

基于 API 调用的抗混淆 Android 应用相似性检测方法

谷杰铭^{1,2}, 孙博文^{1,2}, 吴鹏³, 李祺^{1,2}, 郭燕慧^{1,2}

(1. 北京邮电大学网络空间安全学院, 北京 100876;
2. 天地互联与融合北京市重点实验室, 北京 100876;
3. 四川大学计算机学院, 四川 成都 610015)

摘要: 传统的移动应用相似性检测方法存在特征提取复杂、检测效率较低等问题。针对上述问题, 提出基于 API 调用的抗混淆 Android 应用相似性检测方法, 首先从 DEX 文件中提取引用 API 序列, 应用反编译后对引用 API 进行频数统计; 然后构建应用的特征向量, 以此计算应用之间的相似度。实验结果证明, 该方法具有较高的准确率。

关键词: 应用相似性; Android; 逆向工程; 抗混淆

中图分类号: TP399

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2018001

Anti-obfuscation Android application similarity detection method based on API call

GU Jiemin^{1,2}, SUN Bowen^{1,2}, WU Peng³, LI Qi^{1,2}, GUO Yanhui^{1,2}

1. School of Cyberspace Security, Beijing University of Post and Telecommunications, Beijing 100876, China;
2. Beijing Key Laboratory of integration with the world Internet, Beijing 100876, China;
3. School of Computer Science, Sichuan University, Chengdu 610015, China

Abstract: The traditional Android application similarity detection methods have the problem of complex feature extraction and low detecting efficiency. Regarding the issue above, an anti-obfuscation Android application similarity detection method based on API call was proposed. Firstly, it extracts referenced API sequence from the dex file. Counting the number of referenced API after decompiling the application to build feature vector. Finally, calculate the application similarity through feature vector. The experimental results show that the method has high accuracy.

Key words: application similarity, Android, reverse engineering, anti-obfuscation

1 引言

Android 系统自 2007 年发布后迅速占据了移动终端平台的主导地位。数据显示, 截至 2016 年第 3 季度, Android 操作系统的市场份额达到 86.8%^[1], Android 应用的数量随着操作系统的普

及而快速增长, 越来越多的开发者加入 Android 应用的开发队伍中, Google Play 的 Android 应用数量在 2017 年 1 月增长到 270 万, 并且在 2016 年的下载量突破 640 亿次^[2]。

Android 操作系统的开放性给开发商提供了宽松自由的环境, 开发者可以自主开发 Android

收稿日期: 2017-10-17; 修回日期: 2017-12-05

通信作者: 谷杰铭, gujiemin35@163.com

基金项目: 国家自然科学基金资助项目 (No.61401038)

Foundation Item: The National Natural Science Foundation of China (No.61401038)

应用并将应用提交到应用市场,用户通过应用市场下载应用。Google Play 采用多种安全机制对提交的应用进行安全检测,并不断完善安全审核机制以提高检测准确性与全面性,但是依然无法避免仿冒应用和恶意应用的存在。应用市场的应用审核机制不严格,安全保障机制不够完善,导致市场内仿冒应用和恶意应用的比例居高不下^[3]。

由于 Android 应用反编译后的源代码容易阅读,部分开发者为了牟取私利,对市场上的知名应用进行逆向工程,对应用稍作修改,在嵌入广告或恶意代码后重新打包,作为自己的产品上传到应用市场^[4]。这种行为严重扰乱应用市场环境,影响相关产业的健康发展,损坏开发者的合法权益。

应用市场的运营方应该负担起上架应用的审核和管理工作,避免仿冒应用应用在市场上出现。多数应用市场以书面申请为主、人工测评为辅的方式审核应用版权及内容,这种审核方式工作量大、人工参与度高、效率较低。因此,需要一种应用相似性检测工具来提高应用市场的审核工作效率。

