Bern University of Applied Sciences

BTI7301 - PROJECT 1

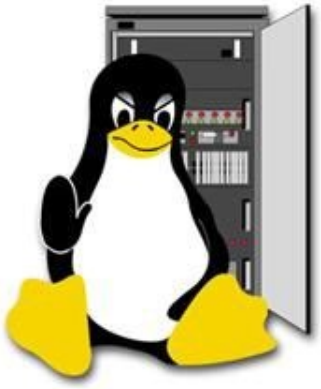
Mail Server Set-Up & Security-Hardening Script

User Manual

Authors:
Fridolin Zurlinden
Ismael Riedo
Jan Henzi

Tutor:
Dr. Simon Kramer

January 23, 2019



Abstract

This paper gives you an overview understanding, what is this hardening script about and what happens on the server when you execute it. It escorts you through every step of the installation. Starting with the run options, then firewall, DNS, internal user management, SSH, mail, and at the end the web part. It demonstrates the contrast between a non-hardened and a hardened server by this script. Gives you a full manual how to configure your email client and it concludes everything with some future works ideas.

Contents

List of Figures

1 Introduction

In this document, we descripted a full installation of follow components: Firewall, DNS, SSH, Email and Web. As well, we show you all possibilities you can take within the provided script. We let you understand that the components are hardened and give you some thoughts about the future. Everything start with the walkthrough chapter, a complete walkthrough through the scripts explained based on the output. You can quickly and clearly follow up what is happening where and how. There is an overview code directory tree, which indicates all the scripts which are made. After it starts with all the components, which will be installed.

- **Overview**: This Section is about the main script, which bundles all components. The user also has the possibility to create his individual setup and if necessary to perform uninstallation and modifications on a second run.
- **Firewall**:The firewall can be extended with additional rules with the help of a configuration file. The file can be found in the "files" directory under the name "fw.conf".
- **DNS**: In the DNS part, two DNS servers will be installed. Both are from nlnetlabs: unbound and NSD. Unbound is used as resolver, to handle all requests from this server and NSD is used as authoritative name server. Such a separation increases security.
- **User management**: Since some services also require Unix users, scripts have been written to make it easier to create and assign users to services. Both the mail part and the SSH part need such users.
- **SSH**: The SSH part is not only about making the server more secure by forbidding the root user to log in, but also about equipping new or existing users with right and ssh keys so that a login is still possible via specific users.
- **Email**: A secure mail server with postfix is set up in the email part. Unix users are also required here.
- **Web**: In the web part nginx and apache are used. The nginx is used as reverse proxy and the apache as frontend webserver.

Results are important, so the hardening Tests section is about giving you a feeling about what one can expect from a successful complete run of the script. Based on common hardening pages and tools, tests were made to show how secure the server is, before the script and after a complete run of the scripts.

- **Firewall**: The firewall tests were performed with nmap. The results of the firewall test can seem a bit irritating at first: more ports are open than before. However, this makes sense, because certain ports are needed by the services. What is open or closed before also depends on the host of the server.
- **DNS**: It was important not only to make a DNS secure, but also to make it independent. With the own resolver this was very successful and so the user of the scripts has a DNS detached from big companies like Google or Cloudflare.
- **SSH**: Apart from forbidding the root user from logging in, we also made sure that after the SSH configuration only algorithms are used that are currently considered as secure.
- **Mail**: With secure protocols and antispam measures, the mail server was configured so that it received very good marks during the tests. We tested it with https://emailsecuritygrader.com and https://www.hardenize.com.

- **Web**: Also the web part could be tested via https://www.hardenize.com . There we also achieved very good values.

In addition, you will find a small step-by-step guide (currently only macOS guide) to set up the email client to work with your server. Moreover, in the conclusion we discuss about extended functionalities like multiple domains / e-mail addresses, more hardening possibilities, containerization and code migration. At the very end, you find all configured config files of each component.

1.1 Prerequisits

In order to start a complete run of the scripts, it is worth making some things ready in advance so that the run can go clean and fast.

1.1.1 Ubuntu 18.04 Server

You need your own Ubuntu Server (Version 18.04), which is an accessible from the internet. You need root access.

1.1.2 Domain

You need your own domain. A free test domain can easily be found with a small search in any web search engine.

1.1.3 Minimal Linux knowledge

The script is in command line only, so you need some minimal Linux knowledge. You should know how to navigate and execute a command inside the terminal.

1.2 Architecture overview

Here you see a simple architecture overview, how your server will look like, if you install all the components.

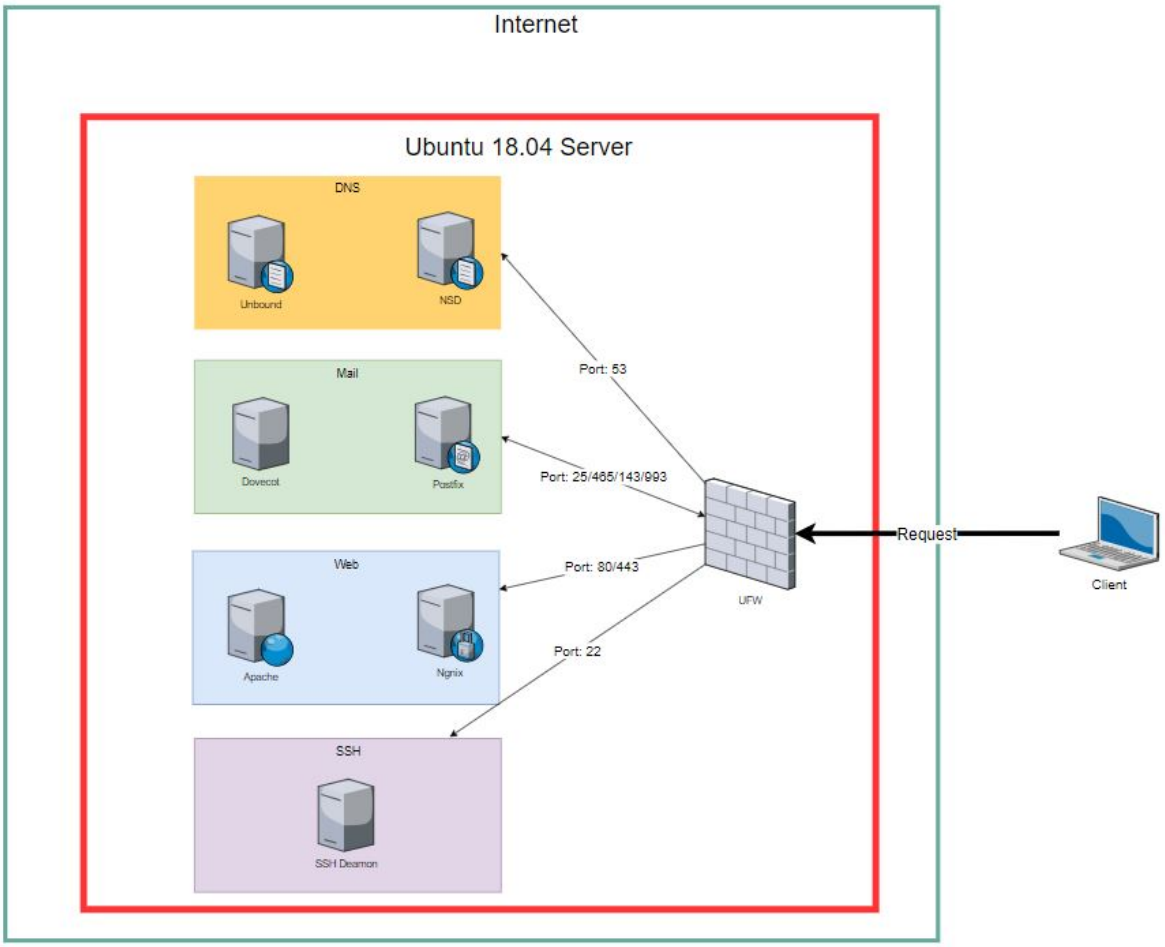


Figure 1.1: Architecture overview

2 Walkthrough

2.1 Code directory tree

```
.
├── dns
│   ├── dns.sh
│   ├── dns_nsd.sh
│   ├── dns_unbound.sh
│   ├── nsd
│   │   ├── configBackwardsZoneDNS_nsd.sh
│   │   ├── configDNS_nsd.sh
│   │   ├── configForwardZoneDNS_nsd.sh
│   │   ├── finalisationDNS_nsd.sh
│   │   ├── installDNS_nsd.sh
│   │   └── testDNS_nsd.sh
│   ├── unbound
│   │   ├── configDNSAccess_unbound.sh
│   │   ├── configDNSHardening_unbound.sh
│   │   ├── configDNSListening_unbound.sh
│   │   ├── finalisationDNS_unbound.sh
│   │   ├── installDNS_unbound.sh
│   │   └── testDNS_unbound.sh
│   └── uninstall_dns.sh
├── files
│   └── fw.conf -> fw/fw.conf
├── fw
│   ├── controllTraffic.sh
│   ├── enableUfw.sh
│   ├── fw.conf
│   ├── fw.sh
│   ├── specificConfigurations.sh
│   └── uninstall_fw.sh
├── mail
│   ├── alias.sh
│   ├── checkDomain.sh
│   ├── clientCertificate.sh
│   ├── dkim.sh
│   ├── dmarc.sh
│   ├── dnsRecords.sh
│   ├── dovecot.sh
│   ├── hardeningMail.txt
│   ├── mail.sh
│   ├── restart.sh
│   ├── spf.sh
│   ├── tls.sh
│   └── uninstall_mail.sh
├── setup.sh
├── ssh
│   ├── config.sh
│   ├── restart.sh
│   ├── ssh.sh
│   └── sshkeys.sh
├── utils
│   ├── checkPackage.sh
│   ├── chooseIp.sh
│   ├── getAllIpv4.sh
│   ├── getAllIpv6.sh
│   ├── getIpv4.sh
│   ├── getIpv6.sh
│   ├── logging.sh
│   ├── removeFolder.sh
│   ├── removePackage.sh
│   ├── revIpv4.sh
│   ├── summary.sh
│   ├── user.sh
│   └── valid_ipv4.sh
└── web
    ├── apache
    │   ├── configureApache.sh
    │   └── enableApache.sh
    ├── nginx
    │   ├── configureNginx.sh
    │   ├── enableNginx.sh
    │   └── nginxCertConfig.sh
    ├── uninstall_web.sh
    └── web.sh
```

DNS

The DNS setup is based on two complety independent servers:

- nsd as authoritative nameserver (queries from the internet to this domain).
- unbound as local dns resolver (queries from this host).

Firewall

The firewall configuration is loaded from this file (files/fw.conf). Standard ports are already defined, additional ports can be specified in this file.

Anti-spam measures

Following DNS based anti-spam measures are configured for the mailserver. They makes sure spam mail is recognized during recieving and all sent mails, reach their destination without being classified as spam from the recieving side:

- DKIM
- DMARC
- SPF

Entrypoint

This is the main entrypoint for the setup (./setup.sh). From here on the user is guided through the whole setup process.

Webserver

As webserver two components interact together:

- Nginx is used as a reverse proxy to terminate SSL connections and provide a secure HTTPS connection.
- Apache is used as a web server to provide webpages, could later also be used as application server (see section 5.1.3).

2.2 Overview

2.2.1 Complete run

In this section we make a full configuration with the administartions script “setup.sh”. We describe every step.
First, we will install the ufw (uncomplicated firewall), which will then be configured by the script.

```
<INFO> - Tue Jan  8 11:14:31 UTC 2019 - No Modification Flag found. Seems to be the
    first run. Will start hardening now.
*** QUESTION *** Do you wish to perform a complete run (Firewall, DNS, SSH, Mail, Web)
    [y/n]?  y
<INFO> - Tue Jan  8 11:14:39 UTC 2019 - Complete run set to true

[...]
```

At the end of the whole configuration a modification flag is set, which is checked at a rerun. So you have the option modify and delete at a later time (visible in the next section).

```

[...]

<INFO> - Tue Jan 8 11:25:19 UTC 2019 - Set modification Flag.
<INFO> - Tue Jan 8 11:25:19 UTC 2019 - Done. Finished with configurations
```

2.2.2 Rerun run

If you run the script again at a later time, there are some small changes in the possibilities. New you will have the option “Modify”, which makes it possible to configure all or certain components again (in the example only the firewall was configured again), or also the option “Delete”, with which you could remove certain components.

```
*** QUESTION *** Modification Flag found. Please choose option: modify/uninstall [m/
u]?  m
<INFO> - Wed Jan  9 08:45:33 UTC 2019 - Modification choosen
*** QUESTION *** Do you wish to perform a complete run (Firewall, DNS, SSH, Mail, Web)
    [y/n]?  n
<INFO> - Wed Jan  9 08:45:34 UTC 2019 - Complete run set to false.
<INFO> - Wed Jan  9 08:45:34 UTC 2019 - Start the specific selection for single parts.
*** QUESTION *** Do you wish to perform action on fw  [y/n]?  y
<INFO> - Wed Jan  9 08:45:36 UTC 2019 - Action for fw set to true
*** QUESTION *** Do you wish to perform action on dns  [y/n]? n
<INFO> - Wed Jan  9 08:45:37 UTC 2019 - Action for dns set to false (will skip it).
*** QUESTION *** Do you wish to perform action on ssh  [y/n]?  n
<INFO> - Wed Jan  9 08:45:37 UTC 2019 - Action for ssh set to false (will skip it).
*** QUESTION *** Do you wish to perform action on mail  [y/n]?  n
<INFO> - Wed Jan  9 08:45:40 UTC 2019 - Action for mail set to false (will skip it).
*** QUESTION *** Do you wish to perform action on web  [y/n]?  n
<INFO> - Wed Jan  9 08:45:43 UTC 2019 - Action for web set to false (will skip it).

[...]

<INFO> - Wed Jan  9 08:45:55 UTC 2019 - Set modification Flag.
<INFO> - Wed Jan  9 08:45:55 UTC 2019 - Done. Finished with configurations
```

Explanation of [...]

At this point specific components are configured, which are explained separately in this document. This section is only about the administration script, which triggers the whole processes.

2.2.3 Overview process diagram

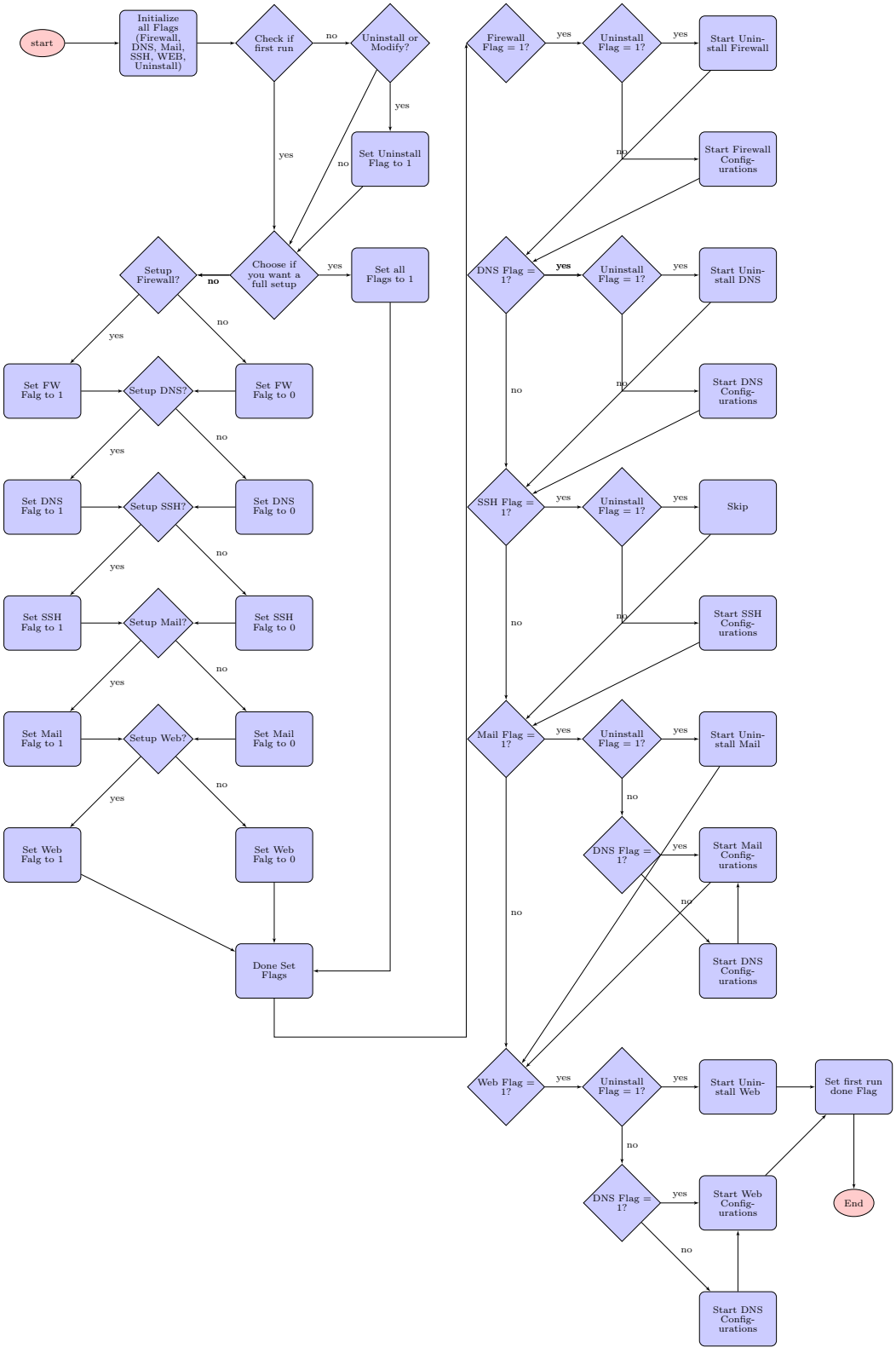


Figure 2.1: Setup process diagram

2.3 Firewall

In this section we make a full Firewall configuration. We describe every step.
First, we will install the ufw (uncomplicated firewall), which will then be configured by the script.

```
<INFO> - Tue Jan  8 11:14:39 UTC 2019 - Starting Firewall Configurations.
<INFO> - Tue Jan  8 11:14:39 UTC 2019 - Will install 'ufw' now. Please wait...
..............
<INFO> - Tue Jan 8 11:14:52 UTC 2019 - Package 'ufw' is installed now.
```

After the successful installation it goes on with a basic security. This includes enabling all traffic out and blocking all traffic in. So that nobody is locked out of his own server right at the beginning, seperat ssh on port 22 is enabled and configured as the only access from outside at this time.

```
<INFO> - Tue Jan  8 11:14:53 UTC 2019 - Ufw is enabled now.
<FW> - Tue Jan  8 11:14:53 UTC 2019 - UFW enable done.
<INFO> - Tue Jan  8 11:14:53 UTC 2019 - Start Firewall Hardening. (close all non
    relevant ports)
<INFO> - Tue Jan  8 11:14:54 UTC 2019 - All incoming and outgoing traffic is handeled
    now.
<FW> - Tue Jan  8 11:14:54 UTC 2019 - Traffic controll done.
<INFO> - Tue Jan 8 11:14:54 UTC 2019 - Activate SSH Connection for host 'XYZ'.
```

After setting up the base security, special configurations are loaded, which the user can add by himself. He does this by adding the necessary rules to the config-file “fw.conf” in the folder “files”. The user has the possibility to say whether he wants to allow (ALLOW) or deny (DENY) a certain access. Listed in the output are the minimum accesses needed for a comlete run of the scripts. These configurations are already present in the configuration file by default. At the very end a list of the now activated rules will be displayed.

```
<INFO> - Tue Jan  8 11:14:55 UTC 2019 - Looking for Firewall Config file for specific
    configurations
<INFO> - Tue Jan  8 11:14:55 UTC 2019 - File Found. /root/files/fw.conf
# SSH
<INFO> - Tue Jan  8 11:15:19 UTC 2019 - Working on 'allow 22/tcp'.
<INFO> - Tue Jan  8 11:15:19 UTC 2019 - Working on 'allow 22/udp'.
# DNS
<INFO> - Tue Jan  8 11:15:20 UTC 2019 - Working on 'allow 53/tcp'.
<INFO> - Tue Jan  8 11:15:20 UTC 2019 - Working on 'allow 53/udp'.
# MAIL
<INFO> - Tue Jan  8 11:15:20 UTC 2019 - Working on 'allow 25/tcp'.
<INFO> - Tue Jan  8 11:15:20 UTC 2019 - Working on 'allow 25/udp'.
# SECURE SMTP
<INFO> - Tue Jan  8 11:15:21 UTC 2019 - Working on 'allow 465/tcp'.
<INFO> - Tue Jan  8 11:15:21 UTC 2019 - Working on 'allow 465/udp'.
# IMAP
<INFO> - Tue Jan  8 11:15:21 UTC 2019 - Working on 'allow 143/tcp'.
<INFO> - Tue Jan  8 11:15:21 UTC 2019 - Working on 'allow 143/udp'.
# IMAP TLS
<INFO> - Tue Jan  8 11:15:21 UTC 2019 - Working on 'allow 993/tcp'.
<INFO> - Tue Jan  8 11:15:22 UTC 2019 - Working on 'allow 993/udp'.
# HTTP HTTPS
<INFO> - Tue Jan  8 11:15:22 UTC 2019 - Working on 'allow 80/tcp'.
<INFO> - Tue Jan  8 11:15:22 UTC 2019 - Working on 'allow 443/tcp'.
<INFO> - Tue Jan  8 11:15:22 UTC 2019 - Done Specific configurations.
<FW> - Tue Jan  8 11:15:22 UTC 2019 - Specific Configurations of UFW done.
<INFO> - Tue Jan  8 11:15:22 UTC 2019 - Firewall Configurations done.
Status:  active

To                         Action      From
```

```
--                         ------      ----
22/tcp                     ALLOW       Anywhere
22/udp                     ALLOW       Anywhere
53/tcp                     ALLOW       Anywhere
53/udp                     ALLOW       Anywhere
25/tcp                     ALLOW       Anywhere
25/udp                     ALLOW       Anywhere
465/tcp                    ALLOW       Anywhere
465/udp                    ALLOW       Anywhere
143/tcp                    ALLOW       Anywhere
143/udp                    ALLOW       Anywhere
993/tcp                    ALLOW       Anywhere
993/udp                    ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
22/udp (v6)                ALLOW       Anywhere (v6)
53/tcp (v6)                ALLOW       Anywhere (v6)
53/udp (v6)                ALLOW       Anywhere (v6)
25/tcp (v6)                ALLOW       Anywhere (v6)
25/udp (v6)                ALLOW       Anywhere (v6)
465/tcp (v6)               ALLOW       Anywhere (v6)
465/udp (v6)               ALLOW       Anywhere (v6)
143/tcp (v6)               ALLOW       Anywhere (v6)
143/udp (v6)               ALLOW       Anywhere (v6)
993/tcp (v6)               ALLOW       Anywhere (v6)
993/udp (v6)               ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
443/tcp (v6)               ALLOW       Anywhere (v6)

<FW> - Tue Jan  8 11:15:22 UTC 2019 - UFW Configurations done.
<FW> - Tue Jan 8 11:15:22 UTC 2019 - Actions on Firewall Done
```

2.3.1 Firewall process diagram

Here we have process diagram of how the script works with all possible outcomes.

