double free

原理：double Free其实就是同一个指针free两次。虽然一般把它叫做double free。其实只要是free一个指向堆内存的指针都有可能产生可以利用的漏洞

double free的原理其实和堆溢出的原理差不多，都是通过unlink这个双向链表删除的宏来利用的。只是double free需要由自己来伪造整个chunk并且欺骗操作系统。

漏洞程序：

void Malloc();

void Free();

void Edit();

void Print();

漏洞程序主要含有上面四个函数，作用分别是创建堆块，删除堆块，修改堆块内容，打印堆块内容。

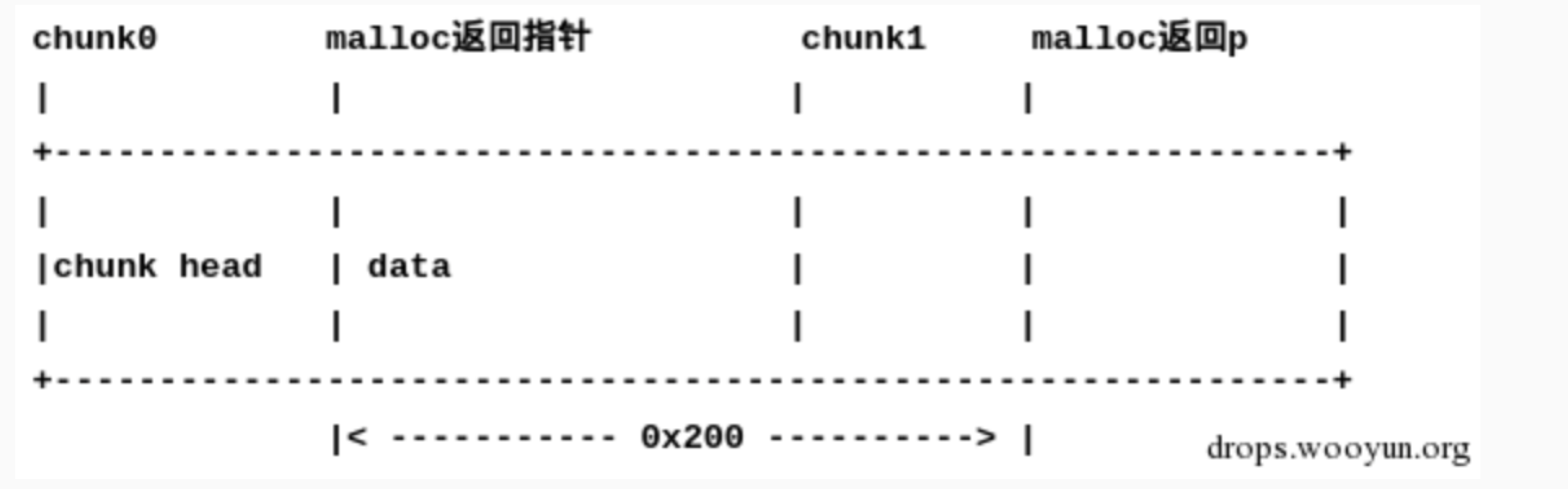
攻击程序：

要利用Double Free的漏洞。我们就要让系统进行unlink的操作，达到篡改指针的目的。但是一般的情况下，我们两次释放同一块内存会被操作系统给检测出来，怎么欺骗过操作系统才是最重要的。漏洞程序在free的时候很明显的没有检验指针的有效性，而且没有在free之后将野指针清零。而且可以任意的指定每一个chunk的大小。所以可以很容易的构造double free。我们首先构造一个野指针。

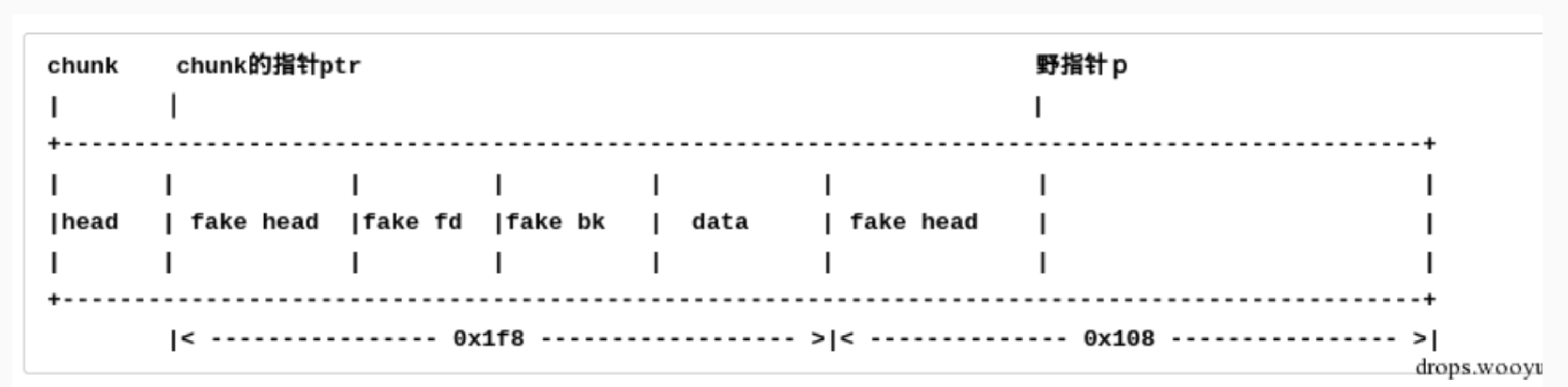
>malloc(504)

>malloc(512)

然后释放这2块内存。这样子我们就可以在距离第一个指针偏移量为0×200的地方有了一个野指针。



我们留下了一个野指针p指向偏移为0×200的地方。然后我们需要做的就是伪造chunk。再free野指针p。首先是申请一块更大的内存，大小应该等于我们刚才申请的内存的总和。最好和刚才2块内存大小总和一样，如果不一样大也也可以，就是待会伪造第二快内存块的大小的时候，要让伪造的大小等于我们申请的chunk的大小，否则会无法绕过检查。会被系统检查出double free。



可以看到现在我们在内存中伪造了出了２个chunk。它们的结构就像图中我们看到的样子。首先是第一个chunk的chunk头部分。我们分别填上了0x０和0x1f9代表了前一个chunk正在使用，当前chunk的大小是１f8。然后就是伪造的双向链表指针了。为了绕过unlink中的检查，这里需要稍微构造一下这个双向链表的指针了。就可以绕过保护了。

结合而源代码，可以看到现在FD->bk的值正好也是指向我们伪造的chunk的头部分。然后我们在野指针ｐ的前面又伪造了一个chunk头。prve\_size部分填0x1f8正好是前一个伪造chunk的大小。然后size部分填的是0×108。这样的话两个chunk正将我们申请的空间填满。然后第二个伪造chunk的size中最低位的flag置为０．这样free指针ｐ的时候，就会将前一个伪造的chunk给unlink。

现在，只要在free一次指针ｐ。就可以触发漏洞了。这时候，我们的操作系统不会报错，而且我们本来正常的指针ptr已经变成了ptr-0xc。这要如果我们如果调用Edit函数来修改这个chunk的话，就可以干各种各样的事情了。