《漏洞利用及渗透测试基础》实验报告

姓名：申宗尚 学号：2213924 班级：信息安全

**实验名称：**

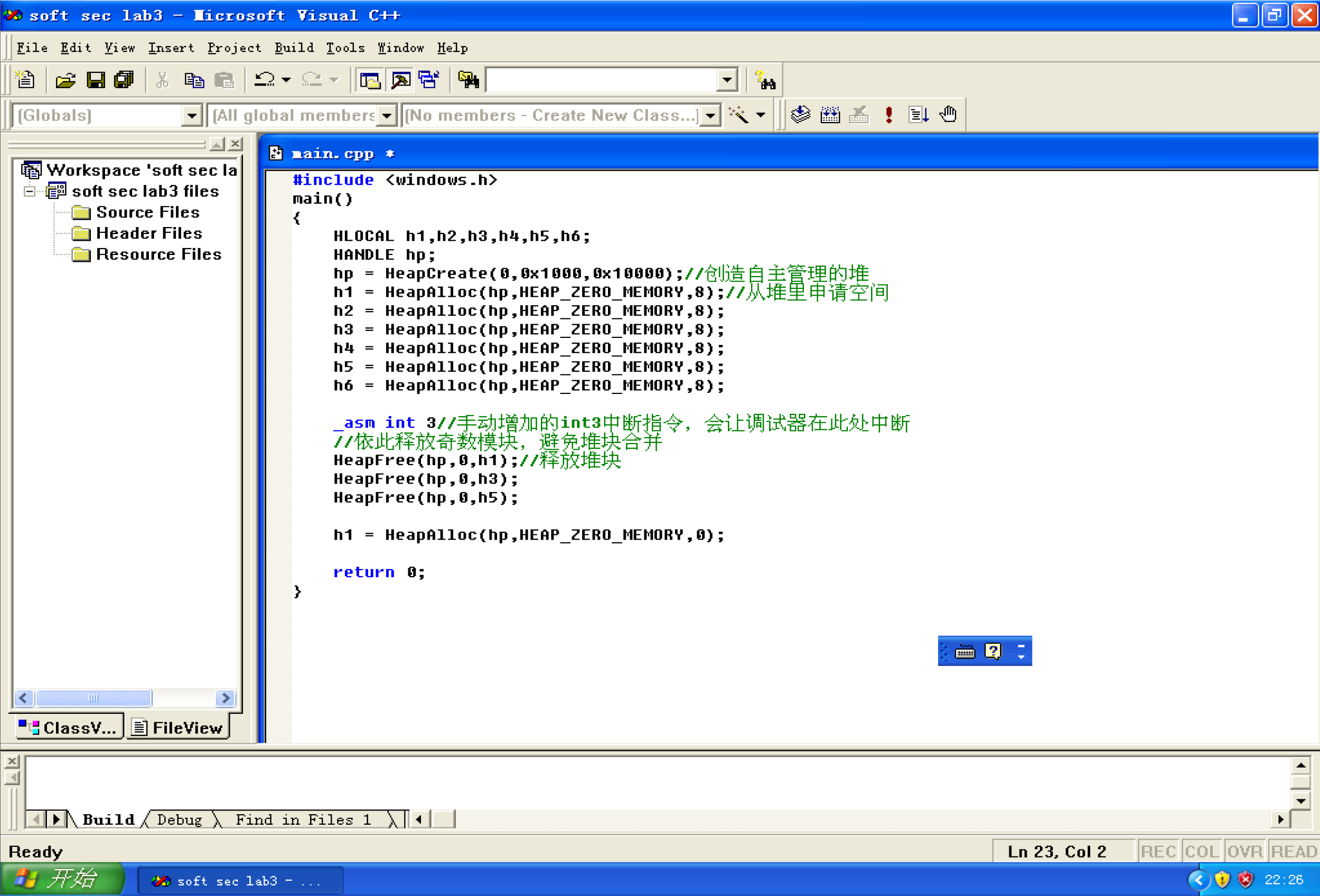
堆溢出Dword Shoot模拟实验

**实验要求：**

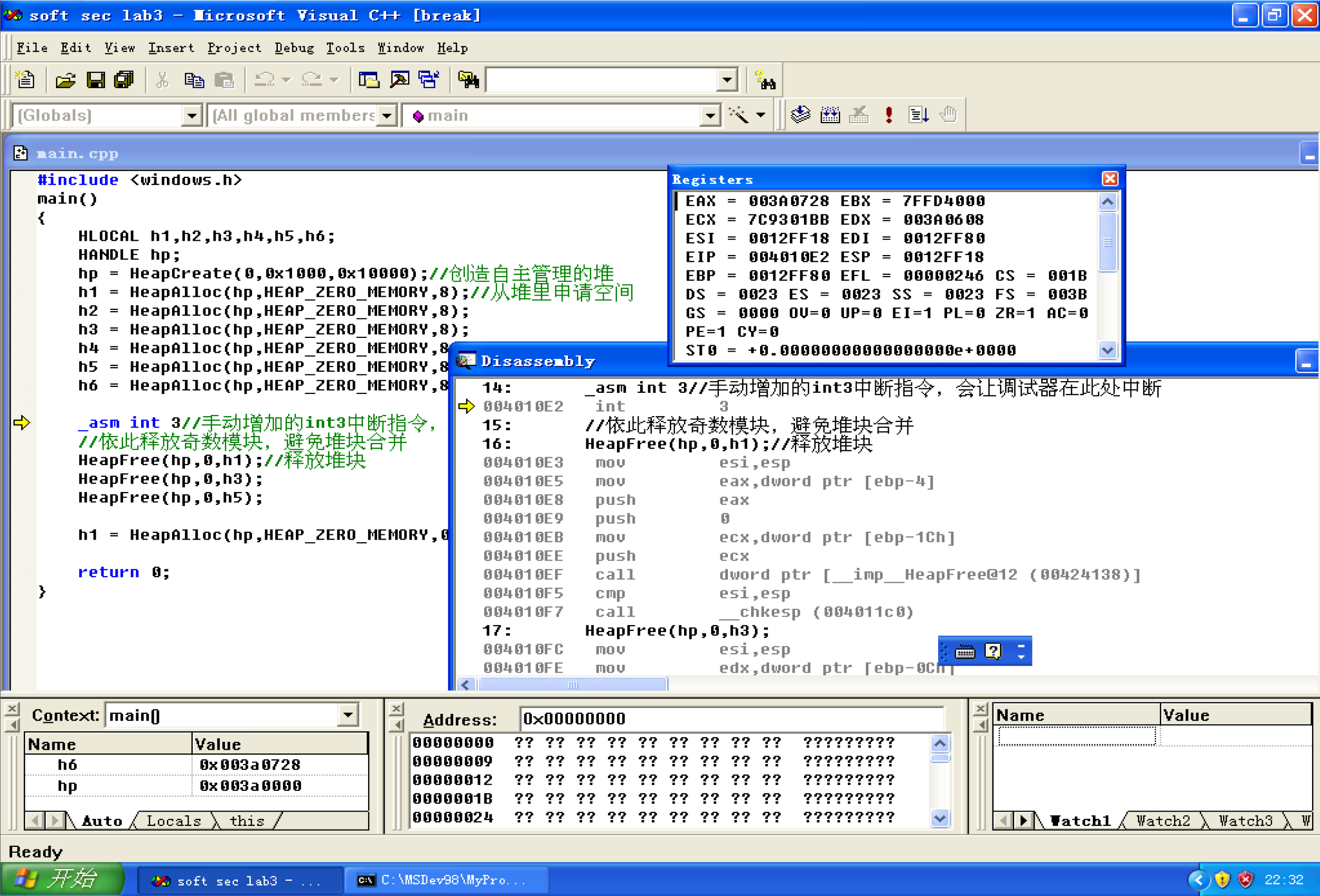
以第四章示例4-4代码为准，在VC IDE中进行调试，观察堆管理结构，记录Unlink节点时的双向空闲链表的状态变化，了解堆溢出漏洞下的Dword Shoot攻击。

