DOI:

一种针对 Android 平台恶意代码的检测方法及系统实现

胡文君[1]，赵双[2]，陶敬[1]，马小博[1]，陈亮[3,4]

（1.西安交通大学智能网络与网络安全教育部重点实验室，710049，西安；2.中国科学院信息工程研究所，100093，北京；
3.中电长城网际与国家信息安全测评中心联合实验室数据分析实验室，100093，北京；
4.OWASP 中国北京区，100093，北京）

摘要： 针对 Android 恶意代码泛滥的问题，综合静态和动态分析技术，设计实现了 Android 恶意代码检测系统。在静态分析部分，提取 Android 程序中的权限、API 调用序列、组件、资源以及 APK 结构构建特征向量，应用相似性度量算法，检测已知恶意代码家族的恶意代码样本；在动态分析部分，通过修改 Android 源码、重新编译成内核镜像，使用该镜像文件加载模拟器，实时监控 Android 程序的文件读写、网络连接、短信发送以及电话拨打等行为，基于行为的统计分析检测未知恶意代码。经过实际部署测试，所提检测方法具有较高的检测率和较低的误报率。所开发 Android 恶意代码检测系统已经在互联网上发布，免费提供分析检测服务。

关键词： Android；恶意代码检测；静态分析；动态分析

中图分类号： TP393 **文献标志码：** A **文章编号：** 0253-987X(2013)10-0000-00

A Detection Method and System Implementation for Android Malware

HU Wenjun[1], ZHAO Shuang[2] , TAO Jing[1], MA Xiaobo[1], CHEN Liang[3,4]

(1.Key Lab for Intelligent and Network Security, Xi'an Jiaotong University, Xi'an 710049, China;2.Institute of Information Engineering, CAS, Beijing 100093, China;3. Associated Laboratories of Cyber Space Great Wall and China Information Technology Security Evaluation Center, Beijing 100093, China; 4.OWASP Beijing Area, Beijing 100093, China)

Abstract: An Android malware detection system is designed and implemented to focus on the problem that malware on Android becomes widespread. The system combines static and dynamic analysis technologies. APK features such as permission, API call sequences, component, resource and structure are extracted to form a feature vector in static analysis, and a similarity-based method is proposed to detect known malware samples using these features. Android source code is then updated to generate new kernel images in dynamic analysis. The new kernel images can monitor Android program's behaviors such as file reading and writing, network connection, SMS sending and telephone calling, etc. Unknown malware samples can be successfully identified through analyzing these behaviors. Experimental results show that the proposed system is efficient and performs well on detecting Android malware. The proposed system has been released online and free use of the system is available on the Internet.

Keywords: Android; malware detection; static analysis; dynamic analysis

近年来，智能手机操作系统层出不穷。其中，Android 系统发展最为迅速，2012 年第 3 季度市场占有率达到 75%。Android 系统的繁荣也伴随着其上应用程序数量的爆炸式增长，2012 年 10 月，Google 宣称其官方 Android 电子市场 Google Play 上的应用程序数量已经达到 700000。同时，除了 Google Play 官方市场，还存在 Amazon、安智等众多第 3 方电子市场。

Android 系统快速发展的同时，恶意代码也随之出现。2010 年 8 月，卡巴斯基病毒实验室发现 Android 系统首个木马程序 Trojan-SMS.AndroidOS.FakePlayer。

收稿日期：2012-12-21. 作者简介：胡文君（1989—），男，硕士生；陶敬（通信作者），男，高级工程师. 基金项目：国家自然科学基金资助项目（61175039，60921003，61103241，60574087，91018011，61103240）；国家杰出青年基金资助项目（60825202）.
网络出版时间： 2013-07-03 10:47 网络出版地址：

此后，Android 恶意代码呈现快速发展的趋势，安全公司 F-Secure 2011 年第 1 季度共发现 10 个新的恶意代码家族，而仅仅一年后，2012 年第 1 季度发现 37 个新的恶意代码家族。

Zhou 等在对第 3 方电子市场研究后发现，恶意代码往往通过篡改并将自身注入到正常应用程序（即重打包）的方法，达到伪装并欺骗用户下载的目的[1]。他们提出了 Fuzzy Hashing 的方法实现重打包检测，实验结果显示第 3 方电子市场 5%至 13%应用程序被黑客进行了重打包。

