

文章编号:2095-6134(2015)06-0807-09

基于有向信息流的 Android 隐私泄露类恶意应用检测方法^{*}

吴敬征^{1,2†}, 武延军^{1,2}, 武志飞¹, 杨牧天¹, 罗天悦¹, 王永吉^{2,3}

(1 中国科学院软件研究所总体部, 北京 100190; 2 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;

3 中国科学院软件研究所基础软件国家工程中心, 北京 100190)

(2014 年 10 月 11 日收稿; 2015 年 3 月 27 日收修改稿)

Wu J Z, Wu Y J, Wu Z F, et al. An Android privacy leakage malicious application detection approach based on directed information flow[J]. Journal of University of Chinese Academy of Sciences, 2015,32(6):807-815.

摘 要 Android 系统占据智能移动终端市场 81.9% 的份额,预计还会持续增长.同时,针对 Android 系统的恶意应用日益增多,Android 恶意应用程序检测技术已经成为安全领域研究的热点问题.本文提出一种基于有向信息流的针对 Android 隐私泄露类恶意应用的检测方法.该方法首先反编译应用程序,分析配置文件中的权限申明;基于隐私点数据集构建隐私数据有向信息流模型;通过在信息流模型中对隐私点的跟踪分析,检测隐私数据是否被发送出去而导致信息泄露.该方法在对 Android 第三方市场的 7 985 个应用程序检测中,发现 357 个恶意应用.通过实验方式验证了检测结果的准确性.结果表明该方法对 Android 隐私泄露类恶意应用具有很好的检测效果.

关键词 Android 应用; 隐私泄露; 有向信息流; 恶意应用检测; 反编译

中图分类号:TP393 **文献标志码:**A **doi:**10. 7523/j. issn. 2095-6134. 2015. 06. 013

An Android privacy leakage malicious application detection approach based on directed information flow

WU Jingzheng^{1,2}, WU Yanjun^{1,2}, WU Zhifei¹, YANG Mutian¹, LUO Tianyue¹, WANG Yongji^{2,3}

(1 Institute of Software, Chinese Academy of Sciences, Beijing 100190, China; 2 State Key Laboratory of

Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China; 3 National

Engineering Research Center for Fundamental Software, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract Android devices occupy 81.9% of the total smart phone market. However, the malicious applications of Android system are increasing, and the detection technology has become the hot topic in security research. We propose a new Android detection approach of privacy leakage malicious application based on directed information flow. This approach first decompiles the application and analyzes the permissions. Then, it builds directed information flow model according to the privacy points. By tracking the flows of the points, the information flows are monitored and the privacy

^{*} 国家自然科学基金(61303057,61170072)和核高基国家科技重大专项(2012ZX01039-004)资助

[†] 通信作者, E-mail: jingzheng08@iscas.ac.cn

leakages are detected. We tested 7 985 applications and detected 357 privacy leakage ones. We analyzed one of the results and confirmed that it was indeed a privacy leakage application. The results show that this new approach has good detection capacity.

Key words Android application; privacy leakage; directed information flow; malicious application detection; decompile

全球信息技术研究和咨询公司 Gartner 2014 年调研报告显示,2013 年全球智能手机销量首次超过功能机,共出货 9.68 亿部同比增长 3.5%,达到总出货量 18.1 亿部的 53.6%。其中,Android 设备占据市场的绝对份额,仅 2013 年第 3 季度,Android 的市场份额就达到 81.9%,远远超过 iOS 系统及 Windows 系统。同时由于厂商、开发人员的推动,以及市场需求的增长,预计 2014 年,Android 系统仍然会统领智能手机市场。

随着移动市场的逐渐成熟,智能手机生态系统也逐渐成熟。Apple 公司的 App Store 和 Google 公司的 Android 市场均提供超百万规模的付费和免费的应用程序。应用程序购买、安装、使用、更新等简化的流程降低了用户的使用门槛,帮助用户随时随地获得和使用海量的应用程序。对于 Android 系统而言,Google 的应用市场提供了相对正规的应用下载途径,但大多数国产手机基本无法使用 Google 服务。统计数据表明,中国大陆地区只有 6% 的应用是通过 Google 市场安装的。这也很大程度上促进了国内第三方市场的繁荣发展。

国内第三方 Android 应用市场的缺乏监管、Android 应用开发、部署模式以及 Permission 权限粒度过粗等问题导致 Android 应用存在大量的安全问题,如隐私泄露、恶意软件、安全模型主观恶意等^[1]。同时,Android 手机集成了大量传感器,第三方应用利用安全缺陷及漏洞可能泄漏用户身份信息、位置信息、周围环境信息等,甚至在 Android 手机后台录制音频、拍摄图片和视频从而造成更严重的安全隐患。

