软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(6):1373−1388 [doi: 10.13328/j.cnki.jos.005221]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

大规模移动应用第三方库自动检测和分类方法*

王浩宇[1], 郭 耀[2,3], 马子昂[2,3], 陈向群[2,3]

[1](智能通信软件与多媒体北京市重点实验室(北京邮电大学 计算机学院),北京 100876)

[2](高可信软件技术教育部重点实验室(北京大学),北京 100871)

[3](北京大学 信息科学技术学院 软件研究所,北京 100871)

通讯作者: 郭耀, E-mail: yaoguo@pku.edu.cn

摘 要: 移动应用中,广泛使用第三方库来帮助开发和增强应用功能.很多关于移动应用分析以及访问控制的研究工作,需要在分析之前对第三方库进行检测、过滤或者对其进行功能分类.当前,大部分研究工作都以使用白名单的方式来检测第三方库或者对其功能进行分类.然而,通过白名单检测第三方库不完善且不准确,其原因包括:(1) 第三方库的种类和数量很大;(2) 常见的代码混淆或者第三方库伪装等技术使得白名单方法不能准确地识别第三方库.提出一种第三方库自动检测和分类方法,包括基于多级聚类技术准确识别第三方库以及基于机器学习对第三方库的功能进行准确分类.实验对超过 130 000 个 Android 应用进行分析,验证所提出方法的有效性.实验总共检测到 4 916 个不同的第三方库.在人工标记的数据集上,通过十折交叉验证,对第三方库分类的准确率达到 84.28%.将训练好的分类器应用于全部 4 916 个检测到的第三方库,人工进行抽样验证的准确率达到 75%.

关键词: Android;第三方库;广告库;移动应用;机器学习

中图法分类号: TP311

中文引用格式: 王浩宇,郭耀,马子昂,陈向群.大规模移动应用第三方库自动检测和分类方法.软件学报,2017,28(6):1373−1388 http://www.jos.org.cn/1000-9825/5221.htm

英文引用格式: Wang HY, Guo Y, Ma ZA, Chen XQ. Automated detection and classification of third-party libraries in large scale Android apps. Ruan Jian Xue Bao/Journal of Software, 2017,28(6):1373−1388 (in Chinese). http://www.jos.org.cn/1000-9825/5221.htm

Automated Detection and Classification of Third-Party Libraries in Large Scale Android Apps

WANG Hao-Yu[1], GUO Yao[2,3], MA Zi-Ang[2,3], CHEN Xiang-Qun[2,3]

[1](Beijing Key Laboratory of Intelligent Telecommunication Software and Multimedia, School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China)

[2](Key Laboratory of High Confidence Software Technologies of Ministry of Education (Peking University), Beijing 100871, China)

[3](Software Engineering Institute, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

Abstract: Third-Party libraries are widely used in mobile applications such as Android apps. Much research on app analysis or access control needs to detect or classify third-party libraries first in order to provide accurate results. Most previous studies use a whitelist to identify third-party libraries and manually categorize them. However, it is impossible to build a complete whitelist of third-party libraries and classify them because: (1) there are too many of them; and (2) common techniques such as library obfuscation and library masquerading cannot be handled with a whitelist. In this paper, an automated approach is proposed to detect and classify frequently-used third-party libraries in Android apps. A multi-level clustering based method is presented to identify third-party libraries, and a machine

* 基金项目: 国家自然科学基金(61421061, 61421091); 国家高技术研究发展计划(863)(2015AA017202)

Foundation item: National Natural Science Foundation of China (61421061, 61421091); National High Technology Research and Development Program of China (863) (2015AA017202)

收稿时间: 2016-05-08; 修改时间: 2016-07-15, 2016-09-09; 采用时间: 2016-12-22; jos 在线出版时间: 2017-02-20

CNKI 网络优先出版: 2017-02-20 15:14:44, http://www.cnki.net/kcms/detail/11.2560.TP.20170220.1514.030.html

learning based technique is applied to classify the libraries. Experiments on more than 130 000 apps show that 4 916 third-party libraries can be detected without prior knowledge. The classification result of 10-folds cross validation on sampled libraries is 84.28%. With the trained classifier, the proposed approach is able to classify more than 75% of the 4 916 libraries into six categories with an accuracy of 75%.

Key words: Android; third-party library; advertisement library; mobile apps; machine learning

第三方库是一种重要的可复用软件资源,并且在移动平台得到应用开发者的重视和日益广泛的应用.开发者可以在应用中使用广告库来增加收入,也可以嵌入社交网络库,方便用户的登录或交流,还可以使用各种工具库帮助应用开发和增强应用功能.Google Play 中,有超过 60%的应用使用广告库[1,2],其他第三方库,例如社交网络库和第三方分析库也非常流行.甚至有些应用中,使用超过 20 个第三方库[3].

第三方库的使用,带来新的安全与隐私问题.研究表明:第三方库会对移动用户的隐私造成威胁,甚至包括一些流行的第三方库[4].很多第三方库使用位置信息和设备唯一标识符(UDID)来追踪用户,目的是推送定制化广告或收集用户隐私信息来谋取利益.研究发现,一些流行的第三方库也存在着侵犯用户隐私的行为[5],例如收集用户的邮箱地址等.此外,很多广告库中存在越权行为[4].广告库通常会在其使用文档中说明需要使用的权限以及开发者可选权限,但很多广告库会在运行时动态检查应用是否申请其他敏感权限(例如 READ_CONTACTS 权限),并尝试越权行为.现有的 Android 平台权限机制中,广告库没有跟宿主应用的核心代码权限分离,即,跟宿主应用拥有同样的权限.因此,一旦广告库发现宿主应用拥有某些权限,广告库就会进行一些敏感操作,例如调用 API 来读取用户的联系人信息等.这样,隐私信息会在用户和应用开发者不知情的情况下被泄露给广告网络.更为严重的是,最近研究[5]发现一些以第三方库为中心的安全威胁,例如修改现有的第三方库以及将恶意代码伪装成正常的第三方库等.

很多研究工作关注于移动应用中的第三方库,研究点包括将第三方库与应用核心代码权限分离[6−8],分析隐私信息的使用意图[9−11],分析第三方库中的隐私威胁[12,4]和应用克隆检测[13−19]等.这些研究工作需要在分析之前,对第三方库进行检测、过滤或者对第三方库进行功能分类.

从应用字节码中识别第三方库存在很大挑战,而对第三方库的功能分类更是困难.大部分工作使用白名单(whitelist)的方式来识别第三方库,即,将代码中的包名与已知的第三方库包名进行比较.例如,Centroid[18]使用包含 73 个第三方库的白名单;Lin 等人[9,10]手动标记近 400 个第三方库,并对其进行功能分类.然而,白名单方法基于先验知识,很难手动标记完整的第三方库列表.同时,随着移动平台的发展,不断会有新的第三方库出现,给第三方库白名单的维护增加困难.大部分的研究工作只标记少于 100 个第三方库,远远少于市场中可用的第三方库数量.除此之外,基于包名检测第三方库并不准确:一方面,研究表明[8],很多第三方库都有着不同程度的代码混淆,使得通过比较包名来识别第三方库更加困难;另一方面,Hu 等人[5]研究发现,一些恶意开发者会修改现有的第三方库,并将恶意代码伪装成正常的第三方库.这种情况下,白名单的检测方法会将它们误检测为正常的第三方库.

本文研究目的是准确识别 Android 应用中的第三方库,并对第三方库的功能进行分类,研究结果可以用于优化移动应用分析的相关研究.本文提出自动化的方法来检测 Android 应用中第三方库并对第三方库进行功能分类,该方法主要包括两个关键技术.

