PSP0201 Week 5 Writeup

Group Name: Hacktocrats

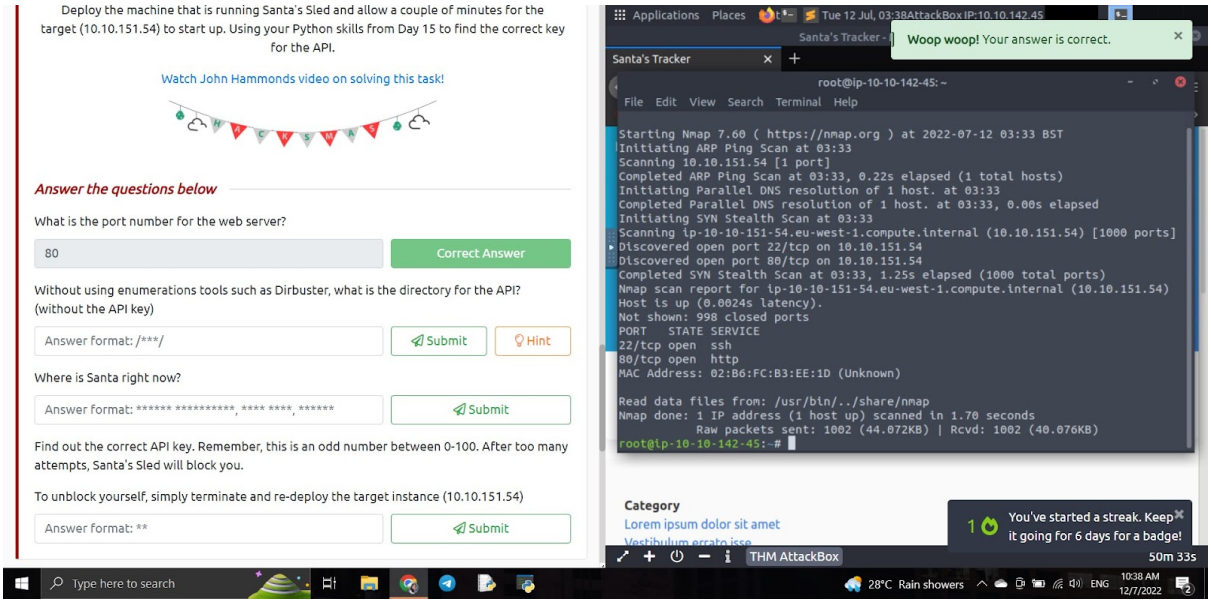
Members

ID	Name	Role
1211103194	NUR FARAHIYA AIDA BINTI ABD RAZAK	Leader
1211103602	NUR ALIA AMELISA SYAZREEN BINTI MOHD SULEI	Member
1211103430	AINA SOFEA BINTI AMIER HAMZAH	Member
1211103237	NURUL AIN BINTI KAMARUDIN	Member

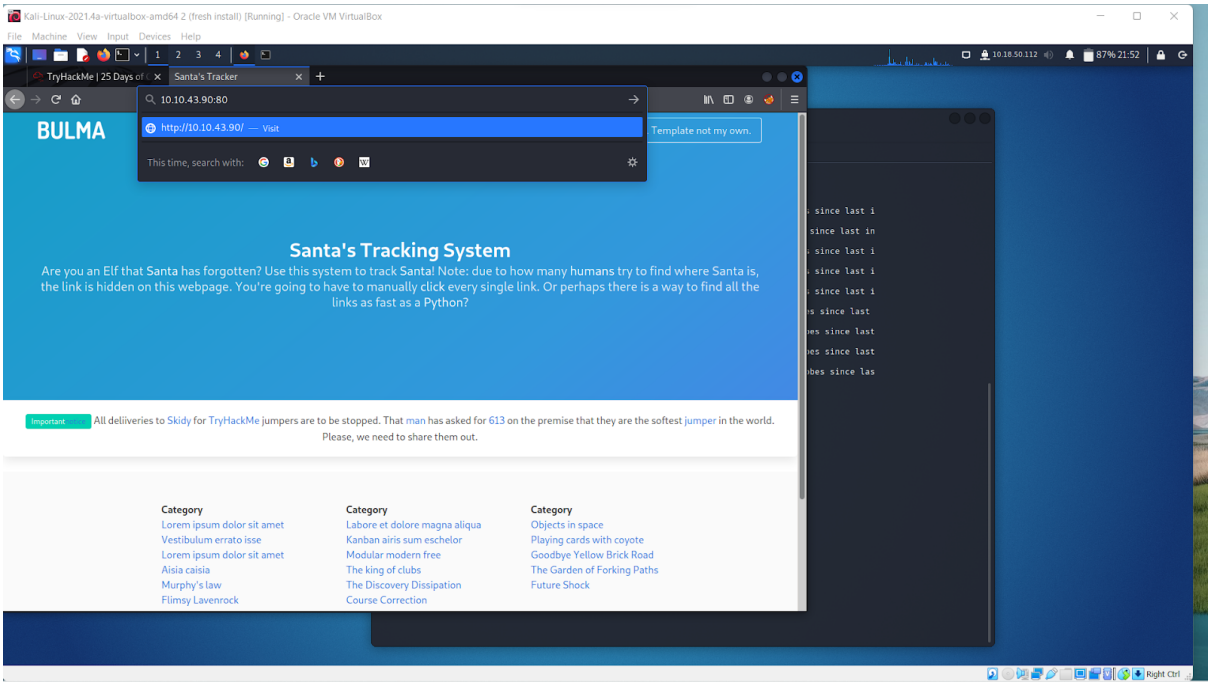
Day 16: Help! Where Is Santa?
Tools used: Terminal, Firefox

Solution / walkthrough:

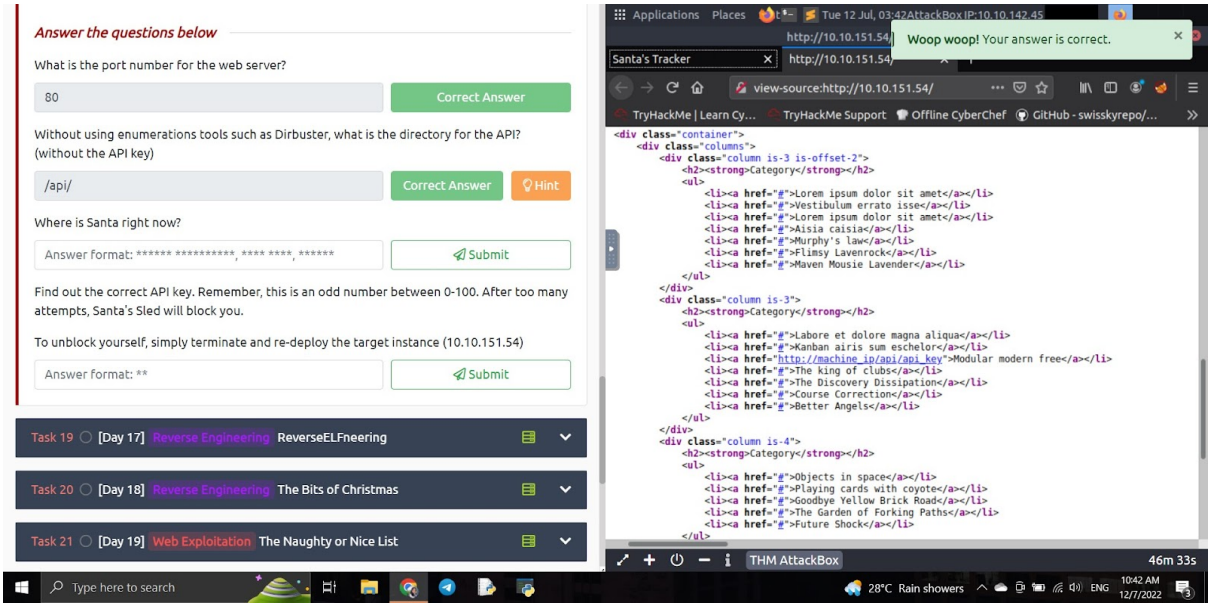
Question 1: nmap has been interred followed by our port number which lead us to know what was the port number for the web server



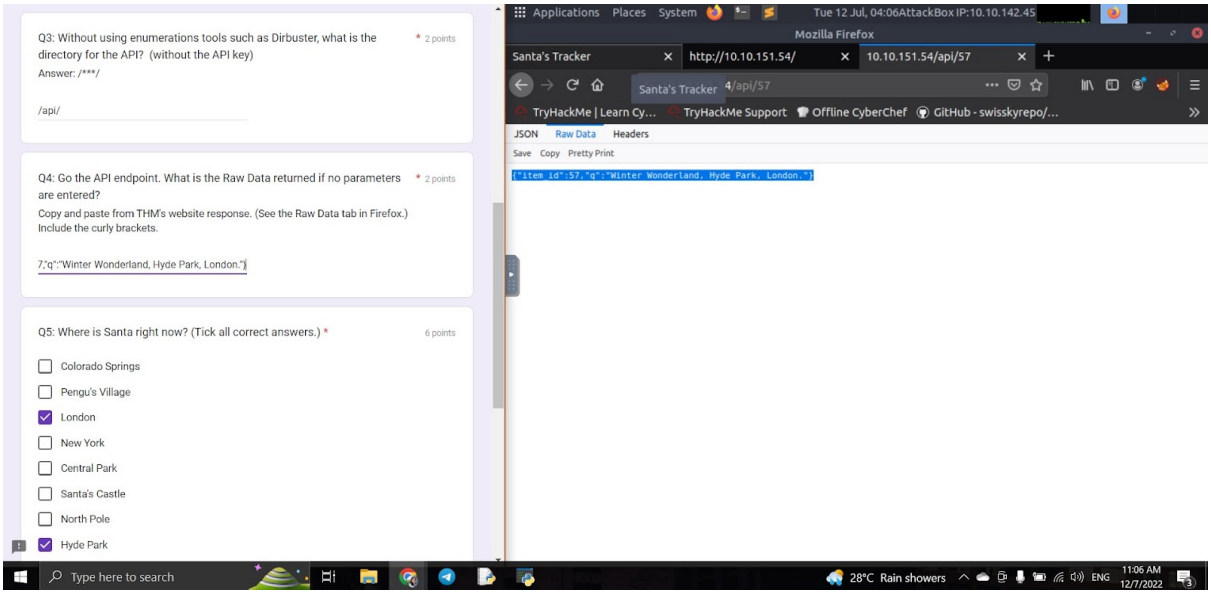
Question 2: this answer was obtained as we searched for our port number in firefox, it led us to the web browser. BULMA was clearly shown at the top left in the web browser.



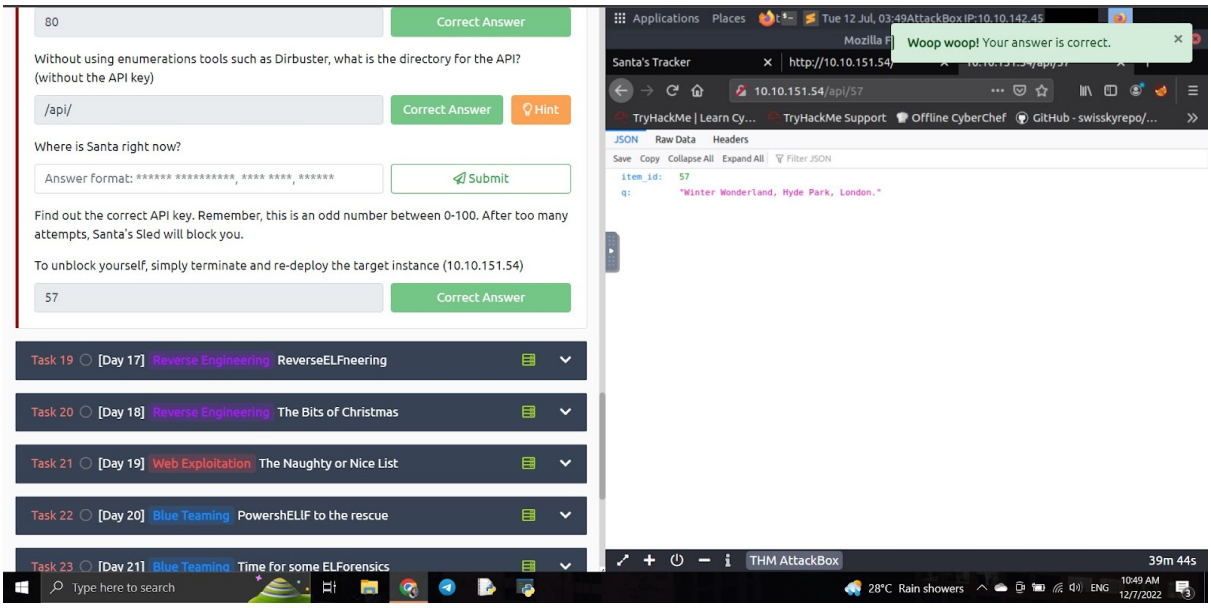
Question 3: Then we proceed to firefox. We searched http followed by our port number and /api/. By pressing *ctrl u* we have obtained the directory for the API without using enumerations tools



Question 4 & Question 5: answer for question 4 and question 5 has been obtained by entering (our port number/api/*the correct API key). It showed where Santa was.



