Writeup Kessoku Band Kualifikasi HackToday2023

Anggota tim:
Chaerla (rach#5368)
azuketto (azuketto#3583)
Ryo (reee#9594)

Daftar Isi

Cry

Spam

Today is your birthday, your friends send you a file that has password on it. They said the password will be send from fake email. But because of your birthday, many people send you an email. Here's what you got on email.

nc 103.181.183.216 18001

Pembahasan

Berikut merupakan chall soal:

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

from Crypto.Util.number import bytes_to_long, long_to_bytes, getPrime, inverse,
GCD
from random import sample, randint, shuffle

with open('spam.txt','r') as spam:
    spam = spam.read().splitlines()
    jumlah = randint(100, 200)
    email = sample(spam, jumlah)

with open('password.txt','r') as password:
    password = password.read().splitlines()
    full_password = ''.join(password)
    email.extend(password)
    shuffle(email)
```

```
with open('flag.txt','r') as flag:
    FLAG = flag.read().strip()

for idx in enumerate(email):
    indeks = idx[0]+1
    message = idx[1]
    while True:
        p = getPrime(512)
        q = getPrime(16)
        phi = (p-1)*(q-1)
        e = 65537
        d = inverse(e,phi)
        if GCD(e,phi) == 1 and d != -1:
            break

    m = bytes_to_long(message.encode())
    n = p*q
    c = pow(m,e,n)

    print(f'n{indeks} = {n}\n')
    print(f'c{indeks} = {c}\n')

answer = input('Input Full Password = ').strip()

if answer == full_password:
    print(f"Correct Password!\nHere's Your Flag\n{FLAG}\n")
else:
    print('Wrong Password!')
```

Diberikan ratusan ciphertext dan public key dengan e yang sama (RSA), dan terlihat bahwa pemilihan prima sangat lemah. Sagemath bisa langsung memfaktorkan semua n tersebut.

Kita coba decrypt saja (sage):

```
from pwn import *
from libnum import *
```

```
e = 65537
r = remote("103.181.183.216", int(18001))
cts = []
while True:
    tex = r.recvuntil(b'=')
    if (b'Input' in tex):
        break
    n = int(r.recvline().strip().decode())
    r.recvuntil(b'=')
    c = int(r.recvline().strip().decode())
    cts.append((c, n))

email = b'happy_birthday_'
pw = b''
for pair in cts:
    n = pair[1]
    ct = pair[0]

    fac = factor(n)
    p = fac[0][0]
    q = fac[1][0]
    d = power_mod(e, -1, (p-1) * (q-1))
    if email not in n2s(power_mod(ct, d, n)):
        pw += n2s(power_mod(ct, d, n))
pw = b'1Nst1Tut_p3Rt4n14N_b0G0R'
r.sendline(pw)
r.interactive()
```

Setelah membaca decrypted text secara manual, kita dapatkan 3 string unik, yang jika disusun membentuk password 1Nst1Tut_p3Rt4n14n_b0G0R

```
[*] Switching to interactive mode
/home/azunyan/.sage/local/lib/python3.10/site-packages/pwnlib/tubes/tube.py:877: Depr
  while not go.isSet():
 /home/azunyan/.sage/local/lib/python3.10/site-packages/pwnlib/tubes/tube.py:896: Dep
  while not go.isSet():
Correct Password!
Here's Your Flag
hacktoday{H4pPy_b1Rthd4Y}

[*] Got EOF while reading in interactive
```

Kita kirim dan flag diperoleh

Flag: hacktoday{H4pPy_b1Rthd4Y}

Reverse RSA

x

Zantos was too bored with plain RSA so he tried create a cipher by reversing the RSA. is it secure??

-

Hint : LLL is one of many way to recover random key

Berikut merupakan chall soal:

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import random
import os
import hashlib

def generate_prime():
    while True:
        a,b = random.getrandbits(512), random.getrandbits(512)
        if isPrime(pow(a+b,2) - 2*a*b):
            return pow(a+b,2) - 2*a*b,a,b

def encrypt(m: str, a: int, b: int):
    key1 = hashlib.sha256(long_to_bytes(a)).digest()[:16]
    key2 = hashlib.sha256(long_to_bytes(b)).digest()[:16]
    enc = AES.new((key1+key2), AES.MODE_ECB)
```

```
    return enc.encrypt(pad(m.encode(),16)).hex()

def hehe():
    e = 3
    m = open("flag.txt", "r").read()
    while True:
        p,a1,b1 = generate_prime()
        q,a2,b2 = generate_prime()
        phi = (p-1)*(q-1)
        d = inverse(e,phi)
        if d > 1:
            break
    return [encrypt(m[:len(m)//2],a1,b1),
encrypt(m[len(m)//2:],a2,b2)],p,q,e,phi,d

def main():
    c = os.urandom(128)
    c = bytes_to_long(c)
    arr,p,q,e,phi,d = hehe()
    n = p*q
    c = pow(c,e,n)
    with open("output.txt", "w") as f:
        f.write(f"{arr = }\n")
        f.write(f"{n = }\n")
        f.write(f"hint1 = {pow(d,e,n)}\n")
        f.write(f"hint2 = {pow(phi,e,n)}\n")
        f.write(f"{c = }\n")

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

Kita diberikan encrypted flag dengan AES, dengan key dari bilangan prima yang digunakan dalam enkripsi RSA. Kita diberikan dua hint untuk mencoba memfaktorkan n, yaitu pow(d, e, n) dan pow(phi, e, n). Dari sini, langsung dicurigai kemungkinan celah Franklin-Reiter related message attack, karena jika kita menghitung nilai pow(e,e,n) * pow(d,e,n) = pow(d*e, e, n), kita memperoleh pow(k*phi + 1, e, n) untuk suatu k bulat >= 1.

Kemudian, setelah mencoba untuk mengobservasi nilai k pada nilai e = 3 (seperti pada soal), kita memperoleh nilai k hampir selalu sama dengan 2.

test.py:

```
from Crypto.Util.number import *
for i in range(10):
    p = getStrongPrime(1024)
    q = getStrongPrime(1024)
    phi = (p-1) * (q-1)
    e = 3
    while True:
        try:
            d = pow(e, -1, phi)
            break
        except:
            p = getStrongPrime(1024)
            q = getStrongPrime(1024)
            phi = (p-1) * (q-1)
    print((e*d-1)//phi)
```

Dari sini, kita langsung melakukan franklin reiter attack untuk memperoleh nilai phi, dan dari situ, menghitung p dan q dari quadratic equation yang dihasilkan phi = n - (p+q) + 1 dan n = pq.

rec_p.sage:

```
arr = ['157ac614902984894fdeebbfbb071706c567595ec75d8cfa15b4fa78daec262d',
'595c471aa4f4fd674f1e3d2d92451989241fc1e5483943462f22139c40f60d26']
n = 9404... #redacted
hint1 = 55939...#redacted
hint2 = 544...#redacted
c = 799169293...#redacted
e = 3

def gcd(a, b):
    while b:
        a, b = b, a % b
    return a.monic()
```

```
def franklinreiter(C1, C2, e, N, a, b):
    P.<X> = PolynomialRing(Zmod(N))
    g1 = (a*X + b)^e - C1
    g2 = X^e - C2
    print("Result")
    result = -gcd(g1, g2).coefficients()[0]
    return result

hint1_new = (hint1 * (power_mod(e, e, n))) % n

phi = franklinreiter(hint1_new, hint2, e, n, 2, 1)
d = pow(int(e), int(-1), int(phi))
assert(power_mod(d, e, n) == hint1)
# phi = n - (p + q) + 1
pplusq = n - int(phi) + 1
q = (pplusq + int((pplusq ** 2 - 4 * n) ** (1/2)))//2
assert(is_prime(q))
assert(n%q == 0)
p = n // q
print(p, q)
```

Kemudian, kita cari nilai a, b sehingga a**2 + b**2 = p atau q dengan menggunakan equation sagemath, dan dari situ, bisa langsung mendekripsi flag.

solve.sage:

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
import hashlib

def decrypt(m: str, a: int, b: int):
    ct = bytes.fromhex(m)
    key1 = hashlib.sha256(long_to_bytes(a)).digest()[:16]
    key2 = hashlib.sha256(long_to_bytes(b)).digest()[:16]
    enc = AES.new((key1+key2), AES.MODE_ECB)
```

```
    return unpad(enc.decrypt(ct), 16)
flag = b''
p =
74468804622926887174397400355580273413063467295804798537637200405797367975378842234798486847015213579016860751524831726412426588519437837073724062647486353051230727916122248164905460294114313479122336553929506188583028165235277317095703187612912055510263915344332628112192496130308319656162753951786159497573
q =
12629077989656144391628070957004170554107242784146249264419983746077068872460293248318559335565888212064470483601495696268641217182776902007275183178324525189666042478680114799673638510403753064279686441324281784730782730178251579709887299854852183128441655637610003316080752538628846606340539475728227436879 3
arr = ['157ac614902984894fdeebbfbb071706c567595ec75d8cfa15b4fa78daec262d',
'595c471aa4f4fd674f1e3d2d92451989241fc1e5483943462f22139c40f60d26']

for num in [p, q]:
    x, y = var('x y')
    assume(x,"integer")
    assume(x > 0)
    assume(y, "integer")
    assume(y>0)

    soln = (solve([x ** 2 + y ** 2 == num], x, y))
    for sol in soln:
        (a, b) = sol
        for c in arr:
            try:
                flag += (decrypt(c, int(a), int(b)))
            except:
                continue

print(flag)
```

Flag: hacktoday{R3v3rS3_RS4_W1th_S0mE_Alg3brSa_1s_Aw3S0mE!!!}

AES Enjoyer

Legenda mengatakan, "CTF gak afdhol kalo di crypto gak ada aes-nya"
nc 103.181.183.216 18002

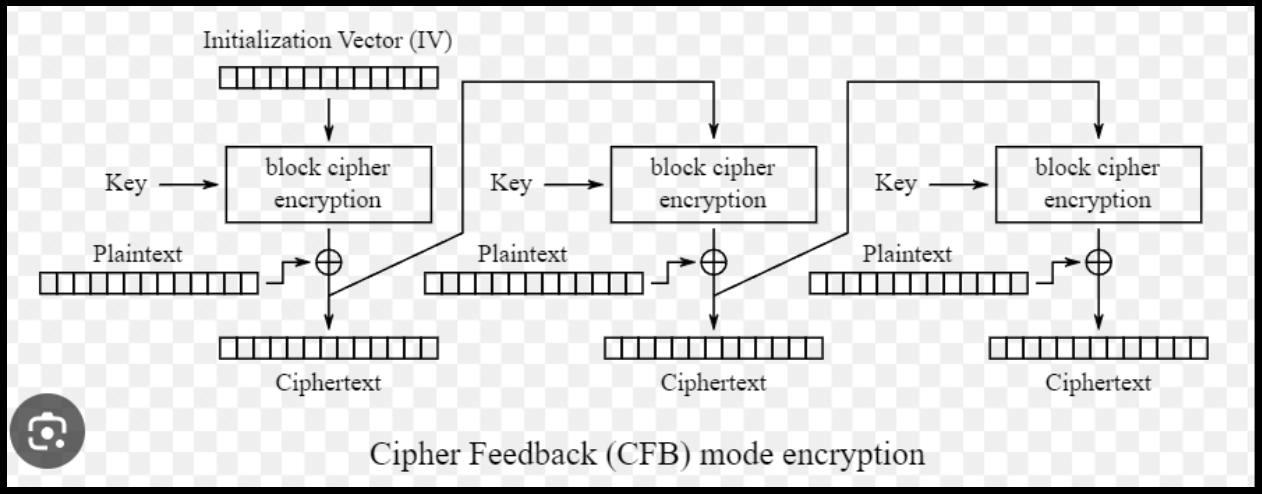
Berikut merupakan chall soal:

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

import os
import sys
from Crypto.Cipher import AES
from Crypto.Util.number import bytes_to_long
from Crypto.Util.Padding import pad

class Unbuffered(object):
  def __init__(self, stream):
    self.stream = stream
  def write(self, data):
    self.stream.write(data)
    self.stream.flush()
  def writelines(self, datas):
    self.stream.writelines(datas)
    self.stream.flush()
  def __getattr__(self, attr):
    return getattr(self.stream, attr)

sys.stdout = Unbuffered(sys.stdout)

with open("flag.txt", "rb") as f:
    flag = f.read()
    f.close()
```

```python
def encrypt(pt: bytes, iv: bytes, key: bytes):
    aes = AES.new(key, AES.MODE_CFB, iv=iv, segment_size=128)
    pt = pt + flag[:len(flag)//2]
    pt = pad(pt, 16)
    ct = aes.encrypt(pt)
    return iv+ct

def generate_key():
    return bin(bytes_to_long(os.urandom(2)))[5:].zfill(16)

def gift(pt : bytes):
    key = generate_key()
    iv = b"hektoday"*2
    assert len(key) == 16 and len(iv) == 16
    print(key)
    aes = AES.new(key.encode(), AES.MODE_CBC, iv=iv)
    return aes.encrypt(pt)

def menu():
    print(
"""[1] Encrypt
[2] Flag
[3] Exit""")


def main():
    key = os.urandom(16)
    for _ in range(4):
        menu()
        op = input("[>] ")
        if op == "1":
            iv = bytes.fromhex(input("[>] IV (hex): "))
            pt = bytes.fromhex(input("[>] Plaintext (hex): "))
            iv = pad(iv, 16) if len(iv) < 0x10 else iv
            pt = pad(pt, 16)
            ct = encrypt(pt, iv, key)
            print("[*] Ciphertext:", ct.hex())
```

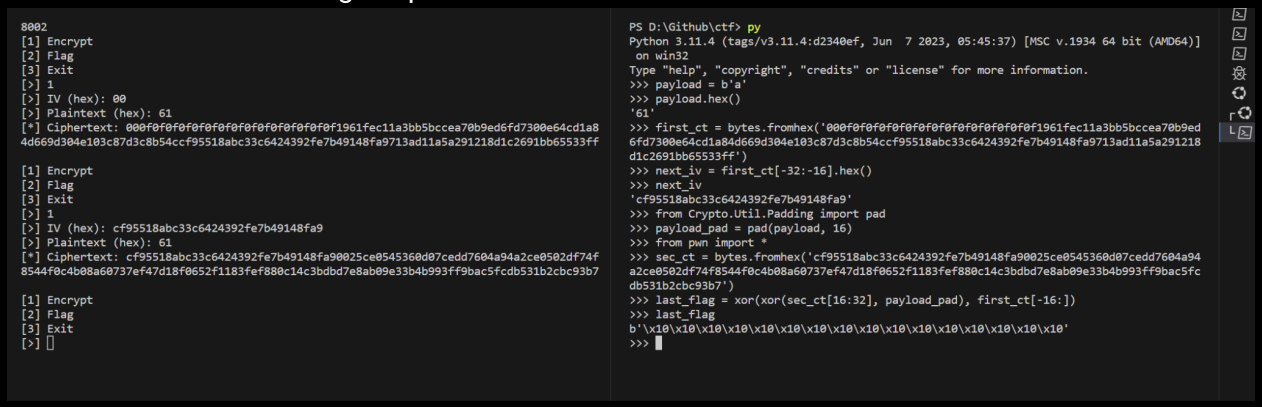
```
        elif op == "2":
            print("[*] Encrypted Flag:", gift(gift(flag[16:])).hex())
        elif op == "3":
            break
        else:
            print("[?] Are you drunk?")
        print()

    return 0


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

Dari chall soal, sepertinya ada dua bagian yang bisa diserang, yaitu padded encryption CFB, dan dobel enkripsi flag menggunakan key lemah (vuln to meet in the middle attack). Sepertinya kita belum bisa melakukan MITM karena kita tidak memiliki 16 byte header flag (`hacktoday{` kurang panjang), sehingga kita coba cari header flag dari tempat lain terlebih dahulu.

