1. 存在off\_by\_one漏洞：
2. 信息泄露：

fastbin chunk最后一位会补0，不能通过show函数泄露地址。所以通过分配三个small chunk，free前两个，第一个chunk的fd会指向main\_arena,bk会指向free掉的第二个chunk的地址。

1. 漏洞利用

利用了fastbin而非unlink，因为edit时先free再malloc的，会导致异常。也没发现存放size的地址。

《1》最终要实现的情况：free a，free b，free a

Fast bin的空闲chunk通过单链表fd指针相连：a->b->a->0

《2》之后进行四次malloc

《3》第一次malloc得到a；

//修改其fd指针指向malloc\_hook地址相关，为了过安全检查，修改为malloc\_hook-0x23(申请的堆大小为0x71,此处的0x7f可以用来利用)

第二次malloc得到b

第三次malloc得到a

第四次malloc得到malloc\_hook-0x23，所以malloc\_hook-0x13地址起始地址可被利用。

// malloc\_hook-0x13地址可以通过one\_gadget得到的地址覆盖malloc\_hook，之后执行malloc函数就实现成功。

1. 漏洞构造(即构造重复的fast\_chunk)

构造重叠堆：

1. 构造small chunk大小为0x100，free此small chunk，会产生一次堆的合并；
2. small chunk地址-pre\_size处的chunk与small chunk进行合并；
3. 存在限制：伪造堆块->fd->bk指向伪造堆块，伪造堆块->bk->fd指向伪造堆块

|  |  |  |
| --- | --- | --- |
| 块a\_addr | Presize | 0x101块a |
|  | Fd(块b\_addr) | Bk(块b\_addr) |
| 块b\_addr |  | 0x70块 b |
|  | Fd(块a\_addr) | 块a\_addr |
| 块c\_addr | presize为a+b | 0x100 块c |

构造如图所示，这此时free块c，再malloc一个small chunk一个fast chunk，就可以分配到已经分配出来的块b。即构造了漏洞利用中《1》的背景条件。

这里可以利用edit功能里的free再malloc实现。

此时保存堆地址数组里就会有两个相同的块b，可以实现刚才的情况