南開大學
Nankai University

计算机学院

汇编语言与逆向技术实验报告

Lab6 Reverse Engineering Challenge

姓名：杨冰雪

学号：2110508

专业：计算机科学与技术

2023 年 12 月 11 日

目录

1 实验目的

1. 熟悉静态反汇编工具 IDA Pro；
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

2 实验原理

1. 通过 IDA 得到二进制代码的反汇编代码。
2. 不修改二进制代码，分析汇编代码的计算过程、条件判断、分支结构等信息，逆向推理出程序的正确输入数据，完成逆向分析挑战。

3 实验内容

3.1 反汇编代码

实验 IDA Freeware 得到的 ELF 文件的反汇编代码截图如下：

- 反汇编代码的图形化显示

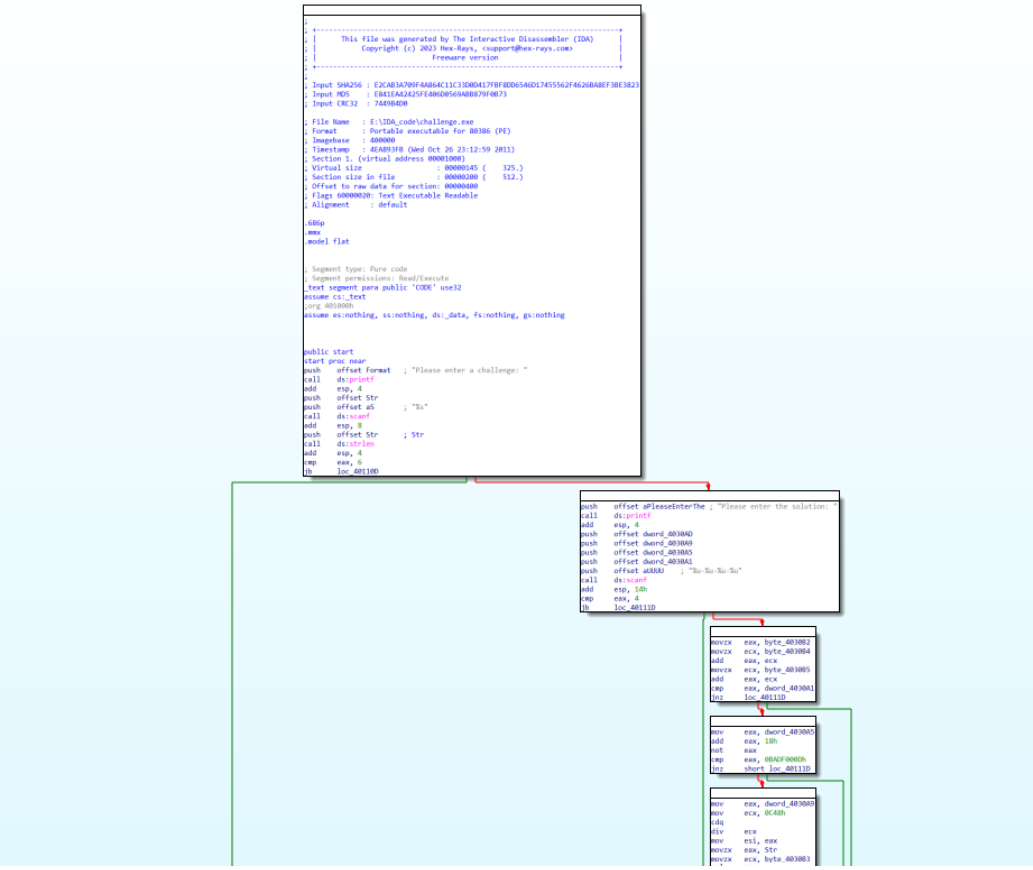

图 3.1: 反汇编代码的图形化显示

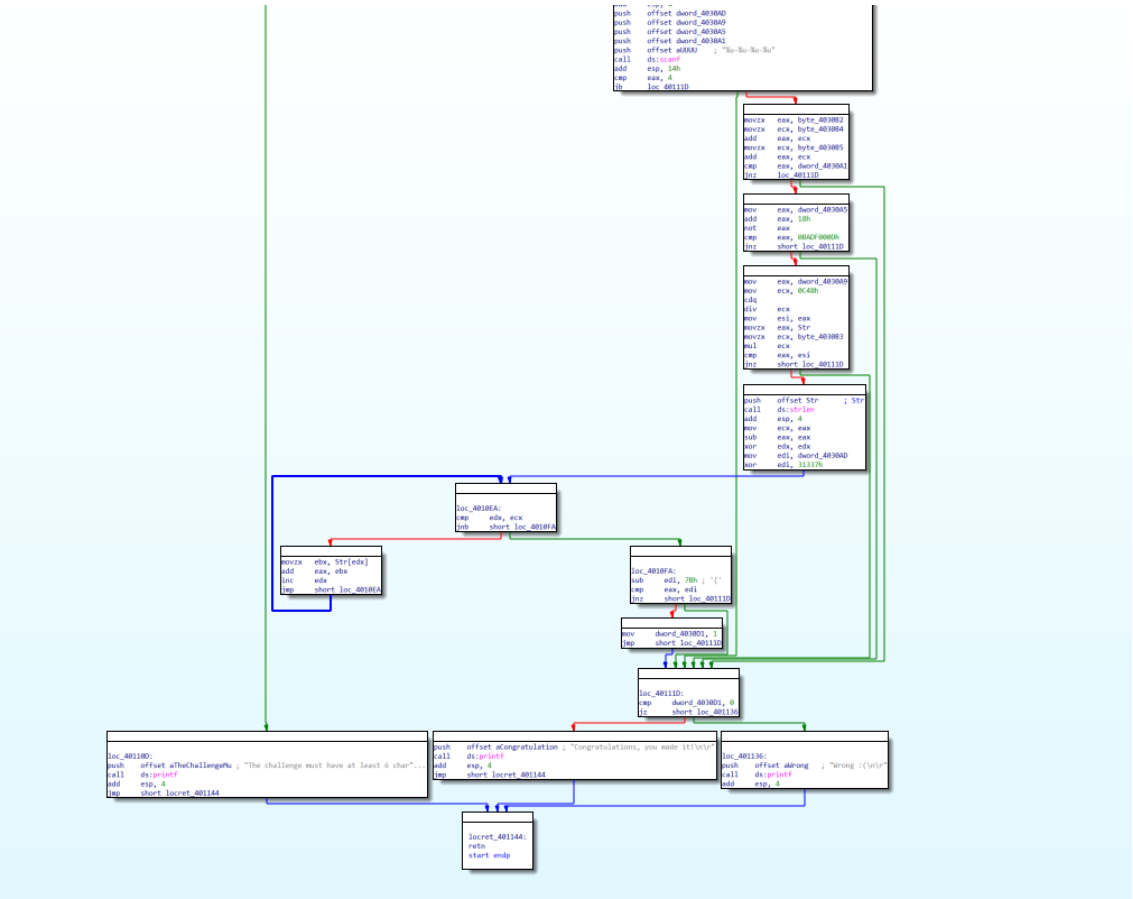

图 3.2: 反汇编代码的图形化显示

- 反汇编代码

```
.text:00401000
.text:00401000 ; =============== S U B R O U T I N E =======================================
.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
```

图 3.3: 反汇编代码

```
.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
00004DE 004010DE: start+DE (Synchronized with Hex View-1)
```

图 3.4: 反汇编代码

```
.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
.idata:00402000 ; Section 2. (virtual address 00002000)
.idata:00402000 ; Virtual size                : 00000070 (    112.)
.idata:00402000 ; Section size in file        : 00000200 (    512.)
```

