Template walkthrough

Index

Index 1
List of pictures 1
Disclaimer 2
Reconnaissance 2
Initial foothold 2
User flag 3
Privilege escalation 8
Personal comments 12
References 13

List of pictures

Figure 1 - nMap scan results 2
Figure 2 - SMB shares found 2
Figure 3 - ffuf scan results 3
Figure 4 - SQL Injection found via SQLMap 3
Figure 5 - DB user privileges 4
Figure 6 - Reading of 000-default.conf file 4
Figure 7 - writer.esgi file 5
Figure 8 - __init__.py file 5
Figure 9 - Malicious fake image file 6
Figure 10 - Reverse shell invoking 6
Figure 11 - Database credentials found 7
Figure 12 - Kyle credentials 7
Figure 13 - Kyle credentials cracked 7
Figure 14 - User flag 8
Figure 15 - LinPEAS output 8
Figure 16 - Postfix version 9
Figure 17 - Disclaimer file permissions 9
Figure 18 - User info 9
Figure 19 - First shell as john user 10
Figure 20 - John SSH key 10
Figure 21 - Login as john via SSH 11
Figure 22 - Folder permission 11
Figure 23 - APT task files 11
Figure 24 - Cronjob scheduled 12
Figure 25 - root shell and root flag 12

Disclaimer

I do this box to learn things and challenge myself. I'm not a kind of penetration tester guru who always knows where to look for the right answer. Use it as a guide or support. Remember that it is always better to try it by yourself. All data and information provided on my walkthrough are for informational and educational purpose only. The tutorial and demo provided here is only for those who are willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

Just to say: I am not an English native person, so sorry if I did some grammatical and syntax mistakes.

Reconnaissance

The results of an initial nMap scan are the following:

```
┌──(k14d1u5㉿kali)-[~/…/Linux/Medium/Writer/nMap]
└─$ nmap -sT -sV -p- -A 10.10.11.101 -oA Writer
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-07-02 03:02 PDT
Nmap scan report for 10.10.11.101
Host is up (0.042s latency).
Not shown: 65531 closed tcp ports (conn-refused)
PORT    STATE SERVICE     VERSION
22/tcp  open  ssh         OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 98:20:b9:d0:52:1f:4e:10:3a:4a:93:7e:50:bc:b8:7d (RSA)
|   256 10:04:79:7a:29:74:db:28:f9:ff:af:68:df:f1:3f:34 (ECDSA)
|_  256 77:c4:86:9a:9f:33:4f:da:71:20:2c:e1:51:10:7e:8d (ED25519)
80/tcp  open  http        Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Story Bank | Writer.HTB
139/tcp open  netbios-ssn Samba smbd 4.6.2
445/tcp open  netbios-ssn Samba smbd 4.6.2
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_nbstat: NetBIOS name: WRITER, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
| smb2-time:
|   date: 2025-07-02T10:02:36
|_  start_date: N/A
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled but not required

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 29.02 seconds
```

Figure 1 - nMap scan results

Open ports are 22, 80, 139 and 445. Therefore, enabled services are SSH (22) and SMB (139, 445). Also, a web application is running on port 80. Lastly, nMap tool recognized Linux as operative system.

Initial foothold

First of all, I investigated SMB service and I looked for some interesting shares:

```
┌──(k14d1u5㉿kali)-[~/Desktop]
└─$ smbclient -L //10.10.11.101 -N

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        writer2_project Disk
        IPC$            IPC       IPC Service (writer server (Samba, Ubuntu))
Reconnecting with SMB1 for workgroup listing.
smbXcli_negprot_smb1_done: No compatible protocol selected by server.
Protocol negotiation to server 10.10.11.101 (for a protocol between LANMAN1 and NT1) failed: NT_STATUS_INVALID_NETWORK_RESPONSE
Unable to connect with SMB1 -- no workgroup available
```

Figure 2 - SMB shares found

However, I was not able to access to them because I hadn't credentials. Therefore, I investigated the web application running on port 80. Luckily, I found two interesting new paths:

Figure 3 - ffuf scan results

User flag

Since I found a login form on the new path, I tried to exploit it via SQL Injection. It was possible and I used SQLMap to accomplish this goal:

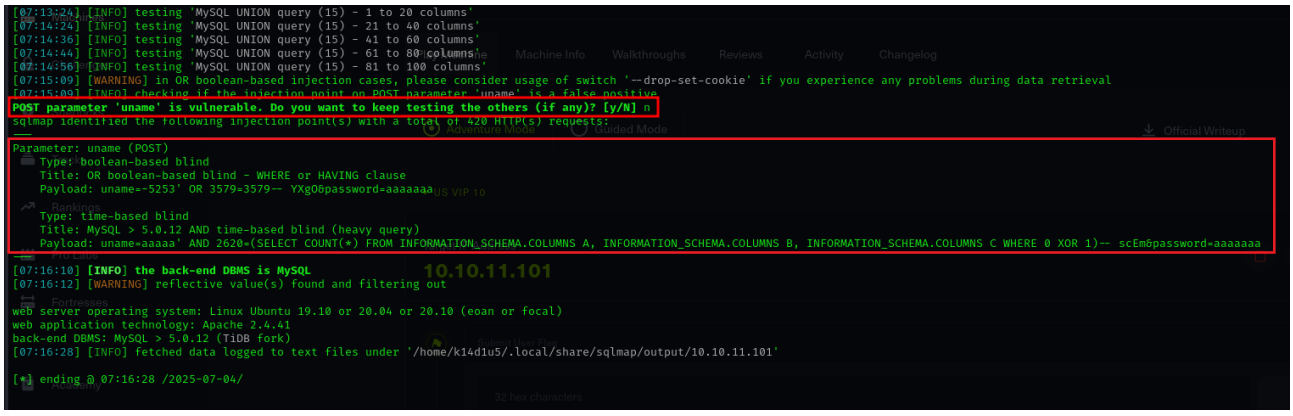

Figure 4 - SQL Injection found via SQLMap

Investigating deeper the SQLInjection, I found out that the user has interesting privileges:

