

汇编语言与逆向技术实验报告

Lab5- Reverse Engineering Challenge

学号：2112060 姓名：孙蒨 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、实验原理

1. 通过 IDA 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 ; ===== SUBROUTINE =====
.text:00401000
.text:00401000 public start
.text:00401000 proc near
.text:00401000 start
.text:00401000     push offset Format      ; "Please enter a challenge: "
.text:00401005     call ds:printf
.text:00401008     add     esp, 4
.text:0040100E     push offset Str
.text:00401013     push offset aS      ; "%5s"
.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_4030A0
.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     ecx, ecx
.text:00401081     movzx  ecx, byte_4030B5
.text:00401088     add     ecx, 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

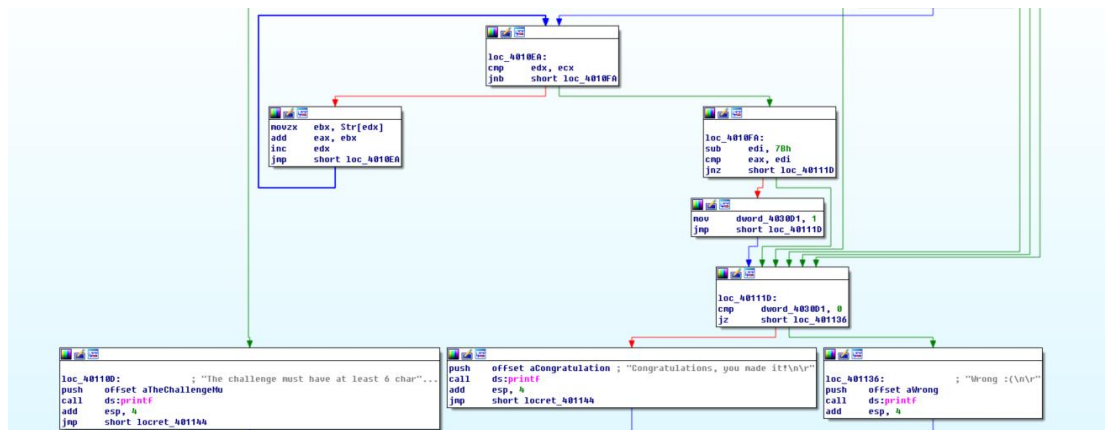


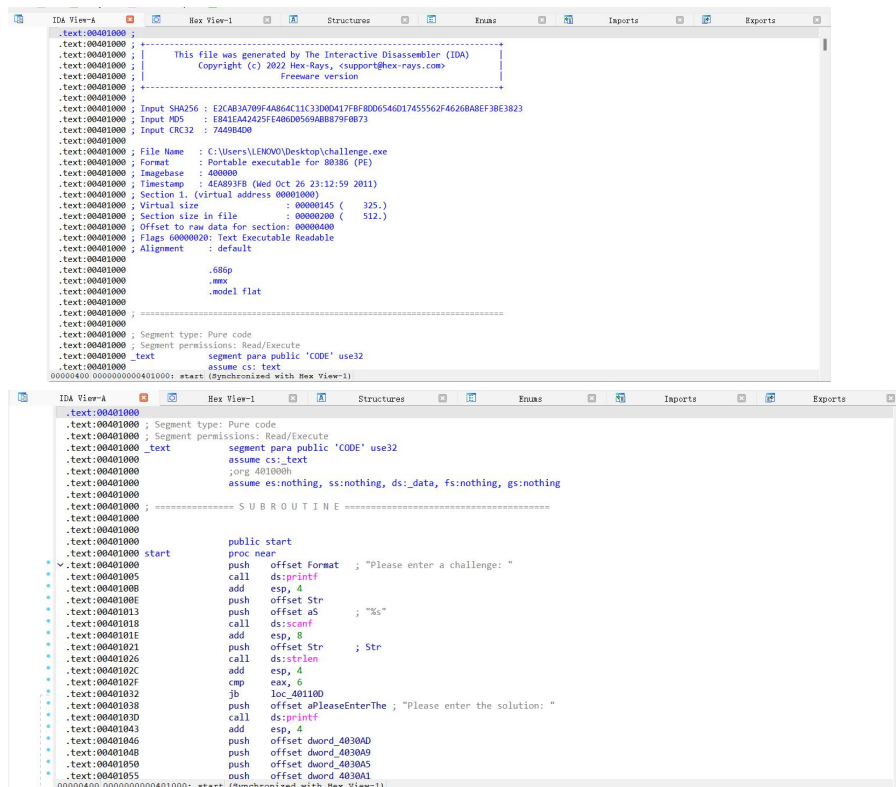
图 2 challenge.exe 的反汇编代码的图形化显示

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。

```
Please enter a challenge: 
Please enter the solution: 
Congratulations, you made it!
```

