

WriteUp ApoorvCTF 2025

Cyber Alliance

hush1a & mshazaw



Reverse Engineering - Holy Rice (Unfinished).....	2
OSINT - I Love Japan: Identity Game - 100 pts.....	3
OSINT - Sakura beads (Unfinished).....	4
OSINT - 404 Location Not Found - 385 pts.....	5
Forensics - Samurai Code (Unfinished).....	7
Binary Exploitation - “Kogarashi Cafe - The First Visit” (Unfinished).....	9

Reverse Engineering - Holy Rice (Unfinished)

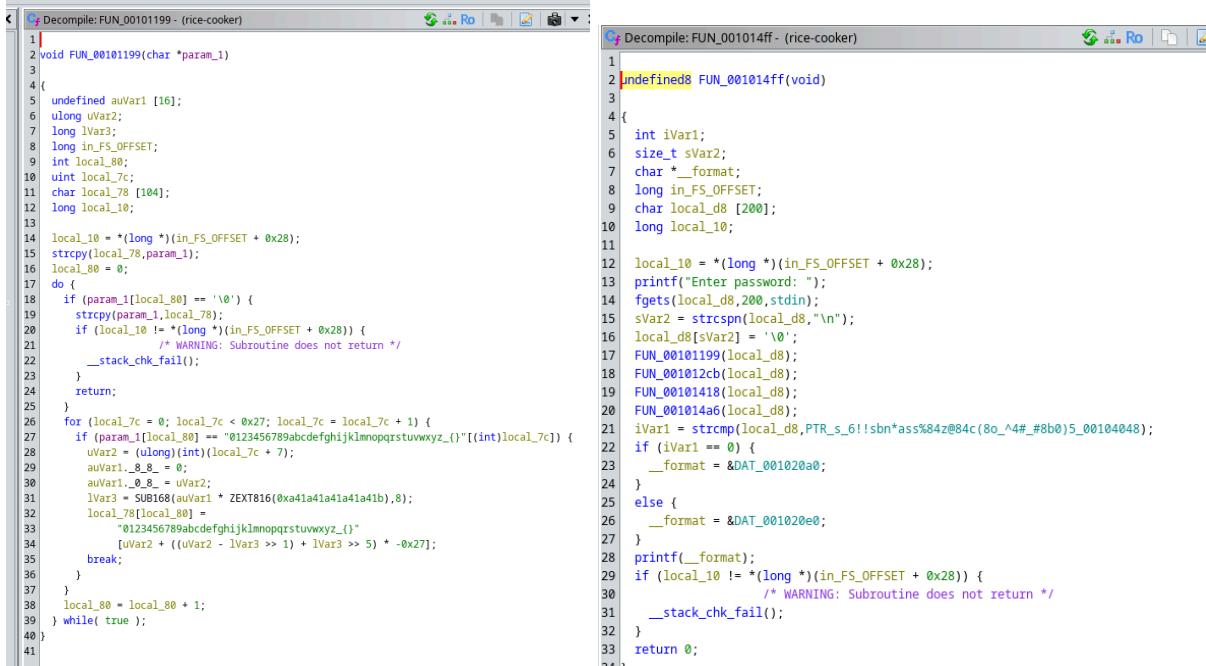
A wise monk made a "totally unhackable" locker to guard his holy rice. Spoiler: it wasn't. The pigeon mafia is already on it — crack it first and claim the rice for yourself.  

Author: hampter

```
└─ A └─ ~/Downloads/CTF/ApoorvCTF/rev
    └─ chmod +x rice-cooker

└─ A └─ ~/Downloads/CTF/ApoorvCTF/rev
    └─ ./rice-cooker
Enter password: test
no rice for you bro 
```

I added the option to execute the binary file, and it prompted me for a password.