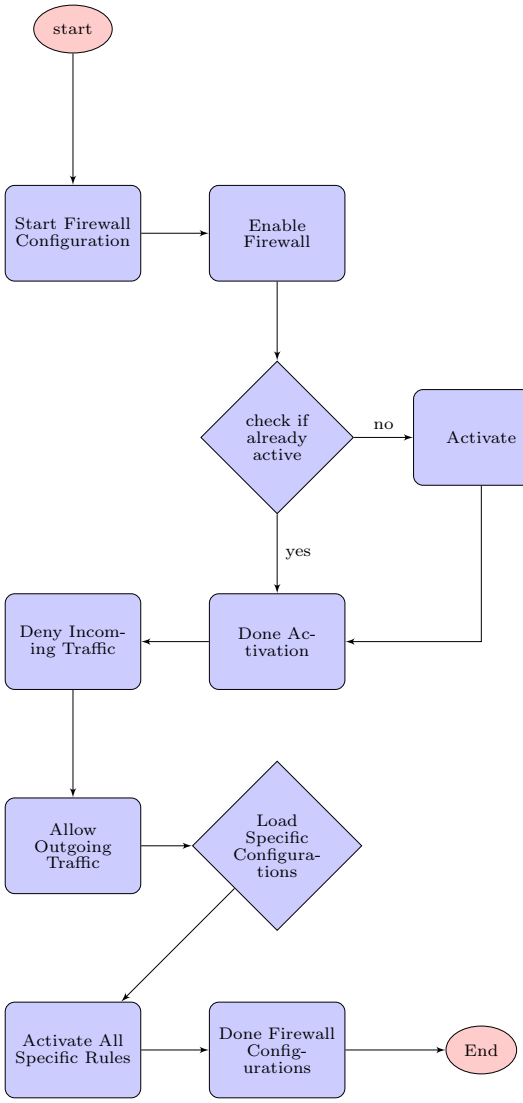


Figure 2.2: Firewall process diagram

2.4 DNS

In this section we make a full DNS configuration. We describe every step.
First we install **unbound** , a DNS resolver which will be used from now for all DNS requests from this server.

```
<INFO> - Tue Jan  8 11:15:22 UTC 2019 - Starting DNS Configurations.
*NOTE* We install two DNS Server, one for internal DNS requests (for this server and/
    or home clients) and one authoritative DNS Server for your domain
*PART 0: We install the basic configuration for unbound - we come back to it later
<INFO> - Tue Jan  8 11:15:22 UTC 2019 - Install DNS
<INFO> - Tue Jan  8 11:15:24 UTC 2019 - Will install 'unbound' now. Please wait...
............
<INFO> - Tue Jan  8 11:15:36 UTC 2019 - Package 'unbound' is installed now.
<INFO> - Tue Jan  8 11:15:37 UTC 2019 - Configure DNS Hardening (Hide version, use
    root-hints file, use trust-anchored zones for DNSSEC requests)
<INFO> - Tue Jan  8 11:15:37 UTC 2019 - Configure DNS Ports, IPs
<INFO> - Tue Jan  8 11:15:37 UTC 2019 - Server will listen with localhost on port 53
<INFO> - Tue Jan  8 11:15:37 UTC 2019 - Configure DNS Access
<INFO> - Tue Jan  8 11:15:37 UTC 2019 - Configure this Client
<INFO> - Tue Jan  8 11:15:37 UTC 2019 - Server will use localhost as DNS
```

After we continue with the **authoritative Name Server: NSD**, have ready your domain (highlighted).

```
*PART 1: We start with the authoritative Name Server: NSD

!!CAUTION!! you need your own domain - IF NOT the server wont be functional
DO NOT use a domain which does not belong to you, it may be illegal
*NOTE* If you want to test it only, you can get a free domain like .tk or .ga - just
    search in your favorite web search engine (duckduckgo, google etc..)

Press enter to continue

 *** QUESTION *** do you have your own domain? (y/n/abort) y
<INFO> - Tue Jan  8 11:15:59 UTC 2019 -

 *** QUESTION *** please enter your domain:   examplerun.cf

 *** QUESTION *** is examplerun.cf correct? (y/n/abort)   y

 <INFO> - Tue Jan  8 11:16:15 UTC 2019 - We will configure the authoritative DNS
    Server with the domain: examplerun.cf
```

Once the domain is set, check if the follow output is your extern IP, if yes continue.

```
 *** QUESTION *** is this 104.248.137.212 your external IP address ? (y (default)/n/
    abort)   y
<INFO> - Tue Jan  8 11:16:48 UTC 2019 - We will configure the authoritative DNS Server
    with this: 104.248.137.212
<INFO> - Tue Jan  8 11:16:48 UTC 2019 - Install authoritative DNS for : examplerun.cf
<INFO> - Tue Jan  8 11:16:48 UTC 2019 - Will install 'nsd' now. Please wait...
..........
<INFO> - Tue Jan  8 11:16:57 UTC 2019 - Package 'nsd' is installed now.
<INFO> - Tue Jan  8 11:16:59 UTC 2019 - Will install 'ldnsutils' now. Please wait...
........
<INFO> - Tue Jan  8 11:17:06 UTC 2019 - Package 'ldnsutils' is installed now.
<INFO> - Tue Jan  8 11:17:06 UTC 2019 - Configure NSD
<INFO> - Tue Jan  8 11:17:06 UTC 2019 - Configure Forward Zone
<INFO> - Tue Jan  8 11:17:06 UTC 2019 - Configure Backward Zone
<INFO> - Tue Jan  8 11:17:06 UTC 2019 - Final steps
<INFO> - Tue Jan  8 11:17:11 UTC 2019 - Test NSD
```

Now you can change, as described, your domain (Glue Records).

```
PART 2: You have a full functional authoritative Name Server BUT your domain hoster
    does not know it!
 !! VERY IMPORTANT !! GO to your domain hoster, change the name server for your domain
     to :
                ns1.examplerun.cf with IP: 104.248.137.212
                ns2.examplerun.cf with IP: 104.248.137.212
 !! VERY IMPORTANT !! DO the same for the Glue Records, with the same name server and
     IPs
NOTE: It may take some time to change it - if you have difficulties with this part use
     your favorite web search engine

If you are done, press enter to continue
```

In the last part, if you use the server in your home/work network you can make the domain resolver we installed (unbound) accessible for your local clients. Mostly it is not the case so you can continue with “enter”. At the end we test to resolve a **ipv4** and a **ipv6** address.

```
PART 3: *** QUESTION *** Do you rent this server or is it in your internal network
    area? If you dont know what it means just press enter. (intern / <enter> (default)
    )  <enter>
<INFO> - Tue Jan  8 11:17:40 UTC 2019 - Test local DNS
<INFO> - Tue Jan  8 11:17:40 UTC 2019 - Test ipv4 address
www.google.com.          3600    IN      A       216.58.210.4
<INFO> - Tue Jan  8 11:17:40 UTC 2019 - Test ipv6 address
ipv6.google.com.         604800  IN      CNAME   ipv6.l.google.com.
ipv6.l.google.com.       3600    IN      AAAA    2a00:1450:4005:800::200e

Successfully installed NSD and Unbound
```

And we are done with the DNS part!

2.4.1 DNS architecture diagram

For a better understanding of how a domain name will be resolved, here is a small diagram which indicates how those two servers are separated.

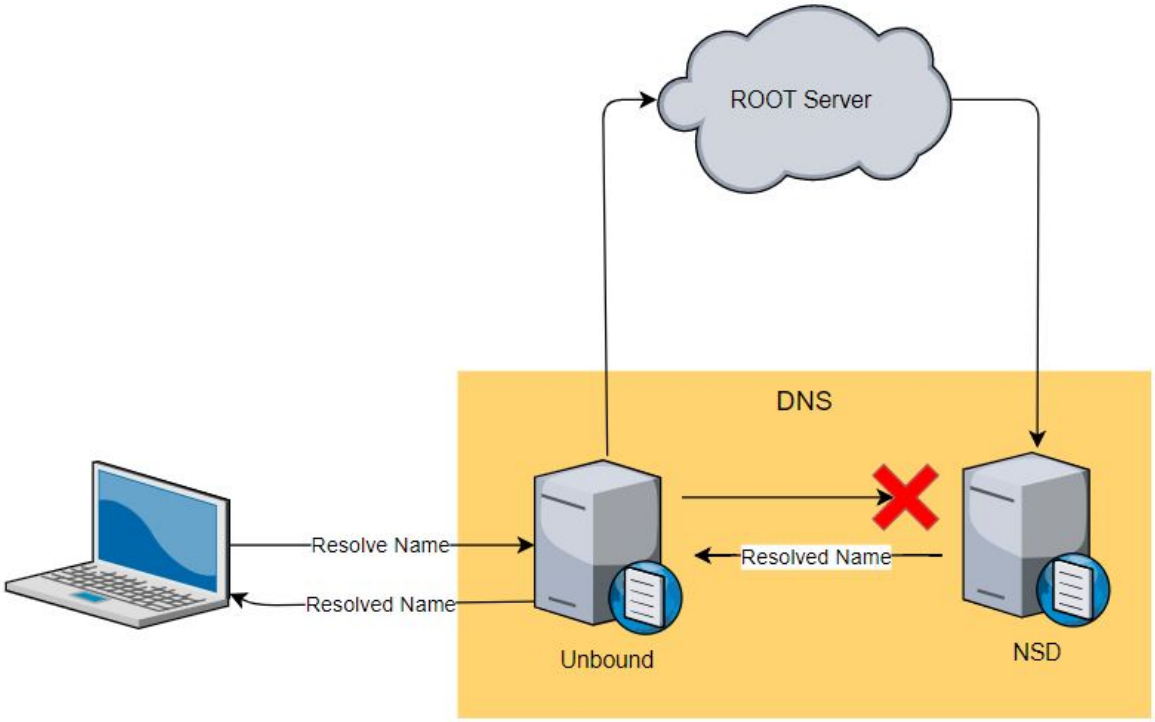


Figure 2.3: Architecture DNS

2.4.2 DNS process diagram

Here we have process diagram of how the script works with all possible outcomes.

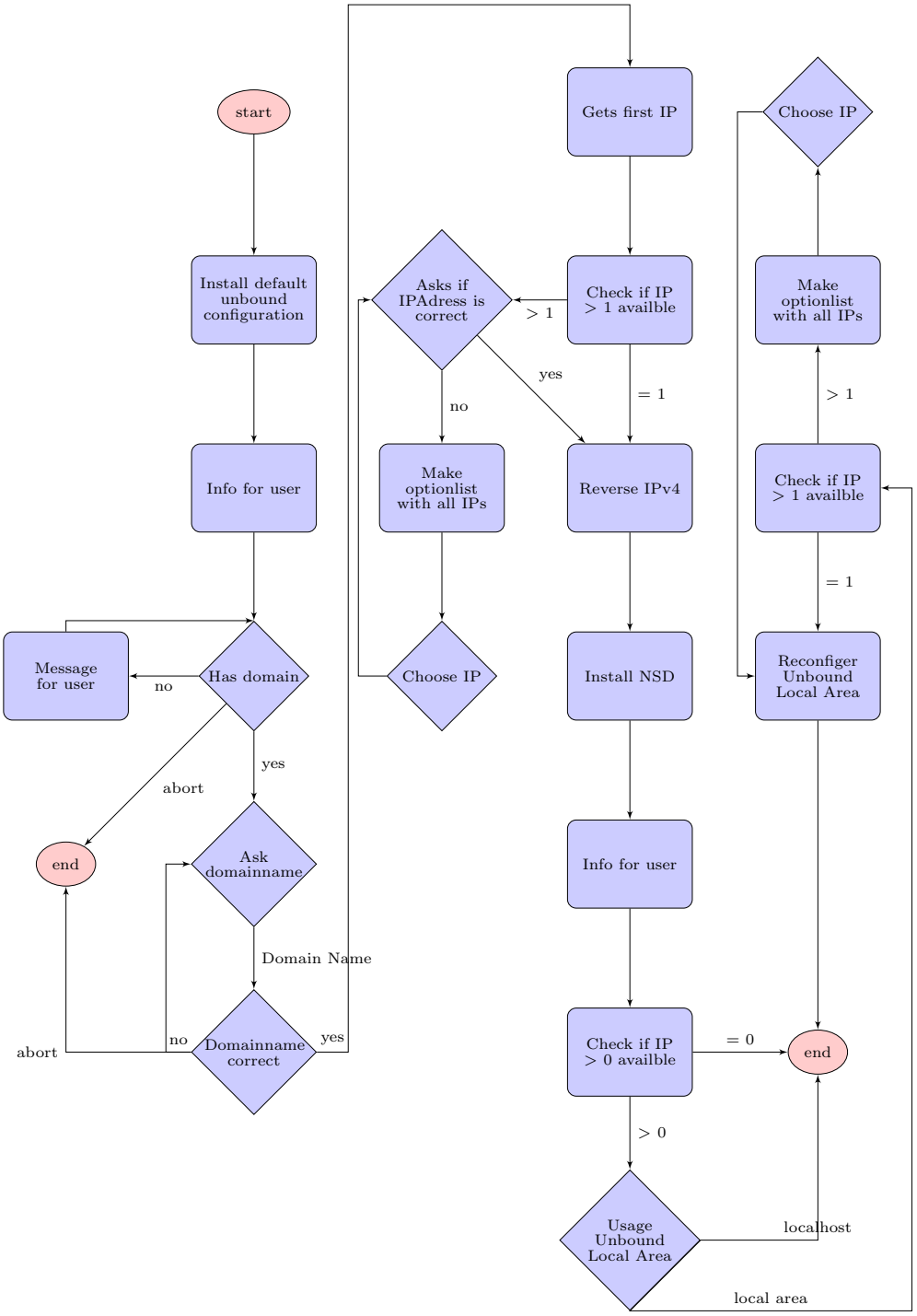


Figure 2.4: DNS process diagram

2.4.3 Multiple domains

After installation you can use multiple sub domains of your domain. All domains will be resolved, as it is configured with a wild-card: (in this example) *.examplerun.cf. As the script was designed for someone with basic understanding of computer technology, to have multiple domains on the same server is not possible.

2.5 User management

The usermanagment is used whenever a list of users on the unix system must be selected for a service. In the following subsections you find a brief overview of all the available actions.

2.5.1 Actions

Here you find a short example for each action, inputs are highlighted.

Help

The help text displays at the start of the function end everytime the command help ist entered.

```
<INFO> - Mon Jan 14 09:44:29 UTC 2019 - Doing user handling for SSH configuration
Usage:
        This function helps you manage the users on this system and select the ones
           you wish to provision for the ssh service.
        Following actions are available:
                help:           Display this help
                display:        Show all unix users on this system
                add:            Add a unix user to this system (this implies the
                   select action)
                delete:         Remove a unix user from this system (this implies the
                   unselect action)
                select:         Add an existing unix user to the list of users which
                   will be provisioned for the service ssh
                unselect:       Remove a user from the list of users which will be
                   provisioned for the service ssh
                show:           Show the list of users which will be provisioned for
                   the service ssh
                quit:           Exit this function
```

Display

Show all unix users on the system:

```
<INFO> - Mon Jan 14 09:44:29 UTC 2019 - Number of users selected: 0
*** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
    unselect/show/quit)  display

<INFO> - Mon Jan 14 09:44:42 UTC 2019 - Displaying users for this system
root
sync
```

Add

Add a unix user to the system (this implies the select action)

```
<INFO> - Mon Jan 14 09:44:42 UTC 2019 - Number of users selected: 0
*** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
    unselect/show/quit)  add

<INFO> - Mon Jan 14 09:45:07 UTC 2019 - Adding user for this system
*** QUESTION *** please enter the desired username to be added?  alice

id: 'alice': no such user
Adding user 'alice' ...
Adding new group 'alice' (1000) ...
```

```
Adding new user 'alice' (1000) with group 'alice' ...
Creating home directory '/home/alice' ...
Copying files from '/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for alice
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
<INFO> - Mon Jan 14 09:45:22 UTC 2019 - Successfully added user alice, adding it to
    the list for ssh

*** QUESTION *** Do you want to add sudo privileges for the user alice? (y/N)  y
<INFO> - Mon Jan 14 09:45:28 UTC 2019 - Adding sudo privileges for user alice
<INFO> - Mon Jan 14 09:45:28 UTC 2019 - Successfuly added sudo privileges for user
    alice
```

Show

Show the list of users which will be provisioned for the service

```
<INFO> - Mon Jan 14 09:45:28 UTC 2019 - Number of users selected: 1
*** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
    unselect/show/quit)  alice
<INFO> - Mon Jan 14 09:45:46 UTC 2019 - Displaying selected users for service ssh
<INFO> - Mon Jan 14 09:45:46 UTC 2019 - alice
```

Unselect

Remove a user from the list of users which will be provisioned for the service

```
<INFO> - Mon Jan 14 09:45:46 UTC 2019 - Number of users selected: 1
*** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
    unselect/show/quit)  unselect
<INFO> - Mon Jan 14 09:45:56 UTC 2019 - Unselecting user for service ssh
*** QUESTION *** please enter the desired username to be removed from selection?
     alice
<INFO> - Mon Jan 14 09:45:59 UTC 2019 - Removed alice from selection for ssh
```

Select

Add an existing unix user to the list of users which will be provisioned for the service

```
<INFO> - Mon Jan 14 09:45:59 UTC 2019 - Number of users selected: 0
*** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
    unselect/show/quit)  select
<INFO> - Mon Jan 14 09:46:07 UTC 2019 - Selecting user for service ssh
*** QUESTION *** please enter the desired username to be selected?  alice
<INFO> - Mon Jan 14 09:46:09 UTC 2019 - Selected alice for ssh
```

Delete

Remove a unix user from the system (this implies the unselect action)

```
<INFO> - Mon Jan 14 09:46:13 UTC 2019 - Number of users selected: 1
*** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
    unselect/show/quit)  delete
<INFO> - Mon Jan 14 09:46:20 UTC 2019 - Removing user for this system
*** QUESTION *** please enter the desired username to be deleted?  sync
Removing files ...
Removing user 'sync' ...
Warning: group 'nogroup' has no more members.
Done.
<INFO> - Mon Jan 14 09:46:23 UTC 2019 - Successfully deleted user sync
```

Quit

Exit the function

```
<INFO> - Mon Jan 14 09:46:46 UTC 2019 - Number of users selected: 1
*** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
    unselect/show/quit)  quit
```

2.5.2 User management process diagram

Here we have process diagram of how the script works with all possible outcomes.

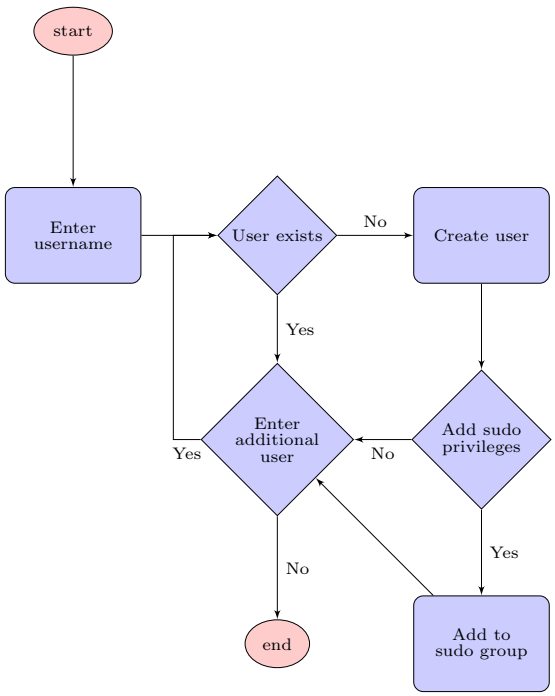


Figure 2.5: User management process diagram

2.6 SSH

2.6.1 Configuration

This is an example of the SSH configuration part, all inputs are highlighted in yellow.

