汇编语言与逆向技术实验报告

Lab5- Reverse Engineering Challenge

学号：2112514　姓名：辛浩然　专业：信息安全、法学

一、 实验目的

1.熟悉静态反汇编工具 IDA Freeware；

2.熟悉反汇编代码的逆向分析过程；

3.掌握反汇编语言中的数学计算、数据结构、条件判断、分支结构的识别和逆向分析。

二、 实验内容

1.使用 IDA，获得二进制代码的反汇编代码；

2.逆向分析二进制代码的计算过程、数据结构、条件判断、分支结构等信息；

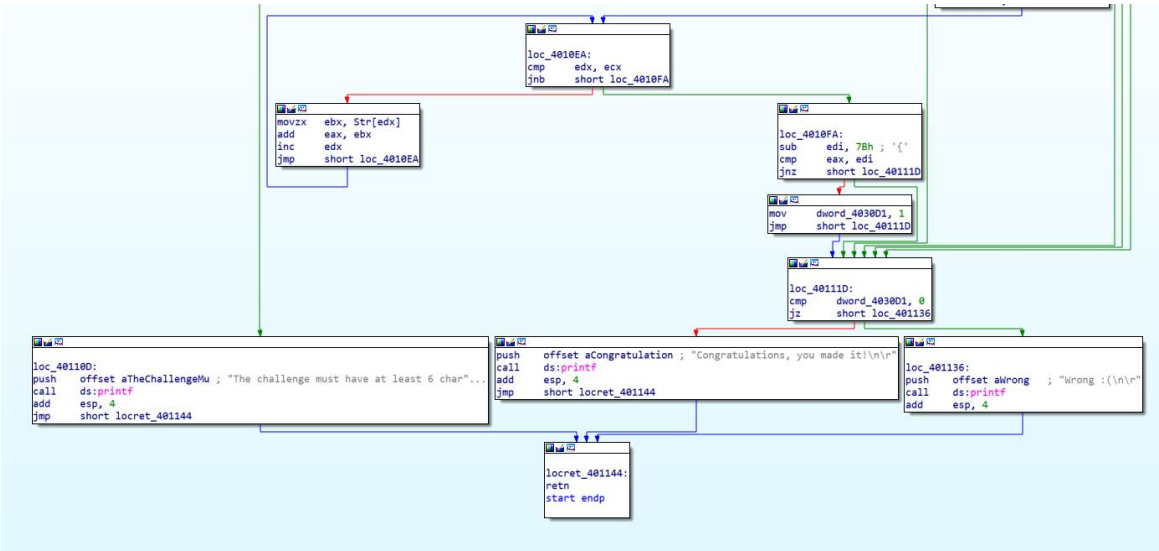
3.运行程序，根据提示输入字符串和逆向挑战的结果，获得“Congratulations，you made it！”输出。

