

# PSP0201

## Week 5

# Writeup

Group Name: Hack Me No

Members

ID	Name	Role
1211102630	Chan Kar Kin	Leader
1211100925	Ang Jin Nan	Member
1211103311	Ng Yun Shi	Member
1211102777	Tai Qi Tong	Member

## **Day 16: Scripting – Help! Where is Santa?**

**Tools used:** Kali Linux, Firefox

**Solution/Walkthrough:**

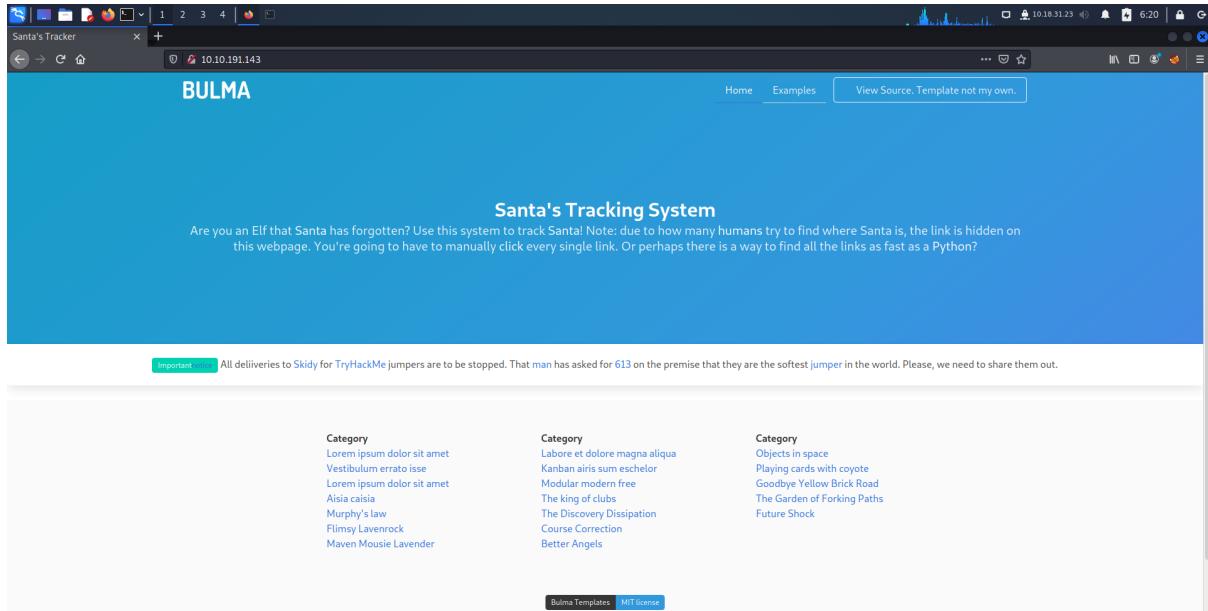
### Question 1

We first start to find the port number using nmap. We found a http port is open, and the port number is **80**.

```
└─(1211100925㉿kali)-[~]
$ nmap -sC -sV 10.10.191.143
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-15 06:15 EDT
Nmap scan report for 10.10.191.143
Host is up (0.32s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 31:4e:6f:1b:9b:4d:a6:9f:34:f0:ca:3e:96:31:a6:9e (RSA)
|   256 60:5d:1b:59:24:8b:b8:7a:5f:1c:75:55:5f:bf:e0:83 (ECDSA)
|_  256 05:08:d8:66:d1:04:cf:91:8c:6a:56:55:df:07:a4:d6 (ED25519)
80/tcp    open  http     uicorn
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.1 404 Not Found
|       date: Fri, 15 Jul 2022 10:16:39 GMT
|       server: uicorn
|       content-length: 22
|       content-type: application/json
|       {"detail":"Not Found"}
|   GetRequest:
|     HTTP/1.1 200 OK
|       date: Fri, 15 Jul 2022 10:16:28 GMT
|       server: uicorn
|       content-type: text/html; charset=utf-8
|       content-length: 7014
|       last-modified: Tue, 29 Dec 2020 00:35:06 GMT
|       etag: fad18236c6876faf561b8ae1bf30c41e
|       <!DOCTYPE html>
|       <html>
|       <head>
|       <meta charset="utf-8">
|       <meta http-equiv="X-UA-Compatible" content="IE=edge">
|       <meta name="viewport" content="width=device-width, initial-scale=1">
|       <title>Santa's Tracker</title>
|       <link rel="shortcut icon" href="" type="image/x-icon">
|       <link rel="stylesheet" type="text/css" href="../static/bulma.css">
|       <!-- Bulma Version 0.9.0 -->
|       <link rel="stylesheet" type="text/css" href="../hero.css">
|       <!-- <link rel="stylesheet" href="https://unpkg.com/bulma-modal-fx/dist/css/modal-fx.min.css" /> -->
|   </head>
|   <body>
|   <section class="hero is-info is-medium is-bold">
|       <HTTPOptions:
|           HTTP/1.1 405 Method Not Allowed
|           date: Fri, 15 Jul 2022 10:16:37 GMT
|           server: uicorn
|           content-length: 31
|           content-type: application/json
|           {"detail":"Method Not Allowed"}
|       <_http-server-header: uicorn
```

## Question 2

After finding the port number, we open our browser and search for our target machine's IP address. A webpage appeared, we found that the templates used by this webpage is called **BULMA** which is located at the top left of the website.



## Question 3

We then proceeded to look at the source code of the main page, and found a URL in the source code. By looking at the URL, we can know that the directory for the API is **/api/**.

```
<div class="column is-3">
  <h2><strong>Category</strong></h2>
  <ul>
    <li><a href="#">Labore et dolore magna aliqua</a></li>
    <li><a href="#">Kanban airis sum eschelor</a></li>
    <li><a href="http://machine_ip/api/api_key">Modular modern free</a></li>
    <li><a href="#">The king of clubs</a></li>
    <li><a href="#">The Discovery Dissipation</a></li>
    <li><a href="#">Course Correction</a></li>
    <li><a href="#">Better Angels</a></li>
  </ul>
</div>
```

#### Question 4

We navigated to the page and tested the page without a API key. A webpage appeared, we then check on the source code and found the raw data which is {"detail":"Not Found"}.

The first screenshot shows a browser window with tabs for "Santa's Tracker", "http://10.10.191.143/", "10.10.191.143/api/", and "http://10.10.191.143/api/". The URL bar shows "10.10.191.143/api/". Below the tabs, there are buttons for "JSON", "Raw Data", and "Headers". Under "Raw Data", the content is {"detail": "Not Found"}. The second screenshot shows a browser window with tabs for "Santa's Tracker", "http://10.10.191.143/", "10.10.191.143/api/", and "http://10.10.191.143/api/". The URL bar shows "view-source:http://10.10.191.143/api/". Below the tabs, the content is {"detail": "Not Found"}.

#### Question 5

By looking at the hint provided from the question, we know that the correct API key is an odd number between 0-100 and Santa's Sled will block our IP address if we tried too many times.

Find out the correct API key. Remember, this is an odd number between 0-100. After too many attempts, Santa's Sled will block you.

To unblock yourself, simply terminate and re-deploy the target instance (MACHINE\_IP)

We opened our terminal and started a new python file called request.py to run through odd numbers appended to the URL and print out if we get the correct API key.