针对 Android 恶意应用快速增长的现状,研究人员提出多种恶意应用自动检测和分类技术,主要分为 2 种基本类型:静态检测和动态检测。静态检测技术通过逆向分析获得 APK 样本源代码进而从中识别恶意代码,检测恶意行为。动态检测技术通过在受控环境中部署和运行 APK 样本,监控其运行状态进而识别恶意行为发现恶意应用。RiskRanker^[2]、DroidChameleon^[3]、AppInk^[4] 以及

Passive Content Leaks^[5]等检测方法,DENDROID^[6]等方法属于静态检测范畴,通过对 App 的静态分析对应用进行分类或者数据挖掘,从而判断是否存在恶意行为,是否会窃取用户手机中 IMEI、IMSI、ICC-ID、Cookies 隐私敏感信息。TaintDroid、DroidBox 等方法为 Android 应用提供了受控的运行环境,以模拟器方式安装、运行样本 App 并分析其行为。基于污点传播技术对隐私数据进行标记跟踪,如果被标记的数据发送出受控环境,则认为存在用户隐私泄露行为。虽然研究人员提出了各种恶意应用检测方法,但是这些方法均不同程度受限于特定环境^[7]。

Android 恶意应用检测是目前学术界研究的热点问题,本文提出一种针对 Android 隐私泄漏类恶意应用的基于有向信息流的静态检测方法。该方法首先反编译 App,分析 Manifest.xml 文件中的权限申明,然后为有疑似权限的应用构建有向信息流模型,标识其中的隐私数据点,通过在信息流模型中对隐私点的跟踪分析,最终判断隐私数据是否会被泄漏。基于该方法在对 Android 第三方市场的 7 985 个应用程序检测实验中,发现 357 个恶意应用。最终通过对检测结果中的 App 验证性分析,证实了结果的真实性,实验证明该方法具有很好的检测性能。

1 Android 隐私泄露恶意应用

Android 系统的普及导致其安全问题成为学术界与工业界关心的热点问题^[8]。由于 Android 自身的开放性与安全机制特性,Android 系统及应用存在着特权处理问题、授权和访问控制机制问题以及开源特性导致的安全漏洞问题。

1.1 Android 恶意应用

根据其功能,恶意应用主要用于入侵、窃取、破坏、隐藏、资费资源消耗等,类型包括病毒、蠕虫、木马、后门、僵尸网络等^[1,9]。Android 恶意应用可以组合上述几种类型,形成独特的恶意行为模式,诱导用户下载恶意代码,以短信、彩信、邮件、即时通

信等形式通过蓝牙、WiFi 或者网络进行传播。Android 恶意软件主要目标是隐私窃取,包括窃取保存在手机中的个人资料或用户账户信息^[10]。

例如,木马 SMS.AndroidOS.FakePlayer.b 伪装成媒体播放器,诱导用户安装。安装完成后,如果用户启动假的应用程序,木马便发送付费和注册服务的 SMS 短信,导致用户费用被窃取到攻击者。Bickford 分析了另外 3 个典型的 Rootkit,第 1 个 Rootkit 允许攻击者远程监听用户的 GSM 保密通话;第 2 个 Rootkit 通过给攻击者发送短信获取用户当前的 GPS 位置信息;最后 1 个 Rootkit 利用 GPS 和蓝牙提供的 Powerintensive 服务,耗尽智能手机上的电池电量,导致拒绝服务。

Papathanasiou 发现了一种通过特定来电次数触发的 Rootkit。该程序基于 Linux 内核可加载模块,触发后打开一个 shell 并允许通过 3G 或者 WiFi 进行反向 TCP 连接,这将导致应用具有 Android 手机的 root 访问权限。通过这种方式,攻击者可以读取设备上的所有短信、来电信息、GPS 信息以及其他应用的帐号及隐私信息,给用户造成严重的安全威胁。

基于攻击者不同的目的,恶意应用可以组合多种恶意行为。例如,ISAM 采用了恶意软件的 6 个不同特点:传播逻辑、僵尸网络控制逻辑、隐蔽收集机密数据、发送大量恶意短信、应用级拒绝服务攻击、网络级拒绝服务攻击。此外,ISAM 能够反向连接僵尸主控机进行更新,从而执行同步攻击等行为。

1.2 Android 隐私泄漏恶意应用模型

Android 手机存储了大量的用户隐私信息,如短信、通讯录、GPS、银行卡、账户信息等,同时由于手机的随身携带性,导致隐私窃取是恶意应用的最典型行为^[11]。从开发人员的角度,Android SDK 提供了操作各种信息与功能的编程接口,声明权限并调用接口就可以实现相应的功能。由于开发人员可以任意调用这些编程接口,因此 Android 手机的安全风险相对较高。

以下面代码段为例,getUserInfo() 函数通过 Android 编程接口获取设备信息、用户信息并发送给远端攻击者。第 4 行代码(TelephonyManager) getSystemService("phone") 获取用户设备;第 6 行通过访问设备 id 即 localTelephonyManager. getDeviceId(), 获得 IMEI 编号;第 8 行获得手机