Question 6: this need to be done by using the try and error method, we have tried a few numbers till we obtained the right key which was 57



Thought process and methodology:

First thing first, we use a map to scan the IP address and we get the port number of the IP Address. After that, we go to the web browser and search for "MACHINE_IP:port" using our IP address and the port number that we got before. Then we view the page source of the website to find the directory for the API. From what we learn on day 15, we use python to obtain the correct API key. Finally we are able to know the information about Santa.

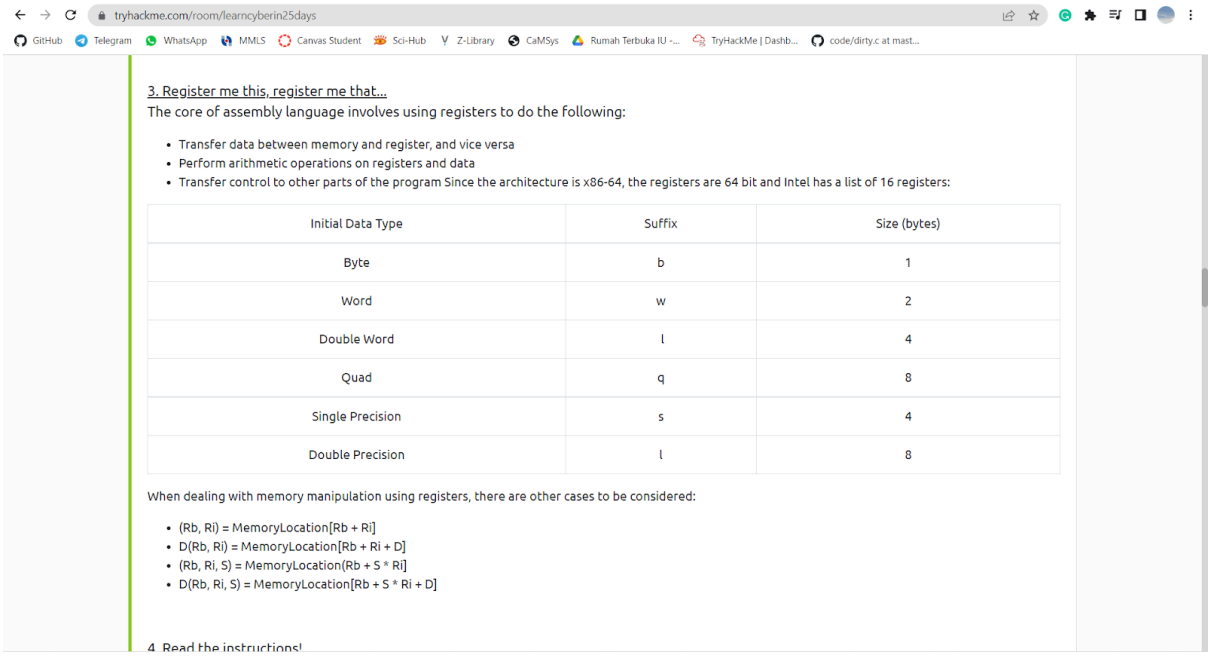
Day 17 : Reverse Engineering - ReverseELFneering

Tools used: Terminal, Google Chrome

Solution / Walkthrough:

Question 1 :

Refer TryHackMe web page on topic 3(Register me this, register me that…) to match data type in registers with their respective size in bytes



tryhackme.com/room/learncyberin25days

GitHub Telegram WhatsApp MMLS Canvas Student Sci-Hub Z-Library CaMSys Rumah Terbuka IU -... TryHackMe | Dashb... code/dirty.c at mast...

3. Register me this, register me that...

The core of assembly language involves using registers to do the following:

- Transfer data between memory and register, and vice versa
- Perform arithmetic operations on registers and data
- Transfer control to other parts of the program Since the architecture is x86-64, the registers are 64 bit and Intel has a list of 16 registers:

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

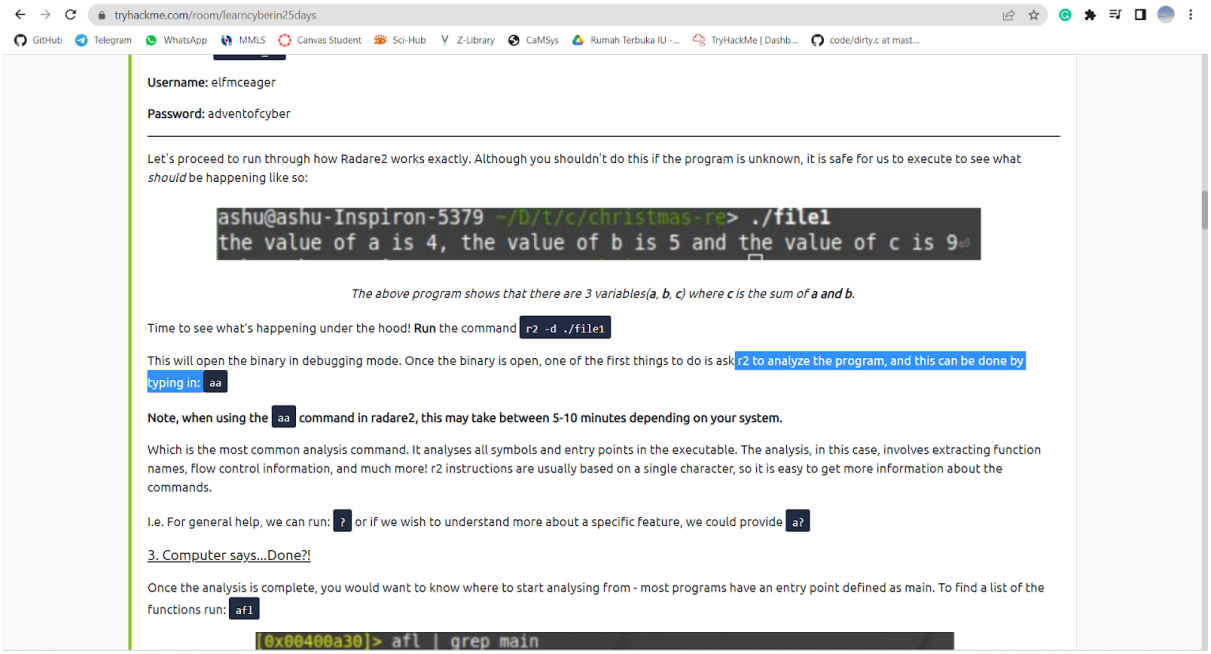
When dealing with memory manipulation using registers, there are other cases to be considered:

- (Rb, Ri) = MemoryLocation[Rb + Ri]
- D(Rb, Ri) = MemoryLocation[Rb + Ri + D]
- (Rb, Ri, S) = MemoryLocation(Rb + S * Ri]
- D(Rb, Ri, S) = MemoryLocation[Rb + S * Ri + D]

4. Read the instructions!

Question 2:

Browse through TryHackMe website to find command to analyse the program in radare2 which is: aa



Username: elfmceager

Password: adventofcyber

Let's proceed to run through how Radare2 works exactly. Although you shouldn't do this if the program is unknown, it is safe for us to execute to see what *should* be happening like so:

```
ashu@ashu-Inspiron-5379 ~/D/t/c/christmas-re> ./file1
the value of a is 4, the value of b is 5 and the value of c is 9
```

*The above program shows that there are 3 variables(**a**, **b**, **c**) where **c** is the sum of **a and b**.*

Time to see what's happening under the hood! **Run** the command `r2 -d ./file1`

This will open the binary in debugging mode. Once the binary is open, one of the first things to do is ask r2 to analyze the program, and this can be done by typing in: `aa`

Note, when using the `aa` command in radare2, this may take between 5-10 minutes depending on your system.

Which is the most common analysis command. It analyses all symbols and entry points in the executable. The analysis, in this case, involves extracting function names, flow control information, and much more! r2 instructions are usually based on a single character, so it is easy to get more information about the commands.

I.e. For general help, we can run: `?` or if we wish to understand more about a specific feature, we could provide `a?`

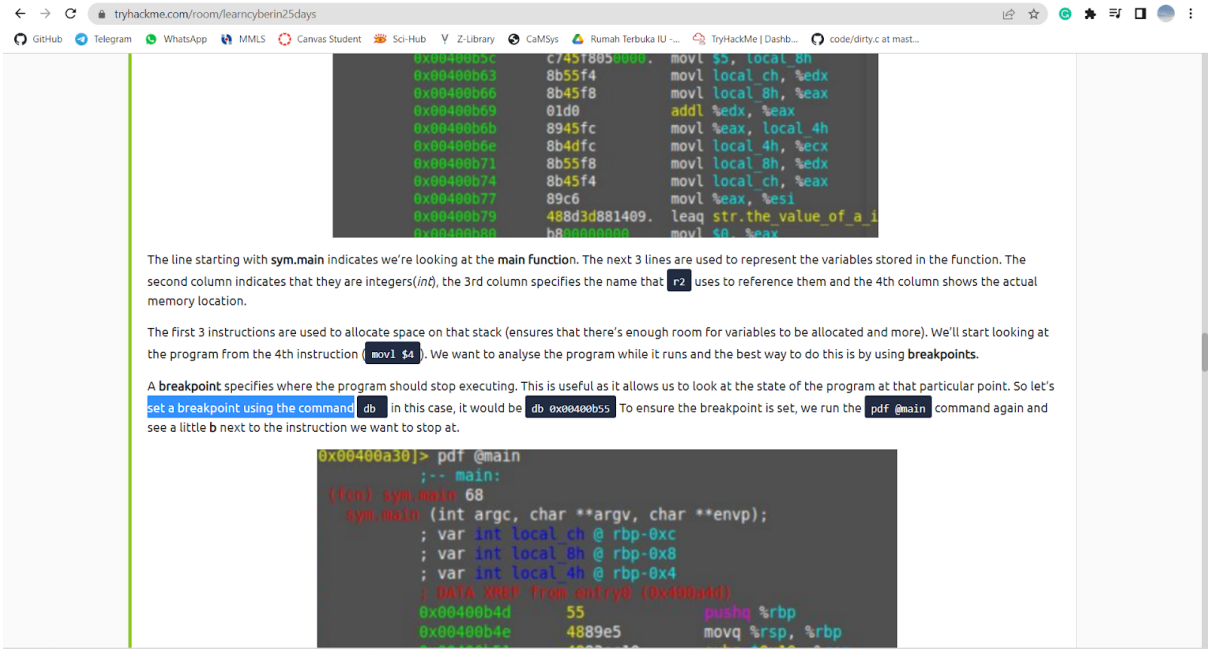
3. Computer says...Done?!

Once the analysis is complete, you would want to know where to start analysing from - most programs have an entry point defined as main. To find a list of the functions run: `afl`

```
[0x00400a30]> afl | grep main
```

Question 3:

Browse further through TryHackMe webpage to find command to set a breakpoint in radare2 which is: db



```
0x00400b63      8b55f4          movl local_ch, %edx
0x00400b66      8b45f8          movl local_8h, %eax
0x00400b69      01d0            addl %edx, %eax
0x00400b6b      8945fc          movl %eax, local_4h
0x00400b6e      8b4dfc          movl local_4h, %ecx
0x00400b71      8b55f8          movl local_8h, %edx
0x00400b74      8b45f4          movl local_ch, %eax
0x00400b77      89c6            movl %eax, %esi
0x00400b79      488d3d881409.   leaq str.the_value_of_a_i
```

The line starting with **sym.main** indicates we're looking at the **main function**. The next 3 lines are used to represent the variables stored in the function. The second column indicates that they are integers(*int*), the 3rd column specifies the name that `r2` uses to reference them and the 4th column shows the actual memory location.

