

PSP0201

WEEKLY WRITE UP

WEEK 5

GROUP 7

Group Name : Sang Haekeo (The Hackers)

Sang

- Taken from a Malay word , [sang](#) , meaning 'the'

Haekeo

- Taken from a Korean word , [해커](#) , meaning 'hacker'

ID	NAME	ROLE
1211102162	AMILIA NADZEERA BINTI BAHARUDIN	Leader
1211100930	KU NAJWA SYAUQINA BINTI KU AZRIN	Member
1211101693	SAVITHA MURUGUMUNISEGARAN	Member

Day 16: Scripting ; Help! Where is Santa?

Question 1: What is the port number for the web server?

The port number can be found using nmap and syntax , host number is shown.

Answer 1: 80

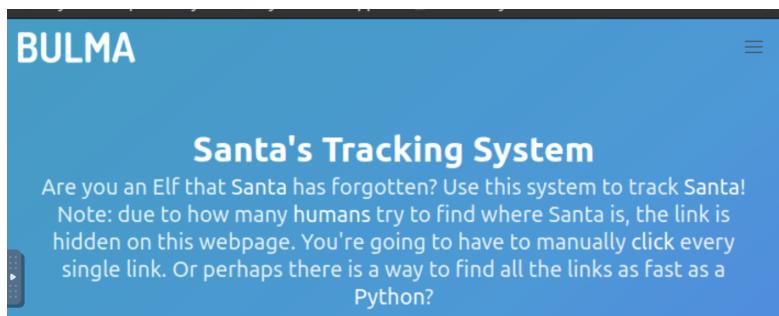
```
root@ip-10-10-96-38:~#
File Edit View Search Terminal Help
root@ip-10-10-96-38:~# nmap 10.10.223.177
Starting Nmap 7.60 ( https://nmap.org ) at 2022-07-13 06:36 BST
Nmap scan report for ip-10-10-223-177.eu-west-1.compute.internal (10.10.223.177)
Host is up (0.0079s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 02:9E:7D:3D:0C:0B (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.73 seconds
```

Question 2: What templates are being used?

For this , we checked the top left of the website

Answer 2: BULMA



Question 3: Without using enumeration tools such as Dirbuster, what is the directory for the API?

Open the developers tools in your browser , then open your elements tab. Scroll down and check the footer to find the HTML tag

Answer 3: /api/

```

76      <li><a href="#">Lorem ipsum dolor sit amet</a></li>
77      <li><a href="#">Alistair Croll</a></li>
78      <li><a href="#">Murphy's Law</a></li>
79      <li><a href="#">Flimsy Lavenrock</a></li>
80      <li><a href="#">Haven Mouse Lavender</a></li>
81    </ul>
82  </div>
83  <div class="column is-3">
84    <h2><strong>Category</strong></h2>
85    <ul>
86      <li><a href="#">Labore et dolore magna aliqua</a></li>
87      <li><a href="#">Xanhan airis sum eschelor</a></li>
88      <li><a href="http://machine.lib/api/api_key">Modular modern free</a></li>
89      <li><a href="#">The King of Cliffs</a></li>
90      <li><a href="#">The Dark Side of Bulk</a></li>
91      <li><a href="#">Course Correction</a></li>
92      <li><a href="#">Better Angels</a></li>
93    </ul>
94  </div>
95  <div class="column is-4">
96    <h2><strong>Category</strong></h2>
97    <ul>
98      <li><a href="#">Objects in space</a></li>
99      <li><a href="#">Playing cards with coyote</a></li>
100     <li><a href="#">The Dark Road</a></li>
101     <li><a href="#">The Garden of Forking Paths</a></li>
102     <li><a href="#">Future Shock</a></li>
103   </ul>
104 </div>
105 </div>
106 <div class="content has-text-centered">
107   <p>
108     <a class="icon" href="https://github.com/BulmaTemplates/bulma-template">
109       <i class="fa fa-github"></i>
110     </a>
111   </p>
112   <div class="control level-item">
113     <a href="https://github.com/BulmaTemplates/bulma-templates">
114       <div class="level-item has-action">
115         <span>Fork</span>
116         <span>Dark Bulma Template</span>
117       </div>
118     </a>
119   </div>
120 </div>

```

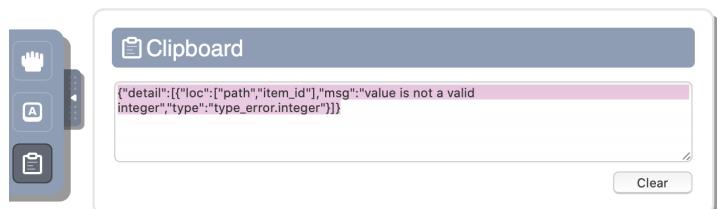
It looks like you haven't started Firefox in a while. Do you want to clean it? Refresh Firefox

Question 4: Go to the API endpoint. What is the Raw Data returned if no parameters are entered?

the raw data returned was retrieved from the endpoint of the API.

Answer 4: {"detail": [{"loc": ["path", "item_id"], "msg": "value is not a valid integer", "type": "type_error.integer"}]}

```
{"detail": [{"loc": ["path", "item_id"], "msg": "value is not a valid integer", "type": "type_error.integer"}]}
```



Question 5: Where is Santa right now?

Answer 5: Winter Wonderland , Hyde Park , London

Question 6: Find out the correct API key. Remember, this is an odd number between 0-100.

Answer 6: 57

For both questions , we had to try our luck and key in an odd number between 0-100 to know where Santa is right now. 57 is

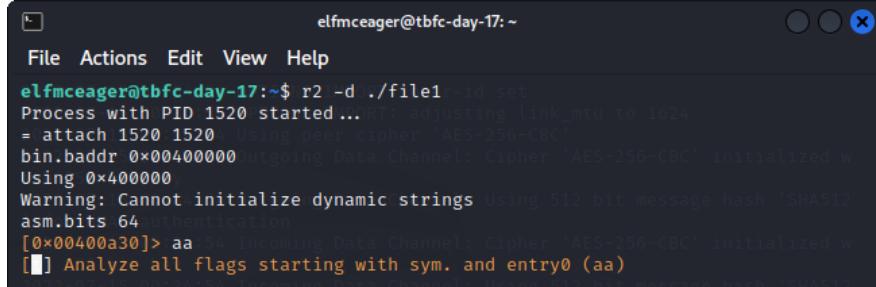
an odd number and is fortunately the answer for Question 6 and they key to finding out where Santa is

```
$ cat apibrute.py
#!/usr/bin/env python3

# Import the libraries we downloaded earlier
# if you try importing without installing them, this step will fail
import requests
{"item_id":49,"q":"Error. Key not valid!"}
api_key: 51
{"item_id":51,"q":"Error. Key not valid!"}
api_key: 53
{"item_id":53,"q":"Error. Key not valid!"}
api_key: 55
{"item_id":55,"q":"Error. Key not valid!"}
api_key: 57
{"item_id":57,"q":"Winter Wonderland, Hyde Park, London."}
api_key: 59
{"item_id":59,"q":"Error. Key not valid!"}
api_key: 61
{"item_id":61,"q":"Error. Key not valid!"}
api_key: 63
{"item_id":63,"q":"Error. Key not valid!"}
api_key: 65
{"item_id":65,"q":"Error. Key not valid!"}
api_key: 67
```

