doi :10.3969/j.issn.1671-1122.2015.09.015

Android 操作系统恶意软件检测技术研究

李汶洋
（中国电信集团系统集成有限责任公司宁夏分公司，宁夏银川 750001）

摘　要：随着 Android 系统智能手机使用人数的上升，Android 系统平台上恶意软件的数量也在大幅度增加。如何对 Android 系统恶意软件进行检测已经成为一个值得研究的问题。文章针对目前国内外关于 Android 系统恶意软件检测技术的研究进行总结，对比了静态分析与动态分析，同时列举了目前国内外的一些研究成果。

关键词：Android 系统；恶意软件；检测；静态分析；动态分析
中图分类号：TP309　　**文献标识码：**A　　**文章编号：**1671-1122（2015）09-0062-04

中文引用格式：李汶洋 .Android 操作系统恶意软件检测技术研究 [J]. 信息网络安全，2015，（9）：62-65.
英文引用格式：LI W Y. Research on Android Malware Detection Technology[J].Netinfo Security, 2015, (9) : 62-65.

Research on Android Malware Detection Technology

LI Wen-yang
(*China Telecom System Integration Co.Ltd., Ningxia Branch, Yinchuan Ningxia* 750001, *China*)

Abstract: With the popularity of Android smart phones, the number of malware on Android platform has increased greatly. How to realize the malware detection in Android system has become a problem worthy of study. This paper summarizes the current research on Android malware detection, compares the static analysis and the dynamic analysis, and lists some related domestic and foreign research results.

Key words: Android system; malware; detection; static analysis; dynamic analysis

0 引言

当前 Android 系统已经成为世界上应用最为广泛的手机操作系统，Android 系统手机受到广大用户的欢迎。Android 系统平台上的应用软件涉及个人通讯、数据存储、多媒体、娱乐等各个方面。随着 Android 系统移动设备的广泛使用，Android 系统平台上的应用软件数量大幅度增加，恶意软件数量也急剧上升。Android 系统成为许多恶意人员的攻击目标。如何有效、准确地检测 Android 系统平台上的恶意软件，已经成为当前安全领域中的一个热点课题[1]。

通过人工分析来识别恶意软件，耗时费力且容易出错；通过自动化的方式对恶意软件进行识别，速度较快且能够分析大量软件样本。目前采用自动化方式检测 Android 系统恶意软件的技术主要有两种：静态分析和动态分析。

本文对静态分析与动态分析这两种技术进行介绍，概括当前国内外对 Android 系统恶意软件识别检测方面的研究，同时列举了一些研究成果。

1 相关背景

1.1 Android系统程序

Android 系统采用了分层架构，如图 1 所示。Android 系统分为 4 层，从高到低分别是应用程序层、应用程序框架层、系统库和运行时环境以及 Linux 内核层。

收稿日期：2015-07-15
作者简介：李汶洋（1988-），男，辽宁，本科，主要研究方向：信息安全、移动通信安全、等级保护等。
通讯作者：李汶洋 18995118631@189.cn

图1 Android系统架构图

Android 系统应用程序包文件（APK）是一种 Android 操作系统上的应用程序安装文件格式。一个 APK 文件包含被编译的代码文件（.dex 文件）、文件资源（resources）、资产（assets）、证书（certificates）和清单文件（manifest file）[2]。APK 文件中，classes.dex 是 classes 文件通过 DEX 编译后的文件格式，也是在 Dalvik 虚拟机上运行的主要代码部分。AndroidManifest.xml 是一个传统的 Android 清单文件，用于描述该应用程序的名字、版本号、所需权限、注册的服务、链接的其他应用程序。

1.2 恶意行为

具有什么样行为的 Android 系统软件被判断成恶意软件，从目前看来，恶意软件的恶意行为主要有以下 4 种：

1）重新打包和更新攻击

恶意软件通常伪装成正常软件，其中一种主要方式就是重新打包。即把作为载体的正常软件反编译后，加入恶意部分的代码作为负载，重新打包成 APK 上传到应用商店。由于包含的恶意负载容易被静态分析识别出来，近期有些恶意软件在安装包里并不直接加入恶意负载，而是封装了一个在手机上安装完之后自动获取恶意负载的组件，当网络连接可用时，该组件会偷偷下载恶意负载。