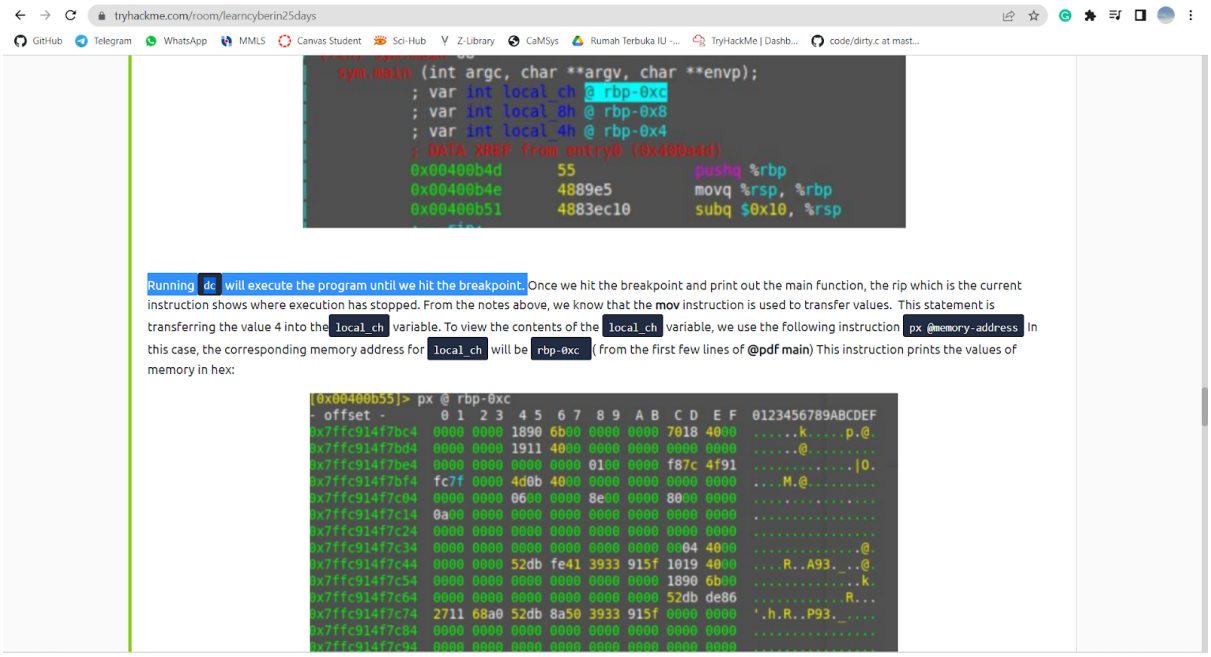
The first 3 instructions are used to allocate space on that stack (ensures that there's enough room for variables to be allocated and more). We'll start looking at the program from the 4th instruction (`movl $4`). We want to analyse the program while it runs and the best way to do this is by using **breakpoints**.

A **breakpoint** specifies where the program should stop executing. This is useful as it allows us to look at the state of the program at that particular point. So let's set a breakpoint using the command `db` in this case, it would be `db 0x00400b55` To ensure the breakpoint is set, we run the `pdf @main` command again and see a little **b** next to the instruction we want to stop at.

```
0x00400a30]> pdf @main
            ;-- main:
(fcn) sym.main 68
  sym.main (int argc, char **argv, char **envp);
            ; var int local_ch @ rbp-0xc
            ; var int local_8h @ rbp-0x8
            ; var int local_4h @ rbp-0x4
            ; DATA XREF from entry0 (0x400a4d)
            0x00400b4d      55              pushq %rbp
            0x00400b4e      4889e5          movq %rsp, %rbp
```

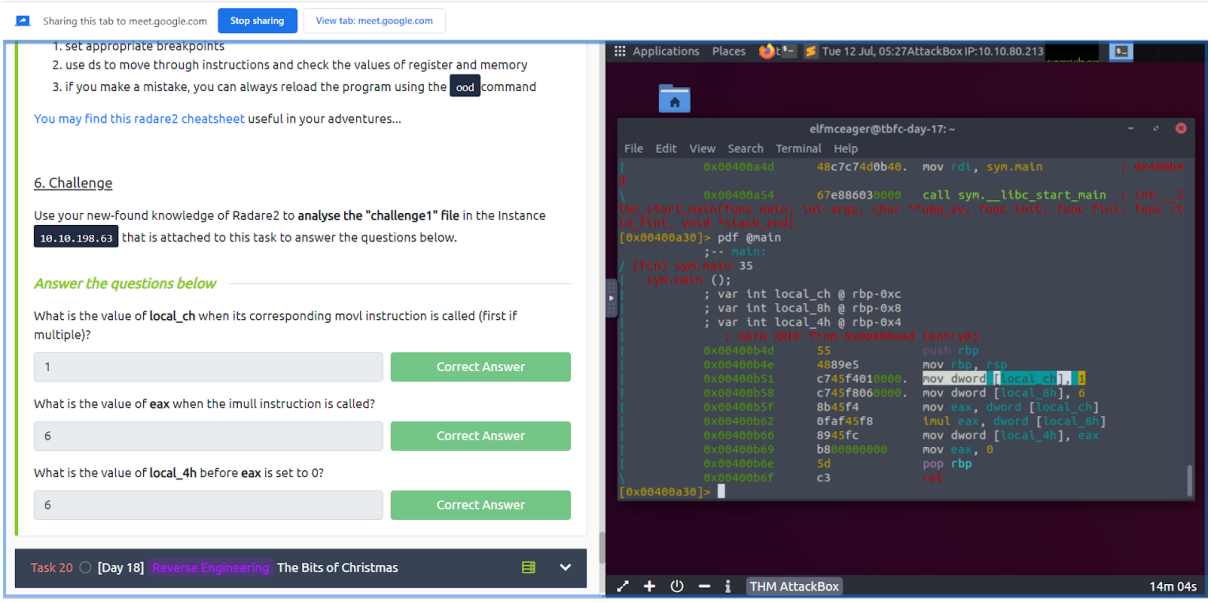
Question 4:

Scroll down to 4.Read the instructions! section in TryHackMe webpage to find commands to execute the program until we hit a breakpoint. The command is dc



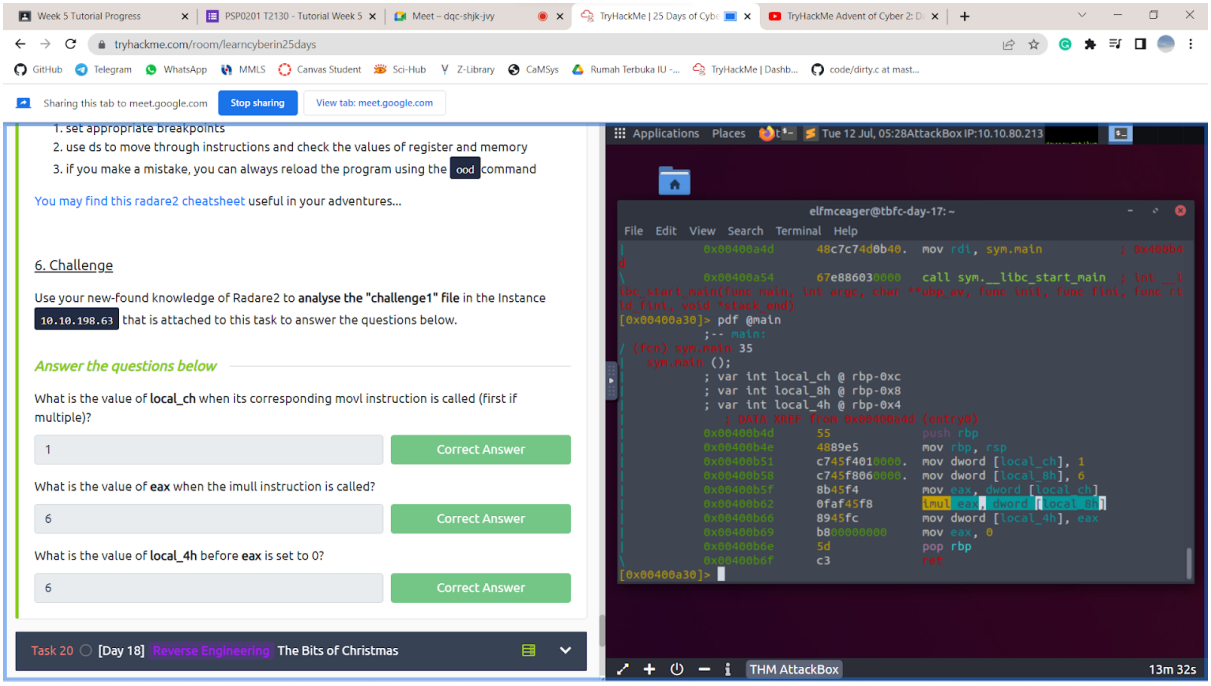
Question 5:

If we run ds command, it will execute mov instruction that will move value 1 into local_ch variable



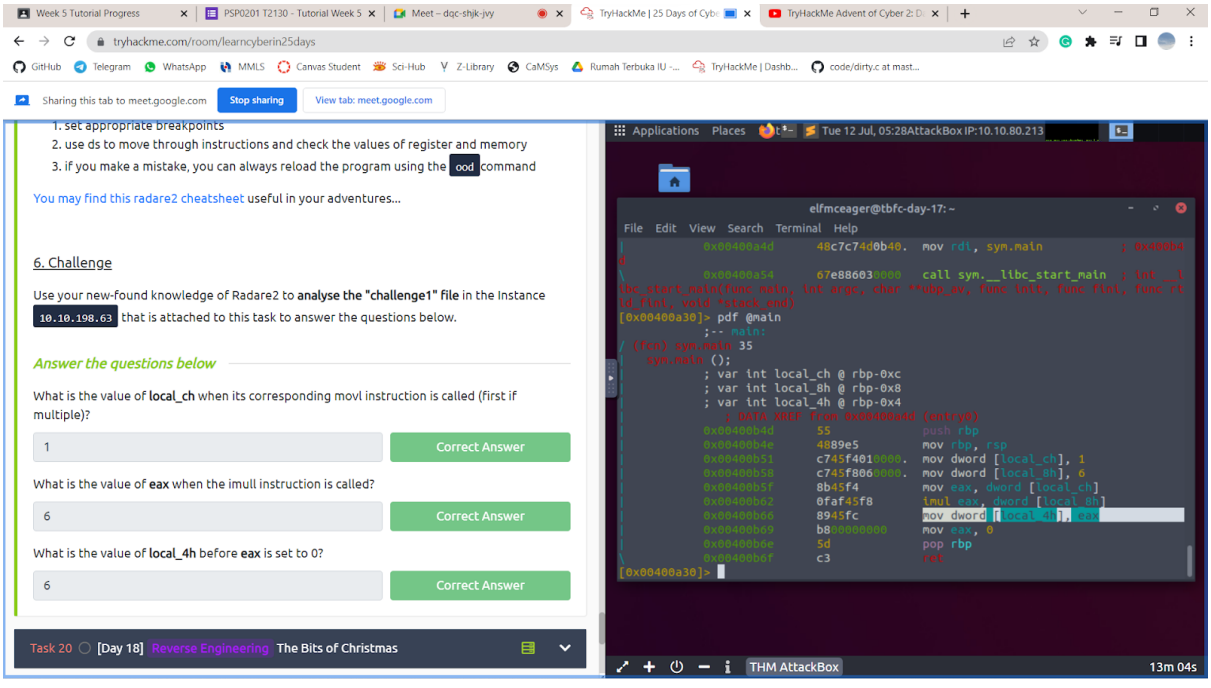
Question 6:

When imull instruction is called, the value inside eax will be multiplied by the value inside dword[local_8h]. At first, value 1 inside dword[local_ch] was moved into eax. After that, the value 1 is multiplied by the value inside dword[local_8h] which is 6. Therefore, the product value is 6



Question 7:

The value 6 inside the eax variable was transferred into dword[local_4h]. Thus, the final value for local_4h before eax is set to 0 is 6.



