移动程序中安全信息流静态检测问题的综述

刘晓建[1]，杜茜

（西安科技大学计算机科学与技术学院 陕西 西安 710054）

摘　要：恶意隐私信息泄露问题是当今移动程序重要的安全问题之一，其本质可以追溯到上世纪 70 年代由 D.E.Denning 提出的**安全信息流**问题。移动程序中安全信息流检测的研究工作非常活跃，提出的方法和分析工具在保障个人信息安全方面发挥了重要作用，但是现有方法仍然面临提高分析精度、提升分析效率、增强方法的可扩展性和智能化程度等一系列挑战性问题。聚焦安卓移动应用程序隐私信息泄露问题，回顾了安全信息流的数学定义，针对静态检测技术路线，对现有方法、技术和分析工具进行了分类综述，最后指出了静态信息流泄露检测方法存在的主要问题以及未来可能的努力方向。

关键词： 信息安全，信息流，程序静态分析，污点分析，Android

Approaches to static detecting secure information flow in mobile applications: a survey

LIU Xiaojian, DU Xi

（School of Computer Science and Technology, Xi'an University of Science and Technology Xi'an, Shaanxi 710054,China）

Abstract： Malicious privacy information leakage is one of the important security problems of mobile programs today, and its essence can be traced back to the security information flow problem proposed by D.E. Denning in the 1970s. Research work on security information flow detection in mobile programs is very active. The proposed methods and analysis tools have played an important role in ensuring the security of personal information, but existing methods still face improvements in analysis accuracy, analysis efficiency, and scalability of methods. And a series of challenging issues such as the degree of intelligence. Focusing on the privacy information leakage problem of Android mobile applications, the mathematical definition of security information flow is reviewed, and the existing methods, technologies and analysis tools are classified and summarized for the static detection technology route. Finally, the existence of static information flow leakage detection methods is pointed out Main issues and possible future directions.

Keywords： Information security, Information flow, Static analysis, Taint analysis, Android

---

收稿日期：xxxx-xx-xx.

基金项目：国家自然科学基金(61702408)，

教育部协同育人项目(2010918001)

陕西省科技计划项目(2017JM6105).

作者简介：刘晓建，副教授，博士；

杜茜，女，学士学位.

通信作者：刘晓建，副教授，email：780209965@qq.com.

隐私信息泄露问题本质上是**安全信息流**问题（Secure information flow）[1-4]，即通过一个低密级对象 y 的取值能否推断某个高密级对象 x 的取值的问题，或者高密级输入 x（即隐私数据）是否传播到低密级输出 y（即公开端口）的问题，该问题也称为**数据降级问题**（Downgrade 或 Declassify）[5,6]。

安全信息流问题是由 D.E.Denning 早在 1976 年就提出的挑战性信息安全问题[1,2]，在当今移动互联应用普及的情况下，该问题表现得更加突出[7,8]。文献[9]指出，60%互联网应用的脆弱性来自于不安全的信息流，信息的完整性和保密性遭到破坏。隐私信息泄露、恶意扣费、勒索软件等不同形态的恶意程序中都包含着不安全的信息流，而传统的访问控制技术[10]、数据加密技术[11]等均不能有效解决端对端的安全信息流问题。

针对这一问题，国内外学者开展了深入细致的研究工作，取得了丰硕的研究成果。很多重要 IT 企业也发布了相关安全检测、漏洞挖掘以及加固产品[12-14]。这些方法、技术和产品，在保障个人信息安全方面发挥着重要作用，但是这些方法、技术和工具仍然面临提高分析精度、提升分析效率、增强方法的可扩展性和智能化程度等一系列挑战性问题，需要进一步开展深入研究。

本文综述了安全信息流的基本定义，并以安卓应用程序为研究对象，聚焦恶意隐私信息泄露中的安全信息流检测问题，对安卓应用程序的静态检测方法进行了分类和讨论，最后指出了其中存在的问题，并展望了静态检测方法的发展趋势。

1 安全信息流的定义

隐私信息泄露问题本质上是安全信息流问题，我们首先必须回答什么是信息流以及安全信息流的问题，即给出安全信息流的数学定义。

通过分析信息流安全模型的相关研究工作，我们大致可以总结出安全信息流定义的两个角度：一是 Shannon 信息论的角度，二是程序语义模型的角度。第一个角度的基本思想是：把程序执行看作一个从“前状态”到“后状态”的信息传递过程，或通信过程，程序命令序列就是“通信信道”。通信过程实质上是从信宿变量值推测信源变量值的过程，因此类比通信过程，可以借助信息论对程序执行过程中的信息传递进行定义和分析；第二个角度是从程序执行的语义模型出发，使用程序设计人员所熟知的术语来定义安全信息流。其基本思想是把程序看作从输入到输出的一个状态转换函数，如果低密级输出与高密级输入不发生干扰（Noninterference）或二者无关，那么就没有从输入到输出的信息流，或者说信息流是安全性的。相比而言，从第一个角度进行的定义最为深刻，甚至可以用它来解释从第二个角度进行的定义。下面我们主要介绍从信息论角度定义安全信息流的工作，并对第二个角度的工作进行简要回顾。

第一个角度最典型的定义是由 D.E.Denning 给出[15]，主要包括两部分：第一部分是基于格的信息流多级安全策略（Security policy），所谓安全策略即是一个系统的安全需求，可以用“格”这一代数结构来描述；第二部分采用信息论，给出“信息流”的定义。把这两个部分结合起来就可以定义“安全信息流”的概念。

1.1 安全策略

一个安全信息流要求程序对象（如一个文件、程序变量、内存区域、进程甚至用户）中的信息按照特定的安全策略进行流动，通常要求信息只能从一个对象向同级别或高级别对象流动。为了描述安全策略，D.E.Denning 提出了一种描述安全级别的格模型：一个安全策略可以被定义为一个“格”（Lattice）代数结构：

（SC, ≤），

其中 SC 为安全级别的有限集，“≤”为 SC 上的一个偏序关系。对于我们要讨论的 Android 安全信息流检测问题，通常将安全级别可分为 High 和 Low 两个级别，即 SC={High, Low}，Low≤High。High 代表敏感信息的安全级别，Low 代表可以被外部观察到的公开信息的安全级别；如果信息从具有 High 级别的对象流向具有 Low 级别的对象，那么就会造成信息泄露。