The image shows two side-by-side code editors in Ghidra. The left editor contains the assembly code for function `FUN_00101199`, which includes a loop that reads a character from standard input, performs some calculations on local variables, and then compares the character against a specific value. The right editor contains the assembly code for function `FUN_001014ff`, which is a simple `printf` call followed by a `getchar` call to read a password from the user.

```
1 void FUN_00101199(char *param_1)
2 {
3     undefined auVar1 [16];
4     ulong lVar2;
5     long lVar3;
6     long in_FS_OFFSET;
7     int local_80;
8     uint local_7c;
9     char local_78 [104];
10    long local_10;
11
12    local_10 = *(long *)in_FS_OFFSET + 0x28;
13    strcpy(local_78,param_1);
14    local_80 = 0;
15    do {
16        if (param_1[local_80] == '\0') {
17            strcpy(param_1,local_78);
18            if (local_10 != *(long *)in_FS_OFFSET + 0x28) {
19                /* WARNING: Subroutine does not return */
20                _stack_chk_fail();
21            }
22        }
23        return;
24    }
25    for (local_7c = 0; local_7c < 0x27; local_7c = local_7c + 1) {
26        if (param_1[local_80] == "0123456789bcdefghijklmnopqrstuvwxyz_")[(int)local_7c] {
27            uVar2 = (ulong)(int)(local_7c + 7);
28            auVar1._8_8 = 0;
29            auVar1._0_8 = uVar2;
30            lVar3 = SUB168(auVar1 * ZEXT816(0xa41a41a41a41b), 8);
31            local_78[local_80] =
32                "0123456789bcdefghijklmnopqrstuvwxyz_";
33            uVar2 += ((uVar2 - lVar3 >> 1) + lVar3 >> 5) * -0x27;
34            break;
35        }
36    }
37    local_80 = local_80 + 1;
38 } while( true );
39
40 }
```

```
1 undefined8 FUN_001014ff(void)
2
3 {
4     int iVar1;
5     size_t sVar2;
6     char *_format;
7     long in_FS_OFFSET;
8     char local_d8 [200];
9     long local_10;
10
11    local_10 = *(long *)in_FS_OFFSET + 0x28;
12    printf("Enter password: ");
13    fgets(local_d8,200,stdin);
14    sVar2 = strcspn(local_d8,"\\n");
15    local_d8[sVar2] = '\\0';
16    FUN_00101199(local_d8);
17    FUN_001012cb(local_d8);
18    FUN_00101418(local_d8);
19    FUN_001014a6(local_d8);
20
21    iVar1 = strcmp(local_d8, PTR_S_6!!sbn*ass%84z@84c(80_4#.#8b0)5_00104048);
22    if (iVar1 == 0) {
23        __format = &DAT_001020a0;
24    }
25    else {
26        __format = &DAT_001020e0;
27    }
28    printf(__format);
29    if (local_10 != *(long *)in_FS_OFFSET + 0x28) {
30        /* WARNING: Subroutine does not return */
31        _stack_chk_fail();
32    }
33
34 }
```

When analyzing the code in Ghidra, I found this function in the main that seemingly does an encoding operation and also character shifting by 7. But there are other functions that I did not understand what they do.

There is also a charset for the password.

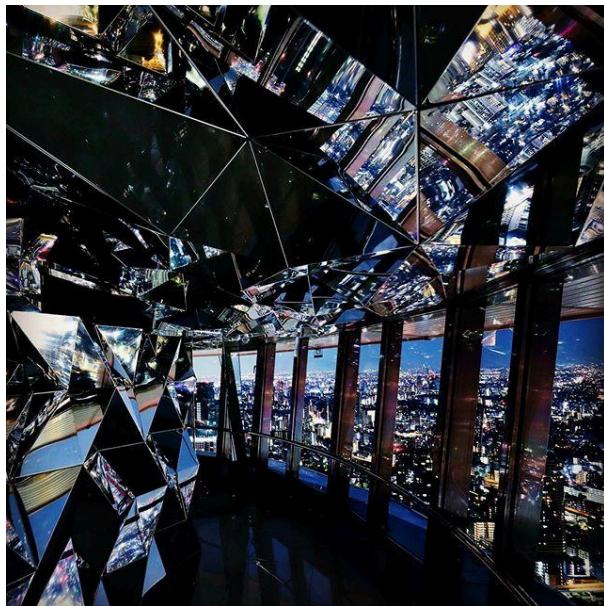
I tried to write a script to reverse the process but I couldn't find a string that the code compares with an inputted password, so :(

OSINT - I Love Japan: Identity Game - 100 pts

Oh! My friend send me this! So Pretty isn't it ^_^ I wonder who the designer could be?

Flag Format: apoorvctf{firstname_lastname} Note: The Flag is not a username/ nickname

Author: Stargazer



This picture is the Tokyo Tower, specifically the Top Deck.
When I searched for designers, a name came up : Kaz Shirane.



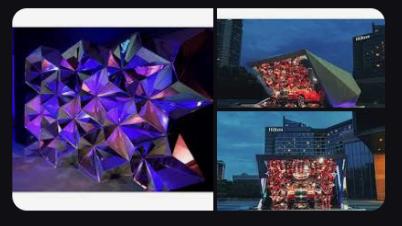
itsmachas.com

<https://www.itsmachas.com> › talent › view › kaz-shirane

⋮

Kaz Shirane - Machas

Internationally renowned artist Kaz Shirane (Masakazu Shirane) operates in spatial art and interior design. Trained as an architect, Kaz Shirane explores the ...



After searching his name, I found that it was a professional name and his real name is actually Masakazu Shirane.

