

# 汇编语言与逆向技术课程实验报告

## 实验七：Reverse Engineering Challenge



学院: 密码与网络空间安全学院

专业: 信息安全

学号: 2412950

姓名: 路浩斌

班级: 信安一班

## 一、实验目的

- 1、熟悉反编译工具 Binary Ninja；
  - 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. 使用 Binary Ninja 打开待分析的 PE 文件 challenge.exe

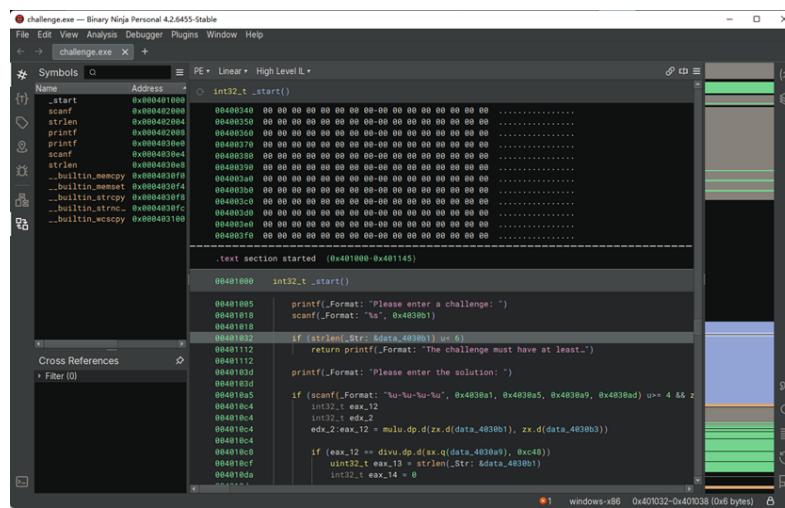


图 1: 打开 challenge.exe

2. 切换反汇编视图，得到二进制代码的反汇编代码，如图 2 所示

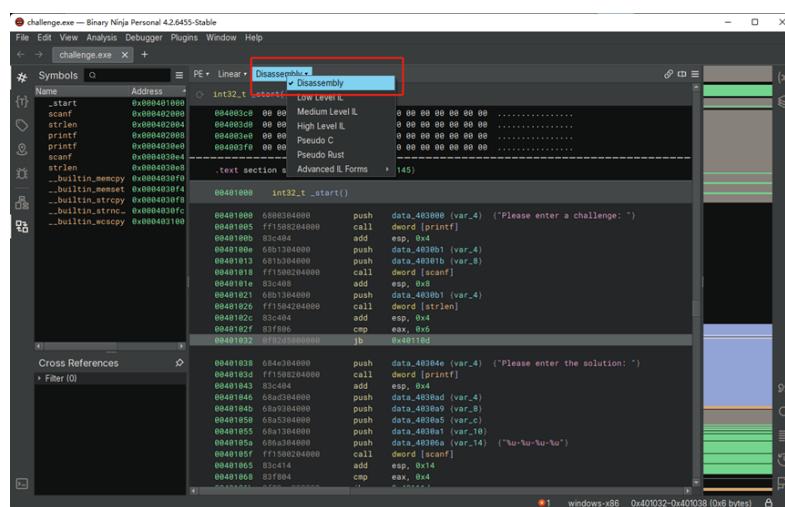


图 2: 查看 challenge.exe 的反汇编代码

### 3. 切换到汇编代码的图形化视窗，分析程序的执行逻辑

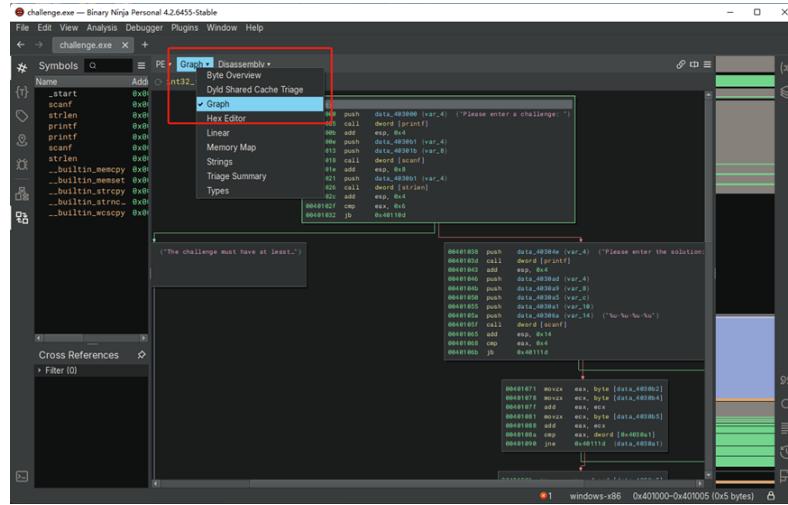


图 3: challenge.exe 的反汇编代码的图形化显示

### 4. 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。

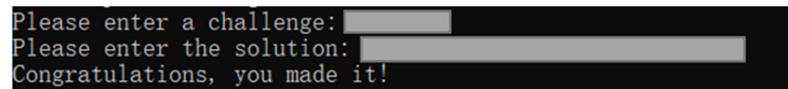


图 4: 逆向分析，完成挑战

### 三、实验报告内容

#### 1. 使用 Binary Ninja, 获得二进制代码的反汇编代码及图形化视窗, 提供截图。

```
00401000 6800304000    push    data_403000 {var_4} {"Please enter a challenge: "}
00401005 ff1508204000    call    dword [printf]
0040100b 83c404    add     esp, 0x4
0040100e 68b1304000    push    data_4030b1 {var_4}
00401013 68b1304000    push    data_4030b1 {var_8}
00401018 ff1508204000    call    dword [scanf]
0040101e 83c408    add     esp, 0x8