2 相关工作

现有的 Android 应用相似性检测方法^[5]主要包括行为相似性检测方法和文件相似性检测方法。在行为相似性检测方面,Hanna 等^[6]提出一种根据行为相似性检测 Android 应用代码重用的方法,该方法首先通过反编译 APK 得到 Android 可执行字节码,从中提取应用的行为序列,采用特征散列算法对行为序列进行处理,得到应用的行为特征,然后采用群落系数相似性度量方法计算行为特征相似性,该方法适用于检测应用重打包、代码抄袭。DroidMOSS^[7]通过反编译 APK 得到 Android 可执行字节码,从中提取应用的指令序列,然后对指令序列采用散列算法得到应用的行为特征,比较行为特征得到应用间的相似度,该方法可以定位到二次打包应用的代码修改点。

在文件相似性检测方面,Li^[8]采用文件目录结构对应用相似性进行评估,用树结构表示应用的目录结构,计算树之间的编辑距离,由此得到应用之间的距离。该方法不受代码混淆的影响,适用于检测应用重打包、嵌入已知恶意代码的应用。

上述研究存在以下问题。一是特征提取,行为相似性检测首先需要逆向分析得到应用行为序列,但是逆向过程容易受到代码混淆的影响^[9],重打包人员只需重新配置混淆文件,把方法、字段、包等 Java 元素混淆成无意义的名称,计算得到的散列结果相差较大,导致相似性检测方法失效;文件相似性检测得益于恶意开发者基本不改变文件目录结构,该方法容易受垃圾文件的干扰,在重打包时被恶意插入大量垃圾文件后,文件目录结构和文件数量发生变化,进而大幅度影响相似性检测结果^[10]。二是检测效率,传统检测方法将应用反编译到 Java 源代码进行分析^[11],然而反编译得到的源代码不完全符合 Java 原生代码的语法结构,同时在反编译过程中会不可避免地丢失部分信息。分析源代码耗费大量的时间和空间,同时信息丢失情况较严重,造成检测效率低且检测结果与实际情况有较大偏差。

应用相似性检测不但要用精确的应用特征描述样本应用,而且需要提高检测效率^[12]。针对目前研究成果存在的不足,本文提出基于 API 调用的抗混淆 Android 应用相似性检测方法,通过选取合适的抗混淆应用特征,可以有效地解决混淆环境下的应用相似性检测问题。首先分析 DEX 文件结构,提取应用 API 序列,根据方法的定义性质,筛选出引用 API 序列,引用 API 序列由系统方法构成,系统方法不受代码混淆的影响,同时具有普遍性、代表性,能够表示应用行为特征,因此选用引用 API 序列作为应用的抗混淆特征。Smali^[13]语言是 Dalvik 的寄存器语言,Smali 语言在语法上完整地实现了 DEX 可执行文件的所有功能^[14]。与传统反编译 DEX 文件的源代码相比,转换为 Smali 语言效率更高。对 Android 应用反编译后得到的 Smali 文件进行引用 API 的频数统计,能够避免信息丢失的情况出现,同时提高检测效率。

3 DEX 文件结构

构建应用特征向量,首先将 API 信息从 DEX 文件中抽取出来,所以需要对 DEX 文件结构^[15]进行分析,从 DEX 文件的相关结构信息中抽取相关 API 信息。DEX 文件结构如表 1 所示。

表 1 DEX 文件结构

名称	格式	描述
header	header_item	文件头结构
string_ids	string_id_item[]	字符串列表，包含文件中使用的所有字符串
type_ids	type_id_item[]	类型标识符列表，包含文件引用的所有类型（类、数组、基本类型）的标识符
proto_ids	proto_id_item[]	方法原型标识符列表，包含文件引用的所有方法的原型
field_ids	field_id_item[]	字段标识符列表，包含文件中引用的所有字段的标识符
method_ids	method_id_item[]	方法标识符列表，包含文件中引用的所有方法的标识符
class_defs	class_def_item[]	类定义列表
data	ubyte[]	数据区，包含上述各表的全部支撑数据
link_data	ubyte[]	静态链接文件所使用的数据

