南开大学

RE Challenge

（汇编语言与逆向技术实验 7）

姓名：申宗尚

学号：2213924

专业：信息安全

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．实验环境

1、ida64

2、challenge.exe

三．实验原理

通过 IDA Freeware 可以得到二进制代码的反汇编代码，如图 1 和图 2 所示。

```
.text:00401000 ; ===========================================================================
.text:00401000
.text:00401000 ; Segment type: Pure code
.text:00401000 ; Segment permissions: Read/Execute
.text:00401000 _text           segment para public 'CODE' use32
.text:00401000                 assume cs:_text
.text:00401000                 ;org 401000h
.text:00401000                 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
.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 word_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 ptr word_4030A1
.text:00401090                 jnz     loc_40111D
00000400 00401000: start
```

图 1　challenge.exe 的反汇编代码

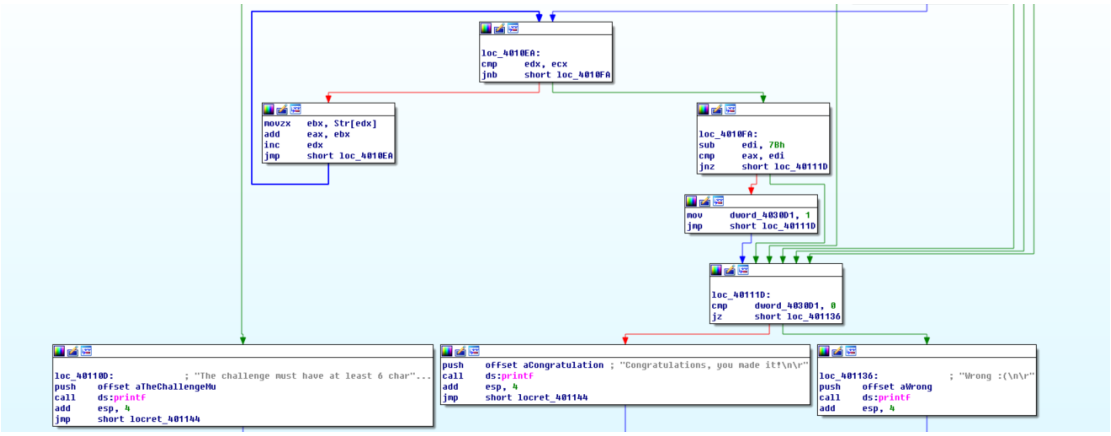


图 2 challenge.exe 的反汇编代码的图形化显示

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。

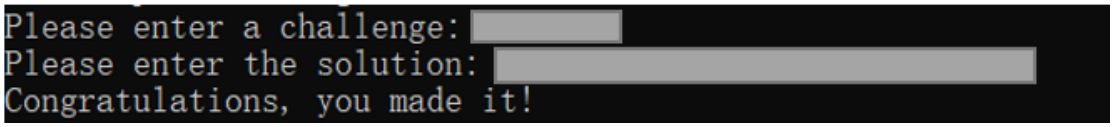
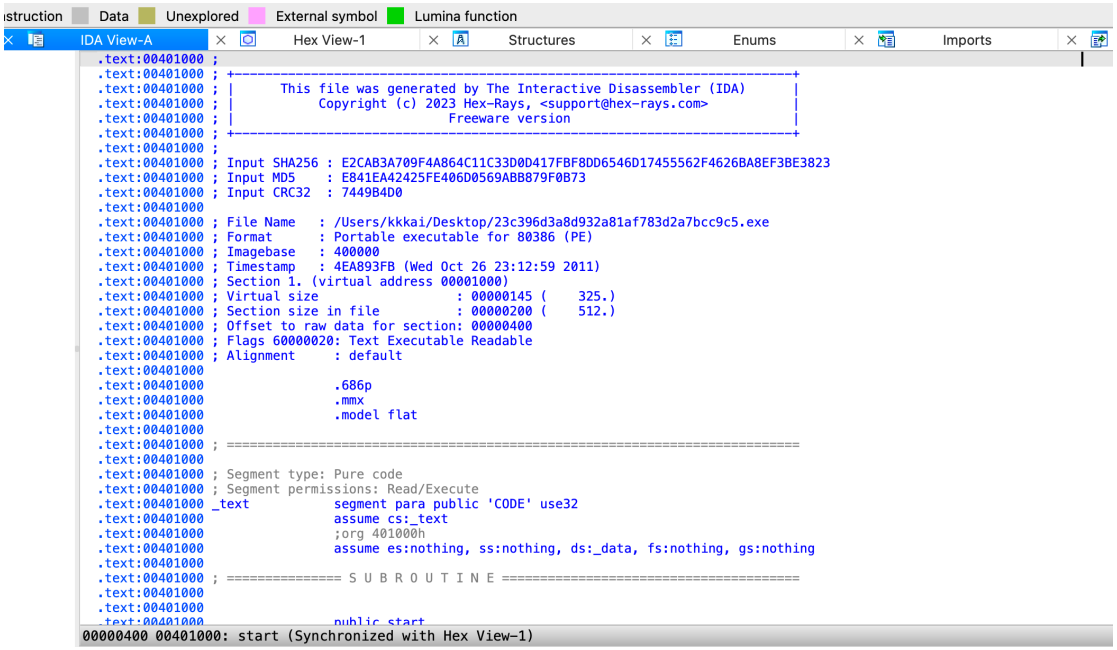
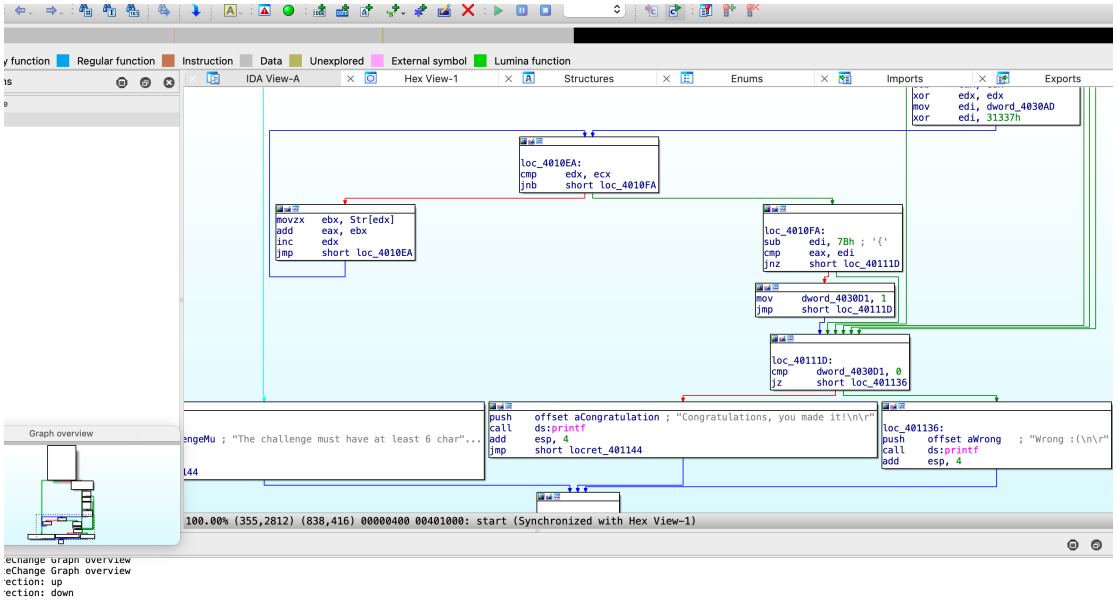


