Write Up CTF JOINTS 2020

P.i.n.G

Ahmad Fauzzan Maghribi

Rio Darmawan

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

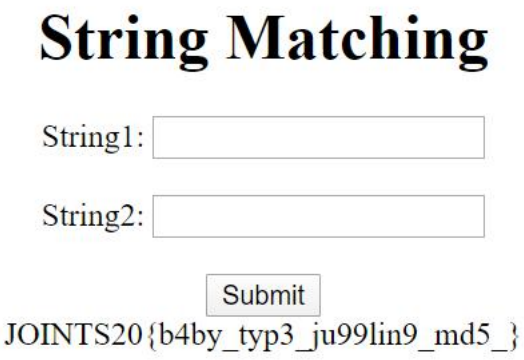
Web

ezStringMatching

Cara Pengerjaan

Diberikan sebuah website menampilkan 2 buah kolom inputan, yang mana saya simpulkan berdasarkan judul soal inimerupakan compare string dari 2 buah kolom inputan tersebut.

Kondisinya Ketika kita menginputkan 2 buah string yang sama , makan akan muncul pesan “Match bosqquee!!” dan Ketika kita menginputkan 2 buah string yang berbeda akan muncul pesan “Not Match bosqquee!!” dari sini saya simpulkan bahwa bagaimana memasukan string yang berbeda tetapi memiliki value yang sama. saya dapat referensi di https://stackoverflow.com/questions/22140204/why-md5240610708-is-equal-to-md5qnkcdzo



Flag

`JOINTS20{b4by_typ3_ju99lin9_md5_}`

Crypto

Classic

Cara Pengerjaan

Diberikan dua buah file, cipher dan soal.py. Langsung dilihat isi dari soal.py.

```
1  from string import printable
2  import random
3  from constants import flag,key
4
5  assert len(key)==15
6  prepare = ''.join(bin(ord(i))[2:].rjust(8,'0') for i in flag)
7
8  c = ''
9  for i in range(len(prepare)):
10     c += chr(ord(prepare[i]) + ord(key[i%len(key)]) - ord('A'))
11
12 f = open('cipher','w')
13 f.write(c)
14 f.close()
```

Pertama cari dulu keynya, kami menggunkan script berikut.

```
c = open('cipher','r').read()
flag = 'JOINTS20{'
cek = ''.join(bin(ord(i))[2:].rjust(8,'0') for i in flag)

k = ''
for u in range(len(c)):
    k += chr(ord(c[u]) + ord('A') - ord(cek[u%len(cek)]))

print k
```

Hasilnya didapat seperti dibawah ini, masih agak berantakan. Namun kami berasumsi bahwa key-nya adalah, ‘hopeyoufindthis’.

```
nangiiid@nangiiid-ubuntu ~/CTF/Joints/Penyihan/classic ‹ruby-2.6.5›
$ python solve.py
hoqeyoufinethishopfyoufinduhishopezoufindtiishopeyoufindthishopeyoufindthithopdzoufjndthisi
pofynuejocsgishoofyoufioctihrhpoezpugiodtiisiooeynuehndtihtgppeypufindthirhppeypufimdtgjsho
peyovfindthishoqdzoufhndtiishnoeyoufjnctgjshppeyntfinetijshoqeyntgindthhtiopeypvfinetgishop
eyougindthhrinoeypvfjme
```

Kemudian tambahkan script selanjutnya untuk melakukan decrypt seperti ini

```
c = open('cipher','r').read()
flag = 'JOINTS20{'
cek = ''.join(bin(ord(i))[2:].rjust(8,'0') for i in flag)

k = 'hopeyoufindthis'
# for u in range(len(c)):
#    k += chr(ord(c[u]) + ord('A') - ord(cek[u%len(cek)]))

p = ''
for i in range(len(c)):
    p += chr(ord(c[i]) - ord(k[i%len(k)]) + ord('A'))

pisah = [p[i:i+8] for i in range(0,len(p),8)]

h = ''
for b in pisah:
    h += chr(int(b,2))
print h
```

Jalankan, hasilnya flag.

```
nangiiid@nangiiid-ubuntu ~/CTF/Join
$ python solve.py
joints20{i_t0Ld_y0U_Cl4s5!cal_iS_b4d}
```

*Catatan: kami submit ngga bisa, ‘joints’ harus di uppercase semua.

