

PSP0201

Week 5

Writeup

Group Name : Fsociety

Members :

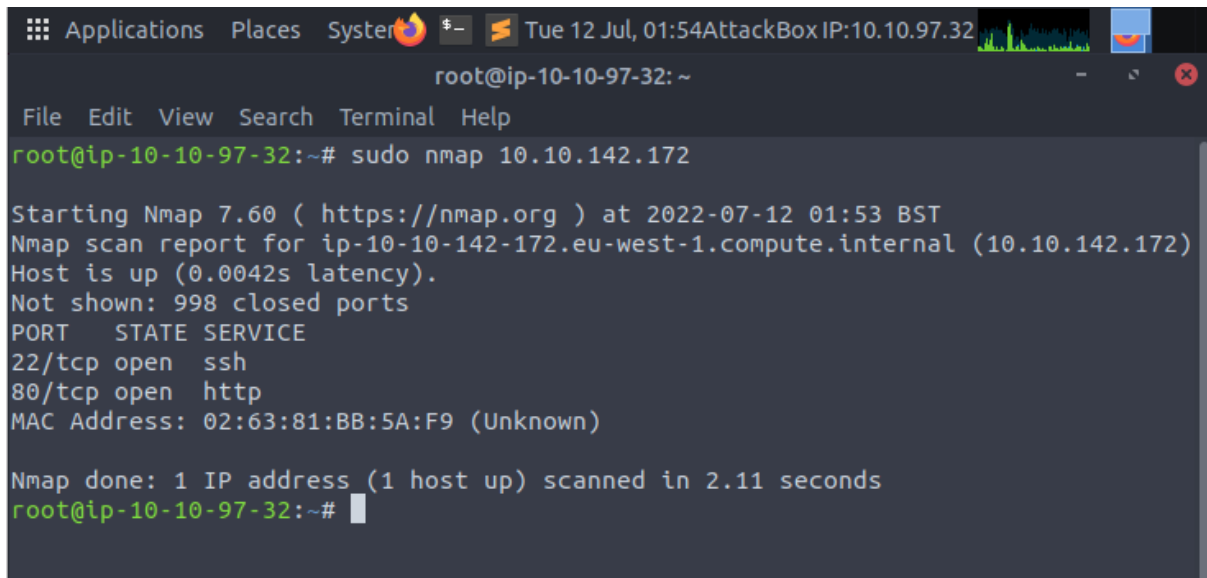
ID	Name	Role
1211102908	Wan Muhammad Ilhan Bin Wan Zil Azhar	Leader
1211101583	Luqman Hakim Bin Noorazmi	Member
1211203101	Jazlan Zuhair Bin Mohamed Zafrualam	Member
1211102054	Mithesh Kumar	Member

Day 16
(Help! Where is Santa?)

Tools used: Firefox, Terminal

Question 1:

We used nmap to find the port.



```
root@ip-10-10-97-32: ~
File Edit View Search Terminal Help
root@ip-10-10-97-32:~# sudo nmap 10.10.142.172

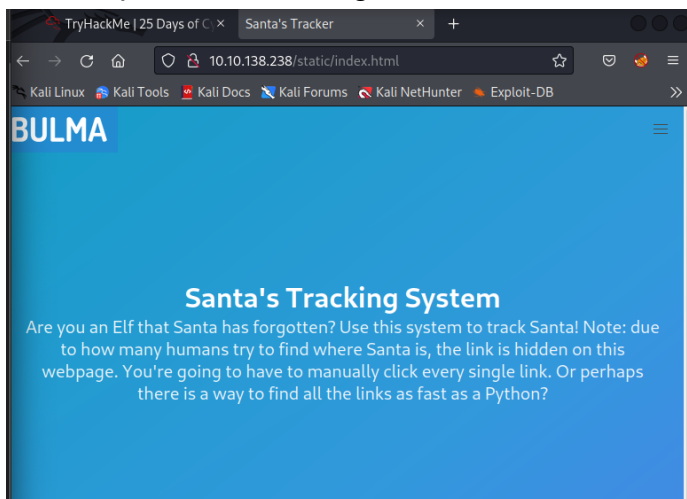
Starting Nmap 7.60 ( https://nmap.org ) at 2022-07-12 01:53 BST
Nmap scan report for ip-10-10-142-172.eu-west-1.compute.internal (10.10.142.172)
Host is up (0.0042s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 02:63:81:BB:5A:F9 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 2.11 seconds
root@ip-10-10-97-32:~#
```

The port number for the web server is 80

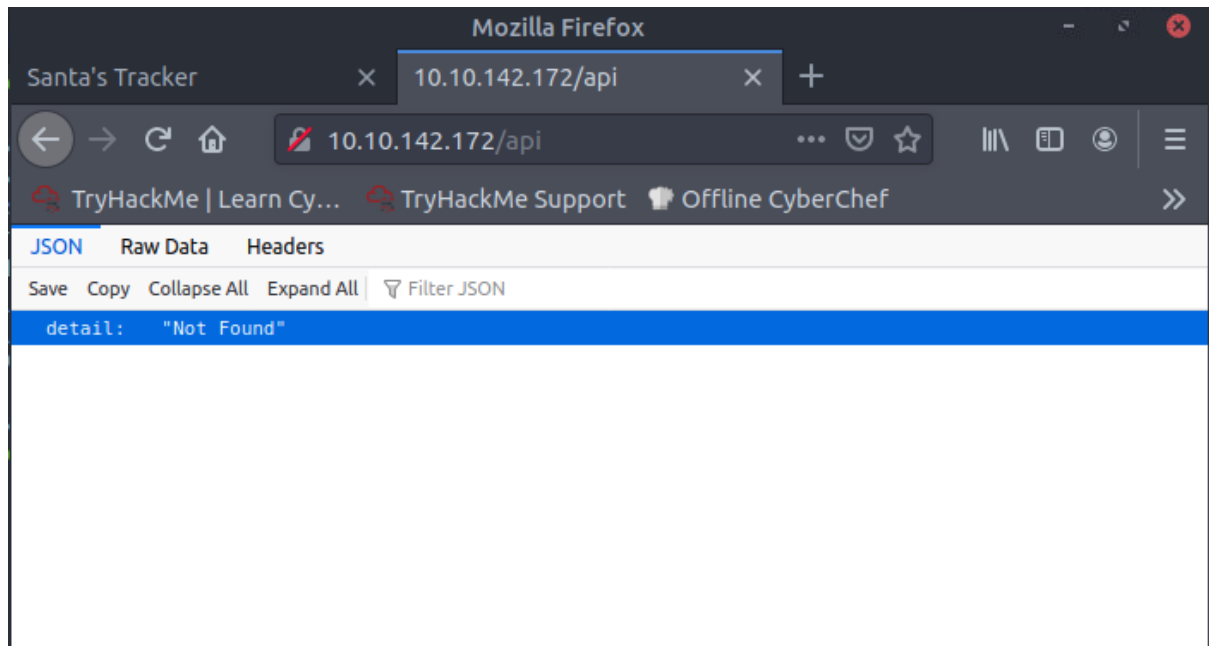
Question 2:

The template that is being used is BULMA.



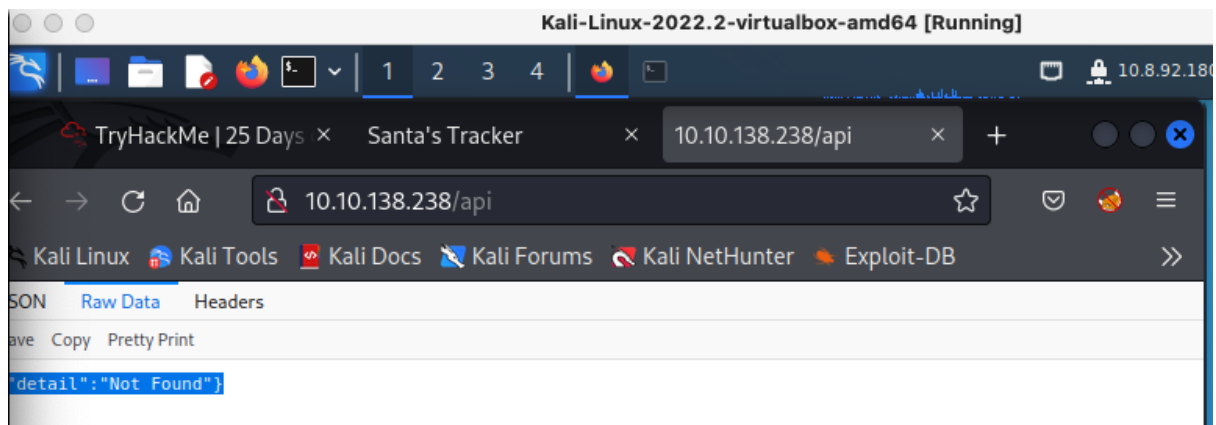
Question 3:

The directory for the API is */api*



Question 4:

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



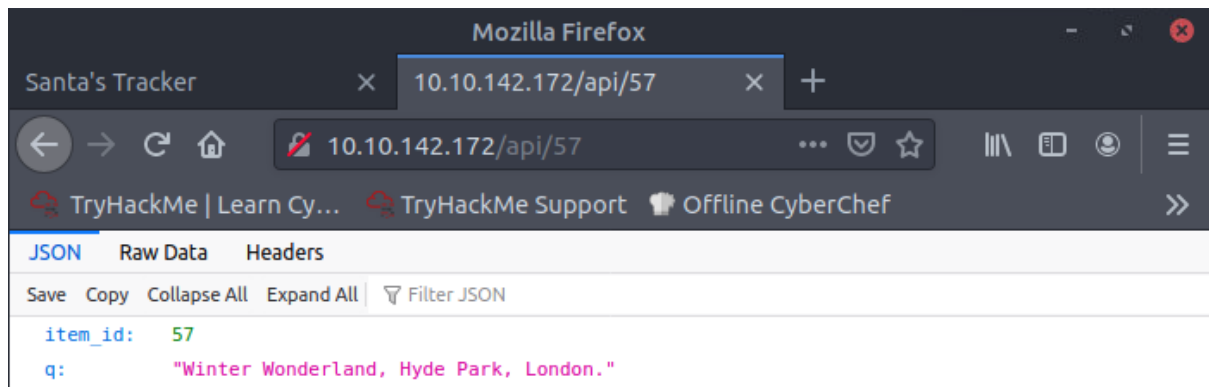
Question 5:

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

```
JSON  Raw Data  Headers
Save Copy Collapse All Expand All Filter JSON
{
  "item_id": 57,
  "q": "Winter Wonderland, Hyde Park, London."
}
```

Question 6:

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



The screenshot shows a Mozilla Firefox browser window with the address bar displaying `10.10.142.172/api/57`. The page content shows the JSON response from the Santa's Tracker API, which is identical to the one in Question 5:

```
JSON  Raw Data  Headers
Save Copy Collapse All Expand All Filter JSON
{
  "item_id": 57,
  "q": "Winter Wonderland, Hyde Park, London."
}
```

Thought Process / Methodology:

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

Day 17
(ReverseELFneering)

