****

**分析报告**

|  |  |
| --- | --- |
| **样本名** | **qqllk.exe** |
| **班级** | **软安41期** |
| **作者** | **张海龙** |
| **时间** | **2020年12月1日** |
| **平台** | **Win10** |

15PB信息安全研究院(协议分析报告)

# 目录

[目录 2](#_Toc18557)

[1．样本概况 3](#_Toc23915)

[1.1 应用程序信息 3](#_Toc979)

[1.2 分析环境及工具 3](#_Toc1141)

[1.3 分析目标 3](#_Toc31922)

[2．具体分析过程 3](#_Toc25729)

[2.1 找到真正的程序并去掉广告 3](#_Toc16048)

[2.2 寻找连连看地图首地址 6](#_Toc12738)

[2.3 寻找使用道具CALL 9](#_Toc28766)

[2.4 寻找消除CALL 11](#_Toc22277)

[2.5 寻找获取坐标CALL 12](#_Toc15823)

[参考文献 15](#_Toc29345)

[致 谢 16](#_Toc11336)

# 1．样本概况

## 1.1 应用程序信息

应用程序名称：qqllk.exe

大小: 1796608 bytes

修改时间: 2011年11月15日, 19:37:34

MD5: 814DE98DC72E4AB0001BA7F287239D2D

SHA1: 0DF1596821188E2333B8F5B8AE94DB775C796B3F

CRC32: 8437D6BE

qqllk.exe程序打开后并没有直接进入游戏，而是弹出了一个登录器程序，通过点击开始游戏后，再弹出一个广告，才会进入游戏。

## 1.2 分析环境及工具

系统环境:win10 64位

工具:OD CE DIE CodeinEX MFC注入工具

## 1.3 分析目标

1. 找到程序本身的exe

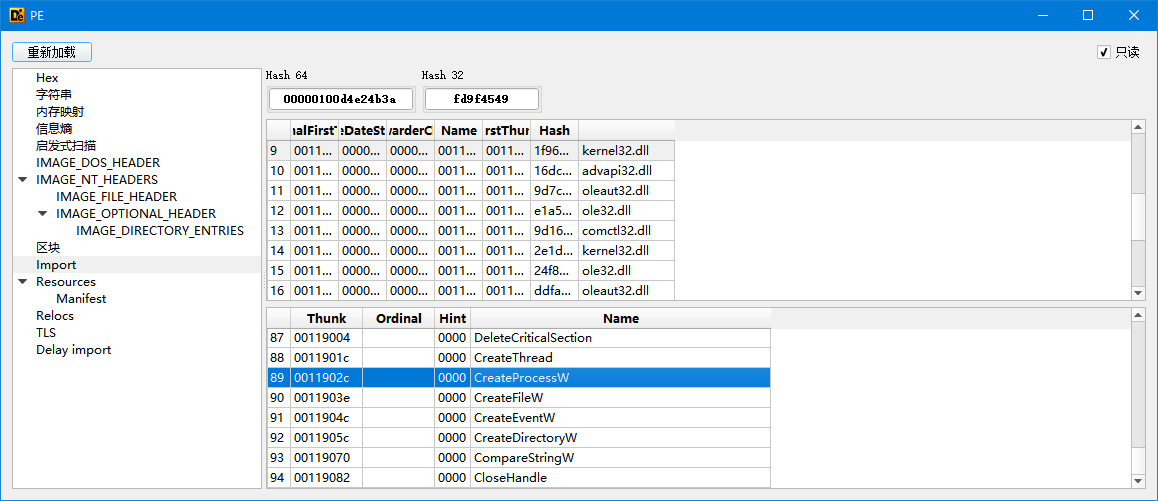
2. 去掉程序中的广告

3. 写一个QQ连连看的外挂

# 2．具体分析过程

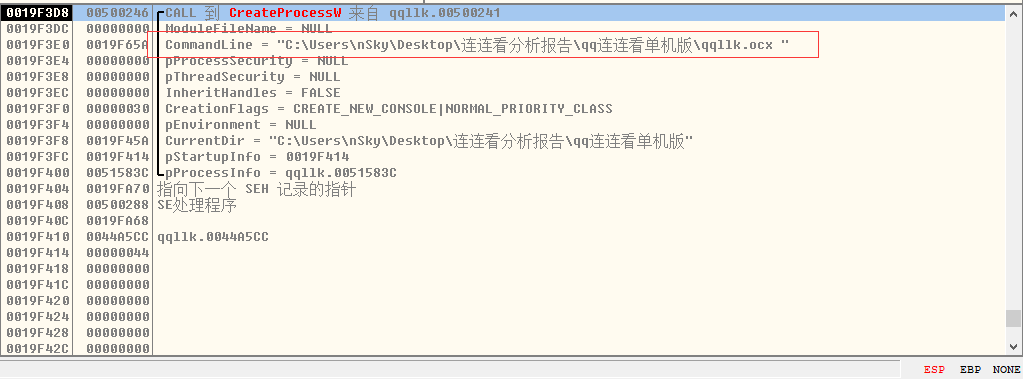
## 2.1 找到真正的程序并去掉广告

1、使用DIE查看一下qqllk.exe程序的导入表。



使用了CreateProcessW，根据前面是打开游戏过程，推测，可能是这个CreateProcessW函数创建了真正的游戏进程。

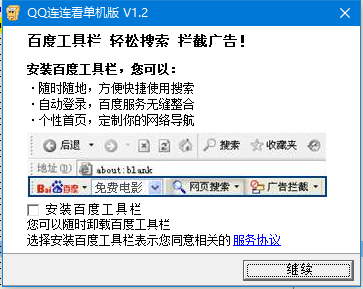
1. 使用OD打开程序qqllk.exe，运行程序后，下API断点CreateProcessW
2. 点击开始游戏，发现程序断下