1.2 安全信息流

信息流是伴随程序执行过程（即状态迁移过程）而发生的，为了考察状态迁移过程中的信息流，需要对程序的状态进行扩充，即一个对象的状态不仅包括它的值，还包括它的安全级别。扩充后的程序状态 s 定义为如下函数：

$$s: \mathrm{OBJ} \rightarrow \mathrm{VAL} \times \mathrm{SC}$$

其中 OBJ 为程序对象的有限集，VAL 为对象的值集。对于对象 $x \in \mathrm{OBJ}$，其状态 $s(x)=(x_s, \underline{x}_s)$，其中 $x_s \in \mathrm{VAL}$ 表示 x 在状态 s 下的值，$\underline{x}_s \in \mathrm{SC}$ 表示 x 在 s 下的安全级别。一个对象的安全级别可能是固定的，也可能是可变的，即随着程序状态变化而发生变化。

设程序在状态 s 下，执行命令序列α后，迁移

到了状态 s'，记为：

$$s \rightarrow_{\alpha} s'.$$

我们称 s 为“前状态”（pre-state），s'为“后状态”（post-state）。设 $x\in$OBJ 为前状态 s 下的一个对象，$y\in$OBJ 为后状态 s'下的一个对象，用 $H_{y'}(x)$表示在状态 s'时 y 的值为给定 $y_{s'}$时 x_s 的条件熵，$H_y(x)$表示在状态 s 时 y 的值为给定 y_s时 x_s 的条件熵。如果

$$H_{y'}(x)< H_y(x)$$

即能够从 $y_{s'}$推断出 x_s 的更多信息（或减少 x_s 的不确定性），那么我们称：在状态 s 下执行α导致了从 x 到 y 的信息流，记为

$$x_s \rightarrow_{\alpha} y_{s'} \text{ 。}$$

如果存在状态 s，s'以及命令序列α，使得 $x_s \rightarrow_{\alpha} y_{s'}$，那么我们把 x 到 y 的信息流记为 $x \rightarrow_{\alpha} y$，或简记为 $x \rightarrow y$。

我们称一个信息流 $x_s \rightarrow_{\alpha} y_{s'}$ 是“安全的”，当且仅当 $\underline{x}_s \leqslant \underline{y}_{s'}$成立，即信息流必须从一个安全级别的对象流向同级或更高级安全级别的对象。

安全信息流的这一定义，给出了信息流安全性的一个判据，但是这一判据仅是一个“存在性”判据（Sanity checking）。实际上，这一判据过于保守，很多情况下即使存在一定程度的信息泄露，也并不影响系统的安全性。

1.3 基于程序语义模型的安全信息流定义

上述信息流的定义是从信息论的观点出发，把程序命令序列看作信息传输的信道，并认为如果信源对象（即上例中的 x）通过程序处理后会产生熵减，即通过信宿对象能够推断出信源对象的新信息，那么就会产生信息流泄露。但是，对于大部分计算机科学家而言，总是希望能够从程序执行的角度（如状态迁移、输入/输出变换）来定义和解释安全信息流问题。从这一角度出发，产生了若干安全信息流模型，典型的有通用流模型 FM(Flow Model)[17]、非干扰模型(Noninterference)[18]和非推测性(Nondeducibility)[19]。

非干扰安全模型基于如下几个系统假设：（1）程序是确定性（Deterministic）程序（即非概率程序）；（2）系统的输出由系统的输入序列唯一决定；（3）系统的输入可分为高级别输入集合 High-in 和低级别输入集合 Low-in；系统的输出可分为高级别用户能够观察到的输出集合 High-out 和低级别用户能够观察到的输出集合 Low-out，并且 High-out 不能由 Low-in 唯一决定。

我们用 w 表示一个输入序列，即 $w\in$(High-in | Low-in)*。我们用 out(u, w)表示在给定一个输入序列 w 时，用户 u 所能观察到的系统输出。系统所产生的输入到输出的信息流是安全的，当且仅当满足以下非干扰条件：

$$\text{out}(Low, w)=\text{out}(Low, w')$$

其中，Low 表示低级别用户，w'表示把 w 中的 High-in 输入去掉后得到的输入序列。上述非干扰条件说明了，一个低级别用户所能观察到的输出与高级别输入无关，即高级别输入信息不会流入低级别输出，确保了信息只能向同级或高级别用户流动。

McLean 提出了一个通用的流模型 FM（Flow Model），并以此模型为标准，比较和分析了几类典型信息流安全模型（如 Nondeducibility、Noninterference、广义非干扰 (Generalized Noninterference)和 BLP 模型)的差异，并指出一个系统如果满足如下条件，则信息流是安全的：

$$\text{p}(L_t \mid H_s \,\&\, L_s)= \text{p}(L_t \mid L_s)$$

其中，L_t 表示在状态 t 时低级别对象的状态值，H_s 和 L_s 分别表示高级别对象和低级别对象在状态 t 之前的所有状态下的值的序列。p($L_t \mid L_s$)表示条件概率，即从一个低级别对象的历史状态推测该对象的下一个状态值的概率。上述定义说明了，低级别对象在状态 t 时的值 L_t 不依赖于高级别对象值的历史。

FM 模型是一种更为通用的模型，可以用它来表达上面的非干扰模型，即

$$\text{p}(\text{Low-out}_t \mid \text{High-in}_s \,\&\, \text{Low-in}_s)= \text{p}(\text{Low-out}_t \mid \text{Low-in}_s)$$

前面所述的这些安全模型都是针对确定性程序的，而 Gray[20]针对非确定性（Nondeterministic）系统，采用概率有限状态机模型给出了安全信息流的定义，对 McLean 的通用流模型进行了具体化，并通过定义机密和非机密变量之间的信道容量从而建立起与传统信息论之间的显式联系。McIver 和 Morgan[21,22]则针对概率程序设计了一种新的信息流和信道容量的信息理论定义。

前面讨论的安全信息流的这两种定义角度实际上是统一的，比如[23]证明了：对于一个确定性程序 c，一个信息流 $x\rightarrow_{\alpha} y$ 是安全的当且仅当高级别对象 x 在执行α时，不干扰低级别对象 y 的取值。

基于 FM 和非干扰模型等安全信息流定义，可以进行安全或不安全信息流的判断，但是实际程序完全没有信息泄露几乎是不可能的，通常只有在信息流泄露达到一定量的情况下，才认为会