Tools used: Firefox, Terminal, Kali, Radare2

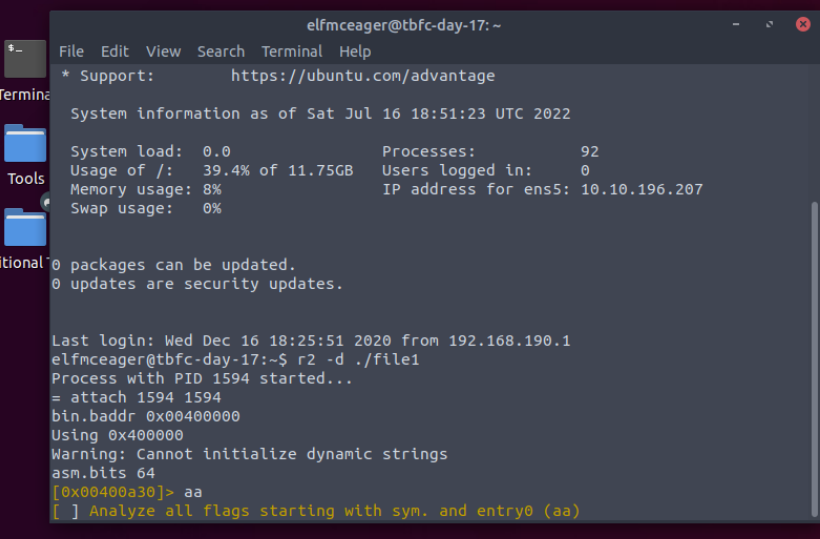
Question 1:

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

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

Question 2:

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



```
elfmceager@tbfc-day-17:~  
File Edit View Search Terminal Help  
* Support: https://ubuntu.com/advantage  
System information as of Sat Jul 16 18:51:23 UTC 2022  
System load: 0.0 Processes: 92  
Usage of /: 39.4% of 11.75GB Users logged in: 0  
Memory usage: 8% IP address for ens5: 10.10.196.207  
Swap usage: 0%  
0 packages can be updated.  
0 updates are security updates.  
Last login: Wed Dec 16 18:25:51 2020 from 192.168.190.1  
elfmceager@tbfc-day-17:~$ r2 -d ./file1  
Process with PID 1594 started...  
= attach 1594 1594  
bin.baddr 0x00400000  
Using 0x400000  
Warning: Cannot initialize dynamic strings  
asm.bits 64  
[0x00400a30]> aa  
[ ] Analyze all flags starting with sym. and entry0 (aa)
```

Question 3:

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

```
[0x00400a30]> db 0x00400b55
[0x00400a30]> pdf @main
;-- main:
/ (fcn) sym.main 68
|   sym.main ();
|   ; var int local_ch @ rbp-0xc
|   ; var int local_8h @ rbp-0x8
|   ; var int local_4h @ rbp-0x4
|       ; DATA XREF from 0x00400a4d
|   0x00400b4d      55
|   0x00400b4e      4889e5
|   0x00400b51      4883ec10
|   0x00400b55 b    c745f4040000.
|   0x00400b5c      c745f8050000.
|   0x00400b63      8b55f4
```

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

Question 4:

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

```
[0x00400a30]> dc
hit breakpoint at: 400b55
[0x00400b55]> 
```

Question 5:

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

```
elfmceager@tbfc-day-17: ~
File Actions Edit View Help
This host key is known by the following other names/addresses:
~/.ssh/known_hosts:7: [hashed name]
~/.ssh/known_hosts:9: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.222.218' (ED25519) to the list of known hosts.
elfmceager@10.10.222.218'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 12:19:27 UTC 2022

System load:  0.15          Processes:      99
Usage of /:   39.4% of 11.75GB Users logged in: 0
Memory usage: 8%           IP address for ens5: 10.10.222.218
Swap usage:  0%

0 packages can be updated.
0 updates are security updates.

Last login: Wed Dec 16 18:25:51 2020 from 192.168.190.1
elfmceager@tbfc-day-17:~$ 
```

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

```
Last login: Wed Dec 16 18:25:51 2020 from 192.168.190.1
elfmceager@tbfc-day-17:~$ ls
challenge1  file1
```

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

```
elfmceager@tbfc-day-17:~$ r2 -d ./file1
Process with PID 1585 started...
= attach 1585 1585
bin.baddr 0x00400000
Using 0x400000
Warning: Cannot initialize dynamic strings
asm.bits 64
[0x00400a30]> 
```

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

```
[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)
[0x00400a30]> afl | grep main
0x00400b4d 1 68 sym.main
0x00400e10 10 1007 → 219 sym.__libc_start_main
0x00403870 39 661 → 629 sym._nl_find_domain
0x00403b10 308 5366 → 5301 sym._nl_load_domain
0x00415fe0 1 43 sym._IO_switch_to_main_get_area
0x0044cf00 1 8 sym._dl_get_dl_main_map
0x00470520 1 49 sym._IO_switch_to_main_wget_area
0x0048fae0 7 73 → 69 sym._nl_finddomain_subfreeres
0x0048fb30 16 247 → 237 sym._nl_unload_domain
[0x00400a30]> 
```

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

```
[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 8b45f4 mov eax, dword [local_ch]
|   0x00400b62 0faf45f8acting 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]> db 0x00400b51
```


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

```
[0x00400a30]> dc
hit breakpoint at: 400b51
[0x00400b51]> px @ rbp-0xc
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x7ffd31f5aa74 0000 0000 1890 6b00 0000 0000 0000 4018 4000 .....k.....@.
0x7ffd31f5aa84 0000 0000 e910 4000 0000 0000 0000 0000 0000 .....@.....
0x7ffd31f5aa94 0000 0000 0000 0000 0100 0000 a8ab f531 .....1.....
0x7ffd31f5aaa4 fd7f 0000 4d0b 4000 0000 0000 0000 0000 ...M.@.....
0x7ffd31f5aab4 0000 0000 1700 0000 0100 0000 0000 0000 .....
0x7ffd31f5aac4 0000 0000 0000 0000 0200 0000 0000 0000 .....
0x7ffd31f5aad4 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffd31f5aae4 0000 0000 0000 0000 0000 0000 0004 4000 .....@.
0x7ffd31f5aaf4 0000 0000 3317 93cd 3c9d ad08 e018 4000 ....3...<.....@.
0x7ffd31f5ab04 0000 0000 0000 0000 0000 0000 1890 6b00 .....k.
0x7ffd31f5ab14 0000 0000 0000 0000 0000 0000 3317 33a8 .....3.3.
0x7ffd31f5ab24 57fe 57f7 3317 27dc 3c9d ad08 0000 0000 W.W.3.'.<...
0x7ffd31f5ab34 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffd31f5ab44 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffd31f5ab54 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffd31f5ab64 0000 0000 0000 0000 0000 0000 0000 0000 .....
[0x00400b51]>
```