图 3 逆向分析，完成挑战

四．实验内容

如图，将 challenge.exe 导入 ida64 Freeware 中，可以得到二进制代码的反汇编代码

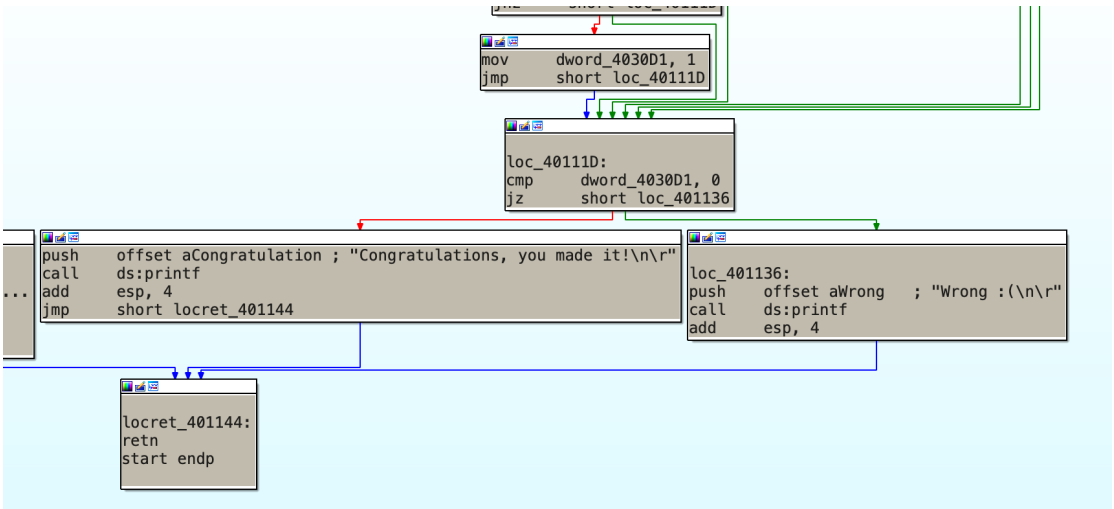




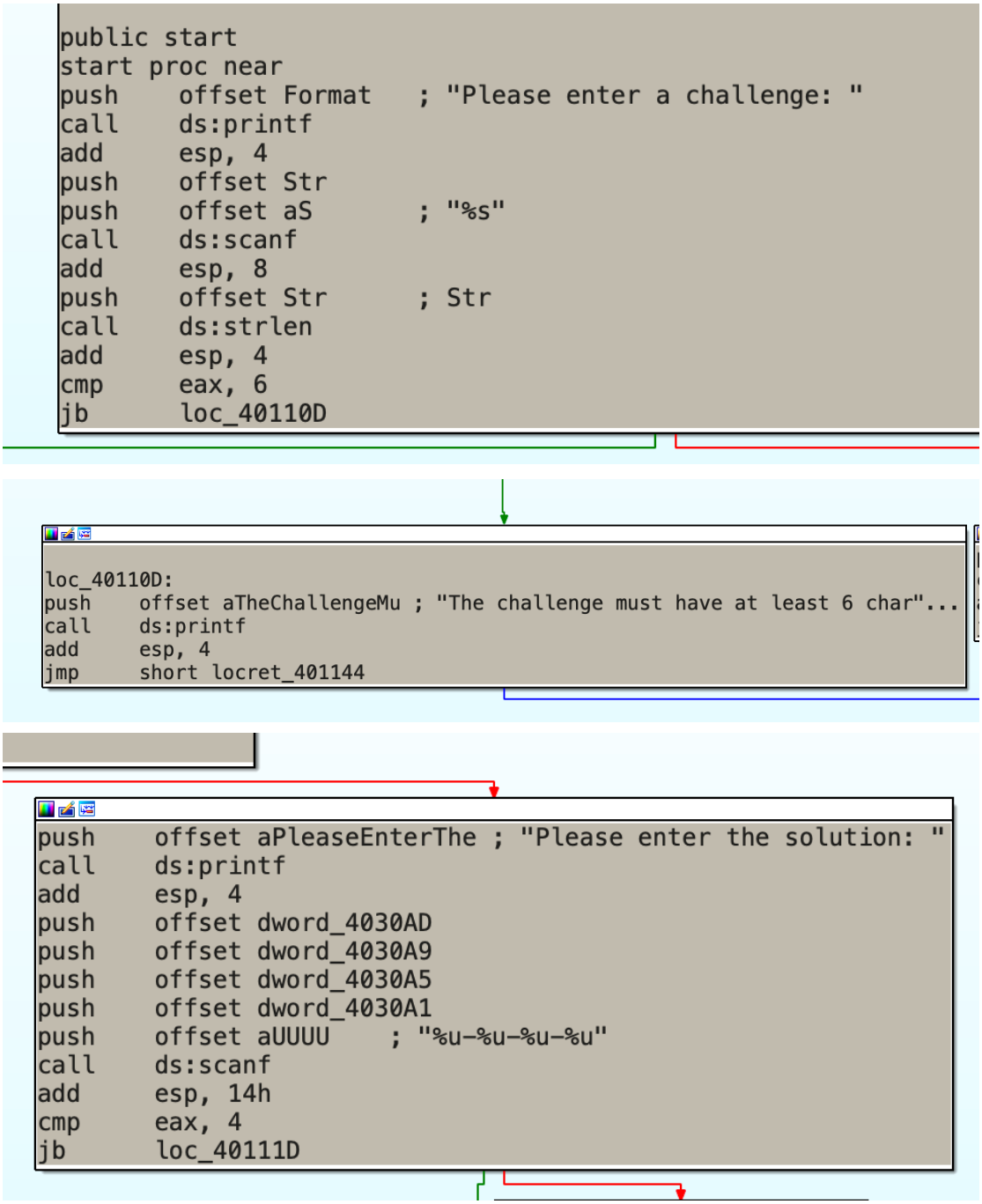
源代码见附件。

从而，我们可以通过对该反汇编代码的计算过程、数据结构、条件判断、分支结构等信息进行分析，逆向推理出程序的正确输入数据，完成逆向分析挑战。

由代码分析可知，要实现目标字符串“Congratulations, you made it!”的输出，需要满足指向该输出代码块所在代码块的条件