号;第 10、12、14 行分别获取手机的区号、运营商编号以及 E-mail 帐号。第 17 行将获得的隐私信息构造为参数字符串;第 18 行将字符串以 Post 请求方式向远端服务器发送。

绝大部分 Android 隐私泄露应用都基于这种“隐私获取 + 发送”的模式。其中由隐私获取方式和发送方式的不同而衍生出多种不同的攻击方式。从信息流角度进行抽象,其特征为隐私信息流流出 Android 手机,因此本文方法针对应用的隐私泄露点进行信息流建模,并基于信息流模型进行检测。

```
1. //获取用户信息并被其他函数调用
2. public void getUserInfo() {
3.     if( getPackageManager(). hasSystemFeature
(" android. hardware. telephony" )) {
4.         localTelephonyManager = ( Telephony
Manager) getSystemService( " phone" );
5.         //获取设备 IMEI
6.         this. deviceId = localTelephonyManager.
getDeviceId();
7.         //获取手机号码
8.         str2 = localTelephonyManager. getLine1
Number();
9.         //获取国际长途区号
10.        this. country = localTelephonyManager.
getNetworkCountryIso(). toUpperCase();
11.        //获取运营商名称
12.        this. carrier = localTelephonyManager.
getNetworkOperatorName();
13.        //获取账户 email 信息
14.        this. accounts = AccountManager. get
(getApplicationContext()). getAccounts();
15.        //在 Notifier 的构造函数中对变量赋值,
形成 URL,之后通过请求 URL 将获取的信息传给
远程服务器,泄漏了用户隐私:
16.        public Notifier( String paramString1, String
paramString2, String paramString3, String param
String4, String paramString5 ) {
17.            this. params = ( " appId = 3&deviceId = "
+ paramString1 + " &mobile = " + paramString2
+ " &country = " + paramString3 + " &carrier = "
+ paramString4 + " &email = " + paramString5 );
18.            this. pollURL = ( " http://www. xxx. com/
```


android_notifier/notifier. php?" + this. params); }

2 基于有向信息流的检测方法

Android 恶意应用检测有很多值得借鉴的技术和方法,主要包括静态、动态检测技术,以及基于特征的数据挖掘技术^[12].

2.1 Android 恶意应用检测方法

Grace 等^[2]设计了恶意应用自动检测系统 RiskRanker. 该工具将潜在的风险分成高中低3 种类型. 其中,高风险级恶意应用利用系统级应用的安全漏洞获取 root 级权限破坏手机的完整性和机密性;中风险级恶意应用利用应用漏洞窃取用户隐私数据,消耗用户资费导致经济损失;低风险级恶意应用只收集设备相关的常规信息或者用户信息. 基于此风险分类,RiskRanker 对 Android 市场上的应用执行自动化的二阶段检测. RiskRanker 系统分析的第 1 阶段目标是识别高风险级的应用. 例如,如果应用中包含平台级漏洞利用代码,将被标记为高风险级的应用. 第 2 阶段,RiskRanker 将执行更深入的分析以发现疑似的恶意应用. 例如,如果漏洞利用代码被加密以规避第 1 阶段检测,则需要进一步进行分析. 通过对高中等级风险应用程序的分析,可以大幅减少需要后续核查的可疑应用程序数量. RiskRanker 对 118 318 个应用程序进行检测,发现 3 281 个疑似恶意应用,高中风险等级应用有 2 461 个,真实的恶意应用 718 个,其中 322 个(占总额的 9. 81%)是 0-day 恶意应用程序. 试验结果表明 RiskRanker 系统具有很好的检测效率以及可扩展性.

与 RiskRanker 思路不同,Guillermo 等^[6]提出 DENDROID 分类器检测方法,该方法针对代码结构和应用程序组件进行文本分类挖掘分析,自动分析恶意应用的样本和特征簇. DENDROID 分析包括建模阶段和分析阶段. 建模阶段从所有不同的代码结构中提取恶意代码样本数据集,然后使用向量空间模型将每个样本和特征簇的特征向量相关联. 分析阶段基于建模结果进行文本挖掘,分析待测数据集中的恶意应用. DENDROID 基于代码结构的分析能够快速查找应用特征,利用文本挖掘技术准确快速地检测恶意应用.

2.2 信息流检测技术

基于信息流跟踪的检测技术是恶意程序检测

的经典方法,该方法对分析对象进行信息流建模,根据分析对象代码构建信息流公式,最后使用定理证明器进行证明. 1976 年,Denning^[13]提出语法信息流分析方法. 1990 年,Tsai 等^[14]扩展了 Denning 的方法,增加了语义分析,实现了语义信息流方法.