三、 反汇编代码

```
.text:00401000
.text:00401000
.text:00401000                  public start
.text:00401000 start            proc near
.text:00401000                  push    offset Format   ; "Please enter a challenge: "
.text:00401005                  call    ds:printf
.text:0040100B                  add     esp, 4
.text:0040100E                  push    offset Str
.text:00401013                  push    offset aS       ; "%s"
.text:00401018                  call    ds:scanf
.text:0040101E                  add     esp, 8
.text:00401021                  push    offset Str      ; Str
.text:00401026                  call    ds:strlen
.text:0040102C                  add     esp, 4
.text:0040102F                  cmp     eax, 6
.text:00401032                  jb      loc_40110D
.text:00401038                  push    offset aPleaseEnterThe ; "Please enter the solution: "
.text:0040103D                  call    ds:printf
.text:00401043                  add     esp, 4
.text:00401046                  push    offset dword_4030AD
.text:0040104B                  push    offset dword_4030A9
.text:00401050                  push    offset dword_4030A5
.text:00401055                  push    offset dword_4030A1
.text:0040105A                  push    offset aUUUU    ; "%u-%u-%u-%u"
.text:0040105F                  call    ds:scanf
.text:00401065                  add     esp, 14h
.text:00401068                  cmp     eax, 4
.text:0040106B                  jb      loc_40111D
.text:00401071                  movzx   eax, byte_4030B2
.text:00401078                  movzx   ecx, byte_4030B4
.text:0040107F                  add     eax, ecx
.text:00401081                  movzx   ecx, byte_4030B5
.text:00401088                  add     eax, ecx
.text:0040108A                  cmp     eax, dword_4030A1
.text:00401090                  jnz     loc_40111D
.text:00401096                  mov     eax, dword_4030A5
.text:0040109B                  add     eax, 18h
.text:0040109E                  not     eax
.text:004010A0                  cmp     eax, 0BADF000Dh
```

```
.text:004010A5                jnz     short loc_40111D
.text:004010A7                mov     eax, dword_4030A9
.text:004010AC                mov     ecx, 0C48h
.text:004010B1                cdq
.text:004010B2                div     ecx
.text:004010B4                mov     esi, eax
.text:004010B6                movzx   eax, Str
.text:004010BD                movzx   ecx, byte_4030B3
.text:004010C4                mul     ecx
.text:004010C6                cmp     eax, esi
.text:004010C8                jnz     short loc_40111D
.text:004010CA                push    offset Str      ; Str
.text:004010CF                call    ds:strlen
.text:004010D5                add     esp, 4
.text:004010D8                mov     ecx, eax
.text:004010DA                sub     eax, eax
.text:004010DC                xor     edx, edx
.text:004010DE                mov     edi, dword_4030AD
.text:004010E4                xor     edi, 31337h
.text:004010EA
.text:004010EA loc_4010EA:                            ; CODE XREF: start+F8↓j
.text:004010EA                cmp     edx, ecx
.text:004010EC                jnb     short loc_4010FA
.text:004010EE                movzx   ebx, Str[edx]
.text:004010F5                add     eax, ebx
.text:004010F7                inc     edx
.text:004010F8                jmp     short loc_4010EA
.text:004010FA ; ---------------------------------------------------------------------------
.text:004010FA
.text:004010FA loc_4010FA:                            ; CODE XREF: start+EC↑j
.text:004010FA                sub     edi, 7Bh ; '{'
.text:004010FD                cmp     eax, edi
.text:004010FF                jnz     short loc_40111D
.text:00401101                mov     dword_4030D1, 1
.text:0040110B                jmp     short loc_40111D
.text:0040110D ; ---------------------------------------------------------------------------
```

```
.text:0040110D
.text:0040110D loc_40110D:                            ; CODE XREF: start+32↑j
.text:0040110D                push    offset aTheChallengeMu ; "The challenge must have at least 6 char"...
.text:00401112                call    ds:printf
.text:00401118                add     esp, 4
.text:0040111B                jmp     short locret_401144
.text:0040111D ; ---------------------------------------------------------------------------
.text:0040111D
.text:0040111D loc_40111D:                            ; CODE XREF: start+6B↑j
.text:0040111D                                        ; start+90↑j ...
.text:0040111D                cmp     dword_4030D1, 0
.text:00401124                jz      short loc_401136
.text:00401126                push    offset aCongratulation ; "Congratulations, you made it!\n\r"
.text:0040112B                call    ds:printf
.text:00401131                add     esp, 4
.text:00401134                jmp     short locret_401144
.text:00401136 ; ---------------------------------------------------------------------------
.text:00401136
.text:00401136 loc_401136:                            ; CODE XREF: start+124↑j
.text:00401136                push    offset aWrong   ; "Wrong :(\n\r"
.text:0040113B                call    ds:printf
.text:00401141                add     esp, 4
.text:00401144
.text:00401144 locret_401144:                         ; CODE XREF: start+11B↑j
.text:00401144                                        ; start+134↑j
.text:00401144                retn
.text:00401144 start          endp
.text:00401144
.text:00401144 ; ---------------------------------------------------------------------------
.text:00401145                align 100h
.text:00401200                dd 380h dup(?)
.text:00401200 _text          ends
.text:00401200
```



四、 逆向分析

```
.text:00401000                push    offset Format   ; "Please enter a challenge: "
.text:00401005                call    ds:printf
.text:0040100B                add     esp, 4
.text:0040100E                push    offset Str
.text:00401013                push    offset aS       ; "%s"
.text:00401018                call    ds:scanf
.text:0040101E                add     esp, 8
.text:00401021                push    offset Str      ; Str
.text:00401026                call    ds:strlen
.text:0040102C                add     esp, 4
.text:0040102F                cmp     eax, 6
.text:00401032                jb      loc_40110D
```

