Android 恶意软件检测方法研究综述

作者	李江华，邱晨
机构	江西理工大学 信息工程学院
DOI	10.3969/j.issn.1001-3695.2018.01.0015
基金项目	国家自然科学基金资助项目（61463021，61762046）；江西省教育厅科技项目（GJJ160599，GJJ170516）
预排期卷	《计算机应用研究》 2019 年第 36 卷第 1 期
摘要	基于 Android 系统恶意软件检测的全流程，对比和分析了国内外的研究现状和进展，从样本获取的角度介绍了标准化数据样本的来源及作用，从特征选择的角度阐述了特征选择应遵循的原则；重点从检测方法的角度对比和分析了各种检测方法的优缺点，同时总结和归纳了特征数据集筛选方法以及实验结果评估方法。最后结合实际应用和需求，展望了未来 Android 恶意软件检测方法的研究和发展方向。
关键词	恶意软件检测；特征；机器学习；混淆矩阵
作者简介	李江华（1976-），男，河南新野人，博士，主要研究方向为信息安全、语义 Web、大数据分析与处理；邱晨（1992-），男（通信作者），江西九江人，硕士，主要研究方向为信息安全，机器学习（1805661106@qq.com）.
中图分类号	TP309.2
访问地址	http://www.arocmag.com/article/02-2019-01-067.html
投稿日期	2018 年 1 月 2 日
修回日期	2018 年 2 月 28 日
发布日期	2018 年 4 月 3 日
引用格式	李江华，邱晨. Android 恶意软件检测方法研究综述[J/OL]. 2019, 36(1). [2018-04-03]. http://www.arocmag.com/article/02-2019-01-067.html.

Android 恶意软件检测方法研究综述 *

李江华，邱　晨†

(江西理工大学 信息工程学院，江西 赣州 341000)

摘　要：基于 Android 系统恶意软件检测的全流程，对比和分析了国内外的研究现状和进展，从样本获取的角度介绍了标准化数据样本的来源及作用，从特征选择的角度阐述了特征选择应遵循的原则；重点从检测方法的角度对比和分析了各种检测方法的优缺点，同时总结和归纳了特征数据集筛选方法以及实验结果评估方法。最后结合实际应用和需求，展望了未来 Android 恶意软件检测方法的研究和发展方向。

关键词：恶意软件检测；特征；机器学习；混淆矩阵

中图分类号：TP309.2　　doi: 10.3969/j.issn.1001-3695.2018.01.0015

Survey of Android malware detection methods

Li Jianghua, Qiu Chen†

(*School of Information Engineering Jiangxi University of Science & Technology*, *Ganzhou Jiangxi* 341000, *China*)

Abstract: Based on the whole process of malware detection in the Android system, this paper compared and analyzed the status and progress of the research at home and abroad, introduced the source and function of the standardized data samples from the point of view of sample acquisition, and expounded the principles to be followed in the feature selection. This paper compared and analyzed the advantages and disadvantages of various detection methods from the point of view of detection methods, and summed up and summarized the selection methods of feature data sets and the evaluation methods of experimental results. Finally, it prospected the research and development direction of the future Android malware detection method.

Key words: malware detection; feature; machine learning; confusion matrix

0　引言

Android 系统由 Google 公司在 2007 年 11 月 5 日正式发布，作为一款基于 Linux 内核的操作系统，其具备了开源、自由的特性。这使得 Android 系统以极快的速度超越 Symbian 系统，成为市场占有量最大的智能移动设备操作系统。然而，在其备受广大 App 开发者和用户欢迎的同时，也成为恶意犯罪者的首选目标。随着 2010 年 9 月第一款针对 Android 系统的恶意应用 FakePlayer[1]的出现，越来越多的恶意应用被发现。北卡州立大学的 Zhou 和 Jiang[1]在其后的一年多的时间里发现了几千个恶意应用，并从中筛选出了可归类为 49 个恶意应用系列的 1200 多个具有代表性的恶意应用程序。

张玉清等人[2]对恶意应用的定义是携带有恶意代码，会对系统和其他应用产生威胁的应用。针对 Android 恶意应用，各大防病毒软件公司也相继推出自己的防病毒软件，从 Android 官方市场能够下载到相应的应用程序。比较有代表性的是 AVG Antivirus、Lookout Security &Antivirus、Norton Mobile Security 和 TrendMicro Mobile Security。但是蒋旭先等人[1]通过实验证明这些防病毒软件的检测效果并不能令人满意：最好的情况下，Lookout 在 39 个系列中检测到 1003 个恶意软件样本，检测率为 79.6%；而最坏的情况下，诺顿(Norton)在 36 个系列中检测到最少的恶意样本仅 254 个，检测率为 20.2%。这个或许就是商业软件的弊端，Rastogi 等人[3]的实验也证明了这一点。他们利用自己构建的框架对包含 AVG、Lookout、Kaspersky 等 10 个公司开发的 Android 防病毒软件进行了系统的分析，不同的是，他们的实验更偏重于检验防病毒软件对抗恶意软件转换技术的能力。

针对 Android 恶意应用程序的检测也引起了学术界广大研究者的关注，对 Android 恶意应用程序的检测方法进行了大量研究，成果显著。本文将从样本获取、特征选择、静态特征检测、动态特征检测、动静态结合特征检测、特征数据集筛选、实验结果评估方法等方面对 Android 恶意软件的检测方法进行系统的总结与分析，并且提出了几点未来的研究方向。

收稿日期：2018-01-02；**修回日期：**2018-02-28　　**基金项目：**国家自然科学基金资助项目（61463021，61762046）；江西省教育厅科技项目（GJJ160599，GJJ170516）

作者简介：李江华（1976-），男，河南新野人，博士，主要研究方向为信息安全、语义 Web、大数据分析与处理；邱晨（1992-），男（通信作者），江西九江人，硕士，主要研究方向为信息安全，机器学习（1805661106@qq.com）.