Flag

JOINTS20{i_t0Ld_y0U_Cl4s5!cal_iS_b4d}

Crypto

Modulo

Cara Pengerjaan

Diberikan 3 buah file, flag.enc, modulo.py dan pub.key. Langsung buka file modulo.py yang merupakan sourcecode enkripsinya.

```
1  from sympy import mod_inverse
2  from sympy import isprime
3  from secret import flag
4  from Crypto.Util.number import getPrime
5
6  e = 65537
7  while True:
8      p = getPrime(1024)
9      q = mod_inverse(e,p)
10     if isprime(q):
11         break
12
13 N=p*q
14
15 m=int(flag.encode("hex"),16)
16 c=pow(m,e,N)
17 f=open("pub.key","a")
18 f.write("e:" +str(e)+"\n")
19 f.write("N:" +str(N))
20 f.close()
21
22 f=open("flag.enc","w")
23 f.write(str(c))
24 f.close()
```

Terlihat custom untuk generate nilai p dan q, dimana nilai q merupakan mod inverse dari e dan p. Terlihat cukup sepele namun itu bisa menjadi celah yang mempermudah menemukan nilai tersebut. Setelah googling, saya menemukan soal yang mirip dengan ini yaitu https://fireshellsecurity.team/tokyowesterns-revolutional-secure-angou/

Langsung saja, saya menjalakan script yang ada disitu untuk mendapatkan kembali nilai p dan q nya. Nilai e dan N sesuai dengan yang ada di file pub.key

```python
#!/usr/bin/python
from gmpy2 import isqrt
from sys import exit

e = 65537
N = 10277572973210015396114509978461325789039520167452352513097530
delta = 50

for x in range(1, e):
    q_approx = isqrt(N*x/e)
    for q in range(q_approx - delta, q_approx + delta):
        if N % q == 0:
            print 'P:', N/q
            print 'Q:', q
            exit(0)
```

```
nangiiid@nangiiid-ubuntu ~/CTF/Joints/Penyisihan/modulo ‹ruby-2.6.5›
└─$ python solver.py
P: 10545494429912192155690664892466939967373115437257452509789664345790596621533684508566667766771885571337706408269362781296791730944630331276152047274610243508220487913476476751021212332909344551598438431685859018800192111358177059642771372331752960671316701152662389263503845530519703612406438800032734247164 7
Q: 97459375105806133098230341823235366273075492592552204649730746005438890424195828816525921738535417441259472013680633067974439104605699056217705601313547038894959871819497878123175120707333747471079576259265622327950628774698577620038051250345547297242826182982812089798419353356503564154024930228784144511081
```

Setelah didapatkan nilai p dan q, langsung saja lakukan decrypt menggunakan script di bawah ini.

```python
from sympy import mod_inverse

q = 97459375105806133098230341823235366273075492592552204649730746
p = 10545494429912192155690664892466939967373115437257452509789664
e = 65537
c = 92827591149528087823381590992008442340271926207930093097714223

N = p*q

phi = (p-1)*(q-1)
d = mod_inverse(e,phi)
m = pow(c,d,N)

flag = hex(m)[2:-1].decode('hex')
print flag
```

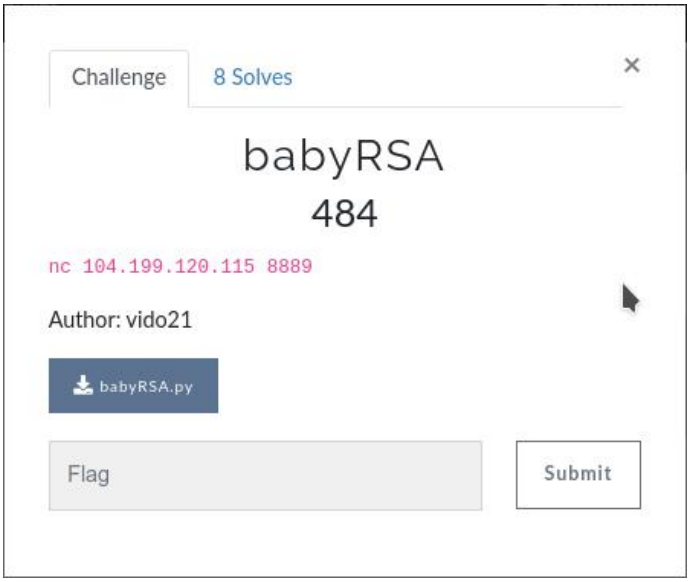
```
nangiiid@nangiiid-ubuntu ~/CTF/Joint
└─$ python decrypt.py
JOINTS20{M0dul4r_4r1thm3t1c}
```

