

**WRITEUP OPEN RECRUITMENT
DIVISI SECURITY POROS
2020**



POROS

OLEH: ALDI FIANDA PUTRA
TEKNIK INFORMATIKA 2019
195150201111034
Jumlah Soal Solve : 19

Daftar Isi

Daftar Isi.....	2
Miscellaneous.....	4
Free_flag :.....	4
Jarum dalam jerami :.....	4
Hexify.....	5
Batu Kamu.....	7
Forensic.....	8
Pusing Kepala.....	8
Nizhe.....	10
Pembobol.....	11
Tempat Sampah Berjalan.....	13
Limao Sikocheng Brutal.....	15
Cryptography.....	22
Pemanasan.....	22
Blaise.....	22
2nd Stage Encryptor.....	24
Mbolan mbaleni.....	27
RiSAu.....	29
Kaisar2020.....	30
Reverse Engineering.....	31
Hello Reverse.....	31
Clang+Cling+Clung.....	32
UlaRE.....	33
Tidak Dipretelin.....	38
Pretelin.....	40
Binary Hacking.....	42
Welcome to Binaries.....	42

overflow0x01.....	42
Web Exploitation.....	43
Hack Me Plz!.....	43
Lorem Femto Ipsum.....	44

Miscellaneous

Free_flag :

Free flag merupakan percobaan untuk mencoba format flagnya, sehingga tidak ada langkah-langkah tertentu yang perlu dilakukan

Hal yang perlu dilakukan adalah mensubmit flag yang telah ada yaitu

flag{oprec_2020_t357_fl4g}

Jarum dalam jerami :

Pada soal jarum dalam jerami, terdapat sebuah zip dimana kita akan mencari string yang benar. Tetapi di dalam zip tersebut terdapat 65536 folder. Tentu akan kesulitan untuk mencari string tersebut di dalam 65536 folder yang mana mayoritas atau bahkan hampir semua isi dari folder tersebut kosong. Maka saya menggunakan script yaitu sebagai berikut untuk menghapus semua folder yang kosong

```
find /home/aldifianda/tumpukan_jerami -type d -empty -delete
```

Kegunaan dari script ini adalah untuk menghapus folder yang kosong, sehingga nantinya tersisa 5 folder yaitu folder 9613, 10195, 19040, 21098 dan 50607. Dari 5 folder ini terdapat 4 flag palsu dan 1 flag asli, disini saya tidak mencari string yang mengandung kalimat jarum, tetapi saya memilih text file yang isinya berbeda sendiri yaitu saya dapatkan pada folder nomor 21098. Sehingga diperoleh flag sebagai berikut (karena folder yang lainnya mengatakan kalau flag mereka itu palsu) :

flag{linux_7r33_ds_w1th_gr3p}

Hexify

Pada soal hexify, terlihat bahwa Ucup mengubah suatu string menggunakan source code buatannya yaitu :

```
for (int i = 0; i < myst.length; i++) {  
    if (i%2==0) System.out.print((myst[i] ^ 70) + " ");  
    else if (i%2==1) System.out.print((myst[i] ^ 80) + " ");  
}
```

Sehingga diperoleh string sebagai berikut :

203c27373d3a1365320f720f73390b207715190275260322736128663
b

Jika dilihat dari hasil stringnya dan juga clue dari soalnya. Dapat dipastikan string ini dalam bentuk hexadecimal karena mengandung beberapa huruf (a-f).

Untuk menyelesaikan soal ini saya memecah kode hexadecimal ini menjadi masing-masing terdiri dari 2 buah angka, sehingga hasilnya sebagai berikut

20 3c 27 37 3d 3a 13 65 32 0f 72 0f 73 39 0b 20 77
15 19 02 75 26 03 22 73 61 28 66 3b

Lalu dari angka-angka yang sudah dipisah ini saya ubah menjadi bilangan decimal, sehingga menjadi

32 60 39 55 61 58 19 101 50 15 114 15 115 57 11 32 119
21 25 02 117 38 03 34 115 97 40 102 59

Lalu, untuk satu buah substring yang terdiri dari 2 buah angka(misal 32) saya XOR kan, karena dari source code Ucup dia menggunakan

XOR untuk mengubah nilainya. Sehingga untuk substring nomor 0 (karena index dimulai dari 0, sehingga 32 merupakan substring ke 0) saya XOR kan dengan 70 karena dari rumusnya index yang habis dimodulo 2 harus di XOR dengan 70, dan untuk index yang dimodulo 2 hasilnya 1 di XOR dengan 80, sehingga diperoleh hasil sebagai berikut :

102 108 97 103 123 106 85 53 116 95 52 95 53 105 77 112 49
69 95 82 51 118 69 114 53 49 110 54 125

Setelah didapatkan kode seperti yang ada di atas, maka tinggal kita rubah ke kode ascii, kode ascii lengkapnya dapat dilihat di

<http://www.asciitable.com/>

Sehingga diperoleh hasil sebagai berikut

f l a g { j U 5 t _ 4 _ 5 i M p 1 E _
R 3 v E r 5 1 n 6 }

Maka, karena kalimatnya dapat dibaca maka inilah string yang aslinya. Sehingga flagnya adalah :

flag{jU5t_4_5iMp1E_R3vEr51n6}

Batu Kamu

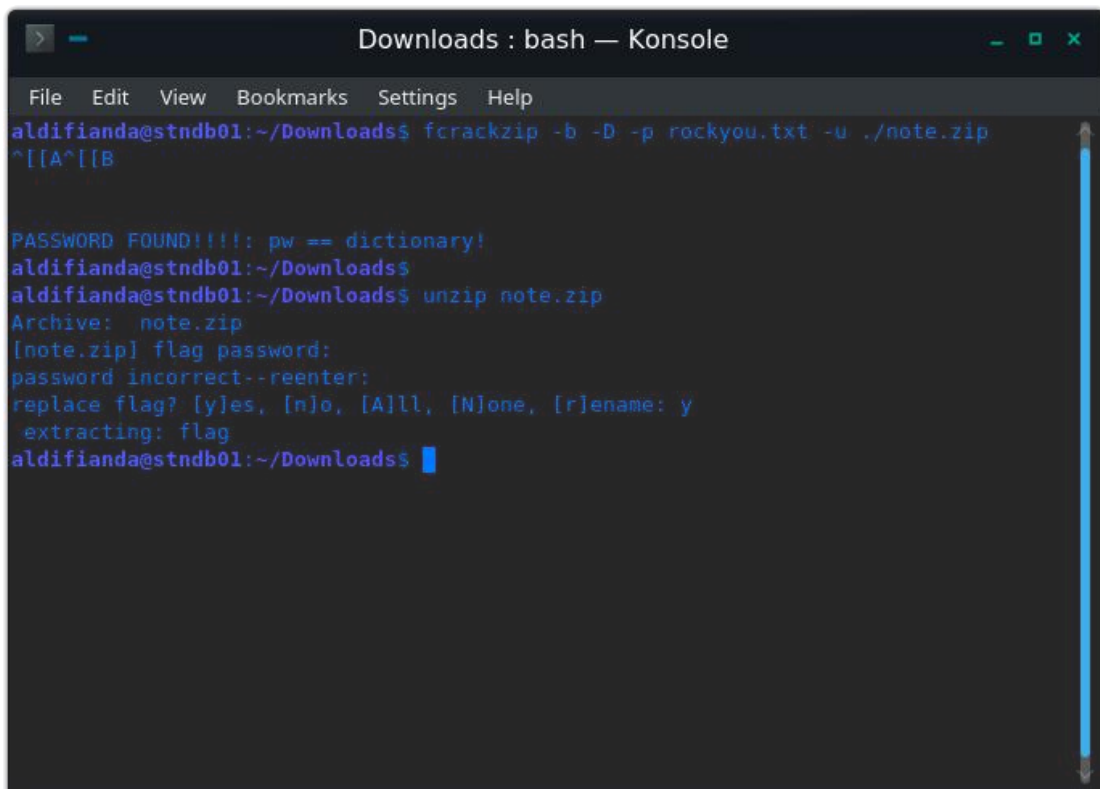
Pada soal batu kamu, kita diminta untuk memecahkan kode password zipnya Pak Lavi. Untuk memecah kode password ini saya menggunakan sebuah database kemungkinan password yakni Rockyou.txt yang dapat diperoleh di

https://github-production-release-asset-2e65be.s3.amazonaws.com/97553311/d4f580f8-6b49-11e7-8f70-7f460f85ab3a?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20200306%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200306T163209Z&X-Amz-Expires=300&X-Amz-Signature=970747bf9fc87f143bfceb6cdad89d7d1e584411bbc8d420c3e9f5d93d61b5ff&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename%3Drockyou.txt&response-content-type=application%2Foctet-stream

Selain Rockyou, saya juga menggunakan tool yakni fcrackzip untuk mendapatkan passwordnya yang dikombinasikan dengan file Rockyou yaitu dengan script berikut

```
fcrackzip -b -D -p rockyou.txt -u ./note.zip
```

Prosesnya agak lama karena membrute force banyak kemungkinan password. Maka setelah itu diperoleh passwordnya dan kita bisa mengekstrak filenya.



```
Downloads : bash — Konsole
File Edit View Bookmarks Settings Help
aldefianda@stndb01:~/Downloads$ fcrackzip -b -D -p rockyou.txt -u ./note.zip
^[[A^[[B

PASSWORD FOUND!!!!: pw == dictionary!
aldefianda@stndb01:~/Downloads$
aldefianda@stndb01:~/Downloads$ unzip note.zip
Archive: note.zip
[note.zip] flag password:
password incorrect--reenter:
replace flag? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
extracting: flag
aldefianda@stndb01:~/Downloads$
```

Lalu dalam file flag ternyata dalam base64, maka tinggal di decode saja dan diperoleh flag sebagai berikut

flag{z1p_c4n_b3_Cr4cK3d_u51nG_d1ct10NaRY_aTt4ck}

Forensic

Pusing Kepala

Pada soal pusing kepala, kita mendownload file png yang tidak bisa dibuka. Apabila di jalankan kode "file flag.png" maka dia akan mengatakan bahwa file flag.png berformat data, seharusnya berformat png. Apabila dibuka dengan hexeditor, maka terdapat kesalahan pada headernya yaitu sebagai berikut

```
Downloads : hexedit — Konsole
File Edit View Bookmarks Settings Help
00000000 05 F4 74 D9 9A 10 87 00 00 00 0D 49 48 44 52 ...t.....IHDR
00000010 00 00 04 38 00 00 04 38 08 06 00 00 00 EC 10 6C ...8...8.....l
00000020 8F 00 00 09 E3 7A 54 58 74 52 61 77 20 70 72 6F ....zTXtRaw pro
00000030 66 69 6C 65 20 74 79 70 65 20 65 78 69 66 00 00 file type exif..
00000040 78 DA ED 98 59 72 23 39 0E 86 DF 79 8A 39 02 41 x...Yr#9...y.9.A
00000050 12 5C 8E 43 70 89 98 1B CC F1 E7 43 4A 76 97 AB .\Cp.....CJv..
00000060 7B 7A AA A3 5E ED B4 94 72 2A 93 04 F1 2F 00 1D {z...r*.../..
00000070 CE 7F FE 7D C3 BF F8 C9 B9 F7 50 B4 F5 3A 6A 8D ...}.....P...j.
00000080 FC 94 51 46 9A 7C E8 F1 F5 33 9F 77 89 E5 79 7F ..QF...3.w.y.
00000090 7E 72 7F FE F2 BF BF 5C 0F 9F 5F 24 2E 65 BF F3 ~r....\...$.e..
000000A0 F5 45 AF EF C1 0E D7 13 F7 A7 F7 F5 F5 3A CB E4 .E.....
000000B0 BA FE 30 D0 38 EF 2F EC EB 17 F3 3D 50 EA EF 09 ..0.8./.....=P...
000000C0 DE D7 3F 26 CA F2 9A 20 EE F7 40 F3 3D 50 4E EF ..?&...@.=PN.
000000D0 99 5F F1 45 7B CF 5C 47 6F 3F 2E E1 FD 9C 2F 5F ..E{.Go?..../_
000000E0 DE E7 E7 15 FC AD E4 96 AA 56 69 85 F7 92 62 6B .....Vi...bk
000000F0 75 F0 B9 A7 58 1A 79 DB 1E E8 5D 69 F8 73 6A AF u...X.y...li.sj.
00000100 71 7E FE 3B 7C DC 9A 88 29 9D 2C 39 3E EF E5 15 q~;|...),9>...
00000110 65 7E BD E6 73 9E BC 94 FB 84 E3 E3 8A E4 F6 E4 e~..s.....
00000120 37 06 20 23 04 22 1F EF DC BE 97 EA D9 FC 92 9B 7. #. ....
00000130 8F F3 FF F8 09 BF B2 AC 37 1D BE C0 FD F9 E9 4D .....7.....M
00000140 83 F0 A7 2F 7E A2 41 3D EF EB F9 27 F4 EA E7 F9 .../~.A=...
00000150 B9 1E 7E FE 42 F4 AF E1 7E 30 FD 21 A2 56 3E 27 ...~.B...~0.!V>
00000160 4E 5F 22 5A 29 DA 97 45 F7 3F 5E F7 EE 7E EF 79 N_"Z)..E.?^...y
--- flag2.png --0x0/0x595D---
```

Harusnya file png memakai header .PNG pada headernya, maka kita tinggal rubah kode hexnya saja, jadi saya mengganti 8 angka hexadecimal pada baris pertama sesuai dengan hex headernya png yaitu 89 50 4E 47 0D 0A 1A 0A lalu di save.

