

BERN UNIVERSITY OF APPLIED SCIENCES

BTI7301 - PROJECT 1

Mail Server Set-Up & Security-Hardening Script

User Manual

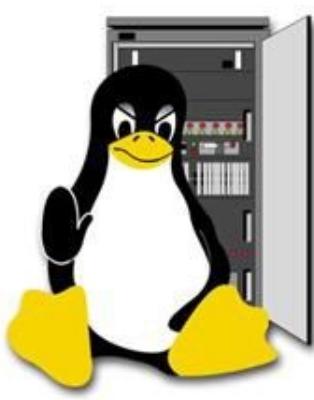
Authors:

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

Abstract

List of Figures	5
1 Introduction	6
1.1 Prerequisites	7
1.1.1 Ubuntu 18.04 Server	7
1.1.2 Domain	7
1.1.3 Minimal Linux knowledge	7
1.2 Architecture overview	8
2 Walkthrough	9
2.1 Code directory tree	9
2.2 Overview	10
2.2.1 Complete run	10
2.2.2 Rerun run	11
2.2.3 Overview process diagram	12
2.3 Firewall	13
2.3.1 Firewall process diagram	15
2.4 DNS	16
2.4.1 DNS architecture diagram	18
2.4.2 DNS process diagram	19
2.4.3 Multiple domains	19
2.5 User management	20
2.5.1 Actions	20
2.5.2 User management process diagram	22
2.6 SSH	23
2.6.1 Configuration	23
2.6.2 SSH process diagram	25
2.7 E-Mail	26
2.7.1 Configurations	26
2.7.2 E-Mail process diagram	31
2.7.3 Multiple e-mail addresses	31
2.8 Web	32
2.8.1 Web architecture diagram	33
3 Hardening Tests	34
3.1 Firewall	34
3.2 DNS	37
3.2.1 Domain name resolver	37
3.2.2 Authoritative DNS	39
3.3 SSH	40
3.3.1 SSH daemon	40
3.4 E-Mail	42
3.4.1 E-Mail server configuration	42
3.4.2 E-Mail header	44
3.5 Web	45



4 E-Mail Client configuration	47
4.1 Mail on macOS Mojave	47
4.1.1 Mail server config	47
4.1.2 Mail SMTP settings	48
4.1.3 Mail IMAP TLS setting	49
5 Future Work	50
5.1 Extended functionalities	50
5.1.1 Multiple domains	50
5.1.2 Multiple e-mail addresses	50
5.1.3 Web application server	50
5.2 More Hardening	50
5.3 Containerization	51
5.4 Code Migration	51
6 Conclusion	52
7 License	53
7.1 MIT license	53
8 Slides	54
9 Glossary	64
Bibliography	67
Appendices	68



List of Figures

1.1	Architecture overview	8
2.1	Setup process diagram	12
2.2	Firewall process diagram	15
2.3	Architecture DNS	18
2.4	DNS process diagram	19
2.5	User management process diagram	22
2.6	SSH process diagram	25
2.7	Email process diagram	31
2.8	Architecture Web	33
3.1	Firewall (without DNS) BEFORE	34
3.2	Firewall (with DNS) BEFORE	35
3.3	Firewall setup AFTER	36
3.4	Name resolver BEFORE	37
3.5	Name resolver details BEFORE	37
3.6	Name resolver AFTER	38
3.7	Name resolver details AFTER	38
3.8	Authoritative DNS test BEFORE	39
3.9	Authoritative DNS test AFTER	39
3.10	SSH daemon BEFORE	40
3.11	SSH daemon AFTER	41
3.12	Mail BEFORE (emailsecuritygrader.com)	42
3.13	Mail BEFORE (hardenize.com)	42
3.14	Mail AFTER (emailsecuritygrader.com)	43
3.15	Mail AFTER (hardenize.com)	43
3.16	Mail header BEFORE	44
3.17	Mail header AFTER	44
3.18	Web BEFORE	45
3.19	Web AFTER	46
4.1	Mail server config	47
4.2	Mail SMTP settings	48
4.3	Mail IMAP TLS setting	49



1 Introduction

In this document, we described 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 nlmetlabs: 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 Prerequisites

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 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.

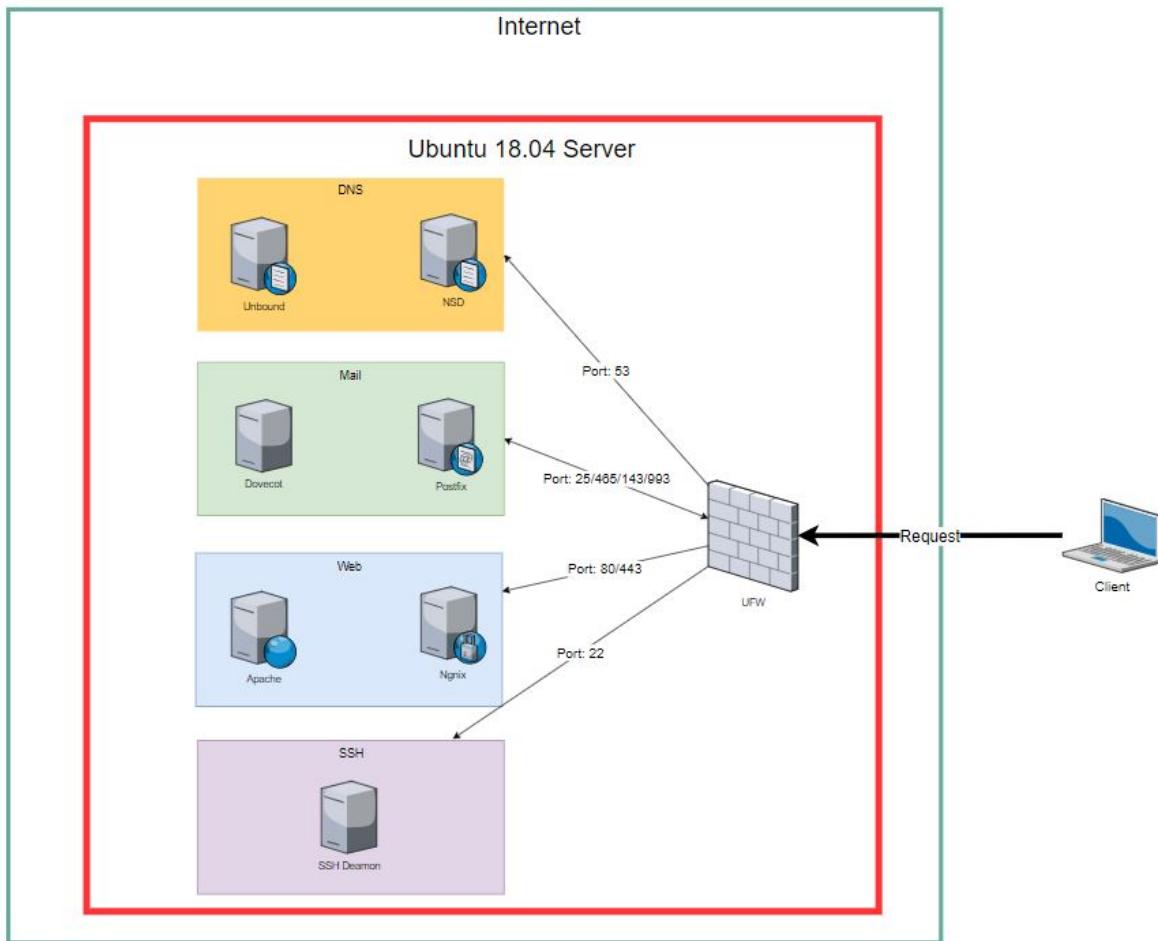


Figure 1.1: Architecture overview

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
|
└── files
    └── fw.conf -> fw/fw.conf
|
└── fw
    ├── controlTraffic.sh
    ├── enableUfw.sh
    ├── fw.conf
    ├── fw.sh
    ├── specificConfigurations.sh
    └── uninstall_fw.sh
|
└── mail
    ├── alias.sh
    ├── checkDomain.sh
    ├── clientCertificate.sh
    ├── dkim.sh
    ├── dmrc.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 completely 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 make sure spam mail is recognized during receiving and all sent mails reach their destination without being classified as spam from the receiving 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.

```
1 <INFO> - Tue Jan  8 11:14:31 UTC 2019 - No Modification Flag found. Seems to be the  
2   first run. Will start hardening now.  
3 *** QUESTION *** Do you wish to perform a complete run (Firewall, DNS, SSH, Mail, Web)  
4     [y/n]? y  
5 <INFO> - Tue Jan  8 11:14:39 UTC 2019 - Complete run set to true  
6  
7 [...]
```

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).

```
1 [...]  
2  
3  
4 <INFO> - Tue Jan  8 11:25:19 UTC 2019 - Set modification Flag.  
5 <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. Now 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.

```
1 *** QUESTION *** Modification Flag found. Please choose option: modify/uninstall [m/u]? m
2 <INFO> - Wed Jan 9 08:45:33 UTC 2019 - Modification choosen
3 *** QUESTION *** Do you wish to perform a complete run (Firewall, DNS, SSH, Mail, Web) [y/n]? n
4 <INFO> - Wed Jan 9 08:45:34 UTC 2019 - Complete run set to false.
5 <INFO> - Wed Jan 9 08:45:34 UTC 2019 - Start the specific selection for single parts.
6 *** QUESTION *** Do you wish to perform action on fw [y/n]? y
7 <INFO> - Wed Jan 9 08:45:36 UTC 2019 - Action for fw set to true
8 *** QUESTION *** Do you wish to perform action on dns [y/n]? n
9 <INFO> - Wed Jan 9 08:45:37 UTC 2019 - Action for dns set to false (will skip it).
10 *** QUESTION *** Do you wish to perform action on ssh [y/n]? n
11 <INFO> - Wed Jan 9 08:45:37 UTC 2019 - Action for ssh set to false (will skip it).
12 *** QUESTION *** Do you wish to perform action on mail [y/n]? n
13 <INFO> - Wed Jan 9 08:45:40 UTC 2019 - Action for mail set to false (will skip it).
14 *** QUESTION *** Do you wish to perform action on web [y/n]? n
15 <INFO> - Wed Jan 9 08:45:43 UTC 2019 - Action for web set to false (will skip it).
16
17 [...]
18
19
20 <INFO> - Wed Jan 9 08:45:55 UTC 2019 - Set modification Flag.
21 <INFO> - Wed Jan 9 08:45:55 UTC 2019 - Done. Finished with configurations
```

Explanation [...]

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

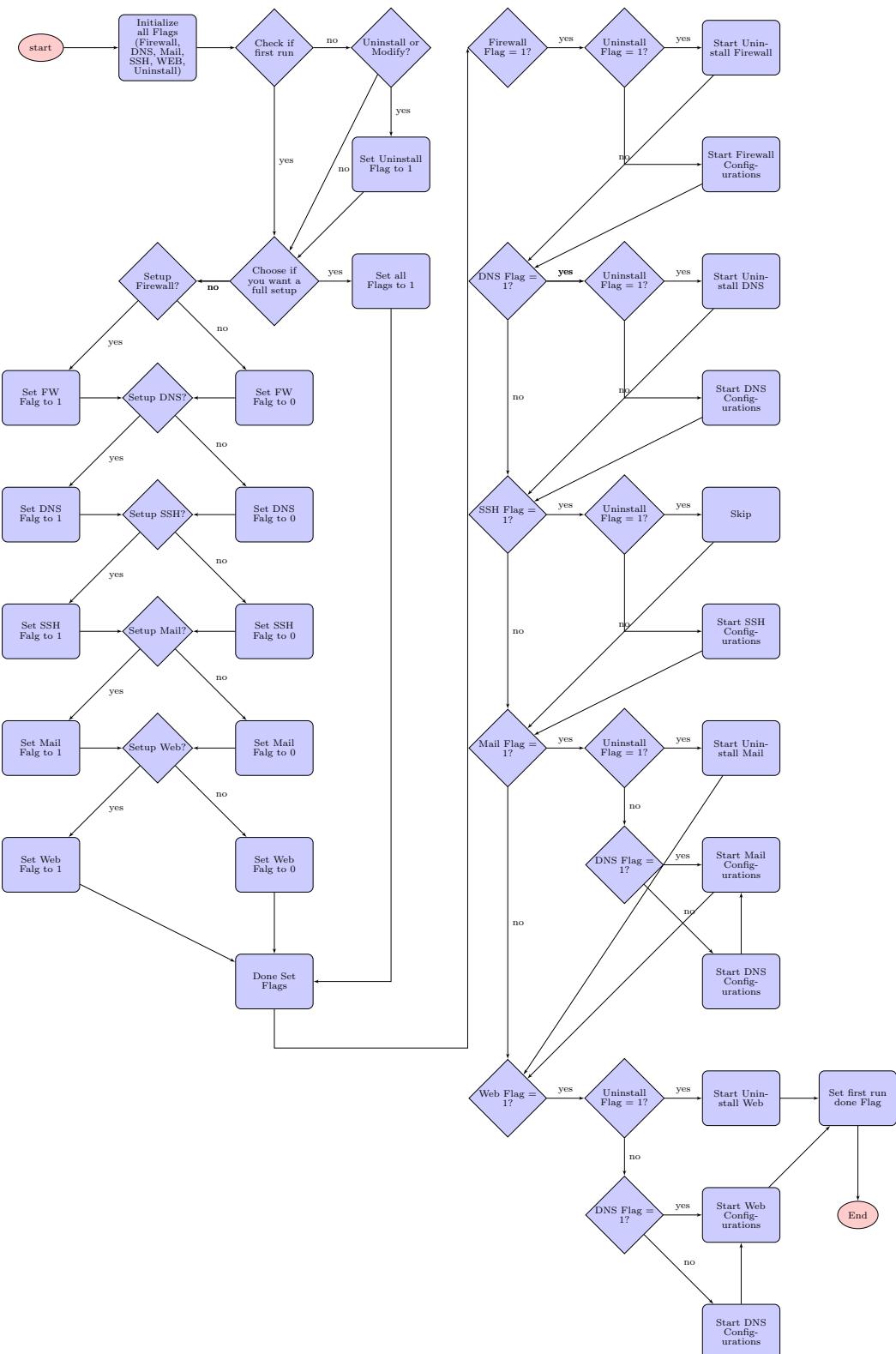


Figure 2.1: Setup process diagram

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.

```

1 <INFO> - Tue Jan  8 11:14:39 UTC 2019 - Starting Firewall Configurations.
2 <INFO> - Tue Jan  8 11:14:39 UTC 2019 - Will install 'ufw' now. Please wait...
3 .....
4 <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.

```

1 <INFO> - Tue Jan  8 11:14:53 UTC 2019 - Ufw is enabled now.
2 <FW> - Tue Jan  8 11:14:53 UTC 2019 - UFW enable done.
3 <INFO> - Tue Jan  8 11:14:53 UTC 2019 - Start Firewall Hardening. (close all non
      relevant ports)
4 <INFO> - Tue Jan  8 11:14:54 UTC 2019 - All incoming and outgoing traffic is handled
      now.
5 <FW> - Tue Jan  8 11:14:54 UTC 2019 - Traffic controll done.
6 <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.

```

1 <INFO> - Tue Jan  8 11:14:55 UTC 2019 - Looking for Firewall Config file for specific
      configurations
2 <INFO> - Tue Jan  8 11:14:55 UTC 2019 - File Found. /root/files/fw.conf
3 <INFO> - Tue Jan  8 11:15:19 UTC 2019 - Working on 'allow 22/tcp'.
4 <INFO> - Tue Jan  8 11:15:19 UTC 2019 - Working on 'allow 22/udp'.
5 <INFO> - Tue Jan  8 11:15:20 UTC 2019 - Working on 'allow 53/tcp'.
6 <INFO> - Tue Jan  8 11:15:20 UTC 2019 - Working on 'allow 53/udp'.
7 <INFO> - Tue Jan  8 11:15:20 UTC 2019 - Working on 'allow 25/tcp'.
8 <INFO> - Tue Jan  8 11:15:20 UTC 2019 - Working on 'allow 25/udp'.
9 <INFO> - Tue Jan  8 11:15:21 UTC 2019 - Working on 'allow 465/tcp'.
10 <INFO> - Tue Jan  8 11:15:21 UTC 2019 - Working on 'allow 465/udp'.
11 <INFO> - Tue Jan  8 11:15:21 UTC 2019 - Working on 'allow 143/tcp'.
12 <INFO> - Tue Jan  8 11:15:21 UTC 2019 - Working on 'allow 143/udp'.
13 <INFO> - Tue Jan  8 11:15:21 UTC 2019 - Working on 'allow 993/tcp'.
14 <INFO> - Tue Jan  8 11:15:22 UTC 2019 - Working on 'allow 993/udp'.
15 <INFO> - Tue Jan  8 11:15:22 UTC 2019 - Working on 'allow 80/tcp'.
16 <INFO> - Tue Jan  8 11:15:22 UTC 2019 - Working on 'allow 443/tcp'.
17 <INFO> - Tue Jan  8 11:15:22 UTC 2019 - Done Specific configurations.
18 <FW> - Tue Jan  8 11:15:22 UTC 2019 - Specific Configurations of UFW done.
19 <INFO> - Tue Jan  8 11:15:22 UTC 2019 - Firewall Configurations done.
20 Status: active
21
22 To                  Action    From
23 ---                ----
24 22/tcp              ALLOW     Anywhere
25 22/udp              ALLOW     Anywhere
26 53/tcp              ALLOW     Anywhere
27 53/udp              ALLOW     Anywhere
28 25/tcp              ALLOW     Anywhere
29 25/udp              ALLOW     Anywhere

```



```
30 465/tcp          ALLOW      Anywhere
31 465/udp          ALLOW      Anywhere
32 143/tcp          ALLOW      Anywhere
33 143/udp          ALLOW      Anywhere
34 993/tcp          ALLOW      Anywhere
35 993/udp          ALLOW      Anywhere
36 80/tcp           ALLOW      Anywhere
37 443/tcp          ALLOW      Anywhere
38 22/tcp (v6)       ALLOW      Anywhere (v6)
39 22/udp (v6)       ALLOW      Anywhere (v6)
40 53/tcp (v6)       ALLOW      Anywhere (v6)
41 53/udp (v6)       ALLOW      Anywhere (v6)
42 25/tcp (v6)       ALLOW      Anywhere (v6)
43 25/udp (v6)       ALLOW      Anywhere (v6)
44 465/tcp (v6)      ALLOW      Anywhere (v6)
45 465/udp (v6)      ALLOW      Anywhere (v6)
46 143/tcp (v6)      ALLOW      Anywhere (v6)
47 143/udp (v6)      ALLOW      Anywhere (v6)
48 993/tcp (v6)      ALLOW      Anywhere (v6)
49 993/udp (v6)      ALLOW      Anywhere (v6)
50 80/tcp (v6)       ALLOW      Anywhere (v6)
51 443/tcp (v6)      ALLOW      Anywhere (v6)
52
53 <FW> - Tue Jan  8 11:15:22 UTC 2019 - UFW Configurations done.
54 <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.