Flag

JOINTS20{M0dul4r_4r1thm3t1c}

Crypto

BabyRSA



Cara Pengerjaan

Diberikan sebuah file sourcecode babyRSA.py dan koneksi netcat. Setelah dilihat-lihat, bagian penting pada source code adalah berikut ini.

```
def main():

    rsa1 = RSA(3,getPrime(2048),getPrime(2048))
    p=getPrime(2048)
    rsa2 = RSA(65537,p,next_prime(p))
    rsa3 = RSA(65537,getPrime(128),getPrime(128))
    random.seed(os.urandom(8))

    list_rsa=[rsa1,rsa2,rsa3]
    random.shuffle(list_rsa)

    print("""
Complete the following three questions
""")
    try:
        for i in range(3):
            m=int(os.urandom(16).encode("hex"),16)
            print("e :" + list_rsa[i].getExponent())
            print("N :" + list_rsa[i].getModulus())
            print("c :" + list_rsa[i].encrypt(m))
            ans=raw_input("m >> ").strip()
            print ans

            if ans!=str(m):
                print("Wrong !!")
                exit(0)

        print flag
```

Intinya kita disuruh menghitung 3 buah nilai m, dari 3 algortima rsa yang berbeda. Karena koneksi netcat soal tidak ada batasan waktu, maka kami memutuskan mengerjakannya secara manual tanpa otomasi.

Rsa1, meskipun nilai p dan q besar, namun nilai e kecil maka nilai m bisa didapatkan dengan script berikut.

```
import gmpy
c = 50639156990056809733821909365827368020933071809083825532795388500832191215
m = gmpy.root(c,3)[0]
print m
```

```
┌─nangiiid@nangiiid-ubuntu ~/CTF/Joints/Penyisihan/BabyRSA
└─$ python rsa1.py
79706821155208655878263470561756119905
```

Hasil nilai m dimasukkan dan valid,

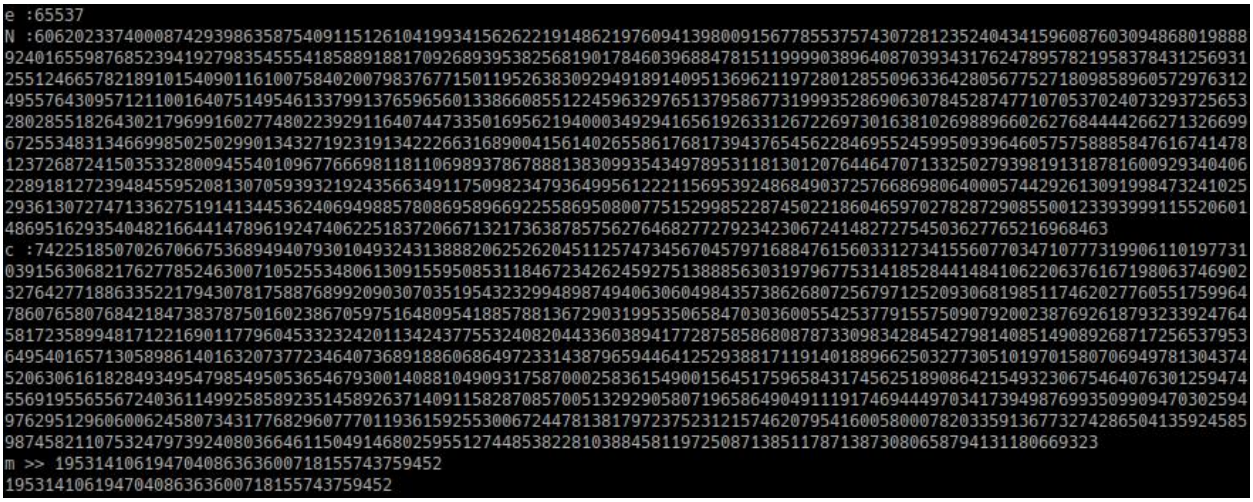
```
└─$ nc 104.199.120.115 8889                                                    130 ↵

Complete the following three questions

e :3
N :38231998047358136656685038192842694310785442852795927721244912753472213904045320203142603371674557614138526274967236604653
69774877374150128529033547965900945394563511796302518424153259923243826728798775897444153551052725619453773987923570521643919
00209005614619730644745974774890281216394169161322418982842736319620992724153207477317765856337573473094335305496009415153721
28070884237170422877419878593384847728867736511327360091531498574763679225596914930157593484460347058316969811559995269237769
83648672040966047723662485888850006048519793562989762922990269836595730985629159013334987432789915751596020674547938013148 3266
56330080837900516791060976819978853317300840089584261426765093138797867984202757590834023477164789088955588163578106762721041 5
87935497310001413497852852445984553846185326656312182660621636353332970465955093538647087378155863189922035896494092215973519
38393750520326806221549192781333992163953972481998235625904842920207851289047974547174247085605982415089730634993385663359059
18598380728302979607923170498260166026619744669621170175766554487288886381832288786975213050127886951237843040751907548688080
632356577068902023333904474218726009512271798215561354394424959428671335186631866201056738768869951575874902419
c :506391569900568097338219093658273680209330718090838255327953885008321912154215451641533890979559854513042948142625
m >> 79706821155208655878263470561756119905
79706821155208655878263470561756119905
```

