国内第三方 Android 应用市场安全性的检测

闫晋佩 何 晖 安文欢 张小辉 任建宝 齐 勇
(西安交通大学计算机科学与技术系 西安 710049)

摘 要 根据目前第三方 Android 应用市场应用存在的重新打包行为,随机选取国内官方的 150 个应用以及作为对比的第三方应用市场的 572 个同款应用,设计了 Android 重新打包应用安全检测系统。该系统先进行相似性计算,细粒度识别出重新打包应用,再通过逆向工程获得其资源文件,根据系统 API 与权限之间的映射匹配分析其越权行为,并根据构建的方法控制流图分析其权限滥用行为。系统通过并行化处理检测出第三方应用市场存在 33.17%的重新打包应用,其中 19.58%修改了权限。在修改过权限的应用中,45.95%存在越权行为,27.03%存在滥用权限行为。

关键词 Android,重新打包,隐私与安全
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.12.031

Detecting Security of Applications in Chinese Third-party Android Market

YAN Jin-pei HE Hui AN Wen-huan ZHANG Xiao-hui REN Jian-bao QI Yong
(Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract At present, repackaged apps exist in third-party Android application markets. In this paper, 150 official apps are selected randomly and 572 third-party markets apps are used as contrast. Android repackaged apps security detection system was designed. First, we fine-grained identified repackaged apps by calculating their similarity, then gained resource files through reverse engineering, analyzed overprivileged behaviors according to the mappings matcher between system API and permission, and analyzed permission abused behaviors according to constructed methods CFG. By parallel processing, the system detects that there are 33.17% repackaged apps in third-party markets, 19.58% permissions are modified, and modified permission apps include 45.95% overprivileged behaviors and 27.03% permission abused behaviors.

Keywords Android, Repackaged, Privacy and security

随着当今智能手机的普及率越来越高,智能手机应用的种类和数量也迅速增长。最新的报告显示[1],截止 2014 年 7 月 14 日仅 Android 官方应用市场上的应用数量就接近 130 万。这些特色各异的应用,无论是基于用户个性化定制的应用程序,还是基于地域性特色的应用程序,都可供用户进行选择和下载到手机设备上,使得手机本身的功能得到极大的扩展,满足了用户社交、娱乐和工作等诸方面需求。而 Google's Android Market 采用免费或者收费方式提供 Android 应用程序,便于开发者和平台管理者上传和管理应用,防止恶意程序发布。

根据 Android 系统的开源特性和地域性需求,国内外同时也出现了大量的第三方应用市场,国外的第三方市场主要有 Cydia、Amazon AppStore 以及 xda-developer 论坛等;国内的第三方市场主要有机锋市场、安智市场、安卓市场、N 多网市场、木蚂蚁、豌豆荚和优亿市场等。它们凭借着成熟的运作模式和 Android 手机与日俱增的销量迅速发展。然而,这些应用分布在不同的市场中,一些重新打包的应用程序也存在于第三方应用市场中,与原始版本相比有所改动,这些改动可能隐藏着安全风险。可以通过分析第三方市场的应用与官方应用的不同之处对应用程序的安全性进行系统的检测与分析。对此,国内外学者做了大量相关工作,有些学者对收集的 Android 恶意应用进行人工分析与评估,但工程量大且缺乏自动分析系统。有些学者通过权限使用行为对 Android 系统及其应用程序进行检测分析,但不能细粒度分析其内部恶意行为。也有些学者通过开发应用对比工具诊断出重新打包的应用,但使用的方法获取不到函数级控制流的对比结果,不利于进一步分析其越权等行为。

本文设计了一套针对第三方市场 Android 重新打包应用程序的安全检测系统。其中识别子系统是基于哈希算法对解析的 Android 二进制文件进行类和方法相似性的计算,再应用回溯算法对构建的控制流进行对比来获得调用的类、方法、访问类型、接口类以及域等详细信息,从而更全面地识别出重新打包应用;匹配子系统则是基于逆向工程对重新打包的应用进行代码反编译获得应用程序中修改的权限、API 以及方法调用流程图等,根据 Android 系统 API 与权限之间的映射[2],将应用程序中修改的权限和 Java 源码中的 API 进行对