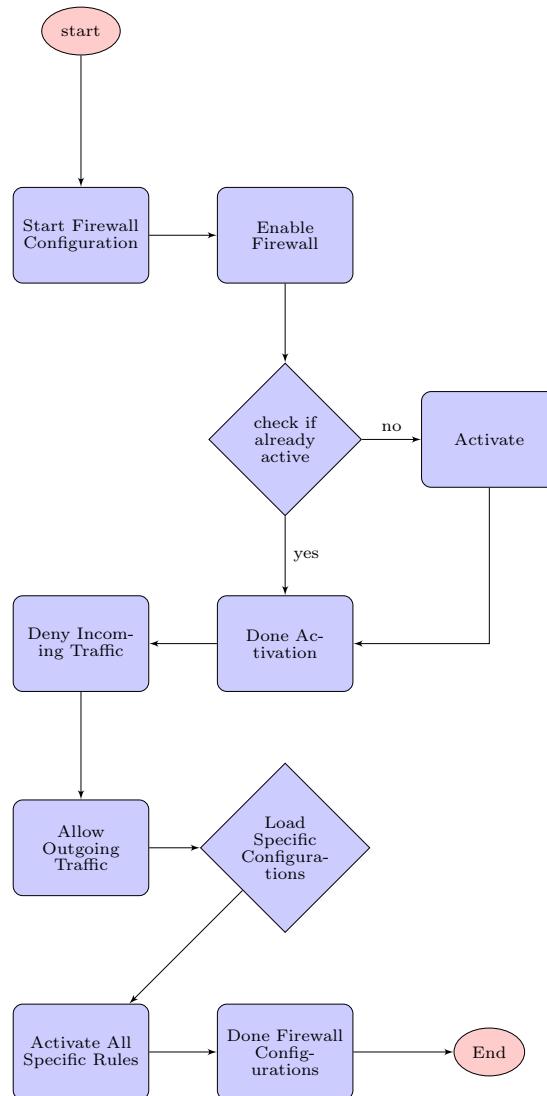


Figure 2.2: Firewall process diagram

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.

```

1 <INFO> - Tue Jan  8 11:15:22 UTC 2019 - Starting DNS Configurations.
2 *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
3 *PART 0: We install the basic configuration for unbound - we come back to it later
4 <INFO> - Tue Jan  8 11:15:22 UTC 2019 - Install DNS
5 <INFO> - Tue Jan  8 11:15:24 UTC 2019 - Will install 'unbound' now. Please wait...
6 .....
7 <INFO> - Tue Jan  8 11:15:36 UTC 2019 - Package 'unbound' is installed now.
8 <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)
9 <INFO> - Tue Jan  8 11:15:37 UTC 2019 - Configure DNS Ports, IPs
10 <INFO> - Tue Jan  8 11:15:37 UTC 2019 - Server will listen with localhost on port 53
11 <INFO> - Tue Jan  8 11:15:37 UTC 2019 - Configure DNS Access
12 <INFO> - Tue Jan  8 11:15:37 UTC 2019 - Configure this Client
13 <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).

```

1 *PART 1: We start with the authoritative Name Server: NSD
2
3 !!CAUTION!! you need your own domain - IF NOT the server wont be functional
4 DO NOT use a domain which does not belong to you, it may be illegal
5 *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..)
6
7 Press enter to continue
8
9 *** QUESTION *** do you have your own domain? (y/n/abort) y
10 <INFO> - Tue Jan  8 11:15:59 UTC 2019 -
11
12 *** QUESTION *** please enter your domain: exemplерун.cf
13
14 *** QUESTION *** is exemplерун.cf correct? (y/n/abort) y
15
16 <INFO> - Tue Jan  8 11:16:15 UTC 2019 - We will configure the authoritative DNS
   Server with the domain: exemplерун.cf

```

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

```

1 *** QUESTION *** is this 104.248.137.212 your external IP address ? (y (default)/n/
   abort) y
2 <INFO> - Tue Jan  8 11:16:48 UTC 2019 - We will configure the authoritative DNS Server
   with this: 104.248.137.212
3 <INFO> - Tue Jan  8 11:16:48 UTC 2019 - Install authoritative DNS for : exemplерун.cf
4 <INFO> - Tue Jan  8 11:16:48 UTC 2019 - Will install 'nsd' now. Please wait...
5 .....
6 <INFO> - Tue Jan  8 11:16:57 UTC 2019 - Package 'nsd' is installed now.
7 <INFO> - Tue Jan  8 11:16:59 UTC 2019 - Will install 'ldnsutils' now. Please wait...
8 .....
9 <INFO> - Tue Jan  8 11:17:06 UTC 2019 - Package 'ldnsutils' is installed now.
10 <INFO> - Tue Jan  8 11:17:06 UTC 2019 - Configure NSD
11 <INFO> - Tue Jan  8 11:17:06 UTC 2019 - Configure Forward Zone
12 <INFO> - Tue Jan  8 11:17:06 UTC 2019 - Configure Backward Zone
13 <INFO> - Tue Jan  8 11:17:06 UTC 2019 - Final steps
14 <INFO> - Tue Jan  8 11:17:11 UTC 2019 - Test NSD

```



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

```

1 PART 2: You have a full functional authoritative Name Server BUT your domain hoster
   does not know it!
2 !! VERY IMPORTANT !! GO to your domain hoster, change the name server for your domain
   to :
3         ns1.examplerun.cf with IP: 104.248.137.212
4         ns2.examplerun.cf with IP: 104.248.137.212
5 !! VERY IMPORTANT !! DO the same for the Glue Records, with the same name server and
   IPs
6 NOTE: It may take some time to change it - if you have difficulties with this part use
   your favorite web search engine
7
8 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.

```

1 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>
2 <INFO> - Tue Jan  8 11:17:40 UTC 2019 - Test local DNS
3 <INFO> - Tue Jan  8 11:17:40 UTC 2019 - Test ipv4 address
4 www.google.com.      3600    IN     A      216.58.210.4
5 <INFO> - Tue Jan  8 11:17:40 UTC 2019 - Test ipv6 address
6 ipv6.google.com.    604800   IN     CNAME   ipv6.l.google.com.
7 ipv6.l.google.com.   3600    IN     AAAA   2a00:1450:4005:800::200e
8
9 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 the how those two servers are separated.

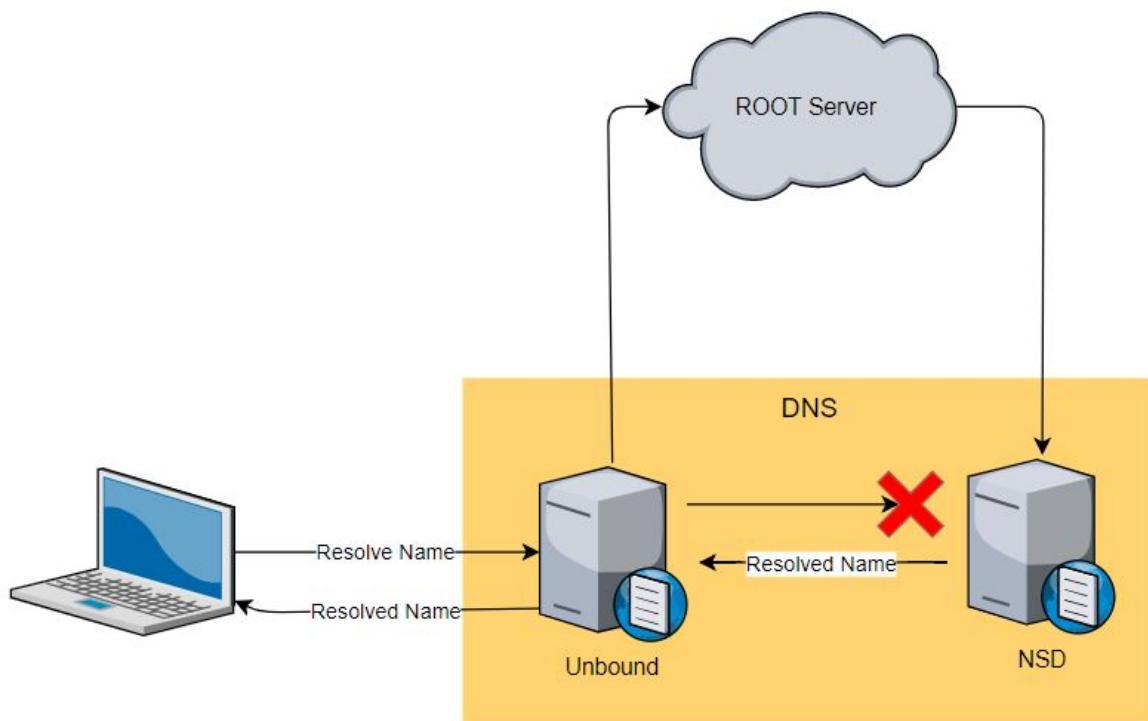


Figure 2.3: Architecture DNS

2.4.2 DNS process diagram

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

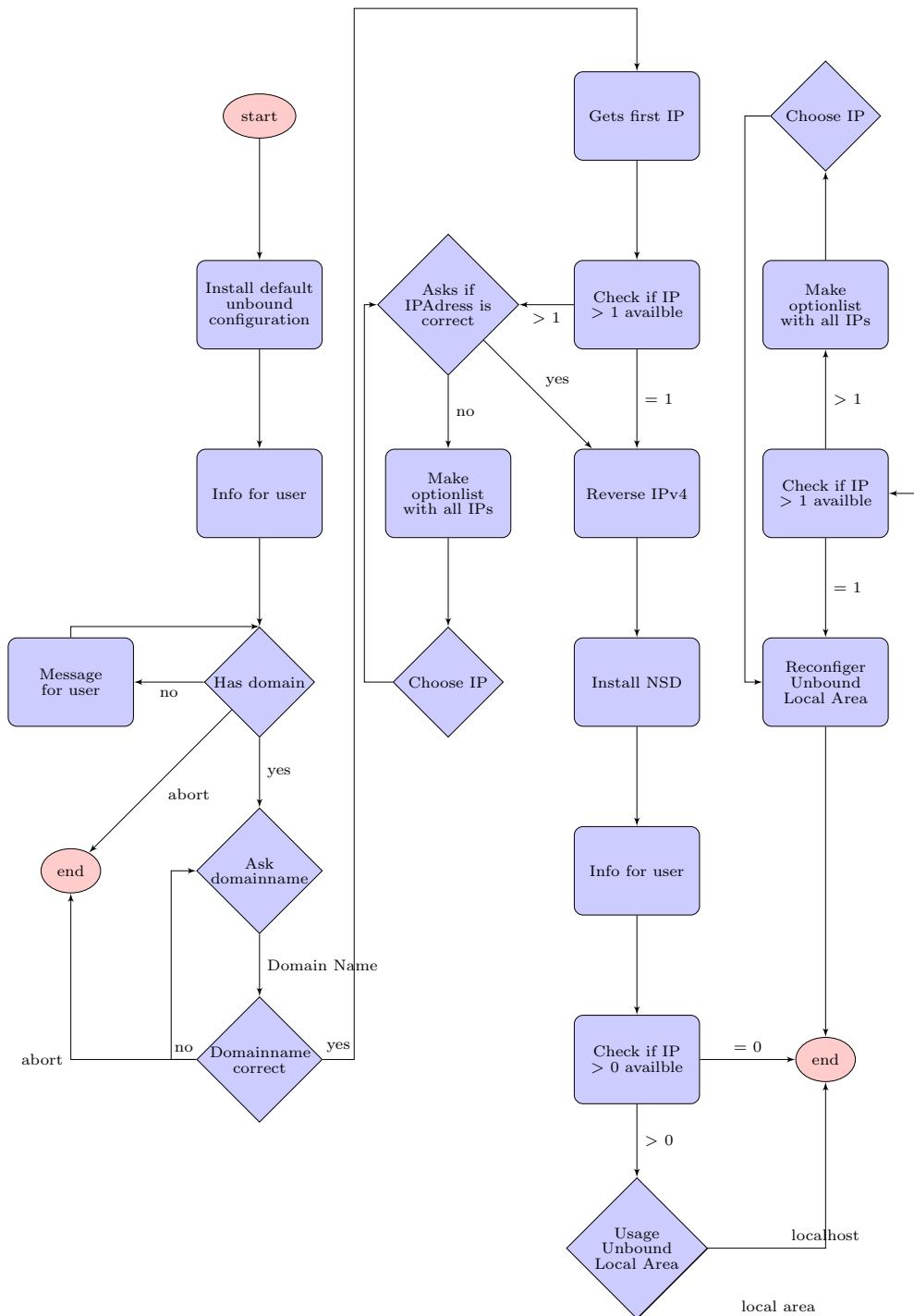


Figure 2.4: DNS process diagram

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 usermanagement 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.

```

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

```

Display

Show all unix users on the system:

```

1 <INFO> - Mon Jan 14 09:44:29 UTC 2019 - Number of users selected: 0
2 *** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
  unselect/show/quit) display
3
4 <INFO> - Mon Jan 14 09:44:42 UTC 2019 - Displaying users for this system
5 root
6 sync

```

Add

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

```

1 <INFO> - Mon Jan 14 09:44:42 UTC 2019 - Number of users selected: 0
2 *** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
  unselect/show/quit) add
3
4 <INFO> - Mon Jan 14 09:45:07 UTC 2019 - Adding user for this system
5 *** QUESTION *** please enter the desired username to be added? alice
6
7 id: 'alice': no such user
8 Adding user 'alice' ...
9 Adding new group 'alice' (1000) ...

```



```

10 Adding new user 'alice' (1000) with group 'alice' ...
11 Creating home directory '/home/alice' ...
12 Copying files from '/etc/skel' ...
13 Enter new UNIX password:
14 Retype new UNIX password:
15 passwd: password updated successfully
16 Changing the user information for alice
17 Enter the new value, or press ENTER for the default
18     Full Name []:
19     Room Number []:
20     Work Phone []:
21     Home Phone []:
22     Other []:
23 Is the information correct? [Y/n] y
24 <INFO> - Mon Jan 14 09:45:22 UTC 2019 - Successfully added user alice, adding it to
      the list for ssh
25
26 *** QUESTION *** Do you want to add sudo privileges for the user alice? (y/N) y
27 <INFO> - Mon Jan 14 09:45:28 UTC 2019 - Adding sudo privileges for user alice
28 <INFO> - Mon Jan 14 09:45:28 UTC 2019 - Successfully added sudo privileges for user
      alice

```

Show

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

```

1 <INFO> - Mon Jan 14 09:45:28 UTC 2019 - Number of users selected: 1
2 *** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
      unselect/show/quit) alice
3 <INFO> - Mon Jan 14 09:45:46 UTC 2019 - Displaying selected users for service ssh
4 <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

```

1 <INFO> - Mon Jan 14 09:45:46 UTC 2019 - Number of users selected: 1
2 *** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
      unselect/show/quit) unselect
3 <INFO> - Mon Jan 14 09:45:56 UTC 2019 - Unselecting user for service ssh
4 *** QUESTION *** please enter the desired username to be removed from selection?
      alice
5 <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

```

1 <INFO> - Mon Jan 14 09:45:59 UTC 2019 - Number of users selected: 0
2 *** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
      unselect/show/quit) select
3 <INFO> - Mon Jan 14 09:46:07 UTC 2019 - Selecting user for service ssh
4 *** QUESTION *** please enter the desired username to be selected? alice
5 <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)

```

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