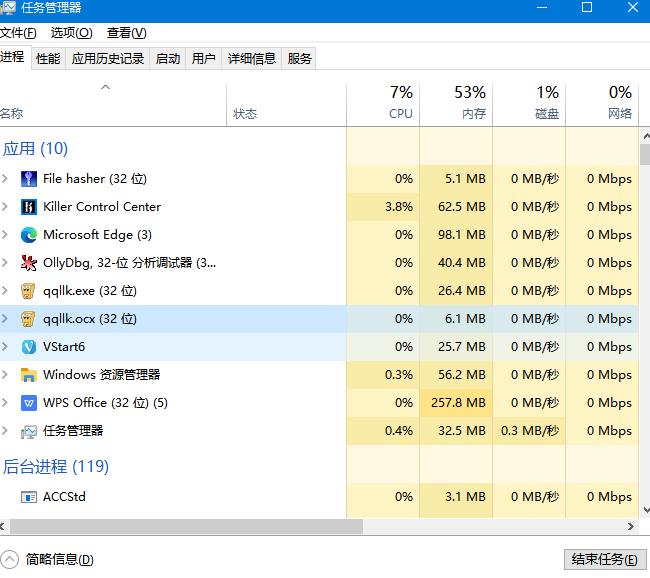
CreateProcessW的CommandLine参数是\*qqllk.ocx

