基于Android平台的隐私泄漏静态检测工具的分析与比较

燕季薇[1,2] 李明素[3] 卢 琼[4] 严 俊[1,2,4] 高红雨[3]

(中国科学院软件研究所计算机科学国家重点实验室 北京 100190)[1]

(中国科学院大学计算机与控制学院 北京 100049)[2] (北京工业大学计算机学院 北京 100124)[3]

(中国科学院软件研究所软件工程技术研究开发中心 北京 100190)[4]

摘 要 近年来,Android平台应用程序的隐私泄漏问题受到越来越多的关注。应用程序恶意获取用户隐私信息将会增加智能手机用户的隐私泄漏风险,针对该问题,国内外研究人员研究并提出了多种Android平台应用程序的隐私泄漏检测工具。对9种Android平台应用程序的隐私泄漏静态检测工具进行了分析与比较,总结了这些静态检测工具的检测对象、检测方法、能够检测的错误类型和检测效果,并为两种开源工具FlowDroid和IccTA设计了相关实验,以检验其性能及检测效果。针对50个下载的应用程序,FlowDroid成功检测出9个应用存在隐私泄漏,IccTA成功检测到7个组件间泄漏;针对12个自主设计的测试集,FlowDroid和IccTA都成功检测出其中涉及的多种隐私泄漏。

关键词 Android应用,隐私泄漏,静态检测

中图法分类号 TP311.1 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.10.025

Analysis and Comparison of Privacy Leak Static Detection Tools for Android Applications

YAN Ji-wei[1,2] LI Ming-su[3] LU Qiong[4] YAN Jun[1,2,4] GAO Hong-yu[3]

(State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[1]

(College of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China)[2]

(College of Computer Science, Beijing University of Technology, Beijing 100124, China)[3]

(Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[4]

Abstract In recent years, the problems of privacy leak in Android applications attract more and more attention. The maliciously access of private information will increase the risk of users' privacy leak. To solve this problem, researchers have proposed many privacy-leak detection tools that have differences in emphasis point and performance. In order to facilitate the understanding and using for researchers, this paper analyzed and compared nine kinds of privacy leak static detection tools for Android apps. We summarized the detection targets, methods, types of error detection and their efficiency. We also designed and conducted experiments for two open source tools, FlowDroid and IccTA, to test their per-for-mance and detecting ability. For the 50 downloaded apps, FlowDroid successfully detected 9 apps possessing privacy leak and IccTA detected 7 apps possessing ICC leak. For the 12 self-designed test cases, FlowDroid and IccTA can successfully detect all privacy leaks.

Keywords Android application, Privacy leak, Static detection

1 引言

近年来,移动应用已成为人们日常生活中必不可少的部分,但大量应用程序开发周期短、测试不完备,其安全问题日益凸显,其中Android应用程序的隐私泄漏问题受到了越来越多的关注。应用程序被允许合理地访问用户隐私信息,但某些恶意程序会把用户隐私信息泄漏给服务器,服务器则根据这些信息向用户提供有针对性的广告等。造成隐私泄漏的原因有:1)Android 6.0发布前,用户需在应用安装前预先同意应用申请的所有权限,否则无法安装,用户为了使用应用而同意权限申请;2)权限分类机制的粗粒度使得申请到的某类权限远大于应用的实际需求;3)JNI(Java本地接口)的使用也会帮助恶意应用绕过权限检查机制。

为防止恶意泄漏用户隐私的应用程序在市场上传播,国内外研究人员进行了大量的工作,提出了许多Android平台应用程序的隐私泄漏检测方法,包括动态检测方法和静态检

到稿日期:2016-09-06 返修日期:2016-12-14 本文受国家自然科学基金(91418206)资助。

燕季薇(1992-),女,硕士生,主要研究方向为软件测试,E-mail:yanjw@ios.ac.cn;**李明素**(1993-),女,主要研究方向为软件测试;**卢 琼**(1990-),女,硕士生,主要研究方向为软件测试;**严 俊**(1980-),男,副研究员,主要研究方向为程序分析、软件测试;**高红雨**(1968-),男,副教授,主要研究方向为编译技术。

测方法。其中,静态检测方法不需要实际安装并运行应用,而是直接分析应用程序本身,从而判断应用程序是否具有恶意倾向。

目前已有多种相应的隐私泄漏静态检测工具,它们在应用场景、使用技术等方面各不相同,在检测的侧重点和性能上也存在诸多差异。本文将主要关注隐私泄漏相关的静态检测工具,通过对它们的分析与比较,帮助研究人员在实验前更好地了解这些工具并根据不同实验场景与需求选择合适的工具,这对 Android 隐私泄漏检测的相关工作具有一定的指导意义。本文的主要贡献如下:

1)调研了 9 种基于 Android 平台的隐私泄漏静态检测工具,分析并比较了其原理与性能,总结了它们在检测对象、检测方法、能够检测的错误类型、检测效果等方面的差异。

2)自主开发了隐私泄漏相关应用测试集,测试集涉及联系人、位置、照片、偏好设置等隐私信息的泄漏,考虑了隐私信息的组件内泄露和组件间泄漏两种情况。

3)为开源工具 FlowDroid 和 IccTA 设计了实验,分析并比较其性能及泄漏检测能力,测试集包括从 Android 市场下载的应用程序以及自主开发的隐私泄漏测试集。

