**[置顶] Unity3d DLL脚本通用解密方法**

ss 1 发表于 2017-02-26 21:05

2 0

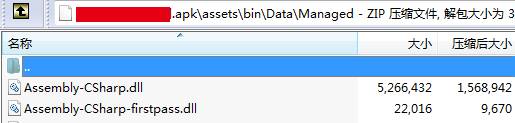
[Unity](https://msd.misuland.com/home/toSearch?label=Unity) [解密](https://msd.misuland.com/home/toSearch?label=解密) [Assembly-CSharp.dll](https://msd.misuland.com/home/toSearch?label=Assembly-CSharp.dll)

   网上已经有很多文章讲解过Unity3d脚本DLL 解密，基本方法原理都差不多，就是通过HOOK或者调试下断mono\_image\_open\_from\_data\_with\_name这个函数拿到解密后的DLL。这些方法都需要比较专业的知识(Hook或者过反调试)，另外对于使用了强度比较高的第三方保护，也无能为力。下面我们来探讨一下更为通用简便的U3D脚本DLL解密方法。

一、脚本解密原理

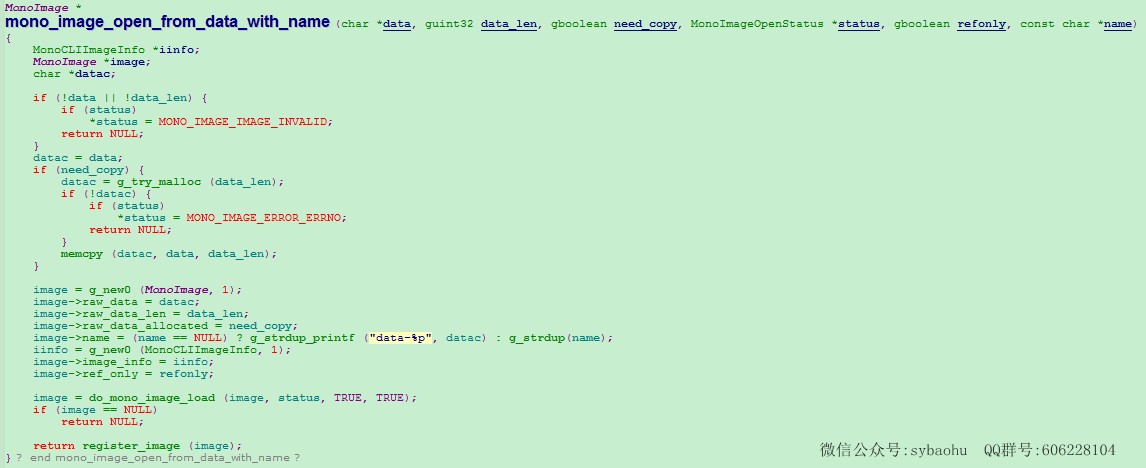
      先来理一下通过mono\_image\_open\_from\_data\_with\_name解密脚本DLL的原理。

      Unity3d游戏的脚本默认情况下是非加密形态，保存于如下图的压缩包目录下,一般情况名字为Assembly-CSharp.dll或Assembly-CSharp-firstpass.dll。



 这样的脚本DLL是由C#语言编写的，可以被类似Reflector之类的工具反编译成源码。破解者得到源码就可以对其进行修改，实现内购破解、修改游戏人物属性(修改金币、HP等)、修改战斗技能(无敌、强制胜利等)。

   我们来看下mono\_image\_open\_from\_data\_with\_name这个函数的源码



几个主要函数的参数意义:

data: 脚本内容

data\_len:脚本长度

name:脚本名称

这个函数执行了如下几步操作:

1.把data指向的脚本拷贝到新申请的内存

2.填充一个MonoImage结构体

3.使用do\_mono\_image\_load初步加载该脚本

4.注册并返回MonoImage结构体指针

    这个原始mono\_image\_open\_from\_data\_with\_name函数，如果输入的data指向的是个加密过的脚本DLL，在这个函数执行之前，先要对该data指向的内存进行解密。

