

Write Up CTF JOINTS 2020

P.i.n.G



Ahmad Fauzzan Maghribi

Rio Darmawan

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

Web

ezStringMatching

Challenge

46 Solves

×

ezStringMatching

100

<http://ctf.joints.id:40002>

Author: vido21

Flag

Submit

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.

String Matching

String1:

String2:

Submit

Kondisinya Ketika kita menginputkan 2 buah string yang sama , maka 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>

String Matching

String1:

String2:

JOINTS20{b4by_typ3_ju99lin9_md5_}

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 menggunakan script berikut.

```

1 c = open('cipher','r').read()
2 flag = 'JOINTS20{'
3 cek = ''.join(bin(ord(i))[2:].rjust(8,'0') for i in flag)
4
5 k = ''
6 for u in range(len(c)):
7     k += chr(ord(c[u]) + ord('A') - ord(cek[u%len(cek)]))
8
9 print k

```

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

```

nangliid@nangliid-ubuntu ~/CTF/Joins/Python/classic <ruby-2.6.5>
$ python solve.py
hoqeyoufinethishopfyofinduhishopezoufindtiishopeyoufindthishopeyoufindthithopdzoufjndthisi
pofynuejocsgishoofyoufiocthrhpoezpugiodtiisiooeynehndtihtgpeypufindthirhppeypufimdtgjsho
peyovfindthishoqdzoufhndtiishnoeyoufjntgjshppeyntfinetijshoeyntgindthhtiopeypvfinetgishop
eyougindthhrinoeypvfjme

```

Kemudian tambahkan script selanjutnya untuk melakukan decrypt seperti ini

```

1 c = open('cipher','r').read()
2 flag = 'JOINTS20{'
3 cek = ''.join(bin(ord(i))[2:].rjust(8,'0') for i in flag)
4
5 k = 'hopeyoufindthis'
6 # for u in range(len(c)):
7 #     k += chr(ord(c[u]) + ord('A') - ord(cek[u%len(cek)]))
8
9 p = ''
10 for i in range(len(c)):
11     p += chr(ord(c[i]) - ord(k[i%len(k)]) + ord('A'))
12
13 pisah = [p[i:i+8] for i in range(0,len(p),8)]
14
15 h = ''
16 for b in pisah:
17     h += chr(int(b,2))
18 print h

```

Jalankan, hasilnya flag.

```

nangliid@nangliid-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 menjalankan script yang ada disitu untuk mendapatkan kembali nilai p dan q nya. Nilai e dan N sesuai dengan yang ada di file pub.key


```

1  #!/usr/bin/python
2  from gmpy2 import isqrt
3  from sys import exit
4
5  e = 65537
6  N = 102775729732100153961145099784613257890395201674523525130975308
7  delta = 50
8
9  for x in range(1, e):
10     q_approx = isqrt(N*x/e)
11     for q in range(q_approx - delta, q_approx + delta):
12         if N % q == 0:
13             print 'P:', N/q
14             print 'Q:', q
15             exit(0)
16

```

```

nangiid@nangiid-ubuntu ~/CTF/Joints/Penyisihan/modulo <ruby-2.6.5>
$ python solver.py
P: 1054549442991219215569066489246693996737311543725745250978966434579059662153368450856666
7766771885571337706408269362781296791730944630331276152047274610243508220487913476476751021
2123329093445515984384316858590188001921113581770596427713723317529606713167011526623892635
038455305197036124064388000327342471647
Q: 9745937510580613309823034182323536627307549259255220464973074600543889042419582881652592
1738535417441259472013680633067974439104605699056217705601313547038894959871819497878123175
207073337474710795762592656223279506287746985776200380512503455472972428261829828120897984
19353356503564154024930228784144511081

```

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