图 3 逆向分析，完成挑战

三. 使用 IDA，获得二进制代码的反汇编代码



```
IDA View-A Hex View-1 Structures Enums Imports Exports
.text:00401050 push offset dword_4030A5
.text:00401055 push offset dword_4030A1
.text:0040105A push offset a0000 ; "X-X-X-X"
.text:0040105F call ds:scanf
.text:00401065 add esp, 14h
.text:00401068 cmp eax, 4
.text:0040106B jb loc_40111D
.text:00401071 movzx ecx, byte_403082
.text:00401078 movzx ecx, byte_403084
.text:0040107F add ecx, ecx
.text:00401081 movzx ecx, byte_403085
.text:00401088 add ecx, ecx
.text:0040108A cmp ecx, dword_4030A1
.text:00401090 jnz loc_40111D
.text:00401096 mov ecx, dword_4030A5
.text:00401098 add ecx, 18h
.text:0040109E not ecx
.text:004010A0 cmp ecx, 0BADF000h
.text:004010A5 jnz short loc_40111D
.text:004010A7 mov ecx, dword_4030A9
.text:004010AC mov ecx, 0C48h
.text:004010B2 cdq
.text:004010B4 div ecx
.text:004010B6 mov esi, eax
.text:004010B8 movzx ecx, Str
.text:004010BD movzx ecx, byte_403083
.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
00004050 0000000000401050: start+50 (Synchronized with Hex View-1)
```

```
IDA View-A Hex View-1 Structures Enums Imports Exports
.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 loc_4010EA: ; CODE XREF: start+F81j
.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+EC1j
.text:004010FA sub edi, 7Bh ; '('
.text:004010FD cmp eax, edi
.text:004010FF jnz short loc_40111D
.text:00401101 mov dword_4030D1, 1
.text:00401108 jmp short loc_40111D
.text:0040110D loc_40110D: ; CODE XREF: start+321j
.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 locret_401144:
00004050 00000000004010D5: start+D5 (Synchronized with Hex View-1)
```

```
IDA View-A Hex View-1 Structures Enums Imports Exports
.text:0040111D loc_40111D: ; CODE XREF: start+6B1j
.text:0040111D cmp dword_4030D1, 0 ; start+901j ...
.text:00401124 jz short loc_401136
.text:00401126 push offset aCongratulation ; "Congratulations, you made it!\n\n"
.text:0040112B call ds:printf
.text:00401131 add esp, 4
.text:00401134 jmp short locret_401144
.text:00401136 loc_401136: ; CODE XREF: start+1241j
.text:00401136 push offset aWrong ; "Wrong :(\n\n"
.text:0040113B call ds:printf
.text:00401141 add esp, 4
.text:00401144 locret_401144: ; CODE XREF: start+11B1j
.text:00401144 retn ; start+1341j
.text:00401144 start endp
.text:00401144 align 100h
.text:00401200 dd 380h dup(?)
.text:00401200 _text ends
.text: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
00000510 000000000040111D: start+loc_40111D (Synchronized with Hex View-1)
```