调试解密可在return处下断，此时data指向的内存已经解密，把该处内存拷贝出来即是解密后的DLL

     HOOK解密原理是待mono\_image\_open\_from\_data\_with\_name原始函数返回后拷贝data指向的内存

    不过往往第三方保护都是带反调试的，新手要过掉反调试还是比较费周折的，有些厉害的反调试，老手都不一定过得了。对SO做HOOK也是要有一定的技术基础才能做到。

    对于强度稍微高点的保护，这个函数处是拿不到解密后的DLL的，等这个函数返回的时候，内存可能是空的。

    绝大部分加密，都紧盯着data指向的内存来做文章，认为只要把mono\_image\_open\_from\_data\_with\_name参数里的data加密搞定就万事大吉了，而忽略掉了另一块永久存在的处于解密状态的DLL内存。

    mono\_image\_open\_from\_data\_with\_name第一步就是把data内存使用memcpy拷贝到一块新申请的内存里,这块内存将永久存在，因为C#语言是需要动态解析类、函数等信息的，时不时要用到这块内存。而data待这个函数调用完后，将会被释放掉。

    这也是某些强度比较高的第三方保护，通过那两种方法都得不到解密DLL的关键所在，它在mono\_image\_open\_from\_data\_with\_name函数返回前就把data指向的内存给清空了。

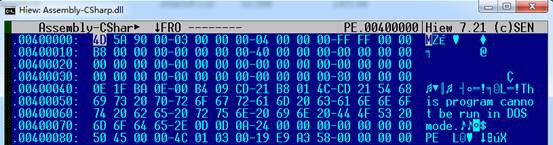
二、脚本解密方法

    下面就来讲一下我们的U3D脚本DLL解密方法，原理即是利用了内存中存在的那个memcpy拷贝的DLL。

    由于内存中永久存在这个DLL，我们只要全内存搜索这个DLL就可以了。如何全内存搜索呢，自己写个工具还是挺麻烦的一件事。

    其实用手游分析者比较熟悉的一个现成工具就可以做到:烧饼修改器(八门神器等其它修改器也可以)，烧饼修改器可以对游戏进行数值搜索。那我们要搜索什么数值呢？

   先用二进制工具hiew来看下Assembly-CSharp.dll



 这是个PE结构的文件，以4d 5a 90 00这四个字节为magic head,这可以做为C# DLL脚本的特征，我们只要搜索0x905a4d这个数值就可以了，由于烧饼修改器是使用10进制数值，将其转换成10进制值:9460301

    下图是使用烧饼修改器搜索一个3D卡车小游戏的示例:

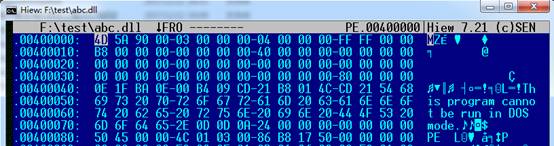


　 显示出来的是搜索结果，一共搜索到了12个结果

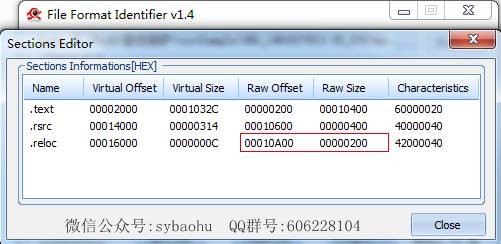
   　 第一个搜索结果，即是以52c4c008起始地址的脚本DLL，我们可以使用memdump工具把这块内存DUMP下来,由于不知道大小,先dump 0x3000大小来看一下。

http://img.blog.csdn.net/20170226205852122

保存为abc.dll，再用hiew看下abc.dll



  可以看到DUMP下来的确实是个CSharp脚本DLL,DLL实际大小可通过下图这个FFI工具，解析出来，将第3行的0x10A00加上0x200即是实际大小。可通过这个方法来确定这12个DLL，哪个是你需要解密的DLL。



   这个方法的优点是可以不用去管反调试，也不用去学习怎么HOOK，用现成的工具组合起来就可以做到解密脚本。最重要一点是对保护强度比较高的保护也可以无视。

       (有兴趣的同学可以关注我的微信公众号:sybaohu，以及QQ群:606228104)