```

1  from sympy import mod_inverse
2
3  q = 974593751058061330982303418232353662730754925925522046497307460
4  p = 105454944299121921556906648924669399673731154372574525097896643
5  e = 65537
6  c = 928275911495280878233815909920084423402719262079300930977142237
7
8  N = p*q
9
10 phi = (p-1)*(q-1)
11 d = mod_inverse(e,phi)
12 m = pow(c,d,N)
13
14 flag = hex(m)[2:-1].decode('hex')
15 print flag

```

```

nangiid@nangiid-ubuntu ~/CTF/Joint
$ python decrypt.py
JOINTS20{M0dul4r_4r1thm3t1c}

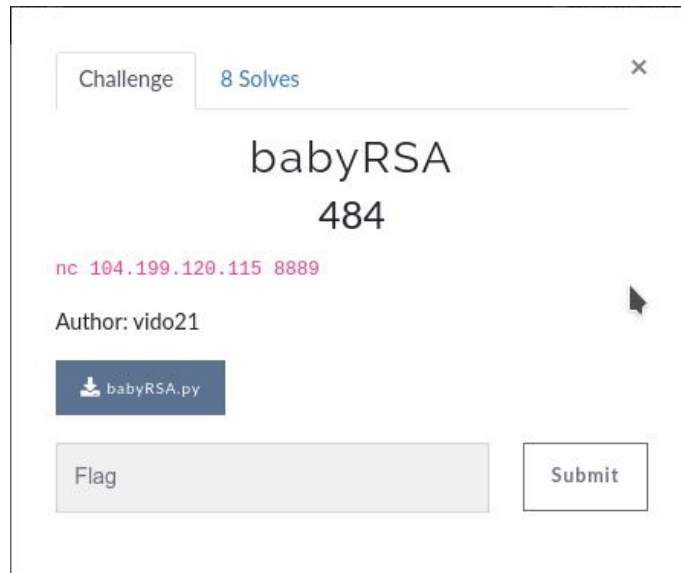
```

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.

```
44 def main():
45
46     rsa1 = RSA(3, getPrime(2048), getPrime(2048))
47     p = getPrime(2048)
48     rsa2 = RSA(65537, p, next_prime(p))
49     rsa3 = RSA(65537, getPrime(128), getPrime(128))
50     random.seed(os.urandom(8))
51
52     list_rsa = [rsa1, rsa2, rsa3]
53     random.shuffle(list_rsa)
54
55     print("""
56 Complete the following three questions
57 """)
58     try:
59         for i in range(3):
60             m = int(os.urandom(16).encode("hex"), 16)
61             print("e : " + list_rsa[i].getExponent())
62             print("N : " + list_rsa[i].getModulus())
63             print("c : " + list_rsa[i].encrypt(m))
64             ans = raw_input("m >> ").strip()
65             print ans
66
67             if ans != str(m):
68                 print("Wrong !!")
69                 exit(0)
70
71     print flag
```


Intinya kita disuruh menghitung 3 buah nilai m, dari 3 algoritma 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.

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

```
nangiid@nangiid-ubuntu ~/CTF/Joints/Pyenvisihan/BabyRSA
$ python rsa1.py
79706821155208655878263470561756119905
```

Hasil nilai m dimasukkan dan valid,

```
$ nc 104.199.120.115 8889
Complete the following three questions
e :3
N :38231998047358136656685038192842694310785442852795927721244912753472213904045320203142603371674557614138526274967236604653
69774877374150128529033547965900945394563511796302518424153259923243826728798775897444153551052725619453773987923570521643919
00209005614619730644745974774890281216394169161322418982842736319620992724153207477317765856337573473094335305496009415153721
2807088423717042287741987859338484772886773651132736009153149857476367922559691493015759348446034705831696981155995269237769
8364867204096604772366248588850060485197935629897629229902608365957300856291590133349874327899157515960206745470380131483266
56330080837900516791060976819978853317300840089584261426765093138797867984202757590834023477164789088955881635781067627210415
879354973100014134978528524459845538461853266563121826606216363533297046595509353864708737815586318992203589649402215973519
38393750520326806221549192781333992163953972481998235625904842920207851289047974547174247085605982415089730634993385663359059
18598380728302979607923170498260166026619744669621170175766554487288886381832288786975213050127886951237843040751907548688080
63235657706890202333904474218726009512271798215561354394424959428671335186631866201056738768869951575874902419
c :506391569900568097338219093658273680209330718090838255327953885008321912154215451641533890979559854513042948142625
m >> 79706821155208655878263470561756119905
79706821155208655878263470561756119905
```

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