Borja 等根据 Android 应用程序使用的权限，采用多种分类算法对应用程序进行分类，以判别应用程序是否为恶意[2]。实验结果显示，部分分类算法可以获得 80%以上的准确度，但此方法仅考虑了权限信息，不能完全反映 Android 应用程序的特性。实验结果与具体的分类算法相关性较大，不同的分类算法准确度有较大差异。Burguera 等收集同一应用程序在不同用户使用的情况下产生的行为数据，并对产生的行为数据构建特征向量，在此基础上使用 k-means 算法进行聚类，判断应用程序是否恶意，在测试中发现对 PJApps 和 HongToutou 的检测正确率分别为 100%和 85%[3]。但是，该方法需要用户的参与，需要收集用户在使用应用程序的过程中产生的行为数据，因此在真正的实现过程中会涉及用户隐私数据泄漏的问题。Enck 等采用动态 Taint 分析技术监控智能手机上的敏感数据的访问，对 30 个应用程序分析结果显示，有 28 个应用程序存在对用户隐私数据的潜在滥用情况，但是并没有提出具体的恶意代码检测方案[4]。

本文综合了静态分析和动态分析技术，设计实现了 Android 恶意代码分析检测系统，主要贡献如下。

静态分析：以特征匹配为基础，可检测已知恶意代码家族的恶意代码样本。现有工作[2]只以 Android 应用程序权限为特征，有一定局限性，本文则提取 Android 应用程序中权限、API 调用序列、组件、资源以及 APK 结构构建特征向量，可达到更好检测效果。同时，本文对第 3 方电子市场的应用程序，通过与 Google Play 的应用程序进行包名和数字签名的匹配，可快速检测出重打包的应用程序。

动态分析：对 Android 源码进行修改，重新编译生成内核文件，并加载运行模拟器，监控应用程序运行时行为，包括文件读写、网络访问、短信发送、电话拨打等。最后，对行为数据进行模式匹配，检测未知的恶意代码样本。

本文实现了一个 Android 恶意代码分析检测系统，并已在互联网 (http://sanddroid.xjtu.edu.cn) 上发布，免费提供分析检测服务。已经有 405 个不同的 Android 应用程序文件，分析检测出了 20 个恶意代码，其中未知恶意代码 6 个。

1 背景介绍

Android 基于 Linux 内核，主要用于移动设备的操作系统，在架构上可分为 5 部分，从下至上分别是 Linux Kernel，Android Runtime 和 Libraries，Application Framework 以及 Application。Android 应用程序以 APK（Android Package）文件的形式发布，APK 文件实际上是一个压缩包，其结构如图 1 所示。其中 META-INF 文件夹存放应用程序的签名信息，用来保证 APK 包的完整性；res 文件夹存储资源文件，包括图片、字符串、UI 布局文件等；AndroidManifest.xml 是应用程序的配置文件，其中声明了应用程序的包名、SDK 版本、权限、组件等信息；classes.dex 则是 java 字节码文件，可运行于 Android 虚拟机 Dalvik 上。

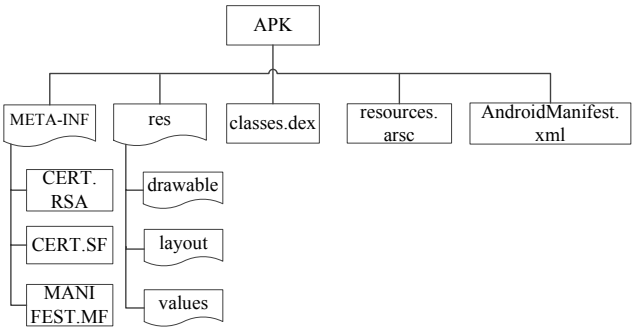

图 1 APK 文件的基本结构

Android 系统具有严格的权限管理机制[5-6]，应用程序使用特定功能时，须在 AndroidManifest.xml 文件中使用<uses-permission>标签声明相应的权限，例如访问网络时，需要声明“android.permission.INTERNET”，否则在程序运行时会触发安全异常。