1　样本获取

研究人员要研究 Android 恶意软件检测方法，其在进行实验时必要做的第一件事情就是选定基于所提出的检测系统的数据样本。Android 恶意软件检测是一个比较新的领域，最初，研究人员没有一个可以用于研究的固有的、标准的样本数据集。甚至很多人倾向于自己编写恶意软件，然后对自己的恶意软件进行评估检测。有一些研究人员尝试通过一些分享 Android 恶意软件的网站（如 VirusShare[4]，Contagio[5]等）收集样本，魏理豪等人[6]的研究中所使用的恶意软件样本就来源于 VirusShare.com。有限的恶意软件样本导致系统的检测结果达不到预期效果，这是研究人员不得不面对的问题。

2012 年 MalGenome[1]数据样本发布，共包含 1260 个恶意软件样本，分为 49 个不同恶意软件系列。这个数据样本极具价值，其可用性和易得性使大多数研究人员的实验设想得以付诸现实。通过阅读相关文献发现，仅 2012 年当年就有 4 项研究成果的出现与该数据样本相关。2014 年 3 月有 Yerima 等人[7]，F-Secure[8]，Gianazza 等人[9]，Ham and Lee[10]，Ham 等人[11]，Deshotels 等人[12]，Seo 等人[13]在其研究过程中使用了 MalGenome。

然而，随着技术的发展，恶意软件为避开检测也改变了其形态和感染技术。因此，研究人员必须更新数据样本以开发出更有效的检测系统。通过在 2014 年引入 DREBIN[14]，满足了这一需求。DREBIN 是从 2010 年 8 月至 2012 年 10 月期间收集的来自 179 个不同系列的 5560 个恶意软件的集合。通过阅读相关文献发现，在 2014 年，该数据样本被用于 19 项研究工作。这表明研究人员都热衷于使用标准化数据样本。

2　特征选择

最早出现的恶意软件检测方法之一是基于签名的检测。这种方法中，检测系统构建恶意软件的唯一签名，通过签名与收集到的数据库相匹配来检测恶意软件。然而，只需要一些细微的改变就可能导致一个新的恶意软件变体的出现，并且能够绕过基于签名的检测方法[15]。DroidAnalytics[16]就是用基于签名的检测方法开发的，它能够自动收集、提取和分析 Android 应用程序文件的签名。它使用 Java 代码和类作为签名来检测已知的恶意软件，但是，对于未知的恶意软件却无法检测到。秦中元等人[17]设计了一种多级签名匹配算法以检测 Android 恶意应用，它基于恶意样本库来实现签名匹配，这需要实时地更新、补充恶意样本库，这项工作庞大而繁杂。然而，Android 恶意软件的变体数量一直是稳步上升的，有效的检测它们至关重要。

为了克服基于签名的检测方法的局限性，研究人员转而采用机器学习方法进行分类检测。用已经收集到的数据训练机器学习算法，以检测异常达到发现新的恶意软件的目的。2009 年，Garcia-Teodoro 等人[15]在他们的研究中使用 k-means 算法进行恶意软件检测，达到了 83.79%的检测精度。同样，2012 年，Sahs 和 Khan[18]做了一个实验，他们在自己的数据集上应用了支持向量机（SVM）算法，对算法进行了训练，达到了 93%的精度。机器学习算法被证明在恶意软件检测领域具有极大的作用。

在机器学习算法中，选择适当的特征是实验中最重要的步骤之一。然而，在具体选用什么特征进行实验研究的问题上，广大研究人员未能达成一致意见。Android 应用程序有许多特征，从 Android 应用市场上下载到的应用程序是 APK 文件，它是一种压缩文件，其中包含安装和运行应用程序所需的全部代码和数据。APK 文件由 APK Builder 合成，可以使用 APK Tool 对其进行反编译。反编译成功后可得到.dex、.xml、res、META-INF 等四类文件，如图 1 所示。

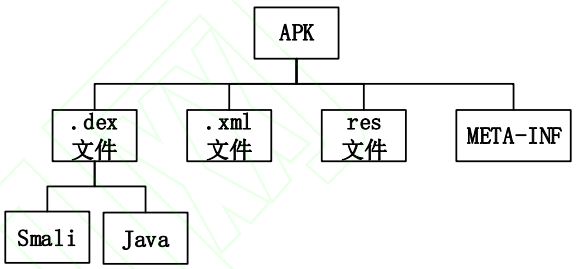

图 1　APK 文件结构

DEX 是 Dalvik Executable 的缩写，DEX 文件是 Dalvik 虚拟机可执行的字节码文件，拥有独立的指令集[19]，可被反编译成 java 代码文件和 smali 代码文件两种。

XML 是 extensible Markup Language 的缩写，XML 文件是用于定义用户界面布局和值的文件。比较特殊的是 AndroidManifest.xml 文件，它是包含应用程序执行所需的清单文件，包含了 Android 框架为运行应用程序所需要的所有参数，这包括权限、包名、组件和 API 等信息。

RES 是 Resource 的缩写，res 文件是资源文件，是包含应用程序执行所需资源如图像、视频、音频等的文件。

META-INF 存放签名信息。为了发布 APK，开发人员对初创建的未签名的 APK 文件使用 Jarsigner 利用私钥进行自签名，然后将开发人员的签名和公钥添加到 APK 文件中，用来保证 APK 包的完整性。

这些组成 APK 包的所有组成部分都有可能成为能够选用的特征。同时，Android 应用程序在移动设备上运行时的部分行为也有可能成为能够选用的特征。

在静态特征中，研究人员一直很重视 Android 应用的权限，这说明研究者们非常了解这个特征的有效性，并根据推理选择它们作为研究对象。权限是 Android 应用程序针对攻击者的第一道防线，这是将其作为特征选择的合理原因。

Java 代码也是使用比较广泛的一个静态特征。任何一个 Android 应用软件都是由 Java 代码编写而成，恶意活动的实现更加离不开 Java 代码，所以可以认为 Java 代码是恶意软件恶意活动的根源，它成为了研究工作的重点关注对象。然而，分

析 Java 代码比分析权限困难的多，因为恶意软件开发人员会使用模糊处理、加密等各种技术来避免可用的检测方法[20]。所以，研究人员一直在开发更合理的方法来检测 Java 代码中的威胁[12; 21; 22]。