User management ssh

Here is a minimal example for the ssh user handling, for further information see section 2.5

```
<SSH> - Mon Jan 14 09:44:29 UTC 2019 - Perform actions on SSH
<SSH> - Mon Jan 14 09:44:29 UTC 2019 - Perform install on SSH
<INFO> - Mon Jan 14 09:44:29 UTC 2019 - Doing user handling for SSH configuration
Usage:
        This function helps you manage the users on this system and select the ones
            you wish to provision for the ssh service.
        Following actions are available:
                help:           Display this help
                display:        Show all unix users on this system
                add:            Add a unix user to this system (this implies the
                    select action)
                delete:         Remove a unix user from this system (this implies the
                    unselect action)
                select:         Add a existing unix user to the list of users which
                    will be provisioned for the service ssh
                unselect:       Remove a user from the list of users which will be
                    provisioned for the service ssh
                show:           Show the list of users which will be provisioned for
                    the service ssh
                quit:           Exit this function

<INFO> - Mon Jan 14 09:44:42 UTC 2019 - Number of users selected: 0
 *** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
    unselect/show/quit) add

<INFO> - Mon Jan 14 09:45:07 UTC 2019 - Adding user for this system
 *** QUESTION *** please enter the desired username to be added?  alice

id: 'alice': no such user
Adding user 'alice' ...
Adding new group 'alice' (1000) ...
Adding new user 'alice' (1000) with group 'alice' ...
Creating home directory '/home/alice' ...
Copying files from '/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for alice
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
<INFO> - Mon Jan 14 09:45:22 UTC 2019 - Successfully added user alice, adding it to
    the list for ssh

 *** QUESTION *** Do you want to add sudo privileges for the user alice? (y/N)  y
<INFO> - Mon Jan 14 09:45:28 UTC 2019 - Adding sudo privileges for user alice
<INFO> - Mon Jan 14 09:45:28 UTC 2019 - Successfuly added sudo privileges for user
    alice

<INFO> - Mon Jan 14 09:46:46 UTC 2019 - Number of users selected: 1
```

```
46  *** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
        unselect/show/quit)  quit
```

SSH key generation

For every user a personal ssh key-pair is generated, the user has to enter the passphrase. When the setup is complete the user can download all his keys, certificates and passphrases from the server.

```
1  <INFO> - Mon Jan 14 09:46:55 UTC 2019 - Leaving user management
2  <INFO> - Mon Jan 14 09:46:55 UTC 2019 - Generating SSH keys for users
3  <INFO> - Mon Jan 14 09:46:55 UTC 2019 - Generating SSH key for user alice
4  <INFO> - Mon Jan 14 09:46:55 UTC 2019 - IMPORTANT - make sure you remember ALL the
       passphrases and save your keys to some secure location - IMPORTANT
5
6  <INFO> - Mon Jan 14 09:46:55 UTC 2019 - IMPORTANT - !!!passphrase MUST be minimum 5
       characters long!!! - IMPORTANT
7  Generating public/private rsa key pair.
8  Enter passphrase (empty for no passphrase): ********
9  Enter same passphrase again: ********
10 Your identification has been saved in /home/alice/.ssh/id_rsa.
11 Your public key has been saved in /home/alice/.ssh/id_rsa.pub.
12 The key fingerprint is:
13 SHA256:82nk2iy0lS6n+KJdIIfGeR/TBbkglLoxihMZVMdYif0 alice@examplerun.cf
14 The keys randomart image is:
15 +---[RSA 4096]----+
16 ....*+o.  ..
17 .  o.+o .  ..
18  o ...  .  ..
19 o .+o E ...
20  o .*++ S o.
21 o ...+ o.Bo.
22  .  .o+=
23  ..o+=o
24  ..oo+=o
25 +----[SHA256]-----+
26 <INFO> - Mon Jan 14 09:57:45 UTC 2019 - IMPORTANT - This is your private key, this is
       the only thing you need right to save. All of your certificate and keys are saved
       to your home. You need this key to download them. - IMPORTANT
27 -----BEGIN RSA PRIVATE KEY-----
28 Proc-Type: 4,ENCRYPTED
29 DEK-Info: AES-128-CBC,8B5BFD485A805BA25316C21C266CCDCF
30
31 BCh9X2Lo6jxZBtVRprliAhCp/TVX+60EPxBu59sUVWukOnB8CKy/bqEhkOb6DVsh
32 ...
33 VrxQPgOeipL3zr54Zq9SY6NC2BCu5OygDHWXsKwrBTnx0Hi262jo6bX7Kqmog4qX
34 -----END RSA PRIVATE KEY-----
```

SSH hardening & cleanup

At the end the user keys are moved to the corresponding user home and the SSH configuration is hardenend [5]:

- Root login is not permited
- Passwort login is not permited
- X11 is not permited
- Only secure alogrithms are permited

```
<INFO> - Mon Jan 14 09:57:45 UTC 2019 - Cleaning up..
<INFO> - Mon Jan 14 09:57:45 UTC 2019 - Hardening SSH daemon config

<INFO> - Mon Jan 14 09:57:45 UTC 2019 - Hardening sshd config (disable X11Forwarding,
    enable domainname lookup, disable root login, enabling only strong algorithms)

<INFO> - Mon Jan 14 09:57:45 UTC 2019 - Hardening complete

<INFO> - Mon Jan 14 09:57:45 UTC 2019 - Finishing up, restarting services

<INFO> - Mon Jan 14 09:57:45 UTC 2019 - Restarting all components for SSH

<INFO> - Mon Jan 14 09:57:45 UTC 2019 - SSH daemon configuration complete.
<SSH> - Mon Jan 14 09:57:45 UTC 2019 - Actions on SSH Done
```

2.6.2 SSH process diagram

Here we have a process diagram of how the script works with all possible outcomes.

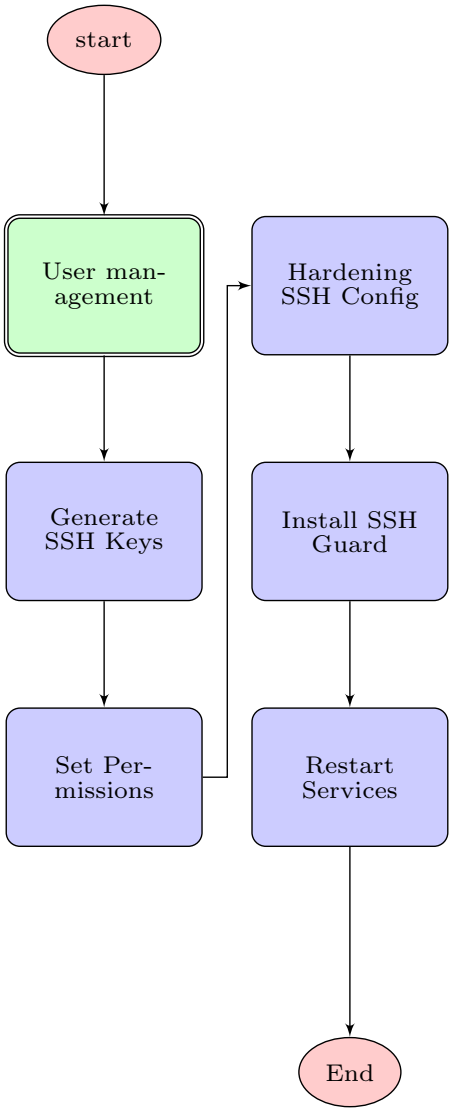


Figure 2.6: SSH process diagram

2.7 E-Mail

2.7.1 Configurations

This is an example of the Email configuration part, all inputs are highlighted in yellow.

Package installation

First all the neccessary packages are installed, this includes:

- postfix
- mailutils
- letsencrypt
- dovecot
- opendkim
- opendmarc
- zip

```
<Mail> - Mon Jan 14 11:29:52 UTC 2019 - Perform install on Mail
<INFO> - Mon Jan 14 11:29:52 UTC 2019 - Setting up MX and SPF records in dns

<INFO> - Mon Jan 14 11:29:52 UTC 2019 - Appending DNS records for the mailserver to zonefile

<INFO> - Mon Jan 14 11:29:52 UTC 2019 - Reloading zone files..

<INFO> - Mon Jan 14 11:29:52 UTC 2019 - Installing mailserver packages (postfix, mailutils, dovecot)
<INFO> - Mon Jan 14 11:30:15 UTC 2019 - Will install 'postfix-pcre' now. Please wait ...
......
<INFO> - Mon Jan 14 11:30:21 UTC 2019 - Package 'postfix-pcre' is installed now.
<INFO> - Mon Jan 14 11:30:21 UTC 2019 - Will install 'postfix-policyd-spf-python' now. Please wait...
.......
<INFO> - Mon Jan 14 11:30:27 UTC 2019 - Package 'postfix-policyd-spf-python' is installed now.
<INFO> - Mon Jan 14 11:30:28 UTC 2019 - Will install 'mailutils' now. Please wait...
........
<INFO> - Mon Jan 14 11:30:36 UTC 2019 - Package 'mailutils' is installed now.
<INFO> - Mon Jan 14 11:30:36 UTC 2019 - Will install 'letsencrypt' now. Please wait...
..............
<INFO> - Mon Jan 14 11:30:51 UTC 2019 - Package 'letsencrypt' is installed now.
<INFO> - Mon Jan 14 11:30:51 UTC 2019 - Will install 'dovecot-core' now. Please wait ...
.....................
<INFO> - Mon Jan 14 11:31:11 UTC 2019 - Package 'dovecot-core' is installed now.
<INFO> - Mon Jan 14 11:31:11 UTC 2019 - Will install 'dovecot-imapd' now. Please wait ...
..............
<INFO> - Mon Jan 14 11:31:24 UTC 2019 - Package 'dovecot-imapd' is installed now.
<INFO> - Mon Jan 14 11:31:25 UTC 2019 - Will install 'opendkim' now. Please wait...
........
<INFO> - Mon Jan 14 11:31:33 UTC 2019 - Package 'opendkim' is installed now.
<INFO> - Mon Jan 14 11:31:33 UTC 2019 - Will install 'opendkim-tools' now. Please wait ...
......
<INFO> - Mon Jan 14 11:31:38 UTC 2019 - Package 'opendkim-tools' is installed now.
```

```
<INFO> - Mon Jan 14 11:31:39 UTC 2019 - Will install 'opendmarc' now. Please wait...
..........
<INFO> - Mon Jan 14 11:31:48 UTC 2019 - Package 'opendmarc' is installed now.
<INFO> - Mon Jan 14 11:31:48 UTC 2019 - Will install 'zip' now. Please wait...
......
<INFO> - Mon Jan 14 11:31:54 UTC 2019 - Package 'zip' is installed now.
```

Client certificates

The setup allows only logins with personal certificates, the following are generated here. This is a minimal configuration for the user managment, for further information see section 2.5

```
<INFO> - Mon Jan 14 11:31:54 UTC 2019 - Configure Mail Hardening (TLS, SPF, DKIM,
    DMARC, dovecot, client certificate login)
Usage:
        This function helps you manage the users on this system and select the ones
            you wish to provision for the mail service.
        Following actions are available:
                help:       Display this help
                display:    Show all unix users on this system
                add:        Add a unix user to this system (this implies the select
                    action)
                delete:     Remove a unix user from this system (this implies the
                    unselect action)
                select:     Add a existing unix user to the list of users which will
                    be provisioned for the service mail
                unselect:   Remove a user from the list of users which will be
                    provisioned for the service mail
                show:       Show the list of users which will be provisioned for the
                    service mail
                quit:       Exit this function

<INFO> - Mon Jan 14 11:34:31 UTC 2019 - Number of users selected: 0
 *** QUESTION *** what action do you like to choose? (display/add/delete/select/
    unselect/show/quit)  select

<INFO> - Mon Jan 14 11:34:33 UTC 2019 - Selecting user for service mail

 *** QUESTION *** please enter the desired username to be selected?  alice

<INFO> - Mon Jan 14 11:34:35 UTC 2019 - Selected alice for mail

<INFO> - Mon Jan 14 11:34:35 UTC 2019 - Number of users selected: 1
 *** QUESTION *** what action do you like to choose? (display/add/delete/select/
    unselect/show/quit)  quit
<INFO> - Mon Jan 14 11:34:39 UTC 2019 - Leaving user management
```

Postfix configuration

In this setup postfix acts as the SMTP Server to send an recieve mail. The script now configures all the neccessary postfix components [4]:

- User mappings (alias, canonical)
- Service users
- TLS (letsencrypt)
- Anti spam measures (SPF, DKIM, DMARC)

```
1
2  <INFO> - Mon Jan 14 11:34:39 UTC 2019 - Mapping users to mail addresses
3
4  <INFO> - Mon Jan 14 11:34:39 UTC 2019 - Adding users to alias and canonical file
5
6  <INFO> - Mon Jan 14 11:34:39 UTC 2019 - Adding supplementary postmaster user for dmarc
       reporting
7
8  <INFO> - Mon Jan 14 11:34:39 UTC 2019 - Setting up TLS with letsencrypt
9
10 <INFO> - Mon Jan 14 11:34:39 UTC 2019 - Running letsencrypt to obtain a certificate
11
12 <INFO> - Mon Jan 14 11:34:40 UTC 2019 - Will install 'certbot' now. Please wait...
13 ...
14 <INFO> - Mon Jan 14 11:34:42 UTC 2019 - Package 'certbot' is installed now.
15 Saving debug log to /var/log/letsencrypt/letsencrypt.log
16 Plugins selected: Authenticator standalone, Installer None
17 Obtaining a new certificate
18 Performing the following challenges:
19 http-01 challenge for mail.examplerun.cf
20 Waiting for verification...
21 Cleaning up challenges
22
23 IMPORTANT NOTES:
24  - Congratulations! Your certificate and chain have been saved at:
25    /etc/letsencrypt/live/mail.examplerun.cf/fullchain.pem
26    Your key file has been saved at:
27    /etc/letsencrypt/live/mail.examplerun.cf/privkey.pem
28    Your cert will expire on 2019-04-14. To obtain a new or tweaked
29    version of this certificate in the future, simply run certbot
30    again. To non-interactively renew *all* of your certificates, run
31    "certbot renew"
32
33  - Your account credentials have been saved in your Certbot
34    configuration directory at /etc/letsencrypt. You should make a
35    secure backup of this folder now. This configuration directory will
36    also contain certificates and private keys obtained by Certbot so
37    making regular backups of this folder is ideal.
38
39  - If you like Certbot, please consider supporting our work by:
40
41    Donating to ISRG / Lets Encrypt:   https://letsencrypt.org/donate
42    Donating to EFF:                    https://eff.org/donate-le
43
44 <INFO> - Mon Jan 14 11:34:50 UTC 2019 - Configuring TLS for postfix
45
46 <INFO> - Mon Jan 14 11:34:51 UTC 2019 - TLS configuration for postfix complete
47
48 <INFO> - Mon Jan 14 11:34:51 UTC 2019 - Restarting postfix service
49
50 <INFO> - Mon Jan 14 11:34:53 UTC 2019 - Setting up SPF (anti spam measure)
51
52 <INFO> - Mon Jan 14 11:34:53 UTC 2019 - Adding SPF configuration to unbound
53
54 <INFO> - Mon Jan 14 11:34:53 UTC 2019 - Adding SPF configuration to postfix config
55
56 <INFO> - Mon Jan 14 11:34:53 UTC 2019 - Setting up DKIM (anti spam measure)
57
58 <INFO> - Mon Jan 14 11:34:53 UTC 2019 - Creating users for DKIM
59
60 <INFO> - Mon Jan 14 11:34:53 UTC 2019 - Configuring opendkim
61
62 opendkim-genkey: generating private key
63 opendkim-genkey: private key written to 2019011411.private
64 opendkim-genkey: extracting public key
```

```
opendkim-genkey: DNS TXT record written to 2019011411.txt

<INFO> - Mon Jan 14 11:34:54 UTC 2019 - Reloading systemd units

<INFO> - Mon Jan 14 11:34:55 UTC 2019 - Generating DNS records for opendkim

<INFO> - Mon Jan 14 11:34:55 UTC 2019 - Setting up DMARC (anti spoofing measure)

<INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configurting opendmarc

<INFO> - Mon Jan 14 11:34:55 UTC 2019 - Reloading systemd units

<INFO> - Mon Jan 14 11:34:55 UTC 2019 - Adding DNS records for opendmarc

<INFO> - Mon Jan 14 11:34:55 UTC 2019 - Integrating opendmarc into postfix
```

Dovecot configuration

Dovecot acts as the IMAP server to enable clients to fetch mail from the server. The authentication is done via client certificates [11]. At the end the generated certificates for the user can be downloaded over a secure SSH connection. This includes:

- Dovecot SSL (letsencrypt)
- Authentication via certificates
- Preparation of artifacts (ZIP file with certificates) and download command

```
<INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring dovecot as imap server

<INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring dovecot

<INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring dovecot service

<INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring dovecot SSL

<INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring dovecot SSL

<INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring external auth extension

<INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring postfix for client certificates

<INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring client certificate authentication

<INFO> - Mon Jan 14 11:34:57 UTC 2019 - Generating certificate authority, please enter
    a passphrase when promted:

Enter New CA Key Passphrase: ******
Re-Enter New CA Key Passphrase: ******
Generating RSA private key, 4096 bit long modulus
.....................................................................
.....................................................................
.....................................................................
.....................................++
..................++
e is 65537 (0x010001)
Enter pass phrase for /root/src/EasyRSA-3.0.5/pki/private/ca.key: ******
/root/src

<INFO> - Mon Jan 14 11:35:07 UTC 2019 - Generating key and certificate for user alice
<INFO> - Mon Jan 14 11:35:07 UTC 2019 -
    IMPORTANT - make sure you remember ALL the passphrases!  You can download your certificate and key
after the setup.  - IMPORTANT
```

```
34
35 Signature ok
36 subject=CN = alice, emailAddress = alice@examplerun.cf
37 Getting CA Private Key
38 Enter pass phrase for /etc/ssl/private/examplerun.cf.ca.key:
39
40 <INFO> - Mon Jan 14 11:35:13 UTC 2019 -
       IMPORTANT - certificate and key for the user "alice" are saved to his home.  He can download it
41 later over a secure SSH connection - IMPORTANT
42
43 <INFO> - Mon Jan 14 11:35:13 UTC 2019 - Cleaning up..
44 <INFO> - Mon Jan 14 11:35:13 UTC 2019 - Creating zip file for alice user artifacts
45   adding: id_rsa (deflated 24%)
46   adding: id_rsa.pub (deflated 20%)
47   adding: alice.examplerun.cf.clientcert.pem (deflated 27%)
48  <INFO> - Mon Jan 14 11:35:13 UTC 2019 - This is your command to download your files
       to your local directory (rsync needs to be installed on your client):
49  rsync -e \ssh -i PATH_TO_YOUR_SSH_PRIVATE_KEY" --remove-source-files -av alice@examplerun.cf:/home/ali
50  ce/alice_artifacts.zip ./
51  <INFO> - Mon Jan 14 11:35:13 UTC 2019 - Finishing up, restarting services
52
53 <INFO> - Mon Jan 14 11:35:13 UTC 2019 - Restarting all components of the mailserver
54
55 <INFO> - Mon Jan 14 11:35:17 UTC 2019 - Mailserver configuration complete.
56 <Mail> - Mon Jan 14 11:35:17 UTC 2019 - Actions on Mail Done
```

2.7.2 E-Mail process diagram

Here we have process diagram of how the script works with all possible outcomes.

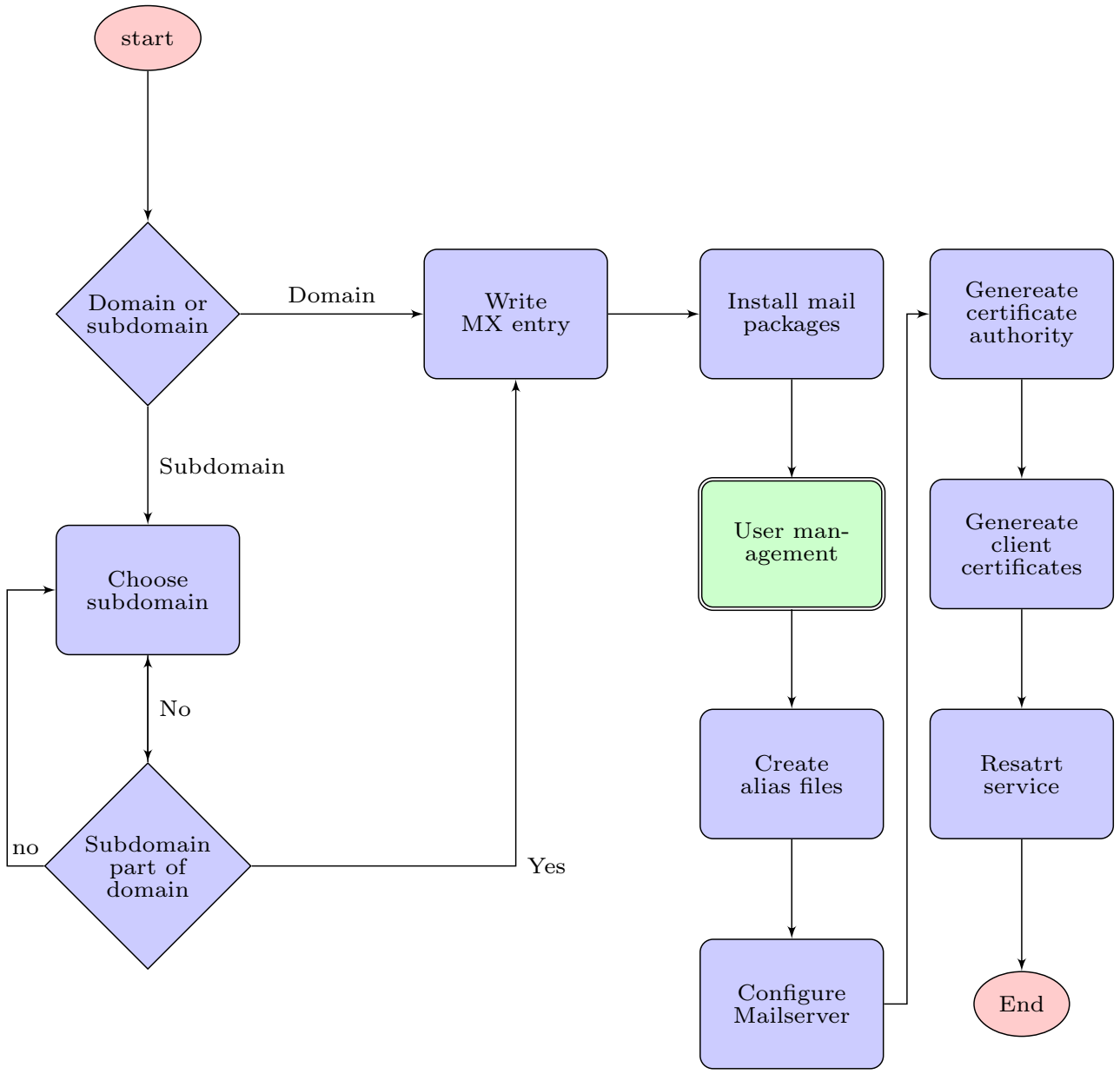


Figure 2.7: Email process diagram

2.7.3 Multiple e-mail addresses

With the user management you can create multiple users. All of them will get their own mail address. In this version of the script it is not possible to have multiple mail addresses per user. See subsection 5.1.2.

