逆向迷宫题总结（持续更新） 2020华南师大CTF新生赛maze，攻防世界新手区：NJUPT CTF 2017，BUUCTF：不一样的flag

原创

June_gjy 于 2021-02-02 22:03:23 发布

分类专栏： 网络安全 文章标签： 信息安全

CTF逆向入门：迷宫题学习记录（持续更新）

**

目录

（前言）

本文将不断更新以具体题目为载体，分析CTF逆向迷宫题的解题思路与方法。由于本人是新手，学识有限，若有错漏或不当之处，欢迎指出。

提示：以下是本篇文章正文内容，下面案例可供参考

一、 逆向迷宫题概述

迷宫题当然是走迷宫最后得到flag啦！

主要步骤：

1.分析程序的主函数，找到迷宫

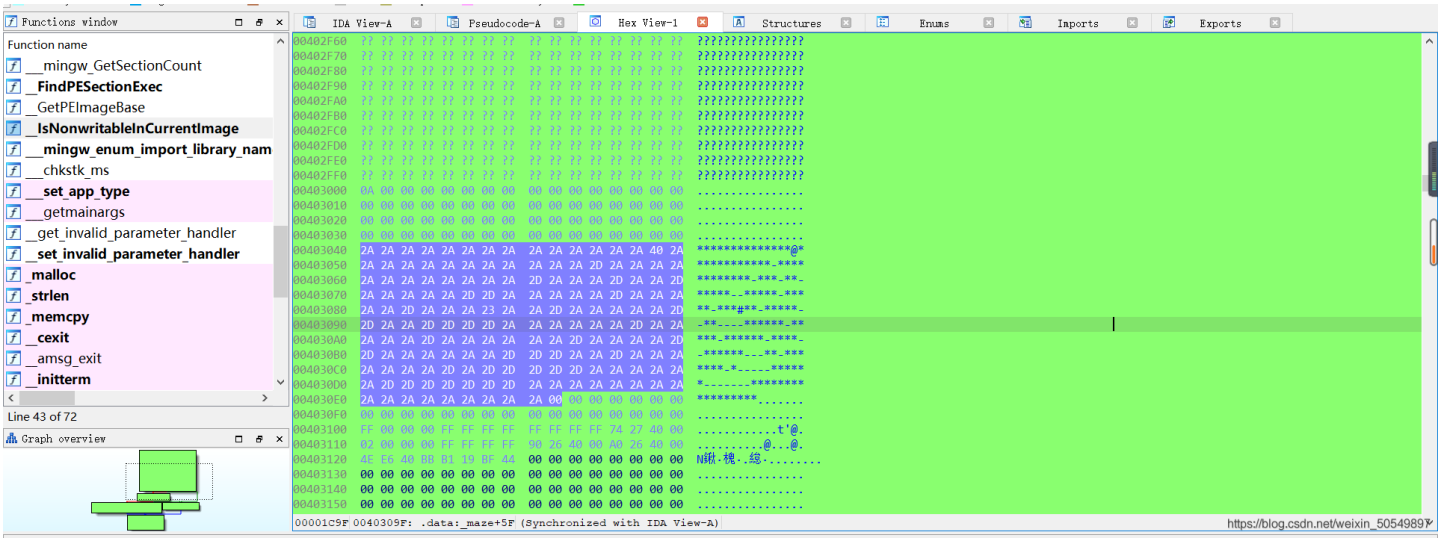
2分析程序，确定走迷宫的方法（上下左右...怎么走）

3写程序或手走迷宫，得到flag

二、 具体题目分析

1. 2019华南师大CTF新生赛maze

题目地址：https://github.com/scnu-sloth/hsctf-2019-freshmen



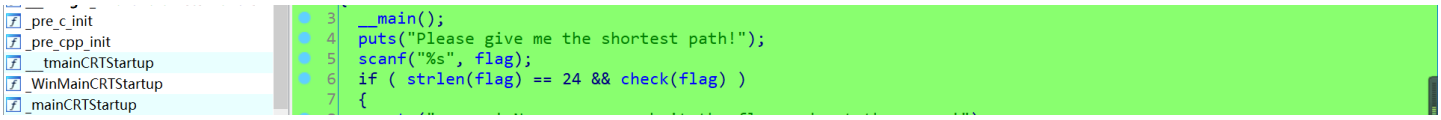
在hex-view里看到了迷宫，下一步是找走迷宫的规则和方法

点开check函数看到

```
bool __cdecl check(char *flag)
{
  char *v1; // eax
  int v2; // eax
  char *cur; // [esp+Ch] [ebp-4h]170个字符，其中有一个是空字符'0'

  cur = &maze[14];//这是一个含14*12的迷宫
  while ( *flag && *cur != '*' )//由此可见'*'是迷宫的墙
  {
    v1 = flag++;
    v2 = *v1;
    if ( v2 == 'd' )
    {
      ++cur;//d为向右走一步
    }
    else if ( v2 > 'd' )
    {
      if ( v2 == 's' )
      {
        cur += 13;//s为向右13步，在本二维数组中为向左下方一步或向右13步
      }
      else
      {
        if ( v2 != 'w' )
          return 0;
        cur -= 13;//w为向右上方一步或向左13步
      }
    }
    else
    {
      if ( v2 != 'a' )
        return 0;
      --cur;//a为向左一步
    }
  }
  return *cur == '#';//'#'是迷宫终点
}
```