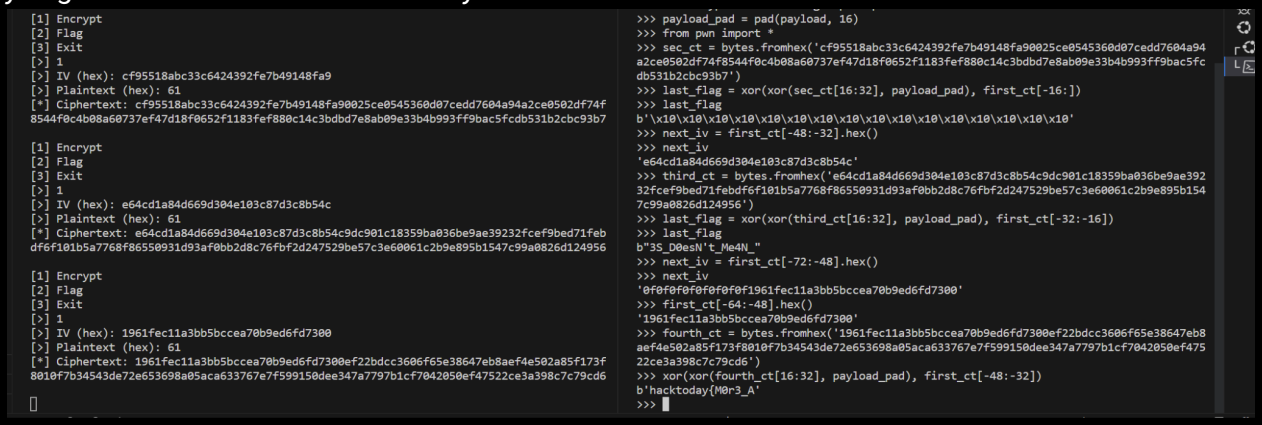


Kita pertama serang bagian padded encryption CFB secara manual. Pada dasarnya karena kita bisa mengatur IV pada CFB, dan sebenarnya kita memiliki blok yang difeed ke block flag. Jadi, kita tinggal melakukan enkripti satu kali untuk melihat blok ciphertext yang akan difeed ke blok-blok flag, kemudian memasukkannya sebagai IV di enkripsi berikutnya, dan melihat hasil enkripsi IV dengan melakukan xor ciphertext block pertama dan plaintext block pertama. Hasil tersebut akan kemudian menjadi key xor untuk dekripsi flag.

Kita ambil sembarang ciphertext:



Kemudian, kita masukkan IV dari block kedua terakhir untuk mendecrypt block terakhir, dan ternyata seluruhnya padding. Kita lakukan proses yang sama untuk blok lainnya:



Diperoleh partial flag: hacktoday{M0r3_A3S_D0esN't_Me4N_, yang berukuran 32 byte. Kita bisa menggunakan ini sebagai payload untuk attack meet in the middle pada weak double encryption flag dari command get_flag.

solve.py:

```
from pwn import *
from Crypto.Util.number import *
from libnum import *
from Crypto.Cipher import AES
from Crypto.Util.number import bytes_to_long
from Crypto.Util.Padding import pad
from azunyan.math import find_in_sorted_list

def generate_key(c: bytes):
```

```
    return bin(bytes_to_long(c))[5:].zfill(16)

enc =
bytes.fromhex('c78a3e31ffaca3ea927e1902f6cdff45e819e67ed16ee0c2f9eb9208d2f18a15
aa5b4e1b101e46dcbd307f36d369c0cd')
known = b"hacktoday{M0r3_A3S_D0esN't_Me4N_"[16:]
assert(len(known) == 16)
mp = {}
head = []

for i in range(256):
    for j in range(256):
        c = n2s(i) + n2s(j)
        key = generate_key(c)
        iv = b"hektoday"*2
        assert len(key) == 16 and len(iv) == 16
        aes = AES.new(key.encode(), AES.MODE_CBC, iv=iv)
        mid = aes.encrypt(known)
        mp[mid] = c
        head.append(mid)

head.sort()

for i in range(256):
    for j in range(256):
        c = n2s(i) + n2s(j)
        key = generate_key(c)
        iv = b"hektoday"*2
        assert len(key) == 16 and len(iv) == 16
        aes = AES.new(key.encode(), AES.MODE_CBC, iv=iv)
        mid = aes.decrypt(enc)
        idx = find_in_sorted_list(mid[:16], head)
        if idx != -1:
            key = mp[head[idx]]
            key = generate_key(key)
            aes = AES.new(key.encode(), AES.MODE_CBC, iv=iv)
            pt = aes.decrypt(mid)
            print(pt)
```

```
            exit()
```

```
PS D:\Github\ctf>  d:; cd 'd:\Github\ctf'; & 'C:\Users\LEGION\Ap
thonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '61
b"3S_D0esN't_Me4N_M0r3_S3cuR3!!_I_Th1nK_p4dp4dp4d}"
PS D:\Github\ctf>
```

Catatan: find_in_sorted_list merupakan fungsi dari library pribadi penulis, yang berfungsi melakukan binary search.

Flag: hacktoday{M0r3_A3S_D0esN't_Me4N_M0r3_S3cuR3!!_I_Th1nK_p4dp4dp4d}

Lo Lo Lo Gak Bahaya Ta?

Bahaya Gak Sih?.
Hint : Is private key vector is the shortest basis?

Berikut merupakan chall soal:

```python
from Crypto.Util.number import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import os, random, hashlib

with open("flag.txt","rb") as f:
    FLAG = f.read().rstrip()
    f.close()

NBIT = 2048
e = 65537

def get_factor(NBIT: int) -> tuple:
    p, q = getStrongPrime(NBIT//2), getStrongPrime(NBIT//2)
```

```
    return (p*q, p, q)

def encrypt(m: int,n: int) -> int:
    return pow(m, e, n)

def encryptFlag(plain: bytes, key: int) -> str:
    IV = os.urandom(16)
    cipher = AES.new(hashlib.sha256(str(key).encode()).digest()[:16],
AES.MODE_CBC, iv=IV)
    return (cipher.iv + cipher.encrypt(pad(plain,16))).hex()

def main():
    sizeSlice = len(FLAG) // 4

    sliceFLAG = [FLAG[i*sizeSlice:(i+1)*sizeSlice] for i in range(4)]
    list_pub = [random.getrandbits(1024) for _ in range(3)]
    list_priv = [random.getrandbits(256) for _ in range(3)]
    last_priv = random.getrandbits(512)
    S = sum([i*j for i, j in zip(list_pub, list_priv)]) - last_priv

    (n, p, q) = get_factor(NBIT)
    a = random.randint(1,1000)
    b = random.randint(1,1000)
    hint_1 = a * p - b * q
    enc_pub_1 = encrypt(list_pub[0], n)
    list_pub = list_pub[1:]

    LIST_ENC_FLAG = [encryptFlag(plain, key) for plain, key in zip(sliceFLAG,
list_priv + [last_priv])]

    with open("output.txt","w") as f:
        f.write(f"{LIST_ENC_FLAG = }\n")
        f.write(f"{list_pub = }\n")
        f.write(f"{n = }\n")
        f.write(f"{hint_1 = }\n")
        f.write(f"{enc_pub_1 = }\n")
        f.write(f"{S = }")
        f.close()
```

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

Dilakukan enkripsi pada flag menggunakan AES key dari priv, dan ada beberapa hint dari public, dimana salah satunya encrypted menggunakan RSA. Pertama, kita akan solve bagian RSA.

Diberikan hint a*b - p*q, dan kita tau n = p *q, jadi kita tinggal bruteforce semua nilai a dan b (1000 * 1000 kali), dan cek apakah solusi quadratic equation yang dihasilkan benar prima.

rec_p.sage:

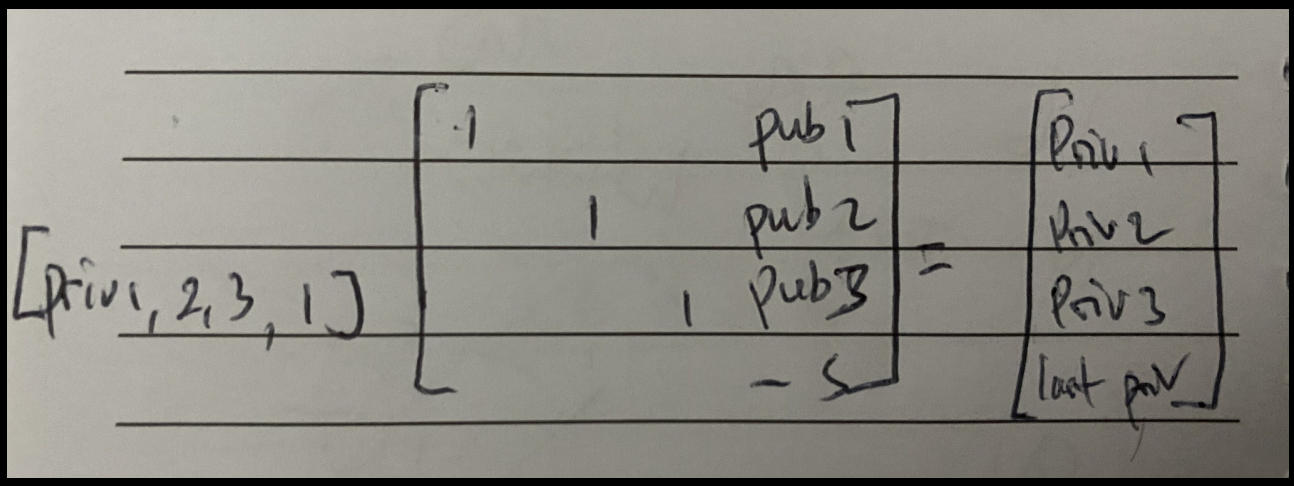
```
from tqdm import tqdm
LIST_ENC_FLAG =
['239e377818228c060ab188496a2e7f77b1ec38eeff349afb7735e8562a8bfb5b',
'd1241bf60201fc71555ca707376d6338235528ad0e28dbb94c35d9d405a97bed',
'c8d47782567ef3dad154862d17db0dbf73befe92dd132d3d25732fe71250e66c',
'b9ce3a4342c3e3e54efbdee828592ac150cbf8bba6d62b2651b72dff714c8ed0']
list_pub =
[116132014000574970780570903040217208911794320865265657170613600983470409932919
830200520251430101276696937898077350978028796991533542843684485290087816165224 5
157822632054874950107527331790775786305531664230957993869300578653829300618564 1
256620211227607195020292883627163775258244792372350180950943976346726972 6,
835094316560602771757266316927517124360418529668889517055528458273213948472112 2
155658703541168241499732580007826345375106003594596212492308752874171447409924 2
255408202045054024606382599284102887556148511907680936715648903771899109605565 4
748471445210593443702526067126366717267896897340998405700781928829795 37]
n =
214947065061128149952265691810908522804223009171768776386352956376737596313291 5
469064282696589461460415605353861643984560605961452786024506827743242338038239 6
247506224840091250264795793963597982629753616509755553000555775865926855812974 8
921982136515130484042079946306183954582292949289633586230603720023533961150968 3
981934886257545052463022510852285691679718026224521229813232658163704268123859 7
678451521277161418040781182375793321316654017168907386657102122550308282176667 6
363632323602745926148650242130545451685201098010084457128221445444182625219079 3
471941574002225280223340297665703935633823413981043420825839187 9
```

```
hint_1 =
-12725665031192943029428959168909222671848504795640878911533635431080110462397675704345145381637594923572807513686286966203734577213074466616190990790834686274166067773495604225301402414259563858944140502228123924366317249254358296455812627820008650063252492561663830845787479318454323715747217214715807134215506
enc_pub_1 =
16069120540708758665775471206808262045684214488817908051921134338076093858359110201479218439833113175260648133283996173844502588194497061367984923256245442177266598457151340361415096459946116313607970196111166072923147328483447655412954553953286147260344806678107993012767621897081000400982634197535111732715472640893416932410197061453856539429044124498417372798386628073274579316345085603620456586993105998953719806759366174567064640546414233054872060916311881700933997337512856672608264803359635119489786500730230512806871162805362632343966353353748002480803399281419922246583366138119486779651680047080340593379967
S =
10426898420426810240569695316876227128971175509482864539898388868035265237500111319570743203059477808496968350951669657087379832787623948039178629387468793556748170643254161248083144094338727826317944523958864159816411945022004758108836141332602962869135439690333562721651219115117716886792223916085240356669593984620412320618909317646689958901011337366289570705948553072541119031047551
def recover_p():
    for a in tqdm(range(1, 1001)):
        for b in range(1, 1001):
            num = hint_1**2 + 4*a*b*n
            c = int(num ** (1/2))
            c -= hint_1
            if c % (2*b) == 0 and is_prime(c // (2 * b)):
                return (c // (2 * b))

p = recover_p()
assert(n % p == 0)
q = n // p
print(p, q)
```

Dari nilai p tersebut, kita bisa dekripsi public dan mendapatkan array public lengkap. Kemudian, public tersebut diapakan? Kita lihat sesuatu yang sus: bit length priv dan last_priv jauh lebih pendek dari pub:

```
list_pub = [random.getrandbits(1024) for _ in range(3)]
list_priv = [random.getrandbits(256) for _ in range(3)]
last_priv = random.getrandbits(512)
S = sum([i*j for i, j in zip(list_pub, list_priv)]) - last_priv
```

Sehingga, bisa langsung modelkan menjadi CVP saja. Kita bentuk matrix sbb:



Matrix tersebut dengan linear combination [priv_1, priv_2, priv_3, 1] akan membentuk vektor “pendek” [priv_1, priv_2, priv_3, last_priv]. Namun, perhatikan bahwa priv_i memiliki size 256 bits, sedangkan last_priv memiliki size 512 bits. Agar hasil LLL lebih mungkin keluar dengan benar, kita lakukan scaling pada M[i][i] for i in range(3), dengan mengalikannya dengan 2**256 agar bit size pada vector hasil sama.