2 系统设计

本文设计的系统主要由静态分析和动态分析两个模块组成。如图 2 所示，静态分析部分主要通过提取

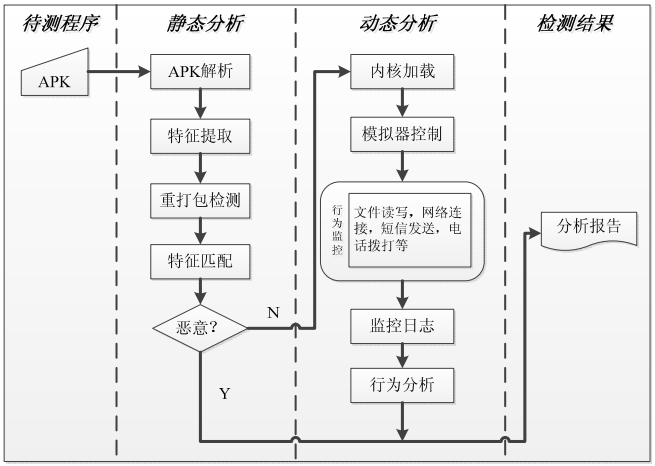

图 2　系统架构

待测 Android 程序的特征信息，并与 Android 恶意代码特征库进行匹配，可以检测已知恶意代码家族的恶意代码样本，同时进行重打包检测。静态分析无法应对代码混淆、加密等情况，所以如果静态分析未检测出恶意行为，则进行动态分析，以弥补静态分析的不足。动态分析主要监控 Android 程序在模拟器中实时运行的行为，并与预定义的恶意行为模式进行匹配，实现检测恶意代码特征库中未知的恶意代码样本。

3　静态分析

3.1　恶意代码静态特征分析

静态分析部分基于特征匹配的方法，本文提取 APK 文件中的权限、API 函数调用序列、组件、资源以及 APK 结构构建特征向量。

（1）权限：由于 Android 严格的权限管理机制，很多与用户体验和数据相关的功能设定了各自的使用权限，因此恶意代码在实现其恶意行为时，需要相应的权限。例如，具有恶意吸费功能的 HippoSMS[7]申请了 SEND_SMS 权限发送短信。然而，很多正常应用程序也会申请短信发送权限，例如通讯类应用程序 Youni 会申请 SEND_SMS、READ_SMS、WRITE_SMS 等敏感权限，因此仅仅根据权限使用情况无法准确判断一个应用程序是否存在恶意行为。

（2）API 函数调用序列：考虑同一个家族的恶意代码，其恶意行为表现在函数调用序列上具有一定的相似性。以 HippoSMS 的两个样本（MD5：A835B82DE9E15330893DDF2DA67A6A49，1A4FB41-B 95C3CBAB05636EE966043C9E）为例，其短信发送行为的函数调用序列为 OnCreate→ sendmsms→ sendTextMessage。本文以敏感权限对应的 API 为搜寻起始点，在 classes.dex 反汇编后生成的 smali 文件中进行函数调用序列的回溯，提取完整的函数调用序列。为了找出敏感权限，本文对 214 个 Android 恶意代码（如表 1 所示）样本进行统计分析，从中提取使用频率最高的前 20 个权限作为高危敏感权限，使用的敏感权限如图 3 所示。

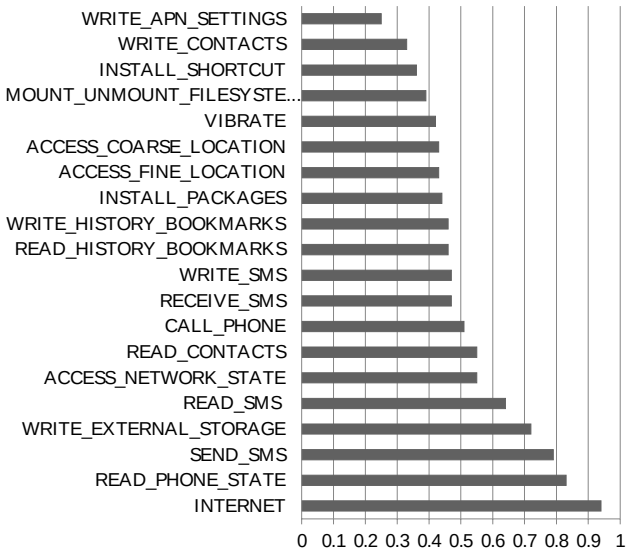

图 3　Android 恶意程序的权限使用率

表 1　Android 恶意代码样本统计

类别	数量	所占比例/%
TrojanSMS	98	45.8
Geinimi	25	11.7
Kmin	40	18.7
Lightdd	6	2.8
PJApps	14	6.5
BaseBridge	5	2.3
Steek	14	6.5
Plankton	6	2.8
其他	6	2.8
合计	214	100

