WRITE-UP GEMASTIK 2023
DIVISI II KEAMANAN SIBER
KUALIFIKASI

30 Juli 2023

anak kemaren sore

(IPB University)

» Patar Isac Pardomuan «
» Aulia Rochman «
» Muhammad Jundi Fathan «

Daftar Isi

Forensic

GreyHat (500 pts)

Description

A DFIR enjoyer who love analyze dump memory ask your help for analyze sus file. He says "Aku menemukan ini dari hasil dump memory client ku, client-ku berkata bahwa file dia pada direktori tertentu tiba-tiba menghilang, dan ia terakhir kali menjalankan sebuah executables, dan mungkin ini penyebabnya. Aku mengambil beberapa file yang sus dan sepertinya ini kunci untuk menemukan penyebab hilangnya file-file tersebut. Bisa kah kau menolong ku?"

Notes : Jangan di run exe-nya ya, kecuali di sandbox yang emang buat coba-coba

Author: blacowhait

Attachment

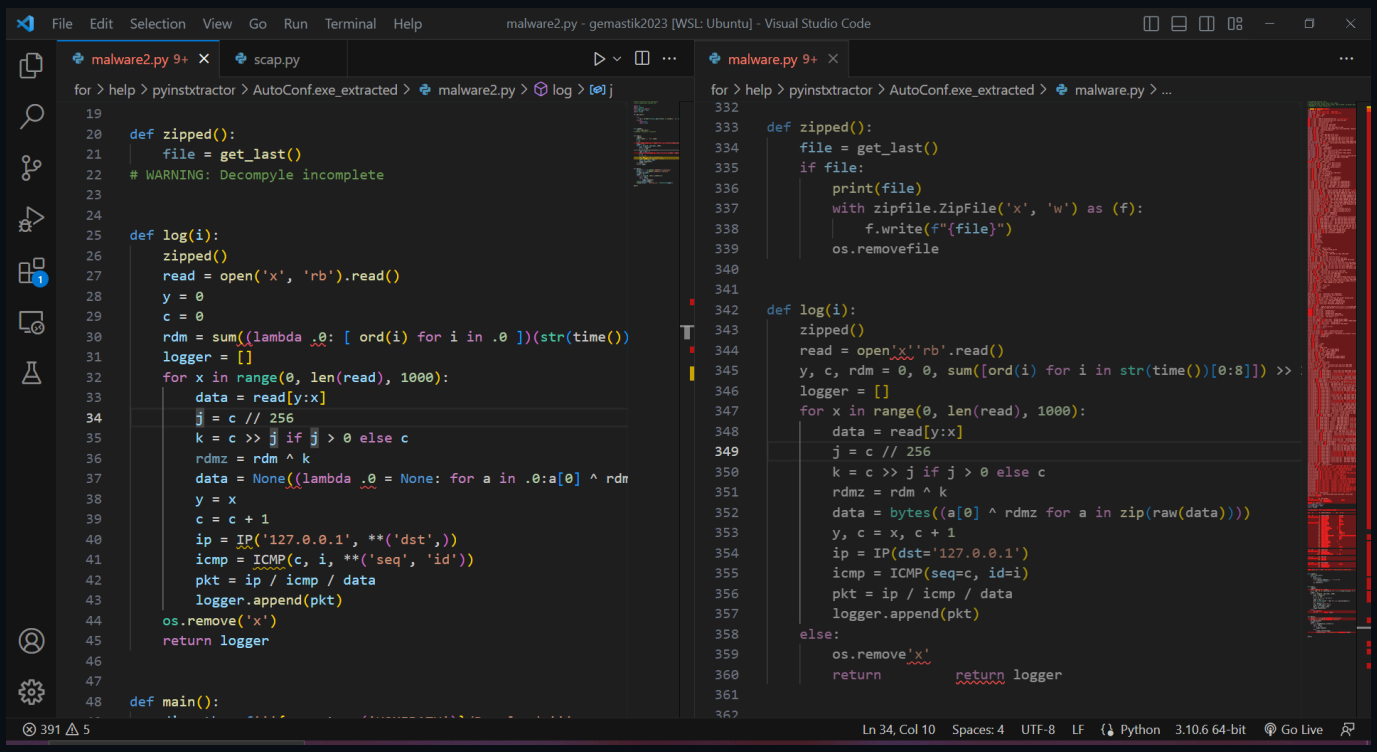
File : https://mega.nz/file/a2oixaib#Y7hu00KLlvhqVV43BzYYKcqxy8hEkUbZ1iL2rDg2wCM

