

汇编语言与逆向技术课程实验报告

实验九：Reverse Engineering Exercises –Advanced



学院: 密码与网络空间安全学院

专业: 信息安全

学号: 2412950

姓名: 路浩斌

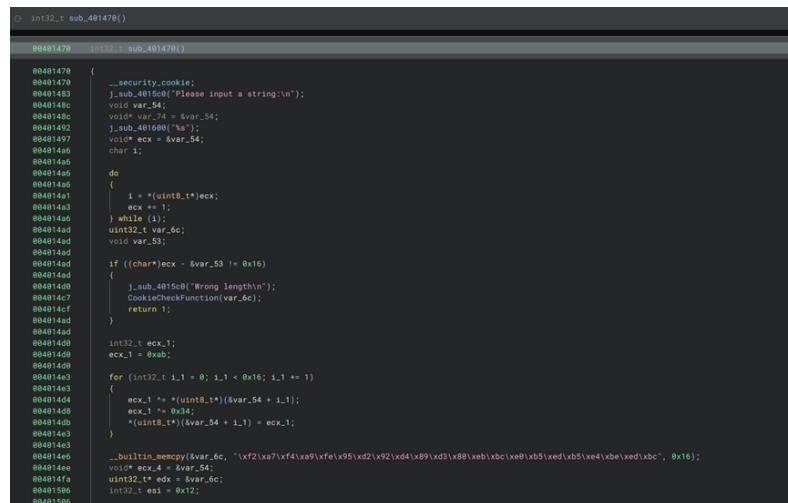
班级: 信安一班

一、实验目的

- 1、进一步熟悉静态反汇编工具 Binary Ninja；
- 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

1. 通过 Binary Ninja 得到 task3.exe 和 task4.exe 的反汇编代码。
2. 使用 Binary Ninja 的反编译功能 (F5 快捷键) 得到伪 C 代码，如图 1 所示，右键点击数字对象选择 “Display as” 可实现数制转换。



```
0x001470 ; sub_401470()

00401470 {
    __security_cookie;
    j_sub_4015c0("Please input a string:\n");
    void* var_54 = &var_54;
    void* var_74 = &var_54;
    j_sub_401680("%s");
    void* ecx = &var_54;
    char i;
    void* var_53;
    void* var_60;
    do
    {
        i = *(uint8_t*)ecx;
        ecx += 1;
    } while (i);
    uint32_t var_60;
    uint32_t var_53;
    void var_53;
    if ((char)ecx - &var_53 != 0x16)
    {
        j_sub_4015c0("Wrong length\n");
        CookieCheckFunction(var_60);
        return 1;
    }
    int32_t ecx_1;
    ecx_1 = 0xb0;
    for (int32_t i_1 = 0; i_1 < 0x16; i_1 += 1)
    {
        ecx_1 *= *(uint8_t*)(var_54 + i_1);
        ecx_1 ^= 0x3f;
        *(uint8_t*)(var_54 + i_1) = ecx_1;
    }
    ...builtin_memcpy(var_60, "\xf2\x07\x41\x91\xfe\x95\xd2\x21\x41\x89\xd3\x80\xeb\xbc\xe0\xb5\xed\xb5\xe4\xbe\xed\xbc", 0x16);
    void* ecx_4 = &var_54;
    uint32_t* edx = &var_60;
    int32_t es1 = 0x12;
}
```

图 1: 逆向分析，完成 task3 和 task4 练习

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++、Python 等) 对待输入字符串进行计算，完成逆向分析挑战。