2 相关理论基础

本节将对隐私泄漏的基本概念和静态分析技术等相关背景知识进行介绍。

2.1 隐私泄漏

隐私泄漏指用户的隐私信息在未经用户许可的情况下被非法获取,并被发送到应用或设备之外的情况。隐私信息将从敏感信息的源点传播到敏感信息的汇点,在汇点从应用程序泄露。

2.1.1 隐私泄漏源点

隐私泄漏源点指能返回隐私信息的 API 调用。通过对相关文献的总结,可将以下信息划为隐私信息,与以下信息的返回相关的 API 调用即为隐私泄漏源点。

1)联系人信息:即手机通讯录,包含联系人的姓名、手机号码、邮件等信息。

2)短信和通话记录信息:可能包含用户不希望被他人获取的信息,如果这些信息被恶意取得,可能会造成用户的重大损失。

3)手机硬件信息:如本机号码、IMEI 码(手机的唯一识别号码)和 IMSI 码(SIM 卡中用于区别移动用户有效信息的号码)等。

4)手机地理位置信息:许多应用基于用户的地理位置来提供服务,若在未经用户许可的情况下获取用户位置,也会导致隐私泄漏。

5)手机安装的软件信息:用户安装的应用列表、应用的具体版本信息等。

6)多媒体信息:用户的照片、视频、录音等,未经用户许可而窃取用户的多媒体信息也属于侵犯用户隐私。

2.1.2 隐私泄漏汇点

隐私泄漏汇点通常指可以将隐私信息传送到网络、文件或通过短信发送隐私信息的 API 调用。

1)网络。通过网络泄漏用户隐私是目前最常见的泄漏途径。例如,应用可以通过声明 ACCESS_INTERNET 来获取网络连接权限,并在 URLConnection 的 OutputStream 中添加用户的隐私信息或将用户隐私信息作为 HTTP 的 GET 和 POST 参数,从而泄漏用户隐私。

2)短信。恶意软件可以向指定号码发送包含用户隐私信息的短信,从而泄漏用户的隐私信息。例如,通过 SmsManager 中的 sendTextMessage()方法发送短信。

3)文件。恶意软件先以文件的形式将用户的隐私信息存储在手机内,然后通过网络或短信向外泄漏。

2.2 静态分析技术

静态分析技术指在不实际执行程序的情况下通过分析程序的控制逻辑来完成对目标程序的检测。它不依赖于程序的执行,分析覆盖率高,但分析的时间较长,且有一定的误报率。下面介绍 4 种静态分析技术。

2.2.1 污点分析技术

污点分析技术指通过分析程序运行时的数据流来标记并跟踪特定数据的过程[1]。污点分析可以分为 4 部分:1)污点,即需要跟踪的特定数据;2)传播过程,即污点在程序中的传播路径;3)源点,即敏感信息提供点;4)汇点,即敏感信息泄漏点。污点分析要求预先定义好源点和汇点,并检测源点和汇点之间是否存在来自源点的数据流。

2.2.2 程序切片技术

程序切片技术[2]指将程序中用户感兴趣的代码抽取出来组成一个新的程序,新程序即为原程序的切片。程序切片是由程序中的一些语句和判定表达式组成的集合,这些语句和表达式可能会影响程序中某个位置 i 上定义或使用的变量 v 的值。$\langle i,v\rangle$ 被称为切片准则,根据切片准则的不同,生成的切片也各不相同。本文主要关注静态程序切片,即在不运行程序的情况下进行切片,构造程序切片时使用静态数据流和控制流的分析方法,以程序的静态信息为依据。

2.2.3 符号执行技术

符号执行的基本思想是使用抽象符号表示程序中变量的值,然后模拟程序执行来进行相关分析,主要用于分析程序中变量间的约束关系。符号执行不需要确定具体的输入数据,而是将变量作为代数中的抽象符号,结合程序的约束条件进行推理。

2.2.4 抽象解释技术

抽象解释技术在抽象的对象域上进行计算,抽象逼近具体对象域上的计算,使抽象执行的结果尽可能反映程序的真实运行结果。抽象解释追求计算效率和精度之间的均衡,通过损失计算精度求得计算的可行性,再通过迭代计算增强计算精度。抽象解释将程序中某些具体值概括成抽象值,一个好的抽象解释可以把多个具体值映射为一个抽象值,虽然最后的结果损失了计算精度,但仍然能正确地总结出程序的某些性质。

3 隐私泄漏静态检测工具的比较

本节介绍一些已有的 Android 应用静态分析工具的检测目标、检测方法、检测效果以及目前存在的不足等。表 1 列出

了本文调研的静态分析工具。

表 1　静态分析工具信息

工具名称	检测对象	是否开源	检测方法
ScanDal	Dalvik 字节码	否	静态污点分析 抽象解释
LeakMiner	Java 字节码	否	静态污点分析 程序切片
AndroidLeaks	Java 字节码	否	静态污点分析 程序切片
AppIntent	Dalvik 字节码	否	静态污点分析 符号执行
FlowDroid	Jimple 中间码	是	静态污点分析
IccTA	Jimple 中间码	是	静态污点分析
PCLeaks	Dalvik 字节码	否	静态污点分析
AppCaulk	Dalvik 字节码	否	字节码插桩 静态＋动态污点分析
AppContext	Jimple 中间码	否	静态污点分析

