《漏洞利用及渗透测试基础》实验报告

姓名：管昀玫 学号：2013750 班级：计科一班

**实验名称：**

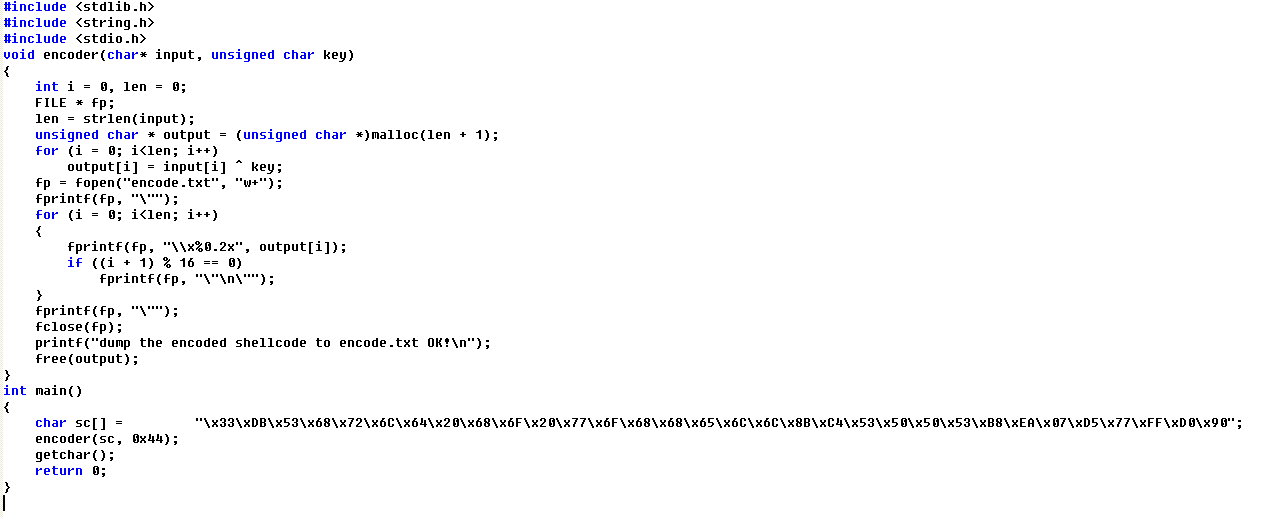
shellcode编写及编码

**实验要求：**

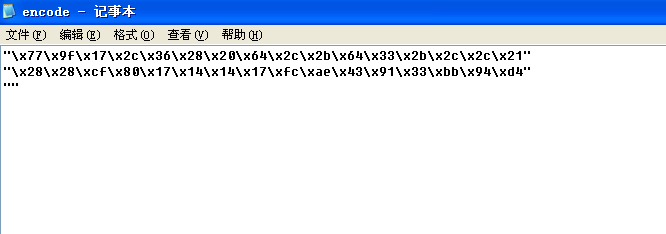
复现第五章实验三，并将产生的编码后的shellcode在示例5-1中进行验证，阐述shellcode编码的原理、shellcode提取的思想。

**实验过程：**

1.编码程序

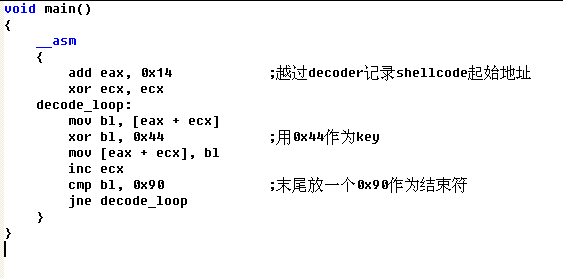


使用异或编码，使0等特殊字符不再为0.在此使用0x44作为秘钥，0x90作为结束标志。生成文件如下：



此时不能直接利用，而是需要再解码。因此我们需要解码程序。

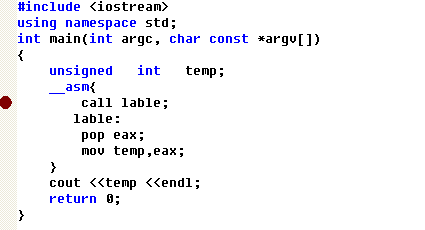
2.解码程序



EAX必须为当前指令的起始地址（第三步将详述如何获取）。Add eax, 0x14之后，EAX将指向这一段程序的下一个地址（因为整段loop的长度为0x14）。将shellcode和这段代码连起来之后，将指向shellcode的起始地址。

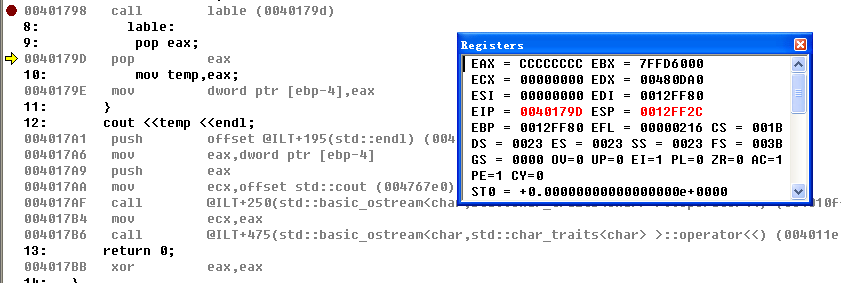
在其中，bl充当一个中间变量，记录了地址，围绕bl的三句代码完成了解码任务。Cmp是用来验证shellcode的末尾是否是0x90.