Rsa2, nilai q merupakan nilai next prime dari p. Sehingga nilai p dan q bisa didaptkan kembali. Lalu tinggal lakukan dekripsi untuk mendapatkan nilai m.

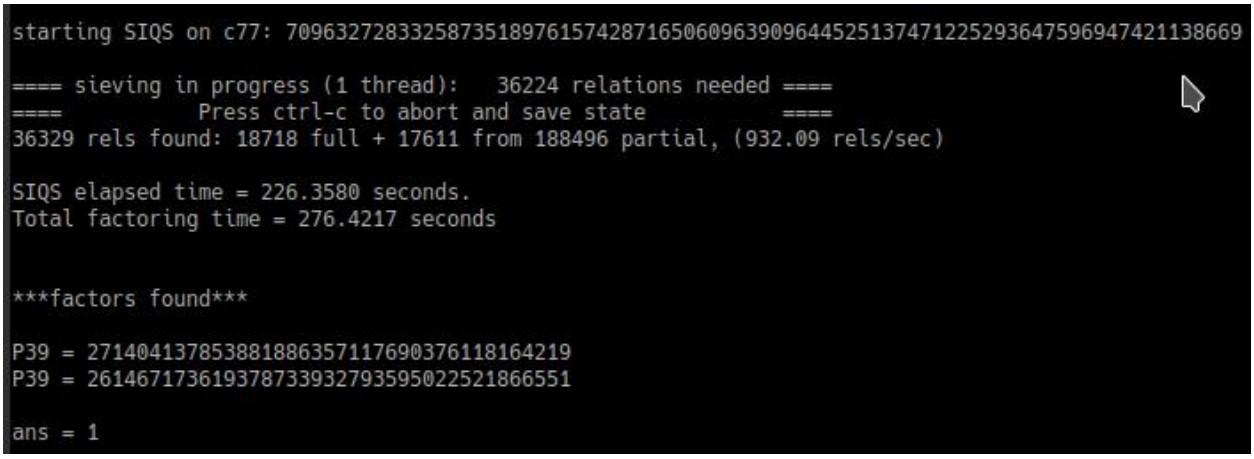
```
import gmpy2
from sympy import mod_inverse

n = 60620233740008742939863587540911512610419934156262219148621976094
c = 74225185070267066753689494079301049324313888206252620451125747345

sq,b = gmpy2.iroot(n,2)
while n%sq != 0:
    sq += 1

p = sq
q = n /sq

phi = (p-1)*(q-1)
d = mod_inverse(65537,phi)

m = pow(c,d,n)
print m
```

```
┌─nangiiid@nangiiid-ubuntu ~/CTF/Joints/Penyisihan/BabyRSA
└─$ python rsa2.py
195314106194704086363600718155743759452
```

Setelah mendapat nilai m, dimasukkan valid.