发生不良行为。反之，只要产生的泄露足够少，我们就能确保系统的安全性。

2 移动程序隐私信息泄露检测的主要技术路线

尽管上面给出了安全信息流的一般性定义，但是在实际问题中，难以直接应用定义式进行安全信息流检测。针对具体领域问题，需要发掘信息泄露的主要矛盾，以及安全信息流问题的主要特征，进而发展出具有针对性的、有效的安全信息流检测方法。下面我们聚焦于移动程序这一具体领域，综述检测恶意信息流泄露（以简称 MIF，Malicious Information Flow）的主要技术路线和方法。

有关移动程序恶意信息泄露行为检测的研究工作非常丰富，我们首先对主要技术路线做一梳理，然后重点分析静态检测的若干主要方法，最后指出现有研究工作的不足。

隐私信息泄露问题本质上是安全信息流问题。对于我们要讨论的 Android 安全信息流检测问题，安全级别可分为 High 和 Low 两个级别。High 代表敏感信息的安全级别，Low 代表可以被外部观察到的公开信息的安全级别；如果信息从具有 High 级别的对象流向具有 Low 级别的对象，那么就会造成了敏感信息的泄露。相关文献及其分类如表 1 所示。

静态检测方法不执行程序，而是直接对程序文本进行分析，其实质是对程序行为做某种程度的抽象和逼近，因此存在分析复杂度较高，分析方法可能过于保守，因而具有一定误报率等内在挑战性问题。另一方面，其优点也非常明显：路径覆盖度较高、检测不依赖于运行环境、有利于分析恶意行为机理等。

表 1 静态分析方法和工具的主要文献

Table 1 Main literature on static analysis methods and tools

基于数据流的污点分析	FlowDroid[25],AmanDroid[26],DroidSafe[27], SuSi[32],SCanDroid[46], 2009~2018
	DroidChameleon[37],SEFA[47], UID[36], RiskRanker[48], AndroidLeaks[49],EdgeMiner[51] 2012~2015
	SCANDAL[50],CHEX[35],Eppic[33],FlowCog[38],MultiFlow[39],2012~2019
	IccTA[31],IC3[34],ICCDetector[51] ,StubDroid[52],2012~2016
基于类型的污点分析	DFLow & DroidInfer[43],2015
	Security Type System[40,41,42], 1996~2013
	Permission-Dependent Type System[44],2017

3 静态检测的主要方法

通过对大量相关文献的分析和总结，我们将从两个方面阐述隐私信息泄露行为的静态检测方法。

3.1 基于数据流分析的污点传播分析方法

该分析方法的基本思路是：首先对程序中的隐私数据进行污点标记，然后定义污点随程序语句不断传播的逻辑，并采用数据流分析框架，分析污点最终是否传播到特定的信息输出端口（如文件、短信、网络等）来判断是否产生了信息泄露。图 1 所示为一段 Android 应用程序代码，运行该段程序会导致用户的密码数据通过发送短信的方式泄露.污点分析首先要识别引入敏感数据的接口(source，污点源)并进行污点标记。即识别第 4 行的 passwordText 接口为污点源，并对 pwd 变量进行标记。如果被标记的变量又通过程序依赖关系传播了给其他变量，那么根据相关传播规则继续标记对应的变量。当被标记的变量到达信息泄露的位置(sink，污点汇聚点)时，则根据对应的安全策略进行检测。

IFDS 框架[24]是一个精确高效的上下文敏感和流敏感的数据流分析框架。IFDS 的全称是过程间(interprocedural)、有限的(finite)、满足分配率的(distributive)、子集合(subset)问题。该问题作用在有限的数据流域中，且数据值(data flow fact)需要通过并(或交)的集合操作满足分配率。满足上述限定的问题都可以利用 IFDS 算法进行求解，而污点分析问题正是满足 IFDS 问题要求的。IFDS 问题求解算法的核心思想就是将程序分析问题转化为图可达问题。由于安卓程序普遍采用面向对象程序设计语言来开发，并使用消息通信进行构件交互，因此在进行污点分析时，还要结合组件间消息传递（以下简称 ICC）、过程间数据流分析（Interprocedural dataflow analysis）、指向和别名分析（Points-to & Alias analysis）等分析方法。静态污点分析方法具有不同的分析精度，一般按照流敏感、上下文敏感、对象属性敏感和路径敏感等维度来划分精度，分析精度越高、复杂度和开销也就越高，因此要在二者之间达到平衡。基于数据流分析的静态检测方法和相关工具发展情况如图 1 所示。

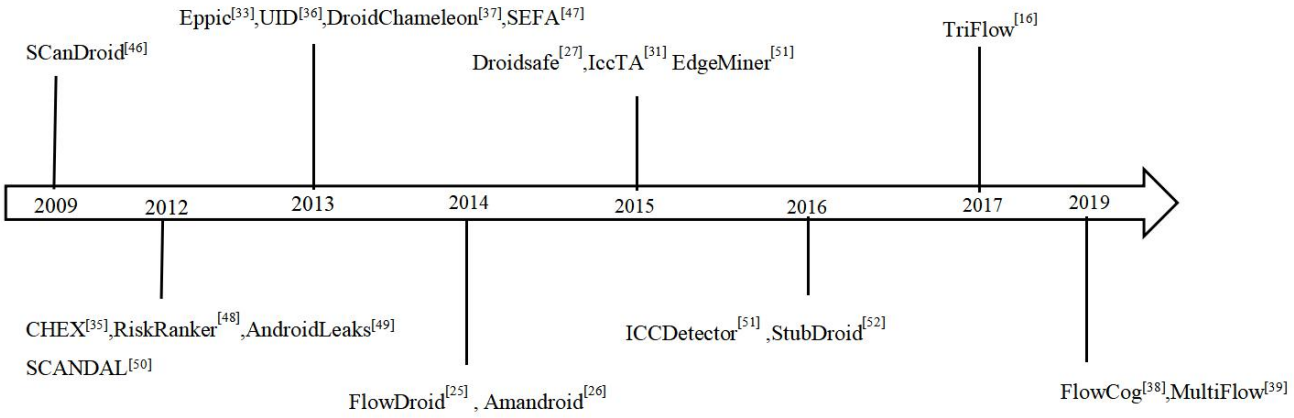

图 1 基于信息流分析的静态检测工具发展情况

Fig.1 Development of static analysis tools based on information flow analysis

静态污点分析最具代表性的工作有德国 Darmstadt 技术大学的 Eric Bodden 小组提出的 FlowDroid[25]、美国南佛罗里达大学 Fengguo Wei 等人提出的 Amandroid[26]以及美国 MIT Michael I. Gordon 等人提出 DroidSafe[27]。FlowDroid 采用 Dexpler 反编译器，把 APK 反编译成 Jimple 中间格式，然后在应用程序中搜索生命周期和回调方法以及对源和汇的调用。然后 FlowDroid 基于 Soot[28]在 iCFG（Interprocedural CFG）上应用 IFDS/IDE 数据流分析框架进行污点分析，整个分析与执行顺序相关，主要分为两个部分，向前的污点分析用于找出被污染的变量传递到了何处和向后的按需别名分析用于查找源点之前所有对同一被污染的别名。

