatch Parameter Name*
    - *Multiple Definition Of The Same Argument*
    - *Must Return Methods*
    - *Unknown Parameter Name*

        - *No array_merge() In Loops*
        - *Ellipsis Usage*
        - *PHP 80 Named Parameter Variadic*
        - *Reserved Keywords In PHP 7*
        - *Signature Trailing Comma*
        - *Spread Operator For Array*
        - *Unpacking Inside Arrays*
        - *array_merge() And Variadic*
        - *File Usage*
        - *Iffectations*
        - *Repeated Regex*
        - *Should Use Operator*
        - *Useless Instructions*
        - *Pack Format Inventory*
        - *Yii usage*
- @
    - @
        - *ext/mssql*
        - *ext/yaml*
        - *Use PHP Attributes*
        - *Too Complex Expression*
        - *@ Operator*
        - *Useless Instructions*
        - *Email Addresses*
        - *Invalid Octal In String*
        - *Remove Noscream @*
- ***A***
    - *AF_INET*
        - *ext/sockets*
    - *AMQPChannel*
        - *ext/amqp*
    - *AMQPConnection*
        - *ext/amqp*
    - *ArrayAccess*
        - *$this Is Not An Array*
        - *Is An Extension Interface*

        - *Switch To Switch*
        - *Switch With Too Many Default*
        - *Switch Without Default*
        - *Unconditional Break In Loop*
        - *Unreachable Code*
        - *Use Case Value*
        - *Useless Switch*

- ***C***
    - *CAL_GREGORIAN*
        - *ext/calendar*
    - *COM*
        - *ext/com*
    - *COUNT_NORMAL*
        - *Use Count Recursive*
    - *COUNT_RECURSIVE*
        - *Use Count Recursive*
    - *CURLOPT_FILE*
        - *ext/curl*
    - *CURLOPT_HEADER*
        - *ext/curl*
    - *CURLOPT_SSL_VERIFYPEER*
        - *Safe Curl Options*
    - *CURLOPT_URL*
        - *Safe Curl Options*
    - *CURLPIPE_HTTP1*
        - *PHP 7.4 Constant Deprecation*
    - *CURLVERSION_NOW*
        - *curl_version() Has No Argument*
    - *Cairo*
        - *ext/cairo*
    - *CairoContext*
        - *ext/cairo*
    - *CairoPSSurface*
        - *ext/cairo*
    - *Closure*
        - *Follow Closure Definition*

- *chroot()*
  - *No Hardcoded Path*
- *class_alias()*
  - *Set Class_Alias Definition*
- *class_exists()*
  - *Undefined ::class*
- *closure*
  - *Parent, Static Or Self Outside Class*
  - *Should Use Local Class*
  - *Using $this Outside A Class*
  - *Follow Closure Definition*
  - *Propagate Calls*
  - *Collect Parameter Counts*
  - *Cannot Use Static For Closure*
  - *Closure Could Be A Callback*
  - *Closures Glossary*
  - *Could Be Typehinted Callable*
  - *Could Be Static Closure*
  - *Function With Dynamic Code*
  - *Functions Glossary*
  - *Multiple Identical Closure*
  - *Multiple Definition Of The Same Argument*
  - *Real Functions*
  - *Semantic Typing*
  - *Unbinding Closures*
  - *Unused Inherited Variable In Closure*
  - *Hidden Use Expression*
  - *Avoid set_error_handler $context Argument*
  - *Closure May Use $this*
  - *Should Use array_filter()*
  - *Use Closure Trailing Comma*
  - *preg_replace With Option e*
  - *No Static Variable In A Method*
- *collator_compare()*
  - *Strpos()-like Comparison*
- *collator_get_sort_key()*

* *ext/cairo*
* *ext/cmark*
* *ext/com*
* *ext/crypto*
* *ext/curl*
* *ext/ds*
* *ext/eaccelerator*
* *ext/eio*
* *ext/enchant*
* *ext/event*
* *ext/fam*
* *ext/fann*
* *ext/fdf*
* *ext/ffi*
* *ext/filter*
* *ext/gender*
* *ext/geoip*
* *ext/grpc*
* *ext/pecl_http*
* *ext/igbinary*
* *ext/judy*
* *ext/lapack*
* *ext/leveldb*
* *ext/libsodium*
* *ext/mail*
* *ext/mongo*
* *ext/mongodb*
* *ext/msgpack*
* *ext/mssql*
* *ext/mysql*
* *ext/mysqli*
* *ext/newt*
* *ext/oci8*
* *ext/opencensus*
* *ext/pcov*
* *ext/phalcon*

* *ext/psr*
* *ext/rar*
* *ext/rdkafka*
* *ext/recode*
* *ext/redis*
* *ext/sdl*
* *ext/seaslog*
* *ext/sockets*
* *ext/sphinx*
* *ext/sqlsrv*
* *ext/ssh2*
* *ext/svm*
* *ext/swoole*
* *ext/tokyotyrant*
* *ext/uopz*
* *ext/uuid*
* *ext/v8js*
* *ext/vips*
* *ext/wasm*
* *ext/weakref*
* *ext/wikidiff2*
* *ext/xmlrpc*
* *ext/xxtea*
* *ext/zbarcode*
* *ext/zend_monitor*
* *ext/0mq*
* *Use Named Boolean In Argument Definition*
* *Could Be Static Closure*
* *Hardcoded Passwords*
* *Method Has Fluent Interface*
* *Insufficient Typehint*
* *Exit-like Methods*
* *Mismatch Parameter Name*
* *Multiple Returns*
* *No Boolean As Default*
* *No Class As Typehint*

* *Nullable With Constant*
* *Should Use Constants*
* *Should Yield With Key*
* *Too Many Parameters*
* *Wrong Function Name Case*
* *Empty Interfaces*
* *Abstract Away*
* *An OOP Factory*
* *array_key_exists() Speedup*
* *No mb_substr In Loop*
* *Memoize MagicCall*
* *Avoid glob() Usage*
* *Slow Functions*
* *Assumptions*
* *Avoid mb_dectect_encoding()*
* *Cant Use Return Value In Write Context*
* *Group Use Trailing Comma*
* *Must Call Parent Constructor*
* *No Reference For Static Property*
* *Not A Scalar Type*
* *PHP 7.0 Scalar Typehints*
* *PHP 7.1 Scalar Typehints*
* *PHP 7.3 Last Empty Argument*
* *PHP Resources Turned Into Objects*
* *Php 8.0 Variable Syntax Tweaks*
* *Return With Parenthesis*
* *Should Use array_column()*
* *Should Use Function*
* *Use Cookies*
* *Use DateTimeImmutable Class*
* *Yield Usage*
* *GLOB_BRACE Usage*
* *PSR-11 Usage*
* *Compare Hash*
* *Check Crypto Key Length*
* *Safe Curl Options*

* *filter_input() As A Source*
* *Integer Conversion*
* *Minus One On Error*
* *move_uploaded_file Instead Of copy*
* *No Net For Xml Load*
* *No Weak SSL Crypto*
* *Safe HTTP Headers*
* *Session Lazy Write*
* *Set Cookie Safe Arguments*
* *Should Use Prepared Statement*
* *Unserialize Second Arg*
* *Upload Filename Injection*
* *Bail Out Early*
* *Concat Empty String*
* *Dangling Array References*
* *Don't Read And Write In One Expression*
* *Else Usage*
* *Empty Try Catch*
* *Foreach With list()*
* *Forgotten Whitespace*
* *Logical Mistakes*
* *Mail Usage*
* *Nested Ternary*
* *Next Month Trap*
* *No Append On Source*
* *No Hardcoded Hash*
* *No Hardcoded Ip*
* *No Hardcoded Port*
* *Regex Delimiter*
* *Should Chain Exception*
* *Suspicious Comparison*
* *Timestamp Difference*
* *Use Constant*
* *Use Debug*
* *Use System Tmp*
* *Http Headers*

* *Path lists*
* *URL List*
* *Strange Name For Variables*
* *Variables With One Letter Names*
* *Codeigniter usage*
* *FuelPHP Usage*
* *Laravel usage*
* *Phalcon Usage*
* *Symfony usage*
* *Wordpress usage*
* *Yii usage*
* *php-cs-fixable*
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable
* report-php-cs-fixable

        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
        * report-php-cs-fixable
    - *compact()*
        * *Create Compact Variables*
        * *Compact Inexistant Variable*
    - *constant()*
        * *Dynamic Class Constant*
        * *Variable Constants*
        * *Fully Qualified Constants*
        * *PHP 7.4 Reserved Keyword*
    - *continue*
        * *Upload Filename Injection*
        * *Bail Out Early*
        * *Break Outside Loop*
        * *Continue Is For Loop*
        * *No Need For Else*
        * *No Return Or Throw In Finally*
        * *More Than One Level Of Indentation*

        - *Unconditional Break In Loop*
        - *Unreachable Code*
        - *Useless Instructions*
    - *convert_cyr_string()*
        - *PHP 7.4 Removed Functions*
        - *PHP 8.0 Removed Functions*
    - *copy()*
        - *Protocol lists*
    - *count()*
        - *$this Is Not For Static Methods*
        - *Use Constant As Arguments*
        - *Uses Default Values*
        - *PHP Interfaces*
        - *Cache Variable Outside Loop*
        - *No Count With 0*
        - *Use is_countable*
        - *Always Positive Comparison*
        - *Useless Check*
    - *countable*
        - *Use is_countable*
        - *Can't Count Non-Countable*
    - *create_function()*
        - *PHP 7.2 Deprecations*
        - *PHP 7.2 Removed Functions*
        - *PHP 8.0 Removed Functions*
    - *crypt()*
        - *ext/password*
        - *Use password_hash()*
        - *crypt() Without Salt*
    - *curl_exec()*
        - *Strpos()-like Comparison*
    - *curl_init()*
        - *PHP Resources Turned Into Objects*
        - *Safe Curl Options*
    - *curl_multi_errno()*
        - *New Functions In PHP 7.1*

- *DateTimeImmutable*
  - *Use DateTimeImmutable Class*
- *DateTimeZone*
  - *ext/date*
- *Datetime*
  - *Use DateTimeImmutable Class*
- *Define()*
  - *Constant Case Preference*
- *Die*
  - *Die Exit Consistence*
- *Die()*
  - *Print And Die*
- *Directory*
  - *ext/ldap*
- *DirectoryIterator*
  - *Protocol lists*
- *DivisionByZeroError*
  - *Throw*
- *date()*
  - *Abstract Away*
- *dateTime*
  - *Clone Usage*
- *date_create()*
  - *PHP 7.1 Microseconds*
- *datetime*
  - *ext/date*
  - *Date Formats*
  - *Use DateTimeImmutable Class*
  - *Timestamp Difference*
- *datetimeimmutable*
  - *Use DateTimeImmutable Class*
- *debug_backtrace()*
  - *Use Debug*
- *debug_zval_dump()*
  - *Use Debug*
- *define()*

    - *Htmlentities Calls*
- *EV_PERSIST*
    - *ext/libevent*
- *EV_READ*
    - *ext/libevent*
- *EXTR_OVERWRITE*
    - *Configure Extract*
- *EXTR_PREFIX_ALL*
    - *Configure Extract*
- *EXTR_SKIP*
    - *Configure Extract*
- *E_ALL*
    - *Dynamic Class Constant*
    - *Is Global Constant*
    - *ext/sockets*
    - *error_reporting() With Integers*
- *E_DEPRECATED*
    - *error_reporting() With Integers*
- *E_ERROR*
    - *Use Constant As Arguments*
- *E_NOTICE*
    - *crypt() Without Salt*
    - *error_reporting() With Integers*
- *E_PARSE*
    - *Use Constant As Arguments*
- *E_STRICT*
    - *error_reporting() With Integers*
- *E_USER_ERROR*
    - *ext/oci8*
    - *PHP Handlers Usage*
    - *Trigger Errors*
- *E_USER_NOTICE*
    - *PHP Handlers Usage*
- *E_USER_WARNING*
    - *PHP Handlers Usage*
- *E_WARNING*

        - *Use Constant As Arguments*
        - *error_reporting() With Integers*
    - *Each()*
        - *While(List() = Each())*
    - *Ev*
        - *ext/ev*
    - *EvTimer*
        - *ext/ev*
    - *Event*
        - *ext/event*
    - *EventBase*
        - *ext/event*
    - *EventBufferEvent*
        - *ext/event*
    - *EventDnsBase*
        - *ext/event*
    - *EventUtil*
        - *ext/event*
    - *Exit*
        - *Multiple Returns*
        - *Die Exit Consistence*
    - *each()*
        - *PHP 7.2 Deprecations*
        - *PHP 7.2 Removed Functions*
        - *PHP 8.0 Removed Functions*
    - *easter_days()*
        - *Use Constant As Arguments*
    - *eio_event_loop()*
        - *ext/eio*
    - *empty()*
        - *No Count With 0*
        - *Cant Use Return Value In Write Context*
        - *Missing __isset() Method*
        - *Empty With Expression*
        - *Modernize Empty With Expression*
        - *No isset() With empty()*

    - *Variable Is Not A Condition*
    - *Useless Check*
- *enchant_broker_init()*
    - *ext/enchant*
    - *PHP Resources Turned Into Objects*
- *enchant_broker_request_dict()*
    - *PHP Resources Turned Into Objects*
- *enchant_broker_request_pwl_dict()*
    - *PHP Resources Turned Into Objects*
- *ereg()*
    - *PHP 7.0 Removed Functions*
- *ereg_replace()*
    - *PHP 7.0 Removed Functions*
- *eregi()*
    - *PHP 7.0 Removed Functions*
- *eregi_replace()*
    - *PHP 7.0 Removed Functions*
- *error_clear_last()*
    - *New Functions In PHP 7.0*
- *error_get_last()*
    - *$php_errormsg Usage*
- *error_log()*
    - *Error_Log() Usage*
- *error_reporting()*
    - *Use Constant As Arguments*
    - *PHP Handlers Usage*
- *ev*
    - *ext/ev*
    - *Encoded Simple Letters*
- *eval()*
    - *Function With Dynamic Code*
    - *Can't Disable Function*
    - *Eval() Usage*
    - *preg_replace With Option e*
- *event*
    - *ext/ev*

        - *ext/event*
        - *ext/inotify*
        - *ext/libevent*
        - *ext/swoole*
        - *Minus One On Error*
        - *Empty Try Catch*
    - *event_base_new()*
        - *ext/libevent*
    - *event_new()*
        - *ext/libevent*
    - *exec()*
        - *Shell Favorite*
        - *Can't Disable Function*
        - *Shell commands*
    - *exit*
        - *ext/dba*
        - *ext/event*
        - *ext/ftp*
        - *ext/gearman*
        - *ext/libevent*
        - *ext/pcntl*
        - *ext/zip*
        - *Exit-like Methods*
        - *PHP Handlers Usage*
        - *Don't Echo Error*
        - *Die Exit Consistence*
        - *Else Usage*
        - *Error Messages*
        - *Exit() Usage*
        - *Print And Die*
    - *exit()*
        - *ext/event*
        - *ext/mysqli*
        - *ext/pcntl*
        - *Exit-like Methods*
        - *Use PHP Object API*

- *FilesystemIterator*
  - *ext/spl*
- *For()*
  - *Sequences In For*
- *Foreach()*
  - *Altering Foreach Without Reference*
  - *Should Use Foreach*
  - *Useless Check*
  - *Use List With Foreach*
- *fdf_create()*
  - *ext/fdf*
- *feof()*
  - *Possible Infinite Loop*
- *ffi*
  - *ext/ffi*
- *ffmpeg_movie*
  - *ext/ffmpeg*
- *fgetc()*
  - *Strpos()-like Comparison*
- *fgetcsv()*
  - *Possible Infinite Loop*
- *fgets()*
  - *Possible Infinite Loop*
- *fgetss()*
  - *PHP 8.0 Removed Functions*
  - *Possible Infinite Loop*
- *file()*
  - *Joining file()*
- *file_exists()*
  - *Protocol lists*
- *file_get_contents()*
  - *Joining file()*
  - *Strpos()-like Comparison*
- *file_put_contents()*
  - *No array_merge() In Loops*
  - *Strpos()-like Comparison*

- *filesize()*
  - *Protocol lists*
- *filter_input()*
  - *Use Constant As Arguments*
  - *filter_input() As A Source*
- *filter_input_array()*
  - *filter_input() As A Source*
- *filter_var()*
  - *Use Constant As Arguments*
- *fopen()*
  - *Wrong fopen() Mode*
  - *Fopen Binary Mode*
  - *@ Operator*
  - *Possible Infinite Loop*
  - *Protocol lists*
- *for()*
  - *Bracketless Blocks*
  - *Constant Conditions*
  - *For Using Functioncall*
- *foreach()*
  - *Foreach() Favorite*
  - *Should Yield With Key*
  - *Useless Referenced Argument*
  - *Slow Functions*
  - *Foreach Don't Change Pointer*
  - *preg_match_all() Flag*
  - *Should Use array_column()*
  - *Should Use array_filter()*
  - *Bracketless Blocks*
  - *Break Outside Loop*
  - *Dont Change The Blind Var*
  - *Overwritten Source And Value*
  - *Foreach With list()*
  - *Find Key Directly*
  - *Avoid array_unique()*
  - *No Direct Usage*

- ***H***
  - *HTML_ENTITIES*
    - *Is PHP Constant*
  - *hash()*
    - *Directly Use File*
  - *hash_algos()*
    - *Hash Algorithms*
  - *hash_equals()*
    - *Compare Hash*
  - *hash_file()*
    - *Directly Use File*
  - *hash_hmac()*
    - *Directly Use File*
  - *hash_update()*
    - *Directly Use File*
  - *hash_update_file()*
    - *Directly Use File*
  - *header()*
    - *Use Cookies*
    - *Should Use SetCookie()*
    - *Http Headers*
  - *hebrevc()*
    - *PHP 7.4 Removed Functions*
    - *PHP 8.0 Removed Functions*
  - *highlight_file()*
    - *Directly Use File*
  - *highlight_string()*
    - *Directly Use File*
  - *html_entity_decode()*
    - *Use Constant As Arguments*
  - *htmlentities()*
    - *Use Constant As Arguments*
    - *Uses Default Values*
    - *Htmlentities Calls*
  - *htmlspecialchars()*
    - *Use Constant As Arguments*

- *iconv_strpos()*
  - *Strpos()-like Comparison*
- *iconv_strrpos()*
  - *Strpos()-like Comparison*
- *idn_to_ascii()*
  - *idn_to_ascii() New Default*
- *image2wbmp()*
  - *PHP 7.3 Removed Functions*
  - *PHP 8.0 Removed Functions*
- *imagecolorallocate()*
  - *Strpos()-like Comparison*
- *imagecolorallocatealpha()*
  - *Strpos()-like Comparison*
- *imagepsbbox()*
  - *PHP 7.0 Removed Functions*
- *imagepsencodefont()*
  - *PHP 7.0 Removed Functions*
- *imagepsextendfont()*
  - *PHP 7.0 Removed Functions*
- *imagepsfreefont()*
  - *PHP 7.0 Removed Functions*
- *imagepsloadfont()*
  - *PHP 7.0 Removed Functions*
- *imagepsslantfont()*
  - *PHP 7.0 Removed Functions*
- *imagepstext()*
  - *PHP 7.0 Removed Functions*
- *imagick*
  - *ext/imagick*
- *imap_last_error()*
  - *ext/imap*
- *imap_open()*
  - *Can't Disable Function*
- *implode()*
  - *Joining file()*
  - *Avoid Concat In Loop*

    - *Implode One Arg*
    - *Implode() Arguments Order*
    - *Use Array Functions*
- *import_request_variables()*
    - *Register Globals*
- *in_array()*
    - *Logical To in_array*
    - *Processing Collector*
    - *Slow Functions*
    - *Strict Comparison With Booleans*
- *ini_get()*
    - *PHP 8.0 Removed Directives*
- *ini_set()*
    - *Definitions Only*
- *inotify_init()*
    - *ext/inotify*
- *inotify_queue_len()*
    - *ext/inotify*
- *inotify_read()*
    - *ext/inotify*
- *instanceof*
    - *Usage Of class_alias()*
    - *Class Usage*
    - *self, parent, static Outside Class*
    - *Scalar Or Object Property*
    - *Undefined Classes*
    - *Undefined ::class*
    - *Unresolved Instanceof*
    - *Use Instanceof*
    - *ext/psr*
    - *Could Typehint*
    - *Already Parents Interface*
    - *Cant Implement Traversable*
    - *Interfaces Usage*
    - *Is An Extension Interface*
    - *Undefined Interfaces*

- *is_null()*
  - *Use === null*
  - *Should Use Operator*
- *is_object()*
  - *Use Instanceof*
  - *Should Use Operator*
- *is_real()*
  - *Avoid Real*
- *is_resource()*
  - *PHP Resources Turned Into Objects*
- *is_scalar()*
  - *Use Instanceof*
- *is_string()*
  - *Use Instanceof*
  - *Check All Types*
- *isset*
  - *Use Instanceof*
  - *ext/session*
  - *ext/xml*
  - *Must Return Methods*
  - *array_key_exists() Speedup*
  - *Isset() On The Whole Array*
  - *Slow Functions*
  - *Assert Function Is Reserved*
  - *Cookies Variables*
  - *Isset Multiple Arguments*
  - *Session Variables*
  - *Should Use array_column()*
  - *Should Use array_filter()*
  - *Should Use Coalesce*
  - *Too Complex Expression*
  - *isset() With Constant*
  - *No isset() With empty()*
  - *Variable Is Not A Condition*
  - *Useless Check*
- *iterator_to_array()*

    - *No Net For Xml Load*
- *LIBXML_ERR_ERROR*
    - *ext/libxml*
- *LIBXML_ERR_FATAL*
    - *ext/libxml*
- *LIBXML_ERR_WARNING*
    - *ext/libxml*
- *LIBXML_NOENT*
    - *No Net For Xml Load*
- *LOG_DEBUG*
    - *ext/rdkafka*
- *Lapack*
    - *ext/lapack*
- *LevelDB*
    - *ext/leveldb*
- *LevelDBWriteBatch*
    - *ext/leveldb*
- *List()*
    - *List With Appends*
- *Locale*
    - *ext/intl*
- *LogicException*
    - *PHP Exception*
- *Lua*
    - *ext/lua*
- *lapack*
    - *ext/lapack*
- *ldap_sort()*
    - *PHP 8.0 Removed Functions*
- *leveldb*
    - *ext/leveldb*
- *libxml_clear_errors()*
    - *ext/libxml*
- *libxml_get_errors()*
    - *ext/libxml*
    - *ext/xsl*

- *MCRYPT_MODE_CBC*
  - *ext/mcrypt*
- *MCRYPT_MODE_CFB*
  - *mcrypt_create_iv() With Default Values*
- *MCRYPT_RAND*
  - *ext/mcrypt*
- *MCRYPT_RIJNDAEL_128*
  - *ext/mcrypt*
- *MHASH_MD5*
  - *ext/mhash*
- *MYSQLI*
  - *Should Use Prepared Statement*
- *M_PI*
  - *Use Constant*
- *Match()*
  - *Could Use Match*
- *Memcache*
  - *ext/memcache*
- *Memcached*
  - *ext/memcached*
- *MessageFormatter*
  - *Null On New*
- *MessagePack*
  - *ext/msgpack*
- *Mongo*
  - *ext/mongodb*
- *MongoClient*
  - *ext/mongo*
- *MongoDB*
  - *ext/mongo*
  - *ext/mongodb*
- *MongoDb*
  - *ext/mongodb*
- *Mongodb*
  - *ext/mongodb*
- *MySQLI*

- *md5_file()*
  - *Directly Use File*
- *memcache*
  - *ext/memcache*
- *memcached*
  - *ext/memcache*
  - *ext/memcached*
- *microtime()*
  - *Use random_int()*
- *mkdir()*
  - *Keep Files Access Restricted*
  - *Mkdir Default*
- *money_format()*
  - *PHP 7.4 Removed Functions*
  - *PHP 8.0 Removed Functions*
- *mongo*
  - *ext/mongo*
  - *ext/mongodb*
- *mongodb*
  - *ext/mongo*
  - *ext/mongodb*
- *move_uploaded_file()*
  - *move_uploaded_file Instead Of copy*
- *msg_get_queue()*
  - *PHP Resources Turned Into Objects*
- *mt_rand()*
  - *Use random_int()*
- *mt_srand()*
  - *Use random_int()*
- *mysql_error()*
  - *ext/mysql*
  - *Don't Echo Error*
- *mysqli*
  - *ext/mysql*
  - *ext/mysqli*
  - *Use PHP Object API*

        - *Should Use Prepared Statement*
    - *mysqli_connect_errno()*
        - *ext/mysqli*
        - *Use PHP Object API*
    - *mysqli_connect_error()*
        - *ext/mysqli*
        - *Use PHP Object API*
- *N*
    - *NCURSES_COLOR_BLACK*
        - *ext/ncurses*
    - *NCURSES_COLOR_GREEN*
        - *ext/ncurses*
    - *NCURSES_COLOR_RED*
        - *ext/ncurses*
    - *NCURSES_COLOR_WHITE*
        - *ext/ncurses*
    - *NULL*
        - *Null Or Boolean Arrays*
        - *Hidden Nullable*
        - *Static Methods Can't Contain $this*
        - *$this Belongs To Classes Or Traits*
        - *$this Is Not For Static Methods*
        - *Used Static Properties*
        - *ext/eio*
        - *ext/event*
        - *ext/xmlwriter*
        - *array_key_exists() Speedup*
        - *Should Use Coalesce*
        - *Check All Types*
        - *Check JSON*
        - *Strpos()-like Comparison*
    - *Null*
        - *Null Or Boolean Arrays*
        - *Avoid Optional Properties*
        - *Scalar Or Object Property*
        - *Typehint Must Be Returned*

    - *Indices Are Int Or String*
    - *Duplicate Literal*
    - *Set Typehints*
- *NumberFormatter*
    - *ext/intl*
- *ncurses_init()*
    - *ext/ncurses*
- *ncurses_start_color()*
    - *ext/ncurses*
- *next()*
    - *Foreach Don't Change Pointer*
    - *Static Loop*
    - *Strpos()-like Comparison*
- *nl2br()*
    - *Joining file()*
- *null*
    - *Null Or Boolean Arrays*
    - *Avoid Optional Properties*
    - *Cyclic References*
    - *Don't Send $this In Constructor*
    - *Don't Unset Properties*
    - *Hidden Nullable*
    - *Insufficient Property Typehint*
    - *Make Global A Property*
    - *Mismatch Properties Typehints*
    - *Non Nullable Getters*
    - *Null On New*
    - *Parent First*
    - *Scalar Or Object Property*
    - *Should Deep Clone*
    - *Uninitialized Property*
    - *Weak Typing*
    - *Set Class Property Definition With Typehint*
    - *ext/amqp*
    - *ext/eio*
    - *ext/inotify*

* *ext/newt*
* *ext/oci8*
* *ext/sdl*
* *ext/uopz*
* *Not Definitions Only*
* *Mismatched Default Arguments*
* *Mismatch Type And Default*
* *Nullable With Constant*
* *Nullable Without Check*
* *Optional Parameter*
* *Unbinding Closures*
* *Useless Type Check*
* *Methods Without Return*
* *Assumptions*
* *Cast Unset Usage*
* *Use === null*
* *No Reference For Ternary*
* *Php 8.0 Only TypeHints*
* *Reserved Keywords In PHP 7*
* *Scalar Are Not Arrays*
* *Should Use Coalesce*
* *Use Browscap*
* *Use Nullable Type*
* *Use NullSafe Operator*
* *PSR-16 Usage*
* *PSR-7 Usage*
* *Comparison Is Always True*
* *Break With Non Integer*
* *Casting Ternary*
* *Check All Types*
* *Check JSON*
* *Constant Conditions*
* *Could Use array_fill_keys*
* *Indices Are Int Or String*
* *isset() With Constant*
* *Mismatched Ternary Alternatives*

    - *Invalid Pack Format*
    - *Pack Format Inventory*
- *parent*
    - *Abstract Or Implements*
    - *Abstract Static Methods*
    - *Cancel Common Method*
    - *Constant Used Below*
    - *Could Be Abstract Class*
    - *Could Be Parent Method*
    - *Cyclic References*
    - *Defined Class Constants*
    - *Defined Parent MP*
    - *Disconnected Classes*
    - *Fossilized Method*
    - *Identical Methods*
    - *Incompatible Signature Methods*
    - *Incompatible Signature Methods With Covariance*
    - *Is Upper Family*
    - *Locally Unused Property*
    - *Method Used Below*
    - *Mismatch Properties Typehints*
    - *Class Without Parent*
    - *self, parent, static Outside Class*
    - *Overwritten Class Const*
    - *Parent First*
    - *Never Used Properties*
    - *Property Used Above*
    - *Property Used Below*
    - *Parent, Static Or Self Outside Class*
    - *Redefined Property*
    - *Should Use Local Class*
    - *Too Many Children*
    - *Undefined Parent*
    - *Undefined static:: Or self::*
    - *Unresolved Classes*
    - *Used Protected Method*

        * *Useless Constructor*
        * *Set Parent Definition*
        * *ext/pcntl*
        * *Empty Function*
        * *Unused Arguments*
        * *Avoid Self In Interface*
        * *Repeated Interface*
        * *Unused Interfaces*
        * *Use Lower Case For Parent, Static And Self*
        * *Must Call Parent Constructor*
        * *Use Contravariance*
        * *Use Covariance*
        * *Could Use __DIR__*
        * *PHP7 Dirname*
        * *Already Parents Trait*
        * *Set Typehints*
    - *parse_ini_file()*
        * *Use Constant As Arguments*
        * *Directly Use File*
    - *parse_ini_string()*
        * *Use Constant As Arguments*
        * *Directly Use File*
    - *parse_str()*
        * *$this Belongs To Classes Or Traits*
        * *PHP 7.2 Deprecations*
        * *parse_str() Warning*
        * *Register Globals*
        * *Use Url Query Functions*
    - *parse_url()*
        * *Use Constant As Arguments*
        * *Pathinfo() Returns May Vary*
        * *Use Url Query Functions*
    - *parsekit_compile_file()*
        * *Directly Use File*
    - *parsekit_compile_string()*
        * *Directly Use File*

- *passthru()*
  - *Must Call Parent Constructor*
- *password_hash()*
  - *ext/password*
  - *Use password_hash()*
  - *Compare Hash*
- *password_verify()*
  - *Compare Hash*
- *pathinfo()*
  - *Use Constant As Arguments*
  - *Pathinfo() Returns May Vary*
  - *Use Pathinfo*
  - *Use pathinfo() Arguments*
- *pcntl_fork()*
  - *ext/pcntl*
  - *ext/proctitle*
- *pcntl_getpriority()*
  - *Strpos()-like Comparison*
- *pdo*
  - *ext/pdo*
  - *Or Die*
- *pg_result_status()*
  - *Use Constant As Arguments*
- *pg_select()*
  - *Use Constant As Arguments*
- *phar*
  - *ext/phar*
  - *Can't Disable Class*
  - *Use Basename Suffix*
- *pharexception*
  - *Could Use Try*
- *php_egg_logo_guid()*
  - *Functions Removed In PHP 5.5*
- *php_logo_guid()*
  - *Functions Removed In PHP 5.5*
- *php_real_logo_guid()*

- *preg_match_all()*
  - *preg_match_all() Flag*
  - *Regex Delimiter*
- *preg_replace()*
  - *Make One Call With Array*
  - *Processing Collector*
  - *Slow Functions*
  - *Possible Missing Subpattern*
  - *preg_replace With Option e*
  - *Regex Delimiter*
  - *Regex Inventory*
- *preg_replace_callback()*
  - *Make One Call With Array*
  - *Regex On Arrays*
  - *preg_replace With Option e*
  - *Regex Delimiter*
- *preg_replace_callback_array()*
  - *Make One Call With Array*
  - *Regex On Arrays*
  - *New Functions In PHP 7.0*
  - *preg_replace With Option e*
  - *Regex Delimiter*
- *preg_split()*
  - *Use Constant As Arguments*
  - *No mb_substr In Loop*
  - *Optimize Explode()*
- *prev()*
  - *Strpos()-like Comparison*
- *printf()*
  - *ext/ffi*
  - *Echo Or Print*
  - *Printf Number Of Arguments*
  - *Printf Format Inventory*
- *proc_nice()*
  - *New Functions In PHP 7.2*
- *proc_open()*

- *random_int()*
  - *Abstract Away*
  - *Use random_int()*
  - *New Functions In PHP 7.0*
  - *Random Without Try*
- *rdkafka*
  - *ext/rdkafka*
- *read_exif_data()*
  - *PHP 8.0 Removed Functions*
- *readdir()*
  - *Strpos()-like Comparison*
- *readfile()*
  - *Joining file()*
- *readline_info()*
  - *ext/readline*
- *recode()*
  - *Directly Use File*
- *recode_file()*
  - *Directly Use File*
- *recode_string()*
  - *Directly Use File*
- *redis*
  - *ext/redis*
  - *ext/swoole*
- *reflection*
  - *ext/reflection*
  - *Reflection Export() Is Deprecated*
- *register_shutdown_function()*
  - *Definitions Only*
  - *Callback Function Needs Return*
- *register_tick_function()*
  - *Callback Function Needs Return*
- *restore_include_path()*
  - *PHP 7.4 Removed Functions*
  - *PHP 8.0 Removed Functions*
- *round()*

- *SORT_REGULAR*
  - *Use Constant As Arguments*
- *SORT_STRING*
  - *Use Constant As Arguments*
- *SPH_MATCH_ANY*
  - *ext/sphinx*
- *SQLITE3_ASSOC*
  - *Fetch One Row Format*
- *SQLITE3_BOTH*
  - *Fetch One Row Format*
- *SQLITE3_NUM*
  - *Fetch One Row Format*
- *SQLite3*
  - *ext/sqlite3*
  - *Queries In Loops*
- *SVM*
  - *ext/svm*
- *SWFAction*
  - *ext/ming*
- *SWFMovie*
  - *ext/ming*
- *SWFShape*
  - *ext/ming*
- *SWFSprite*
  - *ext/ming*
- *SeasLog*
  - *ext/seaslog*
- *Seaslog*
  - *ext/seaslog*
- *Self*
  - *Avoid Self In Interface*
- *SimpleXMLElement*
  - *ext/simplexml*
- *SoapClient*
  - *ext/soap*
- *Sphinx*

    - *ext/sphinx*
- *SplFileObject*
    - *Must Call Parent Constructor*
- *SplQueue*
    - *PHP 7.2 Scalar Typehints*
- *SplTempFileObject*
    - *Must Call Parent Constructor*
- *Sqlite3*
    - *ext/sqlite3*
    - *Fetch One Row Format*
    - *Set Aside Code*
- *Static*
    - *Abstract Static Methods*
    - *Assign Default To Properties*
    - *Non Static Methods Called In A Static*
    - *Normal Methods*
    - *Parent, Static Or Self Outside Class*
    - *Should Use Local Class*
    - *Static Methods Can't Contain $this*
    - *Static Methods Called From Object*
    - *$this Is Not For Static Methods*
    - *Wrong Access Style to Property*
    - *Undefined static:: Or self::*
    - *Set Class Method Remote Definition*
    - *Collect Local Variable Counts*
    - *ext/reflection*
    - *Closure Could Be A Callback*
    - *Function With Dynamic Code*
    - *No Reference For Static Property*
    - *Static Loop*
    - *Variable May Be Non-Global*
    - *Should Be Single Quote*
    - *Real Variables*
- *StdClass*
    - *PHP 7.2 Scalar Typehints*
    - *Array_Fill() With Objects*

- *Stdclass*
  - *Is An Extension Class*
  - *Should Deep Clone*
  - *Unresolved Catch*
  - *Global Import*
  - *array_key_exists() Works On Arrays*
  - *Could Use array_fill_keys*
  - *Missing Parenthesis*
  - *Avoid get_class()*
- *Strtr()*
  - *Strtr Arguments*
- *Substr()*
  - *Drop Substr Last Arg*
- *Switch()*
  - *Missing Cases In Switch*
- *scandir()*
  - *Use Constant As Arguments*
  - *Avoid glob() Usage*
- *seaslog*
  - *ext/seaslog*
- *self*
  - *Abstract Static Methods*
  - *Constant Class*
  - *Constant Used Below*
  - *Could Be Private Class Constant*
  - *Could Be Protected Class Constant*
  - *Method Could Be Static*
  - *Defined Class Constants*
  - *Defined static:: Or self::*
  - *Is Not Class Family*
  - *Non Static Methods Called In A Static*
  - *self, parent, static Outside Class*
  - *No Self Referencing Constant*
  - *Overwritten Class Const*
  - *Property Used In One Method Only*
  - *Parent, Static Or Self Outside Class*

- *socket_addrinfo_bind()*
  - *PHP Resources Turned Into Objects*
- *socket_addrinfo_connect()*
  - *PHP Resources Turned Into Objects*
- *socket_connect()*
  - *ext/sockets*
- *socket_create()*
  - *ext/sockets*
  - *PHP Resources Turned Into Objects*
- *socket_create_listen()*
  - *PHP Resources Turned Into Objects*
- *socket_import_stream()*
  - *PHP Resources Turned Into Objects*
- *socket_last_error()*
  - *ext/sockets*
- *socket_read()*
  - *Use Constant As Arguments*
- *socket_wsaprotocol_info_import()*
  - *PHP Resources Turned Into Objects*
- *sort()*
  - *Use Constant As Arguments*
- *sphinx*
  - *ext/sphinx*
- *spl_autoload_register()*
  - *Definitions Only*
- *split()*
  - *Optimize Explode()*
  - *PHP 7.0 Removed Functions*
- *spliti()*
  - *PHP 7.0 Removed Functions*
- *sprintf()*
  - *Printf Format Inventory*
- *sql_regcase()*
  - *PHP 7.0 Removed Functions*
- *sqlite3*
  - *ext/sqlite3*

    - *Sqlite3 Requires Single Quotes*
    - *Set Aside Code*
- *sqlsrv_errors()*
    - *ext/sqlsrv*
- *srand()*
    - *Use random_int()*
- *static*
    - *Abstract Static Methods*
    - *Ambiguous Static*
    - *Cant Instantiate Class*
    - *Class Usage*
    - *Method Could Be Static*
    - *Defined Parent MP*
    - *Defined static:: Or self::*
    - *Dependant Abstract Classes*
    - *No Direct Call To Magic Method*
    - *Don't Send $this In Constructor*
    - *Don't Unset Properties*
    - *Dynamic Classes*
    - *Dynamic Methodcall*
    - *Dynamic Property*
    - *Is Not Class Family*
    - *Is Upper Family*
    - *Forgotten Visibility*
    - *Non Static Methods Called In A Static*
    - *self, parent, static Outside Class*
    - *Normal Methods*
    - *Only Static Methods*
    - *Order Of Declaration*
    - *Overwritten Class Const*
    - *Property Names*
    - *Property Used In One Method Only*
    - *Parent, Static Or Self Outside Class*
    - *Should Use Local Class*
    - *Static Methods Can't Contain $this*
    - *Static Methods*

* *Static Methods Called From Object*
* *Static Properties*
* *$this Belongs To Classes Or Traits*
* *$this Is Not For Static Methods*
* *Too Many Dereferencing*
* *Magic Visibility*
* *Wrong Access Style to Property*
* *Undefined static:: Or self::*
* *Unused Private Methods*
* *Unused Private Properties*
* *Use ::Class Operator*
* *Used Classes*
* *Used Private Methods*
* *Used Static Properties*
* *Useless Abstract Class*
* *Use This*
* *Using $this Outside A Class*
* *Propagate Calls*
* *SetA rray Class Definition*
* *Set String Method Definition*
* *Constant Dynamic Creation*
* *Constant Order*
* *ext/ffi*
* *ext/libevent*
* *ext/reflection*
* *ext/xdebug*
* *Not Definitions Only*
* *Cannot Use Static For Closure*
* *Closure Could Be A Callback*
* *Could Be Typehinted Callable*
* *Could Be Static Closure*
* *Mismatch Type And Default*
* *Modified Typed Parameter*
* *No Literal For Reference*
* *No Return Used*
* *Only Variable For Reference*

* *Only Variable Passed By Reference*
* *Too Many Local Variables*
* *Unbinding Closures*
* *Wrong Type Returned*
* *An OOP Factory*
* *Use PHP7 Encapsed Strings*
* *Cant Use Return Value In Write Context*
* *Use Lower Case For Parent, Static And Self*
* *Detect Current Class*
* *No Reference For Static Property*
* *Only Container For Reference*
* *Php 8.0 Variable Syntax Tweaks*
* *Reserved Match Keyword*
* *Dynamic Library Loading*
* *No Net For Xml Load*
* *Constant Conditions*
* *Constant Scalar Expressions*
* *Could Be Static*
* *Declare Static Once*
* *Dynamic Calls*
* *Foreach Needs Reference Array*
* *Avoid Large Array Assignation*
* *No Hardcoded Hash*
* *No Need For get_class()*
* *Only Variable Returned By Reference*
* *Static Global Variables Confusion*
* *Static Loop*
* *Useless Unset*
* *Variable May Be Non-Global*
* *Dependant Trait*
* *No Static Variable In A Method*
* *Real Variables*
* *Static Variables*
* *Undefined Variable*
* *Used Once Variables*
* *FuelPHP Usage*

        - *ext/tidy*
    - *TokyoTyrant*
        - *ext/tokyotyrant*
    - *Traversable*
        - *Cant Implement Traversable*
    - *throwable*
        - *Can't Throw Throwable*
    - *tidy*
        - *ext/tidy*
        - *Use PHP Object API*
    - *time()*
        - *Conditioned Constants*
        - *ext/zip*
        - *time() Vs strtotime()*
        - *Use random_int()*
        - *Date Formats*
        - *Session Variables*
        - *Use Cookies*
        - *Should Use SetCookie()*
        - *Set Cookie Safe Arguments*
        - *Timestamp Difference*
    - *token_get_all()*
        - *@ Operator*
    - *traversable*
        - *Cant Implement Traversable*
        - *Could Be Iterable*
    - *trigger_error()*
        - *Use Constant As Arguments*
        - *Trigger Errors*
    - *trim()*
        - *Substring First*
        - *Substr To Trim*

- ***U***
    - *Unset()*
        - *Unset() Or (unset)*
        - *Multiple Unset()*

- *Usort()*
  - *Usort Sorting In PHP 7.0*
- *uasort()*
  - *Slow Functions*
  - *Usort Sorting In PHP 7.0*
- *uksort()*
  - *Slow Functions*
  - *Usort Sorting In PHP 7.0*
- *uniqid()*
  - *ext/eio*
  - *Use random_int()*
- *unpack()*
  - *Invalid Pack Format*
  - *Pack Format Inventory*
- *unserialize()*
  - *Unserialize Second Arg*
- *unset()*
  - *Unset() Or (unset)*
  - *Multiple Unset()*
- *urlencode()*
  - *Use Url Query Functions*
- *usleep()*
  - *Avoid sleep()/usleep()*
- *usort()*
  - *Slow Functions*
- *uuid_create()*
  - *ext/uuid*

- *V*
  - *V8Js*
    - *ext/v8js*
  - *V8JsException*
    - *ext/v8js*
  - *VARNISH_CONFIG_HOST*
    - *ext/varnish*
  - *VARNISH_CONFIG_PORT*
    - *ext/varnish*

        - *ext/xsl*
    - *xdebug_call_class()*
        - *ext/xdebug*
    - *xdebug_call_file()*
        - *ext/xdebug*
    - *xdebug_call_function()*
        - *ext/xdebug*
    - *xdebug_call_line()*
        - *ext/xdebug*
    - *xhprof_disable()*
        - *ext/xhprof*
    - *xml_parser_create()*
        - *ext/xml*
        - *PHP Resources Turned Into Objects*
    - *xml_parser_create_ns()*
        - *PHP Resources Turned Into Objects*
- ***Y***
    - *yaml_parse()*
        - *Directly Use File*
    - *yaml_parse_file()*
        - *Directly Use File*
- ***Z***
    - *ZBarCodeImage*
        - *ext/zbarcode*
    - *ZBarCodeScanner*
        - *ext/zbarcode*
    - *ZMQ*
        - *ext/0mq*
    - *ZMQDevice*
        - *ext/0mq*
    - *ZipArchive*
        - *ext/zip*
    - *ZooKeeper*
        - *ext/zookeeper*
    - *Zookeeper*
        - *PHP Exception*

        - *ext/zookeeper*
    - *ZookeeperException*
        - *PHP Exception*
    - *zend_logo_guid()*
        - *Functions Removed In PHP 5.5*
    - *zmq*
        - *ext/0mq*
    - *zookeeper*
        - *ext/zookeeper*
- _
    - *_()*
        - *ext/gettext*
    - *__CLASS__*
        - *Undefined Properties*
        - *Detect Current Class*
        - *::class*
        - *Interpolation*
        - *Non Ascii Variables*
    - *__DIR__*
        - *ext/wasm*
        - *Use PHP7 Encapsed Strings*
        - *Could Use __DIR__*
        - *__DIR__ Then Slash*
        - *No Hardcoded Path*
        - *PHP Sapi*
    - *__FILE__*
        - *Magic Constant Usage*
        - *ext/cairo*
        - *ext/fann*
        - *ext/grpc*
        - *ext/inotify*
        - *ext/sem*
        - *__halt_compiler*
        - *Could Use __DIR__*
        - *No Hardcoded Path*
    - *__FUNCTION__*

    - *Use Const And Functions*
    - *PHP Overridden Function*
- *__LINE__*
    - *Magic Constant Usage*
- *__METHOD__*
    - *Anonymous Classes*
    - *Class Usage*
    - *Non Static Methods Called In A Static*
    - *Should Have Destructor*
    - *Already Parents Interface*
- *__call*
    - *Check On __Call Usage*
    - *No Direct Call To Magic Method*
    - *Has Magic Property*
    - *Magic Methods*
    - *$this Belongs To Classes Or Traits*
    - *Useless Typehint*
    - *Must Return Methods*
- *__callStatic*
    - *Has Magic Property*
    - *Magic Methods*
    - *Must Return Methods*
- *__clone*
    - *No Direct Call To Magic Method*
    - *Has Magic Property*
    - *Magic Methods*
    - *Should Deep Clone*
    - *Magic Visibility*
    - *Set Clone Link*
    - *Direct Call To __clone()*
- *__construct*
    - *Anonymous Classes*
    - *Avoid Optional Properties*
    - *Cant Instantiate Class*
    - *Constructors*
    - *Don't Send $this In Constructor*

* *Has Magic Property*
* *Insufficient Property Typehint*
* *Assign Default To Properties*
* *Make Global A Property*
* *Non Nullable Getters*
* *Old Style Constructor*
* *Parent First*
* *Property Could Be Local*
* *Redefined Default*
* *Scalar Or Object Property*
* *Should Deep Clone*
* *Should Have Destructor*
* *Should Use Local Class*
* *Strange Names For Methods*
* *Throw In Destruct*
* *Too Many Injections*
* *DI Cyclic Dependencies*
* *Uninitialized Property*
* *Unitialized Properties*
* *Useless Constructor*
* *Illegal Name For Method*
* *Wrong Typed Property Default*
* *Create Default Values*
* *Set Class Method Remote Definition*
* *Collect Method Counts*
* *Could Be Static Closure*
* *Useless Return*
* *Courier Anti-Pattern*
* *Dependency Injection*
* *Could Use Promoted Properties*
* *Must Call Parent Constructor*
* *Signature Trailing Comma*
* *Typed Property Usage*
* *Avoid Large Array Assignation*
* *Set Aside Code*
* *Should Chain Exception*

13.5 Directory by PHP Error message

Exakat helps reduce the amount of error and warning that code is producing by reporting pattern that are likely to emit errors.

133 PHP error message detailled :

- *“continue” targeting switch is equivalent to “break”. Did you mean to use “continue 2”?*
- *A function with return type must return a value (did you mean “return null;” instead of “return;”?)*
- *Access level to Bar::$publicProperty must be public (as in class Foo)*
- *Access level to c::iPrivate() must be public (as in class i)*
- *Access level to x::foo() must be public (as in class i)*
- *Access level to xx::$x must be public (as in class x)*
- *Access to undeclared static property*
- *Access to undeclared static property: x::$y*
- *Accessing static property aa::$a as non static*
- *An alias (%s) was defined for method %s(), but this method does not exist*
- *Argument #1 ($line) must be passed by reference*
- *Argument 1 passed to foo() must be of the type integer, string given*
- *Argument cannot be passed by reference*
- *Argument cannot be passed by reference*
- *Argument cannot be passed by reference*
- *Argument must be of type int, array given*
- *Array and string offset access syntax with curly braces is deprecated*
- *Call to a member function m() on null*
- *Call to private Y::__construct() from invalid context*
- *Call to undefined function*
- *Call to undefined method theParent::bar()*
- *Call to undefined method x::y()*
- *Can’t inherit abstract function A::bar()*
- *Cannot access parent:: when current class scope has no parent*
- *Cannot access parent:: when current class scope has no parent*
- *Cannot access parent:: when current class scope has no parent*
- *Cannot access private const*
- *Cannot access static:: when no class scope is active*
- *Cannot bind an instance to a static closure*
- *Cannot inherit previously-inherited or override constant A from interface i*
- *Cannot override final method Foo::Bar()*
- *Cannot override final method Foo::FooBar()*

- *Cannot pass parameter 1 by reference*
- *Cannot pass parameter 1 by reference*
- *Cannot pass parameter 1 by reference*
- *Cannot perform bitwise not on array*
- *Cannot perform bitwise not on bool*
- *Cannot perform bitwise not on object*
- *Cannot perform bitwise not on resource*
- *Cannot unpack array with string keys*
- *Cannot use "parent" when no class scope is active*
- *Cannot use "self" when no class scope is active*
- *Cannot use "static" when no class scope is active*
- *Cannot use a scalar value as an array*
- *Cannot use isset() on the result of an expression (you can use "null !== expression" instead)*
- *Cannot use lexical variable $b as a parameter name*
- *Cannot use object of type Foo as array*
- *Cannot use parent when current class scope has no parent*
- *Case-insensitive constants are deprecated. The correct casing for this constant is "A"*
- *Class 'PARENT' not found*
- *Class 'x' not found*
- *Class BA contains 1 abstract method and must therefore be declared abstract or implement the remaining methods (A::aFoo)*
- *Class b cannot implement previously implemented interface i*
- *Class b cannot implement previously implemented interface i*
- *Class c contains 1 abstract method and must therefore be declared abstract or implement the remaining methods (a::foo)*
- *Class fooThrowable cannot implement interface Throwable, extend Exception or Error instead*
- *Class x contains 2 abstract methods and must therefore be declared abstract or implement the remaining methods (x::m1, x::m2)*
- *Class x must implement interface Traversable as part of either Iterator or IteratorAggregate*
- *Could not check compatibility between xx::bar(B $a) and foo::bar(A $a), because class A is not available*
- *Creating default object from empty value*
- *Declaration of FooParent::Bar() must be compatible with FooChildren::Bar()*
- *Declaration of a::foo($a) should be compatible with ab1::foo($a)*
- *Declaration of ab::foo($a) must be compatible with a::foo($a = 1)*
- *Declaration of ab::foo($a) must be compatible with a::foo($a = 1)*
- *Declaration of ab::foo($a) should be compatible with a::foo($a = 1)*
- *Declaration of ab::foo($a) should be compatible with a::foo($a = 1)*

- *Default value for parameters with a int type can only be int or NULL*
- *Defining a custom assert() function is deprecated, as the function has special semantics*
- *Delimiter must not be alphanumeric or backslash*
- *Deprecated: Required parameter $y follows optional parameter $x*
- *Generators cannot return values using "return"*
- *Generators cannot return values using "return"*
- *Headers already sent*
- *Indirect modification of overloaded property c::$b has no effect*
- *Invalid numeric literal*
- *Method name must be a string*
- *Methods with the same name as their class will not be constructors in a future version of PHP; %s has a deprecated constructor*
- *Non-static method A::B() should not be called statically*
- *Octal escape sequence overflow 500 is greater than 377*
- *Old style constructors are DEPRECATED in PHP 7.0, and will be removed in a future version. You should always use __construct() in new code.*
- *Only variable references should be returned by reference*
- *Only variable references should be returned by reference*
- *Only variables can be passed by reference*
- *Only variables can be passed by reference*
- *Only variables should be passed by reference*
- *Private methods cannot be final as they are never overridden by other classes*
- *Redefinition of parameter $b*
- *Return value of foo() must be an instance of Bar, none returned*
- *Return value of foo() must be of the type int, string returned*
- *The (real) cast is deprecated, use (float) instead*
- *The behavior of unparenthesized expressions containing both '.' and '+'/'-' will change in PHP 8: '+'/'-' will take a higher precedence*
- The behavior of unparenthesized expressions containing both '.' and '>>'/'
- *The each() function is deprecated. This message will be suppressed on further calls*
- *The parent constructor was not called: the object is in an invalid state*
- *Too few arguments to function Foo::Bar(), 1 passed*
- *Too few arguments to function foo(), 1 passed and exactly 2 expected*
- *Too few arguments to function foo(), 1 passed and exactly 2 expected*
- *Trait 'T' not found*
- *Trait 'a' not found*
- *Trait method M has not been applied, because there are collisions with other trait methods on C*

- *Trait method f has not been applied, because there are collisions with other trait methods on x*
- *Trying to access array offset on value of type boolean*
- *Trying to access array offset on value of type float*
- *Trying to access array offset on value of type int*
- *Trying to access array offset on value of type null*
- *Trying to access array offset on value of type null*
- *Uncaught ArgumentCountError: Too few arguments to function, 0 passed*
- *Undefined class constant*
- *Undefined constant 'A'*
- *Undefined constant 'y'*
- *Undefined function*
- *Undefined property: x::$e*
- *Undefined variable:*
- *Unknown named parameter $d in*
- *Unparenthesized a ? b : c ? d : e is deprecated. Use either (a ? b : c) ? d : e or a ? b : (c ? d : e)*
- *Unsupported operand types*
- *Use of undefined constant y - assumed 'y' (this will throw an Error in a future version of PHP)*
- *Using $this when not in object context*
- *Using $this when not in object context*
- *Using $this when not in object context*
- *__autoload() is deprecated, use spl_autoload_register() instead*
- *__clone method called on non-object*
- *array_merge() expects at least 1 parameter, 0 given*
- *b cannot implement a - it is not an interface*
- *define(): Declaration of case-insensitive constants is deprecated*
- *iconv(): Wrong charset, conversion from UTF-8' to ASCII//TRANSLIT' is not allowed*
- *include(a.php): failed to open stream: No such file or directory*
- *pack(): Type t: unknown format code*
- *printf(): Too few arguments*
- *syntax error, unexpected ','*
- *syntax error, unexpected '-', expecting '='*
- *syntax error, unexpected 'match'*
- *unpack(): Type t: unknown format code*

CHAPTER 14

Rulesets

14.1 Introduction

Exakat provides unique 1371 rules to detect BUGS, CODE SMELLS, SECURITY OR QUALITY ISSUES in your PHP code.

For more smoothly usage, the ruleset concept allow you to run a set of rules based on a decidated focus. Beawre that a Ruleset run all the associated rules and any needed dependencies.

Rulesets are configured with the -T option, when running exakat in command line. For example :

```
php exakat.phar analyze -p <project> -T <Security>
```

14.2 Summary

Here is the list of the current rulesets supported by Exakat Engine.

Name	Description
Analyze	Check for common best practices.
Attributes	This ruleset gathers all rules that rely on PHP 8.0 attributes.
CE	List of rules that are part of the Community Edition
CI-checks	Quick check for common best practices.
ClassReview	A set of rules dedicate to class hygiene
Coding conventions	List coding conventions violations.
Compatibility-PHP53	List features that are incompatible with PHP 5.3.
Compatibility-PHP54	List features that are incompatible with PHP 5.4.
Compatibility-PHP55	List features that are incompatible with PHP 5.5.
Compatibility-PHP56	List features that are incompatible with PHP 5.6.
Compatibility-PHP70	List features that are incompatible with PHP 7.0.
Compatibility-PHP71	List features that are incompatible with PHP 7.1.
Compatibility-PHP72	List features that are incompatible with PHP 7.2.
Compatibility-PHP73	List features that are incompatible with PHP 7.3.
Compatibility-PHP74	List features that are incompatible with PHP 7.4.
Compatibility-PHP80	List features that are incompatible with PHP 8.0.
Compatibility-PHP81	List features that are incompatible with PHP 8.1.
Dead code	Check the unused code or unreachable code.
LintButWontExec	Check the code for common errors that will lead to a Fatal error on production, but lint fine.
Performances	Check the code for slow code.
Rector	Suggests configuration to apply changes with Rector
Security	Check the code for common security bad practices, especially in the Web environnement.
Semantics	Checks the meanings found the names of the code.
Suggestions	List of possible modernisation of the PHP code.
Top10	The most common issues found in the code
Typechecks	Checks related to types.
php-cs-fixable	Suggests configuration to apply changes with PHP-CS-FIXER

Note : in command line, don't forget to add quotes to rulesets' names that include white space.

14.3 List of rulesets

14.3.1 Analyze

This ruleset centralizes a large number of classic trap and pitfalls when writing PHP.

Total : 419 analysis

- *Adding Zero*
- *Ambiguous Array Index*
- *Multiple Index Definition*
- *Empty Classes*
- *Forgotten Visibility*
- *Non Static Methods Called In A Static*
- *Old Style Constructor*
- *Static Methods Called From Object*
- *Constants With Strange Names*
- *Empty Function*
- *Redeclared PHP Functions*
- *Methods Without Return*
- *Empty Interfaces*
- *Incompilable Files*
- *error_reporting() With Integers*
- *Eval() Usage*
- *Exit() Usage*
- *Forgotten Whitespace*
- *Iffectations*
- *Multiply By One*
- *@ Operator*
- *Not Not*
- *include_once() Usage*
- *Strpos()-like Comparison*
- *Throws An Assignement*
- *var_dump()... Usage*
- *__toString() Throws Exception*
- *Non Ascii Variables*
- *Used Once Variables*
- *Bad Constants Names*
- *Empty Traits*
- *Use With Fully Qualified Name*
- *Useless Instructions*
- *Abstract Static Methods*
- *Invalid Constant Name*
- *Multiple Constant Definition*

- *Wrong Optional Parameter*
- *Use === null*
- *$this Is Not An Array*
- *One Variable String*
- *Static Methods Can't Contain $this*
- *While(List() = Each())*
- *Several Instructions On The Same Line*
- *Multiples Identical Case*
- *Switch Without Default*
- *$this Belongs To Classes Or Traits*
- *Nested Ternary*
- *Non-constant Index In Array*
- *Undefined Constants*
- *Instantiating Abstract Class*
- *Class, Interface Or Trait With Identical Names*
- *Empty Try Catch*
- *Undefined Classes*
- *Htmlentities Calls*
- *Undefined Class Constants*
- *Used Once Variables (In Scope)*
- *Undefined Functions*
- *Deprecated PHP Functions*
- *Dangling Array References*
- *Queries In Loops*
- *Var Keyword*
- *Aliases Usage*
- *Uses Default Values*
- *Wrong Number Of Arguments*
- *Hardcoded Passwords*
- *Unresolved Classes*
- *Useless Constructor*
- *Implement Is For Interface*
- *Use const*
- *Unresolved Use*
- *Undefined Parent*
- *Undefined static:: Or self::*

- *Accessing Private*
- *Access Protected Structures*
- *Parent, Static Or Self Outside Class*
- *list() May Omit Variables*
- *Or Die*
- *Written Only Variables*
- *Must Return Methods*
- *Empty Instructions*
- *Overwritten Exceptions*
- *Foreach Reference Is Not Modified*
- *Don't Change Incomings*
- *Compared Comparison*
- *Useless Return*
- *Unused Classes*
- *Unpreprocessed Values*
- *Undefined Properties*
- *Short Open Tags*
- *Strict Comparison With Booleans*
- *Lone Blocks*
- *$this Is Not For Static Methods*
- *Global Usage*
- *PHP Keywords As Names*
- *Logical Should Use Symbolic Operators*
- *Could Use self*
- *Catch Overwrite Variable*
- *Deep Definitions*
- *Repeated print()*
- *Avoid Parenthesis*
- *Objects Don't Need References*
- *Lost References*
- *Constants Created Outside Its Namespace*
- *Fully Qualified Constants*
- *Never Used Properties*
- *No Real Comparison*
- *Should Use Local Class*
- *No Direct Call To Magic Method*

- *String May Hold A Variable*
- *Echo With Concat*
- *Unused Global*
- *Useless Global*
- *Preprocessable*
- *Useless Final*
- *Use Constant*
- *Useless Unset*
- *Buried Assignation*
- *No array_merge() In Loops*
- *Useless Parenthesis*
- *Unresolved Instanceof*
- *Use PHP Object API*
- *Unthrown Exception*
- *Old Style __autoload()*
- *Altering Foreach Without Reference*
- *Use Pathinfo*
- *Should Use Constants*
- *Hash Algorithms*
- *No Parenthesis For Language Construct*
- *No Hardcoded Path*
- *No Hardcoded Port*
- *Use Constant As Arguments*
- *Implied If*
- *Overwritten Literals*
- *Assign Default To Properties*
- *No Public Access*
- *Should Chain Exception*
- *Useless Interfaces*
- *Undefined Interfaces*
- *Concrete Visibility*
- *Double Instructions*
- *Should Use Prepared Statement*
- *Print And Die*
- *Unchecked Resources*
- *No Hardcoded Ip*

- *Else If Versus Elseif*
- *Unset In Foreach*
- *Could Be Static*
- *Multiple Class Declarations*
- *Empty Namespace*
- *Could Use Short Assignation*
- *Useless Abstract Class*
- *Static Loop*
- *Pre-increment*
- *Only Variable Returned By Reference*
- *Indices Are Int Or String*
- *Should Typecast*
- *No Self Referencing Constant*
- *No Direct Usage*
- *Break Outside Loop*
- *Avoid Substr() One*
- *Double Assignation*
- *Empty List*
- *Useless Brackets*
- *preg_replace With Option e*
- *eval() Without Try*
- *Relay Function*
- *func_get_arg() Modified*
- *Avoid get_class()*
- *Silently Cast Integer*
- *Timestamp Difference*
- *Unused Arguments*
- *Switch To Switch*
- *Wrong Parameter Type*
- *Redefined Class Constants*
- *Redefined Default*
- *Wrong fopen() Mode*
- *Negative Power*
- *Already Parents Interface*
- *Use random_int()*
- *Can't Extend Final*

- *Ternary In Concat*
- *Using $this Outside A Class*
- *Undefined Trait*
- *No Hardcoded Hash*
- *Identical Conditions*
- *Unkown Regex Options*
- *No Choice*
- *Common Alternatives*
- *Logical Mistakes*
- *Uncaught Exceptions*
- *Same Conditions In Condition*
- *Return True False*
- *Useless Switch*
- *Could Use __DIR__*
- *Should Use Coalesce*
- *Make Global A Property*
- *If With Same Conditions*
- *Throw Functioncall*
- *Use Instanceof*
- *Results May Be Missing*
- *Always Positive Comparison*
- *Empty Blocks*
- *Throw In Destruct*
- *Use System Tmp*
- *Dependant Trait*
- *Hidden Use Expression*
- *Should Make Alias*
- *Multiple Identical Trait Or Interface*
- *Multiple Alias Definitions*
- *Nested Ifthen*
- *Cast To Boolean*
- *Failed Substr Comparison*
- *Should Make Ternary*
- *Unused Returned Value*
- *Modernize Empty With Expression*
- *Use Positive Condition*

- *Drop Else After Return*
- *Use ::Class Operator*
- *Don't Echo Error*
- *Useless Type Casting*
- *No isset() With empty()*
- *Useless Check*
- *Bail Out Early*
- *Dont Change The Blind Var*
- *Avoid Using stdClass*
- *Too Many Local Variables*
- *Illegal Name For Method*
- *Class Should Be Final By Ocramius*
- *Long Arguments*
- *Assigned Twice*
- *No Boolean As Default*
- *Forgotten Thrown*
- *Multiple Alias Definitions Per File*
- *__DIR__ Then Slash*
- *self, parent, static Outside Class*
- *Used Once Property*
- *Property Used In One Method Only*
- *No Need For Else*
- *Strange Name For Variables*
- *Strange Name For Constants*
- *Too Many Finds*
- *Should Use SetCookie()*
- *Check All Types*
- *Missing Cases In Switch*
- *Repeated Regex*
- *No Class In Global*
- *Crc32() Might Be Negative*
- *Could Use str_repeat()*
- *Suspicious Comparison*
- *Strings With Strange Space*
- *No Empty Regex*
- *Alternative Syntax Consistence*

- *Randomly Sorted Arrays*
- *Only Variable Passed By Reference*
- *No Return Used*
- *No Reference On Left Side*
- *Implemented Methods Are Public*
- *Mixed Concat And Interpolation*
- *Too Many Injections*
- *Could Make A Function*
- *Forgotten Interface*
- *Avoid Optional Properties*
- *Mismatched Ternary Alternatives*
- *Mismatched Default Arguments*
- *Mismatched Typehint*
- *Scalar Or Object Property*
- *Assign With And Precedence*
- *No Magic Method With Array*
- *Logical To in_array*
- *Pathinfo() Returns May Vary*
- *Multiple Type Variable*
- *Is Actually Zero*
- *Unconditional Break In Loop*
- *Could Be Else*
- *Next Month Trap*
- *Printf Number Of Arguments*
- *Ambiguous Static*
- *Don't Send $this In Constructor*
- *No get_class() With Null*
- *Maybe Missing New*
- *Unknown Pcre2 Option*
- *Parent First*
- *Invalid Regex*
- *Use Named Boolean In Argument Definition*
- *Same Variable Foreach*
- *Never Used Parameter*
- *Identical On Both Sides*
- *Identical Consecutive Expression*

- *No Reference For Ternary*
- *Unused Inherited Variable In Closure*
- *Inclusion Wrong Case*
- *Missing Include*
- *Useless Referenced Argument*
- *Useless Catch*
- *Possible Infinite Loop*
- *Test Then Cast*
- *Foreach On Object*
- *Property Could Be Local*
- *Too Many Native Calls*
- *Redefined Private Property*
- *Don't Unset Properties*
- *Strtr Arguments*
- *Missing Parenthesis*
- *Callback Function Needs Return*
- *Wrong Range Check*
- *Cant Instantiate Class*
- *strpos() Too Much*
- *Typehinted References*
- *Weak Typing*
- *Method Signature Must Be Compatible*
- *Mismatch Type And Default*
- *Check JSON*
- *Dont Mix ++*
- *Can't Throw Throwable*
- *Abstract Or Implements*
- *Incompatible Signature Methods*
- *Ambiguous Visibilities*
- *Undefined ::class*
- *Assert Function Is Reserved*
- *Could Be Abstract Class*
- *Continue Is For Loop*
- *Must Call Parent Constructor*
- *Undefined Variable*
- *Undefined Insteadof*

- *Method Collision Traits*
- *Class Could Be Final*
- *Inconsistent Elseif*
- *Only Variable For Reference*
- *Wrong Access Style to Property*
- *Invalid Pack Format*
- *Repeated Interface*
- *Don't Read And Write In One Expression*
- *Should Yield With Key*
- *Useless Alias*
- *Method Could Be Static*
- *Possible Missing Subpattern*
- *Assign And Compare*
- *Variable Is Not A Condition*
- *Insufficient Typehint*
- *Typehint Must Be Returned*
- *Clone With Non-Object*
- *Check On __Call Usage*
- *Avoid option arrays in constructors*
- *Already Parents Trait*
- *Trait Not Found*
- *Casting Ternary*
- *Concat Empty String*
- *Concat And Addition*
- *No Append On Source*
- *Memoize MagicCall*
- *Unused Class Constant*
- *Infinite Recursion*
- *Null Or Boolean Arrays*
- *Dependant Abstract Classes*
- *Wrong Type Returned*
- *Overwritten Source And Value*
- *Avoid mb_dectect_encoding()*
- *array_key_exists() Works On Arrays*
- *Class Without Parent*
- *Scalar Are Not Arrays*

- *array_merge() And Variadic*
- *Implode() Arguments Order*
- *strip_tags Skips Closed Tag*
- *No Spread For Hash*
- *Max Level Of Nesting*
- *Should Use Explode Args*
- *Use array_slice()*
- *Too Many Array Dimensions*
- *Coalesce And Concat*
- *Comparison Is Always True*
- *Incompatible Signature Methods With Covariance*
- *Interfaces Is Not Implemented*
- *No Literal For Reference*
- *Interfaces Don't Ensure Properties*
- *Non Nullable Getters*
- *Too Many Dereferencing*
- *Cant Implement Traversable*
- *Is_A() With String*
- *Mbstring Unknown Encoding*
- *Mbstring Third Arg*
- *Merge If Then*
- *Wrong Type With Call*
- *Not Equal Is Not !==*
- *Dont Collect Void*
- *Wrong Typed Property Default*
- *Hidden Nullable*
- *Fn Argument Variable Confusion*
- *Missing Abstract Method*
- *Undefined Constant Name*
- *Using Deprecated Method*
- *Cyclic References*
- *Double Object Assignation*
- *Wrong Argument Type*
- *Mismatch Properties Typehints*
- *No Need For Triple Equal*
- *Array_merge Needs Array Of Arrays*

- *Wrong Type For Native PHP Function*
- *Catch Undefined Variable*
- *Swapped Arguments*
- *Different Argument Counts*
- *Unknown Parameter Name*
- *Missing Some Returntype*
- *Don't Pollute Global Space*
- *Mismatch Parameter Name*
- *Multiple Declaration Of Strict_types*
- *Mismatch Parameter And Type*
- *Array_Fill() With Objects*
- *Modified Typed Parameter*
- *Assumptions*
- *Unsupported Types With Operators*
- *Could Be Stringable*
- *Wrong Attribute Configuration*
- *Cancelled Parameter*
- *Constant Typo Looks Like A Variable*
- *Array_Map() Passes By Value*
- *Missing __isset() Method*
- *Modify Immutable*
- *Only Container For Reference*
- *Cannot Use Static For Closure*

14.3.2 Attributes

This ruleset gathers all rules that rely on PHP 8.0 attributes.

Total : 3 analysis

- *Exit-like Methods*
- *Using Deprecated Method*
- *Modify Immutable*

14.3.3 CE

This ruleset is the Community Edition list. It holds all the analysis that are in the community edition version of Exakat.