由题意，这是个14*12的迷宫，@是起点，#是终点，flag走迷宫的路径

d:向前一步，a:向后一步，s:向前13步，w:向后13步



由main函数可见，flag长度即走迷宫步数一共为24步

手画迷宫如下，X表示不可走，O表示可走

写代码生成迷宫

```cpp
int main()
{
 char s[]="**************@***********-***********-***-**-******--*****-*****-***#**-******--**----******-*****-*
*****-****--*******---**-*******-*-----******-------*****************";
 int i,j;
 for(i=0;i<12;i++)
 {
  for(j=0;j<14;j++)
   {
    if(s[i*14+j]=='-')
    cout<<"0"<<' ';
    else if(s[i*14+j]=='*')
    cout<<"X"<<' ';
    else
    cout<<s[i*14+j]<<' ';
   }
  cout<<endl;
 }
 return 0;
}
```

```
C:\Users\June\Desktop\C++总\洛谷\未命名1.exe
X X X X X X X X X X X X X X
@ X X X X X X X X X X X X 0
X X X X X X X X X X X 0 X
X X 0 X X 0 X X X X X 0 0 X
X X X X 0 X X X X X 0 X X X
# X X 0 X X X X X 0 0 X X 0
0 0 0 X X X X X X 0 X X X X
X 0 X X X X X X 0 X X X X 0
0 X X X X X X 0 0 0 X X 0 X
X X X X X X 0 X 0 0 0 0 0 X
X X X X X 0 0 0 0 0 0 0 X X
X X X X X X X X X X X X X X


--------------------------------
Process exited after 0.6049 seconds with return value 0
请按任意键继续. . .
```


用BFS走迷宫（代码如下）

```cpp
#include<bits/stdc++.h>
using namespace std;
char mp[12*15];
int vis[12*15];
int dir[4]={1,-1,13,-13};
char S[4]={'d','a','s','w'};
struct node
{
 int x;
 string s;
};
queue<node> Q;
void bfs()
{
 node tmp;
 tmp.x=15;tmp.s="";
 Q.push(tmp);
 while(!Q.empty())
 {
  node now=Q.front();
  Q.pop();
  vis[now.x]=1;
  if(mp[now.x]=='#')
  {
   cout<<"flag{"<<now.s<<"}"<<endl;
   return;
  }
  for(int k=0;k<4;k++)
  {
   int ux=now.x+dir[k];
   if(ux<1||ux>168||mp[ux]=='X'||vis[ux]==1)
   continue;
   tmp.x=ux;
   tmp.s=now.s+S[k];
   Q.push(tmp);
  }
 }
}
int main()
{
 int t=0;
 for(int i=1;i<=12;i++)
 {
  for(int j=1;j<=14;j++)
  {
   cin>>mp[++t];
  }
 }
 memset(vis,0,sizeof(vis));
 bfs();
 return 0;
}
```

输出结果

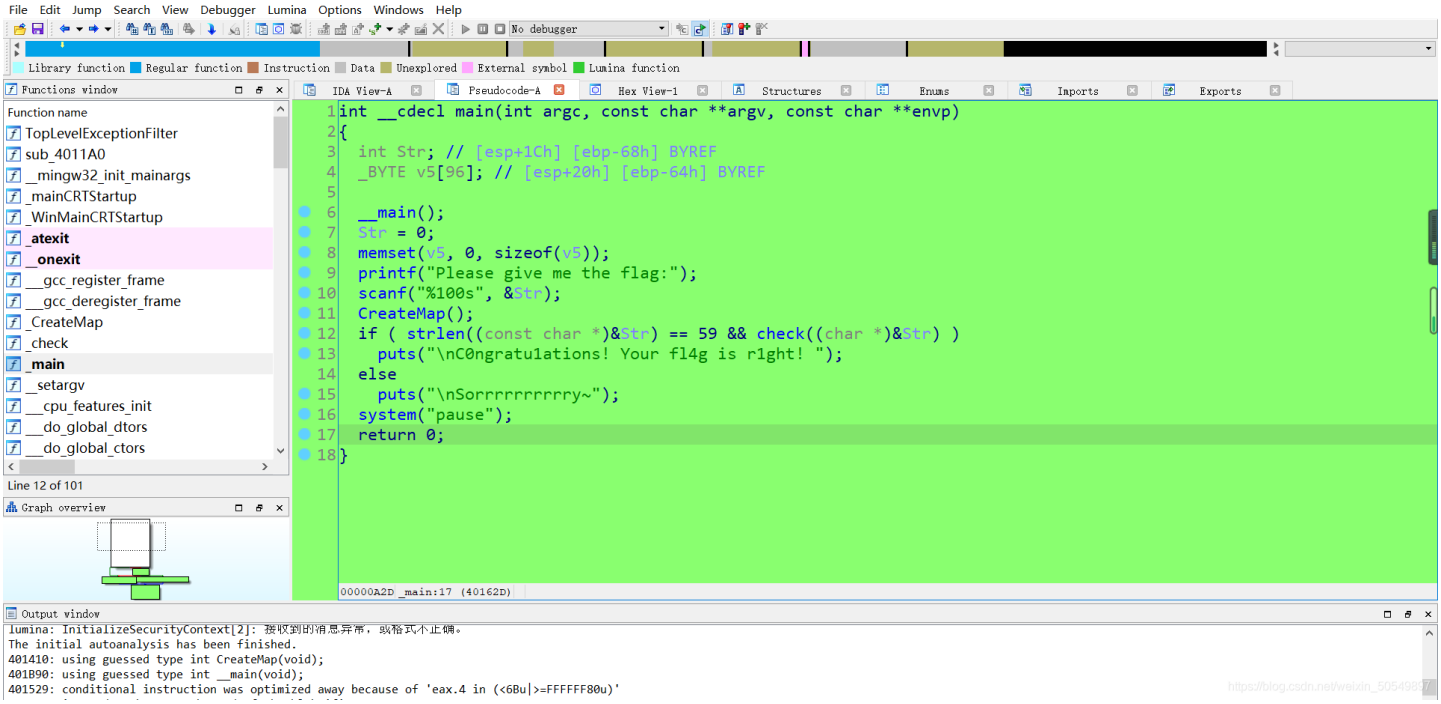
```
flag{sssssdsssddsdddwwdwwaaaw}

--------------------------------

Process exited after 5.285 seconds with return value 0
请按任意键继续. . .
```