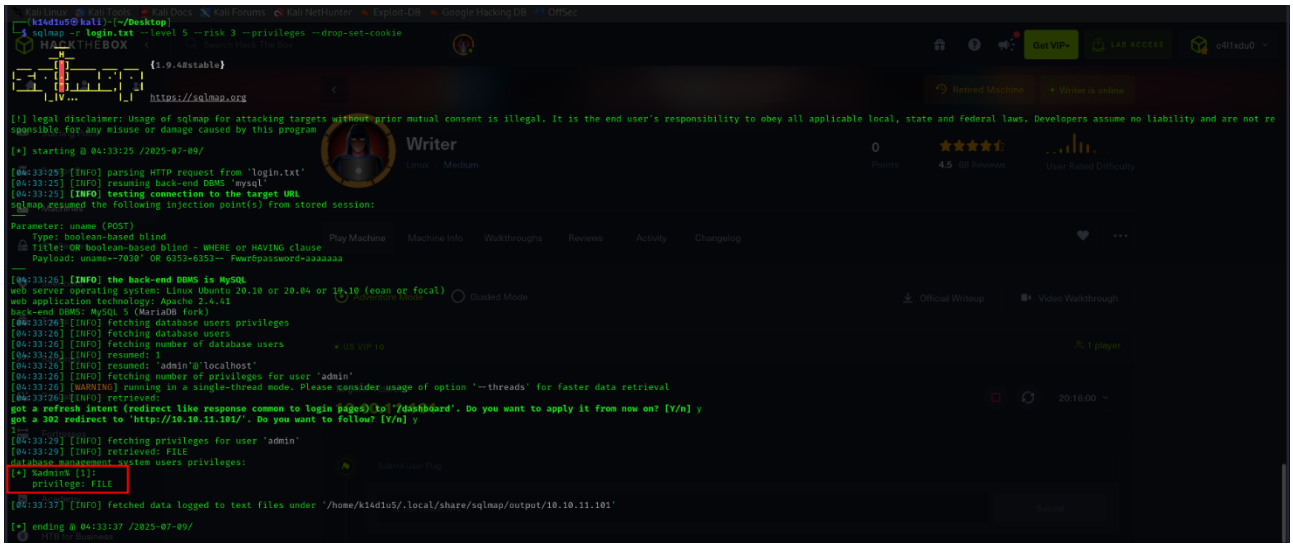

Figure 5 - DB user privileges

Due to this privilege, I was able to read some interesting files. One of these, allowed me to find a new virtual host, as shown I the following picture:

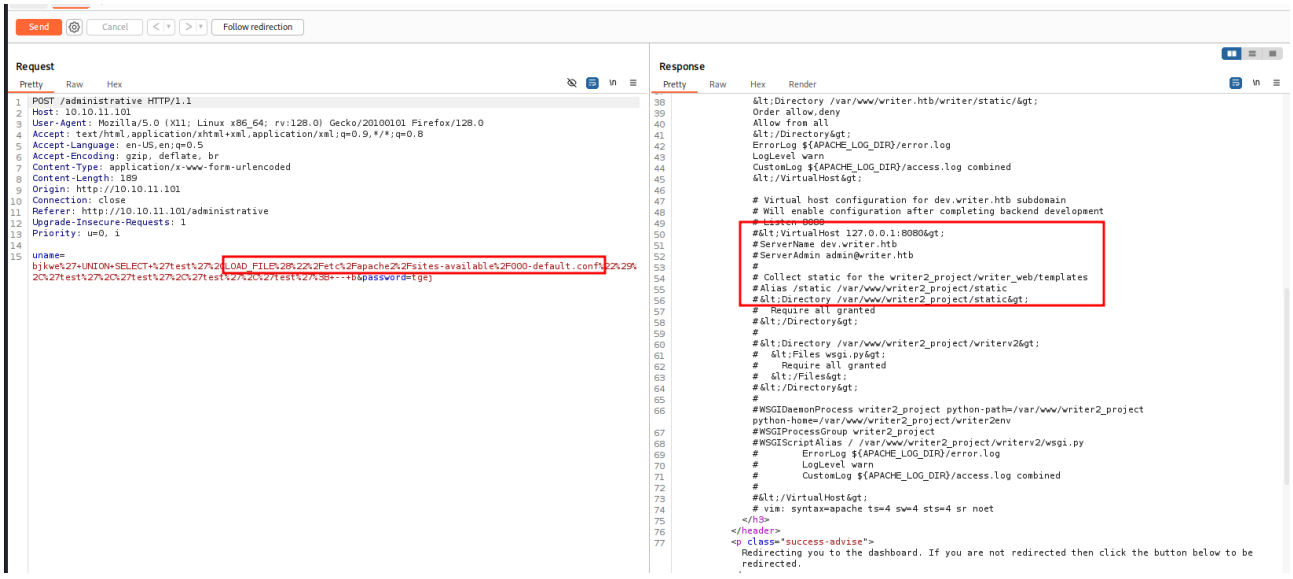

Figure 6 - Reading of 000-default.conf file

Also, I found a file named $writer.wsgi$:

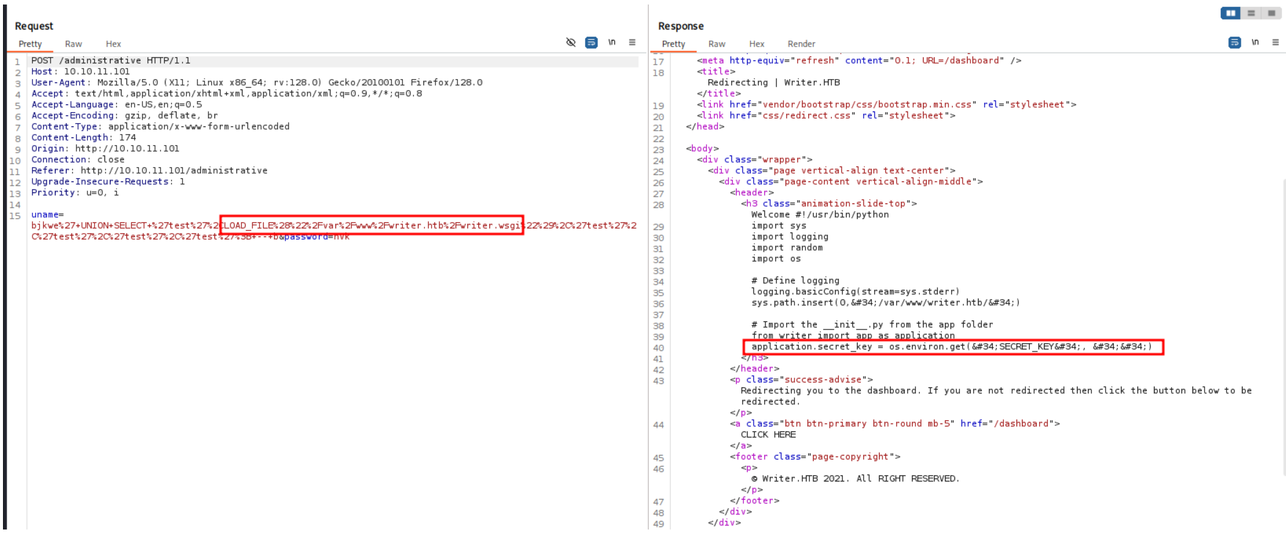

Figure 7 - writer.esgi file

The interesting thing about this file is that it is write in Python. Usually, a python project has a *__init__.py* file. Therefore, I tried to read this file in the same way I just did. Luckily, this file existed and I downloaded it. Analyzing this file, I found the implementation of web application file upload functionality:

