Haaga-Helia
University of Applied Sciences

Linux hardening with configuration management software

Arch Linux and SaltStack

Tuomo Kuure

Practical work

Tietoturvan hallinta

ICT4TN026

17th May 2020

Table of contents

1 Introduction

Hardening in a computing context means securing a system by minimizing it's surface of vulnerabitility. General purpose systems often come installed with various software packages many of which are often completely unnecessary, or even harmful. Hardening also means introducing firewalls, intrusion detection systems (IDS) and advanced access control mechanisms.

The objective of this work is to construct a hardened Linux server in a cloud platform. The hardening is done and maintained with configuration management software. The initial installation process is also covered. A learning objective is to familiarize the author in hardening processes and different methods of implementing security controls in Linux systems.

This work is technical in it's nature, and the reader is expected to have familiarity with Linux systems.

2 Tools and methods

2.1 Arch Linux

Arch Linux was chosen as the Linux distribution, as it offers a minimalistic environment to start with while being a general purpose operating system that can be extended according to users needs. Additionally Arch is a rolling-release Linux distribution, which means updates to packages are continuously delivered. One telltale of this is that Arch Linux doesn't supply a version number. The package management offers always up to date software, libraries and the Linux kernel itself.

https://wiki.archlinux.org/index.php/Arch_Linux

2.2 Cloud environment

Two cloud environments are used in this work, the first one being finnish UpCloud where a command and control server is estabilished with a public IP address. Second cloud environmet is Haaga-Helia University of Applied Science's Rovius Cloud Platform, which is behind a NAT firewall. In Rovius CP, a small cluster of virtual machines is installed, that are controlled and hardened by the master server.

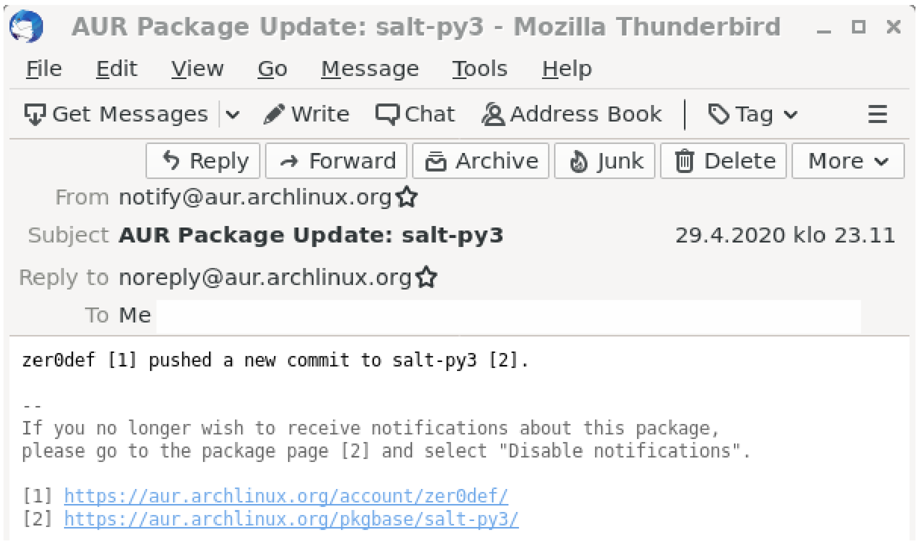
2.3 Configuration management software: SaltStack

SaltStack is an open-source configuration management software. It is suitable for managing large infrastructure deployments. The terminology involves a 'salt-master' as a remote execution server, and respectively 'salt-minions' are the machines controlled with full system administration capability. As with many other similar software, mutual authentication via a public key infrastructure and encrypted communications are included.

The communication SaltStack's Salt uses is a publish-subscribe pattern. This means that the minions are the active parts in the communication by listening to the master's publications. The master publishes it's commands and minions listen to them and make actions accordingly after which they report to the master server of the results. From a security standpoint, no ports have to be open for incoming connections on the minions. Only master needs 2 ports opened to which the minions can connect to.

Two recent vulnerabilities, CVE-2020-11651 and CVE-2020-11652, were discovered from SaltStack by F-secure, where an attacker can bypass all authentication controls and gain full remote execution as root. The vulnerabilities have been patched, and it is considered best practice to not expose the ports Salt uses to the public Internet. This has been taken into account in this work (see 4.7 Firewall setup), and Arch Linux supplied updates to both python 2 and 3 versions the same day they were published.

As the SaltStack version used in this work is not part of the official Arch repositories, notifications about updates to this package should be obtained. Registering into Arch User Repositories (AUR) enables the ability to receive notifications via email when the package is updated.


AUR Package Update: salt-py3 - Mozilla Thunderbird
File Edit View Go Message Tools Help
Get Messages
Write
Chat
Address Book
Tag
Reply
Forward
Archive
Junk
Delete
More

From notify@aur.archlinux.org
Subject **AUR Package Update: salt-py3** 29.4.2020 klo 23.11
Reply to noreply@aur.archlinux.org
To Me

```
zer0def [1] pushed a new commit to salt-py3 [2].

--
If you no longer wish to receive notifications about this package,
please go to the package page [2] and select "Disable notifications".

[1] https://aur.archlinux.org/account/zer0def/
[2] https://aur.archlinux.org/pkgbase/salt-py3/
```

The email notification of the update concerning the CVE-2020-11651 and CVE-2020-11652 security patches.

https://docs.saltstack.com/en/getstarted/system/communication.html
https://labs.f-secure.com/advisories/saltstack-authorization-bypass

2.4 Hardening checklist

Center for Internet Security (CIS) is a US based non-profit organization who distributes security guidelines, security benchmarks in form of checklists and hardened operating system images for free. Arch Linux is not on their list of provided images, and as the purpose of this work is to study the hardening process, a distribution independent benchmark was chosen to guide this work. It contains about 250 items which are all geared towards hardening a Linux system. The list was made to a spreadsheet, which is included in Appendix 7.

https://www.cisecurity.org/cis-benchmarks

3 Initial Setup

As Arch Linux installation procedure involves some manual work, installation scripts were devised to automate the installation process. Already in this phase the CIS benchmarks were taken into account, and the scripts were modified accordingly. In this chapter the scripts and their items related to the installation process are discussed. This chapter covers partly chapter 1.1 - Filesystem Configuration of the CIS benchmarks. All the scripts used in this work have been published in **'https://github.com/tqre/hardening-arch-linux'.**

3.1 Partitioning schemes

Partitioning recommendations are fine grained in CIS recommendations. They include separating directories /var, /var/tmp, /var/log, /var/log/audit and /home into their own partitions. The reasons behind separating the variable data directory are due to the risk of resource exhaustion. A full root partition can render a system unusable, and if logs are not rotated properly while residing in the root partition, they can be filled due to a misconfiguration or even malicious intent. CIS p.37-69

The partition maps are best to be created on the installation phase. How big should the partitions be depends on the actual use case and resources available. The '/var' itself should be decently sized, and recommendations average on around 10G. Log rotation and package manager cache management has to be considered as well.

A file containing partition scheme is read when the installation script is run, and the partitions are created according to benchmark guidelines. The scripts and the partition schemes are presented in Appendices 1 and 2.

Partition mountpoint	Master	Minion	Type
boot	1M	1M	vfat
/	8G	8G	ext4
/var	16G	8G	ext4
/var/tmp	8G	2G	ext4
/var/log	8G	2G	ext4
/var/log/audit	8G	2G	ext4
/home	2G	10G	ext4
TOTAL	50G	32G	

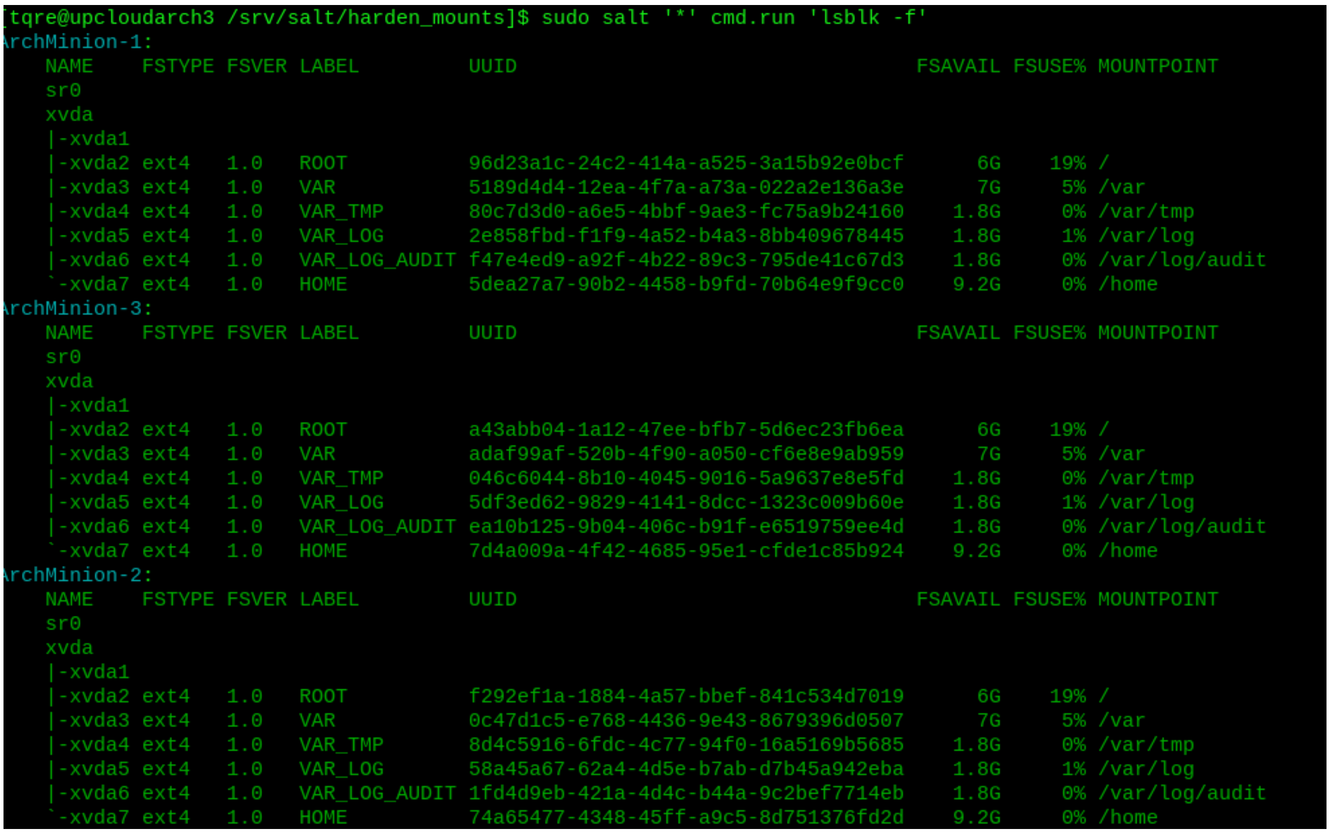
General overview of the partitioning scheme.

3.2 Notes on partition configurations

Arch Linux's installation scripts include 'genfstab', which generates the initial filesystem table '/etc/fstab'. As we are using configuration management software, one problem is to identify device names across all the network machines. Using partition UUID's becomes difficult and unclear with configuration management. The ext4 partitioning allows labels and the following command is used to label the partitions in the partitioning script (Appendix 1).

```
# tune2fs -L <labelname> /dev/<partition>
```

Using genfstab's -L option in the script, we have same labels on all the partitions across all the salt-minions. The configuration management can now easilty to refer for '/home' partitions with a label 'HOME'.



```
[tqre@upcloudarch3 /srv/salt/harden_mounts]$ sudo salt '*' cmd.run 'lsblk -f'
ArchMinion-1:
    NAME    FSTYPE FSVER LABEL         UUID                                 FSAVAIL FSUSE% MOUNTPOINT
    sr0
    xvda
    |-xvda1
    |-xvda2 ext4   1.0   ROOT          96d23a1c-24c2-414a-a525-3a15b92e0bcf      6G    19% /
    |-xvda3 ext4   1.0   VAR           5189d4d4-12ea-4f7a-a73a-022a2e136a3e      7G     5% /var
    |-xvda4 ext4   1.0   VAR_TMP       80c7d3d0-a6e5-4bbf-9ae3-fc75a9b24160    1.8G     0% /var/tmp
    |-xvda5 ext4   1.0   VAR_LOG       2e858fbd-f1f9-4a52-b4a3-8bb409678445    1.8G     1% /var/log
    |-xvda6 ext4   1.0   VAR_LOG_AUDIT f47e4ed9-a92f-4b22-89c3-795de41c67d3    1.8G     0% /var/log/audit
    `-xvda7 ext4   1.0   HOME          5dea27a7-90b2-4458-b9fd-70b64e9f9cc0    9.2G     0% /home
ArchMinion-3:
    NAME    FSTYPE FSVER LABEL         UUID                                 FSAVAIL FSUSE% MOUNTPOINT
    sr0
    xvda
    |-xvda1
    |-xvda2 ext4   1.0   ROOT          a43abb04-1a12-47ee-bfb7-5d6ec23fb6ea      6G    19% /
    |-xvda3 ext4   1.0   VAR           adaf99af-520b-4f90-a050-cf6e8e9ab959      7G     5% /var
    |-xvda4 ext4   1.0   VAR_TMP       046c6044-8b10-4045-9016-5a9637e8e5fd    1.8G     0% /var/tmp
    |-xvda5 ext4   1.0   VAR_LOG       5df3ed62-9829-4141-8dcc-1323c009b60e    1.8G     1% /var/log
    |-xvda6 ext4   1.0   VAR_LOG_AUDIT ea10b125-9b04-406c-b91f-e6519759ee4d    1.8G     0% /var/log/audit
    `-xvda7 ext4   1.0   HOME          7d4a009a-4f42-4685-95e1-cfde1c85b924    9.2G     0% /home
ArchMinion-2:
    NAME    FSTYPE FSVER LABEL         UUID                                 FSAVAIL FSUSE% MOUNTPOINT
    sr0
    xvda
    |-xvda1
    |-xvda2 ext4   1.0   ROOT          f292ef1a-1884-4a57-bbef-841c534d7019      6G    19% /
    |-xvda3 ext4   1.0   VAR           0c47d1c5-e768-4436-9e43-8679396d0507      7G     5% /var
    |-xvda4 ext4   1.0   VAR_TMP       8d4c5916-6fdc-4c77-94f0-16a5169b5685    1.8G     0% /var/tmp
    |-xvda5 ext4   1.0   VAR_LOG       58a45a67-62a4-4d5e-b7ab-d7b45a942eba    1.8G     1% /var/log
    |-xvda6 ext4   1.0   VAR_LOG_AUDIT 1fd4d9eb-421a-4d4c-b44a-9c2bef7714eb    1.8G     0% /var/log/audit
    `-xvda7 ext4   1.0   HOME          74a65477-4348-45ff-a9c5-8d751376fd2d    9.2G     0% /home
```

Screenshot from early testing phase, where SaltStack is used to obtain block device information from three salt-minions.

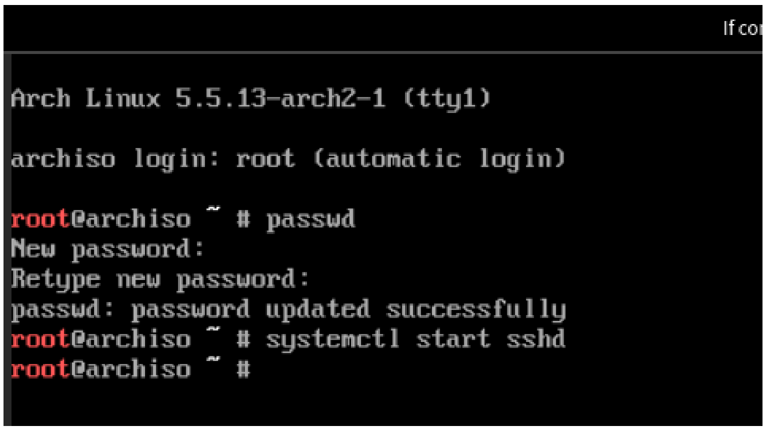
4 Salt-master installation to UpCloud

This chapter describes the steps necessary to install the command and control server and the latest version of the configuration management software. Special focus is given to security tools used and their working principles.

4.1 Deploy the server

Installing Arch Linux to cloud environment is not as straightforward as compared to other distributions. The default installation ISO is going to be used, although it is possible to construct a fully customized ISO that the distribution provides. Detailed instructions on installing written by the author have can be read from:

https://upcloud.com/community/tutorials/install-arch-linux/



```
Arch Linux 5.5.13-arch2-1 (tty1)

archiso login: root (automatic login)

root@archiso ~ # passwd
New password:
Retype new password:
passwd: password updated successfully
root@archiso ~ # systemctl start sshd
root@archiso ~ #
```

In short, the installation ISO is comes with a SSH daemon, which is employed immediately The server has a public IP address, so first we log with the console to get a terminal and set up a temporary strong password, and log in with SSH as root. This phase is quite vulnerable and using a very strong password is advisable. We are not going to allow root login or password authentication with SSH later, but at this stage it is necessary to do the actual installation.

Even if it is possible to do the installation from a cloud providers' console connection, these consoles are usually used as a failsafe if a remote SSH connection stops working.

With the SSH connection, we upgrade the package database and the archlinux-keyring. Then we install git to download the install scripts and needed files from this project's public github repository. Arch Linux's package manager is called 'pacman', and it is similar to apt, dpkg, yum, zypper and other package managers found across different Linux distributions.

```
# pacman -Sy
# pacman -S archlinux-keyring git
# git clone https://github.com/tqre/hardening-arch-linux
```

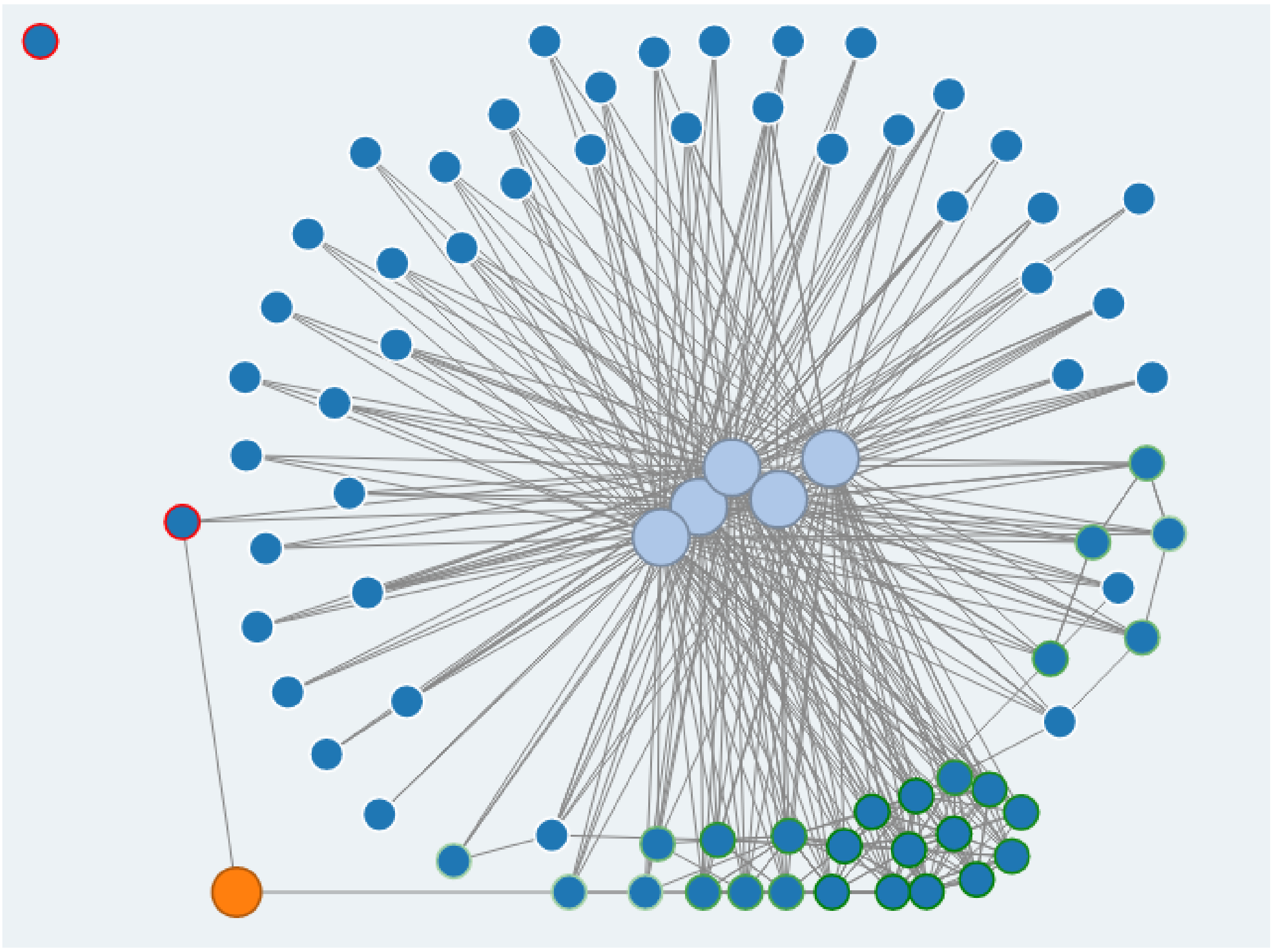
4.1.1 Arch Linux package management trust model

Arch Linux is maintained by volunteers, and a method of authenticating the official packages has to be defined. GNU Privacy Guard (GnuPG, GPG) keys are used to form a 'web of trust'. There are 5 Master signing keys, of which 3 are needed to sign Developers or Trusted Users keys.

