Android 恶意软件检测方法综述

王思远[1] 张仰森[1,2] 曾健荣[1] 黄改娟[1]

[1](北京信息科技大学智能信息处理研究所 北京 100101)

[2](国家经济安全预警工程北京实验室 北京 100044)

摘 要 Android 系统是市场占有率最高的移动端操作系统,然而 Android 系统上的恶意应用种类和数量疯狂增长,对用户构成极大的威胁,因此对 Android 系统恶意软件检测方法的研究具有非常重要的意义。分析 Android 系统的安全机制,介绍 Android 恶意软件的分类,总结恶意软件的攻击技术,研究目前的检测方法,比较各类方法的典型系统,列举当前主流厂商的安全软件技术,分析当前研究中存在的问题,对未来恶意软件的检测方向进行展望。

关键词 Android 恶意软件检测 静态检测 动态检测 网络数据特征检测

中图分类号 TP3 **文献标志码** A DOI: 10.3969/j.issn.1000-386x.2021.09.001

OVERVIEW OF ANDROID MALWARE DETECTION METHODS

Wang Siyuan[1] Zhang Yangsen[1,2] Zeng Jianrong[1] Huang Gaijuan[1]

[1](*Institute for Intelligent Information Processing, Beijing Information Technology University, Beijing 100101, China*)

[2](*National Economic Security Early Warning Engineering Beijing Laboratory, Beijing 100044, China*)

Abstract The Android system was the mobile terminal operating system with the highest market share. However, the variety and number of malicious applications on the Android system are growing madly, posing a great threat to users. Therefore, the study on the detection method of malware in Android system is of great significance. This paper analyzed the security mechanism of Android system, introduced the classification of Android malware, summarized the attack technology of malware, studied the current detection methods, compared the typical systems of various methods, listed the security software technologies of current mainstream vendors, analyzed the problems existing in the current research, and finally looked forward to the detection direction of malware in the future.

Keywords Android Malware detection Static detection Dynamic detection Network data feature detection

0 引 言

Android 系统是 Google 公司于 2007 年发布的基于 Linux 内核的操作系统。经过 11 年的优化和升级,截至 2018 年 8 月,Google Play 的应用程序总数已超过 280 万[1],是全球最受欢迎及市场占有率最高的移动端操作系统。但与此同时,由于其系统的开源性、自由性等特征,也使得大量的恶意应用乘虚而入。根据相关报告[2],Android 系统的恶意应用程序数量于 2015 年达到峰值,在随后的几年内,恶意应用程序数量虽然有所下降,但总体态势不容乐观。此外关于恶意应用的检测形式依然严峻,因为随着 Android 版本更新和漏洞的修补,恶意应用的攻击方法也在不断地更新和变化。

当前,应用安全检测已经成为了一个非常热门的研究领域,而且市场上各大安全厂商也推出了各自的安全检测软件,如 AVG Antivirus、Lookout Security&Antivirus、

收稿日期:2019-10-07。国家自然科学基金项目(61772081);科技创新服务能力建设—科研基地建设—北京实验室—国家经济安全预警工程北京实验室项目(PXM2018_014224_000010)。王思远,硕士生,主研领域:网络安全,自然语言处理。张仰森,教授。曾健荣,硕士生。黄改娟,高级实验师。

Norton Mobile Security 和 Trend Micro Mobile Security 等，但检测结果表明，市场上现有的安全检测软件在恶意应用程序的检测方面效果不佳[3]。同时恶意应用程序的攻击手段也在不断更新，这对恶意应用检测技术提出了更高要求。

在市场发展与学术界的共同推动下，恶意应用检测的研究目前已形成一套较为成熟的理论成果，但学界在这一领域缺乏系统回溯与总结。本文从 Android 系统安全机制、恶意应用程序、恶意应用程序检测方法等方面对恶意应用程序检测进行分析，列举总结了当前主流厂商的安全软件技术，并在此基础上提出当前研究的不足和今后发展方向。

1 Android 系统的安全机制

Android 系统是基于 Linux 内核实现的，同时在 Linux 原有的安全机制基础上结合移动端的特性，设计了应用程序签名机制、权限机制、进程沙箱隔离机制等安全机制，保证了应用程序在发布、安装和运行过程中的安全[4]。

1.1 应用程序签名机制

Android 应用程序签名机制是指在应用程序发布时，开发者对应用进行签名和在安装应用程序时系统通过签名对应用进行验证的机制。签名机制有如下作用：(1) 应用程序在未安装时，通过数字签名实现对程序的版本控制；(2) 应用程序开发者使用私钥对应用程序进行签名，Android 系统使用公钥对应用程序进行验证溯源，与程序开发者建立信任关系；(3) 数字签名还可以对应用程序安装包的完整性进行验证，保证其安全。

应用程序签名机制有两个版本的可选择方案，如图 1 所示。当对应用程序进行签名时，V1 版本对安装包压缩文件的非目录和非过滤文件进行签名，而 V2 版本则在 V1 版本基础上对应用程序安装包文件整体进行签名。当对应用程序进行验证时，V1 版本需要解压安装包，然后对每个文件进行验证，而 V2 版本直接对安装包压缩文件整体进行验证，这种验证方式防止了安装包文件被篡改，同时加快了签名验证速度。应用程序签名机制，实现了对应用重打包、应用篡改等恶意行为的检测和恶意应用的过滤。