2）路过式下载攻击

用户在访问了某个网站、浏览了恶意邮件或是误点了某个链接后，可能会在不经意间下载恶意软件。这种攻击尤其适合屏幕大小有限制的手机，因为浏览器会隐藏地址栏从而使这种攻击不易被发现。

3）远程控制

这种恶意软件远程控制感染的手机，这些手机大多通过 HTTP 网络通信来接收从攻击者服务器发来的命令。

4）信息收集

恶意软件会收集存储在手机上的有价值的个人信息，如短信、邮件、手机号、账户信息等，将收集到的这些信息偷偷发送给攻击者。

1.3 机器学习

机器学习算法[3]是一种对数据自动分析获得规律，利用规律对未知数据进行预测的算法。机器学习算法存在 4 种完全不同的学习方式：分类学习[4]、关联学习、聚类和数值预测。机器学习广泛应用于数据挖掘、计算机、自然语言处理、生物特征识别等领域。

2 研究状况

目前 Android 系统恶意软件检测技术主要分为两大类：静态分析和动态分析。不管是静态分析还是动态分析，都是通过分析一些恶意软件样本，提取、收集恶意软件的恶意行为的代表性特征，采用机器学习方法或其他方式，实现对恶意软件的识别检测。这里对静态分析与动态分析这两种方式进行介绍和比较，并列举当前国内外的一些研究成果。

2.1 静态分析

静态分析指在没有实际运行应用程序的情况下，通过分析程序控制流、数据流等信息来分析程序行为。静态分析通常用于找出应用程序中的漏洞，以及在不必运行应用软件的情况下分析恶意行为。

静态分析通常的流程是：首先，对应用程序进行反编译，从而获得 manifest 文件及 java 代码，或者利用现有的一些反编译工具包如 dexlib2.jar、AXMLPrinter2.jar 来读取应用程序代码信息及 manifest 文件信息；然后，对 manifest 文件进行解析，获得权限、活动、服务等信息；最后，分析代码，构造函数调用图（CallGraphs）、程序控制流图（CFG）和程序依赖图（PDG）等，对程序的控制流、数据流及函数调用等进行分析[5]。

在目前的研究中[6-9]，通过静态分析识别 Android 系统恶意软件占了很大一部分，这些研究中提取出的特征主要有以下几种：权限、API 调用、控制流、数据流、函数调用图、

硬件组件、Intent 信息传递。这些都是通过静态分析方式提取出来的、被认为比较能够反映 Android 系统中恶意软件行为的常见特征。对这些特征进行处理筛选后，研究者们大多采用机器学习的方法构造模型，实现对 Android 系统恶意软件的检测。

2.2 动态分析

在应用程序运行时分析它的行为，这种方式称为动态分析。静态分析虽然开销小，但分析的毕竟是没有运行起来的程序代码，无法对程序运行后的实时行为进行监控。动态分析能够跟踪一个应用软件真正做了什么，通过人工或自动分析记录文件来达到分析程序行为的目的。

动态分析通常的流程是：首先，在一个虚拟运行环境即“沙箱”中运行 Android 系统应用程序，并记录程序的所有行为；然后，对收集到的行为进行过滤，去除那些不敏感、与恶意行为无关的环境，留下有用信息；最后，对获得的信息进行分析处理，判断并挑选出软件的恶意行为，如泄露隐私数据、私下发送短信等 [10]。

动态分析也是目前实施 Android 系统恶意软件检测的一种主流方式。在使用动态分析的研究中 [11-15]，主要监控了以下几个方面：系统调用、网络、SMS（即短信）、CPU、电量、进程、内存、虚拟内存、实时通信量、过滤事件。以上这些因素能够反映出被监控的程序执行时的行为，能够与一些敏感行为挂钩，从而实现对恶意软件的识别 [16]。

2.3 静态分析与动态分析比较

静态分析开销比较小、速度快、效率高，但是具有一定的局限性。静态分析大多是基于人工制作的检测模式，攻击者可以使用多种混淆技术来逃脱静态分析的检测。这让静态分析检测多形态恶意软件的能力受到很大限制，经常无法识别新出现的恶意软件。

