doi 10.3969/j.issn.1671-1122.2016.05.011

基于 Android 可执行文件重组的混淆方案的设计与实现

文伟平，张汉，曹向磊

（北京大学软件与微电子学院，北京 102600）

摘　要：随着移动智能终端的高速发展，Android 操作系统已经成为世界上使用最广泛的移动智能操作系统之一。Android 操作系统的设计者选择了开发者众多、跨平台性和效率都比较好的 Java 语言进行开发。Java 语言的特性使得 Java 程序很容易被反编译工具反编译并逆向分析，这就使得 Android 应用程序面临很大的风险。文章以保护 Android 应用程序、提高攻击者逆向分析的难度、不增加程序执行时的额外开销为目的，深入研究代码混淆中的外形混淆技术，在 Android 可执行文件重组的基础上，设计并实现了一种 Android 混淆工具，并对该工具进行了测试与性能分析。Android 混淆工具提高了 Android 软件的安全性，保护了 Android 应用程序开发者的知识产权，在一定程度上避免了 Android 应用程序被逆向分析、盗版以及恶意篡改。

关键词：Android；可执行文件；重组；代码混淆

中图分类号：TP309　　文献标识码：A　　文章编号：1671-1122（2016）05-0071-07

中文引用格式：文伟平，张汉，曹向磊 . 基于 Android 可执行文件重组的混淆方案的设计与实现 [J]. 信息网络安全，2016（5）：71-77.

英文引用格式：WEN Weiping，ZHANG Han，CAO Xianglei. Design and Implementation of the Scheme of Obfuscation Based on the Recombination of the Android Executable File[J].Netinfo Security, 2016（5）：71-77.

Design and Implementation of the Scheme of Obfuscation Based on the Recombination of the Android Executable File

WEN Weiping, ZHANG Han, CAO Xianglei

(*School of Software&Microelectronics*, *Peking University*, *Beijing* 102600, *China*)

Abstract: With the rapid development of mobile intelligent terminals, Android operating system has become one of the most widely used mobile intelligent operating systems in the world. Java is famous for its features of good cross-platform, high efficiency and a large amount of developers, therefore the designers of Android choose Java as the system development language. The characteristics of the Java language make Java program easy be decompiled by decompilation tools and be analyzed, which makes Android applications face great risks. This paper focuses on the study of code obfuscation technology for the purpose of protecting Android applications, improving the difficulty of the attacker's reverse analysis and adding no extra time cost for the execution of the program. Based on Android executable file reorganization, this paper designs and implements a kind of Android obfuscation tool and carries out test and performances analysis. This Android obfuscation tool enhances the security of Android applications, protects Android applications developers' intellectual property rights, and avoids reverse analysis, piracy and malicious tampering to Android applications to a certain extent.

Key words: Android; executable file; recombination; code obfuscation

收稿日期：2016-04-18

基金项目：国家自然科学基金 [61170282]

作者简介：文伟平（1976—），男，湖南，副教授，博士，主要研究方向为网络攻击与防范、恶意代码研究、信息系统逆向工程等；张汉（1990—），男，河南，硕士研究生，主要研究方向为网络与系统安全；曹向磊（1990—），男，河南，硕士研究生，主要研究方向为 Android 安全。

通信作者：文伟平 weipingwen@ss.pku.edu.cn

0 引言

Android 操作系统现已成为使用最广泛的智能操作系统之一[1]。Android 操作系统使用非常方便，且是开源的[2]，这些特性使得 Android 操作系统发展非常迅速。伴随着 Android 操作系统的发展，Android 应用程序开发过程中出现的问题也越来越多。

Android 操作系统的设计者选用了 Java 语言作为开发工具，其字节码文件不包含与运行该字节码的机器相关的信息，但包含大量与 Java 源代码相关的信息。源代码中变量和方法的引用信息以及类的引用信息等都保留在字节码文件中，而这些引用信息一般都带有许多语义信息，这对于 Java 源程序的反编译非常容易。Android 可执行文件 (DEX 文件) 是由字节码文件转化而来的，因此也带有许多语义信息。通过反编译工具 dex2jar 以及 jar 文件查看工具 jd-gui 非常容易查看 APK 文件的源代码[3,4]，这使得 Android 开发人员和公司投入大量人力和物力努力编写的源代码非常容易被攻击者窃取，Android 应用程序中包含的重要算法也会被泄露。因此，对于 Android 应用程序的代码保护技术的研究显得非常重要。