```

Quit

Exit the function

```

1 <INFO> - Mon Jan 14 09:46:46 UTC 2019 - Number of users selected: 1
2 *** 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.

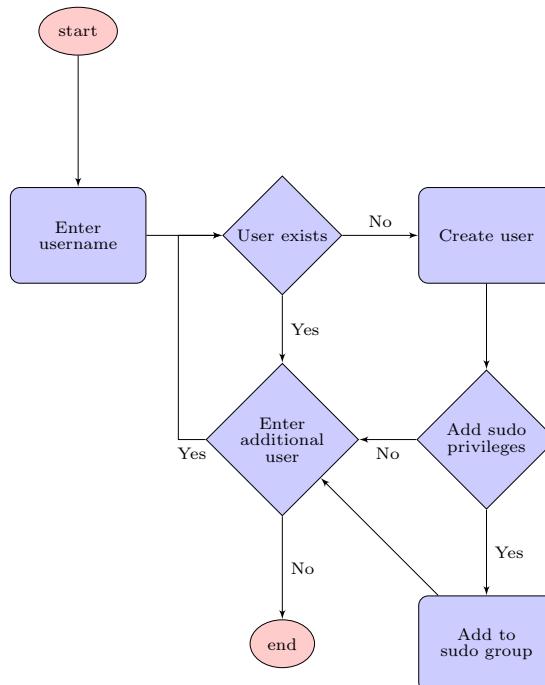


Figure 2.5: User management process diagram

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

```

1 <SSH> - Mon Jan 14 09:44:29 UTC 2019 - Perform actions on SSH
2 <SSH> - Mon Jan 14 09:44:29 UTC 2019 - Perform install on SSH
3 <INFO> - Mon Jan 14 09:44:29 UTC 2019 - Doing user handling for SSH configuration
4 Usage:
5     This function helps you manage the users on this system and select the ones
       you wish to provision for the ssh service.
6     Following actions are available:
7         help:           Display this help
8         display:        Show all unix users on this system
9         add:            Add a unix user to this system (this implies the
                   select action)
10        delete:         Remove a unix user from this system (this implies the
                   unselect action)
11        select:         Add a existing unix user to the list of users which
                   will be provisioned for the service ssh
12        unselect:       Remove a user from the list of users which will be
                   provisioned for the service ssh
13        show:           Show the list of users which will be provisioned for
                   the service ssh
14        quit:           Exit this function
15
16 <INFO> - Mon Jan 14 09:44:42 UTC 2019 - Number of users selected: 0
17 *** QUESTION *** what action do you like to choose? (help/display/add/delete/select/
   unselect/show/quit) add
18
19 <INFO> - Mon Jan 14 09:45:07 UTC 2019 - Adding user for this system
20 *** QUESTION *** please enter the desired username to be added? alice
21
22 id: 'alice': no such user
23 Adding user 'alice' ...
24 Adding new group 'alice' (1000) ...
25 Adding new user 'alice' (1000) with group 'alice' ...
26 Creating home directory '/home/alice' ...
27 Copying files from '/etc/skel' ...
28 Enter new UNIX password:
29 Retype new UNIX password:
30 passwd: password updated successfully
31 Changing the user information for alice
32 Enter the new value, or press ENTER for the default
33     Full Name []:
34     Room Number []:
35     Work Phone []:
36     Home Phone []:
37     Other []:
38 Is the information correct? [Y/n] y
39 <INFO> - Mon Jan 14 09:45:22 UTC 2019 - Successfully added user alice, adding it to
   the list for ssh
40
41 *** QUESTION *** Do you want to add sudo privileges for the user alice? (y/N) y
42 <INFO> - Mon Jan 14 09:45:28 UTC 2019 - Adding sudo privileges for user alice
43 <INFO> - Mon Jan 14 09:45:28 UTC 2019 - Successfully added sudo privileges for user
   alice
44
45 <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 SHA256:82nk2iy01S6n+KJdIIIfGeR/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/bqEhk0b6DVsh
32 ...
33 VrxQPg0eipL3zr54Zq9SY6NC2BCu50ygDHWXsKwrBTnx0Hi262jo6bX7Kqmog4qX
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 hardened:

- Root login is not permitted
- Password login is not permitted
- X11 is not permitted
- Only secure algorithms are permitted



```

1 <INFO> - Mon Jan 14 09:57:45 UTC 2019 - Cleaning up..
2 <INFO> - Mon Jan 14 09:57:45 UTC 2019 - Hardening SSH daemon config
3
4 <INFO> - Mon Jan 14 09:57:45 UTC 2019 - Hardening sshd config (disable X11Forwarding,
   enable domainname lookup, disable root login, enabling only strong algorithms)
5
6 <INFO> - Mon Jan 14 09:57:45 UTC 2019 - Hardening complete
7
8 <INFO> - Mon Jan 14 09:57:45 UTC 2019 - Finishing up, restarting services
9
10 <INFO> - Mon Jan 14 09:57:45 UTC 2019 - Restarting all components for SSH
11
12 <INFO> - Mon Jan 14 09:57:45 UTC 2019 - SSH daemon configuration complete.
13 <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.

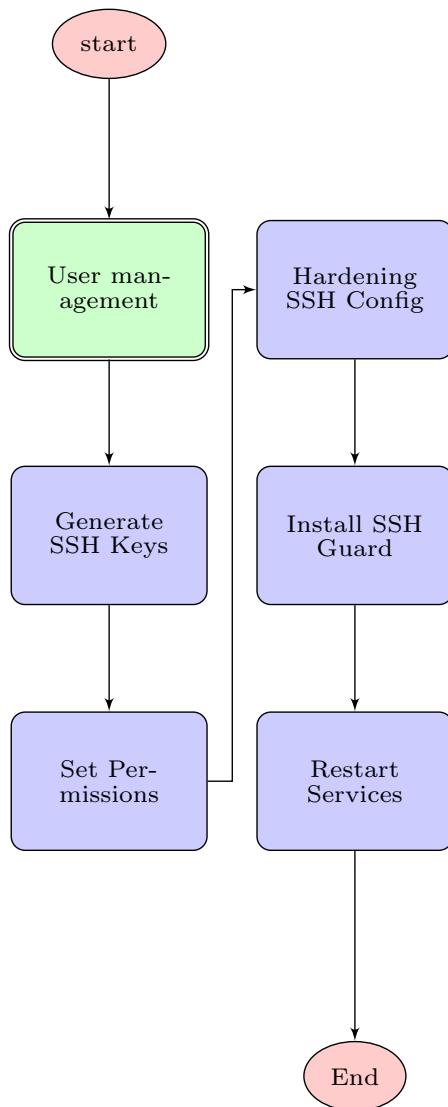


Figure 2.6: SSH process diagram

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

```

1 <Mail> - Mon Jan 14 11:29:52 UTC 2019 - Perform install on Mail
2 <INFO> - Mon Jan 14 11:29:52 UTC 2019 - Setting up MX and SPF records in dns
3
4 <INFO> - Mon Jan 14 11:29:52 UTC 2019 - Appending DNS records for the mailserver to
   zonefile
5
6 <INFO> - Mon Jan 14 11:29:52 UTC 2019 - Reloading zone files..
7
8 <INFO> - Mon Jan 14 11:29:52 UTC 2019 - Installing mailserver packages (postfix,
   mailutils, dovecot)
9 <INFO> - Mon Jan 14 11:30:15 UTC 2019 - Will install 'postfix-pcre' now. Please wait
   ...
10 .....
11 <INFO> - Mon Jan 14 11:30:21 UTC 2019 - Package 'postfix-pcre' is installed now.
12 <INFO> - Mon Jan 14 11:30:21 UTC 2019 - Will install 'postfix-policyd-spf-python' now.
   Please wait...
13 .....
14 <INFO> - Mon Jan 14 11:30:27 UTC 2019 - Package 'postfix-policyd-spf-python' is
   installed now.
15 <INFO> - Mon Jan 14 11:30:28 UTC 2019 - Will install 'mailutils' now. Please wait...
16 .....
17 <INFO> - Mon Jan 14 11:30:36 UTC 2019 - Package 'mailutils' is installed now.
18 <INFO> - Mon Jan 14 11:30:36 UTC 2019 - Will install 'letsencrypt' now. Please wait...
19 .....
20 <INFO> - Mon Jan 14 11:30:51 UTC 2019 - Package 'letsencrypt' is installed now.
21 <INFO> - Mon Jan 14 11:30:51 UTC 2019 - Will install 'dovecot-core' now. Please wait
   ...
22 .....
23 <INFO> - Mon Jan 14 11:31:11 UTC 2019 - Package 'dovecot-core' is installed now.
24 <INFO> - Mon Jan 14 11:31:11 UTC 2019 - Will install 'dovecot-imapd' now. Please wait
   ...
25 .....
26 <INFO> - Mon Jan 14 11:31:24 UTC 2019 - Package 'dovecot-imapd' is installed now.
27 <INFO> - Mon Jan 14 11:31:25 UTC 2019 - Will install 'opendkim' now. Please wait...
28 .....
29 <INFO> - Mon Jan 14 11:31:33 UTC 2019 - Package 'opendkim' is installed now.
30 <INFO> - Mon Jan 14 11:31:33 UTC 2019 - Will install 'opendkim-tools' now. Please wait
   ...
31 .....
32 <INFO> - Mon Jan 14 11:31:38 UTC 2019 - Package 'opendkim-tools' is installed now.

```



```

33 <INFO> - Mon Jan 14 11:31:39 UTC 2019 - Will install 'opendmarc' now. Please wait...
34 .....
35 <INFO> - Mon Jan 14 11:31:48 UTC 2019 - Package 'opendmarc' is installed now.
36 <INFO> - Mon Jan 14 11:31:48 UTC 2019 - Will install 'zip' now. Please wait...
37 .....
38 <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 management, for further information see section 2.5

```

1 <INFO> - Mon Jan 14 11:31:54 UTC 2019 - Configure Mail Hardening (TLS, SPF, DKIM,
2   DMARC, dovecot, client certificate login)
3 Usage:
4   This function helps you manage the users on this system and select the ones
5     you wish to provision for the mail service.
6   Following actions are available:
7     help:           Display this help
8     display:        Show all unix users on this system
9     add:            Add a unix user to this system (this implies the select
10    action)
11    delete:         Remove a unix user from this system (this implies the
12      unselect action)
13    select:         Add a existing unix user to the list of users which will
14      be provisioned for the service mail
15    unselect:       Remove a user from the list of users which will be
16      provisioned for the service mail
17    show:           Show the list of users which will be provisioned for the
18      service mail
19    quit:           Exit this function
20
21 <INFO> - Mon Jan 14 11:34:31 UTC 2019 - Number of users selected: 0
22 *** QUESTION *** what action do you like to choose? (display/add/delete/select/
23   unselect/show/quit) select
24
25 <INFO> - Mon Jan 14 11:34:33 UTC 2019 - Selecting user for service mail
26
27 *** QUESTION *** please enter the desired username to be selected? alice
28
29 <INFO> - Mon Jan 14 11:34:35 UTC 2019 - Selected alice for mail
30
31 <INFO> - Mon Jan 14 11:34:35 UTC 2019 - Number of users selected: 1
32 *** QUESTION *** what action do you like to choose? (display/add/delete/select/
33   unselect/show/quit) quit
34
35 <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 and receive mail. The script now configures all the necessary postfix components:

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



```
1 <INFO> - Mon Jan 14 11:34:39 UTC 2019 - Mapping users to mail addresses
2 <INFO> - Mon Jan 14 11:34:39 UTC 2019 - Adding users to alias and canonical file
3
4 <INFO> - Mon Jan 14 11:34:39 UTC 2019 - Adding supplementary postmaster user for dmarc
5   reporting
6
7 <INFO> - Mon Jan 14 11:34:39 UTC 2019 - Setting up TLS with letsencrypt
8
9 <INFO> - Mon Jan 14 11:34:39 UTC 2019 - Running letsencrypt to obtain a certificate
10
11 <INFO> - Mon Jan 14 11:34:40 UTC 2019 - Will install 'certbot' now. Please wait...
12 ...
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
```



```

65 opendkim-genkey: DNS TXT record written to 2019011411.txt
66
67 <INFO> - Mon Jan 14 11:34:54 UTC 2019 - Reloading systemd units
68
69 <INFO> - Mon Jan 14 11:34:55 UTC 2019 - Generating DNS records for opendkim
70
71 <INFO> - Mon Jan 14 11:34:55 UTC 2019 - Setting up DMARC (anti spoofing measure)
72
73 <INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring opendmarc
74
75 <INFO> - Mon Jan 14 11:34:55 UTC 2019 - Reloading systemd units
76
77 <INFO> - Mon Jan 14 11:34:55 UTC 2019 - Adding DNS records for opendmarc
78
79 <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. 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

```

1 <INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring dovecot as imap server
2
3 <INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring dovecot
4
5 <INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring dovecot service
6
7 <INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring dovecot SSL
8
9 <INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring dovecot SSL
10
11 <INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring external auth extension
12
13 <INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring postfix for client certificates
14
15 <INFO> - Mon Jan 14 11:34:55 UTC 2019 - Configuring client certificate authentication
16
17 <INFO> - Mon Jan 14 11:34:57 UTC 2019 - Generating certificate authority, please enter
     a passphrase when prompted:
18
19 Enter New CA Key Passphrase: *****
20 Re-Enter New CA Key Passphrase: *****
21 Generating RSA private key, 4096 bit long modulus
22 .....
23 .....
24 .....
25 ..... ++
26 ..... ++
27 e is 65537 (0x010001)
28 Enter pass phrase for /root/src/EasyRSA-3.0.5/pki/private/ca.key: *****
29 /root/src
30
31 <INFO> - Mon Jan 14 11:35:07 UTC 2019 - Generating key and certificate for user alice
32 <INFO> - Mon Jan 14 11:35:07 UTC 2019 -
     IMPORTANT - make sure you remember ALL the passphrases! You can download your certificate and key
33 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/alice/alice_artifacts.zip ./
50
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.

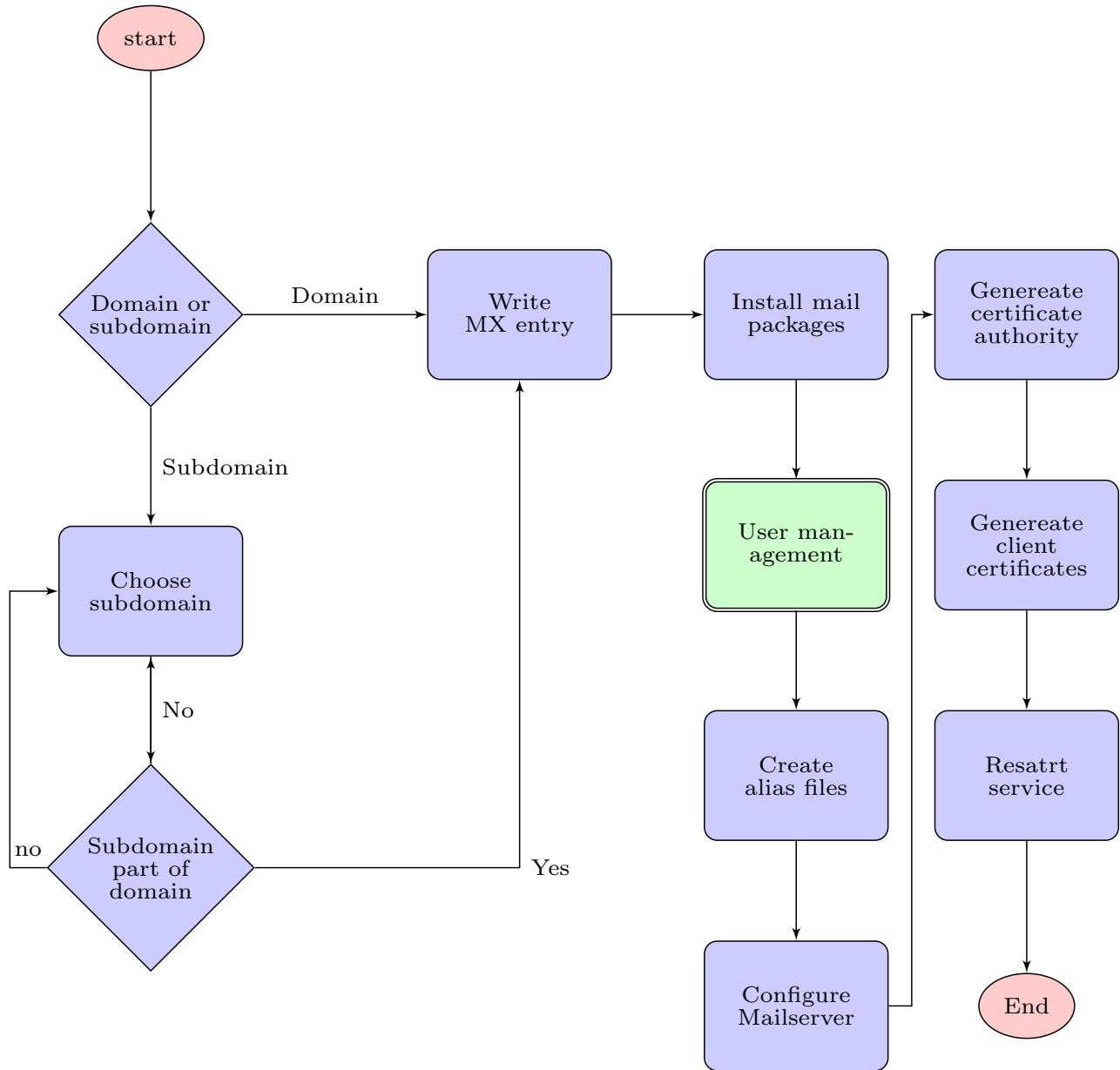


Figure 2.7: Email process diagram

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.