Berikut solver (solve.sage):

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad,unpad
import os, random, hashlib

p =
167908195847942274599392266903301613322757769528461234822354500979591372654788713594991806920781568964917057076203786062708212158720100253111915670605808713109132199954001552838611735984981590822310613171365875098306948139832539371137942788944569538623715966303100385175652258967053527134565151906295315944283
q =
128014635602293227561678615848279306246698021289136306912309893278398930824377
```

```
335083382139852453860669799202813914568739466332836871779342610889414656454090120127705848446821563407118648450561211416089001554004599171709515366103721262152478620396944002206259505135482541618121295804797717077966801001366414131
LIST_ENC_FLAG =
['239e377818228c060ab188496a2e7f77b1ec38eeff349afb7735e8562a8bfb5b',
'd1241bf60201fc71555ca707376d6338235528ad0e28dbb94c35d9d405a97bed',
'c8d47782567ef3dad154862d17db0dbf73befe92dd132d3d25732fe71250e66c',
'b9ce3a4342c3e3e54efbdee828592ac150cbf8bba6d62b2651b72dff714c8ed0']
list_pub =
[116132014000574970780570903040217208911794320865265657170613600983470409932919830200520251430101276696937898077350978028796991533542843684485290087816165224515782263205487495010752733179077578630553166423095799386930057865382930061856412566202112276071950202928836271637752582447923723501809509439763467269726,
835094316560602771757266316927517124360418529668889517055528458273213948472112215565870354116824149973258000782634537510600359459621249230875287417144740992422554082020450540246063825992841028875561485119076809367156489037718991096055654748471445210593443702526067126366717267896897340998405700781928829795375]
n =
214947065061128149952265691810908522804223009171768776386352956376737596313291546906428269658946146041560535386164398456060596145278602450682774324233803823962475062248400912502647957939635979826297536165097555530005557758659268558129748921982136515130484042079946306183954582292949289633586230603720023533961150968398193488625754505246302251085228569167971802622452122981323265816370426812385976784515212771614180407811823757933213166540171689073866571021225503082821766676363632323602745926148650242130545451685201098010084457128221445444182625219079347194157400222528022334029766570393563382341398104342082583918791879
hint_1 =
-127256650311929430294289591689092226718485047956408789115336354310801104623976757043451453816375949235728075136862869662037345772130744666161909907908346862741660677734956042253014024142595638589441405022281239243663172492543582964558126278200086500632524925616638308457874793184543237157472172147158071342155064
enc_pub_1 =
160691205407087586657754712068082620456842144888179080519211343380760938583591102014792184398331131752606481332839961738445025881944970613679849232562454421772665984571513403614150964599461163136079701961111660729231473284834476554129545539532861472603448066781079930127676218970810004009826341975351117327154726408934169324101970614538565394290441244984173727983866280732745793163450856036204565869931059989537198067593661745670646405464142330548720609163118817009339973375
```

```
128566726082648033596351194897865007302305128068711628053626323439663533537480024808033992814199222465833661381194867796516800470803405933799675
S =
104268984204268102405696953168762271289711755094828645398983888680352652375001113195707432030594778084969683509516696570873798327876239480391786293874687935567481706432541612480831440943387278263179445239588641598164119450220047581088361413326029628691354396903335627216512191151177168867922239160852403566695939846204123206189093176466899589010113373662895707059485530725411190310475 51

def encryptFlag(plain: bytes, key: int) -> str:
    IV = os.urandom(16)
    cipher = AES.new(hashlib.sha256(str(key).encode()).digest()[:16],
AES.MODE_CBC, iv=IV)
    return (cipher.iv + cipher.encrypt(pad(plain,16))).hex()

def decryptFlag(ct: str, key: int) -> bytes:
    ct = bytes.fromhex(ct)
    iv = ct[:16]
    ct = ct[16:]
    cipher = AES.new(hashlib.sha256(str(key).encode()).digest()[:16],
AES.MODE_CBC, iv=iv)
    return unpad(cipher.decrypt(ct), 16)

assert(n % p == 0)
assert(q == n // p)
e = 0x10001
d = e.inverse_mod((p-1)*(q-1))
pub_1 = power_mod(enc_pub_1, d, n)
pub = [pub_1] + list_pub

m = [[0 for i in range(4)] for j in range(4)]
for i in range(3):
    m[i][i] = 2 ** 256
    m[i][3] = pub[i]
m[3][3] = S * -1

M = Matrix(ZZ, m)
M = M.LLL()
```

```python
for row in M:
    arr = row
    if arr[0] < 0:
        arr = [c * -1 for c in row]
    for i in range(3):
        assert(arr[i] % (2**256) == 0)
        arr[i] //= (2**256)
    f = True
    flag = b''
    for c in zip(LIST_ENC_FLAG, arr):
        try:
            flag += decryptFlag(c[0], c[1])
        except:
            f = False
    if f:
        print(flag)
```

Setelah memperoleh priv, kita dapat dengan mudah mendecrypt flag.

Flag: hacktoday{LoLoLo_LLL_Ga_Bahaya_Ta?_afd213456781aefcd__ZafiN}

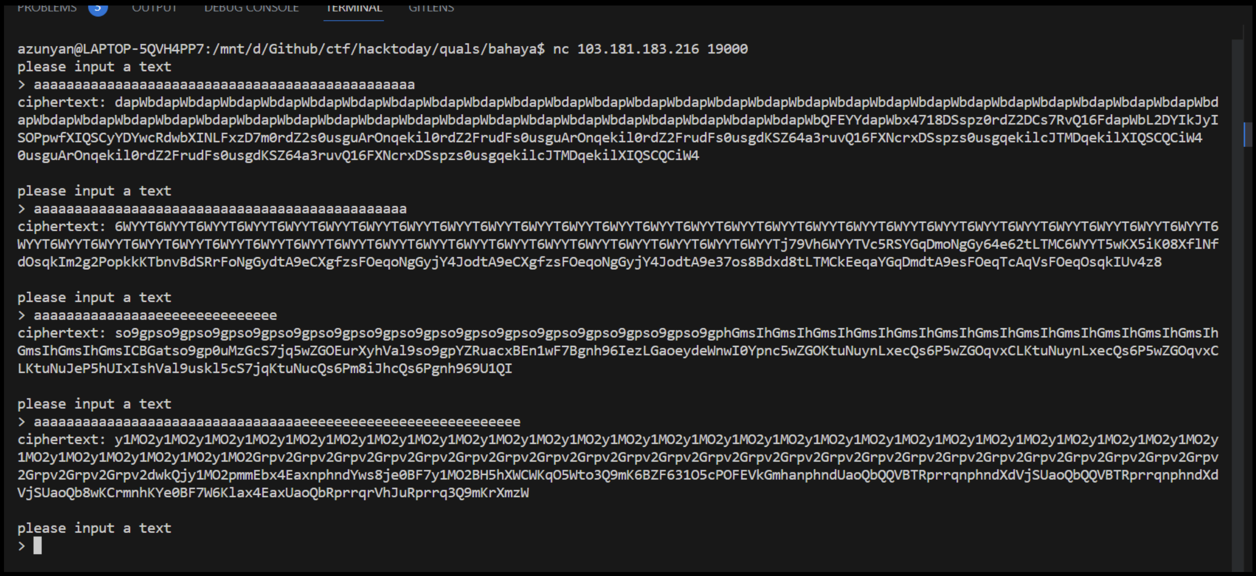
Unknown Cipher

x

Unknown Description
nc 103.181.183.216 19000

Soal blackbox :D

Kita coba-coba enkripsi sesuatu, dan nampaknya tiap karakter selalu diencrypt menjadi 5 karakter yang sama, tetapi enkripsi karakter berbeda tiap iterasi (yes, sangat dukun).



Hal tersebut dapat diverifikasi dari perbedaan panjang flag dan kesamaan tiap blok 5 karakter. Namun, tampaknya ada padding yang ditambahkan dibelakang.

Kita coba saja menguji hipotesis. Kita kirim saja semua karakter printable, membuat dictionary, dan coba decrypt padding pada text (yang sangat dicurigai merupakan flag).

Berikut solver (solve.py):

```
from pwn import *
import string
printable = string.printable.split('\t')[0]
r = remote("103.181.183.216", 19000)
r.recvuntil(b'>')
r.sendline(printable.encode())
r.recvuntil(b'ciphertext: ')
ct = r.recvline().strip().decode()
assert(len(ct) % 5 == 0)
i = 0
mp = {}
for c in printable:
    mp[ct[i:i+5]] = c
    i += 5
flag = ""
while i < len(ct):
```

```
    flag += mp[ct[i: i+5]]
    i += 5
print(flag)
```

Dan ternyata flag diperoleh! Dukun success~

```
PS D:\Github\ctf>  d:; cd 'd:\Github\ctf'; & 'C:\Users\LEGION
s\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\a
e.py'
[x] Opening connection to 103.181.183.216 on port 19000
[x] Opening connection to 103.181.183.216 on port 19000: Tryi
[+] Opening connection to 103.181.183.216 on port 19000: Done
hacktoday{3NCrYpt_S4tU_S4tU_T1dAk_4m4N}
[*] Closed connection to 103.181.183.216 port 19000
PS D:\Github\ctf>
```

Flag: hacktoday{3NCrYpt_S4tU_S4tU_T1dAk_4m4N}

DejaVu

have you ever seen LCG used as key generator in stream cipher?
if you haven't, this is the time.
-
https://youtu.be/oigcRpBOoZk
-
nc 103.181.183.216 18000
-
hint :
+ did u have enough sleep?
+ n = getPrime(105)

Berikut merupakan chall soal:

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

```python
import time
from secret import get_all_key

import sys


class Unbuffered(object):
    def __init__(self, stream):
        self.stream = stream

    def write(self, data):
        self.stream.write(data)
        self.stream.flush()

    def writelines(self, datas):
        self.stream.writelines(datas)
        self.stream.flush()

    def __getattr__(self, attr):
        return getattr(self.stream, attr)


sys.stdout = Unbuffered(sys.stdout)


def bytes2bin(msg: bytes):
    return bin(int.from_bytes(msg, "big"))[2:]


def bin2bytes(msg: bytes):
    return int(msg, 2).to_bytes((int(msg, 2).bit_length() + 7) >> 3, "big") or
b"\x00"


class KeyGen:
    def __init__(self, x: int, m: int, c: int, n: int):
        self.m = m
        self.c = c
```

```
        self.n = n
        self.state = x % n
        self.bitstate = bin(self.state)[2:]

    def update_state(self, isflag=0):
        self.state = (self.state * self.m + self.c) % self.n
        self.bitstate = bin(self.state)[2:]
        if isflag:
            return
        time.sleep(1)

    def get_bit(self, isflag=0):
        b = self.bitstate[-1]
        self.bitstate = self.bitstate[:-1]
        if not self.bitstate.isdigit():
            self.update_state(isflag)
        return int(b)


class StreamCipher:
    def __init__(self):
        m, c, n, x = get_all_key()
        self.keygen = KeyGen(x, m, c, n)

    def encrypt(self, msg: bytes, isflag=0):
        return bin2bytes(
            "".join([str(int(b) ^ self.keygen.get_bit(isflag)) for b in
bytes2bin(msg)])
        )


def menu():
    print("""[1] Encrypt a message\n[2] Get an encrypted flag\n[3] Exit""")


def main():
    cipher = StreamCipher()
    with open("flag.txt", "rb") as f:
```

```
        fl4g = f.read()
        f.close()
    enc_fl4g = cipher.encrypt(fl4g, 1)
    print("Welcome!")
    start = time.time()
    while time.time() - start  <= 45:
        menu()
        opcode = input("[>] ").strip()
        if opcode == "3":
            break
        elif opcode == "1":
            msg = input("Message to encrypt : ").strip().encode()
            ct = cipher.encrypt(msg)
        elif opcode == "2":
            msg = "flag"
            ct = enc_fl4g
        else:
            print("Maksud?")
            continue
        if len(msg) == 0:
            print("Invalid message.")
            continue
        print("Encrypted message :", ct.hex())
    return 0


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

Chall tampak straightforward, pertama menakut-nakuti dengan kemiripan dengan truncated LCG, tapi ternyata semua bit state LCG diberikan, tetapi satu per satu wkwk. Tapi ternyata, bit diberikan dalam satu continuous stream, kita tidak bisa membedakan antara bit dari satu state dengan state lainnya (kecuali mungkin jika melakukan timing attack dengan mengirimkan enkripsi satu-per-satu bit wkwk, tapi lagi malas). Jadi, kita coba tebak saja bit length dari state (atau dari modulo lcg), dengan menaikkan ukuran payload sampai menu timeout. Kita

juga dapat membantu aproksimasi dengan mengukur waktu respon di lokal.