The revocation keys for the master keys are not held by the owners. Revoking keys are held by another 5 individuals. With this arrangement, no individual has absolute root trust.

A visualization is provided on the web pages to elaborate the relationships between the Master and Developer keys.

- The 5 Master keys are in the middle, blue balls represent developers and trusted users.
- The orange ball represents CA Cert Signing Authority.
- A cluster of green outlined cross-signed key signatures is visible on the lower right corner.
- Keys with red outline are not allowed to sign official packages (retired founder in top left)



Arch Linux web of trust: https://www.archlinux.org/master-keys/#visualization

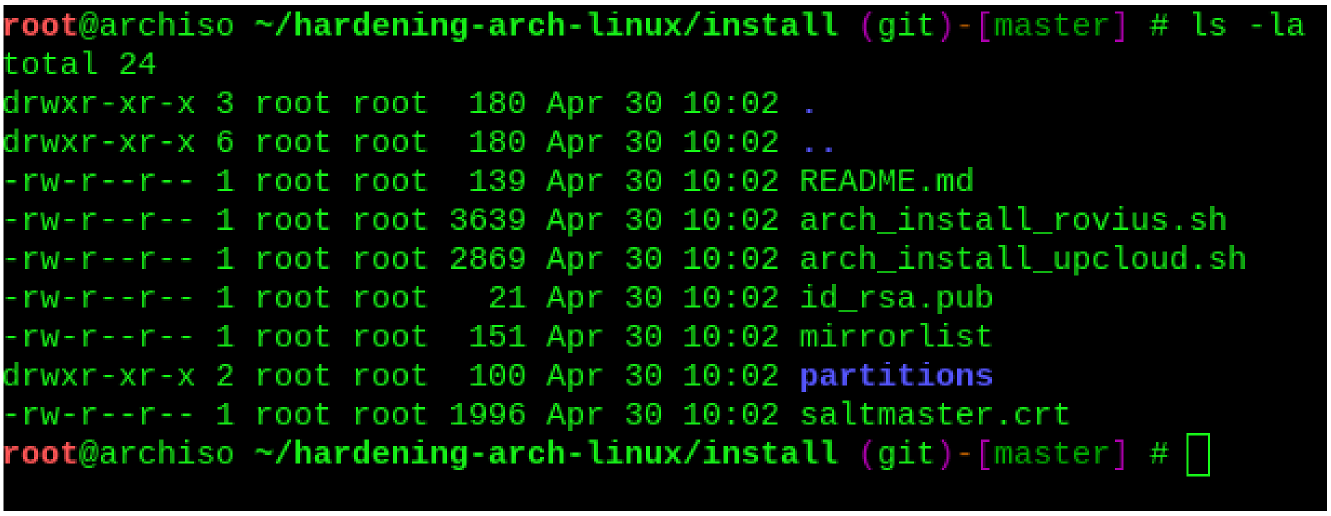
In this work, two points in this web of trust are considered: official and custom. The official packages are used in most of the software packages, and as we later in chapter 4.10 install Salt from outside the official repostories, we are going to generate our own GPG key to package and sign the software.

https://www.gnupg.org/gph/en/manual.html#AEN385
https://www.archlinux.org/master-keys/
https://wiki.archlinux.org/index.php/Pacman/Package_signing
https://wiki.archlinux.org/index.php/DeveloperWiki:Keyring_Package

4.2 Preparing the installation scripts

Once the scripts have been downloaded from git, some configuration is needed. The scripts provide variables where timezone, locale, keyboard and hostname can be changed. Additionally, SSH default port and the block device file descriptor name can be changed. The script is presented in Appendix 3.

The partitioning related files are located in partitions -directory, which contains 2 partitioning templates and a script to partition, format and label the templates. The file 'mirrorlist' contains the address of a mirror server to the official distributions' software packages. The server TLS certificate 'saltmaster.crt' is going to be generated anew in chapter 4.9.



```
root@archiso ~/hardening-arch-linux/install (git)-[master] # ls -la
total 24
drwxr-xr-x 3 root root  180 Apr 30 10:02 .
drwxr-xr-x 6 root root  180 Apr 30 10:02 ..
-rw-r--r-- 1 root root  139 Apr 30 10:02 README.md
-rw-r--r-- 1 root root 3639 Apr 30 10:02 arch_install_rovius.sh
-rw-r--r-- 1 root root 2869 Apr 30 10:02 arch_install_upcloud.sh
-rw-r--r-- 1 root root   21 Apr 30 10:02 id_rsa.pub
-rw-r--r-- 1 root root  151 Apr 30 10:02 mirrorlist
drwxr-xr-x 2 root root  100 Apr 30 10:02 partitions
-rw-r--r-- 1 root root 1996 Apr 30 10:02 saltmaster.crt
root@archiso ~/hardening-arch-linux/install (git)-[master] #
```

4.3 Salt-master remote management SSH keys

This is the second most important key, and it is used to remotely and securely connect to the salt-master virtual machine via public Internet. SSH is configured in the install scripts to only accept RSA key logins. Additionally, this keypair is going to be protected with a strong password. This means that SSH login is only possible when possessing the secret key and knowing the passphrase.

The keys are created with 'ssh-keygen' on the machine that is used to manage the master, and the public part of the key has to be copied to 'id_rsa.pub' -file. The install scripts then take care it is added to administrative account's '~/.ssh/authorized_keys' -file to enable remote access.

4.4 A note on cloud credentials

The most important credentials are to both cloud provider's management interfaces. Both cloud platforms offer two-factor authentication for securing the account further. From a security perspecive, there are many additional layers to consider. For example the browser that is used to access the account can be compromised, or the API gateway's transport security could be verified. These matters fall out of the scope of this work, but should of course be considered if deploying production systems.

4.5 Running the script

The install scripts create one account with full sudo rights. This password is critical when managing the salt-master, and provides a way to log in via the cloud provider's console in emergency situations. The password has to be set manually after the script has run. A notification is provided.

4.6 Detach the installation ISO and reboot

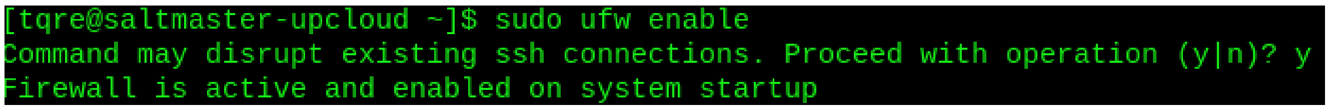
After rebooting, the installation is now ready to be configured further. SSH connection with passworded RSA authentication is used.

4.7 Firewall setup

Uncomplicated firewall (UFW) was installed, and it is used to configure the master's firewall. UFW uses Linux's native iptables to set up firewall rules, and provides an easy command line syntax to provide rules. We are going to need 5 rules, and the commands to achieve it are listed here. Note that the port numbers presented here are standard ports, in the actual project, some workarounds had to be done as outside connections from Rovius Cloud Platform were very limited.

Default SSH port 22 is used for remote administration. The minions are behind a NAT network, so all the connections come from the same address. Port 443 is open because Nginx http server is going to be set up to serve additional files to minions. SaltStack uses ports 4505 and 4506 on default. These firewall rules allow only IPv4 connections, IPv6 rules should be made separately if needed.

```
sudo ufw default deny
sudo ufw allow from 0.0.0.0/0 to any port 22 proto tcp
sudo ufw allow from 193.166.13.253 to any port 443 proto tcp
sudo ufw allow from 193.166.13.253 to any port 4505 proto tcp
sudo ufw allow from 193.166.13.253 to any port 4506 proto tcp
sudo ufw enable
```

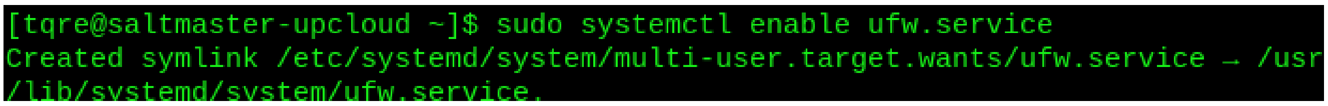


```
[tqre@saltmaster-upcloud ~]$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
```

Important: even if UFW claims that the firewall is enabled on system startup, this is NOT true!

This might be true to other distributions as they usually enable services as they are installed via package manager. With Arch Linux, this does not happen, and to start a service start on system boot, a symlink has to be created with systemctl:

```
sudo systemctl enable ufw.service
```



```
[tqre@saltmaster-upcloud ~]$ sudo systemctl enable ufw.service
Created symlink /etc/systemd/system/multi-user.target.wants/ufw.service → /usr
/lib/systemd/system/ufw.service.
```

4.8 Webserver configuration

Nginx was installed on the server, and it is going to be configured as a file server via HTTPS. As it happens, there is a CIS benchmark for securing Nginx too. This was however not done in great detail to limit the project's scope. The Nginx configuration file used is presented in Appendix 4.

As per firewall rules, only connections from a certain address are allowed to this server. The served files are going to be located in '/srv/http'. The Nginx fileserver is configured to use Transpor Layer Security version 1.3, which is the latest version enabling end-to-end encryption communications, authenticating the server and ensuring data integrity in transit.

Well change the ownership of the /srv/http directory, and put our administrative user into http group. We will give write access to the directory for the group.

```
sudo chown root:http /srv/http
sudo chmod 775 /srv/http
sudo usermod -aG http <USER>
```

These permissions can and should be tightened later.

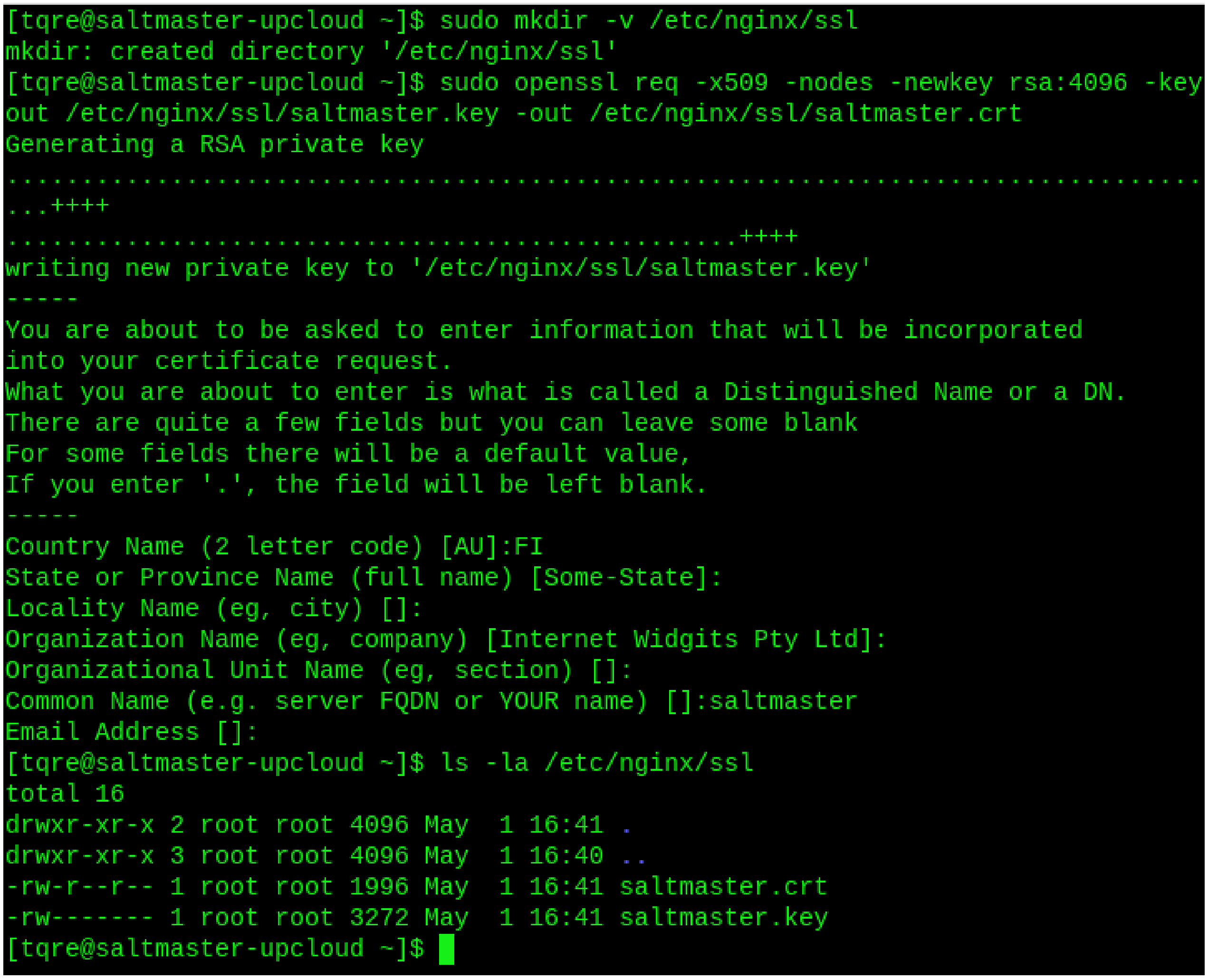
4.9 Server TLS Certificate

Transport Layer Security (TLS) secures communications using symmetric cryprography to encrypt the data in transit. Authentication is provided in form of certificates, and message integrity is checked with a message authentication code.

The web communications have to contain a certain amount of readable data, which includes for example IP addresses and port numbers for routers to be able to route the packets to their destinations. To achieve secure communications, TLS v.1.3 uses authenticated encryption with associated data (AEAD). It combines encrypted communications with a message authentication code, which ensures integrity and authenticity for the plaintext content aswell. The encrypted content adds confidentiality to the communications.

To authenticate the fileserver to minions, and encrypt the communications, we are going to generate a public key TLS certificate. The public part of the certificate key is going to be handed over to the minions, thus estabilishing a miniature public key infrastructure to the environment. OpenSSL is used to generate the certificate, which doesn't have a password (-nodes option) because the web server will ask the password every time it is restarted. The following command achieves this:

```
sudo mkdir -v /etc/nginx/ssl
sudo openssl req -x509 -nodes -newkey rsa:4096 \
                      -keyout /etc/nginx/ssl/saltmaster.key \
                      -out /etc/nginx/ssl/saltmaster.crt
```



```
[tqre@saltmaster-upcloud ~]$ sudo mkdir -v /etc/nginx/ssl
mkdir: created directory '/etc/nginx/ssl'
[tqre@saltmaster-upcloud ~]$ sudo openssl req -x509 -nodes -newkey rsa:4096 -key
out /etc/nginx/ssl/saltmaster.key -out /etc/nginx/ssl/saltmaster.crt
Generating a RSA private key
................................................................................
...++++
.................................................++++
writing new private key to '/etc/nginx/ssl/saltmaster.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FI
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:saltmaster
Email Address []:
[tqre@saltmaster-upcloud ~]$ ls -la /etc/nginx/ssl
total 16
drwxr-xr-x 2 root root 4096 May  1 16:41 .
drwxr-xr-x 3 root root 4096 May  1 16:40 ..
-rw-r--r-- 1 root root 1996 May  1 16:41 saltmaster.crt
-rw------- 1 root root 3272 May  1 16:41 saltmaster.key
[tqre@saltmaster-upcloud ~]$
```

When generating certificates, the Common Name (CN) field is important. The certificate is only valid if the authorization request's hostname matches the CN of the certificate.

'salmaster.key' is the secret part of the certificate keypair and is used to identify and authenticate the server.
'saltmaster.crt' is the public key, and it needs to be copied over to the salt-minions. This is done in the minion configuration phase, chapter 5.4. It has to be copied over to the '/srv/http' directory to be downloadable for the minions.

Now that we have the certificate, we can enable and start the web server with systemctl:

```
sudo systemctl enable nginx
sudo systemctl start nginx
```

https://www.ssl.com/faqs/what-is-https/
https://www.ssl.com/faqs/faq-what-is-ssl/
https://www.ssl.com/faqs/what-is-an-x-509-certificate/

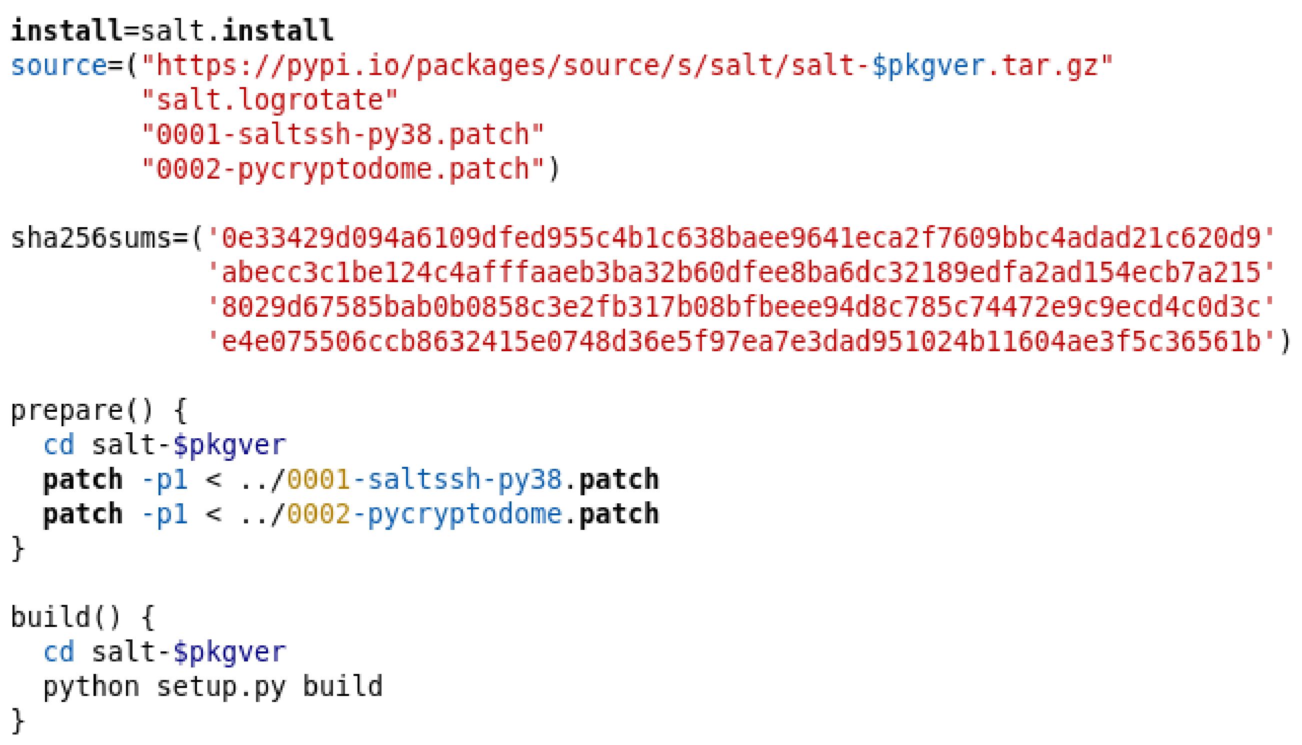
4.10 SaltStack installation

In chapter 2.3 it was mentioned that the latest version of the configuration management software is not available from the official Arch Linux repositories. Some additional checks and a method of signing the package before distributing it is introduced along with the installation of the salt-master itsef.