[Day 17] - Reverse Engineering - Reverse ELFneering

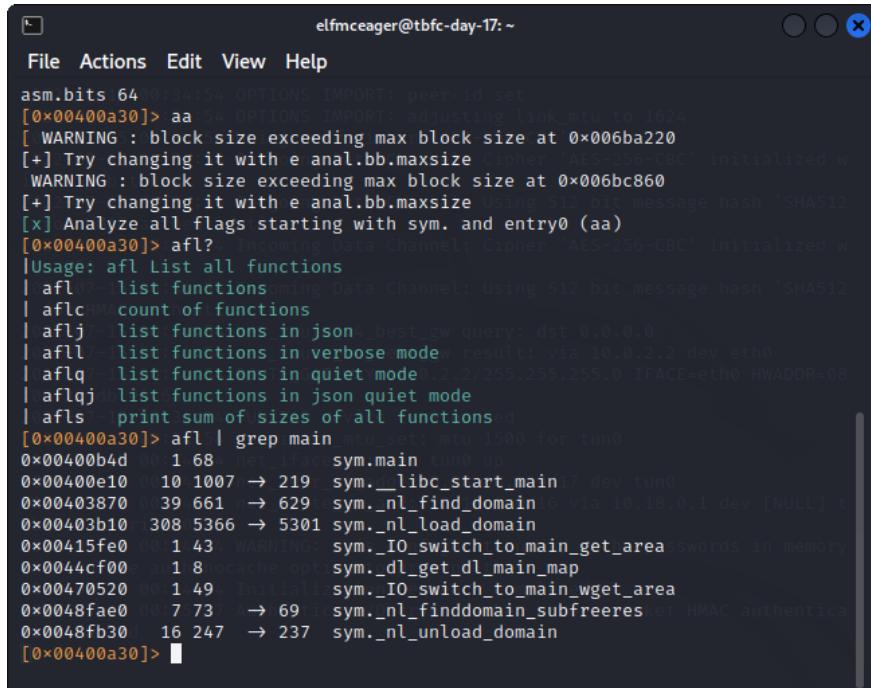
Log into the instance with the given credentials. Opens the binary files in debugging mode using command r2 -d ./file1



```
elfmceager@tbfc-day-17:~$ r2 -d ./file1
Process with PID 1520 started ...
RT: adjusting link_mtu to 1624
= attach 1520 1520 Using peer cipher 'AES-256-CBC'
bin.baddr 0x00400000 outgoing Data Channel: Cipher 'AES-256-CBC' initialized w
Using 0x40000
Warning: Cannot initialize dynamic strings Using 512 bit message hash 'SHA512
asm.bits 64 authentication
[0x00400a30]> aa 54 Incoming Data Channel: Cipher 'AES-256-CBC' initialized w
[!] Analyze all flags starting with sym. and entry0 (aa)
```

Once the analyzing completes we can seek out the entry point to the file.

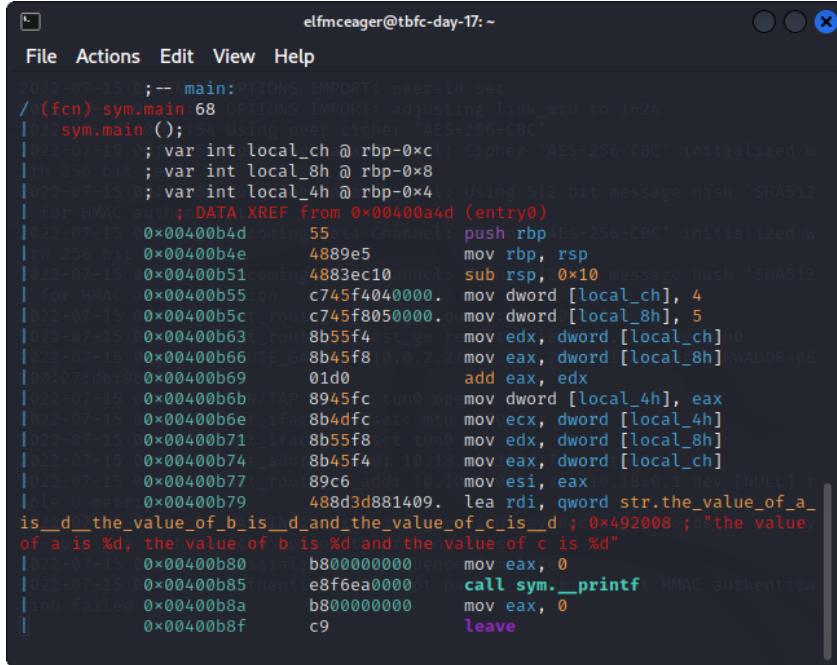
```
afl | grep main
```



```
asm.bits 64 10:34:54 OPTIONS IMPORT: peer-id set
[0x00400a30]> aa 54 OPTIONS IMPORT: adjusting link_mtu to 1624
[+] WARNING : block size exceeding max block size at 0x006ba220
[+] Try changing it with e anal.bb.maxsize Cipher 'AES-256-CBC' initialized w
WARNING : block size exceeding max block size at 0x006bc860
[+] Try changing it with e anal.bb.maxsize Using 512 bit message hash 'SHA512
[x] Analyze all flags starting with sym. and entry0 (aa)
[0x00400a30]> afl?
[!] Usage: afl List all functions
| afl - list functions
| aflc - count of functions
| aflj - list functions in json
| best_gw query: dst 0.0.0.0
| aflen - list functions in verbose mode
| result: via 10.0.2.2 dev eth0
| afll - list functions in quiet mode
| aflo - list functions in json quiet mode
| afls - print sum of sizes of all functions
[0x00400a30]> afl | grep main
mtu_set: mtu 1500 for tun0
0x00400b4d 0 1 684 net_ifare sym.main tun0 in
0x00400e10 0 10 1007 → 219 sym._libc_start_main 7 dev tun0
0x00403870 0 39 661 → 629 sym._nl_find_domain 16 via 10.18.0.1 dev [NULL] t
0x00403b10 0 308 5366 → 5301 sym._nl_load_domain
0x00415fe0 0 1 434 WARNING: sym._IO_switch_to_main_get_area swords in memory
0x0044cf00 0 1 8 locate option sym._dl_get_dl_main_map
0x00470520 0 1 494 Initialize sym._IO_switch_to_main_wget_area
0x0048fae0 0 7 737 → 691 sym._nl_fnddomain_subfreeres net HMAC authentication
0x0048fb30 0 16 247 → 237 sym._nl_unload_domain
[0x00400a30]>
```

Now we can see where in memory the program starts from. Since we found a function here we can examine it with the pdf (Print Disassembly Function) command.

```
pdf @main
```