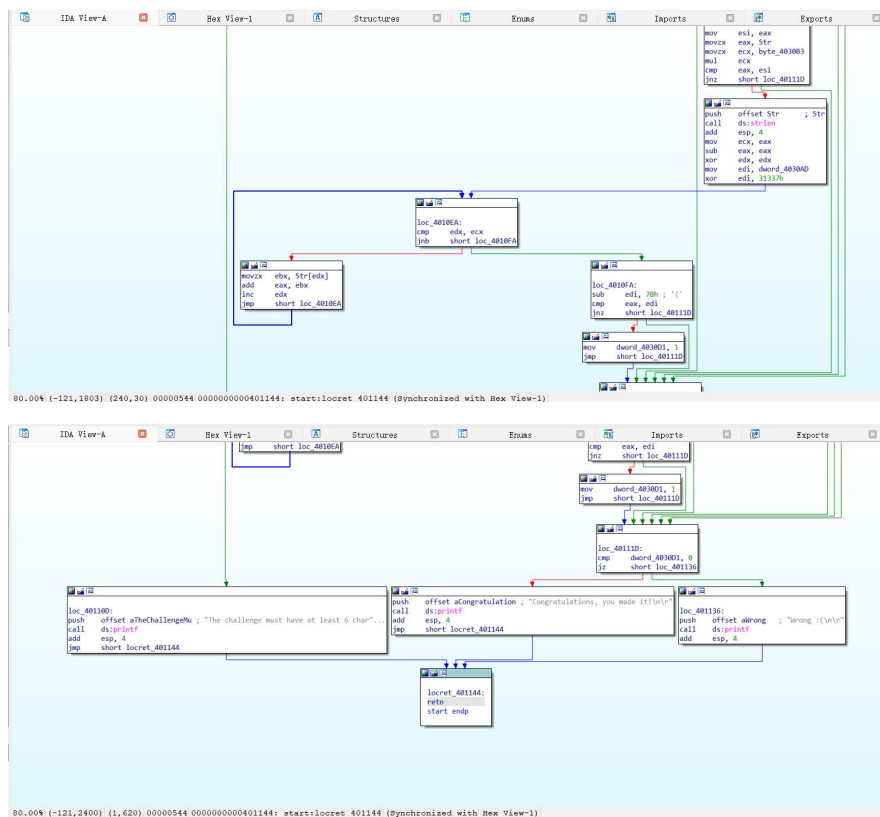
```
IDA View-A Hex View-1 Structures Enums Imports Exports
.idata:00402000 ; Section 2. (virtual address 00002000)
.idata:00402000 ; Virtual size : 00000070 ( 112.)
.idata:00402000 ; Section size in file : 00000200 ( 512.)
.idata:00402000 ; Offset to raw data for section: 00000600
.idata:00402000 ; Flags 40000040: Data Readable
.idata:00402000 ; Alignment : default
.idata:00402000 ; Imports from msvcrt.dll
.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+181p
.idata:00402000 ; start+5F1p
.idata:00402000 ; DATA XREF: ...
.idata:00402004 ; size_t (__cdecl *strlen)(const char *Str)
.idata:00402004 ; extrn strlen: dword ; CODE XREF: start+261p
.idata:00402004 ; start+CF1p
.idata:00402004 ; DATA XREF: ...
.idata:00402008 ; int (*printf)(const char *const Format, ...)
.idata:00402008 ; extrn printf: dword ; CODE XREF: start+51p
.idata:00402008 ; start+301p ...
.idata:0040200C
.idata:0040200C
.idata:00402010 ; Segment type: Pure data
.idata:00402010 ; Segment permissions: Read
.idata:00402010 ; _rdata segment para public 'DATA' use32
.idata:00402010 ; assume cs: data
00000600 0000000000402000: .idata:scanf (Synchronized with Hex View-1)
```

```
IDA View-A Hex View-1 Structures Enums Imports Exports
.rdata:0040200C
.rdata:0040200C
.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
0000000C 000000000040200C: .rdata:0040200C (Synchronized with Hex View-1)
```

```
IDA View-A Hex View-1 Structures Enums Imports Exports
.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 ; Import names for msvcrt.dll
.rdata:00402038 ;
.rdata:00402038 off_402038 dd rva word_402052 ; DATA XREF: .rdata:___IMPORT_DESCRIPTOR_msvcrtfo
.rdata:0040203C dd rva word_402054
.rdata:00402040 dd rva word_402048
.rdata:00402044 dd 0
.rdata:00402048 word_402048 dw 281h ; DATA XREF: .rdata:00402040fo
.rdata:0040204A db 'printf',0
.rdata:00402051 align 2
.rdata:00402052 word_402052 dw 281h ; DATA XREF: .rdata:off_402038fo
.rdata:00402054 db 'scanf',0
.rdata:0040205A word_40205A dw 2A1h ; DATA XREF: .rdata:0040203Cfo
.rdata:0040205C db 'strlen',0
.rdata:00402063 align 4
.rdata:00402064 aMsvcrtDll db 'msvcrt.dll',0 ; DATA XREF: .rdata:0040201Cfo
.rdata:0040206F align 1000h
.rdata:0040206F .rdata ends
.rdata:0040206F
.rdata:00403000 ; Section 3. (virtual address 00003000)
.rdata:00403000 ; Virtual size : 00000005 ( 213.)
.rdata:00403000 ; Section size in file : 00000200 ( 512.)
.rdata:00403000 ; Offset to raw data for section: 00000000
.rdata:00403000 ; Flags 00000040: Data Readable Writable
.rdata:00403000 ; Alignment : default
00000040 0000000000403000: .rdata:00402032 (Synchronized with Hex View-1)
```

```