（3）组件：Android 应用程序主要由 4 大组件构成，包括活动（Activity）、广播接收器（Broadcast Receiver）、服务（Service）和内容提供器（Content Provider）。除广播接收器可以在程序中动态注册，其

他组件需要在 AndroidManifest 文件中声明。恶意代码行为的实现需要通过组件的形式表达，如为了达到隐蔽的目的，恶意代码常以服务的形式实现其恶意行为，Android 系统中的服务在后台运行，不与用户进行交互，采用此种方式用户一般很难发现。同一家族的恶意代码，其组件有一定的相似性，以 Geinimi 两个样本（MD5：4E26A200AB149819DCDCF273F5AB171A，543E9D86DD28005342A3313BDC588009）为例，两个样本都包含名为“com.geinimi.custom.GoogleKeyboard”的服务，且内容完全一样，因此 Android 应用程序组件信息在一定程度上可以代表其运行时的行为。

（4）资源：一些 Android 恶意代码的 APK 文件附带资源文件，例如加密后的可执行文件。恶意代码在运行时可以动态解密并加载执行。以 DroidKungfu 的一个样本为例，解压缩后可发现在 asserts 目录下含有一个名为 ratc 的文件，而该文件实际上是经过加密的漏洞利用攻击程序，可以影响 Android2.2 及以下的系统。恶意代码运行时，动态解密并执行该程序，进而可以获得 Root 权限。因此，Android 应用程序使用的资源文件与其自身行为有一定的关系，通常恶意代码会故意混淆其资源文件名，例如一个后缀为 png 的图片文件很可能实际上是一段漏洞利用攻击程序。

（5）APK 结构：由于 APK 文件结构的特殊性，APK 很容易被重新打包并发布。很多恶意代码制作者为了能够有效地传播其恶意代码，会自动化、批量式地将含有恶意功能的代码注入到正常的应用程序中，因此APK文件的结构可以在一定程度上代表同一家族恶意代码之间的关联性。同样以上文的两个 Geinimi 样本为例，两个样本中都含有 com.geinimi 的包结构，事实上，所有恶意功能都由该包下的文件实现。

3.2 重打包检测

在对 Android 恶意代码样本分析过程中，我们发现大部分恶意代码都会以注入到其他正常应用程序中来进行有效地传播。因为 APK 文件可以很容易地被反编译，同时 Google 采取开发者自签名的方式对 APK 文件进行保护，这使得恶意代码制作者只需要使用现有的工具就可自动化、批量式地完成恶意代码注入的过程。恶意代码制作者一般会选取当前比较流行的应用程序作为重打包对象，以免费、破解等方式诱使用户下载，而这种重打包现象一方面会对原程序开发者的利益造成损害，打击开发者的积极性，另一方面容易助长 Android 恶意代码肆虐之势。总之，重打包的存在会导致 Android 生态环境的恶化，不利于 Android 的长期发展。

文献[1]从文件相似性比对出发，使用 Fuzzy Hashing 的方式实现重打包检测，虽然其获得了 5%~13%的检测率，但是其检测效率较低，无法进行快速地检测。本文根据 Android 系统对程序安装的特点，提出通过比对包名和签名的方法来快速实现检测。包名是 Android 应用程序的唯一标识，Android 系统利用包名和签名信息来判断是否允许一个 Android 应用程序升级，只有在包名和签名都相同的情况下才进行升级，而重打包后的应用程序，其包名一般不发生变化，但是签名信息则不同。因此，可以从 Google Play 官方电子市场爬取应用程序，标记为原始未重打包，对待测应用程序，提取其包名和签名信息，并与来自官方电子市场的应用程序进行比对，如果包名相同而签名信息不同，则认为经过了重打包。此方法可以快速地实现重打包检测，易于实时检测。

3.3 静态分析检测部分的设计

静态分析检测部分包括 4 个模块：APK 解析模块对 APK 进行解压缩等相关的预处理，便于后续特征的提取；特征提取模块从解析后的 APK 文件中提取 3.1 小节描述的特征信息，构建特征向量；重打包检测模块使用 3.2 小节的方法进行快速判断检测；特征匹配模块将从待测 Android 应用程序中提取的特征与恶意代码特征库进行匹配，检测是否为恶意代码。

