PSP0201
Week 5
Writeup

Group Name : Fsociety
Members :

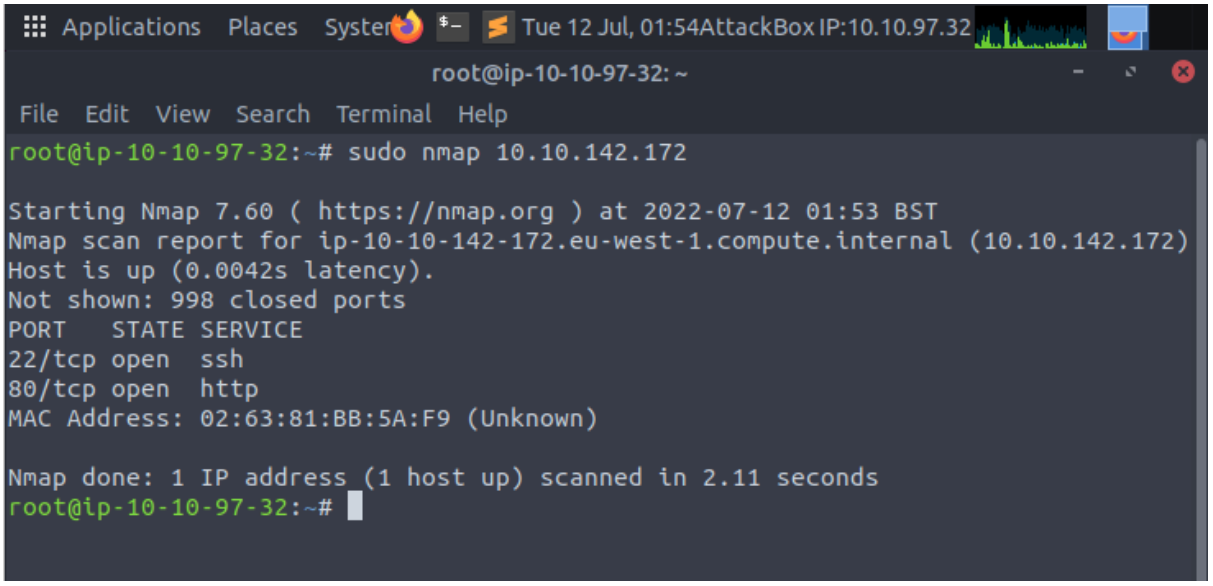
ID	Name	Role
1211102908	Wan Muhammad Ilhan Bin Wan Zil Azhar	Leader
1211101583	Luqman Hakim Bin Noorazmi	Member
1211203101	Jazlan Zuhair Bin Mohamed Zafrualam	Member
1211102054	Mithesh Kumar	Member

Day 16
(Help! Where is Santa?)

Tools used: Firefox, Terminal

Question 1:

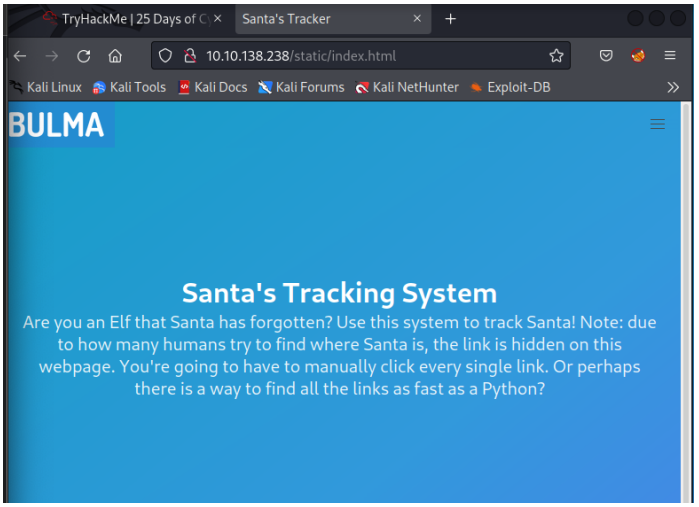
We used nmap to find the port.



The port number for the web server is 80

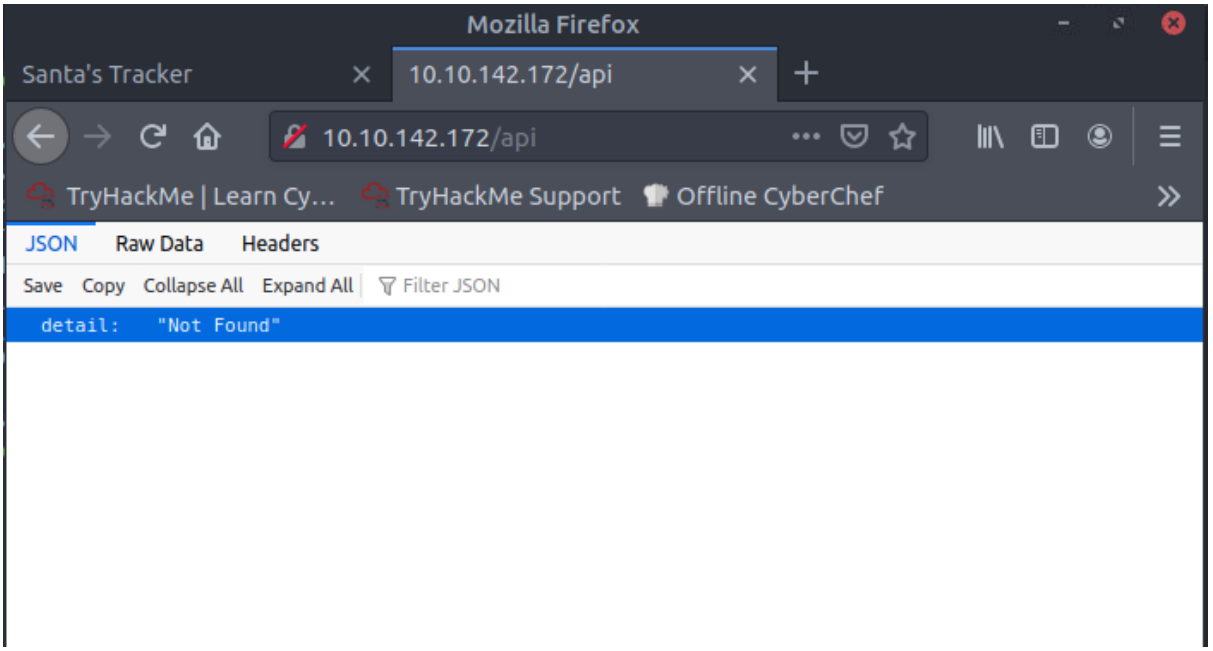
Question 2:

The template that is being used is BULMA.