Solution

Diberikan suatu file zip bernama help.zip yang berisikan 2 file yaitu AutoConf.exe dan desktop,ini. Awalnya kami kira kami harus menganalisis file AutoConf ini dengan IDA, tapi ketika dicek dengan strings, ternyata ada string menarik seperti python38.dll. Sehingga, kami mencoba untuk mengekstrak file .exe ini dengan pyinstxtractor. Untuk file desktop,ini kami menemukan bahwa ini merupakan file pcap capture file, sehingga bisa kita buka dengan menggunakan wireshark

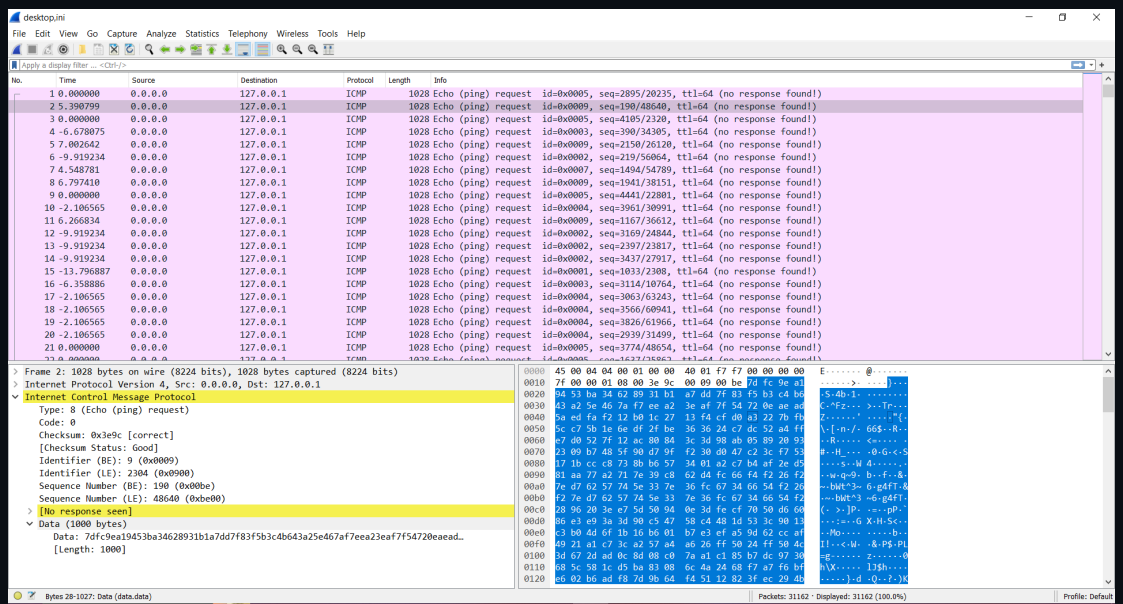
Setelah dengan pyinstxtractor, kami menemukan terdapat file malware.pyc. Kami kemudian coba decompile dengan uncompyle6, tetapi ada bagian yang broken, sehingga kami decompile juga dengan pycdc, lalu kami save di malware.py dan malware.pyc

```
/mnt/d/CTF/gemastik2023/for/help/pyinstxtractor/AutoConf.exe_extracted
 🕐 4:22:16
$ ls
PYZ-00.pyz              base_library              pyimod01_archive.pyc
PYZ-00.pyz_extracted    base_library.zip          pyimod02_importers.pyc
VCRUNTIME140.dll        libcrypto-1_1.dll         pyimod03_ctypes.pyc
_bz2.pyd                libssl-1_1.dll            pyimod04_pywin32.pyc
_hashlib.pyd            malware.py                python38.dll
_lzma.pyd               malware.pyc               select.pyd
_socket.pyd             malware2.py               struct.pyc
_ssl.pyd                pyiboot01_bootstrap.pyc   unicodedata.pyd
```



