《软件安全》实验报告

姓名： 冯思程 学号：2112213 班级： 1120（张健）

**实验名称：**

Shellcode编码实验

**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. **获得核心shellcode代码**（调用 Messagebox 输出“hello world”的

Shellcode）

首先，要获得到Messagebox函数的入口地址，这里我采用了第二种方法，调用GetProcAddress 函数检索指定的动态链接库（DLL）中的输出库函数地址。代码如下：

#include <windows.h>

#include <stdio.h>

int main()

{

HINSTANCE LibHandle;

FARPROC ProcAdd;

LibHandle = LoadLibrary("user32");

//获取 user32.dll 的地址

printf("user32 = 0x%x \n", LibHandle);

//获取 MessageBoxA 的地址

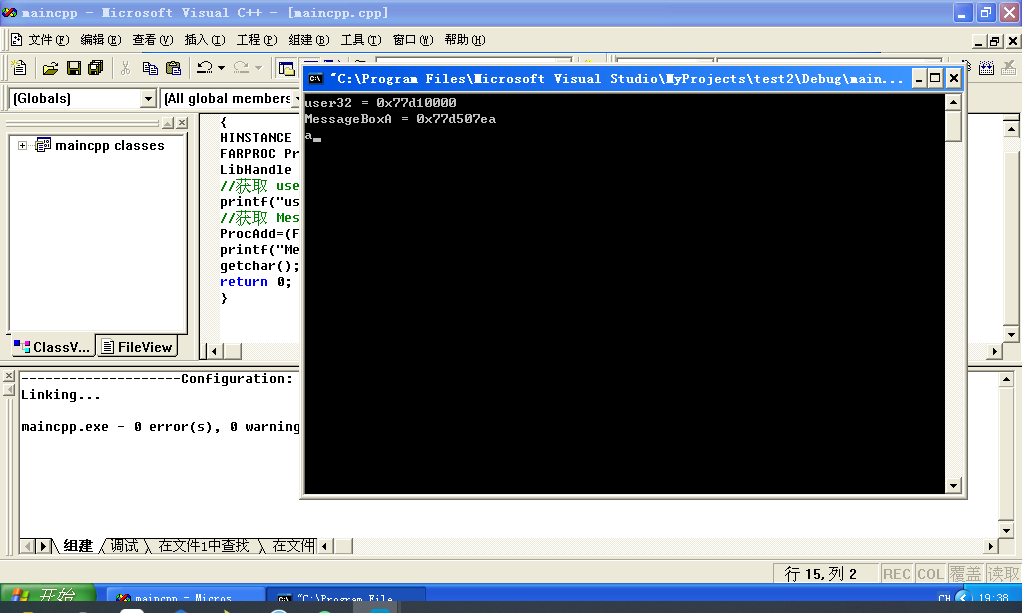
ProcAdd=(FARPROC)GetProcAddress(LibHandle,"MessageBoxA");

printf("MessageBoxA = 0x%x \n", ProcAdd);

getchar();

return 0;

}



得到了Messagebox函数的地址是0x77D507EA。

然后，编写调用 Messagebox 输出“hello world”的Shellcode，这里“hello world”的ASCII代码是\x68\x65\x6C\x6C\x6F\x20\x77\x6F\x72\x6C\x64\x20，利用esp获取字符串的地址，编写代码如下：

#include <stdio.h>

#include <windows.h>

void main()

{

LoadLibrary("user32.dll");//加载 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

