

南開大學

汇编语言与逆向技术课程实验报告

实验九： re_exercises_advanced



学 院 网络空间安全学院
专 业 信息安全
学 号 2211044
姓 名 陆皓喆
班 级 信息安全

一、实验目的

- 1、进一步熟悉静态反汇编工具 IDA Freeware;
- 2、熟悉将反汇编代码进行反编译的过程;
- 3、掌握对于反编译伪代码的逆向分析;
- 4、运用熟悉的编程语言, 实现简单的脚本编写。

二、实验原理

(一) task3

- 1、通过 IDA Freeware 得到 task3.exe 的反汇编代码, 如图 1 和图 2 所示。

```
.text:00402A70      sub     esp, 0A4h
.text:00402A76      mov     eax, __security_cookie
.text:00402A7B      xor     eax, esp
.text:00402A7D      mov     [esp+0A4h+var_4], eax
.text:00402A84      mov     [esp+0A4h+var_80], 42h ; 'B'
.text:00402A89      xor     ecx, ecx
.text:00402A8B      mov     [esp+0A4h+var_7F], 7Eh ; '~'
.text:00402A90      mov     [esp+0A4h+var_7E], 77h ; 'w'
.text:00402A95      mov     [esp+0A4h+var_7D], 73h ; 's'
.text:00402A9A      mov     [esp+0A4h+var_7C], 61h ; 'a'
.text:00402A9F      mov     [esp+0A4h+var_7B], 77h ; 'w'
.text:00402AA4      mov     [esp+0A4h+var_7A], 32h ; '2'
.text:00402AA9      mov     [esp+0A4h+var_79], 7Bh ; '{'
.text:00402AAE      mov     [esp+0A4h+var_78], 7Ch ; '|'
.text:00402AB3      mov     [esp+0A4h+var_77], 62h ; 'b'
.text:00402AB8      mov     [esp+0A4h+var_76], 67h ; 'g'
.text:00402ABD      mov     [esp+0A4h+var_75], 66h ; 'f'
.text:00402AC2      mov     [esp+0A4h+var_74], 32h ; '2'
.text:00402AC7      mov     [esp+0A4h+var_73], 73h ; 's'
.text:00402ACC      mov     [esp+0A4h+var_72], 32h ; '2'
.text:00402AD1      mov     [esp+0A4h+var_71], 61h ; 'a'
.text:00402AD6      mov     [esp+0A4h+var_70], 66h ; 'f'
.text:00402ADB      mov     [esp+0A4h+var_6F], 60h ; '`'
.text:00402AE0      mov     [esp+0A4h+var_6E], 7Bh ; '{'
.text:00402AE5      mov     [esp+0A4h+var_6D], 7Ch ; '|'
.text:00402AEA      mov     [esp+0A4h+var_6C], 75h ; 'u'
.text:00402AEF      mov     [esp+0A4h+var_6B], 28h ; '('
.text:00402AF4      mov     [esp+0A4h+var_6A], 18h
.text:00402AF9      mov     [esp+0A4h+var_69], 12h
.text:00402AFE      xchg    ax, ax
```