Total : 458 analysis

- *Adding Zero*
- *Array Index*

- *Multidimensional Arrays*
- *PHP Arrays Index*
- *Classes Names*
- *Constant Definition*
- *Magic Methods*
- *Forgotten Visibility*
- *Old Style Constructor*
- *Static Methods*
- *Static Methods Called From Object*
- *Static Properties*
- *Constants Usage*
- *Constants Names*
- *Magic Constant Usage*
- *PHP Constant Usage*
- *Defined Exceptions*
- *Thrown Exceptions*
- *ext/apc*
- *ext/bcmath*
- *ext/bzip2*
- *ext/calendar*
- *ext/crypto*
- *ext/ctype*
- *ext/curl*
- *ext/date*
- *ext/dba*
- *ext/dom*
- *ext/enchant*
- *ext/ereg*
- *ext/exif*
- *ext/fdf*
- *ext/fileinfo*
- *ext/filter*
- *ext/ftp*
- *ext/gd*
- *ext/gmp*
- *ext/gnupgp*

- *ext/hash*
- *ext/iconv*
- *ext/json*
- *ext/kdm5*
- *ext/ldap*
- *ext/libxml*
- *ext/mbstring*
- *ext/mcrypt*
- *ext/mongo*
- *ext/mssql*
- *ext/mysql*
- *ext/mysqli*
- *ext/odbc*
- *ext/openssl*
- *ext/pcre*
- *ext/pdo*
- *ext/pgsql*
- *ext/phar*
- *ext/posix*
- *ext/readline*
- *ext/reflection*
- *ext/sem*
- *ext/session*
- *ext/shmop*
- *ext/simplexml*
- *ext/snmp*
- *ext/soap*
- *ext/sockets*
- *ext/spl*
- *ext/sqlite*
- *ext/sqlite3*
- *ext/ssh2*
- *ext/standard*
- *ext/tidy*
- *ext/tokenizer*
- *ext/wddx*

- *ext/xdebug*
- *ext/xmlreader*
- *ext/xmlrpc*
- *ext/xmlwriter*
- *ext/xsl*
- *ext/yaml*
- *ext/zip*
- *ext/zlib*
- *Closures Glossary*
- *Functions Glossary*
- *Recursive Functions*
- *Redeclared PHP Functions*
- *Typehints*
- *Interfaces Glossary*
- *Aliases*
- *Namespaces Glossary*
- *Autoloading*
- *Goto Names*
- *__halt_compiler*
- *Incompilable Files*
- *Labels*
- *Throw*
- *Trigger Errors*
- *Caught Expressions*
- *error_reporting() With Integers*
- *Eval() Usage*
- *Exit() Usage*
- *Forgotten Whitespace*
- *@ Operator*
- *Not Not*
- *include_once() Usage*
- *Using Short Tags*
- *Binary Glossary*
- *Email Addresses*
- *Heredoc Delimiter Glossary*
- *Hexadecimal Glossary*

- *Md5 Strings*
- *Nowdoc Delimiter Glossary*
- *Octal Glossary*
- *URL List*
- *Variable References*
- *Static Variables*
- *Variables With Long Names*
- *Variable Variables*
- *Abstract Class Usage*
- *Abstract Methods Usage*
- *Clone Usage*
- *Variable Constants*
- *Redefined PHP Traits*
- *Traits Usage*
- *Trait Names*
- *PHP Alternative Syntax*
- *Short Syntax For Arrays*
- *Inclusions*
- *ext/file*
- *ext/array*
- *ext/ffmpeg*
- *ext/info*
- *ext/math*
- *$HTTP_RAW_POST_DATA Usage*
- *ext/yis*
- *Wrong Optional Parameter*
- *Assertions*
- *Cast Usage*
- *Function Subscripting*
- *Nested Loops*
- *<?= Usage*
- *ext/pcntl*
- *ext/ming*
- *ext/redis*
- *Is An Extension Function*
- *Is An Extension Interface*

- *Is An Extension Constant*
- *ext/cyrus*
- *ext/sqlsrv*
- *Ellipsis Usage*
- *ext/0mq*
- *ext/memcache*
- *ext/memcached*
- *Is Extension Trait*
- *Dynamic Function Call*
- *Has Variable Arguments*
- *Multiple Catch*
- *Dynamically Called Classes*
- *Conditioned Function*
- *Conditioned Constants*
- *Is Generator*
- *Try With Finally*
- *Dereferencing String And Arrays*
- *Constant Scalar Expressions*
- *ext/imagick*
- *ext/oci8*
- *ext/imap*
- *Overwritten Class Const*
- *Dynamic Class Constant*
- *Dynamic Methodcall*
- *Dynamic New*
- *Dynamic Property*
- *Dynamic Classes*
- *Multiple Classes In One File*
- *File Uploads*
- *ext/intl*
- *ext/cairo*
- *Dynamic Code*
- *ext/pspell*
- *No Direct Access*
- *ext/opcache*
- *ext/expect*

- *ext/recode*
- *ext/parsekit*
- *ext/runkit*
- *ext/gettext*
- *Super Global Usage*
- *Global Usage*
- *Namespaces*
- *Deep Definitions*
- *Not Definitions Only*
- *Usage Of class_alias()*
- *ext/apache*
- *ext/eaccelerator*
- *ext/fpm*
- *ext/iis*
- *ext/xcache*
- *ext/wincache*
- *Resources Usage*
- *Shell Usage*
- *File Usage*
- *Mail Usage*
- *Dynamic Calls*
- *Test Class*
- *Mark Callable*
- *ext/dio*
- *ext/phalcon*
- *Composer Usage*
- *Composer's autoload*
- *Composer Namespace*
- *ext/apcu*
- *ext/trader*
- *ext/mailparse*
- *ext/mail*
- *Scalar Typehint Usage*
- *Return Typehint Usage*
- *ext/ob*
- *ext/geoip*

- *ext/event*
- *ext/amqp*
- *ext/gearman*
- *ext/com*
- *ext/gmagick*
- *ext/ibase*
- *ext/inotify*
- *ext/proctitle*
- *ext/wikidiff2*
- *ext/xdiff*
- *ext/libevent*
- *ext/ev*
- *ext/php-ast*
- *ext/xml*
- *ext/xhprof*
- *Else Usage*
- *Anonymous Classes*
- *Coalesce*
- *Directives Usage*
- *Global In Global*
- *ext/fann*
- *Use Web*
- *Use Cli*
- *Error Messages*
- *Php7 Relaxed Keyword*
- *ext/pecl_http*
- *Uses Environment*
- *Redefined Methods*
- *Is CLI Script*
- *PHP Bugfixes*
- *ext/tokyotyrant*
- *ext/v8js*
- *Yield Usage*
- *Yield From Usage*
- *Pear Usage*
- *ext/lua*

- *List With Keys*
- *ext/suhosin*
- *Can't Disable Function*
- *Functions Using Reference*
- *List Short Syntax*
- *Use Nullable Type*
- *Multiple Exceptions Catch()*
- *ext/rar*
- *ext/nsapi*
- *ext/newt*
- *ext/ncurses*
- *Use Composer Lock*
- *String*
- *ext/mhash*
- *ext/zbarcode*
- *ext/mongodb*
- *Error_Log() Usage*
- *SQL queries*
- *ext/libsodium*
- *ext/ds*
- *Use Cookies*
- *Group Use Declaration*
- *ext/sphinx*
- *Try With Multiple Catch*
- *ext/grpc*
- *Use Browscap*
- *Use Debug*
- *PSR-16 Usage*
- *PSR-7 Usage*
- *PSR-6 Usage*
- *PSR-3 Usage*
- *PSR-11 Usage*
- *PSR-13 Usage*
- *ext/stats*
- *Dependency Injection*
- *Courier Anti-Pattern*

- *ext/gender*
- *ext/judy*
- *Yii usage*
- *Codeigniter usage*
- *Laravel usage*
- *Symfony usage*
- *Wordpress usage*
- *Ez cms usage*
- *Joomla usage*
- *Non Breakable Space In Names*
- *Multiple Functions Declarations*
- *ext/swoole*
- *Manipulates NaN*
- *Manipulates INF*
- *Const Or Define*
- *strict_types Preference*
- *Declare strict_types Usage*
- *Encoding Usage*
- *Ticks Usage*
- *ext/lapack*
- *ext/xattr*
- *ext/rdkafka*
- *ext/fam*
- *ext/parle*
- *Regex Inventory*
- *Too Complex Expression*
- *Drupal Usage*
- *Phalcon Usage*
- *FuelPHP Usage*
- *Argon2 Usage*
- *Crypto Usage*
- *Type Array Index*
- *Incoming Variable Index Inventory*
- *ext/vips*
- *Dl() Usage*
- *Environment Variables*

- *ext/igbinary*
- *Fallback Function*
- *ext/hrtime*
- *ext/xxtea*
- *ext/uopz*
- *ext/varnish*
- *ext/opencensus*
- *ext/leveldb*
- *ext/db2*
- *ext/zookeeper*
- *ext/cmark*
- *ext/eio*
- *ext/csprng*
- *ext/lzf*
- *ext/msgpack*
- *Case Insensitive Constants*
- *Handle Arrays With Callback*
- *Detect Current Class*
- *Trailing Comma In Calls*
- *Can't Disable Class*
- *ext/seaslog*
- *Don't Read And Write In One Expression*
- *Pack Format Inventory*
- *Printf Format Inventory*
- *idn_to_ascii() New Default*
- *ext/decimal*
- *ext/psr*
- *ext/sdl*
- *ext/async*
- *ext/wasm*
- *Path lists*
- *Typed Property Usage*
- *ext/weakref*
- *ext/pcov*
- *Constant Dynamic Creation*
- *PHP 8.0 Removed Functions*

- *PHP 8.0 Removed Constants*
- *An OOP Factory*
- *PHP Overridden Function*
- *ext/svm*
- *ext/ffi*
- *ext/password*
- *ext/zend_monitor*
- *ext/uuid*
- *Concat And Addition*
- *New Functions In PHP 7.4*
- *curl_version() Has No Argument*
- *Php 7.4 New Class*
- *New Constants In PHP 7.4*
- *PHP 7.4 Removed Functions*
- *mb_strrpos() Third Argument*
- *array_key_exists() Works On Arrays*
- *Reflection Export() Is Deprecated*
- *Unbinding Closures*
- *Numeric Literal Separator*
- *Scalar Are Not Arrays*
- *PHP 7.4 Reserved Keyword*
- *No More Curly Arrays*
- *Overwritten Properties*
- *Set Parent Definition*
- *PHP 7.4 Constant Deprecation*
- *PHP 7.4 Removed Directives*
- *Hash Algorithms Incompatible With PHP 7.4-*
- *openssl_random_pseudo_byte() Second Argument*
- *Use Covariance*
- *Use Contravariance*
- *Use Arrow Functions*
- *Environment Variable Usage*
- *Indentation Levels*
- *Spread Operator For Array*
- *Nested Ternary Without Parenthesis*
- *Cyclomatic Complexity*

- *Collect Literals*
- *Collect Parameter Counts*
- *Collect Local Variable Counts*
- *Dump/DereferencingLevels*
- *Foreach() Favorite*
- *Collect Mbstring Encodings*
- *Filter To add_slashes()*
- *Typehinting Stats*
- *Typo 3 usage*
- *Concrete usage*
- *Immutable Signature*
- *Shell commands*
- *Dump/Inclusions*
- *Typehint Order*
- *New Order*
- *Links Between Parameter And Argument*
- *Collect Class Interface Counts*
- *Collect Class Depth*
- *Collect Class Children Count*
- *Constant Order*
- *Php 8.0 Variable Syntax Tweaks*
- *New Functions In PHP 8.0*
- *Php 8.0 Only TypeHints*
- *Union Typehint*
- *Signature Trailing Comma*
- *Throw Was An Expression*
- *Collect Property Counts*
- *Collect Method Counts*
- *Collect Class Constant Counts*
- *Protocol lists*
- *Call Order*
- *Uses PHP 8 Match()*
- *Collect Parameter Names*
- *Dump/FossilizedMethods*
- *Dump/CollectClassChanges*
- *Use PHP Attributes*

- *Use NullSafe Operator*
- *Use Closure Trailing Comma*
- *Collect Variables*
- *Dump/CollectGlobalVariables*
- *Collect Readability*
- *Dump/CollectDefinitionsStats*
- *Collect Class Traits Counts*
- *Collect Native Calls Per Expressions*
- *Cast Unset Usage*
- *$php_errormsg Usage*
- *Mismatch Parameter Name*
- *Collect Files Dependencies*
- *Collect Atom Counts*
- *Collect Classes Dependencies*
- *Collect Php Structures*
- *Collect Use Counts*
- *PHP 8.0 Removed Directives*
- *Unsupported Types With Operators*
- *Negative Start Index In Array*
- *Nullable With Constant*
- *PHP Resources Turned Into Objects*
- *PHP 80 Named Parameter Variadic*
- *Final Private Methods*
- *Array_Map() Passes By Value*

14.3.4 CI-checks

This ruleset is a collection of important rules to run in a CI pipeline.

Total : 176 analysis

- *Adding Zero*
- *Multiple Index Definition*
- *Forgotten Visibility*
- *Non Static Methods Called In A Static*
- *Static Methods Called From Object*
- *Constants With Strange Names*
- *Redeclared PHP Functions*
- *error_reporting() With Integers*

- *Exit() Usage*
- *Forgotten Whitespace*
- *Multiply By One*
- *@ Operator*
- *Not Not*
- *Strpos()-like Comparison*
- *Throws An Assignement*
- *var_dump()... Usage*
- *Useless Instructions*
- *Multiple Constant Definition*
- *Wrong Optional Parameter*
- *Use === null*
- *One Variable String*
- *Static Methods Can't Contain $this*
- *While(List() = Each())*
- *Multiples Identical Case*
- *Switch Without Default*
- *Nested Ternary*
- *Undefined Constants*
- *Htmlentities Calls*
- *Undefined Class Constants*
- *Undefined Functions*
- *Deprecated PHP Functions*
- *Dangling Array References*
- *Aliases Usage*
- *Uses Default Values*
- *Wrong Number Of Arguments*
- *Use const*
- *list() May Omit Variables*
- *Or Die*
- *Must Return Methods*
- *Overwritten Exceptions*
- *Foreach Reference Is Not Modified*
- *Undefined Properties*
- *Strict Comparison With Booleans*
- *Lone Blocks*

- *Logical Should Use Symbolic Operators*
- *Repeated print()*
- *Avoid Parenthesis*
- *Objects Don't Need References*
- *No Real Comparison*
- *No Direct Call To Magic Method*
- *Useless Final*
- *Use Constant*
- *Useless Unset*
- *No array_merge() In Loops*
- *Useless Parenthesis*
- *Use PHP Object API*
- *Altering Foreach Without Reference*
- *Use Pathinfo*
- *No Parenthesis For Language Construct*
- *Use Constant As Arguments*
- *Implied If*
- *Should Chain Exception*
- *Undefined Interfaces*
- *Should Use Prepared Statement*
- *Print And Die*
- *Unchecked Resources*
- *Else If Versus Elseif*
- *Multiple Class Declarations*
- *Empty Namespace*
- *Could Use Short Assignation*
- *Pre-increment*
- *Indices Are Int Or String*
- *Should Typecast*
- *Avoid Substr() One*
- *Useless Brackets*
- *preg_replace With Option e*
- *eval() Without Try*
- *Avoid get_class()*
- *Silently Cast Integer*
- *Timestamp Difference*

- *Wrong Parameter Type*
- *Redefined Class Constants*
- *Redefined Default*
- *Wrong fopen() Mode*
- *Negative Power*
- *Use random_int()*
- *Ternary In Concat*
- *Undefined Trait*
- *Identical Conditions*
- *No Choice*
- *Logical Mistakes*
- *Same Conditions In Condition*
- *Return True False*
- *Could Use __DIR__*
- *Should Use Coalesce*
- *If With Same Conditions*
- *Throw Functioncall*
- *Use Instanceof*
- *Results May Be Missing*
- *Always Positive Comparison*
- *Empty Blocks*
- *Throw In Destruct*
- *Use System Tmp*
- *Hidden Use Expression*
- *Should Make Alias*
- *Multiple Identical Trait Or Interface*
- *Multiple Alias Definitions*
- *Failed Substr Comparison*
- *Should Make Ternary*
- *Drop Else After Return*
- *Use ::Class Operator*
- *Don't Echo Error*
- *Useless Type Casting*
- *No isset() With empty()*
- *Useless Check*
- *Multiple Alias Definitions Per File*

- *__DIR__ Then Slash*
- *Repeated Regex*
- *No Class In Global*
- *Could Use str_repeat()*
- *Strings With Strange Space*
- *No Empty Regex*
- *No Reference On Left Side*
- *Assign With And Precedence*
- *No Magic Method With Array*
- *Is Actually Zero*
- *Unconditional Break In Loop*
- *Next Month Trap*
- *Printf Number Of Arguments*
- *Invalid Regex*
- *Same Variable Foreach*
- *Identical On Both Sides*
- *No Reference For Ternary*
- *Unused Inherited Variable In Closure*
- *Useless Catch*
- *Don't Unset Properties*
- *Strtr Arguments*
- *Missing Parenthesis*
- *Callback Function Needs Return*
- *strpos() Too Much*
- *Typehinted References*
- *Check JSON*
- *Undefined ::class*
- *Undefined Variable*
- *Undefined Insteadof*
- *Wrong Access Style to Property*
- *Invalid Pack Format*
- *Should Yield With Key*
- *Useless Alias*
- *Possible Missing Subpattern*
- *Assign And Compare*
- *Typehint Must Be Returned*

- *Check On __Call Usage*
- *Casting Ternary*
- *Concat And Addition*
- *Wrong Type Returned*
- *Class Without Parent*
- *Scalar Are Not Arrays*
- *Implode() Arguments Order*
- *strip_tags Skips Closed Tag*
- *Should Use Explode Args*
- *Use array_slice()*
- *Coalesce And Concat*
- *Interfaces Is Not Implemented*
- *No Literal For Reference*
- *Cant Implement Traversable*
- *Is_A() With String*
- *Mbstring Unknown Encoding*
- *Mbstring Third Arg*
- *Merge If Then*
- *Wrong Type With Call*
- *Not Equal Is Not !==*
- *Wrong Typed Property Default*
- *Wrong Type For Native PHP Function*
- *Unknown Parameter Name*
- *Missing Some Returntype*

14.3.5 ClassReview

This ruleset focuses on classes construction issues, and their related structures : traits, interfaces, methods, properties, constants.

Total : 55 analysis

- *Final Class Usage*
- *Final Methods Usage*
- *Classes Mutually Extending Each Other*
- *Could Use self*
- *Constant Class*
- *Redefined Property*
- *Useless Interfaces*

- *Could Be Class Constant*
- *Could Be Static*
- *No Self Referencing Constant*
- *Property Could Be Private Property*
- *Could Be Protected Property*
- *Raised Access Level*
- *Could Be Private Class Constant*
- *Could Be Protected Class Constant*
- *Method Could Be Private Method*
- *Could Be Protected Method*
- *Property Could Be Local*
- *Could Be Abstract Class*
- *Class Could Be Final*
- *Wrong Access Style to Property*
- *Unreachable Class Constant*
- *Avoid Self In Interface*
- *Self Using Trait*
- *Method Could Be Static*
- *Avoid option arrays in constructors*
- *Memoize MagicCall*
- *Unused Class Constant*
- *Dependant Abstract Classes*
- *Wrong Type Returned*
- *Disconnected Classes*
- *Class Without Parent*
- *Interfaces Is Not Implemented*
- *Interfaces Don't Ensure Properties*
- *Non Nullable Getters*
- *Insufficient Property Typehint*
- *Exceeding Typehint*
- *Nullable Without Check*
- *Fossilized Method*
- *Uninitialized Property*
- *Wrong Typed Property Default*
- *Hidden Nullable*
- *Missing Abstract Method*

- *Unused Trait In Class*
- *Cyclic References*
- *Double Object Assignation*
- *Mismatch Properties Typehints*
- *Different Argument Counts*
- *Could Be Parent Method*
- *Cancel Common Method*
- *Modified Typed Parameter*
- *Useless Typehint*
- *Final Private Methods*
- *Missing __isset() Method*
- *No Static Variable In A Method*

14.3.6 Coding conventions

This ruleset centralizes all analysis related to coding conventions. Sometimes, those are easy to extract with static analysis, and so here they are. No all o them are available.

Total : 0 analysis

-

14.3.7 CompatibilityPHP53

This ruleset centralizes all analysis for the migration from PHP 5.2 to 5.3.

Total : 79 analysis

- *Non Static Methods Called In A Static*
- *ext/dba*
- *ext/fdf*
- *Use Lower Case For Parent, Static And Self*
- *Break With 0*
- *Binary Glossary*
- *Malformed Octal*
- *Short Syntax For Arrays*
- *New Functions In PHP 5.4*
- *New Functions In PHP 5.5*
- *New Functions In PHP 5.6*
- *Multiple Definition Of The Same Argument*
- *Function Subscripting*
- *Closure May Use $this*

- *Switch With Too Many Default*
- *ext/ming*
- *Ellipsis Usage*
- *Exponent Usage*
- *Dereferencing String And Arrays*
- *::class*
- *Foreach With list()*
- *Use Const And Functions*
- *Constant Scalar Expressions*
- *__debugInfo() Usage*
- *Mixed Keys Arrays*
- *Const With Array*
- *Methodcall On New*
- *Hash Algorithms Incompatible With PHP 5.3*
- *Class Const With Array*
- *Variable Global*
- *Null On New*
- *isset() With Constant*
- *Anonymous Classes*
- *Unicode Escape Syntax*
- *New Functions In PHP 7.0*
- *PHP 7.0 New Classes*
- *PHP 7.0 New Interfaces*
- *Parenthesis As Parameter*
- *PHP5 Indirect Variable Expression*
- *Php 7 Indirect Expression*
- *Unicode Escape Partial*
- *Define With Array*
- *No List With String*
- *PHP7 Dirname*
- *Php7 Relaxed Keyword*
- *Cant Use Return Value In Write Context*
- *Php 7.1 New Class*
- *List With Keys*
- *List Short Syntax*
- *Use Nullable Type*

- *Multiple Exceptions Catch()*
- *No String With Append*
- *Group Use Declaration*
- *New Functions In PHP 7.3*
- *Cant Inherit Abstract Method*
- *Group Use Trailing Comma*
- *Child Class Removes Typehint*
- *No Substr Minus One*
- *Integer As Property*
- *No get_class() With Null*
- *Php 7.2 New Class*
- *List With Reference*
- *PHP 7.3 Last Empty Argument*
- *Flexible Heredoc*
- *Const Visibility Usage*
- *Hash Algorithms Incompatible With PHP 7.1-*
- *PHP 7.0 Scalar Typehints*
- *PHP 7.1 Scalar Typehints*
- *PHP 7.2 Scalar Typehints*
- *Continue Is For Loop*
- *Trailing Comma In Calls*
- *Direct Call To __clone()*
- *No Return For Generator*
- *No Reference For Static Property*
- *Typed Property Usage*
- *Concat And Addition*
- *Unpacking Inside Arrays*
- *Generator Cannot Return*
- *Coalesce Equal*

14.3.8 CompatibilityPHP54

This ruleset centralizes all analysis for the migration from PHP 5.3 to 5.4.

Total : 75 analysis

- *Non Static Methods Called In A Static*
- *Use Lower Case For Parent, Static And Self*
- *Functions Removed In PHP 5.4*

- *Break With Non Integer*
- *Calltime Pass By Reference*
- *Malformed Octal*
- *New Functions In PHP 5.5*
- *New Functions In PHP 5.6*
- *Multiple Definition Of The Same Argument*
- *Switch With Too Many Default*
- *crypt() Without Salt*
- *Ellipsis Usage*
- *Exponent Usage*
- *Dereferencing String And Arrays*
- *::class*
- *Foreach With list()*
- *Use Const And Functions*
- *Constant Scalar Expressions*
- *__debugInfo() Usage*
- *Mixed Keys Arrays*
- *Const With Array*
- *Hash Algorithms Incompatible With PHP 5.3*
- *Hash Algorithms Incompatible With PHP 5.4/5.5*
- *Class Const With Array*
- *Variable Global*
- *Null On New*
- *isset() With Constant*
- *Anonymous Classes*
- *Unicode Escape Syntax*
- *New Functions In PHP 7.0*
- *PHP 7.0 New Classes*
- *PHP 7.0 New Interfaces*
- *Parenthesis As Parameter*
- *PHP5 Indirect Variable Expression*
- *Php 7 Indirect Expression*
- *Unicode Escape Partial*
- *Define With Array*
- *No List With String*
- *PHP7 Dirname*

- *Php7 Relaxed Keyword*
- *Cant Use Return Value In Write Context*
- *Php 7.1 New Class*
- *List With Keys*
- *List Short Syntax*
- *Use Nullable Type*
- *Multiple Exceptions Catch()*
- *ext/mhash*
- *No String With Append*
- *Group Use Declaration*
- *New Functions In PHP 7.3*
- *Cant Inherit Abstract Method*
- *Group Use Trailing Comma*
- *Child Class Removes Typehint*
- *No Substr Minus One*
- *Integer As Property*
- *No get_class() With Null*
- *Php 7.2 New Class*
- *List With Reference*
- *PHP 7.3 Last Empty Argument*
- *Flexible Heredoc*
- *Const Visibility Usage*
- *Hash Algorithms Incompatible With PHP 7.1-*
- *PHP 7.0 Scalar Typehints*
- *PHP 7.1 Scalar Typehints*
- *PHP 7.2 Scalar Typehints*
- *Continue Is For Loop*
- *Trailing Comma In Calls*
- *Direct Call To __clone()*
- *No Return For Generator*
- *No Reference For Static Property*
- *Typed Property Usage*
- *Concat And Addition*
- *Unpacking Inside Arrays*
- *Generator Cannot Return*
- *Coalesce Equal*

14.3.9 CompatibilityPHP55

This ruleset centralizes all analysis for the migration from PHP 5.4 to 5.5.

Total : 67 analysis

- *Non Static Methods Called In A Static*
- *ext/apc*
- *ext/mysql*
- *Functions Removed In PHP 5.5*
- *Malformed Octal*
- *New Functions In PHP 5.6*
- *Multiple Definition Of The Same Argument*
- *Switch With Too Many Default*
- *Ellipsis Usage*
- *Exponent Usage*
- *Use password_hash()*
- *Use Const And Functions*
- *Constant Scalar Expressions*
- *__debugInfo() Usage*
- *Const With Array*
- *Hash Algorithms Incompatible With PHP 5.3*
- *Hash Algorithms Incompatible With PHP 5.4/5.5*
- *Class Const With Array*
- *Variable Global*
- *Null On New*
- *isset() With Constant*
- *Anonymous Classes*
- *Unicode Escape Syntax*
- *New Functions In PHP 7.0*
- *PHP 7.0 New Classes*
- *PHP 7.0 New Interfaces*
- *Parenthesis As Parameter*
- *PHP5 Indirect Variable Expression*
- *Php 7 Indirect Expression*
- *Unicode Escape Partial*
- *Define With Array*
- *No List With String*
- *PHP7 Dirname*

- *Php7 Relaxed Keyword*
- *Php 7.1 New Class*
- *List With Keys*
- *List Short Syntax*
- *Use Nullable Type*
- *Multiple Exceptions Catch()*
- *No String With Append*
- *Group Use Declaration*
- *New Functions In PHP 7.3*
- *Cant Inherit Abstract Method*
- *Group Use Trailing Comma*
- *Child Class Removes Typehint*
- *No Substr Minus One*
- *Integer As Property*
- *No get_class() With Null*
- *Php 7.2 New Class*
- *List With Reference*
- *PHP 7.3 Last Empty Argument*
- *Flexible Heredoc*
- *Const Visibility Usage*
- *Hash Algorithms Incompatible With PHP 7.1-*
- *PHP 7.0 Scalar Typehints*
- *PHP 7.1 Scalar Typehints*
- *PHP 7.2 Scalar Typehints*
- *Continue Is For Loop*
- *Trailing Comma In Calls*
- *Direct Call To __clone()*
- *No Return For Generator*
- *No Reference For Static Property*
- *Typed Property Usage*
- *Concat And Addition*
- *Unpacking Inside Arrays*
- *Generator Cannot Return*
- *Coalesce Equal*

14.3.10 CompatibilityPHP56

This ruleset centralizes all analysis for the migration from PHP 5.5 to 5.6.

Total : 57 analysis

- *Non Static Methods Called In A Static*
- *Malformed Octal*
- *$HTTP_RAW_POST_DATA Usage*
- *Multiple Definition Of The Same Argument*
- *Switch With Too Many Default*
- *Hash Algorithms Incompatible With PHP 5.3*
- *Hash Algorithms Incompatible With PHP 5.4/5.5*
- *Variable Global*
- *Null On New*
- *isset() With Constant*
- *Anonymous Classes*
- *Unicode Escape Syntax*
- *New Functions In PHP 7.0*
- *PHP 7.0 New Classes*
- *PHP 7.0 New Interfaces*
- *Parenthesis As Parameter*
- *PHP5 Indirect Variable Expression*
- *Php 7 Indirect Expression*
- *Unicode Escape Partial*
- *Define With Array*
- *No List With String*
- *PHP7 Dirname*
- *Php7 Relaxed Keyword*
- *Php 7.1 New Class*
- *List With Keys*
- *List Short Syntax*
- *Use Nullable Type*
- *Multiple Exceptions Catch()*
- *No String With Append*
- *Group Use Declaration*
- *New Functions In PHP 7.3*
- *Cant Inherit Abstract Method*
- *Group Use Trailing Comma*

- *Child Class Removes Typehint*
- *No Substr Minus One*
- *Integer As Property*
- *No get_class() With Null*
- *Php 7.2 New Class*
- *List With Reference*
- *PHP 7.3 Last Empty Argument*
- *Flexible Heredoc*
- *Const Visibility Usage*
- *Hash Algorithms Incompatible With PHP 7.1-*
- *PHP 7.0 Scalar Typehints*
- *PHP 7.1 Scalar Typehints*
- *PHP 7.2 Scalar Typehints*
- *Continue Is For Loop*
- *Trailing Comma In Calls*
- *Direct Call To __clone()*
- *No Return For Generator*
- *No Reference For Static Property*
- *Typed Property Usage*
- *Concat And Addition*
- *Unpacking Inside Arrays*
- *Generator Cannot Return*
- *Coalesce Equal*
- *Php 8.0 Only TypeHints*

14.3.11 CompatibilityPHP70

This ruleset centralizes all analysis for the migration from PHP 5.6 to 7.0.

Total : 49 analysis

- *ext/ereg*
- *mcrypt_create_iv() With Default Values*
- *Magic Visibility*
- *Hash Algorithms Incompatible With PHP 5.3*
- *Hash Algorithms Incompatible With PHP 5.4/5.5*
- *Reserved Keywords In PHP 7*
- *Break Outside Loop*
- *PHP 7.0 Removed Functions*

- *Empty List*
- *List With Appends*
- *Simple Global Variable*
- *Foreach Don't Change Pointer*
- *Php 7 Indirect Expression*
- *PHP 7.0 Removed Directives*
- *preg_replace With Option e*
- *Setlocale() Uses Constants*
- *Usort Sorting In PHP 7.0*
- *Hexadecimal In String*
- *func_get_arg() Modified*
- *set_exception_handler() Warning*
- *Php 7.1 New Class*
- *List With Keys*
- *List Short Syntax*
- *Use Nullable Type*
- *Multiple Exceptions Catch()*
- *New Functions In PHP 7.3*
- *Cant Inherit Abstract Method*
- *Group Use Trailing Comma*
- *Child Class Removes Typehint*
- *No Substr Minus One*
- *Integer As Property*
- *No get_class() With Null*
- *Php 7.2 New Class*
- *List With Reference*
- *PHP 7.3 Last Empty Argument*
- *Flexible Heredoc*
- *Const Visibility Usage*
- *Hash Algorithms Incompatible With PHP 7.1-*
- *PHP 7.1 Scalar Typehints*
- *PHP 7.2 Scalar Typehints*
- *Continue Is For Loop*
- *Trailing Comma In Calls*
- *No Reference For Static Property*
- *Typed Property Usage*

- *Concat And Addition*
- *Unpacking Inside Arrays*
- *Coalesce Equal*
- *Php 8.0 Only TypeHints*
- *Union Typehint*

14.3.12 CompatibilityPHP71

This ruleset centralizes all analysis for the migration from PHP 7.0 to 7.1.

Total : 36 analysis

- *ext/mcrypt*
- *Hash Algorithms Incompatible With PHP 5.3*
- *Hash Algorithms Incompatible With PHP 5.4/5.5*
- *Avoid Substr() One*
- *PHP 7.0 Removed Functions*
- *PHP 7.0 Removed Directives*
- *preg_replace With Option e*
- *Hexadecimal In String*
- *Use random_int()*
- *Using $this Outside A Class*
- *PHP 7.1 Removed Directives*
- *New Functions In PHP 7.1*
- *PHP 7.1 Microseconds*
- *Invalid Octal In String*
- *New Functions In PHP 7.3*
- *Cant Inherit Abstract Method*
- *Group Use Trailing Comma*
- *Child Class Removes Typehint*
- *Integer As Property*
- *No get_class() With Null*
- *Php 7.2 New Class*
- *List With Reference*
- *PHP 7.3 Last Empty Argument*
- *Flexible Heredoc*
- *PHP 7.2 Scalar Typehints*
- *Continue Is For Loop*
- *Trailing Comma In Calls*

- *No Reference For Static Property*
- *Typed Property Usage*
- *String Initialization*
- *Concat And Addition*
- *Unpacking Inside Arrays*
- *Coalesce Equal*
- *Php 8.0 Only TypeHints*
- *Union Typehint*
- *Signature Trailing Comma*

14.3.13 CompatibilityPHP72

This ruleset centralizes all analysis for the migration from PHP 7.1 to 7.2.

Total : 29 analysis

- *Undefined Constants*
- *Hash Algorithms Incompatible With PHP 5.3*
- *Hash Algorithms Incompatible With PHP 5.4/5.5*
- *preg_replace With Option e*
- *PHP 7.2 Deprecations*
- *PHP 7.2 Removed Functions*
- *New Functions In PHP 7.2*
- *New Constants In PHP 7.2*
- *New Functions In PHP 7.3*
- *PHP 7.2 Object Keyword*
- *No get_class() With Null*
- *Php 7.2 New Class*
- *Avoid set_error_handler $context Argument*
- *Hash Will Use Objects*
- *Can't Count Non-Countable*
- *List With Reference*
- *PHP 7.3 Last Empty Argument*
- *Flexible Heredoc*
- *Continue Is For Loop*
- *Trailing Comma In Calls*
- *No Reference For Static Property*
- *Typed Property Usage*
- *Concat And Addition*

- *Unpacking Inside Arrays*
- *Coalesce Equal*
- *Php 8.0 Only TypeHints*
- *Union Typehint*
- *Signature Trailing Comma*
- *Throw Was An Expression*

14.3.14 CompatibilityPHP73

This ruleset centralizes all analysis for the migration from PHP 7.2 to 7.3.

Total : 18 analysis

- *New Functions In PHP 7.3*
- *Unknown Pcre2 Option*
- *Compact Inexistant Variable*
- *Case Insensitive Constants*
- *Assert Function Is Reserved*
- *Continue Is For Loop*
- *PHP 7.3 Removed Functions*
- *Don't Read And Write In One Expression*
- *Typed Property Usage*
- *Concat And Addition*
- *Unpacking Inside Arrays*
- *Numeric Literal Separator*
- *PHP 74 New Directives*
- *Coalesce Equal*
- *Php 8.0 Only TypeHints*
- *Union Typehint*
- *Signature Trailing Comma*
- *Throw Was An Expression*

14.3.15 CompatibilityPHP74

This ruleset centralizes all analysis for the migration from PHP 7.3 to 7.4.

Total : 30 analysis

- *Detect Current Class*
- *Don't Read And Write In One Expression*
- *idn_to_ascii() New Default*
- *Concat And Addition*

- *New Functions In PHP 7.4*
- *curl_version() Has No Argument*
- *Php 7.4 New Class*
- *New Constants In PHP 7.4*
- *PHP 7.4 Removed Functions*
- *mb_strrpos() Third Argument*
- *array_key_exists() Works On Arrays*
- *Reflection Export() Is Deprecated*
- *Unbinding Closures*
- *Scalar Are Not Arrays*
- *PHP 7.4 Reserved Keyword*
- *No More Curly Arrays*
- *PHP 7.4 Constant Deprecation*
- *PHP 7.4 Removed Directives*
- *Hash Algorithms Incompatible With PHP 7.4-*
- *openssl_random_pseudo_byte() Second Argument*
- *Nested Ternary Without Parenthesis*
- *Filter To add_slashes()*
- *Php 8.0 Variable Syntax Tweaks*
- *New Functions In PHP 8.0*
- *Php 8.0 Only TypeHints*
- *Union Typehint*
- *Signature Trailing Comma*
- *Throw Was An Expression*
- *Uses PHP 8 Match()*
- *Avoid get_object_vars()*

14.3.16 CompatibilityPHP80

This ruleset centralizes all analysis for the migration from PHP 7.4 to 8.0.

Total : 18 analysis

- *Old Style Constructor*
- *Wrong Optional Parameter*
- *PHP 8.0 Removed Functions*
- *PHP 8.0 Removed Constants*
- *Concat And Addition*
- *Cast Unset Usage*

- *$php_errormsg Usage*
- *Mismatch Parameter Name*
- *PHP 8.0 Removed Directives*
- *Unsupported Types With Operators*
- *Negative Start Index In Array*
- *Nullable With Constant*
- *PHP Resources Turned Into Objects*
- *PHP 80 Named Parameter Variadic*
- *Final Private Methods*
- *Array_Map() Passes By Value*
- *Reserved Match Keyword*
- *Avoid get_object_vars()*

14.3.17 CompatibilityPHP81

This ruleset centralizes all analysis for the migration from PHP 8.0 to 8.1.

Total : 0 analysis

-

14.3.18 Dead code

This ruleset focuses on dead code : expressions or even structures that are written, valid but never used.

Total : 26 analysis

- *Unused Use*
- *Unused Private Properties*
- *Unused Private Methods*
- *Unused Functions*
- *Unused Constants*
- *Unreachable Code*
- *Empty Instructions*
- *Unused Methods*
- *Unused Classes*
- *Locally Unused Property*
- *Unresolved Instanceof*
- *Unthrown Exception*
- *Unused Label*
- *Unused Interfaces*
- *Unresolved Catch*

- *Unset In Foreach*
- *Empty Namespace*
- *Can't Extend Final*
- *Exception Order*
- *Undefined Caught Exceptions*
- *Unused Protected Methods*
- *Unused Returned Value*
- *Rethrown Exceptions*
- *Unused Inherited Variable In Closure*
- *Self Using Trait*
- *Useless Type Check*

14.3.19 LintButWontExec

This ruleset focuses on PHP code that lint (php -l), but that will not run. As such, this ruleset tries to go further than PHP, by connecting files, just like during execution.

Total : 30 analysis

- *Final Class Usage*
- *Final Methods Usage*
- *Classes Mutually Extending Each Other*
- *Must Return Methods*
- *Concrete Visibility*
- *No Self Referencing Constant*
- *Using $this Outside A Class*
- *Undefined Trait*
- *Raised Access Level*
- *self, parent, static Outside Class*
- *No Magic Method With Array*
- *Method Signature Must Be Compatible*
- *Mismatch Type And Default*
- *Can't Throw Throwable*
- *Abstract Or Implements*
- *Incompatible Signature Methods*
- *Undefined Insteadof*
- *Method Collision Traits*
- *Only Variable For Reference*
- *Repeated Interface*

- *Useless Alias*
- *Typehint Must Be Returned*
- *Clone With Non-Object*
- *Trait Not Found*
- *Interfaces Is Not Implemented*
- *Cant Implement Traversable*
- *Wrong Typed Property Default*
- *Mismatch Properties Typehints*
- *Could Be Stringable*
- *Only Container For Reference*

14.3.20 Performances

This ruleset focuses on performances issues : anything that slows the code's execution.

Total : 46 analysis

- *Eval() Usage*
- *For Using Functioncall*
- *@ Operator*
- *While(List() = Each())*
- *Avoid array_unique()*
- *Echo With Concat*
- *Slow Functions*
- *No array_merge() In Loops*
- *Could Use Short Assignation*
- *Pre-increment*
- *Avoid Substr() One*
- *Global Inside Loop*
- *Joining file()*
- *Simplify Regex*
- *Make One Call With Array*
- *No Count With 0*
- *Use ::Class Operator*
- *time() Vs strtotime()*
- *Getting Last Element*
- *Avoid array_push()*
- *Should Use Function*
- *Fetch One Row Format*

- *Avoid glob() Usage*
- *Avoid Large Array Assignation*
- *Should Use array_column()*
- *Avoid Concat In Loop*
- *Use pathinfo() Arguments*
- *Simple Switch*
- *Substring First*
- *Use PHP7 Encapsed Strings*
- *Slice Arrays First*
- *Double array_flip()*
- *Processing Collector*
- *Do In Base*
- *Cache Variable Outside Loop*
- *Use The Blind Var*
- *Closure Could Be A Callback*
- *fputcsv() In Loops*
- *Isset() On The Whole Array*
- *array_key_exists() Speedup*
- *Autoappend*
- *Make Magic Concrete*
- *Regex On Arrays*
- *Always Use Function With array_key_exists()*
- *No mb_substr In Loop*
- *Optimize Explode()*

14.3.21 Rector

RectorPHP is a reconstructor tool. It applies modifications in the PHP code automatically. Exakat finds results which may be automatically updated with rector.

Total : 3 analysis

- *Preprocessable*
- *Else If Versus Elseif*
- *Is_A() With String*

14.3.22 Security

This ruleset focuses on code security.

Total : 44 analysis

- *Eval() Usage*
- *Phpinfo*
- *var_dump()... Usage*
- *Hardcoded Passwords*
- *Direct Injection*
- *Avoid sleep()/usleep()*
- *parse_str() Warning*
- *Avoid Those Hash Functions*
- *No Hardcoded Port*
- *Should Use Prepared Statement*
- *No Hardcoded Ip*
- *Compare Hash*
- *preg_replace With Option e*
- *eval() Without Try*
- *Register Globals*
- *Safe Curl Options*
- *Use random_int()*
- *No Hardcoded Hash*
- *Random Without Try*
- *Indirect Injection*
- *Unserialize Second Arg*
- *Don't Echo Error*
- *Should Use session_regenerateid()*
- *Encoded Simple Letters*
- *Set Cookie Safe Arguments*
- *No Return Or Throw In Finally*
- *Mkdir Default*
- *Switch Fallthrough*
- *Upload Filename Injection*
- *Always Anchor Regex*
- *Session Lazy Write*
- *Sqlite3 Requires Single Quotes*
- *No Net For Xml Load*

- *Dynamic Library Loading*
- *Configure Extract*
- *move_uploaded_file Instead Of copy*
- *filter_input() As A Source*
- *Safe HTTP Headers*
- *Integer Conversion*
- *Minus One On Error*
- *No ENT_IGNORE*
- *No Weak SSL Crypto*
- *Keep Files Access Restricted*
- *Check Crypto Key Length*

14.3.23 Semantics

This ruleset focuses on human interpretation of the code. It reviews special values of literals, and named structures.

Total : 13 analysis

- *Variables With One Letter Names*
- *One Letter Functions*
- *Property Variable Confusion*
- *Class Function Confusion*
- *Similar Integers*
- *Duplicate Literal*
- *Parameter Hiding*
- *Weird Array Index*
- *Wrong Typehinted Name*
- *Semantic Typing*
- *Fn Argument Variable Confusion*
- *Prefix And Suffixes With Typehint*
- *Mismatch Parameter And Type*

14.3.24 Suggestions

This ruleset focuses on possibly better syntax than the one currently used. Those may be code modernization, alternatives, more efficient solutions, or simply left over from older versions.

Total : 100 analysis

- *While(List() = Each())*
- *Function Subscripting, Old Style*
- *** For Exponent*

- *Too Many Children*
- *Empty With Expression*
- *list() May Omit Variables*
- *Unreachable Code*
- *Overwritten Exceptions*
- *Return With Parenthesis*
- *Strict Comparison With Booleans*
- *Logical Should Use Symbolic Operators*
- *Could Use self*
- *Preprocess Arrays*
- *Repeated print()*
- *Echo With Concat*
- *No Parenthesis For Language Construct*
- *Unused Interfaces*
- *Avoid Substr() One*
- *PHP7 Dirname*
- *preg_match_all() Flag*
- *Already Parents Interface*
- *Could Use __DIR__*
- *Should Use Coalesce*
- *Could Use Alias*
- *Drop Else After Return*
- *Unitialized Properties*
- *Should Use array_column()*
- *Randomly Sorted Arrays*
- *No Return Used*
- *Could Make A Function*
- *Use session_start() Options*
- *Mismatched Ternary Alternatives*
- *Isset Multiple Arguments*
- *Should Use Foreach*
- *Substring First*
- *Use List With Foreach*
- *Slice Arrays First*
- *Parent First*
- *Never Used Parameter*

- *Should Use array_filter()*
- *Reuse Variable*
- *Should Use Math*
- *Could Use Compact*
- *Could Use array_fill_keys*
- *Use Count Recursive*
- *Too Many Parameters*
- *Should Preprocess Chr()*
- *Possible Increment*
- *Drop Substr Last Arg*
- *One If Is Sufficient*
- *Could Use array_unique*
- *Compact Inexistant Variable*
- *Should Use Operator*
- *Could Be Static Closure*
- *Use is_countable*
- *Detect Current Class*
- *Avoid Real*
- *Use json_decode() Options*
- *Closure Could Be A Callback*
- *Add Default Value*
- *Named Regex*
- *Could Use Try*
- *Use Basename Suffix*
- *Don't Loop On Yield*
- *Should Have Destructor*
- *Directly Use File*
- *Isset() On The Whole Array*
- *Multiple Usage Of Same Trait*
- *array_key_exists() Speedup*
- *Should Deep Clone*
- *Multiple Unset()*
- *Implode One Arg*
- *Useless Default Argument*
- *No Need For get_class()*
- *Substr To Trim*

- *Complex Dynamic Names*
- *Could Be Constant*
- *Use DateTimeImmutable Class*
- *Set Aside Code*
- *Use Array Functions*
- *Use Case Value*
- *Use Url Query Functions*
- *Too Long A Block*
- *Static Global Variables Confusion*
- *Possible Alias Confusion*
- *Too Much Indented*
- *Dont Compare Typed Boolean*
- *Abstract Away*
- *Large Try Block*
- *Cancel Common Method*
- *Useless Typehint*
- *Could Use Promoted Properties*
- *Use get_debug_type()*
- *Use str_contains()*
- *Unused Exception Variable*
- *Searching For Multiple Keys*
- *Long Preparation For Throw*
- *No Static Variable In A Method*
- *Declare Static Once*
- *Could Use Match*

14.3.25 Top10

This ruleset is a selection of analysis, with the top 10 most common. Actually, it is a little larger than that.

Total : 28 analysis

- *For Using Functioncall*
- *Strpos()-like Comparison*
- *Used Once Variables*
- *Dangling Array References*
- *Queries In Loops*
- *Use const*
- *Logical Should Use Symbolic Operators*

- *Repeated print()*
- *Objects Don't Need References*
- *No Real Comparison*
- *No array_merge() In Loops*
- *Unresolved Instanceof*
- *Avoid Substr() One*
- *No Choice*
- *Failed Substr Comparison*
- *Unitialized Properties*
- *Could Use str_repeat()*
- *Logical Operators Favorite*
- *Avoid Concat In Loop*
- *Next Month Trap*
- *Substring First*
- *Use List With Foreach*
- *Don't Unset Properties*
- *Avoid Real*
- *Should Yield With Key*
- *fputcsv() In Loops*
- *Possible Missing Subpattern*
- *Concat And Addition*

14.3.26 Typechecks

This ruleset focuses on typehinting. Missing typehint, or inconsistent typehint, are reported.

Total : 23 analysis

- *Argument Should Be Typehinted*
- *Useless Interfaces*
- *No Class As Typehint*
- *Mismatched Default Arguments*
- *Mismatched Typehint*
- *Child Class Removes Typehint*
- *Not A Scalar Type*
- *Mismatch Type And Default*
- *Insufficient Typehint*
- *Bad Typehint Relay*
- *Wrong Type With Call*

- *Missing Typehint*
- *Fossilized Method*
- *Could Be String*
- *Could Be Void*
- *Could Be Callable*
- *Wrong Argument Type*
- *Could Be Integer*
- *Could Be Null*
- *Could Be Iterable*
- *Could Be Float*
- *Could Be Self*
- *Could Be Parent*

14.3.27 php-cs-fixable

[PHP-CS-FIXER](https://github.com/FriendsOfPHP/PHP-CS-Fixer) is a tool to automatically fix PHP Coding Standards issues. It applies modifications in the PHP code automatically. Exakat finds results which may be automatically updated with PHP-CS-FIXER.

Total : 11 analysis

- *Use === null*
- *** For Exponent*
- *Logical Should Use Symbolic Operators*
- *Use Constant*
- *Else If Versus Elseif*
- *PHP7 Dirname*
- *Could Use __DIR__*
- *Isset Multiple Arguments*
- *Don't Unset Properties*
- *Multiple Unset()*
- *Implode One Arg*

CHAPTER 15

Reports

15.1 Introduction

Exakat provides multiple view to explore issue or metric generated by the rules.

15.2 Summary

- *Marmelab*
- *Meters*
- *Migration74*
- *Migration80*
- *None*
- *Owasp*
- *Perfile*
- *PhpCompilation*
- *PhpConfiguration*
- *Phpcity*
- *Phpcsfixer*
- *PlantUml*
- *RadwellCode*
- *Rector*
- *Sarb*
- *Sarif*
- *SimpleTable*
- *Stats*
- *Stubs*
- *StubsJson*
- *Text*
- *Top10*
- *Topology Order*
- *TypeChecks*
- *TypeSuggestion*
- *Uml*
- *Xml*
- *Yaml*

15.3 List of Reports

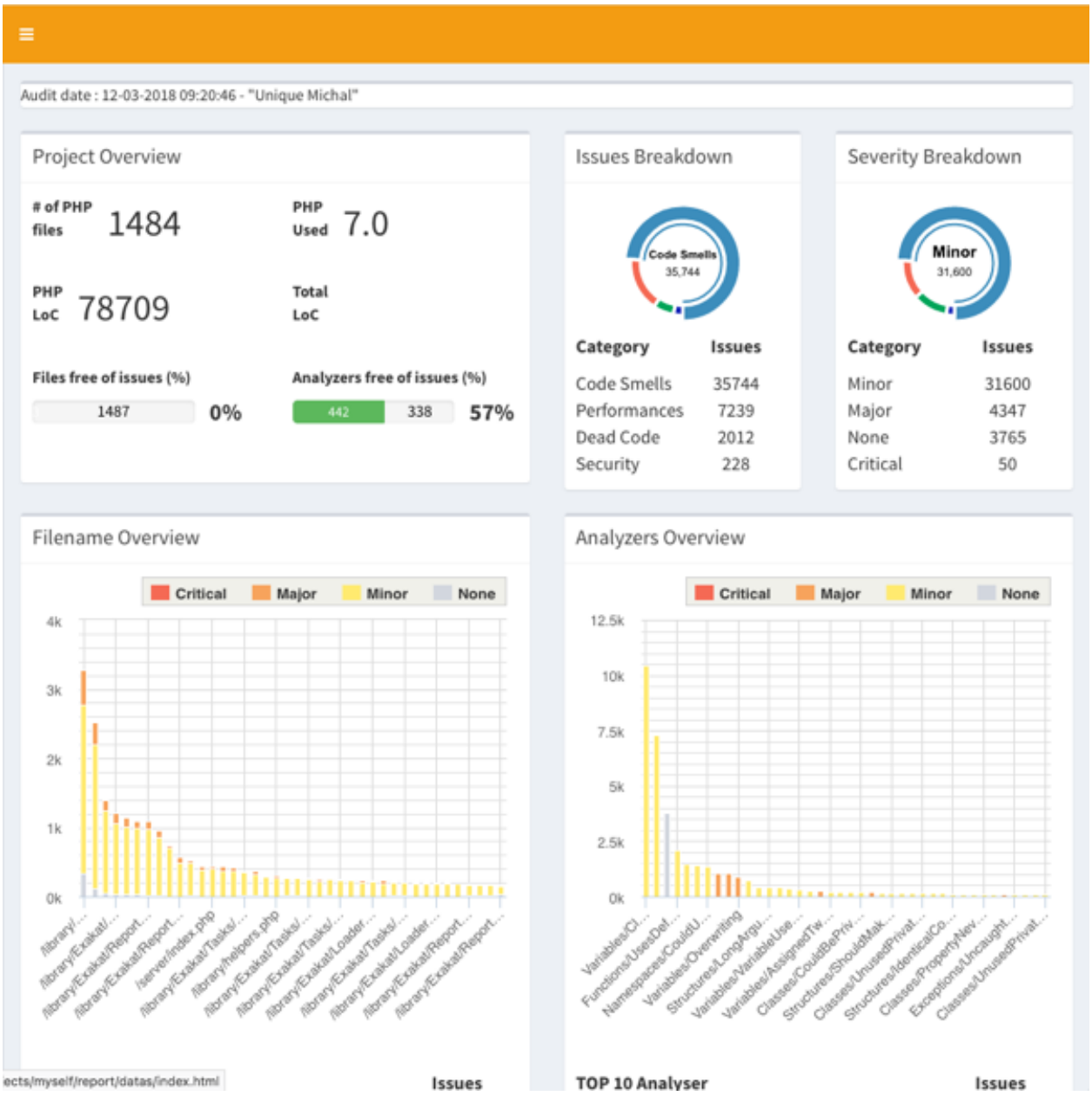
15.3.1 Ambassador

Ambassador

Ambassador is the most complete Exakat report. It used to be the default report, until Exakat 1.7.0

The Ambassador report includes :

- Full configuration for the audit
- Full documentation of the analysis
- All results, searchable and browsable by file and analysis
- **Extra reports for**
  - Minor versions compatibility
  - PHP Directive usage
  - PHP compilation recommendations
  - Error messages list
  - List of processed files



Ambassador includes the report from 3 other reports : PhpCompilation, PhpConfiguration, Stats.

Ambassador is a HTML report format.

Ambassador depends on the following 20 themes : *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP74*, *CompatibilityPHP80*, *Analyze*, Preferences, Inventory, *Performances*, Appinfo, Appcontent, Dead code, *Security*, *Suggestions*, Custom.

15.3.2 BeautyCanon

BeautyCanon

The Beauty Canon report lists all rules that report no issues.

The Beauty Canon report displays one result per line. This report lists all issues in the provided ruleset that are reporting no error.

The title of the analysis is listed on the left, and the analysis short name is listed on the right, for further documentation.

This analysis uses Analysis as default rule. It may otherwise parametered with the -T option.

Compare Hash *Compare Hash* Configure Extract *Configure Extract* Dynamic Library Loading *Dynamic Library Loading* Encoded Simple Letters *Encoded Simple Letters* Indirect Injection *Indirect Injection* Integer Conversion *Integer Conversion* Minus One On Error *Minus One On Error* Mkdir Default *Mkdir Default* No ENT_IGNORE *No ENT_IGNORE* No Hardcoded Hash *No Hardcoded Hash* No Hardcoded Ip *No Hardcoded Ip* No Hardcoded Port *No Hardcoded Port*

```
Compare Hash                                                    Security/
 ↪CompareHash
Configure Extract                                               Security/
 ↪ConfigureExtract
Dynamic Library Loading                                         Security/
 ↪DynamicDl
Encoded Simple Letters                                          Security/
 ↪EncodedLetters
Indirect Injection                                              Security/
 ↪IndirectInjection
Integer Conversion                                              Security/
 ↪IntegerConversion
Minus One On Error                                              Security/
 ↪MinusOneOnError
Mkdir Default                                                   Security/
 ↪MkdirDefault
No ENT_IGNORE                                                   Security/
 ↪NoEntIgnore
No Hardcoded Hash                                               Structures/
 ↪NoHardcodedHash
No Hardcoded Ip                                                 Structures/
 ↪NoHardcodedIp
No Hardcoded Port                                               Structures/
 ↪NoHardcodedPort
```

BeautyCanon is a Text report format.

BeautyCanon accepts any arbitrary list of results.

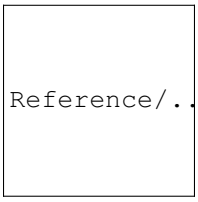
15.3.3 ClassReview

ClassReview

The ClassReview report focuses on reviewing classes, traits and interfaces.

The ClassReview report focuses on good code hygiene for classes, interfaces and traits.

It checks the internal structure of classes, and suggest visibility, typehint updates.



ClassReview is a HTML report format.

ClassReview depends on the following theme : ClassReview.

15.3.4 Classes dependendies HTML

Classes dependendies HTML

This reports displays the class dependencies, based on definition usages.

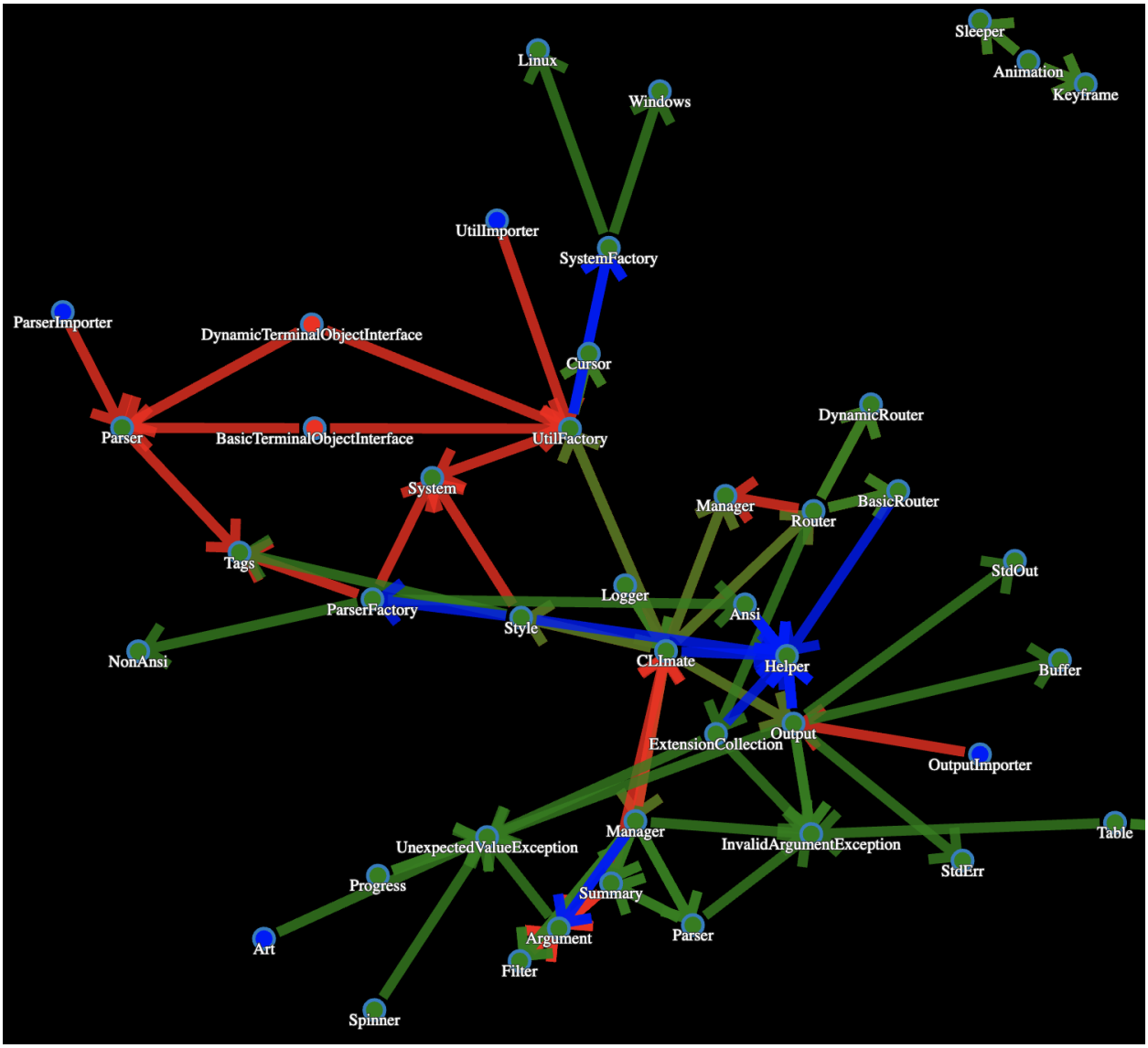
This report displays all dependencies between classes, interfaces and traits. A class (or interface or trait) depends on another class (or interface or trait) when it makes usage of one of its definitions : extends, implements, use, and static calls.

For example, *A* depends on *B*, because *A* extends *B*.

The resulting diagram is in HTML file, which is readable with most browsers, from a web server.

Warning : for browser security reasons, the report will NOT load as a local file. It needs to be served by an HTTP server, so all resources are correctly located.

Warning : large applications (> 1000 classes) will require a lot of resources to open.



Classes dependendies HTML is a HTML report format.

Classes dependendies HTML doesn't depend on themes.

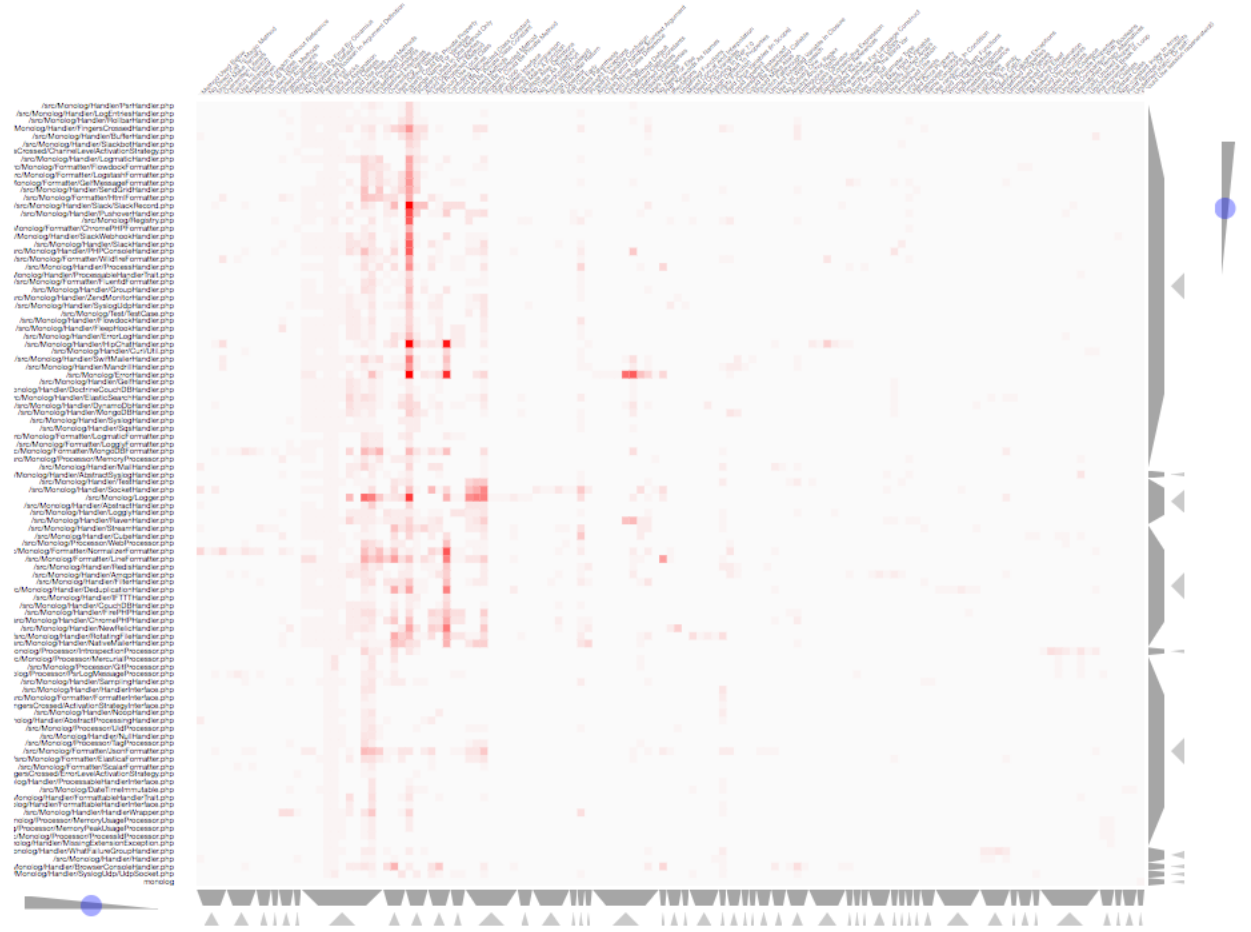
15.3.5 Clustergrammer

Clustergrammer

The Clustergrammar report format data for a clustergrammer diagram.

Clustergrammer is a visualisation tool that may be found online. After generation of this report, a TEXT file is available in the project directory. Upload it on [http://amp.pharm.mssm.edu/clustergrammer/{]}(http://amp.pharm.mssm.edu/clustergrammer/) to visualize it.

See a live report here : [Clustergrammer](http://amp.pharm.mssm.edu/clustergrammer/viz_sim_mats/5a8d41bf3a82d32a9dacddd9/clustergrammer.txt).



Clustergrammer is a TEXT report format.

Clustergrammer doesn't depend on themes.

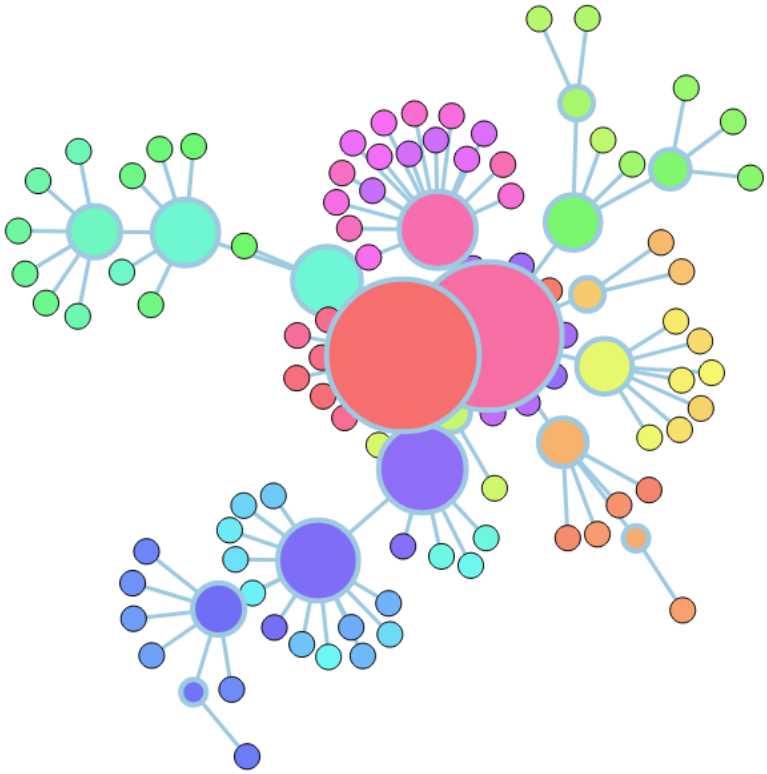
15.3.6 Code Flower

Code Flower

The Code Flower represents hierarchies in a code source.

Codeflower is a javascript visualization of the code. It is based on Francois Zaninotto's [CodeFlower Source code visualization](http://www.redotheweb.com/CodeFlower/).

It represents : + Class hierarchy + Namespace hierarchy + Inclusion



Code Flower is a HTML report format.

Code Flower doesn't depend on themes.

15.3.7 Code Sniffer

Code Sniffer

The CodeSniffer report exports in the CodeSniffer format.

This format reports analysis using the Codesniffer's result format.

See also [Code Sniffer Report](https://github.com/squizlabs/PHP_CodeSniffer/wiki/Reporting).

```
FILE : /Path/To/View/The/File.php
--------------------------------------------------------------------------------
FOUND 3 ISSUES AFFECTING 3 LINES
--------------------------------------------------------------------------------
 32 | MINOR | Could Use Alias
 41 | MINOR | Could Make A Function
 43 | MINOR | Could Make A Function
--------------------------------------------------------------------------------
```

Code Sniffer is a TEXT report format.

Code Sniffer accepts any arbitrary list of results.

15.3.8 Composer

Composer

The Composer report provide elements for the require attribute in the composer.json.

It helps documenting the composer.json, by providing more information, extracted from the code.

This report makes a copy then updates the composer.json, if available. It creates a totally new composer.json if the latter is not available.

It is recommended to review manually the results of the suggested composer.json before using it.

```
Name,File,Line
0,/features/bootstrap/FeatureContext.php,61
10000,/features/bootstrap/FeatureContext.php,61
777,/features/bootstrap/FeatureContext.php,63
20,/features/bootstrap/FeatureContext.php,73
0,/features/bootstrap/FeatureContext.php,334
0,/features/bootstrap/FeatureContext.php,339
0,/features/bootstrap/FeatureContext.php,344
0,/features/bootstrap/FeatureContext.php,362
0,/features/bootstrap/FeatureContext.php,366
0,/features/bootstrap/FeatureContext.php,368
0,/features/bootstrap/FeatureContext.php,372
777,/features/bootstrap/FeatureContext.php,423
777,/features/bootstrap/FeatureContext.php,431
0,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,68
1,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,69
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,84
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,150
```

Composer is a JSON report format.

Composer depends on the following theme : Appinfo.

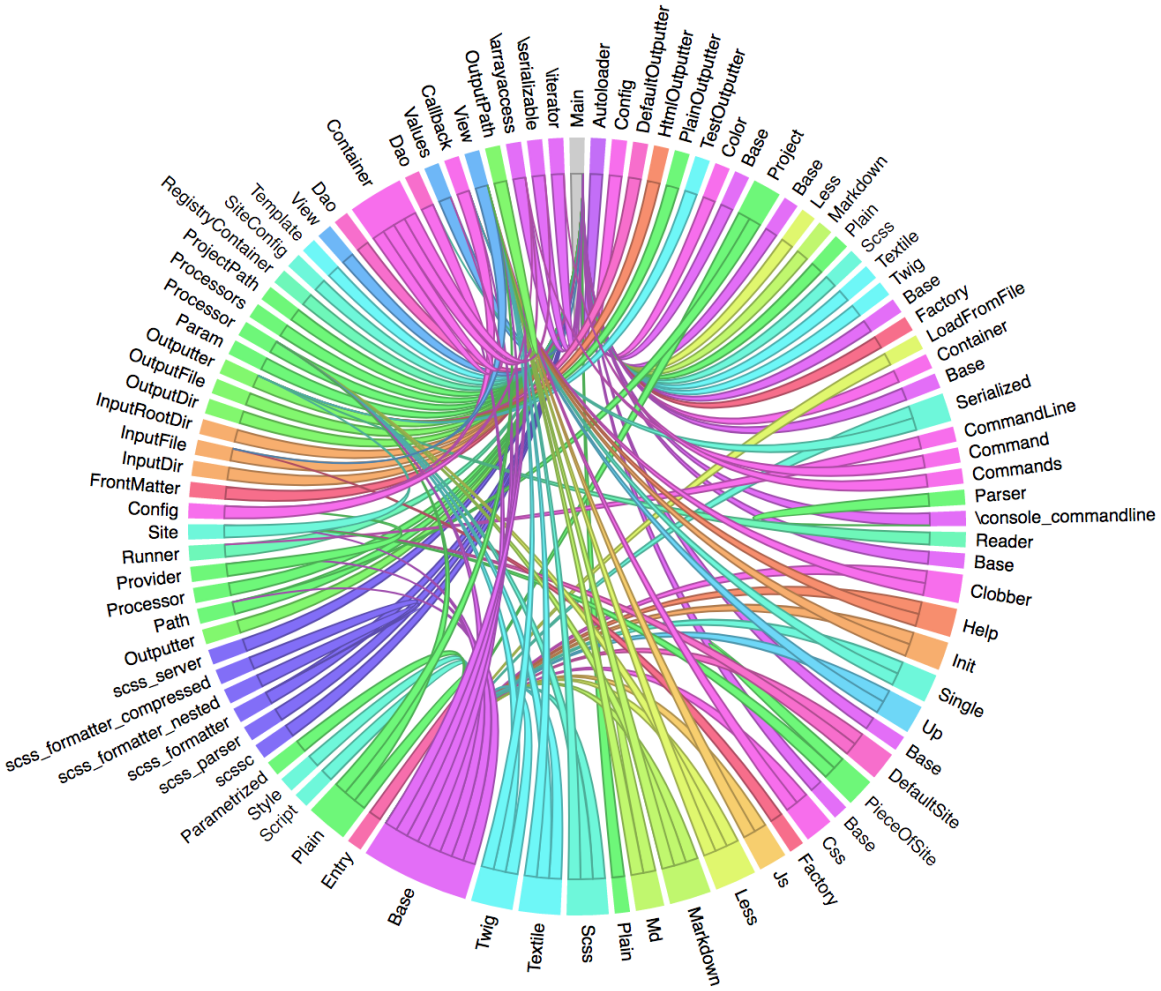
15.3.9 Dependency Wheel

Dependency Wheel

The DependencyWheel represents dependencies in a code source.