Setelah di save maka format gambar akan berubah menjadi png dan gambar bisa dibuka sehingga diperoleh gambar berikut :



Maka flag untuk soal ini adalah

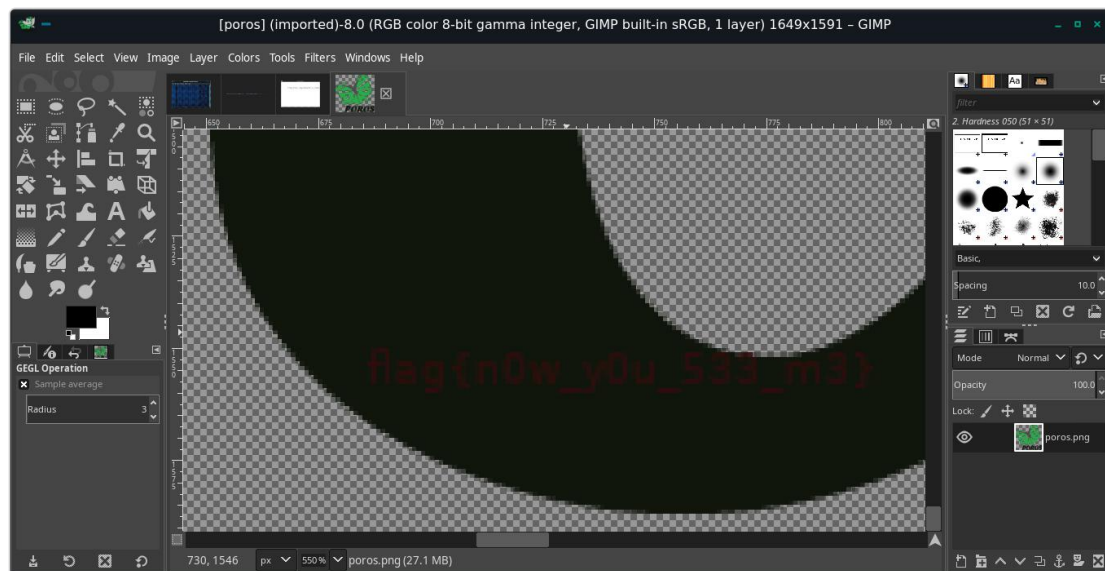
Flag{h3x_5ign4tuR3_is_fuN}

Nizhe

Pada soal nizhe, kita mendapatkan logo poros dan mencari flag pada gambar tersebut



Pada soal ini, untuk mencari jawabannya kita tinggal mengedit foto dan menukar opacitynya saja lalu di zoom di huruf POROS nya sehingga diperoleh :

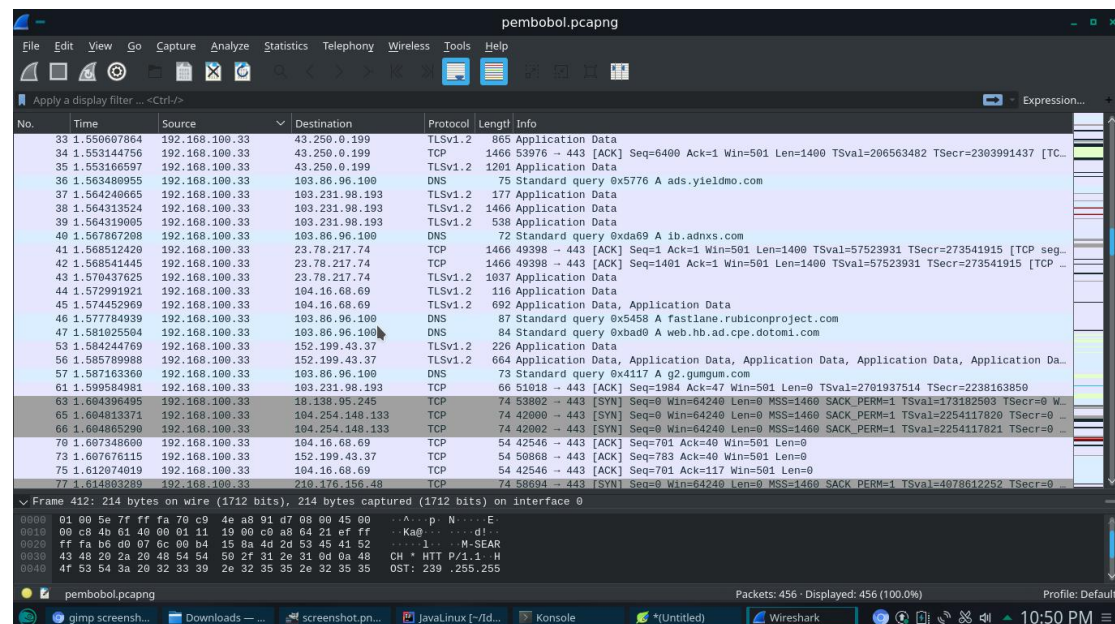


Maka didapatkan flag untuk soal ini maka flagnya :

flag{n0w_y0u_533_m3}

Pembobol

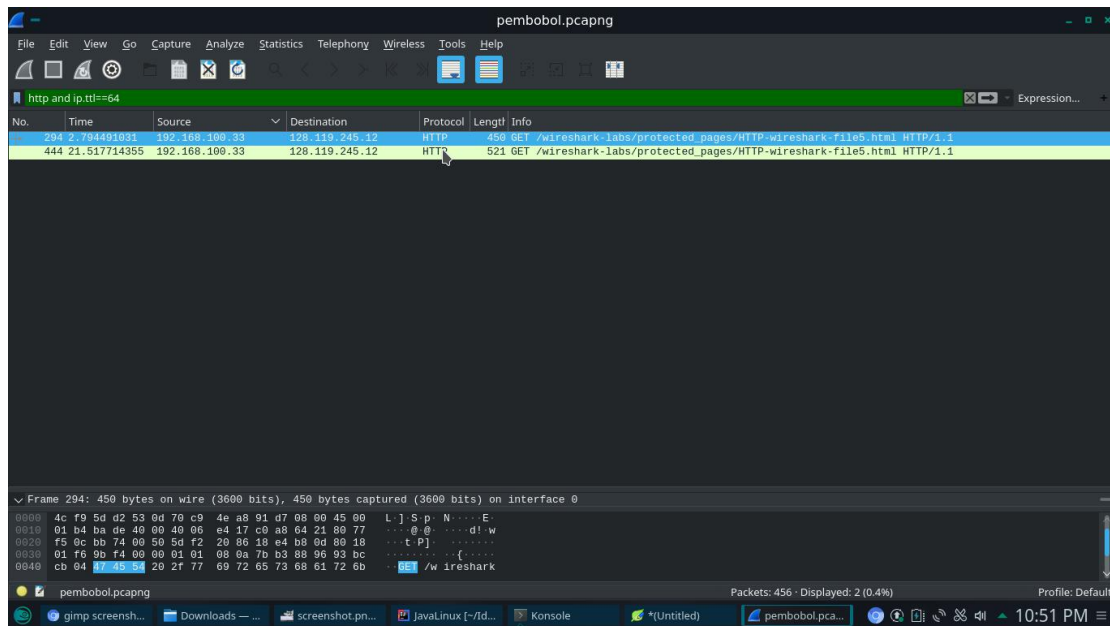
Pada soal ini kita diberikan file berformat pcapng, file ini adalah file untuk aplikasi wireshark sehingga kita tinggal buka saja dengan wireshark sehingga diperoleh data-data sebagai berikut



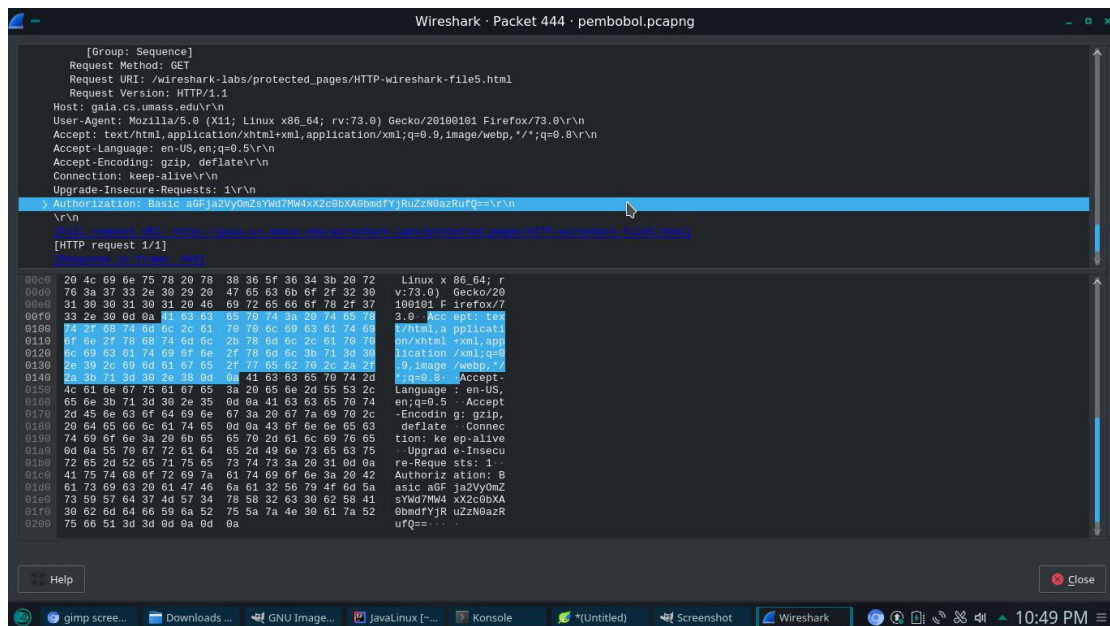
Untuk memudahkan mencarinya, saya mengurutkan sourcenya sehingga terlihat sumber-sumber pacakaganya.

Disini kita mencari package yang ditinggalkan oleh pembobol, namun pada source terakhir yang bukan merupakan ip address hanya diperoleh vendor gadget yang digunakan oleh pembobol.

Karena setelah dilihat tidak ada yang aneh, saya mencoba mencari jejakny pada file htmlnya, untuk memudahkan mencarinya saya memfilternya dengan "html and ip.ttl==64" seperti gambar dibawah



Lalu saya mencoba membuka html tersebut dan melihat semua isinya, awalnya tidak ada yang aneh. Namun pada salah satu html terdapat sebuah kode yang sama seperti tipe base64, kode tersebut saya highlight seperti gambar dibawah



aGFja2VyOmZsYWd7MW4xX2c0bXA0bmdfYjRuZzN0azRufQ== menurut saya adalah keynya, lalu tinggal di decode code base64 tersebut ke ascii sehingga diperoleh kalimat berikut hacker:flag{1n1_g4mp4ng_b4ng3tk4n}