A terminal window titled "Mousepad" with the file path "/home/1211100925/request.py". The window has a menu bar with File, Edit, Search, View, Document, Help. Below the menu is a toolbar with icons for file operations. A red warning bar at the top says "Warning: you are using the root account. You may harm your system." The main text area contains the following Python code:

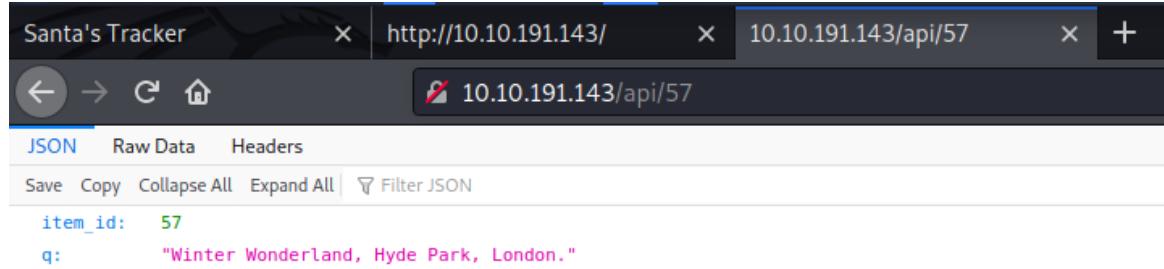
```
1 import requests
2 import json
3
4 for i in range(1,101,2):
5     url = 'http://10.10.112.186/api/{}'.format(i)
6     response = requests.get(url)
7     response_content = json.loads(response.content)
8
9     print(response_content)
10
```

After writing the python file, we start to run the file in our terminal. We then found the correct API key of the webpage and Santa's location which is **Winter Wonderland, Hyde Park, London**.

```
[File Actions Edit View Help]
http://121.111.100.925/Santa's Tracker
(121.111.100.925㉿kali)[-~]corn
(121.111.100.925㉿kali)[-~]spite returning data. If you know the
$ python3 request.py
{'item_id': 1, 'q': 'Error. Key not valid!'}$P=x86_64-pc-linux
{'item_id': 3, 'q': 'Error. Key not valid!'}te:\x20Fri,\x2015\x
{'item_id': 5, 'q': 'Error. Key not valid!'}rn\n\r\ncontent-type:
{'item_id': 7, 'q': 'Error. Key not valid!'}207014\r\nlast-modi
{'item_id': 9, 'q': 'Error. Key not valid!'}T\x\netag:\x20fad18
{'item_id': 11, 'q': 'Error. Key not valid!'}\n\x\nhtml>\n\x20\x20
{'item_id': 13, 'q': 'Error. Key not valid!'}\x20charset=UTF-
{'item_id': 15, 'q': 'Error. Key not valid!'}quiv=\"X-UA-Compatible
{'item_id': 17, 'q': 'Error. Key not valid!'}\x20\x20\x20<meta\x
{'item_id': 19, 'q': 'Error. Key not valid!'}\x20initial-scale=1
{'item_id': 21, 'q': 'Error. Key not valid!'}20Tracker</title>\x
{'item_id': 23, 'q': 'Error. Key not valid!'}ut\x20icon\x20h
{'item_id': 25, 'q': 'Error. Key not valid!'}20\x20\x20\x20<li
{'item_id': 27, 'q': 'Error. Key not valid!'}=\\".\static/bul
{'item_id': 29, 'q': 'Error. Key not valid!'}ulma\x20Version\x2
{'item_id': 31, 'q': 'Error. Key not valid!'}\x20rel="stylesheet"
{'item_id': 33, 'q': 'Error. Key not valid!'}\x20\x20\x20\x20
{'item_id': 35, 'q': 'Error. Key not valid!'}\x20\x20\x20\x20
{'item_id': 37, 'q': 'Error. Key not valid!'}\x20/\x20-->\n
{'item_id': 39, 'q': 'Error. Key not valid!'}\x20\x20\x20\x20\x2
{'item_id': 41, 'q': 'Error. Key not valid!'}diuum\x20is-bold\">
{'item_id': 43, 'q': 'Error. Key not valid!'}K20405\x20Method\x
{'item_id': 45, 'q': 'Error. Key not valid!'}2x2010:30:48\x20G
{'item_id': 47, 'q': 'Error. Key not valid!'}content-type:\x20a
{'item_id': 49, 'q': 'Error. Key not valid!'}Allowed\")\x20(Fo
{'item_id': 51, 'q': 'Error. Key not valid!'}\x20Fri,\x2015\x
{'item_id': 53, 'q': 'Error. Key not valid!'}lvinorn\r\ncontent
{'item_id': 55, 'q': 'Error. Key not valid!'}\x20\r\n\x20\x20\x20
{'item_id': 57, 'q': 'Winter Wonderland, Hyde Park, London.'}
{'item_id': 59, 'q': 'Error. Key not valid!'}ux_kernel
{'item_id': 61, 'q': 'Error. Key not valid!'}
{'item_id': 63, 'q': 'Error. Key not valid!'}y incorrect result
{'item_id': 65, 'q': 'Error. Key not valid!'}n 108.43 seconds
{'item_id': 67, 'q': 'Error. Key not valid!'}
{'item_id': 69, 'q': 'Error. Key not valid!'}
{'item_id': 71, 'q': 'Error. Key not valid!'}
{'item_id': 73, 'q': 'Error. Key not valid!'}
{'item_id': 75, 'q': 'Error. Key not valid!'}
{'item_id': 77, 'q': 'Error. Key not valid!'}se" for more infor
{'item_id': 79, 'q': 'Error. Key not valid!'}
{'item_id': 81, 'q': 'Error. Key not valid!'}
{'item_id': 83, 'q': 'Error. Key not valid!'}
{'item_id': 85, 'q': 'Error. Key not valid!'}
{'item_id': 87, 'q': 'Error. Key not valid!'}format(i)
{'item_id': 89, 'q': 'Error. Key not valid!'}
{'item_id': 91, 'q': 'Error. Key not valid!'}e.content)
{'item_id': 93, 'q': 'Error. Key not valid!'}
{'item_id': 95, 'q': 'Error. Key not valid!'}
{'item_id': 97, 'q': 'Error. Key not valid!'}
{'item_id': 99, 'q': 'Error. Key not valid!'}
```

## Question 6

We navigate to that page with the API key **57**, and different webpage had appeared.



The screenshot shows a browser window with three tabs. The active tab displays a JSON response. The URL bar shows `http://10.10.191.143/` and `10.10.191.143/api/57`. Below the URL bar are buttons for back, forward, refresh, and home. The main content area has tabs for "JSON", "Raw Data", and "Headers". Under "JSON", there is a table with two rows:

item_id:	57
q:	"Winter Wonderland, Hyde Park, London."

Below the table are buttons for "Save", "Copy", "Collapse All", "Expand All", and a "Filter JSON" input field.

## **Thought Process/Methodology:**

To search for the port number of the IP address, we used the nmap scan. A open port has shown, which is port number **80**. After that, we search our attack machine's IP address on our browser and a webpage appeared. We found that the webpage used **BULMA** as their templates. We proceed to look for the source code and found a URL in the source code. By looking it, we know that the directory for API is `/api/`. Next, we tried the URL without typing the API key, and found the raw data of that page which is `{"detail":"Not Found"}`. By looking at the question given from TryHackMe, we know that when we try a different API key for many times, our IP address will be block. The hint given by the question is the correct API key is an odd number between 0-100. After knowing the hint, we open a new python file and write a code into it that could let us know which is the correct API key. After writing it, we start the program and run it in our terminal. The correct API key was shown and we also get to know Santa's location which is **Winter Wonderland, Hyde Park, London**. We then tried the correct API key using our browser and the result was different with other wrong correct API key. From this, we know that **57** is the correct API key.

## **Day 17: Reverse Engineering – ReverseELFneering**

**Tools used:** Kali Linux, Firefox

**Solution/Walkthrough:**

### Question 1

Match the data type with size in bytes based on the notes given in TryHackMe.

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

Read the notes in TryHackMe, it is given that command **aa** is used to analyse program in radare2

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**

### Question 3

TryHackMe notes stated that command to set a breakpoint in radare2 is **db**

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.

