PSP0201

Week 5

Writeup

Group Name: Cylert

Members

ID	Name	Role
1211101022	Ashley Sim Ci Hui	Leader
1211102285	Chin Shuang Ying	Member
1211102398	Nicholas Tiow Kai Bo	Member
1211103427	Law Chin Keat	Member

Day 16: Scripting - Help! Where is Santa?

Tools used: Kali, Python, BPython, FireFox, Nano, Nmap

Solution/Walkthrough:

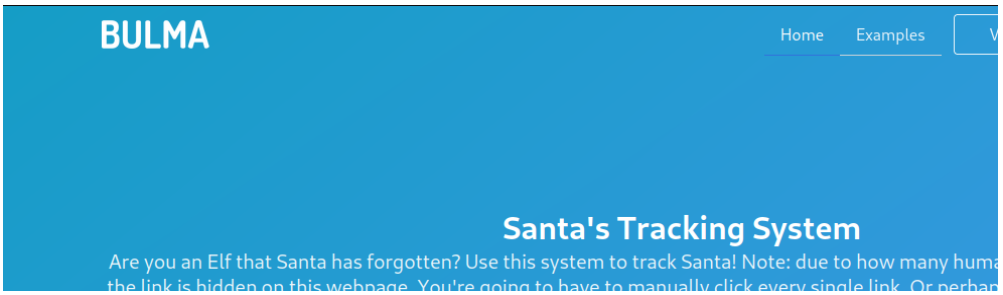
Question 1

Use nmap to scan the port number, and we find that **80** is the port number for the web server.

```
└─$ nmap 10.10.138.190
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-11 21:46 EDT
Nmap scan report for 10.10.138.190
Host is up (0.22s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

Question 2

After browsing the target IP with its port number, as we can see at the top left of the website, we know **BULMA** is the template being used.



Question 3

We first import bs4, which is the library that will be used later.

```
<li><a href="http://machine_ip/api/api_key">Modular modern free</a></li>
└─$ bpython
bpython version 0.22.1 on top of Python 3.10.4 /usr/bin/python3
>>>
>>>
>>> import bs4
>>>
KeyboardInterrupt
```

Then we create and edit a python script, we named it here day16.py.

```
└─$ nano day16.py
```

We utilise the python script created in Day 15, but we need to modify it with the following steps:

1. Change the URL to the target URL together with port number
2. Add .text behind html. Otherwise an error will occur.
3. Change a href to a for searching all the a tag.
4. To only print what we want, which is href, we specify it by following the 4th yellow box.

```
  GNU nano 6.2                                day16.py *
# Import the libraries we downloaded earlier
# if you try importing without installing them, this step will fail
from bs4 import BeautifulSoup
import requests

# replace testurl.com with the url you want to use.
# requests.get downloads the webpage and stores it as a variable
html = requests.get("http://10.10.138.190:80/")

# this parses the webpage into something that beautifulsoup can read over
soup = BeautifulSoup(html.text, "lxml")
# lxml is just the parser for reading the html

# this is the line that grabs all the links # stores all the links in the links variable
links = soup.find_all("a")

for link in links:
    # prints each link
    if "href" in link.attrs:
        print(link["href"])
```

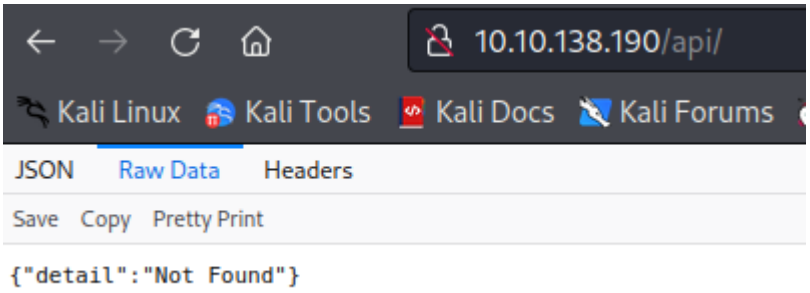
We save and run it together with | uniq to avoid the same output being printed many times. That gives us the directory for the API, which is **/api/**.

```
└─$ python3 day16.py | uniq
../

https://github.com/BulmaTemplates/bulma-templates/blob/master/templates/hero.html
https://tryhackme.com
#
http://machine_ip/api/api_key
#
https://github.com/BulmaTemplates/bulma-templates
```

Question 4

We browse the API of the target IP without the API key in FireFox. This gives us a webpage. We click on the Raw Data tab and the Raw Data returned is **{"detail":"Not Found"}**.



Question 5 & 6

Based on THM, the correct API key is between 0 to 100 and is an odd number. Thus, we can find this by using the loop function in Python. We create another python script named day16_2.py here, and edit it using nano and utilise Python's loop function.

