



Hardening- Lab 5

Group 4

Nguyen Ngo - AE8880

Dung Doan - AA7785

Syed Fawaz - AD9946

Jasper (Franciscus) van de Klundert - AG9056

Sanka De Silva - AC4892

Exercise Lab 5

Hardening

21/4/2025

Bachelor's Program of Information and Communication Technology

Contents

| | | |
|------------|-------------------------------------------------------|-----------|
| 1 | Introduction | 4 |
| 2 | Theory of the Lab..... | 4 |
| 2.1 | MFA..... | 4 |
| 2.2 | Google Authenticator | 5 |
| 3 | Lab Work Progress | 5 |
| 3.1 | Adding 2fa plugin to Wordpress container | 5 |
| 3.2 | Adding 2FA to Centos | 13 |
| 4 | Conclusion | 20 |
| 4.1 | Reflection | 21 |
| | References | 22 |

Figures

| | |
|--------------------------------------------------------------------------------------|----|
| Figure 1 VLE Environment | 4 |
| Figure 2 WP 2FA installation | 6 |
| Figure 3 Successfully installed..... | 7 |
| Figure 4 Setup wizard..... | 7 |
| Figure 5 Configure methods and Policies | 8 |
| Figure 6 All user enforcement..... | 9 |
| Figure 7 Grace Period Choice | 10 |
| Figure 8 QR code for implementing the application..... | 11 |
| Figure 9 Backup code generate..... | 11 |
| Figure 10 Authentication Code Requirement | 12 |
| Figure 11 Instructions for Installing multi-factor authenticator for ssh login | 13 |
| Figure 12 Time-based tokens..... | 13 |
| Figure 13 QR code on a web server | 14 |
| Figure 14 Emergency codes | 15 |
| Figure 15 Security settings | 15 |

| | |
|----------------------------------------------------------------------------|----|
| Figure 16 ssh-keygen command | 16 |
| Figure 17 id_rsa..... | 17 |
| Figure 18 Puttygen | 18 |
| Figure 19 PubkeyAuthentication | 18 |
| Figure 20 UsePam setting | 19 |
| Figure 21 ChallengeResponseAuthentication..... | 19 |
| Figure 22 AuthenticationMethods..... | 19 |
| Figure 23 Editing the pam.d/sshd file | 19 |
| Figure 24 Multi-factor authentication when opening an SSH connection | 20 |

1 Introduction

In this lab, the purpose of this exercise is to explore **Multi-Factor Authentication (MFA)** and configure it for use with **WordPress** and **SSH login** to the web server. The exercise will be carried out in the **VLE environment**, as illustrated in **Figure 1**.

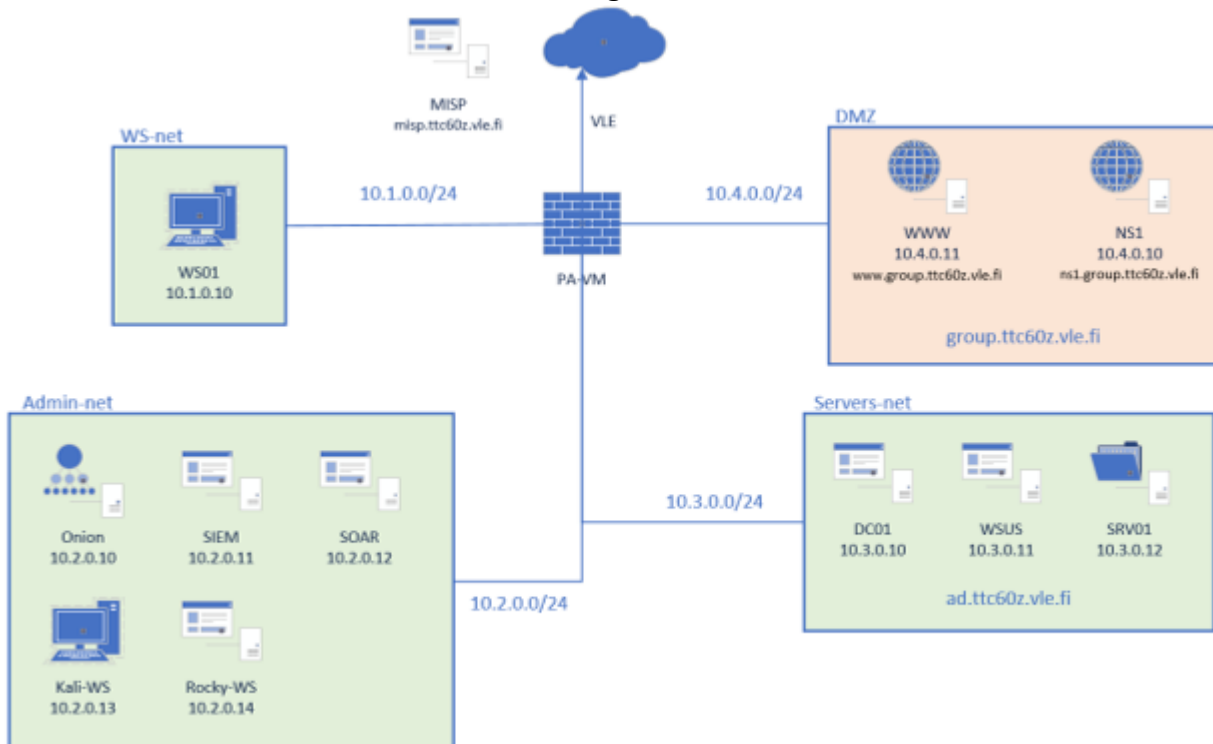


Figure 1 VLE Environment

2 Theory of the Lab

2.1 MFA

Multi-Factor Authentication (MFA) is a security measure that verifies a user's identity using two or more authentication methods. Typically, this means that in addition to a username and password, a third form of authentication is required—such as a code sent to the user's phone or an approval via a mobile app. (Stephen, 2025)

The main goal of MFA is to prevent unauthorized access to user accounts and enhance overall security. Even if a password is compromised, an attacker cannot log in without the additional authentication method. This significantly reduces the risk of phishing attacks and password breaches. (Eliza, 2025)

MFA is especially important for services that handle sensitive information, such as email accounts or internal company systems. It is an effective way to protect user accounts and minimize their vulnerability to cyberattacks. (Stephen, 2025)

2.2 Google Authenticator

In both parts of the lab, we'll use the **Google Authenticator** app to enable multi-factor authentication. Once configured for **WordPress** or **SSH**, logging in will require not only a password or key but also a code generated by the app on a mobile device.

Google Authenticator generates **one-time passwords (OTP)** using two algorithms: **HOTP (Event-based)** and **TOTP (Time-based)**.

1. Secret Key

Both the server and the client (Google Authenticator) share a **secret key**, securely stored on both sides. This key is usually provided in the form of a **QR code**.

2. HOTP (Event-based)

HOTP uses the secret key along with a **counter**. Every time a new password is generated and used, the counter increases. Both the server and client stay in sync by tracking this counter.

3. TOTP (Time-based)

TOTP also uses the secret key, but instead of a counter, it relies on the **current time**. Both the client and server synchronize time—typically using **Network Time Protocol (NTP)**—to ensure the generated codes match.

4. Code Generation

Google Authenticator generates a 6-digit one-time password by combining the secret key with either the time or counter through a secure algorithm. The user enters this code during login, and the server verifies it by calculating the same code.

5. Security

Because the secret key and time/counter are shared securely, both the client and server can generate matching codes **without needing to communicate directly** during login. This makes the system highly secure.