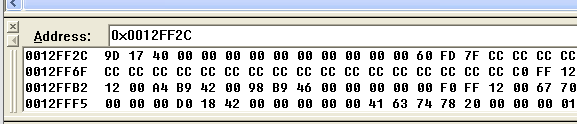
3.得到当前指令的地址



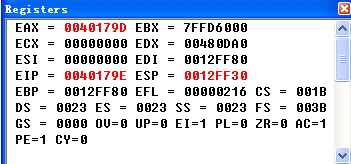


Call将下一条指令的返回地址入栈，之后将Push EIP。按下F10之后进一步查看：





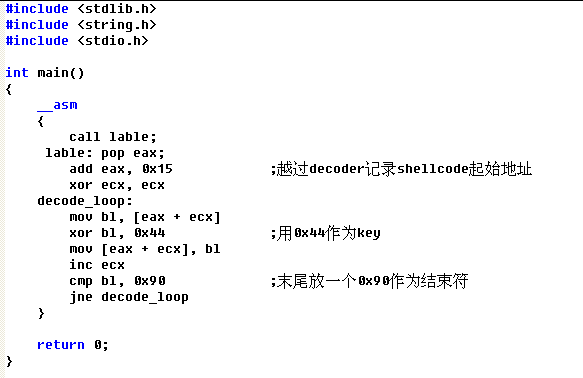
其中存储的正是0040179D。之后又Pop EAX，再次查看EAX的值：

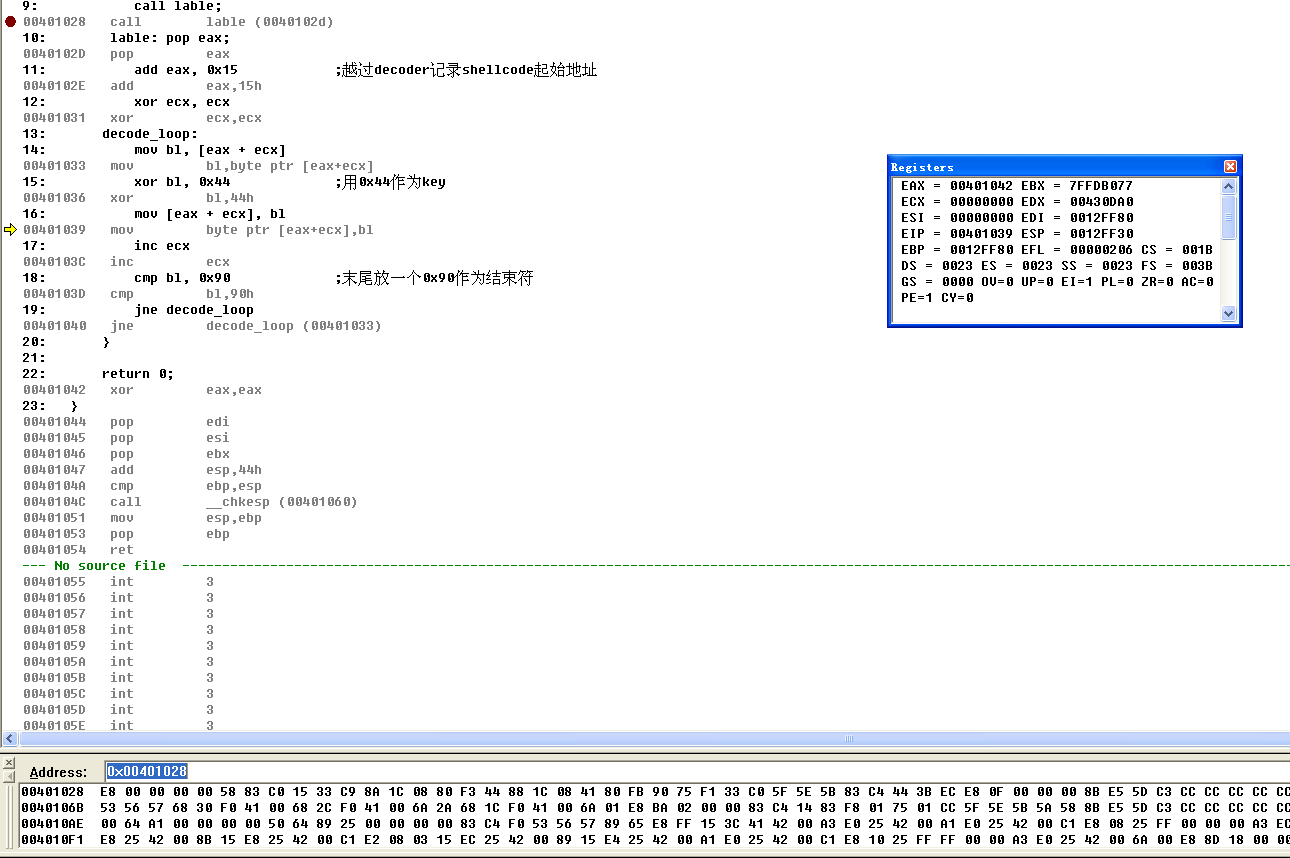


EAX中的值即是pop指令的地址，正是Add eax, 0x14这条指令地址提前一个字节的值。

在这里使用pop eax做示例，其实可以pop其他任意的寄存器。

4. 将以上代码连接，得到一个完整的shellcode



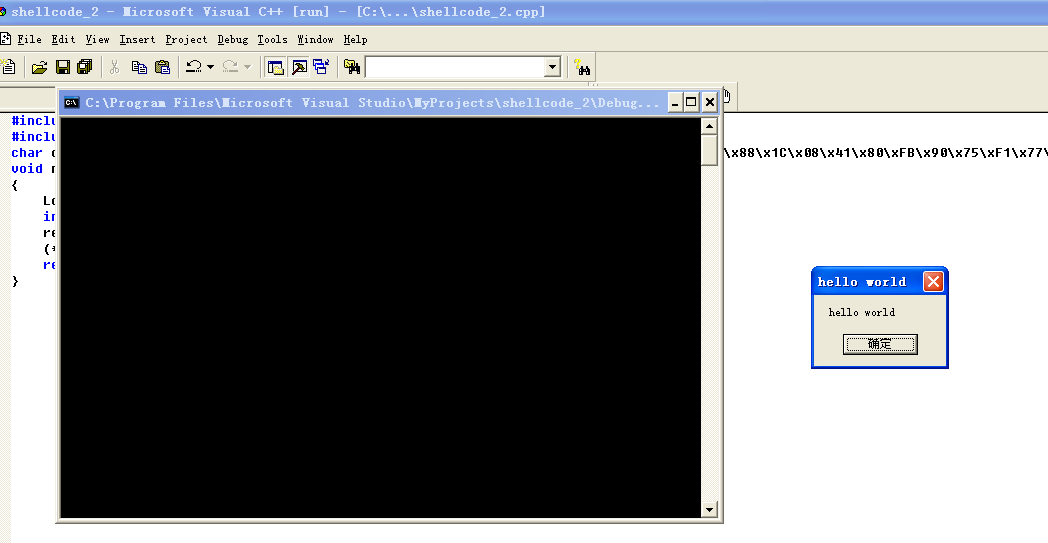


前两句得到EAX的地址，后面是一个解码程序

E8 00 00 00 00 58 83 C0 15 33 C9 8A 1C 08 80 F3 44 88 1C 08 41 80 FB 90 75 F1再加上之前的shellcode（\x33\xDB\x53\x68\x72\x6C\x64\x20\x68\x6F\x20\x77\x6F\x68\x68\x65\x6C\x6C\x8B\xC4\x53\x50\x50\x53\xB8\xEA\x07\xD5\x77\xFF\xD0\x90），再加上解码程序，就能组成一个完整的shellcode。

