

南開大學

## 汇编语言与逆向技术课程实验报告

### 实验七：Reverse Engineering Challenge



学 院 网络空间安全学院  
专 业 信息安全  
学 号 2211044  
姓 名 陆皓喆  
班 级 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 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 ; ===== SUBROUTINE =====
.text:00401000
.text:00401000
.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          ; "%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 aUUUUU      ; "%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

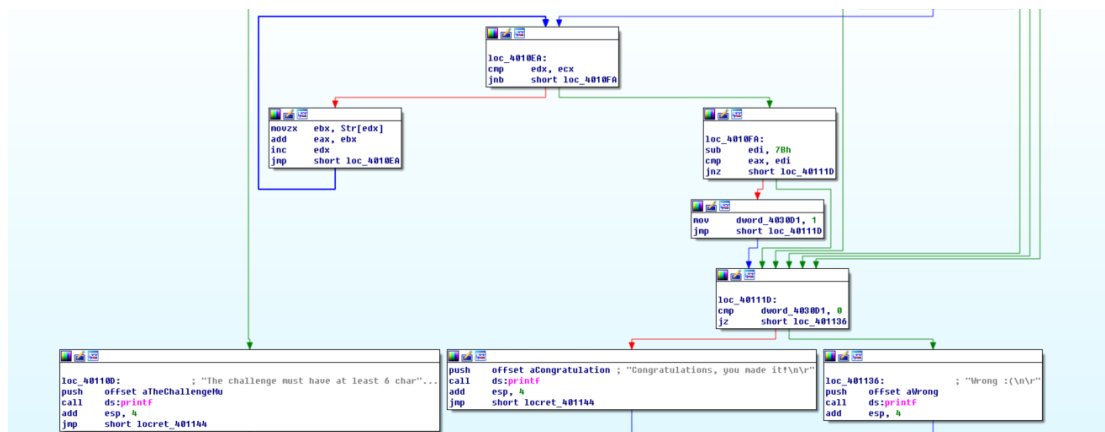


图 2 challenge.exe 的反汇编代码的图形化显示

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。

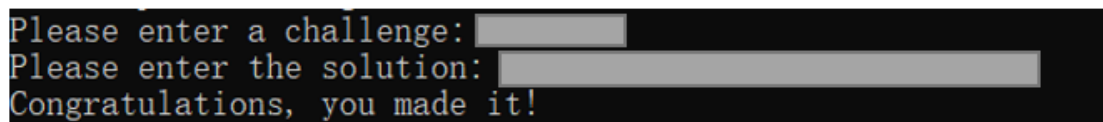
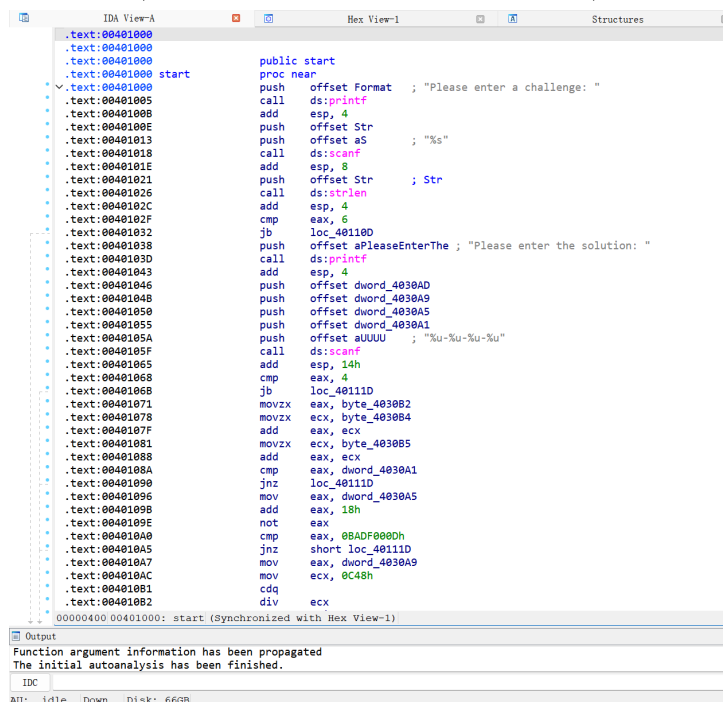
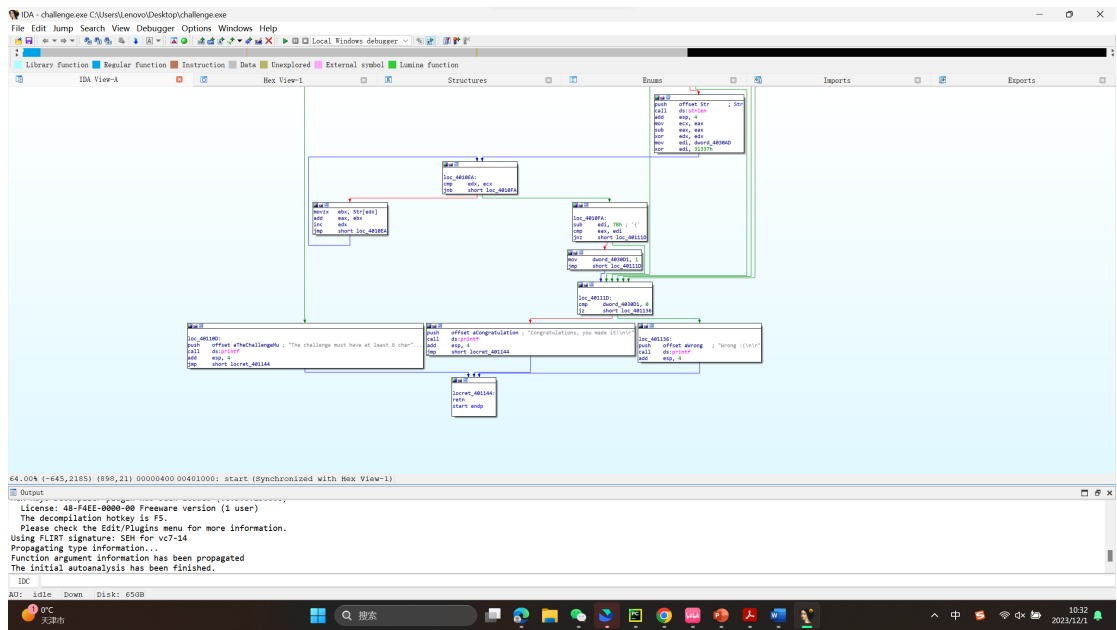
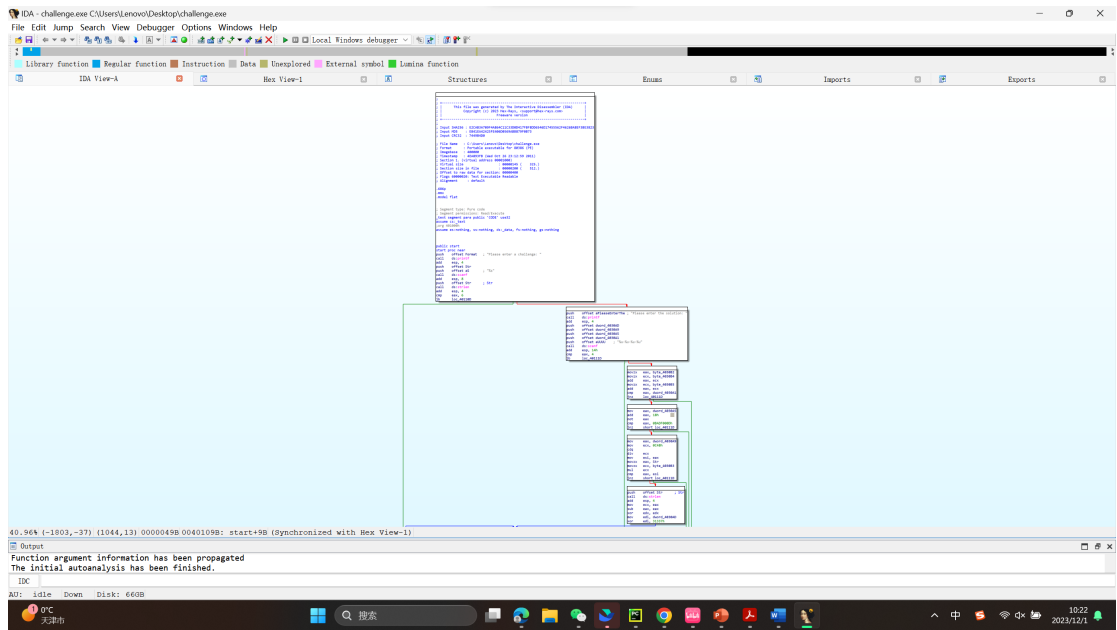


