**Unity 3d游戏逆向之.NET Reflector工具使用介绍**

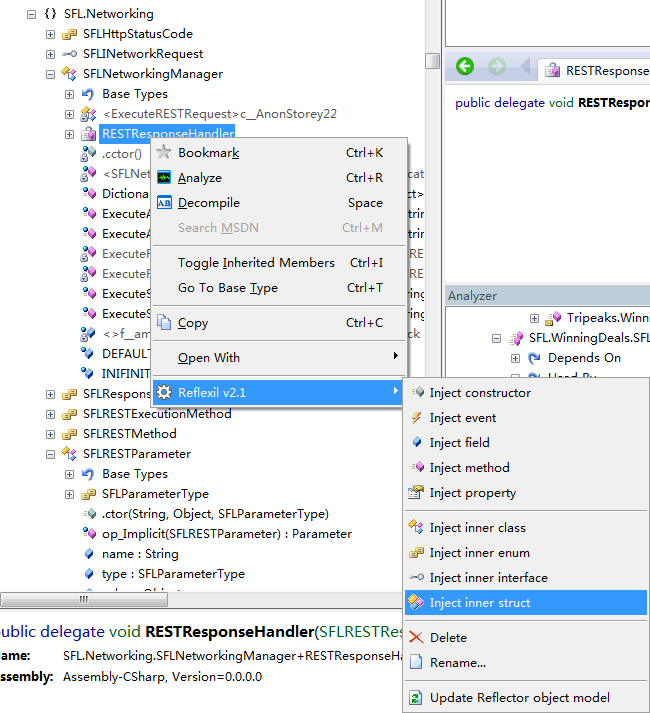
移动平台游戏框架主要有unity 3d和cocos 2d。我们首先得识别游戏使用的框架。

**识别Unity游戏**  
Android平台的apk包可以直接解压，看是否有./assets/bin/Data/Managed目录，也可以查看lib文件夹下面包含的一些so，如果有libmono,libunity等模块，基本可以确定是unity游戏了。  
Android平台中C#编写的主逻辑模块代码静态编辑之后存储于Assembly-CSharp.dll文件中。因为unity的跨平台，Android平台是unity编译的游戏，那么其对应的IOS平台上也是unity编译出来的。如果希望直接从IOS上面去看是否是unity游戏，可以提取游戏中的主模块查看是否有unity之类的函数即可。

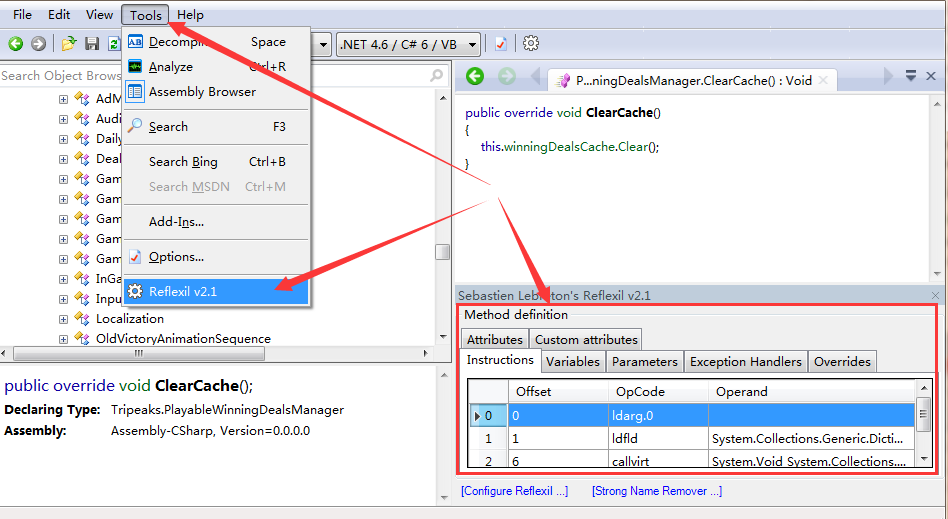
**破解思路**  
下面列举了一些破解版思路，如果能直接下断点在函数头修改寄存器可直接修改寄存器测试，遇到一些不能直接修改的，就用第二种方法，把修改后的Assembly-CSharp.dll注入到游戏中，让游戏执行我们修改后的代码。另外也可以动静态修改二进制实现。  
  