IDA View-A Hex View-1 Structures Enums Imports Exports
.rdata:00403000 ; Flags 00000040: Data Readable Writable
.rdata:00403000 ; Alignment : default
.rdata:00403000 ;
.rdata:00403000 ; Segment type: Pure data
.rdata:00403000 ; Segment permissions: Read/Write
.rdata:00403000 .rdata segment para public 'DATA' use32
.rdata:00403000 assume cs:_rdata
.rdata:00403000 ;org 403000h
.rdata:00403000 ; char Format[]
.rdata:00403000 Format db 'Please enter a challenge: ',0
.rdata:00403000 ; DATA XREF: startfo
.rdata:00403010 ; char aS[]
.rdata:00403010 aS db 'S',0 ; DATA XREF: start+13fo
.rdata:0040301E ; char aTheChallenge[]
.rdata:0040301E aTheChallenge db 'The challenge must have at least 6 characters',0Ah
.rdata:0040301E ; DATA XREF: start:loc_40118Dfo
.rdata:0040301E db 00h,0
.rdata:0040301E ; char aPleaseEnterThe[]
.rdata:0040304E aPleaseEnterThe db 'Please enter the solution: ',0
.rdata:0040304E ; DATA XREF: start+38fo
.rdata:0040306A ; char aUUUU[]
.rdata:0040306A aUUUU db 'XU-XU-XU-XU',0 ; DATA XREF: start+5Afo
.rdata:00403076 ; char aWrong[]
.rdata:00403076 aWrong db 'Wrong :(',0Ah ; DATA XREF: start:loc_401136fo
.rdata:00403076 db 00h,0
.rdata:00403081 ; char aCongratulation[]
.rdata:00403081 aCongratulation db 'Congratulations, you made it!',0Ah
.rdata:00403081 ; DATA XREF: start+126fo
.rdata:00403081 db 00h,0
.rdata:004030A1 ; dword_4030A1 dd 0 ; DATA XREF: start+55fo
.rdata:004030A1 ; start+8Afo
00000080 0000000000403000: .rdata:Format (Synchronized with Hex View-1)
```

```
IDA View-A Hex View-1 Structures Enums Imports Exports
.rdata:004030A1 ; dword_4030A1 dd 0 ; DATA XREF: start+55fo
.rdata:004030A1 ; start+8Afo
.rdata:004030A5 ; dword_4030A5 dd 0 ; DATA XREF: start+50fo
.rdata:004030A5 ; start+96fo
.rdata:004030A9 ; dword_4030A9 dd 0 ; DATA XREF: start+48fo
.rdata:004030A9 ; start+A7fo
.rdata:004030AD ; dword_4030AD dd 0 ; DATA XREF: start+46fo
.rdata:004030AD ; start+DEfo
.rdata:004030B1 ; char Str
.rdata:004030B1 Str db 0 ; DATA XREF: start+Efo
.rdata:004030B1 ; start+21fo ...
.rdata:004030B2 byte_4030B2 db 0 ; DATA XREF: start+71fo
.rdata:004030B3 byte_4030B3 db 0 ; DATA XREF: start+80fo
.rdata:004030B4 byte_4030B4 db 0 ; DATA XREF: start+78fo
.rdata:004030B5 byte_4030B5 db 0 ; DATA XREF: start+81fo
.rdata:004030B6 db 0
.rdata:004030B7 db 0
.rdata:004030B8 db 0
.rdata:004030B9 db 0
.rdata:004030BA db 0
.rdata:004030BB db 0
.rdata:004030BC db 0
.rdata:004030BD db 0
.rdata:004030BE db 0
.rdata:004030BF db 0
.rdata:004030C0 db 0
.rdata:004030C1 db 0
.rdata:004030C2 db 0
.rdata:004030C3 db 0
.rdata:004030C4 db 0
.rdata:004030C5 db 0
.rdata:004030C6 db 0
000000A1 00000000004030A1: .rdata:dword_4030A1 (Synchronized with Hex View-1)
```