代码混淆技术的相关研究工作一直都是国内外安全研究者的研究重点。COLLBERG[5] 等人对代码混淆技术的研究做出了很多贡献。COLLBERG 等人将混淆转换分为 4 类 :1) 外形混淆 ;2) 数据混淆 ;3) 控制混淆 ;4) 预防混淆。另外，COLLBERG 等人还首次提出了评估混淆的 3 个标准 : 开销 (Cost)、强度 (Potency) 和弹性 (Resilience)[6]。随后，COLLBERG 等人又提出了另外一个评估标准——隐蔽性 (Stealth)，即混淆前后程序的相识性。WANG[7] 深入研究了静态分析工具获取程序语义的原理，在此基础上使用控制流压平技术和保护控制变量的技术来增加静态分析工具分析程序的难度。WANG 提出了代码混淆算法，这些算法实现了对程序的混淆，能够有效防止静态分析。此外，WANG 还从理论上证明了其所提出的代码混淆算法对静态分析来说是 NP 难问题。

国内对于代码混淆的研究也有一定的成果。史扬等人在文献 [8] 中对代码混淆的定义、类别、评估方法等都做了概要性介绍。赵玉洁[9,10] 等人在 COLLBERG 等人评估代码混淆方法的基础上，提出了代码混淆有效性评估的框架，该框架面向逆向分析，能够验证混淆算法的有效性。此外，赵玉洁等人还对逆向攻击的各种相关评估指标进行了计算，并通过相关具体实验对方法的可行性进行了验证，这为软件保护技术的评估研究提供了一种新思路，对代码混淆技术的研究起到了一定的促进作用。

1 相关技术介绍

1.1 Android系统体系结构

Android 操作系统采用层次化的系统架构，分为 4 层 :1) 应用层 ;2) 应用程序框架层 ;3) 系统运行库层 ;4) Linux 内核层。如图 1 所示。

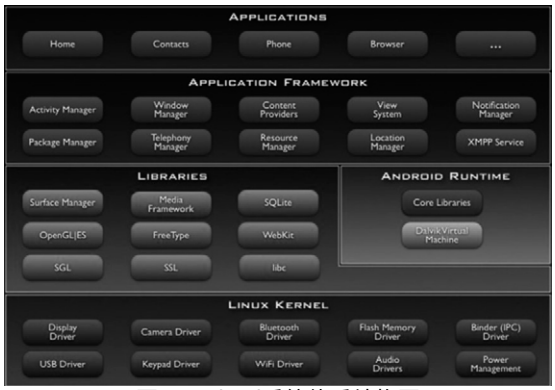

图1 Android系统体系结构图

由图 1 可知，Linux 内核层在 Android 系统体系结构的底层，该层实现了 Android 核心系统服务。此外，Linux 内核层还包含了支持 Android 平台的驱动，且对移动设备做了优化，修改了部分 Bug，是增强的内核。

系统运行库层由 Android 运行时和系统类库两部分组成[11]。Android 运行时由核心库和 Dalvik 虚拟机两部分组成。核心库包含了 Java 程序开发所需要的核心类库，实现了 JDK 中 rt.jar 包所提供的绝大多数 API。Dalvik 虚拟机是基于寄存器的 Java 虚拟机，适应内存有限并且处理速度有限的移动设备。系统类库是为 Android 系统的各个组件使用而提供的，这些系统库基本上都用 C/C++ 语言开发完成。系统类库非常重要，作为一个重要纽带，连接了上面的应用程序框架层和下面的 Linux 内核层。

应用程序框架层封装了底层的系统类库，提供了 API 接口并简化了 Android 组件之间的重用，使得开发者能够根据需求来使用不同的组件并调用相关的 API 接口，为开发者高效开发 Android 应用程序提供了保障。

Android 应用程序可以分为两种，一种是 Android 操作系统自带的核心应用程序，另一种是 Android 应用程序开发者开发的应用程序。

1.2 APK文件结构分析

Android 应用程序开发者在编码完成之后，会将源代码文件、资源文件、AndroidManifest.xml 配置文件等文件通过 aapt 资源打包工具、aidl 工具、编译器、dx 工具以及 apkbuilder 工具进行处理，生成 APK 文件。APK 文件是一个压缩文件，是应用程序安装包，解压后的内容如图 2 所示。