Flag: **apoorvctf{Masakazu_shirane}**

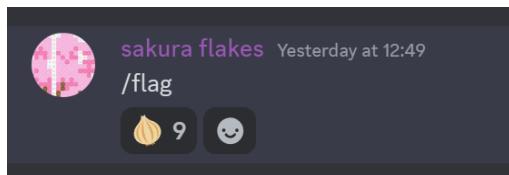
OSINT - Sakura beads (Unfinished)

Hey everyone, I have a friend Sakura. She wanted to learn how programming works so I gave her the best advice anyone could have given her- participate in ApoorvCTF.

She's trying her best to find flags and could only find the welcome flag, sending /flag XD.

If you consider yourself real OSINTers — stalk her.

Author: cooker



Found the discord account, which has a link to her reddit.

A screenshot of three consecutive Reddit posts by the user 'u/flaky_sakura'.

- The first post is titled 'Perfume madness' and discusses the user's frustration with perfume, mentioning spending \$150 on a niche, artisanal fragrance that smells like a 'fairy's whisper'. It has 1 upvote and 18 comments.
- The second post is titled 'Uhhh Lip Gloss' and discusses the user's dislike of lip gloss, calling it a 'scam' that causes hair damage and disappears mysteriously. It has 1 upvote and 1 comment.
- The third post is titled 'pls help with python code' and asks for help with a Python project. It includes a screenshot of a terminal showing a 'SyntaxError: invalid syntax' error. The code in the screenshot is:

```
rr(): try: response = requests.get(api_url)
```

Bunch of rants,

A screenshot of a single Reddit post by the user 'u/flaky_sakura'. The post is titled 'pls help with python code' and contains the same rant and Python code as the third post in the previous screenshot. The terminal output shows a 'SyntaxError: invalid syntax' error.

```
error here:  
rr(): try: response = requests.get(api_url)
```

This post was an odd one, but I couldn't find anything :/

OSINT - 404 Location Not Found - 385 pts

My friend Daniel S loved traveling. He used to send me pictures of his trips across Europe. Here's one from his favourite trail. I want to visit the place in his memory, help me find it.

flag_format: `apoovctf{<coordinates>}`

coordinates should be truncated to the 4th decimal position

Example:

Eiffel Tower

Coordinates: 48.8583951,2.2949823

Flag: apoovctf{48.8583,2.2949}

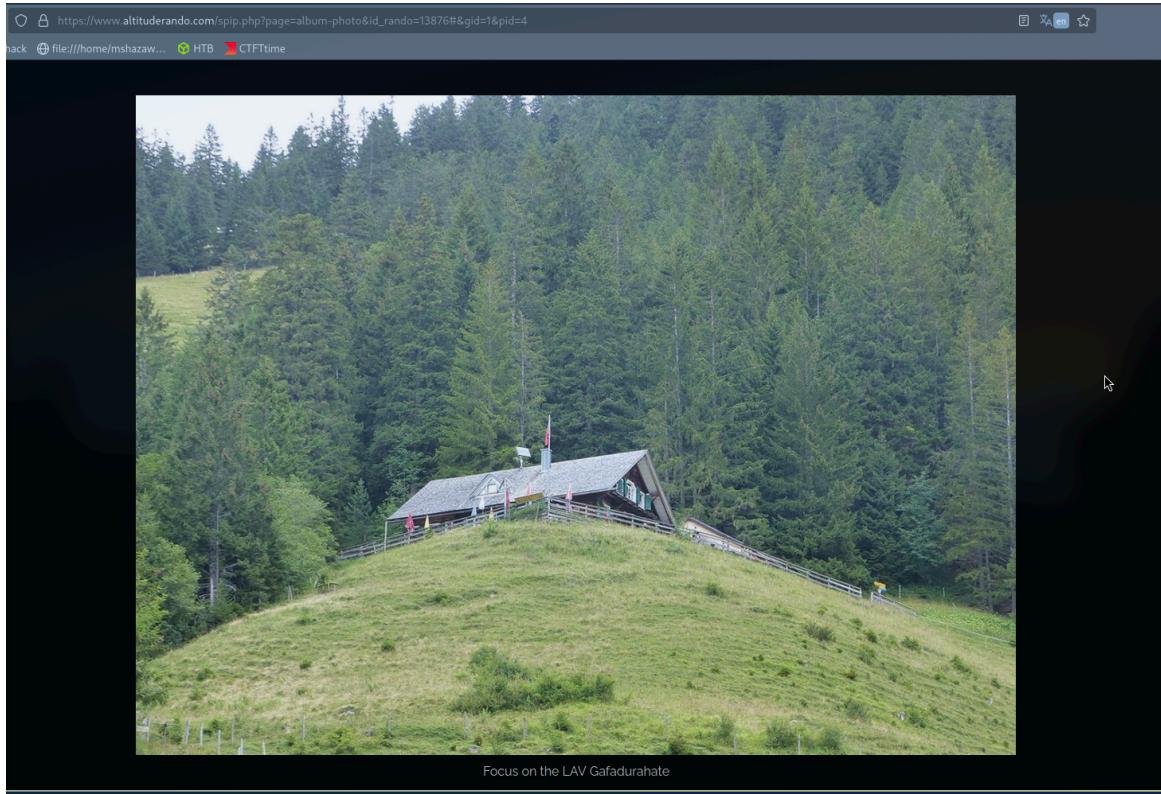
Author: cooker



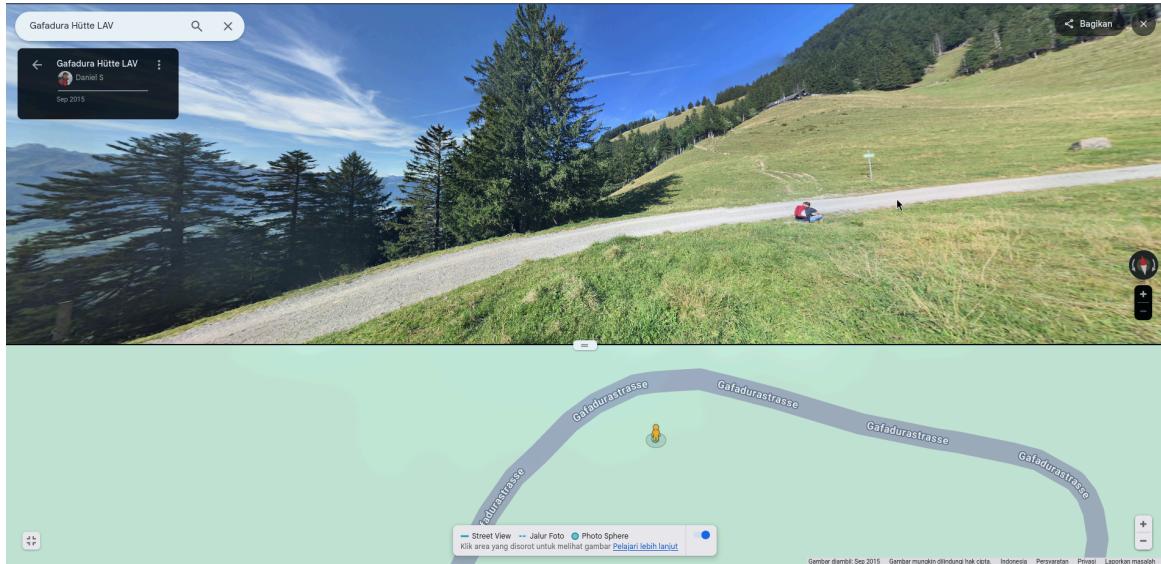
So we were initially given this image and at first aside from it looking like it's in the Alps we have nothing to go on.



Then I noticed this hut in the corner of the image and tried reverse image searching it using google lens.



And that led me to this photo album which mentions the name LAV Gafadurahate and that was it.



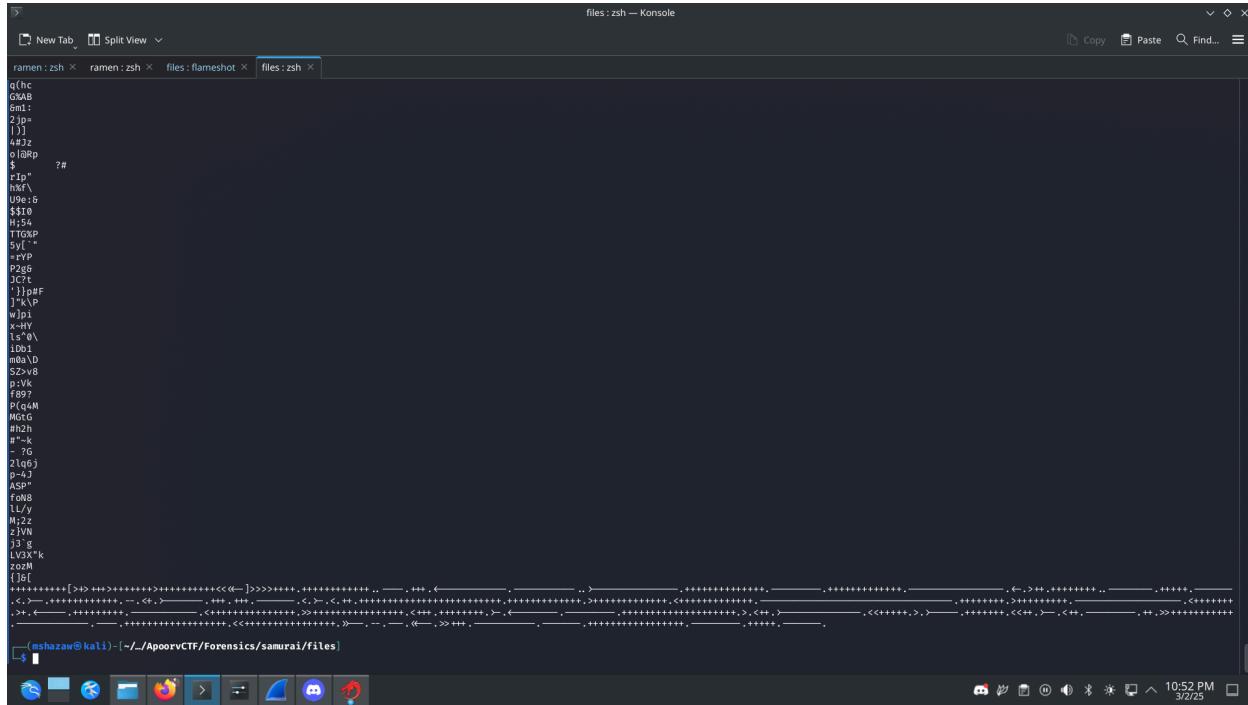
All I needed to do from then on was just search it up on google maps and try the placeable spots on google maps one by one.

