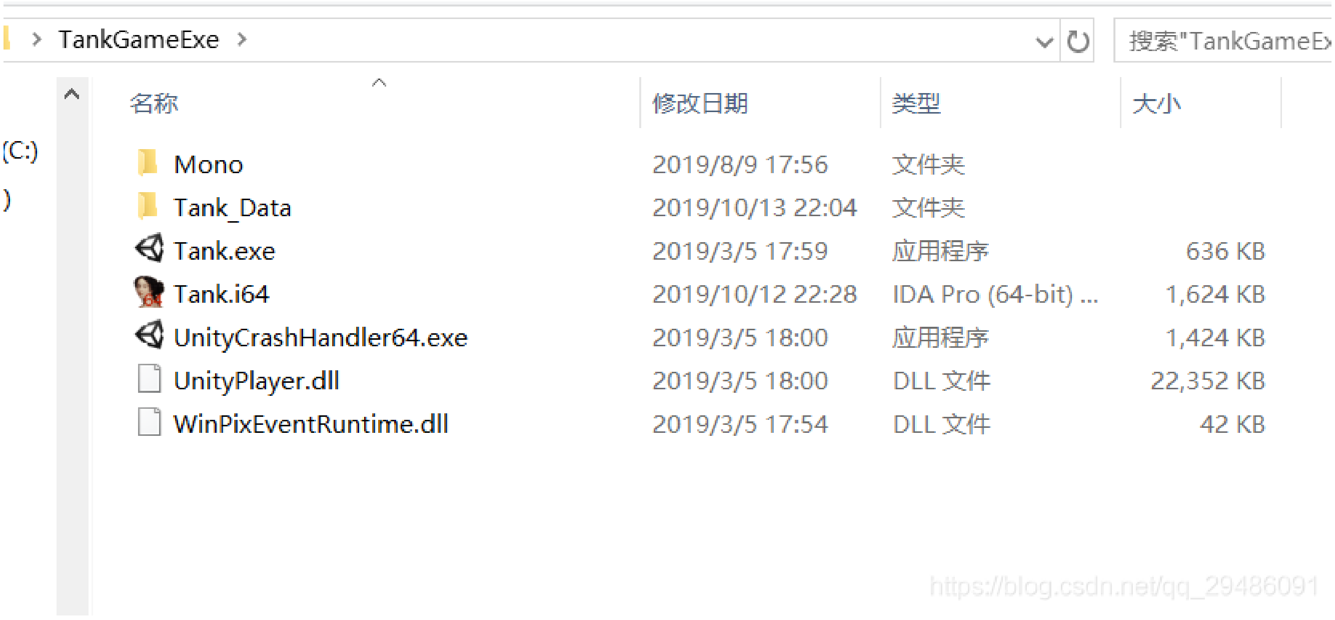
Roar CTF 2019 坦克大战 Writeup

原创

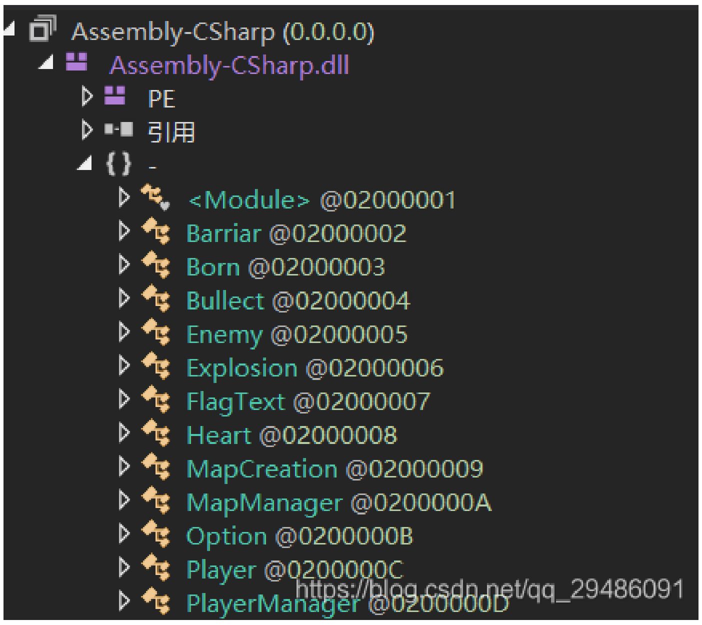
Secz 于 2019-10-15 19:56:13 发布 1308 收藏 4
文章标签： CTF Roar TankGame 坦克大战
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/qq_29486091/article/details/102573099
版权