2.8 Web

The web part was developed in addition to the mandatory requirements. It runs through without the user having to do anything. For these reasons (especially the second one), the descriptions are also rather small. The code snippets here above show how a clean run without errors looks like. In the web part, as in all other parts, everything necessary will be installed first.

```
<INFO> - Tue Jan  8 11:24:21 UTC 2019 - Starting WEB Configurations.
<INFO> - Tue Jan  8 11:24:21 UTC 2019 - Will install 'nginx' now. Please wait...
...........
<INFO> - Tue Jan  8 11:24:32 UTC 2019 - Package 'nginx' is installed now.
<INFO> - Tue Jan  8 11:24:33 UTC 2019 - Will install 'certbot' now. Please wait...
....
<INFO> - Tue Jan  8 11:24:36 UTC 2019 - Package 'certbot' is installed now.
<INFO> - Tue Jan  8 11:24:36 UTC 2019 - Will install 'python-certbot-nginx' now.
    Please wait...
.......
<INFO> - Tue Jan  8 11:24:43 UTC 2019 - Package 'python-certbot-nginx' is installed
    now.
<INFO> - Tue Jan  8 11:24:43 UTC 2019 - Will install 'apache2' now. Please wait...
..............
<INFO> - Tue Jan  8 11:24:57 UTC 2019 - Package 'apache2' is installed now.
```

As the next step after installation, the nginix is configured.

```
<INFO> - Tue Jan  8 11:24:57 UTC 2019 - Starting nginx Configurations.
<INFO> - Tue Jan  8 11:24:57 UTC 2019 - Nginx is already activated.
<INFO> - Tue Jan  8 11:24:57 UTC 2019 - Start Nginx Hardening. (TLS, redirect http->
    https, secuirty headers, no server token, timeouts)
```

With openssl a certificate will be created in a next step. The certificate is then used for ssl termination.

```
<INFO> - Tue Jan  8 11:24:57 UTC 2019 - Start openssl to generate a ssl pem file.
Generating DSA parameters, 4096 bit long prime
.............+.......+.....+.........+..............
++++++++++++++++++++++++++++++++++++++++++++++++++++++++*
.............+..+.+.................+.....+...+.....
.........+......+.......+............................
<INFO> - Tue Jan  8 11:25:08 UTC 2019 - Done. Your file is located here: /etc/ssl/
    dh4096.pem. Will start certbot.
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator nginx, Installer nginx
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for examplerun.cf
http-01 challenge for www.examplerun.cf
Waiting for verification...
Cleaning up challenges

IMPORTANT NOTES:
- Congratulations! Your certificate and chain have been saved at:
/etc/letsencrypt/live/examplerun.cf/fullchain.pem
Your key file has been saved at:
/etc/letsencrypt/live/examplerun.cf/privkey.pem
Your cert will expire on 2019-04-08. To obtain a new or tweaked
version of this certificate in the future, simply run certbot
again. To non-interactively renew *all* of your certificates, run
"certbot renew"
- If you like Certbot, please consider supporting our work by:

Donating to ISRG / Lets Encrypt:   https://letsencrypt.org/donate
Donating to EFF:                    https://eff.org/donate-le
```

Nginx will then be hardened [12]:

- Enable secure SSL protocols only (>=TLSv1.2)
- Secure cipher sets (no known vulnerabilities)
- Redirect all connections from HTTP to HTTPS
- Turn off server tokens

```
<INFO> - Tue Jan  8 11:25:19 UTC 2019 - Will remove default sites of nginx
<INFO> - Tue Jan  8 11:25:19 UTC 2019 - Will start to setup nginx.conf file
<INFO> - Tue Jan  8 11:25:19 UTC 2019 - Done. Your file is located under '/etc/nginx/
    nginx.conf'.
<INFO> - Tue Jan  8 11:25:19 UTC 2019 - Will start specific Configurations
<INFO> - Tue Jan  8 11:25:19 UTC 2019 - Done. Your file is located under '/etc/nginx/
    conf.d/examplerun.cf.conf'.
<INFO> - Tue Jan  8 11:25:19 UTC 2019 - Will check Syntax and activate.
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

In the next and last step the apache will be configured. This setup places apache behind nginx as pure webserver. All connections are passed through nginx where SSL is terminated. Later on it would be possible to extend this setup with a WAF like ModSecurity which would provide an aditional security layer. See section 5.2.

```
<INFO> - Tue Jan  8 11:25:19 UTC 2019 - Starting apache Configurations.
<INFO> - Tue Jan  8 11:25:19 UTC 2019 - Apache is already activated.
<INFO> - Tue Jan  8 11:25:19 UTC 2019 - Found enabled default site, removing symlink
<INFO> - Tue Jan  8 11:25:19 UTC 2019 - Will Setup a default mini webpage.
<INFO> - Tue Jan  8 11:25:19 UTC 2019 - Will Setup a seperate ports.conf file.
<INFO> - Tue Jan  8 11:25:19 UTC 2019 - Will Setup avaible sites.
<INFO> - Tue Jan  8 11:25:19 UTC 2019 - Will check Syntax and activate.
Syntax OK
```

2.8.1 Web architecture diagram

For a better understanding of how the proxy server interacts with the web server, see this small diagram.

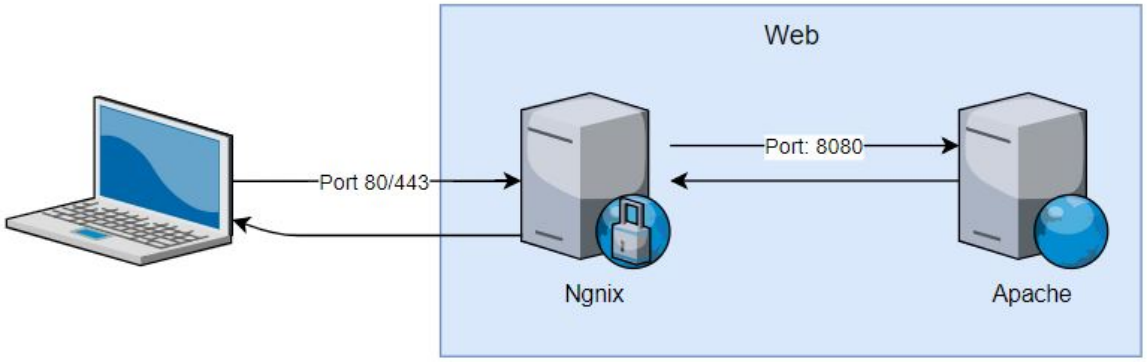


Figure 2.8: Architecture Web

3 Hardening Tests

3.1 Firewall

The firewall is an important factor in security. Open or incorrectly configured ports can quickly make a server vulnerable, especially if you have other components running on it. The firewall was tested with nmap [3]

BEFORE script

It should also be mentioned that the "before" run looks worse than the "after" run at first sight (more open ports). This is because ports needed for the components must be opened. The rest of the traffic is safely closed for this, so the server owner has control over it.

```
Starting Nmap ( https://nmap.org ) at 2019-01-11 09:17 UTC
NSE: Loaded 40 scripts for scanning.
Initiating Ping Scan at 09:17
Scanning 104.248.137.212 [4 ports]
Completed Ping Scan at 09:17, 0.22s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 09:17
Scanning mail.examplerun.cf (104.248.137.212) [100 ports]
Discovered open port 22/tcp on 104.248.137.212
Completed SYN Stealth Scan at 09:17, 1.26s elapsed (100 total ports)
Initiating Service scan at 09:17
Scanning 1 service on mail.examplerun.cf (104.248.137.212)
Completed Service scan at 09:17, 0.05s elapsed (1 service on 1 host)
NSE: Script scanning 104.248.137.212.
Initiating NSE at 09:17
Completed NSE at 09:17, 0.00s elapsed
Initiating NSE at 09:17
Completed NSE at 09:17, 0.00s elapsed
Nmap scan report for mail.examplerun.cf (104.248.137.212)
Host is up (0.029s latency).
Not shown: 99 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
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 2.45 seconds
           Raw packets sent: 104 (4.552KB) | Rcvd: 101 (4.044KB)
```

Figure 3.1: Firewall (without DNS) **BEFORE**

```
Starting Nmap ( https://nmap.org ) at 2019-01-11 09:14 UTC
NSE: Loaded 40 scripts for scanning.
Initiating Ping Scan at 09:14
Scanning examplerun.cf (104.248.137.212) [4 ports]
Completed Ping Scan at 09:14, 0.22s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 09:14
Scanning examplerun.cf (104.248.137.212) [100 ports]
Discovered open port 22/tcp on 104.248.137.212
Discovered open port 53/tcp on 104.248.137.212
Discovered open port 80/tcp on 104.248.137.212
Completed SYN Stealth Scan at 09:14, 1.34s elapsed (100 total ports)
Initiating Service scan at 09:14
Scanning 3 services on examplerun.cf (104.248.137.212)
Completed Service scan at 09:14, 6.03s elapsed (3 services on 1 host)
NSE: Script scanning 104.248.137.212.
Initiating NSE at 09:14
Completed NSE at 09:14, 0.06s elapsed
Initiating NSE at 09:14
Completed NSE at 09:14, 0.00s elapsed
Nmap scan report for examplerun.cf (104.248.137.212)
Host is up (0.013s latency).
Not shown: 97 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
53/tcp open  domain
80/tcp open  http    nginx 1.14.0 (Ubuntu)
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 8.43 seconds
           Raw packets sent: 136 (5.960KB) | Rcvd: 188 (8.924KB)
```

Figure 3.2: Firewall (with DNS) **BEFORE**

AFTER script