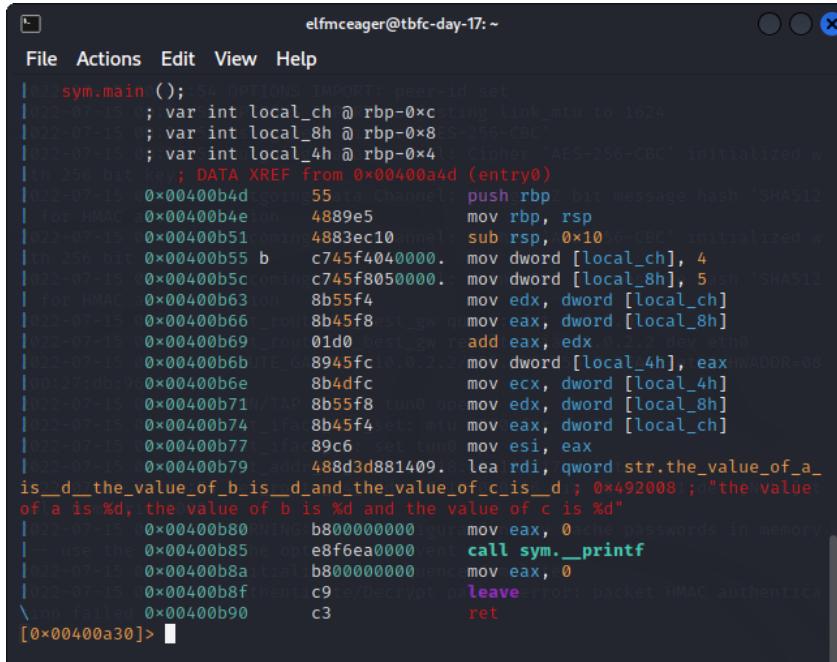
The screenshot shows the GDB PDF viewer window with the assembly code for the main function. The code is color-coded to highlight different instructions and labels. Key labels include `sym.main()`, `entry0`, and `leave`. The assembly instructions involve moving values between registers (e.g., `rbp`, `rsp`, `eax`, `ecx`) and memory locations, and performing HMAC calculations using the `c745f4040000` instruction.

```
022-07-15 0;-- main: OPTIONS IMPORT: peer-id set
/ (fcn) sym.main 68 OPTIONS IMPORT: adjusting link_mtu to 1624
022-07-15 0; var int local_ch @ rbp-0xc1: Cipher 'AES-256-CBC' initialized w
ith 256 bit ; var int local_8h @ rbp-0x8
022-07-15 0; var int local_4h @ rbp-0x4: using 512-bit message hash 'SHA512
for HMAC a@0x00400a4d ; DATA XREF from 0x00400a4d (entry0)
022-07-15 0x00400b4d going 55 to Channel: push rbp AES-256-CBC' initialized w
ith 256 bit 0x00400b4e 4889e5 mov rbp, rsp
022-07-15 0x00400b51 on 4883ec10annel: sub rsp, 0x10 message hash 'SHA512
for HMAC a@0x00400b55 on c745f4040000. mov dword [local_ch], 4
022-07-15 0x00400b5c _rcd c745f8050000. mov dword [local_8h], 5
022-07-15 0x00400b63 _rcd 8b55f4_st_gw r mov edx, dword [local_ch]@0
022-07-15 0x00400b66 TE_G 8b45f8 0.0.2.2 mov eax, dword [local_8h]@WADDR=0B
00:27:db:960*0x00400b69 01d0 add eax, edx
022-07-15 0x00400b6b /TAP 8945fc tun0 op mov dword [local_4h], eax
022-07-15 0x00400b6e _ifa 8b4dfc et; mtu mov ecx, dword [local_4h]
022-07-15 0x00400b71 _ifa 8b55f8 et tun0 mov edx, dword [local_8h]
022-07-15 0x00400b74 _add 8b45f4; 10:18 mov eax, dword [local_ch]
022-07-15 0x00400b77 _rcd 89c6 addi 10:1 mov esi, eax@.18.0,1 dev [NULL].t
ble 0 metric@0x00400b79 488d3d881409. lea rdi, qword str.the_value_of_a_
is_d_the_value_of_b_is_d_and_the_value_of_c_is_d ; 0x492008 ;"the value
of a is %d, the value of b is %d and the value of c is %d"
022-07-15 0x00400b80 tial b800000000@.end mov eax, 0
022-07-15 0x00400b85 hent e8f6ea0000@.ent call sym.__printf HMAC authenti
cation Failed 0x00400b8a b800000000 mov eax, 0
0x00400b8f c9 leave
```

We can set a breakpoint with db.

```
db 0x00400b55
```

After setting our break point we, we can again run pdf @main and now we will see a lowercase b right after our break point location



The screenshot shows the GDB PDF viewer window with the assembly code for the main function. A breakpoint is set at address `0x00400b55`, indicated by the label `b` preceding the instruction. The assembly code is identical to the previous screenshot, showing the initialization of HMAC contexts and the preparation for printing the values of `a`, `b`, and `c`.

```
022-07-15 0;-- main: OPTIONS IMPORT: peer-id set
022-07-15 0; var int local_ch @ rbp-0xc1: Cipher 'AES-256-CBC' initialized w
ith 256 bit ; var int local_8h @ rbp-0x8
022-07-15 0; var int local_4h @ rbp-0x4: using 512-bit message hash 'SHA512
for HMAC a@0x00400a4d ; DATA XREF from 0x00400a4d (entry0)
022-07-15 0x00400b4d going 55 to Channel: push rbp AES-256-CBC' initialized w
ith 256 bit 0x00400b4e 4889e5 mov rbp, rsp
022-07-15 0x00400b51 on 4883ec10annel: sub rsp, 0x10 message hash 'SHA512
for HMAC a@0x00400b55 b c745f4040000. mov dword [local_ch], 4
022-07-15 0x00400b5c _rcd c745f8050000. mov dword [local_8h], 5 sh 'SHA512
022-07-15 0x00400b63 _rcd 8b55f4 mov edx, dword [local_ch]
022-07-15 0x00400b66 _rcd 8b45f8 st_gw qu mov eax, dword [local_8h]
022-07-15 0x00400b69 _rcd 01d0 best_gw add eax, edx@.0.2.2 dev eth0
022-07-15 0x00400b6b TE_GA 8945fc 0.0.2.2/mov dword [local_4h], eax@WADDR=0B
00:27:db:960*0x00400b6e 8b4dfc mov ecx, dword [local_4h]
022-07-15 0x00400b71 _ifa 8b55f8 et tun0 mov edx, dword [local_8h]
022-07-15 0x00400b74 _ifa 8b45f4 et; mtu mov eax, dword [local_ch]
022-07-15 0x00400b77 _rcd 89c6 addi 10:1 mov esi, eax
022-07-15 0x00400b79 488d3d881409. lea rdi, qword str.the_value_of_a_
is_d_the_value_of_b_is_d_and_the_value_of_c_is_d ; 0x492008 ;"the value
of a is %d, the value of b is %d and the value of c is %d"
022-07-15 0x00400b80 tial b800000000@.end mov eax, 0
022-07-15 0x00400b85 hent e8f6ea0000@.ent call sym.__printf
022-07-15 0x00400b8a b800000000@.end mov eax, 0
0x00400b8f c9 leave
[0x00400a30]>
```

Next, we use the dc command to run the program until it hits the break point. If we run pdf, the memory address at our breakpoint will be highlighted.

