《漏洞利用及渗透测试基础》实验报告

姓名：平世龙　学号：2012656　班级：1074

1　实验名称

Shellcode 编写及编码

2　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。

3　实验过程

1. 在过去的实验中，我们已经通过反汇编工具得到了调用 MessageBoxA 函数的 shellcode, 然后我们通过下面的程序输入 shellcode 代码后，输出异或后的 shellcode 编码。

```
#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);
```

```
22      printf("dump the encoded shellcode to encode.txt OK!\n");
23      free(output);
24  }
25  int main()
26  {
27  char sc[] = "\x33\xDB\x53\x68\x72\x6C\x64\x20\x68\x6F\x20\x77
28  \x6F\x68\x68\x65\x6C\x6C\x8B\xC4\x53\x50\x50\x53\xB8\xEA\x07
29  \xD5\x77\xFF\xD0\x90";
30      encoder(sc, 0x44);
31      getchar();
32      return 0;
```

程序读入 shellcode 的各个字符并与指定的 key 进行异或操作，并将编码后的 shellcode 以十六进制格式写到指定文件中，每编码 16 个字符进行一次换行。

程序运行后结果如下所示：

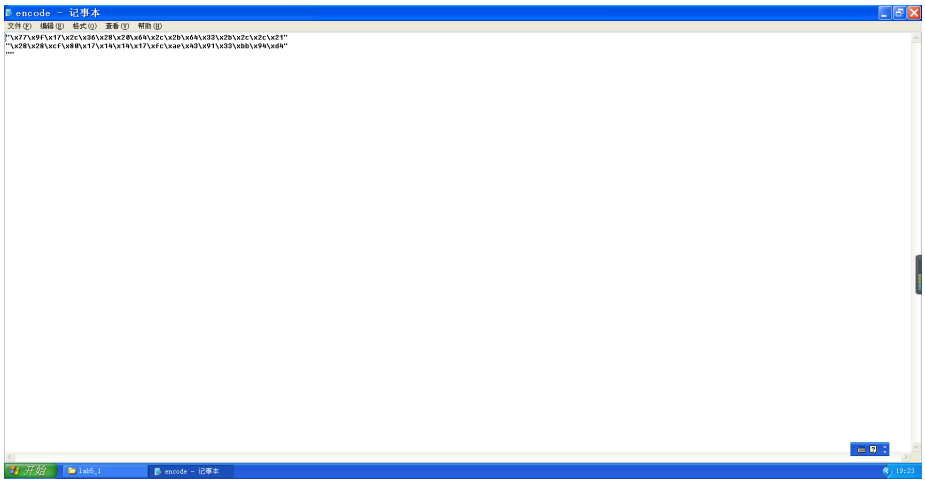

图 1: “hello world” 的 Shellcode 的编码

2. 编写解码代码，所生成的解码器会与编码后的 shellcode 联合执行。下面的程序中，默认 eax 在 shellcode 开始时对准 shellcode 起始位置，之后的代码将每次将 shellcode 的代码异或特定 key（下例为 0x44）后重新覆盖原先 shellcode 的代码，并在末尾放一个空指令 0x90 作为结束符。

```
1  #include <stdlib.h>
2  #include <string.h>
3  #include <stdio.h>
```

```
4  void main()
5  {
6      __asm
7      {
8          add eax, 0x14 ;越过 decoder 记录 shellcode 起始地址
9          xor ecx, ecx
10         decode_loop:
11         mov bl, [eax + ecx]
12         xor bl, 0x44 ;用 0x44 作为 key
13         mov [eax + ecx], bl
14         inc ecx
15         cmp bl, 0x90 ;末尾放一个 0x90 作为结束符
16         jne decode_loop
17   }
18 }
```

此程序机器码长 0x14 个字节，为了保证 [eax+0x14] 为 shellcode 数据段，应该保证 eax 中存储第一句汇编语句的地址，修改代码如下:

```
1  #include <stdlib.h>
2  #include <string.h>
3  #include <stdio.h>
4  int main()
5  {
6      __asm
7      {
8          call lable;
9          lable:
10             pop eax;
11             add eax, 0x15 ;
12             xor ecx, ecx
13         decode_loop:
14             mov bl, [eax + ecx]
15             xor bl, 0x44 ;
16             mov [eax + ecx], bl
17             inc ecx
18             cmp bl, 0x90 ;
19             jne decode_loop
20     }
21     return 0;
22 }
```