下载压缩包 解压看到



名称	修改日期	类型	大小
Mono	2019/8/9 17:56	文件夹	
Tank_Data	2019/10/13 22:04	文件夹	
Tank.exe	2019/3/5 17:59	应用程序	636 KB
Tank.i64	2019/10/12 22:28	IDA Pro (64-bit) ...	1,624 KB
UnityCrashHandler64.exe	2019/3/5 18:00	应用程序	1,424 KB
UnityPlayer.dll	2019/3/5 18:00	DLL 文件	22,352 KB
WinPixEventRuntime.dll	2019/3/5 17:54	DLL 文件	42 KB

可知该游戏由Unity3D编写,由于没有相关逆向经验,上网查阅得知游戏代码部分都在**Assembly-CSharp.dll**文件中,通过dnspy可以打开此类型的dll.



```
Assembly-CSharp (0.0.0.0)
  Assembly-CSharp.dll
    PE
    引用
    { } -
      <Module> @02000001
      Barriar @02000002
      Born @02000003
      Bullect @02000004
      Enemy @02000005
      Explosion @02000006
      FlagText @02000007
      Heart @02000008
      MapCreation @02000009
      MapManager @0200000A
      Option @0200000B
      Player @0200000C
      PlayerManager @0200000D
```

直接查看跟题目相关的函数,在MapManager下有名为WinGame的函数.源码如下

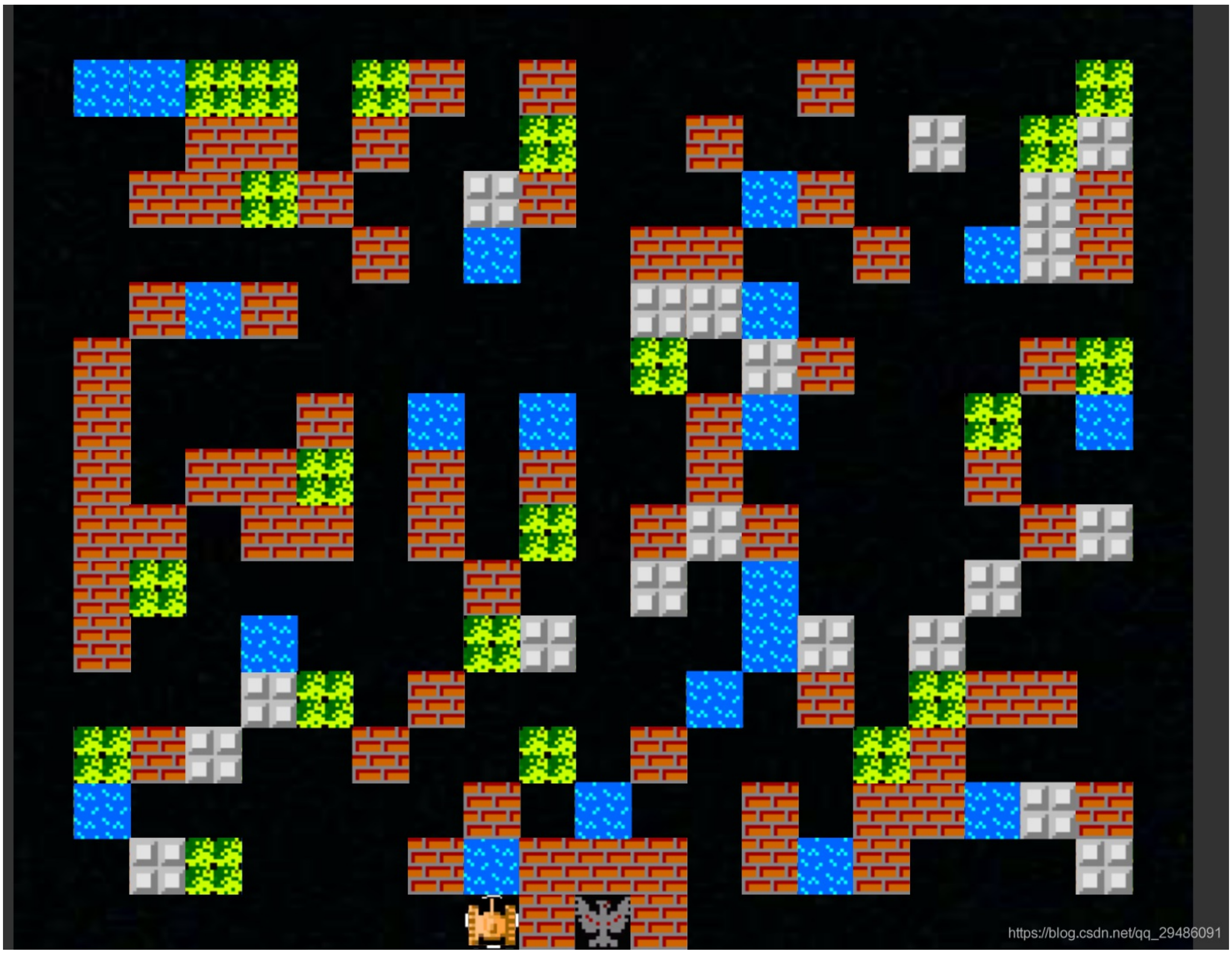
```
// MapManager
// Token: 0x06000029 RID: 41 RVA: 0x00003050 File Offset: 0x00001250
public static void WinGame()
{
 if (!MapManager.winGame && (MapManager.nDestroyNum == 4 || MapManager.nDestroyNum == 5))
 {
  string text = "clearlove9";
  for (int i = 0; i < 21; i++)
  {
   for (int j = 0; j < 17; j++)
   {
    text += MapManager.MapState[i, j].ToString();
   }
  }
  string a = MapManager.Sha1(text);
  if (a == "3F649F708AAFA7A0A94138DC3022F6EA611E8D01")
  {
   FlagText._instance.gameObject.SetActive(true);
   FlagText.str = "RoarCTF{wm-" + MapManager.Md5(text) + "}";
   MapManager.winGame = true;
  }
 }
}
```

