**《软件安全》实验报告**

姓名：禹相祐 学号：2312900 班级：计算机科学与技术

**实验名称：**

**Shellcode 编写以及编码**

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

**实验过程：**

1. **shellcode代码的提取**

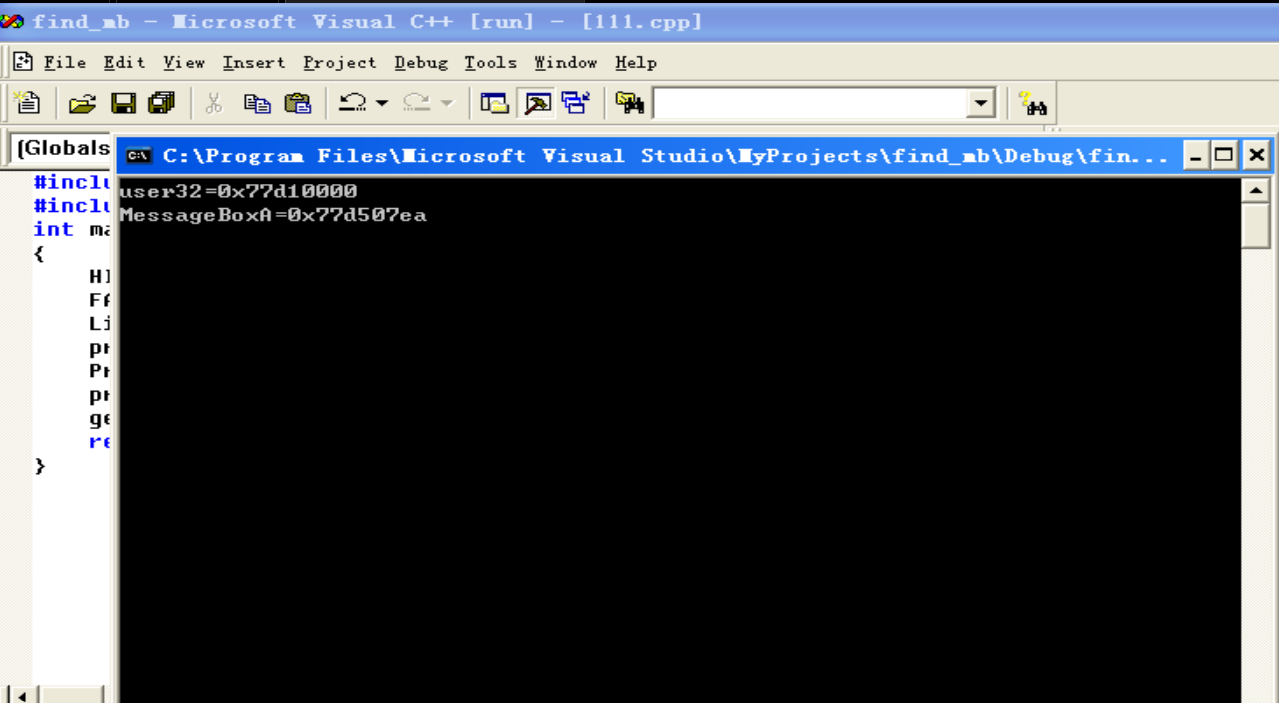
**首先输入5-2内的源代码，并在MessageBox处设置断点，进行反汇编获得指令，如图：**

****

然后寻找出MessageBox在系统中的位置，借助以下代码：

|  |  |  |
| --- | --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14 | #include <windows.h>  #include <stdio.h>  int main()  {  HINSTANCE LibHandle;  FARPROC ProcAdd;  LibHandle **=** LoadLibrary("user32");  printf("user32 = 0x%x \n", LibHandle);  ProcAdd**=**(FARPROC)GetProcAddress(LibHandle,"MessageBoxA");    printf("MessageBoxA = 0x%x \n", ProcAdd);    getchar();  **return** 0;   } |  |

