

PSP0201

Week 5

Writeup

Group Name: UrKomputerHasPirus

Members:

| ID | Name | Role |
|------------|-------------------------|--------|
| 1211102272 | Tee Cheng Jun | Leader |
| 1211101114 | Chong Yi Jing | Member |
| 1211101591 | Ian Leong Tsung Jii | Member |
| 1211101734 | Ernest Leong Zheng Yang | Member |

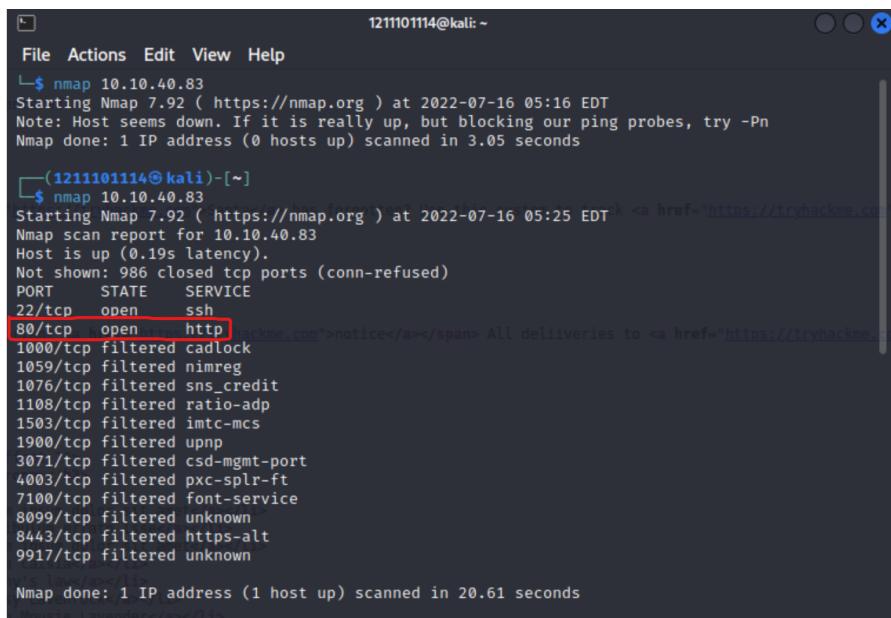
Day 16 : Help! Where is Santa?

Tools used: nmap, Python3

Solution/Walkthrough

Question 1

Use nmap to find the http port



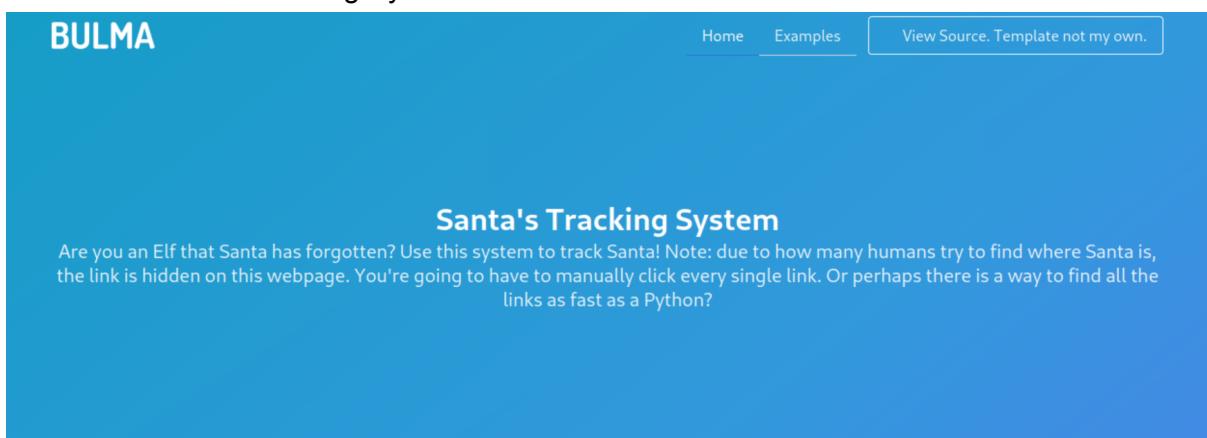
```
File Actions Edit View Help
$ nmap 10.10.40.83
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-16 05:16 EDT
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.05 seconds

(1211101114㉿kali)-[~]
$ nmap 10.10.40.83
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-16 05:25 EDT <a href="https://tryhackme.com">notice</a><span> All deliveries to <a href="https://tryhackme.com">notice</a>
Nmap scan report for 10.10.40.83
Host is up (0.19s latency).
Not shown: 986 closed tcp ports (conn-refused)
PORT      STATE     SERVICE
22/tcp    open      ssh
80/tcp    open      http
1000/tcp  filtered cadlock
1059/tcp  filtered nimreg
1076/tcp  filtered sns_credit
1108/tcp  filtered ratio-adp
1503/tcp  filtered imtc-mcs
1900/tcp  filtered upnp
3071/tcp  filtered csd-mgmt-port
4003/tcp  filtered pxc-splr-ft
7100/tcp  filtered font-service
8099/tcp  filtered unknown
8443/tcp  filtered https-alt
9917/tcp  filtered unknown
Nmap done: 1 IP address (1 host up) scanned in 20.61 seconds
```

Pretty clear that the http is the port for the website.

Question 2

This is the Santa's Tracking System website



BULMA

Home Examples View Source. Template not my own.