图2 APK文件结构图

META-INF 文件夹中存放了签名信息，Android 应用程序开发者在发布 APK 文件之前需要对 APK 文件进行签名 [12,13]。采用 Android 签名机制给 Android 应用程序打上标记 [14]，有助于防止 Android 应用程序被随意篡改。

APK 文件中的一些资源文件存放在 res 文件夹中，如布局资源和图片资源。

AndroidManifest.xml 文件位于 APK 文件解压后的根目录下，包含了Android 应用程序的相关配置信息和声明信息。AndroidManifest.xml 文件还包含了应用程序的启动方式、命名空间、SDK 最小版本号以及应用程序的包名等信息。

DEX 文件（classes.dex 文件）整合了组成该文件的所有 class 文件中的常量，对于重复的字符串常量只保留了一份，消除了其中的冗余信息，从而重新组合成一个常量池，这样所有的类文件就会共享同一个常量池 [15,16]。DEX 文件使类的查找更容易，减少了运行时所需要的内存。

resources.arsc 文件是编译后生成的资源文件。

1.3 DEX文件结构分析

DEX 文件从逻辑上可以分为 3 部分 ：文件头（Header）、索引区（Table）和数据区（Data）。DEX 文件各部分内容如图 3 所示。

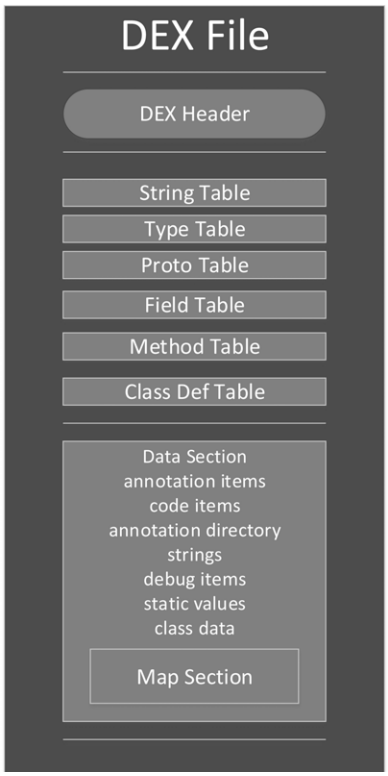

图3 DEX文件结构图

DEX 文件头主要包括 8 字节的魔数 magic、4 字节的 checksum 值、20 字节的 signature 值、DEX 文件长度 file_size、DEX 文件头长度 header_size 以及各字段的个数和各字段的偏移地址等内容。DEX 文件头的长度固定，占 112 个字节。

索引区存放的是 string（字符串）、type（类型）、proto（原型）、field（字段）、method（方法）以及 class（类）等数据的索引值或者在数据区的偏移地址。索引区按照先后顺序存放了 string_id_item、type_id_item 以及 proto_id_item 等内容。

数据区存放了索引区指向的各类数据，包括字符串、类型、原型、字段、方法、类以及 Map 等数据，是 DEX 文件中的重要组成部分。数据区的字符串、类型、原型、字段、方法等数据都从属于某个类，类是数据区的核心内容。类在数据区的表示形式为 class_def_item。

1.4 代码混淆

程序 Program1 经过某种变换变成了程序 Program2，如果 Program1 和 Program2 在输入数据相同的情况下运行得到的结果相同或相似，那么就可以称从 Program1 到 Program2 的变换为一个混淆变换 [17]，如图 4 所示。对程

序代码的混淆变换称为代码混淆。代码混淆的一般定义是：通过各种混淆技术对程序代码或数据进行保留语义的变换，在保持混淆前后程序功能一致性的前提下，降低反编译后代码的可读性，使得攻击者对混淆后的代码进行逆向分析的难度增大，进而达到保护软件代码安全的目的[18]。

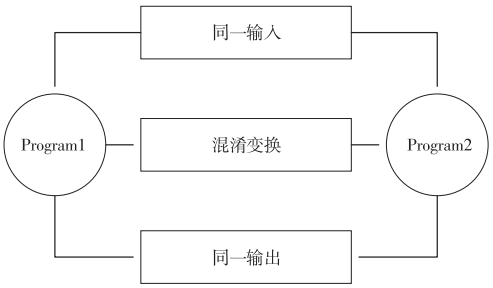

图4 混淆变换

按照代码混淆的原理和混淆内容的不同，代码混淆可以分为 4 类：外形混淆、数据混淆、控制混淆和预防混淆[19,20]。

外形混淆的原理就是通过对函数名和变量名进行变换，尽可能地破坏见名知义的软件工程设计原则，从而达到迷惑逆向工程师以及增大逆向分析难度的目的[21,22]。

数据混淆就是打乱程序中数据的组织形式，变换程序中的数据信息，从而迷惑攻击者。数据混淆主要包括交换静动态数组数据、数值变量糅合以及重构数组[23]3 种类型。数据混淆是导致程序分析难度增大的一种因素，在程序中引入数据混淆会影响逆向分析工程师对程序的分析结果。

