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’

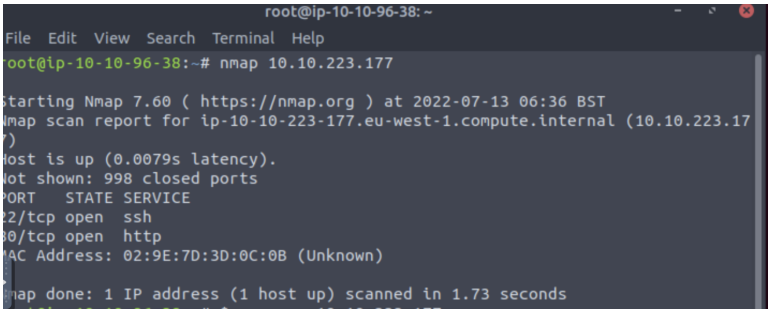
ID	NAME	ROLE
1211102162	AMILIA NADZEERA BINTI BAHARUDIN	Leader
1211100930	KU NAJWA SYAUQINA BINTI KU AZRIN	Member
1211101693	SAVITHA MURUGUMUNISEGARAN	Member

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.

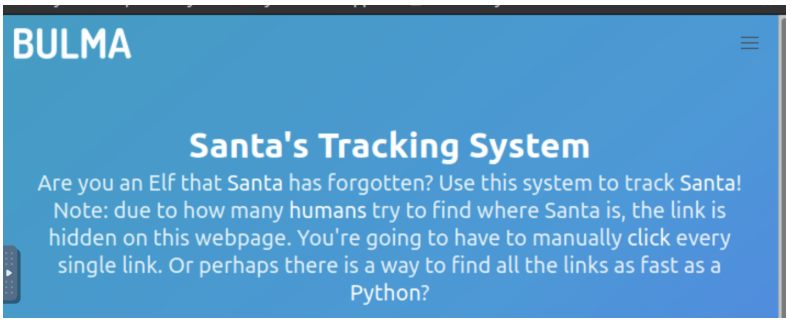
Answer 1: 80



Question 2: What templates are being used?

For this , we checked the top left of the website

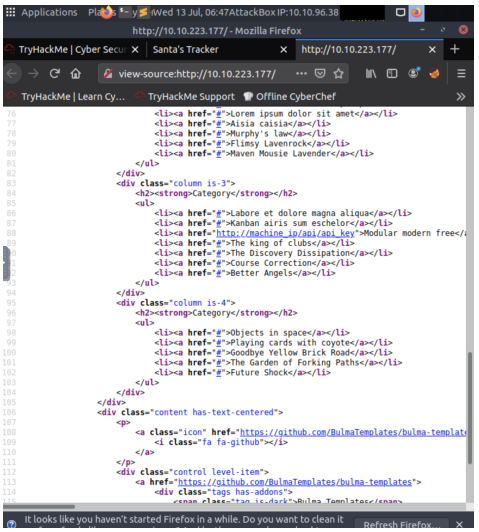
Answer 2: BULMA



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

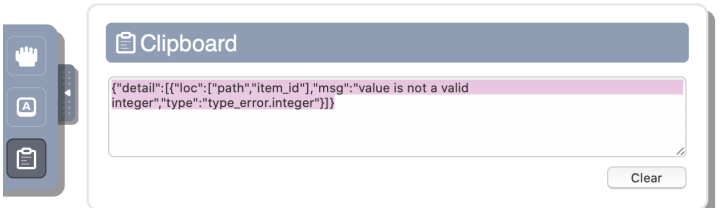
Answer 3: /api/



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"}]}



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

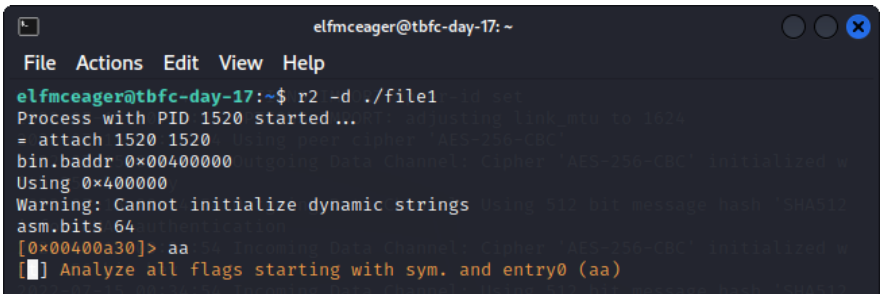
an odd number and is fortunately the answer for Question 6 and they key to finding out where Santa is

```
└─$ 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
```

