**用jemalloc替换 东方深秘录 中的内存分配器**

**一、概述**

 在《东方深秘录 反调试分析》中提到一个奇怪的现象：当使用调试器加载程序时，游戏会运行得非常缓慢、卡顿。经过分析后得知，这是由于动态内存分配函数HeapAlloc与HeapFree在此情况下运行速度比平常慢导致的。文中给出的解决方案之一是把程序中原来的内存分配器用高效的内存分配器替换掉。本文的目的就是实现这一解决方案。

 本文中，所选用的高效内存分配器是jemalloc。它从 2005 年开始被 FreeBSD 的 libc 所采用，是速度最快的内存分配器之一。

**二、C语言动态内存分配函数简介**

 C语言的动态内存分配函数主要由4个函数组成：

1. malloc(n): 分配一段大小为n的内存
2. calloc(n, s): 分配一段大小为n \* s的已清零内存
3. free(p): 释放p所指向的内存
4. realloc(p, s): 将p指向内存分配的大小重置为s

**三、思路**

 最初的思路很简单，也很直接：

1. 修改程序让程序开始执行时加载jemalloc的动态链接库
2. 修改原始内存分配器的入口点指令，使其跳转到jemalloc中的对应函数

然而，事情并非想象得那么顺利。上面的思路遇到了一个问题，在程序开始执行时，似乎已经有某段代码调用了内存分配函数。因此，旧内存分配器与新内存分配器的衔接便成了问题（主要由于realloc）。因此只好修改原来的思路：

1. 修改程序让程序加载我们自己编写的rmalloc中间层
2. 修改原始内存分配器的入口点指令，使其跳转到rmalloc中的对应函数
3. rmalloc检查调用请求。若传入的内存是旧内存分配器分配的，则交由旧内存分配器处理；若是jemalloc分配的，交由jemalloc处理。

**四、实现**

1. 编写shellcode

shellcode是写入游戏程序里的一段代码，它的作用是加载并调用patch.dll中我们编写的程序。由于没有链接器的支持，我们编写的代码不能直接调用Windows提供的API函数，所以要使用shellcode的方式来寻找API函数的入口点。

1. 动态链接库patch.dll

patch.dll 的主要任务是加载rmalloc.dll，并且把旧内存分配器的入口点修改为跳转到新内存分配器的指令。它还会顺便把patch.dll中的反调试代码禁用掉。

1. 动态链接库rmalloc.dll

rmalloc.dll的主要任务是检查传入的内存指针是由哪个内存分配器管理的，并且交由对应的内存分配器管理。

1. 编译jemalloc

由于我们编写的rmalloc假设jemalloc不向系统释放内存。所以jemalloc必须以 --disable-munmap 选项编译。

**五、rmalloc实现细节**

 rmalloc使用内存指针所在页面来判断该指针是否由jemalloc管理。由于内存页面是按照4KB划分的。在32位系统下，用户空间最多只有2GB/4KB=512K个页面。因此rmalloc使用一个512KB的数组来记录该页面是否由jemalloc管理。

 rmalloc假定jemalloc不向系统释放内存，因此一旦jmelloc返回了一个指向某个新页面的指针，就说明这个页面是由jemalloc管理的，而且这个关系在进程结束前都不会改变。因此rmalloc没有对页面记录数组使用互斥量，这是没有问题的。最坏情况下，两个进程同时写入同一个标记，然而无论写入顺序如何，结果总是正确的。

**六、修改程序**

 我们修改程序时，将shellcode写入到了原来反调试器的代码位置。由于程序加载时会修改用到外部函数的指令，因此我们的shellcode必须避开被修改的位置。为此，我们把原始的shellcode分割为三段，每段之间用JMP指令跳转。

 另外我们还修改了程序入口点的代码，这样程序刚开始执行时就会立刻运行我们编写的代码。

**七、总结**

 通过上述的修改，程序运行时首先会运行shellcode，它将找到LoadLibraryA与GetProcAddress的地址，并使用它们加载patch.dll和调用patch()函数。紧接着patch()函数加载并初始化rmalloc.dll，同时修改程序的动态内存分配代码的入口点，另外还禁用掉了反调试代码。在rmalloc.dll初始化时，会加载jemalloc.dll。

 当程序调用动态内存分配函数时，由于入口点已被修改，所以会跳转到rmalloc中的对应函数。malloc会根据内存指针判断是调用jemalloc还是调用原来的内存分配器。

**八、参考资料**

1. 动态定位API的shellcode （编写shellcode）

http://www.cnblogs.com/unixstudio/archive/2012/11/09/2761883.html

1. 汇编语言---GCC内联汇编 （编写shellcode）

http://www.cnblogs.com/taek/archive/2012/02/05/2338838.html

1. 在 Windows 上搭建 MSYS + TDM-GCC 编译环境 （编译jemalloc）

http://www.zbyzbyzby.com/other/msys\_tdmgcc\_on\_windows