《漏洞利用及渗透测试基础》实验报告

姓名：齐明杰　　学号：2113997　　班级： 信安 2 班

实验名称：

shellcode 代码的提取、编码与解码

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：

一、 Shellcode 提取

我们编写如下代码：

```
#include <stdio.h>
#include <windows.h>
void main(){
    MessageBox(NULL,NULL,NULL,0);
    return;
}
```

下断点，进入反汇编模式，观察到如下的汇编代码：

```
4:        MessageBox(NULL,NULL,NULL,0);
00401028  mov         esi,esp
0040102A  push        0
0040102C  push        0
0040102E  push        0
00401030  push        0
00401032  call        dword ptr [__imp__MessageBoxA@16 (0042428c)]
00401038  cmp         esi,esp
0040103A  call        __chkesp (00401070)
5:        return;
6:    }
0040103F  pop         edi
00401040  pop         esi
00401041  pop         ebx
00401042  add         esp,40h
00401045  cmp         ebp,esp
00401047  call          chkesp (00401070)
```

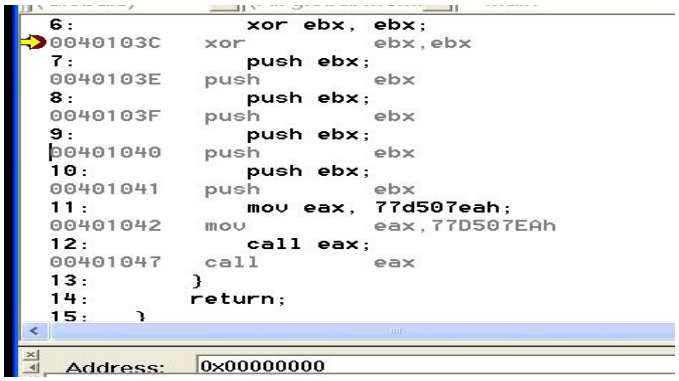
可见，MessageBox 的函数调用包括四个参数的入栈，以及一个 call 调用。

由于 shellcode 中最好不要出现 0x00，因此不能直接写 push 0,我们使用寄存器来保存值，据此编写出如下汇编代码：

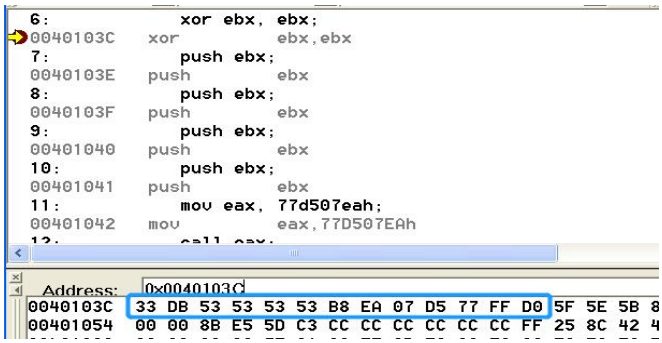
```
#include <stdio.h>
#include <windows.h>
void main(){
    LoadLibrary("user32.dll");
    _asm{
        xor ebx, ebx;
        push ebx;
        push ebx;
        push ebx;
        push ebx;
        mov eax, 77d507eah;
        call eax;
    }
    return;
}
```

