



CyberItech Jr

Anggota

Reja Revaldy F

Muhammad Khairin Noor

Ridho Tri Wibowo

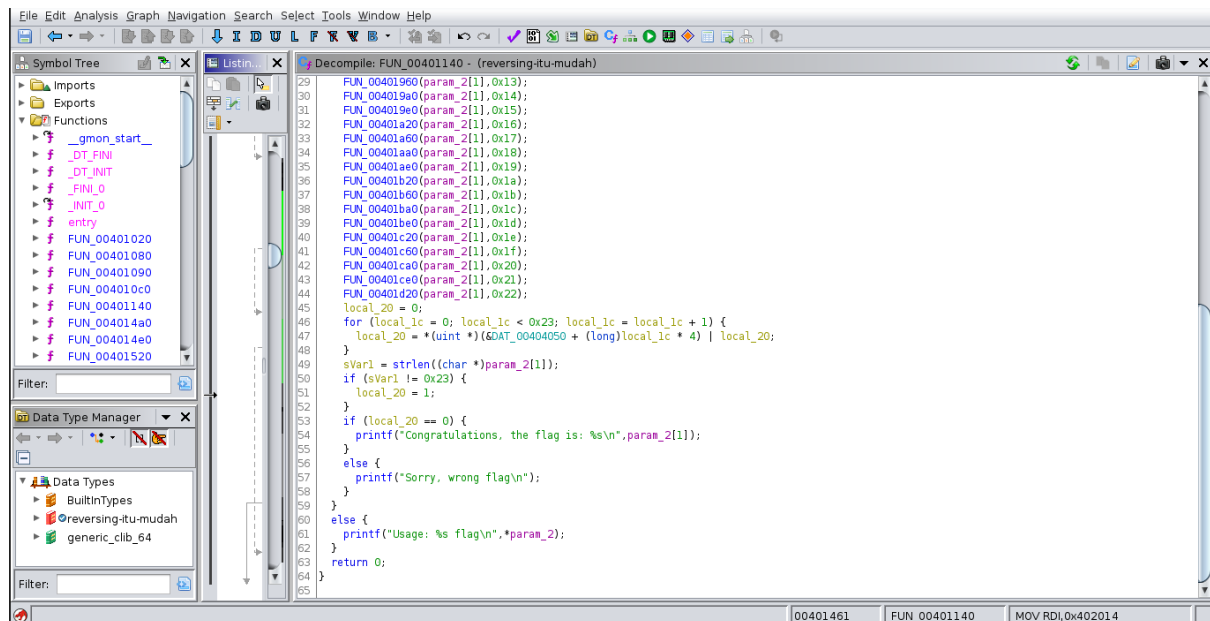
{Reverse Engineering}

Code Juggling

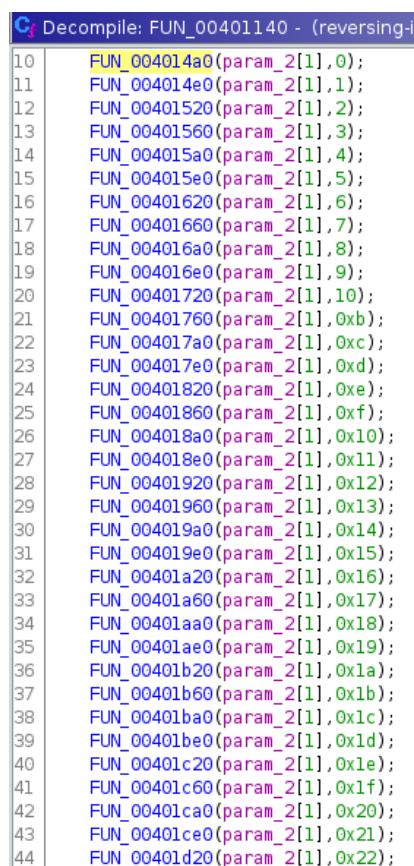


Penyelesaian :

Diberikan sebuah file bertipe ELF, disini setelah saya coba jalankan ternyata ini adalah program pengecekan flag, lalu saya coba menggunakan ghidra untuk melakukan reverse dan saya mencoba untuk melakukan search string lalu mendapatkan codenya



Setelah saya lakukan analisa terhadap kodenya, flag sepertinya disimpan di dalam function dan setelah saya cek function tersebut ternyata benar



```
C:\Decompile: FUN_004014a0 - (reversing-itu-mudah)
1
2 void FUN_004014a0(long param_1,int param_2)
3
4 {
5     *(uint *)&DAT_00404050 + (long)param_2 * 4 = (uint)(*(char *)(param_1 + param_2) != 'G');
6     return;
7 }
8
```

lalu langsung saja saya urutkan dari function pertama sampai terakhir lalu saya coba susun dan saya dapatkan flagnya

```
➔ ./reversing-itu-mudah Gemastik2022{st45iUn_MLG_k07a_b4rU}
Congratulations, the flag is: Gemastik2022{st45iUn_MLG_k07a_b4rU}
```