Flag: **apoovctf{47.1851,9.5663}**

Forensics - Samurai Code (Unfinished)

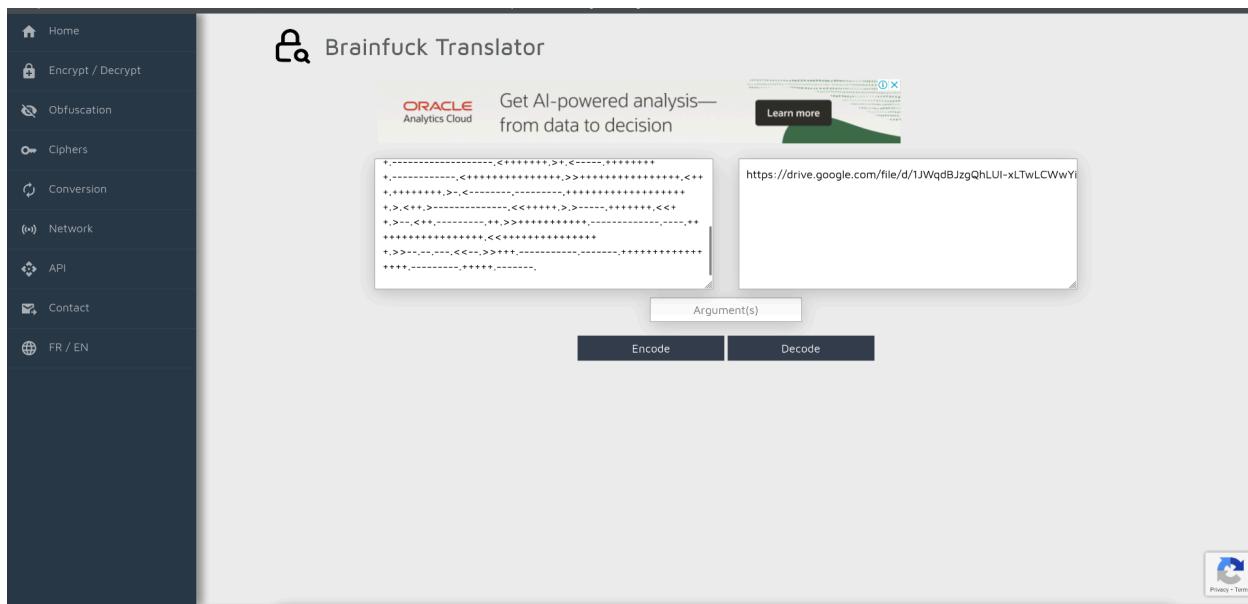
Unveil the lost code of the Samurai and unlock the mystery hidden within.

Ok for this challenge, the first thing we do is just try and perform the command strings on the file sam.jpg and we receive brainfuck code on the bottom.



The screenshot shows a terminal window titled "files : zsh — Konsole". The terminal has multiple tabs open: "ramen : zsh", "ramen : zsh", "files : flameshot", and "files : zsh". The current tab displays a large amount of brainfuck code. Below the code, the prompt shows the user is in a directory named "samurai" under "/~/.ApoorvCTF/Forensics". The terminal window is part of a desktop environment with icons for Home, File, Mail, and a browser at the bottom.

When we try and decode the code in a brainfuck translator we see that it leads to a google drive which gives us a file named samurai.