```

1 <INFO> - Tue Jan  8 11:24:21 UTC 2019 - Starting WEB Configurations.
2 <INFO> - Tue Jan  8 11:24:21 UTC 2019 - Will install 'nginx' now. Please wait...
3 .....
4 <INFO> - Tue Jan  8 11:24:32 UTC 2019 - Package 'nginx' is installed now.
5 <INFO> - Tue Jan  8 11:24:33 UTC 2019 - Will install 'certbot' now. Please wait...
6 .....
7 <INFO> - Tue Jan  8 11:24:36 UTC 2019 - Package 'certbot' is installed now.
8 <INFO> - Tue Jan  8 11:24:36 UTC 2019 - Will install 'python-certbot-nginx' now.
   Please wait...
9 .....
10 <INFO> - Tue Jan  8 11:24:43 UTC 2019 - Package 'python-certbot-nginx' is installed
    now.
11 <INFO> - Tue Jan  8 11:24:43 UTC 2019 - Will install 'apache2' now. Please wait...
12 .....
13 <INFO> - Tue Jan  8 11:24:57 UTC 2019 - Package 'apache2' is installed now.

```

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

```

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

```

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

```

1 <INFO> - Tue Jan  8 11:24:57 UTC 2019 - Start openssl to generate a ssl pem file.
2 Generating DSA parameters, 4096 bit long prime
3 ....+....+....+....+.....
4 ++++++++
5 .....+..+.+.....+....+.....
6 .....+....+....+.....
7 <INFO> - Tue Jan  8 11:25:08 UTC 2019 - Done. Your file is located here: /etc/ssl/
  dh4096.pem. Will start certbot.
8 Saving debug log to /var/log/letsencrypt/letsencrypt.log
9 Plugins selected: Authenticator nginx, Installer nginx
10 Obtaining a new certificate
11 Performing the following challenges:
12 http-01 challenge for exemplerrun.cf
13 http-01 challenge for www.examplerun.cf
14 Waiting for verification...
15 Cleaning up challenges
16
17 IMPORTANT NOTES:
18 - Congratulations! Your certificate and chain have been saved at:
19 /etc/letsencrypt/live/examplerun.cf/fullchain.pem
20 Your key file has been saved at:
21 /etc/letsencrypt/live/examplerun.cf/privkey.pem
22 Your cert will expire on 2019-04-08. To obtain a new or tweaked
23 version of this certificate in the future, simply run certbot
24 again. To non-interactively renew *all* of your certificates, run
25 "certbot renew"
26 - If you like Certbot, please consider supporting our work by:
27
28 Donating to ISRG / Lets Encrypt:  https://letsencrypt.org/donate
29 Donating to EFF:                  https://eff.org/donate-le

```



Nginx will then be hardened:

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