标识符列表中索引 DEX 文件中的字符串、类型、方法原型、字段以及方法信息，其结构体的开始位置和项数均来自文件头中的记录。

根据 DEX 文件结构，下面对标识符列表中结构体的含义进行说明。

首先，针对 `method_id_item` 进行信息描述。`method_id_item` 描述方法的相关信息，包括方法

的所属类、方法原型和方法列表，如表 2 所示。

然后，本文分析重点 `proto_id_item` 的信息描述。`proto_id_item` 描述方法原型的相关信息，包括方法返回值类型、方法参数信息和方法描述，如表 3 所示。

最后，对 `class_def_item` 的信息描述。`class_def_item` 描述具体的 class 类型，如表 4 所示。

表 2 `method_id_item` 描述

名称	描述
class_idx	类型标识符列表的索引，用于描述方法的所属类。所指向的类不能为基本数据类型
proto_idx	方法原型标识符列表的索引，用于描述方法的原型
name_idx	字符串列表的索引，用于描述方法的名称

表 3 `proto_id_item` 描述

名称	描述
shorty_idx	字符串列表的索引，由方法的返回类型和参数列表组成的字符串，用于描述方法原型
retrun_type_idx	类型标识符列表的索引，用于描述方法原型的返回值类型
parameters_off	指向 <code>DexTypeList</code> 结构体。 <code>DexTypeList</code> 结构存放方法的参数列表，如果方法没有参数，值为 0

表 4 `class_def_item` 描述

名称	描述
class_idx	类型标识符列表的索引，描述具体的 class 类型
access_flags	描述 class 的访问类型，如 <code>public</code> 、 <code>final</code> 、 <code>static</code> 等
superclass_idx	类型标识符列表的索引，描述 superclass 的类型
interfaces_off	如果类中含有接口声明或实现， <code>interfaceOff</code> 会指向一个 <code>DexTypeList</code> 结构，否则这里的值为 0
source_file_idx	字符串列表的索引，表示类所在的源文件名称
annotations_off	指向注解目录结构，根据类型不同会有注解类、注解方法、注解字段与注解参数，如果类中没有注解，值为 0
class_data_off	指向 <code>DexClassData</code> 结构，记录类用到的数据
static_value_off	指向 <code>DexEncodedArray</code> 结构，记录类中的静态数据

4 整体检测流程

Android 应用相似性检测方法的整体流程如图 1 所示。通过分析 DEX 文件头定位到方法标识符列表, 根据标识符列表中的地址偏移提取应用 API 序列, 根据方法所属类是否在 `class_defs` 区域, 筛选出引用 API 序列。得到引用 API 序列后, 使用 APKTool 反编译应用得到 smali 文件夹, 然后统计每个引用 API 的总调用频数并得到引用 API 序列的频数序列, 共同构建应用的特征向量, 最后计算应用之间的相似度。本文方法过滤自定义类方法和第三方库函数, 不受代码混淆的影响, 同时避免特征在使用上的不足, 保留应用特征的整体性和代表性。

方法检测流程主要分为 6 个关键步骤, 分别是提取 API 序列、筛选引用 API 序列、应用反编译、频数统计、构建特征向量、相似性计算。

1) 提取 API 序列

定义应用 API 序列为 $\{API_1, API_2, \dots, API_n\}$, 其中, n 为方法数量。分析 DEX 文件结构可知, 方法信息存放在方法标识符列表 (`method_ids`) 指向的数据区中, 从数据区提取方法信息, 得到方法所属类、方法原型、方法返回类型、方法参数等信息, 根据所得信息构建应用 API 序列。

通过实际应用 (MD5: CAF89261805E496528490F337A49861A) 展示应用 API 序列的提取过程。根据 DEX 文件头中的地址偏移, 定位到方法标识符列表 (`method_ids`) 和方法原型标识符列表 (`proto_ids`), 从中读取应用的方法信息^[16]。