3.1　检测工具

本小节将简要介绍每种检测工具的基本原理。

3.1.1　ScanDal

ScanDal[3] 是基于抽象解释框架而设计的分析器。它以 apk 文件为输入，先将 Dalvik 字节码转换为 Dalvik Core，然后构建程序控制流图。Dalvik 指令超过 220 条，其中包括许多功能相近的指令，而 Dalvik Core 将指令简化到 15 条，具有同样的表达能力。为了检测出所有可能的泄漏情况，ScanDal 考虑了应用执行时可能发生的每一个机器状态，并为每个状态计算估计值，它收集从源点处生成的值并用程序计数器记录，如果生成的值通过汇点流出，则认为出现隐私泄漏。

3.1.2　LeakMiner

LeakMiner[4] 检测隐私泄漏的步骤为：预处理、敏感信息确认和信息流传播。在预处理阶段，它将 dex 字节码转换为 Java 字节码，提取 Manifest 中的权限配置信息。根据权限与 API 调用之间的映射关系，在所有被使用的 API 中，只有相应权限被申请的 API 才会被分析。LeakMiner 检测所有依赖敏感信息源点的指令，分析源点可能的传播路径，如果敏感信息被传播到网络或打印到本地日志系统，则该泄漏路径就会被报告给用户。

3.1.3　AndroidLeaks

AndroidLeaks[5] 能够快速地对大规模应用进行检测。它也需要将 dex 字节码反编译为 Java 文件，分析应用申请的权限，根据应用 API 调用和权限之间的映射，使用 API 调用的子集作为隐私信息数据流分析源点和汇点的集合。AndroidLeaks会生成函数调用图(CG)，确定源点或汇点的调用点，并执行污点分析以确定敏感数据是否从源点被传输到汇点，若检测到隐私泄漏路径，则生成相应的报告。

3.1.4　AppIntent

AppIntent[6] 的提出是为了检测隐私泄漏并判断该隐私的传播是否符合用户意图，只有在用户未知的情况下传播信息才属于隐私泄漏。它采用污点分析方法提取所有可能的数据传输路径以及与每条路径相关的事件，从而构建事件空间约束图，得到数据传播过程对应的一组事件执行序列，即一组 GUI 操作。AppIntent 使用 InstrumentationTestRunner 自动执行该程序序列，通过敏感数据传播的可视化为分析人员直观地判断敏感数据传输是否符合用户意图提供方便。

3.1.5　FlowDroid

FlowDroid[7] 提出了一种高精度的动态污点分析方法，通过构造精确的 Android 生命周期模型来分析系统回调函数，并通过上下文敏感、流敏感、字段敏感和对象敏感分析来消除误报。FlowDroid 扩展了 Soot[8] 框架，它搜索应用的生命周期、回调方法和对源点、汇点的调用；分析得到主函数，并生成 CG 和过程间控制流图(ICFG)，从检测到的源方法开始，通过遍历 ICFG 来跟踪污点；最后，FlowDroid 将报告检测到的所有从源点到汇点的数据流，将提取的对象链接到它们的前驱或生成语句，生成一张包含所有隐私泄漏的相关赋值语句图。

3.1.6　IccTA

IccTA[9] 针对 Android 应用组件间隐私泄漏(ICC)的问题，提出了检测分析方法。它首先将 Dalvik 字节码转换成 Jimple 表示；其次提取 ICC 链接，将其与 ICC 调用参数等数据一起存储到数据库中；然后修改 Jimple 以直接连接组件，从而实现组件间的数据流分析；最后在 FlowDroid 和组件间污点分析工具的基础上构造了更加完整的控制流图，这使得上下文(Intents 值)可以在组件间传播并能够进行高度精确的数据流分析。

3.1.7　PCLeaks

PCLeaks[10] 关注于潜在的组件泄漏问题，它针对潜在的 ICC 缺陷进行数据流分析，从而找到这些泄漏点可能被其他组件利用的方式并进行验证。PCLeaks 提取 Manifest 文件获得可达的组件列表，并通过更改 FlowDroid 仅为可达组件建立控制流图(CFG)。然后根据源点和汇点方法集合进行污点分析，由于关注的是潜在组件泄漏，因此也将所有组件的入口点和出口点分别当作源点方法和汇点方法。对于每个潜在泄漏点，PCLeaks 验证器将生成一个相应的 Android 测试程序，该程序试图利用这个漏洞来验证该泄漏是否为误报。

3.1.8　AppCaulk

AppCaulk[11] 使用动态分析和静态分析相结合的方式来检测并消除应用程序中的数据泄漏。它首先定义数据泄漏，然后对应用程序进行反编译和数据流分析，该分析需要两次遍历代码，第一次得到后向切片，确定通向汇点的传播路径并清除与向汇点传播的无关语句；第二次得到前向切片，清除所有未从源点开始的传播路径。最后，通过向程序中插入动态代码来跟踪数据流并处理数据泄漏，从而得到不存在数据泄漏威胁的应用。