完整的shellcode如下：\xE8\x00\x00\x00\x00\x58\x83\xC0\x15\x33\xC9\x8A\x1C\x08\x80\xF3\x44\x88\x1C\x08\x41\x80\xFB\x90\x75\xF1\x77\x9f\x17\x2c\x36\x28\x20\x64\x2c\x2b\x64\x33\x2b\x2c\x2c\x21\x28\x28\xcf\x80\x17\x14\x14\x17\xfc\xae\x43\x91\x33\xbb\x94\xd4

之后再次运行shellcode程序



成功得到hello world 窗口。

**心得体会：**

心得：

这个实验的基本逻辑思路还是比较好理解的，也就是跳转到自己指定地址上去执行自己编写的代码；并且为了防止出现有空字符结束的情况导致直接结束，所以需要进行异或的操作防止出现0（同时要注意的是，这个异或操作选取的密钥必须是机器码中没有出现过的，否则异或的时候还是会出现0）。编码过后的机器码自然是不能执行的（或许这个时候不能叫做机器码了），所以需要一个解码程序把编码后的“机器码”解码并跳转执行。

通过这次的实验，感觉到了更多关于漏洞的乐趣，能够把自己想要执行的代码放到程序中，让程序去执行，达到自己想要的结果。

C和汇编混写的代码，可以加快处理的速度和性能。

Shellcode提取的原理和思想：

用C语言书写要提取的shellcode→换成对应的汇编代码→根据汇编代码，找到对应地址中的机器码

利用eax记录shellcode当前的起始地址，提取到机器码，再与想要执行的命令的编码进行拼接，就能得到完整的shellcode。Shellcode就像漏洞利用的弹头，利用如上方法进行漏洞利用并执行恶意程序，会对Windows安全造成极大危害。