tebak_panjang.py

```python
from pwn import *
import time

def bytes2bin(msg: bytes):
    return bin(int.from_bytes(msg, "big"))[2:]

payload = b'a' * 500
start = time.time()
r = remote("103.181.183.216", 18000, level = 'debug')

r.recvuntil(b'[>] ')
r.sendline(b'1')
r.recvuntil(b'Message to encrypt : ')
r.sendline(payload)
r.recvuntil(b'Encrypted message :')
print(time.time() - start)
state = bytes.fromhex(r.recvline().strip().decode())

r.recvuntil(b'[>] ')
r.sendline(b'2')
r.recvuntil(b'Encrypted message :')
ct = bytes.fromhex(r.recvline().strip().decode())
print(time.time() - start)

r.recvuntil(b'[>] ', timeout=2000)

payload = bytes2bin(payload)
state = bytes2bin(state)
assert(len(payload) >= len(state))
print((len(payload) - len(state)))
state = "0" * (len(payload) - len(state)) + state

lcg = []
for a, b in zip(state, payload):
    lcg.append(int(a) ^ int(b))
```

```python
with open(r'D:\Github\ctf\hacktoday\quals\dejavu\test2.txt', 'w+') as f:
    f.write(f'ct = 0x{ct.hex()}\n')
    f.write(f'state = {lcg}')
```

```
PS D:\Github\ctf>  d:; cd 'd:\Github\ctf'; & 'C:\Users\LEGION\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\LEGION\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '62927' '--' 'D:\Github\ctf\hacktoday\quals\dejavu\tebak_panjang.py'
[x] Opening connection to 103.181.183.216 on port 18000
[x] Opening connection to 103.181.183.216 on port 18000: Trying 103.181.183.216
[+] Opening connection to 103.181.183.216 on port 18000: Done
[DEBUG] Received 0x46 bytes:
    b'Welcome!\n'
    b'[1] Encrypt a message\n'
    b'[2] Get an encrypted flag\n'
    b'[3] Exit\n'
    b'[>] '
[DEBUG] Sent 0x2 bytes:
    b'1\n'
[DEBUG] Received 0x15 bytes:
    b'Message to encrypt : '
[DEBUG] Sent 0x1f5 bytes:
    b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\n'
[DEBUG] Received 0x43a bytes:
    b'Encrypted message : 453b97873776424a0c5f8a00e16472ad0b7108708500ebdbcde60d10c8046893967ad1f6f8f4d56d7b6978b057c90958252651f4343e374982894a6094
6294a6edd780f99bd74f74122402a8423a97b51f8639fbfa756f01075882c0620be6e991e300fa8ca742bdb7af7af373c67bc70ebb9e8f4f27ba92b92f5a9832f7702c1c328d5f2aa54c
4d8881b7ee2337c27db21ce2930c8a4d160897bd170ca01faa1219ac32f12fbcfc9426ba9735456842c09b27eb8849917bac81b48844d3729806334e0f90386b8e7993518994ff193433
1d530eeafe4564cddb3c3921181b08bc4b5077f9dd43bd9f30f06a4891940dfe1fde6173400e29222fc7181b48d38a111b91a0c61476e70ed8565c947688a5aa4675d204107bb0a369cb
28efaa6729023b0709b2c359b50e0ece0392cef093223ccd311b2e8b44a2c984009115eb4a8e06395ee9a17bff7ec1ea3bfc09b8b3e9a29d43eafd0e0810b7ee0cad870cef943ba413df
d12c448db68fff0bbd1525cb9532220b953e6eeb7816084f0dde43b998dc33d0800339b5850ce04eb1f7532edd269177fda636a510ddafdeffcfba1a3c57efc2da0917da38a3979eb739
a6811804708dab45e45e94a0443a98ed0c079a0ff8364f02e6c3f474881c317985a9a11ccd9580026b8d91193d2c227db1361a2f95ad2eae65242577ae2410220d9cae5ef6\n'
    b'[1] Encrypt a message\n'
    b'[2] Get an encrypted flag\n'
    b'[3] Exit\n'
    b'[>] '
39.14725613594055
[DEBUG] Sent 0x2 bytes:
    b'2\n'
[DEBUG] Received 0x9a bytes:
    b'Encrypted message : 61e625a69ba3d8122f7d294b8ef506f931fa0ba9401c7d8648fc9818981c92dc9a35abc1\n'
    b'[1] Encrypt a message\n'
    b'[2] Get an encrypted flag\n'
    b'[3] Exit\n'
    b'[>] '
39.157307147979736
0
[*] Closed connection to 103.181.183.216 port 18000
PS D:\Github\ctf>
```

Dari beberapa kali run, kita prediksi besar modulo sekitar 100-105. Kita dapat melakukan brute force pada bit-length state dari LCG yang kita peroleh, dan mencoba mencari parameter LCG:

solve.py:

```python
from azunyan.lcg import recover_mod, recover_vars
from libnum import *
from tqdm import tqdm
def bytes2bin(msg: bytes):
    return bin(int.from_bytes(msg, "big"))[2:]

def bin2bytes(msg: bytes):
```

```
    return int(msg, 2).to_bytes((int(msg, 2).bit_length() + 7) >> 3, "big") or
b"\x00"

def arr2num(arr):
    sum = 0
    for i, c in enumerate(arr):
        sum += (c * 2 ** i)
    return sum

exec(open(r'D:\Github\ctf\hacktoday\quals\dejavu\test2.txt').read())

def inv(num, mult, inc, mod):
    num -= inc
    num = num % mod
    num *= pow(mult, -1, mod)
    num %= mod
    return num

def recover():
    tar = 100
    mini = 20000
    dif = 6
    r = 105
    for start in tqdm(range(r)):
        for a in range(start + r - dif, start + r):
            for b in range(a + r - dif, a + r):
                for c in range(b + r - dif, b + r):
                    for d in range(c + r - dif, c + r):
                        for e in range(d + r - dif, d + r):
                            stops = [a, b, c, d, e]
                            arr = []
                            prev = start
                            for num in stops:
                                arr.append(arr2num(state[prev: num]))
                                prev = num
                            m = recover_mod(arr)
                            for i in range(2, 10000):
                                if m==1:
```

```
                                    break
                                while m%i == 0:
                                    m/=i
                            f = True
                            for num in arr:
                                if m < num:
                                    f = False
                            if not f:
                                continue
                            try:
                                params = recover_vars(arr, m)
                                print(arr, m, params)
                                return arr, m, params
                            except:
                                continue
arr, m, params = recover()
mult, inc = params
start = arr[0]
nums = [inv(start, mult, inc, m)]
for i in range(2):
    nums.append(inv(nums[-1], mult, inc, m))
nums = nums[::-1]
c = ""
for num in nums:
    c += bin(num)[2:][::-1]
ct =
bytes.fromhex("08e6782965c1cf88249dddaad9e712eb2445fa1fee8c4fb5a22c9b0192c27caf
7e5ffe8a")
ct = bytes2bin(ct)
c = c[3: len(ct) + 3]
flag = ""
for a, b in zip(ct, c):
    flag += str(int(a) ^ int(b))
print(bin2bytes(flag))
```

Pemilihan banyak parameter (a-e), dan variable r, dif, dan start sendiri dilakukan dari berbagai eksperimen. Start sendiri juga penting karena ingat bahwa enkripsi flag dilakukan terlebih dahulu, yang

berarti bit pertama yang kita terima bisa jadi merupakan bagian dari state sebelumnya, bukan dari state baru. Perhatikan pula setelah memperoleh parameter lcg dan melakukan invers, perlu dilakukan offset dari bit yang dihasilkan karena panjang ciphertext dan flag memang belum tentu sama (angka 3 dipilih dari eksperimen).

Reversal LCG sendiri dilakukan dengan kode dari library penulis. Berikut potongan azunyan/lcg.py:

```python
from math import gcd as gcd

def recover_mod(arr):
    assert(len(arr) > 3)
    t = []
    for i in range(1, len(arr)):
        t.append(arr[i]-arr[i-1])
    u = []
    for i in range(1, len(t)-1):
        u.append(t[i+1]*t[i-1] - t[i]*t[i])
    ans = u[0]
    for c in u:
        ans = gcd(ans, c)
    return ans

def recover_vars(arr, mod):
    assert(len(arr) > 2)
    try:
        mult = ((arr[2]-arr[1]) % mod)* pow(arr[1]-arr[0], -1, mod)
        mult = mult % mod
        inc = arr[1] - arr[0] * mult
        inc = inc % mod
        return mult, inc
    except:
        return recover_vars(arr[1:], mod)
```

Flag: hacktoday{51MP13_LCG_Cr4CK1N6_1NN17}

Rev

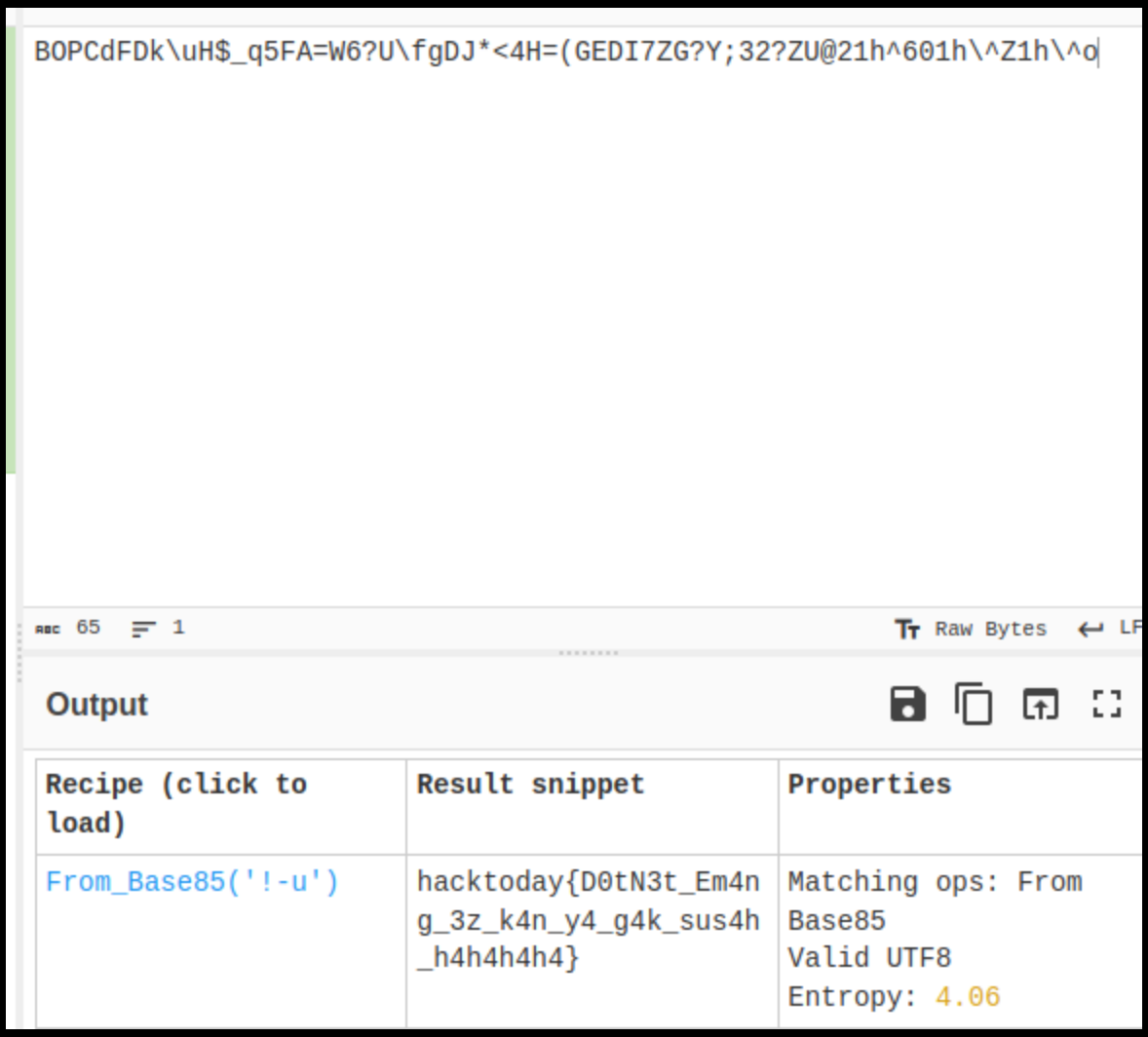
OnlyAdminCanSee

100

my friend sent a file to me and he asked me to hack an application that Mr. John can login. Can you help me reversing it so im able to see what is inside? Help him to login to the app

Diberikan sebuah program exe yang dibuat dengan .NET. Saya melihat dua buah link yang terdapat pada program tersebut.
https://pastebin.com/raw/VWgc4jWn dan
https://pastebin.com/raw/yX2XfwWb kami mencoba untuk melihat link tersebut dan mendapatkan dua buah string berikut
BOPCdFDk\uH$_q5FA=W6?U\fgDJ*<4H=(GEDI7ZG?Y;32?ZU@21h^601h\^Z1h\^o
dan flag{pas5wOrd_nya_7uH_Gampan9_Bro}.

Kami mencoba untuk mensubmit format ini, namun ternyata hasilnya salah dan kami mencoba untuk mendecode string pertama yang didapatkan dan ternytata itu udah flagnya



Flag: hacktoday{D0tN3t_Em4ng_3z_k4n_y4_g4k_sus4h_h4h4h4h4}

kurang-lebih+

452

siapa yang benci soal beranak??, well here we go

-

HINT : If something goes wrong why don't just debug it ^-^

-

NOTE : hacktoday{String yang ditemukan}

Diberikan sebuah file python yang berisi equation equation

```
def check(flag: bytes) → bool:
    return ((flag[22]-flag[28]+flag[4]+flag[35]-flag[29])==-25)and(((flag[26]-flag[0]+flag[23]+flag
    [11]-flag[16])==-19)and(((flag[9]-flag[21]+flag[37]+flag[34]-flag[14])==195)and(((flag[19]-flag
    [18]+flag[6]+flag[13]-flag[12])==112)and(((flag[36]-flag[38]+flag[33]+flag[32]-flag[3])==168)and
    (((flag[2]-flag[20]+flag[7]+flag[10]-flag[30])==49)and(((flag[5]-flag[25]+flag[27]+flag[39]-flag
    [17])==87)and(((flag[24]-flag[8]+flag[15]+flag[31]-flag[1])==102)and(((flag[22]+flag[28]-flag[4]
    -flag[35]+flag[29])==105)and(((flag[26]+flag[0]-flag[23]-flag[11]+flag[16])==83)and(((flag[9]
    +flag[21]-flag[37]-flag[34]+flag[14])==49)and(((flag[19]+flag[18]-flag[6]-flag[13]+flag[12])
    ==122)and(((flag[36]+flag[38]-flag[33]-flag[32]+flag[3])==-96)and(((flag[2]+flag[20]-flag[7]
    -flag[10]+flag[30])==147)and(((flag[5]+flag[25]-flag[27]-flag[39]+flag[17])==123)and(((flag[24]
    +flag[8]-flag[15]-flag[31]+flag[1])==120)and(((flag[22]-flag[28]-flag[4]+flag[35]+flag[29])
    ==-47)and(((flag[26]-flag[0]-flag[23]+flag[11]+flag[16])==-37)and(((flag[9]-flag[21]-flag[37]
    +flag[34]+flag[14])==223)and(((flag[19]-flag[18]-flag[6]+flag[13]+flag[12])==136)and(((flag[36]
    -flag[38]-flag[33]+flag[32]+flag[3])==28)and(((flag[2]-flag[20]-flag[7]+flag[10]+flag[30])==41)
    and(((flag[5]-flag[25]-flag[27]+flag[39]+flag[17])==-27)and(((flag[24]-flag[8]-flag[15]+flag[31]
    +flag[1])==90)and(((flag[22]+flag[28]+flag[4]-flag[35]-flag[29])==127)and(((flag[26]+flag[0]
    +flag[23]-flag[11]-flag[16])==101)and(((flag[9]+flag[21]+flag[37]-flag[34]-flag[14])==21)and
    (((flag[19]+flag[18]+flag[6]-flag[13]-flag[12])==98)and(((flag[36]+flag[38]+flag[33]-flag[32]
    -flag[3])==44)and(((flag[2]+flag[20]+flag[7]-flag[10]-flag[30])==155)and(((flag[5]+flag[25]+flag
    [27]-flag[39]-flag[17])==237)and(((flag[24]+flag[8]+flag[15]-flag[31]-flag[1])==132)and(((flag
    [22]-flag[28]+flag[4]-flag[35]+flag[29])==105)and(((flag[26]-flag[0]+flag[23]-flag[11]+flag[16])
    ==93)and(((flag[9]-flag[21]+flag[37]-flag[34]+flag[14])==173)and(((flag[19]-flag[18]+flag[6]
    -flag[13]+flag[12])==134)and(((flag[36]-flag[38]+flag[33]-flag[32]+flag[3])==64)and(((flag[2]
    -flag[20]+flag[7]-flag[10]+flag[30])==153)and(((flag[5]-flag[25]+flag[27]-flag[39]+flag[17])
    ==95)and(((flag[24]-flag[8]+flag[15]-flag[31]+flag[1])==262)and(((flag[22]+flag[28]-flag[4]+flag
    [35]-flag[29])==-25)and(((flag[26]+flag[0]-flag[23]+flag[11]-flag[16])==-29)and(((flag[9]+flag
    [21]-flag[37]+flag[34]-flag[14])==71)and(((flag[19]+flag[18]-flag[6]+flag[13]-flag[12])==100)and
    (((flag[36]+flag[38]-flag[33]+flag[32]-flag[3])==8)and(((flag[2]+flag[20]-flag[7]+flag[10]-flag
    [30])==43)and(((flag[5]+flag[25]-flag[27]+flag[39]-flag[17])==115)and(((flag[24]+flag[8]-flag
    [15]+flag[31]-flag[1])==-40)and(((flag[22]-flag[28]-flag[4]-flag[35]-flag[29])==-305)and(((flag
    [26]-flag[0]-flag[23]-flag[11]-flag[16])==-329)and(((flag[9]-flag[21]-flag[37]-flag[34]-flag
    [14])==-231)and(((flag[19]-flag[18]-flag[6]-flag[13]-flag[12])==-330)and(((flag[36]-flag[38]
```