Dependency Wheel is a javascript visualization of the classes dependencies in the code. Every class, interface and trait are represented as a circle, and every relation between the classes are represented by a link between them, inside the circle.

It is based on Francois Zaninotto's DependencyWheel and the d3.js.



Dependency Wheel is a HTML report format.

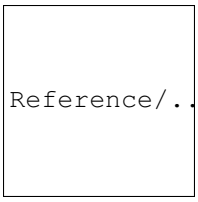
Dependency Wheel doesn't depend on themes.

15.3.10 Diplomat

Diplomat

The Diplomat is the default human readable report.

The Diplomat report is the default report since Exakat 1.7.0. It is a light version of the Ambassador report, and uses a shorter list of analysis.



Diplomat is a HTML report format.

Diplomat depends on the following 15 themes : *CompatibilityPHP53*, *CompatibilityPHP54*, *CompatibilityPHP55*, *CompatibilityPHP56*, *CompatibilityPHP70*, *CompatibilityPHP71*, *CompatibilityPHP72*, *CompatibilityPHP73*, *CompatibilityPHP74*, *CompatibilityPHP80*, *Top10*, Preferences, Appinfo, Appcontent, *Suggestions*.

15.3.11 Exakatyaml

Exakatyaml

Builds a list of ruleset, based on the number of issues from the previous audit.

Exakatyaml helpls with the configuration of exakat in a CI. It builds a list of ruleset, based on the number of issues from the previous audit.

Continuous Integration require steps that yield no issues. This is good for analysis that yield no results : in a word, all analysis that are currently clean should be in the CI. That way, any return will be monitored.

On the other hand, other analysis that currently yield issues needs to be fully cleaned before usage.

```
project: my_project
project_name: my_project
project_themes: {  }
project_reports:
    - Ambassador
rulesets:
    ruleset_0: # 0 errors found
         "Accessing Private":                                  Classes/AccessPrivate
         "Adding Zero":                                        Structures/AddZero
         "Aliases Usage":                                      Functions/AliasesUsage
         "Already Parents Interface":                          Interfaces/
↪AlreadyParentsInterface
         "Already Parents Trait":                              Traits/
↪AlreadyParentsTrait
         "Altering Foreach Without Reference":                 Structures/
↪AlteringForeachWithoutReference
         "Alternative Syntax Consistence":                     Structures/
↪AlternativeConsistenceByFile
         "Always Positive Comparison":                         Structures/NeverNegative
# Other results here
    ruleset_1: # 1 errors found
         "Constant Class":                                     Classes/ConstantClass
         "Could Be Abstract Class":                            Classes/
↪CouldBeAbstractClass
         "Dependant Trait":                                    Traits/DependantTrait
         "Double Instructions":                                Structures/
↪DoubleInstruction
# Other results here
    ruleset_2: # 2 errors found
         "Always Anchor Regex":                                Security/AnchorRegex
```

(continues on next page)

(continued from previous page)

```
        "Forgotten Interface":                                  Interfaces/
↪CouldUseInterface
# Other results here
    ruleset_3: # 3 errors found
        "@ Operator":                                           Structures/Noscream
        "Indices Are Int Or String":                            Structures/
↪IndicesAreIntOrString
        "Modernize Empty With Expression":                      Structures/ModernEmpty
        "Property Variable Confusion":                          Structures/
↪PropertyVariableConfusion
# Other results here
    ruleset_4: # 4 errors found
        "Buried Assignation":                                   Structures/
↪BuriedAssignation
        "Identical Consecutive Expression":                     Structures/
↪IdenticalConsecutive
# Other results here
    ruleset_122: # 122 errors found
        "Method Could Be Static":                               Classes/CouldBeStatic
```

```
project: page_manager
project_name: drupal_page_manager
project_themes: {  }
project_reports:
    - Ambassador
rulesets:
    ruleset_0: # 0 errors found
        "$HTTP_RAW_POST_DATA Usage":                            Php/RawPostDataUsage
        "$this Belongs To Classes Or Traits":                   Classes/ThisIsForClasses
        "$this Is Not An Array":                                Classes/ThisIsNotAnArray
        "$this Is Not For Static Methods":                      Classes/
↪ThisIsNotForStatic
        "Abstract Or Implements":                               Classes/
↪AbstractOrImplements
        "Access Protected Structures":                          Classes/AccessProtected
        "Accessing Private":                                    Classes/AccessPrivate
        "Adding Zero":                                          Structures/AddZero
        "Aliases Usage":                                        Functions/AliasesUsage
        "Already Parents Interface":                            Interfaces/
↪AlreadyParentsInterface
        "Already Parents Trait":                                Traits/
↪AlreadyParentsTrait
        "Altering Foreach Without Reference":                   Structures/
↪AlteringForeachWithoutReference
        "Alternative Syntax Consistence":                       Structures/
↪AlternativeConsistenceByFile
        "Always Positive Comparison":                           Structures/NeverNegative
        "Ambiguous Array Index":                                Arrays/AmbiguousKeys
        "Ambiguous Static":                                     Classes/AmbiguousStatic
        "Ambiguous Visibilities":                               Classes/
↪AmbiguousVisibilities
        "Anonymous Classes":                                    Classes/Anonymous
        "Assert Function Is Reserved":                          Php/
↪AssertFunctionIsReserved
        "Assign And Compare":                                   Structures/
↪AssigneAndCompare
```

(continues on next page)

(continued from previous page)

```
        "Assign Default To Properties":                    Classes/MakeDefault
        "Assign With And":                                 Php/AssignAnd
        "Assigned Twice":                                  Variables/
↪AssignedTwiceOrMore
        "Avoid Parenthesis":                               Structures/
↪PrintWithoutParenthesis
        "Avoid Those Hash Functions":                      Security/
↪AvoidThoseCrypto
        "Avoid Using stdClass":                            Php/UseStdclass
        "Avoid get_class()":                               Structures/UseInstanceof
        "Avoid option arrays in constructors":            Classes/
↪AvoidOptionArrays
        "Avoid set_error_handler $context Argument":       Php/
↪AvoidSetErrorHandlerContextArg
        "Avoid sleep()/usleep()":                          Security/NoSleep
        "Bad Constants Names":                             Constants/
↪BadConstantnames
        "Callback Needs Return":                           Functions/
↪CallbackNeedsReturn
        "Can't Count Non-Countable":                       Structures/
↪CanCountNonCountable
        "Can't Extend Final":                              Classes/CantExtendFinal
        "Can't Throw Throwable":                           Exceptions/CantThrow
        "Cant Inherit Abstract Method":                    Classes/
↪CantInheritAbstractMethod
        "Cant Instantiate Class":                          Classes/
↪CantInstantiateClass
        "Case Insensitive Constants":                      Constants/
↪CaseInsensitiveConstants
        "Cast To Boolean":                                 Structures/CastToBoolean
        "Casting Ternary":                                 Structures/
↪CastingTernary
        "Catch Overwrite Variable":                        Structures/
↪CatchShadowsVariable
        "Check All Types":                                 Structures/CheckAllTypes
        "Check JSON":                                      Structures/CheckJson
        "Check On __Call Usage":                           Classes/CheckOnCallUsage
        "Child Class Removes Typehint":                    Classes/
↪ChildRemoveTypehint
        "Class Function Confusion":                        Php/
↪ClassFunctionConfusion
        "Class Should Be Final By Ocramius":               Classes/FinalByOcramius
        "Class, Interface Or Trait With Identical Names":  Classes/CitSameName
        "Classes Mutually Extending Each Other":           Classes/MutualExtension
        "Clone With Non-Object":                           Classes/
↪CloneWithNonObject
        "Common Alternatives":                             Structures/
↪CommonAlternatives
        "Compact Inexistant Variable":                     Php/CompactInexistant
        "Compare Hash":                                    Security/CompareHash
        "Compared Comparison":                             Structures/
↪ComparedComparison
        "Concat And Addition":                             Php/ConcatAndAddition
        "Concat Empty String":                             Structures/ConcatEmpty
        "Concrete Visibility":                             Interfaces/
↪ConcreteVisibility
        "Configure Extract":                               Security/
↪ConfigureExtract
```

(continues on next page)

(continued from previous page)

```
        "Const Visibility Usage":                          Classes/
↪ConstVisibilityUsage
        "Constants Created Outside Its Namespace":         Constants/
↪CreatedOutsideItsNamespace
        "Constants With Strange Names":                    Constants/
↪ConstantStrangeNames
        "Continue Is For Loop":                            Structures/
↪ContinueIsForLoop
        "Could Be Else":                                   Structures/CouldBeElse
        "Could Be Static":                                 Structures/CouldBeStatic
        "Could Use Short Assignation":                     Structures/
↪CouldUseShortAssignation
        "Could Use __DIR__":                               Structures/CouldUseDir
        "Could Use self":                                  Classes/ShouldUseSelf
        "Could Use str_repeat()":                          Structures/
↪CouldUseStrrepeat
        "Crc32() Might Be Negative":                       Php/Crc32MightBeNegative
        "Dangling Array References":                       Structures/
↪DanglingArrayReferences
        "Deep Definitions":                                Functions/
↪DeepDefinitions
        "Define With Array":                               Php/DefineWithArray
        "Deprecated Functions":                            Php/Deprecated
        "Direct Call To __clone()":                        Php/DirectCallToClone
        "Direct Injection":                                Security/DirectInjection
        "Don't Change Incomings":                          Structures/
↪NoChangeIncomingVariables
        "Don't Echo Error":                                Security/DontEchoError
        "Don't Read And Write In One Expression":          Structures/
↪DontReadAndWriteInOneExpression
        "Don't Send $this In Constructor":                 Classes/
↪DontSendThisInConstructor
        "Don't Unset Properties":                          Classes/
↪DontUnsetProperties
        "Dont Change The Blind Var":                       Structures/
↪DontChangeBlindKey
        "Dont Mix ++":                                     Structures/
↪DontMixPlusPlus
        "Double Assignation":                              Structures/
↪DoubleAssignation
        "Dynamic Library Loading":                         Security/DynamicDl
        "Echo With Concat":                                Structures/
↪EchoWithConcat
        "Else If Versus Elseif":                           Structures/ElseIfElseif
        "Empty Blocks":                                    Structures/EmptyBlocks
        "Empty Instructions":                              Structures/EmptyLines
        "Empty Interfaces":                                Interfaces/
↪EmptyInterface
        "Empty Namespace":                                 Namespaces/
↪EmptyNamespace
        "Empty Traits":                                    Traits/EmptyTrait
        "Empty Try Catch":                                 Structures/EmptyTryCatch
        "Encoded Simple Letters":                          Security/EncodedLetters
        "Eval() Usage":                                    Structures/EvalUsage
        "Exception Order":                                 Exceptions/AlreadyCaught
        "Exit() Usage":                                    Structures/ExitUsage
        "Failed Substr Comparison":                        Structures/
↪FailingSubstrComparison
```

(continues on next page)

(continued from previous page)

```
        "Flexible Heredoc":                                  Php/FlexibleHeredoc
        "Foreach On Object":                                 Php/ForeachObject
        "Foreach Reference Is Not Modified":                 Structures/
↪ForeachReferenceIsNotModified
        "Forgotten Visibility":                              Classes/NonPpp
        "Forgotten Whitespace":                              Structures/
↪ForgottenWhiteSpace
        "Fully Qualified Constants":                         Namespaces/
↪ConstantFullyQualified
        "Functions/BadTypehintRelay":                        Functions/
↪BadTypehintRelay
        "Global Usage":                                      Structures/GlobalUsage
        "Group Use Declaration":                             Php/GroupUseDeclaration
        "Group Use Trailing Comma":                          Php/
↪GroupUseTrailingComma
        "Hash Algorithms Incompatible With PHP 5.3":         Php/HashAlgos53
        "Hash Algorithms":                                   Php/HashAlgos
        "Hash Will Use Objects":                             Php/HashUsesObjects
        "Hexadecimal In String":                             Type/HexadecimalString
        "Hidden Use Expression":                             Namespaces/HiddenUse
        "Htmlentities Calls":                                Structures/
↪Htmlentitiescall
        "Identical Conditions":                              Structures/
↪IdenticalConditions
        "Identical On Both Sides":                           Structures/
↪IdenticalOnBothSides
        "If With Same Conditions":                           Structures/
↪IfWithSameConditions
        "Illegal Name For Method":                           Classes/WrongName
        "Implement Is For Interface":                        Classes/
↪ImplementIsForInterface
        "Implemented Methods Are Public":                    Classes/
↪ImplementedMethodsArePublic
        "Implicit Global":                                   Structures/
↪ImplicitGlobal
        "Implied If":                                        Structures/ImpliedIf
        "Inclusion Wrong Case":                              Files/InclusionWrongCase
        "Incompatible Signature Methods":                    Classes/
↪IncompatibleSignature
        "Incompilable Files":                                Php/Incompilable
        "Indirect Injection":                                Security/
↪IndirectInjection
        "Integer As Property":                               Classes/
↪IntegerAsProperty
        "Integer Conversion":                                Security/
↪IntegerConversion
        "Invalid Class Name":                                Classes/WrongCase
        "Invalid Constant Name":                             Constants/InvalidName
        "Invalid Pack Format":                               Structures/
↪InvalidPackFormat
        "Invalid Regex":                                     Structures/InvalidRegex
        "Is Actually Zero":                                  Structures/IsZero
        "List Short Syntax":                                 Php/ListShortSyntax
        "List With Appends":                                 Php/ListWithAppends
        "List With Reference":                               Php/ListWithReference
        "Logical Mistakes":                                  Structures/
↪LogicalMistakes
```

(continues on next page)

(continued from previous page)

```
        "Logical Should Use Symbolic Operators":            Php/LogicalInLetters
        "Lone Blocks":                                      Structures/LoneBlock
        "Lost References":                                  Variables/LostReferences
        "Make Global A Property":                           Classes/
↪MakeGlobalAProperty
        "Method Collision Traits":                          Traits/
↪MethodCollisionTraits
        "Method Signature Must Be Compatible":              Classes/
↪MethodSignatureMustBeCompatible
        "Minus One On Error":                               Security/MinusOneOnError
        "Mismatch Type And Default":                        Functions/
↪MismatchTypeAndDefault
        "Mismatched Default Arguments":                     Functions/
↪MismatchedDefaultArguments
        "Mismatched Ternary Alternatives":                  Structures/
↪MismatchedTernary
        "Mismatched Typehint":                              Functions/
↪MismatchedTypehint
        "Missing Cases In Switch":                          Structures/MissingCases
        "Missing Include":                                  Files/MissingInclude
        "Missing New ?":                                    Structures/MissingNew
        "Missing Parenthesis":                              Structures/
↪MissingParenthesis
        "Mixed Concat And Interpolation":                   Structures/
↪MixedConcatInterpolation
        "Mkdir Default":                                    Security/MkdirDefault
        "Multiple Alias Definitions Per File":              Namespaces/
↪MultipleAliasDefinitionPerFile
        "Multiple Class Declarations":                      Classes/
↪MultipleDeclarations
        "Multiple Constant Definition":                     Constants/
↪MultipleConstantDefinition
        "Multiple Exceptions Catch()":                      Exceptions/MultipleCatch
        "Multiple Identical Trait Or Interface":            Classes/
↪MultipleTraitOrInterface
        "Multiple Index Definition":                        Arrays/
↪MultipleIdenticalKeys
        "Multiple Type Variable":                           Structures/
↪MultipleTypeVariable
        "Multiples Identical Case":                         Structures/
↪MultipleDefinedCase
        "Multiply By One":                                  Structures/MultiplyByOne
        "Must Call Parent Constructor":                     Php/
↪MustCallParentConstructor
        "Must Return Methods":                              Functions/MustReturn
        "Negative Power":                                   Structures/NegativePow
        "Nested Ternary":                                   Structures/NestedTernary
        "Never Used Parameter":                             Functions/
↪NeverUsedParameter
        "New Constants In PHP 7.2":                         Php/Php72NewConstants
        "New Functions In PHP 7.0":                         Php/Php70NewFunctions
        "New Functions In PHP 7.1":                         Php/Php71NewFunctions
        "New Functions In PHP 7.2":                         Php/Php72NewFunctions
        "New Functions In PHP 7.3":                         Php/Php73NewFunctions
        "Next Month Trap":                                  Structures/NextMonthTrap
        "No Choice":                                        Structures/NoChoice
        "No Direct Call To Magic Method":                   Classes/
↪DirectCallToMagicMethod
```

(continues on next page)

(continued from previous page)

```
        "No Direct Usage":                                  Structures/NoDirectUsage
        "No Empty Regex":                                   Structures/NoEmptyRegex
        "No Hardcoded Hash":                                Structures/
↪NoHardcodedHash
        "No Hardcoded Ip":                                  Structures/NoHardcodedIp
        "No Hardcoded Path":                                Structures/
↪NoHardcodedPath
        "No Hardcoded Port":                                Structures/
↪NoHardcodedPort
        "No Magic With Array":                              Classes/NoMagicWithArray
        "No Parenthesis For Language Construct":            Structures/
↪NoParenthesisForLanguageConstruct
        "No Real Comparison":                               Type/NoRealComparison
        "No Reference For Ternary":                         Php/
↪NoReferenceForTernary
        "No Reference On Left Side":                        Structures/
↪NoReferenceOnLeft
        "No Return For Generator":                          Php/NoReturnForGenerator
        "No Return Or Throw In Finally":                    Structures/
↪NoReturnInFinally
        "No Return Used":                                   Functions/NoReturnUsed
        "No Self Referencing Constant":                     Classes/
↪NoSelfReferencingConstant
        "No String With Append":                            Php/NoStringWithAppend
        "No Substr Minus One":                              Php/NoSubstrMinusOne
        "No Substr() One":                                  Structures/NoSubstrOne
        "No get_class() With Null":                         Structures/
↪NoGetClassNull
        "No isset() With empty()":                          Structures/
↪NoIssetWithEmpty
        "Non Ascii Variables":                              Variables/
↪VariableNonascii
        "Non Static Methods Called In A Static":            Classes/
↪NonStaticMethodsCalledStatic
        "Non-constant Index In Array":                      Arrays/NonConstantArray
        "Not A Scalar Type":                                Php/NotScalarType
        "Not Not":                                          Structures/NotNot
        "Objects Don't Need References":                    Structures/
↪ObjectReferences
        "Old Style Constructor":                            Classes/
↪OldStyleConstructor
        "Old Style __autoload()":                           Php/oldAutoloadUsage
        "One Variable String":                              Type/OneVariableStrings
        "Only Variable For Reference":                      Functions/
↪OnlyVariableForReference
        "Only Variable Passed By Reference":                Functions/
↪OnlyVariablePassedByReference
        "Only Variable Returned By Reference":              Structures/
↪OnlyVariableReturnedByReference
        "Or Die":                                           Structures/OrDie
        "Overwritten Exceptions":                           Exceptions/
↪OverwriteException
        "Overwritten Literals":                             Variables/
↪OverwrittenLiterals
        "PHP 7.0 New Classes":                              Php/Php70NewClasses
        "PHP 7.0 New Interfaces":                           Php/Php70NewInterfaces
        "PHP 7.0 Removed Directives":                       Php/
↪Php70RemovedDirective
```

(continues on next page)

(continued from previous page)

```
        "PHP 7.0 Removed Functions":                          Php/
↪Php70RemovedFunctions
        "PHP 7.0 Scalar Typehints":                           Php/PHP70scalartypehints
        "PHP 7.1 Microseconds":                               Php/Php71microseconds
        "PHP 7.1 Removed Directives":                         Php/
↪Php71RemovedDirective
        "PHP 7.1 Scalar Typehints":                           Php/PHP71scalartypehints
        "PHP 7.2 Deprecations":                               Php/Php72Deprecation
        "PHP 7.2 Object Keyword":                             Php/Php72ObjectKeyword
        "PHP 7.2 Removed Functions":                          Php/
↪Php72RemovedFunctions
        "PHP 7.2 Scalar Typehints":                           Php/PHP72scalartypehints
        "PHP 7.3 Last Empty Argument":                        Php/
↪PHP73LastEmptyArgument
        "PHP 7.3 Removed Functions":                          Php/
↪Php73RemovedFunctions
        "PHP7 Dirname":                                       Structures/PHP7Dirname
        "Parent First":                                       Classes/ParentFirst
        "Parent, Static Or Self Outside Class":               Classes/PssWithoutClass
        "Parenthesis As Parameter":                           Php/
↪ParenthesisAsParameter
        "Pathinfo() Returns May Vary":                        Php/PathinfoReturns
        "Php 7 Indirect Expression":                          Variables/
↪Php7IndirectExpression
        "Php 7.1 New Class":                                  Php/Php71NewClasses
        "Php 7.2 New Class":                                  Php/Php72NewClasses
        "Php7 Relaxed Keyword":                               Php/Php7RelaxedKeyword
        "Phpinfo":                                            Structures/PhpinfoUsage
        "Possible Infinite Loop":                             Structures/
↪PossibleInfiniteLoop
        "Possible Missing Subpattern":                        Php/MissingSubpattern
        "Preprocessable":                                     Structures/
↪ShouldPreprocess
        "Print And Die":                                      Structures/PrintAndDie
        "Printf Number Of Arguments":                         Structures/
↪PrintfArguments
        "Property Could Be Local":                            Classes/
↪PropertyCouldBeLocal
        "Queries In Loops":                                   Structures/QueriesInLoop
        "Random Without Try":                                 Structures/
↪RandomWithoutTry
        "Redeclared PHP Functions":                           Functions/
↪RedeclaredPhpFunction
        "Redefined Class Constants":                          Classes/
↪RedefinedConstants
        "Redefined Default":                                  Classes/RedefinedDefault
        "Redefined Private Property":                         Classes/
↪RedefinedPrivateProperty
        "Register Globals":                                   Security/RegisterGlobals
        "Repeated Interface":                                 Interfaces/
↪RepeatedInterface
        "Repeated Regex":                                     Structures/RepeatedRegex
        "Repeated print()":                                   Structures/RepeatedPrint
        "Results May Be Missing":                             Structures/
↪ResultMayBeMissing
        "Rethrown Exceptions":                                Exceptions/Rethrown
        "Return True False":                                  Structures/
↪ReturnTrueFalse
```

(continues on next page)

(continued from previous page)

```
        "Safe Curl Options":                                 Security/CurlOptions
        "Safe HTTP Headers":                                 Security/SafeHttpHeaders
        "Same Variables Foreach":                           Structures/
↪AutoUnsetForeach
        "Scalar Or Object Property":                        Classes/
↪ScalarOrObjectProperty
        "Self Using Trait":                                 Traits/SelfUsingTrait
        "Session Lazy Write":                               Security/
↪SessionLazyWrite
        "Set Cookie Safe Arguments":                        Security/SetCookieArgs
        "Setlocale() Uses Constants":                       Structures/
↪SetlocaleNeedsConstants
        "Several Instructions On The Same Line":            Structures/
↪OneLineTwoInstructions
        "Short Open Tags":                                  Php/ShortOpenTagRequired
        "Should Chain Exception":                           Structures/
↪ShouldChainException
        "Should Make Alias":                                Namespaces/
↪ShouldMakeAlias
        "Should Typecast":                                  Type/ShouldTypecast
        "Should Use Constants":                             Functions/
↪ShouldUseConstants
        "Should Use Prepared Statement":                    Security/
↪ShouldUsePreparedStatement
        "Should Use SetCookie()":                           Php/UseSetCookie
        "Should Yield With Key":                            Functions/
↪ShouldYieldWithKey
        "Silently Cast Integer":                            Type/SilentlyCastInteger
        "Sqlite3 Requires Single Quotes":                   Security/
↪Sqlite3RequiresSingleQuotes
        "Static Methods Can't Contain $this":              Classes/
↪StaticContainsThis
        "Strange Name For Constants":                       Constants/StrangeName
        "Strange Name For Variables":                       Variables/StrangeName
        "String Initialization":                            Arrays/
↪StringInitialization
        "String May Hold A Variable":                      Type/StringHoldAVariable
        "Strings With Strange Space":                       Type/
↪StringWithStrangeSpace
        "Strpos()-like Comparison":                         Structures/StrposCompare
        "Strtr Arguments":                                  Php/StrtrArguments
        "Suspicious Comparison":                            Structures/
↪SuspiciousComparison
        "Switch Fallthrough":                               Structures/Fallthrough
        "Switch To Switch":                                 Structures/
↪SwitchToSwitch
        "Switch Without Default":                           Structures/
↪SwitchWithoutDefault
        "Ternary In Concat":                                Structures/
↪TernaryInConcat
        "Test Then Cast":                                   Structures/TestThenCast
        "Throw Functioncall":                               Exceptions/
↪ThrowFunctioncall
        "Throw In Destruct":                                Classes/ThrowInDestruct
        "Throws An Assignement":                            Structures/
↪ThrowsAndAssign
        "Timestamp Difference":                             Structures/
↪TimestampDifference
```

(continues on next page)

(continued from previous page)

```
        "Too Many Finds":                                   Classes/TooManyFinds
        "Too Many Native Calls":                            Php/TooManyNativeCalls
        "Trailing Comma In Calls":                          Php/TrailingComma
        "Traits/TraitNotFound":                             Traits/TraitNotFound
        "Typehint Must Be Returned":                        Functions/
↪TypehintMustBeReturned
        "Typehinted References":                            Functions/
↪TypehintedReferences
        "Unchecked Resources":                              Structures/
↪UncheckedResources
        "Unconditional Break In Loop":                      Structures/
↪UnconditionLoopBreak
        "Undeclared Static Property":                       Classes/
↪UndeclaredStaticProperty
        "Undefined Constants":                              Constants/
↪UndefinedConstants
        "Undefined Insteadof":                              Traits/
↪UndefinedInsteadof
        "Undefined static:: Or self::":                     Classes/
↪UndefinedStaticMP
        "Unicode Escape Syntax":                            Php/UnicodeEscapeSyntax
        "Unknown Pcre2 Option":                             Php/UnknownPcre2Option
        "Unkown Regex Options":                             Structures/
↪UnknownPregOption
        "Unpreprocessed Values":                            Structures/
↪Unpreprocessed
        "Unreachable Code":                                 Structures/
↪UnreachableCode
        "Unset In Foreach":                                 Structures/
↪UnsetInForeach
        "Unthrown Exception":                               Exceptions/Unthrown
        "Unused Constants":                                 Constants/
↪UnusedConstants
        "Unused Global":                                    Structures/UnusedGlobal
        "Unused Inherited Variable In Closure":             Functions/
↪UnusedInheritedVariable
        "Unused Interfaces":                                Interfaces/
↪UnusedInterfaces
        "Unused Label":                                     Structures/UnusedLabel
        "Unused Private Methods":                           Classes/
↪UnusedPrivateMethod
        "Unused Private Properties":                        Classes/
↪UnusedPrivateProperty
        "Unused Returned Value":                            Functions/
↪UnusedReturnedValue
        "Upload Filename Injection":                        Security/
↪UploadFilenameInjection
        "Use Constant As Arguments":                        Functions/
↪UseConstantAsArguments
        "Use Constant":                                     Structures/UseConstant
        "Use Instanceof":                                   Classes/UseInstanceof
        "Use Nullable Type":                                Php/UseNullableType
        "Use PHP Object API":                               Php/UseObjectApi
        "Use Pathinfo":                                     Php/UsePathinfo
        "Use System Tmp":                                   Structures/UseSystemTmp
        "Use With Fully Qualified Name":                    Namespaces/
↪UseWithFullyQualifiedNS
```

(continues on next page)

(continued from previous page)

```
        "Use const":                                    Constants/
↪ConstRecommended
        "Use random_int()":                             Php/BetterRand
        "Used Once Variables":                          Variables/
↪VariableUsedOnce
        "Useless Abstract Class":                       Classes/UselessAbstract
        "Useless Alias":                                Traits/UselessAlias
        "Useless Brackets":                             Structures/
↪UselessBrackets
        "Useless Casting":                              Structures/
↪UselessCasting
        "Useless Constructor":                          Classes/
↪UselessConstructor
        "Useless Final":                                Classes/UselessFinal
        "Useless Global":                               Structures/UselessGlobal
        "Useless Instructions":                         Structures/
↪UselessInstruction
        "Useless Interfaces":                           Interfaces/
↪UselessInterfaces
        "Useless Parenthesis":                          Structures/
↪UselessParenthesis
        "Useless Return":                               Functions/UselessReturn
        "Useless Switch":                               Structures/UselessSwitch
        "Useless Unset":                                Structures/UselessUnset
        "Var Keyword":                                  Classes/OldStyleVar
        "Weak Typing":                                  Classes/WeakType
        "While(List() = Each())":                       Structures/WhileListEach
        "Wrong Number Of Arguments":                    Functions/
↪WrongNumberOfArguments
        "Wrong Optional Parameter":                     Functions/
↪WrongOptionalParameter
        "Wrong Parameter Type":                         Php/
↪InternalParameterType
        "Wrong Range Check":                            Structures/WrongRange
        "Wrong fopen() Mode":                           Php/FopenMode
        "__DIR__ Then Slash":                           Structures/DirThenSlash
        "__toString() Throws Exception":                Structures/
↪toStringThrowsException
        "error_reporting() With Integers":              Structures/
↪ErrorReportingWithInteger
        "eval() Without Try":                           Structures/
↪EvalWithoutTry
        "ext/ereg":                                     Extensions/Extereg
        "ext/mcrypt":                                   Extensions/Extmcrypt
        "filter_input() As A Source":                   Security/
↪FilterInputSource
        "func_get_arg() Modified":                      Functions/
↪funcGetArgModified
        "include_once() Usage":                         Structures/OnceUsage
        "isset() With Constant":                        Structures/
↪IssetWithConstant
        "list() May Omit Variables":                    Structures/ListOmissions
        "move_uploaded_file Instead Of copy":           Security/
↪MoveUploadedFile
        "parse_str() Warning":                          Security/
↪parseUrlWithoutParameters
        "preg_replace With Option e":                   Structures/pregOptionE
```

(continues on next page)

(continued from previous page)

```
        "self, parent, static Outside Class":                Classes/
↪NoPSSOutsideClass
        "set_exception_handler() Warning":                   Php/
↪SetExceptionHandlerPHP7
        "var_dump()... Usage":                               Structures/VardumpUsage
    ruleset_1: # 1 errors found
        "Constant Class":                                    Classes/ConstantClass
        "Could Be Abstract Class":                           Classes/
↪CouldBeAbstractClass
        "Dependant Trait":                                   Traits/DependantTrait
        "Double Instructions":                               Structures/
↪DoubleInstruction
        "Drop Else After Return":                            Structures/
↪DropElseAfterReturn
        "Empty Classes":                                     Classes/EmptyClass
        "Forgotten Thrown":                                  Exceptions/
↪ForgottenThrown
        "Inconsistent Elseif":                               Structures/
↪InconsistentElseif
        "Instantiating Abstract Class":                      Classes/
↪InstantiatingAbstractClass
        "List With Keys":                                    Php/ListWithKeys
        "Logical To in_array":                               Performances/
↪LogicalToInArray
        "No Need For Else":                                  Structures/NoNeedForElse
        "Same Conditions In Condition":                      Structures/
↪SameConditions
        "Should Use session_regenerateid()":                 Security/
↪ShouldUseSessionRegenerateId
        "Static Loop":                                       Structures/StaticLoop
        "Too Many Injections":                               Classes/
↪TooManyInjections
        "Undefined Caught Exceptions":                       Exceptions/
↪CaughtButNotThrown
        "Unresolved Catch":                                  Classes/UnresolvedCatch
        "Unserialize Second Arg":                            Security/
↪UnserializeSecondArg
        "Use Positive Condition":                            Structures/
↪UsePositiveCondition
        "Useless Catch":                                     Exceptions/UselessCatch
        "Useless Check":                                     Structures/UselessCheck
    ruleset_2: # 2 errors found
        "Always Anchor Regex":                               Security/AnchorRegex
        "Forgotten Interface":                               Interfaces/
↪CouldUseInterface
        "No Class As Typehint":                              Functions/
↪NoClassAsTypehint
        "No array_merge() In Loops":                         Performances/
↪ArrayMergeInLoops
        "Pre-increment":                                     Performances/
↪PrePostIncrement
        "Randomly Sorted Arrays":                            Arrays/
↪RandomlySortedLiterals
        "Should Make Ternary":                               Structures/
↪ShouldMakeTernary
        "Should Use Coalesce":                               Php/ShouldUseCoalesce
        "Use === null":                                      Php/IsnullVsEqualNull
```

(continues on next page)

(continued from previous page)

```
    ruleset_3: # 3 errors found
          "@ Operator":                                  Structures/Noscream
          "Indices Are Int Or String":                   Structures/
↪IndicesAreIntOrString
          "Modernize Empty With Expression":             Structures/ModernEmpty
          "Property Variable Confusion":                 Structures/
↪PropertyVariableConfusion
          "Too Many Local Variables":                    Functions/
↪TooManyLocalVariables
          "Unused Classes":                              Classes/UnusedClass
          "Usort Sorting In PHP 7.0":                    Php/UsortSorting
    ruleset_4: # 4 errors found
          "Buried Assignation":                          Structures/
↪BuriedAssignation
          "Identical Consecutive Expression":            Structures/
↪IdenticalConsecutive
          "Nested Ifthen":                               Structures/NestedIfthen
          "No Boolean As Default":                       Functions/
↪NoBooleanAsDefault
          "Use Named Boolean In Argument Definition":    Functions/
↪AvoidBooleanArgument
    ruleset_5: # 5 errors found
          "Avoid Optional Properties":                   Classes/
↪AvoidOptionalProperties
          "Empty Function":                              Functions/EmptyFunction
          "Relay Function":                              Functions/RelayFunction
          "Strict Comparison With Booleans":             Structures/
↪BooleanStrictComparison
          "Use Class Operator":                          Classes/UseClassOperator
          "strpos() Too Much":                           Performances/
↪StrposTooMuch
    ruleset_6: # 6 errors found
          "Used Once Property":                          Classes/UsedOnceProperty
    ruleset_7: # 7 errors found
          "No Class In Global":                          Php/NoClassInGlobal
          "Uncaught Exceptions":                         Exceptions/
↪UncaughtExceptions
          "Unused Functions":                            Functions/
↪UnusedFunctions
          "Wrong Number Of Arguments In Methods":        Functions/
↪WrongNumberOfArgumentsMethods
    ruleset_8: # 8 errors found
          "Could Make A Function":                       Functions/
↪CouldCentralize
          "Insufficient Typehint":                       Functions/
↪InsufficientTypehint
          "Long Arguments":                              Structures/LongArguments
          "Property Used In One Method Only":            Classes/
↪PropertyUsedInOneMethodOnly
          "Static Methods Called From Object":           Classes/
↪StaticMethodsCalledFromObject
    ruleset_9: # 9 errors found
          "PHP Keywords As Names":                       Php/ReservedNames
          "Undefined Trait":                             Traits/UndefinedTrait
          "Written Only Variables":                      Variables/
↪WrittenOnlyVariable
    ruleset_10: # 10 errors found
```

(continues on next page)

(continued from previous page)

```
        "Bail Out Early":                                   Structures/BailOutEarly
        "Hardcoded Passwords":                              Functions/
↪HardcodedPasswords
        "Multiple Alias Definitions":                       Namespaces/
↪MultipleAliasDefinitions
    ruleset_11: # 11 errors found
        "Variable Is Not A Condition":                      Structures/
↪NoVariableIsACondition
    ruleset_13: # 13 errors found
        "Undefined Functions":                              Functions/
↪UndefinedFunctions
        "Unused Use":                                       Namespaces/UnusedUse
    ruleset_14: # 14 errors found
        "Iffectations":                                     Structures/Iffectation
        "No Public Access":                                 Classes/NoPublicAccess
    ruleset_16: # 16 errors found
        "Overwriting Variable":                             Variables/Overwriting
    ruleset_17: # 17 errors found
        "No Net For Xml Load":                              Security/NoNetForXmlLoad
        "Unresolved Instanceof":                            Classes/
↪UnresolvedInstanceof
    ruleset_21: # 21 errors found
        "Undefined Class Constants":                        Classes/
↪UndefinedConstants
    ruleset_27: # 27 errors found
        "Locally Unused Property":                          Classes/
↪LocallyUnusedProperty
        "Never Used Properties":                            Classes/
↪PropertyNeverUsed
    ruleset_35: # 35 errors found
        "Useless Referenced Argument":                      Functions/
↪UselessReferenceArgument
    ruleset_38: # 38 errors found
        "Uses Default Values":                              Functions/
↪UsesDefaultArguments
    ruleset_47: # 47 errors found
        "Unused Arguments":                                 Functions/
↪UnusedArguments
    ruleset_49: # 49 errors found
        "Undefined Properties":                             Classes/
↪UndefinedProperty
    ruleset_77: # 77 errors found
        "Undefined Parent":                                 Classes/
↪UndefinedParentMP
    ruleset_78: # 78 errors found
        "Undefined ::class":                                Classes/
↪UndefinedStaticclass
    ruleset_82: # 82 errors found
        "Class Could Be Final":                             Classes/CouldBeFinal
    ruleset_86: # 86 errors found
        "Unused Protected Methods":                         Classes/
↪UnusedProtectedMethods
    ruleset_89: # 89 errors found
        "Unresolved Classes":                               Classes/
↪UnresolvedClasses
    ruleset_94: # 94 errors found
        "Used Once Variables (In Scope)":                   Variables/
↪VariableUsedOnceByContext
```

(continues on next page)

(continued from previous page)

```
    ruleset_122: # 122 errors found
         "Method Could Be Static":                                  Classes/CouldBeStatic
    ruleset_133: # 133 errors found
         "Should Use Local Class":                                  Classes/ShouldUseThis
    ruleset_159: # 159 errors found
         "Undefined Interfaces":                                    Interfaces/
↪UndefinedInterfaces
    ruleset_160: # 160 errors found
         "Unused Methods":                                          Classes/UnusedMethods
    ruleset_183: # 183 errors found
         "Undefined Variable":                                      Variables/
↪UndefinedVariable
    ruleset_337: # 337 errors found
         "Unresolved Use":                                          Namespaces/UnresolvedUse
    ruleset_595: # 595 errors found
         "Undefined Classes":                                       Classes/UndefinedClasses
```

Exakatyaml is a Yaml report format.

Exakatyaml doesn't depend on themes.

15.3.12 File dependendies

File dependendies

This reports displays the file dependencies, based on definition usages.

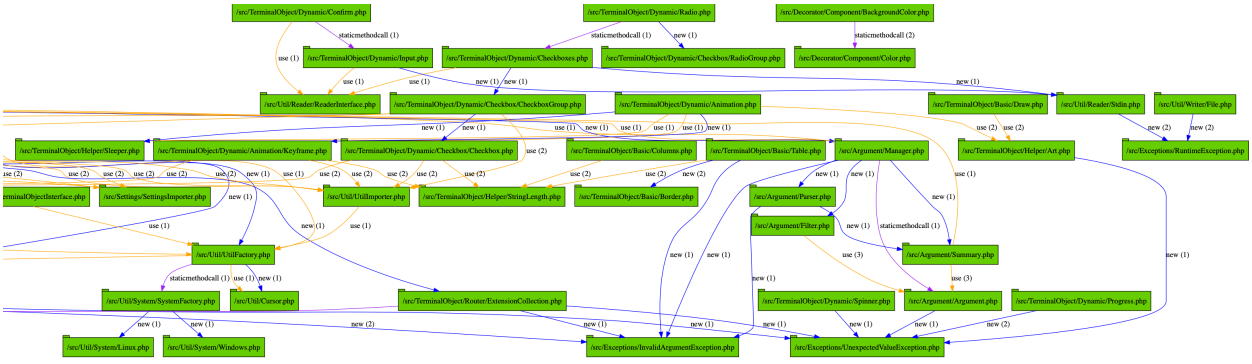
This report displays all dependencies between files. A file depends on another when it makes usage of one of its definitions : constant, functions, classes, traits, interfaces.

For example, *A.php* depends on *B.php*, because *A.php* uses the function *foo*, which is defined in the *B.php* file. On the other hand, *B.php* doesn't depends on *A.php*, as a function may be defined, but not used.

This diagram shows which files may be used without others.

The resulting diagram is a DOT file, which is readable with [Graphviz](https://www.graphviz.org/about/). Those viewers will display the diagram, and also convert it to other format, such as PNG, JPEG, PDF or others.

Another version of the same diagram is called Filedependencieshtml



File dependendies is a DOT report format.

File dependendies doesn't depend on themes.

15.3.13 File dependendies HTML

File dependendies HTML

This reports displays the file dependencies, based on definition usages.

This report displays all dependencies between files. A file depends on another when it makes usage of one of its definitions : constant, functions, classes, traits, interfaces.

For example, *A.php* depends on *B.php*, because *A.php* uses the function *foo*, which is defined in the *B.php* file. On the other hand, *B.php* doesn't depends on *A.php*, as a function may be defined, but not used.

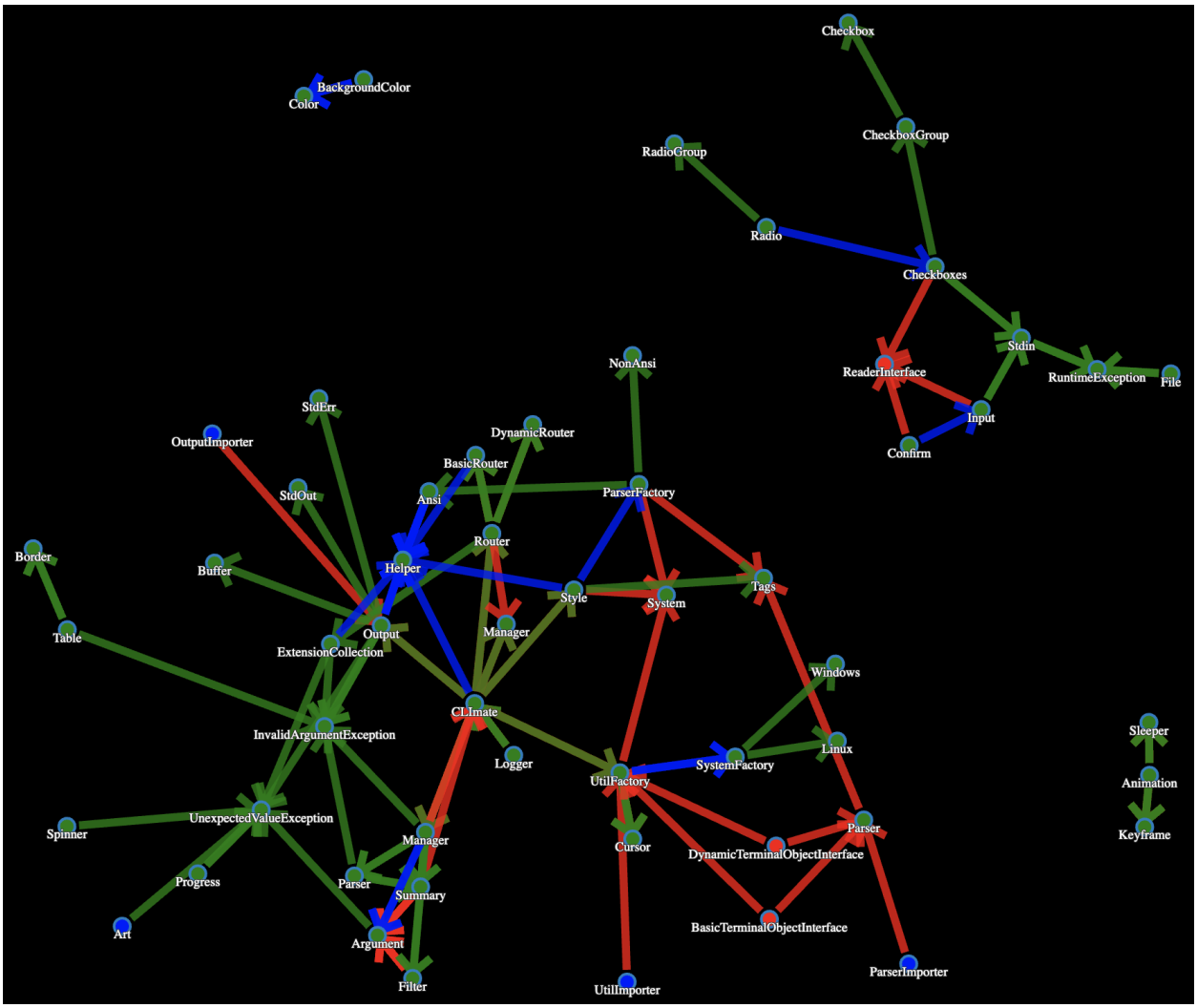
This diagram shows which files may be used without others.

The resulting diagram is in HTML file, which is readable with most browsers, from a web server.

Warning : for browser security reasons, the report will NOT load as a local file. It needs to be served by an HTTP server, so all resources are correctly located.

Warning : large applications (> 1000 files) will require a lot of resources to open.

Another version of the same diagram is called Filedependencies, and produces a DOT file



File dependendies HTML is a HTML report format.

File dependendies HTML doesn't depend on themes.

15.3.14 History

History

The History report collects meta information between audits. It saves the values from the current audit into a separate 'history.sqlite' database.

The history tables are the same as the dump.sqlite tables, except for the extra 'serial' table. Each audit comes with 3 identifiers :

- 'dump_timestamp' : this is a timmestamp taken when the dump was build
- 'dump_serial' : this is a serial number, based on the previous audit, and incremented by one. This is handy to keep the values in sequence
- 'dump_id' : this is a unique random id, which helps distinguish audits which may have inconsistency between serial or timestamp.

This report provides a 'history.sqlite' database. The following tables are inventoried :

- hash
- resultsCounts

History is a Sqlite report format.

History doesn't depend on themes.

15.3.15 Inventories

Inventories

The Inventories report collects literals and names from the code.

This report provides the value, the file and line where a type of value is present.

The following values and names are inventoried :

- Variables
- Incoming Variables
- Session Variables
- Global Variables
- Date formats
- Constants
- Functions
- Classes
- Interface names
- Trait names
- Namespaces
- Exceptions

- Regex
- SQL queries
- URL
- Unicode blocks
- Integers
- Reals numbers
- Literal Arrays
- Strings

Every type of values is exported to a file. If no value of such type was found during the audit, the file only contains the headers. It is always produced.

```
Name,File,Line
0,/features/bootstrap/FeatureContext.php,61
10000,/features/bootstrap/FeatureContext.php,61
777,/features/bootstrap/FeatureContext.php,63
20,/features/bootstrap/FeatureContext.php,73
0,/features/bootstrap/FeatureContext.php,334
0,/features/bootstrap/FeatureContext.php,339
0,/features/bootstrap/FeatureContext.php,344
0,/features/bootstrap/FeatureContext.php,362
0,/features/bootstrap/FeatureContext.php,366
0,/features/bootstrap/FeatureContext.php,368
0,/features/bootstrap/FeatureContext.php,372
777,/features/bootstrap/FeatureContext.php,423
777,/features/bootstrap/FeatureContext.php,431
0,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,68
1,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,69
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,84
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,150
```

Inventories is a CSV report format.

Inventories depends on the following theme : Inventories.

15.3.16 Json

Json

The JSON report exports in JSON format.

Simple Json format. It is a structured array with all results, described as object.

```
Filename => [
                errors   => count,
                warning  => count,
                fixable  => count,
                filename => string,
                message  => [
                    line => [
                        type,
                        source,
                        severity,
```

(continues on next page)

(continued from previous page)

```
                        fixable,
                        message
                    ]
                ]
            ]
```

```
{
   "\/src\/Path\/To\/File.php":{
      "errors":0,
      "warnings":105,
      "fixable":0,
      "filename":"\/src\/Path\/To\/File.php",
      "messages":{
         "55":[
            [
               {
                  "type":"warning",
                  "source":"Php/EllipsisUsage",
                  "severity":"Major",
                  "fixable":"fixable",
                  "message":"... Usage"
               }
            ]
         ],
         }
    }
}
```

Json is a Json report format.

Json accepts any arbitrary list of results.

15.3.17 Marmelab

Marmelab

The Marmelab report format data to use with a graphQL server.

Marmelab is a report format to build GraphQL server with exakat's results. Export the results of the audit in this JSON file, then use the [json-graphql-server](https://github.com/marmelab/json-graphql-server) to have a GraphQL server with all the results.

You may also learn more about GraphQL at [Introducing Json GraphQL Server](https://marmelab.com/blog/2017/07/12/json-graphql-server.html).

```
php exakat.phar report -p -format Marmelab -file marmelab
cp projects/myproject/marmelab.json path/to/marmelab
json-graphql-server db.json
```

Marmelab is a JSON report format.

Marmelab depends on the following theme : Analyze.

15.3.18 Meters

Meters

The Meters report export various dimensions of the audited code.

Exakat measures a large number of code dimensions, such as number of files, lines of code, tokens. All those are collected in this report.

```
{
```

    loc: 95950, locTotal: 140260, files: 1824, tokens: 677213

}

Meters is a JSON report format.

Meters depends on the following theme : None.

15.3.19 Migration74

Migration74

The Migration74 is the report dedicated to migrating PHP code to version 7.4.

The Migration74 report runs the backward incompatibilities tests for PHP 7.4, from a PHP 7.3 compatible code.

```
Name,File,Line
0,/features/bootstrap/FeatureContext.php,61
10000,/features/bootstrap/FeatureContext.php,61
777,/features/bootstrap/FeatureContext.php,63
20,/features/bootstrap/FeatureContext.php,73
0,/features/bootstrap/FeatureContext.php,334
0,/features/bootstrap/FeatureContext.php,339
0,/features/bootstrap/FeatureContext.php,344
0,/features/bootstrap/FeatureContext.php,362
0,/features/bootstrap/FeatureContext.php,366
0,/features/bootstrap/FeatureContext.php,368
0,/features/bootstrap/FeatureContext.php,372
777,/features/bootstrap/FeatureContext.php,423
777,/features/bootstrap/FeatureContext.php,431
0,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,68
1,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,69
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,84
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,150
```

Migration74 is a HTML report format.

Migration74 depends on the following 2 themes : *CompatibilityPHP73*, *Suggestions*.

15.3.20 Migration80

Migration80

The Migration80 is the report dedicated to migrating PHP code to version 8.0.

The Migration 80 report runs the backward incompatibilities tests for PHP 8.0, from a PHP 7.4 compatible code.

```
Name,File,Line
0,/features/bootstrap/FeatureContext.php,61
10000,/features/bootstrap/FeatureContext.php,61
777,/features/bootstrap/FeatureContext.php,63
20,/features/bootstrap/FeatureContext.php,73
0,/features/bootstrap/FeatureContext.php,334
0,/features/bootstrap/FeatureContext.php,339
0,/features/bootstrap/FeatureContext.php,344
0,/features/bootstrap/FeatureContext.php,362
0,/features/bootstrap/FeatureContext.php,366
0,/features/bootstrap/FeatureContext.php,368
0,/features/bootstrap/FeatureContext.php,372
777,/features/bootstrap/FeatureContext.php,423
777,/features/bootstrap/FeatureContext.php,431
0,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,68
1,/src/Behat/Behat/Context/ContextClass/SimpleClassGenerator.php,69
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,84
0,/src/Behat/Behat/Context/Environment/InitializedContextEnvironment.php,150
```

Migration80 is a HTML report format.

Migration80 depends on the following 2 themes : *CompatibilityPHP80*, *Suggestions*.

15.3.21 None

None

None is the empty report. It runs the report generating stack, but doesn't produce any result.

None is a utility report, aimed to test exakat's installation.

None is a None report format.

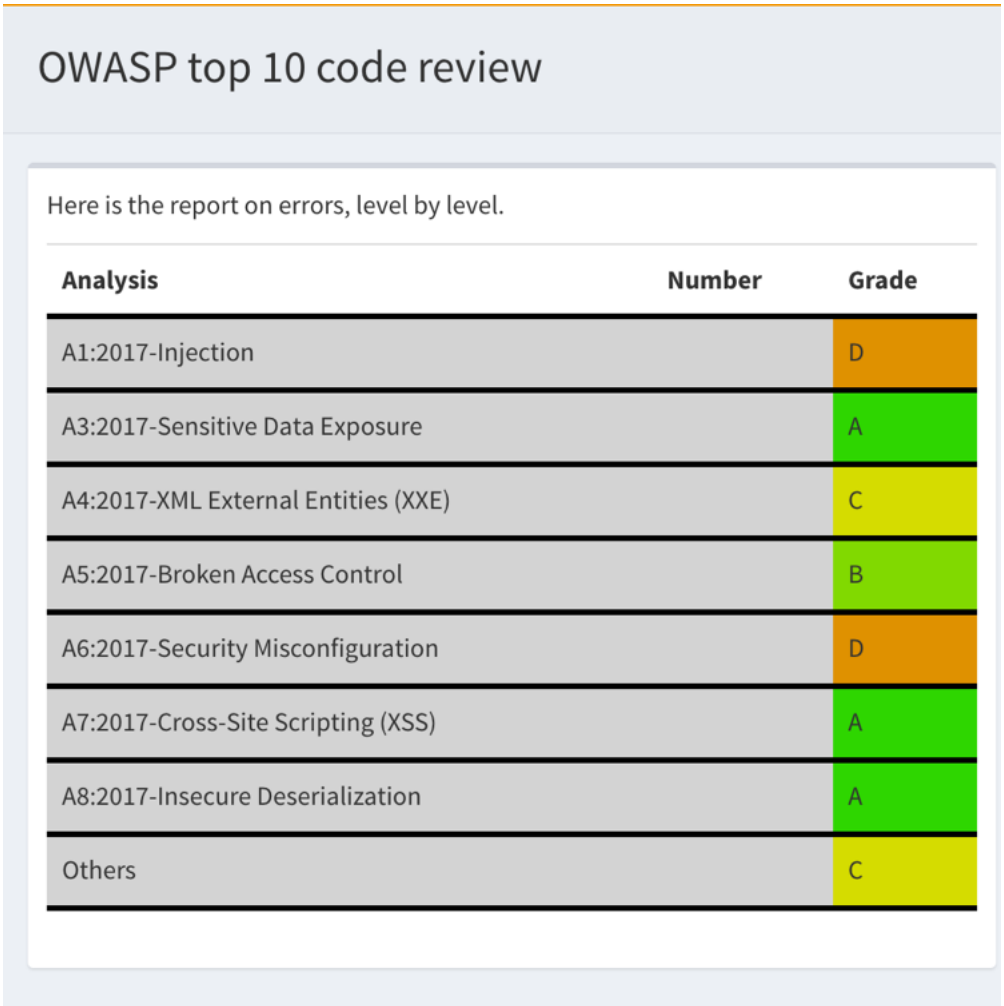
None depends on the following theme : Any.

15.3.22 Owasp

Owasp

The OWASP report is a security report.

The OWASP report focuses on the [OWASP top 10](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project). It reports all the security analysis, distributed across the 10 categories of vulnerabilities.



OWASP top 10 code review

Here is the report on errors, level by level.

Analysis	Number	Grade
A1:2017-Injection		D
A3:2017-Sensitive Data Exposure		A
A4:2017-XML External Entities (XXE)		C
A5:2017-Broken Access Control		B
A6:2017-Security Misconfiguration		D
A7:2017-Cross-Site Scripting (XSS)		A
A8:2017-Insecure Deserialization		A
Others		C

Owasp is a HTML report format.

Owasp depends on the following theme : Security.

15.3.23 Perfile

Perfile

The Perfile report lays out the results file per file.

The Perfile report displays one result per line, grouped by file, and ordered by line number :

```
/path/from/project/root/to/file:line[space]name of analysis
```

This format is fast, and fitted for human review.

```
------------------------------------------------------------
 line  /themes/Rozier/Controllers/LoginController.php
------------------------------------------------------------
   34  Multiple Alias Definitions
   36  Unresolved Use
   43  Multiple Alias Definitions
```

(continues on next page)

(continued from previous page)

```
   51  Class Could Be Final
   58  Undefined Interfaces
   81  Undefined Interfaces
   81  Unused Arguments
   81  Used Once Variables (In Scope)
   91  Undefined Interfaces
   91  Unused Arguments
   91  Used Once Variables (In Scope)
  101  Undefined Interfaces
  103  Nested Ifthen
  104  Unresolved Classes
  106  Buried Assignation
  106  Iffectations
  106  Use Positive Condition
  121  Uncaught Exceptions
  121  Unresolved Classes
  129  Uncaught Exceptions
-------------------------------------------------------
```

Perfile is a Text report format.

Perfile accepts any arbitrary list of results.

15.3.24 PhpCompilation

PhpCompilation

The PhpCompilation suggests a list of compilation directives when compiling the PHP binary, tailored for the code

PhpCompilation bases its selection on the code and its usage of features. PhpCompilation also recommends disabling unused standard extensions : this helps reducing the footprint of the binary, and prevents unused features to be available for intrusion. PhpCompilation is able to detects over 150 PHP extensions.

```
;;;;;;;;;;;;;;;;;;;;;;;;;
; Suggestion for php.ini ;
;;;;;;;;;;;;;;;;;;;;;;;;;

; The directives below are selected based on the code provided.
; They only cover the related directives that may have an impact on the code
;
; The list may not be exhaustive
; The suggested values are not recommendations, and should be reviewed and adapted
;


[date]
; It is not safe to rely on the system's timezone settings. Make sure the
; directive date.timezone is set in php.ini.
date.timezone = Europe/Amsterdam



[pcre]
; More information about pcre :
;http://php.net/manual/en/pcre.configuration.php
```

(continues on next page)

(continued from previous page)

```
[standard]
; This sets the maximum amount of memory in bytes that a script is allowed to
; allocate. This helps prevent poorly written scripts for eating up all available
; memory on a server. It is recommended to set this as low as possible and avoid
; removing the limit.
memory_limit = 120

; This sets the maximum amount of time, in seconds, that a script is allowed to
; run. The lower the value, the better for the server, but also, the better has
; the script to be written. Avoid really large values that are only useful for
; admin, and set them per directory.
max_execution_time = 90

; Exposes to the world that PHP is installed on the server. For security reasons,
; it is better to keep this hidden.
expose_php = Off

; This determines whether errors should be printed to the screen as part of the
; output or if they should be hidden from the user.
display_errors = Off

; Set the error reporting level. Always set this high, so as to have the errors
; reported, and logged.
error_reporting = E_ALL

; Always log errors for future use
log_errors = On

; Name of the file where script errors should be logged.
error_log = Name of a writable file, suitable for logging.

; More information about standard :
;http://php.net/manual/en/info.configuration.php

; Name of the file where script errors should be logged.
disable_functions = curl_init,ftp_connect,ftp_ssl_connect,ldap_connect,mail,mysqli_
 ↪connect,mysqli_pconnect,pg_connect,pg_pconnect,socket_create,socket_accept,socket_
 ↪connect,socket_listen
disable_classes = mysqli
```

PhpCompilation is a Text report format.

PhpCompilation depends on the following theme : Appinfo.

15.3.25 PhpConfiguration

PhpConfiguration

The PhpConfiguration suggests a list of directives to check when setting up the hosting server, tailored for the code

PhpConfiguration bases its selection on the code, and classic recommendations. For example, memory_limit or expose_php are always reported, though they have little impact in the code. Extensions also get a short list of important directive, and offer a link to the documentation for more documentation.

```
;;;;;;;;;;;;;;;;;;;;;;;;;
; Suggestion for php.ini ;
;;;;;;;;;;;;;;;;;;;;;;;;;

; The directives below are selected based on the code provided.
; They only cover the related directives that may have an impact on the code
;
; The list may not be exhaustive
; The suggested values are not recommendations, and should be reviewed and adapted
;


[date]
; It is not safe to rely on the system's timezone settings. Make sure the
; directive date.timezone is set in php.ini.
date.timezone = Europe/Amsterdam



[pcre]
; More information about pcre :
;http://php.net/manual/en/pcre.configuration.php



[standard]
; This sets the maximum amount of memory in bytes that a script is allowed to
; allocate. This helps prevent poorly written scripts for eating up all available
; memory on a server. It is recommended to set this as low as possible and avoid
; removing the limit.
memory_limit = 120

; This sets the maximum amount of time, in seconds, that a script is allowed to
; run. The lower the value, the better for the server, but also, the better has
; the script to be written. Avoid really large values that are only useful for
; admin, and set them per directory.
max_execution_time = 90

; Exposes to the world that PHP is installed on the server. For security reasons,
; it is better to keep this hidden.
expose_php = Off

; This determines whether errors should be printed to the screen as part of the
; output or if they should be hidden from the user.
display_errors = Off

; Set the error reporting level. Always set this high, so as to have the errors
; reported, and logged.
error_reporting = E_ALL

; Always log errors for future use
log_errors = On

; Name of the file where script errors should be logged.
error_log = Name of a writable file, suitable for logging.

; More information about standard :
```

(continues on next page)

(continued from previous page)

```
;http://php.net/manual/en/info.configuration.php

; Name of the file where script errors should be logged.
disable_functions = curl_init,ftp_connect,ftp_ssl_connect,ldap_connect,mail,mysqli_
↪connect,mysqli_pconnect,pg_connect,pg_pconnect,socket_create,socket_accept,socket_
↪connect,socket_listen
disable_classes = mysqli
```

PhpConfiguration is a Text report format.

PhpConfiguration depends on the following theme : Appinfo.

15.3.26 Phpcity

Phpcity

The Phpcity report represents your code as a city.

Phpcity is a code visualisation tool : it displays the source code as a city, with districts and buildings. Ther will be high sky crappers, signaling large classes, entire districts of small blocks, large venues and isolated parks. Some imagination is welcome too.

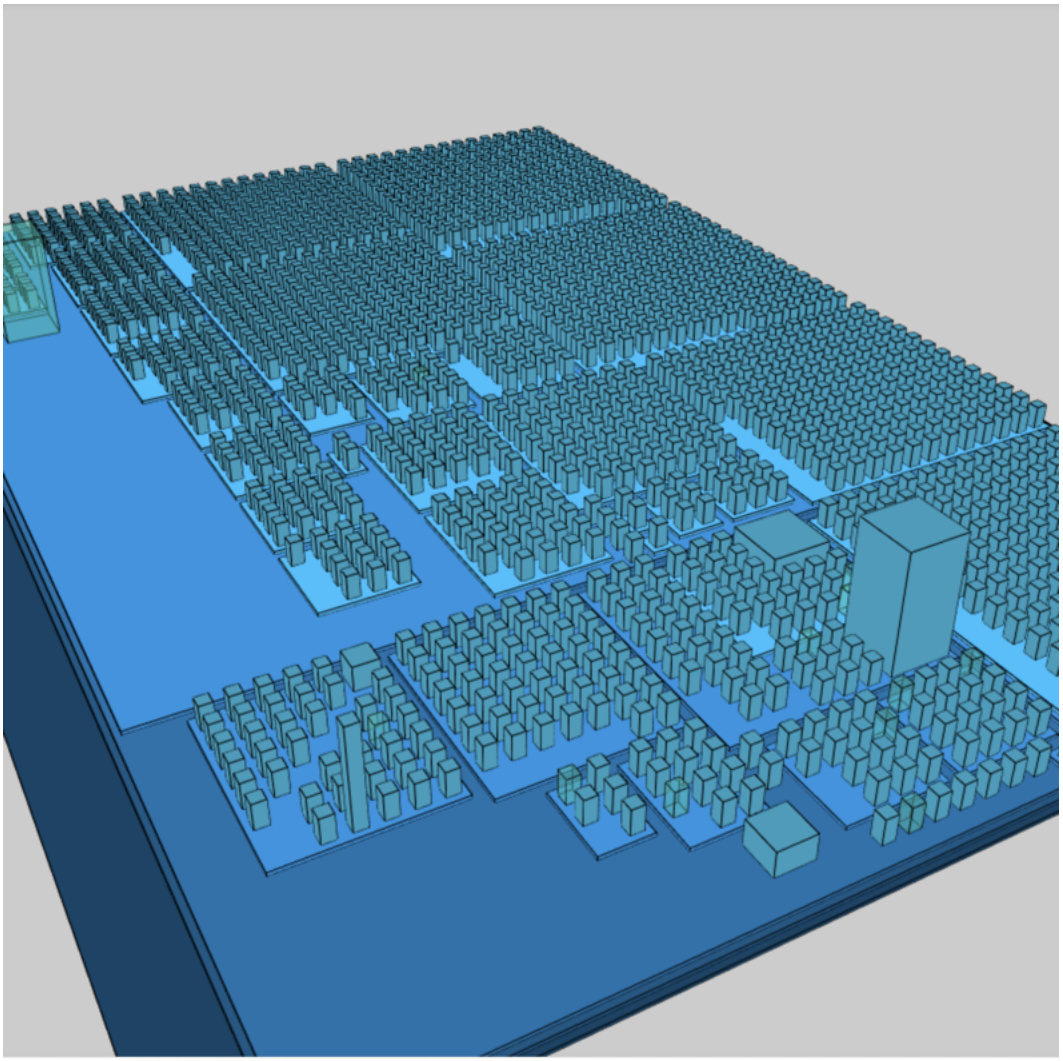
The original idea is Richard Wettel's [Code city](https://wettel.github.io/codecity.html), which has been adapted to many languages, including PHP. The PHP version is based on the open source [PHPcity project](https://github.com/adrianhuna/PHPCity), which is itself build with [JScity](https://github.com/ASERG-UFMG/JSCity/wiki/JSCITY).

To use this tool, run an exakat audit, then generate the 'PHPcity' report : *php exakat.phar report -p mycode -format PHPcity -v*

This generates the *exakat.phpcity.json* file, in the *projects/mycode/* folder.

You may test your own report online, at [Adrian Huna](https://github.com/adrianhuna)'s website, by [uploading the results](https://adrianhuna.github.io/PHPCity/) and seeing it live immediately.

Or, you can install the [PHPcity](https://github.com/adrianhuna/PHPCity) application, and load it locally.



Phpcity is a JSON report format.

Phpcity doesn't depend on themes.

15.3.27 Phpcsfixer

Phpcsfixer

The Phpcsfixer report provides a configuration file for php-cs-fixer, that automatically fixes issues found in related analysis in exakat.

This report builds a configuration file for php-cs-fixer.

- *Use === null* : **is_null**
- Structures/ElseIfElseif : **elseif**
- *Multiple Unset()* : **combine_consecutive_unsets**
- Classes/DontUnsetProperties: **no_unset_on_property**
- Structures/UseConstant : **function_to_constant**

- *PHP7 Dirname* : **combine_nested_dirname**
- Structures/CouldUseDir : **dir_constant**
- *Isset Multiple Arguments* : **combine_consecutive_issets**
- Php/LogicalInLetters : **logical_operators**
- *Not Not* : **no_short_bool_cast**

PHP-cs-fixer is a tool to automatically fix PHP Coding Standards issues. Some of the modifications are more than purely coding standards, such has replacing `dirname(dirname($path))` with `dirname($path, 2)`.

Exakat builds a configuration file for php-cs-fixer, that will automatically fix a number of results from the audit. Here is the process :

- Run exakat audit
- Get Phpcsfixer report from exakat : `php exakat.phar report -p <project> -format Phpcsfixer`
- Update the target repository in the generated code
- Save this new configuration in a file called '.php_cs'
- Run php-cs-fixer on your code : `php php-cs-fixer.phar fix /path/to/code --dry-run`
- Fixed your code with php-cs-fixer : `php php-cs-fixer.phar fix /path/to/code`
- Run a new exakat audit

This configuration file should be reviewed before being used. In particular, the target files should be updated with the actual repository : this is the first part of the configuration.

It is also recommended to use the option '–dry-run' with php-cs-fixer to check the first run.

Php-cs-fixer runs fixes for coding standards : this reports focuses on potential fixes. It is recommended to complete this base report with extra coding conventions fixes. The building of a coding convention is outside the scope of this report.

Exakat may find different issues than php-cs-fixer : using this report reduces the number of reported issues, but may leave some issues unsolved. In that case, manual fixing is recommended.

Phpcsfixer is a JSON report format.

Phpcsfixer depends on the following theme : php-cs-fixable.

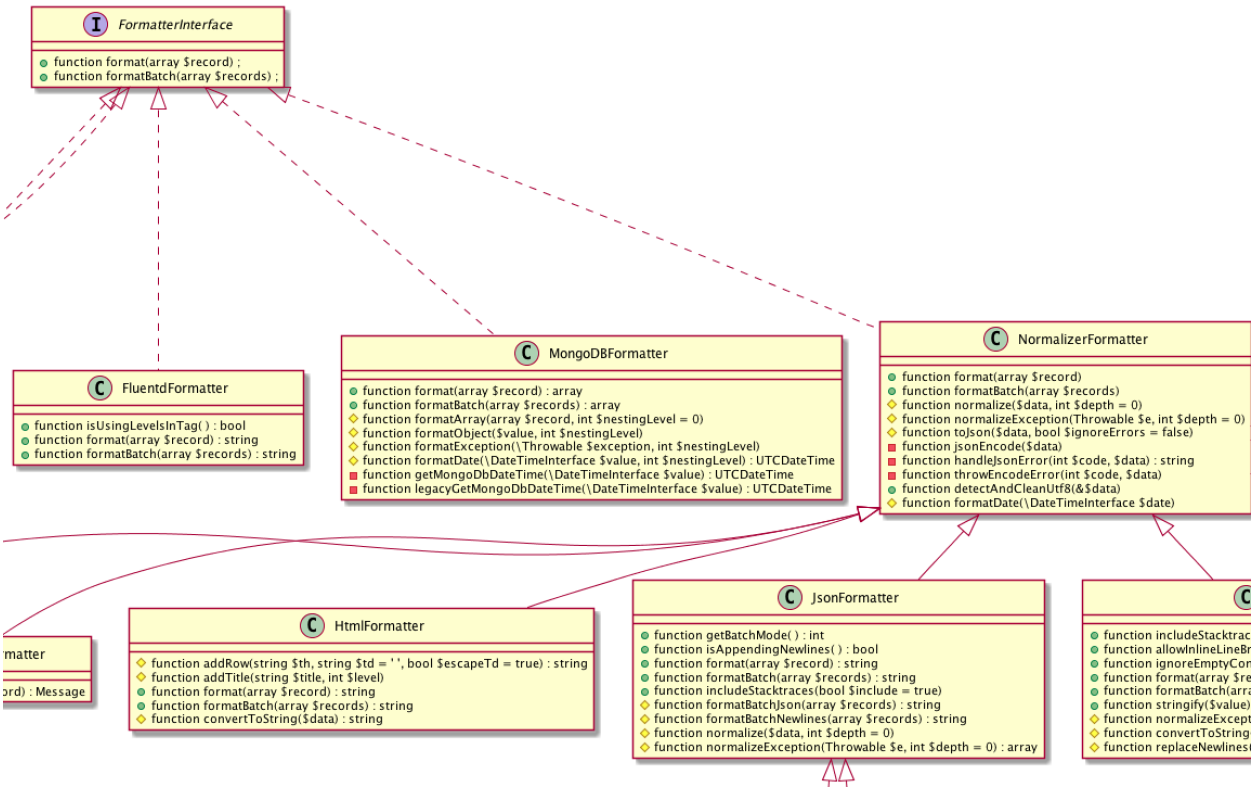
15.3.28 PlantUml

PlantUml

The PlantUml export data structure to PlantUml format.

This report produces a .puml file, compatible with [PlantUML](http://plantuml.com/).

PlantUML is an Open Source component that dislays class diagrams.



PlantUml is a puml report format.

PlantUml doesn't depend on themes.

15.3.29 RadwellCode

RadwellCode

The RadwellCode is a report based on Oliver Radwell's [PHP Do And Don't](https://blog.radwell.codes/2016/11/php-dos-donts-aka-programmers-dont-like/).