FLAG : Gemastik2022{st45iUn_MLG_k07a_b4rU}

{Forensic}

Traffic Enjoyer

Challenge

100 Solves


×

Traffic Enjoyer

500

P balap first blood

author - deomkicer#3362

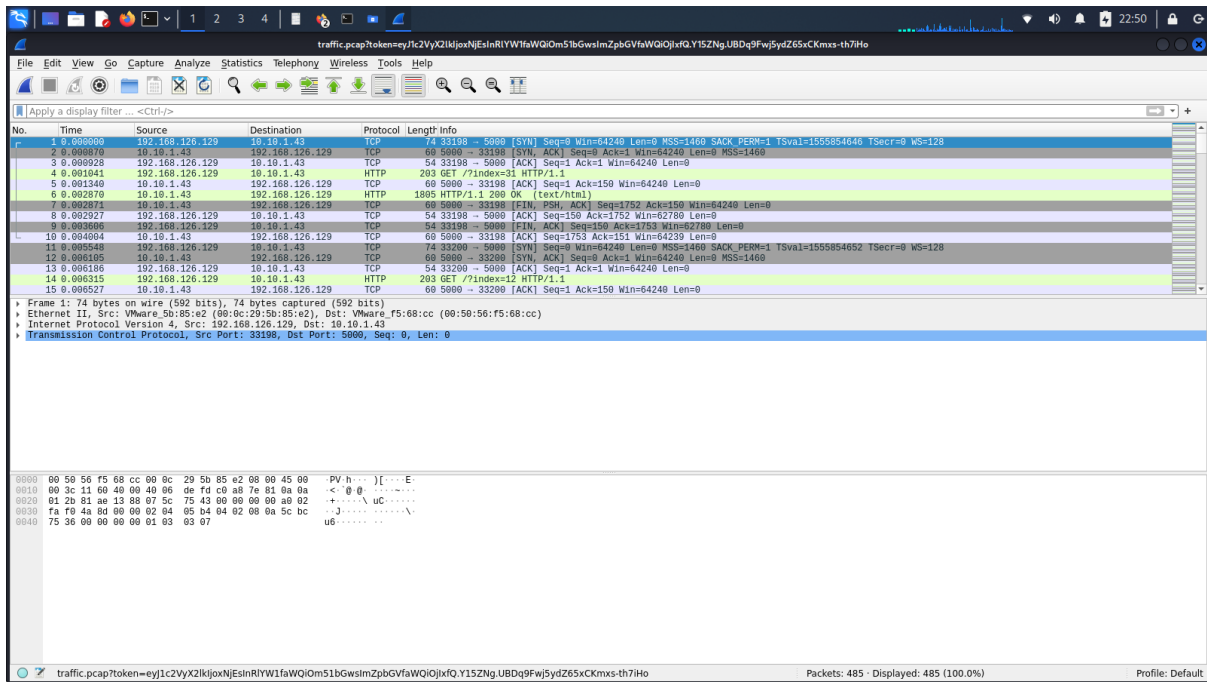
 traffic.pcap

Flag

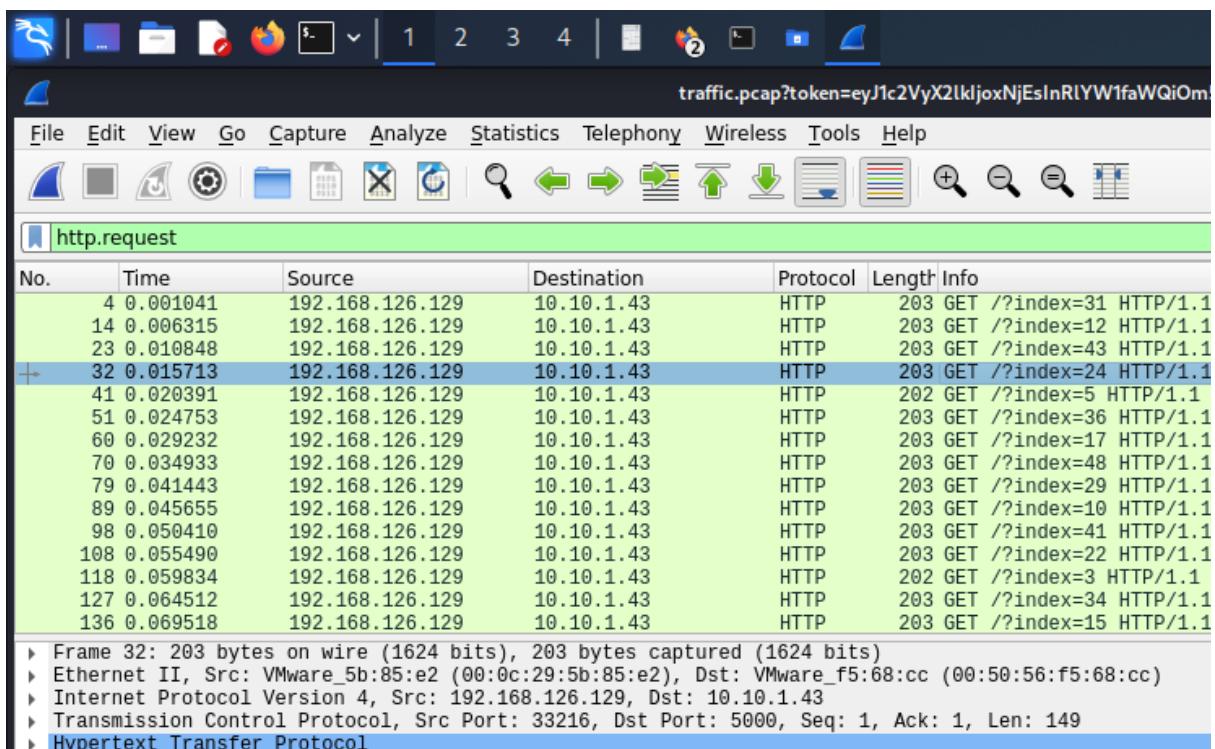
Submit

Penyelesaian :

Diberikan sebuah file bernama "traffic.pcap", karena ekstensi file tersebut adalah ".pcap" yang terlintas dikepala saya adalah wireshark, saya langsung mencoba membukanya menggunakan wireshark



selanjutnya saya mencoba untuk filter displaynya menggunakan "http.request"



setelah itu saya coba untuk follow http streamnya (ctrl+alt+shift+h)