在动态特征中，网络流量特征的使用引起了较多的关注。Feizollah 等人[23]使用网络流量特征来检测移动恶意软件。作者选择了 TCP 大小、连接持续时间和 GET/POST 参数的数量三个网络特征，并且提供了使用每个参数的理由。特征选择合适合理，检测率达到 99.94%。CREDROID[24]是另一个使用网络流量分析进行 Android 恶意软件检测的例子。

然而，相对于其他动态特征，系统调用是研究人员使用最频繁的一个。Android 应用程序的所有活动都是通过系统调用完成的，所以选择它作为一个特征是适合的。

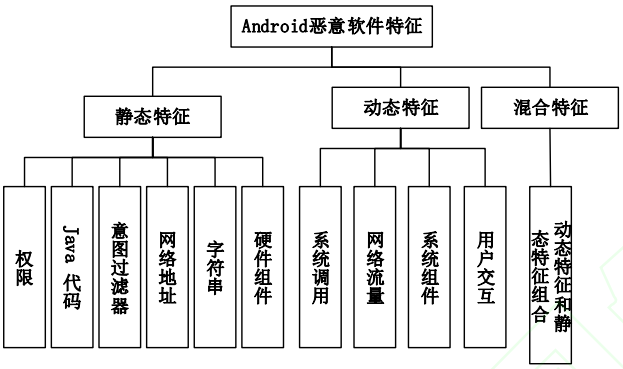

图 2　Android 恶意软件特征的种类

特征的种类和每种特征的子类如图 2 所示。虽然广大研究人员在研究中选用的实验特征多种多样，但是通过阅读有关 Android 恶意软件检测中特征选择的相关文献并进行对比，能够发现这种选择都遵循了合理化选择的原则。

3　静态特征检测

静态特征，是指已经存在的、不会发生变化的特征。对于 Android 应用程序来说，组成一个 APK 文件的内容就属于不会发生变化的东西，因为它已经是一个完整的应用程序，且已经被发布在应用市场上了。APK 文件有各种特征，如权限、Java 代码、意图过滤器、网络地址、字符串和硬件组件等。这些特征都曾被研究人员作为 Android 恶意软件检测的实验对象。

3.1　Android 权限特征

Android 应用程序在被下载到移动设备上安装的时候，它会向用户提供所请求权限的列表。在用户授予相应权限后，应用程序才能安装到移动设备上。Felt 等人[25]的研究指出在 Android 2.2 中有 134 个官方 Android 权限。Google 将这些权限分为四组，即正常，危险，签名，和签名或系统[26]。

研究人员在分析 Android 权限时采用了不同的方法。使用权限特征检测应用程序的有很多，比如文献[27~29]。他们根据可能的风险评分使用概率生成模型，定量安全风险评估对应用程序进行排名。何文才等人[30]对权限信息使用数量统计方法进行了两次去冗余，而后对最终获得的权限信息进行排序。也有很多如文献[7，31，32]的研究只是简单地提取了权限特征并利用机器学习方法来检测应用程序。这种方法被证明是有缺陷的。为解决这个问题，许艳萍等人[33]在其研究中对提取的权限特征使用信息增益算法进行优化选择，同时对所使用的朴素贝叶斯算法进行了改进，其最终检测的准确率达 90%以上。这种方式对提高检测率具有一定效果，但并不能令人满意。

权限是最常用的静态特征，如 3 节中所述，权限是攻击者的第一个障碍。尽管 Java 代码中包含恶意代码，但代码中的一些 API 调用也依然需要被调用的权限[34]。受权限保护的 API 调用是 Android 操作系统的安全特征的一部分。例如，在发送消息或访问网络之前，Android 系统会检查应用程序是否有相应权限[25]。基于这种情况，研究人员的重点是比其他静态特征更多的权限，以根据需要的权限检测恶意软件。

周裕娟等人[35]在其研究中不仅提取了应用程序的权限特征作为机器学习分类器的输入，同时将具有权限提升攻击威胁的特征作为分类依据之一。Moonsamy 等人[36]，Huang 等人[37]认为仅分析所请求的权限不足以检测恶意应用程序。他们分析了除请求的权限之外的使用权限，以期更准确的检测出恶意软件。

3.2　Android Java 代码特征

一般情况下，开发人员用 Java 语言编写 Android 应用程序，通过 Java 编译器编译生成在 Java 虚拟机（JVM）上运行的类文件（class 文件），然后使用 Android SDK 中包含的 DX 转换器将类文件转换为 DEX 文件。DEX 文件是一种特殊格式的文件，只能在 Dalvik 虚拟机上运行，Dalvik 虚拟机是 Android 操作系统独有的。Google 在 4.4 版本中引入了 ART（Android runtime）。虽然 ART 提供了提前编译、改进垃圾收集、开发和调试改进等新功能，但是 Dalvik 仍然是 Android 操作系统[38]的默认运行环境。

研究人员对 Java 代码使用了各种分析方法。为实现 Android 恶意软件的高效检测，李挺等人[39]提出了一种形式化描述和分析 Android 恶意代码的方法。一些研究人员使用应用程序编程接口即 API 来检测恶意软件如文献[3，12，16，40，41]。每个 Android 应用程序都需要具有 API 调用才能与设备进行交互。例如，对操作系统的电话管理器进行 API 调用以检索电话 ID 和用户 ID。方法中的 API 调用是有序的。研究人员认为这样的一个序列可以作为该应用程序唯一的应用程序签名。更改 API 调用的顺序是攻击者绕过检测过程所采用的策略，这种技术被称为代码混淆技术。秦中元等人[17]提出多级签名匹配算法，其第一级签名就与 API 调用直接相关。祝小兰等人[42]在其研究中使用了与敏感权限相关的系统 API 调用函数图序列。

分析 Java 代码的控制流程是研究人员采用的另一种方法，这在如文献[43~47]的研究中都有体现。王志强等人[48]在其 Android 恶意行为检测算法的研究中提出用有限长度的 API 函数调用序列在一定程度上表征 Android 应用程序的行为，同时