(Udoh, 2023)

3 Lab Work Progress

3.1 Adding 2fa plugin to Wordpress container

We began adding multi-factor authentication to WordPress by logging into the WordPress admin dashboard at:

www.ttc60z.vle.fi

From the **Plugins** tab, we searched for “**WP 2FA**” and clicked “**Install Now**” to install the plugin (see **Figure 2**).

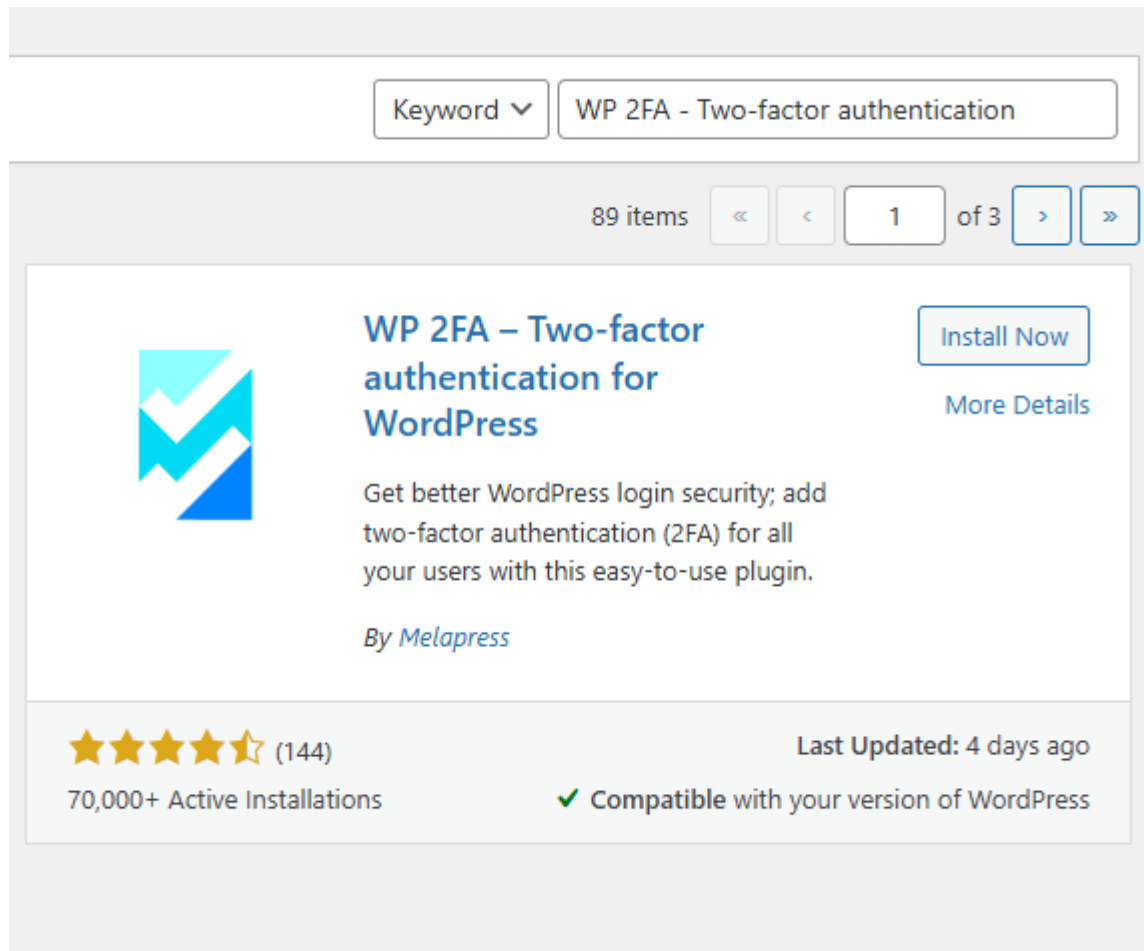
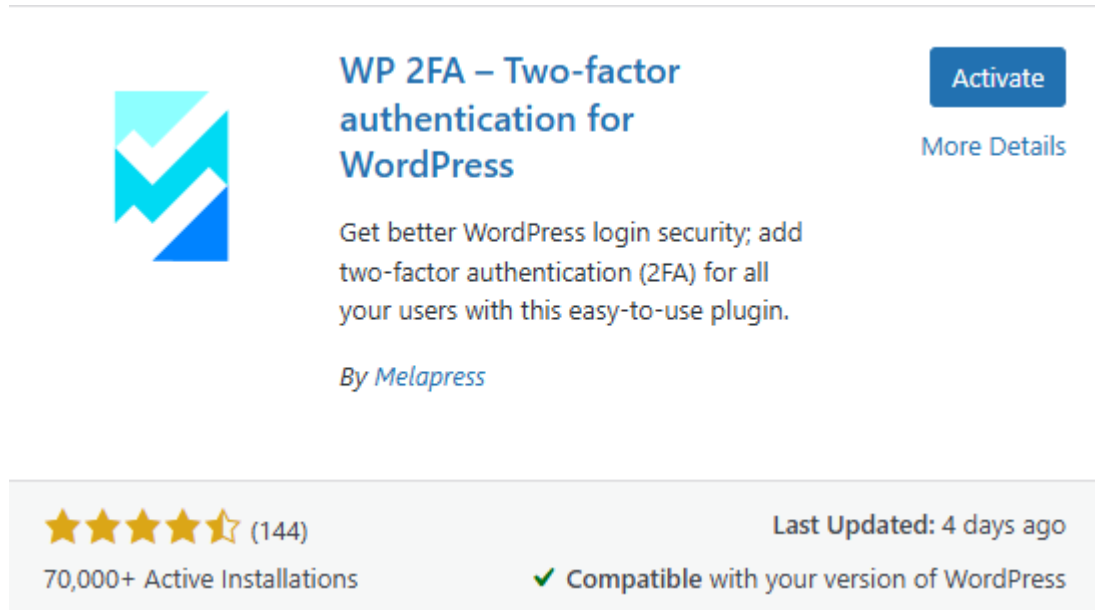


Figure 2 WP 2FA installation

According to the instructions, we were supposed to encounter an error at this stage stating that the folder could not be created. However, we had already completed this step during a previous

exercise when updating plugins. (Figure 3)



The screenshot shows the WordPress plugin interface for 'WP 2FA – Two-factor authentication for WordPress'. On the left is a blue and white geometric logo. To its right, the plugin title is displayed in a large, bold, blue font. Below the title is a short description: 'Get better WordPress login security; add two-factor authentication (2FA) for all your users with this easy-to-use plugin.' and the developer's name 'By Melapress' in a smaller blue font. In the top right corner, there is a blue 'Activate' button and a link for 'More Details'. At the bottom, a light gray box contains the plugin's rating (5 stars, 144 reviews), the number of active installations ('70,000+ Active Installations'), and a green checkmark indicating it is 'Compatible with your version of WordPress', along with the update date 'Last Updated: 4 days ago'.

WP 2FA – Two-factor authentication for WordPress

Get better WordPress login security; add two-factor authentication (2FA) for all your users with this easy-to-use plugin.