程序的控制流包含许多重要信息。控制混淆通过改变源程序的控制流来迷惑和扰乱攻击者获得程序流程，进而阻止攻击者逆向分析程序。

一般情况下，常用的反编译工具都有一些特定的缺陷，预防混淆技术就是利用反编译工具的这些特定缺陷来设计混淆方案[24]。

2 混淆工具的设计与实现

本文设计并实现了一种 Android 混淆工具 AIRO（Android Identifier Renaming Obfuscator），该混淆工具完成了 Android 应用程序中标识符的代码混淆，并且去除了 DEX 文件中一些冗余字节。本章将介绍 DEX 文件的混淆重组。在实现了对 DEX 文件的混淆重组之后，也就完成了对 AIRO 工具的设计。

混淆重组算法通过 APK 文件解压、DEX 文件解析、字符串混淆、DEX 文件重组以及生成 APK 文件等过程实现对 APK 文件的保护。混淆重组算法流程如图 5 所示。

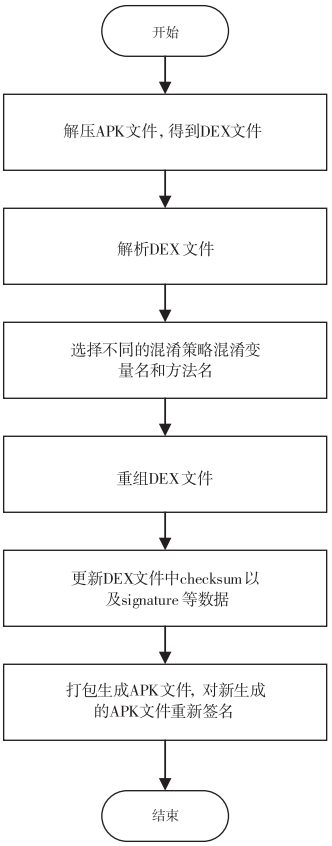

图5 混淆算法流程图

混淆重组算法的步骤如下：

1）对 APK 文件进行解压。解压后得到 classes.dex 文件。

2）对 DEX 文件进行解析。解析流程如图 6 所示。

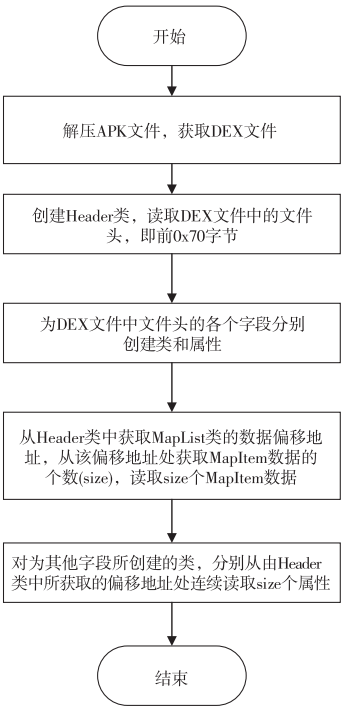

图6 DEX文件解析流程图

从步骤 1）中得到 classes.dex 文件之后，开始对原 DEX 文件进行解析。Android 应用程序中的可执行文件（DEX 文件）主要包括文件头、索引区和数据区 3 个部分。从文件头 (即前 0x70 字节) 可以读取各个索引表的首地址及数据区的首地址。操作步骤如下 ：

（1）为原 classes.dex 文件创建 Header 类，用于存储原 classes.dex 文件的文件头的各个字段。每个字段在 Header 类中用两个属性来分别表示这个字段的个数和这个字段的类型存储在索引区的偏移地址。