The screenshot shows the Immunity Debugger interface. The assembly window displays the program's code, including a call to `sym.__printf`. The memory dump window shows the memory contents starting at address `0x00400b55`, which includes the variable `local_ch`.

```
elfmceager@tbfc-day-17: ~
File Actions Edit View Help
0x00400b55 ; var int local_ch @ rbp-0xc
0x00400b55 ; var int local_8h @ rbp-0x8
0x00400b55 ; var int local_4h @ rbp-0x4
; DATA XREF from 0x00400a4d (entry0)
0x00400b4d 55 push rbp
0x00400b4e 4889e5 mov rbp, rsp
0x00400b51 4883ec10 sub rsp, 0x10
; -- rip:
0x00400b55 b c745f4040000. mov dword [local_ch], 4
0x00400b55 c745f8050000. mov dword [local_8h], 5
0x00400b63 8b55f4 mov edx, dword [local_ch]
0x00400b66 8b45f8 mov eax, dword [local_8h]
0x00400b69 01d0 best_gw add eax, edx
0x00400b6b 8945fc1000202020 mov dword [local_4h], eax
0x00400b6e 8b4dfc mov ecx, dword [local_4h]
0x00400b71 8b55f8 mov edx, dword [local_8h]
0x00400b74 8b45f4 mov eax, dword [local_ch]
0x00400b77 89c6 set rno mov esi, eax
0x00400b79 488d3d881409. lea rdi, qword str.the_value_of_a_
is_d_the_value_of_b_is_d_and_the_value_of_c_is_d ; 0x492008 ; "the value
of a is %d, the value of b is %d and the value of c is %d"
0x00400b80 b800000000 mov eax, 0
0x00400b85 e8f6ea0000 call sym.__printf
0x00400b8a b800000000 mov eax, 0
0x00400b8f c9 leave
\on failed 0x00400b90 C3 ret
[0x00400b55]>
```

Next, to view the contents of, in our case, the `local_ch` variable, we will run `px @memory-address`

```
px @rbp-0cx
```

The screenshot shows the Immunity Debugger interface. The assembly window displays the program's code, including a call to `sym.__printf`. The memory dump window shows the memory contents starting at address `0x00400b55`, which includes the variable `local_ch`.

```
[0x00400b55]> px @ rbp-0xc
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x7ffd291902c4 0000 0000 1890 6b00 0000 0000 7018 4000 .....k....p.0.
0x7ffd291902d4 0000 0000 1911 4000 0000 0000 0000 0000 .....@.....
0x7ffd291902e4 0000 0000 0000 0000 0100 0000 f803 1929 ..... ....)
0x7ffd291902f4 fd7f 0000 4d0b 4000 0000 0000 0000 0000 .....M.@.....R=08
0x7ffd29190304 0000 0000 1700 0000 0100 0000 0000 0000 .....@.....
0x7ffd29190314 0000 0000 0000 0000 0200 0000 0000 0000 .....@.....
0x7ffd29190324 0000 0000 0000 0000 0000 0000 0000 0000 .....@.....
0x7ffd29190334 0000 0000 0000 0000 0000 0000 0004 4000 .....@.....
0x7ffd29190344 0000 0000 ac61 4748 1987 d477 1019 4000 .....aGH...w..@.
0x7ffd29190354 0000 0000 0000 0000 0000 0000 1890 6b00 .....k....L] t
0x7ffd29190364 0000 0000 0000 0000 0000 ac61 677d .....ag}
0x7ffd29190374 abd5 2e88 ac61 3359 1987 d477 0000 0000 .....a3Y...w....emory
0x7ffd29190384 0000 0000 0000 0000 0000 0000 0000 0000 .....@.....
0x7ffd29190394 0000 0000 0000 0000 0000 0000 0000 0000 .....@.....
0x7ffd291903a4 0000 0000 0000 0000 0000 0000 0000 0000 .....@.....
0x7ffd291903b4 0000 0000 0000 0000 0000 0000 0000 0000 .....@.....
[0x00400b55]>
```

But we do not see our variable 4 so we will step forward using `ds`. Then we will run `px @rbp-0cx` again.

```
[0x00400b55]> px @ rbp-0xc
- offset -
  0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0x7ffd291902c4 0400 0000 1890 6b00 0000 0000 7018 4000 ... .k....p@.
0x7ffd291902d4 0000 0000 1911 4000 0000 0000 0000 0000 ... ..@...
0x7ffd291902e4 0000 0000 0000 0000 0100 0000 f803 1929 ... ..... )
0x7ffd291902f4 fd7f 0000 4d0b 4000 0000 0000 0000 0000 ... M.@.....@-08
0x7ffd29190304 0000 0000 1700 0000 0100 0000 0000 0000 ... .....
0x7ffd29190314 0000 0000 0000 0000 0200 0000 0000 0000 ... .....
0x7ffd29190324 0000 0000 0000 0000 0000 0000 0000 0000 ... .....
0x7ffd29190334 0000 0000 0000 0000 0000 0004 4000 ... .@.
0x7ffd29190344 0000 0000 ac61 4748 1987 d477 1019 4000 ... .aGH...w...@.
0x7ffd29190354 0000 0000 0000 0000 0000 1890 6b00 10 ... .k..] t
0x7ffd29190364 0000 0000 0000 0000 0000 ac61 677d ... .....ag}
0x7ffd29190374 abd5 2e88 ac61 3359 1987 d477 0000 0000 ... .a3Y...w...emory
0x7ffd29190384 0000 0000 0000 0000 0000 0000 0000 0000 ... .....
0x7ffd29190394 0000 0000 0000 0000 0000 0000 0000 0000 ... .....
0x7ffd291903a4 0000 0000 0000 0000 0000 0000 0000 0000 ... .....rica
0x7ffd291903b4 0000 0000 0000 0000 0000 0000 0000 0000 ... .....
[0x00400b55]>
```

And we now can see the variable 4 returned. Going ahead to the challenge file, we follow the same initial steps

```
elfmceager@tbfc-day-17:~$ r2 -d ./challenge1
Process with PID 1555 started ... add: 10.10.0.0/16 via 10.10.0.1 dev tun0
= attach 1555 1555
bin.baddr 0x00400000 WARNING: this configuration may cache passwords in memory
Using 0x400000
Warning: Cannot initialize dynamic strings Completed
asm.bits 64
asm.os 35107 Authenticate-Decrypt packet error: packet HMAC authentication failed
[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 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
[0x00400a30]>
```

pdf @ main

```
[0x00400a30]> pdf @main
;-- main: outgoing Data Channel: Cipher 'AES-256-CBC' initialized w
/ (fcn) sym.main 35
|_ sym.main ():: outgoing Data Channel: Using 512 bit message hash 'SHA512'
| for HMAC ; var int local_ch @ rbp-0xc
|_ 22-07-19 ; var int local_8h @ rbp-0x8 ; Cipher 'AES-256-CBC' initialized w
| h 256 bit
|_ 22-07-19 ; DATA XREF from 0x00400a4d (entry0) bit message hash 'SHA512'
| for HMAC
| 0x00400b4d 55 push rbp
| 22-07-19 0x00400b4e 4889e5 mov rbp, rsp 0.0
| 22-07-19 0x00400b51 40 c745f4010000 mov dword [local_ch], 1 END
| 22-07-19 0x00400b58 40 c745f8060000 mov dword [local_8h], 6 HWADDR=00
| 22-07-19 0x00400b5f 8b45f4 mov eax, dword [local_ch]
| 22-07-19 0x00400b62 40 0faf45f8 imul eax, dword [local_8h]
| 22-07-19 0x00400b66 40 8945fc mov dword [local_4h], eax
| 22-07-19 0x00400b69 40 b800000000 mov eax, 0
| 22-07-19 0x00400b6e 5d pop rbp dev tun0
\ 0x00400b6f c3 ret
```

Q1 : Match the data type with the size in bytes:

Byte = 1

Word = 2

Double word = 4

Quad = 8

Single precision = 4

```
Double precision = 8
```

Q2: What is the command to analyse the program in radare2?