图 1 task3.exe 的反汇编代码

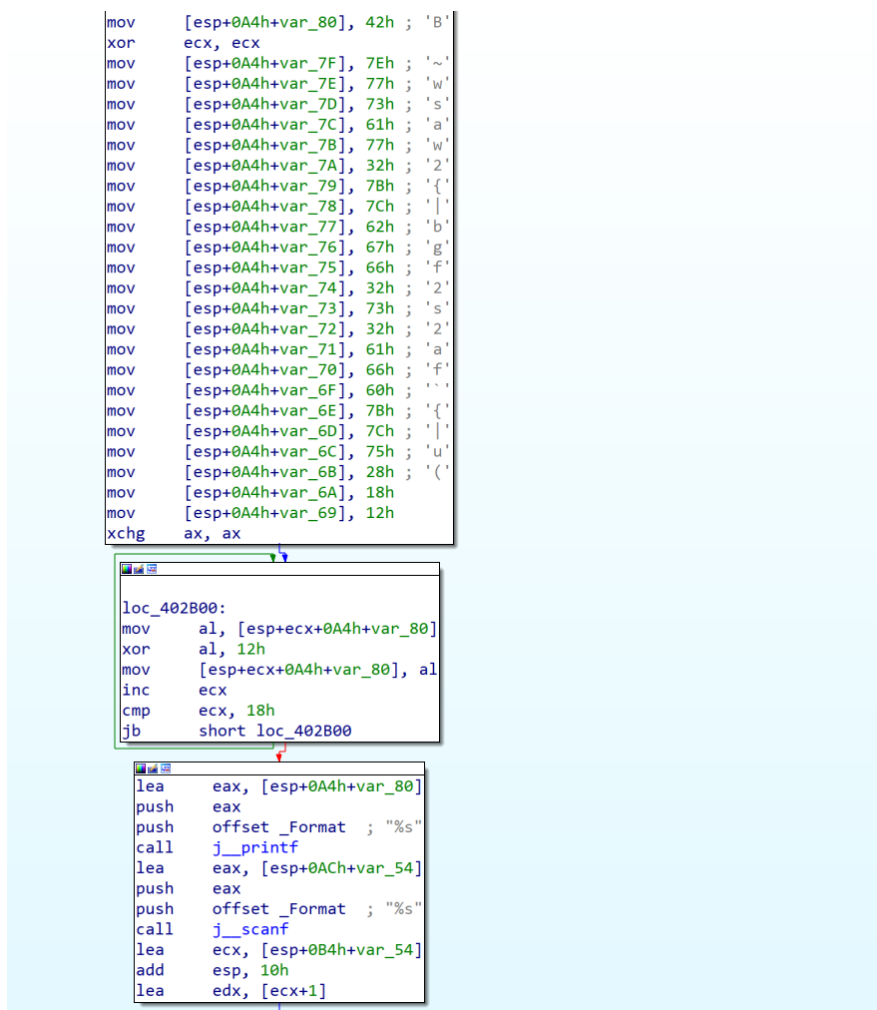


图 2 task3.exe 反汇编代码的图形化显示

2、使用 IDA 的反编译功能（F5 快捷键）得到伪代码，如图 3 所示。

```

1 int __cdecl main()
2 {
3     unsigned int v0; // ecx
4     unsigned int v1; // ecx
5     const char *v2; // eax
6     int v3; // edx
7     unsigned int v4; // ecx
8     unsigned int v6; // ecx
9     char v7[8]; // [esp+30Ch] [ebp-A4h] BYREF
10    char v8; // [esp+314h] [ebp-9Ch] BYREF
11    char v9[8]; // [esp+315h] [ebp-9Bh] BYREF
12    char v10; // [esp+320h] [ebp-90h] BYREF
13    char v11[12]; // [esp+321h] [ebp-8Fh] BYREF
14    char v12[24]; // [esp+330h] [ebp-80h] BYREF
15    char v13[20]; // [esp+348h] [ebp-68h] BYREF
16    char v14[80]; // [esp+35Ch] [ebp-54h] BYREF
17
18    qmemcpy(v12, "B~wsaw2{ |bgf2s2af`{ |u(", 22);
19    v0 = 0;
20    v12[22] = 24;
21    v12[23] = 18;
22    do
23    {
24        v12[v0++] ^= 0x12u;
25        while ( v0 < 0x18 );
26        j__printf("%s", v12);
27        j__scanf("%s", v14);
28        if ( strlen(v14) == 20 )
29        {
30            v13[0] = -15;
31            v3 = 0;
32            v13[1] = -55;
33            v13[2] = -31;
34            v13[3] = -1;
35            v13[4] = -25;

```

图 3 task3.exe 的反编译伪代码

3、通过对反汇编命令及反编译伪代码的分析，逆向推理出待输入字符串的计算公式；

4、使用熟悉的编程语言（C++、Java、Python 等）对待输入字符串进行计算，完成逆向分析挑战。

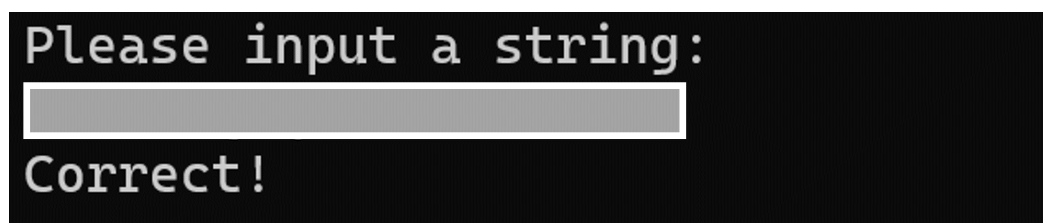


图 4 逆向分析，完成 task3 练习

(二) task4

1、通过 IDA Freeware 得到 task4.exe 的反汇编代码，如图 5 和图 6 所示。

```
.text:00401470 _main          proc near          ; CODE XREF: _main_0↑j
.text:00401470          = byte ptr -12Ch
.text:00401470 input        = byte ptr -0D8h
.text:00401470 target      = dword ptr -6Ch
.text:00401470 var_6C       = dword ptr -68h
.text:00401470 var_68       = dword ptr -64h
.text:00401470 var_64       = dword ptr -60h
.text:00401470 var_60       = dword ptr -5Ch
.text:00401470 var_5C       = word ptr -58h
.text:00401470 var_58       = byte ptr -54h
.text:00401470 var_54       = dword ptr -4
.text:00401470
v.text:00401470          sub     esp, 6Ch
.text:00401473          mov     eax, __security_cookie
.text:00401478          xor     eax, esp
.text:0040147A          mov     [esp+6Ch+var_4], eax
.text:0040147E          push   offset _Format ; "Please input a string:\n"
.text:00401483          call  j__printf
.text:00401488          lea     eax, [esp+70h+var_54]
.text:0040148C          push   eax
.text:0040148D          push   offset aS        ; "%s"
.text:00401492          call  j__scanf
.text:00401497          lea     ecx, [esp+78h+var_54]
.text:0040149B          add     esp, 0Ch
.text:0040149E          lea     edx, [ecx+1]
.text:004014A1
.text:004014A1 loc_4014A1:          ; CODE XREF: _main+36↓j
.text:004014A1          mov     al, [ecx]
.text:004014A3          inc     ecx
.text:004014A4          test    al, al
.text:004014A6          jnz     short loc_4014A1
```