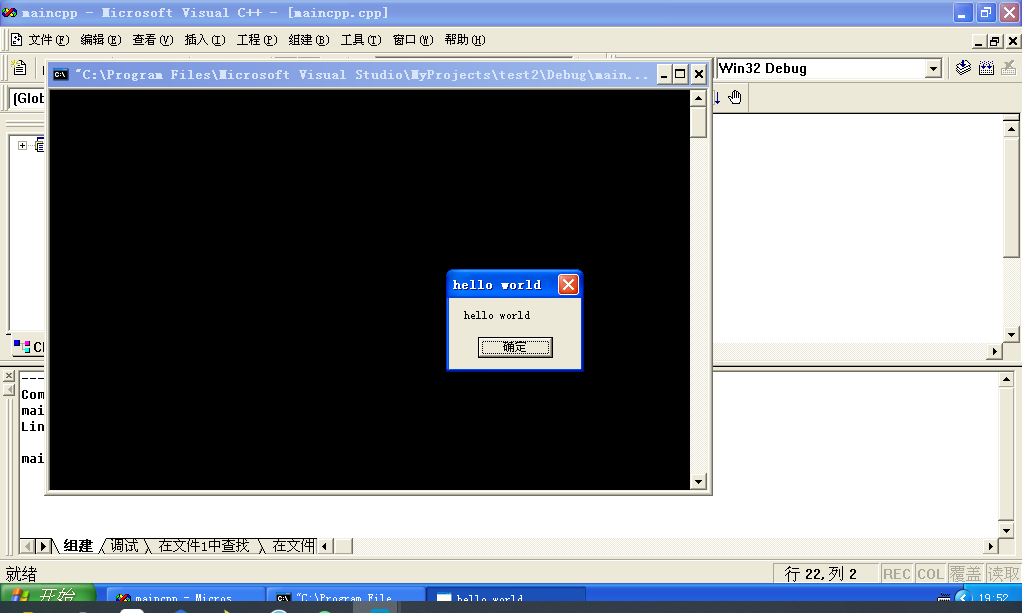
mov eax, 77d507eah// 77d507eah 这个是 MessageBox 函数在系统中的地址，在上文已经获取到。

call eax

}

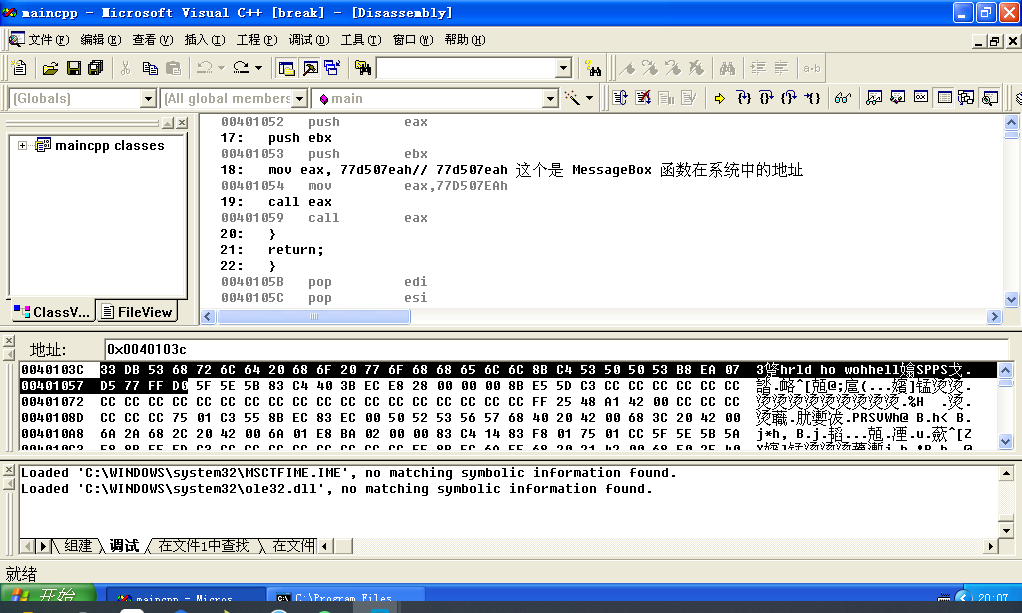
return;

}



然后进入反汇编窗口，找到xor ebx,ebx语句和call eax语句，找到这两个语句对应的地址，第一句地址是0x0040103c，第二句地址是0x00401059，于是上文代码中汇编代码部分对应的机器码是这两个地址之间的机器码，call要向后读两个bytes，于是机器码是从0x0040103c读到0x0040105a，机器码即是（也是获取到的shellcode代码）：

\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  
截图如下：



1. **然后进行对上文获取到的核心shellcode代码进行编码**，使用异或方法进行加密,代码如下：（将上文得到的shellcode输入，结尾要加一个0x90空指令作为结束符。）