### Question 4

Read the TryHackMe notes, command to execute the program until we hit a breakpoint is **dc**

Running **dc** will **execute the program until we hit the breakpoint**. Once we hit the breakpoint and print out the main function, the rip which is the current instruction shows where execution has stopped. From the notes above, we know that the **mov** instruction is used to transfer values. This statement is transferring the value 4 into the **local\_ch** variable. To view the contents of the **local\_ch** variable, we use the following instruction **px @memory-address**. In this case, the corresponding memory address for **local\_ch** will be **rbp-0xc** (from the first few lines of **@pdf main**) This instruction prints the values of memory in hex:

## Question 5

Access to target machine using ssh

```
elfmceager@tbfc-day-17:~  
File Actions Edit View Help  
└──(1211102630㉿kali)-[~]  
$ ssh elfmceager@10.10.89.73  
The authenticity of host '10.10.89.73 (10.10.89.73)' can't be established.  
ED25519 key fingerprint is SHA256:+Yl8Ef3BjQ7HNTMf6qew50LnmiqEXXSzLqgX82k/RSg.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '10.10.89.73' (ED25519) to the list of known hosts.  
elfmceager@10.10.89.73's password:  
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-128-generic x86_64)  
  
* Documentation: https://help.ubuntu.com  
* Management: https://landscape.canonical.com  
* Support: https://ubuntu.com/advantage  
  
System information as of Tue Jul 12 03:44:52 UTC 2022  
  
System load: 0.08 Processes: 90  
Usage of /: 39.4% of 11.75GB Users logged in: 0  
Memory usage: 8% IP address for ens5: 10.10.89.73  
Swap usage: 0%  
  
0 packages can be updated.  
0 updates are security updates.
```

Open binary in debugging mode and ask r2 to analyse the program challenge1

```
elfmceager@tbfc-day-17:~  
File Actions Edit View Help  
  
Last login: Wed Dec 16 18:25:51 2020 from 192.168.190.1  
elfmceager@tbfc-day-17:~$ ls  
challenge1 file1  
elfmceager@tbfc-day-17:~$ r2 -d ./challenge1  
Process with PID 1645 started...  
- attach 1645 1645  
bin.baddr 0x00400000  
Using 0x400000  
Warning: Cannot initialize dynamic strings  
asm.bits 64  
[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)
```

Find list of functions with the entry point called 'main'

```
elfmceager@tbfc-day-17:~  
File Actions Edit View Help  
0x00490950 9 92 sym.free_mem_6  
0x004909b0 17 218 → 200 sym.free_slotinfo  
0x00490a90 72 963 → 939 sym.free_mem_7  
0x00490e60 237 4239 → 4172 sym.arena_thread_freeres  
0x00491ef0 1 9 sym._fini  
0x006ba220 1 1021 obj._dl_init_static_tls  
0x006bc860 1 1020 obj._dl_wait_lookup_done  
[0x00400a30]> afl | grep main  
0x00400b4d 1 35 sym.main  
0x00400de0 10 1007 → 219 sym._libc_start_main  
0x00403840 39 661 → 629 sym._nl_find_domain  
0x00403ae0 308 5366 → 5301 sym._nl_load_domain  
0x00415ef0 1 43 sym._IO_switch_to_main_get_area  
0x0044ce10 1 8 sym._dl_get_dl_main_map  
0x00470430 1 49 sym._IO_switch_to_main_wget_area  
0x0048f9f0 7 73 → 69 sym._nl_fnddomain_subfreeres  
0x0048fa40 16 247 → 237 sym._nl_unload_domain
```

Examine the assembly code by running **pdf@main** command

```
[0x00400a30]> pdf @main
    ;-- 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    8045f4  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
[0x00400a30]>
```

Based on the assembly code, the value 1 is moved to local\_ch and local\_ch gets the value **1**.

```
[0x00400b4d    55      push rbp
0x00400b4e    4889e5  mov rbp, rsp
0x00400b51    c745f4010000. mov dword [local_ch], 1
0x00400b58    c745f8060000. mov dword [local_8h], 6
0x00400b5f    8045f4  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
[0x00400a30]>
```

## Question 6

Based on the assembly code, value 6 is copied into local\_8h, value of local\_ch which is 1 is copied to eax, imull instruction multiplies the values of local\_8h and eax ( $6 * 1 = 6$ ), and the value is return back to eax, so value of eax when imull instruction is called is **6**

```
[0x00400b4d    55      push rbp
0x00400b4e    4889e5  mov rbp, rsp
0x00400b51    c745f4010000. mov dword [local_ch], 1
0x00400b58    c745f8060000. mov dword [local_8h], 6
0x00400b5f    8045f4  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
[0x00400a30]>
```

## Question 7

The value of eax previously is copied to local\_4h, so the value of local\_4h before it is set to 0 is **6**

```
[0x00400b4d    55      push rbp
0x00400b4e    4889e5  mov rbp, rsp
0x00400b51    c745f4010000. mov dword [local_ch], 1
0x00400b58    c745f8060000. mov dword [local_8h], 6
0x00400b5f    8045f4  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
[0x00400a30]>
```

### **Thought Process/Methodology:**

Data type is matched with size in bytes based on the notes given in TryHackMe. Read the notes given in TryHackMe and we found that the command to analyse the program in radare2 is **aa**. Based on the notes in TryHackMe, we also found that the command to set a breakpoint in radare2 is **db** and the command to execute the program in until we hit a breakpoint is **dc**. We access the target machine using ssh and open binary in debugging mode to ask r2 to analyse the program challenge1. We use ‘afl | grep main’ to find the list of functions with the entry point that is defined as main. Then, **pdf@main** command is run to examine the assembly code. According to the assembly code, the value 1 is copied to local\_ch. Next, value 6 is copied into local\_8h, and value of local\_ch which is 1 is copied to eax. The imul instruction multiplies the values of local\_8h and eax ( $6*1=6$ ), and the value is returned back to eax, so when imul instruction is called we get an eax value of **6**. Next, the previous value of eax, which is 6, is copied to local\_4h, so the value of local\_4h before it is set to 0 is 6.

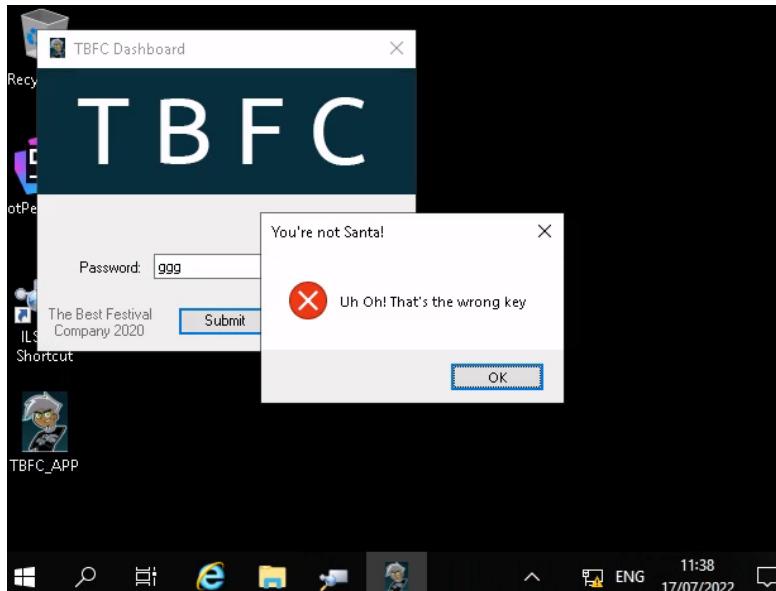
## **Day 18: Reverse Engineering – The Bits of Christmas**

**Tools used:** Kali Linux, Firefox

**Solution/Walkthrough:**

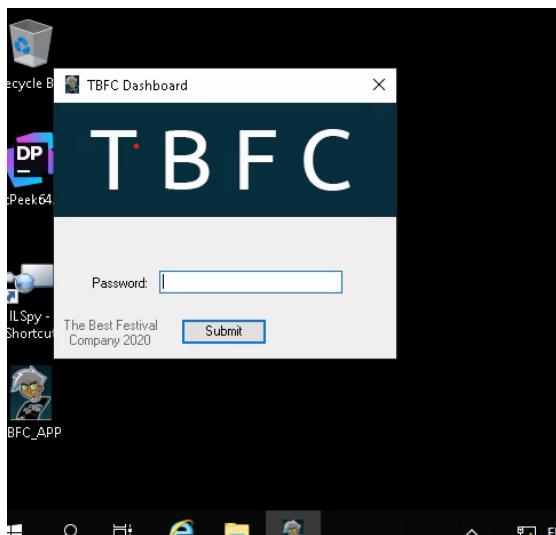
### **Question 1**

The message will be “Uh Oh! That’s the wrong key”



