jamk

Hardening- Lab 4

Group 4

Nguyen Ngo - AE8880
Dung Doan - AA7785
Syed Fawaz - AD9946
Jasper (Franciscus) van de Klundert - AG9056
Sanka De Silva - AC4892

Exercise Lab 4

Hardening

7/4/2025

Bachelor’s Program of Information and Communication Technology

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

Contents

Figures

1 Introduction

This exercise aims to explore the hardening of Linux, Docker, and WordPress to enhance their security and protection. Specifically, the focus will be on hardening Linux using the Lynis software, a tool that assesses system security and offers recommendations for fixing vulnerabilities. The exercise will be carried out in a Virtual Learning Environment (VLE), specifically on a WWW server (Figure 1).

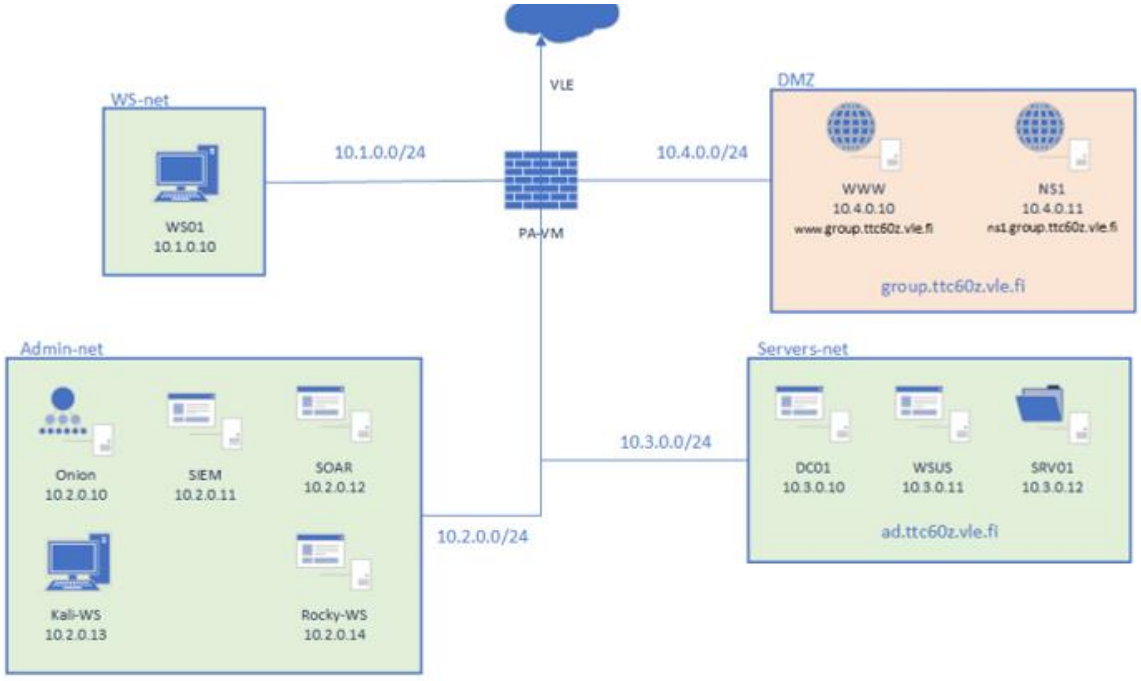


Figure 2 VLE Environment

During the exercise, we will create individual user accounts for team members and ensure that SSH connections are secure. Additionally, we will disable SSH connections for the root user to prevent the use of the administrator account for direct logins, which facilitates easier management and auditing tracking. This process will enhance the security of the WWW server in use, enable an effective monitoring practice, and prepare the team for implementing and maintaining security improvements.

2 Theory of the Lab

Securing web services, such as WordPress sites, is a critical part of hardening and reducing the attack surface. These sites often serve as the initial target of interest for attackers. Without the necessary security measures, web services can be quite vulnerable, allowing easy penetration deeper into a company's network and data systems. (Mallory, 2020)

2.1 Hardening

Hardening refers to the process of enhancing the security of a system or application by reducing its vulnerability surface. Hardening can involve several measures (Schrader, 2023), such as:

- Removing unnecessary services from application servers
- Removing sample applications
- Changing the default ports of services
- Removing example user accounts
- Changing default passwords
- Modifying weak default settings
- Disabling unnecessary protocols

2.2 WordPress

WordPress is a popular content management system, which makes it a prime target for attackers because the more popular the platform, the more attack methods are developed against it. Common vulnerabilities of WordPress sites include:

- Outdated software versions and plugins
- Weak passwords and usernames
- Database exposure
- XSS (Cross-Site Scripting) vulnerabilities

(Juviler, 2024)

2.3 Lynis

Lynis is an extensive open-source security auditing tool for UNIX-based systems, including Linux, macOS, and BSD. Its main goal is to assess system security and provide recommendations for hardening the system. Lynis performs a thorough security scan directly on the system, checking general system information, identifying vulnerable software packages, and detecting potential configuration issues. (Zorz, 2024)

2.4 SELinux

SELinux (Security-Enhanced Linux) provides a security framework designed for Linux systems. It enhances the management of system resources using Mandatory Access Control (MAC), which enables setting permissions for applications and processes based on their security context—this determines if access is allowed or denied. In the context of Docker containers, SELinux plays a critical role by blocking access to important files on the host machine, which could otherwise be accessed if SELinux were not active.(Zivanov, 2023)

2.5 Rootless mode

When Docker containers are operated under the root user, they can access critical files like the shadow file, which holds usernames and passwords, creating a security vulnerability. Therefore, we configured Docker to run in rootless mode, using a user with limited permissions. This setup ensures that even if an attacker compromises the container, they lack the necessary privileges to affect the underlying Linux system, minimizing the potential attack surface.(Jack, 2024)

2.6 WordPress plugins

WordPress plugins or extensions carry potential security risks, particularly if they accumulate in large numbers and become outdated. Even if the rest of the WordPress site is regularly updated, attackers might exploit these neglected plugins. The additional code they introduce can expand the site's vulnerability to attacks, making forgotten plugins a significant threat to security.(Editorial Staff, 2024)

3 Lab Work Progress

Before implementing the hardening measures, we enabled SSH connections from the WS01 terminal to the WWW server and created individual accounts for each team member. After that, we disabled root user logins. We also established a security rule in the firewall that allows traffic from the WS network to the DMZ. (Figure 2)

Figure 3 Security Rule

On the WWW server, before creating the SSH connection, it was necessary to edit the /etc/ssh/sshd_config file to include "PasswordAuthentication yes" to enable SSH connections. This setting had already been configured in our previous lab exercises.

Now, we were able to establish an SSH connection to the WWW server from the WS01 device using the PuTTY software. (Figure 3).

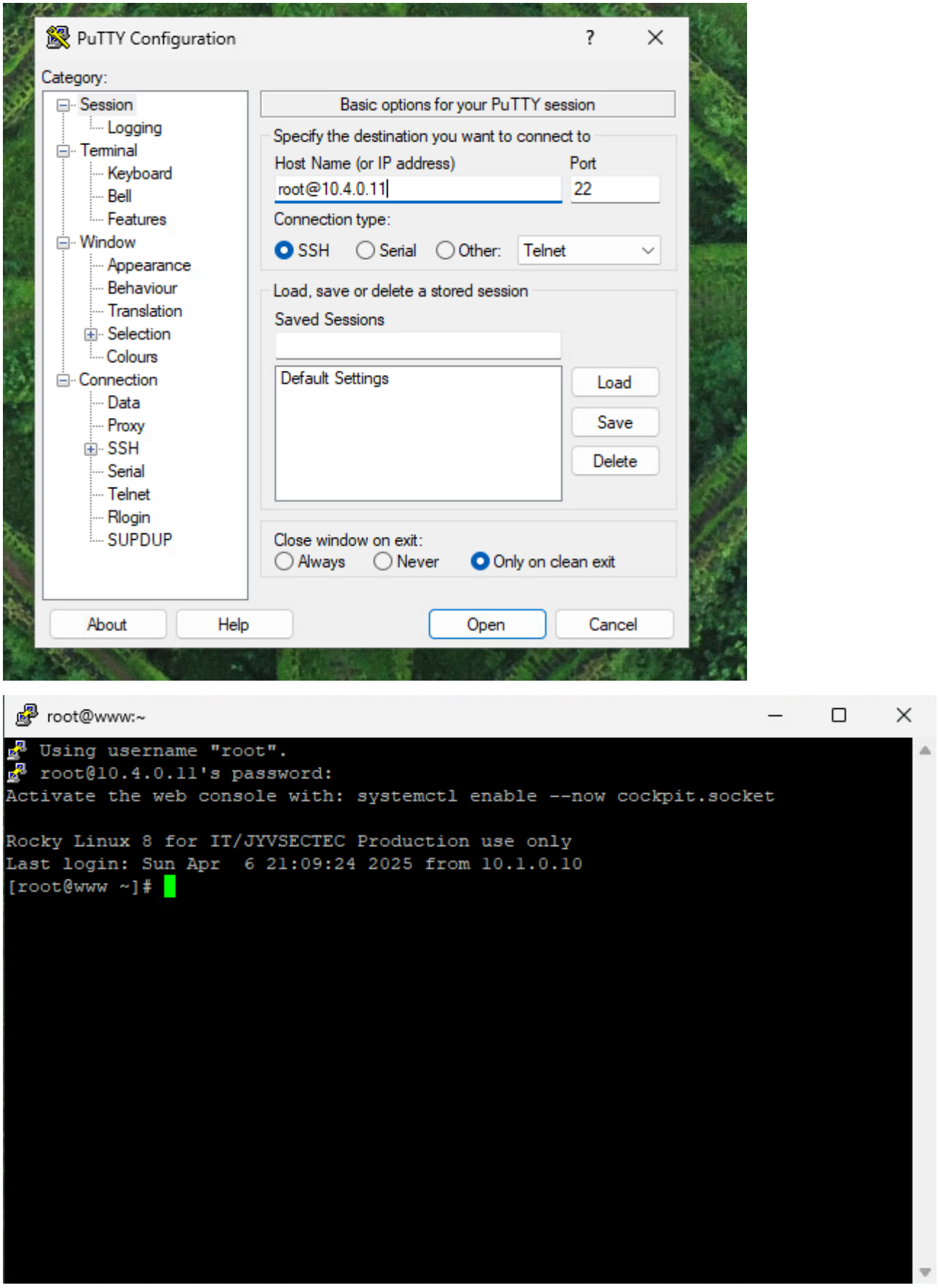


Figure 4 SSH Connection via Putty

We created individual user accounts for each team member and initially set everyone's password to Root66 (which everyone will change themselves). (Figure 4).

```
Last login: Tue Apr  1 22:12:16 2025
[root@www ~]# useradd Nguyen
[root@www ~]# useradd Dung
[root@www ~]# useradd Jasper
[root@www ~]# useradd Fawaz
[root@www ~]# useradd Sand
[root@www ~]# echo "Nguyen:Root66" | chpasswd
[root@www ~]# echo "Dung:Root66" | chpasswd
[root@www ~]# echo "Dunwg:Root66" | chpasswd
chpasswd: line 1: user 'Dunwg' does not exist
chpasswd: error detected, changes ignored
[root@www ~]# echo "Jasper:Root66" | chpasswd
[root@www ~]# echo "Fawaz:Root66" | chpasswd
[root@www ~]# echo "Sand:Root66" | chpasswd
```

Figure 5 User Create

Next, we granted sudo privileges to all users. Sometimes, instead of the sudo group, the group might be called wheel. This can be checked by examining the /etc/sudoers file using the visudo command. As shown in Figure 5, the wheel group is used in this setup.(Figure 5)

```
## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL

## Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL

## Allows members of the users group to mount and unmount the
## cdrom as root
# %users  ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom

## Allows members of the users group to shutdown this system
# %users  localhost=/sbin/shutdown -h now

## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
#includedir /etc/sudoers.d
```

Figure 6 Wheel group

We added a wheel group for our created users. (Figure 6).

```
[root@www ~]# usermod -aG wheel Nguyen
[root@www ~]# usermod -aG wheel Dung
[root@www ~]# usermod -aG wheel Jasper
[root@www ~]# usermod -aG wheel Fawaz
[root@www ~]# usermod -aG wheel Sand
```

Figure 7 Adding a wheel user group

Next, we disabled login for the root user both locally and via SSH, as this poses a security risk. If an attacker manages to log in as the root user on the server, they could cause more damage than a user without such extensive permissions.

The user "root" is also widely known, making it a common target for hackers who might try to log in using this username and guess passwords. Bots may also scan SSH ports and use "root" as the username in their attempts to guess passwords into the system. Using the sudo command can enhance security, and having system administrators use separate usernames facilitates auditing if discrepancies occur. It limits damages to a single user account. (Verhage, 2024.)

