《软件安全》实验报告

姓名：梁婧涵 学号：2112155 班级：1120

实验名称：堆溢出Dword Short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。

#include <windows.h>

main()

{

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

HANDLE hp;

hp=HeapCreate(0,0x1000,0x10000);

h1=HeapAlloc(hp,HEAP\_ZERO\_MEMORY,8);

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

HeapFree(hp,0,h1);

HeapFree(hp,0,h3);

HeapFree(hp,0,h5);

h1=HeapAlloc(hp,HEAP\_ZERO\_MEMORY,8);

return 0;

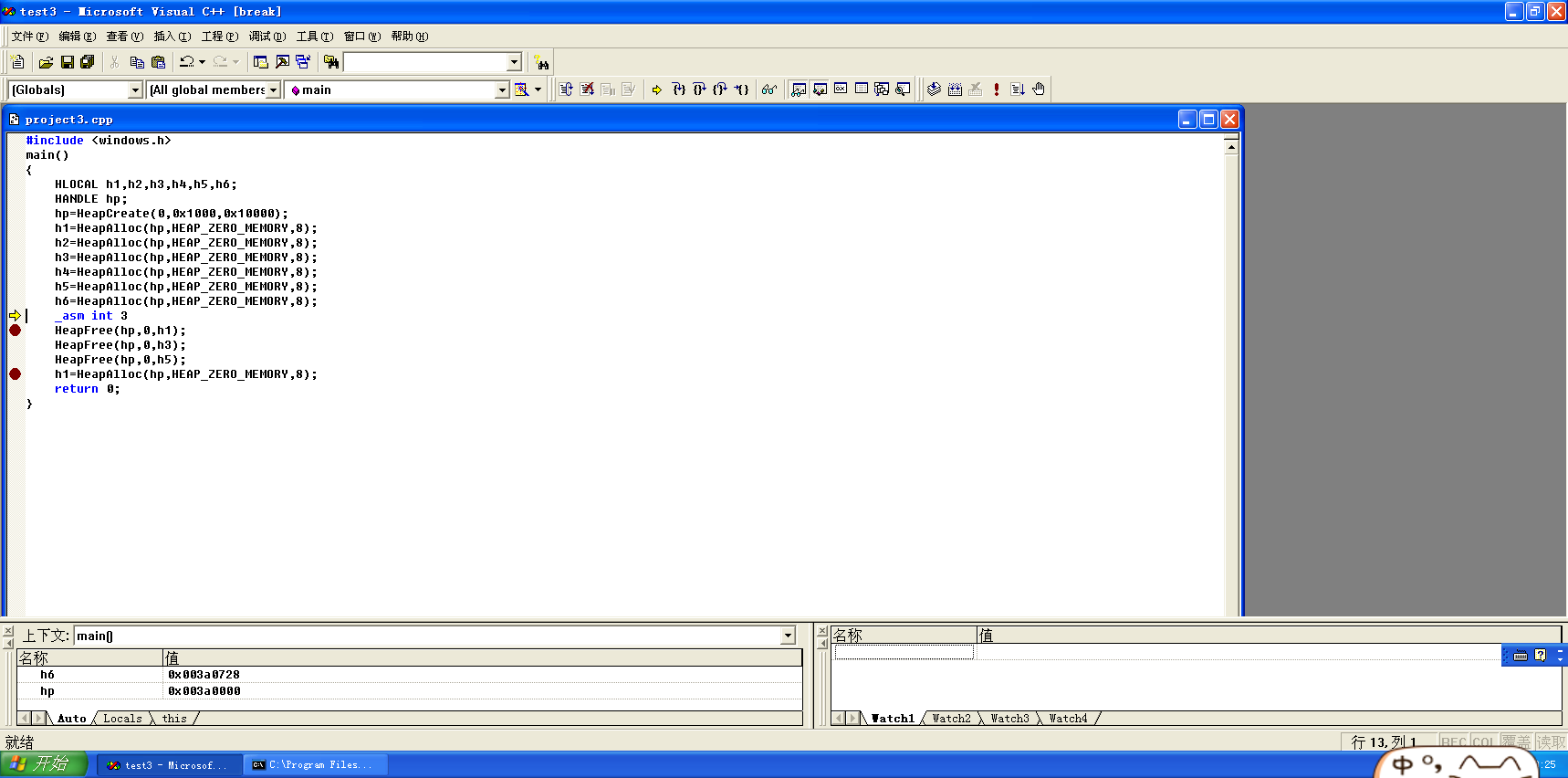
}

实验过程：

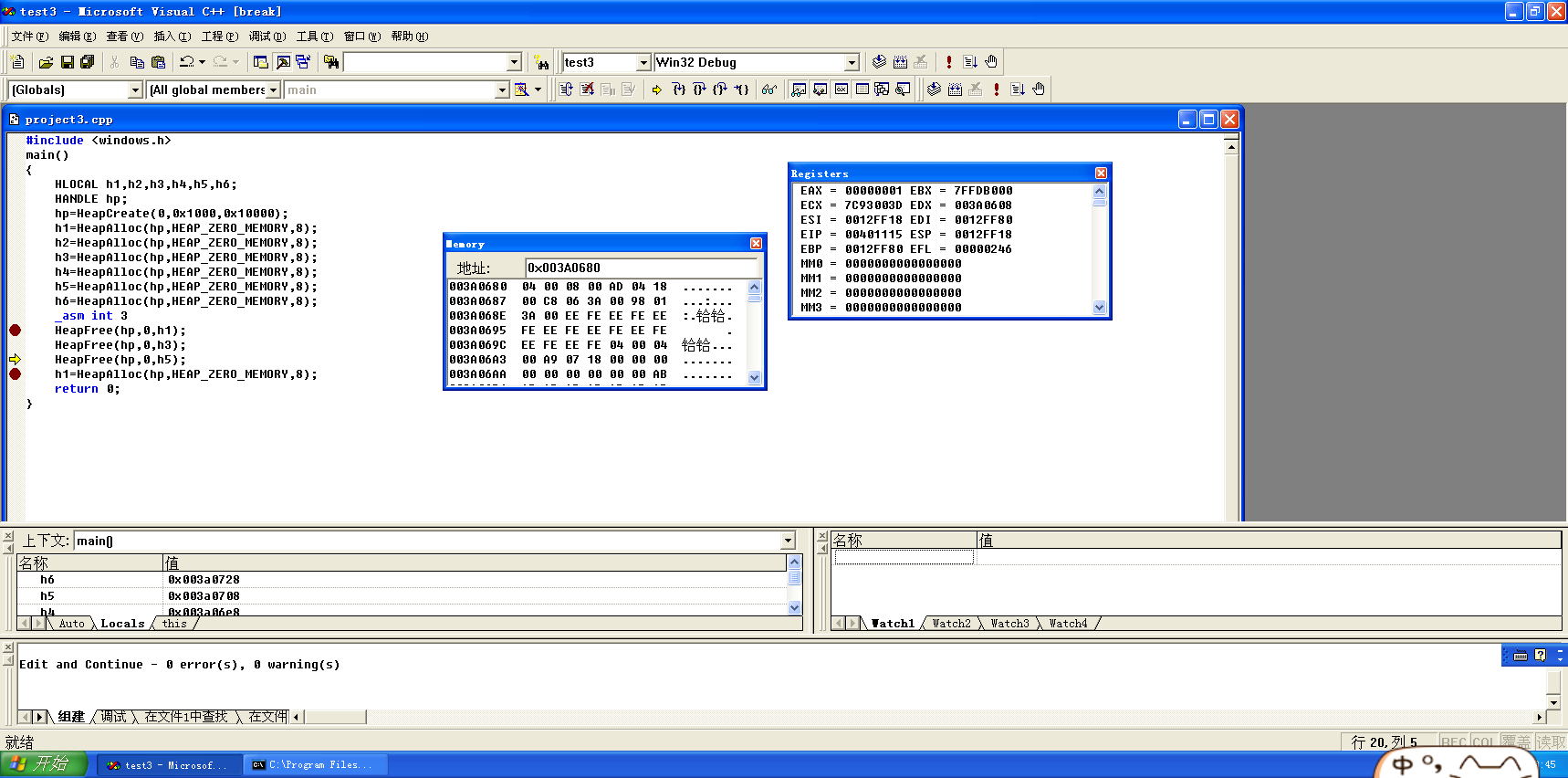
1. 创建一个自主管理的堆，从堆中申请6个堆块，根据之前动调空表的经验直接找到申请的堆块的位置，六块申请了8个字节的空间，算上堆头8个字节，就相当于六块16个字节的[内存](https://so.csdn.net/so/search?q=%E5%86%85%E5%AD%98&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/Misaka10046/article/details/_blank)，直接从尾块上面“切”下来。
2. 释放奇数位的堆块，防止堆块合并发现h1，h3，h5都链入了Freelist[2]依次释放3个，避免连续块发生合并，放在freelist【2】中申请字节。