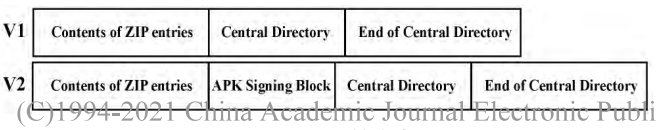

图 1 Android 签名机制

1.2 权限机制

权限机制是指应用程序向系统申请该应用所需服务的权限，并根据获取的权限运行服务的机制。权限即定义一个对象是否具有使用系统某个服务的能力，权限的定义信息包括包名、标签、描述和保护级别。在 Android 特有的进程沙箱隔离机制下，应用程序只可访问当前 UID 下的系统资源和全局可访问资源，当需要使用其他系统资源时，需要根据资源的安全级别说明(如表 1 所示)，进行相应的申请操作。

表 1 Android 权限级别

权限安全级别	说明
Normal	应用声明即可默认获得授权
Dangerous	应用声明，通过显示请求得到用户确认才可获得授权
Signature	只有与当前权限使用相同证书的应用才可声明获得授权
SignatureOrSystem	拥有与当前权限使用相同签名证书密钥的应用才可声明获得授权或者系统镜像的部分应用

在 Android 6.0 之前的版本，系统对应用程序的权限授权结果只有两种为永久授权和不授权，这种权限机制称为粗粒度的权限管理，这就使得一些恶意应用程序有了可乘之机。在 Android 6.0 版本之后，系统对安全级别为 dangerous 的服务进行授权时，应用程序必须在运行状态并且需要用户授权同意，才可以获得授权，这种方式做到了细粒度的权限管理，加强了用户对应用权限的可控性，提高了系统的安全系数。

1.3 进程沙箱隔离机制

进程沙箱隔离机制则是应用程序运行在系统为其开辟的独立 Dalvik 虚拟机中，拥有独立的运行空间和系统资源空间，保证应用程序间独立和系统安全的机制。Android 系统将 Linux 系统的用户隔离机制转化为应用程序的隔离机制，并在此基础上提出了进程沙箱隔离机制，其结构如图 2 所示。应用程序在安装时，被系统赋予了唯一的用户标识码 UID，在程序存续期间保持不变。程序在 Dalvik 虚拟机上运行期间，系统对同一 UID 下的资源进行唯一标识，使得应用程序各自的资源互不干扰，进而使得各个应用程序的资源都得到安全隔离和保护。在多个应用程序间需要共享资源时，Android 系统提供了以下几种资源共享方式：① 完全暴露(资源对系统所有应用程序完全共享)；② 权限提示暴露(当其他应用需要请求资源文件时，通过向用户请求，实现资源的共享)；③ 私有暴露(通过设置不

同应用的共享 UID，实现共享 UID 下应用程序资源的共享如图 3 所示）。

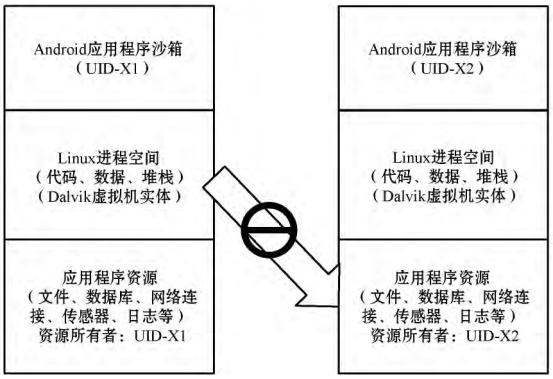

图 2　Android 应用程序沙箱机制

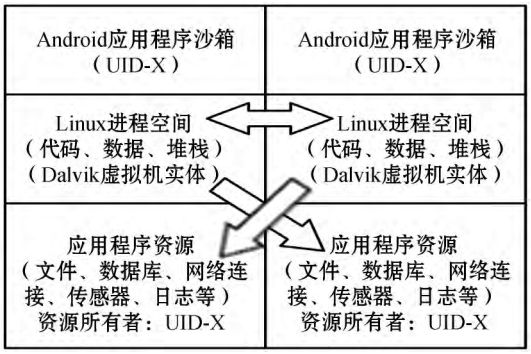

图 3　Android 应用程序共享 UID

这种安全机制既能保证应用运行空间的独立性，又能满足资源共享的需求。在发现恶意软件之后，可以通过隔离其所在的虚拟机，以保证系统其他应用程序的安全。

2　Android 恶意应用检测

2.1　Android 恶意应用程序

Android 恶意应用程序种类变换多样，有很多分类方法。一种是按照传统进行分类，卿斯汉[7]认为可以分为木马类、病毒类、后门类、僵尸类、间谍软件类、恐吓软件类、勒索软件类、广告软件类和跟踪软件类等。另一种是按照恶意软件特征进行分类，Zhou 等[5]认为可以分为恶意软件安装、恶意软件运行、恶意负载、权限使用等；同样地按照恶意软件特征进行分类，Li 等[9]认为可以分为恶意扣费类、资源消耗类、诈骗误导类和系统破坏类等。而一种新颖的分类方法是从恶意应用包含的恶意内容和恶意行为两个角度进行分类[5-8]。该方法中包含恶意内容的应用程序是应用程序自身对移动终端设备无任何安全隐患，其带来的危害主要是由应用程序传播的内容或后台人员行为导致的，如通过伪造正规网站对用户实施诈骗的应用程序、传播淫秽色情视频的应用程序等；而包含恶意行为的应用程序主要指应用程序通过恶意代码或者恶意行为对用户或移动设备产生危害的应用程序。