Santa's Tracking System

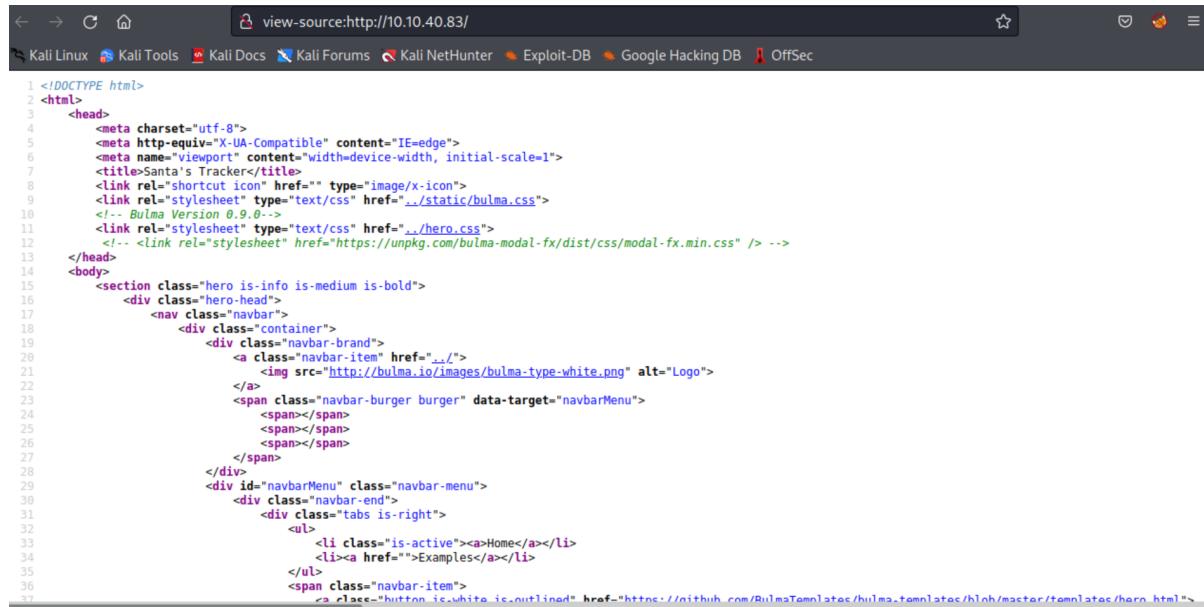
Are you an Elf that Santa has forgotten? Use this system to track Santa! Note: due to how many humans try to find where Santa is, the link is hidden on this webpage. You're going to have to manually click every single link. Or perhaps there is a way to find all the links as fast as a Python?

By viewing the page source, we can tell that there are many mentionings of Bulma.io having us safely assume the template being used is BULMA.

```
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title>Santa's Tracker</title>
8     <link rel="shortcut icon" href="" type="image/x-icon">
9     <link rel="stylesheet" type="text/css" href="../static/bulma.css">
10    <!-- Bulma Version 0.9.0 -->
11    <link rel="stylesheet" type="text/css" href="../hero.css">
12    <!-- <link rel="stylesheet" href="https://unpkg.com/bulma-modal-fx/dist/css/modal-fx.min.css" /> -->
13  </head>
14  <body>
15    <section class="hero is-info is-medium is-bold">
16      <div class="hero-head">
17        <nav class="navbar">
18          <div class="container">
19            <div class="navbar-brand">
20              <a class="navbar-item" href="#">/>
21              
22            </a>
23            <span class="navbar-burger burger" data-target="navbarMenu">
24              <span></span>
25              <span></span>
26              <span></span>
27            </span>
28          </div>
29          <div id="navbarMenu" class="navbar-menu">
30            <div class="navbar-end">
31              <div class="tabs is-right">
32                <ul>
33                  <li class="is-active"><a>Home</a></li>
34                  <li><a href="">Examples</a></li>
35                </ul>
36                <span class="navbar-item">
37                  <a class="button is-white is-outlined" href="https://github.com/BulmaTemplates/bulma-templates/blob/master/templates/hero.html">
38                    <span class="icon">
```

Question 3

After entering Santa's Tracker by entering “10.10.40.83:80”, you can right click the page OR use Ctrl + U to open the source page.



The screenshot shows a Kali Linux terminal window with a browser tab open to "view-source:10.10.40.83". The browser title bar says "view-source:10.10.40.83". The page content is the raw HTML source code of the website, which includes the Bulma CSS imports and the hero section structure.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title>Santa's Tracker</title>
8     <link rel="shortcut icon" href="" type="image/x-icon">
9     <link rel="stylesheet" type="text/css" href="../static/bulma.css">
10    <!-- Bulma Version 0.9.0 -->
11    <link rel="stylesheet" type="text/css" href="../hero.css">
12    <!-- <link rel="stylesheet" href="https://unpkg.com/bulma-modal-fx/dist/css/modal-fx.min.css" /> -->
13  </head>
14  <body>
15    <section class="hero is-info is-medium is-bold">
16      <div class="hero-head">
17        <nav class="navbar">
18          <div class="container">
19            <div class="navbar-brand">
20              <a class="navbar-item" href="#">/>
21              
22            </a>
23            <span class="navbar-burger burger" data-target="navbarMenu">
24              <span></span>
25              <span></span>
26              <span></span>
27            </span>
28          </div>
29          <div id="navbarMenu" class="navbar-menu">
30            <div class="navbar-end">
31              <div class="tabs is-right">
32                <ul>
33                  <li class="is-active"><a>Home</a></li>
34                  <li><a href="">Examples</a></li>
35                </ul>
36                <span class="navbar-item">
37                  <a class="button is-white is-outlined" href="https://github.com/BulmaTemplates/bulma-templates/blob/master/templates/hero.html">
38                    <span class="icon">
```

After some filtering through the code, you should be able to find the directory for the API.