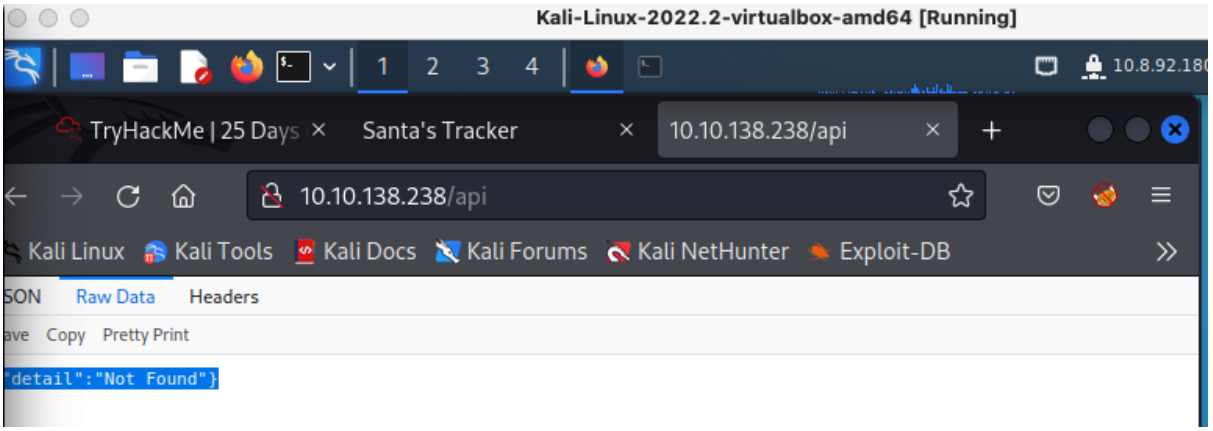
Question 3:

The directory for the API is */api*



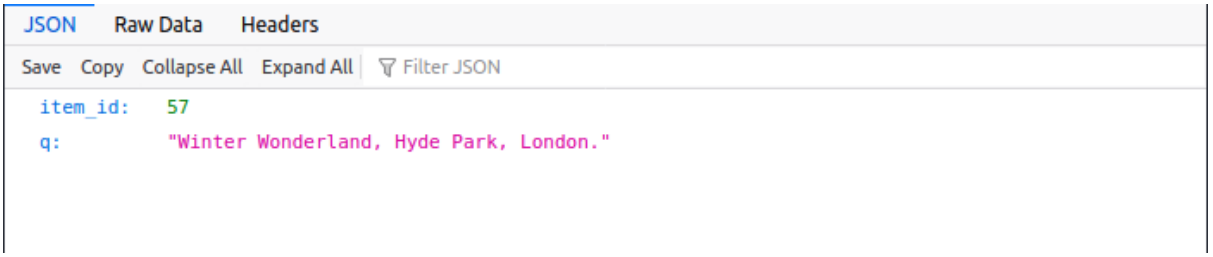
Question 4:

If no parameters are entered, it will return *{"detail":"Not Found"}*



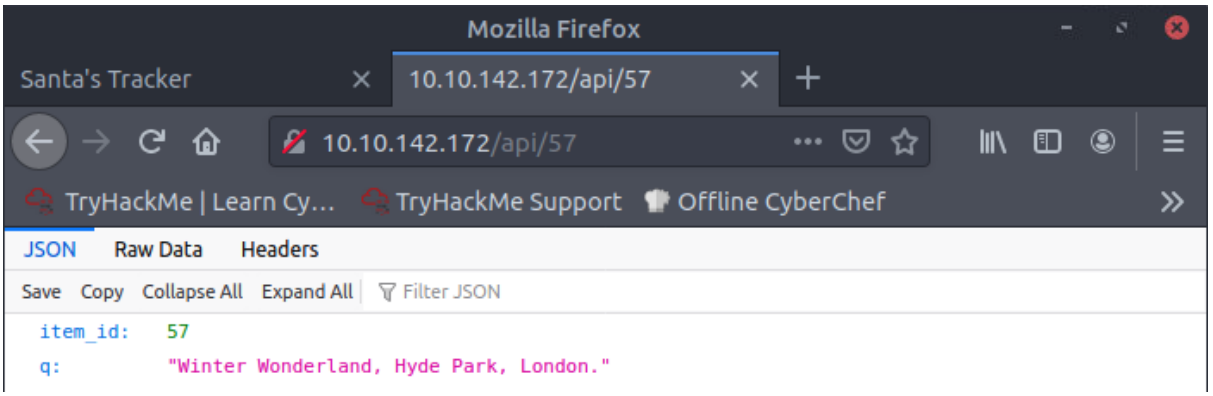
Question 5:

After figuring out the API key, we can now see Santa's location which is Winter Wonderland, Hyde Park, London.



Question 6:

After several attempts, the API key turned out to be 57.



Thought Process / Methodology:

After using nmap to find the port, head to the specified URL. On the top left of the website we can see the website template. To find the directory for the API, navigate to <Machine_IP>/api. We can now see how the website looks when there is no api key inserted on the URL. Finding the api key will take some amount of time of trial and error, but when the input for the api key is correct, the website will reveal Santa's location.

Day 17
(ReverseELFneering)

Tools used: Firefox, Terminal, Kali, Radare2

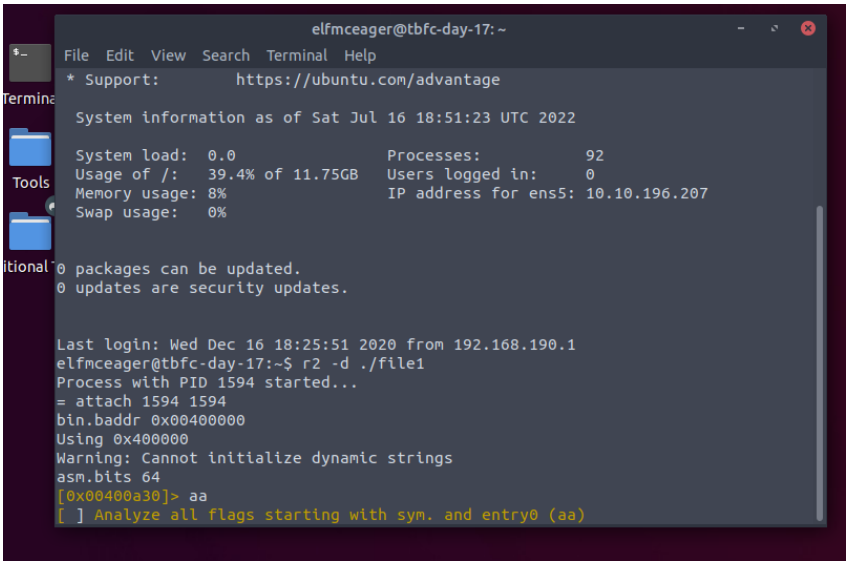
Question 1:

According to the information table below, we can figure out the data type and sizes :

Initial Data Type	Suffix	Size (bytes)
Byte	b	1
Word	w	2
Double Word	l	4
Quad	q	8
Single Precision	s	4
Double Precision	l	8

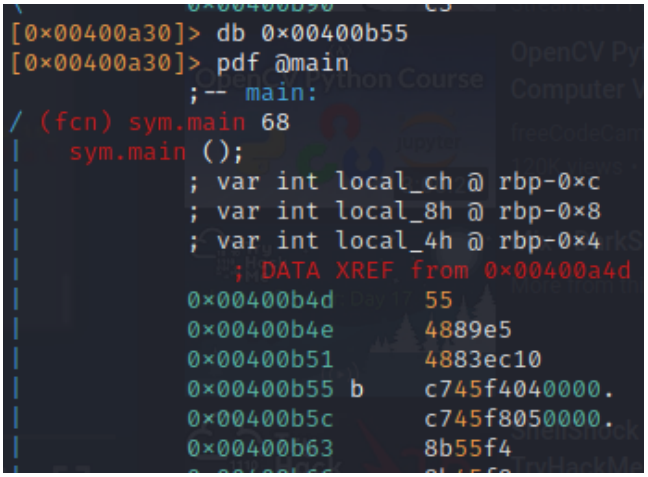
Question 2:

After connecting to radare2, we can analyse the program using the command : aa



Question 3:

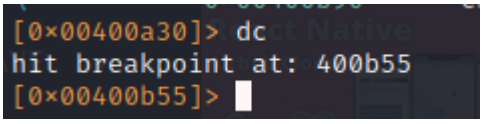
We can use the command `db <address>` to set a breakpoint.



To confirm that breakpoint has been set, check the address and find the letter `b` next to it. We've successfully set a breakpoint.