图 3 逆向分析，完成挑战

## 三、实验报告

### 3.1 使用 IDA Freeware，获得二进制代码的反汇编代码，提供截图

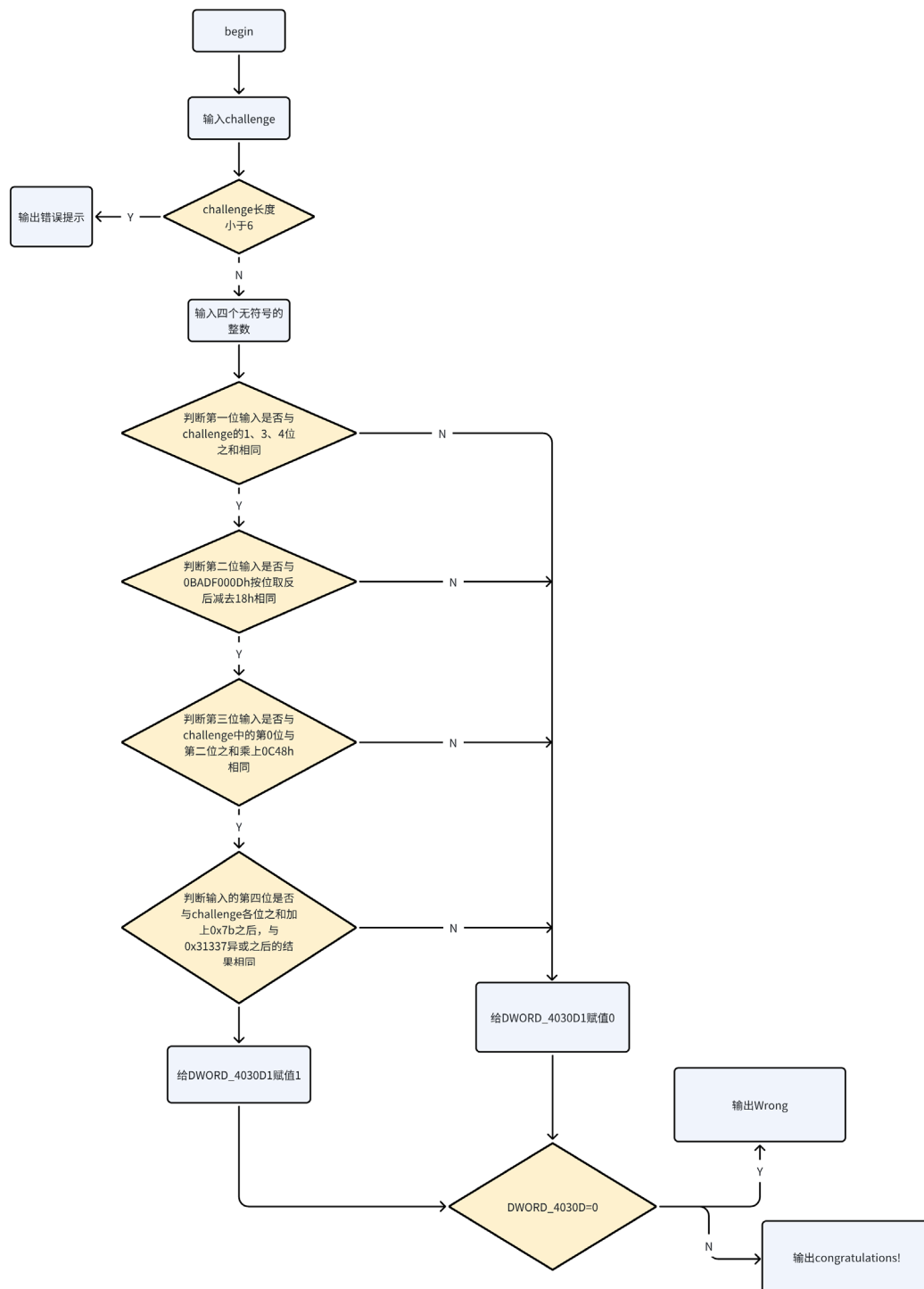




上图就是提供的二进制文件的反汇编代码。

### 3.2 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

先根据 IDA 提供的信息，画出流程表。



然后再对每一步逆向的结果进行详细的分析。

```

public start
start proc near
push    offset Format    ; "Please enter a challenge: "
call    ds:printf
add     esp, 4
push    offset Str
push    offset aS        ; "%S"
call    ds:scanf
add     esp, 8
push    offset Str        ; Str
call    ds:strlen
add     esp, 4
cmp     eax, 6
jb      loc_40110D

```

该段代码的功能是：

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。

```

push    offset aPleaseEnterThe ; "Please enter the solution: "
call    ds:printf
add     esp, 4
push    offset dword_4030AD
push    offset dword_4030A9
push    offset dword_4030A5
push    offset dword_4030A1
push    offset aUUUU        ; "%u-%u-%u-%u"
call    ds:scanf
add     esp, 14h
cmp     eax, 4
jb      loc_40111D

```

该段代码的功能是：

