BlueHens UDCTF 2024 Writeup

Challenge: Welcome Letter

Category: MISC

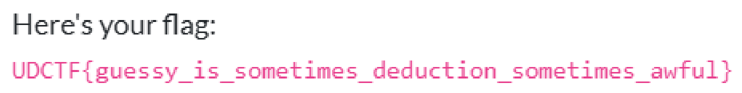
Solution:

Provided was a Welcome Letter with a few notes about the CTF

An important note was some authors used udctf{} others used UDCTF{}

It's important to read all instructions because it allowed me to assist other team members in understanding what to search for in other challenges, lowercase and uppercase format.

Flag was at end of the message.



Here's your flag:

```
UDCTF{guessy_is_sometimes_deduction_sometimes_awful}
```

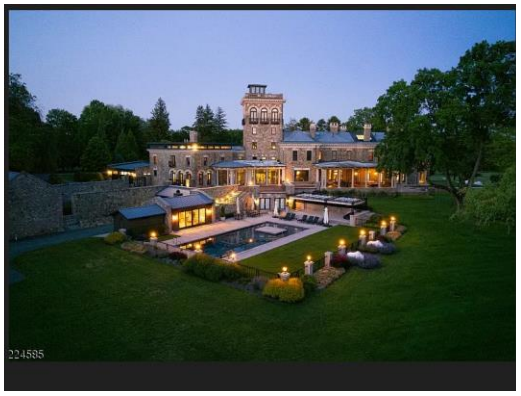
Challenge: Training Problem Intro to OSINT

Category: OSINT

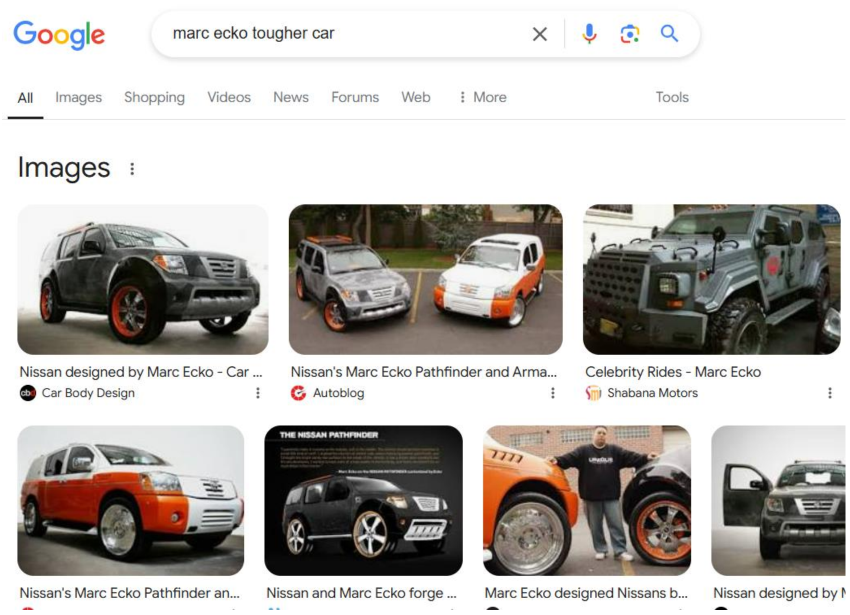
Description: A famous person is selling their house. In this market, who wouldn't? Can you tell me who owns this house, and what the license plate of their "tough" car is? Flag format: udctf{FirstLast_licenseplate}

Solution:

Provided image to download called osint1.png



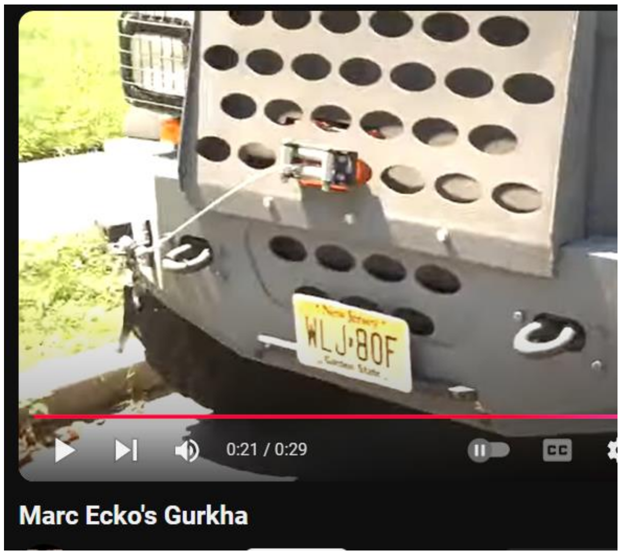
First I check the info of image. Then I reverse search image using google. A name of Marc Ecko appears as homeowner. I google "Marc Ecko tougher car".


