《漏洞利用及渗透测试基础》实验报告

姓名： 许健 学号：2013018 班级：信安班

**实验名称：**

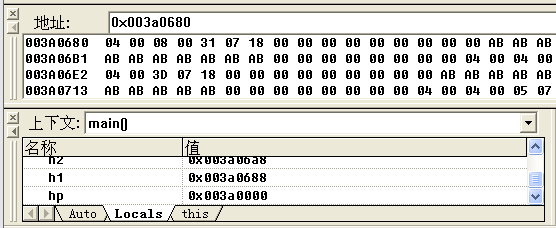
Dword Shoot攻击

**实验要求：**

以第四章示例4-4代码为准，在VC IDE中进行调试，观察堆管理结构，记录Unlink节点时的双向空闲链表的状态变化，了解堆溢出漏洞下的Dword Shoot攻击

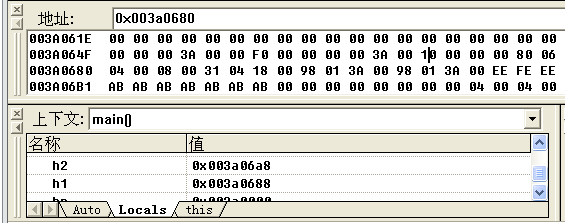
**实验过程：**

1. **代码流程解析**
2. 程序首先创建了一个大小为0x1000的堆区，然后连续申请了6个块身大小8字节的堆块，加上块首实际上是6个十六字节的堆块
3. 释放奇堆块是为了防止堆块的合并
4. 释放的堆块会被依次链入到freelist[2]所标识的空表上
5. 再次申请8字节的堆块则会从freelist[2]上取下第一个空闲态的堆块，即h1
6. 如果把h1的指针存储地址修改，就会发生DwordShoot攻击
7. **VC6调试观察堆内存变化**
8. 未释放前

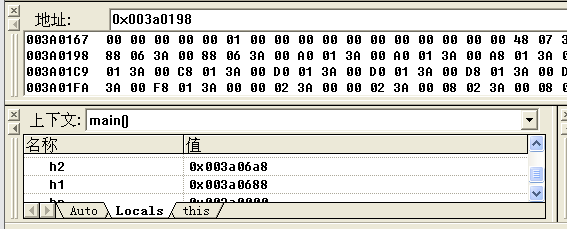


可以看到，hp为0x003a0688,h1为0x003a0688，h1块首地址应该为0x003a0680

1. 释放h1



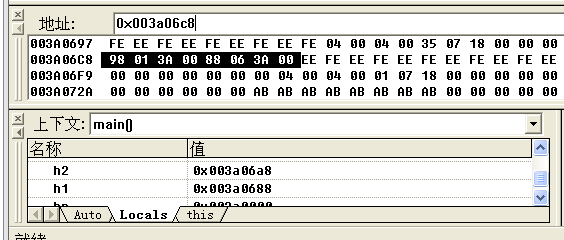
对比释放前后，可以看到不仅h1的块首信息变了，0x003a0688开始的块身位置的前8个字节（Flink和Blink）发生了变化，由0x000000都变为0x003a0198，实际上就是freelist[2]的地址。



跳转到0x003a0198处，可以看到freelist[2]的flink和blink存储的都是0x003a0688，说明h1是freelist[2]上唯一后继节点。

1. 释放h3、h5

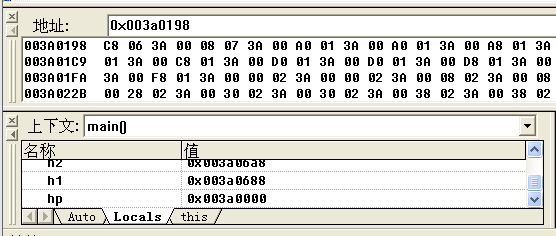
查看h3的地址，Flink指向freelist[2],blink指向h1



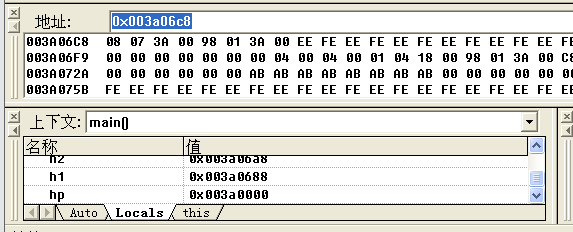
再释放掉h5，此时此时freelist[2]链表状态为：freelist[2]<=>h1<=>h3<=>h5。

1. **执行HeapAlloc(hp,HEAP\_ZERO\_MEMORY,8)语句**

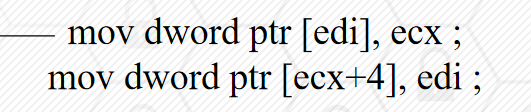
摘除h1后查看freelist[2]（地址为0x003a0198）

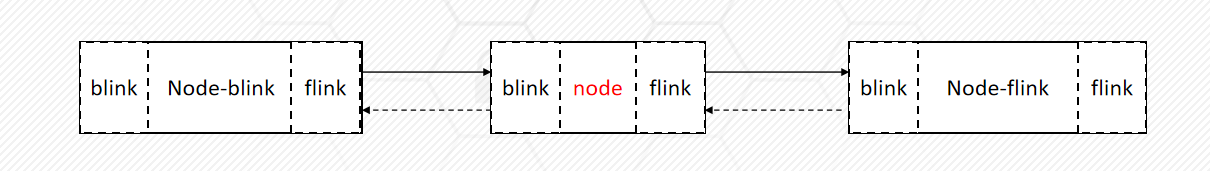


查看h3（地址为0x003a06c8）,后向指针指向freelist[2]



实际上，摘除h1发生了指针值的改写（附上汇编代码和图解）





**假设在执行该语句之前，h1的Flink和Blink被改写为特定地址和特定数值，那么就完成一次Dword Shoot攻击。**

**心得体会：**

通过使用vc6的调试器，查看堆区创建、释放过程中堆块前向指针、后向指针的变化，明白堆块工作的机理。

在这个过程中，如果堆块的前向指针、后向指针被恶意修改成特定地址和数值，就会完成DwordShoot攻击。

修改则利用堆溢出的机理，因此要养成检查数组是否越界、边界检查的编程习惯！