```

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

```

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

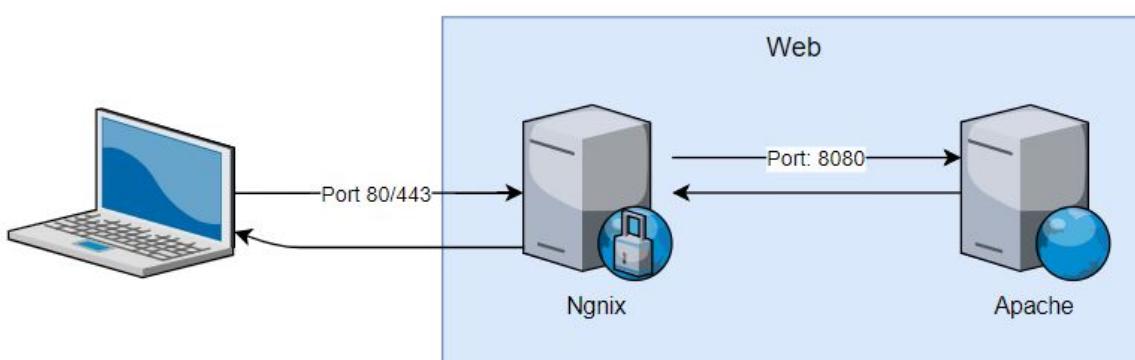


Figure 2.8: Architecture Web

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: <https://nmap.org/>

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 exemplерun.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 exemplерun.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 exemplерun.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 exemplерun.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 exemplерun.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 exemplерun.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 exemplерun.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 exemplерun.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 smptd
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 smtpts
993/tcp   open  ssl/imap Dovecot imapd (Ubuntu)
Service Info: Host: mail.exemplерun.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, Cloudflare 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/>

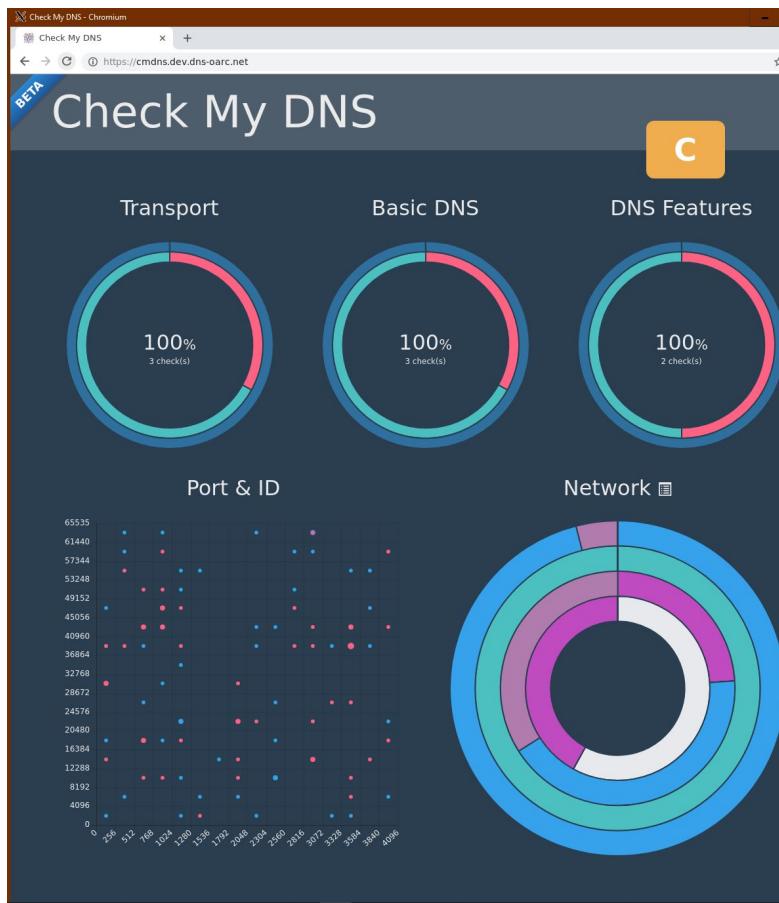


Figure 3.4: Name resolver BEFORE

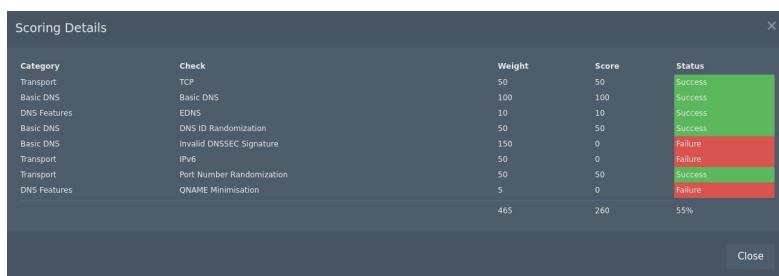


Figure 3.5: Name resolver details BEFORE

AFTER script

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

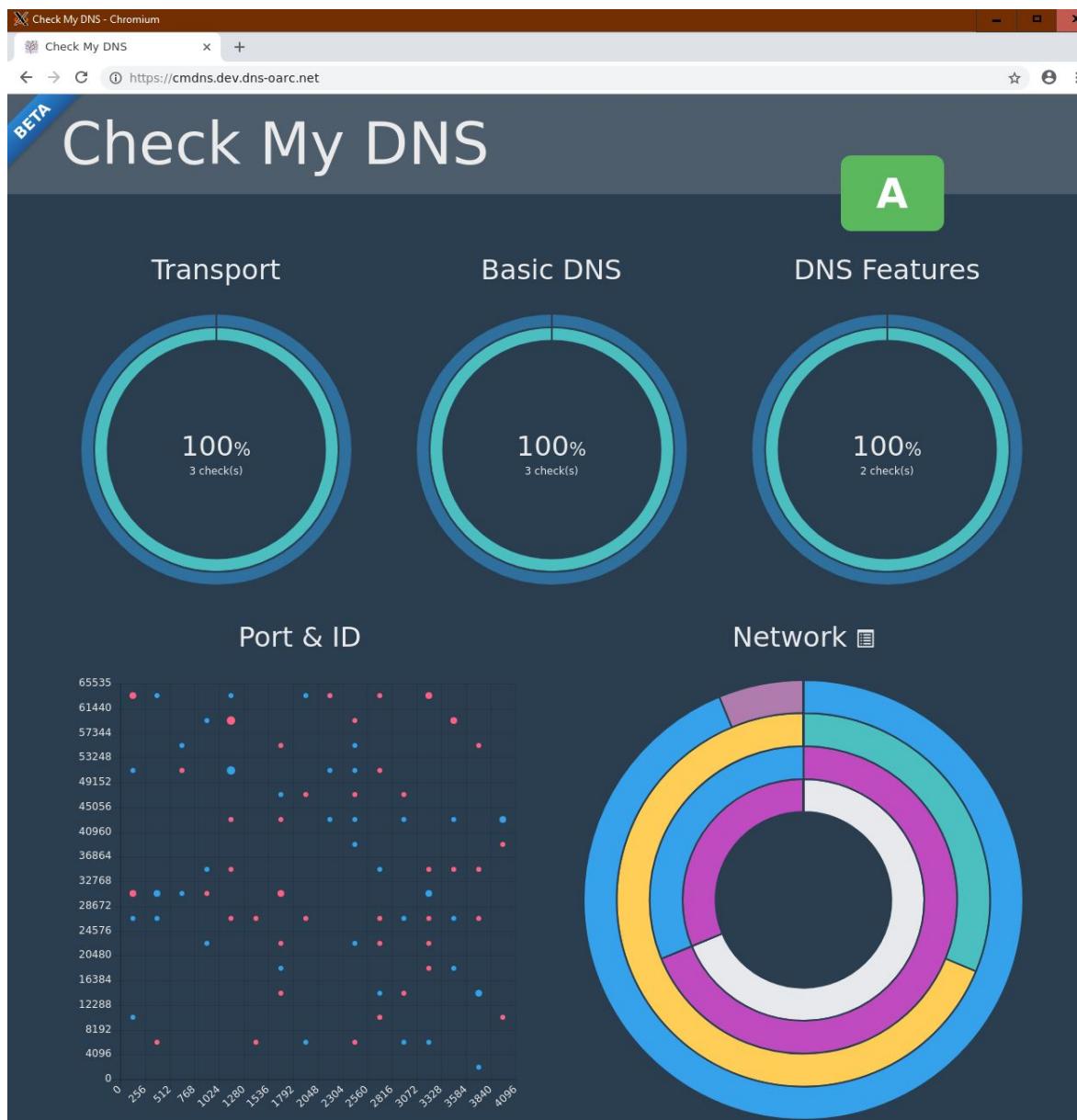


Figure 3.6: Name resolver **AFTER**

Scoring Details				
Category	Check	Weight	Score	Status
Transport	IPv6	50	50	Success
Transport	Port Number Randomization	50	50	Success
DNS Features	QNAME Minimisation	5	5	Success
Transport	TCP	50	50	Success
Basic DNS	Basic DNS	100	100	Success
DNS Features	EDNS	10	10	Success
Basic DNS	DNS ID Randomization	50	50	Success
Basic DNS	Invalid DNSSEC Signature	150	150	Success
		465	465	100%
Close				

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/>

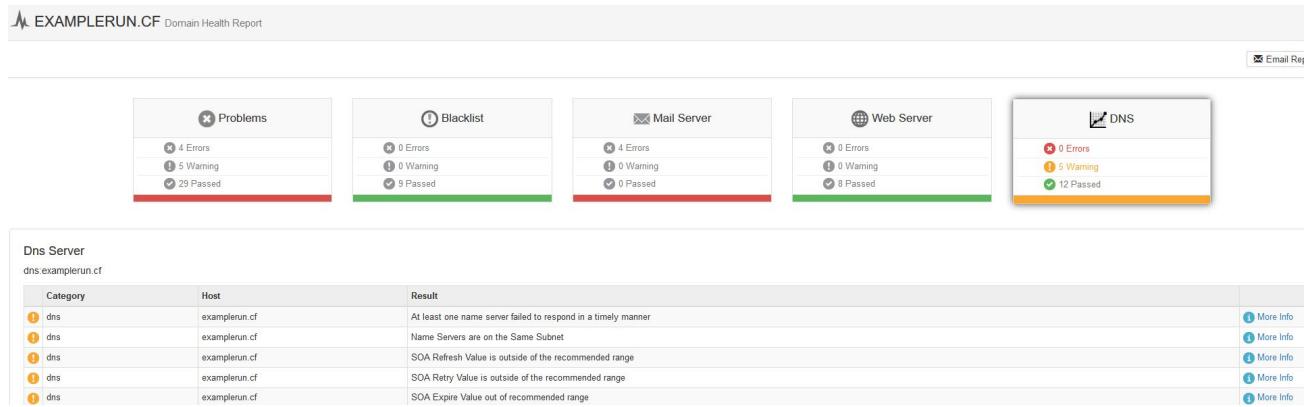


Figure 3.8: Authoritative DNS test BEFORE

After script

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

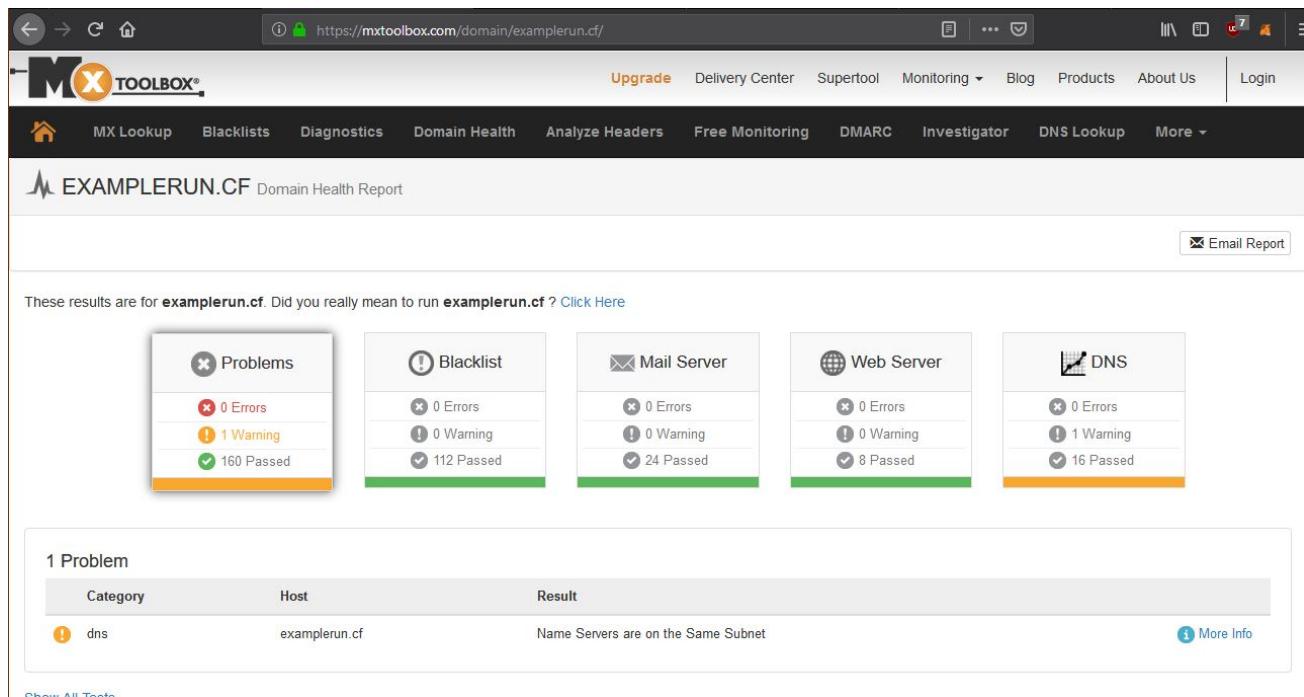


Figure 3.9: Authoritative DNS test AFTER

3.3 SSH

3.3.1 SSH daemon

You receive 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 receive the following result, some of the “Key Exchange Algorithms” and “MAC Algorithms” are weak.

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-nistp511	Elliptic Curve Diffie-Hellman on NIST P-511 curve with SHA-512 hash ⓘ Possible NSA backdoor	Secure
diffie-hellman-group1-sha1	Diffie-Hellman with 2048-bit Oakley Group 1 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.

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 23:00:00 UTC (3 seconds ago)

Key Exchange Algorithms

diffie-hellman-group-exchange-sha256	Diffie-Hellman with MODP Group Exchange 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

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

hmac-sha2-512-etm@openssh.com	Hash-based MAC using SHA-512 (Encrypt-then-MAC) by OpenSSH	Secure
hmac-sha2-256-etm@openssh.com	Hash-based MAC using SHA-256 (Encrypt-then-MAC) by OpenSSH	Secure
umac-128-etm@openssh.com	128-bit Universal Hashing MAC (Encrypt-then-MAC) by OpenSSH ⓘ	Secure
hmac-sha2-512	Hash-based MAC using SHA-512 ⓘ	Secure
hmac-sha2-256	Hash-based MAC using SHA-256 ⓘ	Secure
umac-128@openssh.com	128-bit Universal Hashing MAC by OpenSSH ⓘ	Secure

Compression Algorithms

none	No compression ⓘ	Secure
zlib@openssh.com	zlib compression by OpenSSH ⓘ	Secure

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>, <https://emailsecuritygrader.com>)

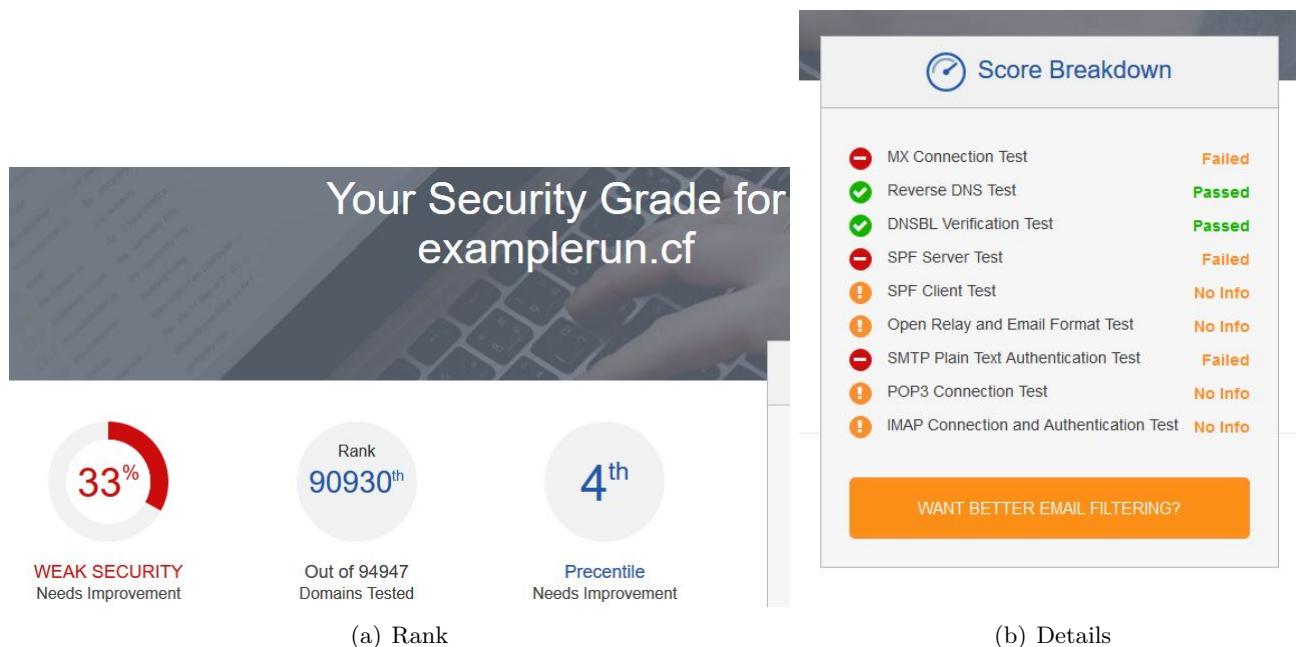


Figure 3.12: Mail BEFORE (emailsecuritygrader.com)

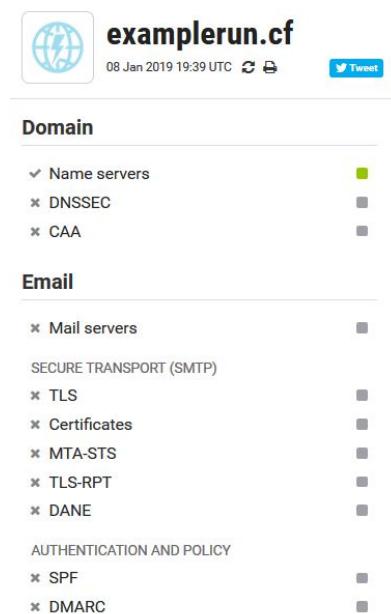


Figure 3.13: Mail BEFORE (hardenize.com)

AFTER script

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

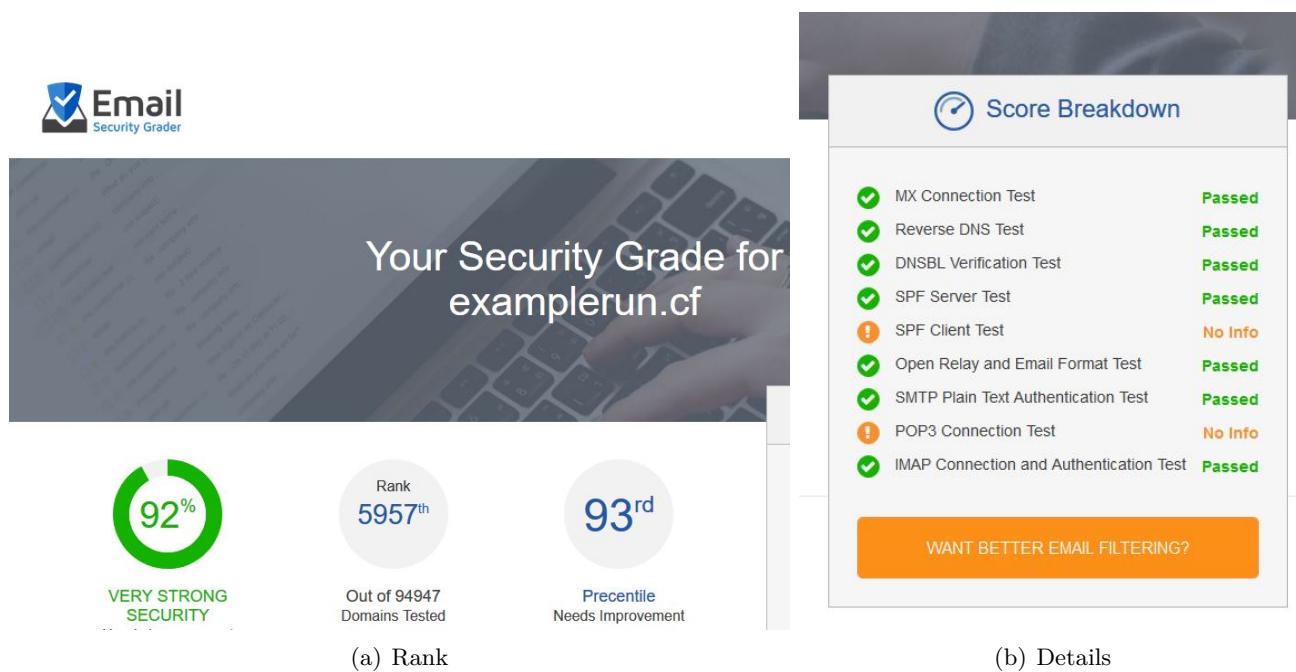


Figure 3.14: Mail **AFTER** (emailsecuritygrader.com)

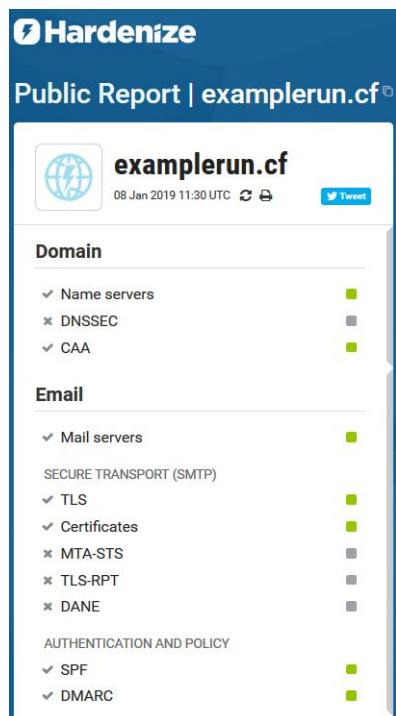


Figure 3.15: Mail **AFTER** (hardenize.com)

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:

Originalnachricht

Nachrichten-ID	<20190108191134.5186C40481@mail.examplerun.cf>
Erstellt am:	8. Januar 2019 um 20:11 (Nach 30 Sekunden zugestellt)
Von:	test@mail.examplerun.cf Mit mail (GNU Mailutils 3.4) gesendet
An:	[REDACTED]@gmail.com
Betreff:	this is a test
SPF:	PASS mit IP-Adresse 104.248.137.212 Weitere Informationen
DMARC:	'FAIL' Weitere Informationen

Figure 3.16: Mail header BEFORE

AFTER script

And here after:

Originalnachricht

Nachrichten-ID	<20190108172509.2C6C14088C@mail.examplerun.cf>
Erstellt am:	8. Januar 2019 um 18:25 (Nach 0 Sekunden zugestellt)
Von:	test@examplerun.cf Mit mail (GNU Mailutils 3.4) gesendet
An:	[REDACTED]@gmail.com
Betreff:	this is a test
SPF:	PASS mit IP-Adresse 104.248.137.212 Weitere Informationen
DKIM:	'PASS' mit Domain examplerun.cf Weitere Informationen
DMARC:	'PASS' Weitere Informationen

Figure 3.17: Mail header AFTER



3.5 Web

The web part could be tested very well with <https://www.hardenize.com>. 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

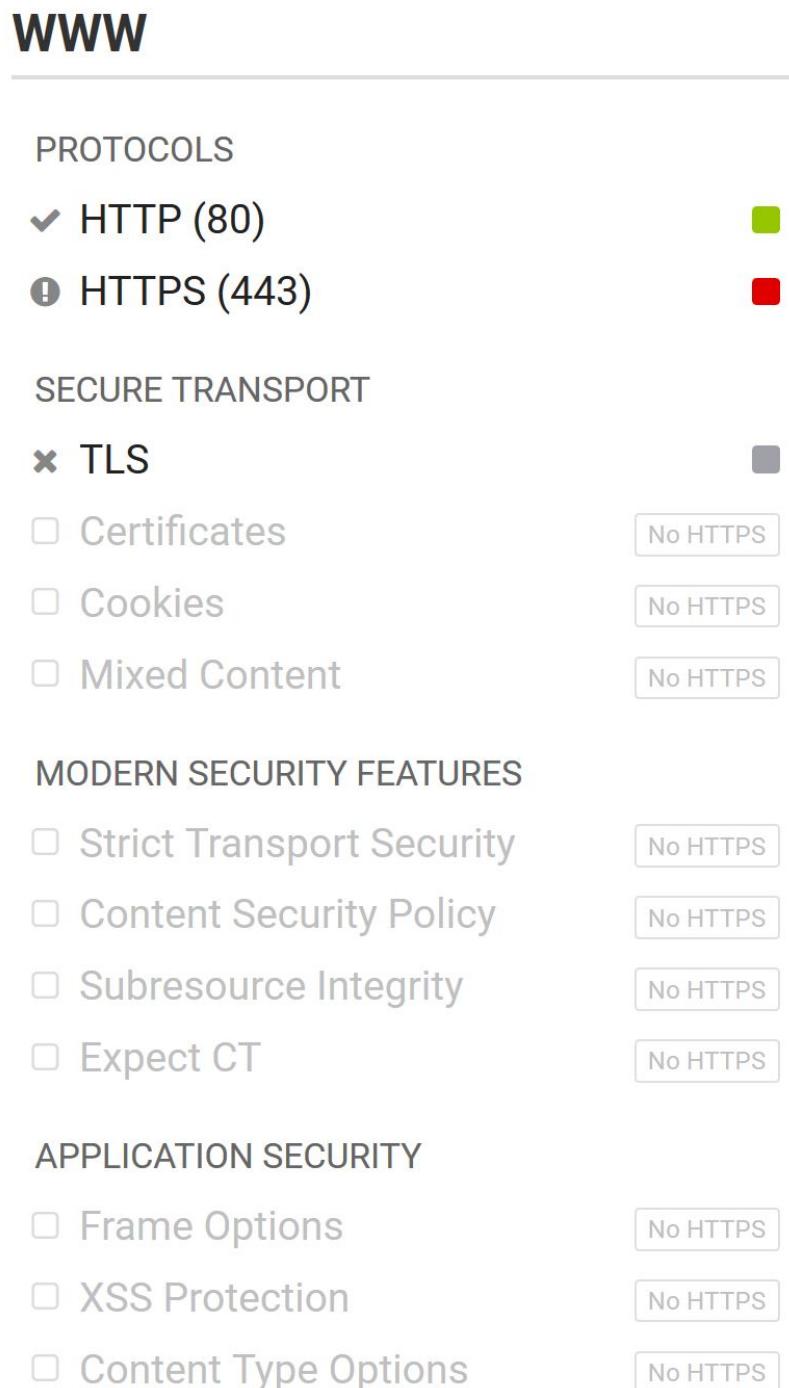


Figure 3.18: Web BEFORE

AFTER script

WWW	
<hr/>	
PROTOCOLS	
✓ HTTP (80)	
✓ HTTPS (443)	
<hr/>	
SECURE TRANSPORT	
✓ TLS	
✓ Certificates	
✓ Cookies	
✓ Mixed Content	
<hr/>	
MODERN SECURITY FEATURES	
✓ Strict Transport Security	
✗ Content Security Policy	
✓ Subresource Integrity	
✗ Expect CT	
<hr/>	
APPLICATION SECURITY	
✓ Frame Options	
✓ XSS Protection	
✓ Content Type Options	

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

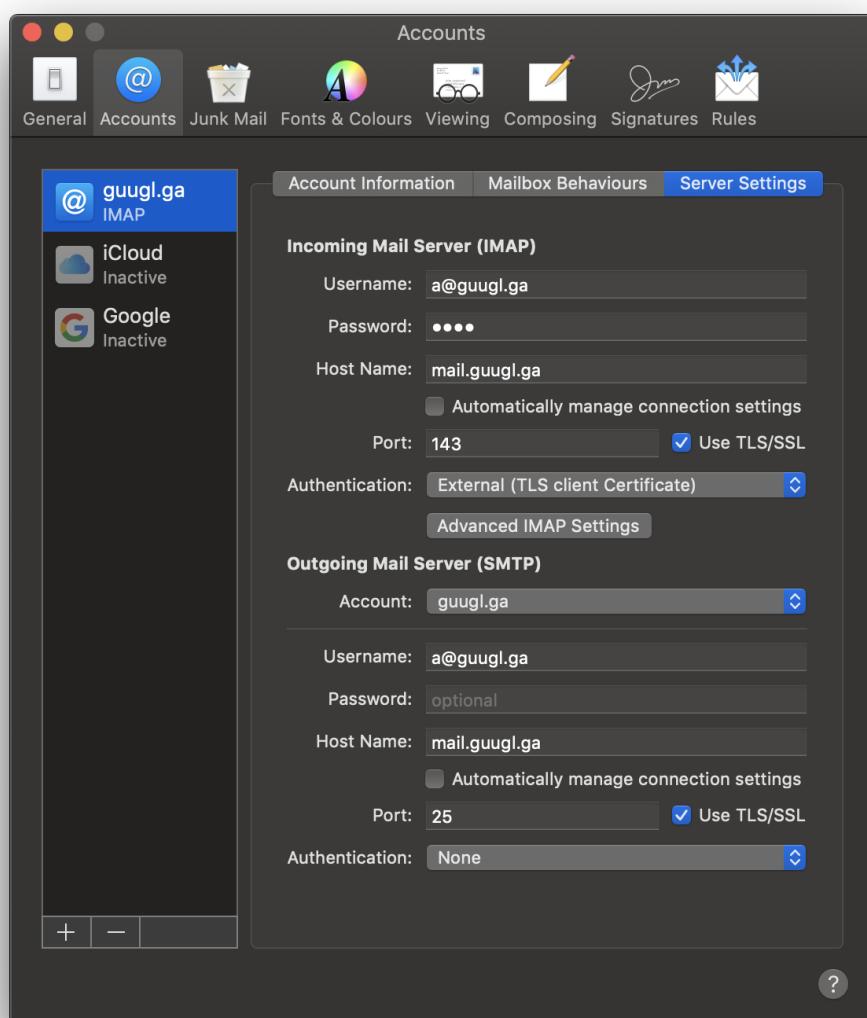


Figure 4.1: Mail server config

4.1.2 Mail SMTP settings

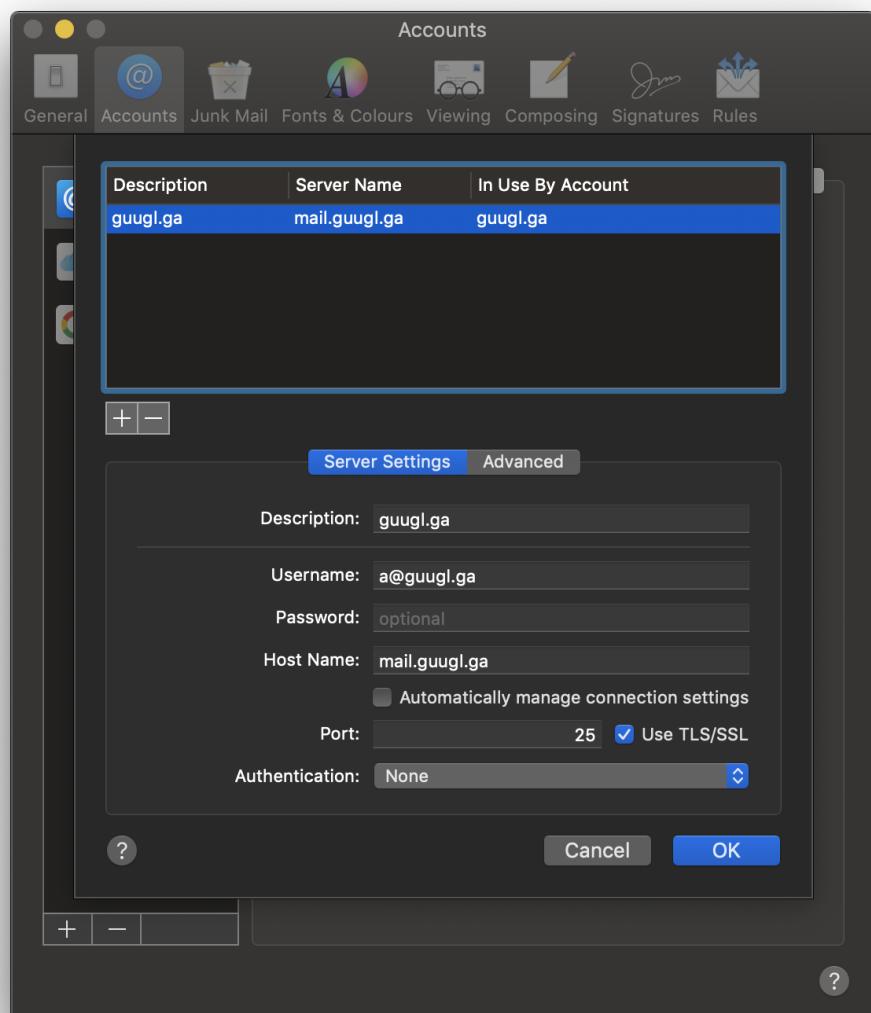


Figure 4.2: Mail SMTP settings

4.1.3 Mail IMAP TLS setting

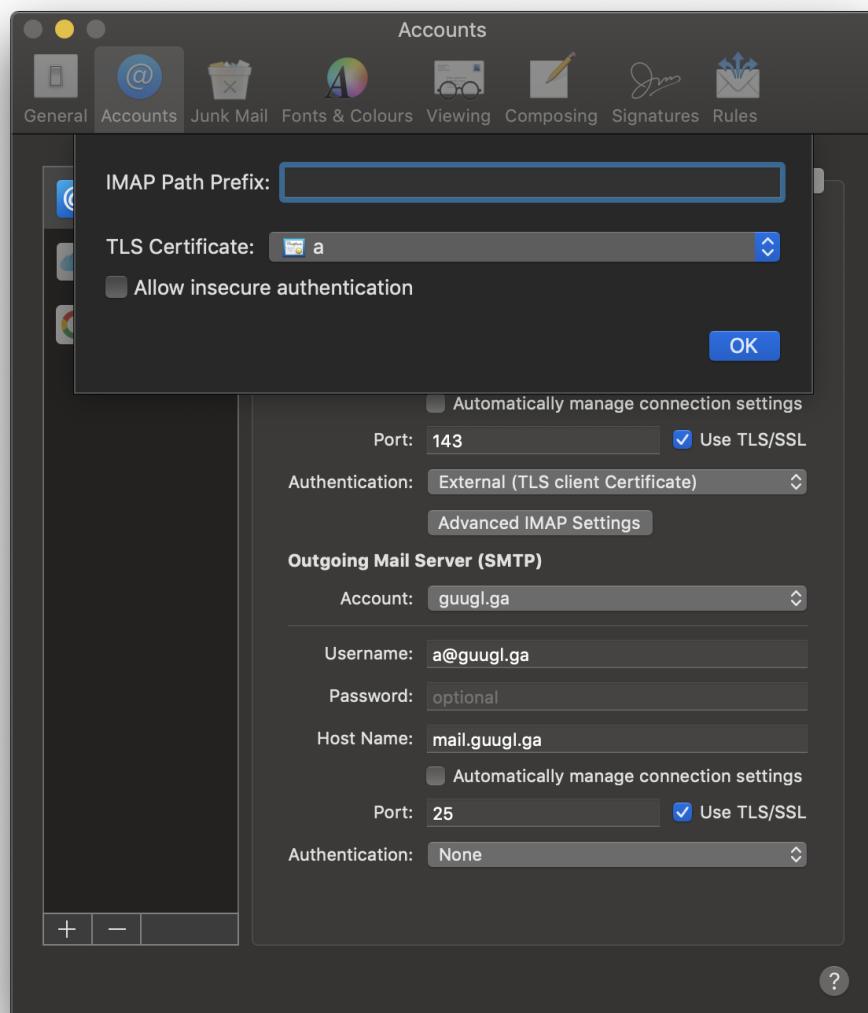


Figure 4.3: Mail IMAP TLS setting

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 subroutines 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 privilege to do this in a secure and anonymous matter. We built this script to provide every user, a maximal secure server, with a minimal need of information. Although 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” [2], 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 [3]
- DMARC Setup [1]
- Configuring HTTPS servers [4]

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

Copyright 2018 Ismael Riedo, Jan Henzi, Fridolin Zurlinden, Bern University of Applied Sciences

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



8 Slides



Berner
Fachhochschule

BTI7301 Project 1

Mail Server Set-Up &
Security-Hardening Script
Bern University of Applied Sciences

Frido Zurlinden
Ismael Riedo
Jan Henzi
Tutor: Dr. Simon Kramer



Introduction

Berner
Fachhochschule

IT security for everyone

- Transparency
- Simple
- Vendor independent



Berner
Fachhochschule

Contents

Introduction

Goals and Requirements

Technical Solution / Realization

Challenges



Goals and Requirements

Berner
Fachhochschule

- Easy as pie “Complexity is the enemy of security”
- Brick by brick
- Power to the people (No vendor lock-in)
- Security by default

Technical Solution

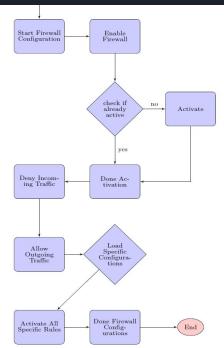
Berner
Fachhochschule

Introduction

- Setup
- Firewall
- DNS
- SSH
- Mail

Firewall

Berner
Fachhochschule



src/fw/fw.sh
 src/fw/enableUfw.sh
 src/fw/controlITraffic.sh
 src/fw/specificConfigurations.sh
 src/files/fw.conf

Setup

Berner
Fachhochschule

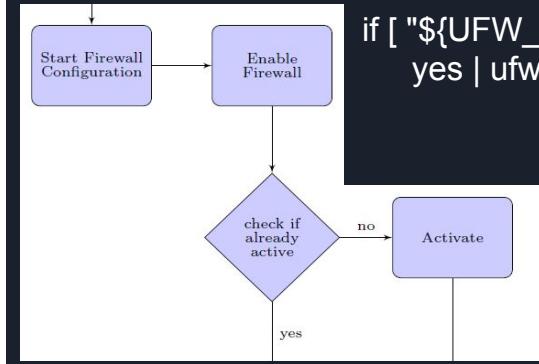
“Main Script” / “Entry point”

Flags - Leading pleasantly through the script

Handle State - What happens at the second time?

Firewall

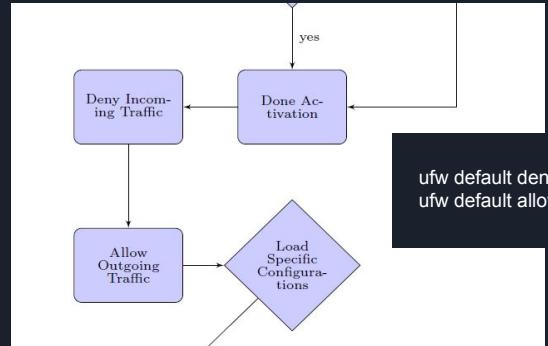
Berner
Fachhochschule





Firewall

Berner
Fachhochschule



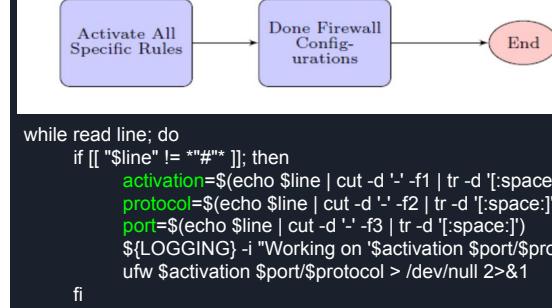
Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script

9



Firewall

Berner
Fachhochschule