marc ecko tougher car
All
Images
Shopping
Videos
News
Forums
Web
More
Tools
Images
Nissan designed by Marc Ecko - Car ...
Car Body Design
Nissan's Marc Ecko Pathfinder and Arma...
Autoblog
Celebrity Rides - Marc Ecko
Shabana Motors
Nissan's Marc Ecko Pathfinder an...
Nissan and Marc Ecko forge ...
Marc Ecko designed Nissans b...

What car appears tough?

The more gangster one or the SWAT looking vehicle?

The swat looking vehicle. I then reverse search image that SWAT looking vehicle. I discover the name of vehicle, Gurkha. The licencse plate in all images are blurred. I then google "the gurhka marc ecko". A youtube video appears with his license plate number.



Piecing together the flag: udctf{MarcEcko_wlj80f}

Challenge: Whispers of the Feathered Messenger

Category: FORENSICS

Description: In a world where secrets flutter through the air, the bluehen carries a hidden message. A message that has been salted.... however its still a message... maybe the bluehen ignores the salt. This image holds more than meets the eye.

shasum: e717eefe9b41212b017152756b0e640f9a4f3763
bird.jpeg

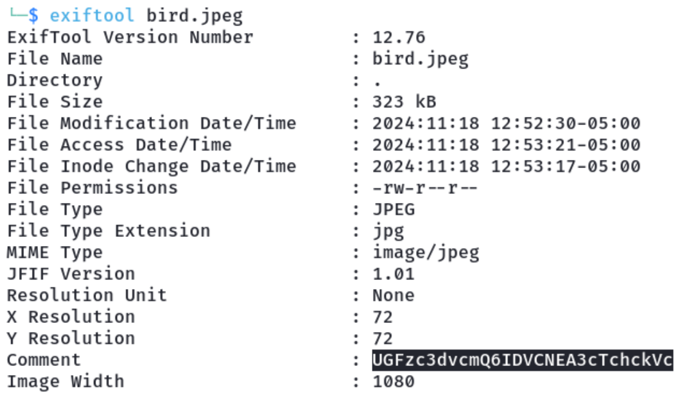
Solution:

Download image.

I check file info, open image in a notepad and ctrl+f keywords like udctf, maybe flag is in notes. Then follow the steganography steps from https://georgeom.net/StegOnline/checklist

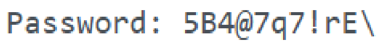
1. Just to be sure what file you are facing with, check its type with `file filename`.
2. View all strings in the file with `strings -n 7 -t x filename.png`. We use `-n 7` for strings of length 7+, and `-t x` to view- their position in the file.
3. Exiftool to check all metadata
4. Binwalk to check image for hidden embedded files. My preferred syntax is `binwalk -Me filename.png`. -Me is used to recursively extract any files.
5. Found a password (or not) I use steghide to extract data `steghide extract -sf filename.png`.
6. Java -jar stegsolve.jar used to explore colour & bit planes (Full Red, Inverse, LSB etc). Im looking static at the top of any planes. This tool also allows checking of RBGA values

Exiftool metadata has a comment on the image written in Base64 that's interesting.


```
└─$ exiftool bird.jpeg
ExifTool Version Number         : 12.76
File Name                       : bird.jpeg
Directory                       : .
File Size                       : 323 kB
File Modification Date/Time     : 2024:11:18 12:52:30-05:00
File Access Date/Time           : 2024:11:18 12:53:21-05:00
File Inode Change Date/Time     : 2024:11:18 12:53:17-05:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : None
X Resolution                    : 72
Y Resolution                    : 72
Comment                         : UGFzc3dvcmQ6IDVCNEA3cTchckVc
Image Width                     : 1080
```

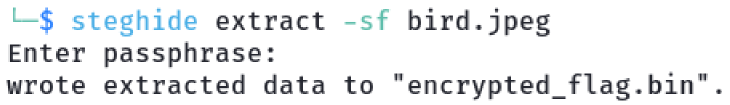
Decoding methods:

Base64 -> plaintext: Unfruitful, I receive more base64 which I run through the decoder again to no avail.