1.首先**获取 challenge**。栈顶指针 ESP 增加 4 个字节，将 scanf 函数参数入栈，即%s(表示读入字符串)和要读入的字符串 Str。随后栈顶指针 ESP 又增加 8 个字节，将读入的字符串入栈，调取 strlen 函数，获取长度存入 eax 寄存器中。将 eax 的值与 6 比较，即将字符串的长度与 6 比较。

如果长度小于 6，跳至地址 40110D 处继续执行，输出输入不合法的提示，然后跳至**地址 401144 处，程序结束**。**这里说明：输入的 challenge 必须至少六个字符！**

```
.text:0040110D loc_40110D:                            ; CODE XREF: start+32↑j
.text:0040110D                push    offset aTheChallengeMu ; "The challenge must have at least 6 char"...
.text:00401112                call    ds:printf
.text:00401118                add     esp, 4
.text:0040111B                jmp     short locret_401144
```

```
.text:00401144 locret_401144:                         ; CODE XREF: start+11B↑j
.text:00401144                                        ; start+134↑j
.text:00401144                retn
.text:00401144 start          endp
```

观察 **Str 的数据存放**：Challenge 第二、三、四、五字符的地址分别为 4030B2、4030B3、4030B4、4030B5.

```
.data:004030B1 Str            db 0                    ; DATA XREF: start+E↑o
.data:004030B1                                        ; start+21↑o ...
.data:004030B2 byte_4030B2    db 0                    ; DATA XREF: start+71↑r
.data:004030B3 byte_4030B3    db 0                    ; DATA XREF: start+BD↑r
.data:004030B4 byte_4030B4    db 0                    ; DATA XREF: start+78↑r
.data:004030B5 byte_4030B5    db 0                    ; DATA XREF: start+81↑r
.data:004030B6                db    0
.data:004030B7                db    0
.data:004030B8                db    0
.data:004030B9                db    0
.data:004030BA                db    0
.data:004030BB                db    0
.data:004030BC                db    0
.data:004030BD                db    0
.data:004030BE                db    0
.data:004030BF                db    0
.data:004030C0                db    0
.data:004030C1                db    0
.data:004030C2                db    0
.data:004030C3                db    0
.data:004030C4                db    0
.data:004030C5                db    0
.data:004030C6                db    0
```

2.如果长度大于等于 6，**获取 solution**。读入四个十进制数，中间以-相连。这四个数存放地址分别为 4030AD、4030A9、4030A5、4030A1。

```
.text:00401038                push    offset aPleaseEnterThe ; "Please enter the solution: "
.text:0040103D                call    ds:printf
.text:00401043                add     esp, 4
.text:00401046                push    offset dword_4030AD
.text:0040104B                push    offset dword_4030A9
.text:00401050                push    offset dword_4030A5
.text:00401055                push    offset dword_4030A1
.text:0040105A                push    offset aUUUU    ; "%u-%u-%u-%u"
.text:0040105F                call    ds:scanf
.text:00401065                add     esp, 14h
.text:00401068                cmp     eax, 4
.text:0040106B                jb      loc_40111D
```

判断是否读入的是 4 个数，如果不是，**跳至地址 40111D 处继续执行**。因为 4030D1 处的数据初始为 0，所以跳至 401136 地址处，输出错误提示。然后**跳至地址 401144 处，程序结束**。说明：**读入的 solotion 必须是四个数**！

```
.text:0040111D loc_40111D:                            ; CODE XREF: start+6B↑j
.text:0040111D                                        ; start+90↑j ...
.text:0040111D                cmp     dword_4030D1, 0
.text:00401124                jz      short loc_401136
.text:00401126                push    offset aCongratulation ; "Congratulations, you made it!\n\r"
.text:0040112B                call    ds:printf
.text:00401131                add     esp, 4
.text:00401134                jmp     short locret_401144
```

```
.text:00401136 loc_401136:                            ; CODE XREF: start+124↑j
.text:00401136                push    offset aWrong   ; "Wrong :(\n\r"
.text:0040113B                call    ds:printf
.text:00401141                add     esp, 4
```