通过对恶意应用的攻击方法进行分析总结，典型的恶意应用攻击方法包括重打包攻击、提权攻击、其他攻击、新型攻击。

2.1.1　重打包攻击

重打包攻击指的是反编译第三方应用程序，它将恶意代码植入应用程序，伪造第三方应用的签名，重新打包伪装成第三方应用进行发布。病毒、木马、广告等恶意软件可通过该技术进行传播，对用户造成危害，针对这种攻击方式 Android 官方通过更新 V2 版本的签名机制，结合签名密钥轮换机制，防止了恶意应用程序对第三方安装包文件的篡改。

重打包攻击进一步发展后，还出现了更新包攻击，这是将恶意代码的植入方式从静态 apk 包植入转到应用程序运行时动态植入，绕过了静态检测的安全检查，这对恶意应用检测提出了更高的要求。

2.1.2　提权攻击

提权攻击是指恶意应用将普通应用程序的系统服务权限提升，获取到恶意应用需要的权限。攻击手段包括恶意利用 Android 系统漏洞，通过第三方代理软件获取授权等。由于 Android 版本碎片化严重，恶意应用程序对不同 Android 版本采用了不同的攻击手段，而恶意应用安全检测无法及时更新全版本的 Android 检测方法，所以如果当恶意应用获取到系统最高权限，就可以执行任意权限的操作，这对系统安全构成了极大的威胁。

2.1.3　其他攻击

除此之外，恶意应用攻击方法还有资费消耗和隐私泄露等。资费消耗包括话费吸取、流量消耗等恶意消耗用户资费的行为。话费吸取的方式如强行定制特定 SP(service provider) 服务等，流量消耗则表现为后台静态访问导流网站、广告等恶意行为。

隐私泄露指隐私窃取模块搜集移动端用户通信录、短信、定位数据，同时收集用户访问第三方应用的数据等隐私信息，将用户隐私数据传输到服务器后在黑市进行交易。隐私泄露带来多维的用户危害，如通过对用户多维数据分析进行精确画像实施电子诈骗，通过用户位置、社交、通信等信息对用户精确定位跟踪等行为，对用户安全构成严重危害。

2.1.4　新型攻击

随着 Android 系统的更新和攻击方式变化，市场

上也出现了新的恶意攻击方式，包括：利用调试接口传播（当 Android 系统的 ADB 调试端口开启时，可用于恶意应用程序的传播，如挖矿蠕虫 ADB. Miner 恶意应用软件的传播）；利用 Kotlin 编程语言开发恶意应用软件，如远程控制，隐私窃取等恶意软件；通过篡改剪切板内容，达到特定目的，如利用篡改支付宝吱口令赚取赏金，篡改比特币账户地址盗取比特币等新型恶意攻击方法。

2.2 静态的恶意应用检测方法

静态检测方法是指在应用程序不运行的情况下，基于应用程序特征进行系统检测分析[10-11]。静态检测的关键技术点在于如何选择特征和如何分析特征，其中常用的特征包括应用权限、Java 代码、意图过滤器、网络地址、字符串和硬件组件等，而分析特征的方法包括差异分析、特征匹配、权限检测、函数检查、类别检测等。静态检测的优点是检测速度快、可以上传安装包文件到服务器端检测等，缺点是无法有效识别利用 Android 系统漏洞和静态检测对抗技术的恶意应用程序，其中常见的对抗技术有代码混淆、多态变换等。几种常用的静态检测方法包括基于应用程序安装包文件特征的检测、基于应用程序代码分析的检测，以及基于应用程序其他特征分析的检测等。

2.2.1 基于应用程序安装包文件特征的恶意应用检测

Android 应用程序安装包文件解压缩后如表 2 所示，对其中非资源文件和非签名文件进行静态检测。

表 2 Android 安装包文件资源

文件类型	描述
assets 文件夹	Android 原生资源文件
res 文件夹	可编译的资源文件
META-INF 文件夹	签名文件
AndroidManifest. xml 文件	项目清单文件
classes. dex 文件	Java 源码编译后的字节码文件
resources. arsc 文件	资源索引文件

针对应用程序安装包文件的静态检测，首先对包含其中的签名文件、资源文件、代码文件、清单文件等文件进行特征表征，然后进行相似性比较，判断是否为恶意应用程序。DroidMOSS[12] 将应用程序代码片段进行哈希计算得到哈希值，以此作为应用程序特征；通过计算编辑距离进行相似性比较，以此判断是否为恶意应用程序。当应用程序使用无用方法注入、代码混淆等对抗攻击方法时，该检测方法失效。如何实现对使用对抗攻击技术的恶意应用的检测成了最大的问题，为此 Zheng 等[13] 提出了基于应用程序方法/类和 API 调用序列生成应用程序的三级签名方法，通过应用签名进行相似性比较，该方法虽然可以检测到使用代码混淆等攻击方法的恶意应用，但是检测耗时较长，效率低下。为了在保证检测效果的基础上缩短检测时间，秦中元等[14] 提出一种基于多层次签名（API 签名、Method 签名、Class 签名、APK 签名等）的检测方法，与恶意应用样本库进行比对，匹配恶意应用程序，虽然该方法检测过程较繁琐，但在保证检测效果的基础上，有效加快了检测速度，缩短了检测时间。

