分道扬镳 Write UP

前言：做完后才发现、好像不是分给我们组的题 orz【背锅】【逃】

标题：分道扬镳

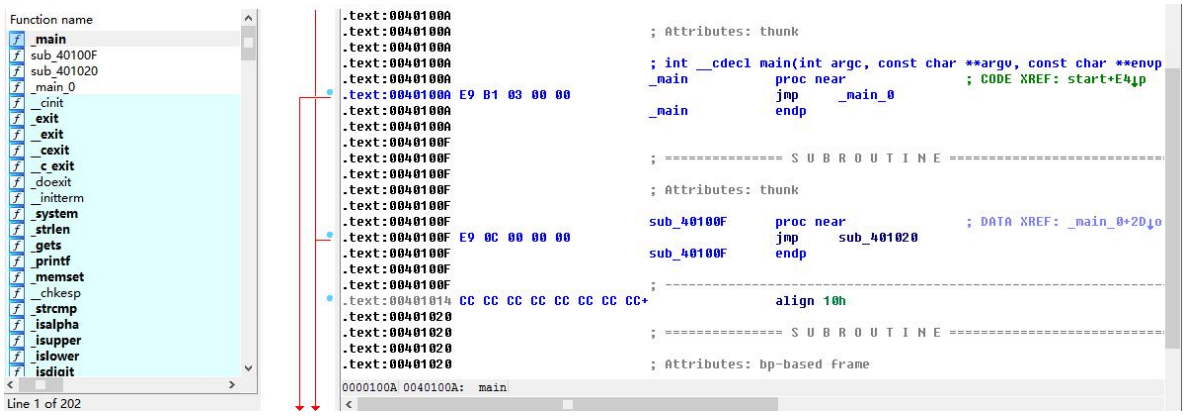
描述：注意进入正确的流程，用最短的步骤走完迷宫

咳、做完题的我来解释下题目的含义：本题有坑，别走错路，正确的路是要走迷宫的（多么简单直白啊）（心疼当时懵懂无知的我一秒）

PS：其实该题除了有个小坑外，还是比较水的

让我们开始欢快地解题吧

按照国际惯例，放入 IDA



有个 main 方法耶，看看内容：恩，好像很直白啊，执行_main_0 方法

那我们来看看_main_0 是什么吧

F5 一下

IDA View-A | Pseudocode-C | Pseudocode-

```
int __cdecl main_0()
{
  size_t v0; // eax@9
  size_t v1; // eax@11
  char v3; // [sp+Ch] [bp-7Ch]@1
  char v4[48]; // [sp+4Ch] [bp-3Ch]@1
  unsigned int v5; // [sp+7Ch] [bp-Ch]@1
  unsigned int v6; // [sp+80h] [bp-8h]@1
  int v7; // [sp+84h] [bp-4h]@1

  memset(&v3, -858993460, 0x7Cu);
  v6 = 0;
  v5 = 0;
  v7 = 0 / 0;
  printf("Please input your key:\n");
  gets(v4);
  if ( strlen(v4) > 0x20 )
  {
    printf("Too long!\n");
  }
  else
  {
    v5 = 0;
    v4[strlen(v4)] = 0;
    if ( strlen(v4) )
    {
      do
      {
        if ( !isalpha(v4[v5]) && v4[v5] != 123 && v4[v5] != 125 && v4[v5] != 95 )
        {
          printf("Sorry,you are wrong!\n");
          exit(0);
        }
        ++v5;
        v0 = strlen(v4);
      }
      while ( v5 < v0 );
    }
    _strlwr(v4);
    v6 = 0;
    if ( strlen(v4) )
    {
      do
      {
        v4[v6++] += -128;
        v1 = strlen(v4);
      }
      while ( v6 < v1 );
    }
    if ( strcmp(v4, "翳轶咦筮徇驷腺哝,,,") )
      printf("Sorry,you are wrong!\n");
    else
      printf("Good!\n");
    system("pause");
  }
  system("pause");
  return 0;
}
```