4.10.1 Preparation

In chapter 2.3 it was mentioned that the latest version of the configuration management software is not available from the official Arch Linux repositories. The installation of unofficial packages is always a security risk, so we have to take a closer look what we are actually installing. The important file in Arch Linux package management is a PKGBUILD file:

https://aur.archlinux.org/cgit/aur.git/tree/PKGBUILD?h=salt-py3



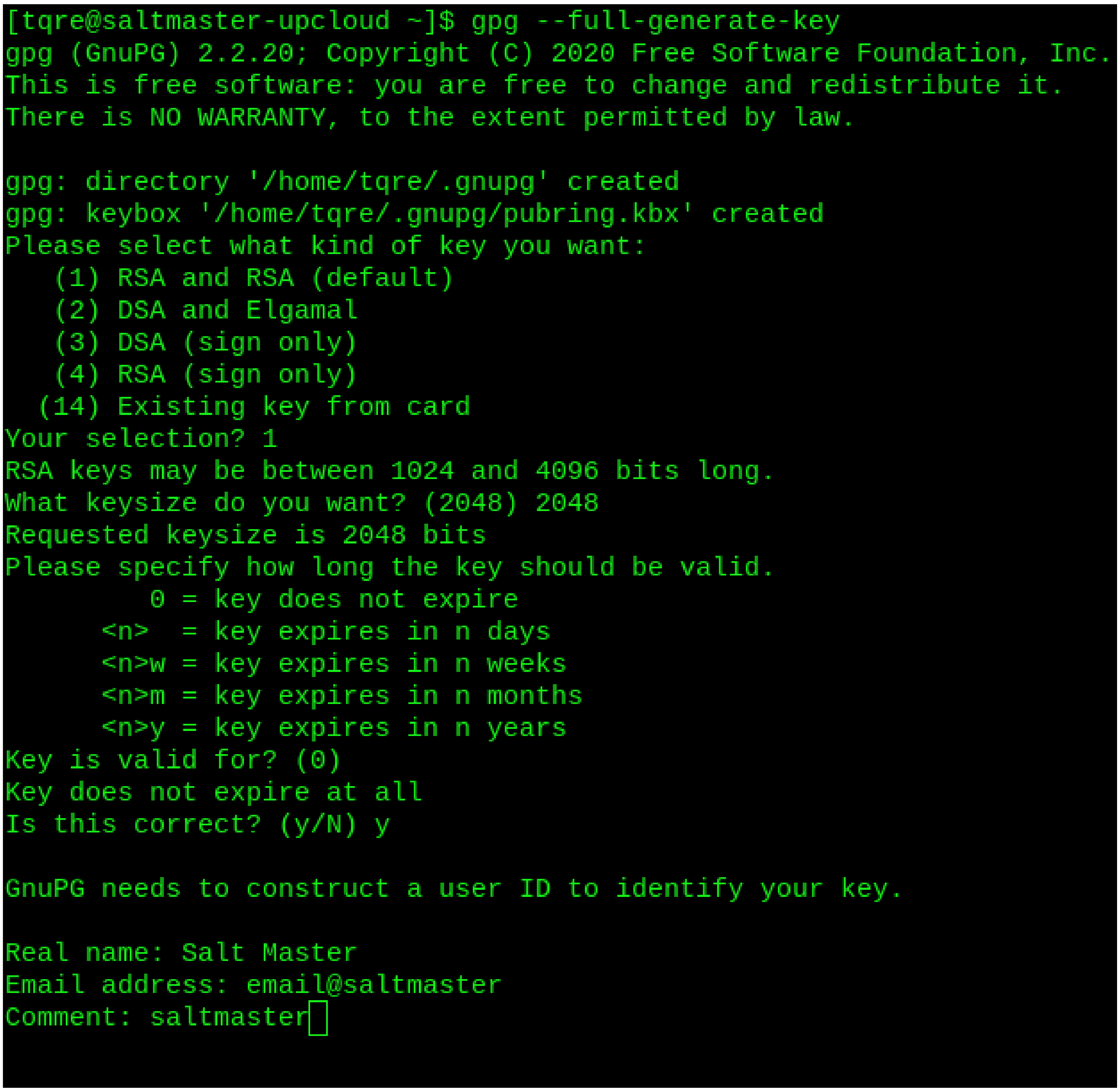
```
install=salt.install
source=("https://pypi.io/packages/source/s/salt/salt-$pkgver.tar.gz"
        "salt.logrotate"
        "0001-saltssh-py38.patch"
        "0002-pycryptodome.patch")

sha256sums=('0e33429d094a6109dfed955c4b1c638baee9641eca2f7609bbc4adad21c620d9'
            'abecc3c1be124c4afffaaeb3ba32b60dfee8ba6dc32189edfa2ad154ecb7a215'
            '8029d67585bab0b0858c3e2fb317b08bfbeee94d8c785c74472e9c9ecd4c0d3c'
            'e4e075506ccb8632415e0748d36e5f97ea7e3dad951024b11604ae3f5c36561b')

prepare() {
  cd salt-$pkgver
  patch -p1 < ../0001-saltssh-py38.patch
  patch -p1 < ../0002-pycryptodome.patch
}

build() {
  cd salt-$pkgver
  python setup.py build
}
```

The essential information from the PKGBUILD is in the picture, and we can see that a compressed file from pypi.io is used. SaltStack is coded with python, and pypi is an official repository for python projects. Looking at pypi.org's information, the author of the package is the same as the main developer of Salt itself. The packed file itself contains the source code, which is also obtainable from the official SaltStack github repository. The default setup script is run with python on the last line.

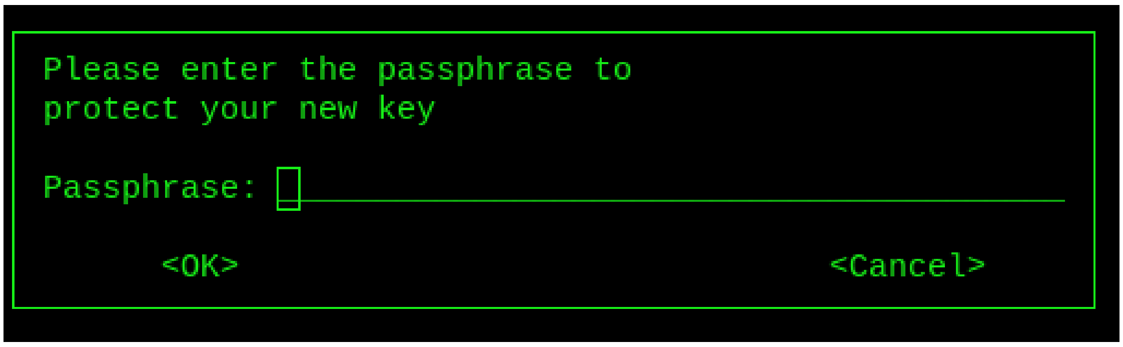
4.10.2 GPG Key for package signing

We are going to sign the generated package, so the minions who are using the fileserver to download the salt-package will be able to verify that the package was packaged by the salt-master. So before creating the package, we need to create another key to sign it.

GPG stands for GNU Privacy Guard. It is an open-source version of Symantec's PGP cryptographic software suite. As was studied in chapter 4.1.1, GPG keys are used in Arch Linux package management, so we are going to follow that practise and use our own GPG key to sign the salt-py3 package we are making on the master server.


```
[tqre@saltmaster-upcloud ~]$ gpg --full-generate-key
gpg (GnuPG) 2.2.20; Copyright (C) 2020 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: directory '/home/tqre/.gnupg' created
gpg: keybox '/home/tqre/.gnupg/pubring.kbx' created
Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
  (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 2048
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: Salt Master
Email address: email@saltmaster
Comment: saltmaster
```

After this, the GPG software checks for the information, and requests for a passphrase:


```
Please enter the passphrase to
protect your new key

Passphrase: 

      <OK>                              <Cancel>
```

GPG creates files to '~/.gnupg' -directory, including a revocation key which is used in the case the key gets compromised. It is possible to upload the key to a public keyserver for a wider access, this is a good way to ensure the GPG keys' validity. Major keyservers accross the internet synchronize their data between each other, and if a revocation key is passed if one wants to invalidate a certain key, that information is available from there too.

We are not going to use a public keyserver, but we are going to distribute the public part of the GPG key via the fileserver in minions' installation process. The minions need to add the master's GPG key to their trust databases, which is covered in chapter 5.5.

```
gpg --armor --export --output /srv/http/saltmaster.gpg
```

As GPG-keys are in binary format, the 'armor' command refers to ASCII-armor, which is a procedure where a binary file is converted to ASCII -format, making it readable and transportable across different platforms and mediums. In the past it has been even possible for an ASCII-armored signature to execute a malicious payload once it is converted back to a binary by a parser program.

https://www.itprotoday.com/strategy/windows-pgp-ascii-armor-parser-vulnerability
https://www.gnupg.org/gph/en/manual/x457.html
https://gnupg.org/gph/en/manual.html

4.10.3 Installing SaltStack with AUR tools

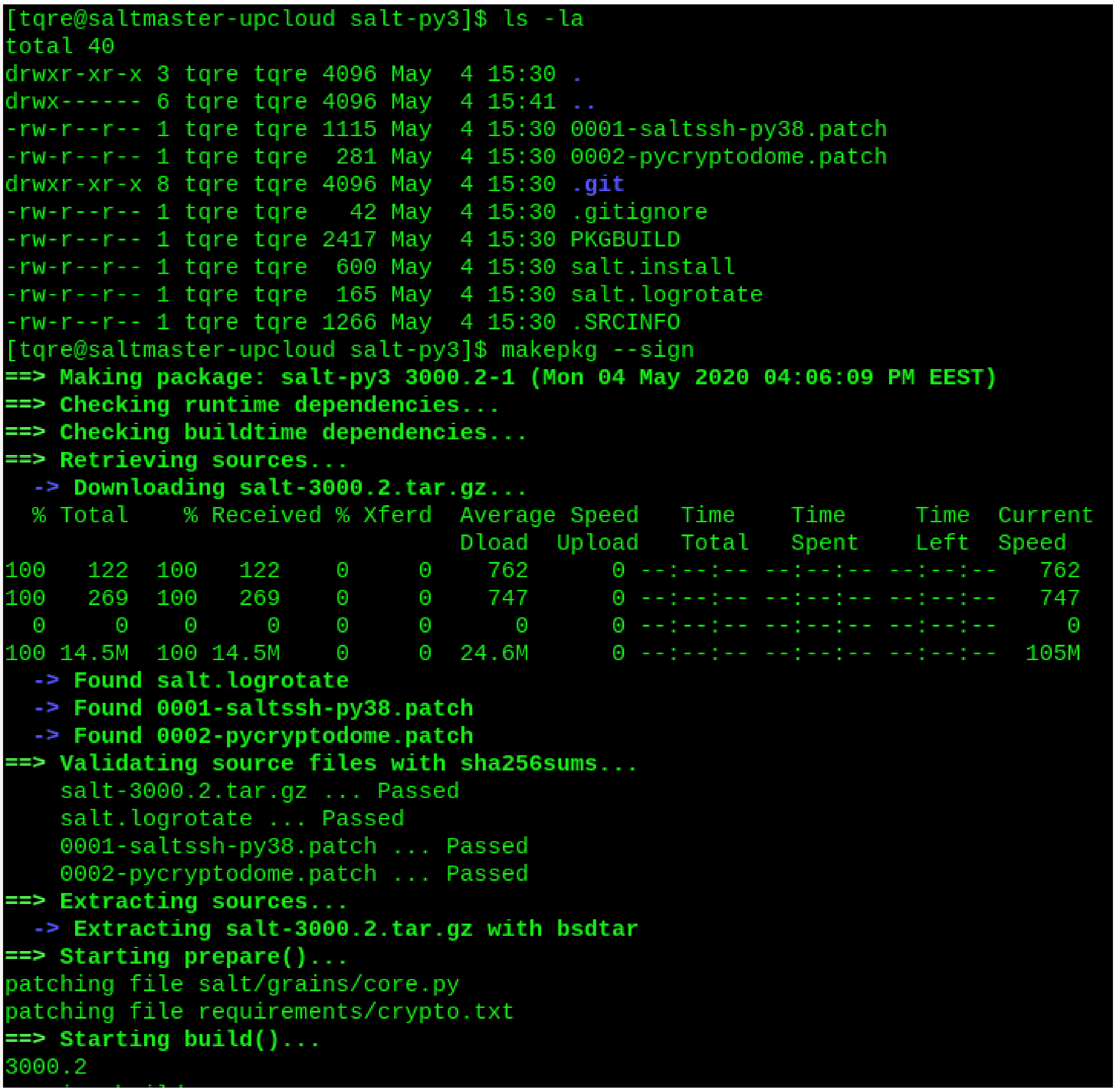
We are using makepkg -tool, which creates a package for SaltStack, and install it to the master. The package is going to be signed with the GPG key generated in 4.10.2, and it is going to be served exclusively to the minions with Nginx as a fileserver. The minions can install SaltStack with the Arch Linux's default package manager 'pacman' with few additional configuration stepts, and they can validate that the package was indeed built by the salt-master server.

We have to cloned the salt-py3 repository first:

```
$ git clone https://aur.archlinux.org/salt-py3.git
```

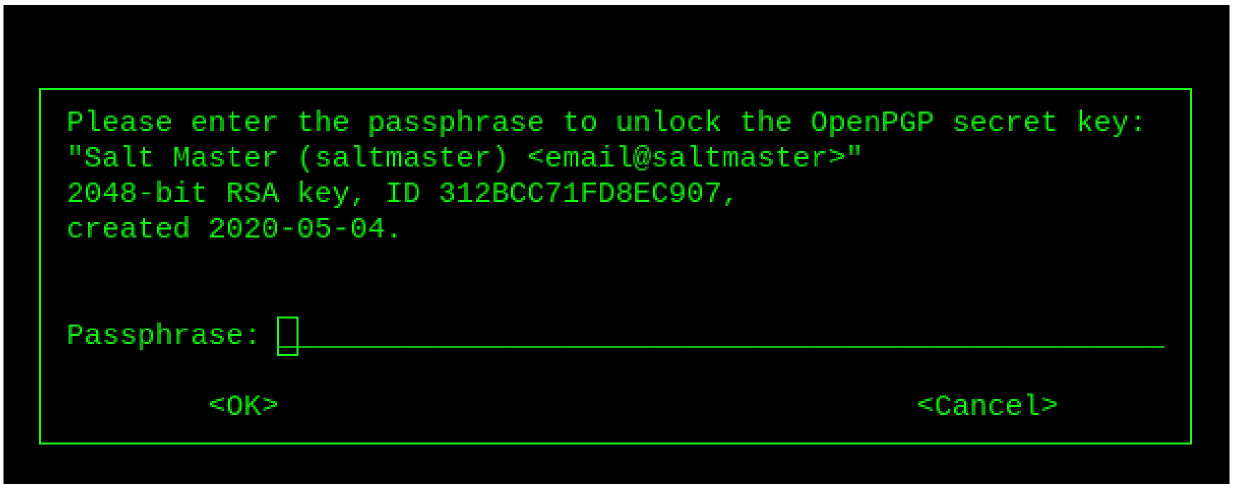
After the cloning, we use makepkg -utility tool to sign and package the software from source.

```
$ makepkg --sign
```



```
[tqre@saltmaster-upcloud salt-py3]$ ls -la
total 40
drwxr-xr-x 3 tqre tqre 4096 May  4 15:30 .
drwx------ 6 tqre tqre 4096 May  4 15:41 ..
-rw-r--r-- 1 tqre tqre 1115 May  4 15:30 0001-saltssh-py38.patch
-rw-r--r-- 1 tqre tqre  281 May  4 15:30 0002-pycryptodome.patch
drwxr-xr-x 8 tqre tqre 4096 May  4 15:30 .git
-rw-r--r-- 1 tqre tqre   42 May  4 15:30 .gitignore
-rw-r--r-- 1 tqre tqre 2417 May  4 15:30 PKGBUILD
-rw-r--r-- 1 tqre tqre  600 May  4 15:30 salt.install
-rw-r--r-- 1 tqre tqre  165 May  4 15:30 salt.logrotate
-rw-r--r-- 1 tqre tqre 1266 May  4 15:30 .SRCINFO
[tqre@saltmaster-upcloud salt-py3]$ makepkg --sign
==> Making package: salt-py3 3000.2-1 (Mon 04 May 2020 04:06:09 PM EEST)
==> Checking runtime dependencies...
==> Checking buildtime dependencies...
==> Retrieving sources...
  -> Downloading salt-3000.2.tar.gz...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   122  100   122    0     0    762      0 --:--:-- --:--:-- --:--:--   762
100   269  100   269    0     0    747      0 --:--:-- --:--:-- --:--:--   747
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 14.5M  100 14.5M    0     0  24.6M      0 --:--:-- --:--:-- --:--:--  105M
  -> Found salt.logrotate
  -> Found 0001-saltssh-py38.patch
  -> Found 0002-pycryptodome.patch
==> Validating source files with sha256sums...
    salt-3000.2.tar.gz ... Passed
    salt.logrotate ... Passed
    0001-saltssh-py38.patch ... Passed
    0002-pycryptodome.patch ... Passed
==> Extracting sources...
  -> Extracting salt-3000.2.tar.gz with bsdtar
==> Starting prepare()...
patching file salt/grains/core.py
patching file requirements/crypto.txt
==> Starting build()...
3000.2
```

After the package is compressed, we are asked to sign the package:



```
Please enter the passphrase to unlock the OpenPGP secret key:
"Salt Master (saltmaster) <email@saltmaster>"
2048-bit RSA key, ID 312BCC71FD8EC907,
created 2020-05-04.


Passphrase: _________________________________________________

        <OK>                                        <Cancel>
```

The package is not installed yet, and we can examine the package source code at this point if we want. The makepkg utility creates the following files:

```
-rw-r--r-- 1 tqre tqre 11793336 May  4 16:07 salt-py3-3000.2-1-any.pkg.tar.xz
-rw-r--r-- 1 tqre tqre      310 May  4 16:07 salt-py3-3000.2-1-any.pkg.tar.xz.sig
```

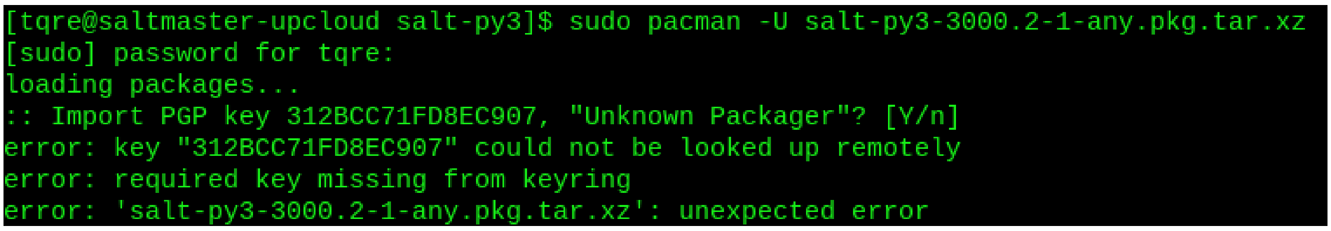
The package itself and the signature file are copied over to the fileserver's public directory.

```
cp /home/<user>/salt-py3/salt-py3-3000.2-1any.pkg.tar.xz* /srv/http
```

4.10.4 Installing and configuring SaltStack

Installing the local package can now be done with the default package manager 'pacman'.

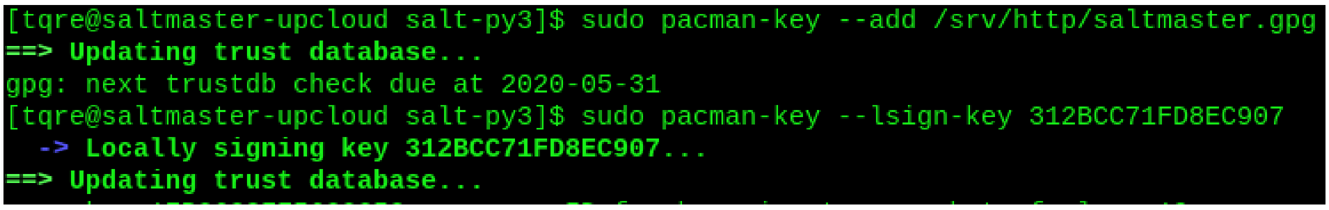
```
pacman -U salt-py3-3000.2-1-any.pkg.tar.xz
```



```
[tqre@saltmaster-upcloud salt-py3]$ sudo pacman -U salt-py3-3000.2-1-any.pkg.tar.xz
[sudo] password for tqre:
loading packages...
:: Import PGP key 312BCC71FD8EC907, "Unknown Packager"? [Y/n]
error: key "312BCC71FD8EC907" could not be looked up remotely
error: required key missing from keyring
error: 'salt-py3-3000.2-1-any.pkg.tar.xz': unexpected error
```

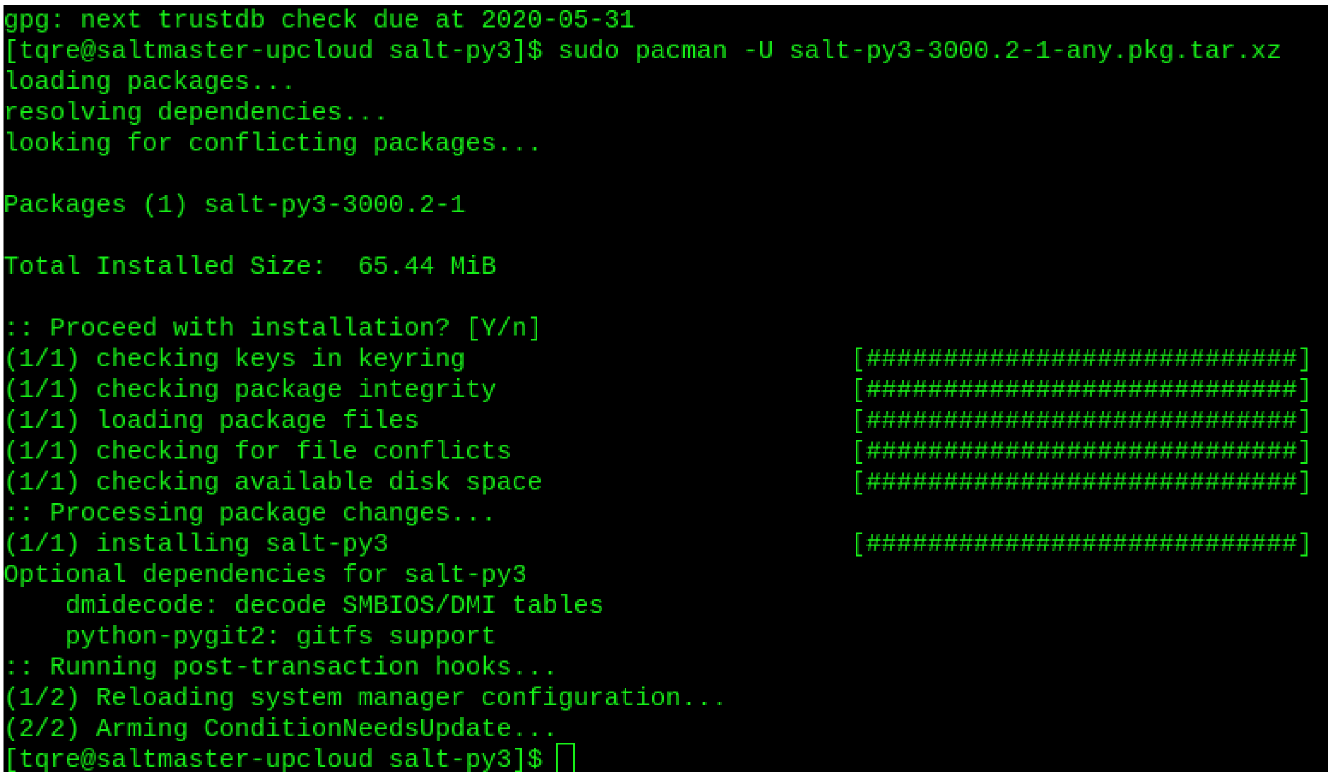
We have signed the package with our own key, but the key needs to be added to the trusted keys keyring. The public ASCII-armored key was copied over to '/srv/http', so this can be done with the commands:

```
sudo pacman-key --add /srv/http/saltmaster.gpg
sudo pacman-key --lsign-key <KEYID>
```