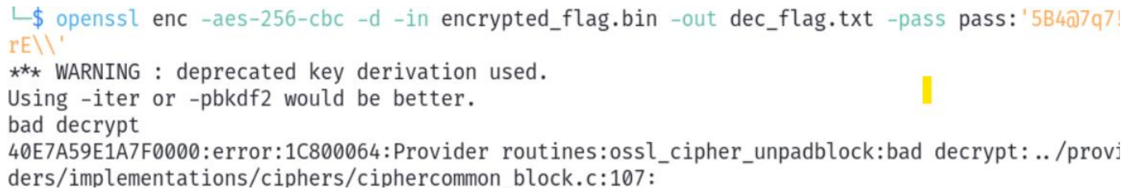

```
Password: 5B4@7q7!rE\
```

Base64 -> binary -> plaintext:

Decoded message was a password so I use the tool steghide to extract data


```
└─$ steghide extract -sf bird.jpeg
Enter passphrase:
wrote extracted data to "encrypted_flag.bin".
```

I check file type of encrypted_flag.bin which describes it as “openssl enc’d data with salted password”. Using binwalk tool against file I check description and the salt is revealed. I use the OpenSSL tool to decrypt the file.


```
└─$ openssl enc -aes-256-cbc -d -in encrypted_flag.bin -out dec_flag.txt -pass pass:'5B4@7q7!
rE\\'
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
bad decrypt
40E7A59E1A7F0000:error:1C800064:Provider routines:ossl_cipher_unpadblock:bad decrypt:../provi
ders/implementations/ciphers/ciphercommon block.c:107:
```

What was I doing wrong? Can you spot it?

TLDR: I only needed done backlash.

I tried the command again with -pbkdf2 but that was also wrong. I felt I was close to decrypting this file but needed to research more how to use openssl. I reached out to a team member and said “am I headed in the right direction” , showing him this command snippet, which the team member replied “its not pbkdf2 and your password is slightly wrong”

My error was I thought, in most shell env, the backlash escapes the next character, and if nothing follows it, it escapes the newline character.

I concatenate the dec_flag.txt file and receive flag:

UDCTF{m0AybE_YoR3$!_a_f0recnicsEs_3xpEr^t}

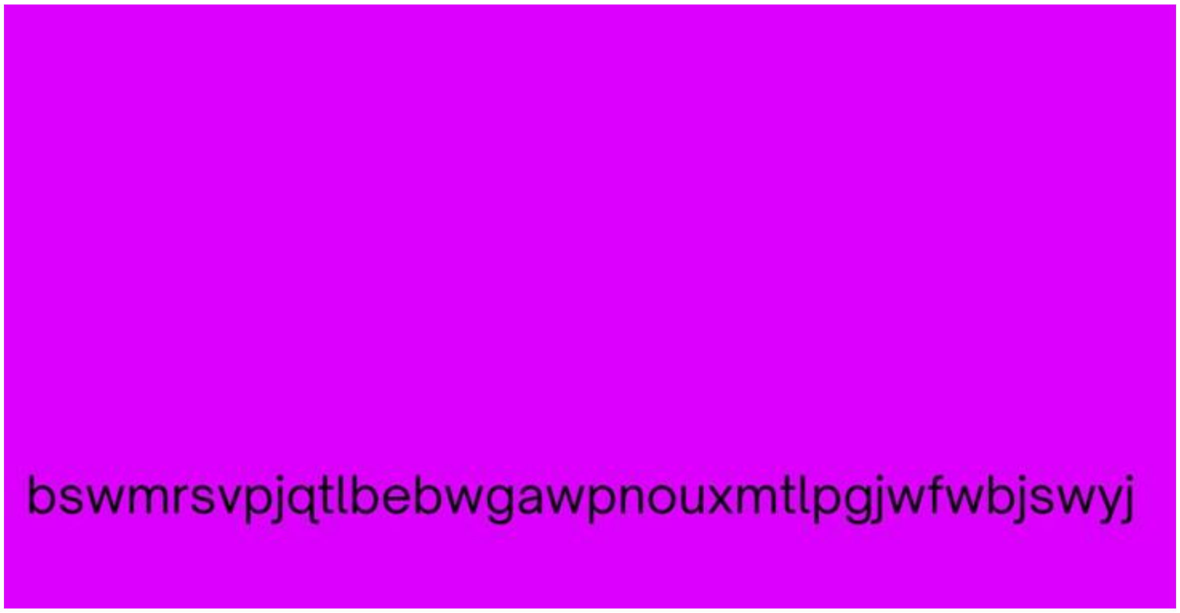
Challenge: I’m Hungry

Category: OSINT

Description: Google is your frienda_paper_football_player.jpg

Solution:

Download image titled “a_paper_football_player”

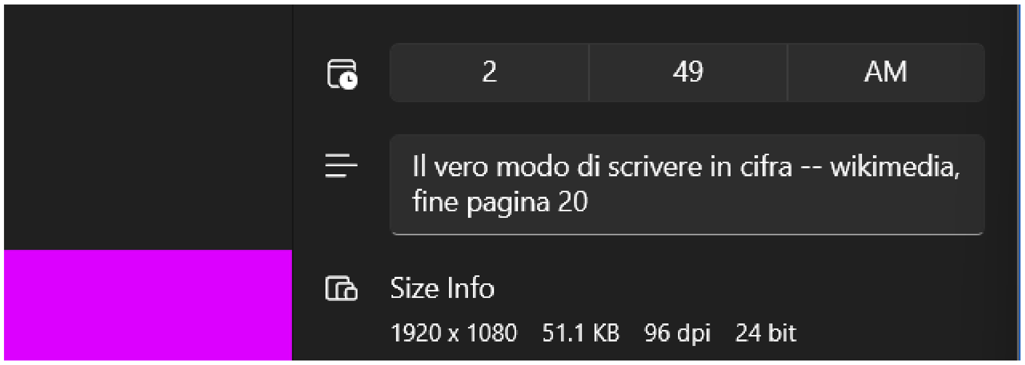


The text on the image is copied down:

Bswmrsvpjqtlbebwgawpnouxmtlpgjwfwbjswyj

Checked if it could be decoded from base64 to plaintext? Unfruitful

Foundational checks (*1), picture info, discover a comment in Italian:



2 49 AM

Il vero modo di scrivere in cifra -- wikimedia, fine pagina 20

Size Info

1920 x 1080 51.1 KB 96 dpi 24 bit

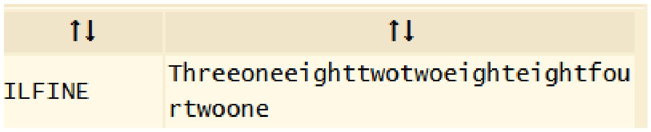
Translated it says "The true way of writing in cipher -- wikimedia, end of page 20."

Historical research shows that this title belongs to a book by Giovan Battista Bellaso, but the cipher famously got misattributed to Blaise de Vigenere and is known as the Vigenere Cipher.

Ahh, Italiano

Lets decipher the text provided on the image using https://www.dcode.fr/vigenere-cipher

Using no key and automatic decryption I receive:



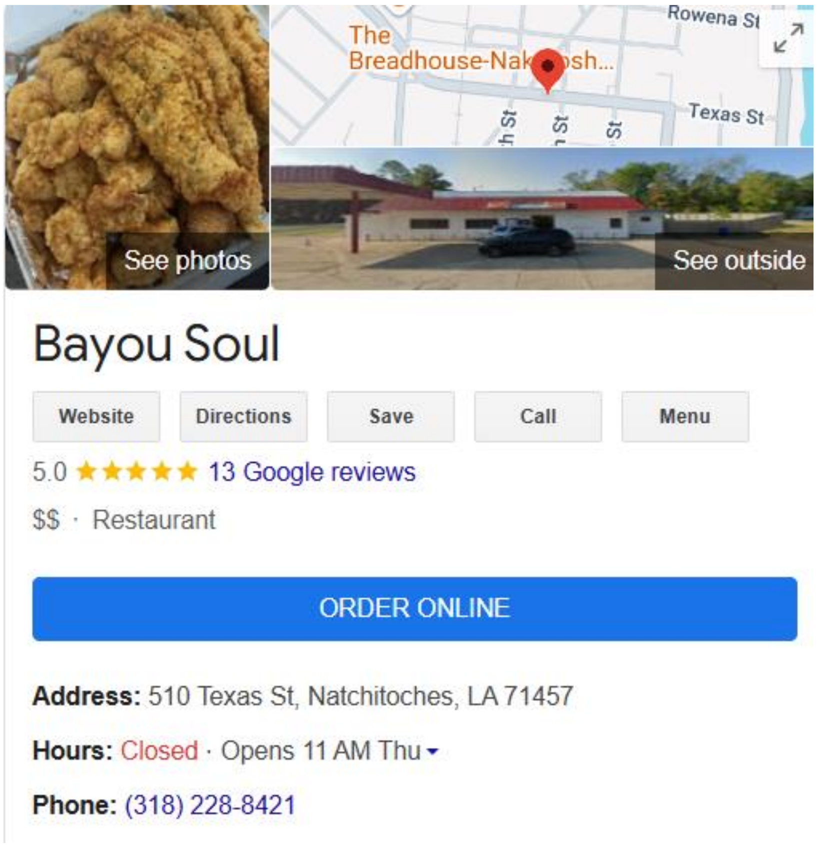
↑↓	↑↓
ILFINE	Threeoneeighttwotwoeighteightfourtwoone

What's that look like to you? Lets write it out

ThreeOneEightTwoTwoEightEightFourOneTwo

3318228412 BINGO, a phone number

I threw this number into google and one single restaurant popped up



Bayou Soul

5.0 ★★★★★ 13 Google reviews

$$ · Restaurant

Address: 510 Texas St, Natchitoches, LA 71457

Hours: Closed · Opens 11 AM Thu

Phone: (318) 228-8421

Ok now what? This part took me some time to recognize and discover

What is the next direction you would head?

Surprisingly , the title of the image is the next clue "a_paper_football_player"

TLDR: Rabbit hole
First, I rabbit holed thinking paper football had more meaning, possibly paper as educational or student and football as college football. The title of the challenge is I'm Hungry, I stumbled upon the bayou classic college football game held during what time? Thanksgiving, but how would this further lead to a flag…

Let's look at the title of the image in a simpler way.

What's paper football? A game

How do you play this game? You flick a piece of paper into a goal

You "flick", based on the previous clues I've found they steer you in directions like

Italian book -> Vigenère cipher

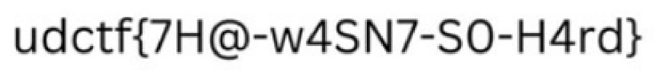
Phone number -> restaurant

Paper football -> flick

I google "flick" nothing too fruitful

I google “flick bayou soul”

Wow, the website flickr has an account Bayou Soul with an image of what?



udctf{7H@-w4SN7-S0-H4rd}

(*1) Foundational checks

1. Just to be sure what file you are facing with, check its type with `file filename`.
2. View all strings in the file with `strings -n 7 -t x filename.png`. We use `-n 7` for strings of length 7+, and `-t x` to view- their position in the file.
3. Exiftool to check all metadata
4. Binwalk to check image for hidden embedded files. My preferred syntax is `binwalk -Me filename.png`. -Me is used to recursively extract any files.
5. Found a password (or not) I use steghide to extract data `steghide extract -sf filename.png`.
6. Java -jar stegsolve.jar used to explore colour & bit planes (Full Red, Inverse, LSB etc). Im looking static at the top of any planes. This tool also allows checking of RBGA values

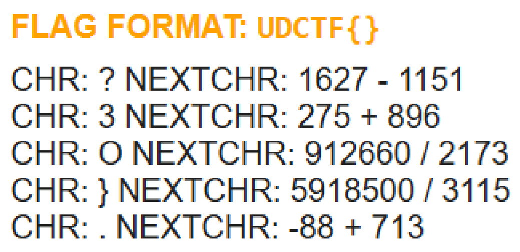
Challenge: lists of JSONs

Category: Web

Description: Web...ish...

