基于权限组合的Android应用程序安全检测方法与工具研究

李伟芳

北京科技大学

**密　　　　级：\_\_\_\_\_\_\_\_\_\_\_\_\_**

论文题目：基于权限组合的Android应用程序安全

检测方法与工具研究

S20140930

学　　号：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

李伟芳

作　　者：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

软件工程

专业名称：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2016年 12月21日

基于权限组合的Android应用程序安全检测方法

与工具研究

Research on Permission Combination based Security Detection Technique of Android Programs and Its Supporting Tool

研究生姓名：李伟芳

指导教师姓名： 孙昌爱

北京科技大学计算机与通信工程学院

北京100083，中国

Master Degree Candidate：Li Weifang

Supervisor：Sun Chang-ai

School of Computer and Communication Engineering

University of Science and Technology Beijing

30 Xueyuan Road，Haidian District

Beijing 100083，P.R.CHINA

分类号：\_\_\_\_\_\_\_\_\_\_\_\_ 密　　级：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TP311

１０００８

ＵＤＣ：\_\_\_\_\_\_\_\_\_\_\_\_ 单位代码：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**北京科技大学硕士学位论文**

基于权限组合的Android应用程序安全

**论文题目：**

检测方法与工具研究

李伟芳

**作者：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

孙昌爱 教授

北京科技大学

**指 导 教 师： 单位：**

**指导小组成员： 单位：**

**单位：**

**论文提交日期：**2016年 12月21日

**学位授予单位：北 京 科 技 大 学**

致谢

在研究生生活即将结束之际，我不禁回想起这两年半时光中的点点滴滴，在这段无比珍贵的回忆中，同学、师长的帮助和谆谆教诲历历在目，这一路求学历程既艰辛又充满欢乐，正是由于他们的指引和帮助，才助我克服求学路上的种种困难，感谢您们为我的无私付出和帮助。

首先要感谢的是我的导师孙昌爱老师，孙老师严谨的治学态度、以身作则的高尚师德、渊博的学术知识，都始终激励我不断发奋进取，是我未来生活和事业的榜样。孙老师在生活中也教会我许多做人的道理，使我能够更加积极的面对生活和挑战，在此谨向孙老师致以最诚挚的感谢。

感谢实验室中和我一起度过研究时光的同学们，是你们的帮助和鼓励使我克服学习和生活中的困难和挫折。在实验室讨论中我学到很多东西，也培养了团队精神，愿我们友谊长久，感谢你们的陪伴和帮助。

我还要深深的感谢我的父母，他们始终在背后默默地支持着我，他们的期望和对我的信心是我前进路上最大的动力，感谢他们为我的无私付出。

摘要

随着移动互联网络的高速发展以及智能手机的普及，人们广泛使用Android程序进行社交生活和支付消费。Android程序向用户提供便捷服务的同时，也在获取用户的隐私数据。在Android程序数量激增以及Android程序安全质量不能保证的情况下，用户隐私数据的泄露问题变得越来越严重。

本文从Android安全访问机制出发，研究如何检测Android应用程序的隐私数据泄露问题。提出一种基于安全组合的Android应用程序安全检测方法，开发了相应的支持工具。本文取得的主要研究成果如下：

1. **提出一组扩展的导致隐私数据泄露隐患的访问权限组合规则**：在对Android系统安全机制分析的基础上，对Kirin安全规则进行扩展，提出一组扩展的导致隐私数据泄露的危险权限组合规则。
2. **提出一种基于权限组合的Android程序安全检测方法**：该方法自动解析Android程序权限集合，依据危险访问权限组合规则，设置需要检测的隐私数据类型，并跟踪隐私数据在Android系统的信息流动，在函数出口处检测隐私数据的泄露。
3. **设计并实现一个基于权限组合的安全检测工具DroidProtector**：该工具通过分析Android程序包含的危险权限组合检测隐私数据的泄露，通知用户应用程序可能泄露的隐私数据。
4. **对提出的检测方法与工具进行评估**：采用224个应用程序验证了基于权限组合的安全检测方法与工具的有效性，并对DroidProtector进行性能测试。

