文章编号：1004-4353(2015)01-0064-04

面向 cocos2dx 的代码混淆器的设计与实现

王晓宇， 蔡京哲*
（延边大学工学院 计算机科学与技术学科 智能信息处理研究室，吉林 延吉 133002）

摘要：针对目前使用的代码混淆技术不能完全适用于手机的问题，在分析常用的代码混淆技术的基础上，提出用于手机引擎 cocos2dx 的基于代码匹配的混淆方法，即在外形(layout)混淆的基础上利用代码匹配技术搜索相同功能的代码并替换成更难让人读懂的形态. 试验分析表明，该方法的混淆效果好于传统外形混淆效果，执行程序过程中不仅没有降低运行效率，在某些情况下运行效率还有所提升.

关键词：代码混淆；手机软件；代码匹配
中图分类号：TP399 **文献标识码**：A

Design and implementation of code obfuscator for cocos2dx

WANG Xiaoyu， CAI Jingzhe*
(*Intelligent Information Processing Lab.*, *Department of Computer Science & Technology*, *College of Engineering*, *Yanbian University*, *Yanji* 133002, *China*)

Abstract: Aiming at the problem that currently used obfuscated code techniques are inapplicable to mobile phones, we propose a code matching based obfuscation method used for mobile phone engine cocos2dx, by analyzing commonly used obfuscated code techniques. That is based on the layout obfuscation, using the code matching technique to search code with the same function and replace it into a more unreadable form. Test analysis shows that the proposed method achieves better effect of obfuscation than traditional layout obfuscation. In the process of execution program, the operation efficiency is not only not reduced, but also slightly increased in some cases.

Key words: obfuscated code; mobile phone software; code matching

目前，随着多平台的游戏、软件日益增多，在外用手机在家用电脑成为大多数人的生活习惯，作为其基础的多平台开发引擎，如 cocos2dx、unity3d 以及多平台标记语言 javascript 等被广泛使用. javascript 等标记语言由于其本身的特性，打包成软件时并不编译，容易被破解和修改，因此往往需要通过混淆的方法来防止被破解. 目前，常用的混淆器如 JSMin、Javascript Cruncher-Compressor 等都是为网页而设计的[1]，并不适合于手机软件. 为此，本文根据手机平台的特点(如实时性要求高，内存和 CPU 能力有限等)提出了一种面向 cocos2dx 的代码混淆方法，该方法具有混淆效果好，且不影响程序效率的特点.

1 游戏作弊及代码混淆技术

1.1 游戏作弊

随着手机游戏的快速发展和壮大，针对手机游戏的代码跟踪和分析、游戏外挂以及游戏修改器等作弊手段也越来越多，使得手机软件的安全问题面临越来越多的威胁. 游戏作弊的定义为：玩

收稿日期：2014-12-03
* **通信作者**：蔡京哲(1969—)，男，副教授，研究方向为自然语言处理.

家违反游戏规则，通过各种方式使自己获得额外的优势及利益或者实现其他诚实玩家无法实现的目标[2]. 在游戏作弊手段中，外挂[3]经常被玩家所使用，外挂中技术含量最高的是全自动脱机外挂[4]，这种外挂根据掌握的游戏命令封包的格式，按照预先制定的规则和逻辑由程序自动与服务器交换信息，实现全自动脱机操作. 由于这种外挂需要了解软件的逻辑和程序，所以代码混淆是应对这种外挂的有效对抗手段.

1.2　代码混淆技术

代码混淆技术实质是尽可能去除非功能性信息，使生成的程序难以被人或者相关工具解析和理解，从而有效地保护软件. Boaz Barak 等[5]在理论上证明了代码混淆技术不可能对软件提供彻底的保护，但在实际应用中使用代码混淆技术可以加大攻击者的难度，提高修改代码的成本，以此达到保护代码的目的[6]. 目前，代码混淆技术分为外形(layout)混淆、控制结构(control)混淆、数据(data)混淆、预防(preventive)混淆[7]等几种.

为了提高混淆后的程序复杂度，数据混淆和控制混淆通常会在程序中引入大量的布尔变量或额外的结构，这种方法虽然增加了程序的抗攻击性，但大大增加了程序的代码量和程序的时间空间开销，降低了程序的执行效率[8]，因此不利于手机使用. 目前，常用的外形混淆主要有滥用标识符混淆技术[9]和重载无关联混淆技术. 滥用标识符混淆技术是通过对 Java 字节码中的标识符进行重命名来实现的，其目的是最大限度地重复使用同一个标识符代替一个类中出现的所有标识符. 该策略之所以能够迷惑攻击者，是因为攻击者不得不去理解反编译后程序中出现的每个标识符的行为. 重载无关联混淆技术主要依赖于扩展转换和方法重载，通过使用相同的标识符重命名被混淆类中所有实体的方法来实现. 这种方法在任何情况下都适用，而且不影响字节码的行为，这是因为 Java 程序中的方法在执行混淆操作之前的某个编译时刻已通过一个象征性的参考，必然会被执行. 上述这些外形混淆技术虽然没有引入额外的执行开销，但其只是对字节码中的标识符进行了简单的名字替换，因此其抗攻击能力不高.