- 基于多级聚类的方法来准确识别第三方库.

Android 应用中的第三方库使用有两个特点:(1) 第三方库一般会被很多应用使用;(2) 开发者在使用第三方库时一般不会对其进行修改.因此,可以通过聚类的方法检测第三方库.根据这两个特点,如果对大量应用在代码包(package)粒度提取 API 特征并进行聚类,那么属于第三方库的代码就会被聚到相对较大的集合中.这种方法对于代码混淆也具有鲁棒性,因为聚类所使用的特征跟包名和标识符名无关.这种方法也可以在没有先验知识的情况下检测出第三方库的不同版本.除此之外,通过多级聚类的方法,能够将第三方库完整地识别出(包括核心功能部分以及辅助功能部分),而不是分散的代码包.

- 基于机器学习的方法来对第三方库进行分类.

首先,使用静态分析技术从检测到的第三方库中提取多种不同特征,包括构件级别的特征(例如 Activity 和 Service 的使用)、代码级别的特征(例如 API 和 Intent 的使用)、权限级别的特征(例如使用的权限)以及数据级别的特征(例如数据源).通过使用这些特征来训练一个分类器,对第三方库的功能进行自动分类.

本文实现了一个原型系统,并在超过 130 000 个 Android 应用上进行实验来验证方法的有效性.实验结果表明:工具总共检测到 4 916 个第三方库,比之前研究工作通过白名单方式标记的第三方库数量提高一个数量级以上.在人工标记的数据集上,通过十折交叉验证,对第三方库分类的准确率达到 84.28%.将训练好的分类器应用于全部 4 916 个检测到的第三方库,人工进行抽样验证的准确率达到 75%.

1 背景知识及相关工作

首先介绍 Android 应用中的第三方库以及它们在应用中如何被使用,然后介绍第三方库的功能分类,接着对第三方库检测的动机以及挑战进行介绍.

1.1 Android应用中的第三方库

Android 应用一般由 Java 编写(Android 应用也可以包含用 C/C++编写的原生代码(native code)),然后被编译成 Dalvik 字节码.Android 应用的一个特点是很多应用都使用第三方库.这些第三方库通常是被发布成“.class”或者“.jar”文件.Android 应用开发者可以使用这些库来构建自己的应用.一个应用可以使用多个第三方库,包括广告库、社交网络库和第三方分析库等.在 Android 应用的编译过程中,Java 源代码首先被编译成类文件.然后,这些类文件以及包含第三方库会被编译成 Dalvik 字节码.如图 1 所示:以 Groupon 应用为例,在开发者编写源代码时,主程序与第三方库有明显的界限.然而,当源代码被编译为 Dalvik 字节码后,第三方库与主程序之间的界限很难界定,很难判断一个类文件是否属于第三方库,尤其是当代码的包名被混淆之后.

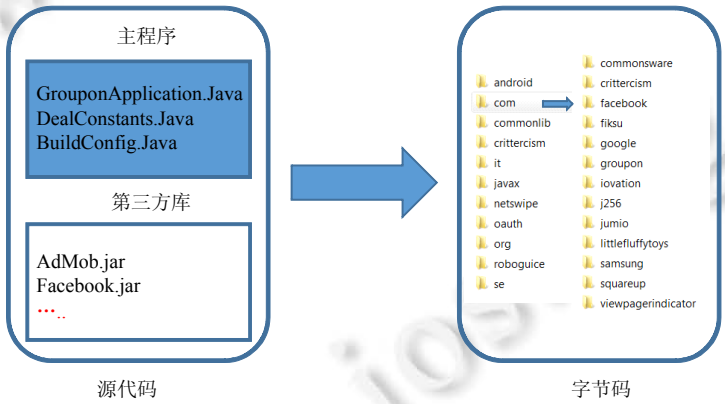

Fig.1 Source code to bytecode in Android

图 1 Android 应用从源代码到字节码示意图

1.2 第三方库的分类

第三方库提供丰富的功能,除了被广泛使用的广告服务,很多第三方库被用来帮助应用开发和增强应用功能.从细粒度分类来看,第三方库可以被划分为许多类别,例如图像处理库(android-jhlabs,JJIL)、3D 引擎库(Dwarf,Godot)、数据库(EasySQLite,Flyway)和字体库(Fontain,FontDroid)等.在本文中,对第三方库的功能分类见表 1.第三方库的分类包括广告库、社交网络库、第三方分析、地图和位置服务、游戏引擎和开发工具.这个功能分类是基于 AppBrain[20]和前人研究工作[9,10]基础上构建.AppBrain 只将第三方库粗粒度的分为 3 类,包括“广告库” “社交网络”和“开发工具”.Lin 等人[9,10]将第三方库的功能手动标记为 9 类,但其中一些类别中只有很少数量的第三方库,例如“二级市场”和“支付”,以及一些类别很相似,例如“开发帮助”和“效用库”.在本文中,将其合并

为同一种类别“开发工具”.除此之外,本文还新增加一个类别“地图和定位服务”,用来表示提供地图或者定位服务的第三方库,例如 OSMDroid.本文将按照此分类来对第三方库进行功能划分.

Table 1 A taxonomy of third-party libraries

表 1 第三方库的功能分类

功能类别	描述	例子
广告库	提供在应用内部嵌入广告,为应用开发者增加收入	AdMob,InMobi
社交网络	在移动应用中嵌入社交网络服务	Facebook,Twitter4j
第三方分析	给应用开发者提供应用的使用数据,包括应用的使用频率以及使用方式等	Flurry,Analytics,Umeng
地图/定位服务	为移动应用提供地图或者定位服务	osmdroid,RouteDrawer
游戏引擎	提供游戏开发框架来帮助开发游戏	Unity3D,Badlogic
开发工具	帮助应用开发	Jsoup,Oauth

1.3 相关工作及挑战

很多研究工作关注 Android 应用中的第三方库,在对应用进行分析之前,需要检测应用中使用的第三方库或者识别第三方库的功能类别.例如:一些研究工作[6−8]关注于对第三方库做访问控制,需要首先识别出第三方库与开发者自定义代码之间的界限;对于应用克隆检测的研究工作[13−19],需要首先识别并且过滤第三方库,以免对应用相似度比较造成干扰;对于分析隐私信息的使用意图[9−11],需要首先识别使用隐私信息的第三方库及其功能.

表 2 中对现有第三方库检测工作进行比较.白名单方法是最常用也是最简单的,但是如前所述,这种方法的准确率以及覆盖率都比较低,不能应对代码混淆以及恶意修改库的问题.AdDetect[21]和 PEDAL[8]使用机器学习方法来检测广告库,它们都从代码中提取各种特征;AdDetect 还使用模块解耦(module decoupling)技术分析代码包之间的依赖关系来找到完整的广告库.但是它们只能够检测广告库,并不能使用机器学习的方法来检测其他类型的第三方库.AnDarwin[22]使用基于语义块的特征聚类,检测的粒度是基于代码块,会造成一定的误报,并且不能将第三方库完整地检测出.此外,Ruiz 等人[2,23]使用正则表达式[aA][dD]匹配的方式来检测第三方库,由于准确率和覆盖率都比较低,与白名单方式类似,因此没有将其列在表 2 中进行比较.Ma 等人[24]实现了一个工具 LibRadar,该工具使用基于聚类的方法来检测第三方库,其目的是为了快速分析应用中使用的第三方库.它能够识别应用中属于第三方库的代码,但不能完整地将第三方库的根目录识别出来;同时,该工作并没有考虑第三方库的功能分类.