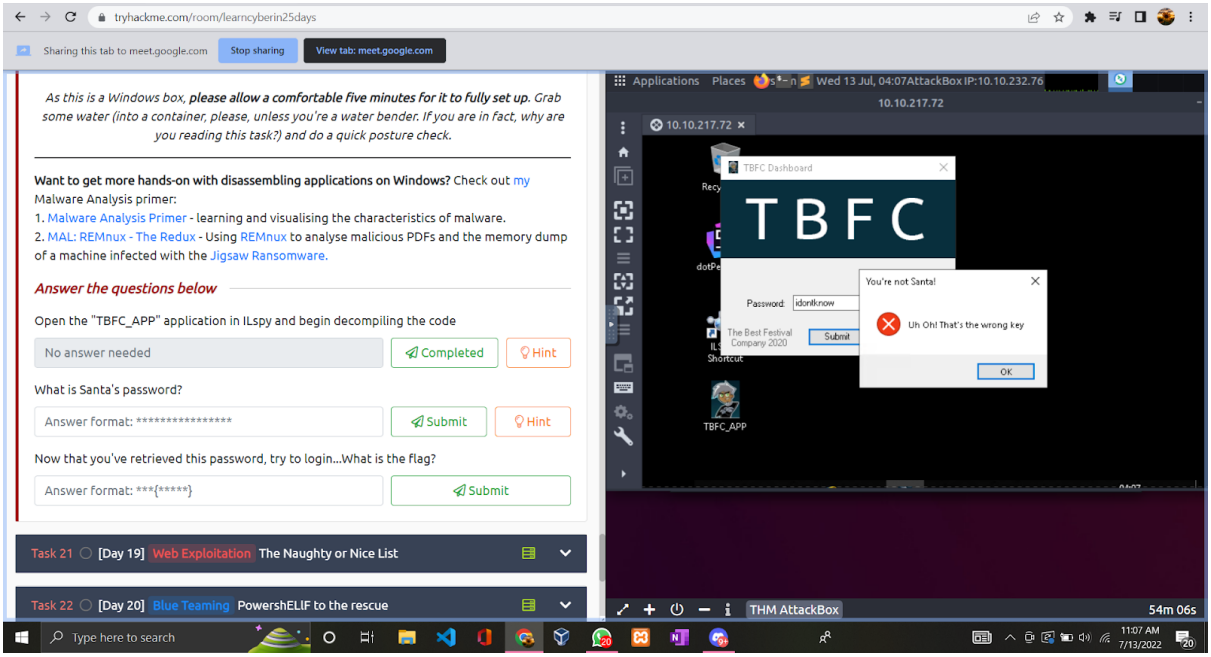
Thought process and methodology:
We need to SSH the target ip address given prior running the binary in debugging mode using radare2. After that, we ask radare2 to analyse the program using aa command and open a list of functions using afl command. Next, we need to find the main function in the list and examine the assembly code inside the main function using command pdf @main. Before executing the instructions, a breakpoint was set to allow us to look at the state of the program at a particular point. Run dc command to execute the program until breakpoint, and px @memory-address command to view the contents in the variables. Run ds command to execute the current instruction that we are at and go to the next one. In Day 17 task, ds command allows us to move a value into the variable stated. We can also run the dr command to view the register variable and check if the values are correct.

Day 18 - [Reverse Engineering] The Bits of Christmas

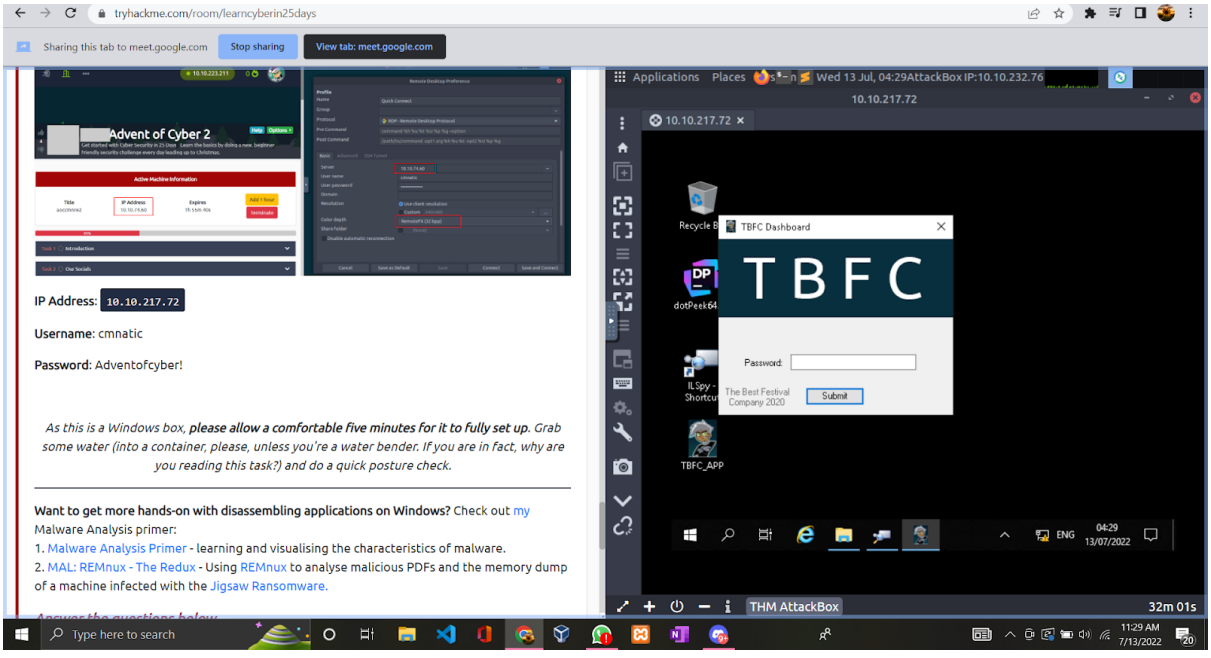
Tools used: Remmina, TBFC_APP, ILSPY Window.

Solution/walkthrough:

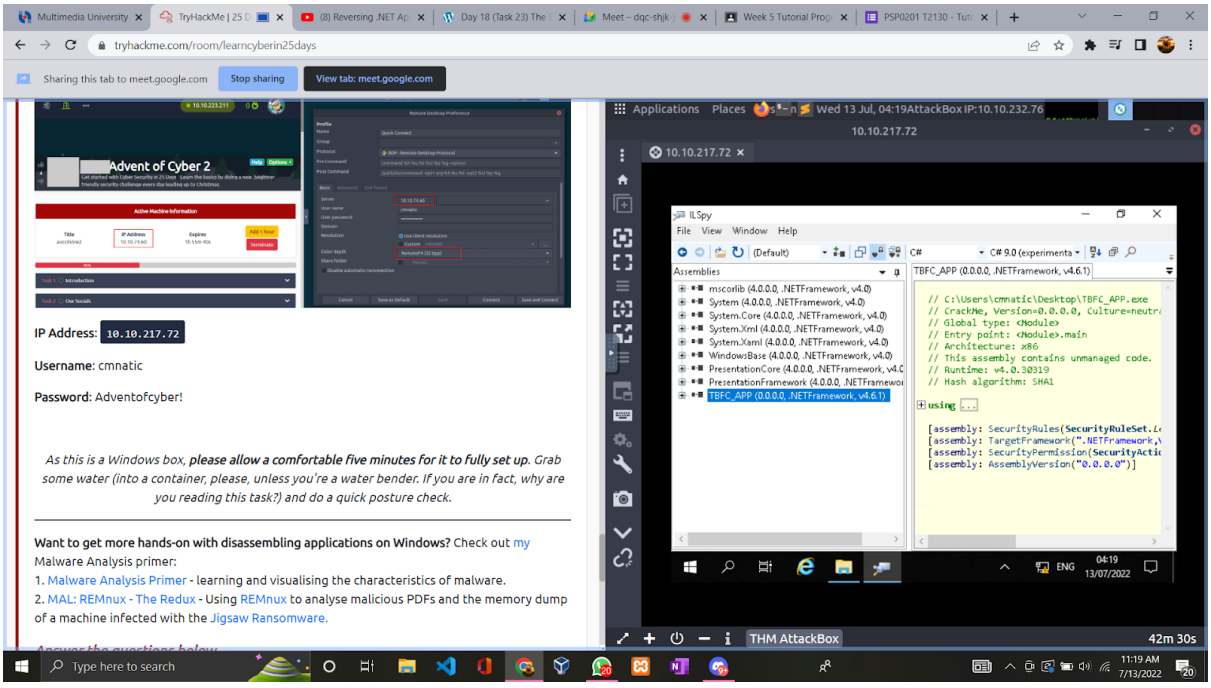
Question 1: open remmina, then enter IP address, hit enter, and accept certificate. Enter the username of cmnatic and the password of Adventofcyber! After logging in, open the TBFC_APP and enter any keys for the password.

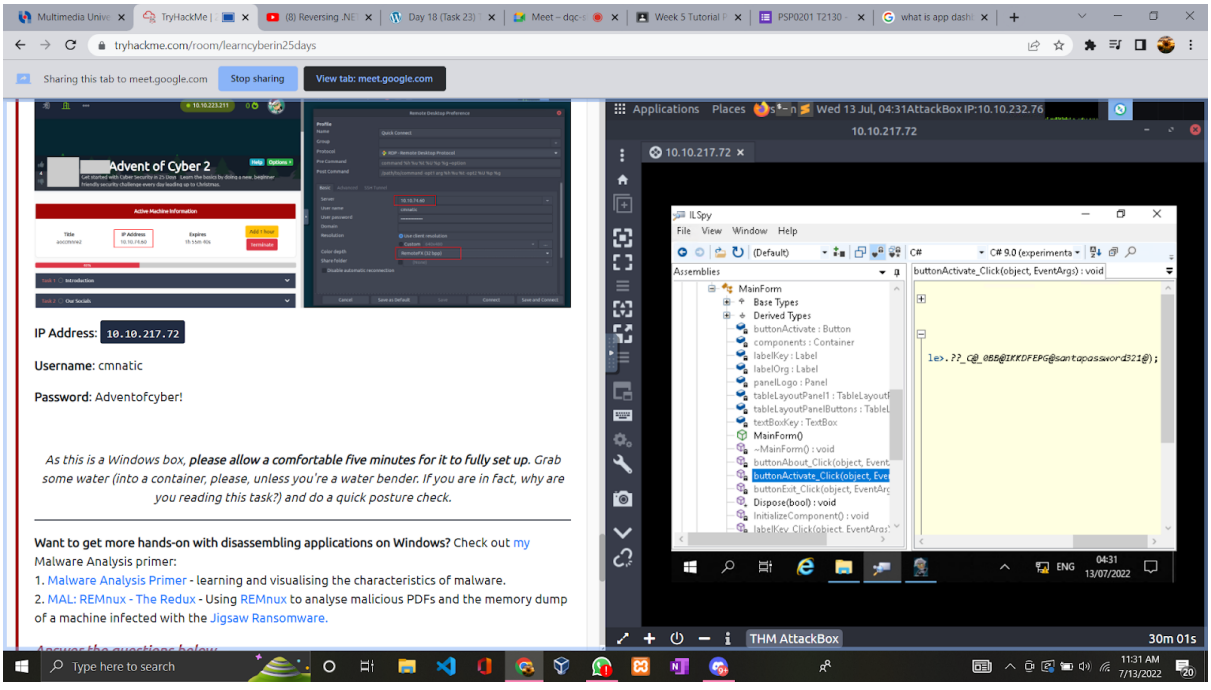


Question 2: when open the TBFC, the dashboard App itself will show the words TBFC stand for.