Maka flag untuk soal ini adalah

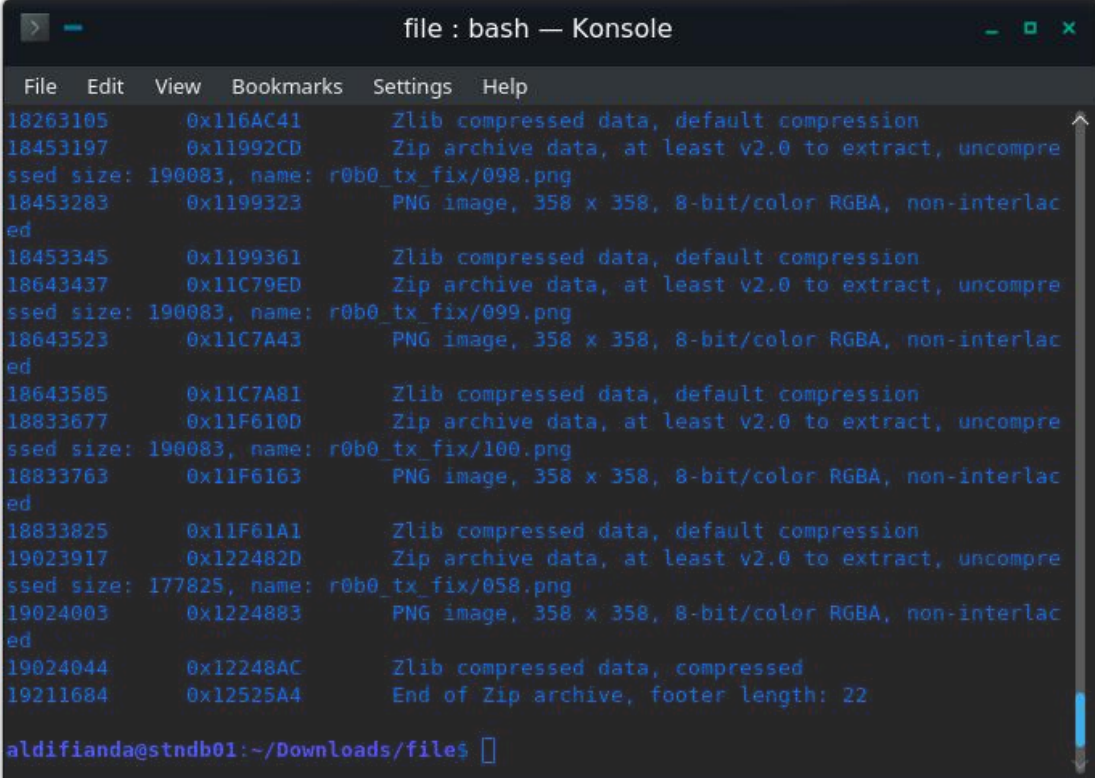
flag{1n1_g4mp4ng_b4ng3tk4n}

Tempat Sampah Berjalan

Untuk soal ini, didapatkan cluenya dari judulnya yaitu binwalk. Binwalk adalah tool untuk mengecek isi dari suatu file. Maka kita tinggal masukan script berikut ke terminal

```
binwalk robot.jpg
```

Nanti akan muncul semua file yang tersimpan pada gambar robot ini, terdapat 1 buah folder berisi gambar robot dan 1 buah file zip yang berisi 100 gambar robot yang sama, tetapi pada akhir file. Gambar robot yang ke 058 terletak paling akhir, harusnya berada di tengah-tengah jika diurutkan, apabila diperhatikan juga, gambar 058.png ini memiliki ukuran yang berbeda dari 99 gambar yang lainnya. Maka kemungkinan besar flag ada pada gambar ini



```
file : bash — Konsole
File Edit View Bookmarks Settings Help
18263105 0x116AC41 Zlib compressed data, default compression
18453197 0x11992CD Zip archive data, at least v2.0 to extract, uncompress
ssed size: 190083, name: r0b0_tx_fix/098.png
18453283 0x1199323 PNG image, 358 x 358, 8-bit/color RGBA, non-interlac
ed
18453345 0x1199361 Zlib compressed data, default compression
18643437 0x11C79ED Zip archive data, at least v2.0 to extract, uncompress
ssed size: 190083, name: r0b0_tx_fix/099.png
18643523 0x11C7A43 PNG image, 358 x 358, 8-bit/color RGBA, non-interlac
ed
18643585 0x11C7A81 Zlib compressed data, default compression
18833677 0x11F610D Zip archive data, at least v2.0 to extract, uncompress
ssed size: 190083, name: r0b0_tx_fix/100.png
18833763 0x11F6163 PNG image, 358 x 358, 8-bit/color RGBA, non-interlac
ed
18833825 0x11F61A1 Zlib compressed data, default compression
19023917 0x122482D Zip archive data, at least v2.0 to extract, uncompress
ssed size: 177825, name: r0b0_tx_fix/058.png
19024003 0x1224883 PNG image, 358 x 358, 8-bit/color RGBA, non-interlac
ed
19024044 0x12248AC Zlib compressed data, compressed
19211684 0x12525A4 End of Zip archive, footer length: 22
aldifianda@stndb01:~/Downloads/files$
```

Maka kita tinggal ekstrak semua isi yang terkandung pada robot.jpg dengan menggunakan script berikut

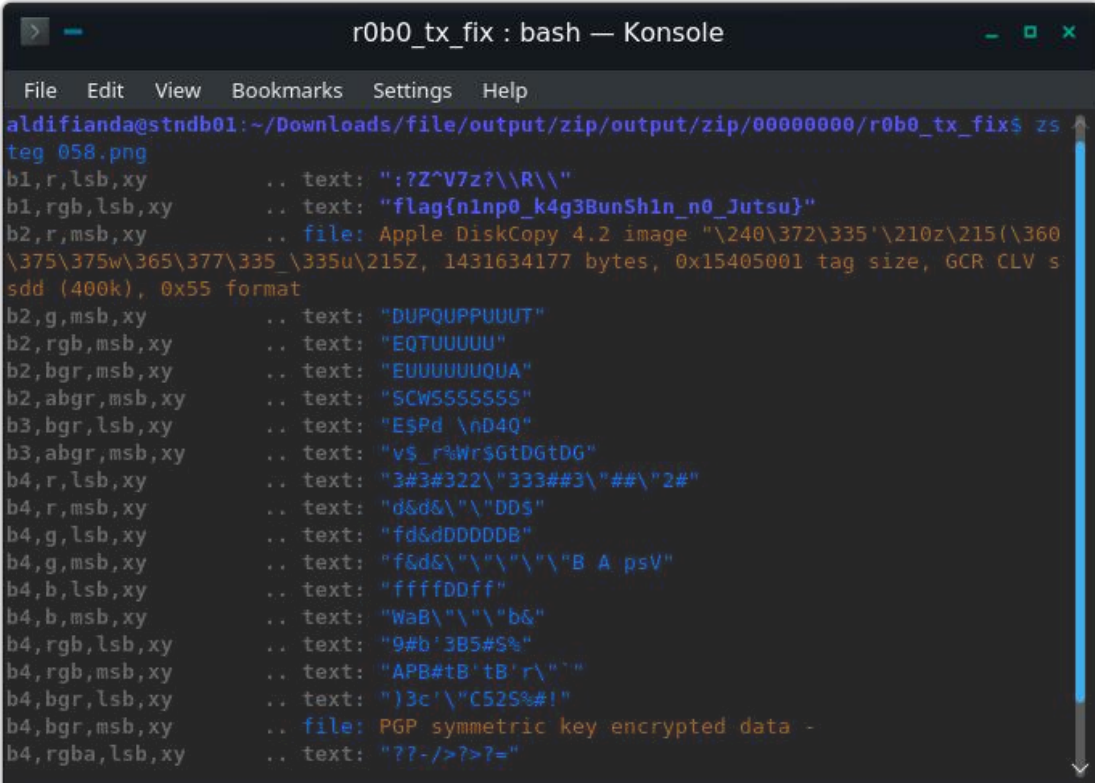
```
foremost robot.jpg
```

Nantinya script ini akan membuat folder output dan 2 folder tadi (png dan zip) serta 1 buah file bernama audit.txt yang berisikan informasi file yang diekstrak tadi.

Lalu kita tinggal cek gambar 058.png tersebut dengan menggunakan tools bernama zsteg dengan scriptnya sebagai berikut :

zsteg 058.png

Maka akan muncul tampilan seperti berikut



```
r0b0_tx_fix : bash — Konsole
File Edit View Bookmarks Settings Help
aldifianda@stndb01:~/Downloads/file/output/zip/output/zip/00000000/r0b0_tx_fix$ zsteg 058.png
b1,r,lsb,xy      .. text: "7Z^V7z?\\R\\"
b1,rgb,lsb,xy    .. text: "flag{n1np0_k4g3BunSh1n_n0_Jutsu}"
b2,r,msb,xy      .. file: Apple DiskCopy 4.2 image "\240\372\335'\210z\215(\360
\375\375w\365\377\335_\335u\215Z, 1431634177 bytes, 0x15405001 tag size, GCR CLV s
sdd (400k), 0x55 format
b2,g,msb,xy      .. text: "DUPQUPPUUUT"
b2,rgb,msb,xy    .. text: "EQTUUUUU"
b2,bgr,msb,xy    .. text: "EUUUUUUUQUA"
b2,abgr,msb,xy   .. text: "SCWSSSSSSSS"
b3,bgr,lsb,xy    .. text: "E$Pd \nD4Q"
b3,abgr,msb,xy   .. text: "v$_r%Wr$GtDGtDG"
b4,r,lsb,xy      .. text: "3#3#322\"333##3\"##\"2#"
b4,r,msb,xy      .. text: "d&d&\\"DDs"
b4,g,lsb,xy      .. text: "fd&dDDDDDB"
b4,g,msb,xy      .. text: "fd&d&\\"B A psV"
b4,b,lsb,xy      .. text: "ffffDDff"
b4,b,msb,xy      .. text: "WaB\\"b&"
b4,rgb,lsb,xy    .. text: "9#b'3B5#S%"
b4,rgb,msb,xy    .. text: "APB#tB'tB'r\""
b4,bgr,lsb,xy    .. text: ")3c'\\"C52S%#!"
b4,bgr,msb,xy    .. file: PGP symmetric key encrypted data -
b4,rgba,lsb,xy   .. text: "?7- />?>?"
```

Maka kita mendapatkan flagnya yaitu

flag{n1np0_k4g3BunSh1n_n0_Jutsu}"

Limao Sikocheng Brutal

Untuk soal ini, didapatkan cluenya adalah hal yang berhubungan dengan LSB dan juga dari hintnya didapatkan juga teknik mengencodinya yaitu dengan cara steganography, tetapi dengan cara-cara yang ada di youtube CSI dan juga cara yang ada pada stackoverflow maupun github, tidak ada satupun yang memperoleh hasil.

Pertama, saya mencoba dengan beberapa source code yang ada pada stackoverflow yakni sebagai berikut

```
#!/usr/bin/python3

# -*- coding: utf-8 -*-
# @author: Andrew Quach and Stanislav Lyakhov
# @version: 3.0.0
#
# Basic LSB Encoder / Decoder

import sys
from PIL import Image, ImageMath

class LSB:
    SUPPORTED = ['RGB', 'RGBA', 'L', 'CMYK']
    def _set_bits(self, bits):
        self.bits = int(bits)
        if not 0 <= self.bits <= 8:
            print('[!] Number of bits needs to be between 0-8.')
            sys.exit()

    def _get_image(self, path, itype):
        try:
            img = Image.open(path)
        except IOError as e:
            print('[!] {} image could not be opened.'.format(itype.title()))
```



```

        print('[!] {}'.format(e))
        sys.exit()

    print('[*] {} image mode: {}'.format(itype.title(), img.mode))
    if img.mode not in self.SUPPORTED:
        print('[!] Nonsupported image mode.')
        sys.exit()
    return img

def _save_img(self, img, outfile):
    try:
        img.save(outfile)
    except IOError as e:
        print('[!] {} image could not be written.'.format(outfile))
        print('[!] {}'.format(e))
        sys.exit()
    except Exception as e:
        print('[!] Unable to save file.')
        print('[!] {}'.format(e))
        sys.exit()

class LSBEncode(LSB):
    def __init__(self, cover, secret, bits, outfile, mode=None):
        print('[*] Attempting LSB Encoding with bits = {}'.format(bits))
        self._set_bits(bits)
        self.outfile = outfile
        self.cover = self._get_image(cover, 'cover')
        if mode != None:
            self.cover = self.cover.convert(mode.upper())
            print('[*] Converted cover image mode to {}'.format(self.cover.mode))
        self.secret = self._get_image(secret, 'secret').convert(self.cover.mode)
        print('[*] Converted secret image mode to {}'.format(self.cover.mode))
        self._encode_img()

```



```

def _encode_img(self):
    c = self.cover.split()
    s = self.secret.split()

    expr = 'convert((c & (256 - 2**bits)) + ((s & (256 - 2**(8 - bits)) - 1) >> (8 - bits)), "L")'

    out = [ImageMath.eval(expr, c = c[k], s = s[k], bits = self.bits) for k in range(len(c))]

    out = Image.merge(self.cover.mode, out)
    self.cover.paste(out, (0, 0))

    self._save_img(self.cover, self.outfile)
    print('[*] Created outfile at {}'.format(self.outfile))

```

```

class LSBDecode(LSB):

```

```

    def __init__(self, steg, bits, outfile):
        print('[*] Attempting LSB Decoding with bits = {}'.format(bits))
        self._set_bits(bits)
        self.outfile = outfile
        self.steg = self._get_image(steg, 'steg')
        self._decode_img()

```

```

    def _decode_img(self):
        s = self.steg.split()

        expr = 'convert((s & 2**bits - 1) << (8 - bits), "L")'

        out = [ImageMath.eval(expr, s = s[k], bits = self.bits) for k in range(len(s))]
        out = Image.merge(self.steg.mode, out)

        self._save_img(out, self.outfile)
        print('[*] Created outfile at {}'.format(self.outfile))

```

```

def usage():

```

```

    print(''''Encoding Usage: steglsb -e cover_img secret_img bits outfile [mode]

```

```

> Embed a secret image into a cover image using LSB

```

Positional Arguments:

cover_img - path to cover image
secret_img - path to secret image
bits - number of rightmost bits to use (between 0-8)
outfile - path to output file

