Capture the Flag
ARA 4.0 2023

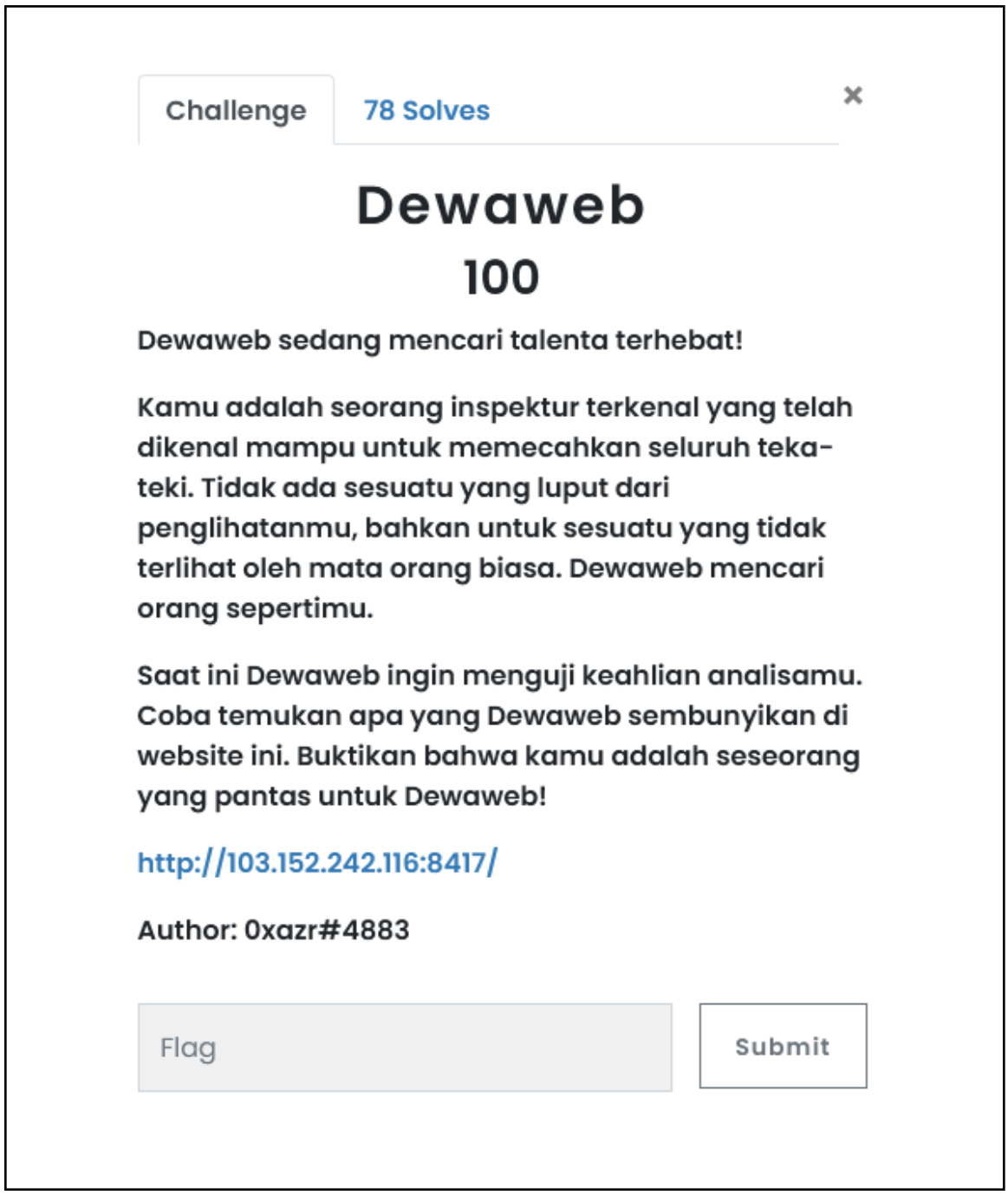
TIM
sudah dapet orang


mitm
mxlyk
Plasma

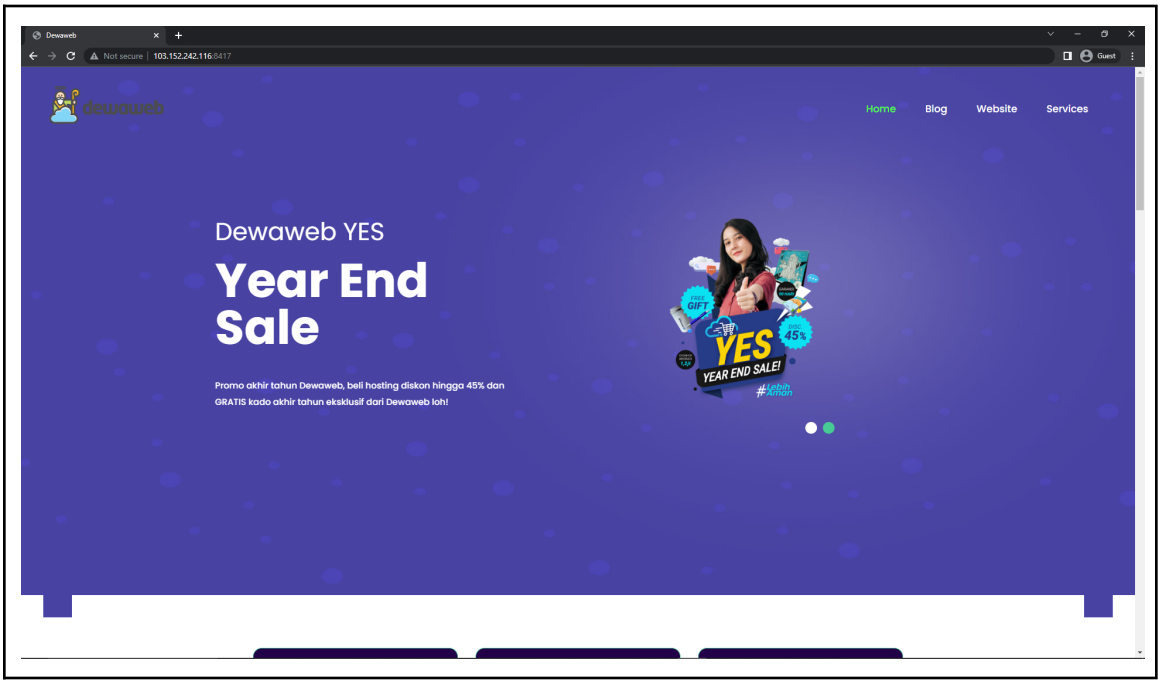
DAFTAR ISI

WEB EXPLOITATION

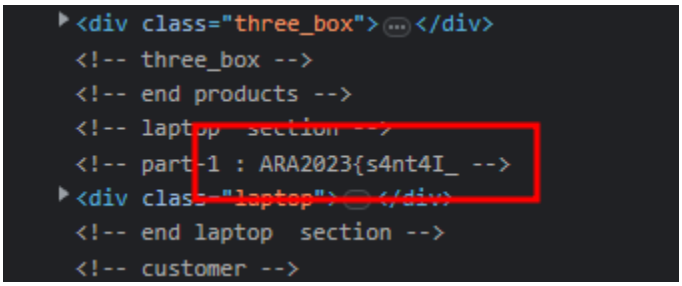
- Dewaweb



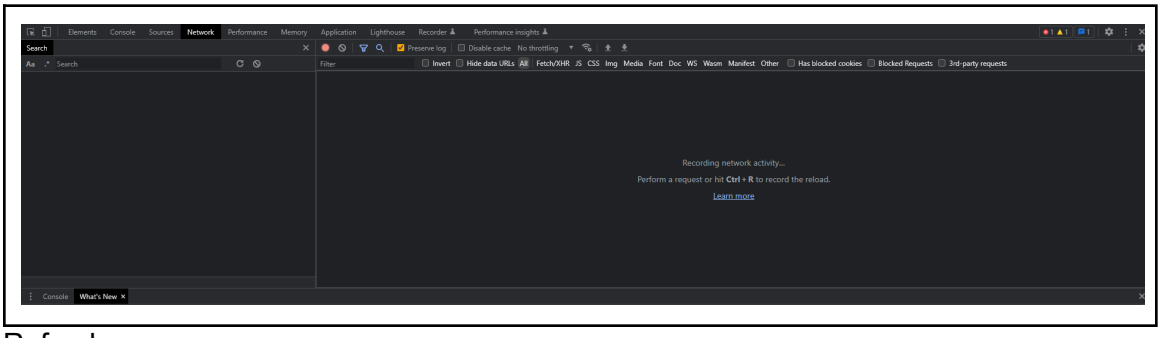
Pertama mari kita buka web nya terlebih dahulu



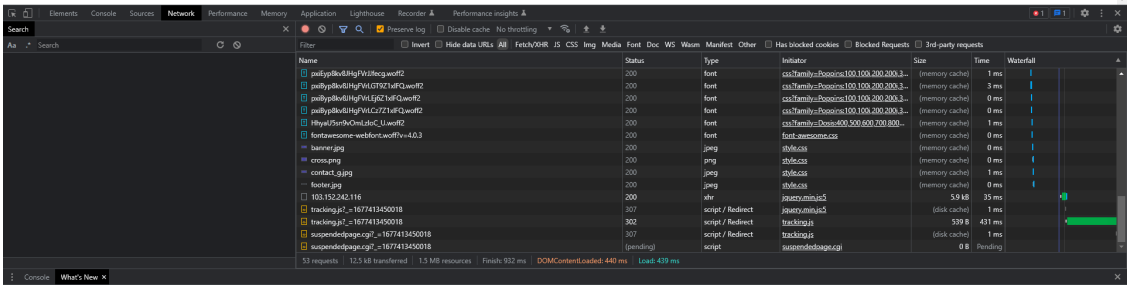
Tidak terlihat ada yang menarik, dan keliahatan nya hanya static page, setelah membuka inspect element, berikut yang dapat kita lihat



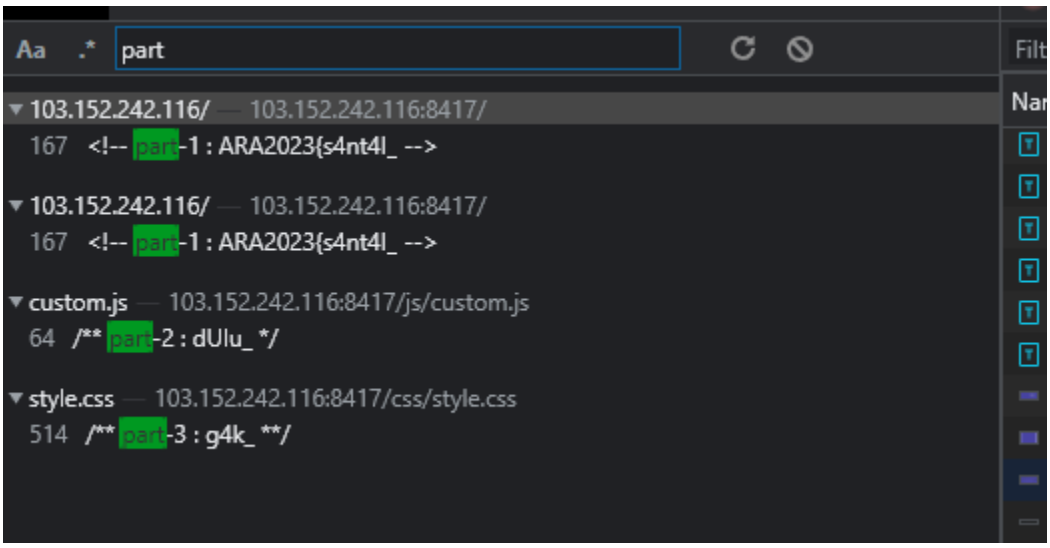
Dari sini karena chrome memiliki fungsi untuk mencari seluruh request, kita tinggal buka network tab



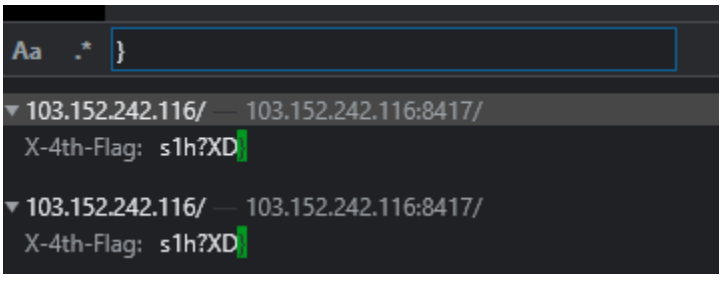
Refresh page nya



Lalu cari part di bagian kiri



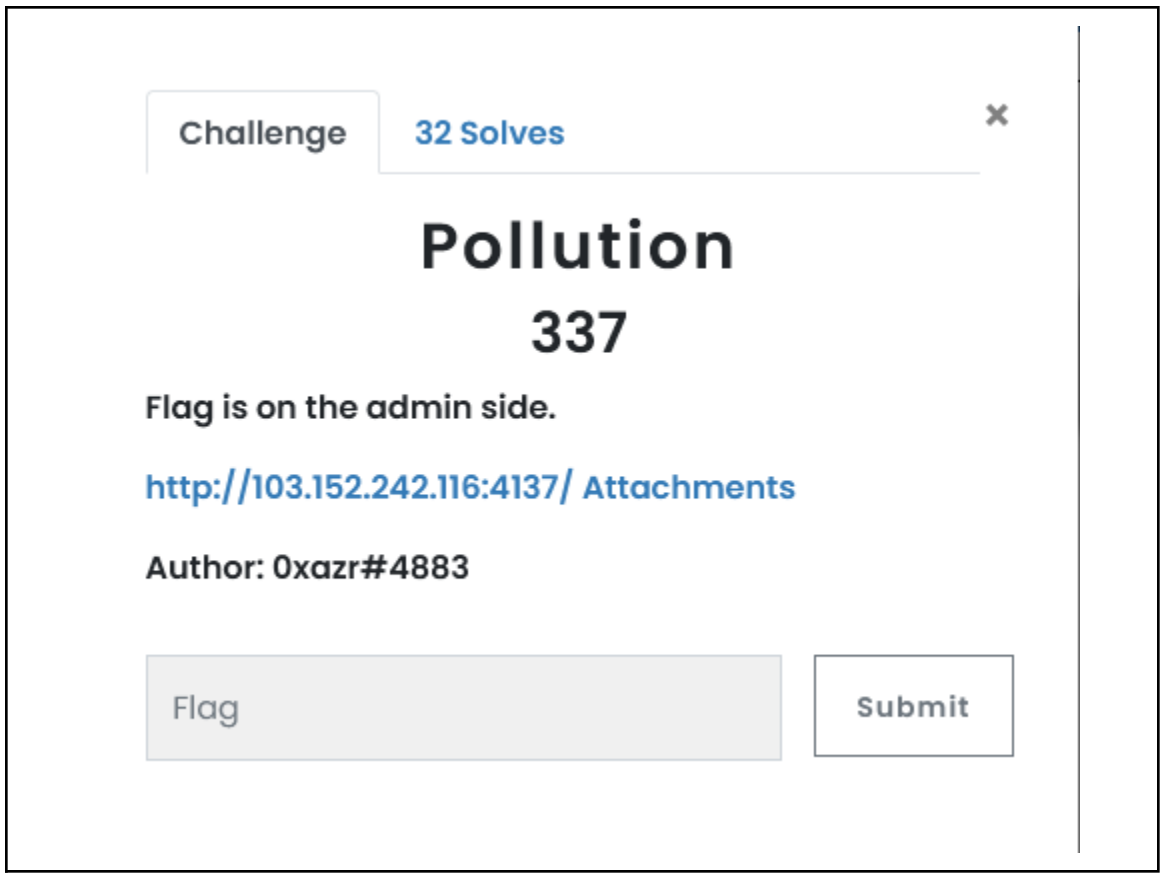
Namun ternyata kurang 1 part lagi, disini kita cari saja tutup kurung nya **}** seperti berikut



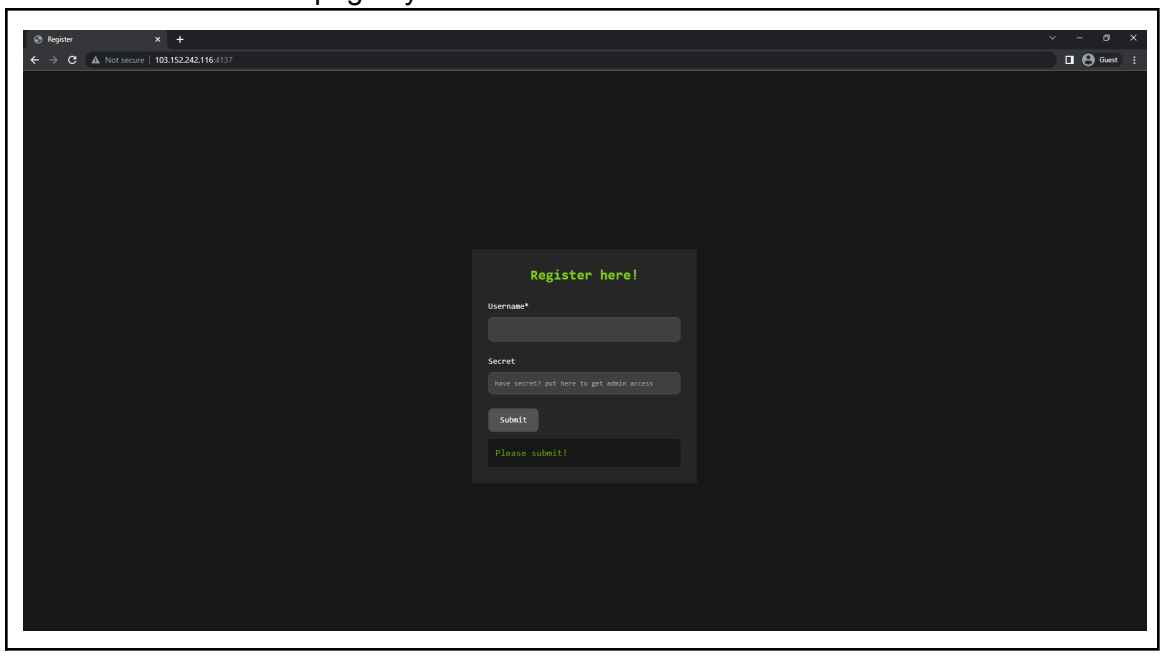
Dan terlihat flag terakhir, yang ternyata terdapat di header nya

FLAG:
ARA2023{s4nt4l_dUlu_g4k_s1h?XD}

- Pollution



Pertama mari kita buka page nya



Kelihatan nya kita bisa mensubmit sesuatu, sebelum mencoba mari kita buka source nya terlebih dahulu

```
const express = require('express');
const bodyParser = require('body-parser');
const secret = require('./secret');
const path = require('path');
const app = express();

app.use(bodyParser.text());
app.use('/static', express.static(path.resolve('static')));

const baseUser = {
    "picture": "profile.jpg"
}

app.get('/', (req, res) => {
    res.sendFile('/views/index.html', { root: __dirname });
})

app.post('/register', (req, res) => {
    let user = JSON.parse(req.body);

    // Haha, even you can set your role to Admin, but you don't have the secret!
    if (user.role == "Admin") {
        console.log(user.secret);
        if(user.secret !== secret.value) return res.send({
            "message": "Wrong secret! no Admin!"
        });
        return res.send({
            "message": "Here is your flag!",
            secret: secret.value
        });
    }

    let newUser = Object.assign(baseUser, user);
    if(newUser.role === "Admin") {
        return res.send({
            "message": "Here is your flag!",
            secret: secret.value
        });
    }

    else return res.send({
        "message": "No Admin? no flag!"
    });
})

const port = 1337;
app.listen(port, () => {
    console.log(`Listening at port:${port}`);
})
```

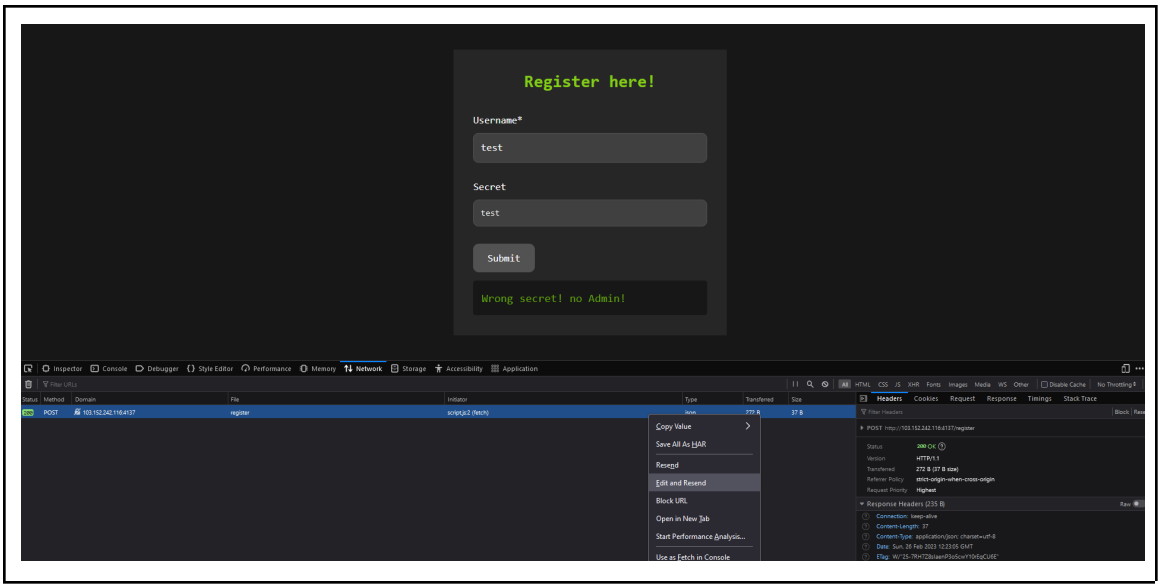
Berikut merupakan source code dari web tersebut, bila kita perhatikan di bagian atas, kita bisa menggunakan role Admin, namun kita butuh tahu secret nya, yang merupakan flag

Namun di bagian kedua seperti nya role nya di check lagi, namun kali ini tidak di check apakah si user memiliki secrete atau tidak

Dan sebelum check tersebut, terdapat object assignment terhadap newUser.role

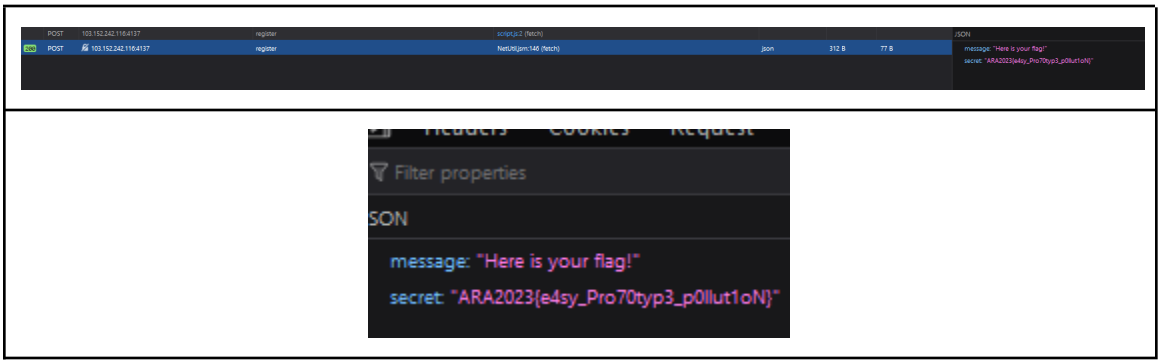
Dari beberapa hal tersebut dapat disimpulkan, kemungkinan besar serangan yang harus dilakukan adalah prototype pollution