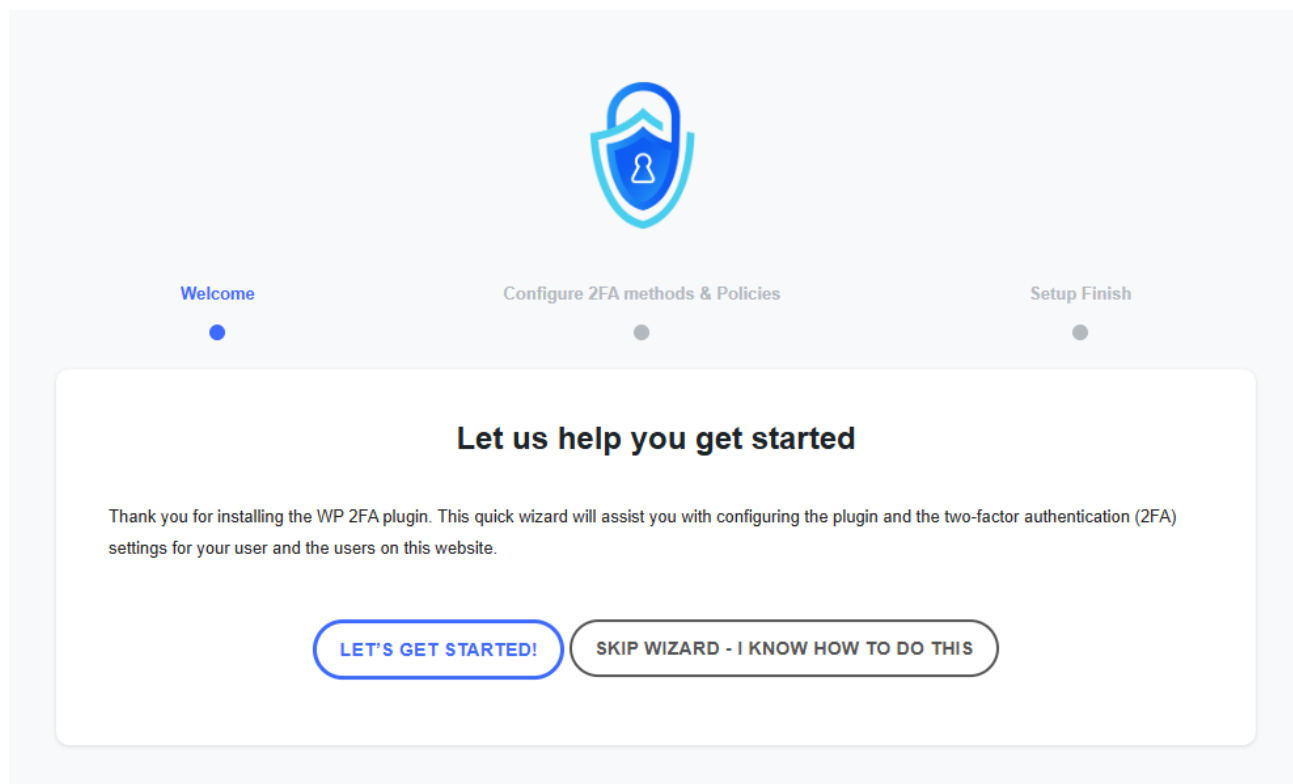
By Melapress

★★★★★ (144)
70,000+ Active Installations

Last Updated: 4 days ago
✓ Compatible with your version of WordPress

Figure 3 Successfully installed

After installing the plugin, we clicked activate and automatically moved to the Setup Wizard. (Figure 4)



The screenshot displays the 'Setup Wizard' for the WP 2FA plugin. At the top center is a blue shield icon with a white keyhole. Below the icon is a progress bar with three steps: 'Welcome' (active, indicated by a blue dot), 'Configure 2FA methods & Policies' (inactive, indicated by a gray dot), and 'Setup Finish' (inactive, indicated by a gray dot). The main content area has a white background with the heading 'Let us help you get started'. Below this heading is a paragraph: 'Thank you for installing the WP 2FA plugin. This quick wizard will assist you with configuring the plugin and the two-factor authentication (2FA) settings for your user and the users on this website.' At the bottom of the white area are two buttons: 'LET'S GET STARTED!' (highlighted with a blue border) and 'SKIP WIZARD - I KNOW HOW TO DO THIS' (with a gray border).

Welcome Configure 2FA methods & Policies Setup Finish

Let us help you get started

Thank you for installing the WP 2FA plugin. This quick wizard will assist you with configuring the plugin and the two-factor authentication (2FA) settings for your user and the users on this website.

LET'S GET STARTED! SKIP WIZARD - I KNOW HOW TO DO THIS

Figure 4 Setup wizard

In the next step, we did not enable email verification—instead, we only used **one-time codes** generated by the **authenticator app**. (Figure 5)

Welcome **Configure 2FA methods & Policies** Setup Finish

① **2FA methods** ② Alternative methods ③ 2FA policy

Which 2FA methods can your users use?

When you uncheck any of the below 2FA methods it won't be available for your users to use. You can always change this later on from the plugin's settings.

☒ **One-time code via 2FAApp (TOTP)** - [complete list of supported 2FA apps](#).

When using this method, users will need to configure a 2FA app to get the one-time login code. The plugin supports all standard 2FA apps. Refer to the [guide on how to set up 2FA apps](#) for more information. Allowing users to set up a secondary 2FA method is highly recommended. You can do this in the next step of the wizard. This will allow users to log in using an alternative method should they, for example lose access to their phone.

☐ **One-time code via email (HOTP)** - ensure email deliverability with the free plugin [WP Mail SMTP](#).

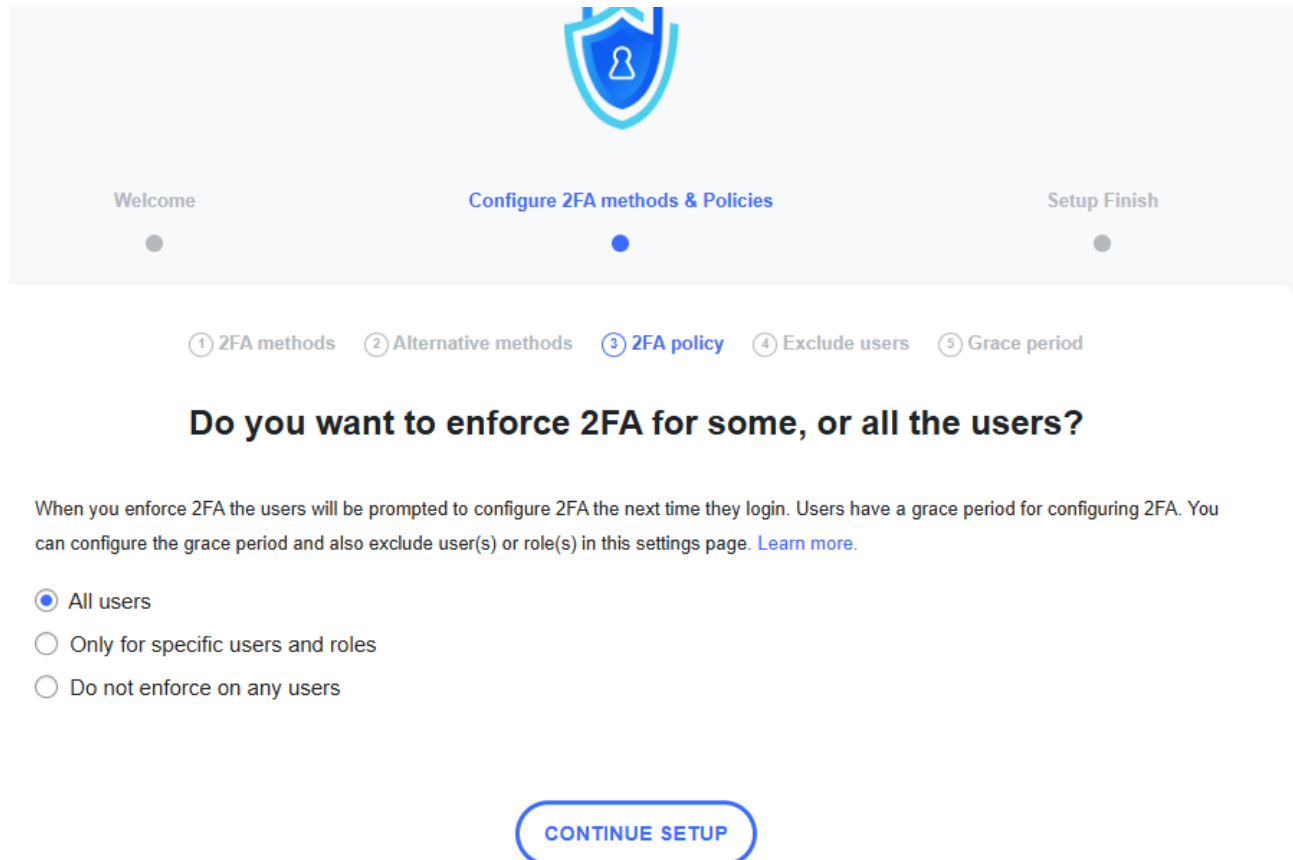
When using this method, users will receive the one-time login code over email. Therefore, email deliverability is very important. Users using this method should whitelist the address from which the codes are sent. By default, this is the email address configured in your WordPress. You can run an email test from the plugin's settings to confirm email deliverability. If you have had email deliverability / reliability issues, we highly recommend you to install the free plugin [WP Mail SMTP](#).

