

《软件安全》实验报告

姓名：蒋薇 学号：2110957 班级：计科1班

实验名称：

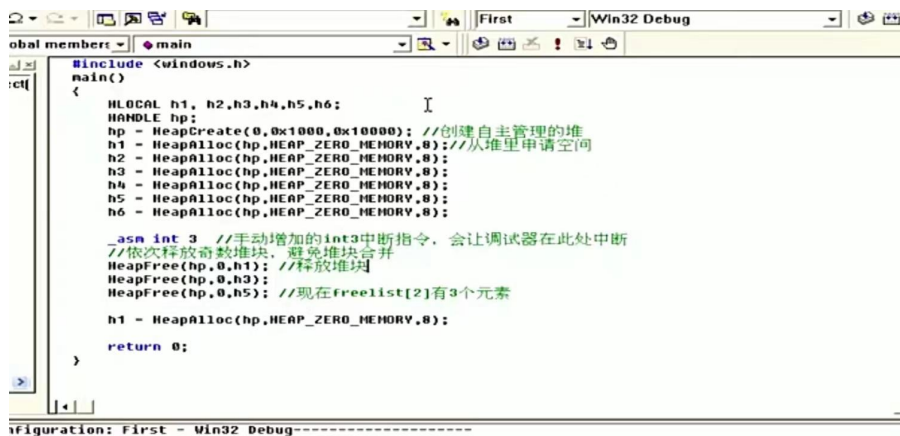
堆溢出 Dword Shoot 模拟实验

实验要求：

根据第四章示例 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. 在 VC IDE 进行调试，观察堆管理结构，



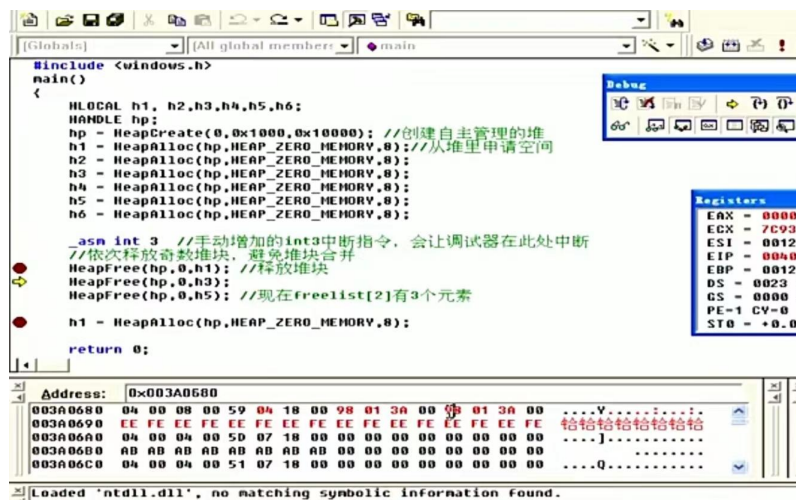
```
#include <windows.h>
main()
{
    LOCAL h1, h2, h3, h4, h5, h6;
    HANDLE hp;
    hp = HeapCreate(0, 0x1000, 0x1000); //创建自主管理的堆
    h1 = HeapAlloc(hp, HEAP_ZERO_MEMORY, 8); //从堆里申请空间
    h2 = HeapAlloc(hp, HEAP_ZERO_MEMORY, 8);
    h3 = HeapAlloc(hp, HEAP_ZERO_MEMORY, 8);
    h4 = HeapAlloc(hp, HEAP_ZERO_MEMORY, 8);
    h5 = HeapAlloc(hp, HEAP_ZERO_MEMORY, 8);
    h6 = HeapAlloc(hp, HEAP_ZERO_MEMORY, 8);

    _asm int 3 //手动增加的int3中断指令，会让调试器在此处中断
    //依次释放奇数堆块，避免堆块合并
    HeapFree(hp, 0, h1); //释放堆块
    HeapFree(hp, 0, h3);
    HeapFree(hp, 0, h5); //现在freelist[2]有3个元素

    h1 = HeapAlloc(hp, HEAP_ZERO_MEMORY, 8);

    return 0;
}
```

创建自主管理的堆，申请 6 个堆块，后依次释放，释放三个奇数块后，避免连续堆块合并，再次申请时从空闲堆块摘取一个，发生 Unlink 操作



```
#include <windows.h>
main()
{
    LOCAL h1, h2, h3, h4, h5, h6;
    HANDLE hp;
    hp = HeapCreate(0, 0x1000, 0x1000); //创建自主管理的堆
    h1 = HeapAlloc(hp, HEAP_ZERO_MEMORY, 8); //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);

    _asm int 3 //手动增加的int3中断指令，会让调试器在此处中断
    //依次释放奇数堆块，避免堆块合并
    HeapFree(hp, 0, h1); //释放堆块
    HeapFree(hp, 0, h3);
    HeapFree(hp, 0, h5); //现在freelist[2]有3个元素

    h1 = HeapAlloc(hp, HEAP_ZERO_MEMORY, 8);

    return 0;
}
```

Registers:

- EAX = 00000000
- ECX = 7C930000
- ESI = 0012F000
- EIP = 00401000
- EBP = 0012F000
- DS = 0023
- CS = 0000
- PE = 1
- CV = 0
- ST0 = +0.00

Memory Dump:

Address: 0x003A0680

003A0680 04 00 00 00 59 04 18 00 98 01 3A 00 00 01 3A 00 ...V.....

003A0690 EE FE EE FE EE FE EE FE EE FE EE FE EE FE EE FE ...Q.....

003A06A0 04 00 04 00 5D 07 18 00 00 00 00 00 00 00 00 ...Q.....

003A06B0 AD AD AD AD AD AD AD AD AD AD AD AD AD AD AD ...Q.....

003A06C0 04 00 04 00 51 07 18 00 00 00 00 00 00 00 00 ...Q.....

2. 记录 Unlink 节点时的双向空闲链表的状态变化

Freelist[] 有三个元素，再去申请八字节的空间，从 freelist[2] 中摘取 16 字节的块，为之前释放的 h1 块，在此 Unlink 过程中，发生 DWord Shoot 攻击

Asm int 3, 已手动增加 int3 中断指令，让调试器在此处中断，进入 debug 模式；

F10 调试后，h1 释放，放入空闲双向链表中，块状态改写，块首状态，link 写为 003A98，

Freelist[2]后项指针法发生变化, 变为 h3 的值, 06C8, h3 放到链表末尾
进入 0198 地址处, 前后项指针 003A0688, freelist[2]只有 h1 块
F10, free list[2]后向指针改变, 为 h3, h1 后为 h3,
F10, h5 放到 freelist[2]末尾
找到 h1, 如果可以篡改, 发生 DWord Shoot 攻击
Alloc h1 演示摘走块后 freelist[2]状态, 前向指针变化, 从指向 h1 变为指 h3

(此处根据实际操作过程, 留下具体操作步骤、附加一些自己的理解, 即可)

心得体会:

通过实验, 知道了堆溢出漏洞用来发动 Dword Shoot 攻击, 堆管理工具, Unlink 过程中发生的系列工作;

此外, 通过本实验, 可用 ollydebug 修改, 发动 DWord Shoot 攻击