为了完整表征 Android 应用程序行为，在 API 函数调用序列的基础上加入控制流序列，即编程过程中调用某个类的方法的序列。虽然攻击者可以改变 API 调用的顺序或重命名 API 调用以避开检测系统，但是 Java 代码的流程并没有改变，研究人员可以用它开发更强大的检测系统。

3.3 其他静态特征

除了权限和 Java 代码之外，一些研究人员还分析了如 Intent 过滤器、网络地址、字符串、硬件组件等在内的其他几个静态特征。

Intent 过滤器是清单文件中描述的元素之一。它是与某一个操作相关的抽象信息，通过它可以推断应用程序的意图。例如，选择联系人、拍照、拨号等。根据 Intent 过滤器，Android 系统执行相应的活动。研究人员已经对恶意软件的检测使用 Intent 过滤器，因为攻击者命令恶意软件向他们传送隐私数据的过程需要在 AndroidManifest.xml 文件的 Intent 过滤器部分存在意图。在 DroidMat[34]中，提取并分析了包括 Intent 过滤器在内的 Android 文件的各种特征。作者使用了几种机器学习算法，如 k-means，kNN 和朴素贝叶斯来开发恶意软件检测系统。DroidMat 的检测同时显示出了超过类似系统的改进。Luoshi Z 等人[49]发表了一个名为 A3 的系统，它考虑了 Android 安装文件中包含 Intent 过滤器在内的几个特征，构建了一个表示 Java 代码执行流程的调用图。然后使用 A*算法来确定随后显示恶意软件行为的最短路径。DREBIN[14]提出了广义的静态分析法。该方法收集 Android 安装文件的静态特征，包括 Intent 过滤器。他们使用支持向量机（SVM）的机器学习法算法以达到检测目的。其实验结果表明，DERBIN 检测到恶意软件的误报率很小，准确率达到 94%。

网络地址也是一个可用的特征。攻击者指示恶意软件与他们联系并报告状态后发送用户个人数据。为此，攻击者将服务器的地址嵌入到恶意软件的代码中，那个服务器被叫做命令和控制服务器（C&C 服务器）。研究人员在 Android 安装文件的代码中查找 C&C 服务器的网络地址或 IP 地址。Luoshi Z 等人[49]和 Arp 等人[14]将网络地址作为其检测系统中的静态特征之一。

Sanz 等人[32]提出经典恶意软件检测中广泛使用的技术之一是分析文件中可用的字符串。他们通过提取 Android 文件中的所有的如应用程序中的菜单或应用程序连接的服务器地址一样的可打印字符串，为 Android 恶意软件采用了相同的技术。他们使用向量空间模型（VSM）[50]将字符串表示为多维空间中的向量。然后，使用距离测量，如曼哈顿距离、欧氏距离和余弦相似度来计算数据的异常。他们使用了 666 个 Android 应用程序样本进行实验，实验精度为 83.51%，TPR 为 94%。

DERBIN[14]除使用 Intent 过滤器外还使用硬件组件作为静态特征。作为 AndroidManifest.xml 文件的一部分，应用程序需要有与之相配的硬件组合才能运行，比如相机或 GPS。所请求的硬件的组合意味着应用程序可能的有害性，例如 GPS 访问意味着恶意软件将用户的位置报告给攻击者。张国印等人[19]的研究将权限、API 调用、组件声明和字符串信息都作为要提取的特征，使用贝叶斯网络分类器模型进行恶意软件检测。

除了上述静态特征以外，其他一些特征也有可能用作 Android 静态特征。Shabtai 等[51]使用诸如.apk 大小、zip 条目数、每个文件类型的文件数、每个元素名称的 XML 元素和特征的数量等特征。程运安和汪弈祥[52]在其研究中就提取了权限特征和包括图片文件的数量、XML 文件的数量以及四大 Android 组件的数量在内的资源文件特征。

4 动态特征检测

动态特征，就是会产生变化的特征，一般与操作系统或网络连接相互的应用程序的行为都可以认为是动态特征。使用较多的动态特征有两种：系统调用和网络流量。每个应用程序都要通过发出如读取、写入和打开等的系统调用来向操作系统请求所需的资源和服务。应用程序倾向于连接到网络已发送和接收数据，接收更新后恶意泄露个人数据给攻击者。监控移动设备的网络流量是一种行之有效的检测恶意软件的方法。

4.1 Android 系统调用特征

Linux 内核中有 250 多个系统调用可以在 Android 中使用[53]。分析系统调用可以检测应用程序的异常行为[23]。因为不能直接与 Android 操作系统交互，应用程序使用系统调用来执行特定的任务，如读取、写入和打开。在用户模式下，发出系统调用时，Android 操作系统转向内核模式来执行所需的任务。阅读文献发现，在动态特征检测中，系统调用是最受欢迎的特征。诸如文献[53~57]都提取并分析系统调用特征以检测恶意软件。蔡志标和彭新光[58]在其研究中提出使用应用程序的系统调用名和相对应的频数信息使用 kNN 分类算法进行恶意行为识别。

4.2 Android 网络流量特征

无论是正常程序或恶意程序，大多数都需要网络连接。Yajin Zhou 和 Xuxian Jiang[1]指出，他们收集的 Android 恶意样本中有 93%需要网络连接才能连接到攻击者，从 C&C 服务器接收命令。另外 Sarma 等[31]在 2012 年发布了一项研究成果，他们分析了 Android 文件的权限，检查了超过 15 万个应用程序，发现正常应用程序的 68.5%需要网络访问，93.8%的恶意应用程序需要网络访问。无独有偶，Yerima 等人[7]分析了 2000 个应用程序的权限，超过 93%的恶意应用程序要求网络连接。显然，大多数应用程序请求网络访问，特别是恶意应用程序。因此，研究人员应该专注于分析网络流量以进行有效的 Android 恶意软件检测。2016 年，Jyoti Malik 和 Rishabh Kaushal[24]提出一种名为 CREDROID 的方法，通过在离线状态下对网络流量日志进行深入分析，根据其域名服务器（DNS）查询及其传输到远程服务器的数据，识别恶意应用程序模式。