虽然看着后缀不是.exe但是是一个可执行程序

弹出消息框



但是这个也不是我们的游戏程序，

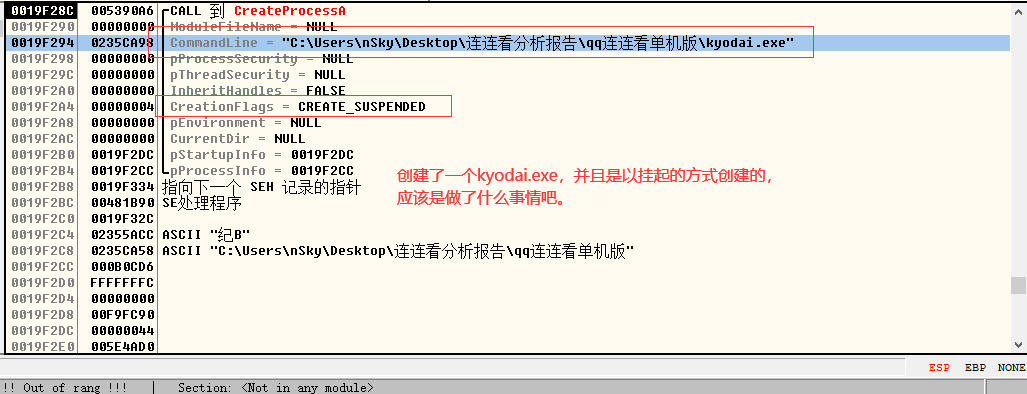
查看一下TaskMgr

发现进程中多了一个qqllk.ocx说明这个窗口是qqllk.ocx程序。

1. 使用DIE查看一下qqllk.ocx程序的导入函数，

并没有发现说明可以的创建进程的函数

1. 使用OD附加这个qqllk.ocx程序，直接下断点CreateProcessW、CreateProcessA



先不管他，我们直接跑起来，

连连看游戏直接出来了。

1. 双击kyodai.exe运行。

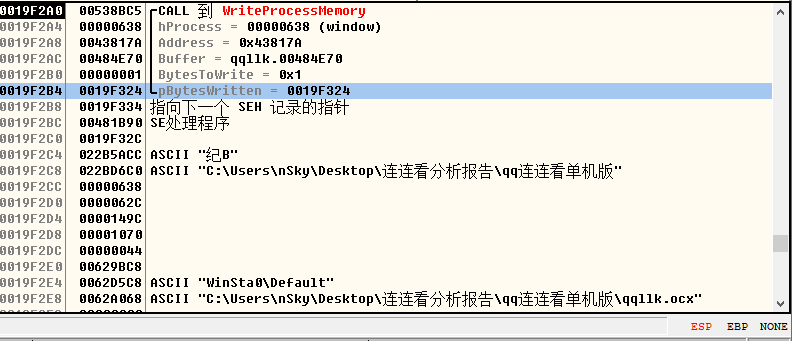
直接运行不起来。

推测应该是qqllk.ocx程序启动kyodai.exe的时候在挂起的时候对kyodai.exe做了一些事情。