图 3.5: 反汇编代码

```
.data:00403000 ; Segment permissions: Read/Write
.data:00403000 _data           segment para public 'DATA' use32
.data:00403000                 assume cs:_data
.data:00403000                 ;org 403000h
.data:00403000 ; char Format[]
.data:00403000 Format          db 'Please enter a challenge: ',0
.data:00403000                                         ; DATA XREF: start↑o
.data:0040301B ; char aS[]
.data:0040301B aS              db '%s',0               ; DATA XREF: start+13↑o
.data:0040301E ; char aTheChallengeMu[]
.data:0040301E aTheChallengeMu db 'The challenge must have at least 6 characters',0Ah
.data:0040301E                                         ; DATA XREF: start:loc_40110D↑o
.data:0040304C                 db 0Dh,0
.data:0040304E ; char aPleaseEnterThe[]
.data:0040304E aPleaseEnterThe db 'Please enter the solution: ',0
.data:0040304E                                         ; DATA XREF: start+38↑o
.data:0040306A ; char aUUUU[]
.data:0040306A aUUUU           db '%u-%u-%u-%u',0      ; DATA XREF: start+5A↑o
.data:00403076 ; char aWrong[]
.data:00403076 aWrong          db 'Wrong :(',0Ah       ; DATA XREF: start:loc_401136↑o
.data:0040307F                 db 0Dh,0
.data:00403081 ; char aCongratulation[]
.data:00403081 aCongratulation db 'Congratulations, you made it!',0Ah
.data:00403081                                         ; DATA XREF: start+126↑o
.data:0040309F                 db 0Dh,0
.data:004030A1 dword_4030A1    dd 0                    ; DATA XREF: start+55↑o
.data:004030A1                                         ; start+8A↑r
.data:004030A5 dword_4030A5    dd 0                    ; DATA XREF: start+50↑o
.data:004030A5                                         ; start+96↑r
.data:004030A9 dword_4030A9    dd 0                    ; DATA XREF: start+4B↑o
.data:004030A9                                         ; start+A7↑r
.data:004030AD dword_4030AD    dd 0                    ; DATA XREF: start+46↑o
.data:004030AD                                         ; start+DE↑r
.data:004030B1 ; char Str
.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
```

图 3.6: 反汇编代码

3.2 逆向分析

3.2.1 数据结构

1. 将最开始写入 challenge 的 6 个字符型数存入内存 004030B1-004030B6

```
.data:004030B1 ; char Str
.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
```

图 3.7: 数据结构

2. 将写入 solution 的 4 个 dword 双字型数分别存入内存 004030A1、004030A5、004030A9、004030AD

```
.data:004030A1 dword_4030A1    dd 0                    ; DATA XREF: start+55↑o
.data:004030A1                                         ; start+8A↑r
.data:004030A5 dword_4030A5    dd 0                    ; DATA XREF: start+50↑o
.data:004030A5                                         ; start+96↑r
.data:004030A9 dword_4030A9    dd 0                    ; DATA XREF: start+4B↑o
.data:004030A9                                         ; start+A7↑r
.data:004030AD dword_4030AD    dd 0                    ; DATA XREF: start+46↑o
.data:004030AD                                         ; start+DE↑r
```