图 2: 逆向分析，完成 task3 和 task4 练习

三、实验报告内容

3.1. task3

3.1.1 反汇编代码和伪 C 代码

```

00402a70 81ec0a000000 sub esp, 0xa4
00402a75 31ff mov eax, dword [var_a4]
00402a7d 33c4 xor eax, esp [var_a4]
00402a7d 89942400000000 mov dword [esp+0x0a0 (var_4)], eax
00402a84 c644242442 mov byte [esp+0x24 (var_80)], 0x42
00402a89 33c9 xor ecx, ecx (0x0)
00402a90 c64424257e mov byte [esp+0x25 (var_7f)], 0x7e
00402a90 c64424257f mov byte [esp+0x26 (var_7e)], 0x77
00402a95 c644242773 mov byte [esp+0x27 (var_7d)], 0x79
00402a9a c644242861 mov byte [esp+0x28 (var_7c)], 0x61
00402a9f c644242977 mov byte [esp+0x29 (var_7b)], 0x77
00402aa4 c644242a32 mov byte [esp+0x2a (var_7a)], 0x32
00402aa9 c644242b7b mov byte [esp+0x2b (var_79)], 0x7b
00402aae c644242c7c mov byte [esp+0x2c (var_78)], 0x7c
00402ab2 c644242d62 mov byte [esp+0x2d (var_77)], 0x62
00402ab6 c644242e67 mov byte [esp+0x2e (var_76)], 0x67
00402abd c644242f65 mov byte [esp+0x2f (var_75)], 0x66
00402ac2 c644243032 mov byte [esp+0x30 (var_74)], 0x32
00402act c644243173 mov byte [esp+0x31 (var_73)], 0x70
00402ad2 c644243202 mov byte [esp+0x32 (var_72)], 0x32
00402ad1 6844243361 mov byte [esp+0x33 (var_71)], 0x61
00402ad6 c644243466 mov byte [esp+0x34 (var_70)], 0x66
00402ad9 c644243560 mov byte [esp+0x35 (var_6f)], 0x60
00402ade c644243676 mov byte [esp+0x36 (var_6e)], 0x7b
00402ae5 c64424377c mov byte [esp+0x37 (var_6d)], 0x7c
00402aef c644243875 mov byte [esp+0x38 (var_6c)], 0x75
00402aef c644243928 mov byte [esp+0x39 (var_6b)], 0x28
00402af4 c644243a18 mov byte [esp+0x3a (var_6a)], 0x18
00402af9 c644243b12 mov byte [esp+0x3b (var_69)], 0x12
00402afe 6690 nop
00402b00 8a440c24 mov al, byte [esp+ecx+0x24 (var_80)]

```

图 3: task3 反汇编代码

task3 反汇编伪 C 代码:

```

char var_a4;
int32_t var_4 = __security_cookie ^ &var_a4;
char var_80;
__builtin_memcpy(&var_80, "B~wsaw2{|bf2s2af`{|u", 0x16);
int32_t i = 0;
char var_6a = 0x18;
char var_69 = 0x12;

do
{
    int32_t eax_1;
    (uint8_t*)eax_1 = (&var_80)[i];
    (uint8_t)eax_1 ^= 0x12;
    (&var_80)[i] = (uint8_t)eax_1;
    i += 1;
} while (i < 0x18);

char* var_a8 = &var_80;
j_sub_401660("%s");
void var_54;
void* var_b0 = &var_54;
j_sub_401e10("%s");
char* ecx = &var_54;
char i_1;

do
{
    i_1 = *(uint8_t*)ecx;
    ecx = &ecx[1];
} while (i_1);
void var_53;
char* eax_2;

```

图 5: (1)

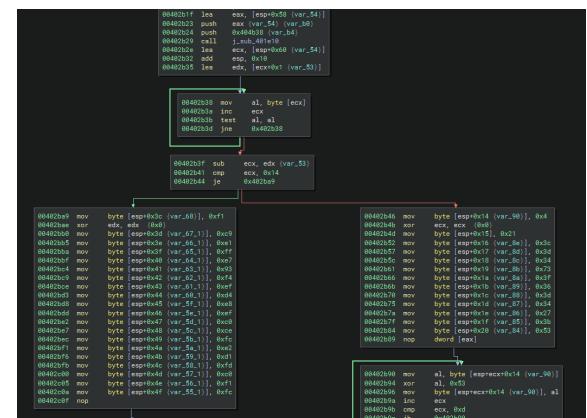


图 4: task3 反汇编图形化

```

if (ecx - &var_53 == 0x14)
{
    char var_68 = 0xf1;
    int32_t edx_1 = 0;
    char var_67_1 = 0xc9;
    char var_66_1 = 0xe1;
    char var_65_1 = 0xff;
    char var_64_1 = 0xe7;
    char var_63_1 = 0x93;
    char var_62_1 = 0xf4;
    char var_61_1 = 0xef;
    char var_60_1 = 0xd4;
    char var_5f_1 = 0xe8;
    char var_5e_1 = 0xef;
    char var_5d_1 = 0xc0;
    char var_5c_1 = 0xce;
    char var_5b_1 = 0xfc;
    char var_5a_1 = 0xe2;
    char var_59_1 = 0xd1;
    char var_58_1 = 0xfd;
    char var_57_1 = 0xc0;
    char var_56_1 = 0xf1;
    char var_55_1 = 0xfc;
    uint32_t eax_3;

    while (((int32_t)*(uint8_t*)(&var_54 + edx_1) ^ 0xa5)
           == (uint32_t)(&var_68)[edx_1])
    {
        edx_1 += 1;

        if (edx_1 >= 0x14)
        {
            char var_9c = 0x14;
            int32_t i_2 = 0;