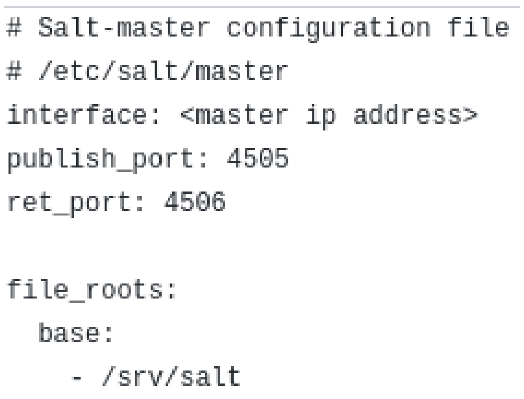
```
[tqre@saltmaster-upcloud salt-py3]$ sudo pacman-key --add /srv/http/saltmaster.gpg
==> Updating trust database...
gpg: next trustdb check due at 2020-05-31
[tqre@saltmaster-upcloud salt-py3]$ sudo pacman-key --lsign-key 312BCC71FD8EC907
  -> Locally signing key 312BCC71FD8EC907...
==> Updating trust database...
```

First the key is added to the local keyring, then it is locally signed (lsign).After this operation, pacman agrees to install the package.



```
gpg: next trustdb check due at 2020-05-31
[tqre@saltmaster-upcloud salt-py3]$ sudo pacman -U salt-py3-3000.2-1-any.pkg.tar.xz
loading packages...
resolving dependencies...
looking for conflicting packages...

Packages (1) salt-py3-3000.2-1

Total Installed Size:  65.44 MiB

:: Proceed with installation? [Y/n]
(1/1) checking keys in keyring                       [#############################]
(1/1) checking package integrity                     [#############################]
(1/1) loading package files                          [#############################]
(1/1) checking for file conflicts                    [#############################]
(1/1) checking available disk space                  [#############################]
:: Processing package changes...
(1/1) installing salt-py3                            [#############################]
Optional dependencies for salt-py3
    dmidecode: decode SMBIOS/DMI tables
    python-pygit2: gitfs support
:: Running post-transaction hooks...
(1/2) Reloading system manager configuration...
(2/2) Arming ConditionNeedsUpdate...
[tqre@saltmaster-upcloud salt-py3]$ 
```

A configuration file in /etc/salt/master is edited:



```
# Salt-master configuration file
# /etc/salt/master
interface: <master ip address>
publish_port: 4505
ret_port: 4506

file_roots:
  base:
    - /srv/salt
```

After installing and configuring, salt-master is ready to be enabled and started:

```
sudo systemctl enable salt-master
sudo systemctl start salt-master
```

4.11 Set up a private package manager repository

Finally we are going to configure a custom package manager repository, so salt-minions can get updates to salt-py3 package from the salt-master's file server with pacman. This is accomplished with just one command to create a package database file:

```
sudo repo-add /srv/http/saltmaster.db.tar.gz \
     /srv/http/salt-py3-3000.2.1-any.pkg.tar.xz
```

The signature file is automatically included in the package manager database. We also have to configure the repository for the minions. This is done in chapter 5.5.

4.12 Keeping the salt-py3 package updated

Good practice is to keep all the software up to date, including salt-py3 package. This would be best to automate, but the package signing needs to be done manually. This is a drawback we accept for having signed packages. As Python 3 supported Saltstack is not in the Arch's official repostiories solely because of the 'python-msgpack' old version dependency, there is a possiblity that this might change in the future.

https://github.com/saltstack/salt/issues/56007

5 Salt-minion installation to Rovius Cloud Platform

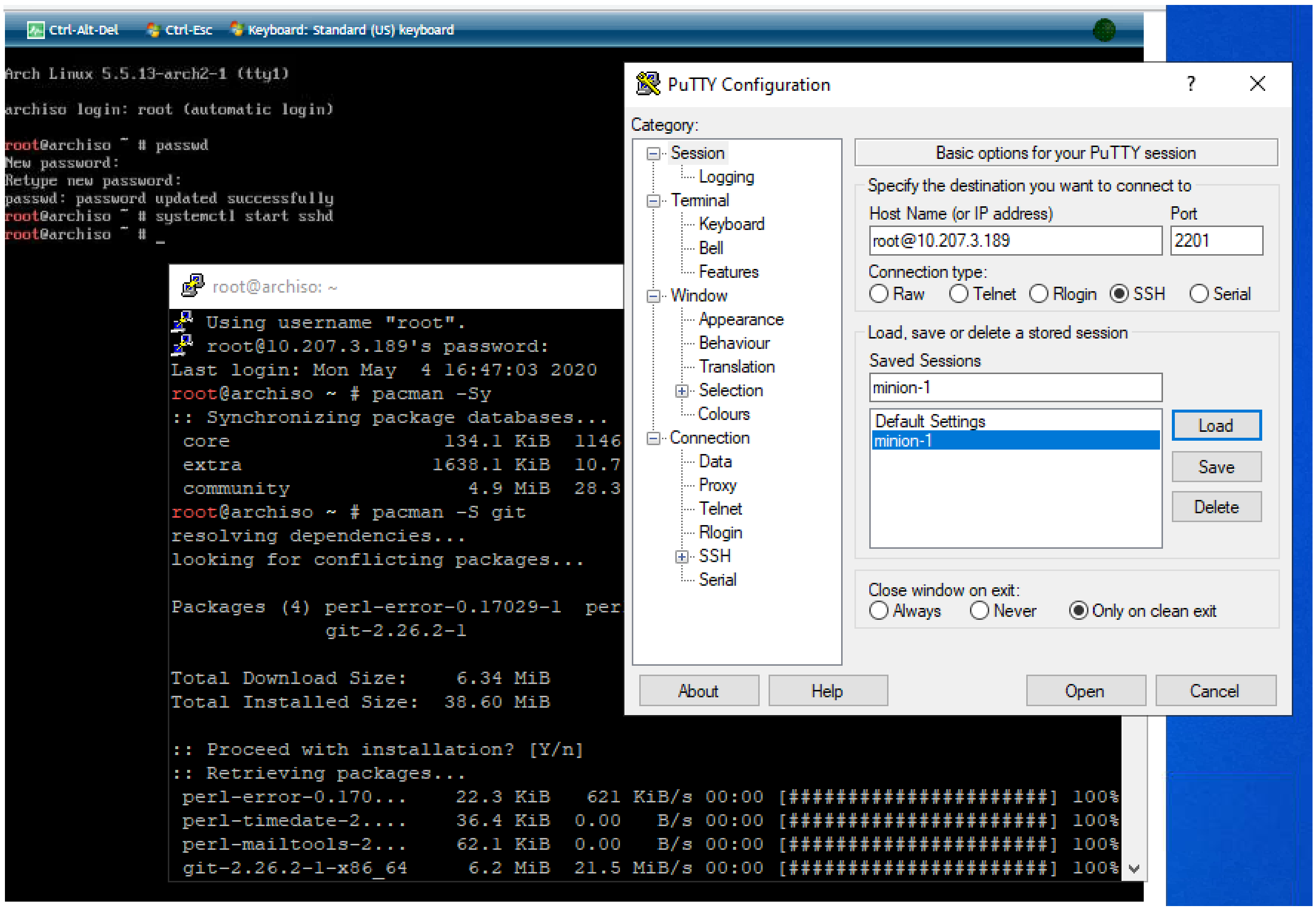
Salt-minions are installed in another cloud platform, and are also going to be Arch Linux based. Separate installation scripts are needed as the environment these virtual machines reside is behind a NATted firewall. The maintenance SSH connection is only possible via a VDI connection running Windows, to which the project has only very basic privileges.

Minion installation is performed otherwise similarily to the master configuration, except that the installation sets up the virtual machine instance as a salt-minion, and connects after the first reboot. Thus the salt-master server is able to control it right from the start. As the configuration management software runs on root privileges, we can do all necessary configuration to the minions remotely. This means that users and even passwords can be managed remotely in a centralized fashion.

5.1 Deploy the server on cloud environment

We are following same procedure in the start as we did on the master:
1. Starting the VM from the default installation ISO.
2. Setting up an ad-hoc SSH connection.
3. Installing git and cloning the projects github repository to the minion.

The cloud environment specific network settings are not covered here. The minions need an internet access to be succesfully deployed, and port forwarding rules are needed to estabilish SSH connection.



```
Arch Linux 5.5.13-arch2-1 (tty1)

archiso login: root (automatic login)

root@archiso ~ # passwd
New password:
Retype new password:
passwd: password updated successfully
root@archiso ~ # systemctl start sshd
root@archiso ~ # _
```

root@archiso: ~

```
Using username "root".
root@10.207.3.189's password:
Last login: Mon May  4 16:47:03 2020
root@archiso ~ # pacman -Sy
:: Synchronizing package databases...
 core                     134.1 KiB  1146
 extra                   1638.1 KiB  10.7
 community                  4.9 MiB  28.3
root@archiso ~ # pacman -S git
resolving dependencies...
looking for conflicting packages...

Packages (4) perl-error-0.17029-1  per
             git-2.26.2-1

Total Download Size:    6.34 MiB
Total Installed Size:  38.60 MiB

:: Proceed with installation? [Y/n]
:: Retrieving packages...
 perl-error-0.170...     22.3 KiB   621 KiB/s 00:00 [######################] 100%
 perl-timedate-2....     36.4 KiB  0.00   B/s 00:00 [######################] 100%
 perl-mailtools-2...     62.1 KiB  0.00   B/s 00:00 [######################] 100%
 git-2.26.2-1-x86_64      6.2 MiB  21.5 MiB/s 00:00 [######################] 100%
```

PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): root@10.207.3.189 Port: 2201

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: minion-1

Default Settings
minion-1

Close window on exit: Always Never [✓ Only on clean exit]

After cloning the project repository, all that is left is few configuration settings in the installation script: username, hostname and the salt-master's IP-address . We are also going to need the public GPG key id of the salt-master. Some other settings can be configured here aswell:

```
# General settings:
TZ="Europe/Helsinki"
LOC="en_US.UTF-8"
KEYBOARD="us"
SUDOUSER="user"
PASSWORD="user"
GPG_KEYID=""

# Salt settings
MASTER_IP="<ip-address>"
FILESERVER_PORT="80"
M_PORT="4506"
P_PORT="4505"
HOSTNAME="minion-"
```

In the following chapters the key elements of the installation script are discussed. Partition maps are set up as described in chapter 3.1.

5.2 Installed packages

The minions need less packages installed, which in turn makes the attack surface smaller. However the dependencies of salt-py3 have to be satisfied.

5.3 Hostname and sudo user

An administrative can be set up in the installation process, but it can also be managed with configuration management. A maintenance account is needed for troubleshooting, and in testing the scripts and configuration management states it is essential. Saltmasters ip-address is set in '/etc/hosts' file, and can be referred as 'saltmaster' later in the scripts and configuration fiiles.

In other words, the method creating a user with a password emplyoed in the installation script is not intended to be the final solution. Passwords in plain text should not be used anywhere.

5.4 Saltmaster server's TLS certificate, PCKS#11

The public part of salt-masters TLS certificate is downloaded when the github repository is cloned, and it is copied over to the installation. The certificates have to be included as trust anchors with the following commands in the script. Note that the project's git repository was used as an intermediate storage for the certificate file, and was downloaded upon cloning the repository in chapter 5.1.

```
cp saltmaster.crt /mnt/etc/ssl/private/saltmaster.crt
trust anchor /etc/ssl/private/saltmaster.crt
update-ca-trust
```

These commands create a 'saltmaster.p11-kit 'file into '/etc/ca-certificates/trust-source' -directory. PKCS#11 is a cryptographic standard, which defines a complex API in C programming language to manage cryptographic tokens such as the certificate we are handling here.

P11-kit is a Linux software package that provide the implementation for loading and enumerating PKCS#11 modules, and the file created in the above process is a wrapped certificte that can be used with other software that complies with PKCS#11, for example a web browser or any other program that needs to perform cryptographic tasks.

http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.html
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/security_hardening/configuring-applications-to-use-cryptographic-hardware-through-pkcs-11_security-hardening
https://en.wikipedia.org/wiki/List_of_applications_using_PKCS_11

5.5 Salt-master server's GPG key and custom pacman repository

With certificate, we can use wget to download the public GPG key from the fileserver running on master. The GPG key's public keyid has to be provided in the installation script settings, and it is added to the package manager's keychain signing it locally.

```
wget -P /home/$SUDOUSER/.gnupg https://saltmaster:$FILESERVER_PORT/saltmaster.gpg
pacman-key --add /home/$SUDOUSER/.gnupg/saltmaster.gpg
pacman-key --lsign-key $GPG_KEYID
```

The package manager address is set in '/etc/hosts' file, and the pacman configuration file '/etc/pacman.conf' is edited in the script to include salt-master as a package server.

```
echo -e "[saltmaster]\nServer = https://saltmaster:$FILESERVER_PORT\n" >> /etc/pacman.conf
```

5.6 SaltStack configuration

Salt-minions need to have a unique name, and they need the master server's IP address. This is set on the installation process, and the address is written into '/etc/hosts' file so that the controlling server can be referred as 'saltmaster'.

```
echo -e "master: saltmaster\nmaster_port: $M_PORT\npublish_port: $P_PORT\nid: $HOSTNAME"
> /etc/salt/minion
```

5.7 Restart, Salt PKI, template creation possibility

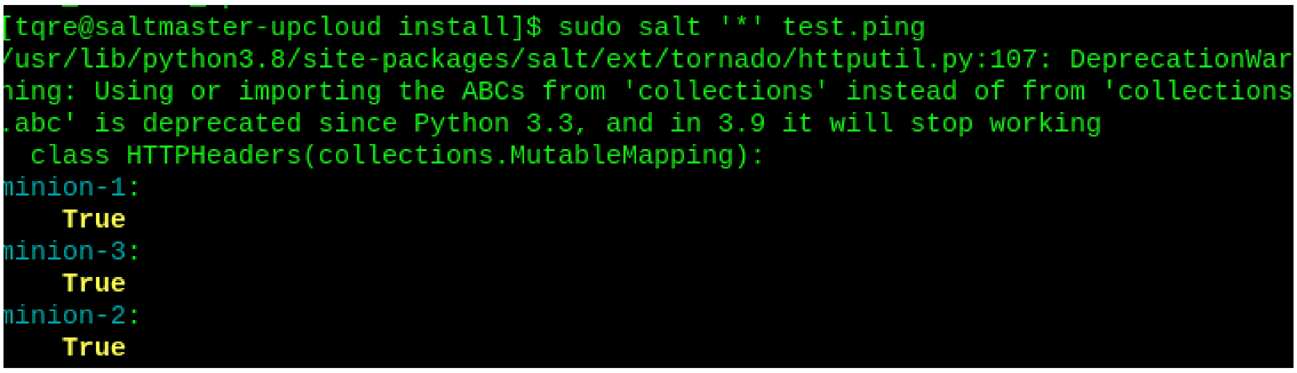
After the configuration script has run it's course, the new virtual machine has to be rebooted, and the installation ISO removed. Later configurations can be done exclusively with SaltStack. There is a possibility to create a template from the server at this point, but there are some points to consider before doing it.

Salt's public key infrastructure keys are created when the salt-minion.service is started for the first time. The authentication mechanism is a typical PKI solution, where the keys have their private and public parts respectively. It the service was started before creating a template from the server image, the minion's RSA-keypair would also be duplicated in the process. In the scripts the service is only enabled on startup, but not started.

As we are managing the network settings with systemd, we should clear /etc/machine-id file before templatizing the machine, as this id is unique and used in the dhcp client implementation of systemd. There are other factors, such as MAC-addressess of the networking interfaces and partition UUID's that get cloned and have to be taken care of. The process of templatization was left out from this work.

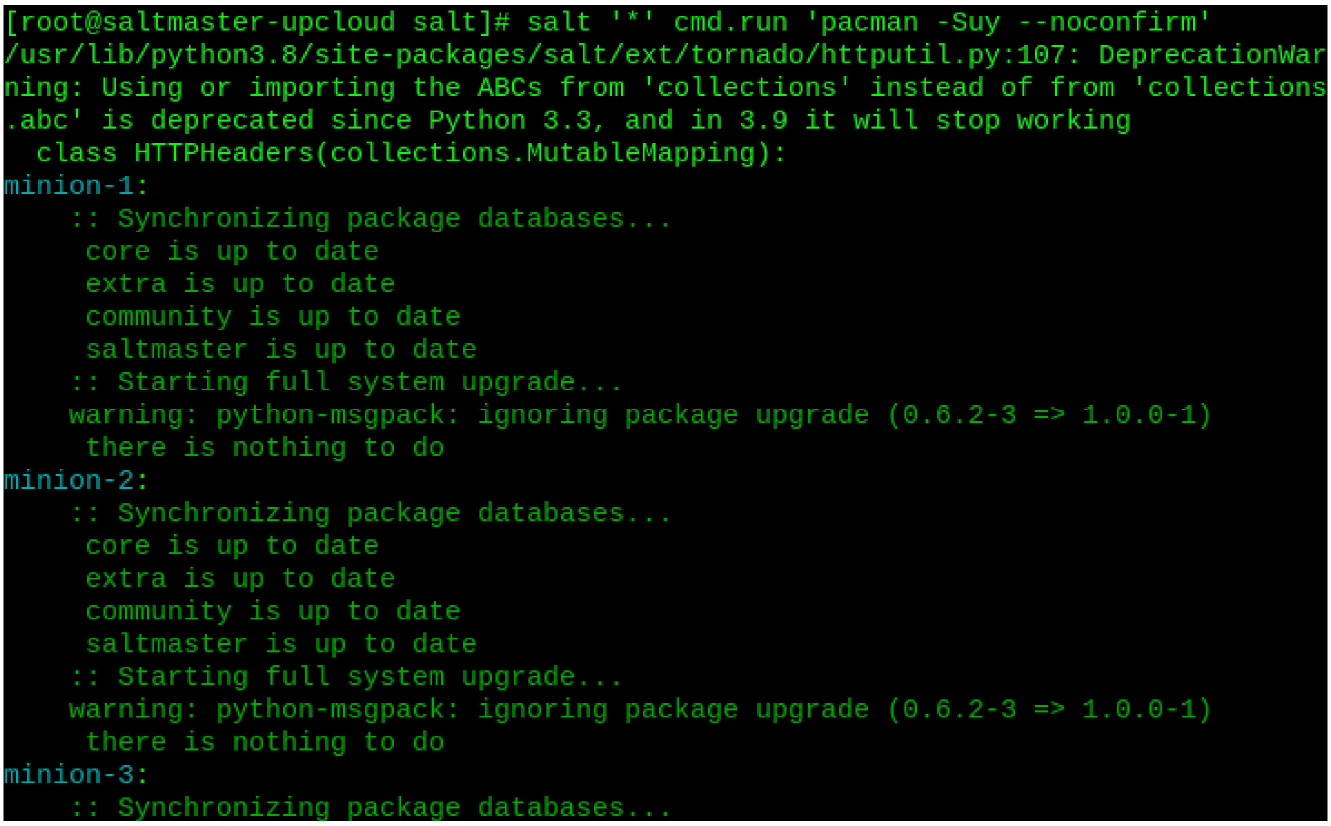
5.8 Test connections

After accepting the keys on master with 'sudo salt-key -A', the master shows that all 3 minions that were installed are answering.



```
[tqre@saltmaster-upcloud install]$ sudo salt '*' test.ping
/usr/lib/python3.8/site-packages/salt/ext/tornado/httputil.py:107: DeprecationWar
ning: Using or importing the ABCs from 'collections' instead of from 'collections
.abc' is deprecated since Python 3.3, and in 3.9 it will stop working
  class HTTPHeaders(collections.MutableMapping):
minion-1:
    True
minion-3:
    True
minion-2:
    True
```

And a fresh install needs no updates:



```
[root@saltmaster-upcloud salt]# salt '*' cmd.run 'pacman -Suy --noconfirm'
/usr/lib/python3.8/site-packages/salt/ext/tornado/httputil.py:107: DeprecationWar
ning: Using or importing the ABCs from 'collections' instead of from 'collections
.abc' is deprecated since Python 3.3, and in 3.9 it will stop working
  class HTTPHeaders(collections.MutableMapping):
minion-1:
    :: Synchronizing package databases...
     core is up to date
     extra is up to date
     community is up to date
     saltmaster is up to date
    :: Starting full system upgrade...
    warning: python-msgpack: ignoring package upgrade (0.6.2-3 => 1.0.0-1)
     there is nothing to do
minion-2:
    :: Synchronizing package databases...
     core is up to date
     extra is up to date
     community is up to date
     saltmaster is up to date
    :: Starting full system upgrade...
    warning: python-msgpack: ignoring package upgrade (0.6.2-3 => 1.0.0-1)
     there is nothing to do
minion-3:
    :: Synchronizing package databases...
```

The saltmaster custom package repository can be seen, and the package ignored for updates.

5.9 Updating considerations

Linux servers should need rebooting only when the kernel or the init system (systemd) gets updated. With Arch Linux, the kernel is updated weekly. Systems with very high demands on availability can be considered to set up with a long term support (LTS) kernel. Rebooting the minions with configuration management is however a fast process, especially now when there is a minimal operating system onboard.