flag{sssssdsssddsdddwwdwwaaaw}

2.2020华南师大CTF新生赛maze

题目地址https://github.com/scnu-sloth/hsctf-2020-freshmen

拖进IDA反编译，查看main函数



flag长度限制为59,应该是迷宫的最短路径

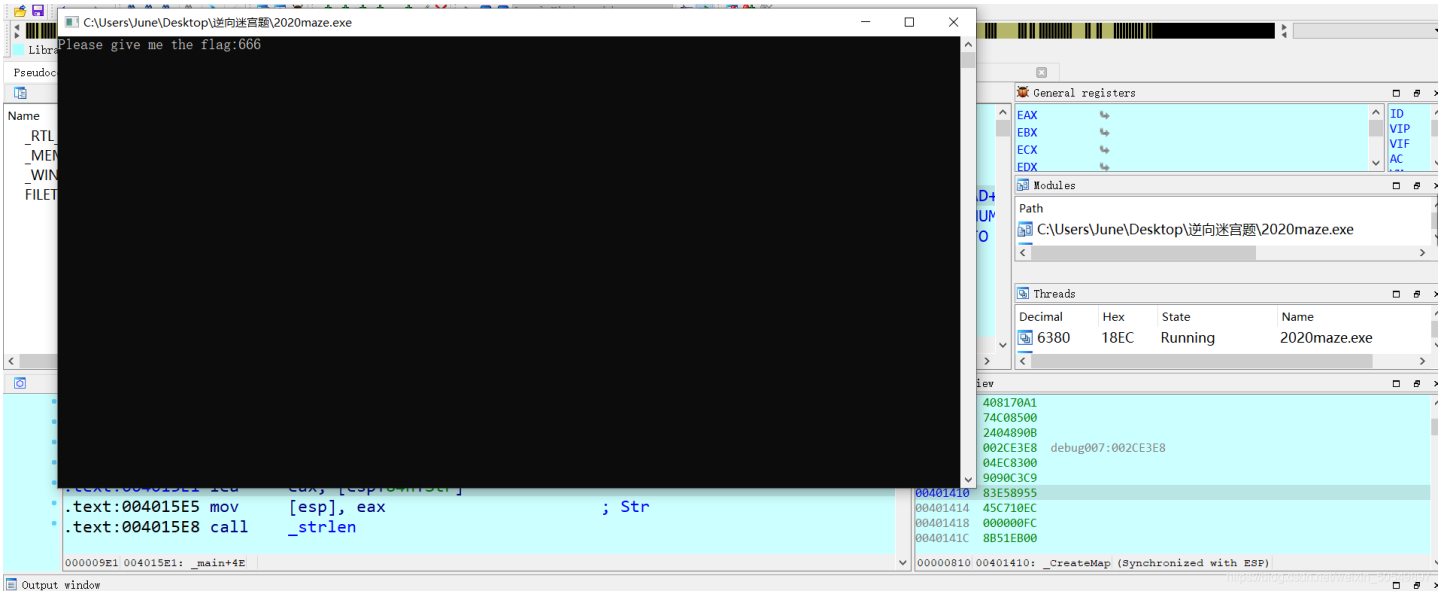
并且发现了一个CreateMap函数，应该里面有关于迷宫的信息。点开来看

```
int CreateMap()
{
  int result; // eax
  int v1; // [esp+4h] [ebp-Ch]
  unsigned __int16 v2; // [esp+Ah] [ebp-6h]
  int i; // [esp+Ch] [ebp-4h]

  for ( i = 0; i <= 15; ++i )
  {
    v2 = num[i];//num数组有16个数
    v1 = 1;//v2是无符号整型
    do
    {
      map[16 * i + 16 - v1] = v2 & 1;//map是二维数组，有256个数。
      v2 >>= 1;//相当于v2=v2/2^1
      result = v1++;
    }//这是转二进制
    //v2按位与1，提取二进制最后一位数，v2在除以2，意思是不断把
    //二进制每一位提取出来。相当于进制转换。
    while ( result && v1 <= 16 );
  }
  return result;
}
```