到稿日期:2015-02-08 返修日期:2015-03-17 本文受国家自然科学基金(61272460),国家教育部博士点基金项目(20120201110010)资助。
闫晋佩(1987—),女,博士生,主要研究方向为计算机网络安全,E-mail:yjp2013@xjtu.stu.edu.cn;何 晖(1970—),女,博士,讲师,主要研究方向为操作系统;安文欢(1991—),女,硕士生,主要研究方向为系统安全;张小辉(1989—),男,硕士生,主要研究方向为系统安全;任建宝(1986—),男,博士生,主要研究方向为系统安全;齐 勇(1957—),男,博士,教授,主要研究方向为分布式系统等。

比从而确定申请的权限是否在程序中使用，即是否存在越权情况；分析子系统是根据传统算法[3]建立方法的控制流图并参照申请的权限分析应用中越权的具体情况以及是否存在权限滥用行为。本文选择国内主流五大第三方 Android 应用市场(机锋市场、安卓市场、安智市场、木蚂蚁应用市场、N 多网应用市场)作为评估对象，对下载率较高的 572 个应用进行分析，检测出第三方应用市场存在 33.17%的重新打包应用，其中在对权限进行过修改的应用中，45.95%存在越权行为，27.03%存在滥用权限行为。

本文第 1 节对 Android 系统背景、恶意重新打包行为以及针对本文检测设计方法对前提条件假设进行概述；第 2 节对第三方市场中重新打包应用程序的情况及其安全性检测进行相关实验设计与实现；第 3 节对检测的结果进行分析，并得出评估结论；第 4 节对现有的相关工作进行总结；最后总结全文并展望未来工作。

1 背景概述

目前第三方 Android 应用市场中应用数量极速递增，同时第三方应用市场中存在一些重新打包的应用程序，而这些重新打包的应用可能对程序内容和权限进行恶意修改而不被用户和第三方应用市场发现。在本节中，针对第三方 Android 应用市场对 Android 系统背景进行简单概述，依据第三方应用市场中重新打包的应用程序对恶意重新打包行为进行概述，最后针对本文检测设计方法对前提条件假设进行概述。

1.1 Android 系统背景概述

Android 是一个开放的移动设备操作系统平台，包括了基于 Linux 的内核、中间件和核心的应用。一个 Android. apk 文件包括应用程序代码、数据以及资源信息等内容，能从应用市场上下载。应用程序通常是用 Java 语言写的. class 类文件，并由标准的组件组成。其中，Activity 组件主要控制 UI 屏幕；Service 组件负责后台进程，并不与 UI 交互；BroadcastReceiver组件接收来自 Android 应用程序框架的消息；ContentProvider组件提供对应用程序管理数据的创建、读取、更新以及删除操作。为了协调这些组件，Android 提供了基于 URI 的消息路由系统，发送的消息被称为 Intents。Android 也使用了权限策略来限制应用程序的行为，每一个权限都会保护相关的敏感数据或者具体的操作系统功能。例如，android. permission. READ_CONTACTS 允许程序读取用户联系人的数据，android. permission. BRICK 请求能够禁用设备等。应用程序在被安装之前，展现给用户一个请求权限列表，只有当用户授予应用程序所有请求的权限，应用程序才能被安装。安装之后，用户就不能任意修改程序请求的权限，除非卸载此应用程序[4,5]。

1.2 恶意重新打包行为概述

随着第三方市场中应用程序种类及数量的快速增长，应用程序的“山寨”行为也因利益驱动加速扩展，这些重新打包程序在官方应用中嵌入广告插件甚至恶意代码，通过用户点击广告或者后台下载软件来产生非法利益。这种重新打包行为是恶意应用程序作者盗取用户隐私信息或话费/流量等常用的手段之一。本质上，他们可能会先定位和下载目标应用，将其反编译后进行恶意行为植入，再重新打包并上传到官方或者第三方应用市场。用户下载和授权安装后，这些重新打包程序在手机中会以通知栏、悬浮窗提醒、广告展示等多种形式诱使用户点击，同时还会窃取用户的隐私信息并上传至远程服务器或在后台默默下载各种软件。尤其对于第三方应用市场，应用并非从官方或者 Google's Android Market 上直接下载，所以这些应用当中可能存在重新打包的行为，这会威胁到用户的手机安全。为了功能增值或适应市场特点而进行的重新打包一般不会带来安全方面的问题，但是恶意重新打包行为就会对用户信息安全和个人隐私带来严重威胁。这些行为会严重影响第三方应用市场的发展前景。