相比较而言，动态分析在检测应用软件的恶意行为上更为有效 [17]。由于监控的是程序的实际行为，所以不存在受到混淆技术干扰的问题。然而动态分析系统开销大，在环境部署和人工调查方面代价很高。此外，一些资源上的约束如 CPU、电量、内存等，使得很难直接把用动态分析实现的检测技术搬到 Android 系统手机环境上。动态分析的实现上也存在一些问题。恶意软件并不是随时随地都在执行恶意行为，需要有一定的触发条件或者出现“受害”软件才会执行。如何模拟出让软件中的恶意参数活跃起来的合适条件成为动态分析的一个主要难点。再有，动态分析观察每个恶意软件出现的恶意行为所需要的时间阶段不明确，而全时段监控代价很大。

总的说来，静态分析与动态分析各有利弊，也能够弥补对方的不足。因此，有的研究者采用了静态分析与动态分析相结合的方式实现对 Android 系统恶意软件的检测 [18,19]。

2.4 研究成果举例

2.4.1 Drebin

Drebin[20] 是 Daniel 等人提出的一种检测 Android 系统恶意软件的轻量级方法，它可以自动推断出检测模式，并能够在手机上直接识别恶意软件。Drebin 流程如图 2 所示。

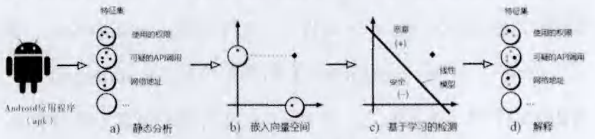

图2 Drebin流程

Drebin 首先从应用程序中收集尽可能多的特征来进行静态分析。它收集的特征集合主要有 8 组，4 组来自 manifest 文件的特征集合：硬件组件、请求的权限、APP 组件和过滤后的 intents；4 组来自反汇编代码的特征集合：受限制的 API 调用、使用的权限、可疑的 API 调用和网络地址。接着将这些特征组织成字符串集合，嵌入向量空间，几何分析这些特征的模式与结合方式。最后使用机器学习的方式，实现用嵌入的特征集合识别 Android 系统恶意软件，并对检测结果进行解释 [21]。

2.4.2 Crowdroid

Crowdroid[22] 是 Burguera 等人在早期研究的基础上利用动态分析实现的 Android 系统恶意软件检测工具。其检测框架和流程如图 3 和图 4 所示。

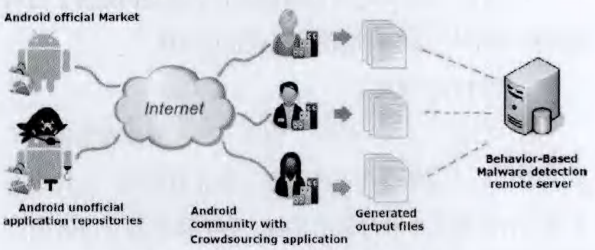

图3 Crowdroid恶意软件检测框架

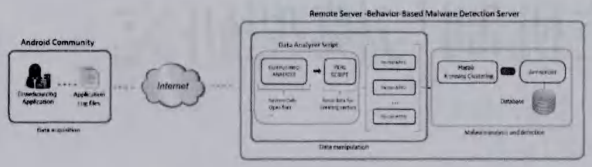

图4 Crowdroid恶意软件检测流程

由图3可知，Crowdroid应用软件是通过一个中心服务器从多个真实用户处收集系统跟踪信息，动态分析程序行为。在每个Crowdroid应用软件客户端监控Linux系统调用，经过预处理后，客户端将收集到的信息发送给中心服务器，使用划分聚类算法即2-means算法，对每个数据集进行聚类，分析数据并创建一个系统调用向量。最后，基于用户行为进行恶意软件动态分析。

2.4.3 Droid-Sec

为了更系统地描述Android系统应用软件(包括恶意软件和正常软件)，Zhenlong Yuan等人采用静态分析和动态分析相结合的方式实现了Droid-Sec[23]。该方法从Android系统程序样本中提取了202个特征用于深度学习，实现对Android系统恶意软件的检测。这用于学习的202个特征中，既包含权限、敏感API等通过静态分析得到的特征，也包含了一些通过动态分析获得的行为特征。

3 结束语

本文对Android系统恶意软件检测技术方面的研究进行了概括和总结，详细介绍了静态分析与动态分析这两种技术，比较了它们的优势与不足，并列举了目前国内外的一些研究成果。静态分析与动态分析各有短板，在未来的研究工作中，还会有更令人欣喜的分析检测技术出现。（责编 马珂）

参考文献：

[1] 赵洋，胡龙，熊虎，等．基于沙盒的Android恶意软件动态分析方案[J]. 信息网络安全，2014，(12)：21-26.