Disini lagi lagi karena mager, kita bisa mengirimkan request nya melalui mozila



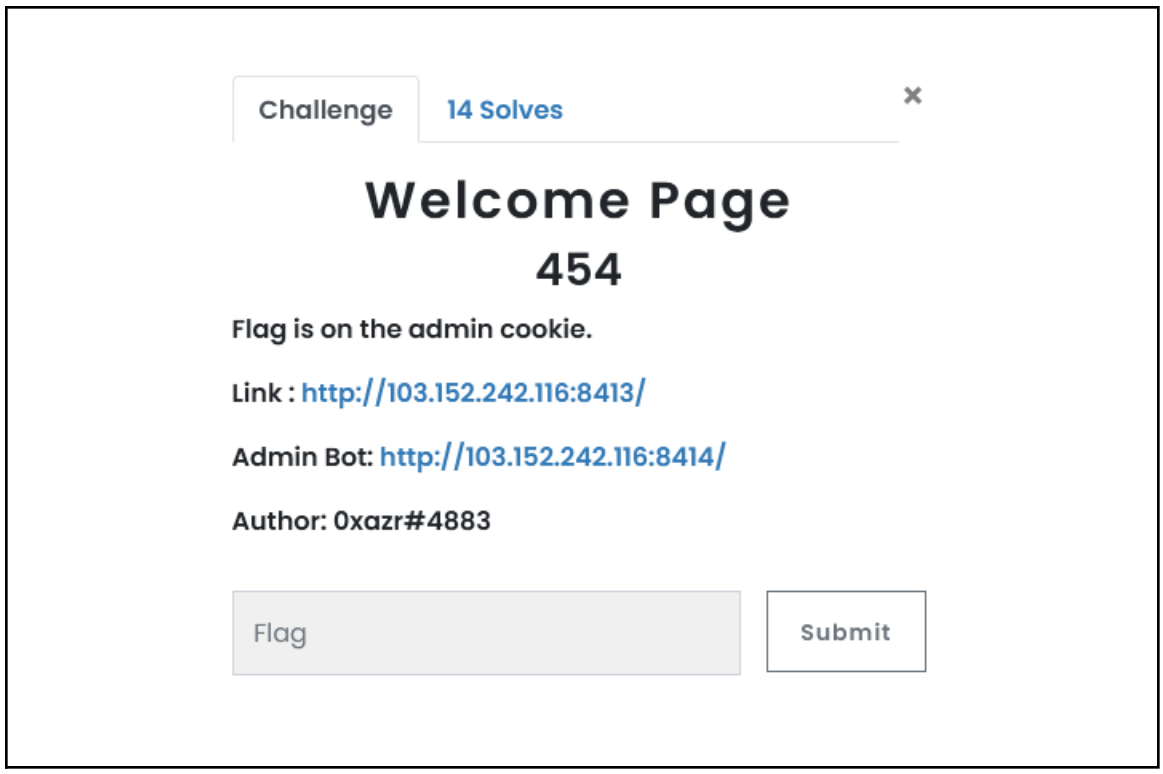
Lalu sekarang kita tinggal buat request nya sesuai keinginan kita

Tinggal kita kirim dan…

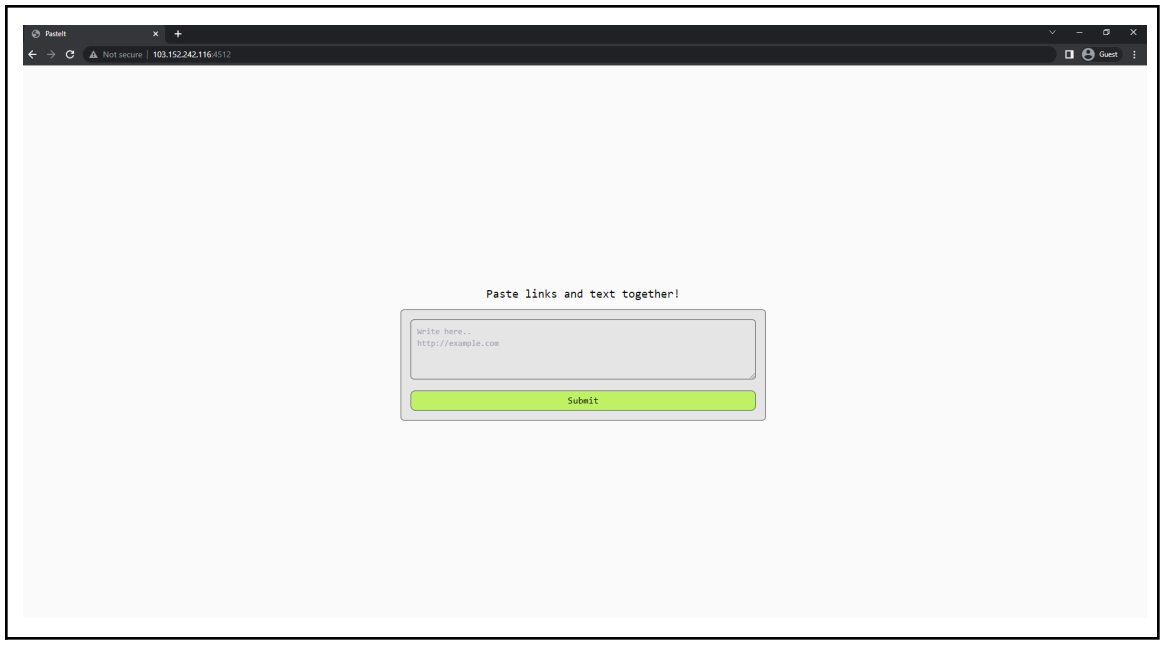


FLAG:
ARA2023{e4sy_Pro70typ3_p0llut1oN}

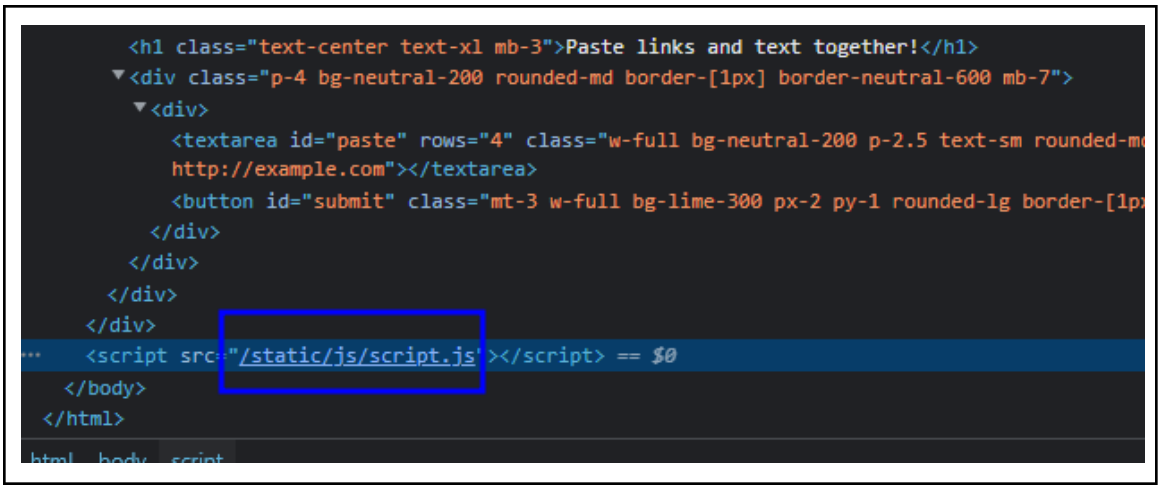
- Paste it



Tentu nya pertama kita buka dulu web nya



Kelihatan nya web kita dapat mengirim kan sesuatu, mari kita buka inspect element nya



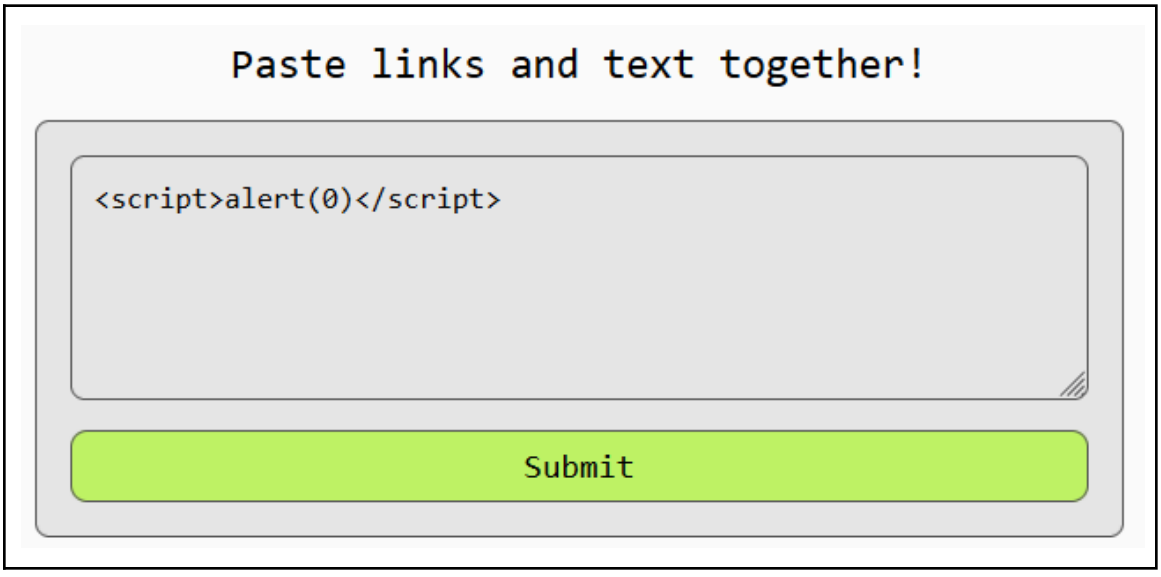
Ternyata hanya ada satu file js, mari kita buka untuk menganalisa lebih lanjut

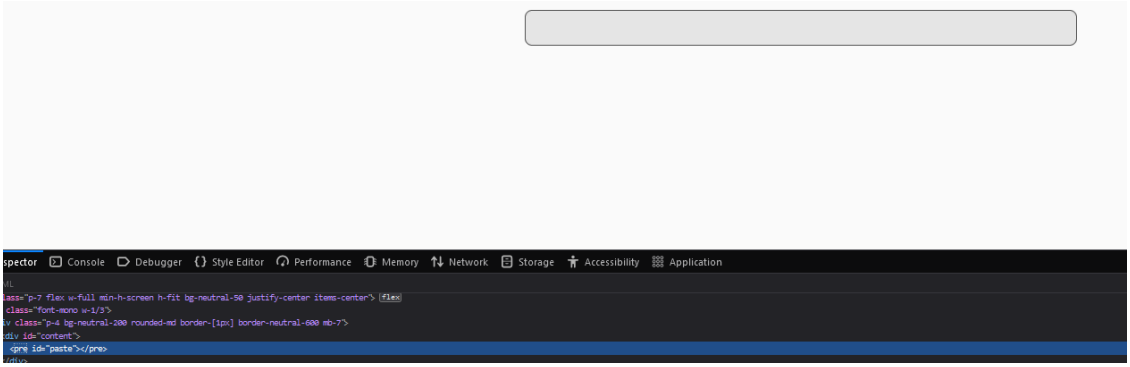
```
function addPaste() {
    const paste = document.getElementById("paste").value;
    console.log("test")
    fetch("/", {
        method: "POST",
        headers: {
            "Content-Type": "application/json"
        },
        body: JSON.stringify({
            paste: paste
        })
    })
    .then((response) => response.json())
    .then((data) => {
        return window.location.href = `/${data.id}`;
    });
}

const submit = document.getElementById("submit");
if(submit) {
    submit.addEventListener("click", addPaste);
}
```

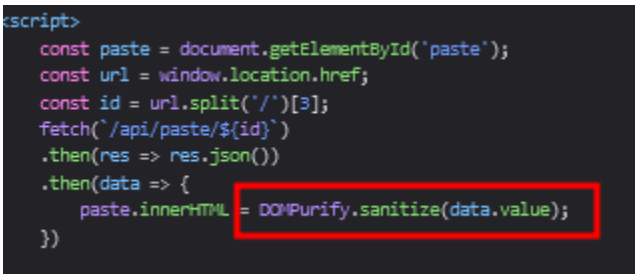
Kelihatan nya data yang kita isi akan di kirim kan ke server nya lagi, mari kita coba test web nya

Seperti nya terdapat xss pada page tersebut, mari kita kirim kan payload xss





Kelihatan nya terdapat semacam filter pada web tersebut, dan setelah melihat lihat pada source web tersebut berikut yang saya temukan



Dari sini kita tahu bahwa web tersebut menggunakan DOMPurify untuk memfilter user input

Setelah mencari cari payload untuk membypass, berikut yang saya temukan

DOMPurify bypass

So let's get back to the payload that bypasses DOMPurify:

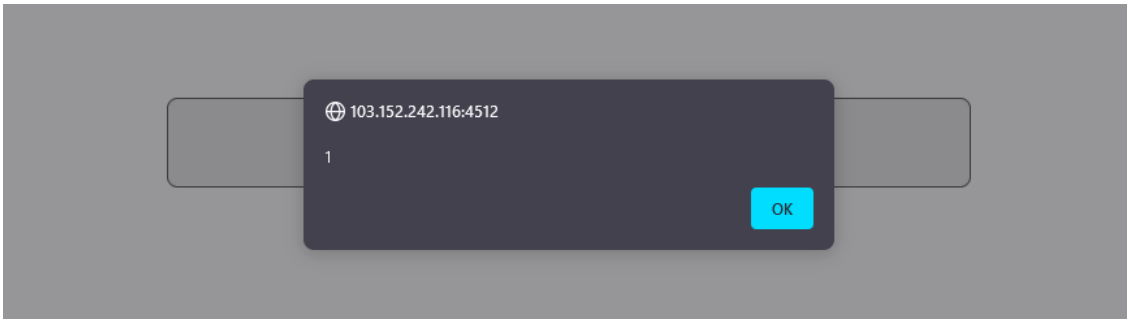
```
<form><math><mtext></form><form><mglyph><style></math><img src onerror=alert(1)>
```

The payload makes use of the mis-nested `html form` elements, and also contains `mglyph` element. It produces the following DOM tree:

```
└<html form>
    └<math math>
        └<math mtext>
            └<html form>
                └<html mglyph>
                    └<html style>
                        └#text: "</math><img src onerror=alert(1)>"
```

This DOM tree is harmless. All elements are in the allow-list of DOMPurify. Note that `mglyph` is in HTML namespace. And the snippet that looks like XSS payload is just a text within `html style`. Because there's a nested `html form`, we can be pretty sure that this DOM tree is going to be mutated on reparsing.

Dan bila kita masukan kode tersebut kedalam web nya berikut hasil nya



Dapat di simpulkan bahwa bypass nya berhasil

Namun tidak cukup di sini, kalau kita ingat sempat terdapat cuplikan kode experiment, yaitu berikut

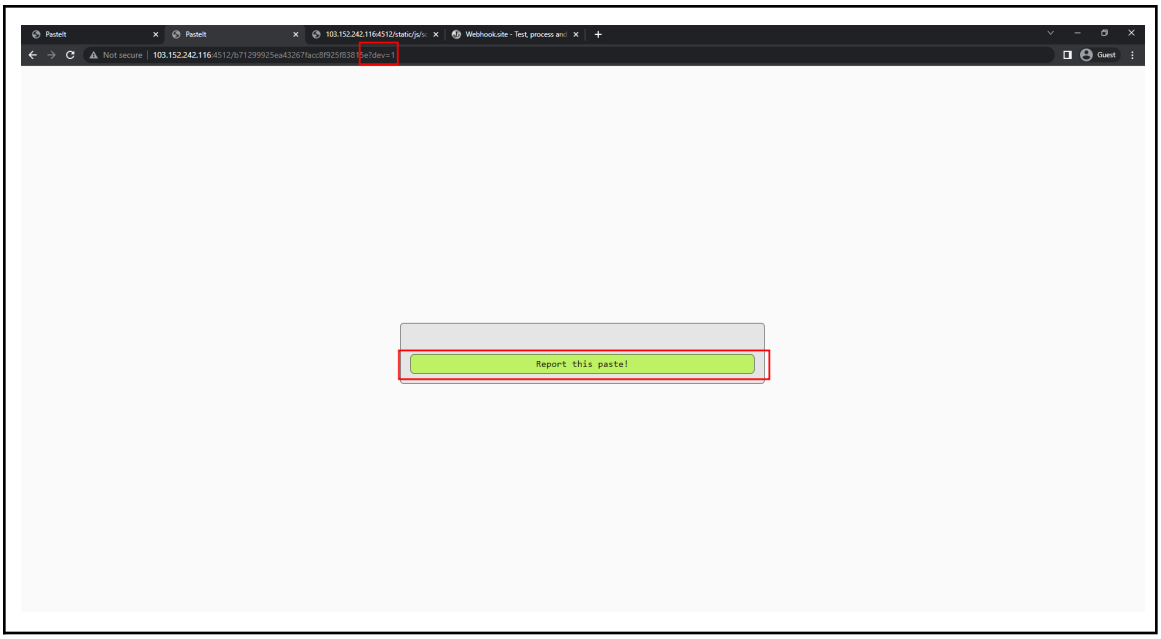
```
// Experimental feature
if(Arg.parse(location.search).dev) {
    console.log("You are in dev mode. Now you can report your paste with Admin.");

    const content = document.getElementById("content");
    const reportToAdmin = document.createElement("button");
    reportToAdmin.id = "reportToAdmin";
    reportToAdmin.setAttribute("class", "mt-3 w-full bg-lime-300 px-2 py-1 rounded-lg border-[1px] border-neutral-600 focus:bg-lime-500");
    reportToAdmin.innerHTML = "Report this paste!";
    content.appendChild(reportToAdmin);

    const reportToAdminButton = document.getElementById("reportToAdmin");
    reportToAdminButton.addEventListener("click", () => {
        fetch(`/api/report/`, {
            method: "POST",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify({
                id: id
            })
        })
        .then(res => res.json())
        .then(data => {
            if(data.success) {
                alert("Your paste has been reported to Admin. Thank you for your contribution.");
            } else {
                alert("Something went wrong. Please try again later.");
            }
        })
    })
}
```

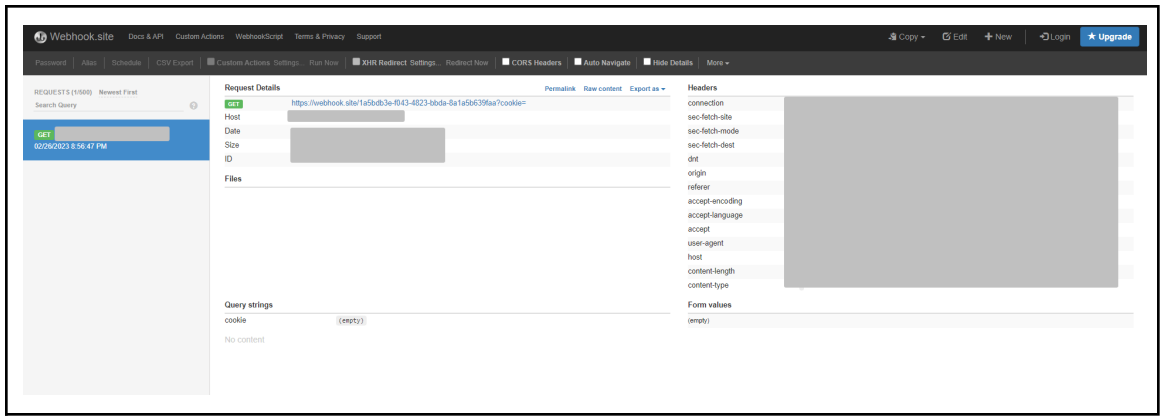
Dari sini dapat disimpulkan nanti nya page tersebut dapat di report atau kirimkan ke admin yang memiliki cookie

Dan ternyata untuk membuka fungsionalitas tersebut kita bisa tinggal menambah parameter **?dev=1** pada link page nya seperti berikut

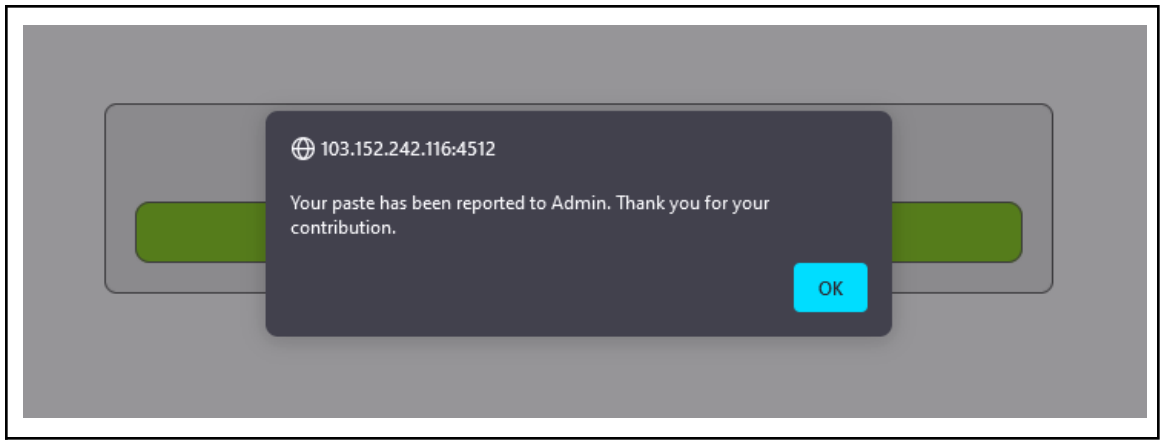


Sekarang kita tinggal buka webhook dan cookie stealer nya

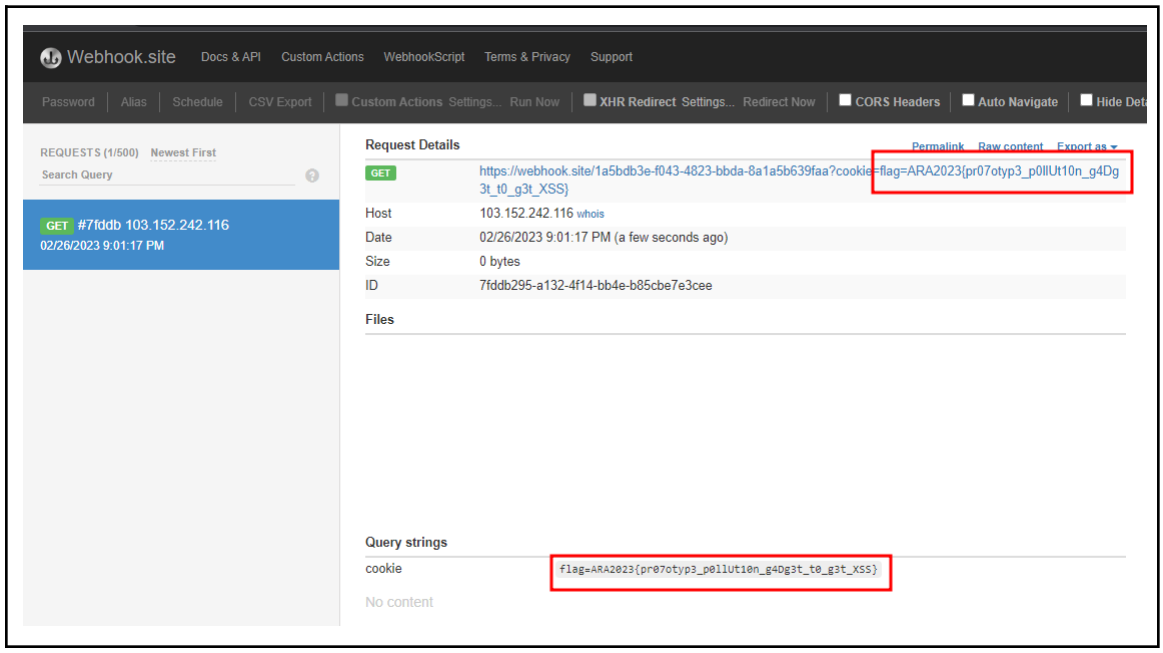
Tinggal kita kirim dan



Berhasil sekarang kita tinggal kirimkan page yang sama ke admin nya



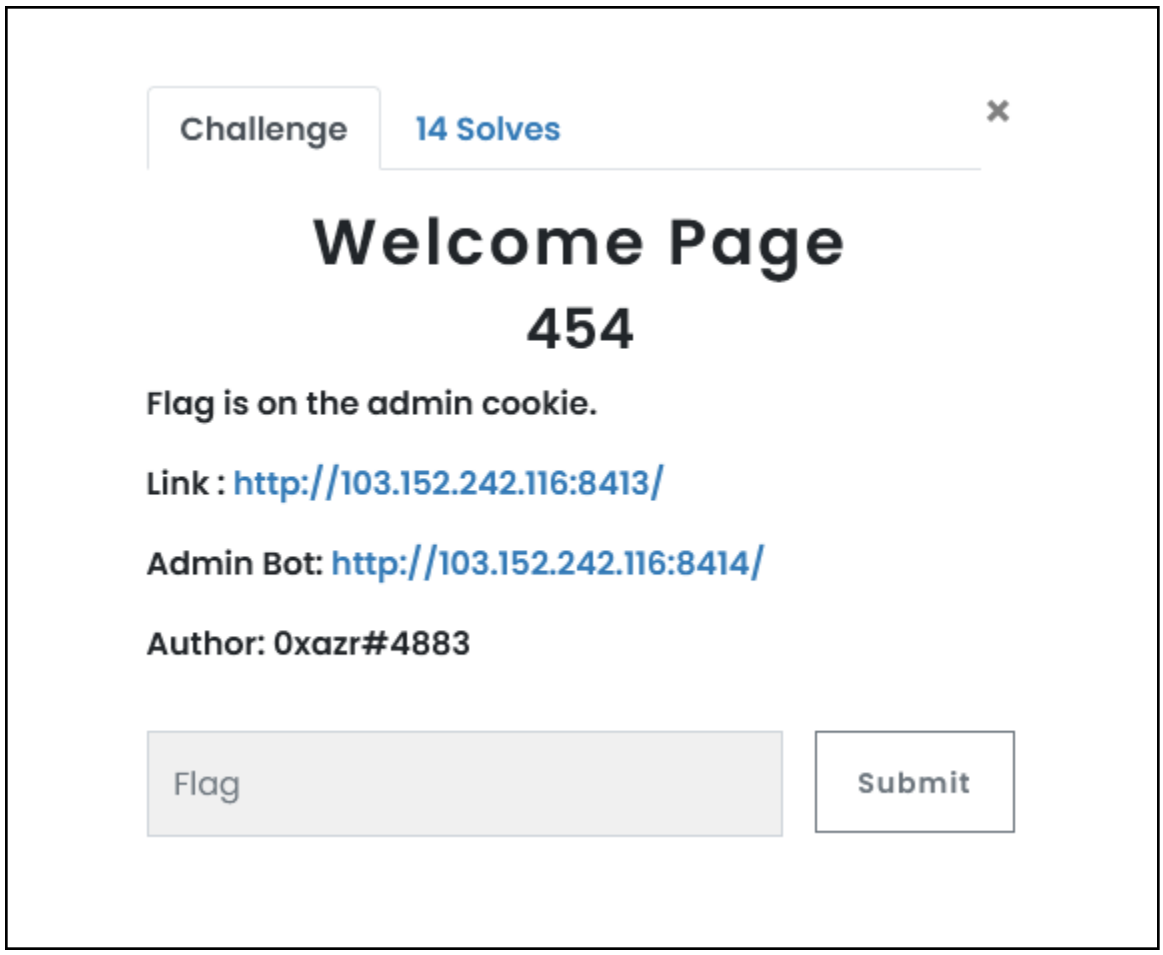
Dan berhasil, sekarang kita tinggal lihat di webhook, apakah cookie nya di dump