```
└─$ nano day16_2.py
```

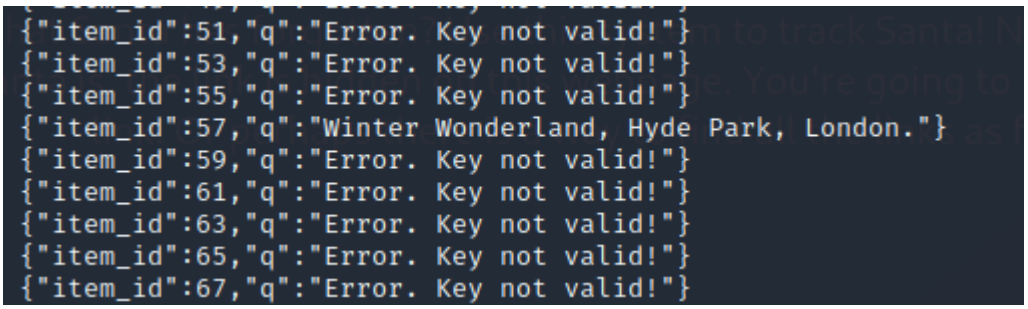
```
  GNU nano 6.2                                  day16_2.py
#!/usr/bin/env python3

import requests

for api_key in range (1,100,2):
    html = requests.get(f'http://10.10.138.190:80/api/{api_key}')
    print(html.text)
```

We run the python script, and it will loop automatically with the odd number (1, 3, 5, … 99). We notice that when the item id is 57, it shows the name of a place which is **Winter Wonderland, Hyde Park, London** rather than an error. With that, we know that **57** is the correct API key.

```
└─$ python3 day16_2.py
{"item_id":1,"q":"Error. Key not valid!"}
{"item_id":3,"q":"Error. Key not valid!"}
{"item_id":5,"q":"Error. Key not valid!"}
{"item_id":7,"q":"Error. Key not valid!"}
{"item_id":9,"q":"Error. Key not valid!"}
{"item_id":11,"q":"Error. Key not valid!"}
{"item_id":13,"q":"Error. Key not valid!"}
{"item_id":15,"q":"Error. Key not valid!"}
{"item_id":17,"q":"Error. Key not valid!"}
{"item_id":19,"q":"Error. Key not valid!"}
```



Thought Process/Methodology:

We first scanned the IP address of the target machine and found that 80 is the port number for the web server. To find the directory for the API, we could utilise the Python script in Day 15 by doing some modifications. After getting the completed URL, we needed the last item which was the API key. Given by THM that the correct API key was between 0 to 100 and was an odd number. We could use the loop function of Python. We needed to create another Python script and used the loop function. When we run it, the python script loops automatically through odd numbers from 1 to 99. We noticed that when the item id was 57, it showed the name of a place rather than an error like others. With that, we knew the correct API key was 57 and Santa is at Winter Wonderland, Hyde Park, London right now.

Day 17: Reverse Engineering - ReverseELFneering

Tools used: AttackBox, Terminal, Radare2, SSH

Solution/Walkthrough:

Question 1

According to the Try Hack Me website, we can find the size in bytes for each respective data type under the **Register me this, register me that…** section.

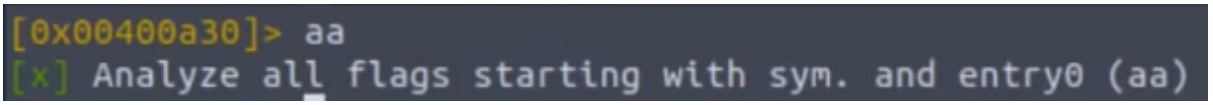
Initial Data Type	Suffix	Size (bytes)
Byte	b	1
Word	w	2
Double Word	l	4
Quad	q	8
Single Precision	s	4
Double Precision	l	8

Question 2

Based on the Try Hack Me website, we can see that the command to analyse the program in radare2 is **aa**.

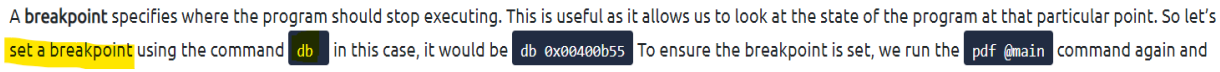
This will open the binary in debugging mode. Once the binary is open, one of the first things to do is ask r2 to analyze the program, and this can be done by typing in: aa

We try to type command **aa** in the terminal, and after a while, it shows that it has finished analysing all flags, symbols and entry points in the program.

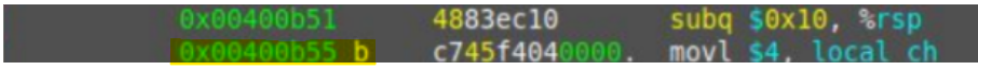


Question 3

We are able to find the command to set a breakpoint in radare2 is **db** based on the Try Hack Me website.

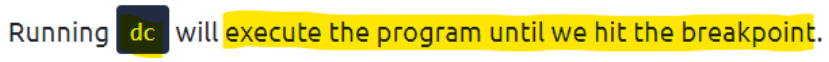


We try to set a breakpoint by using command **db** for a specific instruction we want to stop at in the terminal. Then, we run **pdf @main** command, and we see a **b** next to the instruction, i.e. the instruction we want to stop at.



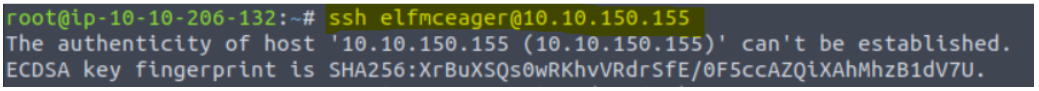
Question 4

While reading through the instructions in the Try Hack Me website, we can find that the command to execute the program until we hit a breakpoint is **dc**.

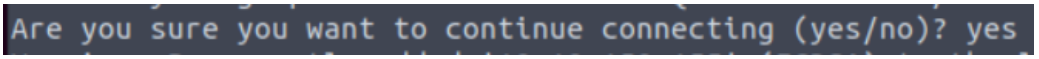


Question 5

Start the AttackBox, type in the command **ssh elfmceager@10.10.150.155** in the terminal.



Then, it will show us a confirmation prompt. Type in **yes** in order to proceed.



Key in the password **adventofcyber**. We have then successfully logged in.

Type in the command **r2 -d ./challenge1** to access the file.