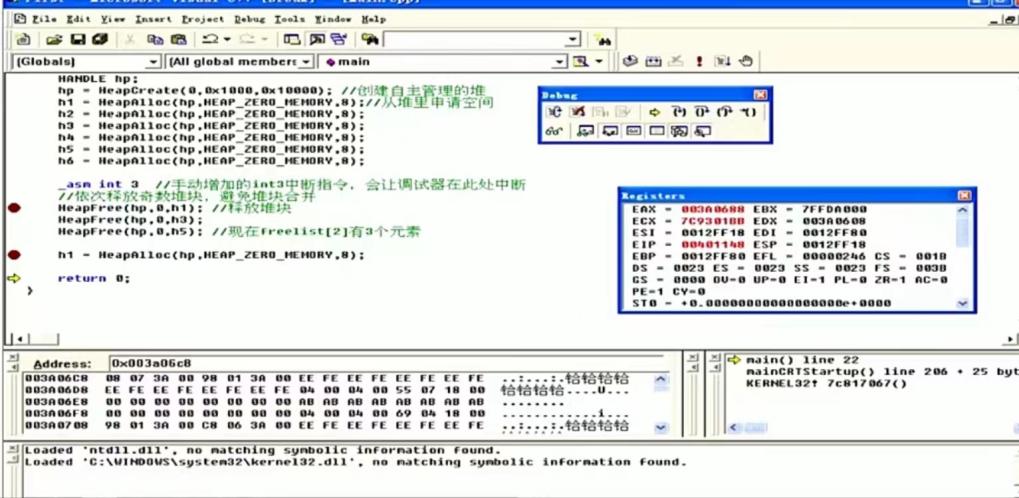
此时堆块的占用状况：

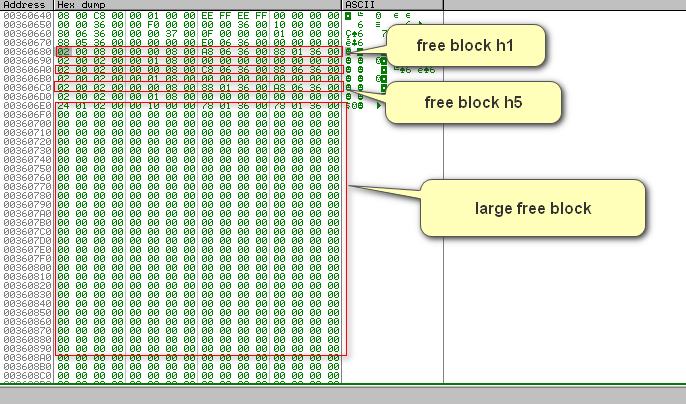
|  |  |  |  |
| --- | --- | --- | --- |
| NAME | Flag | 向前指针 | 向后指针 |
| h1 | 1占用态 | 0x003a06a8(h3) | 0x003a0188(Freelist[2]) |
| h2 | 0空闲态 | NULL | NULL |
| h3 | 1占用态 | 0x003a06c8(h5) | 0x003a0688(h1) |
| h4 | 0空闲态 | NULL | NULL |
| h5 | 1占用态 | 0x003a0188(Freelist[2]) | 0x003a06a8(h3) |
| h6 | 0空闲态 | NULL | NULL |

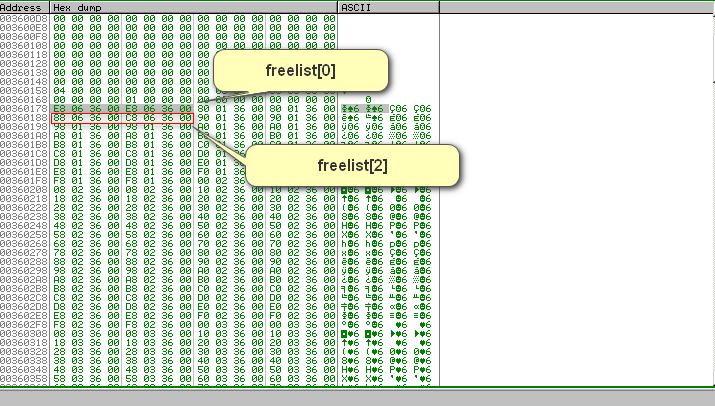
3.在HeapFree(hp,0,h1);和h1=HeapAlloc(hp,HEAP\_ZERO\_MEMORY,8);行打断点，F5进入debug模式