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

```
[0x00400b51]> ds
[0x00400b51]> px @ rbp-0xc
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x7ffc01e7b874 0100 0000 1890 6b00 0000 0000 4018 4000 .....k.....@.
0x7ffc01e7b884 0000 0000 e910 4000 0000 0000 0000 0000 .....@.....
0x7ffc01e7b894 0000 0000 0000 0000 0100 0000 a8b9 e701 .....
0x7ffc01e7b8a4 fc7f 0000 4d0b 4000 0000 0000 0000 0000 ...M.@.....
0x7ffc01e7b8b4 0000 0000 1700 0000 0100 0000 0000 0000 .....
0x7ffc01e7b8c4 0000 0000 0000 0000 0200 0000 0000 0000 .....
0x7ffc01e7b8d4 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffc01e7b8e4 0000 0000 0000 0000 0000 0000 0004 4000 .....@.
0x7ffc01e7b8f4 0000 0000 7d02 56fa dccb 33da e018 4000 .....}.V...3...@.
0x7ffc01e7b904 0000 0000 0000 0000 0000 0000 1890 6b00 .....k.
0x7ffc01e7b914 0000 0000 0000 0000 0000 0000 7d02 f6bb .....}...
0x7ffc01e7b924 93c8 cb25 7d02 e2eb dccb 33da 0000 0000 ...%}.....3.
0x7ffc01e7b934 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffc01e7b944 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffc01e7b954 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffc01e7b964 0000 0000 0000 0000 0000 0000 0000 0000 .....
```

Question 6:

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

```
(fcn) sym.main 35
| sym.main ();
| ; var int local_ch @ rbp-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
| ;-- rip:
| 0x00400b51 c745f4010000. mov dword [local_ch], 1
| 0x00400b58 c745f8060000. mov dword [local_8h], 6
| 0x00400b5f 8b45f4 mov eax, dword [local_ch]
| 0x00400b62 0faf45f8 imul eax, dword [local_8h]
| 0x00400b66 8945fc mov dword [local_4h], eax
| 0x00400b69 b800000000 mov eax, 0
| 0x00400b6e 5d pop rbp
| 0x00400b6f c3 ret
[0x00400b51]> db 0x00400b62
[0x00400b51]> dc
hit breakpoint at: 400b62
```

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

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

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

```
[0x00400b51]> dr
rax = 0x00000006
rbx = 0x00400400
rcx = 0x0044b9a0
rdx = 0x7fffbac0be98
r8 = 0x01000000
r9 = 0x006bb8e0
r10 = 0x00000015
r11 = 0x00000000
r12 = 0x004018e0
r13 = 0x00000000
r14 = 0x006b9018
r15 = 0x00000000
rsi = 0x7fffbac0be88
rdi = 0x00000001
rsp = 0x7fffbac0bd60
rbp = 0x7fffbac0bd60
rip = 0x00400b66
rflags = 0x00000246
orax = 0xffffffffffffffff
```

Question 7:

First, set a breakpoint at the `local_4h` line.

```
[0x00400b51]> pdf @main
gccmainre1  ;-- main: 10.10.222.218 56m 15s
/ (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 b0faf45f8  imul eax, dword [local_8h]
;-- rip:
0x00400b66 8945fc      mov dword [local_4h], eax
0x00400b69 b800000000  mov eax, 0
0x00400b6e 5d          pop rbp
0x00400b6f c3          ret
[0x00400b51]> db 0x00400b66
```

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

```
[0x00400b51]> dc
hit breakpoint at: 400b66
[0x00400b51]> ds
```

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

```
[0x00400b51]> px @ rbp-0x4
- offset -      0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x7ffffbac0bd5c 0600 0000 4018 4000 0000 0000 e910 4000 . ... @.@.....@.
0x7ffffbac0bd6c 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffffbac0bd7c 0100 0000 88be c0ba ff7f 0000 4d0b 4000 . .... M.@.
0x7ffffbac0bd8c 0000 0000 0000 0000 0000 0000 1700 0000 .....
0x7ffffbac0bd9c 0100 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffffbac0bdac 0200 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffffbac0bdbc 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffffbac0bdcc 0000 0000 0004 4000 0000 0000 4aa1 7a7a .....@.....J.zz
0x7ffffbac0bddc 8184 9eaf e018 4000 0000 0000 0000 0000 .....@.....
0x7ffffbac0bdec 0000 0000 1890 6b00 0000 0000 0000 0000 .....k.....
0x7ffffbac0bdfc 0000 0000 4aa1 1a30 80f1 6150 4aa1 ce6b ....J..0..aPJ..k
0x7ffffbac0be0c 8184 9eaf 0000 0000 0000 0000 0000 0000 .....
0x7ffffbac0be1c 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffffbac0be2c 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffffbac0be3c 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x7ffffbac0be4c 0000 0000 0000 0000 0000 0000 0000 0000 .....
```

Thought Process/Methodology :