图 2 例示了一个污点分析的过程。它的具体过程：① w 作为被污染了的变量，被向前传递给了 x.f；② 继续追踪 x.f；③每当堆中有成员被污染，使用向后分析找出相关对象的各种别名。如发现 x 和 z.g 是引用的堆中的同一位置；④-⑥z 是作为参数传递进来的，继续向后找看到调用者的 a 即被调函数中的 z，再往后发现 b 也是别名；⑦对 b.f 做前向分析，判断出其被传递到 sink，从而展现了从 source 到 sink 的整个路径。

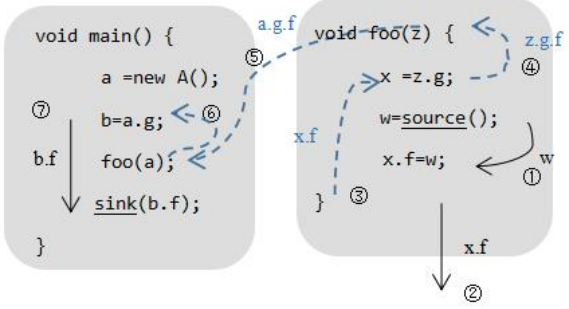

图 2 污点分析的一个实例

Fig.2 An example of static analysis

Eric Bodden 小组在 SuSi[29]工作中，对约 11 万个安卓 SDK 的类方法，采用监督机器学习算法进行了敏感数据源和汇的划分，认定了 13 类共 18000 多个敏感数据源类方法和 8000 多个数据汇类方法，这为 FlowDroid 的实际应用提供了基础。当然，SuSi 只认定了 SDK 中内建的敏感数据源和汇，实际上很多敏感数据（如用户名、密码、个人健康信息等）是通过用户 UI 输入的，为了认定这些通过 UI 界面传递的隐私信息，SUPOR[30]提出了一种对应用程序 UI 敏感性进行分析的静态方法，可以看作是对 SuSi 的有益补充。

有些隐私信息泄露路径跨越多个构件边界，因此必须分析端到端的污点传播，而 FlowDroid 侧重于分析单个组件内部的污点传播，对于跨组件的污点传播支持较弱，DroidSafe[27]、IccTA[31]和 AmanDroid[26]对此进行了改进和增强。由于安卓普遍使用隐式 intent 通信作为组件间传递数据流和控制流的手段，而隐式通信的目标组件通常是以字符串变量值的方式指定，因此跨组件件的污点分析都必须很好的解决字符串变量值的静态分析这一基础性问题。WEI 等人[26]提出了一种对 Android 应用程序进行安全审查的静态分析新方法，并构建了一个通用的框架，称为 AmanDroid，用于通过 Android 应用程序组件以流和上下文敏感的方式来确定 Android 应用程序中所有对象的指向信息。而且 AmanDroid 对输入 app 的每个组件进行数据流和数据依赖分析，还跟踪组件间的通信活动。AmanDroid 可以将组件级信息缝合到应用程序级信息中，执行应用程序内部和应用程序之间的分析。DroidSafe[27]采用 JSA(JSA String Analyzer)解析出 intent 和 uri 中包含的字符串变量值的正则表达式，并通过一个映射表将源构件与目标构件关联起来，然后通过程序变换技术把组件间的间接通信模拟成构件直接调用，这样就可以用传统的过程间数据流分析方法解决跨组件的污点分析问题。而 IccTA[31]使用 Dexpler[32]将 Dalvik 字节码转换成 Jimple，IccTA 将提取 ICC 链接，将它们以及收集到的所有数据(例如，ICC 调用参数或 Intent Filter 值)存储到数据库中。基

于 ICC 链接，IccTA 通过对 FlowDroid 的改进版本，即针对 Android 应用程序的高精度组件内污染分析工具，构建了完整的 Android 应用程序的控制流图。在 Android 组件之间传播上下文(例如 Intents 的值)，并生成高度精确的数据流分析。IccTA 主要的贡献就是对 ICC 链路的提取。IccTA 利用 Epicc[33]获取 ICC 方法及其参数(如 intent 动作)。Epicc 是一个基于 Soot 的工具，用于识别 ICC 方法及其参数值(如 action、category)，接着 IccTA 通过解析名为 AndroidManifest 的应用程序配置文件来识别所有可能的目标组件，以检索 Intent filter 的值。IccTA 的解决方法与 DroidSafe 类似，不同之处在于它使用了 IC3[34]字符串静态分析工具。IccTA 扩展了 FlowDroid，它可以通过常规的 Intent 调用和返回来跟踪数据流。但是，IccTA 尚未跟踪调用绑定服务组件中的方法的特殊类别 ICC(称为远程过程调用(RPC))。CHEX[35]使用了一种不同的方法来对 Android 环境进行建模，它将从入口点可访问的代码片段(称为分割)链接起来，作为一种发现 Android 应用程序组件之间的数据流的方法，但是它没有通过 Intent 通道来处理数据流。UID[36]工具遍历 Java 字节码，生成上下文敏感的过程间和过程内数据流依赖图，以及基于事件的信息流，处理 AndroidIntents 和 GUI 相关隐式方法调用之间的数据流，计算应用程序中用户输入触发敏感函数调用的比例生成其可信度值。DroidChameleon[37]则利用了现有 Android 恶意软件代码混淆技术的代码特征，用于评估现有安全软件对抗代码混淆的能力。但是，这类方法需要应用程序指令和模板完全匹配，当攻击使用指令替换或者重新排序仍然有可能绕过。FlowCog[38]通过控件或数据依赖项静态地找到所有与给定流相关的 Android 视图，然后从这些视图和相关的布局中提取语义，例如文本和图像。然后 FlowCog 采用自然语言处理(NLP)方法来推断所提取的语义是否与给定的流相关从而判断是良性还是恶意的。对于图 3 所示，(a)(b)是新闻应用程序的代码片段,它是一个没有隐私泄露行为的应用软件，(c)是恶意软件 DroidKungfu 的变体。分析我们可以发现在新闻应用中，由于发送信息是在不同的模块下，因此在一次执行过程中，,必然是只能发送地理位置信息到网络就不会发送用户的设备信息，反之亦然。然而对比恶意软件，由于地理信息和用户设备等信息是恶意软件同时需要的，因此它们往往是绑定在一起发送到网络的。对于一般的污点分析工具是无法无法区分两者的恶意行为的。

```
1  public void onlocationChanged (Location location){
2      String latLng = updateLocation(location);
3      ...
4   }

5  private String updateLocation(Location location){
6        String latLng;
7    if (location!=null){
8      double lat=location.getLatitude();
9      double lng=location.getLongitude();
10     latLng="Lari:"+lat+"Longi:"+lng;
11     }else {
12      latLng="Can't acess your location";
13     }
14      return latLng;
15 }
```