（2）为原 classes.dex 文件的文件头的各个字段分别创建一个包含若干属性的类来存储这些字段代表的具体内容。例如，为文件头的 Map 字段创建 MapList 类。MapList 类包括 size 属性 (字节类型)、map_size 属性 (整型) 和 map_size 个 MapItem 类。一个 MapItem 类是 MapList 类的一个属性，表示某一具体 Map 数据。一个 MapItem 类包含 type、unuse、size 和 offset 等属性。

（3）Map 字段在文件头中比较特殊，文件头中没有这个字段的个数，只有这个字段在数据区的偏移地址。从这个偏移地址开始的 4 个字节表示这个字段的个数，将这 4 个字节赋值给 MapList 类的 size 属性，把 size 转化为整数形式赋给 map_size 属性，接着读取 map_size 个 MapItem 数据，完成对 Map 字段数据的获取。

（4）对由步骤（2）为其他字段所创建的类，由于文件头中包含了字段的个数（size）和字段在数据区的偏移地址，故可以从由 Header 类中所获取的偏移地址处连续读取 size 个属性，完成 DEX 文件的解析。

对一些比较复杂的类，不能从文件头获取对应的数据，这些数据在 MapList 类里都有对应的记录。

3）通过对 APK 文件中 AndroidManifest.xml 等文件的分析，能够比较准确地获得 Android 应用程序中需要混淆的变量名和方法名。将得到的可以混淆的变量名和方法名使用不同的混淆策略进行混淆。混淆策略包括 3 种 ：最小词汇排序法、MD5 混淆算法和 SHA1 混淆算法。

4）对变量名和方法名进行混淆后对 DEX 文件进行重组。DEX 文件重组流程如图 7 所示。

操作步骤如下 ：

（1）删除原 DEX 文件，在同一路径下创建同名的目标 DEX 文件，目标 DEX 文件和原 DEX 文件格式大致基本相同，相同的格式便于用统一的程序对 DEX 文件进行更新、插入和删除等操作。

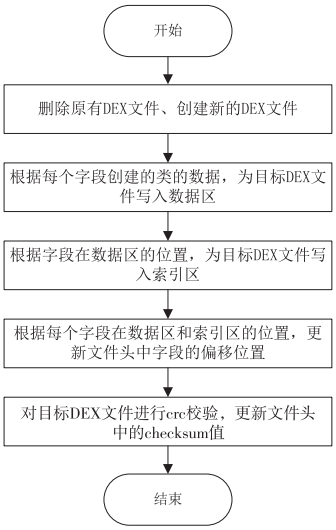

图7 DEX文件重组流程图

（2）从内存中读取步骤 2）所创建的类的内容，写到目标 DEX 文件中，包括 ：

①根据描述原 DEX 文件各个字段的类，将类中的 Item 数据依次写入目标 DEX 文件的数据区。而对于 Map 字段数据，应该在前 4 个字节先写入该字段的个数，再依次将数据写入目标 DEX 文件。

②针对目标 DEX 文件，从 0x70 地址处（文件头结束地址）将索引区的相关信息也根据为各个字段创建的类写入。

（3）根据目标 DEX 文件数据区中各个字段的存储位置以及索引区中的偏移地址，重新矫正目标 DEX 文件头中各个字段的偏移地址。

（4）对整个目标 DEX 文件进行 crc 校验，对得到的校验值的表示形式进行转换，更新 checksum 的内容为对目标 DEX 文件进行 crc 校验的校验值，完成 DEX 文件重组，得到重组后的目标 DEX 文件。

5）完成 DEX 文件重组操作之后，用重组后的目标 DEX 文件替换原 DEX 文件，完成混淆操作。将原解压文件重新打包生成 APK 文件，并使用签名证书 keystore 对 APK 文件进行签名。

总的来说，首先获得 classes.dex 文件，根据 DEX 文件的文件格式解析出各个字段的 Item 数据。然后根据

AndroidManifest.xml 等文件获得 Android 应用程序中需要混淆的变量名和方法名。接着对字段的 Item 数据进行混淆处理，将处理后的数据按照 DEX 文件的格式要求重新生成目标 DEX 文件。最后替换原 DEX 文件，重新生成 APK 文件并签名。

3 测试与性能分析

代码混淆的目的在于增大逆向分析的难度，使得逆向分析工程师无法从反编译后的程序中得到有价值的信息，且混淆后的程序在不引入执行开销的情况下能正常运行。由于 Android 平台相对于 PC 在计算能力以及存储能力上有很大的局限性，因此 Android 软件对软件防护系统的性能要求更高。