```

图 6: (2)

```

if (edx_1 >= 0x14)
{
    char var_9c = 0x14;
    int32_t i_2 = 0;
    char var_9b;
    __builtin_memcpy(&var_9b, "8%24#vW", 8);

    do
    {
        (uint8_t)eax_3 = (&var_9c)[i_2];
        (uint8_t)eax_3 ^= 0x57;
        (&var_9c)[i_2] = (uint8_t)eax_3;
        i_2 += 1;
    } while (i_2 < 9);

    char* var_a8_1 = &var_9c;
    j_sub_401660("%s\n");
    CookieCheckFunction(var_a4);
    return 0;
}

var_a4 = 0xc7;
int32_t i_3 = 0;
char var_a3_1 = 0xe2;
char var_a2_1 = 0xff;
char var_a1_1 = 0xfe;
char var_a0_1 = 0xf7;
char var_9f_1 = 0xb1;
char var_9e_1 = 0x90;

```

图 7: (3)

```

do
{
    (uint8_t)eax_3 = (&var_a4)[i_3];
    (uint8_t)eax_3 ^= 0x90;
    (&var_a4)[i_3] = (uint8_t)eax_3;
    i_3 += 1;
} while (i_3 < 7);

eax_2 = &var_a4;
}
else
{
    char var_90 = 4;
    int32_t i_4 = 0;
    char var_8f;
    __builtin_memcpy(&var_8f, "!<=4s?6=4\\'S", 0xc);

    do
    {
        (&var_90)[i_4] ^= 0x53;
        i_4 += 1;
    } while (i_4 < 0xd);

    eax_2 = &var_90;
}

char* var_a8_2 = eax_2;
j_sub_401660("%s\n");
CookieCheckFunction(var_a4);
return 1;

```

图 8: (4)

3.1.2 逆向分析

首先对一段以加密形式存储的提示字符串进行逐字节解密，即与 0x12 进行异或运算还原出可读文本，并将其输出以提示用户输入字符串。随后，程序读取用户输入内容，并将其存储在内存空间中。

之后程序通过指针递增的方式遍历输入字符串，通过检测字符串结束符，从而计算输入字符串的实际长度。接下来判输入字符串长度是否为 20，若长度不为 20，程序进入错误处理分支，对另一段加密字符串逐字节和 0x53 进行异或运算解密，输出“Wrong length” 提示信息。

若长度为 20，程序以一个长度为 20 的参照字符串为基准，对用户输入的字符串进行逐字节处理：每一位输入字符首先与固定常量 0xA5 进行异或运算，然后将运算结果与参照字符串中对应位置的数据进行比较。若任意一位比较结果不相等，程序立即判定输入不合法，并进入错误分支，输出解密后的“Wrong!” 提示信息。

只有当输入字符串的全部 20 个字符均通过上述校验条件时，程序才会进入成功分支。在该分支中，程序同样采用异或解密的方式对成功提示字符串进行还原，并最终输出“correct!”。

由于异或运算具有可逆性，因此可以反向推导出程序所期望的正确输入字符串。设参照序列中第 i 位数据为 $test[i]$ ，输入字符串第 i 位字符为 $Input[i]$ 由此可得正确输入字符串的计算公式为：

$$Input[i] = test[i] \oplus 0xA5, \quad 0 \leq i < 20$$

可以成功还原出程序要求的正确输入字符串，其结果为：

TLDZB6QJqMJekYGtXeTY

3.1.3 编写脚本计算正确字符串

task3 程序中有三处需要解密得出正确字符串：

(1)

```
1 #include<iostream>
2 using namespace std;
3 int main() {
4     int a = 0x12;
5     int test[24] = {
6         0x42, 0x7e, 0x77, 0x73, 0x61, 0x77, 0x32, 0x7b, 0x7c, 0x62
7         , 0x67, 0x66, 0x32, 0x61, 0x66, 0x60, 0x7b, 0x7c, 0x75
8         , 0x28, 0x18, 0x12
9     };
10    for (int i = 0; i < 24; i++) {
11        cout << char(test[i] ^ a);
12    }
13    cout << endl;
14    return 0;
15 }
```