[2] Wu D, Mao C, Wei T, et al. DroidMat: Android Malware Detection through Manifest and API Calls Tracing[C] //Conference on Information Security. IEEE, 2012:62 - 69.

[3] Sahs J, Khan L. A Machine Learning Approach to Android Malware Detection[C] //European Intelligence and Security Informatics Conference. IEEE, 2012:141 - 147.

[4] Grampurohit V, Kumar V, Rawat S, et al. Category Based Malware Detection for Android[J]. Communications in Computer & Information Science, 2014，(2)：35-42.

[5] 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.

[6] Talha K A, Alper D I, Aydin C. APK Auditor: Permission-based Android malware detection system[J]. Digital Investigation, 2015，3(1):1-14.

[7] Sheen S, Anitha R, Natarajan V. Android based malware detection using a Multifeature Collaborative Decision Fusion approach[J]. Neurocomputing, 2015,6(1): 151.

[8] Aafer Y, Du W, Yin H. DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android[J]. Lecture Notes of the Institute for Computer Sciences Social Informatics & Telecommunications Engineering, 2013，(2)：86-103.

[9] Grace M, Zhou Y, Zhang Q, et al. Riskranker: scalable and accurate zero-day android malware detection[C]//Proceedings of the 10th international conference on Mobile systems, applications, and services. ACM, 2012: 281-294.

[10] 高岳，胡爱群．基于权限分析的Android隐私数据泄露动态检测方法[J]. 信息网络安全，2014，(2)：27-31.

[11] Egele M, Scholte T, Kirda E, et al. A Survey on Automated Dynamic Malware Analysis Techniques and Tools[J]. Acm Computing Surveys, 2012, 44(2):169-192.

[12] Rieck K, Trinius P, Willems C, et al. Automatic analysis of malware behavior using machine learning[J]. Journal of Computer Security, 2011, 19(4)：30-39.

[13] Jin R, Wang B. Malware Detection for Mobile Devices Using Software-Defined Networking[R].Research & Educational Experiment Workshop. IEEE, 2013:81 - 88.

[14] Asaf Shabtai, Uri Kanonov, Yuval Elovici, et al. "Andromaly"：a behavioral malware detection framework for android devices[J]. Journal of Intelligent Information Systems, 2011, 38(1):161-190.

[15] Peng G, Shao Y, Wang T, et al. Research on Android Malware Detection and Interception Based on Behavior Monitoring[J]. Wuhan University Journal of Natural Sciences, 2012, 17(5):421-427.

[16] 刘智伟，孙其博．基于权限管理的Android应用行为检测[J]. 信息网络安全，2014，(6)：72-77.

[17] Min L X, Cao Q H. Runtime-based Behavior Dynamic Analysis System for Android Malware Detection[J]. Information Technology Applications in Industry Computer Engineering & Materials Science, 2013，(6)：25-41.

[18] Yuan Z, Lu Y, Wang Z, et al. Droid-Sec: deep learning in android malware detection[C]//Proceedings of the 2014 ACM conference on SIGCOMM. ACM, 2014: 371-372.

[19] 包佳敏，胡爱群．Android系统文件监听技术的研究[J]. 信息网络安全，2014，(3)：46-51.

[20] Arp D, Spreitzenbarth M, H ü bner M, et al. Drebin: Effective and explainable detection of android malware in your pocket[C]//Proc. of NDSS. 2014.

[21] Gascon H, Yamaguchi F, Arp D, et al. Structural detection of android malware using embedded call graphs[C]//Proceedings of the 2013 ACM workshop on Artificial intelligence and security. ACM, 2013: 45-54.

[22] Burguera I, Zurutuza U, Nadjm-tehrani S. Crowdroid: Behavior-Based Malware Detection System for Android[C]//Proceedings of the Acm Conference on Computer & Communications Security, 2011:15-26.

[23] Enck W, Cox L P, Jung J, et al. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones[C]// Proceedings of Usenix Conference on Operating Systems Design & Implementation Osdi, 2010, 57(3):99-106.