运行这段代码，结果与直接调用 MessageBox(NULL,NULL,NULL,0)完全一致，说明这段代码可以实现弹窗的功能。

我们下断点，进入反汇编界面，如下图：



直接在下面内存地址中输入 0x0040103C(即指令所在地址)，即可获取到汇编代码对应的**机器码**：

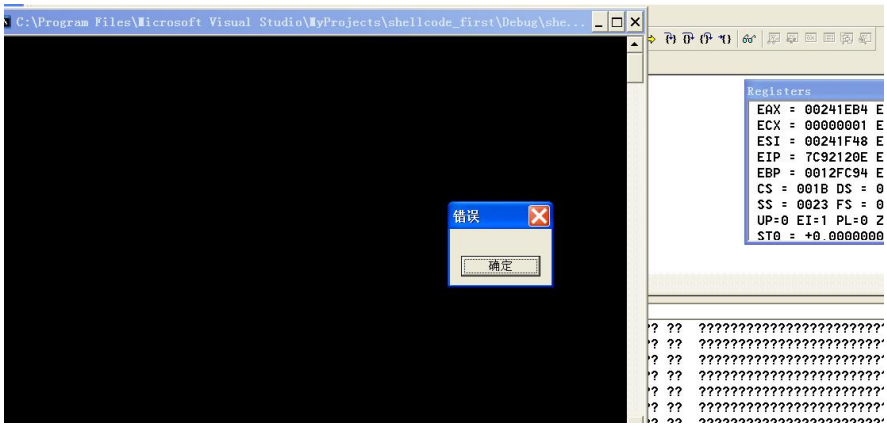


即\x33\xDB\x53\x53\x53\x53\xB8\xEA\x07\xD5\x77\xFF\xD0

我们测试一下上述二进制代码的正确性，将其加入下面的代码中：

```
#include <stdio.h>
#include <windows.h>
char ourshellcode[]="\x33\xDB\x53\x53\x53\x53\xB8\xEA\x07\xD5\x77\xFF\xD0";
void main(){
    LoadLibrary("user32.dll");
    int *ret;
    ret=(int*)&ret+2;
    (*ret)=(int)ourshellcode;
    return;
}
```

运行发现成功弹窗：

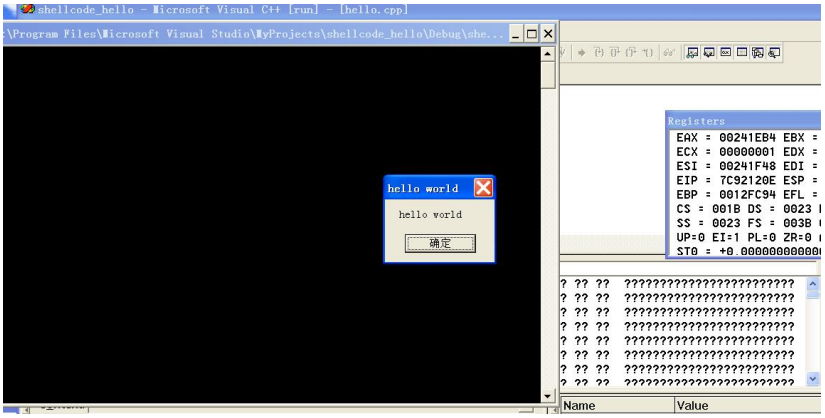


二、 Shellcode 编写

上述我们编写出了弹出一个空信息框的代码，现在我们来编写一个内容和标题均为“hello world”的代码，如下图：

```
#include<stdio.h>
#include<windows.h>
void main(){
    LoadLibrary("user32.dll");
    _asm{
        xor ebx,ebx
        push ebx
        push 20646C72h
        push 6F77206Fh
        push 6C6C6568h
        mov eax,esp
        push ebx
        push eax
        push eax
        push ebx
        mov eax,77d507eah
        call eax
    }
    return;
}
```

其中三个 push 分别对应 hello world 对应的 **ascii 码**(即\x68\x65\x6C\x6C\x6F\x20\x77\x6F\x72\x6C\x64\x20，由于**小端序**反向入栈)，然后使用 mov eax,esp 来保存字符串指针，再根据上述的四个参数入栈，即可完成 hello world 弹窗的实现，最终结果如下图：



提取 shellcode 的方法不再赘述，进入内存区提取出 hello world 代码对应的二进制代码如下：\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\xFF\xD0。