这个时候我犯下了本题的第二个错误（第一个错误是没有先查看字符串）
此时的我发现这个函数有两个不对劲的地方：

1、 `v7 = 0 / 0;`　　诶、这条语句是怎么执行通过的 0.0

```
if ( strlen(v4) > 0x20 )
{
  printf("Too long!\n");
}
```

2、 如果输入的字符串大于 32 位，输出 Too long！

于是我就试了一下（对还是最开始那张图）

诶，怎么回事，为什么还是输出 Sorry you are wrong！ 0.0

现在回想，这是多么大的两个提示啊 orz，后来发现这个函数执行到 0/0 那句就被强行跳过了，第二个简直就是赤裸裸地提醒你走错路了嘛……

然而，当时的我大概是失了智了，在错误的方向上一路狂奔……

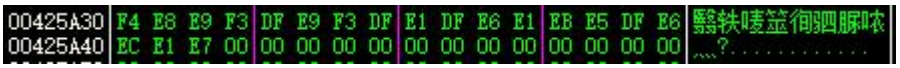
分析一下这个函数，没什么特别的，首先输入必须为字母或_或{}，然后对每个字符-128，和一个字符串比较，如果一样就 ok。

看下这个字符串是什么：

```
.data:00425A30                          ; char aSsSSCsixCSc[]
.data:00425A30 F4 E8 E9 F3 DF E9 F3 DF+aSsSSCsixCSc     db '鬻铁唛筮徇驷腺哝灬',0
.data:00425A30 E1 DF E6 E1 EB E5 DF E6+                 ;
```

嗯？好像不是很全？（可能只是我操作问题 orz）（谁知道怎么在 IDA 里看完整字符串求指点）

于是我开了个吾爱破解看内存



写个 python 跑下

this_is_a_fake_flag

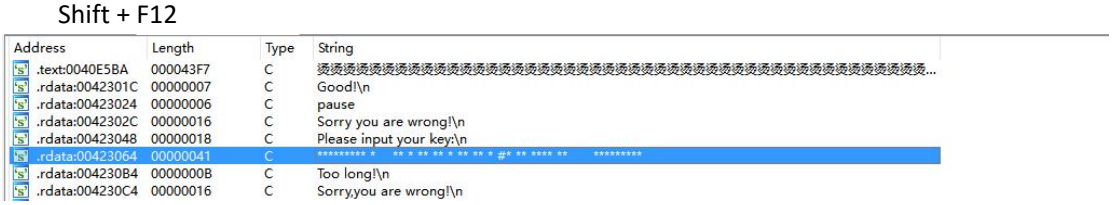
顺带附上一版强行删掉了那句除法的 rev2_1_1.exe，可以自行感受一下【摊手】

.....失了智的我被成功嘲讽 >.< ……

不死心的我还是把这句拿去试了试...结果不用说大家都懂的

那我们回过头来看看吧、恩先从字符串开始

Shift + F12



Address	Length	Type	String
.text:0040E5BA	000043F7	C	烫烫...
.rdata:0042301C	00000007	C	Good!\n
.rdata:00423024	00000006	C	pause
.rdata:0042302C	00000016	C	Sorry you are wrong!\n
.rdata:00423048	00000018	C	Please input your key:\n
.rdata:00423064	00000041	C	********* * ** * ** ** * ** ** * #* ** **** ** *********
.rdata:004230B4	0000000B	C	Too long!\n
.rdata:004230C4	00000016	C	Sorry,you are wrong!\n