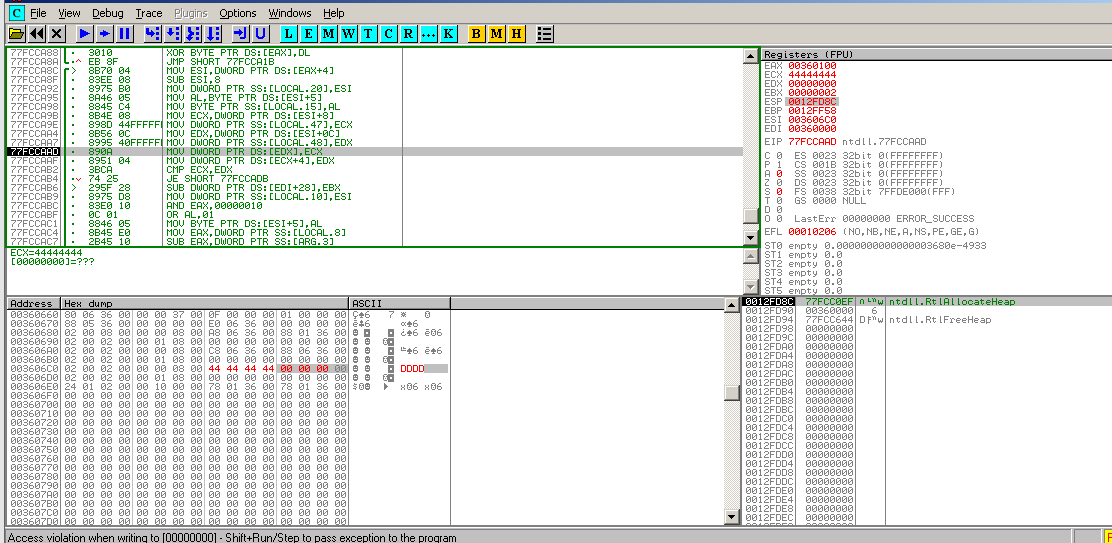
断在004010E2 int 3处，F10进入堆HeapFree(hp,0,h1);hp为自主管理堆的初始地址。F10后当前块的状态被改写。Address改成0x003A198，前后项指针都为003A0688



转到0x003A0198释放，198位置变为0708。h1为0680如果在此处篡改（01），就会发生一次Dword Short攻击。演示该条语句被去除以后的状态，flink改变h3的块变成第一个空闲块06C8变成0198，改动成功即发生Dword Short攻击。







心得体会：

通过实验，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，掌握了堆溢出漏洞下的Dword Shoot攻击。(node->blink->flink=node->flink)

（1）内存变量：修改能够影响程序执行的重要标志变量，往往可以改变程序流程。例如，更改身份验证函数的返回值就可以直接通过认证机制。栈溢出时溢出的数据必须连续，而 DWORD SHOOT 可以更改内存中任意地址的数据。

（2）代码逻辑：修改代码段重要函数的关键逻辑有时可以达到一定攻击效果，例如，程序分支处的判断逻辑，或者把身份验证函数的调用指令覆盖为 0x90(nop)。

（3）函数返回地址：栈溢出通过修改函数返回地址能够劫持进程，堆溢出也一样可以利用DWORD SHOOT 更改函数返回地址。但由于栈帧移位的原因，函数返回地址往往是不固定的，甚至在同一操作系统和补丁版本下连续运行两次栈状态都会有不同，故 DWORD SHOOT 在这种情况下有一定局限性，因为移动的靶子不好瞄准。

（4）攻击异常处理机制：当程序产生异常时，Windows 会转入异常处理机制。堆溢出很容易引起异常，因此异常处理机制所使用的重要数据结构往往会成为 DWORD SHOOT 的上等目标，这包括 S.E.H、F.V.E.H、进程环境块中的 U.E.F 、线程环境块中存放的第一个S.E.H 指针。

（5）函数指针：系统有时会使用一些函数指针，比如调用动态链接库中的函数、C++中的虚函数调用等。改写这些函数指针后，在函数调用发生后往往可以成功地劫持进程。但可惜的是，不是每一个漏洞都可以使用这项技术，这取决于软件的开发方式。