

# 《软件安全》实验报告

姓名：辛浩然      学号：2112514      班级：信息安全、法学

实验名称：

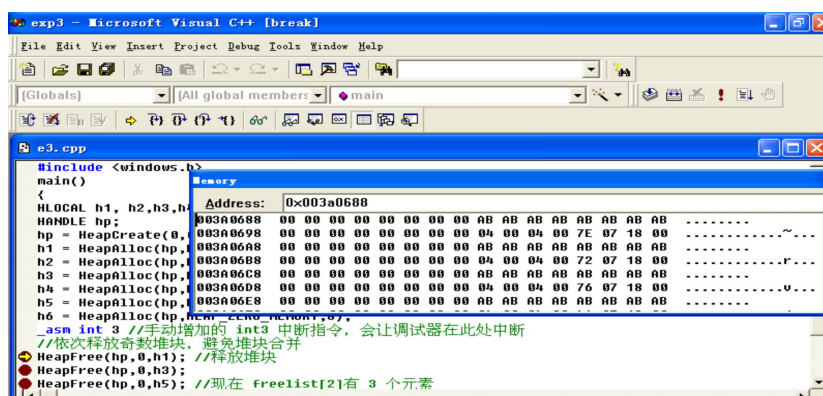
堆溢出 **DWORD SHOOT** 模拟实验

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.使用 VC6 自身的调试器调试程序。

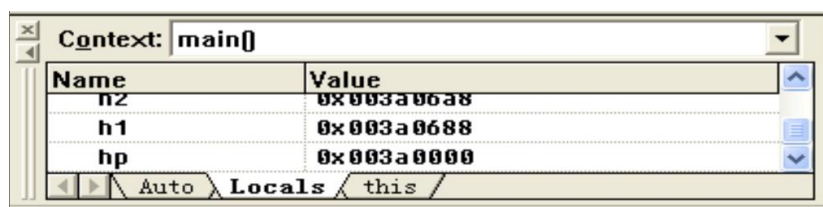


2.堆管理中的内存变化：

(1)创建堆

程序首先创建了一个大小为 0x1000 的堆区，并从其中连续申请了 6 个块身大小为 8 字节的堆块，加上块首实际上是 6 个 16 字节的堆块。

(2)执行 **HeapFree(hp,0,h1)**语句，释放 h1 堆块



可以知道，h1 为 0x003a0688.根据堆的结构，h1 堆块块身的起始位置为 0x003a0688，块首起始位置为 0x003a0600.

执行前，0x003a0688 开始的块身位置的前 8 个字节(Flink 和 Blink)为

0x00000000.

Memory															
Address:		0x003a0680													
003A0680	04	00	08	00	FC	07	18	00	00	00	00	00	00	00	00
003A0690	AB	AB	AB	AB	AB	AB	AB	00	00	00	00	00	00	00	00

执行后, 0x003a0688 开始的块身位置的前 8 个字节(Flink 和 Blink)变为具体的有效地址。

Memory															
Address:		0x003a0680													
003A0680	04	00	08	00	FC	04	18	00	98	01	3A	00	98	01	3A
003A0690	EE	FE	EE	FE	EE	FE	EE	FE	EE	FE	EE	FE	EE	FE	EE

16 字节的 h1 堆块被释放后, 被链入 freelist[2]空表中, 此时 **Flink** 和 **Blink** 的值均为 0x003a0198, 即 freelist[2]的地址。

转到 freelist[2]的地址处:

Memory															
Address:		0x003a0198													
003A0188	88	01	3A	00	88	01	3A	00	90	01	3A	00	90	01	3A
003A0198	88	06	3A	00	88	06	3A	00	A0	01	3A	00	A0	01	3A
003A01A8	A8	01	3A	00	A8	01	3A	00	B0	01	3A	00	B0	01	3A

可以发现, freelist[2]的 Flink 和 Blink 都是 0x003a0688, 也就是刚刚空闲的 h1 块的地址。

**freelist[2]链表状态为: freelist[2]↔h1**

对比其他地址(freelist[3]、freelist[4]), 可以发现它们的 Flink 和 Blink 都指向自身, 说明都是空表。

(3)执行 **HeapFree(hp,0,h3)**语句, 释放 h3 堆块

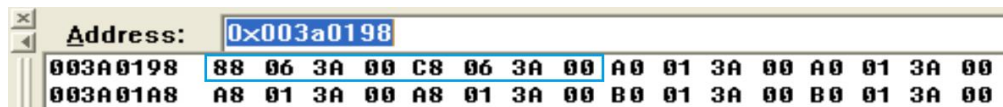
h3 的块身地址为 0x003a06c8.转到该地址处:

Address:		0x003a06c8													
003A06C8	98	01	3A	00	88	06	3A	00	EE	FE	EE	FE	EE	FE	EE
003A06D8	EE	FE	EE	FE	EE	FE	EE	FE	04	00	04	00	C7	07	18

执行后, 0x003a06c8 开始的块身位置的前 8 个字节(Flink 和 Blink)变为具体的有效地址。

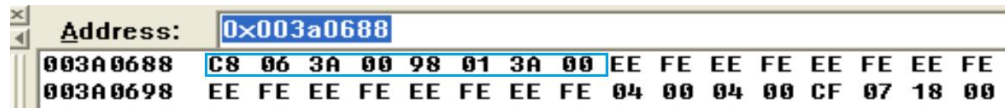
16 字节的 h3 堆块被释放后, 被链入 freelist[2]空表中, 此时 **Flink** 值为 0x003a0198, 是 freelist[2]的地址; **Blink** 值为 0x003a0688, 是空闲的 h1 块的地址。

转到 freelist[2]的地址处:



可以发现, freelist[2]的 Flink 为 0x003a0688, 是空闲的 h1 块的地址; Blink 为 0x003a06c8, 也就是刚刚空闲的 h3 块的地址。

转到 h1 地址处:



h1 的 Flink 为 0x003a06c8, 是刚刚空闲的 h3 块的地址; Blink 为 0x003a0198, 是 freelist[2]的地址。

freelist[2]链表状态为: freelist[2] ⇌ h1 ⇌ h3 (左为 blink 指向, 右为 flink 指向)

(4) 执行 HeapFree(hp, 0, h5) 语句, 释放 h5 堆块

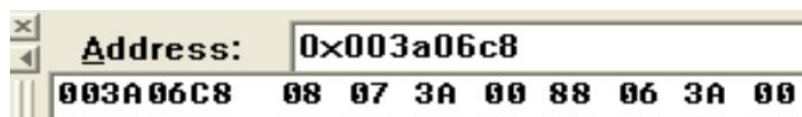
h5 的块身地址为 0x003a0708. 转到该地址处:



执行后, 0x003a0708 开始的块身位置的前 8 个字节(Flink 和 Blink)变为具体的有效地址。

16 字节的 h5 堆块被释放后, 被链入 freelist[2] 空表中, 此时 Flink 值为 0x003a0198, 是 freelist[2]的地址; Blink 值为 0x003a06c8, 是空闲的 h3 块的地址。

转到 h3 地址处:



h3 的 Flink 为 0x003a0708, 是刚刚空闲的 h5 块的地址; Blink 为 0x003a0688, 是 h1 堆块的地址。

转到 h1 地址处:



h1 的 Flink 为 0x003a06c8, 是 h3 块的地址; Blink 为 0x003a0198, 是 freelist[2] 的地址。

转到 freelist[2] 地址处:



freelist[2]的 Flink 为 0x003a0688，是 h1 块的地址；Blink 为 0x003a0708，也就是刚刚空闲的 h5 块的地址。

freelist[2]链表状态为：freelist[2]↔h1↔h3↔h5

(5)执行 HeapAlloc(hp,HEAP\_ZERO\_MEMORY,8)语句时，摘走 h1 堆块

摘走 h1 之后，内存变化：

freelist[2](地址为 0x003a0198)的前 4 个字节变为 0x003a06c8，即 h3 的地址。实际发生了将 h1 后向指针(值为 0x003a0198)地址处的值写为 h1 前向指针的值。



h3(地址为 0x003a06c8)的 Blink 变为 h1->Blink，即 0x003a0198，也就是 freelist[2]的地址。实际发生了将 h1 前向指针(值为 0x003a06c8)地址处的值写为 h1 后向指针的值。



(6)Dword Shoot 攻击

假设在执行(5)步摘取堆区的语句之前，h1 的 Flink 和 Blink 被改写为特定地址和特定数值，那么就完成一次 DwordShoot 攻击。

心得体会：

1.通过实验，认识了堆管理结构。通过调试程序，具体观察、了解了堆管理过程中的堆内存变化。

2.通过实验，了解了堆溢出漏洞下的 Dword Shoot 攻击。