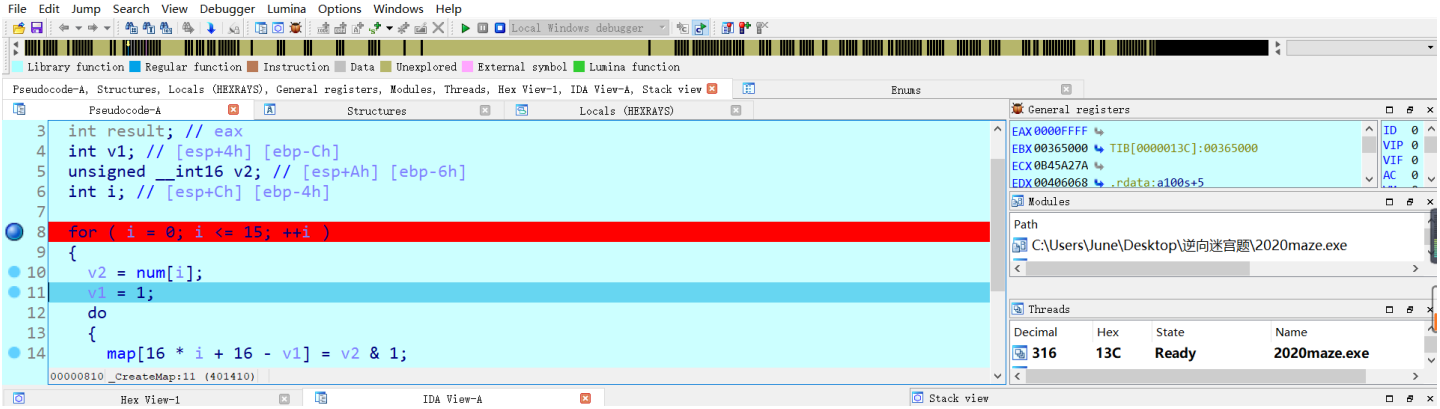
这道迷宫题与众不同，要我们根据函数生成迷宫

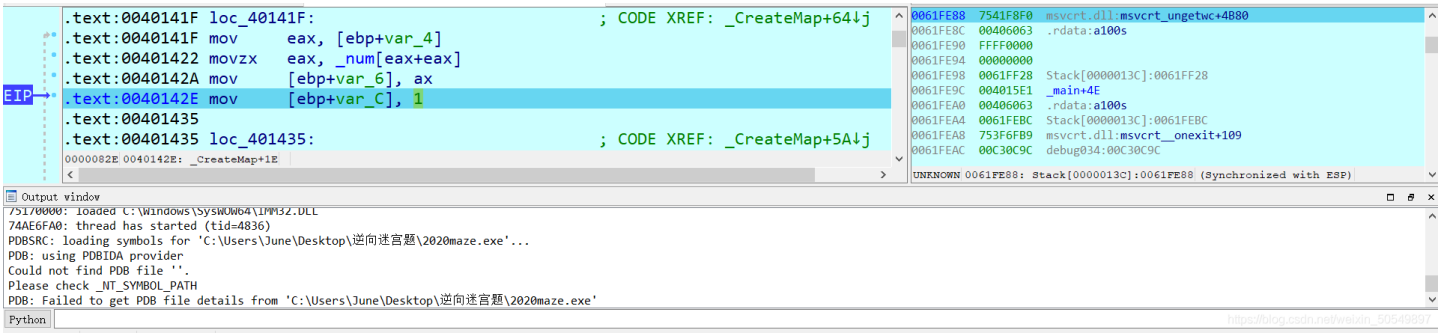
该程序的意思是把num中的每个数转为无符号整型再**转为二进制**储存再map数组中。原本16个数字变成16*16=256个数字。根据题意该迷宫是个16X16迷宫。因此我们想要得到迷宫，**必须知道循环中每个v2的值**。我想到通过动态调试一一记录下16个v2的值（该方法要有耐心，肯定有更好的方法）