```
$ cat ./files/fw.conf
# SSH
allow - tcp - 22
allow - udp - 22
# DNS
allow - tcp - 53
allow - udp - 53
```

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script

10



DNS

Berner
Fachhochschule



Unbound is a validating, recursive,
caching DNS resolver



NSD is an authoritative DNS name
server

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script

11



DNS

Berner
Fachhochschule



Unbound is a validating, recursive,
caching DNS resolver



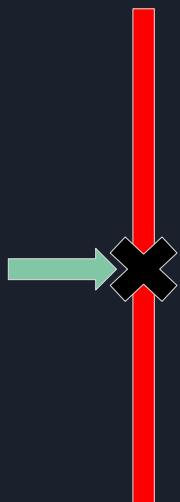
NSD is an authoritative DNS name
server

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script

12

DNS

Berner
Fachhochschule



Unbound is a validating, recursive,
caching DNS resolver

NSD is an authoritative DNS name
server

13

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script

Unbound

Berner
Fachhochschule

```
# 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"
```

```
# 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

14

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script

Unbound

Berner
Fachhochschule

```
# 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"
```

```
# 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

15

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script

Unbound

Berner
Fachhochschule

```
# 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"
```

```
# 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

16

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script



NSD

Berner

Fachhochschule

```
$ORIGIN exemplerun.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=<DKIM KEY>")
_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=afr\059 rua=mailto:postmaster@examplerun.cf"
```

17

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script



SSH

Berner

Fachhochschule

- SSH config hardening

```
X11Forwarding no
UseDNS yes
PermitRootLogin no
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-nistp256
KexAlgorithms
curve25519-sha256@libssh.org,ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,diffie-hellman-group-exchange-sha256
Ciphers
chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr
```

18

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script



SSH

Berner

Fachhochschule

- Secure SSH Keys

```
ssh-keygen -b 4096 -C "$user@$DOMAIN" -E sha256 -N mys3cr3t -t rsa -f
/home/alice/.ssh/id_rsa
```

19

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script



Mail - Postfix

Berner

Fachhochschule



POSTFIX

20

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script



Mail - TLS

Berner
Fachhochschule

- TLS - Avoid clear text whenever possible
- DOVECOT
 - IMAP Server
 - Mutual Authentication



Mail - TLS

Berner
Fachhochschule

- Letsencrypt to obtain certificate


```
certbot certonly --agree-tos --standalone -n -m postmaster@examplerun.cf -d mail.examplerun.cf
```
- Only use secure protocols (>TLSv1.2) and strong ciphers

```
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
smtpd_tls_exclude_ciphers = EXP, MEDIUM, LOW, DES, 3DES, SSLV2
smtpd_tls_ciphers = EXP, MEDIUM, LOW, DES, 3DES, SSLV2
tls_high_cipherlist =
KEDCDSA:+KEDCDSHA:KEDH:+KEDH+SHA:+KEDH+CAMELLIA:KEDCDSA:+KEDCDSHA:RSA:+RSA+SHA:+RSA+CAMELLIA:!aNULL:!eNULL:!SSLV2:!RC4:!MD5:!DES:!EXP:!IS
ECDH:!IDEA:!3DES:!SHA
tls_preempt_cipherlist = yes
smtp_tls_ciphers = high
smtpd_tls_ciphers = high
```



Mail - Dovecot

Berner
Fachhochschule

- Require mutual authentication

```
ssl_verify_client_cert = yes
ssl_cert_username_field = CN
auth_ssl_username_from_cert = yes
```

- Map users

```
echo "alice: alice" >> /etc/aliases
echo "alice@example.cf alice@example.cf" >> /etc/postfix/canonical
echo "alice:::::" >> /etc/dovecot/users-external
```



Mail - Anti-SPAM Measures

Berner
Fachhochschule

- SPF (Sender Policy Framework)
- DKIM (DomainKeys Identified Mail)
- ADSP (Author Domain Signing Practices)
- DMARC (Domain-based Message Authentication, Reporting and Conformance)



Mail - SPF

Berner
Fachhochschule

```
/etc/postfix/master.cf
policyd-spf unix - n n - o spawn
    user=policyd-spf argv=/usr/bin/policyd-spf
```

```
/etc/postfix/main.cf
policyd-spf_time_limit = 3600
smtpd_recipient_restrictions =
...
reject_unauth_destination,
check_policy_service unix:private/policyd-spf,
```

DNS zone file:
IN TXT "v=spf1 mx a ~all"

25

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script



Mail - SPF

Berner
Fachhochschule

```
/etc/postfix/master.cf
```

Betreff:	this is a test
SPF:	PASS mit IP-Adresse 192.168.1.100

```
policyd-spf[21065]: Pass; identity@mailfrom; client-ip=127.0.0.1; helo=mail.examplelerun.cf;
envelope-from=test@examplelerun.cf; receiver=ismaelmartin.riedo@bfh.ch
```

reject_unauth_destination,
check_policy_service unix:private/policyd-spf,

DNS zone file:
IN TXT "v=spf1 mx a ~all"



26

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script



Mail - DKIM & ADSP

Berner
Fachhochschule

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=examplelerun.cf;
s=2019010811; t=154696309; bh=WN9jjFu//8qtE8luk5bNJWZe8eu6jn90bWUEPF3q5DU=;
h=Subject:To:Date:From;
b=SMsc9anWhENzauKUPtLtt1qHXH2Ig9InVCPhOb7uShzCISTzn/MOSmawxf7uBbe
mowBa2etGicYBjsMYzGrvNOCLiGznb9AJzD4EICCAbsRJCjHQILtyKPERcxius9wc
0wiIV/z13F2u90EJN5gtIMCdqXb9a2loncdYAUu52Fr0MEs6qyWlzj2KUNz1bBvht+
CkGhW8NJ10Bfrs4EPevI/q 2019010917._domainkey IN TXT (
dEKVz4WSYIRNqLNPFezXjw "v=DKIM1\059 h=sha256\059 k=rsa\059 s=email\059 p="
fEKZ8bjBPiT1Q==
```

27

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script



Mail - DKIM & ADSP

Berner
Fachhochschule

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=examplelerun.cf;
s=2019010811; t=154696309; bh=WN9jjFu//8qtE8luk5bNJWZe8eu6jn90bWUEPF3q5DU=;
h=Subject:To:Date:From;
b=SMsc9anWhENzauKUPtLtt1qHXH2Ig9InVCPhOb7uShzCISTzn/MOSmawxf7uBbe
mowBa2etGicYBjsMYzGrvNOCLiGznb9AJzD4EICCAbsRJCjHQILtyKPERcxius9wc
0wiIV/z13F2u90EJN5gtIMCdqXb9a2loncdYAUu52Fr0MEs6qyWlzj2KUNz1bBvht+
CkGhW8NJ10Bfrs4EPevI/q 2019010917._domainkey IN TXT (
dEKVz4WSYIRNqLNPFezXjw "v=DKIM1\059 h=sha256\059 k=rsa\059 s=email\059 p="
fEKZ8bjBPiT1Q==
```

```
"MIIBIjANBgkqhkiG9w0BAQEAAQCAQ8AMII1BCgKCQAEEA6N+Xk5S5yT9WNMgbIS7CvNKdW
FRpSR7Tfo6trVOM1606BHsPiSp5U" "kbq/vrK/xgx5c4k5BIOk/yL/jd/O/BqjTGdnc/
p1895L1Ne52+vW1h4FEw9gmwk3etecUP0CYZzs5PgvdLBPgfwyitryjy+pYlxsFBORXZPlr
pQRFnNypSR/eAXWF3REli07NquSSec985dpbZWQ/3MHm" "W8ZVvv5oDfh/kMQ9727qMxp
OED02Qxam+d8f3YHQiimP71R0pd2oocH5gl2LX7gc6MwIDAQAB"
_adsp._domainkey IN TXT "dkim=all"
```