We then opened the /etc/ssh/sshd_config file with nano and changed the setting for PermitRootLogin to "no". (Figure 7)

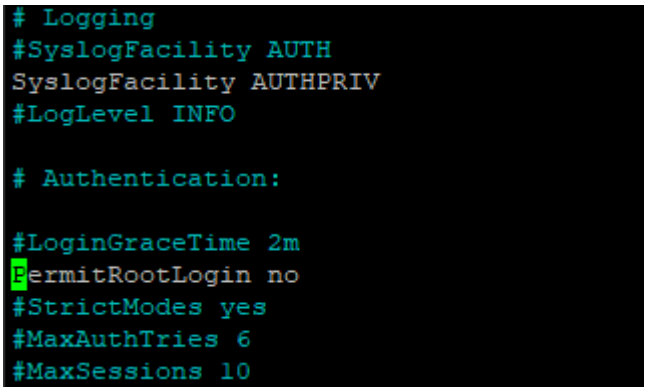


Figure 8 Disabled Root Login

Then , we disabled local root login by editing the /etc/passwd file and changing the shell location for the root user to /sbin/nologin. (Figure 8). By modifying the /etc/passwd file to change the shell for the root user to /sbin/nologin, we effectively disabled local root login. This ensures that it is not possible to log in as root directly from the console, further enhancing the security of our system. (Figure 8).

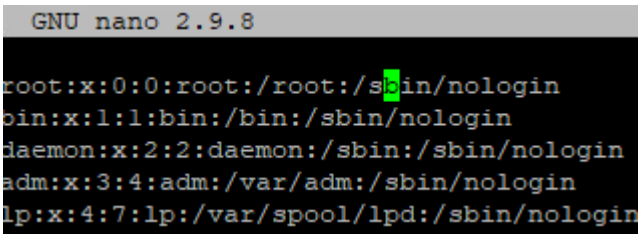


Figure 9 Disabling local root logins.

We quickly realized that some hardening tasks require the use of the root user, so we temporarily reinstated it to perform these specific actions.

3.1 Linux hardening

The hardening of Linux was conducted using the Lynis software. This tool helped assess and improve the system's security by providing specific recommendations based on the current configuration and installed packages.

We then installed Git on the WWW server using the command sudo dnf install git. (Figure 9)

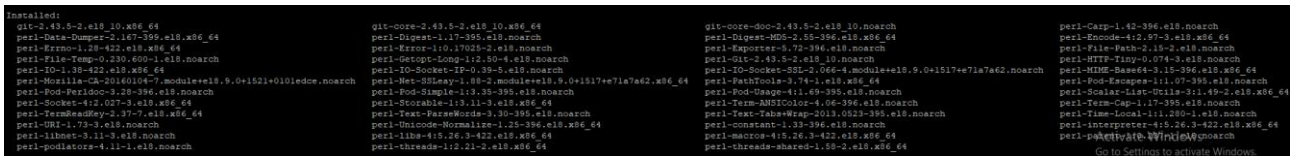


Figure 10 Git installation

Next, we downloaded Lynis from GitHub using the command shown in Figure 10.

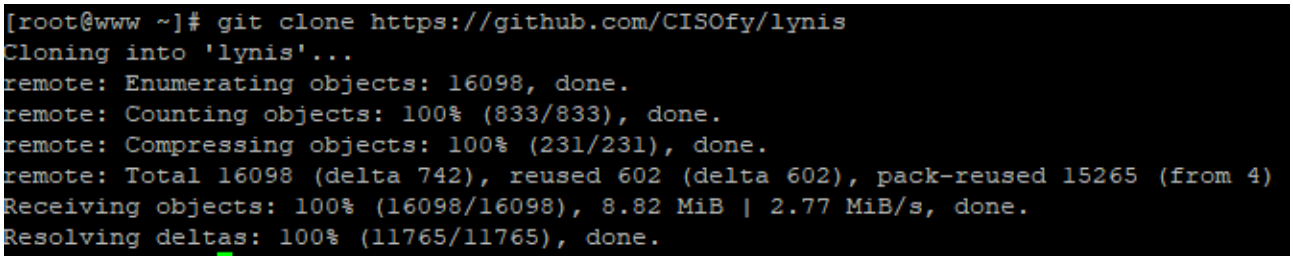


Figure 11 Cloning Lynis

We ran the command ./lynis audit system to test the hardening of the environment, specifically the WWW server. Lynis provided us with 39 suggestions on how to further harden the server. We selected a few of these hardening measures to implement.

First, we implemented hardening measures according to Lynis' AUTH-9230 test. The specific suggestions for hardening are detailed in Figure 11.

The AUTH-9230 test relates to the number of rounds for the hashing algorithm used in system passwords. It identifies if the minimum number of rounds specified for the hashing algorithm is

too low. The settings SHA_CRYPT_MIN_ROUNDS and SHA_CRYPT_MAX_ROUNDS indicate the minimum and maximum number of rounds that the encryption algorithm should execute on a password, respectively. A password is better protected when it is iterated through more rounds, though this also requires more processing power. The password is iterated using this algorithm as many times as necessary to better guard against brute-force attacks.(forest, 2019)

```
* Configure password hashing rounds in /etc/login.defs [AUTH-9230]
  - Related resources
    * Article: Linux password security: hashing rounds: https://linux-audit.com/authentication/configure-the-minimum-password-length-on-linux-systems/
    * Website: https://cisofy.com/lynis/controls/AUTH-9230/
```

```
[root@www lynis]# sudo ./lynis show details AUTH-9230
2025-04-06 22:43:09 Performing test ID AUTH-9230 (Check password hashing rounds)
2025-04-06 22:43:09 Test: Checking SHA_CRYPT_{MIN,MAX}_ROUNDS option in /etc/login.defs
2025-04-06 22:43:09 Result: number of password hashing rounds is not configured
2025-04-06 22:43:09 Suggestion: Configure password hashing rounds in /etc/login.defs [test:AUTH-9230] [details:-] [solution:-]
2025-04-06 22:43:09 Hardening: assigned partial number of hardening points (0 of 2). Currently having 22 points (out of 30)
2025-04-06 22:43:09 ====
```

Figure 12 AUTH-9230 SHA-rounds

We added lines to the /etc/login.defs file, as shown in Figure 12, at the end of the document. This change sets the minimum and maximum rounds for the hashing algorithm, enhancing the security of password encryption on the system.

```
# Use SHA512 to encrypt password.
ENCRYPT_METHOD SHA512
SHA_CRYPT_MIN_ROUNDS 5000
SHA_CRYPT_MAX_ROUNDS 5000
```

Figure 13 SHA512 Encryption Rounds

After comitting the changes, we reran the test to check if the security issue was resolved. According to Figure 13, the settings have been successfully applied and are now in effect.

```
[root@www lynis]# ./lynis show details AUTH-9230
2025-04-06 23:07:37 Performing test ID AUTH-9230 (Check password hashing rounds)
2025-04-06 23:07:37 Test: Checking SHA_CRYPT_{MIN,MAX}_ROUNDS option in /etc/login.defs
2025-04-06 23:07:37 Result: number of password hashing rounds is 5000
2025-04-06 23:07:37 Hardening: assigned maximum number of hardening points for this item (2). Currently having 24 points (out of 30)
2025-04-06 23:07:37 ====
```

Figure 14 AUTH-9230 after hardening

Lynis suggested implementing a minimum and maximum age for passwords under test identifier AUTH-9286. Password policies are configured in the /etc/login.defs file. We set the minimum age at 7 days and the maximum at 30 days, ensuring that passwords must be changed at regular intervals to enhance security.

Implementing minimum and maximum age requirements for password aging enhances the security of Linux systems by ensuring that old, potentially compromised passwords do not give hackers

the opportunity to log into the system. If the system relies solely on password authentication, it's crucial that these passwords are difficult to crack. By requiring users to change their passwords frequently, the risks associated with password leaks or brute-force attacks are significantly mitigated. This policy helps maintain a higher level of security by continuously refreshing the credentials necessary for system access.

```
# Password aging controls:
#
#       PASS_MAX_DAYS   Maximum number of days a password may be used.
#       PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#       PASS_MIN_LEN    Minimum acceptable password length.
#       PASS_WARN_AGE   Number of days warning given before a password expires.
#
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   7
PASS_MIN_LEN    5
PASS_WARN_AGE   7
```

Figure 15 Password aging controls

After implementing the new settings, the password policies were removed from the recommendation list, indicating that the hardening measures were successfully applied. This confirms that the system's security has been enhanced as per the guidelines provided by Lynis.(Figure 15)

```
[root@www lynis]# ./lynis show details AUTH-9286
2025-04-06 23:07:39 Performing test ID AUTH-9286 (Checking user password aging)
2025-04-06 23:07:39 Test: Checking PASS_MIN_DAYS option in /etc/login.defs
2025-04-06 23:07:39 Result: password needs to be at least 7 days old
2025-04-06 23:07:39 Hardening: assigned maximum number of hardening points for this item (3). Currently having 30 points (out of 36)
2025-04-06 23:07:39 Test: Checking PASS_MAX_DAYS option in /etc/login.defs
2025-04-06 23:07:39 Result: password aging limits are not configured
2025-04-06 23:07:39 Suggestion: Configure maximum password age in /etc/login.defs [test:AUTH-9286] [details:-] [solution:-]
2025-04-06 23:07:39 Hardening: assigned partial number of hardening points (0 of 1). Currently having 30 points (out of 37)
2025-04-06 23:07:39 ====
```

Figure 16 Hardened AUTH-9286

One of the hardening suggestions (USB-1000) was that USB storage was enabled and should be disabled. We added the line blacklist usb-storage to the file /etc/modprobe.d/usb-storage.conf, which prevents the use of USB storage devices. After rerunning the test, Lynis confirmed that this action was effective. This measure enhances security by reducing the risk of unauthorized data transfer and malware introduction via USB devices.(Figure 16)

Disabling USB storage means that USB devices cannot be mounted on the system. Essentially, the system will not recognize these devices at all but will block them when an attempt is made to connect them. This precaution can prevent potential security threats, such as the execution of malicious scripts on a Linux system using physical devices. It's a critical security measure for protecting

against unauthorized data access and malware infections that might be introduced via USB ports.

```
[root@www lynis]# ./lynis show details USB-1000
2025-04-06 23:15:21 Performing test ID USB-1000 (Check if USB storage is disabled)
2025-04-06 23:15:21 Test: Checking USB storage driver in directory /etc/modprobe.d and configuration file /etc/modprobe.conf
2025-04-06 23:15:21 Result: found usb-storage driver in disabled state (blacklisted)
2025-04-06 23:15:21 Result: usb-storage driver is disabled
2025-04-06 23:15:21 Hardening: assigned maximum number of hardening points for this item (3). Currently having 124 points (out of 160)
2025-04-06 23:15:21 ====
```

Figure 17 Block access to USB storage

Lynis also suggested that the /etc/hosts file should include the address of our own website, so we added it there under the recommendation NAME-4404. This helps in directing the hostname to the correct IP address internally without relying on external DNS, which can enhance performance and security by reducing reliance on external network infrastructure for DNS resolution.(Figure 17)

```
  GNU nano 2.9.8

127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.4.0.11   www.group4.ttc60z.vle.fi
```

Figure 18 contents of the etc/hosts file

Using Lynis, it could find the useful information for hosts file. (Figure 18)

```
[root@www lynis]# ./lynis show details NAME-4404
2025-04-06 23:21:11 Performing test ID NAME-4404 (Check /etc/hosts contains an entry for this server name)
2025-04-06 23:21:11 Test: Check /etc/hosts contains an entry for this server name
2025-04-06 23:21:11 Result: Found entry for www in /etc/hosts
2025-04-06 23:21:11 ====
```

Figure 19 NAME-4404

Lynis also recommended installing Wazuh(HRDN-7230), a security monitoring tool. However, since this installation is planned for Lab Exercise 6 on Security Controls, we decided to leave it un-addressed at this stage. This approach allows us to focus on current tasks and systematically address additional security measures in subsequent lab activities.(Figure 19)

```
* Harden the system by installing at least one malware scanner, to perform periodic file system scans [HRDN-7230]
  - Solution : Install a tool like rkhunter, chkrootkit, OSSEC, Wazuh
  - Related resources
    * Article: Antivirus for Linux: is it really needed?: https://linux-audit.com/malware/antivirus-for-linux-really-needed/
    * Article: Monitoring Linux Systems for Rootkits: https://linux-audit.com/monitoring-linux-systems-for-rootkits/
    * Website: https://cisofy.com/lynis/controls/HRDN-7230/
```

Figure 20 HRDN-7230

3.2 SSH Hardening

Next, we hardened SSH a bit more. SSH is critical to secure because it provides remote server access, which can lead to significant security breaches.

Our new security protocol mandates that SSH login requires both a password and an SSH key. By requiring both for authentication, we ensure higher security against unauthorized access. Even with a weak password, access is blocked without the corresponding private key. Similarly, if an attacker has the key, they still need the password to gain entry.

Each user must have their own key, meaning a hacker must obtain the specific key, username, and password associated with that user. The key generation process is outlined in figure 20.