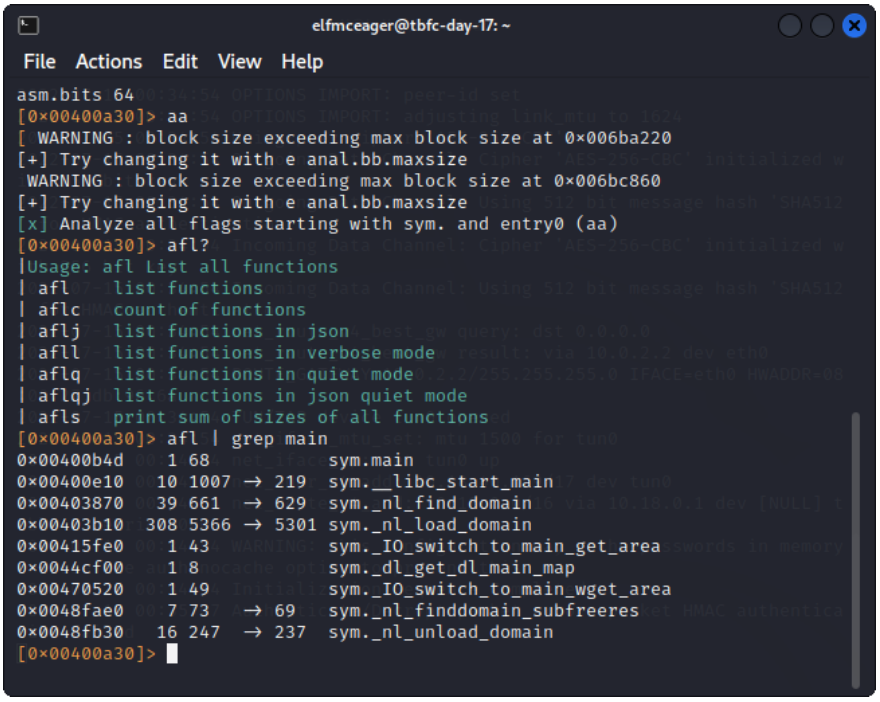
[Day 17] - Reverse Engineering - Reverse ELFneering

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



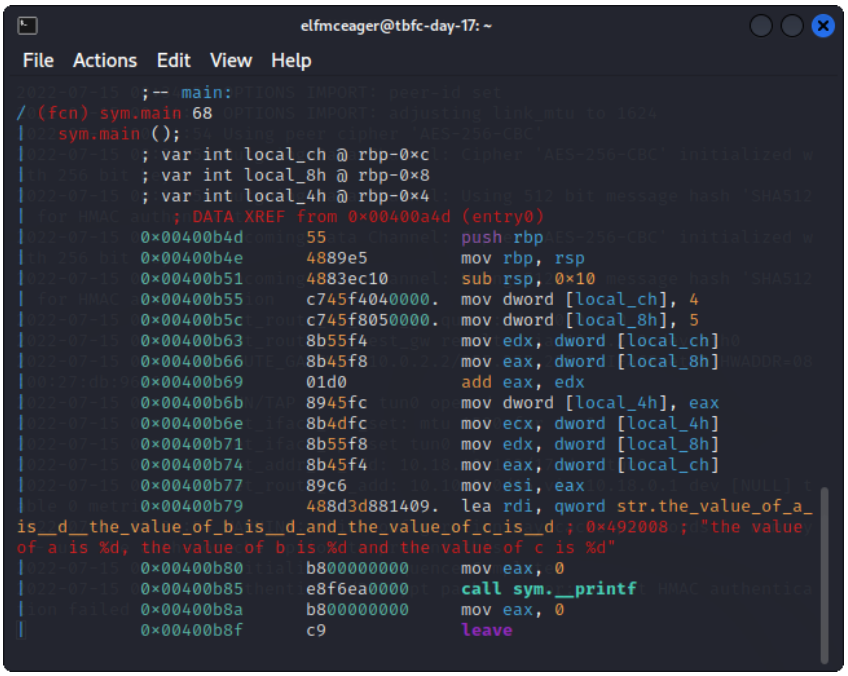
Once the analyzing completes we can seek out the entry point to the file.