关键词： Android应用程序，权限组合，程序分析，程序安全

本文通过扩展Kirin安全规则提出一组扩展的访问危险权限组合规则，研究了基于扩展权限组合的Android程序安全检测方法、开发了相应的支持工具DroidProtector。采用本文开发的安全检测方法与支持工具，可以有效减少用户隐私数据的泄露，从而提高Android应用程序的安全性。

**Research on Permission Combination based Security Detection Technique of Android Programs and Its Supporting Tool**

Abstract

With the high speed development of mobile networking and wide adoption of smart mobile phone, Android applications have been widely used for social life and payment. On one hand, Android applications can deliver convenient services; on the other hand, they also access the user’s privacy data. The user’s privacy data leakage is becoming a serious issue, especially when the number of Android applications is significantly increasing and the security of Android applications is not assured.

This thesis proposes how to detect the privacy data leakage of Android applications from the perspective of security access mechanism. As a result, we present an extended permission combination rules based detection method, and develop a supporting tool. The main contributions made in this thesis are as follows:

1. A set of extended access rules of permission combination that may result in privacy data leakage is proposed. Based on an analysis of Android security mechanism, we proposed to extend Kirin security rules and consequently proposed a set of access rules of permission combination.
2. An extended permission combination rules based detection method is proposed, which first automatically extracts the permission set of Android applications, then set privacy data which needs to be detected based on the extended access rules of permission combination, and finally tracks the flow of privacy data and their leakage in the exist of function invoking.
3. A detection tool called DroidProtector is developed which detects the privacy data leakage according to the extended access rules of permission combination, and notify users if this happens.
4. An empirical study has been conducted to evaluate the effectiveness of the proposed detection technique and tool, where 224 Android applications are used as subject programs and the performance of DroidProtector is evaluated.

This thesis proposed an extended set of access rules of permission combination based on Kirin security rules, proposed a detection method based on the extended set of access rules, and developed a supporting tool. With the aid of the presented detection method and supporting tool, one can effectively reduce the risk of privacy data leakage, and thus improve the security of Android applications.

Key Words：Android Application, Access Rule Combination, Program Analysis, Program Security

目录

[致谢 I](#_Toc469646136)

[摘要 III](#_Toc469646137)

[Abstract V](#_Toc469646138)

[插图和附表清单 IX](#_Toc469646139)

[1 引言 1](#_Toc469646140)

[1.1 研究背景与意义 1](#_Toc469646141)

[1.2 研究内容及成果 2](#_Toc469646142)

[1.3 论文组织结构 3](#_Toc469646143)

[2 文献综述 4](#_Toc469646144)

[2.1 国内外研究现状 4](#_Toc469646145)

[2.1.1 国外研究现状 5](#_Toc469646146)

[2.1.2 国内研究现状 6](#_Toc469646147)

[2.2 相关概念及技术 7](#_Toc469646148)

[2.2.1 Android架构 7](#_Toc469646149)

[2.2.2 应用程序组件 8](#_Toc469646150)

[2.3 Android安全机制 11](#_Toc469646151)

[2.4 Kirin安全规则 15](#_Toc469646152)

[2.5 TaintDroid框架 17](#_Toc469646153)

[2.6 小结 18](#_Toc469646154)

[3 基于扩展权限组合的Android程序安全检测方法 19](#_Toc469646155)

[3.1 扩展权限组合规则 19](#_Toc469646156)

[3.2 基于扩展权限组合的安全检测框架 21](#_Toc469646157)

[3.2.1 权限解析器 23](#_Toc469646158)

[3.2.2 隐私数据访问与污点标记 23](#_Toc469646159)

[3.2.3 隐私数据着色和传播 25](#_Toc469646160)

[3.2.4 出口边界检查 26](#_Toc469646161)