尽管网络流量特征在移动恶意软件检测中很有效，但并没有吸引到其他动态特征的研究人员的关注。因为，利用网络流量，面临着在数据集中处理大量网络记录的挑战，甚至有可能

多达一百万条记录。此外，分析收集的网络流量需要对网络架构有深入了解。

4.3 其他动态特征

除了上述的系统调用和网络流量两个主要的动态特征外，研究人员也一直在使用其他动态特征。比如，系统组件，用户交互等。

移动设备中具有的与个人计算机相似的组件，如 CPU 和内存，就是系统组件。一些研究人员利用系统组件研究 Android 恶意软件的检测。在 MADAM[59]中，研究人员分析了被视为操作系统内核级的移动设备的 CPU 使用率，可用内存和运行进程。此外，还检查了用户/应用程序级特征，如设备的蓝牙和 Wi-Fi 状态。采集到的数据被用于训练 kNN 算法。2013 年，STREAM[60]被引入到 Android 操作系统。它收集有关系统组件的数据，如 cpuUser，cpuIdle，cpuSystem，cpuOther，memActive 和 memMapped。随后使用机器学习算法训练系统，以便检测 Android 恶意软件。Hyo-Sik 和 Mi-Jung[61]以及 Hoffmann 等人[62]也使用系统组件作为动态特征。

理论上，任何一个用户都有可能成为恶意应用程序的受害者，分析用户与应用程序之间的交互式恶意软件检测的可能解决方案之一。PuppetDroid[9]捕获用户与设备的交互，比如按下按钮，缩放和浏览页面等。检查了安装有 15 个 Android 应用程序的系统，期望在捕获与恶意软件相关的用户交互后，系统会寻找类似的用户交互来检测恶意应用程序。Dynodroid[63]的研究是基于用户交互分析开发的另一个系统，它收集用户的活动，如点击屏幕，长按和拖动。Dynodroid 的检测分析了 50 个 Android 应用程序，结果发现了 Android 应用程序的 bug。

5 动静态结合特征检测

动静态结合特征，有些研究人员称之为混合特征，就是在检测系统中一起使用的静态和动态特征。混合特征应该是最全面的特征，不仅涉及审核 Android 应用程序安装文件，还有在运行时分析应用程序的行为。Blasing 等人[64]开发了 AASandBox，分析静态和动态特征，它从 APK 文件中提取权限和 Java 代码，将其作为静态特征，然后安装应用程序，记录系统调用，将其作为动态特征。潘夏福[65]在其研究中采用了动静态结合的方式获取特征，他选取了权限和系统调用函数作为静态特征，通过模拟运行，使用工具软件获取系统调用日志和网络数据等信息作为动态特征。一些选择混合特征的类似研究还有文献[45，66，67]。

比较三种不同类型的特征，静态特征检测是最方便的，所选用特征最容易提取，但是难以解决代码混淆问题。动态特征检测比静态特征检测更全面，但是动态特征相对难以提取，需要 root 设备。混合特征检测，选用动静态结合特征，可以认为是最全面的特征集合，但是问题同样明显，特征提取过程复杂，静态特征和动态特征都必须要提取，增加了检测难度，也更费资源。

6 特征数据集筛选

为了得到更好的实验结果，一些研究人员对所提取的原始特征集进行了一定的筛选。Jensen 和 Shen[68]提出特征筛选和排序是基于开发的算法选择提供最有用和最重要的特征的原始特征的子集的任务。该算法接收收集的数据集并使用数学计算方法来对数据集中的特征进行排序。Shabtai 和 Elovici[69]使用特征排序算法从 88 个收集的特征中选择特征的子集。他们设法从原始数据集中选择前 10，20 和 50 个特征。Shabtai 等人[70]分析 Android 应用的网络流量，他们使用特征选择算法来选择网络流量数据中大量特征中最有用的特征。张国印等人[19]在其研究中提取了多种特征，而后，先后使用卡方统计方法和关联规则进行特征筛选，去除无影响的和冗余的特征。王聪等人[71]在其研究中对提取的权限特征利用了关联规则的 Apriori 算法（连接和剪枝）对权限特征集进行处理。张思琪[72]在其研究中提取了恶意代码行为及所需权限特征作为研究对象，而后使用卡方检验方法对提取的特征进行预处理，剔除与所需特征值不相关的行为。同样使用卡方检验方法去冗余的还有张锐和杨吉云[73]。如 4.1 节所述，许艳萍等人[33]在其研究中使用了信息增益算法对特征进行优化选择。何文才等人[30]在其研究中多次对所提取的权限特征信息去冗余，其中之一是利用权限使用频率的平方差去冗余。卢文清等人[74]设法将提取的特征形成特征向量，而后使用主成分分析（PCA）法对其进行降维处理，达到去除冗余信息的目的。

无论使用什么方法对特征数据集进行筛选，其最终目的都是为了去除对实验结果无影响或影响很小的数据，达到提高检测准确率和效率的目的。

7 实验结果评估方法

研究人员用检测恶意软件的准确程度来评估所提出系统的有效性，而对于如何评估检测恶意软件的准确程度，研究人员使用了各种各样的方法。

实验结果可以用被称为混淆矩阵的表格的形式表示，如表 1 所示，它有四种参数类型。TP（true positive）指的是将正确分类为正例的数量，这意味着系统检测非恶意软件为非恶意。TP 越大，结果越好。FP（False Positive）指的是错误分类为正例的数量，这意味着系统将恶意软件检测为非恶意软件。FP 越小，系统就越准确。TN（True Negative）指的是正确分类为负例的数量，这意味着系统检测恶意软件为恶意成功。FN（False Negative）指的是错误分类为负例的数量，这意味着系统将非恶意软件检测为恶意软件。

表 1 混淆矩阵

	Actual positive	Actual negative
Predicted positive	TP	FP
Predicted negative	FN	TN

混淆矩阵可以很直观的显示出系统的检测结果，为了进一步评估系统的性能，以表 1 所示的混淆矩阵为基础，研究人员推算出了更多的评价指标如下。

a）TPR,真正率。如 4.3 节提及某研究的实验结果，TPR 为 94%。这里的 TPR 也称为灵敏度（sensitivity）或召回率（recall），代表的是检测为正的正样本数占实际正样本数的比例，计算公式如下：