Terdapat beberapa fungsi pada file malware.py ini, tapi kita fokuskan pada fungsi log. Perhatikan bagaimana cara dia mengambil data per 1000 byte, menginisialisasi nilai y, c, j, k, dan rdm. Kemudian, nilai rdmz didapat dengan melakukan xor antara rdm dan k, dan nilai data akan di-xor dengan nilai rdmz. Setelah itu, nilai c akan diiterasi dan nilai y akan diganti dengan nilai x agar data yang dibaca adalah per 1000 byte. Setelah itu, akan disimpan nilai IP pada ip, nilai ICMP dengan nilai c dan i berupa seq dan id pada icmp, dan di-wrap menjadi satu pada pkt, berupa ip, icmp, dan data.

Ketika file desktop, ini dibuka dengan wireshark, kita bisa amati bahwa setiap packet memiliki panjang 1028 bytes, dengan 28 bytes header dan 1000 bytes data. Sehingga, isi dari desktop,ini sesuai dengan kode pada malware.pyc



Kita cukup reverse proses dari fungsi log ke dalam solver, dan juga untuk membaca file desktop,ini, diperlukan library scapy. Karena dia diacak urutannya, tapi tetap menyimpan nilai c alias sequencenya, kita tetap bisa mengurutkannya dengan melakukan sorting pada seq. Full solver yang kami gunakan adalah sebagai berikut :

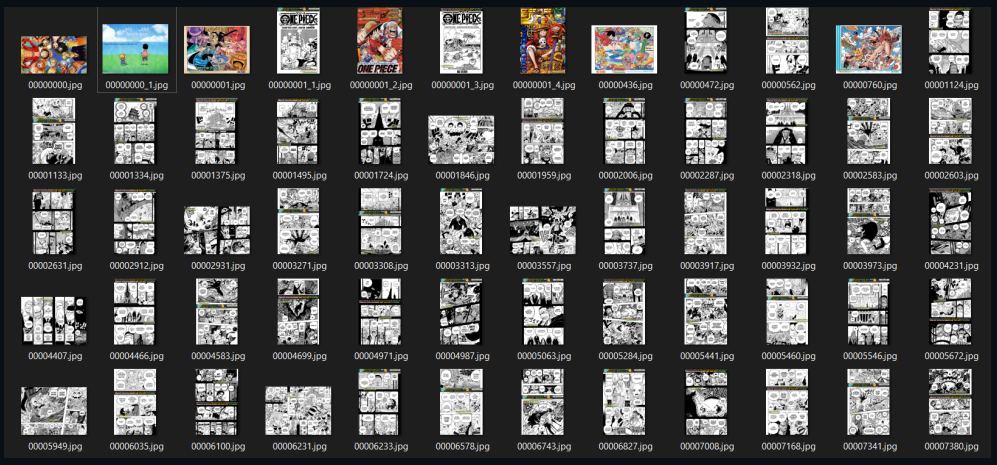
scap.py

```
from scapy.all import *
f = rdpcap('desktop,ini')
isidata = {}
for pck in f:
    c = pck[ICMP].seq - 1
    i = pck[ICMP].id
    j = c // 256
    k = c >> j if j > 0 else c
    rdmz = (sum([ord(urut) for urut in str(pck.time)[0:8]]) >> 3) ^ k
    try:
        isi = pck[Raw].load
    except:
        continue
    plain = bytes([a[0] ^ rdmz for a in zip(raw(isi))])
    if not isidata.get(i):
        isidata[i] = {}
    isidata[i][c] = plain

semuafile = []
for i in range(1,11):
    mentahan = b""
    seqs = list(isidata[i].keys())
    seqs.sort()
    for s in seqs:
        mentahan += isidata[i][s]
    semuafile.append(mentahan)

for i in range(1,11):
    namafile = f"file{i}"
    with open(namafile, "wb") as out:
        out.write(semuafile[i-1])
        out.close()
```