Ans : aa

Q3: What is the command to set a breakpoint in radare2 ?

Ans : db

Q4: What is the command to execute the program until we hit a breakpoint?

Ans : dc

Question 5

What is the value of local_ch when its corresponding movl instruction is called (first if multiple)?

Ans : 1

Find the address of local_ch which is 0x00400b51 and put a breakpoint at this location using command db. Now use dc to run the program. We can analyse the contents of local_ch with the command px @ rbp-0xc. Run command ds to step forward and command px @ rbp-0xc again. We see the value 1.

The screenshot shows the radare2 debugger interface. At the top, there's a menu bar with File, Actions, Edit, View, Help. Below the menu, there's a status bar showing the current file path: elfmceager@tbfc-day-17:~. The main window has two panes. The left pane displays assembly code:

```
elfmceager@tbfc-day-17:~
```

```
File Actions Edit View Help
```

```
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
c3          ret
```

```
[0x00400a30]> db @0x00400b51 IMPORTS: adjusting link_mtu to 1624
[0x00400a30]> dc
hit breakpoint at: 400b51 in Data Channel Cipher 'AES-256-CBC' initialized w
[0x00400b51]> px @ rbp-0xc
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF 512
0x7ffdbe3c4744 0000 0000 1890 6b00 0000 0000 4018 4000 .....k.....@.
0x7ffdbe3c4754 0000 0000 e910 4000 0000 0000 0000 0000 .....@.
0x7ffdbe3c4764 0000 0000 0000 0000 0100 0000 7848 3cbe .....xH<.
0x7ffdbe3c4774 fd7f 0000 40db 4000 0000 0000 0000 0000 ..M.@. 0x00400b512
0x7ffdbe3c4784 0000 0000 1700 0000 0100 0000 0000 0000 ..... .
0x7ffdbe3c4794 0000 0000 0000 0000 0200 0000 0000 0000 ..... .
0x7ffdbe3c47a4 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x7ffdbe3c47b4 0000 0000 0000 0000 0000 0004 4000 .....@. 0x00400b512
0x7ffdbe3c47c4 0000 0000 a30f 2f0f 0766 1c6d e018 4000 .../.f.m..@.
0x7ffdbe3c47d4 0000 0000 0000 0000 0000 0000 1890 6b00 .....k.
0x7ffdbe3c47e4 0000 0000 0000 0000 0000 0000 a30f 6fb1 .....o.
0x7ffdbe3c47f4 ff1a e792 a30f 9ble 0766 1c6d 0000 0000 .....f.m..
0x7ffdbe3c4804 0000 0000 0000 0000 0000 0000 0000 0000 .....t.
0x7ffdbe3c4814 0000 0000 0000 0000 0000 0000 0000 0000 ..... ].
0x7ffdbe3c4824 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x7ffdbe3c4834 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
[0x00400b51]> ds
[0x00400b51]> px @ rbp-0xc
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF 512
0x7ffdbe3c4744 0100 0000 1890 6b00 0000 0000 4018 4000 .....k.....@.
0x7ffdbe3c4754 0000 0000 e910 4000 0000 0000 0000 0000 .....@.
0x7ffdbe3c4764 0000 0000 0000 0000 0100 0000 7848 3cbe .....xH<.
0x7ffdbe3c4774 fd7f 0000 40db 4000 0000 0000 0000 0000 ..M.@.
0x7ffdbe3c4784 0000 0000 1700 0000 0100 0000 0000 0000 ..... .
0x7ffdbe3c4794 0000 0000 0000 0000 0200 0000 0000 0000 ..... .
0x7ffdbe3c47a4 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x7ffdbe3c47b4 0000 0000 0000 0000 0000 0004 4000 .....@.
0x7ffdbe3c47c4 0000 0000 a30f 2f0f 0766 1c6d e018 4000 .../.f.m..@.
0x7ffdbe3c47d4 0000 0000 0000 0000 0000 0000 1890 6b00 .....k.
0x7ffdbe3c47e4 0000 0000 0000 0000 0000 0000 a30f 6fb1 .....o.
0x7ffdbe3c47f4 ff1a e792 a30f 9ble 0766 1c6d 0000 0000 .....f.m..
0x7ffdbe3c4804 0000 0000 0000 0000 0000 0000 0000 0000 .....t.
0x7ffdbe3c4814 0000 0000 0000 0000 0000 0000 0000 0000 ..... ].
0x7ffdbe3c4824 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x7ffdbe3c4834 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
[0x00400b51]>
```

Question 6

What is the value of eax when the imull instruction is called?

Ans : 6

Next, we want to find the value of eax at the first imul instruction. We can simply use ds to move to the next line and then use dr to see the value of eax. Since we didn't see any number, Run command ds to step forward and dr to see the value of eax. We see that the value is 6.

```
[0x00400b51]> ds -- OPTIONS IMPORT: adjusting link_mtu to 1624
[0x00400b51]> pdf @main
-- main: Using peer cipher 'AES-256-CBC'
/ (fcn) sym.main 35
| 22. sym.main () ;4 Outgoing Data Channel: Cipher 'AES-256-CBC' initialized w
| for HMAC ; var int local_ch @ rbp-0xc
| 22-07-15 ; var int local_8h @ rbp-0x8; Cipher 'AES-256-CBC' initialized w
| h 256 bit ; var int local_4h @ rbp-0x4
| 22-07-15 0h ; DATA XREF from 0x00400a4d (entry0) 2 bit message hash 'SHA512
| for HMAC 0x004004dton 55 push rbp
| 22-07-15 0x004004e4 r0ff8e5 sti .4889e5 mov rbp, rsp 0
| 22-07-15 0x00400b51 brc0 c745f4010000. mov dword [local_ch], 1 tho
| 22-07-15 0x00400588 8945fc stl c745f8060000. mov dword [local_8h], 6 HWADDR=00
| 0273:db:90 0x00400bf5 8b45f4 mov eax, dword [local_ch]
| 22-07-15 ; rip: UN/TAP device tun0 opened
| 22-07-15 0x00400b52 0faf45f8 imul eax, dword [local_8h]
| 22-07-15 0x00400b66 8945fc stl c745f8060000. mov dword [local_4h], eax
| 22-07-15 0x00400b69 b800000000 mov eax, 0 dev tun0
| 22-07-15 0x00400b6e r0ff4_dd7 5d4_dd7 10 pop rbp via 10.18.0.1 dev [NULL] t
\ 0x00400b6f c3 ret
```