Also a command line induced reboot is considerably faster than restarting it from the cloud providers' interface. It of course depends on the provider, but as a reference, the master server that is used on this project boots with 'sudo reboot' command in about 10 seconds, and SSH connection is immediately available as on some platforms it is possible for the random number generator's entropy to includes CPU's built-in generator mechanism. This kernel parameter is set on the master to enable the said function:

```
sed -i '/LINUX_DEF/c\GRUB_CMDLINE_LINUX_DEFAULT="loglevel=3 quiet random.trust_cpu=on"'
```

Minions are updated and booted manually for now with the commands used in pictures in chapter 5.8, and if a kernel or systemd update is observed, a reboot is also done.

```
sudo salt '*' cmd.run 'pacman -Suy --noconfirm'
sudo salt '*' cmd.run 'reboot'
```

https://askubuntu.com/questions/1070433/will-ubuntu-enable-random-trust-cpu-in-the-kernel-and-what-would-be-the-effect
https://wiki.debian.org/BoottimeEntropyStarvation

6 Hardening minions with SaltStack

Center for Internet Security (CIS) provides a distribution independent benchmark, which is being used here. As noted before, Arch Linux is not the most popular Linux distribution, and there are not ready made checklists specifically for it, but the DIY mentality suits the project's purpose perfectly.

Minions are hardened when executing the salt 'states', but we must not forget the master! There is a way to run the same configurations to the master by configuring itself as a minion, and it will be experimented at some point in the project.

A tree view of the configured states is presented in Appendix 6.

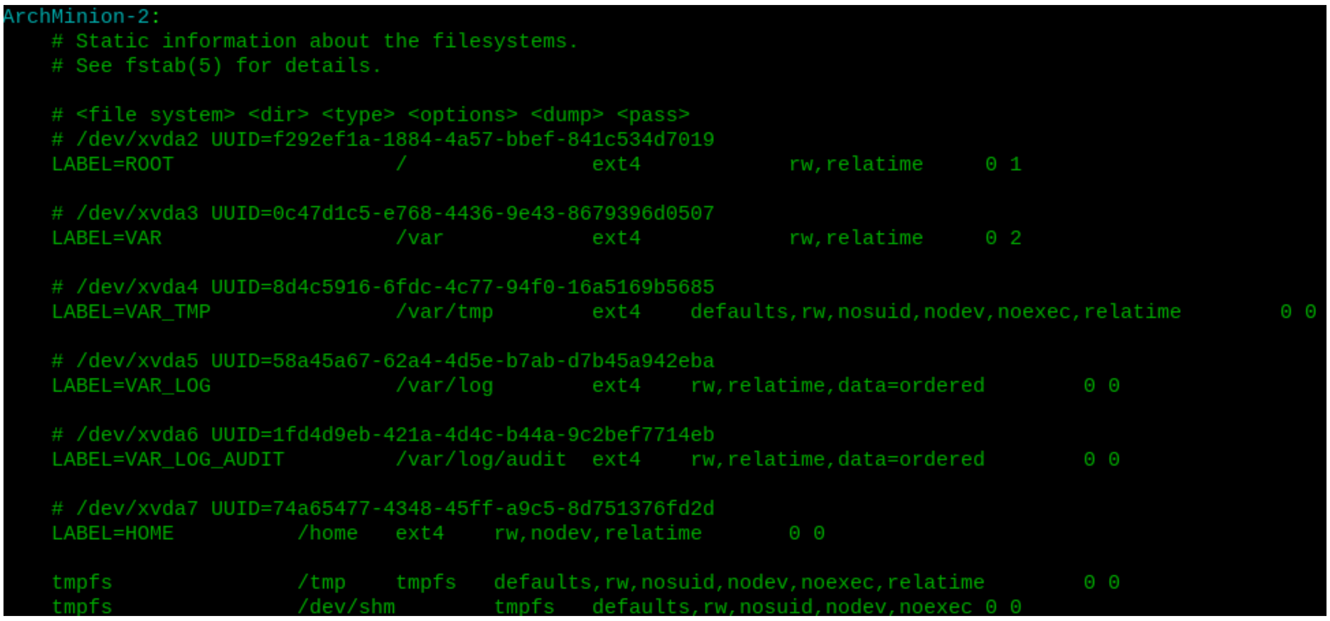
6.1 Filesystems

6.1.1 Partition configurations

Temporary data storages are often world-writable, and they can be configured to disallow code execution and device mounting. The configurations can be achieved via the filesystem table: '/etc/fstab'. This makes most sense to do in the installation phase, when the initial fstab -file is created. All temporary filesystems (tmpfs) should be mounted with noexec, nodev and nosuid options in fstab.

One special device, '/dev/shm', which is a shared memory virtual filesystem is userspace processes for faster temporary storage is also subject to these. Arch Linux defaults to nodev and nosuid, but doesn't enable the noexec option on default. After installation we can append the tmpfs configurations to fstab file with Salt.

The changes to mounting options were made according to the CIS recommendations 1.1.2-17, and here is an example of what a salt-minion's fstab file looks like after running the state:



```
ArchMinion-2:
    # Static information about the filesystems.
    # See fstab(5) for details.

    # <file system> <dir> <type> <options> <dump> <pass>
    # /dev/xvda2 UUID=f292ef1a-1884-4a57-bbef-841c534d7019
    LABEL=ROOT                  /               ext4            rw,relatime     0 1

    # /dev/xvda3 UUID=0c47d1c5-e768-4436-9e43-8679396d0507
    LABEL=VAR                   /var            ext4            rw,relatime     0 2

    # /dev/xvda4 UUID=8d4c5916-6fdc-4c77-94f0-16a5169b5685
    LABEL=VAR_TMP               /var/tmp        ext4    defaults,rw,nosuid,nodev,noexec,relatime        0 0

    # /dev/xvda5 UUID=58a45a67-62a4-4d5e-b7ab-d7b45a942eba
    LABEL=VAR_LOG               /var/log        ext4    rw,relatime,data=ordered        0 0

    # /dev/xvda6 UUID=1fd4d9eb-421a-4d4c-b44a-9c2bef7714eb
    LABEL=VAR_LOG_AUDIT         /var/log/audit  ext4    rw,relatime,data=ordered        0 0

    # /dev/xvda7 UUID=74a65477-4348-45ff-a9c5-8d751376fd2d
    LABEL=HOME          /home   ext4    rw,nodev,relatime       0 0

    tmpfs               /tmp    tmpfs   defaults,rw,nosuid,nodev,noexec,relatime        0 0
    tmpfs               /dev/shm        tmpfs   defaults,rw,nosuid,nodev,noexec 0 0
```

CIS p.37-69

6.1.2 Disable automounting

Automounting in Linuxes is handled by a program called 'autofs', which is usually included in Linux distributions. Arch Linux doesn't have it installed on default.

6.1.3 Removable device filesystems and udev rules

USB storage is unneeded on a cloud server and removable media partition mounts also have to be treated with the nosuid, nodev and noexec options. The whole USB subsystem could be disabled from the kernel. In this work, the same was achieved by installing the kernel module 'usb-storage' into '/bin/true' on startup. See chapter 6.1.5 for detailed explanation.

6.1.4 Filesystem sticky bit set on world-writable directories

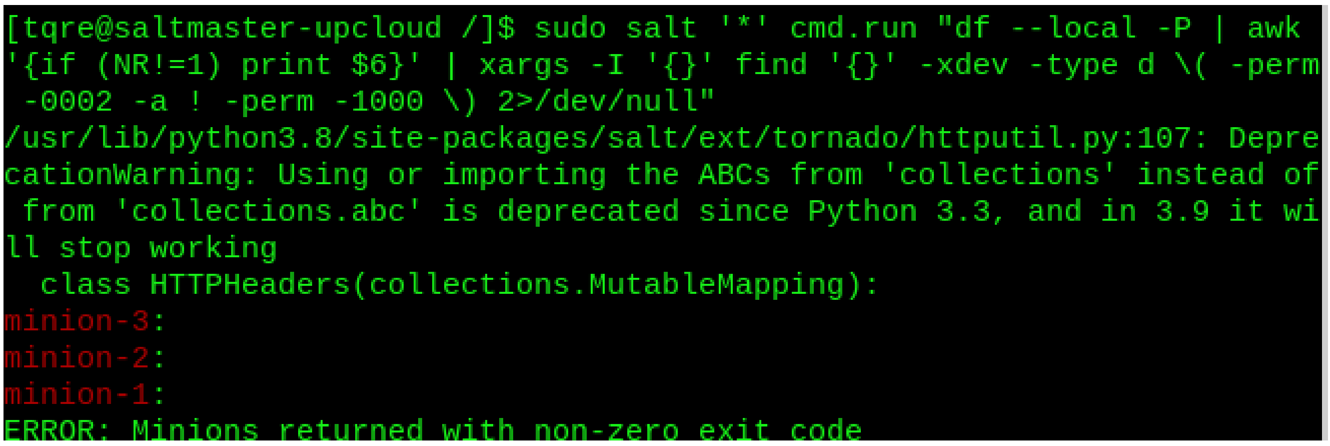
Sticky bit is an access right flag, and setting it on directories that are accessable to all users prevents other than a file's or directory's owner from deleting or renaming a file. Without the sticky bit it would be possible for any user to rename or delete files with directory write and execute permissions. Sticky bit is denoted as a 't' in unix-like permission notation, and is typically found on '/tmp' directory.

```
drwxrwxrwt   8 root root   160 May 16 11:27 tmp
```

CIS benchmarks provide a command to go through all world-writable directories, and return output if a world-writable directory is found without the sticky bit set.

```
df --local -P | awk '{if (NR!=1) print $6}' | xargs -I '{}' find '{}' -xdev -
type d \( -perm -0002 -a ! -perm -1000 \) 2>/dev/null
```

The command can be run as it is with salt:



```
[tqre@saltmaster-upcloud /]$ sudo salt '*' cmd.run "df --local -P | awk
'{if (NR!=1) print $6}' | xargs -I '{}' find '{}' -xdev -type d \( -perm
 -0002 -a ! -perm -1000 \) 2>/dev/null"
/usr/lib/python3.8/site-packages/salt/ext/tornado/httputil.py:107: Depre
cationWarning: Using or importing the ABCs from 'collections' instead of
 from 'collections.abc' is deprecated since Python 3.3, and in 3.9 it wi
ll stop working
  class HTTPHeaders(collections.MutableMapping):
minion-3:
minion-2:
minion-1:
ERROR: Minions returned with non-zero exit code
```

The non-zero exit code returned is related to salt's inner workings, as the state of the command is unknown to the state system. The command was executed correctly on minions, and the output is correct (no output expected).

https://docs.saltstack.com/en/master/ref/states/all/salt.states.cmd.html
CIS p.70

6.1.5 Disabling unused filesystems

The Linux kernel supports a wide variety of filesystems. The main rationale behind disabling unneeded filesystem support is to reduce the local attack surface of the system. There are two main approaches to do this:

- compile the Linux kernel without the filesystem support
- blacklist the kernel modules providing filesystem support

As the chosen distribution uses rolling release as it's update policy, compiling, testing and installing a new custom kernel about once a week takes too much resources in the scope of this study. Blacklisting the kernel modules is relatively easy, and configuration management can be used to accomplish this.

Kernel modules can be blacklisted by creating a configuration -file in '/etc/modprobe.d'. These files support a blacklisting keyword, but that is not enough. Blacklisting the kernel modules only prevent the modules loading on initial startup, but if a need ror a dependency rises for the module it gets loaded. This is traditionally circumvented by installing the module to '/bin/true'. This instructs a custom command (/bin/true) to be run instead of inserting the module into kernel, resulting module loading fail.

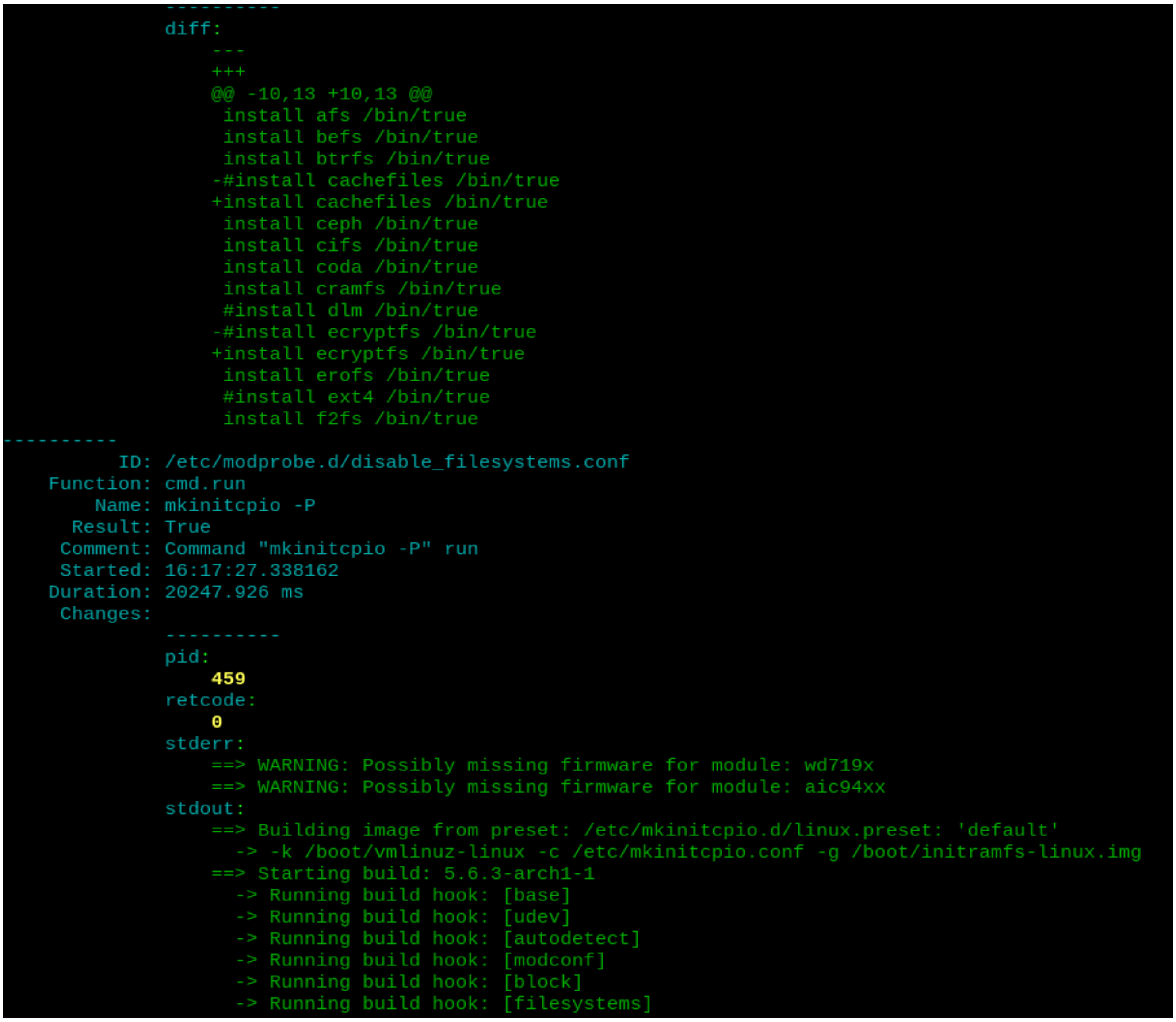
We have to find out what modules we actually have on the system. The installed filesystem kernel modules are in '/lib/modules/<kernel version>/kernel/fs'. Most of the modules can be blacklisted, but some are essential to the system. Reading Linux kernel documentation and after some tests, a list of filesystem modules to keep shrinks considerably. In the following table some of the special kernel modules are explained:

Filesystem module	**Enabled**	**Function**
cachefiles	No, enable if needed	Userspace cache management for SELinux/AppArmor
dlm	No	Distributed locking manager for OCFS2 filesystem
ext4	Yes	The default native Linux filesystem
jbd2	Yes	Filesystem journal for file transactions
lockd	No, enable if needed	File locks for networking filesystems (nfd)
nls	No	Native language support for some filesystems
quota	Yes	User specific disk quotas

Important thing to note here, is that to fully implement the blacklist, two additional steps are required. modprobe.d is read on boot, and a new initramfs has to be recreated, so the salt-state needs a hook to regenerate them:

```
/etc/modprobe.d/disable_filesystems.conf:
  file.managed:
    - source: salt://managed_files/etc/modprobe.d/disable_filesystems.conf
  cmd.run:
    - name: mkinitcpio -P
    - onchanges:
      - file: /etc/modprobe.d/disable_filesystems.conf
```

A reboot is also needed after creating a new initramfs.



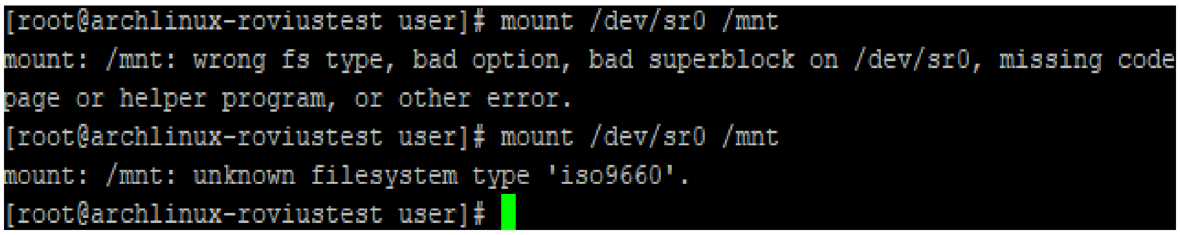
```
              ----------
              diff:
                  ---
                  +++
                  @@ -10,13 +10,13 @@
                   install afs /bin/true
                   install befs /bin/true
                   install btrfs /bin/true
                  -#install cachefiles /bin/true
                  +install cachefiles /bin/true
                   install ceph /bin/true
                   install cifs /bin/true
                   install coda /bin/true
                   install cramfs /bin/true
                   #install dlm /bin/true
                  -#install ecryptfs /bin/true
                  +install ecryptfs /bin/true
                   install erofs /bin/true
                   #install ext4 /bin/true
                   install f2fs /bin/true
----------
          ID: /etc/modprobe.d/disable_filesystems.conf
    Function: cmd.run
        Name: mkinitcpio -P
      Result: True
     Comment: Command "mkinitcpio -P" run
     Started: 16:17:27.338162
    Duration: 20247.926 ms
     Changes:
              ----------
              pid:
                  459
              retcode:
                  0
              stderr:
                  ==> WARNING: Possibly missing firmware for module: wd719x
                  ==> WARNING: Possibly missing firmware for module: aic94xx
              stdout:
                  ==> Building image from preset: /etc/mkinitcpio.d/linux.preset: 'default'
                    -> -k /boot/vmlinuz-linux -c /etc/mkinitcpio.conf -g /boot/initramfs-linux.img
                  ==> Starting build: 5.6.3-arch1-1
                    -> Running build hook: [base]
                    -> Running build hook: [udev]
                    -> Running build hook: [autodetect]
                    -> Running build hook: [modconf]
                    -> Running build hook: [block]
                    -> Running build hook: [filesystems]
```

https://www.kernel.org/doc/Documentation/filesystems/

https://www.kernel.org/doc/html/latest/filesystems/ext4/journal.html?highlight=jbd2

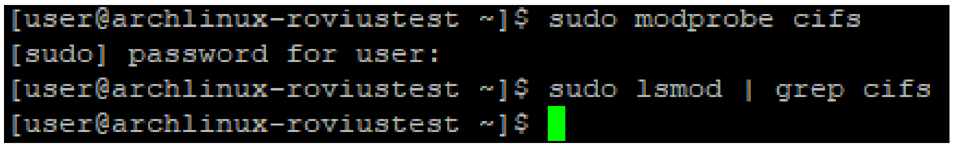
6.1.5.1 Testing if a filesystem is disabled

The filesystem blacklisting functionality is tested with the following procedure:
- reboot the virtual machine and attach a virtual ISO image
- try to mount the ISO image



```
[root@archlinux-roviustest user]# mount /dev/sr0 /mnt
mount: /mnt: wrong fs type, bad option, bad superblock on /dev/sr0, missing code
page or helper program, or other error.
[root@archlinux-roviustest user]# mount /dev/sr0 /mnt
mount: /mnt: unknown filesystem type 'iso9660'.
[root@archlinux-roviustest user]#
```

Testing a manual module insertion with 'modprobe' command:



```
[user@archlinux-roviustest ~]$ sudo modprobe cifs
[sudo] password for user:
[user@archlinux-roviustest ~]$ sudo lsmod | grep cifs
[user@archlinux-roviustest ~]$
```

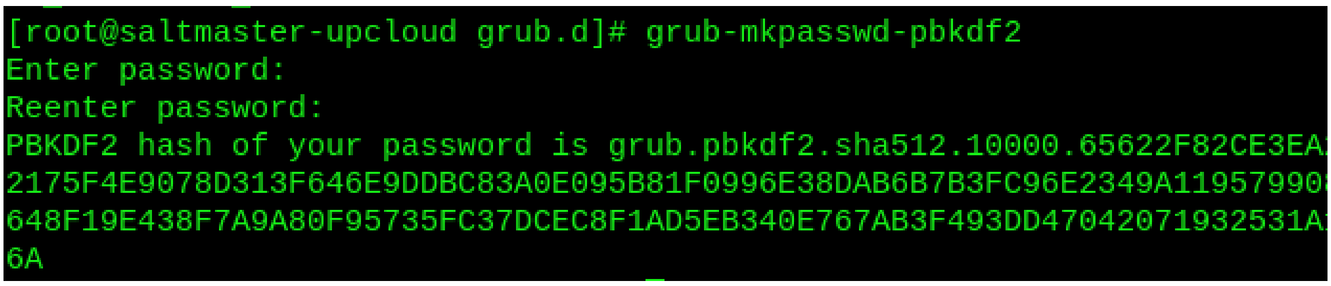
Once more it is good to remind that with an access to the cloud management interface, it is possible to boot the virtual machines with any ISO and gain access that way into the filesystem.