2　基于代码匹配的混淆方法

本文针对 cocos2dx 引擎提出了基于代码匹配的混淆方法.

1) 外形混淆. 由于 cocos2dx 中变量名、函数名并不完全只存在于代码中，在部分资源文件中也会存在类、方法和属性的名称，因此普通的混淆器无法在此平台上通用. Javascript 作为一种解释执行的脚本语言，其自身的一些特性，会对代码混淆带来很大的困难. 首先，Javascript 是一种基于原型的语言，没有严格的类型定义. 在自定义的类中，对于需要外部访问的属性和方法，不能进行混淆；对于内部访问的属性和方法，需要进行混淆；但 Javascript 语言本身无法对属性和方法进行这样的区分，为此需要寻找一种机制来确定哪些类的属性和方法名称需要进行混淆. 其次，Javascript 语言本身定义了大量的核心的类、方法和属性，引擎中也定义了大量的系统的类、方法和属性，而且这些类、方法和属性都不能够被混淆. 最后，全局变量和局部变量的表现形式是一样的，其中所有的局部变量都是可以和应该被混淆的，而全局变量有的需要混淆有的不需要混淆. 因此难以区分局部变量和全局变量，而且全局变量本身更无法明确是否需要被混淆.

为了解决这些问题，本文设计了一个专门用于 cocos2dx 引擎的 Javascript 混淆器，其原理为：首先，由于 cocos2dx 是以 Javascript 作为上层通过一种映射关系来调用底层的 C++代码，因此引擎中提供了 Javascript 到 C++的函数的映射表，所有不能够被混淆的类、方法和属性都能在这张表中找到. 其次，cocos2dx 中的界面、场景是在特定工具中编辑成资源文件后由代码驱动的，因此大部分的方法、属性名称和全部的类名都会在资源文件中出现.

本文设计的混淆器所采用的外形混淆方法的具体步骤如下：①遍历所有的资源文件，统计其中的类、方法和属性名称并对其分类、记录；②在代码中寻找与之对应的类、方法和属性统一进行混淆；③对代码中所有可以混淆的方法进行混淆；④对所有非 var 定义的变量(cocos2dx 中全局变量和类中公用变量)进行全工程范围的统计并

对其混淆;⑤最后以单个文件为范围对 var 定义的变量(cocos2dx 中的局部变量)进行混淆.

2) 代码匹配. 代码匹配就是判断两段代码的相似度[10],与传统的代码相似度检测不同,本文需要的是寻找功能完全相同的代码. 在混淆过程中如果出现本可以混淆的代码未被混淆,并不会对程序的运行产生任何影响,只会略微降低混淆效果;如果出现不能被混淆的代码被混淆,则可能改变原程序的功能,会对程序的运行产生毁灭性的影响. 故本文采用基于数据流的代码匹配方法,即通过分析数据的变化过程来判断两段代码是否相同,其优点是准确率极高,召回率较好. 基于数据流的代码匹配方法的具体步骤如下:

Step1 遍历并记录代码中所有变量;

Step2 分别遍历所有变量在这段代码中所执行的运算和它与其他变量的关系并作记录;

Step3 用不同的类代替运算、判断和循环,根据上述记录利用指针建立变量的运算关系图(图1);

Step4 改变上述结果中变量的顺序并与目标对比,如果相同则将目标代码按照特定变量顺序替换原代码,反之结束.

在具体实现时,由于 Javascript 是弱类型的,故只需为变量建立一个类. 在 Step1 中将所有变量实例化后存入一个向量,再建立一个存放运算的基类,并由这个类派生出各种不同的运算类,并用指针将这些类和变量链接起来,如运算"$a=b*5$"就由变量 a 指向乘法运算类,再在乘法运算类中标明数值 5 并定义指针指向变量 b.

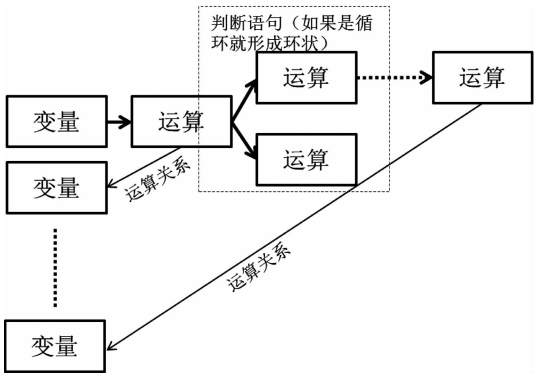

图1 变量的运算关系图

Step4 中的对比步骤为:①如果变量数量不同,则直接结束并返回不匹配,反之则继续下一步;②依次取待混淆代码中变量的运算关系,执行下一步;③将②中的运算关系与目标中所有变量的运算关系依次对比,如果有相同的则建立对应关系,反之则结束并返回不匹配;④ 根据上述对应关系进行代码替换.