1.3 前提条件假设

在本文中，国内的大多数应用程序都有可供下载的官方市场，但是汉化工具、个人开发的游戏和翻译工具等应用并没有相应的官方市场可供下载，所以以 Google's Android Market 为官方市场。系统使用的安全检测方法针对第三方 Android 应用市场，并以官方市场为参照对象。假定官方应用安全可信，无恶意重新打包行为。由于检测和分析的重点针对 Android 应用程序内部的 Java 代码，因此不考虑本地代码，因为本地代码的修改相对困难，而且仅仅只有很少比例(5%～7%)[6]的应用程序包含本地代码。最后，假定来自应用开发者的签名密钥没有泄露，故排除重新打包应用程序被相同的开发者进行签名。

2 系统的设计与实现

根据现存重新打包应用的特性和行为，它们实质上通过植入恶意代码诱导用户或者在后台自动下载应用直接或间接获取利益，使得用户和第三方市场处于被动接受状态。为了能够主动分析重新打包的应用是否是相对安全的，本节主要针对 Android 重新打包应用的识别以及安全性检测进行系统设计及实现。

首先，整个检测分析系统采用并行化多进程的方式进行处理，它由 3 个子系统组成，依照顺序分别是识别子系统、匹配子系统和分析子系统。系统将依次随机爬虫下载 Android 官方市场和第三方市场的主流应用。apk 包作为整个系统的输入，在识别子系统中采用静态方法通过解析的 Android 二进制文件和利用特征哈希方法提取出特征值及其信息计算出对比应用中类和方法的相似性；接着进一步对每个匹配的方法构建控制流并应用回溯算法对需要对比的控制流进行细粒度匹配，从而获得程序所需调用的类、方法、访问类型、接口类以及域等信息，敏感识别出重新打包的应用；然后，在匹配子系统中对识别子系统识别出的重新打包应用进行代码反编译，通过逆向工程转换 Android 文件类型，从而获得 apk 包所需要的资源文件，如 Java 类文件、framework-res 框架以及 AndroidManifest. xml 等，也获得应用程序中修改的权限，API 以及方法调用流程图；再根据 Android 系统中已覆盖 API 与权限之间的映射，应用程序中修改的权限和 Java 源码中的 API 进行关键值匹配从而确定申请的权限是否对应在程序中；最后，根据识别子系统生成的方法控制流图以及匹配子系统检测出越权应用申请所修改的权限，再结合上述生成的源文件进行重新打包应用越权行为的具体分析以及权限滥用情况分析，人工地进一步分析出存在权限滥用行为的应用所使用修改权限的情况，依据应用对应的具体代码分析出检测的重新打包应用是否存在安全隐患，以及可能会威胁到用户的隐私信息。综上所述，检测分析系统设计的总体过程如图 1 所示。

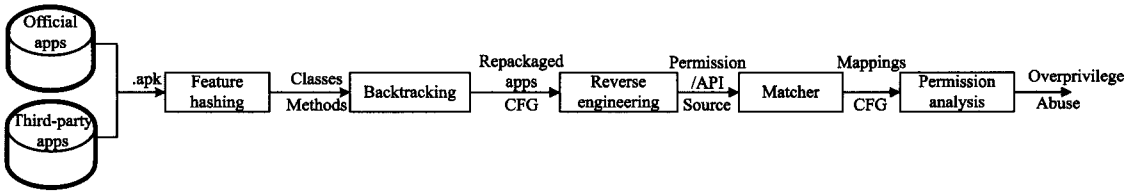

图 1　系统设计总览

结合现存重新打包应用的实际情况，系统选择了多个主流 Android 第三方应用市场进行比较，确定官方应用程序的数量以及发布最新的版本号，在第三方应用市场则相应地下载相同版本的应用程序。应用的类型包括系统工具、通话通信、影音播放、金融理财、资讯新闻、聊天社区和生活娱乐等。

