《软件安全》实验报告

姓名：刘星宇 学号：2212824 班级：信息安全法学双学位班

**实验名称：**

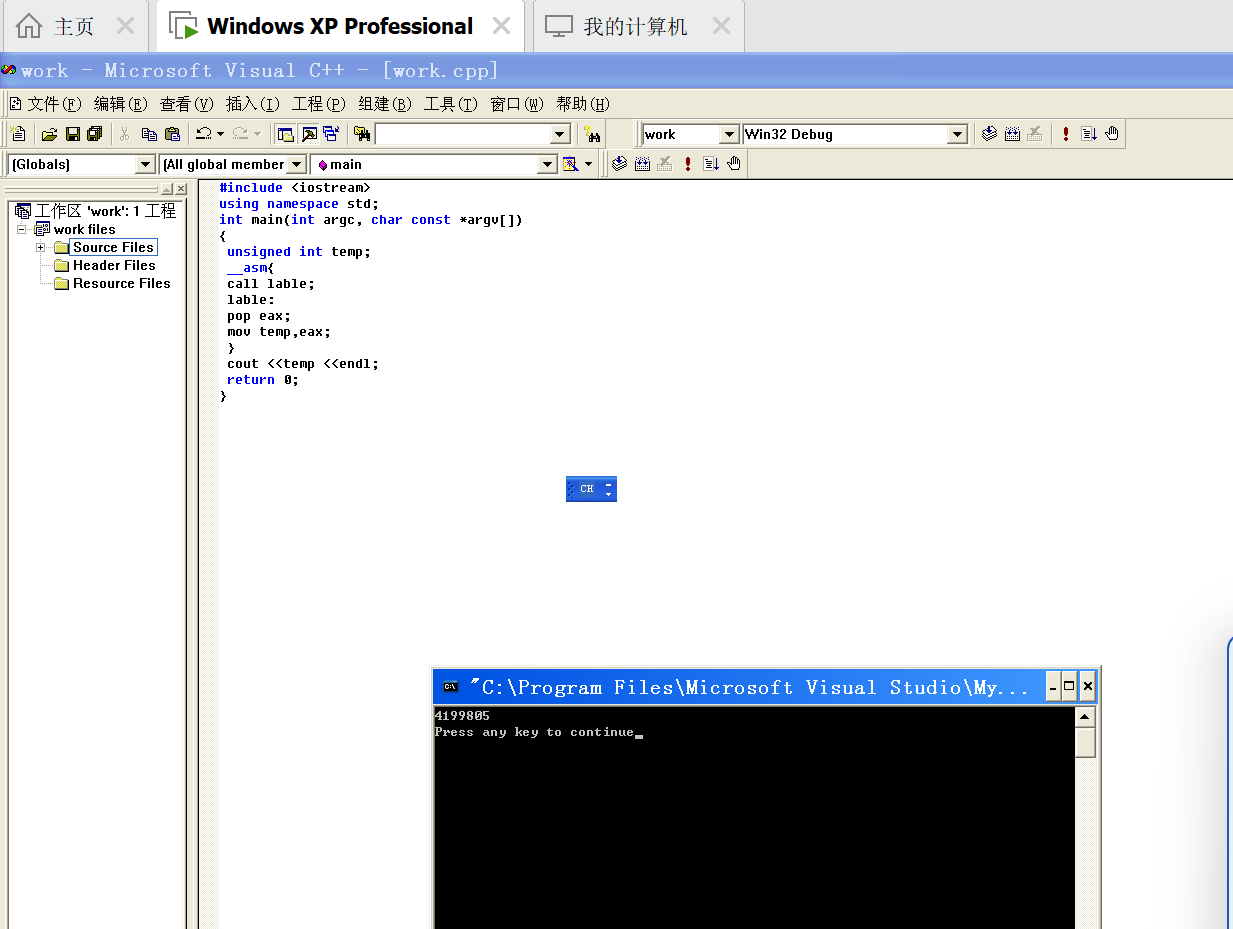
shellcode 进行编码后再进行利用

**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5-4中进行验证,阐述shellcode编码的原理、shellcode提取的思想。

**实验过程：**

1. 复现第五章实验三



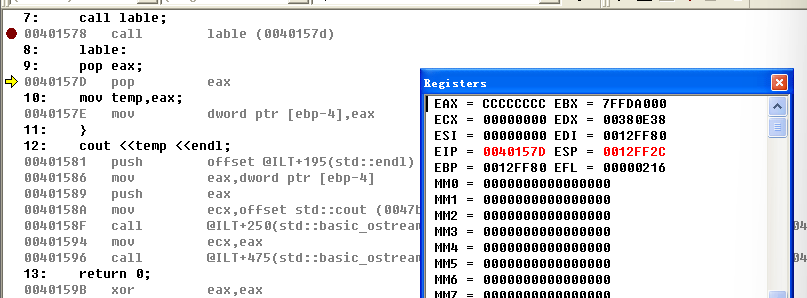
该段代码的核心语句在于、

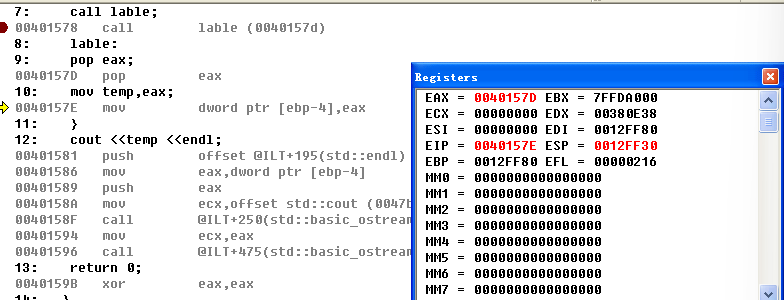
“call lable;

lable:

pop eax;”