6.2 Bootloader configurations

GRUB bootloader is found on most of the Linuxes today. Securing boot loader prevents unauthorized users from changing the boot parameters on the server. The configuration file '/boot/grub/grub.cfg' is set with restrictive permissions, and the bootloader menu is configured with a password in such a way, that the users are not allowed to make edits to the boot entries and don't have an access to the grub command console.

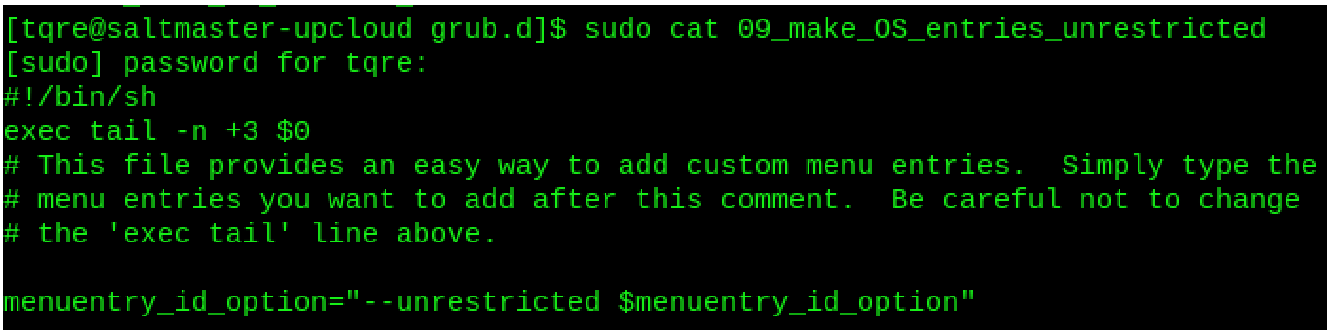
In the installation bootloader menu timeout was set to zero, which means the menu is not shown when the machine boots. In the cloud, only possible access to the boot loader menu is anyway via the cloud providers' console connection. This setting might make more sense in a desktop system, but it is included here as it includes setting up a password to yet another component.

GRUB provides a way to generate a password hash, and uses PBKDF2 as a key derivation function. The function uses a cryptographic technique known as key stretching, and it makes password cracking from the hash computationally more expensive. PBKDF2 relies on generating a derived key from the user's supplied password, but a good and long password is needed nevertheless, as large password dictionaries are available online.



```
[root@saltmaster-upcloud grub.d]# grub-mkpasswd-pbkdf2
Enter password:
Reenter password:
PBKDF2 hash of your password is grub.pbkdf2.sha512.10000.65622F82CE3EA
2175F4E9078D313F646E9DDBC83A0E095B81F0996E38DAB6B7B3FC96E2349A11957990
648F19E438F7A9A80F95735FC37DCEC8F1AD5EB340E767AB3F493DD47042071932531A
6A
```

To avoid locking the bootloader completely, the menu entries have to be set with '--unrestricted' flag. There is a risk for locking down the system completely if the settings are done to a file that gets updated in the updating process.



```
[tqre@saltmaster-upcloud grub.d]$ sudo cat 09_make_OS_entries_unrestricted
[sudo] password for tqre:
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.

menuentry_id_option="--unrestricted $menuentry_id_option"
```

The salt state then manages this and the password configuration file.

These configurations are however rendered useless, if the machine can be booted from an external drive or equivalent. In cloud context taking advantage of this requires access again to the cloud provider's management interface.

This hardening step is somewhat questionable, as implemeting security controls and additional passwords where they are not needed might make the whole of the system more vulnerable. Also the risk of locking down the system completely makes the usage of these configurations worth considering in more relaxed environments.

https://wiki.archlinux.org/index.php/Talk:GRUB/Tips_and_tricks
https://wiki.archlinux.org/index.php/GRUB/Tips_and_tricks#Password_protection_of_GRUB_edit_and_console_options_only

6.3 Network configurations

6.3.1 Process filesystem /proc

In Linux systems, sysctl is a software that is used to control various parameters in the running kernel. The parameters are readable from the process filesystem in '/proc'. It does not contain regular files, but abstractions of realtime system information. It is essentially a controlling centre for the kernel, and reading and writing the files in '/proc' result in changes to the running system parameters.

Networking parameters are found from '/proc/sys/net', and sysctl can be used to control for example to control IP traffic forwarding. For example, the following command enables IPv4 forwarding on all interfaces:

```
sysctl -w net.ipv4.ip_forward=1
```

To demonstrate the file abstractions in '/proc', the following command does exactly the same:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

These settings can be controlled from '/etc/sysctl.d/' -directory on startup. In this work, these settings are managed with a distributed file '/etc/sysctl.d/network.conf', and the parameters are also checked with separate Salt states.

https://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html

6.3.2 Networking parameters

Many of the configurations ensure that the hardened machine can not be used as a router, and disable various functionalities that a router needs to perform. A compromised system could be used to corrupt local routing tables. Most interesting items of the configuration are discussed.

Source routing used to be a diagnosis feature on networks, and it can be used to circumvent firewalls or even VLAN segregating by explicitly defining routes or partial routes the packets must take. This is ensure to be disabled.

Enabling reverse path filtering makes sure that if a system has multiple network interfaces configured, packets are not forwarded to toher interfaces and logged with log_martians setting. Reverse path filtering cannot however be used on networks that use dynamic routing protocols, such as BGP or OSPF.

Enabling TCP SYN cookies makes a system resistant to SYN flood attacks. These attacks rely on the fact that SYN queue on the victim system is filled by sending continuous connection requests that are never filled. When the queue is full, the victim cannot accept any more connections resulting in denial-of-service. SYN cookies make use of a special TCP sequence number, and allows the system to verify a legitimate connection attempt even if the SYN queue is full. This allows a connection to a system that is under a SYN flood attack.

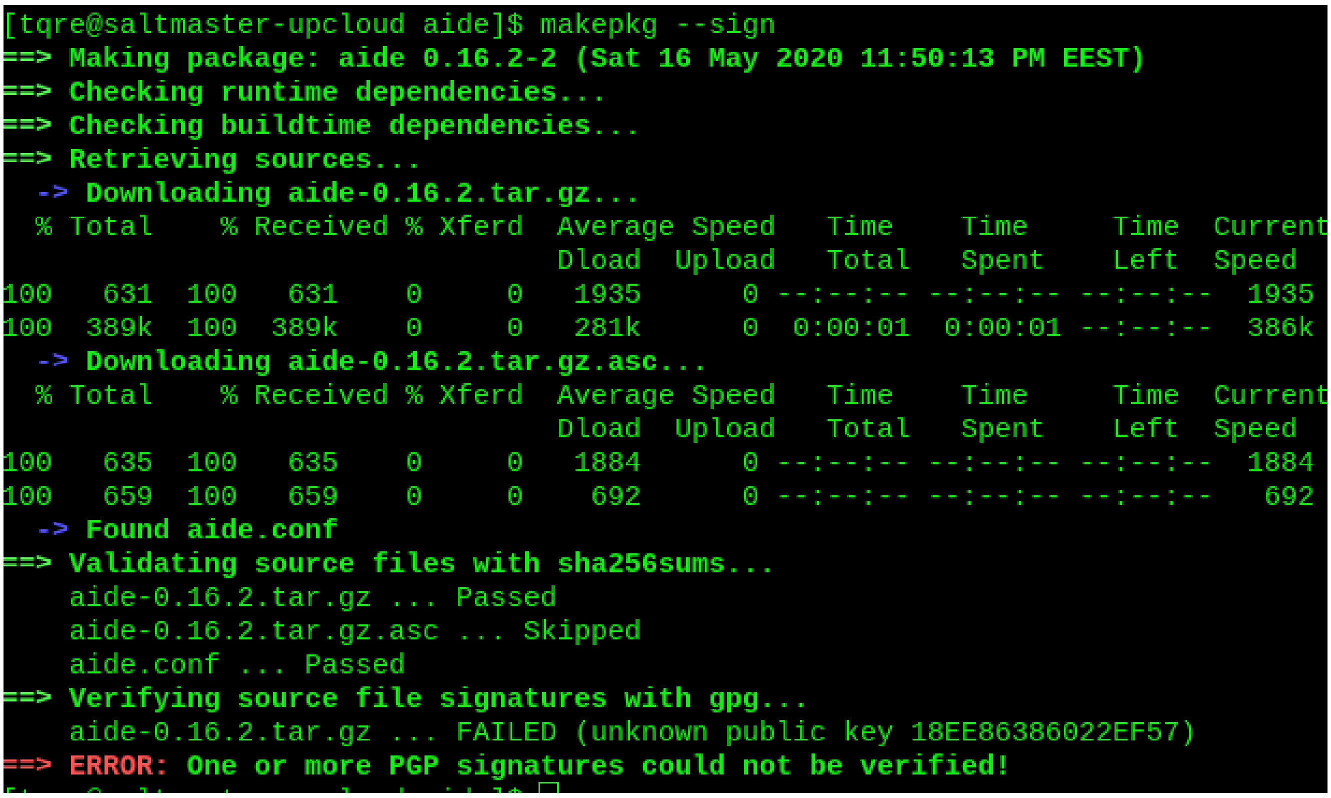
CIS, p.222-248
https://tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.kernel.rpf.html
https://nmap.org/book/firewall-subversion.html
https://nmap.org/book/firewall-subversion.html#fw-subversion-example

7 AIDE - Advanced Intrusion Detection Environment

An intrusion detection system recommended by CIS is an orphaned package in Arch User Repository (AUR). It is actually a file and directory integrity checker based on message digest algortithms, also known as hashes. It constructs a database of files and their attributes, and can be configured to check regularly on inconsistencies.

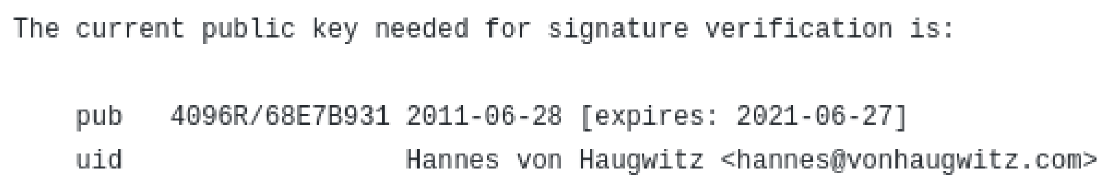
As AIDE is not in the official repositories, we are going to do the same as we did for SaltStack. Package the file on the master server, sign it, and distribute it to minions with the fileserver that has been set up.

```
git clone https://aur.archlinux.org/aide.git
pacman -S mhash
makepkg --sign
```



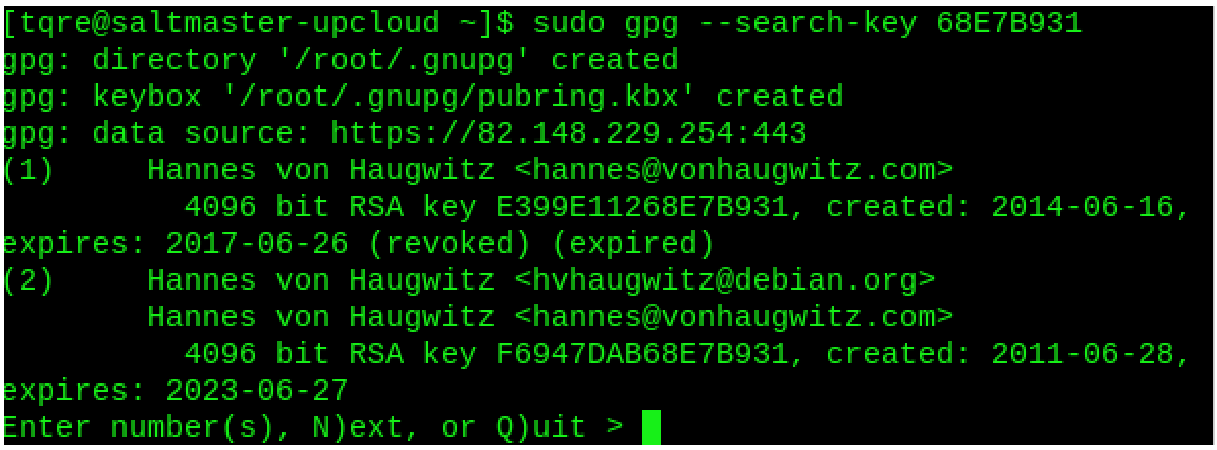
```
[tqre@saltmaster-upcloud aide]$ makepkg --sign
==> Making package: aide 0.16.2-2 (Sat 16 May 2020 11:50:13 PM EEST)
==> Checking runtime dependencies...
==> Checking buildtime dependencies...
==> Retrieving sources...
  -> Downloading aide-0.16.2.tar.gz...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   631  100   631    0     0   1935      0 --:--:-- --:--:-- --:--:--  1935
100  389k  100  389k    0     0   281k      0  0:00:01  0:00:01 --:--:--  386k
  -> Downloading aide-0.16.2.tar.gz.asc...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   635  100   635    0     0   1884      0 --:--:-- --:--:-- --:--:--  1884
100   659  100   659    0     0    692      0 --:--:-- --:--:-- --:--:--   692
  -> Found aide.conf
==> Validating source files with sha256sums...
    aide-0.16.2.tar.gz ... Passed
    aide-0.16.2.tar.gz.asc ... Skipped
    aide.conf ... Passed
==> Verifying source file signatures with gpg...
    aide-0.16.2.tar.gz ... FAILED (unknown public key 18EE86386022EF57)
==> ERROR: One or more PGP signatures could not be verified!
```

This package is signed by it's current maintainer, so we have to add him to our trusted keyring. First we'll verify the key, and cross-check it:



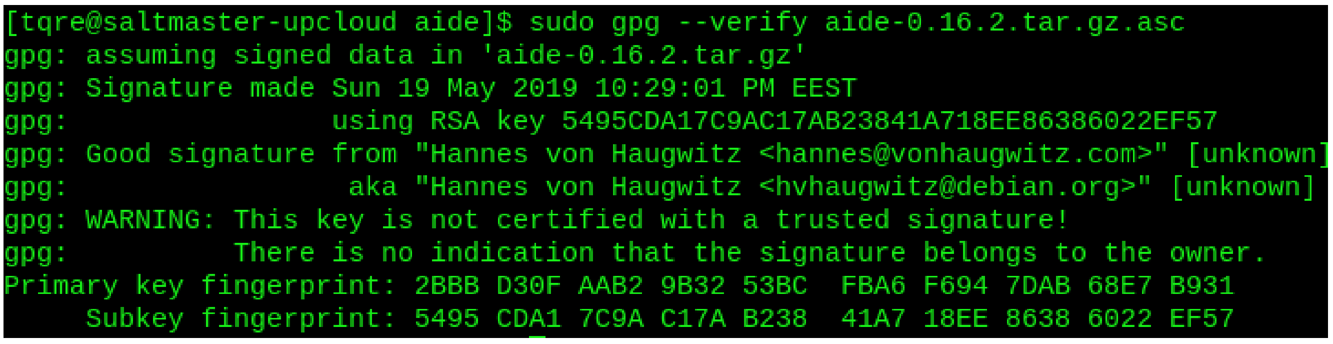
```
The current public key needed for signature verification is:

    pub   4096R/68E7B931 2011-06-28 [expires: 2021-06-27]
    uid                  Hannes von Haugwitz <hannes@vonhaugwitz.com>
```

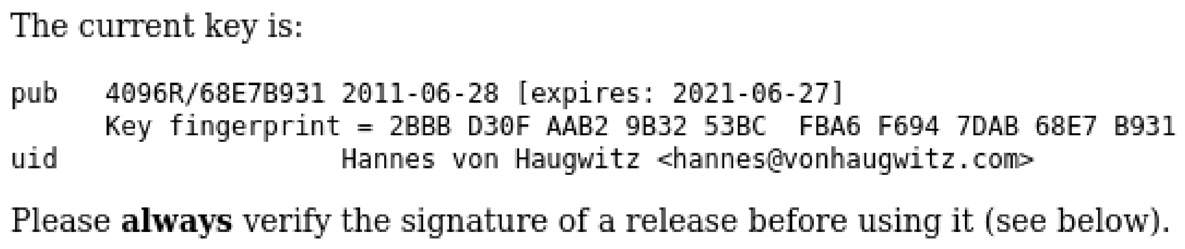
First we have to import the key from a public keyserver:



```
[tqre@saltmaster-upcloud ~]$ sudo gpg --search-key 68E7B931
gpg: directory '/root/.gnupg' created
gpg: keybox '/root/.gnupg/pubring.kbx' created
gpg: data source: https://82.148.229.254:443
(1)     Hannes von Haugwitz <hannes@vonhaugwitz.com>
          4096 bit RSA key E399E11268E7B931, created: 2014-06-16,
expires: 2017-06-26 (revoked) (expired)
(2)     Hannes von Haugwitz <hvhaugwitz@debian.org>
        Hannes von Haugwitz <hannes@vonhaugwitz.com>
          4096 bit RSA key F6947DAB68E7B931, created: 2011-06-28,
expires: 2023-06-27
Enter number(s), N)ext, or Q)uit >
```

Next we can verify it, and compare the fingerprints:


```
[tqre@saltmaster-upcloud aide]$ sudo gpg --verify aide-0.16.2.tar.gz.asc
gpg: assuming signed data in 'aide-0.16.2.tar.gz'
gpg: Signature made Sun 19 May 2019 10:29:01 PM EEST
gpg:                using RSA key 5495CDA17C9AC17AB23841A718EE86386022EF57
gpg: Good signature from "Hannes von Haugwitz <hannes@vonhaugwitz.com>" [unknown]
gpg:                 aka "Hannes von Haugwitz <hvhaugwitz@debian.org>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 2BBB D30F AAB2 9B32 53BC  FBA6 F694 7DAB 68E7 B931
     Subkey fingerprint: 5495 CDA1 7C9A C17A B238  41A7 18EE 8638 6022 EF57
```


The current key is:

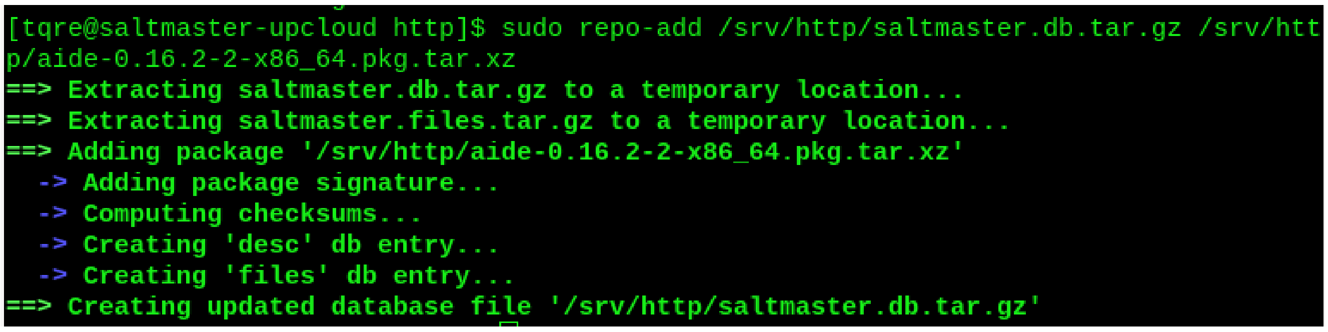
```
pub   4096R/68E7B931 2011-06-28 [expires: 2021-06-27]
      Key fingerprint = 2BBB D30F AAB2 9B32 53BC  FBA6 F694 7DAB 68E7 B931
uid                  Hannes von Haugwitz <hannes@vonhaugwitz.com>
```

Please **always** verify the signature of a release before using it (see below).

