Mirai

Written By https://github.com/dungwoong

Mirai is a machine on hackthebox. It is not too difficult to get into the machine, but it is somewhat difficult to get the root flag. No privilege escalation though!
Overall, it's a relatively easy machine.

Some Hints

- D e f a u l t C r e d e n t i a l s...where could you try those?
- The linux command 'df' tells you about free disk space and disks that you have mounted

Fun Facts

Mirai was the name of a botnet that targeted Linux machines. It would look for certain open ports, and try to brute-force the login credentials. Mirai is now one of the biggest botnets in the world.

The creator of the botnet, a user known as Anna-Senpai, released the source code on Hack Forums. The botnet was initially used to DDoS minecraft servers

Mirai means Future in Japanese.

It's also a character in an anime. Anime is cool.

Sorry, I was sidetracked. Let's get into it!

Reconnaissance

We do a standard port scan with Nmap. We scan every TCP port on the system, looking for information on the technologies that are running on the target machine, and the versions of the technologies.

```
┌──(kali㉿kali)-[~]
└─$ nmap -T4 -p-  -A -Pn  10.10.10.48
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan
times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-09 23:40 EST
Nmap scan report for 10.10.10.48
Host is up (0.10s latency).
Not shown: 65529 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 6.7p1 Debian 5+deb8u3 (protocol 2.0)
| ssh-hostkey:
|   1024 aa:ef:5c:e0:8e:86:97:82:47:ff:4a:e5:40:18:90:c5 (DSA)
|   2048 e8:c1:9d:c5:43:ab:fe:61:23:3b:d7:e4:af:9b:74:18 (RSA)
|   256 b6:a0:78:38:d0:c8:10:94:8b:44:b2:ea:a0:17:42:2b (ECDSA)
|_  256 4d:68:40:f7:20:c4:e5:52:80:7a:44:38:b8:a2:a7:52 (ED25519)
53/tcp    open  domain  dnsmasq 2.76
| dns-nsid:
|_  bind.version: dnsmasq-2.76
80/tcp    open  http    lighttpd 1.4.35
|_http-server-header: lighttpd/1.4.35
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
1712/tcp  open  upnp    Platinum UPnP 1.0.5.13 (UPnP/1.0 DLNADOC/1.50)
32400/tcp open  http    Plex Media Server httpd
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|_  Server returned status 401 but no WWW-Authenticate header.
|_http-cors: HEAD GET POST PUT DELETE OPTIONS
|_http-favicon: Plex
|_http-title: Unauthorized
32469/tcp open  upnp    Platinum UPnP 1.0.5.13 (UPnP/1.0 DLNADOC/1.50)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 712.34 seconds
```

Here are some things we found:

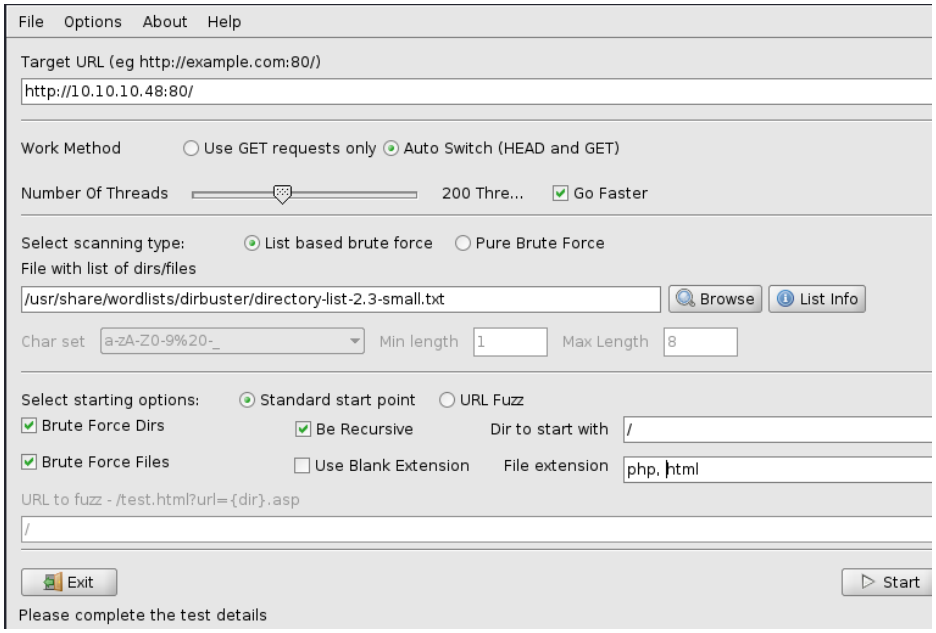
- We see that there is an SSH service running. In addition, port 80 is also open. There is also a DNS server.
- SSH has historically been secure, and the only thing we can really do with it is to try to brute-force the passwords. Let's check out port 80 before that.
- Also, we see that the operating system is likely Linux.

Directory Busting with Dirbuster

We go to 10.10.10.48, and we see a blank page. Are there directories we don't know about?