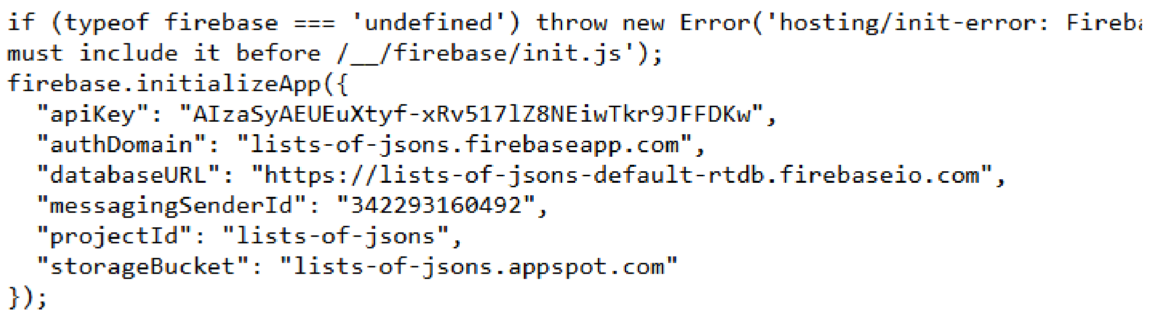
Attempted Solution:

Proceeded to given website which gave partial json



```
FLAG FORMAT: UDCTF{}
CHR: ? NEXTCHR: 1627 - 1151
CHR: 3 NEXTCHR: 275 + 896
CHR: O NEXTCHR: 912660 / 2173
CHR: } NEXTCHR: 5918500 / 3115
CHR: . NEXTCHR: -88 + 713
```

First, I inspected the page source code and after sifting through the files I found a javascript file with the following code:

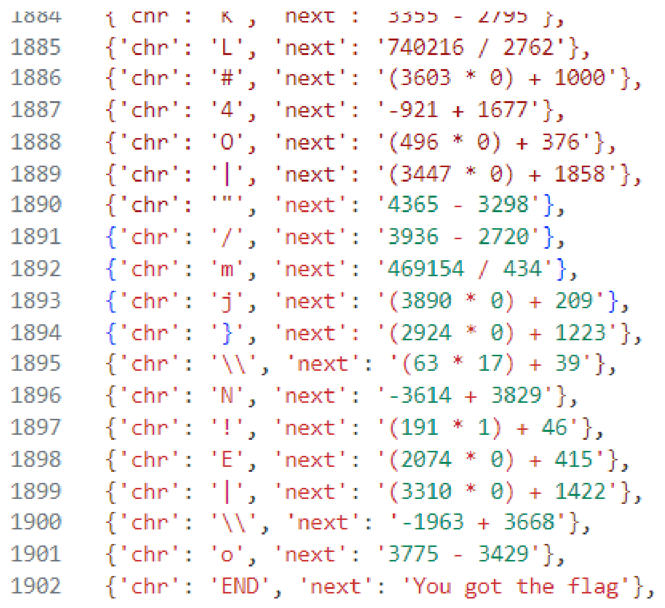


```
if (typeof firebase === 'undefined') throw new Error('hosting/init-error: Fireba
must include it before /__/firebase/init.js');
firebase.initializeApp({
  "apiKey": "AIzaSyAEUEuXtyf-xRv517lZ8NEiwTkr9JFFDKw",
  "authDomain": "lists-of-jsons.firebaseapp.com",
  "databaseURL": "https://lists-of-jsons-default-rtdb.firebaseio.com",
  "messagingSenderId": "342293160492",
  "projectId": "lists-of-jsons",
  "storageBucket": "lists-of-jsons.appspot.com"
});
```

Any word with "key" in it is likely useful, the apikey is a publicly shareable key that is used to interact with Firebase services. Using a curl to output the firestore database.

curl 'https://lists-of-jsons-default-rtdb.firebaseio.com/data.json?auth=AIzaSyAEUEuXtyf-xRv517lZ8NEiwTkr9JFFDKw'

