《软件安全》实验报告

姓名：邢清画 学号： 2211999 班级： 1023（物联网）

**实验名称：**

API函数自搜索实验

**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。

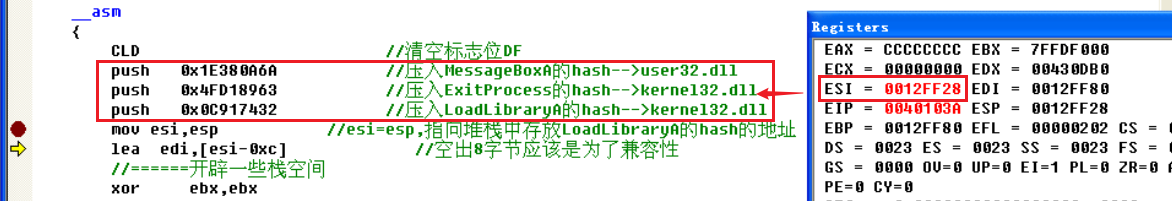
**实验过程：**

1. 复现第五章实验七

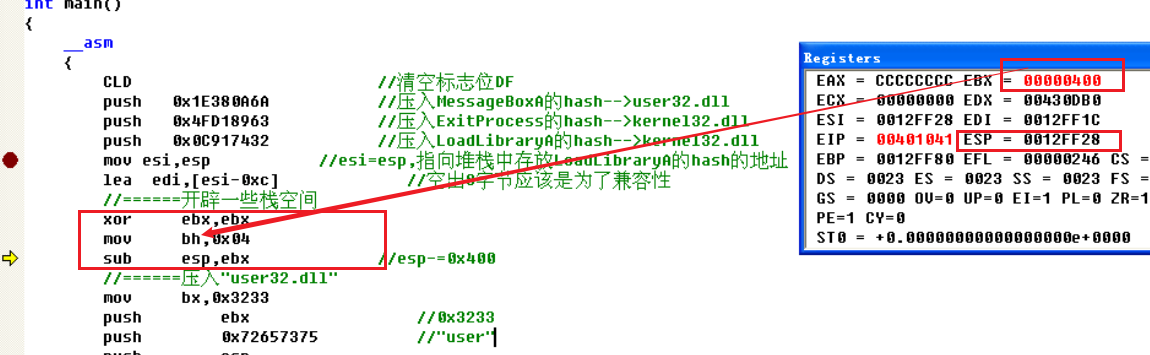
通过加断点在合适的位置，便于了解API函数自搜索的机制。

断点mov esi,esp 前的3个push，用来push MessageBoxA，ExitProcess和LoadLibraryA的哈希值（通过独立的程序算出），在后面做函数名的比较时，比较的是哈希值而不是字符串。

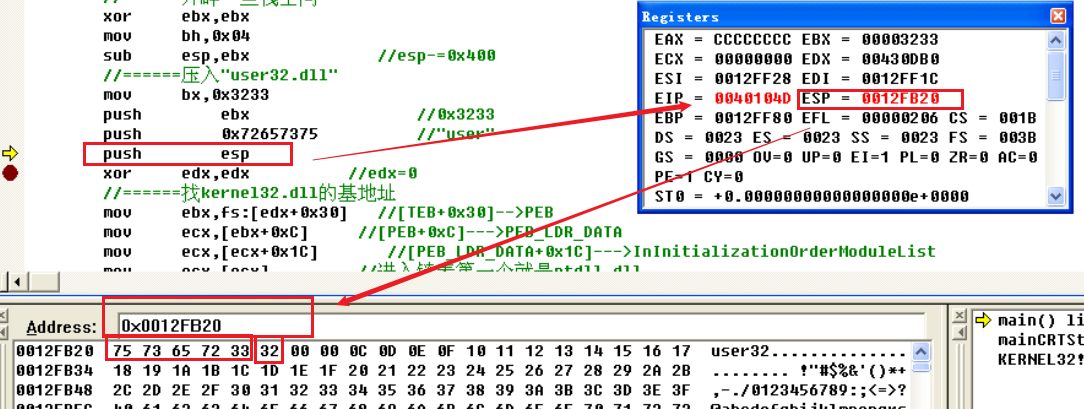
Esi的值0012FF28在整个程序里面始终不会改变，始终指向在栈中的三个哈希值.