The screenshot shows a terminal window with a dark theme. At the top, there's a menu bar with File, Actions, Edit, View, Help. Below the menu, the assembly code is displayed:

```
elfmceager@tbfc-day-17:~
```

```
File Actions Edit View Help
0x00400b51> 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
```

Below the assembly code, the debugger session continues:

```
[0x00400b51]> dr -- OPTIONS IMPORT: peer_id set
rax = 0x00000001:4 OPTIONS IMPORT: adjusting link_mtu to 1624
rbx = 0x00400400:4 Using peer cipher 'AES-256-CBC'
rcx = 0x0044b9a0:4 Outgoing Data Channel: Cipher 'AES-256-CBC' initialized w
rdx = 0x7ffdbe3c4888
r8 = 0x01000000:4 Outgoing Data Channel: Using 512 bit message hash 'SHA512
r9 = 0x006bb8e0 authentication
r10 = 0x00000015:4 Incoming Data Channel: Cipher 'AES-256-CBC' initialized w
r11 = 0x00000000
r12 = 0x004018e0:4 Incoming Data Channel: Using 512 bit message hash 'SHA512
r13 = 0x00000000 authentication
r14 = 0x006b9018:4 net_route_v4_best_gw query: dst 0.0.0.0
r15 = 0x00000000:4 net_route_v4_best_gw result: via 10.0.2.2 dev eth0
rsi = 0x7ffdbe3c4878 ROUTE_GATEWAY 10.0.2.2/255.255.255.0 IFACE=eth0 HWADDR=00
rdi = 0x00000001
rsp = 0x7ffdbe3c4750 UN/TAP device tun0 opened
rbp = 0x7ffdbe3c4750 net_iface_mtu_set: mtu 1500 for tun0
rip = 0x00400b62:4 net_iface_up: set tun0 up
rflags = 0x00000246 net_addr_v4_add: 10.18.40.169/17 dev tun0
orax = 0xfffffffffffffff net_route_v4_add: 10.10.8.16 via 10.18.0.1 dev [NULL] t
[0x00400b51]> ds
[0x00400b51]> dr:4 WARNING: this configuration may cache passwords in memory
rax = 0x00000006-nocache option to prevent this
rbx = 0x00400400:4 Initialization Sequence Completed
rcx = 0x0044b9a0:4 Authenticate/Decrypt packet error: packet HMAC authentication failed
rdx = 0x7ffdbe3c4888
r8 = 0x01000000
r9 = 0x006bb8e0
r10 = 0x00000015
r11 = 0x00000000
r12 = 0x004018e0
r13 = 0x00000000
r14 = 0x006b9018
r15 = 0x00000000
rsi = 0x7ffdbe3c4878
rdi = 0x00000001
rsp = 0x7ffdbe3c4750
rbp = 0x7ffdbe3c4750
rip = 0x00400b66
rflags = 0x00000246
orax = 0xfffffffffffffff
[0x00400b51]>
```

Question 7

What is the value of `local_4h` before `eax` is set to 0?

Ans : 6

Lastly, we want to find the value of `local_4h` before the `eax` is set to 0. Use the `ds` command to get to that point and then check the value of `local_4h` with command `px @ rbp-0x4`. The value here is 6.

```
[0x00400b51]> pdf @main
[0x00400b51]> ;-- main:
[0x00400b51]> / (fcn) sym.main 35 outgoing Data Channel: Using 512 bit message hash 'SHA512'
[0x00400b51]> | sym.main ();
[0x00400b51]> | 256 byte(s) allocated
[0x00400b51]> | var int local_ch @ rbp-0xc cipher 'AES-256-GCM' initialized w/ 0
[0x00400b51]> | var int local_8h @ rbp-0x8
[0x00400b51]> | var int local_4h @ rbp-0x4: Using 512 bit message hash 'SHA512'
[0x00400b51]> | For HMAC auth; DATA XREF from 0x00400a4d (entry0)
[0x00400b51]> | 0x00400b4d      55          push    rbp    0.0.0.0
[0x00400b51]> | 0x00400b4e      4899e5     mov     rbp, rsp 0.2.2 dev eth0
[0x00400b51]> | 0x00400b51 b  c745f4010000  mov     dword [local_ch], 10 HWADDR=08
[0x00400b51]> | 0x00400b58      c745f8060000  mov     dword [local_8h], 6
[0x00400b51]> | 0x00400b5f      8b45f4     mov     eax, dword [local_ch]
[0x00400b51]> | 0x00400b62      0faf45f8   imul   eax, dword [local_8h]
[0x00400b51]> | -- rip: set_itface_ip_set_time up
[0x00400b51]> | 0x00400b66      8945fc     add    rbp, esp 0.0.0.0
[0x00400b51]> | 0x00400b69      b800000000  mov     eax, 0 10.0.0.0 dev [NULL] t
[0x00400b51]> | 0x00400b6e      5d          pop    rbp
[0x00400b51]> \ 0x00400b6f C3  ret     This configuration may cache passwords in memory
[0x00400b51]> ds
[0x00400b51]> px @ rbp-0x4
[0x00400b51]> px @ rbp-0x4|Initialization Sequence Completed
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF ica
0x7ffdbe3c474c 0600 0000 4018 4000 0000 0000 e910 4000 ... @. .... @.
0x7ffdbe3c475c 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x7ffdbe3c476c 0100 0000 7848 3cbe fd7f 0000 4d0b 4000 ... xH< ... M.@.
0x7ffdbe3c477c 0000 0000 0000 0000 0000 0000 1700 0000 ..... .
0x7ffdbe3c478c 0100 0000 0000 0000 0000 0000 0000 0000 ..... .
0x7ffdbe3c479c 0200 0000 0000 0000 0000 0000 0000 0000 ..... .
0x7ffdbe3c47ac 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x7ffdbe3c47bc 0000 0000 0004 4000 0000 0000 a30f 2f0f ..... @. .... /
0x7ffdbe3c47cc 0766 1c6d e018 4000 0000 0000 0000 0000 .f.m..@.
0x7ffdbe3c47dc 0000 0000 1890 6b00 0000 0000 0000 0000 ..... k.
0x7ffdbe3c47ec 0000 0000 a30f 6fb1 ff1a e792 a30f 9b1e ..... o..
0x7ffdbe3c47fc 0766 1c6d 0000 0000 0000 0000 0000 0000 .f.m...
0x7ffdbe3c480c 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x7ffdbe3c481c 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x7ffdbe3c482c 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
0x7ffdbe3c483c 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
[0x00400b51]> ]
```

Thought Process/Methodology:

Log into the instance with the given credentials. Opens the binary files in debugging mode using command r2 -d ./file1 . Once the analyzing completes we can seek out the entry point to the file. Now we can see where in memory the program starts from. Since we found a function here we can examine it with the pdf (Print Disassembly Function) command. We can set a breakpoint with db. After setting our break point we can again run pdf @main and now we will see a lowercase b right after our break point location. Next, we use the dc command to run the program until it hits the break point. If we run pdf, the memory address at our breakpoint will be highlighted. Next, to view the contents of, in our case, the local_ch variable, we will run px @memory-address. But we do not see our variable 4 so we will step forward using ds. Then we will run px @rbp-0cx again. And we now can see the variable 4 returned. Going ahead to the challenge file, we follow the same initial steps. Find the address of local_ch which is 0x00400b51 and put a breakpoint at this location using command db. Now use dc to run the program. We can analyse the contents of local_ch with the command px @ rbp-0xc. Run command ds to step forward and command px @ rbp-0xc again. We see the value 1. Next, we want to find the value of eax at the first imul instruction. We can simply use ds to move to the next line and then use dr to see the value of eax. Since we didn't see any number, Run command ds to step forward and dr to see the value of eax. We see that the value is 6. Lastly, we want to find the value of local_4h before the eax is set to 0. Use the ds command to get to that point and then check the value of local_4h with command px @ rbp-0x4. The value here is 6.