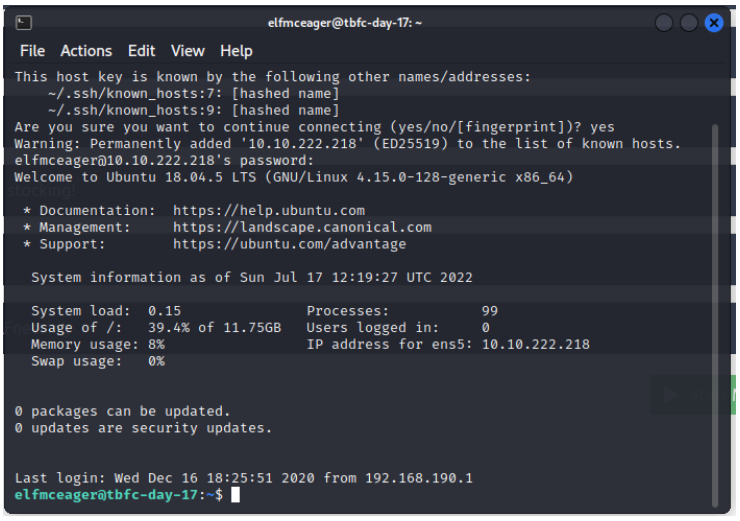
Question 4:

After setting up the breakpoint, we can use the command `dc` to execute radare2 until it hits the breakpoint.

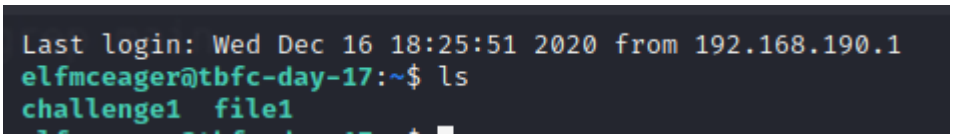


Question 5:

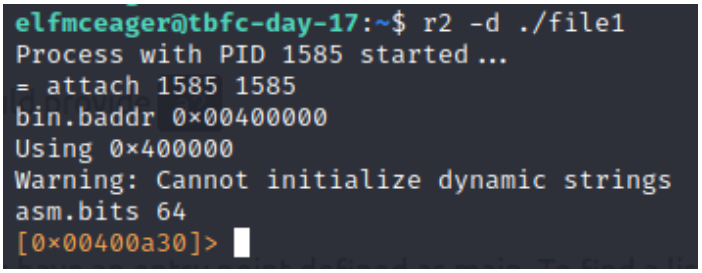
First, we need to use SSH to enter the target machine.



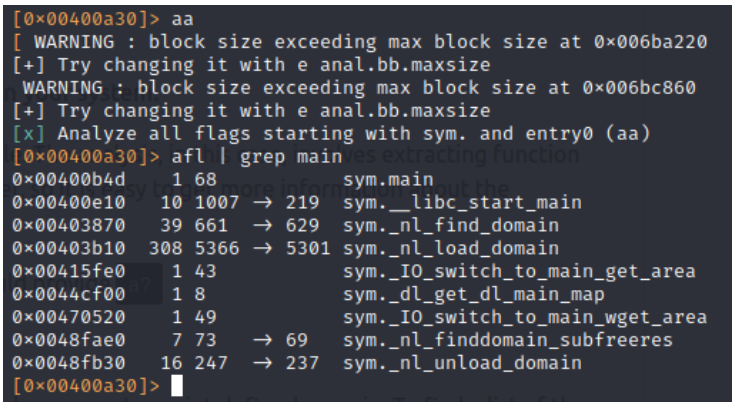
Next, we use the ls command to list all files found in the target directory. Here, we found challenge1 and file1.



We can then proceed to use radare2 to debug the file.

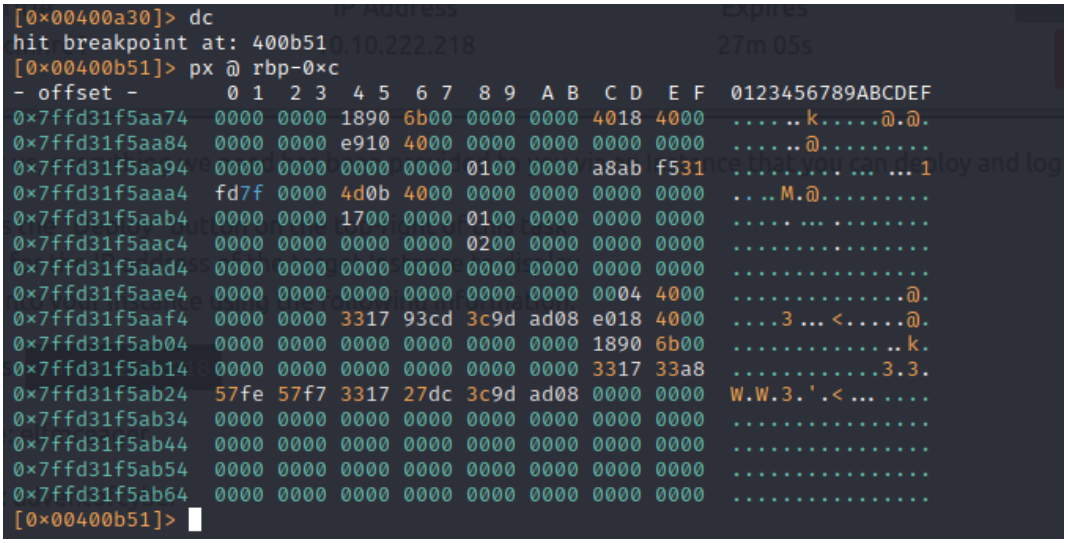


Using the command aa, we analyze the program and use afl | grep main to find the main function out of the entire list of functions.



After using the command pdf @main to get the main function, we can see local_ch. Use db to set up a breakpoint for local_ch, in this case it's db 0x00400b51.

We can now use dc to execute until breakpoint is hit, then using px @ rbp-0xc, we can see the current memory value. Right now it is 0, but we can't confirm it until we run our ds command.



From here, run the ds command and check the value again. We can see that our local_ch value is 1.

Question 6:

Again, use pdf @main to check which value has imul eax. In this case, 0x0040062 is the memory we're looking for. Set a breakpoint towards the value, and use dc to hit it.

```
/ (fcn) sym.main 35
|   sym.main ();
|           ; var int local_ch @ rbp-0x c
|           ; var int local_8h @ rbp-0x8
|           ; var int local_4h @ rbp-0x4
|              ; DATA XREF from 0x00400a4d (entry0)
|           0x00400b4d      55              push rbp
|           0x00400b4e      4889e5          mov rbp, rsp
|           ;-- rip:
|           0x00400b51      c745f4010000.   mov dword [local_ch], 1
|           0x00400b58      c745f8060000.   mov dword [local_8h], 6
|           0x00400b5f      8b45f4          mov eax, dword [local_ch]
|           0x00400b62      0faf45f8        imul eax, dword [local_8h]
|           0x00400b66      8945fc          mov dword [local_4h], eax
|           0x00400b69      b800000000      mov eax, 0
|           0x00400b6e      5d              pop rbp
\           0x00400b6f      c3              ret
[0x00400b51]> db 0x00400b62
[0x00400b51]> dc
hit breakpoint at: 400b62
```