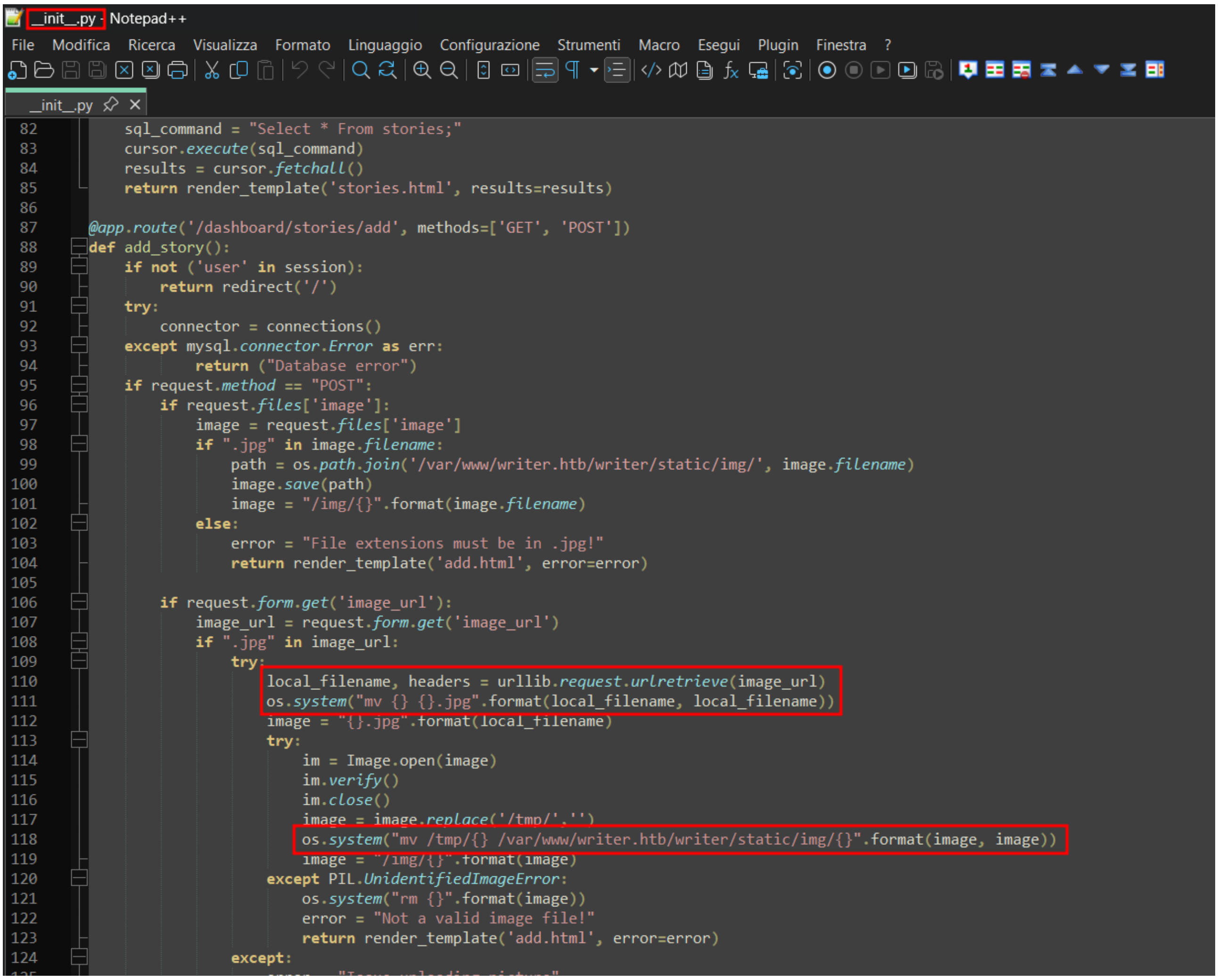

Figure 8 - __init__.py file

In particular, I learnt that the uploaded image was moved in the $/static/img$ path. Also, it used the $urlretrive$ function that allow the user to provide a local file path, using the $file://$ schema. Therefore, I tried to login in the web application. Using one of the SQLInjection payload used by SQLMap, I was able to access to the web application administrative panel. At this point, I tried to exploit the file upload functionality and I exploited the SSRF vulnerability. To do it, I created a fake image file which name has a reverse shell command encoded in base64, as shown in the following picture:

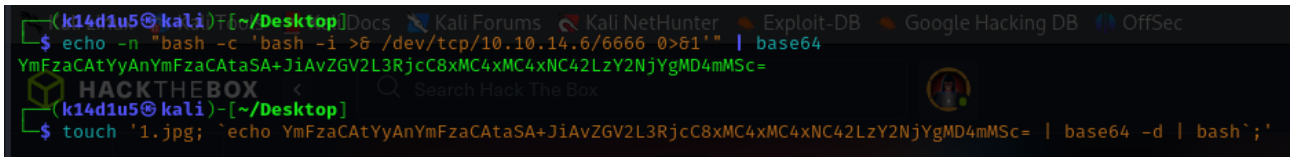

Figure 9 - Malicious fake image file

After I uploaded it, I was able to invoke the payload uploading it again. This time, I uploaded the file from the file system, as shown in the following picture:

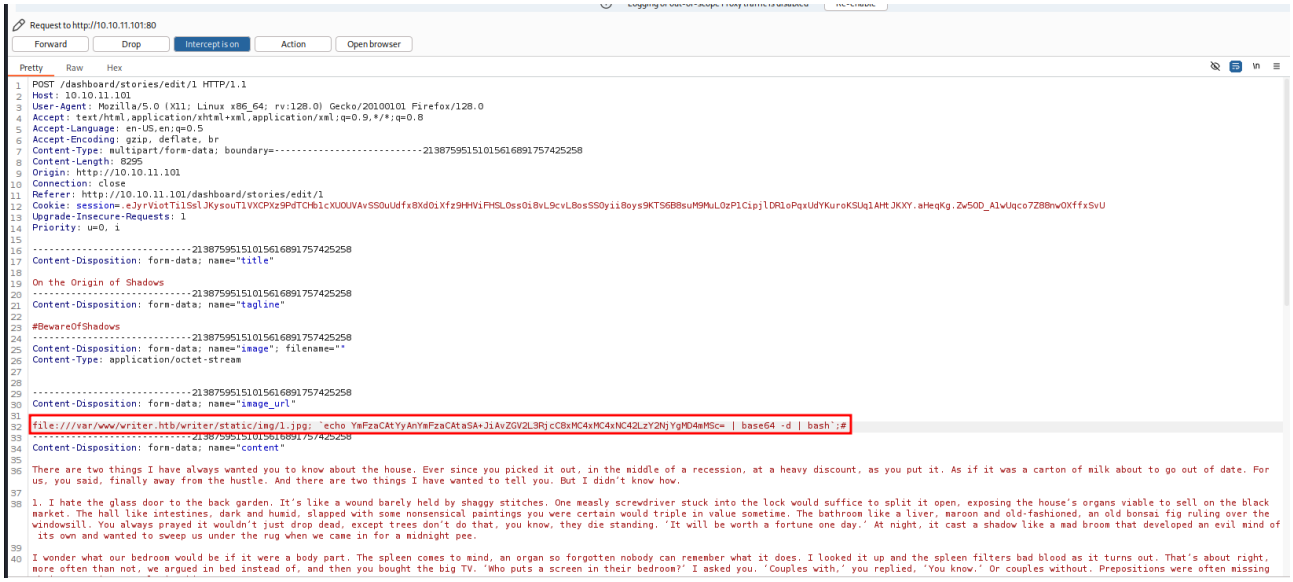

Figure 10 - Reverse shell invoking

In this way, I was able to obtain a shell as $www-data$ user. Therefore, I started to look for some interesting information. Luckily, I found out some database credentials:

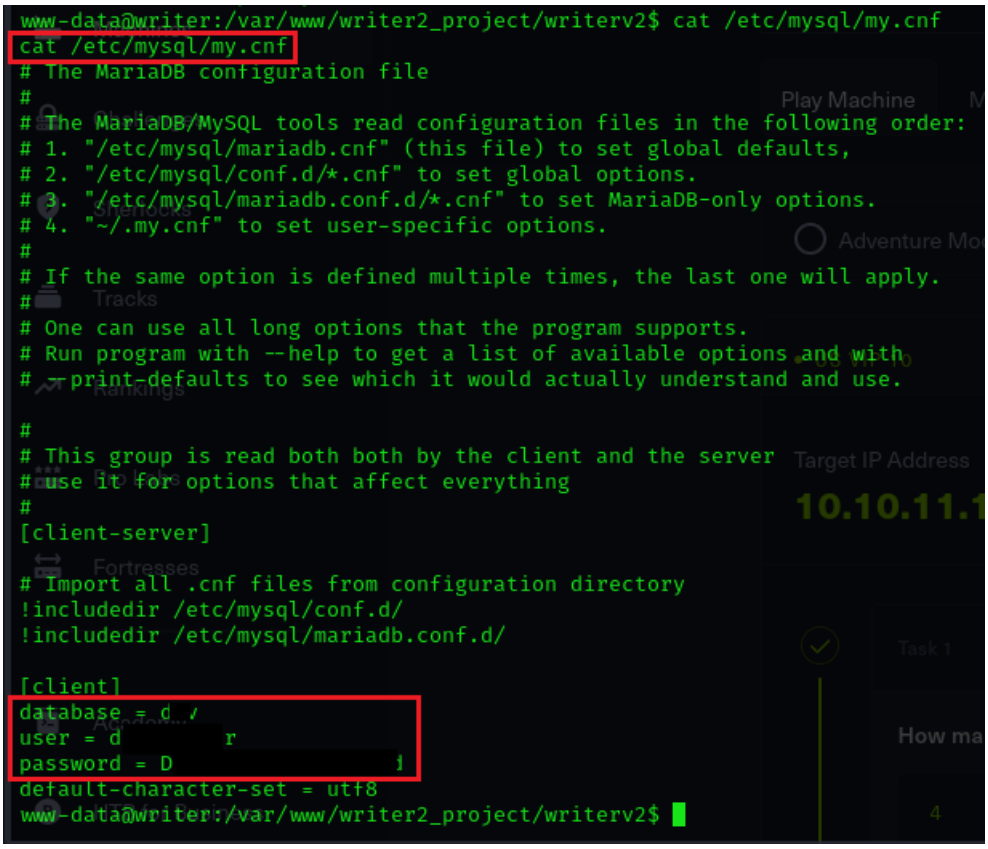

Figure 11 - Database credentials found

Next, I just connected to the database and I explored it. I found new credentials regarding the *kyle* user:

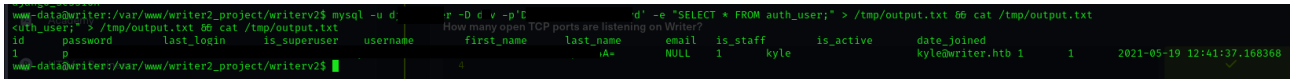

Figure 12 - Kyle credentials

Obviously, credentials are hashed. Next step was cracking it and, luckily, I was successful:

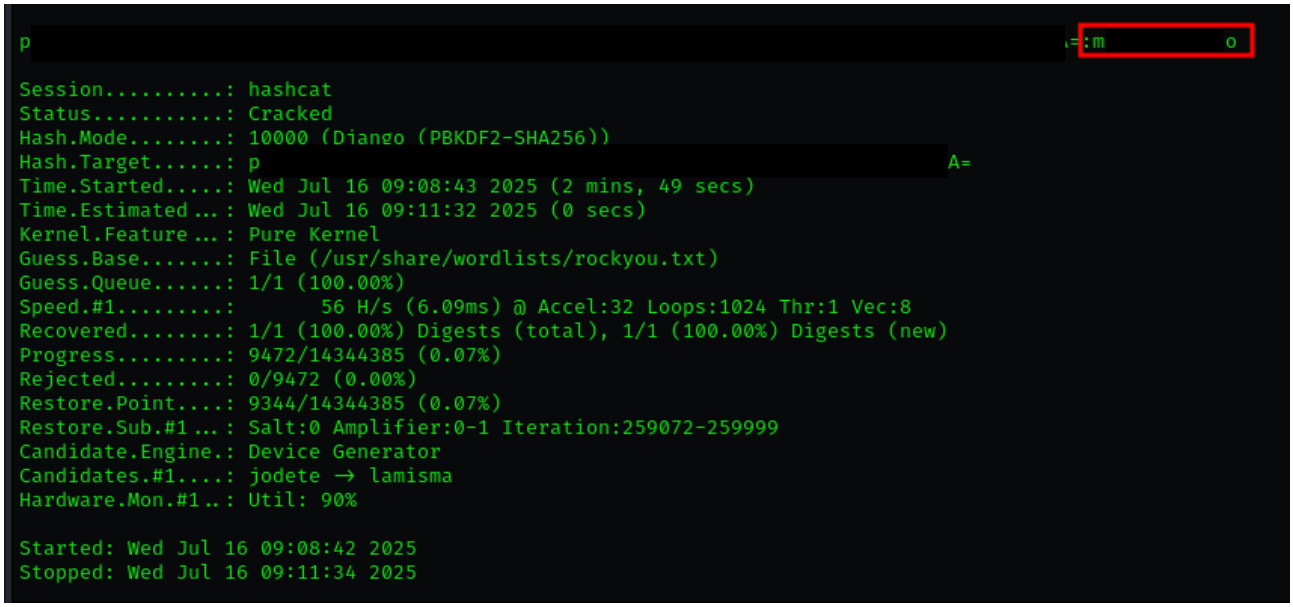

Figure 13 - Kyle credentials cracked

Finally, I was able to connect to the target via SSH as *kyle* user and I retrieved the user flag:

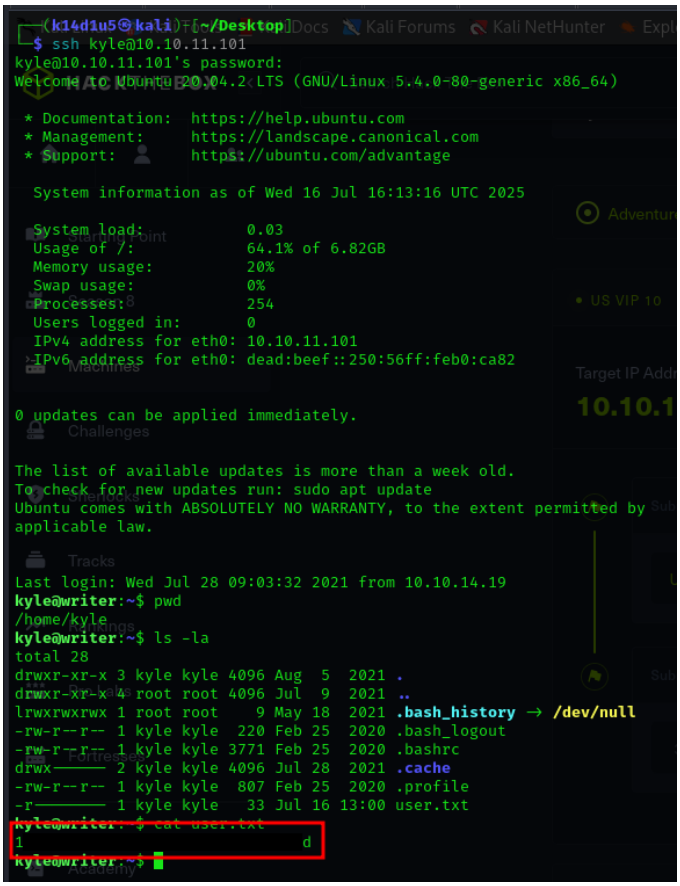

Figure 14 - User flag

Privilege escalation

After all the effort I put until now, I started to find a way to escalate my privileges. First of all, I run LinPEAS tool. From its output, I found a Postfix interesting file used by *john* user:

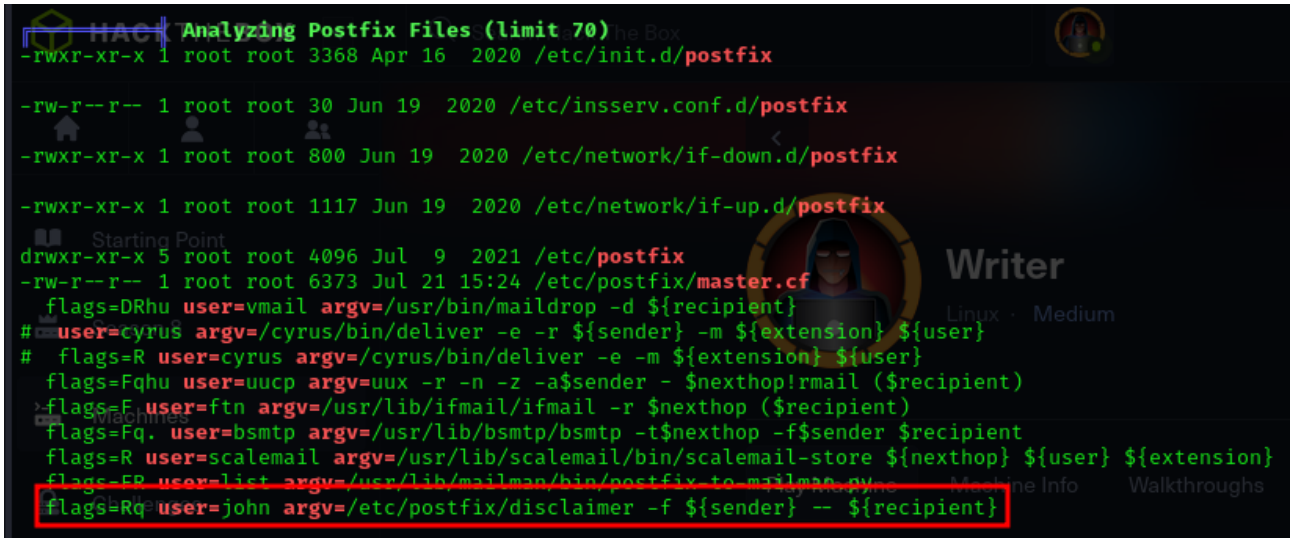

Figure 15 - LinPEAS output

At this point, I verified which Postfix version was installed:

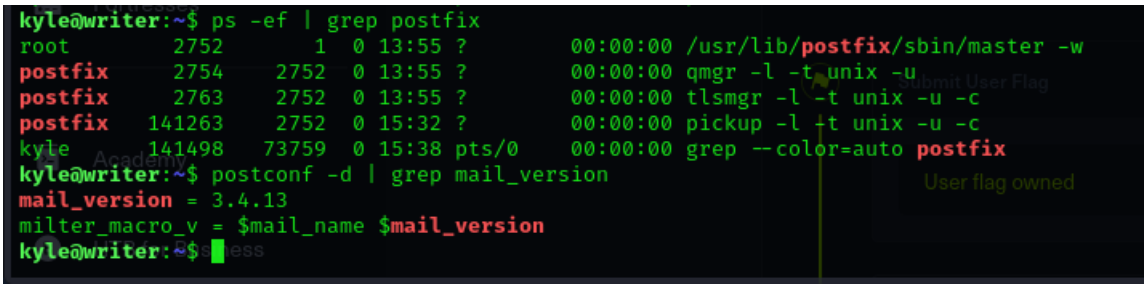

Figure 16 - Postfix version

Also, I investigated the disclaimer file. In particular, I noted that it was writable by $filter$ group:

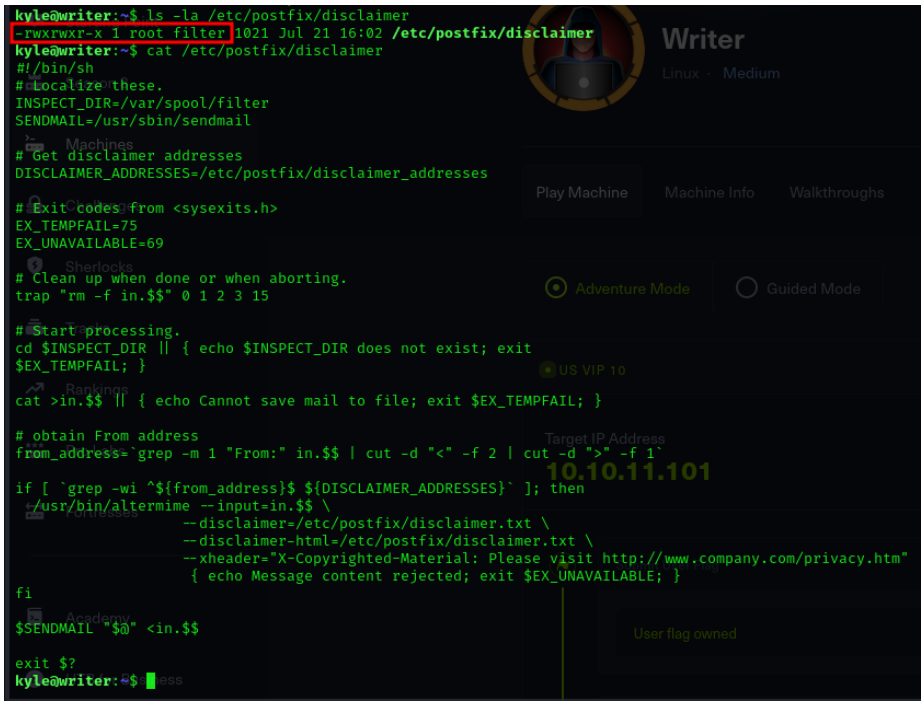

Figure 17 - Disclaimer file permissions

Therefore, I checked my user information. Luckily, my user was in the $filter$ group:

Figure 18 - User info

At this point, I looked for an interesting exploit on the Internet and I performed it. In this way, I was able to obtain a first shell as $john$ user, as shown in the following picture:

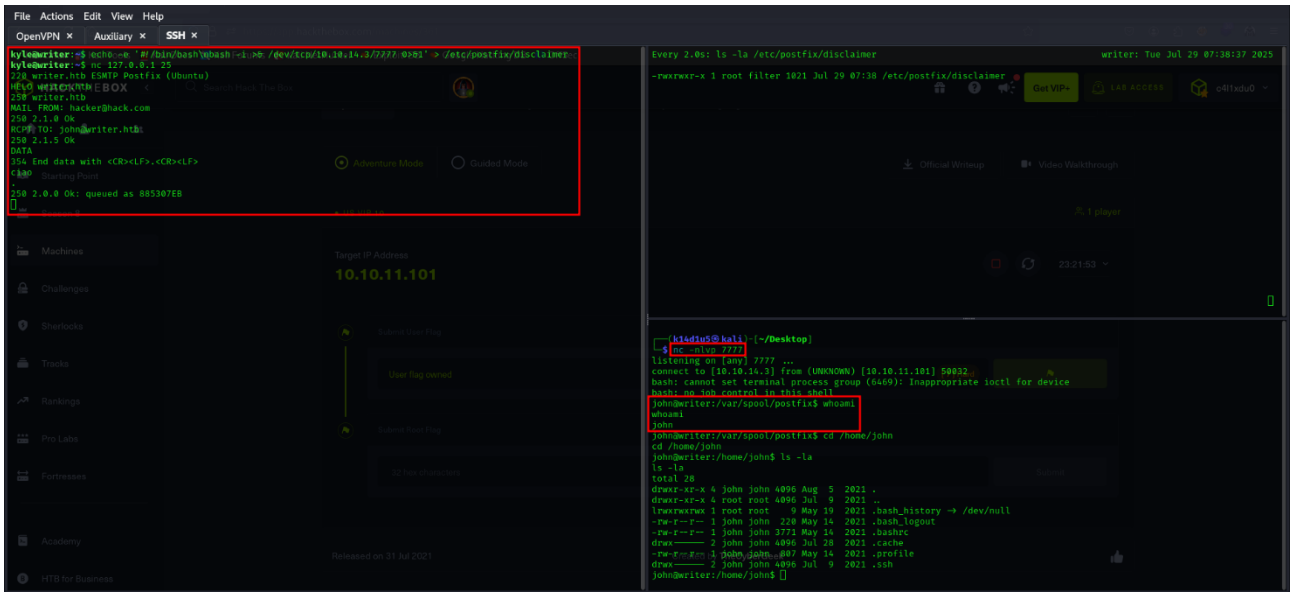

Figure 19 - First shell as john user

Sadly, this shell has not the environment variable set. This means that I was a little limited. Anyway, I looked for some interesting information and I found the *john* ssh key:

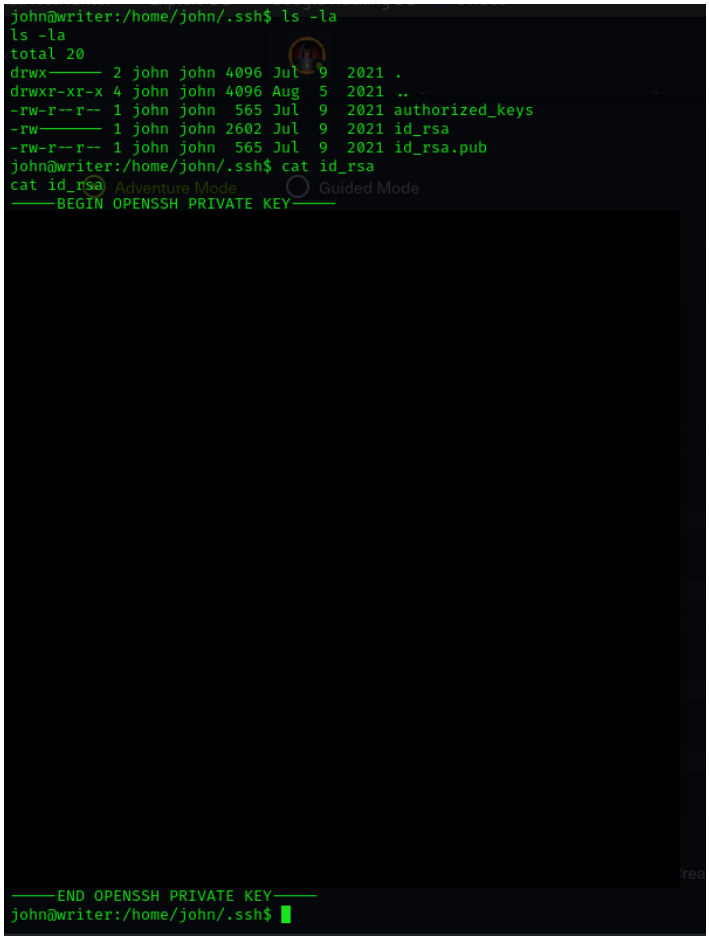

Figure 20 - John SSH key

I just used it to log in on the target as john and have a “complete” shell. Also, I found out that *john* user is in the *maagement* group:

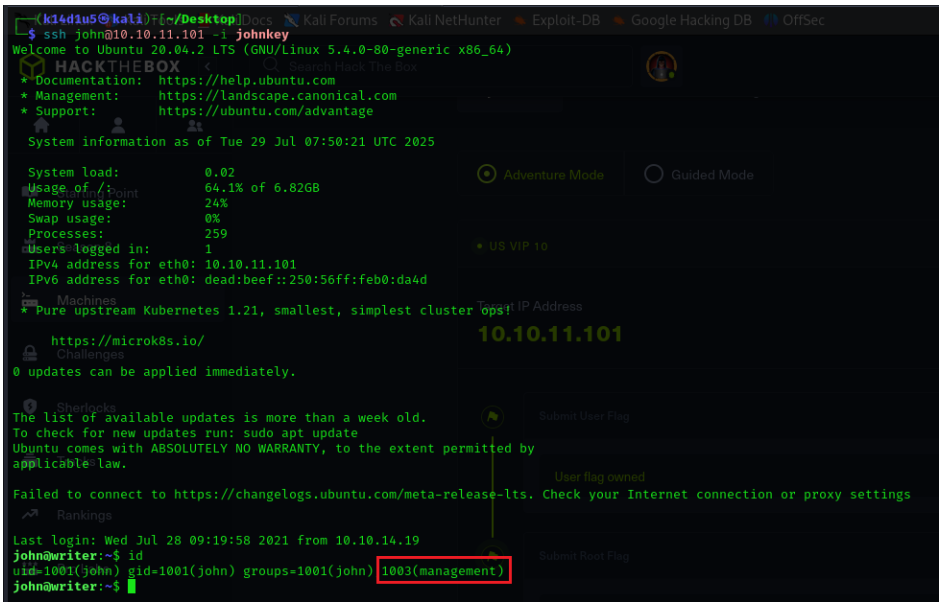

Figure 21 - Login as john via SSH

Therefore, I looked for some usable file from that group. I found out some apt tool relative file. These files are located in a folder in which the *management* group can do anything:

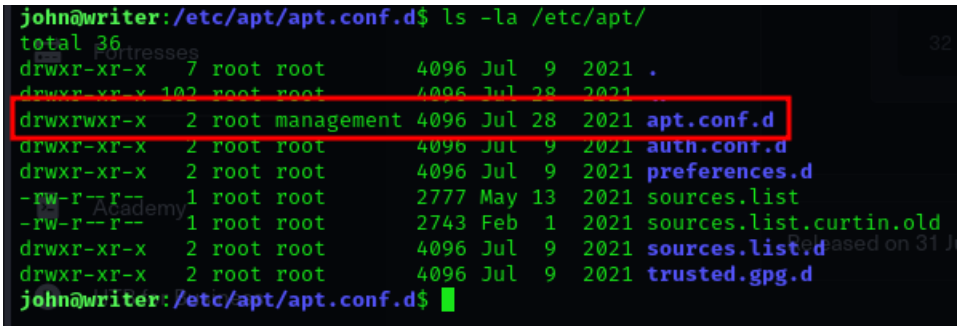

Figure 22 - Folder permission

However, this group can't modify the specific file:

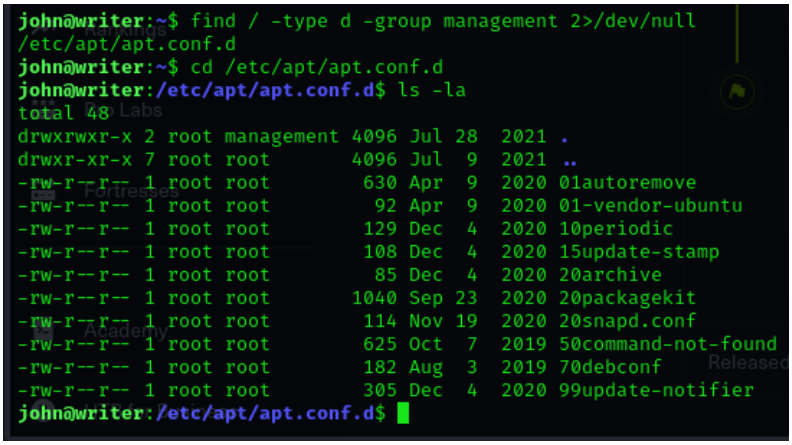

Figure 23 - APT task files

Also, I found out that all files in this folder run as root and are scheduled by cronjob. I found these information running the *pspy* tool:

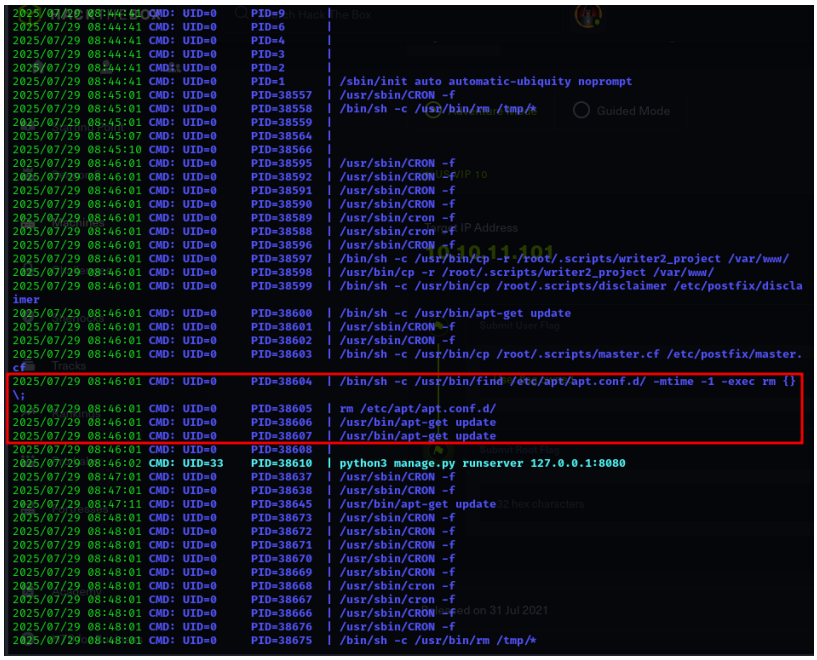

Figure 24 - Cronjob scheduled

I identify them as a cronjob because these commands were executed periodically. Analyzing some of those files, I tried to create a new one to obtain a reverse shell. Also, looking possible exploit on the Internet, I learnt that these files were executed in alphabetic order. Lastly, I created my file to be executed as first one. In this way, I obtained the root shell and I retrieved the root flag:

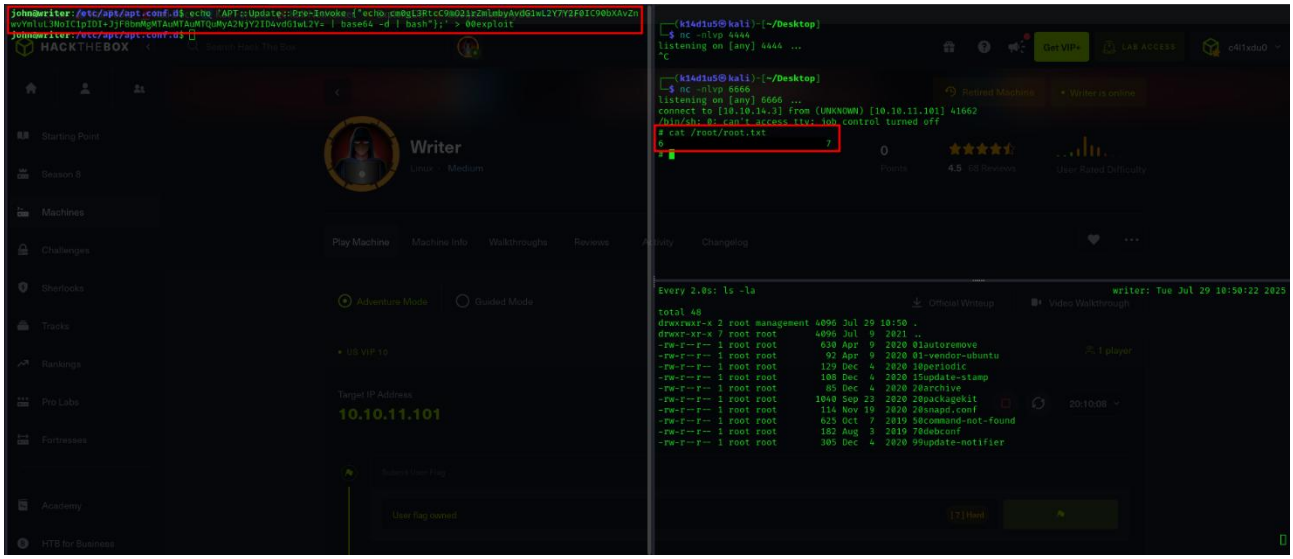

Figure 25 - root shell and root flag

Personal comments

This box was literally crazy. There were several points that make this box both interesting and challenging. I needed a very uncommon flag ($--drop-set-cookie$) to let SQLMap work. At least, it was uncommon to me. Another uncommon task was reading file using SQLInjection. This task is very specific and it was pretty new to me. Also, it was needed to check database user permission to verify it was able to read files, another check I never seen before. Keep going on, I needed to hypothesize the existence of $\_\_init\_\_.py$ file just because I found a python file. In my opinion, this is a forced thought. Lastly, the first shell obtained as $john$ hadn't the environment variable set. It was a new situation I never found before and I found out that it

is limiting. Therefore, I needed to understand that with a different and “complete” shell, I was able to retrieve more details. In conclusion, this box is very interesting and very instructive, but it can’t be evaluated as Medium in my opinion. I evaluate it Hard or at least very close to this difficulty.

References

1. Privilege escalation via APT tool: https://www.hackingarticles.in/linux-for-pentester-apt-privilege-escalation/.