基于动态分析的 Android 应用程序安全研究

宁 卓 胡 婷 孙知信
(南京邮电大学物联网学院 南京 210003)

摘 要 Android 操作系统由于其功能强大、开发方便,短短几年就已经成为全球第一份额的智能手机操作系统,同时也成为了恶意攻击的首选目标。首先简单介绍 Android 恶意软件及其检测方法;然后对 Android 安全中比较准确的动态分析技术进行综述,详细介绍各种动态分析技术的工作原理、技术方案以及技术的性能水平和检测效果,分析并比较它们各自的优缺点;最后,提出几个值得深入研究的技术方向。

关键词 Android,恶意软件,动态分析
中图法分类号 TP319 **文献标识码** A

Security Survey on Android Application Based on Dynamic Analysis

NING Zhuo HU Ting SUN Zhi-xin
(Department of the Internet of Things, Nangjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract Because Android operating system is powerful and it is easy to develop, it has not only become the world's first share of the smartphone in the past few years, but also become a prime target for malicious attacks. In this paper, we briefly introduced the Android malware and its detection methods firstly. Then we summarized and analyzed some latest and accurate dynamic analysis techniques in Android security and introduced the technology works, technical solutions, technology performance levels and test results from multidimensional perspectives. Finally, we presented a few directions that are worthy of further research.

Keywords Android, Malware, Dynamic analysis

1 引言

Android 是一种基于 Linux 的自由及开放源代码的操作系统,主要应用于移动设备。Android 应用程序属于 Android 系统总体架构的应用层,除了系统内置的基本应用,包括桌面、邮件、电话、短信等,更为用户常用和交互的是由 Java 语言和本地代码编写的第三方应用程序,其对设备的功能进行新增、扩展和优化。

近年来,智能手机销量突飞猛进。据权威分析机构 Strategy Analytics 的统计[1],在 2015 年第四季度全球智能手机的销售达到 1.168 亿部,较上一季度增长 24.3%。其中,Android 的销售量占到全球份额的 80%以上,具有绝对优势。另一方面,Android 系统的开放性也受到应用程序开发者的青睐,用户使用移动设备花费的时间中 87%消耗在应用程序上[2]。巨大的市场份额也吸引了恶意攻击者的注意。图 1 是 McAfee 实验室 2016 年发布的对恶意软件的统计数据图[3]。如图中数据所示,恶意软件的数量在过去的几年中激增,同时其复杂性也在不断提高。虽然发布安全补丁可以缓解大多数软件的恶意行为,但是恶意软件编写者会继续寻找新的漏洞并发布新的恶意软件或其变种,因此,需要发布新的补丁来修复新发现的漏洞,恶意软件的发布和修补进入恶性循环。

解决 Android 安全问题迫在眉睫,学术研究人员和商业反恶意软件公司已经意识到,随着代码混淆和加密等技术的出现,传统的基于签名的分析方法和静态分析方法已经不能满足需求,动态分析方法应运而生。为了更全面地分析其恶意性,动静态结合的方法成为应用程序安全检测的主流方法。

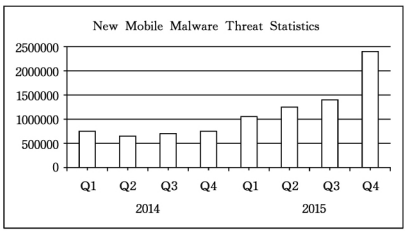

图 1 2014－2015 年恶意程序走势图

本文第 2 节简单介绍了 Android 恶意软件及其检测方法;第 3 节对 Android 安全检测中比较准确的动态分析技术进行综述,详细介绍了各种动态分析技术的工作原理、技术方案以及技术的性能水平和检测效果,分析比较它们各自的优缺点;最后提出了几个值得深入研究的技术方向。

2 Android 恶意软件及其检测方法

尽管 Android 系统精心设计了安全机制,但 Android 的开放性以及其性能和安全方面的折中,都有可能被开发者利

本文受国家自然科学基金(61170276,61373135)资助。
宁 卓(1975－),女,讲师,主要研究方向为网络安全、物联网应用等,E-mail:ningz@njupt.edu.cn;胡 婷(1992－),女,硕士生,主要研究方向为物联网应用技术;孙知信(1964－),男,教授,博士生导师,主要研究方向为计算机网络与安全、多媒体通信、移动互联网。