寻本溯源，在程序的最开始的输入代码块中，需要输入一个字符串：



该输入代码块实现的功能有：

1、 调用 printf 函数，输出“Please enter a challenge”，提示用户输入一个字符串

2、 调用 scanf 函数，将用户输入的字符串存入 Str 中

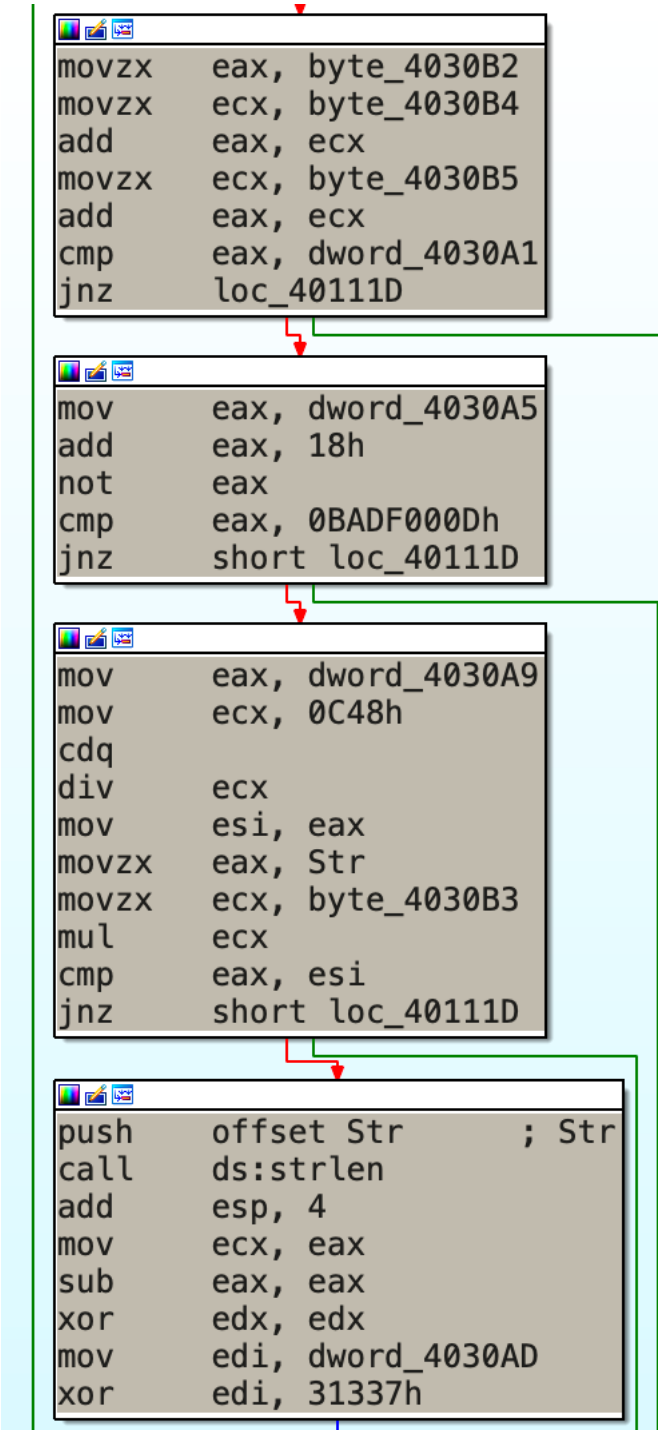
3、 调用 strlen 函数，获取 Str 中字符串的长度，将其与数字 6 比较，如果长度大于等于 6,则进入下一段代码块,否则提示输出“The challenge must have at least 6 char…”