The screenshot shows the "Brainfuck Translator" application interface. On the left is a sidebar with various tools: Home, Encrypt / Decrypt, Obfuscation, Ciphers, Conversion, Network, API, Contact, and FR / EN. The main area features the title "Brainfuck Translator" with a logo of a brain inside a hexagon. Below the title is an Oracle Analytics Cloud advertisement. The central part of the interface contains a large text area with brainfuck code, a smaller text area with a URL, and two buttons at the bottom labeled "Encode" and "Decode". The URL in the smaller text area is <https://drive.google.com/file/d/1JWqdBJzgQhLUI-xLTwLCWwY>.

Upon performing basic analysis of this file, we see that it's just data and we don't even know anything about it other than the size.

```
(mshazaw㉿kali)-[~/.../ApoorvCTF/Forensics/samurai/files]
└─$ file samurai
samurai: data

(mshazaw㉿kali)-[~/.../ApoorvCTF/Forensics/samurai/files]
└─$ exiftool samurai
ExifTool Version Number      : 13.10
File Name                   : samurai
Directory                   : .
File Size                   : 217 kB
File Modification Date/Time : 2025:03:02 21:26:06+07:00
File Access Date/Time       : 2025:03:02 21:26:07+07:00
File Inode Change Date/Time: 2025:03:02 21:26:06+07:00
File Permissions            : -rw-rw-r--
Error                       : Unknown file type

(mshazaw㉿kali)-[~/.../ApoorvCTF/Forensics/samurai/files]
└─$
```

The next thing I tried to check was the hexdata, I noticed that this looked like a badly made jpg file so I tried to go down that route.

```
00000000 FF 08 DD E1 00 BC 45 78 69 66 00 00 01 01 48 00
00000010 46 00 00 00 00 BB FF A3 00 01 00 01 01 01 01 01 01 01
00000020 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000030 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000040 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000050 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000060 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000070 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000080 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000090 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
000000A0 11 00 03 08 01 04 03 FA 11 01 02 00 01 13 13 03
000000B0 FF 01 00 C6 00 1E 01 00 03 04 01 01 00 01 00 00
000000C0 00 00 00 00 00 00 03 00 05 04 01 06 07 02 09 00
000000D0 FF 0A 00 C4 01 1B 03 00 01 01 01 01 00 01 00 00
000000E0 00 00 00 00 00 00 02 00 01 00 05 04 01 06 DA F7
000000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000100 04 11 93 04 05 BC A9 91 00 00 06 A0 01 28 C4 1D
00000110 14 06 48 C1 07 01 00 18 5A 0C 59 B0 00 E6 00 00
00000120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 45 94
00000130 41 2C 00 86 4F 18 13 A3 00 64 44 01 81 D5 30 D3
00000140 00 00 D1 37 02 83 03 05 54 05 A9 76 80 09 28 00
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000160 C9 17 00 10 00 00 00 00 00 00 00 00 00 00 00 00
00000170 00 00 13 02 53 A2 AC B1 00 60 90 FB 03 F9 88 6A
00000180 AC 00 B8 9A 31 AD 00 C6 88 34 46 B5 00 0C 0A 00
00000190 BC 83 33 31 00 AD 40 01 43 01 20 0A 0A 06 51 44
000001A0 20 28 00 00 00 AD F8 BE A8 00 4C 00 00 00 00 00
000001B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001C0 13 00 E4 EC 89 4C 18 00 94 26 30 01 00 00 A1 14
000001D0 00 1E 02 00 3A CA 04 50 00 C0 00 14 09 C5 68 14
000001E0 52 00 90 42 00 A0 18 00 20 B5 26 B5 00 00 00 14
000001F0 00 00 00 00 00 00 00 00 00 CE 00 C6 80
00000200 98 C5 00 00 F6 09 26 72 03 5C 02 02 40 78 00 13
00000210 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000220 05 05 A0 28 29 18 44 21 30 A1 00 00 00 17 17 3A
00000230 00 98 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000240 23 00 23 3A 15 1B 60 62 E2 03 E6 76 EC 4D 02 00
00000250 AD 89 4C 00 00 00 08 BD C0 60 00 00 00 B2 34 B0
00000260 00 4F 50 00 14 03 51 24 01 80 00 49 80 42 00 00
00000270 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000280 00 00 00 00 00 00 11 90 00 00 00 00 00 00 00 00
00000290 6A 00 60 30 4F 4C 02 68 00 60 5E 00 06 C3 00 00
000002A0 90 05 81 75 78 A2 02 00 02 80 94 82 0C 50 98 14
000002B0 50 A2 60 98 62 00 82 D5 30 CF 00 00 00 00 00 00
000002C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002E0 47 28 00 60 00 C3 3A 1A 59 00 00 00 00 8A 62
000002F0 30 8A 29 00 45 21 00 50 01 C0 5B F1 7C 41 00 98
00000300 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 23 00
^G Help ^C Exit (No Save) ^T goto Offset ^X Exit and Save ^W Search ^U Undo ^L Redraw ^E Text Mode ^R CharSet ^P Spacing ^N Find Next ^S Color
```

But after multiple tries, I still couldn't fix the file and that's where I threw in the towel.