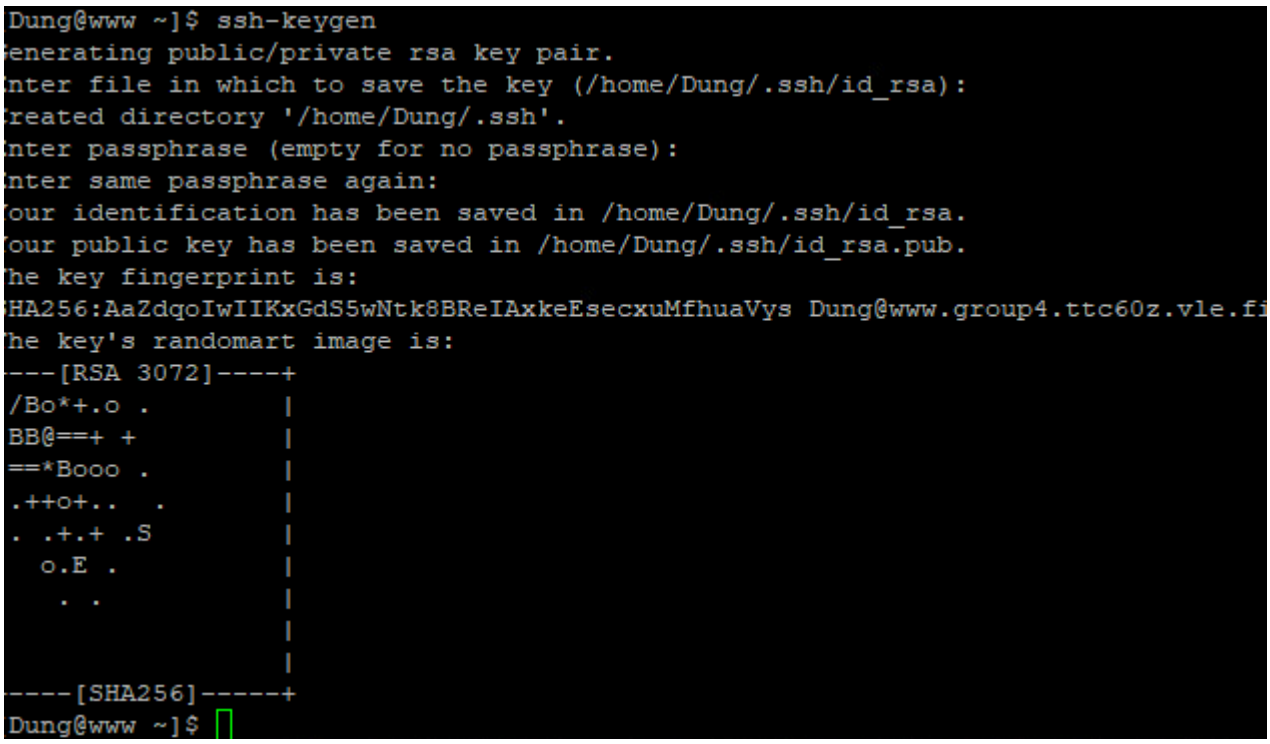


Figure 21 SSH Key Generation

The generated SSH keys can be found in the .ssh directory.(Figure 21)

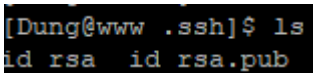


Figure 22 .ssh files

We placed the public key file, named id_rsa.pub, into the authorized keys directory. This directory is used to verify whether the key exists. If the key is present, then access to the system is allowed.(Figure 22)

```
[Dung@www .ssh]$ cp id_rsa.pub authorized_keys
[Dung@www .ssh]$ ls
authorized_keys  id_rsa  id_rsa.pub
```

Figure 23 Copying the .Id_rsa.pub file.

We transferred the SSH key, specifically the id_rsa file, to a text file on workstation WS01 and launched the PuttyGen tool to enable the key for SSH authentication.(Figure 23)

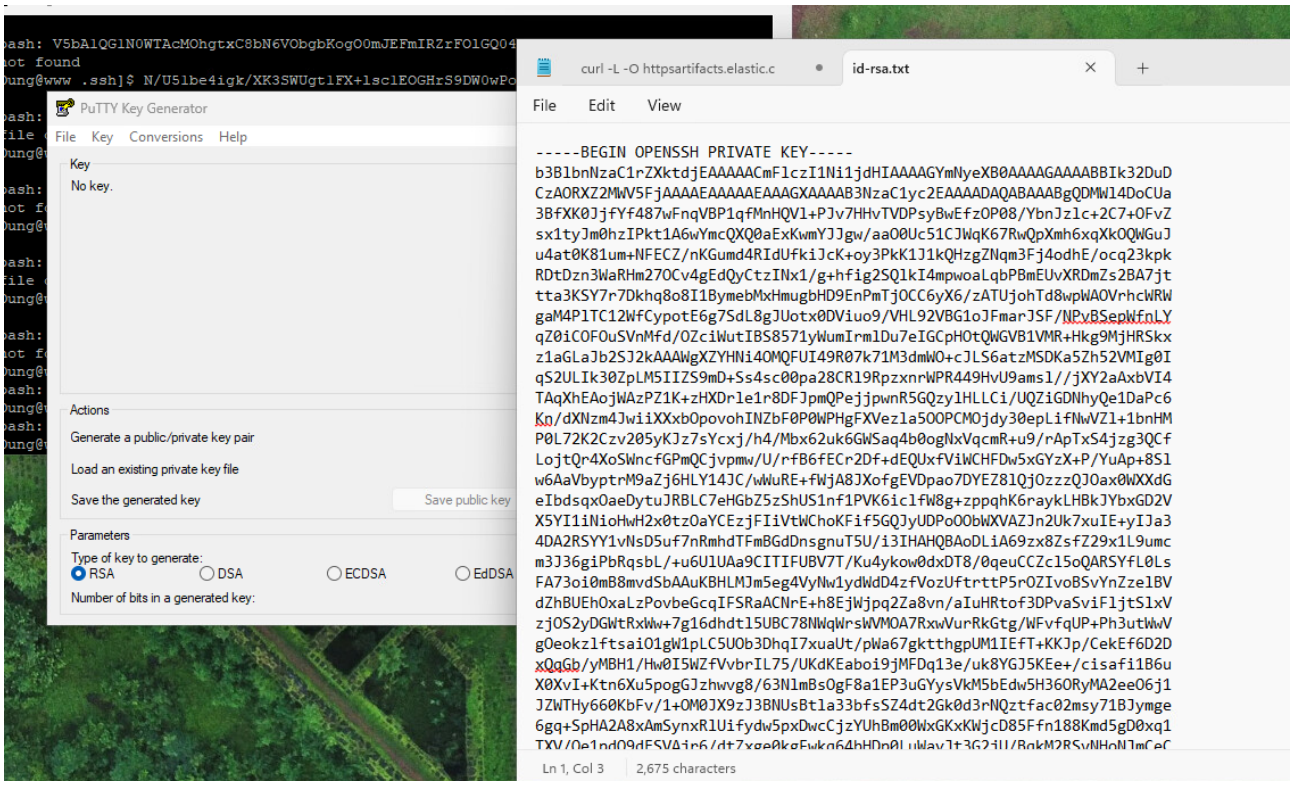


Figure 24 PuTTY Key Generator

Puttygen recognized the SSH key from the text file.(Figure 24)

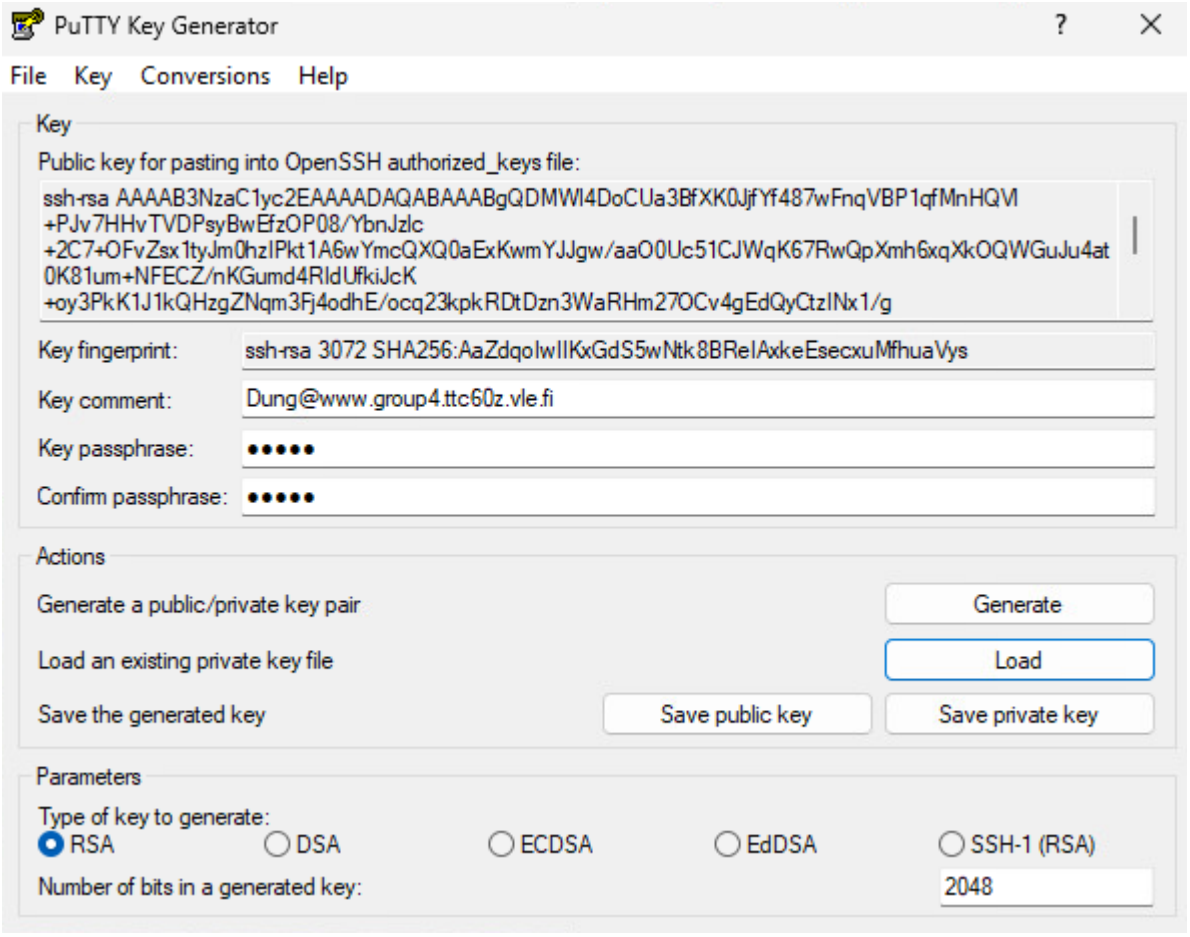


Figure 25 Putty Key Detected

We changed the login settings to require a key by modifying the /etc/ssh/sshd_config file to set the PubkeyAuthentication value to yes.(Figure 25)

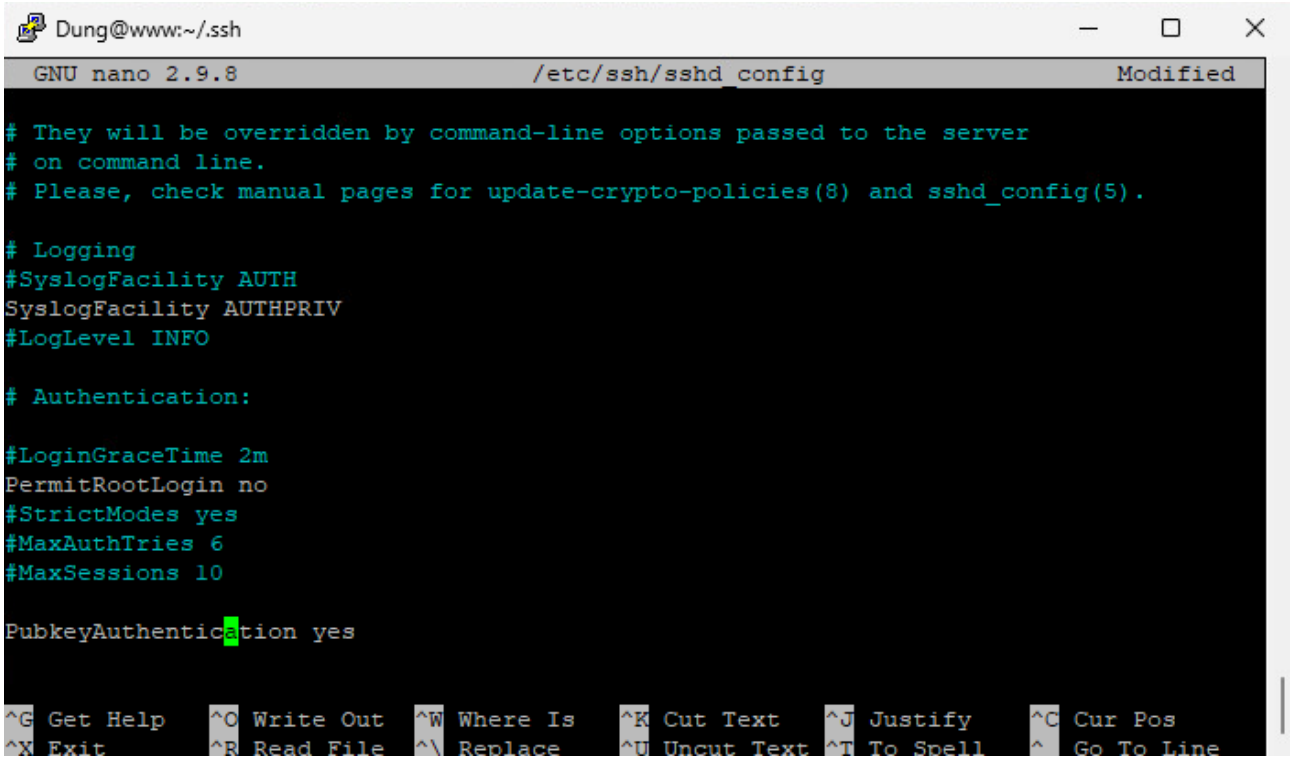


Figure 26 Modification of sshd_config

We configured Putty to require a private key for authentication by adding it under the auth section. During login with Putty, SSH prompts for the passphrase associated with the SSH key.(Figure