mov bh,0x04抬高bh,影响的是EBP的值。

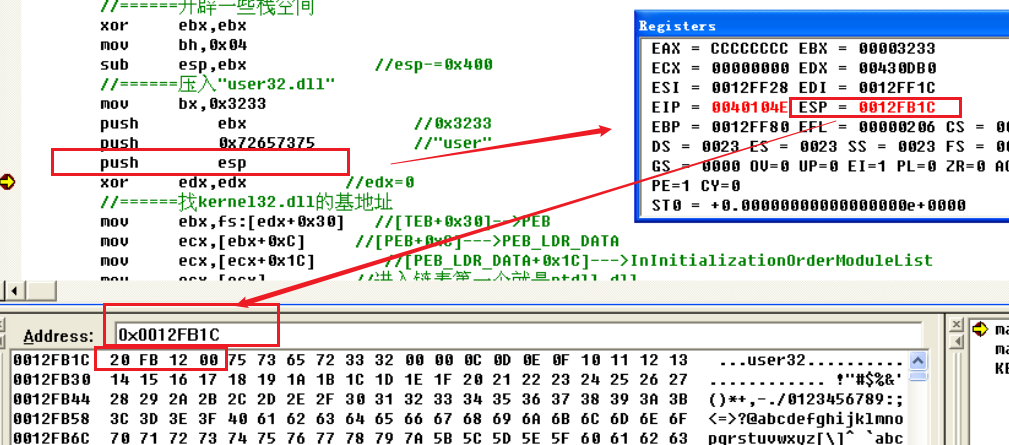
sub esp, ebx从栈顶指针esp中减去ebp的值，把esp抬高（值变小）。



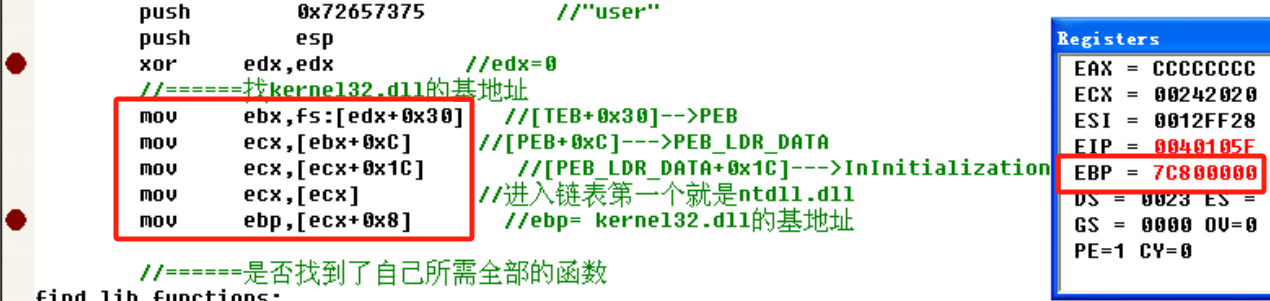
之后压入"user32.dll"，两个push用来处理字符串“user”和“32”，