00401021 68b1304000    push    data_4030b1 {var_4}
00401026 ff1504204000    call    dword [strlen]
0040102c 83c404    add     esp, 0x4
0040102f 83f806    cmp     eax, 0x6
00401032 0f82d5000000    jb     0x40110d

00401038 684e304000    push    data_40304e {var_4} {"Please enter the solution: "}
0040103d ff1508204000    call    dword [printf]
00401043 83c404    add     esp, 0x4
00401046 68ad304000    push    data_4030ad {var_4}
0040104b 68a9304000    push    data_4030a9 {var_8}
00401059 68a5304000    push    data_4030a5 {var_c}
00401055 68a1304000    push    data_4030a1 {var_10}
0040105f ff1508204000    call    dword [scanf]
00401065 83c414    add     esp, 0x14
00401068 83f804    cmp     eax, 0x4
0040106b 0f82ac000000    jb     0x40111d

00401071 0fb605b2304000    movzx   eax, byte [data_4030b2]
00401078 0fb600db4304000    movzx   ecx, byte [data_4030b4]
0040107f 03c1    add     eax, ecx
00401081 0fb600db5304000    movzx   ecx, byte [data_4030b5]
00401088 03c1    add     eax, ecx
0040108a 3b05a1304000    cmp     eax, dword [data_4030a1]
00401090 0f8587000000    jne    0x40111d
```