Kami mencoba untuk membuat script dengan z3 solver untuk menyelesaikan equation yang diberikan

```
from z3 import *

flag = [Int(f'{i}') for i in range(48)]

s = Solver()

s.add(flag[22]-flag[28]+flag[4]+flag[35]-flag[29] == -25)
s.add(flag[26]-flag[0]+flag[23]+flag[11]-flag[16] ==-19)
s.add(flag[9]-flag[21]+flag[37]+flag[34]-flag[14]==195)
s.add(flag[19]-flag[18]+flag[6]+flag[13]-flag[12]==112)
s.add(flag[36]-flag[38]+flag[33]+flag[32]-flag[3]==168)
s.add(flag[2]-flag[20]+flag[7]+flag[10]-flag[30]==49)
s.add(flag[5]-flag[25]+flag[27]+flag[39]-flag[17]==87)
s.add(flag[24]-flag[8]+flag[15]+flag[31]-flag[1]==102)
s.add(flag[22]+flag[28]-flag[4]-flag[35]+flag[29]==105)
s.add(flag[26]+flag[0]-flag[23]-flag[11]+flag[16]==83)
s.add(flag[9]+flag[21]-flag[37]-flag[34]+flag[14]==49)
s.add(flag[19]+flag[18]-flag[6]-flag[13]+flag[12]==122)
s.add(flag[36]+flag[38]-flag[33]-flag[32]+flag[3]==-96)
s.add(flag[2]+flag[20]-flag[7]-flag[10]+flag[30]==147)
s.add(flag[5]+flag[25]-flag[27]-flag[39]+flag[17]==123)
s.add(flag[24]+flag[8]-flag[15]-flag[31]+flag[1]==120)
s.add(flag[22]-flag[28]-flag[4]+flag[35]+flag[29]==-47)
s.add(flag[26]-flag[0]-flag[23]+flag[11]+flag[16]==-37)
s.add(flag[9]-flag[21]-flag[37]+flag[34]+flag[14]==223)
s.add(flag[19]-flag[18]-flag[6]+flag[13]+flag[12]==136)
s.add(flag[36]-flag[38]-flag[33]+flag[32]+flag[3]==28)
s.add(flag[2]-flag[20]-flag[7]+flag[10]+flag[30]==41)
s.add(flag[5]-flag[25]-flag[27]+flag[39]+flag[17]==-27)
s.add(flag[5]-flag[25]-flag[27]+flag[39]+flag[17]==-27)
s.add(flag[24]-flag[8]-flag[15]+flag[31]+flag[1]==90)
s.add(flag[22]+flag[28]+flag[4]-flag[35]-flag[29]==127)
s.add(flag[26]+flag[0]+flag[23]-flag[11]-flag[16] ==101)
s.add(flag[9]+flag[21]+flag[37]-flag[34]-flag[14]==21)
s.add(flag[19]+flag[18]+flag[6]-flag[13]-flag[12]==98)
s.add(flag[36]+flag[38]+flag[33]-flag[32]-flag[3]==44)
s.add(flag[2]+flag[20]+flag[7]-flag[10]-flag[30]==155)
s.add(flag[5]+flag[25]+flag[27]-flag[39]-flag[17]==237)
s.add(flag[24]+flag[8]+flag[15]-flag[31]-flag[1]==132)
s.add(flag[22]-flag[28]+flag[4]-flag[35]+flag[29]==105)
s.add(flag[26]-flag[0]+flag[23]-flag[11]+flag[16]==93)
s.add(flag[9]-flag[21]+flag[37]-flag[34]+flag[14]==173)
s.add(flag[19]-flag[18]+flag[6]-flag[13]+flag[12]==134)
s.add(flag[36]-flag[38]+flag[33]-flag[32]+flag[3]==64)
s.add(flag[2]-flag[20]+flag[7]-flag[10]+flag[30]==153)
s.add(flag[5]-flag[25]+flag[27]-flag[39]+flag[17]==95)
s.add(flag[24]-flag[8]+flag[15]-flag[31]+flag[1]==262)
```

```
s.add(flag[22]+flag[28]-flag[4]+flag[35]-flag[29]==-25)
s.add(flag[26]+flag[0]-flag[23]+flag[11]-flag[16]==-29)
s.add(flag[9]+flag[21]-flag[37]+flag[34]-flag[14]==71)
s.add(flag[19]+flag[18]-flag[6]+flag[13]-flag[12]==100)
s.add(flag[36]+flag[38]-flag[33]+flag[32]-flag[3]==8)
s.add(flag[2]+flag[20]-flag[7]+flag[10]-flag[30]==43)
s.add(flag[5]+flag[25]-flag[27]+flag[39]-flag[17]==115)
s.add(flag[24]+flag[8]-flag[15]+flag[31]-flag[1]==-40)
s.add(flag[22]-flag[28]-flag[4]-flag[35]-flag[29]==-305)
s.add(flag[26]-flag[0]-flag[23]-flag[11]-flag[16]==-329)
s.add(flag[9]-flag[21]-flag[37]-flag[34]-flag[14]==-231)
s.add(flag[19]-flag[18]-flag[6]-flag[13]-flag[12]==-330)
s.add(flag[36]-flag[38]-flag[33]-flag[32]-flag[3]==-260)
s.add(flag[2]-flag[20]-flag[7]-flag[10]-flag[30]==-267)
s.add(flag[5]-flag[25]-flag[27]-flag[39]-flag[17]==-199)
s.add(flag[24]-flag[8]-flag[15]-flag[31]-flag[1]==-198)
s.add(flag[22]+flag[28]+flag[4]+flag[35]+flag[29]==385)
s.add(flag[26]+flag[0]+flag[23]+flag[11]+flag[16]==393)
s.add(flag[9]+flag[21]+flag[37]+flag[34]+flag[14]==475)
s.add(flag[19]+flag[18]+flag[6]+flag[13]+flag[12]==564)
s.add(flag[36]+flag[38]+flag[33]+flag[32]+flag[3]==332)
s.add(flag[2]+flag[20]+flag[7]+flag[10]+flag[30]==463)
s.add(flag[5]+flag[25]+flag[27]+flag[39]+flag[17]==409)
s.add(flag[24]+flag[8]+flag[15]+flag[31]+flag[1]==420)

print(s.check())
print(s.model())
```

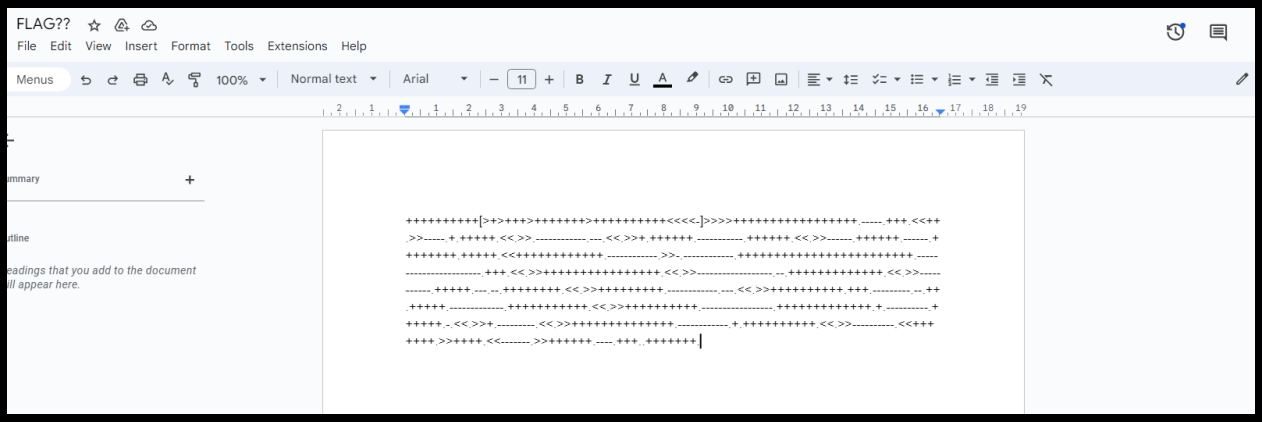
Ternyata equation tersebut satisfied dan setelah dilakukan mapping didapat

```
ans = {39: 41,
 7: 107,
 28: 108,
 38: 36,
 27: 102,
 35: 32,
 37: 94,
 6: 110,
 11: 45,
 25: 116,
 33: 116,
 8: 47,
 0: 105,
 4: 108,
```

```
 18: 104,
 20: 104,
 34: 119,
 31: 32,
 32: 98,
 13: 111,
 10: 51,
 21: 32,
 15: 118,
 23: 110,
 24: 111,
 5: 105,
 2: 98,
 36: 36,
 19: 117,
 9: 122,
 26: 32,
 22: 40,
 1: 112,
 17: 45,
 30: 103,
 3: 46,
 12: 122,
 14: 108,
 16: 101,
 29: 97}
for i in range(40):
  print(chr(ans[i]), end= "")
```

```
 12 = 122,
 14 = 108,
 16 = 101,
 29 = 97]
ipb.link/z3-zolve-huh (not flag btw $^$)
```

Kami pun mendapatkan sebuah link ipb.link/z3-zolve-huh, setelah menuju link tersebut kami diberikan sebuah google docs yang berisi brainfuck



Setelah di decode ternyata menghasilkan sebuah string “ups not the flag again, maybe u can check the original version of this i'm sorry”

Kami pun menuju docs history dan merecover file aslinya

```
,>++++++++++[-<----------->]+<--[[-]>-<]>[-,>++++++++++[-<----------->]+<++[
[-]>-<]>[-,>++++++++++++[-<----------->]+<+++[[-]>-<]>[-,>++++++++++[-<------
------>]+<-----[[-]>-<]>[-,>++++++++++++[-<---------->]+<++++[[-]>-<]>[-,>+++++
+++++[-<---------->]+<-------[[-]>-<]>[-,>++++++++++[-<----------->]+<[[-]>-<
]>[-,>++++++++++[-<---------->]+<[[-]>-<]>[-,>++++++++++[-<---------->]+<+++
++[[-]>-<]>[-,>++++++++++[-<----------->]+<+[[-]>-<]>[-,>+++++[-<---------->
]+<+[[-]>-<]>[-,>++++++++++[-<----------->]+<[[-]>-<]>[-,>++++++++++[-<-----
------>]+<+++[[-]>-<]>[-,>++++++++++[-<---------->]+<+++++[[-]>-<]>[-,>++
++++++++[-<---------->]+<+++++[[-]>-<]>[-,>++++++++++[-<---------->]+<++++
++[[-]>-<]>[-,>+++++++++[-<----------->]+<--[[-]>-<]>[-,>++++++++++[-<------
---->]+<--[[-]>-<]>[-,>++++++++++[-<---------->]+<-[[-]>-<]>[-,>++++++++++[-
<----------->]+<----[[-]>-<]>[-,>++++++++++[-<---------->]+<-----[[-]>-<]>[
-,>+++++++++[-<---------->]+<-----[[-]>-<]>[-,>++++++++++[-<---------->]+<
++++[[-]>-<]>[-,>++++++++++[-<----------->]+<-[[-]>-<]>[-,>++++++++++[-<---
------>]+<++++[[-]>-<]>[-,>++++++++++[-<---------->]+<++[[-]>-<]>[-,>+++++++
+++[-<---------->]+<++++++[[-]>-<]>[-,>++++++++++[-<----------->]+<[[-]>-<
]>[-,>+++++++++[-<---------->]+<---[[-]>-<]>[-,>+++++++++[-<---------->]+<
[[-]>-<]>[-,>++++++++++[-<---------->]+<+++[[-]>-<]>[-,>++++++++[-<--------
--->]+<-------[[-]>-<]>[-,>++++++++++[-<---------->]+<----[[-]>-<]>[-,>+++++
+++++[-<----->]+<-[[-]>-<]>[-,>++++++++++[-<---------->]+<++++++[[-]>-<]>[-
,>+++++[-<---------->]+<-[[-]>-<]>[-++++++++++[>+>+++>+++++++>++++++++++<<<<
-]>>>>-.++++++++++++.+++..-------------.--.+++++++++++++++++.<<++.>>-----.-.
----------.<<.>>+++++++++.-----.------.++.<<.>>++++++++++++++++++++.---------
-.++++++.<<+++++++.>>---.-------------.<<-------.>>+++++++++.+.+++++.[-]][-]
][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]
][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]
][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]][-]]++++++++++[>+>+
+++>+++++++>++++++++++<<<<-]>>++.>>+++++++++++++++++++.-----.---.-.-------.<<
.>>----------.+.-.
```

Setelah kami decode string yang dihasilkan ialah “Wrong ^_^”. Kami curiga bahwa brainfuck ini merupakan sebuah checker flag lagi. Kami memilih untuk mentranspile kode brainfuck ini ke bahasa c dengan library https://github.com/paulkaefer/bftoc didapat hasil decode nya seperti ini.

```
/* This is a translation of input.bf, generated by bftoc.py (by Paul Kaefer)
 * It was generated on Saturday, August 26, 2023 at 12:58AM
 */

#include <stdio.h>

void main(void)
{
    int size = 1000;
    int tape[size];
    int i = 0;

    /* Clearing the tape (array) */
    for (i=0; i<size; i++)
        tape[i] = 0;

    int ptr = 0;

    tape[ptr] = getchar();
    ptr += 1;
    tape[ptr] += 10;
    while (tape[ptr] != 0)
    {
        tape[ptr] -= 1;
        ptr -= 1;
        tape[ptr] -= 11;
        ptr += 1;
    }
    tape[ptr] += 1;
    ptr -= 1;
    tape[ptr] -= 2;
    while (tape[ptr] != 0)
    {
        while (tape[ptr] != 0)
        {
            tape[ptr] -= 1;
        }
        ptr += 1;
        tape[ptr] -= 1;
        ptr -= 1;
```

```
    }
    ptr += 1;
    while (tape[ptr] != 0)
    {
        tape[ptr] -= 1;
        tape[ptr] = getchar();
        ptr += 1;
        tape[ptr] += 10;
        while (tape[ptr] != 0)
        {
            tape[ptr] -= 1;
            ptr -= 1;
            tape[ptr] -= 11;
            ptr += 1;
        }
        tape[ptr] += 1;
        ptr -= 1;
        tape[ptr] += 2;
        while (tape[ptr] != 0)
        {
            while (tape[ptr] != 0)
            {
                tape[ptr] -= 1;
            }
            ptr += 1;
            tape[ptr] -= 1;
            ptr -= 1;
        }
        ptr += 1;
        while (tape[ptr] != 0)
        {
            tape[ptr] -= 1;
            tape[ptr] = getchar();
            ptr += 1;
            tape[ptr] += 12;
            while (tape[ptr] != 0)
            {
                tape[ptr] -= 1;
                ptr -= 1;
                tape[ptr] -= 10;
                ptr += 1;
            }
            tape[ptr] += 1;
            ptr -= 1;
            tape[ptr] += 3;
            while (tape[ptr] != 0)
            {
                while (tape[ptr] != 0)
```

```
                {
                    tape[ptr] -= 1;
                }
                ptr += 1;
                tape[ptr] -= 1;
                ptr -= 1;
            }
__SNIP__
```

Jika dilihat kode akan mengambil input satu persatu dan mengoperasikan input yang diberikan dengan pengurangan atau penjumlahan pada while loop.