图 5 task4.exe 的反汇编代码

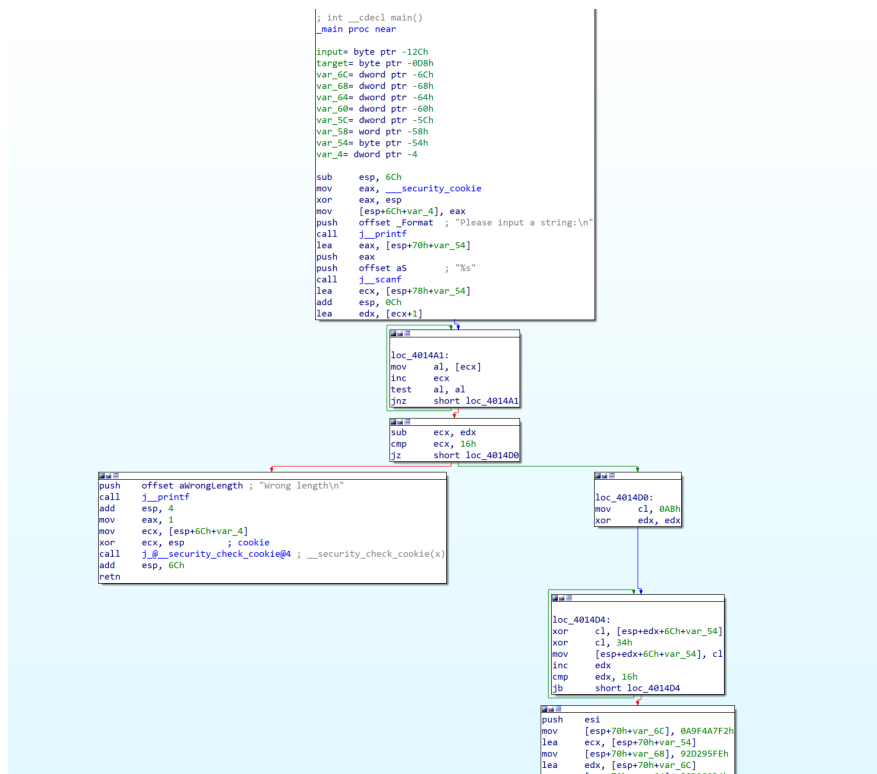


图 6 task4.exe 反汇编代码的图形化显示

2、使用 IDA 的反编译功能（F5 快捷键）得到伪代码，如图 7 所示。右键点击数字对象可实现数制转换。

```

3 char v1; // cl
4 unsigned int i; // edx
5 char *v3; // ecx
6 int *v4; // edx
7 unsigned int v5; // esi
8 bool v6; // cf
9 int v7[5]; // [esp+C0h] [ebp-6Ch] BYREF
10 __int16 v8; // [esp+D4h] [ebp-58h]
11 char v9[80]; // [esp+D8h] [ebp-54h] BYREF
12
13 j_printf("Please input a string:\n");
14 j_scanf("%s", v9);
15 if ( strlen(v9) == 22 )
16 {
17     v1 = 0xAB;
18     for ( i = 0; i < 0x16; ++i )
19     {
20         v1 ^= v9[i] ^ 0x34;
21         v9[i] = v1;
22     }
23     v7[0] = 0xA9F4A7F2;
24     v3 = v9;
25     v7[1] = 0x92D295FE;
26     v4 = v7;
27     v7[2] = 0x80D389D4;
28     v5 = 0x12;
29     v7[3] = 0xB5E0BCEB;
30     v7[4] = 0xBEE4B5ED;
31     v8 = 0xBCED;
32     while ( *(_DWORD *)v3 == *v4 )
33     {
34         v3 += 4;
35         ++v4;
36         v6 = v5 < 4;
37         v5 -= 4;
38         if ( v6 )
39         {
40             if ( *(_WORD *)v3 == *(_WORD *)v4 )
41             {
42                 j_printf("Correct");
43                 return 0;
44             }
45         }
46     }
47 }