Allowing users to set up a secondary 2FA method is highly recommended. You can do this in the next step of the wizard. This will allow users to log in using an alternative method should they, for example lose access to their phone.

CONTINUE SETUP

Figure 5 Configure methods and Policies

We then enabled the option that **requires all users** to use **multi-factor authentication**.(Figure 6)



1 2FA methods 2 Alternative methods 3 2FA policy 4 Exclude users 5 Grace period

Do you want to enforce 2FA for some, or all the users?

When you enforce 2FA the users will be prompted to configure 2FA the next time they login. Users have a grace period for configuring 2FA. You can configure the grace period and also exclude user(s) or role(s) in this settings page. [Learn more](#).

☒ All users

☐ Only for specific users and roles

☐ Do not enforce on any users

CONTINUE SETUP

Figure 6 All user enforcement

Next, we selected the option that **requires users to enable multi-factor authentication immediately, with no grace period.** (Figure 7)

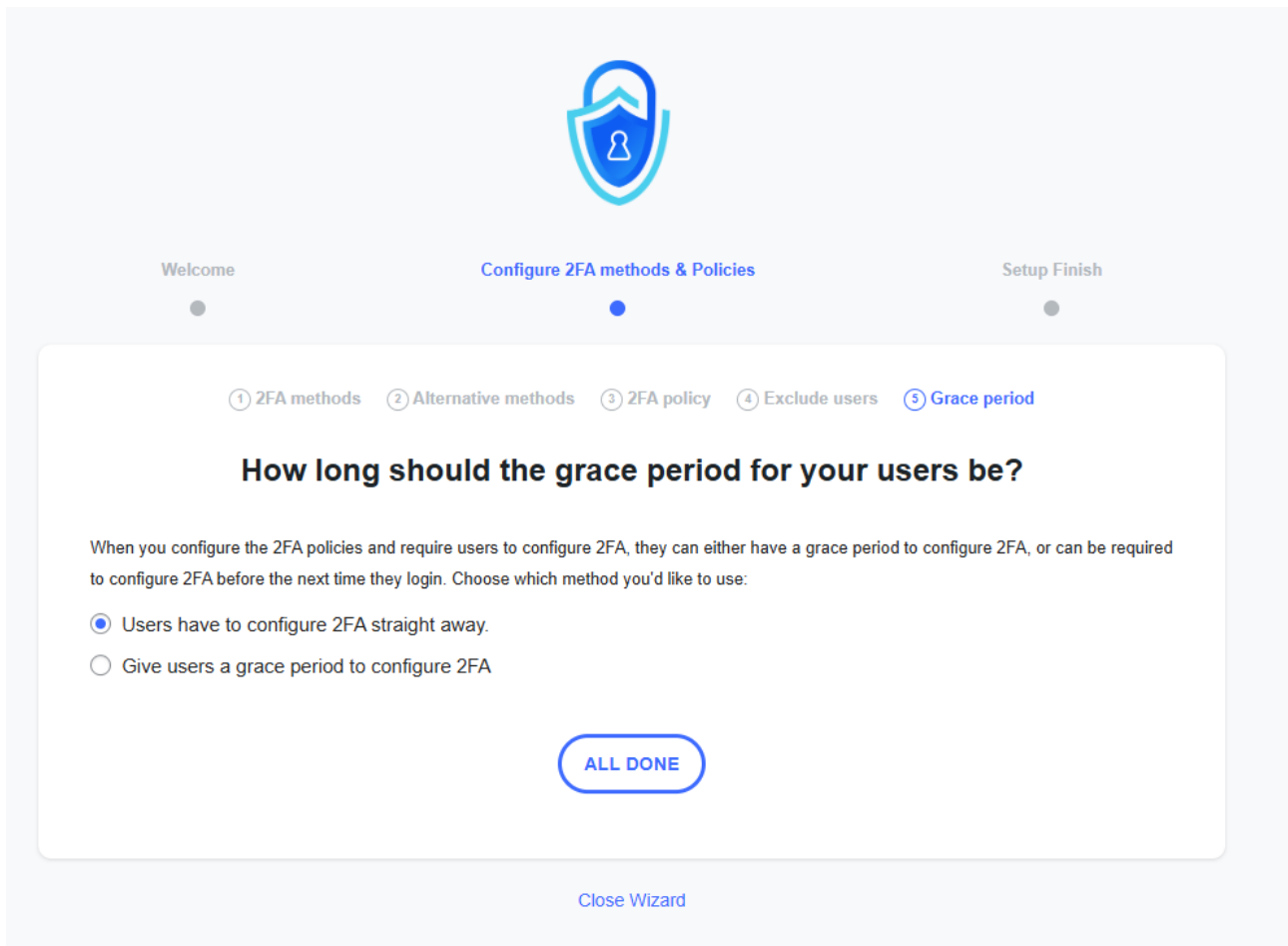


Figure 7 is a screenshot of a configuration wizard for 2FA. At the top, there is a blue shield icon with a white person silhouette. Below it, a progress bar shows three steps: 'Welcome', 'Configure 2FA methods & Policies' (active), and 'Setup Finish'. The main content area has a header with five numbered steps: '1 2FA methods', '2 Alternative methods', '3 2FA policy', '4 Exclude users', and '5 Grace period'. The title of the current step is 'How long should the grace period for your users be?'. The text below the title reads: 'When you configure the 2FA policies and require users to configure 2FA, they can either have a grace period to configure 2FA, or can be required to configure 2FA before the next time they login. Choose which method you'd like to use:'. There are two radio button options: 'Users have to configure 2FA straight away.' (selected) and 'Give users a grace period to configure 2FA'. At the bottom of the main content area is a blue 'ALL DONE' button. Below the main content area is a 'Close Wizard' link.

Figure 7 Grace Period Choice

We clicked **“All done”** and then selected **“Configure 2FA now.”**

As the authentication method, we chose the **Google Authenticator app.**

We scanned the **QR code** (shown in **Figure 8**), and received a **6-digit code** in the authenticator app on our phone.

Additionally, we created a **dedicated email address for our group** to use for authentication purposes.