Optional Arguments:

mode - image mode to use 'RGB', 'RGBA', 'L', 'CMYK'

Decoding Usage: steglsb -d steg_img bits outfile

> Extract a secret image from a steganographic image using LSB

Positional Arguments:

steg_img - path to steg image
bits - number of rightmost bits to use (between 0-8)
outfile - path to output file

""")

def main():

if len(sys.argv) in (6, 7) and sys.argv[1] == '-e':

LSBEncode(*sys.argv[2:])

elif len(sys.argv) == 5 and sys.argv[1] == '-d':

LSBDecode(*sys.argv[2:])

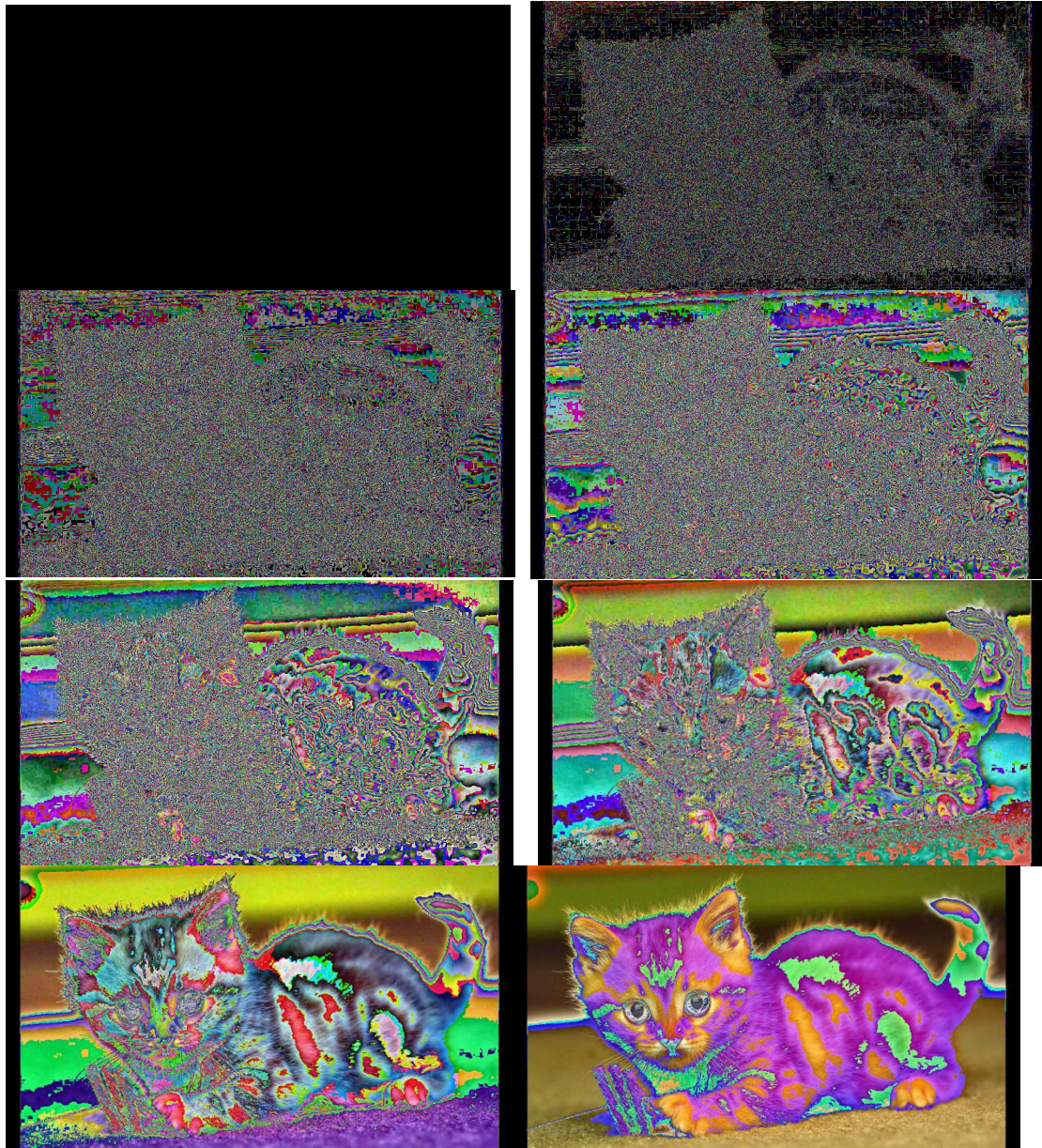
else:

usage()

if __name__ == '__main__':

main()

Kode tersebut adalah kode untuk merubah bit pada binarynya sehingga diperoleh warna yang berbeda. Setelah kode tersebut dijalankan dan ditargetkan pada gambar kucing tersebut maka akan terekstrak gambar sebagai berikut



Jika menggunakan tools bernama stegsolve, maka akan dihasilkan juga gambar yang sama.

Lalu dengan menggunakan source code yang berbeda

```
from PIL import Image
```

```
extracted_bin = []
```

```

with Image.open("steik.png") as img:

    width, height = img.size

    byte = []

    for x in range(0, width):

        for y in range(0, height):

            pixel = list(img.getpixel((x, y)))

            for n in range(0,3):

                extracted_bin.append(pixel[n]&1)

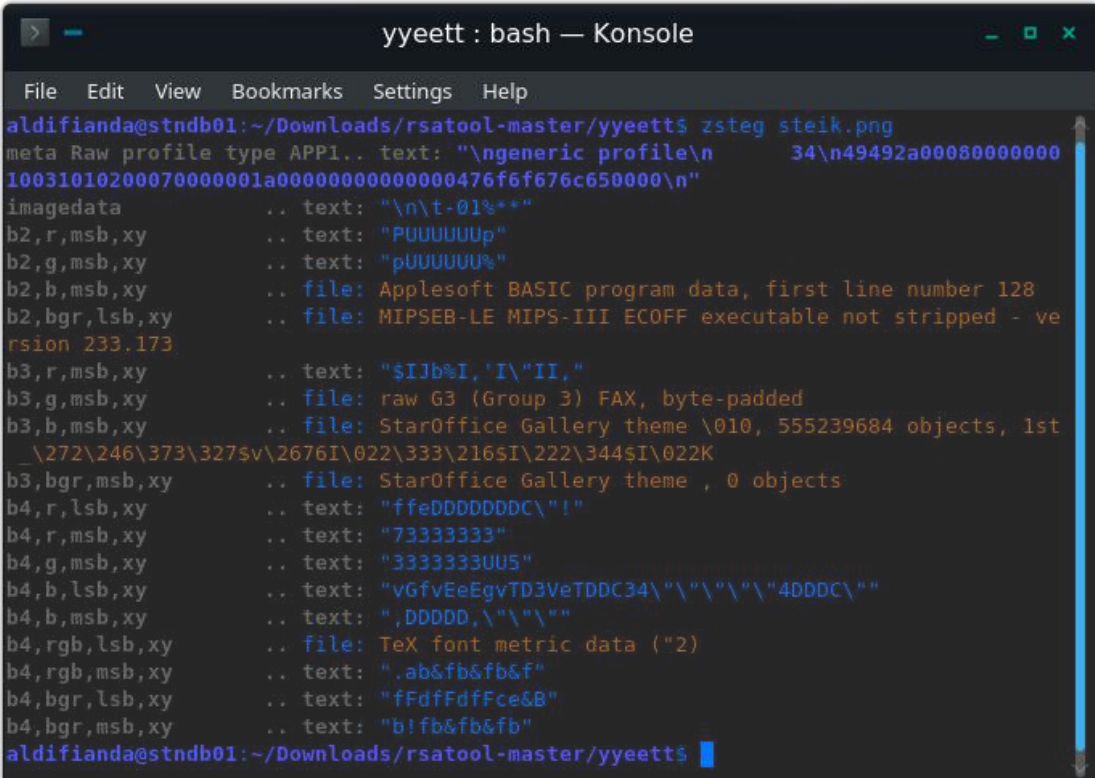
data = "".join([str(x) for x in extracted_bin])

print(data)

```

Diperoleh kalimat-kalimat yang tidak termasuk dalam ascii sehingga tidak bisa dibaca

Karena judulnya memberikan clue yaitu LSB, saya mencoba zsteg untuk mencoba apakah akan menghasilkan kode juga, tetapi tidak ditemukan satupun flag

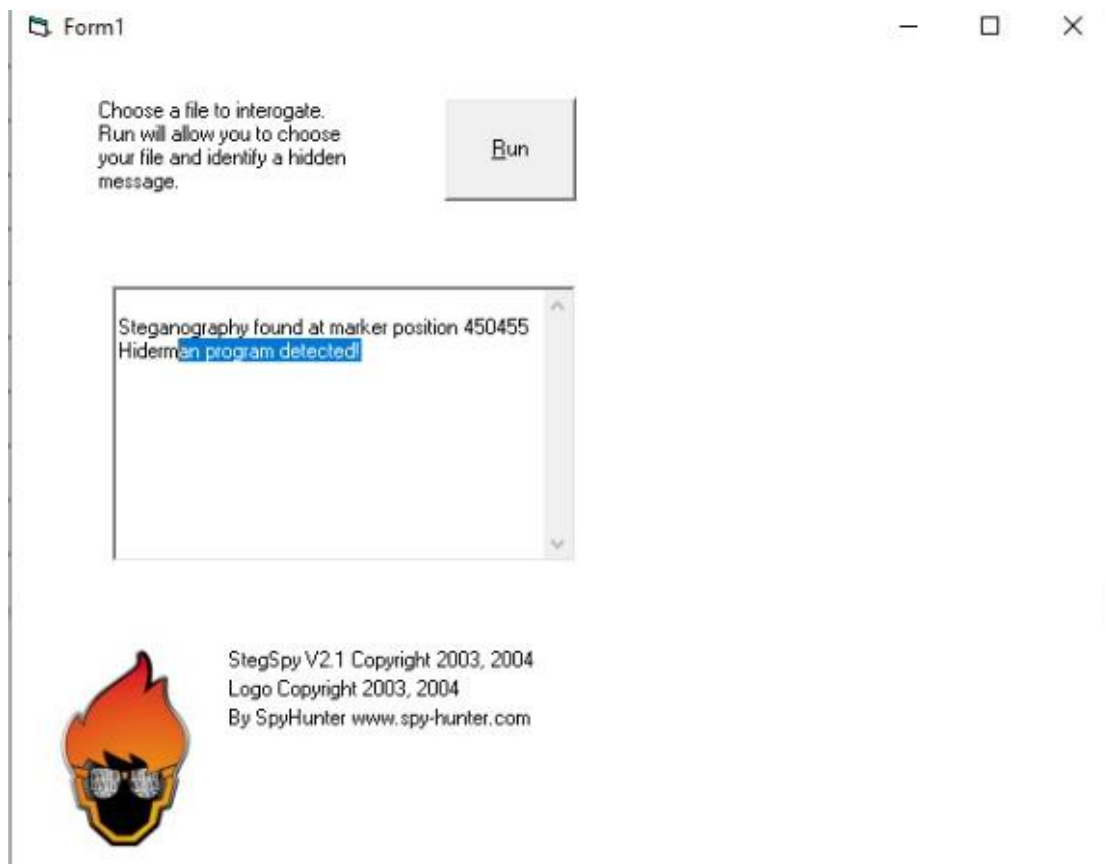


```

> yyeett : bash — Konsole
File Edit View Bookmarks Settings Help
aldifianda@stndb01:~/Downloads/rsatool-master/yyeett$ zsteg steik.png
meta Raw profile type APP1.. text: "\ngeneric profile\n          34\n49492a00080000000
10031010200070000001a0000000000000476f6f676c650000\n"
imagedata      .. text: "\n\t-01%*"
b2,r,msb,xy    .. text: "PUUUUUUp"
b2,g,msb,xy    .. text: "pUUUUUUU%"
b2,b,msb,xy    .. file: Applesoft BASIC program data, first line number 128
b2,bgr,lsb,xy  .. file: MIPSEB-LE MIPS-III ECOFF executable not stripped - ve
rsion 233.173
b3,r,msb,xy    .. text: "$Ijb%I,'I\"II,"
b3,g,msb,xy    .. file: raw G3 (Group 3) FAX, byte-padded
b3,b,msb,xy    .. file: StarOffice Gallery theme \010, 555239684 objects, 1st
_\272\246\373\327sv\2676I\022\333\216sI\222\344sI\022K
b3,bgr,msb,xy  .. file: StarOffice Gallery theme , 0 objects
b4,r,lsb,xy    .. text: "ffeDDDDDDDC\"!"
b4,r,msb,xy    .. text: "733333333"
b4,g,msb,xy    .. text: "3333333UU5"
b4,b,lsb,xy    .. text: "vGfvEeEgvTD3VeTDDC34\"\"\"\"4DDDC\""
b4,b,msb,xy    .. text: ",DDDDDD,\"\"\"\"
b4,rgb,lsb,xy  .. file: TeX font metric data ("2)
b4,rgb,msb,xy  .. text: ".ab&fb&fb&f"
b4,bgr,lsb,xy  .. text: "fFdfFdfFce&B"
b4,bgr,msb,xy  .. text: "b!fb&fb&fb"
aldifianda@stndb01:~/Downloads/rsatool-master/yyeett$

```

Apabila didetect pun juga memberi tahu bahwa ada text tersembunyi



Maka untuk soal ini, saya tidak menemukan flagnya.