[3.3 小结 27](#_Toc469646162)

[4 DroidProtector系统的分析与设计 28](#_Toc469646163)

[4.1 需求分析 28](#_Toc469646164)

[4.2 工具的设计与实现 29](#_Toc469646165)

[4.2.1 系统架构设计 30](#_Toc469646166)

[4.2.2 工具实现 31](#_Toc469646167)

[4.2.3 工具实现关键问题 34](#_Toc469646168)

[4.3 工具演示 35](#_Toc469646169)

[4.4 小结 39](#_Toc469646170)

[5 实验评估 40](#_Toc469646171)

[5.1 实验环境及配置 40](#_Toc469646172)

[5.2 实验对象 40](#_Toc469646173)

[5.3 实验过程 42](#_Toc469646174)

[5.4 实验结果分析 43](#_Toc469646175)

[5.5 性能评估 49](#_Toc469646176)

[5.6 方法的局限性 51](#_Toc469646177)

[5.7 小结 52](#_Toc469646178)

[6 工作总结与展望 53](#_Toc469646179)

[参考文献 55](#_Toc469646180)

[作者简历及在学研究成果 58](#_Toc469646181)

[独创性说明 59](#_Toc469646182)

[关于论文使用授权的说明 59](#_Toc469646183)

[学位论文数据集 1](#_Toc469646184)

插图和附表清单

[图1- 1 智能手机市场份额 1](#_Toc466491263)

[图1- 2 2014年第三季度Android平台手机软件恶意特征分类 2](#_Toc466491264)

[图2- 1 Android架构图 7](#_Toc469324618)

[图2- 2 Activity生命周期状态 9](#_Toc469324619)

[图2- 3 Service生命周期状态 10](#_Toc469324620)

[图2- 4 Android组件交互方式 11](#_Toc469324621)

[图2- 5 Manifest.xml配置文件 14](#_Toc469324622)

[图2- 6 Android系统权限访问机制 15](#_Toc469324623)

[图2- 7 TaintDroid系统架构图 17](#_Toc469324624)

[图3- 1 基于权限组合的Android程序安全检测框架 22](#_Toc469510049)

[图3- 2 基于权限组合分析的流程图 23](#_Toc469510050)

[图3- 3 短信发送流程示意图 26](#_Toc469510051)

[图4- 1 DroidProtector的用例模型 28](#_Toc469412078)

[图4- 2 DroidProtector系统架构 30](#_Toc469412079)

[图4- 3 隐私数据检测类图 32](#_Toc469412080)

[图4- 4 污点添加的本地方法 34](#_Toc469412081)

[图4- 5 污点添加的具体实现 35](#_Toc469412082)

[图4- 6 沃邮箱首页 36](#_Toc469412083)

[图4- 7 DroidProtector首页 36](#_Toc469412084)

[图4- 8 DroidProtector介绍 37](#_Toc469412085)

[图4- 9 DroidProtector系统通知 37](#_Toc469412086)

[图4- 10 隐私数据泄露详细信息 38](#_Toc469412087)

[图4- 11 应用程序权限信息 38](#_Toc469412088)

[图4- 12 应用程序危险权限组合 39](#_Toc469412089)

[图5- 1 209个APP检测分布 43](#_Toc469643632)

[图5- 2 通过规则1泄露数据的应用程序所申请的危险权限组合 46](#_Toc469643633)

[图5- 3 36款应用程序危险权限组合使用分布 47](#_Toc469643634)

[图5- 4 100个正常软件危险权限组合使用分布 47](#_Toc469643635)

[图5- 5 三个系统性能测试的结果 50](#_Toc469643636)

[表2- 2 正常软件与恶意软件危险权限使用情况 16](#_Toc469562289)

[表3- 1 与用户隐私数据相关的危险权限 19](#_Toc469512149)

[表3- 2 两权限组合 20](#_Toc469512150)

[表3- 3 权限、API和污点标签的对应关系 25](#_Toc469512151)