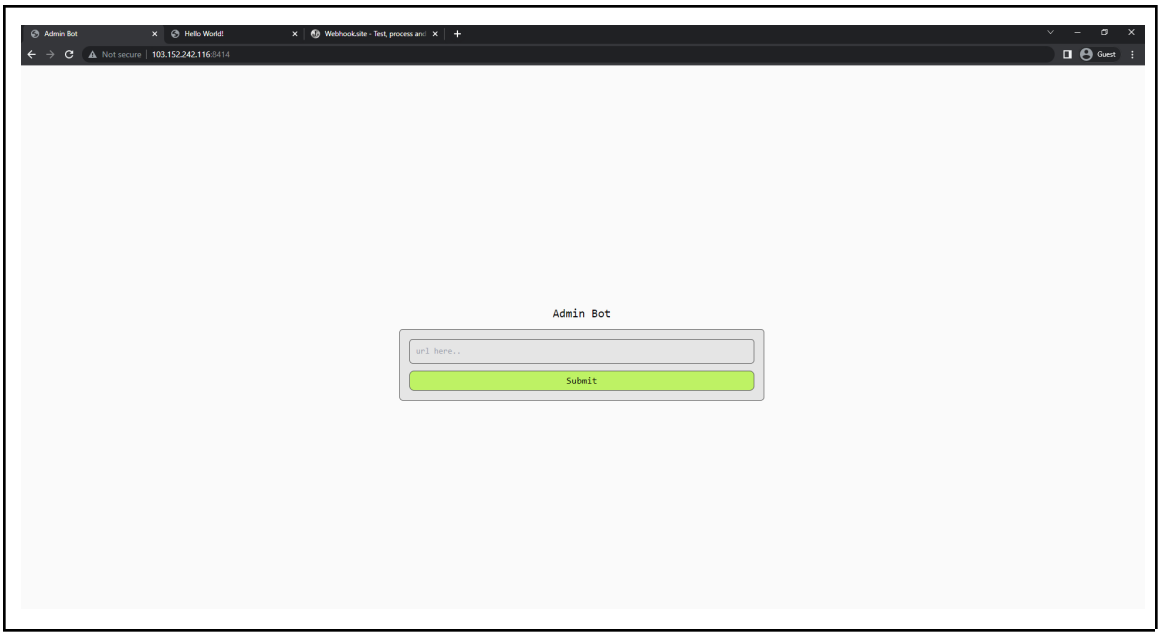
Dan seperti yang bisa kita lihat, berhasil

FLAG:
ARA2023{pr07otyp3_p0llUt10n_g4Dg3t_t0_g3t_XSS}

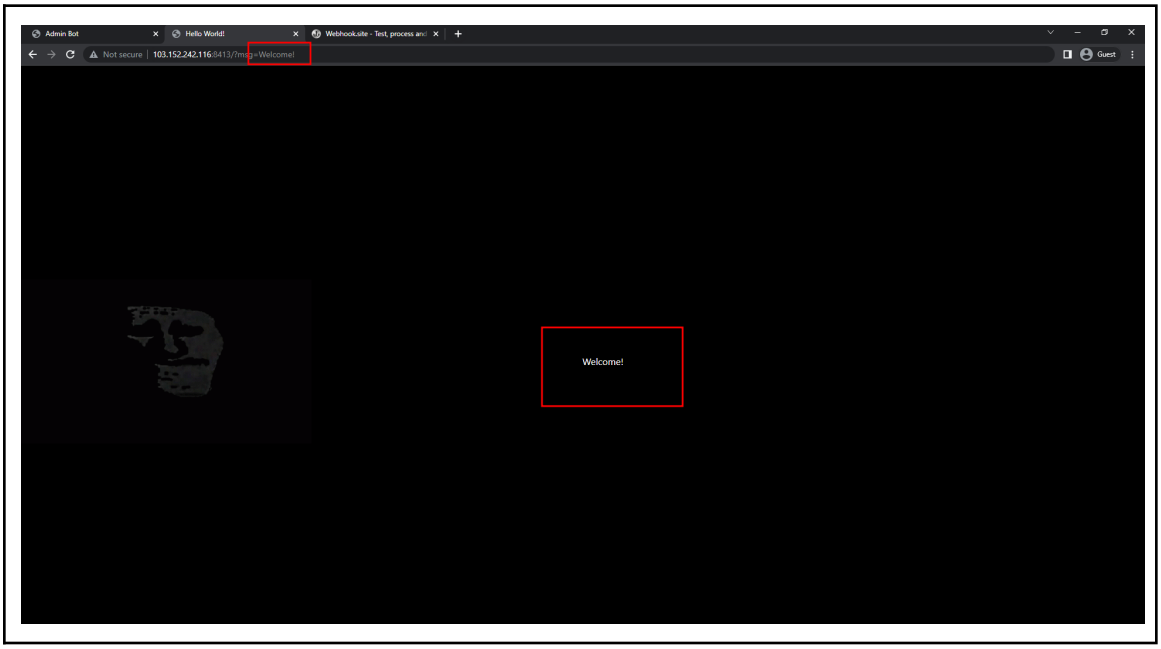
- Welcome Page



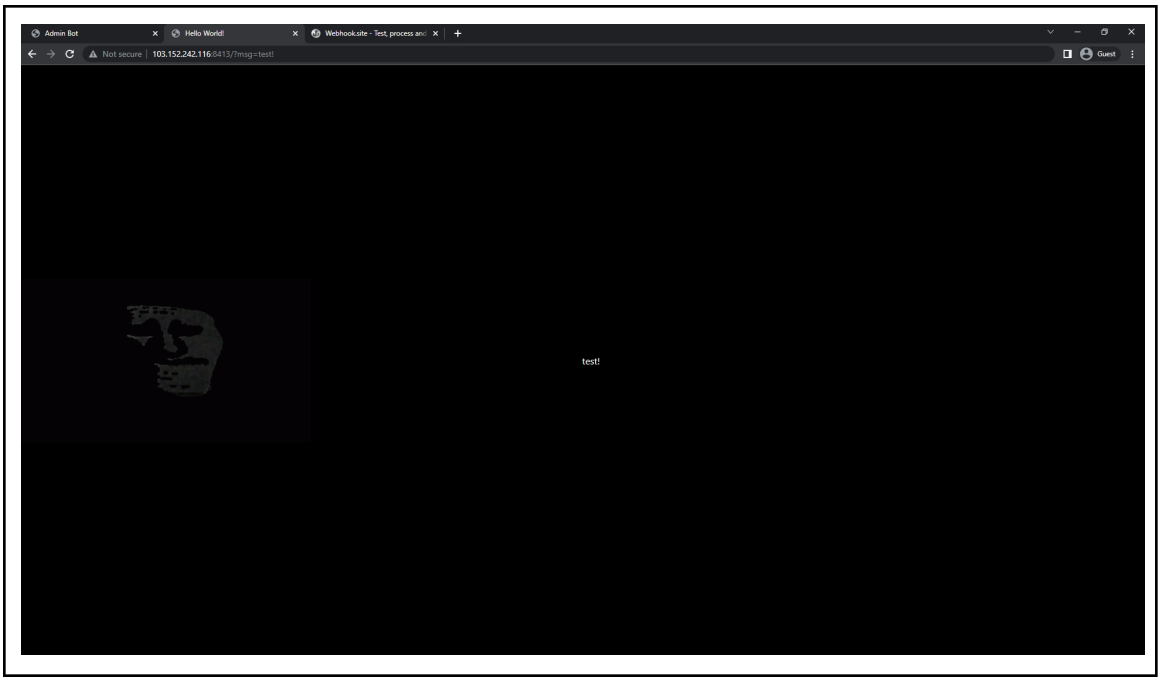
Pertama tentu nya buka dulu ke 2 web nya



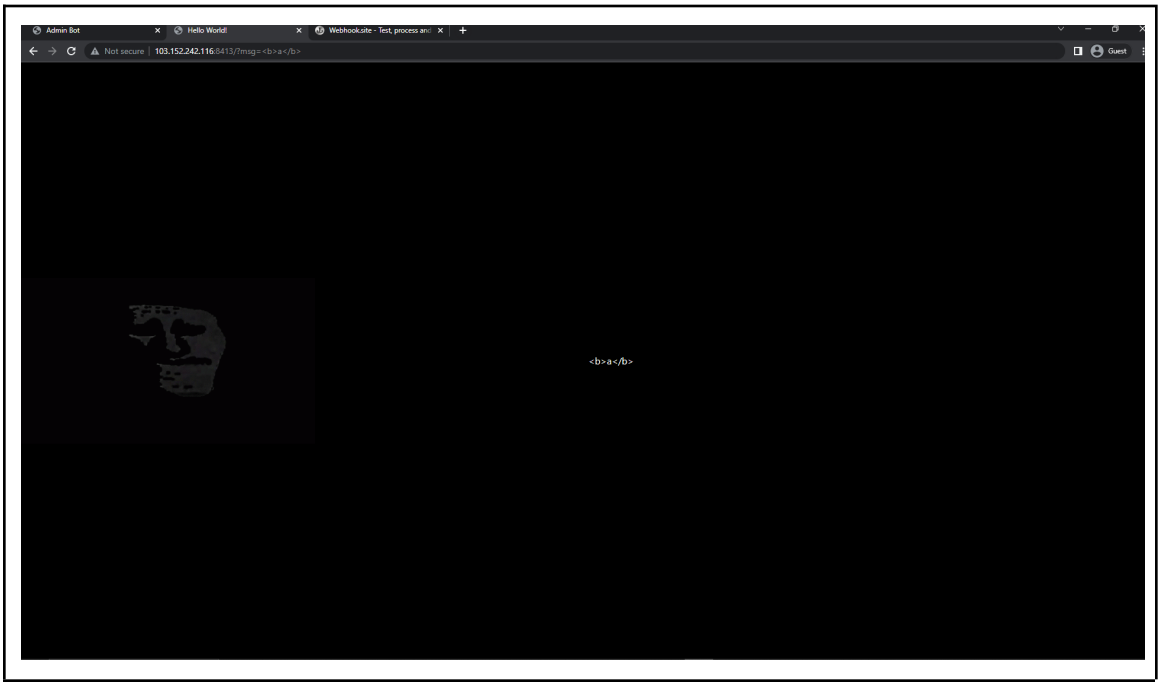
Seperti nya ini page untuk mengirimkan page yang sudah kita berikan exploit



Lalu di site ke dua, kelihatan message yang ada di atas sama dengan yang ada di tengah page

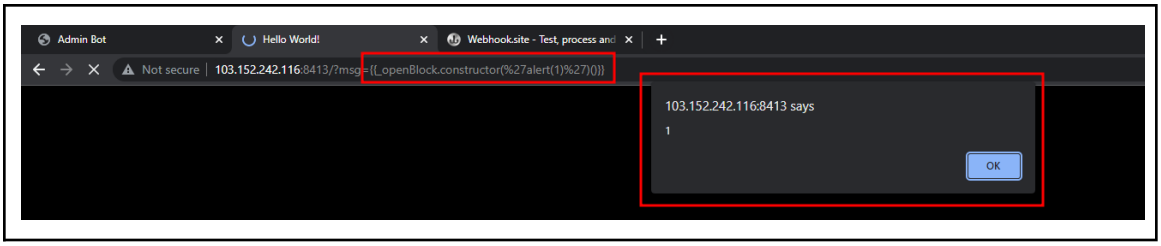


Dan kalau kita ubah ternyata berubah juga



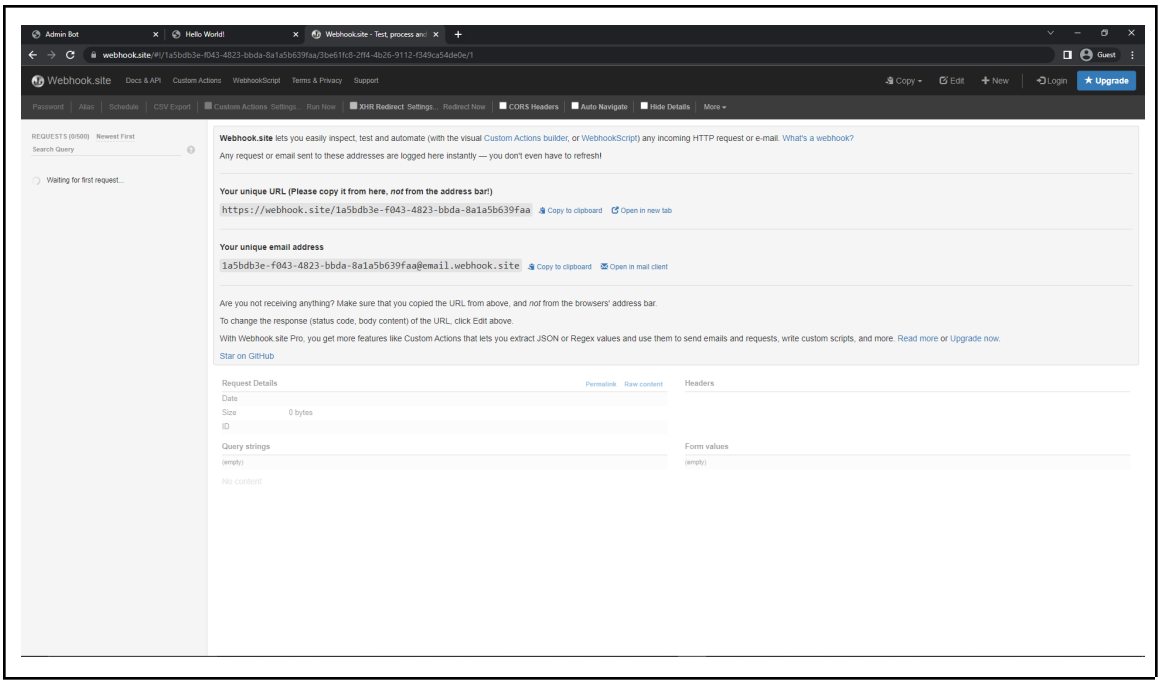
Namun kelihatan nya kalau kita buat tag, web nya mensanitasi input nya

Dan kalau kita buka inspect element, seperti nya web tersebut menggunakan vue

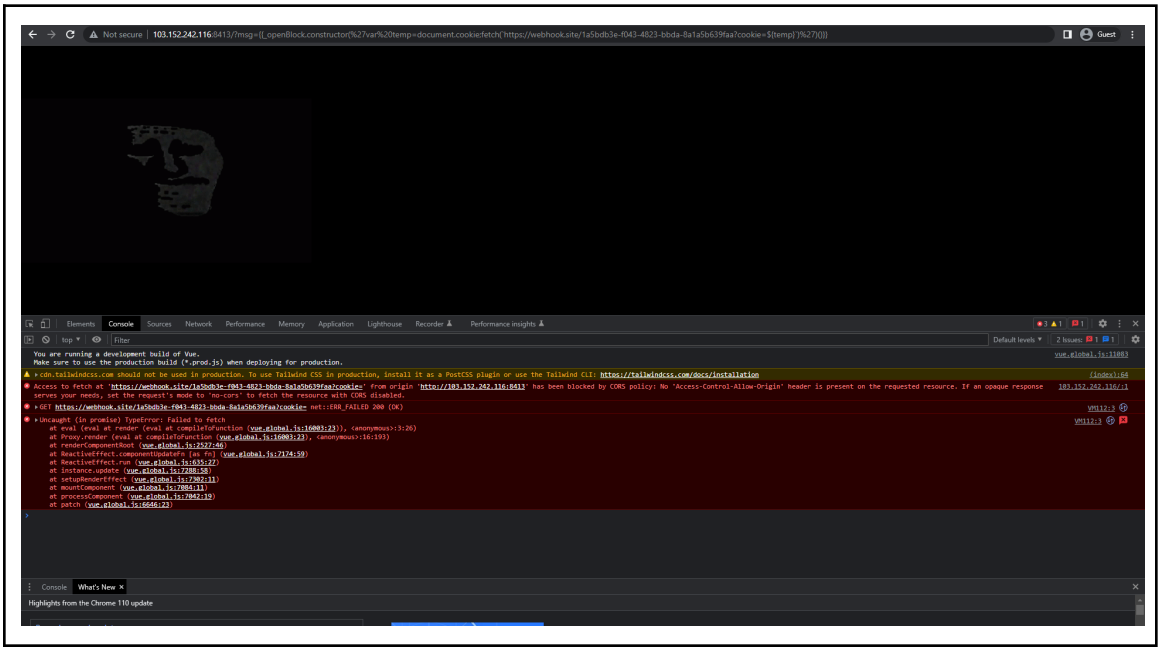


Dan kelihatannya kalau kita injeksi dengan xss vue, berhasil

Sekarang kita tinggal exploitasi, dengan membuka webhook, dan membuat exploit nya