afl | grep main



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

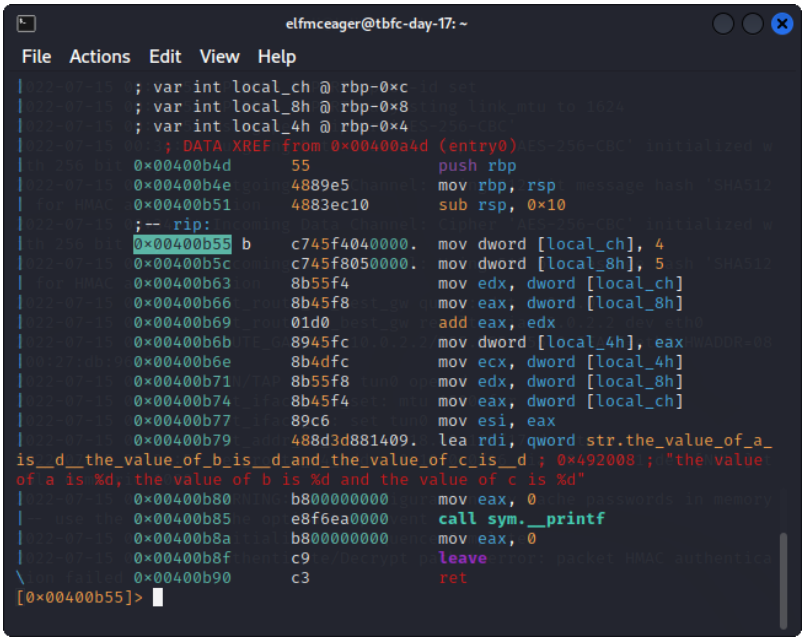


We can set a breakpoint with db.

db 0x00400b55

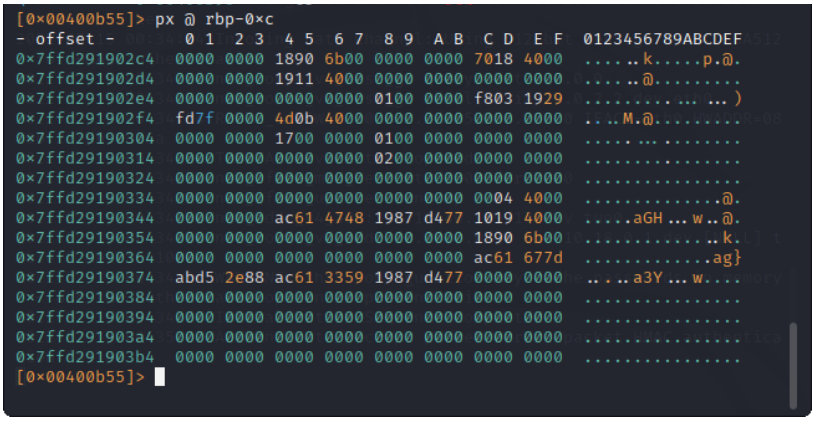
After setting our break point we, we can again run pdf @main and now we will see a lowercase b right after our break point location

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.

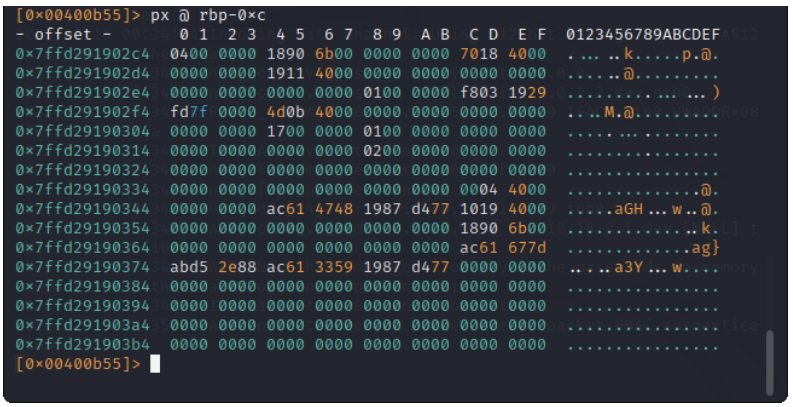


Next, to view the contents of, in our case, the local_ch variable, we will run px @memory-address