具体就是对数组进行字符化累加后,判断Hash值是否与给定值相等.
其中nDestroyNum和MapManager.MapState是关键所在.
先看MapManager.MapState(在MapManager.init()中定义.

```
// MapManager
// Token: 0x06000028 RID: 40 RVA: 0x000027AC File Offset: 0x000009AC
public static void Init()
{
 MapManager.MapState = new int[,]
 {
  {
   8,
   8,
   8,
   8,
   8,
   8,
   8,
   8,
   8,
   8,
   8,
   8,
   8,
   8,
   8,
   8,
   8
  },
  {
   8,
   8,
   4,
   5,
   8,
   1,
   1,
   1,
   1,
   1,
   1,
   8,
   8,
   8,
   8,
   4,
   8
  },
  {
   8,
   2,
   8,
   1,
   8,
   8,
   5,
   1,
   8,
   8,
   8,
   1,
   8,
   1,
   8,
   4,
   8
  },
  {
   8,
   5,
   8,
   2,
   8,
   8,
   8,
   8,
   1,
   8,
   8,
   4,
   8,
   1,
   1,
   5,
   8
  },
  {
   8,
   8,
   8,
   8,
   2,
   4,
   8,
   1
  1,
  8,
  8,
  1,
  8,
  5,
  1,
  5,
  8
 },
 {
  8,
  8,
  8,
  8,
  5,
  8,
  8,
  1,
  5,
  1,
  8,
  8,
  8,
  1,
  8,
  8,
  8
 },
 {
  8,
  8,
  8,
  1,
  8,
  8,
  8,
  8,
  8,
  8,
  8,
  8,
  1,
  8,
  1,
  5,
  8
 },
 {
  8,
  1,
  8,
  8,
  1,
  8,
  8,
  1,
  1,
  4,
  8,
  8,
  8,
  8,
  8,
  1,
  8
 },
 {
  8,
  4,
  1,
  8,
  8,
  5,
  1,
  8,
  8,
  8,
  8,
  8,
  4,
  2,
  8,
  8,
  8
 },
 {
  1,
  1,
  8,
  5,
  8,
  2,
  8,
  5,
  1,
  4,
  8,
  8,
  8,
  1,
  5,
  1,
  8
 },
 {
  0,
  1,
  4,
  8,
  8,
  8,
  8,
  8,
  8,
  8,
  8,
  8,
   8,
   8,
   8,
   8,
   8
  },
  {
   1,
   1,
   8,
   1,
   8,
   8,
   2,
   1,
   8,
   8,
   5,
   2,
   1,
   8,
   8,
   8,
   8
  },
  {
   8,
   8,
   8,
   8,
   4,
   8,
   8,
   2,
   1,
   1,
   8,
   2,
   1,
   8,
   1,
   8,
   8
  },
  {
   8,
   1,
   1,
   8,
   8,
   4,
   4,
   1,
   8,
   4,
   2,
   4,
   8,
   4,
   8,
  8,
  8
 },
 {
  8,
  4,
  8,
  8,
  1,
  2,
  8,
  8,
  8,
  8,
  1,
  8,
  8,
  1,
  8,
  1,
  8
 },
 {
  8,
  1,
  1,
  5,
  8,
  8,
  8,
  8,
  8,
  8,
  8,
  8,
  1,
  8,
  8,
  8,
  8
 },
 {
  8,
  8,
  1,
  1,
  5,
  2,
  8,
  8,
  8,
  8,
  8,
  8,
  8,
  8,
  2,
  8,
  8
```

```
},
{
 8,
 8,
 4,
 8,
 1,
 8,
 2,
 8,
 1,
 5,
 8,
 8,
 4,
 8,
 8,
 8,
 8
},
{
 8,
 8,
 2,
 8,
 1,
 8,
 8,
 1,
 8,
 8,
 1,
 8,
 2,
 2,
 5,
 8,
 8
},
{
 8,
 2,
 1,
 8,
 8,
 8,
 8,
 2,
 8,
 4,
 5,
 8,
 1,
 1,
 2,
 5,
 8
},
{
 8,
    8,
    8,
    8,
    8,
    8,
    8,
    8,
    8,
    8,
    8,
    8,
    8,
    8,
    8,
    8,
    8
  }
 };
}
```

查看游戏地图.



对游戏地图分析.大致可以知道关键部分

0 对应 自己的基地.

1 对应 墙体

8 对应 空白处

9 对应 毁坏后的基地

再看nDestroyNum在dll中的引用

```
// Bullect
// Token: 0x0600000A RID: 10 RVA: 0x00002898 File Offset: 0x00000A98
private void OnTriggerEnter2D(Collider2D collision)
{
 int num = (int)collision.gameObject.transform.position.x;
 int num2 = (int)collision.gameObject.transform.position.y;
 string tag = collision.tag;
 if (tag != null)
 {
  if (!(tag == "Tank"))
  {
   if (!(tag == "Heart"))
   {
    if (!(tag == "Enemy"))
    {
     if (!(tag == "Wall"))
     {
      if (tag == "Barrier")
      {
       if (this.isPlayerBullect)
       {
        collision.SendMessage("PlayAudio");
       }
       UnityEngine.Object.Destroy(base.gameObject);
      }
     }
     else
     {
      MapManager.MapState[num + 10, num2 + 8] = 8;
      MapManager.nDestroyNum++;
      UnityEngine.Object.Destroy(collision.gameObject);
      UnityEngine.Object.Destroy(base.gameObject);
     }
    }
    else if (this.isPlayerBullect)
    {
     collision.SendMessage("Die");
     UnityEngine.Object.Destroy(base.gameObject);
    }
   }
   else
   {
    MapManager.MapState[num + 10, num2 + 8] = 9;
    MapManager.nDestroyNum++;
    collision.SendMessage("Die");
    UnityEngine.Object.Destroy(base.gameObject);
   }
  }
  else if (!this.isPlayerBullect)
  {
   collision.SendMessage("Die");
   UnityEngine.Object.Destroy(base.gameObject);
  }
 }
}
```