运行结果如图：

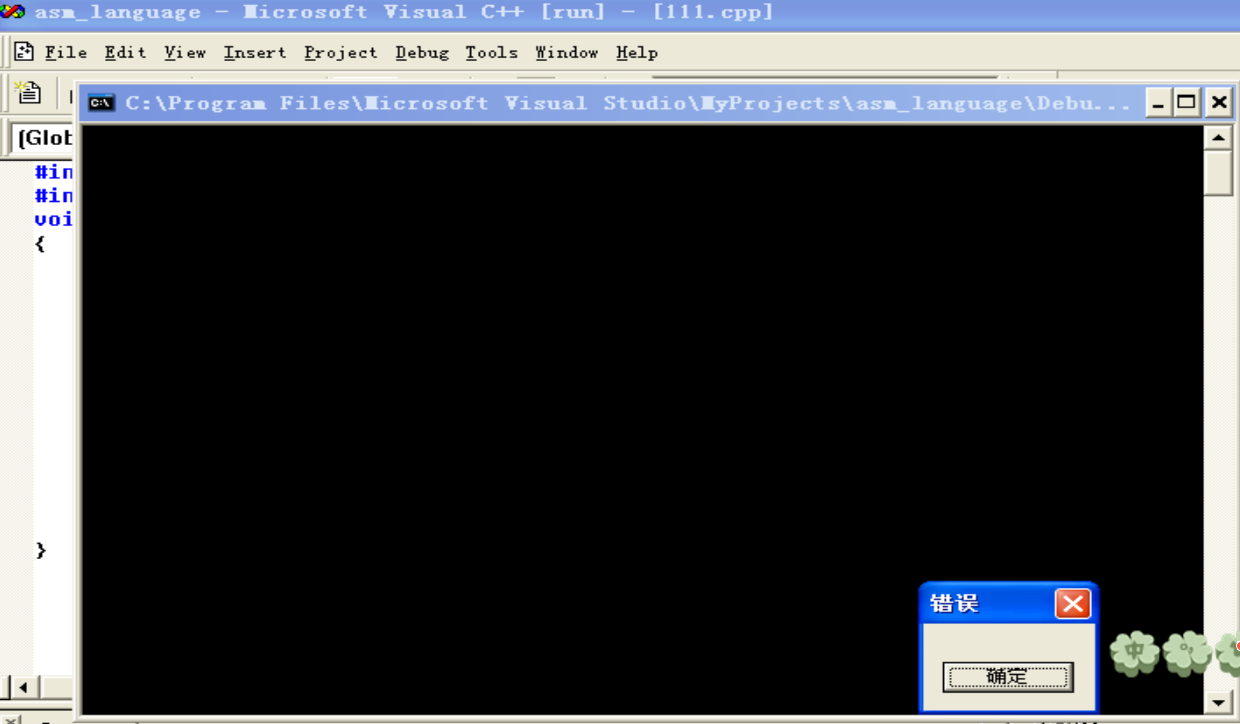


即MessageBox地址为: 0x77d507ea

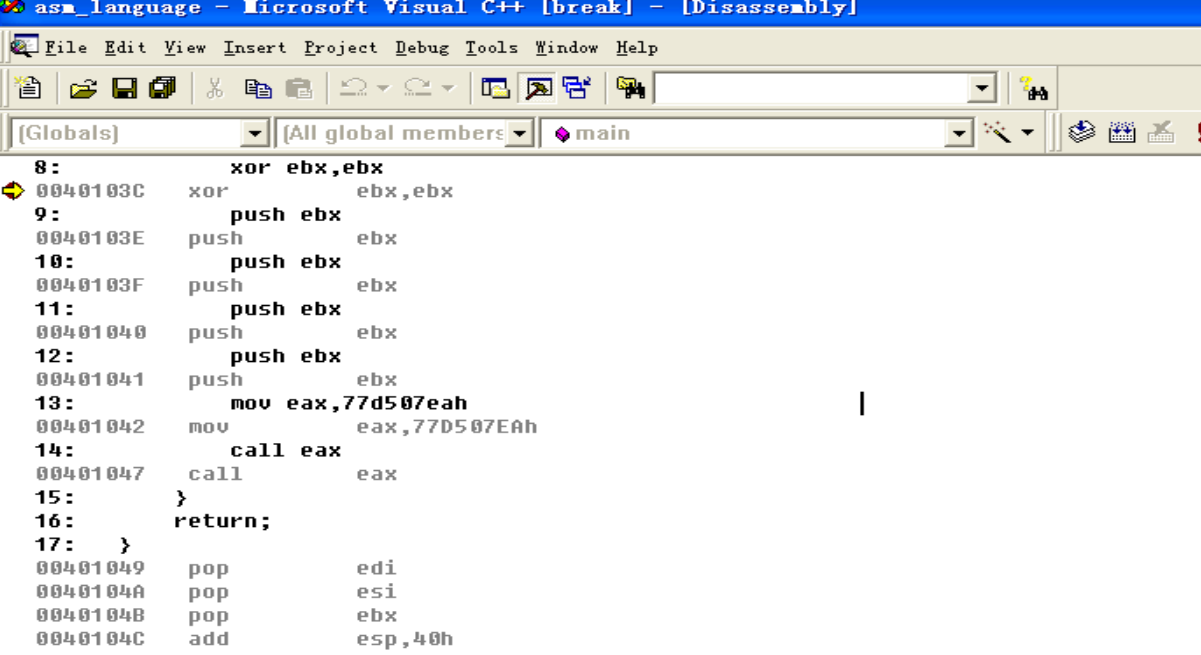
下面我们对汇编语句进行适当修改并利用asm实现在c语言内插入汇编指令，代码如下:

|  |  |  |
| --- | --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17 | #include<stdio.h>  #include<windows.h>  void main()  {  LoadLibrary("user32.dll"):    \_asm   {    xor ebx,ebx    push ebx    push ebx    push ebx    push ebx    mov eax,77d507eah    call eax   }  **return**;   } |  |

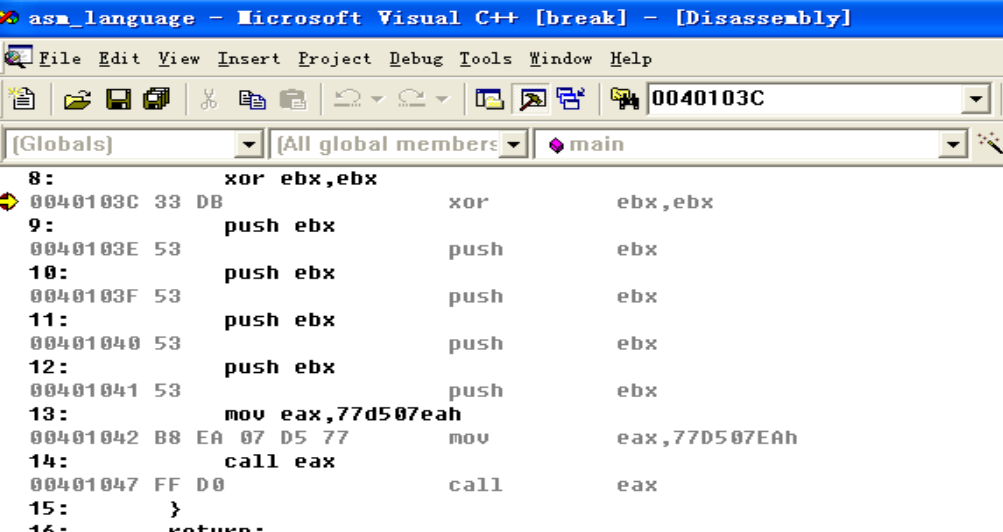
此处不再使用push 0来实现将0寄存器入栈，而是使用ebx与自身异或xor得到0后将ebx推入栈，运行如下图：