APK 文件的解析，包括对 APK 文件的解压缩、AndroidManifest.xml 文件的解码、classes.dex 文件的反编译以及数字证书的提取。

特征的提取在 APK 文件解析后进行，可以从 AndroidManifest.xml 文件中方便地提取出权限的使用信息，并根据 3.1 小节中列出的 20 个高危敏感权限进行过滤，进一步从 classes.dex 反编译生成的 smali 文件中提取 API 函数调用序列；组件中除了广播接收器可以在程序中动态注册外，其他的组件必须在 AndroidManifest.xml 文件中声明，本文仅提取 AndroidManifest.xml 文件中声明的组件信息；资源文件以及 APK 结构可以从解压缩后的 APK 文件中获取相应的信息。

完成对 APK 的特征提取后，可以进行重打包检测

和特征匹配的工作。本文重打包检测采用 3.2 小节中提出的方法，将来自 Google Play 官方电子市场 APK 的包名和签名信息存入数据库，对待测 APK 文件，获取其包名和数字签名信息，并利用数据库进行快速检测。

特征匹配部分基于 NCD（Normalized Compression Distance）算法[8]，该算法可以计算两个序列之间的相似度。对于给定的两个序列 x 和 y，并且假设相连接后的序列为 S，使用的压缩算法为 Comp，序列压缩后的序列长度表示为 $L(\mathrm{Comp}(S))$，序列的长度可以通过压缩后字符串的字节数或者字符个数等表示。因此，用 NCD 算法计算序列 x 和 y 之间的距离 $d_{\mathrm{NCD}}(x, y)$，表示为

$$d_{\mathrm{NCD}}(x,y)=\frac{L(\mathrm{Comp}(S))-\min\{L(\mathrm{Comp}(x)),L(\mathrm{Comp}(y))\}}{\max\{L(\mathrm{Comp}(x),L(\mathrm{Comp}(y))\}}$$

$d_{\mathrm{NCD}}(x, y)$ 返回值在 0.0（完全相同）到 1.0（完全不同）之间。

4　动态分析

静态分析只能检测出特征库中已有的恶意代码样本，而无法检测未知的恶意代码，同时静态分析很难应对代码混淆、反射、加密等情况，针对静态分析的缺点，本文设计实现了动态分析。动态分析以 Android 模拟器为运行环境，将 APK 文件安装到模拟器中并运行，同时监控 APK 文件运行时的行为，并与恶意行为模式进行匹配，判断是否为恶意代码。

4.1　恶意代码的动态行为模式分析

本文对 Android 恶意代码进行了分析并总结出其动态行为模式，如表 2 所示，具体阐述如下。

表 2　动态行为模式

编号	行为模式名称	行为模式描述
1	关键路径和数据访问	命令执行、隐私数据访问
2	恶意域名访问	C&C 服务器等域名访问
3	恶意吸费	发送吸费短信、拨打吸费电话
4	权限绕过	未授权行为执行

（1）关键路径和数据访问：Android 系统基于 Linux 内核，同样存在一些敏感路径，比如系统可执行程序目录/system/xbin，恶意代码可以调用该目录下的系统程序执行命令。以 Root 漏洞利用恶意代码 GingerMaster 为例，其在恶意行为执行过程中会调用 chmod、mount 等程序执行更改文件权限，挂载文件等命令；短信，通讯录等隐私信息存储于特定的数据库，如短信数据库为 mmsms.db，一些获取个人隐私信息的恶意代码会对该数据库进行访问。关键路径和数据信息如表 3 所示。

表 3　关键路径和数据

编号	关键路径/数据	描述
1	/system/xbin	系统程序目录
2	/system/bin	系统程序目录
3	/data/system/accounts.db	个人帐号信息
4	/data/data/com.android.providers.contacts/databases/contacts2.db	通讯录数据库
5	/data/data/com.android.providers.telephony/database/mmsms.db	短信数据库

（2）恶意域名访问：隐私窃取类的恶意代码会收集用户的个人信息上传至服务器；僵尸木马类的恶意代码会访问 C&C 服务器，获得控制命令。例如 Geinimi 会从 www.widifu.com:8080 获取控制命令，因此可以收集这类恶意域名，并设置成黑名单，作为恶意行为判别的一个衡量因子。