```

图 7 task4.exe 的反编译伪代码

3、通过对反汇编命令及反编译伪代码的分析，逆向推理出待输入字符串的计算公式

4、使用熟悉的编程语言（C++、Java、Python 等）对待输入字符串进行计算，完成逆向分析挑战。

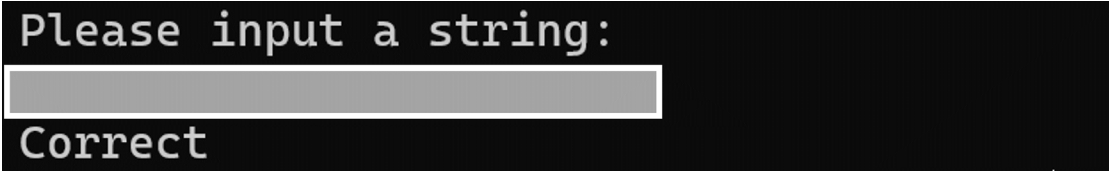


图 8 逆向分析，完成 task4 练习

三、实验报告

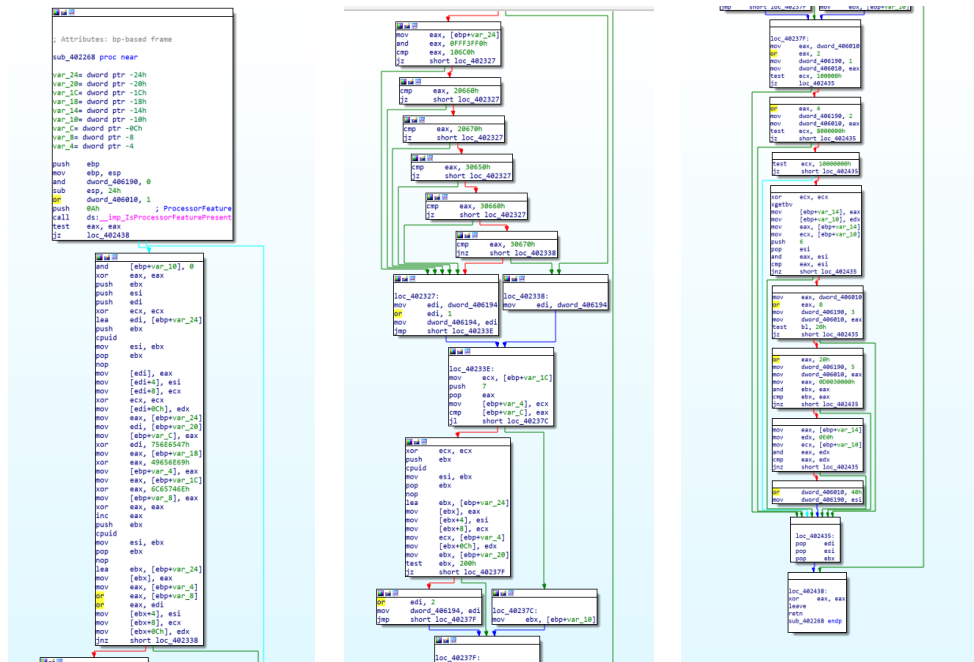
3.1 分别针对 task3、task4 使用 IDA Freeware，获得可执行文件的反汇编代码及反编译伪代码，提供截图。

3.1.1 task3

反汇编代码：

```
.text:00401075      or      dword_406010, 1
.text:0040227C      push   0Ah          ; ProcessorFeature
.text:0040227E      call   ds:__imp_IsProcessorFeaturePresent
.text:00402284      test   eax, eax
.text:00402286      jz     loc_402438
.text:0040228C      and     [ebp+var_10], 0
.text:00402298      xor     eax, eax
.text:00402292      push   ebx
.text:00402293      push   esi
.text:00402294      push   edi
.text:00402295      xor     ecx, ecx
.text:00402297      lea     edi, [ebp+var_24]
.text:0040229A      push   ebx
.text:00402298      cpuid
.text:0040229D      mov     esi, ebx
.text:0040229F      pop     ebx
.text:004022A0      nop
.text:004022A1      mov     [edi], eax
.text:004022A3      mov     [edi+4], esi
.text:004022A6      mov     [edi+8], ecx
.text:004022A9      xor     ecx, ecx
.text:004022AB      mov     [edi+0Ch], edx
.text:004022AE      mov     eax, [ebp+var_24]
.text:004022B1      mov     edi, [ebp+var_20]
.text:004022B4      mov     [ebp+var_C], eax
.text:004022B7      xor     edi, 756E6547h
.text:004022BD      mov     eax, [ebp+var_18]
.text:004022C0      xor     eax, 49656E69h
.text:004022C5      mov     [ebp+var_4], eax
.text:004022C8      mov     eax, [ebp+var_1C]
.text:004022CB      xor     eax, 6C65746Eh
.text:004022D0      mov     [ebp+var_8], eax
.text:004022D3      xor     eax, eax
.text:004022D5      inc     eax
.text:004022D6      push   ebx
.text:004022D7      cpuid
.text:004022D9      mov     esi, ebx
.text:004022DB      pop     ebx
.text:004022DC      nop
00001675 00402275: sub_402268+D (Synchronized with Hex View-1)
```