之后，eax 的值就是当前指令地址。 call lable的时候，会将当前 EIP 的值入栈。





~~~~~~~

1. 查找产生的编码后的shellcode

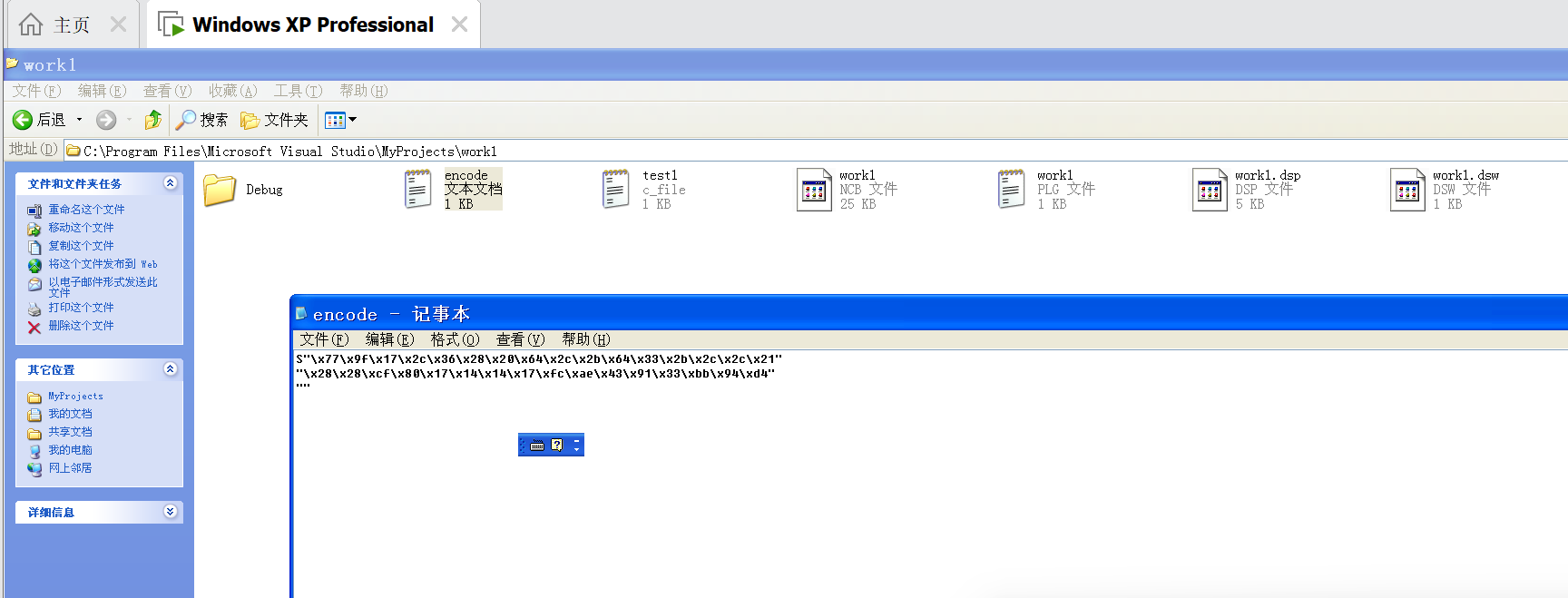


得到提取码为



即"\xE8\x00\x00\x00\x00\x58\x83\xC0\x15\x33\xC9\x8A\x1C\x08\x80\xF3\x44\x88\x1 C\x08\x41\x80\xFB\x90\x75\xF1"

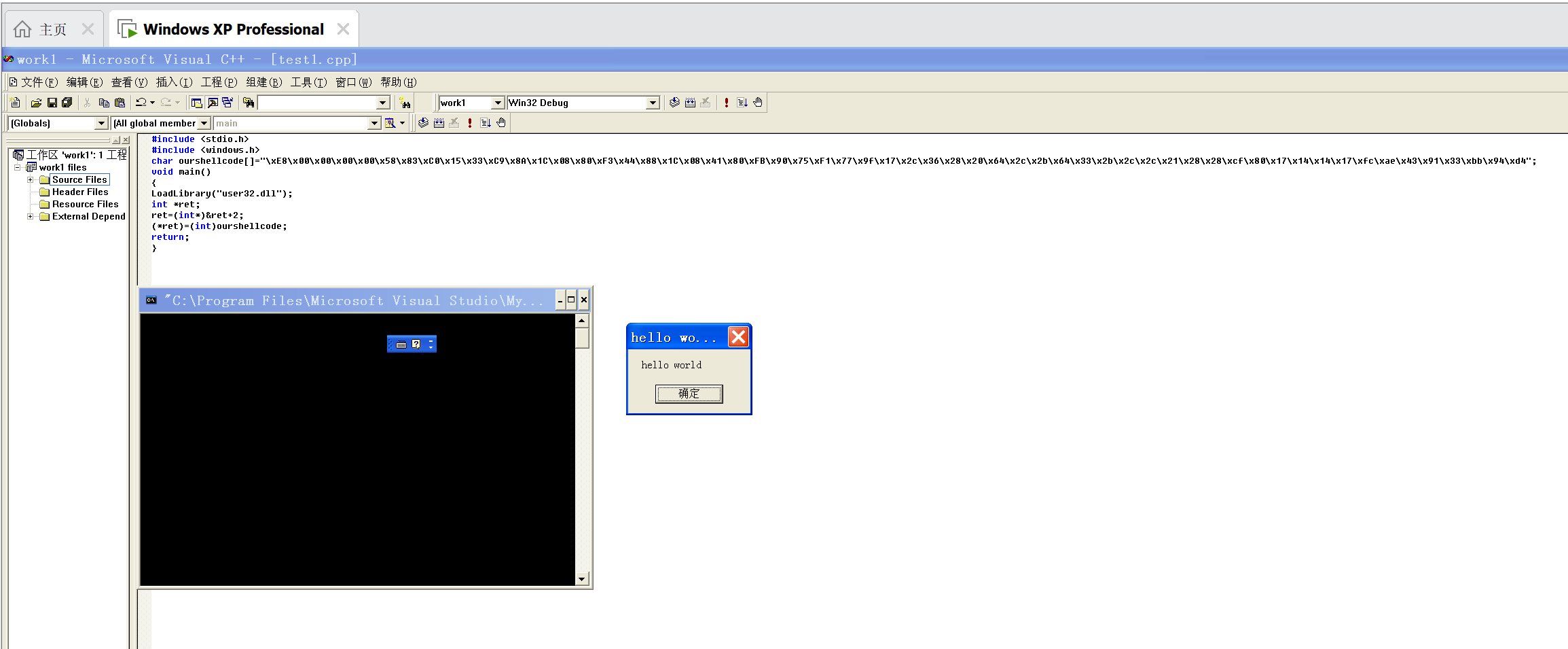
基于示例 5-5 的编码程序，得到调用 Messagebox 输出“hello world”的 Shellcode 的 编码为："\x77\x9f\x17\x2c\x36\x28\x20\x64\x2c\x2b\x64\x33\x2b\x2c\x2c\x21" "\x28\x28\xcf\x80\x17\x14\x14\x17\xfc\xae\x43\x91\x33\xbb\x94\xd4"



链接两段机器码后，得到完整 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"