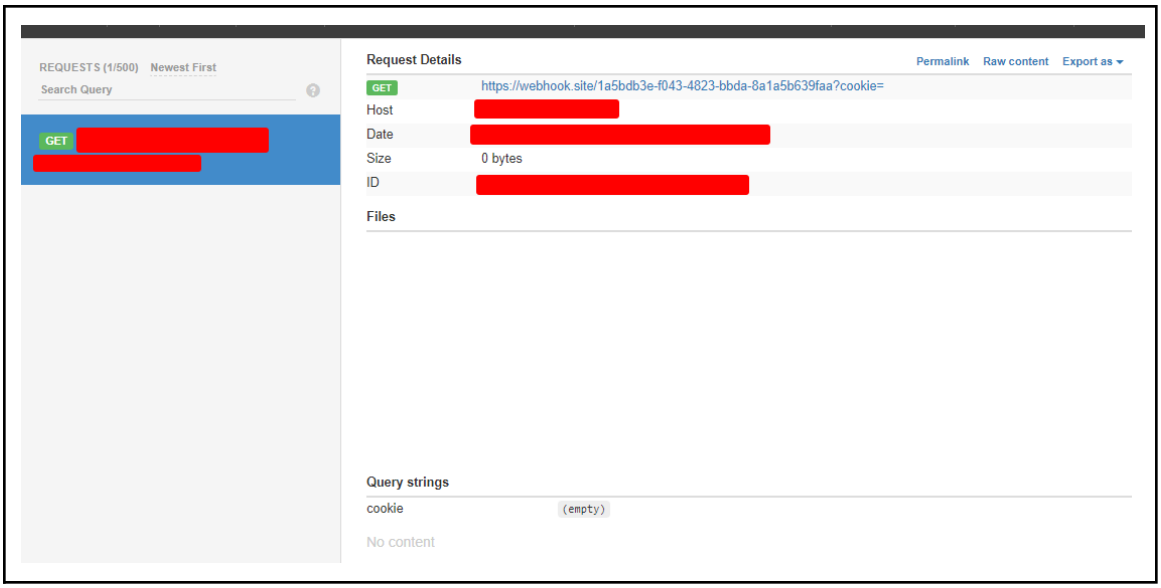


Sekarang kita tinggal buat payload nya

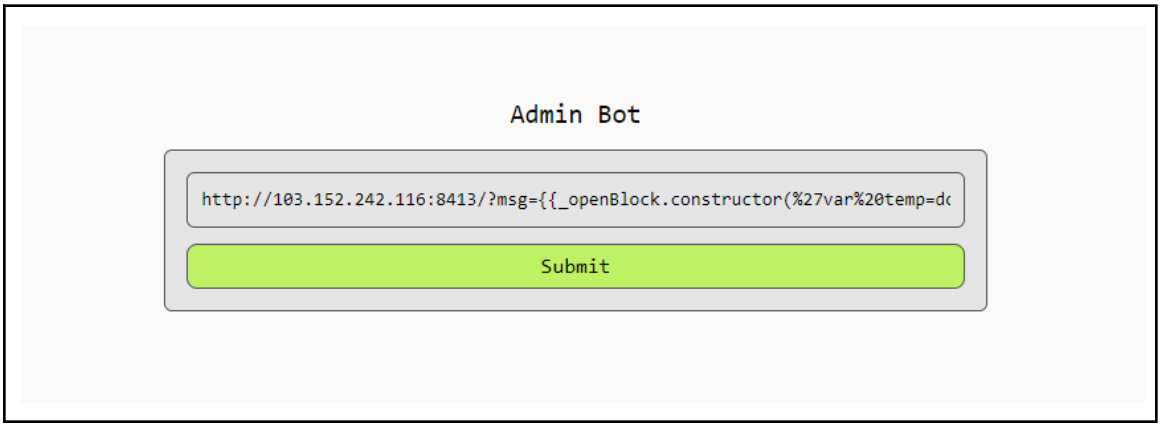


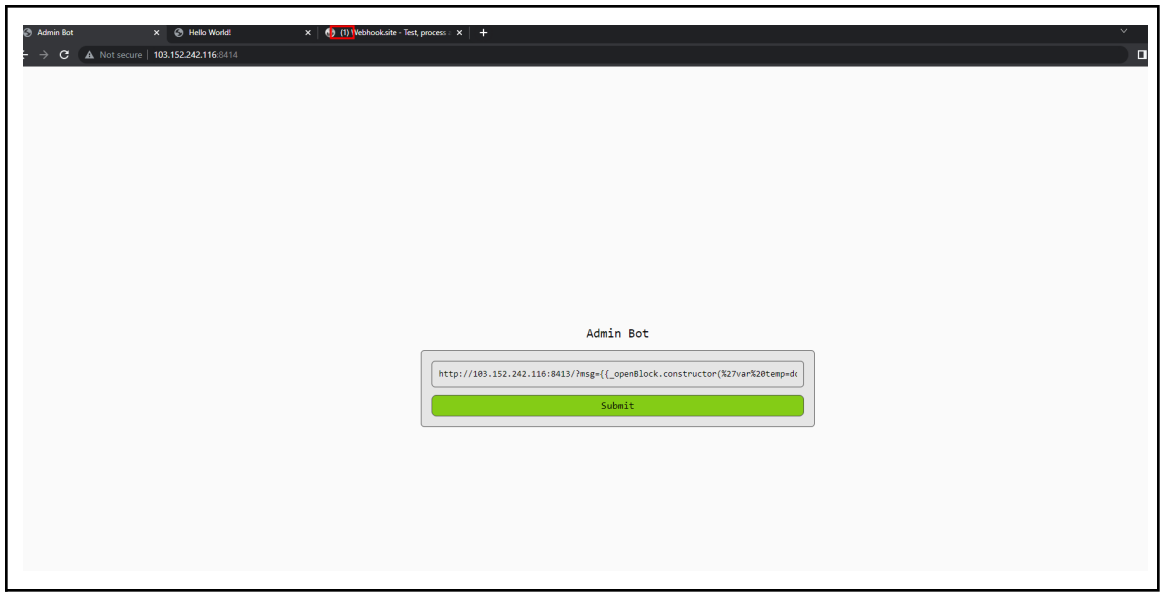
Di sini kelihatan nya payload nya gagal

Namun bila kita buka webhook nya

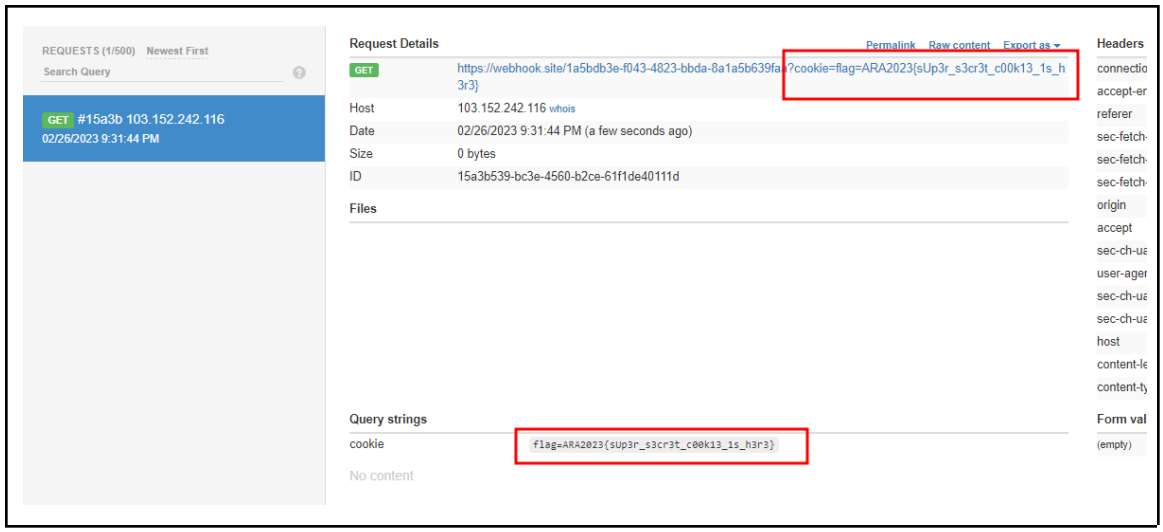


Sekarang kita tinggal kirimkan link tersebut ke bot nya



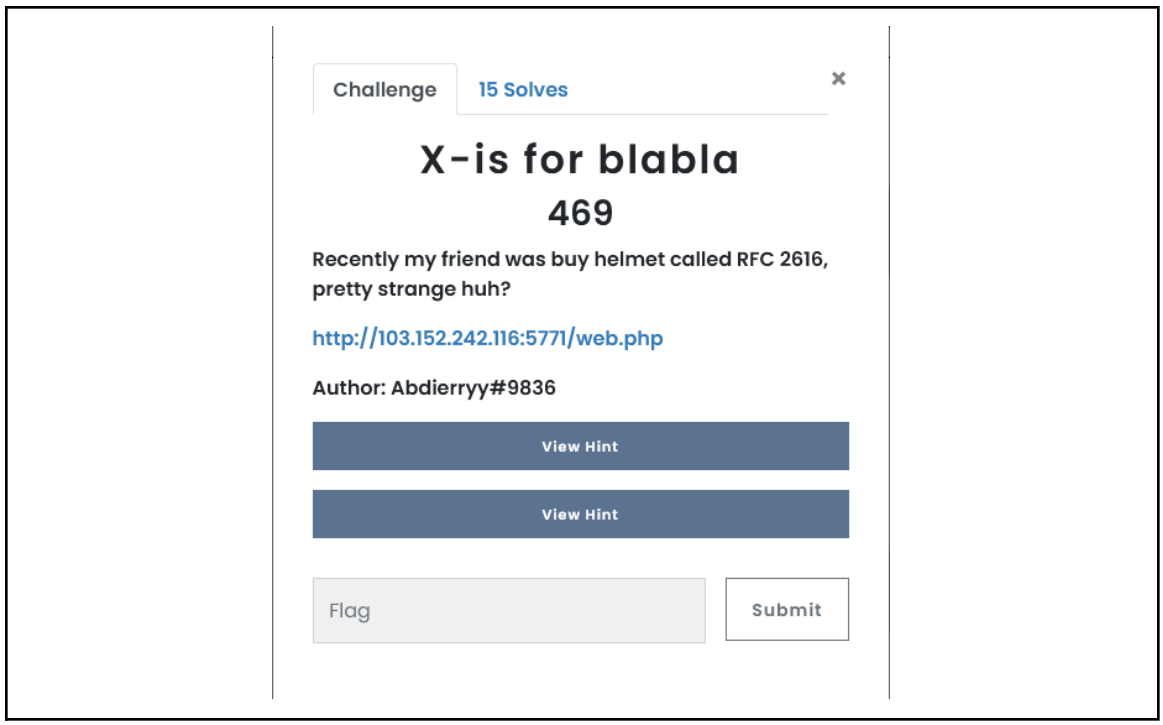


Dan kelihatan nya berhasil di dapat flag nya

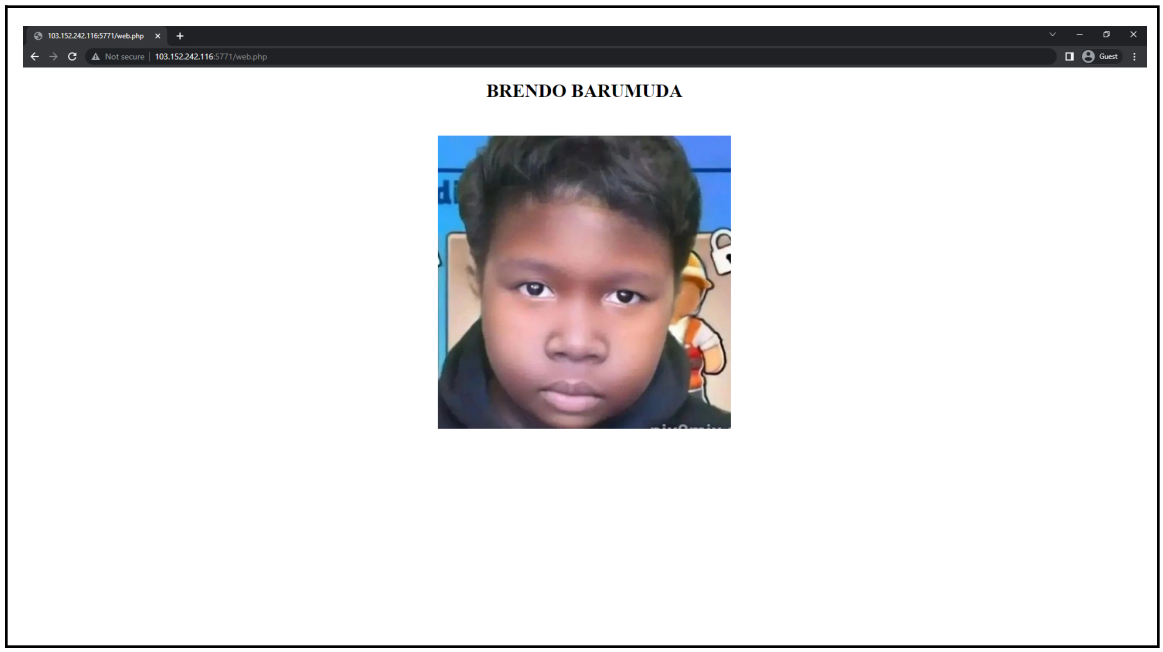


FLAG:
ARA2023{sUp3r_s3cr3t_c00k13_1s_h3r3}

- X-is for bla bla

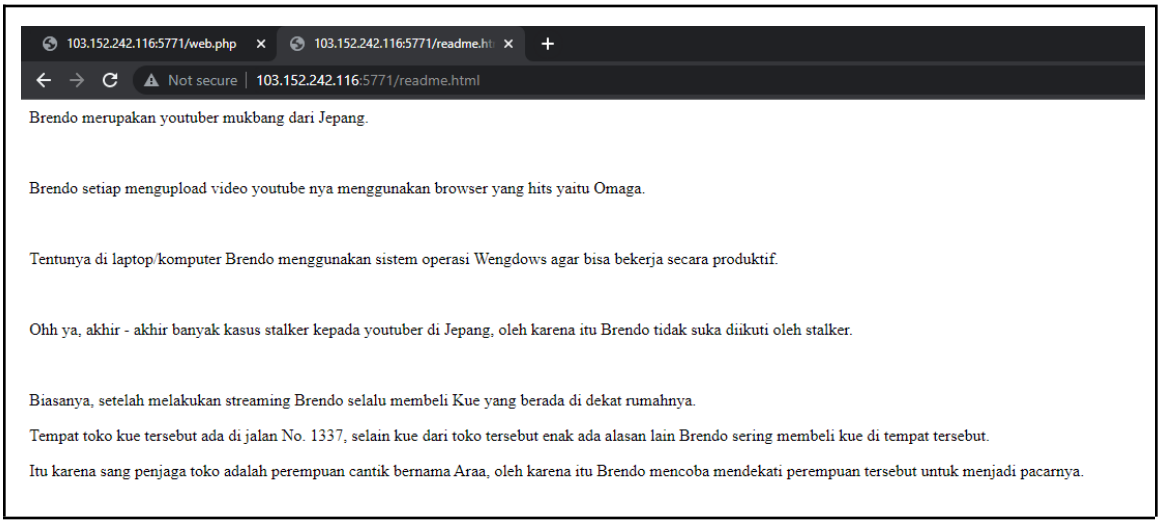


Pertama mari kita buka page nya



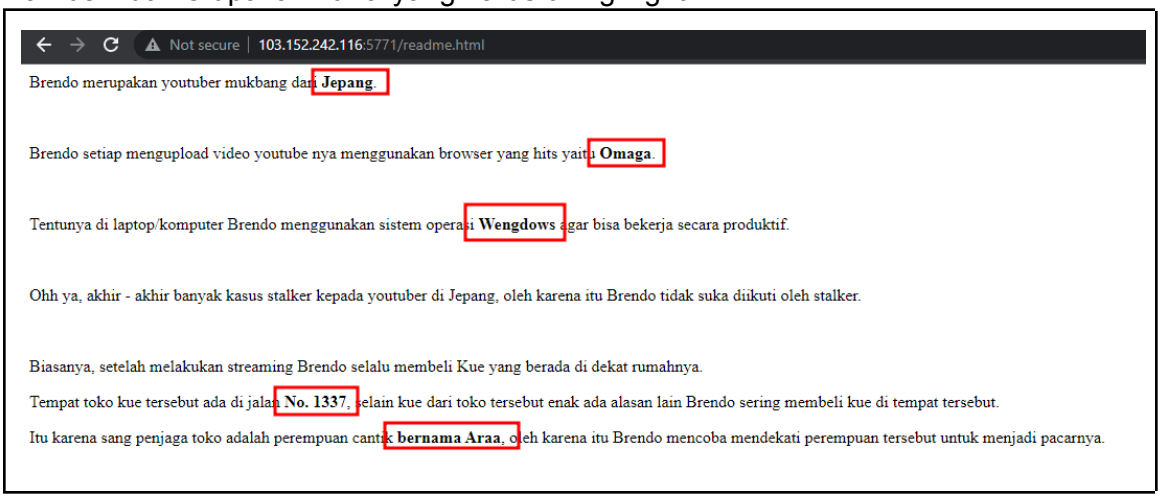
Dari sini tidak ada yang menarik, jadi langsung saja di inspect element

Dan seperti yang bisa kita lihat terdapat file bernama readme.html, dan berikut merupakan hasil page tersebut

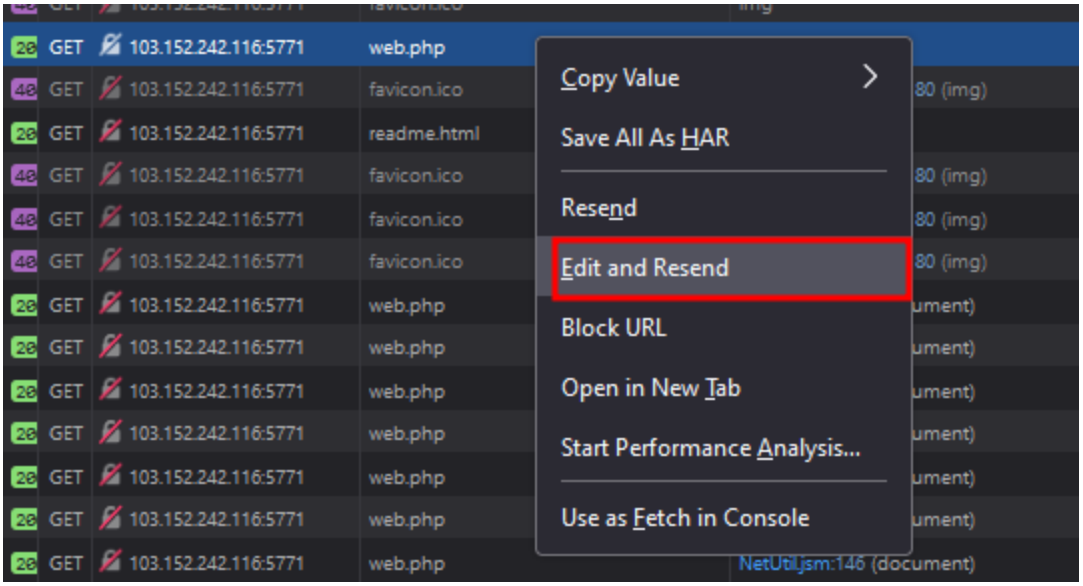


Setelah berlama melihat page tersebut kita hanya perlu mengubah header yang ada di request agar sesuai dengan deskripsi yang ada di page tersebut

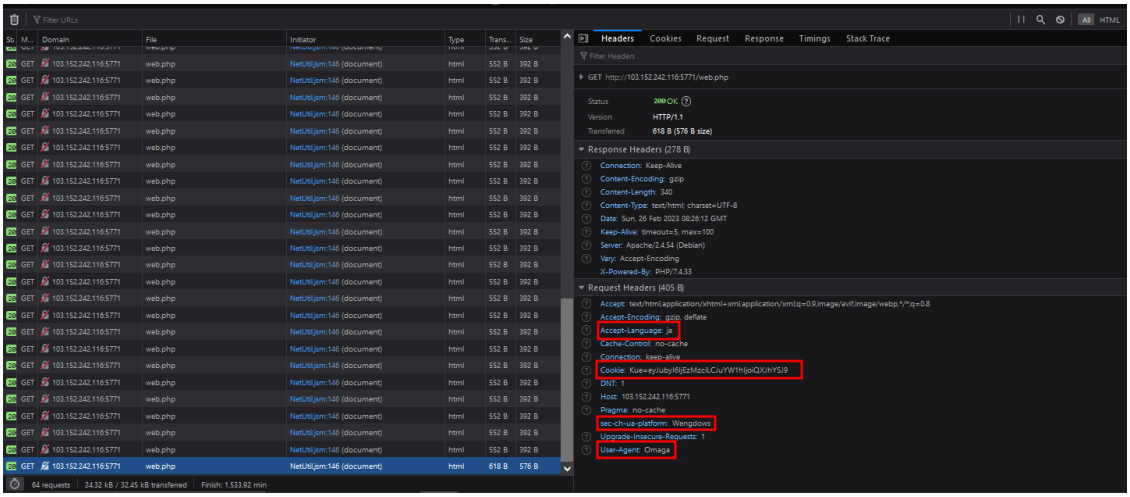
Dan berikut merupakan kunci yang harus di highlight



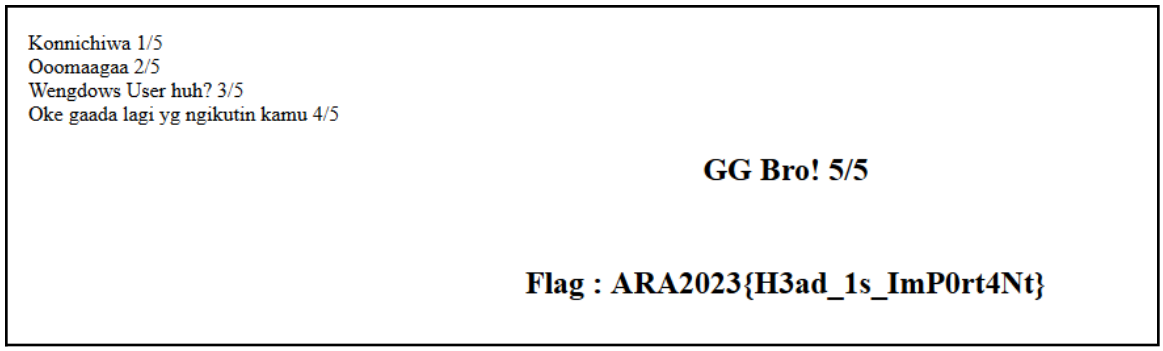
Dan karena disini saya mager buka linux, burp, dkk saya buka lagi page web tadi di firefox agar bisa mengubah request nya



Dan setelah beberapa (banyak) trial and error berikut merupakan header yang pas untuk mengeluar kan flag nya



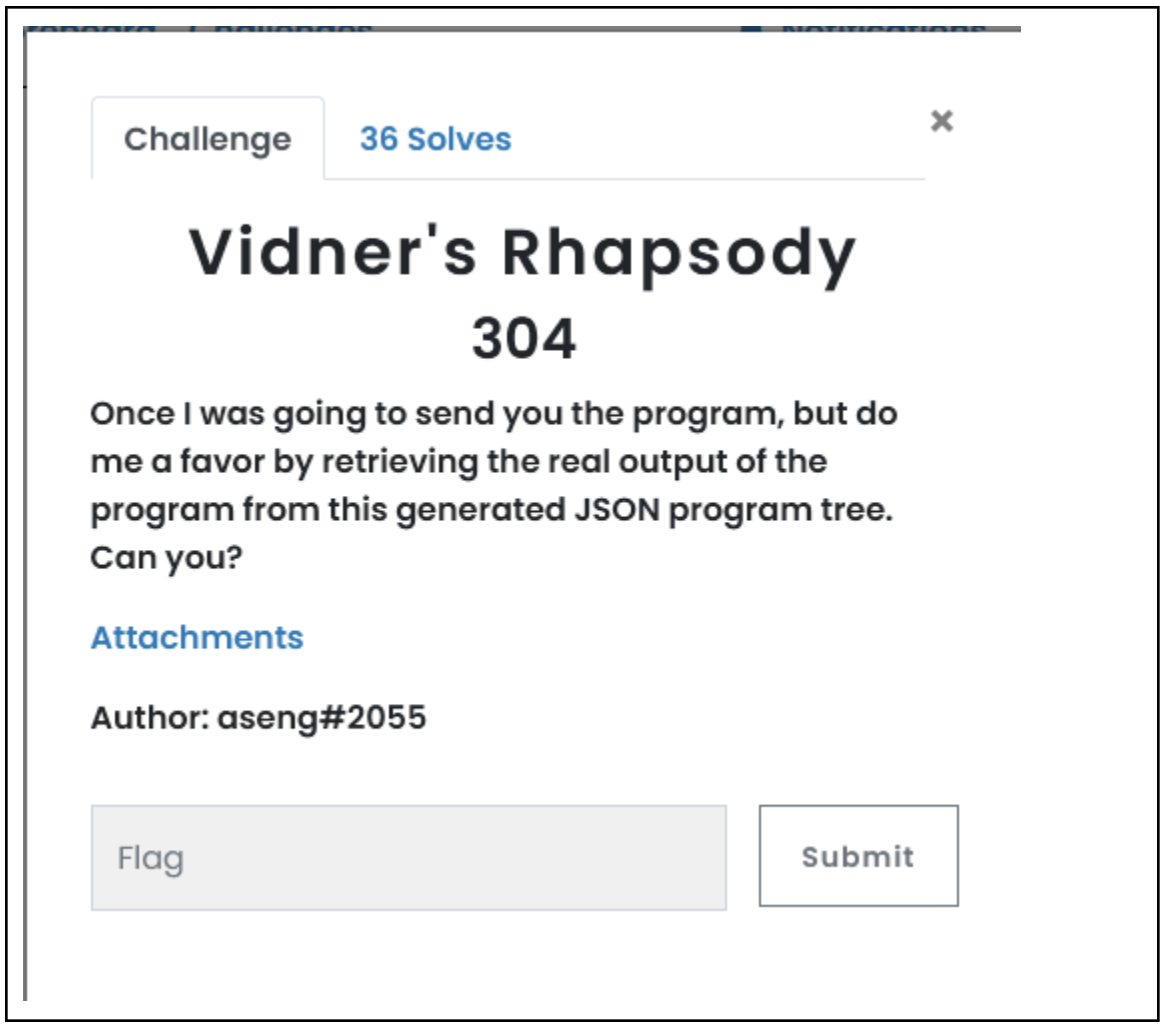
Dan hasil yang didapat adalah seperti berikut



FLAG:
ARA2023{H3ad_1s_ImP0rt4Nt}

REVERSE ENGINEERING

- Vidner's Rhapsody

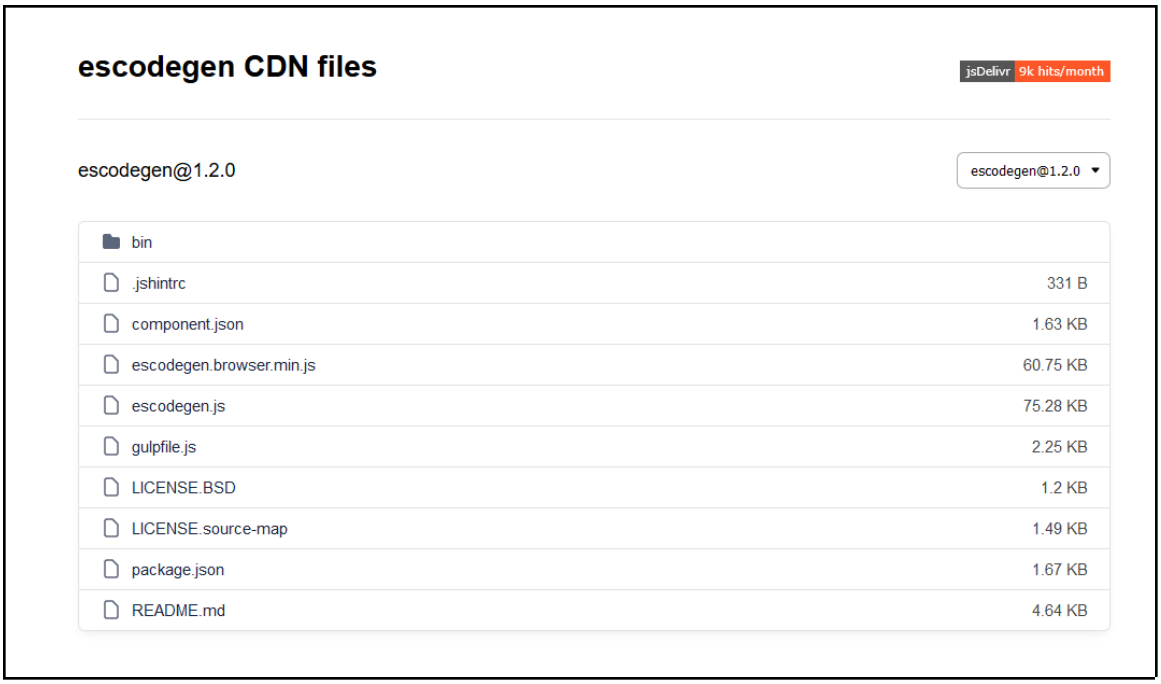


Pertama mari kita buka file nya

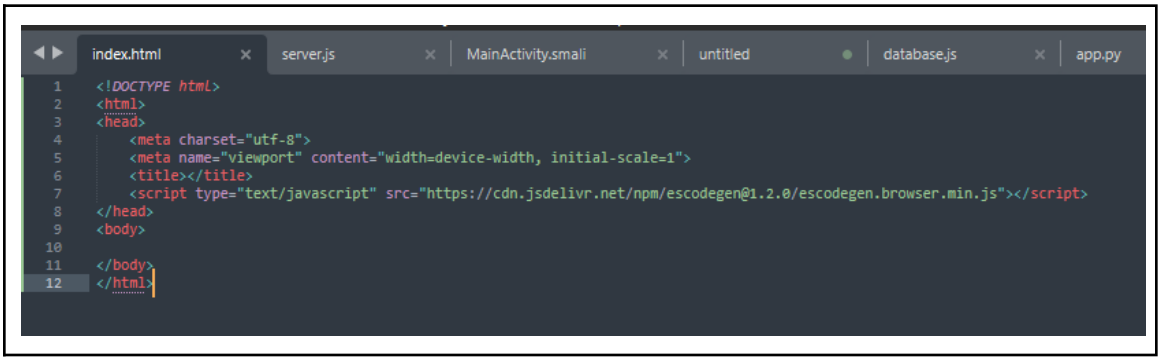
Kelihatan nya isi nya adalah json dump dari sebuah program