反汇编代码图形化显示：



反编译伪代码:

```

1 int __cdecl main_0(int argc, const char **argv, const char **envp)
2 {
3     unsigned int v1; // ecx
4     unsigned int v4; // ecx
5     char v5; // eax
6     int v6; // eax
7     unsigned int v7; // ecx
8     unsigned int v9; // ecx
9     char v10[8]; // [esp+0h] [ebp+40h] BYREF
10    char v11; // [esp+8h] [ebp+44h] BYREF
11    char v12[8]; // [esp+10h] [ebp+48h] BYREF
12    char v13; // [esp+14h] [ebp+4Ch] BYREF
13    char v14[12]; // [esp+18h] [ebp+50h] BYREF
14    char ArgList[24]; // [esp+1Ch] [ebp+54h] BYREF
15    char v16[20]; // [esp+20h] [ebp+58h] BYREF
16    char ArgList[80]; // [esp+24h] [ebp+5Ch] BYREF
17
18    qmemcpy(ArgList, "B-usaw2(b?7z2zf{lu", 22);
19    v1 = 0;
20    ArgList[22] = 24;
21    ArgList[23] = 15;
22    do
23    {
24        v14[v1++] = 0x2u;
25        while (v5 < 0x18);
26        sub_4011E5("%s", (char*)ArgList);
27        sub_4011E5("%s", (char*)ArgList);
28        if (strlen(ArgList) == 20)
29        {
30            v16[0] = -15;
31            v6 = 0;
32            v16[1] = -55;
33            v16[2] = -31;
34            v16[3] = -1;
35            v16[4] = -25;
36            v16[5] = -109;
37            v16[6] = -12;
38            v16[7] = -17;
39            v16[8] = -46;
40            v16[9] = -24;
41            v16[10] = -17;
42            v16[11] = -64;
43            v16[12] = -50;
44            v16[13] = -4;
45            v16[14] = -30;
46            v16[15] = -47;
47            v16[16] = -3;
48            v16[17] = -64;
49            v16[18] = -15;
50            v16[19] = -4;
51            while ( (ArgList[v1] * 0xAS) == (unsigned __int8)v16[v1] )
52            {
53                if (v1 >= 20)
54                {
55                    v1 = 20;
56                    v7 = 0;
57                    qmemcpy(v10, "8X24nW", sizeof(v10));
58                    do
59                    {
60                        v12[v1++] = 0x57u;
61                        while (v7 < 9);
62                        return 0;
63                    }
64                    v16[0] = -57;
65                    v9 = 0;
66                    v16[1] = -30;
67                    v16[2] = -31;
68                    v16[3] = -2;
69                    v16[4] = -9;
70                    v16[5] = -79;
71                    v16[6] = -112;
72                    do
73                    {
74                        v16[v1++] = 0x9Bu;
75                        while (v9 < 7);
76                        v9 = v16[1];
77                    }
78                    else
79                    {
80                        v13 = 4;
81                        v4 = 0;
82                        qmemcpy(v14, "!<=4s?6=4", sizeof(v14));
83                        do
84                        {
85                            v14[v4++] = 1;
86                            while (v4 < 0x53u;
87                                v5 = &v13;
88                                sub_4011E5("%s\n", (char*)v5);
89                                return 1;
90                        }
91                    }
92                }
93            }
94        }
95    } while (v1 < 0x18);
96    return 0;
97 }

```

3.1.2 task4

反汇编代码:

```

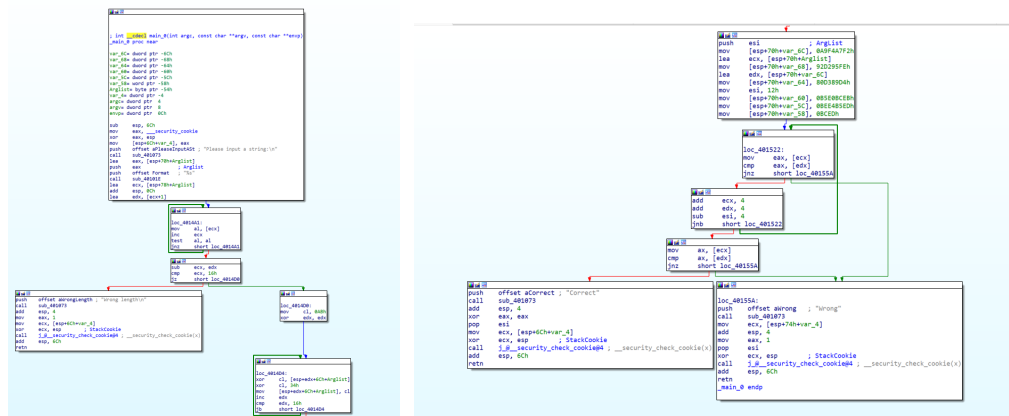
.text:0040144C      push    dword ptr [esp+10h]
.text:00401450      call   sub_4011CC
.text:00401455      push    dword ptr [eax+4]
.text:00401458      push    dword ptr [eax]
.text:0040145A      call   ds:__imp__stdio_common_vfscanf
.text:00401460      add     esp, 18h
.text:00401463      retn

.text:00401464      align 10h

.text:00401470      ;===== SUBROUTINE =====
.text:00401470      .text:00401470      ; int __cdecl main_0(int argc, const char **argv, const char **envp)
.text:00401470      .text:00401470      proc near
.text:00401470      var_6C      = dword ptr -6Ch
.text:00401470      var_68      = dword ptr -68h
.text:00401470      var_64      = dword ptr -64h
.text:00401470      var_60      = dword ptr -60h
.text:00401470      var_5C      = dword ptr -5Ch
.text:00401470      var_58      = word ptr -58h
.text:00401470      ArgList     = byte ptr -54h
.text:00401470      var_4        = dword ptr -4
.text:00401470      argc        = dword ptr 4
.text:00401470      argv        = dword ptr 8
.text:00401470      envp        = dword ptr 0Ch
.text:00401470
.text:00401470      sub     esp, 6Ch
.text:00401473      mov     eax, __security_cookie
.text:00401478      xor     eax, esp
.text:0040147A      mov     [esp+6Ch+var_4], eax
.text:0040147E      push    offset aPleaseInputASt ; "Please input a string:\n"
.text:00401483      call   sub_401073
.text:00401488      lea     eax, [esp+70h+ArgList]
.text:0040148C      push    eax
.text:0040148D      push    offset Format ; "%s"
.text:00401492      call   sub_40101E
.text:00401497      lea     ecx, [esp+78h+ArgList]