用从而形成恶意软件。恶意软件是指携带恶意代码，对系统和其他应用的正常运行和关键数据等产生安全威胁的应用。恶意应用通常在热门应用中插入恶意攻击代码从而吸引用户安装，并利用安全管理混乱的第三方应用市场进行发布和传播。同时，随着设备性能的增强和研究的深入，恶意应用的隐私窃取能力也日益提高。

智能手机的安全不只局限于恶意软件，应用程序使用不安全的方式存储或传输个人或企业的敏感数据也是当前最值得关注的。即使是从官方应用商店下载的合法的没有恶意功能的应用程序也存在高风险的安全问题。应用程序的恶意行为主要存在两种情况：1）Android 应用程序本身就是由恶意开发者开发用以实施恶意行为的程序，即病毒程序；2）Android 应用程序在设计上存在缺陷，可以被攻击者利用，对 Android 终端实施恶意行为。实际中后一种情况更为常见，危害性也常常不为人重视。

传统的分析恶意应用程序的方法为静态分析方法，静态分析在安装前检测 Android 应用程序中的安全漏洞，通常的分析步骤是：下载 Android 系统的应用程序安装包（Android Application Package File，APK），通过如 Ded[4] 和 Dex2Jar[5] 之类的工具进行反编译，将 APK 文件反编译为 Java 源文件或 JAR 文件，然后对其进行源代码级的解析。目前，静态源码分析技术存在的问题是：1）针对 Android 源码的反编译技术还不完善，汇编指令种类纷杂，缺少源码级别的代码结构信息和类型信息，导致反编译后的代码可读性差；2）Android 应用普遍采用代码混淆和软件加壳技术对源代码进行安全加固，增加了反编译的难度和反编译代码的可读性。此外，由于应用程序代码设计与程序运行时实际行为存在一致性问题，对程序进行静态分析仅能判断应用程序可能存在的安全风险，并不能确定存在真正的恶意行为。因此在 Android 应用软件安全检测中，静态分析技术只能作为辅助。

动态分析技术是指在严格控制的环境中（如沙盒、虚拟机、物理隔绝的主机等）执行软件的安装、运行等操作，借助受控环境监测并记录该对象的行为，如 API 调用、网络通信、文件读写、进程操作等。通过对实际行为的分析检测软件是否具有窃密隐私、恶意吸费、系统破坏、非法内容传播等恶意行为。动态分析技术能实时地监控待测应用运行过程中的各种行为并记录相关的操作和数据，与静态代码分析相比，其检测结果更加真实可信，对 Android 应用程序安全检测也具有很大的研究价值。

3 动态分析技术

软件行为分析技术是安全检测中十分重要的组成部分，对软件行为进行分析不仅可以检测出恶意的软件行为，还可以发现软件行为的实际后果是否会造成漏洞，通过建立软件行为模型可以快速筛选出恶意软件。软件动态行为分析技术主要由两个部分组成：1）对软件运行时实际行为的获取；2）对所捕获到的行为特征进行分析。目前，软件行为的获取分析主要从主动、被动这两个角度入手，即内部行为的主动发掘方式与外部行为的被动监控方式，一般通过动态监控软件实现[6]。

3.1 研究现状

尽管 Android 已经拥有了较好的安全机制来保证系统安全，但是在巨大商业利益的激励下，无数攻击者针对系统及软件漏洞，对 Android 用户展开了各种形式的攻击。然而，Android 应用的迅猛发展和安全问题的日益突出与反制措施极不协调。对 Android 应用程序安全的关键技术进行研究具有前沿性和必要性。众多学者在这一领域做出了贡献，取得了一系列研究成果。我们调查和研究了一些具有前景的检测方法，如图 2 所示。