Setelah mencari cari di internet berikut merupakan tools yang saya temukan bisa membantu untuk mengubah json nya ke js kembali

Dan tenyata ada terdapat cdn nya



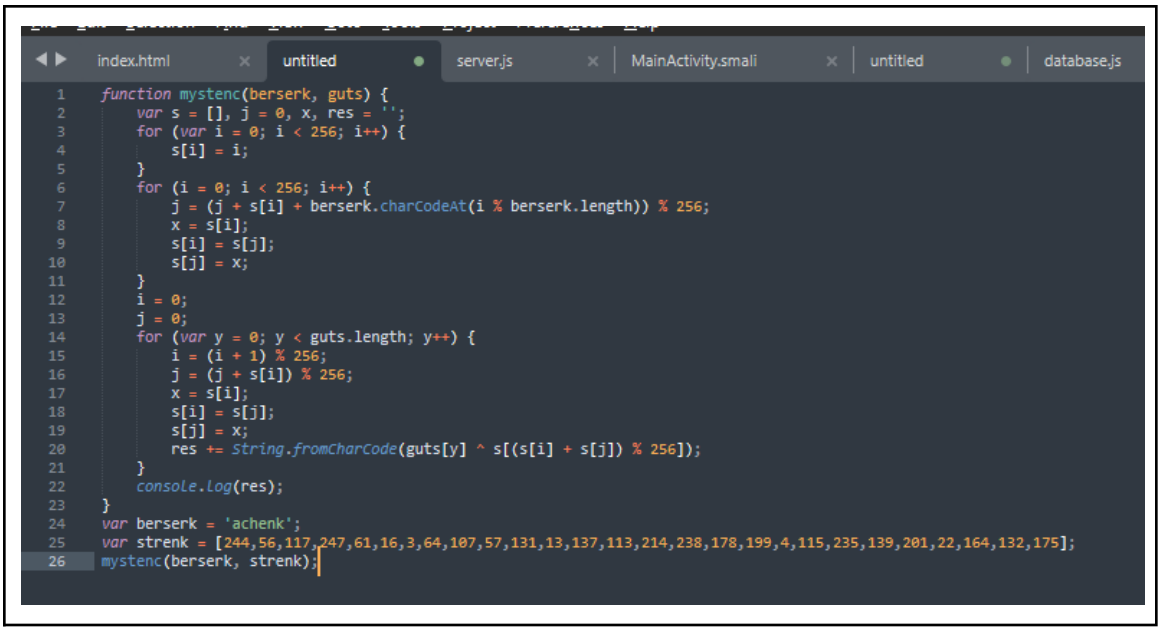
Dari sini kita bisa tinggal buat local page nya



Dari situ kita tinggal copy json dump tadi ke dalam escodegen tersebut seperti berikut

```
escodegen.generate({
  "type": "Program",
  "start": 0,
  "end": 669,
  "body": [
    {
      "type": "FunctionDeclaration",
      "start": 0,
      "end": 480,
      "id": {
        "type": "Identifier",
        "start": 9,
        "end": 16,
        "name": "mystenc"
      },
      "expression": false,
      "generator": false,
      "async": false,
      "params": [
        {
          "type": "Identifier",
          "start": 17,
          "end": 24,
          "name": "berserk"
        },
        {
          "type": "Identifier",
          "start": 26,
```

Sekarang kita tinggal jalankan, dan berikut merupakan kode yang di generate



```
function mystenc(berserk, guts) {
    var s = [], j = 0, x, res = '';
    for (var i = 0; i < 256; i++) {
        s[i] = i;
    }
    for (i = 0; i < 256; i++) {
        j = (j + s[i] + berserk.charCodeAt(i % berserk.length)) % 256;
        x = s[i];
        s[i] = s[j];
        s[j] = x;
    }
    i = 0;
    j = 0;
    for (var y = 0; y < guts.length; y++) {
        i = (i + 1) % 256;
        j = (j + s[i]) % 256;
        x = s[i];
        s[i] = s[j];
        s[j] = x;
        res += String.fromCharCode(guts[y] ^ s[(s[i] + s[j]) % 256]);
    }
    console.log(res);
}
var berserk = 'achenk';
var strenk = [244,56,117,247,61,16,3,64,107,57,131,13,137,113,214,238,178,199,4,115,235,139,201,22,164,132,175];
mystenc(berserk, strenk);
```

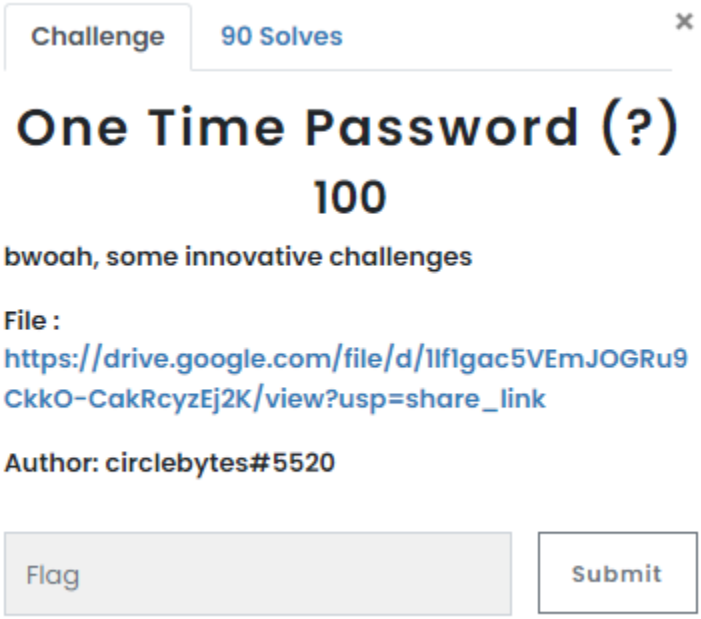
Lalu bila kita jalankan js ini di browser, berikut yang kita dapat

```
mystenc(berserk, strenk);"
function mystenc(berserk, guts) {
    var s = [], j = 0, x, res = '';
    for (var i = 0; i < 256; i++) {
        s[i] = i;
    }
    for (i = 0; i < 256; i++) {
        j = (j + s[i] + berserk.charCodeAt(i % berserk.length)) % 256;
        x = s[i];
        s[i] = s[j];
        s[j] = x;
    }
    i = 0;
    j = 0;
    for (var y = 0; y < guts.length; y++) {
        i = (i + 1) % 256;
        j = (j + s[i]) % 256;
        x = s[i];
        s[i] = s[j];
        s[j] = x;
        res += String.fromCharCode(guts[y] ^ s[(s[i] + s[j]) % 256]);
    }
    console.log(res);
}
var berserk = 'achenk';
var strenk = [244,56,117,247,61,16,3,64,107,57,131,13,137,113,214,238,178,199,4,115,235,139,201,22,164,132,175];
mystenc(berserk, strenk);
j4vAST_l!ke_84831_t0wer_lol
undefined
```

FLAG:
ARA2023{j4vAST_l!ke_84831_t0wer_lol}

CRYPTOGRAPHY

- One Time Password (?)



Tahap pengerjaan:

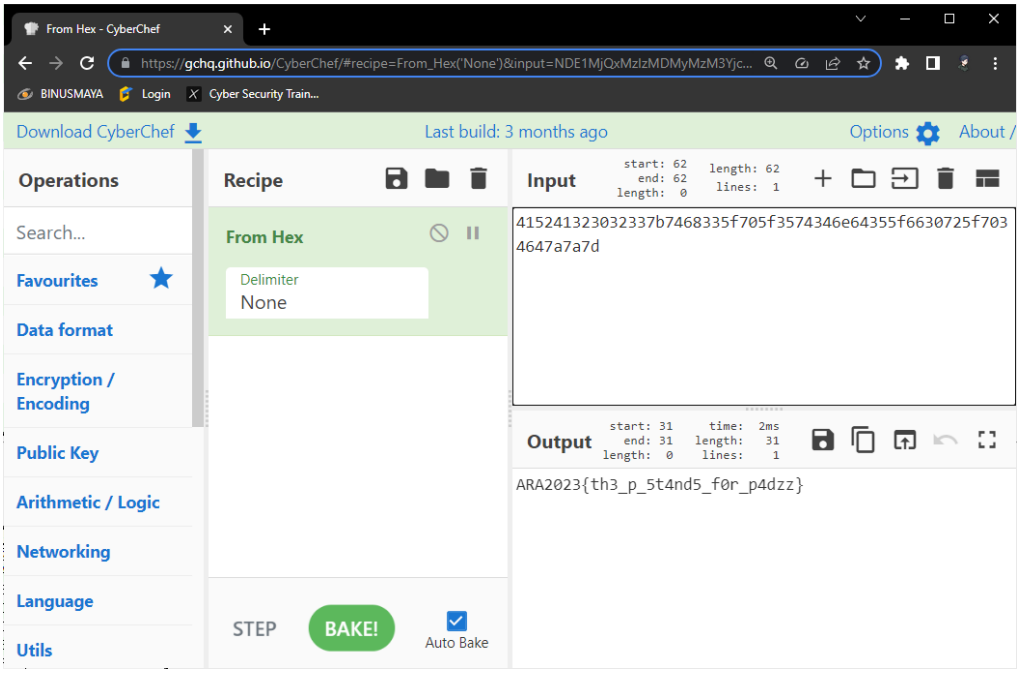
1. Download file dalam *google drive*

```
A: 161a1812647a765b37207a1c3b1a7b54773c2b660c46643a1a50662b3b3e42
B: 151d616075737f322e2d130b381666547d3d4470054660287f33663d2a2e32

XOR: 415241323032337b7468335f705f3574346e64355f6630725f7034647a7a7d
```

   Didapati 3 value sebagai berikut.

2. Gunakan *cyberchef* untuk menganalisa ketiga file, dan ternyata didapati pada hasil **XOR** merupakan flagnya



FLAG : ARA2023{th3_p_5t4nd5_f0r_p4dzz}

- Secrets Behind a Letter

Tahap pengerjaan:

1. Download file dari *Attachments* lalu didapat beberapa value seperti ini

```
p:
125753336941212676905219718556916381441368103311882482367708803389058118834850641048656498349278197256176955544721003413618961620223116533
1532810101344273
q:
124974834261750724658521679369605262322848918767879810806711627835614115216758091122045736173583897427325462935027095851292058857260784924
7109867512398747
c:
360629344957317929086395350628331806510228135895355928518025722643282990274064139273468524542176277933151448929420268869808236222401574057174
749978795994304054073412214283889848276754127677837091303824669912963572714656139422011853028133556111405072526509839846701570133437746102
72764498234471257184433228021 8

e = 65537
```

2. Diketahui bahwa ini merupakan **RSA**, karena identik dengan p,q,c, dan e.

   Untuk mendapatkan nilai n saya mengalikan p dan q

```
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 125753336941212676905219718556916381441368103311882482367708803389058118834850641048656498349278197256176955544721003413618961620223116533015328101013442 73 * 124974834261750724658521679369605262322848918767879810806711627835614115216758091122045736173583897427325462935027095851292058857260784924171098675123987 47
157160024420901491275151303069558764137050893302421347934387021008089237099810507796321179096121981028783897823233363219558663447491261939665357360882456316085702311589787337774153460302744843626838158086562582492360568567614325107754450077486851324804696560335578971886327235046960425388107307559439500825931
```

   n =

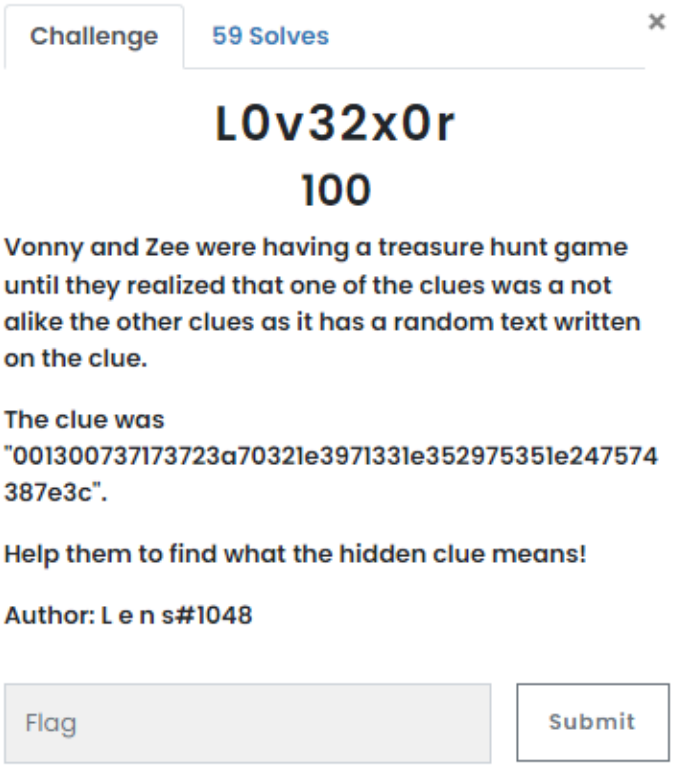
   157160024420901491275151303069558764137050893302421347934387021008089237099810507796321179096121981028783897823233363219558663447491261939665357360882456316085702311589787337774153460302744843626838158086562582492360568567614325107754450077486851324804696560335578971886327235046960425388107307559439500825931

626838158086562582492360568567614325107754450077486851324804696560335578971886327235046960425388107307559439500825931

3. Lalu gunakan tools sepert *dcoder* untuk RSA decryption
   https://www.dcode.fr/rsa-cipher

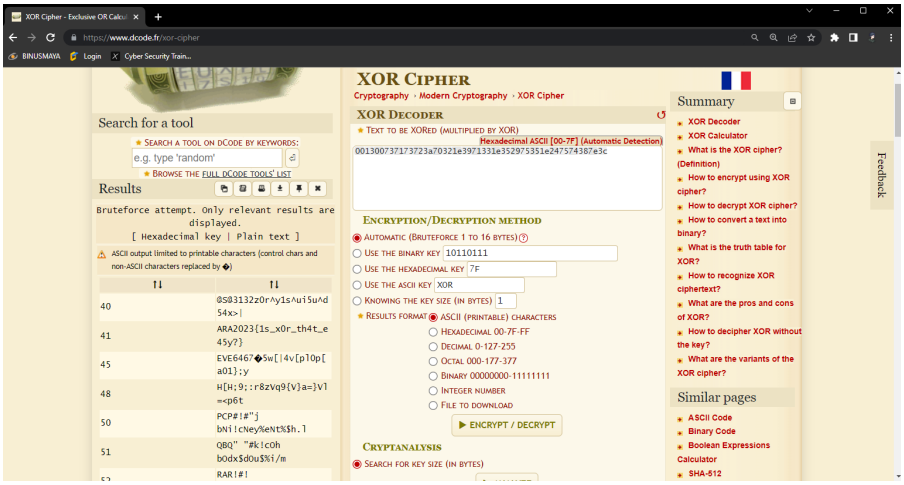
FLAG : ARA2023{1t_turn5_0ut_to_b3_an_rsa}

- L0v32x0r



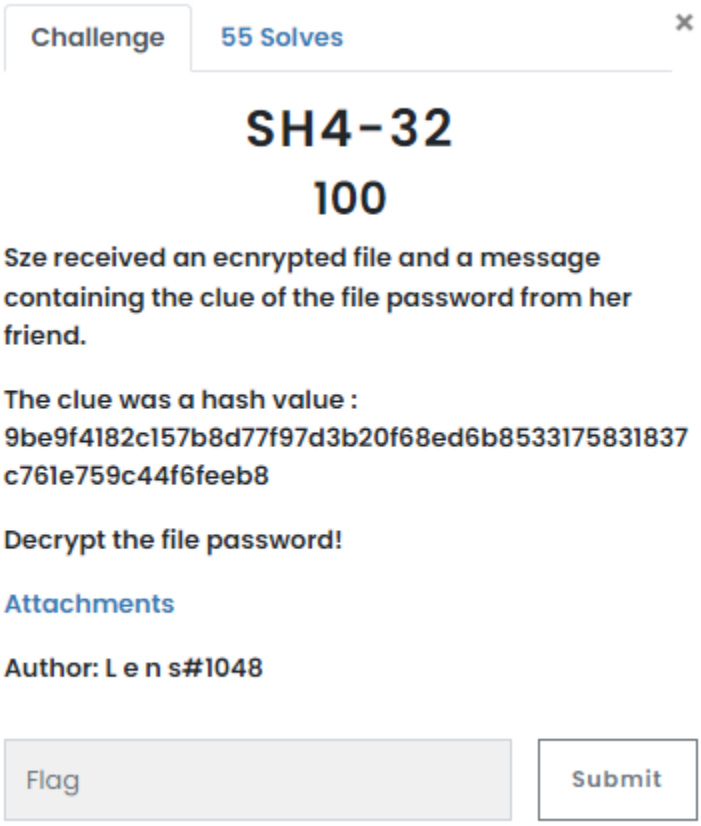
Tahap pengerjaan:

1. Disini pertama didapati hint pada judul soal : Love to XOR, jadi saya menggunakan *dcoder* lagi untuk memproses cluenya https://www.dcode.fr/xor-cipher



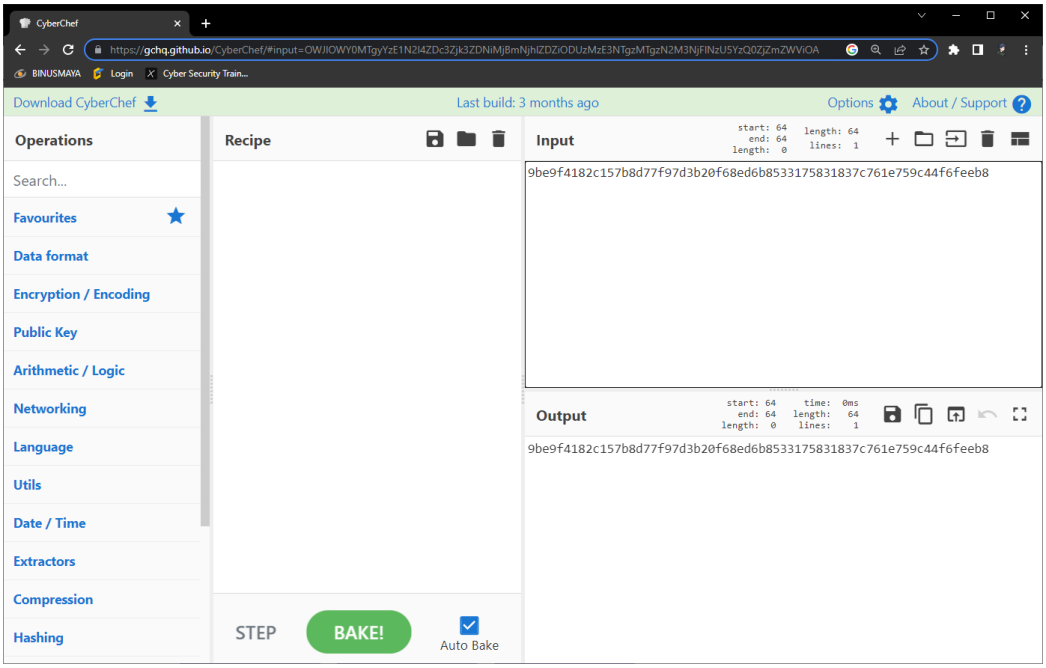
FLAG : ARA2023{1s_x0r_th4t_e45y?}

- SH4-32

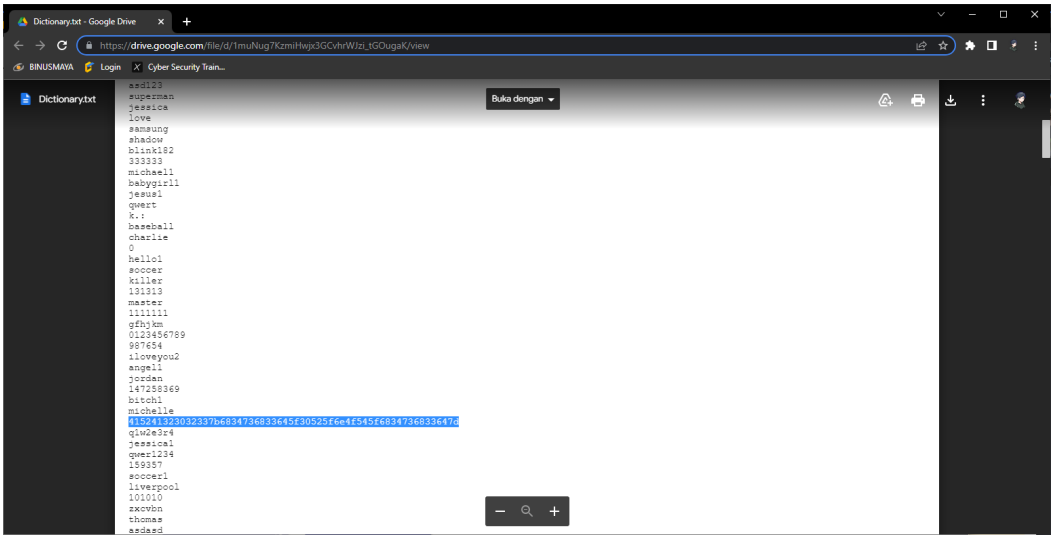


Tahap pengerjaan:

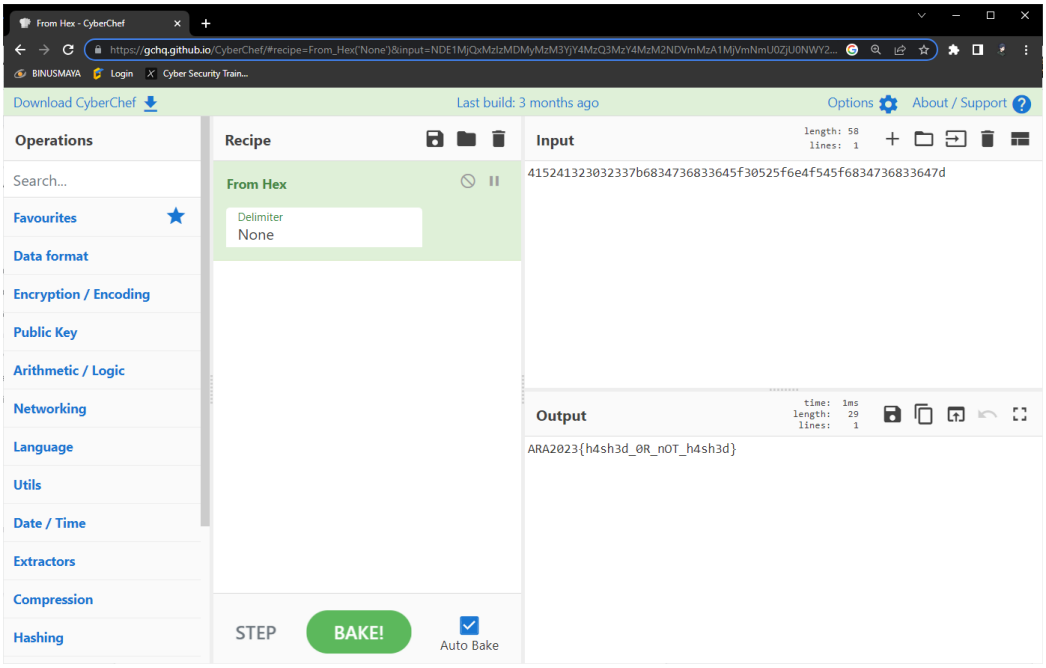
1. Disini pertama saya mencoba untuk memproses cluenya menggunakan *cyberchef* namun tidak mendapatkan apa apa