Table 2 A comparison of third-party library detection approaches

表 2 现有第三方库检测方法的比较

代表工作	DroidMoss[14],Juxtapp[15]	AdDetect[21]	PEDAL[8]	AnDarwin[22]
方法类型	白名单	机器学习	机器学习	代码块聚类
特征	包名	各种代码特征	各种代码特征	API 特征
检测的第三方库类别	所有第三方库	广告库	广告库	所有第三方库
粒度	代码包	代码包	子包	代码块
覆盖率	低	高	高	高
准确率	低	高	高	一般
处理混淆	否	是	是	是

因此,从应用字节码中识别完整的第三方库并不容易,而对第三方库进行功能分类更加困难.第三方库的自动检测和功能分类主要面临以下挑战.

- 如何处理代码混淆问题?

代码混淆(code obfuscation)是反逆向工程中常用的技术.在 Android 应用中,常用的混淆方式是名字混淆,即,将应用的包名、类名以及标识符名混淆成不易阅读和识别的名字.Liu 等人[8]手动检查 200 个应用,发现其中 107 个应用都包含代码混淆后的广告库.实验中发现,很多应用中的包名被混淆成类似于 com/a/b 格式.同时,很多混淆后的代码没有完整的包名,反编译后在代码的根目录下,使得很难通过比较包名来检测第三方库.此外,

有研究工作[5]也表明:一些恶意开发者会修改第三方库或者创建恶意的第三方库,然后伪装成正常第三方库的包名.因此,仅通过比较包名来识别第三方库并不准确.

- 如何将第三方库完整识别出?

第三方库一般是层级结构,包含着一系列的代码包.如图 3 所示,第三方库“mobwin”在根目录下有 6 个代码包.代码包 com.tencent.exmobwin.core 包含着第三方库的主要功能,其他的代码包提供辅助功能.

有研究[8,19,22,24]提出了一些方法来识别属于第三方库的代码或者构件,然而完整地识别出第三方库并不容易.研究目标是找到每个第三方库的根目录,从而将第三方库完整识别出来,而不是识别出一些分散的代码.需要注意的是:不同的第三方库可能使用相同的包名前缀,使得识别第三方库的根目录更加困难.如图 2 所示,com.tencent.weibo.sdk 和 com.tencent.mm.sdk 是两个提供社交服务的第三方库,而 com.tencent.map 是一个提供地图服务的第三方库,但是它们具有相同前缀 com.tencent.如果存在应用同时使用这些库,需在检测第三方库时不能直接将使用的第三方库检测为 com.tencent 库(事实上也不存在这个库).

- 如何提取有用的特征来对第三方库进行分类?

对于检测出的第三方库,需要提取代表性的特征来对其进行分类.不同种类的第三方库会表现出相似的行为(例如获取用户的位置信息然后与服务器通信),使用单一种类特征对库进行分类比较困难.除此之外,需要保证使用的特征能够应对代码混淆,即使在第三方库混淆的情况下也能够准确检测.之前研究工作均不能自动化地对第三方库进行功能分类.因此,如何提取有代表性的特征来对第三方库进行自动分类是研究的挑战.

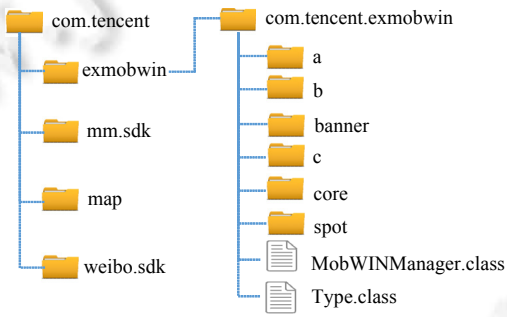

Fig.2 Package Structure of an Ad library “MobWin”

图 2 广告库“MobWin”的目录结构示意图

2 研究方法

2.1 研究思路

为了能够准确识别第三方库并对其功能分类,本文提出如下两个关键技术.

- 自动化的多级聚类方法来准确识别第三方库.

第三方库的使用通常有两个主要特征:第 1 个特征是开发者在使用第三方库时一般不会对其进行修改,开发者会直接将第三方库的 class 或者 jar 文件直接导入应用项目中,因此,不同应用中使用相同的第三方库会包含完全相同的代码特征;第 2 个特征是第三方库会被很多应用使用,因此可以在没有先验知识的情况下,通过聚类的方法来识别第三方库.由于第三方库被很多应用使用,属于第三方库的代码会被聚类到一个比较大的集合中.除此之外,可以通过库的代码层级结构来识别第三方库的根目录.为了将第三方库完整地识别出,本文引入了多级聚类的方法.在反编译的 Smali 代码中,对每一个文件夹都提取特征,然后使用一种自底向上的方式对这些特征进行聚类.例如,如果包 com.google.ads 和 com.google.ads.util 都被聚类成第三方库,那么将会识别 com.google.ads 为第三方库的根目录,而 com.google.ads.util 只是属于第三方库中的一个包.

- 基于多种语义特征的机器学习方法来对第三方库进行功能分类.

对应用进行静态分析来提取多种特征,包括构件级别的特征(例如,Activity,Service,Content Provider Uri 和

Broadcast 的使用)、代码级别的特征(例如 Android API 和 Intent 的使用)、权限级别的特征(例如使用的权限)以及数据级别的特征(例如数据源和数据泄露点).这些特征均与第三方库的功能密切相关.根据这些特征,使用机器学习技术来训练分类器从而对第三方库进行功能分类.

2.2 整体架构

第三方库检测和功能分类的流程如图 3 所示:对于每个应用,首先使用 Apktool[25]将其从 DEX 字节码反编译为 Smali 中间代码;然后,对 Smali 代码中的每个目录提取静态特征,包括高层级目录例如“com.google.ads”,也包括低层级目录例如“com.google.ads.util”,其原因是为了在代码层级结构中找到最大可能的第三方库(第三方库根目录);然后,使用特征聚类的方法来检测第三方库,属于第三方库的包会被聚到相对较大的集合.对于检测出的第三方库,提取多种特征然后使用有监督的机器学习方法来训练一个分类器.这个分类器能够将识别的第三方库划分为 6 个功能类别(或者判断其不属于任意一个类别).

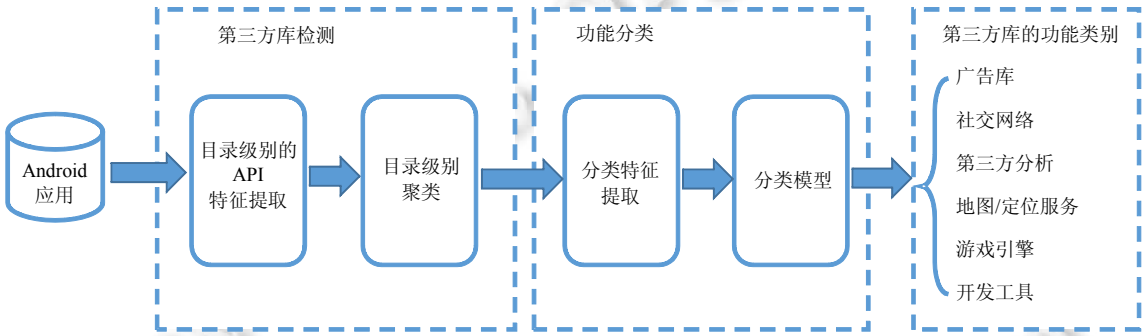

Fig.3 Overview of third-party library detection and classification

图 3 第三方库的自动检测和分类

3 基于聚类技术的第三方库检测

如前所述,在第三方库检测中主要存在两个挑战:代码混淆和识别第三方库的根目录.为了处理代码混淆,在聚类时提取静态常量特征(包括受权限保护的敏感 API,Intent 和 Content Provider),这些特征在对代码的命名混淆中一般保持不变.