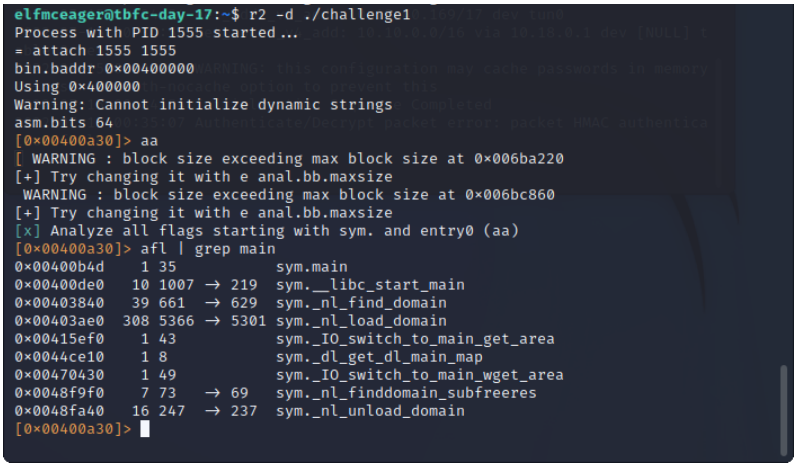
px @rbp-0cx



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



pdf @ main

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?

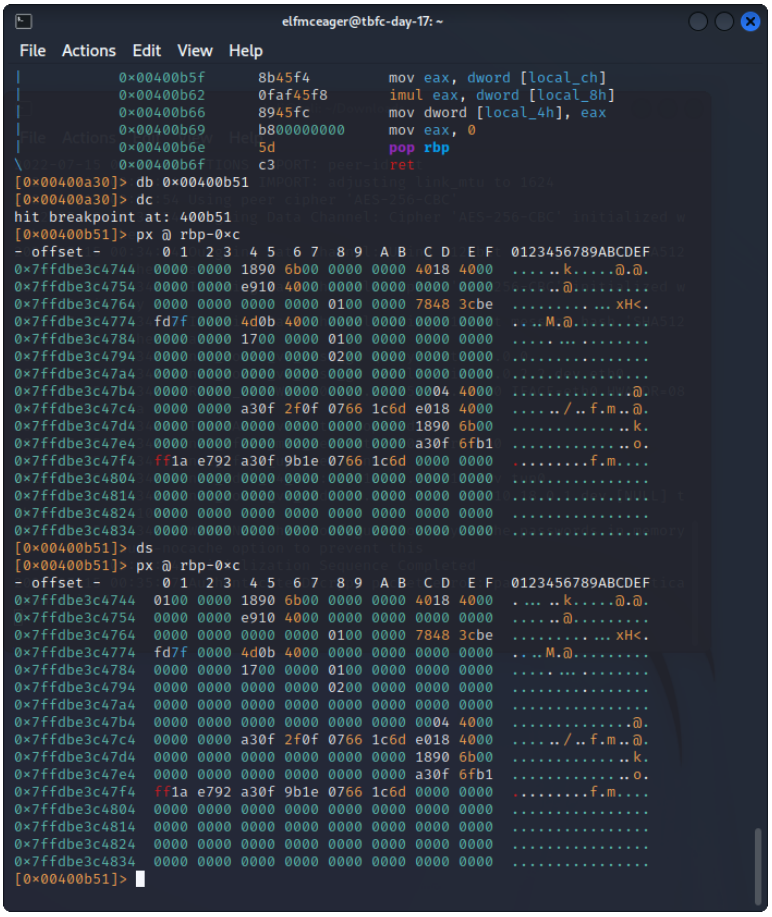
Ans : dc

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.