### **Question 2**

TBFC stand for **The Best Festival Company 2020**



### **Question 3**

The module that catches attention is **CrackMe**

The screenshot shows the ILSpy interface with the assembly **TBFC\_APP** selected. The left pane displays the assembly structure, and the right pane shows the assembly's manifest and security information.

```
// C:\Users\cmnatic\Desktop\TBFC_APP.exe
// CrackMe, Version=0.0.0.0, Culture=neutral
// Global type: <Module>
// Entry point: <Module>.main
// Architecture: x86
// This assembly contains unmanaged code.
// Runtime: v4.0.30319
// Hash algorithm: SHA1

[assembly: SecurityRules(SecurityRuleSet.Level1)]
[assembly: TargetFramework(".NETFramework, Version=v4.6.1")]
[assembly: SecurityPermission(SecurityAction.Demand, Level=Level1)]
[assembly: AssemblyVersion("0.0.0.0")]
```

#### Question 4

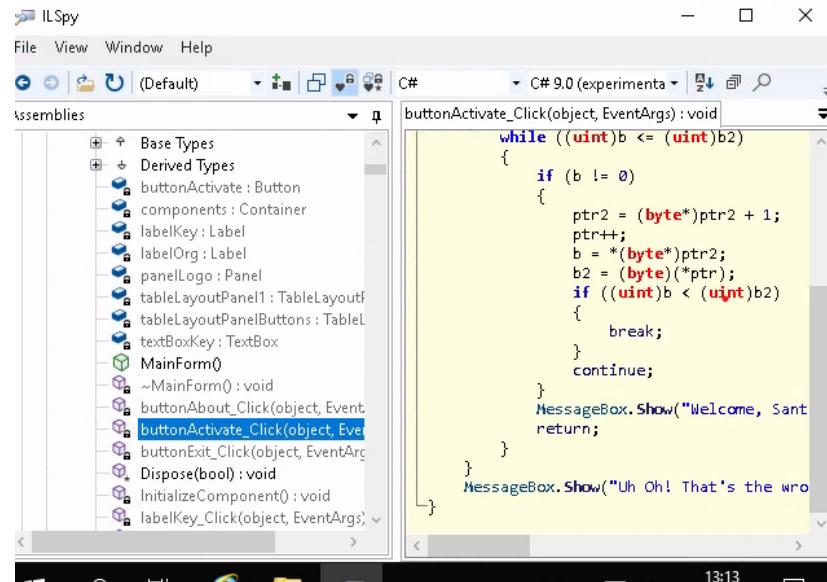
**MainForm** contains the information we are looking for.

The screenshot shows the ILSpy interface with the class **MainForm** selected. The left pane displays the class structure, and the right pane shows the class definition.

```
// CrackMe.MainForm
public class MainForm : Form
{
    private Label labelKey;
    private TextBox textBoxKey;
    private Panel panelLogo;
    private TableLayoutPanel tableLayoutPanel;
    private Button buttonActivate;
    private TableLayoutPanel tableLayoutPanel1;
    private Label labelOrg;
    private Container components;
```

## Question 5

buttonActivate\_Click contains the information we are seeking.

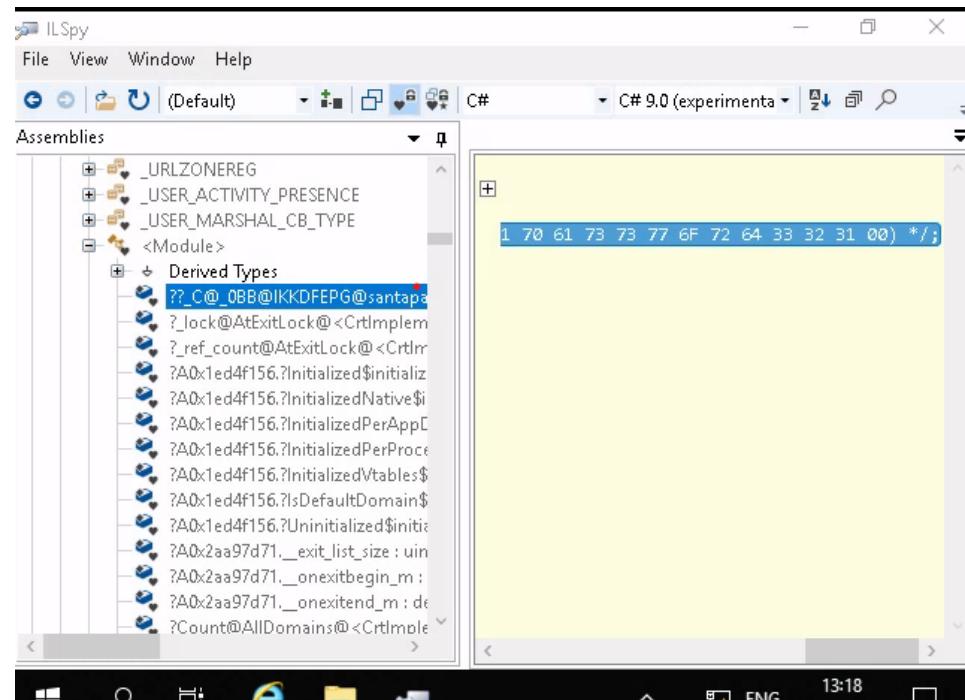


The screenshot shows the IL Spy interface with the assembly browser on the left and the decompiled code on the right. The assembly browser lists various types and methods, including 'buttonActivate\_Click(object, EventArgs) : void'. The decompiled code for this method is as follows:

```
buttonActivate_Click(object, EventArgs) : void
{
    while ((uint)b <= (uint)b2)
    {
        if (b != 0)
        {
            ptr2 = (byte*)ptr2 + 1;
            ptr++;
            b = *(byte*)ptr2;
            b2 = *(byte*)(*ptr);
            if ((uint)b < (uint)b2)
            {
                break;
            }
            continue;
        }
        MessageBox.Show("Welcome, Sant");
        return;
    }
    MessageBox.Show("Uh Oh! That's the wro");
}
```

## Question 6

When we open buttonActivate\_Click , we can see that the coding have some sensitive values and the password we can saw is santapassword123, but we are not sure whether it's the correct password or no, so we double click on it and it will direct us to another file which contain a hexadecimal number. We then use cyberchef to convert it into words and we can get the password that we are expecting, which is **santapassword321**.



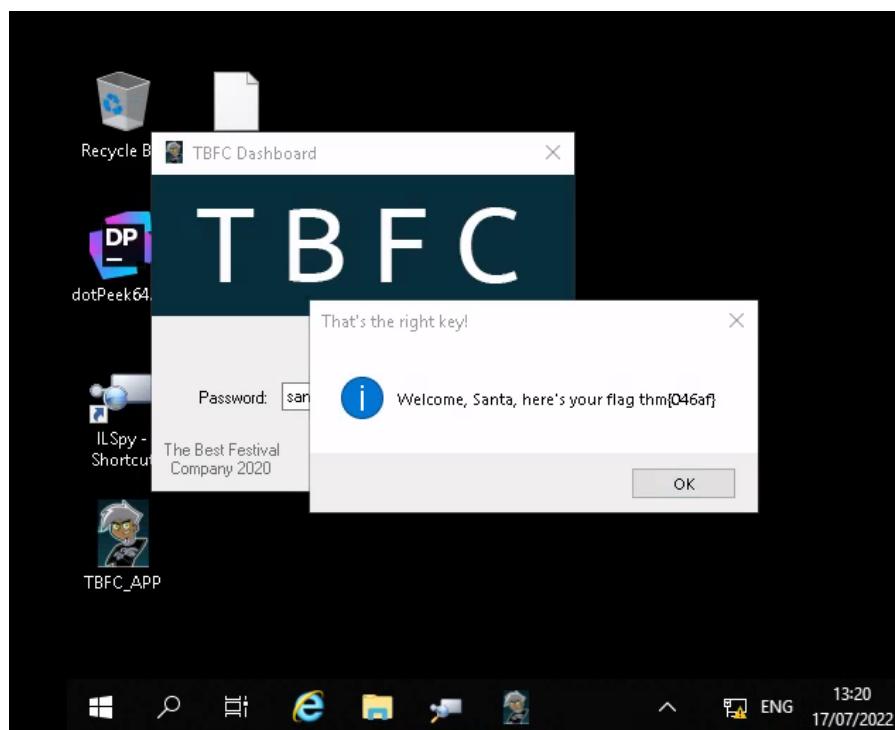
The screenshot shows the IL Spy interface with the assembly browser on the left. A specific module is selected, showing its derived types. One type, `??_C@_0BB@IKKDFEPG@santap`, is highlighted. The decompiled code for this type is shown in the main pane:

```
1 70 61 73 73 77 6F 72 64 33 32 31 00) */;
```

The screenshot shows the CyberChef interface. In the 'Input' section, there is a hex dump of the password: 73 61 6E 74 61 70 61 73 73 77 6F 72 64 33 32 C0. The 'Output' section shows the decoded password: santapassword321.