Note that all rules are not implemented, especially the 'coding conventions' ones, as this is beyond the scope of this tool.

```
/Phrozn/Vendor/Extra/scss.inc.php:594 Slow PHP built-in functions
/Phrozn/Vendor/Extra/scss.inc.php:2554 Too many nested if statements
/Phrozn/Vendor/Extra/scss.inc.php:1208 Long if-else blocks
/Phrozn/Vendor/Extra/scss.inc.php:1208 Too many nested if statements
/Phrozn/Vendor/Extra/scss.inc.php:3935 Wrong function / class name casing
/Phrozn/Vendor/Extra/scss.inc.php:3452 Too many nested if statements
/Phrozn/Site/View/OutputPath/Entry/Parametrized.php:58 Slow PHP built-in functions
/Phrozn/Runner/CommandLine/Callback/Init.php:82 Extra brackets and braces and quotes
```

RadwellCode is a Text report format.

RadwellCode depends on the following theme : RadwellCodes.

15.3.30 Rector

Rector

Suggest configuration for Rector refactoring tool.

The Rector report is a helper report for [Tomas Votruba](https://twitter.com/VotrubaT)'s [Rector](https://getrector.org/) tool.

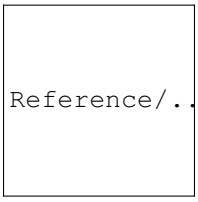
Some issues spotted by Exakat may be fixed automagically by Rector. Rector offers more than 550 (and counting) rules, that may save countless hours of work.

For example, [CombinedAssignRector](https://github.com/rectorphp/rector/blob/master/docs/AllRectorsOverview.md#combinedassignrector), simplifies `$value = $value + 5` into `+$value += 5;`. On Exakat, the rule [Structures/CouldUseShortAssignation]((https://exakat.readthedocs.io/en/latest/Rules.html#could-use-short-assignation) spot those too.

Not all exakat rules are covered by Rector, and vice-versa. [CompactToVariablesRector](https://github.com/rectorphp/rector/blob/master/docs/AllRectorsOverview.md#compacttovariablesrector) aims à skipping usage of compact(), while [Structures/CouldUseCompact](https://exakat.readthedocs.io/en/latest/Rules.html#could-use-compact) suggest the contrary.

Rector and Exakat both use different approaches to code review. It is recommended to review the changes before commiting them.

Check [RectorPHP](https://getrector.org/) website, its [rector github](https://github.com/rectorphp/rector) repository, and [Tomas Votruba](https://twitter.com/VotrubaT) account.



Rector is a Text report format.

Rector depends on the following theme : Rector.

15.3.31 Sarb

Sarb

The Sarb report is a compatibility report with SARB

SARB is the Static Analysis Results Baseliner. SARB is used to create a baseline of these results. As work on the project progresses SARB can takes the latest static analysis results, removes those issues in the baseline and report the issues raised since the baseline. SARB does this, in conjunction with git, by tracking lines of code between commits. SARB is the brainchild of Dave Liddament.


```
[
    {
        "type": "Classes\/NonPpp",
        "file": "\/home\/exakat\/elation\/code\/include\/base_class.php",
        "line": 37
    },
    {
        "type": "Structures\/NoSubstrOne",
        "file": "\/home\/exakat\/elation\/code\/include\/common_funcs.php",
        "line": 890
```


(continues on next page)

(continued from previous page)


```
    },
    {
        "type": "Structures\/DropElseAfterReturn",
        "file": "\/home\/exakat\/elation\/code\/include\/smarty\/SmartyValidate.class.
↪php",
        "line": 638
    },
    {
        "type": "Variables\/UndefinedVariable",
        "file": "\/home\/exakat\/elation\/code\/components\/ui\/ui.php",
        "line": 174
    },
    {
        "type": "Functions\/TooManyLocalVariables",
        "file": "\/home\/exakat\/elation\/code\/include\/dependencymanager_class.php",
        "line": 43
    }
]
```


Sarb is a Json report format.

Sarb accepts any arbitrary list of results.

15.3.32 Sarif

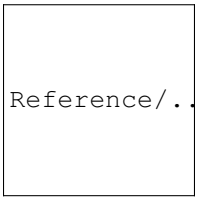
Sarif

The SARIF report publishes the results in SARIF format.

Static Analysis Results Interchange Format (SARIF) a standard format for the output of static analysis tools. The format is referred to as the "Static Analysis Results Interchange Format" and is abbreviated as SARIF.

SARIF is a flexible JSON format, that describes in details the rules, the issues and their context.

More details are available at sarifweb and SARIF support for code scanning at Github.



Sarif is a Json report format.

Sarif accepts any arbitrary list of results.

15.3.33 SimpleTable

SimpleTable

The Simpletable is a simple table presentation.

Simpletable is suitable for any list of results provided by exakat. It is inspired from the Clang report. The result is a HTML file, with Javascript and CSS.

Code	File	Line
(3) Preprocess Arrays		
$transformations = array( )	/src/Behat/Behat/Transformation/Context/Annotation /TransformationAnnotationReader.php	67
$lines = array( )	/src/Behat/Behat/Definition/Printer /ConsoleDefinitionInformationPrinter.php	126
$lines = array( )	/src/Behat/Behat/Definition/Printer /ConsoleDefinitionInformationPrinter.php	77
(2) Ambiguous Static		
$template	/src/Behat/Behat/Context/ContextClass /SimpleClassGenerator.php	25
$template	/src/Behat/Behat/Context/Snippet/ContextSnippet.php	30
(27) Avoid Optional Properties		
(38) Constant Class		
(1) Property Could Be Private Property		
(74) Could Be Private Class Constant		
(232) Property Could Be Private Method		
(78) Could Be Protected Class Constant		
(114) Could Be Protected Method		
(1) No Direct Call To Magic Method		
(16) Class Should Be Final By Ocramius		

SimpleTable is a HTML report format.

SimpleTable doesn't depend on themes.

15.3.34 Stats

Stats

The Stats report collects various stats about the code.

Stats reports PHP structures definition, like class, interfaces, variables, and also features, like operator, control flow instructions, etc.

```
{
    "Summary": {
        "Namespaces": 82,
        "Classes": 59,
        "Interfaces": 29,
        "Trait": 0,
        "Functions": 0,
        "Variables": 4524,
        "Constants": 0
```

(continues on next page)

(continued from previous page)


```
    },
    "Classes": {
        "Classes": 59,
        "Class constants": 10,
        "Properties": 140,
        "Methods": 474
    },
    "Structures": {
        "Ifthen": 568,
        "Else": 76,
        "Switch": 15,
        "Case": 62,
        "Default": 9,
        "Fallthrough": 0,
        "For": 5,
        "Foreach": 102,
        "While": 21,
        "Do..while": 0,
        "New": 106,
        "Clone": 0,
        "Class constant call": 34,
        "Method call": 1071,
        "Static method call": 52,
        "Properties usage": 0,
        "Static property": 65,
        "Throw": 35,
        "Try": 12,
        "Catch": 12,
        "Finally": 0,
        "Yield": 0,
        "Yield From": 0,
        "?  :": 60,
        "?: ": 2,
        "Variables constants": 0,
        "Variables variables": 7,
        "Variables functions": 1,
        "Variables classes": 5
    }
}
```


Stats is a JSON report format.

Stats depends on the following theme : Stats.

15.3.35 Stubs

Stubs

Stubs produces a skeleton from the source code, with all defined structures : constants, functions, classes, interfaces, traits and namespaces.

Stubs takes the original code, and export all defined structures (constants, functions, classes, interfaces, traits and namespaces) in a single and compilable PHP file.

This is convenient for tools that requires documentations for completion, such as IDE.

Constants are exported with their values, properties too. Methods hold their full signature.

The resulting report is in one file, called *stubs.php*.

```php
<?php

namespace  {
    /* No constant definitions */
    /* No function definitions */

    class IndexController extends ViewController  {
        /* No class constants */
        /* No properties */
        /* No methods */
    }

    class Bootstrap extends \yaf\bootstrap_abstract  {
        /* No class constants */
        /* No properties */

        public function _init(Dispatcher $dispatcher) { /**/ }
    }

    class HookPlugin extends \yaf\plugin_abstract  {
        /* No class constants */
        /* No properties */

        public function routerStartup($request, $response) { /**/ }
        public function routerShutdown($request, $response) { /**/ }
        public function dispatchLoopStartup($request, $response) { /**/ }
        public function preDispatch($request, $response) { /**/ }
        public function postDispatch($request, $response) { /**/ }
        public function dispatchLoopShutdown($request, $response) { /**/
        public function preResponse($request, $response) { /**/ }
    }
```

Stubs is a PHP report format.

Stubs doesn't depend on themes.

15.3.36 StubsJson

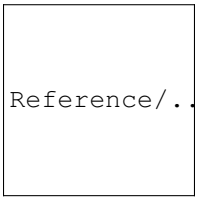
StubsJson

StubsJson produces a complete description of definitions from the code.

The StubsJson report includes :

- Global variables

- Functions
- Constants
- Classes + constants + properties + methods
- Interfaces + constants + methods
- Traits + properties + methods



StubsJson is a JSON report format.

StubsJson doesn't depend on themes.

15.3.37 Text

Text

The Text report is a very simple text format.

The Text report displays one result per line, with the following format :

```
/path/from/project/root/to/file:line[space]name of analysis
```

This format is fast, and fitted for machine communications.

```
/classes/test.php:1002      Php/ShouldUseFunction   Should Use Function     array_
↪values(array_unique(array_merge($classTags, $annotations['tags'])))
/classes/test.php:1002      Php/ShouldUseFunction   Should Use Function     array_
↪merge($classTags, $annotations['tags'])
/classes/test.php:1005      Structures/NoArrayUnique        Avoid array_unique()
↪array_unique(array_merge($classTags, $this->testMethods[$testMethodName]['tags']))
/classes/test.php:1005      Performances/SlowFunctions      Slow Functions  array_
↪unique(array_merge($classTags, $this->testMethods[$testMethodName]['tags']))
```

Text is a Text report format.

Text accepts any arbitrary list of results.

15.3.38 Top10

Top10

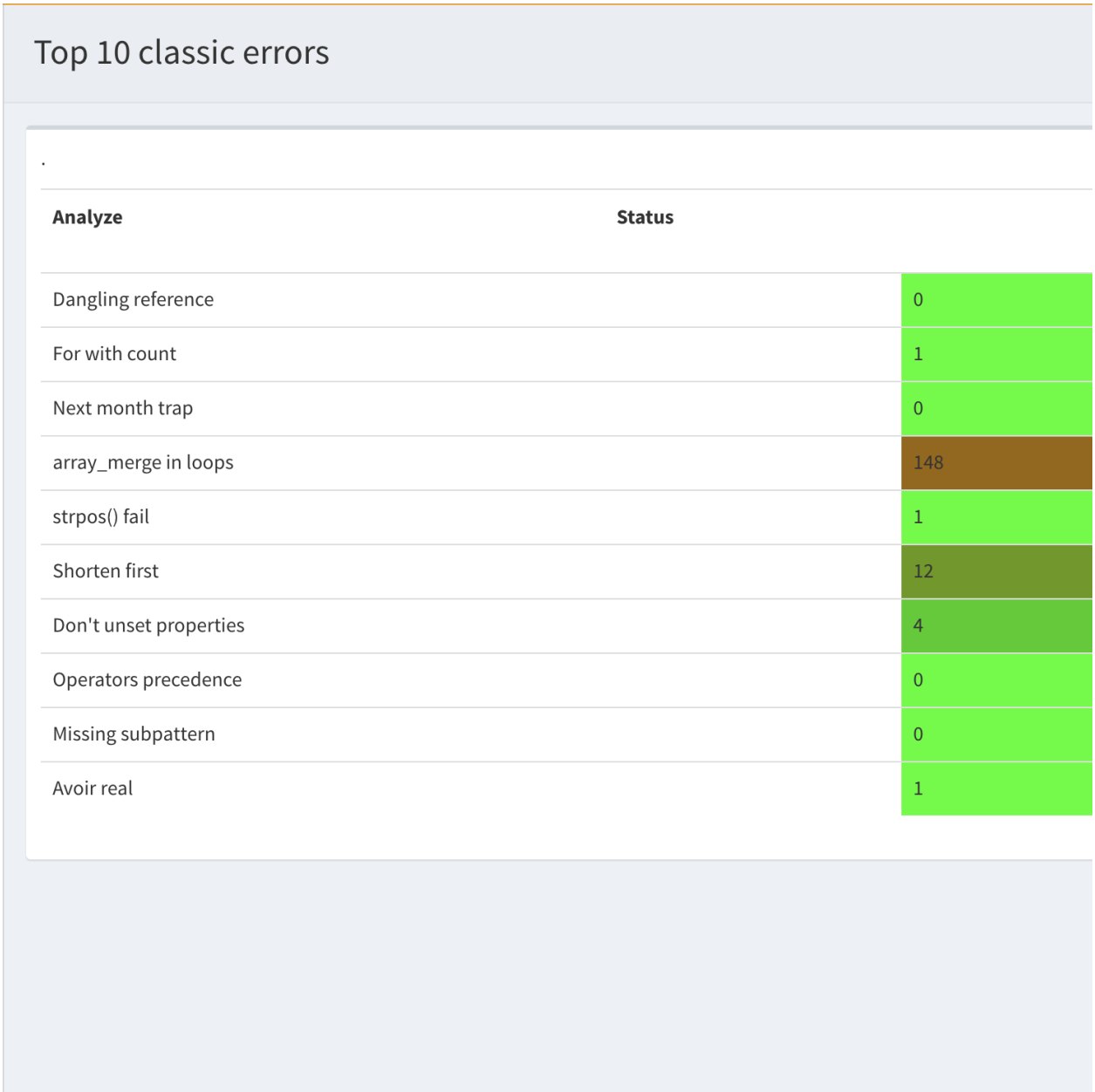
The top 10 is the companion report for the 'Top 10 classic PHP traps' presentation.

The Top 10 report is based on the 'Top 10 classic PHP traps' presentation. You can run it on your code and check immediately where those classic traps are waiting for you. Read the whole presentation online

Top 10 classic errors

.

Analyze	Status
Dangling reference	0
For with count	1
Next month trap	0
array_merge in loops	148
strpos() fail	1
Shorten first	12
Don't unset properties	4
Operators precedence	0
Missing subpattern	0
Avoir real	1



Top 10 classic errors

.

Analyze	Status
Dangling reference	0
For with count	1
Next month trap	0
array_merge in loops	148
strpos() fail	1
Shorten first	12
Don't unset properties	4
Operators precedence	0
Missing subpattern	0
Avoir real	1

Top10 is a HTML report format.

Top10 depends on the following theme : Top10.

15.3.39 Topology Order

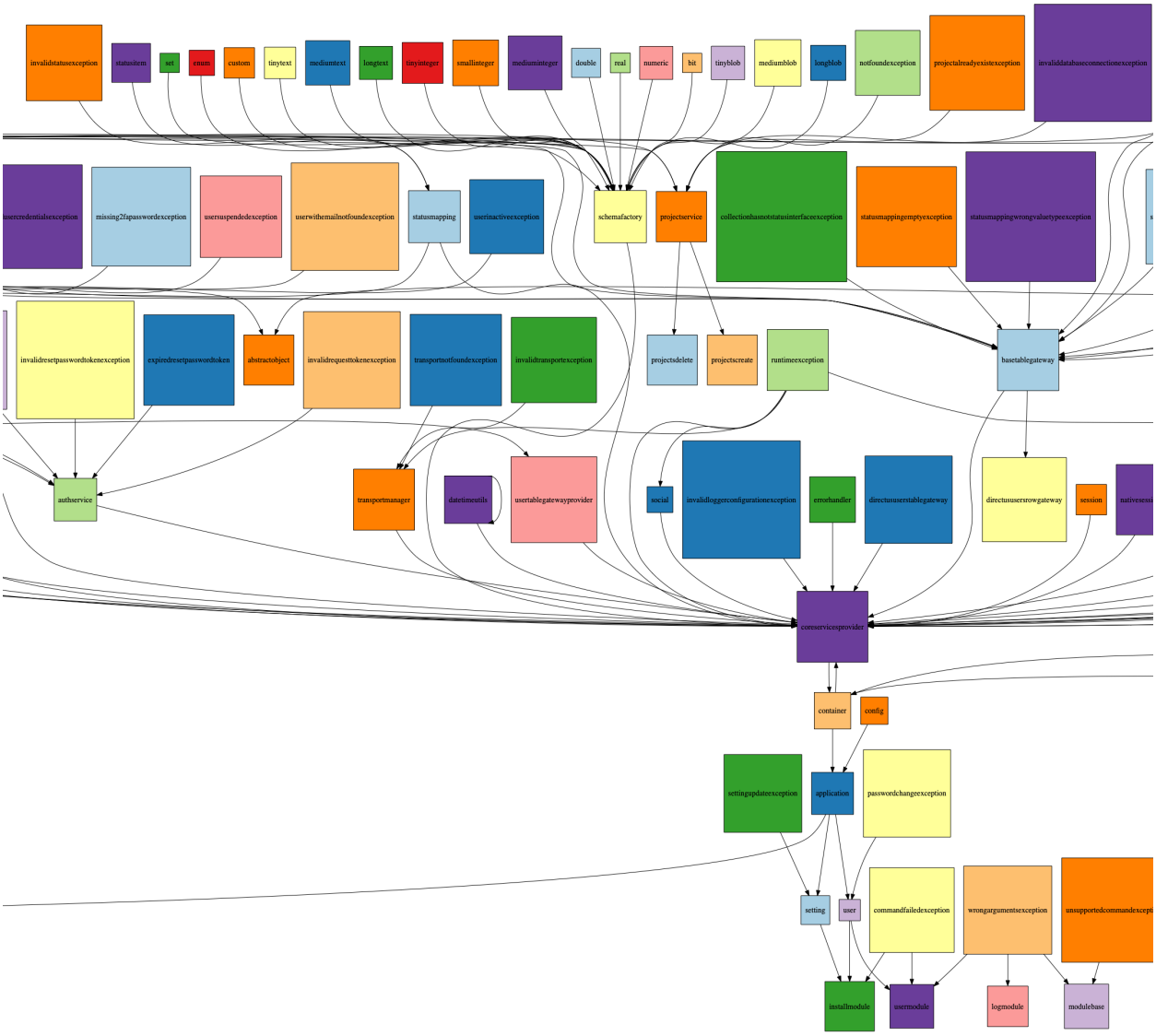
Topology Order

This represents a topological order in the code.

Topology displays all dependencies between code structures. Such dependencies lead to a code hierarchy, which is presented here.

There are currently two topology available:

- Typehint Order : it represents the order in which classes are organized, based on argument and return type.
- New Order : it represents the order in which classes are instantiated, with new.




```
Reference/../images/report.topology.typehints.png
```

Topology Order is a DOT report format.

Topology Order doesn't depend on themes.

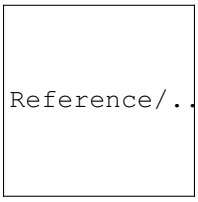
15.3.40 TypeChecks

TypeChecks

The TypeChecks report focuses on reviewing typehint usage.

The TypeChecks report focuses on usage and good usage of typehints.

It checks the presence of typehint, suggests possible type hinting, and check the systemic organisation of the types.



TypeChecks is a HTML report format.

TypeChecks depends on the following theme : TypeChecks.

15.3.41 TypeSuggestion

TypeSuggestion

The TypeSuggestion report provides suggestions to add typehints to methods and properties.

The TypeSuggestion offers suggestions to add typehints to methods and properties.

It provides its suggestion based on the way the code is implemented : by usage or by calling.

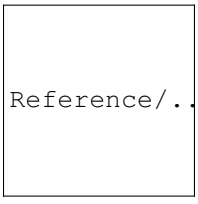
Type usage is the way a typed container is use later. For example, an argument that is used later with the array syntax `$x['a']` or as an object ‘‘$x->b‘‘will receive a suggestion for using array or object.

Type calling is the way the typed container is assigned. For example, a property may receive integer or boolean during assignations : they will receive such suggestions.

Not all types can be guessed : for example, a property may simply hold a value, for later use, such as in a cache system. In such situation, no type is suggested.

`mixed` is not used as suggestion : rather a list of possible types is offered, and it may be upgraded to `mixed`.

This report is ready for PHP 8.0 : the suggestions may be combined together, and multiples suggestions are possible.



TypeSuggestion is a HTML report format.

TypeSuggestion depends on the following theme : TypeChecks.

15.3.42 Uml

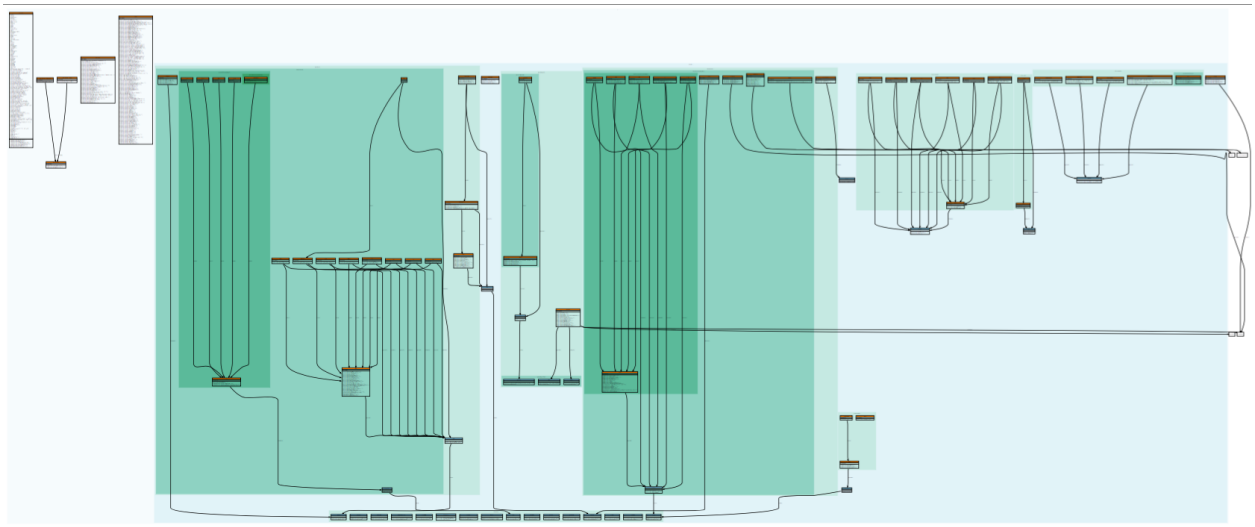
Uml

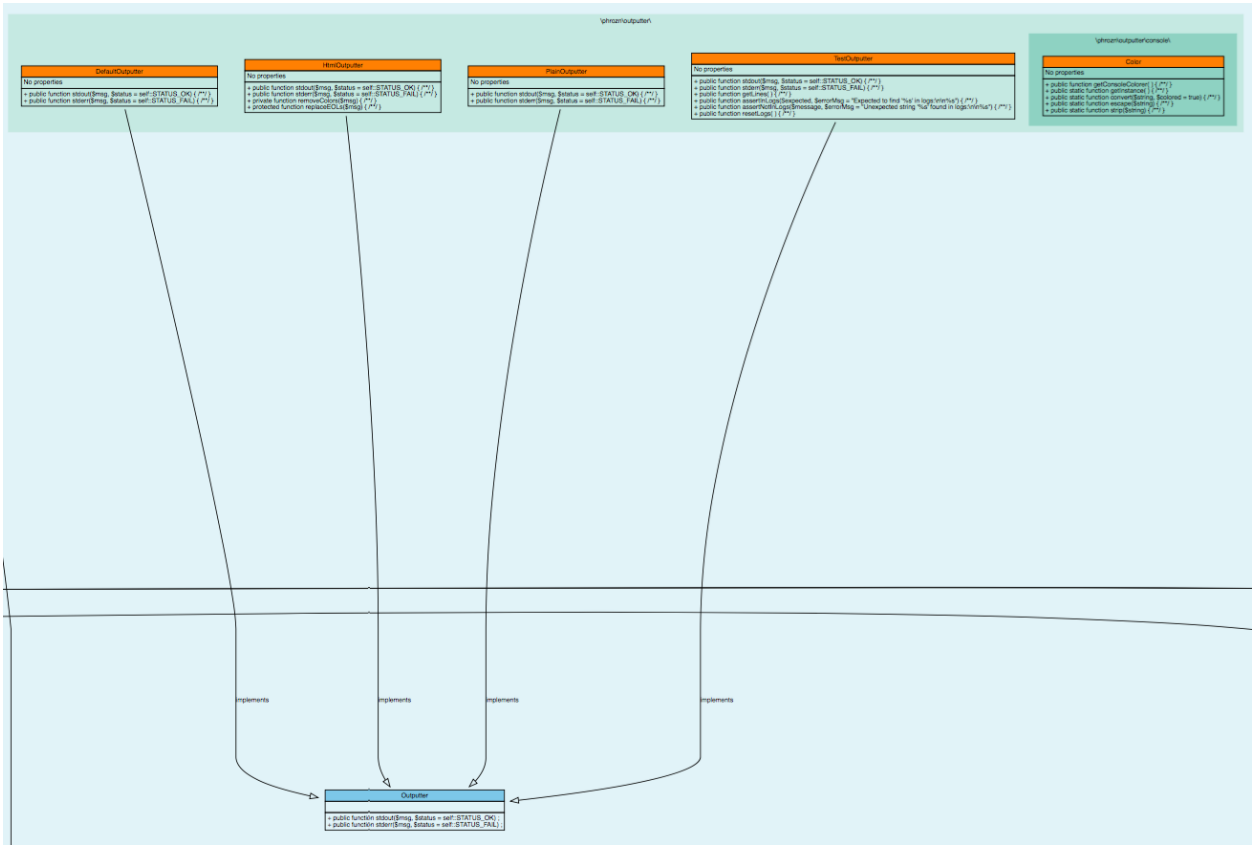
The Uml exports data structure to UML format.

This report produces a dot file with a representation of the classes used in the repository.

Classes, interfaces and traits are represented, along with their constants, methods and properties.

.dot files are best seen with [graphviz](http://www.graphviz.org/) : they are easily convert into PNG or PDF.





Uml is a dot report format.

Uml doesn't depend on themes.

15.3.43 Xml

Xml

The Xml report exports in XML format.

XML version of the reports. It uses the same format than PHP Code Sniffer to output the results.

```
<?xml version="1.0" encoding="UTF-8"?>
<phpcs version="0.8.6">
<file name="/src/NlpTools/Stemmers/PorterStemmer.php" errors="0" warnings="105"␣
↪fixable="0">
 <warning line="55" column="0" source="Php/EllipsisUsage" severity="Major" fixable="0
↪">... Usage</warning>
```

Xml is a XML report format.

Xml accepts any arbitrary list of results.

15.3.44 Yaml

Yaml

The Yaml report exports in Yaml format.

Simple Yaml format. It is a structured array with all results, described as object.

```
Filename => [
                errors   => count,
                warning  => count,
                fixable  => count,
                filename => string,
                message  => [
                    line => [
                        type,
                        source,
                        severity,
                        fixable,
                        message
                    ]
                ]
            ]
```

```
/src/Altax/Module/Task/Resource/RuntimeTask.php:
    errors: 0
    warnings: 22
    fixable: 0
    filename: /src/Altax/Module/Task/Resource/RuntimeTask.php
    messages: { 77: [[{ type: warning, source: Structures/Iffectation, severity:␣
↪Minor, fixable: fixable, message: Iffectations, fullcode: '$args = $this->
↪getArguments( )' }]], 67: [[{ type: warning, source: Structures/Iffectation,␣
↪severity: Minor, fixable: fixable, message: Iffectations, fullcode: '$args = $this->
↪input->getArgument(''args'')' }, { type: warning, source: Structures/
↪BuriedAssignation, severity: Minor, fixable: fixable, message: 'Buried Assignation',
↪ fullcode: '$args = $this->input->getArgument(''args'')' }]], 114: [[{ type:␣
↪warning, source: Variables/WrittenOnlyVariable, severity: Minor, fixable: fixable,␣
↪message: 'Written Only Variables', fullcode: $input }, { type: warning, source:␣
↪Variables/VariableUsedOnceByContext, severity: Minor, fixable: fixable, message:
↪'Used Once Variables (In Scope)', fullcode: $input }, { type: warning, source:␣
↪Classes/UndefinedClasses, severity: Major, fixable: fixable, message: 'Undefined␣
↪Classes', fullcode: 'new ArrayInput($arguments)' }]], 13: [[{ type: warning,␣
```

(continues on next page)

(continued from previous page)

Yaml is a Yaml report format.

Yaml accepts any arbitrary list of results.

CHAPTER 16

Cobblers

16.1 Introduction

Cobblers mend PHP code. They apply a transformation to it.

Cobblers are a complement to code analysis : the analysis spot code to be fixed, the cobbler mends the code. Later, the analysis doesn't find those issues anymore.

16.2 List of Cobblers

16.2.1 Set Typehints

Automagically add scalar typehints to methods and properties. Arguments and return values are both supported.

When multiple possible types are identified, no typehint is added. If a typehint is already set, no typehint is added.

Magic methods, such as __get(), __set(), __construct(), __desctruct(), etc are not modified by this cobbler.

Methods which have parent's methods (resp. children's) are skipped for argument typing (resp return typing) : this may introduce a incompatible definition. On the other hand, methods which have children's methods (resp. parents') are modified for argument typing (resp return typing), thanks to covariance (resp. contravariance).

Void (as a scalar type) and Null types are processed in a separate cobbler.

By default, and in case of conflict, array is chosen over iterable and int is chosen over float. There are parameter to alter this behavior.

Before

```
<?php

class x {
```

(continues on next page)

(continued from previous page)

```
    private int $p = 2;

    function foo(int $a = 1) : int {
        return intdiv($a, $this->p);
    }
}
?>
```

After

```
<?php

class x {
    private int $p = 2;

    function foo(int $a = 1) : int {
        return intdiv($a, $this->p);
    }
}
?>
```

Parameters

Name	Default	Type	Description
array_or_iterable	array	string	When array and iterable are the only suggestions, choose 'array', 'iterable', or 'omit'. By default, it is array.
int_or_float	float	string	When int and float are the only suggestions, choose 'int', 'float', or 'omit'. By default, it is float.

Suggested Analysis

- *Could Be Void*

Related Cobblers

- No anchor for Classes/VarToPublic
- No anchor for Classes/SplitPropertyDefinitions
- No anchor for Functions/SetNullType
- No anchor for Functions/SetTypeVoid

Specs

Short Name	Functions/SetTypehints
Exakat version	2.3.0

16.2.2 Plus One To Pre Plusplus

Transforms a + *1* or - *1* operation into a plus-plus (or minus-minus).

Before

```
<?php
    $a = $a + 1;
?>
```

After

```
<?php
    ++$a;
?>
```

Specs

Short Name	Structures/PlusOneToPre
Exakat version	2.3.0

16.2.3 Post to Pre Plusplus

Transforms a post plus-plus (or minus-minus) operator, into a pre plus-plus (or minus-minus) operator.

Before

```
<?php
    $a++;
?>
```

After

```
<?php
    ++$a;
?>
```

Specs

Short Name	Structures/PostToPre
Exakat version	2.3.0

16.2.4 Remove Noscream @

Removes the @ operator.

Before

```
<?php
    @$a;
?>
```

After

```
<?php
    $a;
?>
```

Suggested Analysis

- *@ Operator*

Reverse Cobbler

- This cobbler is its own reverse.

Specs

Short Name	Structures/RemoveNoScream
Exakat version	2.3.0

CHAPTER 17

Real Code Cases

17.1 Introduction

All the examples in this section are real code, extracted from major PHP applications.

17.2 List of real code Cases

17.2.1 Ambiguous Array Index

PrestaShop

Ambiguous Array Index, in src/PrestaShopBundle/Install/Install.php:532.

Null, as a key, is actually the empty string.

```
$list = array(
            'products' => _PS_PROD_IMG_DIR_,
            'categories' => _PS_CAT_IMG_DIR_,
            'manufacturers' => _PS_MANU_IMG_DIR_,
            'suppliers' => _PS_SUPP_IMG_DIR_,
            'stores' => _PS_STORE_IMG_DIR_,
            null => _PS_IMG_DIR_.'l/', // Little trick to copy images in img/l/ path␣
↪with all types
        );
```

Mautic

Ambiguous Array Index, in app/bundles/CoreBundle/Entity/CommonRepository.php:314.

True is turned into 1 (integer), and false is turned into 0 (integer).

```
foreach ($metadata->getAssociationMappings() as $field => $association) {
                    if (in_array($association['type'], [ClassMetadataInfo::ONE_TO_ONE,
↪ ClassMetadataInfo::MANY_TO_ONE])) {
                        $baseCols[true][$entityClass][]  = $association['joinColumns
↪'][0]['name'];
                        $baseCols[false][$entityClass][] = $field;
                    }
                }
```

17.2.2 Getting Last Element

Thelia

Getting Last Element, in /core/lib/Thelia/Core/Security/AccessManager.php:61.

This code extract the last element with array_slice (position -1) as an array, then get the element in the array with current().

```
current(\array_slice(self::$accessPows, -1, 1, true))
```

17.2.3 Multiple Index Definition

Magento

Multiple Index Definition, in app/code/core/Mage/Adminhtml/Block/System/Convert/Gui/Grid.php:80.

'type' is defined twice. The first one, 'options' is overwritten.

```
$this->addColumn('store_id', array(
            'header'    => Mage::helper('adminhtml')->__('Store'),
            'type'      => 'options',
            'align'     => 'center',
            'index'     => 'store_id',
            'type'      => 'store',
            'width'     => '200px',
        ));
```

MediaWiki

Multiple Index Definition, in resources/Resources.php:223.

'target' is repeated, though with the same values. This is just dead code.

```
// inside a big array
    'jquery.getAttrs' => [
            'targets' => [ 'desktop', 'mobile' ],
            'scripts' => 'resources/src/jquery/jquery.getAttrs.js',
            'targets' => [ 'desktop', 'mobile' ],
    ],
    // big array continues
```

17.2.4 Non-constant Index In Array

Dolibarr

Non-constant Index In Array, in htdocs/includes/OAuth/Common/Storage/DoliStorage.php:245.

The *state* constant in the *$result* array is coming from the SQL query. There is no need to make this a constant : making it a string will remove some warnings in the logs.

```
public function hasAuthorizationState($service)
    {
        // get state from db
        dol_syslog("get state from db");
        $sql = "SELECT state FROM ".MAIN_DB_PREFIX."oauth_state";
        $sql.= " WHERE service='".$this->db->escape($service)."'";
        $resql = $this->db->query($sql);
        $result = $this->db->fetch_array($resql);
        $states[$service] = $result[state];
        $this->states[$service] = $states[$service];

        return is_array($states)
        && isset($states[$service])
        && null !== $states[$service];
    }
```

Zencart

Non-constant Index In Array, in app/library/zencart/Services/src/LeadLanguagesRoutes.php:112.

The *fields* constant in the *$tableEntry* which holds a list of tables. It seems to be a SQL result, but it is conveniently abstracted with *$this->listener->getTableList()*, so we can't be sure.

```
public function updateLanguageTables($insertId)
    {
        $tableList = $this->listener->getTableList();
        if (count($tableList) == 0) {
            return;
        }
        foreach ($tableList as $tableEntry) {
            $languageKeyField = issetorArray($tableEntry, 'languageKeyField',
↪'language_id');
            $sql = " INSERT IGNORE INTO :table: (";
            $sql = $this->dbConn->bindVars($sql, ':table:', $tableEntry ['table'],
↪'noquotestring');
            $sql .= $languageKeyField. ", ";
            $fieldNames = "";
            foreach ($tableEntry[fields] as $fieldName => $fieldType) {
                $fieldNames .= $fieldName . ", ";
            }
```

17.2.5 Randomly Sorted Arrays

Contao

Randomly Sorted Arrays, in system/modules/core/dca/tl_module.php:259.

The array array('maxlength', 'decodeEntities', 'tl_class') is configured multiple times in this file. Most of them is in the second form, but some are in the first form. (Multiple occurrences in this file).

```
array('maxlength' => 255, 'decodeEntities' => true, 'tl_class' => 'w50') // Line 246
array('decodeEntities' => true, 'maxlength' => 255, 'tl_class' => 'w50'); // ligne 378
```

Vanilla

Randomly Sorted Arrays, in applications/dashboard/models/class.activitymodel.php:308.

'Photo' moved from last to second. This array is used with a 'Join' key, and is the base for a SQL table JOIN. As such, order is important. If this is the case, it seems unusual that the order is not the same for a join using the same tables. If it is not the case, arrays may be reordered.

```
/* L 305 */        Gdn::userModel()->joinUsers(
            $result->resultArray(),
            ['ActivityUserID', 'RegardingUserID'],
            ['Join' => ['Name', 'Email', 'Gender', 'Photo']]
        );

// L 385
        Gdn::userModel()->joinUsers($result, ['ActivityUserID', 'RegardingUserID'], [
↪'Join' => ['Name', 'Photo', 'Email', 'Gender']]);
```

17.2.6 Slice Arrays First

WordPress

Slice Arrays First, in modules/InboundEmail/InboundEmail.php:1080.

Instead of reading ALL the keys, and then, keeping only the first fifty, why not read the 50 first items from the array, and then extract the keys?

```
$results = array_slice(array_keys($diff), 0 ,50);
```

17.2.7 Abstract Or Implements

Zurmo

Abstract Or Implements, in app/protected/extensions/zurmoinc/framework/views/MassEditProgressView.php:30.

The class MassEditProgressView extends ProgressView, which is an abstract class. That class defines one abstract method : abstract protected function headerLabelPrefixContent(). Yet, the class MassEditProgressView doesn't implements this method. This means that the class can't be instatiated, and indeed, it isn't. The class MassEditProgressView is subclassed, by the class MarketingListMembersMassSubscribeProgressView, which implements the method headerLabelPrefixContent(). As such, MassEditProgressView should be marked abstract, so as to prevent any instantiation attempt.

```
class MassEditProgressView extends ProgressView {
    /**/
}
```

17.2.8 Ambiguous Visibilities

Typo3

Ambiguous Visibilities, in typo3/sysext/backend/Classes/Controller/NewRecordController.php:90.

$allowedNewTables is declared once protected and once public. $allowedNewTables is rare : 2 occurences. This may lead to confusion about access to this property.

```
class NewRecordController
{
/.. many lines../
    /**
     * @var array
     */
    protected $allowedNewTables;

class DatabaseRecordList
{
/..../
    /**
     * Used to indicate which tables (values in the array) that can have a
     * create-new-record link. If the array is empty, all tables are allowed.
     *
     * @var string[]
     */
    public $allowedNewTables = [];
```

17.2.9 Avoid Optional Properties

ChurchCRM

Avoid Optional Properties, in src/ChurchCRM/BackupManager.php:401.

Backuptype is initialized with null, and yet, it isn't checked for any invalid valid values, in particular in switch() structures.

```
// BackupType is initialized with null
  class JobBase
  {
      /**
       *
       * @var BackupType
       */
      protected $BackupType;

// In the child class BackupJob, BackupType may be of any type
  class BackupJob extends JobBase
  {
      /**
       *
       * @param String $BaseName
       * @param BackupType $BackupType
       * @param Boolean $IncludeExtraneousFiles
       */
      public function __construct($BaseName, $BackupType, $IncludeExtraneousFiles,
↪$EncryptBackup, $BackupPassword)
```

(continues on next page)

(continued from previous page)

```
      {
          $this->BackupType = $BackupType;


// Later, Backtype is not checked with all values :
          try {
              $this->DecryptBackup();
              switch ($this->BackupType) {
              case BackupType::SQL:
                $this->RestoreSQLBackup($this->RestoreFile);
                break;
              case BackupType::GZSQL:
                $this->RestoreGZSQL();
                break;
              case BackupType::FullBackup:
                $this->RestoreFullBackup();
                break;
// Note  : no default case here
            }
```

Dolibarr

Avoid Optional Properties, in htdocs/product/stock/class/productlot.class.php:149.

$this->fk_product is tested for value 11 times while being used in this class. All detected situations were checking the presence of the property before usage.

```
class Productlot extends CommonObject
{
// more code
    /**
     * @var int ID
     */
    public $fk_product;

// Checked usage of fk_product
// line 341
            $sql .= ' fk_product = '.(isset($this->fk_product) ? $this->fk_product :
↪"null").',';
```

17.2.10 Cant Instantiate Class

WordPress

Cant Instantiate Class, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

17.2.11 Class, Interface Or Trait With Identical Names

shopware

Class, Interface Or Trait With Identical Names, in engine/Shopware/Components/Form/Interfaces/Element.php:30.

Most Element classes extends ModelEntity, which is an abstract class. There is also an interface, called Element, for forms. And, last, one of the class Element extends JsonSerializable, which is a PHP native interface. Namespaces are definitely crucial to understand which Element is which.

```
interface Element { /**/ } // in engine/Shopware/Components/Form/Interfaces/Element.
↪php:30

class Element implements \JsonSerializable { /**/ }          // in engine/Shopware/
↪Bundle/EmotionBundle/Struct/Element.php:29

class Element extends ModelEntity { /**/ }  // in /engine/Shopware/Models/Document/
↪Element.php:37
```

NextCloud

Class, Interface Or Trait With Identical Names, in lib/private/Files/Storage/Storage.php:33.

Interface Storage extends another Storage class. Here, the fully qualified name is used, so we can understand which storage is which at read time : a 'use' alias would make this line more confusing.

```
interface Storage extends \OCP\Files\Storage { /**/ }
```

17.2.12 Could Be Abstract Class

Edusoho

Could Be Abstract Class, in src/Biz/Task/Strategy/BaseStrategy.php:14.

BaseStrategy is extended by NormalStrategy, DefaultStrategy (Not shown here), but it is not instantiated itself.

```
class BaseStrategy {
    // Class code
}
```

shopware

Could Be Abstract Class, in engine/Shopware/Plugins/Default/Core/PaymentMethods/Components/GenericPaymentMethod.php:31.

A 'Generic' class sounds like a class that could be 'abstract'.

```
class GenericPaymentMethod extends BasePaymentMethod {
    // More class code
}
```

17.2.13 Could Be Private Class Constant

Phinx

Could Be Private Class Constant, in src/Phinx/Db/Adapter/MysqlAdapter.php:46.

The code includes a fair number of class constants. The one listed here are only used to define TEXT columns in MySQL, with their maximal size. Since they are only intented to be used by the MySQL driver, they may be private.

```
class MysqlAdapter extends PdoAdapter implements AdapterInterface
{

//.....
    const TEXT_SMALL   = 255;
    const TEXT_REGULAR = 65535;
    const TEXT_MEDIUM  = 16777215;
    const TEXT_LONG    = 4294967295;
```

17.2.14 Method Could Be Static

FuelCMS

Method Could Be Static, in fuel/modules/fuel/models/Fuel_assets_model.php:240.

This method makes no usage of $this : it only works on the incoming argument, $file. This may even be a function.

```
public function get_file($file)
    {
            // if no extension is provided, then we determine that it needs to be␣
↪decoded
            if (strpos($file, '.') === FALSE)
            {
                    $file = uri_safe_decode($file);
            }
            return $file;
    }
```

ExpressionEngine

Method Could Be Static, in system/ee/legacy/libraries/Upload.ph:859.

This method returns the list of mime type, by using a hidden global value : ee() is a functioncall that give access to the external storage of values.

```
/**
     * List of Mime Types
     *
     * This is a list of mime types.  We use it to validate
     * the allowed types set by the developer
     *
     * @param       string
     * @return      string
     */
    public function mimes_types($mime)
    {
            ee()->load->library('mime_type');
            return ee()->mime_type->isSafeForUpload($mime);
    }
```

17.2.15 Disconnected Classes

WordPress

Disconnected Classes, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

17.2.16 Don't Send $this In Constructor

Woocommerce

Don't Send $this In Constructor, in includes/class-wc-cart.php:107.

WC_Cart_Session and WC_Cart_Fees receives $this, the current object, at a moment where it is not consistent : for example, tax_display_cart hasn't been set yet. Although it may be unexpected to have an object called WC_Cart being called by the session or the fees, this is still a temporary inconsistence.

```
/**
     * Constructor for the cart class. Loads options and hooks in the init method.
     */
    public function __construct() {
            $this->session          = new WC_Cart_Session( $this );
            $this->fees_api         = new WC_Cart_Fees( $this );
            $this->tax_display_cart = $this->is_tax_displayed();

            // Register hooks for the objects.
            $this->session->init();
```

Contao

Don't Send $this In Constructor, in system/modules/core/library/Contao/Model.php:110.

$this is send to $objRegistry. $objRegistry is obtained with a factory, ModelRegistry::getInstance(). It is probably fully prepared at that point. Yet, $objRegistry is called and used to fill $this properties with full values. At some point, $objRegistry return values without having a handle on a fully designed object.

```
/**
     * Load the relations and optionally process a result set
     *
     * @param \Database\Result $objResult An optional database result
     */
    public function __construct(\Database\Result $objResult=null)
    {
        // Some code was removed
                    $objRegistry = \Model\Registry::getInstance();

                    $this->setRow($arrData); // see #5439
                    $objRegistry->register($this);

        // More code below
        // $this-> are set
```

(continues on next page)

(continued from previous page)

```
        // $objRegistry is called
    }
```

17.2.17 Don't Unset Properties

Vanilla

Don't Unset Properties, in applications/dashboard/models/class.activitymodel.php:1073.

The _NotificationQueue property, in this class, is defined as an array. Here, it is destroyed, then recreated. The unset() is too much, as the assignation is sufficient to reset the array

```
/**
     * Clear notification queue.
     *
     * @since 2.0.17
     * @access public
     */
    public function clearNotificationQueue() {
        unset($this->_NotificationQueue);
        $this->_NotificationQueue = [];
    }
```

Typo3

Don't Unset Properties, in typo3/sysext/linkvalidator/Classes/Linktype/InternalLinktype.php:73.

The property errorParams is emptied by unsetting it. The property is actually defined in the above class, as an array. Until the next error is added to this list, any access to the error list has to be checked with isset(), or yield an 'Undefined' warning.

```
public function checkLink($url, $softRefEntry, $reference)
    {
        $anchor = '';
        $this->responseContent = true;
        // Might already contain values - empty it
        unset($this->errorParams);
//....

abstract class AbstractLinktype implements LinktypeInterface
{
    /**
     * Contains parameters needed for the rendering of the error message
     *
     * @var array
     */
    protected $errorParams = [];
```

17.2.18 Empty Classes

WordPress

Empty Classes, in wp-includes/SimplePie/Core.php:54.

Empty class, but documented as backward compatibility.

```
/**
 * SimplePie class.
 *
 * Class for backward compatibility.
 *
 * @deprecated Use {@see SimplePie} directly
 * @package SimplePie
 * @subpackage API
 */
class SimplePie_Core extends SimplePie
{

}
```

17.2.19 Incompatible Signature Methods

SuiteCrm

Incompatible Signature Methods, in modules/Home/Dashlets/RSSDashlet/RSSDashlet.php:138.

The class in the RSSDashlet.php file has an 'array' typehint which is not in the parent Dashlet class. While both files compile separately, they yield a PHP warning when running : typehinting mismatch only yields a warning.

```
// File /modules/Home/Dashlets/RSSDashlet/RSSDashlet.php
    public function saveOptions(
        array $req
        )
    {

// File /include/Dashlets/Dashlets.php
    public function saveOptions( $req ) {
```

17.2.20 Incompatible Signature Methods With Covariance

SuiteCrm

Incompatible Signature Methods With Covariance, in modules/Home/Dashlets/RSSDashlet/RSSDashlet.php:138.

The class in the RSSDashlet.php file has an 'array' typehint which is not in the parent Dashlet class. While both files compile separately, they yield a PHP warning when running : typehinting mismatch only yields a warning.

```
// File /modules/Home/Dashlets/RSSDashlet/RSSDashlet.php
    public function saveOptions(
        array $req
        )
    {

// File /include/Dashlets/Dashlets.php
    public function saveOptions( $req ) {
```

17.2.21 Assign Default To Properties

LiveZilla

Assign Default To Properties, in livezilla/_lib/functions.external.inc.php:174.

Flags may default to array() in the class definition. Filled array(), with keys and values, are also possible.

```
class OverlayChat
{
    public $Botmode;
    public $Human;
    public $HumanGeneral;
    public $RepollRequired;
    public $OperatorCount;
    public $Flags;
    public $LastMessageReceived;
    public $LastPostReceived;
    public $IsHumanChatAvailable;
    public $IsChatAvailable;
    public $ChatHTML;
    public $OverlayHTML;
    public $PostHTML;
    public $FullLoad;
    public $LanguageRequired = false;
    public $LastPoster;
    public $EyeCatcher;
    public $GroupBuilder;
    public $CurrentOperatorId;
    public $BotTitle;
    public $OperatorPostCount;
    public $PlaySound;
    public $SpeakingToHTML;
    public $SpeakingToAdded;
    public $Version = 1;

    public static $MaxPosts = 50;
    public static $Response;

    function __construct()
    {
        $this->Flags = array();
        VisitorChat::$Router = new ChatRouter();
    }
```

phpMyAdmin

Assign Default To Properties, in libraries/classes/Console.ph:55.

_isEnabled may default to true. It could also default to a class constant.

```
class Console
{
    /**
     * Whether to display anything
     *
     * @access private
```

(continues on next page)

(continued from previous page)

```
     * @var bool
     */
    private $_isEnabled;

// some code ignored here
    /**
     * Creates a new class instance
     */
    public function __construct()
    {
        $this->_isEnabled = true;
```

17.2.22 Forgotten Visibility

FuelCMS

Forgotten Visibility, in /fuel/modules/fuel/controllers/Module.php:713.

Missing visibility for the index() method,and all the methods in the Module class.

```
class Module extends Fuel_base_controller {

    // --------------------------------------------------------------------

    /**
     * Displays the list (table) view
     *
     * @access      public
     * @return      void
     */
    function index()
    {
            $this->items();
    }
```

LiveZilla

Forgotten Visibility, in livezilla/_lib/objects.global.users.inc.php:2516.

Static method that could be public.

```
class Visitor extends BaseUser
{
// Lots of code

    static function CreateSPAMFilter($_userId,$_base64=true)
    {
        if(!empty(Server::$Configuration->File[gl_sfa]))
        {
```

17.2.23 Non Static Methods Called In A Static

Dolphin

Non Static Methods Called In A Static, in Dolphin-v.7.3.5/xmlrpc/BxDolXMLRPCFriends.php:11.

getIdByNickname() is indeed defined in the class 'BxDolXMLRPCUtil' and it calls the database. The class relies on functions (not methods) to query the database with the correct connexion.

```
class BxDolXMLRPCFriends
{
    function getFriends($sUser, $sPwd, $sNick, $sLang)
    {
        $iIdProfile = BxDolXMLRPCUtil::getIdByNickname ($sNick);
```

Magento

Non Static Methods Called In A Static, in app/code/core/Mage/Paypal/Model/Payflowlink.php:143.

Mage_Payment_Model_Method_Abstract is an abstract class : this way, it is not possible to instantiate it and then, access its methods. The class is extended, so it could be called from one of the objects. Although, the troubling part is that isAvailable() uses $this, so it can't be static.

```
Mage_Payment_Model_Method_Abstract::isAvailable($quote)
```

17.2.24 Var Keyword

xataface

Var Keyword, in SQL/Parser/wrapper.php:24.

With the usage of var and a first method bearing the name of the class, this is PHP 4 code that is still in use.

```
class SQL_Parser_wrapper {

    var $_data;
    var $_tableLookup;
    var $_parser;

    function SQL_Parser_wrapper(&$data, $dialect='MySQL'){
```

17.2.25 Parent First

shopware

Parent First, in wp-admin/includes/misc.php:74.

Here, the parent is called last. Givent that $title is defined in the same class, it seems that $name may be defined in the BaseContainer class. In fact, it is not, and BasecContainer and FieldSet are fairly independant classes. Thus, the parent::__construct call could be first here, though more as a coding convention.

```
/**
 * Class FieldSet
 */
class FieldSet extends BaseContainer
```

(continues on next page)

(continued from previous page)

```
{
    /**
     * @var string
     */
    protected $title;

    /**
     * @param string $name
     * @param string $title
     */
    public function __construct($name, $title)
    {
        $this->title = $title;
        $this->name = $name;
        parent::__construct();
    }
```

PrestaShop

Parent First, in controllers/admin/AdminPatternsController.php:30.

A good number of properties are set in the current object even before the parent AdminController(Core) is called. 'table' and 'lang' acts as default values for the parent class, as it (the parent class) would set them to another default value. Many properties are used, but not defined in the current class, nor its parent. This approach prevents the constructor from requesting too many arguments. Yet, as such, it is difficult to follow which of the initial values are transmitted via protected/public properties rather than using the __construct() call.

```
class AdminPatternsControllerCore extends AdminController
{
    public $name = 'patterns';

    public function __construct()
    {
        $this->bootstrap = true;
        $this->show_toolbar = false;
        $this->context = Context::getContext();

        parent::__construct();
    }
```

17.2.26 Property Could Be Local

Mautic

Property Could Be Local, in app/bundles/EmailBundle/Model/SendEmailToContact.php:47.

$translator is a private property, provided at construction time. It is private, and only used in the processBadEmails() method. $translator may be turned into a parameter for processBadEmails(), and make the class slimmer.

```
class SendEmailToContact
{
    /**
     * @var TranslatorInterface
```

(continues on next page)

(continued from previous page)

```
     */
    private $translator;

// Skipped code

    /**
     * SendEmailToContact constructor.
     *
     * @param MailHelper          $mailer
     * @param StatRepository      $statRepository
     * @param DoNotContact        $dncModel
     * @param TranslatorInterface $translator
     */
    public function __construct(MailHelper $mailer, StatHelper $statHelper,␣
↪DoNotContact $dncModel, TranslatorInterface $translator)
    {
        $this->mailer     = $mailer;
        $this->statHelper = $statHelper;
        $this->dncModel   = $dncModel;
        $this->translator = $translator;
    }

// Skipped code

    /**
     * Add DNC entries for bad emails to get them out of the queue permanently.
     */
    protected function processBadEmails()
    {
        // Update bad emails as bounces
        if (count($this->badEmails)) {
            foreach ($this->badEmails as $contactId => $contactEmail) {
                $this->dncModel->addDncForContact(
                    $contactId,
                    ['email' => $this->emailEntityId],
                    DNC::BOUNCED,
                    $this->translator->trans('mautic.email.bounce.reason.bad_email'),
                    true,
                    false
                );
            }
        }
    }
```

Typo3

Property Could Be Local, in typo3/sysext/install/Classes/Updates/MigrateUrlTypesInPagesUpdate.php:28.

$urltypes is a private property, with a list of protocols for communicationss. It acts as a constant, being only read in the executeUpdate() method : constants may hold arrays. If this property has to evolve in the future, an accessor to update it will be necessary. Until then, this list may be hardcoded in the method.

```
/**
 * Merge URLs divided in pages.urltype and pages.url into pages.url
 * @internal This class is only meant to be used within EXT:install and is not part␣
↪of the TYPO3 Core API.
```

(continues on next page)

(continued from previous page)

```
 */
class MigrateUrlTypesInPagesUpdate implements UpgradeWizardInterface
{
    private $urltypes = ['', 'http://', 'ftp://', 'mailto:', 'https://'];

// Skipped code

    /**
     * Moves data from pages.urltype to pages.url
     *
     * @return bool
     */
    public function executeUpdate(): bool
    {
        foreach ($this->databaseTables as $databaseTable) {
            $connection = GeneralUtility::makeInstance(ConnectionPool::class)
                ->getConnectionForTable($databaseTable);

            // Process records that have entries in pages.urltype
            $queryBuilder = $connection->createQueryBuilder();
            $queryBuilder->getRestrictions()->removeAll();
            $statement = $queryBuilder->select('uid', 'urltype', 'url')
                ->from($databaseTable)
                ->where(
                    $queryBuilder->expr()->neq('urltype', 0),
                    $queryBuilder->expr()->neq('url', $queryBuilder->
↪createPositionalParameter(''))
                )
                ->execute();

            while ($row = $statement->fetch()) {
                $url = $this->urltypes[(int)$row['urltype']] . $row['url'];
                $updateQueryBuilder = $connection->createQueryBuilder();
                $updateQueryBuilder
                    ->update($databaseTable)
                    ->where(
                        $updateQueryBuilder->expr()->eq(
                            'uid',
                            $updateQueryBuilder->createNamedParameter($row['uid'],␣
↪\PDO::PARAM_INT)
                        )
                    )
                    ->set('url', $updateQueryBuilder->createNamedParameter($url),␣
↪false)
                    ->set('urltype', 0);
                $updateQueryBuilder->execute();
            }
        }
        return true;
    }
```

17.2.27 Never Used Properties

WordPress

Never Used Properties, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

17.2.28 Property Used In One Method Only

Contao

Property Used In One Method Only, in calendar-bundle/src/Resources/contao/modules/ModuleEventlist.php:38.

Date is protected property. It is used only in the compile() method, and it is not used by the parent class. As such, it may be turned into a local variable.

```
class ModuleEventlist extends Events
{

    /**
     * Current date object
     * @var Date
     */
    protected $Date;

// Date is used in function compile() only
```

17.2.29 Redefined Default

Piwigo

Redefined Default, in admin/include/updates.class.php:34.

default_themes is defined as an empty array, then filled with new values. Same for default_plugins. Both may be defined as declaration time, and not during the constructor.

```
class updates
{
  var $types = array();
  var $plugins;
  var $themes;
  var $languages;
  var $missing = array();
  var $default_plugins = array();
  var $default_themes = array();
  var $default_languages = array();
  var $merged_extensions = array();
  var $merged_extension_url = 'http://piwigo.org/download/merged_extensions.txt';

  function __construct($page='updates')
  {
    $this->types = array('plugins', 'themes', 'languages');
```

(continues on next page)

(continued from previous page)

```
    if (in_array($page, $this->types))
    {
      $this->types = array($page);
    }
    $this->default_themes = array('clear', 'dark', 'Sylvia', 'elegant', 'smartpocket
↪');
    $this->default_plugins = array('AdminTools', 'TakeATour', 'language_switch',
↪'LocalFilesEditor');
```

17.2.30 Redefined Private Property

Zurmo

Redefined Private Property, in app/protected/modules/zurmo/models/OwnedCustomField.php:51.

The class OwnedCustomField is part of a large class tree : OwnedCustomField extends CustomField, CustomField extends BaseCustomField, BaseCustomField extends RedBeanModel, RedBeanModel extends BeanModel.

Since $canHaveBean is distinct in BeanModel and in OwnedCustomField, the public method getCanHaveBean() also had to be overloaded.

```
class OwnedCustomField extends CustomField
    {
        /**
         * OwnedCustomField does not need to have a bean because it stores no␣
↪attributes and has no relations
         * @see RedBeanModel::canHaveBean();
         * @var boolean
         */
        private static $canHaveBean = false;

/..../

        /**
         * @see RedBeanModel::getHasBean()
         */
        public static function getCanHaveBean()
        {
            if (get_called_class() == 'OwnedCustomField')
            {
                return self::$canHaveBean;
            }
            return parent::getCanHaveBean();
        }
```

17.2.31 Scalar Or Object Property

SugarCrm

Scalar Or Object Property, in SugarCE-Full-6.5.26/data/Link.php:54.

The _relationship property starts its life as a string, and becomes an object later.

```
class Link {

    /* Private variables.*/
    var $_log;
    var $_relationship_name; //relationship this attribute is tied to.
    var $_bean; //stores a copy of the bean.
    var $_relationship= '';

/// More code.....

// line 92
            $this->_relationship=new Relationship();
```

17.2.32 Could Use self

WordPress

Could Use self, in wp-admin/includes/misc.php:74.

Securimage could be called self.

```
class Securimage
{
// Lots of code
            Securimage::$_captchaId = $id;
}
```

LiveZilla

Could Use self, in livezilla/_lib/objects.global.users.inc.php:1599.

Using self makes it obvious that Operator::GetSystemId() is a local call, while Communication::GetParameter() is external.

```
class Operator extends BaseUser
{
    static function ReadParams()
    {
        if(!empty($_POST[POST_EXTERN_REQUESTED_INTERNID]))
            return Communication::GetParameter(POST_EXTERN_REQUESTED_INTERNID,,$c,
↪FILTER_SANITIZE_SPECIAL_CHARS,null,32);
        else if(!empty($_GET[operator]))
        {
            $userid = Communication::GetParameter(operator,,$c,FILTER_SANITIZE_
↪SPECIAL_CHARS,null,32,false,false);
            $sysid = Operator::GetSystemId($userid);
}
```

17.2.33 Static Methods Can't Contain $this

xataface

Static Methods Can't Contain $this, in Dataface/LanguageTool.php:48.

$this is hidden in the arguments of the static call to the method.

```
public static function loadRealm($name){
            return self::getInstance($this->app->_conf['default_language'])->
↪loadRealm($name);
    }
```

SugarCrm

Static Methods Can't Contain $this, in SugarCE-Full-6.5.26/modules/ACLActions/ACLAction.php:332.

Notice how $this is tested for existence before using it. It seems strange, at first, but we have to remember that if $this is never set when calling a static method, a static method may be called with $this. Confusingly, this static method may be called in two ways.

```
static function hasAccess($is_owner=false, $access = 0){

        if($access != 0 && $access == ACL_ALLOW_ALL || ($is_owner && $access == ACL_
↪ALLOW_OWNER))return true;
       //if this exists, then this function is not static, so check the aclaccess␣
↪parameter
        if(isset($this) && isset($this->aclaccess)){
            if($this->aclaccess == ACL_ALLOW_ALL || ($is_owner && $this->aclaccess ==␣
↪ACL_ALLOW_OWNER))
            return true;
        }
        return false;
    }
```

17.2.34 $this Belongs To Classes Or Traits

OpenEMR

$this Belongs To Classes Or Traits, in ccr/display.php:24.

$this is used to call the document_upload_download_log() method, although this piece of code is not part of a class, nor is included in a class.

```
<?php
require_once(dirname(__FILE__) . "/../interface/globals.php");

$type = $_GET['type'];
$document_id = $_GET['doc_id'];
$d = new Document($document_id);
$url =  $d->get_url();
$storagemethod = $d->get_storagemethod();
$couch_docid = $d->get_couch_docid();
$couch_revid = $d->get_couch_revid();

if ($couch_docid && $couch_revid) {
    $couch = new CouchDB();
    $data = array($GLOBALS['couchdb_dbase'],$couch_docid);
    $resp = $couch->retrieve_doc($data);
    $xml = base64_decode($resp->data);
    if ($content=='' && $GLOBALS['couchdb_log']==1) {
```

(continues on next page)

(continued from previous page)

```
        $log_content = date('Y-m-d H:i:s')." ==> Retrieving document\r\n";
        $log_content = date('Y-m-d H:i:s')." ==> URL: ".$url."\r\n";
        $log_content .= date('Y-m-d H:i:s')." ==> CouchDB Document Id: ".$couch_docid.
↪"\r\n";
        $log_content .= date('Y-m-d H:i:s')." ==> CouchDB Revision Id: ".$couch_revid.
↪"\r\n";
        $log_content .= date('Y-m-d H:i:s')." ==> Failed to fetch document content␣
↪from CouchDB.\r\n";
        //$log_content .= date('Y-m-d H:i:s')." ==> Will try to download file from␣
↪HardDisk if exists.\r\n\r\n";
        $this->document_upload_download_log($d->get_foreign_id(), $log_content);
        die(xlt("File retrieval from CouchDB failed"));
    }
```

17.2.35 Too Many Children

Typo3

Too Many Children, in typo3/sysext/backend/Classes/Form/AbstractNode.php:26.

More than 15 children for this class : 15 is the default configuration.

```
abstract class AbstractNode implements NodeInterface, LoggerAwareInterface {
```

Woocommerce

Too Many Children, in includes/abstracts/abstract-wc-rest-controller.php:30.

This class is extended 22 times, more than the default configuration of 15.

```
class WC_REST_Controller extends WP_REST_Controller {
```

17.2.36 Too Many Injections

NextCloud

Too Many Injections, in lib/private/Share20/Manager.php:130.

Well documented Manager class. Quite a lot of injections though, it must take a long time to prepare it.

```
/**
     * Manager constructor.
     *
     * @param ILogger $logger
     * @param IConfig $config
     * @param ISecureRandom $secureRandom
     * @param IHasher $hasher
     * @param IMountManager $mountManager
     * @param IGroupManager $groupManager
     * @param IL10N $l
     * @param IFactory $l10nFactory
     * @param IProviderFactory $factory
```

(continues on next page)

(continued from previous page)

```
     * @param IUserManager $userManager
     * @param IRootFolder $rootFolder
     * @param EventDispatcher $eventDispatcher
     * @param IMailer $mailer
     * @param IURLGenerator $urlGenerator
     * @param \OC_Defaults $defaults
     */
    public function __construct(
                    ILogger $logger,
                    IConfig $config,
                    ISecureRandom $secureRandom,
                    IHasher $hasher,
                    IMountManager $mountManager,
                    IGroupManager $groupManager,
                    IL10N $l,
                    IFactory $l10nFactory,
                    IProviderFactory $factory,
                    IUserManager $userManager,
                    IRootFolder $rootFolder,
                    EventDispatcher $eventDispatcher,
                    IMailer $mailer,
                    IURLGenerator $urlGenerator,
                    \OC_Defaults $defaults
    ) {
            $this->logger = $logger;
            $this->config = $config;
            $this->secureRandom = $secureRandom;
            $this->hasher = $hasher;
            $this->mountManager = $mountManager;
            $this->groupManager = $groupManager;
            $this->l = $l;
            $this->l10nFactory = $l10nFactory;
            $this->factory = $factory;
            $this->userManager = $userManager;
            $this->rootFolder = $rootFolder;
            $this->eventDispatcher = $eventDispatcher;
            $this->sharingDisabledForUsersCache = new CappedMemoryCache();
            $this->legacyHooks = new LegacyHooks($this->eventDispatcher);
            $this->mailer = $mailer;
            $this->urlGenerator = $urlGenerator;
            $this->defaults = $defaults;
    }
```

Thelia

Too Many Injections, in core/lib/Thelia/Core/Event/Delivery/DeliveryPostageEvent.php:58.

Classic address class, with every details. May be even shorter than expected.

```
//class DeliveryPostageEvent extends ActionEvent
    public function __construct(
        DeliveryModuleInterface $module,
        Cart $cart,
        Address $address = null,
        Country $country = null,
```

(continues on next page)

(continued from previous page)

```
        State $state = null
    ) {
        $this->module = $module;
        $this->cart = $cart;
        $this->address = $address;
        $this->country = $country;
        $this->state = $state;
    }
```

17.2.37 Wrong Access Style to Property

HuMo-Gen

Wrong Access Style to Property, in wp-admin/includes/misc.php:74.

lame_binary_path is a static property, but it is accessed as a normal property in the exception call, while it is checked with a valid syntax.

```
protected function wavToMp3($data)
    {
        if (!file_exists(self::$lame_binary_path) || !is_executable(self::$lame_
↪binary_path)) {
            throw new Exception('Lame binary  . $this->lame_binary_path .  does not␣
↪exist or is not executable');
        }
```

17.2.38 Undefined Properties

WordPress

Undefined Properties, in wp-admin/includes/misc.php:74.

Properties are not defined, but they are thoroughly initialized when the XML document is parsed. All those definition should be in a property definition, for clear documentation.

```
$this->DeliveryLine1 = '';
        $this->DeliveryLine2 = '';
        $this->City = '';
        $this->State = '';
        $this->ZipAddon = '';
```

MediaWiki

Undefined Properties, in includes/logging/LogFormatter.php:561.

parsedParametersDeleteLog is an undefined property. Defining the property with a null default value is important here, to keep the code running.

```
protected function getMessageParameters() {
            if ( isset( $this->parsedParametersDeleteLog ) ) {
                    return $this->parsedParametersDeleteLog;
            }
```

17.2.39 Undefined static:: Or self::

xataface

Undefined static:: Or self::, in actions/forgot_password.php:194.

This is probably a typo, since the property called public static $EX_NO_USERS_WITH_EMAIL = 501; is defined in that class.

```
if ( !$user ) throw new Exception(df_translate('actions.forgot_password.null_user',
↪"Cannot send email for null user"), self::$EX_NO_USERS_FOUND_WITH_EMAIL);
```

SugarCrm

Undefined static:: Or self::, in code/SugarCE-Full-6.5.26/include/SugarDateTime.php:574.

self::$sugar_strptime_long_mon refers to the current class, which extends DateTime. No static property was defined at either of them, with the name '$sugar_strptime_long_mon'. This has been a Fatal error at execution time since PHP 5.3, at least.

```
if ( isset($regexp['positions']['F']) && !empty($dateparts[$regexp['positions']['F
↪']])) {
                // FIXME: locale?
    $mon = $dateparts[$regexp['positions']['F']];
    if(isset(self::$sugar_strptime_long_mon[$mon])) {
        $data["tm_mon"] = self::$sugar_strptime_long_mon[$mon];
    } else {
        return false;
    }
}
```

17.2.40 Unitialized Properties

SPIP

Unitialized Properties, in ecrire/public/interfaces.php:584.

The class Critere (Criteria) has no method at all. When using a class as an array, to capture values, one of the advantage of the class is in the default values for the properties. In particular, the last property here, called $not, should be initialized with a false.

```
/**
 * Description d'un critère de boucle
 *
 * Sous-noeud de Boucle
 *
 * @package SPIP\Core\Compilateur\AST
 **/
class Critere {
    /**
     * Type de noeud
     *
     * @var string
     */
```

(continues on next page)

(continued from previous page)

```
    public $type = 'critere';

    /**
     * Opérateur (>, <, >=, IN, ...)
     *
     * @var null|string
     */
    public $op;

    /**
     * Présence d'une négation (truc !op valeur)
     *
     * @var null|string
     */
    public $not;
```

17.2.41 Unresolved Instanceof

WordPress

Unresolved Instanceof, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
private function resolveTag($match)
    {
        $tagReflector = $this->createLinkOrSeeTagFromRegexMatch($match);
        if (!$tagReflector instanceof Tag\SeeTag && !$tagReflector instanceof␣
↪Tag\LinkTag) {
            return $match;
        }
```

17.2.42 Unused Private Properties

OpenEMR

Unused Private Properties, in entities/User.php:46.

This class has a long list of private properties. It also has an equally long (minus one) list of accessors, and a __toString() method which exposes all of them. $oNotes is the only one never mentionned anywhere.

```
class User
{
    /**
     * @Column(name=id, type=integer)
     * @GeneratedValue(strategy=AUTO)
     */
    private $id;

    /**
     * @OneToMany(targetEntity=ONote, mappedBy=user)
     */
    private $oNotes;
```

phpadsnew

Unused Private Properties, in lib/OA/Admin/UI/component/Form.php:23.

$dispatcher is never used anywhere.

```
class OA_Admin_UI_Component_Form
    extends HTML_QuickForm
{
    private $dispatcher;
```

17.2.43 Use ::Class Operator

Typo3

Use ::Class Operator, in typo3/sysext/install/Configuration/ExtensionScanner/Php/ConstructorArgumentMatcher.php:4.

`TYPO3\CMS\Core\Package\PackageManager` could be `TYPO3\CMS\Core\Package\PackageManager::class`.

```
return [
    'TYPO3\CMS\Core\Package\PackageManager' => [
        'required' => [
            'numberOfMandatoryArguments' => 1,
            'maximumNumberOfArguments' => 1,
```

17.2.44 Use Instanceof

TeamPass

Use Instanceof, in includes/libraries/Database/Meekrodb/db.class.php:506.

In this code, `is_object()` and `instanceof` have the same basic : they both check that $ts is an object. In fact, `instanceof` is more precise, and give more information about the variable.

```
protected function parseTS($ts) {
    if (is_string($ts)) return date('Y-m-d H:i:s', strtotime($ts));
    else if (is_object($ts) && ($ts instanceof DateTime)) return $ts->format('Y-m-d␣
↪H:i:s');
  }
```

Zencart

Use Instanceof, in includes/modules/payment/firstdata_hco.php:104.

In this code, `is_object()` is used to check the status of the order. Possibly, $order is false or null in case of incompatible status. Yet, when $object is an object, and in particular being a global that may be assigned anywhere else in the code, it seems that the method 'update_status' is magically always available. Here, using instance of to make sure that $order is an 'paypal' class, or a 'storepickup' or any of the payment class.