There were over 1000 lines of json data, which I transferred to a Json file

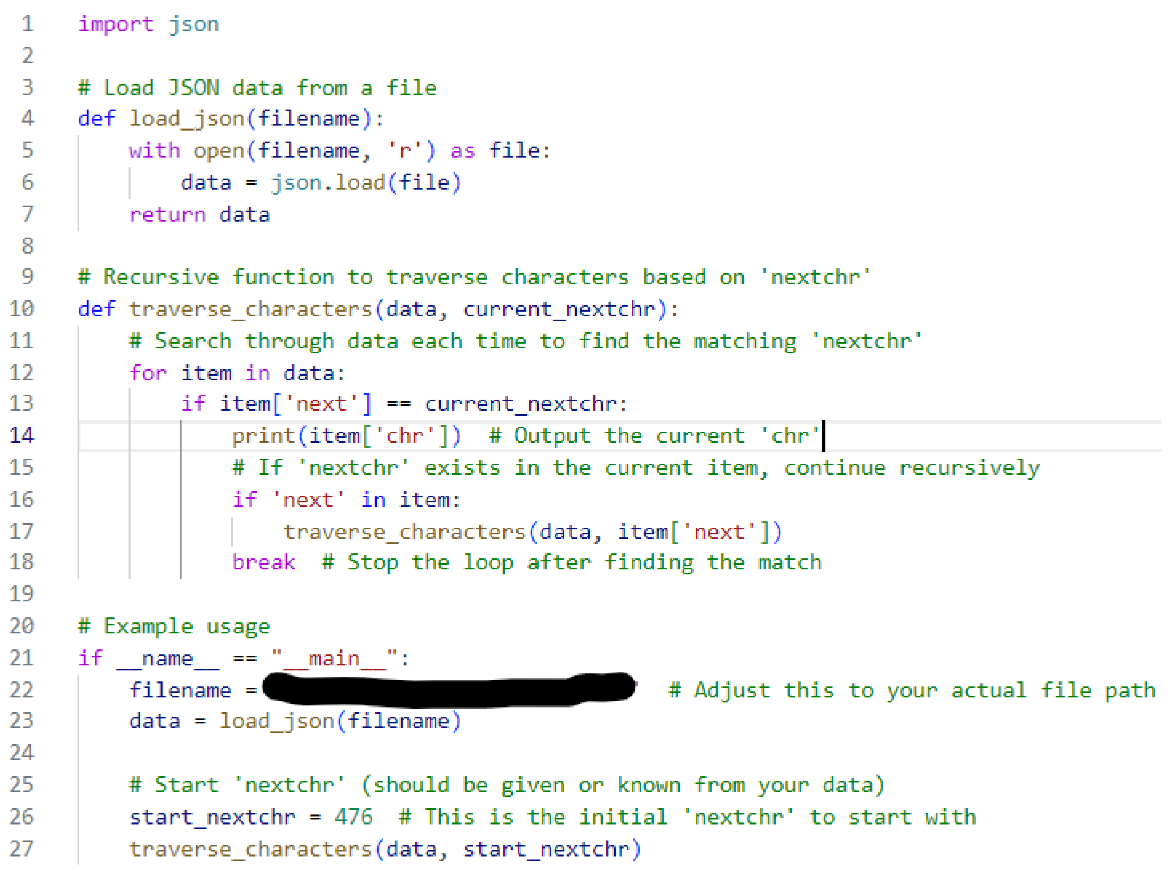


```
1884   {'chr': 'K', 'next': '3555 - 2795'},
1885   {'chr': 'L', 'next': '740216 / 2762'},
1886   {'chr': '#', 'next': '(3603 * 0) + 1000'},
1887   {'chr': '4', 'next': '-921 + 1677'},
1888   {'chr': 'O', 'next': '(496 * 0) + 376'},
1889   {'chr': '|', 'next': '(3447 * 0) + 1858'},
1890   {'chr': '"', 'next': '4365 - 3298'},
1891   {'chr': '/', 'next': '3936 - 2720'},
1892   {'chr': 'm', 'next': '469154 / 434'},
1893   {'chr': 'j', 'next': '(3890 * 0) + 209'},
1894   {'chr': '}', 'next': '(2924 * 0) + 1223'},
1895   {'chr': '\\', 'next': '(63 * 17) + 39'},
1896   {'chr': 'N', 'next': '-3614 + 3829'},
1897   {'chr': '!', 'next': '(191 * 1) + 46'},
1898   {'chr': 'E', 'next': '(2074 * 0) + 415'},
1899   {'chr': '|', 'next': '(3310 * 0) + 1422'},
1900   {'chr': '\\', 'next': '-1963 + 3668'},
1901   {'chr': 'o', 'next': '3775 - 3429'},
1902   {'chr': 'END', 'next': 'You got the flag'},
```