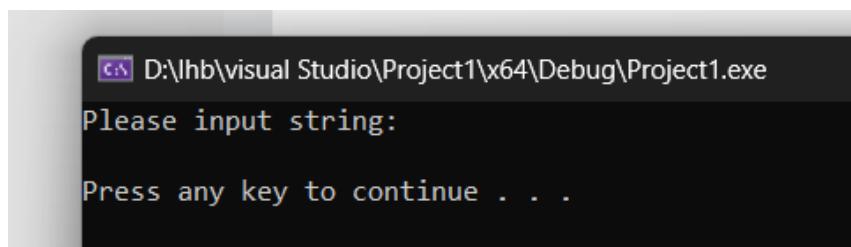


图 9：解密得到 Please input string:

(2)

```
1 #include<iostream>
2 using namespace std;
3 int main() {
4     int a = 0x53;
5     int test[13] = { 0x4, 0x21, 0x3c, 0x3d, 0x34, 0x73,
6                     0x3f, 0x36, 0x3d, 0x34, 0x27, 0x3b, 0x53 };
7     for (int i = 0; i < 13; i++) {
8         cout << char(test[i] ^ a);
```

```
9     }
10    cout << endl;
11    return 0;
12 }
```

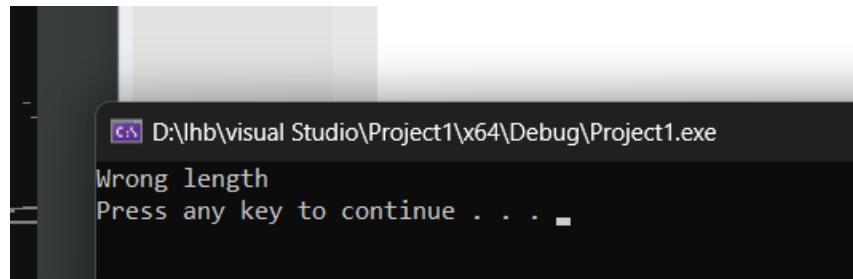


图 10: 解密得到 Wrong length

(3)

```
1 #include<iostream>
2 using namespace std;
3 int main() {
4     int a = 0x90;
5     int test[7] = { 0xc7,0xe2,0xff,0xfe,0xf7,0xb1,0x90 };
6     for (int i = 0; i < 7; i++) {
7         cout << char(test[i] ^ a);
8     }
9     cout << endl;
10    return 0;
11 }
```

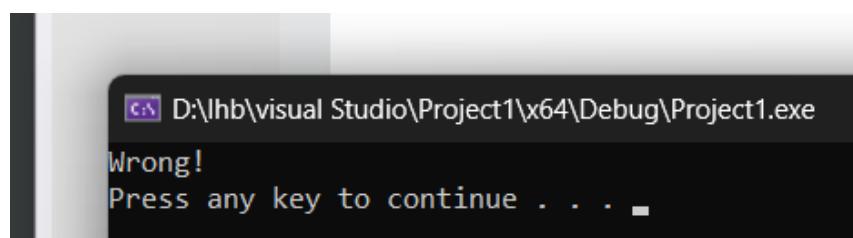


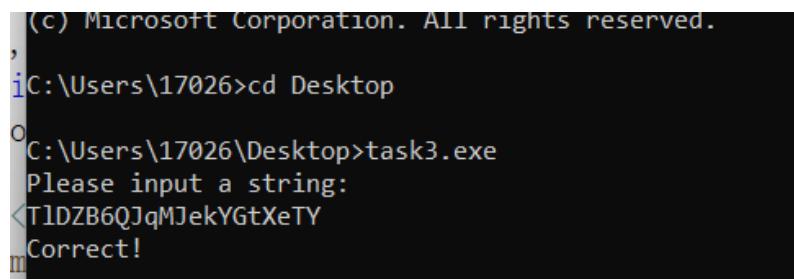
图 11: 解密得到 Wrong!

