HeapSpray：堆与栈的协同攻击

在使用HeapSpray的时候，一般会将eip指向堆区的0x0c0c0c0c位置，然后用jacaScript申请大量堆内存，并用包含着0x90和shellcode的“内存片”覆盖这些内存。

通常，JavaScript会从内存低址向高址分配内存，因此申请的内存超过200MB（200MB=200*1024*1024=0x0C800000>0x0C0C0C0C）后，0x0C0C0C0C将被含有shellcode的内存片覆盖。只要内存片中的0x90能够命中0x0C0C0C0C的位置，shellcode就能最终得到执行。这个过程如图所示

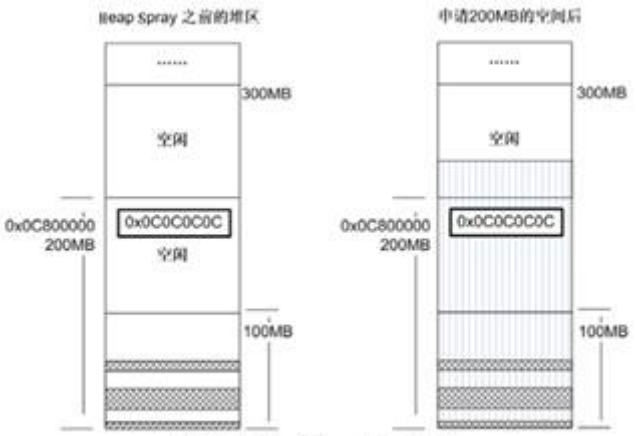


图 6.4.1 Heap Spray 技术示意图

对于这段JavaScript解释如下

（1） 每个内存片大小为1MB

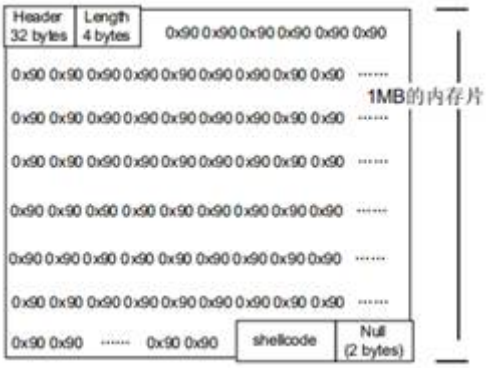
（2） 首先产生一个大小为1MB且全部被0x90填满的内存块

（3） 由于Java会为申请到的内存填上一些额外信息，为了保证内存片恰好是1MB，我们将这些额外信息所占的空间减去。具体说来，这些信息如表所示

在考虑了上述因素及shellcode的长度后，nop = nop.substring(0,0x100000/2 - 32/2 - 4/2 - shellcode.length - 2/2)将一个内存片恰好凑成1MB大小

	SIZE	说　明
malloc header	32 bytes	堆块信息
string length	4 bytes	表示字符串长度
terminator 2	bytes	字符串结束符，两个字节的NULL

（4） 如图，最终我们将使用200个这种形式的内存片来覆盖堆内存，只要其中任意一片的nop区能够覆盖0x0C0C0C0C，攻击就可以成功



为什么采用1MB大小作为内存片的单位呢？在HeapSpray时，内存片相对于shellcode和额外的内存信息来说应该“足够大”，这样nop区域命中0x0C0C0C0C的几率将相对增加；如果内存片较小，shellcode或额外的内存信息将有可能覆盖0x0C0C0C0C，导致溢出失败。1MB的内存相对于200字节左右的shellcode，可以让exploit拥有足够的稳定性。