```

反汇编代码图形化显示：



反编译伪代码：

```
1 int __cdecl main_0(int argc, const char **argv, const char **envp)
2 {
3     char v3; // si
4     char v5; // cl
5     unsigned int i; // edx
6     char *v7; // ecx
7     int *v8; // edx
8     unsigned int v9; // esi
9     bool v10; // cf
10    char v11; // [ebp-4h] [ebp-70h]
11    int v12[5]; // [ebp+0h] [ebp-6Ch] BYREF
12    __int16 v13; // [ebp+14h] [ebp-58h]
13    char Arglist[80]; // [ebp+18h] [ebp-54h] BYREF
14
15    sub_401073("Please input a string:\n", v12[0]);
16    sub_40101E("%s", (char)Arglist);
17    if ( strlen(Arglist) == 22 )
18    {
19        v5 = -85;
20        for ( i = 0; i < 0x16; ++i )
21        {
22            v5 ^= Arglist[i] ^ 0x34;
23            Arglist[i] = v5;
24        }
25        v11 = v3;
26        v12[0] = -1443584014;
27        v7 = Arglist;
28        v12[1] = -1831692882;
29        v8 = v12;
30        v12[2] = -2133620268;
31        v9 = 18;
32        v12[3] = -1243562773;
33        v12[4] = -1092307475;
34        v13 = -17171;
35        while ( *(_DWORD *)v7 == *v8 )
36        {
37            v7 += 4;
38            ++v8;
39
40            v10 = v9 < 4;
41            v9 -= 4;
42            if ( v10 )
43            {
44                if ( *(_WORD *)v7 == *(_WORD *)v8 )
45                {
46                    sub_401073("Correct", v11);
47                    return 0;
48                }
49                break;
50            }
51            sub_401073("Wrong", v11);
52            return 1;
53        }
54        else
55        {
56            sub_401073("Wrong length\n", v12[0]);
57            return 1;
58        }
59    }
```

3.2 分别针对 task3、task4 反编译伪代码的计算过程、数据结构、条件判断等信息进行逆向分析，列出正确输入字符串的计算公式。

3.2.1 task3

观察代码，发现有很多地方都出现了加密的过程，比如说开始时的 `qmemcpy(ArgList, "B~wsaw2{b9f2s2af{u(", 22)`这句话就是对字母进行了加密，我们编写程序（具体程序全部放到后面一个版块），来求解字符串原本的含义。得出第一处的含义为“Please input a string”。

接下来是一个输入函数，将输入的字符串存入 Arglist 中，然后判断其长度是否为 20，对其进行相应处理，最后是结果是输出 v5 的字符串，我们先对长度不为 20 的部分（即错误部分）进行分析。

如果长度不为 20 的话，错误部分首先将 v13 赋值 4，将一个字符串赋给 v14，从 0-1 至 0xd-1 进行循环遍历 v14 字符串进行异或，注意到下图 v13 为 esp+14h，v14 起始为 esp+15h，因此，v14[-1]即为 v13，最后将 v5 指向 v13 的地址，最后输出 v5 的值，所以 v5 输出的即为 v13 开始到 v14 末尾的值。

我们还是用前面的方法解密出字符串，此处字符串的含义为“Wrong length”。

现在对正确的处理逻辑进行分析，当输入的字符串长度为 20 时，先对 v16 进行了一串的赋值操作，然后对输入的数据每一位与 0xA5 进行异或和 v16 每一位进行比较，当遍历到 20 个字符（即输入的字符串已经遍历完毕）且全部匹配时，与之前的 Wrong length 逻辑相同，v11 和 v12 也是相邻的变量，首先将 v11 和 v12 初始化，然后对其每一位与 0x57 进行异或，最后输出结果。

我们先解密出最后的输出结果（猜测是输出 correct），最后解密完发现确实是“correct!”！那么我们就有了思路，发现这一块长度为 20 的代码就是我们要求的那一部分。

还有一处，就是当输出 while 时，那一处遍历出现问题时就会输出“Wrong!”。

最后，写一个程序来计算出我们的求解结果：只需要用原来的一串数字和 0xA5 进行异或操作，公式如下所示：

```
for(i = 0; i < result().length; i++){  
    result[i] = (l[i] & 0xff) ^ 0xA5;  
}
```

最后得出正确的结果“TLDZB6QJqMJekYGtXeTY”。

3.2.2 task4

先输出一个“Please input a string:”，接着输入一个字符，如果字符的长度不等于 22 的话，就进不了 if 循环，直接跳转到 else 界面，输出“wrong length”。现在我们输入一个字符长度为 22 的字符，先给 v5 赋值为-85，接下来，进入 for 循环，做 22 次操作，将输入值的每一位都与 0x34 进行异或运算，再跟 v5

进行异或运算，再将值赋值给输入的第几位。做完 22 次运算后，我们进入下一个环节，先将操作完的字符串赋值给 v7，然后将 v12 的五位数字都进行赋值。

所以在逆向编写 C++ 程序的时候，我们对于 20 次异或运算的操作需要进行反向的编写，具体思路就是用本位与前一位进行异或，再与 0x34 进行异或。第一位的话我们用第一位与 0x34 进行异或，然后与 -85 进行异或。