```
Starting Nmap ( https://nmap.org ) at 2019-01-11 09:29 UTC
NSE: Loaded 40 scripts for scanning.
Initiating Ping Scan at 09:29
Scanning examplerun.cf (104.248.137.212) [4 ports]
Completed Ping Scan at 09:29, 0.23s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 09:29
Scanning examplerun.cf (104.248.137.212) [100 ports]
Discovered open port 80/tcp on 104.248.137.212
Discovered open port 25/tcp on 104.248.137.212
Discovered open port 443/tcp on 104.248.137.212
Discovered open port 993/tcp on 104.248.137.212
Discovered open port 53/tcp on 104.248.137.212
Discovered open port 143/tcp on 104.248.137.212
Discovered open port 22/tcp on 104.248.137.212
Completed SYN Stealth Scan at 09:30, 1.67s elapsed (100 total ports)
Initiating Service scan at 09:30
Scanning 7 services on examplerun.cf (104.248.137.212)
Completed Service scan at 09:30, 12.13s elapsed (7 services on 1 host)
NSE: Script scanning 104.248.137.212.
Initiating NSE at 09:30
Completed NSE at 09:30, 0.18s elapsed
Initiating NSE at 09:30
Completed NSE at 09:30, 0.00s elapsed
Nmap scan report for examplerun.cf (104.248.137.212)
Host is up (0.015s latency).
Not shown: 92 filtered ports
PORT    STATE  SERVICE  VERSION
22/tcp  open   ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
25/tcp  open   smtp     Postfix smtpd
53/tcp  open   domain
80/tcp  open   http     nginx
143/tcp open   imap     Dovecot imapd (Ubuntu)
443/tcp open   ssl/http nginx
465/tcp closed smtps
993/tcp open   ssl/imap Dovecot imapd (Ubuntu)
Service Info: Host:  mail.examplerun.cf; 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 15.15 seconds
           Raw packets sent: 203 (8.908KB) | Rcvd: 16 (692B)
```

Figure 3.3: Firewall setup **AFTER**

3.2 DNS

3.2.1 Domain name resolver

As you have a brand new server you are most probably have use a domain name resolver from a big company like Google, Cloudfare etc. But after the script you have your own resolver which is even better than the one which is by default configured.

BEFORE script

Before running the script you get a C from ***https://cmdns.dev.dns-oarc.net/*** [8]

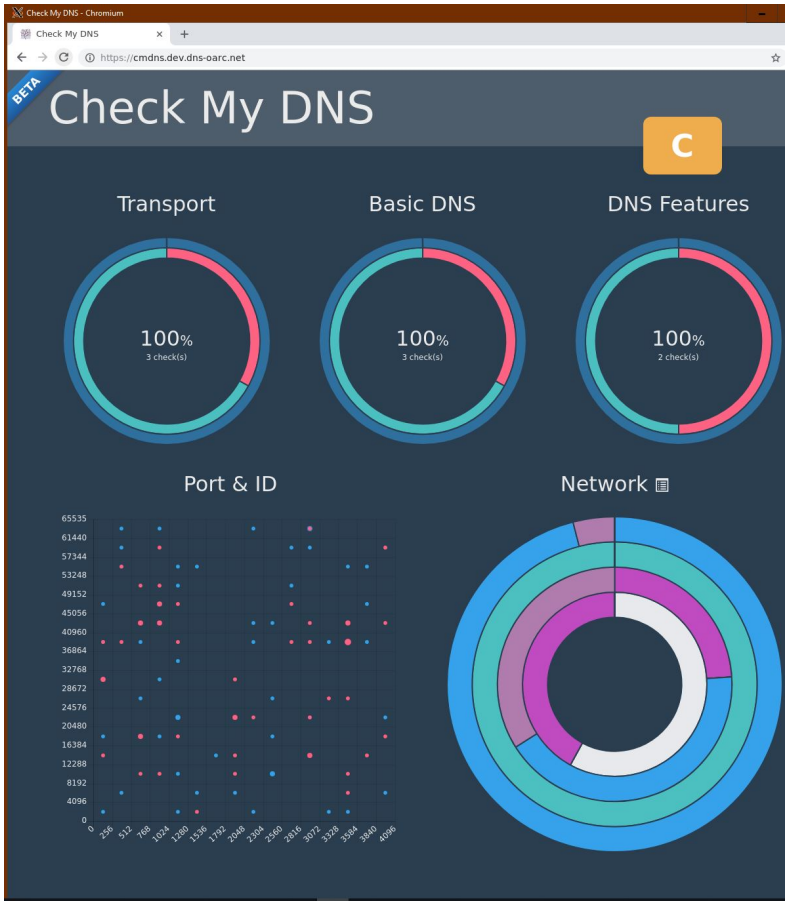


Figure 3.4: Name resolver **BEFORE**

Figure 3.5: Name resolver details **BEFORE**

AFTER script

After the script you have your own domain name resolver and a straight A.

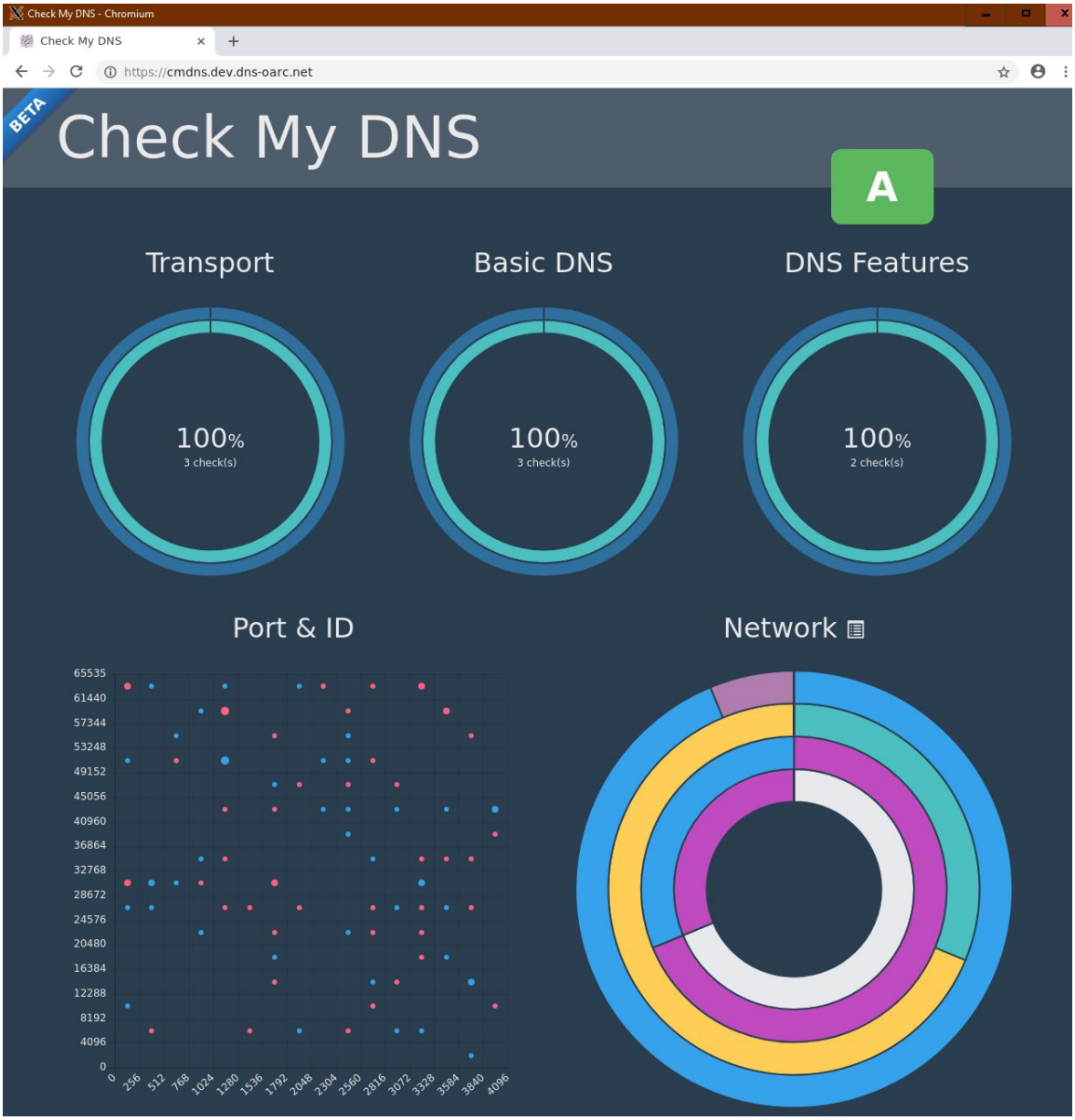


Figure 3.6: Name resolver **AFTER**

Figure 3.7: Name resolver details **AFTER**

3.2.2 Authoritative DNS

To setup a authoritative DNS is not easy, and mistakes are easily made.

BEFORE script

Before running the script if you do by hand, misconfiguration can happen. As you can see from ***https://mxtoolbox.com/*** [6].

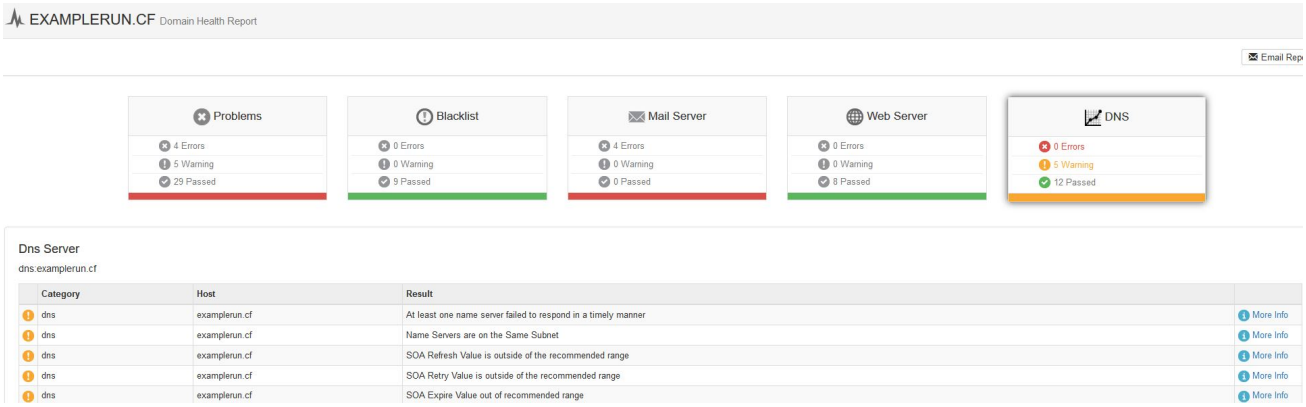


Figure 3.8: Authoritative DNS test **BEFORE**

After script

If you do it with the script, everything will be fine.

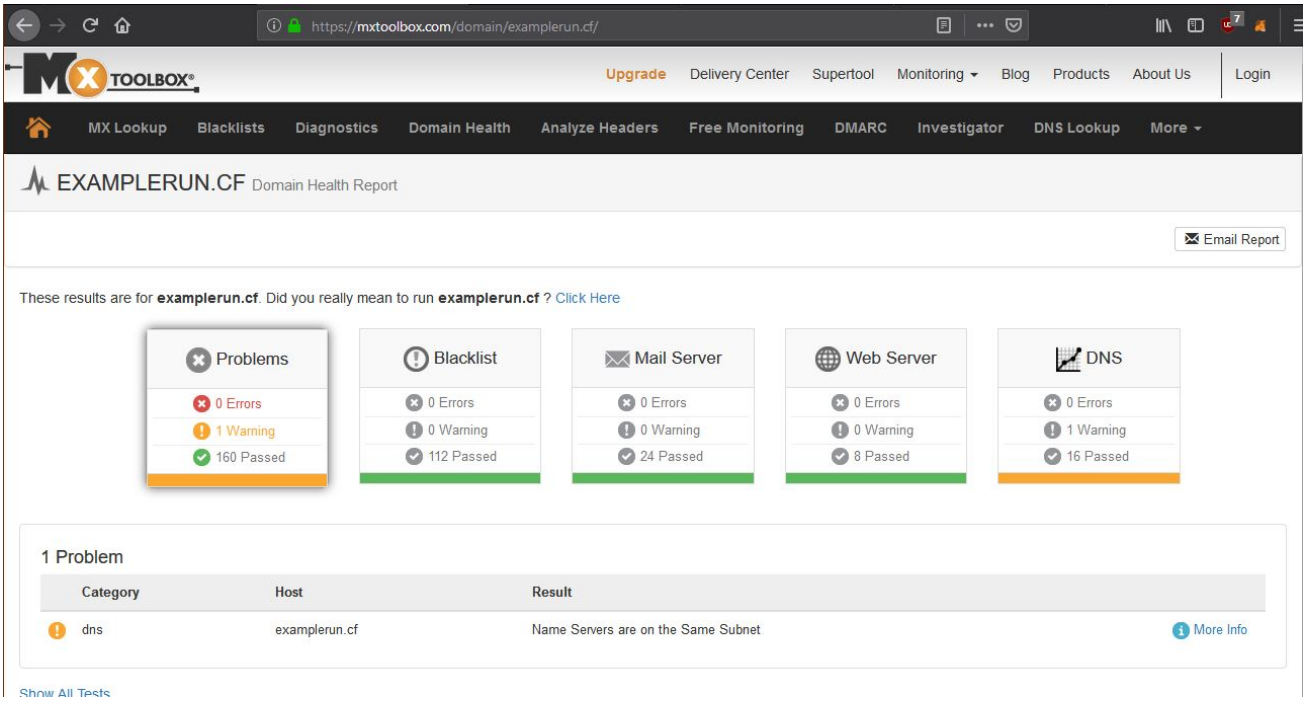


Figure 3.9: Authoritative DNS test **AFTER**

3.3 SSH

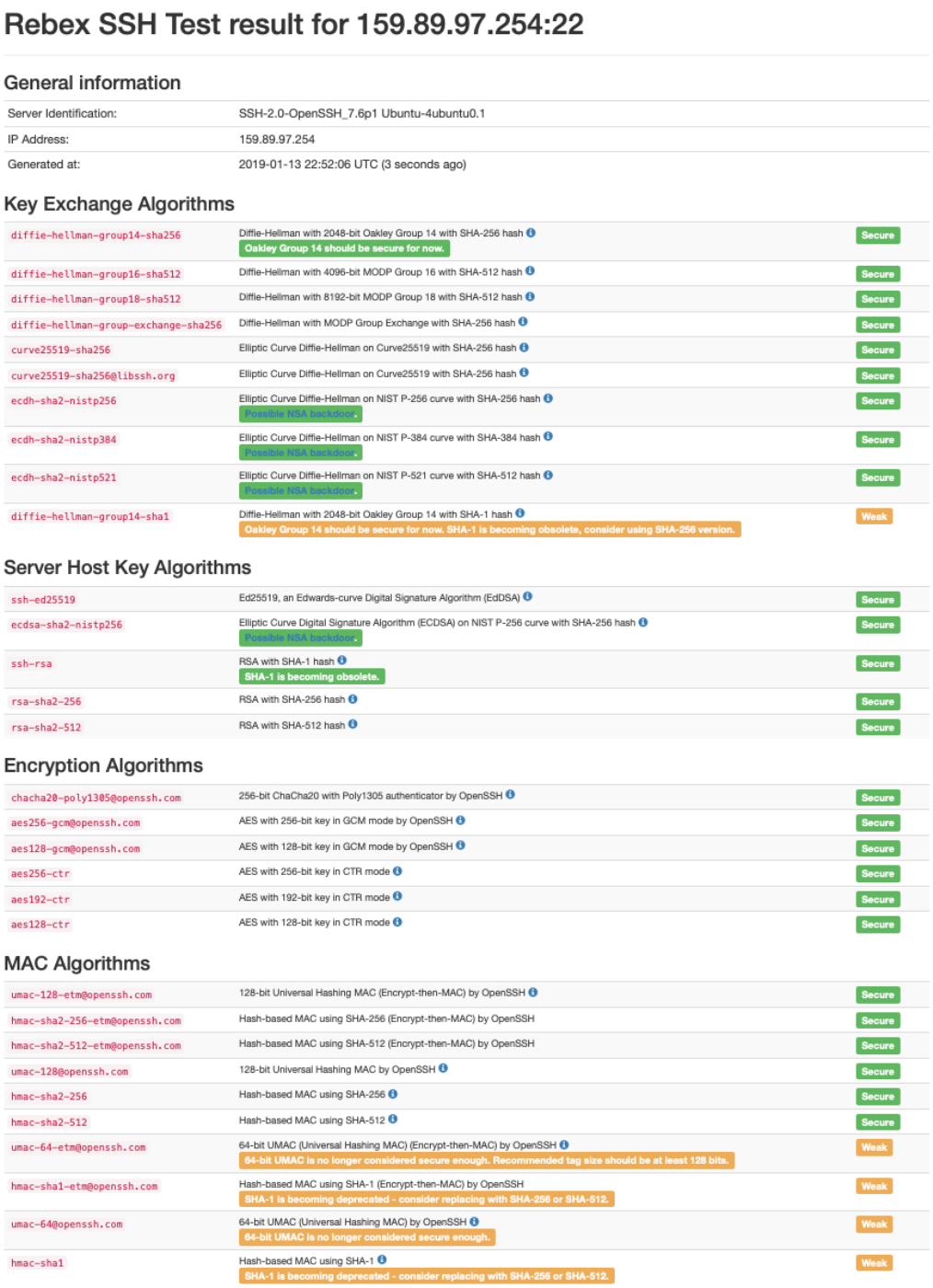
3.3.1 SSH daemon

You recieve a server with a default SSH daemon setup from your provider or have one at home with a default configuration from your Unix/Linux distro.

BEFORE script

Here we are testing a server with a default setup from https://digitalocean.com (the results might differ, depending where your server is hosted).

Before the SSH daemon is hardened we recieve the following result, some of the "Key Exchange Algorithms" and "MAC Algorithms" are weak [10].



Rebex SSH Test result for 159.89.97.254:22

General information

Server Identification:	SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.1
IP Address:	159.89.97.254
Generated at:	2019-01-13 22:52:06 UTC (3 seconds ago)

Key Exchange Algorithms

diffie-hellman-group14-sha256	Diffie-Hellman with 2048-bit Oakley Group 14 with SHA-256 hash Oakley Group 14 should be secure for now.	Secure
diffie-hellman-group16-sha512	Diffie-Hellman with 4096-bit MODP Group 16 with SHA-512 hash	Secure
diffie-hellman-group18-sha512	Diffie-Hellman with 8192-bit MODP Group 18 with SHA-512 hash	Secure
diffie-hellman-group-exchange-sha256	Diffie-Hellman with MODP Group Exchange with SHA-256 hash	Secure
curve25519-sha256	Elliptic Curve Diffie-Hellman on Curve25519 with SHA-256 hash	Secure
curve25519-sha256@libssh.org	Elliptic Curve Diffie-Hellman on Curve25519 with SHA-256 hash	Secure
ecdh-sha2-nistp256	Elliptic Curve Diffie-Hellman on NIST P-256 curve with SHA-256 hash Possible NSA backdoor.	Secure
ecdh-sha2-nistp384	Elliptic Curve Diffie-Hellman on NIST P-384 curve with SHA-384 hash Possible NSA backdoor.	Secure
ecdh-sha2-nistp521	Elliptic Curve Diffie-Hellman on NIST P-521 curve with SHA-512 hash Possible NSA backdoor.	Secure
diffie-hellman-group14-sha1	Diffie-Hellman with 2048-bit Oakley Group 14 with SHA-1 hash Oakley Group 14 should be secure for now. SHA-1 is becoming obsolete, consider using SHA-256 version.	Weak

Server Host Key Algorithms

ssh-ed25519	Ed25519, an Edwards-curve Digital Signature Algorithm (EdDSA)	Secure
ecdsa-sha2-nistp256	Elliptic Curve Digital Signature Algorithm (ECDSA) on NIST P-256 curve with SHA-256 hash Possible NSA backdoor.	Secure
ssh-rsa	RSA with SHA-1 hash SHA-1 is becoming obsolete.	Secure
rsa-sha2-256	RSA with SHA-256 hash	Secure
rsa-sha2-512	RSA with SHA-512 hash	Secure

Encryption Algorithms

chacha20-poly1305@openssh.com	256-bit ChaCha20 with Poly1305 authenticator by OpenSSH	Secure
aes256-gcm@openssh.com	AES with 256-bit key in GCM mode by OpenSSH	Secure
aes128-gcm@openssh.com	AES with 128-bit key in GCM mode by OpenSSH	Secure
aes256-ctr	AES with 256-bit key in CTR mode	Secure
aes192-ctr	AES with 192-bit key in CTR mode	Secure
aes128-ctr	AES with 128-bit key in CTR mode	Secure

MAC Algorithms

umac-128-etm@openssh.com	128-bit Universal Hashing MAC (Encrypt-then-MAC) by OpenSSH	Secure
hmac-sha2-256-etm@openssh.com	Hash-based MAC using SHA-256 (Encrypt-then-MAC) by OpenSSH	Secure
hmac-sha2-512-etm@openssh.com	Hash-based MAC using SHA-512 (Encrypt-then-MAC) by OpenSSH	Secure
umac-128@openssh.com	128-bit Universal Hashing MAC by OpenSSH	Secure
hmac-sha2-256	Hash-based MAC using SHA-256	Secure
hmac-sha2-512	Hash-based MAC using SHA-512	Secure
umac-64-etm@openssh.com	64-bit UMAC (Universal Hashing MAC) (Encrypt-then-MAC) by OpenSSH 64-bit UMAC is no longer considered secure enough. Recommended tag size should be at least 128 bits.	Weak
hmac-sha1-etm@openssh.com	Hash-based MAC using SHA-1 (Encrypt-then-MAC) by OpenSSH SHA-1 is becoming deprecated - consider replacing with SHA-256 or SHA-512.	Weak
umac-64@openssh.com	64-bit UMAC (Universal Hashing MAC) by OpenSSH 64-bit UMAC is no longer considered secure enough.	Weak
hmac-sha1	Hash-based MAC using SHA-1 SHA-1 is becoming deprecated - consider replacing with SHA-256 or SHA-512.	Weak

Figure 3.10: SSH daemon **BEFORE**

AFTER script

After the script is run and the SSH daemon is hardened only secure algorithms are used.

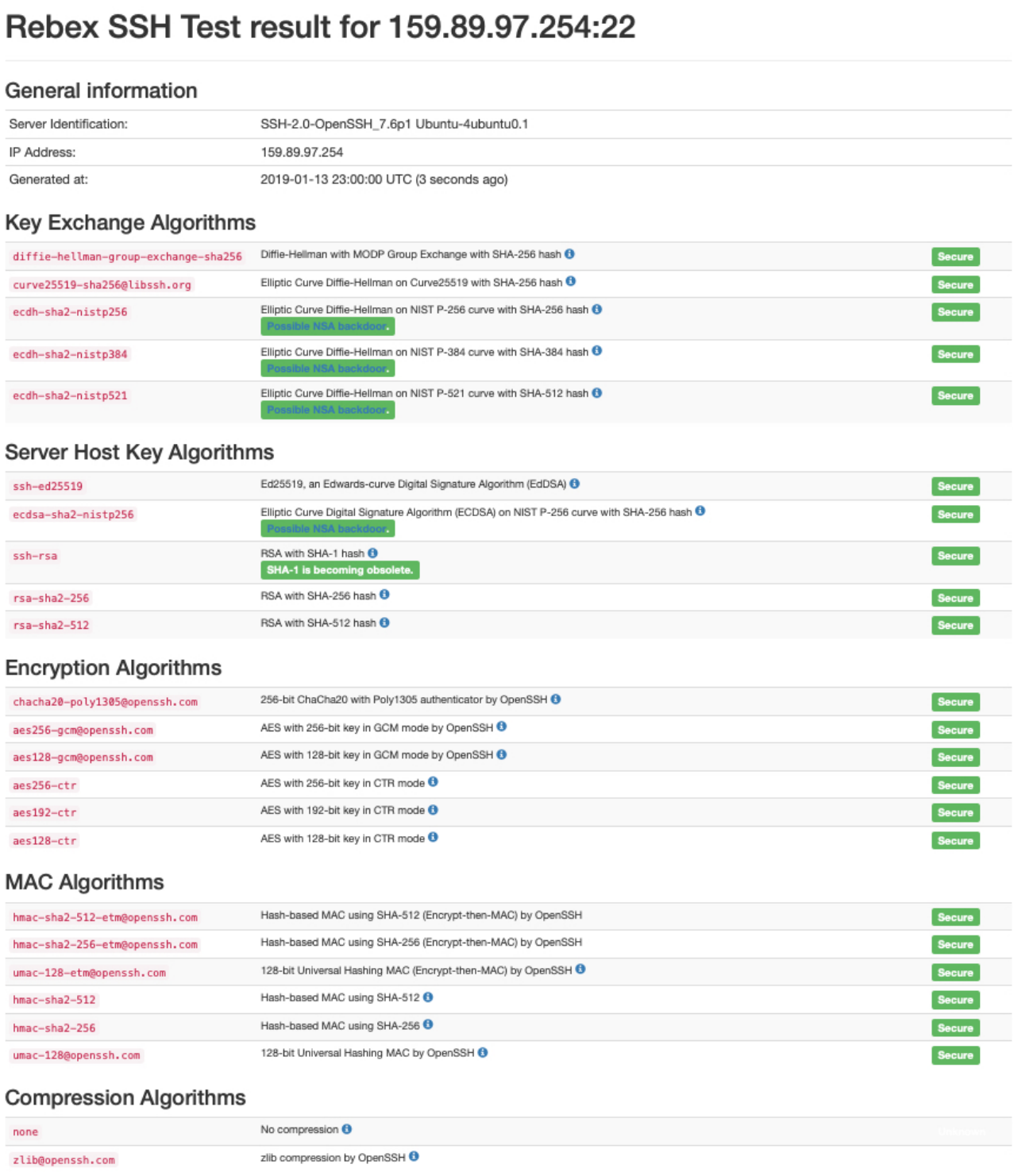


Figure 3.11: SSH daemon **AFTER**

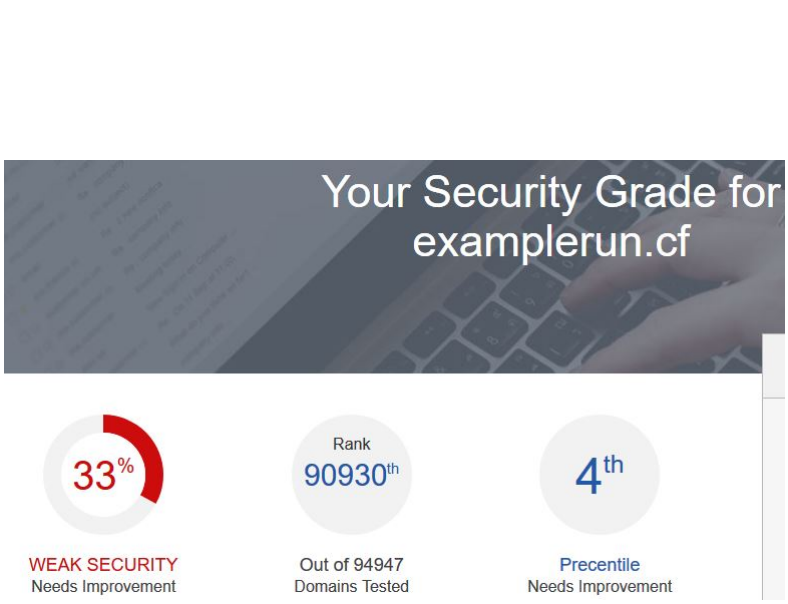
3.4 E-Mail

To run a E-Mail server is not easy at all. Even professional providers which should setup your email server for you do mostly mistakes. A insecure email server is also very attractive for hackers.

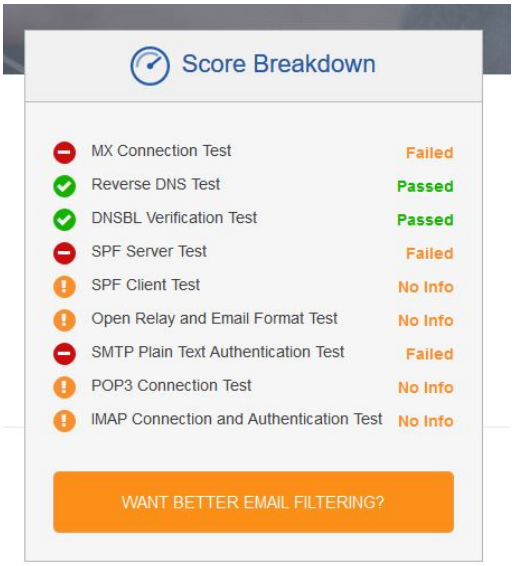
3.4.1 E-Mail server configuration

BEFORE script

If you use a basic email configuration, your email server will mostly look like this (graded from: ***https://www.hardenize.com*** [2], ***https://emailsecuritygrader.com*** [13])



(a) Rank



(b) Details

Figure 3.12: Mail **BEFORE** (emailsecuritygrader.com)

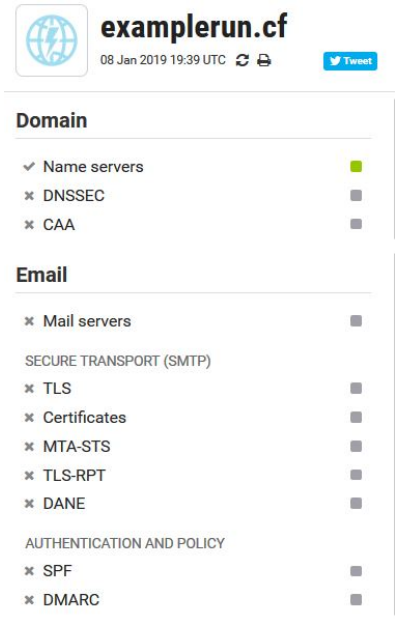
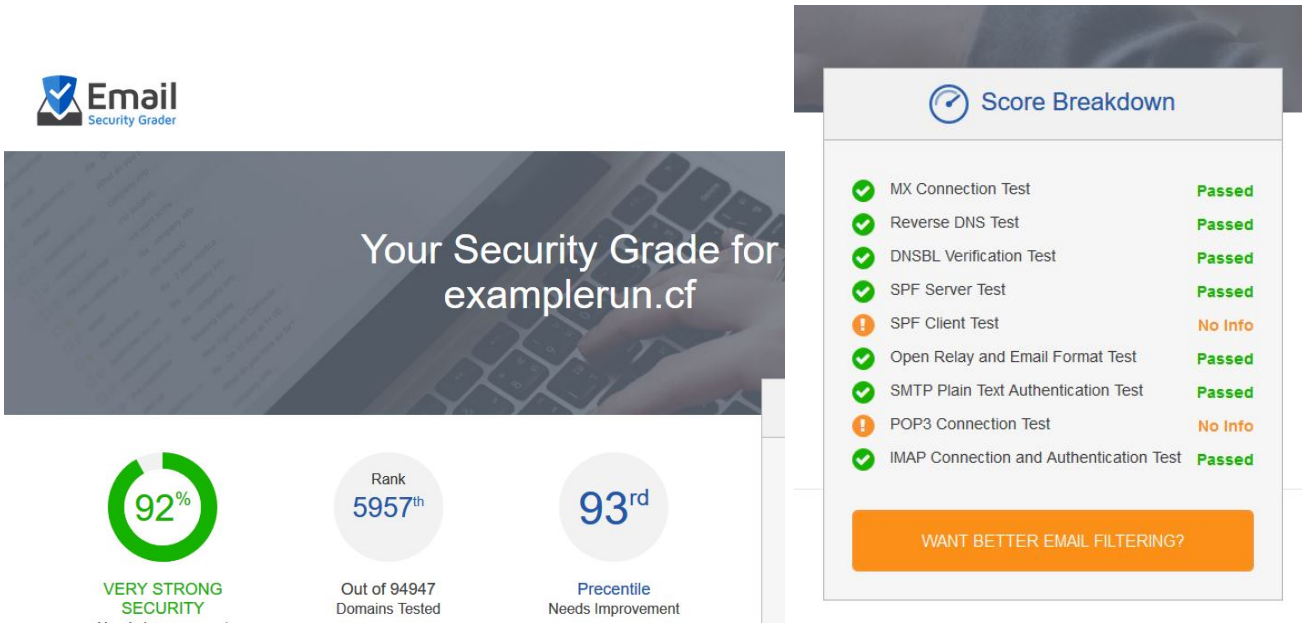


Figure 3.13: Mail **BEFORE** (hardenize.com)

AFTER script

But if you configure your email server with the script, it will look like this:



(a) Rank (b) Details

Figure 3.14: Mail **AFTER** (emailsecuritygrader.com)

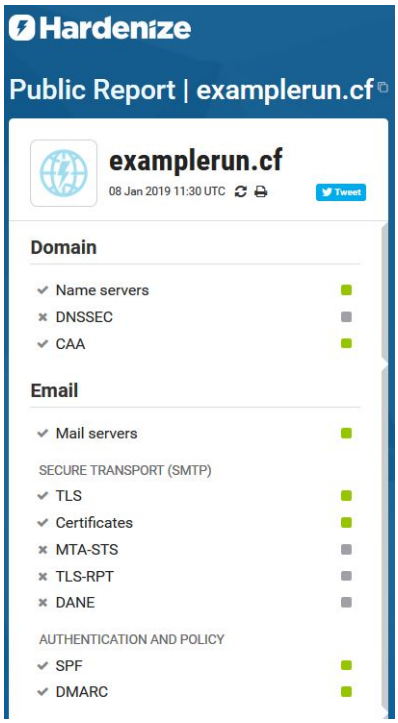


Figure 3.15: Mail **AFTER** (hardenize.com)

3.4.2 E-Mail header

As well if you don't want to end as SPAM your email header should be perfect, if you use the basic configuration, it won't be.

BEFORE script

Here how your header looks like before the script:

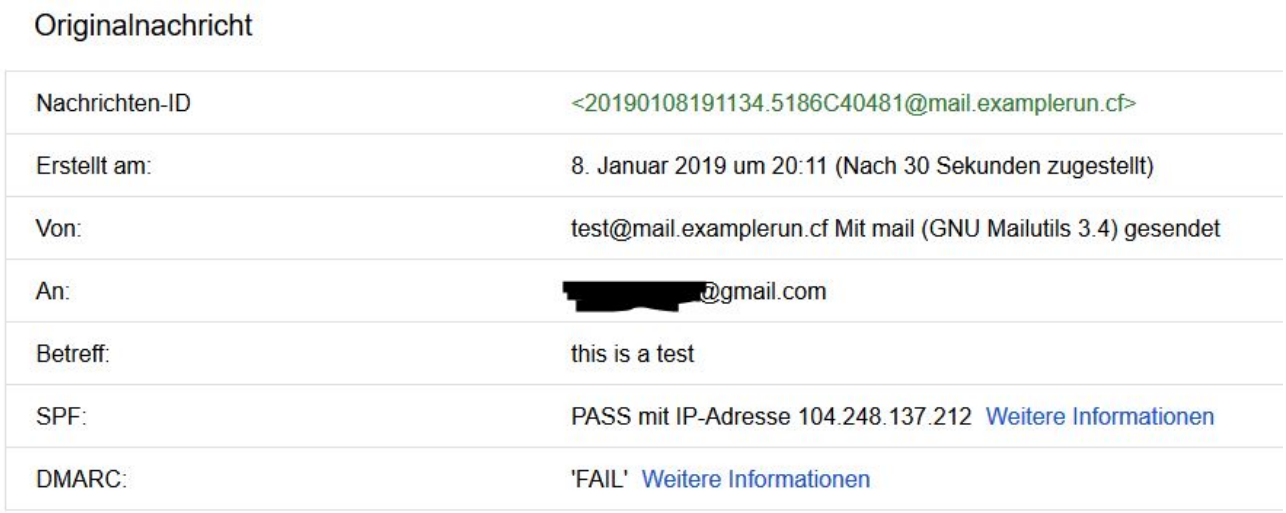


Figure 3.16: Mail header **BEFORE**

AFTER script

And here after:

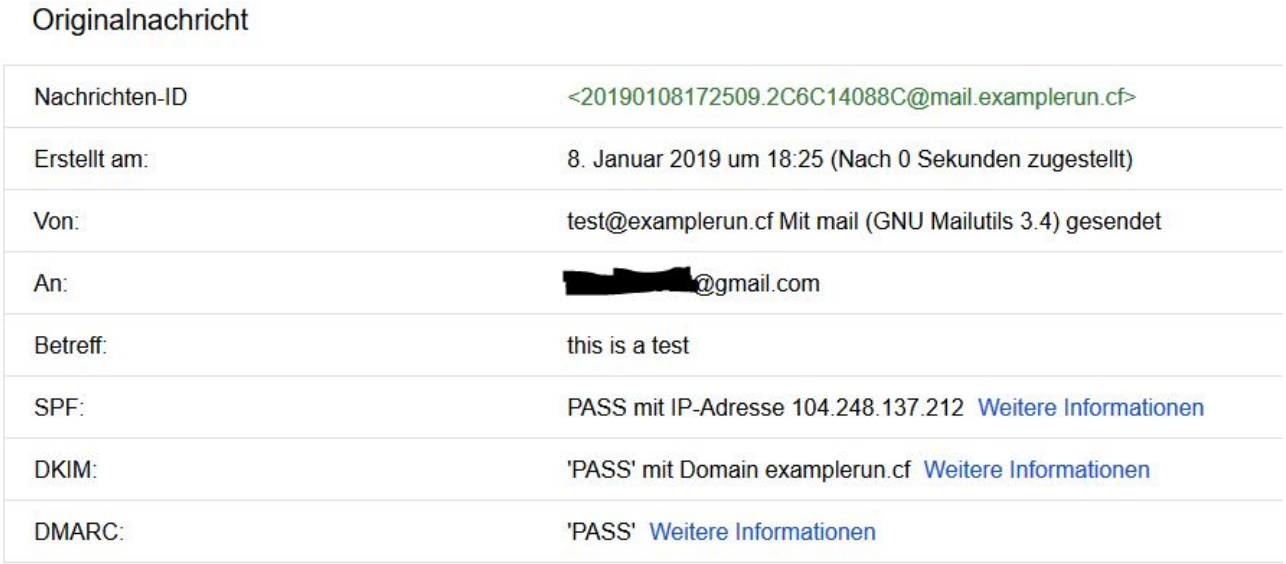


Figure 3.17: Mail header **AFTER**

3.5 Web

The web part could be tested very well with ***https://www.hardenize.com*** [2]. This is by the way the same tool/website with which the email part was checked.
To test the "before" part properly, an nginx had to be installed on the server in advance. It was not included by default on the servers used for testing. This is primarily about showing what it looks like when an unconfigured web service is on the internet versus made more secure with the script from this project.

BEFORE script

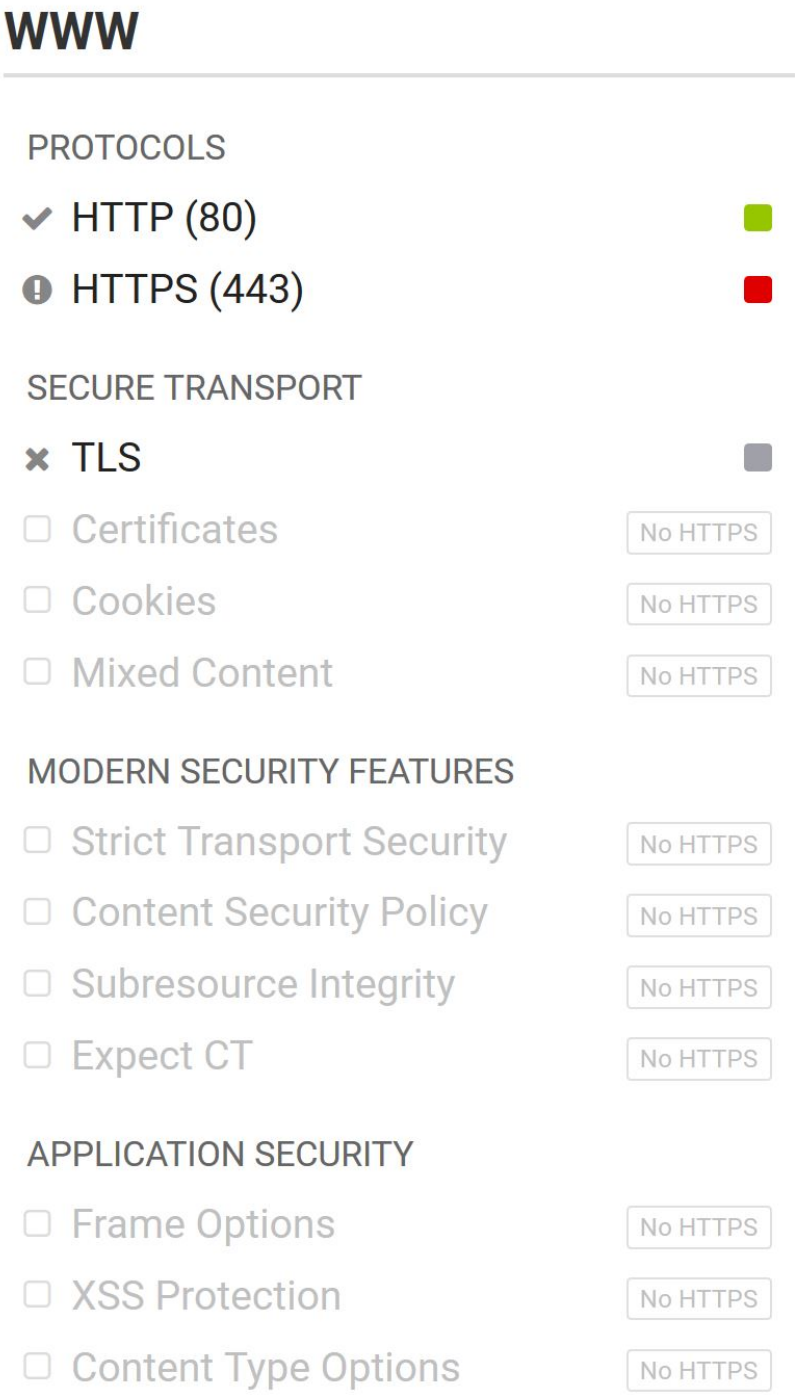


Figure 3.18: Web **BEFORE**

AFTER script

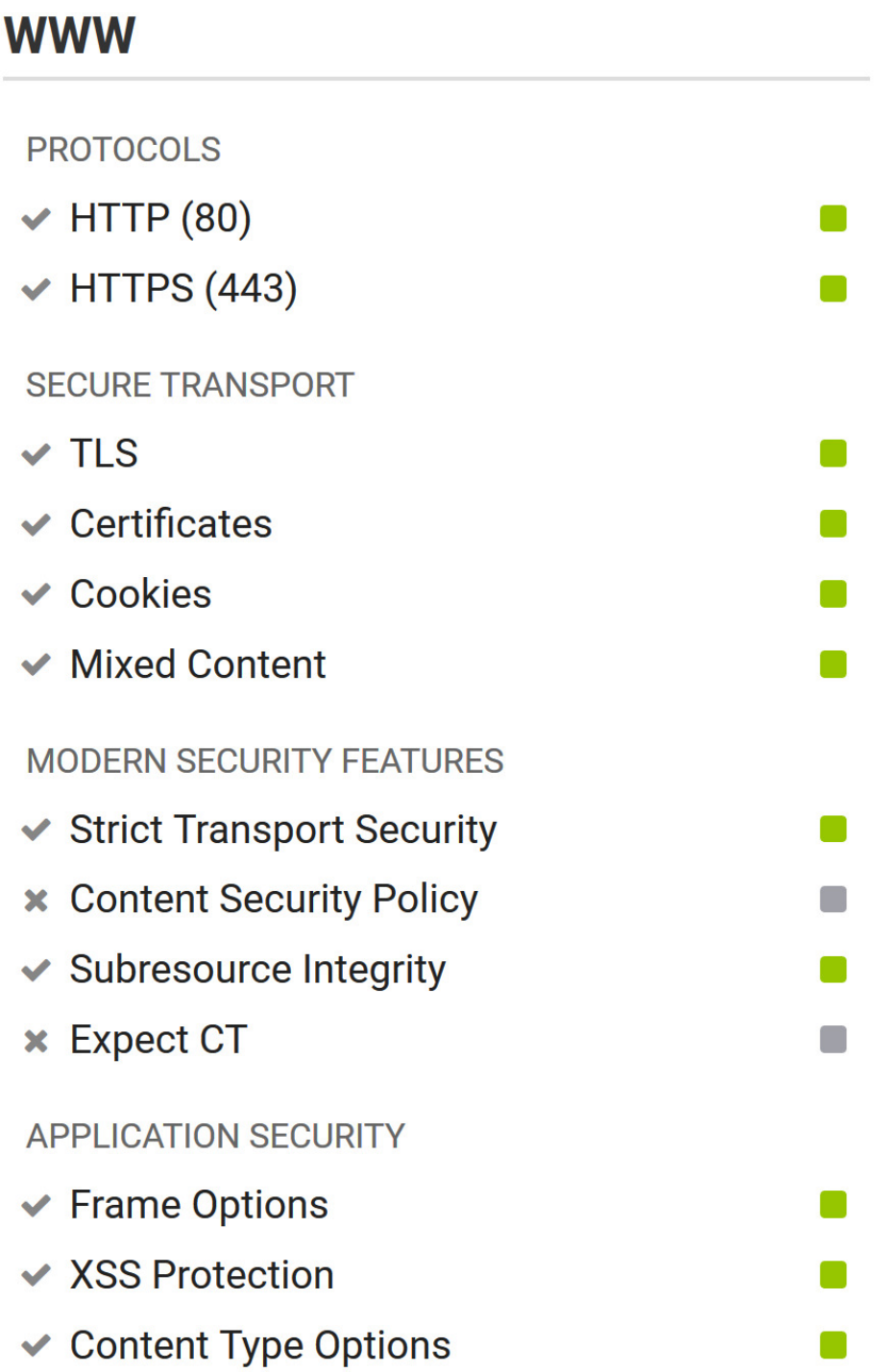


Figure 3.19: Web **AFTER**

4 E-Mail Client configuration

After you set up your secure email server you might want to configure your e-mail client.

The mailserver is only accessible through imaps and requires a TLS certificate for authentication. Therefor you need to set up your mail client with the appropriate configuration.

At the moment there is only one example for “Mail on macOS Mojave”.

4.1 Mail on macOS Mojave

4.1.1 Mail server config

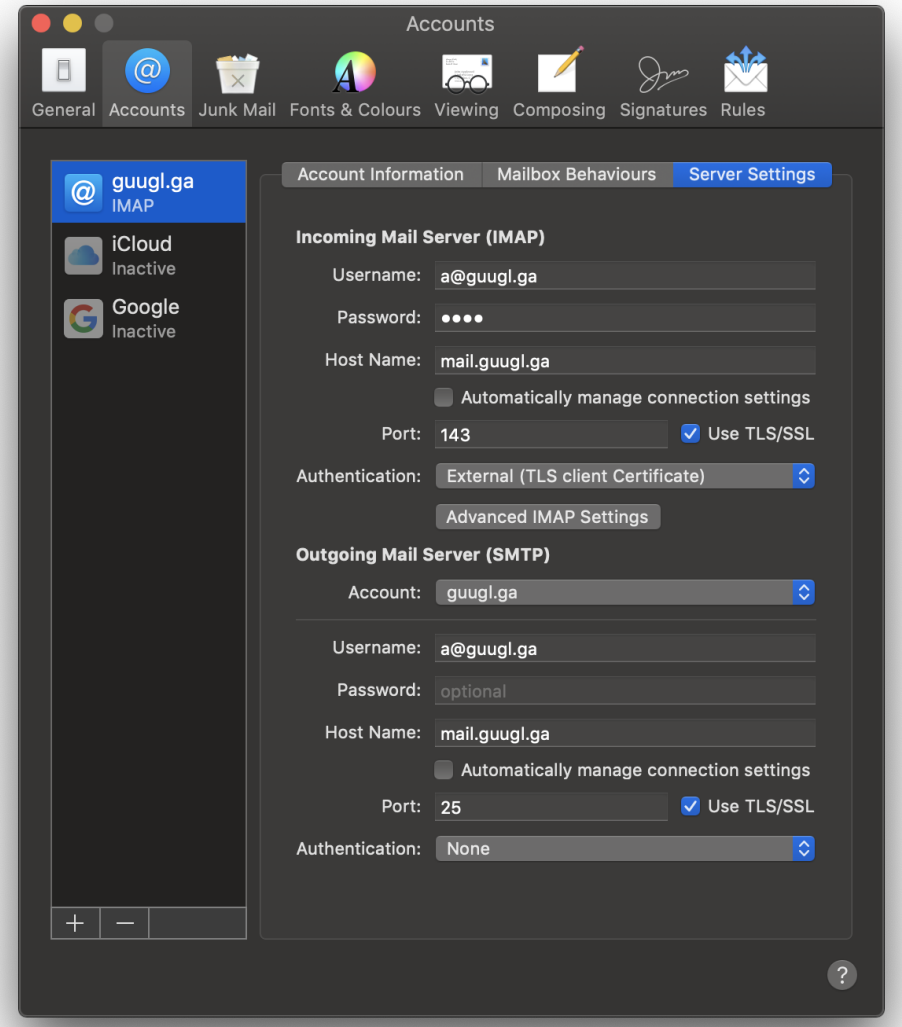


Figure 4.1: Mail server config

4.1.2 Mail SMTP settings

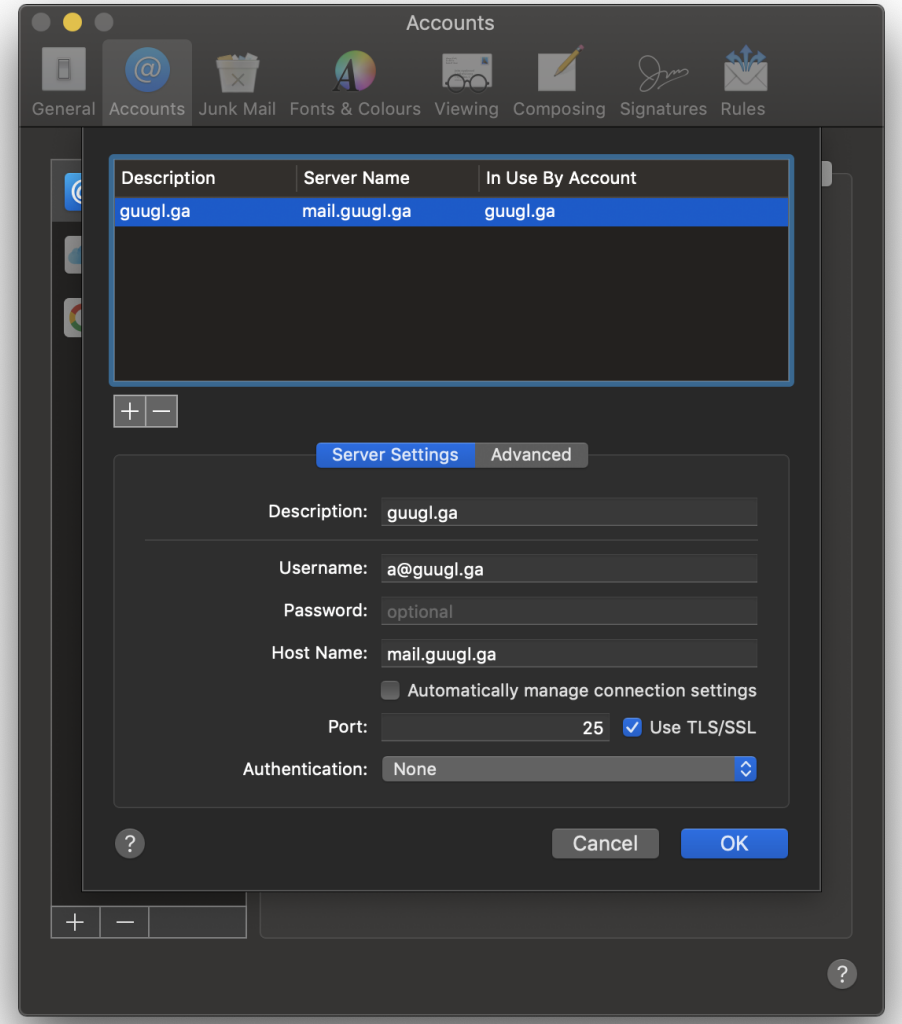


Figure 4.2: Mail SMTP settings

4.1.3 Mail IMAP TLS setting

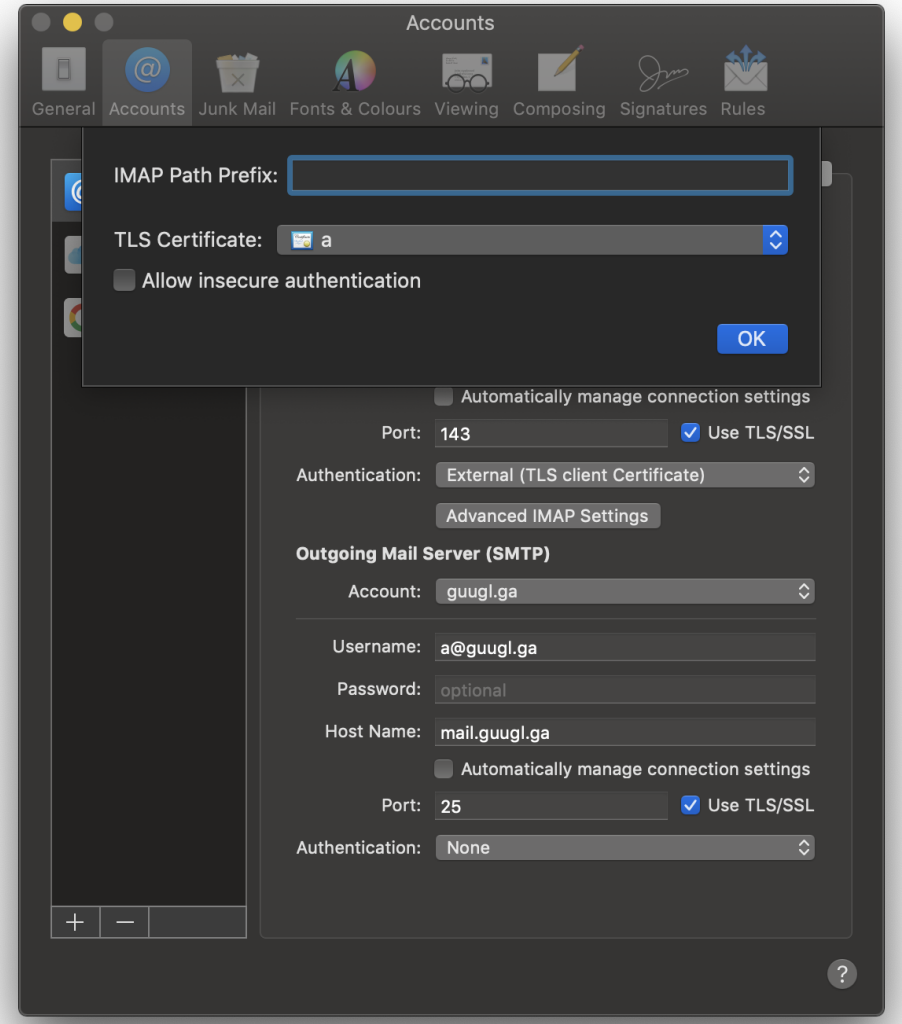


Figure 4.3: Mail IMAP TLS setting

5 Future Work

5.1 Extended functionalities

For somebody with basic needs the functionalities of this script is enough. But if we expand the spectrum, there are still some exciting features missing. Why not have more than one domains on the same server? Why not have multiple e-mail addresses? Why not choose your own address? This could be important for somebody who has a small company for example.

5.1.1 Multiple domains

The base to have multiple domains is already set. With NSD you have a perfect authoritative name server for multiple domains. NSD is not a hobby product, it is a very professional one. It is even used for some root domains (see: *https://en.wikipedia.org/wiki/NSD*). The function to make one domain zone is already here, so we "only" need to make more of it and guide the user through a new process.

5.1.2 Multiple e-mail addresses

Of course it would be interesting to have more than one email address per user. As well, if you want to create an email address it would be nice if you can choose your own local-part (everything before the @) of your address. Postfix is capable of all this things, but it won't do it by it self. This part sounds quite easy, but it is a complex process which is not defined and scripted yet.

5.1.3 Web application server

Instead of using Apache only as a plain web server it could be extended to act a PHP or CGI application server with a database. This could be helpful if the user would like to run small applications next to static website content.

5.2 More Hardening

After installing all components with the script, you have a decent hardened server. Still, it could be more secure! There are things we could not configure for you in this project like:

- TLS 1.3 : An update of TLS 1.2, faster and more secure.
  For more information about TLS 1.3, please check a the comparative study paper (TLS1.2vs1.3.pdf) from our colleges Kandiah Rajina and Doukmak Anna. You can find the PDF in the same directory.
- DNSSEC : To secure your domain, but it needs some interaction with your top-level domain registrar.
- E-Mail

  MTA-STS: For more security in sending and receiving emails.

  DANE: (needs DNSSEC) is a bridge between DNSSEC and TLS.

As well we would have liked to provide you some more components like:

- XMPP-IM WebRTC: For real-time communication.
- Tor Node: For growing the Tor network.

- Snort-IDS: For network intrusion detection and prevention.
- WAF: To add an existing layer of security to the webserver. Especially when the webserver acts as an application server.

All of those are candidates for future work. It may be done in a second project from our university... or you?

5.3 Containerization

The idea of containerization is to put every component into a Docker container. The main benefit would be that every component runs separated in a isolated environment.

- More modular: With a Docker container setup every component (DNS, SSH, Mail, Web) would run in a separate container, which would make the setup more modular.

  Note: The firewall is not useful in a container. It needs to be configured on the Docker host to redirect the necessary ports to the right container.

- Platform independent: With the use of Docker containers the project could be set up on any platform which supports a Docker Engine. This includes most of the modern Unix/Linux distributions and even Windows Systems. Inside of the containers there would still run a Ubuntu image.

5.4 Code Migration

