《软件安全》实验报告

姓名：张丛 学号：2113662 班级：信安一班

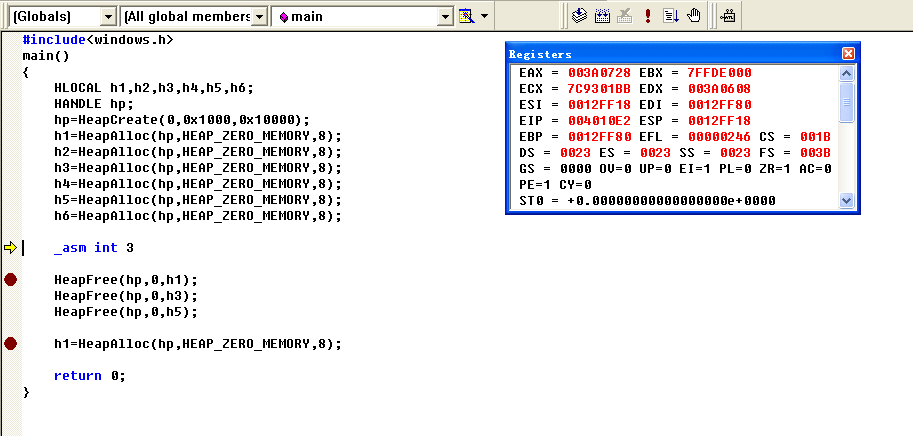
**实验名称：**

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. 设置断点，F5进入调试模式；

代码：

HLOCAL h1,h2,h3,h4,h5,h6;

HANDLE hp;

hp=HeapCreate(0,0x1000,0x10000); //创建自主管理的堆

h1=HeapAlloc(hp,HEAP\_ZERO\_MEMORY,8); //依次从堆里申请空间

h2=HeapAlloc(hp,HEAP\_ZERO\_MEMORY,8);

h3=HeapAlloc(hp,HEAP\_ZERO\_MEMORY,8);

h4=HeapAlloc(hp,HEAP\_ZERO\_MEMORY,8);

h5=HeapAlloc(hp,HEAP\_ZERO\_MEMORY,8);

h6=HeapAlloc(hp,HEAP\_ZERO\_MEMORY,8);

HeapFree(hp,0,h1); //三次释放堆块

HeapFree(hp,0,h3);

HeapFree(hp,0,h5);

h1=HeapAlloc(hp,HEAP\_ZERO\_MEMORY,8);//再次申请，从空表中摘取

值得注意的点：

堆块包括两个部分，即块首和块身。块首标识堆块信息，块身存储数据。

对于空闲态的堆块而言，块首还存储了两个4字节的指针：Flink和Blink。

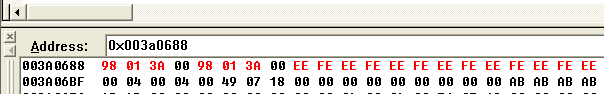
这里我们申请的是8字节，但实际会分配16字节的空间大小，包括8字节块首和8字节块身。

freelist即为空闲双向链表，也称空表，用于索引空闲态堆块。

因为大小是16字节，故三次释放后，三个16字节的空闲堆块放入空表时会依次放入freelist[2]，即h1、h3、h5。

1. 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

h1的块身地址为0x3a688



释放h1时，即h1进入freelist[2]，此时h1的Flink和Blink值均为0x003a1098，也正是freelist[2]的地址。

释放h3后：



h1的Flink值仍为0x003a1098,而Blink值变为0x003a06c8，此地址也正是h3的块身地址



而h3的Flink值为0x003a0688(即指向h1),Blink值为0x003a0198(即freelist[2]）

释放h5:



h3的Flink值为0x003a0688(即h1地址），BLink值为0x003a0708(即h5地址）



h5的Flink值为0x003a06c8(即h3地址）

此时，链表状态为：freelist[2]<==>h1<==>h3<==>h5

摘走h1后：







h3的Flink值为0x003a0198(即freelist[2]的地址，h1的Flink值），Blink的值为0x003a0708(即h5地址）

而freelist[2]的Flink指向了h5（0x003a0708）,Blink指向了h3(0x003a06c8)

可见，实现了在freelist[2]中摘走h1

但同时，也正是此次实验的重点：如果在执行摘走之前，修改h1的Flink值为特定的地址和特定数值，就完成了一次Dword Shoot攻击，这就给了攻击者向任意地址写入任意数据的机会。

**心得体会：**

通过实验，模拟了Dword Shoot攻击，进一步学习到堆溢出漏洞是如何发生的；

学习到堆块的申请与释放，也进一步了解了空闲堆块的在freelist的进出实现；

复习了堆块的结构，以及堆表的知识点。