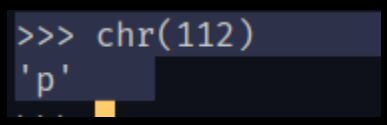
```
    tape[ptr] = getchar();
    ptr += 1;
    tape[ptr] += 10;
    while (tape[ptr] != 0)
    {
        tape[ptr] -= 1;
        ptr -= 1;
        tape[ptr] -= 11;
        ptr += 1;
    }
    tape[ptr] += 1;
    ptr -= 1;
    tape[ptr] -= 2;
    while (tape[ptr] != 0)
    {
        while (tape[ptr] != 0)
        {
            tape[ptr] -= 1;
        }
        ptr += 1;
        tape[ptr] -= 1;
        ptr -= 1;
    }
    ptr += 1;
```

Kita ingin nilai tape[ptr] bernilai nol pada while loop yang kedua, sehingga kita perlu memasukan input yang tepat agar nilai tape[ptr] tidak negatif. Dengan bantuan breakpoint vscode kami mencoba untuk mendebug nilai yang diminta pada setiap while loop dan menggantinya dengan 0 untuk bypass hasil yang diminta.

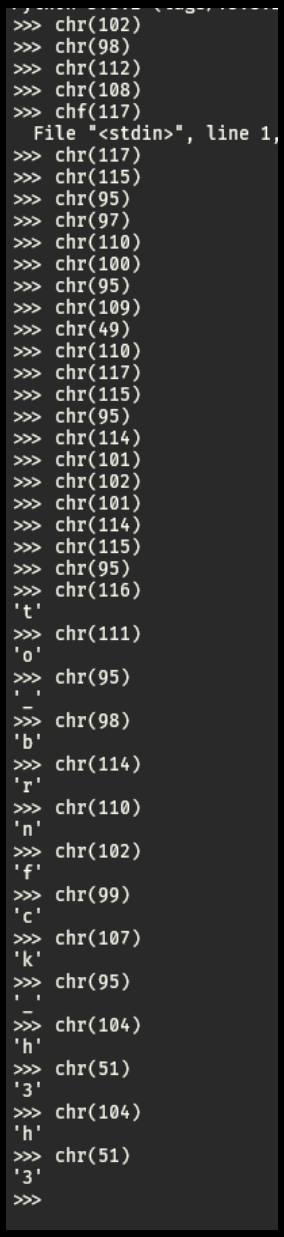
Kami menginputkan newline (0xa) sehingga nilai yang ada pada tape[0] adalah 10

Kita tidak ingin nilai tape[ptr] < 0 karena akan menyebabkan infinite loop.

Karena tadi nilai awal pada tape[0] adalah 10 dan nilai saat ini adalah -102, berarti nilai yang harus kita input ialah 10 - (-102) yaitu 112.



Langkah ini dialnjutkan hingga mencapai loop paling dalam sehingga didapat nilai seperti ini.



Susun huruf huruf yang didapat dan flag pun didapat

Flag: hacktoday{plus_and_m1nus_refers_to_brnfck_h3h3}

Web

LogInspek

100

siapa yang benci soal beranak??, well here we go

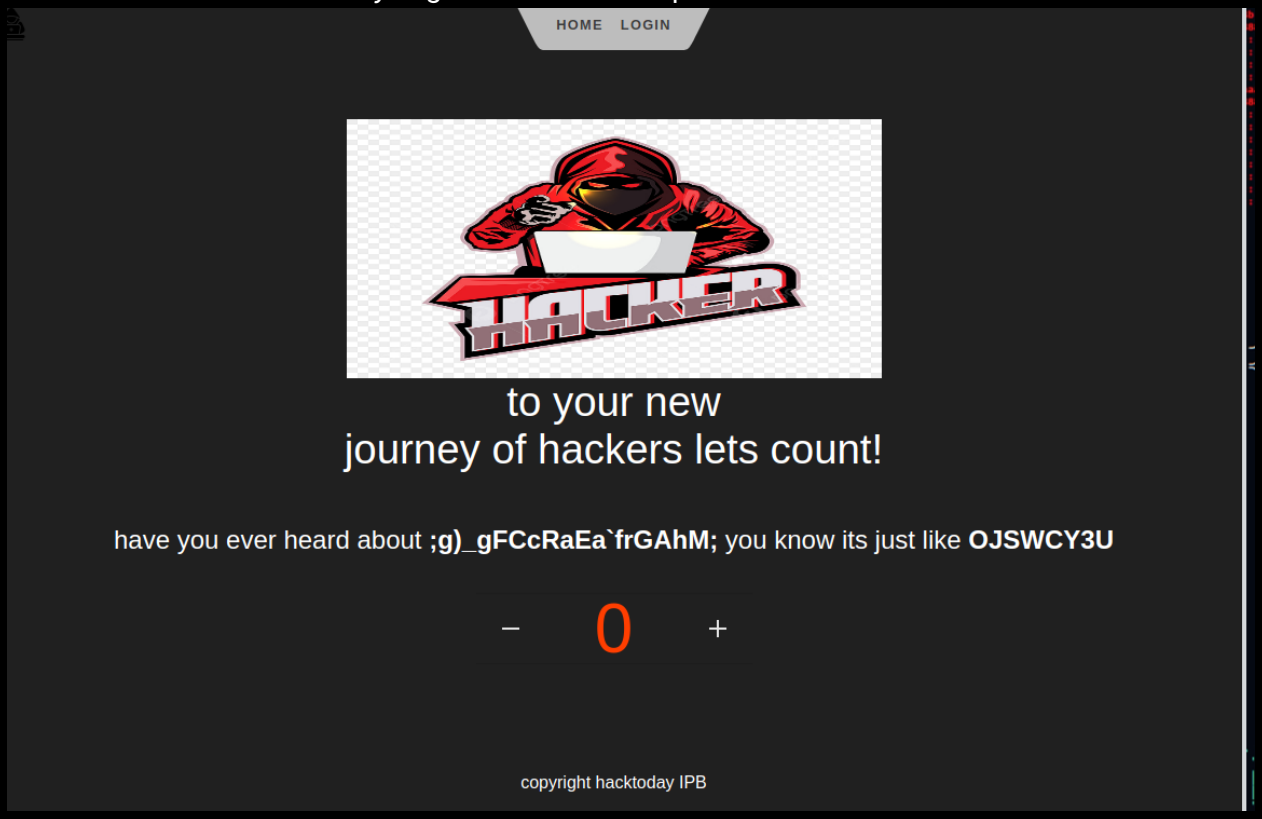
-

HINT : If something goes wrong why don't just debug it ^-^

-

NOTE : hacktoday{String yang ditemukan}

Diberikan sebuah web yang memiliki tampilan aneh



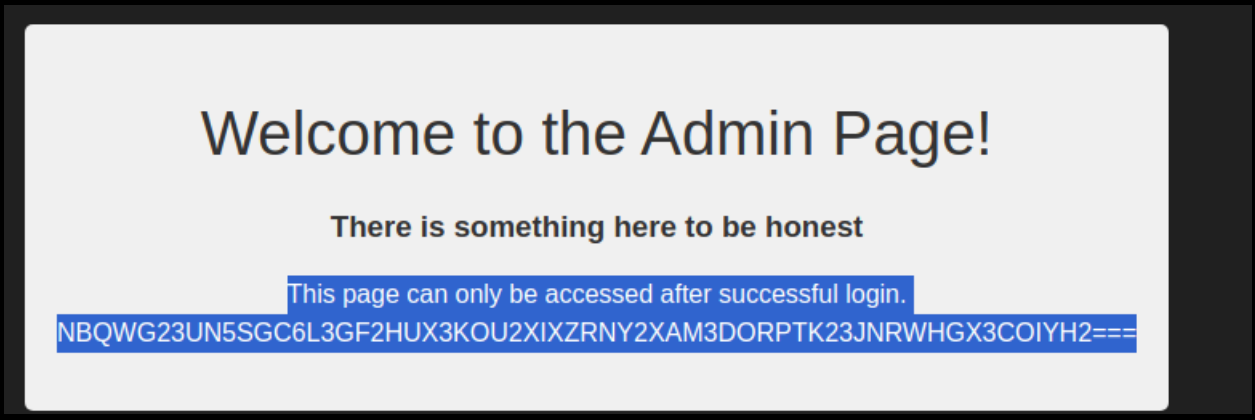
Langkah pertama kami melakukan inspect element namun hasilnya nihil. Setelah membaca deskripsi lebih lanjut, kami menemukan file robots.txt yang berisi

```
# https://www.robotstxt.org/robotstxt.html
User-agent: *
Disallow:

# i see something in nigol.js
```

Kami pun mencoba unuk mengakses nigol.js

```
import { writable } from "svelte/store";

export const loggedIn = writable(true);
export function login(email, password) {
  if (email.trim() === "" || password.trim() === "") {
    alert("Please enter a valid user and password.");
    return;
  }
  const validEmail = "admin";
  const validPassword = "4dm1nP4ss1s33sy";
  if (email === validEmail && password === validPassword) {
    loggedIn.set(true);
    window.location.href = "/UwU";
  } else {
    alert("Invalid user or password. Please try again.");
  }
}
```

Kami langsung mencoba login dengan email dan password yang diberikan



Kami mencoba untuk mendecode dengan base32 dan didapat flag

Flag: hacktoday{1tz_ju5t_1n5p3ct_5kills_br0}

Misc

Simulasi UTBK

405

Mari gan yang kangen atau mau UTBK bisa kerjain soal saya
nc 103.181.183.216 19003

Pembahasan

Setelah mencoba mengerjakan soal UTBK, kami menyimpulkan bahwa kami perlu menyimpan soal dan jawaban yang diberikan dari server. Berikut script yang digunakan:

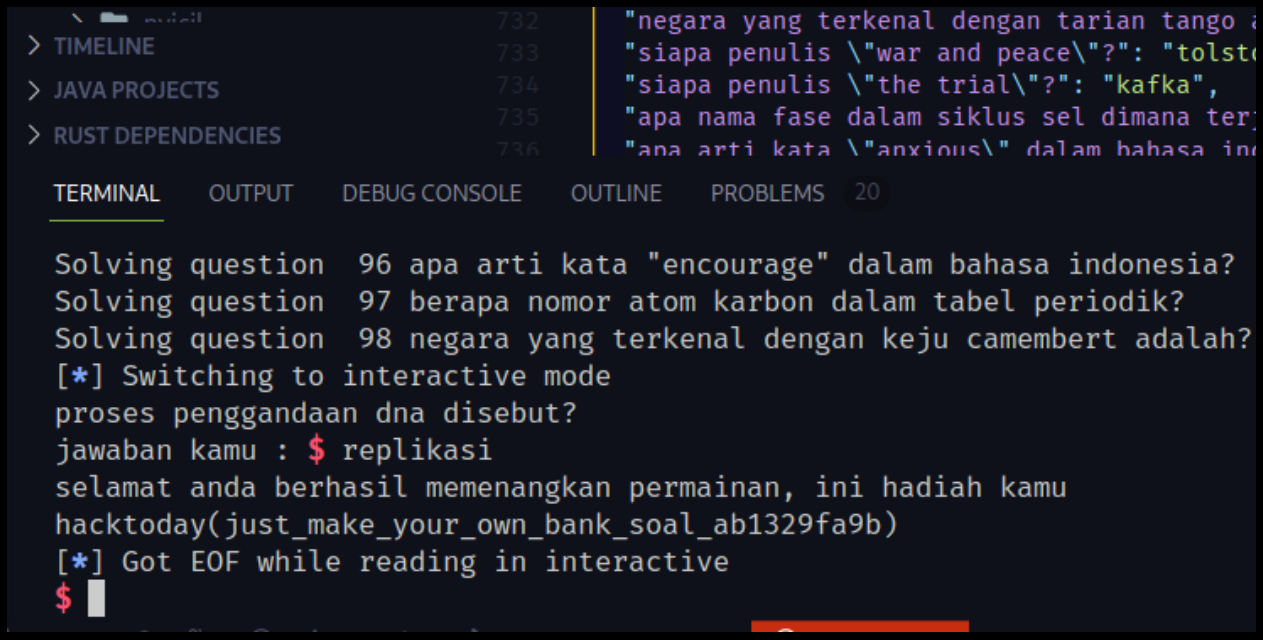
```
#!/usr/bin/python
import json

from pwn import *

CONN = "nc 103.181.183.216 19003".split(" ")
HOST = CONN[1]
PORT = CONN[2]

# ============================================================
#                   WRAPPER FUNCTION
# ============================================================

def sl(x): io.sendline(x)
def sla(x, y): io.sendlineafter(x, y)
def se(x): io.send(x)
def sa(x, y): io.sendafter(x, y)
def ru(x, drop=False): return io.recvuntil(x, drop=drop)
def rl(): return io.recvline()
```

```python
def cl(): io.clean()
def un64(x): return u64(x.ljust(8, b'\x00'))
def leak(name, addr): info(f"{name} @ {hex(addr)}")

# ===========================================================
#                          SETUP
# ===========================================================

# Allows you to switch between local/GDB/remote from terminal
def start(argv=[], *a, **kw):
  return remote(HOST, PORT, *a, **kw)

f = open("./list.json", "r")
wordlist = json.load(f)
f.close()


io = start()


# ============ Get available Wordlist
def getwordlist():
  io = start()

  ru(b"\n\n")
  rl()
  rl()

  def get_problem_and_ans():
    problem = rl().decode().strip()
    sl(b"a")
    ru(b"adalah ")
    ans = rl().decode().strip()
```

```
    rl()
    rl()

    return problem, ans

  for i in range(3):
    problem, ans = get_problem_and_ans()
    wordlist[problem] = ans.strip()

  f = open("list.json", "w")
  f.write(json.dumps(wordlist))
  f.close()

  io.clean()
  io.close()
  sleep(2)

# for i in range(1000):
#   try:
#     getwordlist()
#   except:
#     pass

# ==== Receive input
ru(b"\n\n")
rl()
rl()

def answer(i):
  problem = rl().decode().strip()
  print("Solving question ", i, problem)

  if problem in wordlist:
```

```
    ans = wordlist[problem]
    sl(ans.encode())
    rl()
    rl()
    rl()
  else:
    sl(b"a")
    ru(b"adalah ")
    ans = rl().decode().strip()
    rl()
    rl()


for i in range(99):
  answer(i)
  sleep(0.1)


io.interactive()
```

Kita melakukan getWordList berulang-ulang hingga didapatkan kurang lebih 1000 pertanyaan dan jawabannya.