为了识别第三方库的根目录,本文使用一个多级聚类的方法,能够找到最大可能的第三方库.

3.1 静态特征

为了进行快速准确的比较并且能够应对代码混淆,所选取的特征即使在代码混淆后仍然能够保持不变,被称为静态语义特征.所使用的特征比较简单但是很有效,包括不同 Android API 的调用、受权限保护的 Intent 和 Content Provider Uri 的使用,同时考虑这些特征的使用频率.尽管可以采用无监督学习方法来提取更为复杂的特征,但本文所使用的特征已经足够有效,因为这些特征与第三方库的行为相关,并且在代码命名混淆[26]之后保持不变.因此,基于这些静态特征进行聚类检测第三方库时比较准确.API 调用特征通过分析 Smali 代码得到.受权限保护的 Intent 和 Content Provider Uri 使用是基于 PScout[27]中的权限映射得到.总共使用 97 个与权限相关的 Intent 和 78 个与权限相关的 Content Provider Uri.在反编译后的代码中寻找 Intent 字符串例如 android.provider.Telephony.SMS_RECEIVED 和 Content Provider URI 字符串例如 content://com.android.contacts 来计算特征.

3.2 识别库的根目录

为了识别第三方库根目录,本文采用多级层次聚类的技术.

首先,对应用进行多级特征提取,即,为代码目录结构中的所有目录都计算特征.如图 5 所示,这个例子中有

两个第三方库 mobwin 广告库和 tencent map 地图库.为了识别第三方库的根目录,对其中所有目录都计算特征,包括 com/tencent,com/tencent/exmobwin,com/tencent/map 和 com/tencent/exmobwin/core 等.使用自底向上的方式计算所有目录的特征.每个目录的特征是其下所有子目录和类文件特征的并集.例如在图 4 中,com/tencent/exmobwin 的特征包含所有子目录和类文件的特征.每个目录由一个特征向量表示.

然后,对各级目录按照特征向量进行聚类.假设第三方库在使用时,开发者一般不会对其进行修改,因此不同应用中使用相同的第三方库会具有完全一致的特征.本文在聚类时进行严格比较,即,只有两个目录的特征完全一致时才会被聚类在一起.在聚类时,先根据特征向量中的特征数目进行排序,对于特征数目一致的特征向量再进行详细比较,当且仅当所有特征完全匹配时将它们聚在一起.

假设应用的数据集足够大并且每个第三方库都被很多应用使用,那么属于第三方库的代码就会被聚到相对较大的集合中.但除了第三方库,开发者自定义的代码也可能被聚在一起.例如:对于相同开发者开发的应用或者重打包应用,它们经常包含很多相同代码.通过设定阈值,就能够将属于第三方库的代码包找出来.阈值的设定将在实验评估部分进行说明.由于要识别出第三方库的根目录,因此对属于第三方库的代码包,本文会将其分类为第三方库(根目录)或者第三方库(子目录),如图 4 所示.

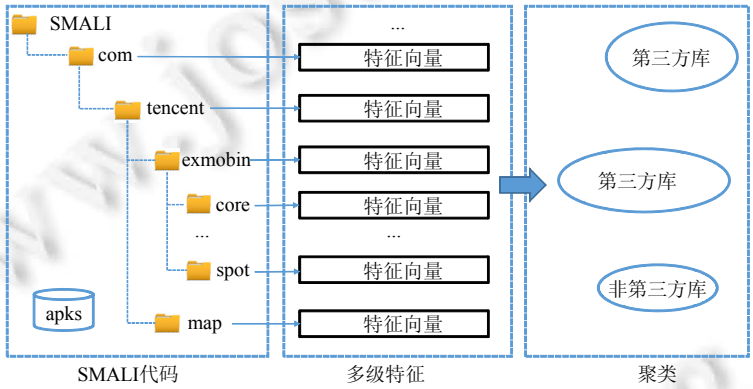

Fig.4 Clustering-based approach to detect third-party libraries

图 4 基于多级聚类的第三方库检测方法

聚类后的结果可以分为 3 类.

1. 非第三方库:这个集合中的目录不属于任何第三方库,它们是应用开发者自定义的代码;
2. 第三方库(根目录):这个集合中的目录属于第三方库,并且它们是第三方库的根目录.例如,com.google.ads 是第三方库 Google ads 的根目录;
3. 第三方库(子目录):这个集合中的目录属于第三方库,但是它们是第三方库中的子目录.例如,com.google.ads.util 属于 Google ads 广告库.

为了区分第三方库(根目录)和第三方库(子目录),使用如下算法对聚类后属于第三方库的包进行遍历.

1. 所有被选择的集合被标记为第三方库(根目录);
2. 对于这个集合中的每个目录,如果它有一个父目录属于第三方库集合,那么这个类就会标记为第三方库(子目录),它不是库的根目录;
3. 对于这个集合中的每个目录,如果它有一个子目录属于其他第三方库集合,那么所有的这些第三方库集合都会标记为第三方库(子目录).

在遍历所有的包并且没有任何集合可以标记为第三方库(子目录)之后,检测过程结束.余下的第三方库(根目录)集合即为检测到的第三方库的根目录.

4 第三方库的分类

对于检测到的第三方库,本文使用有监督的机器学习方法对其进行功能分类:首先,使用静态分析来提取多种不同级别的特征;然后,使用 3 种不同的机器学习方法训练分类器并进行比较.

4.1 特征提取

为了进行准确的分类,本文使用静态分析获取多种不同级别的特征来表示应用的行为,包括构件级别特征、代码级别特征、权限级别特征和数据级别特征.这些特征的描述以及表示方法见表 3.

Table 3 The features used in the classification model

表 3 分类模型所使用的特征

类别	特征	特征描述	特征计算	方法
构件级别特征	Activity	第三方库是否使用 Activity 构件	布尔特征	静态分析
	Service	第三方库是否使用 Service 构件	布尔特征	
	Broadcast	第三方库是否使用 Broadcast 构件	布尔特征	
	Content provider	受权限保护的 Content Provider Uri 的调用次数	一个 78 维的特征向量,其中每个值表示对应 Content Provider 的使用次数	使用 PScout 提供的受权限保护的 Content Provider Uri[28]
代码级别特征	Android API	每种受权限保护 API 的调用次数	一个 680 维的特征向量,其中每个值代表对应 API 调用次数	使用 PScout 提供的受权限保护的 Android API[29]
	Intent	每种受权限保护 Intent 的调用次数	一个 97 维的特征向量,其中每个值代表对应 Intent 的调用次数	使用 PScout 提供的受权限保护的 Intent[30]
	Java 反射	第三方库是否使用 Java 反射	布尔特征	静态分析
	动态加载	第三方库是否使用 DexClassLoader	布尔特征	
权限级别特征	使用的权限	第三方库中使用的权限列表	一个 71 维的特征向量,其中每个值代表对应权限是否使用	使用 PScout 提供的权限映射[27]
	动态权限检查	第三方库是否在运行时检查应用的权限	布尔特征	静态分析
数据级别特征	数据源	第三方库中使用的不同种类的数据源	一个 12 维的特征向量,其中每个值代表对应数据源是否被使用	使用 Susi 提供的 API 与数据源映射关系[31]
	数据泄漏点	第三方库中使用的不同种类的数据泄漏点	一个 15 维的特征向量,其中每个值代表对应数据泄漏点是否被使用	使用 Susi 提供的 API 与数据泄漏点映射关系[31]

- 构件级别特征

这类特征与 Android 构件使用相关,包括 Activity,Service,ContentProvider 和 Broadcast.这些特征与 Android 应用和第三方库的行为很相关.例如,广告库经常使用 Activity 来显示广告.本文使用二元特征来表示 Activity, Service 和 Broadcast 的使用,即,第三方库中是否使用对应构件.对于 Content Provider 的特征,使用一个特征向量来表示.从字节码中提取 Content Provider URI 的使用,然后使用一个包含 78 种受权限保护的 Content Provider Uri 集合[28].每个第三方库的 Content Provider 特征对应一个 78 维的向量,向量中每个元素代表着对应的 Content Provider Uri 使用的频率.