Cryptography

Pemanasan

Pada soal pemanasan, kita diminta untuk memecahkan password pak afis yakni sebagai berikut

160064065065167060162144137061065137061062063ok

Awalnya saya beranggapan kalau ini adalah angka decimal, namun karena jumlahnya ganjil. Maka kemungkinan ini adalah bilangan octal sehingga tinggal dibagi 3 saja setiap hurufnya menjadi

160 064 065 065 167 060 162 144 137 061 065 137 061 062 063 o k
p 4 5 5 w 0 r d _ 1 5 _ 1 2 3 o k

Maka diperoleh flagnya yaitu

Flag{p455w0rd_15_123ok}

Blaise

Pada soal ini, kemungkinan tipe enkripsi dari file ini adalah vignere

karena pasti blaise ini adalah cluenya dan tipe enkripsi yang mendekati nama ini adalah vignere karena vignere dibuat oleh blaise de vignere

cara mencari vigenere chiper adalah dengan rumus berikut

$$P = (C - K) \bmod 26$$

dimana

P adalah plaintext atau huruf awal

C adalah chipertextnya / hasil enkripsi

K adalah key / kunci untuk enkripsinya

clue dari kuncinya adalah "keren" maka dicoba dahulu dengan "keren"

maka hasilnya sebagai berikut

xvxn{kmqylfvl_tc_opr_poambb_lcseaf}

maka hasilnya belum benar, maka kita coba dengan kunci "cool"
maka diperoleh hasil sebagai berikut

flag{vigenere_is_not_secure_enough}

2nd Stage Encryptor

Pada soal ini kita diberikan sebuah source code sebagai berikut

```
public class encryptor {
    static class enkripsi {
        String s = "flag{}";
        String m[];
        String c = "";

        public enkripsi(String msg) {
            this.m = msg.split("_");
            c = encrypt();
        }

        public String encrypt() {
            long kunci[] = { 116, 104, 101, 32, 107, 101, 121 };
            int i = 0;
            for (String minion : m) {
                int counter = 0;
                for (int j = 0; j < minion.length(); j++) {
                    c = c + (minion.charAt(j) ^ (kunci[counter] + i));
                    i += 4;
                    counter++;
                    if (counter == kunci.length) {
                        counter = 0;
                    }
                    c += "Sec0nDSta6e";
                }
                i = 0;
                c += "_";
            }
            return c;
        }
    }
}
```



```

    public void decrypt() {
        // [REDACTED]
    }

    public void printFlag() {
        System.out.println("ok!, here's your flag: \n\t\t" + s);
    }
}

public static void main(String args[]) {
    String message = "[REDACTED]";
    enkripsi cipher = new enkripsi(message);
    System.out.println("\n" + cipher.c + "\n");
}
}

```

Untuk menjawab soal ini, saya melakukan operasi XOR secara terbalik yaitu menXOR kan hasil enkripsinya dengan keynya karena apabila melakukan XOR pasti akan kembali semula seperti apabila

1 XOR 1 = 0, maka 0 XOR 1 nilainya akan menjadi 1, begitu juga dengan angka lainnya. Jadi saya menXORkan enc dengan key dengan keynya bertambah 4 setiap kali melakukan XOR, berikut operasinya, namun sebelumnya saya mengganti Sec0nDSta6e dengan "/" karena menurut saya string itu mengganggu. (hasil xor nya langsung saya ganti ke kode Ascii nya)

30 XOR 116+0 = j

88 XOR 104+4 = 4

27 XOR 101+8 = v

24 XOR 32+12 = 4

_ (i kembali ke 0)

7 XOR 116+0 = s

93 XOR 104+4 = 1

0 XOR 101+8 = m

92 XOR 32+12 = p

23 XOR 107+16 = l

74 XOR 101+20 = 3

-

71	XOR 116+0	= 3
2	XOR 104+4	= n
14	XOR 101+8	= c
94	XOR 32+12	= r
2	XOR 107+16	= y
9	XOR 101+20	= p
166	XOR 121+24	= 7
200	XOR 116+28	= X
184	XOR 104+32	= 0
219	XOR 101+36	= R
-		
69	XOR 116+0	= 1
84	XOR 104+4	= 8
95	XOR 101+8	= 2
79	XOR 32+12	= c
30	XOR 107+16	= e
31	XOR 101+20	= f
245	XOR 121+24	= d
165	XOR 116+28	= 5
188	XOR 104+32	= 4

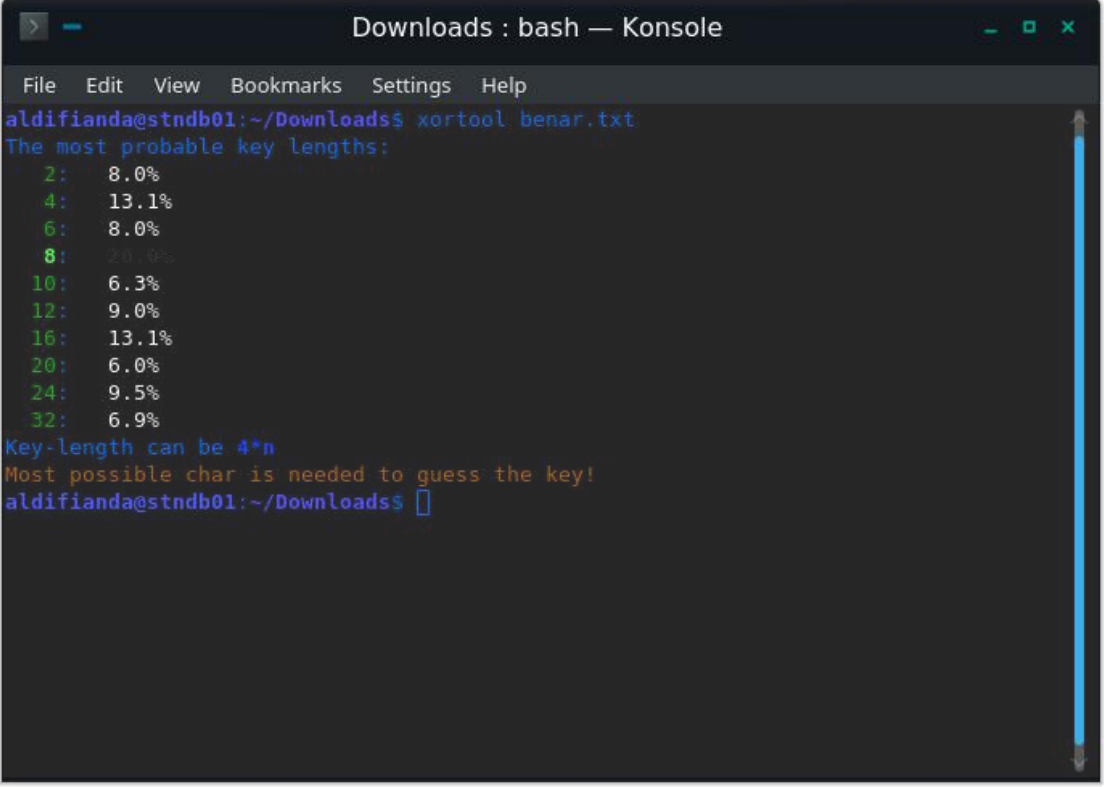
Maka diperoleh flag sebagai berikut

flag{j4v4_s1mpl3_3ncryp7X0R_182cefd54}

Mbolan mbaleni

Pada soal ini diberikan sebuah file yang berisikan string aneh yang tidak dapat dibaca, berdasarkan hint yang diberikan mengatakan ada pada video csi, maka cara mengenkripsinya adalah dengan cara repeated xor

Maka saya menggunakan tools yaitu xortool untuk mendapatkan kunci enkripsinya, maka diperoleh kemungkinan kunci sebagai berikut

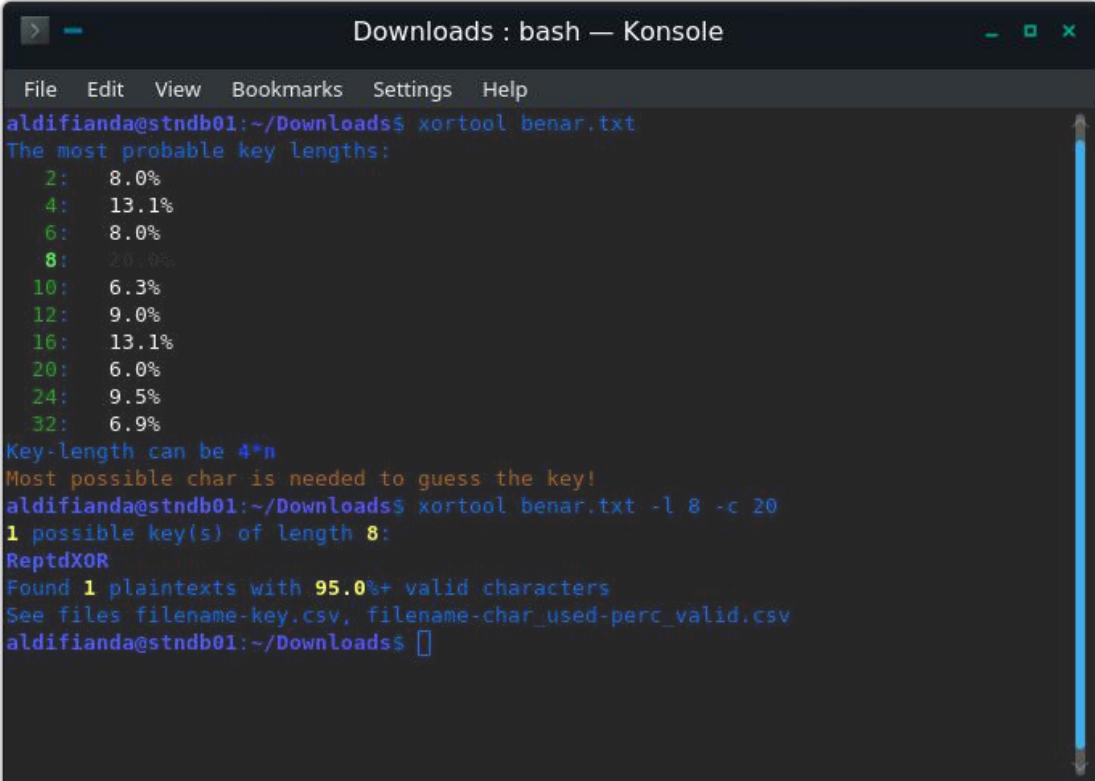


```
Downloads : bash — Konsole
File Edit View Bookmarks Settings Help
aldifianda@stndb01:~/Downloads$ xortool benar.txt
The most probable key lengths:
 2: 8.0%
 4: 13.1%
 6: 8.0%
 8: 20.0%
10: 6.3%
12: 9.0%
16: 13.1%
20: 6.0%
24: 9.5%
32: 6.9%
Key-length can be 4*n
Most possible char is needed to guess the key!
aldifianda@stndb01:~/Downloads$
```

Maka kita tinggal cari kunci tersebut, agar lebih cepat dan tidak perlu mencari banyak kemungkinan, maka masukan script berikut

```
xortool -l 8 -c 20
```

Lalu diperoleh kunci yaitu



```
Downloads : bash — Konsole
File Edit View Bookmarks Settings Help
aldifianda@stndb01:~/Downloads$ xortool benar.txt
The most probable key lengths:
 2: 8.0%
 4: 13.1%
 6: 8.0%
 8: 23.0%
10: 6.3%
12: 9.0%
16: 13.1%
20: 6.0%
24: 9.5%
32: 6.9%
Key-length can be 4*n
Most possible char is needed to guess the key!
aldifianda@stndb01:~/Downloads$ xortool benar.txt -l 8 -c 20
1 possible key(s) of length 8:
ReptdXOR
Found 1 plaintexts with 95.0%+ valid characters
See files filename-key.csv, filename-char_used-perc_valid.csv
aldifianda@stndb01:~/Downloads$
```

setelah diperoleh kunci tersebut, kita tinggal buka folder xortool_out yang telah terbuat sejak awal ketika melakukan xortool benar.txt, lalu dibuka file 0.out nya dan diperoleh text berikut

