《漏洞利用及渗透测试基础》实验报告

姓名：齐明杰　　学号：2113997　　班级： 信安 2 班

实验名称：

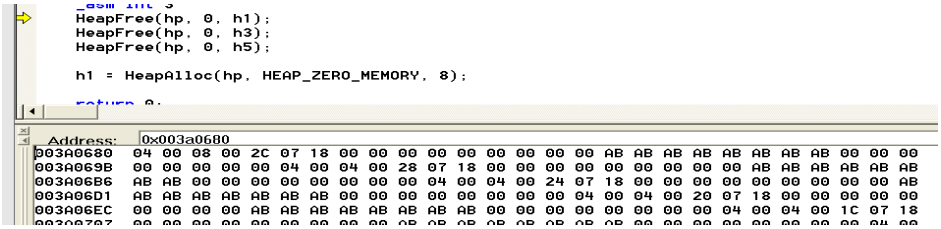
Dword Shoot 攻击实验

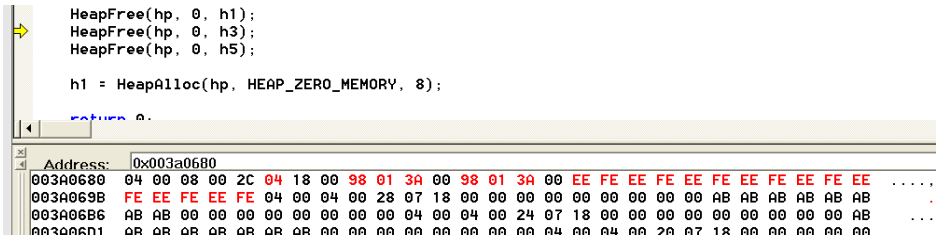
实验要求：

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：

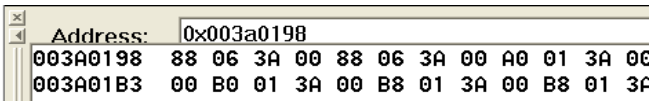
1. 运行实验代码，停止在 int3 断点处，此时已经完成了 h1 至 h6 的堆块初始化(对于每个堆块，申请 8 字节+块首 8 字节=16 字节)
2. 执行 HeapFree(hp,0,h1)语句前，hp 为 0x003a0000,h1 为 0x003a0688，据此可得 h1 堆块起始地址即为 0x003a0688，对应块首地址为 0x003a0680，执行前后内存变化如下图：





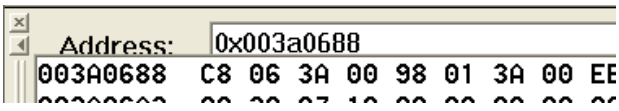
可得 h1 的 Flink 和 Blink 都指向 freelist[2]，即空表结构为 freelist[2]⇔h1。

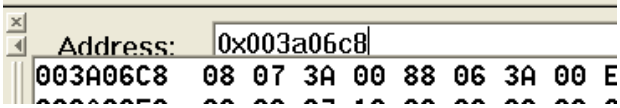
转到 freelist[2]的地址(0x003a0198)，如下图：

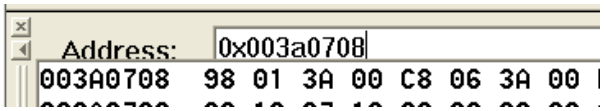


可见，freelist[2]的 Flink 和 Blink 均为 h1 的地址(0x003a0688)

3. 之后执行完 HeapFree(hp,0,h3)和 HeapFree(hp,0,h5)后，h1,h2,h5 的前后指针状态如下图：

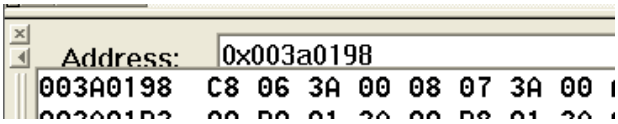


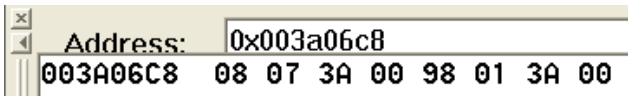




可见，此时的空表结构为 freelist[2]⇔h1⇔h3⇔h5。

4. 执行 HeapAlloc(hp,HEAP_ZERO_MEMORY,8)语句时，需要从 freelist[2]中摘下 16 字节的堆块，首先摘下 h1,此时理论上空表结构为 freelist[2]⇔h3⇔h5，下面观察一下内存区 freelist[2]和 h3 的指针，如下图：





可见，freelist[2]的 Flink 被修改为了 0x3A06C8(h3),h3 的 Blink 被修改为了 0x003A0198(freelist[2])，符合上文所述结构。

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。

此外，通过本实验，掌握了通过堆溢出漏洞来进行 Dword Shoot 攻击的有效手段