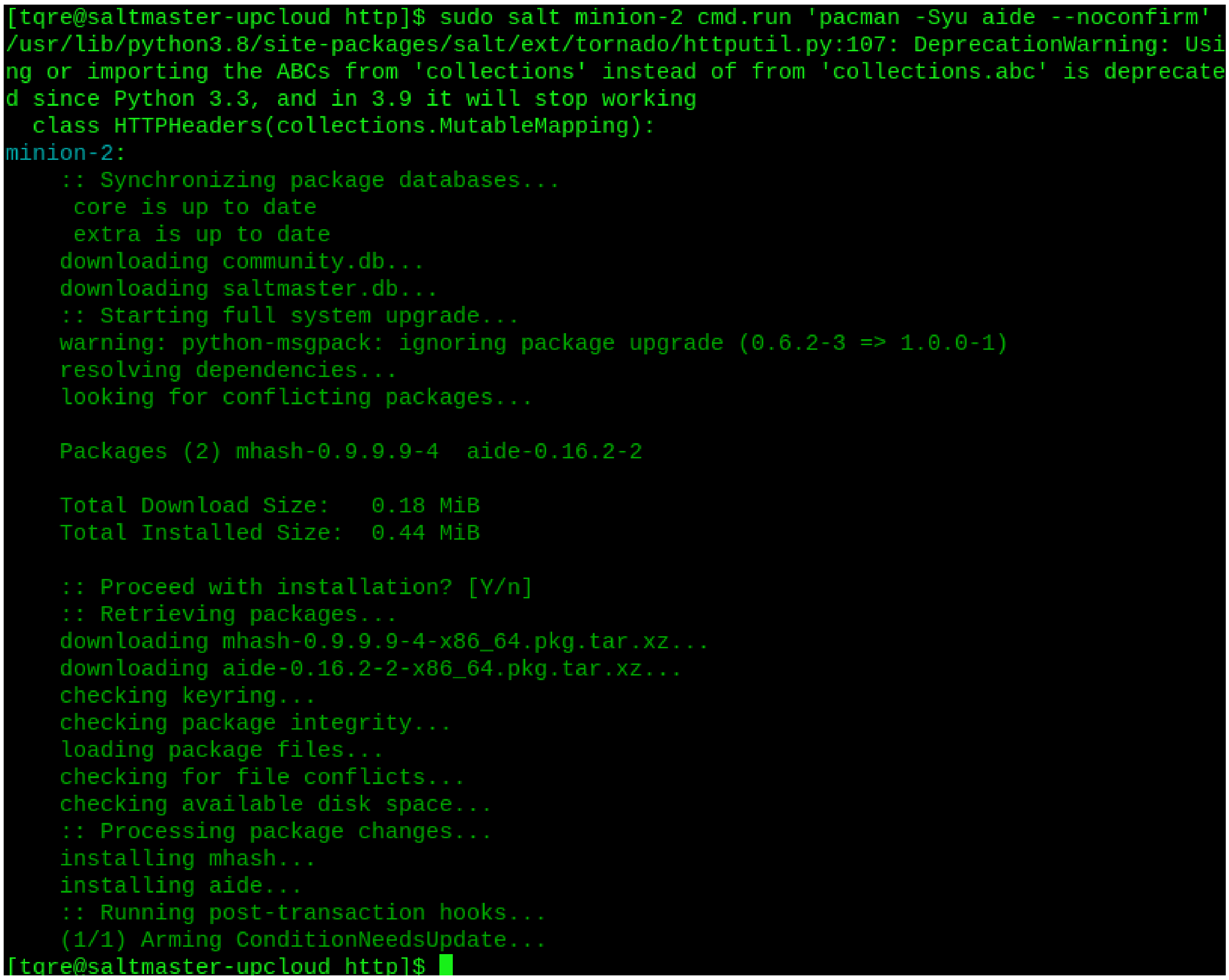
They seem to match. However, we had to use sudo earlier to import the key, and the regular user doesn't have it. I'll import it with ascii armor from the root user. For some reason the keyserver didn't accept normal user queries.

```
sudo gpg --export --armor hannes@vonhaugwitz.com > aide.asc
gpg --import aide.asc
makepkg --sign
```

We are once again asked our own GPG password to sign the package, and we'll put it inside the fileserver directory, and update the package database file as we did in chapter 4.11.


```
[tqre@saltmaster-upcloud http]$ sudo repo-add /srv/http/saltmaster.db.tar.gz /srv/htt
p/aide-0.16.2-2-x86_64.pkg.tar.xz
==> Extracting saltmaster.db.tar.gz to a temporary location...
==> Extracting saltmaster.files.tar.gz to a temporary location...
==> Adding package '/srv/http/aide-0.16.2-2-x86_64.pkg.tar.xz'
  -> Adding package signature...
  -> Computing checksums...
  -> Creating 'desc' db entry...
  -> Creating 'files' db entry...
==> Creating updated database file '/srv/http/saltmaster.db.tar.gz'
```

Finally, testing to install aide on one of the minions:



```
[tqre@saltmaster-upcloud http]$ sudo salt minion-2 cmd.run 'pacman -Syu aide --noconfirm'
/usr/lib/python3.8/site-packages/salt/ext/tornado/httputil.py:107: DeprecationWarning: Usi
ng or importing the ABCs from 'collections' instead of from 'collections.abc' is deprecate
d since Python 3.3, and in 3.9 it will stop working
  class HTTPHeaders(collections.MutableMapping):
minion-2:
    :: Synchronizing package databases...
     core is up to date
     extra is up to date
    downloading community.db...
    downloading saltmaster.db...
    :: Starting full system upgrade...
    warning: python-msgpack: ignoring package upgrade (0.6.2-3 => 1.0.0-1)
    resolving dependencies...
    looking for conflicting packages...

    Packages (2) mhash-0.9.9.9-4  aide-0.16.2-2

    Total Download Size:   0.18 MiB
    Total Installed Size:  0.44 MiB

    :: Proceed with installation? [Y/n]
    :: Retrieving packages...
    downloading mhash-0.9.9.9-4-x86_64.pkg.tar.xz...
    downloading aide-0.16.2-2-x86_64.pkg.tar.xz...
    checking keyring...
    checking package integrity...
    loading package files...
    checking for file conflicts...
    checking available disk space...
    :: Processing package changes...
    installing mhash...
    installing aide...
    :: Running post-transaction hooks...
    (1/1) Arming ConditionNeedsUpdate...
[tqre@saltmaster-upcloud http]$
```

https://aide.github.io/

8 Final words -

The scope of the work was large, and it turned out to expand in detail. It became apparent that this kind of detailed hardening is not a small job, and even small individual details can take a lot of time to understand and implement. The lack of a proper threat model made the work scope unclear, and a general hardened operating system is difficult to build without usage context. The practical use of a system with many hardenings done would be the real test to this work, as it is possible to harden system unusable, or difficult to configure after extensive hardening.

The work done was by no means futile, even if it was not complete. The practical hardenings that were done, were found functioning, and the author was familirized with mechanisms that can be used to harden (or exploit) security configurations in Linux systems.

Additional material

References are mentioned in the chapters. As this is not a thesis, strict reference policy was not heeded.

Mastering Linux Security and Hardening - Second Edition
Donald A. Tevault, Packt Publishing 2020

Linux Hardening in Hostile Networks
Kyle Rankin, Pearson Education 2018

NIST Computer Security Resoruce Center
https://csrc.nist.gov/Projects/United-States-Government-Configuration-Baseline

Michael Zurawski: The Practical Linux Hardening Guide
https://trimstray.github.io/the-practical-linux-hardening-guide/
https://github.com/trimstray/the-practical-linux-hardening-guide

Open Security Content Automation Protocol
https://www.open-scap.org/

Guide to the Secure Configuration of Red Hat Enterprise Linux 7
https://static.open-scap.org/ssg-guides/ssg-rhel7-guide-C2S.html

Security-Enhanced Linux Project
https://github.com/SELinuxProject/
https://www.nsa.gov/what-we-do/research/selinux/

A tool for checking the hardening options in the Linux kernel config
https://github.com/a13xp0p0v/kconfig-hardened-check

Center for Internet Security: benchmarks
https://www.cisecurity.org/cis-benchmarks

NIST Special Publication 800-123: Guide to General Server Security; pub. July 2008
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-123.pdf

ArchWiki: Arch Linux documentation
https://wiki.archlinux.org/index.php/Main_page

http://www.linuxfromscratch.org/lfs/view/8.4-systemd/chapter06/shadow.html

https://wiki.gentoo.org/wiki/Security_Handbook/
https://github.com/konstruktoid/hardening

Appendices

Appendix 1. Partitioning script

```
# Format, label and mount partitions
# Needs the device file extension as an argument.
# Note that $11 transforms to {$1}1 -> xvda1
# The argument get passed the device file: xvda, vda, sda, sdb...

# Format the partitions
mkfs.vfat /dev/$11
mkfs.ext4 /dev/$12
mkfs.ext4 /dev/$13
mkfs.ext4 /dev/$14
mkfs.ext4 /dev/$15
mkfs.ext4 /dev/$16
mkfs.ext4 /dev/$17

# Label the partitions for easier Salt handling
tune2fs -L ROOT /dev/$12
tune2fs -L VAR /dev/$13
tune2fs -L VAR_TMP /dev/$14
tune2fs -L VAR_LOG /dev/$15
tune2fs -L VAR_LOG_AUDIT /dev/$16
tune2fs -L HOME /dev/$17

# Mount and make directories for the partitions
mount /dev/$12 /mnt

mkdir /mnt/var
mount /dev/$13 /mnt/var

mkdir /mnt/var/tmp
mkdir /mnt/var/log
mount /dev/$14 /mnt/var/tmp
mount /dev/$15 /mnt/var/log

mkdir /mnt/var/log/audit
mount /dev/$16 /mnt/var/log/audit

mkdir /mnt/home
mount /dev/$17 /mnt/home
```

Appendix 2. Partitioning schemes

Salt-master:

```
label: gpt
device: /dev/vda
unit: sectors
first-lba: 2048
last-lba: 104857566

/dev/vda1 : start=        2048, size=        2048, type=21686148-6449-6E6F-744E-656564454649
/dev/vda2 : start=        4096, size=    16777216, type=4F68BCE3-E8CD-4DB1-96E7-FBCAF984B709
/dev/vda3 : start=    16781312, size=    33554432, type=0FC63DAF-8483-4772-8E79-3D69D8477DE4
/dev/vda4 : start=    50335744, size=    16777216, type=0FC63DAF-8483-4772-8E79-3D69D8477DE4
/dev/vda5 : start=    67112960, size=    16777216, type=0FC63DAF-8483-4772-8E79-3D69D8477DE4
/dev/vda6 : start=    83890176, size=    16777216, type=0FC63DAF-8483-4772-8E79-3D69D8477DE4
/dev/vda7 : start=   100667392, size=     4190175, type=933AC7E1-2EB4-4F13-B844-0E14E2AEF915
```

Salt-minion:

```
label: gpt
device: /dev/xvda
unit: sectors
first-lba: 2048
last-lba: 67108830

/dev/xvda1 : start=        2048, size=        2048, type=21686148-6449-6E6F-744E-656564454649
/dev/xvda2 : start=        4096, size=    16777216, type=4F68BCE3-E8CD-4DB1-96E7-FBCAF984B709
/dev/xvda3 : start=    16781312, size=    16777216, type=0FC63DAF-8483-4772-8E79-3D69D8477DE4
/dev/xvda4 : start=    33558528, size=     4194304, type=0FC63DAF-8483-4772-8E79-3D69D8477DE4
/dev/xvda5 : start=    37752832, size=     4194304, type=0FC63DAF-8483-4772-8E79-3D69D8477DE4
/dev/xvda6 : start=    41947136, size=     4194304, type=0FC63DAF-8483-4772-8E79-3D69D8477DE4
/dev/xvda7 : start=    46141440, size=    20967391, type=933AC7E1-2EB4-4F13-B844-0E14E2AEF915
```

Appendix 3. Arch Linux installation script for salt-master

```
#!/bin/bash

# USE WITH CARE! The script partitions hard drive according to
# partition_map -file with no questions asked. I take no responsibility
# if you delete your data. Always keep backups.

# Tested and working  on UpCloud
HD_DEVICE=vda
timedatectl set-ntp true

# General settings:
TZ="Europe/Helsinki"
LOC="en_US.UTF-8"
KEYBOARD="us"
HOSTNAME="saltmaster"
SUDOUSER="user"
SSH_PORT="22"
SSH_PUB_KEY=$(cat id_rsa.pub)

# Disk partitioning, formatting and mounting
sfdisk /dev/$HD_DEVICE < partitions/partition_map_upcloud_50
chmod +x partitions/prepare.sh
partitions/prepare.sh $HD_DEVICE

# Overwrite the installation ISO mirrorlist with a supplied one as it gets
# copied over to the new installation in the process.
cat mirrorlist > /etc/pacman.d/mirrorlist

# Main install command - bootstrap Arch Linux
pacstrap /mnt base base-devel linux grub openssh \
       nano wget git ufw nginx-mainline \
       python python-jinja python-yaml python-markupsafe python-requests \
       python-pyzmq python-m2crypto python-systemd python-distro \
       python-pycryptodomex

# Create file system table:
genfstab -L /mnt >> /mnt/etc/fstab

# Settings: here-document is piped to chroot
cat << EOF | arch-chroot /mnt
ln -sf /usr/share/zoneinfo/$TZ /etc/localtime
hwclock --systohc
sed -i 's/#$LOC/$LOC/' /etc/locale.gen
locale-gen
echo "LANG=$LOC" > /etc/locale.conf
echo "KEYMAP=$KEYBOARD" > /etc/vconsole.conf
echo "$SUDOUSER ALL=(ALL) ALL" >> /etc/sudoers

echo -e "[Match]\nName=ens*\n\n[Network]\nDHCP=true" > /etc/systemd/network/dhcp.network
systemctl enable systemd-networkd.service

echo $HOSTNAME > /etc/hostname
echo -e "127.0.0.1 localhost\n::1 localhost" > /etc/hosts

useradd -m $SUDOUSER
runuser $SUDOUSER -c 'mkdir ~/.ssh'
runuser $SUDOUSER -c 'echo $SSH_PUB_KEY > ~/.ssh/authorized_keys'

# SSH configuration
sed -i "s/#Port 22/Port $SSH_PORT/" /etc/ssh/sshd_config
# Disable sftp subsystem
sed -i 's/Subsystem/#Subsystem/' /etc/ssh/sshd_config
sed -i 's/#PasswordAuthentication yes/PasswordAuthentication no/' /etc/ssh/sshd_config
sed -i '/#PermitRootLogin pro/c\PermitRootLogin no' /etc/ssh/sshd_config
```

```
systemctl enable sshd

# GRUB installation
grub-install --target=i386-pc /dev/$HD_DEVICE
sed -i 's/GRUB_TIMEOUT=5/GRUB_TIMEOUT=0/' /etc/default/grub
sed -i '/LINUX_DEF/c\GRUB_CMDLINE_LINUX_DEFAULT="loglevel=3 quiet random.trust_cpu=on"'
/etc/default/grub
grub-mkconfig -o /boot/grub/grub.cfg

# Salt-py3 dependency installation:
sed -i 's/#IgnorePkg   =/IgnorePkg   = python-msgpack/' /etc/pacman.conf
wget -P /var/cache/pacman/pkg \
        https://archive.archlinux.org/packages/p/python-msgpack/python-msgpack-0.6.2-3-
x86_64.pkg.tar.xz
pacman -U --noconfirm /var/cache/pacman/pkg/python-msgpack-0.6.2-3-x86_64.pkg.tar.xz

EOF

# Copy existing DNS settings to new installation
cp /etc/resolv.conf /mnt/etc/resolv.conf

echo
echo To finish the installation, set the password for the user account:
echo Enter the chroot environment with:
echo   arch-chroot /mnt
echo And set the password with:
echo   passwd $SUDOUSER
```

Appendix 4. Nginx configuration file

```
user http;
worker_processes 1;
worker_rlimit_nofile 65535;

events {
	multi_accept on;
	worker_connections 1024;
}

http {
	charset utf-8;
	sendfile on;
	tcp_nopush on;
	tcp_nodelay on;
	server_tokens off;
	log_not_found off;
	types_hash_max_size 4096;

	# MIME
	include mime.types;
	default_type application/octet-stream;

	# logging
	access_log /var/log/nginx/access.log;
	error_log /var/log/nginx/error.log warn;

	# SSL
	ssl_session_timeout 5m;
	ssl_session_cache off;
	ssl_session_tickets off;

	# Mozilla Modern configuration
	ssl_protocols TLSv1.3;

	server {
		listen 443 ssl;

		server_name saltmaster;
		root /srv/http;

		# SSL
		ssl_certificate ssl/saltmaster.crt;
		ssl_certificate_key ssl/saltmaster.key;

		# security headers
		add_header X-Frame-Options "SAMEORIGIN" always;
		add_header X-XSS-Protection "1; mode=block" always;
		add_header X-Content-Type-Options "nosniff" always;
		add_header Referrer-Policy "no-referrer" always;
		add_header Content-Security-Policy "default-src 'self' http: https: 
data: blob: 'unsafe-inline'" always;
		add_header Strict-Transport-Security "max-age=31536000; 
includeSubDomains; preload" always;
	}
}
```

Appendix 5. Arch Linux installation script for salt-minion

```
#!/bin/bash

# USE WITH CARE! The script partitions hard drive according to
# partition_map -file with no questions asked. I take no responsibility
# if you delete your data. Always keep backups.

# Tested and working on Rovius Cloud Platform

# Settings:
HD_DEVICE=xvda
timedatectl set-ntp true

# General settings:
TZ="Europe/Helsinki"
LOC="en_US.UTF-8"
KEYBOARD="us"
SUDOUSER="user"
PASSWORD="user"
GPG_KEYID=""

# Salt settings
MASTER_IP="<ip-address>"
FILESERVER_PORT="80"
M_PORT="4506"
P_PORT="4505"
HOSTNAME="minion-"

# Disk partitioning, formatting, labelling and mounting
sfdisk /dev/$HD_DEVICE < partitions/partition_map_rovius_32
chmod +x partitions/prepare.sh
partitions/prepare.sh $HD_DEVICE

# Overwrite the installation ISO mirrorlist with a supplied one as it gets
# copied over to the new installation in the process.
cat mirrorlist > /etc/pacman.d/mirrorlist

# Main install command - bootstrap Arch Linux
pacstrap /mnt base linux grub openssh sudo nano wget python \
    python-jinja python-yaml python-markupsafe python-requests python-pyzmq \
    python-m2crypto python-systemd python-distro python-pycryptodomex

# Create filesystem table:
genfstab -L /mnt >> /mnt/etc/fstab

# Copy the saltmaster certificate to the new machine
cp saltmaster.crt /mnt/etc/ssl/private/saltmaster.crt

# Chroot setup:
# here-document is piped to chroot
cat << EOF | arch-chroot /mnt

ln -sf /usr/share/zoneinfo/$TZ /etc/localtime
hwclock --systohc
sed -i 's/#$LOC/$LOC/' /etc/locale.gen
locale-gen
echo "LANG=$LOC" > /etc/locale.conf
echo "KEYMAP=$KEYBOARD" > /etc/vconsole.conf
echo "$SUDOUSER ALL=(ALL) ALL" >> /etc/sudoers

echo -e "[Match]\nName=eth0\n\n[Network]\nDHCP=true" > /etc/systemd/network/dhcp.network
systemctl enable systemd-networkd.service

echo $HOSTNAME > /etc/hostname
echo -e "127.0.0.1 localhost\n::1 localhost\n$MASTER_IP saltmaster" > /etc/hosts
```

```
useradd -m $SUDOUSER
echo -e "$PASSWORD\n$PASSWORD" | passwd $SUDOUSER

# SSH Settings:
# Disable sftp subsystem
sed -i 's/Subsystem/#Subsystem/' /etc/ssh/sshd_config
#sed -i 's/#PasswordAuthentication yes/PasswordAuthentication no/' /etc/ssh/sshd_config
sed -i '/#PermitRootLogin pro/c\PermitRootLogin no' /etc/ssh/sshd_config
systemctl enable sshd

# GRUB installation
grub-install --target=i386-pc /dev/$HD_DEVICE
sed -i 's/GRUB_TIMEOUT=5/GRUB_TIMEOUT=0/' /etc/default/grub
sed -i '/LINUX_DEF/c\GRUB_CMDLINE_LINUX_DEFAULT="loglevel=3 quiet"' /etc/default/grub
grub-mkconfig -o /boot/grub/grub.cfg

# Install old python-msgpack for Salt-py3, make pacman to ignore it in the future
sed -i 's/#IgnorePkg   =/IgnorePkg   = python-msgpack/' /etc/pacman.conf
wget -P /var/cache/pacman/pkg \
      https://archive.archlinux.org/packages/p/python-msgpack/python-msgpack-0.6.2-3-
x86_64.pkg.tar.xz
pacman -U --noconfirm /var/cache/pacman/pkg/python-msgpack-0.6.2-3-x86_64.pkg.tar.xz

# Update trust anchors:
trust anchor /etc/ssl/private/saltmaster.crt
update-ca-trust

# Download saltmaster's public GPG key to sudousers home directory
mkdir /home/$SUDOUSER/.gnupg
wget -P /home/$SUDOUSER/.gnupg https://saltmaster:$FILESERVER_PORT/saltmaster.gpg
pacman-key --add /home/$SUDOUSER/.gnupg/saltmaster.gpg
pacman-key --lsign-key $GPG_KEYID

# Set up custom repository
echo -e "[saltmaster]\nServer = https://saltmaster:$FILESERVER_PORT\n" >> /etc/pacman.conf
pacman -Sy
pacman -S --noconfirm salt-py3

# Salt configuration
systemctl enable salt-minion
echo -e "master: saltmaster\nmaster_port: $M_PORT\npublish_port: $P_PORT\nid: $HOSTNAME"
> /etc/salt/minion

EOF

# Copy existing DNS settings to new installation
cp /etc/resolv.conf /mnt/etc/resolv.conf
```

Appendix 6. Salt state tree

```
/srv/salt
├── aide
│   └── init.sls
├── aslr
│   └── init.sls
├── bootloader
│   └── init.sls
├── disable_filesystems
│   └── init.sls
├── disable_usb
│   └── init.sls
├── harden_mounts
│   ├── home.sls
│   ├── init.sls
│   ├── shm.sls
│   ├── tmp.sls
│   ├── var_log_audit.sls
│   ├── var_log.sls
│   └── var_tmp.sls
├── managed_files
│   └── etc
│       ├── aide.conf
│       ├── grub.d
│       │   ├── 09_make_OS_entries_unrestricted
│       │   └── 40_custom
│       ├── modprobe.d
│       │   ├── disable_filesystems.conf
│       │   └── disable_usb.conf
│       ├── pacman.conf
│       ├── pacman.d
│       │   └── mirrorlist
│       ├── sysctl.d
│       │   ├── aslr.conf
│       │   └── network.conf
│       └── systemd
│           └── coredump.conf
├── network
│   ├── disable_forwarding.sls
│   ├── disable_icmp_redirects.sls
│   ├── disable_source_routing.sls
│   ├── icmp_ignore.sls
│   ├── init.sls
│   ├── ipv6_router_ads.sls
│   ├── log_martians.sls
│   ├── reverse_path_filtering.sls
│   └── syncookies.sls
├── package_management
│   └── init.sls
├── systemd_services
│   ├── coredump.sls
│   └── init.sls
└── top.sls
```