Directory busting is a technique that looks for hidden folders and files behind a url that are accessible by browser. For example, if you have a URL like 10.10.10.48, there may be a /admin page that exists and you wouldn't know without directory busting

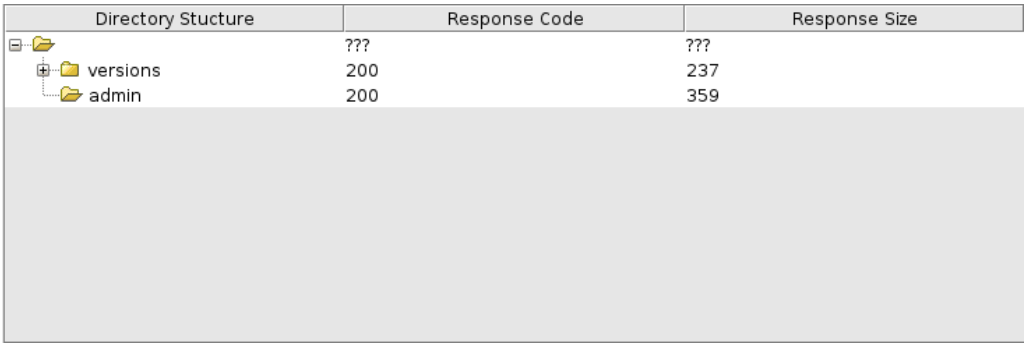
We start up dirbuster using the dirbuster command



Some features to note:

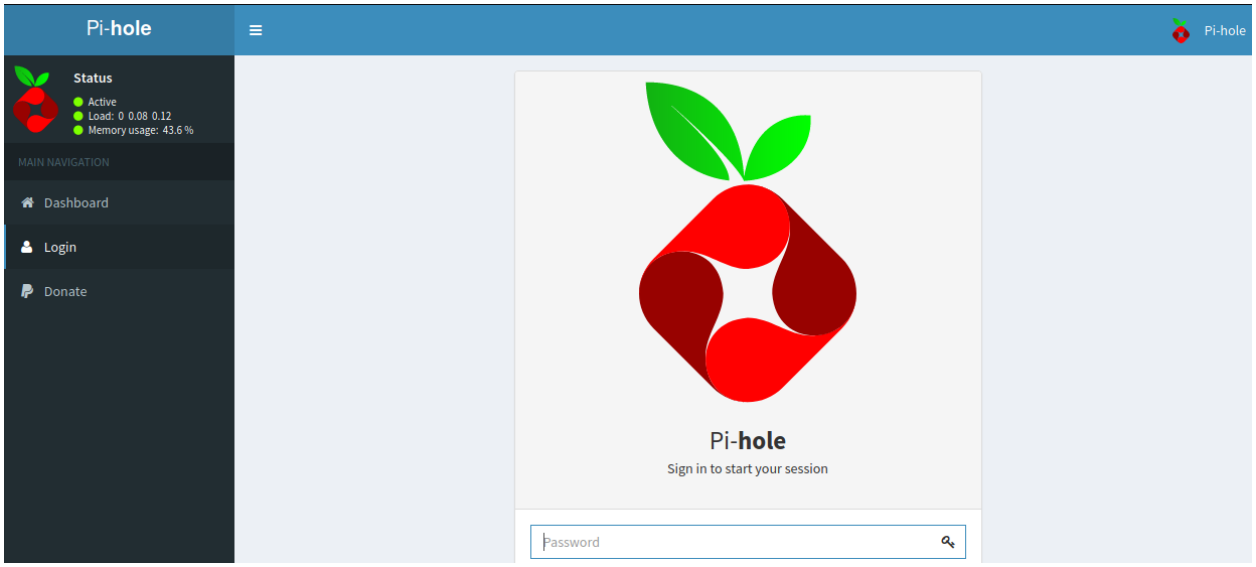
- Check the "go faster" option. There isn't really a reason to go slower in this case.
- The small wordlist should suffice for this task, as the target is not very advanced.

We press start, and get these results



We see an admin page. Let's check that out!

The Admin Page



Well...wouldn't you agree that the machine may be a Raspberry Pi?

The target machine is most likely a Raspberry Pi. Many other walkthroughs say that this has something to do with Mirai botnets, but I wouldn't know.

Anyways, we know that the Mirai botnet gained access to machines through default credentials. Let's try that.

A quick google search tells us that the default username/password combination for a Raspberry Pi machine is **pi:raspberry.**
This combination does not work for the pi-hole admin login, but there is another place we can try it…

SSH Default Credentials

SSH stands for Secure Shell. If you can log in, you are usually allowed remote command execution.

Yup, the default credentials log us into the SSH server! You can do this by typing the following into the terminal:

```
┌──(kali㉿kali)-[~]
└─$ ssh pi@10.10.10.48
pi@10.10.10.48's password: <raspberry>
```

We now have a shell on the machine! Let's look around and find some flags

Post-‘Exploitation’

Did we really exploit anything?

On a Linux machine, we can type sudo -l to check what privileges we have.

```
pi@raspberrypi:~ $ sudo -l
Matching Defaults entries for pi on localhost:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User pi may run the following commands on localhost:
    (ALL : ALL) ALL
    (ALL) NOPASSWD: ALL
pi@raspberrypi:~ $
```

We see that we already have administrator privileges. We don’t have to do any privilege escalation here!