分析代码,num和num2是方块在地图里的坐标(有负数),但经过"**num+10,num2+8**"调整后即为MapState数组中的下标.每当击中墙体,将墙体变成空白(**MapManager.MapState[num + 10, num2 + 8] = 8;**),再增加nDestroyNum的值,同理,击中基地时,将基地变成毁坏后的基地(**MapManager.MapState[num + 10, num2 + 8] = 9;**)再增加nDestroyNum的值.

那么大致思路就出来了: 累计击中4次或5次墙体或基地,进行Hash判断是否与给定Hash相等.游戏内一一尝试击中不太现实.我们可以直接在WinGame函数做点动作.更改后的WinGame函数如下

```
// MapManager
// Token: 0x06000029 RID: 41 RVA: 0x00003044 File Offset: 0x00001244
public static void WinGame()
{
 int num = 65;
 for (int i = 0; i < num - 2; i++)
 {
  for (int j = i + 1; j < num - 1; j++)
  {
   for (int k = j + 1; k < num; k++)
   {
    if (!MapManager.winGame)
    {
     MapManager.Init();
     MapManager.MapState[10, 0] = 9; //10,0是基地在MapState中的下标
     int num2 = -1;
     string text = "clearlove9";
     for (int l = 0; l < 21; l++)
     {
      for (int m = 0; m < 17; m++)
      {
       if (MapManager.MapState[l, m] == 1)
       {
        num2++;
        if (num2 == i || num2 == j || num2 == k)
        {
         MapManager.MapState[l, m] = 8;
        }
       }
       text += MapManager.MapState[l, m].ToString();
      }
     }
     if (MapManager.Sha1(text) == "3F649F708AAFA7A0A94138DC3022F6EA611E8D01")
     {
      FlagText._instance.gameObject.SetActive(true);
      FlagText.str = "RoarCTF{wm-" + MapManager.Md5(text) + "}";
      StreamWriter streamWriter = new StreamWriter("flag.txt");
      streamWriter.WriteLine(FlagText.str);
      streamWriter.Close();
      MapManager.winGame = true;
      return;
     }
    }
   }
  }
 }
}
```

大致就是设置标记位,将墙体标记起来并设置为8,每次循环都调用MapManager.init()以来恢复初始值.
理论上有4种情况

1. 击中3次墙体 + 1次基地(也就是上述代码) (nDestroyNum == 4)
2. 击中4次墙体 (nDestroyNum == 4)
3. 击中4次墙体 + 1次基地 (nDestroyNum == 5)
4. 击中5次墙体 (nDestroyNum == 5)