最后进行比较环节，就是先初始化 v12，v13，然后让 v7 和 v8 的字节每四位比较一次，如果全部相同，那么就输出“correct”，否则就输出“Wrong”。

输入字符串的计算公式为：

$$\text{arg}[n] = \text{arg}[n] \wedge \text{arg}[n-1] \wedge (0x34), n \geq 2$$

$$\text{arg}[n] = \text{arg}[n] \wedge (0x34) \wedge (-85), n = 1$$

最后得出正确的结果：“magic_string_challenge”。

3.3 使用熟悉的编程语言，分别针对 task3、task4 编写脚本，计算出正确的字符串，提供脚本输出结果的截图

3.3.1 task3

计算第一部分的内容的 C++ 代码：

```
#include<iostream>
using namespace std;
int main() {
    char Arglist[80];
    string temp_1 = "B`wsaw2{[bgf2s2af`{[u(";
    for (int i = 0; i < 22; i++) {
        Arglist[i] = temp_1[i];
    }
    Arglist[22] = static_cast<char>('24');
    Arglist[23] = static_cast<char>('18');
    char Arglist_loaded[80];
    for (int i = 0; i < 23; i++) {
        Arglist_loaded[i] = static_cast<char>((static_cast<int>(Arglist[i]) ^ (0x12)));
    }
    for (int i = 0; i < 22; i++) {
        cout << Arglist_loaded[i];
    }
    return 0;
}
```

第一处的结果：

```
Microsoft Visual Studio 调试 × + v
the first answer:Please input a string:
C:\Users\Lenovo\Desktop\C++\???????x64\Debug\??????? .exe (?? 4664)???,?? 0?
????????? . .|
```

第二部分的代码：

```
1 #include<iostream>
2 using namespace std;
3 int main() {
4     char Arglist[80];
5     string temp_1 = "!<=4s?6=4'S";
6     for (int i = 1; i < 13; i++) {
7         Arglist[i] = temp_1[i - 1];
8     }
9     Arglist[0] = static_cast<char>(4);
10
11     char Arglist_loaded[80];
12     for (int i = 0; i < 13; i++) {
13         Arglist_loaded[i] = static_cast<char>((static_cast<int>(Arglist[i]) ^ (0x53)));
14     }
15
16     for (int i = 0; i < 13; i++) {
17         if (i == 0) {
18             cout << "the second answer:" << Arglist_loaded[i];
19         }
20         else {
21             cout << Arglist_loaded[i];
22         }
23     }
24
25     cout << Arglist[0];
26     return 0;
27 }
```

第二部分的输出结果：

```
Microsoft Visual Studio 调试
the second answer:Wrong length
C:\Users\Lenovo\Desktop\C++\???????x64\Debug\????????.exe (?? 22028)???,??? 0?
?????????. . .|
```

第三部分的代码：

```
1 #include<iostream>
2 using namespace std;
3 int main() {
4     char Arglist[80];
5     string temp_1 = "8%24#vW";
6     for (int i = 1; i < 9; i++) {
7         Arglist[i] = temp_1[i - 1];
8     }
9     Arglist[0] = static_cast<char>(20);
10
11     char Arglist_loaded[80];
12     for (int i = 0; i < 9; i++) {
13         Arglist_loaded[i] = static_cast<char>((static_cast<int>(Arglist[i]) ^ (0x57)));
14     }
15
16     for (int i = 0; i < 9; i++) {
17         if (i == 0) {
18             cout << "the third answer:" << Arglist_loaded[i];
19         }
20         else {
21             cout << Arglist_loaded[i];
22         }
23     }
24
25     return 0;
26
27 }
```

第三部分的结果：

```
Microsoft Visual Studio 调试 × + v
the third answer:Correct!
C:\Users\Lenovo\Desktop\C++\???????x64\Debug\????????.exe (?? 16496)???,??? 0?
?????????. . .|
```

第四部分（while 结构）的代码：

```
汇编第九次实验 (全局范围)
1      #include<iostream>
2      using namespace std;
3      int main() {
4          int l[10];
5          l[0] = -57 & 0xff;
6          l[1] = -30&0xff;
7          l[2] = -1 & 0xff;
8          l[3] = -2 & 0xff;
9          l[4] = -9 & 0xff;
10         l[5] = -79& 0xff;
11         l[6] = -112&0xff;
12         for (int i = 0; i < 7; i++) {
13             l[i] = l[i] ^ 0x90;
14         }
15
16         for (int i = 0; i < 7; i++) {
17             cout << static_cast<char>(l[i]);
18         }
19
20         return 0;
21     }
22
```

第四部分的结果：

```
Microsoft Visual Studio 调试 × + v
Wrong!
C:\Users\Lenovo\Desktop\C++\???????x64\Debug\????????.exe (?? 14860)???,??? 0?
?????????. . .|
```