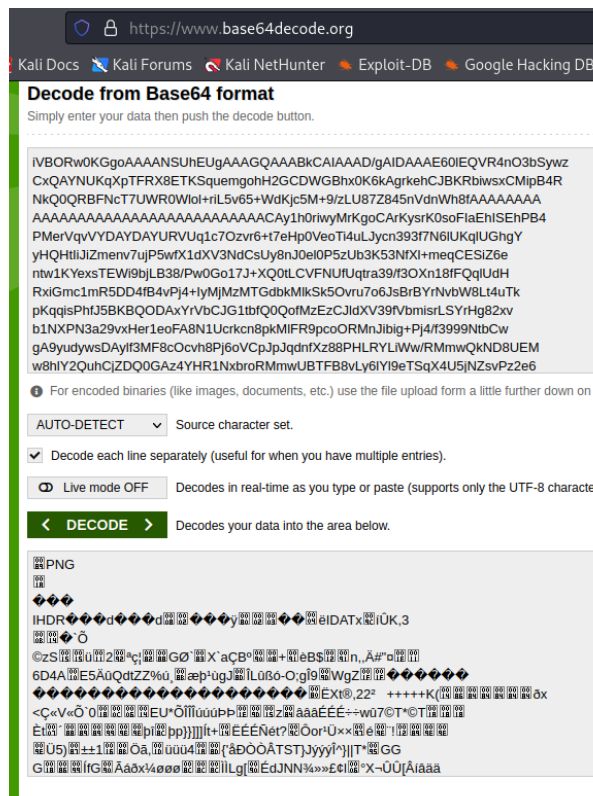
```

GET /?index=24 HTTP/1.1
Host: 10.10.1.43:5000
User-Agent: Gemasteg2022
Accept-Encoding: gzip, deflate, br
Accept: /*/*
Connection: keep-alive

HTTP/1.1 200 OK
Server: Werkzeug/2.2.2 Python/3.8.10
Date: Sun, 30 Oct 2022 04:54:12 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1783
Connection: close

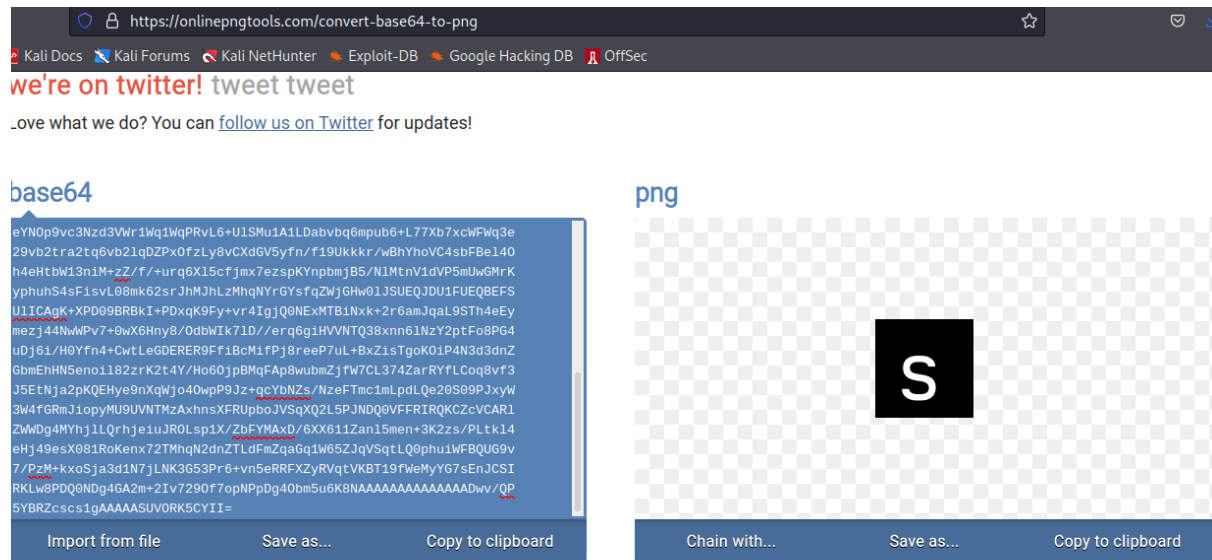
1VB0Rw0KGgoAAAAANSuHEuGAAAGQAAABCAIAAAD/gAIDAAAE60EQVR4n03bSywz
CxQAYNUKqXpFRX8ETKsQuemoghH2GCDWGBhxn0K6kAgrkheCJBKRb1wsxCM1pB4R
K0Q0RBFNCiT7UR0W0L1+r1L5v65wDh3jc5m+9/zL8WZ845vndwnh87AAAAAAA
AAAAAaAAAAAAAAAAAAAAAAAAcAY1h0r1wyMrKgoCArKysRk0soFIaEHIsENPB4
PmerVqvYDAYDAYURVUqkz70zvr6t7eHp0VeoT14uLJycn393f7N61UKqUghgY
yHQHt1JiZmenv7VjL5wPFx1dXV3NdcUsY8nj0ne1P85zUb3K53NFx1+meqCES126e
nWk1KXyxsTEW19bj1B38/Pw0607J1+XQ0tLCVFNuFuqt39/f30Xn18FfQqULd
Rx16mc1mR5DD4fB4Vp14+jYmJ1mTgdabKmkK50vruo6J5sBrBYrNvbW8L4t4Tk
Pqk3sPhf5JBKbQ0DAXrYvCbJgT1bf0Q0fMzEzCJdXV39YvbmIsrLSYrH82xv
b1NXPN3a29vHer1e0FARN2U1crrkc8npkM1FR9pcc0RmNjibg+Pj4/f3999Nt8CW
ga9yudywsDay1f3MF8Cvbn8Pj6oVcPj3QvXZ286PLRYL1lw/RmW0KQND8uW
w8h1Y20uhcJZDQ0GAz4YHR1NxborRmWuBT2FBvly61Y9eTsQX4U5jNZsVpZ2e6
NM8jFosJr7mWdcStLw60g/Dyr607J1+XQ0tLCVFNuFuqt39/f30Xn18FfQqULd
Rxp09pvc3Nz3Vmr1Wq1WqP9j4w+U1SMu1A1LDabvbq6mpub6L77Xb7xcFwQ3e
29vb2tra2qt6vb2lQD2Pxf0ZLybXCGd5yFm/f19Ukkkr/wBhYhoVcn4SfBe140
4dehtbw13ni+m+zZ/f+urqG15xcFjmk7ezspKYnpbmjB5/N1MtnV1dVP5mUwGMkr
yphusS4sFisvL80mk62srJmN1JhLXpRUpb0JvYf5qzGqW1Ghw01JUEQJDU1FUEQBEFS
U1CAgK+XPD09BRBK1+PdxqK9fY+vr4IggJ0XmEXTB1Nxx+2r6mJqal9STh4e4y
meiz44NwPv7+0wGhNhy0Dw0b7k1D/qerq6jHUVNtD30xnm6N1Z2pTf08Pq4
udj61/H0Yfn+CwtLeDERER9F18cM1P7J8repF7+fwZC1StgoK01P4N3d3dnZ
GbmEHNSenoi18zrK2t4Y/Ho06jPqApbAwbwmJzBxL374ZarYfLCoq8r73
J5ETn4zaPQEHye9xnQwJy04owP9Jz+qcYbNZs/NzeFTmc1mLpdLQe20S09pJxyW
3W4f6RmJiopyMU9UVnt2eAxsHvFARUpb0JvYf5qzGqW1Ghw01JUEQJDU1FUEQBEFS
U1CAgK+XPD09BRBK1+PdxqK9fY+vr4IggJ0XmEXTB1Nxx+2r6mJqal9STh4e4y
meiz44NwPv7+0wGhNhy0Dw0b7k1D/qerq6jHUVNtD30xnm6N1Z2pTf08Pq4
udj61/H0Yfn+CwtLeDERER9F18cM1P7J8repF7+fwZC1StgoK01P4N3d3dnZ
GbmEHNSenoi18zrK2t4Y/Ho06jPqApbAwbwmJzBxL374ZarYfLCoq8r73
J5ETn4zaPQEHye9xnQwJy04owP9Jz+qcYbNZs/NzeFTmc1mLpdLQe20S09pJxyW
3W4f6RmJiopyMU9UVnt2eAxsHvFARUpb0JvYf5qzGqW1Ghw01JUEQJDU1FUEQBEFS
U1CAgK+XPD09BRBK1+PdxqK9fY+vr4IggJ0XmEXTB1Nxx+2r6mJqal9STh4e4y
meiz44NwPv7+0wGhNhy0Dw0b7k1D/qerq6jHUVNtD30xnm6N1Z2pTf08Pq4
udj61/H0Yfn+CwtLeDERER9F18cM1P7J8repF7+fwZC1StgoK01P4N3d3dnZ
GbmEHNSenoi18zrK2t4Y/Ho06jPqApbAwbwmJzBxL374ZarYfLCoq8r73
J5ETn4zaPQEHye9xnQwJy04owP9Jz+qcYbNZs/NzeFTmc1mLpdLQe20S09pJxyW
3W4f6RmJiopyMU9UVnt2eAxsHvFARUpb0JvYf5qzGqW1Ghw01JUEQJDU1FUEQBEFS
U1CAgK+XPD09BRBK1+PdxqK9fY+vr4IggJ0XmEXTB1Nxx+2r6mJqal9STh4e4y
meiz44NwPv7+0wGhNhy0Dw0b7k1D/qerq6jHUVNtD30xnm6N1Z2pTf08Pq4