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{1}$$

b）FNR，假负率。代表的是检测为负的正样本数占实际正样本数的比例，计算公式如下：

$$FNR = \frac{FN}{P} = \frac{FN}{TP + FN} \tag{2}$$

c）FPR，假正率。代表的是检测为正的负样本数占实际负样本数的比例，计算公式如下：

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{3}$$

d）TNR,真负率,也称为特异度（Specificity）。代表的是检测为负的负样本数占实际负样本数的比例，计算公式如下：

$$TNR = \frac{TN}{N} = \frac{TN}{FP + TN} \tag{4}$$

e）Accuracy，准确率。系统对整个样本的检测能力，代表系统将正样本检测为正样本，负样本检测为负样本的能力。计算公式如下：

$$\text{Accuracy} = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + FP + FN + TN} \tag{5}$$

f）Error rate，错误率。与准确率相对，表示系统将正样本检测为负样本，负样本检测为正样本的情况。计算公式如下：

$$\text{Error Rate} = \frac{FP + FN}{P + N} = \frac{FP + FN}{TP + FP + FN + TN} \tag{6}$$

g）Precision，精确度。代表的是实际为正的样本数占检测为正的样本数的比例。计算公式如下：

$$\text{Precision} = \frac{TP}{TP + FP} \tag{7}$$

h）F-measure，精确度和召回率的调和平均值，且更接近于两者中较小的那个。研究人员在其研究工作中较少采用。计算公式如下：

$$\text{F- Measure} = \frac{2 \times \text{Precision} \times \text{Recall Rate}}{\text{Precision+ Recall Rate}} \tag{8}$$

i）ROC。ROC 是 receiver operating characteristic（受试者工作特性）的缩写，ROC 曲线，其横坐标为 FPR，纵坐标为 TPR。它表明，当内部阈值发生变化时，检测率会如何变化。AUC（Area Under the Curve）通常与 ROC 曲线成对出现，是 ROC 曲线下的区域面积。它的值在 0~1 变化。当它接近 1 时，系统性能更好。

8 趋势与展望

Android 恶意软件检测相关技术经过广大研究人员的不懈努力，已经取得了一定的成绩，但是能够检测出现有恶意软件的同时还能够应对不断出现的新型恶意软件的成果还未曾出现，因此，针对 Android 恶意软件检测方法的研究依然会是一项需要研究人员继续努力的持久艰巨的任务，未来值得研究的方向如下。

a）应用程序本身是由代码构成，恶意软件的一切行为，最终都是通过代码实现。笔者认为，新型恶意软件的恶意代码极有可能散布在构成 Android 应用程序的各个文件中，而不是单纯的完整的存在于某一个或某几个文件里面。所以，针对应用程序的代码进行研究达到检测恶意软件的目的是值得研究人员继续深入研究的且能够取得较大成果的方向。

b）Android 应用程序元数据也就是用户在下载和安装应用程序之前看到的信息，如应用程序描述，请求的权限，评级和开发人员的信息等。它不属于静态特征，也不属于动态特征，因为它与应用程序本身无关。这是目前并未被广泛使用的一类特征，但是比较容易提取到。笔者认为，针对 Android 应用程序元数据特征进行深入研究是可取的。

c）技术的发展随之而来的是新型恶意软件技术的升级。研究人员需要检测新系统的新型恶意软件，而不是旧的。针对新型恶意软件的挑战提出有效的检测方案，不断改进检测方法，开发新的检测工具。更好的 Android 应用动态行为的模拟工具应该被开发，以便检测出恶意应用潜在的安全威胁并给出相应的安全警报。针对很多规模较大的检测系统的资源占用问题，云计算和大数据资源应该被充分利用以便更好更快地检测 Android 恶意软件。

9 结束语

本文结合 Android 恶意软件检测流程，对 Android 恶意软件检测方法进行了分析与总结。然而，提取什么样的特征进行检测，什么是 Android 恶意软件检测的最佳方法，目前研究界并没有一个共同的、大家都认可的解决方案。相对认可的意见是，研究人员都肯定了机器学习算法在 Android 恶意软件检测的研究中的积极作用。同时，随着新型恶意软件的增多，应该不断更新恶意软件样本库。

参考文献：

[1] Jiang Xuxian, Zhou Yajin. Dissecting Android malware: characterization and evolution [C]// Proc of IEEE Symposium on Security and Privacy. 2012: 95-109.

[2] 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.

[3] Rastogi V, Chen Yan, Jiang Xuxian. Catch me if you can: evaluating Android anti-malware against transformation attacks [J]. IEEE Trans on Information

Forensics & Security, 2013, 9 (1): 99-108.

[4] VirusShare [EB/OL]. https://virusshare. com/.

[5] Contagio [EB/OL]. http://contagiodump. blogspot. com/.

[6] 魏理豪，艾解清，邹洪，等. Android 恶意软件的多特征协作决策检测方法 [J]. 计算机工程与应用, 2016, 52 (20): 5-13.

[7] Yerima S Y, Sezer S, Mcwilliams G. Analysis of Bayesian classification-based approaches for Android malware detection [J]. Information Security Iet, 2014, 8 (1): 25-36.

[8] Backdoor: Android/dendroid. a [EB/OL]. http://www. f-secure.com//v-descs//backdoor_android_dendroid_a. shtml.

[9] Gianazza A, Maggi F, Fattori A, *et al.* PuppetDroid: a user-centric UI exerciser for automatic dynamic analysis of similar Android applications [J]. Computer Science, 2014.

[10] You J H, Lee H W. Detection of malicious android mobile applications based on aggregated system call events [J]. International Journal of Computer & Communication Engineering, 2014, 3 (2): 149-154.

[11] You J H, Moon D, Lee H W, *et al.* Android mobile application system call event pattern analysis for determination of malicious attack [J]. International Journal of Security & Its Applications, 2014, 8 (1): 231-246.