28

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script



Mail - DKIM & ADSP

Berner
Fachhochschule

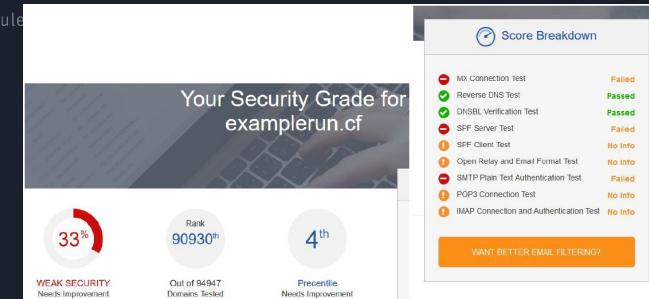
```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/simple; d=examplerun.cf;
s=2019010811; t=1546968309; bh=WN9jjFu//8qtE8luk5bNjWz8eu6jn90bWUEPF3q5DU=;
h=Subject:To:Date:From:From;
b=MSc9anWhENzauKUPTtltlgHXH2vIg9InVCPahOb7
STzn/MOSmawxf7uBbe
mowBaVZetGicYBJsMXzGrvNOCLlGj2nb9AJzD4EI'
djhQILtyKPErcxiu5wc
OwiIV/Z13F2u9OEJN5gtIMCdqXb9a2lonecdYAO'
ds6qyWlzj2KUNz1bBvht+
CkGhWBNJ10BFrs4EPevi/q 2019010917.d
dEKVz4WSVIRNqLNPFezXjW "v=" "56\059 k=rsa\059 s=email\059 p="
w0BAQFEAAOCQAQ8AMIIIBCgKCAQEAE6N+Xk5S5yT9WNMgbIS7CvNRdW
06BhsFisp5U5" "kbQ/vrR/xgx9c4k5BIok/yL/d/o/BqjTGDnC/
4FEw9gmwk3etscUP0CYZZs5PgvrDlBPgfWyyirjt+pYlxsfBORXZPlr
pQRFnNYpl
/F3REli07NquSSec985dpbZWQ/3Mhm" "W8ZVwv5oDfh/kMQ9727gMxp
OED0ZQym14
HK87Rg9zGOJDJs880RC31sd+6tukf7fYyj51TvpRtndLPributKdFgi3e
MMDkQXam+d81sYHQoiMF7lR0pD2oOcH5g1ELX7gc6MwIDAQAB"
_adsp._domainkey IN TXT "dkim=all"
```

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script

29

Mail - Postfix

Berner
Fachhochschule



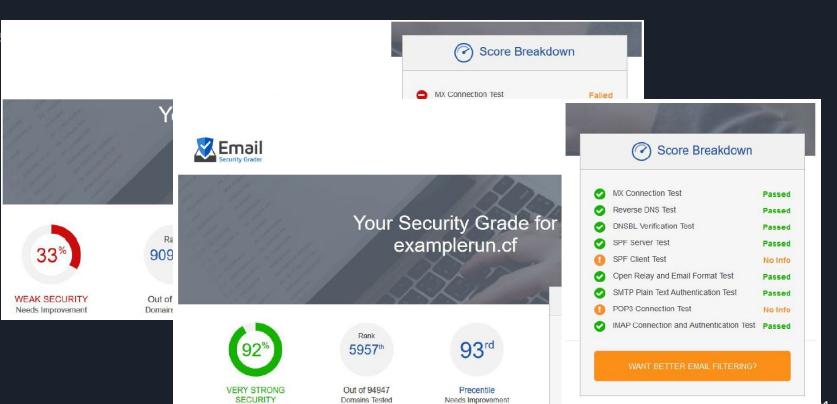
Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script

30



Mail - Postfix

Berner
Fachhochschule



Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script

1

Challenges

Berner
Fachhochschule

Major Event

- From Federated Project to Consolidated Project

Setup

- Shared responsibility common entry point

KISS (Keep It Simple and Stupid)

- Use your own knowledge to not to use it

Fridolin Zurlinden - Jan Henzi - Ismael Riedo | Mail - Server Set-Up & Security-Hardening Script

32

DEMO

All documents and code can be found at :

<https://github.com/ifrido/BTI7301>

END



Web - Optional Part

src/web/apache/
enableApache.sh
configureApache.sh

src/web/nginx/
enableNginx.sh
nginxCertConfig.sh
configureNginx.sh

All documents and code can be found at :

<https://github.com/ifrido/BTI7301>

END



9 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_\(software\)](https://en.wikipedia.org/wiki/Ansible_(software)). 48

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. 8

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. 47

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. 8, 25

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>. 8, 25

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. 8, 15

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. 47

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_\(software\)](https://en.wikipedia.org/wiki/Docker_(software)). 48

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/>. 16



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. 27

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>. 31

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>. 47

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>. 8

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>. 8

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_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)). 48

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_Protocol. 25

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. 8, 25

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. 21

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. 27



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. 25

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_\(anonymity_network\)](https://en.wikipedia.org/wiki/Tor_(anonymity_network)). 47

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>. 12

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_\(DNS_server\)](https://en.wikipedia.org/wiki/Unbound_(DNS_server)). 8

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. 31, 48

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>. 47

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. 17



Bibliography

- [1] Global Cyber Alliance. *DMARC Setup*. 2016. URL: <https://dmarcguide.globalcyberalliance.org> (visited on 10/24/2018).
- [2] Kyle Rankin. *Linux Hardening in Hostile Networks: Server Security from TLS to Tor*. Addison Wesley, 2017.
- [3] 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).
- [4] 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).



Appendices



```
configFiles/dns/unbound/unbound.conf
```

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

```
configFiles/dns/unbound/access.conf
```

```
1 server:
```

```
configFiles/dns/unbound/hardening.conf
```

```
1     ### SOURCE: https://calomel.org/unbound_dns.html ###
2 server:
3     # enable to not answer id.server and hostname.bind queries.
4     hide-identity: yes
5
6     # enable to not answer version.server and version.bind queries.
7     hide-version: yes
8
9     # Read the root hints from this file. Default is nothing, using built in
10    # hints for the IN class. The file has the format of zone files, with root
11    # nameserver names and addresses only. The default may become outdated,
12    # when servers change, therefore it is good practice to use a root-hints
13    # file. get one from https://www.internic.net/domain/named.root
14    root-hints: "/var/lib/unbound/root.hints"
15
16    # Will trust glue only if it is within the servers authority.
17    # Harden against out of zone rrsets, to avoid spoofing attempts.
18    # Hardening queries multiple name servers for the same data to make
19    # spoofing significantly harder and does not mandate dnssec.
20    harden-glue: yes
21
22    # Require DNSSEC data for trust-anchored zones, if such data is absent, the
23    # zone becomes bogus. Harden against receiving dnssec-stripped data. If you
24    # turn it off, failing to validate dnskey data for a trustanchor will trigger
25    # insecure mode for that zone (like without a trustanchor). Default on,
26    # which insists on dnssec data for trust-anchored zones.
27    harden-dnssec-stripped: yes
28
29    # Use 0x20-encoded random bits in the query to foil spoof attempts.
30    # http://tools.ietf.org/html/draft-vixie-dnsext-dns0x20-00
31    # While upper and lower case letters are allowed in domain names, no significance
32    # is attached to the case. That is, two names with the same spelling but
33    # different case are to be treated as if identical. This means calomel.org is the
34    # same as CaLoMeL.Org which is the same as CALOMEL.ORG.
35    use-caps-for-id: yes
36
37    # the time to live (TTL) value lower bound, in seconds. Default 0.
38    # If more than an hour could easily give trouble due to stale data.
39    cache-min-ttl: 3600
40
41    # the time to live (TTL) value cap for RRsets and messages in the
42    # cache. Items are not cached for longer. In seconds.
43    cache-max-ttl: 86400
44
45    # perform prefetching of close to expired message cache entries. If a client
46    # requests the dns lookup and the TTL of the cached hostname is going to
47    # expire in less than 10% of its TTL, unbound will (1st) return the ip of the
48    # host to the client and (2nd) pre-fetch the dns request from the remote dns
49    # server. This method has been shown to increase the amount of cached hits by
50    # local clients by 10% on average.
51    prefetch: yes
52
53    # number of threads to create. 1 disables threading. This should equal the number
54    # of CPU cores in the machine. Our example machine has 4 CPU cores.
55    num-threads: 1
56
57    ## 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 cached 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

```



```

4 # QTYPE to NS when possible.
5
6 # See RFC 7816 "DNS Query Name Minimisation to Improve Privacy" for
7 # details.
8
9 qname-minimisation: yes

```

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

```

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

```

configFiles/dns/nsd/nsd.conf

```

1
2 server:
3   # uncomment to specify specific interfaces to bind (default all).
4   ip-address: 104.248.137.212
5   #ip-address:
6
7   # port to answer queries on. default is 53.
8   port: 53
9
10  # Number of NSD servers to fork.
11  server-count: 1
12
13  # listen only on IPv4 connections
14  ip4-only: yes
15
16  # don't answer VERSION.BIND and VERSION.SERVER CHAOS class queries
17  hide-version: yes
18
19  # identify the server (CH TXT ID.SERVER entry).
20  identity: ""
21
22  logfile: "/var/log/nsd.log"
23
24 # The directory for zonefile: files.
25  zonesdir: "/etc/nsd/zones"
26  pidfile: "/etc/nsd/nsd.pid"
27  username: nsd
28
29 pattern:
30   name: exemplерун.cf
31   zonefile: exemplерун.cf.forward
32 pattern:
33   name: 212.137.248.104.in-addr.arpa
34   zonefile: exemplерун.cf.backward

```

configFiles/dns/nsd/exemplерун.cf.backward

```

1
2 $ORIGIN 212.137.248.104.in-addr.arpa.
3 $TTL 1800
4
5 @ IN SOA ns1.exemplерун.cf. ns2.exemplерун.cf. (
6   2019010917 ; serial number
7   28800       ; Refresh
8   7200        ; Retry
9   1209600     ; Expire
10  86400       ; Min TTL
11 )
12
13      NS      ns1.exemplерун.cf.

```



```

14      NS      ns2.examplerun.cf.
15 ; PTR
16          IN      PTR      examplerun.cf.
17          IN      PTR      mail.examplerun.cf.

```

configFiles/dns/nsd/examplerun.cf.forward

```

1 $ORIGIN exempleren.cf.      ; default zone domain
2 $TTL    86400                ; default time to live
3
4
5
6 @ IN SOA ns1.examplerun.cf. ns2.examplerun.cf. (
7     2019010917      ; serial number
8     28800           ; Refresh
9     7200            ; Retry
10    1209600         ; Expire
11    86400           ; Min TTL
12 )
13
14      NS      ns1.examplerun.cf.
15      NS      ns2.examplerun.cf.
16      MX      10 mail.examplerun.cf.
17
18 exempleren.cf. IN CAA 0 issue "letsencrypt.org"
19 exempleren.cf. IN CAA 0 iodef "mailto:postmaster@examplerun.cf"
20
21      IN A  104.248.137.212
22      IN TXT "v=spf1 mx a ~all"
23 ns1  IN A  104.248.137.212
24 ns2  IN A  104.248.137.212
25 www  IN A  104.248.137.212
26 *    IN A  104.248.137.212
27
28 mail IN A 104.248.137.212
29      IN TXT "v=spf1 mx a ~all"
30 2019010917._domainkey IN TXT (
31 "v=DKIM1\059 h=sha256\059 k=rsa\059 s=email\059 p="
32 "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIIBCgKCAQEA6N+
   Xk5S5yT9WNMgbIS7CvNKdWFKpSR7Tfo6trVOM1606BHsFiSp5U5"
33 "kbQ/vrK/xgx9c4k5BI0k/yL/jd/O/BqjTGDNc/pL89SL1Ne5Z+
   vW1h4FEw9gmwk3etscUPOCYZs5PgvD1BPgfWyitrjy+pYlxsfB0RXZPlrpQRFnNYpSR/
   eAXWF3REli07NquSSec985dpbZWQ/3MHm"
34 "W8ZVvw5oDfh/kMQ9727qMxp0ED0ZQym12kPpdHK87Rg9zGOJDJs880RC3lsd+6
   tukf7fYyj51TvpRtnLPrbuKdFgi3eMMDkQXam+d8f3YHQoiMF71R0pD2o0cH5g1ELX7gc6MwIDAQAB"
   )
35 _adsp._domainkey IN TXT "dkim=all"
36 _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

```

1 # $OpenBSD: sshd_config,v 1.101 2017/03/14 07:19:07 djm Exp $
2
3 # This is the sshd server system-wide configuration file. See
4 # sshd_config(5) for more information.
5
6 # This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin
7
8 # The strategy used for options in the default sshd_config shipped with
9 # OpenSSH is to specify options with their default value where
10 # possible, but leave them commented. Uncommented options override the
11 # default value.
12
13 #Port 22
14 #AddressFamily any

```



```
15 #ListenAddress 0.0.0.0
16 #ListenAddress ::

17

18 #HostKey /etc/ssh/ssh_host_rsa_key
19 #HostKey /etc/ssh/ssh_host_ecdsa_key
20 #HostKey /etc/ssh/ssh_host_ed25519_key
21

22 # Ciphers and keying
23 #RekeyLimit default none
24

25 # Logging
26 #SyslogFacility AUTH
27 #LogLevel INFO
28

29 # Authentication:
30

31 #LoginGraceTime 2m
32 PermitRootLogin no
33 #StrictModes yes
34 #MaxAuthTries 6
35 #MaxSessions 10
36

37 #PubkeyAuthentication yes
38

39 # Expect .ssh/authorized_keys2 to be disregarded by default in future.
40 #AuthorizedKeysFile      .ssh/authorized_keys .ssh/authorized_keys2
41

42 #AuthorizedPrincipalsFile none
43

44 #AuthorizedKeysCommand none
45 #AuthorizedKeysCommandUser nobody
46

47 # For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
48 #HostbasedAuthentication no
49 # Change to yes if you don't trust ~/.ssh/known_hosts for
50 # HostbasedAuthentication
51 #IgnoreUserKnownHosts no
52 # Don't read the user's ~/.rhosts and ~/.shosts files
53 #IgnoreRhosts yes
54

55 # To disable tunneled clear text passwords, change to no here!
56 PasswordAuthentication no
57 #PermitEmptyPasswords no
58

59 # Change to yes to enable challenge-response passwords (beware issues with
60 # some PAM modules and threads)
61 ChallengeResponseAuthentication no
62

63 # Kerberos options
64 #KerberosAuthentication no
65 #KerberosOrLocalPasswd yes
66 #KerberosTicketCleanup yes
67 #KerberosGetAFSToken no
68

69 # GSSAPI options
70 #GSSAPIAuthentication no
71 #GSSAPICleanupCredentials yes
72 #GSSAPIStrictAcceptorCheck yes
73 #GSSAPIKeyExchange no
74

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



```

81 # If you just want the PAM account and session checks to run without
82 # PAM authentication, then enable this but set PasswordAuthentication
83 # and ChallengeResponseAuthentication to 'no'.
84 UsePAM yes
85
86 #AllowAgentForwarding yes
87 #AllowTcpForwarding yes
88 #GatewayPorts no
89 X11Forwarding no
90 #X11DisplayOffset 10
91 #X11UseLocalhost yes
92 #PermitTTY yes
93 PrintMotd no
94 #PrintLastLog yes
95 #TCPKeepAlive yes
96 #UseLogin no
97 #PermitUserEnvironment no
98 #Compression delayed
99 #ClientAliveInterval 0
100 #ClientAliveCountMax 3
101 #UseDNS no
102 #PidFile /var/run/sshd.pid
103 #MaxStartups 10:30:100
104 #PermitTunnel no
105 #ChrootDirectory none
106 #VersionAddendum none
107
108 # no default banner path
109 #Banner none
110
111 # Allow client to pass locale environment variables
112 AcceptEnv LANG LC_*
113
114 # override default of no subsystems
115 Subsystem sftp /usr/lib/openssh/sftp-server
116
117 # Example of overriding settings on a per-user basis
118 #Match User anoncvs
119 #     X11Forwarding no
120 #     AllowTcpForwarding no
121 #     PermitTTY no
122 #     ForceCommand cvs server
123 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
124
125 KexAlgorithms curve25519-sha256@libssh.org,ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-
    sha2-nistp256,diffie-hellman-group-exchange-sha256
126 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
127 Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@openssh.com,
    aes256-ctr,aes192-ctr,aes128-ctr

```

configFiles/mail/main.cf

```

1 # See /usr/share/postfix/main.cf.dist for a commented, more complete version
2
3
4 # Debian specific: Specifying a file name will cause the first
5 # line of that file to be used as the name. The Debian default
6 # is /etc/mailname.
7 #myorigin = /etc/mailname
8
9 smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)
10 biff = no
11