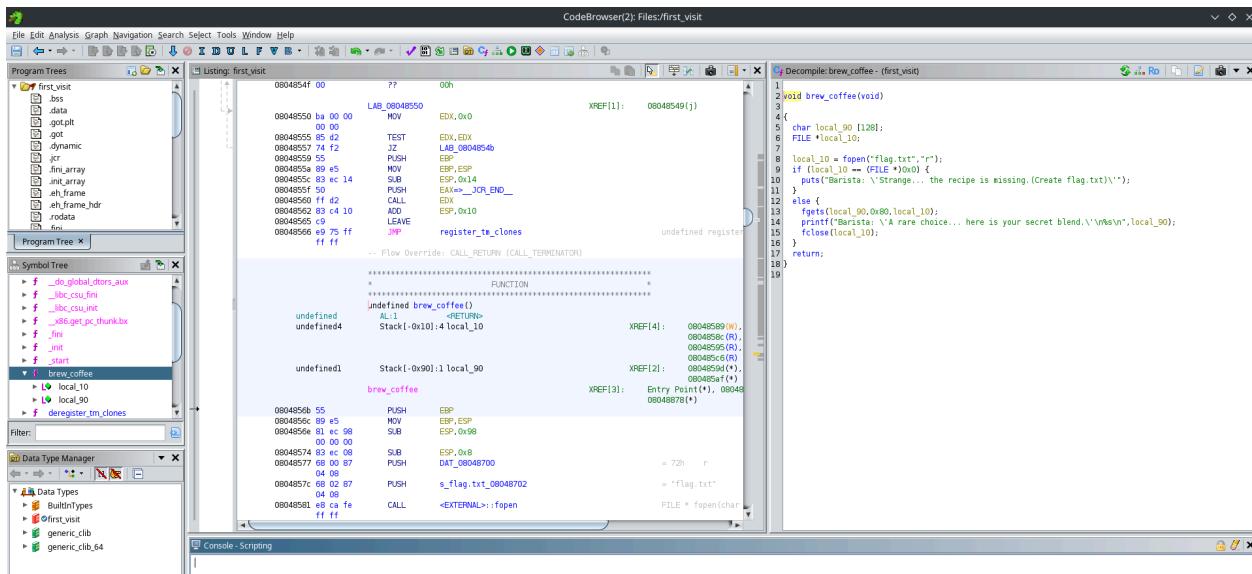
Binary Exploitation - “Kogarashi Café - The First Visit” (Unfinished)

A quiet café, a simple question. The barista waits, but your order may mean more than it seems.

```
nc chals1.apoorvctf.xyz 3001
```

```
(mshazaw㉿kali)-[~/.../CTFComps/ApoorvCTF/BinEx/files]
$ ./first_visit
Welcome to Kogarashi Café.
Barista: 'What will you have?'
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Barista: 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa ... interesting choice.'
Brewing ...
zsh: segmentation fault (core dumped) ./first_visit
(mshazaw㉿kali)-[~/.../CTFComps/ApoorvCTF/BinEx/files]
$
```

This one is supposed to be a pretty standard binary exploitation challenge, where you overload the buffer and get the flag.



When looking at the code in ghidra, we can see the brew function which in this challenge is the win function. We now have to find a way to get here.

CodeBrowser(2): Files/first_visit

Listing: first_visit

```

004861b 83 ec 08 SUB    ESP,0x8
004861e 8d 45 48 LEA    EAX,[local_2c].[ESP + -0x28]
0048621 50 PUSH   EAX
0048622 68 c4 87 PUSH   s_Barista,_%..._interesting_choi_080487c = "Barista: '%...
0048625 c3 RET
0048627 e8 c4 fd CALL   <EXTERNAL>:_printf
ff ff
004862f 8d 45 10 ADD    ESP,0x10
0048632 f3 83 <c 0c SUB    ESP,0xc
0048632 68 ea 87 PUSH   s_Brewing,..._080487e = "Brewing..."
0048637 48 f4 fd CALL   <EXTERNAL>:_puts
ff ff
004863c 83 c4 10 ADD    ESP,0x10
004863f 48 NOP
0048640 c9 LEAVE
0048641 c3 RET

***** FUNCTION *****
undefined main(undefined param_1)
AL1: <RETURN>
Stack[0x4]:1 param_1
Stack[0x0]:4 local_res0
XREF[1]: 08048641(*)
XREF[2]: 08048649(R)
XREF[3]: 08048674(*)
XREF[4]: 08048670(R)
main:
Stack[-0x4]:4 local_c
main:
XREF[1]: Entry Point(*)
XREF[2]: start:0804867(*), 08048688(*)

0048642 8d 4c 24 04 LEA    EAX,<param_1>,[ESP + 0x4]
0048645 ff 71 fc PUSH   dword ptr ECX + local_res0
0048649 55 PUSH   EBSP
004864a 48 NOP
004864f 51 PUSH   ECX
0048650 83 ec 04 SUB    ESP,0x4
0048653 a1 34 a0 MOV    AX,[stdeout]