（3）恶意吸费：Lookout 关于 2012 年手机安全状况报告指出，在 Android 恶意代码分类中，恶意吸费类在 2012 年第 2 季度达到了 62%的比例。这类恶意代码在运行过程中会发送吸费短信、拨打吸费电话，对用户的话费造成损失。我们在动态分析过程中记录程序的短信发送、电话拨打行为，如果号码不在移动运营商之列，如 10086、10000 等，则认为具有恶意吸费行为。

（4）权限绕过：如果程序在 AndroidManifest.xml 文件中没有声明某些权限，而在实际运行过程中又执行了需要该权限的行为，则称之为权限绕过。这种情况一般存在于获取了 Root 权限的恶意代码，恶意代码在获取 Root 权限后可以在不需要其他任何权限的情况下执行敏感行为。

4.2　动态分析检测部分的设计

动态分析检测部分主要包括 4 个模块：内核加载模块在 Android 模拟器中加载可对程序行为进行实时

监控的模块；模拟器控制模块对程序在模拟器中运行进行控制；行为监控模块实时监控程序运行时的行为；行为分析模块对监控到的行为进行分析，判断是否存在恶意行为。

Android 系统中 Logcat 可以记录系统运行期间的日志信息，但是并不包括文件读写、短信发送等信息。我们修改了 Android 源码，在需要监控的函数内添加日志输出功能，使得该函数在调用时可以实时监控其参数信息，将修改后的源码重新编译生成内核文件，使用该内核文件加载启动 Android 模拟器。

模拟器控制模块包括模拟器的启动，APK 的安装、运行，以及模拟器的关闭等，如图 4 所示。为了避免 APK 运行时在模拟器中留下痕迹，影响下一次的分析，可以使用快照技术，一方面在每次分析 APK 时，可以保证该 APK 运行于一个纯净的模拟器环境，另一方面可以加速模拟器的启动。为了模拟用户的交互行为，使用 Monkey 产生用户事件流。

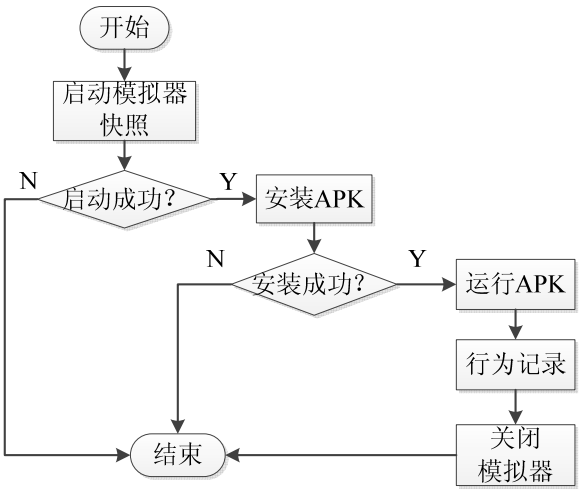

图 4 模拟器控制流程

行为监控模块对 Android 系统产生的日志信息进行过滤和记录；行为分析模块在行为监控模块产生的数据的基础上，使用 4.1 小节提出的恶意行为模式进行匹配分析，判断 Android 应用程序在运行过程中是否有恶意行为存在。

5 实验结果与分析

使用表 1 中恶意代码样本和 200 个正常样本对系统进行测试。表 4 是系统对恶意代码样本的检测结果，系统从 214 个样本中共检测出 194 个，漏报率为 9.3%；

表 4 系统对恶意代码样本的检测结果

类别	数量	检测数量	检测率/%
TrojanSMS	98	95	96.9
Geinimi	25	20	80.0
Kmin	40	35	87.5
Lightdd	6	4	66.7
PJApps	14	12	85.7
BaseBridge	5	4	80.0
Steek	14	14	100
Plankton	6	5	83.3
其他	6	5	83.3
合计	214	194	90.7

在对 200 个正常样本分析中，系统将 3 个样本误判为恶意代码，误报率为 1.5%。从上述结果可知，本文设计的 Android 恶意代码检测系统具有较好的效果。

结果以MD5 为5B087AEF1247591B1EFE78032476BDE7 的样本为例，ESET NOD32 检测为 FakePlayer 恶意代码，由于没有该恶意代码样本的特征，因此本系统静态检测部分没有检测出该恶意代码，但是在动态检测部分，成功监控到该恶意代码向 7132 发送了“846978”、“845784”、“846996”和“844858”4 条短信。