4、 从而得知，该程序需要输入一个长度至少为 6 的字符串，故本程序选择 000000，作为测试输入。

5、 在成功输入 000000 作为 Str 后,跳转到下一代码块,该代码块首先调用了 printf 函数，

输出字符串“Please enter the solution：”，提示用户输入测试 challenge 的答案

6、 随后调用 scanf 函数，将用户输入的“%u-%u-%u-%u”型字符串（%u 为无符号整数）依次存入 dword_4030A1、 dword_4030A5、 dword_4030A9、 dword_4030AD 为起始的数据段，以进行下一步操作。



7、 在其次的操作中，依次对前后四位%u 的每一个进行正确性的检验。

8、 在第一个代码块中，将用户输入的 Str 的第 2 位的 ASCII 码存入 eax，再将 Str 的第 4

位存入 ecx，随后将 eax 与 ecx 相加（结果存在 eax），将 Str 的第 5 位存入 ecx，再将 ecx 与 eax 相加存入 eax，此时，eax 的结果是第 2，4，5 位 ASCII 码相加的和，程序将其与第一个%u 进行比较，从而可以得出，答案的第一部分应该是 2，4，5 位 ASCII 码和对于字符串 000000，48+48+48=144

从而可以得出，答案第一个数值应该是 144

9、 在第二个代码块中，首先让第二个%u 数据存入 eax，加 18h，再将其取反，随后与数据 0BADF000Dh 做比较，相等进入下一位比较。从而可以反推出：第二个数据位应该是 0BADF000Dh 取反后-18h 得到的结果，计算过程如下

0BADF000Dh=1011 1010 1101 1111 0000 0000 0000 1101

对其取反得到 0100 0101 0010 0000 1111 1111 1111 0010

将其转为 16 进制得到 4520FFF2

将其-18h 得到 4520FFDA

将其转为 10 进制，得到 1159790554

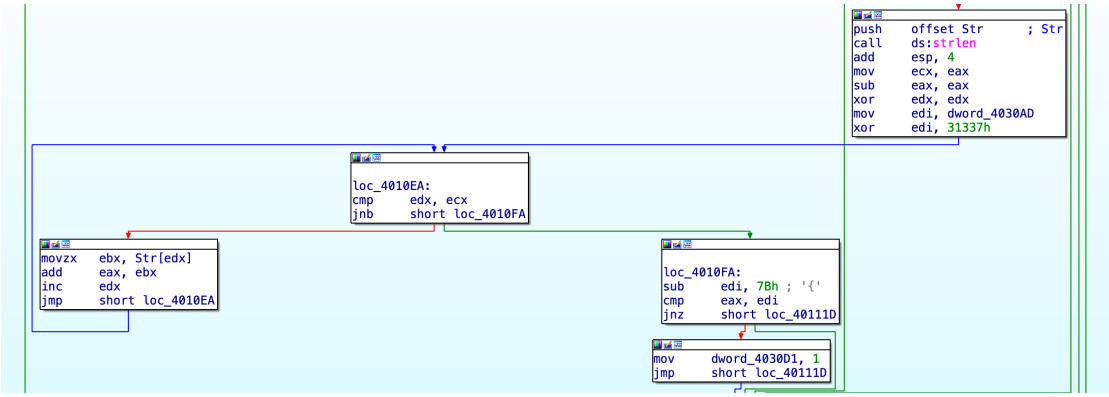
从而可知，第二个数据应该是 1159790554

10、在第三个代码块中，将第三个%u 数据存入 eax 中，除以 ecx 中存储的 0C48h，将结果存入 esi 中，然后将 Str 的第 1 位和第 3 位分别存入 eax，ecx，相乘结果存入 eax，最后将 eax 与 esi 相比，如果相等程序继续进行，从而可以推知

%u/0C48h=Str[1]*Str[3]