26)

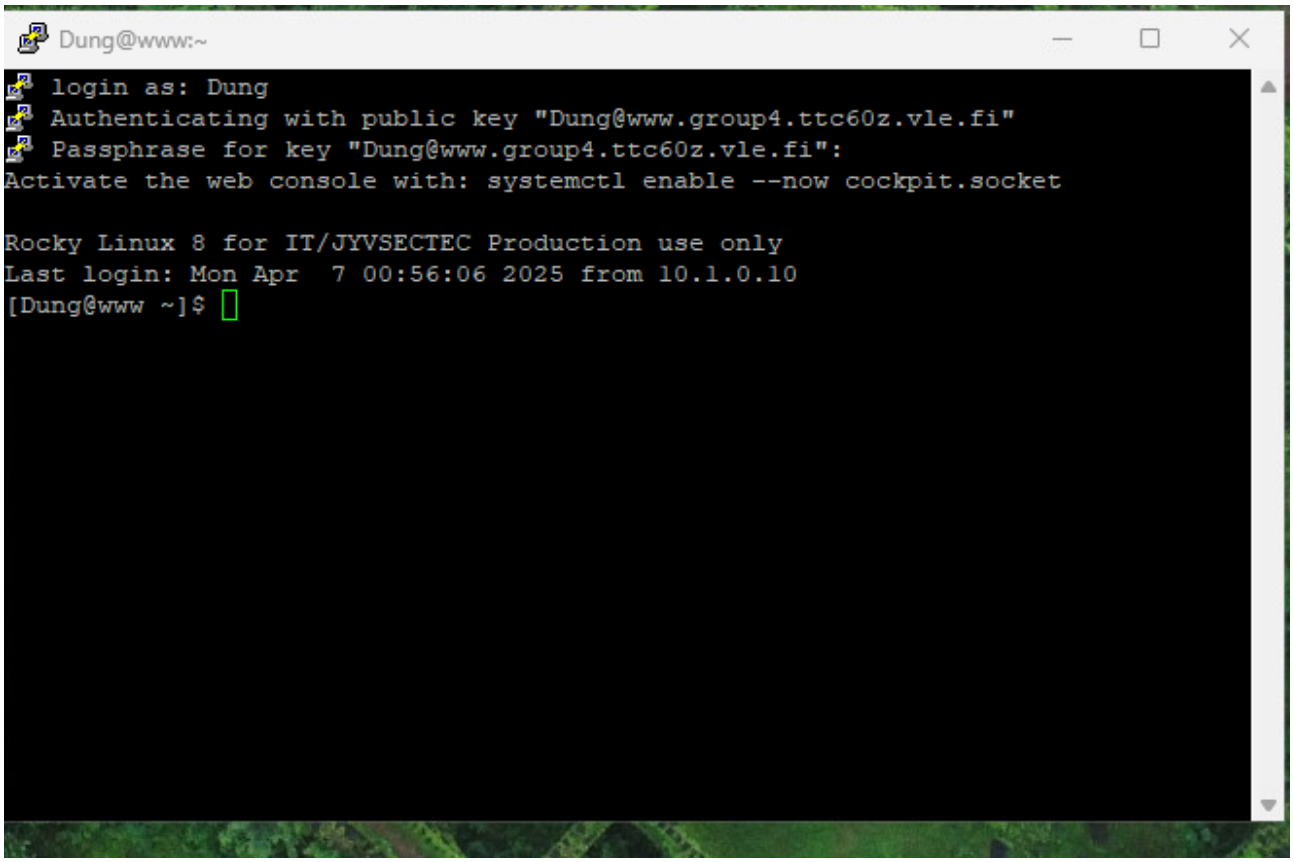


Figure 27 SSH login with SSH key

We updated the /etc/ssh/sshd_config file to enforce a login requirement that necessitates both a key and a password.(Figure 27)

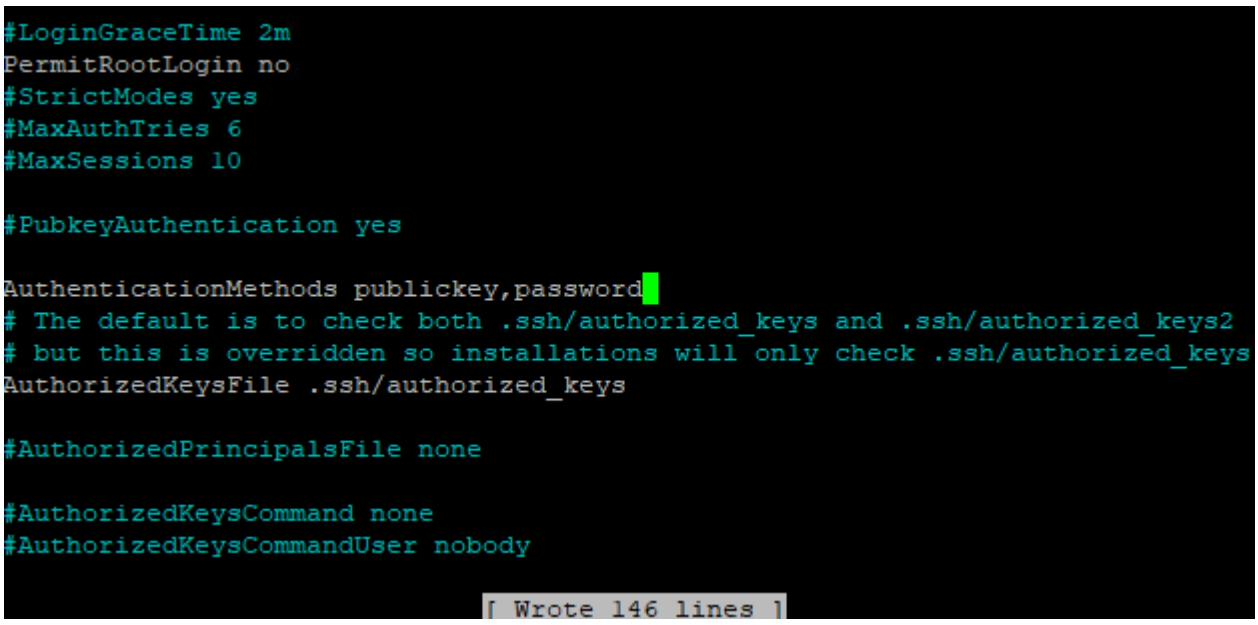


Figure 28 Editing sshd_config for key and password.

We restarted sshd with the command sudo systemctl restart sshd. After this, both the SSH key password and the user password were requested upon logging in.(Figure 28)

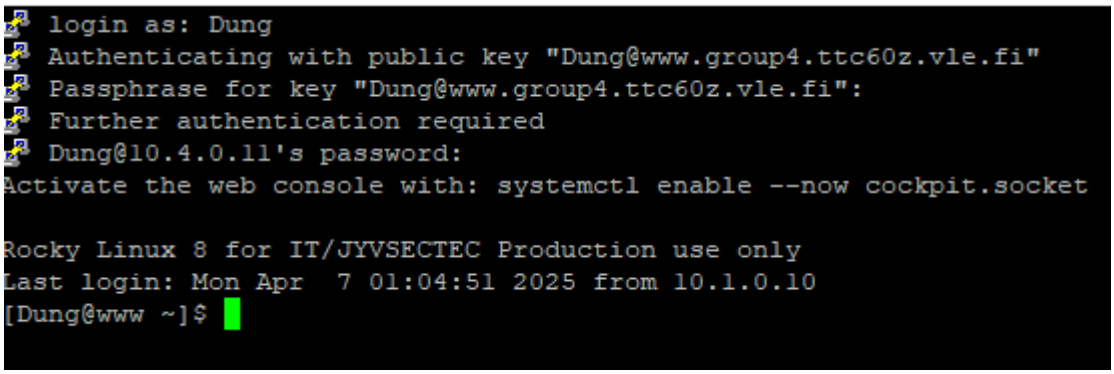


Figure 29 Logging in with a key and password.

We created other private keys for every users so that we can use different user account to log in.

3.3 Certbot

Certbot is an open-source, free tool designed to facilitate the automated use of Let's Encrypt certificates. It enables HTTPS by managing the acquisition and renewal of certificates seamlessly. In this laboratory exercise, we utilize Certbot to implement Let's Encrypt certificates, thereby securing a website with HTTPS. Let's Encrypt acts as a Certificate Authority (CA) that offers SSL/TLS certificates at no cost, ensuring website encryption is accessible to every organization.(Geeksfor-Geeks, 2020)

We set up Certbot by visiting the website cns.vle.fi and adhering to the guidelines provided there.(Figure 29)

CNS.VLE.FI

You can use certbot to request certificates for VLE virtual machines, but this requires the use of DNS-01 challenge. To generate correct DNS config and TSIG keys, enter the domain(s) you wish to validate:

Domain name(s):

www.group4.ttc60z.vle.fi
Create request

NOTE: For multiple domains enter *domain,domain* as a comma separated list, primary domain first.

Restrictions:

- Domain must be a subdomain under vle.fi
- When requesting multiple domain names, they must resolve to the same IP address

Requirements:

Make sure certbot and dns-rfc2136 plugin is installed. Commands for Centos 8:

```
yum install epel-release
yum install certbot python3-certbot-dns-rfc2136
```

Renewals

Most of the time, renewals are handled nowadays with systemd timers and `certbot-renew.service`. If you need to reload a webserver during renewal, use `--deploy-hook "systemctl reload webserversoftware"`.

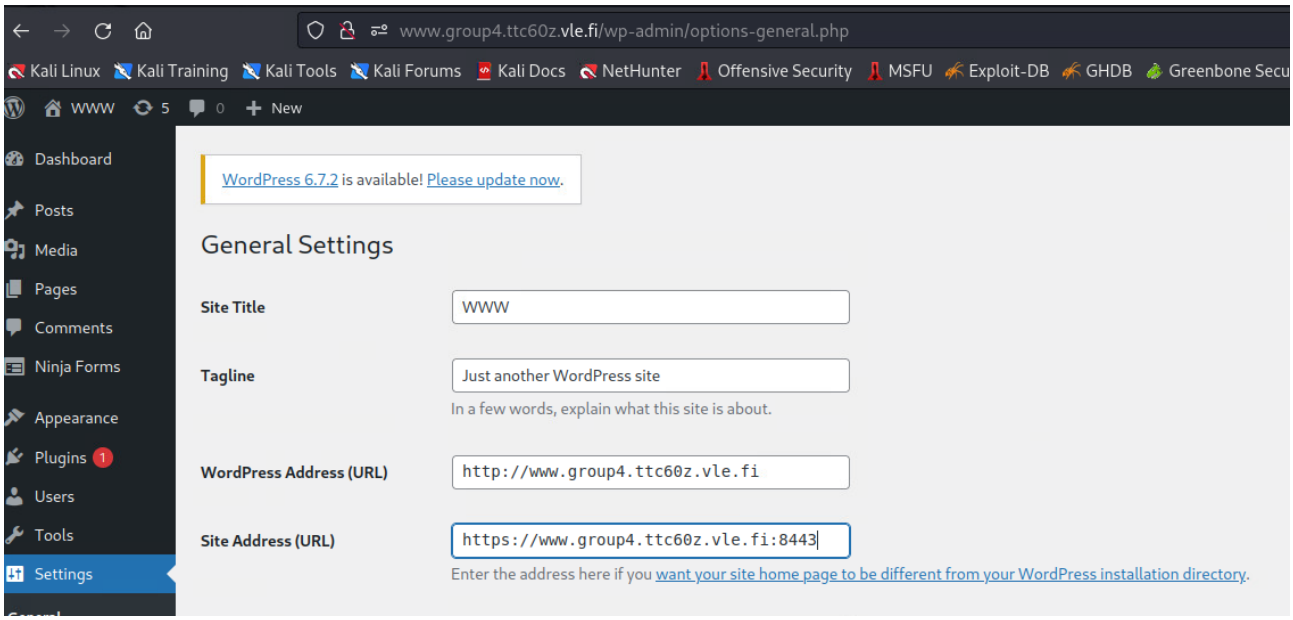
You can include this afterwards using `certbot renew --force-renew --deploy-hook ...` which will force a renewal and update the configuration files for that domain. See certbot documentation for more information about using hooks

Disclaimer

Please note that anyone with access to this system can request this information at a later date.
VLE systems using certificates requested with this information should NOT be used for production purposes.
For production certificates the information shown can be protected from subsequent reads (read-once), please contact VLE administration for more information.