Our script collection is exclusively implemented with bash. So we are close to the operating system and can directly fall back on commands of the operating system. Using other scripting languages or perhaps even a high-level language (object-oriented) would probably be a pay off. With code migration it is always a kind of 'trade off' between what one likes, becoming more modern and/or simplifying.

- Python:

  Also close to the linux operating system. Certain subrutienes would be simpler or smaller in python and in general python is better readable and therefore easier to maintain.

- Ansible:

  A very good example of modernization and machine independence. Ansible is very contemporary and migration to one or more ansible playbooks from our code would certainly be possible.

When it comes to code migration, it must be mentioned that the primary focus is not on creating new code parts, but on refactoring and migrating existing code. Of course, you can create new code in parallel, but you won't be able to avoid rewriting or moving existing code.

6 Conclusion

Every user who exchanges information over the internet should have the privilge to do this in a secure an anonymous matter. We built this script to provide every user, a maximal secure server, with a minimal need of information. Altough we tried to cover as much aspects and components as possible we saw, during our work, that there is much more to do. The further work which could build on top of our project are written down in chapter 5. After all, we learnt a lot for the future and are hoping to make the internet a little more secure for everyone of its users.

7 License

For all the work accomplished in this project we were inspired by a lot of resources. Especially by the book “Linux Hardening in Hostile Networks: Server Security from TLS to Tor” [9], which provided a lot of examples for our work. Furthermore a lot of very well written websites and online guides were used:

- Dovecot and Postfix client certificate authentication [11]
- DMARC Setup [1]
- Configuring HTTPS servers [12]

Nevertheless we paid close attention not to copy any code nor modify any of the components we use. Therefore all the outcome we produced in this project is our own work.

We decided to use the MIT license which has a wide acceptance in the open source community and fits our needs for license without warranty.

7.1 MIT license

8 Glossary

Ansible Ansible is open source software that automates software provisioning, configuration management, and application deployment. Ansible connects via SSH, remote PowerShell or via other remote APIs. .
SOURCE: https://en.wikipedia.org/wiki/Ansible_(software). 51

Apache The Apache HTTP Server, colloquially called Apache, is free and open-source cross-platform web server software, released under the terms of Apache License 2.0. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation.
SOURCE: https://en.wikipedia.org/wiki/Apache_HTTP_Server. 9

DANE DNS-based Authentication of Named Entities (DANE) is an Internet security protocol to allow X.509 digital certificates, commonly used for Transport Layer Security (TLS), to be bound to domain names using Domain Name System Security Extensions (DNSSEC)
SOURCE: https://en.wikipedia.org/wiki/DNS-based_Authentication_of_Named_Entities. 50

DKIM DomainKeys Identified Mail (DKIM) is an email authentication method designed to detect forged sender addresses in emails, (email spoofing), a technique often used in phishing and email spam.
SOURCE: https://en.wikipedia.org/wiki/DomainKeys_Identified_Mail. 9, 27

DMARC DMARC (Domain-based Message Authentication, Reporting and Conformance) is an email-validation system designed to detect and prevent email spoofing, the use of forged sender addresses often used in phishing and email spam. .
SOURCE: https://en.wikipedia.org/wiki/DMARC. 9, 27

DNS The Domain Name System (DNS) is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols.
SOURCE: https://en.wikipedia.org/wiki/Domain_Name_System. 9, 16

DNSSEC The Domain Name System Security Extensions (DNSSEC) is a suite of Internet Engineering Task Force (IETF) specifications for securing certain kinds of information provided by the Domain Name System (DNS) as used on Internet Protocol (IP) networks. It is a set of extensions to DNS which provide to DNS clients (resolvers) origin authentication of DNS data, authenticated denial of existence, and data integrity, but not availability or confidentiality.
SOURCE: https://en.wikipedia.org/wiki/Domain_Name_System_Security_Extensions. 50

Docker Docker is a computer program that performs operating-system-level virtualization, also known as "containerization". It was first released in 2013 and is developed by Docker, Inc.
SOURCE: https://en.wikipedia.org/wiki/Docker_(software). 51

Glue Records Glue Records, or Nameserver Glue, relate a nameserver on the internet to an IP address. This relationship is set up at the domain registrar for the main domain on which the nameservers were created.
SOURCE: https://www.liquidweb.com/kb/what-are-glue-records/. 17

IMAP In computing, the Internet Message Access Protocol (IMAP) is an Internet standard protocol used by email clients to retrieve email messages from a mail server over a TCP/IP connection.[1] IMAP is defined by RFC 3501.
SOURCE: `https://en.wikipedia.org/wiki/Internet_Message_Access_Protocol`. 29

ModSecurity ModSecurity, sometimes called Modsec, is an open-source web application firewall (WAF). Originally designed as a module for the Apache HTTP Server, it has evolved to provide an array of Hypertext Transfer Protocol request and response filtering capabilities along with other security features across a number of different platforms including Apache HTTP Server, Microsoft IIS and NGINX. It is a free software released under the Apache license 2.0.
SOURCE: `https://en.wikipedia.org/wiki/ModSecurity`. 33

MTA-STS MTA-STS (full name SMTP Mail Transfer Agent Strict Transport Security) is a new standard that aims to improve the security of SMTP by enabling domain names to opt into strict transport layer security mode that requires authentication (valid public certificates) and encryption (TLS).
SOURCE: `https://www.hardenize.com/blog/mta-sts`. 50

Nginx Nginx is a web server which can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache. The software was created by Igor Sysoev and first publicly released in 2004.[9] A company of the same name was founded in 2011 to provide support and Nginx plus paid software. Nginx is free and open-source software, released under the terms of a BSD-like license.
SOURCE: `https://en.wikipedia.org/wiki/Nginx`. 9

nsd In Internet computing, NSD (for "name server daemon") is an open-source Domain Name System (DNS) server. It was developed by NLnet Labs of Amsterdam in cooperation with the RIPE NCC, from scratch as an authoritative name server (i.e., not implementing the recursive caching function by design).
SOURCE: `https://en.wikipedia.org/wiki/NSD`. 9

Python Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.
SOURCE: `https://en.wikipedia.org/wiki/Python_(programming_language)`. 51

SMTP Simple Mail Transfer Protocol (SMTP) is an Internet standard for email transmission. First defined by RFC 821 in 1982, it was updated in 2008 with Extended SMTP additions by RFC 5321; which is the protocol in widespread use today.
SOURCE: `https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocolm`. 27

SPF Sender Policy Framework (SPF) is an email authentication method designed to detect forged sender addresses in emails (email spoofing), a technique often used in phishing and email spam.
SOURCE: `https://en.wikipedia.org/wiki/Sender_Policy_Framework`. 9, 27

SSH Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network.[1] Typical applications include remote command-line login and remote command execution, but any network service can be secured with SSH.
SOURCE: `https://en.wikipedia.org/wiki/Secure_Shell`. 23

SSL Transport Layer Security (TLS), and its now-deprecated predecessor, Secure Sockets Layer (SSL),[1] are cryptographic protocols designed to provide communications security over a computer network.[2] Several versions of the protocols find widespread use in applications such as web browsing, email, instant messaging, and voice over IP (VoIP). Websites can use TLS to secure all communications between their servers and web browsers.
SOURCE: `https://en.wikipedia.org/wiki/Transport_Layer_Security`. 29

TLS Transport Layer Security (TLS), and its now-deprecated predecessor, Secure Sockets Layer (SSL),[1] are cryptographic protocols designed to provide communications security over a computer network.[2] Several versions of the protocols find widespread use in applications such as web browsing, email, instant messaging, and voice over IP (VoIP). Websites can use TLS to secure all communications between their servers and web browsers.
SOURCE: `https://en.wikipedia.org/wiki/Transport_Layer_Security`. 27

Tor Tor is free software for enabling anonymous communication. The name is derived from an acronym for the original software project name "The Onion Router". Tor directs Internet traffic through a free, worldwide, volunteer overlay network consisting of more than seven thousand relays to conceal a user's location and usage from anyone conducting network surveillance or traffic analysis.
SOURCE: `https://en.wikipedia.org/wiki/Tor_(anonymity_network)`. 50

ufw The default firewall configuration tool for Ubuntu is ufw. Developed to ease iptables firewall configuration, ufw provides a user friendly way to create an IPv4 or IPv6 host-based firewall. By default UFW is disabled.
SOURCE: `https://help.ubuntu.com/community/UFW`. 13

unbound Unbound is a validating, recursive, and caching DNS resolver product from NLnet Labs. It is distributed free of charge in open-source form under the BSD license.
SOURCE: `https://en.wikipedia.org/wiki/Unbound_(DNS_server)`. 9

WAF A web application firewall (or WAF) filters, monitors, and blocks HTTP traffic to and from a web application. A WAF is differentiated from a regular firewall in that a WAF is able to filter the content of specific web applications while regular firewalls serve as a safety gate between servers. By inspecting HTTP traffic, it can prevent attacks stemming from web application security flaws, such as SQL injection, cross-site scripting (XSS), file inclusion, and security misconfigurations.
SOURCE: `https://en.wikipedia.org/wiki/Web_application_firewall`. 33, 51

WebRTC WebRTC (Web Real-Time Communication) is a free, open-source project that provides web browsers and mobile applications with real-time communication (RTC) via simple application programming interfaces (APIs). It allows audio and video communication to work inside web pages by allowing direct peer-to-peer communication, eliminating the need to install plugins or download native apps.Supported by Google, Microsoft, Mozilla, and Opera, WebRTC is being standardized through the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF).
SOURCE: `https://en.wikipedia.org/wiki/WebRTC`. 50

wild-card In software, a wildcard character is a kind of placeholder represented by a single character, such as an asterisk (*), which can be interpreted as a number of literal characters or an empty string. It is often used in file searches so the full name need not be typed.
SOURCE: `https://en.wikipedia.org/wiki/Wildcard_character`. 19

Bibliography

[1] Global Cyber Alliance. *DMARC Setup*. 2016. URL: https://dmarcguide.globalcyberalliance.org (visited on 10/24/2018).

[2] Hardenize. *Hardenize*. 2018. URL: https://www.hardenize.com (visited on 10/25/2018).

[3] Gordon "Fyodor" Lyon Insecure.Com LLC. *The Official Nmap Project Guide to Network Discovery and Security Scanning*. 2011. URL: https://nmap.org/book/port-scanning-tutorial.html (visited on 10/25/2018).

[4] Todd Knarr Linode. *Configure SPF and DKIM With Postfix on Debian 8*. 2018. URL: https://www.linode.com/docs/email/postfix/configure-spf-and-dkim-in-postfix-on-debian-8/ (visited on 10/25/2018).

[5] Michael Boelen Linux Audit. *OpenSSH security and hardening*. 2018. URL: https://linux-audit.com/audit-and-harden-your-ssh-configuration/ (visited on 10/25/2018).

[6] MXToolbox. *MX Lookup*. 2018. URL: https://mxtoolbox.com/ (visited on 10/25/2018).

[7] NLnet Labs. *unbound & nsd*. 2018. URL: https://github.com/NLnetLabs (visited on 10/25/2018).

[8] OARC, Inc. *Check My DNS*. 2017. URL: https://www.dns-oarc.net/oarc/services/cmdns (visited on 10/25/2018).

[9] Kyle Rankin. *Linux Hardening in Hostile Networks: Server Security from TLS to Tor*. Addison Wesley, 2017.

[10] Rebex. *Rebex*. 2018. URL: https://sshcheck.com (visited on 10/25/2018).

[11] Giel van Schijndel. *Dovecot and Postfix client certificate authentication*. 2017. URL: https://blog.mortis.eu/blog/2017/06/dovecot-and-postfix-with-client-cert-auth.html (visited on 10/24/2018).

[12] Igor Sysoev and Brian Mercer. *Configuring HTTPS servers*. 2018. URL: http://nginx.org/en/docs/http/configuring_https_servers.html (visited on 10/25/2018).

[13] Vircom. *Email Security Grader*. 2018. URL: https://emailsecuritygrader.com/ (visited on 10/25/2018).

Appendices

configFiles/dns/unbound/unbound.conf