**实验过程：**

1. 进入VC6，打开提供的4-4代码进行调试。



2. 按F10进入调试界面，项目停止在手动设置的断点(\_asm int 3)处。

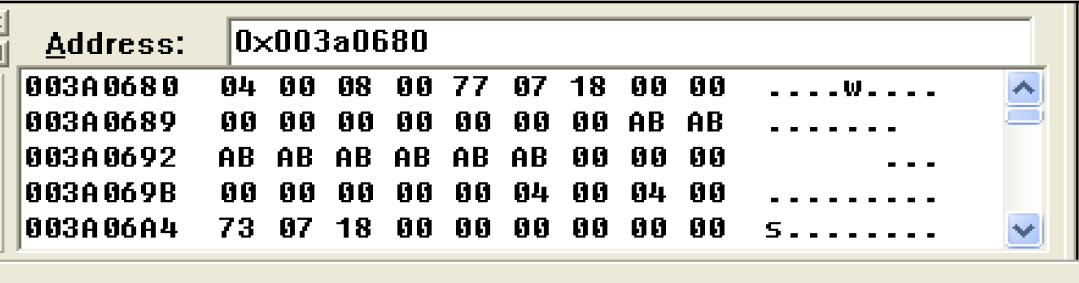


3.对于代码分析得知，此时已经创建了自主管理的堆hp，并在其中申请了6个堆块h1，h2，h3，h4，h5，h6，通过对locals表查询得知h1的值（块值）存储在0x003a0688处。