接下来进行windows本地动态调试

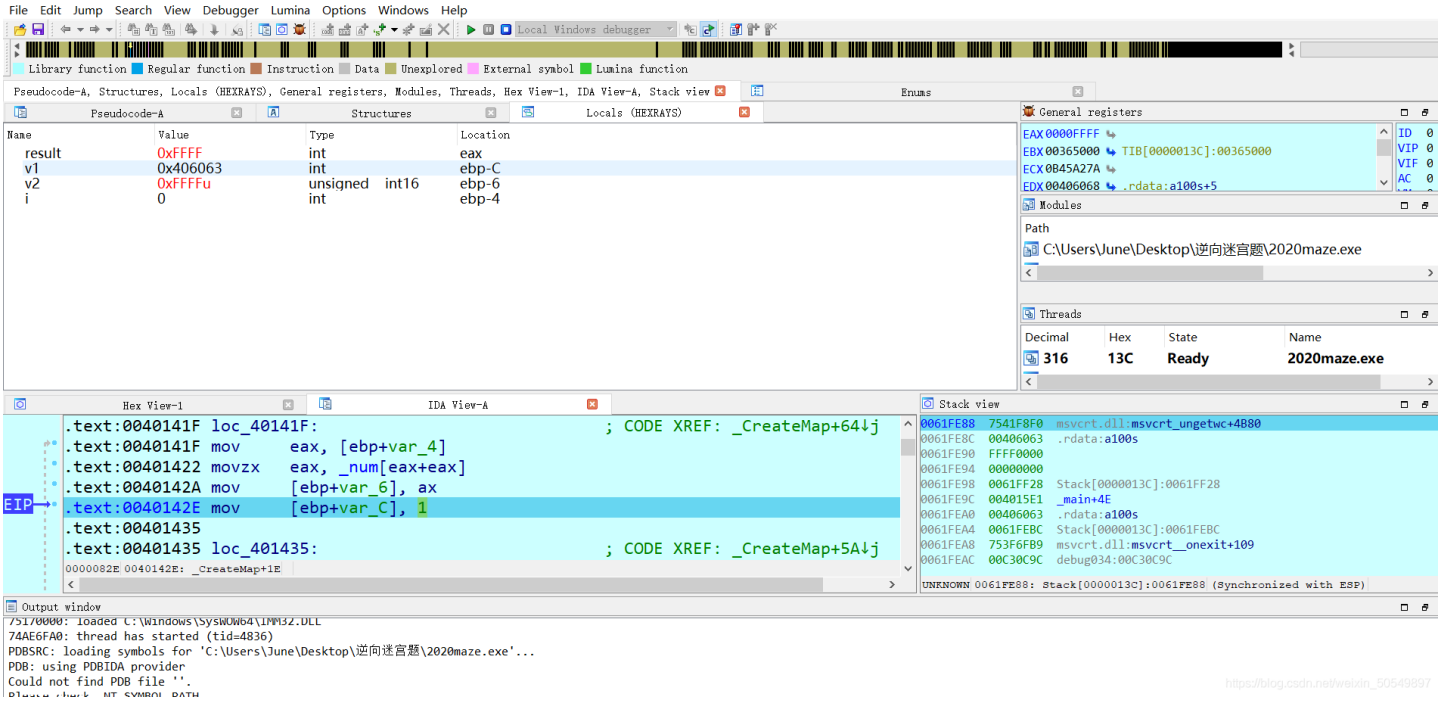


设置断点





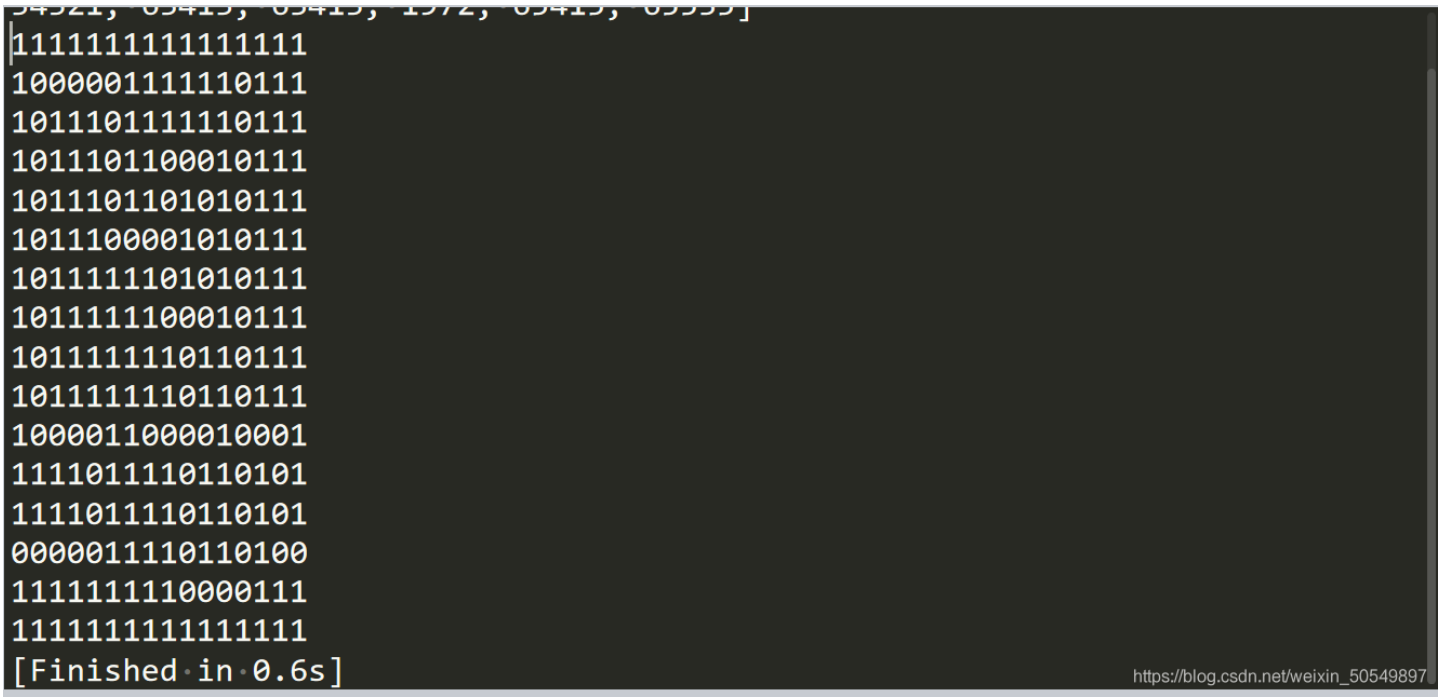
在locals里查看变量的值



关于IDA动态调试可看我的这篇文章逆向工程入门：IDAwindows本地动态调试，linux远程动态调试及虚拟机配置

写python脚本输出迷宫

```
a=[0xffff,0x83F7,0xBBF7,0xBB17,0xBB57,0xB857,0xBF57,0xBF17,0xBFB7,0xBFB7,0x8611,0xF7B5,0xF7B5,0x7B4,0xFF87,0xFFF
F]
print(a)
for i in a:
 i=bin(i)
 print(i)
```