```
elfmceager@tbfc-day-17:~$ r2 -d ./challenge1
Process with PID 1590 started...
= attach 1590 1590
bin.baddr 0x00400000
Using 0x400000
Warning: Cannot initialize dynamic strings
asm.bits 64
[0x00400a30]>
```

Type the command **aa** to analyse the program.

```
[0x00400a30]> aa
[ WARNING : block size exceeding max block size at 0x006ba220
[+] Try changing it with e anal.bb.maxsize
 WARNING : block size exceeding max block size at 0x006bc860
[+] Try changing it with e anal.bb.maxsize
[x] Analyze all flags starting with sym. and entry0 (aa)
```

After the analysing process is done, we type in the command **pdf @main**.

```
[0x00400a30]> pdf @main
```

From the list shown up, we can see that the value of local_ch when its corresponding movl instruction is called (first if multiple) is **1**.

```
        ;-- main:
/ (fcn) sym.main 35
|   sym.main ();
|           ; var int local_ch @ rbp-0xc
|           ; 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
|           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
```

Question 6

Since the value of local_ch when its corresponding movl instruction is called (first if multiple) is **1**, we move the value 1 into **eax** and then multiply with the value of local_8h when its corresponding movl instruction is **6**. In short, we do the operation of **1x6=6**, then we will get the value of eax which is **6** when the imull instruction is called.