We can find the user flag easily. However, we can’t get into the /root folder to find the root flag. Since we have all permissions, we can simply switch to the root user to access the folder:

```
pi@raspberrypi:/ $ cd root
-bash: cd: root: Permission denied
pi@raspberrypi:/ $ sudo su root
root@raspberrypi:/#
```

And now we can access the root.txt. Easy!

```
root@raspberrypi:~# cat root.txt
I lost my original root.txt! I think I may have a backup on my USB stick...
root@raspberrypi:~#
```

...Well that’s awkward...let’s find the USB stick

USB shenanigans

```
root@raspberrypi:/home/pi# df
Filesystem     1K-blocks    Used Available Use% Mounted on
aufs             8856504 2842920   5540652  34% /
tmpfs             102396    4896     97500   5% /run
/dev/sda1        1354528 1354528         0 100% /lib/live/mount/persistence/sda1
/dev/loop0       1267456 1267456         0 100% /lib/live/mount/rootfs/filesystem.squashfs
tmpfs             255988       0    255988   0% /lib/live/mount/overlay
/dev/sda2        8856504 2842920   5540652  34% /lib/live/mount/persistence/sda2
devtmpfs           10240       0     10240   0% /dev
tmpfs             255988       8    255980   1% /dev/shm
tmpfs               5120       4      5116   1% /run/lock
tmpfs             255988       0    255988   0% /sys/fs/cgroup
tmpfs             255988       8    255980   1% /tmp
/dev/sdb            8887      93      8078   2% /media/usbstick
tmpfs              51200       0     51200   0% /run/user/999
tmpfs              51200       0     51200   0% /run/user/1000
root@raspberrypi:/home/pi#
```

The ‘df’ command tells us about filesystems that are mounted on the machine. Upon closer inspection, we see that /media/usbstick is mounted on /dev/sdb

In Linux, the first and second hard disks are typically named sda and sdb. Just a fun fact.
(Honestly, I need to understand this better because I’m new to Linux and filesystems in general. I will update this once I gain a better understanding of it)

After going into the directory, we find yet another awkward moment. James is an idiot! Even more than myself! How will we get the root text??

```
root@raspberrypi:/home/pi# cd /media/usbstick
root@raspberrypi:/media/usbstick# ls
damnit.txt  lost+found
root@raspberrypi:/media/usbstick# cat damnit.txt
Damnit! Sorry man I accidentally deleted your files off the USB stick.
Do you know if there is any way to get them back?

-James
root@raspberrypi:/media/usbstick#
```

Finding the Root Flag

From here, you can run the following command which prints the content of the USB stick, including the flag that was lost

```
root@raspberrypi:/# cat dev/sdb
```

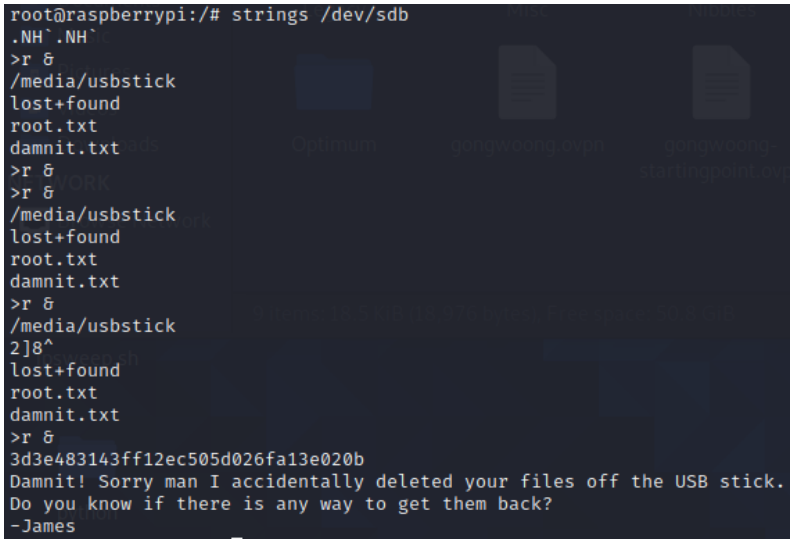
```
lost+found
root.txt
damnit.txt;9Y2Ho;9+/
,.O0+-t0Y1Y1Y1Y
 o!:2Y:2Y:2Y
* !92Y2Y2Y
+ !9;9Y3
8PP
(["  1YS1Y
<Byc[B)r &<yZ.Gum^>
1Y
*,.3d3e483143ff12ec505d026fa13e020b
Damnit! Sorry man I accidentally deleted your files off the USB stick.
Do you know if there is any way to get them back?

-James
root@raspberrypi:/#
```

And there it is!

A more ‘Elegant’ way to find this flag, according to other walkthroughs, is to use the strings command. We can run strings /dev/sdb

The strings command looks for bits of readable text within a large file, which is really useful in this situation as there are a lot of random characters to sort through...imagine if the file was actually somewhat large!



And there’s the flag, but easier!