```



```

12 # appending .domain is the MUA's job.
13 append_dot_mydomain = no
14
15 # Uncomment the next line to generate "delayed mail" warnings
16 #delay_warning_time = 4h
17
18 readme_directory = no
19
20 # See http://www.postfix.org/COMPATIBILITY_README.html -- default to 2 on
21 # fresh installs.
22 compatibility_level = 2
23
24 # TLS parameters
25 smtpd_tls_cert_file = /etc/letsencrypt/live/mail.examplerun.cf/fullchain.pem
26 smtpd_tls_key_file = /etc/letsencrypt/live/mail.examplerun.cf/privkey.pem
27 smtpd_use_tls = yes
28 smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
29 smtp_tls_session_cache_database = btree:/var/lib/postfix/smtp_tls_session_cache
30
31 # See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package for
32 # information on enabling SSL in the smtp client.
33
34 smtpd_relay_restrictions = permit_mynetworks, permit_sasl_authenticated,
   permit_tls_all_clientcerts, reject_unauth_destination
35 myhostname = mail.examplerun.cf
36 alias_maps = hash:/etc/aliases
37 alias_database = hash:/etc/aliases
38 myorigin = /etc/mailname
39 mydestination = $myhostname, $mydomain, mail.examplerun.cf, localhost.examplerun.cf,
   localhost
40 relayhost =
41 mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
42 mailbox_size_limit = 0
43 recipient_delimiter = +
44 inet_interfaces = all
45 inet_protocols = all
46 sender_canonical_maps = hash:/etc/postfix/canonical
47 mydomain = examplerun.cf
48 smtp_tls_security_level = may
49 smtp_tls_note_starttls_offer = yes
50 smtp_tls_loglevel = 1
51 ###https://access.redhat.com/articles/1468593
52 smtpd_tls_mandatory_protocols = !SSLv2, !SSLv3, !TLSv1, !TLSv1.1
53 smtpd_tls_protocols = !SSLv2, !SSLv3, !TLSv1, !TLSv1.1
54 smtpd_tls_mandatory_protocols = !SSLv2, !SSLv3, !TLSv1, !TLSv1.1
55 smtpd_tls_protocols = !SSLv2, !SSLv3, !TLSv1, !TLSv1.1
56 smtpd_tls_exclude_ciphers = EXP, MEDIUM, LOW, DES, 3DES, SSLv2
57 smtpd_tls_exclude_ciphers = EXP, MEDIUM, LOW, DES, 3DES, SSLv2
58 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
59 tls_preempt_cipherlist = yes
60 smtpd_tls_ciphers = high
61 smtpd_tls_ciphers = high
62 policyd-spf_time_limit = 3600
63 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
64 #START OpenDKIM & OpenDMARC
65 milter_protocol = 6
66 milter_default_action = accept
67 smtpd_milters = local:/opendkim/opendkim.sock, local:/opendmarc/opendmarc.sock
68 non_smtpd_milters = local:/opendkim/opendkim.sock, local:/opendmarc/opendmarc.sock
69 #END OpenDKIM & OpenDMARC
70 smtpd_tls_CAfile = /etc/ssl/certs/examplerun.cf.ca.crl.pem

```



```
71| tls_append_default_CA = no
```

configFiles/mail/master.cf

```

1#
2# Postfix master process configuration file. For details on the format
3# of the file, see the master(5) manual page (command: "man 5 master" or
4# on-line: http://www.postfix.org/master.5.html).
5#
6# Do not forget to execute "postfix reload" after editing this file.
7#
8# =====
9# service type  private unpriv  chroot  wakeup  maxproc command + args
10#           (yes)   (yes)   (no)    (never) (100)
11# =====
12smtp      inet  n      -      y      -      -      smtpd
13#smtp      inet  n      -      y      -      1      postscreen
14#smtpd     pass  -      -      y      -      -      smtpd
15#dnsblog   unix  -      -      y      -      0      dnsblog
16#tlsproxy  unix  -      -      y      -      0      tlsproxy
17#submission inet n      -      y      -      -      smtpd
18# -o syslog_name=postfix/submission
19# -o smtpd_tls_security_level=encrypt
20# -o smtpd_sasl_auth_enable=yes
21# -o smtpd_tls_ask_ccert=yes
22# -o smtpd_tls_security_level=encrypt
23# -o smtpd_sasl_auth_enable=yes
24# -o smtpd_tls_auth_only=yes
25# -o smtpd_reject_unlisted_recipient=no
26# -o smtpd_client_restrictions=$mua_client_restrictions
27# -o smtpd_helo_restrictions=$mua_helo_restrictions
28# -o smtpd_sender_restrictions=$mua_sender_restrictions
29# -o smtpd_recipient_restrictions=
30# -o smtpd_relay_restrictions=permit_sasl_authenticated,reject
31# -o milter_macro_daemon_name=ORIGINATING
32#smtps     inet  n      -      y      -      -      smtpd
33# -o syslog_name=postfix/smtps
34# -o smtpd_tls_wrappermode=yes
35# -o smtpd_sasl_auth_enable=yes
36# -o smtpd_reject_unlisted_recipient=no
37# -o smtpd_client_restrictions=$mua_client_restrictions
38# -o smtpd_helo_restrictions=$mua_helo_restrictions
39# -o smtpd_sender_restrictions=$mua_sender_restrictions
40# -o smtpd_recipient_restrictions=
41# -o smtpd_relay_restrictions=permit_sasl_authenticated,reject
42# -o milter_macro_daemon_name=ORIGINATING
43#628      inet  n      -      y      -      -      qmqpd
44pickup    unix  n      -      y      60      1      pickup
45cleanup   unix  n      -      y      -      0      cleanup
46qmgr      unix  n      -      n      300      1      qmgr
47#qmgr     unix  n      -      n      300      1      oqmgr
48tlsmgrp  unix  -      -      y      1000?    1      tlsmgrp
49rewrite   unix  -      -      y      -      -      trivial-rewrite
50bounce    unix  -      -      y      -      0      bounce
51defer     unix  -      -      y      -      0      bounce
52trace     unix  -      -      y      -      0      bounce
53verify    unix  -      -      y      -      1      verify
54flush     unix  n      -      y      1000?    0      flush
55proxymap  unix  -      -      n      -      -      proxymap
56proxywrite unix  -      -      n      -      1      proxymap
57smtp      unix  -      -      y      -      -      smtp
58relay     unix  -      -      y      -      -      smtp
59#          -o syslog_name=postfix/$service_name
60#          -o smtp_helo_timeout=5 -o smtp_connect_timeout=5
61showq     unix  n      -      y      -      -      showq
62error     unix  -      -      y      -      -      error
63retry     unix  -      -      y      -      -      error

```



```

64 discard unix - - y - - discard
65 local unix - n n - - local
66 virtual unix - n n - - virtual
67 lmtp unix - - y - - lmtp
68 anvil unix - - y - 1 anvil
69 scache unix - - y - 1 scache
70 #
71 # =====
72 # Interfaces to non-Postfix software. Be sure to examine the manual
73 # pages of the non-Postfix software to find out what options it wants.
74 #
75 # Many of the following services use the Postfix pipe(8) delivery
76 # agent. See the pipe(8) man page for information about ${recipient}
77 # and other message envelope options.
78 # =====
79 #
80 # maildrop. See the Postfix MAILDROP_README file for details.
81 # Also specify in main.cf: maildrop_destination_recipient_limit=1
82 #
83 maildrop unix - n n - - pipe
84 flags=DRhu user=vmail argv=/usr/bin/maildrop -d ${recipient}
85 #
86 # =====
87 #
88 # Recent Cyrus versions can use the existing "lmtp" master.cf entry.
89 #
90 # Specify in cyrus.conf:
91 # lmtp cmd="lmtpd -a" listen="localhost:lmtp" proto=tcp4
92 #
93 # Specify in main.cf one or more of the following:
94 # mailbox_transport = lmtp/inet:localhost
95 # virtual_transport = lmtp/inet:localhost
96 #
97 # =====
98 #
99 # Cyrus 2.1.5 (Amos Gouaux)
100 # Also specify in main.cf: cyrus_destination_recipient_limit=1
101 #
102 #cyrus unix - n n - - pipe
103 # user=cyrus argv=/cyrus/bin/deliver -e -r ${sender} -m ${extension} ${user}
104 #
105 # =====
106 # Old example of delivery via Cyrus.
107 #
108 #old-cyrus unix - n n - - pipe
109 # flags=R user=cyrus argv=/cyrus/bin/deliver -e -m ${extension} ${user}
110 #
111 # =====
112 #
113 # See the Postfix UUCP_README file for configuration details.
114 #
115 uucp unix - n n - - pipe
116 flags=Fqhu user=uucp argv=uux -r -n -z -a$sender - $nexthop!rmail ($recipient)
117 #
118 # Other external delivery methods.
119 #
120 ifmail unix - n n - - pipe
121 flags=F user=ftn argv=/usr/lib/ifmail/ifmail -r $nexthop ($recipient)
122 bsmtp unix - n n - - pipe
123 flags=Fq. user=bsmtp argv=/usr/lib/bsmtp/bsmtp -t$nexthop -f$sender $recipient
124 scalemail-backend unix - n n - 2 pipe
125 flags=R user=scalemail argv=/usr/lib/scalemail/bin/scalemail-store ${nexthop} ${user}
     } ${extension}
126 mailman unix - n n - - pipe
127 flags=FR user=list argv=/usr/lib/mailman/bin/postfix-to-mailman.py
     ${nexthop} ${user}
128

```



```

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

```

configFiles/mail/canonical.conf

```

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

```

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

```



```
49 ## evaluation of the message. Instead, an Authentication-Results header
50 ## field will be added.
51 #
52 # RejectFailures false
53 #
54 ## Socket socketspec
55 ##     default (none)
56 ##
57 ## Specifies the socket that should be established by the filter to receive
58 ## connections from sendmail(8) in order to provide service. socketspec is
59 ## in one of two forms: local:path, which creates a UNIX domain socket at
60 ## the specified path, or inet:port[@host] or inet6:port[@host] which creates
61 ## a TCP socket on the specified port for the appropriate protocol family.
62 ## If the host is not given as either a hostname or an IP address, the
63 ## socket will be listening on all interfaces. This option is mandatory
64 ## either in the configuration file or on the command line. If an IP
65 ## address is used, it must be enclosed in square brackets.
66 #
67 Socket local:/var/run/opendmarc/opendmarc.sock
68 #
69 ## Syslog { true | false }
70 ##     default "false"
71 ##
72 ## Log via calls to syslog(3) any interesting activity.
73 #
74 Syslog true
75 #
76 ## SyslogFacility facility-name
77 ##     default "mail"
78 ##
79 ## Log via calls to syslog(3) using the named facility. The facility names
80 ## are the same as the ones allowed in syslog.conf(5).
81 #
82 # SyslogFacility mail
83 #
84 ## TrustedAuthservIDs string
85 ##     default HOSTNAME
86 ##
87 ## Specifies one or more "authserv-id" values to trust as relaying true
88 ## upstream DKIM and SPF results. The default is to use the name of
89 ## the MTA processing the message. To specify a list, separate each entry
90 ## with a comma. The key word "HOSTNAME" will be replaced by the name of
91 ## the host running the filter as reported by the gethostname(3) function.
92 #
93 # TrustedAuthservIDs HOSTNAME
94 #
95 ## UMask mask
96 ##     default (none)
97 ##
98 ## Requests a specific permissions mask to be used for file creation. This
99 ## only really applies to creation of the socket when Socket specifies a
100 ## UNIX domain socket, and to the HistoryFile and PidFile (if any); temporary
101 ## files are normally created by the mkstemp(3) function that enforces a
102 ## specific file mode on creation regardless of the process umask. See
103 ## umask(2) for more information.
104 #
105 UMask 0002
106 #
107 ## UserID user[:group]
108 ##     default (none)
109 ##
110 ## Attempts to become the specified userid before starting operations.
111 ## The process will be assigned all of the groups and primary group ID of
112 ## the named userid unless an alternate group is specified.
113 #
114 UserID opendmarc
```



```

115 AutoRestart Yes
116 AutoRestartRate 10/1h
117 PidFile /var/spool/postfix/opendmarc/opendmarc.pid
118 Socket local:/var/spool/postfix/opendmarc/opendmarc.sock
119 AuthservID mail.examplerun.cf
120 TrustedAuthservIDs mail.examplerun.cf
121 Syslog true
122 SyslogFacility mail
123 UMask 0002
124 UserID opendmarc:opendmarc

```

configFiles/mail/opendkim.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/opendkim/examples/opendkim.conf.sample.
4
5 # Log to syslog
6 Syslog yes
7 # Required to use local socket with MTAs that access the socket as a non-
8 # privileged user (e.g. Postfix)
9 UMask 007
10
11 # Sign for example.com with key in /etc/dkimkeys/dkim.key using
12 # selector '2007' (e.g. 2007._domainkey.example.com)
13 #Domain example.com
14 #KeyFile /etc/dkimkeys/dkim.key
15 #Selector 2007
16
17 # Commonly-used options; the commented-out versions show the defaults.
18 #Canonicalization simple
19 #Mode sv
20 #SubDomains no
21
22 # Socket smtp://localhost
23 #
24 # ## Socket socketspec
25 # ##
26 # ## Names the socket where this filter should listen for milter connections
27 # ## from the MTA. Required. Should be in one of these forms:
28 # ##
29 # ## inet:port@address to listen on a specific interface
30 # ## inet:port to listen on all interfaces
31 # ## local:/path/to/socket to listen on a UNIX domain socket
32 #
33 #Socket inet:8892@localhost
34 Socket local:/var/run/opendkim/opendkim.sock
35
36 ## PidFile filename
37 ### default (none)
38 ###
39 ### Name of the file where the filter should write its pid before beginning
40 ### normal operations.
41 #
42 PidFile /var/run/opendkim/opendkim.pid
43
44
45 # Always oversign From (sign using actual From and a null From to prevent
46 # malicious signatures header fields (From and/or others) between the signer
47 # and the verifier. From is oversigned by default in the Debian pacakge
48 # because it is often the identity key used by reputation systems and thus
49 # somewhat security sensitive.
50 OversignHeaders From
51
52 ## ResolverConfiguration filename
53 ## default (none)
54 ##

```



```

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 ## Authentication processes
3 ##
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 ## SSL settings
3 ##
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

```



```

11 allow - tcp - 465
12 allow - udp - 465
13 # IMAP
14 allow - tcp - 143
15 allow - udp - 143
16 # IMAP TLS
17 allow - tcp - 993
18 allow - udp - 993
19 # HTTP HTTPS
20 allow - tcp - 80
21 allow - tcp - 443

```

configFiles/web/nginx/nginx.conf

```

1 user www-data;
2 worker_processes auto;
3 pid /run/nginx.pid;
4 include /etc/nginx/modules-enabled/*.conf;
5
6 events {
7     worker_connections 768;
8     # multi_accept on;
9 }
10
11 http {
12     ##
13     # Basic Settings
14     ##
15
16     sendfile on;
17     tcp_nopush on;
18     tcp_nodelay on;
19     keepalive_timeout 65;
20     types_hash_max_size 2048;
21     server_tokens off;
22
23     # server_names_hash_bucket_size 64;
24     # server_name_in_redirect off;
25
26     include /etc/nginx/mime.types;
27     default_type application/octet-stream;
28
29     ##
30     # Logging Settings
31     ##
32
33     access_log /var/log/nginx/access.log;
34     error_log /var/log/nginx/error.log;
35
36     ##
37     # Gzip Settings
38     ##
39
40     gzip on;
41
42     # gzip_vary on;
43     # gzip_proxied any;
44     # gzip_comp_level 6;
45     # gzip_buffers 16 8k;
46     # gzip_http_version 1.1;
47     # gzip_types text/plain text/css application/json application/javascript text/xml
48     #           application/xml application/xml+rss text/javascript;
49
50     ##
51     # Virtual Host Configs
52     ##

```



```

53     include /etc/nginx/conf.d/*.conf;
54 }
```

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

```

1 server {
2     listen 443 ssl;
3     listen [::]:443 ssl;
4     server_name exemplерун.cf www.exemplерун.cf default_server;
5
6     ssl_prefer_server_ciphers on;
7     ssl_protocols TLSv1.1 TLSv1.2;
8     ssl_ciphers ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
9         ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM
10        -SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-
11        AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256;
12
13     ssl_session_cache shared:SSL:50m;
14     ssl_session_timeout 5m;
15
16     ssl_certificate /etc/letsencrypt/live/examplerun.cf/fullchain.pem; # managed
17         by Certbot
18     ssl_certificate_key /etc/letsencrypt/live/examplerun.cf/privkey.pem; # managed
19         by Certbot
20
21     ssl_dhparam /etc/ssl/dh4096.pem;
22
23     add_header Strict-Transport-Security "max-age=31536000; includeSubDomains"
24         always;
25     add_header X-Content-Type-Options "nosniff" always;
26     add_header X-Xss-Protection "1; mode=block" always;
27     add_header X-Frame-Options "SAMEORIGIN" always;
28     add_header Referrer-Policy "same-origin" always;
29
30     access_log /var/log/nginx/examplerun.cf_ssl_access.log;
31     error_log /var/log/nginx/examplerun.cf_ssl_error.log;
32
33     location / {
34         proxy_set_header X-Real-IP      $remote_addr;
35         proxy_set_header X-Forwarded-For $remote_addr;
36         proxy_set_header Host          $host;
37         proxy_pass http://127.0.0.1:8080;
38     }
39 }
40
41 server {
42     listen 80;
43     listen [::]:80;
44     server_name exemplерун.cf www.exemplерун.cf default_server;
45
46     access_log /var/log/nginx/examplerun.cf_access.log;
47     error_log /var/log/nginx/examplerun.cf_error.log;
48
49     return 301 https://$host$request_uri;
50 }
```

configFiles/web/apache2/ports.conf

```

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

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

```

1 <VirtualHost 127.0.0.1:8080>
2     ServerName exemplерун.cf
3 
```



```
4  ServerName www.examplerun.cf
5  ServerAdmin webmaster@examplerun.cf
6  DocumentRoot /var/www/examplerun
7
8  #LogLevel info ssl:warn
9
10 ErrorLog ${APACHE_LOG_DIR}/ismu.ga_error.log
11 CustomLog ${APACHE_LOG_DIR}/ismu.ga_access.log combined
12 </VirtualHost>
13
14 # vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```