2. Karena tidak mendapatkan apa-apa, saya membuka *Attachment* nya sekilas, dan saya melihat ada string yang berbeda sendiri, yaitu `415241323032337b6834736833645f30525f6e4f545f6834736833647d`

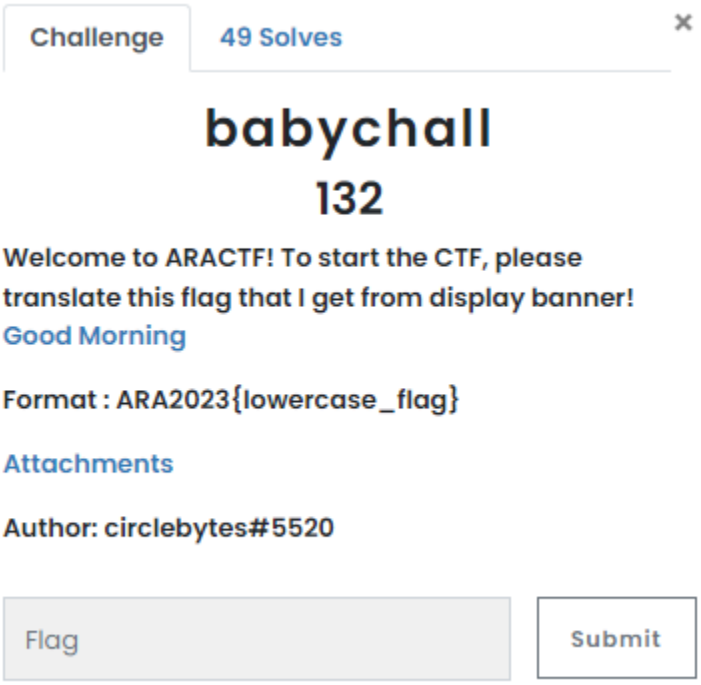


3. Kembali menggunakan *cyberchef* saya mendapati hasil yang ternyata flagnya



FLAG : ARA2023{h4sh3d_0R_nOT_h4sh3d}

- babychall



Tahap pengerjaan:

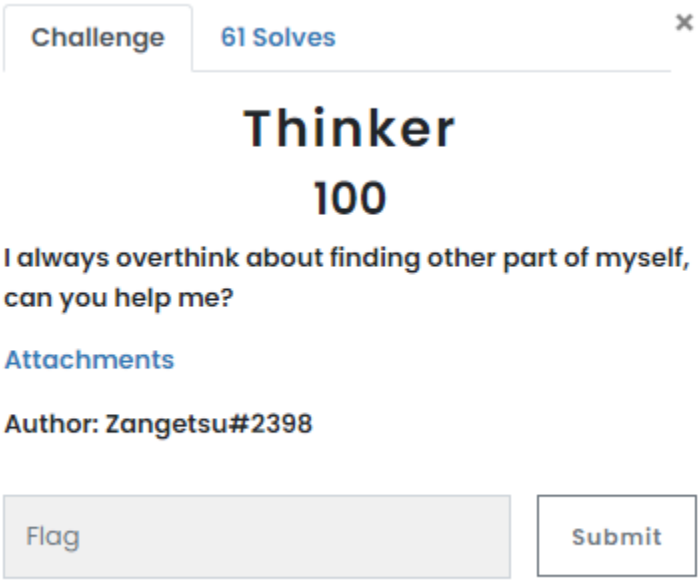
1. Disini saya langsung menggunakan tools Cracking RSA with Chinese Remainder online dengan web https://asecuritysite.com/rsa/rsa_ctf02

FLAG :

ARA2023{s00000_much_c1ph3r_but_5m4ll_e_5t1ll_d0_th3_j0b}

FORENSIC

- Thinker



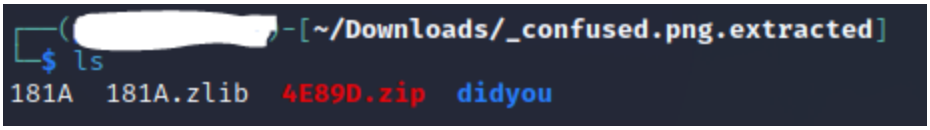
Tahap pengerjaan:

1. Pertama download file png pada *Attachments*. Lalu disini saya mencoba menganalisis png menggunakan **binwalk**

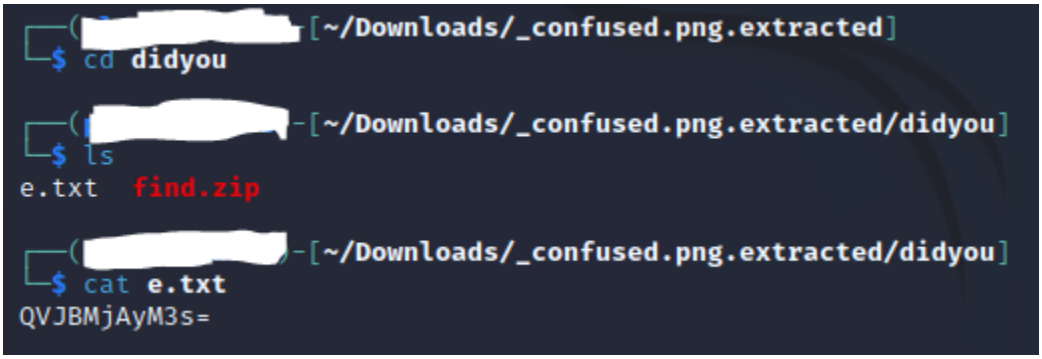
```
└─$ binwalk /home/kali/Downloads/confused.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0×0             PNG image, 720 x 881, 8-bit/color RGB, non-interlaced
6170          0×181A          Zlib compressed data, best compression
321663        0×4E87F         TIFF image data, big-endian, offset of first image directory: 8
321693        0×4E89D         Zip archive data, at least v1.0 to extract, name: didyou/
321758        0×4E8DE         Zip archive data, at least v1.0 to extract, compressed size: 13, un
compressed size: 13, name: didyou/e.txt
321841        0×4E931         Zip archive data, at least v1.0 to extract, compressed size: 10568,
 uncompressed size: 10568, name: didyou/find.zip
332460        0×512AC         End of Zip archive, footer length: 22
332726        0×513B6         End of Zip archive, footer length: 22
```

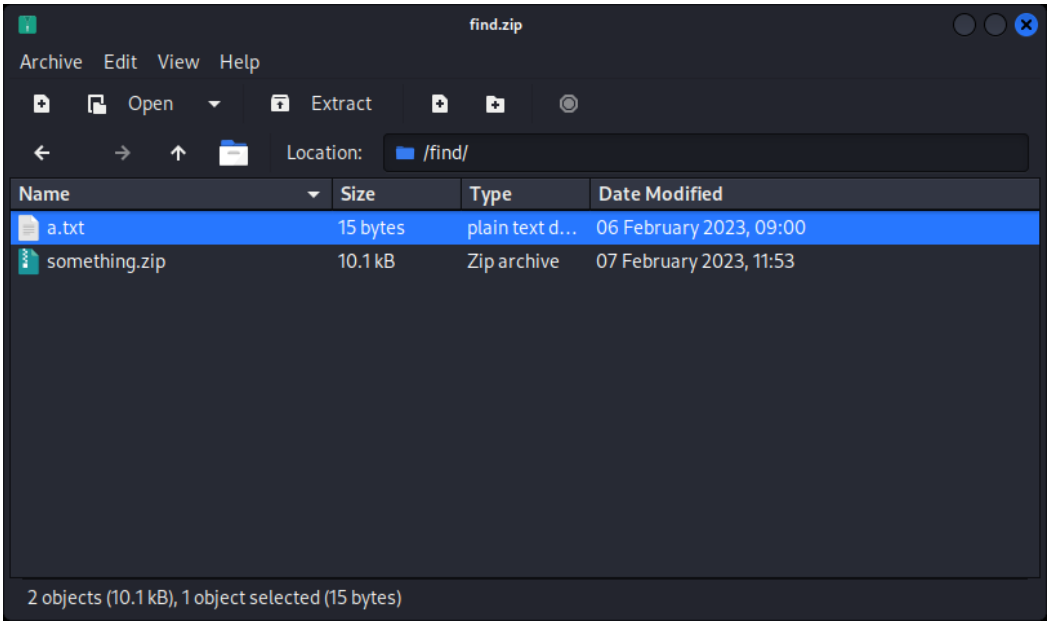
2. Karena disitu ada file *zip* saya mencoba mengekstraknya untuk membaca filenya, dan setelah itu didapati file sebagai berikut



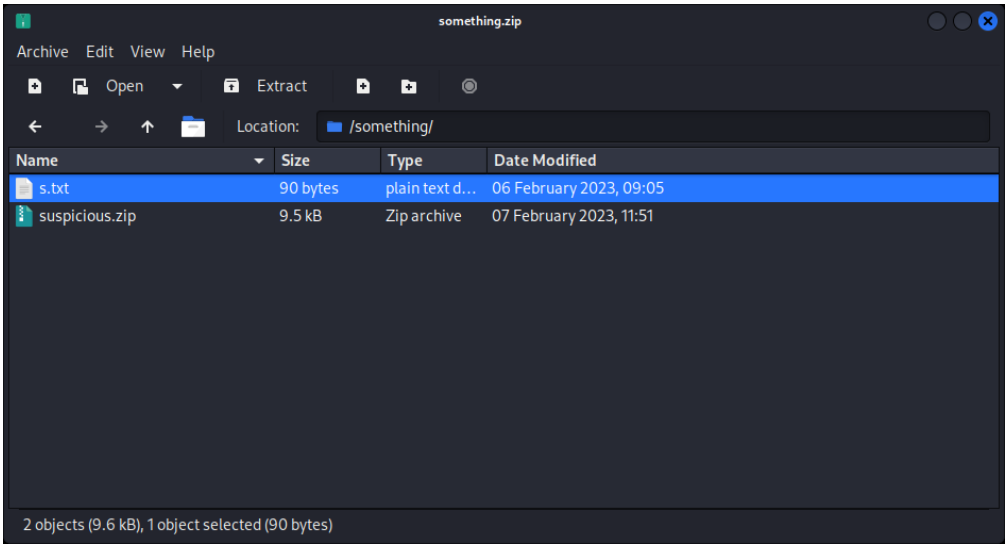
3. Step berikutnya saya coba masuk kedalam file dan menemukan potongan flag sebagai berikut, dan saya menggunakan *cyberchef* lagi untuk mengubah stringnya menjadi bisa terbaca



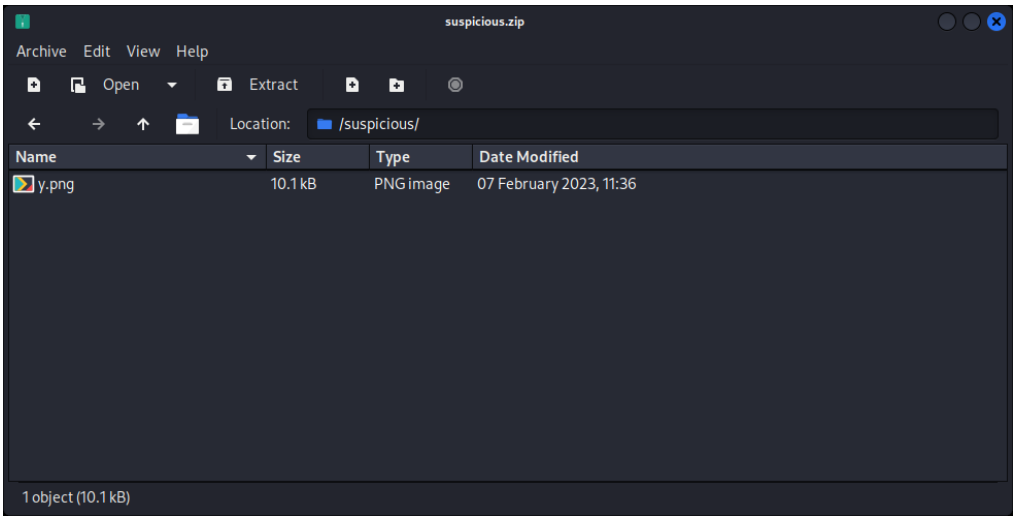
QVJBMjAyM3s= : **ARA2023{** (base64)



35216D706C335F : **5!mpl3_** (hex)

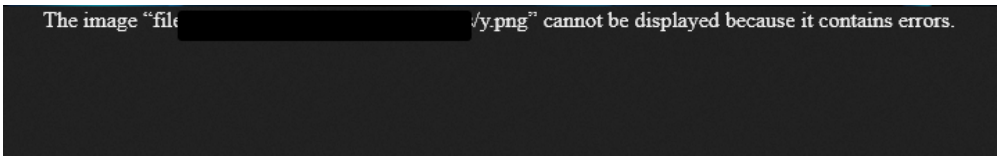


01000011 00110000 01110010 01110010 01110101 01110000 01110100 00110011 01100100 01011111 : **C0rrupt3d_** (binary)



Dan terakhir ada **y.png**

4. Setelah kita extract, kelihatan nya file nya tidak dapat di buka,



kalau kita analisa file nya kita mendapat error seperti berikut

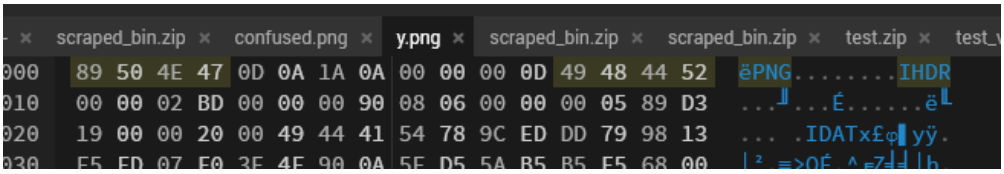
Use sample file:

Read local file: Browse... y.png

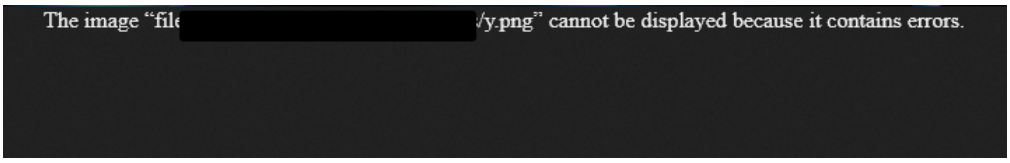
Chunk summary:

Start offset	Raw bytes	Chunk outside	Chunk inside	Errors
0	21 5a 78 52 0d 0a 1a 0a	• Special: File signature • Length: 8 bytes	• "!ZxR␍␊␚␊"	• Value mismatch
8	00 00 00 0d 52 52 48 5c 00 00 02 bd 00 00 00 90 08 06 00 00 00 05 89 d3 19 00 00 20 00 49 44 41 54 78 9c ed dd 79 98 13 f5 fd 07 f0 3e 4f 90 0a 5e d5 5a b5 b5 f5 68 00 59 b9 14 41 51 a1 08 5a c5 7a 50 0f 44 94 ... 86 5e 22 22 22 22 b2 bc ff 07 d7 ab 8b fa 4a 0f 93 55 00 00 00 00 49 45 4e 44 ae 42 60 82	• Special: Unknown • Length: 10 099 bytes		• Unknown format

Dari sini kita tinggal buka file nya dan edit sedikit fle nya



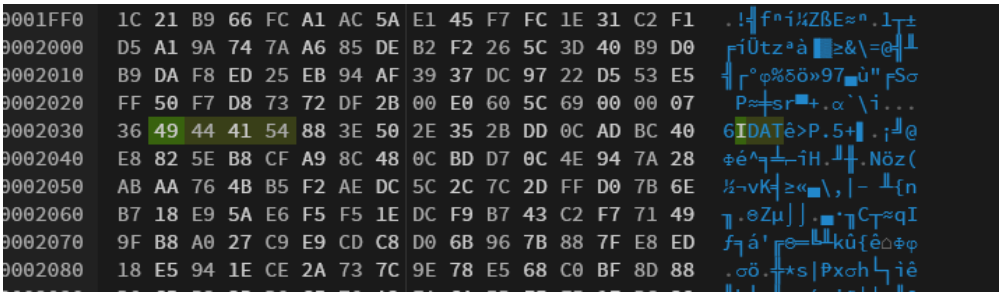
Setelah kita buka image nya seperti nya masih error



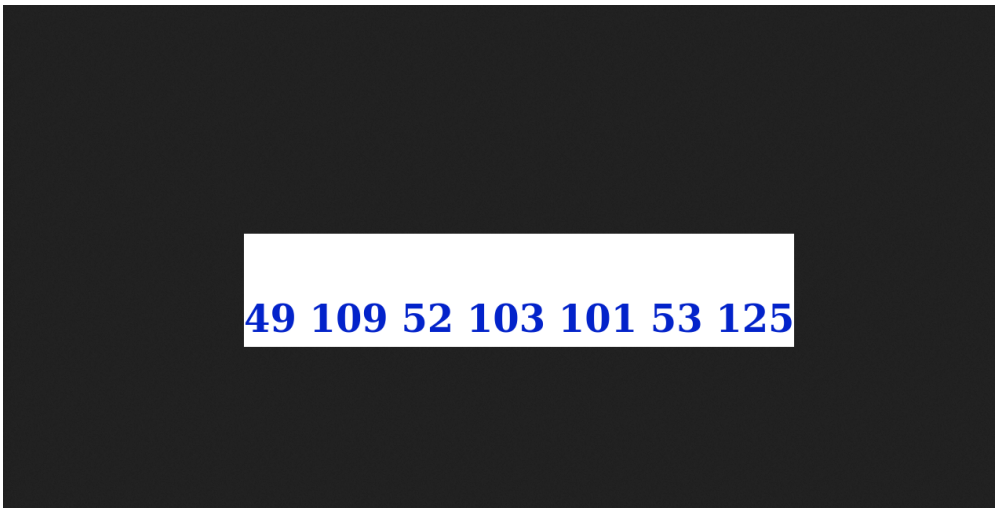
Lalu bila di buka lagi file tersebut di tools untuk analisa png berikut hasil nya

		• CRC-32: E0605C69		
8 237	00 00 07 36 89 50 4e 47 88 3e 50 2e 35 2b dd 0c ad bc 40 e8 82 5e b8 cf a9 8c 48 0c bd d7 0c 4e 94 7a 28 ab aa 76 4b b5 f2 ae dc 5c 2c 7c 2d ff d0 7b 6e b7 18 e9 5a e6 f5 f5 1e dc f9 b7 43 c2 f7 71 49 9f b8 a0 ... 22 22 b2 3c 86 5e 22 22 22 22 b2 3c 86 5e 22 22 22 22 b2 bc ff 07 d7 ab 8b fa 4a 0f 93 55	• Data length: 1 846 bytes • Type: □PNG • Name: Unknown • Critical (0) • Public (0) • Reserved (0) • Unsafe to copy (0) • CRC-32: 4A0F9355		• Type contains non-alphabetic characters • CRC-32 mismatch (calculated from data: 8AF57803)
10 095	00 00 00 00 49 45 4e 44 ae 42 60 82	• Data length: 0 bytes		

Setelah menemukan problem nya tinggal kita patch lagi



Dan kalau kita buka



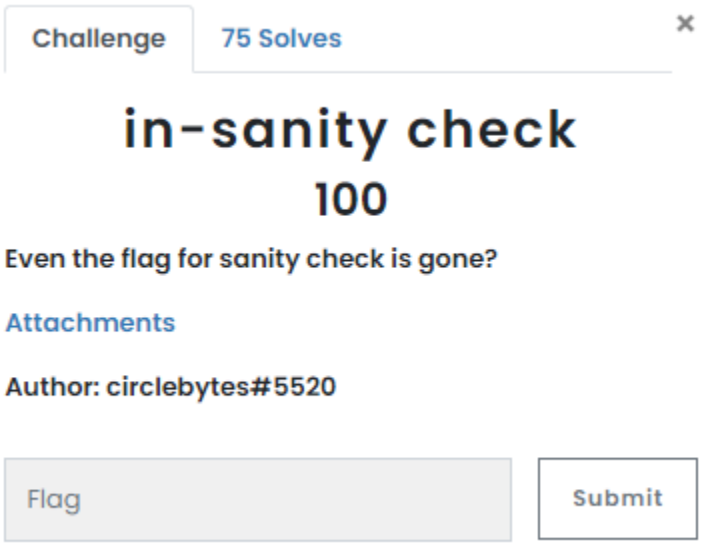
Di dapat image dengan angka angka, dan kalau di translate dari dec ke ascii hasil nya adalah

5. Setelah disusun didapati flagnya

FLAG : ARA2023{5!mpl3_C0rrupt3d_1m4ge5}

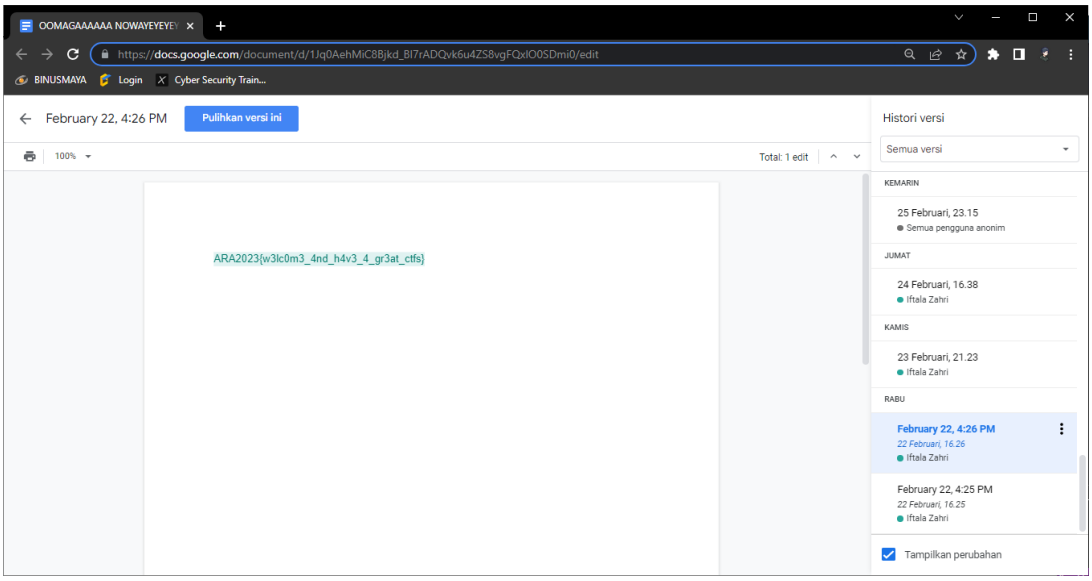
MISC

- in-sanity check



Tahap pengerjaan:

1. Pertama buka *Attachments* dan ternyata disini merupakan file google docs yang dapat diedit-edit, dan kebetulan pada saat saya buka flagnya sudah tidak ada, disini saya memeriksa *file history* dan ternyata flag dapat dilihat di pembaharuan ke dua