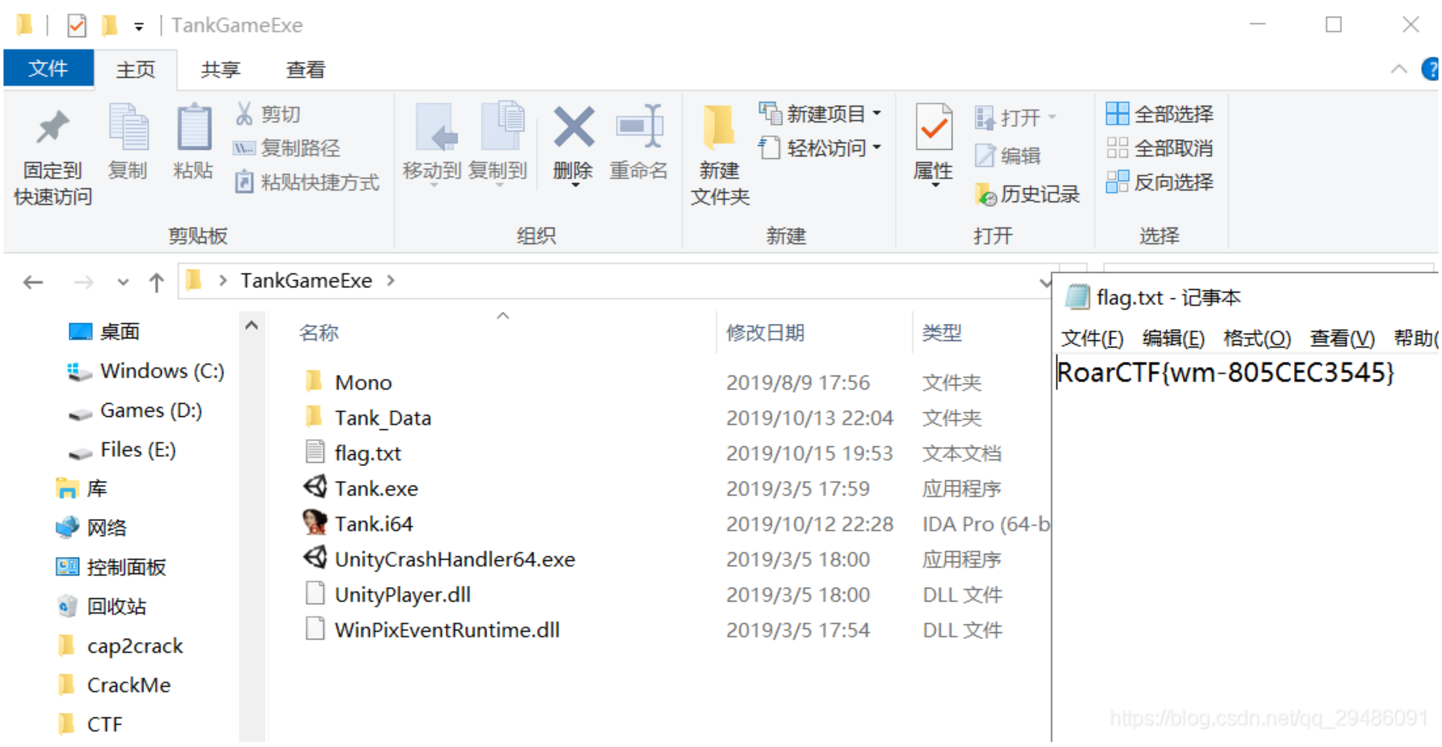
然而,就算得到Flag.str,游戏内好像没有相关函数将他显示出来(也有可能我没看见).所以调用写文件函数.

为了方便起见,在进入游戏时就调用WinGame进行判断,进入游戏后查看有无flag.txt文件就知道是否成功

```
// Option
// Token: 0x0600002F RID: 47 RVA: 0x00003264 File Offset: 0x00001464
private void Update()
{
 if (Input.GetKeyDown(KeyCode.W))
 {
  this.choice = 1;
  base.transform.position = this.posOne.position;
 }
 else if (Input.GetKeyDown(KeyCode.S))
 {
  this.choice = 2;
  base.transform.position = this.posTwo.position;
 }
 if (this.choice == 1 && Input.GetKeyDown(KeyCode.Space))
 {
  MapManager.Init();
  SceneManager.LoadScene(1);
  MapManager.WinGame();
 }
}
```

运行游戏.

查看游戏文件夹.



名称	修改日期	类型
Mono	2019/8/9 17:56	文件夹
Tank_Data	2019/10/13 22:04	文件夹
flag.txt	2019/10/15 19:53	文本文档
Tank.exe	2019/3/5 17:59	应用程序
Tank.i64	2019/10/12 22:28	IDA Pro (64-b
UnityCrashHandler64.exe	2019/3/5 18:00	应用程序
UnityPlayer.dll	2019/3/5 18:00	DLL 文件
WinPixEventRuntime.dll	2019/3/5 17:54	DLL 文件

```
RoarCTF{wm-805CEC3545}
```

得到flag.