```
1 import gmpy2
2 from sympy import mod_inverse
3
4 n = 60620233740008742939863587540911512610419934156262219148621976094
5 c = 74225185070267066753689494079301049324313888206252620451125747345
6
7 sq,b = gmpy2.iroot(n,2)
8 while n%sq != 0:
9     sq += 1
10
11 p = sq
12 q = n /sq
13
14 phi = (p-1)*(q-1)
15 d = mod_inverse(65537,phi)
16
17 m = pow(c,d,n)
18 print m
```

```
nangiid@nangiid-ubuntu ~/CTF/Joints/Pyenvisihan/BabyRSA
$ python rsa2.py
195314106194704086363600718155743759452
```

Setelah mendapat nilai m, dimasukkan valid.

```
e :65537
N :60620233740008742939863587540911512610419934156262219148621976094139800915677855375743072812352404341596087603094868019888
92401655987685239419279835455541858891881709268939538256819017846039688478151199990389640870393431762478957821958378431256931
25512466578218910154090116100758402007983767715011952638309294918914095136962119728012855096336428056775271809858960572976312
49557643095712110016407514954613379913765965601338660855122459632976513795867731999352869063078452874771070537024073293725653
28028551826430217969916027748022392911640744733501695621940003492941656192633126722697301638102698896602627684444266271326699
67255348313466998502502990134327192319134222663168900415614026558617681739437654562284695524599509396460575758885847616741478
12372687241503533280094554010967766698118110698937867888138309935434978953118130120764464707133250279398191318781600929340406
22891812723948455952081307059393219243566349117509823479364995612221156953924868490372576686980640005744292613091998473241025
29361307274713362751914134453624069498857808695896692255869508007751529985228745022186046597027828729085500123393999115520601
486951629354048216644147896192474062251837206671321736387857562764682772792342306724148272754503627765216968463
c :74225185070267066753689494079301049324313888206252620451125747345670457971688476156033127341556077034710777319906110197731
03915630682176277852463007105255348061309155950853118467234262459275138885630319796775314185284414841062206376167198063746902
32764277188633522179430781758876899209030703519543232994898749406306049843573862680725679712520930681985117462027760551759964
78607658076842184738378750160238670597516480954188578813672903199535065847030360055425377915575090792002387692618793233924764
5817235899481712216901177960453323242011342437753240820443360389417728758586808787330983428454279814085149089268717256537953
64954016571305898614016320737723464073689188606864972331438796594464125293881711914018896625032773051019701580706949781304374
52063061618284934954798549505365467930014088104909317587000258361549001564517596584317456251890864215493230675464076301259474
55691955655672403611499258589235145892637140911582870857005132929058071965864904911191746944497034173949876993509909470302594
97629512960600624580734317768296077701193615925530067244781381797237523121574620795416005800078203359136773274286504135924585
98745821107532479739240803664611504914680259551274485382281038845811972508713851178713873080658794131180669323
m >> 195314106194704086363600718155743759452
195314106194704086363600718155743759452
```

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.