Next, we had to print the 'chr' and the next character to be printed was the character who's 'next' value holds the value of the current number of the calculation 'next'

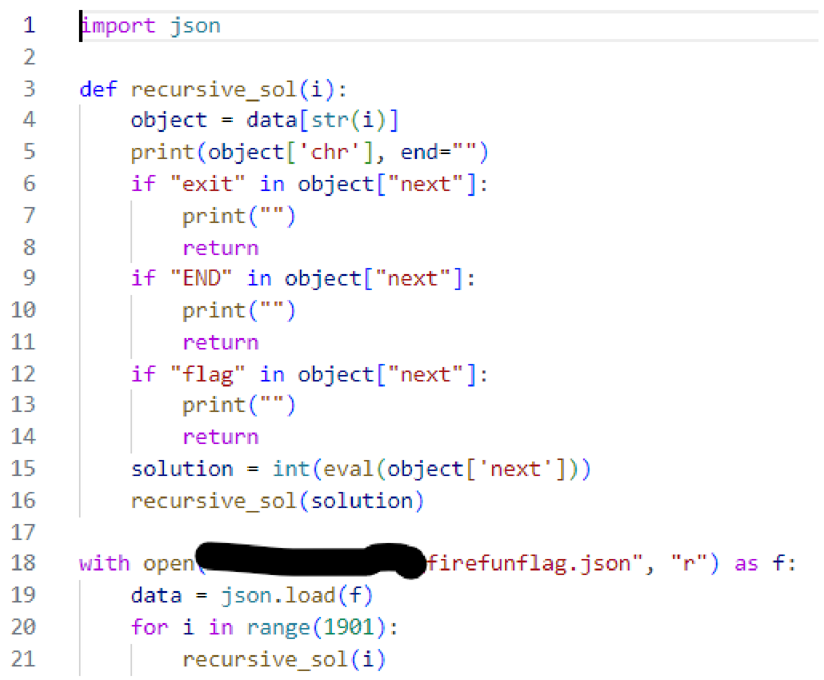
For example, line 1893 the character j would be printed and the 'next' character to be printed needs the 'next' value of 39

I made a python script to traverse the characters based on do the math calculations and then made a script to recursively traverse characters based on 'next'



```python
import json

# Load JSON data from a file
def load_json(filename):
    with open(filename, 'r') as file:
        data = json.load(file)
    return data

# Recursive function to traverse characters based on 'nextchr'
def traverse_characters(data, current_nextchr):
    # Search through data each time to find the matching 'nextchr'
    for item in data:
        if item['next'] == current_nextchr:
            print(item['chr'])  # Output the current 'chr'
            # If 'nextchr' exists in the current item, continue recursively
            if 'next' in item:
                traverse_characters(data, item['next'])
            break  # Stop the loop after finding the match

# Example usage
if __name__ == "__main__":
    filename = [redacted]  # Adjust this to your actual file path
    data = load_json(filename)

    # Start 'nextchr' (should be given or known from your data)
    start_nextchr = 476  # This is the initial 'nextchr' to start with
    traverse_characters(data, start_nextchr)
```

I was unable to complete this challenge, asked a team mate for assistance at which they had solved the problem using the below script.



```python
import json

def recursive_sol(i):
    object = data[str(i)]
    print(object['chr'], end="")
    if "exit" in object["next"]:
        print("")
        return
    if "END" in object["next"]:
        print("")
        return
    if "flag" in object["next"]:
        print("")
        return
    solution = int(eval(object['next']))
    recursive_sol(solution)

with open([redacted]firefunflag.json", "r") as f:
    data = json.load(f)
    for i in range(1901):
        recursive_sol(i)
```