链滴

关于游戏加密

作者：kyochow
原文链接：https://ld246.com/article/1602506295534
来源网站：链滴

说白了

本文主要讨论关于包体相关的加密，内存相关加密不在讨论，归根结底，游戏的加密，都只是仅仅提了被破解的门槛而已，阻挡了一些新朋友使用各种工具提取游戏资源，修改游戏配置乃至代码，并寄望于大神对我们的游戏没那么有兴趣，仅此而已

脚本加密

以Lua为例，很多游戏使用Lua作为逻辑开发语言，将lua打入AssetBundle或者zip包，进行热更，但种情况真的太容易被破解了，常见的解决思路分如下几步

1. 使用luac对脚本进行编译

生成字节码，天然丧失阅读性，还带着一点点加速属性，这是入门级"加密"，算是一举两得，但是破难度也是最低的，因为github上unluac之类的工具太多了，对于有一点编程基础的都可以反编译回来通过修改再封包，就可以达到破解目的

2. 在luac的基础上，对编译出来的字节码进行二次加密

一般兼顾解码速度的话，会使用异或算法，这时候已经可以拦截大多数专业程序员了

3. 在1+2的基础上，强化加密工具库

将异或算法的工具，使用c来写native库，Key写在native库里，增加破解难度，好处是可以进一步速加成，破解更难，但是通过汇编级别的静态代码调试还是可以的，但需要更牛的编程水平

4. 在1+2+3的基础上，异或的KEY，由外部传入

这时候，只能通过汇编级别的动态代码调试，而且两边既有底层语言，又有高级语言，会更加增加分难度，即便是技术大牛，绝大多数也会在这中间疲于奔命

5，再想更多的话，就是加密的增强了

上面只说了最基本的对称加密，好处就是最快速

如果使用非对称加密呢？RSA么?虽然强悍，但也容易造成效率大幅度下降，需要权衡

AssetBundle 加密

上文说到lua加密，我们的bundle也是需要加密的，这样可以提高被提取美术资源的难度，同时，因为ua会被打包进AssetBundle里，所以可以进一步增加破解门槛

1.使用offset + 伪造AssetBundle头部

根据AssetBundle.LoadFromFile的第三个参数，bundle可以在build的时候，根据一定的规则，指定个offset，读取的时候，再次从这个offset读取，这个空白的部分，可以填任意东西来混淆视听，达让网上那些读取工具崩溃或不工作的目的

2.对AssetBundle再次异或

或许有人会担心，对AssetBundle再次异或会不会导致trunk_base被破坏掉，其实并不会，Unity提了一个新的API AssetBundle.LoadFromStream可以很好的解决这个问题

关于异或(抄自百度百科)

异或，英文为exclusive OR，缩写成xor

异或（xor）是一个数学运算符。它应用于逻辑运算。异或的数学符号为“⊕”，计算机符号为“xor。其运算法则为：

a⊕b = (¬a ∧ b) ∨ (a ∧¬b)

如果a、b两个值不相同，则异或结果为1。如果a、b两个值相同，异或结果为0。

异或也叫半加运算，其运算法则相当于不带进位的二进制加法：二进制下用1表示真，0表示假，则异的运算法则为：0⊕0=0，1⊕0=1，0⊕1=1，1⊕1=0（同为0，异为1），这些法则与加法是相同的只是不带进位，所以异或常被认作不进位加法。

异或略称为XOR、EOR、EX-OR

程序中有三种演算子：XOR、xor、⊕。

使用方法如下

z = x ⊕ y

z = x xor y

未完待续