Here, we can use dr to see the current value. Right now, it is 1.

```
[0x00400b51]> dc
hit breakpoint at: 400b62
[0x00400b51]> dr
rax = 0x00000001
rbx = 0x00400400
rcx = 0x0044b9a0
rdx = 0x7fffbac0be98
r8 = 0x01000000
r9 = 0x006bb8e0
r10 = 0x00000015
r11 = 0x00000000
r12 = 0x004018e0
r13 = 0x00000000
r14 = 0x006b9018
r15 = 0x00000000
rsi = 0x7fffbac0be88
rdi = 0x00000001
rsp = 0x7fffbac0bd60
rbp = 0x7fffbac0bd60
rip = 0x00400b62
rflags = 0x00000246
orax = 0xffffffffffffffff
```

Now, using ds to execute the program. Check dr again to see the value. Here, we can see the value is 6.

```
[0×00400b51]> dr
rax = 0×00000006
rbx = 0×00400400
rcx = 0×0044b9a0
rdx = 0×7fffbac0be98
r8 = 0×01000000
r9 = 0×006bb8e0
r10 = 0×00000015
r11 = 0×00000000
r12 = 0×004018e0
r13 = 0×00000000
r14 = 0×006b9018
r15 = 0×00000000
rsi = 0×7fffbac0be88
rdi = 0×00000001
rsp = 0×7fffbac0bd60
rbp = 0×7fffbac0bd60
rip = 0×00400b66
rflags = 0×00000246
orax = 0×ffffffffffffffff
```

Question 7:

First, set a breakpoint at the local_4h line.

```
[0×00400b51]> pdf @main
            ;-- main:
/ (fcn) sym.main 35
|   sym.main ();
|           ; var int local_ch @ rbp-0×c
|           ; var int local_8h @ rbp-0×8
|           ; var int local_4h @ rbp-0×4
|              ; DATA XREF from 0×00400a4d (entry0)
|           0×00400b4d      55              push rbp
|           0×00400b4e      4889e5          mov rbp, rsp
|           0×00400b51      c745f4010000.   mov dword [local_ch], 1
|           0×00400b58      c745f8060000.   mov dword [local_8h], 6
|           0×00400b5f      8b45f4          mov eax, dword [local_ch]
|           0×00400b62 b    0faf45f8        imul eax, dword [local_8h]
|           ;-- rip:
|           0×00400b66      8945fc          mov dword [local_4h], eax
|           0×00400b69      b800000000      mov eax, 0
|           0×00400b6e      5d              pop rbp
\           0×00400b6f      c3              ret
[0×00400b51]> db 0×00400b66
```

Use the dc command to execute the program until it hits a breakpoint, then use the ds command.

```
[0×00400b51]> dc
hit breakpoint at: 400b66
[0×00400b51]> ds
```

After checking the value using `px @ rbp-0x4`, we can see the value which is 6.

```
[0×00400b51]> px @ rbp-0×4
- offset -       0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0×7fffbac0bd5c  0600 0000 4018 4000 0000 0000 e910 4000  ....@.@.......@.
0×7fffbac0bd6c  0000 0000 0000 0000 0000 0000 0000 0000  ................
0×7fffbac0bd7c  0100 0000 88be c0ba ff7f 0000 4d0b 4000  ............M.@.
0×7fffbac0bd8c  0000 0000 0000 0000 0000 0000 1700 0000  ................
0×7fffbac0bd9c  0100 0000 0000 0000 0000 0000 0000 0000  ................
0×7fffbac0bdac  0200 0000 0000 0000 0000 0000 0000 0000  ................
0×7fffbac0bdbc  0000 0000 0000 0000 0000 0000 0000 0000  ................
0×7fffbac0bdcc  0000 0000 0004 4000 0000 0000 4aa1 7a7a  ......@.....J.zz
0×7fffbac0bddc  8184 9eaf e018 4000 0000 0000 0000 0000  ......@.........
0×7fffbac0bdec  0000 0000 1890 6b00 0000 0000 0000 0000  ......k.........
0×7fffbac0bdfc  0000 0000 4aa1 1a30 80f1 6150 4aa1 ce6b  ....J..0..aPJ..k
0×7fffbac0be0c  8184 9eaf 0000 0000 0000 0000 0000 0000  ................
0×7fffbac0be1c  0000 0000 0000 0000 0000 0000 0000 0000  ................
0×7fffbac0be2c  0000 0000 0000 0000 0000 0000 0000 0000  ................
0×7fffbac0be3c  0000 0000 0000 0000 0000 0000 0000 0000  ................
0×7fffbac0be4c  0000 0000 0000 0000 0000 0000 0000 0000  ................
```

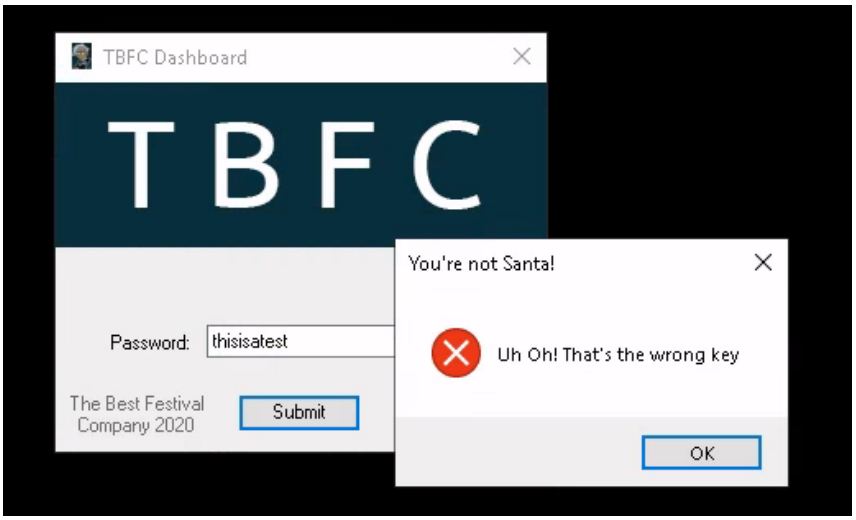
Thought Process/Methodology :

Reading through the radare2 guidelines and commands, I was struggling to understand the functions at first, it is harder than it looks. But the main commands which consisted of db, dc, ds, px and pdf are easier to understand. For the first question, we looked through the information table which reveals the data type and it's sizes. Question 2, 3 and 4 requires understanding of the command lines in a nutshell. For question 5, we started by going through the list of commands and using breakpoints to find out the value. We figured that by using dc and ds, we can eventually find the correct value of the source. Question 6 is a bit trickier, but by figuring out that the local_ch value has been moved to the eax register, we can simply use db and dr to check the value. Once executed, use dr again and we get the final answer. Finally, we set a breakpoint at the local_ch line and execute dc. We then checked the value by using px. This time around, we already used ds so 6 is our final answer.

