《API函数自搜索》实验报告

**实验名称：**

API函数自搜索

**实验要求：**

复现第五章实验七，基于示例5-11，完成API函数自搜索的实验，将生成的exe程序，复制到windows 10操作系统里验证是否成功。

**实验过程：**

1. **进入VC6新建项目并写入如下代码：**

通用型shellcode编写的步骤：

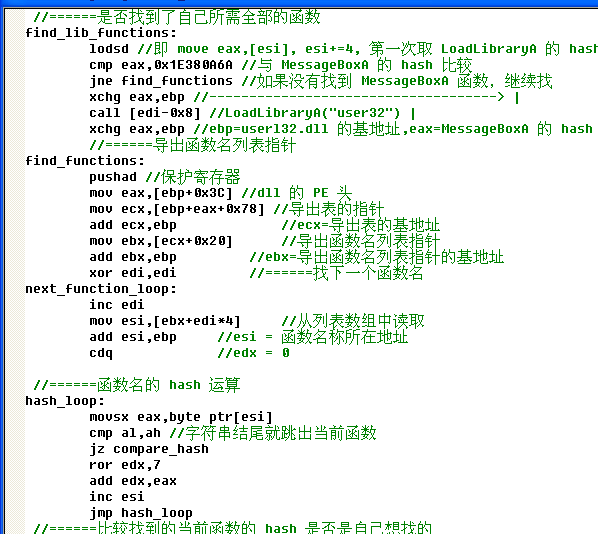
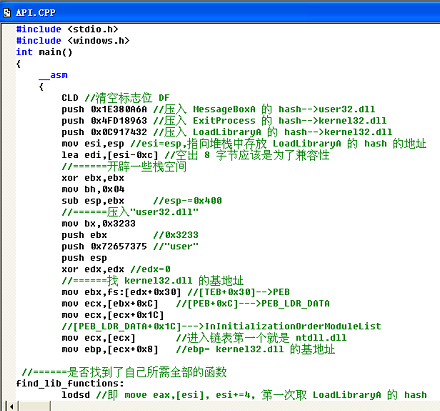
（1）第一步：定位kernel32.dll。