此程序在执行 call 语句时会将当前 EIP 的值（下一条语句地址）压入栈，在执行下一条语句时又将取出入栈的地址，因此动态地获取了 add 语句上一条语句的地址并保存在 eax 中，由于此条语句的机器码长度为 1 字

节，所以需要将偏移调整为 0x15。

3. 运行上述代码，进入反汇编界面，获取此代码机器码如图所示：

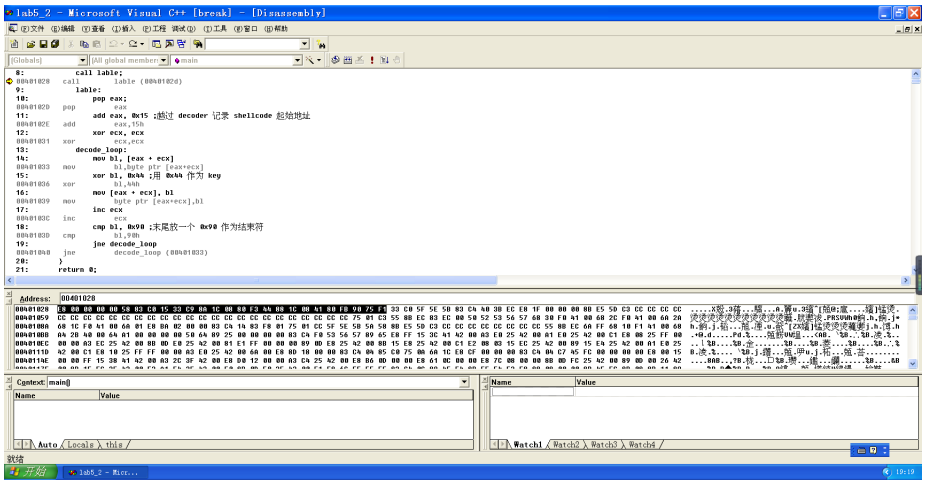

图 2: 产生含有解码程序的 Shellcode

我们将之前实验得到的调用 Messagebox 输出“hello world”的 Shellcode 的编码与此机器码链接得到完整 shellcode 如下：”\xE8\x00\x00\x00\x00\x58\x83\xC0\x15\x33 \xC9 \x8A\x1C\x08\x80\xF3\x44\x88\x1C\x08\x41 \x80\xFB\x90\x75\xF1\x77\x9f\x17\x2c \x36\x28 \x20\x64\x2c\x2b\x64x33\x2b\x2c\x2c\x21 \x28\x28\xcf\x80\x17\x14\x14\x17\xfc \xae\x43\x91\x33\xbb\x94\xd4”。

4. 使用示例 5-4 验证代码如下

```
#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：

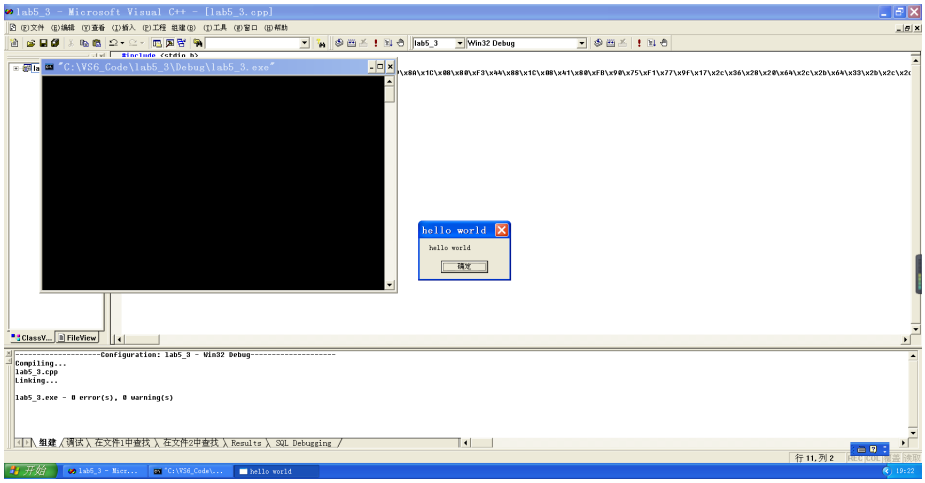

图 3: 验证结果

4 心得体会

shellcode 代码编制过程由于字符集的差异、绕过坏字符、绕过安全防护检测原因需要进行编码，通过实验，动手实现了 Shellcode 编码，了解 shellcode 攻击、shellcode 编码方法及思想，学习了如何将解码程序插入到 shellcode 中。