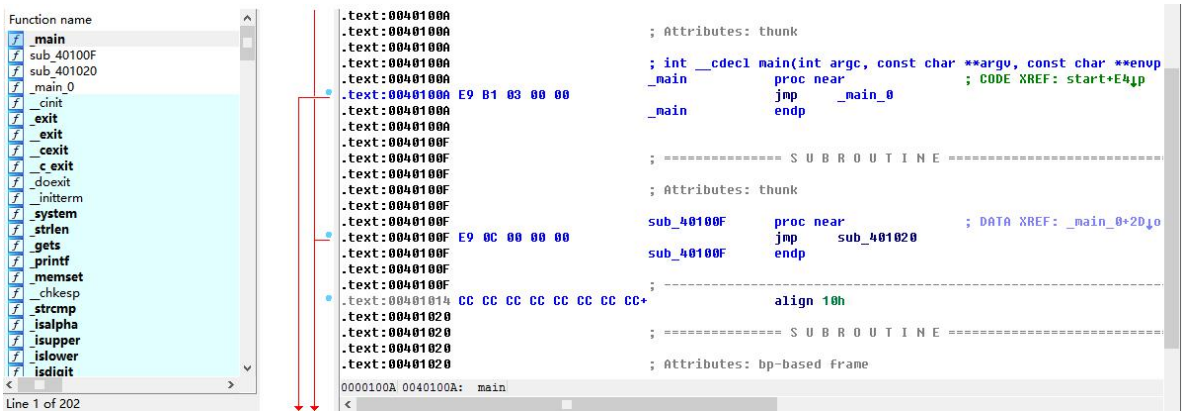
嗯？这一串*是什么东西？

```
3048 50 6C 65 61 73 65 20 69+aPleaseInputYou db 'Please input your key:',0Ah,0
3048 6E 70 75 74 20 79 6F 75+                                          ; DATA XREF: sub_401020+56↑o
3048 72 20 6B 65 79 3A 0A 00                                           ; _main_0+53↑o
3060 00                                 db    0
3061 00                                 db    0
3062 00                                 db    0
3063 00                                 db    0
3064 2A 2A 2A 2A 2A 2A 2A 2A+asc_423064 db '********* *    ** * ** ** * ** ** * #* ** **** **      *********',0
3064 2A 20 2A 20 20 20 20 2A+                                          ; DATA XREF: sub_401020+3C↑o
3005 00                                 db    0
```