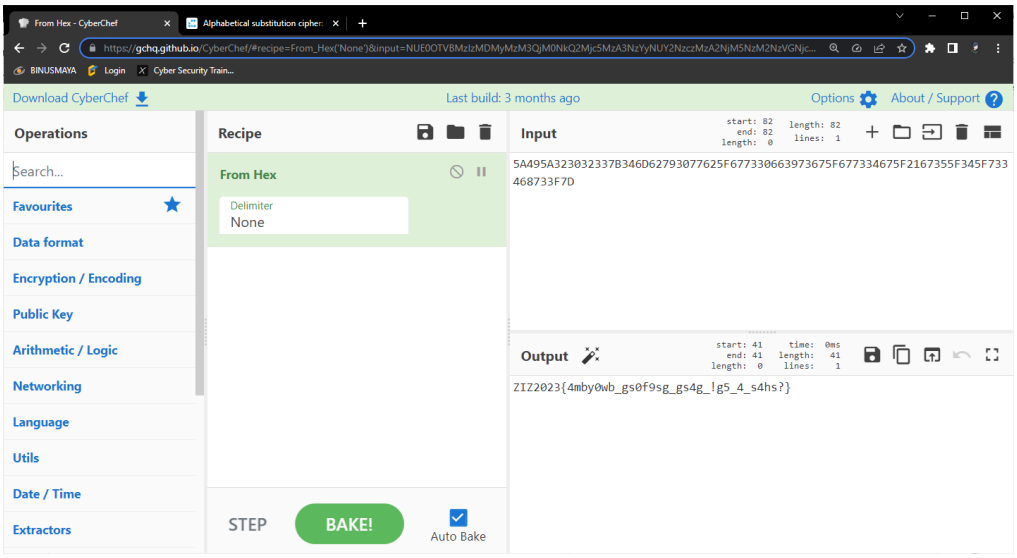


FLAG : ARA2023{w3lc0m3_4nd_h4v3_4_gr3at_ctfs}

- @B4SH

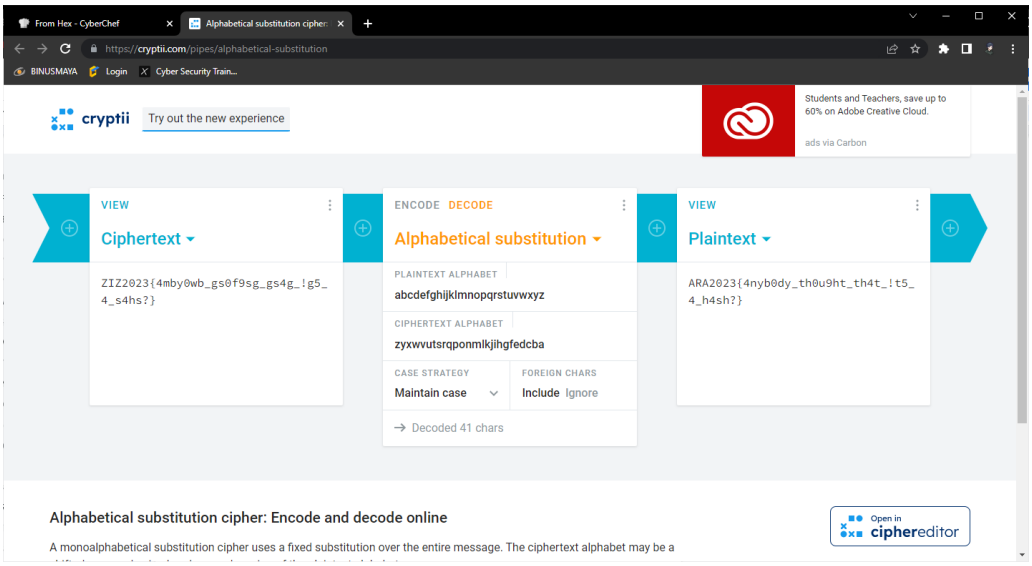
Tahap pengerjaan:

1. Pertama saya menggunakan *cyberchef* untuk memeriksa pesan



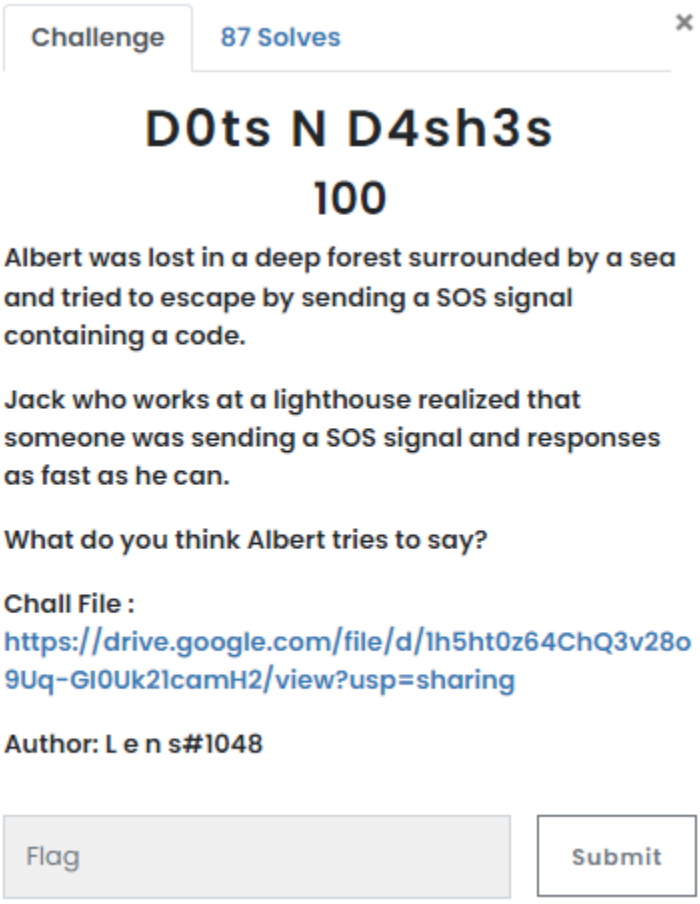
Didapati pesan: ZIZ2023{4mby0wb_gs0f9sg_gs4g_!g5_4_s4hs?}

2. Dari sini saya menggunakan bantuan tools dari https://cryptii.com/pipes/alphabetical-substitution dan ternyata berhasil didapati flagnya



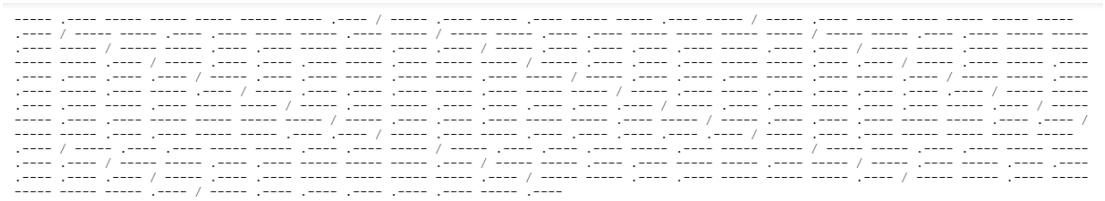
FLAG : ARA2023{4nyb0dy_th0u9ht_th4t_!t5_4_h4sh?}

- D0ts N D4sh3s

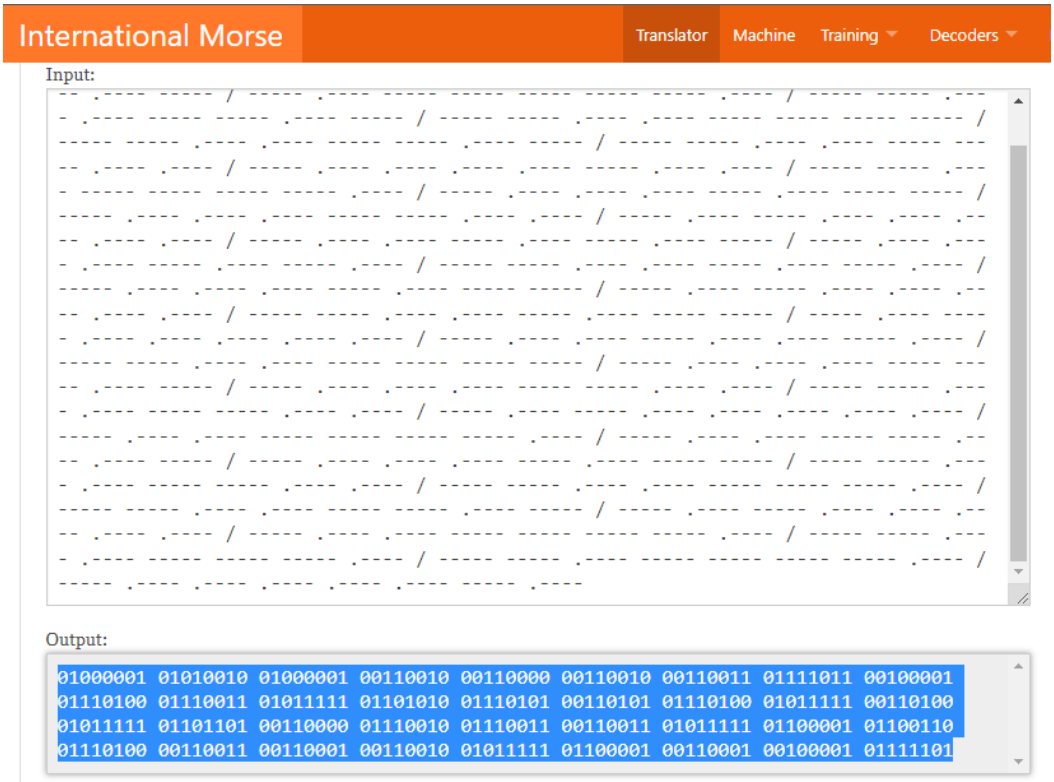


Tahap pengerjaan:

1. Pertama buka file pada *google drive*, ternyata berisi file sandi morse



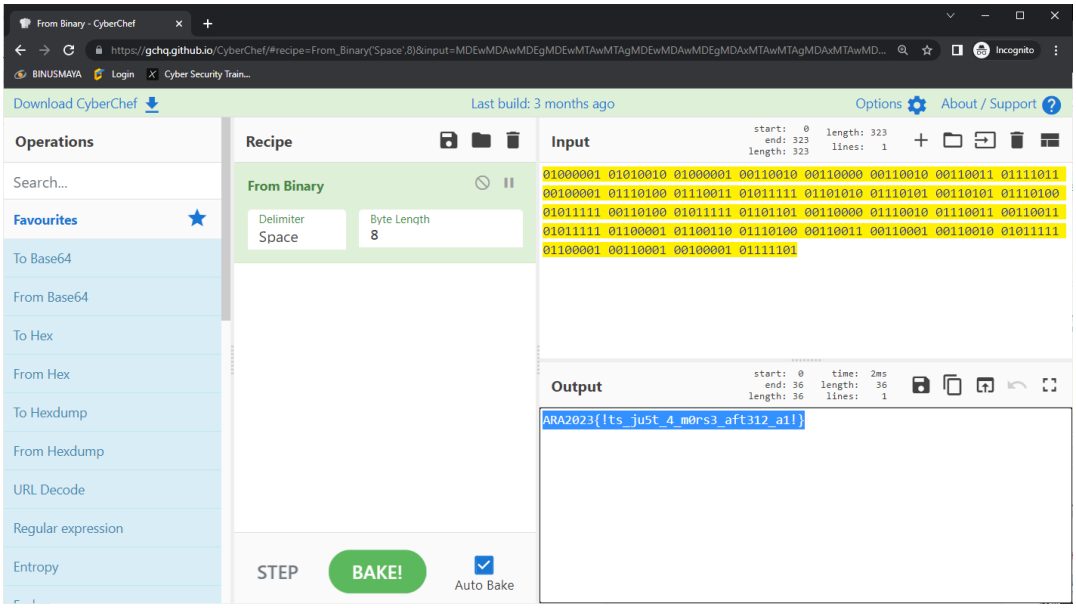
2. Disini saya menggunakan tools: https://morsecode.world/international/translator.html untuk menerjemahkan sandi morse, dan didapati hasil binary seperti ini



Hasil:

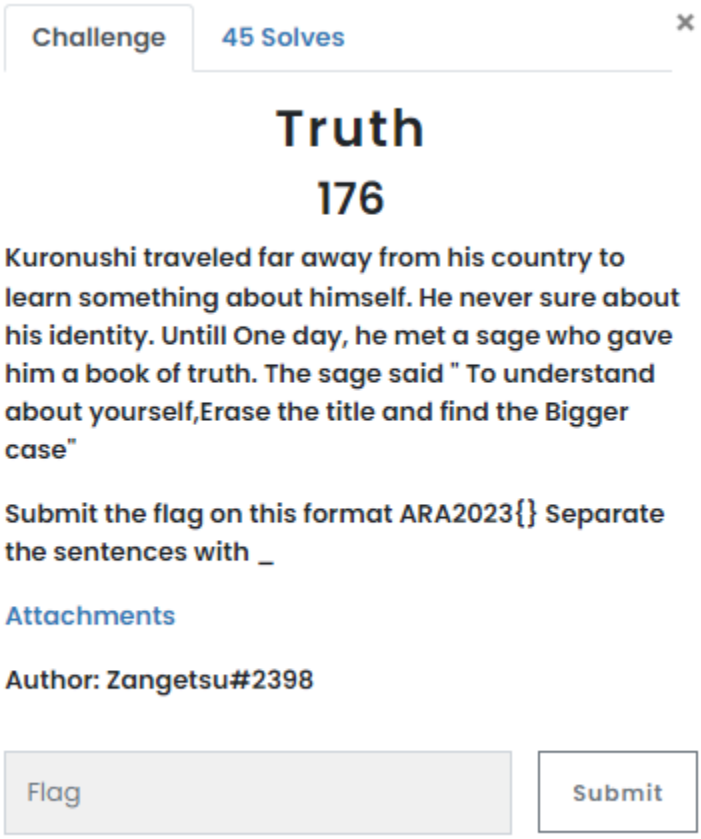
```
01000001 01010010 01000001 00110010 00110000 00110010
00110011 01111011 00100001 01110100 01110011 01011111
01101010 01110101 00110101 01110100 01011111 00110100
01011111 01101101 00110000 01110010 01110011 00110011
01011111 01100001 01100110 01110100 00110011 00110001
00110010 01011111 01100001 00110001 00100001 01111101
```

3. Terjemahkan lagi menggunakan *cyberchef*, dan flag ditemukan



FLAG : ARA2023{!ts_ju5t_4_m0rs3_aft312_a1!}

- Truth



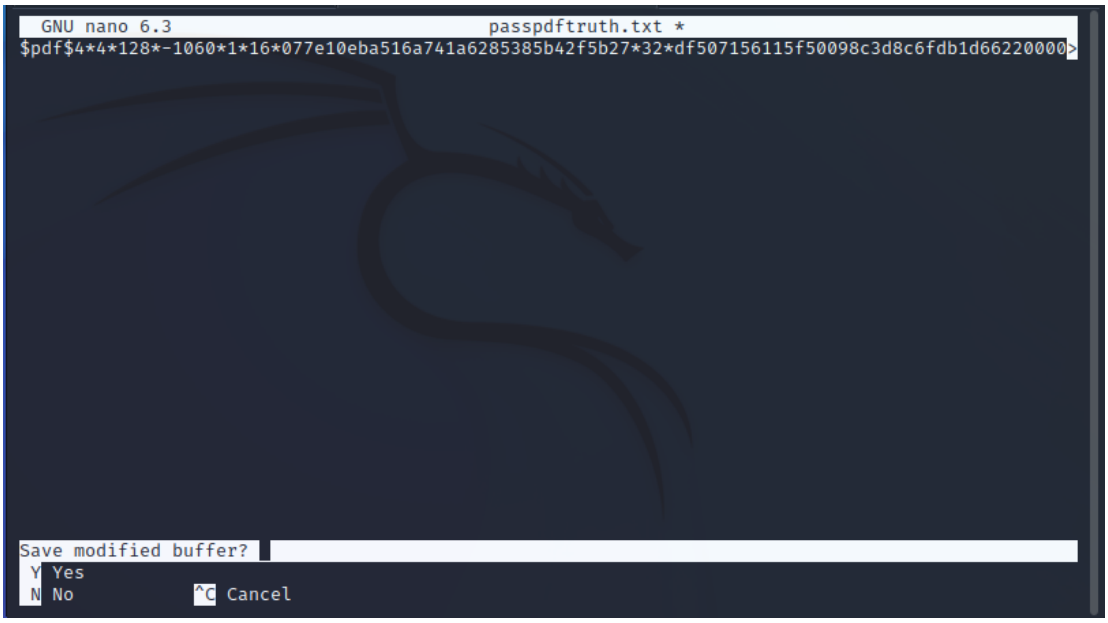
Tahap pengerjaan:

1. Pertama download file pdf pada *attachment*. Disini pdf dikunci jadi perlu dipecahkan dulu passwordnya.
2. Saya menggunakan **pdf2john** dan **hashcat** untuk melakukan *bruteforce* passwordnya

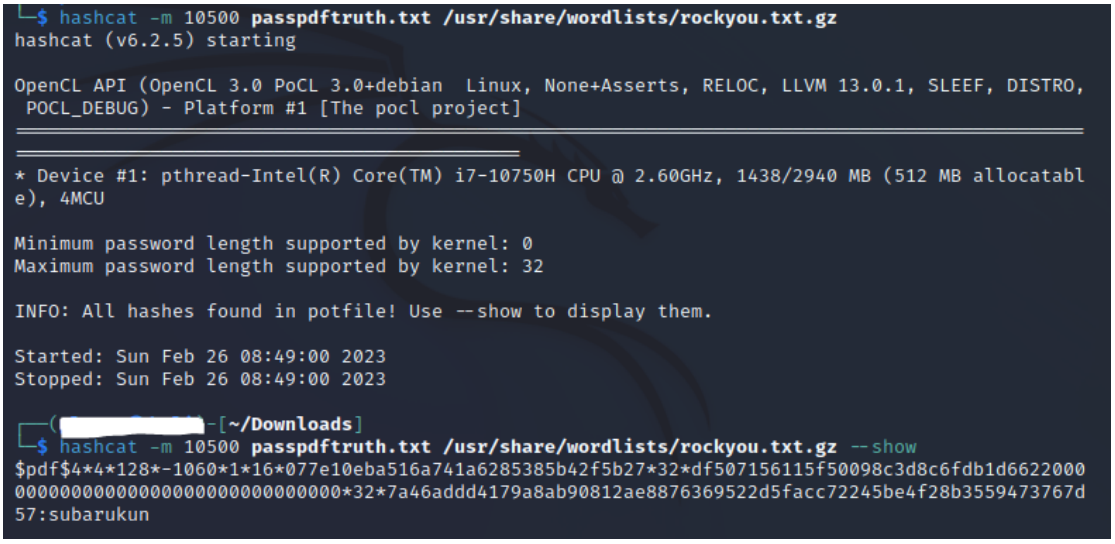
```
└─$ ./pdf2john.pl /home/kali/Downloads/Truth.pdf > /home/kali/Downloads/passpdftruth.txt
```

```
└─$ cat passpdftruth.txt
/home/kali/Downloads/Truth.pdf:$pdf$4*4*128*-1060*1*16*077e10eba516a741a6285385b42f5b27*32*df5071
56115f50098c3d8c6fdb1d662200000000000000000000000000000000*32*7a46addd4179a8ab90812ae8876369522d5
facc72245be4f28b3559473767d57
```

Ubah sedikit *passpdftruth.txt* sampai text diawali dengan **$** agar dapat dilakukan bruteforce menggunakan **hashcat**

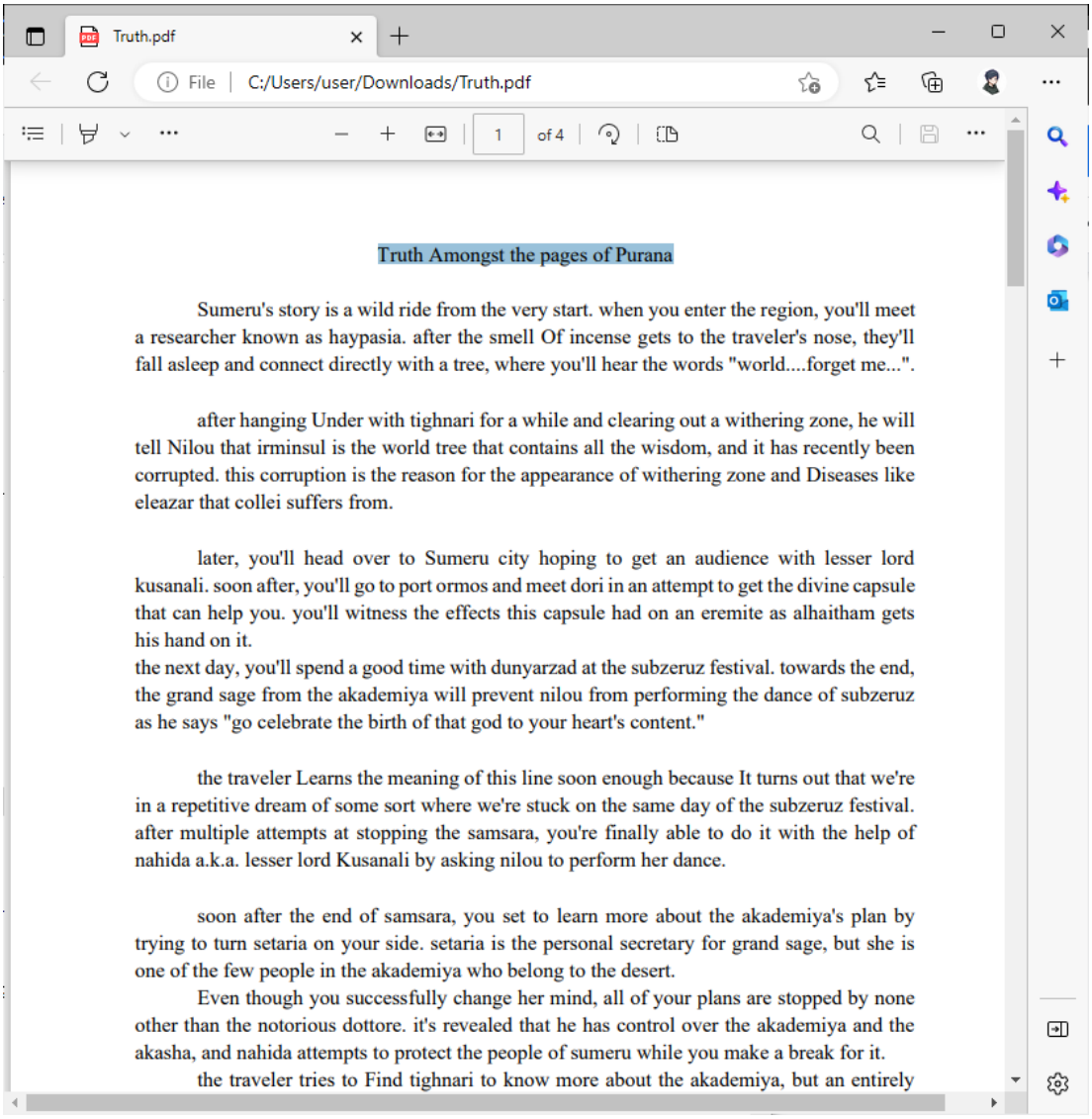


3. Setelah itu lakukan bruteforce dengan tools *hashcat* dan dengan wordlist *rockyou*



*karena tadi sudah dilakukan *bruteforce* maka tampilannya menjadi seperti diatas

Password pdf: **subarukun**



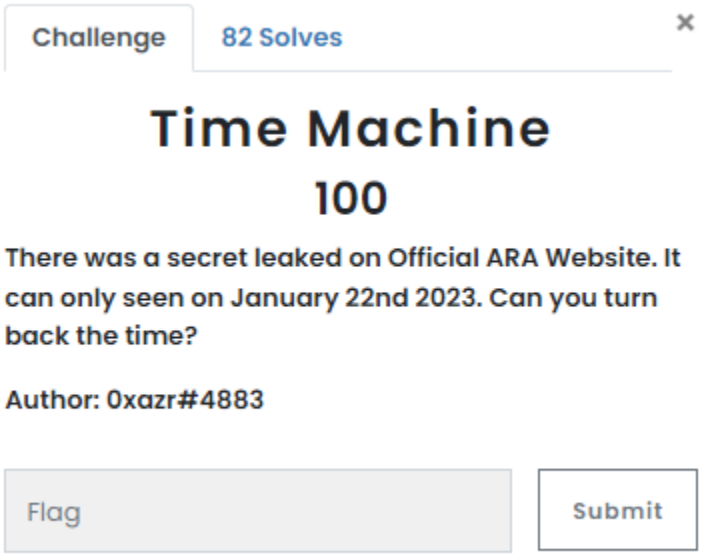
4. Buka pdf, dan sesuai deskripsi soal : **" To understand about yourself,Erase the title and find the Bigger case"** maka disini saya menghiraukan judul dari pdfnya, dan mulai mencari *Uppercase*
5. Didapati hasilnya adalah : SOUNDSLIKEFANDAGO
   FLAG : ARA2023{SOUNDSLIKEFANDAGO}