```
starting SIQS on c77: 70963272833258735189761574287165060963909644525137471225293647596947421138669

==== sieving in progress (1 thread): 36224 relations needed ====
==== Press ctrl-c to abort and save state ====
36329 rels found: 18718 full + 17611 from 188496 partial, (932.09 rels/sec)

SIQS elapsed time = 226.3580 seconds.
Total factoring time = 276.4217 seconds

***factors found***

P39 = 271404137853881886357117690376118164219
P39 = 261467173619378733932793595022521866551

ans = 1
```

Langsung saja bikin script dekripsinya,

```
1 from sympy import mod_inverse
2
3 p = 271404137853881886357117690376118164219
4 q = 261467173619378733932793595022521866551
5 n = p*q
6 c = 15091011538985040766306832138900118826543076255077084208475847171
7
8 phi = (p-1)*(q-1)
9 d = mod_inverse(65537,phi)
10 m = pow(c,d,n)
11
12 print m
```

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

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

```
e :65537
N :70963272833258735189761574287165060963909644525137471225293647596947421138669
c :15091011538985040766306832138900118826543076255077084208475847171009147433697
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 80018 __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.

```

nangiid@nangiid-ubuntu ~/CTF/Joints/Pyisihan/crackme
$ python crack.py
745UI82JFS9KNDBCBJ070UM4G
nangiid@nangiid-ubuntu ~/CTF/Joints/Pyisihan/crackme
$ nc 104.199.120.115 7778
745UI82JFS9KNDBCBJ070UM4G
Invalid serial key

```

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

```
nangiid@nangiid-ubuntu ~/CTF/Joints/Penyisihan/crackme  
$ nc 104.199.120.115 7778  
745UI-82JFS-9KNDB-CBJ07-0UM4G  
JOINTS20{z3_algebra_solver}
```

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 uncompile.

```
# uncompile6 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 constants
# 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(0000000000000000):
    0000000000000000 = open(0000000000000000, 'rb')
    0000000000000000 = open(0000000000000000 + '.enc', 'wb')
    0000000000000000 = int(time.time())
    random.seed(0000000000000000)
    0000000000000000.write('ransom')
    0000000000000000 = ''
    for 0000000000000000 in 0000000000000000.read():
        0000000000000000 += chr(0000000000000000 ^ random.randint(0, 255))

    0000000000000000.write(0000000000000000.encode('charmap'))
    0000000000000000.write('ransom')
    0000000000000000.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
nangiliid@nangiliid-ubuntu ~/CTF/Joins/Penyisihan/ransomp y <ruby-2.6.5>
$
```

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

```

1  import random, time, os
2
3  def enc(input):
4      plain = open(input, 'rb')
5      hasil = open(input + '.enc', 'wb')
6      waktu = int(time.time())
7      random.seed(waktu)
8      hasil.write('ransom')
9      cipher = ''
10     for i in plain.read():
11         cipher += chr(i ^ random.randint(0, 255))
12
13     hasil.write(cipher.encode('charmap'))
14     hasil.write('ransom')
15     hasil.close()
16
17
18     for file in os.listdir('.'):
19         if file.startswith('flag') and not file.endswith('.enc'):
20             enc(file)
21             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.

```

nangliid@nangliid-ubuntu ~/CTF/Joints/Pyenisihan/ransompv <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/nangliid)  Gid: ( 1000/nangliid)
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 tersebut dibuat pada 3 mei 2020 pukul 4:01:11 UTC+7. Kemudian kita cari unixtimestamp nya.

```

nangliid@nangliid-ubuntu ~/CTF/Joints/Pyenisihan/ransompv
$ 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.

```

1  import random
2
3  def dec(input):
4      cipher = open(input, 'rb')
5      hasil = open('flag.pdf', 'wb')
6      waktu = 1588453271 ## liat dari stat file flag.pdf.enc
7      random.seed(waktu)
8
9      akhir = ''
10     for i in cipher.read()[6:-6]:
11         akhir += chr(i ^ random.randint(0, 255))
12
13     hasil.write(akhir.encode('charmap'))
14     hasil.close()
15
16
17 dec('flag.pdf.enc')

```

Setelah itu jalankan, dan buka file pdfnya.

```

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

```

JOINTS20{EZ_Random_S33d}



Flag

JOINTS20{EZ_Random_S33d}