[12] Deshotels L, Notani V, Lakhotia A. DroidLegacy: automated familial classification of Android malware [C]// Proc of ACM SIGPLAN on Program Protection and Reverse Engineering Workshop. 2014: 1-12.

[13] Seo S H, Gupta A, Sallam A M, *et al.* Detecting mobile malware threats to homeland security through static analysis [J]. Journal of Network & Computer Applications, 2014, 38 (1): 43-53.

[14] Arp D, Spreitzenbarth M, Hubner M, *et al.* DREBIN: effective and explainable detection of android malware in your pocket [C]// Proc of Network and Distributed System Security Symposium. 2014.

[15] Garcia-Teodoro P, Diza-Verdejo J, Maciá-Fernández G, *et al.* Anomaly-based network intrusion detection: Techniques, systems and challenges [J]. Computers & Security, 2009, 28 (1-2): 18-28.

[16] Zheng Ming, Sun Mingshen, Lui J C S. Droid analytics: a signature based analytic system to collect, extract, analyze and associate android malware [C]// Proc of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. Washington DC: IEEE Computer Society, 2013: 163-171.

[17] 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.

[18] Sahs J, Khan L. A machine learning approach to Android malware detection [C]// Proc of Intelligence and Security Informatics Conference. 2012: 141-147.

[19] 张国印，曲家兴，李晓光. 基于贝叶斯网络的 Android 恶意行为检测方法 [J]. 计算机工程与应用, 2016, 52 (17): 16-23+191.

[20] Petsas T, Voyatzis G, Athanasopoulos E, *et al.* Rage against the virtual machine: hindering dynamic analysis of Android malware [C]// Proc of European Workshop on System Security. 2014: 1-6.

[21] Suarez-Tangil G, Tapiador J E, Peris-Lopez P, *et al.* Evolution, detection and analysis of malware for smart devices [J]. IEEE Communications Surveys & Tutorials, 2014, 16 (2): 961-987.

[22] Lu Long, Li Zhichun, Wu Zhenyu, *et al.* CHEX: statically vetting Android apps for component hijacking vulnerabilities [C]// Proc of ACM Conference on Computer and Communications Security. 2012: 229-240.

[23] Feizollah A, Anuar N B, Sallen R, *et al.* A study of machine learning classifiers for anomaly-based mobile botnet detection [J]. Malaysian Journal of Computer Science, 2013, 26 (4): 251-265.

[24] Malik J, Kaushal R. CREDROID: Android malware detection by network traffic analysis [C]// Proc of ACM Workshop on Privacy-Aware Mobile Computing. 2016: 28-36.

[25] Felt A P, Chin E, Hanna S, *et al.* Android permissions demystified [C]// Proc of ACM Conference on Computer and Communications Security. 2011: 627-638.

[26] Permission [EB/OL]. http://developer. android. com/guide/topics/manifest/permission-element. html.

[27] Peng Hao, Gates C, Sarma B, *et al.* Using probabilistic generative models for ranking risks of Android apps [C]// Proc of ACM Conference on Computer and Communications Security. 2012: 241-252.

[28] Wang Yang, Zheng Jun, Sun Chen, *et al.* Quantitative security risk assessment of Android permissions and applications [C]// Proc of IFIP Wg 11. 3 Conference on Data and Applications Security and Privacy. 2013: 226-241.

[29] Pandita R, Xiao Xusheng, Yang Wei, *et al.* WHYPER: towards automating risk assessment of mobile applications [C]// Proc of USENIX Conference on Security. 2013: 527-542.

[30] 何文才，闫翔宇，刘培鹤，等. 基于最小距离分类器的 Android 恶意软件检测方案 [J]. 计算机应用研究, 2017, 34 (7): 2184-2188.

[31] Sarma B P, Li Ninghui, Gates C, *et al.* Android permissions: a perspective combining risks and benefits [C]// Proc of ACM Symposium on Access Control MODELS and Technologies. 2012: 13-22.

[32] Sanz B, Santos I, Laorden C, *et al.* PUMA: permission usage to detect malware in Android [M]. Berlin: Springer, 2013: 289-298.

[33] 许艳萍，伍淳华，侯美佳，等. 基于改进朴素贝叶斯的 Android 恶意应用检测技术 [J]. 北京邮电大学学报, 2016, 39 (2): 43-47.

[34] Wu Dongjie, Mao Chinghao, Lee H M, *et al.* DroidMat: Android malware detection through manifest and API calls tracing [C]. Information Security, 2012: 62-69.

[35] 周裕娟，张红梅，张向利，等. 基于 Android 权限信息的恶意软件检测 [J]. 计算机应用研究, 2015, 32 (10): 3036-3040.

[36] Moonsamy V, Rong Jia, Liu Shaowu. Mining permission patterns for contrasting clean and malicious android applications [J]. Future Generation Computer Systems, 2014, 36 (3): 122-132.

[37] Huang Chunying, Tsai Yiting, Hsu Chunghan. Performance evaluation on permission-based detection for Android malware [M]. Berlin: Springer,

2013: 111-120.

[38] Google. Introducing art [M]. 2014.

[39] 李挺, 董航, 袁春阳, 等. 基于 Dalvik 指令的 Android 恶意代码特征描述及验证 [J]. 计算机研究与发展, 2014, 51 (7): 1458-1466.

[40] Grace M, Zhou Yajin, Zhang Qiang, *et al.* RiskRanker: scalable and accurate zero-day android malware detection [C]// Proc of International Conference on Mobile Systems, Applications, and Services. 2012: 281-294.

[41] Yerima S Y, Sezer S, Mcwilliams G, *et al.* A new Android malware detection approach using bayesian classification [C]// Proc of IEEE International Conference on Advanced Information Networking and Applications. 2013: 121-128.

[42] 祝小兰, 王俊峰, 杜垚, 等. 基于敏感权限及其函数调用图的 Android 恶意代码检测 [J]. 四川大学学报: 自然科学版, 2016, 53 (3): 526-533.

[43] Suarez-Tangil G, Tapiador J E, Peris-Lopez P, *et al.* Dendroid: a text mining approach to analyzing and classifying code structures in Android malware families [J]. Expert Systems with Applications, 2014, 41 (4): 1104-1117.