Day 18: The Bits Of Christmas

Reverse Engineering the .Net app

Now we'll open the **TBFC APP** in **ILSpy**. This will allow us to conduct some reverse engineering and look at some of the app's source code.

When we open it, we are prompted to enter Santa's password. After some investigation, We discovered something intriguing in **CrackMe -> MainForm -> buttonActivate Click**.

```
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;
                }
                continue;
            }
            MessageBox.Show("Welcome, Santa, here's your flag thm{046af}", "That's the right key!", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
            return;
        }
        MessageBox.Show("Uh Oh! That's the wrong key", "You're not Santa!", MessageBoxButtons.OK, MessageBoxIcon.Hand);
    }
}
```

It appears that the programmer committed the error of hard-coding several important data! The password we're looking for is santapassword123.

```
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;
                }
                continue;
            }
            MessageBox.Show("Welcome, Santa, here's your flag thm{046af}", "That's the right key!", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
            return;
        }
        MessageBox.Show("Uh Oh! That's the wrong key", "You're not Santa!", MessageBoxButtons.OK, MessageBoxIcon.Hand);
    }
}
```

Question 1: santapassword123

We also see the flag that will be visible when we properly log in!

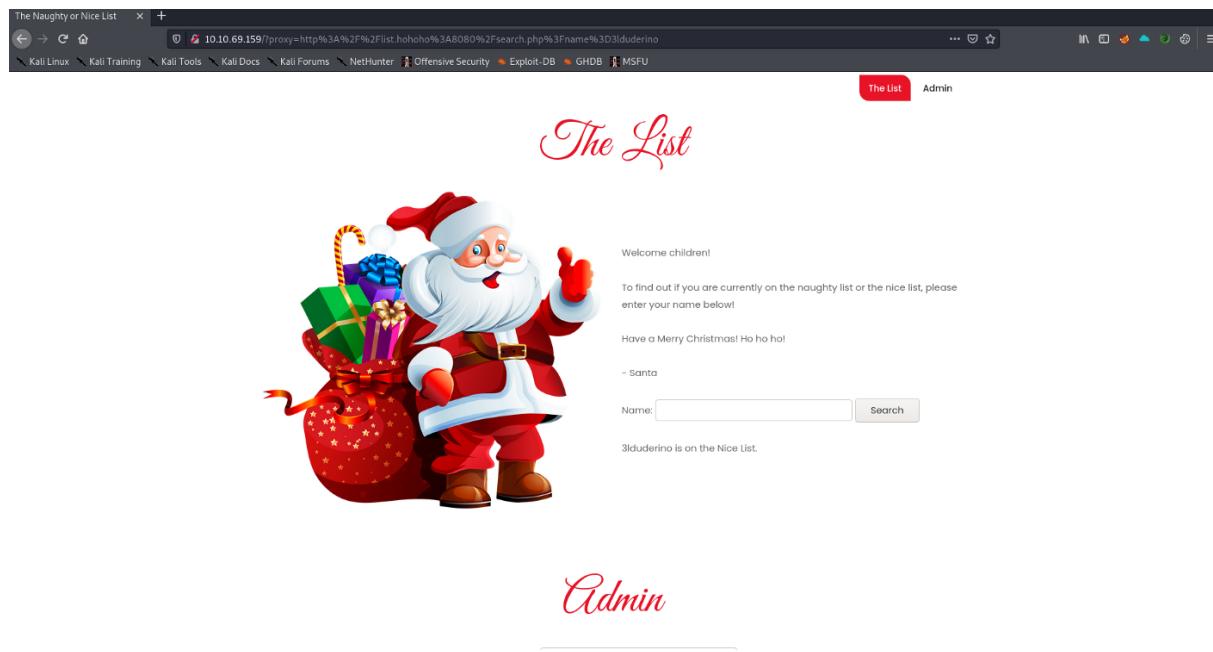
```
private unsafe void buttonActivate_Click(object sender, EventArgs e)
{
    IntPtr value = Marshal.StringToGlobalAnsi(textBoxKey.Text);
    sbyte* ptr = (sbyte*)System.Runtime.CompilerServices.Unsafe.AsPointer<ref <Module>.??_C@_0BB@IKKDFEPG@santa@password123@);
    void* ptr2 = (void*)value;
    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;
                }
                continue;
            }
            MessageBox.Show("Welcome, Santa, here's your flag thm{046af}", "That's the right key!", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
            return;
        }
        MessageBox.Show("Uh Oh! That's the wrong key", "You're not Santa!", MessageBoxButtons.OK, MessageBoxIcon.Hand);
    }
}
```

Question 2: thm{046af}

Day 19: The Naught Or Nice List!

Question 1: What is Santa's password?

When we first visit the website, we see this form, along with an admin login underneath it.



The URL is intriguing:

**http://10.10.69.159/?proxy=http%3A%2F%2Flist.hohoho%3A8080%2Fs
earch.php%3Fname%3d31 Duderino** Also, we're not on the nice list, which is unfortunate. We conducted a short examination of the source code and found nothing of interest.

Burp did a short URL decoding and returned the following:

**http://10.10.69.159/?proxy=http://list.hohoho:8080/search.php?
name=31duderino**

We search strip everything past the port number to determine if there is a root site and create a custom script to check for extra folders. In a failure site, we know what we're searching for (in this case, the root site).

Name:

Search

Not Found

The requested URL was not found on this server.

But for the time being, let's go on and see if there's another option.

So, it appears that we have a local server with port 8080 open; we may experiment with the port number to see what we can obtain. We can use this to perform basic port scanning.

Name:

Search

Failed to connect to list.hohoho port 21: Connection refused

However, it appears that we immediately receive a hit on port 22, which is open on the distant server.

Name:

Search

Recv failure: Connection reset by peer

We attempted to obtain the `passwd` file, but it appears that they are analysing the URL.

Name:

file://etc/passwd is on the Nice List.

When we try to determine whether there is a local only server operating, we are blocked (by their security team).

<http://10.10.69.159/?proxy=http%3A%2F%2Flocalhost>

Name:

Your search has been blocked by our security team.

Let's return to the port issue and see if we can find any local only services running on the server.

<http://10.10.69.159/?proxy=http%3A%2F%2Flist.hohoho.localtest.me:22>

With the connection reset by peer, we received the same problem as before. So far, so good. We may check to see if there are any ports that are exclusively accessible to the localhost. We now have a hit with port 80 containing some goodies, which addresses Question 1.