最后的结果求解代码：

```
汇编第九次实验
1  #include<iostream>
2  using namespace std;
3  int main() {
4      int l[22];
5      l[0] = -15 & 0xff;
6      l[1] = -55&0xff;
7      l[2] = -31& 0xff;
8      l[3] = -1 & 0xff;
9      l[4] = -25& 0xff;
10     l[5] = -109&0xff;
11     l[6] = -12&0xff;
12     l[7] = -17 & 0xff;
13     l[8] = -44 & 0xff;
14     l[9] = -24 & 0xff;
15     l[10]= -17 & 0xff;
16     l[11]= -64 & 0xff;
17     l[12]= -50 & 0xff;
18     l[13]= -4 & 0xff;
19     l[14]= -30 & 0xff;
20     l[15]= -47 & 0xff;
21     l[16] = -3 & 0xff;
22     l[17] = -64 & 0xff;
23     l[18] = -15 & 0xff;
24     l[19] = -4 & 0xff;
25
26
27     for (int i = 0; i < 20; i++) {
28         l[i] = l[i] ^ 0xA5;
29     }
30
31     for (int i = 0; i <20; i++) {
32         cout << static_cast<char>(l[i]);
33     }
34
35     return 0;
36 }
```

求解结果：

```
Microsoft Visual Studio 调试
TLDZB6QJqMJekYGtXeTY
C:\Users\Lenovo\Desktop\C++\???????\x64\Debug\???????.exe (?? 10560)???,?? 0?
??????????. . .|
```

3.3.2 task4

我们使用 C++程序进行逆向的编写，求解出最后的结果。

```
1 #include<iostream>
2 #include<cstring>
3 using namespace std;
4 int main() {
5     int v12[5];
6     v12[0] = -1443584014 & 0xffffffff;
7     v12[1] = -1831692802 & 0xffffffff;
8     v12[2] = -2133620268 & 0xffffffff;
9     v12[3] = -1243562773 & 0xffffffff;
10    v12[4] = -1092307475 & 0xffffffff;
11    int v13 = -17171;
12    v13 = v13 & 0xffff;
13    char str[22]; int temp = 0;
14    for (int i = 0; i < 5; i++) {
15        str[temp++] = v12[i] & 0xff;
16        str[temp++] = v12[i] >> 8 & 0xff;
17        str[temp++] = v12[i] >> 16 & 0xff;
18        str[temp++] = v12[i] >> 24 & 0xff;
19    }
20    str[temp++] = v13 & 0xff;
21    str[temp++] = v13 >> 8 & 0xff;
22    char str_1[22];
23    for (int j = 0; j < 22; j++) {
24        if (j == 0) {
25            str_1[j] = static_cast<char>((str[j] ^ (0x34) ^ (-85 & 0xff)));
26        }
27        else {
28            str_1[j] = static_cast<char>((str[j] ^ (0x34) ^ (str[j-1])));
29        }
30    }
31    for (int i = 0; i < 22; i++) {
32        cout << str_1[i];
33    }
34    return 0;
35 }
36 }
```

最后得出结果：

```
Microsoft Visual Studio 调试
magic_string_challenge
C:\Users\Lenovo\Desktop\C++\????????\x64\Debug\?????????.exe (?? 32940)???,??? 0?
?????????. . .|
```

3.4 分别运行程序 task3、task4，输入计算得到的字符串进行验证，获得“Correct”输出，提供截图。

3.4.1 task3

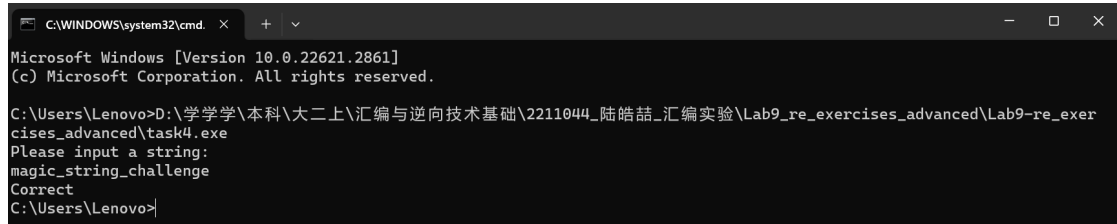
如图，我们输入 T1DZB6QJqMJekYGtXeTY，得到结果 “Correct!”，实验成功！

```
C:\Users\Lenovo>D:\学学学\本科\大二上\汇编与逆向技术基础\2211044-陆皓喆_汇编实验\Lab9_re_exercises_advanced\Lab9-re_exer
cises_advanced\task3.exe
Please input a string:
T1DZB6QJqMJekYGtXeTY
Correct!

C:\Users\Lenovo>
```

3.4.2 task4

如图，我们输入上面得到的字符串“magic_string_challenge”，最后输出结果“Correct”，实验成功！



```
C:\WINDOWS\system32\cmd. x + v
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>D:\学学学\本科\大二上\汇编与逆向技术基础\2211044_陆皓喆_汇编实验\Lab9_re_exercises_advanced\Lab9-re_exer
cises_advanced\task4.exe
Please input a string:
magic_string_challenge
Correct
C:\Users\Lenovo>
```

四、总结与反思

从本次实验中，我学到了更多的逆向分析的知识与方法，处理了两个带有for循环、while循环等复杂结构的逆向分析处理，对逆向分析的思路以及IDA的使用更加熟练了，希望之后我还能继续用自己的汇编知识去解决一些更难的问题。