### Question 7

The flag will be **thm{046af}**



### **Thought Process/Methodology:**

First of all, we should download a tool named Remmina by using the 'sudo apt-get install remmina' command. Then, we can start to connect to the machine using the username and password that have already been given by the website which is cmnatic for username and Adventofcyber! For password. Once we successfully log in, we will see a windows desktop. Then, we will open TBFC\_APP and we will be asked to log in with password. After that, we can then open ILSpy and this will allow us to get a look at the source code of the app. We can then press CrackMe and click MainForm and finally look for

buttonActivate\_Click. We can see inside the buttonActivate\_Click source code, we can see that the coding have some sensitive values and the password we can saw is santapassword123, but we are not sure whether it's the correct password or no, so we double click on it and it will direct us to another file which contain a hexadecimal number. We then use cyberchef to convert it into words and we can get the password that we are expecting, which is santapassword123. We can then use the expected password to log in to the TBFC\_APP and we will then get our flag.

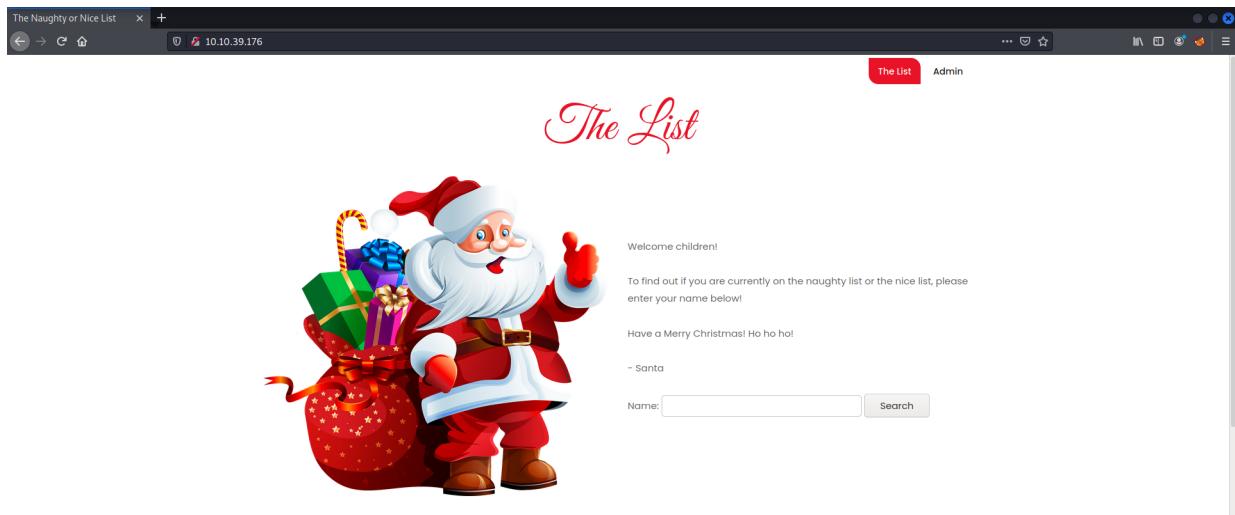
## **Day 19: Web Exploitation – The Naughty or Nice List**

**Tools used:** Kali Linux, Firefox

**Solution/Walkthrough:**

### Question 1

Once the attack machine is deployed, we proceed to connect to the web app. The webpage “The Naughty or Nice List” had appeared.



After opening the webpage, we enter a name in the form and click the “Search” button to see whether the name is on the Naughty List or the Nice List. We entered the first name, “**Ian Chai**” and found it is on the **Nice List**.

- Santa

Name:  Search

Ian Chai is on the Nice List.

We then proceed to the next name which is “**YP**” and found it is on the **Nice List** too.



- [SANTA](#)

Name:

Search

**YP is on the Nice List.**

Next, we entered the name “**JJ**” and found it is on the **Naughty List**.



Name:

Search

**JJ is on the Naughty List.**

We then proceed to the next name, “**Timothy**”, and found it is on the **Naughty List**.

- [SANTA](#)

Name:

Search

**Timothy is on the Naughty List.**

Next, the name “**Tib3rius**” was entered, and we found that it is on the **Nice List**.

- Santa

Name:

Search

**Tib3rius is on the Nice List.**

Last, the name “**Kanes**” was entered, and we found that it is on the **Naughty List**.

Name:

Search

**Kanes is on the Naughty List.**

## Question 2

To fetch the root of the same site, we apply “/?proxy=http%3A%2F%2Flist.hohoho%3A8080%2F” at the end of the web app URL, the page displays the text “**Not Found. The requested URL was not found on this server.**”

Name:

Search

**Not Found**

**The requested URL was not found on this  
server.**

### Question 3

Next, we tried changing the port number from 8080 to just 80. By entering "/?proxy=http%3A%2F%2Flist.hohoho%3A80" at the end of the web app URL, the page displayed the text "**Failed to connect to list.hohoho port 80: Connection refused**" which suggests that port 80 is not open on list.hohoho.

Name:

Search

Failed to connect to list.hohoho port 80:

Connection refused

### Question 4

We then tried changing the port number to 22 which is the default SSH port. We used "/?proxy=http%3A%2F%2Flist.hohoho%3A22" at the end of the URL, and the message now changed into "**Recv failure: Connection reset by peer**" which suggests that port 22 is open but did not understand what was sent.

Name:

Search

Recv failure: Connection reset by peer

### Question 5

We tried another way by replacing the “list.hohoho” hostname with “localhost” by entering “/?proxy=http%3A%2F%2Flocalhost” at the end of the URL. We found that the message returned says “**Your search has been blocked by our security team.**” By looking at the message, we know that the developer has implemented a check to ensure that the hostname provided starts with “list.hohoho”, and will block any hostnames that don’t.