```
|           0x00400b58      c745f8060000.   mov dword [local_8h], 6
|           0x00400b5f      8b45f4          mov eax, dword [local_ch]
|           0x00400b62      0faf45f8        imul eax, dword [local_8h]
```

Question 7

Before eax is set to **0**, we can see that **mov dword [local_4h], eax**. This means we just copy the value of eax which is **6** (result from previous question) to local_4h. Therefore, the value of local_4h before eax is set to 0 is **6**.

Thought Process/Methodology:

For Question 1 to 4, we can obtain the answers when reading through the instructions in Try Hack Me website. For Question 1, we can find the answer under the **Register me this, register me that…** section. We can get the answers for Question 2 (**aa**), Question 3 (**db**), and Question 4 (**dc**) in the instructions. For Question 5, we first try to log in to elfmceager's account by using command **ssh elfmceager@10.10.150.155** and the password provided in the instructions which is **adventofcyber**. After login successfully, we use command **aa** to analyse the program first, and then use **pdf @main** to view the list. We observe the list and find that the value of local_ch when its corresponding movl instruction is called (first if multiple) is **1**. We continue observing the list, get that **imul eax, dword [local_8h]**. By doing some mathematical operation which is **1** (move the value of local_ch when its corresponding movl instruction into eax) **x 6** (value of local_8h when its corresponding movl instruction) = **6**. Then, we get the value of eax when the imull instruction is called which is **6** for Question 6. For Question 7, we see that **mov dword [local_4h], eax**. We copy the value of eax which is **6** to local_4h. Finally, we are able to obtain the value of local_4h before eax is set to 0 which is **6**.

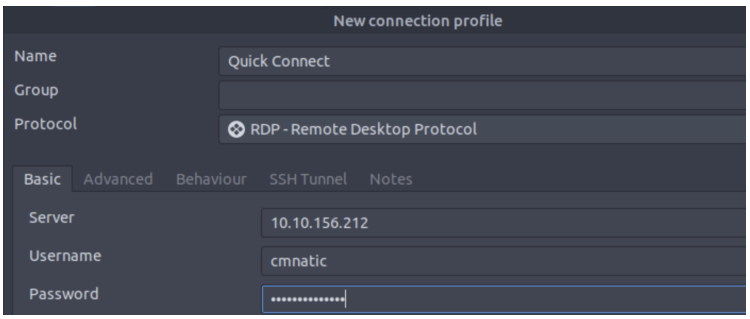
Day 18: Reverse Engineering - The Bits of Christmas

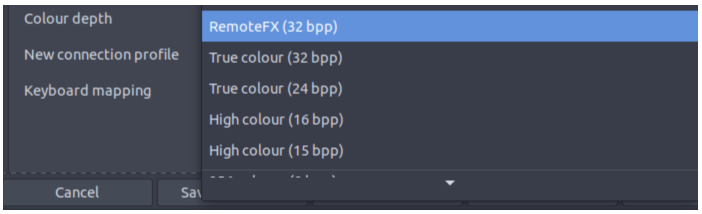
Tools used: Remmina, ILSpy, TBFC_APP

Solution/Walkthrough:

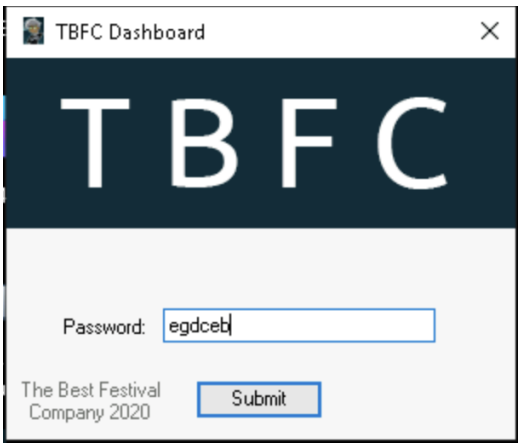
Question 1

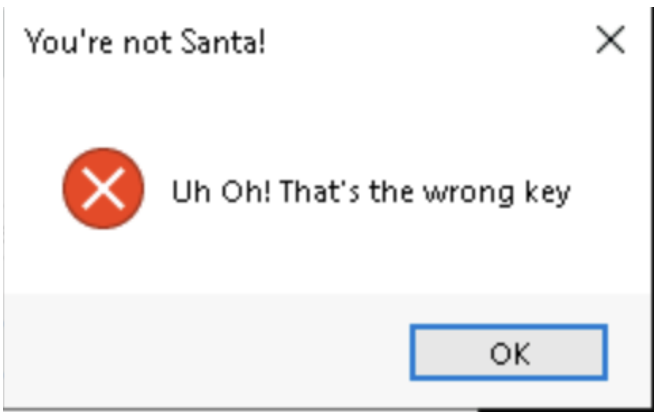
We open up Remmina in the AttackBox to launch a Remote Desktop Protocol and connect to the instance. We simply input the IP address, username and password given by THM, and set the color depth to RemoteFX (32 bpp).





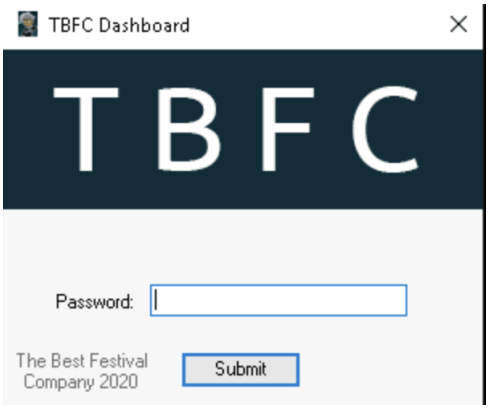
Having successfully logged into the target machine, we run TBFC_APP and input a random incorrect password. This gives us the message "**Uh Oh! That's the wrong key**".





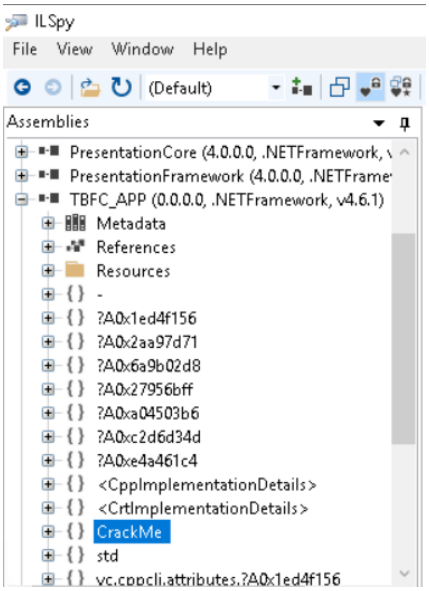
Question 2

According to the TBFC Dashboard, TBFC stands for **The Best Festival Company**.



Question 3

After decompiling TBFC_APP with ILSpy, the module **CrackMe** catches our eye.

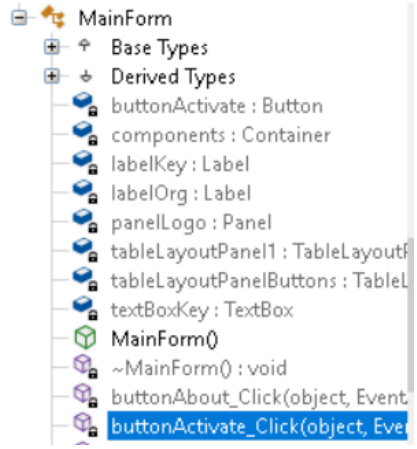


Question 4

Within the module, the form that we are looking for is **MainForm** because it contains the code regarding the login system of TBFC_APP.

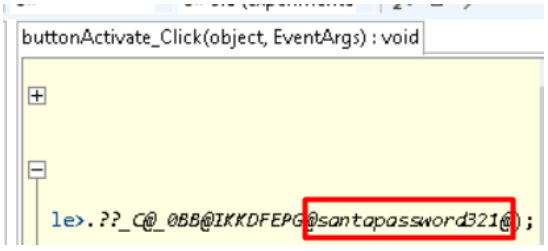
Question 5