(a)良性软件片段 1

```
1  Public  String currentDeviceMark(Context context){
2    TelephonyManager tm=(TelephonyManager)
       context.getSystemService(Context.TELEPHONY_SERVICE;
3    deviceId=tm.getDeviceId();
4    WifiManager wm=(WifiManager)context.
       getSystemService(Context.WIFI_SERVICE);
5    String wlanMacAddr=wm.getConnectionlnfo.getMacAddess();
6    ...
7    If (deviceId!=nul){
8      markId=deviceId;
9    } else if (wifi=State.CONNECTED||wifi==Sate.CONNECTING){
10     markId=wlanMacAdd;
11   }else if (...){
12    ...
13   }
14   ...
```

(b)良性软件片段 2

```
1   protected int onStartCommand(){
2     ...
3     ArrayList list=new ArrayList():
4     List.add(new BasicNameValuePair("pid",tm.getDeviceId()));
5     List.add(new BasicNameValuePair("macaddr",
              tm.getConnectionlnfo(),getMacAddress()));
6     String loc="Lat:"+lm.getLatitude()+"Long:"+lm.getLongitude();
7     List.add(new BasicNameValuePair("loc",loc));
8     ...
9     ServerSession.postRequest(urlEncodedFormEintity(list));
10  }
```

(c)恶意软件片段

图 3 良性软件和恶意软件片段代码

Fig.3 Fragmentary code for benign software and malware

目前的污点分析只能提供单一的污点源和汇聚点的组合结果,并没有考虑多个组合之间的相关性关系。对此王蕾等人[39]提出多源绑定发生的污点分析技术来降低隐私泄露检测的误报率。他们提出了将数据流值扩展以污点变量组成的向量的形式。如图 4 所示的多源绑定发生数据流传播示例，如果数据流值〈 b1,b2〉分别传播到 sink(b1)和 sink(b2),则表示污点变量 b1 和 b2 的源是可以绑定发生的。

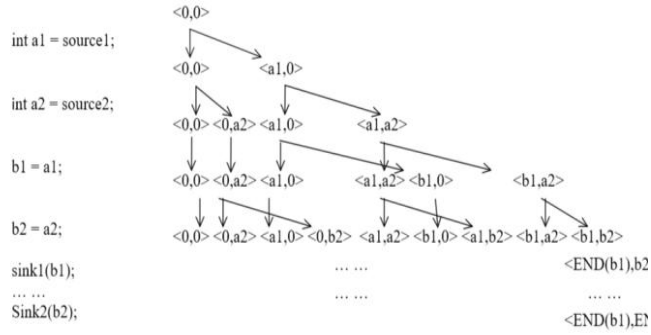

图 4 多源绑定发生数据流传播示例

Fig.4 Example of data flow exmple with multi-source binding

传统的污点分析在未遇到污点汇聚点时并不知道最终的传播路径，因此需要保守地传播所有污点传播的变量。当扩展到多源绑定发生问题时，这种方法将产生大量的无用传播。他们则提出了一种数据结构—数据流值的前驱节点(predecessor)和邻居节点(neighbor)来维护污点源和汇聚点的路径信息，如图 5 所示。数据流值 t1 的前驱节点是 a1。为了区分不同分支的值，令第 7 行处为 b′，第 9 行处为 b，那么在分支结束处会将 b 的邻居节点设置成 b′，那么 b′ 的前驱是 f1,b 的前驱是 f2,它们分别代表不同的路径。有了上述的数据结构，就可以利用触发 sink 的变量进行回溯，求得完整的污点分析路径。

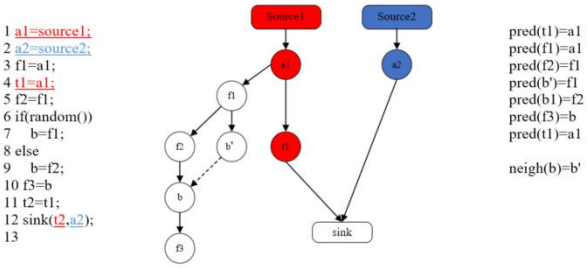

图 5 数据流图前驱和邻居示意

Fig.5 Data flow graph predecessor and neighbors

静态污点分析的优点是：路径覆盖度较高、分析精准，而且由于是白盒方法，有利于分析信息泄露的行为模式和内部机理。基于这些优点，很多研究工作都以 FlowDroid 为基础进一步开展。静态污点分析的主要不足有：1）还无法处理经过加密和代码混淆的应用程序，这是所有静态分析方法所存在的普遍性问题；2）静态污点分析方法仅提供了恶意隐私信息泄露的“必要性”判据，但不是所有信息泄露都具有恶意性，因此还需要额外的人工分析进一步确认泄露是否具有恶意性；3）有些静态污点分析工具的可扩展性不够好。

3.2 基于安全类型系统的污点分析

除了可以用数据流框架静态分析污点传播之外，还可以在程序设计语言层次，利用类型系统（Type system）来分析和检测污点传播[40-42]。基本思路是在程序设计语言原有类型系统的基础上，建立一个安全类型系统（Security-typed language），支持安全信息流策略的规格说明和静态检测。具体方法是在程序变量和表达式的类型中，增加一个表示安全性质的标记（Annotation），用以规范和说明施加在该类型数据上的安全策略，并采用编译时的类型检查对安全信息流性质进行分析和检测。若违反安全性质，则说明存在污点数据传播路径。基于安全类型系统的分析方法具有如下优点：1）仍然属于静态分析范畴，不会给程序运行带来任何开销；2）能够在语言层次对安全策略进行规格说明；3）安全类型检测天然的具有模块化和组合特性，便于安全性质的组合推理。