[表5- 1 实验样本集 40](#_Toc469643637)

[表5- 1 实验样本集（续） 41](#_Toc469643638)

[表5- 1 实验样本集（续） 42](#_Toc469643639)

[表5- 2 隐私数据泄露详细信息 44](#_Toc469643640)

[表5- 3 隐私数据泄露详细信息 45](#_Toc469643641)

[表5- 4 权限组合规则检测到的隐私数据泄露 45](#_Toc469643642)

[表5- 5 新增规则检测到的数据泄露 46](#_Toc469643643)

[表5- 6 TaintDroid样本集对比 48](#_Toc469643644)

[表5- 7 2012年与2016年样本泄露数据对比 49](#_Toc469643645)

[表5- 8 2016年泄露隐私数据的程序 49](#_Toc469643646)

[表5- 9 三个系统性能对比 50](#_Toc469643647)

[表5- 10 三个系统时间开销比 50](#_Toc469643648)

1. 引言

介绍课题研究背景及意义、本文研究内容及成果，以及论文组织结构。

* 1. 研究背景与意义

随着移动互联网和智能手机的快速发展，智能手机已经成为人们生活、娱乐和工作的“利器”，如影随形。目前，市场上主流的智能手机操作系统有IOS、Android、Windows Phone、BlackBerry。根据调研公司Kantar Worldpanel ComTech的最新研究报告显示，Android在中国市场占有率达到73%，Android操作系统以它特有的优势占领了智能终端的大部分市场，如图1-1所示。

**图1- 1 智能手机市场份额**

随着第三方应用程序市场的不断扩大以及Android应用开发的较低门槛等原因，Android市场中应用程序的质量良莠不齐。由此针对Android系统的恶意软件越来越多，成为影响Android手机平台安全问题的一大安全隐患，也逐渐成为一个社会热点。各类手机病毒、木马等恶意软件的出现，时刻提醒着人们手机环境的不安全性，使用户手机成为手机病毒和恶意软件的攻击目标[1]。

Android容易受到病毒和木马攻击的一个主要原因是Android平台的开放性。根据国内移动安全厂商网秦发布的《网秦2014年Q3全球手机安全报告》[2]，Android系统在经历了2011年和2012年的广泛应用之后，Android智能手机成为恶意软件攻击的第一大目标。

从图1-2的统计数据可以看出，对用户隐私信息的窃取已经成为恶意应用的一个重要目的。恶意应用程序往往通过各种手段来获得用户的地理位置信息、短信记录、通话内容、通讯录、网页浏览记录、SD卡上存储内容等个人隐私信息。

**图1- 2 2014年第三季度Android平台手机软件恶意特征分类**

Android智能手机的使用量非常庞大，但是用户的安全意识却有待提高。来自美国《消费者报告》的年度《State of the Net》报告中则指出，在美国40%的手机用户没有采取适当地安全措施，2012年有560万人遭遇过账户未经允许被擅自访问等问题[3]。TurstGo公司的分析应用报告显示，Google Play 上3.5%的应用有可能泄露用户隐私或者存在恶意行为；而国内的第三方应用市场91应用市场上该比例则为19.7%[4]。在中国，使用Android手机的用户不能安装Google Play上的应用，从而导致混乱的第三方应用市场的存在，这些市场上应用程序的质量很难保障，对用户隐私数据的安全造成威胁。

本文重点研究Android程序隐私数据泄露问题，在Kirin安全规则的基础上，扩展Kirin安全规则，提出一组危险访问权限组合规则，提出面向Android程序的安全检测方法，开发支持面向Android程序的安全检测工具DroidProtector。通过实例研究与评估，验证本文所提方法的有效性。

* 1. 研究内容及成果

本文对Android程序隐私数据安全问题进行研究，提出一组基于权限组合的Android程序隐私数据安全检测方法，开发针对Android程序隐私数据泄露检测工具。主要研究成果及创新点如下：