```
function __construct() {
    global $order;

    // more lines, no mention of $order
```

(continues on next page)

(continued from previous page)

```
    if (is_object($order)) $this->update_status();

    // more code
}
```

17.2.45 Weak Typing

TeamPass

Weak Typing, in includes/libraries/Tree/NestedTree/NestedTree.php:100.

The is_null() test detects a special situation, that requires usage of default values. The 'else' handles every other situations, including when the $node is an object, or anything else. $this->getNode() will gain from having typehints : it may be NULL, or the results of mysqli_fetch_object() : a stdClass object. The expected properties of nleft and nright are not certain to be available.

```
public function getDescendants($id = 0, $includeSelf = false, $childrenOnly = false,
↪$unique_id_list = false)
    {
        global $link;
        $idField = $this->fields['id'];

        $node = $this->getNode($id);
        if (is_null($node)) {
            $nleft = 0;
            $nright = 0;
            $parent_id = 0;
            $personal_folder = 0;
        } else {
            $nleft = $node->nleft;
            $nright = $node->nright;
            $parent_id = $node->$idField;
            $personal_folder = $node->personal_folder;
        }
```

17.2.46 Wrong Class Name Case

WordPress

Wrong Class Name Case, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

17.2.47 Illegal Name For Method

PrestaShop

Illegal Name For Method, in admin-dev/ajaxfilemanager/inc/class.pagination.php:200.

__getBaseUrl and __setBaseUrl shouldn't be named like that.

```
/**
     * get base url for pagination links aftr excluded those key
     * identified on excluded query strings
     *
     */
    function __getBaseUrl()
    {

            if(empty($this->baseUrl))
            {

                    $this->__setBaseUrl();
            }
            return $this->baseUrl;
    }
```

Magento

Illegal Name For Method, in app/code/core/Mage/Core/Block/Abstract.php:1139.

public method, called '__'. Example : $this->__();

```
public function __()
    {
        $args = func_get_args();
        $expr = new Mage_Core_Model_Translate_Expr(array_shift($args), $this->
↪getModuleName());
        array_unshift($args, $expr);
        return $this->_getApp()->getTranslator()->translate($args);
    }
```

17.2.48 Bad Constants Names

PrestaShop

Bad Constants Names, in src/PrestaShopBundle/Install/Upgrade.php:214.

INSTALL_PATH is a valid name for a constant. __PS_BASE_URI__ is not a valid name.

```
require_once(INSTALL_PATH . 'install_version.php');
            // needed for upgrade before 1.5
            if (!defined('__PS_BASE_URI__')) {
                define('__PS_BASE_URI__', str_replace('//', '/', '/'.trim(preg_
↪replace('#/(install(-dev)?/upgrade)$#', '/', str_replace('\', '/', dirname($_SERVER[
↪'REQUEST_URI'])))), '/').'/'));
            }
```

Zencart

Bad Constants Names, in zc_install/ajaxTestDBConnection.php:10.

A case where PHP needs help : if the PHP version is older than 5.3, then it is valid to compensate. Though, this __DIR__ has a fixed value, wherever it is used, while the official __DIR__ change from dir to dir.

```
if (!defined('__DIR__')) define('__DIR__', dirname(__FILE__));
```

17.2.49 Use const

phpMyAdmin

Use const, in error_report.php:17.

This may be turned into a *const* call, with a static expression.

```
define('ROOT_PATH', __DIR__ . DIRECTORY_SEPARATOR)
```

Piwigo

Use const, in include/functions_plugins.inc.php:32.

Const works efficiently with literal

```
define('EVENT_HANDLER_PRIORITY_NEUTRAL', 50)
```

17.2.50 Invalid Constant Name

OpenEMR

Invalid Constant Name, in library/classes/InsuranceCompany.class.php:20.

Either a copy/paste, or a generated definition file : the file contains 25 constants definition. The constant is not found in the rest of the code.

```
define("INS_TYPE_OTHER_NON-FEDERAL_PROGRAMS", 10);
```

17.2.51 Multiple Constant Definition

Dolibarr

Multiple Constant Definition, in htdocs/main.inc.php:914.

All is documented here : 'Constants used to defined number of lines in textarea'. Constants are not changing during an execution, and this allows the script to set values early in the process, and have them used later, in the templates. Yet, building constants dynamically may lead to confusion, when developpers are not aware of the change.

```
// Constants used to defined number of lines in textarea
if (empty($conf->browser->firefox))
{
    define('ROWS_1',1);
    define('ROWS_2',2);
    define('ROWS_3',3);
    define('ROWS_4',4);
    define('ROWS_5',5);
    define('ROWS_6',6);
    define('ROWS_7',7);
```

(continues on next page)

(continued from previous page)

```
    define('ROWS_8',8);
    define('ROWS_9',9);
}
else
{
    define('ROWS_1',0);
    define('ROWS_2',1);
    define('ROWS_3',2);
    define('ROWS_4',3);
    define('ROWS_5',4);
    define('ROWS_6',5);
    define('ROWS_7',6);
    define('ROWS_8',7);
    define('ROWS_9',8);
}
```

OpenConf

Multiple Constant Definition, in modules/request.php:71.

The constant is build according to the situation, in the part of the script (file request.php). This hides the actual origin of the value, but keeps the rest of the code simple. Just keep in mind that this constant may have different values.

```
if (isset($_GET['ocparams']) && !empty($_GET['ocparams'])) {
            $params = '';
            if (preg_match_all("/(\w+)--(\w+)_-/", $_GET['ocparams'], $matches)) {
                    foreach ($matches[1] as $idx => $m) {
                            if (($m != 'module') && ($m != 'action') && preg_match("/^
↪[\w-]+$/", $m)) {
                                    $params .= '&' . $m . '=' . urlencode($matches[2][
↪$idx]);
                                    $_GET[$m] = $matches[2][$idx];
                            }
                    }
            }
            unset($_GET['ocparams']);
            define('OCC_SELF', $_SERVER['PHP_SELF'] . '?module=' . $_REQUEST['module
↪'] . '&action=' . $_GET['action'] . $params);
    } elseif (isset($_SERVER['REQUEST_URI']) && strstr($_SERVER['REQUEST_URI'], '?'))
↪{
            define('OCC_SELF', htmlspecialchars($_SERVER['REQUEST_URI']));
    } elseif (isset($_SERVER['QUERY_STRING']) && strstr($_SERVER['QUERY_STRING'], '&
↪')) {
            define('OCC_SELF', $_SERVER['PHP_SELF'] . '?' . htmlspecialchars($_SERVER[
↪'QUERY_STRING']));
    } else {
            err('This server does not support REQUEST_URI or QUERY_STRING','Error');
    }
```

17.2.52 Exception Order

Woocommerce

Exception Order, in includes/api/v1/class-wc-rest-products-controller.php:787.

This try/catch expression is able to catch both WC_Data_Exception and WC_REST_Exception.

In another file, /includes/api/class-wc-rest-exception.php, we find that WC_REST_Exception extends WC_Data_Exception (class WC_REST_Exception extends WC_Data_Exception {}). So WC_Data_Exception is more general, and a WC_REST_Exception exception is caught with WC_Data_Exception Exception. The second catch should be put in first.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
try {
                    $product_id = $this->save_product( $request );
                    $post       = get_post( $product_id );
                    $this->update_additional_fields_for_object( $post, $request );
                    $this->update_post_meta_fields( $post, $request );

                    /**
                     * Fires after a single item is created or updated via the REST␣
↪API.
                     *
                     * @param WP_Post         $post      Post data.
                     * @param WP_REST_Request $request   Request object.
                     * @param boolean         $creating  True when creating item,␣
↪false when updating.
                     */
                    do_action( 'woocommerce_rest_insert_product', $post, $request,␣
↪false );
                    $request->set_param( 'context', 'edit' );
                    $response = $this->prepare_item_for_response( $post, $request );

                    return rest_ensure_response( $response );
            } catch ( WC_Data_Exception $e ) {
                    return new WP_Error( $e->getErrorCode(), $e->getMessage(), $e->
↪getErrorData() );
            } catch ( WC_REST_Exception $e ) {
                    return new WP_Error( $e->getErrorCode(), $e->getMessage(), array(
↪'status' => $e->getCode() ) );
            }
```

17.2.53 Could Use Try

Mautic

Could Use Try, in app/bundles/StageBundle/Controller/StageController.php:78.

$limit is read as a session variable or a default value. There are no check here that $limit is not null, before using it in a division. It is easy to imagine this is done elsewhere, yet a try/catch could help intercept unwanted situations.

```
//set limits
        $limit = $this->get('session')->get(
            'mautic.stage.limit',
            $this->coreParametersHelper->getParameter('default_pagelimit')
        );
/... Code where $limit is read but not modified /
        $count = count($stages);
        if ($count && $count < ($start + 1)) {
            $lastPage = ($count === 1) ? 1 : (ceil($count / $limit)) ?: 1;
```

17.2.54 Rethrown Exceptions

PrestaShop

Rethrown Exceptions, in classes/webservice/WebserviceOutputBuilder.php:731.

The setSpecificField method catches a WebserviceException, representing an issue with the call to the webservice. However, that piece of information is lost, and the exception is rethrown immediately, without any action.

```
public function setSpecificField($object, $method, $field_name, $entity_name)
    {
            try {
                    $this->validateObjectAndMethod($object, $method);
            } catch (WebserviceException $e) {
                    throw $e;
            }

            $this->specificFields[$field_name] = array('entity'=>$entity_name, 'object
↪' => $object, 'method' => $method, 'type' => gettype($object));
            return $this;
    }
```

17.2.55 Throw Functioncall

SugarCrm

Throw Functioncall, in include/externalAPI/cmis_repository_wrapper.php:918.

SugarCRM uses exceptions to fill work in progress. Here, we recognize a forgotten 'new' that makes throw call a function named 'Exception'. This fails with a Fatal Error, and doesn't issue the right messsage. The same error had propgated in the code by copy and paste : it is available 17 times in that same file.

```
function getContentChanges()
    {
        throw Exception("Not Implemented");
    }
```

Zurmo

Throw Functioncall, in app/protected/modules/gamification/rules/collections/GameCollectionRules.php:66.

Other part of the code actually instantiate the exception before throwing it.

```
abstract class GameCollectionRules
    {
        /**
         * @return string
         * @throws NotImplementedException - Implement in children classes
         */
        public static function getType()
        {
            throw NotImplementedException();
        }
```

17.2.56 Useless Catch

Zurmo

Useless Catch, in app/protected/modules/workflows/forms/attributes/ExplicitReadWriteModelPermissionsWorkflowActionAttributeFor

Catch the exception, then return. At least, the comment is honest.

```
try
                {
                    $group = Group::getById((int)$this->type);
                    $explicitReadWriteModelPermissions->addReadWritePermitable(
↪$group);
                }
                catch (NotFoundException $e)
                {
                    //todo: handle exception better
                    return;
                }
```

PrestaShop

Useless Catch, in src/Core/Addon/Module/ModuleManagerBuilder.php:170.

Here, the catch clause will intercept a IO problem while writing element on the disk, and will return false. Since this is a constructor, the returned value will be ignored and the object will be left in a wrong state, since it was not totally inited.

```
private function __construct()
    {
    // More code......
            try {
                $filesystem = new Filesystem();
                $filesystem->dumpFile($phpConfigFile, '<?php return ' . var_export(
↪$config, true) . ';' . \n);
            } catch (IOException $e) {
                return false;
            }
        }
```

17.2.57 Add Default Value

Zurmo

Add Default Value, in wp-admin/includes/misc.php:74.

Default values may be a literal (1, 'abc', ...), or a constant : global or class. Here, MissionsListConfigurationForm::LIST_TYPE_AVAILABLE may be used directly in the signature of the method

```
public function getMetadataFilteredByOption($option)
        {
            if ($option == null)
            {
                $option = MissionsListConfigurationForm::LIST_TYPE_AVAILABLE;
            }
```

Typo3

Add Default Value, in typo3/sysext/indexed_search/Classes/FileContentParser.php:821.

$extension could get a default value to handle default situations : for example, a file is htm format by default, unless better known. Also, the if/then structure could get a 'else' clause, to handle unknown situations : those are situations where the extension is provided but not known, in particular when the icon is missing in the storage folder.

```
public function getIcon($extension)
    {
        if ($extension === 'htm') {
            $extension = 'html';
        } elseif ($extension === 'jpeg') {
            $extension = 'jpg';
        }
        return 'EXT:indexed_search/Resources/Public/Icons/FileTypes/' . $extension .
↪'.gif';
    }
```

17.2.58 Aliases Usage

Cleverstyle

Aliases Usage, in modules/HybridAuth/Hybrid/thirdparty/Vimeo/Vimeo.php:422.

is_writeable() should be written is_writable(). No extra 'e'.

```
is_writeable($chunk_temp_dir)
```

phpMyAdmin

Aliases Usage, in libraries/classes/Server/Privileges.php:5064.

join() should be written implode()

```
join('`, `', $tmp_privs2['Update'])
```

17.2.59 Use Named Boolean In Argument Definition

phpMyAdmin

Use Named Boolean In Argument Definition, in /libraries/classes/Util.php:1929.

$request is an option to *checkParameters*, although it is not visibile with is its actual role.

```
public static function checkParameters($params, $request = false) {
    /**/
}
```

Cleverstyle

Use Named Boolean In Argument Definition, in /core/classes/Response.php:129.

$httponly is an option to *cookie*, and true/false makes it readable. There may be other situations, like fallback, or forcedd usage, so the boolean may be misleading. Note also the *$expire = 0*, which may be a date, or a special value. We need to read the documentation to understand this.

```
public function cookie($name, $value, $expire = 0, $httponly = false) { /**/ }        {
    /**/
}
```

17.2.60 Callback Function Needs Return

Contao

Callback Function Needs Return, in core-bundle/src/Resources/contao/modules/ModuleQuicklink.php:91.

The empty closure returns *null*. The array_flip() array has now all its values set to null, and reset, as intended. A better alternative is to use the array_fill_keys() function, which set a default value to every element of an array, once provided with the expected keys.

```
$arrPages = array_map(function () {}, array_flip($tmp));
```

Phpdocumentor

Callback Function Needs Return, in src/phpDocumentor/Plugin/ServiceProvider.php:24.

The array_walk() function is called on the plugin's list. Each element is registered with the application, but is not used directly : this is for later. The error mechanism is to throw an exception : this is the only expected feedback. As such, no return is expected. May be a 'foreach' loop would be more appropriate here, but this is syntactic sugar.

```
array_walk(
            $plugins,
            function ($plugin) use ($app) {
                /** @var Plugin $plugin */
                $provider = (strpos($plugin->getClassName(), '\') === false)
                    ? sprintf('phpDocumentor\Plugin\%s\ServiceProvider', $plugin->
↪getClassName())
                    : $plugin->getClassName();
                if (!class_exists($provider)) {
                    throw new \RuntimeException('Loading Service Provider for ' .
↪$provider . ' failed.');
                }

                try {
                    $app->register(new $provider($plugin));
                } catch (\InvalidArgumentException $e) {
                    throw new \RuntimeException($e->getMessage());
                }
            }
        );
```

17.2.61 Closure Could Be A Callback

Tine20

Closure Could Be A Callback, in tine20/Tinebase/Convert/Json.php:318.

is_scalar() is sufficient here.

```
$value = array_filter($value, function ($val) { return is_scalar($val); });
```

NextCloud

Closure Could Be A Callback, in apps/files_sharing/lib/ShareBackend/Folder.php:114.

$qb is the object for the methodcall, passed via use. The closure may have been replaced with array($qb, 'createNamedParameter').

```
$parents = array_map(function($parent) use ($qb) {
                            return $qb->createNamedParameter($parent);
                        }, $parents);
```

17.2.62 Could Be Typehinted Callable

Magento

Could Be Typehinted Callable, in wp-admin/includes/misc.php:74.

$objMethod argument is used to call a function, a method or a localmethod. The typehint would save the middle condition, and make a better job than 'is_array' to check if $objMethod is callable. Yet, the final 'else' means that $objMethod is also the name of a method, and PHP won't validate this, unless there is a function with the same name. Here, callable is not an option.

```
public function each($objMethod, $args = [])
    {
        if ($objMethod instanceof \Closure) {
            foreach ($this->getItems() as $item) {
                $objMethod($item, ...$args);
            }
        } elseif (is_array($objMethod)) {
            foreach ($this->getItems() as $item) {
                call_user_func($objMethod, $item, ...$args);
            }
        } else {
            foreach ($this->getItems() as $item) {
                $item->$objMethod(...$args);
            }
        }
    }
```

PrestaShop

Could Be Typehinted Callable, in controllers/admin/AdminImportController.php:1147.

$funcname is tested with is_callable() before being used as a method. Typehint callable would reduce the size of the code.

```
public static function arrayWalk(&$array, $funcname, &$user_data = false)
    {
            if (!is_callable($funcname)) return false;

            foreach ($array as $k => $row)
                    if (!call_user_func_array($funcname, array($row, $k, $user_data)))
                            return false;
            return true;
    }
```

17.2.63 Could Be Static Closure

Piwigo

Could Be Static Closure, in include/ws_core.inc.php:620.

The closure function($m) makes no usage of the current object : using static prevents $this to be forwarded with the closure.

```
/**
   * WS reflection method implementation: lists all available methods
   */
  static function ws_getMethodList($params, &$service)
  {
    $methods = array_filter($service->_methods,
      function($m) { return empty($m["options"]["hidden"]) || !$m["options"]["hidden
↪"];} );
    return array('methods' => new PwgNamedArray( array_keys($methods),'method' ) );
  }
```

17.2.64 Deep Definitions

Dolphin

Deep Definitions, in wp-admin/includes/misc.php:74.

The ConstructHiddenValues function builds the ConstructHiddenSubValues function. Thus, ConstructHiddenValues can only be called once.

```
function ConstructHiddenValues($Values)
{
    /**
     *    Recursive function, processes multidimensional arrays
     *
     * @param string $Name  Full name of array, including all subarrays' names
     *
     * @param array  $Value Array of values, can be multidimensional
     *
     * @return string    Properly consctructed <input type="hidden"...> tags
     */
    function ConstructHiddenSubValues($Name, $Value)
```

(continues on next page)

(continued from previous page)

```
    {
        if (is_array($Value)) {
            $Result = "";
            foreach ($Value as $KeyName => $SubValue) {
                $Result .= ConstructHiddenSubValues("{$Name}[{$KeyName}]", $SubValue);
            }
        } else // Exit recurse
        {
            $Result = "<input type=\"hidden\" name=\"" . htmlspecialchars($Name) . "\
↪" value=\"" . htmlspecialchars($Value) . "\" />\n";
        }

        return $Result;
    }

    /* End of ConstructHiddenSubValues function */

    $Result = '';
    if (is_array($Values)) {
        foreach ($Values as $KeyName => $Value) {
            $Result .= ConstructHiddenSubValues($KeyName, $Value);
        }
    }

    return $Result;
}
```

17.2.65 Empty Function

Contao

Empty Function, in core-bundle/src/Resources/contao/modules/ModuleQuicklink.php:91.

The closure used with array_map() is empty : this means that the keys are all set to the returned value of the empty closure, which is null. The actual effect is to reset the values to NULL. A better solution, without using the empty closure, is to rely on array_fill_keys() to create an array with default values.

```
if (!empty($tmp) && \is_array($tmp))
                    {
                            $arrPages = array_map(function () {}, array_flip($tmp));
                    }
```

17.2.66 Mismatched Default Arguments

SPIP

Mismatched Default Arguments, in ecrire/inc/lien.php:160.

generer_url_entite() takes $connect in, with a default value of empty string. Later, generer_url_entite() receives that value, but uses null as a default value. This forces the ternary test on $connect, to turn it into a null before shipping it to the next function, and having it processed accordingly.

```
// http://code.spip.net/@traiter_lien_implicite
function traiter_lien_implicite($ref, $texte = '', $pour = 'url', $connect = '') {

    // some code was edited here

    if (is_array($url)) {
            @list($type, $id) = $url;
            $url = generer_url_entite($id, $type, $args, $ancre, $connect ? $connect␣
↪: null);
    }
```

17.2.67 Mismatched Typehint

WordPress

Mismatched Typehint, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

17.2.68 Never Used Parameter

Piwigo

Never Used Parameter, in include/functions_html.inc.php:329.

$alternate_url is never explicitely passed to bad_request() : this doesn't show in this extract. It could be dropped from this code.

```
function bad_request($msg, $alternate_url=null)
{
  set_status_header(400);
  if ($alternate_url==null)
    $alternate_url = make_index_url();
  redirect_html( $alternate_url,
    '<div style="text-align:left; margin-left:5em;margin-bottom:5em;">
<h1 style="text-align:left; font-size:36px;">'.l10n('Bad request').'</h1><br>'
.$msg.'</div>',
    5 );
}
```

17.2.69 No Boolean As Default

OpenConf

No Boolean As Default, in openconf/include.php:1264.

Why do we need a *chair* when printing a cell's file ?

```
function oc_printFileCells(&$sub, $chair = false) { /**/ }
```

17.2.70 No Class As Typehint

Vanilla

No Class As Typehint, in library/Vanilla/Formatting/Formats/RichFormat.php:51.

All three typehints are based on classes. When Parser or Renderer are changed, for testing, versioning or moduling reasons, they must subclass the original class.

```
public function __construct(Quill\Parser $parser, Quill\Renderer $renderer,␣
↪Quill\Filterer $filterer) {
        $this->parser = $parser;
        $this->renderer = $renderer;
        $this->filterer = $filterer;
    }
```

phpMyAdmin

No Class As Typehint, in libraries/classes/CreateAddField.php:29.

Although the class is named 'DatabaseInterface', it is a class.

```
public function __construct(DatabaseInterface $dbi)
    {
        $this->dbi = $dbi;
    }
```

17.2.71 No Return Used

SPIP

No Return Used, in ecrire/inc/utils.php:1067.

job_queue_remove() is called as an administration order, and the result is not checked. It is considered as a fire-and-forget command.

```
function job_queue_remove($id_job) {
    include_spip('inc/queue');

    return queue_remove_job($id_job);
}
```

LiveZilla

No Return Used, in livezilla/_lib/trdp/Zend/Loader.php:114.

The loadFile method tries to load a file, aka as include. If the inclusion fails, a PHP error is emitted (an exception would do the same), and there is not error management. Hence, the 'return true;', which is not tested later. It may be dropped.

```
public static function loadFile($filename, $dirs = null, $once = false)
    {
// A lot of code to check and include files
```

(continues on next page)

(continued from previous page)

```
        return true;
    }
```

17.2.72 One Letter Functions

ThinkPHP

One Letter Functions, in ThinkPHP/Mode/Api/functions.php:859.

There are also the functions C, E, G... The applications is written in a foreign language, which may be a base for non-significant function names.

```
function F($name, $value = '', $path = DATA_PATH)
```

Cleverstyle

One Letter Functions, in core/drivers/DB/PostgreSQL.php:71.

There is also function f(). Those are actually overwritten methods. From the documentation, q() is for query, and f() is for fetch. Those are short names for frequently used functions.

```
public function q ($query, ...$params) {
```

17.2.73 Only Variable Passed By Reference

Dolphin

Only Variable Passed By Reference, in administration/charts.json.php:89.

This is not possible, as array_slice() returns a new array, and not a reference. Minimally, the intermediate result must be saved in a variable, then popped. Actually, this code extracts the element at key 1 in the $aData array, although this also works with hash (non-numeric keys).

```
array_pop(array_slice($aData, 0, 1))
```

PhpIPAM

Only Variable Passed By Reference, in functions/classes/class.Thread.php:243.

This is sneaky bug : the assignation $status = 0 returns a value, and not a variable. This leads PHP to mistake the initialized 0 with the variable $status and fails. It is not possible to initialize variable AND use them as argument.

```
pcntl_waitpid($this->pid, $status = 0)
```

17.2.74 Relay Function

TeamPass

Relay Function, in includes/libraries/Goodby/CSV/Import/Standard/Interpreter.php:88.

This example puts actually a name on the events : this method 'delegate' and it does it in the smallest amount of possible work, being given all the arguments.

```
/**
     * delegate to observer
     *
     * @param $observer
     * @param $line
     */
    private function delegate($observer, $line)
    {
        call_user_func($observer, $line);
    }
```

SPIP

Relay Function, in ecrire/inc/json.php:73.

var2js() acts as an alternative for json_encode(). Yet, it used to be directly called by the framework's code and difficult to change. With the advent of json_encode, the native function has been used, and even, a compatibility tool was set up. Thus, the relay function.

```
if (!function_exists('json_encode')) {
    function json_encode($v) {
            return var2js($v);
    }
}
```

17.2.75 Argument Should Be Typehinted

Dolphin

Argument Should Be Typehinted, in Dolphin-v.7.3.5/plugins/intervention-image/Intervention/Image/Gd/Commands/WidenCommand.ph

This closures make immediate use of the $constraint argument, and calls its method aspectRatio. No check is made on this argument, and it may easily be mistaken with another class, or a null. Adding a typehint here will ensure a more verbose development error and help detect misuse of the closure.

```
$this->arguments[2] = function ($constraint) use ($additionalConstraints) {
            $constraint->aspectRatio();
            if(is_callable($additionalConstraints))
                $additionalConstraints($constraint);
        };
```

Mautic

Argument Should Be Typehinted, in app/bundles/PluginBundle/Helper/IntegrationHelper.php:374.

This piece of code inside a 275 lines method. Besides, there are 11 classes that offer a 'getPriority' method, although $returnServices could help to semantically reduce the number of possible classes. Here, typehints on $a and $b help using the wrong kind of object.

```
if (empty($alphabetical)) {
            // Sort by priority
            uasort($returnServices, function ($a, $b) {
                $aP = (int) $a->getPriority();
                $bP = (int) $b->getPriority();

                if ($aP === $bP) {
                    return 0;
                }

                return ($aP < $bP) ? -1 : 1;
            });
```

17.2.76 Should Use Constants

Tine20

Should Use Constants, in tine20/Sales/Controller/Invoice.php:560.

True should be replaced by COUNT_RECURSIVE. The default one is COUNT_NORMAL.

```
count($billables, true)
```

17.2.77 Too Many Local Variables

HuMo-Gen

Too Many Local Variables, in relations.php:813.

15 local variables pieces of code are hard to find in a compact form. This function shows one classic trait of such issue : a large ifthen is at the core of the function, and each time, it collects some values and build a larger string. This should probably be split between different methods in a class.

```
function calculate_nephews($generX) { // handed generations x is removed from common␣
↪ancestor
global $db_functions, $reltext, $sexe, $sexe2, $language, $spantext, $selected_
↪language, $foundX_nr, $rel_arrayX, $rel_arrayspouseX, $spouse;
global $reltext_nor, $reltext_nor2; // for Norwegian and Danish

    if($selected_language=="es"){
            if($sexe=="m") { $neph=__('nephew'); $span_postfix="o "; $grson='nieto'; }
            else { $neph=__('niece'); $span_postfix="a "; $grson='nieta'; }
            //$gendiff = abs($generX - $generY); // FOUT
            $gendiff = abs($generX - $generY) - 1;
            $gennr=$gendiff-1;
            $degree=$grson." ".$gennr.$span_postfix;
            if($gendiff ==1) { $reltext=$neph.__(' of ');}
            elseif($gendiff > 1 AND $gendiff < 27) {
                    spanish_degrees($gendiff,$grson);
                    $reltext=$neph." ".$spantext.__(' of ');
```

(continues on next page)

(continued from previous page)

```
            }
            else { $reltext=$neph." ".$degree; }
    } elseif ($selected_language==he){
            if($sexe=='m') { $nephniece = __('nephew'); }
///............
```

17.2.78 Too Many Parameters

WordPress

Too Many Parameters, in wp-admin/includes/misc.php:74.

11 parameters is a lot for a function. Note that it is more than the default configuration, and reported there. This may be configured.

```
/**
 * [identifyUserRights description]
 * @param  string $groupesVisiblesUser  [description]
 * @param  string $groupesInterditsUser [description]
 * @param  string $isAdmin              [description]
 * @param  string $idFonctions          [description]
 * @return string                       [description]
 */
function identifyUserRights(
    $groupesVisiblesUser,
    $groupesInterditsUser,
    $isAdmin,
    $idFonctions,
    $server,
    $user,
    $pass,
    $database,
    $port,
    $encoding,
    $SETTINGS
) {
```

ChurchCRM

Too Many Parameters, in src/Reports/ReminderReport.php:192.

10 parameters is a lot for a function. Here, we may also identify a family (ID, Name), and a full address (Address1, Address2, State, Zip, Country), which may be turned into an object.

```
public function StartNewPage($fam_ID, $fam_Name, $fam_Address1, $fam_Address2, $fam_
↪City, $fam_State, $fam_Zip, $fam_Country, $fundOnlyString, $iFYID)
{
```

17.2.79 Unused Arguments

ThinkPHP

Unused Arguments, in ThinkPHP/Library/Behavior/AgentCheckBehavior.class.php:18.

$params are requested, but never used. The method is not overloading another one, as the class doesn't extends anything. $params is unused.

```
class AgentCheckBehavior
{
    public function run(&$params)
    {
        //
        $limitProxyVisit = C('LIMIT_PROXY_VISIT', null, true);
        if ($limitProxyVisit && ($_SERVER['HTTP_X_FORWARDED_FOR'] || $_SERVER['HTTP_
↪VIA'] || $_SERVER['HTTP_PROXY_CONNECTION'] || $_SERVER['HTTP_USER_AGENT_VIA'])) {
            //
            exit('Access Denied');
        }
    }
}
```

phpMyAdmin

Unused Arguments, in libraries/classes/Display/Results.php:1985.

Although $column_index is documented, it is not found in the rest of the (long) body of the function. It might have been refactored into $sorted_column_index.

```
/**
     * Prepare parameters and html for sorted table header fields
     *
     * @param array    $sort_expression             sort expression
     * @param array    $sort_expression_nodirection sort expression without direction
     * @param string   $sort_tbl                    The name of the table to which
     *                                             the current column belongs to
     * @param string   $name_to_use_in_sort         The current column under
     *                                             consideration
     * @param array    $sort_direction              sort direction
     * @param stdClass $fields_meta                 set of field properties
     * @param integer  $column_index                The index number to current column
     *
     * @return  array   3 element array - $single_sort_order, $sort_order, $order_img
     *
     * @access  private
     *
     * @see     _getOrderLinkAndSortedHeaderHtml()
     */
    private function _getSingleAndMultiSortUrls(
        array $sort_expression,
        array $sort_expression_nodirection,
        $sort_tbl,
        $name_to_use_in_sort,
        array $sort_direction,
        $fields_meta,
        $column_index
    ) {
    /**/
        // find the sorted column index in row result
        // (this might be a multi-table query)
        $sorted_column_index = false;
```

(continues on next page)

(continued from previous page)

```
    /**/
    }
```

17.2.80 Unused Functions

Woocommerce

Unused Functions, in includes/wc-core-functions.php:2124.

wc_is_external_resource() is unused. This is not obvious immediately, since there is a call from wc_get_relative_url(). Yet since wc_get_relative_url() itself is never used, then it is a dead function. As such, since wc_is_external_resource() is only called by this first function, it also dies, even though it is called in the code.

```
/**
 * Make a URL relative, if possible.
 *
 * @since 3.2.0
 * @param string $url URL to make relative.
 * @return string
 */
function wc_get_relative_url( $url ) {
    return wc_is_external_resource( $url ) ? $url : str_replace( array( 'http://',
↪'https://' ), '//', $url );
}

/**
 * See if a resource is remote.
 *
 * @since 3.2.0
 * @param string $url URL to check.
 * @return bool
 */
function wc_is_external_resource( $url ) {
    $wp_base = str_replace( array( 'http://', 'https://' ), '//', get_home_url( null,
↪'/', 'http' ) );

    return strstr( $url, '://' ) && ! strstr( $url, $wp_base );
}
```

Piwigo

Unused Functions, in admin/include/functions.php:2167.

get_user_access_level_html_options() is unused and can't be find in the code.

```
/**
 * Returns access levels as array used on template with html_options functions.
 *
 * @param int $MinLevelAccess
 * @param int $MaxLevelAccess
 * @return array
 */
function get_user_access_level_html_options($MinLevelAccess = ACCESS_FREE,
↪$MaxLevelAccess = ACCESS_CLOSED)
```

(continues on next page)

(continued from previous page)

```
{
  $tpl_options = array();
  for ($level = $MinLevelAccess; $level <= $MaxLevelAccess; $level++)
  {
    $tpl_options[$level] = l10n(sprintf('ACCESS_%d', $level));
  }
  return $tpl_options;
}
```

17.2.81 Unused Inherited Variable In Closure

shopware

Unused Inherited Variable In Closure, in recovery/update/src/app.php:129.

In the first closuree, $containere is used as the root for the method calls, but $app is not used. It may be dropped. In fact, some of the following calls to $app->map() only request one inherited, $container.

```
$app->map('/applyMigrations', function () use ($app, $container) {
    $container->get('controller.batch')->applyMigrations();
})->via('GET', 'POST')->name('applyMigrations');

$app->map('/importSnippets', function () use ($container) {
    $container->get('controller.batch')->importSnippets();
})->via('GET', 'POST')->name('importSnippets');
```

Mautic

Unused Inherited Variable In Closure, in MauticCrmBundle/Tests/Integration/SalesforceIntegrationTest.php:1202.

$max is relayed to getLeadsToCreate(), while $restart is omitted. It may be dropped, along with its reference.

```
function () use (&$restart, $max) {
                    $args = func_get_args();

                    if (false === $args[2]) {
                        return $max;
                    }

                    $createLeads = $this->getLeadsToCreate($args[2], $max);

                    // determine whether to return a count or records
                    if (false === $args[2]) {
                        return count($createLeads);
                    }

                    return $createLeads;
                }
```

17.2.82 Use Constant As Arguments

Tikiwiki

Use Constant As Arguments, in lib/language/Language.php:112.

E_WARNING is a valid value, but PHP documentation for trigger_error() explains that E_USER constants should be used.

```
trigger_error("Octal or hexadecimal string '" . $match[1] . "' not supported", E_
↪WARNING)
```

shopware

Use Constant As Arguments, in engine/Shopware/Plugins/Default/Core/Debug/Components/EventCollector.php:106.

One example where code review reports errors where unit tests don't : array_multisort actually requires sort order first (SORT_ASC or SORT_DESC), then sort flags (such as SORT_NUMERIC). Here, with SORT_DESC = 3 and SORT_NUMERIC = 1, PHP understands it as the coders expects it. The same error is repeated six times in the code.

```
array_multisort($order, SORT_NUMERIC, SORT_DESC, $this->results)
```

17.2.83 Useless Referenced Argument

Woocommerce

Useless Referenced Argument, in includes/data-stores/class-wc-product-variation-data-store-cpt.php:414.

$product is defined with a reference in the method signature, but it is also used as an object with a dynamical property. As such, the reference in the argument definition is too much.

```
public function update_post_meta( &$product, $force = false ) {
            $meta_key_to_props = array(
                    '_variation_description' => 'description',
            );

            $props_to_update = $force ? $meta_key_to_props : $this->get_props_to_
↪update( $product, $meta_key_to_props );

            foreach ( $props_to_update as $meta_key => $prop ) {
                                    $value   = $product->{get_$prop}( 'edit' );
                                    $updated = update_post_meta( $product->get_id(),
↪$meta_key, $value );
                    if ( $updated ) {
                            $this->updated_props[] = $prop;
                    }
            }

            parent::update_post_meta( $product, $force );
```

Magento

Useless Referenced Argument, in setup/src/Magento/Setup/Module/Di/Compiler/Config/Chain/PreferencesResolving.php:63.

$value is defined with a reference. In the following code, it is only read and never written : for index search, or by itself. In fact, $preferences is also only read, and never written. As such, both could be removed.

```
private function resolvePreferenceRecursive(&$value, &$preferences)
    {
        return isset($preferences[$value])
            ? $this->resolvePreferenceRecursive($preferences[$value], $preferences)
            : $value;
    }
```

17.2.84 Useless Return

ThinkPHP

Useless Return, in library/think/Request.php:2121.

__set() doesn't need a return, unlike __get().

```
public function __set($name, $value)
    {
        return $this->param[$name] = $value;
    }
```

Vanilla

Useless Return, in applications/dashboard/views/attachments/attachment.php:14.

The final 'return' is useless : return void (here, return without argument), is the same as returning null, unless the 'void' return type is used. The other return, is in the two conditions, is important to skip the end of the functioncall.

```
function writeAttachment($attachment) {

        $customMethod = AttachmentModel::getWriteAttachmentMethodName($attachment[
↪'Type']);
        if (function_exists($customMethod)) {
            if (val('Error', $attachment)) {
                writeErrorAttachment($attachment);
                return;
            }
            $customMethod($attachment);
        } else {
            trace($customMethod, 'Write Attachment method not found');
            trace($attachment, 'Attachment');
        }
        return;
    }
```

17.2.85 Wrong Number Of Arguments

xataface

Wrong Number Of Arguments, in actions/existing_related_record.php:130.

df_display() actually requires only 2 arguments, while three are provided. The last argument is completely ignored. df_display() is called in a total of 9 places : this now looks like an API change that left many calls untouched.

```
df_display($context, $template, true);

// in public-api.php :
function df_display($context, $template_name){
    import( 'Dataface/SkinTool.php');
    $st = Dataface_SkinTool::getInstance();

    return $st->display($context, $template_name);
}
```

17.2.86 Wrong Optional Parameter

FuelCMS

Wrong Optional Parameter, in fuel/modules/fuel/helpers/validator_helper.php:78.

The $regex parameter should really be first, as it is compulsory. Though, if this is a legacy function, it may be better to give regex a default value, such as empty string or null, and test it before using it.

```
if (!function_exists('regex'))
{
    function regex($var = null, $regex)
    {
            return preg_match('#'.$regex.'#', $var);
    }
}
```

Vanilla

Wrong Optional Parameter, in applications/dashboard/modules/class.navmodule.php:99.

Note the second parameter, $dropdown, which has no default value. It is relayed to the addDropdown method, which as no default value too. Since both methods are documented, we can see that they should be an addDropdown : null is probably a good idea, coupled with an explicit check on the actual value.

```
/**
     * Add a dropdown to the items array if it satisfies the $isAllowed condition.
     *
     * @param bool|string|array $isAllowed Either a boolean to indicate whether to␣
↪actually add the item
     * or a permission string or array of permission strings (full match) to check.
     * @param DropdownModule $dropdown The dropdown menu to add.
     * @param string $key The item's key (for sorting and CSS targeting).
     * @param string $cssClass The dropdown wrapper's CSS class.
     * @param array|int $sort Either a numeric sort position or and array in the␣
↪style: array('before|after', 'key').
     * @return NavModule $this The calling object.
     */
    public function addDropdownIf($isAllowed = true, $dropdown, $key = '', $cssClass␣
↪= '', $sort = []) {
        if (!$this->isAllowed($isAllowed)) {
            return $this;
        } else {
            return $this->addDropdown($dropdown, $key, $cssClass, $sort);
```

(continues on next page)

(continued from previous page)

```
        }
    }
```

17.2.87 Already Parents Interface

WordPress

Already Parents Interface, in src/Phinx/Db/Adapter/AbstractAdapter.php:41.

SqlServerAdapter extends PdoAdapter, PdoAdapter extends AbstractAdapter. The first and the last both implements AdapterInterface. Only one is needed.

```
/**
 * Base Abstract Database Adapter.
 */
abstract class AbstractAdapter implements AdapterInterface
{

/// In the src/src/Phinx/Db/Adapter/SqlServerAdapter.php, line 45
/**
 * Phinx SqlServer Adapter.
 *
 */
class SqlServerAdapter extends PdoAdapter implements AdapterInterface
{
```

Thelia

Already Parents Interface, in core/lib/Thelia/Core/Template/Loop/BaseSpecificModule.php:35.

PropelSearchLoopInterface is implemented by both BaseSpecificModule and Payment

```
abstract class BaseSpecificModule extends BaseI18nLoop implements␣
↪PropelSearchLoopInterface

/* in file  core/lib/Thelia/Core/Template/Loop/Payment.php, line 28 */

class Payment extends BaseSpecificModule implements PropelSearchLoopInterface
```

17.2.88 Undefined Interfaces

xataface

Undefined Interfaces, in Dataface/Error.php:112.

Exception seems to be a typo, and leads to an always-true expression.

```
public static function isError($obj){
            if ( !PEAR::isError($obj) and !($obj instanceof Exception_) ) return␣
↪false;
            return ($obj->getCode() >= DATAFACE_E_ERROR);
    }
```

17.2.89 Unused Interfaces

Tine20

Unused Interfaces, in tine20/Tinebase/User/LdapPlugin/Interface.php:20.

Tinebase_User_LdapPlugin_Interface is mentioned as a type for a property, in a php doc document. Typehinted properties are available since PHP 7.4

```
interface Tinebase_User_LdapPlugin_Interface {

//----------
// in tine20/Tinebase/User/ActiveDirectory.php
/** @var Tinebase_User_LdapPlugin_Interface $plugin */
```

17.2.90 Useless Interfaces

Woocommerce

Useless Interfaces, in includes/interfaces/class-wc-order-item-data-store-interface.php:20.

WC_Order_Item_Data_Store_Interface is used to structure the class WC_Order_Item_Data_Store. It is not used anywhere else.

```
interface WC_Order_Item_Data_Store_Interface {


////////
//includes/data-stores/class-wc-order-item-data-store.php

class WC_Order_Item_Data_Store implements WC_Order_Item_Data_Store_Interface {
```

17.2.91 Hidden Use Expression

Tikiwiki

Hidden Use Expression, in lib/core/Tiki/Command/DailyReportSendCommand.php:17.

Sneaky error_reporting, hidden among the use calls.

```
namespace Tiki\Command;

use Symfony\Component\Console\Command\Command;
use Symfony\Component\Console\Input\InputArgument;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Input\InputOption;
use Symfony\Component\Console\Output\OutputInterface;
error_reporting(E_ALL);
use TikiLib;
use Reports_Factory;
```

OpenEMR

Hidden Use Expression, in interface/patient_file/summary/browse.php:23.

Use expression is only reached when the csrf token is checked. This probably save some CPU when no csrf is available, but it breaks the readability of the file.

```
<?php
/**
 * Patient selector for insurance gui
 *
 * @package   OpenEMR
 * @link      http://www.open-emr.org
 * @author    Brady Miller <brady.g.miller@gmail.com>
 * @copyright Copyright (c) 2018 Brady Miller <brady.g.miller@gmail.com>
 * @license   https://github.com/openemr/openemr/blob/master/LICENSE GNU General␣
↪Public License 3
 */


require_once(../../globals.php);
require_once($srcdir/patient.inc);
require_once($srcdir/options.inc.php);

if (!empty($_POST)) {
    if (!verifyCsrfToken($_POST[csrf_token_form])) {
        csrfNotVerified();
    }
}

use OpenEMR\Core\Header;
```

17.2.92 Multiple Alias Definitions

ChurchCRM

Multiple Alias Definitions, in Various files:–.

It is actually surprising to find FamilyQuery defined as ChurchCRMBaseFamilyQuery only once, while all other reference are for ChurchCRMFamilyQuery. That lone use is actually useful in the code, so it is not a forgotten refactorisation.

```
use ChurchCRM\Base\FamilyQuery      // in /src/MapUsingGoogle.php:7

use ChurchCRM\FamilyQuery   // in /src/ChurchCRM/Dashboard/EventsDashboardItem.php:8
                            // and 29 other files
```

Phinx

Multiple Alias Definitions, in Various files too:–.

One 'Command' is refering to a local Command class, while the other is refering to an imported class. They are all in a similar name space ConsoleCommand.

```
use Phinx\Console\Command                          //in file /src/Phinx/Console/
↪PhinxApplication.php:34
use Symfony\Component\Console\Command\Command       //in file /src/Phinx/Console/
↪Command/Init.php:31
use Symfony\Component\Console\Command\Command       //in file /src/Phinx/Console/
↪Command/AbstractCommand.php:32
```

17.2.93 No array_merge() In Loops

Tine20

No array_merge() In Loops, in tine20/Tinebase/User/Ldap.php:670.

Classic example of array_merge() in loop : here, the attributures should be collected in a local variable, and then merged in one operation, at the end. That includes the attributes provided before the loop, and the array provided after the loop. Note that the order of merge will be the same when merging than when collecting the arrays.

```
$attributes = array_values($this->_rowNameMapping);
        foreach ($this->_ldapPlugins as $plugin) {
            $attributes = array_merge($attributes, $plugin->getSupportedAttributes());
        }

        $attributes = array_merge($attributes, $this->_
↪additionalLdapAttributesToFetch);
```

17.2.94 Double array_flip()

NextCloud

Double array_flip(), in lib/public/AppFramework/Http/EmptyContentSecurityPolicy.php:372.

The array $allowedScriptDomains is flipped, to unset 'self', then, unflipped (or flipped again), to restore its initial state. Using array_keys() or array_search() would yield the needed keys for unsetting, at a lower cost.

```
if(is_string($this->useJsNonce)) {
                            $policy .= '\'nonce-'.base64_encode($this->useJsNonce).'\'
↪';
                            $allowedScriptDomains = array_flip($this->
↪allowedScriptDomains);
                            unset($allowedScriptDomains['\'self\'']);
                            $this->allowedScriptDomains = array_flip(
↪$allowedScriptDomains);
                            if(count($allowedScriptDomains) !== 0) {
                                    $policy .= ' ';
                            }
                    }
```

17.2.95 Isset() On The Whole Array

Tine20

Isset() On The Whole Array, in tine20/Crm/Model/Lead.php:208.

Only the second call is necessary : it also includes the first one.

```
isset($relation['related_record']) && isset($relation['related_record']['n_fileas'])
```

ExpressionEngine

Isset() On The Whole Array, in system/ee/legacy/libraries/Form_validation.php:1487.

This is equivalent to *isset($this->_field_data[$field], $this->_field_data[$field]['postdata'])*, and the second call may be skipped.

```
!isset($this->_field_data[$field]) OR !isset($this->_field_data[$field]['postdata'])
```

17.2.96 Joining file()

WordPress

Joining file(), in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

SPIP

Joining file(), in ecrire/inc/install.php:109.

When the file is not accessible, file() returns null, and can't be processed by join().

```
$s = @join('', file($file));
```

17.2.97 Logical To in_array

Zencart

Logical To in_array, in admin/users.php:32.

Long list of == are harder to read. Using an in_array() call gathers all the strings together, in an array. In turn, this helps readability and possibility, reusability by making that list an constant.

```
// if needed, check that a valid user id has been passed
if ((($action == 'update' || $action == 'reset') && isset($_POST['user']))
{
  $user = $_POST['user'];
}
elseif ((($action == 'edit' || $action == 'password' || $action == 'delete' || $action␣
↪== 'delete_confirm') && $_GET['user'])
{
  $user = $_GET['user'];
}
elseif((($action=='delete' || $action=='delete_confirm') && isset($_POST['user']))
{
```

(continues on next page)

(continued from previous page)

```
  $user = $_POST['user'];
}
```

17.2.98 Make One Call With Array

HuMo-Gen

Make One Call With Array, in admin/include/kcfinder/lib/helper_text.php:47.

The three calls to str_replace() could be replaced by one, using array arguments. Nesting the calls doesn't reduce the number of calls.

```
static function jsValue($string) {
        return
            preg_replace('/\r?\n/', "\n",
            str_replace('"', "\\"",
            str_replace("'", "\'",
            str_replace("\", "\\",
        $string))));
    }
```

Edusoho

Make One Call With Array, in src/AppBundle/Common/StringToolkit.php:55.

Since str_replace is already using an array, the second argument must also be an array, with repeated empty strings. That syntax allows adding the ' ' and ' ' to those arrays. Note also that trim() should be be called early, but since some of the replacing may generate terminal spaces, it should be kept as is.

```
$text = strip_tags($text);

        $text = str_replace(array(\n, \r, \t), '', $text);
        $text = str_replace(' ', ' ', $text);
        $text = trim($text);
```

17.2.99 Avoid Concat In Loop

SuiteCrm

Avoid Concat In Loop, in include/export_utils.php:433.

$line is build in several steps, then then final version is added to $content. It would be much faster to make $content an array, and implode it once after the loop.

```
foreach($records as $record)
        {
            $line = implode("\"" . getDelimiter() . "\"", $record);
            $line = "\"" . $line;
            $line .= "\"\r\n";
            $line = parseRelateFields($line, $record, $customRelateFields);
            $content .= $line;
        }
```

ThinkPHP

Avoid Concat In Loop, in ThinkPHP/Common/functions.php:720.

The foreach loop appends the $name and builds a fully qualified name.

```
if (!C('APP_USE_NAMESPACE')) {
        $class = parse_name($name, 1);
        import($module . '/' . $layer . '/' . $class . $layer);
    } else {
        $class = $module . '\' . $layer;
        foreach ($array as $name) {
            $class .= '\' . parse_name($name, 1);
        }
        //
        if ($extend) {
            //
            $class = $extend . '\' . $class;
        }
    }
    return $class . $layer;
```

17.2.100 Avoid glob() Usage

Phinx

Avoid glob() Usage, in src/Phinx/Migration/Manager.php:362.

glob() searches for a list of files in the migration folder. Those files are not known, but they have a format, as checked later with the regex : a combinaison of `FilesystemIterator` and `RegexIterator` would do the trick too.

```
$phpFiles = glob($config->getMigrationPath() . DIRECTORY_SEPARATOR . '*.php');

            // filter the files to only get the ones that match our naming scheme
            $fileNames = array();
            /** @var AbstractMigration[] $versions */
            $versions = array();

            foreach ($phpFiles as $filePath) {
                if (preg_match('/([0-9]+)_([_a-z0-9]*).php/', basename($filePath))) {
```

NextCloud

Avoid glob() Usage, in lib/private/legacy/helper.php:185.

Recursive copy of folders, based on scandir(). `DirectoryIterator` and `FilesystemIterator` would do the same without the recursion.

```
static function copyr($src, $dest) {
            if (is_dir($src)) {
                    if (!is_dir($dest)) {
                            mkdir($dest);
                    }
                    $files = scandir($src);
                    foreach ($files as $file) {
```

(continues on next page)

(continued from previous page)

```
                                if ($file != "." && $file != "..") {
                                        self::copyr("$src/$file", "$dest/$file");
                                }
                        }
                } elseif (file_exists($src) && !\OC\Files\Filesystem::isFileBlacklisted(
↪$src)) {
                        copy($src, $dest);
                }
        }
```

17.2.101 No Count With 0

Contao

No Count With 0, in system/modules/repository/classes/RepositoryManager.php:1148.

If $elist contains at least one element, then it is not empty().

```
$ext->found = count($elist)>0;
```

WordPress

No Count With 0, in wp-admin/includes/misc.php:74.

$build or $signature are empty at that point, no need to calculate their respective length.

```
// Check for zero length, although unlikely here
    if (strlen($built) == 0 || strlen($signature) == 0) {
      return false;
    }
```

17.2.102 Pre-increment

ExpressionEngine

Pre-increment, in system/ee/EllisLab/ExpressionEngine/Controller/Utilities/Communicate.php:650.

Using preincrement in for() loops is safe and straightforward.

```
for ($x = 0; $x < $number_to_send; $x++)
                {
                        $email_address = array_shift($recipient_array);

                        if ( ! $this->deliverEmail($email, $email_address))
                        {
                                $email->delete();

                                $debug_msg = ee()->email->print_debugger(array());

                                show_error(lang('error_sending_email').BR.BR.$debug_msg);
                        }
                        $email->total_sent++;
                }
```

Traq

Pre-increment, in src/Controllers/Tickets.php:84.

$this->currentProject->next_ticket_id value is ignored by the code. It may be turned into a preincrement.

```
TimelineModel::newTicketEvent($this->currentUser, $ticket)->save();

            $this->currentProject->next_ticket_id++;
            $this->currentProject->save();
```

17.2.103 Slow Functions

ChurchCRM

Slow Functions, in src/Reports/PrintDeposit.php:35.

You may replace this with a isset() : $_POST can't contain a NULL value, unless it was set by the script itself.

```
array_key_exists("report_type", $_POST);
```

SuiteCrm

Slow Functions, in include/json_config.php:242.

This is a equivalent for nl2br()

```
preg_replace("/\r\n/", "<BR>", $focus->$field)
```

17.2.104 strpos() Too Much

WordPress

strpos() Too Much, in core/traits/Request/Server.php:127.

Instead of searching for `HTTP_`, it is faster to compare the first 5 chars to the literal `HTTP_`. In case of absence, this solution returns faster.

```
if (strpos($header, 'HTTP_') === 0) {
                        $header = substr($header, 5);
                    } elseif (strpos($header, 'CONTENT_') !== 0) {
                        continue;
                    }
```

17.2.105 Substring First

SPIP

Substring First, in ecrire/inc/filtres.php:1694.

The code first makes everything uppercase, including the leading and trailing spaces, and then, removes them : it would be best to swap those operations. Note that spip_substr() is not considered in this analysis, but with SPIP knowledge, it could be moved inside the calls.

```
function filtre_initiale($nom) {
    return spip_substr(trim(strtoupper(extraire_multi($nom))), 0, 1);
}
```

PrestaShop

Substring First, in admin-dev/filemanager/include/utils.php:197.

dirname() reduces the string (or at least, keeps it the same size), so it more efficient to have it first.

```
dirname(str_replace(' ', '~', $str))
```

17.2.106 time() Vs strtotime()

Woocommerce

time() Vs strtotime(), in includes/class-wc-webhook.php:384.

time() would be faster here, as an entropy generator. Yet, it would still be better to use an actual secure entropy generator, like random_byte or random_int. In case of older version, microtime() would yield better entropy.

```
public function get_new_delivery_id() {
            // Since we no longer use comments to store delivery logs, we generate a␣
↪unique hash instead based on current time and webhook ID.
            return wp_hash( $this->get_id() . strtotime( 'now' ) );
    }
```

17.2.107 Assign With And Precedence

xataface

Assign With And Precedence, in Dataface/LanguageTool.php:265.

The usage of 'and' here is a workaround for PHP version that have no support for the coalesce. $autosubmit receives the value of $params['autosubmit'] only if the latter is set. Yet, with = having higher precedence over 'and', $autosubmit is mistaken with the existence of $params['autosubmit'] : its value is actually omitted.

```
$autosubmit = isset($params['autosubmit']) and $params['autosubmit'];
```

17.2.108 Avoid set_error_handler $context Argument

shopware

Avoid set_error_handler $context Argument, in engine/Shopware/Plugins/Default/Core/ErrorHandler/Bootstrap.php:162.

The registered handler is a local method, called `errorHandler`, which has 6 arguments, and relays those 6 arguments to set_error_handler().

```
public function registerErrorHandler($errorLevel = E_ALL)
    {
        // Only register once.  Avoids loop issues if it gets registered twice.
        if (self::$_registeredErrorHandler) {
            set_error_handler([$this, 'errorHandler'], $errorLevel);

            return $this;
        }

        self::$_origErrorHandler = set_error_handler([$this, 'errorHandler'],
↪$errorLevel);
        self::$_registeredErrorHandler = true;

        return $this;
    }
```

Vanilla

Avoid set_error_handler $context Argument, in library/core/functions.error.php:747.

Gdn_ErrorHandler is a function that requires 6 arguments.

```
set_error_handler('Gdn_ErrorHandler', E_ALL & ~E_STRICT)
```

17.2.109 Use random_int()

Thelia

Use random_int(), in core/lib/Thelia/Tools/TokenProvider.php:151.

The whole function may be replaced by random_int(), as it generates random tokens. This needs an extra layer of hashing, to get a long and string results.

```
/**
     * @return string
     */
    protected static function getComplexRandom()
    {
        $firstValue = (float) (mt_rand(1, 0xFFFF) * rand(1, 0x10001));
        $secondValues = (float) (rand(1, 0xFFFF) * mt_rand(1, 0x10001));

        return microtime() . ceil($firstValue / $secondValues) . uniqid();
    }
```

FuelCMS

Use random_int(), in fuel/modules/fuel/libraries/Fuel.php:235.

Security tokens should be build with a CSPRNG source. uniqid() is based on time, and though it changes anytime (sic), it is easy to guess. Those days, it looks like '5b1262e74dbb9';

```
$this->installer->change_config('config', '$config[\'encryption_key\'] = \'\';', '
↪$config[\'encryption_key\'] = \''.md5(uniqid()).'\';');
```

17.2.110 __debugInfo() Usage

Dolibarr

__debugInfo() Usage, in htdocs/includes/stripe/lib/StripeObject.php:108.

_values is a private property from the Stripe Class. The class contains other objects, but only _values are displayed with var_dump.

```
// Magic method for var_dump output. Only works with PHP >= 5.6
    public function __debugInfo()
    {
        return $this->_values;
    }
```

17.2.111 Deprecated PHP Functions

Dolphin

Deprecated PHP Functions, in Dolphin-v.7.3.5/inc/classes/BxDolAdminSettings.php:270.

Split() was abandonned in PHP 7.0

```
split(',', $aItem['extra']);
```

17.2.112 Wrong fopen() Mode

Tikiwiki

Wrong fopen() Mode, in lib/tikilib.php:6777.

This fopen() mode doesn't exists. Use 'w' instead.

```
fopen('php://temp', 'rw');
```

HuMo-Gen

Wrong fopen() Mode, in include/phprtflite/lib/PHPRtfLite/StreamOutput.php:77.

This fopen() mode doesn't exists. Use 'w' instead.

```
fopen($this->_filename, 'wr', false)
```

17.2.113 Incompilable Files

xataface

Incompilable Files, in lib/XML/Tree.php:289.

Compilation error with PHP 7.2 version.

```
syntax error, unexpected 'new' (T_NEW)
```

17.2.114 Wrong Parameter Type

Zencart

Wrong Parameter Type, in admin/includes/header.php:180.

setlocale() may be called with null or '' (empty string), and will set values from the environment. When called with "0" (the string), it only reports the current setting. Using an integer is probably undocumented behavior, and falls back to the zero string.

```
$loc = setlocale(LC_TIME, 0);
        if ($loc !== FALSE) echo ' - ' . $loc; //what is the locale in use?
```

17.2.115 Isset Multiple Arguments

ThinkPHP

Isset Multiple Arguments, in library/think/Request.php:1187.

This may be shortened with isset($sub), $array[$name][$sub])

```
isset($sub) && isset($array[$name][$sub])
```

LiveZilla

Isset Multiple Arguments, in livezilla/_lib/trdp/pchart/class/pDraw.class.php:3852.

This is the equivalent of !(isset($Data["Series"][$SerieA]["Data"]) && isset($Data["Series"][$SerieB]["Data"])), and then, !(isset($Data["Series"][$SerieA]["Data"], $Data["Series"][$SerieB]["Data"]))

```
!isset($Data["Series"][$SerieA]["Data"]) || !isset($Data["Series"][$SerieB]["Data"])
```

17.2.116 Logical Should Use Symbolic Operators

Cleverstyle

Logical Should Use Symbolic Operators, in modules/Uploader/Mime/Mime.php:171.

$extension is assigned with the results of pathinfo($reference_name, PATHINFO_EXTENSION) and ignores static::hasExtension($extension). The same expression, placed in a condition (like an if), would assign a value to $extension and use another for the condition itself. Here, this code is only an expression in the flow.

```
$extension = pathinfo($reference_name, PATHINFO_EXTENSION) and static::hasExtension(
↪$extension);
```

OpenConf

Logical Should Use Symbolic Operators, in chair/export.inc:143.

In this context, the priority of execution is used on purpose; $coreFile only collect the temporary name of the export file, and when this name is empty, then the second operand of OR is executed, though never collected. Since this second argument is a 'die', its return value is lost, but the initial assignation is never used anyway.

```
$coreFile = tempnam('/tmp/', 'ocexport') or die('could not generate Excel file (6)')
```

17.2.117 Possible Missing Subpattern

phpMyAdmin

Possible Missing Subpattern, in libraries/classes/Advisor.php:557.

The last capturing subpattern is `( \[(.*)\])?` and it is optional. Indeed, when the pattern succeed, the captured values are stored in `$match`. Yet, the code checks for the existence of `$match[3]` before using it.

```
if (preg_match("/rule\s'(.*)'( \[(.*)\])?$/", $line, $match)) {
                    $ruleLine = 1;
                    $ruleNo++;
                    $rules[$ruleNo] = ['name' => $match[1]];
                    $lines[$ruleNo] = ['name' => $i + 1];
                    if (isset($match[3])) {
                        $rules[$ruleNo]['precondition'] = $match[3];
                        $lines[$ruleNo]['precondition'] = $i + 1;
                    }
```

SPIP

Possible Missing Subpattern, in ecrire/inc/filtres_dates.php:73.

This code avoid the PHP notice by padding the resulting array (see comment in French : eviter === avoid)

```
if (preg_match("#^([12][0-9]{3}[-/][01]?[0-9])([-/]00)?( [-0-9:]+)?$#", $date,
↪$regs)) {
                            $regs = array_pad($regs, 4, null); // eviter notice php
                            $date = preg_replace("@/@", "-", $regs[1]) . "-00" .
↪$regs[3];
                    } else {
                            $date = date("Y-m-d H:i:s", strtotime($date));
                    }
```

17.2.118 ** For Exponent

Traq

** *For Exponent*, in src/views/layouts/_footer.phtm:5.

pow(1024, 2) could be (1023 ** 2), to convert bytes into Mb.

```
<?=round((microtime(true) - START_TIME), 2); ?>s, <?php echo round((memory_get_peak_
↪usage() - START_MEM) / pow(1024, 2), 3)?>mb
```

TeamPass

** *For Exponent*, in includes/libraries/Authentication/phpseclib/Math/BigInteger.php:286.

pow(2, 62) could also be hard coded with 0x4000000000000000.

```
pow(2, 62)
```

17.2.119 No Class In Global

Dolphin

No Class In Global, in Dolphin-v.7.3.5/inc/classes/BxDolXml.php:10.

This class should be put away in a 'dolphin' or 'boonex' namespace.

```
class BxDolXml {
    /* class BxDolXML code */
}
```

17.2.120 No Reference For Ternary

phpadsnew

No Reference For Ternary, in lib/OA/Admin/Menu/Section.php334:334.

The reference should be removed from the function definition. Either this method returns null, which is never a reference, or it returns $this, which is always a reference, or the results of a methodcall. The latter may or may not be a reference, but the Ternary operator will drop it and return by value.

```
function &getParentOrSelf($type)
    {
        if ($this->type == $type) {
            return $this;
        }
        else {
            return $this->parentSection != null ? $this->parentSection->
↪getParentOrSelf($type) : null;
        }
    }
```

17.2.121 Old Style __autoload()

Piwigo

Old Style __autoload(), in include/phpmailer/PHPMailerAutoload.php:45.

This code handles situations for PHP after 5.1.0 and older. Rare are the applications that are still using those versions in 2019.

```
if (version_compare(PHP_VERSION, '5.1.2', '>=')) {
    //SPL autoloading was introduced in PHP 5.1.2
    if (version_compare(PHP_VERSION, '5.3.0', '>=')) {
        spl_autoload_register('PHPMailerAutoload', true, true);
    } else {
        spl_autoload_register('PHPMailerAutoload');
    }
} else {
    /**
     * Fall back to traditional autoload for old PHP versions
     * @param string $classname The name of the class to load
     */
    function __autoload($classname)
    {
        PHPMailerAutoload($classname);
    }
}
```

17.2.122 Pathinfo() Returns May Vary

NextCloud

Pathinfo() Returns May Vary, in lib/private/Preview/Office.php:56.

$absPath is build with the toTmpFile() method, which may return a boolean (false) in case of error. Error situations include the inability to create the temporary file.

```
$absPath = $fileview->toTmpFile($path);

// More code

                    list($dirname, , , $filename) = array_values(pathinfo($absPath));
                    $pngPreview = $dirname . '/' . $filename . '.png';
```

17.2.123 preg_match_all() Flag

FuelCMS

preg_match_all() Flag, in fuel/modules/fuel/helpers/MY_array_helper.php:205.

Using PREG_SET_ORDER will remove the usage of the ‘‘$key‘‘variable.

```
function parse_string_to_array($str)
    {
            preg_match_all('#(\w+)=([\'"])(.*)\2#U', $str, $matches);
            $params = array();
            foreach($matches[1] as $key => $val)
            {
                    if (!empty($matches[3]))
                    {
                            $params[$val] = $matches[3][$key];
                    }
            }
```

(continues on next page)

(continued from previous page)

```
            return $params;
    }
```

17.2.124 PHP Keywords As Names

ChurchCRM

PHP Keywords As Names, in src/kiosk/index.php:42.

$false may be true or false (or else...). In fact, the variable is not even defined in this file, and the file do a lot of inclusion.

```
if (!isset($_COOKIE['kioskCookie'])) {
    if ($windowOpen) {
        $guid = uniqid();
        setcookie("kioskCookie", $guid, 2147483647);
        $Kiosk = new \ChurchCRM\KioskDevice();
        $Kiosk->setGUIDHash(hash('sha256', $guid));
        $Kiosk->setAccepted($false);
        $Kiosk->save();
    } else {
        header("HTTP/1.1 401 Unauthorized");
        exit;
    }
}
```

xataface

PHP Keywords As Names, in Dataface/Record.php:1278.

This one is documented, and in the end, makes a lot of sense.

```
function &getRelatedRecord($relationshipName, $index=0, $where=0, $sort=0){
            if ( isset($this->cache[__FUNCTION__][$relationshipName][$index][$where][
↪$sort]) ){
                    return $this->cache[__FUNCTION__][$relationshipName][$index][
↪$where][$sort];
            }
            $it = $this->getRelationshipIterator($relationshipName, $index, 1, $where,
↪ $sort);
            if ( $it->hasNext() ){
                    $rec =& $it->next();
                    $this->cache[__FUNCTION__][$relationshipName][$index][$where][
↪$sort] =& $rec;
                    return $rec;
            } else {
                    $null = null;    // stupid hack because literal 'null' can't be␣
↪returned by ref.
                    return $null;
            }
    }
```

17.2.125 Should Preprocess Chr()

phpadsnew

Should Preprocess Chr(), in phpAdsNew-2.0/adview.php:302.

Each call to chr() may be done before. First, chr() may be replace with the hexadecimal sequence "0x3B"; Secondly, 0x3b is a rather long replacement for a simple semi-colon. The whole pragraph could be stored in a separate file, for easier modifications.

```
echo chr(0x47).chr(0x49).chr(0x46).chr(0x38).chr(0x39).chr(0x61).chr(0x01).chr(0x00).
                  chr(0x01).chr(0x00).chr(0x80).chr(0x00).chr(0x00).chr(0x04).
↪chr(0x02).chr(0x04).
                      chr(0x00).chr(0x00).chr(0x00).chr(0x21).chr(0xF9).chr(0x04).
↪chr(0x01).chr(0x00).
                  chr(0x00).chr(0x00).chr(0x00).chr(0x2C).chr(0x00).chr(0x00).
↪chr(0x00).chr(0x00).
                  chr(0x01).chr(0x00).chr(0x01).chr(0x00).chr(0x00).chr(0x02).
↪chr(0x02).chr(0x44).
                  chr(0x01).chr(0x00).chr(0x3B);
```

17.2.126 Should Use array_filter()

xataface

Should Use array_filter(), in actions/manage_build_index.php:38.

This selection process has three tests : the two first are exclusive, and the third is inclusive. They could fit in one or several closures.

```
$indexable = array();
            foreach ( $tables as $key=>$table ){
                    if ( preg_match('/^dataface__/', $table) ){
                            continue;
                    }
                    if ( preg_match('/^_/', $table) ){
                            continue;
                    }

                    if ( $index->isTableIndexable($table) ){
                            $indexable[] = $table;
                            //unset($tables[$key]);
                    }

            }
```

shopware

Should Use array_filter(), in engine/Shopware/Bundle/StoreFrontBundle/Service/Core/VariantCoverService.php:71.

Closure would be the best here, since $covers has to be injected in the array_filter callback.

```
$covers = $this->variantMediaGateway->getCovers(
            $products,
            $context
```

(continues on next page)

(continued from previous page)

```
        );

        $fallback = [];
        foreach ($products as $product) {
            if (!array_key_exists($product->getNumber(), $covers)) {
                $fallback[] = $product;
            }
        }
```

17.2.127 Should Use Coalesce

ChurchCRM

Should Use Coalesce, in src/ChurchCRM/Service/FinancialService.php:597.

ChurchCRM features 5 old style ternary operators, which are all in this SQL query. ChurchCRM requires PHP 7.0, so a simple code review could remove them all.

```
$sSQL = "INSERT INTO pledge_plg
                    (plg_famID,
                    plg_FYID,
                    plg_date,
                    plg_amount,
                    plg_schedule,
                    plg_method,
                    plg_comment,
                    plg_DateLastEdited,
                    plg_EditedBy,
                    plg_PledgeOrPayment,
                    plg_fundID,
                    plg_depID,
                    plg_CheckNo,
                    plg_scanString,
                    plg_aut_ID,
                    plg_NonDeductible,
                    plg_GroupKey)
                    VALUES ('".
          $payment->FamilyID."','".
          $payment->FYID."','".
          $payment->Date."','".
          $Fund->Amount."','".
          (isset($payment->schedule) ? $payment->schedule : 'NULL')."','".
          $payment->iMethod."','".
          $Fund->Comment."','".
          date('YmdHis')."',".
          $_SESSION['user']->getId().",'".
          $payment->type."',".
          $Fund->FundID.','.
          $payment->DepositID.','.
          (isset($payment->iCheckNo) ? $payment->iCheckNo : 'NULL').",'".
          (isset($payment->tScanString) ? $payment->tScanString : 'NULL')."','".
          (isset($payment->iAutID) ? $payment->iAutID : 'NULL')."','".
          (isset($Fund->NonDeductible) ? $Fund->NonDeductible : 'NULL')."','".
          $sGroupKey."')";
```

Cleverstyle

Should Use Coalesce, in modules/Feedback/index.php:37.

Cleverstyle nests ternary operators when selecting default values. Here, moving some of them to ?? will reduce the code complexity and make it more readable. Cleverstyle requires PHP 7.0 or more recent.

```
$Page->content(
    h::{'cs-form form'}(
            h::{'section.cs-feedback-form article'}(
                    h::{'header h2.cs-text-center'}($L->Feedback).
                    h::{'table.cs-table[center] tr| td'}(
                            [
                                    h::{'cs-input-text input[name=name][required]'}(
                                            [
                                                    'placeholder' => $L->feedback_
↪name,
                                                    'value'       => $User->user() ?
↪$User->username() : (isset($_POST['name']) ? $_POST['name'] : '')
                                            ]
                                    ),
                                    h::{'cs-input-text␣
↪input[type=email][name=email][required]'}(
                                            [
                                                    'placeholder' => $L->feedback_
↪email,
                                                    'value'       => $User->user() ?
↪$User->email : (isset($_POST['email']) ? $_POST['email'] : '')
                                            ]
                                    ),
                                    h::{'cs-textarea[autosize]␣
↪textarea[name=text][required]'}(
                                            [
                                                    'placeholder' => $L->feedback_
↪text,
                                                    'value'       => isset($_POST[
↪'text']) ? $_POST['text'] : ''
                                            ]
                                    ),
                                    h::{'cs-button button[type=submit]'}($L->feedback_
↪send)
                            ]
                    )
            )
    )
);
```

17.2.128 Strtr Arguments

SuiteCrm

Strtr Arguments, in includes/vCard.php:221.

This code prepares incoming '$values' for extraction. The keys are cleaned then split with explode(). The '=' sign would stay, as strtr() can't remove it. This means that such keys won't be recognized later in the code, and gets omitted.

```
$values = explode(';', $value);
                    $key = strtoupper($keyvalue[0]);
                    $key = strtr($key, '=', '');
                    $key = strtr($key, ',', ';');
                    $keys = explode(';', $key);
```

17.2.129 Too Many Native Calls

SPIP

Too Many Native Calls, in /ecrire/xml/analyser_dtd.php:58.