好了，迷宫到手了，显然入口是map[13][0],出口是map[13][15]，0可走，1不可走。让我们回到IDA，接下来要看看怎么玩

点开check函数

```
int __cdecl check(char *Str)
{
  int v2; // eax
  char Destination[4]; // [esp+1Ch] [ebp-7Ch] BYREF
  char v4[96]; // [esp+20h] [ebp-78h] BYREF
  size_t v5; // [esp+80h] [ebp-18h]
  int v6; // [esp+84h] [ebp-14h]
  int v7; // [esp+88h] [ebp-10h]
  int i; // [esp+8Ch] [ebp-Ch]

  v5 = strlen(Str);
  *(_DWORD *)Destination = 0;
  memset(v4, 0, sizeof(v4));
  strcpy(Destination, Str);
  v4[1] = 0;
  if ( strcmp(Destination, "flag{") || Str[v5 - 1] != '}' )
    return 0;
  v7 = 13;
  v6 = 0;//表示起点为map[13][0]
  for ( i = 5; i < (int)(v5 - 1); ++i )
  {
    v2 = Str[i];
    if ( v2 == 'l' )
    {
      ++v6;//向右走
    }
    else
    {
      if ( v2 > 'l' )
        return 0;
      switch ( v2 )
      {
        case 'k':
          --v7;//向上走
          break;
        case 'h':
          --v6;//向左走
          break;
        case 'j':
          ++v7;//向下走
          break;
        default:
          return 0;
      }
    }
    if ( map[16 * v7 + v6] )//v6表示左右，v7表示上下
      return 0;
  }
  return 1;
}
```

用BFS走迷宫（代码如下）

```cpp
#include<bits/stdc++.h>
using namespace std;
int dir[4][2]={{1,0},{-1,0},{0,1},{0,-1}};
char s[4]={'l','h','j','k'};
int mp[17][17];
int vis[17][17];
int startx=14;int starty=1;
int flagx=14,flagy=16;
struct node
{
 int x,y;
 string step;
};
queue<node> Q;
void bfs()
{
 node tmp;
 tmp.x=startx,tmp.y=starty,tmp.step="";
 Q.push(tmp);
 while(!Q.empty())
 {
  node now=Q.front();
  Q.pop();
  vis[now.x][now.y]=1;
  if(now.x==flagx&&now.y==flagy)
  {
   cout<<now.step<<endl;
   return;
  }
  for(int k=0;k<4;k++)
  {
   int ux=now.x+dir[k][1];
   int uy=now.y+dir[k][0];
   if(vis[ux][uy]==1||mp[ux][uy]==1||ux<1||ux>16||uy<1||uy>16)
   continue;
   node tmp;
   tmp.x=ux,tmp.y=uy,tmp.step=now.step+s[k];
   Q.push(tmp);
  }
 }
}
int main()
{
 memset(vis,0,sizeof(vis));
 for(int i=1;i<=16;i++)
 {
  string S;
  cin>>S;
  for(int j=1;j<=16;j++)
  {
   if(S[j-1]=='0') mp[i][j]=0;
   else mp[i][j]=1;
  }
 }
 bfs();
 return 0;
}
```

输出得到

```
llll kkkhhhhkkkkkkkkkk llll jjjj lll jj l jjjjjjj lll kkkk ll jjj l

--------------------------------

Process exited after 13.51 seconds with return value 0
请按任意键继续. . .
```

Get the flag! flag{llllkkkhhhhkkkkkkkkkklllljjjjlllljjljjjjjjjlllkkkklljjjl}

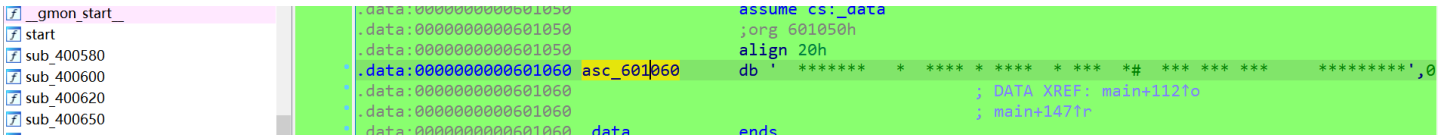
走迷宫路径如下

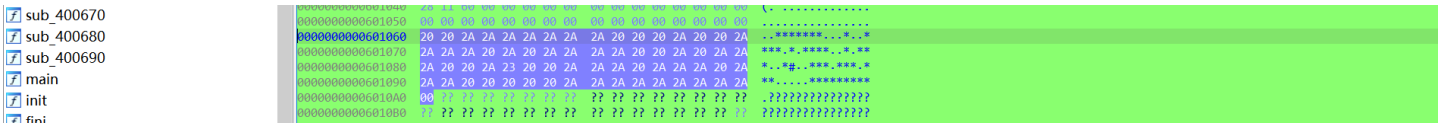
```
1111111111111111
1000001111110111
1011101111110111
1011101100010111
1011101101010111
1011100001010111
1011111101010111
1011111100010111
1011111101110111
1011111101110111
1000001100010001
1111011110110101
1111011110110101
1111011110110101
1111111110000111
1111111111111111
[Finished in 0.5s]
```


3.攻防世界新手区：NJUPT CTF 2017 maze

题目地址：https://adworld.xctf.org.cn/task/answer?type=reverse&number=4&grade=0&id=5084&page=1