3.如果读入的是 4 个数，接下来**判断四个数是否符合条件**。假设 challenge 为 123456：

(1)先判断第一个数：

通过一系列操作，**eax 的值为 challenge 的第二、四、五字符 ASCII 码值之和**。比较 **eax 与 solution 的第一个数**，如果不相等，**跳至地址 40111D 处继续执行**，输入的 solution 为错误的，**输出错误提示**。**然后跳至地址 401144 处，程序结束**。这说明：**solution 的第一个数是 challenge 的第二、四、五字符 ASCII 码值之和的十进制表示**。假设 challenge 为 123456，那么 solution 的第一个数为 32h+34h+35h 的十进制，即为 155.

```
.text:00401071                movzx   eax, byte_4030B2
.text:00401078                movzx   ecx, byte_4030B4
.text:0040107F                add     eax, ecx
.text:00401081                movzx   ecx, byte_4030B5
.text:00401088                add     eax, ecx
.text:0040108A                cmp     eax, dword_4030A1
.text:00401090                jnz     loc_40111D
```

(2)如果第一个数正确，继续判断第二个数：

将 **solution 的第二个数赋值给 eax，eax 加上 18h，取反，与 0BADF00Dh 比较**。

如果不相等，**跳至地址 40111D 处继续执行，输入的 solution 为错误的**，输出错误提示。然后**跳至地址 401144 处，程序结束**。所以如果输入正确的话，最终结果应该与 0BADF00Dh 相等。

对这个过程逆向分析计算：**先将 0BADF00Dh 取反得到 4520FFF2h，然后减去 18h，得到 4520FFDAh，转换为十进制为 1159790554**。所以，**solution 的第二个数是 1159790554.**

```
.text:00401096                  mov     eax, dword_4030A5
.text:0040109B                  add     eax, 18h
.text:0040109E                  not     eax
.text:004010A0                  cmp     eax, 0BADF000Dh
.text:004010A5                  jnz     short loc_40111D
```

(3)如果第二个数也正确，继续判断第三个数：

①将 **solution 的第三个数赋值给 eax，ecx 赋值为 0C48h，eax 的第 31bit 复制到 edx 每个 bit 中，即把 edx 的所有位都设成 eax 最高位的值，将一个 32 位有符号数扩展为 64 位有符号数。**

②**edx:eax 除以 ecx，余数存至 edx，商存至 eax，eax 赋值给 esi。**

③**将 challenge 的第一个字符无符号扩展至 eax，将 challenge 的第三个字符无符号扩展至 ecx，eax 乘 ecx，结果存至 edx:eax。**

④**比较 eax 与 esi**，如果不相等，**跳至地址 40111D 处继续执行，输入的 solution 为错误的**，输出错误提示。然后**跳至地址 401144 处，程序结束**。所以如果输入正确的话，二者应该相等。

对这个过程逆向分析计算：**将 challenge 第一个字符 ASCII 码赋值给 eax，将 challenge 第三个字符 ASCII 码赋值给 ecx。计算 eax 与 ecx 的乘积，取后八位 eax。eax 乘 0C48h，转换为十进制即为 solution 的第三个数**。假设 challenge 为 123456，计算 31h 与 33h 乘积，结果为 9C3，再乘 0C48h，结果为 77E2D8h，转换为十进制为 7856856，此为 solution 的第三个数。

```
.text:004010A7                  mov     eax, dword_4030A9
.text:004010AC                  mov     ecx, 0C48h
.text:004010B1                  cdq
.text:004010B2                  div     ecx
.text:004010B4                  mov     esi, eax
.text:004010B6                  movzx   eax, Str
.text:004010BD                  movzx   ecx, byte_4030B3
.text:004010C4                  mul     ecx
.text:004010C6                  cmp     eax, esi
.text:004010C8                  jnz     short loc_40111D
```

(4)如果第三个数也正确，继续判断第四个数：

①**eax 值为 challenge 的长度，赋值给 ecx。**

②eax 减去自身变为 0，edx 与自己异或也变为 0。

③将 solution 的第四个数赋值给 edi，与 31337h 异或。

④比较 edx 与 ecx。

⑤如果不相等，将 challenge 的第 edx 个字符无符号扩展赋值给 ebx，ebx 加至 eax，edx 加一，然后跳转回地址 3010EA 处，即上述④处 edx 与 ecx 比较。