Setelah didapatkan, kami menjalankan script di atas dan mendapatkan flag

Flag: hacktoday(just_make_your_own_bank_soal_ab1329fa9b)

DCHEZKIBOXS

100

Kali ini Pak Masse menemukan sebuah port aneh yang berisi kalimat rahasia. Untuk melihat kalimat tersebut ia diminta memasukan password berupa Substring terpajang pertama dari string yang diberikan port tersebut yang merupakan "kata DCHEZKIBOXS". Sebuah string dikatakan "kata DCHEZKIBOXS" jika dan hanya jika string tersebut dibelah dua secara horizontal kedua bagian string (atas dan bawah) akan membentuk bagian yang seimbang dan dapat dibentuk menjadi sama persis jika beberapa bagiannya dirotasi. Contoh: "CHECK" merupakan salah satu "kata DCHEZKIBOXS" karena ketika belah dua kedua bagian akan membentuk bagian yang seimbang. visualisasi contoh dapat dilihat di bagian "attachement". Karena string yang diberikan sangat panjang, untuk menemukan passwordnya Pak Masse menggeser-geser jendela ruangannya atau memainkan dua buah pointer miliknya untuk mendapatkan berbagai inspirasi.

nc 103.181.183.216 19001

Berdasarkan hint yang diberikan kita cukup membuat script untuk mendapatkan substring yang memenuhi aturan yang disebutkan untuk mendapatkan password

```
wordlist = "DCHEZKIBOXS"
with open("words.txt", 'r') as file:
    words = file.read()

currPw = ""
maxPw = ""
for word in words:
    if (word in wordlist):
        currPw+=word
    else:
        if (len(currPw) > len(maxPw)):
            maxPw = currPw
        currPw = ""

if(len(currPw) > len(maxPw)):
    print(currPw)
else:
    print(maxPw)
```

```
CVRHZNAXFPXWTIFJBLJOFLWRZORHNYXXAGJDQUZHFVKZYCXRCYEJAVTPDNOVGNGUMNXJIVKKUFKLXMJKXWDFWQVMSNNBTHDSZAHWELKWWHBFEWFCUNQQCETQ
CGYQSWXWMEGGZKSDUHIDGRUHQLVCESZDVZPBJENEYBZZVQDHBMWGBGBJNLGGKAMDCLUQNJIGXHCPMWSPMHOMQAGVNKMXOQYKHIUOTIXTRPSGYGKYJJLXVWSM
SQKUPEJAJSWCODDXFKRWWNFOWNOFYOCDNNAOXWUCRVFLNZANDWFYYJEMHCTCSWKWFRQLTQFGQIYCZVZPROORIGKKKRKGYDOUZMXXBWXVCTHZKNWAEUWOPWLZ
GNZBGEIDQMKUNTRTJYMXLGGINQMWNYBAZUJDFTUHYKBUUMKLBKGPLONBRQNMASPCBLPFFQVUSBBPNLZDVKRLVEPDFXQEILWJKGEOTPVINPWJGSFUMRPEMVEL
MZUEHKAUOUERXJGVTKWDUYJWVMNLUUDESJIEVYQP

Itu stringku, mana passwordmu?

passwordnya adalah substring terpanjang pertama yang merupakan 'kata DCHEZKIBOXS'.
Sebuah string dikatakan 'kata DCHEZKIBOXS' jika dan hanya jika string tersebut dibelah dua secara horizontal kedua bagia
n string (atas dan bawah) akan membentuk bagian yang berimbang dan dapat dibentuk menjadi sama persis jika beberapa bagi
annya dirotasi.
password: KIKDBIDBZIHKH
Congratsss, ini flag buatmu! : hacktoday{Yeyyy_n0w_y0U_kNOw_5Lid1ng_Wind0w5_4l6oR1thMs!}
```

Flag: hacktoday{Yeyyy_n0w_y0U_kNOw_5Lid1ng_Wind0w5_4l6oR1thMs!}

Where is my git?

100

I just playing around with git command, and suddenly, my flag (i mean, my git) is disappear. Can you find it for me?

Diberikan sebuah file zip yang berisi README.md

```
# where-is-my-git
Can you find my flag in this repo?
```

Kami mencoba untuk melakukan git logs namun hanya terdapat satu commit. Setelah itu kami mencoba untuk melakukkan git reflog dan ternyata mendapatkan beberapa commit yang telah di hapus

```
5274a89 HEAD@{44}: checkout: moving from a6a3cd388a922b4d28dd9053e0b04fd957d0ca
1 to 5274a89868ca16ff2a2130b2d803383d8aa04941
a6a3cd3 HEAD@{45}: checkout: moving from 0afcd10d6b36020ec60d0f7a55d402a5a42516
4 to a6a3cd388a922b4d28dd9053e0b04fd957d0ca11
0afcd10 HEAD@{46}: checkout: moving from 1258008d36e6549d0053d14688b118fac70573
7 to 0afcd10d6b36020ec60d0f7a55d402a5a42516b4
1258008 HEAD@{47}: checkout: moving from 23d8d069ba72a9f579742eedb00181de914635
f to 1258008d36e6549d0053d14688b118fac7057317
23d8d06 HEAD@{48}: checkout: moving from 2c9f988c56f84d4ff6bdce8c6c2a062da739dc
f to 23d8d069ba72a9f579742eedb00181de914635bf
2c9f988 HEAD@{49}: checkout: moving from 2fa45ff77d2e445b4bd8b5925645258530e70a
a to 2c9f988c56f84d4ff6bdce8c6c2a062da739dc6f
2fa45ff HEAD@{50}: checkout: moving from 428994df2e27f5a019fbd739164758eb2443a7
3 to 2fa45ff77d2e445b4bd8b5925645258530e70a5a
428994d HEAD@{51}: checkout: moving from f3765432238eb527b5476cf32c1dabc6c8eadc
```

Pada setiap commit yang dihapus terdapat sebuah file-{i}.txt yang didalamnya merupakan sebuah susunan character flag.

Kami langsung menyimpan setiap hash dari commit tersebut ke sebuah file .txt lalu kami membuat script seperti ini untuk solvernya

```python
import os
import subprocess

f = open("gitlog.txt")
flag = b""
os.chdir("./where-is-my-git")
for idx, line in enumerate(f.readlines()):
  if idx % 2 == 1 or idx == 126:
    continue

  info = line.split(" ")
  filename = info[-1].strip()
  commitId = info[1]

  print(commitId, filename)

  subprocess.run(["git", "checkout", commitId])
  res = subprocess.check_output(["cat", filename])
  flag += res

print(flag)
```

```
HEAD is now at 8167fbf Added part-62.txt
f7b2c5151969ab48103314e81bf71d368cf79f5b part-63.txt
Previous HEAD position was 8167fbf Added part-62.txt
HEAD is now at f7b2c51 Added part-63.txt
b'hacktoday{thank_you_for_finding_my_flag_from_this_git_1an23nfa}'
```

Flag: hacktoday{thank_you_for_finding_my_flag_from_this_git_1an23nfa}

Foren

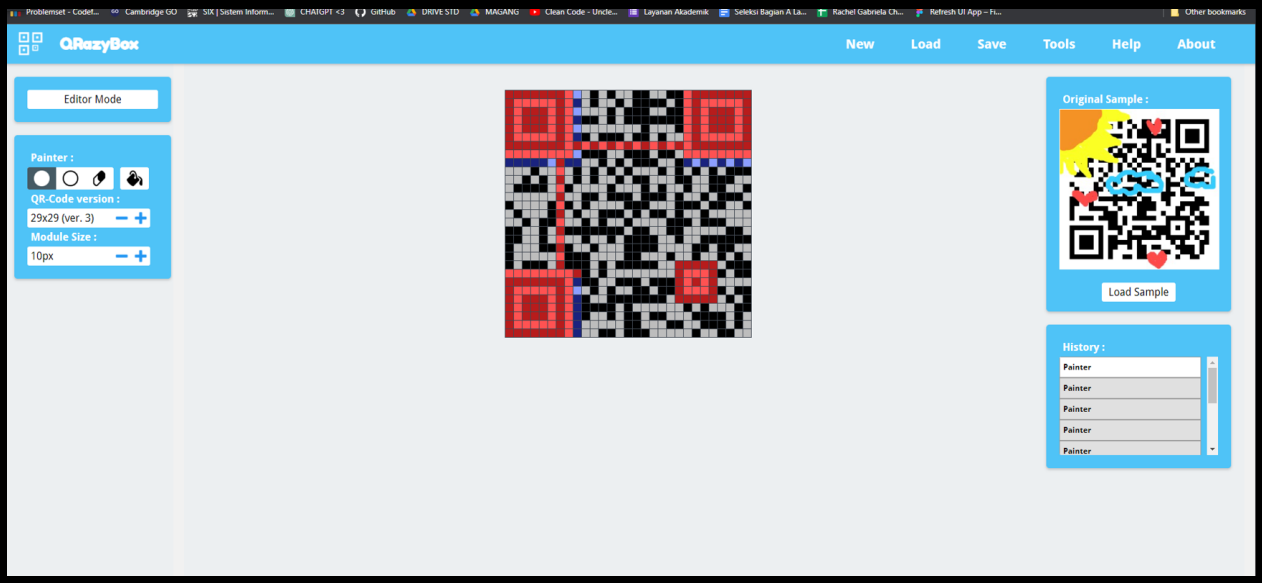
Doodled

100

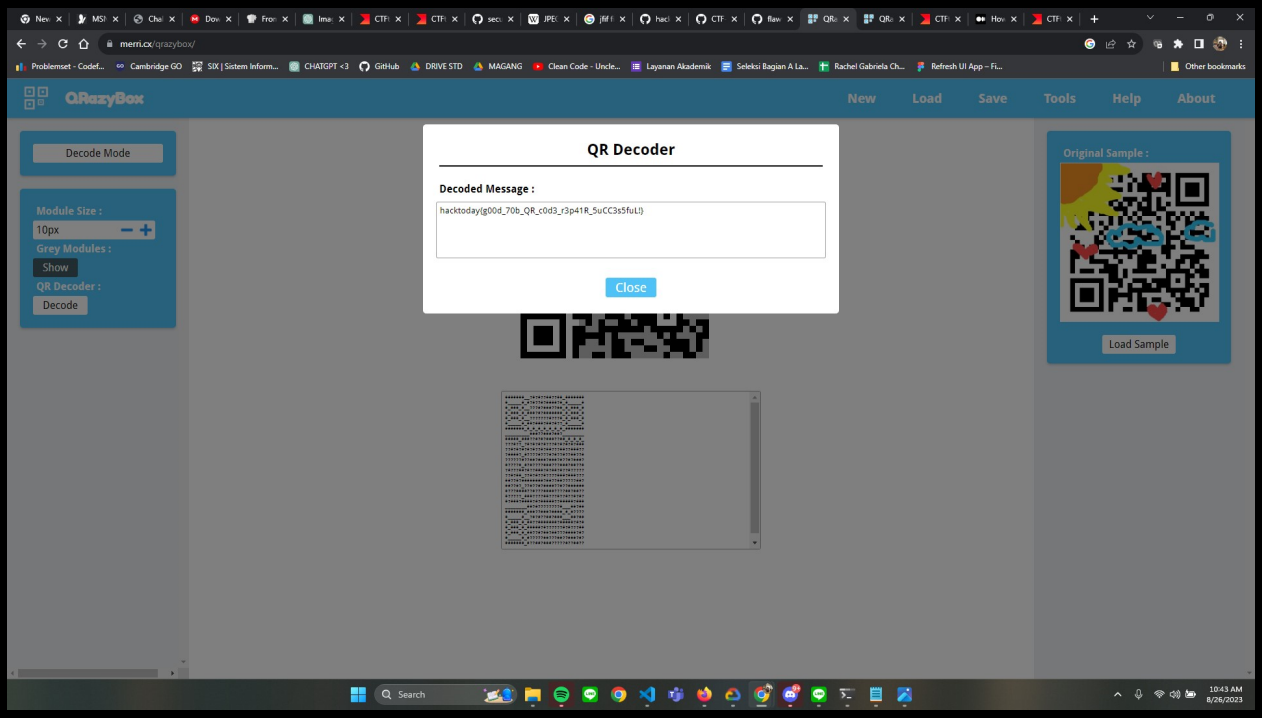
Oh no! My cousin Gio Ferdiansyah has scribbled all over my stuff! Almost everything is covered with his doodles, including an important QR code, making it unreadable. Can you help me retrieve the lost information?

Diberikan sebuah QR yang telah dicoret-coret:

Kami menggunakan QcrazyBox https://merri.cx/qrazybox/ untuk memperbaiki QRCode dengan membuat ulang QRCode di web tersebut dan memilih format info pattern yang benar:



Setelah berhasil memperbaiki QRCode, kami men-decode QR code yang didapatkan:



Flag: hacktoday{g00d_70b_QR_c0d3_r3p41R_5uCC3s5fuL!}

yesterday-afternoon-kidz

380

Our live proxy has detected hacking activity in our logs; analyze the log file to find out what data the hacker retrieved

Diberikan sebuah file yang berisi log. Dalam log tersebut ditemukan payload SQL injection. Dengan pattern

```
%28select+sleep%281%29+from+user+where+BINARY+substring%28secretdata%2C1%2C1%29+%3D+CHAR%2832%29%29+%23+&password=afternoon-yesterday-kidz
```

Dari paylod tersebut, kita mengetahui bahwa attacker berusaha melakukan time-based SQL injection dimana jika karakter yang dikirimkan sesuai, maka akan terjadi delay response.