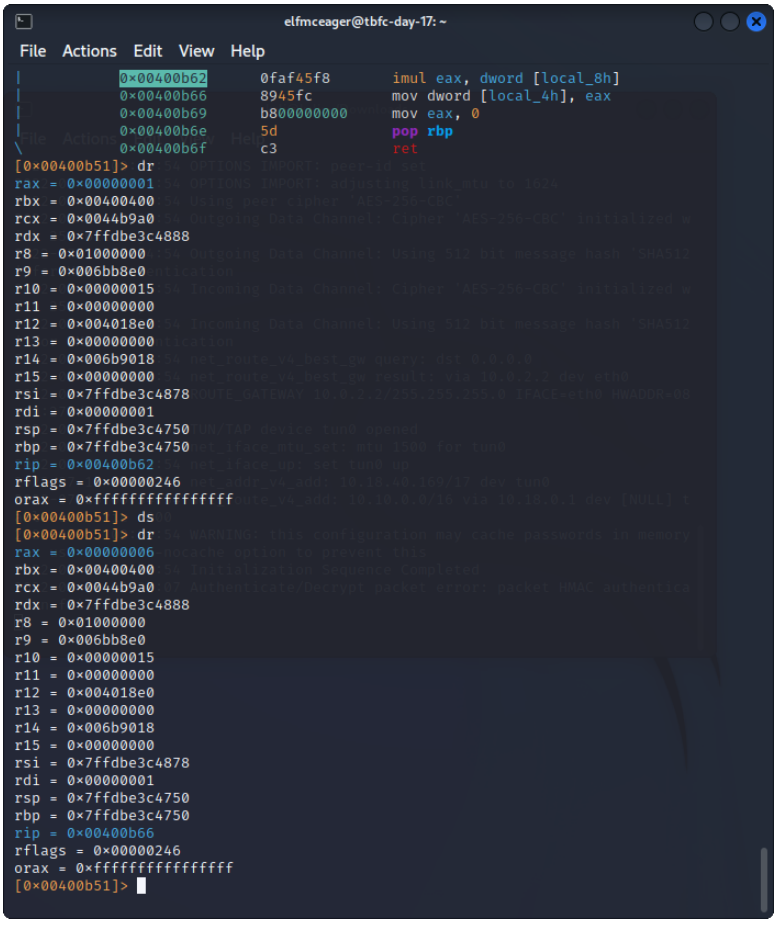


Question 6

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.



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.

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.StringToHGlobalAnsi(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.StringToHGlobalAnsi(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!