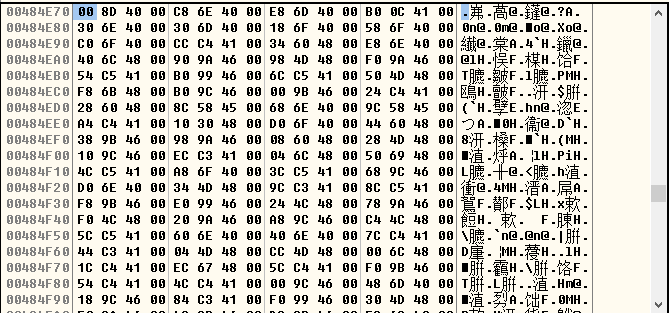
一个程序想要对另一个程序做事情，WriteProcessMemory

我们再次附加qqllk.ocx程序下一个WriteProcessMemry断点

既然是挂起的模式创建，那么肯定还会激活，我们再下一个ResumeThread断点。



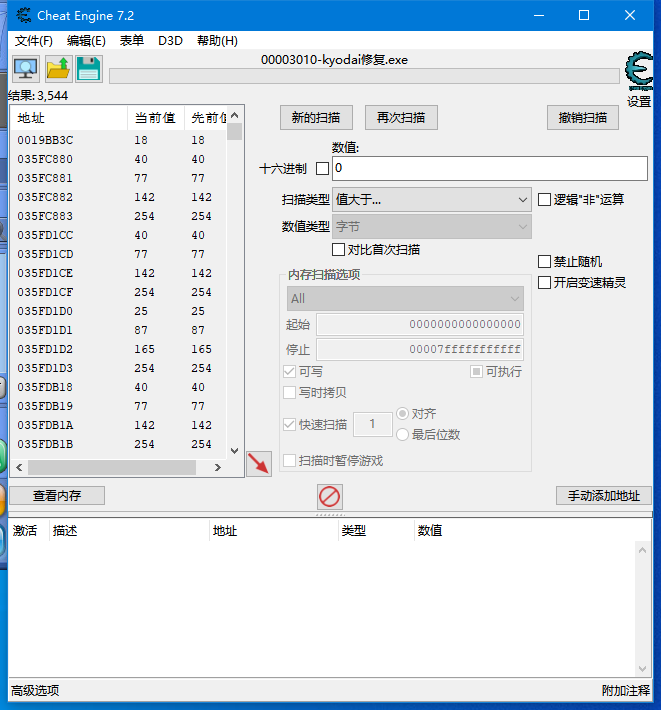
程序断下，向kyodai.exe程序写入了一个字节的数据，查看数据



是0，那么我们可以直接附加启动的程序或者计算0x43817A的FOA将0写入到文件中，我选择第二中方法，那么程序就成功去掉了广告

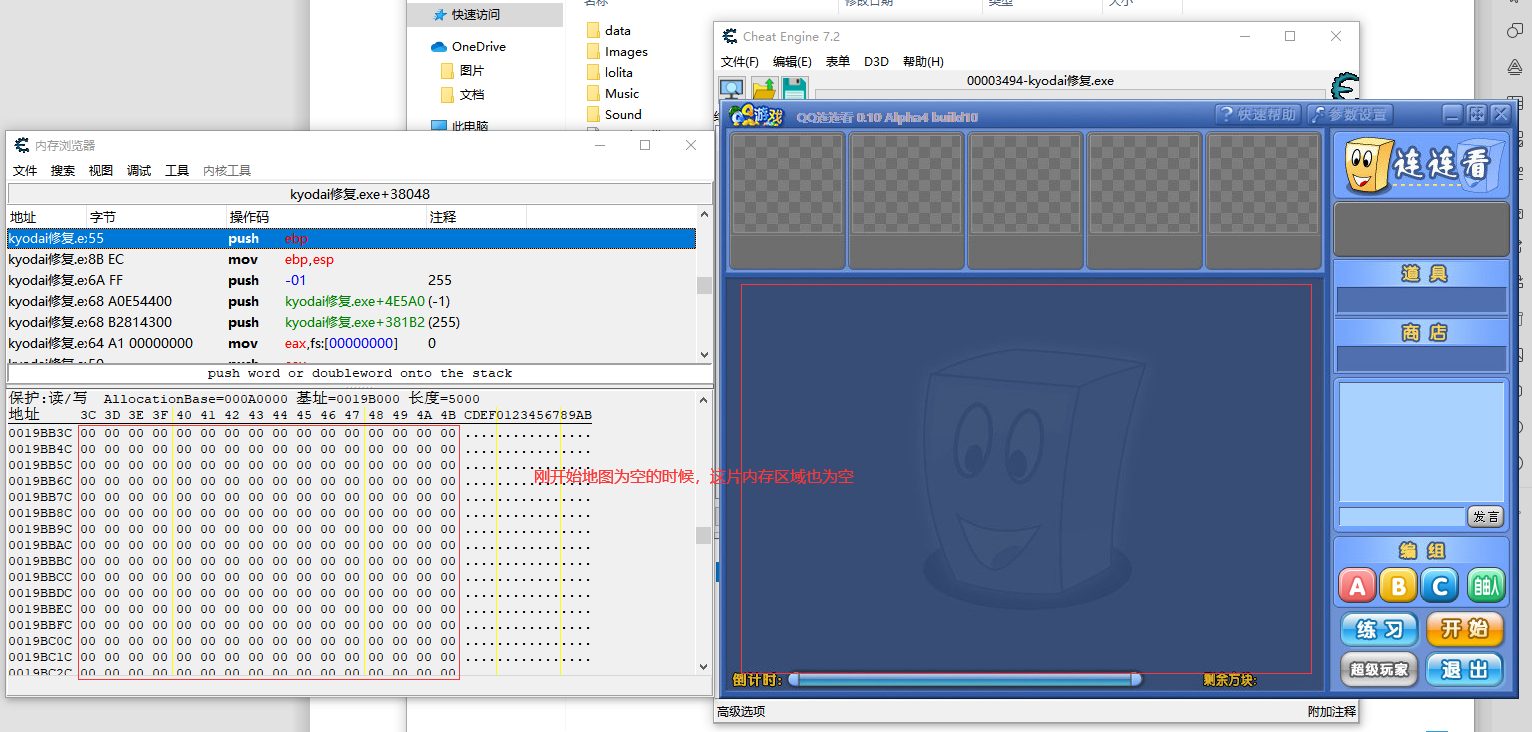
## 2.2 寻找连连看地图首地址

地图数组可能是BYTE类型的我们使用就搜索一个字节，地图为空就是数据0否则就是大于0搜索。