```
include: "/etc/unbound/unbound.conf.d/*.conf"
```

configFiles/dns/unbound/access.conf

```
server:
```

configFiles/dns/unbound/hardening.conf

```
   ### SOURCE: https://calomel.org/unbound_dns.html ###
server:
   # enable to not answer id.server and hostname.bind queries.
    hide-identity: yes

   # enable to not answer version.server and version.bind queries.
    hide-version: yes

  # Read  the  root  hints from this file. Default is nothing, using built in
  # hints for the IN class. The file has the format of  zone files,  with  root
  # nameserver  names  and  addresses  only. The default may become outdated,
  # when servers change,  therefore  it is good practice to use a root-hints
  # file.  get one from https://www.internic.net/domain/named.root
    root-hints: "/var/lib/unbound/root.hints"

    # Will trust glue only if it is within the servers authority.
  # Harden against out of zone rrsets, to avoid spoofing attempts.
  # Hardening queries multiple name servers for the same data to make
  # spoofing significantly harder and does not mandate dnssec.
    harden-glue: yes

  # Require DNSSEC data for trust-anchored zones, if such data is absent, the
  # zone becomes  bogus.  Harden against receiving dnssec-stripped data. If you
  # turn it off, failing to validate dnskey data for a trustanchor will trigger
  # insecure mode for that zone (like without a trustanchor).  Default on,
  # which insists on dnssec data for trust-anchored zones.
    harden-dnssec-stripped: yes

  # Use 0x20-encoded random bits in the query to foil spoof attempts.
  # http://tools.ietf.org/html/draft-vixie-dnsext-dns0x20-00
  # While upper and lower case letters are allowed in domain names, no significance
  # is attached to the case. That is, two names with the same spelling but
  # different case are to be treated as if identical. This means calomel.org is the
  # same as CaLoMeL.Org which is the same as CALOMEL.ORG.
    use-caps-for-id: yes

  # the time to live (TTL) value lower bound, in seconds. Default 0.
  # If more than an hour could easily give trouble due to stale data.
    cache-min-ttl: 3600

  # the time to live (TTL) value cap for RRsets and messages in the
  # cache. Items are not cached for longer. In seconds.
    cache-max-ttl: 86400

  # perform prefetching of close to expired message cache entries.  If a client
  # requests the dns lookup and the TTL of the cached hostname is going to
  # expire in less than 10% of its TTL, unbound will (1st) return the ip of the
  # host to the client and (2nd) pre-fetch the dns request from the remote dns
  # server. This method has been shown to increase the amount of cached hits by
  # local clients by 10% on average.
    prefetch: yes

  # number of threads to create. 1 disables threading. This should equal the number
  # of CPU cores in the machine. Our example machine has 4 CPU cores.
    num-threads: 1

  ## Unbound Optimization and Speed Tweaks ###
```

```
58
59    # the number of slabs to use for cache and must be a power of 2 times the
60    # number of num-threads set above. more slabs reduce lock contention, but
61    # fragment memory usage.
62      msg-cache-slabs: 8
63      rrset-cache-slabs: 8
64      infra-cache-slabs: 8
65      key-cache-slabs: 8
66
67    # Increase the memory size of the cache. Use roughly twice as much rrset cache
68    # memory as you use msg cache memory. Due to malloc overhead, the total memory
69    # usage is likely to rise to double (or 2.5x) the total cache memory. The test
70    # box has 4gig of ram so 256meg for rrset allows a lot of room for cacheed objects.
71      rrset-cache-size: 256m
72      msg-cache-size: 128m
73
74    # buffer size for UDP port 53 incoming (SO_RCVBUF socket option). This sets
75    # the kernel buffer larger so that no messages are lost in spikes in the traffic.
76      so-rcvbuf: 1m
77
78    # Should additional section of secure message also be kept clean of unsecure
79    # data. Useful to shield the users of this validator from potential bogus
80    # data in the additional section. All unsigned data in the additional section
81    # is removed from secure messages.
82      val-clean-additional: yes
83
84    # If nonzero, unwanted replies are not only reported in statistics, but also
85    # a running total is kept per thread. If it reaches the threshold, a warning
86    # is printed and a defensive action is taken, the cache is cleared to flush
87    # potential poison out of it.  A suggested value is 10000000, the default is
88    # 0 (turned off). We think 10K is a good value.
89      unwanted-reply-threshold: 10000
90
91    # Reduce EDNS reassembly buffer size.
92    # Suggested by the unbound man page to reduce fragmentation reassembly problems
93      edns-buffer-size: 1472
```

configFiles/dns/unbound/listening.conf

```
1  server:
2     #set dns listening for ipv4
3     interface: 127.0.0.1
4
5     #set dns listening for ipv6
6     interface: ::1
7
8     # port to answer queries from
9      port: 53
10
11    # Enable IPv4, "yes" or "no".
12     do-ip4: yes
13
14    # Enable IPv6, "yes" or "no".
15     do-ip6: yes
16
17    # Enable UDP, "yes" or "no".
18     do-udp: yes
19
20    # Enable TCP, "yes" or "no".
21     do-tcp: yes
```

configFiles/dns/unbound/qname–minimisation.conf

```
1  server:
2      # Send minimum amount of information to upstream servers to enhance
3      # privacy. Only sends minimum required labels of the QNAME and sets
```

```
    # QTYPE to NS when possible.

    # See RFC 7816 "DNS Query Name Minimisation to Improve Privacy" for
    # details.

    qname-minimisation: yes
```

configFiles/dns/unbound/root–auto–trust–anchor–file.conf

```
server:
    # The following line will configure unbound to perform cryptographic
    # DNSSEC validation using the root trust anchor.
    auto-trust-anchor-file: "/var/lib/unbound/root.key"
```

configFiles/dns/nsd/nsd.conf

```

server:
  # uncomment to specify specific interfaces to bind (default all).
    ip-address: 104.248.137.212
    #ip-address:

  # port to answer queries on. default is 53.
    port: 53

  # Number of NSD servers to fork.
    server-count: 1

  # listen only on IPv4 connections
    ip4-only: yes

  # don't answer VERSION.BIND and VERSION.SERVER CHAOS class queries
    hide-version: yes

  # identify the server (CH TXT ID.SERVER entry).
    identity: ""

    logfile: "/var/log/nsd.log"

# The directory for zonefile: files.
    zonesdir: "/etc/nsd/zones"
    pidfile: "/etc/nsd/nsd.pid"
    username: nsd

pattern:
    name: examplerun.cf
    zonefile: examplerun.cf.forward
pattern:
    name: 212.137.248.104.in-addr.arpa
    zonefile: examplerun.cf.backward
```

configFiles/dns/nsd/examplerun.cf.backward

```

$ORIGIN 212.137.248.104.in-addr.arpa.
$TTL 1800

@ IN SOA ns1.examplerun.cf. ns2.examplerun.cf. (
            2019010917   ; serial number
            28800        ; Refresh
            7200         ; Retry
            1209600      ; Expire
            86400        ; Min TTL
            )

            NS      ns1.examplerun.cf.
```

```
            NS      ns2.examplerun.cf.
; PTR
                        IN      PTR     examplerun.cf.
                        IN      PTR     mail.examplerun.cf.
```

configFiles/dns/nsd/examplerun.cf.forward

```

$ORIGIN examplerun.cf.    ; default zone domain
$TTL    86400               ; default time to live


@ IN SOA ns1.examplerun.cf. ns2.examplerun.cf. (
            2019010917   ; serial number
            28800        ; Refresh
            7200         ; Retry
            1209600      ; Expire
            86400        ; Min TTL
            )

            NS      ns1.examplerun.cf.
            NS      ns2.examplerun.cf.
            MX      10 mail.examplerun.cf.

examplerun.cf.  IN CAA 0 issue "letsencrypt.org"
examplerun.cf. IN CAA 0 iodef "mailto:postmaster@examplerun.cf"

  IN A  104.248.137.212
  IN TXT  "v=spf1 mx a ~all"
ns1   IN A  104.248.137.212
ns2   IN A  104.248.137.212
www   IN A  104.248.137.212
*     IN A  104.248.137.212

mail IN A 104.248.137.212
     IN TXT  "v=spf1 mx a ~all"
2019010917._domainkey       IN      TXT     (
"v=DKIM1\059 h=sha256\059 k=rsa\059 s=email\059 p="
"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA6N+
    Xk5S5yT9WNMgbIS7CvNKdWFKpSR7Tfo6trV0Ml6O6BHsFiSp5U5"
"kbQ/vrK/xgx9c4k5BIOk/yL/jd/O/BqjTGDnC/pL89SL1Ne5Z+
    vW1h4FEw9gmwk3etscUP0CYZZs5PgvDlBPgfWyitrjy+pYlxsFBORXZPlrpQRFnNYpSR/
    eAXWF3REliO7NquSSec985dpbZWQ/3MHm"
"W8ZVwv5oDfh/kMQ9727qMxpOED0ZQyml2kPpdHK87Rg9zGOJDJs880RC3lsd+6
    tukf7fYyj51TvpRtndLPrbutKdFgi3eMMDkQXam+d8f3YHQoiMF7lR0pD2oOcH5glELX7gc6MwIDAQAB"
    )
_adsp._domainkey        IN TXT "dkim=all"
_dmarc IN TXT "v=DMARC1\059 p=quarantine\059 sp=quarantine\059 adkim=r\059 aspf=r\059
    fo=1\059 rf=afrf\059 rua=mailto:postmaster@examplerun.cf"
```

configFiles/ssh/sshd.config

```
#   $OpenBSD: sshd_config,v 1.101 2017/03/14 07:19:07 djm Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

#Port 22
#AddressFamily any
```

```
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile     .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the ChallengeResponseAuthentication and
# PasswordAuthentication.  Depending on your PAM configuration,
# PAM authentication via ChallengeResponseAuthentication may bypass
# the setting of "PermitRootLogin yes
```

```
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and ChallengeResponseAuthentication to 'no'.
UsePAM yes

#AllowAgentForwarding yes
#AllowTcpForwarding yes
#GatewayPorts no
X11Forwarding no
#X11DisplayOffset 10
#X11UseLocalhost yes
#PermitTTY yes
PrintMotd no
#PrintLastLog yes
#TCPKeepAlive yes
#UseLogin no
#PermitUserEnvironment no
#Compression delayed
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no
#PidFile /var/run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
#Banner none

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

# override default of no subsystems
Subsystem       sftp    /usr/lib/openssh/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#       X11Forwarding no
#       AllowTcpForwarding no
#       PermitTTY no
#       ForceCommand cvs server
HostKeyAlgorithms ssh-ed25519-cert-v01@openssh.com,ssh-rsa-cert-v01@openssh.com,ssh-ed25519,ssh-rsa,ecdsa-sha2-nistp521-cert-v01@openssh.com,ecdsa-sha2-nistp384-cert-v01@openssh.com,ecdsa-sha2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp521,ecdsa-sha2-nistp384,ecdsa-sha2-nistp256

KexAlgorithms curve25519-sha256@libssh.org,ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,diffie-hellman-group-exchange-sha256
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,umac-128@openssh.com
Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr
```

configFiles/mail/main.cf

```
# See /usr/share/postfix/main.cf.dist for a commented, more complete version


# Debian specific:  Specifying a file name will cause the first
# line of that file to be used as the name.  The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)
biff = no

```

```
# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

# See http://www.postfix.org/COMPATIBILITY_README.html -- default to 2 on
# fresh installs.
compatibility_level = 2

# TLS parameters
smtpd_tls_cert_file = /etc/letsencrypt/live/mail.examplerun.cf/fullchain.pem
smtpd_tls_key_file = /etc/letsencrypt/live/mail.examplerun.cf/privkey.pem
smtpd_use_tls = yes
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:/var/lib/postfix/smtp_tls_session_cache

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package for
# information on enabling SSL in the smtp client.

smtpd_relay_restrictions = permit_mynetworks, permit_sasl_authenticated,
    permit_tls_all_clientcerts, reject_unauth_destination
myhostname = mail.examplerun.cf
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = $myhostname, $mydomain, mail.examplerun.cf, localhost.examplerun.cf,
    localhost
relayhost =
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
inet_protocols = all
sender_canonical_maps = hash:/etc/postfix/canonical
mydomain = examplerun.cf
smtp_tls_security_level = may
smtp_tls_note_starttls_offer = yes
smtp_tls_loglevel = 1
###https://access.redhat.com/articles/1468593
smtpd_tls_mandatory_protocols = !SSLv2, !SSLv3, !TLSv1, !TLSv1.1
smtpd_tls_protocols = !SSLv2, !SSLv3, !TLSv1, !TLSv1.1
smtp_tls_mandatory_protocols = !SSLv2, !SSLv3, !TLSv1, !TLSv1.1
smtp_tls_protocols = !SSLv2, !SSLv3, !TLSv1, !TLSv1.1
smtp_tls_exclude_ciphers = EXP, MEDIUM, LOW, DES, 3DES, SSLv2
smtpd_tls_exclude_ciphers = EXP, MEDIUM, LOW, DES, 3DES, SSLv2
tls_high_cipherlist = kEECDH:+kEECDH+SHA:kEDH:+kEDH+SHA:+kEDH+CAMELLIA:kECDH:+kECDH+
    SHA:kRSA:+kRSA+SHA:+kRSA+CAMELLIA:!aNULL:!eNULL:!SSLv2:!RC4:!MD5:!DES:!EXP:!SEED:!
    IDEA:!3DES:!SHA
tls_preempt_cipherlist = yes
smtp_tls_ciphers = high
smtpd_tls_ciphers = high
policyd-spf_time_limit = 3600
smtpd_recipient_restrictions = reject_unauth_pipelining, reject_non_fqdn_recipient,
    reject_unknown_recipient_domain, permit_mynetworks, check_policy_service unix:
    private/policyd-spf, reject_rbl_client zen.spamhaus.org, reject_rbl_client bl.
    spamcop.net
#START OpendKIM & OpenDMARC
milter_protocol = 6
milter_default_action = accept
smtpd_milters = local:/opendkim/opendkim.sock, local:/opendmarc/opendmarc.sock
non_smtpd_milters = local:/opendkim/opendkim.sock, local:/opendmarc/opendmarc.sock
#END OpendKIM & OpenDMARC
smtpd_tls_CAfile = /etc/ssl/certs/examplerun.cf.ca.crl.pem
```