The method within the form that we are looking for is **buttonActivate_Click**.



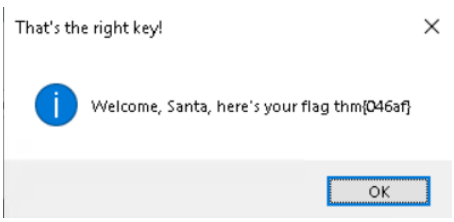
Question 6

After looking through the code in **buttonActivate_Click**, we can see that Santa's password is **santapassword321**.



Question 7

Logging in with Santa's password gives us the flag **thm{046af}**.



Thought Process/Methodology:

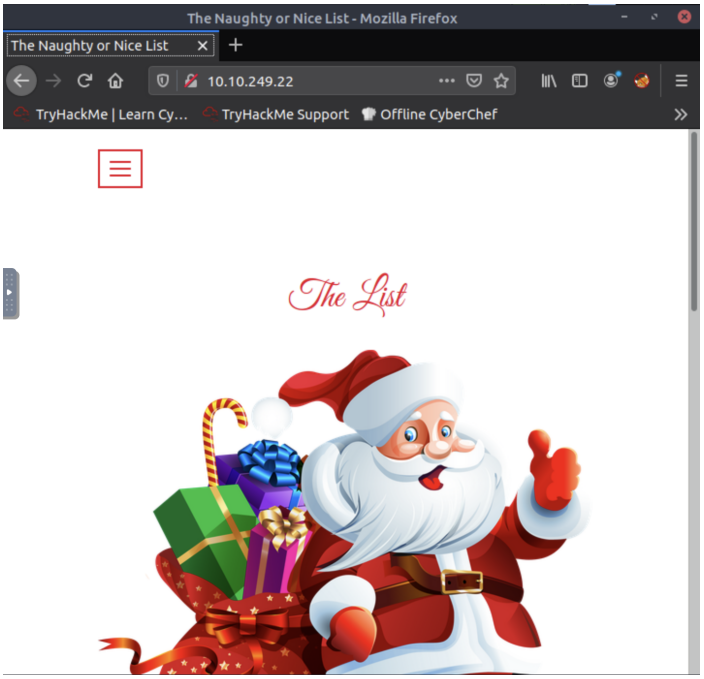
By using Remmina as instructed by THM, we launched a Remote Desktop Protocol (RDP) to connect to the instance. After logging into the target machine, we first opened up TBFC_APP and input an incorrect password which gave us the popup saying "**Uh Oh! That's the wrong key**". The TBFC dashboard also tells us that TBFC stands for **The Best Festival Company** for Question 2. In Question 3, we ran the ILSpy application and decompiled TBFC_APP. Out of all the modules, **CrackMe** is the one that catches our eye. Upon browsing through the forms in that module, the one we are looking for is **MainForm** since it contains the code regarding the login system of TBFC_APP. In Question 5, the form we are looking for is **buttonActivate_Click** since it contains the code regarding what the application does when the Submit button is clicked. After looking through the code in buttonActivate_Click, we find that Santa's password is **santapassword321**. Logging into TBFC_APP using Santa's password gives us the flag **thm{046af}**.

Day 19: Web Exploitation - The Naughty or Nice List

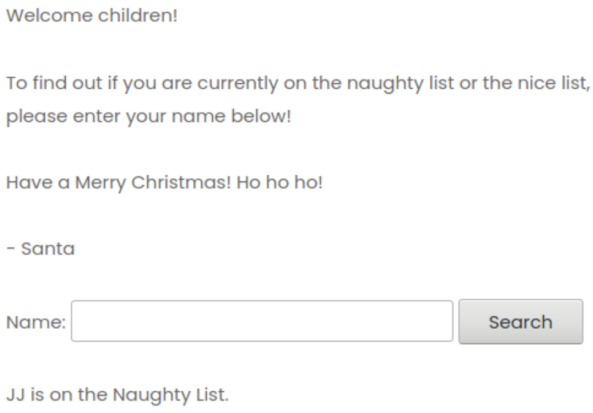
Tools used: AttackBox, FireFox, SSRF

Solution/Walkthrough:

Question 1



We visited The Naughty or Nice List website by entering the target machine IP address into the search box in FireFox.



We entered the name of the person into the search box on the website.

JJ is on the Naughty List.

Kanes is on the Naughty List.

Kanes is on the Naughty List.

Tib3rius is on the Nice List.

Tib3rius is one the Nice List.

Ian Chai is on the Nice List.

Ian Chai is on the Nice List.

Timothy is on the Naughty List.

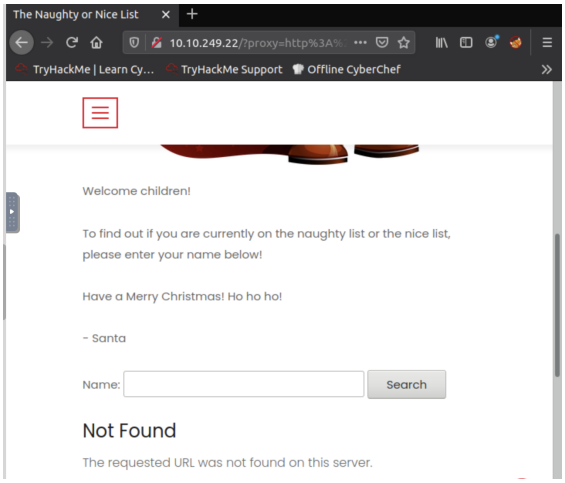
Timothy is on the Naughty List.

YP is on the Nice List.

YP is on the Nice List.

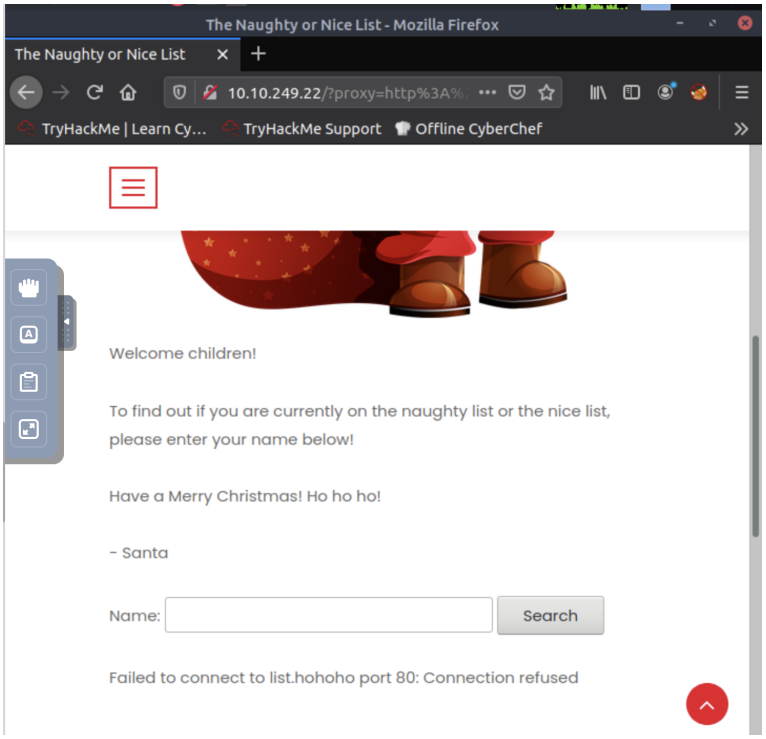
Question 2

When we used "/?proxy=http%3A%2F%2Flist.hohoho%3A8080%2F", the message "**Not Found. The requested URL was not found on this server.**" is displayed on the website.



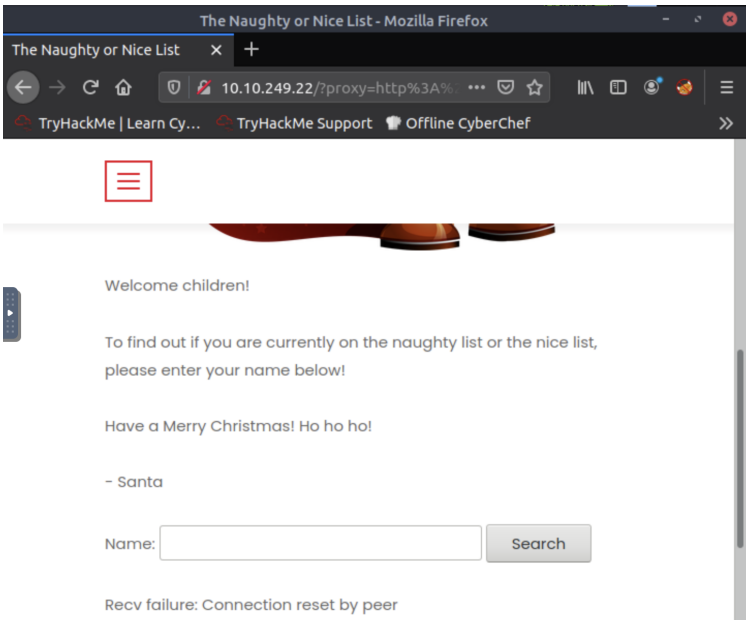
Question 3

When we used "/?proxy=http%3A%2F%2Flist.hohoho%3A80", the message "**Failed to connect to list.hohoho port 80: Connection refused**" is displayed on the website.



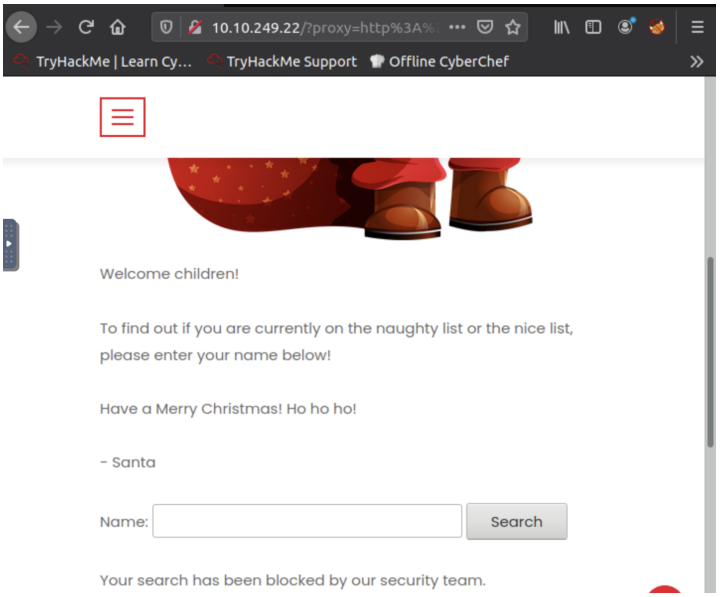
Question 4

When we use "/?proxy=http%3A%2F%2Flist.hohoho%3A22", the message "**Recv failure: Connection reset by peer**" is displayed on the website.



Question 5

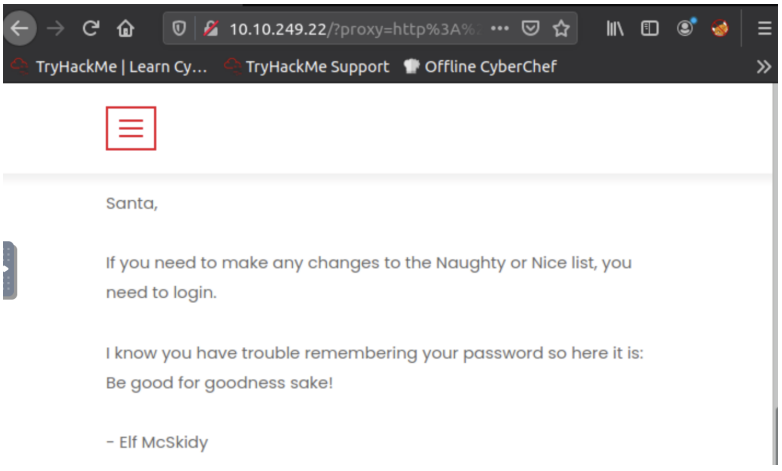
When we used "/?proxy=http%3A%2F%2Flocalhost", the message' “Your **search has been blocked by our security team.**” is displayed on the website.



Question 6

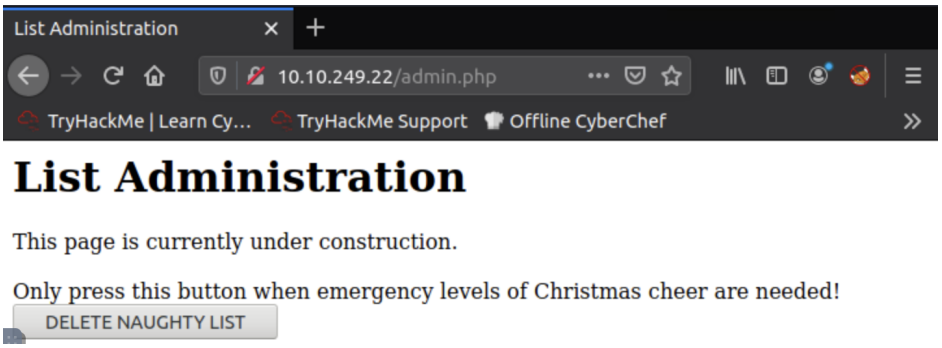