接下来查看地址，如图：



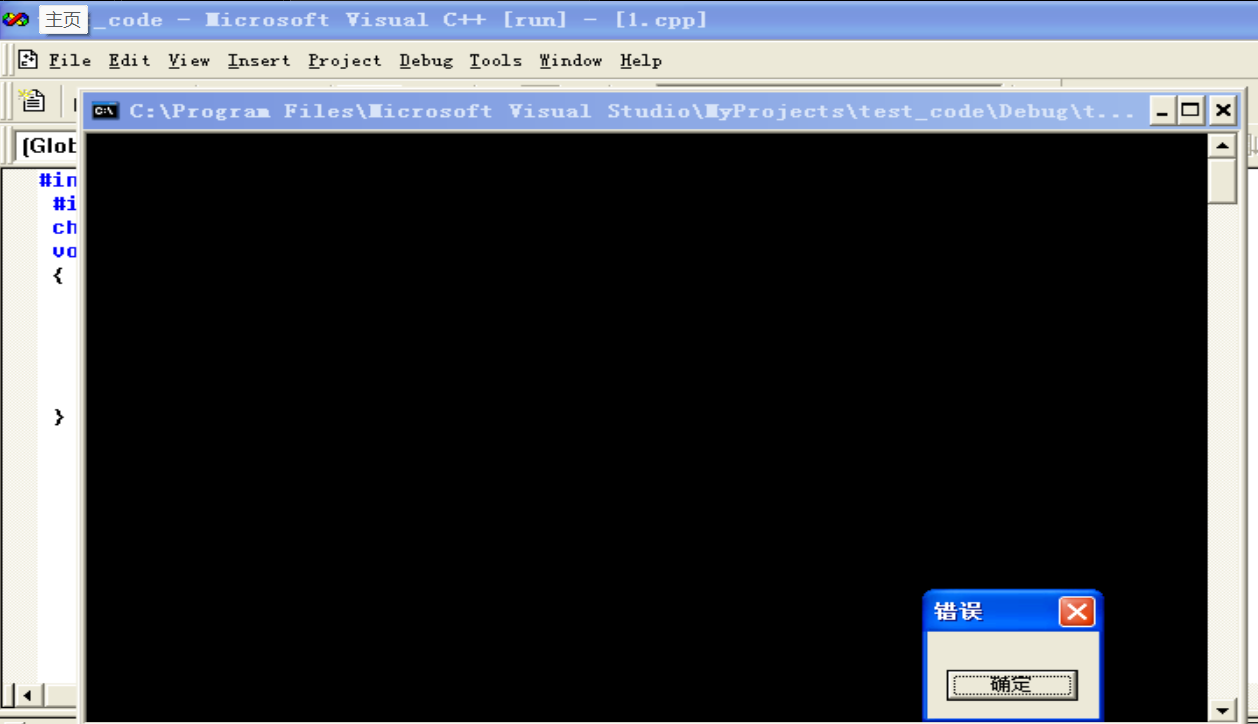
不难知道，编写的asm内的地址由0040103C到00401048，机器码如下，即：**33 DB 53 53 53 53 B8 EA 07 D5 77 FF D0**



至此，提取完成，下面验证下提取的机器码是否正确，如下：

|  |  |  |
| --- | --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11 | #include<stdio.h>   #include<windows.h>   char ourshellcode[]**=**"\x33\xDB\x53\x53\x53\x53\xB8\xEA\x07\xD5\x77\xFF\xD0";   void main()   {    LoadLibirary("user32.dll");    int **\***ret;    ret**=**(int**\***)&ret**+**2;    (**\***ret)**=**(int)ourshellcode;  **return**;   } |  |