Figure 30 Certbot guidelines

Prior to the Certbot installation, we disabled certain Palo Alto configurations that block the installation process. Additionally, we established a NAT rule within Palo Alto to permit traffic through port 8443 (see Figure 30). This setup is intended to allow external network access to websites through HTTPS on a public IP address. We opted for port 8443 for HTTPS because port 443 is occupied by the Global Protect VPN application.



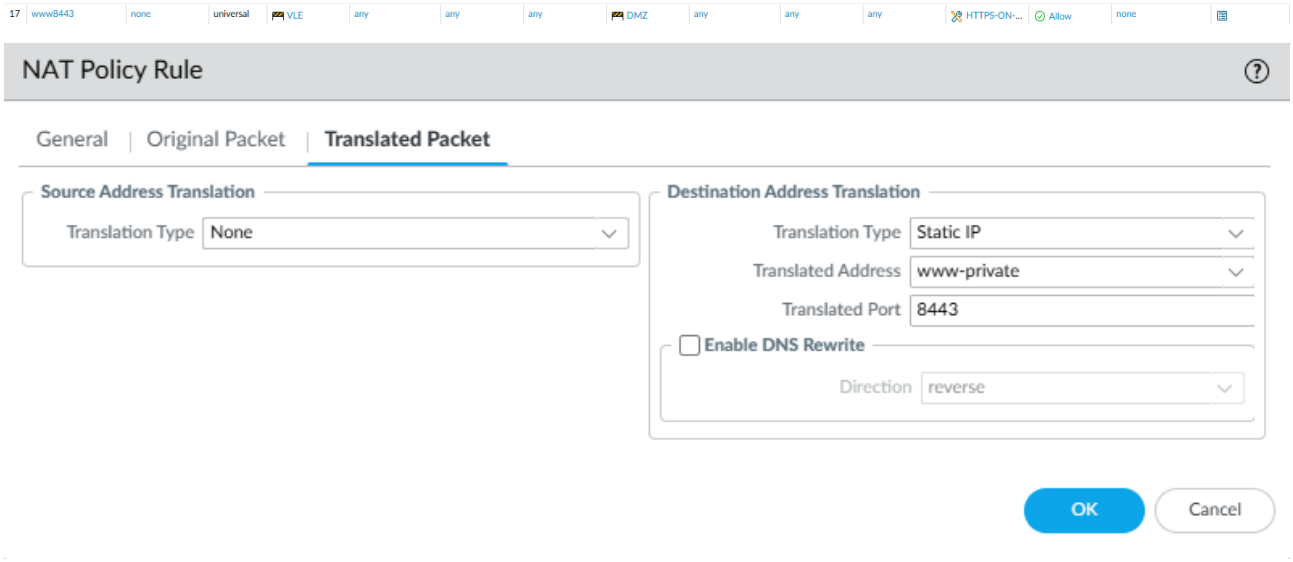


Figure 31 Enable Port 8443

In the initial phase of our project, we deactivated the root login feature. Later, we reactivated this feature and logged in as the root user. We navigated to the wordpress-docker directory on the

server and executed the command 'yum install epel-release'. (Figure 31)

```
[root@www wordpress-docker]# yum install epel-release
Last metadata expiration check: 3:22:08 ago on Mon 07 Apr 2025 09:59:53 AM EEST.
Package epel-release-8-13.el8.noarch is already installed.
Dependencies resolved.
================================================================================
 Package              Architecture    Version              Repository     Size
================================================================================
Upgrading:
 epel-release         noarch          8-21.el8             epel           24 k

Transaction Summary
================================================================================
Upgrade  1 Package

Total download size: 24 k
Is this ok [y/N]: y
Downloading Packages:
epel-release-8-21.el8.noarch.rpm               336 kB/s |  24 kB     00:00
--------------------------------------------------------------------------------
Total                                           80 kB/s |  24 kB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Running scriptlet: epel-release-8-21.el8.noarch                           1/1
  Upgrading        : epel-release-8-21.el8.noarch                           1/2
  Running scriptlet: epel-release-8-21.el8.noarch                           1/2
  Cleanup          : epel-release-8-13.el8.noarch                           2/2
  Running scriptlet: epel-release-8-13.el8.noarch                           2/2
  Verifying        : epel-release-8-21.el8.noarch                           1/2
  Verifying        : epel-release-8-13.el8.noarch                           2/2

Upgraded:
  epel-release-8-21.el8.noarch

Complete!
```

Figure 32 Install epel-release

Following that, we executed the command 'yum install certbot python3-certbot-dns-rfc2136'. (Figure 32)

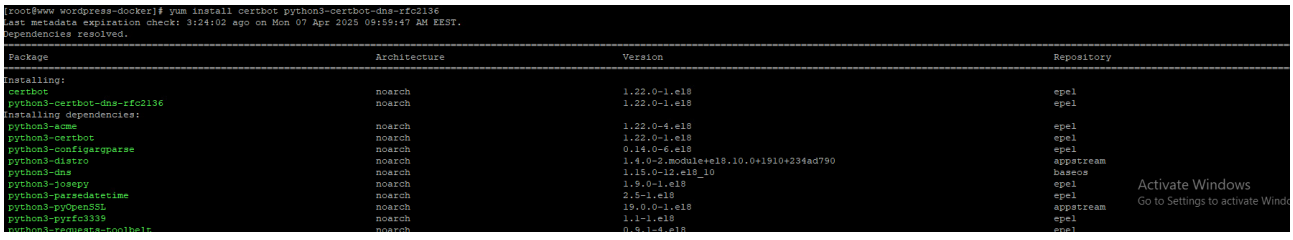


Figure 33 Certbot Installation

We continued according to the instructions on the cns.vle.fi site.(Figure 33)

Instructions for requesting a certificate

Make sure the following information is correct:

- FQDN: `www.group4.ttc60z.vle.fi`

Create configuration for certbot (/etc/pki/tls/rfc2136.ini):

```
dns_rfc2136_server = 198.18.100.7
dns_rfc2136_port = 53
dns_rfc2136_name = www.group4.ttc60z.vle.fi.
dns_rfc2136_secret = rixnVmtaNrFntaNaCz5UVTCnawiJyBDQap7LvsBdA6w=
dns_rfc2136_algorithm = HMAC-SHA256
```

Make sure the file has sane permissions:

```
chmod 600 /etc/pki/tls/rfc2136.ini
```

Finally, request a certificate:

```
certbot certonly --dns-rfc2136 --dns-rfc2136-credentials /etc/pki/tls/rfc2136.ini --dns-rfc2136-propagation-seconds 30 -d www.group4.ttc60z.vle.fi
```

If the request succeeds, certificates will be placed in `/etc/letsencrypt/`. You can add additional parameters such as `--apache` or `--nginx` to further automate certificate handling. See certbot documentation for more information on advanced use.

Figure 34 Certbot instruction

Following Figure 33's guidelines, we inserted specific lines into the /etc/pki/tls/rfc2136.ini file.

(Figure 34)

```
dns_rfc2136_server = 198.18.100.7
dns_rfc2136_port = 53
dns_rfc2136_name = www.group4.ttc60z.vle.fi.
dns_rfc2136_secret = rixnVmtaNrFntaNaCz5UVTCnawiJyBDQap7LvsBdA6w=
dns_rfc2136_algorithm = HMAC-SHA256
```

Figure 35 Rfc2136.ini

We set the permissions for the file using the command 'chmod 600 /etc/pki/tls/rfc2136.ini'.

We initiated a certificate request using the command 'certbot certonly --dns-rfc2136 --dns-rfc2136-credentials /etc/pki/tls/rfc2136.ini --dns-rfc2136-propagation-seconds 30 -d www.group4.ttc60z.vle.fi'. (Figure 35)

```
root@www wordpress-docker]# certbot certonly --dns-rfc2136 --dns-rfc2136-credentials /etc/pki/tls/rfc2136.ini --dns-rfc2136-propagation-seconds 30 -d www.group4.ttc60z.vle.fi
aving debug log to /var/log/letsencrypt/letsencrypt.log
nter email address (used for urgent renewal and security notices)
(Enter 'c' to cancel): AE8880@student.jamk.fi

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
lease read the Terms of Service at
ttps://letsencrypt.org/documents/LE-SA-v1.5-February-24-2025.pdf. You must
gree in order to register with the ACME server. Do you agree?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Y)es/(N)o: y

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
ould you be willing, once your first certificate is successfully issued, to
hare your email address with the Electronic Frontier Foundation, a founding
artner of the Let's Encrypt project and the non-profit organization that
evelops Certbot? We'd like to send you email about our work encrypting the web,
FF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Y)es/(N)o: y
ccount registered.
equesting a certificate for www.group4.ttc60z.vle.fi
aiting 30 seconds for DNS changes to propagate

uccessfully received certificate.
ertificate is saved at: /etc/letsencrypt/live/www.group4.ttc60z.vle.fi/fullchain.pem
ey is saved at:         /etc/letsencrypt/live/www.group4.ttc60z.vle.fi/privkey.pem
his certificate expires on 2025-07-06.
hese files will be updated when the certificate renews.
ertbot has set up a scheduled task to automatically renew this certificate in the background.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
f you like Certbot, please consider supporting our work by:
* Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
* Donating to EFF:                    https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Figure 36 Certification

We noted down the locations of the certificate files. The certificate file is stored at /etc/letsencrypt/live/www.group4.ttc60z.vle.fi/fullchain.pem and the key file at /etc/letsencrypt/live/www.group4.ttc60z.vle.fi/privkey.pem.

We configured the generated certificate for use with the modsecurity container by editing the docker-compose.yml file. Further adjustments included designating the ssl_port to translate www-server port 8443 to Docker container port 443. The /etc/letsencrypt/ directory has also been shared (volumes) with the Docker container, enabling it to retrieve the certificate and key through environment variables (PROXY_SSL_CERT, PROXY_SSL_KEY). The definitive modifications to the file correspond to what is depicted in Figure 36.