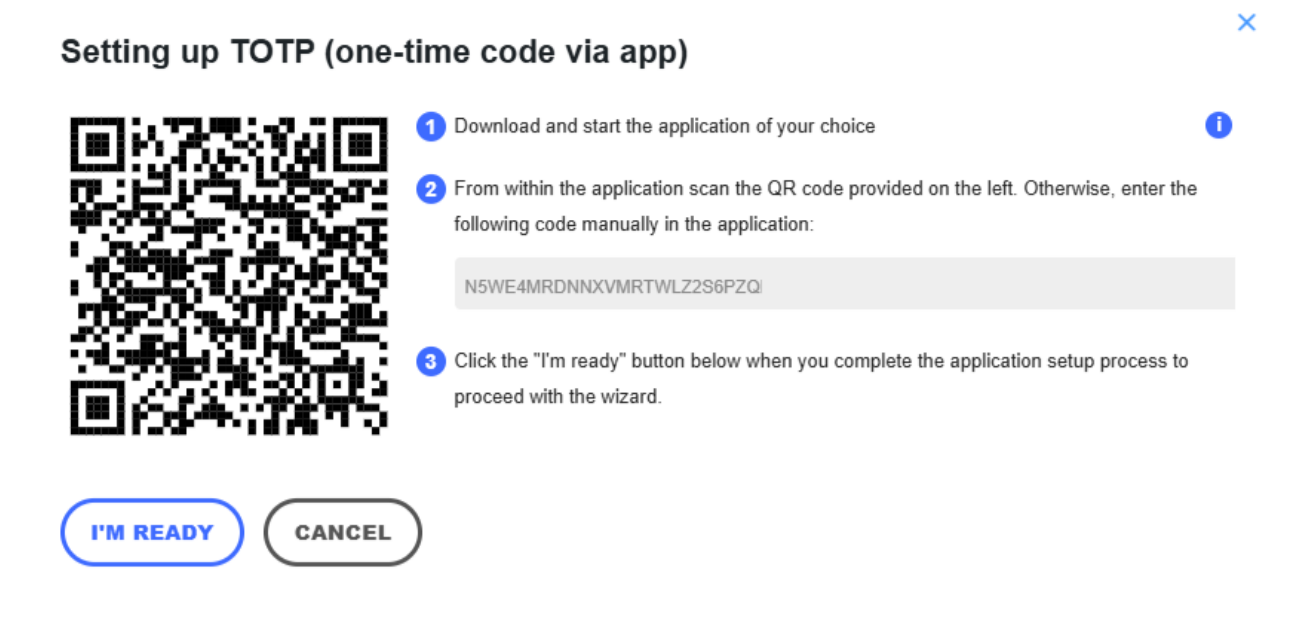


Figure 8 QR code for implementing the application.

Then we need to add the code from the google authenticator app to move to the next step. After entering the code from the authenticator app, we were given the option to **generate backup codes**. We chose to **generate them later**. (Figure 9)

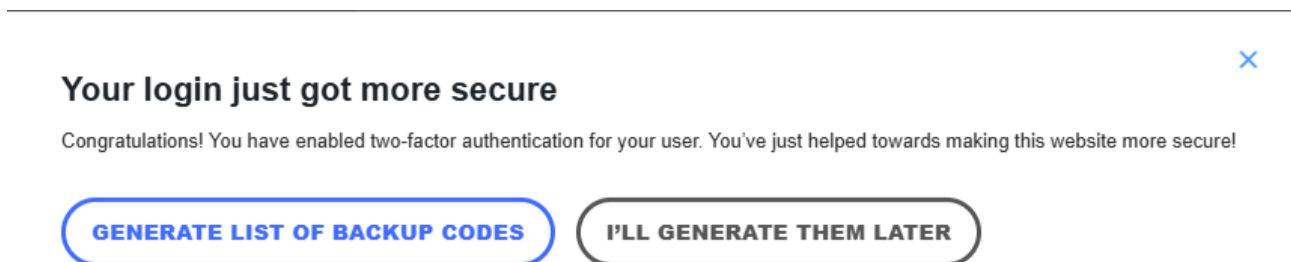


Figure 9 Backup code generate

Next, we tested the login using **two-factor authentication**. When logging into the **WordPress admin dashboard**, we were prompted to enter an **authentication code**. (Figure 10)

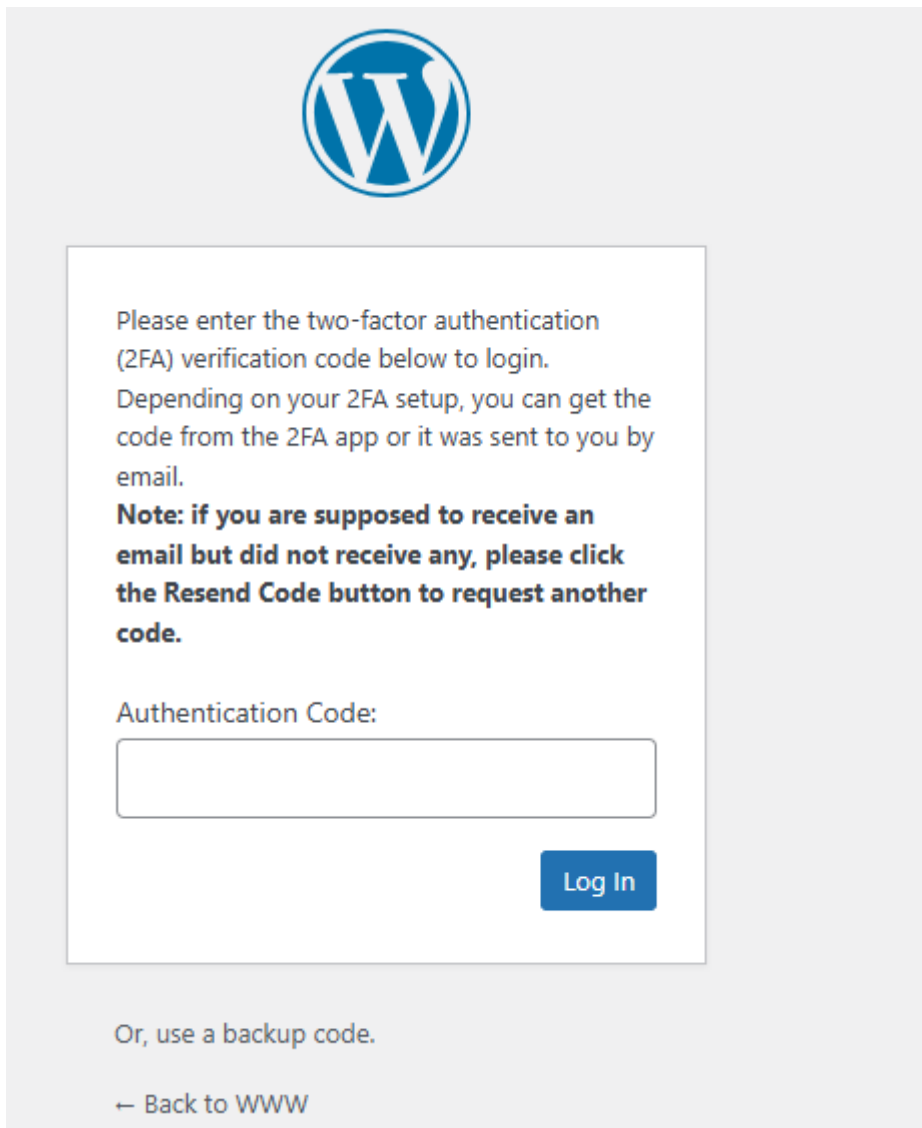
The image shows the WordPress Two-Factor Authentication (2FA) login screen. At the top center is the WordPress logo, a blue circle with a white 'W'. Below the logo is a white rectangular box containing the following text: 'Please enter the two-factor authentication (2FA) verification code below to login. Depending on your 2FA setup, you can get the code from the 2FA app or it was sent to you by email. **Note: if you are supposed to receive an email but did not receive any, please click the Resend Code button to request another code.**' Below this text is a label 'Authentication Code:' followed by a white input field with a thin blue border. To the right of the input field is a blue button with the text 'Log In' in white. Below the white box, the text 'Or, use a backup code.' is displayed. At the bottom left, there is a link '← Back to WWW'.