udj61/H0Yfn+CwtLeDERER9F18cM1P7J8repF7+fwZC1StgoK01P4N3d3dnZ
GbmEHNSenoi18zrK2t4Y/Ho06jPqApbAwbwmJzBxL374ZarYfLCoq8r73
J5ETn4zaPQEHye9xnQwJy04owP9Jz+qcYbNZs/NzeFTmc1mLpdLQe20S09pJxyW
3W4f6RmJiopyMU9UVnt2eAxsHvFARUpb0JvYf5qzGqW1Ghw01JUEQJDU1FUEQBEFS
U1CAgK+XPD09BRBK1+PdxqK9fY+vr4IggJ0XmEXTB1Nxx+2r6mJqal9STh4e4y
meiz44NwPv7+0wGhNhy0Dw0b7k1D/qerq6jHUVNtD30xnm6N1Z2pTf08Pq4
udj61/H0Yfn+CwtLeDERER9F18cM1P7J8repF7+fwZC1StgoK01P4N3d3dnZ
GbmEHNSenoi18zrK2t4Y/Ho06jPqApbAwbwmJzBxL374ZarYfLCoq8r73
J5ETn4zaPQEHye9xnQwJy04owP9Jz+qcYbNZs/NzeFTmc1mLpdLQe20S09pJxyW
3W4f6RmJiopyMU9UVnt2eAxsHvFARUpb0JvYf5qzGqW1Ghw01JUEQJDU1FUEQBEFS
U1CAgK+XPD09BRBK1+PdxqK9fY+vr4IggJ0XmEXTB1Nxx+2r6mJqal9STh4e4y
meiz44NwPv7+0wGhNhy0Dw0b7k1D/qerq6jHUVNtD30xnm6N1Z2pTf08Pq4
udj61/H0Yfn+CwtLeDERER9F18cM1P7J8repF7+fwZC1StgoK01P4N3d3dnZ
GbmEHNSenoi18zrK2t4Y/Ho06jPqApbAwbwmJzBxL374ZarYfLCoq8r73
J5ETn4zaPQEHye9xnQwJy04owP9Jz+qcYbNZs/NzeFTmc1mLpdLQe20S09pJxyW
3W4f6RmJiopyMU9UVnt2eAxsHvFARUpb0JvYf5qzGqW1Ghw01JUEQJDU1FUEQBEFS
U1CAgK+XPD09BRBK1+PdxqK9fY+vr4IggJ0XmEXTB1Nxx+2r6mJqal9STh4e4y
meiz44NwPv7+0wGhNhy0Dw0b7k1D/qerq6jHUVNtD30xnm6N1Z2pTf08Pq4
udj61/H0Yfn+CwtLeDERER9F18cM1P7J8repF7+fwZC1StgoK01P4N3d3dnZ
GbmEHNSenoi18zrK2t4Y/Ho06jPqApbAwbwmJzBxL374ZarYfLCoq8r73
J5ETn4zaPQEHye9xnQwJy04owP9Jz+qcYbNZs/NzeFTmc1mLpdLQe20S09pJxyW
3W4f6RmJiopyMU9UVnt2eAxsHvFARUpb0JvYf5qzGqW1Ghw01JUEQJDU1FUEQBEFS
U1CAgK+XPD09BRBK1+PdxqK9fY+vr4IggJ0XmEXTB1Nxx+2r6mJqal9STh4e4y
meiz44NwPv7+0wGhNhy0Dw0b7k1D/qerq6jHUVNtD30xnm6N1Z2pTf08Pq4
udj61/H0Yfn+CwtLeDERER9F18cM1P7J8repF7+fwZC1StgoK01P4N3d3dnZ
GbmEHNSenoi18zrK2t4Y/Ho06jPqApbAwbwmJzBxL374ZarYfLCoq8r73
J5ETn4zaPQEHye9xnQwJy04owP9Jz+qcYbNZs/NzeFTmc1mLpdLQe20S09pJxyW
3W4f6RmJiopyMU9UVnt2eAxsHvFARUpb0JvYf5qzGqW1Ghw01JUEQJDU1FUEQBEFS
U1CAgK+XPD09BRBK1+PdxqK9fY+vr4IggJ0XmEXTB1Nxx+2r6mJqal9STh4e4y
meiz44NwPv7+0wGhNhy0Dw0b7k1D/qerq6jHUVNtD30xnm6N1Z2pTf08Pq4
udj61/H0Yfn+CwtLeDERER9F18cM1P7J8repF7+fwZC1StgoK01P4N3d3dnZ
GbmEHNSenoi18zrK2t4Y/Ho06jPqApbAwbwmJzBxL374ZarYfLCoq8r73
J5ETn4zaPQEHye9xnQwJy04owP9Jz+qcYbNZs/NzeFTmc1mLpdLQe20S09pJxyW
3W4f6RmJiopyMU9UVnt2eAxsHvFARUpb0JvYf5qzGqW1Ghw01JUEQJDU1FUEQBEFS
U1CAgK+XPD09BRBK1+PdxqK9fY+vr4IggJ0XmEXTB1Nxx+2r6mJqal9STh4e4y
meiz44NwPv7+0wGhNhy0Dw0b7k1D/qerq6jHUVNtD30xnm6N1Z2pTf08Pq4
udj61/H0Yfn+CwtLeDERER9F18cM1P7J8repF7+fwZC1StgoK01P4N3d3dnZ
GbmEHNSenoi18zrK2t4Y/Ho06jPqApbAwbwmJzBxL374ZarYfLCoq8r73
J5ETn4zaPQEHye9xnQwJy04owP9Jz+qcYbNZs/NzeFTmc1mLpdLQe20S09pJxyW
3W4f6RmJiopyMU9UVnt2eAxsHvFARUpb0JvYf5qzGqW1Ghw01JUEQJDU1FUEQBEFS
U1CAgK+XPD09BRBK1+PdxqK9fY+vr4IggJ0XmEXTB1Nxx+2r6mJqal9STh4e4y
meiz44NwPv7+0wGhNhy0Dw0b7k1D/qerq6jHUVNtD30xnm6N1Z2pTf08Pq4
udj61/H0Yfn+CwtLeDERER9F18cM1P7J8repF7+fwZC
```