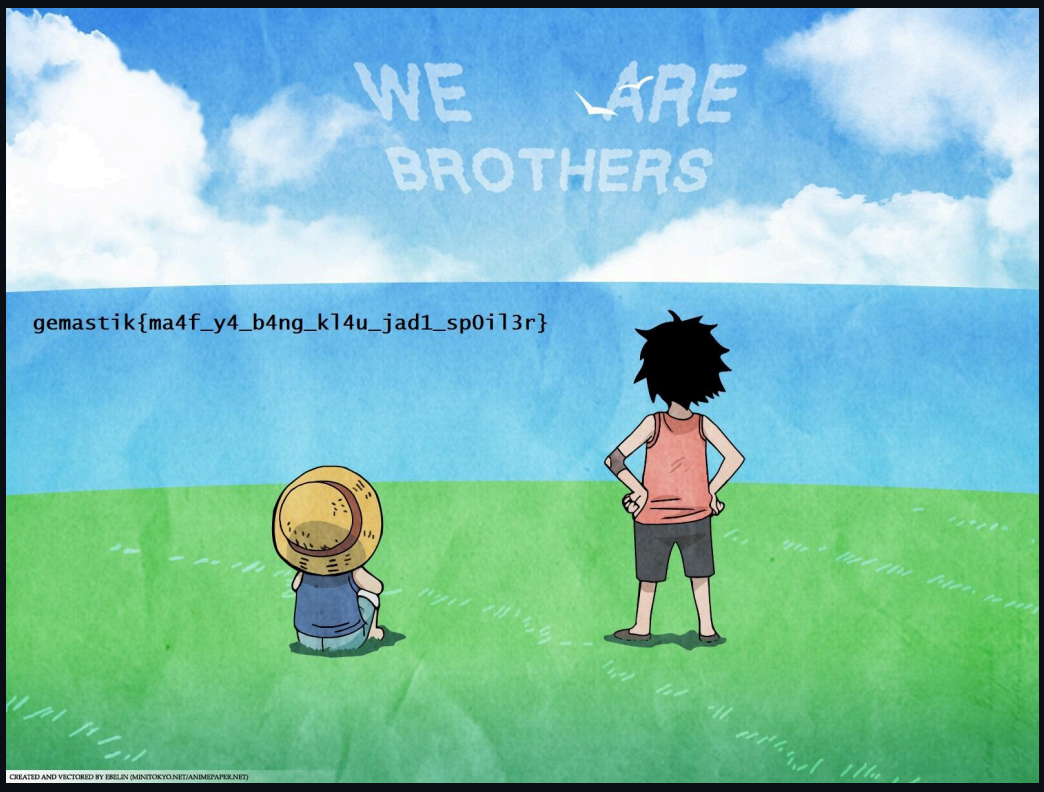
Kita akan mendapat 10 file berbeda. Kemudian, kita gunakan binwalk untuk menganalisis isi dari file-file tersebut

```
/mnt/d/CTF/gemastik2023/for/help 🕐 7:22:12
$ binwalk file*

Scan Time:     2023-07-31 07:23:01
Target File:   /mnt/d/CTF/gemastik2023/for/help/file1
MD5 Checksum:  340c81daef2bee670153da4c29d323f9
Signatures:    411

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------

49            0x31            PDF document, version: "1.7"
670           0x29E           JPEG image data, JFIF standard 1.01
682943        0xA6BBF         Zlib compressed data, default compression
683017        0xA6C09         Zlib compressed data, default compression
683130        0xA6C7A         Zlib compressed data, default compression
683460        0xA6DC4         JPEG image data, JFIF standard 1.01
```

Ternyata terdapat file pdf dan jpeg. Kita coba extract dengan menggunakan foremost



Hasil dari ekstraksi menggunakan foremost menunjukkan banyak gambar komik, dan ada satu gambar yang berbeda sendiri yaitu flag yang kita cari



Flag : gemastik{ma4f_y4_b4ng_kl4u_jad1_sp0il3r}

Cryptography

easy AES (50 pts)