Rsa3, nilai p dan q yang dibangkitkan cukup kecil sehingga memungkinkan untuk mencari nilai p dan q nya. Disini kamu menggunakan tools https://github.com/DarkenCode/yafu untuk mencari nilai p dan q. Setelah menunggu beberapa saat didapatkan nilai p dan q nya.



Langsung saja bikin script dekripsinya,

```
1   from sympy import mod_inverse
2
3   p = 271404137853881886357117690376118164219
4   q = 261467173619378733932793595022521866551
5   n = p*q
6   c = 150910115389850407663068321389001188265430762550770842084758471711
7
8   phi = (p-1)*(q-1)
9   d = mod_inverse(65537,phi)
10  m = pow(c,d,n)
11
12  print m
```

```
nangiiid@nangiiid-ubuntu ~/CTF/Joints/Penyisihan/BabyRSA
$ python rsa3.py
241521762621873345065652589412195395981
```

Setelah mendapat nilai m nya, langsung masukkan dan muncul flagnya.

```
e :65537
N :709632728332587351897615742871650609639096445251374712252936475969474211386691
c :150910115389850407663068321389001188265430762550770842084758471710091474336971
m >> 241521762621873345065652589412195395981
241521762621873345065652589412195395981
JOINST20{Common_Attacks_on_RSA}
```

Flag

JOINST20{Common_Attacks_on_RSA}

Reverse Engineering

crackme

Cara Pengerjaan

Diberikan sebuah file crackme dan koneksi netcat, file crackme merupakan file ELF 64-bit not stripped. Langsung decompile menggunakan IDA Pro, ternyata ada sebuah fungsi pengecekan.

```
1  _BOOL8 __fastcall checker(char *a1)
2  {
3    if ( a1[20] - *a1 != 24 )
4      return 0LL;
5    if ( a1[8] + a1[5] != 126 )
6      return 0LL;
7    if ( a1[14] * a1[5] != 3696 )
8      return 0LL;
9    if ( a1[21] - a1[1] != 33 )
10     return 0LL;
11   if ( a1[10] - *a1 != 2 )
12     return 0LL;
13   if ( a1[17] - *a1 != 19 )
14     return 0LL;
15   if ( a1[17] * a1[1] != 3848 )
16     return 0LL;
17   if ( a1[4] + a1[6] != 123 )
18     return 0LL;
19   if ( a1[13] * a1[16] != 4488 )
20     return 0LL;
21   if ( a1[1] * a1[6] != 2600 )
22     return 0LL;
23   if ( a1[13] * a1[23] != 3536 )
24     return 0LL;
25   if ( a1[8] - a1[5] != 14 )
26     return 0LL;
27   if ( a1[15] + a1[5] != 123 )
28     return 0LL;
29   if ( a1[20] - a1[17] != 5 )
30     return 0LL;
31   if ( a1[17] + a1[16] != 140 )
```

Langsung saja buat solvernya menggunakan z3 solver, berikut script yang kami gunakan.