```

This one is the main function, and as we can see there's no path from it to the brew function and it only calls the order_coffee function.

CodeBrowser(2): Files/first_visit

Program Trees

Symbol Tree

order_coffee(void)

```

void order_coffee(void)
{
    char local_2c [40];
    puts(DAT_08048780);
    puts("Barista: \What will you have?\n");
    gets(local_2c);
    printf("Barista: \%s... interesting choice.\n",local_2c);
    puts("Brewing...");
    return;
}

***** FUNCTION *****
undefined order_coffee()
AL1: <RETURN>
Stack[-0x2c]:1 local_2c
order_coffee:
XREF[1]: 08048501(*)
XREF[2]: 08048514(*)
order_coffee:
XREF[3]: 08048514(*)
XREF[4]: Entry Point(*), main:0804863d(R)
XREF[5]: 08048514(*)
XREF[6]: 08048514(*)
XREF[7]: 08048514(*)
XREF[8]: 08048514(*)
XREF[9]: 08048514(*)
XREF[10]: 08048514(*)
XREF[11]: 08048514(*)
XREF[12]: 08048514(*)
XREF[13]: 08048514(*)
XREF[14]: 08048514(*)

```

And this is the order_coffee function, again with no calling of the brew function.

```

files : zsh — Konsole
New Tab Split View
ramen:zsh × ramen:zsh × files:zsh ×
0x0000000000400000 fopen@plt
0x0000000000400460 __gmon_start__@plt
0x0000000000400470 _start
0x0000000000400480 .x86.get_pc_thunk.bx
0x0000000000400490 deregister_tm_clones
0x0000000000400498 do_global_dtors_aux
0x0000000000400520 __frame_dummy@@plt
0x0000000000400550 brew_coffee
0x0000000000400560 ordr_coffee
0x0000000000400562 main
0x0000000000400650 __libc_csu_init
0x0000000000400660 __libc_csu_fini
0x0000000000400664 __fini
junk#> q

[mshzaw@kali]:~/CTFCmps/ApoorvCTF/BinEx/files]
└$ python3 -c 'import struct, sys; sys.stdout.buffer.write(b"A"*40 + b"\x86\x85\x04\x08")' | ./first_visit
Welcome to Kogarashi Caf .
Barista: 'What will you have?'
Barista: 'AAAAAAAAAAAAAAAAAAAAAAA... interesting choice.'
Brewing...
zsh: done                  python3 -c |
zsh: segmentation fault (core dumped) ./first_visit

[mshzaw@kali]:~/CTFCmps/ApoorvCTF/BinEx/files]
└$ python3 -c 'import struct, sys; sys.stdout.buffer.write(b"A"*40 + b"\x86\x85\x04\x08")' | nc chals1.apoorvctf.xyz 3001
Welcome to Kogarashi Caf .
Barista: 'What will you have?'
^C

[mshzaw@kali]:~/CTFCmps/ApoorvCTF/BinEx/files]
└$ python3 -c 'import sys; sys.stdout.buffer.write(b"A"*40 + b"\x86\x85\x04\x08")' | nc chals1.apoorvctf.xyz 3001
Welcome to Kogarashi Caf .
Barista: 'What will you have?'
^C

[mshzaw@kali]:~/CTFCmps/ApoorvCTF/BinEx/files]
└$ nc chals1.apoorvctf.xyz 3001
Welcome to Kogarashi Caf .
Barista: 'What will you have?'
AAAAAAAAAAAAAAAAAAAAAAA... interesting choice.
Barista: 'AAAAAAAAAAAAAAAAAAAAAAA... interesting choice.'
Brewing...

[mshzaw@kali]:~/CTFCmps/ApoorvCTF/BinEx/files]
└$ 

```

Because of the get() vulnerability we can write as many bytes as possible into the answer and they won't check it. So I tried to overflow the buffer and then afterwards redirect it to the brew_coffee function, but all my tries ended in failure, so I threw in the towel for this one.