```
tls_append_default_CA = no
```

configFiles/mail/master.cf

```
#
# Postfix master process configuration file.  For details on the format
# of the file, see the master(5) manual page (command: "man 5 master" or
# on-line: http://www.postfix.org/master.5.html).
#
# Do not forget to execute "postfix reload" after editing this file.
#
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (no)    (never) (100)
# ==========================================================================
smtp      inet  n       -       y       -       -       smtpd
#smtp      inet  n       -       y       -       1       postscreen
#smtpd     pass  -       -       y       -       -       smtpd
#dnsblog   unix  -       -       y       -       0       dnsblog
#tlsproxy  unix  -       -       y       -       0       tlsproxy
#submission inet n       -       y       -       -       smtpd
#  -o syslog_name=postfix/submission
  -o smtpd_tls_security_level=encrypt
  -o smtpd_sasl_auth_enable=yes
  -o smtpd_tls_ask_ccert=yes
#  -o smtpd_tls_security_level=encrypt
#  -o smtpd_sasl_auth_enable=yes
#  -o smtpd_tls_auth_only=yes
#  -o smtpd_reject_unlisted_recipient=no
#  -o smtpd_client_restrictions=$mua_client_restrictions
#  -o smtpd_helo_restrictions=$mua_helo_restrictions
#  -o smtpd_sender_restrictions=$mua_sender_restrictions
#  -o smtpd_recipient_restrictions=
#  -o smtpd_relay_restrictions=permit_sasl_authenticated,reject
#  -o milter_macro_daemon_name=ORIGINATING
#smtps     inet  n       -       y       -       -       smtpd
#  -o syslog_name=postfix/smtps
#  -o smtpd_tls_wrappermode=yes
#  -o smtpd_sasl_auth_enable=yes
#  -o smtpd_reject_unlisted_recipient=no
#  -o smtpd_client_restrictions=$mua_client_restrictions
#  -o smtpd_helo_restrictions=$mua_helo_restrictions
#  -o smtpd_sender_restrictions=$mua_sender_restrictions
#  -o smtpd_recipient_restrictions=
#  -o smtpd_relay_restrictions=permit_sasl_authenticated,reject
#  -o milter_macro_daemon_name=ORIGINATING
#628       inet  n       -       y       -       -       qmqpd
pickup    unix  n       -       y       60      1       pickup
cleanup   unix  n       -       y       -       0       cleanup
qmgr      unix  n       -       n       300     1       qmgr
#qmgr     unix  n       -       n       300     1       oqmgr
tlsmgr    unix  -       -       y       1000?   1       tlsmgr
rewrite   unix  -       -       y       -       -       trivial-rewrite
bounce    unix  -       -       y       -       0       bounce
defer     unix  -       -       y       -       0       bounce
trace     unix  -       -       y       -       0       bounce
verify    unix  -       -       y       -       1       verify
flush     unix  n       -       y       1000?   0       flush
proxymap  unix  -       -       n       -       -       proxymap
proxywrite unix -       -       n       -       1       proxymap
smtp      unix  -       -       y       -       -       smtp
relay     unix  -       -       y       -       -       smtp
        -o syslog_name=postfix/$service_name
#       -o smtp_helo_timeout=5 -o smtp_connect_timeout=5
showq     unix  n       -       y       -       -       showq
error     unix  -       -       y       -       -       error
retry     unix  -       -       y       -       -       error
```

```
discard   unix  -       -       y       -       -       discard
local     unix  -       n       n       -       -       local
virtual   unix  -       n       n       -       -       virtual
lmtp      unix  -       -       y       -       -       lmtp
anvil     unix  -       -       y       -       1       anvil
scache    unix  -       -       y       -       1       scache
#
# ====================================================================
# Interfaces to non-Postfix software. Be sure to examine the manual
# pages of the non-Postfix software to find out what options it wants.
#
# Many of the following services use the Postfix pipe(8) delivery
# agent.  See the pipe(8) man page for information about ${recipient}
# and other message envelope options.
# ====================================================================
#
# maildrop. See the Postfix MAILDROP_README file for details.
# Also specify in main.cf: maildrop_destination_recipient_limit=1
#
maildrop  unix  -       n       n       -       -       pipe
  flags=DRhu user=vmail argv=/usr/bin/maildrop -d ${recipient}
#
# ====================================================================
#
# Recent Cyrus versions can use the existing "lmtp" master.cf entry.
#
# Specify in cyrus.conf:
#   lmtp    cmd="lmtpd -a" listen="localhost:lmtp" proto=tcp4
#
# Specify in main.cf one or more of the following:
#  mailbox_transport = lmtp:inet:localhost
#  virtual_transport = lmtp:inet:localhost
#
# ====================================================================
#
# Cyrus 2.1.5 (Amos Gouaux)
# Also specify in main.cf: cyrus_destination_recipient_limit=1
#
#cyrus     unix  -       n       n       -       -       pipe
#  user=cyrus argv=/cyrus/bin/deliver -e -r ${sender} -m ${extension} ${user}
#
# ====================================================================
# Old example of delivery via Cyrus.
#
#old-cyrus unix  -       n       n       -       -       pipe
#  flags=R user=cyrus argv=/cyrus/bin/deliver -e -m ${extension} ${user}
#
# ====================================================================
#
# See the Postfix UUCP_README file for configuration details.
#
uucp      unix  -       n       n       -       -       pipe
  flags=Fqhu user=uucp argv=uux -r -n -z -a$sender - $nexthop!rmail ($recipient)
#
# Other external delivery methods.
#
ifmail    unix  -       n       n       -       -       pipe
  flags=F user=ftn argv=/usr/lib/ifmail/ifmail -r $nexthop ($recipient)
bsmtp     unix  -       n       n       -       -       pipe
  flags=Fq. user=bsmtp argv=/usr/lib/bsmtp/bsmtp -t$nexthop -f$sender $recipient
scalemail-backend unix  -   n   n   -   2   pipe
  flags=R user=scalemail argv=/usr/lib/scalemail/bin/scalemail-store ${nexthop} ${user
     } ${extension}
mailman   unix  -       n       n       -       -       pipe
  flags=FR user=list argv=/usr/lib/mailman/bin/postfix-to-mailman.py
  ${nexthop} ${user}
```

```
129
130 policyd-spf  unix  -       n       n       -       0       spawn
131     user=policyd-spf argv=/usr/bin/policyd-spf
```

configFiles/mail/canonical.conf

```
1
2 test@mail.examplerun.cf test@examplerun.cf
3 root@mail.examplerun.cf postmaster@examplerun.cf
```

configFiles/mail/aliases.conf

```
1
2 test: test
3 postmaster: root
```

configFiles/mail/opendmarc.conf

```
1  # This is a basic configuration that can easily be adapted to suit a standard
2  # installation. For more advanced options, see opendkim.conf(5) and/or
3  # /usr/share/doc/opendmarc/examples/opendmarc.conf.sample.
4
5  ##  AuthservID (string)
6  ##      defaults to MTA name
7  ##
8  ##  Sets the "authserv-id" to use when generating the Authentication-Results:
9  ##  header field after verifying a message.  If the string "HOSTNAME" is
10 ##  provided, the name of the host running the filter (as returned by the
11 ##  gethostname(3) function) will be used.
12 #
13 # AuthservID name
14
15 ##  FailureReports { true | false }
16 ##      default "false"
17 ##
18 ##  Enables generation of failure reports when the DMARC test fails and the
19 ##  purported sender of the message has requested such reports.  Reports are
20 ##  formatted per RFC6591.
21 #
22 # FailureReports false
23
24 ##  PidFile path
25 ##      default (none)
26 ##
27 ##  Specifies the path to a file that should be created at process start
28 ##  containing the process ID.
29 #
30 PidFile /var/run/opendmarc/opendmarc.pid
31
32 ##  PublicSuffixList path
33 ##      default (none)
34 ##
35 ##  Specifies the path to a file that contains top-level domains (TLDs) that
36 ##  will be used to compute the Organizational Domain for a given domain name,
37 ##  as described in the DMARC specification.  If not provided, the filter will
38 ##  not be able to determine the Organizational Domain and only the presented
39 ##  domain will be evaluated.
40 #
41 PublicSuffixList /usr/share/publicsuffix
42
43 ##  RejectFailures { true | false }
44 ##      default "false"
45 ##
46 ##  If set, messages will be rejected if they fail the DMARC evaluation, or
47 ##  temp-failed if evaluation could not be completed.  By default, no message
48 ##  will be rejected or temp-failed regardless of the outcome of the DMARC
```

```
##  evaluation of the message.  Instead, an Authentication-Results header
##  field will be added.
#
# RejectFailures false

##  Socket socketspec
##      default (none)
##
##  Specifies the socket that should be established by the filter to receive
##  connections from sendmail(8) in order to provide service.  socketspec is
##  in one of two forms: local:path, which creates a UNIX domain socket at
##  the specified path, or inet:port[@host] or inet6:port[@host] which creates
##  a TCP socket on the specified port for the appropriate protocol family.
##  If the host is not given as either a hostname or an IP address, the
##  socket will be listening on all interfaces.  This option is mandatory
##  either in the configuration file or on the command line.  If an IP
##  address is used, it must be enclosed in square brackets.
#
Socket local:/var/run/opendmarc/opendmarc.sock

##  Syslog { true | false }
##      default "false"
##
##  Log via calls to syslog(3) any interesting activity.
#
Syslog true

##  SyslogFacility facility-name
##      default "mail"
##
##  Log via calls to syslog(3) using the named facility.  The facility names
##  are the same as the ones allowed in syslog.conf(5).
#
# SyslogFacility mail

##  TrustedAuthservIDs string
##      default HOSTNAME
##
##  Specifies one or more "authserv-id" values to trust as relaying true
##  upstream DKIM and SPF results.  The default is to use the name of
##  the MTA processing the message.  To specify a list, separate each entry
##  with a comma.  The key word "HOSTNAME" will be replaced by the name of
##  the host running the filter as reported by the gethostname(3) function.
#
# TrustedAuthservIDs HOSTNAME

##  UMask mask
##      default (none)
##
##  Requests a specific permissions mask to be used for file creation.  This
##  only really applies to creation of the socket when Socket specifies a
##  UNIX domain socket, and to the HistoryFile and PidFile (if any); temporary
##  files are normally created by the mkstemp(3) function that enforces a
##  specific file mode on creation regardless of the process umask.  See
##  umask(2) for more information.
#
UMask 0002

##  UserID user[:group]
##      default (none)
##
##  Attempts to become the specified userid before starting operations.
##  The process will be assigned all of the groups and primary group ID of
##  the named userid unless an alternate group is specified.
#
UserID opendmarc
```

```
AutoRestart Yes
AutoRestartRate 10/1h
PidFile /var/spool/postfix/opendmarc/opendmarc.pid
Socket local:/var/spool/postfix/opendmarc/opendmarc.sock
AuthservID mail.examplerun.cf
TrustedAuthservIDs mail.examplerun.cf
Syslog true
SyslogFacility mail
UMask 0002
UserID opendmarc:opendmarc
```

configFiles/mail/opendkim.conf

```
# This is a basic configuration that can easily be adapted to suit a standard
# installation. For more advanced options, see opendkim.conf(5) and/or
# /usr/share/doc/opendkim/examples/opendkim.conf.sample.

# Log to syslog
Syslog          yes
# Required to use local socket with MTAs that access the socket as a non-
# privileged user (e.g. Postfix)
UMask           007

# Sign for example.com with key in /etc/dkimkeys/dkim.key using
# selector '2007' (e.g. 2007._domainkey.example.com)
#Domain         example.com
#KeyFile        /etc/dkimkeys/dkim.key
#Selector       2007

# Commonly-used options; the commented-out versions show the defaults.
#Canonicalization   simple
#Mode           sv
#SubDomains     no

# Socket smtp://localhost
#
# ##  Socket socketspec
# ##
# ##  Names the socket where this filter should listen for milter connections
# ##  from the MTA.  Required.  Should be in one of these forms:
# ##
# ##  inet:port@address           to listen on a specific interface
# ##  inet:port                   to listen on all interfaces
# ##  local:/path/to/socket       to listen on a UNIX domain socket
#
#Socket                 inet:8892@localhost
Socket          local:/var/run/opendkim/opendkim.sock

##  PidFile filename
###      default (none)
###
###  Name of the file where the filter should write its pid before beginning
###  normal operations.
#
PidFile             /var/run/opendkim/opendkim.pid


# Always oversign From (sign using actual From and a null From to prevent
# malicious signatures header fields (From and/or others) between the signer
# and the verifier.  From is oversigned by default in the Debian pacakge
# because it is often the identity key used by reputation systems and thus
# somewhat security sensitive.
OversignHeaders     From

##  ResolverConfiguration filename
##      default (none)
##
```

```
55 ##  Specifies a configuration file to be passed to the Unbound library that
56 ##  performs DNS queries applying the DNSSEC protocol.  See the Unbound
57 ##  documentation at http://unbound.net for the expected content of this file.
58 ##  The results of using this and the TrustAnchorFile setting at the same
59 ##  time are undefined.
60 ##  In Debian, /etc/unbound/unbound.conf is shipped as part of the Suggested
61 ##  unbound package
62
63 # ResolverConfiguration     /etc/unbound/unbound.conf
64
65 ##  TrustAnchorFile filename
66 ##      default (none)
67 ##
68 ## Specifies a file from which trust anchor data should be read when doing
69 ## DNS queries and applying the DNSSEC protocol.  See the Unbound documentation
70 ## at http://unbound.net for the expected format of this file.
71
72 TrustAnchorFile       /usr/share/dns/root.key
73
74 ##  Userid userid
75 ###      default (none)
76 ###
77 ###  Change to user "userid" before starting normal operation?  May include
78 ###  a group ID as well, separated from the userid by a colon.
79 #
80 UserID                opendkim
81 # This is a basic configuration that can easily be adapted to suit a standard
82 # installation. For more advanced options, see opendkim.conf(5) and/or
83 # /usr/share/doc/opendkim/examples/opendkim.conf.sample.
84
85 # Log to syslog
86 Syslog          yes
87 # Required to use local socket with MTAs that access the socket as a non-
88 # privileged user (e.g. Postfix)
89 UMask           002
90 # OpenDKIM user
91 # Remember to add user postfix to group opendkim
92 UserID          opendkim
93
94 # Map domains in From addresses to keys used to sign messages
95 KeyTable        /etc/opendkim/key.table
96 SigningTable        refile:/etc/opendkim/signing.table
97
98 # Hosts to ignore when verifying signatures
99 ExternalIgnoreList  /etc/opendkim/trusted.hosts
100 InternalHosts       /etc/opendkim/trusted.hosts
101
102 # Commonly-used options; the commented-out versions show the defaults.
103 Canonicalization    relaxed/simple
104 Mode            sv
105 SubDomains      no
106 #ADSPAction     continue
107 AutoRestart     no
108 AutoRestartRate     10/1M
109 Background      yes
110 DNSTimeout      5
111 SignatureAlgorithm  rsa-sha256
112
113 # Always oversign From (sign using actual From and a null From to prevent
114 # malicious signatures header fields (From and/or others) between the signer
115 # and the verifier.  From is oversigned by default in the Debian package
116 # because it is often the identity key used by reputation systems and thus
117 # somewhat security sensitive.
118 OversignHeaders     From
119 ###UBUNTU 18.10
120 PidFile               /var/spool/postfix/opendkim/opendkim.pid
```

```
121 Socket          local:/var/spool/postfix/opendkim/opendkim.sock
```

configFiles/mail/signing.table

```
1 *@examplerun.cf examplerun
```

configFiles/mail/trusted.hosts

```
1 127.0.0.1
2 ::1
3 localhost
4 examplerun.cf
5 mail.examplerun.cf
```

configFiles/mail/users–external.conf

```
1 test:::::::
```

configFiles/mail/dovecot.conf

```
1  ## Dovecot configuration file
2  !include_try /usr/share/dovecot/protocols.d/*.protocol
3
4  !include conf.d/*.conf
5
6  auth default {
7      mechanisms = plain login external
8      user = root
9      socket listen {
10       client {
11         path = /var/spool/postfix/private/auth
12         mode = 0660
13         user = postfix
14         group = postfix
15       }
16     }
17 }
```

configFiles/mail/10–auth.conf

```
1
2  ##
3  ## Authentication processes
4  ##
5
6  #disable_plaintext_auth = yes
7
8  auth_ssl_username_from_cert = yes
9
10 auth_mechanisms = plain login external
11
12 !include auth-system.conf.ext
13 !include auth-passwdfile.conf.ext
```

configFiles/mail/10–ssl.conf

```
1
2  ##
3  ## SSL settings
4  ##
5
6  ssl = yes
7
8  ssl_cert = </etc/letsencrypt/live/mail.examplerun.cf/fullchain.pem
9  ssl_key = </etc/letsencrypt/live/mail.examplerun.cf/privkey.pem
```

```
10
11 ssl_ca = </etc/ssl/certs/examplerun.cf.ca.crl.pem
12
13 ssl_client_ca_dir = /etc/ssl/certs
14
15 ssl_verify_client_cert = yes
16
17 ssl_cert_username_field = CN
18
19 # DH parameters length to use.
20 ssl_dh_parameters_length = 1024
21
22 # SSL protocols to use
23 ssl_protocols = !SSLv2 !SSLv3
24
25 # SSL ciphers to use
26 ssl_cipher_list = kEECDH:+kEECDH+SHA:kEDH:+kEDH+SHA:+kEDH+CAMELLIA:kECDH:+kECDH+SHA:
    kRSA:+kRSA+SHA:+kRSA+CAMELLIA:!aNULL:!eNULL:!SSLv2:!RC4:!MD5:!DES:!EXP:!SEED:!IDEA
    :!3DES
27
28 # Prefer the server's order of ciphers over client's.
29 ssl_prefer_server_ciphers = yes
30
31 # SSL extra options. Currently supported options are:
32 #   no_compression - Disable compression.
33 #   no_ticket - Disable SSL session tickets.
34 #ssl_options =
```

configFiles/mail/auth–passwdfile.conf.ext

```
1
2 # Authentication for passwd-file users. Included from 10-auth.conf.
3 #
4 # passwd-like file with specified location.
5 # <doc/wiki/AuthDatabase.PasswdFile.txt>
6
7 passdb {
8   driver = passwd-file
9   # the PLAIN scheme prevents us from having to hash the empty string
10  args = scheme=PLAIN username_format=%u /etc/dovecot/users-external
11
12  # this option requires Dovecot 2.2.28 (or the patch), without it this setup
13  # is insecure because it permits logins with the empty string as password
14  mechanisms = external
15
16  # explicitly permit empty passwords
17  override_fields = nopassword
18 }
19
20 userdb {
21  driver = passwd-file
22  args = username_format=%u /etc/dovecot/users-external
23 }
```

configFiles/fw/fw.conf

```
1 # SSH
2 allow - tcp - 22
3 allow - udp - 22
4 # DNS
5 allow - tcp - 53
6 allow - udp - 53
7 # MAIL
8 allow - tcp - 25
9 allow - udp - 25
10 # SECURE SMTP
```

```
allow - tcp - 465
allow - udp - 465
# IMAP
allow - tcp - 143
allow - udp - 143
# IMAP TLS
allow - tcp - 993
allow - udp - 993
# HTTP HTTPS
allow - tcp - 80
allow - tcp - 443
```

configFiles/web/nginx/nginx.conf

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
    # multi_accept on;
}

http {
    ##
    # Basic Settings
    ##

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
    server_tokens off;

    # server_names_hash_bucket_size 64;
    # server_name_in_redirect off;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    ##
    # Logging Settings
    ##

    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    ##
    # Gzip Settings
    ##

    gzip on;

    # gzip_vary on;
    # gzip_proxied any;
    # gzip_comp_level 6;
    # gzip_buffers 16 8k;
    # gzip_http_version 1.1;
    # gzip_types text/plain text/css application/json application/javascript text/xml
      application/xml application/xml+rss text/javascript;

    ##
    # Virtual Host Configs
    ##

```

```
    include /etc/nginx/conf.d/*.conf;
}
```

configFiles/web/nginx/conf.d/examplerun.cf.conf

```
server {
        listen 443 ssl;
        listen [::]:443 ssl;
        server_name examplerun.cf www.examplerun.cf default_server;

        ssl_prefer_server_ciphers on;
        ssl_protocols TLSv1.1 TLSv1.2;
        ssl_ciphers ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
            ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM
            -SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-
            AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256;

        ssl_session_cache shared:SSL:50m;
        ssl_session_timeout 5m;

        ssl_certificate /etc/letsencrypt/live/examplerun.cf/fullchain.pem; # managed
            by Certbot
        ssl_certificate_key /etc/letsencrypt/live/examplerun.cf/privkey.pem; # managed
             by Certbot

        ssl_dhparam /etc/ssl/dh4096.pem;

        add_header Strict-Transport-Security "max-age=31536000; includeSubDomains"
            always;
        add_header X-Content-Type-Options "nosniff" always;
        add_header X-Xss-Protection "1; mode=block" always;
        add_header X-Frame-Options "SAMEORIGIN" always;
        add_header Referrer-Policy "same-origin" always;

        access_log /var/log/nginx/examplerun.cf_ssl_access.log;
        error_log /var/log/nginx/examplerun.cf_ssl_error.log;

        location / {
                proxy_set_header X-Real-IP       $remote_addr;
                proxy_set_header X-Forwarded-For $remote_addr;
                proxy_set_header Host            $host;
                proxy_pass http://127.0.0.1:8080;
        }
}

server {
        listen 80;
        listen [::]:80;
        server_name examplerun.cf www.examplerun.cf default_server;

        access_log /var/log/nginx/examplerun.cf_access.log;
        error_log /var/log/nginx/examplerun.cf_error.log;

        return 301 https://$host$request_uri;
}
```

configFiles/web/apache2/ports.conf

```
Listen 8080
# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

configFiles/web/apache2/sites–available/examplerun.cf.conf

```

<VirtualHost 127.0.0.1:8080>
    ServerName examplerun.cf
```

```
    ServerName www.examplerun.cf
    ServerAdmin webmaster@examplerun.cf
    DocumentRoot /var/www/examplerun

    #LogLevel info ssl:warn

    ErrorLog  ${APACHE_LOG_DIR}/ismu.ga_error.log
    CustomLog ${APACHE_LOG_DIR}/ismu.ga_access.log combined
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```