1、修改unity游戏逻辑代码编译成汇编代码相关的值  
（1） 修改传进来的参数，即寄存器，一般是set之类的函数  
（2） 汇编代码中尽量不修改内存，不修改opcode，能改寄存器直接改寄存器  
  
2、反编译Assembly-CSharp.dll，直接修改unity的C#源代码  
（1） 修改函数返回值  
（2） 直接删除函数体，只剩下 ret 指令  
（3） 在对应函数修改，对变量进行处理  
（4） 在对应函数增加一些call处理，主动call  
  
3、分析源码直接修改代码  
（1） 通过分析unity反编译后的源码找到对应的汇编指令下断点修改寄存器  
（2） 通过直接静态分析dll，直接修改IL码的二进制码  
  
4、在加载dll的函数位置dump原来的dll代码，可绕过dll加密，修改源代码  
hook住mono\_image\_open\_from\_data\_full函数，dump出dll可以，用IDA配合jdb挂起进程在那函数位置下断点dump也可以，源代码具体修改方案同“2”和“3”  
  
**常用工具**  
1、IDA工具  
可以进行动态调试和静态分析的工具，能在合适的位置下断点，修改指定寄存器和编写IDC脚本配合分析  
2、ILSpy  
反编译和分析dll代码，可以交叉引用，可以以源码形式保存反编译的代码，提供代码给DirFind等字符串搜索定位工具定位代码位置  
  
3、.NET Reflector + Reflexil  
反编译和分析dll代码，弥补了ILSpy一些功能性的缺陷，可以分析出错误的CLR文件头，一些在ILSpy显示不出的dll文件，如果只是因为dll头部被修改，放在.NET Reflector中是可以分析出的。Reflexil则是.NET Reflector的一款插件，可以反编译和回编译IL码，方便实用可视化。  
  
4、Ilasm和ildasm  
Ildasm可以反编译dll，dump出反编译后的il码，而Ilasm则可以重打包il码，利用命令ilasm /dll \*.il即可。

**常用IL码二进制**  
（1）nop 二进制是 0x00  
（2）ldc.i4.0 二进制是 0x16  
（3）ldc.i4.1 二进制是 0x17  
（4）ret 二进制是 0x2A  
（5）ldc.r4 二进制是 0x22 ，后面跟四个字节

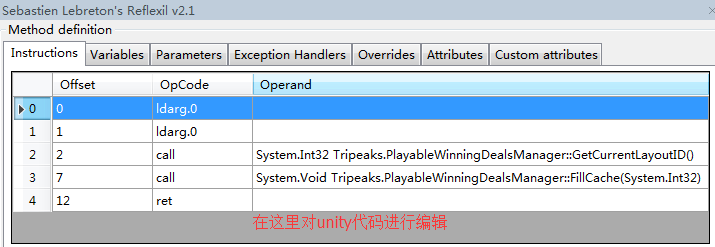
**.NET Reflector + Reflexil的使用**  
用.NET Reflector打开Assembly-CSharp.dll，这里使用的.NET Reflector版本是9.0，Reflexil版本是2.1。如图所示，通过这个Reflexil插件，我们可以插入字段、类、方法等。



通过Tools菜单下的Reflexil选项，可以打开unity代码修改器：



在左边单击对应函数，就可以对该函数代码进行修改了。

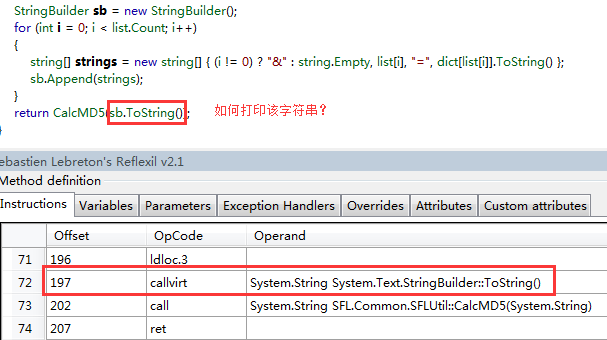


下面举例分析。在逆向中我们经常需要插入log语句打印log信息来辅助分析。unity打印log的语句是：

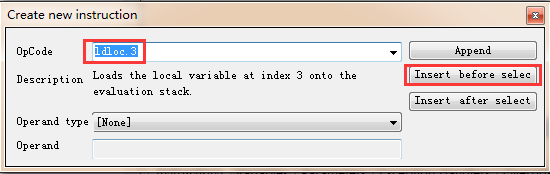
Debug.Log

函数原型是：public static void Log(object message);