表格

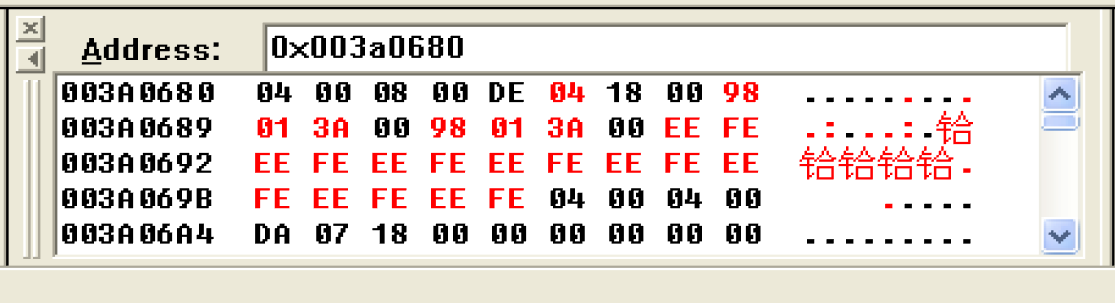
描述已自动生成

由于windows的堆管理系统，知申请过的堆块前含有8字节的块首信息，包括块是否被占用、块长度等，定位至0x003a0688-8=0x003a0680处，看到储存信息如下：

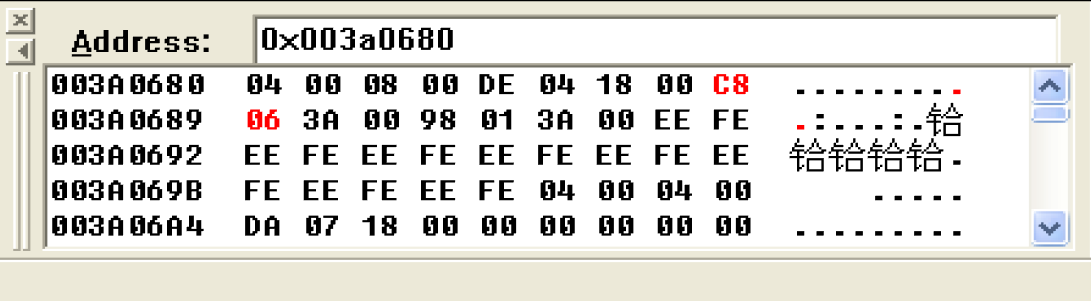


其中，前八位为块首信息，随后的的8字节信息均初始化为0。

4. 运行下一行代码HeapFree(hp,0,h1)，释放h1的堆，看到0x003a0680处信息如图变化，由于堆管理系统，该块长度为8字节，因此会进入FreeLink[2]的链表中，块首信息变为12字节，后面的值被初始化为EFEE，前面的两次003a0198代表其flink、blink，因为此时FreeLink[2]只有一个块，因此均指向FreeLink[2]的地址处。



5.随后运行下一行代码HeapFree(hp,0,h3)，此时只释放奇数堆，由于为了避免堆的合并，释放h3的堆，0x003a0680处的信息如图变化：



可以发现，代表blink的003a0198并未改变，因为h1仍是FreeLink[2]的首元素，而其flink变为003a06c8，指向h3的地址。

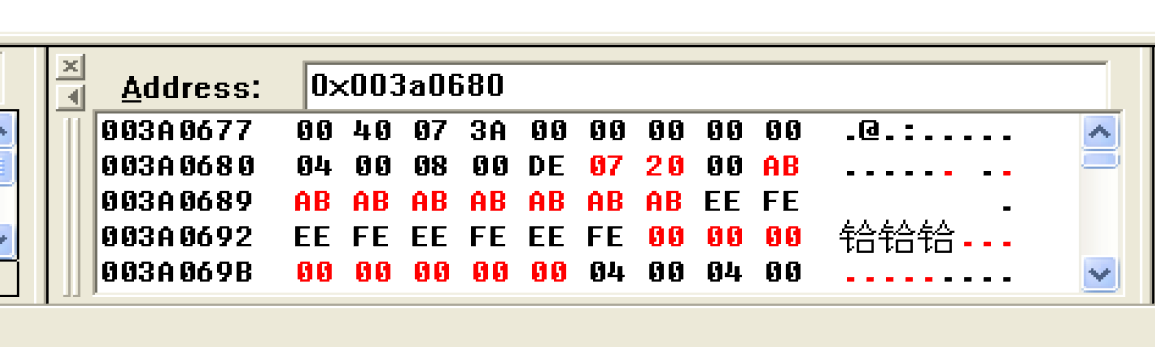
再查看003a06c8-12=003a06bc的位置，发现其blink为003a0688，为h1的地址，而flink为003a0198，为FreeLink[2]的地址。