Namun, mengingat behavior SQL injection pada umumnya, kita menyadari biasanya SQL injection dilakukan dengan men-traverse karakter dari awal hingga akhir. Jika karakter pada indeks ke-i telah tepat ditemukan, maka dilakukan brute-force untuk karakter selanjutnya.

Sehingga, kita hanya perlu mem-parse file logs dan mencari payload terakhir yang dikirim untuk masing-masing indeks.

Script yang digunakan:

```
import re

with open('logs', 'rb') as file:
    content = file.read()

flag = ''
lastchar = ''
i = 1
while lastchar!='}':
    pattern = rb'%28secretdata%2C' + bytes(str(i), 'utf-8') +
rb'%2C1%29\+%3D\+CHAR%28(.{3})'
    matches = re.findall(pattern, content)
    lastchar = chr(int(matches[-1].decode('utf-8').rstrip('%')))
    print(lastchar,end='')
    i+=1
```

```
PS E:\CTF COMPS\HackToday\Penyisihan\foren> python -u "e:\CTF COMPS\HackToday\Penyisihan\foren\solve.py"
hacktoday{it-yesterday_database_secret_sorry_i_need_to_make_this_long_enough_for_manual_player_like_yesterday_afternoon_kidz_or_it_will_be_too_damn_sleepy(1)_right?}
PS E:\CTF COMPS\HackToday\Penyisihan\foren>
```

Flag:
hacktoday{it-yesterday_database_secret_sorry_i_need_to_make_this_long_enough_for_manual_player_like_yesterday_afternoon_kidz_or_it_will_be_too_damn_sleepy(1)_right?}

OSINT

Kuala Lumpur

401

Beberapa tahun yang lalu, aku pergi ke Kuala Lumpur. Suatu sore di sana, aku naik LRT satu kali dari KL Sentral untuk menuju ke sebuah kawasan di KL. Sekeluarnya aku dari stasiun tujuanku, aku dapat melihat menara kembar Petronas dengan jelas, karena memang letak kawasan ini dekat dengan KLCC. Namun setelah itu, aku berbalik arah menuju utara dan menemukan semacam pasar malam yang menjual berbagai street food khas Malaysia. Aku melanjutkan perjalananku ke arah barat dan utara dan akhirnya sampai di suatu Masjid Jamek.

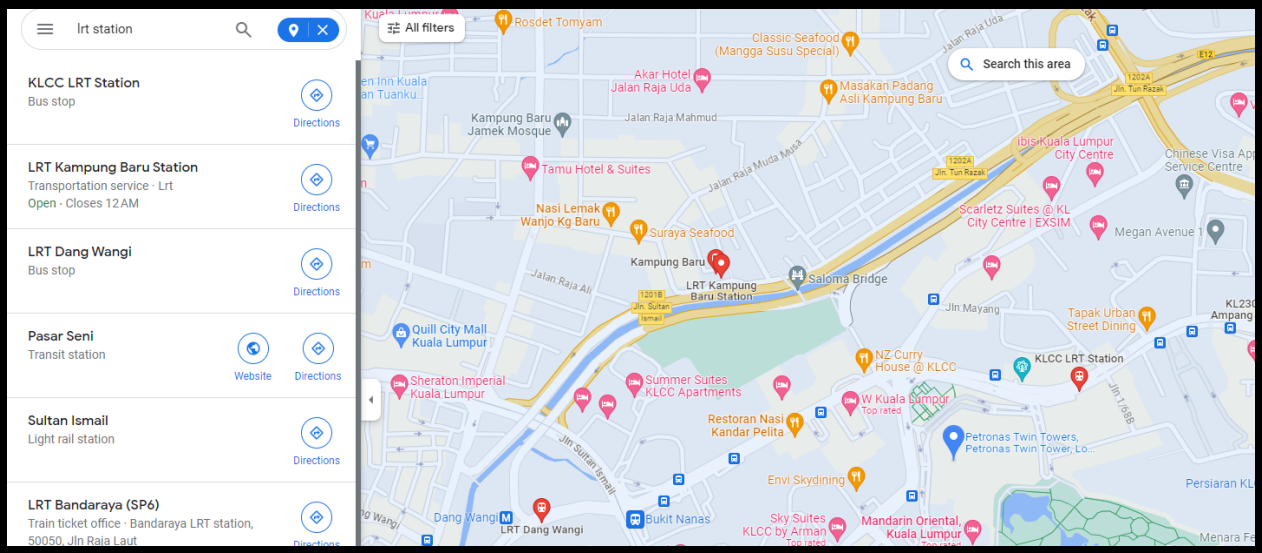
-

Ketika berada di masjid, aku memotret sebuah gambar yang merupakan menara utama masjid. Saat itu merupakan waktu menjelang matahari terbenam dan rupanya, aku juga memotret sebuah benda langit yang berupa sebuah titik putih dan letaknya ada di samping menara. Oh iya, satu hal yang kuingat, ketiga tempat yang kusebutkan di atas (stasiun, pasar malam, dan masjid jamek), semuanya memiliki nama yang sama dengan nama kawasan tersebut.

-

HINT: google "star position tool"

Kami melakukan googling map di sekitar Petronas dan mencari stasiun LRT yang berada di sekitarnya.



Kita mendapatkan stasiun Kampung Baru. Kemudian kami memastikan terdapat Pasar Malam Kampung Baru dan masjid Jamek Kampung Baru. Karena benar, maka kami yakin kawasan yang dimaksud adalah Kampung Baru.

Hal ini kami pastikan saat mendapatkan foto yang sesuai dengan menara yang dimaksud:

Kemudian, kami mencari tanggal foto itu tersebut dengan mem-parse nama file foto.

```
import datetime

timestamp = 1485514151327 / 1000
date = datetime.datetime.utcfromtimestamp(timestamp)
print(date)
```

Didapatkan tanggal 2017-01-27

Kami kemudian menggunakan star position tool untuk mencari bintang pada tanggal tersebut di Kampung Baru:

Flag: hacktoday{KampungBaru_20170127_Venus_Pisces}

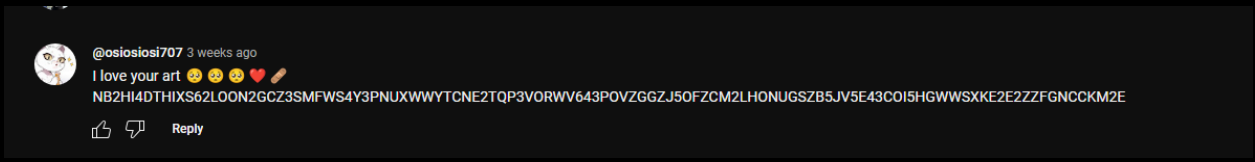
MUA

100

Aku akhir-akhir ini sangat suka sekali make up >< aku juga menyukai salah satu make up artist asal korea. AH!! aku lupa namanya, tapi dia adalah orang yang ada di gambar ini. Aku sangat menyukai make up hasilnya. Selain jago make up dia juga jago menggambar. Aku meninggalkan komentar beberapa hari yang lalu untuk menyemangatinya di salah satu video di channel youtubenya. Video itu berisi vlog dia sedang menggambar.

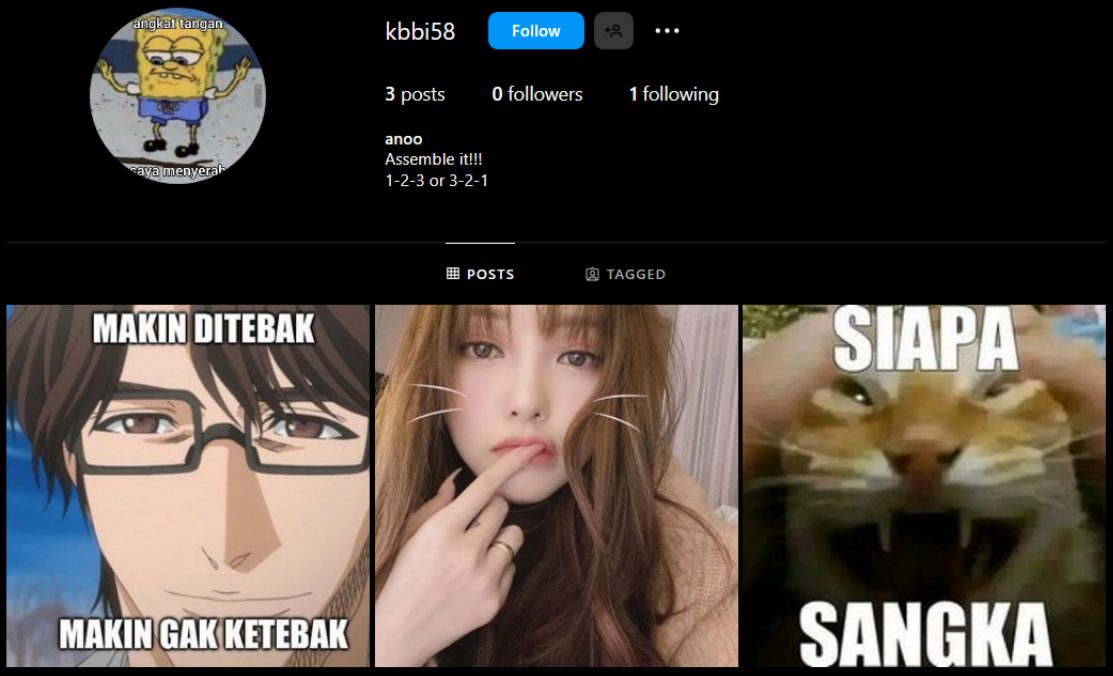
Diberikan sebuah image berupa foto wanita

Setelah dicari di google dan yandex menggunakan reverse image search, didapat bahwa orang ini bernama Pony dan didapat youtube channelnya yaitu https://www.youtube.com/@PONYMakeup/videos. Setelah menonton beberapa video menggambarnya ditemukan sebuah komentar pada video https://www.youtube.com/watch?v=Hcst57-v9XU&t=5s

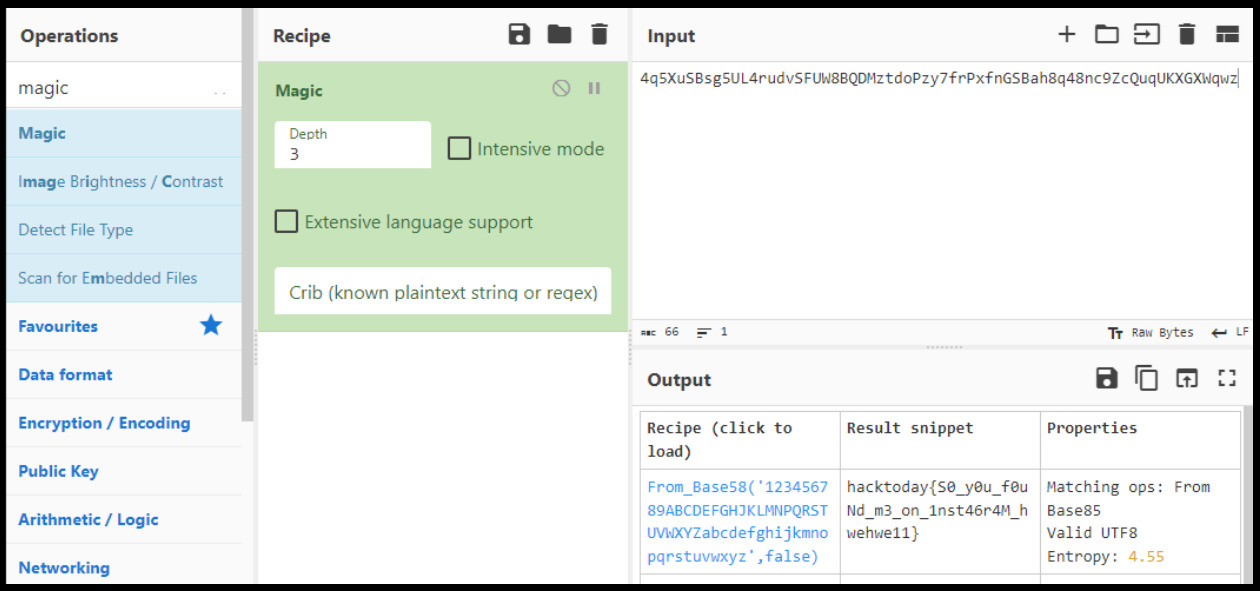


Setelah didecode menggunakan base32 didapat link instagram https://instagram.com/kbbi58?utm_source=qr&igshid=MzNlNGNkZWQ4Mg%3D%3D

Kami pun membuka instagram tersebut dan menemukan 3 buah postingan yang masing2 berisi sebuah string

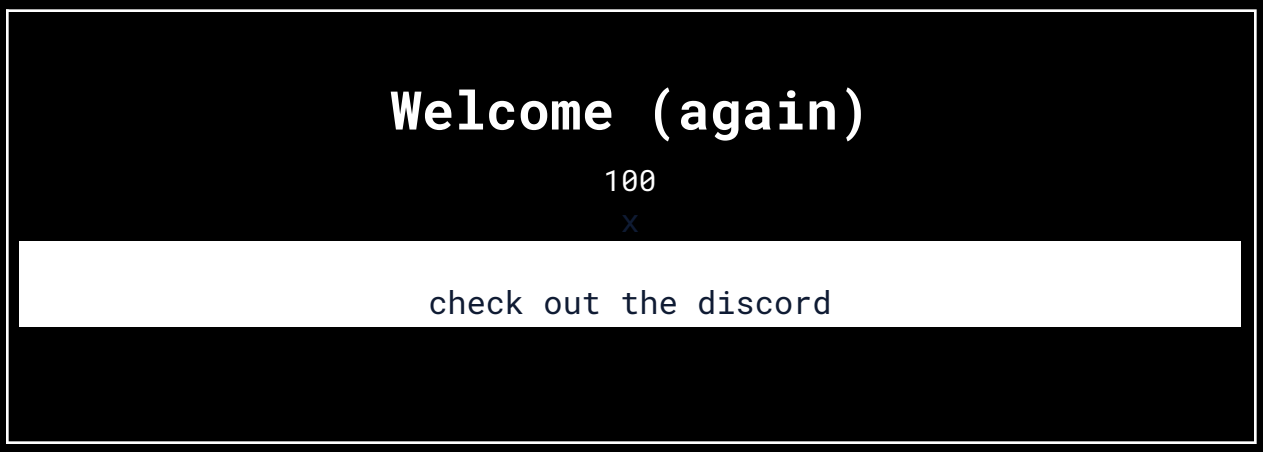


Kami mencoba mengambil masing2 string pada setiap postingan dengan urutan 3-2-1 menjadi 4q5XuSBsg5UL4rudvSFUW8BQDMztdoPzy7frPxfnGSBah8q48nc9ZcQuqUKXGXWqwz. Kami langsung decode di cyberchef dan dapat flag



Flag: hacktoday{S0_y0u_f0uNd_m3_on_1nst46r4M_hwehwe11}

All



Flag: hacktoday{maru_stands_for_molecular_atomic_reconstructed_unit}