3.1.9　AppContext

AppContext[12] 利用安全敏感行为的上下文来区分恶意应用和良性应用。它通过构建 CG 以及其中的安全敏感方法来定位安全敏感行为，然后把 CG 转换成扩展调用图，为其中每个安全敏感方法的调用构建精简的过程间控制流图(RICFG)，遍历 RICFG 查找条件语句集，再通过数据流分析找到上下文因素，最后生成完整的上下文。

3.2　检测结果

本小节介绍研究人员针对这些工具进行的实验，以及各个工具相关的性能评估和分析。

3.2.1 性能分析

表 2 列出了研究人员对被调研工具进行测试的实验结果，包括对这些工具进行测试时使用的测试集、检测到的隐私泄漏数、应用的平均分析用时以及产生的误报等。

表 2 静态分析工具的测试结果

工具名称	测试集	检测出泄漏的应用数量	分析用时	报告：确认和误报
ScanDal	从 Android 市场上下载的 90 个应用	11	19 个检测到泄漏的应用中，最少用时 1s，最多 23049s	路径级：确认 28/448，误报 82/448，未知 338/448
	8 个恶意应用	8		
LeakMiner	从 Android 市场上下载的 1750 个应用	—	平均 150s	路径级：确认 145/305
AndroidLeaks	从 Android 市场上下载的 24350 个应用	7414	平均 4.5s	应用级：确认 2342/7414，确认率大于 32%
AppIntent	从 Google Play 上下载的 1000 个应用	140	平均 180s	应用级：误报 164/582
	750 个恶意应用	442		
FlowDroid	DroidBench(39 个手工构造应用)	30	—	应用级：确认 26/30，误报 4/30，漏报 2/28
	InsecureBank(1 个构造的恶意应用)	1	31s	路径级：确认 7/7，误报 0/7，漏报 0/7
	从 Google Play 上下载的 500 个应用	—	大部分少于 60s，最长 270s	—
	1000 个恶意应用	—	平均 16s(5～71s)	
	SecuriBench Micro 1.08	—	—	路径级：确认 117/126，误报 9/126，漏报 4/121
IccTA	DroidBench 和 ICCBench(31 个应用)	29	20～30s	应用级：确认 28/29，误报 1/29，漏报 1/29
	从 Google Play 上下载的 15000 个应用	337		—
	MalGenome(1260 个恶意应用)	108		
PCLeaks	从 Google Play 下载的 2000 个应用	185	平均 40s	路径级：确认 15/20
AppCaulk	从 Google Play 下载的 48 个应用	15	1595s	路径级：确认 12/15，误报 3/15，漏报 0/18
AppContext	202 个恶意应用、633 个良性应用和 11 个开源应用	192	平均 647s	应用级：确认 192/219，误报 27/219，漏报 10/202

ScanDal 对 Android 市场的 90 个应用进行检测，发现 11 个应用存在隐私泄漏，其中 6 个应用将位置信息发送到广告服务器，5 个应用将位置信息保存到文件中，1 个应用将 IMEI 发送到远程服务器。这 11 个应用共产生 448 条泄漏路径报告，其中 28 条被确认，82 条为误报，338 条难以判断。对恶意应用集合的检测表明，8 个应用都存在泄漏电话号码、IMEI、IMSI、ICC-ID 或位置等隐私信息的情况。

LeakMiner 共检测得到 305 条隐私泄漏报告，通过验证发现，其中 145 条报告中的隐私泄漏问题确实存在。LeakMiner 对每个应用的平均分析时间约为 2.5min。

AndroidLeaks 成功分析了测试集中的 24350 个应用，在 7414 个应用中检测到 57299 个隐私泄漏，其中 2342 个应用经验证确实存在手机身份信息、位置信息、WiFi 状态信息和音频记录信息等隐私信息泄漏问题。每个应用平均分析用时为 4.5s。

AppIntent 检测到 582 个应用存在隐私泄漏，其中 164 个检测结果被确认为误报。在静态分析阶段每个应用平均花费 3.3min，而一次符号执行需花费约 5～134min 用于确认静态分析报告的路径，具体用时取决于搜索空间和应用的复杂度。

FlowDroid 针对多个测试集进行了测试。基于 DroidBench 的测试表明 FlowDroid 有较低的误报和漏报；在对 InsecureBank 的测试中 FlowDroid 能够检测出所有构造的攻击，仅用时 31s；针对 Google Play 的测试集进行的实验结果表明，大量应用会将用户的隐私信息记录到日志或偏好文件中，且分析时间大多在 1min 以内；恶意应用测试集的实验中，平均每个应用包含 1.85 个隐私泄漏，由于样本规模较小，平均分析时间为 16s；基于 SecuriBench 的测试则证明 FlowDroid 在处理 Java 的污点分析问题时同样表现优异。

IccTA 主要用于检测 ICC 隐私泄漏问题，针对 MalGenome 测试集的实验中，在 108 个应用中检测到共 534 个 ICC 泄漏；对于来自 Google Play 的测试集，在 337 个应用中检测到 2395 个泄漏，每个应用检测用时约 20～30s。