- 代码级别特征

使用 4 种代码级别的特征:API 调用特征、Intent 特征、Java 反射和动态加载特征,见表 3.应用使用 Android API 来访问系统资源(例如 GPS 和 WiFi 等).由于 Android 系统中 API 数量巨大,无法将全部 API 作为特征.本文选取受权限保护的 API 作为特征,即,使用这些 API 需申请权限.本文共使用 680 个有文档说明的敏感 API,一共与 51 个权限相关.通过在代码中搜索 API 字符串,例如“requestLocationUpdates”,来获得 API 的使用频率.每个实例被表示为一个 680 维的特征向量,其中,向量中的每一位表示对应 API 的使用频率.同样,还使用受权限保护的 Intent 作为特征.应用可以使用 Intent 机制启动 Activity,与后台服务进行通信,还可以发起一个广播,或者与手机硬件进行交互等.共使用 97 个 Intent 作为特征,每个实例分别被表示为一个 97 维的 Intent 特征向量,向量中的每

一位表示对应 Intent 的使用频率.

使用布尔特征分别来表示第三方库是否使用 Java 反射和第三方库是否使用动态加载,这些特征与代码的行为十分相关.Java 语言的反射机制是一种动态获取信息以及动态调用对象方法的功能.Android 应用可以动态加载 jar 包或者 apk 文件.为了获取 Java 反射的特征,在反编译后的代码中搜索方法 java.lang. reflect.Method: invoke()和 java.lang.reflect.Constructor:newInstance().

同样,通过在反编译的代码中搜索 DexClassLoader 来分析应用是否使用动态加载特征.

- 权限级别特征

对于不同种类的第三方库,权限使用也会有差别.例如:位置权限经常被用在地图和位置服务类别的第三方库中,而大部分开发工具类型第三方库不会使用.广告库和第三方库分析库经常使用 INTERNET 权限,但是大部分的游戏引擎和开发工具类型的第三方库不会使用这个权限.本文使用 PScout 来计算每个第三方库中使用的权限.PScout 列出 71 个重要的权限以及这些权限保护的 API,Intent 和 Content Provider.因此,每个应用被表示成一个 71 维的向量,向量中的每个值代表是否用到对应的权限.除此之外,有研究工作[4]表明,一些第三方库中存在越权行为.第三方库通常会在文档中说明必须的权限以及可选权限,但是一些第三方库会在运行时检查应用是否申请其他敏感权限.分析发现:动态权限检查经常会出现在广告库中,但在其他类型的第三方库,例如开发工具库中很少见到.因此,使用一个二元特征来表示第三方库是否使用动态权限检查.通过分析应用是否使用特定的权限检查 API 可以得到这个特征.

- 数据级别特征

数据级别的特征包括不同种类的数据源和数据泄漏点,这种类型的特征是通过工具 Susi[31]来得到的.Susi 使用机器学习的方法来对 Android 中的数据源和数据泄漏点进行分类,它将 Android API 调用划分为 12 种数据源和 15 种数据泄漏点.因此,使用一个 12 维的二进制向量表示所用数据源的类型和一个 15 维的二进制向量表示用到的数据泄露点类型.

4.2 分类模型

对于不同种类的特征,它们的数值范围差别比较大.因此,对于提取的特征,首先将它们归一化到[0,1],然后使用机器学习技术来训练分类器.本文总共使用 3 种不同的机器学习算法来进行有监督学习分类,包括朴素贝叶斯[32]、最大熵[33]和 C4.5 决策树算法[34].这些算法是基于 Mallet[35]实现的.然后,本文比较这 3 种分类器在不同度量下的性能.

5 实验评估

实验数据来源于 5 个第三方应用市场中超过 130 000 个 Android 应用,应用的分布见表 4.这些应用及反编译后的代码大约占据 4TB 存储空间.首先对应用进行反编译和预处理,然后提取特征进行第三方库检测,最后对检测到的第三方库进行功能分类.

Table 4 Experiment dataset

表 4 实验数据集

应用市场	应用数量	所占比例 (%)
Anzhi (Anzhi 应用市场.http://www.anzhi.com/)	16 754	12.88
Eoe (EOE 应用市场.http://www.eoemarket.com/)	40 038	30.78
Gfan (Gfan 应用市场.http://apk.gfan.com/)	13 663	10.50
Baidu (Baidu 应用市场.http://shouji.baidu.com/)	24 667	18.97
Myapp (MyApp market. http://android.myapp.com/)	34 940	26.86
总计	130 062	100

5.1 第三方库检测

首先计算每个目录的静态特征,将它们聚类,属于第三方库的代码会被聚到相对较大的集合中.从聚类之后

的结果识别属于第三方库的集合,需要选择合适的阈值.对于识别出的第三方库集合,需要对集合进行遍历来找到第三方库的根目录.

5.1.1 第三方库的阈值

对于聚类结果,需要定义一个准确的阈值来识别第三方库.除了第三方库,开发者自定义代码也可能被聚在一起.例如相同开发者开发的应用或者重打包应用,它们经常包含很多相同代码.一般来说,属于第三方库的包会被聚到一个相对较大的集合,因为它们会被很多应用使用.为了确定第三方库的阈值,主要考虑两个因素:首先,这个阈值需要能够将第三方库代码从非第三方库代码中区分出来;其次,由于对所有目录计算特征并聚类,有可能会将两个或者多个不同的第三方库检测为一个较大第三方库(但这个第三方库根本不存在).例如,一个应用同时使用 com.tencent.weibo.sdk 和 com.tencent.mm.sdk 这两个第三方库.如果选择一个较低的阈值,就有可能将代码包 com.tencent 检测为第三方库(因为很可能有些应用同时使用这两个第三方库),应避免这种情况的发生.

考虑这些因素,本文使用两种方法来确定阈值。

- 在对特征进行聚类之前,人工选择 100 个应用,这些应用中包含超过 60 个已知第三方库.通过手动检查它们反编译后的 Smali 代码,对应用中属于第三方库的代码包进行标记.在聚类之后,检查这些代码包的分布情况,即所属集合的大小;
- 对于聚类后不同大小的集合,随机挑选一些样例检查它们的代码,根据检查结果将它们标记为“第三方库”、“开发者自定义代码”或者“不能判断”.

通过结合这两种方法,最终本文选择 50 作为区分第三方库的阈值.即:大小超过 50 的集合会被标记为属于第三方库,注意,这些集合可能是第三方库(根目录)或者第三方库(子目录);而小于 50 的集合会被标记为不属于第三方库,它们会被聚在一起主要是因为相同开发者的代码复用,或者应用重打包等.

5.1.2 检测结果

对于检测到属于第三方库的集合,为了将第三方库完整识别出,需要找出第三方库的根目录.按照本文所提出的算法对每个集合进行遍历,最终有 4 916 个集合被标记为第三方库(根目录),意味着共检测到 4 916 个不同的第三方库.与现有的基于白名单的方法相比,检测到的第三方库数量提高一个数量级以上.