针对安卓程序污点分析，最典型的工作是由 Ana Milanova 提出的 DFlow & DroidInfer[43]。DFlow 是一个支持面向对象语言、上下文敏感的安全类型系统，在类型系统中增加了 tainted、ploy 和 safe 三种安全标记：tainted 表示被污染，safe 表示安全，poly 在某些情况下被解释为污染，在其他情况下是安全的。并采用子类型（Subtyping）约束来检测污点源数据向敏感汇的传播。它的类型推理规则如图 6 所示。其中，Γ 是将变量映射到 Dflow 限定符中的一种类型环境，q^i 是在调用位置 i 下的自适应上下文。规则（TNEW）和（TASSIGN）强制执行预期的子类型约束。字段访问规则（TREAD）和（TWRITE）从接收者 y 的角度适应字段 f，然后实施子类型约束。

$$\text{(TNEW)}\quad \frac{\Gamma(x)=q_x \quad q <: q_x}{\Gamma \vdash x = new\, q\, C} \qquad \text{(TASSIGN)}\quad \frac{\Gamma(x)=q_x \quad \Gamma(y)=q_y \quad q_y <: q_x}{\Gamma \vdash x = y}$$

$$\text{(TWRITE)}\quad \frac{\Gamma(y)=q_y \quad typeof(f)=q_f \quad \Gamma(x)=q_x \quad q_x <: q_y \triangleright q_f}{\Gamma \vdash y.f = x}$$

$$\text{(TREAD)}\quad \frac{\Gamma(y)=q_y \quad typeof(f)=q_f \quad \Gamma(x)=q_x \quad q_y \triangleright q_f <: q_x}{\Gamma \vdash x = y.f}$$

$$\text{(TCALL)}\quad \frac{typeof(m)=q_{this} \quad q_p \to q_{ret} \quad \Gamma(y)=q_y \quad \Gamma(x)=q_x \quad \Gamma(z)=q_z \quad q_y <: q^i \triangleright q_{this} \quad q_z <: q^i \triangleright q_p \quad q^i \triangleright q_{ret} <: q_x}{\Gamma \vdash x = y.m^i(z)}$$

图 6 Dflow 的类型推理规则

Fig.6 the type inference rules of Dflow

为了禁止来自受污染源的流向安全接收源，DFlow 强制执行以下子类型层次结构：safe <: tainted。一旦给出了源和接收器，推理工具就会自动推断出类型限定符，如果类型推断成功，则从源到汇没有泄漏。如果失败，应用程序可能包

含泄漏。则 DroidInfer 是 DFlow 语言的类型推理引擎，它支持安卓程序的多入口点、回调、第三方库以及 ICC 等基本特性。由于回避了指针分析，因此它具有较好的可扩展性；同时由于 DroidInfer 将类型推理问题归结为“上下文无关文法的可达性”问题（CFL-reachability），因而具有较高的分析精度。Chen 等人[44]介绍了一种新型的命令式语言安全信息流系统。他们基于静态分析 Android 应用程序中潜在的信息泄漏问题，并且设计了一个具有 Android 权限模型的轻量级类型系统，在这个系统中，权限被静态地分配给应用程序，并用于在应用程序中实施访问控制。他们的类型系统设计了用于由权限测试引发的条件分支的类型规则，它在安全类型上引入了一个合并操作符，允许执行更精确的安全策略。Lourenco 和 Caires[45]提供了一个精确的依赖类型系统，在这个系统中，安全标签可以被数据结构索引，这些数据结构可以用来编码安全标签对系统中其他值的依赖关系。可以编码我们的安全的概念类型作为依赖类型的设置,把权限明确作为一个额外的参数设置为一个函数或一个服务,并指定安全水平的函数作为一个类型的输出参数的依赖。

4 存在的问题

由于受到根本性计算原理的制约，现有方法仍然面临提高分析精度、提升分析效率、增强方法的可扩展性和智能化程度等一系列挑战性问题。为了更好的解决这些问题，需要在软件模型构建、恶意行为模式提取以及技术极限求解等理论方面有所突破，在检测精确度判断、分析性能提高以及智能化提升等技术方面有进一步提升。

通过以上相关研究工作的分析，可以看出静态检测方法具有一定优势，是未来主要的发展方向。静态分析本质上是“白盒”分析方法，通过对程序内部结构和行为的分析，推断其行为特征，因此更有利于恶意行为内部机理、工作过程和行为模式的探究。当然，为了达到更高的分析精度，静态分析的复杂度可能变得更高，甚至引起“组合爆炸”（如污点爆炸、路径爆炸）等问题，但是这些问题可以通过进一步研究得到较好解决。另外，静态分析方法在安装程序之前就可以对其性质进行判断，如果采用扩展性较好的分析工具，比较适合大规模的应用场景。但是，目前对于静态分析的研究仍然存在一些不足，具体表现在：

（1）对应用程序语义模型的研究仍然不够系统和深入

按照文献[53]提出的安全建模和分析的概念框架，在研究一个系统的安全性时，首先应该建立目标系统的行为模型，这是分析攻击模型和安全性质的基础。由于安卓应用程序是一个开放的、多层次的软件系统，其行为语义较为复杂，如果再考虑到程序中构件属性的组合、构件间隐式消息通信、构件与进程的关系以及权限等安全机制，那么安卓应用程序的行为语义问题就变得更为复杂。就笔者的阅读范围来看，这方面的研究工作较少：工作[54,55]仅对权限系统进行了形式描述，而工作[56]设计了一种类型化语言，描述了 intent 通信，并定义了该语言的操作语义，但是仍然没有涉及构件的配置、构件与进程之间的关系等架构层次的概念模型。总之，针对安卓应用程序行为语义模型的研究工作较少，系统性不强，这促使我们对此需要进一步开展研究。