Capture the Flag (CTF) is a special kind of information security competitions. There are three common types of CTFs: Jeopardy, Attack-Defence and mixed. Jeopardy-style CTFs has a couple of questions (tasks) in range of categories. For example, Web, Forensic, Crypto, Binary or something else. Team can gain some points for every solved task. More points for more complicated tasks usually. The next task in chain can be opened only after some team solve previous task. Then the game time is over sum of points shows you a CTF winner. Famous example of such CTF is Defcon CTF quals. Well, attack-defence is another interesting kind of competitions. Here every team has own network(or only one host) with vulnerable services. Your team has time for patching your services and developing exploits usually. So, then organizers connects participants of competition and the wargame starts! You should protect own services for defence points and hack opponents for attack points. Historically this is a first type of CTFs, everybody knows about DEF CON CTF - something like a World Cup of all other competitions. Oh? you want the flag huh... here: cnA3ZF94MHJfanU1dF9ycDkXzRnNDFuXzRIODQ3ZmZhYwo=n

Kalimat paling bawah adalah kuncinya, tinggal di decode dari base64 ke ascii sehingga diperoleh flagnya yaitu

flag{rp7d_x0r_ju5t_rp7d_4g41n_4e847ffac}

RiSAu

Pada soal ini, kita dapatkan clue untuk cara mencarinya yaitu dengan metode RSA dan diberikan kode sebagai berikut

n =
30064958471180141352963255964320727764941087854957385562672821662
31985402139510096882334110807502092854243744699399411986390256587
43552961884983047613893364384218896364095619361419857868010029237
52627293790265351723795968412774268086467114263767947693310444934
31620539081418580251751469452850133385125508465392518172697873480
48067077404447559083987519648991434945227814054571036973738689728
36201511424363601490903086488506985489526910314474245106338585623
57136954938843486556795198686644530684050539726828188988673801589
19821623714131368859897469319297877656178387503812260367841224981
43172854419447324975505933540511

e = 65537

c =
27456233208801999463737070845466431326382231091595925265664784256
44706584808273308901029419265796597698078283187767182525070142004
44536149576571472071743436070238297135507371441005560133565686235
99332046857871289682182397318925369561237609610680925146051300193
28587851373014496681330160449754619385691874039323851621602918382
64219704361839947035886610725912605550332630429976993818360450844
16689195484076115050857645578915560175548502696278621380200947845
24878587266681295680701504880656692432458873850116850643262851141
29499962789033011964944406972645270628834725560187620906310265603
2117877428691647465815948625931

n adalah plaintext atau text awalnya

e adalah kode enkripsinya

c adalah hasil dari enkripsinya

Untuk rumus RSA sendiri lumayan panjang dimana awalnya dicari p dan q, lalu nilai n diperoleh dari p.q dan nantinya ada juga plaintext awalnya yang berawal dari kode ascii, caranya lumayan panjang dan pada soal hanya diberikan n, e dan c saja. Maka untuk memudahkan penghitungan karena angkanya sangat panjang maka saya menggunakan kalkulator RSA yang diperoleh dari link berikut

<https://github.com/Ganapati/RsaCtfTool>

Untuk penggunaannya sendiri dilakukan dengan cara berikut

```
python RsaCtfTool -n =
30064958471180141352963255964320727764941087854957385562672821662
31985402139510096882334110807502092854243744699399411986390256587
43552961884983047613893364384218896364095619361419857868010029237
52627293790265351723795968412774268086467114263767947693310444934
31620539081418580251751469452850133385125508465392518172697873480
48067077404447559083987519648991434945227814054571036973738689728
36201511424363601490903086488506985489526910314474245106338585623
57136954938843486556795198686644530684050539726828188988673801589
19821623714131368859897469319297877656178387503812260367841224981
43172854419447324975505933540511

-e 65537 --uncipher
27456233208801999463737070845466431326382231091595925265664784256
44706584808273308901029419265796597698078283187767182525070142004
44536149576571472071743436070238297135507371441005560133565686235
99332046857871289682182397318925369561237609610680925146051300193
28587851373014496681330160449754619385691874039323851621602918382
64219704361839947035886610725912605550332630429976993818360450844
16689195484076115050857645578915560175548502696278621380200947845
24878587266681295680701504880656692432458873850116850643262851141
29499962789033011964944406972645270628834725560187620906310265603
2117877428691647465815948625931
```

Sehingga nanti akan diperoleh flag sebagai berikut

Flag{rsa_sm00ll_f4ct0r_is7_53cur3}

Kaisar2020

Pada soal ini diberikan string sebagai berikut

iodj{4vw_h3s3_fu2s0cju7sk2_8tx6332}

Dari soalnya, cara untuk memecahkannya adalah dengan caesar cipher, maka di decrypt dengan caesar cipher sebanyak shift 24 sehingga menjadi

flag{4st_e3p3_cr2p0zgr7ph2_8qu6332}

Namun ini bukanlah flagnya, karena masih ada step lainnya, didapatkan juga hint sebagai berikut yaitu terdapat tulisan tambahan: from: r1val to: kaisar raak yang 9gwp, namun saya tidak mengerti apa maksudnya dan dengan dipecahkan di website quipqiup pun tidak diperoleh kalimat yang jelas

Jadi untuk soal ini saya tidak mendapatkan flagnya

Reverse Engineering

Hello Reverse

Untuk soal ini diberikan class java, jadi tinggal dibuka saja. Saya menggunakan intellij idea jadi ketika class dibuka langsung menampilkan source code aslinya dalam format java. Sehingga diperoleh source code berikut

```
public class soal {  
    public soal() {  
    }  
  
    static String genflag() {  
        String[] var0 = new String[]{"v3", "t0", "c0", "r5", "l", "_", "M3", "W3", "R3",  
"3"};  
        String var1 = var0[7];  
        var1 = var1 + var0[4];  
        var1 = var1 + var0[2] + var0[6];  
        var1 = var1 + var0[5] + var0[1] + var0[5];  
        var1 = var1 + var0[8] + var0[0] + var0[3] + var0[9];  
        return var1;  
    }  
  
    public static void main(String[] var0) {  
        if (var0.length != 3) {  
            System.out.println("input salah");  
        } else {  
            System.out.println("flag{[REDACTED]}");  
        }  
    }  
}
```

Karena sudah diperoleh, maka tinggal diurutkan saja seperti kodenya sehingga diperoleh flag

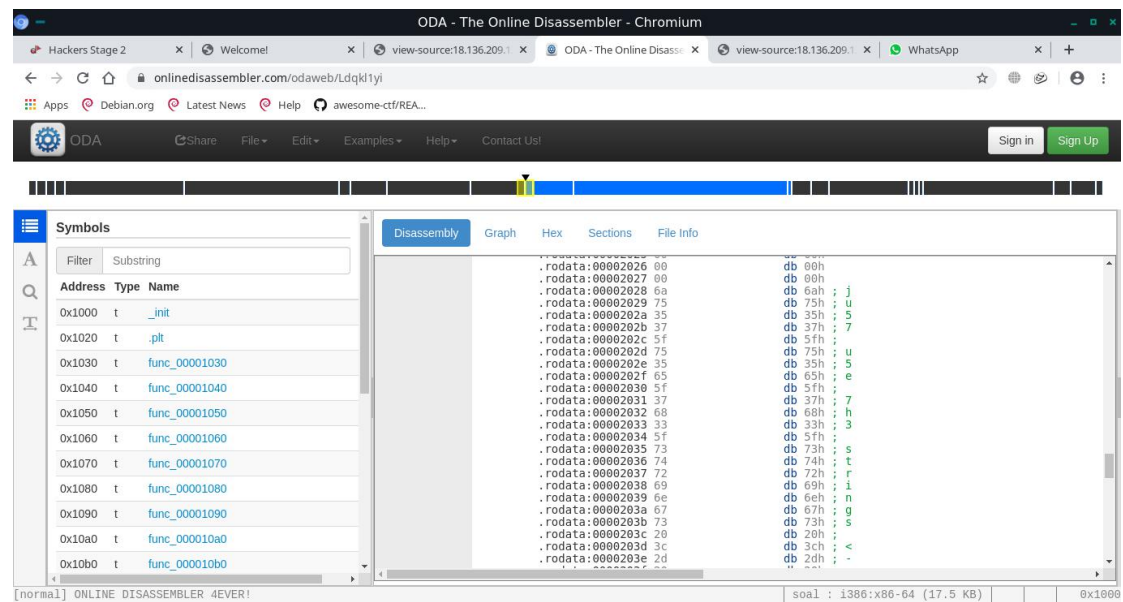
Flag{W3lc0M3_t0_R3v3r53}

Clang+Cling+Clung

Soal clang+cling+clung dapat dijawab dengan membuka filenya, dapat menggunakan

objdump --disassemble soal

Atau dengan software IDA ataupun lewat ODA, saya menggunakan ODA untuk menjawab soal ini. Maka tinggal menuju bagian main programnya dan lihat kodenya.



Maka flag untuk soal ini adalah

flag{ju57_u5e_7h3_strings}

UlaRE

Pada soal ini, kita ubah program tersebut menjadi python, beda dari file sebelumnya yang berformat ELF yang tidak bisa langsung dirubah. Maka kita tinggal merubahnya menjadi python kembali dengan tool uncompyle6

Maka akan diperoleh kode python asli sebagai berikut

```
checked2 = [9887454823508319666176L, 2367687367491881398609906326124363776L, 234187180623265792L, 62748704711297355374086808666112L, 17361742620725829882898847100829696L, 27584547717644288L, 142788163609707759784588649053552640L, 167644010141872405086208L]

checked1 = [57344, 58368, 24576, 52736, 58368, 26624, 39424]
checked0 = [57236, 58319, 24469, 52685, 58273, 26527]
checked4 = [114, 61, 71, 106]
user_input = raw_input('Masukan Serial key: ')
mentah_data = user_input.split('#')

def _print(data):
    print(data)
    exit(0)

if len(mentah_data) != 4:
    print('Siape lo?')
if len(mentah_data[1]) != len(checked1):
    print('maap ni sapa?')
if len(mentah_data[0]) != len(checked2):
    print('halo sapa nehh')
if len(mentah_data[2]) != len(checked0):
    print('LOHALO???)
if len(mentah_data[3]) != len(checked4):
    print('SAHA MANEH!!!')
for c1 in range(len(mentah_data[0])):
    if ord(mentah_data[0][c1]) << ord(mentah_data[0][c1]) != checked2[c1]:
```

```

        print('Tidak!')

for c2 in range(len(checked1)):
    if ord(mentah_data[1][c2]) << 9 != checked1[c2]:
        print('Nope!')

for c3 in range(len(mentah_data[2])):
    if ord(mentah_data[2][c3]) + checked0[c3] != checked1[c3]:
        print('PaanToehhh!')

for c4 in range(len(mentah_data[3])):
    if ord(mentah_data[3][::-1][c4]) ^ len(checked2) != checked4[c4]:
        print('NONONONONONONO!')

print("Congrats Here's your Flag: \nflag{" + user_input.replace('#', '_') + '}')

```

Untuk menjawabnya, saya mencocokkan arraynya terlebih dahulu yakni

```

mentah data 0 = checked 2 = 8
mentah data 1 = checked 1 = 7
mentah data 2 = checked 0 = 6
metanh data 3 = checked 4 = 4

```

Untuk looping C1, karena operasinya SHL dan saya tidak tau angka berapa saja yang diSHL dengan angka itu sendiri hasilnya sama dengan data yang ada pada array checked2, maka saya memSHL kan semua huruf dan angka dengan mereka sendiri, sehingga diperoleh

a-z

```

a=15370263527767281493147526365184
b=31057439705591620336669228531712
c=62748704711297355374086808666112
d=126765060022822940149670320537600
e=256065421246102339102334047485952

```

f=517201444893117595810654907793408
g=1044544094588061026833283441229824
h=2109370598779773724090514133745664
i=4259306016766850789028922770063360
j=8599741671948308259753634545270784
k=17361742620725829882898847100829696
l=3504800379511008649258085022235648
m=70745044697537026438728012485623808
n=142788163609707759784588649053552640
o=288172475648682933383442546271715328
p=581537248155900694395415588872650752
q=1173459090028871044047892170403741696
r=2367687367491881398609906326124363776
s=4776913109852041418248056622882488320
t=9636902969440640078552601187032498176
u=19439959438354394641218178256600039424
v=39212225875655018250662308278270164992
w=79089065749202494437776520086680502272
x=159507359494189904748456847233641349120
y=321673174979949641242721308587843387392
z=648663261943038945977057845416808153088

A-Z

A=2398076729582241710080
B=4869940435459321626624
C=9887454823508319666176
D=20070057552195992158208
E=40730410914750689968128
F=82641413450218791239680
G=167644010141872405086208
H=340010386766614455386112
I=689465506498968201199616
J=1397820478929414983254016
K=2833419889721787128217600

L=5742397643169488579854336
M=11635911013790805806546944
N=23574053482485268906770432
O=47752569874777852400893952
P=96714065569170333976494080
Q=195845982777569926302400512
R=396527668833598369303625728
S=802726744224113772004900864
T=1624796301562061610805100544
U=3288278229351791355200798720
V=6653927711158918977582792704
W=13462597927228510489527975936
X=27234680864278366047780732928
Y=55088331748199422233011027968
Z=111414603535684224740921180160