terdapat sebuah text, saya menduga text tersebut telah di encode menggunakan base64, saya langsung mencoba untuk decode di website "<https://www.base64decode.org/>"

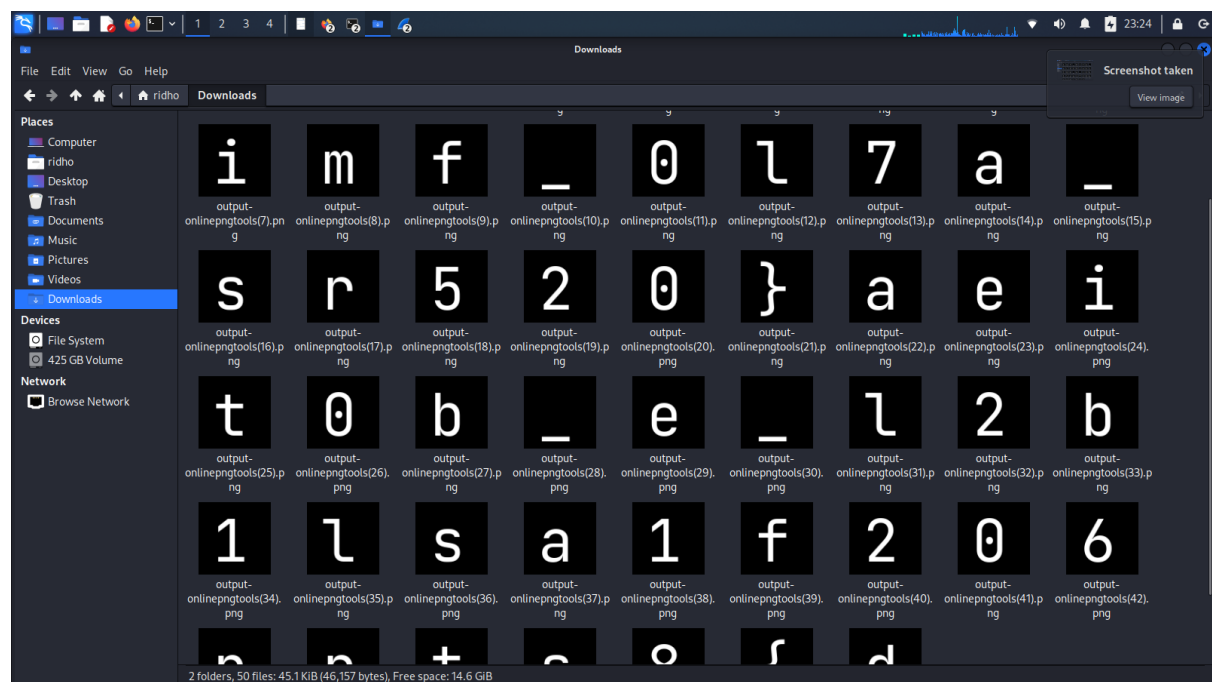


dan ternyata benar, setelah teks tersebut didecode, terdapat header untuk file png, saya langsung mencoba untuk mengconvert teks base64 tadi ke png di website

“<https://onlinepngtools.com/convert-base64-to-png>”



setelah diconvert saya mendapatkan sebuah huruf pada gambar, saya menduga huruf tersebut adalah potongan flagnya, saya langsung mengconvert semua teks base64 yang ada pada list http.request tadi ke png di website yang sama



setelah saya mengumpulkan semua teks gambar tadi, ternyata teksnya masih acak dan harus disusun terlebih dahulu, setelah saya cek lebih teliti ternyata terdapat urutan index pada list http.request tadi

length	Info
202	GET /?index=0 HTTP/1.1
202	GET /?index=1 HTTP/1.1
203	GET /?index=10 HTTP/1.1
203	GET /?index=11 HTTP/1.1
203	GET /?index=12 HTTP/1.1
203	GET /?index=13 HTTP/1.1
203	GET /?index=14 HTTP/1.1
203	GET /?index=15 HTTP/1.1
203	GET /?index=16 HTTP/1.1
203	GET /?index=17 HTTP/1.1
203	GET /?index=18 HTTP/1.1
203	GET /?index=19 HTTP/1.1
202	GET /?index=2 HTTP/1.1
203	GET /?index=20 HTTP/1.1
203	GET /?index=21 HTTP/1.1

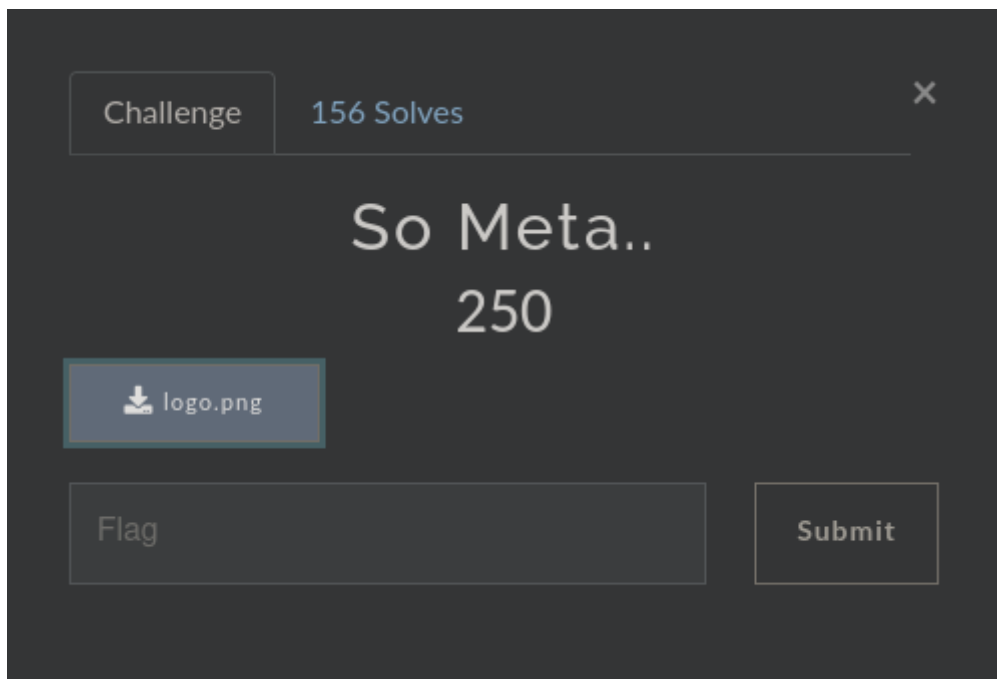
saya langsung menyusun gambar teks tadi berdasarkan urutan index yang ada (0-49) dan saya mendapatkan susunan flagnya

FLAG :