```
---
version: '3'
services:
  modsecurity:
    container_name: modsecurity
    image: owasp/modsecurity-crs:apache-alpine
    environment:
      PROXY: 1
      BACKEND: http://wordpress
      PORT: "8080"
      SSL_PORT: "443"
      METRICS_ALLOW_FROM: All
      PARANOIA: 1
      ANOMALY_INBOUND: 20
      PROXY_SSL_CERT: /etc/letsencrypt/live/www.group4.ttc60z.vle.fi/fullchain.pem
      PROXY_SSL_CERT_KEY: /etc/letsencrypt/live/www.group4.ttc60z.vle.fi/privkey.pem
    volumes:
      - /etc/letsencrypt:/etc/letsencrypt/
    ports:
      - "80:8080"
      - "8443:443"
    depends_on:
      - wordpress
```

Figure 37 Docker-compose.yml

We updated the adjusted yml file for use with the modsecurity container.(Figure 37)

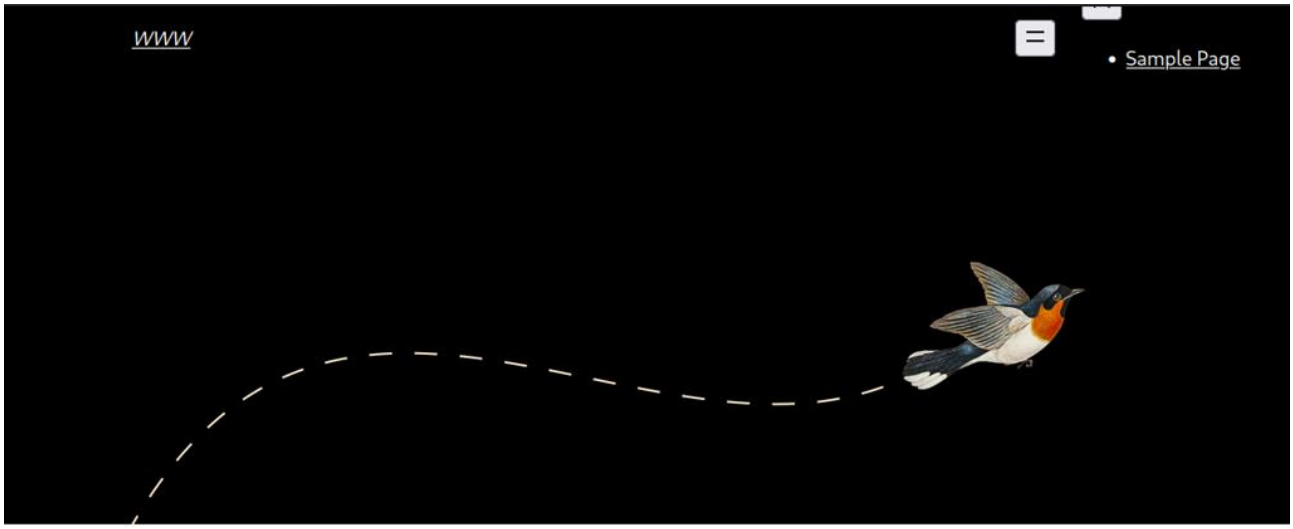
```
[root@www wordpress-docker]# docker-compose up -d
database is up-to-date
wordpress is up-to-date
Recreating modsecurity ... done
```

Figure 38 docker-compose up

The page loaded properly except for the images, which failed to load because they are sourced from an http page, resulting in mixed content warnings as depicted in Figure 38.

Figure 39 Mixed content

The site looks normal when accessed from my personal physical computer. (Figure 39)



Hello world!

Welcome to WordPress. This is your first post. Edit or delete it, then start writing!

January 17, 2025

Figure 40 8443 Wordpress site

3.4 Wordpress Hardening

We accessed the WordPress administration panel at the URL http://www.group4.ttc60z.vle.fi/wp-admin/. (Figure 40)

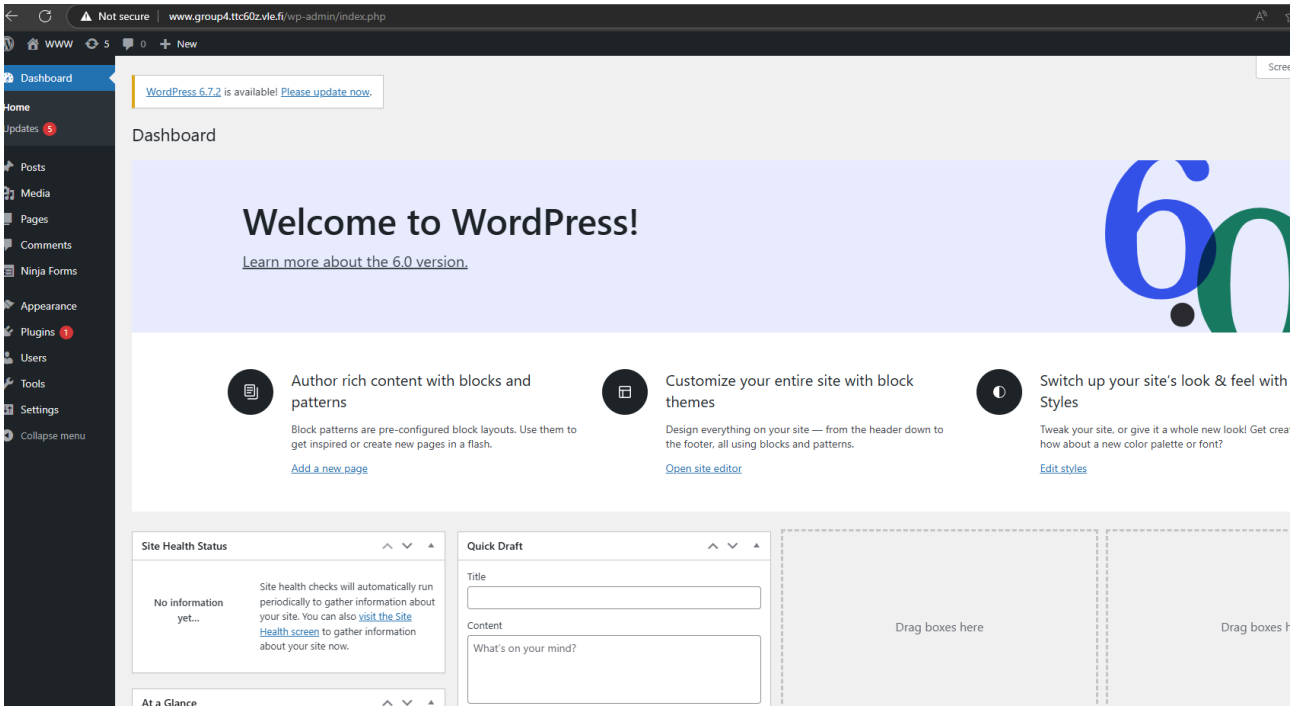


Figure 41 Wordpress admin panel

We began by reviewing the WordPress site for outdated plugins or other necessary updates. We discovered that the site was on version 6.0, while the current available version was 6.7.2, so we chose to upgrade. Prior to the upgrade, we backed up the database using the command 'docker exec -it database mysqldump -uroot -proot66 wordpress > backup.sql'. Updating WordPress is vital, as older versions may have SQL injection and cross-site scripting vulnerabilities, common threats in WordPress environments. Such updates are essential to fortify security

Following the backup, we proceeded to install the updates by selecting 'Update to version 6.7.2.(Figure 41)

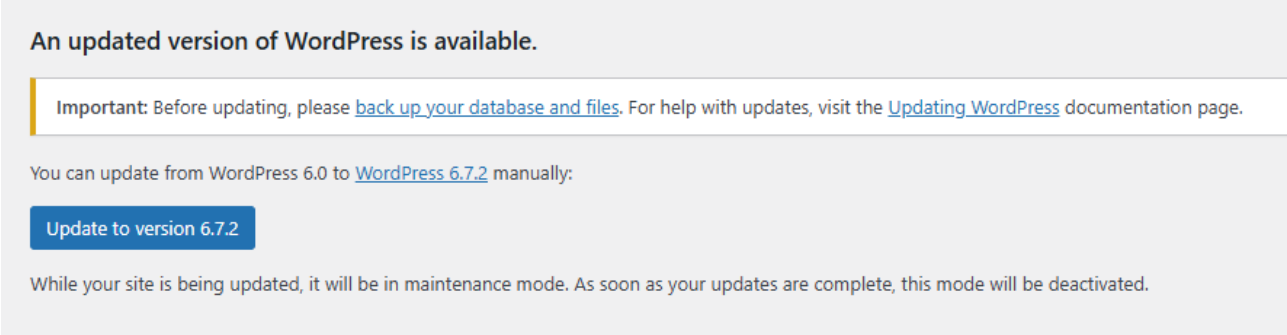


Figure 42 Version Update

The update failed, and we received a notification in accordance with figure 42.

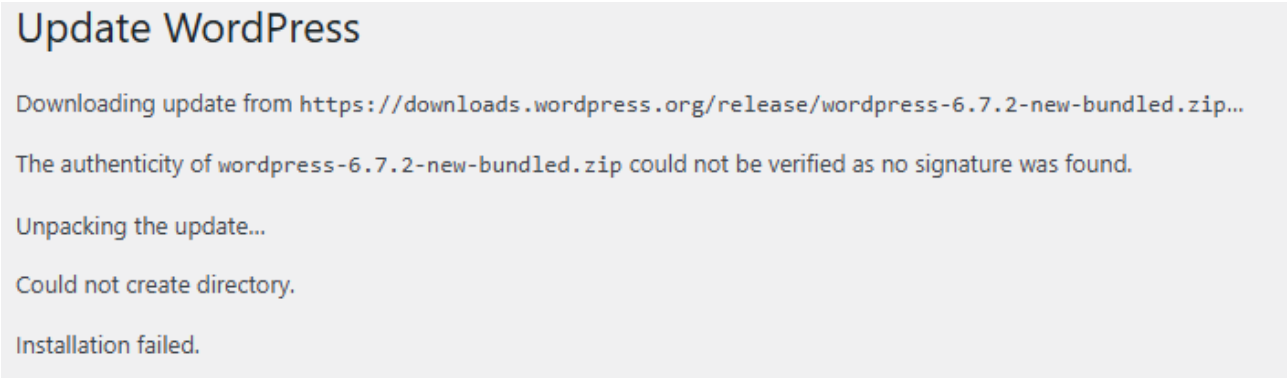


Figure 43 Failed Update

We set permissions for the content within the /var/www/html directory inside the WordPress container. However, this did not solve the problem, as the same error message persisted. After examining the permissions more closely, we noticed some were still absent. Consequently, we applied additional permissions and set the ownership to www-data for the upgrade and uploads directories in the wp-content folder with the command chown -R www-data:www-data

