《软件安全》实验报告

姓名：邢清画 学号：2211999 班级：1023

**实验名称：**

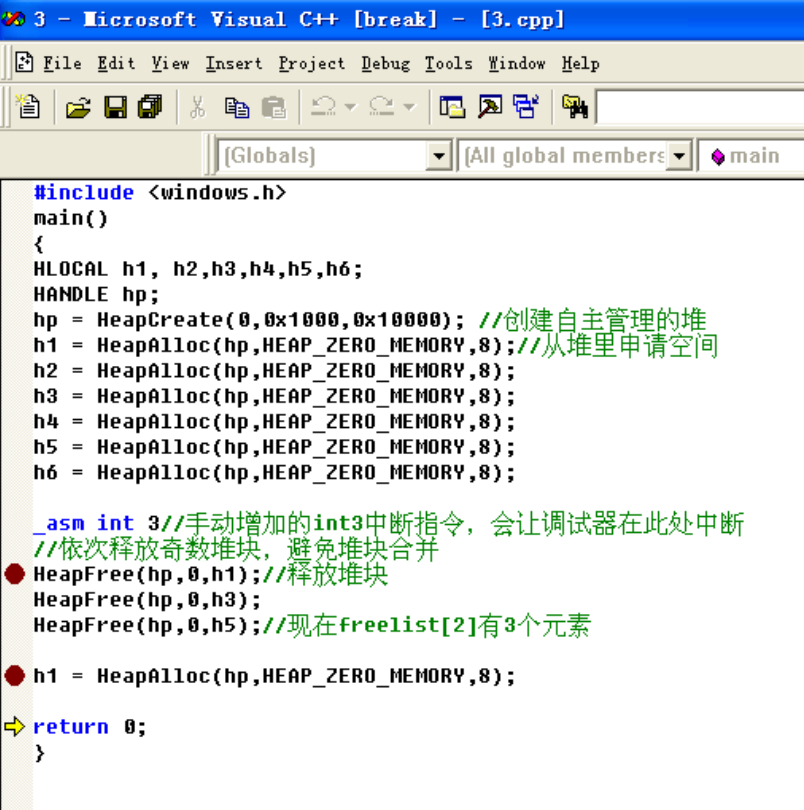
堆溢出Dword Shoot模拟实验

**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.代码分析**

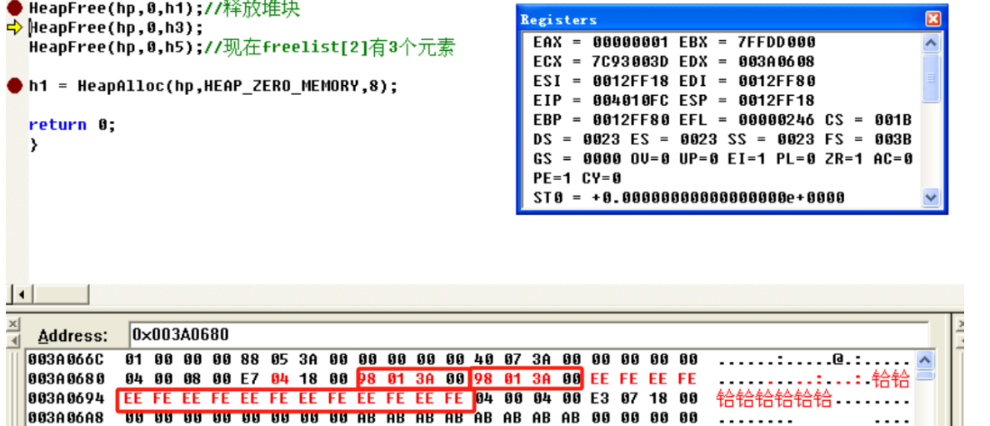


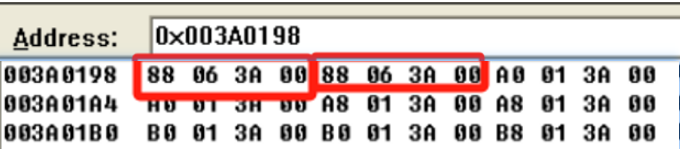
1. 运行实验代码，停止在int3断点处，创建大小为 0x1000 的堆区，此时已经完成了h1至h6的堆块初始化，对于每个堆块，申请8字节+块首8字节=16字节的堆块。
2. 释放三次奇数次申请后，形成三个16个字节的空闲堆块放入空表。
3. 16个字节依次放入freelist[2]所标识的空表。
4. 再次申请8字节的堆区内存，从freelist[2]所标识的空表中摘取第一个空闲堆块（h1）。
5. 手动修改h1块首中的指针，可观察到 DWORD SHOOT。