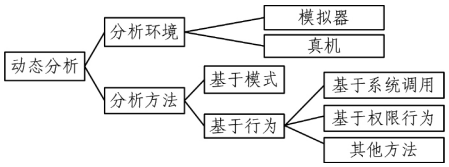

图 2 动态分析分类图

3.1.1 分析环境

多数动态检测方法是在模拟器或沙盒中进行的，这是一种可控环境，可与用户的计算机隔离，代码在被监控的环境中的操作不会对用户计算机有任何影响。

AASandbox[7] 提出了一种新型的动静态结合的方法，即提供快速预检查的静态分析和在完全隔离的环境沙盒中执行动态分析。沙箱放置在内核空间中，可以自动拦截系统调用并记录应用程序在系统级别的行为。整个沙盒和检测算法配置在云服务器上，为 Android 市场上的可疑软件提供快速的分布式的检测，提高了恶意软件的检测效率。TaintDroid[8] 创建于 Dalvik 虚拟机之上，是一种动态污点检测技术，敏感数据被标记后，在虚拟机中触发相应行为，并记录在日志中。DroidBox[9] 是建立在 TaintDroid 基础之上的动态分析工具，污染分析和 API 监控等分析过程都是在虚拟环境下进行的。DroidScope[10] 建立在 QEMU 模拟器上，利用大量 API 来建立恶意软件分析原型系统，与其他动态分析技术相比，该方法保留了模拟器、操作系统监控以及 Dalvik 语义。因此，即使是 Android 内核的权限提升攻击也可以被检测出来。

虽然模拟器或沙盒可以和用户计算机隔离，但有些应用程序通过识别虚拟环境的特征，从而在动态执行的时候隐藏恶意行为。针对这种情况，现有的研究有两种解决方案：第一种是在实际设备中实施动态分析，Paranoid Android[11] 通过在远程服务器上克隆智能手机的状态，建立了一种脱离设备的恶意软件检测方法。Andrubis[12] 是一个基于网页的恶意软件分析平台，用户可以通过网页提交待检测的 App，经过远程服务器发送到真实的手机设备上进行分析，Andrubis 会以网页的形式返回详细的静态和动态分析报告。Apps Playground[13] 和 APIMonitor[14] 也是在实际的智能手机上进行动态分析。Drozer[15] 是 MWR Labs 开发的一款交互式的 Android 安全测试框架，可同时在设备或者模拟器上安装代理程序，在服务器端将用户输入的命令发送到 Android 设备上的代理程序执行。第二种是改进后的具有抗反分析技术的模拟器或沙盒，DroidAnalyst[16] 定义通过修改 Android 虚拟设备内的静态参数模拟真实的 Android 环境为抗反分析技术，这种修改方式可以成功揭示应用程序隐藏的恶意行为。该系统提出的沙盒技术可以监控文件操作、应用下载、可疑的安装、加密的字符串、收费短信以及语音通话等行为。

3.1.2 分析方法

动态分析方法可以分为基于模式分析和基于行为分析两类。

基于模式分析：恶意应用可能会通过有限的硬件资源引起拒绝服务(Denial of Service，Dos)攻击，从 Android 子系统中搜集良性软件和恶意软件的 CPU 使用率、内存使用率、网络流量模式、电池的使用和系统调用等参数，利用机器学习方法自动区分异常行为。基于机器学习的检测恶意软件技术主要通过学习恶意软件和正常程序的差异性发现有关的识别模式，并利用这些模式进行相似性分类以发现含有类似模式的恶意软件。

AASandbox[17]、Andrubis 以及 DroidAnalyst 等方案实现对软件的恶意行为模式进行分析，提取出各种恶意软件特征行为，并建立恶意软件特征行为库，为提取的各种恶意软件特征行为分配权重，建立恶意特征行为与权重之间的映射关系，并与已知的恶意行为模式进行匹配，以此判断目标文件是否具备恶意性。另外，Andrubis 还可以对 App 的恶意行为划分 0－10 的等级，其中 0 为良性，10 的恶意等级最高。实验表明，该系统对零日(zero day)恶意行为的检测非常有效。Shabtai 等人提出了一种基于机器学习的轻量级的 Android 恶意软件检测系统 Andromaly[18]，该系统实时地监控系统的各种度量值，如 CPU 使用率、网络传输的数据量、进程的数量和电池的使用率等。该系统有 4 个主要的组件：1)特征提取器，通过与 Android 内核层和应用程序框架层进行通信来收集特征值；2)特征管理器通过定期触发特征提取器来收集特征值，同时还可以对原始特征数据进行预处理；3)处理器，它是一个分析和处理装置，分析接收到的特征向量进行威胁评估，并传递给威胁加权单元(TWU)，处理器是基于规则的分类器和基于机器学习算法的异常检测器；4)TWU 把收到的所有处理器的分析结果进行集成推导出最终的检测结果。但文献[18]用手写的恶意软件进行实验，缺乏对真正恶意软件的测试。与以上方案不同的是，Crowdroid 利用 2 均值划分聚簇算法来分析远程服务器端处理的特征向量，判断待检测的应用程序为良性软件还是恶意软件，同时，生成一个应用程序报告并存储在远程服务器的数据库中[19]。该算法简单、快速，对处理大数据集是相对可伸缩和高效率的。