本章首先对 AIRO 混淆工具进行功能测试，然后通过测试数据来分析 AIRO 混淆工具的性能，即通过功能测试和性能分析两方面对混淆工具进行分析。

3.1 AIRO混淆工具功能测试

经过混淆工具混淆的 APK 文件，在正确运行的基础上其功能应和混淆前保持一致。本文在研究过程中，按照代码量将 DEX 文件分为不同的量级，对不同量级的多个不同类别的 APK 文件进行测试，测试结果如表 1 所示。

表1 功能测试结果

DEX 文件大小/Kb	量级	样本数量	运行正确样本量	运行正确率
100~500	1	8	8	100%
500~1000	2	22	22	100%
1000~2000	3	22	20	91%
2000~5000	4	38	32	84%
5000 以上	5	10	8	80%
合计		100	89	90%

从表 1 可以看出，经过 AIRO 混淆工具混淆的 APK 文件运行正确率为 90%。随着 DEX 文件代码量级的增加，APK 文件经过混淆后运行正确率有所降低。通过分析发现，随着代码量级的增加，DEX 文件越来越复杂，源代码中类的数量以及类中包含的变量和方法也越来越多，这就导致判断变量名和方法名是否能够混淆的难度增加。如果判断不够准确，混淆工具就会对系统中不能混淆的变量名和方法名进行混淆，从而导致混淆后的 APK 文件不能正常运行。这个问题是 AIRO 混淆工具下一步需要改进的地方。

3.2 AIRO混淆工具性能分析

COLLBERG 提出了评估混淆的 3 个标准：开销（Cost）、强度（Potency）和弹性（Resilience）。开销指代码混淆带来的额外开销，包括运行速度、程序文件大小以及内存消耗等方面。强度指代码混淆后在程序中引入的混乱程度。弹性指混淆后的代码能够抵抗反混淆工具攻击的程度。COLLBERG 等人提出的又一个评估标准隐蔽性（Stealth）指混淆前后程序的相识性。

评估标准中强度、弹性和隐蔽性这 3 个方面很难精确度量，所以本文从开销这个评估标准入手，重点分析 Android 应用程序混淆前后在时间和空间两方面的差异，对文件大小（代码长度）和时间两方面进行定量分析，从而对 AIRO 混淆工具进行性能分析。

混淆前后 DEX 文件大小如表 2 所示。

表2 混淆前后DEX文件大小对比

混淆前 DEX 大小	混淆后 DEX 大小	增加大小	增加比例	减少大小	减少比例
980Kb	982 Kb	2 Kb	0.2%	--	--
1620 Kb	1600 Kb	--	--	20 Kb	1.2%
13480 Kb	13540 Kb	60 Kb	0.4%	--	--
19820 Kb	19610 Kb	--	--	210 Kb	1.1%
22560 Kb	22213 Kb	--	--	347 Kb	1.5%

由表 2 可以看出，大部分 DEX 文件经过 AIRO 混淆工具混淆后大小都有所减少，因为混淆工具在对 DEX 文件进行重组时，删除了部分冗余字节。

混淆前后应用程序启动时间如表 3 所示。

表3 混淆前后应用程序启动时间对比

混淆前启动时间	混淆后启动时间	增加时间	增加比例	减少时间	减少比例
47 ms	45 ms	--	--	2ms	4.2%
20 ms	19 ms	--	--	1ms	5%
32 ms	34 ms	2ms	6.3%	--	--
37 ms	33 ms	--	--	4ms	10.8%
28 ms	25 ms	--	--	3ms	10.7%

从表 3 可以看出，经过 AIRO 混淆工具混淆后，Android 应用程序的启动时间基本不变，表明混淆工具并不会影响应用程序的启动时间，没有给应用程序的运行带来额外的时间开销。

4 结束语

Android 操作系统的开放性和易用性使其成为最受欢迎的移动终端操作系统之一。Android 应用程序开发离不开大量物力和人力的投入，且每个 Android 智能手机用户都

可以在 Google Play 以及各 Android 应用市场下载 Android 应用程序，使得 Android 应用程序非常容易被逆向分析、篡改以及注入恶意代码，这就给用户和 Android 应用程序开发者带来了巨大的经济损失和安全威胁。

