《漏洞利用及渗透测试基础》实验报告

姓名：袁田　学号：2314022　班级：计科三班

实验名称：

API函数自搜索定位技术

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.首先复现代码并对其进行分析，代码如下：

```
#include <stdio.h>
#include <windows.h>
int main()
{
    __asm
    {
            CLD
            push 0x1E380A6A
            push 0x4FD18963
            push 0x0C917432
            mov esi,esp
            lea edi,[esi-0xc]

            xor ebx,ebx
            mov bh,0x04
            sub esp,ebx

            mov bx,0x3233
            push ebx
            push 0x72657375
            push esp
            xor edx,edx

            mov ebx,fs:[edx+0x30]
            mov ecx,[ebx+0xC]
            mov ecx,[ecx+0x1C]
            mov ecx,[ecx]
            mov ebp,[ecx+0x8]
    }
```

(1)首先将三个函数的hash压入栈中，后续程序中对函数名的比较均为对hash值的比较。此时栈顶指针esp记录的为最后一个函数的hash，用寄存器edi记录esp值，即保存三个函数的hash，可以看到此时地址0x0012FF28的位置记录着LoadLibraryA的hash

```
CLD
push 0x1E380A6A
push 0x4FD18963
push 0x0C917432
mov esi,esp
lea edi,[esi-0xc]

xor ebx,ebx
mov bh,0x04
sub esp,ebx

mov bx,0x3233
push ebx
```

```
地址:  0x0012FF28
0012FF28  32 74 91 0C  2t..
0012FF2C  63 89 D1 4F  c壮O
```

```
EAX = CCCCCCCC EBX = 7FFDC000
ECX = 00000000 EDX = 00380E18
ESI = 0012FF28 EDI = 0012FF80
EIP = 0040103A ESP = 0012FF28
```

(2)随后将在ebx中记录0x3233(即ASCII码中的“23”)，并向栈中压入"user"，此时将esp记录了“user32”，于是将esp压入栈中保存数据

```
mov bx,0x3233
push ebx
push 0x72657375
push esp
xor edx,edx

mov ebx,fs:[edx+0x30]
mov ecx,[ebx+0xC]
mov ecx,[ecx+0x1C]
```

```
EAX = CCCCCCCC EBX = 00003233
ECX = 00000000 EDX = 00380E18
ESI = 0012FF28 EDI = 0012FF1C
EIP = 0040104D ESP = 0012FB20
EBP = 0012FF80 EFL = 00000206
```

```
地址:  0x0012FB20
0012FB20  75 73 65 72 33 32 00  user32.
0012FB27  00 0C 0D 0E 0F 10 11  .......
0012FB2E  12 13 14 15 16 17 18  .......
```

(3)随后向ebp中存入kernel32的基地址，此时ebp=7C800000

```
mov ebp,[ecx+0x8]

find_lib_functions:
lodsd
cmp eax,0x1E380A6A
jne find_functions
```

```
EAX = CCCCCCCC EBX = 7FFDC000
ECX = 00242020 EDX = 00000000
ESI = 0012FF28 EDI = 0012FF1C
EIP = 0040105F ESP = 0012FB1C
EBP = 7C800000 EFL = 00000246
```

（4）进入函数find_lib_function，lodsd 将 ESI 第一次存储的 loadlibrary 的哈希值传递给eax

```
find_lib_functions:
            lodsd
            cmp eax,0x1E380A6A
            jne find_functions
            xchg eax,ebp
            call [edi-0x8]
            xchg eax,ebp
```

该函数实现的功能为：检查eax是否为LoadLibraryA函数的hash，若是，则调用函数LoadLibrary("user32"),否则进入到函数find_functions，导出函数名列表指针；随后进入函数next_hash_loop，从列表数组中一个个读取，然后在函数hash_loop中计算其hash值

```
find_functions:
            pushad
            mov eax,[ebp+0x3C]
            mov ecx,[ebp+eax+0x78]
            add ecx,ebp
            mov ebx,[ecx+0x20]
            add ebx,ebp
            xor edi,edi

next_function_loop:
            inc edi
            mov esi,[ebx+edi*4]
            add esi,ebp
            cdq

hash_loop:
            movsx eax,byte ptr[esi]
            cmp al,ah
            jz compare_hash
            ror edx,7
            add edx,eax
            inc esi
            jmp hash_loop
```

(5)在hash计算结束后，进行比较判断是否为当前自己想找的函数哈希值，若不是，则返回继续寻找计算；

如果知道了想找到函数hash值，则进行计算，通过获取相对偏移量和基地址得到函数的基地址

```
compare_hash:
            cmp edx,[esp+0x1C]
            jnz next_function_loop
            mov ebx,[ecx+0x24]
            add ebx,ebp
            mov di,[ebx+2*edi]
            mov ebx,[ecx+0x1C]
            add ebx,ebp
```

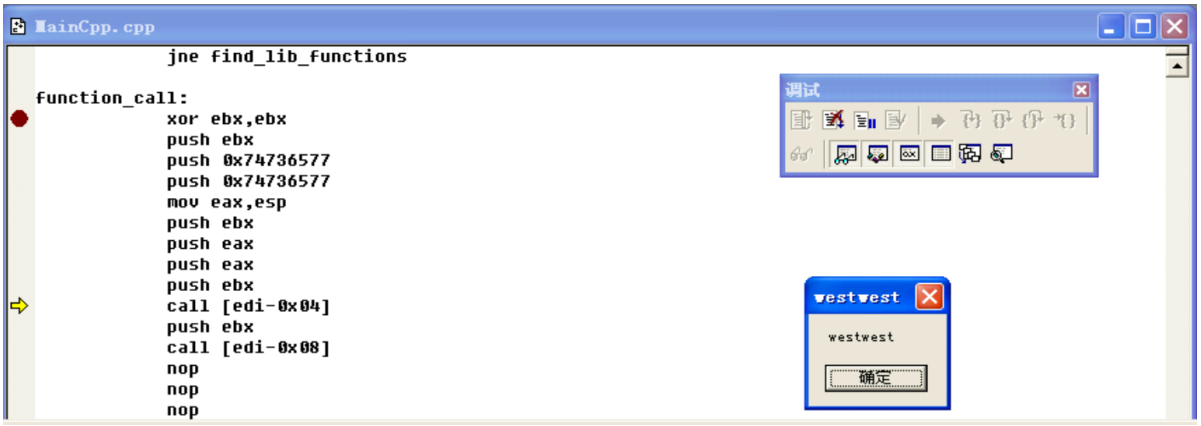
随后将找到函数的虚拟地址存入edi中：

```
pop edi
stosd
push edi
```

popad 保存当前所有寄存器的状态后，cmp 比较当前找到的函数是否为 messagebox，若不是，则跳转到 findlibfunctions；在找到 messsagebox 函数后，则执行find_lib_functions中满足eax=0x1E380A6A的情况(具体代码如下所示)，通过 call 完成对 user32 的调用，ebp原本存的值为 kernel32 的基地址，执行后更改user32 的基地址

```
xchg eax,ebp
call [edi-0x8]
xchg eax,ebp
```

随后在funtion_call中进行调用messagebox函数的shellcode编写，执行代码成功调用 messagebox 函数，且程序在 win10 操作系统中也能成功运行



心得体会：

这段代码没有直接调用LoadLibrary或GetProcAddress，而是手动解析PEB、遍历DLL导出表，通过哈希匹配找到目标函数，掌握这种技术能更深入理解Windows加载器的工作机制，对分析恶意代码或编写安全工具至关重要。