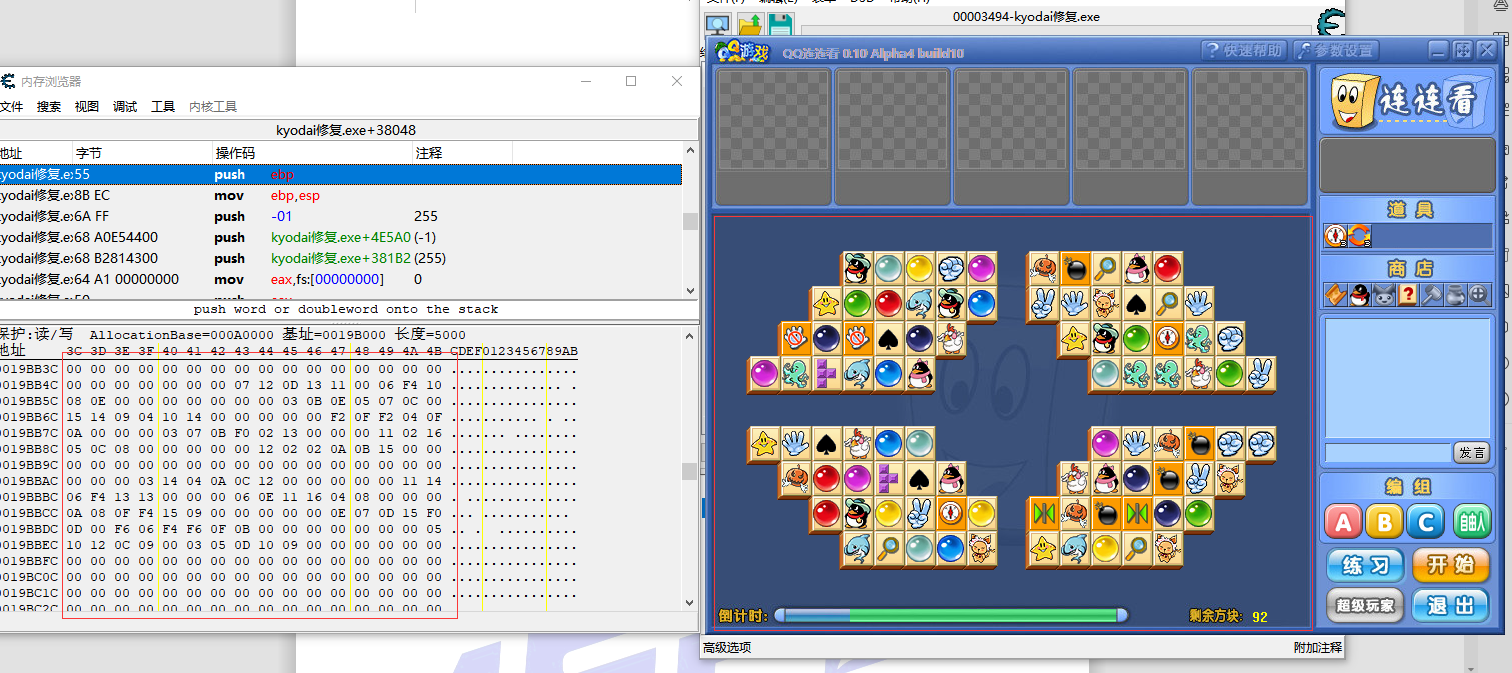


搜索到这里我们发现有很多，并无法确定是哪个，但是发现个规律，只有第一个是0019开头的，这个是win10系统的栈。

那我们就试试这个数据，CE这个数据右键浏览相关内存。



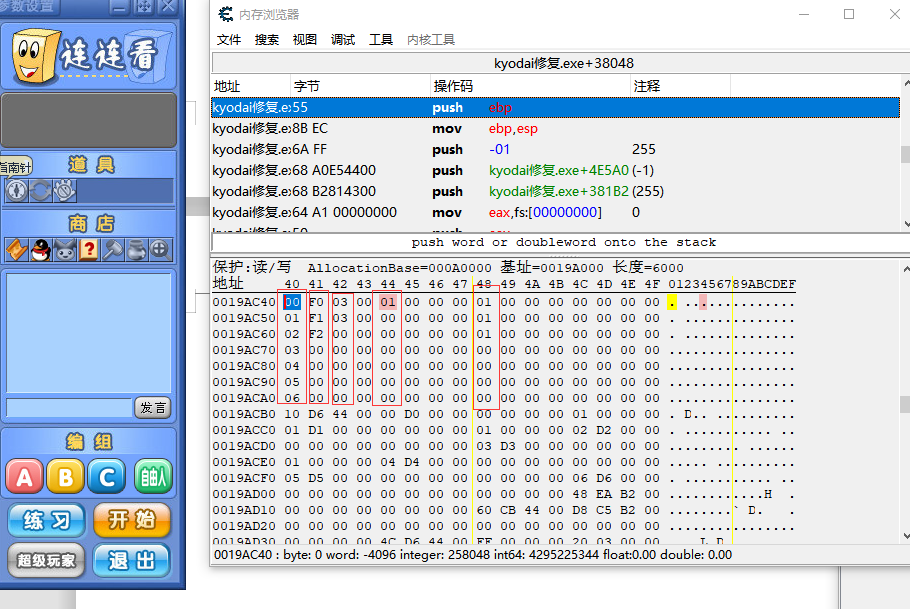
我们点击练习。开始新的游戏



这块内存中的数据和随着地图中的数据一起改变，经过多次测试后发现这个就是我们的地图基地址。

0019BB3C,

顺便使用CE找到了道具栏的基地址：0019AC40



根据观察和多组数据测试得知，

红框第一列是道具栏的下标

第二列是道具栏中物品的标识

第三列是道具的数量，

第四列是当前鼠标选中状态，

第五列标识背包中是否有物品

F0----指南针

F1----重列