2.2.2 基于应用程序代码分析的恶意应用检测

基于应用程序代码分析的检测是将应用程序代码抽象为控制流图、数据流图和程序依赖图，提取特征后通过计算相似性，进而确定是否为恶意应用程序。代码是应用程序的基础，而程序控制流图又是对代码的抽象概括，文献[16-17]提出了一种基于程序三维控制流图实现的检测方法，该方法首先根据控制流图的重心坐标构造恶意应用特征库，然后通过相似性判断，确定是否为恶意应用程序。该方法虽然有较高的检测精确度，但是在特征表示阶段，抽象应用程序的控制流图工作量较大，导致检测效率较低。而且重心坐标计算的精确性会直接影响检测精度，所以当恶意应用使用代码混淆技术时会导致检测失效。为了解决代码混淆技术带来的影响，Tian 等[18] 提出了基于应用程序类依赖图和函数调用图的检测方法，该方法对应用程序的用户交互、API 调用、权限请求等进行特征表征，然后使用分类器进行分类，实现对恶意应用的检测，降低了对程序代码的依赖。由于该方法对应用程序进行了全面的特征表征，故导致检测效率较低。同样地，Fan 等[19] 提出了基于敏感 API 函数子图的检测方法，该方法对敏感 API 函数子图进行特征表征，然后使用随机森林、决策树等方法进行分类完成检测。该方法降低了对代码特征的依赖，可以抵御典型的混淆攻击技术，但无法检测使用加密攻击方法的恶意应用。由于提取代码深层语义信息，可以有效降低代码形式变化带来的影响，汪润等[15] 提出了一种基于程序依赖图的语义信息的检测方法，通过计算应用程序间的 Jaccard 距离进行相似性比较，实现粗分类，然后基于程序依赖图的语义信息进行细粒度分类比较。该方法可以有效抵御常见的混淆攻击，但会存在一定的漏报和误报现象。

2.2.3 基于应用程序其他特征分析的恶意应用检测

基于应用程序其他特征分析的检测，首先是将应用程序的 UI 界面、多媒体文件、APP 名称、图标等做特征表示，然后在进行相似度计算，进而确定是否为恶意应用程序。根据应用布局特征，Sun 等[20] 首先用程序的布局文件得到视图的层次结构，接着计算布局间的

编辑距离和哈希值，通过和样本库对比，最终实现恶意应用的检测。同样地，Lyu 等[21] 基于应用布局文件，得到应用的全局层次结构，然后计算哈希值，进行相似性对比，最终实现恶意应用的检测。根据应用界面特征，Chen 等[17] 先定位应用 Activity 界面和 Dialog 界面，进而得到应用的界面跳转关系图，然后通过计算图重心坐标相似性实现恶意应用检测。该方法有较快的检测速度，可用于大规模的恶意应用检测。将界面特征和代码特征结合，可以兼顾检测速度和精度，所以汪润等[22] 提出基于应用 UI 和程序依赖图的两级恶意应用检测方法，使用 UI 结构进行快速相似性检测，通过程序依赖图特征做相似性对比，该方法在检测速度和准确度上均有较好表现。

2.3　动态的恶意应用检测方法

动态检测方法是指在应用程序运行时，通过收集系统信息构建特征库，之后进行相似度比较，实现对恶意应用检测的方法。动态检测可以分为两大类，一类是基于应用行为分析进行检测，另一类是基于污点跟踪追踪进行检测。动态检测的难点在于选取特征和实现对应用功能的全面检测，其中特征的选取从系统调用、API 调用、资源访问、网络特征等[23] 方面进行，而如何提高应用程序功能检测的覆盖率(代码覆盖率) 仍没有一个最好的解决方案。

2.3.1　基于应用行为分析的恶意应用检测

动态检测只在应用运行时进行检测，无法做到对应用全部功能进行测试，Wong 等[24] 提出 IntelliDroid 应用输入生成器，可以自主配置，然后生成特定的应用功能触发行为，解决了动态检测只能检测应用程序动态运行时产生的恶意行为的问题，可以尽可能多地对应用程序功能进行检测。

对应用功能的检测，可以通过检测对系统服务的调用实现，CopperDroid[25] 结合系统调用和 Binder 通信行为数据，可以重建出恶意应用的高级语义行为，它包括应用系统内核行为和 Android 应用的特定行为，可以用来准确地描述应用程序功能。该方法不需要修改 Android 操作系统即可实现以上功能，但只能对应用程序进行离线分析，而不能在终端实时分析[29] 。

应用运行时，非系统调用信息同样可以对应用进行特征刻画，Gianazza 等[26] 提出 PuppetDroid 恶意应用检测系统，通过搜集用户与应用程序间的实时交互行为，刻画出了恶意应用程序特征，以此进行恶意应用的检测。同样地，Shabtai 等[27] 提出 Andromaly 检测系统，搜集应用运行信息作为应用程序的特征，如 CPU、内存消耗、进程状态等，训练一个有监督的分类器实现对恶意应用的快速检测。Saracino 等[28] 提出 MADAM 检测系统，从内核层面、APP 层面、用户层面和应用层面对应用程序进行特征表征，特征融合对应用程序关联分析，实现恶意应用的检测。

通过上述分析可以看出，现有的动态检测方法无法同时对操作系统与应用程序、应用程序内部进行表征[30] ，应用程序的输入生成也停留在 UI 触发层面，还不能全面地表征应用程序。

2.3.2　基于污点跟踪的恶意应用检测

对于应用数据的特征提取，无法做到对应用功能全流程的刻画，针对这一问题，Enck 等[31] 提出基于污点跟踪的 TaintDroid 恶意应用检测系统，通过修改 Android 操作系统，能够动态跟踪恶意应用的数据流和系统调用。TaintDroid 实现四个层面的追踪: 变量级，方法级，信息级和文件级，但是 TaintDroid 无法实现应用程序控制流的跟踪和信息流泄露的检测。在此基础上，Zhang 等[32] 开发了 VetDroid 检测系统，基于应用对敏感行为、权限和资源的调用，构造出了应用程序的隐私数据权限使用图，重建了细粒度的应用行为，用于恶意应用的检测。