首先，根据上述检测分析系统的总体设计，先在识别子系统中解析输入的对比文件生成 classes. dex 等文件，执行 classes. dex 文件的二元数组比较进行相似性计算，利用 Hungrian 算法让生成的相似性数组寻找最优的匹配，再利用特征哈希算法高效计算出对比应用中类的相似性和方法的相似性，结果输出类和方法的所有匹配对。应用上述结果对每个匹配的方法构建控制流，即遍历所有指令（条件分支、控制语句、返回和异常抛出）去识别直线顺序的基本块，连接所有的基本块构成的列表，从而形成控制流。再根据所有输入控制图可能匹配的节点对需要对比的控制流进行细粒度匹配，从而获得调用的类、方法、访问类型、接口类以及域等信息文件，可以结合使用 Dexdiff[7] 工具进行类与方法的比对以及识别子系统中 extractdiff 脚本程序具体提取输出文本 summary. txt 中调用的类、方法、访问类型、接口类以及域等信息，列出所识别的重新打包应用信息。

然后，应用逆向工程原理在匹配子系统中利用 apktool 和 dex2jar 反编译出重新打包应用的 Java 类文件、framework-res 框架以及 AndroidManifest. xml 等文件。然而在反编译的过程中，由于重新打包应用本身对程序进行的限制保护，我们无法获得具体的类文件来进行深入分析。接着，利用静态分析从 Android OS 源码中提取出权限说明，捕获每个 API 调用的权限请求，生成 API 与权限之间的映射 allmappings 文件、intentpermission 文件以及 publishedapimapping 文件等，结合反编译后获得应用程序中修改的权限并对对应 allmappings 文件进行源码 API 的匹配搜索。如果在源代码中有调用此 API（不包括类中自定义的同名函数），则说明在应用中使用了所申请的权限；否则程序中可能存在越权行为。

最后，通过识别子系统构建并生成的方法控制流图和匹配子系统检测出越权应用申请所修改的权限，获得存在越权行为应用的对应方法控制流 dot 文件，用 dot 命令将 dot 文件转化为 pdf 文件，以便于进一步分析越权以及权限滥用情况。利用 jd-gui 反编译器有助于下一步具体的人工分析，分析出虽没有存在越权行为，但存在权限滥用行为的权限使用情况，依据应用对应的源代码以及方法控制流图分析出被检测的重新打包应用是否存在植入广告、盗取用户隐私信息或者资费等恶意行为，使用户手机应用环境存在一些安全问题，从而得出第三方 Android 应用市场中重新打包应用的分析结果。具体分析情况见 3.2 节和 3.3 节。

3　实验结果分析

3.1　实验环境及运行结果

本文将第三方 Android 市场重新打包应用程序的安全检测系统在 Dell T605 服务器上实现，其中该服务器配置 8 核处理器以及 16GB 内存，内核使用 Linux 3.12 操作系统，Android OS 则选择 4.2.2 版本。根据实验的设计和环境部署，系统采用并行处理方式充分利用多核 CPU 优势检测完所有应用程序，花费了 4.7 天时间，平均每个应用程序需要匹配处理 3385 个类和 22768 个方法，平均分析一个应用程序时间大约为 12min。

3.2　实验结果

根据系统设计以及实现的可行性，随机从官方市场爬虫下载共 150 个当下流行的应用程序，并从国内主流五大第三方 Android 应用市场对应下载共 572 个应用程序，而应用的种类也繁多，包括金融理财、聊天社区、生活娱乐、输入法、通话通信、系统工具、影音播放、游戏以及咨询新闻。下载应用的数目分布情况见表 1。

表 1　官方和第三方 Android 市场下载应用数量

Android 应用市场	应用总数
官方市场	150
机锋应用市场	91(16%)
安卓应用市场	133(23%)
安智应用市场	113(20%)
木蚂蚁应用市场	114(20%)
N 多网应用市场	121(21%)
总计	722

从表 1 中可以看出，左边列出了官方市场和 5 个第三方应用市场，右边列出了相应市场上下载应用程序总数量及其所占比重。从官方随机下载的 150 个应用程序中发现并没有一个第三方应用市场全部均包括。根据识别子系统中实现的算法过程对五大第三方 Android 应用市场里重新打包的应用进行识别检测，检测结果见图 2。

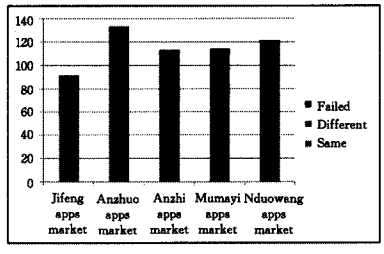

图 2　识别检测实验结果

图 2 所示测试结果显示：对比出不同的应用程序共 189 个，占第三方应用市场的 29.75%～36.28%，平均比例为 33.17%。所以第三方应用市场近 1/3 的应用程序都进行了重新打包。在第三方应用市场测试结果中，测试的 150 组应用程序里有 77 组应用显示完全相同，38 组应用显示完全不同，29 组显示两者兼有。使用工具本身存在的一些问题导致 6 组显示失败，其原因包括重新打包的应用程序嵌套过深导致