文本

描述已自动生成

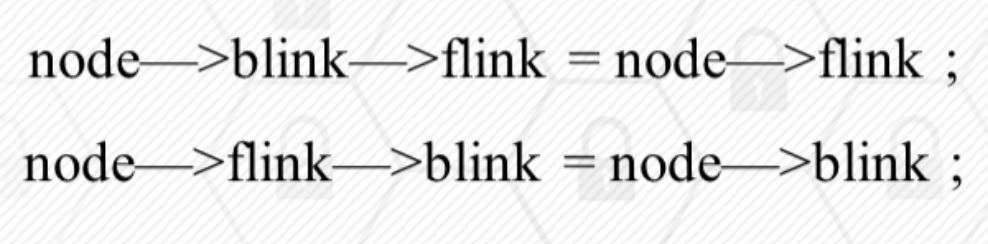
6. 类似地、执行释放h5，可以观察到h3的flink变为h5的地址，h5的blink为h3的地址，flink为FreeLink[2]的地址。

7. 最后进行h1的重新申请，运行h1=HeapAlloc(hp,HEAP\_ZERO\_MEMORY,8);观察到0x003a0680如图变化，故又回到了开始的状态，而在FreeLink中，h3的blink为FreeLink[2]的地址，flink不变，h5的blink不变，flink不变



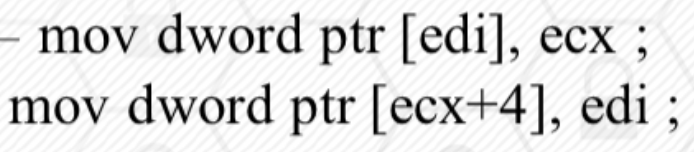
8. Dword Shoot漏洞分析：

由于在过程中，每次对于堆从Freelink的删除，都要进行前后指针的转换



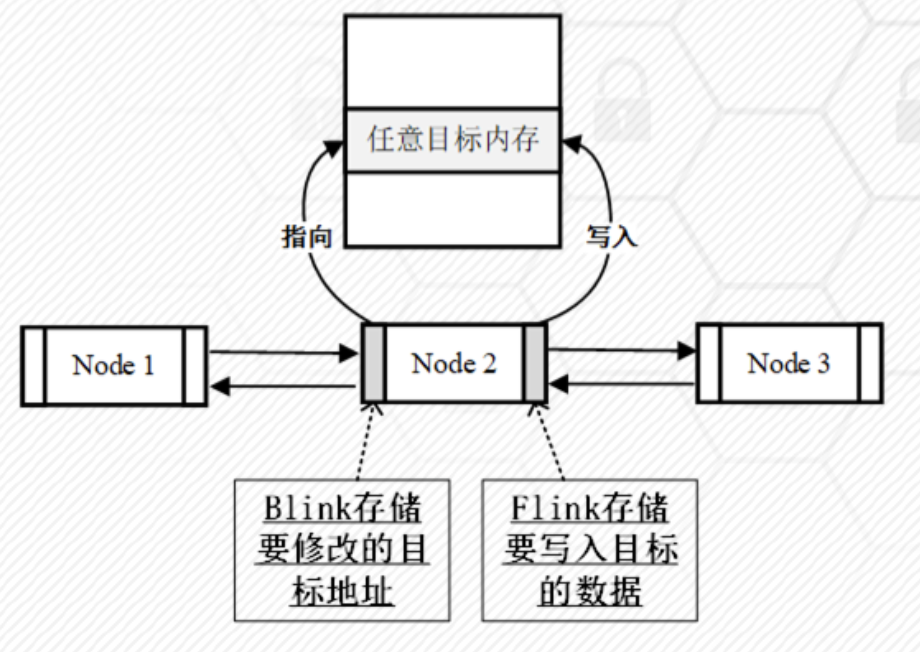
（将前指针的后指针转为当前的后指针，后指针的前指针转为当前的前指针）

在汇编语言中，即为如下指令：



其中，ecx为空闲可分配的堆区块的前向指针，edi为该堆块的后向指针

因此，如果对ecx和edi的值进行修改，以第一条汇编指令为例，就可以在[edi]处植入为ecx的值，完成一次Dword Shoot（对任意地址进行任意值的修改），从而产生安全问题。



**心得体会：**

通过实验，掌握了使用VC6对于windows.h中包含的对堆操作的代码、原理的学习，深刻了解了堆控制系统，和空闲双向链表FreeList原理的掌握，以及申请堆、释放堆的具体电脑实现和代码流程。

同时，对Dword Shoot漏洞的原理进行了学习，通过实践了解了其发生的原因和利用的方式，对堆溢出漏洞有了更深刻的认识。