然而传统的基于模式分析的方法不能探测到标记扩散或侵染其他用户的恶意软件，包括引起内部资料外泄或从恶意源接收指令的隐藏恶意软件，因此基于行为分析的恶意软件检测往往能产生很好的效果。

基于行为分析：让恶意软件在模拟器或 Android 设备中模拟运行程序，通过监控或拦截的方式分析程序运行的行为。

系统调用可以捕获应用程序与底层系统交互的最本质的特征行为，因此可以利用系统调用来描述软件行为，利用系统 hook 技术监视恶意代码执行过程中的系统调用和 API 使用状态来分析恶意代码的功能。

TaintDroid 系统提出了一种动态污点检测技术：对敏感数据进行污点标记，当这些数据通过程序变量、文件和进程间通信等途径扩散时进行跟踪审查。如果污点数据在一个泄漏点(如网络接口)离开系统，TaintDroid 就在日志中记录数据标记、传输数据的应用程序和数据流向目的地，实现对多种敏感数据泄漏源点的追踪。Droidbox 建立在 TaintDroid 基础之上，通过修改 Android 框架层对 API 调用进行分析。另外，DroidBox 在 TaintDroid 基础上重新定义了污点标签的类型，并添加了网络监控、文件读写监控、加密记录、日志分析等功能。然而 TaintDroid 创建于 Dalvik 虚拟机之上，不能监控本地代码的恶意行为，同时，TaintDroid 只考虑数据流，未对控制流(程序间交互、组件间交互、进程间交互)进行分析。Droidbox 继承了 TaintDroid 的缺点，只能监视 Android 框架内执行的任务，现有的系统检测不出来是否是本地代码泄漏了敏感数据。

为此，Lantz 和杨坤设计了 APIMonitor 进行改进[20]，通过反编译技术在 APK 包中插入监控代码(监控 API 调用)，重新打包 APK，然后在 Android 设备上运行，应用程序中的 API 调用存储在日志中。但是本方案中对应的 API 都要进行配置，工作量庞大，同时对于内部代码有自校验或签名校验的 APK 行不通。Crowdroid 利用 Strace 工具来收集系统调用的细节，并向远程服务器发送记录行为数据(即系统调用的细节)的日志文件，由远程服务器进行分析。日志文件中记录基本的设备信息、安装的应用程序以及其他行为数据。SmartDroid[21]通过修改源码的 framework 框架层在敏感 APIs中插入输出日志，便于监控敏感 API 是否被调用，能够自动并且高效地检测出基于本地代码暴露出的敏感行为。Apps Playground 综合使用基于 TaintDroid 的污点跟踪技术、内核级别的系统调用监控等多种技术分析具有恶意行为的应用程序，可以检测出以合法目的泄漏隐私信息的灰色软件。为了处理 Android 中 root 权限的攻击，Apps Playground 用简洁的签名来描述涉及系统调用和内核数据结构的漏洞，这些签名可以集成在动态分析中。

与现有的基于系统调用的方案不同，CopperDroid[22]精确地把 Android 的 API 调用进行逆向操作，通过观察系统调用信息来捕捉 Android 应用程序的核心行为特征，提取出了 Android 级别的语义信息。但是，这个方案仅能识别 Android 系统与应用程序之间的交互，不能构建出有效的、多粒度的应用程序内部的行为分析。而且，CopperDroid 不能过滤不相关的交互行为，分析这些行为会加重系统开销。