拖进IDA发现迷宫所在





分析main函数

```
int v9; // [rsp+0h] [rbp-28h] BYREF
int v10[9]; // [rsp+4h] [rbp-24h] BYREF

v10[0] = 0;
v9 = 0;//迷宫从最左上角开始走
puts("Input flag:");
scanf("%s", &s1);
if ( strlen(&s1) != 24 || strncmp(&s1, "nctf{", 5uLL) || *(&byte_6010BF + 24) != '}' )//如果是nctf{开头且长度为2
```

```
4就进入下一步
  {
LABEL_22://要盯紧它，判断什么情况下会失败
    puts("Wrong flag!");
    exit(-1);
  }
  v3 = 5LL;
  if ( strlen(&s1) - 1 > 5 )
  {
    while ( 1 )
    {
      v4 = *(&s1 + v3);
      v5 = 0;
      if ( v4 > 'N' )
      {
        if ( (unsigned __int8)v4 == 'O' )
        {
          v6 = sub_400650(v10);//这里对v10进行了操作
          goto LABEL_14;
        }
        if ( (unsigned __int8)v4 == 'o' )
        {
          v6 = sub_400660(v10);//这里对v10进行了操作
          goto LABEL_14;
        }
      }
      else
      {
        if ( (unsigned __int8)v4 == '.' )
        {
          v6 = sub_400670(&v9);//这里对&v9进行了操作
          goto LABEL_14;
        }
        if ( (unsigned __int8)v4 == '0' )
        {
          v6 = sub_400680(&v9);//这里对&v9进行了操作
LABEL_14:
          v5 = v6;
          goto LABEL_15;
        }
      }
LABEL_15:
      if ( !(unsigned __int8)sub_400690((__int64)asc_601060, v10[0], v9) )//这里要关注障碍检测，还有非常重要的东西
        goto LABEL_22;
      if ( ++v3 >= strlen(&s1) - 1 )
      {
        if ( v5 )
          break;
LABEL_20:
        v7 = "Wrong flag!";
        goto LABEL_21;
      }
    }
  }
  if ( asc_601060[8 * v9 + v10[0]] != '#' )//终点为#
    goto LABEL_20;
  v7 = "Congratulations!";
LABEL_21:
  puts(v7);
```

```
  return 0LL;
}
```

由伪代码可见，这是个8*8的迷宫，有四个判断条件，分别进入了四个函数。
‘O’,‘o’,‘0’,’.'这四个字符分别控制不同方向
O左，o右，0下，.上（稍后会说怎么判断）
四个方向函数如下

```
bool __fastcall sub_400650(_DWORD *a1)//O
{
  int v1; // eax

  v1 = (*a1)--;//向左一步
  return v1 > 0;//防止越界
}
```

```
bool __fastcall sub_400660(int *a1)//o
{
  int v1; // eax

  v1 = *a1 + 1;//向右一步
  *a1 = v1;
  return v1 < 8;
}
```

```
bool __fastcall sub_400670(_DWORD *a1)//.
{
  int v1; // eax

  v1 = (*a1)--;//向上一步
  return v1 > 0;
}
```

```
bool __fastcall sub_400680(int *a1)//0
{
  int v1; // eax

  v1 = *a1 + 1;
  *a1 = v1;//向下一步
  return v1 < 8;
}
```

可是怎么分辨v9和v10各自控制的是上下还是左右呢？
点开sub_400690函数，看到

```
__int64 __fastcall sub_400690(__int64 a1, int a2, int a3)
{//a1是数组的首地址，a2是v10,a3是v9
  __int64 result; // rax

  result = *(unsigned __int8 *)(a1 + a2 + 8LL * a3);//a3乘以8表明v10在二维数组中表示上下
  LOBYTE(result) = (_DWORD)result == ' ' || (_DWORD)result == '#';
  return result;
}
```

从这里我们得知v10表示行，就是控制上下。v9表示列，就是控制左右。
而且只有‘ ’和#可以走，否则返回return 0。’*'是迷宫的墙。
写代码生成迷宫，0可走，X为墙。

```cpp
#include<iostream>
using namespace std;
int main()
{
 char s[]="  *******   *  **** * ****  * ***  *#  *** *** ***     *********";
 int i,j;
 for(i=0;i<8;i++)
 {
  for(j=0;j<8;j++)
   {
    if(s[i*8+j]==' ')
    cout<<"0"<<' ';
    else if(s[i*8+j]=='*')
    cout<<"X"<<' ';
    else
    cout<<"#"<<' ';
   }
  cout<<endl;
 }
 return 0;
}
```

C:\Users\June\Desktop\C++总\洛谷\未命名1.exe

```
0 X X X X X X
X 0 0 X 0 0 X
X X X X 0 X X
X X 0 0 X 0 X X
X 0 0 X # 0 X
X X X X X X X
X X 0 0 0 0 X
X X X X X X X X

--------------------------------
Process exited after 2.923 seconds with return value 0
请按任意键继续. . .
```