四. 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，在实验报告中记录逆向分析的详细过程。

1.

```

.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

```

输入一个 challenge，challenge 保存在 004030B1。如果 challenge 位数小于 6 位，执行 loc_401160, 否则则正常执行接下来的代码。将输入的 challenge 设为 123456。

2.

```

.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

```

如果 challenge 位数大于等于 6 位，则执行此代码。按照规定格式 %u-%u-%u-%u 输入 solution，如果 solution 位数小于 4 位，跳转执行 loc_401160,否则则正常执行接下来的代码。solution 按位依次保存在 004030AD,004030A9,004030A5,004030A1。将 solution 设置为 a-b-c-d。

3.

```

movzx    eax, byte_4030B2
movzx    ecx, byte_4030B4
add       eax, ecx
movzx    ecx, byte_4030B5
add       eax, ecx
cmp       eax, dword_4030A1
jnz      loc_40111D

```

将 challenge 的第 2、4、5 位对应 ascii 码相加存入 eax，并与 solution 的第一位数据进行比较。如果不相等，则跳转 loc_40111D，否则正常执行接下来的代码。对于输入的 challenge，solution 的第一位 a = 2 的 ascii 码+4 的 ascii 码+5 的 ascii 码=50+52+53=155。

4.

```

mov       eax, dword_4030A5
add       eax, 18h
not       eax
cmp       eax, 0BADF000Dh
jnz      short loc_40111D

```

将 solution 第二位 b 加上 18h 后取反, 与 0BADF000Dh 进行比较。

如果不相等, 则跳转至 loc_40111D, 否则正常执行接下来的代码。

对于 challenge 的 solution 的第二位 b, 满足 $\sim(b+18h)=0BADF000Dh$, 解得 $b=1159790554$ 。

5.

```
mov     eax, dword_4030A9
mov     ecx, 0C48h
cdq
div     ecx
mov     esi, eax
movzx   eax, Str
movzx   ecx, byte_4030B3
mul     ecx
cmp     eax, esi
jnz     short loc_40111D
```

把 solution 第三位除以 0C48h 后, 与 challenge 第一位、第三位对应 ascii 码的乘积相比较, 如果不相等, 则跳转至 loc_40111D, 否则正常执行接下来的代码。对于 challenge 的 solution 的第三位 c, $c=1$ 的 ascii 码 * 3 的 ascii 码 * 0C48h = $49 * 51 * 3144 = 7856856$ 。

6.

```
push    offset Str      ; Str
call    ds:strlen
add     esp, 4
mov     ecx, eax
sub     eax, eax
xor     edx, edx
mov     edi, dword_4030AD
xor     edi, 31337h
```

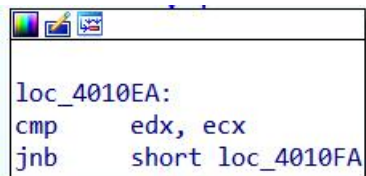
```
loc_4010FA:
sub     edi, 7Bh ; '{'
cmp     eax, edi
jnz     short loc_40111D
```

把 solution 的第四位与 31337h 按位异或后存入 edi, edx 置零。

edi 减去 78h, 与 eax 进行比较, 如果不相等, 则跳转至 loc_40111D, 否则正常执行接下来的代码。

对于 challenge 的 solution 的第四位 d, d=201351。

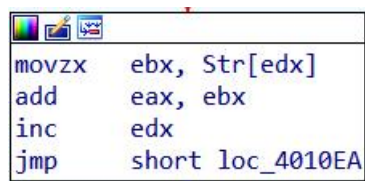
7.



```
loc_4010EA:
cmp     edx, ecx
jnb     short loc_4010FA
```

比较 edx 和 ecx, 如果 edx 大于等于 ecx, 则跳转到 loc_4010FA, 否则正常执行接下来的代码。

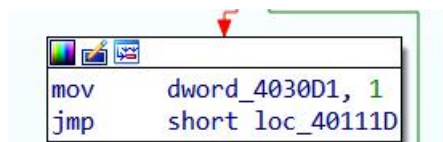
8.



```
movzx   ebx, Str[edx]
add     eax, ebx
inc     edx
jmp     short loc_4010EA
```

依次将 challenge 的内容加入 eax 中, 之后无条件跳转进入 loc_4010EA。

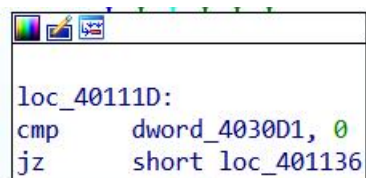
9.



```
mov     dword_4030D1, 1
jmp     short loc_40111D
```

如果 4030D1=1, challenge 和 solution 相匹配, 进入 loc_40111D

10.