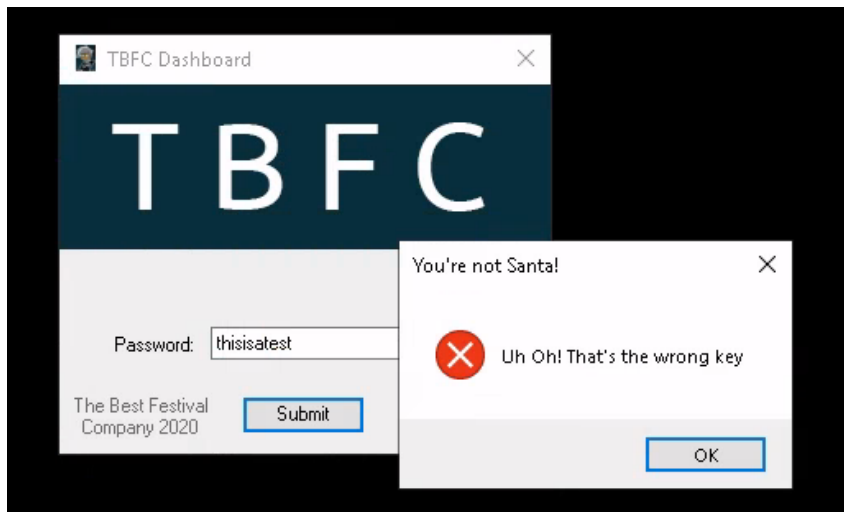
Reading through the radare2 guidelines and commands, I was struggling to understand the functions at first, it is harder than it looks. But the main commands which consisted of db, dc, ds, px and pdf are easier to understand. For the first question, we looked through the information table which reveals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

Day 18
(The Bits of Christmas)

Tools used: Firefox, Terminal, Kali, Remmina, ILSpy

Question 1:

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



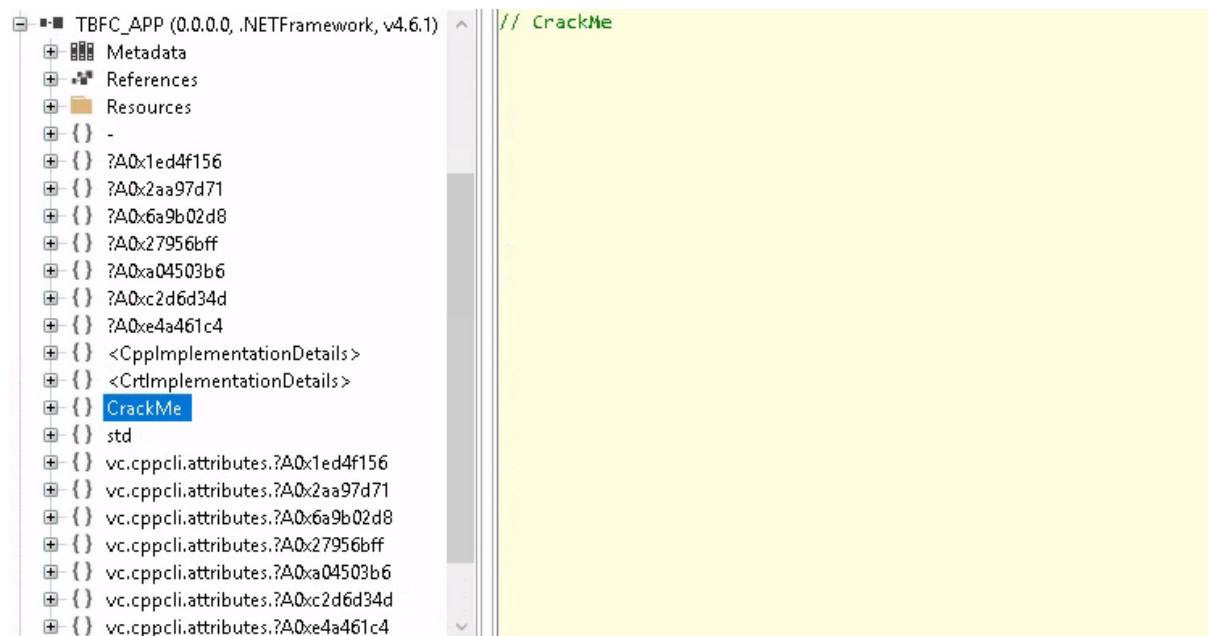
Question 2:

TBFC stands for: *The Best Festival Company*



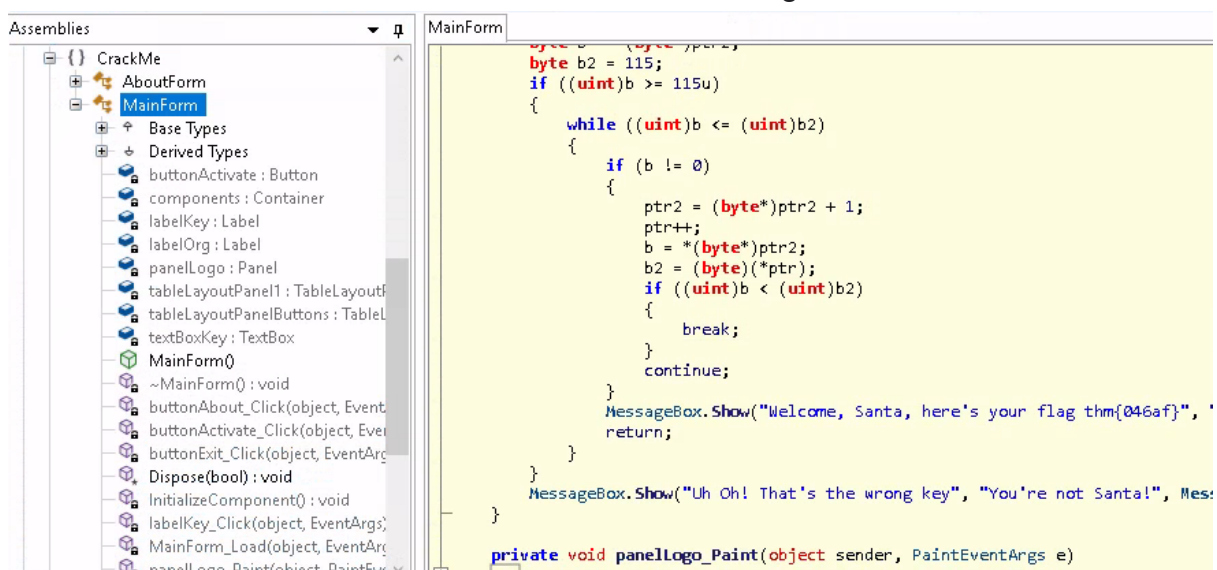
Question 3:

The module that catches my attention is: *CrackMe*



Question 4:

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



Question 5:

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

Question 6:

What is Santa's password?: *santapassword321*

```
buttonActivate_Click(object, EventArgs) : void
+
)
-
ointer(ref <Module>._?_C@_0BB@IKKDFEPG@santapassword321@);
```

Question 7:

The flag is: *thm{046af}*

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

Day 19
(The Naughty or Nice List)

Tools used: Kali, Firefox

Question 1:

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

Name:

Search

Timothy is on the Naughty List.

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

Question 2:

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



Not Found

The requested URL was not found on this server.

Question 3:

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



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

Question 4:

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



Recv failure: Connection reset by peer

Question 5:

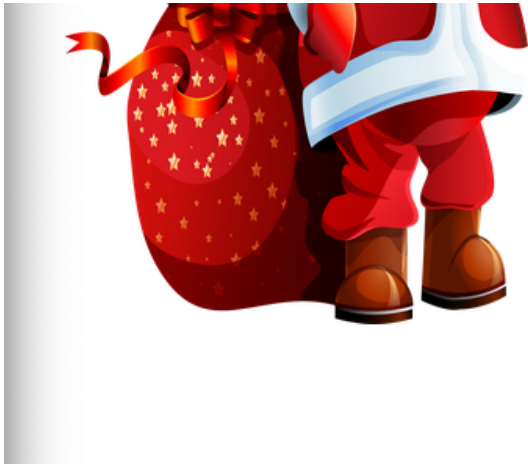
When using `/?proxy=http%3A%2F%2Flocalhost` on the URL, it will display *'Your search has been blocked by our security team'*.



Your search has been blocked by our
security team.

Question 6:

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



Santa,

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

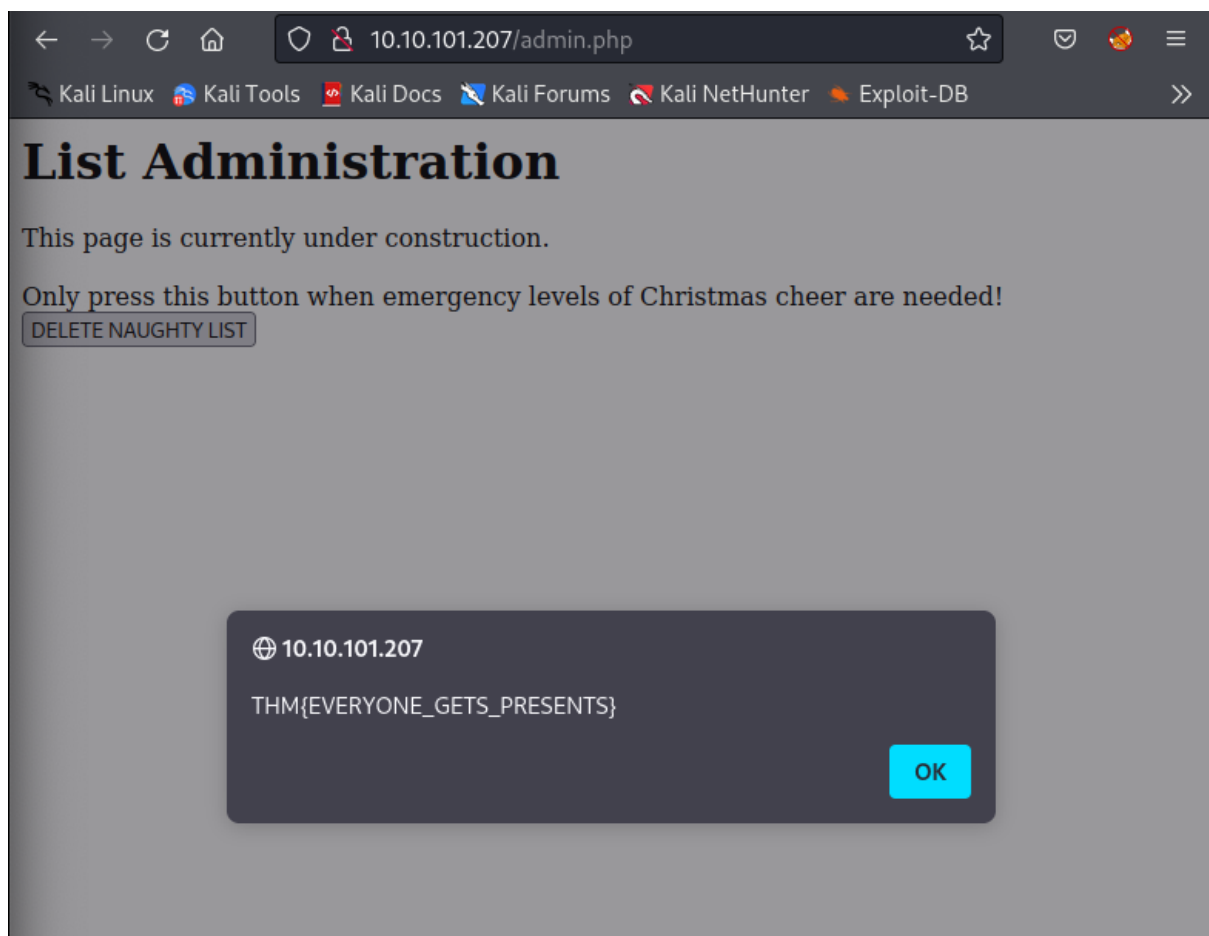
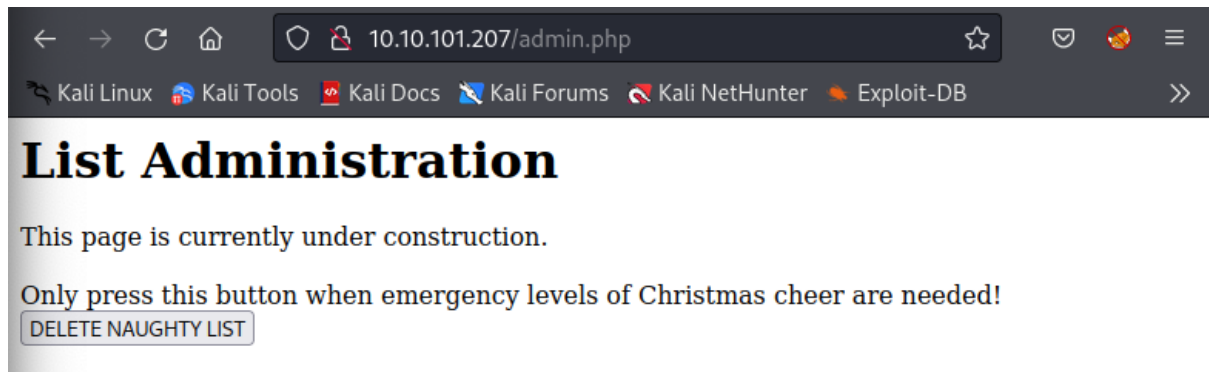
I know you have trouble remembering your
password so here it is: Be good for
goodness sake!