PCLeaks 也是一种 ICC 缺陷检测的工具，它检测了从 Google Play 上下载的 2000 个应用的检测到 43 个应用中存在 143 个 PACL(潜在主动组件泄漏)问题，并在其中的 147 个应用中检测到 843 个 PPCL(潜在被动组件泄漏)问题，检测每个应用，平均用时约 40s。

AppCaulk 的测试集是从 Google Play 上下载的 48 个应用，成功分析了其中的 44 个，平均每个应用用时约 29min，检测到其中 15 个应用存在隐私泄漏，3 个被证明是误报，没有漏报的情况。

AppContext 的测试集包括 846 个 Android 应用，其中包括 633 个良性应用、202 个恶意应用和 11 个开源应用。对于每个应用，AppContext 的分析用时约为 647s。它可以正确地检测到 202 个恶意应用中的 192 个存在隐私泄漏，但也将 27 个正常方法误报为恶意应用，同时存在 10 个应用漏报，精确度为 87.7%，查全率为 95%。

3.2.2 误报

如 3.2.1 节的评估结果所述，本文调研的各个工具都存在一定程度的误报，本节主要分析各个工具产生误报的原因。

ScanDal 产生误报的原因在于分析库调用时扩大了实际泄漏。当在隐私泄漏路径上需要调用库函数时，它对通过参数进行交换的隐私信息进行估计，此过程可能发生误报。

LeakMiner 分析产生误报的主要原因是应用内的长传播路径。大约 40 个误报是由污点分析期间上下文信息不充分所导致，这样的误报可以通过使用上下文敏感的方法来消除。其他误报大部分是由程序中的调试辅助代码导致，在实际运行过程中调试辅助代码不会被执行，因此可以通过消除调试辅助代码的方式来减少这部分误报。

AndroidLeaks 的误报率约为 35%，主要发生在包含广告库的应用中。当多个广告库的 UI 部件共享同一个 Activity 时，AndroidLeaks 认为广告库之间可以通过共享 Activity 的方式互相泄漏隐私信息，但实际上广告库之间可能并未互相泄漏隐私信息，从而产生误报。

AppIntent 的分析结果表明静态分析中许多误报均由上下文信息不充分和死代码导致，比如调试代码在 if(debug)代码段中。

FlowDroid 存在漏报的主要原因是在静态污点整体架构中它不考虑 Service 组件中的回调函数，因此其检测结果中存在与 Service 组件相关的漏报。

PCLeaks 自身提供验证器验证潜在泄漏的报告。经验证发现，产生误报的主要原因在于条件语句不敏感和不充分的字符串分析。当检测到的泄漏路径中存在不可达的条件语句时，它会对所有可能的路径进行过度估计。此外，由于 PCLeaks 验证器只能执行简单的字符串分析，当字符串的值不能确定时，它就直接忽略它们，由此产生了误报。

IccTA 目前不能解析复杂的字符串操作，只能对通过 Intent 传递的附加关键码进行简单的字符串分析，因此在分析用于组件通信的 Intent 的过程中可能会产生误报。

AppContext 的检测过程中有两个可能导致误报的原因：第一种误报来自于由 UI 事件触发且没有上下文的安全敏感方法调用；第二种误报来自于 AppContext 错误地将一些类似的良性行为标记为恶意行为，恶意软件中的一些安全敏感方法调用可能不包含恶意目的，但由于 AppContext 会标记恶意软件中所有的安全敏感方法调用，从而产生误报。

3.3　局限性

表 3 列出了上述静态分析工具存在的一些固有局限，“T”表示工具可以处理的情况，“F”表示工具不能处理的情况，“—”表示在文献[3-6,11-12]中未说明。

表 3　各工具尚且存在的局限性

工具名称	本地代码	反射	多线程	ICC 检测
ScanDal	F	T	—	F
LeakMiner	—	—	—	F
AndroidLeaks	F	—	—	F
AppIntent	F	—	—	F
FlowDroid	T	T	F	F
IccTA	T	T	F	T
PCLeaks	F	F	F	T
AppCaulk	—	—	—	F
AppContext	—	—	—	F

3.3.1　本地代码

JNI 方法定义了与本地代码交互的方式，开发者可以使用 JNI 方法将 C/C++库集成到应用中，通过 JNI 方法获取隐私信息时可能导致新的安全问题。

ScanDal 不支持 JNI 方法，它的目标语言是 Dalvik 字节码，因此不能在 JNI 库上进行分析。在 ScanDal 的测试集中，90 个应用中有 16 个包含 JNI 方法，这可能导致分析结果存在问题。AndroidLeaks 同样不支持本地代码，在其测试集中，25976 个应用中有 1988 个(7%)至少包含一个本地代码，因此针对这些应用的分析可能与实际的隐私泄漏情况存在差异。PCLeaks 和 AppIntent 也不分析应用中的本地方法，无法检测本地代码中存在的隐私泄漏。FlowDroid 为大部分本地方法调用定义了污点传播规则，并按照污点传播规则进行分析，对于没有定义规则的本地方法调用，FlowDroid 定义了默认的处理方法：如果调用过程中有参数被污染，那么被调参数和返回值就会被污染。IccTA 是基于 FlowDroid 设计的工具，因此其处理方法与 FlowDroid 相同。