F2----禁手

F3----闹钟

F4----炸弹

F5----蒙眼

F6----镜子

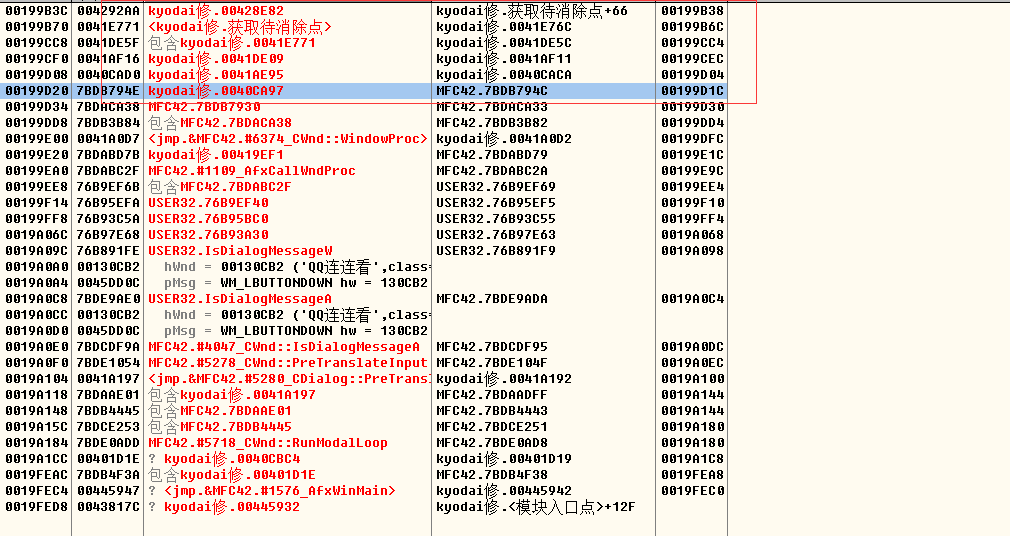
F7----障碍

F8----复活

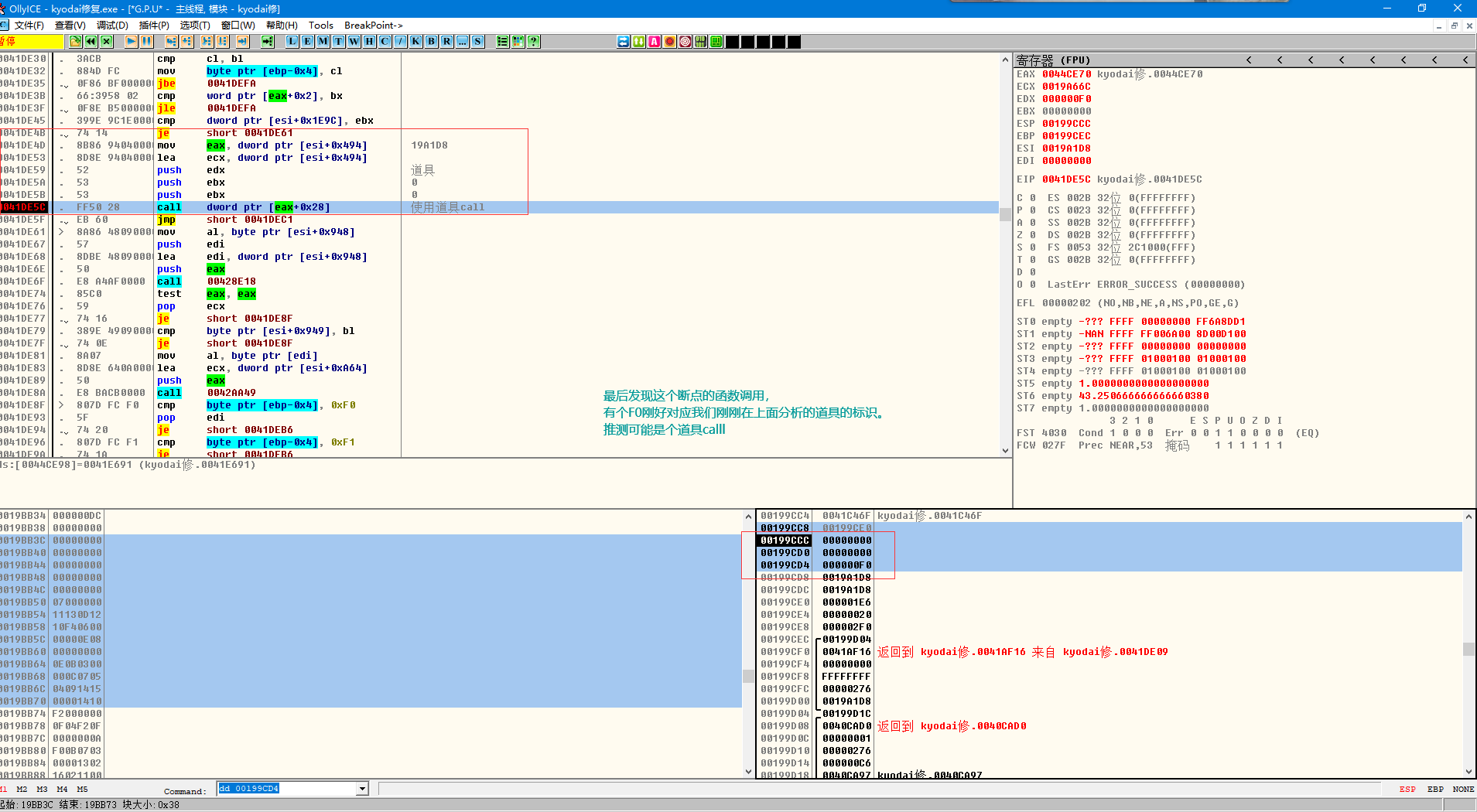
F9---一小时

## 2.3 寻找使用道具CALL

在运行期间，选中地图下一个内存访问断点，然后点击使用道具，看调用堆栈，



对红框中的函数调用挨个下断点进行排查



构建汇编代码注入程序

pushad

mov esi,0x19A1D8

mov eax, dword ptr [esi+0x494]

lea ecx, dword ptr [esi+0x494]

push 0xF4

push 0

push 0

call dword ptr [eax+0x28]