地址压栈，用于后续的LoadLibraryA调用，0x72657375是"user"的ASCII码。

之后push esp，此时esp中存储的值变小，现在的0x0012FB1C存储的是之前的0x0012FB20（user32 字符串的地址）

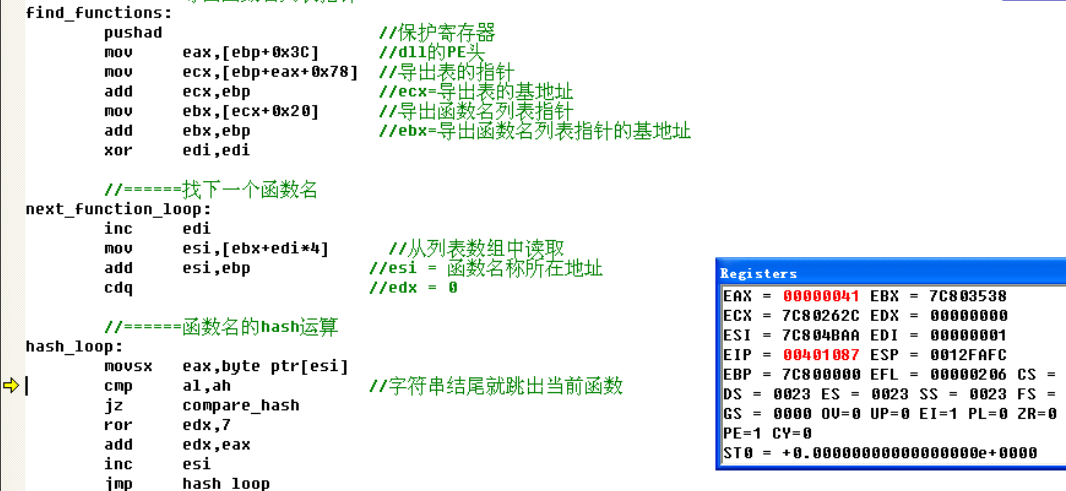


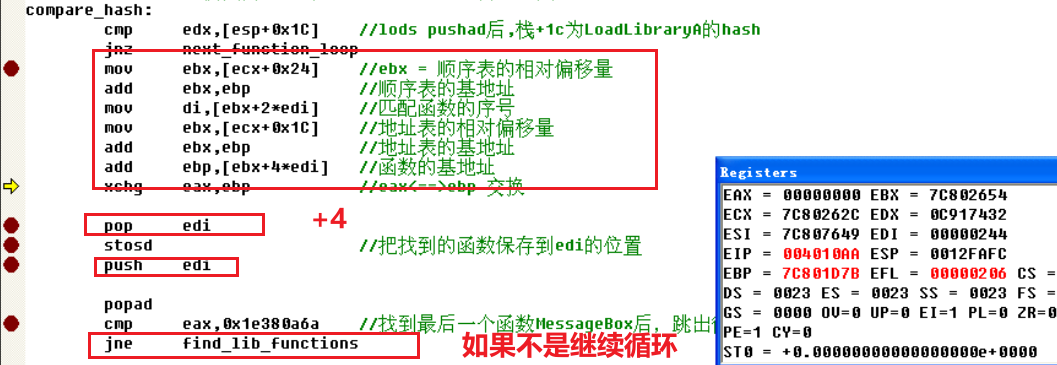
然后获取PEB的地址，访问PEB\_LDR\_DATA结构来遍历加载的模块列表，动态解析函数地址，最终找到kernel32.dll的基地址（EBP存储的7C800000）。



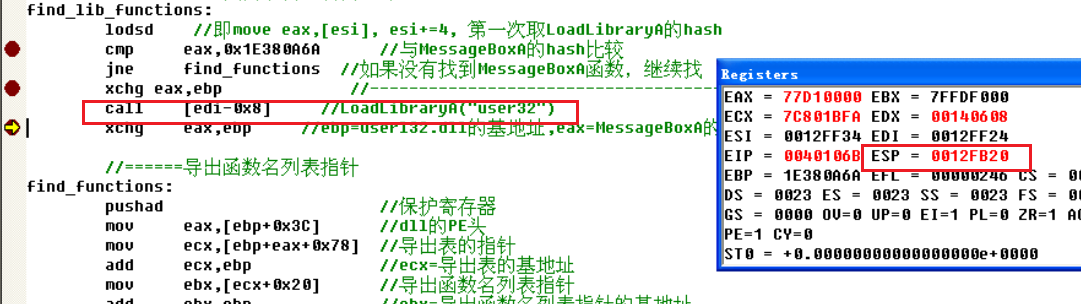
Lodsd从esi指向的地址加载一个值到eax并将esi增加4，用来逐个检查压栈的哈希值。如0C917432就是LoadLibraryA的hash。

Cmp作比较，判断是否是最后一个需要查找的哈希值，如果不是，jne跳转到find\_functions中，通过访问DLL的PE头和导出表，可以找到函数名和它们对应的实际地址。在导出表中逐一检查每个函数名。且ebp会被更新为目标函数的地址。不断寻找，直到找到后开始计算虚拟地址，最后把找到的函数保存到edi中。

