

计算机系统基础 Lab2 实验报告

1. 实验内容:

完成 ReverseLab。

2. 实验步骤:

1. 安装 IDA。
2. 做题。
3. 在网上平台提交获得的 flag。
4. 撰写实验报告。

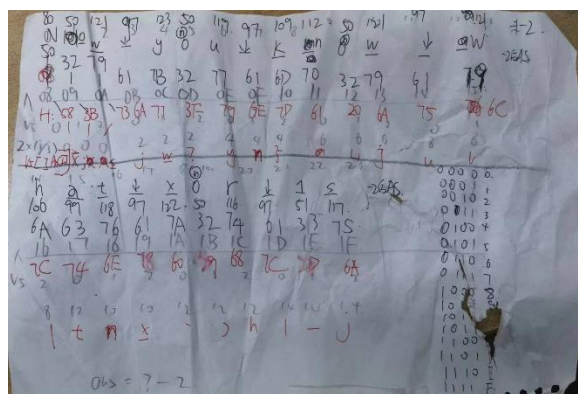
3. 解题步骤:

1. 反编译程序得知 flag 与目标字符串 (Welcome_to_the_reverse_world!)

存在 1-1, o-0 的字符互换关系, 从而得到 flag。

2. 反编译程序得知 flag 与目标字符组 (见下图) 存在特定操作关系, 对各个目标字符逆推, 从而得到 flag。

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int i; // [rsp+20h] [rbp-38h]
4     __int64 v5[5]; // [rsp+30h] [rbp-28h]
5
6     puts("Input the password:");
7     sub_140001150("%s", byte_140003040);
8     v5[0] = (__int64)"X1j3y5a7u9t;=|";
9     v5[1] = (__int64)"5w7n9;len?A-";
10    v5[2] = (__int64)"s9?;sj?|AxChEj";
11    for ( i = 0; i < 24; ++i )
12    {
13        if ( ((i + 8) ^ *(char *) (v5[i % 3] + 2 * (i / 3))) - byte_140003040[i] != 2 )
14        {
15            puts("flag(This_is_a_flag)");
16            break;
17        }
18    }
19    system("pause");
20    return 10086;
21 }
```



3. 阅读反编译程序得知对输入字符串进行的操作与判断, 根据详细方式编写

c 程序进行枚举, 获得 md5 加密字符串, 利用程序附带的加解密网站获得

flag。

4. 查看 main 函数中 if 语句所调用的未知函数，得到程序对输入字符串的判断方式与目标字符串位置，并依此进行逆推，即可获得 flag。
5. 查看 main 函数得知对输入字符串进行的操作与判断，根据详细方式编写 c 程序进行枚举，即可获得 flag。

```

1 #include <iostream>
2 using namespace std;
3 char temp=0;
4 char now=0;
5 int i;
6 string target = "Zxb3xo.qe4.Dob@q";
7 bool check(char nowhat,int which) //检查对不对，对的话返回1
8 {
9     if(nowhat==target[which])
10         return 1;
11     else
12         return 0;
13 }
14
15 // target str
16 // Zxb3xo.qe4.Dob@q
17 // Z x b 3 x . o _ q e 4 . D o b @ q
18 // 90 120 98 51 120 111 95 113 101 52 95 68 111 98 64 113
19 int main(void)
20 {
21     char ans[16] = {0};
22     for ( int i = 0; i < 16; ++i )
23     {
24         for(temp = 0;temp<255;temp++)
25         {
26             if ( temp < 65 || temp > 90 )
27             {
28                 if ( temp < 97 || temp > 122 )
29                 {
30                     now = temp;
31                     if(check(now,i))
32                     {
33                         ans[i] = temp;
34                         break;
35                     }
36                 }
37             }
38         }
39     }
40 }

```

(程序过长，剩余部分即为反编译程序判断部分的重构，不再赘述)

6. 本题难度不大，但数据操作过于繁琐：首先阅读反编译程序获得数据处理的详细步骤，发现全过程中 dword_1400040A8[4]作为参数对 Str 进行操