使用本系统对来自第 3 方电子市场的应用程序进行检测，从 Google Play 和安智市场收集 APK 文件，如表 5 所示。将 Google Play 的 APK 文件作为重打包检测的比对标准，对来自安智市场的 APK 文件进行重打包检测，发现 36 个文件经过重打包，占所有检测文件的 1.0%。以 MD5 值为 56C9E4B48CB043732AC45-E6F3013AD7E 的 APK 文件为例，该 APK 文件包名为“com.xxstudio.fallingblock”，与之对应的 Google Play 上原始 APK 文件 MD5 值为 69E12C585E42E41E108E-50B4D711B45E。包组织结构对比如图 5 所示。重打包的 APK 中增加了“cn.domob.android”包，该包是多盟（智能手机广告平台）为开发者提供的广告投放 API 包。该重打包是为了在正常 APK 文件中插入广告来获取盈利。

表 5 Google Play 和安智市场数据源

市场名称	数量	收集日期
Google Play	15917	2012.3~2012.5
安智市场	3514	2012.3

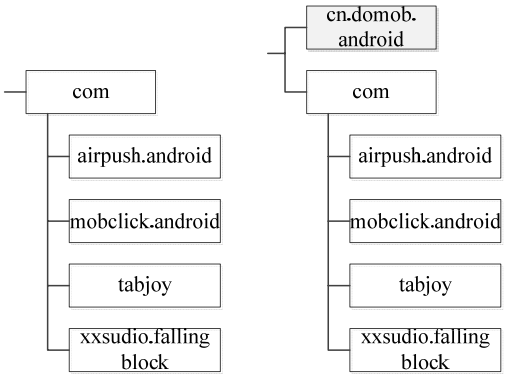

图 5　重打包前（左）后（右）包组织结构对比

使用静态分析方法成功检测出了 DroidKungfu.A、GoldDream.A、Plankton 等恶意代码，如表 6 所示。使用动态分析方法成功检测出 GingerMaster、Lightdd 等恶意代码，如表 7 所示。

表 6　静态分析检测结果

类别	数量
Plankton	6
TrojanSMS	1
Steek	2
TGLoader	1
GoldDream.A	3
DroidKungfu.A	2
合计	15

表 7　动态分析检测结果

类别	数量
Gappusin.A	7
Adware.Airpush.A	5
Lightdd	1
FakeUpdates.B	1
GingerMaster	1
合计	15

7　结论与展望

本文综合静态分析和动态分析技术，全方位分析检测 Android 程序，实现了 Android 恶意代码分析和检测系统。通过对第 3 方市场 Android 程序的分析检测以及对系统误报和漏报的测试表明，本系统检测效果良好。同时，本系统已在互联网上发布，提供免费的分析检测服务。在动态分析过程中，程序的部分行为需要相应的触发机制才能运行，因此如何触发这些行为是下一步行为监控技术的研究重点。

参考文献：

[1] ZHOU W, ZHOU Y, JIANG X, et al. Detecting repackaged smartphone applications in third-party Android marketplaces [C]// Proceedings of the Second ACM Conference on Data and Application Security and Privacy. New York, USA: ACM, 2012: 317-326.

[2] BORJA S, IGOR S, CARLOS L, et al. PUMA: permission usage to detect malware in Android[C]//International Joint Conference CISIS' 12-ICEUTE' 12-SOCO' 12 Special Sessions. Berlin, Germany: Springer, 2012: 289-298.

[3] BURGUERA I, ZURUTUZA U, NADJM-TEHRANI S. Crowdroid: behavior-based malware detection system for Android[C]//Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. New York, USA: ACM, 2011: 15-26.

[4] ENCK W, GILBERT P, CHUN B G, et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones[C]//Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation. Berkeley, CA, USA: USENIX, 2010: 1-6.

[5] ENCK W, ONGTANG M, MCDANIEL P. Understanding Android security[J]. IEEE Security & Privacy, 2009, 7(1):50–57.

[6] POCATILU P. Android applications security[J]. Informatica Economică, 2011, 15(3): 163-171.

[7] JIANG X. Security alert: new Android malware- HippoSMS-found in alternative Androidmarkets[EB/OL].[2012-10-07]. http://www.csc.ncsu.edu/faculty/ jiang/HippoSMS/.

[8] CILIBRASI R, VITÁNYI P M B. Clustering by compression [J]. IEEE Transactions on Information Theory, 2005, 51(4): 1523-1545.

（编辑　荆树蓉）