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。

```

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

```

该段代码的功能是：

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 **loc\_40111D**。

```

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

```

该段代码的功能是：

将第二位数字加上 18h 然后按位取反，和 BADF000Dh 进行比较，如果不相等，跳转到 loc\_40111D。

```

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

```

该段代码的功能是：

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0, 2位相乘。比较esi和eax是否相等，如果不相等，跳转到loc\_40111D。

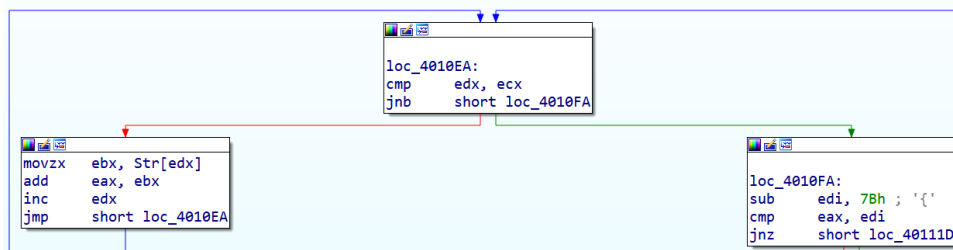
```

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

```

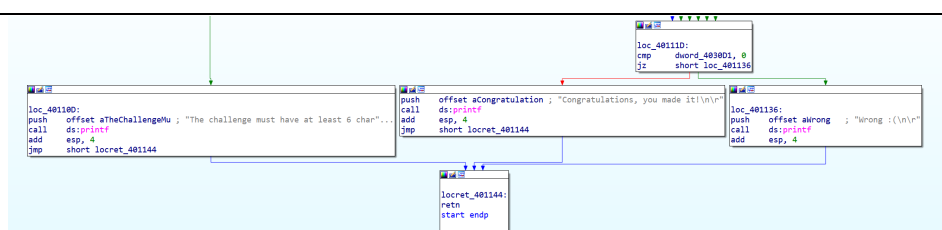
该段代码的功能是：

ecx中存放challenge的长度，eax和edx全部置零，将第四位数字和31337h进行异或并保存到edi中。



该段代码的功能是：

用eax循环累加challenge 的每一位，并将edi减去7Bh后比较eax和edi是否相等，如果不相等跳转到loc\_40111D。



该段代码的功能是：

当以上条件均满足的时候，会正常进行 `mov dword_4030D1,1`的操作，此时loc\_40111D中的判断不能满足，正常输出Congratulations, you made it!。如果又任何一步不满足，就会直接跳转到dword\_40111D，而此时dword\_3040D1的值为0，导致判断满足条件，跳转到loc\_401136 输出Wrong:(。