栈中内存分配不足，对重新打包应用下标越界的情况进行了中断和 classes.dex 文件结构与分析结果不匹配等。这些均说明使用工具不能满足此类程序的对比分析。

在匹配子系统中，在对识别出的重新打包应用代码进行反编译后，发现重新打包应用情况包括以下几种：增加的资源、删除的资源、修改的资源、权限修改和未知情况。修改情况分类结果见表 2。

表 2　第三方 Android 市场修改分类统计

	机锋	安卓	安智	木蚂蚁	N 多网	总组数/总数(比例)
增加的资源	20	23	21	19	17	42/83(43.92%)
删除的资源	4	4	8	6	5	12/27(14.29%)
修改的资源	28	33	32	28	28	56/149(78.84%)
权限修改	7	9	8	7	6	15/37(19.58%)
未知情况	8	18	13	14	15	22/68(35.98%)

从表 2 可以看出，修改内容的比例最高，占整个重新打包应用程序的 78.84%；修改权限则占 19.58%；未知情况占 35.98%，发现这是由于使用工具存在一些错误，因此出现类文件比对结果一致的情况，也出现类文件和 AndroidManifest.xml 文件反编译失败致使源码和权限无法获取，这并不排除应用程序本身对类文件和 AndroidManifest.xml 文件反编译进行了限制设置。其中第三方修改的内容主要包括修改 API 连接抛出的异常判断、对调用方法出现错误情况的判断、apk 包名、应用 id 号、标签名以及程序执行逻辑等，修改权限主要包括系统权限和自定义权限。由于删减权限行为减少了不安全因素，因此仅对增加的权限进行越权和权限滥用系统分析。增加权限使用情况和系统分析情况分别见表 3、表 4，具体分析见实例分析。

表 3　增加权限使用情况

增加的权限	使用次数
GET_TASKS INSTALL_SHORTCUT(user-defined)	4
ACCESS_COARSE_LOCATION MOUNT_UNMOUNT_FILESYSTEMS READ_PHONE_STATE WRITE_EXTERNAL_STORAGE VIBRATE	3
READ_LOGS WAKE_LOCK READ_SMS ACCESS_FINE_LOCATION CHANGE_NETWORK_STATE WRITE_SETTINGS SYSTEM_ALERT_WINDOW ACCESS_WIFI_STATE UNINSTALL_SHORTCUT(user-defined) READ_SETTINGS(user-defined)	2
RECORD_AUDIO DEVICE_STATE INTERNET ACCESS_NETWORK_STATE CALL_PHONE SIGNAL_PERSISTENT_PROCESSES KILL_BACKGROUND_PROCESSES RECEIVE_BOOT_COMPLETED com.alipay.mobile.command.trigger.permission WRITE_SETTINGS(user-defined) BAIDU_LOCATION_SERVICE(user-defined) DEVICE_STATE(user-defined) com.alipay.mobile.command.trigger.permission(user-defined) BILLING(user-defined)	1

表 4　权限分析结果

Android 应用市场	越权应用	权限滥用应用
机锋应用市场	3	1
安卓应用市场	4	4
安智应用市场	2	1
木蚂蚁应用市场	4	2
N 多网应用市场	4	2
总计	17(8.99%)	10(5.29%)

根据表 3 可知所增加敏感权限的具体情况，左列是所增加的具体权限(系统权限和自定义权限)，右列是使用次数。修改权限的应用中共增加 31 个权限，包括系统权限 23 个，自定义权限 8 个。

表 4 显示出越权和权限滥用应用情况，左列是五大第三方应用市场，中间是越权应用的情况，右列是权限滥用应用的情况。其中，越权的应用占整个第三方市场重新打包应用程序的 8.99%，占权限修改应用的 45.95%，第三方应用市场间的比例占 4.88%～11.11%；权限滥用的应用占整个第三方市场重新打包应用程序的 5.29%，占权限修改应用的 27.03%，第三方应用市场间的比例占 2.44%～9.09%。