~~~~~~~

1. 使用示例 5-4 验证 shellcode 的正确性。



验证成功。

**心得体会：**

通过实验，我学会了如何去找当前指令所在的地址：

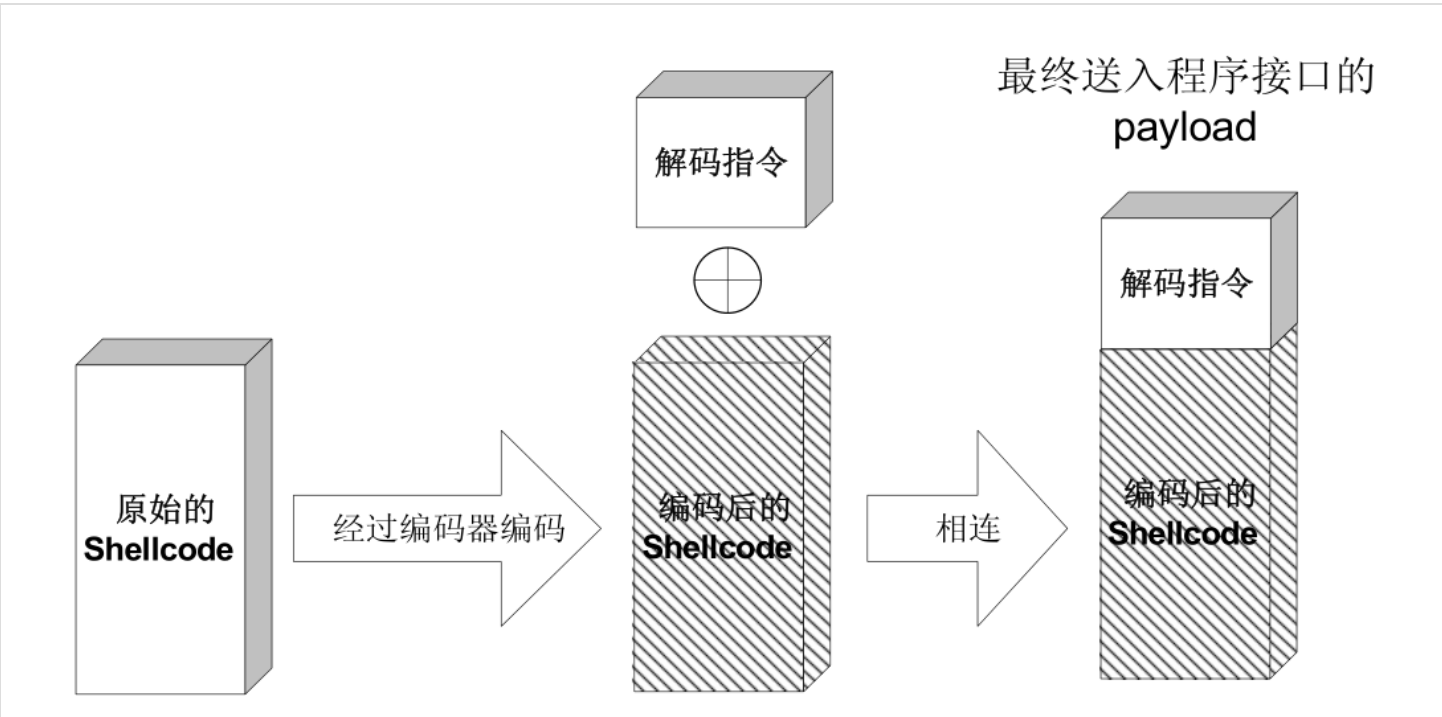
call label；

label：

pop \*\*\*；

同时我还学会了shellcode编码的原理和解码的思想：

1. Shellcode编码的原理



上图非常清晰的展示了shellcode编码的原理。

Shellcode编码的原理主要是基于一种转换过程，其目的在于将原始的shellcode转换为一种能够在特定环境中执行且不易被检测到的形式。这种转换通常涉及对shellcode中的字节序列进行重新排列、替换或加密，以避开安全机制的限制和检测。

原理的核心在于，原始的shellcode可能包含一些在特定环境下被禁止或限制使用的字符或字节序列。。为了绕过这些限制，我们可以对shellcode进行编码，将其转换为一个看似无害或不容易被识别的形式。

然而，仅仅对shellcode进行编码是不够的。因为编码后的shellcode在执行前需要被解码回原始形式。因此，编码过程中通常还会生成一个解码器，这个解码器会在shellcode执行前被注入到目标环境中，负责将编码后的shellcode解码回原始形式。这样，当解码器执行完毕后，原始的shellcode就可以被正确地执行，从而实现攻击者的目的。

2. 解码思想

解码主要通过解码器进行，解码代码。所生成的解码器会与编码后的 shellcode 联合执行。例如本次实验的解码器，默认 EAX 在 shellcode 开始时对准 shellcode 起始位置，之后的代码将每次将 shellcode 的代码异或 特定 key（下例为 0x44）后重新覆盖原先 shellcode 的代码。末尾，放一个空指令 0x90 作为 结束符。