根据 DEX 文件头中记录的大小和偏移地址定位到方法标识符列表 (`method_ids`)。方法标识符列表 (`method_ids`) 是一个结构体数组, 该数组的元素为 `method_id_item` 结构体。遍历结构体

数组, 利用获得的 `class_idx` 和 `name_idx` 分别从类型标识符列表 (`type_ids`)、字符串列表 (`string_ids`) 中得到方法所属类和方法名称。

根据 DEX 文件头中记录的大小和偏移地址定位到方法原型标识符列表 (`proto_ids`)。方法原型标识符列表 (`proto_ids`) 是一个结构体数组, 该数组的元素为 `proto_id_item` 结构体。遍历结构体数组, 利用获得的 `shorty_idx` 从字符串列表 (`string_ids`) 中得到方法返回。

类型和参数列表, 利用获得的 `parameters_off` 从 `data` 数据区得到方法参数信息。

提取结果显示应用的 API 序列共有 10 975 项。

2) 筛选引用 API 序列

引用 API 序列由系统方法构成, 系统方法不受代码混淆的影响, 同时具有普遍性、代表性, 能够表示应用行为特征, 因此选用引用 API 序列作为应用的抗混淆特征。提取的 API 序列中包含应用自定义类方法和第三方库函数, 需要过滤两者避免代码混淆的影响。DEX 文件结构中, `class_defs` 区域内存放 class definitions, 包含应用中所有定义的类信息。根据方法所属的 class 类型是否在 `class_defs` 区域, 判定方法是本地方法还是引用方法, 从而筛选出引用 API 序列。

筛选结果显示应用 (MD5: CAF89261805E496528490F337A49861A) 的引用 API 序列共有 2272 项。引用 API 序列过滤自定义类方法和第三方库函数, 只保留应用中的系统方法。

3) 应用反编译

应用反编译是检测工作的基础, 针对 Android 应用的反编译, 主要是反编译应用中的 DEX 文件和对解码资源文件^[17], 检测方法需要用到的 smali 文件包含在 DEX 文件中, 所以本文选取 APKTool^[18]反编译 APK, 得到 smali 文件。

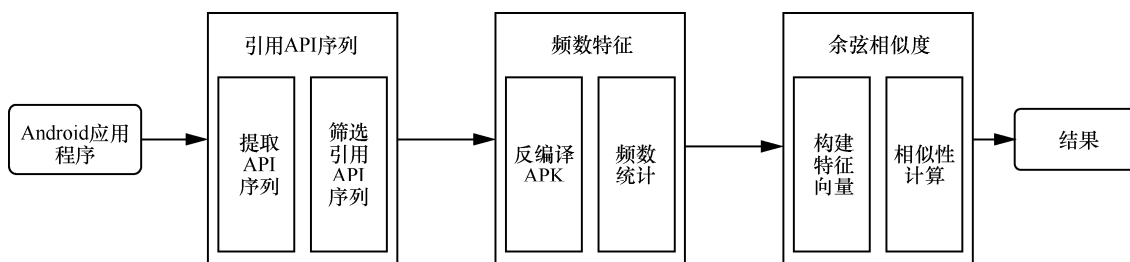


图1 Android 应用相似性检测方法流程

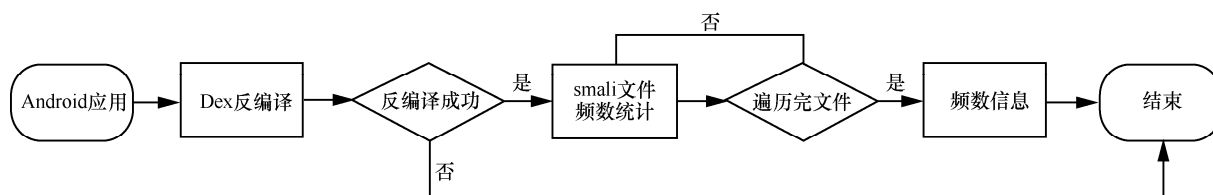


图2 频数统计流程

4) 频数统计

频数统计流程如图2所示，首先调用APKTool对应用进行反编译处理，得到Smali文件夹。Smali语言在语法上完整地实现了DEX文件的所有功能，包括程序代码、注释等大量信息^[13]，因此对得到的smali文件进行频数统计，可以避免信息丢失的情况出现。这里的频数统计指的是统计引用API在文件中出现的总调用频数，最后得到引用API序列的频数序列，跟引用API序列一起构建应用的特征向量。