- Feedback

**Hanya mengisi form feedback*

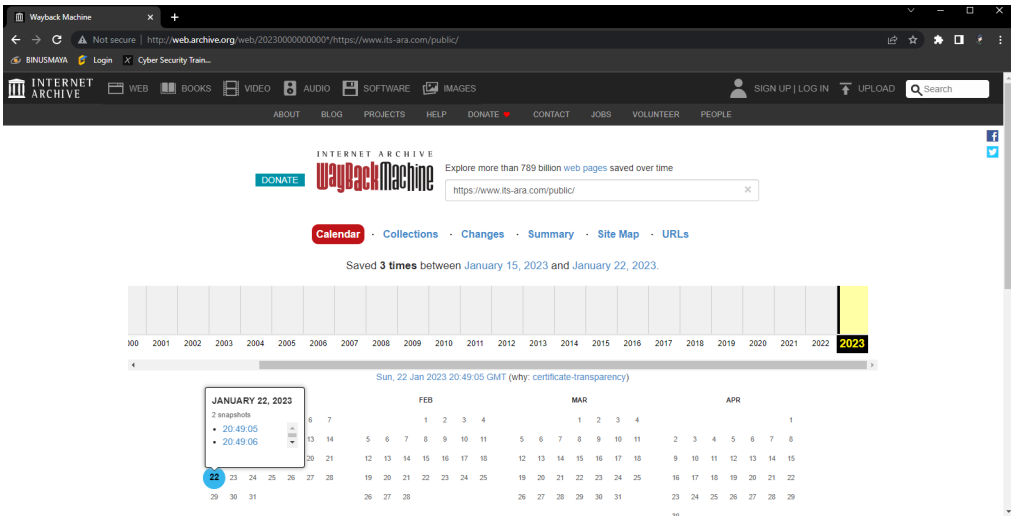
OSINT

- Time Machine

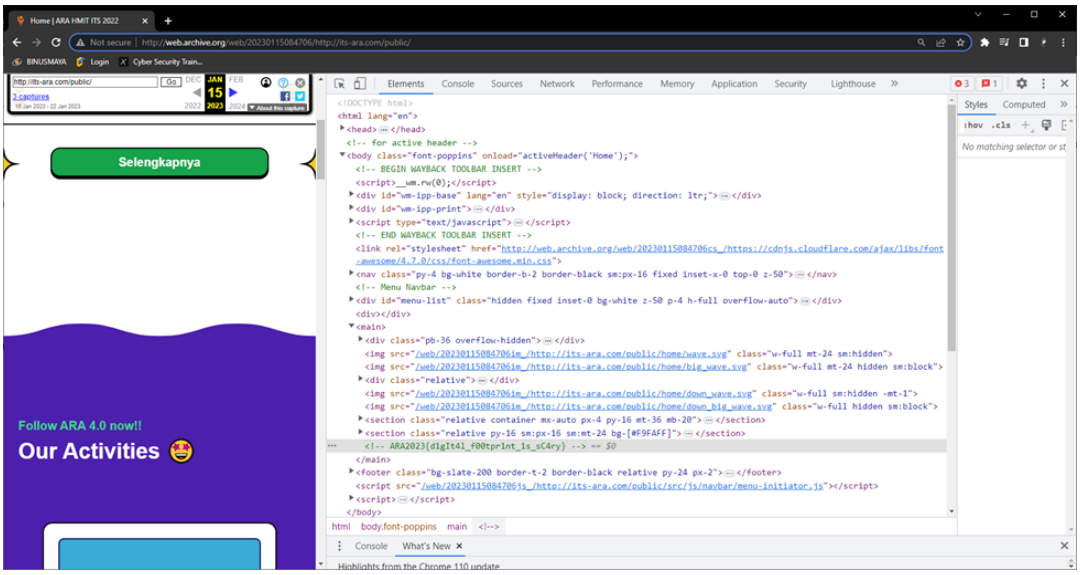


Tahap pengerjaan:

1. Pertama saya menggunakan *wayback machine* http://web.archive.org/ lalu mencari https://www.its-ara.com/public/ pada situsnya



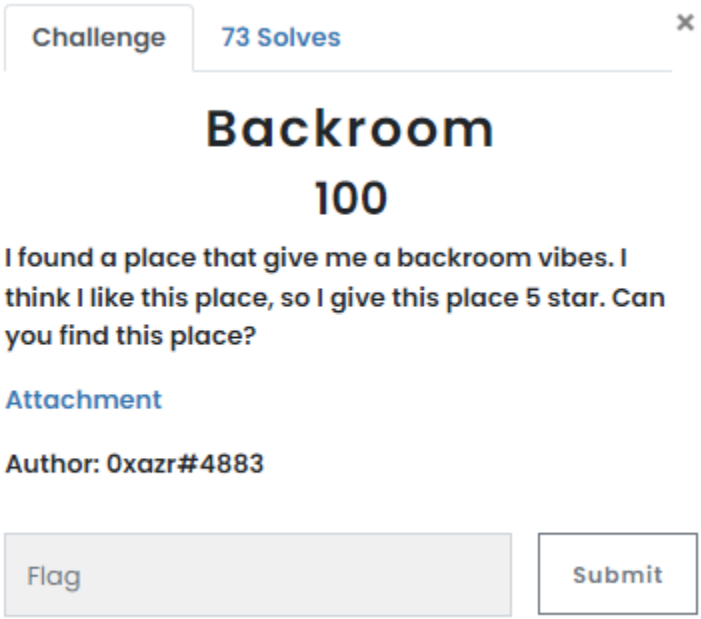
2. Setelah itu saya cari flag pada *inspect element*, dan ditemukan flagnya



<!-- ARA2023{d1glt4l_f00tpr1nt_1s_sC4ry} -->

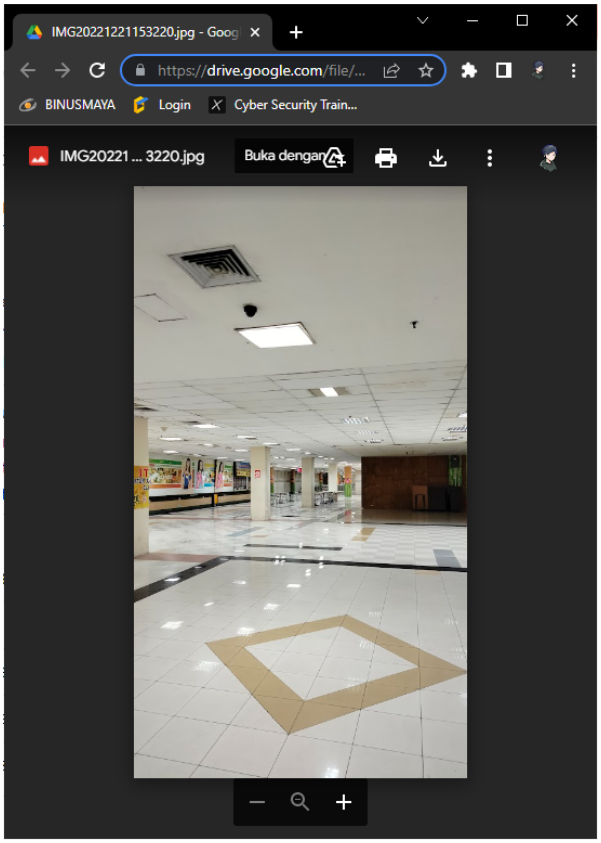
FLAG : ARA2023{d1glt4l_f00tpr1nt_1s_sC4ry}

- Backroom

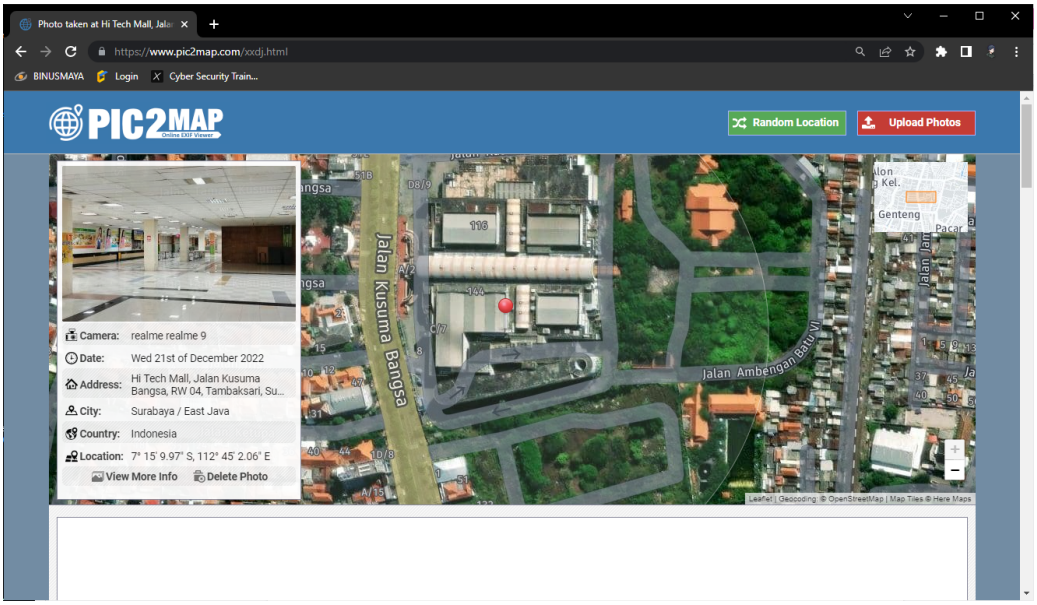


Tahap pengerjaan:

1. Pertama download file pada *attachment* dan didapati file foto sebagai berikut:

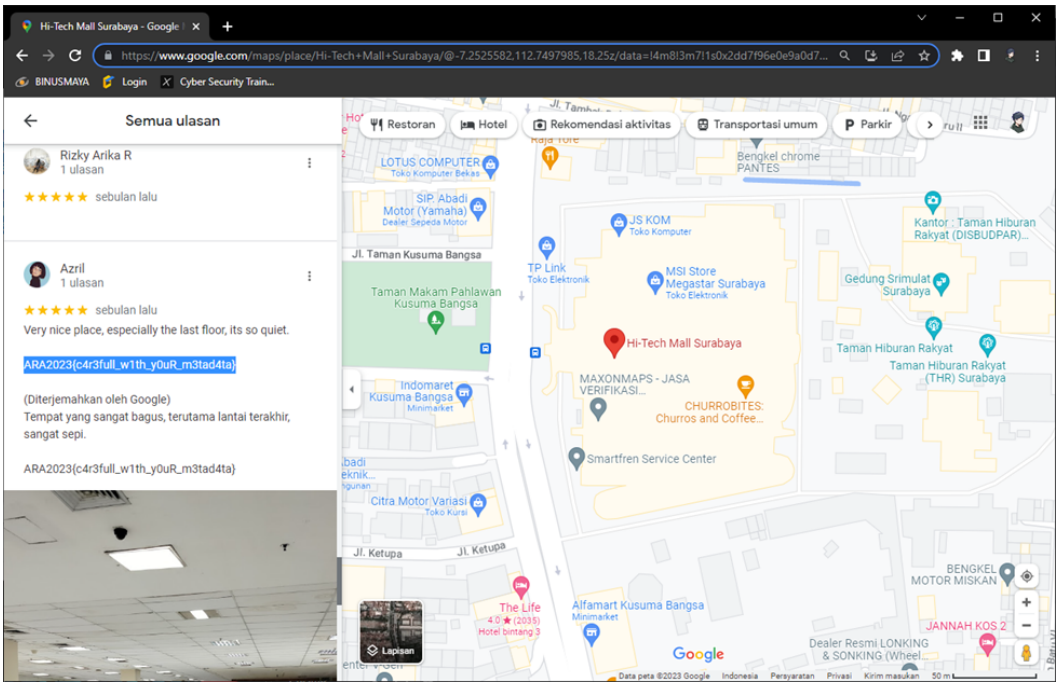


2. Disini dapat diperiksa dimana foto diambil menggunakan *online image locator* seperti https://www.pic2map.com/



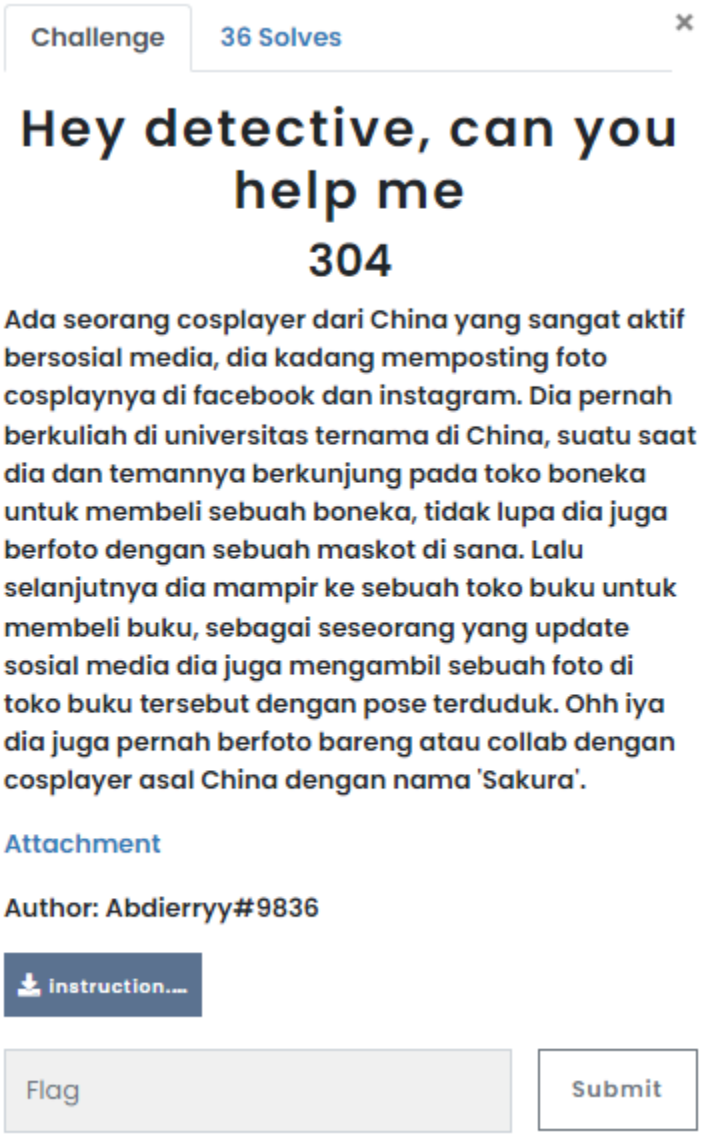
Lokasinya berada di Hi-Tech Mall Surabaya

3. Disini berdasarkan soal, ditemukan bahwa pada lokasi sudah diberi bintang 5, maka saya mencoba untuk melihat satu persatu rating di *google maps,* dan ternyata didapati flag sebagai berikut



FLAG : ARA2023{c4r3full_w1th_y0uR_m3tad4ta}

- Hey detective, can you help me



Tahap pengerjaan:

1. Pertama download file instruction dan didapati ketentuan flag sebagai berikut

```
Flag dibagi dalam 5 bagian :
1. ID Sosmed
2. Nama Universitas dia berkuliah
cukup singkatannya saja, contoh Insititut Teknologi Sepuluh Nopember menjadi ITS
3. Nama maskot
4. Waktu saat upload foto di toko buku
5. Komentar yang terdapat pada saat dia foto bersama Sakura

Format sebagai dibawah :
ARA2023{ProfileIDSosmed_NamaUniversitas_NamaMaskot_TanggalBulanTahun-Jam:Menit_RedactedFlag}
Contoh :
ARA2023{46152324397_UTL_Felda_7Mei2017-13:02_r3d4cTED}
```

2. Disini juga diberikan video pada *attachment*. Disini saya memanfaatkan hint "**pernah berfoto bareng atau collab dengan cosplayer asal China dengan nama 'Sakura'.**" dan dari situ saya mengetahui bahwa ada cosplayer china bernama *sakuragun*

3. Karena saya mengetahui bahwa orang yang bersamanya adalah *Kenko* maka saya melakukan pencarian pada Facebook, Instagram, dan Weiboo. Berikut Data yang saya peroleh:

   Source video yang diberikan pada soal:
   https://www.instagram.com/p/CZQpWSBoGXX/

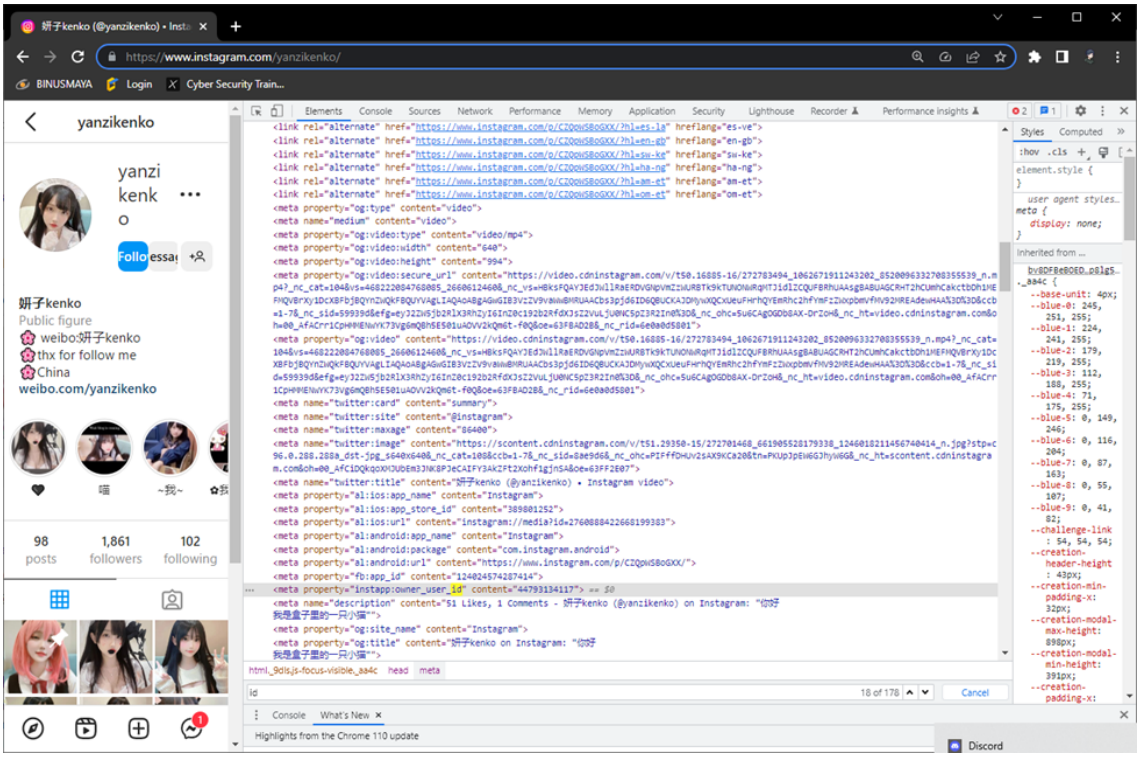
Media sosial:

https://www.instagram.com/yanzikenko/

https://www.weibo.com/yanzikenko

https://www.facebook.com/yanzikenko.hii/photos

4. Dari sini saya mulai cari satu persatu, mulai dari ID yang panjangnya 11 angka, saya mendapati ID ini pada *instagram*



ID : **44793134117**

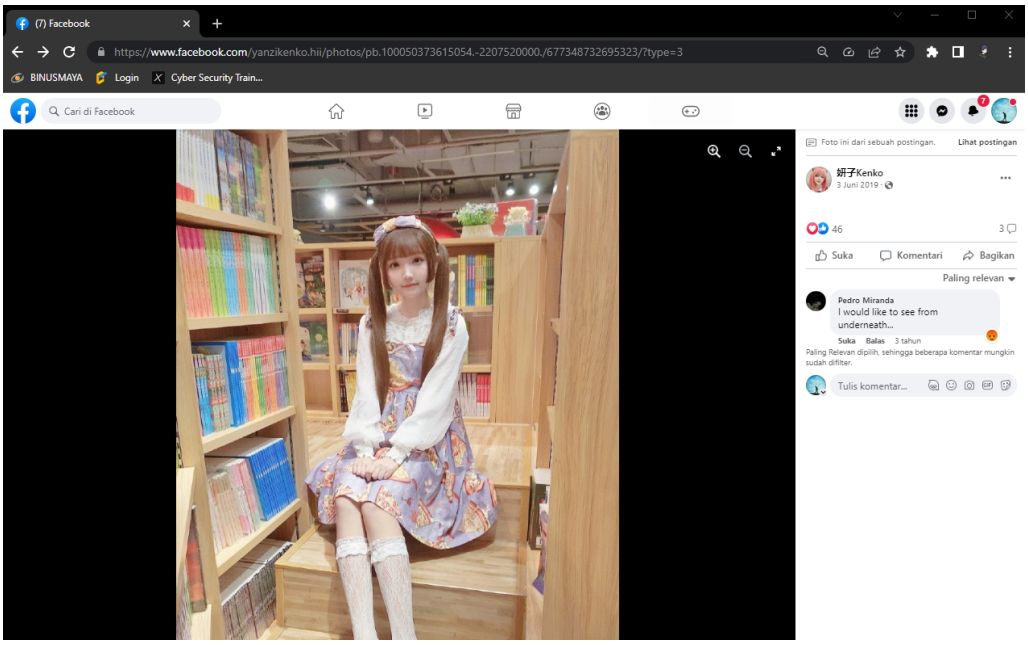
5. Untuk universitas saya dapati melalui facebook pada page : https://www.facebook.com/yanzikenko.hii/photos/pb.100050373615054.-2207520000./981433412286852/?type=3

Universitas : Beijing Normal University = **BNU**

6. Untuk Mascot, saya dapati di facebook pada page : https://www.facebook.com/photo/?fbid=1315564617262 30&set=pcb.131556688392874

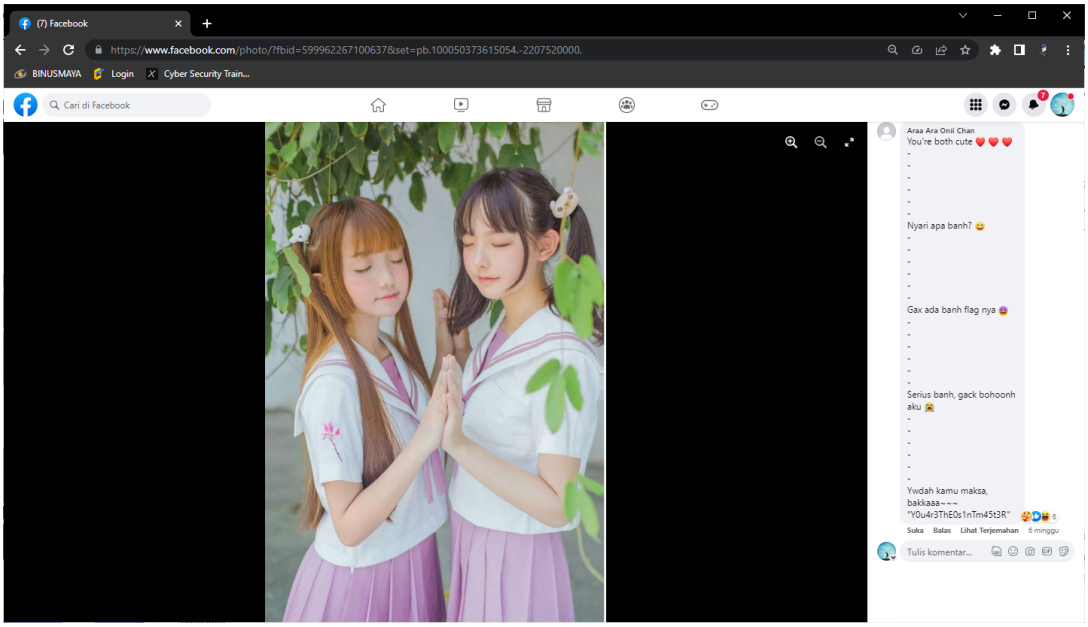
Lalu saya cari mascot ini menggunakan *Yandex*, dan ditemukan bernama **Molly**

7. Untuk tanggal foto duduk di toko buku saya dapati di facebook pada page: https://www.facebook.com/yanzikenko.hii/photos/pb.100050373615054.-2207520000./677348732695323/?type=3



Ditemukan postingan ini pada **3 Juni 2019 10:25**

8. Terakhir ditemukan *hidden flag* nya pada postingan: https://www.facebook.com/photo/?fbid=599962267100637&set=pb.100050373615054.-2207520000.



Hidden flagnya : **Y0u4r3ThE0s1nTm45t3R**

9. Susun flagnya

FLAG :

ARA2023{44793134117_BNU_Molly_3Juni2019-10:25_Y0u4r3ThE0s1nTm45t3R}