（2）第二步：定位kernel32.dll的导出表。

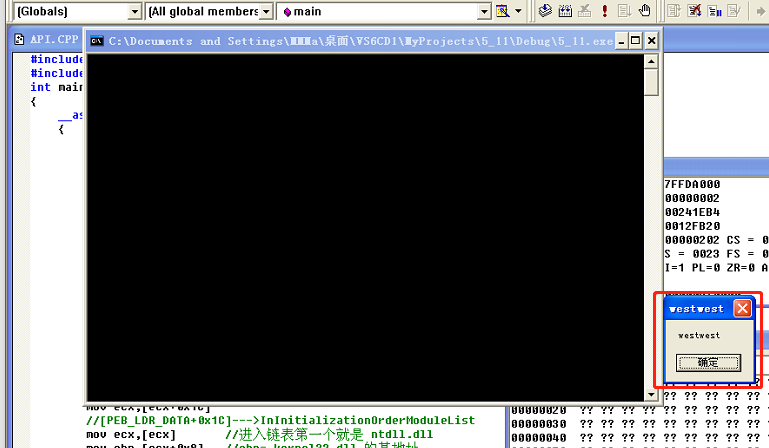
（3）第三步：搜索定位LoadLibrary等目标函数。

（4）第四步：基于找到的函数地址，完成Shellcode的编写。

编写完整API函数自搜索代码程序，Shellcode功能为弹出Messagebox，内容为westwest。代码图如下：



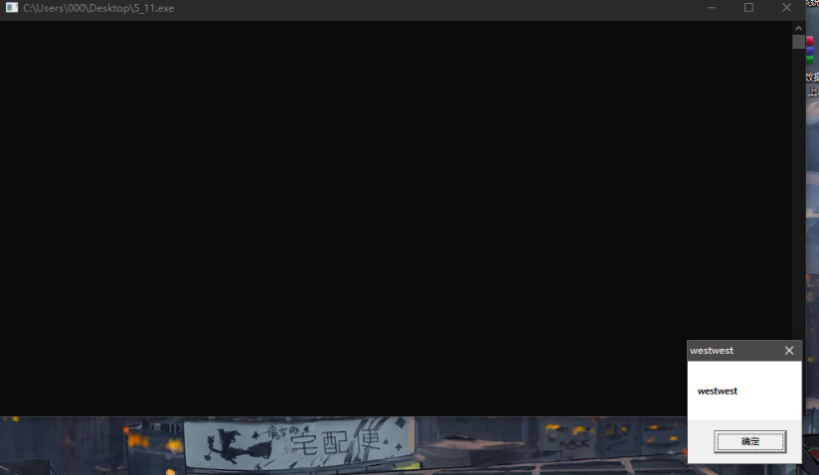
1. **运行代码检验程序。**



发现成功实现预期功能，说明API函数自搜索成功。

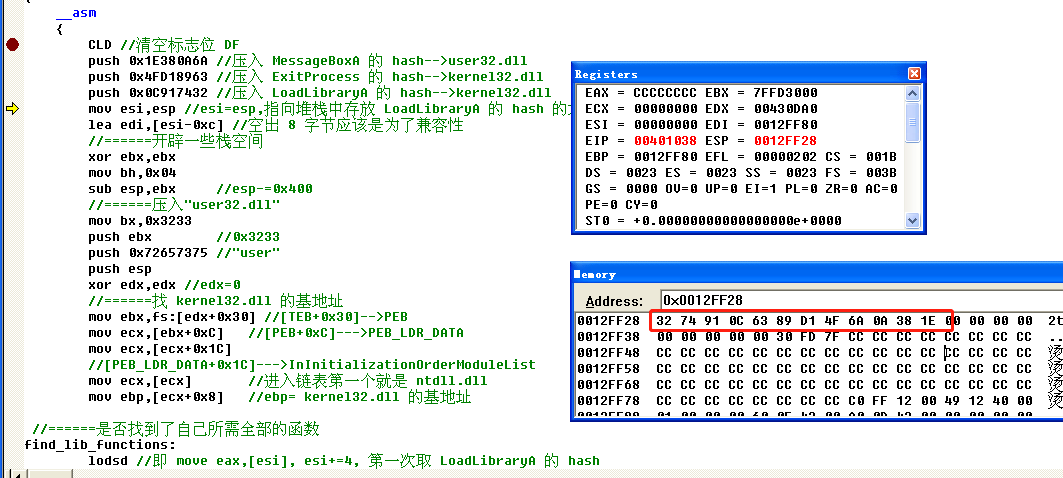
**3. 将生成的exe文件传入windows10中进行验证**

运行如图：



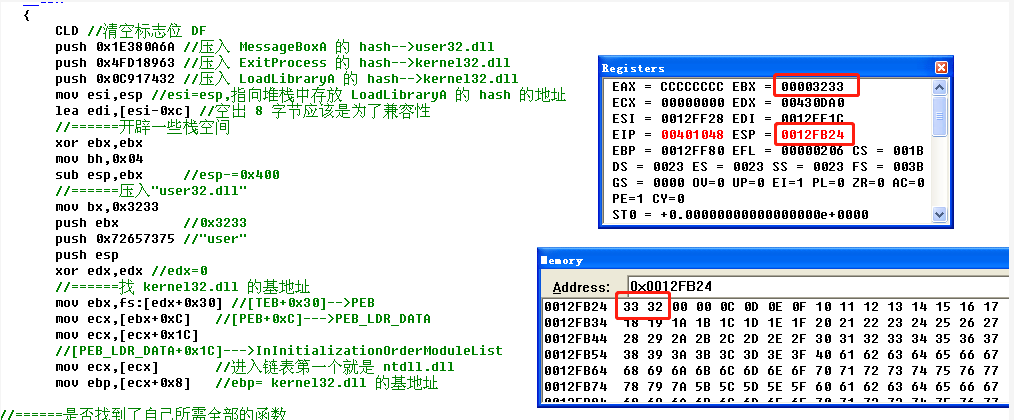
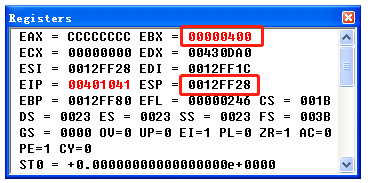
实现预期功能，成功实现了API函数的自搜索。