This expression counts 4 usages of count(), which is more than the default level of 3 PHP calls in one expression.

```
spip_log("Analyser DTD $avail $grammaire (" . spip_timer('dtd') . ") " . count($dtc->
↪macros) . ' macros, ' . count($dtc->elements) . ' elements, ' . count($dtc->
↪attributs) . " listes d'attributs, " . count($dtc->entites) . " entites")
```

17.2.130 Use PHP Object API

WordPress

Use PHP Object API, in wp-includes/functions.php:2558.

Finfo has also a class, with the same name.

```
finfo_open(FILEINFO_MIME_TYPE)
```

PrestaShop

Use PHP Object API, in admin-dev/filemanager/include/utils.php:174.

transliterator_transliterate() has also a class named Transliterator

```
transliterator_transliterate('Accents-Any', $str)
```

17.2.131 Use Pathinfo

SuiteCrm

Use Pathinfo, in include/utils/file_utils.php:441.

Looking for the extension ? Use pathinfo() and PATHINFO_EXTENSION

```
$exp = explode('.', $filename);
```

17.2.132 Use pathinfo() Arguments

Zend-Config

Use pathinfo() Arguments, in src/Factory.php:74:90.

The *$filepath* is broken into pieces, and then, only the 'extension' part is used. With the PATHINFO_EXTENSION constant used as a second argument, only this value could be returned.

```
$pathinfo = pathinfo($filepath);

        if (! isset($pathinfo['extension'])) {
            throw new Exception\RuntimeException(sprintf(
                'Filename "%s" is missing an extension and cannot be auto-detected',
                $filename
            ));
        }

        $extension = strtolower($pathinfo['extension']);
        // Only $extension is used beyond that point
```

ThinkPHP

Use pathinfo() Arguments, in ThinkPHP/Extend/Library/ORG/Net/UploadFile.class.php:508.

Without any other check, pathinfo() could be used with PATHINFO_EXTENSION.

```
private function getExt($filename) {
        $pathinfo = pathinfo($filename);
        return $pathinfo['extension'];
    }
```

17.2.133 Use session_start() Options

WordPress

Use session_start() Options, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

17.2.134 Compare Hash

Traq

Compare Hash, in src/Models/User.php:105.

This code should also avoid using SHA1.

```
sha1($password) == $this->password
```

LiveZilla

Compare Hash, in livezilla/_lib/objects.global.users.inc.php:1391.

This code is using the stronger SHA256 but compares it to another string. $_token may be non-empty, and still be comparable to 0.

```
function IsValidToken($_token)
{
    if(!empty($_token))
        if(hash("sha256",$this->Token) == $_token)
            return true;
    return false;
}
```

17.2.135 Configure Extract

Zurmo

Configure Extract, in app/protected/modules/marketing/utils/GlobalMarketingFooterUtil.php:127.

This code intent to overwrite *$hash* and *$preview* : it is even literally in the code. The overwrite is intended too, and could even skip the initialisation of the variables. Although the compact()/extract() combinaison is safe as now, it could be safer to only relay the array index, instead of extracting the variables here.

```
public static function resolveManageSubscriptionsUrlByArray(array $queryStringArray,
↪$preview = false)
        {
            $hash = $preview = null;
            extract(static::resolvePreviewAndHashFromArray($queryStringArray));
            return static::resolveManageSubscriptionsUrl($hash, $preview);
        }

// Also with :
        protected static function resolvePreviewAndHashFromArray(array
↪$queryStringArray)
        {
            $preview    = static::resolvePreviewFromArray($queryStringArray);
            $hash       = static::resolveHashByArray($queryStringArray);
            return compact('hash', 'preview');
        }
```

Dolibarr

Configure Extract, in htdocs/includes/restler/framework/Luracast/Restler/Format/HtmlFormat.php:224.

The extract() has been cleverly set in a closure, with a limited scope. The potential overwrite may impact existing variables, such as *$_*, *$nav*, *$form*, and *$data* itself. This may impact the following including. Using EXTR_SKIP would give existing variables priority, and avoid interference.

```
$template = function ($view) use ($data, $path) {
            $form = function () {
                return call_user_func_array(
                    'Luracast\Restler\UI\Forms::get',
                    func_get_args()
```

(continues on next page)

(continued from previous page)

```
            );
        };
        if (!isset($data['form']))
            $data['form'] = $form;
        $nav = function () {
            return call_user_func_array(
                'Luracast\Restler\UI\Nav::get',
                func_get_args()
            );
        };
        if (!isset($data['nav']))
            $data['nav'] = $nav;

        $_ = function () use ($data, $path) {
            extract($data);
            $args = func_get_args();
            $task = array_shift($args);
            switch ($task) {
                case 'require':
                case 'include':
                    $file = $path . $args[0];
                    if (is_readable($file)) {
                        if (
                            isset($args[1]) &&
                            ($arrays = Util::nestedValue($data, $args[1]))
                        ) {
                            $str = '';
                            foreach ($arrays as $arr) {
                                extract($arr);
                                $str .= include $file;
                            }
                            return $str;
                        } else {
                            return include $file;
                        }
                    }
                    break;
                case 'if':
                    if (count($args) < 2)
                        $args[1] = '';
                    if (count($args) < 3)
                        $args[2] = '';
                    return $args[0] ? $args[1] : $args[2];
                    break;
                default:
                    if (isset($data[$task]) && is_callable($data[$task]))
                        return call_user_func_array($data[$task], $args);
            }
            return '';
        };
        extract($data);
        return @include $view;
    };
```

17.2.136 Safe Curl Options

OpenConf

Safe Curl Options, in openconf/include.php:703.

The function that holds that code is only used to call openconf.com, over http, while openconf.com is hosted on https, nowadays. This may be a sign of hard to access certificates.

```
$ch = curl_init();
                    curl_setopt($ch, CURLOPT_URL, $f);
                    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
                    curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
                    curl_setopt($ch, CURLOPT_AUTOREFERER, true);
                    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
                    curl_setopt($ch, CURLOPT_MAXREDIRS, 5);
                    curl_setopt($ch, CURLOPT_HEADER, false);
                    $s = curl_exec($ch);
                    curl_close($ch);
                    return($s);
```

17.2.137 Don't Echo Error

ChurchCRM

Don't Echo Error, in wp-admin/includes/misc.php:74.

This is classic debugging code that should never reach production. mysqli_error() and mysqli_errno() provide valuable information is case of an error, and may be exploited by intruders.

```
if (mysqli_error($cnInfoCentral) != '') {
        echo gettext('An error occured: ').mysqli_errno($cnInfoCentral).'--'.mysqli_
↪error($cnInfoCentral);
    } else {
```

Phpdocumentor

Don't Echo Error, in src/phpDocumentor/Plugin/Graphs/Writer/Graph.php:77.

Default development behavior : display the caught exception. Production behavior should not display that message, but log it for later review. Also, the return in the catch should be moved to the main code sequence.

```
public function processClass(ProjectDescriptor $project, Transformation
↪$transformation)
    {
        try {
            $this->checkIfGraphVizIsInstalled();
        } catch (\Exception $e) {
            echo $e->getMessage();

            return;
        }
```

17.2.138 Encoded Simple Letters

Zurmo

Encoded Simple Letters, in yii/framework/web/CClientScript.php:783.

This actually decodes into a copyright notice.

'function cleanAndSanitizeScriptHeader(& $output)

{ $requiredOne = Copyright © Zurmo Inc., 2013. All rights reserved.;. . . '

```
eval(\x66\x75\x6e\x63\x74\x69\x6f\x6e\x20\x63\x6c\x65\x61\x6e\x41\x6e\x64\x53\x61\x6e\x69\x74\x69\x7a
↪.
    ␣
↪\x69\x70\x74\x48\x65\x61\x64\x65\x72\x28\x26\x20\x24\x6f\x75\x74\x70\x75\x74\x29\x0d\x0a\x20\x20\x2
↪.
    ␣
↪\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x7b\x0d\x0a\x20\x20\x20\x2
↪.
    ␣
↪\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x24\x72\x65\x7
↪.
     // several more lines like that
```

17.2.139 Mkdir Default

Mautic

Mkdir Default, in app/bundles/CoreBundle/Helper/AssetGenerationHelper.php:120.

This code is creating some directories for Javascript or CSS (from the directories names) : those require universal reading access, but probably no execution nor writing access. 0711 would be sufficient in this case.

```
//combine the files into their corresponding name and put in the root media folder
                if ($env == 'prod') {
                    $checkPaths = [
                        $assetsFullPath,
                        $assetsFullPath/css,
                        $assetsFullPath/js,
                    ];
                    array_walk($checkPaths, function ($path) {
                        if (!file_exists($path)) {
                            mkdir($path);
                        }
                    });
```

OpenEMR

Mkdir Default, in interface/main/backuplog.php:27.

If $BACKUP_EVENTLOG_DIR is a backup for an event log, this should be stored out of the web server reach, with low rights, beside the current user. This is part of a CLI PHP script.

```
mkdir($BACKUP_EVENTLOG_DIR)
```

17.2.140 Register Globals

TeamPass

Register Globals, in api/index.php:25.

The API starts with security features, such as the whitelist(). The whitelist applies to IP addresses, so the query string is not sanitized. Then, the QUERY_STRING is parsed, and creates a lot of new global variables.

```
teampass_whitelist();

parse_str($_SERVER['QUERY_STRING']);
$method = $_SERVER['REQUEST_METHOD'];
$request = explode("/", substr(@$_SERVER['PATH_INFO'], 1));
```

XOOPS

Register Globals, in htdocs/modules/system/admin/images/main.php:33:33.

This code only exports the POST variables as globals. And it does clean incoming variables, but not all of them.

```
// Check users rights
if (!is_object($xoopsUser) || !is_object($xoopsModule) || !$xoopsUser->isAdmin(
↪$xoopsModule->mid())) {
    exit(_NOPERM);
}

//  Check is active
if (!xoops_getModuleOption('active_images', 'system')) {
    redirect_header('admin.php', 2, _AM_SYSTEM_NOTACTIVE);
}

if (isset($_POST)) {
    foreach ($_POST as $k => $v) {
        ${$k} = $v;
    }
}

// Get Action type
$op = system_CleanVars($_REQUEST, 'op', 'list', 'string');
```

17.2.141 Should Use Prepared Statement

Dolibarr

Should Use Prepared Statement, in htdocs/product/admin/price_rules.php:76.

This code is well escaped, as the integer type cast will prevent any special chars to be used. Here, a prepared statement would apply a modern approach to securing this query.

```
$db->query("DELETE FROM " . MAIN_DB_PREFIX . "product_pricerules WHERE level = " .␣
↪(int) $i)
```

17.2.142 Unserialize Second Arg

Piwigo

Unserialize Second Arg, in admin/configuration.php:491.

unserialize() extracts information from the $conf variable : this variable is read from a configuration file. It is later tested to be an array, whose index may not be all set (@$disabled[$type];). It would be safer to make $disabled an object, add the class to unserialize, and set default values to the needed properties/index.

```
$disabled = @unserialize(@$conf['disabled_derivatives']);
```

LiveZilla

Unserialize Second Arg, in livezilla/_lib/objects.global.inc.php:2600.

unserialize() only extract a non-empty value here. But its content is not checked. It is later used as an array, with multiple index.

```
$this->Customs = (!empty($_row["customs"])) ? @unserialize($_row["customs"]) :␣
↪array();
```

17.2.143 Adding Zero

Thelia

Adding Zero, in core/lib/Thelia/Model/Map/ProfileResourceTableMap.php:250.

This return statement is doing quite a lot, including a buried '0 + $offset'. This call is probably an echo to '1 + $offset', which is a little later in the expression.

```
return serialize(array((string) $row[TableMap::TYPE_NUM == $indexType ? 0 + $offset :␣
↪static::translateFieldName('ProfileId', TableMap::TYPE_PHPNAME, $indexType)],␣
↪(string) $row[TableMap::TYPE_NUM == $indexType ? 1 + $offset :␣
↪static::translateFieldName('ResourceId', TableMap::TYPE_PHPNAME, $indexType)]));
```

OpenEMR

Adding Zero, in interface/forms/fee_sheet/new.php:466:534.

$main_provid is filtered as an integer. $main_supid is then filtered twice : one with the sufficent (int) and then, added with 0.

```
if (!$alertmsg && ($_POST['bn_save'] || $_POST['bn_save_close'] || $_POST['bn_save_
↪stay'])) {
    $main_provid = 0 + $_POST['ProviderID'];
    $main_supid  = 0 + (int)$_POST['SupervisorID'];
    //.....
```

17.2.144 Altering Foreach Without Reference

Contao

Altering Foreach Without Reference, in core-bundle/src/Resources/contao/classes/Theme.php:613.

$tmp[$kk] is &$vv.

```
foreach ($tmp as $kk=>$vv)
                                                                {
                                                                        // Do not use the␣
↪FilesModel here - tables are locked!
                                                                        $objFile = $this->
↪Database->prepare(SELECT uuid FROM tl_files WHERE path=?)
                                                                                        ␣
↪                                       ->limit(1)
                                                                                        ␣
↪                                       ->execute($this->customizeUploadPath($vv));

                                                                        $tmp[$kk] =
↪$objFile->uuid;
                                                                }
```

WordPress

Altering Foreach Without Reference, in wp-admin/includes/misc.php:74.

$ids[$index] is &$rrid.

```
foreach($ids as $index => $rrid)
                {
                    if($rrid == $this->Id)
                    {
                        $ids[$index] = $_id;
                        $write = true;
                        break;
                    }
                }
```

17.2.145 Bail Out Early

OpenEMR

Bail Out Early, in interface/modules/zend_modules/module/Carecoordination/src/Carecoordination/Controller/EncounterccdadispatchC

This is a typical example of a function mostly controlled by one condition. It could be rewrite as 'if($validResult != 'existingpatient')' then return. The 'else' clause is not used anymore, and the whole block of code is now the main sequence of the method.

```
public function ccdaFetching($parameterArray = array())
    {
        $validResult = $this->getEncounterccdadispatchTable()->valid(
↪$parameterArray[0]);
        // validate credentials
        if ($validResult == 'existingpatient') {
```

(continues on next page)

(continued from previous page)

```
/// Long bloc of code
        } else {
            return '<?xml version=1.0 encoding=UTF-8?>
                    <!-- Edited by XMLSpy -->
                    <note>

                            <heading>Authetication Failure</heading>
                            <body></body>
                    </note>
                    ';
        }
```

opencfp

Bail Out Early, in chair/assign_auto_reviewers_weighted_topic_match.inc:105.

This long example illustrates two aspects : first, the shortcut to the end of the method may be the 'then' clause, not necessarily the 'else'. '!in_array($pid.'-'.$rid, $conflictAR)' leads to return, and the 'else' should be removed, while keeping its content. Secondly, we can see 3 conditions that all lead to a premature end to the method. After refactoring all of them, the method would end up with 1 level of indentation, instead of 3.

```
function oc_inConflict(&$conflictAR, $pid, $rid=null) {
    if ($rid == null) {
            $rid = $_SESSION[OCC_SESSION_VAR_NAME]['acreviewerid'];
    }
    if (!in_array($pid.'-'.$rid, $conflictAR)) {
            return false; // not in conflict
    } else {
            $tempr = ocsql_query("SELECT COUNT(*) AS `count` FROM `" . OCC_TABLE_
↪PAPERREVIEWER . "` WHERE `paperid`='" . safeSQLstr($pid) . "' AND `reviewerid`='" .␣
↪safeSQLstr($rid) . "'");
            if ((ocsql_num_rows($tempr) == 1)
                    && ($templ = ocsql_fetch_assoc($tempr))
                    && ($templ['count'] == 1)
            ) {
                    return false; // assigned as reviewer
            } else {
                    $tempr = ocsql_query("SELECT COUNT(*) AS `count` FROM `" . OCC_
↪TABLE_PAPERADVOCATE . "` WHERE `paperid`='" . safeSQLstr($pid) . "' AND␣
↪`advocateid`='" . safeSQLstr($rid) . "'");
                    if ((ocsql_num_rows($tempr) == 1)
                            && ($templ = ocsql_fetch_assoc($tempr))
                            && ($templ['count'] == 1)
                    ) {
                            return false; // assigned as advocate
                    }
            }
    }
    return true;
}
```

17.2.146 Use Basename Suffix

NextCloud

Use Basename Suffix, in lib/private/URLGenerator.php:176.

This code removes the 4 last letters from the images. It may be 'png', 'jpg' or 'txt'.

```
substr(basename($image), 0, -4)
```

Dolibarr

Use Basename Suffix, in htdocs/core/website.inc.php:42.

The extension '.tpl.php' is dropped from the file name, unless it appears somewhere else in the $websitepagefile variable.

```
str_replace(array('.tpl.php', 'page'), array('', ''), basename($websitepagefile))
```

17.2.147 Strict Comparison With Booleans

Phinx

Strict Comparison With Booleans, in src/Phinx/Db/Adapter/MysqlAdapter.php:1131.

ìsNull( )' always returns a boolean : it may be only be `true` or `false`. Until typehinted properties or return typehint are used, isNull() may return anything else.

```
$column->isNull( ) == false
```

Typo3

Strict Comparison With Booleans, in typo3/sysext/lowlevel/Classes/Command/FilesWithMultipleReferencesCommand.php:90.

When `dry-run` is not defined, the getOption() method actually returns a `null` value. So, comparing the result of getOption() to false is actually wrong : using a constant to prevent values to be inconsistent is recommended here.

```
$input->getOption('dry-run') != false
```

17.2.148 Buried Assignation

XOOPS

Buried Assignation, in htdocs/image.php:170.

Classic iffectation : the condition also collects the needed value to process the drawing. This is very common in PHP, and the Yoda condition, with its constant on the left, shows that extra steps were taken to strengthen that piece of code.

```
if (0 < ($radius = $radii[2] * $q)) { // left bottom
        imagearc($workingImage, $radius - 1, $workingHeight - $radius, $radius * 2,
↪$radius * 2, 90, 180, $alphaColor);
        imagefilltoborder($workingImage, 0, $workingHeight - 1, $alphaColor,
↪$alphaColor);
    }
```

Mautic

Buried Assignation, in app/bundles/CoreBundle/Controller/ThemeController.php:47.

The setting of the variable $cancelled is fairly hidden here, with its extra operator !. The operator is here for the condition, as $cancelled needs the 'cancellation' state, while the condition needs the contrary. Note also that isset() could be moved out of this condition, and made the result easier to read.

```
$form        = $this->get('form.factory')->create('theme_upload', [], ['action' =>
↪$action]);

        if ($this->request->getMethod() == 'POST') {
            if (isset($form) && !$cancelled = $this->isFormCancelled($form)) {
                if ($this->isFormValid($form)) {
                    $fileData = $form['file']->getData();
```

17.2.149 Cast To Boolean

MediaWiki

Cast To Boolean, in includes/page/WikiPage.php:2274.

$options['changed'] and $options['created'] are documented and used as boolean. Yet, SiteStatsUpdate may require integers, for correct storage in the database, hence the type casting. `(int)` `(bool)` may be an alternative here.

```
$edits = $options['changed'] ? 1 : 0;
            $pages = $options['created'] ? 1 : 0;


            DeferredUpdates::addUpdate( SiteStatsUpdate::factory(
                    [ 'edits' => $edits, 'articles' => $good, 'pages' => $pages ]
            ) );
```

Dolibarr

Cast To Boolean, in htdocs/societe/class/societe.class.php:2777.

Several cases are built on the same pattern there. Each of the expression may be replaced by a cast to `(bool)`.

```
case 3:
                                $ret=(!$conf->global->SOCIETE_IDPROF3_UNIQUE?false:true);
                                break;
```

17.2.150 Catch Overwrite Variable

PhpIPAM

Catch Overwrite Variable, in app/subnets/scan/subnet-scan-snmp-route.php:58.

$e is used both as 'local' variable : it is local to the catch clause, and it is a blind variable in a foreach(). There is little overlap between the two occurrences, but one reader may wonder why the caught exception is shown later on.

```
try {
        $res = $Snmp->get_query(get_routing_table);
        // remove those not in subnet
        if (sizeof($res)>0) {
           // save for debug
           $debug[$d->hostname][$q] = $res;

           // save result
           $found[$d->id][$q] = $res;
        }
    } catch (Exception $e) {
       // save for debug
       $debug[$d->hostname][$q] = $res;
       $errors[] = $e->getMessage();
    }

// lots of code
// on line 132
    // print errors
    if (isset($errors)) {
        print <hr>;
        foreach ($errors as $e) {
            print $Result->show (warning, $e, false, false, true);
        }
    }
```

SuiteCrm

Catch Overwrite Variable, in modules/Emails/EmailUIAjax.php:1082.

$e starts as an Email(), in the 'getMultipleMessagesFromSugar' case, while a few lines later, in 'refreshSugarFolders', $e is now an exception. Breaks are in place, so both occurrences are separated, yet, one may wonder why an email is a warning, or a mail is a warning.

```
// On line 900, $e is a Email
        case getMultipleMessagesFromSugar:
            $GLOBALS['log']->debug(********** EMAIL 2.0 - Asynchronous - at:␣
↪getMultipleMessagesFromSugar);
            if (isset($_REQUEST['uid']) && !empty($_REQUEST['uid'])) {
                $exIds = explode(,, $_REQUEST['uid']);
                $out = array();

                foreach ($exIds as $id) {
                    $e = new Email();
                    $e->retrieve($id);
                    $e->description_html = from_html($e->description_html);
                    $ie->email = $e;
                    $out[] = $ie->displayOneEmail($id, $_REQUEST['mbox']);
                }

                echo $json->encode($out);
            }

            break;
```

(continues on next page)

(continued from previous page)

```
// lots of code
// on line 1082
        case refreshSugarFolders:
            try {
                $GLOBALS['log']->debug(********** EMAIL 2.0 - Asynchronous - at:␣
↪refreshSugarFolders);
                $rootNode = new ExtNode('', '');
                $folderOpenState = $current_user->getPreference('folderOpenState',
↪'Emails');
                $folderOpenState = (empty($folderOpenState)) ?  : $folderOpenState;
                $ret = $email->et->folder->getUserFolders(
                    $rootNode,
                    sugar_unserialize($folderOpenState),
                    $current_user,
                    true
                );
                $out = $json->encode($ret);
                echo $out;
            } catch (SugarFolderEmptyException $e) {
                $GLOBALS['log']->warn($e);
                $out = $json->encode(array(
                    'message' => 'No folder selected warning message here...',
                ));
                echo $out;
            }
            break;
```

17.2.151 Check All Types

Zend-Config

Check All Types, in src/Writer/Ini.php:122.

$value must be an array or a string here.

```
foreach ($config as $key => $value) {
            $group = array_merge($parents, [$key]);

            if (is_array($value)) {
                $iniString .= $this->addBranch($value, $group);
            } else {
                $iniString .= implode($this->nestSeparator, $group)
                            .  ' = '
                            .  $this->prepareValue($value)
                            .  \n;
            }
        }
```

Vanilla

Check All Types, in library/core/class.form.php:2488.

When $this->_FormValues is not null, then it is an array or an object, as it may be used immediately with foreach(). A check with is_array() would be a stronger option here.

```
public function formDataSet() {
        if (is_null($this->_FormValues)) {
            $this->formValues();
        }

        $result = [[]];
        foreach ($this->_FormValues as $key => $value) {
```

17.2.152 Check JSON

Woocommerce

Check JSON, in includes/admin/helper/class-wc-helper-plugin-info.php:66.

In case the body is an empty string, this will be correctly decoded, but will yield an object with an empty-named property.

```
$results = json_decode( wp_remote_retrieve_body( $request ), true );
            if ( ! empty( $results ) ) {
                    $response = (object) $results;
            }

            return $response;
```

17.2.153 Common Alternatives

Dolibarr

Common Alternatives, in htdocs/admin/facture.php:531.

The opening an closing tag couldd be moved outside the if condition : they are compulsory in both cases.

```
// Active
                                if (in_array($name, $def))
                                {
                                    print '<td class="center">'."\n";
                                    print '<a href="'.$_SERVER["PHP_SELF"].'?
↪action=del&value='.$name.'">';
                                    print img_picto($langs->trans("Enabled"), 'switch_
↪on');
                                    print '</a>';
                                    print '</td>';
                                }
                                else
                                {
                                    print '<td class=center\>'."\n";
                                    print '<a href="'.$_SERVER["PHP_SELF"].'?
↪action=set&value='.$name.'&scan_dir='.$module->scandir.'&label='.urlencode($module->
↪name).'">'.img_picto($langs->trans("SetAsDefault"), 'switch_off').'</a>';
                                    print "</td>";
                                }
```

NextCloud

Common Alternatives, in apps/encryption/lib/KeyManager.php:436.

$shareKey = $this->getShareKey($path, $uid);* is common to all three alternatives. In fact, *$uid = $this->getPublicShareKeyId();* is not common, and that shoul de reviewed, as *$uid will be undefined.

```
if ($this->util->isMasterKeyEnabled()) {
                    $uid = $this->getMasterKeyId();
                    $shareKey = $this->getShareKey($path, $uid);
                    if ($publicAccess) {
                            $privateKey = $this->getSystemPrivateKey($uid);
                            $privateKey = $this->crypt->decryptPrivateKey($privateKey,
↪ $this->getMasterKeyPassword(), $uid);
                    } else {
                            // when logged in, the master key is already decrypted in␣
↪the session
                            $privateKey = $this->session->getPrivateKey();
                    }
            } else if ($publicAccess) {
                    // use public share key for public links
                    $uid = $this->getPublicShareKeyId();
                    $shareKey = $this->getShareKey($path, $uid);
                    $privateKey = $this->keyStorage->getSystemUserKey($this->
↪publicShareKeyId . '.privateKey', Encryption::ID);
                    $privateKey = $this->crypt->decryptPrivateKey($privateKey);
            } else {
                    $shareKey = $this->getShareKey($path, $uid);
                    $privateKey = $this->session->getPrivateKey();
            }
```

17.2.154 Continue Is For Loop

XOOPS

Continue Is For Loop, in htdocs/kernel/object.php:711.

break is used here for cases, unless the case includes a if/then structures, in which it becomes a continue. It really should be a break.

```
foreach ($this->vars as $k => $v) {
            $cleanv = $v['value'];
            if (!$v['changed']) {
            } else {
                $cleanv = is_string($cleanv) ? trim($cleanv) : $cleanv;
                switch ($v['data_type']) {
                    case XOBJ_DTYPE_TIMESTAMP:
                        $cleanv = !is_string($cleanv) && is_numeric($cleanv) ? date(_
↪DBTIMESTAMPSTRING, $cleanv) : date(_DBTIMESTAMPSTRING, strtotime($cleanv));
                        break;
                    case XOBJ_DTYPE_TIME:
                        $cleanv = !is_string($cleanv) && is_numeric($cleanv) ? date(_
↪DBTIMESTRING, $cleanv) : date(_DBTIMESTRING, strtotime($cleanv));
                        break;
                    case XOBJ_DTYPE_DATE:
                        $cleanv = !is_string($cleanv) && is_numeric($cleanv) ? date(_
↪DBDATESTRING, $cleanv) : date(_DBDATESTRING, strtotime($cleanv));
```

(continues on next page)

(continued from previous page)

```
                        break;
                    case XOBJ_DTYPE_TXTBOX:
                        if ($v['required'] && $cleanv != '0' && $cleanv == '') {
                            $this->setErrors(sprintf(_XOBJ_ERR_REQUIRED, $k));
                            continue 2;
                        }
                        if (isset($v['maxlength']) && strlen($cleanv) > (int)$v[
↪'maxlength']) {
                            $this->setErrors(sprintf(_XOBJ_ERR_SHORTERTHAN, $k, (int)
↪$v['maxlength']));
                            continue 2;
                        }
```

17.2.155 Could Be Else

SugarCrm

Could Be Else, in SugarCE-Full-6.5.26/modules/Emails/ListViewGroup.php:79.

The first condition makes different checks if 'query' is in $_REQUEST or not. The second only applies to $_REQUEST['query'], as there is no else. There is also no visible sign that the first condition may change $_REQUEST or not

```
if(!isset($_REQUEST['query'])){
    //_pp('loading: '.$currentModule.'Group');
    //_pp($current_user->user_preferences[$currentModule.'GroupQ']);
    $storeQuery->loadQuery($currentModule.'Group');
    $storeQuery->populateRequest();
} else {
    //_pp($current_user->user_preferences[$currentModule.'GroupQ']);
    //_pp('saving: '.$currentModule.'Group');
    $storeQuery->saveFromGet($currentModule.'Group');
}

if(isset($_REQUEST['query'])) {
    // we have a query
    if(isset($_REQUEST['email_type']))                                $email_type = $_
↪REQUEST['email_type'];
    if(isset($_REQUEST['assigned_to']))                               $assigned_to = $_
↪REQUEST['assigned_to'];
    if(isset($_REQUEST['status']))                                    $status = $_
↪REQUEST['status'];
    // More code
}
```

OpenEMR

Could Be Else, in library/log.inc:653.

Those two if structure may definitely merged into one single instruction.

```
$success = 1;
    $checksum = ;
    if ($outcome === false) {
```

(continues on next page)

(continued from previous page)

```
        $success = 0;
    }

    if ($outcome !== false) {
        // Should use the $statement rather than the processed
        // variables, which includes the binded stuff. If do
        // indeed need the binded values, then will need
        // to include this as a separate array.

        //error_log(STATEMENT: .$statement,0);
        //error_log(BINDS: .$processed_binds,0);
        $checksum = sql_checksum_of_modified_row($statement);
        //error_log(CHECKSUM: .$checksum,0);
    }
```

17.2.156 Could Be Static

Dolphin

Could Be Static, in inc/utils.inc.php:673.

Dolphin pro relies on HTMLPurifier to handle cleaning of values : it is used to prevent xss threat. In this method, oHtmlPurifier is first checked, and if needed, created. Since creation is long and costly, it is only created once. Once the object is created, it is stored as a global to be accessible at the next call of the method. In fact, oHtmlPurifier is never used outside this method, so it could be turned into a 'static' variable, and prevent other methods to modify it. This is a typical example of variable that could be static instead of global.

```
function clear_xss($val)
{
    // HTML Purifier plugin
    global $oHtmlPurifier;
    if (!isset($oHtmlPurifier) && !$GLOBALS['logged']['admin']) {

        require_once(BX_DIRECTORY_PATH_PLUGINS . 'htmlpurifier/HTMLPurifier.
↪standalone.php');

/..../

        $oHtmlPurifier = new HTMLPurifier($oConfig);
    }

    if (!$GLOBALS['logged']['admin']) {
        $val = $oHtmlPurifier->purify($val);
    }

    $oZ = new BxDolAlerts('system', 'clear_xss', 0, 0,
        array('oHtmlPurifier' => $oHtmlPurifier, 'return_data' => &$val));
    $oZ->alert();

    return $val;
}
```

Contao

Could Be Static, in system/helper/functions.php:184.

$arrScanCache is a typical cache variables. It is set as global for persistence between calls. If it contains an already stored answer, it is returned immediately. If it is not set yet, it is then filled with a value, and later reused. This global could be turned into static, and avoid pollution of global space.

```
function scan($strFolder, $blnUncached=false)
{
    global $arrScanCache;

    // Add a trailing slash
    if (substr($strFolder, -1, 1) != '/')
    {
            $strFolder .= '/';
    }

    // Load from cache
    if (!$blnUncached && isset($arrScanCache[$strFolder]))
    {
            return $arrScanCache[$strFolder];
    }
    $arrReturn = array();

    // Scan directory
    foreach (scandir($strFolder) as $strFile)
    {
            if ($strFile == '.' || $strFile == '..')
            {
                    continue;
            }

            $arrReturn[] = $strFile;
    }

    // Cache the result
    if (!$blnUncached)
    {
            $arrScanCache[$strFolder] = $arrReturn;
    }

    return $arrReturn;
}
```

17.2.157 Could Use array_fill_keys

ChurchCRM

Could Use array_fill_keys, in src/ManageEnvelopes.php:107.

There are two initialisations at the same time here : that should make two call to array_fill_keys().

```
foreach ($familyArray as $fam_ID => $fam_Data) {
        $envelopesByFamID[$fam_ID] = 0;
        $envelopesToWrite[$fam_ID] = 0;
    }
```

PhpIPAM

Could Use array_fill_keys, in functions/scripts/merge_databases.php:418.

Even when the initialization is mixed with other operations, it is a good idea to extract it from the loop and give it to array_fill_keys().

```
$arr_new = array();
                                foreach ($arr as $type=>$objects) {
                                        $arr_new[$type] = array();
                                        if(sizeof($objects)>0) {
                                                foreach($objects as $ok=>$object) {
                                                        $arr_new[$type][] = $highest_ids_
↪append[$type] + $object;
                                                }
                                        }
                                }
```

17.2.158 Could Use array_unique

Dolibarr

Could Use array_unique, in htdocs/includes/restler/framework/Luracast/Restler/Format/XmlFormat.php:250.

This loop has two distinct operations : the first collect keys and keep them unique. A combinaison of array_keys() and array_unique() would do that job, while saving the in_array() lookup, and the configuration check with 'static::$importSettingsFromXml'. The second operation is distinct, and could be done with array_map().

```
$attributes = $xml->attributes();
            foreach ($attributes as $key => $value) {
                if (static::$importSettingsFromXml
                    && !in_array($key, static::$attributeNames)
                ) {
                    static::$attributeNames[] = $key;
                }
                $r[$key] = static::setType((string)$value);
            }
```

OpenEMR

Could Use array_unique, in gacl/gacl_api.class.php:441:441.

This loop is quite complex : it collects $aro_value in $acl_array['aro'][$aro_section_value], but also creates the array in $acl_array['aro'][$aro_section_value], and report errors in the debug log. array_unique() could replace the collection, while the debug would have to be done somewhere else.

```
foreach ($aro_value_array as $aro_value) {
                                        if ( count($acl_array['aro'][$aro_section_value])␣
↪!= 0 ) {
                                                if (!in_array($aro_value, $acl_array['aro
↪'][$aro_section_value])) {
                                                        $this->debug_text("append_acl():␣
↪ARO Section Value: $aro_section_value ARO VALUE: $aro_value");
                                                        $acl_array['aro'][$aro_section_
↪value][] = $aro_value;
```

(continues on next page)

(continued from previous page)

```
                                                $update=1;
                                            } else {
                                                $this->debug_text("append_acl():␣
↪Duplicate ARO, ignoring... ");
                                            }
                                        } else { //Array is empty so add this aro value.
                                                $acl_array['aro'][$aro_section_value][] =
↪$aro_value;
                                                $update = 1;
                                        }
                                }
```

17.2.159 Could Use Compact

WordPress

Could Use Compact, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

17.2.160 Could Use __DIR__

Woocommerce

Could Use __DIR__, in includes/class-wc-api.php:162.

All the 120 occurrences use *dirname(__FILE__)*, and could be upgraded to __DIR__ if backward compatibility to PHP 5.2 is not critical.

```
private function rest_api_includes() {
            // Exception handler.
            include_once dirname( __FILE__ ) . '/api/class-wc-rest-exception.php';

            // Authentication.
            include_once dirname( __FILE__ ) . '/api/class-wc-rest-authentication.php
↪';
```

Piwigo

Could Use __DIR__, in include/random_compat/random.php:50.

dirname(__FILE__) is cached into $RandomCompatDIR, then reused three times. Using __DIR__ would save that detour.

```
$RandomCompatDIR = dirname(__FILE__);

    require_once $RandomCompatDIR.'/byte_safe_strings.php';
    require_once $RandomCompatDIR.'/cast_to_int.php';
    require_once $RandomCompatDIR.'/error_polyfill.php';
```

17.2.161 Could Use Short Assignation

ChurchCRM

Could Use Short Assignation, in src/ChurchCRM/utils/GeoUtils.php:74.

Sometimes, the variable is on the other side of the operator.

```
$distance = 0.6213712 * $distance;
```

Thelia

Could Use Short Assignation, in local/modules/Tinymce/Resources/js/tinymce/filemanager/include/utils.php:70.

/= is rare, but it definitely could be used here.

```
$size = $size / 1024;
```

17.2.162 Could Use str_repeat()

Zencart

Could Use str_repeat(), in includes/functions/functions_general.php:1234.

That's a 45 repeat of

```
if ( (!zen_browser_detect('MSIE')) && (zen_browser_detect('Mozilla/4')) ) {
      for ($i=0; $i<45; $i++) $pre .= ' ';
    }
```

17.2.163 Dangling Array References

Typo3

Dangling Array References, in typo3/sysext/impexp/Classes/ImportExport.php:322.

foreach() reads $lines into $r, and augment those lines. By the end, the $r variable is not unset. Yet, several lines later, in the same method but with different conditions, another loop reuse the variable $r. If is_array($this->dat['header']['pagetree'] and is_array($this->remainHeader['records']) are arrays at the same moment, then both loops are called, and they share the same reference. Values of the latter array will end up in the formar.

```
if (is_array($this->dat['header']['pagetree'])) {
    reset($this->dat['header']['pagetree']);
    $lines = [];
    $this->traversePageTree($this->dat['header']['pagetree'], $lines);

    $viewData['dat'] = $this->dat;
    $viewData['update'] = $this->update;
    $viewData['showDiff'] = $this->showDiff;
    if (!empty($lines)) {
        foreach ($lines as &$r) {
            $r['controls'] = $this->renderControls($r);
            $r['fileSize'] = GeneralUtility::formatSize($r['size']);
```

(continues on next page)

(continued from previous page)

```
            $r['message'] = ($r['msg'] && !$this->doesImport ? '<span class=text-
↪danger>' . htmlspecialchars($r['msg']) . '</span>' : '');
        }
        $viewData['pagetreeLines'] = $lines;
    } else {
        $viewData['pagetreeLines'] = [];
    }
}
// Print remaining records that were not contained inside the page tree:
if (is_array($this->remainHeader['records'])) {
    $lines = [];
    if (is_array($this->remainHeader['records']['pages'])) {
        $this->traversePageRecords($this->remainHeader['records']['pages'], $lines);
    }
    $this->traverseAllRecords($this->remainHeader['records'], $lines);
    if (!empty($lines)) {
        foreach ($lines as &$r) {
            $r['controls'] = $this->renderControls($r);
            $r['fileSize'] = GeneralUtility::formatSize($r['size']);
            $r['message'] = ($r['msg'] && !$this->doesImport ? '<span class=text-
↪danger>' . htmlspecialchars($r['msg']) . '</span>' : '');
        }
        $viewData['remainingRecords'] = $lines;
    }
}
```

SugarCrm

Dangling Array References, in SugarCE-Full-6.5.26/modules/Import/CsvAutoDetect.php:165.

There are two nested foreach here : they both have referenced blind variables. The second one uses $data, but never changes it. Yet, it is reused the next round in the first loop, leading to pollution from the first rows of $this->_parser->data into the lasts. This may happen even if $data is not modified explicitely : in fact, it will be modified the next call to foreach($row as ...), for each element in $row.

```
foreach ($this->_parser->data as &$row) {
    foreach ($row as &$data) {
        $len = strlen($data);
        // check if it begins and ends with single quotes
        // if it does, then it double quotes may not be the enclosure
        if ($len>=2 && $data[0] == '"' && $data[$len-1] == '"') {
            $beginEndWithSingle = true;
            break;
        }
    }
    if ($beginEndWithSingle) {
        break;
    }
    $depth++;
    if ($depth > $this->_max_depth) {
        break;
    }
}
```

17.2.164 __DIR__ Then Slash

Traq

__DIR__ Then Slash, in src/Kernel.php:60.

When executed in a path '/a/b/c', this code will require '/a../../vendor/autoload.php.

```
static::$loader = require __DIR__.'../../vendor/autoload.php';
```

17.2.165 Don't Loop On Yield

Dolibarr

Don't Loop On Yield, in htdocs/includes/sabre/sabre/dav/lib/DAV/Server.php:912.

Yield from is a straight replacement here.

```
if (($newDepth === self::DEPTH_INFINITY || $newDepth >= 1) && $childNode instanceof␣
↪ICollection) {
    foreach ($this->generatePathNodes($subPropFind) as $subItem) {
        yield $subItem;
    }
}
```

Tikiwiki

Don't Loop On Yield, in lib/goal/goallib.php:944.

The replacement with `yield from`` is not straigthforward here. Yield is only called when $user hasn't been ``$done` : this is a unicity check. So, the double loop may produce a fully merged array, that may be reduced further by array_unique(). The final array, then, can be used with yield from.

```
$done = [];

foreach ($goal['eligible'] as $groupName) {
    foreach ($userlib->get_group_users($groupName) as $user) {
            if (! isset($done[$user])) {
                    yield ['user' => $user, 'group' => null];
                    $done[$user] = true;
            }
    }
}
```

17.2.166 Dont Mix ++

Contao

Dont Mix ++, in core-bundle/src/Resources/contao/drivers/DC_Table.php:1272.

Incrementing and multiplying at the same time.

```
$this->Database->prepare("UPDATE " . $this->strTable . " SET sorting=? WHERE id=?")
               ->execute(($count++ * 128), $objNewSorting->id);
```

Typo3

Dont Mix ++, in typo3/sysext/backend/Classes/Controller/SiteConfigurationController.php:74.

The post-increment is not readable at first glance.

```
foreach ($row['rootline'] as &$record) {
                $record['margin'] = $i++ * 20;
            }
```

17.2.167 Echo With Concat

Phpdocumentor

Echo With Concat, in src/phpDocumentor/Bootstrap.php:76.

Simply replace the dot by a comma.

```
echo 'PROFILING ENABLED' . PHP_EOL
```

TeamPass

Echo With Concat, in includes/libraries/Authentication/Yubico/PEAR.php:162.

This is less obvious, but turning print to echo, and the double-quoted string to single quoted string will yield the same optimisation.

```
print "PEAR constructor called, class=$classname\n";
```

17.2.168 Else If Versus Elseif

TeamPass

Else If Versus Elseif, in items.php:819.

This code could be turned into a switch() structure.

```
if ($field[3] === 'text') {
                echo '
                        <input type=text id=edit_field_.$field[0]._.$elem[0].␣
↪class=edit_item_field input_text text ui-widget-content ui-corner-all size=40 data-
↪field-type=.$field[3]. data-field-masked=.$field[4]. data-field-is-mandatory=.
↪$field[5]. data-template-id=.$templateID.>';
            } else if ($field[3] === 'textarea') {
                echo '
                        <textarea id=edit_field_.$field[0]._.$elem[0]. class=edit_
↪item_field input_text text ui-widget-content ui-corner-all colums=40 rows=5 data-
↪field-type=.$field["3"]. data-field-masked=.$field[4]. data-field-is-mandatory=.
↪$field[5]. data-template-id=.$templateID.></textarea>';
            }
```

Phpdocumentor

Else If Versus Elseif, in src/phpDocumentor/Plugin/Core/Transformer/Writer/Xsl.php:112.

The first then block is long and complex. The else block, on the other hand, only contains a single if/then/else. Both conditions are distinct at first sight, so a if / elseif / then structure would be the best.

```
if ($transformation->getQuery() !== '') {
/** Long then block **/
        } else {
            if (substr($transformation->getArtifact(), 0, 1) == '$') {
                // not a file, it must become a variable!
                $variable_name = substr($transformation->getArtifact(), 1);
                $this->xsl_variables[$variable_name] = $proc->transformToXml(
↪$structure);
            } else {
                $relativeFileName = substr($artifact, strlen($transformation->
↪getTransformer()->getTarget()) + 1);
                $proc->setParameter('', 'root', str_repeat('../', substr_count(
↪$relativeFileName, '/')));

                $this->writeToFile($artifact, $proc, $structure);
            }
        }
```

17.2.169 Empty Blocks

Cleverstyle

Empty Blocks, in modules/Blogs/api/Controller.php:44.

Else is empty, but commented.

```
public static function posts_get ($Request) {
            $id = $Request->route_ids(0);
            if ($id) {
                    $post = Posts::instance()->get($id);
                    if (!$post) {
                            throw new ExitException(404);
                    }
                    return $post;
            } else {
                    // TODO: implement latest posts
            }
    }
```

PhpIPAM

Empty Blocks, in wp-admin/includes/misc.php:74.

The `then` block is empty and commented : yet, it may have been clearer to make the condition != and omitted the whole empty block.

```
/* checks */
if($_POST['action'] == delete) {
```

(continues on next page)

(continued from previous page)

```
    # no cecks
}
else {
    # remove spaces
    $_POST['name'] = trim($_POST['name']);

    # length > 4 and < 12
    if( (mb_strlen($_POST['name']) < 2) || (mb_strlen($_POST['name']) > 24) )       {
↪$errors[] = _('Name must be between 4 and 24 characters'); }
```

17.2.170 Empty Instructions

Zurmo

Empty Instructions, in app/protected/core/widgets/MentionInput.php:84.

There is no need for a semi-colon after a if/then structure.

```
public function run()
        {
            $id = $this->getId();
            $additionalSettingsJs = showAvatars: . var_export($this->showAvatars,␣
↪true) . ,;
            if ($this->classes)
            {
                $additionalSettingsJs .=  $this->classes . ',';
            };
            if ($this->templates)
            {
                $additionalSettingsJs .=  $this->templates;
            };
```

ThinkPHP

Empty Instructions, in ThinkPHP/Library/Vendor/Smarty/sysplugins/smarty_internal_configfileparser.php:83.

There is no need for a semi-colon after a class structure, unless it is an anonymous class.

```
class TPC_yyStackEntry
{
    public $stateno;       /* The state-number */
    public $major;         /* The major token value.  This is the code
                    ** number for the token at this stack level */
    public $minor; /* The user-supplied minor token value.  This
                    ** is the value of the token  */
};
```

17.2.171 Empty Try Catch

LiveZilla

Empty Try Catch, in livezilla/_lib/trdp/Zend/Mail/Protocol/Pop3.php:237.

This is an aptly commented empty try/catch : the emited exception is extra check for a Zend Mail Protocol Exception. Hopefully, the Zend_Mail_Protocol_Exception only covers a already-closed situation. Anyhow, this should be logged for later diagnostic.

```
public function logout()
    {
        if (!$this->_socket) {
            return;
        }

        try {
            $this->request('QUIT');
        } catch (Zend_Mail_Protocol_Exception $e) {
            // ignore error - we're closing the socket anyway
        }

        fclose($this->_socket);
        $this->_socket = null;
    }
```

Mautic

Empty Try Catch, in app/bundles/ReportBundle/Model/ExportHandler.php:66.

Removing a file : if the file is not 'deleted' by the method call, but raises an error, it is hidden. When file destruction is impossible because the file is already destroyed (or missing), this is well. If the file couldn't be destroyed because of missing writing privileges, hiding this error will have serious consequences.

```
/**
     * @param string $fileName
     */
    public function removeFile($fileName)
    {
        try {
            $path = $this->getPath($fileName);
            $this->filePathResolver->delete($path);
        } catch (FileIOException $e) {
        }
    }
```

17.2.172 Empty With Expression

HuMo-Gen

Empty With Expression, in fanchart.php:297.

The test on $pid may be directly done on $treeid[$sosa][0]. The distance between the assignation and the empty() makes it hard to spot.

```
$pid=$treeid[$sosa][0];
                    $birthyr=$treeid[$sosa][1];
                    $deathyr=$treeid[$sosa][4];
                    $fontpx=$fontsize;
                    if($sosa>=16 AND $fandeg==180) { $fontpx=$fontsize-1; }
```

(continues on next page)

(continued from previous page)

```
                    if($sosa>=32 AND $fandeg!=180) { $fontpx=$fontsize-1; }
                    if (!empty($pid)) {
```

17.2.173 error_reporting() With Integers

SugarCrm

error_reporting() With Integers, in modules/UpgradeWizard/silentUpgrade_step1.php:436.

This only displays E_ERROR, the highest level of error reporting. It should be checked, as it happens in the 'silentUpgrade' script.

```
ini_set('error_reporting', 1);
```

17.2.174 Eval() Usage

XOOPS

Eval() Usage, in htdocs/modules/system/class/block.php:266.

eval() execute code that was arbitrarily stored in $this, in one of the properties. Then, it is sent to output, but collected before reaching the browser, and put again in $content. May be the echo/ob_get_contents() could have been skipped.

```
ob_start();
                    echo eval($this->getVar('content', 'n'));
                    $content = ob_get_contents();
                    ob_end_clean();
```

Mautic

Eval() Usage, in app/bundles/InstallBundle/Configurator/Step/CheckStep.php:238.

create_function() is actually an eval() in disguise : replace it with a closure for code modernization

```
create_function('$cfgValue', 'return $cfgValue > 100;')
```

17.2.175 eval() Without Try

FuelCMS

eval() Without Try, in fuel/modules/fuel/controllers/Blocks.php:268.

The @ will prevent any error, while the try/catch allows the processing of certain types of error, namely the Fatal ones.

```
@eval($_name_var_eval)
```

ExpressionEngine

eval() Without Try, in system/ee/EllisLab/Addons/member/mod.member_memberlist.php:637.

$cond is build from values extracted from the $fields array. Although it is probably reasonably safe, a try/catch here will collect any unexpected situation cleaningly.

```
elseif (isset($fields[$val['3']]))
                                        {
                                                if (array_key_exists('m_field_id_'.
↪$fields[$val['3']], $row))
                                                {
                                                        $v = $row['m_field_id_'.$fields[
↪$val['3']]];

                                                        $lcond = str_replace($val['3'], "
↪$v", $lcond);
                                                        $cond = $lcond.' '.$rcond;
                                                        $cond = str_replace("\|", "|",
↪$cond);

                                                        eval("$result = ".$cond.";");
```

17.2.176 Exit() Usage

Traq

Exit() Usage, in src/Controllers/attachments.php:75.

This acts as a view. The final 'exit' is meant to ensure that no other piece of data is emitted, potentially polluting the view. This also prevent any code cleaning to happen.

```
/**
     * View attachment page
     *
     * @param integer $attachment_id
     */
    public function action_view($attachment_id)
    {
        // Don't try to load a view
        $this->render['view'] = false;

        header(Content-type: {$this->attachment->type});
        $content_type = explode('/', $this->attachment->type);

        // Check what type of file we're dealing with.
        if($content_type[0] == 'text' or $content_type[0] == 'image') {
            // If the mime-type is text, we can just display it
            // as plain text. I hate having to download files.
            if ($content_type[0] == 'text') {
                header(Content-type: text/plain);
            }
            header("Content-Disposition: filename=\"{$this->attachment->name}\"");
        }
        // Anything else should be downloaded
        else {
```

(continues on next page)

(continued from previous page)

```
            header("Content-Disposition: attachment; filename=\"{$this->attachment->
↪name}\"");
        }

        // Decode the contents and display it
        print(base64_decode($this->attachment->contents));
        exit;
    }
```

ThinkPHP

Exit() Usage, in ThinkPHP/Library/Vendor/EaseTemplate/template.core.php:60.

Here, exit is used as a rudimentary error management. When the version is not correctly provided via EaseTemplateVer, the application stop totally.

```
$this->version              = (trim($_GET['EaseTemplateVer']))?die('Ease Templae E3!
↪'):'';
```

17.2.177 Failed Substr Comparison

Zurmo

Failed Substr Comparison, in app/protected/modules/zurmo/modules/SecurableModule.php:117.

filterAuditEvent compares a six char string with 'AUDIT_EVENT_' which contains 10 chars. This method returns only FALSE. Although it is used only once, the whole block that calls this method is now dead code.

```
private static function filterAuditEvent($s)
        {
            return substr($s, 0, 6) == 'AUDIT_EVENT_';
        }
```

MediaWiki

Failed Substr Comparison, in includes/media/DjVu.php:263.

$metadata contains data that may be in different formats. When it is a pure XML file, it is 'Old style'. The comment helps understanding that this is not the modern way to go : the Old Style is actually never called, due to a failing condition.

```
private function getUnserializedMetadata( File $file ) {
            $metadata = $file->getMetadata();
            if ( substr( $metadata, 0, 3 ) === '<?xml' ) {
                    // Old style. Not serialized but instead just a raw string of XML.
                    return $metadata;
            }
```

17.2.178 Foreach Reference Is Not Modified

Dolibarr

Foreach Reference Is Not Modified, in htdocs/product/reassort.php:364.

$wh is an array, and is read for its index 'id', but it is not modified. The reference sign is too much.

```
if($nb_warehouse>1) {
    foreach($warehouses_list as &$wh) {

        print '<td class=right>';
        print empty($product->stock_warehouse[$wh['id']]->real) ? '0' : $product->
↪stock_warehouse[$wh['id']]->real;
        print '</td>';
    }
}
```

Vanilla

Foreach Reference Is Not Modified, in applications/vanilla/models/class.discussionmodel.php:944.

$discussion is also an object : it doesn't need any reference to be modified. And, it is not modified, but only read.

```
foreach ($result as $key => &$discussion) {
    if (isset($this->_AnnouncementIDs)) {
        if (in_array($discussion->DiscussionID, $this->_AnnouncementIDs)) {
            unset($result[$key]);
            $unset = true;
        }
    } elseif ($discussion->Announce && $discussion->Dismissed == 0) {
        // Unset discussions that are announced and not dismissed
        unset($result[$key]);
        $unset = true;
    }
}
```

17.2.179 Overwritten Source And Value

ChurchCRM

Overwritten Source And Value, in edusoho/vendor/symfony/symfony/src/Symfony/Component/VarDumper/Dumper/CliDumper.php:194

$str is actually processed as an array (string of characters), and it is also modified along the way.

```
foreach ($str as $str) {
                if ($i < $m) {
                    $str .= \n;
                }
                if (0 < $this->maxStringWidth && $this->maxStringWidth < $len = mb_
↪strlen($str, 'UTF-8')) {
                    $str = mb_substr($str, 0, $this->maxStringWidth, 'UTF-8');
                    $lineCut = $len - $this->maxStringWidth;
                }
                //.... More code
```

ExpressionEngine

Overwritten Source And Value, in system/ee/EllisLab/ExpressionEngine/Service/Theme/ThemeInstaller.php:595.

Looping over $filename.

```
foreach (directory_map($to_dir) as $directory => $filename)
                        {
                                if (is_string($directory))
                                {
                                        foreach ($filename as $filename)
                                        {
                                                unlink($to_dir.$directory.'/'.$filename);
                                        }

                                        @rmdir($to_dir.$directory);
                                }
                                else
                                {
                                        unlink($to_dir.$filename);
                                }
                        }
```

17.2.180 Function Subscripting, Old Style

OpenConf

Function Subscripting, Old Style, in openconf/include.php:1469.

Here, $advocateid may be directly read from ocsql_fetch_assoc(), although, checking for the existence of 'advocateid' before accessing it would make the code more robust

```
$advocateid = false;
    if (isset($GLOBALS['OC_configAR']['OC_paperAdvocates']) && $GLOBALS['OC_configAR
↪']['OC_paperAdvocates']) {
            $ar = ocsql_query(SELECT `advocateid` FROM ` . OCC_TABLE_PAPERADVOCATE .␣
↪` WHERE `paperid`=' . safeSQLstr($pid) . ') or err('Unable to retrieve advocate');
            if (ocsql_num_rows($ar) == 1) {
                    $al = ocsql_fetch_assoc($ar);
                    $advocateid = $al['advocateid'];
            }
    }
```

17.2.181 Identical Conditions

WordPress

Identical Conditions, in wp-admin/theme-editor.php:247.

The condition checks first if $has_templates or $theme->parent(), and one of the two is sufficient to be valid. Then, it checks again that $theme->parent() is activated with &&. This condition may be reduced by calling $theme->parent(), as $has_template is unused here.

```
<?php if ( ( $has_templates || $theme->parent() ) && $theme->parent() ) : ?>
```

Dolibarr

Identical Conditions, in htdocs/core/lib/files.lib.php:2052.

Better check twice that $modulepart is really 'apercusupplier_invoice'.

```
$modulepart == 'apercusupplier_invoice' || $modulepart == 'apercusupplier_invoice'
```

17.2.182 Identical On Both Sides

phpMyAdmin

Identical On Both Sides, in libraries/classes/DatabaseInterface.php:323.

This code looks like `($options & DatabaseInterface::QUERY_STORE) == DatabaseInterface::QUERY_STORE`, which would make sense. But PHP precedence is actually executing `$options & (DatabaseInterface::QUERY_STORE == DatabaseInterface::QUERY_STORE)`, which then doesn't depends on QUERY_STORE but only on $options.

```
if ($options & DatabaseInterface::QUERY_STORE == DatabaseInterface::QUERY_STORE) {
    $tmp = $this->_extension->realQuery('
        SHOW COUNT(*) WARNINGS', $this->_links[$link], DatabaseInterface::QUERY_STORE
    );
    $warnings = $this->fetchRow($tmp);
} else {
    $warnings = 0;
}
```

HuMo-Gen

Identical On Both Sides, in include/person_cls.php:73.

In that long logical expression, $personDb->pers_cal_date is tested twice

```
// *** Filter person's WITHOUT any date's ***
                    if ($user[group_filter_date]=='j'){
                            if ($personDb->pers_birth_date=='' AND $personDb->pers_
↪bapt_date==''
                            AND $personDb->pers_death_date=='' AND $personDb->pers_
↪buried_date==''
                            AND $personDb->pers_cal_date=='' AND $personDb->pers_cal_
↪date==''
                            ){
                                    $privacy_person='';
                            }
                    }
```

17.2.183 If With Same Conditions

phpMyAdmin

If With Same Conditions, in libraries/classes/Response.php:345.

The first test on $this->_isSuccess settles the situation with _JSON. Then, a second check is made. Both could be merged, also the second one is fairly long (not shown).

```
if ($this->_isSuccess) {
            $this->_JSON['success'] = true;
        } else {
            $this->_JSON['success'] = false;
            $this->_JSON['error']   = $this->_JSON['message'];
            unset($this->_JSON['message']);
        }

        if ($this->_isSuccess) {
```

Phpdocumentor

If With Same Conditions, in src/phpDocumentor/Transformer/Command/Project/TransformCommand.php:239.

$templates is extracted from $input. If it is empty, a second source is polled. Finally, if nothing has worked, a default value is used ('clean'). In this case, each attempt is an alternative solution to the previous failing call. The second test could be reported on $templatesFromConfig, and not $templates.

```
$templates = $input->getOption('template');
        if (!$templates) {
            /** @var Template[] $templatesFromConfig */
            $templatesFromConfig = $configurationHelper->getConfigValueFromPath(
↪'transformations/templates');
            foreach ($templatesFromConfig as $template) {
                $templates[] = $template->getName();
            }
        }

        if (!$templates) {
            $templates = array('clean');
        }
```

17.2.184 Inconsistent Concatenation

FuelCMS

Inconsistent Concatenation, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

17.2.185 Indices Are Int Or String

Zencart

Indices Are Int Or String, in includes/modules/payment/paypaldp.php:2523.

All those strings ends up as integers.

```
// Build Currency format table
    $curFormat = Array();
    $curFormat[036]=2;
    $curFormat[124]=2;
    $curFormat[203]=2;
    $curFormat[208]=2;
    $curFormat[348]=2;
    $curFormat[392]=0;
    $curFormat[554]=2;
    $curFormat[578]=2;
    $curFormat[702]=2;
    $curFormat[752]=2;
    $curFormat[756]=2;
    $curFormat[826]=2;
    $curFormat[840]=2;
    $curFormat[978]=2;
    $curFormat[985]=2;
```

Mautic

Indices Are Int Or String, in app/bundles/CoreBundle/Entity/CommonRepository.php:315.

$baseCols has 1 and 0 (respectively) for index.

```
foreach ($metadata->getAssociationMappings() as $field => $association) {
                    if (in_array($association['type'], [ClassMetadataInfo::ONE_TO_ONE,
↪ ClassMetadataInfo::MANY_TO_ONE])) {
                        $baseCols[true][$entityClass][]  = $association['joinColumns
↪'][0]['name'];
                        $baseCols[false][$entityClass][] = $field;
                    }
                }
```

17.2.186 Invalid Regex

SugarCrm

Invalid Regex, in SugarCE-Full-6.5.26/include/utils/file_utils.php:513.

This yields an error at execution time : ‘‘Compilation failed: invalid range in character class at offset 4 ‘‘.

```
preg_replace('/[^\w-._]+/i', '', $name)
```

17.2.187 Is Actually Zero

Dolibarr

Is Actually Zero, in htdocs/compta/ajaxpayment.php:99.

Here, the $amountToBreakDown is either $currentRemain or $result.

```
$amountToBreakdown = ($result - $currentRemain >= 0 ?

↪$currentRemain :                                                                    //␣
↪Remain can be fully paid

↪$currentRemain + ($result - $currentRemain));    // Remain can only partially be paid
```

SuiteCrm

Is Actually Zero, in modules/AOR_Charts/lib/pChart/class/pDraw.class.php:523.

$Xa may only amount to $iX2, though the expression looks weird.

```
if ( $X > $iX2 ) { $Xa = $X-($X-$iX2); $Ya = $iY1+($X-$iX2); } else { $Xa = $X; $Ya =
↪$iY1; }
```

17.2.188 list() May Omit Variables

OpenConf

list() May Omit Variables, in openconf/author/privacy.php:29.

The first variable in the list(), $none, isn't reused anywhere in the script. In fact, its name convey the meaning that is it useless, but is in the array nonetheless.

```
list($none, $OC_privacy_policy) = oc_getTemplate('privacy_policy');
```

FuelCMS

list() May Omit Variables, in wp-admin/includes/misc.php:74.

$a is never reused again. $b, on the other hand is. Not assigning any value to $a saves some memory, and avoid polluting the local variable space.

```
list($b, $a) = array(reset($params->me), key($params->me));
```

17.2.189 Logical Mistakes

Dolibarr

Logical Mistakes, in htdocs/core/lib/admin.lib.php:1165.

This expression is always true. When *$nbtabsql* is *$nbtablib*, the left part is true; When *$nbtabsql* is *$nbtabsqlsort*, the right part is true; When any other value is provided, both operands are true.

```
$nbtablib != $nbtabsql || $nbtabsql != $nbtabsqlsort
```

Cleverstyle

Logical Mistakes, in modules/HybridAuth/Hybrid/Providers/DigitalOcean.php:123.

This expression is always false. When *$data->account->email_verified* is *true*, the right part is false; When *$data->account->email_verified* is *$data->account->email*, the right part is false; The only viable solution is to have ' $data->account->email'true : this is may be the intend it, though it is not easy to understand.

```
TRUE == $data->account->email_verified and $data->account->email == $data->account->
↪email_verified
```

17.2.190 Lone Blocks

ThinkPHP

Lone Blocks, in ThinkPHP/Library/Vendor/Hprose/HproseReader.php:163.

There is no need for block in a case/default clause. PHP executes all command in order, until a break or the end of the switch. There is another occurrence of that situation in this code : it seems to be a coding convention, while only applied to a few switch statements.

```
for ($i = 0; $i < $len; ++$i) {
            switch (ord($this->stream->getc()) >> 4) {
                case 0:
                case 1:
                case 2:
                case 3:
                case 4:
                case 5:
                case 6:
                case 7: {
                    // 0xxx xxxx
                    $utf8len++;
                    break;
                }
                case 12:
                case 13: {
                    // 110x xxxx   10xx xxxx
                    $this->stream->skip(1);
                    $utf8len += 2;
                    break;
                }
```

Tine20

Lone Blocks, in tine20/Addressbook/Convert/Contact/VCard/Abstract.php:199.

A case of empty case, with empty blocks. This is useless code. Event the curly brackets with the final case are useless.

```
switch ( $property['TYPE'] ) {
                        case 'JPG' : {}
                        case 'jpg' : {}
                        case 'Jpg' : {}
                        case 'Jpeg' : {}
                        case 'jpeg' : {}
```

(continues on next page)

(continued from previous page)

```
                        case 'PNG' : {}
                        case 'png' : {}
                        case 'JPEG' : {
                            if (Tinebase_Core::isLogLevel(Zend_Log::DEBUG))
                                Tinebase_Core::getLogger()->warn(__METHOD__ . '::' . _
↪_LINE__ . ' Photo: passing on invalid ' . $property['TYPE'] . ' image as is (' .␣
↪strlen($property->getValue()) .')' );
                            $jpegphoto = $property->getValue();
                            break;
                        }
```

17.2.191 Long Arguments

Cleverstyle

Long Arguments, in core/drivers/DB/MySQLi.php:40.

This query is not complex, but its length tend to push the end out of the view in the IDE. It could be rewritten as a variable, on the previous line, with some formatting. The same formatting would help without the variable too, yet, mixing the SQL syntax with the PHP methodcall adds a layer of confusion.

```
$this->instance->query("SET SESSION sql_mode='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,
↪NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_
↪ENGINE_SUBSTITUTION'")
```

Contao

Long Arguments, in core-bundle/src/Resources/contao/widgets/CheckBoxWizard.php:145.

This one-liner includes 9 members and 6 variables : some are formatted by sprintf, some are directly concatenated in the string. Breaking this into two lines improves readbility and code review.

```
sprintf('<span><input type="checkbox" name="%s" id="opt_%s" class="tl_checkbox" value=
↪"%s"%s%s onfocus="Backend.getScrollOffset()"> %s<label for="opt_%s">%s</label></
↪span>', $this->strName . ($this->multiple ? '[]' : ''), $this->strId . '_' . $i, (
↪$this->multiple ? \StringUtil::specialchars($arrOption['value']) : 1), (((\is_array(
↪$this->varValue) && \in_array($arrOption['value'], $this->varValue)) || $this->
↪varValue == $arrOption['value']) ? ' checked="checked"' : ''), $this->
↪getAttributes( ), $strButtons, $this->strId . '_' . $i, $arrOption['label'])
```

17.2.192 Mismatched Ternary Alternatives

phpadsnew

Mismatched Ternary Alternatives, in phpAdsNew-2.0/admin/lib-misc-stats.inc.php:219.

This is an unusual way to apply a condition. $bgcolor is '#FFFFFF' by default, and if $i % 2, then $bcolor is '#F6F6F6';. A more readable ternary option would be '$bgcolor = = $i % 2 ? "#FFFFFF" : "#F6F6F6";', and make a matched alternative branches.

```
$bgcolor = #FFFFFF;
    $i % 2 ? 0 : $bgcolor = #F6F6F6;
```

OpenEMR

Mismatched Ternary Alternatives, in portal/messaging/messages.php:132.

IS_DASHBOARD is defined as a boolean or a string. Later, it is tested as a boolean, and displayed as a integer, which will be cast to string by echo. Lots of transtyping are happening here.

```
// In two distinct if/then branch
l:29) define('IS_DASHBOARD', false);
l:41) define('IS_DASHBOARD', $_SESSION['authUser']);

l:132) echo IS_DASHBOARD ? IS_DASHBOARD : 0;
?>
```

17.2.193 Missing Cases In Switch

Tikiwiki

Missing Cases In Switch, in lib/articles/artlib.php:1075.

This switch handles 3 cases, plus the default for all others. There are other switch structures which also handle the '' case. There may be a missing case here. In particular, projects/tikiwiki/code//article_image.php host another switch with the same case, plus another 'topic' case.

```
switch ($image_type) {
                        case 'article':
                                $image_cache_prefix = 'article';
                                break;
                        case 'submission':
                                $image_cache_prefix = 'article_submission';
                                break;
                        case 'preview':
                                $image_cache_prefix = 'article_preview';
                                break;
                        default:
                                return false;
                }
```

17.2.194 Mixed Concat And Interpolation

SuiteCrm

Mixed Concat And Interpolation, in modules/AOW_Actions/actions/actionSendEmail.php:89.

How long did it take to spot the hidden $checked variable in this long concatenation ? Using a consistent method of interpolation would help readability here.

```
"<input type='checkbox' id='aow_actions_param[" . $line . "][individual_email]' name=
↪'aow_actions_param[" . $line . "][individual_email]' value='1' $checked></td>"
```

Edusoho

Mixed Concat And Interpolation, in src/AppBundle/Controller/Admin/SiteSettingController.php:168.

Calling a method from a property of an object is possible inside a string, though it is rare. Setting the method outside the string make it more readable.

```
"{$this->container->getParameter('topxia.upload.public_url_path')}/" . $parsed['path']
```

17.2.195 Multiples Identical Case

SugarCrm

Multiples Identical Case, in modules/ModuleBuilder/MB/MBPackage.php:439.

It takes a while to find the double 'required' case, but the executed code is actually the same, so this is dead code at worst.

```
switch ($col) {
    case 'custom_module':
            $installdefs['custom_fields'][$name]['module'] = $res;
            break;
    case 'required':
            $installdefs['custom_fields'][$name]['require_option'] = $res;
            break;
    case 'vname':
            $installdefs['custom_fields'][$name]['label'] = $res;
            break;
    case 'required':
            $installdefs['custom_fields'][$name]['require_option'] = $res;
            break;
    case 'massupdate':
            $installdefs['custom_fields'][$name]['mass_update'] = $res;
            break;
    case 'comments':
            $installdefs['custom_fields'][$name]['comments'] = $res;
            break;
    case 'help':
            $installdefs['custom_fields'][$name]['help'] = $res;
            break;
    case 'len':
            $installdefs['custom_fields'][$name]['max_size'] = $res;
            break;
    default:
            $installdefs['custom_fields'][$name][$col] = $res;
}//switch
```

ExpressionEngine

Multiples Identical Case, in ExpressionEngine_Core2.9.2/system/expressionengine/controllers/cp/admin_content.php:577.

'deft_status' is doubled, with a fallthrough. This looks like some forgotten copy/paste.

```
switch ($key){
                                                    case 'cat_group':
                                                        //PHP code
                                                            break;
                                                    case 'status_group':
                                                    case 'field_group':
```

(continues on next page)

(continued from previous page)

```
                                                    //PHP code
                                                        break;
                                                case 'deft_status':
                                                case 'deft_status':
                                                    //PHP code
                                                        break;
                                                case 'search_excerpt':
                                                    //PHP code
                                                        break;
                                                case 'deft_category':
                                                    //PHP code
                                                        break;
                                                case 'blog_url':
                                                case 'comment_url':
                                                case 'search_results_url':
                                                case 'rss_url':
                                                    //PHP code
                                                        break;
                                                default :
                                                    //PHP code
                                                        break;
                                            }
```

17.2.196 Multiple Type Variable

Typo3

Multiple Type Variable, in typo3/sysext/backend/Classes/Form/Element/InputDateTimeElement.php:270.

$fullElement is an array most of the time, but finally ends up being a string. Since the array is not the final state, it may be interesting to make it a class, which collects the various variables, and export the final string. Such class would be usefull in several places in this repository.

```
$fullElement = [];
            $fullElement[] = '<div class=checkbox t3js-form-field-eval-null-
↪placeholder-checkbox>';
            $fullElement[] =     '<label for= . $nullControlNameEscaped . >';
            $fullElement[] =         '<input type=hidden name= .
↪$nullControlNameEscaped .  value= . $fallbackValue .  />';
            $fullElement[] =         '<input type=checkbox name= .
↪$nullControlNameEscaped .  id= . $nullControlNameEscaped .  value=1' . $checked .
↪$disabled . ' />';
            $fullElement[] =         $overrideLabel;
            $fullElement[] =     '</label>';
            $fullElement[] = '</div>';
            $fullElement[] = '<div class=t3js-formengine-placeholder-placeholder>';
            $fullElement[] =    '<div class=form-control-wrap style=max-width: .
↪$width . px>';
            $fullElement[] =        '<input type=text class=form-control␣
↪disabled=disabled value= . $shortenedPlaceholder .  />';
            $fullElement[] =    '</div>';
            $fullElement[] = '</div>';
            $fullElement[] = '<div class=t3js-formengine-placeholder-formfield>';
            $fullElement[] =    $expansionHtml;
```

(continues on next page)

(continued from previous page)

```
            $fullElement[] = '</div>';
            $fullElement = implode(LF, $fullElement);
```

Vanilla

Multiple Type Variable, in library/core/functions.general.php:1427.

Here, $value may be of different type. The if() structures merges all the incoming format into one standard type (int). This is actually the contrary of this analysis, and is a false positive.

```
if (is_array($value)) {
                        $value = count($value);
                    } elseif (stringEndsWith($field, 'UserID', true)) {
                        $value = 1;
                    }
```

17.2.197 Multiply By One

SugarCrm

Multiply By One, in SugarCE-Full-6.5.26/modules/Relationships/views/view.editfields.php:74.