3.3　实例分析

本小节对游戏类应用 Flappy Bird(1.3 版本)进行对比测试和实例分析，其在五大第三方应用市场均能下载并测试。测试结果显示，官方应用共 1136 个类和 7737 个方法。机锋市场共 281 个类和 1549 个方法，增加了 279 个类和 1547 个方法，修改使用权限；木蚂蚁和 N 多网市场共 1040 个类和 7459 个方法，修改了 3 个类和 4 个方法，增加了 77 个类和 570 个方法，修改使用权限。经过匹配子系统发现此应用存在越权行为，其中机锋市场增加 7 个权限，删减 1 个权限，申请的 GET_TASKS 和 WRITE_EXTERNAL_STORAGE 等权限在 allmapping 中并没有找到相关 API；而且由于这是一款单机游戏应用，申请并使用 READ_PHONE_STATE 和 ACCESS_COARSE_LOCATION 等敏感权限，这些权限被允许访问 CellID 或 WiFi 热点来获取粗略位置以及允许访问 WiFi 状态等，可能泄露用户的地理位置、手机信息等隐私信息。木蚂蚁和 N 多网市场除了修改 GameActivity.class 类文件中相关类的引入、调用参数名和相关参数初始化，增加了其父类调用、执行命令以及 onPause()和 onResume()方法的调用等内容外，比机锋市场增加了 2 个敏感权限，使被允许访问精良位置(如 GPS)和改写系统设置；其余两个应用市场对比结果相同。经综合分析，此应用超越了游戏使用的基本功能，它在用户不知道的情况下通过连接网络定位用户的准确行踪，存在越权和权限滥用行为。

此外，其他类型应用也存在越权以及权限滥用行为。例如，系统工具类的 WiFi 万能钥匙(2.9.27 版本)修改了 sdk 地图版本号，被允许读取用户短信息；生活娱乐类的大众点评(6.6.5 版本)被允许读取底层系统日志文件等。由于篇幅限制，本节不再展开分析。

4　相关工作

智能手机的安全和隐私一直是近年来关注的焦点，许多工作都以检测智能手机平台的安全隐患为重点。针对官方市场和第三方应用市场智能手机应用安全的相关工作，国内外学者主要从第三方市场应用安全检测方面和应用程序权限使用方面系统地进行检测分析[8,9]。前者通过开发分析工具或

修改系统对应用程序进行检测，主要关注重新打包的应用程序，其中 DroidMOSS[6]通过指印生成技术来检测第三方 Android 应用市场中重新打包的应用；DNADroid[10]检测 Android 应用程序中的相似性并分析了相似性行为；PiggyApp[11]检测现存 Android 市场中 piggybacked 的应用程序；DroidRanger[12]检测现存官方和第三方 Android 市场中的恶意程序，但未根据重新打包应用的变化进行准确定位并细粒度地检测出应用程序中相关类、方法、接口以及域的信息改变情况从而利于进一步的自动分析。后者通过权限使用分析来确定越权行为，其中 Stowaway[13]提取出 Android APIs 到权限之间的映射，并且据此检测应用程序中的越权行为；PScout[2]分析了 Android 系统中的权限映射等。

本文的工作则关注于国内第三方市场重新打包应用程序的现存情况，对国内主流五大第三方 Android 应用市场中 572 个应用程序系统地进行了对比检测分析，具体有效地检测出现在 Android 第三方应用市场重新打包应用程序的所占比例、修改分类，以及修改后存在的越权和权限滥用行为，针对不同情况进行了相应分析。

结束语 本文设计了一套针对重新打包应用程序的安全检测系统，并对国内五大主流 Android 第三方应用市场中重新打包应用程序进行了安全检测及分析。分析结果显示，市场中 29.75%～36.28%的应用程序是重新打包过的。其中，78.84%的应用程序存在资源以及代码修改，19.58%的应用对权限进行了修改，而这些应用程序对官方市场同款应用的标签或者名字进行了替换，对程序逻辑进行了修改。在对权限进行过修改的应用中，45.95%存在越权行为，27.03%存在滥用权限行为。第三方市场需要更严格地检测与分析这些存在的现象和问题，规范并保证第三方应用市场的安全。

目前针对第三方 Android 应用市场中重新打包应用程序的越权行为还未考虑如何从 Android 系统的角度来检测权限是否将用户使用权限修改为 ROOT 权限以及更好地协调用户选取权限与应用功能运行效果。本文工作未深入涉及这些问题，而这些问题将作为本文未来的研究工作。