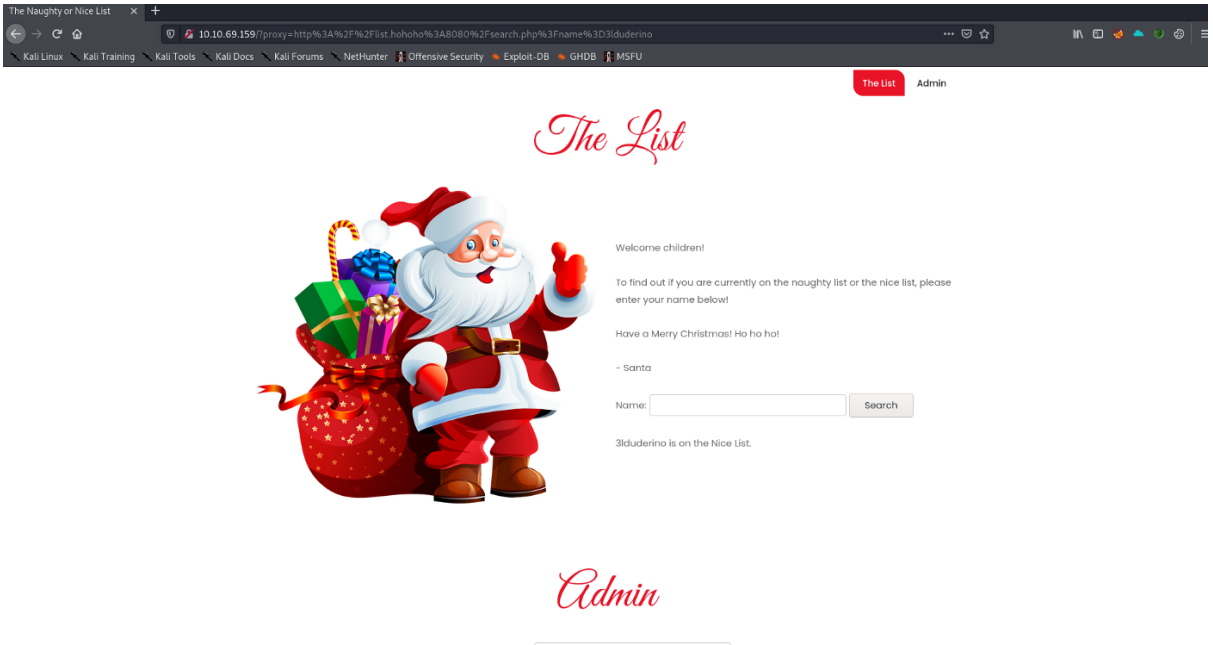
```
private unsafe void buttonActivate_Click(object sender, EventArgs e)
{
    IntPtr value = Marshal.StringToHGlobalAnsi(textBoxKey.Text);
    sbyte* ptr = (sbyte*)System.Runtime.CompilerServices.Unsafe.AsPointer(ref <Module>.??_C@_0BB@IKKDFEPG@santapassword321@);
    void* ptr2 = (void*)value;
    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 2: `thm{046af}`

Day 19: The Naught Or Nice List!

Question 1: What is Santa's password?

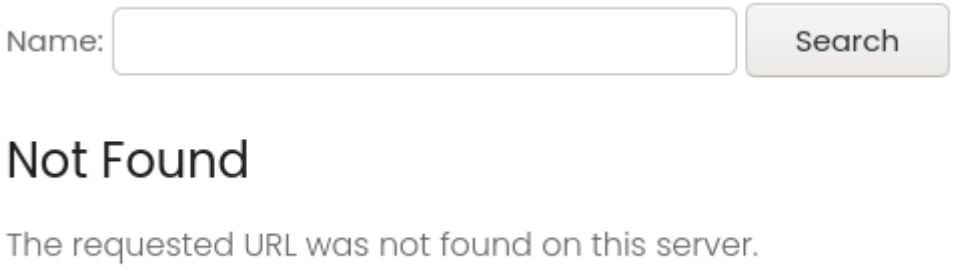
When we first visit the website, we see this form, along with an admin login underneath it.



The URL is intriguing: **http://10.10.69.159/**?proxy=**http%3A%2F%2Flist.hohoho%3A8080%2Fsearch.php%3Fname%3d3l 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.

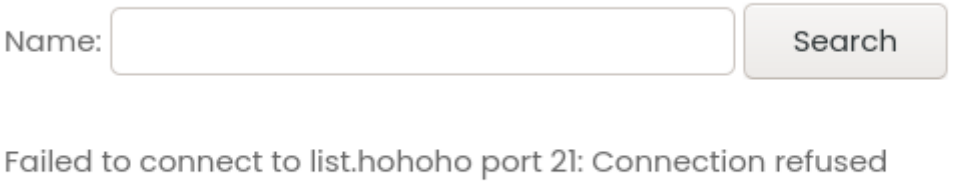
Burp did a short URL decoding and returned the following: **http://10.10.69.159/**?proxy=**http://list.hohoho:8080/search.php**?name=**3lduderino**

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).

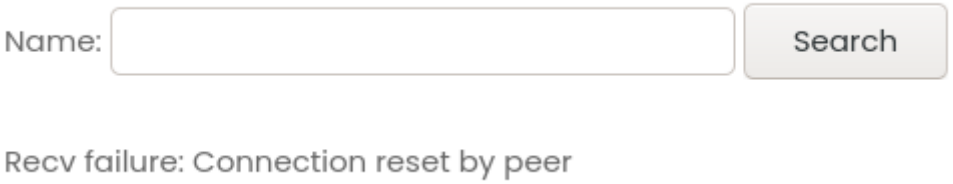


But for the time being, let's go on and see if there's another option.

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.



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



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

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

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

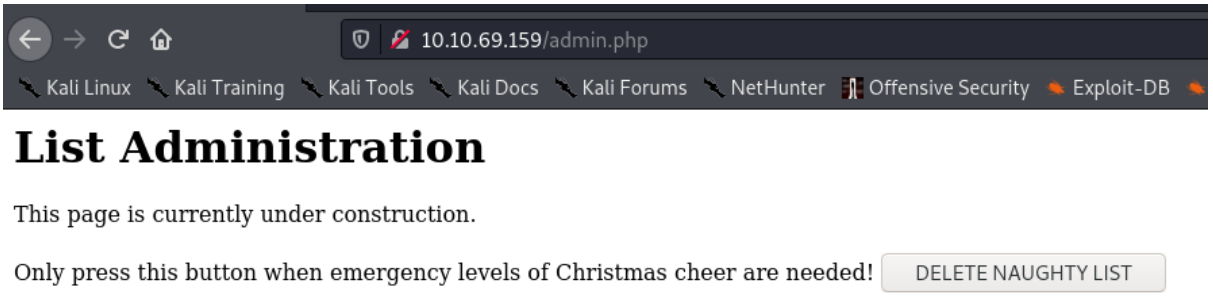
Santa,

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.



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

Day 20: PowershELIF 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 e1fone.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 .\e1fone.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                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                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\3lf
thr3e\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\3lf
r3e\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:redlyglvbfimc
\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
```

```
C:\Users\mceager\Desktop\elf2wo> Select-String -Path 'C:\Windows\System32\3lf
r3e\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.