/var/www/html/wp-content. (Figure 43)

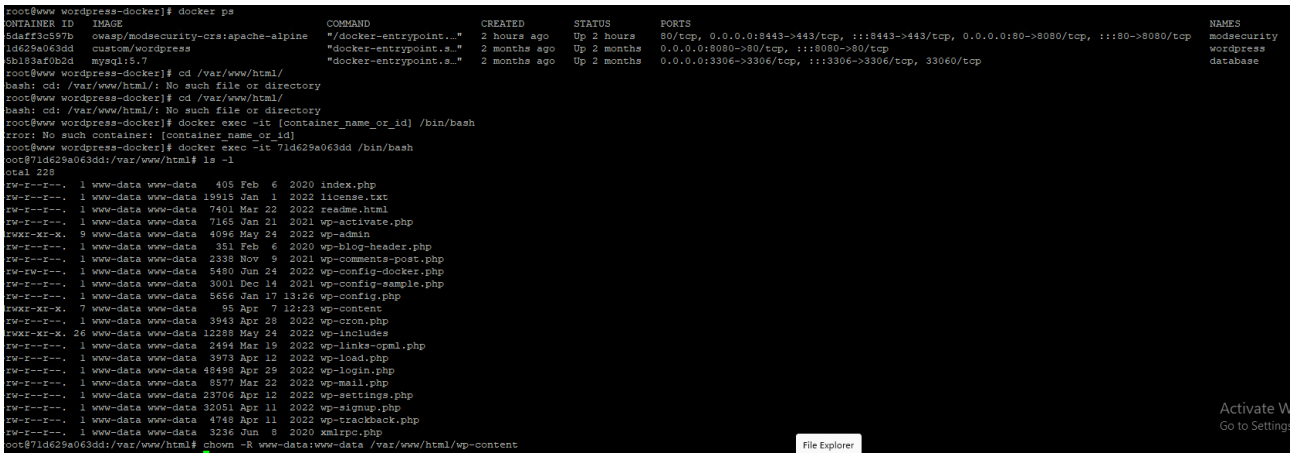


Figure 44 set permisson

Following the assignment of permissions and ownership, we attempted to update again through the admin panel, and this time the update succeeded. WordPress subsequently prompted us to update the database. (Figure 44)

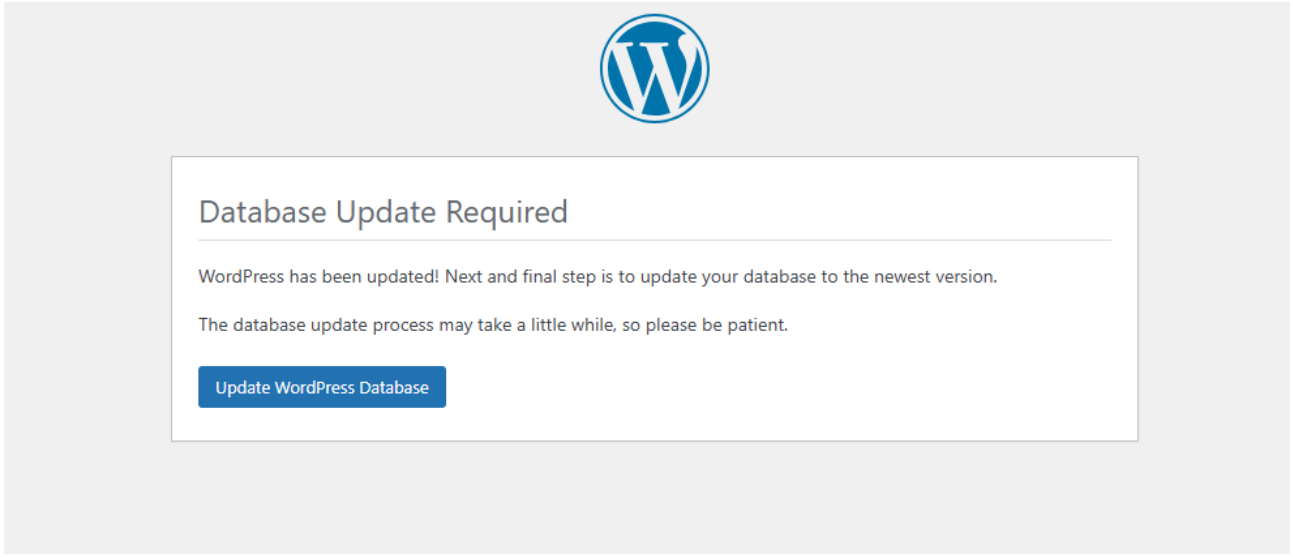


Figure 45 Database update

After selecting 'Update WordPress Database,' we were automatically taken to a page that described the latest WordPress version.(Figure 45)

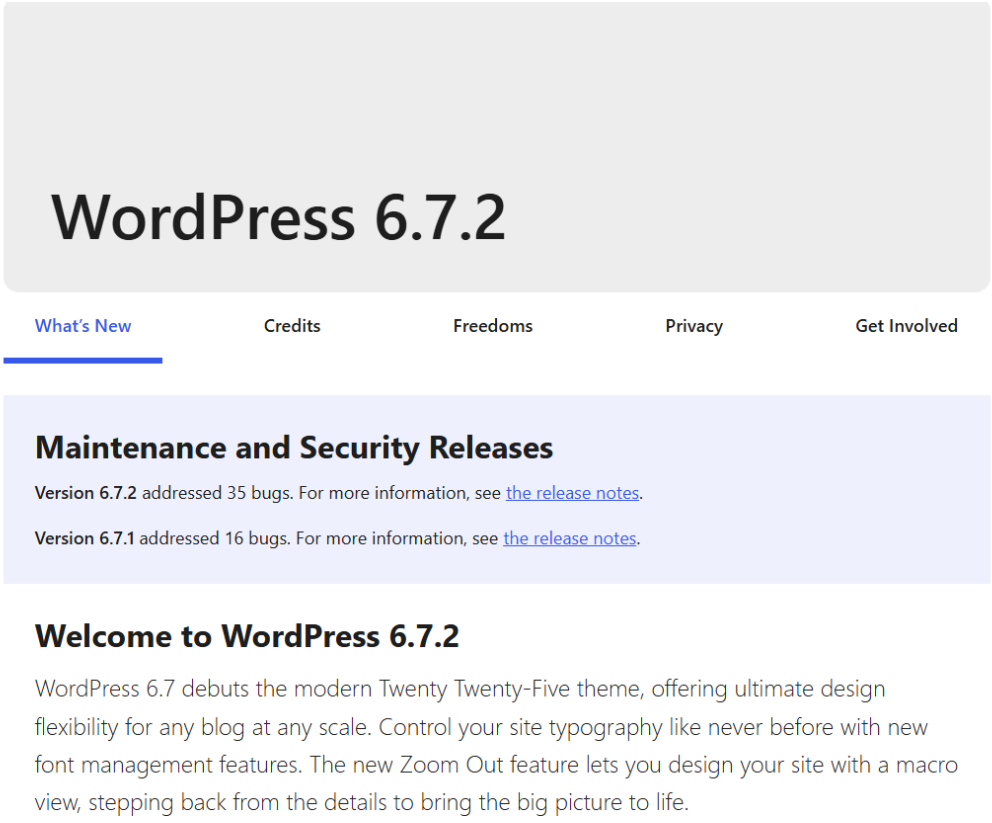


Figure 46 Wordpress 6.7.2

We then reviewed the status of the installed plugins and found that the Akismet Anti-Spam plugin was out of date, which led us to update it. (Figure 46)

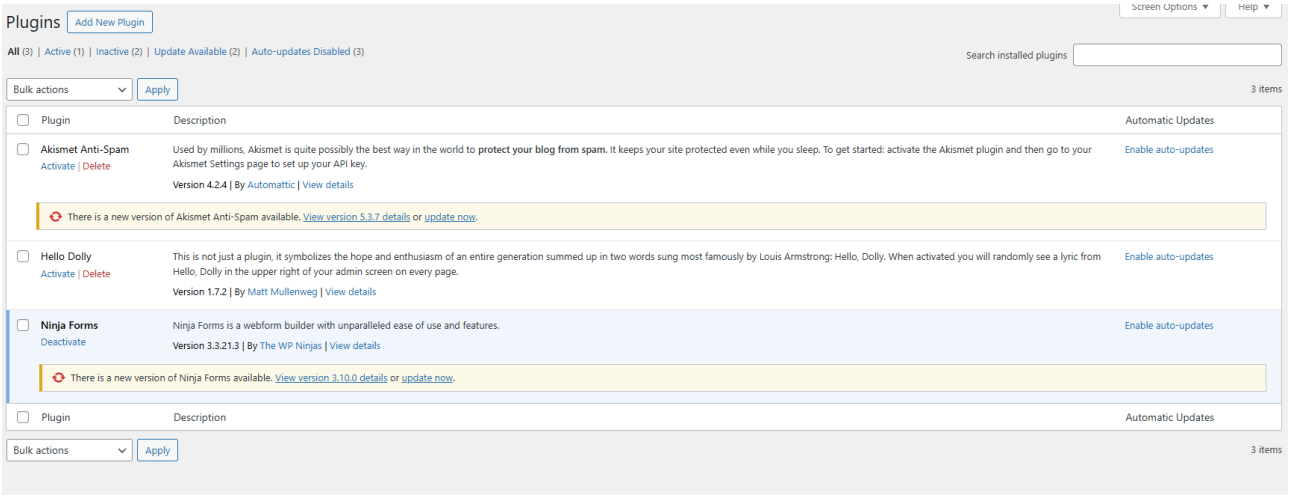


Figure 47 WordPress Plugins

The update went through successfully.(Figure 47)

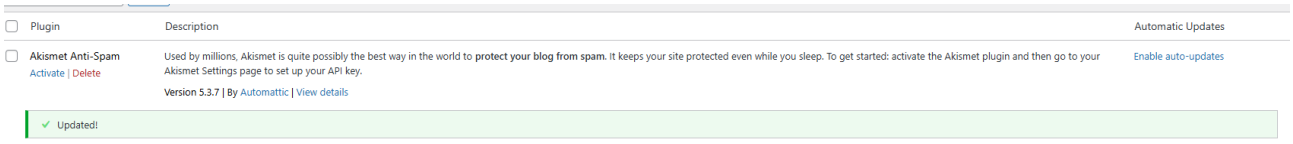


Figure 48 Akismet updated

We also updated the Ninja Forms plugin.(Figure 48)

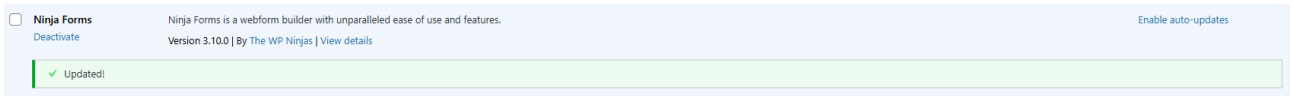


Figure 49 Ninja Forms Update

The Hello Dolly plugin, which has no practical function, was pre-installed in WordPress. We decided to uninstall it, reasoning that any unnecessary components could potentially be a security risk.(Figure 49)

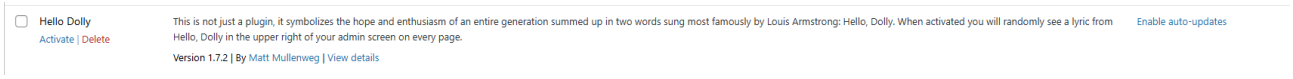


Figure 50 Hello Dolly Removal

We discovered that our WordPress site contained outdated themes. Given that these outdated themes pose significant security risks, we proceeded to update them.(Figure 50)

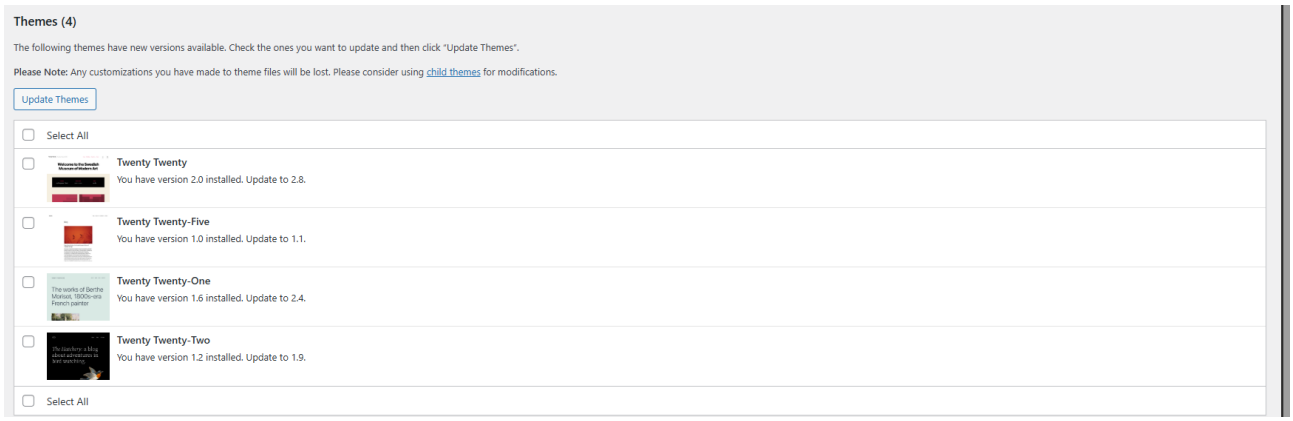


Figure 51 Outdated themes

We updated the themes went smoothly.(Figure 51)

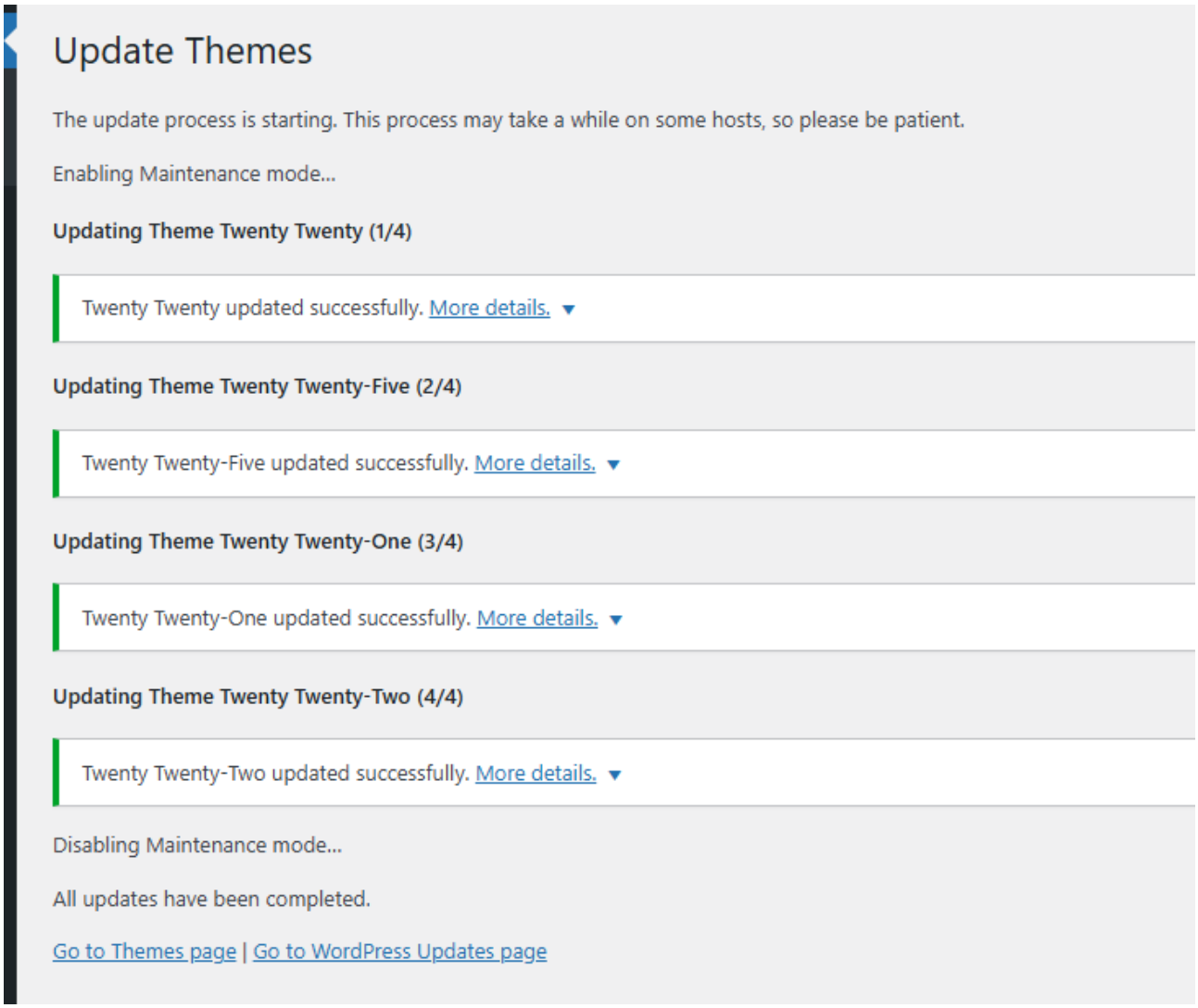


Figure 52 Themes updated

Although unused themes can pose security risks, we chose not to remove them yet, allowing for the possibility that we might use them later.

3.5 Docker Hardening

Next, we focused on hardening Docker containers. This was achieved by implementing SELinux and installing Docker to operate in rootless mode, meaning without root user privileges.

3.5.1 SELinux

We enabled SELinux by modifying the /etc/docker/daemon.json file according to Figure 52.

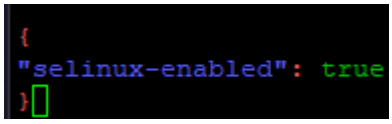


Figure 53 Enabling selinux

After rebooting the Docker service, we verified through the 'docker info' command that SELinux had been successfully activated.(Figure 53)

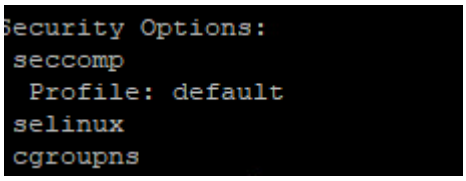


Figure 54 SELinux enabled

To further assess functionality, we conducted tests with BusyBox. We tried adding the user BADUSER to the /host_shadow file and subsequently attempted to access this file.(Figure 54)

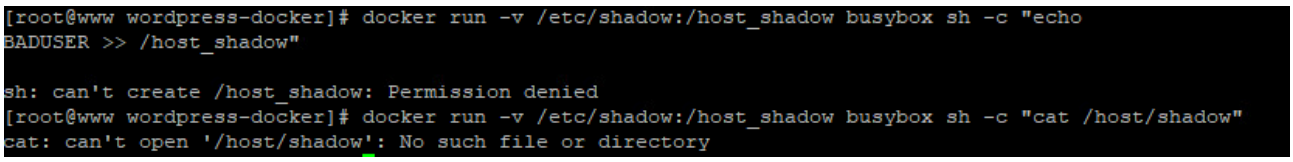


Figure 55 SELinux Busy Box Test

The 'aureport -a' command displayed the attempted user addition.(Figure 55)

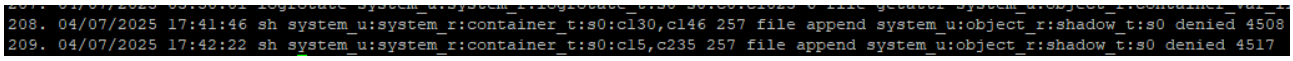


Figure 56 auereport -a

3.5.2 Rootless

For this purpose, we created a user testuser who does not have root privileges.(Figure 56)