DroidAnalyst 构建了一个统一的分析模型，结合了多种测试方法的优点，是一种新型的应用程序审查和恶意软件分析框架，静态分析与动态分析协同处理提高了分析检测的效率和准确性，能够同时分析 Dalvik 虚拟机上的字节码和基于系统调用的本地代码，该方法可以发现应用程序在和系统交互的过程中的隐藏的行为。该方法已经对现实生活中的各种良性和恶意软件进行了评估。Andrubis 和 Mobile Sandbox[23]与 DroidAnalyst 相类似，采用动态分析工具 Droidbox 和隐私泄漏检测工具 TaintDroid 进行自动分析。然而，Andrubis 局限于 Android API 9 版本，虽然 Mobile Sandbox 改进了 DroidBox，可以支持 Android API 17 版本，但是上述两种方案均利用已经存在的工具修改 Android 平台，相对于修改操作系统，DroidAnalyst 可以直接支持高版本 API 的设备检测第三方开发者开发的 APK。

Android 中大多数系统资源由 Android 框架管理和保护，应用与系统的相互交互(访问联系人和通话记录等)发生在高级语义层，而系统调用在 Linux 核心层，Android 为应用程序提供具体的 API 来访问系统资源并控制访问规则，使用系统调用来了解应用程序与 Android 的交互行为将失去访问 An-

droid 资源的语义视图，降低重构行为的质量和精确度；传统的解决方案仅拦截与 Binder 驱动交互的系统调用，隐藏了应用程序执行的真正行为。考虑到独特的基于权限的隔离机制，Vetdroid[24]从权限使用的视角重建 Android 应用程序的敏感行为，通过自动采样软件中申请的权限信息进行显式权限使用分析，通过自动跟踪软件中使用通过权限获得的数据的行为进行隐式权限使用分析。具体来说，集中分析了应用程序如何使用权限来访问系统资源以及这些权限敏感的资源之后如何被应用程序利用。安全分析者能够通过权限使用行为流程很容易地检测应用程序内部的敏感行为。但是该方法只能针对引起权限检查的行为进行分析。

除了上述的基于系统调用和权限分析的解决方案，其他基于行为分析的检测方法有：1）Aurasium[25]利用策略执行模块重装应用程序，可以把策略应用在单个或多个应用程序上，通过强制执行任意运行时间的安全策略来控制应用程序的执行。因此，不需要操纵 Android 操作系统来监视应用的行为就可以把任何隐私泄漏行为报告给用户。虽然 Aurasium 会自动干预应用程序访问敏感信息的行为，但却无法检测隐形恶意软件，且由于 Aurasium 需要重装应用程序，所以无法反汇编（或汇编）代码转换的应用程序。2）Drozer 是 MWR Labs 开发的一款交互式的 Android 安全测试框架。Drozer 采用了模块化的设计，用户可以定制开发需要的测试模块，还可以与 Dalvik 虚拟机、其他应用程序的 IPC 端点以及底层操作系统进行交互暴露其漏洞。Drozer 通过攻击测试暴露 Android App 的漏洞，可以大大缩减 Android 安全评估的耗时，但这种方法需要用户人工干预。

动态分析另一个需要考虑的问题是如何触发恶意行为，为了能够在实际设备或模拟器中自动安装应用程序，并自动模拟用户操作（如按键输入、触摸屏输入等），大多数检测方案采用自动控制脚本策略。TaintDroid，DroidBox，Apps Playground，Andrubis 等均采用 Android SDK 提供的 MonkeyRunner 工具，该工具能够产生足够数量的随机事件流来确保一次可以触发大量的交互行为。但不同的应用程序对于不同的用户事件流的频率存在较大的差异，单纯的随机触发不能保证应用程序的针对性。DroidTrace 采用物理修改的方式触发不同的动态加载行为，利用静态分析发现具有动态加载行为的函数，并为这些函数插入触发代码生成转发执行路径，打包成新的应用程序[25]。该方法触发的行为具有针对性，但在程序运行前加入触发代码，没有考虑到应用程序运行时的实时性。以上两种方法并没有考虑到事件触发应用程序的某些行为。例如在程序运行过程中，用户在某地点转到另一地点的过程中，地理位置发生了变化，单纯的测试就无法实现这种行为。

3.2 比较与浅析

表 1 总结了一些效果较好的恶意软件分析与检测方案的目标、方法以及部署情况，从表 1 的数据可以看出，大多数的分析工具只提供了分析结果日志，并没有对待检测的应用程序给出详细的风险评估报告和解决方案报告；基于行为的动态分析方法仍是现今主流的分析方法；为了减少开销、节省检测时间，大多检测方案部署在服务器端，当然，为了提高检测的实时性，也有少部分采用同时部署在智能手机端和服务器端的分布式方式。