3.3.2　反射

ScanDal 不完全支持对反射相关的 API 的检测，只能处理简单的反射调用。在 ScanDal 测试实验中收集的 8 个恶意应用都使用反射来泄漏隐私信息，在 8 个恶意应用中，类和方法名是以字符串常量的方式给出的。恶意开发者可以通过反射来混淆恶意行为，因此在不采用更复杂的字符串分析技术的情况下，分析者不能精确地检测到隐私泄漏。FlowDroid 同样可以在参数是字符串常量时分析反射调用。一些 Java 平台上的反射分析工具(如 TamiFlex)可以帮助静态分析工具处理运行时发生的反射调用，但是，目前 Android 平台不支持这些反射分析工具。IccTA 基于 FlowDroid 实现，同样只有在参数是字符串常量的情况下才能处理反射调用。PCLeaks 无法分析反射调用存在的情况，因此无法检测反射调用相关的隐私泄漏。

3.3.3　多线程

目前，FlowDroid 不能处理多线程，FlowDroid 假定线程按照一种任意的线性顺序执行，而这样的假设通常是不充分的，因此可能会导致整体架构中的误报。IccTA 基于 FlowDroid 进行分析，也存在同样的问题。PCLeaks 同样不能处理多线程，目前 PCLeaksValidator 只执行单个方法内的字符串分析，这可能会导致误报。

3.3.4　组件间泄漏

由于本文调研的工具中大部分是针对组件内泄漏的检测工具，因此只有 IccTA 和 PCLeaks 可以检测 ICC 问题。

3.3.5　其他

AndroidLeaks 不能分析 Android 专有的控制和数据流，包括 Intent 和 Content Provider。PCLeaks 不能处理 Content Provider 中的潜在组件泄漏，因为它不能处理 URI，而 URI 在 Content Provider 组件中使用最多。AppIntent 不能分析某些不能进行反编译的应用，因此，未来 AppIntent 将直接使用 Dexpler 来解析 dex 文件，这样就不需要进行从 dex 文件到 Java 字节码的反编译过程。

4　实验

在上述 9 种工具中，开源工具 FlowDroid 和 IccTA 作为两种代表性工具，可分别对组件内和组件间的隐私泄漏情况进行检测。本节将针对它们的设计进行实验并对工具的实验结果进行评估。

4.1　实验准备

采用随机方式从国内的 Android 市场下载了 50 个应用程序，包括下载量较小的新型应用与下载量大于千万次的热门应用，涵盖了生活工具、商务办公、购物商城、网络通讯、影音图像等多个类别，代表了用户在日常使用中可能下载的应用程序。

除从应用市场下载的程序之外，本文还针对不同隐私类型设计并自主开发了相应的应用测试集，用以验证 FlowDroid 和 IccTA 的检测效果。泄漏类型包括组件内泄漏和组件间泄漏。泄漏方式分别为：获取用户通讯录中的联系人信息，并且将联系人信息泄漏出去；获取手机的 IMEI 和 IMSI

号，并通过发送短信的方式泄漏信息；获取用户所在的位置信息，并将位置信息记录到日志系统中；获取并展示用户的照片，将照片信息记录到日志系统中；获取用户偏好设置中存储的用户名和密码等隐私信息，并将信息记录到文件中；获取用户手机的 mac 地址，进而获得用户所在的位置信息。

4.2 实验结果与分析

4.2.1 FlowDroid 实验结果与分析

本文对从 Android 市场上下载的 50 个应用进行了实验与分析。实验结果表明，FlowDroid 可以成功分析 32 个应用并检测到 9 个应用存在 21 个隐私泄漏，大多数应用的分析时间在 1min 以内。

FlowDroid 默认进行高精确度的分析，但有时分析对精确度的要求并不高，而对分析时间要求尽可能短，或者希望能够分析更大的应用，这可能会导致更大的内存开销。因此，FlowDroid 提供了不同的参数以满足不同精确度的要求。参数-aliasflowins 使得别名分析对流不敏感，可能导致更高的误报率，但也会大大降低大型应用的运行时的开销；参数-nostatic 使得 FlowDroid 不跟踪静态字段，加快了分析速度，但可能造成漏报；参数-nocallbacks 不进行 Android 回调的模拟，可减少运行时的开销，但也可能导致漏报。

表 4 列出了 FlowDroid 在不同精度下进行测试时的内存开销和时间开销，F 表示在当前精度配置下未检测到隐私泄漏。实验结果表明，-aliasflowins，-nostatic 和—nocallbacks 这 3 个参数都可以在一定程度上减少分析时的内存或时间开销。但在用-nostatic 参数控制精度后再进行分析时，未检测到“挖财宝理财”应用中的隐私泄漏；同样地，在用-nocallbacks 参数控制精度的情况下，也遗漏了多个应用中的隐私泄漏。