Description

Attack on AES OFB

Author: prajnapras19
nc ctf-gemastik.ub.ac.id 10002

Attachments

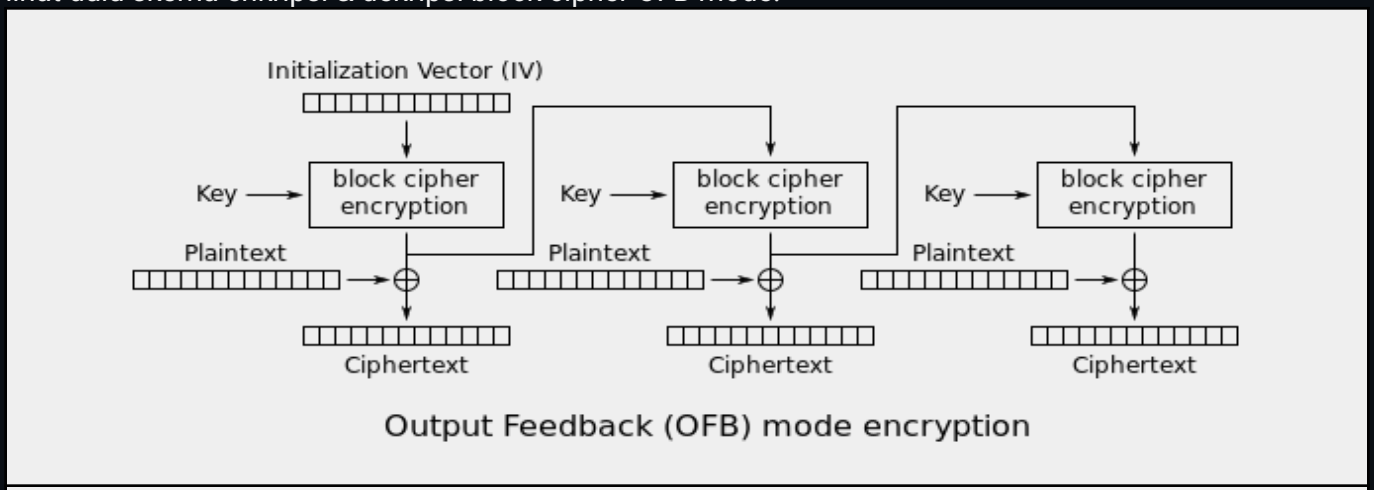
chall.py

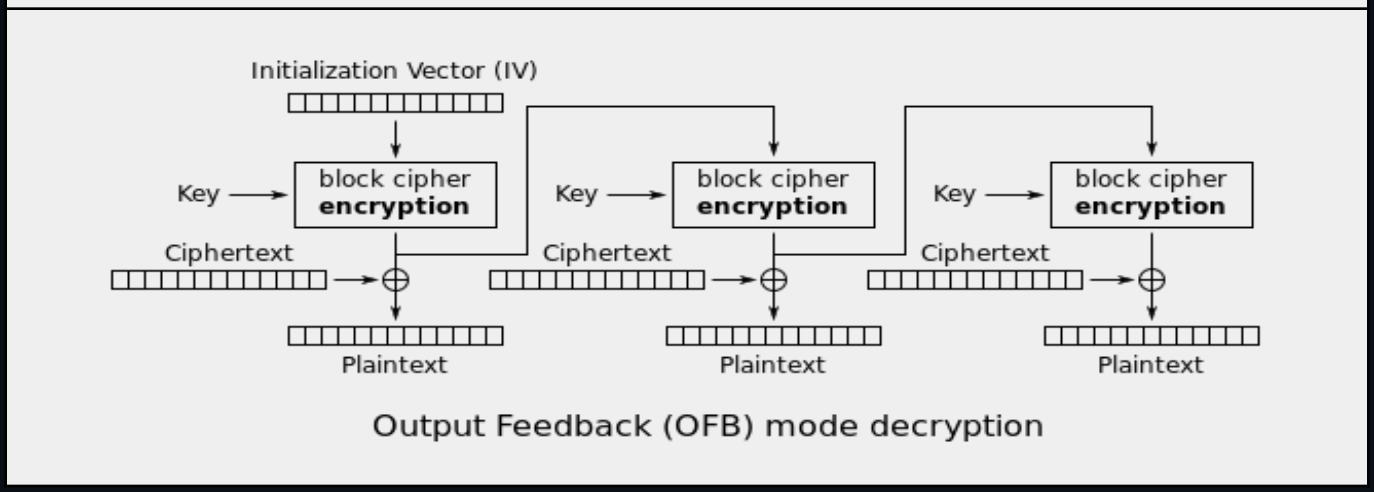
```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from Crypto.Util.number import bytes_to_long, long_to_bytes
import os

key = os.urandom(AES.key_size[0])
iv = os.urandom(AES.block_size)
secret = bytes_to_long(os.urandom(128))

def encrypt(pt):
    bytes_pt = long_to_bytes(pt)
    cipher = AES.new(key, AES.MODE_OFB, iv)
    padded_pt = pad(bytes_pt, AES.block_size)
    return bytes_to_long(cipher.encrypt(padded_pt))

def menu():
    print('===== Menu =====')
    print('1. Encrypt')
    print('2. Get encrypted secret')
    print('3. Get flag')
    print('4. Exit')
    choice = int(input('> '))
    return choice

def get_flag():
    res = int(input('secret: '))
    if secret == res:
        os.system('cat flag.txt')
        print()

while True:
    try:
        choice = menu()
        if choice == 1:
            pt = int(input('plaintext = '))
            ciphertext = encrypt(pt)
            print(f'{ciphertext = }')
        if choice == 2:
            ciphertext = encrypt(secret)
```

```
            print(f'{ciphertext = }')
        if choice == 3:
            get_flag()
            break
        if choice == 4:
            break
    except:
        print('something error happened.')
        break

print('bye.')
```