图 5: 反汇编代码

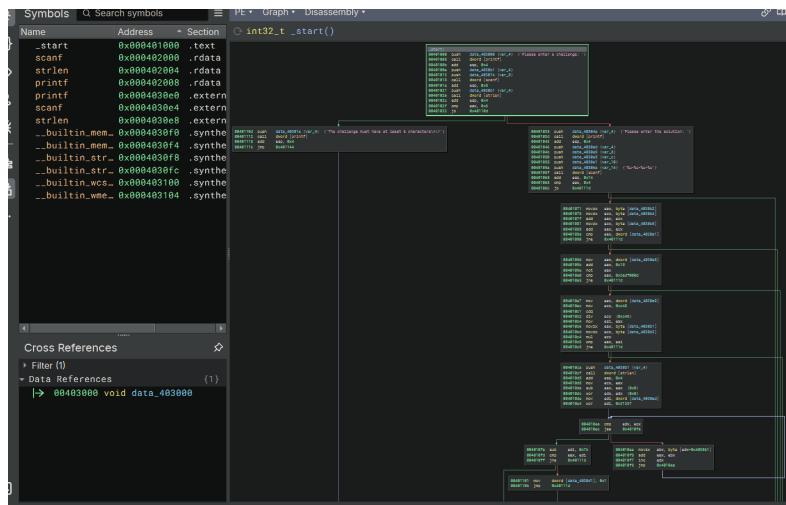
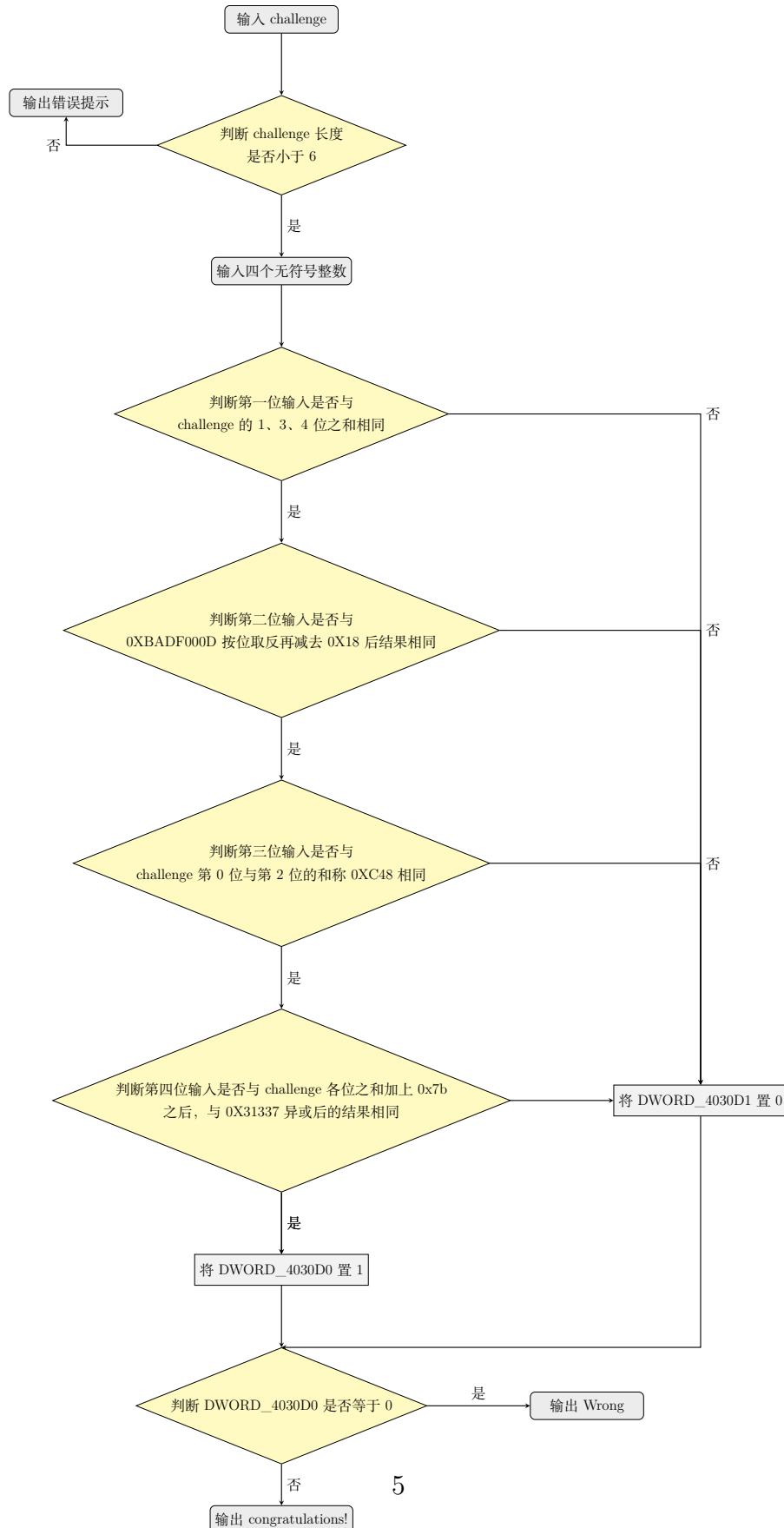


图 6: 图形化视窗

2.2 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，在实验报告中记录逆向分析的详细过程。

对反汇编代码进行分析后，得到总体流程图如下：



接下来对反汇编代码进行详细分析：

```
1 _start:
2 00401000 push    data_403000 {var_4} {"Please\u0020enter\u0020a\u0020challenge:\u0020"}
3 00401005 call    dword [printf]
4 0040100b add     esp, 0x4
5 0040100e push    data_4030b1 {var_4}
6 00401013 push    data_40301b {var_8}
7 00401018 call    dword [scanf]
8 0040101e add     esp, 0x8
9 00401021 push    data_4030b1 {var_4}
10 00401026 call   dword [strlen]
11 0040102c add    esp, 0x4
12 0040102f cmp    eax, 0x6
13 00401032 jb    0x40110d
```

这串代码的功能是：输入 challenge 并判断其长度是否小于 6，如果小于 6 的话就实行跳转到错误信息。

```
1 00401038 push    data_40304e {var_4} {"Please\u0020enter\u0020the\u0020solution:\u0020"
2 }
3 0040103d call    dword [printf]
4 00401043 add     esp, 0x4
5 00401046 push    data_4030ad {var_4}
6 0040104b push    data_4030a9 {var_8}
7 00401050 push    data_4030a5 {var_c}
8 00401055 push    data_4030a1 {var_10}
9 0040105a push    data_40306a {var_14} {"%u-%u-%u-%u"}
10 0040105f call   dword [scanf]
11 00401065 add    esp, 0x14
12 00401068 cmp    eax, 0x4
13 0040106b jb    0x40111d
```