Figure 10 Authentication Code Requirement

From this, we entered the 6-digit code displayed in Google Authenticator and accessed the control panel.

3.2 Adding 2FA to Centos

The next task was to configure **multi-factor authentication for SSH login** to the web server.

We installed **Google Authenticator** on the server using the commands shown in **Figure 11**.

```
sudo dnf install -y epel-release  
  
sudo dnf install -y google-authenticator qrencode qrencode-libs
```

Then run the `google-authenticator` command to create a new secret key in the `~/.ssh/` directory.

```
google-authenticator -s ~/.ssh/google_authenticator
```

Figure 11 Instructions for Installing multi-factor authenticator for ssh login

The **last command in Figure 11** generates a **secret key** for authentication and saves it in the **.ssh** directory under the name **google_authenticator**.

Next, we were asked whether we want the authentication tokens to be **time-based**, and we answered **yes**. (Figure 12)

```
Complete!  
[root@www ~]# google-authenticator -s ~/.ssh/google_authenticator  
Do you want authentication tokens to be time-based (y/n)
```

Figure 12 Time-based tokens

After responding to the prompt, a **QR code** appeared in the terminal. We scanned it using the **Google Authenticator app** on our phone. (Figure 13)



Figure 13 QR code on a web server

We entered the code displayed in the app into the **command line**. We were then given a set of **emergency backup codes**, which we saved in a **secure location**. (Figure 14)

```
Your new secret key is: QWJFIURPCIO7HY26CTZB5GBRHY
Enter code from app (-1 to skip): 882274
Code confirmed
Your emergency scratch codes are:
  51618362
  90595530
  31839444
  30637376
  87210033

Do you want me to update your "/root/.ssh/google_authenticator" file? (y/n) y
```

Figure 14 Emergency codes

Next, we answered “**yes**” to all the prompts shown in **Figure 15**.

These questions were related to **security settings**. For example, the final question asked whether to enable **rate limiting**, which helps protect against **brute-force attacks**. The first one is about the multiple access using the same token to prevent the MITM(Man-in-the-middle) attack.

```
Do you want me to update your "/root/.ssh/google_authenticator" file? (y/n) y

Do you want to disallow multiple uses of the same authentication
token? This restricts you to one login about every 30s, but it increases
your chances to notice or even prevent man-in-the-middle attacks (y/n) y

By default, a new token is generated every 30 seconds by the mobile app.
In order to compensate for possible time-skew between the client and the server,
we allow an extra token before and after the current time. This allows for a
time skew of up to 30 seconds between authentication server and client. If you
experience problems with poor time synchronization, you can increase the window
from its default size of 3 permitted codes (one previous code, the current
code, the next code) to 17 permitted codes (the 8 previous codes, the current
code, and the 8 next codes). This will permit for a time skew of up to 4 minutes
between client and server.
Do you want to do so? (y/n) y

If the computer that you are logging into isn't hardened against brute-force
login attempts, you can enable rate-limiting for the authentication module.
By default, this limits attackers to no more than 3 login attempts every 30s.
Do you want to enable rate-limiting? (y/n) y
```

Figure 15 Security settings

Next, we configured the **SSH daemon to use Google Authenticator** for login.

Our group had previously been using **SSH keys** for user authentication, but during the final stages of the previous exercise, we had to **reset the web server**. So, we started by reconfiguring SSH key access.

As an example, we created an **SSH key for the root user** by running the command:

ssh-keygen. (Figure 16)

```
[root@www ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:6JR1oUUfw4eqkWrBYhnpCn5uN/dj/5qhthztuvdRL0 root@www.group4.ttc60z.vle.fi
The key's randomart image is:
+---[RSA 3072]---+
|      . .+.o.      |
|      o  o 000.    |
|      . + o.....  |
|      = o+o..      |
| .. o .+oSo        |
| ... oo . . . =    |
| o  .. . o E o     |
| o o ..+++.o o     |
| . . o +BB=+o      |
+---[SHA256]-----+
```

Figure 16 ssh-keygen command

We copy the public key to the authorized keys folder using the command `cp id_rsa.pub authorized_keys`. We also copied the key `id_rsa` to WS01 for the Putty login. (Figure 17)

```
File Edit View

-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktdjEAAAAAAAAcmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAAGAAAABCCaHdcaX
fPPRPH/26wmAxxAAAAEAAAAEAAAGXAAAAAB3NzaC1yc2EAAAADAQABAAQgQCuwUirT+zu
fY1GXt9iCtvtRTuECuQQBMTA0sQQySanJ3W19vxPEaixWGBEE1J46RctRKvqKFATBFfYf7
b6cRWlNv8v1GWcWi+y8G3Yq1d3ai/CfY2ynkup001eK224y8uoq7anYT9dZPew1IYVaugh
O/YJZso62J70s7za44R2s/G47aL6XDGY0/13E0PhVYRSdRt+XkIjT67kIvekn3BAGRZnoF
tfJNMw6XFUZZxgczePkj3ZnnNhoEM86s44R5DNkD4u1sKKxLW3hAJNH23dAaxBUP2HH+UD
e2kSS0ITbIx9HT1yK5dW+H8jEJKoexqgX1LITC3bTJ92DmbB6t6cmmGB7Js006CFXAMF/+
BwhwORNndgQ0qxGGpk6xDdB+1K50f1LI0kXpc2ov09M1Pfs4E06t0KCPPtdQwJWxYcj1IT
n9TjVNXkXIAKKSyTX0WDSQak1bTe808uUm5ikxBV122LVkX0/b2xbh0NTyrBzWXJpAvC4G
mjVUBDvOAGfjMAAAWgS2kS9t69tqPb2Bmj97xc5Bz4L0AAk17D5HKK8SSTbPUOdvTXx0A
gX9rkSFOYHASwioN6i10Nm8FNJFgR0mDRwhS08sy743ShFke/hMiZgqE01102mqUMcuvOz
yDU+fsFqoM6IKUpRLznUBjAUiwe+7jG5ovnj4/hwNE8dXyqEfidGir0Y1fv8Vzia9mEhXz
51cADLMFczQGWP+f2G3NKMnWLuDu5spSc9SUBDE0+41Fi70fI1crJXz7Bh6HWM/gXj26f
BxUMwFNfu+C4RF/5tEWCX3Q9Z11yL+IPMnLF2sJ8XpgFTR0p7ZqCCIDDxvqXIhXnU2/2Pg
FKNX38h0bYoG0Ty2JBPNVxVf1Ii2Ho9J1fY9ZC1cplu7wd6uzHW1DbR69Fz2Pk/1v9iCjs
9vvQG96dtWrIcA/ZVmPZGUrQp+/XzPm7bc1ei45AFS3w3GGki08TEa1ZbcpwtcUxhu79Zs
U12g0xUB2fWFQ0GeIsKDVjDDurg6jLIj+TwM323aQCtA1082k8gZMIv9PbiStXHP8mA8MP
fSI6h4WsratYMG1rVbvZG7zaYSDPIz0Mzaa0p1N5Kv+NLPWU3RxxKH9DM1gN4xtjdJd36N1
9/jw/6UqzGv1DbGGFeq+d7LFA5cJNYP8Gsud3porZvAZmzmSIJqnL7qn6LTuiFjB3n50wX
a7azWmwiBT3EBKumAKUxSY8gNOYvMQ7+j530dxfgT3LdZ5D6Dk/LushhtsPmxt7kU14Ybd
pD0zT9wR1S6D6nohdZu9dobDgJhkogL4THRexck3K55KNCvod0rG/RnxgzN9mAVcK8dHp
h23DE+5SK4qSuISSCQYKKiWwSdNkgn5f1m66IR1fYJHHAJr92nxvTcii1GsRXLQ+7sNNx
kRVL8sohSBQorR888/KK7Unu+hZKPcPPQrQK2mxfgKGUGB11sA1yndikHQyyTiPkyVhV8S
XhKJ5VxKwOfc3fee9qM9/dPQ0cfc1eEKK1PJc0sx6aFiKscS/CS2snGFAir1KykwJ8Xkb
D2IaJxbggNjQ6jFTd5Q46C7NIcohY9cQLlot5i5c2dDIVN9oZz3V6OuIKP+20rt1IZ1wzD
G0stScYUsRU6/mjT0Mrf13S298Vvg+Tr6S/5DSkQ5TwsZRYfh0ZKr3UCaVkw1ExtY2/pts
pCvXoyZO1X6ud1B1BenAwg91Cq8hVYEndVAEDCO+8A01i1TWqWemCasrkui/2BFvFGANKY
1myGz3mV3yKyzqpjjUPYJRR1Z4brsKYcbxfNBWyc7eYEC3Tvbh6ryZqj7cVTeBh12m8p91
hTWg6PzTT8VQDcrtIg+NY/VoKfCqK1F76IHDnFTZQwsVaJxIW+BSH+p8yC4mT0m4e6ef/
Y/6T1Z+MSusptM5wZ3wnadIS7AhSjQ5yIpoP0kwfsGcXLA60PUrqhaPL6XcbRr+b+Cfo5S
G0ZLnjTHuGyrvKEqDzAKHmaW6u8V1n32Wodczqb0zg4GVWQXmJ6vz0kptqyQsDenLBVhvg
AIMFL/Msyi2oLYVQbeeMeXiWpvdLfdIR/eCI13ZK0gZBrrt0bc79HE/yooI9QbH1JGWOLC
Afy7CAM34f41zI6jaNwMIa6skSMI3qDMbfGSwrhxR4hLOPknYpTqy94kRie1gguaCjW0wk
0kr9rFfwtwDeeZ5HRA2Y2RatjeOkciHiuqOSFh0GZo7uYafZ3EDHdIkS+yWKKs-fPD9tvx0
zeIKyMtEgSWdEnbjhfvjtoY8GD1DuTKFiyelIwgb1mgYGX2tPfQyrFqA+gDfXmNbpaUT8e
RwRt+3UDSogayjZ+/MoCvvJ0db9xbUoG+LY//jqxXor8FrNvPaWg0G06ZK2x1ZpZgY5ifx
yCV7hr3EETpWC8jzirAOrIvq7FxaQUU2EX12TanPwr4SHyXF
-----END OPENSSH PRIVATE KEY-----|
```