Gemastik2022{balapan_f1rst_b100d_is_real_f580c176}

{ Forensic }

So Meta

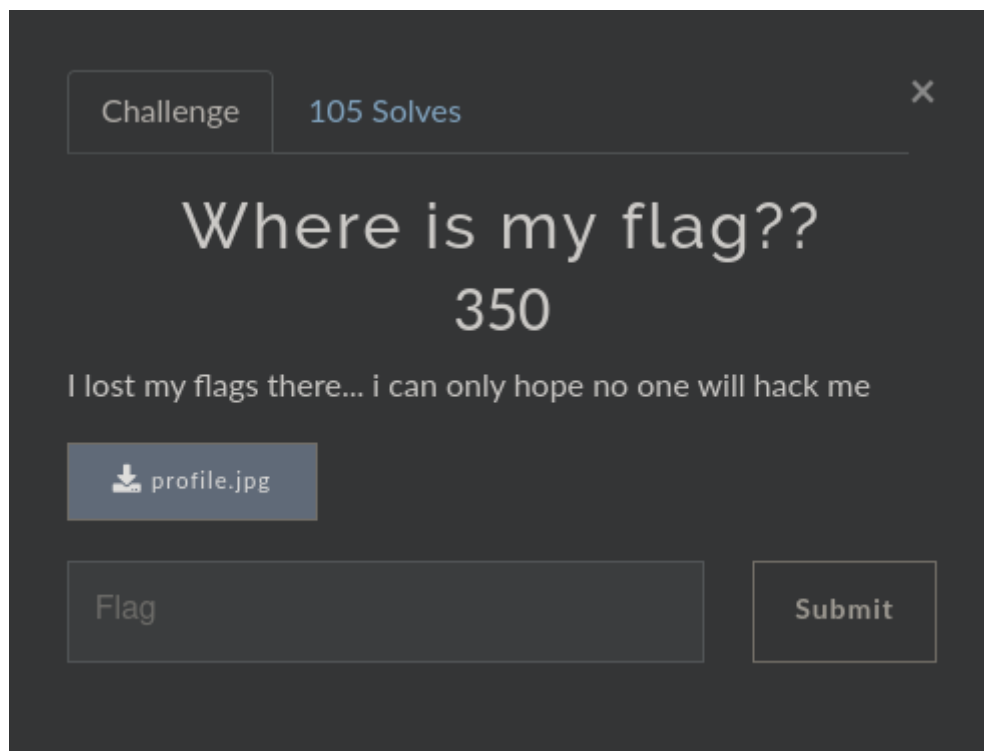


Diberikan sebuah file png, dan langsung saja saya coba menggunakan exiftool untuk melakukan pengecekan terhadap meta datanya dan saya langsung menemukan flagnya

```
[EVA-01] as revv in ~/Cybersecurity/ctf/gemastik2022/forensic/meta
→ exiftool logo.png
ExifTool Version Number      : 12.16
File Name                    : logo.png
Directory                    : .
File Size                    : 33 KiB
File Modification Date/Time   : 2022:10:23 16:20:59+08:00
File Access Date/Time        : 2022:10:23 16:20:58+08:00
File Inode Change Date/Time   : 2022:10:23 16:21:11+08:00
File Permissions              : rw-r--r--
File Type                    : PNG
File Type Extension          : png
MIME Type                    : image/png
Image Width                  : 510
Image Height                  : 60
Bit Depth                    : 8
Color Type                   : RGB with Alpha
Compression                  : Deflate/Inflate
Filter                       : Adaptive
Interlace                    : Noninterlaced
Exif Byte Order              : Big-endian (Motorola, MM)
X Resolution                  : 72
Y Resolution                  : 72
Resolution Unit               : inches
Y Cb Cr Positioning          : Centered
Camera Serial Number         : Gemastik2022{n4s1_J4g03nG}
Image Size                   : 510x60
Megapixels                   : 0.031
```

FLAG : Gemastik2022{n4s1_J4g03nG}

Where is my flag??

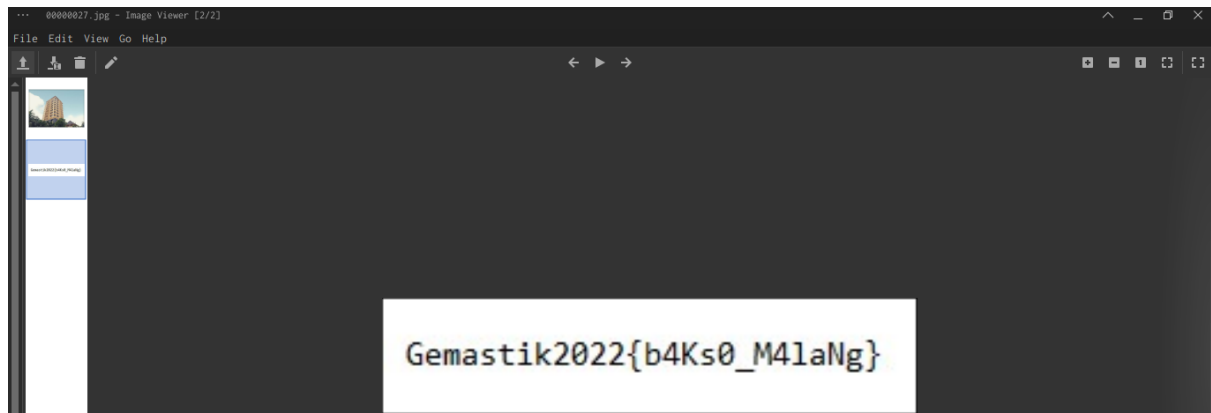


Diberikan sebuah file jpg dan kita disuruh untuk mencari flagnya, disini saya mencoba menggunakan binwalk untuk melakukan pengecekan apakah terdapat file lagi di dalam file jpg tersebut, setelah saya coba eksekusi ternyata terdapat 2 file didalam file tersebut.

```
[EVA-01] as revv in ~/Cybersecurity/ctf/gemastik2022/forensic/where my flag
> binwalk profile.jpg
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
14153	0x3749	JPEG image data, JFIF standard 1.01

Langsung saja saya extract dan membuka file tersebut dan menemukan flagnya



FLAG = Gemastik2022{b4Ks0_M41aNg}

{ Kriptografi }

Thanks God, it's easy

Challenge

103 Solves

×

Thanks God, it's easy
100

VUC29CWU{m4354b_m1zr3b_p0b_dr3_GsX}

Flag

Submit

terdapat sebuah format flag yang telah diencrypt pada deskripsi, disini kita harus mencari ciphernya, dan jika dilihat dari polanya saya menduga flag tersebut diencrypt menggunakan antara ROT13 / Caesar Cipher, saya langsung mencoba untuk mendecryptnya di web "<https://www.dcode.fr/caesar-cipher>" dan ternyata benar, saya mendapatkan flagnya menggunakan Caesar Cipher



FLAG : LKS29SMK{c4354r_c1ph3r_f0r_th3_WiN}

x0r



Diberikan sebuah file python dan file yang berisikan cipher text hasil dari encrypt file python tersebut, setelah saya membuka file python tersebut saya mengidentifikasikan bahwa

cipher text tersebut di enkripsi menggunakan xor dengan key "n0k3y" saya langsung saja mencoba untuk buat script python menggunakan informasi yang sudah ada

```
def xor_crypt_string(data, key = 'n0k3y', encode = False, decode = False):  
    from itertools import izip, cycle  
    import base64  
  
    if decode:  
        data = base64.decodestring(data)  
        xored = ''.join(chr(ord(x) ^ ord(y)) for (x,y) in izip(data, cycle(key)))  
  
    if encode:  
        return base64.encodestring(xored).strip()  
    return xored  
secret_data = "XOR procedure"  
  
print("The cipher text is")  
print xor_crypt_string(secret_data, encode = True)  
print("The plain text fetched")  
print xor_crypt_string("KVUGUgoaWQABSVwCEEo2G28PABocSRtHJgMDFg==", decode = True)
```

FLAG = Gemastik2022{y0u_d3crypt_m3}

Encoder-Decoder



Diberikan sebuah file python dan file yang berisikan cipher text hasil dari encryption file python tersebut,

```
def encrypt(plaintext):
    plaintext = plaintext[::-1]
    ciphertext = ""

    for i in plaintext:
        copy = "X" * ((ord(i) ^ 0x50) + 9)
        copy += "-"
        ciphertext += copy

    return ciphertext
```

dari function tersebut plain text di encrypt menggunakan xor dengan kunci "0x50" lalu hasil dari encrypt tersebut di konversikan menjadi string "X" sesuai dari encryption xor tersebut lalu ditambah 9 dan dihubungkan dengan "-", setelah itu saya mencoba untuk buat function untuk mengonversikan string "X" tersebut ke bentuk encrypt asalnya

Output : [23, 53, 61, 49, 35, 36, 57, 59, 98, 96, 98, 98, 43, 36, 24, 49, 62, 27, 35, 15, 54, 96, 34, 15, 54, 97, 62, 52, 97, 62, 55, 15, 61, 99, 45]

setelah saya mendapatkan bilangan hasil encryption nya maka saya mencoba untuk melakukan decrypt dan menemukan flagnya