- 版本数最多的第三方库

由于所用方法执行严格匹配,因此能够识别出不同版本的第三方库,即,使用相同包名但是被聚到不同集合的第三方库.表 5 展示了检测到版本数最多的第三方库.在这个数据中,第三方分析库 Umeng 总共有 52 个不同的版本.对于其他的一些第三方库,例如 Google Ads 和 Facebook SDK,都有超过 20 个不同的版本.

Table 5 Libraries with most versions

表 5 检测结果中版本数最多的第三方库

包名	类别	版本数
com.umeng	第三方分析	52
android.support.v4	开发工具	46
com.google.ads	广告库	23
cn.domob.android	广告库	22
com.waps	广告库	20
com.facebook	社交网络	20
com.kuguo	广告库	17
com.baidu.mapapi	地图和位置服务	17
com.adview	广告库	16
com.admogo	广告库	15

- 应用使用最多的第三方库

然后分析在数据集中应用最多的第三方库.对于具有相同包名的集合,将它们聚类在一起,然后按照包名计算每个第三方库出现的次数.表 6 展示了数据集中使用最多的第三方库.其中,有超过 20%的应用使用 Google

ads 库.这个比例低于之前研究工作[8]的结果,原因是实验中的应用大部分来自国内应用市场,使用的很多广告库例如 Umeng Ads 和 Kuguo 都是针对中国用户.

Table 6 Top 10 libraries by package name

表 6 应用使用最多的第三方库

包名	出现次数	比例 (%)
com.google.ads	27 454	21.11
android.support.v4	24 245	18.64
cn.domob.android	12 945	9.95
com.mobclick.android	8 765	6.74
com.apache	8 519	6.55
com.umeng	8 440	6.49
com.madhouse.android.ads	7 514	5.78
com.kuguo	6 065	4.66
com.feedback	5 313	4.08
com.flurry.android	5 158	3.97

- 代码混淆的第三方库

检测结果发现很多经过代码混淆的第三方库.代码混淆的第三方库主要分为两类:对于第 1 类第三方库集合,其中所有代码包的名字都被混淆,因此无法获取该第三方库的包名,在实验中发现,很多集合中元素的包名都被混淆为类似于 com.a.b 这种格式;对于第 2 类第三方集合,集合中部分代码包的名字被混淆,但是仍然有代码包的名字没有被混淆.因此,可以通过未混淆的包名来分析这些第三方库.例如,对于检测到的第三方库 com.kuguo,其对应集合中的元素包名包括 com.b,com.cooguo 和 com.kuguo 等.这也说明基于聚类的方法能够应对代码混淆,并且能够在没有先验知识的情况下检测第三方库.

5.1.3 检测准确率

本文使用两种方法来分别衡量第三方库检测的误报率和漏报率.

- 对于检测到的第三方库,手动选择 50 个集合(每个集合的大小都超过 50,因此这些集合中共包括超过 4 000 个 package)来检测这些集合中的代码包是否属于第三方库.通过使用 Google 来搜索这些包名(如果没有被混淆)以及在反编译代码中寻找一些特定的字符串进行搜索,最终发现只有一个集合属于误报.这个集合中包名经常出现的模式是 com.mobi.livewallpaper.*,它们是一系列(超过 50 个)的壁纸应用,属于同一个开发者开发,使用几乎完全相同的代码.由于使用 50 作为阈值来识别第三方库,它们会被错误识别为第三方库.但是这种情况非常少见;
- 对于标记为非第三方库的集合,手动选择 500 个集合(每个集合的大小都小于 50,共包括超过 1 000 个 package),然后使用一些已知的第三方库名字来寻找是否存在漏报.结果发现:一些代码包例如 net.youmi.android 本应属于第三方库,但是被标记为非第三方库.然而在已经找到的第三方库中,也存在相同包名的库.经过比较这些代码包,发现漏报的包和检测的第三方库相比,代码可能被开发者或者自动化的代码优化工具进行修改,例如死代码消除,因此,这些代码包的特征与检测到的第三方库略有差别.由于本文假设开发者在使用第三方库时一般不会对其进行修改,因此在聚类时它们并没有被聚在一起,这种情况考虑为漏报.

5.2 第三方库的功能分类

为了对第三方库进行功能分类,本文使用有监督的机器学习方法:首先,基于 App Brain[20]和 PrivacyGrade[3]对 138 个第三方库的功能类别进行手动标记,包括 30 个广告库、25 个第三方分析库、22 个开发工具库、20 个游戏引擎库、21 个地图和位置相关的库以及 20 个社交网络库;然后,使用不同的分类算法来训练分类器并评价分类器的性能;最后,将训练好的分类器对检测到的所有 4 916 个第三方库进行分类.

5.2.1 评价方法

使用十折交叉验证来评价不同分类器的性能,即:将数据集平均划分为 10 份,其中 9 份用来训练,1 份用来测试.为了评价分类器的性能,本文使用不同的分类指标,包括针对单个分类类别的评价指标以及对于分类器的整

体评价指标.

- 单个类别的评价指标

为了分析对每个分类类别的分类性能,实验中对每个类别分别衡量真阳性(true positive)、假阳性(false positive)、真阴性(true negative)和假阴性(false negative)的实例数目.除此之外,还使用评测指标“精确度(precision)”“召回率(recall)”和“F 值(f-measure)”来评价分类的质量.精确度是正确识别的实例数与识别出的实例数的比值.召回率是正确识别的实例数与测试集中实际存在的实例数目的比值.F 值是准确率和召回率的调和平均数.计算方法分别如下:

$$Precision=\frac{TP}{TP+FP},$$

$$Recall=\frac{TP}{TP+FN},$$

$$F\text{-}measure=\frac{2\times Precision\times Recall}{Precision+Recall}.$$

- 分类器的整体评价指标

使用准确度(accuracy)以及宏平均(macro-average)度量指标来衡量文本分类器的精确度和召回率.对于宏平均的度量指标,首先计算每个分类的精确度和召回率,然后计算它们的平均.宏平均是每一个分类性能指标的算术平均,每个分类具有相同的权重.它们的计算方法如下:

$$Accuracy=\frac{\sum_{i=1}^{c}TP_i}{N},$$

$$MacroAvg_{precision}=\frac{\sum_{i=1}^{c}Precision_i}{c},$$

$$MacroAvg_{recall}=\frac{\sum_{i=1}^{c}Recall_i}{c}.$$

5.2.2 在标记数据上的分类结果

十折交叉验证的结果见表 7.最大熵算法表现最好,分类准确度能够达到 84.28%.表 8 展示了对于每个分类的详细结果.对于不同功能类别,结果差别较大.地图和位置服务类别结果最好,精确度和召回率都能够达到100%;社交网络和开发工具类别有较高的精确度,但召回率较低.表 9 中详细展示了最大熵分类器上的混淆矩阵结果.每个元素的值表示十折交叉验证结果之和,每列元素代表分类器预测的分类结果,而每行元素代表实例的真实分类结果.广告库类别有最多的误报(见列 L1,38 个实例中有 10 个属于误报),其中 5 个误报的实例属于社交网络库.开发工具类别有最多的漏报(见行 L6,22 个实例中有 8 个属于漏报),这些漏报的实例大部分被错误的识别为第三方分析库及游戏引擎库.