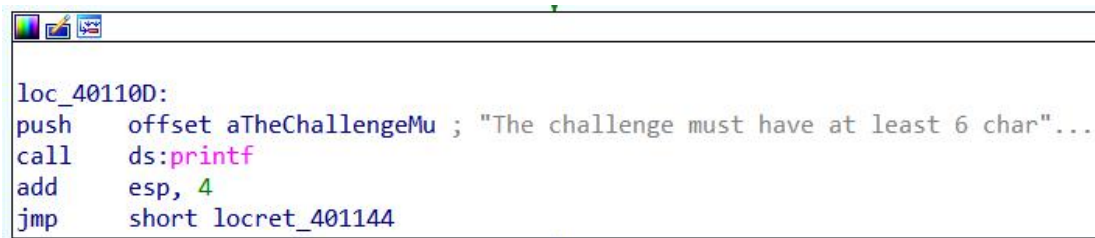


```
loc_40111D:
cmp     dword_4030D1, 0
jz      short loc_401136
```

比较 4030D1 和 0, 4030D1=0 说明 challenge 和 solution 不匹配, 进

入 loc_401136

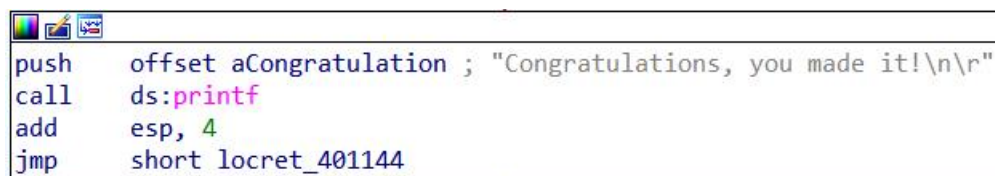
11.



```
loc_40110D:
push    offset aTheChallengeMu ; "The challenge must have at least 6 char"...
call    ds:printf
add     esp, 4
jmp     short locret_401144
```

如果输入的 challenge 位数小于 6，则到 loc_40110D 执行 “The challenge must have at least 6 char” 的输出，无条件跳转到 locret_401144.

12.

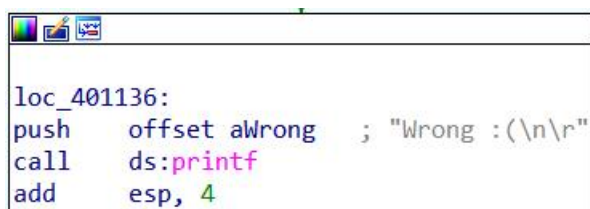


```
push    offset aCongratulation ; "Congratulations, you made it!\n\r"
call    ds:printf
add     esp, 4
jmp     short locret_401144
```

如果 challenge 与 solution 相匹配，到此处执行

“Congratulations,you made it!” 的输出，无条件跳转到 locret_401144.

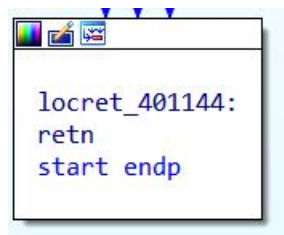
13.



```
loc_401136:
push    offset aWrong ; "Wrong :(\n\r"
call    ds:printf
add     esp, 4
```

如果 challenge 和 solution 不匹配，到此处执行 “Wrong:(” 的输出.

14.



程序结束。

五. 程序执行结果

A screenshot of a terminal window with a black background and white text. The window title bar shows the file path 'C:\Users\LENOVO\Desktop\cf'. The terminal output is as follows:

```
Please enter a challenge: 123456  
Please enter the solution: 155-1159790554-7856856-201351  
Congratulations, you made it!
```