3 实验结果及分析

本文利用现有的两个游戏程序和 cocos2dx 中的一个 demo 对本文提出的混淆方法进行测试. 为了更加突出代码混淆对程序性能的影响,选用了性能最差的 iphone 4 手机作为测试平台,并记录了各个界面的平均帧率(一般情况下,游戏最高帧率是 60 帧/s),结果如表 1 所示.

表1 各界面帧率 帧/s

混淆类型	游戏1(较复杂)		游戏2(较简单)		demo
	菜单(含动画)	游戏	菜单(无动画)	游戏	
未混编	31.2	26.5	60	45.6	59.8
外形混淆	31.2	26.7	60	45.8	59.8
基于代码匹配的混淆	31.2	26.7	60	45.8	59.8

由表 1 可以看出:在游戏 2 中无动画的菜单中没有代码在运行,其帧率一直保持最高 60 帧;demo 的内容较少,只有在少数物体被创建时会掉帧;游戏 1 中的帧率较低,这是因为游戏 1 较为复杂,对硬件的要求较高. 由于混淆去除了代码中非功能性信息,使代码变短,因此在反复需要创建和销毁对象的游戏中略微提高了帧率. 由此表明,基于代码匹配的混淆并不会影响原代码的执行效率. 为了评价代码混淆的有效性,本文采用一种基于图灵机代码混淆评价方法[11]对上述 3 个工程混淆结果进行评价,具体方法是:统计代码中混淆后变化的语句,根据难易理解程度给以不同的分

值并进行累加，最后除以代码量得出评价结果（最高值是 100，即所有语句都变为最难理解的形态）. 另外，利用第 3 方混淆器“packer”对上述 3 个程序进行混淆（混淆后程序不能运行，仅用于代码混淆的有效性比较）作为对比，其结果如表 2 所示.

表 2　混淆结果评价

混淆类型	评价值		
	游戏 1（较复杂）	游戏 2（较简单）	demo
外形混淆	27	21	16
基于代码匹配的混淆	40	35	18
控制结构混淆	44	39	21
预防混淆	43	38	19

由表 2 可以看出：在游戏 1 和游戏 2 中，基于代码匹配的混淆效果与控制结构混淆和预防混淆的效果接近，且均好于外形混淆. 另外，需要说明的是，由于 demo 的代码本身就非常精炼，所以上述几种混淆方法都没有太好的混淆效果.

4　结论

本文提出了基于代码匹配的混淆方法，试验结果表明，这种算法不会对代码的执行效率产生太大的影响，同时其混淆效果又接近于控制结构混淆和预防混淆的混淆效果. 由于本文提出的方法是以 cocos2dx 引擎为基础设计的，所以只适用于 cocos2dx 引擎，今后还需进一步提高程序的通用性；提高本文方法的混淆效果需要庞大的代码库，但本文只建立了一个简单的测试代码库，因此在以后的工作中还需要不断补充和完善代码库.

参考文献：

[1] 霍建雷，范训礼，房鼎益. Java 标识符重命名混淆算法及其实现[J]. 计算机工程，2010(1)：146-147.

[2] Lan Xiao, Zhang Yichun, Xu Pin. An overview on game cheeting and its counter-measures[C]// International Symposium on Computer Science and Computational Technology. Hefei: IEEE, 2009：195-200.

[3] 兰晓. 中国网络游戏外挂问题现状分析[D]. 中国传媒大学信息工程学院，2010：71-77.

[4] Greg Hogland, Gary McGraw. Exploiting Online Games[M]. Cheating Massively Distributed Systems. Addison-Wesley Professional, 2007：14-25；30-40.

[5] Barak B, Goldreich O, Impagliazzo R, et al. On the (im)possibility of obfuscating programs[C]// Lecture Notes in Computer Science. Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology. London: Springer-Verlag, 2001：1-18.

[6] 刁俊峰. 软件安全中的若干关键技术研究[D]. 北京：北京邮电大学，2007.

[7] 史扬，曹立明，王小平. 混淆算法研究综述[J]. 同济大学学报：自然科学版，2005，33(6)：80-81.

[8] 霍建雷. 用于 Java 软件保护的代码混淆技术研究与实现[D]. 陕西：西北大学，2009.

[9] Cimato S, Santis A D, Petrillo U F. Overcoming the obfuscation of Java programs by identifier renaming[J]. Journal of Systems and Software, 2005, 78(1)：60-72.

[10] 胡正军. 程序代码相似度检测方法研究及应用[D]. 湖南：中南大学，2012.

[11] 王冬. 一种基于图灵机的代码混淆评价方法[D]. 天津：南开大学，2012.