运行结果如图：



1. **Shellcode代码编写**

我们自己编写一个调用MessageBox输出“Hello World”的shellcode编码：

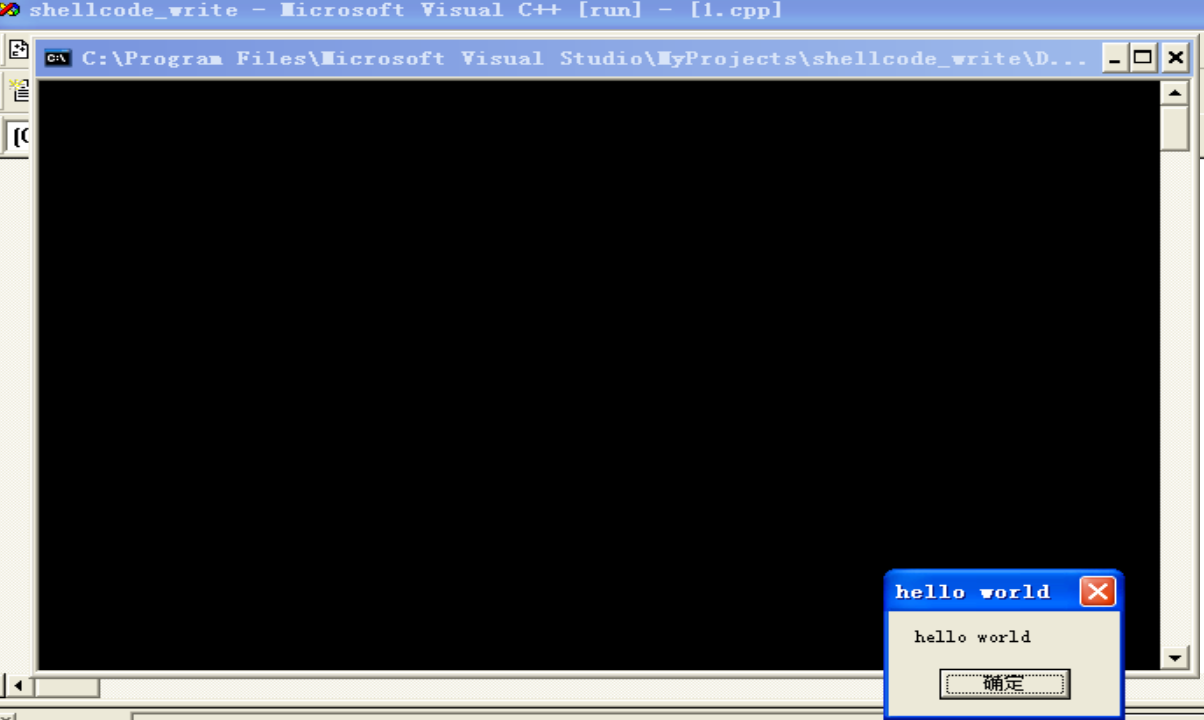
1. 编写c语言的代码：

“Hello world”的ASCII码： \x68\x65\x6C\x6F\x20\x77\x6F\x72\x6C\x64\x20

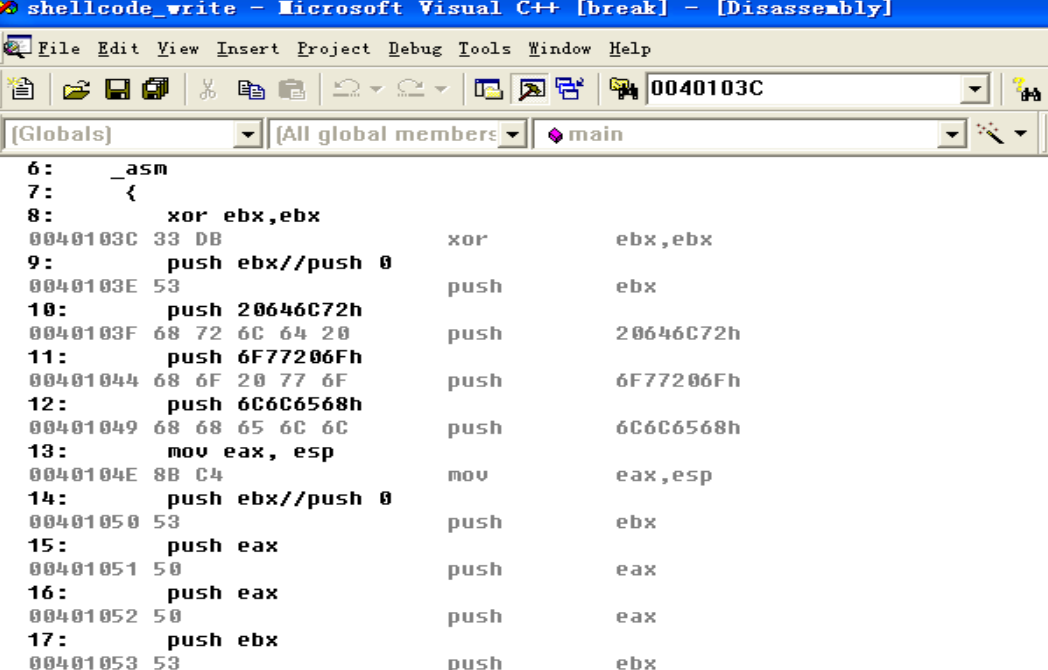
代码如下：

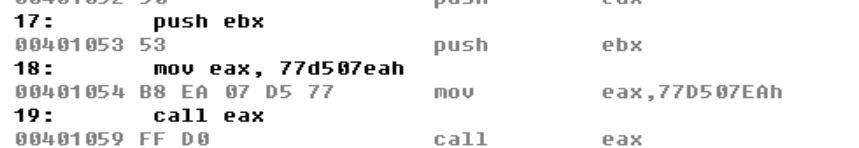
|  |  |  |
| --- | --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22 | #include <stdio.h>  #include <windows.h>  void main()  {  LoadLibrary("user32.dll");  \_asm   {    xor ebx,ebx    push ebx**//**push 0    push 20646C72h    push 6F77206Fh    push 6C6C6568h    mov eax, esp    push ebx**//**push 0    push eax    push eax    push ebx    mov eax, 77d507eah    call eax   }  **return**;   } |  |