（2）信息泄露的度量问题需要进一步研究

目前相关研究工作主要集中在信息泄露的“存在性”检测方面，但是存在信息泄露不一定就会造成系统的不安全性，只有泄露达到一定量时，才认为会发生不良行为。只要产生的泄露足够少，我们就能确保系统的安全性。这就启发我们需要进一步研究信息泄露的量化问题，这一问题的研究可以从这样三个方面开展：一是研究主流程序设计语言（如 Java 或其变种）的概率语义问题。为了计算通过程序执行所传播的信息量，需要给出基本语句的概率语义。静态污点分析中常用的污点传播规则没有考虑信息流的量化问题，因此需要进一步扩展。同时，为了高效的计算信息流的量化值，需要发展程序行为的概率精化（probabilistic refinement）和概率互模拟（probabilistic bisimulation）理论，以便为程序行为提供较高的抽象层次，进而提高量化信息流的计算效率。二是研究针对不同安全策略、攻击模型和应用场景的量化信息流模型。对于移动程序的 MIF 量化问题，就是要具体研究：安全级别的划分、信源安全级别的认定、信源的先验概率分布计算、常见攻击模式及其概率分布特性等问题。三是研究和开发自动量化信息分析和计算工具。目前的量化分析工具只能适用于简单程序设计语言和案例，面临运行效率较低，内存消耗较大，扩展性较差等问题，距离实际应用还有不小差距。

（3）现有检测工具的精度、分析性能、可扩展性以及智能化程度仍需进一步提升

现有分析技术和分析工具往往局限于恶意行为检测的某一方面，不可避免的存在误报率较高、结果精度不够等问题，因此需要人工进一步确认和细化分析结果，在面对较大规模的应用时，人工确认的工程量十分庞大。静态分析的不可判定性原理[57]决定了静态分析只能是对程序行为的一种保守逼近，必然会带来一定的误报率。程序分析的内在挑战性问题决定了检测工具的分析精

度和分析性能是一对矛盾体，逼近精度越高，需要的计算资源越多，性能下降越快。因此需要针对不同安全性需要，进行技术极限求解。总之，要使现有检测技术和工具很好的解决实际问题，还需要长期的研究和实践。

5 总结

本文以安卓移动应用程序为研究对象，聚焦恶意隐私信息泄露中的安全信息流检测问题，重点综述了恶意隐私信息泄露的静态分析方法。首先，对安全信息流定义的两个角度进行了区分，重点回顾了基于信息论的安全信息流定义；然后，针对移动应用程序中的信息泄露问题，分析了静态基本技术路线的优缺点，并对典型工作进行了介绍，将其主要分析和检测方法分为两类，并对每类工作进行了详细展开。最后，指出了静态方法存在的挑战性问题以及未来可能的努力方向。尽管这一综述性工作针对移动应用程序的信息泄露问题，但是显然这一问题具有一定普遍性，解决这一问题的思路、方法和技术也可推广到其它领域的安全信息流检测问题之中。

参考文献

[1] Denning, Dorothy E . “A lattice model of secure information flow,” *Communications of the ACM(CACM)* ,vol. 19,no. 5,pp. 236-243,1976.

[2] Denning, Dorothy E. , and P. J. Denning . “Certification of programs for secure information flow,” *Communications of the ACM(CACM)* , vol. 20,no. 7,pp. 504-513,1977.

[3] Mclean J. “Security models and information flow,” *IEEE symposium on security and privacy*,pp. 180-187, 1990.

[4] Andrews, Gregory R , and R. P. Reitma . “An axiomatic approach to information flow in programs,” *Acm Transactions on Programming Languages & Systems* vol. 2,no. 1,pp. 56-76,1980.

[5] Chong, Stephen , and A. C. Myers . “Security policies for downgrading,” *Proceedings of the 11th ACM Conference on Computer and Communications Security,* CCS 2004, Washington, DC, USA, October pp.25-29,2004.

[6] Sabelfeld, Andrei , and D. Sands . “Declassification: Dimensions and principles,” *Journal of computer security*, vol. 17,no. 5,pp. 517-548,2009.

[7] 许艳萍,马兆丰,王中华,钮心忻,杨义先, Android 智能终端安全综述, *通信学报*, 2016, 37(6): 169-184

Y. Xu,Z. Ma,Z. Wang,X. Xin and Y. Yang, “Survey of security for Android smart terminal,” *Journal on Communications*, vol. 37,no. 6,pp. 169-184,2016.

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

S. Qing, “Research Progress on Android Security,” *Journal of Software*,vol. 27,no. 1,pp.45−71,2016.

[9] Tripp, Omer, et al. “ANDROMEDA: accurate and scalable security analysis of web applications,” *Proceedings of the 16th international conference on Fundamental Approaches to Software Engineering* ,vol.7793,pp. 210-225, 2013.

[10] 刘宏月, 范九伦, 马建峰. 访问控制技术研究进展[J]. *小型微型计算机系统*, 2004, 25(1):56-59.

H. Liu, J. Fan and J. Ma, “Research Advanceson Access Control,” *MINI— MICRO SYSTEMS*, vol. 25,no. 1,pp. 56-59,2004.

[11] 彭雪峰, 乔阳,数据加密技术[J]. *佳木斯大学学报（自然科学版）*, 2001, 19(1):40-44

X. Peng, Y. Qiao, “Technology of data encryption,” *Journal of Jiamusi University(Natural Science Edition)*, vol. 19,no.1,pp. 40-44,2001.

[12] AppScan.https://www.ibm.com/cn-zh/security/application-security/appscan

[13] SCA.https://www.microfocus.com/en-us/products/static-code-analysis-sast/overview

[14] EndpointSecurity.https://www.mcafee.com/enterprise/zh-cn/products/endpoint-security.html

[15] Robling Denning, Dorothy Elizabeth, “Cryptography and data security,” Addison-Wesley Longman Publishing Co., Inc., 1982.

[16] Mirzaei, Omid, et al. “Triflow: Triaging android applications using speculative information flows,” *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*,pp. 640-651,2017.

[17] McLean, “Security models and information flow,” *In Proceeding of the 1990 IEEE Symposium on Security and Privacy*, Oakland, California, May 1990.

[18] Goguen, Joseph A. , and José Meseguer. “Security Policies and Security Models,” *1982 IEEE Symposium on Security and Privacy* IEEE, 1982.

[19] D. Sutherland, “A Model of Information,” *In Proc. Of the 9th Nattonal Computer Security Conference*, 1986.

[20] W. III Gray. “Toward a mathematical foundation for informatin flow security,” *In IEEE Security and Privacy*, pp. 21–35, 1991.