这串代码的功能是：依次输入 4 个数，并且比较输入的数量是否等于 4，如果不等于 4 的话，就跳转到错误信息。

```
1 00401071 movzx   eax, byte [data_4030b2]
2 00401078 movzx   ecx, byte [data_4030b4]
3 0040107f add    eax, ecx
4 00401081 movzx   ecx, byte [data_4030b5]
5 00401088 add    eax, ecx
6 0040108a cmp    eax, dword [data_4030a1]
```

```
7 00401090 jne 0x40111d
```

这串代码的功能是：将 challenge 的 1、3、4 位取出加和，并判断第一位输入的数字是否和其相等，如果不相等，跳转到 0x40111D。

```
1 00401096 mov eax, dword [data_4030a5]  
2 0040109b add eax, 0x18  
3 0040109e not eax  
4 004010a0 cmp eax, 0xbadf000d  
5 004010a5 jne 0x40111d
```

这串代码的功能是：将第二位数字加上 0x18 然后按位取反，和 0xBADF000D 进行比较，如果不相等，跳转到 0x40111D。

```
1 004010a7 mov eax, dword [data_4030a9]  
2 004010ac mov ecx, 0xc48  
3 004010b1 cdq  
4 004010b2 div ecx {0xc48}  
5 004010b4 mov esi, eax  
6 004010b6 movzx eax, byte [data_4030b1]  
7 004010bd movzx ecx, byte [data_4030b3]  
8 004010c4 mul ecx  
9 004010c6 cmp eax, esi  
10 004010c8 jne 0x40111d
```

这串代码的功能是：将第三位数和 0C48h 相除，并用 edx 进行高位拓展，eax 存商，edx 存余数，将商存到 esi 中。将 challenge 的第 0 位，第 2 位相乘。比较 esi 和 eax 是否相等，如果不相等，跳转到 0x40111D。

```
1 004010ca push data_4030b1 {var_4}  
2 004010cf call dword [strlen]  
3 004010d5 add esp, 0x4  
4 004010d8 mov ecx, eax  
5 004010da sub eax, eax {0x0}  
6 004010dc xor edx, edx {0x0}  
7 004010de mov edi, dword [data_4030ad]  
8 004010e4 xor edi, 0x31337
```

这串代码的功能是：ecx 中存放 challenge 的长度，并将 eax 和 edx 全部置零，将第四位数字和 0x31337 进行异或后保存到 edi 中。

```

004010fa sub edi, 0x7b
004010fd cmp eax, edi
004010ff jne 0x40111d

004010ee movzx ebx, byte [edx+0x4030b1]
004010f5 add eax, ebx
004010f7 inc edx
004010f8 jmp 0x4010ea

00401101 mov dword [data_4030d1], 0x1
0040110b jmp 0x40111d

```

图 7: 查看 challenge.exe 的反汇编代码

这部分的功能是：用 eax 循环累加 challenge 的每一位，并将 edi 减去 7Bh 后比较 eax 和 bdi 是否相等，如果不相等跳转到 0x40111D。

```

0040111d cmp dword [data_4030d1], 0x8
00401124 je 0x401136 ("hv00")

00401136 push data_4030f6 (var_4) ("Wrong :(\n\r")
00401138 call dword [printf]
00401141 add esp, 0x4

00401135 push data_4030f1 (var_4) ("Congratulations, you made it!\n\r")
00401136 call dword [printf]
00401139 add esp, 0x4
0040113a jmp 0x401144

00401144 retn (...return_addr)

```

这部分的功能是：当以上条件均满足的时候，会将 dword\_4030D1 置为 1，输出 Congratulations, you made it!。如果上面任何一步不满足，就会直接跳转到 dword\_40111D，而此时 dword\_3040D1 的值为 0，跳转到 0x401136，输出 Wrong:(

用数学形式更直观的体现上述功能：

设输入的 challenge 为  $c[1], c[2], \dots, c[n-1]$  ( $n \geq 6$ )，输入的四个无符号整数为  $s1, s2, s3, s4$ 。

- $s1 = c[1] + c[3] + c[4]$
- $s2 = 1159790554$
- $s3 = (0xc48) \times c[0] \times c[2]$
- $s4 = (c[0] + c[1] + \dots + c[n-1] + 0x7b) xor (0x31337)$

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。

```
Administrator:命令提示符
Microsoft Windows [Version 10.0.26200.7171]
(c) Microsoft Corporation. All rights reserved.

C:\Users\17026>cd Desktop

C:\Users\17026\Desktop>challenge.exe
Please enter a challenge: 177247
Please enter the solution: 157-1159790554-8473080-201344
Congratulations, you made it!

C:\Users\17026\Desktop>
```

#### 四、程序中出现的一个 bug

经过几次测试发现，当输入的 challenge 过长时，solution 无论输入什么，都可以得到 Congratulations, you made it!

通过分析程序猜测是因为输入 challenge 过长，会将输出结果的内存空间覆盖掉，导致最后 dword\_4030D1 一定为 1。应当对 challenge 增加长度限制或检测来解决。