```

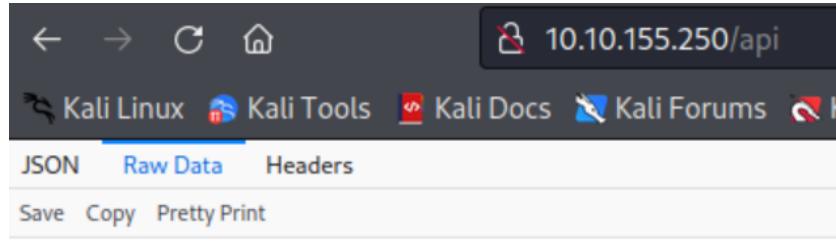
<footer class="footer">
  <div class="container">
    <div class="columns">
      <div class="column is-3 is-offset-2">
        <h2><strong>Category</strong></h2>
        <ul>
          <li><a href="#">Lorem ipsum dolor sit amet</a></li>
          <li><a href="#">Vestibulum errato isse</a></li>
          <li><a href="#">Lorem ipsum dolor sit amet</a></li>
          <li><a href="#">Aisia caisia</a></li>
          <li><a href="#">Murphy's law</a></li>
          <li><a href="#">Flimsy Lavenrock</a></li>
          <li><a href="#">Maven Mousie Lavender</a></li>
        </ul>
      </div>
      <div class="column is-3">
        <h2><strong>Category</strong></h2>
        <ul>
          <li><a href="#">Labore et dolore magna aliqua</a></li>
          <li><a href="#">Kanhan 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>
      <div class="column is-4">
        <h2><strong>Category</strong></h2>
        <ul>

```

OR

If it wasn't obvious enough just put in "/api/"

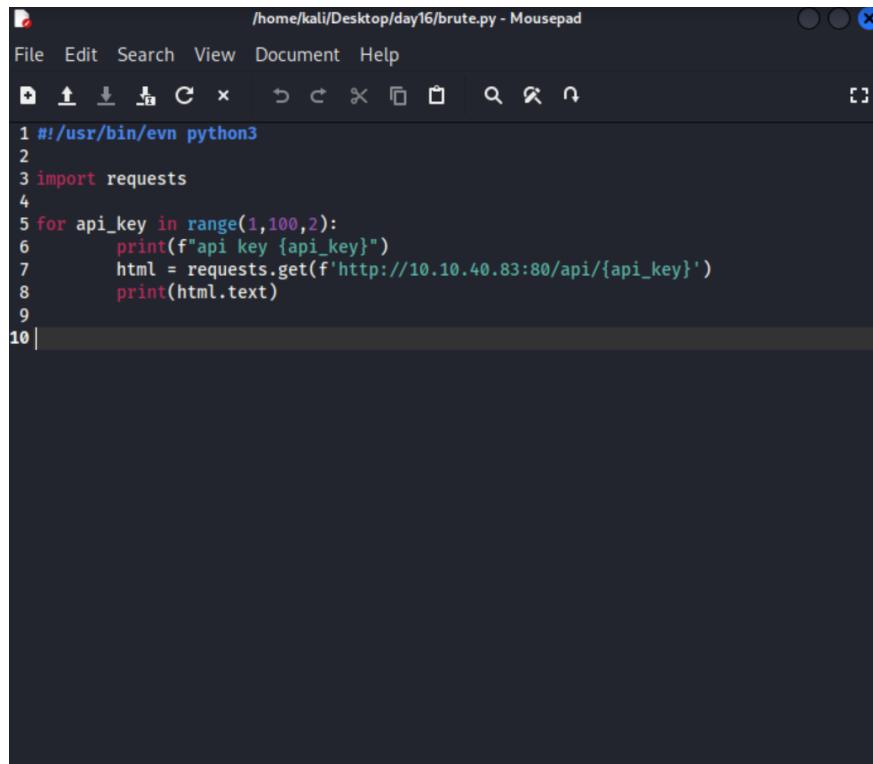
Question 4



By opening the raw data on "machine_ip/api/" it's shown that the website's response is {"detail": "Not Found"}

Question 5

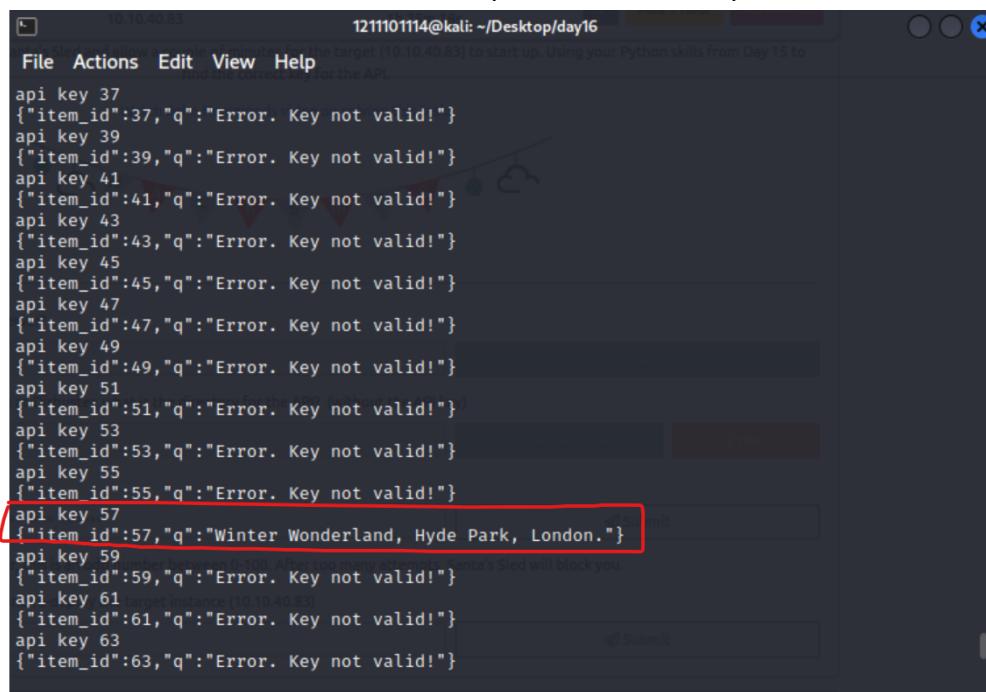
Create this file and run the python file in order to find which API key is valid



The screenshot shows a terminal window titled '/home/kali/Desktop/day16/brute.py - Mousepad'. The window contains a Python script with the following code:

```
1 #!/usr/bin/python3
2
3 import requests
4
5 for api_key in range(1,100,2):
6     print(f"api key {api_key}")
7     html = requests.get(f'http://10.10.40.83:80/api/{api_key}')
8     print(html.text)
9
10|
```

After creating this, run the script in order to find the correct API key. The correct API key should stick out like a sore thumb compared to the other prints.



The screenshot shows a terminal window titled '10.10.40.83' with the command '1211101114@kali: ~/Desktop/day16'. The window displays the output of the 'brute.py' script, which prints numerous API keys and their corresponding error messages. The correct API key, '57', is highlighted with a red box and shows a response indicating Santa's location: '{"item_id":57,"q":"Winter Wonderland, Hyde Park, London."}'.

Thus Santa is currently in "Winter Wonderland, Hyde Park, London."

Question 6

Referring to the screenshot above, this implies that the correct API key is 57.

Thought Process :

To begin with, we started using nmap on the given machine_ip in hopes of finding the port for the website. Through the nmap scan it was pretty obvious that the http service was the website. After opening the website, we went straight for the page source, we were able to extract the website using plenty of BULMA templates and a hint as to how to find the api we needed. It states that the link follows the given domains "http://machine_ip/api/api_key". But if we were to simply brute force it 1 by 1, we would most certainly be IP banned and fail the task. To avoid this, we simply made a simple Python script in order to get a number of responses from the website. Running the script it followed the given instructions and went through 1 api key at a time. Soon enough, we've received a unique response from api key 57 which returned "Winter Wonderland, Hyde Park, London." which was as clear as day that it had to be Santa's location.

Day 17 : ReverseELFneering

Tools used: radare2, SSH

Solution/Walkthrough

Question 1

Referring to the info given in THM, it shows the sizes of different data types

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 |

Question 2

This one is pretty simple, referring to the radare2 Cheatsheet (provided by THM). It shows that to analyse all is “aa”

Analysis [a]

| | |
|------------------------------|------------|
| Analyse all | aa [a [a]] |
| Analyse function calls | aac |
| Analyse consecutive function | aat |
| Graphviz output | ag <addr> |
| ASCII Graph | agf |
| Xrefs from | axf |
| Xrefs to | axt |
| Rename function | afn |
| Rename locals/args | afvn |
| List functions | afl |

Question 3

Also by referring to the cheatsheet. Setting a breakpoint is “db”

Debugger [d]

| | |
|----------------------------|-------------|
| Set breakpoint | db [addr] |
| Remove breakpoint | db - [addr] |
| Continue execution | dc |
| Show memory maps | dm |
| Show memory maps with bars | dm= |
| Start process | do |
| Attach to process | dp |
| Show register | dr |
| Step | ds [num] |
| Step over | dso [num] |
| Show heap graph | dmhg |
| List heap chunks | dmh |
| List heap chunks of arena | dmh <arena |
| List bins | dmhb |
| List fastbins | dmhf |

Question 4

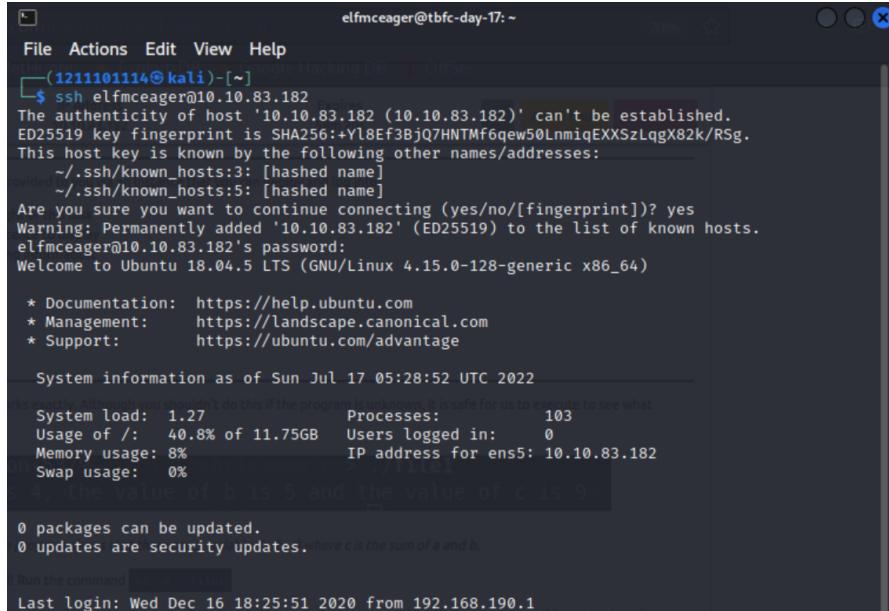
Additionally, by referring to the cheat sheet, we find out that the command is “dc”

Debugger [d]

| | |
|----------------------------|-------------|
| Set breakpoint | db [addr] |
| Remove breakpoint | db - [addr] |
| Continue execution | dc |
| Show memory maps | dm |
| Show memory maps with bars | dm= |
| Start process | do |
| Attach to process | dp |
| Show register | dr |
| Step | ds [num] |
| Step over | dso [num] |
| Show heap graph | dmhg |
| List heap chunks | dmh |
| List heap chunks of arena | dmh <arena |
| List bins | dmhb |
| List fastbins | dmhf |

Question 5

We can start by opening the cmd prompt and using ssh with command “ssh `elfmceager@machine_ip`”. Then logging into it with the “adventofcyber” password



The screenshot shows a terminal window titled "elfmceager@tbfc-day-17: ~". The user has run the command `ssh elfmceager@10.10.83.182`. The system prompts for confirmation of the host's fingerprint and asks for the password. Once logged in, the user sees a standard Ubuntu 18.04 LTS welcome message and system information. The terminal also displays a warning about the value of variables `a`, `b`, and `c`. At the bottom, it shows the last login details.

```
elfmceager@tbfc-day-17:~$ ssh elfmceager@10.10.83.182
The authenticity of host '10.10.83.182 (10.10.83.182)' can't be established.
ED25519 key fingerprint is SHA256:+Yl8Ef3BjQ7HNTMf6qew50LnmiqEXXSzLqgX82k/RSg.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:3: [hashed name]
  ~/.ssh/known_hosts:5: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.83.182' (ED25519) to the list of known hosts.
elfmceager@10.10.83.182's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-128-generic x86_64)

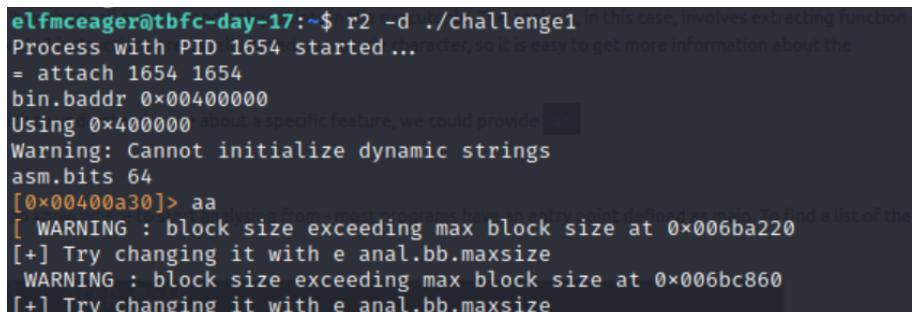
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun Jul 17 05:28:52 UTC 2022
System load: 1.27      Processes:          103
Usage of /: 40.8% of 11.75GB   Users logged in:     0
Memory usage: 8%           IP address for ens5: 10.10.83.182
Swap usage: 0%
[1] the value of b is 5 and the value of c is 9
0 packages can be updated.
0 updates are security updates. here c is the sum of a and b.

Run the command [REDACTED] to see more.