Table 7 The result of third-party library classification

表 7 第三方库功能分类结果

分类算法	准确度(%)	宏平均精确度(%)	宏平均召回率(%)
朴素贝叶斯	76.43	82.98	77.58
最大熵	84.28	87.36	83.75
C4.5 决策树	68.57	69.48	68.50

Table 8 The result of third-party library classification for each category (Maximum entropy)

表 8 第三方库功能分类结果(最大熵)

类别	精确度(%)	召回率(%)	F-值(%)
广告库	76.83	89.67	81.13
社交网络	100	64.17	69.90
第三方分析	84	95.50	83.17
地图和位置服务	100	100	100
游戏引擎	70.83	86.67	73.24
开发工具	92.50	66.50	66.19

Table 9 The confusion matrix of third-party libraries classification (Maximum entropy)

表 9 第三方库分类结果的混淆矩阵(最大熵算法)

类别	L1	L2	L3	L4	L5	L6	总计
L1:广告库	28	0	1	0	1	0	30
L2:社交网络	5	14	0	0	0	1	20
L3:第三方分析	0	0	24	0	1	0	25
L4:地图和位置服务	0	0	0	21	0	0	21
L5:游戏引擎	3	0	0	0	17	0	20
L6:开发工具	2	0	3	0	3	14	22
总计	38	14	28	21	22	15	138

5.2.3 对所有第三方库的分类结果

然后,将训练好的最大熵算法分类器对检测到的所有 4 916 个应用进行分类.除了已标记的 138 个实例,还有 4 778 个未标记的第三方库.分类器会对每个第三方库计算一个 6 维的相似度向量,其中每个值表示其属于对应功能类别的概率(概率之和为 1).如果其属于最大可能类别的概率大于 30%,则将该第三方库的功能标记属于这个类别;否则,将这个第三方库的类别标记为“其他”.

图 5 展示了最终的功能分类结果(包含 138 个手动标记的实例).结果表明:分类器能够将 75%的第三方库划分为 6 类,但仍有 25%的第三方库被标记为其他类别.其中,有超过 1/4 的第三方库被识别为广告库,仅有约 2%的第三方库被标记为社交网络库或者地图和位置服务库.为了验证分类的准确度,我们手动检查每个类别中最流行的 100 个第三方库.表 10 中了展示检查结果.社交网络库和开发工具库的分类准确度能够超过 94%,而游戏引擎库仅有不到 40%的分类准确度.然后,我们进一步分析游戏引擎库中被错误分类的实例.结果发现:游戏引擎库分类准确度较低的主要原因是用于分类训练的第三方库不足,而游戏引擎又很复杂并且功能各异,从 2D 引擎到 3D 引擎,有些游戏引擎甚至包含复杂的声响系统等.

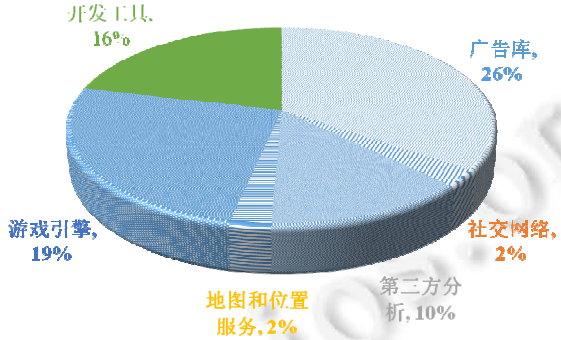

Fig.5 Classification results on all 4 916 libraries detected

图 5 4 916 个第三方库的分类结果

Table 10 The accuracy of classification on all libraries detected

表 10 对 4 916 个第三方库分类的准确度

类别	检查数	TP	FP	不能判断	准确度 (%)
广告库	100	87	9	4	87.00
社交网络	86	81	5	0	94.19
第三方分析	100	70	16	14	70.00
地图和位置服务	71	46	25	0	64.79
游戏引擎	100	39	50	11	39.00
开发工具	100	94	2	4	94.00
总计	557	417	107	33	74.87

总体而言,对于所有未标记的第三方库样本,分类器能够达到 75%的准确度.若不考虑在手动检查中不能判

断的第三方库,分类准确度能够达到80%.尽管如此,该结果仍比在标记实例上做交叉验证的准确率低,其主要原因是用于分类训练的第三方库样本数量不足.

6 讨 论

本节讨论本文研究工作的局限性以及可能的改进方法.

6.1 第三方库的修改

在第三方库检测中,本文假设第三方库在使用时一般不会被开发者修改.然而在实验中我们发现,一小部分的第三方库被修改,因此它们的特征不同.它们可能被开发者或者自动化的优化工具修改,例如进行死代码消除等.由于在特征聚类时使用严格匹配,被修改的库不能被聚到相对较大的集合中.为了检测到这些被修改的第三方库,可以对本文所用的聚类方法进行改进.例如,可以设定一个较低的阈值(例如 95%),当两个代码包的相似度高于该阈值时聚类在一起,或可以使用机器学习方法来检测第三方库的不同版本或者不同实现.

6.2 第三方库的多标签分类

一些第三方库可能属于多种类别.例如,Servo(包名 com.medialets)第三方库同时具有广告和第三方分析的功能.由于本文使用单标签分类方法,所以不能处理这种情况,而且可能造成错误分类.因此在后续工作中,可以使用多标签分类相关技术来对本文工作进行改进.

6.3 第三方库的类别

本文使用机器学习的方法将第三方库划分为6个类别.对4 916个第三方库分类结果表明:对于25%的第三方库我们不能对其进行分类,主要原因是所使用第三方库类别的不完善.此外,根据分类用途的不同,本文所使用的类别可能过于粗粒度.然而,本文的方法可以简单进行改进来检测新的第三方库类别,只需增加新类别的训练样本以及训练一个新的分类器.

7 小 结

本文首先提出了基于聚类的第三方库检测方法,通过提取静态特征,并在代码包级别进行聚类,可以准确检测应用中第三方库,所提出的方法可以在没有先验知识的情况下检测第三方库,能够检测第三方库的不同版本,并且能够应对代码混淆问题;其次,本文提出基于多种特征的机器学习方法来对第三方库的功能进行分类,对于检测到的第三方库,使用静态分析来提取多种不同级别的特征,包括构件级别特征、代码级别特征、权限级别特征和数据级别特征.通过在130 000个Android应用上进行实验,验证所提出方法的有效性.实验总共检测到超过4 900个第三方库.在人工标记的数据集上,通过十折交叉验证,对第三方库分类的准确率达到84.28%.将训练好的分类器应用于全部4 916个检测到的第三方库,人工进行抽样验证的准确率达到75%.

References:

[1] Viennot N, Garcia E, Nieh J. A measurement study of Google play. In: Proc. of the 2014 ACM Int'l Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS 2014). New York: ACM Press, 2014. 221–233. [doi: 10.1145/2591971.2592003]

[2] Ruiz MIJ, Nagappan M, Adams B, Berger T, Dienst S, Hassan AE. Analyzing ad library updates in Android apps. IEEE Software, 2016,33(2):74–80. [doi: 10.1109/MS.2014.81]

[3] PrivacyGrade: Grading the privacy of Smartphone apps. http://privacygrade.org/

[4] Stevens R, Gibler C, Crussell J, Erickson J, Chen H. Investigating user privacy in Android ad libraries. In: Proc. of the Workshop on Mobile Security Technologies (MoST). 2012.