#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\x8

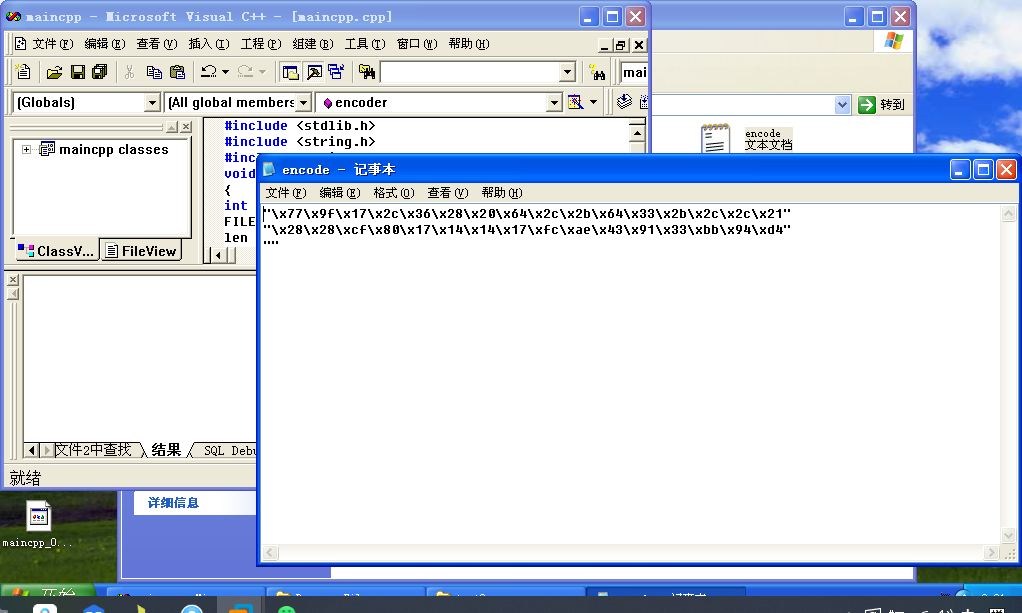
B\xC4\x53\x50\x50\x53\xB8\xEA\x07\xD5\x77\xFF\xD0\x90";

encoder(sc, 0x44);

getchar();

return 0;

}



然后去找到文件位置的名字叫encode的txt文件，打开，里面就是加密后的shellcode，\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. **产生含有解码程序的shellcode**

程序如下：

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

int main()

{

\_\_asm

{

call lable;

lable: pop eax;

add eax, 0x15 ;越过 decoder 记录 shellcode 起始地址

xor ecx, ecx

decode\_loop:mov bl, [eax + ecx]