找到它的引用地 sub_401020

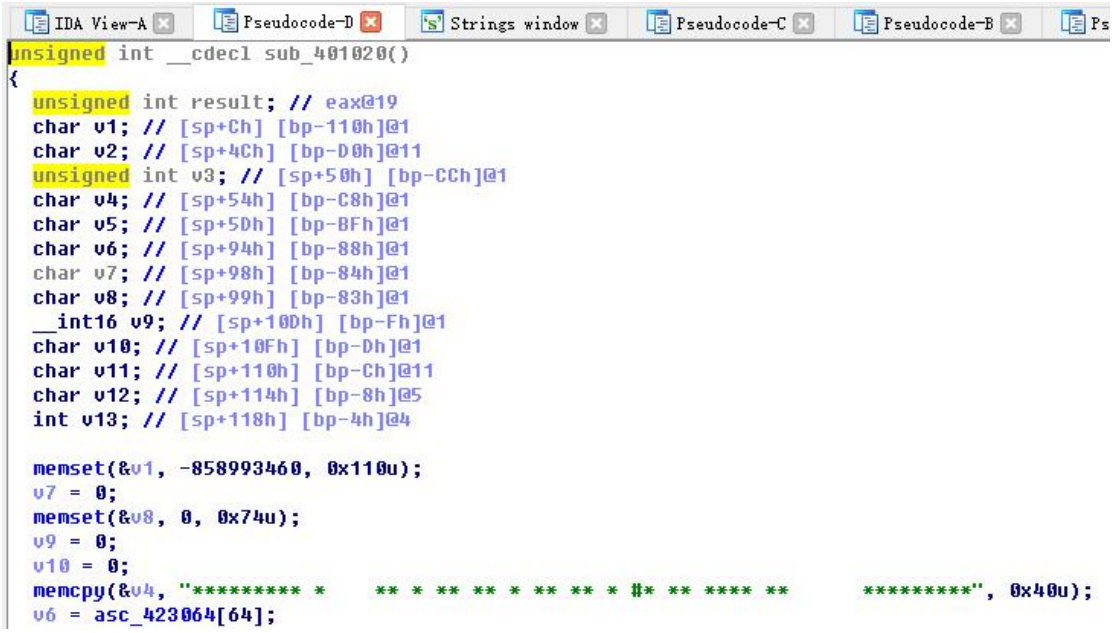
好像，有那么一点熟？

再看看 main 的那张图：



有没有发现什么？ 对，就是 main 函数在_main_0 后，下一个调的函数就是 sub_401020

F5 一下



```
  v3 = (unsigned int)&v5;
  printf("Please input your key:\n");
  gets(&v7);
  if ( strlen(&v7) != 22 )
  {
    printf("Sorry you are wrong!\n");
    system("pause");
    exit(1);
  }
  v13 = 0;
  do
  {
    v12 = *(&v7 + v13);
    if ( v12 != 107 && v12 != 106 && v12 != 104 && v12 != 108 )
    {
      printf("Sorry you are wrong!\n");
      system("pause");
      exit(2);
    }
    v11 = *(&v7 + v13);
    v2 = v11;
    if ( v11 == 104 )
    {
      --v3;
      if ( v3 < (unsigned int)&v4 || v3 > (unsigned int)&v6 || (result = *(_BYTE *)v3, result == 42) )
      {
        printf("Sorry you are wrong!\n");
        system("pause");
        exit(3);
      }
      if ( *(_BYTE *)v3 == 35 )
      {
LABEL_41:
        printf("Good!\n");
        system("pause");
        exit(0);
      }
    }
    else
    {
      if ( v2 == 106 )
      {
        v3 += 8;
        if ( v3 < (unsigned int)&v4 || v3 > (unsigned int)&v6 || *(_BYTE *)v3 == 42 )
        {
          printf("Sorry you are wrong!\n");
          system("pause");
          exit(3);
```

后面就不放了，意思基本一样

分析下它干了什么（注意那些地址都是负值，应该是存在堆栈里的关系）

（因为字符串在-0xC8~-0x88 之间，可以对应下看得懂的数字 0-64）

1、把那一串字符串放到了从 v4（-0xC8，0）开始的内存里（其实就是-0xC8~-0x88）

2、v3 值初始化为 v5 的地址（-0xBF，9） <-- 迷宫开始点

3、检查输入是不是 22 位，否，报错并退出

4、检查输入是不是 104、106、107、108 中的一个(h j k l)，否，报错并退出

5、如果是 104 h --> --v3 ; 106 j --> v3+8 ; 107 k --> v3-8 ; 108 l --> v3++

6、如果运算后 v3 的值超出了 v4-v6 的地址范围(-0xC8~-0x88，0~64)，或者 v3 的值是 42(换成 ASCII 码就是*)，报错并退出

7、如果运算后 v3 的值为 35(#)，输出 Good 并退出 <-- 迷宫结束点

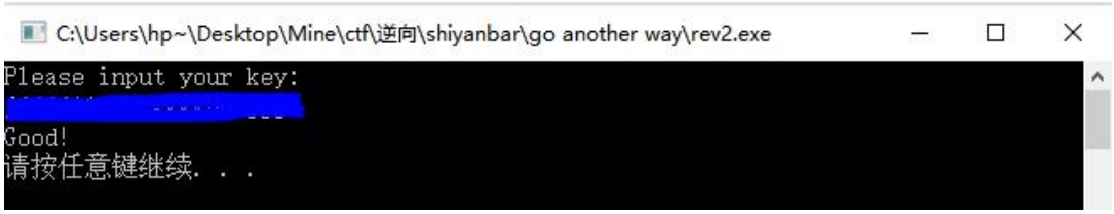
想到那串奇怪的字符串，想到走迷宫，再想到 8，and 题目描述里的最短步骤

********* *　　** * ** ** * ** ** * #* ** **** **　　*********

-->(word 好像对不齐，直接上图)

```
********
* *    *
* * ** *
* * ** *
* * #* *
* **** *
*      *
********
```

然后就，走迷宫嘛~



你的输入就是你的 flag 了~