```

1 int __cdecl main(int argc, const char **argv, const char *
2 {
3     int j; // [rsp+20h] [rbp-18h]
4     int i; // [rsp+24h] [rbp-14h]
5     int v6; // [rsp+28h] [rbp-10h]
6
7     sub_140001B80("%s", Str, 40i64);
8     if ( strlen(Str) == 32 )
9     {
10         dword_1400040A8[4] += 16;
11         dword_1400040A8[3] += Str[3] * dword_1400040A8[4];
12         dword_1400040A8[4] += 3;
13         dword_1400040A8[3] += Str[2] * dword_1400040A8[4];
14         dword_1400040A8[4] -= 10;
15         dword_1400040A8[0] += Str[3] * dword_1400040A8[4];
16         dword_1400040A8[4] -= 2;
17         dword_1400040A8[1] += Str[2] * dword_1400040A8[4];
18         dword_1400040A8[4] += 13;
19         dword_1400040A8[0] += Str[1] * dword_1400040A8[4];
20         dword_1400040A8[4] -= 8;
21         dword_1400040A8[2] += Str[1] * dword_1400040A8[4];
22         dword_1400040A8[4] -= 7;
23         dword_1400040A8[2] -= 3481;
24         dword_1400040A8[4] += 3;
25         dword_1400040A8[1] -= 2422;
26         dword_1400040A8[4] += 9;
27     }
28 }
00000460 main:1 (140001060)

```

作并保存到 dword_1400040A8 的其他位置中，
由 for 循环内判断条件得知最终
dword_1400040A8 的其他位置均为 0。

$$\begin{aligned}
 12W + 20X + 17Y + 9Z - 458 &= 0 \\
 7W + 9X + 7Y + 8Z - 2422 &= 0 \\
 13W + 12X + 15Y + 5Z - 3781 &= 0 \\
 19W + 11X + 17Y + 16Z - 5006 &= 0
 \end{aligned}$$

81. 84. 69 77

```

49 | dword_140004048[4] += 2;
50 | dword_140004048[1] += Str[1] * dword_140004048[4];
51 | for ( i = 0; i < 4; ++i )
52 | {
53 |     if ( dword_140004048[i] )
54 |         goto LABEL_11;
55 | }
56 | dword_140004040 = 0x7C007C00100064i64;
57 | *(&dword_140004040 + 1) = 0x34002F00170022i64;
58 | *(&dword_140004040 + 2) = 0x3B000000130013i64;
59 | *(&dword_140004040 + 3) = 0xF001B001Fi64;
60 | *(&dword_140004040 + 4) = 0x3C00720005001Fi64;
61 | *(&dword_140004040 + 5) = 0x40036001F0062i64;
62 | *(&dword_140004040 + 6) = 0x3F0032002F001Fi64;
63 | *(&dword_140004040 + 7) = 0i64;
64 | for ( j = 0; j < 28; ++j )
65 | {
66 |     if ( (Str[j % 4] ^ Str[j + 4]) != *((char *)&dword_140004040 + 2 * j) )
67 |         goto LABEL_11;
68 | }
69 | puts("right!");
70 | }
71 | LABEL_11:
72 | system("pause");
73 | return 10000;
}
00000074 main:49 (140001974)

```

从而列四元一次方程解得 Str 内各数值，并利用 c 程序由这已知的前四位数使用最后的 for 循环进行递推，从而得到全部的 Str 内容，即为 flag。

7. 本题为 qt 程序，调用了众多 qt 库函数，使反编译与阅读变得困难，故先从调用的 qt 函数入手。发现调用了 qt 的 bytearray 加密函数。又从 string 窗口发现五个“奇妙的”字符串（aHFrcWstMjE4MjMtamNoZGts，YWJjZGUtMzg0ODMta2RraHly，YWJjZGUtMTIzNDUzZ2hpamts，eHh4eXktMTIzNDUtamtvcGlw，UXRmdW4tMTAwODYtRlVJdG9v）。发现依此通过在线解密网站所解密出的字符串与点击 1000 此后所得密钥存在雷同，故尝试其余解密后字符串，即得 flag。
8. 阅读反编译程序，发现 jmp 语句跳转至无效地址，判断为花指令。对其进行处理后即得正确程序，从而获得 flag。
9. 本题使用了多样的花指令，并且使用 IsDebuggerPresent 函数对两个线程的函数进行了干扰，导致动态调试错误。通过逐步分析两个线程内的程序，追踪程序控制流活动，按照正确的顺序处理不合理的 jmp、互补跳转与多余的 ret 即可获得正确的字符串加密函数与动态调试的干扰函数。