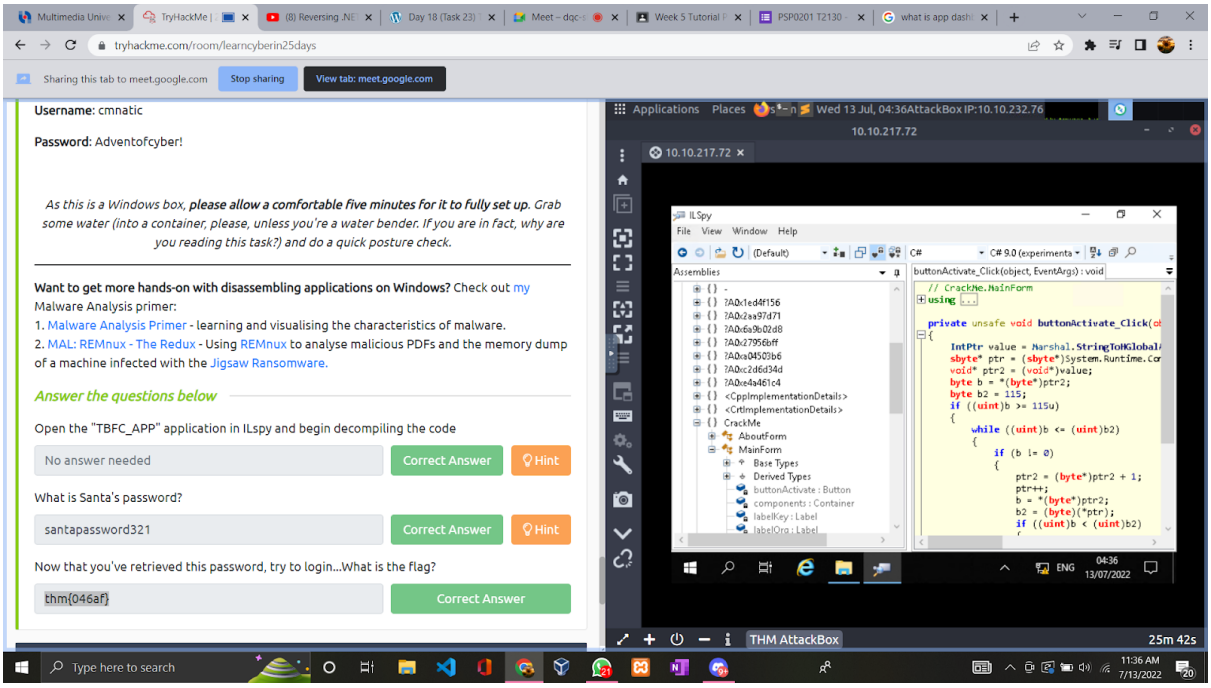


Question 3: Open the ILSpy – Shortcut and then drap the TBFC_APP into the ILSPY Window. The Crackme file, has the password for the TBFC_APP

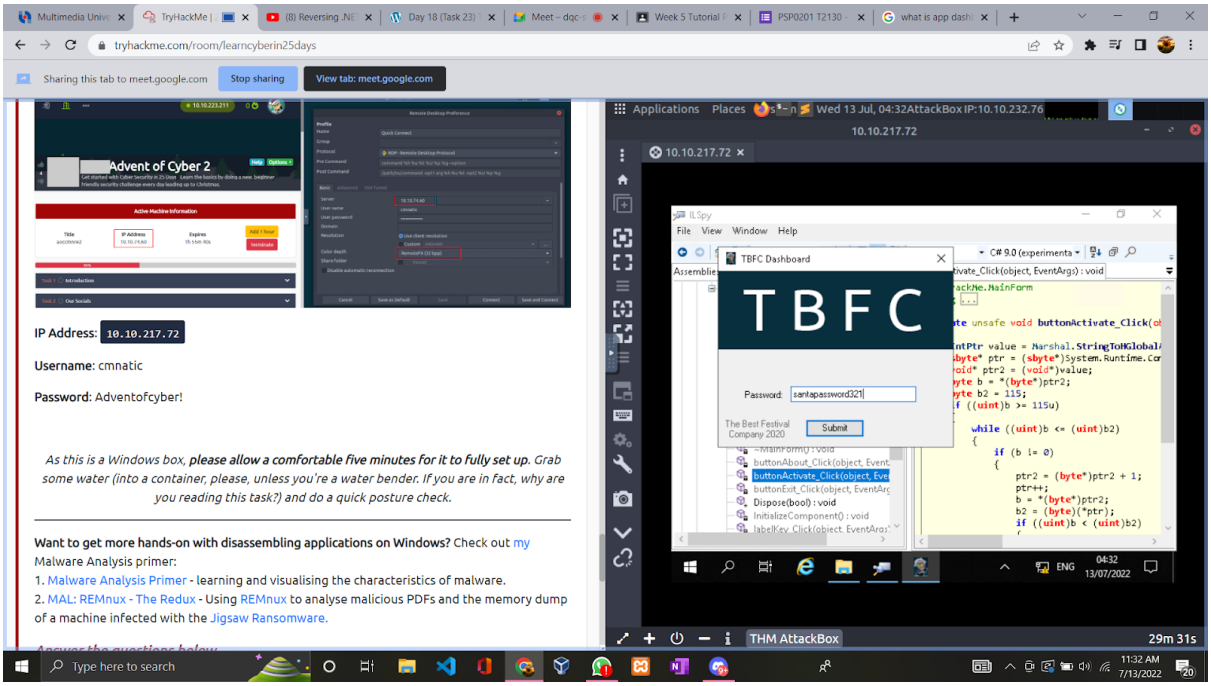




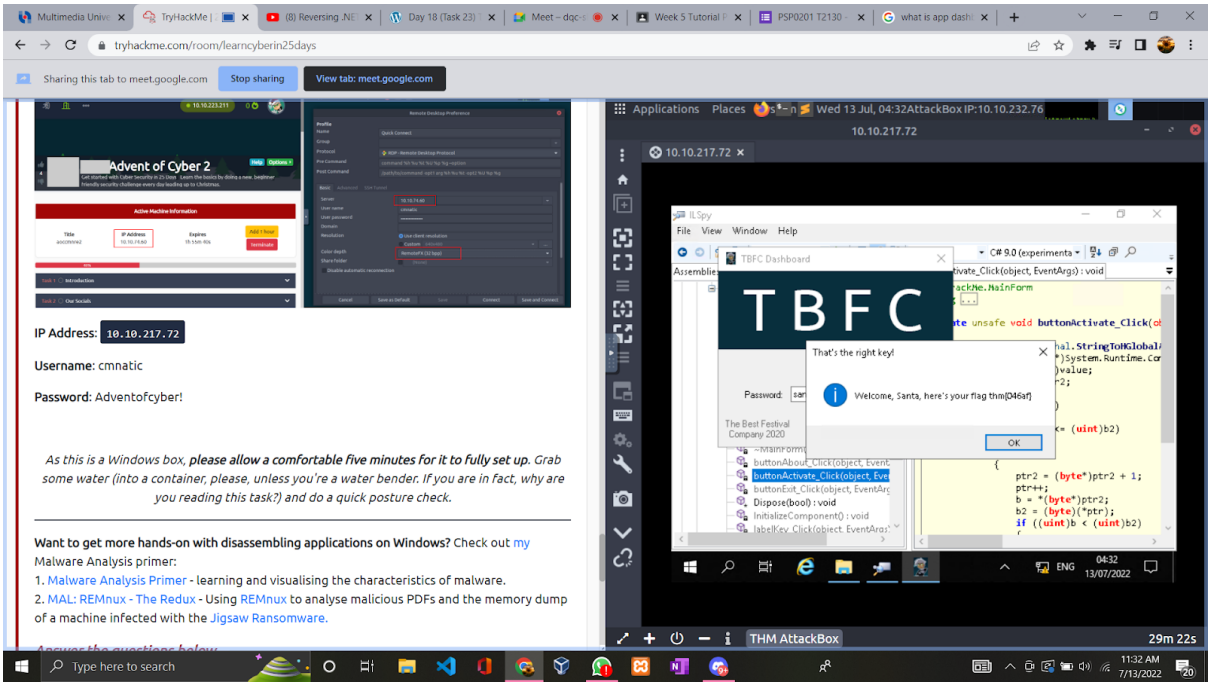
Question 4 and question 5: We can see it appears in MainForm.buttonActivate_Click. That makes sense because when we click the submit button the error message appears. Double click on the MainForm and look at the code.



Question 6: There's a reference to santapassword321, maybe that's the password!



Question 7: After submitting the password, the flag is shown.



Thought Process/Methodology:

The TBFC APP code will be decompiled using ILspy. Finding the password will allow us to access the programme. The flag will be sent to us once we log in. We can use the ILspy and try to find the hidden password in the specific files. Thus, we have to find the files and seek the hidden password in those codes.

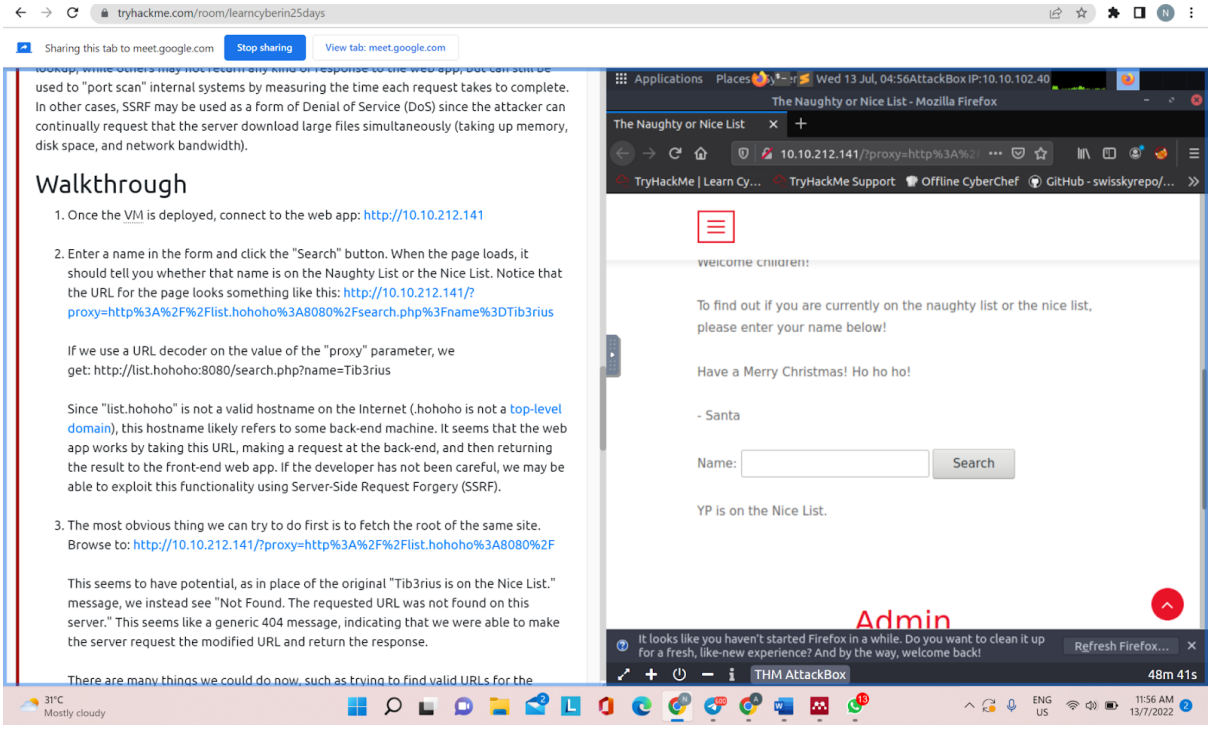
DAY 19 : Web Exploitation- The Naughty or Nice List.