Kode

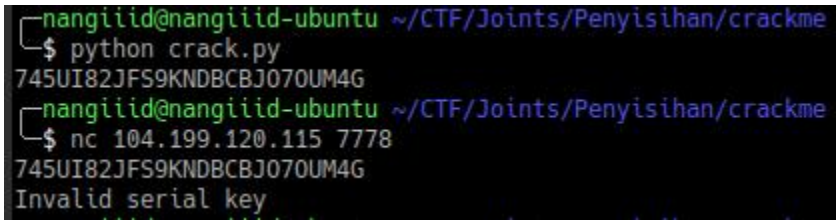
crack.py

```
from z3 import *
a1 = [BitVec('x{}'.format(x), 32) for x in range(25)]
s = Solver()


for i in range(len(a1)):
  s.add(a1[i] >= 0x20)
  s.add(a1[i] < 0x7f)


s.add(a1[20] - a1[0] == 24)
s.add(a1[8] + a1[5] == 126)
s.add(a1[14] * a1[5] == 3696)
s.add(a1[21] - a1[1] == 33)
s.add(a1[10] - a1[0] == 2)
s.add(a1[17] - a1[0] == 19)
s.add(a1[17] * a1[1] == 3848)
s.add(a1[4] + a1[6] == 123)
s.add(a1[13] * a1[16] == 4488)
s.add(a1[1] * a1[6] == 2600)
s.add(a1[13] * a1[23] == 3536)
s.add(a1[8] - a1[5] == 14)
s.add(a1[15] + a1[5] == 123)
s.add(a1[20] - a1[17] == 5)
s.add(a1[17] + a1[16] == 140)
s.add(a1[16] + a1[14] == 132)
```

```
s.add(a1[3] * a1[6] == 4250)
s.add(a1[18] + a1[14] == 145)
s.add(2 * a1[13] == 136)
s.add(a1[17] - a1[10] == 17)
s.add(a1[11] + a1[8] == 145)
s.add(a1[9] + a1[1] == 135)
s.add(a1[11] + a1[24] == 146)
s.add(a1[3] - a1[7] == 11)
s.add(a1[0] - a1[2] == 2)
s.add(a1[11] - a1[13] == 7)
s.add(a1[3] + a1[4] == 158)
s.add(a1[3] - a1[16] == 19)
s.add(a1[4] - a1[14] == 7)
s.add(a1[12] * a1[1] == 4056)
s.add(a1[20] + a1[8] == 149)
s.add(a1[9] - a1[4] == 10)
s.add(a1[9] - a1[6] == 33)
s.add(a1[9] * a1[13] == 5644)
s.add(a1[16] + a1[5] == 122)
s.add(a1[16] - a1[10] == 9)
s.add(a1[17] + a1[24] == 145)
s.add(a1[20] - a1[13] == 11)
s.add(a1[18] * a1[11] == 5925)
s.add(a1[21] * a1[23] == 4420)
s.add(a1[22] * a1[7] == 5698)
s.add(a1[15] - a1[19] == 12)
s.add(a1[16] - a1[1] == 14)
```

```
s.add(a1[3] - a1[13] == 17)
s.add(a1[12] * a1[8] == 5460)
s.add(a1[21] * a1[13] == 5780)
s.add(a1[7] * a1[1] == 3848)
s.add(a1[22] + a1[6] == 127)
s.add(a1[13] + a1[5] == 124)
s.add(a1[24] + a1[1] == 123)

if s.check() == z3.sat:
    model = s.model()
    solution = ''.join([chr(int(str(model[a1[i]]))) for i in range(25)])
    print solution
```

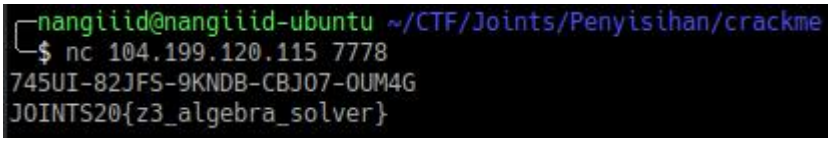
Setelah mendapatkan hasilnya, coba masukkan ke server, ternyata masih gagal.



Coba cek ulang apa yang salah, ternyata input yang diminta dipisahkan dengan ‘-’ setiap 5 karakter.

```
10  v9 = __readfsqword(0x28u);
11  __isoc99_scanf("%5c-%5c-%5c-%5c-%5c", &v4, v5, v6, &v6[5], &v7);
12  v8 = 0;
13  if ( (unsigned int)checker(&v4) )
14    unlock(&v4);
15  else
16    printf("Invalid serial key");
17  return 0;
18}
```