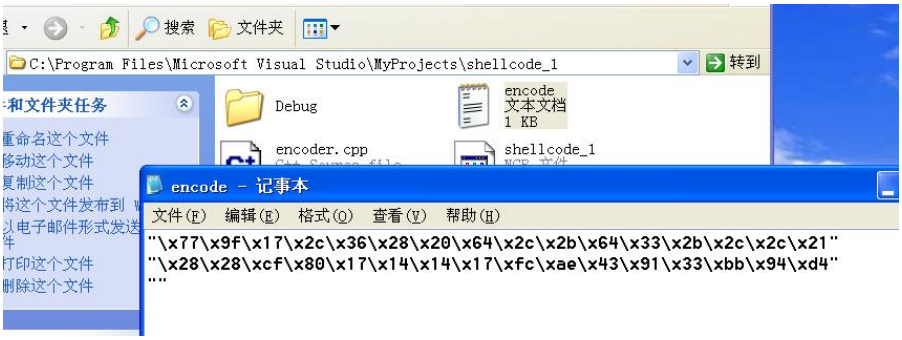
据此，我们完成了 hello world 弹窗机器码的编写提取。

三、 Shellcode 编码

我们编写 shellcode 代码后，需要对其进行编码，以确保其能正确运作，我们对其进行**异或编码**，实现如下代码，对刚刚的机器码进行编码，如下图所示：

```
#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\
    encoder(sc, 0x44);
    getchar();
    return 0;
```

其中，我们使用循环来获取每个字节，然后和 0x44 取异或值，最终写入到”encode.txt”文件中，完成编码，如下图所示：



四、 Shellcode 解码

二进制代码编码后，必然需要运行解码程序来对编码后的机器码进行解码操作，并且这段**解码程序将拼接在待解码程序的前面**。

解码汇编代码如下图所示：

```
#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;
}
```

首先，call lable 等价于 push eip，jmp lable，即 lable 地址入栈，转入 lable 地址处继续执行，而继续执行发现是 pop eax 指令，因此 eax 将保存 lable 的地址，即我们获取到了**当前指令地址**的值。

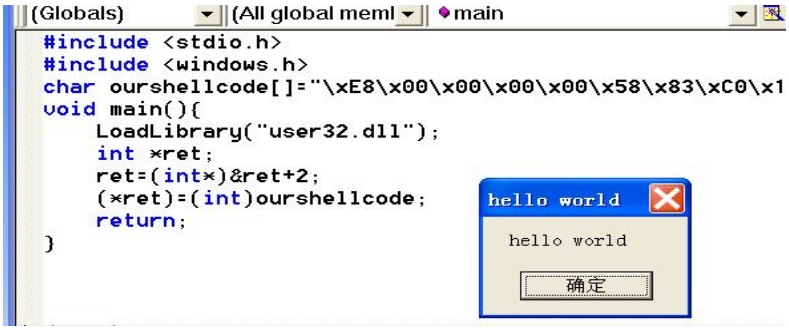
接下来，将 eax 加上 0x15，此时 eax 为待解码的代码所在起始地址。我们通过一个循环来取异或，对代码进行还原，最终解码出原代码。同时，我们在 hello world 的 shellcode 末尾加入了一个 0x90 字节，因此循环结束的标志就是解码后的字节为 0x90。

将这段代码进行 shellcode 的提取，提取出的代码如下：\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

完整 shellcode：\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

最终使用示例 5-4 代码对编码解码代码进行测试，最终成功：

```
#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\xF
void main(){
    LoadLibrary("user32.dll");
    int *ret;
    ret=(int*)&ret+2;
    (*ret)=(int)ourshellcode;
    return;
}
```



据此，shellcode 的提取，编写，编码，解码操作已经全部完成。

心得体会：

通过实验，掌握了 shellcode 代码提取的原理，以及其编码的思想，也掌握了对指令所在地址的获取，知道了如何编写解码程序。

此外，通过本实验，掌握了更多汇编语言的精妙用法，对汇编代码的编写更加熟练。