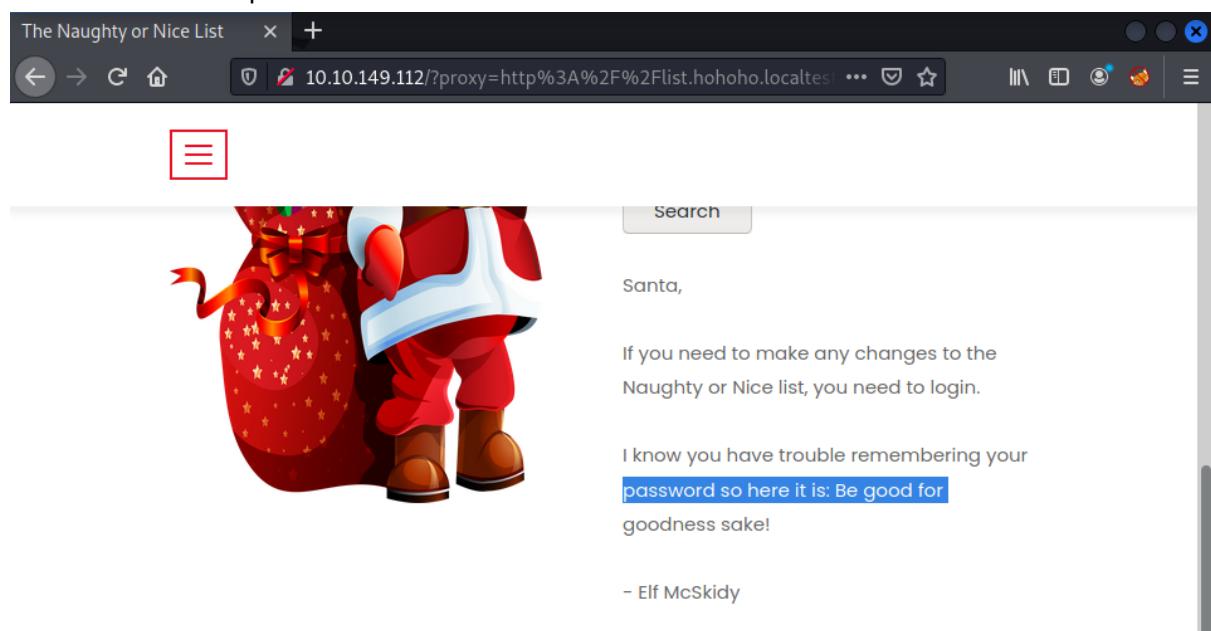
Name:

Search

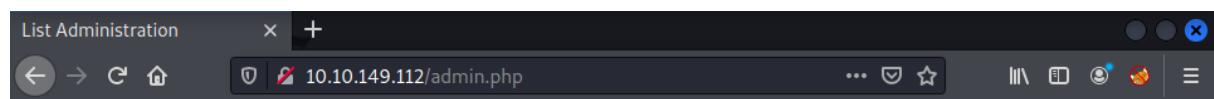
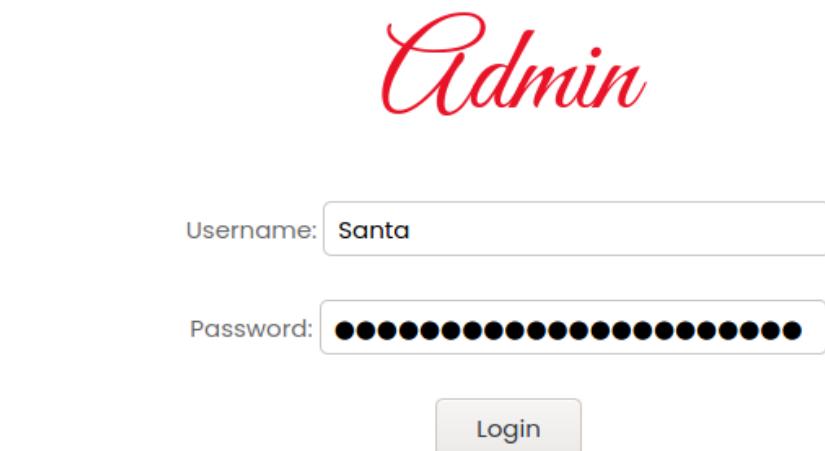
Your search has been blocked by our  
security team.

### Question 6

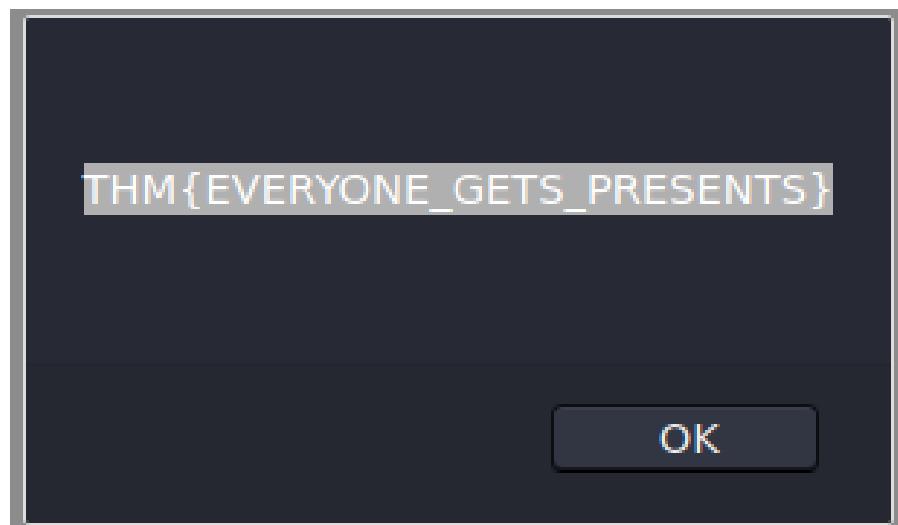
We know that the hostname needs to starts with “list.hohoho”, we than set the hostname in the URL to “list.hohoho.localtest..me”. We used “/?proxy=http%3A%2F%2Flocalhost” at the end of the URL. By using it, we successfully access the web server and found a message from Elf McSkidy that contains the Santa’s password.



We know that Santa's password is “**Be good for goodness sake!**”. We then use the password provided by Elf McSkidy and successfully entered the admin page.



After we entered the admin page, we clicked the “DELETE NAUGHTY LIST” and the challenge flag appeared which is **THM{EVERYONE\_GETS\_PRESENTS}**.



### **Thought Process/Methodology:**

After we navigated to the web app, we entered the names in the form and click the “Search” button to check whether the name is on the Naughty List or Nice List. Next, we tried to fetch the root of the same site and browse to `http://machine_ip/?proxy=http%3A%2F%2Flist.hohoho%3A8080%2F`. The message, “**Not Found. The requested URL was not found on this server.**” has displayed on the web app. We know that we were able to make the server request the modified URL and return the response. We then tried changing the port number from 8080 to just 80. We entered the URL, `http://machine_ip/?proxy=http%3A%2F%2Flist.hohoho%3A80` and the message now change to “**Failed to connect to list.hohoho port 80: Connection refused**” which suggests that port 80 is not open on list.hohoho. We then tried by changing the port number to 22 and entered the URL `http://machine_ip/?proxy=http%3A%2F%2Flist.hohoho%3A22`, the message now changes to “**Recv failure: Connection reset by peer**” which suggests that port 22 is open but did not understand what was sent. We tried another way by replacing the list.hohoho hostname with “localhost”. We entered the URL `http://machine_ip/?proxy=http%3A%2F%2Flocalhost`, the message returned says “**Your search has been blocked by our security team.**” By looking at the message, we know that the developer has implemented a check to ensure that the hostname provided starts with “list.hohoho”, and will block any hostnames that don't. By knowing this, we tried and set the hostname in the URL to “list.hohoho.localtest.me”, and we had successfully access the local service by entering the URL `http://machine_ip/?proxy=http%3A%2F%2Flist.hohoho.localtest.me`. The admin password is given in the message written by Elf McSkidy. After knowing the password, we clicked at the “Admin” link at the top and proceeded to login to it. We put the username as “Santa” and the password we found to login as santa. A list administration page appeared after we have login. Lastly, we proceeded to delete the naughty list and the challenge flag appeared.