Day 18
(The Bits of Christmas)
Tools used: Firefox, Terminal, Kali, Remmina, ILSpy

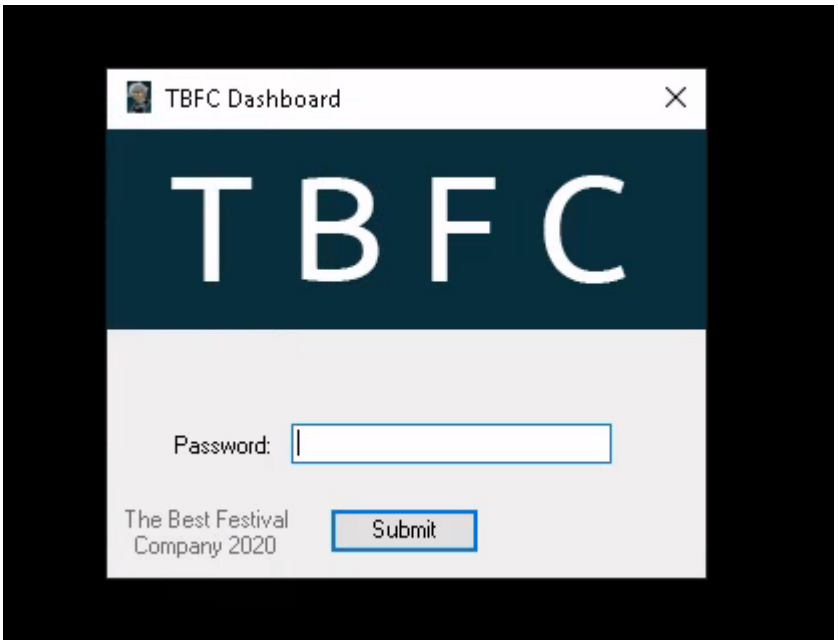
Question 1:

We used Remmina provided in AttackBox to connect a remote access machine that has our TBFC app in, the message that shows up if you enter the wrong password for TBFC_APP is: *You're not Santa!*



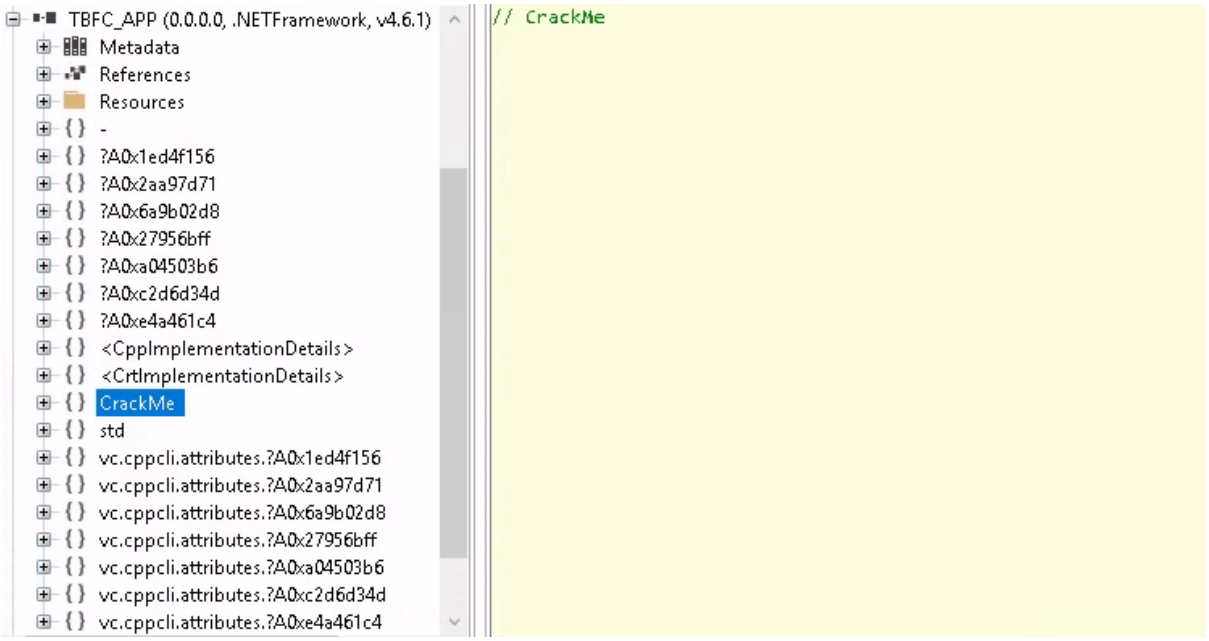
Question 2:

TBFC stands for: *The Best Festival Company*



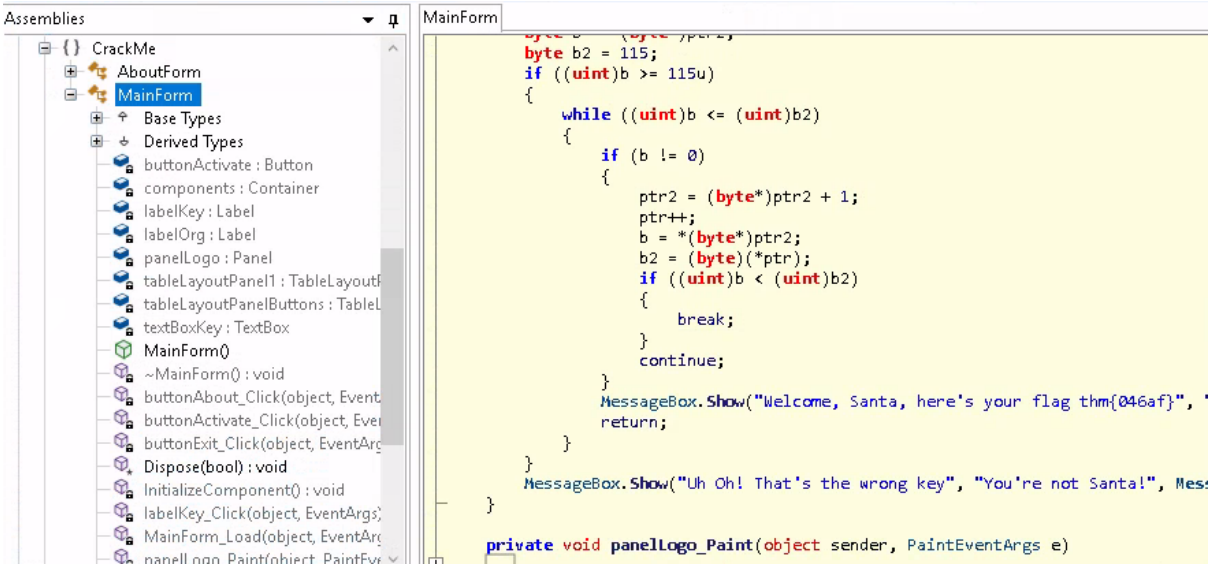
Question 3:

The module that catches my attention is: *CrackMe*



Question 4:

The module that contains the information we are looking for is: *MainForm*

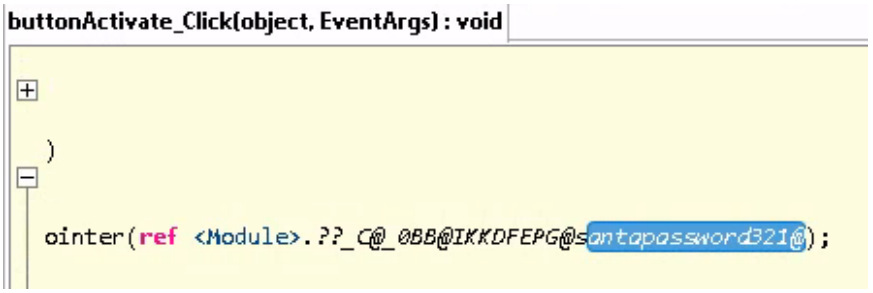


Question 5:

The method within the form that will contain the information we are seeking is: *buttonActivate_Click*

Question 6:

What is Santa's password?: *santapassword321*



Question 7:

The flag is: *thm{046af}*

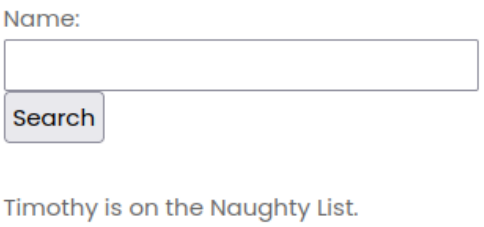
Thought Process/Methodology : We started off by connecting to the remote accessed machine through Remmina, and once connected with the IP address and credentials provided in the challenge, we were able to access the machine that included the TBFC file and ILSpy to use, opening the TBFC program lead us to login screen that have no other way of breaking in, so we used ILSpy to dissect the program as its written in dotNet. Upon further inspecting, we found a section named CrackMe and began finding its code, we soon found the appropriate credentials needed for this challenge.