Solution

Pada challenge ini kita harus memberikan secret yang random untuk mendapatkan flag. Pertama, kita lihat dulu skema enkripsi & dekripsi block cipher OFB mode.





Dari skema tersebut, kita ketahui bahwa enkripsi dan dekripsi OFB mode sama persis, hanya ditukar saja plaintext dan ciphertext-nya. Oleh karena itu, pada challenge ini kita bisa tinggal mengambil encrypted secret, kemudian encrypt kembali encrypted secret tersebut untuk mendapatkan secret yang asli. Tapi jangan lupa kalau secret hanya terdiri dari 128 byte sedangkan enkripsi pada service ditambahkan padding, sehingga kita harus mengambil 128 byte awal saja untuk mengambil secret.

Berikut adalah full solvernya.

solver.py

```python
#!/usr/bin/env python3
from patsac import *

if not args.LOCAL:
    with open("nc.sh") as f:
        NC = f.read().strip().split()
        f.close()
    SRVR = NC[1]
    PORT = NC[2]
    r = remote(SRVR, PORT, level="warning")
else:
    r = process("./chall.py", level="debug")


def main():
    r.sendlineafter(b"> ", b"2")
    r.recvuntil(b"ciphertext = ")
    ct0 = int(r.recvline(0).decode())
    r.sendlineafter(b"> ", b"1")
    r.sendlineafter(b"plaintext = ", str(ct0).encode())
    r.recvuntil(b"ciphertext = ")
    ct1 = int(r.recvline(0).decode())
    secret = str(bytes_to_long(long_to_bytes(ct1)[:128])).encode()
    r.sendlineafter(b"> ", b"3")
    r.sendlineafter(b"secret: ", secret)
    print(r.recv().decode())
    return 0


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

Screenshot

```
patsac ~/ctf/2023/gemastik_qual/cry/easyaes
➜ ./solver.py
gemastik{33d4cfb92569d8f21851f9c3dc96c244c3e344064ffb0b35603a550802b7531e}
```

Flag : gemastik{33d4cfb92569d8f21851f9c3dc96c244c3e344064ffb0b35603a550802b7531e}

k-1 (416 pts)

Description

Shamir said we need k shares to recover the secret.

Author: prajnapras19

nc ctf-gemastik.ub.ac.id 10000

Attachments

chall.py

```
import random
import os

bits = 1024
k = random.randint(20, 35)
password = random.getrandbits(bits) % 1000000

def get_shares():
    coeffs = [password] + [random.getrandbits(bits) for _ in range(k - 1)]
    x_list = set()
    while len(x_list) < k - 1:
        x_list.add(random.getrandbits(bits))

    shares = []
    for x in x_list:
        y = sum(map(lambda i : coeffs[i] * pow(x, i), range(len(coeffs))))
        shares.append((x, y))

    print(f'{k = }')
    for share in shares:
        print(share)

def get_flag():
    res = int(input('password: '))
    if password == res:
        os.system('cat flag.txt')
        print()

try:
    get_shares()
    get_flag()
except:
    print('something error happened.')
```

Solution

Untuk mendapatkan flag, kita perlu menghitung nilai *password*. Pada challenge ini kita diberikan nilai share yang isinya adalah x dan y dengan deskripsi sebagai berikut.

$$x = random\ 1024\ bits\ integer$$
$$y = \sum_{i=0}^{k-1} coeffs[i] \cdot x^i$$
$$coeffs[0] = password$$

Kita sudah tahu pasti coeffs[0] adalah password yang artinya password tidak dikalikan dengan x karena x pangkat 0 adalah 1. Itu artinya jika kita menghitung y modulo x, kita akan mendapatkan nilai password.

Berikut adalah full solvernya.

solver.py