⑥循环上述，直到 edx 与 ecx 相等。

⑦相等时，跳至地址 4030AD 处。

⑧将 edi 减去 7Bh，比较 eax 与 edi，如果不相等，**跳至地址 40111D 处继续执行，输入的 solution 为错误的**，输出错误提示。然后**跳至地址 401144 处，程序结束**。

⑨如果相等，将 4030D1 处的数据修改为 1，跳至地址 40111D 处继续执行。在之前的运行 40111D 处程序时，4030D1 处的数据初始为 0，所以输出错误。而本次，其值不为 0，输出正确提示，程序结束。

对上述过程逆向分析计算：**eax 的值为 challenge 的所有字符 ASCII 码值加和。eax 加上 7Bh，是第四个数与 31337h 异或的结果。由此可以求得第四个数**。假设 challenge 为 123456，eax 值为 135h，值为 1B0.1B0 是 solution 的第四个数的 16 进制与 31337h 的异或值。求得第四个数的十进制数为 201351.

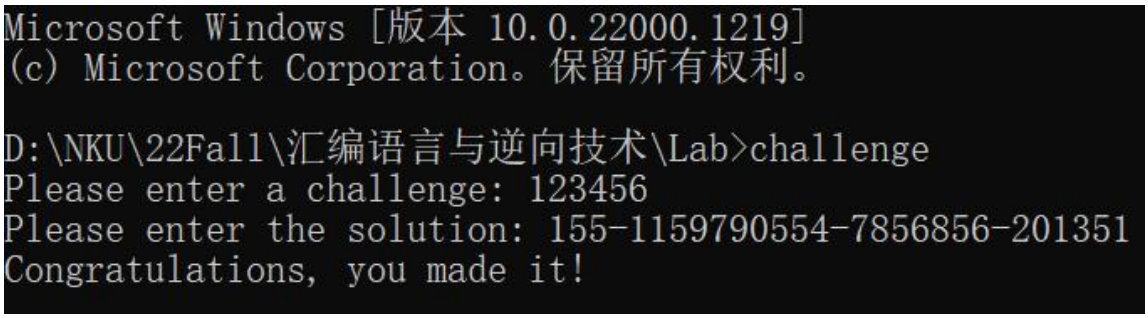
```
.text:004010CA                push    offset Str      ; Str
.text:004010CF                call    ds:strlen
.text:004010D5                add     esp, 4
.text:004010D8                mov     ecx, eax
.text:004010DA                sub     eax, eax
.text:004010DC                xor     edx, edx
.text:004010DE                mov     edi, dword_4030AD
.text:004010E4                xor     edi, 31337h
.text:004010EA
.text:004010EA loc_4010EA:                            ; CODE XREF: start+F8↓j
.text:004010EA                cmp     edx, ecx
.text:004010EC                jnb     short loc_4010FA
.text:004010EE                movzx   ebx, Str[edx]
.text:004010F5                add     eax, ebx
.text:004010F7                inc     edx
.text:004010F8                jmp     short loc_4010EA
```

```
.text:004010FA loc_4010FA:                            ; CODE XREF: start+EC↑j
.text:004010FA                sub     edi, 7Bh ; '{'
.text:004010FD                cmp     eax, edi
.text:004010FF                jnz     short loc_40111D
.text:00401101                mov     dword_4030D1, 1
.text:0040110B                jmp     short loc_40111D
```

```
.text:0040111D loc_40111D:                            ; CODE XREF: start+6B↑j
.text:0040111D                                        ; start+90↑j ...
.text:0040111D                cmp     dword_4030D1, 0
.text:00401124                jz      short loc_401136
.text:00401126                push    offset aCongratulation ; "Congratulations, you made it!\n\r"
.text:0040112B                call    ds:printf
.text:00401131                add     esp, 4
.text:00401134                jmp     short locret_401144
```

(5)四个数都正确，输出正确提示，程序结束。

五、 实验截图


Microsoft Windows [版本 10.0.22000.1219]
(c) Microsoft Corporation。保留所有权利。
D:\NKU\22Fall\汇编语言与逆向技术\Lab>challenge
Please enter a challenge: 123456
Please enter the solution: 155-1159790554-7856856-201351
Congratulations, you made it!