Figure 17 id_rsa

We created the SSH key from a **text file using the PuTTYgen application** and saved it by clicking **“Save private key.”** (Figure 18)

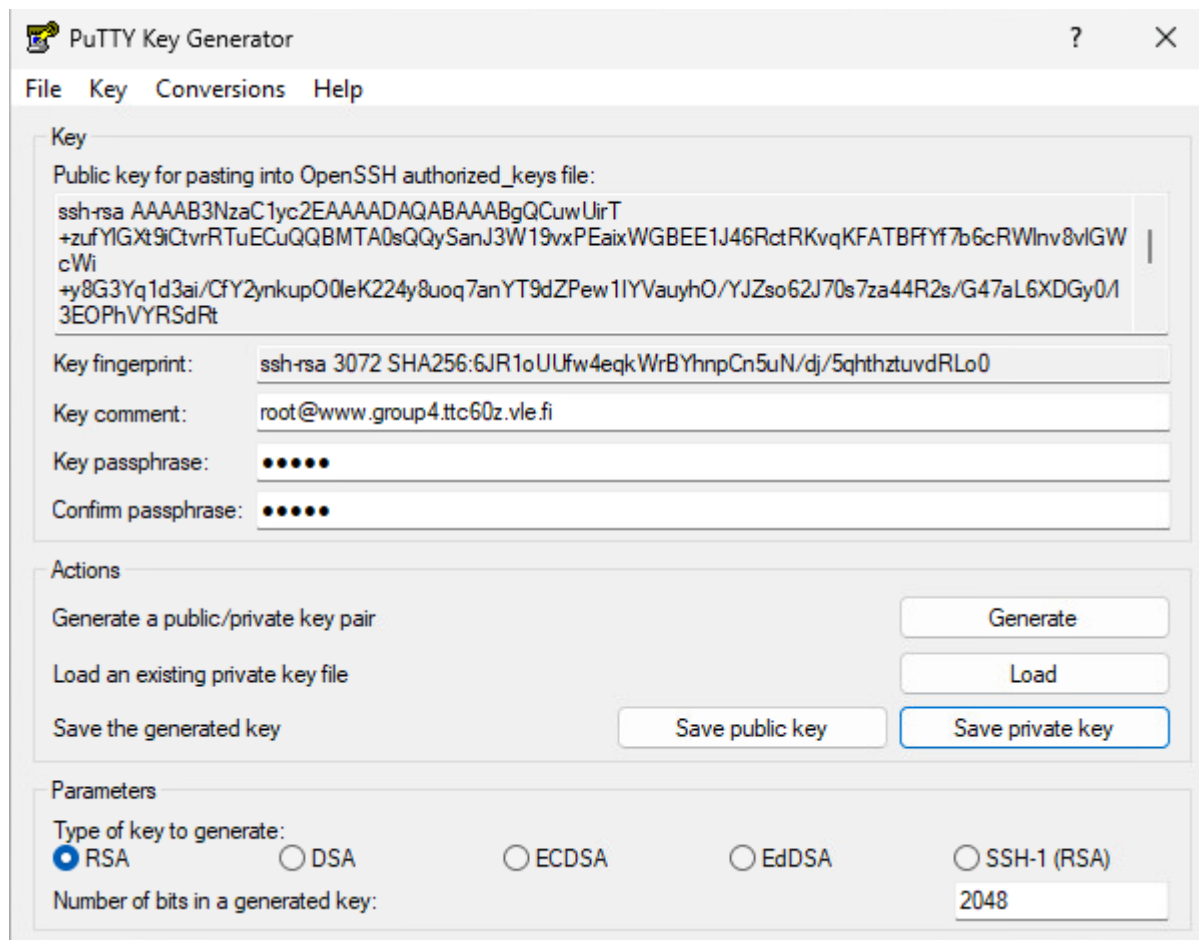


Figure 18 Puttygen

We edited the `/etc/ssh/sshd_config` file to require SSH key authentication when logging in. (Figure

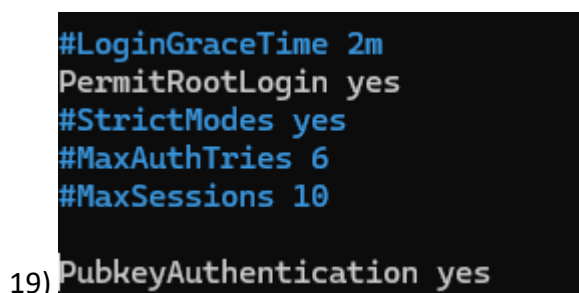


Figure 19 PubkeyAuthentication

We restarted the SSH daemon using the command: **systemctl restart sshd** to apply the new settings.