```

.text:004011AE FF 15 00 20 40 00 call ds:Sleep
.text:004011B4 64 8B 1D 30 00 00 00 mov ebx, large fs:30h
.text:004011B8 0F B6 58 02 movzx ebx, byte ptr [ebx+2]
.text:004011BF 90 nop
.text:004011C0 90 nop
.text:004011C1 90 nop
.text:004011C2 90 nop
.text:004011C3 90 nop
.text:004011C4 90 nop
.text:004011C5 90 nop
.text:004011C6 90 nop
.text:004011C7 90 nop
.text:004011C8 90 nop
.text:004011C9 58 pop eax
.text:004011CA 83 C3 09 add ebx, 9
.text:004011CD 03 C3 add eax, ebx
.text:004011CF 50 push eax
.text:004011D0 90 nop
.text:004011D1 90 nop
.text:004011D2 90 nop
.text:004011D3 FF C0 inc eax
.text:004011D5 48 dec eax
.text:004011D6 8B 4D FC mov ecx, [ebp+var_4]
.text:004011D9 0F B6 91 6C 33 40 00 movzx edx, byte_40336C[ecx]
.text:004011E0 83 C2 23 add edx, 23h

```

编写程序进行穷举即可获得 falg。

```

1 | DWORD __stdcall StartAddress(LPVOID lpThreadParameter)
2 | {
3 |     int v2; // [esp+0h] [ebp-18h]
4 |     int i; // [esp+14h] [ebp-4h]
5 |
6 |     CreateThread(0, 0, hHandle, 0, 0, 0);
7 |     WaitForSingleObject(hHandle, 0xFFFFFFFF);
8 |     for ( i = 0; i < 42; ++i )
9 |     {
10 |         byte_40336C[i] = (byte_40336C[i] << 6) ^ ((int)(unsigned __int8)byte_40336C[i] >> 2);
11 |         byte_40336C[i] ^= 0x23u;
12 |         Sleep(6u);
13 |         v2 = NtCurrentPeb()->BeingDebugged + 9;
14 |         byte_40336C[i] += 35;
15 |     }
16 |     return 0;
17 | }

```



```
21 for(int i=0;i<42;i++)
22 {
23     flag=0;
24     for(char trynow=0;trynow<=126;trynow++)
25     {
26         //cout << "try now: " << trynow << ' ';
27
28         char left = char(trynow<<6);
29         char right = (int)(unsigned)trynow >> 2;
30         unsigned char ans1 = left ^ right;
31         unsigned char ans2 = ans1 ^ 0x23u;
32         unsigned char ans3 = ans2 + 35;
33         if(ans3 == ansint[i])
34         {
35             //cout << endl << endl << (char)trynow << endl << endl;
36             cout << (char)trynow;
37             flag=1;
38             break;
39         }
40     }
41     if(flag==0)
42     ;
43     // cout << "|failed: " << i << endl << endl;
44 }
45 }
```

10. 使用Detectiteasy 对程序进行分析,发现程序使用UPX 进行了打包处理。

故先从网上搜索 UPX 工具进行解包,再进行分析。我们可以在程序推出前设置断点,输入各种字符观察内存数据变化,从而寻找数据与输入的对应关系,进而获得 flag。

4. 实验收获与心得:

熟悉了汇编语言, ida 使用方法与程序编写过程中常用的花指令。

动用 ida 之前应该先使用 die 判断程序操作系统及有无加密,对程序进行更改前应及时保存,应该熟练利用编程工具进行辅助,最好别用笔手搓。