3.1.4 成功截图

编写 c++ 代码解析出逆向结果：

```
1 #include<iostream>
2
3 using namespace std;
4
5 int main() {
6     int N = 20;
7     int a = 0xa5;
8     int test[20] = {0xf1,0xc9,0xe1,0xff,0xe7,0x93,0xf4
9         ,0xef,0xd4,0xe8,0xef,0xc0,0xce,0xfc,0xe2,0xd1,0xfd,0xc0,0xf1,0xfc
10        };
11
12     for (int i = 0; i < N; i++) {
13         cout << char(test[i] ^ a);
14     }
15
16     cout << endl;
17
18     return 0;
19 }
20 }
```

根据 c++ 程序,我们可以得到结果为”TLDZB6QJqMJekYGtXeTY”, 运行 task3.exe 并输入后得到截图:



The screenshot shows a terminal window with the following text:
(c) Microsoft Corporation. All rights reserved.
C:\Users\17026>cd Desktop
C:\Users\17026\Desktop>task3.exe
Please input a string:
<TLDZB6QJqMJekYGtXeTY
Correct!

图 12: task3.exe 的成功截图

得到 correct, 结果正确。

3.2 task4

3.2.1 反汇编代码和伪 C 代码

```

00401463 c3          retf    (.__return_addr)
00401464 02 00 00 00 00 00 00 00 00 00 00 00 ..... .
00401470 int32_t sub_401470()

00401478 83ec5c      sub    esp, 0x6c
00401479 a1000040000 mov    esx, dword [.__security_cookie]
0040147a xor    eax, esp, [var_6c]
0040147b 89442468    mov    dword [esp+0x8], eax
0040147c 68304040000 push   data_404b30 (..._saved_esi) ("Please input a string:\n")
0040147d e8e1bfffff  call   j._sub_4015e8
0040147e 84424414    lea    ecx, [esp+0x24 (var_54)]
0040147f push   eax, [var_54]
00401480 83ec4040000 push   0x404b4c (var_78)
00401492 89770fffff  call   j._sub_401600
00401497 8d4c2424    lea    ecx, [esp+0x24 (var_54)]
00401498 83c400       add    esp, 0x4
0040149e 805101     lea    edx, [ecx+0x1 (var_53)]

004014a1 8a01        mov    al, byte [ecx]
004014a2 inc    cx
004014a3 84c0         test   al, al
004014a6 jne    0x4014a1

004014a8 83c916     sub    ecx, edx [var_53]
004014a9 cmp    cl, 0x16
004014ad je    0x4014d9

004014af 68304040000 push   data_404b50 (..._saved_esi) ("Wrong length\n")
004014b0 897affffff  call   j._sub_4015c0
004014b1 83c401     add    esp, 0x4
004014b2 805101     lea    edx, [ecx+0x1 (var_53)]

```

图 13: task4 反汇编代码

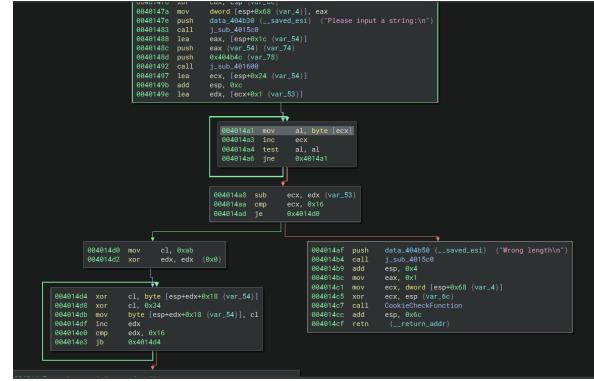


图 14: task4 图形化显示

task4 反汇编伪 C 代码:

```

__security_cookie;
j_sub_40150("Please input a string:\n");
void var_54;
void* var_74 = &var_54;
j_sub_401600("%s");
char* ecx = &var_54;
char i;

do
{
    i = *(uint8_t*)ecx;
    ecx = &ecx[1];
} while (i);
uint32_t var_6c;
void var_53;

if (ecx - &var_53 != 0x16)
{
    j_sub_4015c0("Wrong length\n");
    CookieCheckFunction(var_6c);
    return 1;
}

int32_t ecx_1;
(uint8_t)ecx_1 = 0xab;

for (int32_t i_1 = 0; i_1 < 0x16; i_1 += 1)
{
    (uint8_t)ecx_1 ^= *(uint8_t*)(&var_54 + i_1);
    (uint8_t)ecx_1 ^= 0x34;
    *(uint8_t*)(&var_54 + i_1) = (uint8_t)ecx_1;
}