**4. 对代码添加断点进行调试分析**

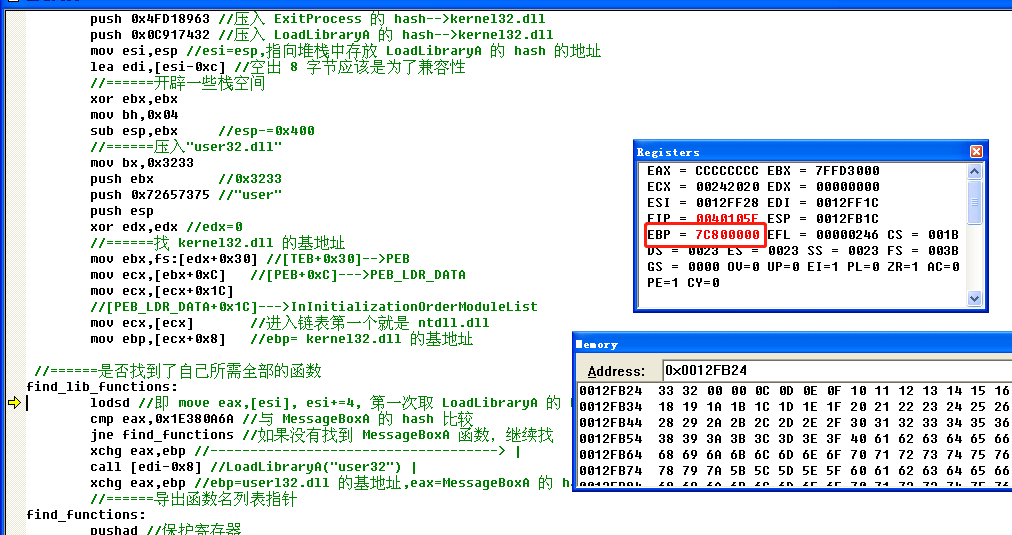
****

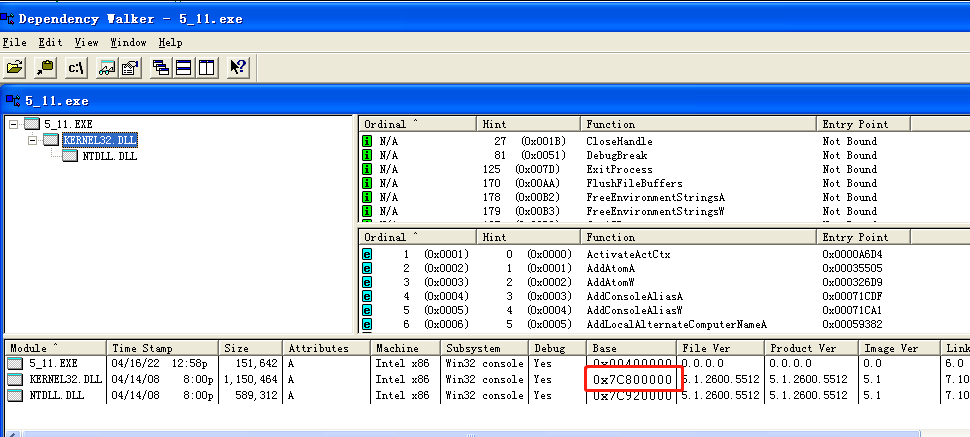
首先，程序开头清空DF标志位，确保si和di递增，然后分别压入MessageBoxA、ExitProcess、LoadLibraryA三个函数的hash值。如图，通过ESP查看栈中内容发现为三个函数hash值

之后将esi赋为esp，此时esi指向堆栈中存放LoadLibraryA的地址。之后通过lea指令让edi保存三个哈希值的地址。



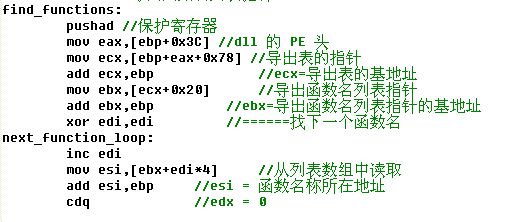
之后通过xor将ebx值变为0后将0x04放入bh再对esp减，即将esp抬高了0x400，开辟足够的栈空间。之后将user32入栈，此时esp即指向存储user32的地址，再将esp入栈。如上图，EBX为0x0400，ESP的值减了0x0400，根据ESP可以看到栈帧中为0x3233。