运行结果如图：



依然获取机器码：





机器码如下：

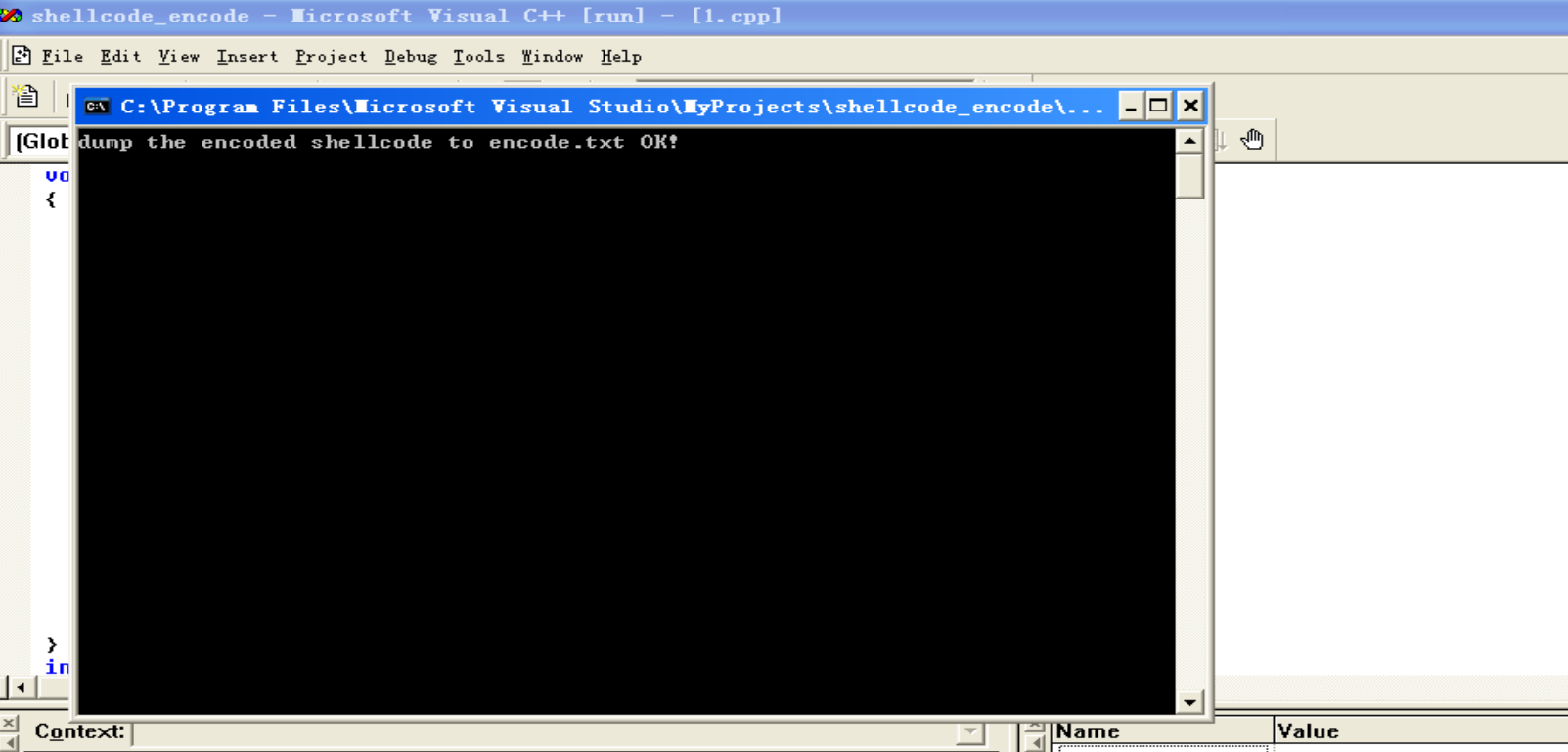
\x33\xDB\x53\x68\x72\x6C\x64\x20\x68\x6F\x20\x77\x6F\x68\x68\x65\x6C\x6C\x8B\xC4\ x53\x50\x50\x53\xB8\xEA\x07\xD5\x77\xBB\xD0