Referring to the walkthrough provided in THM, The developer has implemented a check to ensure that the hostname provided starts with "list.hohoho", and will block any hostnames that don't. We can therefore set the hostname in the URL to "list.hohoho.localtest.me", bypass the check, and access local services. By using “http://10.10.249.22/?proxy=http%3A%2F%2Flist.hohoho.localtest.me”, we successfully received the message left by Elf McSkidy.

Santa's password is “**Be good for goodness sake!**”

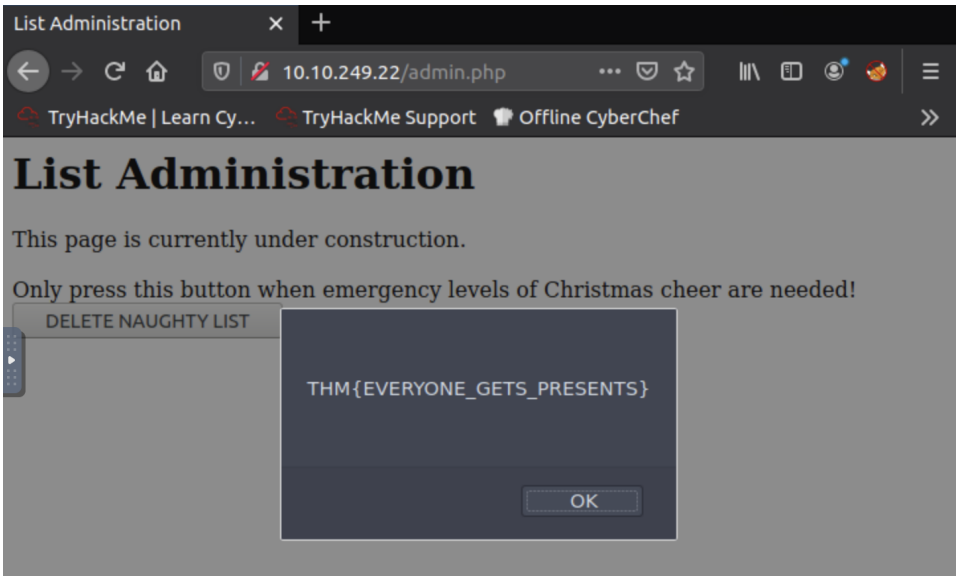


Question 7

We successfully logged in as Santa by using the username “Santa” and the password we got.



We successfully deleted the Naughty List and found our challenge flag which is **THM{EVERYONE_GETS_PRESENTS}.**



Thought Process/Methodology:

We visited The Naugthy or Nice List website by entering the target machine IP into the search box in FireFox. For question 1, we entered the name of every person into the search box on the website and clicked search to see if each of them is on the Nice List or the Naughty List. According to the result, **JJ, Kanes and Timothy are on the Naughty List** and **Tib3rius, Ian Chai and YP are on the Nice List**. Then, we tried to fetch the root of the same site by using "/?proxy=http%3A%2F%2Flist.hohoho%3A8080%2F" but the message "**Not Found. The requested URL was not found on this server."** is displayed on the page. We also tried changing the port number from 8080 to just 80 (the default HTTP port) using "/?proxy=http%3A%2F%2Flist.hohoho%3A80" but the message "**Failed to connect to list.hohoho port 80: Connection refused**" is displayed on the page. Next, we tried changing the port number to 22 (the default SSH port) using "/?proxy=http%3A%2F%2Flist.hohoho%3A22" but the message "**Recv failure: Connection reset by peer**" is displayed on the page. Another thing we tried to do with SSRF is access services running locally on the server. We have done this by replacing the list.hohoho hostname with "localhost" using "/?proxy=http%3A%2F%2Flocalhost", The message "**Your search has been blocked by our security team.**" is displayed on the page. The developer has implemented a check to ensure that the hostname provided starts with "list.hohoho", and will block any hostnames that don't. We can therefore set the hostname in the URL to "list.hohoho.localtest.me", bypass the check, and access local services. By using "http://10.10.249.22/?proxy=http%3A%2F%2Flist.hohoho.localtest.me" , we successfully received the message left by Elf McSkidy. Santa's password is "**Be good for goodness sake!**" By entering the username as "Santa" with the password we got, we successfully logged in as Santa and deleted the Naughty List. We received our challenge flag which is **THM{EVERYONE_GETS_PRESENTS}.**

Day 20: Blue Teaming - PowershELlF to the rescue

Tools used: SSH, Powershell

Solution/Walkthrough:

Question 1

Typing **ssh** into the command prompt provides us with the manual which tells us that the parameter -l is for **login_name**.