Last login: Wed Dec 16 18:25:51 2020 from 192.168.190.1
```

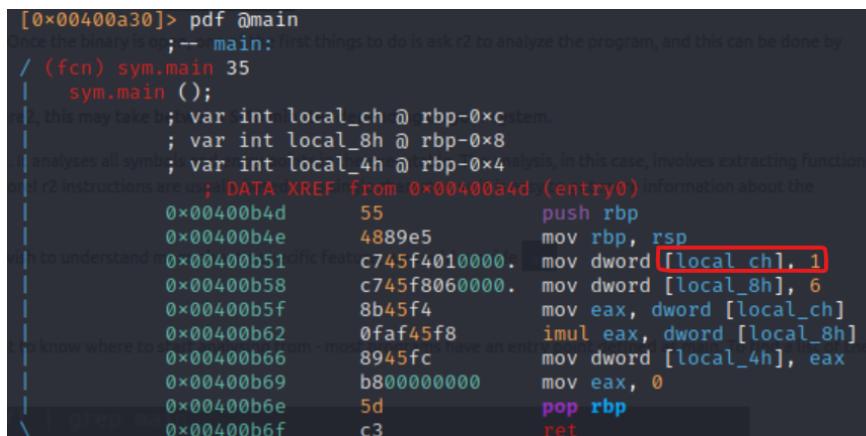
Then using radare2 input the command : “`r2 -d ./challenge1`” to open the file challenge1 using radare2. Then continue by using the “aa” command to analyse the system



The screenshot shows a terminal window with the command `r2 -d ./challenge1` running. The output indicates that the process with PID 1654 has started. The user then runs the `aa` command to analyze the assembly code at address `0x00400a30`. The analysis shows several warnings about block sizes exceeding the maximum allowed by the debugger.

```
elfmceager@tbfc-day-17:~$ r2 -d ./challenge1
Process with PID 1654 started ...
= attach 1654 1654
bin.baddr 0x00400000
Using 0x400000 about a specific feature, we could provide [REDACTED]
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
```

Using “`pdf @main`” you’ll be able to find the value of `local_ch` which is 1



The screenshot shows a terminal window with the command `[0x00400a30]> pdf @main` running. The output shows the assembly code for the `main` function, specifically the `sym.main` entry point. The assembly code includes instructions to push `rbp`, move `rbp` to `rsp`, and then move the value `1` into the memory location `[local_ch]`.

```
[0x00400a30]> pdf @main
Since the binary is not yet analyzed, first things to do is ask r2 to analyze the program, and this can be done by
/ (fcn) sym.main 35
|_ sym.main ();
|, this may take some time.
|; var int local_ch @ rbp-0xc stem.
|; var int local_8h @ rbp-0x8
analyses all symbols; var int local_4h @ rbp-0x4, in this case, involves extracting function
| r2 instructions are used; DATA-XREF from 0x00400a4d (entry0) information about the
| 0x00400b4d      55          push rbp
| 0x00400b4e      4889e5      mov rbp, rsp
| to understand more about this specific feature.
| 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]
| know where to look for more information - most likely have an entry point or a function
| 0x00400b66      8945fc      mov dword [local_4h], eax
| 0x00400b69      b800000000  mov eax, 0
| 0x00400b6e      5d          pop rbp
\ ret          0x00400b6f      c3          ret
```

Question 6

```
[0x00400a30]> pdf @main
Once the binary is open, the first things to do is ask r2 to analyze the program, and this can be done by
/ (fcn) sym.main 35
    sym.main ();
    ; this may take bytes: var sint local_ch@rbp-0xc item.
    ; var int local_8h @ rbp-0x8
    ; analyses all symbols; var int local_4h@rbp-0x4
    ; r2 instructions are useful; DATA XREF from 0x00400a4d (entry0) information about the
        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
```

By reading this, we can understand that we're moving local_ch = 1 into eax and then multiplying it with local_8h = 6. Hence giving us a value of 6

Question 7