3.3 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，将成功的截图复制到实验报告中

```

C:\WINDOWS\system32\cmd. x + v

Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>C:\Users\Lenovo\Desktop\challenge.exe
Please enter a challenge: 114514
Please enter the solution: 151-1159790554-8010912-201372
Congratulations, you made it!

C:\Users\Lenovo>

```

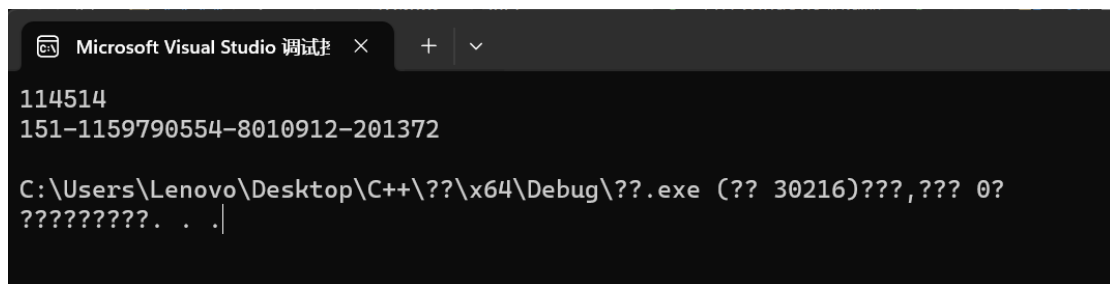


为了更好的解出结果，我简单地编写了一个C++程序来解出结果，相当于一个反方向的过程。

C++代码如下所示：

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5.     string str;
6.     cin >> str;
7.     int len = str.length();
8.     unsigned int num1 = 0, num2 = 0, num3 = 0, num4 = 0;
9.     num1 = str[1] + str[3] + str[4];
10.    num2 = ((~0xBADF000D) - 0x18);
11.    num3 = (str[0] * str[2]) * 0x0c48;
12.    for (int i = 0; i < len; i++)
13.        num4 += str[i];
14.    num4 = (num4 + 0x7b) ^ 0x31337;
15.    cout << num1 << "-" << num2 << "-" << num3 << "-"
        << num4 << endl;
16.
17.    return 0;
18.}
```

用该程序就可以解出最后的结果，使其输出“Congratulations,you made it!”



```
Microsoft Visual Studio 调试  ×  +  v
114514
151-1159790554-8010912-201372
C:\Users\Lenovo\Desktop\C++\??\x64\Debug\???.exe (?? 30216)???,?? 0?
?????????. .|
```

(如上图所示，输入114514即可输出我们想要得到的结果)



经过查看内存空间，我发现：当challenge的长度大于等于36时，会使得输出结果的内存空间被覆盖，导致其直接输出1，使最后的结果不管怎么样都会输出正确的“Congratulations, you made it! ”。这其实是该程序的一个漏洞，可以通过限制其输入长度或者改变输出结果内存存储位置去解决这个问题。

```
.data:004030B1 Str db 0 ; DATA XREF: start+Efo
.data:004030B1 ; start+21fo ...
.data:004030B2 Str_1 db 0 ; DATA XREF: start+71fr
.data:004030B3 byte_4030B3 db 0 ; DATA XREF: start+80fr
.data:004030B4 Str_3 db 0 ; DATA XREF: start+78fr
.data:004030B5 byte_4030B5 db 0 ; DATA XREF: start+81fr
.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+101fw
; start:loc_40111Dfr
```

(如图，查看输出结果的内存空间)

希望在接下来的逆向分析课程中，能够继续深入学习反汇编的一些知识。