将多种动态检测特征结合，Lindorfer 等[33] 提出的 AppsPlayground 检测系统，综合了污点分析、API 监控、系统调用监控等多种动态分析方法，对应用程序进行检测分析，对隐私泄露和恶意应用行为的检测效果较好。可以看出，现有的污点分析技术只能针对数据流和系统调用的追踪，对于控制流和本地代码无法进行有效跟踪[34] 。

2.4　基于网络流量的恶意应用检测方法

基于网络流量数据的检测是指，检测系统首先对网络数据进行特征提取，通过相似性计算，最终实现对恶意应用的检测。移动端的恶意应用程序，大都通过网络数据传输实现对恶意行为的控制，Zhou 等[3] 指出，90% 以上的恶意软件通过僵尸网络的网络数据传输控制恶意行为，所以在网络流量数据中，可以通过特征表征实现恶意应用程序的检测。根据特征提取的途径不同，把基于网络流量的检测分为两大类，一类是基于网络流量数据统计特征进行检测，另一类是基于网络流量数据内容特征进行检测。

2.4.1　基于网络流量数据统计特征的恶意应用检测

由于网络流量数据量巨大，无法对全部数据进行特征构建，Gu 等[35] 基于深度包解析(DPI) 技术提出 BotHunter 检测系统，通过对恶意应用程序产生的网络数据包进行分析，得到恶意应用程序网络流量的特征字符串，通过对特征字符串的匹配实现恶意应用检测。

基于网络异常数据，识别僵尸网络命令与控制通信的行为数据并进行特征表征，同时对网络数据包大小进行统计特征表征，通过聚类方法实现恶意应用的检测，提出 BotMiner 检测系统[36-37]，该系统检测不受网络协议和僵尸网络结构影响，有较好的可扩展性，但是检测精度较低。

对网络数据传输时间特征进行分析，Zhao 等[38] 基于网络请求间隔时间构建特征集，然后结合机器学习算法实现对恶意应用的检测，但是网络请求时间间隔的影响因素较多，使得该系统的准确率较低，误判率较高。通过分析网络解析协议，Wang 等[39] 基于 DNS 解析，匹配恶意域名和 IP 地址，实现了对僵尸网络的解析。该系统对恶意应用的检测，基于已经构建好的恶意应用请求特征库，所以检测速度快，但无法检测新出现的恶意应用。同样地，Sharifnya 等[40] 基于 DNS 流量分析提出 DFBotKiller 僵尸网络检测系统，通过对可疑群体活动和可疑故障进行特征表征，实现对僵尸网络的识别。流量包数据对应用也有一定的区分度，Arora 等[41] 基于流量包统计信息，构建了恶意应用特征库，最终了对恶意应用的检测，检测速度较快，但是该系统基于已知恶意应用特征库，无法检测新出现的恶意应用。

加密流量的出现，使得传统网络数据统计方式失效，Taylor 等[42] 基于 TCP 和 IP 报文头部结构数据进行特征表征，提出 APPScanner 检测系统。系统共提取 54 种特征，然后通过随机森林算法实现对恶意应用程序的检测，该系统可以对加密流量进行检测。Conti 等[43] 基于加密流量进行特征统计，构建了应用的加密流量特征库，实现了对恶意应用的识别，该系统不涉及流量内容的识别，可以检测加密流量。对网络协议进行挖掘，Kwon 等[44] 基于 DNS 响应时间特征进行恶意应用的检测，实现了对加密流量的检测，同时由于影响 DNS 响应时间的因素较多，导致该方法会有较大的误报率。

2.4.2 基于网络流量数据内容特征的恶意应用检测

网络流量数据的内容和应用程序行为密切相关，Asdroid 等[45] 基于网络流量内容匹配进行恶意应用检测，通过将 UI 数据内容和应用程序实际调用操作进行对比，进行恶意应用特征表征，最终实现恶意应用检测。信息的交互是通过网络数据传输实现的，Ren 等[46] 对 HTTP 流量文本进行分析，通过对敏感信息、关键字匹配，判断应用是否存在用户隐私泄露等恶意应用行为，但是该方法不适用于检测加密流量。Nan 等[47] 通过对比应用申明行为和实际行为，构建恶意应用特征库，实现了对恶意应用程序的识别。因为对应用程序的网络行为进行了细粒度的重现，所以检测精度较高。

网络数据传输需要在网络协议下进行，而网络协议中肯定包含网络报文，Zaman 等[48] 通过对 HTTP 报文中的 URL 进行匹配，然后将 URL 与恶意域名库比对，实现对恶意应用程序的检测。同样地，Wang 等[49] 分别基于 TCP 和 HTTP 报文数据提出了 TrafficAV 检测系统，由于 HTTP 报文包含信息较多，使得系统具有较高的检测精度，但是不能检测加密流量，而 TCP 报文既能保证检测精度又不受加密流量的影响。

因为对流量数据内容的依赖，始终无法对加密流量有好的检测效果，Arora 等[50] 基于网络层信息特征，通过朴素贝叶斯算法实现恶意应用程序的检测。该方法中的特征选取算法，有效减少了对特征数的依赖，而是选取了报文接收时间间隔、网络传输字节数等信息作为网络层特征，解决了无法对加密流量数据进行检测的问题。

3 主流厂商的 Android 软件检测技术

当前提供 Android 软件检测服务的国内外厂商有腾讯、百度、360、Google、LookOut、ZimPerium 等。其中腾讯和 Google 的 Android 软件检测技术在国内外厂商中又具有代表性，因此本文对这两家公司的 Android 软件检测技术进行分析和总结。