```
def encrypt(plaintext):
    plaintext = plaintext[::-1]
    ciphertext = ""

    for i in plaintext:
        copy = "X" * ((ord(i) ^ 0x50) + 9)
        copy += "-"
        ciphertext += copy

    return ciphertext

def decrypt(crypt):
    crypt = crypt[::-1]
    flag = crypt.split("-")
    hex_decode = []
    for x in flag:
        number = x.count("X") - 9
        if number > 0:
            hex_decode.append(number)

    return hex_decode

def main():
    cipher_file = open("encrypted.txt", "r")
    cipher_text = cipher_file.readlines()[0]
    hex_decode = decrypt(cipher_text)
    flag = ""

    for x in hex_decode:
        flag += chr(x ^ 0x50)

    return flag

if __name__ == "__main__":
    print(main())
}
```

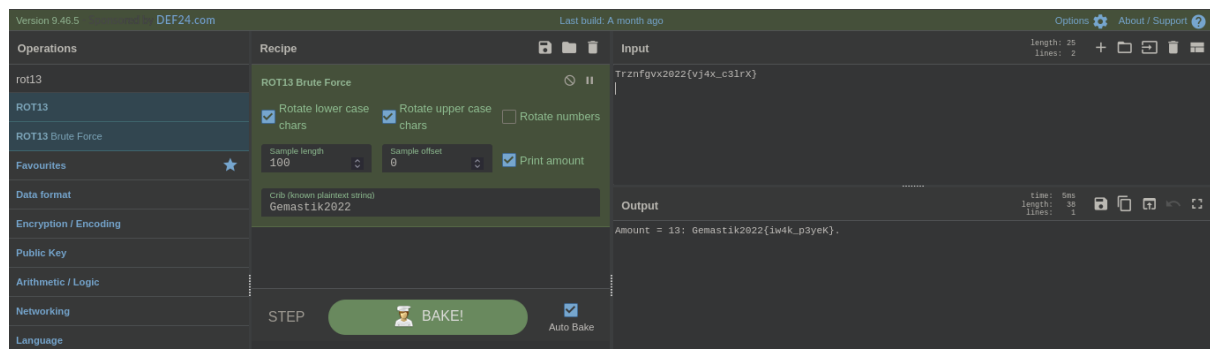
FLAG = Gemastik2022{tHanKs_f0r_f1nd1ng_m3}

Heavy Rotation 13



Diberikan sebuah text yang sepertinya di encrypt menggunakan Rot 13, disini saya mencoba untuk melakukan decrypt di website

“<https://cyberchef.org/>” dan format flagnya adalah Gemastik2022{}



FLAG = Gemastik2022{iw4k_p3yeK}

{ Web }

Duh!

Challenge

55 Solves

×

Duh!

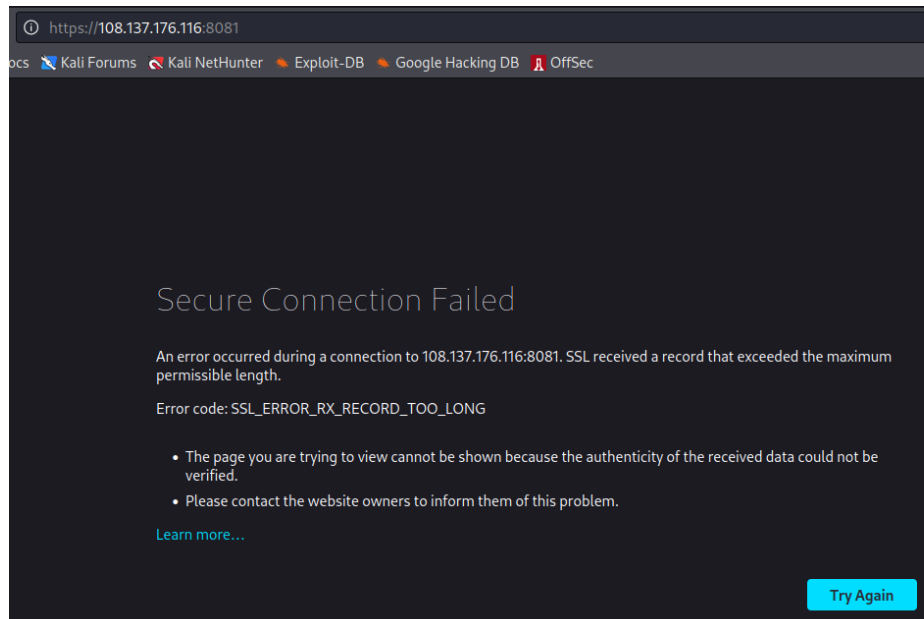
150

108.137.176.116:[8081-8083]

Flag

Submit

Diberikan sebuah alamat IP dengan port 8081, 8082, 8083, awalnya saya mencoba untuk mengakses ke semua port dan mendapati bahwa port 8081 tidak dapat diakses jadi saya fokus mencari flag di port lainnya



selanjutnya saya terpikir untuk menggunakan dirb pada port 8082 dan 8083, dengan wordlist common.txt saya mencoba untuk melakukan brute force untuk mendapatkan directory/subdirectory yang ada pada kedua port tersebut

command : `dirb http://108.137.176.116:[port]/`
`/usr/share/dirb/wordlists/common.txt`

```
File Actions Edit View Help
(root@kali)~[/home/ridho]
# dirb http://108.137.176.116:8082/ /usr/share/dirb/wordlists/common.txt

DIRB v2.22
By The Dark Raver

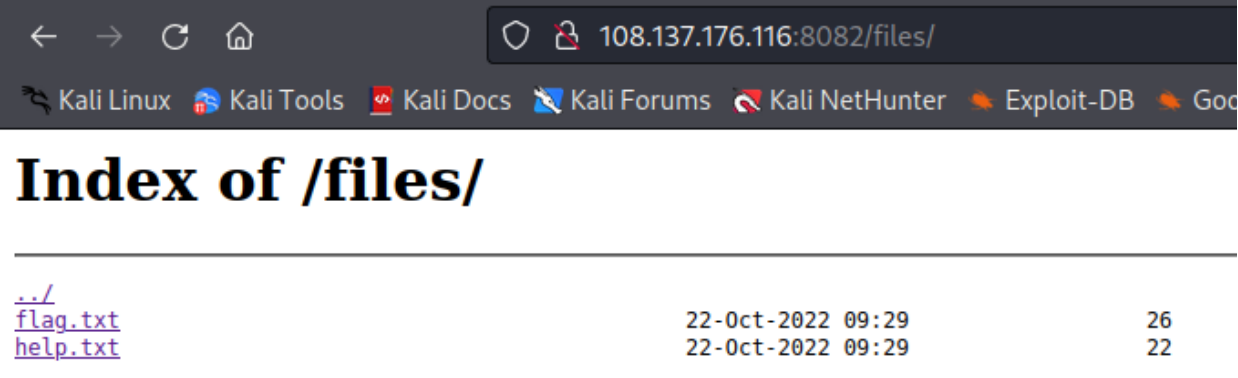
START_TIME: Mon Oct 24 20:12:27 2022
URL_BASE: http://108.137.176.116:8082/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612