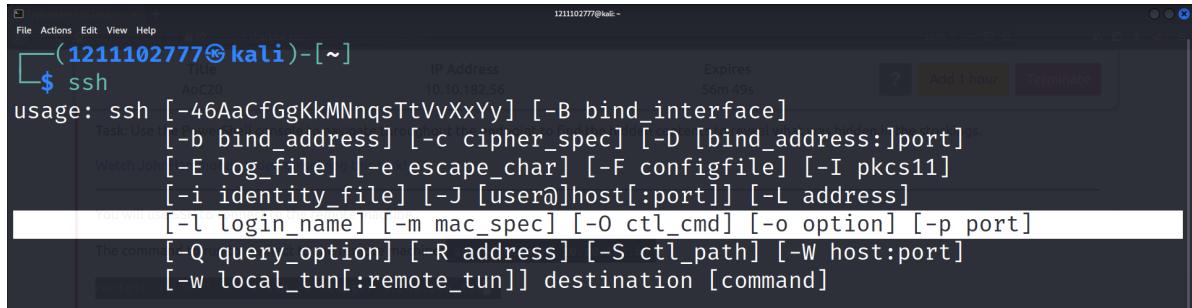
## **Day 20: Blue Teaming – Powershell to the rescue**

**Tools used:** Kali Linux, Firefox

**Solution/Walkthrough:**

### Question 1

We accessed the terminal in Kali Linux to check ssh manual, we know that parameter -l do **login name**

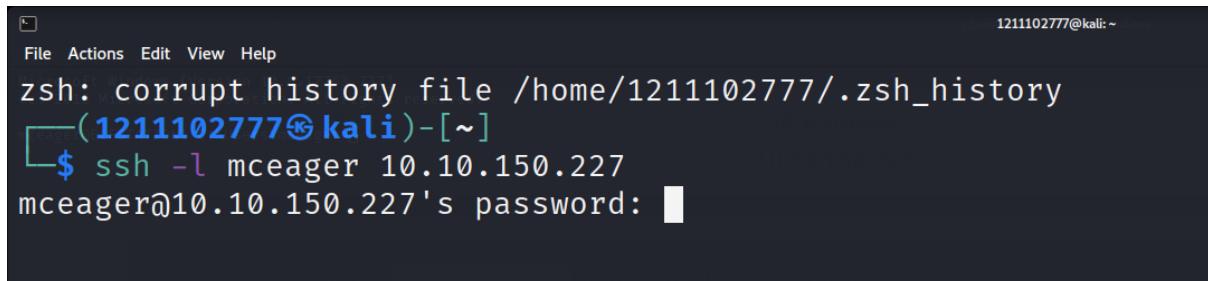


A screenshot of a terminal window titled '(1211102777㉿kali)-[~]'. The window shows the ssh man page. The command entered was '\$ ssh'. The output includes the usage information for ssh, which includes the '-l' option for specifying a login name.

```
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface]
           [-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]] destination [command]
```

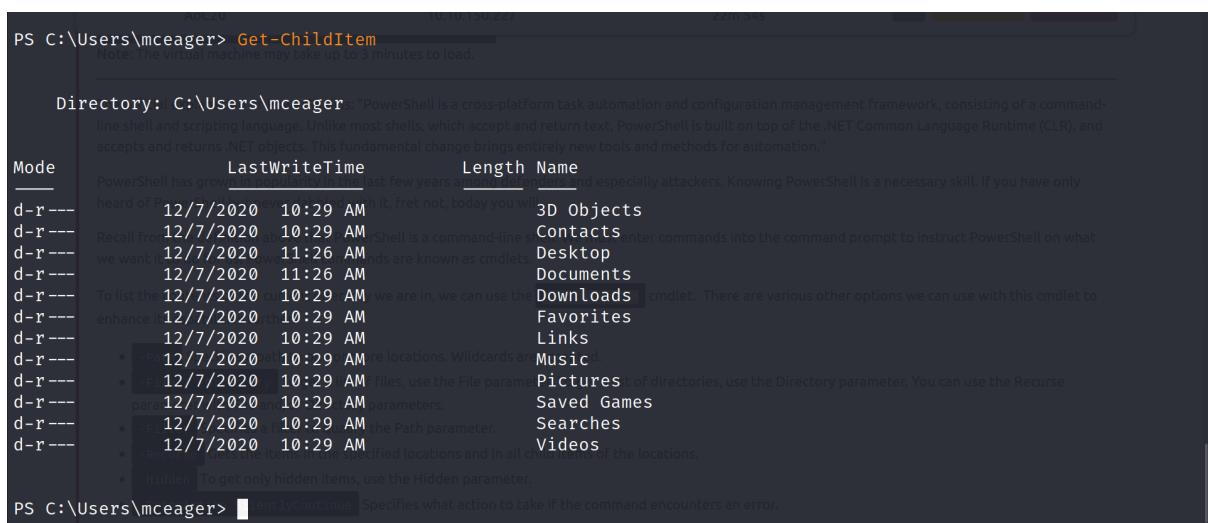
### Question 2

We connected to the remote machine by entering `ssh -l mceager MACHINE_IP` and password r0ckStar!



A screenshot of a terminal window titled '1211102777㉿kali: ~'. The user has entered the command '\$ ssh -l mceager 10.10.150.227'. The prompt asks for 'mceager@10.10.150.227's password:'. The password 'r0ckStar!' is being typed.

We checked for the current directory we are in by using `Get-ChildItem`



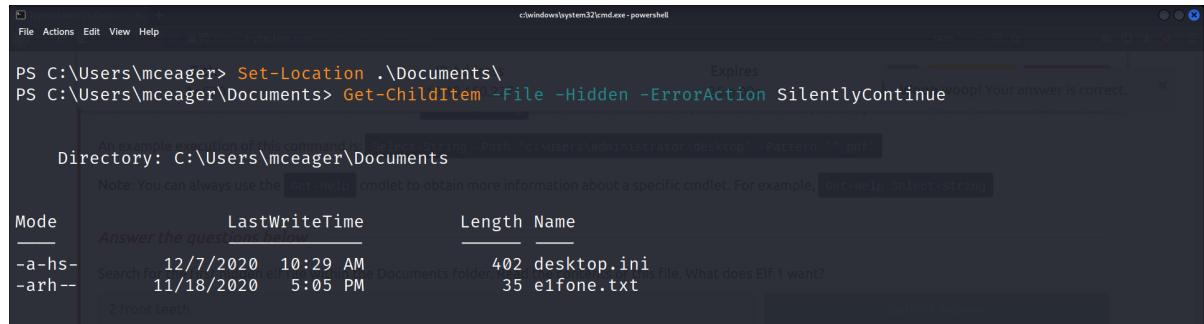
A screenshot of a PowerShell window titled 'PS C:\Users\mceager>'. The user has run the command 'Get-ChildItem'. The output shows the contents of the 'C:\Users\mceager' directory, including subfolders like 'Desktop', 'Documents', 'Downloads', 'Favorites', 'Links', 'Music', 'Pictures', 'Saved Games', 'Searches', and 'Videos'.

```
PS C:\Users\mceager> Get-ChildItem
    Directory: C:\Users\mceager
Mode                LastWriteTime         Length Name
d-r----
```

Then, we managed to get into Documents folder

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

We proceeded with searching for the hidden content



```
PS C:\Users\mceager> Set-Location .\Documents\  
PS C:\Users\mceager\Documents> Get-ChildItem -File -Hidden -ErrorAction SilentlyContinue  
pop! Your answer is correct.  
An example execution of this command is: Select-String -Path 'c:\users\administrator\Desktop' -Pattern '*.pdf'  
Directory: C:\Users\mceager\Documents  
Note: You can always use the Get-Help cmdlet to obtain more information about a specific cmdlet. For example, Get-Help Select-String  


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

  
Answer the question  
2 front teeth
```

By reading the content of hidden elf file, we know that Elf 1 want **2 front teeth**

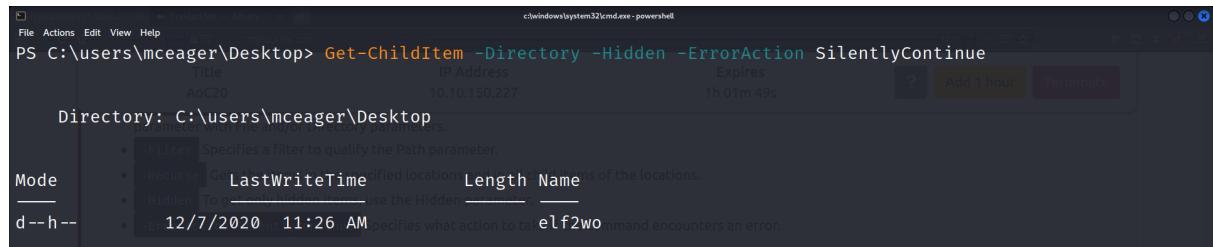
```
PS C:\Users\mceager\Documents> Get-Content -Path e1fone.txt  
All I want is my '2 front teeth'!!!  
PS C:\Users\mceager\Documents>
```

