《漏洞利用及渗透测试基础》实验报告

姓名：蒋薇　　学号：2110957　　班级： 张健老师班

实验名称：

ShellCode 编写及编码

实验要求：

复现第五章实验三，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，将产生的编码后的 shellcode 在示例 5-4 中进行验证

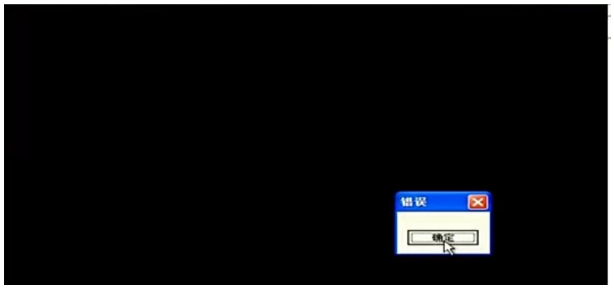
Shellcode 编码是必要的，因为字符集有差异，绕过坏字符，绕过安全防护检测。

一个简单的编码是异或编码，选取不存在的一个机器码、一个字节进行异或，构造带有编码的 Shellcode。

编码程序和打包到 Shellcode 的代码分离，是单独过程。

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

执行弹出一个对话框，“Hello World!”,进行编码，字符 0 等特殊字符异或，



目标：异或 Shellcode 后解码程序能执行，同时能完成解码。

```
void main()
{
    __asm
    {
        add eax, 0x14              ;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
    }
}
```

编码器、主函数，输入要编码对象，encode()函数完成编码，密钥“0x44”，

核心逻辑：output[i] = input[i] ^ key,输出写到文件中

编码后的 Shellcode 不能直接运行，需写解码程序

解码程序：add eax,0x14:越过 decoder 记录 shellcode 起始地址

xor bl,0x44:0x44 作为 key

Cmp bl,0x90 :末尾放一个 0x90 作为结束符

decode_loop 与 ecx 完成循环,结束符 0x90,

eax,怎么得到当前指令地址:可结合汇编指令、pop、栈顶等

```
call lable;
lable:
pop eax;
```

提取 Shellcode,起 0x00401028,止 0x00401042，结合 encode,形成带有解码程序的完整的 Shellcode

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

运用程序框架验证，ourshellcode[] = “...0x90”，

2. shellcode 编码的原理

Shellcode 编码的原理是将原始的二进制代码转换为另一种形式，以避开一些安全程序的检测。编码的方式可以是简单的异或运算，也可以是更复杂的算法，如 Base64 或者 AES 加密。在执行时，需要将编码后的 Shellcode 解码还原为原始二进制代码，然后才能正常执行。这样可以有效地绕过一些安全程序的检测，但也会给攻击者带来额外的工作量和复杂度。

3. shellcode 提取的思想

Shellcode 提取的思想是通过对程序或者系统进行漏洞利用，将自己的二进制代

码注入到目标系统的内存中，并在运行时执行这段代码，从而实现攻击者的目的。在进行漏洞利用时，攻击者需要寻找目标系统的漏洞，并利用这些漏洞来实现代码注入。一般来说，攻击者会将 Shellcode 编写成一个小型的程序，然后将它插入到目标系统的内存中，并在系统运行时执行该程序来达到攻击的目的。为了提取 Shellcode，可以使用一些工具，如反汇编器、调试器、内存分析工具等，来分析目标系统的内存中的代码，并提取出 Shellcode 的二进制代码。同时，攻击者也可以利用一些技巧来隐藏 Shellcode，如使用加密、混淆等技术来使 Shellcode 难以被检测和分析。

.txt 得到机器码，“\x”文本空格替换，

演 示 截 图 ：

ss

心得体会：

通过实验， C、汇编结合加快运行性能；

eax,怎么得到当前指令地址:可结合汇编指令、pop、栈顶等

```
call lable;
lable:
pop eax;
```