5) 构建特征向量

应用的特征向量表述为 $A = (A_1, A_2, \dots, A_k)$ 。其中， A_i 表示引用 API_i 被调用的频数。

6) 相似性计算

余弦值的范围在 $[-1, 1]$ ，值越趋近于1，代表2个向量的方向越接近；越趋近于-1，它们的方向越相反^[19]。得到样本应用和可疑应用的特征向量后，计算2个向量的余弦值，就可以知道2个应用的相似性情况。

设特征向量 $A = (A_1, A_2, \dots, A_n)$ ， $B = (B_1, B_2, \dots, B_n)$ 。

$$\cos \theta = \frac{\sum_{i=1}^n (A_i B_i)}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

5 实验

5.1 数据来源

实验选取的样本应用来自国内知名Android应用市场，涵盖10种不同类型，包括社交类、游戏类、影音类、摄影类、工具类、金融类、阅读类、旅游类、生活类、教育类。每种类型选取功能相近300个应用进行相似性检测。

5.2 结果分析

图3为10类样本应用的相似度统计结果，使

用Android相似性检测工具Androguard^[20]进行检测，发现社交类应用的相似度最高，教育类应用的相似度相对较低。结合应用下载量分析发现，使用频率越高的应用类型，应用的相似概率越高。

使用本文提出的基于API调用的抗混淆Android应用相似性检测方法进行检测，对比实验结果，本文提出的方法在相同样本的情况下，检出率更高，具有更好的区分能力。实验结果表明，本文方法在简化特征提取的同时具有较高的准确率。

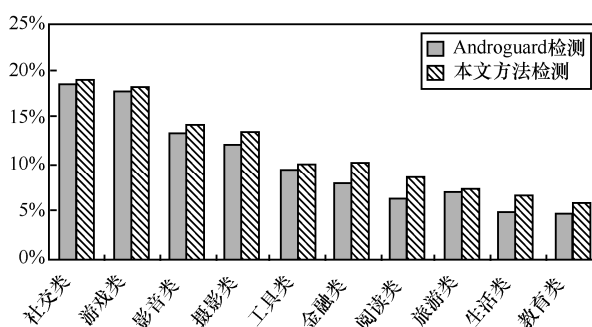


图3 样本相似度

6 结束语

本文提出一种基于API调用的抗混淆Android应用相似性检测方法，从DEX文件中提取引用API序列，应用反编译后对引用API进行频数统计，然后构建应用的特征向量，以此计算应用之间的相似度。特征向量过滤应用自定义类和第三方库，具有很好的抗混淆性。实验结果证明，本文提出的应用相似性检测方法具有较高的准确率。

方法依赖静态检测技术，如果应用采用加固技术进行安全保护，影响API序列提取和频数统计，导致结果失真。后期工作将结合动态脱壳技术对加固应用进行预处理。

参考文献：

- [1] Smartphone OS Market Share, 2016 Q3[EB/OL]. <http://www.idc.com/promo/smartphone-market-share/os>.