```
root@ip-10-10-181-142:~# ssh
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-b bind_address] [-c cipher_spec]
           [-D [bind_address:]port] [-E log_file] [-e escape_char]
           [-F configfile] [-I pkcs11] [-i identity_file]
           [-J [user@]host[:port]] [-L address] [-l login_name] [-m mac_spec]
           [-O ctl_cmd] [-o option] [-p port] [-Q query_option] [-R address]
           [-S ctl_path] [-W host:port] [-w local_tun[:remote_tun]]
           [user@]hostname [command]
```

Question 2

First, we connect to the remote machine using SSH.

```
root@ip-10-10-63-18:~# ssh -l mceager 10.10.150.216
```

After successfully logging in using the password given by THM, we use the commands **Set-Location .\Documents** and **Get-ChildItem -File -Hidden**, we can get a list of hidden files located in the Documents folder. This includes the file **e1fone.txt**, which we then read the contents of using **Get-Content e1fone.txt**. That lets us know that Elf 1 wants his **2 front teeth**.

```
PS C:\Users\mceager> Set-Location .\Documents\
PS C:\Users\mceager\Documents> Get-ChildItem -File -Hidden
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a-hs-        12/7/2020  10:29 AM            402 desktop.ini
-arh--       11/18/2020   5:05 PM             35 e1fone.txt