xor bl, 0x44 ;用 0x44 作为 key

mov [eax + ecx], bl

inc ecx

cmp bl, 0x90 ;末尾放一个 0x90 作为结束符

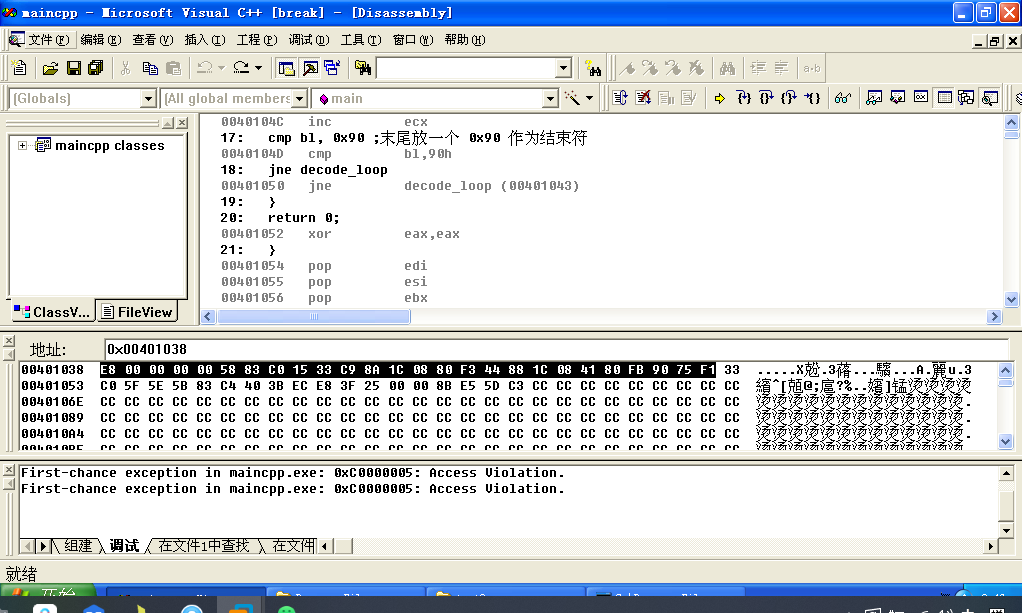
jne decode\_loop

}

return 0;

}

然后根据程序提取出含有解码程序的shellcode，方法类似上文中获取弹出“hello world”窗口的shellcode。对应机器码的指令地址从0x00401038到0x00401051，截图如下：



获得含有解码程序的shellcode是：

\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

**4.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\x1

C\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

利用如下程序进行验证：

#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\x64\x33\x2b\x2c\x2c\x21\x28\x28\xcf\x80\x17\x14\x14\x17\xfc\xae\x43\x91\x33\xbb\x94\xd4";

void main()

{

LoadLibrary("user32.dll");

int \*ret;

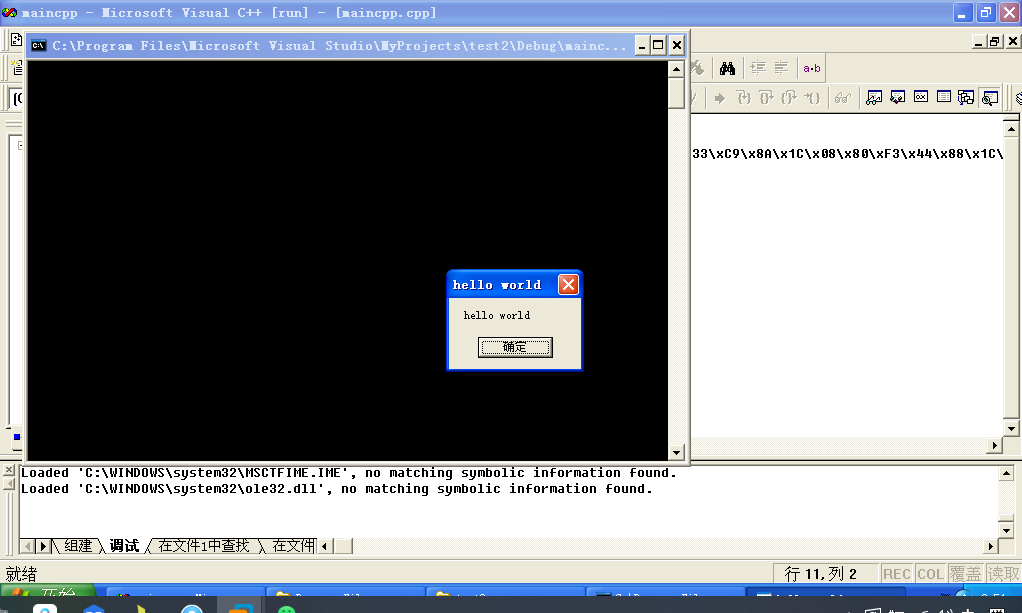
ret=(int\*)&ret+2;

(\*ret)=(int)ourshellcode;

return;

}

结果如下：（成功实现目标功能）



**心得体会：**

1. 对汇编代码有进一步了解，比如在平时不常见到的xor指令
2. 对shellcode代码原理进一步了解，对于生成原理，作用原理有更深刻的认识。
3. 对shell code攻击有了一定的理解，攻击原理和攻击模式有了更深刻的认识。
4. 对加密原理也有了更深刻的认识，同时也增加了利用异或加密的实践经验。
5. 整体原理：
   1. 通过书写一些可执行的代码，转换为对应2进制机器码
   2. 为了避免坏字符（如0x00），同时绕过安全防护，进行加密（可以使用异或加密）
   3. 添加解密代码对应的2进制机器码
   4. 提供入口，例如使用栈覆盖，给返回地址赋新值。