Denning 的语法信息流分析方法是信息流分析方法的基础,该方法用形式化的方式对信息流建模,如

$$FM = \langle N, P, SC, \oplus, \rightarrow \rangle, \tag{1}$$

其中, $N = \{a, b, \cdots\}$ 是系统中存储元素的集合, N 中的元素可以是文件、代码段、变量或者是其他逻辑元素. $P = \{p, q, \cdots\}$ 是进程的集合,是信息流系统中的响应主体. $SC = \{A, B, \cdots\}$ 是安全等级的集合,Denning 公式中 SC 用于判断操作系统中不同等级之间的操作行为是否合法. 操作符 \oplus 是安全级的运算上确界, $A \oplus B$ 的结果是安全级 A 和 B 的最小公共上界. 操作符 \rightarrow 代表偏序关系,表示信息流在不同等级主体间的流动,只有 A 中的信息允许流向 B 时,才能够表示为 $A \rightarrow B$.

基于以上形式化表述,Denning 给出的语法信息流检测法主要步骤包括以下 3 个部分:

- 1) 信息流抽象. 基于分析对象源码,从每行代码语句中抽象信息流,形成语义. 例如,在赋值语句中,变量之间的关系为“明流”,条件及判断语句抽象为变量之间的“暗流”.
- 2) 设计信息流策略. 将信息流策略应用于系统设计规范上,从而生成信息流公式. 例如,代码“ $y := x$ ”表示信息从变量 x 向 y 流动,在操作系统安全级的约束下,变量 y 的安全级必须支配变量 x 的安全级,即 $y \rightarrow x$.
- 3) 信息流公式证明. 使用定理证明器证明信息流公式是否符合安全级约定,从而判断公式的正确性. 如果公式不能被证明,则可能存在安全隐患;该隐患需要深入分析,从而区分是否是误报,或是真实的安全问题.

3 基于有向信息流的隐私泄露应用检测方法

Denning 的语法信息流分析方法虽然针对安全操作系统隐蔽信道进行分析,但是其思想对分析恶意应用及标识隐私数据点都能够起到分析和检测作用. 本文参考 Denning 的方法,设计基于有向信息流的 Android 隐私泄露恶意应用检测方法.

3.1 基于有向信息流的隐私泄露应用检测方法设计

基于有向信息流的隐私泄露应用检测方法主要分为 3 个阶段,首先是应用程序反编译;其次是基于隐私点数据集构建隐私数据有向信息流模型;最后生成与输出结果,如图 1 所示。

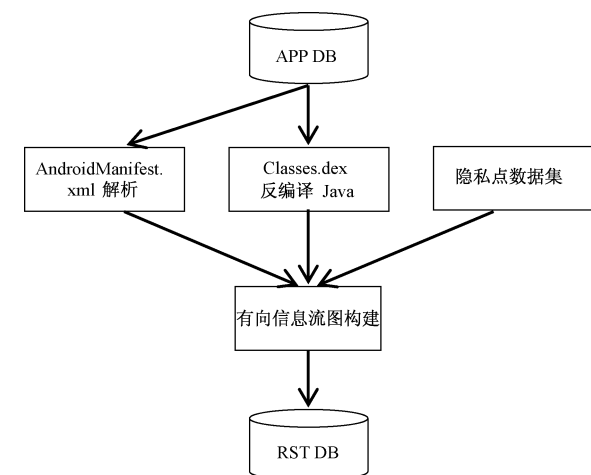


图 1 基于有向信息流的隐私泄露应用检测方法
体系结构图

Fig. 1 Architecture of privacy leakage app detection
based on the directed information flow

应用程序反编译阶段包括对配置文件 Manifest.xml 的解析、权限分析与处理;以及将 Classes.dex 文件编译生成 Java 源代码文件。

隐私点数据集源于对恶意应用的定义以及恶意应用的特征值分析,包括设备信息、短信、来电信息、GPS 信息以及其他应用的帐号及隐私信息。

有向信息流模块基于信息流模型,依据隐私点数据集,分析 Java 源代码中所有调用及读写隐私数据的函数调用,最终形成信息流模型,如果信息流的最顶端是外部环境,则认为存在隐私泄露行为。例如,如果顶端函数是网络连接函数,并将隐私数据作为连接参数传递出去,或者顶端函数是短信发送函数,并将隐私数据作为短信内容发送出去,都认为应用为恶意应用,会导致用户的隐私信息被窃取。

输出模块对检测结果进行整理、统计、分析与输出,最终形成完整的分析报告,列出恶意应用的具体内容。

3.2 Android 恶意应用构建有向信息流模型

参考 Denning 的语法信息流分析实现方法,

Android 恶意应用的信息流模型形式化描述如下

$$AM = \langle P, O, F, \rightarrow \rangle, \quad (2)$$

其中, AM 是 Android 恶意应用的信息流模型; P 是 Android 设备中所有隐私泄露点的集合, 即 $P = \{sms, gps, call, email, \dots\}$; O 是所有 Android 设备中所有外部交互函数的集合, $O = \{netOut, smsSend, writeSDcard, \dots\}$; F 是应用程序中的调用函数和操作函数的集合;操作符 \rightarrow 是调用操作符。上述描述存在以下关系

$$f \rightarrow p, f \in F, p \in P. \quad (3)$$

该关系表示隐私泄露点 p 能够被函数 f 调用和操作,即在 Android 系统中,短信、GPS 等信息能够被正常的应用使用。对于任意隐私点访问的函数调用分支形成有向信息流,即

$$f_n \rightarrow \dots \rightarrow \dots f_2 \rightarrow f_1 \rightarrow p, f_i \in F, p \in P. \quad (4)$$