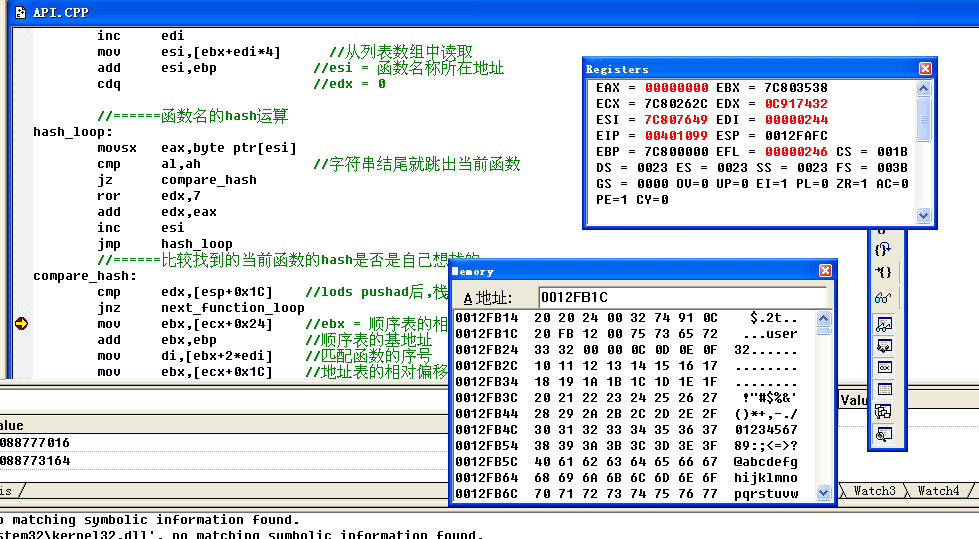
之后运行find\_lib\_functions，通过段选择字FS在内存中找到当前线程环境块TEB，偏移0x30找到指向进程环境块PEB的指针，其中存放着被装在进进程的动态链接库信息， PEB\_LDR\_DATA结构体偏移位置为0x1C的地址存放着指向模块初始化链表的头指针InInitializationOrderModuleList，其中按顺序存放着 PE 装入运行时初始化模块的信息，第二个链表结点就是 kernel32.dll。kernel32.dll的结点再偏移0x08就是 kernel32.dll 在内存中的加载基地址，将存入ebp中。即如图中，ebp值为0x7c800000。

如上图，利用Depends工具查看Kernel32.dll的加载基址发现确实是0x7c800000。

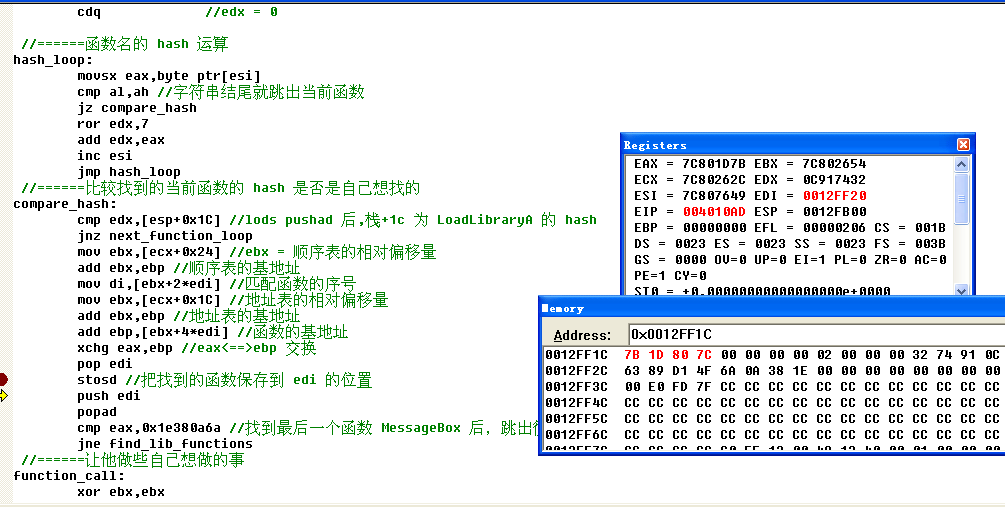
之后进入find\_lib\_functions判断是否找到所需全部函数，第一次取LoadlibraryA的hash值，之后与MessageBoxA函数的哈希值比较判断是否寻找结束。