Here, '$count % 1' is always true, after the first loop of the foreach. There is no need for % usage.

```
$count = 0;
        foreach($this->fields as $def)
        {
            if (!empty($def['relationship_field'])) {
                $label = !empty($def['vname']) ? $def['vname'] : $def['name'];
                echo <td> . translate($label, $this->module) . :</td>
                   . <td><input id='{$def['name']}' name='{$def['name']}'>  ;

                if ($count%1)
                    echo </tr><tr>;
                $count++;
            }
        }
        echo </tr></table></form>;
```

Edusoho

Multiply By One, in wp-admin/includes/misc.php:74.

1 is useless here, since 24 * 3600 is already an integer. And, of course, a day is not 24 * 3600... at least every day.

```
'yesterdayStart' => date('Y-m-d', strtotime(date('Y-m-d', time())) - 1 * 24 * 3600),
```

17.2.198 Named Regex

Phinx

Named Regex, in src/Phinx/Util/Util.php:127.

$matches[1] could be renamed by $matches['filename'], if the capturing subpattern was named 'filename'.

```
const MIGRATION_FILE_NAME_PATTERN = '/^\d+_([\w_]+).php$/i';
//.... More code with class definition
    public static function mapFileNameToClassName($fileName)
    {
        $matches = [];
        if (preg_match(static::MIGRATION_FILE_NAME_PATTERN, $fileName, $matches)) {
            $fileName = $matches[1];
        }

        return str_replace(' ', '', ucwords(str_replace('_', ' ', $fileName)));
    }
```

shopware

Named Regex, in engine/Library/Enlight/Components/Snippet/Resource.php:207.

$_match[3] is actually extracted two preg_match() before : by the time we read its usage, the first regex has been forgotten. A named subpattern would be useful here to remember what was captured.

```
if (!preg_match("!(.?)(name=)(.*?)(?=(\s|$))!", $_block_args, $_match) && empty($_
↪block_default)) {
                throw new SmartyException('"' . $_block_tag . '" missing name␣
↪attribute');
            }
            $_block_force = (bool) preg_match('#[\s]force#', $_block_args);
            $_block_json = (bool) preg_match('#[\s]json=["\']true["\']\W#', $_block_
↪args);
            $_block_name = !empty($_match[3]) ? trim($_match[3], '\'"') : $_block_
↪default;
```

17.2.199 Nested Ifthen

LiveZilla

Nested Ifthen, in livezilla/_lib/objects.global.inc.php:847.

The first condition is fairly complex, and could also return early. Then, the second nested if could be merged into one : this would reduce the number of nesting, but make the condition higher.

```
if(isset(Server::$Configuration->File["gl_url_detect"]) && !Server::$Configuration->
↪File["gl_url_detect"] && isset(Server::$Configuration->File["gl_url"]) && !
↪empty(Server::$Configuration->File["gl_url"]))
        {
            $url = Server::$Configuration->File["gl_url"];
        }
        else if(isset($_SERVER["HTTP_HOST"]) && !empty($_SERVER["HTTP_HOST"]))
        {
            $host = $_SERVER["HTTP_HOST"];
            $path = $_SERVER["PHP_SELF"];
```

(continues on next page)

(continued from previous page)

```
            if(!empty($path) && !Str::EndsWith(strtolower($path),strtolower($_file)) &
↪& strpos(strtolower($path),strtolower($_file)) !== false)
            {
                if(empty(Server::$Configuration->File["gl_kbmr"]))
                {
                    Logging::DebugLog(serialize($_SERVER));
                    exit("err 888383; can't read $_SERVER[\"HTTP_HOST\"] and $_
↪SERVER[\"PHP_SELF\"]");
                }
            }

            define("LIVEZILLA_DOMAIN",Communication::GetScheme() . $host);
            $url = LIVEZILLA_DOMAIN . str_replace($_file,"",htmlentities($path,ENT_
↪QUOTES,"UTF-8"));
        }
```

MediaWiki

Nested Ifthen, in includes/Linker.php:1493.

There are 5 level of nesting here, from the beginning of the method, down to the last condition. All work on local variables, as it is a static method. May be breaking this into smaller functions would help readability.

```
public static function normalizeSubpageLink( $contextTitle, $target, &$text ) {
            $ret = $target; # default return value is no change

            # Some namespaces don't allow subpages,
            # so only perform processing if subpages are allowed
            if (
                    $contextTitle && MediaWikiServices::getInstance()->
↪getNamespaceInfo()->
                    hasSubpages( $contextTitle->getNamespace() )
            ) {
                    $hash = strpos( $target, '#' );
                    if ( $hash !== false ) {
                            $suffix = substr( $target, $hash );
                            $target = substr( $target, 0, $hash );
                    } else {
                            $suffix = '';
                    }
                    # T9425
                    $target = trim( $target );
                    $contextPrefixedText = MediaWikiServices::getInstance()->
↪getTitleFormatter()->
                            getPrefixedText( $contextTitle );
                    # Look at the first character
                    if ( $target != '' && $target[0] === '/' ) {
                            # / at end means we don't want the slash to be shown
                            $m = [];
                            $trailingSlashes = preg_match_all( '%(/+)$%', $target, $m␣
↪);
                            if ( $trailingSlashes ) {
                                    $noslash = $target = substr( $target, 1, -strlen(
↪$m[0][0] ) );
```

(continues on next page)

(continued from previous page)

```
                                } else {
                                        $noslash = substr( $target, 1 );
                                }

                                $ret = $contextPrefixedText . '/' . trim( $noslash ) .
↪$suffix;
                                if ( $text === '' ) {
                                        $text = $target . $suffix;
                                } # this might be changed for ugliness reasons
                        } else {
                                # check for .. subpage backlinks
                                $dotdotcount = 0;
                                $nodotdot = $target;
                                while ( strncmp( $nodotdot, "../", 3 ) == 0 ) {
                                        ++$dotdotcount;
                                        $nodotdot = substr( $nodotdot, 3 );
                                }
                                if ( $dotdotcount > 0 ) {
                                        $exploded = explode( '/', $contextPrefixedText );
                                        if ( count( $exploded ) > $dotdotcount ) { # not␣
↪allowed to go below top level page
                                                $ret = implode( '/', array_slice(
↪$exploded, 0, -$dotdotcount ) );
                                                # / at the end means don't show full path
                                                if ( substr( $nodotdot, -1, 1 ) === '/' )
↪{
                                                        $nodotdot = rtrim( $nodotdot, '/'␣
↪);
                                                        if ( $text === '' ) {
                                                                $text = $nodotdot .
↪$suffix;
                                                        }
                                                }
                                                $nodotdot = trim( $nodotdot );
                                                if ( $nodotdot != '' ) {
                                                        $ret .= '/' . $nodotdot;
                                                }
                                                $ret .= $suffix;
                                        }
                                }
                        }
                }

                return $ret;
        }
```

17.2.200 Nested Ternary

SPIP

Nested Ternary, in ecrire/inc/utils.php:2648.

Interesting usage of both if/then, for the flow control, and ternary, for data process. Even on multiple lines, nested ternaries are quite hard to read.

```
// le script de l'espace prive
    // Mettre a "index.php" si DirectoryIndex ne le fait pas ou pb connexes:
    // les anciens IIS n'acceptent pas les POST sur ecrire/ (#419)
    // meme pb sur thttpd cf. http://forum.spip.net/fr_184153.html
    if (!defined('_SPIP_ECRIRE_SCRIPT')) {
            define('_SPIP_ECRIRE_SCRIPT', (empty($_SERVER['SERVER_SOFTWARE']) ? '' :
                    preg_match(',IIS|thttpd,', $_SERVER['SERVER_SOFTWARE']) ?
                            'index.php' : ''));
    }
```

Zencart

Nested Ternary, in app/library/zencart/ListingQueryAndOutput/src/formatters/TabularProduct.php:143.

No more than one level of nesting for this ternary call, yet it feels a lot more, thanks to the usage of arrayed properties, constants, and functioncalls.

```
$lc_text .= '<br />' . (zen_get_show_product_switch($listing->fields['products_id'],
↪'ALWAYS_FREE_SHIPPING_IMAGE_SWITCH') ? (zen_get_product_is_always_free_shipping(
↪$listing->fields['products_id']) ? TEXT_PRODUCT_FREE_SHIPPING_ICON . '<br />' : '')␣
↪: '');
```

17.2.201 Always Positive Comparison

Magento

Always Positive Comparison, in app/code/core/Mage/Dataflow/Model/Profile.php:85.

strlen(($actiosXML) will never be negative, and hence, is always false. This exception is never thrown.

```
if (strlen($actionsXML) < 0 &&
        @simplexml_load_string('<data>' . $actionsXML . '</data>', null, LIBXML_
↪NOERROR) === false) {
            Mage::throwException(Mage::helper('dataflow')->__("Actions XML is not␣
↪valid."));
        }
```

17.2.202 Next Month Trap

Contao

Next Month Trap, in system/modules/calendar/classes/Events.php:515.

This code is wrong on August 29,th 30th and 31rst : 6 months before is caculated here as February 31rst, so march 2. Of course, this depends on the leap years.

```
case 'past_180':
                            return array(strtotime('-6 months'), time(), $GLOBALS['TL_
↪LANG']['MSC']['cal_empty']);
```

Edusoho

Next Month Trap, in src/AppBundle/Controller/Admin/AnalysisController.php:1426.

The last month is wrong 8 times a year : on 31rst, and by the end of March.

```
'lastMonthStart' => date('Y-m-d', strtotime(date('Y-m', strtotime('-1 month')))),
            'lastMonthEnd' => date('Y-m-d', strtotime(date('Y-m', time())) - 24 *␣
↪3600),
            'lastThreeMonthsStart' => date('Y-m-d', strtotime(date('Y-m', strtotime('-
↪2 month')))),
```

17.2.203 No Choice

NextCloud

No Choice, in build/integration/features/bootstrap/FilesDropContext.php:71.

Token is checked, but processed in the same way each time. This actual check is done twice, in the same class, in the method droppingFileWith().

```
public function creatingFolderInDrop($folder) {
            $client = new Client();
            $options = [];
            if (count($this->lastShareData->data->element) > 0){
                    $token = $this->lastShareData->data[0]->token;
            } else {
                    $token = $this->lastShareData->data[0]->token;
            }
            $base = substr($this->baseUrl, 0, -4);
            $fullUrl = $base . '/public.php/webdav/' . $folder;

            $options['auth'] = [$token, ''];
```

Zencart

No Choice, in admin/includes/functions/html_output.php:179.

At least, it always choose the most secure way : use SSL.

```
if ($usessl) {
        $form .= zen_href_link($action, $parameters, 'NONSSL');
      } else {
        $form .= zen_href_link($action, $parameters, 'NONSSL');
      }
```

17.2.204 No Direct Usage

Edusoho

No Direct Usage, in edusoho/src/AppBundle/Controller/Admin/FinanceSettingController.php:107.

Glob() returns false, in case of error. It returns an empty array in case everything is fine, but nothing was found. In case of error, array_map() will stop the script.

```
array_map('unlink', glob($dir.'/MP_verify_*.txt'));
```

XOOPS

No Direct Usage, in htdocs/Frameworks/moduleclasses/moduleadmin/moduleadmin.php:585.

Although the file is readable, file() may return false in case of failure. On the other hand, implode doesn't accept boolean values.

```
$file = XOOPS_ROOT_PATH . /modules/{$module_dir}/docs/changelog.txt;
            if ( is_readable( $file ) ) {
                $ret .= implode( '<br>', file( $file ) ) . \n;
            }
```

17.2.205 No Empty Regex

Tikiwiki

No Empty Regex, in lib/sheet/excel/writer/worksheet.php:1925.

The initial 's' seems to be too much. May be a typo ?

```
// Strip URL type
        $url = preg_replace('s[^internal:]', '', $url);
```

17.2.206 No Hardcoded Hash

shopware

No Hardcoded Hash, in engine/Shopware/Models/Document/Data/OrderData.php:254.

This is actually a hashed hardcoded password. As the file explains, this is a demo order, for populating the database when in demo mode, so this is fine. We also learn that the password are securily sorted here. It may also be advised to avoid hardcoding this password, as any demo shop has the same user credential : it is the first to be tried when a demo installation is found.

```
'_userID' => '3',
    '_user' => new ArrayObject([
            'id' => '3',
            'password' => '$2y$10$GAGAC6.1kMRvN4RRcLrYleDx.EfWhHcW./cmoOQg11sjFUY73SO.
↪C',
            'encoder' => 'bcrypt',
            'email' => 'demo@shopware.com',
            'customernumber' => '20005',
```

SugarCrm

No Hardcoded Hash, in SugarCE-Full-6.5.26/include/Smarty/Smarty.class.php:460.

The MD5('Smarty') is hardcoded in the properties. This property is not used in the class, but in parts of the code, when a unique delimiter is needed.

```
/**
     * md5 checksum of the string 'Smarty'
     *
     * @var string
     */
    var $_smarty_md5            = 'f8d698aea36fcbead2b9d5359ffca76f';
```

17.2.207 No Hardcoded Ip

OpenEMR

No Hardcoded Ip, in wp-admin/includes/misc.php:74.

Although they are commented just above, the values provided here are suspicious.

```
// FTP parameters that you must customize.  If you are not sending
 // then set $FTP_SERVER to an empty string.
 //
 $FTP_SERVER = 192.168.0.30;
 $FTP_USER   = openemr;
 $FTP_PASS   = secret;
 $FTP_DIR    = ;
```

NextCloud

No Hardcoded Ip, in config/config.sample.php:1561.

Although they are documented as empty array, 3 values are provided as examples. They do not responds, at the time of writing, but they may.

```
/**
 * List of trusted proxy servers
 *
 * You may set this to an array containing a combination of
 * - IPv4 addresses, e.g. `192.168.2.123`
 * - IPv4 ranges in CIDR notation, e.g. `192.168.2.0/24`
 * - IPv6 addresses, e.g. `fd9e:21a7:a92c:2323::1`
 *
 * _(CIDR notation for IPv6 is currently work in progress and thus not
 * available as of yet)_
 *
 * When an incoming request's `REMOTE_ADDR` matches any of the IP addresses
 * specified here, it is assumed to be a proxy instead of a client. Thus, the
 * client IP will be read from the HTTP header specified in
 * `forwarded_for_headers` instead of from `REMOTE_ADDR`.
 *
 * So if you configure `trusted_proxies`, also consider setting
 * `forwarded_for_headers` which otherwise defaults to `HTTP_X_FORWARDED_FOR`
 * (the `X-Forwarded-For` header).
 *
 * Defaults to an empty array.
 */
'trusted_proxies' => array('203.0.113.45', '198.51.100.128', '192.168.2.0/24'),
```

17.2.208 No Hardcoded Path

Tine20

No Hardcoded Path, in tine20/Tinebase/DummyController.php:28.

When this script is not run on a Linux system, the file save will fail.

```
file_put_contents('/var/run/tine20/DummyController.txt', 'success ' . $n)
```

Thelia

No Hardcoded Path, in local/modules/Tinymce/Resources/js/tinymce/filemanager/include/php_image_magician.php:2317.

The *iptc.jpg* file is written. It looks like the file may be written next to the php_image_magician.php file, but this is deep in the source code and is unlikely. This means that the working directory has been set to some other place, though we don't read it immediately.

```
private function writeIPTC($dat, $value)
    {

            # LIMIT TO JPG

            $caption_block = $this->iptc_maketag(2, $dat, $value);
            $image_string = iptcembed($caption_block, $this->fileName);
            file_put_contents('iptc.jpg', $image_string);
    }
```

17.2.209 No Hardcoded Port

WordPress

No Hardcoded Port, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

17.2.210 No isset() With empty()

XOOPS

No isset() With empty(), in htdocs/class/tree.php:297.

Too much vlaidation

```
isset($this->tree[$key]['child']) && !empty($this->tree[$key]['child']);
```

17.2.211 No Need For Else

Thelia

No Need For Else, in core/lib/Thelia/Core/Template/Loop/Address.php:92.

After checking that $currentCustomer is null, the method returns. The block with Else may be removed and its code may be moved one level up.

```
if ($customer === 'current') {
            $currentCustomer = $this->securityContext->getCustomerUser();
            if ($currentCustomer === null) {
                return null;
            } else {
                $search->filterByCustomerId($currentCustomer->getId(),␣
↪Criteria::EQUAL);
            }
        } else {
            $search->filterByCustomerId($customer, Criteria::EQUAL);
        }
```

ThinkPHP

No Need For Else, in projects/thinkphp/code//ThinkPHP/Library/Org/Util/Rbac.class.php:187.

This code has both good and bad example. Good : no use of else, after $_SESSION[$accessGuid] check. Issue : else usage after usage of !isset($accessList[strtoupper($appName)][strtoupper(CONTROLLER_NAME)][strtoupper(ACTION_NAME)])

```
if (empty($_SESSION[C('ADMIN_AUTH_KEY')])) {
                if (C('USER_AUTH_TYPE') == 2) {
                    //
                    //
                    $accessList = self::getAccessList($_SESSION[C('USER_AUTH_KEY')]);
                } else {
                    //
                    if ($_SESSION[$accessGuid]) {
                        return true;
                    }
                    //
                    $accessList = $_SESSION['_ACCESS_LIST'];
                }
                //
                if (!isset($accessList[strtoupper($appName)][strtoupper(CONTROLLER_
↪NAME)][strtoupper(ACTION_NAME)])) {
                    $_SESSION[$accessGuid] = false;
                    return false;
                } else {
                    $_SESSION[$accessGuid] = true;
                }
```

17.2.212 No Parenthesis For Language Construct

Phpdocumentor

No Parenthesis For Language Construct, in src/Application/Renderer/Router/StandardRouter.php:55.

No need for parenthesis with require(). instanceof has a higher precedence than return anyway.

```
$this[] = new Rule(function ($node) { return ($node instanceof NamespaceDescriptor); }
↪, $namespaceGenerator);
```

phpMyAdmin

No Parenthesis For Language Construct, in db_datadict.php:170.

Not only echo() doesn't use any parenthesis, but this syntax gives the illusion that echo() only accepts one argument, while it actually accepts an arbitrary number of argument.

```
echo (($row['Null'] == 'NO') ? __('No') : __('Yes'))
```

17.2.213 @ Operator

Phinx

@ Operator, in src/Phinx/Util/Util.php:239.

fopen() may be tested for existence, readability before using it. Although, it actually emits some errors on Windows, with network volumes.

```
$isReadable = @\fopen($filePath, 'r') !== false;

        if (!$filePath || !$isReadable) {
            throw new \Exception(sprintf(Cannot open file %s \n, $filename));
        }
```

PhpIPAM

@ Operator, in functions/classes/class.Log.php:322.

Variable and index existence should always be tested with isset() : it is faster than using `@`.

```
$_SESSION['ipamusername']
```

17.2.214 Avoid Substr() One

ChurchCRM

Avoid Substr() One, in src/Login.php:141.

No need to call substr() to get only one char.

```
if (substr($LocationFromGet, 0, 1) == "/") {
    $LocationFromGet = substr($LocationFromGet, 1);
}
```

LiveZilla

Avoid Substr() One, in livezilla/_lib/objects.global.inc.php:2243.

No need to call substr() to get only one char.

```
$_hex = str_replace("#", "", $_hex);
            if(strlen($_hex) == 3) {
            $r = hexdec(substr($_hex,0,1).substr($_hex,0,1));
            $g = hexdec(substr($_hex,1,1).substr($_hex,1,1));
            $b = hexdec(substr($_hex,2,1).substr($_hex,2,1));
        } else {
            $r = hexdec(substr($_hex,0,2));
            $g = hexdec(substr($_hex,2,2));
            $b = hexdec(substr($_hex,4,2));
        }
        $rgb = array($r, $g, $b);
        return $rgb;
```

17.2.215 Not Not

Cleverstyle

Not Not, in modules/OAuth2/OAuth2.php:190.

This double-call returns `$results` as a boolean, preventing a spill of data to the calling method. The `(bool)` operator would be clearer here.

```
$result = $this->db_prime()->q(
                    [
                            DELETE FROM `[prefix]oauth2_clients`
                            WHERE `id` = '%s',
                            DELETE FROM `[prefix]oauth2_clients_grant_access`
                            WHERE `id`       = '%s',
                            DELETE FROM `[prefix]oauth2_clients_sessions`
                            WHERE `id`       = '%s'
                    ],
                    $id
            );
            unset($this->cache->{'/'});
            return !!$result;
```

Tine20

Not Not, in tine20/Calendar/Controller/MSEventFacade.php:392.

It seems that !! is almost superfluous, as a property called 'is_deleted' should already be a boolean.

```
foreach ($exceptions as $exception) {
                $exception->assertAttendee($this->getCalendarUser());
                $this->_prepareException($savedEvent, $exception);
                $this->_preserveMetaData($savedEvent, $exception, true);
                $this->_eventController->createRecurException($exception, !!
↪$exception->is_deleted);
            }
```

17.2.216 Objects Don't Need References

Zencart

Objects Don't Need References, in includes/library/illuminate/support/helpers.php:484.

No need for & operator when $class is only used for a method call.

```
/**
     * @param $class
     * @param $eventID
     * @param array $paramsArray
     */
    public function updateNotifyCheckoutflowFinishedManageSuccessOrderLinkEnd(&$class,
↪ $eventID, $paramsArray = array())
    {
        $class->getView()->getTplVarManager()->se('flag_show_order_link', false);
    }
```

XOOPS

Objects Don't Need References, in htdocs/class/theme_blocks.phps:221.

Here, $template is modified, when its properties are modified. When only the properties are modified, or read, then & is not necessary.

```
public function buildBlock($xobject, &$template)
    {
        // The lame type workaround will change
        // bid is added temporarily as workaround for specific block manipulation
        $block = array(
            'id'      => $xobject->getVar('bid'),
            'module'  => $xobject->getVar('dirname'),
            'title'   => $xobject->getVar('title'),
            // 'name'        => strtolower( preg_replace( '/[^0-9a-zA-Z_]/', '', str_
↪replace( ' ', '_', $xobject->getVar( 'name' ) ) ) ),
            'weight'  => $xobject->getVar('weight'),
            'lastmod' => $xobject->getVar('last_modified'));

        $bcachetime = (int)$xobject->getVar('bcachetime');
        if (empty($bcachetime)) {
            $template->caching = 0;
        } else {
            $template->caching        = 2;
            $template->cache_lifetime = $bcachetime;
        }
        $template->setCompileId($xobject->getVar('dirname', 'n'));
        $tplName = ($tplName = $xobject->getVar('template')) ? db:$tplName :
↪'db:system_block_dummy.tpl';
        $cacheid = $this->generateCacheId('blk_' . $xobject->getVar('bid'));
// more code to the end of the method
```

17.2.217 include_once() Usage

XOOPS

include_once() Usage, in /htdocs/xoops_lib/modules/protector/admin/center.php:5.

Loading() classes should be down with autoload(). autload() may be build in several distinct functions, using spl_autoload_register().

```
require_once dirname(__DIR__) . 'class/gtickets.php'
```

Tikiwiki

include_once() Usage, in tiki-mytiki_shared.php :140.

Turn the code from tiki-mytiki_shared.php into a function or a method, and call it when needed.

```
include_once('tiki-mytiki_shared.php');
```

17.2.218 One If Is Sufficient

Tikiwiki

One If Is Sufficient, in lib/wiki-plugins/wikiplugin_trade.php:152.

empty($params[‘inputtitle’]) should have priority over $params[‘wanted’] == ‘n’.

```
if ($params['wanted'] == 'n') {
            if (empty($params['inputtitle'])) {
                    $params['inputtitle'] = 'Payment of %0 %1 from user %2 to %3';
            }
    } else {
            if (empty($params['inputtitle'])) {
                    $params['inputtitle'] = 'Request payment of %0 %1 to user %2 from
↪%3';
            }
    }
```

17.2.219 Several Instructions On The Same Line

Piwigo

Several Instructions On The Same Line, in tools/triggers_list.php:993.

There are two instructions on the line with the if(). Note that the condition is not followed by a bracketed block. When reviewing, it really seems that echo ‘<br>’ and $f=0; are on the same block, but the second is indeed an unconditional expression. This is very difficult to spot.

```
foreach ($trigger['files'] as $file)
      {
        if (!$f) echo '<br>'; $f=0;
        echo preg_replace('#\((.+)\)#', '(<i>$1</i>)', $file);
      }
```

Tine20

Several Instructions On The Same Line, in tine20/Calendar/Controller/Event.php:1594.

Here, $_event->attendee is saved in a local variable, then the property is destroyed. Same for $_event->notes; Strangely, a few lines above, the properties are unset on their own line. Unsetting properties leads to surprise bugs, and hidding the unset after ; makes it harder to spot.

```
$futurePersistentExceptionEvents->setRecurId($_event->getId());
                unset($_event->recurid);
                unset($_event->base_event_id);
                foreach(array('attendee', 'notes', 'alarms') as $prop) {
                    if ($_event->{$prop} instanceof Tinebase_Record_RecordSet) {
                        $_event->{$prop}->setId(NULL);
                    }
                }
                $_event->exdate = $futureExdates;

                $attendees = $_event->attendee; unset($_event->attendee);
                $note = $_event->notes; unset($_event->notes);
                $persistentExceptionEvent = $this->create($_event, $_
↪checkBusyConflicts && $dtStartHasDiff);
```

17.2.220 Or Die

Tine20

Or Die, in scripts/addgrant.php:34.

Typical error handling, which also displays the MySQL error message, and leaks informations about the system. One may also note that mysql_connect is not supported anymore, and was replaced with mysqli and pdo : this may be a backward compatibile file.

```
$link = mysql_connect($host, $user, $pass) or die("No connection: " . mysql_error( ))
```

OpenConf

Or Die, in openconf/chair/export.inc:143.

or die() is also applied to many situations, where a blocking situation arise. Here, with the creation of a temporary file.

```
$coreFile = tempnam('/tmp/', 'ocexport') or die('could not generate Excel file (6)')
```

17.2.221 PHP7 Dirname

OpenConf

PHP7 Dirname, in include.php:61.

Since PHP 7.0, dirname(, 2); does the job.

```
$OC_basepath = dirname(dirname($_SERVER['PHP_SELF']));
```

MediaWiki

PHP7 Dirname, in includes/installer/Installer.php:1173.

Since PHP 7.0, dirname(, 2); does the job.

```
protected function envPrepPath() {
            global $IP;
            $IP = dirname( dirname( __DIR__ ) );
            $this->setVar( 'IP', $IP );
    }
```

17.2.222 Phpinfo

Dolphin

Phpinfo, in Dolphin-v.7.3.5/install/exec.php:4.

An actual phpinfo(), available during installation. Note that the phpinfo() is actually triggered by a hidden POST variable.

```
<?php

    if (!empty($_POST['phpinfo']))
        phpinfo();
    elseif (!empty($_POST['gdinfo']))
        echo '<pre>' . print_r(gd_info(), true) . '</pre>';

?>
<center>

    <form method=post>
        <input type=submit name=phpinfo value="PHP Info">
    </form>
    <form method=post>
        <input type=submit name=gdinfo value="GD Info">
    </form>

</center>
```

17.2.223 Possible Increment

Zurmo

Possible Increment, in app/protected/modules/workflows/utils/SavedWorkflowsUtil.php:196.

There are suspicious extra spaces around the +, that give the hint that there used to be something else, like a constant, there. From the name of the methods, it seems that this code was refactored from an addition to a simple method call.

```
$timeStamp =  + $workflow->getTimeTrigger()->resolveNewTimeStampForDuration(time());
```

MediaWiki

Possible Increment, in includes/filerepo/file/LocalFile.php:613.

That is a useless assignation, except for the transtyping to integer that PHP does silently. May be that should be a +=, or completely dropped.

```
$decoded[$field] = +$decoded[$field]
```

17.2.224 preg_replace With Option e

Edusoho

preg_replace With Option e, in vendor_user/uc_client/lib/uccode.class.php:32.

This call extract text between [code] tags, then process it with $this->codedisp() and nest it again in the original string. preg_replace_callback() is a drop-in replacement for this piece of code.

```
$message = preg_replace("/\s*\[code\](.+?)\[\/code\]\s*/ies", "$this->codedisp('\1')",
↪ $message);
```

17.2.225 Printf Number Of Arguments

PhpIPAM

Printf Number Of Arguments, in functions/classes/class.Common.php:1174.

16 will not be displayed.

```
sprintf('%032s', gmp_strval(gmp_init($ipv6long, 10), 16);
```

17.2.226 Property Variable Confusion

PhpIPAM

Property Variable Confusion, in functions/classes/class.Admin.php:16.

There is a property called '$users'. It is easy to mistake $this->users and $users. Also, it seems that $this->users may be used as a cache system, yet it is not employed here.

```
/**
     * (array of objects) to store users, user id is array index
     *
     * @var mixed
     * @access public
     */
    public $users;

////////////

    /**
     * Fetches all users that are in group
     *
     * @access public
     * @return array of user ids
     */
    public function group_fetch_users ($group_id) {
```

(continues on next page)

(continued from previous page)

```
            $out = array ();
            # get all users
            $users = $this->fetch_all_objects(users);
            # check if $gid in array
            if($users!==false) {
                    foreach($users as $u) {
                            $group_array = json_decode($u->groups, true);
                            $group_array = $this->groups_parse($group_array);

                            if(sizeof($group_array)>0) {
                                    foreach($group_array as $group) {
                                            if(in_array($group_id, $group)) {
                                                    $out[] = $u->id;
                                            }
                                    }
                            }
                    }
            }
            # return
            return isset($out) ? $out : array();
    }
```

17.2.227 Queries In Loops

TeamPass

Queries In Loops, in install/install.queries.php:551.

The value is SELECTed first in the database, and it is INSERTed if not. This may be done in one call in most databases.

```
foreach ($aMiscVal as $elem) {
    //Check if exists before inserting
    $tmp = mysqli_num_rows(
        mysqli_query(
            $dbTmp,
            SELECT * FROM `.$var['tbl_prefix'].misc`
            WHERE type='.$elem[0].' AND intitule='.$elem[1].'
        )
    );
    if (intval($tmp) === 0) {
        $queryRes = mysqli_query(
            $dbTmp,
            INSERT INTO `.$var['tbl_prefix'].misc`
            (`type`, `intitule`, `valeur`) VALUES
            ('.$elem[0].', '.$elem[1].', '.
            str_replace(', , $elem[2]).');
        ); // or die(mysqli_error($dbTmp))
    }

    // append new setting in config file
    $config_text .= '.$elem[1].' => '.str_replace(', , $elem[2]).',;
                        }
```

OpenEMR

Queries In Loops, in contrib/util/deidentification/deidentification.php:287.

The value is SELECTed first in the database, and it is INSERTed if not. This may be done in one call in most databases.

```
$query = select * from facility;
$result = mysqli_query($con, $query);
while ($row = mysqli_fetch_array($result)) {
    $string = update facility set

          `name`     = 'Facility_{$row['id']}',
          `phone`    = '(000) 000-0000'

        where `id` = {$row['id']};

    mysqli_query($con, $string) or print Error altering facility table \n;
    $string = '';
}
```

17.2.228 Repeated print()

Edusoho

Repeated print(), in app/check.php:71.

All echo may be merged into one : do this by turning the ; and . into ‘,’, and removing the superfluous echo. Also, echo_style may be turned into a non-display function, returning the build style, rather than echoing it to the output.

```
echo PHP_EOL;
echo_style('title', 'Note');
echo '  The command console could use a different php.ini file'.PHP_EOL;
echo_style('title', '~~~~');
echo '  than the one used with your web server. To be on the'.PHP_EOL;
echo '      safe side, please check the requirements from your web'.PHP_EOL;
echo '      server using the ';
echo_style('yellow', 'web/config.php');
echo ' script.'.PHP_EOL;
echo PHP_EOL;
```

HuMo-Gen

Repeated print(), in menu.php:71.

Simply calling print once is better than three times. Here too, echo usage would reduce the amount of memory allocation due to concatenation prior display.

```
print '<input type=text name=quicksearch value=.$quicksearch. size=10 '.$pattern.'␣
↪title=.__(Minimum:).$min_chars.__(characters).>';
                    print ' <input type=submit value=.__(Search).>';
            print </form>;
```

17.2.229 Repeated Regex

Vanilla

Repeated Regex, in library/core/class.pluginmanager.php:1200.

This regex is actually repeated 4 times across the Vanilla database, including this variation : '#^(https?:)?//#i'.

```
'`^https?://`'
```

Tikiwiki

Repeated Regex, in tiki-login.php:369.

This regex is use twice, identically, in the same file, with a few line of distance. It may be federated at the file level.

```
preg_match('/(tiki-register|tiki-login_validate|tiki-login_scr)\.php/', $url)
```

17.2.230 Return True False

Mautic

Return True False, in app/bundles/LeadBundle/Model/ListModel.php:125.

$isNew could be a typecast.

```
$isNew = ($entity->getId()) ? false : true;
```

FuelCMS

Return True False, in fuel/modules/fuel/helpers/validator_helper.php:254.

If/then is a lot of code to produce a boolean.

```
function length_min($str, $limit = 1)
    {
            if (strlen(strval($str)) < $limit)
            {
                    return FALSE;
            }
            else
            {
                    return TRUE;
            }
    }
```

17.2.231 Same Conditions In Condition

TeamPass

Same Conditions In Condition, in sources/identify.php:1096.

$result == 1 is use once in the main if/then, then again the second if/then/elseif structure. Both are incompatible, since, in the else, *$result* will be different from 1.

```
if ($result == 1) {
                $return = "";
                $logError = "";
                $proceedIdentification = true;
                $userPasswordVerified = false;
                unset($_SESSION['hedgeId']);
                unset($_SESSION['flickercode']);
            } else {
                if ($result < -10) {
                    $logError = "ERROR: ".$result;
                } elseif ($result == -4) {
                    $logError = "Wrong response code, no more tries left.";
                } elseif ($result == -3) {
                    $logError = "Wrong response code, try to reenter.";
                } elseif ($result == -2) {
                    $logError = "Timeout. The response code is not valid anymore.";
                } elseif ($result == -1) {
                    $logError = "Security Error. Did you try to verify the response␣
↪from a different computer?";
                } elseif ($result == 1) {
                    $logError = "Authentication successful, response code correct.
                          <br /><br />Authentification Method for SecureBrowser␣
↪updated!";
                    // Add necessary code here for accessing your Business Application
                }
                $return = "agses_error";
                echo '[{"value" : "'.$return.'", "user_admin":"',
                isset($_SESSION['user_admin']) ? $_SESSION['user_admin'] : "",
                '", "initial_url" : "'.@$_SESSION['initial_url'].'",
                "error" : "'.$logError.'"}]';

                exit();
            }
```

Typo3

Same Conditions In Condition, in typo3/sysext/recordlist/Classes/RecordList/DatabaseRecordList.php:1696.

$table == 'pages is caught initially, and if it fails, it is tested again in the final else. This won't happen.

```
} elseif ($table === 'pages') {
                                $parameters = ['id' => $this->id, 'pagesOnly' => 1,
↪'returnUrl' => GeneralUtility::getIndpEnv('REQUEST_URI')];
                                $href = (string)$uriBuilder->buildUriFromRoute('db_new
↪', $parameters);
                                $icon = '<a class="btn btn-default" href="' .␣
↪htmlspecialchars($href) . '" title="' . htmlspecialchars($lang->getLL('new')) . '">'
                                    . $spriteIcon->render() . '</a>';
                            } else {
                                $params = '&edit[' . $table . '][' . $this->id .
↪']=new';
                                if ($table === 'pages') {
                                    $params .= '&overrideVals[pages][doktype]=' .␣
↪(int)$this->pageRow['doktype'];
                                }
                                $icon = '<a class="btn btn-default" href="#" onclick="
↪' . htmlspecialchars(BackendUtility::editOnClick($params, '', -1))
```

(continues on next page)

(continued from previous page)

```
                                        . '" title="' . htmlspecialchars($lang->getLL('new
↪')) . '">' . $spriteIcon->render() . '</a>';
                                }
```

17.2.232 Should Chain Exception

Magento

Should Chain Exception, in lib/Mage/Backup/Filesystem/Rollback/Ftp.php:81.

Instead of using the exception message as an argument, chaining the exception would send the whole exception, including the message, and other interesting information like file and line.

```
protected function _initFtpClient()
    {
        try {
            $this->_ftpClient = new Mage_System_Ftp();
            $this->_ftpClient->connect($this->_snapshot->getFtpConnectString());
        } catch (Exception $e) {
            throw new Mage_Backup_Exception_FtpConnectionFailed($e->getMessage());
        }
    }
```

Tine20

Should Chain Exception, in tine20/Setup/Controller.php:81.

Here, the new exception gets an hardcoded message. More details about the reasons are already available in the $e exception, but they are not logged, not chained for later processing.

```
try {
            $dirIterator = new DirectoryIterator($this->_baseDir);
        } catch (Exception $e) {
            Setup_Core::getLogger()->warn(__METHOD__ . '::' . __LINE__ . ' Could not␣
↪open base dir: ' . $this->_baseDir);
            throw new Tinebase_Exception_AccessDenied('Could not open Tine 2.0 root␣
↪directory.');
        }
```

17.2.233 Should Make Ternary

ChurchCRM

Should Make Ternary, in src/CartToFamily.php:57.

$sState could be the receiving part of a ternary operator.

```
if ($sCountry == 'United States' || $sCountry == 'Canada') {
            $sState = InputUtils::LegacyFilterInput($_POST['State']);
        } else {
            $sState = InputUtils::LegacyFilterInput($_POST['StateTextbox']);
        }
```

17.2.234 Preprocessable

phpadsnew

Preprocessable, in phpAdsNew-2.0/adview.php:302.

Each call to chr() may be done before. First, chr() may be replace with the hexadecimal sequence "0x3B"; Secondly, 0x3b is a rather long replacement for a simple semi-colon. The whole pragraph could be stored in a separate file, for easier modifications.

```
echo chr(0x47).chr(0x49).chr(0x46).chr(0x38).chr(0x39).chr(0x61).chr(0x01).chr(0x00).
                chr(0x01).chr(0x00).chr(0x80).chr(0x00).chr(0x00).chr(0x04).
↪chr(0x02).chr(0x04).
                     chr(0x00).chr(0x00).chr(0x00).chr(0x21).chr(0xF9).chr(0x04).
↪chr(0x01).chr(0x00).
                chr(0x00).chr(0x00).chr(0x00).chr(0x2C).chr(0x00).chr(0x00).
↪chr(0x00).chr(0x00).
                chr(0x01).chr(0x00).chr(0x01).chr(0x00).chr(0x00).chr(0x02).
↪chr(0x02).chr(0x44).
                chr(0x01).chr(0x00).chr(0x3B);
```

17.2.235 Should Use Foreach

ExpressionEngine

Should Use Foreach, in system/ee/EllisLab/ExpressionEngine/Service/Model/Query/Builder.php:241.

This code could turn the string into an array, with the explode() function, and use foreach(), instead of calculating the length() initially, and then building the loop.

```
$length = strlen($str);
            $words = array();

            $word = '';
            $quote = '';
            $quoted = FALSE;

            for ($i = 0; $i < $length; $i++)
            {
                    $char = $str[$i];

                    if (($quoted == FALSE && $char == ' ') || ($quoted == TRUE &&
↪$char == $quote))
                    {
                            if (strlen($word) > 2)
                            {
                                    $words[] = $word;
                            }

                            $quoted = FALSE;
                            $quote = '';
                            $word = '';

                            continue;
                    }
```

(continues on next page)

(continued from previous page)

```
                    if ($quoted == FALSE && ($char == ' || $char == ") && ($word === '
↪' || $word == '-'))
                    {
                            $quoted = TRUE;
                            $quote = $char;
                            continue;
                    }

                    $word .= $char;
            }
```

Woocommerce

Should Use Foreach, in includes/libraries/class-wc-eval-math.php:84.

This loops reviews the 'stack' and updates its elements. The same loop may leverage foreach and references for more efficient code.

```
$stack_size = count( $stack );
                    for ( $i = 0; $i < $stack_size; $i++ ) { // freeze the␣
↪state of the non-argument variables
                            $token = $stack[ $i ];
                            if ( preg_match( '/^[a-z]\w*$/', $token ) and !␣
↪in_array( $token, $args ) ) {
                                    if ( array_key_exists( $token, self::$v )␣
↪) {
                                            $stack[ $i ] = self::$v[ $token ];
                                    } else {
                                            return self::trigger( "undefined␣
↪variable '$token' in function definition" );
                                    }
                            }
                    }
```

17.2.236 Should Use Math

OpenEMR

Should Use Math, in controllers/C_Prescription.class.php:638.

$pdf->ez['leftMargin'] is now 0.

```
function multiprint_body(& $pdf, $p)
    {
        $pdf->ez['leftMargin'] += $pdf->ez['leftMargin'];
        $pdf->ez['rightMargin'] += $pdf->ez['rightMargin'];
        $d = $this->get_prescription_body_text($p);
        if ($pdf->ezText($d, 10, array(), 1)) {
            $pdf->ez['leftMargin'] -= $pdf->ez['leftMargin'];
            $pdf->ez['rightMargin'] -= $pdf->ez['rightMargin'];
            $this->multiprint_footer($pdf);
            $pdf->ezNewPage();
            $this->multiprint_header($pdf, $p);
```

17.2.237 Should Use Operator

Zencart

Should Use Operator, in includes/modules/payment/paypal/paypal_curl.php:378.

Here, $options is merged with $values if it is an array. If it is not an array, it is probably a null value, and may be ignored. Adding a 'array' typehint will strengthen the code an catch situations where TransactionSearch() is called with a string, leading to clearer code.

```
function TransactionSearch($startdate, $txnID = '', $email = '', $options) {
    // several lines of code, no mention of $options
      if (is_array($options)) $values = array_merge($values, $options);
    }
    return $this->_request($values, 'TransactionSearch');
  }
```

SugarCrm

Should Use Operator, in include/utils.php:2093:464.

$override should an an array : if not, it is actually set by default to empty array. Here, a typehint with a default value of 'array()' would offset the parameter validation to the calling method.

```
function sugar_config_union( $default, $override ){
    // a little different then array_merge and array_merge_recursive.  we want
    // the second array to override the first array if the same value exists,
    // otherwise merge the unique keys.  it handles arrays of arrays recursively
    // might be suitable for a generic array_union
    if( !is_array( $override ) ){
            $override = array();
    }
    foreach( $default as $key => $value ){
            if( !array_key_exists($key, $override) ){
                    $override[$key] = $value;
            }
            else if( is_array( $key ) ){
                    $override[$key] = sugar_config_union( $value, $override[$key] );
            }
    }
    return( $override );
}
```

17.2.238 Simplify Regex

Zurmo

Simplify Regex, in app/protected/core/components/Browser.php:73.

Here, strpos() or stripos() is a valid replacement.

```
preg_match('/opera/', $userAgent)
```

OpenConf

Simplify Regex, in openconf/include.php:964.

%e is not a special char for PCRE regex, although it look like it. It is a special char for date() or printf(). This preg_replace() may be upgraded to str_replace()

```
$conv = iconv($cp, 'utf-8', strftime(preg_replace("/\%e/", '%#d', $format), $time));
```

17.2.239 Strpos()-like Comparison

Piwigo

Strpos()-like Comparison, in admin/include/functions.php:2585.

preg_match may return 0 if not found, and null if the $pattern is erroneous. While hardcoded regex may be checked at compile time, dynamically built regex may fail at execution time. This is particularly important here, since the function may be called with incoming data for maintenance : 'clear_derivative_cache($_GET['type']);' is in the /admin/maintenance.php.

```
function clear_derivative_cache_rec($path, $pattern)
{
  $rmdir = true;
  $rm_index = false;

  if ($contents = opendir($path))
  {
    while (($node = readdir($contents)) !== false)
    {
      if ($node == '.' or $node == '..')
        continue;
      if (is_dir($path.'/'.$node))
      {
        $rmdir &= clear_derivative_cache_rec($path.'/'.$node, $pattern);
      }
      else
      {
        if (preg_match($pattern, $node))
```

Thelia

Strpos()-like Comparison, in core/lib/Thelia/Controller/Admin/FileController.php:198.

preg_match is used here to identify files with a forbidden extension. The actual list of extension is provided to the method via the parameter $extBlackList, which is an array. In case of mis-configuration by the user of this array, preg_match may fail : for example, when regex special characters are provided. At that point, the whole filter becomes invalid, and can't distinguish good files (returning false) and other files (returning NULL). It is safe to use === false in this situation.

```
if (!empty($extBlackList)) {
            $regex = "#^(.+)\.(".implode("|", $extBlackList).")$#i";

            if (preg_match($regex, $realFileName)) {
                $message = $this->getTranslator()
                    ->trans(
```

(continues on next page)

(continued from previous page)

```
                        'Files with the following extension are not allowed:
↪%extension, please do an archive of the file if you want to upload it',
                        [
                            '%extension' => $fileBeingUploaded->
↪getClientOriginalExtension(),
                        ]
                    );
            }
        }
```

17.2.240 Drop Substr Last Arg

SuiteCrm

Drop Substr Last Arg, in modules/UpgradeWizard/uw_utils.php:2422.

substr() is even trying to go beyond the end of the string.

```
substr($relativeFile, 1, strlen($relativeFile))
```

Tine20

Drop Substr Last Arg, in tine20/Calendar/Frontend/Cli.php:95.

Omitting the last character would yield the same result.

```
substr($opt, 18, strlen($opt))
```

17.2.241 Suspicious Comparison

PhpIPAM

Suspicious Comparison, in app/tools/vrf/index.php:110.

if $subnet[‘description’] is a string, the comparison with 0 turn it into a boolean. false’s length is 0, and true length is 1. PHP saves the day.

```
$subnet['description'] = strlen($subnet['description']==0) ? "/" : $subnet[
↪'description'];
```

ExpressionEngine

Suspicious Comparison, in ExpressionEngine_Core2.9.2/system/expressionengine/libraries/simplepie/SimplePie/Misc.php:1925.

If trim($attribs[‘’][‘mode’]) === ‘base64’, then it is set to lowercase (although it is already), and added to the && logical test. If it is ‘BASE64’, this fails.

```
if (isset($attribs['']['mode']) && strtolower(trim($attribs['']['mode']) === 'base64
↪'))
```

17.2.242 Switch To Switch

Thelia

Switch To Switch, in core/lib/Thelia/Controller/Admin/TranslationsController.php:100.

The two first comparison may be turned into a case, and the last one could be default, or default with a check on empty().

```
if($modulePart == 'core') { /**/ } elseif($modulePart == 'admin-includes') { /**/ }␣
↪elseif(!empty($modulePart)) { /**/ }
```

XOOPS

Switch To Switch, in htdocs/search.php:74.

Here, converting this structure to switch requires to drop the === usage. Also, no default usage here.

```
if($action === 'results') { /**/ } elseif($action === 'showall') { /**/ } elseif(
↪$action === 'showallbyuser') { /**/ }
```

17.2.243 Switch Without Default

Zencart

Switch Without Default, in admin/tax_rates.php:15.

The 'action' is collected from $_GET and then, compared with various strings to handle the different actions to be taken. The default behavior is implicit here : if no 'action', display the initial form for taxes to be changed. This has to be understood as a general philosophy of ZenCart project, or by reading the rest of the HTML code. Adding a 'default' case here would help understand what happens in case 'action' is absent or unrecognized.

```
$action = (isset($_GET['action']) ? $_GET['action'] : '');

  if (zen_not_null($action)) {
    switch ($action) {
      case 'insert':
        // PHP code
        break;
      case 'save':
        // PHP code
        break;
      case 'deleteconfirm':
        // PHP code
        break;
    }
  }
?> .... HTML code
```

Traq

Switch Without Default, in src/Helpers/Ticketlist.php:311.

The default case is actually processed after the switch, by the next if/then structure. The structure deals with the customFields, while the else deals with any unknown situations. This if/then could be wrapped in the 'default' case of switch, for consistent processing. The if/then condition would be hard to use as a 'case' (possible, though).

```
public static function dataFor($column, $ticket)
    {
        switch ($column) {
            // Ticket ID column
            case 'ticket_id':
                return $ticket['ticket_id'];
                break;

            // Status column
            case 'status':
            case 'type':
            case 'component':
            case 'priority':
            case 'severity':
                return $ticket[{$column}_name];
                break;

            // Votes
            case 'votes':
                return $ticket['votes'];
                break;
        }

        // If we're still here, it may be a custom field
        if ($value = $ticket->customFieldValue($column)) {
            return $value->value;
        }

        // Nothing!
        return '';
    }
```

17.2.244 Ternary In Concat

TeamPass

Ternary In Concat, in includes/libraries/protect/AntiXSS/UTF8.php:5409.

The concatenations in the initial comparison are disguised casting. When $str2 is empty too, the ternary operator yields a 0, leading to a systematic failure.

```
$str1 . '' === $str2 . '' ? 0 : strnatcmp(self::strtonatfold($str1),␣
↪self::strtonatfold($str2))
```

17.2.245 Test Then Cast

Dolphin

Test Then Cast, in wp-admin/includes/misc.php:74.

$aLimits['per_page'] is tested for existence and not false. Later, it is cast from string to int : yet, a '0.1' string value would pass the test, and end up filling $aLimits['per_page'] with 0.

```
if (isset($aLimits['per_page']) && $aLimits['per_page'] !== false)
            $this->aCurrent['paginate']['perPage'] = (int)$aLimits['per_page'];
```

SuiteCrm

Test Then Cast, in modules/jjwg_Maps/controller.php:1035.

$marker['lat'] is compared to the string '0', which actually transtype it to integer, then it is cast to string for map_marker_data_points() needs and finally, it is cast to float, in case of a correction. It would be safer to test it in its string type, since floats are not used as array indices.

```
if ($marker['lat'] != '0' && $marker['lng'] != '0') {

            // Check to see if marker point already exists and apply offset if needed
            // This often occurs when an address is only defined by city, state, zip.
            $i = 0;
            while (isset($this->map_marker_data_points[(string) $marker['lat
↪']][(string) $marker['lng']]) &&
            $i < $this->settings['map_markers_limit']) {
                $marker['lat'] = (float) $marker['lat'] + (float) $this->settings[
↪'map_duplicate_marker_adjustment'];
                $marker['lng'] = (float) $marker['lng'] + (float) $this->settings[
↪'map_duplicate_marker_adjustment'];
                $i++;
            }
```

17.2.246 Timestamp Difference

Zurmo

Timestamp Difference, in app/protected/modules/import/jobs/ImportCleanupJob.php:73.

This is wrong twice a year, in countries that has day-ligth saving time. One of the weeks will be too short, and the other will be too long.

```
/**
         * Get all imports where the modifiedDateTime was more than 1 week ago.  Then
         * delete the imports.
         * (non-PHPdoc)
         * @see BaseJob::run()
         */
        public function run()
        {
            $oneWeekAgoTimeStamp =␣
↪DateTimeUtil::convertTimestampToDbFormatDateTime(time() - 60 * 60 *24 * 7);
```

shopware

Timestamp Difference, in engine/Shopware/Controllers/Backend/Newsletter.php:150.

When daylight saving strike, the email may suddenly be locked for 1 hour minus 30 seconds ago. The lock will be set for the rest of the hour, until the server catch up.

```
// Check lock time. Add a buffer of 30 seconds to the lock time (default request time)
            if (!empty($mailing['locked']) && strtotime($mailing['locked']) > time() -
↪ 30) {
                echo "Current mail: '" . $subjectCurrentMailing . "'\n";
                echo "Wait " . (strtotime($mailing['locked']) + 30 - time()) . "␣
↪seconds ...\n";
                return;
            }
```

17.2.247 Unconditional Break In Loop

LiveZilla

Unconditional Break In Loop, in wp-admin/includes/misc.php:74.

Only one row is read from the DBManager, and the rest is ignored. The result has no more than one result, basedd on the *LIMIT 1* clause in the SQL. The while loop may be removed.

```
$result = DBManager::Execute(true, "SELECT * FROM `" . DB_PREFIX . DATABASE_STATS_
↪AGGS . "` WHERE `month`>0 AND ((`year`='" . DBManager::RealEscape(date("Y")) . "'␣
↪AND `month`<'" . DBManager::RealEscape(date("n")) . "') OR (`year`<'" .␣
↪DBManager::RealEscape(date("Y")) . "')) AND (`aggregated`=0 OR `aggregated`>" .␣
↪(time() - 300) . ") AND `day`=0 ORDER BY `year` ASC,`month` ASC LIMIT 1;");
        if ($result)
            while ($row = DBManager::FetchArray($result)) {
                if (empty($row["aggregated"])) {
                    DBManager::Execute(true, "UPDATE `" . DB_PREFIX . DATABASE_STATS_
↪AGGS . "` SET `aggregated`=" . time() . " WHERE `year`=" . $row["year"] . " AND␣
↪`month`=" . $row["month"] . " AND `day`=0 LIMIT 1;");
                    $this->AggregateMonth($row["year"], $row["month"]);
                }
                return false;
            }
```

MediaWiki

Unconditional Break In Loop, in includes/htmlform/HTMLFormField.php:138.

The final break is useless : the execution has already reached the end of the loop.

```
for ( $i = count( $thisKeys ) - 1; $i >= 0; $i-- ) {
                    $keys = array_merge( array_slice( $thisKeys, 0, $i ), $nameKeys );
                    $data = $alldata;
                    foreach ( $keys as $key ) {
                            if ( !is_array( $data ) || !array_key_exists( $key, $data␣
↪) ) {
                                    continue 2;
                            }
                            $data = $data[$key];
                    }
                    $testValue = (string)$data;
                    break;
            }
```

17.2.248 Unpreprocessed Values

Zurmo

Unpreprocessed Values, in app/protected/core/utils/ZurmoTranslationServerUtil.php:79.

It seems that a simple concatenation could be used here. There is another call to this expression in the code, and a third that uses 'PATCH_VERSION' on top of the two others.

```
join('.', array(MAJOR_VERSION, MINOR_VERSION))
```

Piwigo

Unpreprocessed Values, in include/random_compat/random.php:34.

PHP_VERSION is actually build with PHP_MAJOR_VERSION, PHP_MINOR_VERSION and PHP_RELEASE_VERSION. There is also a compact version : PHP_VERSION_ID

```
explode('.', PHP_VERSION);
```

17.2.249 Unused Global

Dolphin

Unused Global, in Dolphin-v.7.3.5/modules/boonex/forum/classes/DbForum.php:548.

$gConf is not used in this method, and may be safely avoided.

```
function getUserPostsList ($user, $sort, $limit = 10)
    {
        global $gConf;

        switch ($sort) {
            case 'top':
                $order_by = " t1.`votes` DESC ";
                break;
            case 'rnd':
                $order_by = " RAND() ";
                break;
            default:
                $order_by = " t1.`when` DESC ";
        }

        $sql =  "
        SELECT t1.`forum_id`, t1.`topic_id`, t2.`topic_uri`, t2.`topic_title`, t1.
↪`post_id`, t1.`user`, `post_text`, t1.`when`
            FROM " . TF_FORUM_POST . " AS t1
        INNER JOIN " . TF_FORUM_TOPIC . " AS t2
            ON (t1.`topic_id` = t2.`topic_id`)
        WHERE  t1.`user` = '$user' AND `t2`.`topic_hidden` = '0'
        ORDER BY " . $order_by . "
        LIMIT $limit";

        $a = $this->getAll ($sql);
        $this->_cutPostText($a);
```

(continues on next page)

(continued from previous page)

```
        return $a;
    }
```

17.2.250 Use Count Recursive

WordPress

Use Count Recursive, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

PrestaShop

Use Count Recursive, in controllers/admin/AdminSearchController.php:342.

This could be improved with count() recursive and a array_filter call, to remove empty $list.

```
$nb_results = 0;
            foreach ($this->_list as $list) {
                if ($list != false) {
                    $nb_results += count($list);
                }
            }
```

17.2.251 Useless Brackets

ChurchCRM

Useless Brackets, in src/Menu.php:72.

Difficut to guess what was before the block here. It doesn't have any usage for control flow.

```
$new_row = false;
        $count_people = 0;

        {
            foreach ($peopleWithBirthDays as $peopleWithBirthDay) {
                if ($new_row == false) {
                    ?>

                    <div class=row>
                <?php
                    $new_row = true;
                } ?>
                <div class=col-sm-3>
```

Piwigo

Useless Brackets, in picture.php:342.

There is no need for block braces with case. In fact, it does give a false sense of break, while the case will still fall over to the next one.

```
case 'rate' :
    {
      include_once(PHPWG_ROOT_PATH.'include/functions_rate.inc.php');
      rate_picture($page['image_id'], $_POST['rate']);
      redirect($url_self);
    }
```

17.2.252 Useless Type Casting

FuelCMS

Useless Type Casting, in fuel/codeigniter/core/URI.php:214.

substr() always returns a string, so there is no need to enforce this.

```
if (isset($_SERVER['SCRIPT_NAME'][0]))
            {
                    if (strpos($uri, $_SERVER['SCRIPT_NAME']) === 0)
                    {
                            $uri = (string) substr($uri, strlen($_SERVER['SCRIPT_NAME
↪']));
                    }
                    elseif (strpos($uri, dirname($_SERVER['SCRIPT_NAME'])) === 0)
                    {
                            $uri = (string) substr($uri, strlen(dirname($_SERVER[
↪'SCRIPT_NAME'])));
                    }
            }
```

ThinkPHP

Useless Type Casting, in ThinkPHP/Library/Think/Db/Driver/Sqlsrv.class.php:67.

A comparison always returns a boolean, except for the spaceship operator.

```
foreach ($result as $key => $val) {
                $info[$val['column_name']] = array(
                    'name'    => $val['column_name'],
                    'type'    => $val['data_type'],
                    'notnull' => (bool) ('' === $val['is_nullable']), // not null is␣
↪empty, null is yes
                    'default' => $val['column_default'],
                    'primary' => false,
                    'autoinc' => false,
                );
            }
```

17.2.253 Useless Check

Magento

Useless Check, in wp-admin/includes/misc.php:74.

This code assumes that $delete is an array, then checks if it empty. Foreach will take care of the empty check.

```
if (!empty($delete)) {
            foreach ($delete as $categoryId) {
                $where = array(
                    'product_id = ?'  => (int)$object->getId(),
                    'category_id = ?' => (int)$categoryId,
                );

                $write->delete($this->_productCategoryTable, $where);
            }
        }
```

Phinx

Useless Check, in src/Phinx/Migration/Manager.php:828.

If $dependencies is not empty, foreach() skips the loops.

```
private function getSeedDependenciesInstances(AbstractSeed $seed)
    {
        $dependenciesInstances = [];
        $dependencies = $seed->getDependencies();
        if (!empty($dependencies)) {
            foreach ($dependencies as $dependency) {
                foreach ($this->seeds as $seed) {
                    if (get_class($seed) === $dependency) {
                        $dependenciesInstances[get_class($seed)] = $seed;
                    }
                }
            }
        }

        return $dependenciesInstances;
    }
```

17.2.254 Useless Global

Zencart

Useless Global, in admin/includes/modules/newsletters/newsletter.php:25.

$_GET is always a global variable. There is no need to declare it global in any scope.

```
function choose_audience() {
        global $_GET;
```

HuMo-Gen

Useless Global, in relations.php:332.

It is hard to spot that $generY is useless, but this is the only occurrence where $generY is refered to as a global. It is not accessed anywhere else as a global (there are occurrences of $generY being an argument), and it is not even assigned within that function.

```
function calculate_ancestor($pers) {
    global $db_functions, $reltext, $sexe, $sexe2, $spouse, $special_spouseY,
↪$language, $ancestortext, $dutchtext, $selected_language, $spantext, $generY,
↪$foundY_nr, $rel_arrayY;
```

17.2.255 Useless Parenthesis

Mautic

Useless Parenthesis, in code/app/bundles/EmailBundle/Controller/AjaxController.php:85.

Parenthesis are useless around $progress[1], and around the division too.

```
$dataArray['percent'] = ($progress[1]) ? ceil(($progress[0] / $progress[1]) * 100) :␣
↪100;
```

Woocommerce

Useless Parenthesis, in includes/class-wc-coupon.php:437.

Parenthesis are useless for calculating $discount_percent, as it is a divisition. Moreover, it is not needed with $discount, (float) applies to the next element, but it does make the expression more readable.

```
if ( wc_prices_include_tax() ) {
    $discount_percent = ( wc_get_price_including_tax( $cart_item['data'] ) * $cart_
↪item_qty ) / WC()->cart->subtotal;
} else {
    $discount_percent = ( wc_get_price_excluding_tax( $cart_item['data'] ) * $cart_
↪item_qty ) / WC()->cart->subtotal_ex_tax;
}
$discount = ( (float) $this->get_amount() * $discount_percent ) / $cart_item_qty;
```

17.2.256 Useless Switch

Phpdocumentor

Useless Switch, in fuel/modules/fuel/libraries/Inspection.php:349.

This method parses comments. In fact, comments are represented by other tokens, which may be added or removed at time while coding.

```
public function parse_comments($code)
    {
            $comments = array();
            $tokens = token_get_all($code);

            foreach($tokens as $token)
            {
                    switch($token[0])
```

(continues on next page)

(continued from previous page)

```
                    {
                            case T_DOC_COMMENT:
                                    $comments[] = $token[1];
                                    break;
                    }
            }
            return $comments;

    }
```

Dolphin

Useless Switch, in Dolphin-v.7.3.5/inc/classes/BxDolModuleDb.php:34.

$aParams is an argument : this code looks like the switch is reserved for future use.

```
function getModulesBy($aParams = array())
    {
            $sMethod = 'getAll';
        $sPostfix = $sWhereClause = "";

        $sOrderClause = "ORDER BY `title`";
        switch($aParams['type']) {
            case 'path':
                    $sMethod = 'getRow';
                $sPostfix .= '_path';
                $sWhereClause .= "AND `path`='" . $aParams['value'] . "'";
                break;
        }
```

17.2.257 Useless Unset

Tine20

Useless Unset, in tine20/Felamimail/Controller/Message.php:542.

$_rawContent is unset after being sent to the stream. The variable is a parameter, and will be freed at the end of the call of the method. No need to do it explicitly.

```
protected function _createMimePart($_rawContent, $_partStructure)
    {
        if (Tinebase_Core::isLogLevel(Zend_Log::TRACE)) Tinebase_Core::getLogger()->
↪trace(__METHOD__ . '::' . __LINE__ . ' Content: ' . $_rawContent);

        $stream = fopen(php://temp, 'r+');
        fputs($stream, $_rawContent);
        rewind($stream);

        unset($_rawContent);
        //..... More code, no usage of $_rawContent
    }
```

Typo3

Useless Unset, in typo3/sysext/frontend/Classes/Page/PageRepository.php:708.

$row is unset under certain conditions : here, we can read it in the comments. Eventually, the $row will be returned, and turned into a NULL, by default. This will also create a notice in the logs. Here, the best would be to set a null value, instead of unsetting the variable.

```
public function getRecordOverlay($table, $row, $sys_language_content, $OLmode = '')
    {
//....  a lot more code, with usage of $row, and several unset($row)
//...... Reduced for simplicity
                    } else {
                        // When default language is displayed, we never want to␣
↪return a record carrying
                        // another language!
                        if ($row[$GLOBALS['TCA'][$table]['ctrl']['languageField']] >␣
↪0) {
                            unset($row);
                        }
                    }
                }
            }
        }
        foreach ($GLOBALS['TYPO3_CONF_VARS']['SC_OPTIONS']['t3lib/class.t3lib_page.php
↪']['getRecordOverlay'] ?? [] as $className) {
            $hookObject = GeneralUtility::makeInstance($className);
            if (!$hookObject instanceof PageRepositoryGetRecordOverlayHookInterface) {
                throw new \UnexpectedValueException($className . ' must implement␣
↪interface ' . PageRepositoryGetRecordOverlayHookInterface::class, 1269881659);
            }
            $hookObject->getRecordOverlay_postProcess($table, $row, $sys_language_
↪content, $OLmode, $this);
        }
        return $row;
    }
```

17.2.258 Use List With Foreach

MediaWiki

Use List With Foreach, in includes/parser/LinkHolderArray.php:372.

This foreach reads each element from $entries into entry. $entry, in turn, is written into $pdbk, $title and $displayText for easier reuse. 5 elements are read from $entry, and they could be set in their respective variable in the foreach() with a list call. The only on that can't be set is 'query' which has to be tested.

```
foreach ( $entries as $index => $entry ) {
                            $pdbk = $entry['pdbk'];
                            $title = $entry['title'];
                            $query = isset( $entry['query'] ) ? $entry['query'] : [];
                            $key = "$ns:$index";
                            $searchkey = "<!--LINK'\" $key-->";
                            $displayText = $entry['text'];
                            if ( isset( $entry['selflink'] ) ) {
                                $replacePairs[$searchkey] =␣
↪Linker::makeSelfLinkObj( $title, $displayText, $query );
```

(continues on next page)

(continued from previous page)

```
                                        continue;
                                }
                                if ( $displayText === '' ) {
                                        $displayText = null;
                                } else {
                                        $displayText = new HtmlArmor( $displayText );
                                }
                                if ( !isset( $colours[$pdbk] ) ) {
                                        $colours[$pdbk] = 'new';
                                }
                                $attribs = [];
                                if ( $colours[$pdbk] == 'new' ) {
                                        $linkCache->addBadLinkObj( $title );
                                        $output->addLink( $title, 0 );
                                        $link = $linkRenderer->makeBrokenLink(
                                                $title, $displayText, $attribs, $query
                                        );
                                } else {
                                        $link = $linkRenderer->makePreloadedLink(
                                                $title, $displayText, $colours[$pdbk],
↪$attribs, $query
                                        );
                                }

                                $replacePairs[$searchkey] = $link;
                        }
```

17.2.259 Use Positive Condition

SPIP

Use Positive Condition, in ecrire/inc/utils.php:925.

if (isset($time[$t])) { } else { } would put the important case in first place, and be more readable.

```
if (!isset($time[$t])) {
            $time[$t] = $a + $b;
    } else {
            $p = ($a + $b - $time[$t]) * 1000;
            unset($time[$t]);
#                   echo "'$p'";exit;
            if ($raw) {
                    return $p;
            }
            if ($p < 1000) {
                    $s = '';
            } else {
                    $s = sprintf("%d ", $x = floor($p / 1000));
                    $p -= ($x * 1000);
            }

            return $s . sprintf($s ? "%07.3f ms" : "%.3f ms", $p);
    }
```

ExpressionEngine

Use Positive Condition, in system/ee/EllisLab/Addons/forum/mod.forum_core.php:9138.

Let's be positive, and start processing the presence of $topic first. And let's call it empty(), not == ''.

```
if ($topic != '')
                                                {
                                                        $sql .= '('.substr($topic, 0, -3).
↪') OR ';
                                                        $sql .= '('.substr($tbody, 0, -3).
↪') ';
                                                }
                                                else
                                                {
                                                        $sql = substr($sql, 0, -3);
                                                }
```

17.2.260 var_dump()... Usage

Tine20

var_dump()... Usage, in tine20/library/Ajam/Connection.php:122.

Two usage of var_dump(). They are protected by configuration, since the debug property must be set to 'true'. Yet, it is safer to avoid them altogether, and log the information to an external file.

```
if($this->debug === true) {
            var_dump($this->getLastRequest());
            var_dump($response);
        }
```

Piwigo

var_dump()... Usage, in include/ws_core.inc.php:273.

This is a hidden debug system : when the response format is not available, the whole object is dumped in the output.

```
function run()
  {
    if ( is_null($this->_responseEncoder) )
    {
      set_status_header(400);
      @header("Content-Type: text/plain");
      echo ("Cannot process your request. Unknown response format.
Request format: ".@$this->_requestFormat." Response format: ".@$this->_responseFormat.
↪"\n");
      var_export($this);
      die(0);
    }
```

17.2.261 While(List() = Each())

OpenEMR

While(List() = Each()), in library/report.inc:153.

The first while() is needed, to read the arbitrary long list returned by the SQL query. The second list may be upgraded with a foreach, to read both the key and the value. This is certainly faster to execute and to read.

```
function getInsuranceReport($pid, $type = primary)
{
    $sql = select * from insurance_data where pid=? and type=? order by date ASC;
    $res = sqlStatement($sql, array($pid, $type));
    while ($list = sqlFetchArray($res)) {
        while (list($key, $value) = each($list)) {
            if ($ret[$key]['content'] != $value && $ret[$key]['date'] < $list['date
↪']) {
                $ret[$key]['content'] = $value;
                $ret[$key]['date'] = $list['date'];
            }
        }
    }

    return $ret;
}
```

Dolphin

While(List() = Each()), in Dolphin-v.7.3.5/modules/boonex/forum/classes/Forum.php:1875.

This clever use of while() and list() is actually a foreach($a as $r) (the keys are ignored)

```
function getRssUpdatedTopics ()
    {
        global $gConf;

        $this->_rssPrepareConf ();

        $a = $this->fdb->getRecentTopics (0);

        $items = '';
        $lastBuildDate = '';
        $ui = array();
        reset ($a);
        while ( list (,$r) = each ($a) ) {
            // acquire user info
            if (!isset($ui[$r['last_post_user']]) && ($aa = $this->_
↪getUserInfoReadyArray ($r['last_post_user'], false)))
                $ui[$r['last_post_user']] = $aa;

            $td = orca_mb_replace('/#/', $r['count_posts'], '[L[# posts]]') . ' &#183;
↪ ' . orca_mb_replace('/#/', $ui[$r['last_post_user']]['title'], '[L[last reply by
↪#]]') . ' &#183; ' . $r['cat_name'] . ' &#187; ' . $r['forum_title'];
```

17.2.262 Wrong Range Check

Dolibarr

Wrong Range Check, in htdocs/includes/phpoffice/PhpSpreadsheet/Spreadsheet.php:1484.

When $tabRatio is 1001, then the condition is valid, and the ratio accepted. The right part of the condition is not executed.

```
public function setTabRatio($tabRatio)
    {
        if ($tabRatio >= 0 || $tabRatio <= 1000) {
            $this->tabRatio = (int) $tabRatio;
        } else {
            throw new Exception('Tab ratio must be between 0 and 1000.');
        }
    }
```

WordPress

Wrong Range Check, in wp-includes/formatting.php:3634.

This condition may be easier to read as *$diff >= WEEK_IN_SECONDS && $diff < MONTH_IN_SECONDS*. When testing for outside this interval, using not is also more readable : *!($diff >= WEEK_IN_SECONDS && $diff < MONTH_IN_SECONDS)*.

```
} elseif ( $diff < MONTH_IN_SECONDS && $diff >= WEEK_IN_SECONDS ) {
            $weeks = round( $diff / WEEK_IN_SECONDS );
            if ( $weeks <= 1 ) {
                    $weeks = 1;
            }
            /* translators: Time difference between two dates, in weeks. %s: Number␣
↪of weeks */
            $since = sprintf( _n( '%s week', '%s weeks', $weeks ), $weeks );
```

17.2.263 Dependant Trait

Zencart

Dependant Trait, in app/library/zencart/CheckoutFlow/src/AccountFormValidator.php:14.

Note that addressEntries is used, and is also expected to be an array or an object with ArrayAccess. $addressEntries is only defined in a class called 'Guest' which is also the only one using that trait. Any other class using the AccountFormValidator trait must define addressEntries.

```
trait AccountFormValidator
{

    abstract protected function getAddressFieldValue($fieldName);

    /**
     * @return bool|int
     */
    protected function errorProcessing()
    {
        $error = false;
        foreach ($this->addressEntries as $fieldName => $fieldDetails) {
```

(continues on next page)

(continued from previous page)

```
            $this->addressEntries[$fieldName]['value'] = $this->getAddressFieldValue(
↪$fieldName);
            $fieldError = $this->processFieldValidator($fieldName, $fieldDetails);
            $this->addressEntries[$fieldName]['error'] = $fieldError;
            $error = $error | $fieldError;
        }
        return $error;
    }
```

17.2.264 Multiple Usage Of Same Trait

NextCloud

Multiple Usage Of Same Trait, in build/integration/features/bootstrap/WebDav.php:41.