Day 19
(The Naughty or Nice List)

Tools used: Kali, Firefox

Question 1:

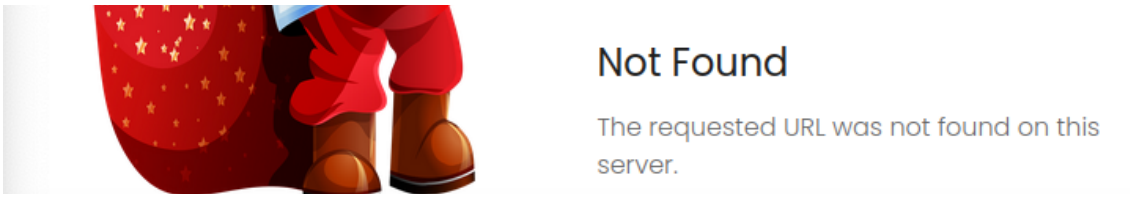
To see whether the names are on the nice or naughty list, enter the name on the website's search bar.



Timothy	Naughty
YP	Nice
Tib3rius	Nice
Kanes	Naughty
Ian Cha	Naughty
JJ	Naughty

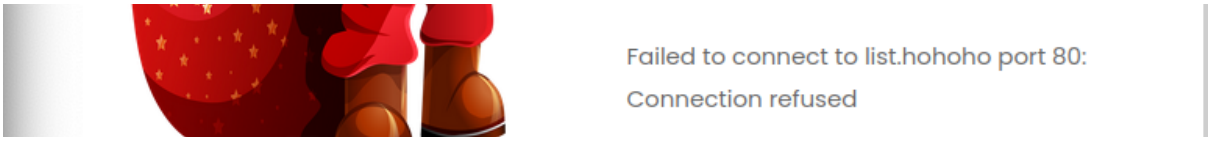
Question 2:

When using /?proxy=http%3A%2F%2Flist.hohoho%3A8080%2F on the URL, it will display *'The requested URL was not found on this server'*.



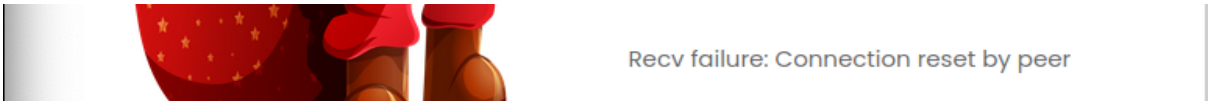
Question 3:

When using /?proxy=http%3A%2F%2Flist.hohoho%3A80 on the URL, it will display *'Failed to connect to list.hohoho port 80: Connection refused'*.



Question 4:

When using /?proxy=http%3A%2F%2Flist.hohoho%3A22 on the URL, it will display '*Recv failure: Connection reset by peer*'.

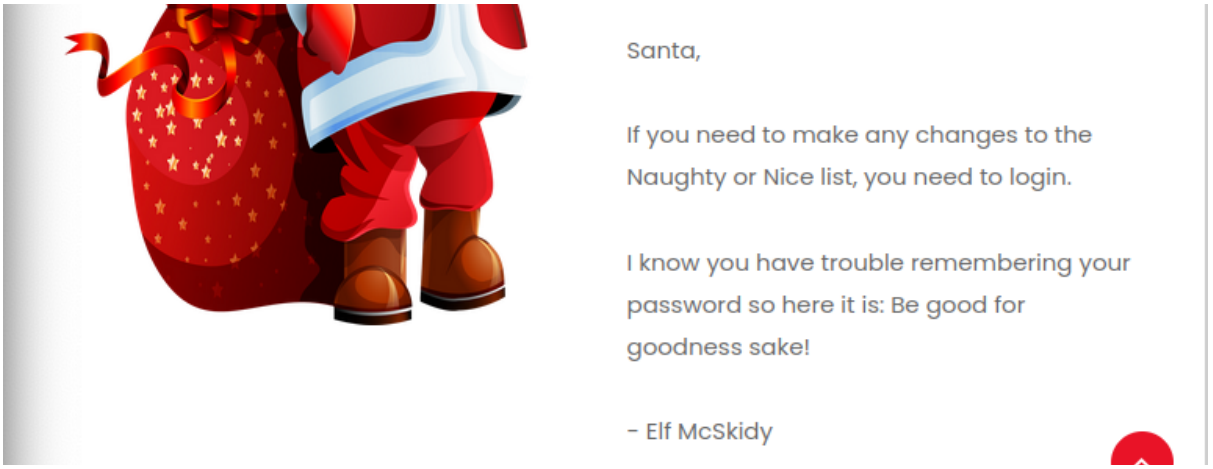


Question 5:

When using /?proxy=http%3A%2F%2Flocalhost on the URL, it will display '*Your search has been blocked by our security team*'.

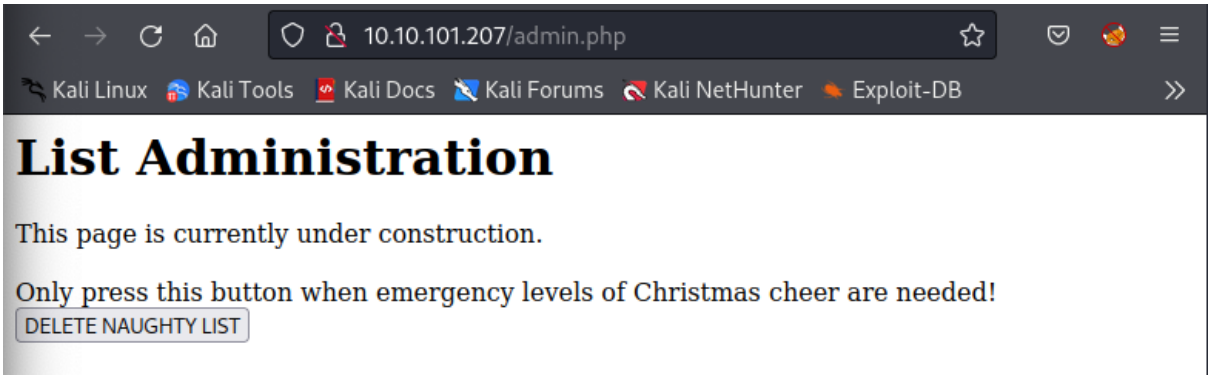
Question 6:

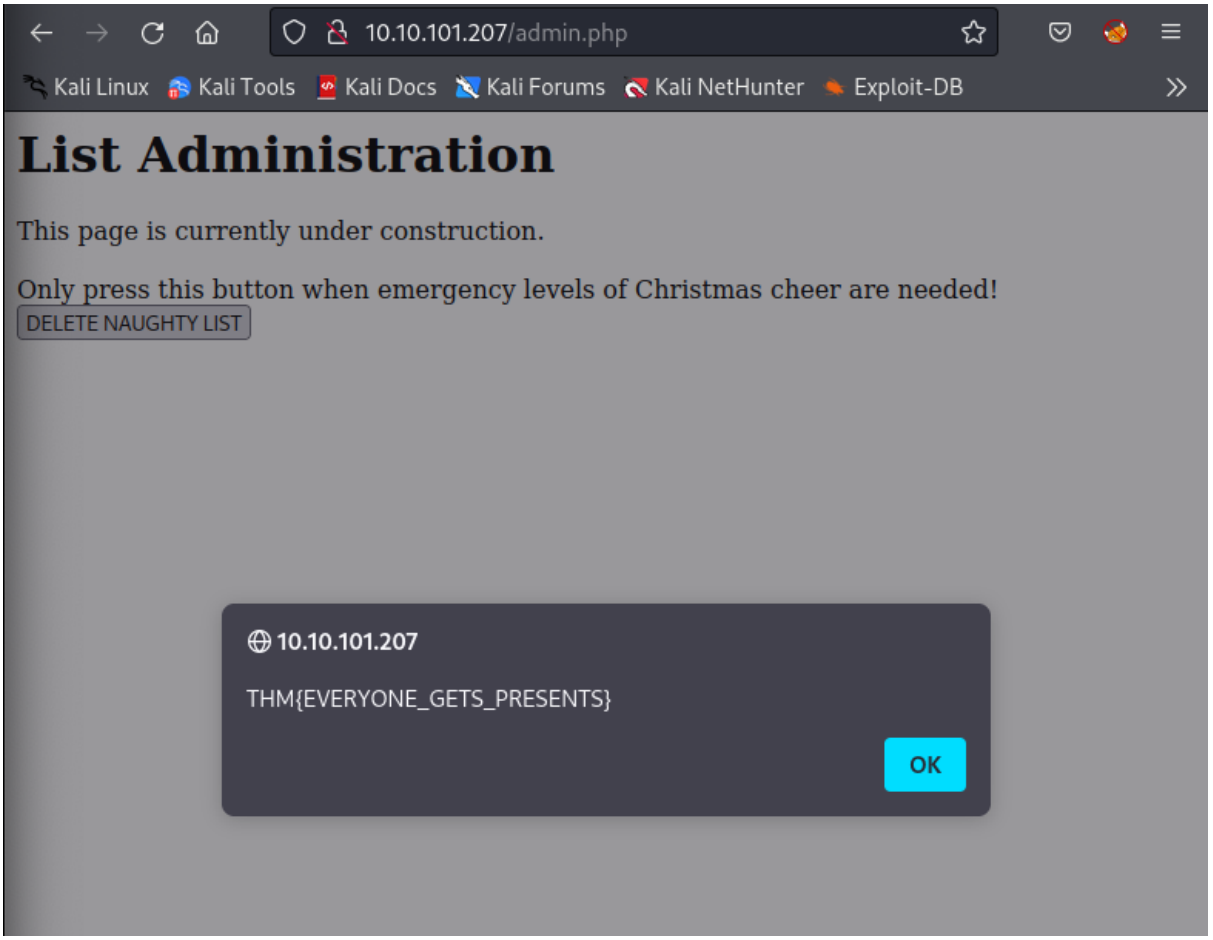
Santa’s password can be obtained by using ?proxy=http%3A%2F%2Flist.hohoho.localtest.me on the URL.



Question 7:

Scroll down and login using the given password. The username is Santa. On this new page, click ‘DELETE NAUGHTY LIST’ and the flag will be revealed.





Thought Process/Methodology :

After navigating to the website, the search bar is to see whether someone is on the naughty or the nice list. Now let's play with the URL a bit. Fetching the root of the same site will display the URL not found message. Now change the port to 80. It's now failed to connect. Using another port (22) will display a Recv failure. Lastly, change 'list.hohoho' on the URL to 'localhost' and the search is now blocked. To obtain the password, use 'list.hohoho.localtest.me' as the hostname to bypass the check. Scroll down until the login panel is visible. The username can be easily guessed. Log in and it will redirect to a new page. Click the DELETE NAUGHTY LIST button and the flag will show up.

Day 20
(PowershELIF to the rescue?)

Tools used: Firefox, Terminal, Kali, Google

Question 1:

The -l command is used to specify the *login name*.
(Via https://explainshell.com/explain?cmd=ssh+-L+-N+-f+-l)

```
-l login_name
       Specifies the user to log in as on the remote machine.  This also may be specified on a per-host
       basis in the configuration file.
```

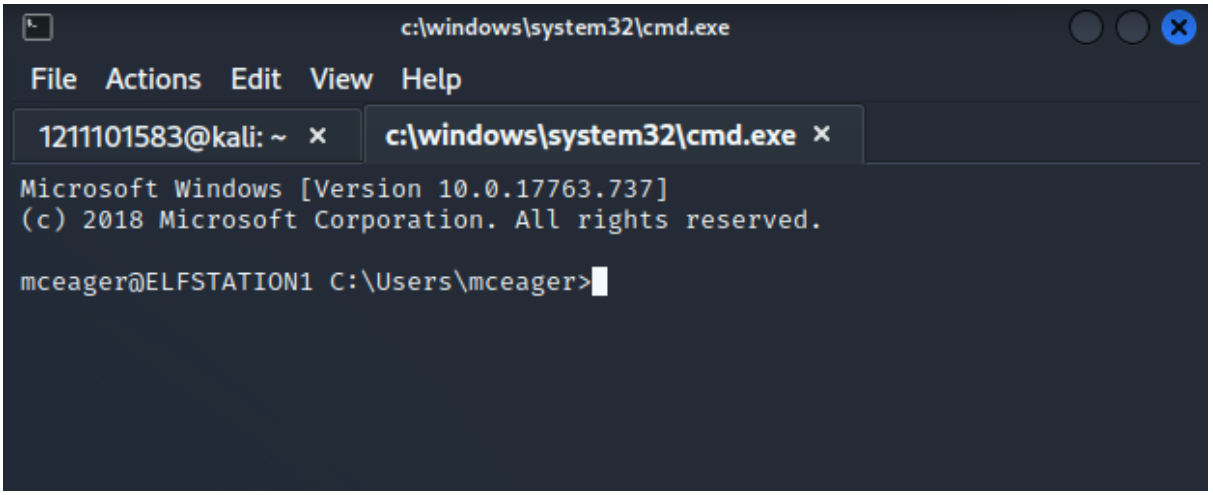
Question 2:

We start by connecting to the remote machine by doing the following command;