```
[0x00400a30]> pdf @main
Once the binary is open, the first things to do is ask r2 to analyze the program, and this can be done by
/ (fcn) sym.main 35
    sym.main ();
    ; this may take bytes: var sint local_ch@rbp-0xc item.
    ; var int local_8h @ rbp-0x8
    ; analyses all symbols; var int local_4h@rbp-0x4
    ; r2 instructions are useful; DATA XREF from 0x00400a4d (entry0) information about the
        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
```

By observing this, it's understood that local_4h = 6 as it takes the value of eax.

Thought Process:

In doing this task, the first thing we did was to understand all the given commands and how it functions by following the guide given by THM to understand radare2. After receiving the challenge, what we first seeked to do was to connect to the machine_ip given. To do that we used "SSH" to login into the given machine_ip under Elf McEager. After inputting the details and logging in, we used radare2 to open the challenge1 file using the radare2 command "r2 -d ./challenge1". As soon as we got access, we just went ahead and analysed the whole system "aa". As to get the main details and find out more about the file, we used "pdf@main" to get the details of sym.main. After getting the details required, by reading the 3rd column we were able to retrieve the values of local_ch and local_8h as well as to discover other details on which details were moved into eax and how the values of eax changed step by step until it was set to the value 0.

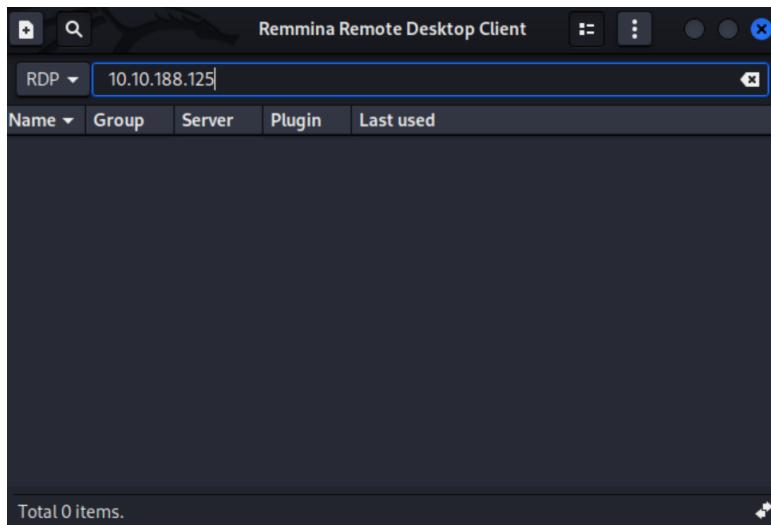
Day 18: The Bits of Christmas

Tools used : Remmina, ILSPy

Solution/Walkthrough

Question 1

After downloading Remmina (sudo apt install remmina), you should be able to open it and connect using the provided machine_ip.

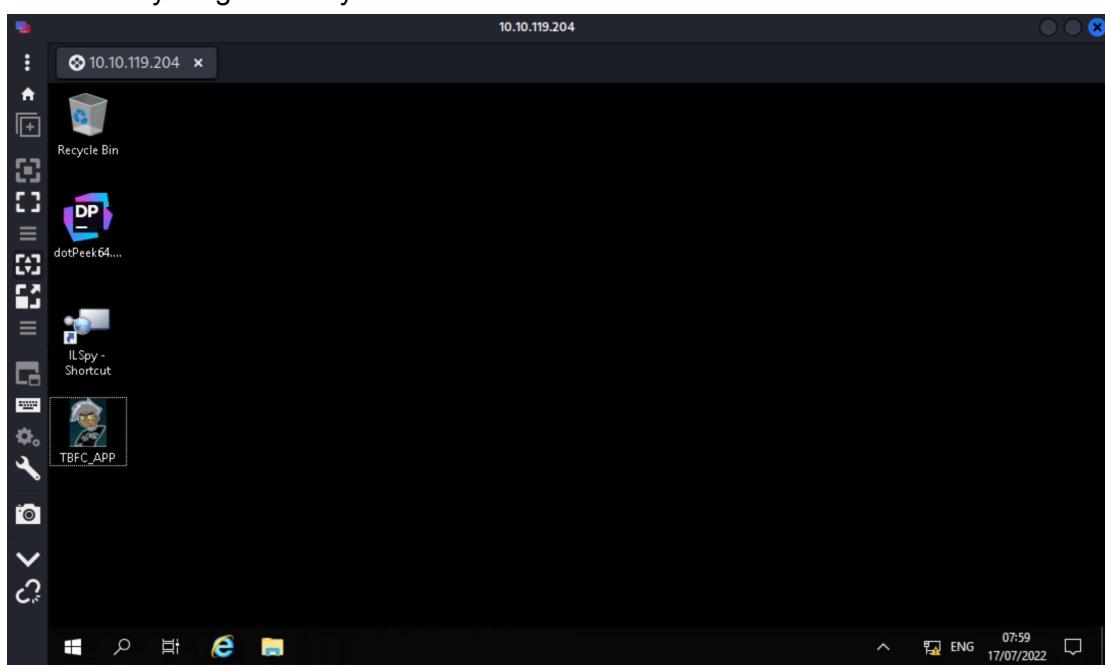


After filling in the provided credentials as follows :

Username : cmnatic

Password : Adventofcyber!

You'll shortly be greeted by this



Now to open the TBFC_APP and input the wrong password. Which will show this



The answer is “Uh Oh! That’s the wrong key”

Question 2

By using the all-powerful ability of *Observation*, you can tell TBFC stands for “The Best Festival Company”

Question 3

A screenshot of the IL Spy application interface. The title bar says "ILSpy". The menu bar includes "File", "View", "Window", and "Help". The toolbar has icons for file operations. The main window shows the assembly details for "TBFC_APP (0.0.0.0, .NETFramework, v4.6.1)". The left pane lists assembly components like "Metadata", "References", and "Resources". The right pane displays the assembly code:

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

using ...

[assembly: SecurityRules(SecurityRuleSet.Level1)]
[assembly: TargetFramework(".NETFramework,Version=v4.6.1", FrameworkDisplayName = ".NET Framework 4.6.1")]
[assembly: SecurityPermission(SecurityAction.RequestMinimum, SkipVerification = true)]
[assembly: AssemblyVersion("0.0.0.0")]

CrackMe
```

The module to have definitely caught the eye would be “CrackMe”

Question 4



Pretty obvious that we want to get into the nitty gritty “Main” details of the form itself rather “About” it

Question 5

```
private void MainForm_Load(object sender, EventArgs e)
{
    ...
}

private void buttonExit_Click(object sender, EventArgs e)
{
    ...
}

private void buttonAbout_Click(object sender, EventArgs e)
{
    ...
}

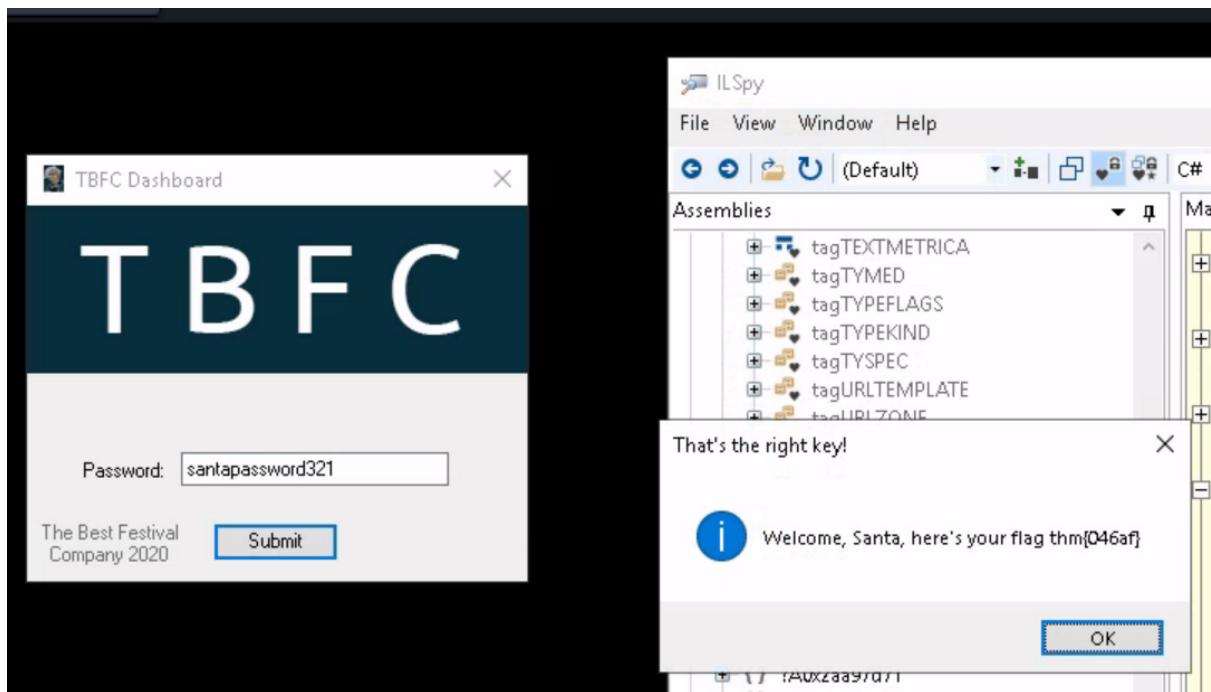
private unsafe void buttonActivate_Click(object sender, EventArgs e)
{
    IntPtr value = Marshal.StringToGlobalAnsi(textBoxKey.Text);
    sbyte* ptr = (sbyte*)System.Runtime.CompilerServices.Unsafe.AsPointer(ref <Module>._C@_0BB@IKKDFEPG@santapassword321@);
    void* ptr2 = (void*)value;
    byte b = *(byte*)ptr2;
    byte b2 = 115;
    if ((uint)b > 115u)
    {
        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;
                }
            }
        }
    }
}
```

