

# 《软件安全》实验报告

姓名：辛浩然      学号：2112514      班级：信息安全、法学

实验名称：

**Shellcode 编写及编码**

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 编码的原理和 Shellcode 提取的思想。

实验过程：

## 1.编写 Shellcode

首先，编写调用 MessageBox 输出“hello world”的 Shellcode.

汇编代码如下：

```
#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
        // 4 字节存入，硬编码空格是 0x20。不足 4 字节，可以在最后的字节里补空格。“hello world”对应的 ASCII 码为\x68\x65\x6C\x6C\x6F\x20\x77\x6F\x72\x6C\x64\x20.但是入栈的话，需要倒着来；考虑 bigendian 编码，存储顺序也得倒过来。
        mov eax, esp
        // 利用 ESP 来获取字符串的地址
        push ebx // push 0
        push eax // 字符串地址压入，作为参数
        push eax // 字符串地址压入，作为参数
        push ebx
        mov eax, 77d507eah // 77d507eah 这个是 MessageBox 函数在系统中的地址
        call eax
    }
}
```

```

    return;
}

```

根据汇编代码，找到对应地址中的机器码

8:	xor ebx,ebx		
0040103C	33 DB	xor	ebx,ebx
9:	push ebx//push 0		
0040103E	53	push	ebx
10:	push 20646C72h		
0040103F	68 72 6C 64 20	push	20646C72h
11:	push 6F77206Fh		
00401044	68 6F 20 77 6F	push	6F77206Fh
12:	push 6C6C6568h		
00401049	68 68 65 6C 6C	push	6C6C6568h
13:	mov eax, esp		
0040104E	8B C4	mov	eax,esp
14:	push ebx//push 0		
00401050	53	push	ebx
15:	push eax		
00401051	50	push	eax
16:	push eax		

提取到的 Shellcode 代码为：

```

\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

```

## 2. 编码

使用异或编码的方法，输出异或后的 Shellcode 编码。程序通过调用 encoder 函数将传入的 Shellcode（即上步得到的 Shellcode，值得注意的是，在上步得到的 Shellcode 后加了空指令 0x90，作为结束符）与 0x44 进行 XOR 运算，并将编码后的结果写入输出文件 encode.txt 中。

代码如下：

```

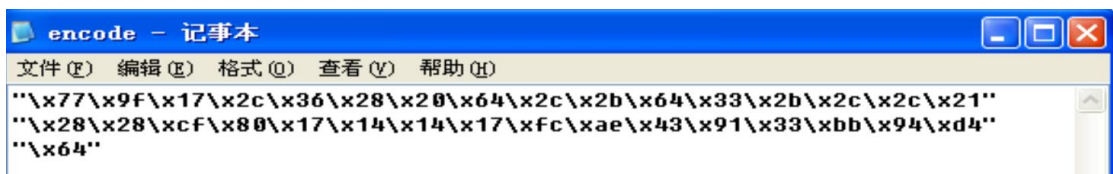
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
void encoder(char *input, unsigned char key)
{
    int i = 0, len = 0;
    FILE *fp;
    len = strlen(input);
    unsigned char *output = (unsigned char *)malloc(len + 1);
    for (i = 0; i < len; i++)
        output[i] = input[i] ^ key;
    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\\n");
    }
    fprintf(fp, "\\n");
    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\\x6
        8\\x65\\x6C\\x6C\\x8B\\xC4\\x53\\x50\\x50\\x53\\xB8\\xEA\\x07\\xD5\\x77\\xFF\\xD0\\x9
        0 ";
    encoder(sc, 0x44);
    getchar();
    return 0;
}

```

得到 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"

```

### 3. 解码

使用下面的程序产生含有解码程序的 Shellcode:

```

#include <stdlib.h>
#include <string.h>
#include <stdio.h>
int main()
{
    __asm
    {
        call lable;
        lable:
        pop eax;
    }
}

```

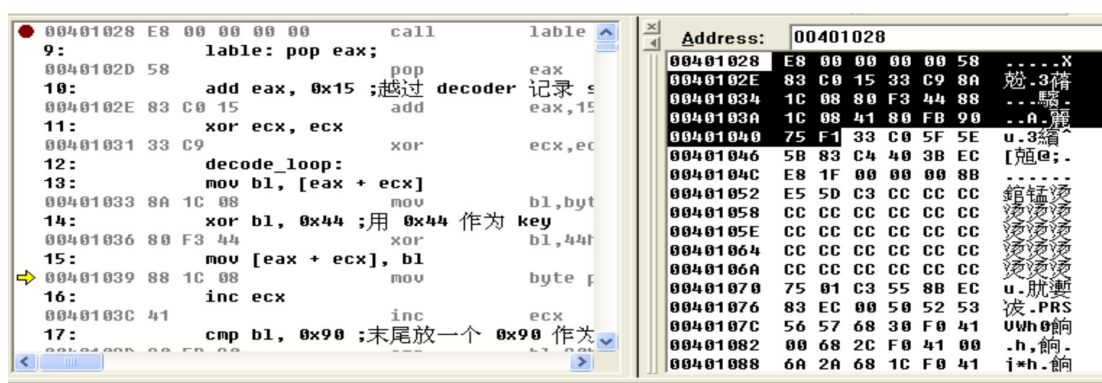
```

    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;
}

```

“call lable; lable: pop eax;”之后，eax 的值就是当前指令地址了。原因是 call lable 的时候，会将当前 EIP 的值（也就是下一条指令 pop eax 的指令地址）入栈。将 eax 加 0x15 到达 Shellcode 开始的位置。之后的代码将每次将 Shellcode 的代码异或 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。

提取机器码：



提取得机器码为：

\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

#### 4.得到 Shellcode

链接两段机器码，得到完整 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

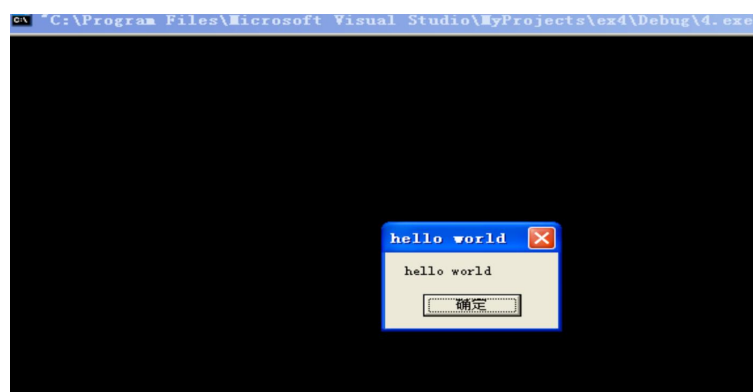
利用该 Shellcode 实现漏洞的利用，如下是一个测试程序：

```

#include <stdio.h>
#include <windows.h>
char ourShellcode[] =
"\xE8\x00\x00\x00\x00\x58\x83\xC0\x15\x33\xC9\x8A\x1C\x08\x80\xF3\x4
4\x88\x1C\x08\x41\x80\FB\x90\x75\xF1\x77\x9f\x17\x2c\x36\x28\x20\x6
4\x2c\x2b\x64\x33\x2b\x2c\x2c\x21\x28\x28\xcf\x80\x17\x14\x14\x17\xfc
c\xae\x43\x91\x33\xbb\x94\xd4";
void main()
{
    LoadLibrary("user32.dll");
    int *ret;
    ret = (int *)&ret + 2;
    (*ret) = (int)ourShellcode;
    return;
}

```

弹出 helloworld 窗口，可以验证 Shellcode 的正确性。



## 5. 总结

### (1) Shellcode 编码的原理

Shellcode 编码的原理是将原始的 Shellcode 转换为一种新的表示形式，以实现混淆和隐藏。这通常涉及将 Shellcode 转换为 ASCII 字符集中的一组字符，并通过各种技术，如简单的替换、移位、异或和加密等来进行编码。这些技术可以使 Shellcode 变得更加难以检测和防御。

在编码后，精心构造精简干练的解码程序，放在 Shellcode 开始执行的地方。当 exploit 成功时，Shellcode 顶端的解码程序首先运行，它会在内存中将真正的 Shellcode 还原成原来的样子，然后执行。

### (2) Shellcode 提取的思想

首先，用 c 语言书写要执行的 Shellcode;

其次，利用调试功能，找到其对应的汇编代码;

最后，根据汇编代码，找到对应地址中的机器码，就得到了 Shellcode.

**心得体会：**

通过本次实验，掌握了漏洞利用的核心思想、shellcode 的概念，理解 shellcode 的编写过程，掌握 shellcode 编码解码技术。