Tools used: Firefox

Solution/Walkthrough:

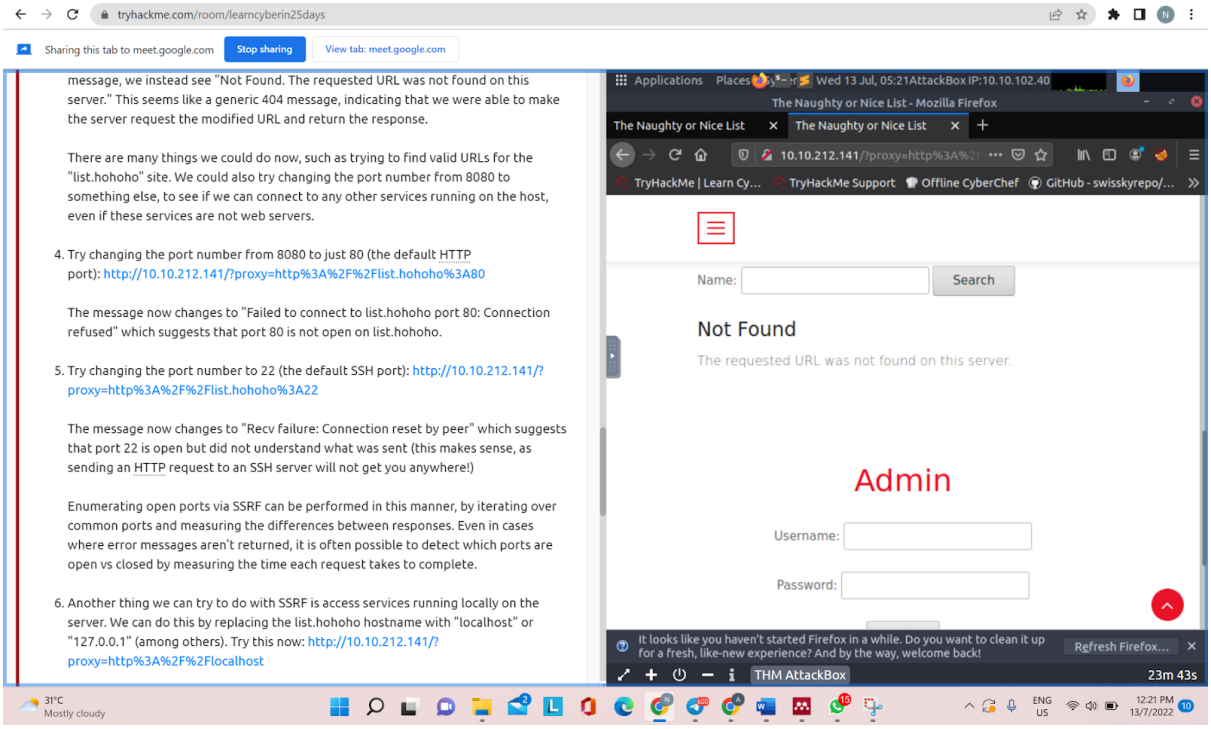
Question 1:

Type the IP address for this task on FireFox. We will be directed to a web page that can be used to look for the naughty and nice kids name. Refer to the google form and enter every name stated in it into the search box.



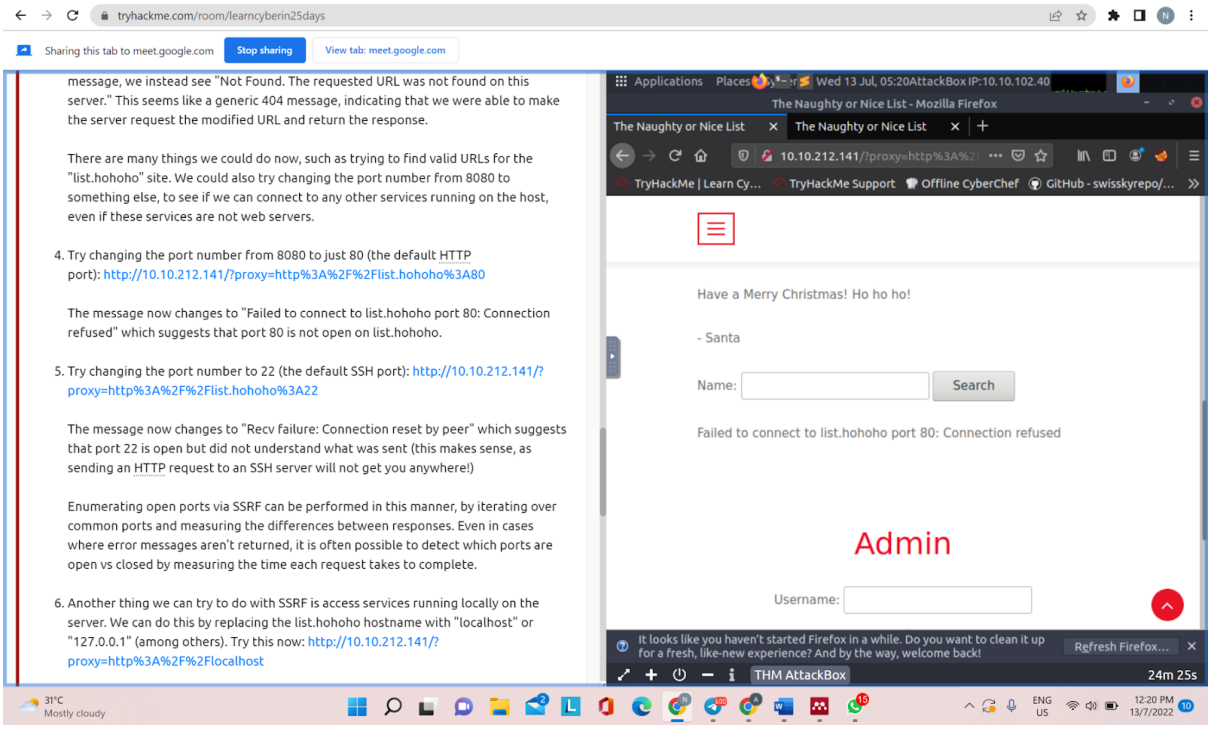
Question 2:

We modify the current URL by fetching the root of the same site. This will return the below message to us. This indicates that we succeeded in requesting the latest URL.



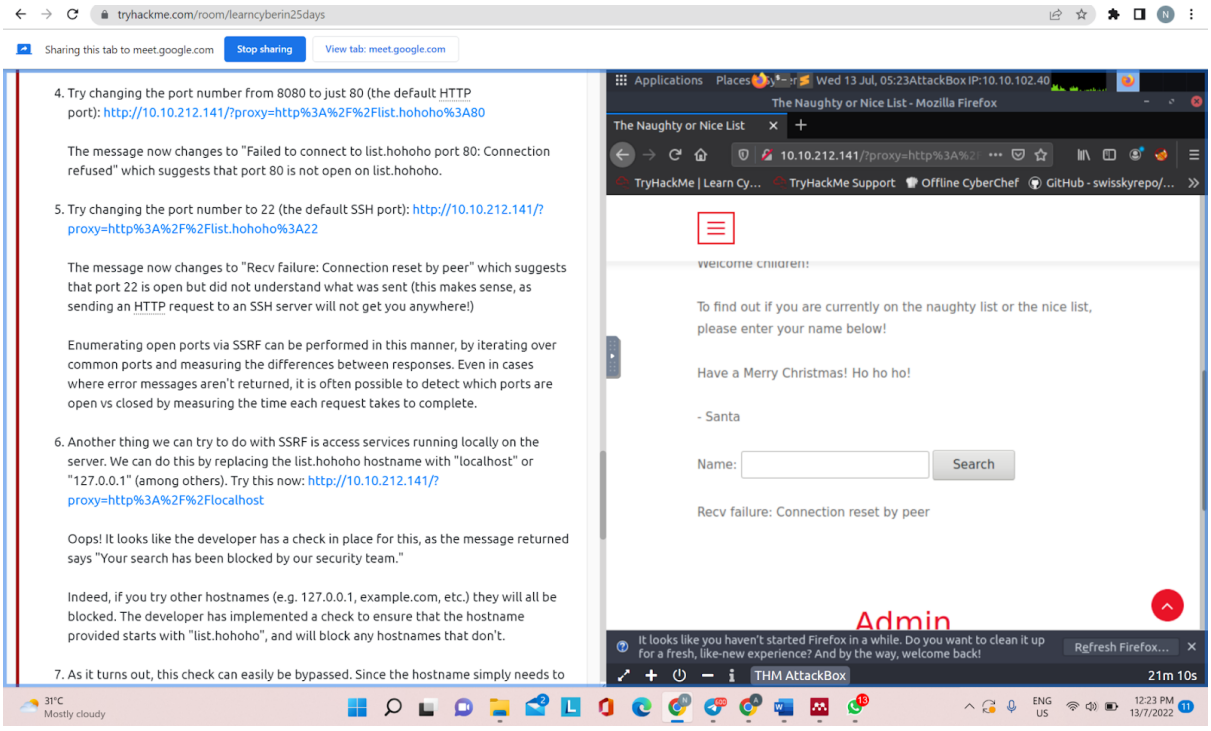
Question 3:

We try to change the port number of the URL from 8080 to 80. The below message will pop up stating that port 80 is not open for list.hohoho.



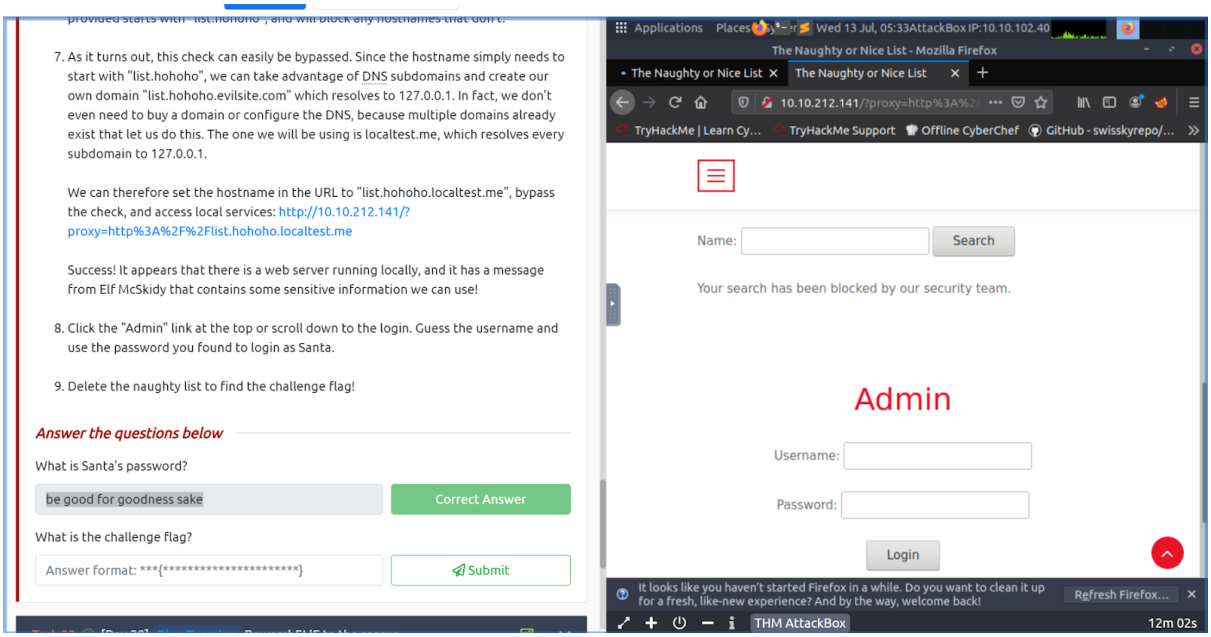
Question 4:

Try changing the port number again by using SSH port,22. It shows that port 22 is open for list.hohoho but since SSH and HTTP are both different protocols, there's nothing we can do about it.