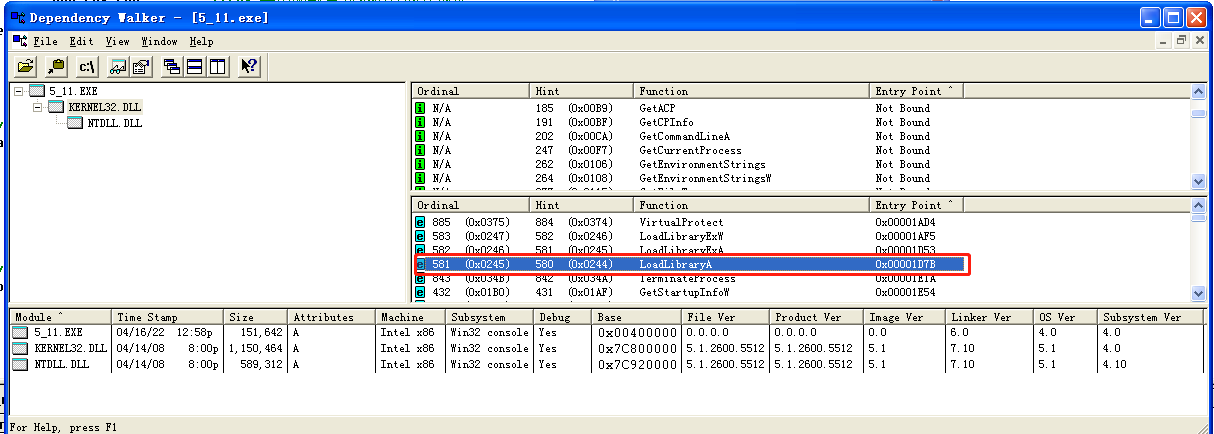
未找到则跳转至find\_functions函数寻找函数名列表指针，eax存入dll的PE头，ecx存入导出表的基地址，ebx存入导出函数名列表指针的基地址。从列表数组中依次取值后进入hash\_loop计算其哈希值，计算完毕后跳转至compare\_hash进行比较，如果不是则返回next\_function\_loop寻找下一个函数继续进行比较，直至成功找到。



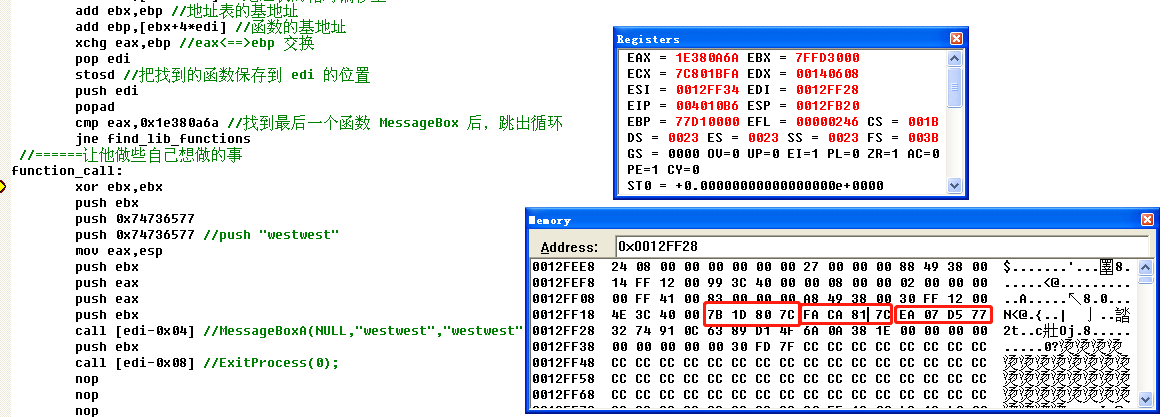
之后算出虚拟地址并存到edi所指向地址，如下图此时EDI为0012FF1C，存储着函数的虚拟地址0x7C801D7B。



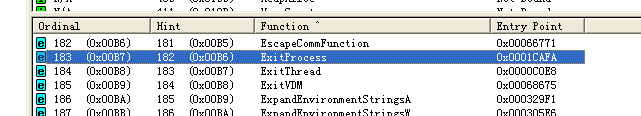
如下图，利用Depends工具查看后发现第一个函数LoadLibraryA的偏移地址为0x00001D78，加上Kernel32.dll的基址后确实为上图栈中的内容。



不断重复上述过程直到找到MessageBox函数，找到则调用LoadLibraryA（“user32”），此时EBP为77D10000，为user32的基地址，程序结束。edi指向的地址存放着三个函数的基地址。



如中间的FACA817C，如下图，通过Depends工具查看可知正是ExitProcess的地址（偏移地址0x0001CAFA+基址0x7C800000 = 0x7C81CAFA）



最后进入shellcode，利用前面找到的函数，输出了弹窗“westwest”，最后退出程序。

**心得体会：**

本次实验加深了我对于windows中程序运行的理解，对于运行中程序的结构一节API函数的位置有了进一步认识，学会了搜索到API函数，此外也学会了如何使用Depends工具进行反汇编，对于汇编语言中寄存器和指针使用也更加熟练。此外，通过本次实验我也认识到，程序的漏洞很难避免，尤其是程序的特性和结构被人了解和掌握之后，找到漏洞也变得更容易，。