PS C:\Users\mceager\Documents> Get-Content e1fone.txt
All I want is my '2 front teeth'!!!
```

Question 3

We navigate to the desktop. This time, we get a list of hidden directories using the command **Get-ChildItem -Directory -Hidden**. This shows us the directory **elf2wo**.

```
PS C:\users\mceager\desktop> Set-Location -Path C:\users\mceager\desktop
PS C:\users\mceager\desktop> Get-ChildItem -Directory -Hidden


    Directory: C:\users\mceager\desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d--h--        12/7/2020  11:26 AM                elf2wo
```

Upon navigating to the directory, we get a list of files inside which shows us the file **e70smsW10Y4k.txt**. It tells us that the movie Elf 2 wants is **Scrooged**.

```
PS C:\users\mceager\desktop> Set-Location .\elf2wo\
PS C:\users\mceager\desktop\elf2wo> Get-ChildItem -File -Hidden
PS C:\users\mceager\desktop\elf2wo> Get-ChildItem -File


    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> Get-Content e70smsW10Y4k.txt
I want the movie Scrooged <3!
```

Question 4

As instructed by THM, we navigate to the system32 folder. The hint given by THM tells us to filter our search results using **-Filter '*3*'**, so we use the full command **Get-ChildItem -Directory -Hidden -Filter '*3*'**. This gives us the hidden folder **3lfthr3e**.

Question Hint

Use -Filter '*3*'

```
PS C:\Windows> Set-Location -Path C:\Windows\system32
PS C:\Windows\system32> Get-ChildItem -Directory -Hidden -Filter '*3*'


    Directory: C:\Windows\system32


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

Question 5

After listing out the files in the 3lfthr3e directory, we see two text files. Upon getting the number of words contained in the file 1.txt using the command **Get-Content -Path 1.txt | Measure-Object -Word**, we can see that the number of words in the file is 9999.

```
PS C:\Windows\system32\3lfthr3e> Get-ChildItem -File -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


PS C:\Windows\system32\3lfthr3e> Get-Content -Path 1.txt | Measure-Object

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

Question 6

We use the command **(Get-Content -Path 1.txt)[index]** in which [index] is replaced with the index number of the string we wish to locate. The index numbers 551 and 6991 gives us the words **Red** and **Ryder** respectively.

```
PS C:\Windows\system32\3lfthr3e> (Get-Content -Path 1.txt)[551]
Red
PS C:\Windows\system32\3lfthr3e> (Get-Content -Path 1.txt)[6991]
Ryder
```

Question 7

We search in the second file for any strings containing the words “redryder” using the command **Get-Content -Path 2.txt | Select-String -Pattern ‘redryder’**. This lets us know that Elf 3 wants a **redryderbbgun**.

```
PS C:\Windows\system32\3lfthr3e> Get-Content -Path 2.txt | Select-String -
n 'redryder'

redryderbbgun
```

Thought Process/Methodology:

The SSH manual in Question 1 was easy enough to find by just typing **ssh** into the command prompt. It showed that the parameter -l was for **login name**. For Question 2, we connected to the remote machine using SSH and logged in. We navigated to the Documents folder and listed out all of the hidden files in it, which gave us the file e1fone.txt. By getting its contents, we found out that Elf 1 wants his **2 front teeth**. For Question 3, we navigated to the Desktop instead and got a list of hidden directories, which included the directory **elf2wo** that contained the file **e70smsW10Y4k.txt**. Getting the contents of the file tells us that Elf 2 wants the movie **Scrooged**. Then, in Question 4, we navigated to the system32 directory and searched for a hidden directory with the filter "3" as instructed by THM. This gave us the folder **3lfthr3e** which contained two text files. For Question 5, we got the number of words in the first file by piping our **Get-Content** results to the **Measure-Object** cmdlet. For Question 6, we located the strings we wanted using their index number (551 and 6991) which gave us **Red** and **Ryder**. In Question 7, we filtered our search of the file 2.txt for any strings containing "redryder" and found out that Elf 3 wants a **redryderbbgun**.