After some snooping around, it slowly becomes clear where the important details can be found, particularly in “buttonActivate_Click” since how else would we “Activate” the application.

Question 6

```
IntPtr value = Marshal.StringToGlobalAnsi(textBoxKey.Text);
sbyte* ptr = (sbyte*)System.Runtime.CompilerServices.Unsafe.AsPointer(ref <Module>._C@_0BB@IKKDFEPG@santapassword321@);
void* ptr2 = (void*)value;
byte b = *(byte*)ptr2;
byte b2 = 115;
if ((uint)b >= 115u)
```

Looking closely, it's pretty easy to make up as to what the password is, so why not give it a shot in the TBFC app?



Sure enough, that turned out to be Santa's password !

Question 7

Refer to the screenshot above, the flag is thm{046af}

Thought Process :

First of all, would've been to open Remmina and access the given IP and fill in the details as the instructions follow. Shortly after, we tried using the TBFC app, however we do not know Santa's password. So instead we tried opening the TBFC app using ILSPY instead to look for anything which may give indicators. Sure enough, we find the "CrackMe" section sticking out like a sore thumb. Choosing between "AboutForm" and the "MainForm" it's pretty obvious as to which contains the data we're looking for. Now looking at the main form, we look back as to what we could possibly need to be able to access into the TBFC app itself, we continue to look into the different sections, we notice that "buttonActivate_Click" contained some familiar details such as the "You're not Santa!" message box. Sure enough, it looked pretty clear as to what may be the password which we soon after found out was "santapassword321" and heck the flag was also in there : thm{046af}.

Day 19: The Naughty or Nice List

Tools used: Kali Linux, Wfuzz, CyberChef

Solution/Walkthrough

Question 1

After making a text file called naughty.txt and putting in the values given to us on google forms we used Wfuzz to iterate through the values to determine who is naughty and nice.

| ID | Response | Lines | Word | Chars | Payload |
|------------|----------|-------|-------|---------|------------|
| 000000001: | 200 | 168 L | 431 W | 5439 Ch | "Timothy" |
| 000000003: | 200 | 178 L | 460 W | 5711 Ch | "Ian Chai" |
| 000000004: | 200 | 168 L | 431 W | 5431 Ch | "YP" |
| 000000006: | 200 | 168 L | 431 W | 5437 Ch | "Tib3rius" |
| 000000002: | 200 | 168 L | 431 W | 5437 Ch | "Kanes" |
| 000000005: | 200 | 168 L | 431 W | 5434 Ch | "JJ" |

Question 2

Next we entered a name into the form and clicked the search button. This will give us a url. We can then decode this url by using CyberChef. We then used this new url to try and access the root of the website. This returns us with a Not Found error.

The List

Welcome children!

To find out if you are currently on the naughty list or the nice list, please enter your name below!

Have a Merry Christmas! Ho ho ho!

- Santa

Name:

name is on the Nice List.

Admin

The List

Welcome children!

To find out if you are currently on the naughty list or the nice list, please enter your name below!

Have a Merry Christmas! Ho ho ho!

- Santa

Name:

Not Found

The requested URL was not found on this server.

Question 3

We changed the port to 80 and it returned us with a connection refused error.



Welcome children!

To find out if you are currently on the naughty list or the nice list, please enter your name below!

Have a Merry Christmas! Ho ho ho!

- Santa

Name:

Failed to connect to list.hohoho port 80: Connection refused

Admin

Question 4

Next we changed the port to 22 and it returned us with a Recv failure.



Welcome children!

To find out if you are currently on the naughty list or the nice list, please enter your name below!

Have a Merry Christmas! Ho ho ho!

- Santa

Name:

Recv failure: Connection reset by peer

Admin

Question 5

Next we replaced the list.hohoho hostname with "localhost". This time, it shows us that our search has been blocked by their security team.



Welcome children!

To find out if you are currently on the naughty list or the nice list, please enter your name below!

Have a Merry Christmas! Ho ho ho!

- Santa

Name:

Your search has been blocked by our security team.

Question 6

We can however bypass this by using localtest.me, which resolves every subdomain to 127.0.0.1. This will return a message which will give us Santa's password which is "Be good for goodness sake!".



Welcome children!

To find out if you are currently on the naughty list or the nice list, please enter your name below!

Have a Merry Christmas! Ho ho ho!

- Santa

Name:

Santa,

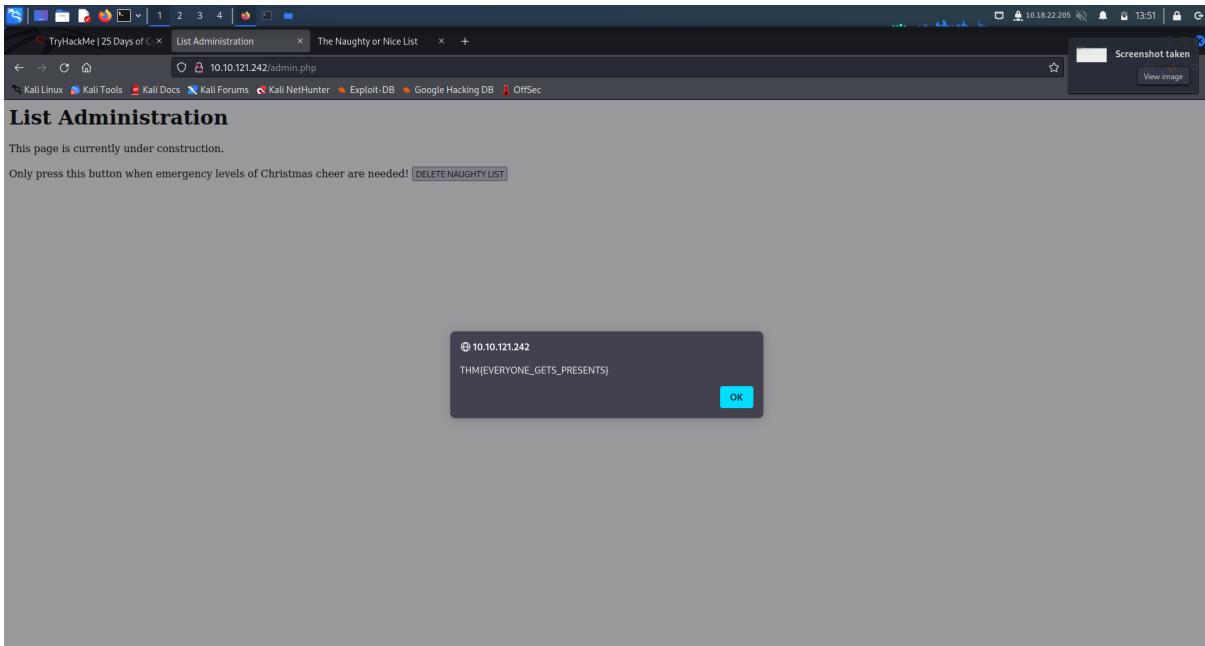
If you need to make any changes to the Naughty or Nice list, you need to login.

I know you have trouble remembering your password so here it is: Be good for goodness sake!

- Elf McSkidy

Question 7

After logging in to Santa's account we can delete the naughty list. This will return to us the challenge flag.



Thought process/ Methodology

From the url we got after pressing the search button, we know the hostname is list.hohoho. But since this is not a valid hostname on the internet, it most likely refers to a backend machine. We can try to exploit this by using Server-Side Request Forgery (SSRF). The first thing we did was to fetch the root of the same site. We were greeted with a "url was not found on the server" message. This indicates that we were able to make the server request the modified URL and return the response. Next we changed the port number from 8080 to 80. The message now changes to "Failed to connect to list.hohoho port 80: Connection refused" which suggests that port 80 is not open on list.hohoho. Next we changed the port number to 22. 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 as it was a http request. Next we tried to access services running locally on the server. We did this by replacing the list.hohoho hostname with "localhost". This will return a message that shows that our search has been blocked by the security team. We can however bypass this by using localtest.me, which resolves every subdomain to 127.0.0.1. This returned us with the password to Santa's account which can then be used to log in as admin and delete the naughty list. The flag will then be shown to us.

Day 20: BlueTeaming - PowershellELF to the rescue

Tools used: Linux, Powershell, SSH

Solution/walkthrough:

Question 1

“man ssh” and scrolling down shows us the function of -l flag

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

Question 2

SSH into the machine with credentials provided

```
(1211102272㉿kali)-[~]
└─$ ssh -l mceager 10.10.213.169

The authenticity of host '10.10.213.169 (10.10.213.169)' can't be established.
ED25519 key fingerprint is SHA256:X2ViBkLLQoHmAsXFoem36jkL9faKH+Fr2lt2dd/kIWY.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:14: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.213.169' (ED25519) to the list of known hosts.
mceager@10.10.213.169's password: ┌
```

Initiate powershell

```
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

mceager@ELFSTATION1 C:\Users\mceager>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\mceager> ┌
```

Change directory to /Documents and Get-ChildItem -Hidden -File to find the hidden file

```

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

    Directory: C:\Users\mceager\Documents

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>

```

Use "Get-Content ./e1fone.txt" to get the text

```

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

```

Question 3

Set-Location to /desktop

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

Find the hidden folder and "Set-Location" to it

```

PS C:\Users\mceager\Desktop> Get-ChildItem -Hidden

    Directory: C:\Users\mceager\Desktop

Mode                LastWriteTime         Length Name
--                -- -- -- -- -- -- -- -- -- -- -- -- --
d--h--          12/7/2020 11:26 AM           160 elf2wo
-a-hs-          12/7/2020 10:29 AM           282 desktop.ini

PS C:\Users\mceager\Desktop> Set-Location .\elf2wo\

```

Find the file and output the contents

```

PS C:\Users\mceager\Desktop\elf2wo> Get-ChildItem

    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

Change to windows directory

```

PS C:\Users\mceager\Desktop\elf2wo> Set-Location ..
PS C:\Users\mceager\Desktop\elf2wo> Set-Location ..
PS C:\Users\mceager\Desktop> Set-Location ..

S C:\> Set-location .\Windows\

```

```

PS C:\Windows> Set-Location .\System32\
PS C:\Windows\System32> Get-ChildItem

```

As a result of a too cluttered folder, we used ‘Filter ‘*3*’ to find the hidden directory

```

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

    Directory: C:\Windows\System32

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

```

Question 5

Find the word count of the file by using “Measure-Object” along with the pipe command

```

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

    Directory: C:\Windows\System32\3lfthr3e

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

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

Count      : 9999
Average    :
Sum        :
Maximum    :
Minimum    :
Property   :

```

Question 6

Find the 2 words in the file by piping the output using “Select-Object” with the “-Index” flag set to 551 and 6991 respectively

```

PS C:\Windows\System32\3lfthr3e> Get-Content .\1.txt | Select-Object -Index 551
Red
PS C:\Windows\System32\3lfthr3e> Get-Content .\1.txt | Select-Object -Index 6991
Ryder

```

Question 7

Find the aligning phrase from the last question using the string “redryder”

```

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

```

Methodology/ thought process:

First, we used “man ssh” to show to manual pages, then slowly scrolled down until we saw the function of -l flag which is “login_name”. Then, we use SSH to access the machine, then initiate the powershell to perform our operations, we then find the hidden directory named “e1fone.txt” then output the contents, revealing the answer, this was confusing as another file “elfone.txt”, also exists which wasted some time.. After that, we changed directories to /Desktop to find the hidden file, then also output the contents. The next step was to switch to

/Windows/System32/ , as the Directory was too cluttered, we used “filter ‘*3*’” to find the hidden directory, after that we used “measure-object” to find out out the word count. To index the file we used “Select-Object -Index” on positions 551 and 6991, which is “red” and “ryder”, to find the aligning phrase, we piped the “Select-String ‘redryder’” into Get-Content and got “redryderbbgun”.