<http://10.10.69.159/?proxy=http%3A%2F%2Flist.hohoho.localtest.me:80>

Name:

Search

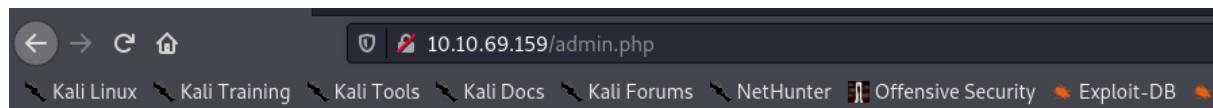
Santa,

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

Please keep in mind that the username is case sensitive.



List Administration

This page is currently under construction.

Only press this button when emergency levels of Christmas cheer are needed!

We will obtain the flag for today's challenge after deleting the naughty list (or reading the source code).

Day 20: Powershell to the rescue

Question 1: Search for the first hidden elf file within the Documents folder. Read the contents of this file. What does Elf 1 want?

We discover a few hidden files after listing the contents of the Documents folder in search of them.

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

```
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 elfone.txt

PS C:\Users\mceager\Documents> .
```

Look at the two files—one is hidden and the other is not. The file names differ, as we can see.

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

Question 2: Search on the desktop for a hidden folder that contains the file for Elf 2. Read the contents of this file. What is the name of that movie that Elf 2 wants?

It should be simple to access the desktop, search for a hidden folder rather than a file, and open the folder's contents.

```

PS C:\Users\mceager\Documents> cd ..\Desktop\
PS C:\Users\mceager\Desktop> Get-ChildItem -Directory -Hidden

    Directory: C:\Users\mceager\Desktop

Mode                LastWriteTime         Length Name
----                -----          ----- 
d--h--        12/7/2020 11:26 AM           0 elf2wo

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

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

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

PS C:\Users\mceager\Desktop\elf2wo> Get-Content .\e70smsW10Y4k.txt
I want the movie Scrooged <3!
PS C:\Users\mceager\Desktop\elf2wo>

```

Question 3: Search the Windows directory for a hidden folder that contains files for Elf 3. What is the name of the hidden folder?

We checked inside the files for anything pertaining to Elf 3 and searched for secret folders, but were unsuccessful. For this one, We had to use a small hint from the task information. After creating the string to use, entering the Windows folder, and allowing it to run for some time.

```
Get-ChildItem -Directory -Hidden -Recurse -Filter '*3*' -ErrorAction SilentlyContinue
```

```

PS C:\Windows> Get-ChildItem -Directory -Hidden -Recurse -Filter '*3*' -ErrorAction SilentlyContinue

    Directory: C:\Windows\System32

Mode                LastWriteTime         Length Name
----                -----          ----- 
d--h--        11/23/2020 3:26 PM           0 3lfthr3e

```

We obtained our secret folder. Let's check it out from inside.

Question 4: How many words does the first file contain?

Just a Measure-Object piped in from a Get-Content will do
(cat)

```
PS C:\Windows> cd .\System32\  
PS C:\Windows\System32> cd .\3lfthr3e\  
PS C:\Windows\System32\3lfthr3e> dir  
PS C:\Windows\System32\3lfthr3e> Get-ChildItem -hidden -file  
  
Directory: C:\Windows\System32\3lfthr3e  
  
Mode                LastWriteTime         Length Name  
----                -----          -----  
-arh--        11/17/2020  10:58 AM           85887 1.txt  
-arh--        11/23/2020  3:26 PM          12061168 2.txt  
  
PS C:\Windows\System32\3lfthr3e> cat 1.txt | Measure-Object  
  
Count      : 9999  
Average    :  
Sum        :  
Maximum    :  
Minimum    :  
Property   :  
  
PS C:\Windows\System32\3lfthr3e>
```

Question 5: What 2 words are at index 551 and 6991 in the first file?

That one is rather simple to solve using indexes and Get-Content.

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

Question 6: This is only half the answer. Search in the 2nd file for the phrase from the previous question to get the full answer. What does Elf 3 want?

We first attempted using the `Select-String` command to look for a Pattern containing "Red Ryder," but we were unable to do so.

```
Select-String -Path 'C:\Windows\System32\3lfthr3e\2.txt' -String "Red Ryder"
```

Following that, We used `-Pattern "Red," "Ryder,"` which ended in matching terms with either Red or Ryder and produced much too much data to identify the solution. We believe We just omitted the "Red" and, in typical lazy form, merely looked for the word "Ryder," which matched a single string.

```
C:\Users\mceager\Desktop\elf2wo> Select-String -Path 'C:\Windows\System32\3lfthr3e\2.txt' -Pattern "Red", "Ryder"

\Windows\System32\3lfthr3e\2.txt:690:oxposlfredlyg
\Windows\System32\3lfthr3e\2.txt:1245:inwqredyooobl
\Windows\System32\3lfthr3e\2.txt:1341:mcrilzllkredb
\Windows\System32\3lfthr3e\2.txt:3211:lfredlyglvbfi
\Windows\System32\3lfthr3e\2.txt:3862:zllkredbgvqag
\Windows\System32\3lfthr3e\2.txt:5731:bgtoxposlfred
\Windows\System32\3lfthr3e\2.txt:6286:gfcinwqredyoo
\Windows\System32\3lfthr3e\2.txt:8252:poslfredlyglv
\Windows\System32\3lfthr3e\2.txt:8807:wqredyoooblox
\Windows\System32\3lfthr3e\2.txt:8903:rilzllkredbgv
\Windows\System32\3lfthr3e\2.txt:10773:redlyglvbffimc
\Windows\System32\3lfthr3e\2.txt:11327:qnrgfcinwqred
\Windows\System32\3lfthr3e\2.txt:11424:lkredbgvqagms
\Windows\System32\3lfthr3e\2.txt:13293:toxposlfredly
\Windows\System32\3lfthr3e\2.txt:13848:cinwqredyooob
\Windows\System32\3lfthr3e\2.txt:13944:pmcrilzllkred
```

```
C:\Users\mceager\Desktop\elf2wo> Select-String -Path 'C:\Windows\System32\3lfthr3e\2.txt' -Pattern "Ryder"

\Windows\System32\3lfthr3e\2.txt:558704:redryderbbgun
```

This is only half the answer. Search in the 2nd file for the phrase from the previous question to get the full answer. What does Elf 3 want? (use spaces when submitting the answer)

Red Ryder BB Gun

Correct Answer

Hint

Thought Process/ Methodology:

First, we launched PowerShell and navigated to the Documents folder. We use powershell command to access the 'Documents' directory. After we navigate into the directory, we can see a list of the directory contents with Get-ChildItem but the result of this command doesn't require the results we want. Then, we add additional flags; Get-ChildItem -Hidden -File to specify the command. Then, we use the command Get-Content to see the content of the file and find what elf1 wants. Next, we navigate into the Desktop directory where it lists all the contents inside it using powershell command; Get-ChildItem -Hidden -Directory and navigate into 'elf2wo' directory. Next, we're looking for a hidden folder that contains files for Elf 3 using command Get-ChildItem -Hidden -Filter '*3*' We navigate into the file we found earlier and list the file inside. Using Measure-Object cmdlet with flag -Word to count all the words. Then we use the common that has been provided (Get-Content .\1.txt) [551, 6991] to find the 2 words in the first file. For the last question, we use command Select-String <path/filename> -Pattern 'redryder' to get the full answer.