**2.DEBUG观察堆内存变化**

1.释放h1：

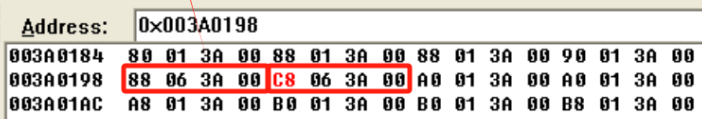
执行 HeapFree(hp,0,h1)语句前，hp为0x003A0000,h1为0x003A0688，加上块首为0x003A0680，执行后内存变化如下：



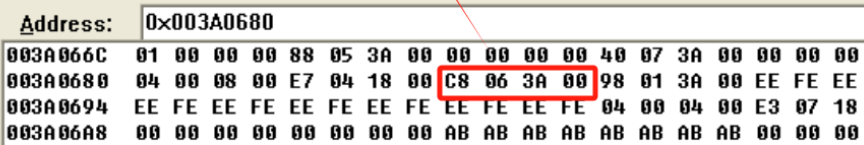
只有h1放进来，freelist[2]的地址(0x003A0198)，freelist[2]的前后向指针都指向h1(0x003A0688)，即释放的h1的块身的首地址，freelist[2]唯一后继节点是空闲的h1块（0x003A0688）。

2.释放h3:

执行HeapFree(hp,0,h3)前h3块身的首地址为0x003A06C8，执行完HeapFree(hp,0,h3)之后，freelist[2]的Blink指向h3块身首地址0x003A06C8。



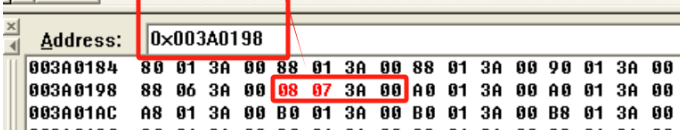
返回0x003A0680查看，前向指针Flink也改变到0x003A06C8 ，即h3变到了h1后面。



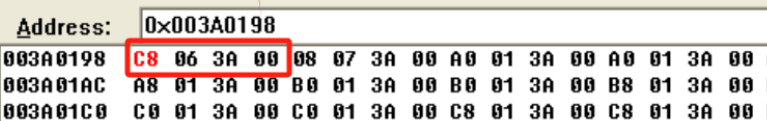
3.释放h5：

执行HeapFree(hp,0,h5)前， h5块身首地址为0x003A0708。

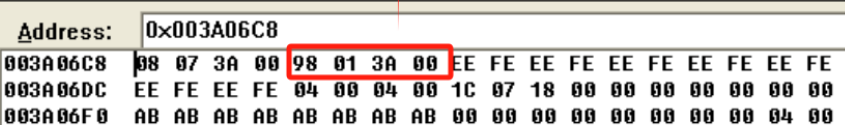
继续按F10释放，变成0x003A0708，即h5块身首地址，h5放到了freelist[2]的末尾。



继续F101，0x003A06C8是h3，在freelist的第一个空闲块。



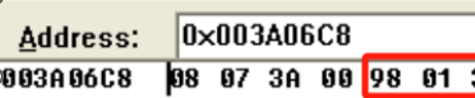
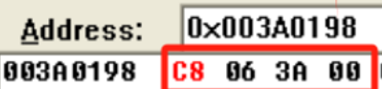
输入h3的地址0x003A06C8，Blink变成0x003A0198，即freelist的第二个。



执行完 HeapFree(hp,0,h3)和 HeapFree(hp,0,h5)后，hl,h2,h5 的前后指针状态依次为：

0x003A0688，0x003A06C8，0x003A0708。

4. 执行 HeapA11oc(hp,HEAP ZERO MEMORY,8)语句时，需要从 freelist[2]中摘下 16字节的堆块，摘下 h1观察内存区freelist[2]和 h3 的指针。freelist[2]的Flink被修改为了0x003A06C8即h3,h3的Blink 被修改为了0x003A0198，即freelist[2]。

5. Dword Shoot攻击：

通过一定办法将h1的Flink和Blink改写成特定地址和特定数值，即完成一次Dword Shoot攻击。

**3.心得体会：**

通过实验，掌握了堆的管理方式，以及空表的结构，存储方式，在进行内存分配或释放时，操作系统的堆管理器会更新这些管理结构，以反映当前堆的状态。

双向链表确保了内存的高效利用和回收。

了解了通过堆溢出漏洞来进行Dword Shoot 攻击的有效手段。