参 考 文 献

[1] AppBrain[OL]. http://www.appbrain.com/stats/number-of-android-apps

[2] Au K W Y, Zhou Yi-fan, Huang Zhen, et al. D. PScout: analyzing the android permission specification[C]// Proceedings of the 19th ACM Conference on Computer and Communications Security. 2012:217-228

[3] Aho A V, Lam M S, Sethi R, et al. Compilers: Principles, Techniques, and Tools[M]. Prentice Hall, 2006:399-408

[4] Gunasekera S. Android Apps Security[M]. Beijing: Publishing House of Electronics Industry, 2013:37-53

[5] Yang Bo, Tang Zhu-shou, Zhu Hao-jin, et al. Method of Android Applications Permission Detection Based on Static Dataflow Analysis[J]. Computer Science, 2012, 39(11A):16-18

[6] Zhou Wu, Zhou Ya-jin, Jiang Xu-xian, et al. DroidMOSS: Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces[C]// Proceedings of the 2nd ACM CODASPY. 2012:317-326

[7] Mitchell M, Tian Guang-yu, Wang Zhi. Systematic Audit of Third-Party Android Phones[C]// Proceedings of the 4th ACM Conference on Data and Application Security and Privacy. 2014: 175-186

[8] Wu Lei, Grace M, Zhou Ya-jin, et al. The Impact of Vendor Customizations on Android Security[C]// Proceedings of the 20th ACM Conference on Computer and Communications Security. 2013:623-634

[9] Zhang Yuan, Yang Min, Xu Bing-quan, et al. Vetting Undesirable Behaviors in Android Apps with Permission Use Analysis [C]// Proceedings of the 20th ACM Conference on Computer and Communications Security. 2013:611-622

[10] Crussell J, Gibler C, Chen H. Attack of the Clones: Detecting Cloned Applications on Android Markets[C]// Proceedings of 17th European Symposium on Research in Computer Security. 2012:37-54

[11] Zhou Wu, Zhou Ya-jin, Grace M, et al. Fast, Scalable Detection of 'Piggybacked' Mobile Applications[C]// Proceedings of the 3nd ACM Conference on Data and Application Security and Privacy. 2013:185-195

[12] Zhou Ya-jin, Wang Zhi, Zhou Wu, et al. Hey, You, Get off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets[C]// Proceedings of the 19th NDSS. 2012

[13] Felt A P, Chin E, Hanna S, et al. Android Permissions Demystified[C]// Proceedings of the 18th ACM Conference on Computer and Communications Security. 2011:627-637

(上接第 123 页)

Liu Yang. Probabilistic Model Checking-based Modeling and Verification for Service Flow[D]. Shanghai: Shanghai University, 2012

[7] Ricca F, Tonella P. Web site analysis: structure and evolution [C]// Proceedings of the International Conference on Software Maintenance. San Jose, California, USA, 2000:76-86

[8] Tramontana P. Reverse Engineering Web Applications [C]// Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM05). Budapest, Hungary, 2005:705-708

[9] Bellettini C, Marchetto A, Trentini A. WebUml: Reverse Engineering of Web Applications[C]// SAC. 2004:1662-1669

[10] 高洪皓. 基于概率模型验证的 Web 服务动态自适应配置[D]. 上海：上海大学，2012
Gao Hong-hao. Dynamic Self-adaption Reconfiguration of Web Service using Probabilistic Model Checking [D]. Shanghai: Shanghai Univerisity, 2012

[11] 王晶，戎玫，张广泉，等. 基于概率模型检测的 Web 服务组合验证[J]. 计算机科学，2012，39(1)：120-123
Wang Jing, Rong Mei, Zhang Guang-quan, et al. Validation of Web Service Composition Based on Probabilitist Model Checking [J]. Computer Science, 2012, 39(1):120-123

[12] Filieri A, Ghezzi C, Tamburrelli G. Run-time efficient probabilistic model checking[C]// Proceedings of the 33rd International Conference on Software Engineering. ACM, 2011:341-350

[13] van der Meulen M J P, Strigini L, Revilla M A. On the effectiveness of run-time checks[M]// Computer Safety, Reliability, and Security. Springer Berlin Heidelberg, 2005:151-164