```

图 15: (1)

```

}
__builtin_memcpy(&var_6c,
    "xf2\x7\xf4\x9\xfe\x95\xd2\x92\xd4\x89\xd3\x80\xeb\xbc\xe0\xb5\xed\xb5\xe4\xbe\x"
    "\xd\xbc",
    0x16);
int16_t* ecx_4 = &var_54;
uint32_t* edx = &var_6c;
int32_t esi = 0x12;

while (*(>uint32_t*)ecx_4 == *(>uint32_t*)edx)
{
    ecx_4 = &ecx_4[1];
    edx = &edx[1];
    int32_t temp0_1 = esi;
    esi -= 4;

    if (temp0_1 < 4)
    {
        int32_t eax_3;
        (uint16_t)eax_3 = *(>uint16_t*)ecx_4;

        if ((uint16_t)eax_3 != *(>uint16_t*)edx)
            break;

        j_sub_4015c0("Correct");
        CookieCheckFunction(var_6c);
        return 0;
    }
}

j_sub_4015c0("Wrong");
CookieCheckFunction(var_6c);
return 1;

```

图 16: (2)

3.2.2 逆向分析

首先输出提示信息，并将用户输入的数据存储在内存空间中。程序通过遍历输入字符串，检测字符串结束符的方式计算输入字符串的实际长度。

判断字符串长度是否为 22。若不为 22，程序直接输出“Wrong length”提示信息。

若为 22，则将输入字符串的每一个字符与它的前一个字符进行异或运算，其中第一个字符与 0xab 进行异或运算。随后每个字符再与 0x34 进行一次异或操作，并将得到的新的字符串写回到输入字符串对应位置。

完成上述操作后，程序将得到的新字符串与程序内的一个 22 位字符串逐段进行比较。只要任意一段数据不匹配，程序便立即终止比较过程，并输出“Wrong”提示信息。只有当输入数据在全部比较过程中均与参考数据完全一致时，程序输出“Correct”

在逆向编写 C++ 程序的时候，我们对于 22 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 0xab 进行异或。

正确输入字符串的计算公式为：

$$input[n] = input'[n] \oplus input[n - 1] \oplus 0x34, \quad n \geq 2$$

$$input[n] = input'[n] \oplus 0x34 \oplus 0xab, \quad n = 1$$

最终可得正确输入字符串为：

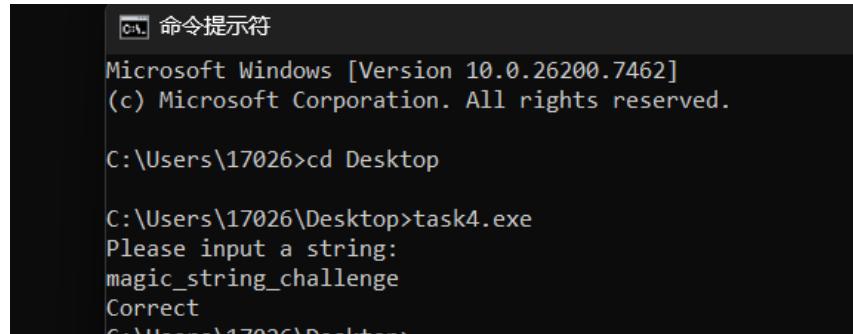
magic_string_challenge

3.3.3 成功截图

编写 c++ 代码将解析出逆向结果：

```
1 #include<iostream>
2 using namespace std;
3 int main() {
4     int a = 0x34;
5     int b = 0xab;
6     int test[22] = {
7         0xf2,0xa7,0xf4,0xa9,0xfe,0x95,0xd2,0x92,0xd4,
8         0x89,0xd3,0x80,0xeb,0xbc,0xe0,0xb5,0xed,0xb5,0xe4,0xbe,0xed,0
9             xbc
10    };
11    cout << char(test[0] ^ a ^ b);
12    for (int i = 1; i < 22; i++) {
13        cout << char(test[i] ^ a ^ test[i-1]);
14    }
15    cout << endl;
16    system("pause");
17    return 0;
}
```

根据 c++ 程序，我们可以得到结果为”magic_string_challenge”，运行 task4.exe 并输入后得到截图：



```
命令提示符
Microsoft Windows [Version 10.0.26200.7462]
(c) Microsoft Corporation. All rights reserved.

C:\Users\17026>cd Desktop

C:\Users\17026\Desktop>task4.exe
Please input a string:
magic_string_challenge
Correct
```

图 17: task4.exe 的成功截图

得到 correct，结果正确。