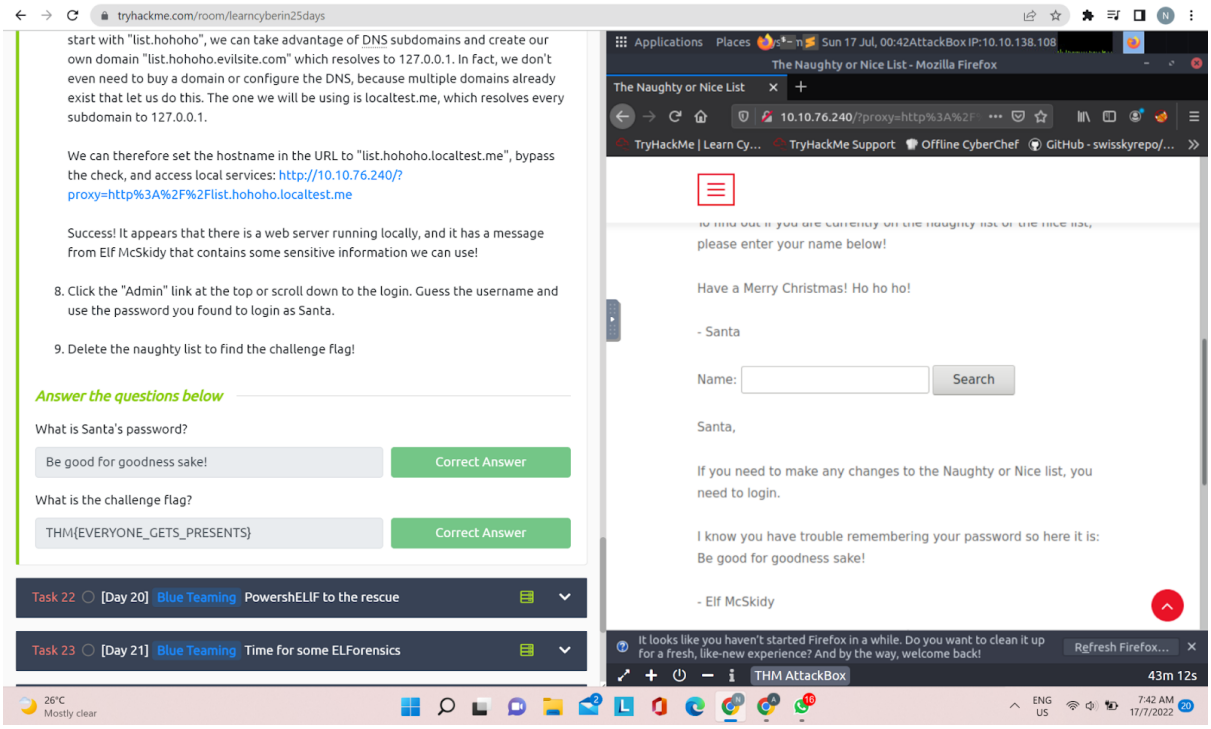
Question 5:

Another thing that we can do is by replacing list.hohoho hostname to localhost. In return, the security team will block our search as they allow hostname "list.hohoho" only.



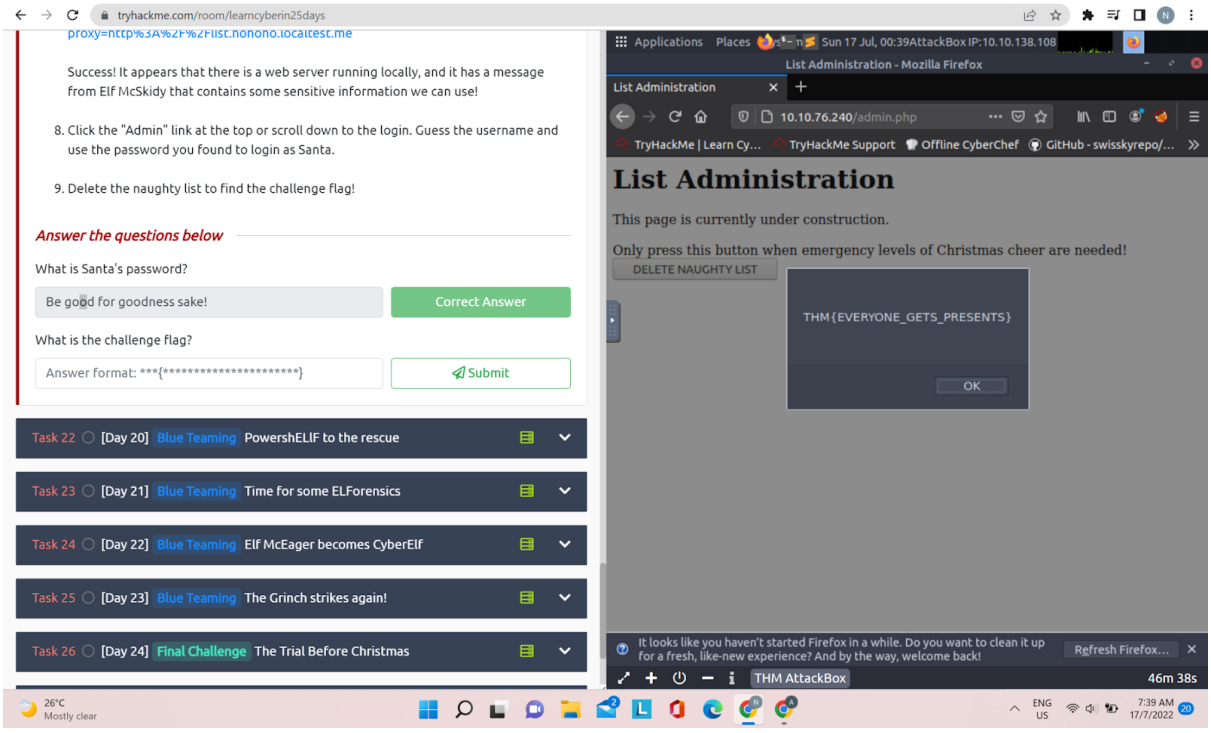
Question 6:

Since we now know that we need to use only list.hohoho hostname, we can create our own domain for example in this task, we use localtest.me to resolve to 127.0.0.1. Add the domain to the hostname and search the current URL. We now found a running web server and we can see Santa's password left by ElfMcskidy.



Question 7:

Login to the admin page using the password we got and username Santa. Click delete the naughty list and now we found the flag!



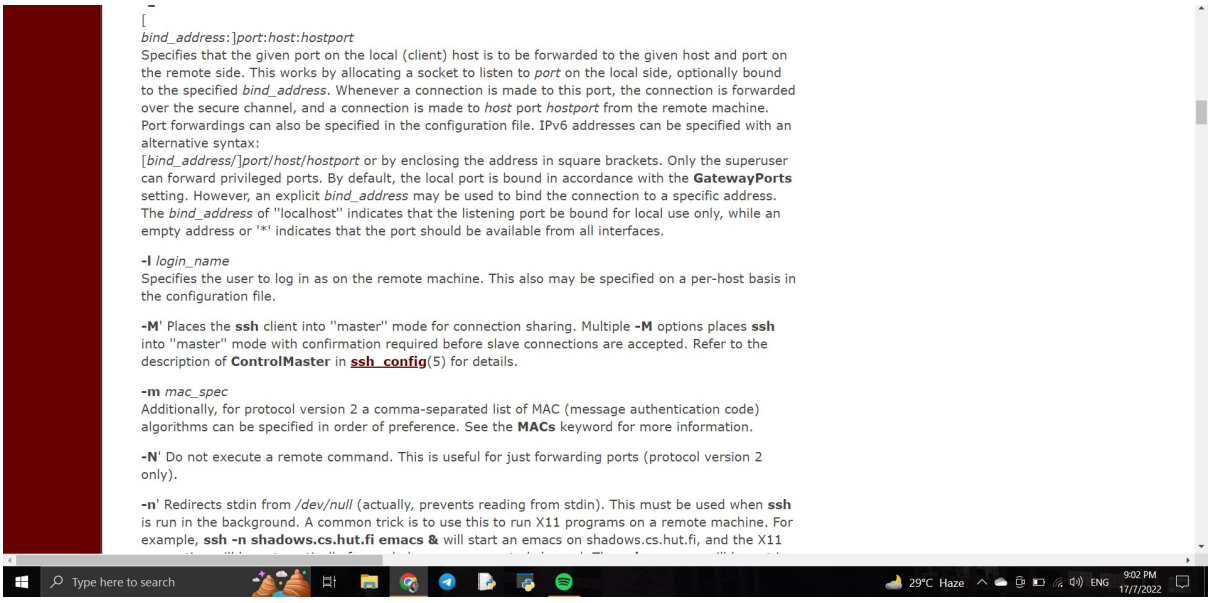
Thought process/Methodology:

In this task, we use Server-Side Request Forgery to force web applications to request resources that would benefit us. Since we want to exploit the web page that contains a list of the nice and naughty kids, we have to figure out the things we can modify to the original URL so that we can access the admin page later on. It could be modifying the port number, domain and whatsoever. Finally, after trying out many changes, we will get the latest modified URL that we can use to see the password to the admin page and login to delete the naughty list. After deleting the list, the flag will be shown.

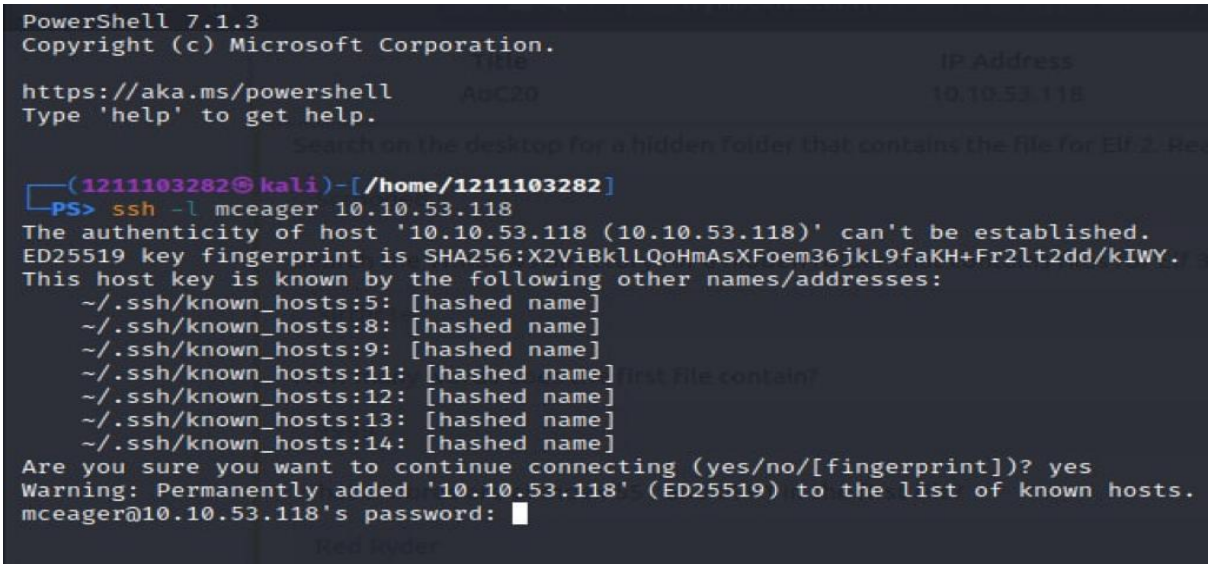
DAY 20 : Blue Teaming - PowershEILF to the rescue
Tools used: kali, powershell

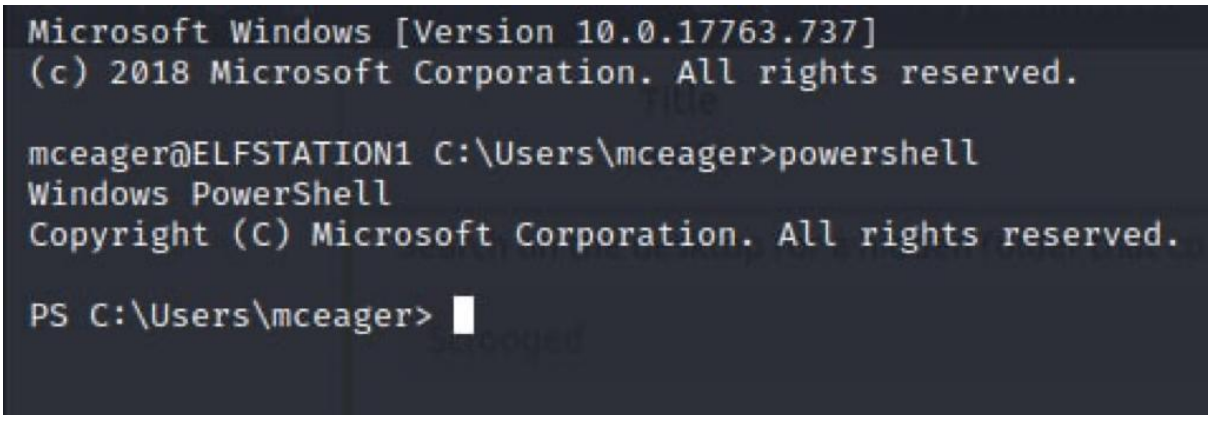
Solution/Walkthrough:

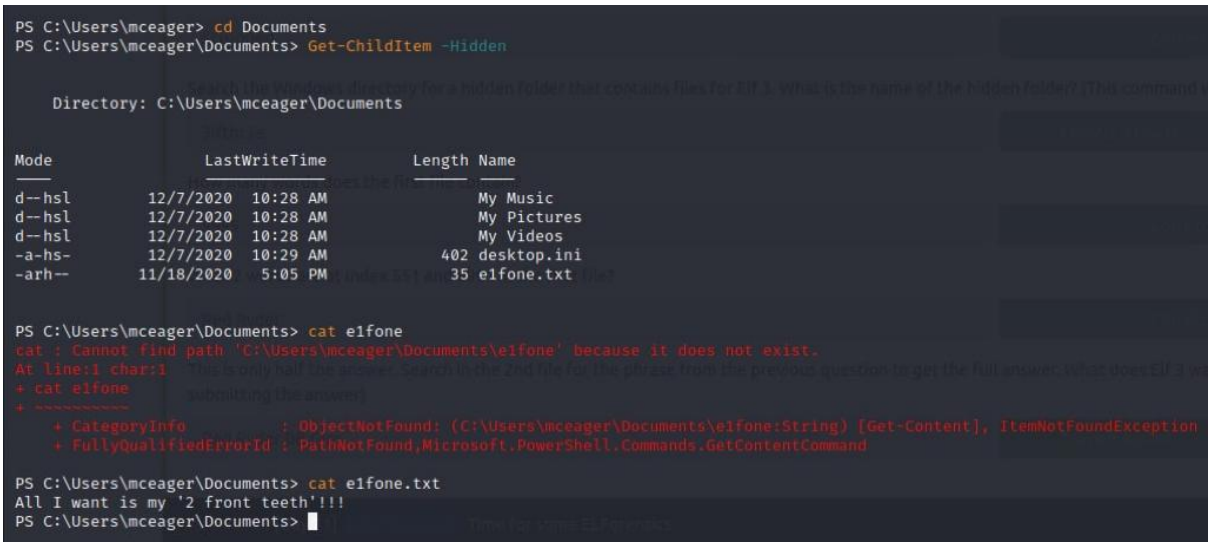
Question 1: as we checked the ssh manual, the -l parameter function has been shown



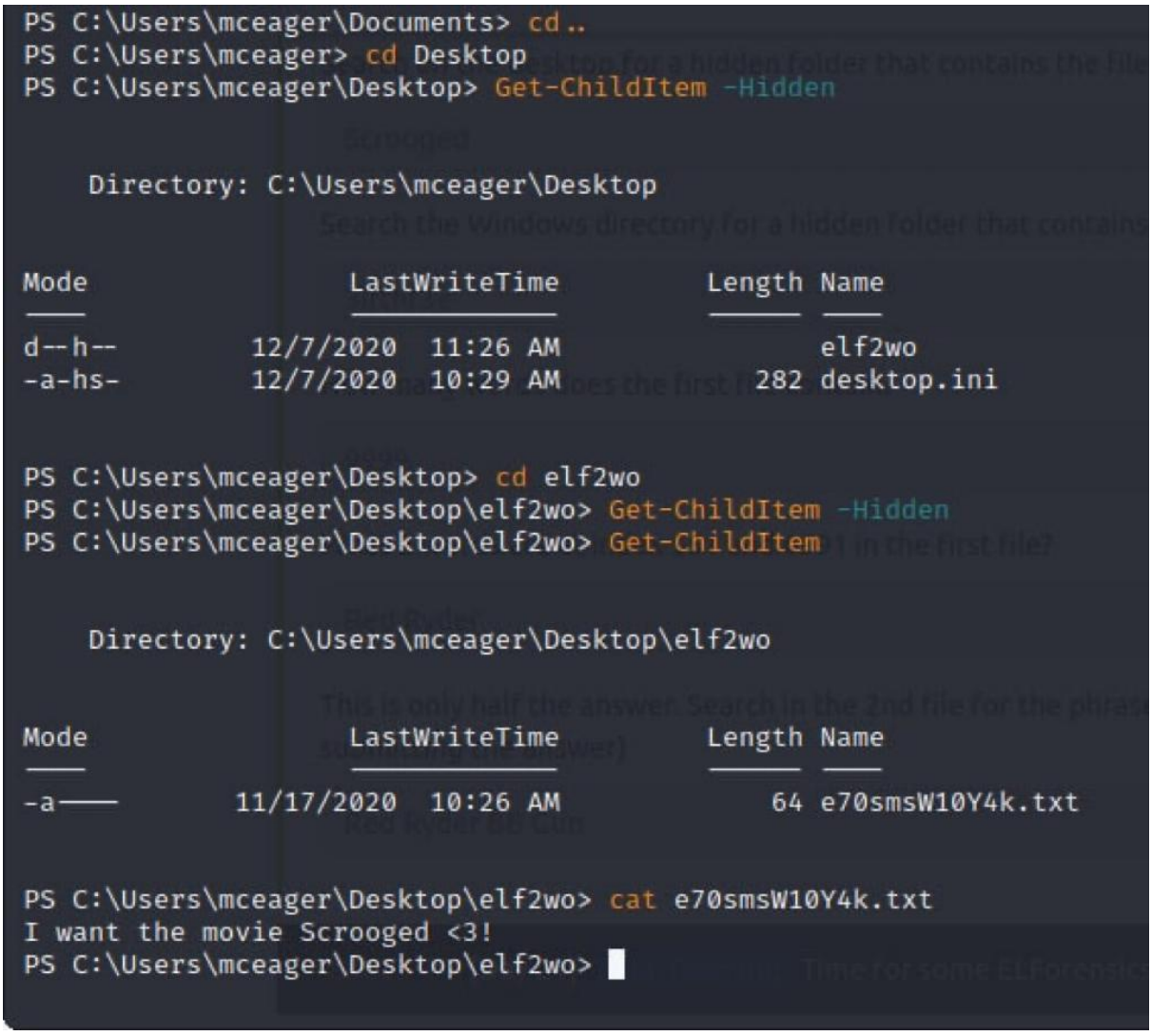
Question 2: the machine has been deployed and then we successfully open the powershell. The cd command has been used to go to the document folder. Then, the Get-Childtm command has been used in order to see the hidden treasure in the folder**.**



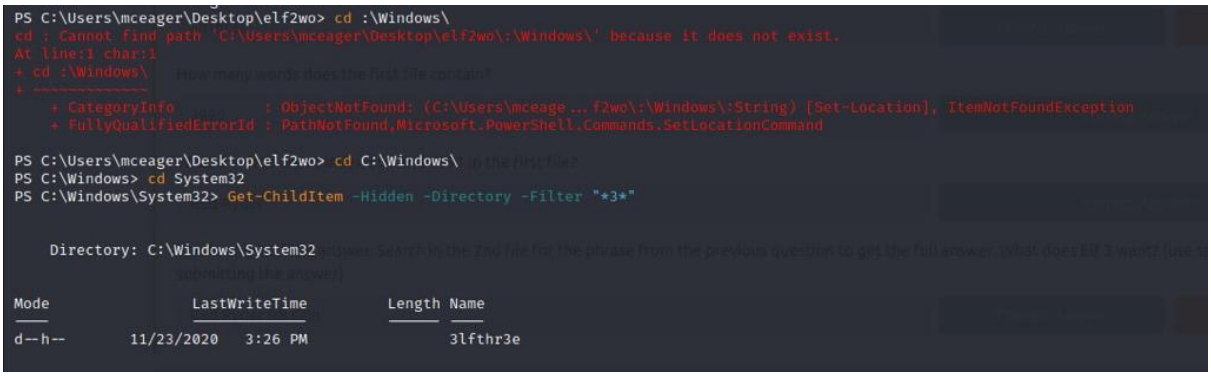




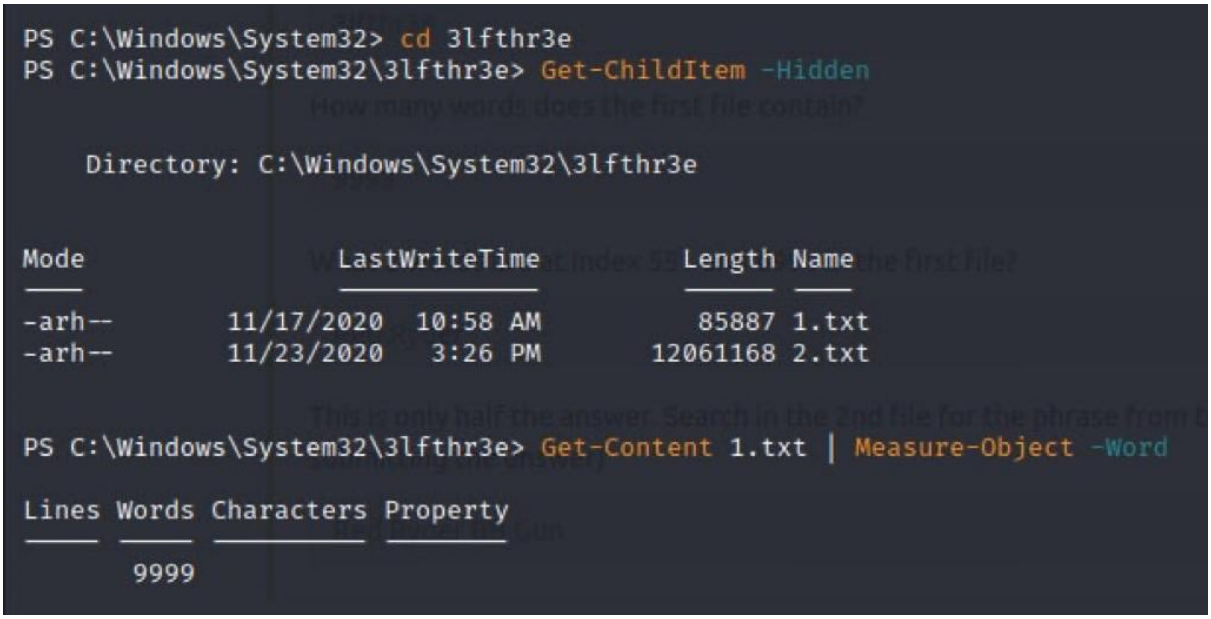
Question 3: the command cd has been used to change the location to the desktop. After that, the Get-Children has been used with the -Hidden parameter to find the hidden folder. A folder named elf2wo has been found using the cd command again, we were in the elf2wo folder. Again, by using the Get-Children command, a file named e70smsW10Y4k.txt has been seen. We opened it using cat command and a file named that elf 2 wants has been seen



Question 4: Then, the directory has been changed to Windows by using the cd command and system32 has been entered by using cd command again. After that, Get-Children command with -Hidden, -Filter "*3*" parameter to find the hidden folder named 3lfthr3e



Question 5: Then, we used the cd command to go to the 3lfthr3e folder and used the Get-Children command with -Hidden parameter to see the files in the folder. After that, we used Get-Content command to see the contents of the first file and pipe the result by using Measure-Object with -Word parameter to see how many word does the first file contain



Question 6: by using the Get-Contact parameter in a bracket to open the first file and using the square brackets to put in the index in order to see the exact position in this file

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

Question 7: last but not least, we used the Get-Content command in order to open the second file and pipe the result using Select-String with -Pattern “redryder” parameter to find what elf 3 wants.

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

redryderbbgun
```

Thoughts Process / Methodology:

First, we launch a terminal and enter our IP address to connect to the host. To start it, we navigate to PowerShell. We go to the Documents directory. Following that, we enter the directory to view all contents. One of the files listed there is called "e1fone.txt." We use the cat command to read the file and display the results. Next, we use the "Get-ChildItem -Directory -Hidden -ErrorAction SilentlyContinue" command to search 'elf2wo. Enter the "elf2wo" directory and list the directories there. It has a file named "e70smsW10Y4k.txt." To display the output, we use the cat command. Next, we go to Windows and run the "Get-ChildItem -Directory -Hidden -ErrorAction SilentlyContinue -Filter 3" command, which lists "3lfthr3e". We enter the directory "3lfthr3e" and list the file there. It contains two files. Using the 'Measure-Object -Word' command, which displayed the words in the first file. Then we use the '(Get-Content.1.txt)[551]' and '(Get-Content.1.txt)[6991]' commands to see the words. Next, we use the 'Select-String -Pattern redryder' command to display the output.