若该信息流确实存在,即

$$f_n \rightarrow \dots \rightarrow \dots f_2 \rightarrow f_1 \rightarrow p, f_i \in F, p \in P, \quad (5)$$

并且存在

$$f_n \in O, \quad (6)$$

则说明隐私信息 p 最终被发送出 Android 设备,导致用户的隐私泄露,这样的应用被检测为疑似恶意应用并作为结果输出到结果数据库中。

对于同一个隐私点 p ,可能存在多条信息流分支,即

$$\begin{aligned} f_n &\rightarrow \dots \rightarrow \dots f_2 \rightarrow f_1 \rightarrow p, \\ f'_n &\rightarrow \dots \rightarrow \dots f'_2 \rightarrow f'_1 \rightarrow p, \\ f_i &\in F, f'_i \in F, p \in P. \end{aligned} \quad (7)$$

只要其中一条分支的顶端函数能够进行外部发送,即 $f_n \in O \cup f'_n \in O$, 则说明隐私信息 p 最终被发送出 Android 设备,将其作为疑似恶意应用程序并输出到结果数据库中。

3.3 Android 隐私泄露恶意应用信息流特征

数据集 P 是 Android 设备中所有隐私点的集合, 即 $P = \{sms, gps, call, email, \dots\}$, 涉及隐私泄露的应用程序都会直接或者间接地访问、调用、发送该隐私信息,典型的 Android 隐私泄露恶意应用程序特征如表 1 所示。

Android 系统的权限控制机制要求访问特定数据或者设备时必须在 Manifest.xml 文件中进行权限声明,声明之后才能够使用。所以本文方法首先对 Manifest.xml 文件进行解析,分析其中隐私

表 1 Android 隐私泄漏恶意应用程序特征

Table 1 Characteristics of Android privacy leakage malicious app

隐私数据	行为属性
短信信息 SMS	读取短信记录、短信内容、短信收发时间等
联系人 Contacts	读取通讯录信息
即时通信信息 IM	读取即使通信 APP 信息,如微信记录、网银记录等
浏览器信息	读取浏览器访问历史、标签数据、Cookies 信息等
通话记录 Call list	读取通话记录、通话的起止时间等信息
社交网络信息	读取社交网络 APP 数据,如微博、点评类 APP 信息
位置 Location	读取用户粗略及精细的位置信息

信息相关的声明.例如,对于短信和地理位置信息的权限声明如下所示.

//短信相关的权限声明

```
android.permission.BROADCAST_SMS
android.permission.READ_SMS
android.permission.RECEIVE_SMS
android.permission.SEND_SMS
android.permission.WRITE_SMS
```

//地理位置信息

```
android.permission.ACCESS_COARSE_LOCATION
android.permission.ACCESS_FINE_LOCATION
android.permission.ACCESS _ LOCATION _
EXTRA_COMMANDS
android.permission.ACCESS MOCK_LOCATION
android.permission.CONTROL _ LOCATION
_updates
android.permission.INSTALL _ LOCATION
_PROVIDER
```

4 实验分析

本文提出的基于有向信息流的隐私泄露应用检测方法验证数据集为 2013 年 12 月到 2014 年 3 月间从机锋、N 多市场、应用巴士等第三方应用商城获取到的 20 000 个 Android 应用程序,对数据集进行去重处理后共包含 7 985 个独立应用.

4.1 Android 应用反编译结果分析

本文利用 APKTools 和 JD-Core 等逆向分析工具并编写解析与分析程序实现了 Android 应用程序的自动化分析工具.该工具首先抽取应用程序配置文件 Manifest.xml;然后对应用程序中的 Classes.dex 文件反编译生成 Java 源代码文件.

1) Android 应用 Manifest.xml 文件解析

通过对 Android 应用 Manifest.xml 文件的分析,得到隐私信息相关的权限属性结果如图 2 所示.7 985 个 Android 应用程序权限统计结果表明,45.5%的应用程序申请了粗略位置信息数据访问权限;39.5%的应用程序申请了精细地理位置信息数据访问权限;10%左右的应用申请了短信相关的访问权限以及其他权限.同时 96.1%的应用申请了网络访问权限,即 android.permission.INTERNET.