[21] A. McIver and C. Morgan, “A probabilistic approach to information hiding,” *In Programming methodology*, pp. 441–460, 2003.

[22] Morgan and Carroll ， “Abstraction, refinement and proof for probabilistic systems,” *Springer Science & Business Media*, 2005.

[23] Clark, D., Hunt, S., Malacaria, P. “Quantitative information flow, relations and polymorphic Types,” J*ournal of Logic and Computation*,vol. 18,no. 2, pp. 181–199, 2005.

[24] Reps T, Horwitz S, Sagiv M. “Precise interprocedural dataflow analysis via graph reachability.” In: Proc. of the 22nd ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages. ACM Press, 1995. 49−61.

[25] S. Arzt, S. Rasthofe , C. Fritz, et al. “FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps,” *Programming language design and implementation*, vol. 49,no. 6,pp. 259-269, 2014.

[26] F. Wei, S. Roy and X. Ou, “Amandroid: A precise and general inter-component data flow analysis framework for security

vetting of android apps," *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*， 2014.

[27] Gordon,I. Michael, et al. "Information flow analysis of android applications in droidsafe," *In NDSS*. vol. 15,no. 201,pp. 110,2015.

[28] Soot. https://github.com/Sable/soot

[29] Rasthofer, Siegfried, S. Arzt, and E. Bodden. "A machine-learning approach for classifying and categorizing android sources and sinks," *In NDSS*. vol. 14, 2014.

[30] J. Huang,Z. Li,X. Xiao, et al. "SUPOR: precise and scalable sensitive user input detection for android apps," *Usenix security symposium*,pp. 977-992, 2015.

[31] L. Li, A. Bartel ,T F. Bissyande, et al. "IccTA: detecting inter-component privacy leaks in Android apps," *International conference on software engineering*, pp. 280-291, 2015.

[32] Bartel A , Klein J , Monperrus M , et al. "Dexpler: Converting Android Dalvik Bytecode to Jimple for Static Analysis with Soot"[J]. 2012.

[33] Octeau, Damien, McDaniel, Patrick, Jha, Somesh, et al. "Effective Inter-Component Communication Mapping in Android: An Essential Step Towards Holistic Security Analysis."[J]. 2013.

[34] D. Octeau , D. Luchaup, M L. Dering, et al. "Composite constant propagation: application to Android inter-component communication analysis," *International conference on software engineering*, pp. 77-88, 2015.

[35] Lu L , Li Z , Wu Z , et al. "CHEX: Statically vetting Android apps for component hijacking vulnerabilities"[C]. Acm Conference on Computer & Communications Security. ACM, 2012.

[36] Elish K O, Yao D D, Ryder B G, et al." A Static Assurance Analysis of Android Applications",2013

[37] Rastogi V , Chen Y , Jiang X . "DroidChameleon: Evaluating Android anti-malware against transformation attacks"[C].Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security. ACM, 2013.

[38] Pan, Xiang, et al. "FlowCog: context-aware semantics extraction and analysis of information flow leaks in android apps." 27th {USENIX} Security Symposium ({USENIX} Security 18), 2018.

[39] 王蕾, 周卿, 何冬杰,等. 面向 Android 应用隐私泄露检测的多源污点分析技术[J]. 软件学报, 2019, 30(02):21-40.

Wang L, Zhou Q, He DJ, et al. " Multi-source taint analysis technique for privacy leak detection of Android Apps." *Journal of Software*, 2019,30(2):211−230 .

[40] A. Sabelfeld, A C. Myers, "Language-based information-flow security," *IEEE Journal on Selected Areas in Communications*, vol. 21,no. 1,pp. 5-19, 2003.

[41] D M. Volpano, C E. Irvine, G. Smith , et al. "A sound type system for secure flow analysis," *Journal of Computer Security*, vol. 4,no. 2,pp. 167-187, 1996.

[42] A. Nanevski, A. Banerjee, D. Garg, et al. "Dependent Type Theory for Verification of Information Flow and Access Control Policies," *ACM Transactions on Programming Languages and Systems*, vol. 35, no. 2, 2013.

[43] W. Huang, Y. Dong, A. Milanova, et al. "Scalable and precise taint analysis for Android," *International symposium on software testing and analysis*,pp. 106-117, 2015.

[44] Chen, Hongxu, et al. "A permission-dependent type system for secure information flow analysis." 2018 IEEE 31st Computer Security Foundations Symposium (CSF). IEEE, 2018.

[45] Lourenço, Luísa, and Luís Caires. "Dependent information flow types." Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. 2015.

[46] Fuchs, Adam P., Avik Chaudhuri, and Jeffrey S. Foster. "Scandroid: Automated security certification of android." 2009.

[47] Wu, Lei, et al. "The impact of vendor customizations on android security." Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. 2013.

[48] Grace, Michael, et al. "Riskranker: scalable and accurate zero-day android malware detection." Proceedings of the 10th international conference on Mobile systems, applications, and services. 2012.

[49] Gibler, Clint, et al. "AndroidLeaks: automatically detecting potential privacy leaks in android applications on a large scale." International Conference on Trust and Trustworthy Computing. Springer, Berlin, Heidelberg, 2012.

[50] Kim, Jinyung, et al. "ScanDal: Static analyzer for detecting privacy leaks in android applications." ,2012

[51] K. Xu, Y. Li, H. Deng, "Iccdetector:Icc-based malware detection on android," *IEEE Transactions on Information Forensics and Security*, vol. 11,no. 6,pp. 1252-1264, 2016.

[52] S. Arzt, E. Bodden, "StubDroid: automatic inference of precise data-flow summaries for the android framework," *.2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*,pp. 725-735, 2016.

[53] J. Bau, C. Mitchell, "Security Modeling and Analysis," *IEEE symposium on security and privacy*,pp. 18-25, 2011.

[54] W. Shin , Kiyomoto S, Fukushima K, et al. "A Formal Model to Analyze the Permission Authorization and Enforcement in the Android Framework," *International conference on social computing*,pp. 944-951, 2010.

[55] D. Geneiatakis, N. Fovino, I. Kounelis, et al. "A Permission verification approach for android mobile applications," *Computers & Security*,pp. 192-205, 2015.

[56] A. Chaudhuri, "Language-based security on Android," *ACM workshop on programming languages and analysis for security*,pp. 1-7, 2009.

[57] Landi W. "Undecidability of static analysis"[J]. ACM Letters on Programming Languages and Systems, 1992, 1(4): 323-337.