To enable **multi-factor authentication**, we made the following changes to the `sshd_config` file:

- Set **UsePAM** yes (see **Figure 20**)
- Set **ChallengeResponseAuthentication** yes (see **Figure 21**)
- Added an additional configuration line as shown in **Figure 22**

```
# problems.
UsePAM yes
```

Figure 20 UsePam setting

```
# Change to no to disable s/key passwords
ChallengeResponseAuthentication yes
#ChallengeResponseAuthentication no
```

Figure 21 ChallengeResponseAuthentication

```
AuthenticationMethods publickey,keyboard-interactive
```

Figure 22 AuthenticationMethods

Because it showed error after we restart the sshd, we edited the `/etc/pam.d/sshd` file by adding the lines shown in **Figure 23**.

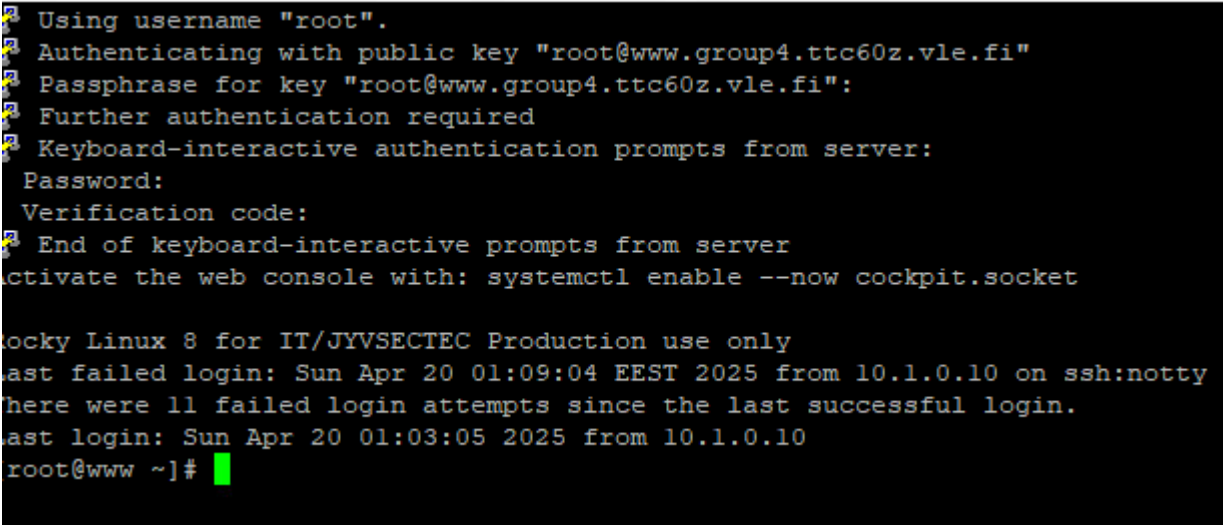
```
##PAM-1.0
auth      substack      password-auth
auth      include        postlogin
#two-factor authentication via Google Authenticator
auth      required      pam_google_authenticator.so secret=${HOME}/.ssh/google_authenticator
account    required      pam_sepermit.so
account    required      pam_nologin.so
account    include       password-auth
password   include       password-auth
# pam_selinux.so close should be the first session rule
session    required      pam_selinux.so close
session    required      pam_loginuid.so
# pam_selinux.so open should only be followed by sessions to be executed in the user context
session    required      pam_selinux.so open env_params
session    required      pam_namespace.so
session    optional      pam_keyinit.so force revoke
session    optional      pam_motd.so
session    include       password-auth
session    include       postlogin
```

Figure 23 Editing the `pam.d/sshd` file

We restarted the SSH daemon again using the command: **systemctl restart sshd** to apply the changes.

When we reconnected to the **web server via SSH using PuTTY**, we were prompted for **two-factor authentication**.

We entered the **verification code from the Google Authenticator app on our phone** (see **Figure 24**).

A terminal window showing the SSH login process. The user 'root' is logging in from '10.1.0.10'. The terminal displays the following text: 'Using username "root".', 'Authenticating with public key "root@www.group4.ttc60z.vle.fi"', 'Passphrase for key "root@www.group4.ttc60z.vle.fi":', 'Further authentication required', 'Keyboard-interactive authentication prompts from server:', 'Password:', 'Verification code:', 'End of keyboard-interactive prompts from server', 'Activate the web console with: systemctl enable --now cockpit.socket', 'rocky Linux 8 for IT/JYVSECTEC Production use only', 'Last failed login: Sun Apr 20 01:09:04 EEST 2025 from 10.1.0.10 on ssh:notty', 'There were 11 failed login attempts since the last successful login.', 'Last login: Sun Apr 20 01:03:05 2025 from 10.1.0.10', and the prompt 'root@www ~]#'.

```
Using username "root".
Authenticating with public key "root@www.group4.ttc60z.vle.fi"
Passphrase for key "root@www.group4.ttc60z.vle.fi":
Further authentication required
Keyboard-interactive authentication prompts from server:
Password:
Verification code:
End of keyboard-interactive prompts from server
Activate the web console with: systemctl enable --now cockpit.socket

rocky Linux 8 for IT/JYVSECTEC Production use only
Last failed login: Sun Apr 20 01:09:04 EEST 2025 from 10.1.0.10 on ssh:notty
There were 11 failed login attempts since the last successful login.
Last login: Sun Apr 20 01:03:05 2025 from 10.1.0.10
root@www ~]#
```

Figure 24 Multi-factor authentication when opening an SSH connection

4 Conclusion

In this lab exercise, we explored the configuration and use of **two-factor authentication (2FA)**. To our surprise, the process was **easier than we expected**, and we completed everything in **under 30 minutes**.

While configuring 2FA for the server, we were pleased to discover that **rate limiting** is also enabled by default. This feature helps protect the server from **DDoS attacks** by limiting the number of incoming requests.

The instructions for setting up multi-factor authentication in **WordPress** were clear, and thanks to the plugin, the installation was **straightforward**.

The setup for **SSH access** was also simple—it required just a few commands and some minor file editing. This made it easy to understand what was happening at each stage of the process.

The **ease of implementation** highlights how important it is to **promote the use of two-factor authentication**, not just in corporate environments but in personal ones as well.

Many organizations still **neglect proper security practices**, and in our view, enabling two-factor authentication should be considered a **basic security requirement**.

4.1 Reflection

This lab gave us practical experience in setting up two-factor authentication (2FA) for both WordPress and SSH access. The process was quicker and easier than expected, taking less than 30 minutes.

We learned how 2FA significantly improves security, and we were pleased to see that features like **rate limiting** were also enabled during setup, offering protection against brute-force and DDoS attacks.

Both the WordPress plugin and SSH configuration were straightforward, helping us understand the underlying mechanics of multi-factor authentication.

Overall, the lab showed how simple it is to implement 2FA and highlighted its importance in both personal and professional environments. It should be considered a basic security practice everywhere.

References

Stephen, J. B., 2025. What is multifactor authentication?. Accessed on April 19, 2025. [What is Multifactor Authentication \(MFA\)? | Definition from TechTarget](#)

Eliza, T., 2025. Multi-Factor Authentication (MFA): Importance and Benefits. Accessed on April 19, 2025. [What is Multi-Factor Authentication, and How Does it Work?](#)

Udoh, B., 2023. How does Google Authenticator work?. Accessed on April 19, 2025. [How does Google Authenticator work? | by Benjamin Udoh | Nerd For Tech | Medium](#)