### Question 3

We changed the directory to desktop.

```
PS C:\users\mceager\Documents> Set-Location -Path c:\users\mceager\Desktop
```

We proceeded to search for the hidden directory.



```
PS C:\users\mceager\Desktop> Get-ChildItem -Directory -Hidden -ErrorAction SilentlyContinue  
pop! Your answer is correct.  
An example execution of this command is: Select-String -Path 'c:\users\administrator\Desktop' -Pattern '*.pdf'  
Directory: C:\users\mceager\Desktop  
Title IP Address Expires  
AoC20 10.10.150.227 1h 01m 49s  


| Mode  | LastWriteTime      | Length                                                           | Name   |
|-------|--------------------|------------------------------------------------------------------|--------|
| d-h-- | 12/7/2020 11:26 AM | pecifies what action to take if the command encounters an error. | elf2wo |

  
• -Filter Specifies a filter to qualify the Path parameter.  
• -Recurse Gets items from all levels of the specified location and its child locations of the locations.  
• -Hidden To get hidden items. To ignore hidden items, use the Hidden parameter.
```

Afterwards, we read the hidden text file and realised **Scrooged** is the name of that movie that Elf 2 wants.

```

For example, if you want to view all of the hidden files in the current directory you are in, you can issue the following command:
PS C:\users\mceager\Desktop> Set-Location .\elf2wo\
PS C:\users\mceager\Desktop\elf2wo> Get-ChildItem
Another user of choice is [Get-Content]. This will allow you to read the contents of a file.

Directory: C:\users\mceager\Desktop\elf2wo.txt
Mode          LastWriteTime    Length Name
-a----- 11/17/2020   10:26 AM      64 e70smsW10Y4k.txt
You run this command as follows: [Get-Content -Path file.txt] | Measure-Object -Word.

PS C:\users\mceager\Desktop\elf2wo> Get-Content -Path e70smsW10Y4k.txt | Measure-Object -Word.
I want the movie Scrooged <3!
PS C:\users\mceager\Desktop\elf2wo> [The index is the numerical value that is the location of the string within the file. Since indexes start at zero, you typically need to subtract one from the original value to get the correct word count. This is not necessary for this exercise.]

```

## Question 4

Then, we changed the directory to Windows by entering `Set-Location -Path C:\Windows`

Hidden folder that contains files for Elf 3 has been found – **3lfthr3**

```

PS C:\users\mceager\desktop\elf2wo> Set-Location -Path C:\Windows
PS C:\Windows> Get-ChildItem -Directory -Hidden -ErrorAction SilentlyContinue -Path C:\Windows\ -Filter '*3*' -Recurse
You can run numerous operations with the [Get-Content] cmdlet to give you more information about the particular file you are inspecting. Such as how many words are in the file and the exact positions for a particular string within a file.

Directory: C:\Windows\System32 a file, you can use the [Get-Content] cmdlet and pipe the results to the [Measure-Object] cmdlet.
Mode          LastWriteTime    Length Name
d-h-- 11/23/2020   3:26 PM      3lfthr3e
You run this command as follows: [Get-Content -Path file.txt] | Measure-Object -Word.

To get the exact position of the string within the file, you can follow this command: ([Get-Content -Path file.txt][index])
[The index is the numerical value that is the location of the string within the file. Since indexes start at zero, you typically need to subtract one from the original value to get the correct word count. This is not necessary for this exercise.]

```

## Question 5

Access the hidden folder by entering `Set-Location .\System32\3lfthr3e`

Later, we searched for the hidden content by entering `Get-ChildItem -Hidden`

In order to find the word count of the first file contain, we entered `Get-Content -Path 1.txt | Measure-Object -Word`

We realised the file contained **9999** words.

```

PS C:\Windows> Set-Location .\System32\3lfthr3e
PS C:\Windows\System32\3lfthr3e> Get-ChildItem -Hidden
    You can run numerous operations with the Get-Content cmdlet to give you more information about the particular file you are inspecting. Such as how many words are in the file and the exact positions for a particular string within a file.
    Directory: C:\Windows\System32\3lfthr3e
    To get the number of words contained within a file, you can use the Get-Content cmdlet and pipe the results to the Measure-Object cmdlet.

Mode          LastWriteTime      Length Name
--           11/17/2020 10:58 AM     85887 1.txt
--           11/23/2020 3:26 PM      12061168 2.txt
    The index value is the location of the word. Since indexes start at zero, you typically need to subtract one from the original value to extract the string at the correct position. This is not necessary for this exercise.

PS C:\Windows\System32\3lfthr3e> Get-Content -Path 1.txt | Measure-Object -Word
    Lines Words Characters Property
    9999
    An example execution of this command is: Select-String -Path "c:\users\administrator\Desktop\" -Pattern "pdf"

PS C:\Windows\System32\3lfthr3e> [cmdlet to obtain more information about a specific cmdlet. For example, Get-Help Select-String]

```

## Question 6

We used `(Get-Content -Path file.txt)[index]` to get the exact position of a string within a file.

We discovered **Red** & **Ryder** are at index 551 and 6691 respectively

```

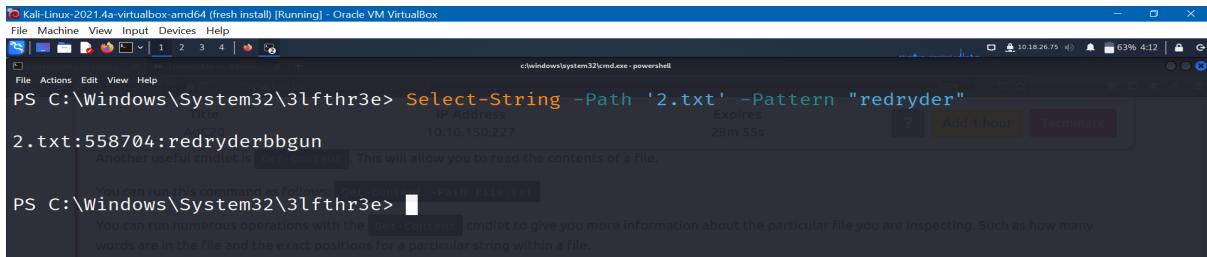
PS C:\Windows\System32\3lfthr3e> (Get-Content -Path 1.txt)[551]
Red
PS C:\Windows\System32\3lfthr3e> (Get-Content -Path 1.txt)[6691]
Ryder
PS C:\Windows\System32\3lfthr3e> [cmdlet to obtain more information about a specific cmdlet. For example, Get-Help Select-String]

```

## Question 7

Finally, we searched in the 2nd file which is 2.txt for the phrase from the previous question by entering `Select-String -Path '2.txt' -Pattern "redryder"`

We eventually found **redryderbbgun**.



```

Kali-Linux-2021.4a-virtualbox-amd64 [fresh install] [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
File Actions Edit View Help
e:\windows\system32\cmd.exe - powershell
PS C:\Windows\System32\3lfthr3e> Select-String -Path '2.txt' -Pattern "redryder"
2.txt:558704:redryderbbgun
Another useful cmdlet is Get-Content. This will allow you to read the contents of a file.

PS C:\Windows\System32\3lfthr3e> [cmdlet to obtain more information about a specific cmdlet. For example, Get-Help Select-String]

```

## **Thought Process/Methodology:**

We begin with accessing the terminal and search for the ssh manual followed by connecting to the remote machine. Once we entered successfully, we directed to the Documents folder and explored its hidden content to find the items Elf 1 wanted, which is **2 front teeth**. Next, we searched on the Desktop folder and explored its hidden content to get the movie that Elf 2 wants, which is **Scrooged**.

Then, we accessed to Windows directory in order to search for the name of the hidden folder in it, which is **3lfthr3e**. After gaining access to the hidden folder, we managed to get the word count of the first file, which is **9999** words. We then proceeded to get the specific words, which is **Red & Ryder** at index 551 and 6991 respectively. Last but not least, we searched in the 2.txt file for the phrase from redryder to find what Elf 3 wants, which is **redryderbbgun**.