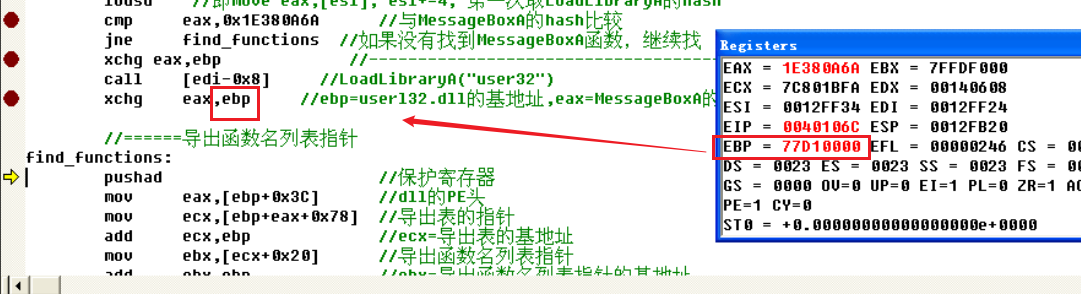




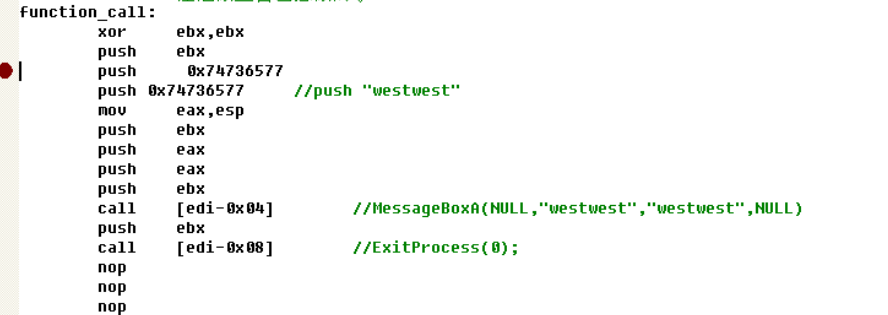
不断重复上述过程，直到找到最后一个函数的hash，跳出循环，执行call语句，观察到esp的值是0x0012FB20，即user32字符串的地址，完成了LoadLibraryA("user32")的调用。



继续执行观察到ebp的值发生变化，77D10000其实是user32.dll的基地址。

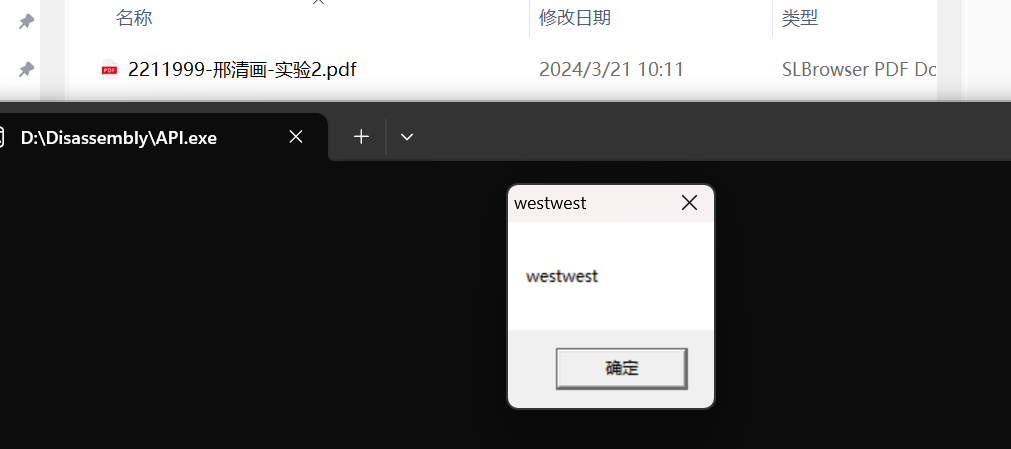


构造调用MessageBoxA的参数，并调用ExitProcess函数以结束进程，使用NOP指令作为填充。



在VC6中运行显示westwest对话框：

2. **将生成的exe程序，复制到windows 10操作系统里验证成功。**



**心得体会：**

学会了动态定位函数API的方法，更加了解PE文件结构，掌握了如何提高Shellcode的跨平台性和适应性，加深了我对汇编语言的理解，提高了汇编语言能力。