走迷宫得到flag为 **nctf{o0oo00O000oooo...OO}**

4.BUUCTF：不一样的flag

题目地址：https://buuoj.cn/challenges

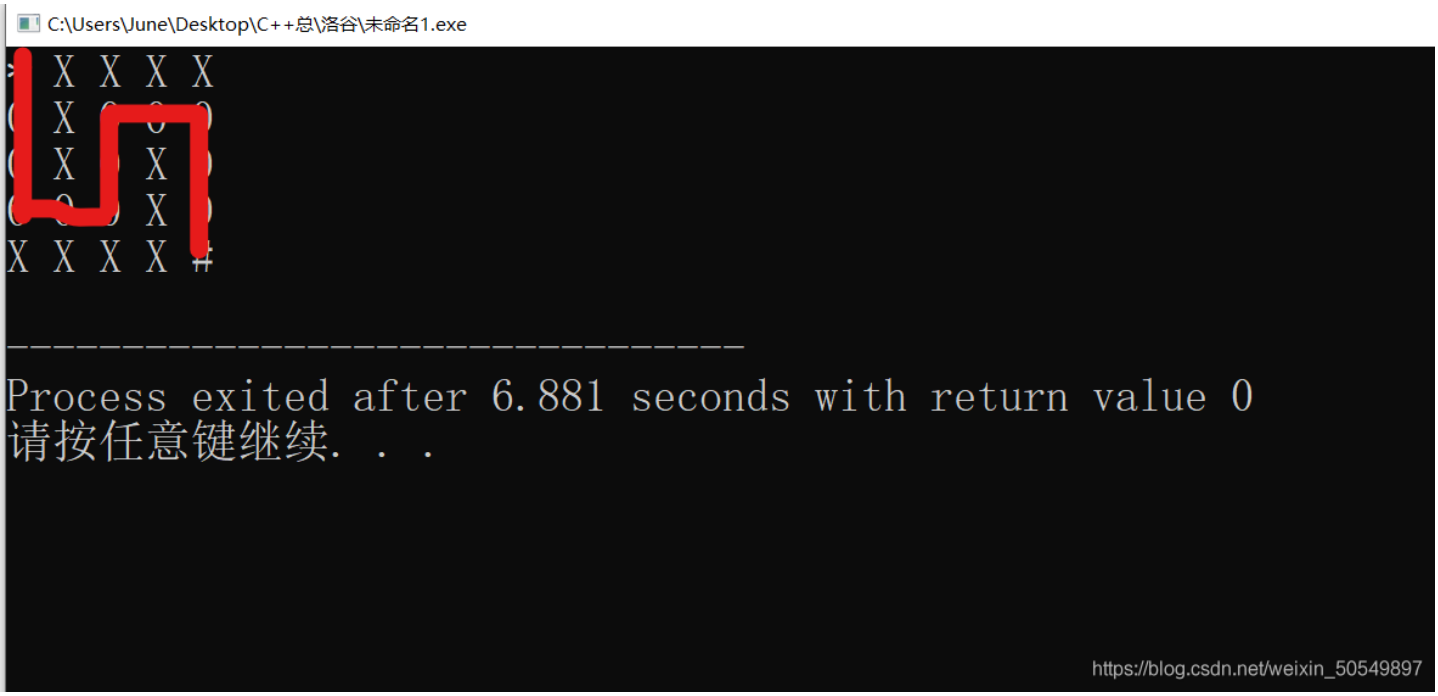
拖进IDA查看main函数

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  char v3[29]; // [esp+17h] [ebp-35h] BYREF
  int v4; // [esp+34h] [ebp-18h]
  int v5; // [esp+38h] [ebp-14h] BYREF
  int i; // [esp+3Ch] [ebp-10h]
  _BYTE v7[12]; // [esp+40h] [ebp-Ch] BYREF

  __main();
  v4 = 0;
  strcpy(v3, "*1111010000101000010111 1#");//这就是迷宫
  while ( 1 )
  {
    puts("you can choose one action to execute");
    puts("1 up");
    puts("2 down");
    puts("3 left");
    printf("4 right\n:");//控制方向
    scanf("%d", &v5);
    if ( v5 == 2 )
    {
      ++*(_DWORD *)&v3[25];
    }
    else if ( v5 > 2 )
    {
      if ( v5 == 3 )
      {
        --v4;
      }
      else
      {
        if ( v5 != 4 )
LABEL_13:
          exit(1);
        ++v4;
      }
    }
    else
    {
      if ( v5 != 1 )
        goto LABEL_13;
      --*(_DWORD *)&v3[25];
    }
    for ( i = 0; i <= 1; ++i )
    {
      if ( *(int *)&v3[4 * i + 25] < 0 || *(int *)&v3[4 * i + 25] > 4 )
        exit(1);
    }
    if ( v7[5 * *(_DWORD *)&v3[25] - 41 + v4] == '1' )
      exit(1);//是个5*5的迷宫且1为墙
    if ( v7[5 * *(_DWORD *)&v3[25] - 41 + v4] == '#' )
    {//‘#’是迷宫终点
      puts("\nok, the order you enter is the flag!");
      exit(0);
    }
  }
}
```

一目了然，显然是一道迷宫题。
1，2，3，4分别控制上下左右

```
using namespace std;
int main()
{
 char s[]="*11110100001010000101111#";
 int i,j;
 for(i=0;i<5;i++)
 {
  for(j=0;j<5;j++)
   {
    if(s[i*5+j]=='0')
    cout<<"0"<<' ';
    else if(s[i*5+j]=='1')
    cout<<"X"<<' ';
    else
    cout<<s[i*5+j]<<' ';
   }
  cout<<endl;
 }
 return 0;
}
```

走迷宫



Get the flag!: flag{222441144222}

三，总结

希望新接触reverse的读者能通过CTF的迷宫题产生对CTF的兴趣。

本人其它文章链接

BUUCTF reverse：[GXYCTF2019]luck_guy,findit,简单注册器题解

封神台靶场尤里的复仇I第一第二第五第六第七章解题思路(持续更新）

ctfhub:网鼎杯第一场2018 reverse-beijing题解

逆向工程入门：IDAwindows本地动态调试，linux远程动态调试及虚拟机配置