表 4 FlowDroid 在不同精度下的性能开销

	最高精度		-aliasflowins		-nostatic		-nocallbacks	
应用名称	内存开销	分析用时	内存开销	分析用时	内存开销	分析用时	内存开销	分析用时
920 返利	93	15	129	10	92	8	130	8
8684 公交	117	15	50	9	66	8	56	8
多陌商家	292	97	252	84	319	102	669	95
挖财宝理财	104	1803	98	1651	F	F	F	F
360 助手	81	36	80	23	79	13	F	F
借贷宝	62	31	63	11	140	11	F	F
万能 wifi	103	27	103	8	101	8	F	F
自拍相机	517	219	705	61	379	56	F	F
阿姨来了	134	9	133	9	94	9	95	6

表 5 列出了 FlowDroid 针对自主研发的应用测试集进行测试的结果，包括应用泄漏的隐私信息类型、测试所用的时间及测试带来的内存开销。实验结果表明，FlowDroid 可以成功检测到自主设计的应用测试集中涉及的隐私类型

表 5 FlowDroid 针对自主研发的应用测试集的测试结果

Benchmark	隐私信息类型	时间/s	内存/MB
Test_Contact	联系人	19	92
Test_Imei	IMEI，IMSI	15	92
Test_Loc	地理位置	15	102
Test_Photo	照片	25	90
Test_Sharedpreferences	偏好设置	20	98
Test_Mac	Mac 地址	21	90

4.2.2 IccTA 实验结果与分析

IccTA 对包含 50 个应用的测试集进行测试，可以成功分析 27 个应用，并检测到其中 7 个应用中存在 ICC 隐私泄漏。IccTA 的检测过程与 FlowDroid 相比运行时间更长，但占用内存更少。

表 6 列出了 IccTA 针对自主开发的应用测试集进行测试的实验结果，包括 6 个自主研发的组件间泄漏应用，即涉及联系人信息、手机的 IMEI 号、用户所在的位置信息、照片、偏好设置、手机的 mac 地址等隐私信息组件间泄漏的测试应用。实验结果表明，IccTA 可以成功检测出自主设计的 Benchmark 中涉及的隐私信息类型。

表 6 IccTA 针对自主研发的应用测试集进行测试的结果

Benchmark	隐私信息类型	T/s
Icc_Contacts	联系人	45
Icc_Imei	IMEI、IMSI	34
Icc_Location	地理位置	75
Icc_Photo	照片	39
Icc_Sharedpreferences	偏好设置	51
Icc_Mac	Mac 地址	40

结束语 本文对现有的 9 种隐私泄漏静态检测工具进行了分析和比较，它们均实现了隐私泄漏的静态检测功能，但在功能特性、关注点与局限性能方面各有不同。如Android-Leaks 可快速对大规模应用进行检测；AppIntent 关注于用户意图分析；IccTA 主要检测 ICC 隐私泄漏；PCLeaks 关注于潜在的组件泄漏问题。除了泄漏报告的查全率与精确率外，本地代码、反射、多线程等特性处理也会影响检测工具的选取。此外，本文还针对其中的开源工具 FlowDroid 和 IccTA 设计了实验，检验其在下载应用及自主开发测试集上的运行性能及检测效果。这些工作可以帮助研究人员在实验前更好地选择分析工具，因此对于 Android 的安全研究工作具有一定的指导意义。

参 考 文 献

[1] 刘涛. 基于过程间分析的 Android 程序隐私泄漏检测的研究[D]. 上海：上海交通大学，2014.

[2] CAI S M. Research on Program slicing technology and its application[J]. Software Guide，2010，9(11)：44-46. (in Chinese)
蔡素梅. 程序切片技术及其应用的研究[J]. 软件导刊，2010，9(11)：44-46.

[3] KIM J，YOON Y，YI K，et al. ScanDal：Static analyzer for detecting privacy leaks in Android applications[OL]. http://lim.univ-reunion.fr/staff/epayet/teaching/securite/scandel.pdf.

[4] YANG Z，YANG M. LeakMiner：Detect information leakage on Android with static taint analysis[C]// Software Engineering (WCSE). IEEE，2012：101-104.

[5] GILBER C，CRUSSELL J，ERICKSON J. AndroidLeaks：automatically detecting potential privacy leaks in Aandroid applications on a large scale[M]. Springer Berlin Heidelberg，2012：291-307.

[6] YANG Z，YANG M，ZHANG Y. Appintent：Analyzing sensitive data transmission in Android for privacy leakage detection[C]// Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. ACM，2013：1043-1054.

[7] ARZT S, RASTHOFER S, FRITZ C. FlowDroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps[J]. ACM SIGPLAN Notices, 2014, 49(6): 259-269.

[8] LAM P, BODDEN E, LHOTÁK O. The Soot framework for Java program analysis: a retrospective[C]// Cetus Users and Compiler Infastructure Workshop (CETUS). 2012.

[9] LI L, BARTEL A, BISSYANDÉ T F. IccTA: Detecting inter-component privacy leaks in Android apps[C]// Proceedings of the 37th International Conference on Software Engineering. IEEE, 2015: 280-291.

[10] LI L, BARTEL A, KLEIN J, et al. Automatically Exploiting Potential Component Leaks in Android Applications[C]// 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, 2014.