```python
#!/usr/bin/env python3
from patsac import *

if not args.LOCAL:
    with open("nc.sh") as f:
        NC = f.read().strip().split()
        f.close()
    SRVR = NC[1]
    PORT = NC[2]
    r = remote(SRVR, PORT, level="warning")
else:
    r = process("./chall.py", level="debug")


def main():
    k = r.recvline(0)
    xy = r.recvline(0)[1:-1].split(b", ")
    x, y = int(xy[0]), int(xy[1])
    ans = str(y % x).encode()
    r.sendlineafter(b"password: ", ans)
    print(r.recv().decode())

    return 0


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

Screenshot

```
patsac ~/ctf/2023/gemastik_qual/cry/k1
➜ ./solver.py
gemastik{a5061d673d03123fe19bbbcc853da16c9f6fee9e55c26f2c9b7b937b60c0ec83}
```

Flag : gemastik{a5061d673d03123fe19bbbcc853da16c9f6fee9e55c26f2c9b7b937b60c0ec83}

naughty-boy (482 pts)

Description

La La La - Naughty Boy

Lyrics La La, La La La La La na na na na na La La na na, La La La La La na na na na na La La, La La La La La na na na na na La La na na, La La La La La na na na na na

Author: Chovid99
nc ctf-gemastik.ub.ac.id 10001

Attachments

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

print(f'Generating secret and hints... Be patient and sing this song :)')
print(f'''
-----------------------------------------
La La La - Naughty Boy

Lyrics
La la, la la la la la na na na na na
La la na na, la la la la la la na na na na na
La la, la la la la la na na na na na
La la na na, la la la la la la na na na na na
...
-----------------------------------------
''')
secret_val = bytes_to_long(os.urandom(100))
z1 = getStrongPrime(512)
z2 = getStrongPrime(512)
z3 = getPrime(256)
modd = getPrime(2048)

n = z1*z2
e = 65537
c = pow(secret_val, e, n)

rand_1 = getRandomNBitInteger(modd.bit_length() - 1013)
rand_2 = bytes_to_long(os.urandom(128))

hidden_val = z1*z2*z3 + rand_1
hint_1 = (z3**8)*z2 + 0x1337*z2*(z1**2) + rand_2
hint_2 = pow(hidden_val, 4*modd, modd)
print(f'Finished generating secret and hints! Below is the known values:')
print(f'{e = }')
print(f'{c = }')
print(f'{n = }')
print(f'{modd = }')
print(f'{hint_1 = }')
print(f'{hint_2 = }')