— Scanning URL: http://108.137.176.116:8082/ —
=> DIRECTORY: http://108.137.176.116:8082/files/
+ http://108.137.176.116:8082/index.html (CODE:200|SIZE:521)
=> DIRECTORY: http://108.137.176.116:8082/static/

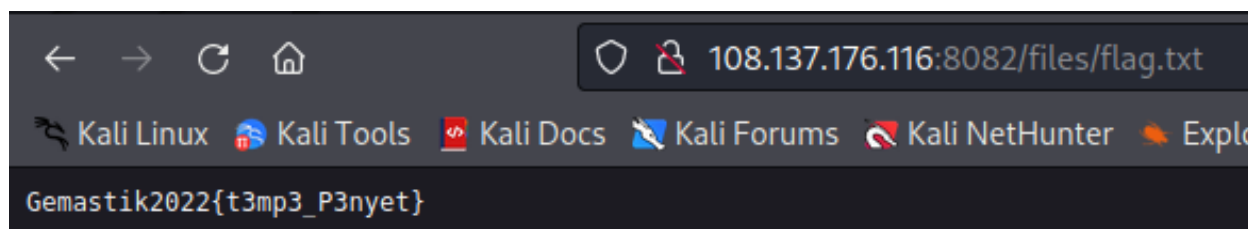
— Entering directory: http://108.137.176.116:8082/files/ —
— Entering directory: http://108.137.176.116:8082/static/ —
```

hasilnya terdapat 2 directory pada port 8082 yaitu /files dan /static, dan saya menemukan flagnya pada "<http://108.137.176.116:8082/files/flag.txt>"



The screenshot shows a web browser window with the address bar displaying '108.137.176.116:8082/files/'. Below the address bar is a navigation bar with links to 'Kali Linux', 'Kali Tools', 'Kali Docs', 'Kali Forums', 'Kali NetHunter', 'Exploit-DB', and 'Google'. The main content area displays the title 'Index of /files/' followed by a table listing files in the directory.

../		
flag.txt	22-Oct-2022 09:29	26
help.txt	22-Oct-2022 09:29	22



FLAG : Gemastik2022{t3mp3_P3nyet}

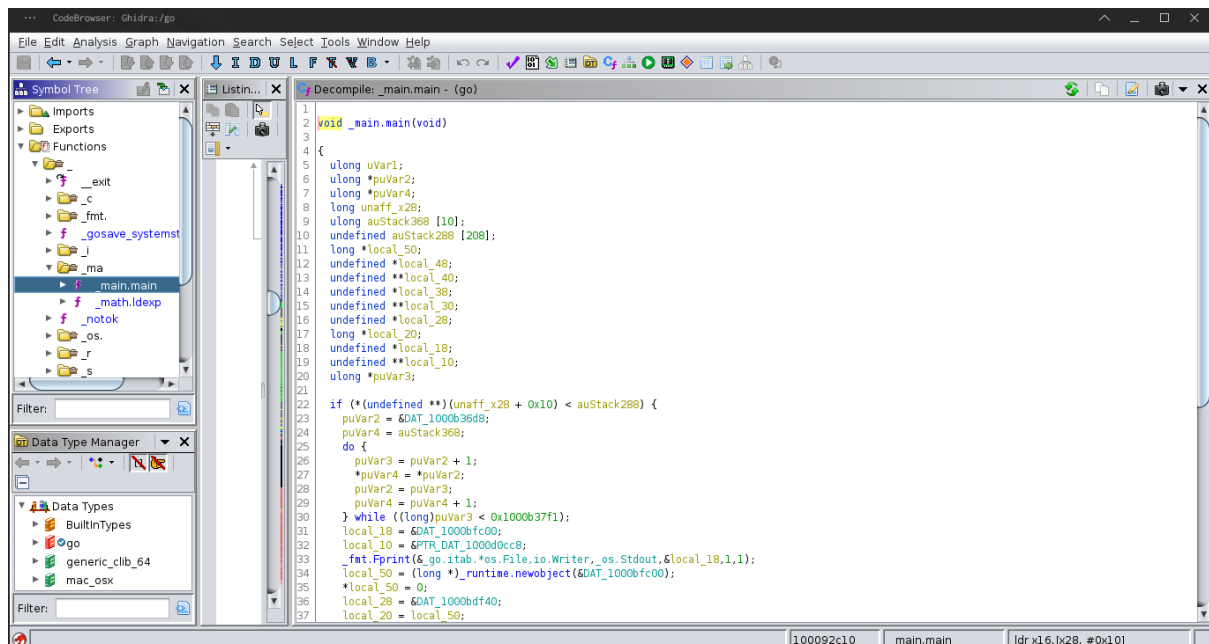
{ Reverse }

ReverseMe

Diberikan sebuah file yang setelah saya cek tipe filenya, file tersebut dibuat menggunakan golang

```
➤ file go
go: Mach-O 64-bit arm64 executable, flags:<|DYLDLINK|PIE>
```

disini saya coba beberapa cara terlebih dahulu seperti mengecek strings dan cat file tersebut dan saya tidak mendapatkan apa apa, setelah itu saya mencoba menggunakan ghidra untuk melakukan reverse dan melakukan pengecekan di function main



setelah saya coba analisa dan melakukan pengecekan terhadap variabel dan kode tersebut saya mencoba untuk mengecek

"0x1000b37f1" di bagian while dan menemukan sebuah potongan flag di bagian listingnya

```
Listing: go
1000b37df 00    ??    00h
1000b37e0 6e    ??    6Eh  n
1000b37e1 00    ??    00h
1000b37e2 00    ??    00h
1000b37e3 00    ??    00h
1000b37e4 00    ??    00h
1000b37e5 00    ??    00h
1000b37e6 00    ??    00h
1000b37e7 00    ??    00h
1000b37e8 47    ??    47h  G
1000b37e9 00    ??    00h
1000b37ea 00    ??    00h
1000b37eb 00    ??    00h
1000b37ec 00    ??    00h
1000b37ed 00    ??    00h
1000b37ee 00    ??    00h
1000b37ef 00    ??    00h
1000b37f0 7d    ??    7Dh  }
1000b37f1 00    ??    00h
1000b37f2 00    ??    00h
1000b37f3 00    ??    00h
1000b37f4 00    ??    00h
1000b37f5 00    ??    00h
1000b37f6 00    ??    00h
1000b37f7 00    ??    00h
1000b37f8 00    ??    00h
1000b37f9 00    ??    00h
1000b37fa 00    ??    00h
1000b37fb 00    ??    00h
1000b37fc 00    ??    00h
1000b37fd 00    ??    00h
1000b37fe 00    ??    00h
1000b37ff 00    ??    00h

_runtime.findfunctab                                XREF[3]:  _runtime.findfunc:100049120(*),
                                                    _runtime.findfunc:100049130(R),
```

setelah saya susun saya mendapatkan sebuah flagnya yaitu

FLAG : Gemastik2022{b3l4j4r_b4r5AmA_g0l4nG}