3.1 Google Play Protect 移动端安全服务

Google Play Protect[51] 是 Google 在 2017 年 5 月提出的，支持云端和离线两种方式扫描恶意应用的安全服务。Google 实现该检测模型的过程如下：(1) 数据集的构建。Google 对谷歌应用商店和网页上可以找到的所有应用，进行静态检测和动态检测分析，构建了应用特征数据集；同时收集谷歌应用商店中用户反馈信息、开发人员信息等对识别恶意应用有价值的数据，构建了应用商店数据集。(2) 检测算法的实现。使用了逻辑回归、深度神经网络等机器学习算法，构建了识别特定类别、特定家族等多级恶意应用检测模型，该模型实现了对恶意应用的检测和分类。

Google2018 年度大会指出，移动设备从谷歌商店下载应用，导致感染一个或多个恶意应用的概率为 0.08%；移动设备从外部来源下载应用，导致感染恶意应用的概率要比前者高出 8 倍为 0.68%。由此可见，Google Play Protect 在一定程度上保证了用户免于恶意应用的侵害。但赛门铁克公司研究发现，Google Play 中存在之前已被发现的恶意应用，通过更改应用名称

而重新发布在 Google Play 的现象，这表明 Google Play Protect 的检测仍然需要进一步完善。

3.2　腾讯手机管家移动端安全服务

腾讯手机管家[52]是腾讯公司研发的，支持传统方式和 AI 云端方式扫描恶意应用的手机安全软件。该软件核心包括两部分: (1) 鹰眼(TAV 反病毒引擎)，使用传统的静态检测、动态检测方法对应用进行分析检测，同时在云端结合大数据、机器学习和数据挖掘技术，对海量样本数据进行分析处理，使其检测方法具有自学习、抗变形、难对抗的特点。(2) 哈勃文件分析系统，在云端动态分析海量应用样本，找到恶意应用的特征，生成恶意样本库，提供给 TAV 引擎。

在 2018 年度 AV-Comparatives 评测中，腾讯手机管家以“100% 检测率”和“零误报”的成绩名列前茅，获得了国内外安全厂商的认可，成为了国内市场占有率较高的手机安全软件。同时，由于国内用户不能访问谷歌应用商店，各大手机厂商如华为、小米、vivo 等应用商店有各自不同的安全检测标准，使得恶意应用更容易流入市场，这就对安全软件的检测能力提出了更高的需求。

当前主流安全软件普遍采用动静结合的分析方法和离线加云端的部署方式对恶意应用进行检测，同时结合机器学习方法来应对海量的恶意应用，但仍然无法做到对恶意应用的完全检测。

4　当前研究中存在的问题

4.1　静态检测存在的问题

静态检测是通过构建恶意应用特征库实现检测的方法，特征包括应用程序安装包文件、通过程序代码构建的各类流图、应用图片、媒体文件、界面布局等。但当前特征提取局限于表层信息，欠缺对深层语义信息的提取，同时特征库的更新也不能做到及时全面，导致漏报和误报的现象。

静态检测目前主要有两种运行形式，一种在应用程序安装时运行，因兼顾用户体验，会简化检测流程，从而导致检测效果较差; 另一种在云平台远程运行，当用户安装时安装包可能被篡改，导致不能完全保证应用安全。

此外，静态检测无法有效应对恶意应用采用对抗技术的场景，如会出现使用代码混淆、无用代码注入、加密技术、多态变化等攻击技术的恶意应用程序，导致检测效果较差或者失效。

4.2　动态检测存在的问题

(1) 由于动态检测必须在应用程序运行时检测，这样一方面会提高能耗缩短待机时间，另一方面也会占用系统资源导致运行卡顿，因此用户体验较差，难以在用户中大规模部署。

(2) 动态检测在执行有效的代码测试用例时才可以检测应用程序对应的功能项，只有覆盖率高的代码测试用例才能做到对应用的全面检测，但是高覆盖率的代码测试用例是当前需要解决和完善的问题。

(3) 动态检测在生成测试用例时，往往是从应用功能出发，制定相应的调用策略，没有结合应用场景中用户的实际使用经验，导致测试用例的真实性较低，实际检测效果较差。

4.3　基于网络流量检测存在的问题

(1) 基于网络流量检测，通过对恶意应用分析，构建恶意应用网络流量数据库，在应用运行期间进行实时数据采集，通过与特征库比对实现检测。存在特征库更新不及时、系统资源开销较大、无法大规模部署的问题。

(2) 基于网络流量检测，需要采集网络协议报文的特定数据做检测[51]，对于加密网络数据，无法提取到相应信息，导致检测效果较差或失效。

5　结　语

Android 恶意应用程序检测在经过十多年的技术积淀后，已经取得了非常丰硕的研究成果，但是目前仍然没有一种检测方法可以实现对恶意应用的完全检测，因此针对 Android 恶意应用程序的检测仍然值得广大学者继续研究。

(1) 加深对代码语义特征的提取。恶意应用检测使用的特征多种多样，但是应用程序代码是构成应用程序的基石，因此对于代码的特征提取仍然是最具价值的。同时，随着混淆技术、多态分布等对抗技术的出现，使得有效代码特征提取更为困难，基于此，在未来的研究中需要继续探索多文件联合提取及基于机器学习、深度学习进行语义信息表征等技术。

(2) 结合多种检测方式进行检测。恶意应用的检测，可以建立多级检测机制，包括静态检测快速初检、动态检测运行时监测、基于网络数据检测云端检测。同时，当前动静结合的检测方法也已经取得了一定成功，而且这种检测方式仍是未来公认的发展方向。

(3) 加大对网络数据特征的挖掘。基于网络数据的检测，一直被认为是静态检测或者动态检测的一个

子项研究，本文单独列出进行分析是因为云端检测对于数据特征的提取更全面，云计算和大数据技术的发展使得检测精度更高，释放了用户系统资源，有良好的用户体验，这将会是未来的发展方向。

(4) 构建全面的恶意应用特征库。当前特征库的构建，局限于从应用本身提取，对于应用的衍生数据收集不足，如应用市场中的用户评价、应用程序功能描述等数据，上述数据都可以深入挖掘，对应用做很好的表征。特征库构建如果打破应用自身的局限，会有更好的发展。

参 考 文 献

[1] App Annie. Google Play 10 年数据纵览报告 [EB/OL]. (2018 - 07 - 11) [2019 - 10 - 07]. https://www.appannie.com/cn/insights/market-data/the-state-of-mobile-2019/.