res = int(input('What is the secret: '))
if secret_val == res:
    print('GG! Here is your prize:')
    os.system('cat flag.txt')
    print()
else:
    print('Try harder naughty boy!')
```

Solution

Untuk mendapatkan flag, kita harus mendapatkan *secret_val*. Pertama, kita bisa membedah terlebih dahulu nilai *hint_2*.

$$hint2 = hiddenval^{4 \cdot modd} \bmod modd$$

$$Fermat\ little\ theorem$$
$$hint2 = hiddenval^{4} \bmod modd$$

Karena *hint2* adalah *hiddenval* pangkat 4 mod *modd*, maka kita bisa menghitung *hiddenval* dengan menghitung quadratic residue dari *hint2* sebanyak dua kali. Hal ini bisa dilakukan dengan syarat $modd \equiv 3 \pmod 4$.

Jika sudah mendapatkan *hiddenval*, selanjutnya mencari z3. Prediksi z3 bisa dihitung dengan mencari dahulu nilai *rand1*.

$$hiddenval = z1 \cdot z2 \cdot z3 + rand1$$
$$hiddenval = n \cdot z3 + rand1$$
$$rand1 \approx hiddenval \bmod n$$
$$z3 = (hiddenval - rand1) \,/\, n$$

Sambil mencari z3, kita juga bisa langsung mencari nilai z2 dengan cara yang hampir sama dengan mencari nilai z3 tetapi dengan hint1.

$$hint1 = z3^{8} \cdot z2 + 0x1337 \cdot z2 \cdot z1^{2} + rand2$$
$$hint1 = z3^{8} \cdot z2 + n \cdot z1 \cdot 0x1337 + rand2$$
$$hint1 \bmod z3^{8} \approx n \cdot z1 \cdot 0x1337 + rand2$$
$$z3^{8} \cdot z2 \approx hint1 - (hint1 \bmod z3^{8})$$
$$z2 \approx [hint1 - (hint1 \bmod z3^{8})] \,/\, z3^{8}$$

Dengan begitu, jika sudah mendapatkan z2, maka kita bisa menghitung z1 serta melakukan dekripsi RSA seperti biasa untuk mendapatkan *secret* 🫠

Berikut full solvernya.

solver.py

```python
#!/usr/bin/env python3
from patsac import *

if not args.LOCAL:
    with open("nc.sh") as f:
        NC = f.read().strip().split()
        f.close()
    SRVR = NC[1]
    PORT = NC[2]
    r = remote(SRVR, PORT, level="warning")
else:
    r = process("./chall.py", level="debug")


def recon():
    global r, SRVR, PORT
    r.close()
    if not args.LOCAL:
```

```
        r = remote(SRVR, PORT, level="warning")
    else:
        r = process("./chall.py", level="debug")


def prev_prime(n):
    n -= 2
    n |= 1
    while not isPrime(n):
        n -= 2
    return n


def main():
    global r
    while True:
        e = 65537
        r.recvuntil(b"c = ")
        c = int(r.recvline(0))
        r.recvuntil(b"n = ")
        n = int(r.recvline(0))
        r.recvuntil(b"modd = ")
        modd = int(r.recvline(0))
        if modd % 4 != 3:
            print("Nilai modd belum cocok. Reconnecting...")
            recon()
            continue
        r.recvuntil(b"hint_1 = ")
        hint1 = int(r.recvline(0))
        r.recvuntil(b"hint_2 = ")
        hint2 = int(r.recvline(0))

        # hitung nilai hidden_val
        test = pow(hint2, divexact(modd + 1, 4), modd)
        test = pow(test, divexact(modd + 1, 4), modd)
        if test.bit_length() > 1280:
            test = -test % modd

        # prediksi nilai z3
        rand1 = test % n
        z123 = test - rand1
        z3 = divexact(z123, n)
        while z3.bit_length() >= 256 and rand1.bit_length() <= 1035:
            z3 = prev_prime(z3)
            z123 = n * z3
            rand1 = test - z123
            if z3.bit_length() == 256 and rand1.bit_length() == 1035:
                # cari nilai z2
                sisa = hint1 % pow(z3, 8)
                z32 = hint1 - sisa
                z2 = int(divexact(z32, pow(z3, 8)))
                # kalau n % z2 == 0, artinya inilah z3 yang benar
                if n % z2 == 0:
                    break
```

```
        # dekripsi rsa seperti biasa
        z1 = divexact(n, z2)
        phi = (z1 - 1) * (z2 - 1)
        d = pow(e, -1, phi)
        secret = str(pow(c, d, n)).encode()
        r.sendlineafter(b"What is the secret: ", secret)
        r.recvuntil(b"GG! Here is your prize:\n")
        print(r.recvline(0).decode())
        break

    return 0


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

Screenshot

```
patsac ~/ctf/2023/gemastik_qual/cry/naughtyboy
➜ ./solver.py
Nilai modd belum cocok. Reconnecting...
Nilai modd belum cocok. Reconnecting...
gemastik{643dba5a7e68ce4f395bc78ed366284baa146d2f7f76cd507128fc8a6dba910d}
```

Flag : gemastik{643dba5a7e68ce4f395bc78ed366284baa146d2f7f76cd507128fc8a6dba910d}

Web

🙏

PWN

🙏

Reverse Engineering

🙏