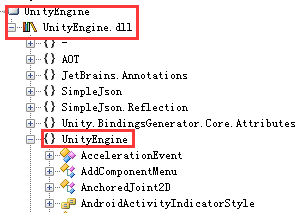
如果我们要打印下面函数中ToString的返回值应该如何操作呢？

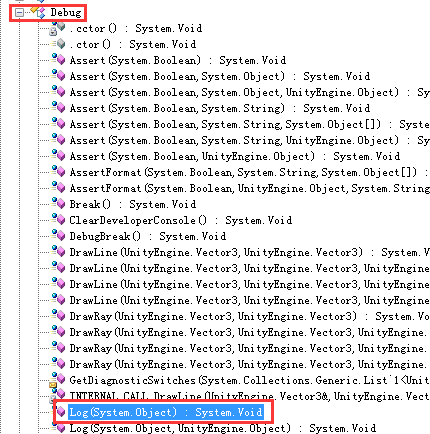


按照正常逻辑，应该是先保存ToString()的值到寄存器，然后当作参数传递给Log函数打印出来。但是我不知道怎么保存这个值，所以想到了另外一个办法，仿照原代码，再次调用ToString()函数，然后将下面调用CalcMD5函数改为调用Debug.Log函数。  
具体怎么操作呢？  
（1）首先右击“ldloc.3”那一行代码（Offset为 196），选择“Create new...”，OpCode选择“ldloc.3”，然后点击“Insert before select”。  
stloc.1 将值从堆栈弹出到局部变量 1  
ldloc.1 将索引 1 处的局部变量加载到计算堆栈上

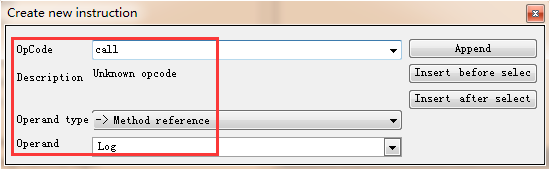


（2）OpCode选择“callvirt”,Operand type选择“Method reference”,Operand选择“ToString”函数（在mscorlib模块找到System.Text.StringBuilder::ToString函数）。然后点击“Insert before select”。  
（3）OpCode选择“call”,Operand type选择“Method reference”,Operand选择“Log”函数。这个函数是在UnityEngine中查找，如图所示：

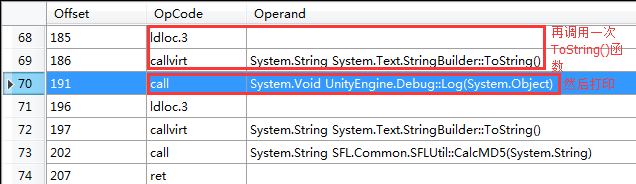




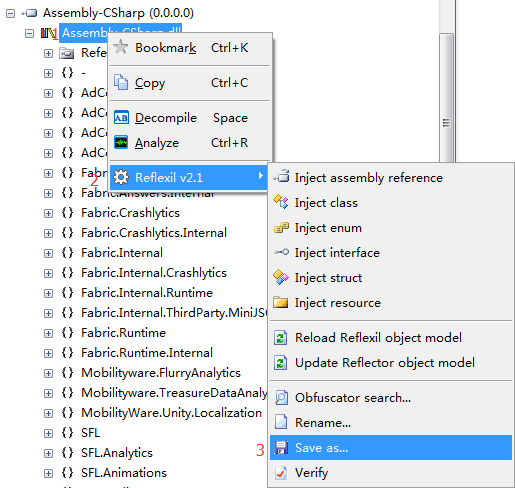
然后点击“Insert before select”，如下图所示：



添加代码后的情况如下：



修改完成后，右键“Assembly-CSharp.dll”，依次点击“Reflexil”，“Save as...”保存dll文件，重新打包安装运行就可以看到log输出信息了。



原文地址：<https://www.cnblogs.com/goodhacker/p/8232037.html>