0-9

0=13510798882111488
1=27584547717644288
2=56294995342131200
3=114841790497947648
4=234187180623265792
5=477381560501272576
6=972777519512027136
7=1981583836043018240
8=4035225266123964416
9=8214565720323784704

_ = 3763337719427556035693337640960

Maka tinggal dicocokkan dengan data yang ada pada checked2 sehingga diperoleh substring pertama yaitu

Cr4ck1nG

Lalu untuk looping C2, dapat dijawab dengan mengSHR kan isi array checked1 dengan 9 dan diperoleh hasilnya lalu dikonversi ke Ascii

Sebelum SHR 9

57344, 58368, 24576, 52736, 58368, 26624, 39424

Setelah SHR 9, lalu konversi angkanya menjadi Ascii

112 114 48 103 114 52 77

Diperoleh substring kedua

pr0gr4M

Untuk looping C3, tinggal mengurangi nilai array checked1 dengan checked0, karena banyak arraynya selisih 1 maka array terakhir pada checked 1 tidak dihitung

checked1 = [57344, 58368, 24576, 52736, 58368, 26624, 39424]

checked0 = [57236, 58319, 24469, 52685, 58273, 26527]

Hasil = 108 49 107 51 95 97

Maka tinggal kita konvert ke ascii dan diperoleh substring ketiga

l1k3_a

Untuk looping C4 caranya adalah menXORkan nilai dari checked2 dengan checked4, lalu dicek dengan list huruf yang di awal tadi yang sama, sehingga diperoleh z 5 O b, karena terdapat [-1] pada kode pythonnya, maka tinggal kita balik sehingga diperoleh substring terakhir yaitu

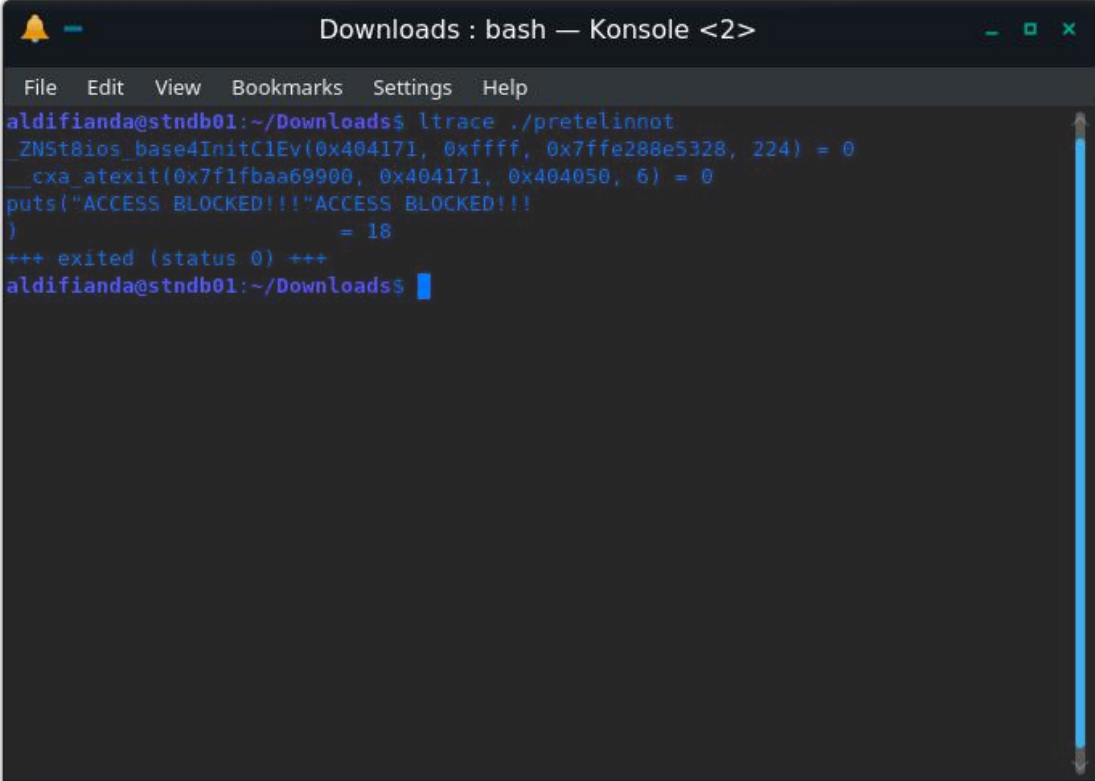
b05z

Lalu kita gabung semuanya sehingga diperoleh flag

flag{Cr4ck1nG_pr0gr4M_l1k3_a_b05z}

Tidak Dipretelin

Soal tidak dipretelin berformat ELF, jadi kemungkinan berasal dari program C ataupun CPP. Jadi tidak bisa langsung di convert seperti pyc. Untuk menjawabnya saya menggunakan tool ltrace

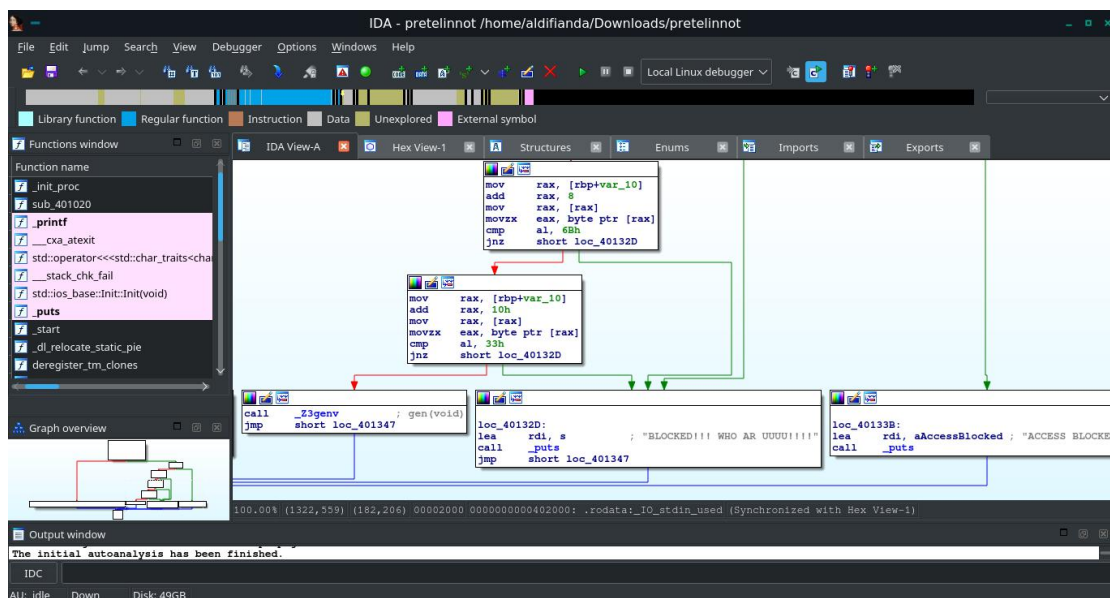


```
Downloads : bash — Konsole <2>
File Edit View Bookmarks Settings Help
aldifianda@stndb01:~/Downloads$ ltrace ./pretelinnot
_ZNSt8Bios_base4InitC1Ev(0x404171, 0xffff, 0x7ffe288e5328, 224) = 0
__cxa_atexit(0x7f1fbaa69900, 0x404171, 0x404050, 6) = 0
puts("ACCESS BLOCKED!!!")ACCESS BLOCKED!!!
) = 18
+++ exited (status 0) +++
aldifianda@stndb01:~/Downloads$
```

Maka akan diberitahukan bahwa diperlukan kode sepanjang 18 dengan 3 buah argumen, namun tetap tidak mengeluarkan flag meskipun sudah dengan 3 argumen

```
Downloads : bash — Konsole <2>
File Edit View Bookmarks Settings Help
aldifianda@stndb01:~/Downloads$ ltrace ./pretelinnot
_ZNSt8Bios_base4InitC1Ev(0x404171, 0xffff, 0x7ffe288e5328, 224) = 0
__cxa_atexit(0x7f1fbaa69900, 0x404171, 0x404050, 6) = 0
puts("ACCESS BLOCKED!!!ACCESS BLOCKED!!!") = 18
+++ exited (status 0) +++
aldifianda@stndb01:~/Downloads$ ltrace ./pretelinnot testing testing test
_ZNSt8Bios_base4InitC1Ev(0x404171, 0xffff, 0x7ffced97a580, 224) = 0
__cxa_atexit(0x7ff97290f900, 0x404171, 0x404050, 6) = 0
puts("BLOCKED!!! WHO AR UUUU!!!!"BLOCKED!!! WHO AR UUUU!!!!") = 27
+++ exited (status 0) +++
aldifianda@stndb01:~/Downloads$
```

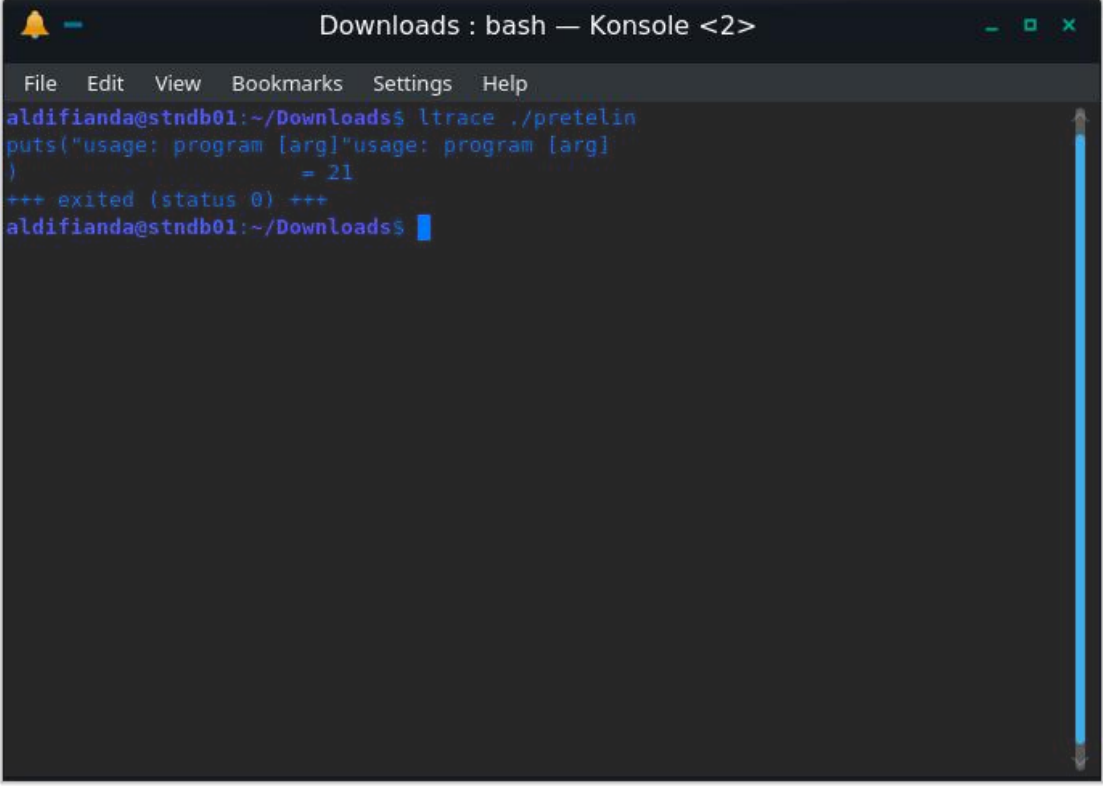
Apabila di cek dengan tool ida, apabila menginputkan argument yang benar juga tidak ditampilkan komennya seperti inputan salah



Jadi untuk soal ini saya tidak menemukan flagnya

Pretelin

Untuk soal ini saya menggunakan cara yang sama dengan tidak dipretelin. Yaitu dengan menggunakan ltrace sehingga diperoleh

A screenshot of a terminal window titled "Downloads : bash — Konsole <2>". The terminal shows the command `ltrace ./pretelin` being executed. The output of the ltrace command is displayed in blue text: `puts("usage: program [arg]"usage: program [arg]`, `)`, `= 21`, and `+++ exited (status 0) +++`. The prompt `aldifianda@stndb01:~/Downloads$` is visible at the end of the output.

```
aldifianda@stndb01:~/Downloads$ ltrace ./pretelin
puts("usage: program [arg]"usage: program [arg]
)
= 21
+++ exited (status 0) +++
aldifianda@stndb01:~/Downloads$
```

Maka untuk bisa membukanya, diperlukan argumen sepanjang 21

Binary Hacking

Welcome to Binaries

Pada soal ini, tinggal connect ke 13.250.32.243 dengan port 1001 saja menggunakan netcat yaitu dengan kode berikut

```
nc 13.250.32.243 1001
```