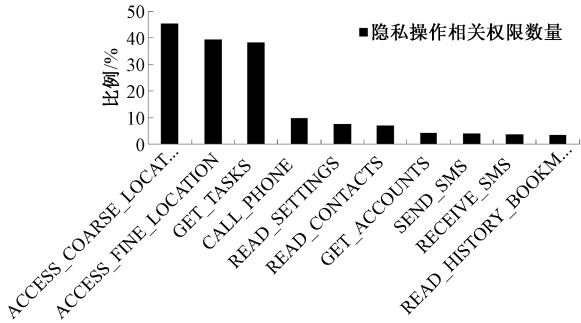


图 2 7 985 个 Android 应用程序隐私相关权限统计结果
Fig. 2 Statistic results of 7 985 Android applications' privacy authority

对于既申请隐私权限又申请短信或网络权限的应用,其都有可能将隐私信息通过短信或者网络发送给攻击者.

2) Classes.dex 文件反编译为 Java 源码

Classes.dex 是 Android 应用中符合 Dalvik 虚拟机规范的可执行二进制文件,是应用的运行实体文件.使用 JD-Core 工具对其进行反编译能够基本还原原程序语义,生成比 smali 代码更可读的 Java 源代码.

例如,某款应用反编译后得到的源代码如图 3 所示,其中程序结构与代码清晰可读,便于后续的信息流构建与深入分析.

4.2 有向信息流模型构建结果分析

Android SDK 规范了应用程序开发 API,对某些资源的访问和使用必须使用该 API 才能够实现.例如,获得 Android 设备 IMEI 号的代码如下所示,首先实例化设备对象,然后使用该设备对象调用函数 getDeviceId().

//实例化设备对象

```
localTelephonyManager = (TelephonyManager)
getSystemService("phone");
```

```
//获取设备 IMEI
this. deviceId = localTelephonyManager.
getDeviceId()
```

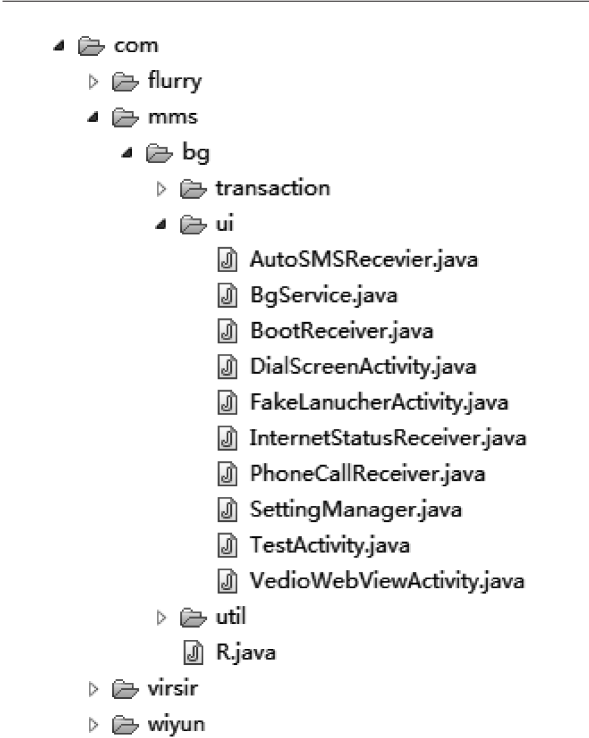


图 3 Classes.dex 文件反编译为 Java 源码

Fig. 3 Decompiling Classes.dex to Java source code

Android 应用大多采用该种调用方式,如表 2 所示. 信息流公式中, $P = \{ getDeviceID(), mPid, getFirstStartTime(), \dots \}$.

表 2 典型的高危 API 列表	
Table 2 The typical high-risk API list	
API	API 源码类
IMEI	localTelephonyManager. getDeviceId();
Phonenum	localTelephonyManager. getLineNumber();
SMSCenter	getSMSCenter();
Handled	String. valueOf(getSMSRoundTotalSend());
Pid	this. mPid;
Installtime	getFirstStartTime();
Sysversion	Build. VERSION. SDK;

根据信息流构建规则

$$f_n \rightarrow \dots \rightarrow \dots f_2 \rightarrow f_1 \rightarrow p, f_i \in F, p \in P, \tag{8}$$

函数 `getDeviceId()` 即为信息流起始节点 p , 其直接的函数调用为

1. `com. mms. ui. SettingManager. savePhoneInfo()`

2. `com. wiyun. ad. b. a`

其中第 1 个调用不仅访问 IMEI 号, 还包含一系列其他隐私信息, 是信息流分支中的第 2 个节点, 其他后续节点如图 4 所示. 其中第 2 个调用为 wiyun 微云移动服务网络的广告函数. 2 个分支构成完整的信息流图, 如果顶层节点中包含网络连接或者短信发送函数, 则认为该应用为疑似应用, 可能会泄漏用户的隐私信息.