表 1　不同动态分析工具的目标、方法和部署方案的总结与比较

工具	目标			方法			部署		
	风险评估	行为分析	检测结果	动静结合	系统调用/API	基于行为	基于模式	手机内部	手机外部	分布式
Andromaly			√				√			√
Andrubis		√	√	√		√	√		√	
Aurasium			√			√			√	
APIMonitor		√			√	√			√	
CopperDroid		√	√		√				√	
Crowdroid			√		√	√	√			√
DroidBox		√			√	√			√	
DroidScope		√			√	√			√	
Drozer	√			√						√
Paranoid Android			√			√			√	
TaintDroid		√			√	√			√	
VetDroid		√				√			√	

为了检测出各种新型的恶意软件，没有单一的解决方案，需要一种结合静态分析方法和动态分析方法的协同框架。这里，我们讨论比较了几种基于沙盒的，并且可在网络上下载到的动静态结合分析的原型系统，其中，除 AASandbox 以外，其余检测系统的动态分析均采用 TaintDroid 技术进行分析。由表 2 可以看出，DroidAnalyst 在 DroidBox 的基础上增加了系统调用分析、机器学习模型以及良好的图形用户界面（GUI），同时增加了抗反分析沙盒功能，克服了恶意应用程序可以容易地检测到特殊环境而不显示任何恶意行为的难题。

表 2　基于沙盒的动态分析工具检测功能的比较

属性	AASandbox	Andrubis	Apps Playground	CopperDroid	DroidAnalyst	DroidBox	SmartDroid
修改平台		√	√				√
GUI 交互					√		√
API Hooking		√	√		√	√	√
日志分析					√	√	
系统调用分析	√				√		√
机器学习模型	√	√		√	√		
防反分析沙盒					√		
识别数据泄漏		√	√	√	√	√	√
识别短信/电话			√		√	√	
网络流量分析		√			√	√	
监控文件操作		√			√	√	
本地代码分析							√
在智能手机上分析		√	√				

结束语　本文简单介绍并分析了 Android 恶意软件及其检测方法，着重介绍了动态分析技术，对 Android 安全中动态分析的最新研究技术进展进行了总结，并分析比较了其优缺点。Android 应用程序动态分析技术能够解决静态分析中不能解决的加密问题和代码混淆问题，提高了检测结果的正确率。但是目前所采用的动态分析技术大多采用人工干预的方式，对操作人员的技术能力要求较高，自动化程度较低，且难以将待检测应用程序的所有可能路径都触发，无法确保其检测的全面性。目前还无法实现一种成熟、通用、商品化的 Android 应用程序安全检测软件。

基于上述分析，本文提出了动态分析中几个值得进一步研究的方向。

（1）运行期间权限使用情况。权限机制的一个问题就在于用户在安装时期并没有足够的上下文来判断程序是否需要

使用不同的权限，这使得用户对程序所申请的权限无法进行完全的判断。如果能分析应用程序在运行过程中如何使用权限和系统进行交互，将帮助用户更好地理解程序权限的使用情况。因此，结合使用安装时期的权限授权以及运行时期的权限使用情况反馈能够对用户的个人数据安全带来更大的帮助，这也是接下来需要进行的研究工作。

(2)利用动态污染跟踪技术自动跟踪程序内部逻辑。动态污染分析属于数据流分析，而数据流分析主要聚焦检验程序中的数据域特性，这是 Android 移动终端系统的用户最为在意的隐患。用户信息泄漏是最常见也是危害最大的恶意行为，数据流分析对这种类型的漏洞具有较好的分析能力，能够发现潜在的信息泄漏。同时需要解决动态污染追踪技术不能监控本地代码的行为，完成对程序内部敏感数据泄漏的监控和检测。

(3)为了使程序自动运行，并完全模拟真实设备中出现的各种用户行为，一方面利用 MonkeyRunner 工具，编写脚本产生大量随机 UI 事件流模拟人机交互，另一方面通过构建事件触发器来触发事件行为。最终实现多粒度的 Android 应用程序行为的监控。