本文提供了一种混淆方案，主要对源码中的变量名和方法名进行混淆。该方案在保证效率的情况下，成功实现了对 DEX 文件的混淆。如果需要进一步提高对 Android 软件的防护程度，则需要综合各种技术来对 APK 文件进行防护，包括 so 库加密、关键代码隐藏等。另外，对 Android 应用程序的保护如果偏向于底层，那么保护效果会更好，但与此同时保护所带来的时间以及空间上的额外开销也更大[25]。所以下一阶段将在不断提高性能的前提下，研究各种技术的综合运用，增大软件混淆及防护的强度。（责编 马珂）

参考文献:
[1] 中国互联网络信息中心 . 第 37 次中国互联网络发展状况统计报告 [EB/OL]. http://222.29.159.114/files/1229000000007E5D/cnnic.cn/hlwfzyj/hlwxzbg/201601/P020160122469130059846.pdf, 2016-01-22.
[2] 王敏惠 . 基于 SIP 的 Android 终端 VoIP 系统研究与开发 [D]. 南京 : 南京邮电大学 ,2014.
[3] 杨勇义 . 基于 Android 平台的软件保护技术研究 [D]. 北京 : 北京邮电大学 ,2012.
[4] 吴家顺 , 高静 . Android App Anti-repackage 方法研究综述 [J]. 科技展望 ,2015, 25(25): 13.
[5]COLLBERG C,THOMBORSON C,LOW D.A Taxonomy of Obfuscating Transformations[EB/OL].https://www.researchgate.net/publication/37987523_A_Taxonomy_of_Obfuscating_Transformations,2016-03-15.
[6]COLLBERG C.Watermarking,Tamper-proofing and Obfuscation Tools for Software Protection [J].IEEE Transactions on Software Engineering, 2002，28(8):735-746.
[7]WANG Chenxi. A Security Architecture for Survivability Mechanisms[D].Charlottesville: University of Virginia,2000.
[8] 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): 813-819.
[9] 赵玉洁 . 面向逆向工程的代码混淆有效性研究与实践 [D]. 西安 : 西北大学 ,2011.
[10] 赵玉洁 , 汤站勇 , 王妮 , 等 . 代码算法有效性评估 [J]. 软件学报 ,2012, 23(3): 700-710.
[11] 钱海龙 . 移动终端应用安全加固关键技术研究 [D]. 北京 : 北京邮电大学 ,2014.
[12] 巫志文 , 李炜 . 基于 Android 平台的软件加固方案的设计与实现 [J]. 电信工程技术与标准化 ,2015, 28(1): 36-37.
[13] 徐剑 , 武爽 , 孙琦 , 等 . 面向 Android 应用程序的代码保护方法研究 [J]. 信息网络安全 ,2014 (10): 11-17.
[14] 王琳 . 基于 Android 平台软件保护方法研究 [D]. 西安 : 西北大学 ,2014.
[15] 丰生强 .Android 软件安全与逆向分析 [M]. 北京 : 人民邮电出版 ,2013.
[16] 王华斌 . 基于 Android 平台的智能终端安全研究 [D]. 北京 : 北京交通大学 ,2013.
[17] 邹宏 , 谢余强 . 混淆技术研究初探 [J]. 信息技术大学学报 ,2015,9(1): 97-99.
[18] 张宝国 . 基于 Java 的代码混淆研究 [D]. 成都 : 电子科技大学 ,2008.
[19] 雷植洲 . 代码混淆技术及其在软件安全保护中的应用研究 [D]. 武汉 : 华中科技大学 ,2007.
[20] 焦义奎 . 基于虚拟机的软件保护系统研究与设计 [D]. 武汉 : 华中科技大学 ,2007.
[21] 刘晓英 , 沈金龙 . 软件开发中的一个重要环节——混淆 [J]. 南京邮电学院学报 (自然科学版),2004,24(1):59-63.
[22] 杨旭辉 , 周庆国 , 韩根亮 , 等 . 一种基于源代码的 Java 代码混淆器的设计与实现 [J]. 甘肃科学学报 ,2015, 27(2): 28-31.
[23]COLLBERG C.CS620 Security through Obscurity[EB/OL].http://www.cs.arizona.edu/.collberg/Teaching/SoftwareSecurity.html,2016-03-15.
[24] 刘衍斐 , 封莎 . 移动应用软件防篡改技术研究 [J]. 现代电信科技 ,2015(1): 66-74.
[25] 张玉清 , 王凯 , 杨欢 , 等 .Android 安全综述 [J]. 计算机研究与发展 ,2014, 51(7): 1385-1395.