[2] 2018 年 Android 恶意软件专题报告 [EB/OL]. (2019 - 02 - 18) [2019 - 10 - 07]. http://zt.360.cn/1101061855.php?dtid=1101061451&did=610100815.

[3] Zhou Y J, Jiang X X. Dissecting Android malware: characterization and evolution [C]//2012 IEEE Symposium on Security and Privacy (SP). IEEE, 2012: 95 - 109.

[4] Google Inc. Android security [EB/OL]. (2013 - 06 - 26) [2019 - 10 - 07]. https://source.android.com/tech/security/.

[5] Zhou Y J, Jiang X X. Detecting passive content leaks and pollution in Android applications [C]//Proceedings of the 20th Annual Network and Distributed System Security Symposium. ACM, 2013: 1 - 16.

[6] 张玉清，王凯，杨欢，等. Android 安全综述 [J]. 计算机研究与发展，2014，51(7)：1385 - 1396.

[7] 卿斯汉. Android 安全研究进展 [J]. 软件学报，2016，27(1)：45 - 71.

[8] 刘剑，苏璞睿，杨珉，等. 软件与网络安全研究综述 [J]. 软件学报，2018，29(1)：42 - 68.

[9] Li J, Sun L C, Yan Q B, et al. Android malware detection [J]. IEEE Transactions on Industrial Informatics, 2018, 14(7): 3216 - 3225.

[10] 孙伟，孙雅杰，夏孟友. 一种静态 Android 重打包恶意应用检测方法 [J]. 信息安全研究，2017，3(8)：692 - 700.

[11] 蒋煦，张长胜，戴大蒙，等. Android 平台恶意应用程序静态检测方法 [J]. 计算机系统应用，2016，25(4)：1 - 7.

[12] Zhou W, Zhou Y J, Jiang X X, et al. Detecting repackaged smartphone application in third-party Android marketplaes [C]//Proceedings of the 2nd ACM conference on data and application security and privacy. ACM, 2012: 317 - 326.

[13] Zheng M, Sun M S, Lui J C S. Droid analytics: A signature based analytic system to collect, extract, analyze and associate Android malware [C]//2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, 2013: 163 - 171.

[14] 秦中元，王志远，吴伏宝，等. 基于多级签名匹配算法的 Android 恶意应用检测 [J]. 计算机应用研究，2016，33(3)：891 - 895.

[15] 汪润，唐奔宵，王丽娜. DroidFAR：一种基于程序语义的 Android 重打包应用抗混淆检测方法 [J]. 武汉大学学报(理学版)，2018，64(5)：407 - 414.

[16] Chen K, Liu P, Zhang Y J. Achieving accuracy and scalability simultaneously in detecting application clones on Android markets [C]//Proceedings of the 36th International Conference on Software Engineering. ACM, 2014: 175 - 186.

[17] Chen K, Wang P, Lee Y, et al. Finding unknown malice in 10 seconds: Mass vetting for new threats at the Google-Play Scale [C]//Proceedings of the USENIX Conference on Security Symposium. USENIX Association, 2015: 659 - 674.

[18] Tian K, Yao D F, Ryder B G, et al. Detection of repackaged Android malware with code-heterogeneity features [J]. IEEE Transactions on Dependable and Secure Computing, 2020, 17(1): 64 - 77.

[19] Fan M, Liu J, Wang W, et al. DAPASA: Detecting Android piggybacked apps through sensitive subgraph analysis [J]. IEEE Transactions on Information Forensics and Security, 2017, 12(8): 1772 - 1785.

[20] Sun M S, Li M M, Lui J C S. Droideagle: Seamless detection of visually similar Android apps [C]//Proceedings of the 8th ACM Conference on Security and Privacy in Wireless and Mobile Networks. ACM, 2015: 1 - 12.

[21] Lyu F, Lin Y P, Yang J F, et al. SUIDroid: An efficient hardening-resilient approach to Android app clone detection [C]//2016 Trustcom/BigDataSE/ISPA. IEEE, 2016: 511 - 518.

[22] 汪润，王丽娜，唐奔宵，等. SPRD：基于应用 UI 和程序依赖图的 Android 重打包应用快速检测方法 [J]. 通信学报，2018，39(3)：159 - 171.

[23] 彭国军，李晶雯，孙润康，等. Android 恶意软件检测研究与进展 [J]. 武汉大学学报(理学版)，2015，61(1)：21 - 33.

[24] Wong M Y, Lie D. IntelliDroid: A targeted input generator for the dynamic analysis of Android malware [C]//23rd Annual Network and Distributed System Security Symposium. The Internet Society, 2016: 21 - 24.

[25] Tam K, Khan S J, Fattori A, et al. CopperDroid: Automatic reconstruction of Android malware behaviors [C]//22nd Annual Network and Distributed System Security Symposium. The Internet Society, 2015.