WebDav uses Sharing, and Sharing uses Webdav. Once using the other is sufficient.

```
trait WebDav {
    use Sharing;

}
//Trait Sharing is in /build/integration/features/bootstrap/Sharing.php:36
```

17.2.265 No Real Comparison

Magento

No Real Comparison, in app/code/core/Mage/XmlConnect/Block/Catalog/Product/Options/Configurable.php:74.

Compare prices and physical quantities with a difference, so as to avoid rounding errors.

```
if ((float)$option['price'] != 0.00) {
                        $valueNode->addAttribute('price', $option['price']);
                        $valueNode->addAttribute('formated_price', $option['formated_
↪price']);
                    }
```

SPIP

No Real Comparison, in ecrire/maj/v017.php:37.

Here, the current version number is stored as a real number. With a string, though a longer value, it may be compared using the version_compare() function.

```
$version_installee == 1.701
```

17.2.266 One Variable String

Tikiwiki

One Variable String, in lib/wiki-plugins/wikiplugin_addtocart.php:228.

Double-quotes are not needed here. If casting to string is important, the (string) would be more explicit.

```
foreach ($plugininfo['params'] as $key => $param) {
            $default["$key"] = $param['default'];
    }
```

NextCloud

One Variable String, in build/integration/features/bootstrap/BasicStructure.php:349.

Both concatenations could be merged, independantly. If readability is important, why not put them inside curly brackets?

```
public static function removeFile($path, $filename) {
            if (file_exists("$path" . "$filename")) {
                    unlink("$path" . "$filename");
            }
    }
```

17.2.267 Should Typecast

xataface

Should Typecast, in Dataface/Relationship.php:1612.

This is an exact example. A little further, the same applies to intval($max))

```
intval($min);
```

OpenConf

Should Typecast, in author/upload.php:62.

This is another exact example.

```
intval($_POST['pid']);
```

17.2.268 Silently Cast Integer

MediaWiki

Silently Cast Integer, in includes/debug/logger/monolog/AvroFormatter.php:167.

Too many ff in the masks.

```
private function encodeLong( $id ) {
            $high   = ( $id & 0xffffffff00000000 ) >> 32;
            $low    = $id & 0x00000000ffffffff;
            return pack( 'NN', $high, $low );
    }
```

17.2.269 Strings With Strange Space

OpenEMR

Strings With Strange Space, in library/globals.inc.php:3270.

The name of the contry contains both an unsecable space (the first, after Tonga), and a normal space (between Tonga and Islands). Translations are stored in a database, which preserves the unbreakable spaces. This also means that fixing the translation must be applied to every piece of data at the same time. The xl() function, which handles the translations, is also a good place to clean the spaces before searching for the right translation.

```
'to' => xl('Tonga (Tonga Islands)'),
```

Thelia

Strings With Strange Space, in templates/backOffice/default/I18n/fr_FR.php:647.

This is another example with a translation sentence. Here, the unbreakable space is before the question mark : this is a typography rule, that is common to many language. This would be a false positive, unless typography is handled by another part of the software.

```
'Mot de passe oublié ?'
```

17.2.270 Inconsistent Variable Usage

WordPress

Inconsistent Variable Usage, in wp-includes/IXR/class-IXR-client.php:86.

$request is used successively as an object (IXR_Request), then as a string (The POST). Separatring both usage with different names will help readability.

```
$request = new IXR_Request($method, $args);
        $length = $request->getLength();
        $xml = $request->getXml();
        $r = "\r\n";
        $request  = "POST {$this->path} HTTP/1.0$r";
```

17.2.271 Lost References

WordPress

Lost References, in wp-admin/includes/misc.php:74.

This code actually loads the file, join it, then split it again. file() would be sufficient.

```
$markerdata = explode( "\n", implode( '', file( $filename ) ) );
```

17.2.272 Strange Name For Variables

FuelCMS

Strange Name For Variables, in fuel/modules/fuel/libraries/parser/dwoo/Dwoo/Adapters/CakePHP/dwoo.php:86.

Three _ is quite a lot for variables. Would they not be parameters but global variables, that would still be quite a lot.

```
public function _render($___viewFn, $___data_for_view, $___play_safe = true,
↪$loadHelpers = true) {
    /**/
}
```

PhpIPAM

Strange Name For Variables, in app/admin/sections/edit-result.php:56.

$sss is the end-result of a progression, from $subsections (3s) to $ss to $sss. Although it is understandable from the code, a fuller name, like $subsection_subnet or $one_subsection_subnet would make this more readable.

```
//fetch subsection subnets
            foreach($subsections as $ss) {
                    $subsection_subnets = $Subnets->fetch_section_subnets($ss->id); //
↪fetch all subnets in subsection
                    if(sizeof($subsection_subnets)>0) {
                            foreach($subsection_subnets as $sss) {
                                    $out[] = $sss;
                            }
                    }
                    $num_subnets = $num_subnets + sizeof($subsection_subnets);
                    //count all addresses that will be deleted!
                    $ipcnt = $Addresses->count_addresses_in_multiple_subnets($out);
            }
```

17.2.273 Non Ascii Variables

Magento

Non Ascii Variables, in dev/tests/functional/tests/app/Mage/Checkout/Test/Constraint/AssertOrderWithMultishippingSuccessPlacedMes

The initial C is actually a russian C.

```
$heckoutMultishippingSuccess
```

17.2.274 Used Once Variables

shopware

Used Once Variables, in _sql/migrations/438-add-email-template-header-footer-fields.php:115.

In the updateEmailTemplate method, $generatedQueries collects all the generated SQL queries. $generatedQueries is not initialized, and never used after initialization.

```
private function updateEmailTemplate($name, $content, $contentHtml = null)
    {
        $sql = <<<SQL
UPDATE `s_core_config_mails` SET `content` = "$content" WHERE `name` = "$name" AND␣
↪dirty = 0
SQL;
        $this->addSql($sql);

        if ($contentHtml != null) {
            $sql = <<<SQL
UPDATE `s_core_config_mails` SET `content` = "$content", `contentHTML` = "$contentHtml
↪" WHERE `name` = "$name" AND dirty = 0
SQL;
            $generatedQueries[] = $sql;
        }

        $this->addSql($sql);
    }
```

Vanilla

Used Once Variables, in library/core/class.configuration.php:1461.

In this code, $cachedConfigData is collected after storing date in the cache. Gdn::cache()->store() does actual work, so its calling is necessary. The result, collected after execution, is not reused in the rest of the method (long method, not all is shown here). Removing such variable is a needed clean up after development and debug, but also prevents pollution of the variable namespace.

```
// Save to cache if we're into that sort of thing
                $fileKey = sprintf(Gdn_Configuration::CONFIG_FILE_CACHE_KEY, $this->
↪Source);
                if ($this->Configuration && $this->Configuration->caching() &&␣
↪Gdn::cache()->type() == Gdn_Cache::CACHE_TYPE_MEMORY && Gdn::cache()->
↪activeEnabled()) {
                    $cachedConfigData = Gdn::cache()->store($fileKey, $data, [
                        Gdn_Cache::FEATURE_NOPREFIX => true,
                        Gdn_Cache::FEATURE_EXPIRY => 3600
                    ]);
                }
```

17.2.275 Used Once Variables (In Scope)

shopware

Used Once Variables (In Scope), in _sql/migrations/438-add-email-template-header-footer-fields.php:115.

In the updateEmailTemplate method, $generatedQueries collects all the generated SQL queries. $generatedQueries is not initialized, and never used after initialization.

```
private function updateEmailTemplate($name, $content, $contentHtml = null)
    {
        $sql = <<<SQL
UPDATE `s_core_config_mails` SET `content` = "$content" WHERE `name` = "$name" AND␣
↪dirty = 0
```

(continues on next page)

(continued from previous page)

```
SQL;
        $this->addSql($sql);

        if ($contentHtml != null) {
            $sql = <<<SQL
UPDATE `s_core_config_mails` SET `content` = "$content", `contentHTML` = "$contentHtml
↪" WHERE `name` = "$name" AND dirty = 0
SQL;
            $generatedQueries[] = $sql;
        }

        $this->addSql($sql);
    }
```

17.2.276 Written Only Variables

Dolibarr

Written Only Variables, in htdocs/ecm/class/ecmdirectory.class.php:692.

$val is only written, as only the keys are used. $val may be skipped by applying the foreach to array_keys($this->cats), instead of the whole array.

```
// We add properties fullxxx to all elements
            foreach($this->cats as $key => $val)
            {
                    if (isset($motherof[$key])) continue;
                    $this->build_path_from_id_categ($key, 0);
            }
```

SuiteCrm

Written Only Variables, in modules/Campaigns/utils.php:820.

$email_health is used later in the method; while $email_components is only set, and never used.

```
//run query for mail boxes of type 'bounce'
        $email_health = 0;
        $email_components = 2;
        $mbox_qry = "select * from inbound_email where deleted ='0' and mailbox_type␣
↪= 'bounce'";
        $mbox_res = $focus->db->query($mbox_qry);

        $mbox = array();
        while ($mbox_row = $focus->db->fetchByAssoc($mbox_res)) {
            $mbox[] = $mbox_row;
        }
```

CHAPTER 18

Training Database

18.1 Introduction

lorem lipsum @cba A number of applications were scanned in order to find real life examples of patterns. They are listed here :

18.2 List of Applications

- ChurchCRM
- Cleverstyle
- Contao
- Dolibarr
- Dolphin
- Edusoho
- ExpressionEngine
- FuelCMS
- HuMo-Gen
- LiveZilla
- Magento
- Mautic
- MediaWiki
- NextCloud
- OpenConf

- OpenEMR
- Phinx
- PhpIPAM
- Phpdocumentor
- Piwigo
- PrestaShop
- SPIP
- SugarCrm
- SuiteCrm
- TeamPass
- Thelia
- ThinkPHP
- Tikiwiki
- Tine20
- Traq
- Typo3
- Vanilla
- Woocommerce
- WordPress
- XOOPS
- Zencart
- Zend-Config
- Zurmo
- opencfp
- phpMyAdmin
- phpadsnew
- shopware
- xataface

CHAPTER 19

Installation

19.1 Summary

19.2 Requirements

Exakat relies on external components : mandatroy or optional.

Basic requirements :

- exakat.phar, the main code.
- Gremlin server : exakat uses this graph database and the Gremlin 3 traversal language. Currently, only Gremlin Server is supported, with the tinkergraph and neo4j storage engine. Version 3.4.x is the recommended version, while version 3.3.x are still supported. Gremlin version 3.2.* and 3.5.* are unsupported.
- Java 8.x. Java 9.x/10.x will be supported later. Java 7.x was used, but is not actively supported.
- PHP 8.0 to run. PHP 8.0 is recommended, PHP 7.2 or later are possible. This version requires the PHP extensions curl, hash, phar, sqlite3, tokenizer, mbstring and json.

Optional requirements :

- PHP 5.2 to 8.1-dev for analysis purposes. Those versions only require the ext/tokenizer extension.
- VCS (Version Control Software), such as Git, SVN, bazaar, Mercurial. They all are optional, though git is recommended.
- Archives, such as zip, tgz, tbz2 may also be opened with optional helpers (See *Installation guide for optional tools*).

OS requirements : Exakat has beed tested on OSX, Debian and Ubuntu (up to 20.04). Exakat should work on Linux distributions, may be with little work. Exakat hasn't been tested on Windows at all.

For installation, curl or wget, and zip are needed.

19.3 Download exakat.phar

You can download exakat directly from https://www.exakat.io/.

This server also provides older versions of Exakat. It is recommended to always download the last version, which is available with https://www.exakat.io/versions/index.php?file=latest.

For each version, MD5 and SHA256 signatures are available. The downloaded MD5 must match the one in the related .md5 file. The .md5 also has the version number, for extra check.

```
curl -o exakat.phar 'https://www.exakat.io/versions/index.php?file=latest'

curl -o exakat.phar.md5 'https://www.exakat.io/versions/index.php?file=latest.md5'
//19485adb7d43b43f7c01b7153ae82881  exakat-2.0.0.phar
md5sum exakat.phar.md5
// Example :
//19485adb7d43b43f7c01b7153ae82881  exakat.phar

curl -o exakat.phar.sha256 'https://www.exakat.io/versions/index.php?file=latest.
↪sha256'
//d838c9ec9291e15873137693da2a0038a67c2f15c2282b89f09f27f23d24d27f  exakat-2.0.0.phar
sha256sum exakat.phar.md5
// Example :
//d838c9ec9291e15873137693da2a0038a67c2f15c2282b89f09f27f23d24d27f  exakat.phar

// Check with GPG signature
curl -o exakat.sig 'https://www.exakat.io/versions/index.php?file=latest.sig'
// Optional step : Download the Key
gpg --recv-keys 5EDF7EA4
// Check with GPG signature
gpg --verify exakat.sig exakat.phar
// Good result :
//gpg: Signature made Tue Nov  5 07:48:34 2019 CET using RSA key ID 5EDF7EA4
//gpg: Good signature from "Seguy Damien <damien.seguy@gmail.com>" [ultimate]
```

19.4 Installation with exakat.phar

Exakat.phar includes its own installation script, as long as PHP is available. Exakat will then check different pre-requisites, and proceed to install some of the last elements.

Exakat checks for Java and Zip installations. Then, it downloads tinkergraph and the Neo4j plugin from exakat.io and runs the *doctor* command.

The script is based on the one displayed on the next section.

You can use the *install* command this way :

```
mkdir exakat
cd exakat
curl -o exakat.phar 'https://www.exakat.io/versions/index.php?file=latest'
php exakat.phar install
```

19.5 Installation on OSX

Paste the following commands in a terminal prompt. It downloads Exakat, and installs tinkerpop version 3.4.8. PHP 7.0 or more recent, curl, homebrew are required.

19.5.1 OSX installation with tinkergraph 3.4.8

This is the installation script for Exakat and tinkergraph 3.4.8.

```
mkdir exakat
cd exakat
curl -o exakat.phar 'https://www.exakat.io/versions/index.php?file=latest'
curl -o apache-tinkerpop-gremlin-server-3.4.8-bin.zip 'https://www.exakat.io/versions/
↪apache-tinkerpop-gremlin-server-3.4.8-bin.zip'
unzip apache-tinkerpop-gremlin-server-3.4.8-bin.zip
mv apache-tinkerpop-gremlin-server-3.4.8 tinkergraph
rm -rf apache-tinkerpop-gremlin-server-3.4.8-bin.zip

# Optional : install neo4j engine.
cd tinkergraph
./bin/gremlin-server.sh install org.apache.tinkerpop neo4j-gremlin 3.4.8
cd ..

php exakat.phar doctor
```

19.5.2 OSX installation troubleshooting

It has be reported that installation fails on OSX 10.11 and 10.12, with error similar to 'Error grabbing Grapes'. To fix this, use the following in command line :

```
rm -r ~/.groovy/grapes/
rm -r ~/.m2/
```

They remove some files for grapes, that it will rebuild later. Then, try again the optional install instructions.

19.6 Installation on Debian/Ubuntu

19.6.1 Debian/Ubuntu installation with Tinkergraph 3.4.8

Paste the following commands in a terminal prompt. It installs Exakat most recent version with Tinkergraph 3.4.8. PHP 7.3 (7.0 or more recent), wget and unzip are expected.

```
mkdir exakat
cd exakat
wget -O exakat.phar https://www.exakat.io/versions/index.php?file=latest
wget -O apache-tinkerpop-gremlin-server-3.4.8-bin.zip 'https://www.exakat.io/versions/
↪apache-tinkerpop-gremlin-server-3.4.8-bin.zip'
unzip apache-tinkerpop-gremlin-server-3.4.8-bin.zip
mv apache-tinkerpop-gremlin-server-3.4.8 tinkergraph
rm -rf apache-tinkerpop-gremlin-server-3.4.8-bin.zip

# Optional : install neo4j engine.
cd tinkergraph
./bin/gremlin-server.sh install org.apache.tinkerpop neo4j-gremlin 3.4.8
cd ..

php exakat.phar doctor
```

19.6.2 Prerequisites : Full installation with Debian/Ubuntu

The following commands are an optional pre-requisite to the installation guide, that just follows. If something is missing in the next section, check with this section that all has beed installed correctly.

```
//// Installing PHP from sury.org
apt update
apt install apt-transport-https lsb-release ca-certificates

wget -O /etc/apt/trusted.gpg.d/php.gpg https://packages.sury.org/php/apt.gpg
sh -c 'echo "deb https://packages.sury.org/php/ $(lsb_release -sc) main" > /etc/apt/
↪sources.list.d/php.list'
apt update

apt-get install php7.2 php7.2-common php7.2-cli php7.2-curl php7.2-json php7.2-
↪mbstring php7.2-sqlite3

//// Installing Java JDK
echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee /etc/
↪apt/sources.list.d/webupd8team-java.list
echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee -a /
↪etc/apt/sources.list.d/webupd8team-java.list
apt-get update

echo debconf shared/accepted-oracle-license-v1-1 select true | debconf-set-selections
echo debconf shared/accepted-oracle-license-v1-1 seen true | debconf-set-selections
DEBIAN_FRONTEND=noninteractive  apt-get install -y --force-yes oracle-java8-installer␣
↪oracle-java8-set-default

//// Installing other tools
apt-get update && apt-get install -y --no-install-recommends git subversion mercurial␣
↪lsof unzip
```

19.7 Installation guide with Composer

19.7.1 Composer installation first run

To install Exakat with composer, you can use the following commands:

```
mkdir exakat
cd exakat
composer require exakat/exakat
php vendor/bin/exakat install -v
```

The final command checks for the presence of Java and unZip utility. Then, it installs a local copy of a Gremlin server. This is needed to run Exakat.

To run your first audit, use the following commands:

```
php vendor/bin/exakat init -p sculpin -R 'https://github.com/sculpin/sculpin.git'
php vendor/bin/exakat project -p sculpin
```

The final audit is now in the *projects/sculpin/report* directory.

19.8 Installation guide with Docker

There are multiple ways to use exakat with docker. There is an image with a full exakat installation, which run with a traditional installation, or inside the audited code. Or, You may use Docker with a standard installation, to run useful part, such as a specific PHP version or the central database.

image:: images/exakat-and-docker.png

19.8.1 Docker image for Exakat with projects folder

Installation with Docker is easy, and convenient. It hides the dependency of the graph database, and keeps all files in the 'projects' folder, created in the working directory.

Currently, Docker installation only ships with one PHP version (7.3), and with support for bazaar, composer, git, mercurial, svn, and zip.

- Install Docker
- Start Docker
- Pull exakat. The official docker page is exakat/exakat.

```
docker pull exakat/exakat
```

- Check-run exakat :

```
docker run -it -v $(pwd)/projects:/usr/src/exakat/projects --rm --name my-exakat␣
↪exakat/exakat exakat version
docker run -it -v $(pwd)/projects:/usr/src/exakat/projects --rm --name my-exakat␣
↪exakat/exakat exakat doctor
```

- Init a project :

```
docker run -it -v $(pwd)/projects:/usr/src/exakat/projects --rm --name my-exakat␣
↪exakat/exakat exakat init -p <project name> -R <vcs_url>
```

If you need SSH credentials to clone a project, you may import your SSH keys in the repository by assigning them to the *exakat* user. With the commmand below, the local set of keys of user 'my-user' are assigned to *exakat* in the container. Note that those keys will request the passphrase, which will prevent their usage.

```
docker run -it -v /home/my-user/.ssh:/home/exakat/ssh -v $(pwd)/projects:/usr/src/
↪exakat/projects --rm --name my-exakat exakat/exakat exakat init -p <project name> -
↪R <vcs_url>
```

- Run exakat :

```
docker run -it -v $(pwd)/projects:/usr/src/exakat/projects --rm --name my-exakat␣
↪exakat/exakat exakat project -p <project name>
```

- Run exakat directly in the code base. For that, the code needs to have the .exakat.yml or .exakat.ini file available at the root. Then, you may call exakat with the 'project' command, without other options.

```
docker run -it -v $(pwd)/projects:/usr/src/exakat/projects --rm --name my-exakat␣
↪exakat/exakat exakat project
```

For large code bases, it may be necessary to increase the allocated memory for the graph database. Do this by using the JAVA_OPTIONS environment variable when you start the docker command : this example gives 2Gb of RAM to the graphdb. That should cover medium size applications.

```
docker run -it -e JAVA_OPTIONS="-Xms32m -Xmx2g" -v $(pwd)/projects:/usr/src/exakat/
↪projects --rm --name my-exakat exakat/exakat exakat
```

You may run any exakat command by prefixing it with the following command :

```
docker run -it -v $(pwd)/projects:/usr/src/exakat/projects --rm --name my-exakat␣
↪exakat/exakat exakat
```

You may also create a handy shortcut, by creating an exakat.sh script and put it in your PATH :

```
cat 'docker run -it -v $(pwd)/projects:/usr/src/exakat/projects --rm --name my-exakat␣
↪exakat/exakat exakat $1' > /etc/local/sbin/exakat.sh
chmod u+x  /etc/local/sbin/exakat.sh
./exakat.sh version
```

19.8.2 Docker image for Exakat with projects folder

To run exakat inside the audited code, you must configure the *.exakat.ini* or *.exakat.yaml* file. See Add Exakat To Your CI Pipeline.

Then, you can run the following command, with docker :

```
docker run -it --rm -v `$pwd`:/src exakat/exakat:latest exakat project -v
```

19.8.3 Docker PHP image with Exakat

Exakat recognizes docker images configured as PHP binaries. Instead of configuring exakat with local binaries, such as */usr/bin/php*, you may configure a specific PHP version with a docker image.

Open the *config/exakat.ini* file, at the root of the exakat installation, and use the following value :

```
// configuration with the 'tetraweb/php:5.5' image.
;php55 = tetraweb/php:5.5
php56 = tetraweb/php:5.6
# classic configuration with local binary
php73 = /usr/bin/php
```

The image may be any docker image that provides a PHP binary. We suggest using tetraweb/php, which supports PHP 5.5 to 7.1. There are other images available, and you may also roll out your own.

19.8.4 Docker Gremlin image with Exakat

Exakat is able to use only the central database, Gremlin, as a docker image. This is convenient, as the database is only a temporary database, and those data are not necessary for producing the final reports.

This image is under construction, and will be soon available.

19.9 Installation guide as Github Action

19.9.1 Github Action

Github Action is a way to "Automate, customize, and execute your software development workflows right in your repository". Exakat may be run on Github platform.

19.9.2 Github Action for Exakat

To add Exakat to your repository on Github, create a file *.github/workflows/test.yml*, at the root of your repository (*.github/workflows* might already exists).

In the file, use the following YAML code. It will create an automatic action, on push and pull_request actions, that runs Exakat and display the issues found in the workflow panel. It is also possible to run manually this action.

```
on: [push, pull_request]
name: Test
jobs:
  exakat:
    name: Exakat
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Exakat
      uses: docker://exakat/exakat-ga
```

Note : it is recommended to edit this file directly on github.com, as it cannot be pushed from a remote repository.

Then, you can use the *Action* button, next to 'Pull requests'.

19.9.3 Exakat Docker image for Github Action

A Docker image is released with Exakat's version automatically, to be used with Github Action. It is available at https://hub.docker.com/r/exakat/exakat-ga.

You can run it in any given directory like this:

```
cd /path/to/code
docker pull exakat/exakat-ga
docker run --rm -it -v ${PWD}:/app exakat/exakat-ga:latest
```

19.10 Using multiple PHP versions

You need at least one version of PHP to run exakat. This version needs the curl, hash, tokenizer, hash and sqlite3 extensions. They all are part of the core.

Extra PHP-CLI versions allow more linting of the code. They only need to have the tokenizer extension available.

Exakat recommends PHP 8.0.x (or newer version) to run Exakat. We also recommend the installation of PHP versions 5.6, 7.1, 7.2, 7.3, 7.4, 8.0 and 8.1 (aka php-src master).

To install easily various versions of PHP, use the ondrej repository. Check The main PPA for PHP (8.0, 7.4, 7.3, 7.2, 7.1, 7.0, 5.6). You may also check the dotdeb repository, at dotdeb instruction or compile PHP yourself.

19.11 Installation guide for optional tools

Exakat is able to use a variety of tools to access PHP code to audit. Some external tools are necessary. You can check which tools are recognized locally with the *exakat doctor -v* command.

- bazaar : the *bzr* command must be available.
- composer : the *composer* command must be available.
- CVS : the *cvs* command must be available
- Git : the *git* command must be available.
- mercurial : the *hg* must be available
- Svn : the *svn* command must be available.
- tgz : the *tar* and *gunzip* commands must be available
- tbz : the *tar* and *bunzip2* commands must be available.
- rar : the *rar* commands must be available.
- zip : the *unzip* command must be available.
- 7z : the *7z* command must be available

The binaries above are used with the *init* and *update* commands, to get the source code. They are optional.

CHAPTER 20

Upgrading

20.1 Upgrading

Upgrade exakat with the *upgrade* command.

php exakat.phar upgrade

Exakat returns the current status :

This needs some updating (Current : 0.9.7c, Latest: 2.1.9)

To make exakat update itself, runs the same command, with the *-u* option. Exakat will then download the file, check the sums, and replace itself.

20.2 Upgrading manually

Exakat is a PHP phar archive. Download the latest version from www.exakat.io and replace it.

20.3 Upgrading gremlin-server

Exakat installs the last version of gremlin at installation time. Usually, there is no need to upgrade the database when upgrading : changing the phar file is sufficient.

However, to enjoy the new features, or keep up to date, it is recommended to upgrade the gremlin server.

To upgrade gremlin-server, remove the old 'tinkergraph' folder from your installation. If exakat was installed following the installation instruction, this folder is located next to *exakat.phar*.

Then, run again the installation instruction, only for gremlin.

CHAPTER 21

Configuration

21.1 Common Behavior

21.1.1 General Philosophy

Exakat tries to avoid configuration as much as possible, so as to focus on working out of the box, rather than spend time on pre-requisite.

As such, it probably does more work, but that may be dismissed later, at reading time.

More configuration options appear with the evolution of the engine.

21.1.2 Reminder of precedences

The exakat engine read directives from three places :

1. The command line options
2. The .exakat.ini file at the root of the code
3. The config.ini file in the project directory
4. The exakat.ini file in the config directory
5. The default values in the code

The precedence of the directives is the same as the list above : command line options always have highest priority, config.ini files are in second, when command line are not available, and finally, the default values are read in the code.

Some of the directives are only available in the config.ini files.

21.1.3 Common Options

All options are the same, whatever the command provided to exakat. -f always means files, and -q always means quick.

Any option that a command doesn't understand is ignored.

Any option that is not recognized is ignored and reported (with visibility).

21.2 Engine configuration

Engine configuration is were the exakat engine general configuration are stored. For example, the php binaries or the neo4j folder are there. Engine configurations affect all projects.

21.2.1 Configuration File

The Exakat engine is configured in the 'config/exakat.ini' file.

This file is created with the 'doctor' command, or simply by copying another such file from another installation.

```
php exakat.phar doctor
```

When the doctor can't find the 'config/config.ini' file, it attempts to create one, with reasonable values. It is recommended to use this to create the exakat.ini skeleton, and later, modify it.

21.2.2 Available Options

Here are the currently available options in Exakat's configuration file : config/config.ini

Option	Description
graphdb	The graph database to use. Currently, it may be gsneo4j, or tinkergraph.
gsneo4j_host	The host to connect to reach the graph database, when using gsneo4j driver. The default value is 'localhost'
gsneo4j_host	The port to use on the host to reach the graph database, when using gsneo4j driver.. The default value is '8182
gsneo4j_folder	The folder where the code for the graph database resides, when using gsneo4j driver. The default value is 'tinl
tinkergraph_host	The host to connect to reach the graph database, when using tinkergraph driver. The default value is 'localhos
tinkergraph_port	The port to use on the host to reach the graph database, when using tinkergraph driver. The default value is '8
tinkergraph_folder	The folder where the code for the graph database resides, when using tinkergraph driver. The default value is
gsneo4jv3_host	The host to connect to reach the graph database, when using gsneo4j driver. The default value is 'localhost'
gsneo4jve_host	The port to use on the host to reach the graph database, when using gsneo4j driver.. The default value is '8182
gsneo4jv3_folder	The folder where the code for the graph database resides, when using gsneo4j driver. The default value is 'tinl
tinkergraphv3_host	The host to connect to reach the graph database, when using tinkergraph driver. The default value is 'localhos
tinkergraphv3_port	The port to use on the host to reach the graph database, when using tinkergraph driver. The default value is '8
tinkergraphv3_folder	The folder where the code for the graph database resides, when using tinkergraph driver. The default value is
project_rulesets	List of analysis rulesets to be run. The list may include extra rulesets that are not used by the default reports :
project_themes	Obsolete. Use the one above : project_rulesets
project_reports	The list of reports that can be produced when running 'project' command. This list may automatically add ext
token_limit	Maximum size of the analyzed project, in number of PHP tokens (excluding whitespace). Use this to avoid ru
php	Link to the PHP binary. This binary is the one that runs Exakat. It is recommended to use PHP 7.3, or 7.4. Th
php80	Path to the PHP 8.0.x binary. This binary is needed to test the compilation with the 8.0 series or if the analyze
php74	Path to the PHP 7.4.x binary. This binary is needed to test the compilation with the 7.4 series or if the analyze
php73	Path to the PHP 7.3.x binary. This binary is needed to test the compilation with the 7.3 series or if the analyze
php72	Path to the PHP 7.2.x binary. This binary is needed to test the compilation with the 7.2 series or if the analyze
php71	Path to the PHP 7.1.x binary. This binary is needed to test the compilation with the 7.1 series or if the analyze
php70	Path to the PHP 7.0.x binary. This binary is needed to test the compilation with the 7.0 series or if the analyze
php56	Path to the PHP 5.6.x binary. This binary is needed to test the compilation with the 5.6 series or if the analyze

Option	Description
php55	Path to the PHP 5.5.x binary. This binary is needed to test the compilation with the 5.5 series or if the analyze
php54	Path to the PHP 5.4.x binary. This binary is needed to test the compilation with the 5.4 series or if the analyze
php53	Path to the PHP 5.3.x binary. This binary is needed to test the compilation with the 5.3 series or if the analyze
php52	Path to the PHP 5.2.x binary. This binary is needed to test the compilation with the 5.2 series or if the analyze
php_extensions	List of PHP extensions to use when spotting functions, methods, constants, classes, etc. Default to 'all', which

Note : php** configuration may be either a valid PHP binary path, or a valid Docker image. The path on the system may be */usr/bin/php*, */usr/sbin/php80*, or */usr/local/Cellar/php71/7.1.30/bin/php*. The Docker configuration must have the form *abc/def:tag*. The image's name may be any value, as long as Exakat manage to run it, and get the valid PHP signature, with *php -v*. When using Docker, the docker server must be running.

21.2.3 Custom rulesets

Create custom rulesets by creating a 'config/themes.ini' directive files.

This file is a .INI file, build with several sections. Each section is the name of a ruleset : for example, 'mine' is the name for the ruleset below.

There may be several sections, as long as the names are distinct.

It is recommended to use all low-case names for custom rulesets. Exakat uses names with a first capital letter, which prevents conflicts. Behavior is undefined if a custom ruleset has the same name as a default ruleset.

```
['mine']
analyzer[] = 'Structures/AddZero';
analyzer[] = 'Performances/ArrayMergeInLoops';
```

The list of analyzer in the ruleset is based on the 'analyzer' array. The analyzer is identified by its 'shortname'. Analyzer shortname may be found in the documentation (*Rules* or within the Ambassador report). Analyzers names have a 'A/B' structure.

The list of available rulesets, including the custom ones, is listed with the *doctor* command.

21.3 Check Install

Once the prerequisite are installed, it is advised to run to check if all is found :

php exakat.phar doctor

After this run, you may edit 'config/config.ini' to change some of the default values. Most of the time, the default values will be OK for a quick start.

CHAPTER 22

Commands

22.1 List of commands :

22.2 anonymize

Read files, directory or projects, and produce a anonymized version of the code. Consistence between variables and names is preserved ($a is always replaced with the same name). PHP language structures, such as eval, isset or unset are preserved, though other native functions are not.

File structure is not preserved : all files are renamed, and the hiearchy is flattened in one folder. As such, code is probably un-runnable if it relies on inclusions.

Non-PHP files, non-lintable or files that produces one PHP token are ignored.

22.2.1 Command

```
exakat anonymize -p <project>
exakat anonymize -d <directory>
exakat anonymize -file <filename>
```

22.2.2 Options

Option	Req	Description
-p	No	Project name. Should be filesystem compatible (avoid /, : or) This takes into account <project> configuration
-d	No	Directory to anonymize. Results aree in <directory>.anon
-file	No	File to anonymize. Results are in <file>.anon
-v	No	Verbose mode

22.2.3 Tips

- *-R* is not compulsory : you may omit it, then, provide PHP files in the *projects/<name>/code* folder by the mean you want.

:: _baseline:

22.3 baseline

Baseline manage previous audits that may be used as a baseline for new audits.

A Baseline is a previous audit, that has already reviewed the code. It has identified issues and code. Later, after some code modification, a new audit is run. When we want to know the new issues, or the removed ones, it has to be compared to a baseline.

This is a help command, to help find the available values for various options.

22.3.1 Commands

Command	Description
list	List all available baselines. Default action
remove	Removes a baseline, using its name or its auto-id
save	Save the current audit, when it exists, as the last base, with the provided name.

:: _catalog:

22.4 catalog

Catalog list all available rulesets and reports with the current exakat.

This is a help command, to help find the available values for various options.

22.4.1 Options

Option	Req	Description
-json	No	Returns the catalog as JSON, for further processing.

:: _clean:

22.5 clean

Cleans the provided project of everything except the config.ini and the code.

This is a maintenance command, that removes all produced files and folder, and restores a project to its initial state.

22.5.1 Options

Option	Req	Description
-p	Yes	Project name. Should be an existing project.
-v	No	Verbose mode

:: _cleandb:

22.6 cleandb

Cleans the graph database.

This is a maintenance command, that removes all produced data and scripts, and restores the exakat database to its empty state.

By default, the database is cleaned with graph commands, letting the server do the cleaning.

The -Q option makes the same cleaning with a full restart of the server. This is cleaner, and faster if the database was big or in some instable state.

22.6.1 Options

Option	Req	Description
-Q	No	Cleans the database by restarting it, and removing files.
-stop	No	Stops gremlin server
-start	No	Starts gremlin server, without removing files.
-restart	No	Restarts gremlin server, without removing files.
-v	No	Verbose mode

:: _doctor:

22.7 doctor

Check the current installation of Exakat.

22.7.1 Command

```
exakat doctor
```

22.7.2 Results

```
PHP :
    version              : 7.0.1
    curl                 : Yes
    sqlite3              : Yes
    tokenizer            : Yes

java :
    installed            : Yes
    type                 : Java(TM) SE Runtime Environment (build 1.8.0_40-b25)
    version              : 1.8.0_40
    $JAVA_HOME           : /Library/Java/JavaVirtualMachines/jdk1.8.0_40.jdk/Contents/
↪Home

neo4j :
    version              : Neo4j 2.2.6
    port                 : 7474
    authentication       : Not enabled (Please, enable it)
    gremlinPlugin        : Configured.
    gremlinJar           : neo4j/plugins/gremlin-plugin/gremlin-java-2.7.0-SNAPSHOT.
↪jar
    scriptFolder         : Yes
    pid                  : 20895
    running              : Yes
    running here         : Yes
    gremlin              : Yes
    $NEO4J_HOME          : /Users/famille/Desktop/analyze/neo4j

folders :
    config-folder        : Yes
    config.ini           : Yes
    projects folder      : Yes
    progress             : Yes
    in                   : Yes
    out                  : Yes
    projects/test        : Yes
    projects/default     : Yes
    projects/onepage     : Yes

PHP 5.2 :
    configured           : No
```

(continues on next page)

(continued from previous page)

```
PHP 5.3 :
    configured          : Yes
    installed           : Yes
    version             : 5.3.29
    short_open_tags     : Off
    timezone            : Europe/Amsterdam
    tokenizer           : Yes
    memory_limit        : -1

PHP 5.4 :
    configured          : Yes
    installed           : Yes
    version             : 5.4.45
    short_open_tags     : Off
    timezone            : Europe/Amsterdam
    tokenizer           : Yes
    memory_limit        : 384M

PHP 5.5 :
    configured          : Yes
    installed           : Yes
    version             : 5.5.30
    short_open_tags     : Off
    timezone            : Europe/Amsterdam
    tokenizer           : Yes
    memory_limit        : -1

PHP 5.6 :
    configured          : /usr/local/sbin/php56
    installed           : Yes
    version             : 5.6.16
    short_open_tags     : Off
    timezone            : Europe/Amsterdam
    tokenizer           : Yes
    memory_limit        : -1

PHP 7.0 :
    configured          : Yes
    version             : 7.0.1
    short_open_tags     : Off
    timezone            :
    tokenizer           : Yes
    memory_limit        : -1

PHP 7.1 :
    configured          : Yes
    version             : 7.1.0-dev
    short_open_tags     : Off
    timezone            :
    tokenizer           : Yes
    memory_limit        : 128M

git :
    installed           : Yes
    version             : 2.7.0
```

(continues on next page)

(continued from previous page)

```
hg :
    installed           : Yes
    version             : 3.6.3

svn :
    installed           : Yes
    version             : 1.9.3

bzr :
    installed           : No
    optional            : Yes

composer :
    installed           : Yes
    version             : 1.0.0-alpha11

wget :
    installed           : Yes
    version             : GNU Wget 1.17.1 built on darwin15.2.0.

zip :
    installed           : Yes
    version             : 3.0
```

Tips

- The *PHP* section is the PHP binary used to run Exakat.
- The *PHP x.y* sections are the PHP binaries used to check the code.
- Optional installations (such as svn, zip, etc.) are not necessarily reported if they are not installed.

22.7.3 Options

Option	Req	Description
-p	No	Displays the project-specific configuration. Otherwise, only displays general configuration.
-json	No	Displays the project-specific configuration in json format, to stdout
-v	No	Verbose mode : include helpers configurations
-q	No	Quiet mode : runs doctor, and install checks, but displays nothing. This is useful to automate installation finalization

:: _help:

22.8 help

Displays the help section.

```
php exakat.phar help
```

22.8.1 Results

This displays :

```
[Usage] :   php exakat.phar init -p <Project name> -R <Repository>
            php exakat.phar project -p <Project name>
            php exakat.phar doctor
            php exakat.phar version
```

:: _init:

22.9 init

Initialize a new project.

22.9.1 Command

```
exakat init -p <project> [-R vcs_url] [-git|-svn|-bzr|-hg|-composer|-symlink|-copy|-
↪tgz|-7z|-zip] [-v] [-D]
```

22.9.2 Options

Option	Req	Description
-p	Yes	Project name. Should be filesystem compatible (avoid /, : or)
-R	No	URL to the VCS repository. Anything compatible with the expected VCS.
-git	No	Use git client (also, default value if no clue is given in the VCS URL)
-svn	No	Use SVN client
-bzr	No	Use Bazar client
-hg	No	Use Mercurial (hg) client
-composer	No	Use Composer client
-symlink	No	-R path is symlinked. Directory is never accessed for writing.
-copy	No	-R path is recursively copied.
-zip	No	-R is a ZIP archive, local or remote
-tgz	No	-R is a .tar.gzip archive, local or remote
-tbz	No	-R is a .tar.bz2 archive, local or remote
-rar	No	-R is a .rar archive, local or remote
-7z	No	-R is a .7z archive, local or remote
-v	No	Verbose mode
-D	No	First erase any pre-existing project with the same name

22.9.3 Tips

- *-R* is not compulsory : you may omit it, then, provide PHP files in the *projects/<name>/code* folder by the mean you want.
- Default VCS used is git.
- *-D* removes any previous project before doing the init.

- Archives (zip, tar.gz, tar.bz, 7z, rar, etc.) depends on external tools to unpack them. They depends on PHP to reach the file, locally or remotely.

22.9.4 Examples

```
# Clone Exakat with Git
php exakat.phar init -p exakat -R https://github.com/exakat/exakat.git

# Download Spip with Zip
php exakat init -p spip2 -zip -R http://files.spip.org/spip/stable/spip-3.1.zip

# Download PHPMyadmin,
php exakat.phar init -p pma2 -tgz -R https://files.phpmyadmin.net/phpMyAdmin/4.6.4/
↪phpMyAdmin-4.6.4-all-languages.tar.gz

# Make a local copy of PHPMyadmin,
php exakat.phar init -p copyProject -copy -R projects/phpmyadmin/code/

# Make a local symlink with the local webserver,
php exakat.phar init -p symlinkProject -symlink -R /var/www/public_html
```

:: _project:

22.10 project

Runs a new analyze on a project.

The results of the analysis are available in the *projects/<name>/* folder. *report* and *faceted* are two HTML reports.

22.10.1 Command

```
exakat project -p <project> [-v]
```

22.10.2 Options

Option	Req	Description
-p	Yes	Project name. Should be filesystem compatible (avoid /, : or)
-v	No	Verbose mode

:: _remove:

22.11 remove

Destroy a project. All code source, configuration and any results from exakat are destroyed.

22.11.1 Command

```
exakat remove -p <project> [-v]
```

22.11.2 Options

Option	Req	Description
-p	Yes	Project name. Should be filesystem compatible (avoid /, : or)
-v	No	Verbose mode

:: _remove:

22.12 show

Displays the the full command line to create an exakat project.

22.12.1 Command

```
exakat show -p <project>
```

22.12.2 Options

Option	Req	Description
-p	Yes	Project name. Should be filesystem compatible (avoid /, : or)

:: _report:

22.13 report

Produce a report for a project.

Reports may be produced as soon as exakat has reach the phase of 'analysis'. If the analysis phase hasn't finished, then some results may be unavailable. Run report again later to get the full report. For example, the 'Uml' report may be run fully as soon as exakat is in analysis phase.

It is possible to extract a report even after the graph database has been cleaned. This allows running several projects one after each other, yet have access to several reports.

22.13.1 Command

```
exakat report -p <project> -format <Format> [-file <file>] [-v]
```

22.13.2 Options

Option	Req	Description
-p	Yes	Project name. Should be filesystem compatible (avoid /, : or)
-v	No	Verbose mode
-format	No	Which format to extract. Available formats : Devoops, Faceted, FacetedJson, Json, OnepageJson, Text, Uml, Xml Default is 'Text'
-file	No	File or directory name for the report. Adapted file extension is added. Report is located in the projects/<project>/ folder Default is 'stdout', but varies with format.
-T	No	Ruleset's results. All the analyses in this ruleset are reported. Note that the report format may override this configuration : for example Ambassador manage its own list of analyses. Uses this with Text format. Has priority over the -P option
-P	No	Analyzer's results. Only one analysis's is reported. Note that the report format may override this configuration : for example Ambassador manage its own list of analyses. Uses this with Text format. Has lower priority than the -T option

22.13.3 Report formats

All reports are detailed in the ref:*Reports <reports>* section.

Report	Description
Amabassador	HTML format, with all available reports in one compact format.
Devoops	HTML format, deprecated.
Json	JSON format.
Text	Text format. One issue per line, with description, file, line.
Codesniffer	Text format, similar to Codesniffer report style.
Uml	Dot format. All classes/interfaces/traits hierarchies, and grouped by name- spaces.
Xml	XML format.
All	All availble format, using default naming

:: _update:

22.14 update

Update the code base of a project.

22.14.1 Command

```
exakat update -p <project> [-v]
```

22.14.2 Options

Option	Req	Description
-p	Yes	Project name. Should be filesystem compatible (avoid /, : or)
-v	No	Verbose mode

:: _upgrade:

22.15 upgrade

Upgrade exakat itself. By default, this command only checks for the availability of a new version : it doesn't upgrade immediately.

Use -u option to actually replace the current phar archive.

Use -version option to downgrade or upgrade to a specific version.

In case the upgrade command file, you may also download manually the *.phar* from the exakat.io website : www.exakat.io. Then replace the current version with the new one.

22.15.1 Command

```
exakat upgrade
```

22.15.2 Options

Option	Req	Description
-u	Yes	Actually upgrades exakat. Without it, it is a dry run.
-version	No	Select a specific Exakat version and update to it. By default, it upgrades to the latest version, as published on the https://www.exakat.io/ site. Example value : 1.8.8

22.16 Install

Install exakat's graph dependency. This command is an integrated installation script, and it is only accessible once the .phar is downloaded locally.

22.16.1 Command

```
mkdir exakat
cd exakat

// Download exakat.phar, like this, or any other valid means
curl -o exakat.phar https://www.exakat.io/versions/index.php?file=latest
exakat.phar upgrade
```

22.16.2 Options

Option	Req	Description
-u	Yes	Actually upgrades exakat. Without it, it is a dry run.
-version	No	Select a specific Exakat version and update to it. By default, it upgrades to the latest version, as published on the https://www.exakat.io/ site. Example value : 1.8.8

CHAPTER 23

Frequently Asked Questions

23.1 Summary

23.2 I need special command to get my code

If Exakat has no documented method to reach your code, you may use this process :

```
php exakat.phar init -p <your project name>
cd ./projects/<your project name>
mkdir code
// here, do whatever it takes to put all your code in 'code' folder
cd -
php exakat.phar project -p <your project name>
```

Send a message on Github.com/exakat/exakat to mention your specific method.

23.3 Can I checkout that branch?

Currently (Version 0.12.2), there is no way to request a tag or a branche or a revision when cloning the code.

The best way is to reach the 'code' folder, and make the change there. Unless with 'init' or 'update', exakat doesn't make any change to the code.

```
php exakat.phar init -p myProject -R url://my/git/repository
cd ./projects/myProject/code
git branch notMasterBranch
cd -
php exakat.phar project -p myProject
```

23.4 Can I clone with my ssh keys?

When using git, or any vcs, the current shell user's SSH keys may be used to access the repository. When using a remote installation, or a docker image, the keys won't be accessible.

The fallback solution is to init an empty project, clone the code from the Shell (with the keys), and then run project.

```
php exakat.phar init -p myProject
cd ./projects/myProject
git clone url://myprivate/git/repository code
cd -
php exakat.phar project -p myProject
```

23.5 After init, my project has no code!

Check in the projects/<name>/config.ini file : if values were provided, you'll find them there.

In case the code was not found during init, then do the following :

:: cd projects/<name>/ git clone ssh://project/URL code cd - php exakat.phar files -p <name>

If you're using some other method than git, then just collect the code in a 'code' folder in the <name> project and run the 'files' command.

The 'init' command doesn't overwrite an existing project : if the *code* folder is missing, you should add it manually, or remove the project with *remove* command, and use *init* again.

23.6 The project is too big

There is a soft limit in config/exakat.ini, called 'token_limit' that initially prevents analysis of projects over 1 million tokens. That's roughly 125k LOC, more than most code source.

If you need to run exakat on larger sources, you may change this value to make it as large as possible. Then, the physical capacities of the machine, specially RAM, will be the actual limit.

It may be interesting to 'ignore_dir[]', from projects/<name>/config.ini.

23.7 Java Out Of Memory Error

By default, java is allowed to run with 512mb of RAM. That may be too little for the code being studied.

Set the environment variable $JAVA_OPTIONS to give larger quantities of RAM. For example : 'export JAVA_OPTIONS='-Xms1024m -Xmx6096m'; or 'setenv JAVA_OPTIONS='-Xms1024m -Xmx6096m'

Xms is the memory allocation at start, and Xmx is the maximum allocation. With some experimentation, 6G handles the largest

23.8 How can I run a very large project?

Here are a few steps you can try when running exakat on a very large project.

- Update project/<name>/config.ini, and use ignore_dirs[] and include_dirs[] to exclude as much code as possible. Notably, frameworks, data in PHP files, tests, cache, translations, etc.
- Set environment variable $JAVA_OPTIONS to large quantities of RAM : JAVA_OPTIONS='-Xms1024m -Xmx6096m';
- Check that your installation is running with 'gsneo4j' and not 'tinkergraph', in config/exakat.ini.

23.9 Does exakat runs on Java 8?

Exakat itself runs with PHP 7.0+. Exakat runs with a gremlin database : gremlin-server 3.2.x is supported, which runs on Java 8.

Java 9 is experimental, and is being tested. Java 7 used to be working, but is not supported anymore : it may still work, though.

23.10 Where can I find the report

Reports are available after running at least the following commands :

```
php exakat.phar init -p <your project name> -R <code source repo>
php exakat.phar project -p <your project name>
```

The default report is the HTML report, called Ambassador. You'll find it in ./projects/<your project name>/report.

Other reports, build with 'report' command, will also be saved there, with different names.

23.11 Exakat only produces the default report

After a default installation, Exakat builds the Ambassador report. If you want another report, for example Migration80, you have to request it.

```
php exakat.phar report -p <your project name> --format Migration80  -v
```

You may also access other reports, such as Text, which are always available after an audit.

The 'report' command aborts the report build when insufficient rules have been run. At that point, you must configure the report or the rules, in the projects or the server, and run the audit again.

23.12 Can I run exakat on local code?

There are several ways to do that : use symbolic links, make a copy of the source.

```
php exakat.phar init -p <your project name> -R <path/to/the/code> -symlink
php exakat.phar init -p <your project name> -R <path/to/the/code> -copy
php exakat.phar init -p <your project name> -R <path/to/the/code> -git
```

Symlink will branch exakat directly into the code; -copy makes a copy of the code (this means the code will never be updated without manual intervention); git (or other vcs) may also be used with local repositories.

Exakat do not modify any existing source code : it only access it for reading purpose, then works on a separated database. As a defensive security measure, we suggest that exakat should work on a read-only copy of the code.

23.13 Can I ignore a dir or a file?

Yes. After initing a project, open the projects/<project name>/config.ini file, and update the ignore_dir line. For example, to ignore a behat test folder, and to ignore any file called 'license' :

```
ignore_dirs[] = '/behat/';
ignore_dirs[] = 'license';
```

You may also include files, by using the include_dir[] line. Including files is processed after ignoring them, so you may include files in folders that were previously ignored.

23.14 *Can I audit only one folder in vendor?*

You can use ignore_dirs to exclude everything in the source tree, then use include_dirs to include specific folders.

:: # exclude everything ignore_dirs[] = '/';

    # include intended folder include_dirs[] = '/vendor/exakat';

23.15 Can I run Exakat with PHP 5?

It is recommended to run exakat with PHP 7.4 or even 8.0. PHP 7.3 is still possible, though not supported. PHP 7.2 and below won't work (we checked).

Note that you may test your code on PHP 5.x, while running Exakat on PHP 7.4. There are 2 distinct configuration options in Exakat. 'php' is the path to the PHP binary that runs Exakat : this one should be PHP 7.0+. 'phpxx' are the path to the PHP helpers, that are used to tokenized and lint the target PHP code. This is where PHP 5.x may be configured.

```
; where and which PHP executable are available
php   = /usr/local/sbin/php74

php52 =
php53 = /usr/local/sbin/php53
php54 =
php55 =
php56 =
php70 =
php71 =
php72 =
php73 =
php74 =
php80 =
php81 =
```

Above is an example of a exakat configuration file, where Exakat is run with PHP 7.1 and process code with PHP 5.3.

23.16 I get the error 'The executable 'ansible-playbook' Vagrant is trying to run was not found'

This error is displayed when the host machine doesn't have Ansible installed. Install ansible, and try again to provision.

23.17 Can I run exakat on Windows?

Currently, Windows is not supported, though it might be some day.

Until then, you may run Exakat with Vagrant, or with Docker.

23.18 Does exakat send my code to a central server?

When run from the sources, Exakat has everything it needs to fulfill its mission. There is no central server that does the job, and requires the transmission of the code.

When running an audit on the Saas service of Exakat, the code is processed on our servers.

23.19 "cat: write error: Broken pipe" : is it bad?

Exakat currently runs some piped commands, with xargs so as to make some operations parallel. When the following command ends up before the reading all the data from the first command, such a warning is emitted.

It has no impact on exakat's processing of the code.

See also cat: write error: Broken pipe.

23.20 *Require a [gremlin]Argument*

Running an audit (project command) leads to an error message such as this one :

:: 2/2 [==>] 100.00% 00:00:00

Error : The request message was parseable, but the arguments supplied in the message were in conflict or incomplete. Check the message format and retry the request. : A message with an [eval] op code requires a [gremlin] argument.

========== SERVER TRACE ========== array ( ) ==

on file phar:///exakat-2.1.9/exakat.phar/vendor/brightzone/gremlin-php/src/Connection.php on line 847

This happens when exakat couldn't stop the gremlin database. You should take it down manually, then restart the audit. No version update necessary.

Get the process ID with the following command, and then, kill it.

:: ps aux | grep gsneo4jv3.3.4 ps aux | grep gremlin

CHAPTER 24

Glossary

Here is a list of words, commonly used when using Exakat, with their definitions and their synonyms.

- ***A***

  Analysis An *Analysis* is a pattern that may be detected in the code. The analysis has a human-redable description, and a specific implementation.

- ***D***

  Dump The phase of execution, which prepare the results from the graph database to the data storage for reports.

- ***I***

  Issue The result of an analysis, when an analysis is applied to a code.

- ***L***

  Load The phase of execution, which loads the source code into the central database.

- ***R***

  Report A set of issues, gathered into a consistent format, after running the analysis on the code. A report may include multiple rulesets, and use various format, such as HTML, JSON or Text.

  Rule A synonym for Analysis. This may be more descriptive, and less related to implementation.

  Ruleset A consistent group of analysis, recognizable with a specific name.

CHAPTER 25

Annex

- Credits
- Contribute
- External links

25.1 Credits

The following people helped in the making of Exakat : installing, coding, suggesting, using, documenting, reporting bugs, pushing us to be better.

- (Buck / Leon)
- (Jent / Jean)
- Gérard Ernaelsten
- Philippe Gamache
- Cyrille Granval
- Eshin Kunishima
- Alexis Van Glasow

25.2 Contribute

Exakat is an Open Source project. It is also organized to collect common knowledge and encode it in its databases.

Here are some suggestions of help you may provide to enhance your own usage of Exakat :

- Suggest PHP extensions that are missing in the list of supported extensions (see *Annex*)
- Suggest new analysis, with examples of target code, and examples of good code

- Suggest missing external services
- Suggest reference article for the documentation, in the section 'See also'
- Suggestion application that may be added to the corpus of codes that we use to validate the analysis
- Provide new names and adjectives for the audit names. We like to include any first name of community members, and non-derogatory adjectives.
- Report installation or usage problems
- Report ambiguity in reports and their documentation
- Suggest interesting Coding reference, like Object Calisthenics, PSR, East-Oriented Programming, etc.
- Translate the documentation into other languages
- Suport Exakat on Windows or other OS
- Recommend article for code conception to be added in the docs
- Suggest public code source for review

Visit us on the [github repository](https://github.com/exakat/php-static-analysis-tools), or the [slack channel](https://www.exakat.io/slack-invitation/).

25.3 External links

List of external links mentionned in this documentation.

- #QuandLeDevALaFleme
- $_ENV
- \protect\T1\textdollardocs{[}extension_page{]}
- \protect\T1\textdollardocs{[}home_page{]}
- $GLOBALS
- $HTTP_RAW_POST_DATA variable
- **'$name <$service->homepage>'_**
- **'$this->clearphp <https://github.com/dseguy/clearPHP/tree/master/rules/$this->clearphp.md>'_**
- $x
- ' . $ini['name'] . '
- **'' . $library->name . ' <' . $library->homepage . '>'_**
- ' . $r[2] . $r[3] . '
- 10 GitHub Security Best Practices
- 1003.1-2008 - IEEE Standard for Information Technology - Portable Operating System Interface (POSIX(R))
- 7z
- ::class
- @deprecated
- [blog] array_column()
- [CVE-2017-6090]

- [HttpFoundation] Make sessions secure and lazy #24523
- **‘]*?>‘_**
- __autoload
- __get performance questions with PHP
- __set
- A PHP extension for Redis
- About circular references in PHP
- Add array_key_exists to the list of specialy compiled functions
- Add Exakat To Your CI Pipeline
- Allow a trailing comma in function calls
- Alpine Linux
- Alternative PHP Cache
- Alternative syntax
- Ambassador
- Anonymous functions
- APCU
- Argon2 Password Hash
- Arithmetic Operators
- Aronduby Dump
- array
- Array
- Array Functions
- array_fill_keys
- array_filter
- array_key_exists() with objects
- array_map
- array_merge
- array_search
- array_slice
- array_unique
- ArrayAccess
- Arrays
- Arrays syntax
- Arrow functions
- assert
- Assignation Operators

- Autoloading Classe
- Autoloading Classes
- Avoid Else, Return Early
- Avoid nesting too deeply and return early (part 1)
- Avoid option arrays in constructors
- Avoid optional services as much as possible
- Backward incompatible changes
- Backward incompatible changes PHP 7.0
- basename
- Basics
- bazaar
- BC Math Functions
- Benoit Burnichon
- Bitmask Constant Arguments in PHP
- Bitwise Operators
- Brandon Savage
- browscap
- Bug #50887 preg_match , last optional sub-patterns ignored when empty
- Bzip2 Functions
- Cairo Graphics Library
- Calendar Functions
- Callback / callable
- Callbacks / callables
- Can you spot the vulnerability? (openssl_verify)
- Cant Use Return Value In Write Context
- Carbon
- Carnage
- cat: write error: Broken pipe
- Change the precedence of the concatenation operator
- Changes to variable handling
- Class Abstraction
- Class Constant
- Class Constants
- class_alias
- Classes abstraction
- Classes Abstraction

- Closure class
- Closure::bind
- Cmark
- Codeigniter
- COM and .Net (Windows)
- compact
- Comparison Operators
- composer
- Concrete 5
- Conflict resolution
- Constant definition
- Constant Scalar Expressions
- constant()
- Constants
- Constructors and Destructors
- Cookies
- count
- Courier Anti-pattern
- Covariant Returns and Contravariant Parameters
- crc32()
- Cross-Site Scripting (XSS)
- crypt
- Cryptography Extensions
- CSPRNG
- Ctype funtions
- curl
- Curl for PHP
- curl_version
- CVS
- CWE-484: Omitted Break Statement in Switch
- CWE-625: Permissive Regular Expression
- Cyrus
- Data filtering
- Data structures
- Database (dbm-style) Abstraction Layer
- Date and Time

- DCDFLIB
- Dead Code: Unused Method
- declare
- Declare
- define
- define
- Dependency Injection Smells
- Deprecate and remove continue targeting switch
- Deprecate and remove INTL_IDNA_VARIANT_2003
- Deprecate curly brace syntax
- Deprecated features in PHP 5.4.x
- Deprecated features in PHP 5.5.x
- Deprecated features in PHP 7.2.x
- Deprecation allow_url_include
- Deprecations for PHP 7.2
- Deprecations for PHP 7.4
- Destructor
- DIO
- Dir predefined constants
- directive error_reporting
- Directly calling __clone is allowed
- dirname
- dl
- Do your objects talk to strangers?
- Docker
- Docker image
- Document Object Model
- Don't pass this out of a constructor
- Don't repeat yourself (DRY)
- Don't turn off CURLOPT_SSL_VERIFYPEER, fix your PHP configuration
- dotdeb instruction
- Double quoted
- download
- Drupal
- Dynamically Access PHP Object Properties with $this
- E_WARNING for invalid container read array-access

- Eaccelerator
- elseif/else if
- empty
- Empty Catch Clause
- Empty interfaces are bad practice
- empty()
- Enchant spelling library
- Ereg
- Error Control Operators
- Error reporting
- Escape sequences
- Ev
- eval
- Event
- Exakat
- Exakat cloud
- Exakat SAS
- exakat/exakat
- Exception::__construct
- Exceptions
- Exchangeable image information
- Execution Operators
- EXP30-C. Do not depend on the order of evaluation for side effects
- expect
- explode
- ext-async repository
- ext-http
- ext/ast
- ext/gender manual
- ext/hash extension
- ext/hrtime manual
- ext/inotify manual
- ext/leveldb on Github
- ext/lua manual
- ext/mbstring
- ext/memcached manual

- ext/OpenSSL
- ext/readline
- ext/recode
- ext/SeasLog on Github
- ext/sqlite
- ext/sqlite3
- ext/uopz
- ext/varnish
- ext/zookeeper
- Extension Apache
- extension FANN
- extension mcrypt
- extract
- Ez
- Factory (object-oriented programming)
- FAM
- FastCGI Process Manager
- FDF
- ffmpeg-php
- file_get_contents
- filesystem
- Filinfo
- Final Keyword
- Firebase / Interbase
- Flag Argument
- FlagArgument
- Floating point numbers
- Floats
- Fluent Interfaces in PHP
- fopen
- foreach
- Foreign Function Interface
- Frederic Bouchery
- From assumptions to assertions
- FuelPHP
- Function arguments

- Functions
- Gearman on PHP
- Generalize support of negative string offsets
- Generator delegation via yield from
- Generator Syntax
- Generators overview
- GeoIP
- get_class
- get_object_vars script on 3V4L
- Gettext
- Git
- Github Action
- Github.com/exakat/exakat
- global namespace
- GMP
- Gnupg Function for PHP
- Goto
- graphviz
- Gremlin server
- Group Use Declaration RFC
- GRPC
- Handling file uploads
- Hardening Your HTTP Security Headers
- hash
- HASH Message Digest Framework
- hash_algos
- hash_file
- Heredoc
- Holger Woltersdorf
- How many parameters is too many ?
- How to fix Headers already sent error in PHP
- How to pick bad function and variable names
- htmlentities
- htmlspecialchars
- https://hub.docker.com/r/exakat/exakat-ga
- https://www.exakat.io/

- https://www.exakat.io/versions/index.php?file=latest
- IBM Db2
- Iconv
- iconv()
- ICU
- Ideal regex delimiters in PHP
- idn_to_ascii
- IERS
- igbinary
- IIS Administration
- Image Processing and GD
- Imagick for PHP
- IMAP
- Implement ZEND_ARRAY_KEY_EXISTS opcode to speed up array_key_exists()
- implode
- In a PHP5 class, when does a private constructor get called?
- in_array()
- include
- include_once
- Incrementing/Decrementing Operators
- Insecure Transportation Security Protocol Supported (TLS 1.0)
- Instanceof
- Integer overflow
- Integer Syntax
- Integers
- Interfaces
- Internal Constructor Behavior
- Is it a bad practice to have multiple classes in the same file?
- Isset
- Isset Ternary
- It is the 31st again
- iterable pseudo-type
- Iterables
- Joomla
- json_decode
- Judy C library

- Kafka client for PHP
- Kerberos V
- Lapack
- Laravel
- Late Static Bindings
- Least Privilege Violation
- libeio
- libevent
- libmongoc
- libuuid
- libxml
- Lightweight Directory Access Protocol
- list
- List of function aliases
- List of HTTP header fields
- List of HTTP status codes
- List of Keywords
- List of other reserved words
- List of TCP and UDP port numbers
- list() Reference Assignment
- Logical Expressions in C/C++. Mistakes Made by Professionals
- Logical Operators
- Loosening Reserved Word Restrictions
- lzf
- Magic Constants
- Magic Hashes
- Magic Method
- Magic Methods
- Magic methods
- mail
- Mail related functions
- Marco Pivetta tweet
- match
- Match expression V2
- Match()
- Math predefined constants

- Mathematical Functions
- mb_encoding_detect
- mb_str_split
- Mbstring
- mcrypt_create_iv()
- MD5
- Media Type
- Memcache on PHP
- mercurial
- Method overloading
- mhash
- Microsoft SQL Server
- Microsoft SQL Server Driver
- Migration80
- Ming (flash)
- MongoDB driver
- move_uploaded_file
- msgpack for PHP
- MySQL Improved Extension
- mysqli
- Named Arguments
- Ncurses Terminal Screen Control
- Negative architecture, and assumptions about code
- Nested Ternaries are Great
- Net SNMP
- net_get_interfaces
- New Classes and Interfaces
- New custom object serialization mechanism
- New global constants in 7.2
- New global constants in 7.4
- New object type
- Newt
- No Dangling Reference
- Nowdoc
- Null and True
- Null Coalescing Operator

- Null Object Pattern
- Nullable types
- Object Calisthenics, rule # 2
- Object Calisthenics, rule # 5
- Object cloning
- Object Inheritance
- Object Interfaces
- Object interfaces
- Objects and references
- ODBC (Unified)
- OPcache functions
- opencensus
- OpennSSL [PHP manual]
- openssl_random_pseudo_byte
- Operator Precedence
- Operators Precedence
- Optimization: How I made my PHP code run 100 times faster
- Optimize array_unique()
- Option to make json_encode and json_decode throw exceptions on errors
- Oracle OCI8
- original idea
- Original MySQL API
- Output Buffering Control
- Overload
- pack
- Packagist
- parent
- Parsekit
- Parsing and Lexing
- Passing arguments by reference
- Passing by reference
- Password Hashing
- Password hashing
- Pattern Modifiers
- PCOV
- PCRE

- PEAR
- pecl crypto
- PECL ext/xxtea
- pg_last_error
- Phalcon
- phar
- PHP 7 performance improvements (3/5): Encapsed strings optimization
- PHP 7.0 Backward incompatible changes
- PHP 7.0 Removed Functions
- PHP 7.1 no longer converts string to arrays the first time a value is assigned with square bracket notation
- PHP 7.2's "switch" optimisations
- PHP 7.2's switch optimisations
- PHP 7.3 Removed Functions
- PHP 7.3 UPGRADE NOTES
- PHP 7.4 Removed Functions
- PHP 8: Constructor property promotion
- PHP
- PHP class name constant case sensitivity and PSR-11
- PHP Classes containing only constants
- PHP Clone and Shallow vs Deep Copying
- PHP Constants
- PHP Data Object
- PHP Decimal
- PHP extension for libsodium
- PHP gmagick
- PHP Options And Information
- PHP Options/Info Functions
- PHP return(value); vs return value;
- PHP RFC: Add Stringable interface
- PHP RFC: Allow a trailing comma in function calls
- PHP RFC: Allow abstract function override
- PHP RFC: Allow trailing comma in parameter list
- PHP RFC: Arrays starting with a negative index
- PHP RFC: Arrow Functions
- PHP RFC: Convert numeric keys in object/array casts
- PHP RFC: Deprecate and Remove Bareword (Unquoted) Strings

- PHP RFC: Deprecate left-associative ternary operator
- PHP RFC: Deprecations for PHP 7.2 : Each()
- PHP RFC: Deprecations for PHP 7.4
- PHP RFC: get_debug_type
- PHP RFC: is_countable
- PHP RFC: Nullsafe operator
- PHP RFC: Numeric Literal Separator
- PHP RFC: Scalar Type Hints
- PHP RFC: Shorter Attribute Syntax
- PHP RFC: str_contains
- PHP RFC: Syntax for variadic functions
- PHP RFC: Unicode Codepoint Escape Syntax
- PHP RFC: Union Types 2.0
- PHP RFC: Variable Syntax Tweaks
- PHP Tags
- PHP why pi() and M_PI
- php-ext-wasm
- php-vips-ext
- php-zbarcode
- PHP: When is /tmp not /tmp?
- phpsdl
- PhpStorm 2020.3 EAP #4: Custom PHP 8 Attributes
- phpstorm-stubs/meta/attributes/Immutable.php
- plantuml
- PMB
- PostgreSQL
- Predefined Constants
- Predefined Exceptions
- Predefined Variables
- preg_filter
- Prepare for PHP 7 error messages (part 3)
- Prepared Statements
- printf
- Process Control
- proctitle
- Properties

- Property overloading
- Pspell
- PSR-11 : Dependency injection container
- PSR-13 : Link definition interface
- PSR-16 : Common Interface for Caching Libraries
- PSR-3 : Logger Interface
- PSR-3
- PSR-6 : Caching
- Putting glob to the test
- RabbitMQ AMQP client library
- rar
- Rar archiving
- Refactoring code
- References
- Reflection
- Reflection export() methods
- Regular Expressions (Perl-Compatible)
- resources
- return
- Return Inside Finally Block
- Return Type Declaration
- Returning values
- RFC 7159
- RFC 7230
- RFC 822 (MIME)
- RFC 959
- RFC : Arrow functions
- RFC Preload
- RFC: Return Type Declarations
- runkit
- Salted Password Hashing - Doing it Right
- Scalar type declarations
- Scope Resolution Operator (::)
- Secure Hash Algorithms
- Semaphore, Shared Memory and IPC
- Session

- session_regenerateid()
- Sessions
- Set-Cookie
- set_error_handler
- setcookie
- setlocale
- shell_exec
- SimpleXML
- Single Function Exit Point
- SOAP
- Sockets
- Specification pattern
- Sphinx Client
- Spread Operator in Array Expression
- Spread Operator in Array Expression
- sqlite3
- SQLite3::escapeString
- SSH2 functions
- Standard PHP Library (SPL)
- Static anonymous functions
- Static Keyword
- str_contains
- Strict typing
- Stricter type checks for arithmetic/bitwise operators
- String functions
- Strings
- strip_tags
- strpos not working correctly
- strtr
- Structuring PHP Exceptions
- Structuring PHP Exceptions session
- Subpatterns
- substr
- Suhosin.org
- Sun, iPlanet and Netscape servers on Sun Solaris
- Superglobals

- Supported PHP Extensions
- Supported Protocols and Wrappers
- SVM
- Svn
- Swoole
- Symfony
- Syntax
- Ternary Operator
- tetraweb/php
- Text
- The Basics
- The basics of Fluent interfaces in PHP
- The Closure Class
- The Definitive 2019 Guide to Cryptographic Key Sizes and Algorithm Recommendations
- The Linux NIS(YP)/NYS/NIS+ HOWTO
- The list function & practical uses of array destructuring in PHP
- The main PPA for PHP (8.0, 7.4, 7.3, 7.2, 7.1, 7.0, 5.6)
- Throw Expression
- Throwable
- Tidy
- tokenizer
- tokyo_tyrant
- trader (PECL)
- Trailing Comma In Closure Use List
- Trailing Commas In List Syntax
- Traits
- Traversable
- trigger_error
- trim
- Tutorial 1: Let's learn by example
- Type array
- Type Casting
- Type Declaration
- Type declarations
- Type declarations
- Type Declarations

- Type hinting for interfaces
- Type Juggling
- Type juggling
- Type Juggling Authentication Bypass Vulnerability in CMS Made Simple
- Type Operators
- Typed Properties 2.0
- Typo3
- Unbinding $this from non-static closures
- Understanding Dependency Injection
- Unicode block
- Unicode spaces
- Uniform Resource Identifier
- unserialize()
- unset
- Unset casting
- UPGRADING 7.3
- UPGRADING PHP 8.0
- Use of Hardcoded IPv4 Addresses
- Using namespaces: Aliasing/Importing
- Using namespaces: fallback to global function/constant
- Using non-breakable spaces in test method names
- Using single characters for variable names in loops/exceptions
- Using static variables
- V8 Javascript Engine
- Vagrant file
- Variable basics
- Variable functions
- Variable scope
- Variable Scope
- Variable variables
- Variable-length argument lists
- Variables
- Visibility
- Vladimir Reznichenko
- Void functions
- Warn when counting non-countable types

- Wddx on PHP
- Weak references
- What are the best practices for catching and re-throwing exceptions?
- What's all this 'immutable date' stuff, anyway?
- When empty is not empty
- When to declare classes final
- Why 777 Folder Permissions are a Security Risk
- Why does PHP 5.2+ disallow abstract static class methods?
- Why is subclassing too much bad (and hence why should we use prototypes to do away with it)?
- Why, php? WHY???
- wikidiff2
- Wincache extension for PHP
- Wordpress
- www.exakat.io
- xattr
- xcache
- Xdebug
- xdiff
- XHprof Documentation
- XML External Entity
- XML Parser
- XML-RPC
- xmlreader
- XMLWriter
- XSL extension
- YAML Ain't Markup Language
- Yii
- Yoda Conditions
- Zend Monitor - PHP API
- ZeroMQ
- zip
- Zip
- Zlib