1. **shellcode编码**

利用xor编码实现对shellcode的编码,代码如下：

|  |  |  |
| --- | --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | #include <stdlib.h>  #include <string.h>  #include <stdio.h>  void encoder(char**\*** input, unsigned char key)  {  int i **=** 0, len **=** 0;  FILE **\*** fp;    len **=** strlen(input);    unsigned char **\*** output **=** (unsigned char **\***)malloc(len **+** 1);  **for** (i **=** 0; i<len; i**++**)    output[i] **=** input[i] ^ key;  fp **=** fopen("encode.txt", "w+");  fprintf(fp, "\"");  **for** (i **=** 0; i<len; i**++**)    {    fprintf(fp, "\\x%0.2x", output[i]);  **if** ((i **+** 1) **%** 16 **==** 0)    fprintf(fp, "\"\n\"");    }    fprintf(fp, "\"");    fclose(fp);  printf("dump the encoded shellcode to encode.txt OK!\n");    free(output);   }   int main()   {    char sc[] **=**  "\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";    encoder(sc, 0x44);    getchar();  **return** 0;   } |  |

运行如下：



得到编码如下：

**\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. **shellcode代码的解码：**

|  |  |  |
| --- | --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #include <stdlib.h>  #include <string.h>  #include <stdio.h>  int main()  {    \_\_asm   {    call lable;  lable: pop eax;  add eax, 0x15  xor ecx, ecx    decode\_loop:    mov bl, [eax **+** ecx]    xor bl, 0x44    mov [eax **+** ecx], bl    inc ecx    cmp bl, 0x90  jne decode\_loop  }  **return** 0;  } |  |
|  |  |  |

那如何获取开始位置呢？此处我们通过利用call label实现开始位置的获取：

当我们执行call label时，eip的值被压入栈中，然后我们进行pop操作，这时我们就把eax的值赋值成了当前指令的值了，然后再加上0x15（解码代码的指令长度数），就定位到了我们的shellcode代码的起始位置，问题解决。

解码代码为：**\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**

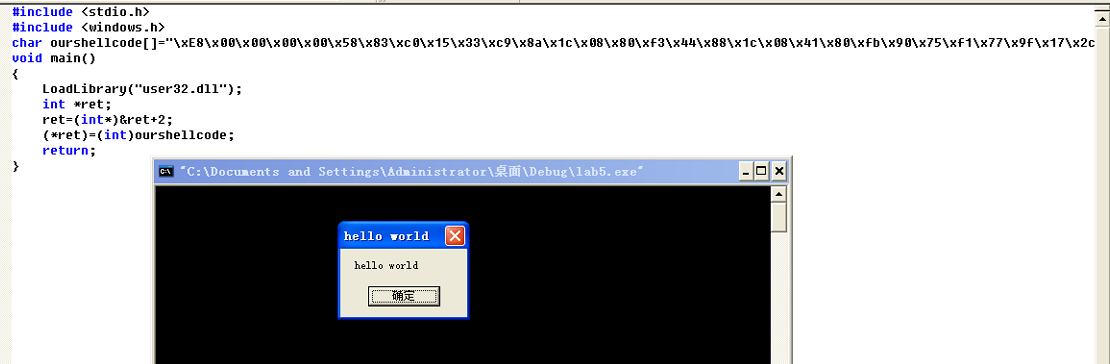
合在一起后有：

**\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\x 2c\x21\x28\x28\xcf\x80\x17\x14\x14\x17\xfc\xae\x43\x91\x33\xbb\x94\xd4**

最后，有：

|  |  |  |
| --- | --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 | #include <stdio.h>   #include <windows.h>   char  ourshellcode[]**=**"\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\x   64\x33\x2b\x2c\x2c\x21\x28\x28\xcf\x80\x17\x14\x14\x17\xfc\xae\x43\x91\x33\xbb\x9   4\xd4";   void main()   {    LoadLibrary("user32.dll");    int **\***ret;    ret**=**(int**\***)&ret**+**2;    (**\***ret)**=**(int)ourshellcode;  **return**;   } |  |

运行如图：

****

**心得体会：**

通过这次实验让我对shellcode的具体注入的全部流程有了更深的认识，不仅仅是要设法制作一个shellcode的字符串，还要设法弄清楚插入位置、实现xor的编码与解码等。