即%u=Str[1]*Str[3]*0C48h

从而对于 000000，%u=48*48*3144=7243776



11、在第四个代码块中，首先先将各个寄存器清零，再将第四个%u 存入 edi，与 31337h 异或。随后按照 Str 的位数进行循环（对于 000000，共循环 6 次，edx 作为循环变量）。每次循环时，将循环的 Str[edx]加入 eax 中，最后，eax 的值应该是 Str 的所有字符的 ASCII 码

之和。最后，将 edi 减去 7Bh，与 eax 比较，如果相等，则继续进行，从而可以推知：

edi-7Bh=eax

即（%u xor 31337h）-7Bh = eax（ASCII SUM）

从而，%u 应该等于 Str 的所有字符 ASCII 之和+7Bh，再与 31337h 异或，计算过程如下：

对于 000000，eax=48+48+48+48+48+48=288

288+7Bh=411=110011011

110011011 xor 31337h=110011011 xor 1100010011100110111=110001001010101100

=201388

综上，第四个数值为 201388

12、由以上分析，可以得出，对于输入字符串 Str=000000，所对应的 solution“%u-%u-%u-%u”应该为 144-1159790554-7243776-201388，下为程序运行截图：

再对输入字符串 Str=abcdef 进行示例，所对应的 solution“%u-%u-%u-%u”由上述的分析，经计算应该为 299-1159790554-30191832-201191，下为程序运行截图：