```
[root@www wordpress-docker]# sudo adduser testuser
[root@www wordpress-docker]# sudo usermod -L testuser
[root@www wordpress-docker]# groups testuser
testuser : testuser
```

Figure 57 Creating a Testuser

Initially, we required the newuidmap and newgidmap packages. We executed the command 'sudo yum install -y shadow-utils' to install these packages and then checked to ensure their installation was successful.(Figure 57)

```
[root@www wordpress-docker]# which newuidmap
/usr/bin/newuidmap
[root@www wordpress-docker]# which newgidmap
/usr/bin/newgidmap
```

Figure 58 Checking Installations

According to Dockerhub's guidelines, we verified that the test user possesses an adequate amount of userIDs and groupIDs.(Figure 58)

```
[testuser@www wordpress-docker]$ whoami
testuser
[testuser@www wordpress-docker]$  grep ^$(whoami): /etc/subuid
testuser:493216:65536
[testuser@www wordpress-docker]$ grep ^$(whoami): /etc/subgid
testuser:493216:65536
[testuser@www wordpress-docker]$
```

Figure 59 UID and GID test

We executed the command 'sudo dnf install -y fuse-overlayfs' to install the fuse-overlayfs packages, after which we closed down the Docker services(Figure 59)

```
  Preparing        :
  Upgrading        : fuse-overlayfs-1.13-1.module+el8.10.0+1948+4b5cd4a9.x86_64
  Running scriptlet: fuse-overlayfs-1.13-1.module+el8.10.0+1948+4b5cd4a9.x86_64
  Cleanup          : fuse-overlayfs-1.7.1-1.module+el8.5.0+710+4c471e88.x86_64
  Running scriptlet: fuse-overlayfs-1.7.1-1.module+el8.5.0+710+4c471e88.x86_64
  Verifying        : fuse-overlayfs-1.13-1.module+el8.10.0+1948+4b5cd4a9.x86_64
  Verifying        : fuse-overlayfs-1.7.1-1.module+el8.5.0+710+4c471e88.x86_64

Upgraded:
  fuse-overlayfs-1.13-1.module+el8.10.0+1948+4b5cd4a9.x86_64

Complete!
```

```
[Nguyen@www wordpress-docker]$ sudo systemctl disable --now docker.service docker.socket
[sudo] password for Nguyen:
Removed /etc/systemd/system/multi-user.target.wants/docker.service.
```

Figure 60 Disable Docker

We created a backup of the database to use it at a later time.(Figure 60)

```
[root@www wordpress-docker]# sudo docker exec -it database mysqldump -uroot -pro
ot66 wordpress > backup.sql
```

```
[root@www wordpress-docker]# ls
backup.sql  docker-compose.yml  Dockerfile
```

Figure 61 Backup of the database

We lacked the iptables module, so we installed it with the root user.(Figure 61)

```
[root@www wordpress-docker]# lsmod | grep ip_tables
[root@www wordpress-docker]# sudo modprobe ip_tables
```

Figure 62 Iptables module installation

We compressed each container into a separate .tar file with the command 'docker save <container name> > container.tar' and transferred these files to the home directory of the user named tes-tuser.

We stopped the containers using the 'docker-compose down' command, and then we ran a com-mand according to what is illustrated in Figure 49.

```
[root@www wordpress-docker]# sudo systemctl disable --now docker.service docker.socket
[root@www wordpress-docker]# sudo rm /var/run/docker.sock
```

Figure 63 Disable Docker

We logged into the testuser account and ran the installation command for rootless mode using dockerd-rootless-setuptool.sh install.(Figure 63)

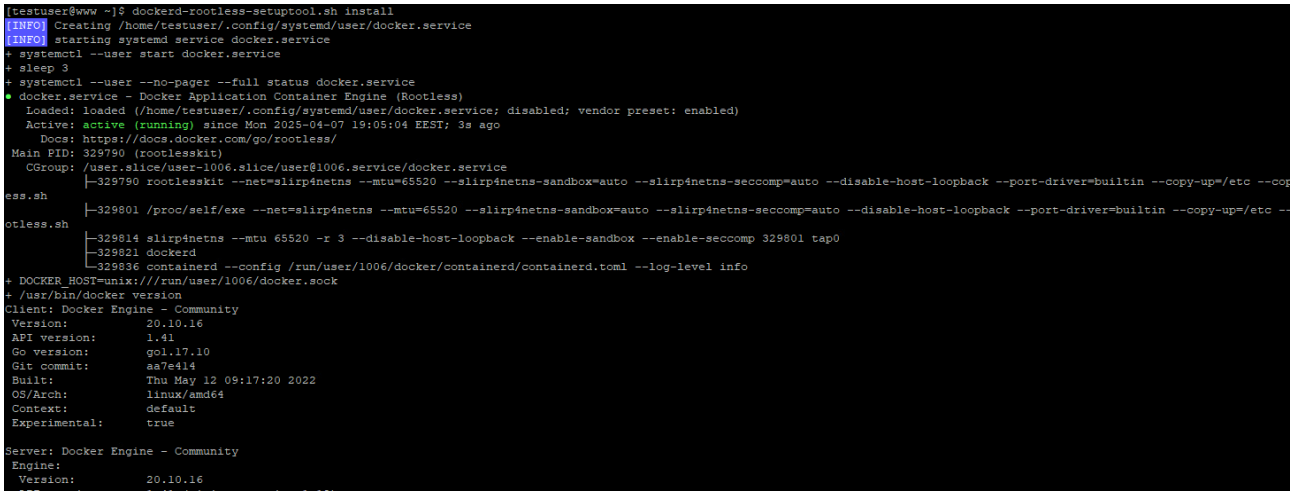
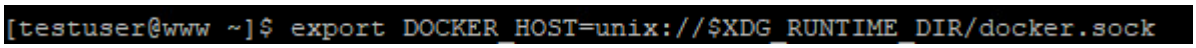


Figure 64 Rootless installation

We set up the Docker socket path following the instructions and commands detailed in Figure 64.



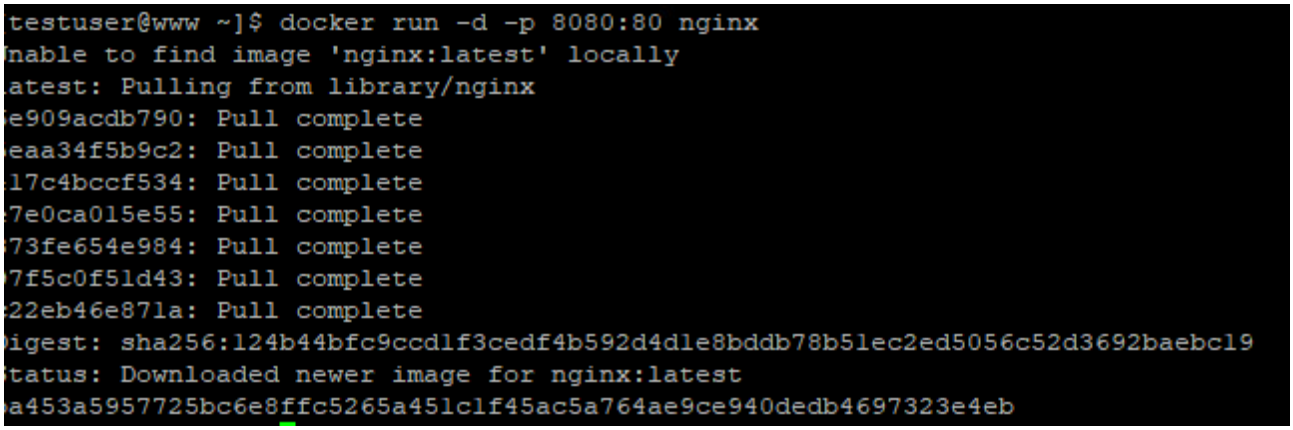


Figure 65 Configure Docker.socket

After restarting the containers with the docker-compose up command, this resulted in errors similar to those described in figure 65, apparently caused by the contents of the /var/lib/mysql directory, which should be empty.

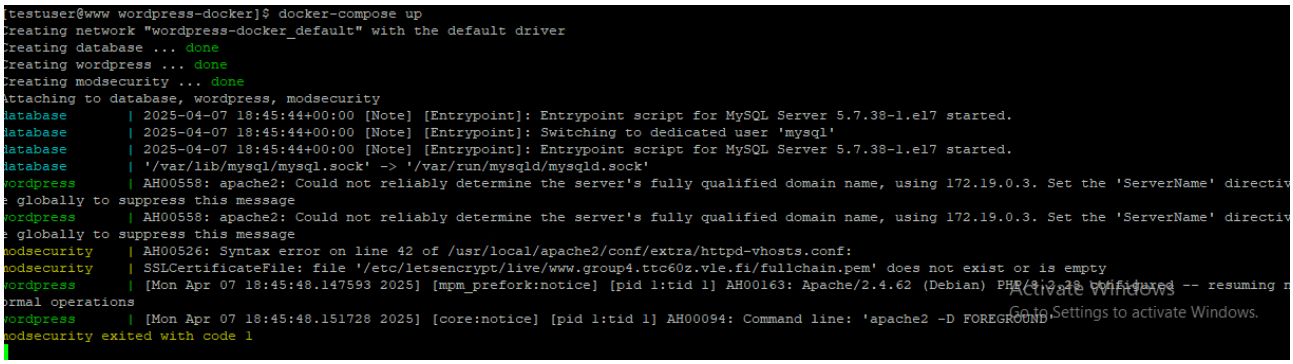


Figure 66 Error log

We added permissions for the testuser with the command sudo chmod -R o+rX /etc/letsencrypt. After this, the containers started without any problems, but the pages were not visible. We decided to give up with rootless and asked the lab team to reset the WWW server. So we run docker with the root user, which is not the most secure option, but time pressure hit us with other lab work.

4 Conclusion

This lab work for this course module was the most challenging so far. Some tasks were easier than others, and there were good instructions and tips on how to proceed. Particularly, implementing Docker in rootless mode was challenging. The course materials did not provide adequate guidance for this, which was frustrating and slowed down the process. Even Docker's own guidelines seemed somewhat incomplete, providing some help but not much in troubleshooting. We extensively searched the internet and investigated potential issues with getting the site to work with rootless Docker. Docker had been discussed only briefly and superficially in previous courses, so our group did not have much expertise, which undoubtedly slowed down and complicated the work. However, we became quite familiar with it during the lab work.

The other parts of the lab work went relatively smoothly for our group and were educational. We could have gone further with hardening WordPress and the www-server on Linux, but we believe we made sufficient enhancements. We learned the basic concepts of using both and can handle them proficiently in the future.

Overall, the lab work was interesting, even though the rootless setup caused a considerable amount of frustration. In addition to the assigned tasks, we learned that not everything can be completed within a limited time. It is possible that we will encounter rootless in the future, and achieving success with it will be particularly rewarding.

References

Mallory, P., 2020. Web server security: Web server hardening. Accessed on April 5, 2025. https://www.infosecinstitute.com/resources/network-security-101/web-server-security-web-server-hardening/

Schrader, D., 2023. What Is System Hardening?. Accessed on April 5, 2025. What is System Hardening and Why is it Important?

Juviler, J., 2024. 14 WordPress Security Issues & Vulnerabilities You Should Know About. Accessed April 5, 2025. 14 WordPress Security Issues & Vulnerabilities You Should Know About [New Research from WCEU 2022]

Zorz, M., 2024. Lynis: Open-source security auditing tool. Accessed on April 5, 2025. Lynis: Open-source security auditing tool - Help Net Security

Verhage, R., 2024. Why Should We Disable Root-Login Over SSH?. Accessed on April 5, 2025. Why Should We Disable Root-Login Over SSH? | Baeldung on Linux

Forest, 2019. What are SHA-rounds?. Accessed April 5, 2025. passwords - What are SHA-rounds? - Information Security Stack Exchange

GeeksforGeeks, 2020. Using Certbot Manually for SSL certificates. Accessed on April 5, 2025. Using Certbot Manually for SSL certificates | GeeksforGeeks

Zivanov, S., 2023. What Is SELinux (Security-Enhanced Linux)?. Accessed on April 5, 2025. What Is SELinux (Security-Enhanced Linux)?

Editorial Staff, 2024. What Are WordPress Plugins? And How Do They Work?. Accessed on April 6, 2025. What Are WordPress Plugins? And How Do They Work?

Jack, W., 2025. How To Run Docker in Rootless Mode. Accessed on April 6, 2025. How To Run Docker in Rootless Mode - The New Stack