1. 提出一组扩展的危险访问权限组合规则。包含与用户隐私数据相关的八种危险权限和13组危险权限组合。
2. 提出一种基于权限组合的Android程序安全检测方法。通过分析与危险权限相关的隐私数据API，对API调用进行污点跟踪检测。
3. 设计并实现基于权限组合的安全检测工具DroidProtector，支持面向Android程序的安全检测技术。
4. 采用实例研究的方式对提出的基于权限组合的Android程序安全检测技术进行评估，并对DroidProtector和TaintDroid进行了性能对比实验。
   1. 论文组织结构

本论文的组织结构安排如下：

1. 引言；
2. 文献综述，包括国内外研究现状，相关概念及技术，Android权限机制、相关工作的介绍；
3. 深入讨论Android程序权限访问机制，提出基于Android程序危险权限组合的检测方法；
4. 讨论面向Android程序的安全检测工具的设计与实现问题，包括需求分析，系统框架，系统实现以及系统的演示；
5. 用开发的工具进行大量实验，进行实验评估；
6. 工作的总结以及未来的展望。
7. 文献综述

介绍国内外研究进展、本文相关的概念。

* 1. 国内外研究现状

智能手机的快速发展，使得智能手机进入到人们生活的方方面面，于此同时由于智能手机的广泛使用，针对Android智能手机的病毒、木马数量也在大肆增长，各大杀毒软件公司也积极投入到针对智能手机安全的研究。

近年来Android操作系统逐渐成为主流的手机操作系统，使得大量的手机病毒针对Android平台。Android操作系统已成为恶意软件选中的新目标。目前Android平台上的主要风险行为有：蓄意扣取手机话费，泄露用户手机信息，传播恶意软件等。这些恶意软件已经严重威胁到手机用户的信息安全，因此国内外人士和相关公司逐渐展开了对针对Android平台的安全研究。

目前，对Android平台的安全问题研究在不断地深入，相关的研究文献也有很多，相关的研究文献也有很多，对Android手机相关的研究主要有一下几个方向[5]：

1. **内核层安全**[6]**：**安全危险主要来自于Linux内核所携带的漏洞。这些安全漏洞往往威胁着整个系统的安全，恶意应用可以利用这些漏洞来提升自己的权限，甚至是获取Root权限，从而肆意窃取数据、破坏系统。
2. **架构层安全：**架构层位于内核层和应用层，系统架构的安全威胁同样来自于安全漏洞，针对这些安全漏洞，相关研究主要集中在一下三个方面：1）SELinux引入系统内核，为系统架构层提供更加安全的服务，实现更加安全的策略；2）完善Android现有的安全机制、细化安全策略也是极为重要的手段；3）Android系统框架API接口的实现如果存在安全问题，将严重威胁到调用该API的应用安全。因此，研究者对API加密方式进行研究，并且在SSL/TLS的加密方式上的研究取得了良好的成果。目前架构层上的安全产品主要有Xen、OKL4、L4Android等[7]。
3. **用户认证机制：**用户认证机制的实现方式多种多样，因此很难在保障安全和低用户参与度中达到良好的平衡，因此，这方面主要在一下三个方面进行了研究：（1）传统认证方式的暴力破解；（2）生物特征的优劣的评判；（3）低参与度的用户认证技术。

针对Android应用程序的安全研究主要集中在恶意应用的检测和应用漏洞的挖掘技术上。目前的检测方案主要有安装时的安全审核[8,9,10]；应用漏洞的挖掘主要集中在利用Fuzzing技术活静态分析技术进行组件漏洞的挖掘[11,12,13,14,15]，或是基于静态分析的安全相关API调用过程的漏洞挖掘[16,17]。目前，对Android平台的安全问题研究在不断地深入。