[11] SCHUTTE J, TITZE D, DE FUENTES J M. AppCaulk: Data leak prevention by injecting targeted taint tracking into Android apps[C]// Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE, 2014: 370-379.

[12] YANG W, XIAO X, ANDOWS B. AppContext: Differentiating malicious and benign mobile app behaviors using context[C]// Software Engineering (ICSE). IEEE, 2015: 303-313.

(上接第 102 页)

[3] HERMAN M A, STROHMER T. High-resolution radar via compressed sensing [J]. IEEE Transactions on Signal Processing, 2009, 57(6): 2275-2284.

[4] DUARTE M F, BARANIUK R G. Spectral compressive sensing [J]. Applied & Computational Harmonic Analysis, 2013, 35(1): 111-129.

[5] YANG Z, XIE L H. On gridless sparse methods for line spectral estimation from complete and incomplete data [J]. IEEE Transactions on Signal Processing, 2015, 63(12): 3139-3153.

[6] TANG G G, BHASKAR B N, SHAH P, et al. Compressed sensing off the grid [J]. IEEE Transactions on Information Theory, 2013, 59(11): 7465-7490.

[7] CHI Y J, SCHARF L L, PEZESHKI A, et al. Sensitivity to basis mismatch in compressed sensing [J]. IEEE Transactions on Signal Processing, 2011, 59(5): 2182-2195.

[8] YANG X M, ZHENG Z, HU B. Off-grid DOA estimation of incoherently distributed non-circular sources via generalized approximate message passing [J]. IEEE Electronics Letters, 2016, 52(4): 262-264.

[9] ZHAO Y H, ZHANG L R, GU Y B. Array covariance matrix-based sparse bayesian learning for off-grid direction-of-arrival estimation [J]. IEEE Electronics Letters, 2006, 52(5): 401-402.

[10] WU X H, ZHU W P, YAN J. Direction of arrival estimation for off-grid signals based on sparse bayesian learning [J]. IEEE Sensors Journal, 2016, 16(7): 2004-2016.

[11] CHEN X S, ZHANG X W, YANG J B, et al. How to solve basis-mismatch: From atomic norm to grid-less compressive sensing [J]. ACTA Automatica Sinica, 2016, 42(3): 335-346. (in Chinese)
陈栩杉,张雄伟,杨吉斌,等. 如何解决基失配问题:从原子范数到无网格压缩感知[J]. 自动化学报, 2016, 42(3): 335-346.

[12] CHANDRASEKARAN V, RECHT B, PARRILO P A, et al. The convex geometry of linear inverse problems [J]. Foundations of Computational Mathematics, 2012, 12(6): 805-849.

[13] CANDÉS E, ROMBERGH J, TAO T. Stable signal recovery from incomplete and inaccurate measurements [J]. Communications on Pure and Applied Mathematics, 2006, 59(8): 1207-1223.

[14] RECHT B, FAZEL M, PARRILO P. Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization [J]. SIAM Review, 2010, 52(3): 471-501.

[15] BOYD S, PARIKH N, CHU B P E, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers [J]. Foundations and Trends in Machine Learning, 2011, 3: 1-122.

[16] PHAM D S, VENKATESH S. Efficient algorithms for robust recovery of images from compressed data [J]. IEEE Transactions on Image Processing, 2013, 22(12): 4724-4737.

[17] SHEN C, CHANG T H, WANG K Y, et al. Distributed robust multi-cell coordinated beamforming with imperfect CSI: An ADMM approach [J]. IEEE Transactions on Signal Processing, 2012, 60(6): 2988-3003.

[18] LEINONEN M, CODREANU M, JUNTTI M. Distributed joint resource and routing optimization in wireless sensor networks via alternating direction method of multipliers [J]. IEEE Transactions on Wireless Communications, 2013, 12(11): 5454-5467.

[19] YANG L, ZHOU J X, XIAO H T. Super-resolution radar imaging using fast continuous compressed sensing [J]. IEEE Electronics Letters, 2015, 51(24): 2043-2045.

[20] YANG L, ZHOU J X, XIAO H T. Compressive high range resolution radar imaging based on continuous dictionary [C]// Proceedings of the IET International Radar Conference. Hangzhou, China, 2015: 1-5.

[21] YANG Z, XIE L H. Enhancing sparsity and resolution via reweighted atomic norm minimization [J]. IEEE Transactions on Signal Processing, 2016, 64(4): 995-1006.

[22] WANG S L, LIAO L Z. Decomposition method with a variable parameter for a class of monotone variational inequality problems [J]. Journal of Optimization Theory and Applications, 2001, 109(2): 415-429.

[23] RUSZCZY′NSKI A. An augmented lagrangian decomposition method for block diagonal linear programming problems [J]. Operations Research Letters, 1989, 8(5): 287-294.

[24] BHASKAR B N, TANG G, RECHT B. Atomic norm denoising with applications to line spectral estimation [J/OL]. https://people. eecs. berkeley. edu/～brecht/papers/12. Bha. EtAl. Denoising. pdf.

[25] CADZOW J. Signal enhancement: A composite property mapping algorithm [J]. IEEE Transactions on Acoustics, Speech, and Signal Processing, 1988, 36(1): 49-62.