(4)为了使用户能对最终的分析结果有直观的理解，需要良好的人机交互界面以及详细可理解的安全解决方案报告，同时，也可以为应用程序的开发者提供建设性的指导意见。

参 考 文 献

[1] Analytics S. Global Business Smartphone Quarterly Tracking Q4 2015 [OL]. https://www.strategyanalytics.com/access-services/enterprise/mobile-workforce/market-data/report-detail/global-business-smartphone-quarterly-tracking-q4-2015 #.VxX2J7IgIjY

[2] The 2015 U.S. Mobile App Report[OL]. https://www.comscore.com/Insights/Presentations-and-Whitepapers/2015/The-2015-US-Mobile-App-Report

[3] Labs M. Mobile Threat Report: What's on the Horizon for 2016 [OL]. http://www.mcafee.com/us/resources/reports/rp-mobile-threat-report-2016.pdf

[4] Ded: Decompiling Android applications [OL]. http://siis.cse.psu.edu/ded

[5] Dex2Jar. Android decompiling with Dex2jar[OL]. http://code.google.com/p/dex2jar

[6] 屈延文. 软件行为学[M]. 北京：电子工业出版社，2004

[7] Bläsing T, Batyuk L, Schmidt A D, et al. An android application sandbox system for suspicious software detection[C]// 2010 5th International Conference on Malicious and Unwanted Software (MALWARE). IEEE, 2010: 55-62

[8] Enck W, Gilbert P, Han S, et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones [J]. ACM Transactions on Computer Systems (TOCS), 2014, 32(2): 5

[9] Desnos A, Lantz P. Droidbox: An android application sandbox for dynamic analysis[OL]. URL. https://code.google.com/p/droidbox, 2014

[10] Yan L K, Yin H. Droidscope: seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis [C]// Presented as part of the 21st USENIX Security Symposium (USENIX Security 12). 2012: 569-584

[11] Portokalidis G, Homburg P, Anagnostakis K, et al. Paranoid Android: versatile protection for smartphones[C]// Proceedings of the 26th Annual Computer Security Applications Conference. ACM, 2010: 347-356

[12] Weichselbaum L, Neugschwandtner M, Lindorfer M, et al. Andrubis: Android malware under the magnifying glass[R]. Vienna University of Technology, Tech. Rep. TRISECLAB-0414, 2014

[13] Rastogi V, Chen Y, Enck W. AppsPlayground: automatic security analysis of smartphone applications[C]// Proceedings of the third ACM conference on Data and application security and privacy. ACM, 2013: 209-220

[14] API Monitor. Rohitab Batra[OL]. http://www.rohitab.com/apimonitor

[15] Drozer-A Comprehensive Security and Attack Framework for Android[OL]. https://www.mwrinfosecurity.com/products/drozer

[16] Faruki P, Bhandari S, Laxmi V, et al. DroidAnalyst: Synergic App Framework for Static and Dynamic App Analysis[M]// Recent Advances in Computational Intelligence in Defense and Security. Springer International Publishing, 2016: 519-552

[17] Shabtai A, Kanonov U, Elovici Y, et al. "Andromaly": a behavioral malware detection framework for android devices[J]. Journal of Intelligent Information Systems, 2012, 38(1): 161-190

[18] 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. ACM, 2011: 15-26

[19] Zheng C, Zhu S, Dai S, et al. Smartdroid: an automatic system for revealing ui-based trigger conditions in android applications [C]// Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. ACM, 2012: 93-104

[20] Tam K, Khan S J, Fattori A, et al. CopperDroid: Automatic Reconstruction of Android Malware Behaviors[C]// NDSS. 2015

[21] Spreitzenbarth M, Schreck T, Echtler F, et al. Mobile-Sandbox: combining static and dynamic analysis with machine-learning techniques[J]. International Journal of Information Security, 2015, 14(2): 141-153

[22] Zhang Y, Yang M, Yang Z, et al. Permission use analysis for vetting undesirable behaviors in android apps[J]. IEEE Transactions on Information Forensics and Security, 2014, 9(11): 1828-1842

[23] Xu R, Saïdi H, Anderson R. Aurasium: Practical policy enforcement for android applications[C]// Presented as part of the 21st USENIX Security Symposium (USENIX Security 12). 2012: 539-552

[24] Zheng M, Sun M, Lui J. Droidtrace: A ptrace based android dynamic analysis system with forward execution capability[C]// 2014 International Wireless Communications and Mobile Computing Conference (IWCMC). IEEE, 2014: 128-133