- Elf McSkidy



Question 7:

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



Thought Process/Methodology :

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

Day 20
(Powershell to the rescue?)

Tools used: Firefox, Terminal, Kali, Google

Question 1:

The -l command is used to specify the *login name*.

(Via <https://explainshell.com/explain?cmd=ssh+-L+-N+-f+-l>)

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

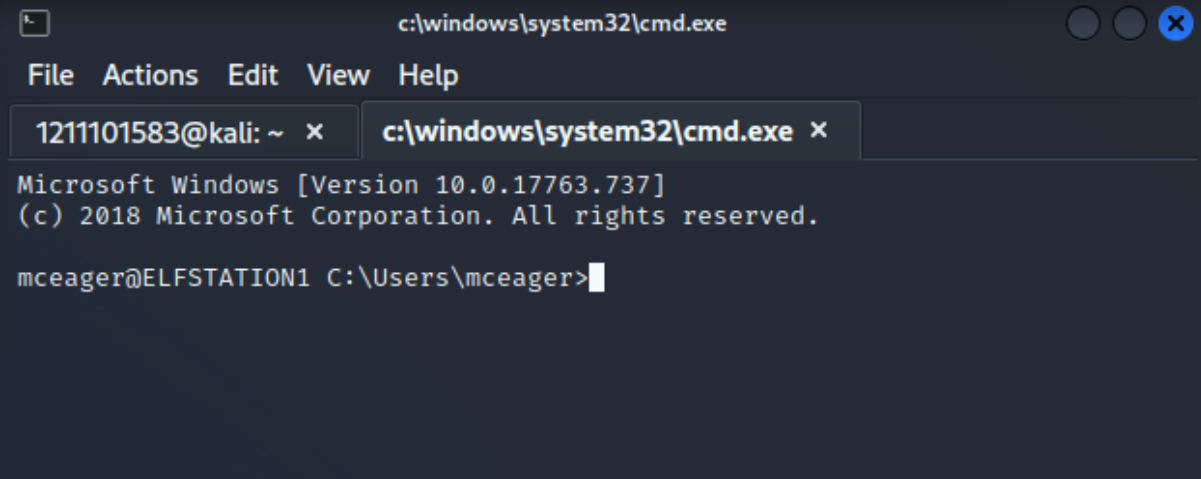
Question 2:

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

```
ssh -l mceager <Machine_IP>
```

When prompted for password, type in *r0ckStar!*

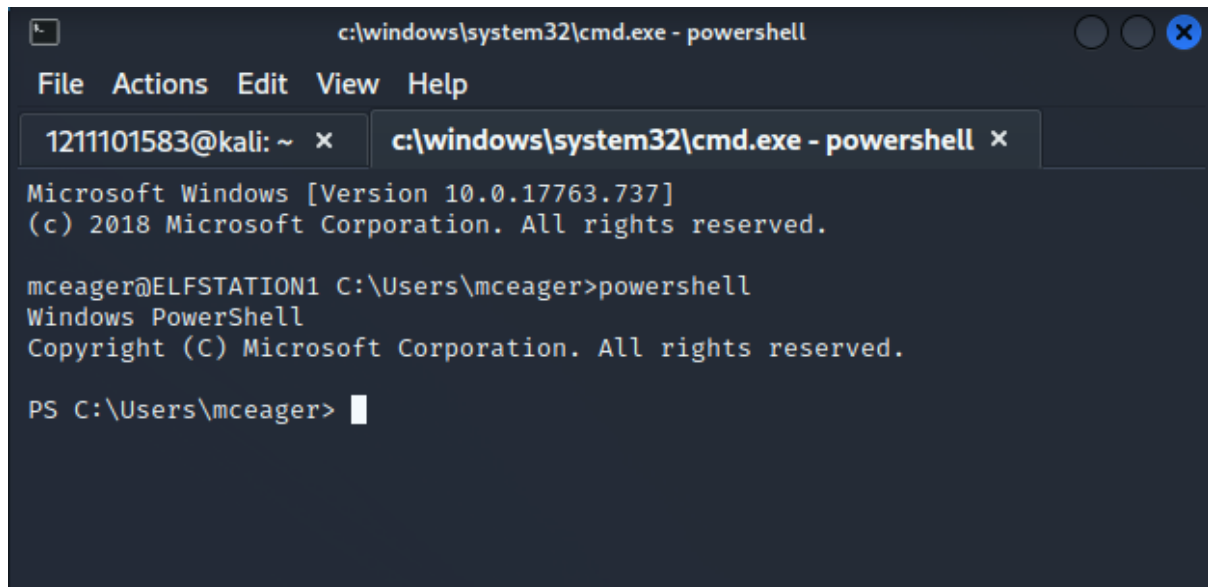
This window will appear when the login is successful