[44] Crussell J, Gibler C, Chen Hao. Attack of the clones: detecting cloned applications on Android markets [C]// Proc of European Symposium on Research in Computer Security, 2012: 37-54.

[45] Xu Jianlin, Yu Yifan, Chen Zhen, *et al.* MobSafe: cloud computing based forensic analysis for massive mobile applications using data mining [J]. 清华大学学报: 自然科学版 (英文版) , 2013, 18 (4): 418-427.

[46] Chin E, Felt A P, Greenwood K, *et al.* Analyzing inter-application communication in Android [J]. Plant & Soil, 2011, 269 (1-2): 309-320.

[47] Jian Chen, Manar H, Alalfi, *et al.* Detecting Android malware using clone detection [J]. 计算机科学技术学报: 英文版, 2015, 30 (5): 942-956.

[48] 王志强, 张玉清, 刘奇旭, 等. 一种 Android 恶意行为检测算法 [J]. 西安电子科技大学学报, 2015, 42 (3): 8-14.

[49] Zhang Luoshi, Yan Niu, Xiao Wu, *et al.* A3: Automatic Analysis of Android Malware [C]// Proc of International Workshop on Cloud Computing & Information Security. 2013: 89-93.

[50] Baeza-Yates R A, Ribeiro-Neto B. Modern information retrieval [M]. Beijing: China Machine Press, 2011: 26–28.

[51] Shabtai A, Fledel Y, Elovici Y. Automated static code analysis for classifying android applications using machine learning [C]// Proc of International Conference on Computational Intelligence and Security. 2011: 329-333.

[52] 程运安, 汪奕祥. 基于多特征的 Android 恶意软件检测方法 [J]. 计算机工程与应用, 2017, 53 (8): 95-101.

[53] Burguera I, Zurutuza U, Nadjm-Tehrani S. Crowdroid: behavior-based malware detection system for Android [C]// Proc of ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. 2011: 15-26.

[54] Zhao Min, Ge Fangbin, Zhang Tao, *et al.* AntiMalDroid: an efficient SVM-based malware detection framework for Android [C]// Proc of International Conference on Information Computing and Applications. 2011: 158-166.

[55] Yan L K, Yin Heng. DroidScope: seamlessly reconstructing the OS and dalvik semantic views for dynamic Android malware analysis [C]// Proc of the 21st USENIX conference on Security. 2013: 29.

[56] Su X, Chuah M, Tan G. Smartphone dual defense protection framework: detecting malicious applications in Android MArkets [C]// Proc of International Conference on Mobile Ad-Hoc and Sensor Networks. 2012: 153-160.

[57] Khune R S, Thangakumar J. A cloud-based intrusion detection system for Android smartphones [C]// Proc of International Conference on Radar, Communication and Computing. 2013: 180-184.

[58] 蔡志标, 彭新光. 基于系统调用的 Android 恶意软件检测 [J]. 计算机工程与设计, 2013 (11): 3757-3761.

[59] Dini G, Martinelli F, Saracino A, *et al.* MADAM: a multi-level anomaly detector for Android malware [C]// Proc of International Conference on Mathematical Methods, Models and Architectures for Computer Network Security. [S. l.] : Computer Network Security, 2012: 240-253.

[60] Amos B, Turner H, White J. Applying machine learning classifiers to dynamic Android malware detection at scale [C]// Proc of the 9th International Wireless Communications and Mobile Computing Conference. 2013: 1666-1671.

[61] Ham H S, Choi M J. Analysis of Android malware detection performance using machine learning classifiers [C]// Proc of International Conference on ICT Convergence. 2013: 490-495.

[62] Hoffmann J, Neumann S, Holz T. Mobile malware detection based on energy fingerprints: A dead end? [C]// Proc of International Workshop on Recent Advances in Intrusion Detection. 2013: 348-368.

[63] Machiry A, Tahiliani R, Naik M. Dynodroid: an input generation system for Android apps [C]// Proc of Joint Meeting on Foundations of Software Engineering. 2013: 224-234.

[64] Blasing T, Batyuk L, Schmidt A D, *et al.* An Android application sandbox system for suspicious software detection [C]// Proc of International Conference on Malicious and Unwanted Software. 2010: 55-62.

[65] 潘夏福. 基于 KNN 算法和 K-means 算法的 Android 恶意软件检测 [J]. 电脑知识与技术, 2016 (14): 216-218.

[66] Spreitzenbarth M, Freiling F, Echtler F, *et al.* Mobile-sandbox: having a deeper look into android applications [C]// Proc of ACM Symposium on Applied Computing. 2013: 1808-1815.

[67] Eder T, Rodler M, Vymazal D, *et al.* ANANAS-a framework for analyzing Android applications [C]// Proc of International Conference on Availability, Reliability and Security. 2013: 711-719.

[68] Jensen R, Shen Qiang. Computational intelligence and feature selection: rough and fuzzy approaches [M]. [S. l] : Wiley-IEEE Press, 2008: 438.

[69] Shabtai A, Elovici Y. Applying behavioral detection on Android-based devices [C]// Proc of International Conference on Mobile Wireless Middleware, Operating Systems, and Applications. 2010: 235-249.

[70] Shabtai A, Tenenboim-Chekina L, Mimran D, *et al.* Mobile malware detection through analysis of deviations in application network behavior [J]. Computers & Security, 2014, 43 (6): 1-18.

[71] 王聪, 张仁斌, 李钢. 基于关联特征的贝叶斯 Android 恶意程序检测技术 [J]. 计算机应用与软件, 2017 (1): 286-292.

[72] 张思琪. 基于改进贝叶斯分类的 Android 恶意软件检测 [J]. 无线电通信技术, 2014 (6): 73-76.

[73] 张锐, 杨吉云. 基于权限相关性的 Android 恶意软件检测 [J]. 计算机应用, 2014, 34 (5): 1322-1325.

[74] 卢文清, 何加铭, 曾兴斌, 等. 基于混合特征的 Android 恶意软件静态检测 [J]. 无线电通信技术, 2014 (6): 64-68.