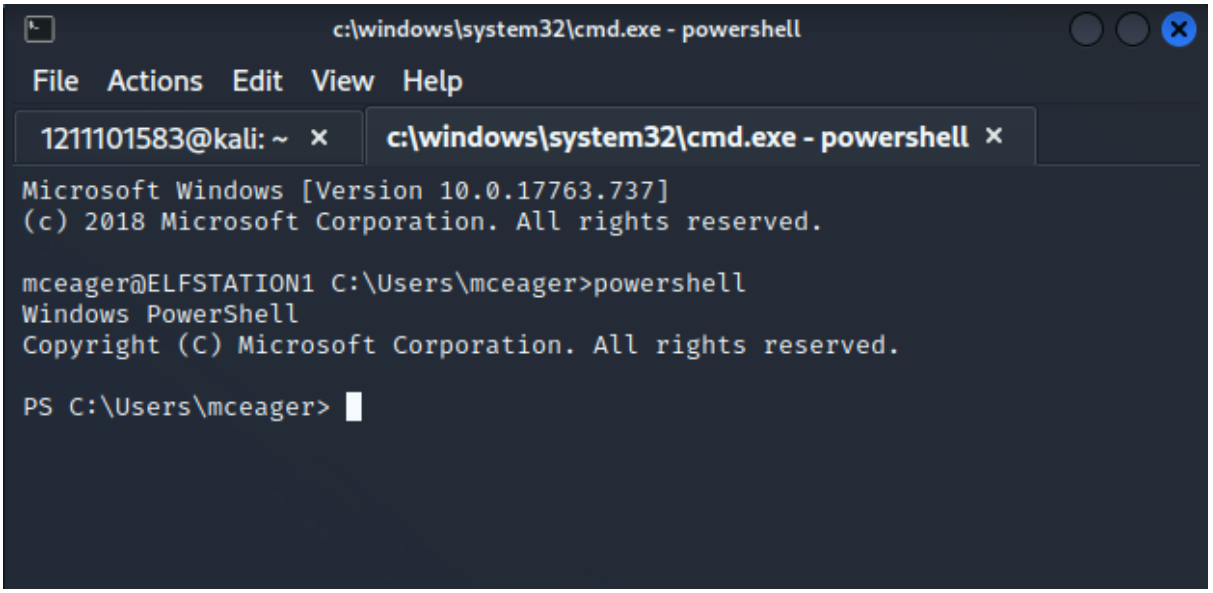
ssh -l mceager <Machine_IP>

When prompted for password, type in *r0ckStar!*

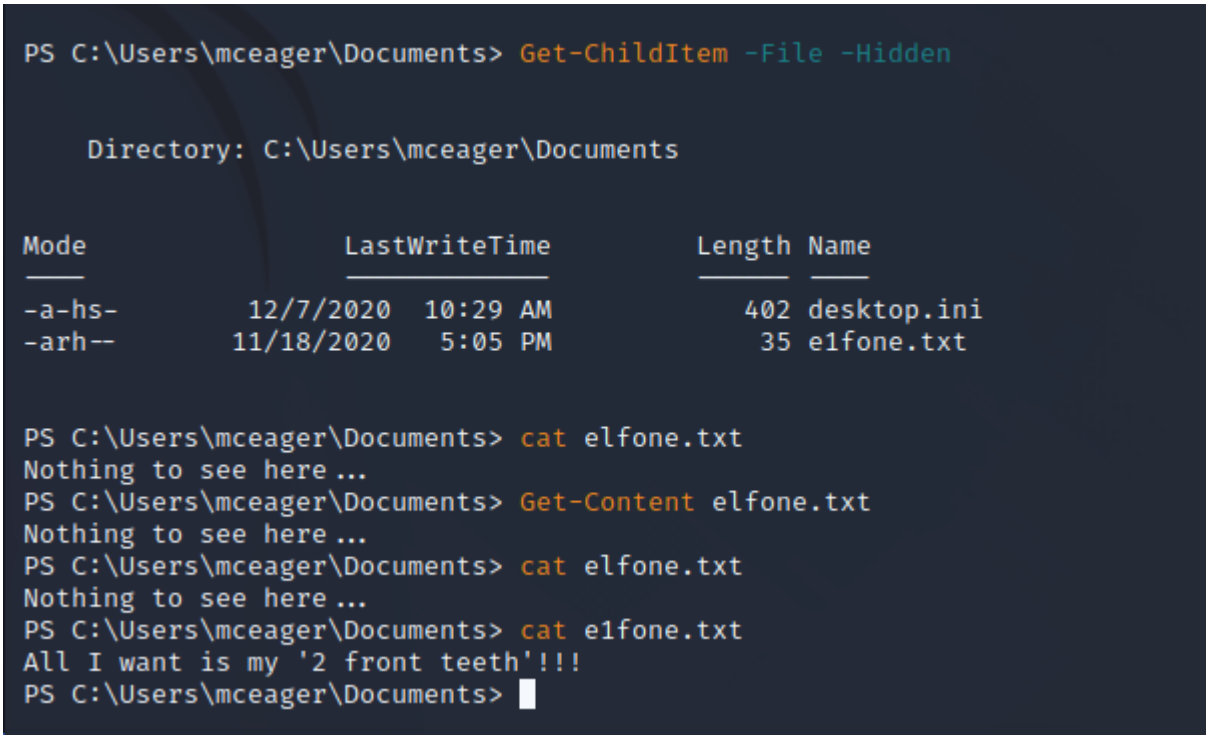
This window will appear when the login is successful



Type in *powershell*



Use Get-ChildrenItem -File- -Hidden to see the hidden files. As we can see, there's a hidden file called e1fone.txt. Then, use cat e1fone.txt to read the content of the file.



Question 3:

In the desktop directory, use ls -Hidden to see the hidden files. There's a file called elf2wo. Inside the file, there's a .txt file. Use the cat command to read the content of the file.

```
PS C:\Users\mceager\Desktop> ls -Hidden


    Directory: C:\Users\mceager\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d--h--         12/7/2020  11:26 AM                elf2wo
-a-hs-         12/7/2020  10:29 AM            282 desktop.ini


PS C:\Users\mceager\Desktop> cat elf2wo
cat : Access to the path 'C:\Users\mceager\Desktop\elf2wo' is denied.
At line:1 char:1
+ cat elf2wo
+ ~~~~~~~~~~
    + CategoryInfo          : PermissionDenied: (C:\Users\mceager\Deskto
   p\elf2wo:String) [Get-Content], UnauthorizedAccessException
    + FullyQualifiedErrorId : GetContentReaderUnauthorizedAccessError,Mi
   crosoft.PowerShell.Commands.GetContentCommand

PS C:\Users\mceager\Desktop> cd elf2wo
PS C:\Users\mceager\Desktop\elf2wo> ls


    Directory: C:\Users\mceager\Desktop\elf2wo


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         11/17/2020  10:26 AM             64 e70smsW10Y4k.txt


PS C:\Users\mceager\Desktop\elf2wo> cat e70smsW10Y4k.txt
I want the movie Scrooged <3!
PS C:\Users\mceager\Desktop\elf2wo>
```

Question 4:

To find the files for elf 3, use *Get-ChildItem -Hidden -Directory -Filter "*3*"* to find the File and its location.

```
PS C:\Windows\System32> Get-ChildItem -Hidden -Directory -Filter "*3*"


    Directory: C:\Windows\System32


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d--h--        11/23/2020   3:26 PM                3lfthr3e
```

Question 5:

In the file that we just discovered, use *Get-ChildItem -Hidden* to reveal the two .txt files.

```
PS C:\Windows\System32> cd 3lfthr3e
PS C:\Windows\System32\3lfthr3e> ls
PS C:\Windows\System32\3lfthr3e> Get-ChildItem -Hidden


    Directory: C:\Windows\System32\3lfthr3e


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-arh--        11/17/2020  10:58 AM          85887 1.txt
-arh--        11/23/2020   3:26 PM       12061168 2.txt
```

Then, use Get-Content 1.txt | Measure-Object -Word to find out the amount of words in the file.

```
PS C:\Windows\System32\3lfthr3e> Get-Content 1.txt | Measure-Object -Word

Lines Words Characters Property
----- ----- ---------- --------
       9999
```

Question 6:

Use *(Get-Content 1.txt)[551]* and *(Get-Content 1.txt)[6991]* to find out the word on the specified location in the file.

```
PS C:\Windows\System32\3lfthr3e> (Get-Content 1.txt)[551]
Red
PS C:\Windows\System32\3lfthr3e> (Get-Content 1.txt)[6991]
Ryder
```

Question 7:

To find out what elf3 wants, use *Get-Content 2.txt* | *Select-String -Pattern "redryder"*

```
PS C:\Windows\System32\3lfthr3e> Get-Content 2.txt | Select-String -Patter
n "redryder"

redryderbbgun
```

Thought Process / Methodology:

The use for the command -l can be found easily with a quick Google search. Then, log in to the remote machine using the provided username and password. After successfully logged in, navigate to the Documents directory and search for hidden files. Read the text file to expose the content. The next file can be found in Desktop. Inside the hidden file there is a .txt file in which you can read the content of the file to find the movie name. The next file requires some extra steps. Start by searching for the file's name and its location. Navigate to the file that was just discovered. Inside, there's two .txt files to be explored. For the first file, use the command that will show the amount of words in the file. For the next one, the location of the word needs to be specified in the command in order for it to reveal the two words. By combining the two words from the last question, we can easily know what elf3 wants.