Coba masukkan lagi dengan melakukan pemisahan, didapatkan flagnya



Flag

`JOINTS20{z3_algebra_solver}`

Reverse Engineering

RansomPy

Cara Pengerjaan

Diberikan sebuah file RansomPy.zip yang berisi 3 buah file, yaitu RansomPy.pyc, flag.pdf.enc dan encrypted. File RansomPy.pyc merupakan python 3.6 byte-compiled, maka decompile dulu menggunakan uncompyle.

```
# uncompyle6 version 3.6.7
# Python bytecode 3.6 (3379)
# Decompiled from: Python 2.7.17 (default, Apr 15 2020, 17:20:14)
# [GCC 7.5.0]
# Warning: this version has problems handling the Python 3 "byte" type in cons

# Embedded file name: ransom.py
# Compiled at: 2020-05-02 14:13:22
# Size of source mod 2**32: 763 bytes
import random, time, os

def enc(O000000000000000):
    O0000000000000000 = open(O000000000000000, 'rb')
    O0000000000000000 = open(O000000000000000 + '.enc', 'wb')
    O0000000000000000 = int(time.time())
    random.seed(O0000000000000000)
    O0000000000000000.write('ransom')
    O0000000000000000 = ''
    for O0000000000000000 in O0000000000000000.read():
        O0000000000000000 += chr(O0000000000000000 ^ random.randint(0, 255))

    O0000000000000000.write(O0000000000000000.encode('charmap'))
    O0000000000000000.write('ransom')
    O0000000000000000.close()


for file in os.listdir('.'):
    if file.startswith('flag') and not file.endswith('enc'):
        enc(file)
        open('encrypted.txt', 'a+').write('{} :)\n'.format(file))
# okay decompiling RansomPy.pyc
nangiiid@nangiiid-ubuntu ~/CTF/Joints/Penyisihan/ransompy ‹ruby-2.6.5›
$
```

Karena hasilnya masih kurang jelas, maka kami rename variabel-variabel menjadi seperti ini.

```python
import random, time, os

def enc(input):
    plain = open(input, 'rb')
    hasil = open(input + '.enc', 'wb')
    waktu = int(time.time())
    random.seed(waktu)
    hasil.write('ransom')
    cipher = ''
    for i in plain.read():
        cipher += chr(i ^ random.randint(0, 255))

    hasil.write(cipher.encode('charmap'))
    hasil.write('ransom')
    hasil.close()


for file in os.listdir('.'):
    if file.startswith('flag') and not file.endswith('enc'):
        enc(file)
        open('encrypted.txt', 'a+').write('{} :)\n'.format(file))
```

Setelah dilihat, ternyata script tersebut melakukan enkripsi xor menggunakan random yang seednya merupakan waktu saat script tersebut dieksekusi tersebut dibuat. Karena waktu pembuatan file sama dengan waktu eksekusi, kami lihat kapan waktu file flag.pdf.enc dibuat.

```
nangiiid@nangiiid-ubuntu ~/CTF/Joints/Penyisihan/ransompy ‹ruby-2.6.5›
└─$ stat flag.pdf.enc
  File: flag.pdf.enc
  Size: 100085          Blocks: 200        IO Block: 4096   regular file
Device: 802h/2050d      Inode: 12463716    Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/nangiiid)   Gid: ( 1000/nangiiid)
Access: 2020-05-03 15:25:14.999393623 +0700
Modify: 2020-05-03 04:01:11.000000000 +0700
Change: 2020-05-03 15:25:14.023783618 +0700
 Birth: -
```

Terlihat pada bagian modify, file tersbut dibuat pada 3 mei 2020 pukul 4:01:11 UTC+7. Kemudian kita cari unixtimestamp nya.

```
nangiiid@nangiiid-ubuntu ~/CTF/Joints/Penyisihan/ransompy
└─$ date -d "2020-05-03 04:01:11" '+%s'
1588453271
```

Setelah didapatkan, kita buat script untuk melakukan dekripsi dengan random.seednya sesuai unixtimestamp yang sudah didapat. Berikut script yang kami gunakan.

```python
import random

def dec(input):
    cipher = open(input, 'rb')
    hasil = open('flag.pdf', 'wb')
    waktu = 1588453271 ## liat dari stat file flag.pdf.enc
    random.seed(waktu)

    akhir = ''
    for i in cipher.read()[6:-6]:
        akhir += chr(i ^ random.randint(0, 255))

    hasil.write(akhir.encode('charmap'))
    hasil.close()


dec('flag.pdf.enc')
```

Setelah itu jalankan, dan buka file pdfnya.

```
nangiiid@nangiiid-ubuntu ~/CTF/Joints/Penyisihan/ransompy
$ python3 solver.py
nangiiid@nangiiid-ubuntu ~/CTF/Joints/Penyisihan/ransompy
$ evince flag.pdf
```

Flag

JOINTS20{EZ_Random_S33d}