Appendix 7. Checklist spreadsheet

CIS Benchmark item	Set	Related files	TODO	Notes
1 Initial Setup				
1.1 Filesystem Configuration				
1.1.1 Disable unused filesystems				
1.1.1.1 Ensure mounting of cramfs filesystems is disabled (Scored)	X	salt/disable_filesystems/init.sls salt/managed_files/etc/modprobe.d/disable_filesystems.conf		All except mandatory filesystems disabled
1.1.1.2 Ensure mounting of freevxfs filesystems is disabled (Scored)	X	salt/disable_filesystems/init.sls salt/managed_files/etc/modprobe.d/disable_filesystems.conf		All except mandatory filesystems disabled
1.1.1.3 Ensure mounting of jffs2 filesystems is disabled (Scored)	X	salt/disable_filesystems/init.sls salt/managed_files/etc/modprobe.d/disable_filesystems.conf		All except mandatory filesystems disabled
1.1.1.4 Ensure mounting of hfs filesystems is disabled (Scored)	X	salt/disable_filesystems/init.sls salt/managed_files/etc/modprobe.d/disable_filesystems.conf		All except mandatory filesystems disabled
1.1.1.5 Ensure mounting of hfsplus filesystems is disabled (Scored)	X	salt/disable_filesystems/init.sls salt/managed_files/etc/modprobe.d/disable_filesystems.conf		All except mandatory filesystems disabled
1.1.1.6 Ensure mounting of squashfs filesystems is disabled (Scored)	X	salt/disable_filesystems/init.sls salt/managed_files/etc/modprobe.d/disable_filesystems.conf		All except mandatory filesystems disabled
1.1.1.7 Ensure mounting of udf filesystems is disabled (Scored)	X	salt/disable_filesystems/init.sls salt/managed_files/etc/modprobe.d/disable_filesystems.conf		All except mandatory filesystems disabled
1.1.1.8 Ensure mounting of FAT filesystems is limited (Not Scored)	X	salt/disable_filesystems/init.sls salt/managed_files/etc/modprobe.d/disable_filesystems.conf		All except mandatory filesystems disabled
1.1.2 Ensure /tmp is configured (Scored)	X	salt/harden_mounts/tmp.sls		
1.1.3 Ensure nodev option set on /tmp partition (Scored)	X	salt/harden_mounts/tmp.sls		
1.1.4 Ensure nosuid option set on /tmp partition (Scored)	X	salt/harden_mounts/tmp.sls		
1.1.5 Ensure noexec option set on /tmp partition (Scored)	X	salt/harden_mounts/tmp.sls		
1.1.6 Ensure separate partition exists for /var (Scored)	X	install/partition_map		
1.1.7 Ensure separate partition exists for /var/tmp (Scored)	X	install/partition_map		
1.1.8 Ensure nodev option set on /var/tmp partition (Scored)	X	salt/harden_mounts/var_tmp.sls		
1.1.9 Ensure nosuid option set on /var/tmp partition (Scored)	X	salt/harden_mounts/var_tmp.sls		
1.1.10 Ensure noexec option set on /var/tmp partition (Scored)	X	salt/harden_mounts/var_tmp.sls		
1.1.11 Ensure separate partition exists for /var/log (Scored)	X	install/partition_map		
1.1.12 Ensure separate partition exists for /var/log/audit (Scored)	X	install/partition_map		
1.1.13 Ensure separate partition exists for /home (Scored)	X	install/partition_map		
1.1.14 Ensure nodev option set on /home partition (Scored)	X	salt/harden_mounts/home.sls		
1.1.15 Ensure nodev option set on /dev/shm partition (Scored)	X	salt/harden_mounts/shm.sls		
1.1.16 Ensure nosuid option set on /dev/shm partition (Scored)	X	salt/harden_mounts/shm.sls		
1.1.17 Ensure noexec option set on /dev/shm partition (Scored)	X	salt/harden_mounts/shm.sls		
1.1.18 Ensure nodev option set on removable media partitions (Not Scored)				Untested because of cloud
1.1.19 Ensure nosuid option set on removable media partitions (Not Scored)				Untested because of cloud
1.1.20 Ensure noexec option set on removable media partitions (Not Scored)				Untested because of cloud
1.1.21 Ensure sticky bit is set on all world-writable directories (Scored)	X			AUDIT: No output expected salt '*' cmd.run "df --local -P | awk '{if (NR!=1) print $6}' | xargs -I '{}' find '{}' -xdev -type d \(-perm -0002 -a ! -perm -1000 \) 2>/dev/null"
1.1.22 Disable Automounting (Scored)	X			Not installed (autofs)

1.1.23 Disable USB Storage (Scored)	X	salt/disable_usb/init.sls salt/managed_files/etc/modprobe.d/disable_usb.conf		Runs mkinitcpio -P on changes: reboot needed
1.2 Configure Software Updates				
1.2.1 Ensure package manager repositories are configured (Not Scored)	X	salt/package_management/init.sls salt/managed_files/etc/pacman.conf salt/managed_files/etc/pacman.d/mirrorlist		
1.2.2 Ensure GPG keys are configured (Not Scored)	X	salt/package_management/init.sls		archlinux-keyring package
1.3 Filesystem Integrity Checking				
1.3.1 Ensure AIDE is installed (Scored)	X	salt/aide/init.sls salt/managed_files/etc/aide.conf		AIDE is in AUR, orphaned package
1.3.2 Ensure filesystem integrity is regularly checked (Scored)			configure, run, regular checks, remote checks	AIDE functionality: aidcheck.service configuration?
1.4 Secure Boot Settings				
1.4.1 Ensure permissions on bootloader config are configured (Scored)	X	salt/bootloader/init.sls		
1.4.2 Ensure bootloader password is set (Scored)	X	salt/bootloader/init.sls salt/managed_files/etc/grub.d/09_make_OS_entries_unrestricted salt/managed_files/etc/grub.d/40_custom		Password has to be created with 'grub-mkpasswd-pbkfd2' and set in 40_custom -file before running the state
1.4.3 Ensure authentication required for single user mode (Scored)	X	/etc/shadow	report	No ! signs in root accounts row
1.4.4 Ensure interactive boot is not enabled (Not Scored)	N/A			Interactive boot with sysconfig not present
1.5 Additional Process Hardening				
1.5.1 Ensure core dumps are restricted (Scored)	X	salt/systemd_services/coredump.sls salt/managed_files/etc/systemd/coredump.conf	report	
1.5.2 Ensure XD/NX support is enabled (Scored)	X		report	AUDIT: salt '*' cmd.run "journalctl -b | grep 'protection: active'"
1.5.3 Ensure address space layout randomization (ASLR) is enabled (Scored)	X	salt/aslr/init.sls salt/managed_files/etc/sysctl.d/aslr.conf	report	
1.5.4 Ensure prelink is disabled (Scored)	X			Not installed
1.6 Mandatory Access Control				
1.6.1 Ensure Mandatory Access Control Software is Installed				
1.6.1.1 Ensure SELinux or AppArmor are installed (Scored)			Install & configure	Requirements: install from scratch https://github.com/archlinuxhardened/selinux
1.6.2 Configure SELinux				
1.6.2.1 Ensure SELinux is not disabled in bootloader configuration (Scored)	N/A			Not installed
1.6.2.2 Ensure the SELinux state is enforcing (Scored)	N/A			Not installed
1.6.2.3 Ensure SELinux policy is configured (Scored)	N/A			Not installed
1.6.2.4 Ensure SETroubleshoot is not installed (Scored)	N/A			Not installed
1.6.2.5 Ensure the MCS Translation Service (mcstrans) is not installed (Scored)	N/A			Not installed
1.6.2.6 Ensure no unconfined daemons exist (Scored)	N/A			Not installed
1.6.3 Configure AppArmor				
1.6.3.1 Ensure AppArmor is not disabled inbootloader configuration (Scored)	N/A			Not installed
1.6.3.2 Ensure all AppArmor Profiles are enforcing (Scored)	N/A			Not installed
1.7 Warning Banners				
1.7.1 Command Line Warning Banners				
1.7.1.1 Ensure message of the day is configured properly (Scored)				
1.7.1.2 Ensure local login warning banner				

is configured properly (Scored)			
1.7.1.3 Ensure remote login warning banner is configured properly (Scored)			
1.7.1.4 Ensure permissions on /etc/motd are configured (Scored)			
1.7.1.5 Ensure permissions on /etc/issue are configured (Scored)			
1.7.1.6 Ensure permissions on /etc/issue.net are configured (Scored)			
1.7.2 Ensure GDM login banner is configured (Scored)			
1.8 Ensure updates, patches, and additional security software are installed (Not Scored)	X	Automate	salt '*' cmd.run 'pacman -Suy --noconfirm' salt '*' cmd.run 'reboot'
2 Services			
2.1 inetd Services			
2.1.1 Ensure chargen services are not enabled (Scored)	X		Not installed
2.1.2 Ensure daytime services are not enabled (Scored)	X		Not installed
2.1.3 Ensure discard services are not enabled (Scored)	X		Not installed
2.1.4 Ensure echo services are not enabled (Scored)	X		Not installed
2.1.5 Ensure time services are not enabled (Scored)	X		Not installed
2.1.6 Ensure rsh server is not enabled (Scored)	X		Not installed
2.1.7 Ensure talk server is not enabled (Scored)	X		Not installed
2.1.8 Ensure telnet server is not enabled (Scored)	X		Not installed
2.1.9 Ensure tftp server is not enabled (Scored)	X		Not installed
2.1.10 Ensure xinetd is not enabled (Scored)	X		Not installed
2.2 Special Purpose Services			
2.2.1 Time Synchronization			
2.2.1.1 Ensure time synchronization is in use (Not Scored)			
2.2.1.2 Ensure ntp is configured (Scored)			
2.2.1.3 Ensure chrony is configured (Scored)			
2.2.1.4 Ensure systemd-timesyncd is configured (Scored)			
2.2.2 Ensure X Window System is not installed (Scored)	X		Not installed
2.2.3 Ensure Avahi Server is not enabled (Scored)	X		Not installed
2.2.4 Ensure CUPS is not enabled (Scored)	X		Not installed
2.2.5 Ensure DHCP Server is not enabled (Scored)	X		Not installed
2.2.6 Ensure LDAP server is not enabled (Scored)	X		Not installed
2.2.7 Ensure NFS and RPC are not enabled (Scored)	X		Not installed
2.2.8 Ensure DNS Server is not enabled (Scored)	X		Not installed
2.2.9 Ensure FTP Server is not enabled (Scored)	X		Not installed
2.2.10 Ensure HTTP server is not enabled (Scored)	X		Not installed
2.2.11 Ensure IMAP and POP3 server is not enabled (Scored)	X		Not installed
2.2.12 Ensure Samba is not enabled (Scored)	X		Not installed
2.2.13 Ensure HTTP Proxy Server is not enabled (Scored)	X		Not installed

2.2.14 Ensure SNMP Server is not enabled (Scored)	X		Not installed
2.2.15 Ensure mail transfer agent is configured for local-only mode (Scored)	X		Not installed
2.2.16 Ensure rsync service is not enabled (Scored)	X		Not installed
2.2.17 Ensure NIS Server is not enabled (Scored)	X		Not installed
2.3 Service Clients			
2.3.1 Ensure NIS Client is not installed (Scored)	X		Not installed
2.3.2 Ensure rsh client is not installed (Scored)	X		Not installed
2.3.3 Ensure talk client is not installed (Scored)	X		Not installed
2.3.4 Ensure telnet client is not installed (Scored)	X		Not installed
2.3.5 Ensure LDAP client is not installed (Scored)	X		Not installed
3 Network Configuration			
3.1 Network Parameters (Host Only)			
3.1.1 Ensure IP forwarding is disabled (Scored)	X	salt/network/disable_forwarding.sls salt/managed_files/etc/sysctl.d/ipforwarding.conf	
3.1.2 Ensure packet redirect sending is disabled (Scored)	X	salt/network/disable_icmp_redirects.sls salt/managed_files/etc/sysctl.d/ICMPredirects.conf	
3.2 Network Parameters (Host and Router)			
3.2.1 Ensure source routed packets are not accepted (Scored)	X	salt/network/disable_source_routing.sls salt/managed_files/etc/sysctl.d/network.conf	
3.2.2 Ensure ICMP redirects are not accepted (Scored)	X	salt/network/disable_icmp_redirects.sls salt/managed_files/etc/sysctl.d/network.conf	
3.2.3 Ensure secure ICMP redirects are not accepted (Scored)	X	salt/network/disable_icmp_redirects.sls salt/managed_files/etc/sysctl.d/network.conf	
3.2.4 Ensure suspicious packets are logged (Scored)	X	salt/network/log_martians.sls salt/managed_files/etc/sysctl.d/network.conf	
3.2.5 Ensure broadcast ICMP requests are ignored (Scored)	X	salt/network/ignore_icmp_requests.sls salt/managed_files/etc/sysctl.d/network.conf	
3.2.6 Ensure bogus ICMP responses are ignored (Scored)	X	salt/network/ignore_icmp_requests.sls salt/managed_files/etc/sysctl.d/network.conf	
3.2.7 Ensure Reverse Path Filtering is enabled (Scored)	X	salt/network/reverse_path_filtering.sls salt/managed_files/etc/sysctl.d/network.conf	This configuration breaks if dynamic routing protocols are used (BGP, OSPF)
3.2.8 Ensure TCP SYN Cookies is enabled (Scored)	X	salt/network/syncookies.sls salt/managed_files/etc/sysctl.d/network.conf	Allows system to accept valid connection even if under SYN flood attack
3.2.9 Ensure IPv6 router advertisements are not accepted (Scored)	X	salt/network/ipv6_router_ads.sls salt/managed_files/etc/sysctl.d/network.conf	
3.3 TCP Wrappers			
3.3.1 Ensure TCP Wrappers is installed (Not Scored)			
3.3.2 Ensure /etc/hosts.allow is configured (Not Scored)			
3.3.3 Ensure /etc/hosts.deny is configured (Not Scored)			
3.3.4 Ensure permissions on/etc/hosts.allow are configured (Scored)			
3.3.5 Ensure permissions on /etc/hosts.deny are configured (Scored)			
3.4 Uncommon Network Protocols			
3.4.1 Ensure DCCP is disabled (Scored)			
3.4.2 Ensure SCTP is disabled (Scored)			
3.4.3 Ensure RDSis disabled (Scored)			
3.4.4 Ensure TIPC is disabled (Scored)			
3.5 Firewall Configuration			
3.5.1 Configure IPv6 ip6tables			

3.5.1.1 Ensure IPv6 default deny firewall policy (Scored)
3.5.1.2 Ensure IPv6 loopback traffic is configured (Scored)
3.5.1.3 Ensure IPv6 outbound and established connections are configured (Not Scored)
3.5.1.4 Ensure IPv6 firewall rules exist for all open ports (Not Scored)

3.5.2 Configure IPv4 iptables

3.5.2.1 Ensure default deny firewall policy (Scored)
3.5.2.2 Ensure loopback traffic is configured (Scored)
3.5.2.3 Ensure outbound and established connections are configured (Not Scored)

3.5.2.4 Ensure firewall rules exist for all open ports (Scored)
3.5.3 Ensure iptables is installed (Scored)

3.6 Ensure wireless interfaces are disabled (Not Scored)
3.7 Disable IPv6 (Not Scored)

4 Logging and Auditing

4.1 Configure System Accounting (auditd)

4.1.1 Configure Data Retention

4.1.1.1 Ensure audit log storage size is configured (Scored)
4.1.1.2 Ensure system is disabled when audit logs are full (Scored)
4.1.1.3 Ensure audit logs are not automatically deleted (Scored)
4.1.2 Ensure auditd is installed (Scored)

4.1.3 Ensure auditd service is enabled (Scored)
4.1.4 Ensure auditing for processes that start prior to auditd is enabled (Scored)
4.1.5 Ensure events that modify date and time information are collected (Scored)

4.1.6 Ensure events that modify user/group information are collected (Scored)
4.1.7 Ensure events that modify the system's network environment are collected (Scored)
4.1.8 Ensure events that modify the system's Mandatory Access Controls are collected (Scored)
4.1.9 Ensure login and logout events are collected (Scored)
4.1.10 Ensure session initiation information is collected (Scored)
4.1.11 Ensure discretionary access control permission modification events are collected (Scored)
4.1.12 Ensure unsuccessful unauthorized file access attempts are collected (Scored)
4.1.13 Ensure use of privileged commands is collected (Scored)
4.1.14 Ensure successful file system mounts are collected (Scored)
4.1.15 Ensure file deletion events by users are collected (Scored)
4.1.16 Ensure changes to systemadministration scope (sudoers) is collected (Scored)
4.1.17 Ensure system administrator actions (sudolog) are collected (Scored)

4.1.18 Ensure kernel module loading and unloading is collected (Scored)
4.1.19 Ensure the audit configuration is immutable (Scored)
4.2 Configure Logging

4.2.1 Configure rsyslog

4.2.1.1 Ensure rsyslog is installed (Scored)
4.2.1.2 Ensure rsyslog Service is enabled (Scored)
4.2.1.3 Ensure logging is configured (Not Scored)
4.2.1.4 Ensure rsyslog default file permissions configured (Scored)
4.2.1.5 Ensure rsyslog is configured to send logs to a remote log host (Scored)
4.2.1.6 Ensure remote rsyslog messages are only accepted on designated log hosts. (Not Scored)
4.2.2 Configure journald

4.2.2.1 Ensure journald is configured to send logs to rsyslog (Scored)
4.2.2.2 Ensure journald is configured to compress large log files (Scored)
4.2.2.3 Ensure journald is configured to write logfiles to persistent disk (Scored)
4.2.3 Ensure permissions on all logfiles are configured (Scored)
4.3 Ensure logrotate is configured (Not Scored)
5 Access, Authentication and Authorization
5.1 Configure cron

5.1.1 Ensure cron daemon is enabled (Scored)
5.1.2 Ensure permissions on /etc/crontab are configured (Scored)
5.1.3 Ensure permissions on /etc/cron.hourly are configured (Scored)
5.1.4 Ensure permissions on /etc/cron.daily are configured (Scored)
5.1.5 Ensure permissions on /etc/cron.weekly are configured (Scored)

5.1.6 Ensure permissions on /etc/cron.monthly are configured (Scored)

5.1.7 Ensure permissions on /etc/cron.d are configured (Scored)
5.1.8 Ensure at/cron is restricted to authorized users (Scored)
5.2 SSH Server Configuration

5.2.1 Ensure permissions on /etc/ssh/sshd_config are configured (Scored)
5.2.2 Ensure permissions on SSH private host key files are configured (Scored)

5.2.3 Ensure permissions on SSH public host key files are configured (Scored)

5.2.4 EnsureSSH Protocol is set to 2 (Scored)
5.2.5 Ensure SSH LogLevel is appropriate (Scored)
5.2.6 Ensure SSH X11 forwarding is disabled (Scored)
5.2.7 Ensure SSH MaxAuthTries is set to 4 or less (Scored)
5.2.8 Ensure SSH IgnoreRhosts is enabled (Scored)
5.2.9 Ensure SSH HostbasedAuthentication is disabled

(Scored)
5.2.10 Ensure SSH root login is disabled (Scored)
5.2.11 Ensure SSH PermitEmptyPasswords is disabled (Scored)
5.2.12 Ensure SSH PermitUserEnvironment is disabled (Scored)
5.2.13 Ensure only strong Ciphers are used (Scored)
5.2.14 Ensure only strong MAC algorithms are used (Scored)
5.2.15 Ensure only strong Key Exchange algorithms are used (Scored)
5.2.16 Ensure SSH Idle Timeout Interval is configured (Scored)
5.2.17 Ensure SSH LoginGraceTime is set to one minute or less (Scored)
5.2.18 Ensure SSH access is limited (Scored)
5.2.19 Ensure SSH warning banner is configured (Scored)
5.2.20 Ensure SSH PAM is enabled (Scored)
5.2.21 Ensure SSH AllowTcpForwarding is disabled (Scored)

5.2.22 Ensure SSH MaxStartups is configured (Scored)
5.2.23 Ensure SSH MaxSessions is set to 4 or less (Scored)

5.3 Configure PAM

5.3.1 Ensure password creation requirements are configured (Scored)
5.3.2 Ensure lockout for failed password attempts is configured (Not Scored)
5.3.3 Ensure password reuse is limited (Not Scored)
5.3.4 Ensure password hashing algorithm is SHA-512 (Not Scored)

5.4 User Accounts and Environment

5.4.1 Set Shadow Password Suite Parameters

5.4.1.1 Ensure password expiration is 365 days or less (Scored)
5.4.1.2 Ensure minimum days between password changes is 7 or more (Scored)

5.4.1.3 Ensure password expiration warning days is 7 or more (Scored)
5.4.1.4 Ensure inactive password lock is 30 days or less (Scored)
5.4.1.5 Ensure all users last password change date is in the past (Scored)
5.4.2 Ensure system accounts are secured (Scored)
5.4.3 Ensure default group for the root account is GID 0 (Scored)
5.4.4 Ensure default user umask is 027 or more restrictive (Scored)
5.4.5 Ensure default user shell timeout is 900 seconds or less (Scored)
5.5 Ensure root login is restricted to system console (Not Scored)
5.6 Ensure access to the su command is restricted (Scored)

6 System Maintenance

6.1 System File Permissions

6.1.1 Audit system file permissions (Not Scored)