[5] Hu WH, Octeau D, McDaniel P, Liu P. Duet: Library integrity verification for Android applications. In: Proc. of the ACM Conf. on Security and Privacy in Wireless and Mobile Networks (WiSec). New York: ACM Press, 2014. 141–152. [doi: 10.1145/2627393.2627404]

[6] Pearce P, Felt AP, Nunez G, Wagner D. AdDroid: Privilege separation for applications and advertisers in Android. In: Proc. of the 7th ACM Symp. on Information, Computer and Communications Security (ASIACCS 2012). New York: ACM Press, 2012. 71−72. [doi: 10.1145/2414456.2414498]

[7] Shekhar S, Dietz M, Wallach DS. AdSplit: Separating smartphone advertising from applications. In: Proc. of the 21st USENIX Conf. on Security Symp. (Security 2012). 2012.

[8] Liu B, Liu B, Jin HX, View R. Efficient privilege de-escalation for ad libraries in mobile apps. In: Proc. of the 13th Int'l Conf. on Mobile Systems, Applications, and Services (MobiSys 2015). New York: ACM Press, 2015. 89−103. [doi: 10.1145/2742647.2742668]

[9] Lin JL, Amini S, Hong JI, Sadeh N, Lindqvist J, Zhang J. Expectation and purpose: Understanding users' mental models of mobile app privacy through crowdsourcing. In: Proc. of the 2012 ACM Conf. on Ubiquitous Computing. New York: ACM Press, 2012. 501−510. [doi: 10.1145/2370216.2370290]

[10] Lin JL, Liu B, Sadeh N, Hong JI. Modeling users' mobile app privacy preferences: Restoring usability in a sea of permission settings. In: Proc. of the 2014 Symp. on Usable Privacy and Security (SOUPS 2014). 2014.

[11] Wang HY, Hong JI, Guo Y. Using text mining to infer the purpose of permission use in mobile apps. In: Proc. of the 2015 ACM Int'l Joint Conf. on Pervasive and Ubiquitous Computing. New York: ACM Press, 2015. 1107−1118. [doi: 10.1145/2750858.2805833]

[12] Grace MC, Zhou W, Jiang XX, Sadeghi AR. Unsafe exposure analysis of mobile in-app advertisements. In: Proc. of the 5th ACM Conf. on Security and Privacy in Wireless and Mobile Networks. New York: ACM Press, 2012. 101−112. [doi: 10.1145/2185448.2185464]

[13] Crussell J, Gibler C, Chen H. Attack of the clones: Detecting cloned applications on Android markets. In: Proc. of the 17th European Symp. on Research in Computer Security. Berlin, Heidelberg: Springer-Verlag, 2012. [doi: 10.1007/978-3-642-33167-1_3]

[14] Zhou W, Zhou YJ, Jiang XX, Ning P. Detecting repackaged smartphone applications in third-party Android marketplaces. In: Proc. of the 2nd ACM Conf. on Data and Application Security and Privacy. New York: ACM Press, 2012. 317−326. [doi: 10.1145/2133601.2133640]

[15] Hanna S, Huang L, Wu E, Li S, Chen C, Song D. Juxtapp: A scalable system for detecting code reuse among Android applications. In: Proc. of the 9th Conf. on Detection of Intrusions and Malware and Vulnerability Assessment. Berlin, Heidelberg: Springer-Verlag, 2012. 62−81. [doi: 10.1007/978-3-642-37300-8_4]

[16] Wang HY, Wang ZY, Guo Y, Chen XQ. Detecting repackaged Android applications based on code clone detection technique. Science China: Information Sciences, 2014,44(1):142−157 (in Chinese with English abstract). [doi: 10.1360/N112013-00130]

[17] Gibler C, Stevens R, Crussell J, Chen H, Zang H, Choi H. AdRob: Examining the landscape and impact of Android application plagiarism. In: Proc. of the 11th Annual Int'l Conf. on Mobile Systems, Applications, and Services. New York: ACM Press, 2013. 431−444. [doi: 10.1145/2462456.2464461]

[18] Chen K, Liu P, Zhang YJ. Achieving accuracy and scalability simultaneously in detecting application clones on Android markets. In: Proc. of the 36th Int'l Conf. on Software Engineering. New York: ACM Press, 2014. 175−186. [doi: 10.1145/2568225.2568286]

[19] Wang HY, Guo Y, Ma ZA, Chen XQ. WuKong: A scalable and accurate two-phase approach to Android app clone detection. In: Proc. of the ACM Int'l Symp. on Software Testing and Analysis. New York: ACM Press, 2015. 71−82. [doi: 10.1145/2771783.2771795]

[20] Android library statistics. http://www.appbrain.com/stats/libraries/

[21] Narayanan A, Chen LH, Chan CK. AdDetect: Automated detection of Android ad libraries using semantic analysis. In: Proc. of IEEE the 9th Int'l Conf. on Intelligent Sensors, Sensor Networks and Information Processing. Washington: IEEE Computer Society, 2014. 1−6. [doi: 10.1109/ISSNIP.2014.6827639]

[22] Crussell J, Gibler C, Chen H. AnDarwin: Scalable detection of Android application clones based on semantics. IEEE Trans. on Mobile Computing, 2015,14(10):2007−2019. [doi: 10.1109/TMC.2014.2381212]

[23] Mojica Ruiz IJ, Nagappan M, Adams B, Berger T, Dienst S, Hassan AE. Impact of ad libraries on ratings of Android mobile apps. IEEE Software, 2014,31(6):86−92. [doi: 10.1109/MS.2014.79]

[24] Ma ZA, Wang HY, Guo Y, Chen XQ. LibRadar: Fast and accurate detection of third-party libraries in Android apps. In: Proc. of the 38th Int'l Conf. on Software Engineering Companion. New York: ACM Press, 2016. 653−656. [doi: 10.1145/2889160.2889178]

[25] Apktool. Android-Apktool. https://code.google.com/p/android-apktool/

[26] ProGuard. ProGuard. https://proguard.sourceforge.net/

[27] Au YKW, Zhou YF, Huang Z, Lie D. PScout: Analyzing the Android permission specification. In: Proc. of the 2012 ACM Conf. on Computer and Communications Security. New York: ACM Press, 2012. 217−228. [doi: 10.1145/2382196.2382222]

[28] Content provider (URI strings) with permissions. http://pscout.csl.toronto.edu/download.php?file=results/jellybean_ contentproviderpermission

[29] Documented API calls mappings. http://pscout.csl.toronto.edu/download.php?file=results/jellybean_publishedapimapping

[30] Intents with permissions. http://pscout.csl.toronto.edu/download.php?file=results/jellybean_intentpermissions

[31] Rasthofer S, Arzt S, Bodden E. A machine-learning approach for classifying and categorizing Android sources and sinks. In: Proc. of the 2014 Network and Distributed System Security Symp. 2014.

[32] Wikipedia. Naive Bayes classifier. http://en.wikipedia.org/wiki/Naive_Bayes_classifier

[33] Wikipedia. Maximum entropy classifier. http://en.wikipedia.org/w/index.php?title=Maximum_entropy_classifier&redirect=no

[34] Wikipedia. C4.5 algorithm. http://en.wikipedia.org/wiki/C4.5_algorithm

[35] Mallet: MAchine learning for LanguagE toolkit. http://mallet.cs.umass.edu/

附中文参考文献:

[16] 王浩宇,王仲禹,郭耀,陈向群.基于代码克隆检测技术的 Android 应用重打包检测.中国科学:信息科学,2014,44(1):142−157. [doi: 10.1360/N112013-00130]

王浩宇(1991－),男,河南周口人,博士,讲师,主要研究领域为移动计算,安全隐私,程序分析.

郭耀(1976－),男,博士,副教授,CCF 高级会员,主要研究领域为操作系统,软件工程,移动计算.

马子昂(1992－),男,硕士生,主要研究领域为移动应用安全分析.

陈向群(1961－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为系统软件,软件工程,移动计算.