五、附件（challenge.exe 源代码）

```
.text:00401000 ;
    .text:00401000 ; +-------------------------------------------------------------------------+
    .text:00401000 ; |      This file was generated by The Interactive Disassembler (IDA)      |
    .text:00401000 ; |           Copyright (c) 2023 Hex-Rays, <support@hex-rays.com>           |
    .text:00401000 ; |                            Freeware version                             |
    .text:00401000 ; +-------------------------------------------------------------------------+
.text:00401000 ;
    .text:00401000 ; Input
    SHA256 :E2CAB3A709F4A864C11C33D0D417FBF8DD6546D17455562F4626BA8EF3BE3823
        .tet:00401000 ; Input MD5    : E841EA42425FE406D0569ABB879F0B73
        .txt:00401000 ; Input CRC32  : 7449B4D0
        .ext:00401000
```

```
        text:00401000 ; File Name   : /Users/kkkai/Desktop/23c396d3a8d932a81af783d2a7bcc9c5.exe
.text:00401000 ; Format      : Portable executable for 80386 (PE)
.text:00401000 ; Imagebase   : 400000
.text:00401000 ; Timestamp   : 4EA893FB (Wed Oct 26 23:12:59 2011)
.text:00401000 ; Section 1. (virtual address 00001000)
.text:00401000 ; Virtual size                  : 00000145 (    325.)
.text:00401000 ; Section size in file          : 00000200 (    512.)
.text:00401000 ; Offset to raw data for section: 00000400
.text:00401000 ; Flags 60000020: Text Executable Readable
.text:00401000 ; Alignment     : default
.text:00401000
.text:00401000                 .686p
.text:00401000                 .mmx
.text:00401000                 .model flat
.text:00401000
    .text:00401000 ; ===========================================================================
.text:00401000
.text:00401000 ; Segment type: Pure code
.text:00401000 ; Segment permissions: Read/Execute
.text:00401000 _text           segment para public 'CODE' use32
.text:00401000                 assume cs:_text
.text:00401000                 ;org 401000h
    .text:00401000                 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
.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
.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
    atleast 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
    madeit!\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.)
.idata:00402000 ; Offset to raw data for section: 00000600
.idata:00402000 ; Flags 40000040: Data Readable
.idata:00402000 ; Alignment     : default
.idata:00402000 ;
.idata:00402000 ; Imports from msvcrt.dll
.idata:00402000 ;
.idata:00402000 ; ===========================================================================
.idata:00402000
.idata:00402000 ; Segment type: Externs
.idata:00402000 ; _idata
.idata:00402000 ; int (*scanf)(const char *const Format, ...)
.idata:00402000                 extrn scanf:dword       ; CODE XREF: start+18↑p
.idata:00402000                                         ; start+5F↑p
.idata:00402000                                         ; DATA XREF: ...
.idata:00402004 ; size_t (__cdecl *strlen)(const char *Str)
.idata:00402004                 extrn strlen:dword      ; CODE XREF: start+26↑p
.idata:00402004                                         ; start+CF↑p
.idata:00402004                                         ; DATA XREF: ...
.idata:00402008 ; int (*printf)(const char *const Format, ...)
.idata:00402008                 extrn printf:dword      ; CODE XREF: start+5↑p
.idata:00402008                                         ; start+3D↑p ...
.idata:0040200C
.idata:0040200C
.rdata:00402010 ; ===========================================================================
.rdata:00402010
.rdata:00402010 ; Segment type: Pure data
.rdata:00402010 ; Segment permissions: Read
.rdata:00402010 _rdata          segment para public 'DATA' use32
.rdata:00402010                 assume cs:_rdata
.rdata:00402010                 ;org 402010h
.rdata:00402010 __IMPORT_DESCRIPTOR_msvcrt dd rva off_402038 ; Import Name Table
.rdata:00402014                 dd 0                    ; Time stamp
.rdata:00402018                 dd 0                    ; Forwarder Chain
.rdata:0040201C                 dd rva aMsvcrtDll       ; DLL Name
.rdata:00402020                 dd rva scanf            ; Import Address Table
.rdata:00402024                 db    0
.rdata:00402025                 db    0
.rdata:00402026                 db    0
```

```
.rdata:00402027                 db    0
.rdata:00402028                 db    0
.rdata:00402029                 db    0
.rdata:0040202A                 db    0
.rdata:0040202B                 db    0
.rdata:0040202C                 db    0
.rdata:0040202D                 db    0
.rdata:0040202E                 db    0
.rdata:0040202F                 db    0
.rdata:00402030                 db    0
.rdata:00402031                 db    0
.rdata:00402032                 db    0
.rdata:00402033                 db    0
.rdata:00402034                 db    0
.rdata:00402035                 db    0
.rdata:00402036                 db    0
.rdata:00402037                 db    0
.rdata:00402038 ;
.rdata:00402038 ; Import names for msvcrt.dll
.rdata:00402038 ;
    .rdata:00402038 off_402038      dd rva word_402052      ;
    DATAXREF: .rdata:__IMPORT_DESCRIPTOR_msvcrt↑o
.rdata:0040203C                 dd rva word_40205A
.rdata:00402040                 dd rva word_402048
.rdata:00402044                 dd 0
.rdata:00402048 word_402048     dw 281h                 ; DATA XREF: .rdata:00402040↑o
.rdata:0040204A                 db 'printf',0
.rdata:00402051                 align 2
.rdata:00402052 word_402052     dw 28Eh                 ; DATA XREF: .rdata:off_402038↑o
.rdata:00402054                 db 'scanf',0
.rdata:0040205A word_40205A     dw 2A1h                 ; DATA XREF: .rdata:0040203C↑o
.rdata:0040205C                 db 'strlen',0
.rdata:00402063                 align 4
.rdata:00402064 aMsvcrtDll      db 'msvcrt.dll',0       ; DATA XREF: .rdata:0040201C↑o
.rdata:0040206F                 align 1000h
.rdata:0040206F _rdata          ends
.rdata:0040206F
.data:00403000 ; Section 3. (virtual address 00003000)
.data:00403000 ; Virtual size                  : 000000D5 (    213.)
.data:00403000 ; Section size in file          : 00000200 (    512.)
.data:00403000 ; Offset to raw data for section: 00000800
.data:00403000 ; Flags C0000040: Data Readable Writable
.data:00403000 ; Alignment     : default
    .data:00403000 ; ===========================================================================
```

```
.data:00403000
.data:00403000 ; Segment type: Pure data
.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
.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
.data:004030C7                 db    0
.data:004030C8                 db    0
.data:004030C9                 db    0
.data:004030CA                 db    0
.data:004030CB                 db    0
.data:004030CC                 db    0
.data:004030CD                 db    0
.data:004030CE                 db    0
.data:004030CF                 db    0
.data:004030D0                 db    0
.data:004030D1 dword_4030D1    dd 0                    ; DATA XREF: start+101↑w
.data:004030D1                                         ; start:loc_40111D↑r
.data:004030D5                 align 1000h
.data:004030D5 _data           ends
.data:004030D5
.data:004030D5
.data:004030D5                 end start
```