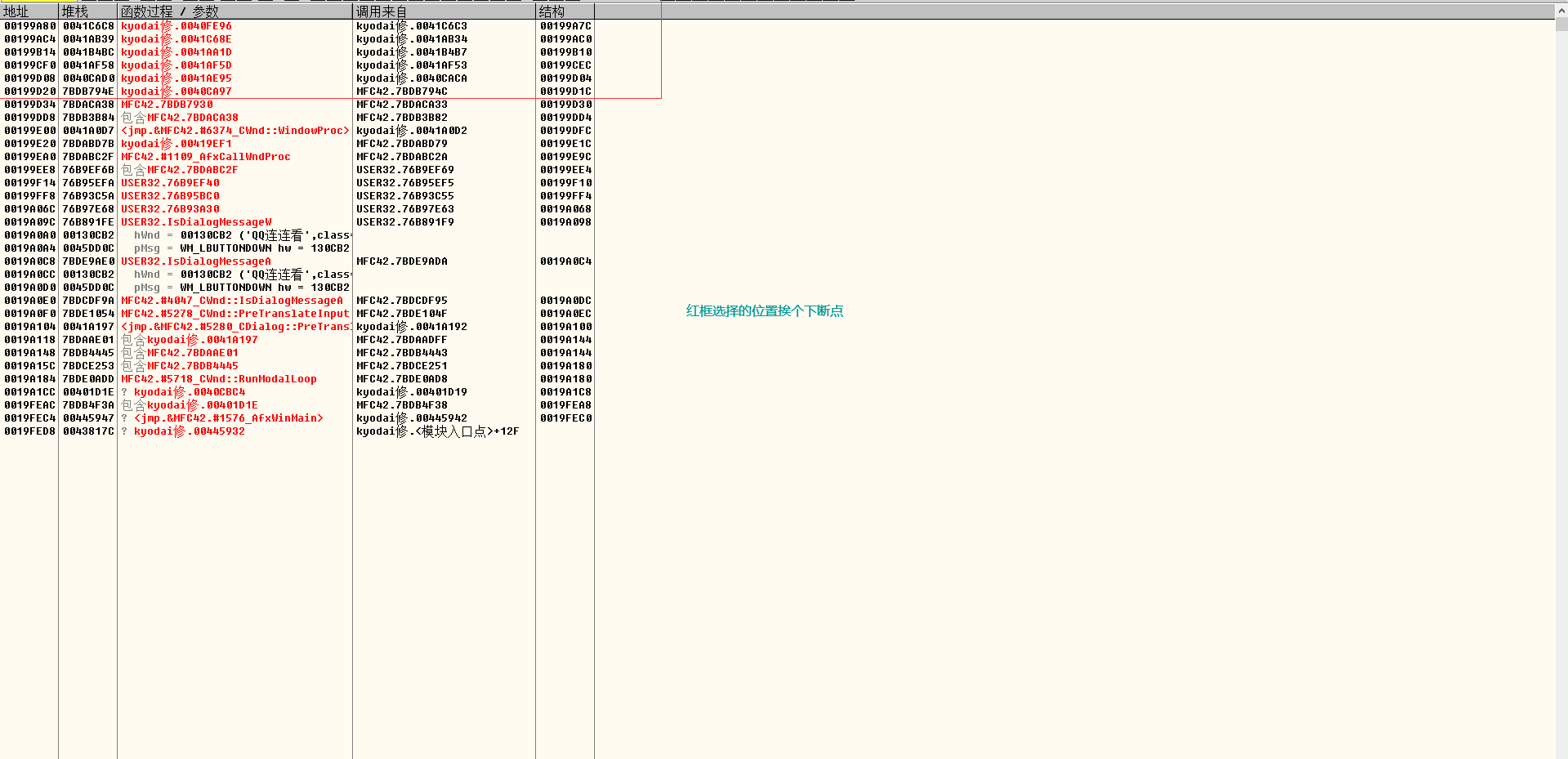
Popad

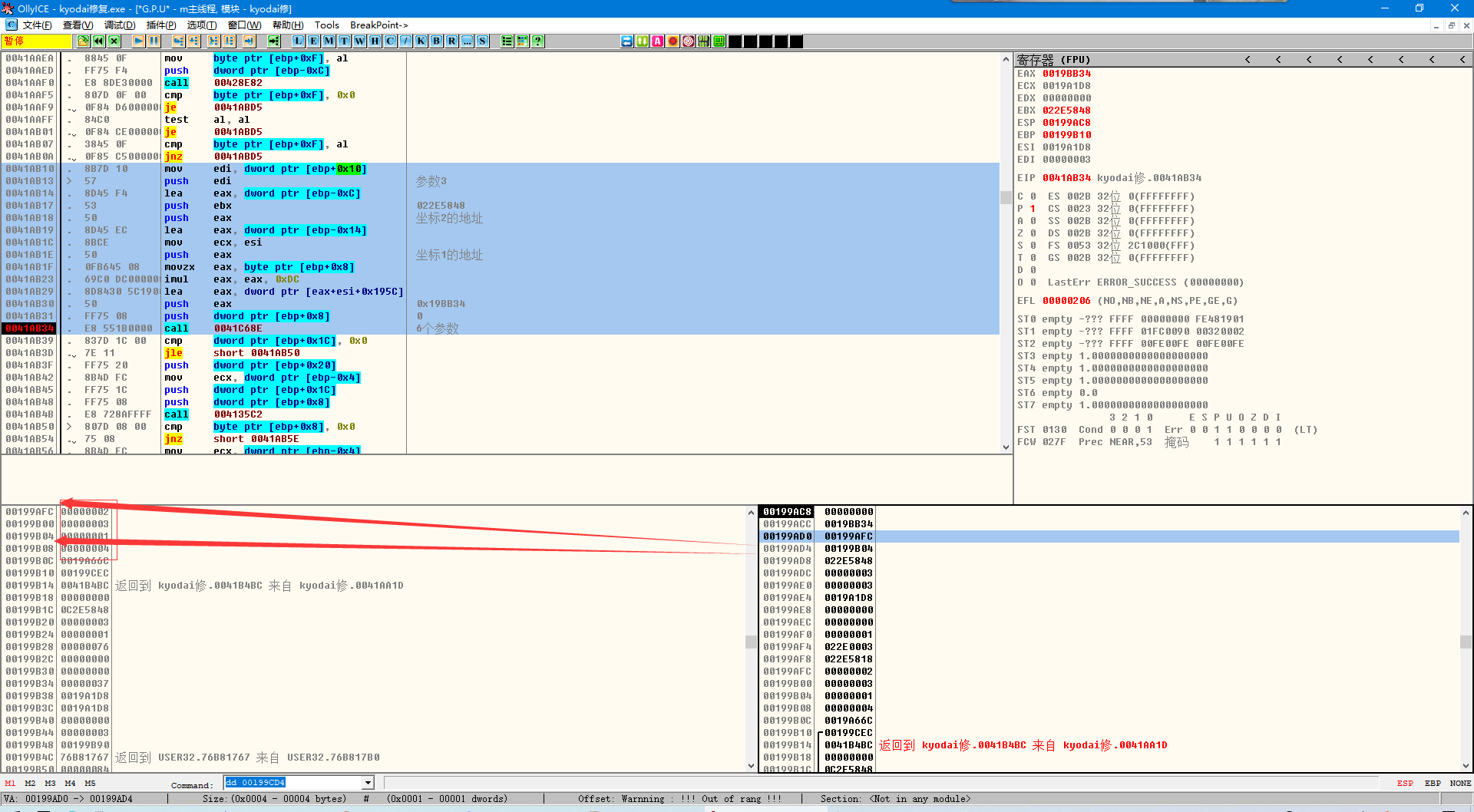


发现能使用道具，这个就是使用道具的call了

## 2.4 寻找消除CALL

在运行期间，选中地图下一个内存写入断点，然后点击可以消除的两个地方，进行消除

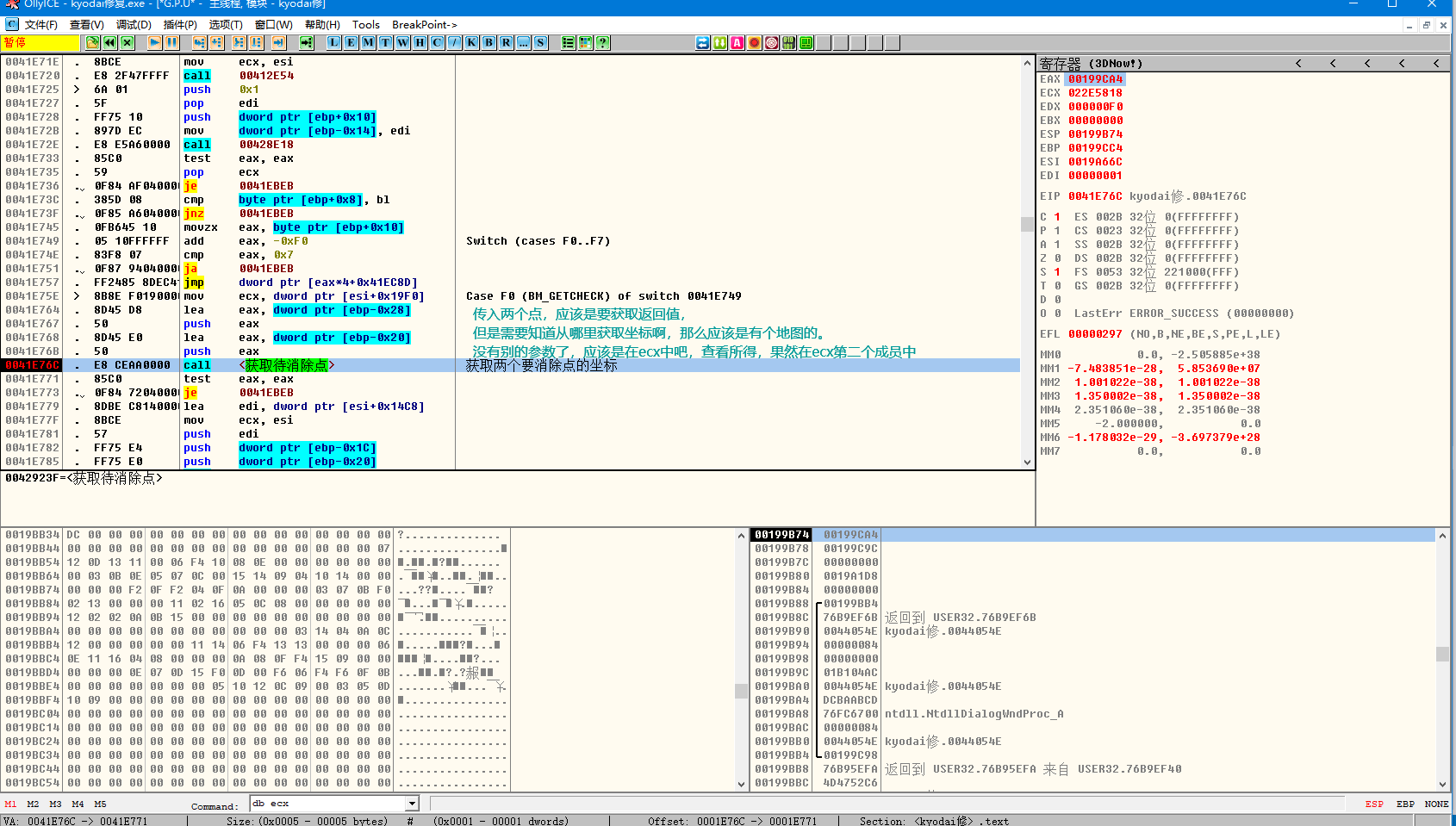




发现这call很适合，两个要消除的坐标。还有个数组的首地址，应该是用于消除后修改数组的数据的。

## 2.5 寻找获取坐标CALL

再次使用同样的方法，对地图数组进行下内存写入断点



构建call

POINT p1;

POINT p2;

pushad

mov esi, 0x0199CA4

mov ecx, dword ptr [esi+0x19F0]

lea eax, dword ptr[p1];

push eax

lea eax, dword ptr[p2];

push eax

mov eax,0042923F

call eax

popad

展示插件效果：

1. 最终效果

单次炸弹



模拟点击秒杀



炸弹秒杀



参考文献

[1] 15PB视频课程-逆向工程实战之连连看分析

致 谢

感谢15PB信息安全研究院。