The screenshot shows a Windows command prompt window titled "c:\windows\system32\cmd.exe". The window has a menu bar with "File", "Actions", "Edit", "View", and "Help". Below the menu bar, there are two tabs: "1211101583@kali: ~" and "c:\windows\system32\cmd.exe". The main content of the window displays the following text:

```
Microsoft Windows [Version 10.0.17763.737]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
mceager@ELFSTATION1 C:\Users\mceager>
```

Type in *powershell*

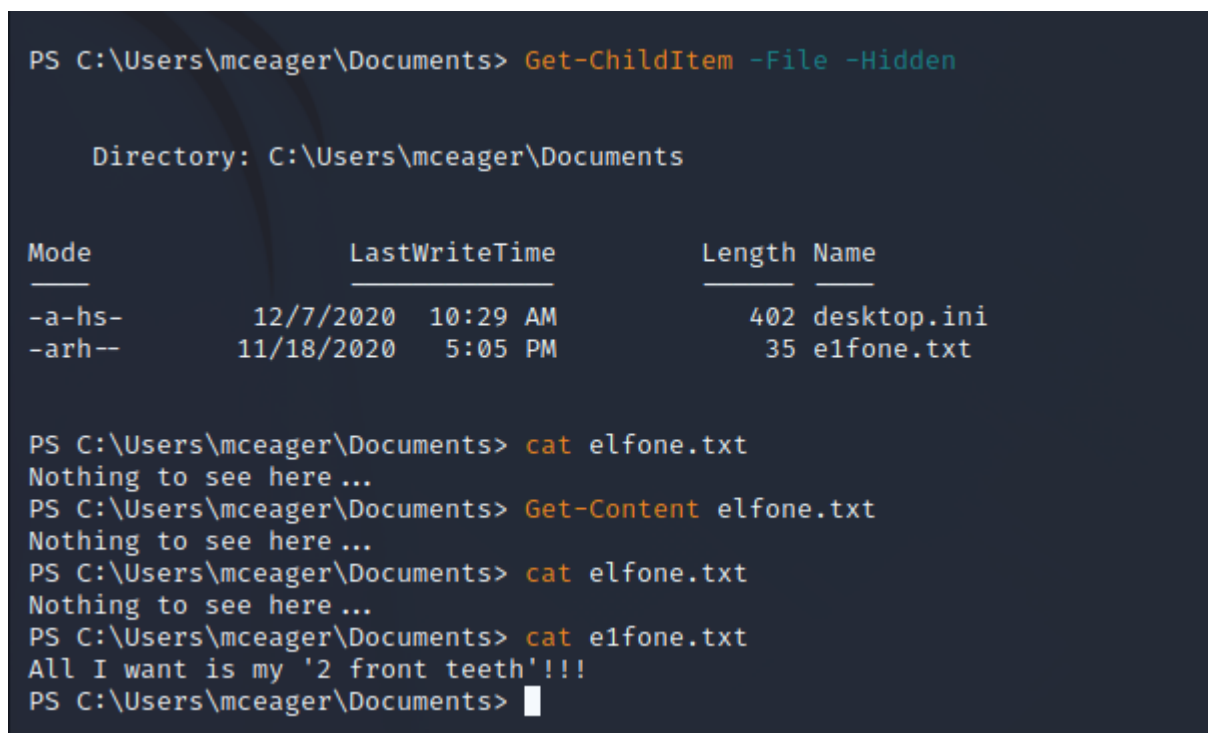


```
c:\windows\system32\cmd.exe - powershell
File Actions Edit View Help
1211101583@kali: ~ x c:\windows\system32\cmd.exe - powershell x
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

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

PS C:\Users\mceager> 
```

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



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

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> cat e1fone.txt
Nothing to see here ...
PS C:\Users\mceager\Documents> Get-Content e1fone.txt
Nothing to see here ...
PS C:\Users\mceager\Documents> cat e1fone.txt
Nothing to see here ...
PS C:\Users\mceager\Documents> cat e1fone.txt
All I want is my '2 front teeth'!!!
PS C:\Users\mceager\Documents> 
```


Question 3:

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

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

```
Directory: C:\Users\mceager\Desktop
```

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

```
PS C:\Users\mceager\Desktop> cat elf2wo
```

```
cat : Access to the path 'C:\Users\mceager\Desktop\elf2wo' is denied.
```

```
At line:1 char:1
```

```
+ cat elf2wo
```

```
+ ~~~~~
```

```
+ CategoryInfo          : PermissionDenied: (C:\Users\mceager\Desktop\elf2wo:String) [Get-Content], UnauthorizedAccessException
```

```
+ FullyQualifiedErrorId : GetContentReaderUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetContentCommand
```

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

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

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

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

```
PS C:\Users\mceager\Desktop\elf2wo> cat e70smsW10Y4k.txt
```

```
I want the movie Scrooged <3!
```

```
PS C:\Users\mceager\Desktop\elf2wo> █
```

Question 4:

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

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

Directory: C:\Windows\System32

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

Question 5:

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

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

Directory: C:\Windows\System32\3lfthr3e

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

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

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

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

Question 6:

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

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

Question 7:

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

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

Thought Process / Methodology:

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