[26] Gianazza A, Maggi F, Fattori A, et al. PuppetDroid: A user-centric UI exerciser for automatic dynamic analysis of similar Android applications [EB]. arXiv: 1402.4826, 2014.

[27] 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.

[28] Saracino A，Scandurra D，Dini G，et al. MADAM: Effective and efficient behavior-based Android malware detection and prevention [J]. IEEE Transactions on Dependable and Secure Computing，2018，15(1)：83－97.

[29] 倪振宇. 基于动态行为分析的 Android 软件恶意行为实时检测 [D]. 南京: 东南大学，2016.

[30] Yan L K，Yin H. DroidScope: Seamlessly reconstructing the OS and Dalvik semantic views for dynamic Android malware analysis [C] //Proceedings of the 21st USENIX Conference on Security Symposium. USENIX Association，2013：29.

[31] Enck W，Gilbert P，Chun B G，et al. TaintDroid: An information flow tracking system for real-time privacy monitoring on smartphones [J]. Communications of the ACM，2014，57(3)：1－29.

[32] Zhang Y，Yang M，Xu B Q，et al. Vetting undesirable behaviors in Android apps with permission use analysis [C] //2013 ACM SIGSAC Conference on Computer and Communications Security. ACM，2013：611－622.

[33] Lindorfer M，Neugschwandtner M，Weichselbaum L，et al. ANDRUBIS-1，000，000 apps later: A view on current Android malware behaviors [C] //2014 3rd International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security(BADGERS). IEEE，2014：3－17.

[34] 龚明明. 基于 SVM-KNN 的 Android 应用安全检测研究 [D]. 南京: 南京邮电大学，2018.

[35] Gu G F，Porras P，Yegneswaran V，et al. BotHunter: Detecting malware infection through IDS-driven dialog correlation [C] //Proceedings of the 16th USENIX Conference on Security Symposium. USENIX Association，2007：167－182.

[36] Gu G F，Zhang J J，Lee W K. BotSniffer: Detecting botnet command and control channels in network traffic [C] //Proceedings of the 15th Annual Network and Distributed System Security Symposium. The Internet Society，2008.

[37] Gu G F，Perdisci R，Zhang J J，et al. Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection [C] //Proceedings of the 17th USENIX Conference on Security Symposium. USENIX Association，2008：139－154.

[38] Zhao D，Traore I，Sayed B，et al. Botnet detection based on traffic behavior analysis and flow intervals [J]. Computers and Security，2013，39：2－16.

[39] Wang K C，Huang C Y，Lin S J，et al. A fuzzy pattern-based filtering algorithm for botnet detection [J]. Computer Networks，2011，55(15)：3275－3286.

[40] Sharifnya R，Abadi M. Dfbotkiller: Domain-flux botnet detection based on the history of group activities and failures in DNS traffic [J]. Digital Investigtion，2015，12(12)：15－26.

[41] Arora A，Garg S，Peddoju S K. Malware detection using network traffic analysis in Android based mobile devices [C] //2014 8th International Conference on Next Generation Mobile Apps，Services and Technologies. IEEE，2014：66－71.

[42] Taylor V F，Spolaor R，Conti M，et al. APPScanner: Automatic fingerprinting of smartphone Apps from encrypted network traffic [C] //2016 IEEE European Symposium on Security and Privacy(EuroS&P). IEEE，2016：439－454.

[43] Conti M，Mancini L V，Spolaor R，et al. Analyzing Android encrypted network traffic to identify user actions [J]. IEEE Transactions on Information Forensics and Security，2016，11(1)：114－125.

[44] Kwon J，Lee J，Lee H，et al. Psybog: A scalable botnet detection method for large-scale DNS traffic [J]. Computer Networks，2016，97：48－73.

[45] Huang J J，Zhang X Y，Tan L，et al. AsDroid: Detecting stealthy behaviors in Android applications by user interface and program behavior contradiction [C] //Proceedings of the 36th International Conference on Software Engineering. ACM，2014：1036－1046.

[46] Ren J J，Rao A，Lindorfer M，et al. Demo: ReCon: Revealing and controlling PII leaks in mobile network traffic [C] //Proceedings of the 14th Annual International Conference on Mobile Systems，Applications，and Services Companion，2016：117.

[47] Nan Y H，Yang M，Yang Z M，et al. UIPicker: User-input privacy identification in mobile applications [C] //Proceedings of the 24th USENIX Conference on Security Symposium. USENIX Association，2015：993－1008.

[48] Zaman M，Siddiqui T，Amin M R，et al. Malware detection in Android by network traffic analysis [C] //1st International Conference on Networking Systems and Security. IEEE，2015：1－5.

[49] Wang S S，Chen Z X，Zhang L，et al. TrafficAV: An effective and explainable detection of mobile malware behavior using network traffic [C] //2016 IEEE/ACM 24th International Symposium on Quality of Service(IWQoS). IEEE，2016：1－6.

[50] Arora A，Peddoju S K. Minimizing network traffic features for Android mobile malware detection [C] //Proceedings of the 18th International Conference on Distributed Computing and Networking. ACM，2017：32.

[51] Google Play Protect. 产品中心 [EB/OL]. (2019－05－02) [2019－10－07]. https://developers.google.com/android/play-protect/.

[52] Tencent. 腾讯 TAV 白皮书 [EB/OL]. (2018－11－02) [2019－10－07]. https://s.tencent.com/files/tav-white-paper.pdf.

[53] 王闪闪. 基于网络流量的 Android 恶意应用识别方法研究 [D]. 济南: 济南大学，2018.