Sehingga nanti diperoleh flag berikut

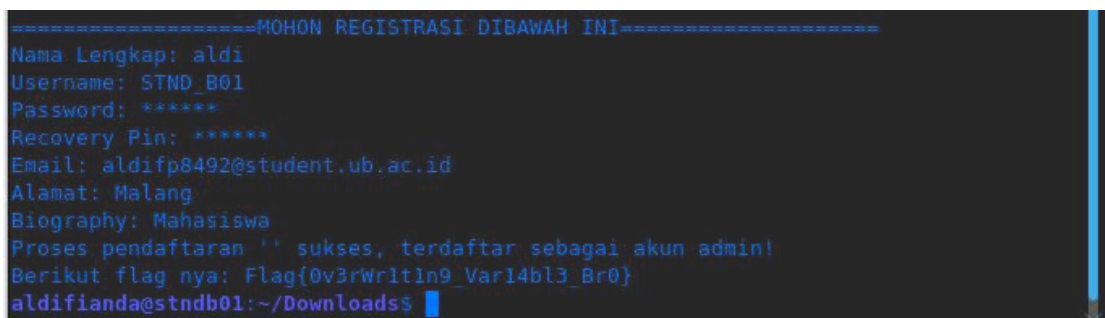
flag{WeLcom3_t0_b1n4ry_h4ckin6_1n7r0}

overflow0x01

Sama seperti soal welcome to binaries, yaitu tinggal connect ke netcat, dan tidak perlu mendownload file vuln

```
nc 13.250.32.243 1002
```

Nanti akan muncul sebuah form sebagai admin untuk memberikan privilege sebagai admin, file vuln juga memberikan form tetapi tidak akan memberikan privilege sebagai admin sehingga nanti tidak akan muncul flag

A screenshot of a netcat session on a terminal. The text is as follows:
=====MOHON REGISTRASI DIBAWAH INI=====
Nama Lengkap: aldi
Username: STND_B01
Password: *****
Recovery Pin: *****
Email: aldifp8492@student.ub.ac.id
Alamat: Malang
Biography: Mahasiswa
Proses pendaftaran '' sukses, terdaftar sebagai akun admin!
Berikut flag nya: Flag{0v3rWr1t1n9_Var14bl3_Br0}
aldifianda@stndb01:~/Downloadss

Maka flag untuk soal ini adalah

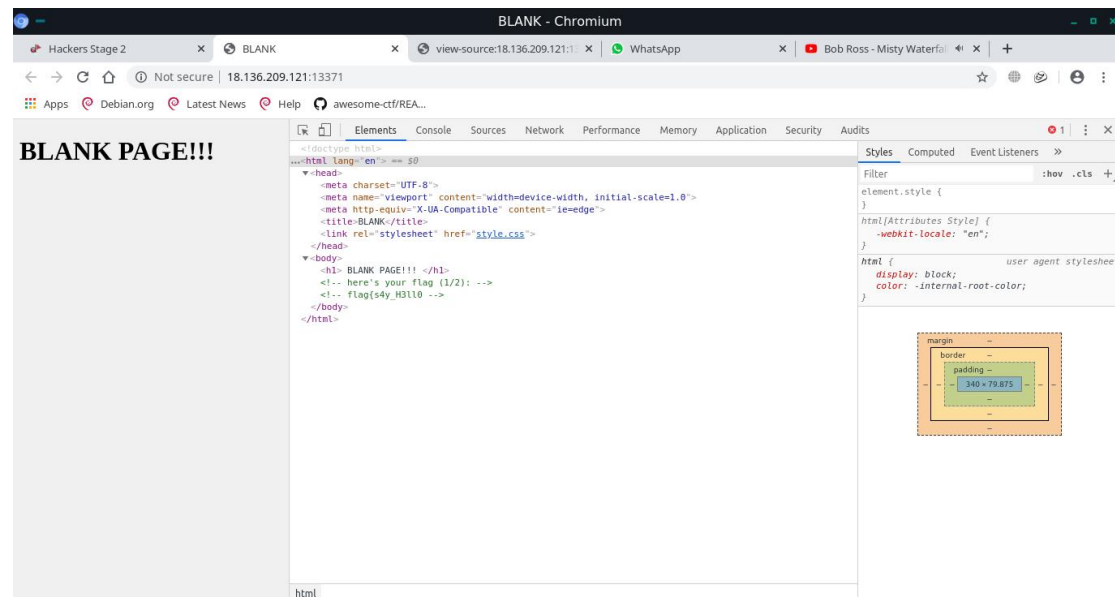
Flag{0v3rWr1t1n9_Var14bl3_Br0}

Web Exploitation

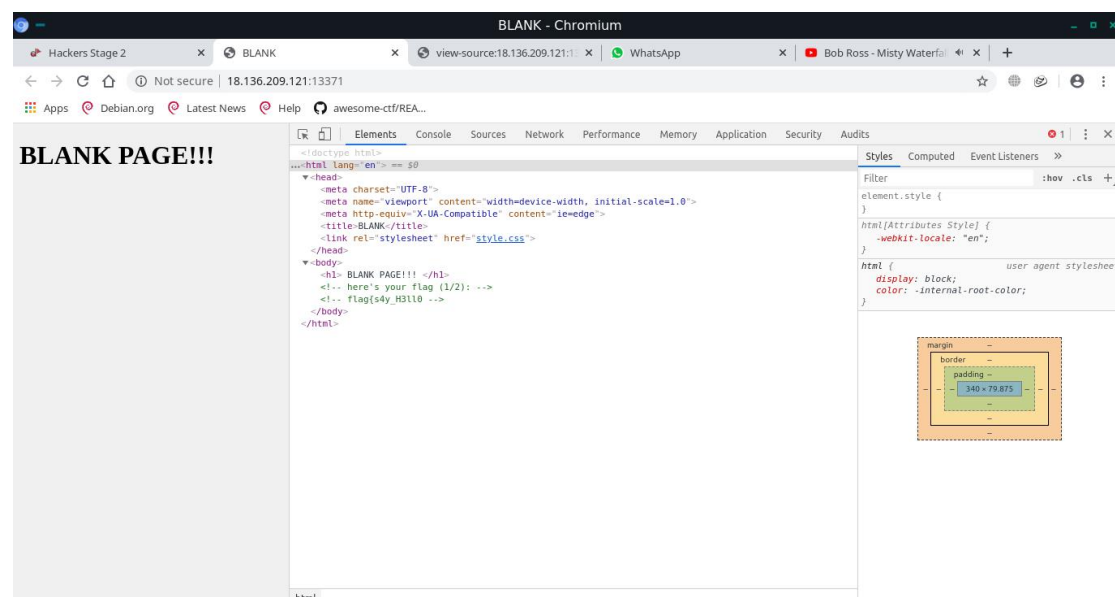
Hack Me Plz!

Untuk soal ini kita akses website <http://18.136.209.121:13371/>

Akan muncul blank page, lalu kita tinggal lihat source/ inspect element saja, maka akan terlihat setengah dari flag tersebut



Setengahnya lagi terdapat pada file cssnya



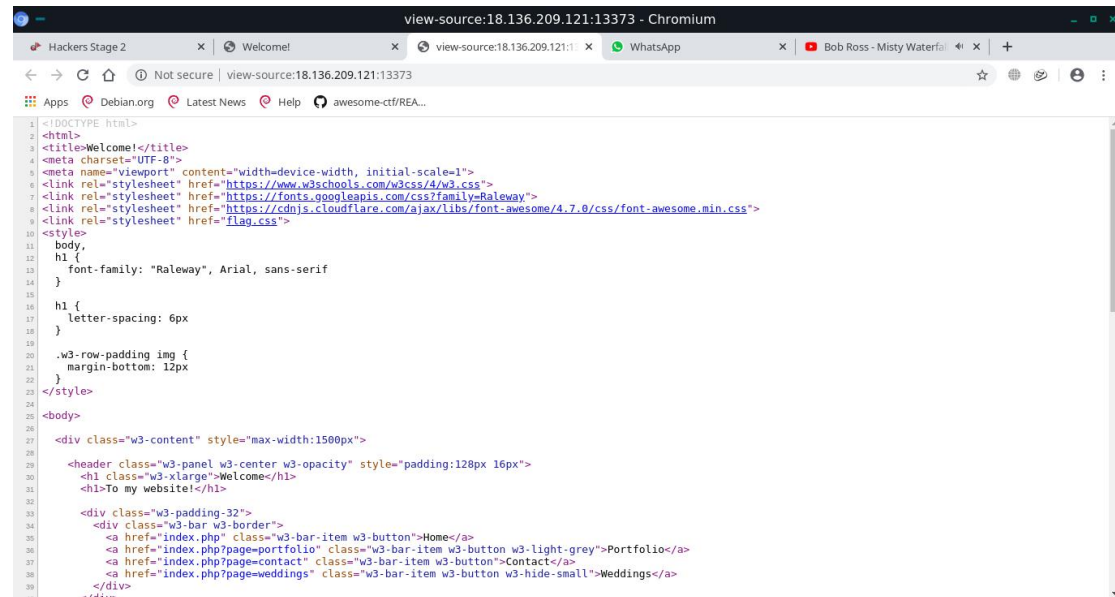
Maka flag untuk soal ini adalah

flag{s4y_H3llo_7o_d3v_t00Lszz}

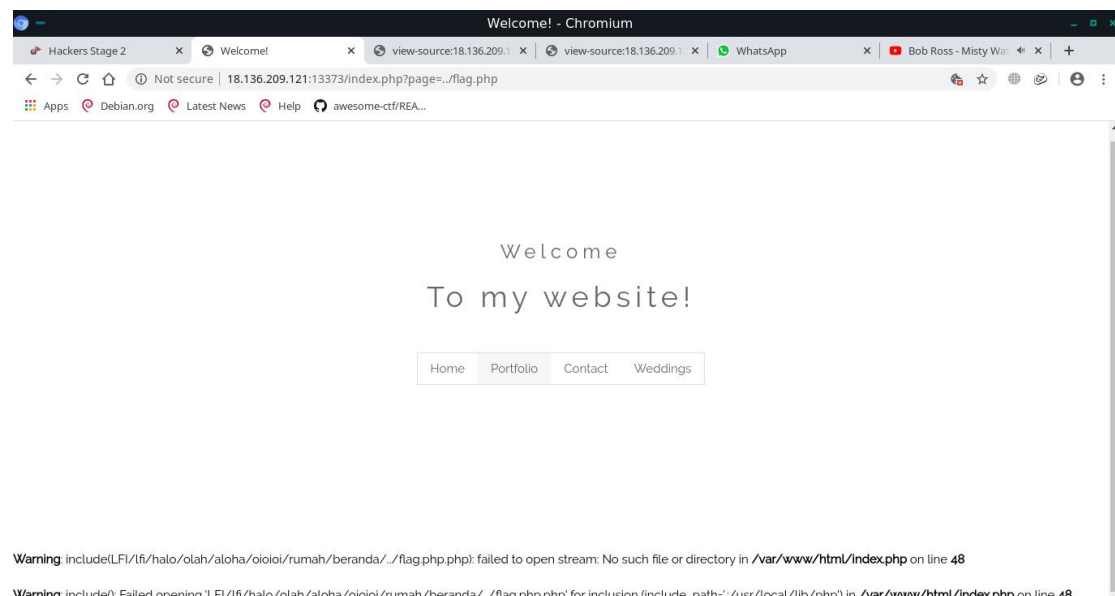
Lorem Femto Ipsum

Dari judul soalnya, diketahui bahwa teknik untuk menyelesaikannya adalah dengan LFI atau Local File Inclusion, untuk soal ini saya tidak menemukan file php nya, dengan script php pun tidak juga muncul file flag nya (karena flag.css berisi w0t_15_d1s_wowow000

Yang sudah dipastikan bukan flag yang benar)



```
1 <!DOCTYPE html>
2 <html>
3 <title>Welcome!</title>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6 <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
7 <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Raleway">
8 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
9 <link rel="stylesheet" href="flag.css">
10 <style>
11 body,
12 h1 {
13   font-family: "Raleway", Arial, sans-serif
14 }
15 h1 {
16   letter-spacing: 6px
17 }
18 .w3-row-padding img {
19   margin-bottom: 12px
20 }
21 </style>
22 </style>
23 </style>
24 <body>
25
26 <div class="w3-content" style="max-width:1500px">
27
28 <header class="w3-panel w3-center w3-opacity" style="padding:128px 16px">
29 <h1 class="w3-xlarge">Welcome</h1>
30 <h1>To my website!</h1>
31
32 <div class="w3-padding-32">
33 <div class="w3-bar w3-border">
34 <a href="index.php" class="w3-bar-item w3-button">Home</a>
35 <a href="index.php?page=portfolio" class="w3-bar-item w3-button w3-light-grey">Portfolio</a>
36 <a href="index.php?page=contact" class="w3-bar-item w3-button">Contact</a>
37 <a href="index.php?page=weddings" class="w3-bar-item w3-button w3-hide-small">Weddings</a>
38 </div>
39 </div>
40
```



Maka untuk soal ini saya tidak menemukan flagnya