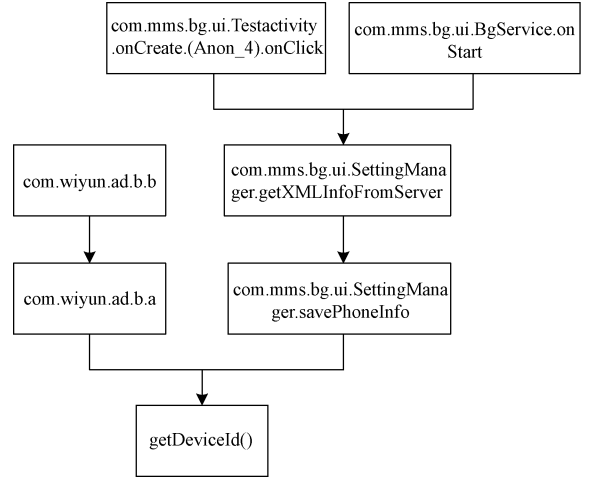


图 4 设备 IMEI 号隐私相关调用函数 `getDeviceID()` 信息流图

Fig. 4 Flow chart of IMEI number privacy related function call of `getDeviceID()`

进一步分析 `getXMLInfoFromServer()`, 发现其调用了 `SeetingManager. openConnection()` 函数, 再次跟踪分析发现该函数是网络连接函数, 如图 5 所示.

对该信息流图再次跟踪, 可见 `getXMLInfoFromServer()` 被以下 2 个函数调用.

1. `com. mms. bg. ui. BgService. onStart()`
2. `com. mms. bg. ui. TestActivity. onCreate. (Anon_4). onClick()`

其中第 1 个调用表示该应用程序服务启动时会调用函数 `getXMLInfoFromServer()`, 该函数调用用户隐私相关函数 `savePhoneInfo()` 并打开连接 `openConnection()` 以 Post 请求的方式将数据发送出去. 其信息流

$$f_n \rightarrow \dots \rightarrow \dots f_2 \rightarrow f_1 \rightarrow p, f_i \in F, p \in P, \tag{9}$$

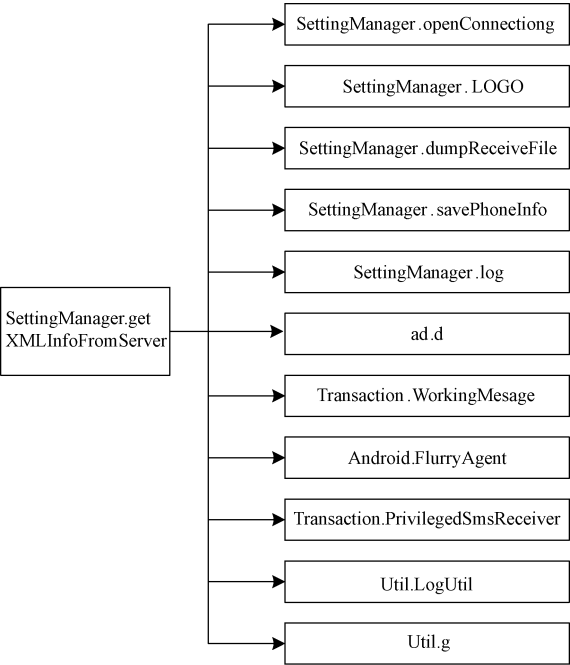


图 5 getXMLInfoFromServer () 函数内部调用函数图
Fig.5 Internal function calls of getXMLInfoFromServer ()

实例化如下所示

//隐私数据信息流图

1. com. mms. bg. ui. BgService. onStart () → com. mms. ui. SettingManager. getXMLInfoFromServer () → com. mms. ui. SettingManager. savePhoneInfo () → localTelephonyManager. getDeviceId ()
2. com. mms. bg. ui. TestActivity. onCreate. (Anon_4). onClick () → com. mms. ui. SettingManager. getXMLInfoFromServer () → com. mms. ui. SettingManager. savePhoneInfo () → localTelephonyManager. getDeviceId ()

并且 getXMLInfoFromServer () 函数调用 openConnection () , 实现了网络传输, 即 $f_n \in O \mid f'_n \in O$, 则认为该应用为疑似隐私泄露应用程序, 输出到结果数据库中.

4.3 基于有向信息流的隐私泄露应用检测结果验证

基于有向信息流的隐私泄露应用检测方法在对 Android 第三方市场的 7 985 个应用程序检测实验中, 发现 357 个恶意应用.

为了验证基于有向信息流的隐私泄露应用检

测方法检测的有效性, 对其输出的部分结果进行实际确认. 例如, 表 3 为上述检测到的恶意应用基本信息.

表 3 检测到的隐私泄漏恶意应用基本信息
Table 3 Basic information of detected privacy leakage malicious apps

名称	com. virsir. android. chinamobile 10086
样本文件格式	APK
样本 MD5 值	955f3696bd38dfe7a49afdd48f31f95
标记	Bgserv_genome

经确认该应用确实为恶意应用, 是 2013 年 6 月 20 日 <http://andrototal.org> 网站爆出的恶意应用程序. 该应用程序获取用户 IMEI、应用安装时间、系统版本等隐私信息, 通过 Post 方式发送给远端恶意服务器, 泄漏用户隐私.