图 3.8: 数据结构

3.2.2 分支结构与条件判断

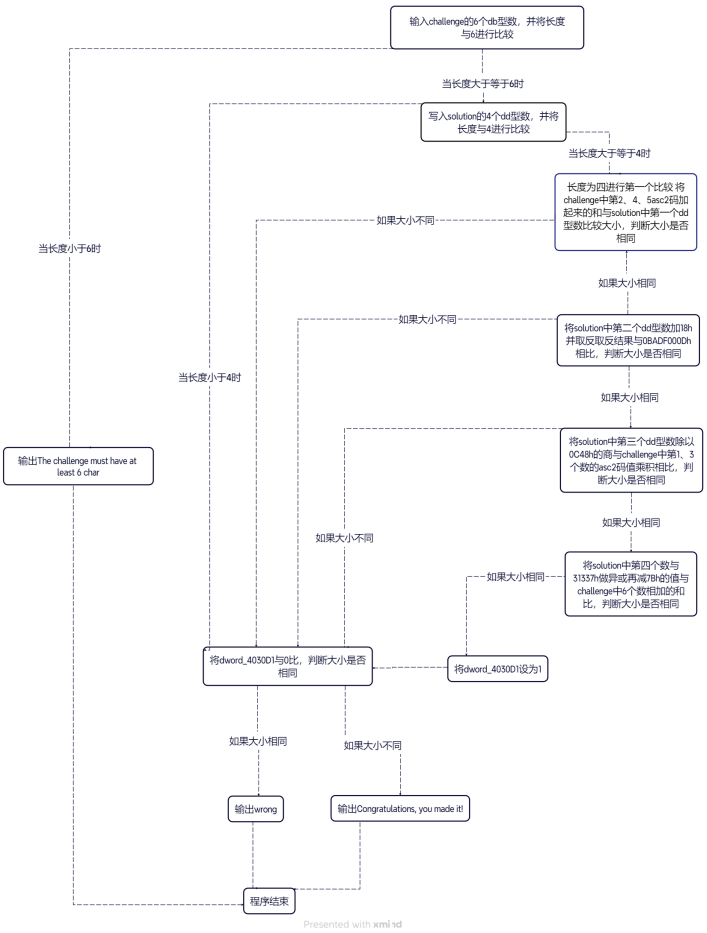

图 3.9: 代码流程图

3.2.3 计算过程

首先需要确定 chellenge 输入的六个数，为了简单起见，我选择输入为 111111。

第一个%u

- 先将用户第一次输入的字符串 Str 的第 2 位的 ASCII 码存入寄存器 eax
- 再将 Str 的第 4 位的 ASCII 码存入寄存器 ecx
- 将 eax 与 ecx 相加（2、4 位 ASCII 相加）
- 将 ecx 与 Str 的第 5 位的 ASCII 码相加
- 将 ecx 相加之后的结果加到 eax 上（此时 eax 就是 2、4、5 位 ASCII 之和）
- 比较第一个%u 与 eax 的大小

计算得出，第一个数的值为 147

第二个%u

- 首先，让第二个%u 加上 18h
- 然后对其取反
- 把取反后的结果与 0BADF000Dh 比较

  计算得出，第二个数值为 1159790554

第三个%u

- 先让第三个%u 除以 0C48h，存入 esi 里
- 然后令 Str 的第 1 位和第 3 位的 ASCII 相乘存入 eax 里
- 比较 esi (第三个%u 除以 0C48h) 和 eax（Str 的第 1 位和第 3 位的 ASCII 相乘）

  计算得出，第三个数值为 7548744

第四个%u

- 将第四个%u 与 31337h 异或，存入 edi
- 按照 Str 的位数进行循环（共 6 位），并将 edx 初始置 0，作为循环变量
- 将每次循环中的 Str[edx] 加到 eax（eax 存放 Str 六位的 ASCII 之和）
- 将 edi（第四个%u 与 31337h 异或）的值减去 7Bh 与 eax（Str 六位的 ASCII 之和）比较

  计算得出，第四个数值为 201366

4 实验结果

在程序中输入最后计算的结果，获得“Congratulations，you made it!”的输出，说明计算正确。

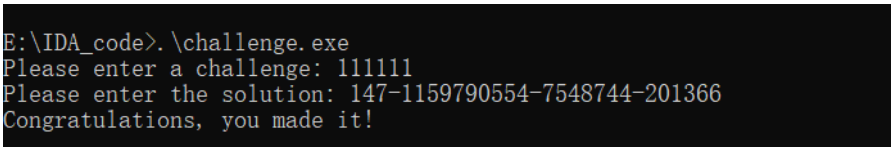

图 4.10: 运行结果

5 实验总结

通过本次实验，使我掌握了 ida 的使用方法和了解了反汇编代码分析的基本步骤，学会用 ida 来自动分析二进制文件，学会通过查看伪代码视图，来更容易地理解反汇编的代码。