- [2] Analysis: how much difference does Google play and App store differ?[EB/OL].<http://www.chinaz.com/manage/2017/0223/662587.shtml>.
- [3] NetQin. Global mobile report of first half of 2012[EB/OL]. <http://cn.nq.com/neirong/2012shang.pdf>.
- [4] SHEKHAR S, DIETZ M, WALLACH D S. AdSplit: separating smartphone advertising from applications[J]. Dissertations & Theses-Gradworks, 2012, 54(1):99.
- [5] ZHOU Y, JIANG X. Dissecting android malware: characterization and evolution[C]//2012 IEEE Symposium on Security and Privacy. 2012: 95-109.
- [6] HANNA S, HUANG L, WU E, et al. Juxtap: a scalable system for detecting code reuse among android applications[C]//The International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. 2012:62-81.
- [7] ZHOU W, ZHOU Y, JIANG X, et al. Detecting repackaged smartphone applications in third-party android marketplaces[C]//ACM Conference on Data and Application Security and Privacy. 2012:317-326.
- [8] LI S. Juxtap and DStruct: detection of similarity among Android applications[EB/OL]. <https://www2.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-111.html>.
- [9] HUANG H, ZHU S, LIU P, et al. A framework for evaluating mobile App repackaging detection algorithms[C]//The International Conference on Trust and Trustworthy Computing. 2013:169-186.
- [10] CHEN K, LIU P, ZHANG Y. Achieving accuracy and scalability simultaneously in detecting application clones on android markets[C]//The 36th International Conference on Software Engineering. 2014:175-186.
- [11] 路程, 张淼, 徐国爱. 基于源代码静态分析技术的 Android 应用恶意行为检测模型[J]. 2011.
LU C, ZHANG M, XU G A. Android application malicious behavior detection model based on source static analysis technology[J]. 2011.
- [12] SOH C, TAN H B K, ARNATOVICH Y L, et al. Detecting clones in Android applications through analyzing user interfaces[C]//IEEE International Conference on Program Comprehension. 2015: 163-173.
- [13] JesusFreke. smali/baksmali: an assembler/disassembler for Android's dex format[EB/OL]. <http://code.google.com/p/smali/>.
- [14] 刘方圆, 孟宪佳, 汤战勇, 等. 基于 smali 代码混淆的 Android 应用保护方法[J]. 山东大学学报(理学版), 2017(3): 44-50.
LIU F Y, MENG X J, TANG Z Y, et al. Android application protection method based on smali code confusion[J]. Journal of Shandong University(Nature Science), 2017(3): 44-50.
- [15] Dedexer: dedexer is a disassembler tool for DEX files [EB/OL]. <http://dedexer.sourceforge.net/>.
- [16] LINARES-VASQUEZ M, HOLTZHAUER A, POSHYVANYK D. On auto-matically detecting similar android apps[C]//The 24th International Conference on Program Comprehension (ICPC). 2016: 1-10.
- [17] ZHOU W, ZHOU Y, GRACE M, et al. Fast, scalable detection of piggybacked mobile applications[C]//The 3rd ACM conference on data and application security and privacy. 2013: 185-196.
- [18] Apktool. A tool for reverse engineering android apk files[EB/OL]. <https://ibotpeaches.github.io/Apktool/>.
- [19] ZHANG Z Y, WANG J, CHENG H M. An approach for spatial index of text information based on cosine similarity[J]. Computer Science, 2005.
- [20] DESNOS A, GUEGUEN G. Android: from reversing to decompilation[J]. Proc of Black Hat Abu Dhabi, 2011.

作者简介:



谷杰铭(1993-), 男, 河北邢台人, 北京邮电大学硕士生, 主要研究方向为移动安全、恶意软件检测。



孙博文(1994-), 男, 辽宁沈阳人, 北京邮电大学硕士生, 主要研究方向为网络安全、恶意软件检测。



吴鹏(1982-), 男, 四川广元人, 四川大学博士生, 主要研究方向为恶意软件检测、软件抄袭、软件安全加固。



李祺(1981-), 女, 北京人, 博士, 北京邮电大学副教授, 主要研究方向为网络安全、机器学习、物联网安全。



郭燕慧(1974-), 女, 河北唐山人, 博士, 北京邮电大学副教授, 主要研究方向为网络安全、机器学习、信息安全管理。