5 结论

目前, 智能移动终端发展迅速, 尤其以 Android 市场占有率最高. 针对 Android 系统的恶意应用也大幅增长, Android 恶意应用程序检测成为安全领域研究的热点问题, 本文提出一种基于有向信息流的针对隐私泄漏类恶意应用的静态检测方法. 该方法首先反编译 App, 分析 Manifest. xml 文件中的权限申明, 然后对于有疑似权限的应用构建有向信息流模型, 标识其中的隐私数据点, 通过在信息流模型中对隐私点的跟踪分析, 从而判断隐私数据是否会被泄漏.

该方法在对 Android 第三方市场的 7 985 个应用程序检测实验中, 发现 357 个恶意应用. 最终通过对检测结果中的 App 验证性分析, 证实了结果的真实性, 实验证明该方法具有很好的检测性能. 今后的研究将从有向信息流模型优化方向技术入手, 进一步提高 Android 隐私泄露类恶意应用的检测效率.

参考文献

[1] La Polla M, Martinelli F, Sgandurra D. A survey on security for mobile devices [J]. IEEE Communications Surveys & Tutorials, 2012, 15 (1) : 446-471.

[2] Grace M, Zhou Y J, Zhang Q, et al. RiskRanker: scalable and accurate zero-day Android malware detection [C] // Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys ' 12). Low Wood Bay, Lake District, UK. 2012; 281-294.

[3] Rastogi V, Chen Y, Jiang X X. DroidChameleon: evaluating Android anti-malware against transformation attacks [C] // Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. Hangzhou, China; ACM. 2013: 329-334.

[4] Zhou W, Zhang X W, Jiang X X. AppInk: watermarking android apps for repackaging deterrence [C] // Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. Hangzhou, China; ACM. 2013: 1-12.

[5] Zhou Y J, Jiang X X. Detecting passive content leaks and pollution in android applications [C] // Proc of the 20th Annual Network and Distributed System Security Symposium (NDSS13). San Diego, California, USA. 2013: 1-16.

[6] 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): 1 104-1 117.

[7] 蒋绍林,王金双,张涛,等. Android 安全研究综述 [J]. 计算机应用与软件, 2012, 29(10): 205-210.

[8] 杨欢,张玉清,胡予濮,等. 基于权限频繁模式挖掘算法的 Android 恶意应用检测方法 [J]. 通信学报, 2013, 34(Z1): 106-115.

[9] 刘潇逸,崔翔,郑东华,等. 一种基于 Android 系统的手机僵尸网络 [J]. 计算机工程, 2011, 37(19): 1-5.

[10] 杨琨,王晓阳,张涛,等. 国内 Android 应用商城中程序隐私泄露分析 [J]. 清华大学学报:自然科学版, 2012, 52(10): 1 420-1 426.

[11] 王浩宇,王仲禹,郭耀,等. 基于代码克隆检测技术的 Android 应用重打包检测 [J]. 中国科学:信息科学, 2014, 44(1): 142-157.

[12] 龚炳江,唐宇敬. Android 平台下软件安全漏洞挖掘方法研究 [J]. 计算机应用与软件, 2014, 31(1): 311-314,333.

[13] Denning D E. A lattice model of secure information flow [J]. Commun ACM, 1976, 19(5): 236-243.

[14] Tsai C R, Gligor V D, Chandrasekaran C S. On the identification of covert storage channels in secure systems [J]. Software Engineering, IEEE Transactions on, 1990, 16(6): 569-580.

(上接第 750 页)

[11] Wang D C, Gong J H, Chen L D, et al. Spatio-temporal pattern analysis of land use/cover change trajectories in Xihe watershed [J]. International Journal of Applied Earth Observation and Geoinformation, 2011, 14:12-21.

[12] Li X, Han J, Kim S, et al. ROAM: Rule- and Motif-based anomaly detection in massive moving object data sets [C] // Proceedings of Siam International Conference on Data Mining. 2007.

[13] Nanni M. Clustering methods for spatio-temporal data [D]. Pisa, Italy: University of Pisa, 2002.

[14] Wang D C, Gong J H, Chen L D, et al. Comparative analysis of land use/cover change trajectories and their driving forces in two small watersheds in the western Loess Plateau of China [J]. International Journal of Applied Earth Observation and Geoinformation, 2013, 21(4):241-252.

[15] Kennedy R E, Cohen W B, Schroeder T A. Trajectory-based change detection for automated characterization of forest disturbance dynamics [J]. Remote Sensing of Environment, 2007, 110(3):370-386.

[16] Lambin M, Land-cover-change Trajectories in Southern Cameroon [J]. Annals of the Association of American Geographers, 2000, 90(3):467-494.

[17] Wang C, Jamison B E, Spicci A A. Trajectory-based warm season grass-land mapping in Missouri prairies with multi-temporal ASTER imagery [J]. Remote Sensing of Environment, 2010, 114(3):531-539.