* + 1. 国外研究现状

目前，国外针对Android的安全性研究起步比起国内的研究比较早，取得了一定的研究进展，其中对Android软件应用的安全挖掘有静态分析方法和动态分析方法。静态分析方法不需要运行程序直接对源码进行分析来发现安全漏洞，依赖定制好的规格或者缺陷模型，主要优点在于可以实现代码的全面覆盖、挖掘具有自动化、高效等优点，动态分析方法不需要获得源码，通过运行应用程序来挖掘漏洞，是通过输入半有效数据触发并挖掘漏洞，具有针对性强、准确率高的优点。针对静态分析和动态分析的优点，国外的许多研究经静态分析和动态分析结合起来，以希望达到更高的准确率。

Enck等人首先研究并介绍了Android的安全机制[18]，并且提出Kirin安全规则，即权限组合安全策略。文中提出如果Android应用程序中申请的权限不能包含Kirin安全规则定义的组合规则，否则就有可能造成用户隐私数据的泄露。并且在应用程序安装时进行静态检查权限，若违反规则拒绝安装。Kirin安全规则考虑的情况不是很全面，并且应用程序一旦含有了某一个或某几个权限组合，系统就会报错。然而有些组合有可能是必须的，而且不会造成用户隐私数据泄露，这就有可能提高误报率。

Enck等人提出TaintDroid扩展Android虚拟机[19]，通过对隐私数据进行着色，当隐私数据离开系统时发出安全警报。该研究没有分析Android程序配置文件的权限信息，并且有着色和脱色的过程，对隐私数据的检测的存在一定的误率报和漏报率。

Chin等人对Android应用间的消息传递进行研究[20]，当消息的发送者和接收者不可信任时，分析消息被窃听、修改、替换和注入等情况，并提供检测工具ComDroid。但是该研究主要重在提出攻击模型，对组合攻击的检测只停留在配置文件的分析上。

Chan等人提出DroidChecker[21]，文中分析应用程序的Manifest.xml文件，并对源代码进行扫描，检测利用数据流泄露用户隐私的应用程序。由于Android应用程序的组件可以在运行时动态注册，该方法没有支持动态注册的组件，另外文中对安全规则定义不全面，因此误报率较高。

Bugiel等人针对应用程序间通信引起隐私泄露的问题设计XManDroid项目[22]，文中为了使应用程序的数据实现隔离，利用权限机制制定了程序间的通信策略，比如，任何具有获取联系人信息的应用程序不能和具有访问网络权限的应用进行通信。XManDroid项目制定的一刀切的策略过于复杂，误报率较高。

Somarriba等人实现了一个针对恶意软件的监控系统[23]，监控应用程序异常函数调用，并且提供一个可视化的图形界面。通过启发式规则，监视应用程序在运行时调用的具有恶意倾向的API钩子函数，并把应用程序的相关信息上传到远程服务器，通过那些钩子函数轨迹绘制一个图形结构，并和已知的恶意软件行为进行匹配。由于Android恶意软件的数量在不断增长，恶意特征也会随之不断增多，对于许多的未知软件不能有效防范，因此该研究需要定时更新数据库，来保证检测的准确性。

* + 1. 国内研究现状

国内针对Android系统的应用程序检测技术相对于国外的研究较晚，国内的研究大部分围绕的是权限机制和特征码匹配来进行的[24]，不仅要在云端维护庞大的恶意软件特征码数据库，而且不能及时检测和防范未知恶意软件。

国内市场上针对恶意软件检测主要有手机卫士、360手机助手、腾讯手机管家、网秦安全等，它们大部分对恶意软件的检测能力比较弱，主要功能是提供内存清理、显示归属地、提供骚扰拦截等功能。

魏松杰等人提出一种基于代码层次结构的系统API调用描述方法[25]。该方法首先提取APK文件中API调用在应用包、对象类、类函数层面的信息，并将这些信息以树形结构表示，通过将不同应用程序的描述树进行逐层对比来计算相似度，揭示恶意应用程序由于在类型 和族群上的差异所带来的API调用特征上的区别，从而为Android应用程序的特征描述和恶意检测提供新的视角。

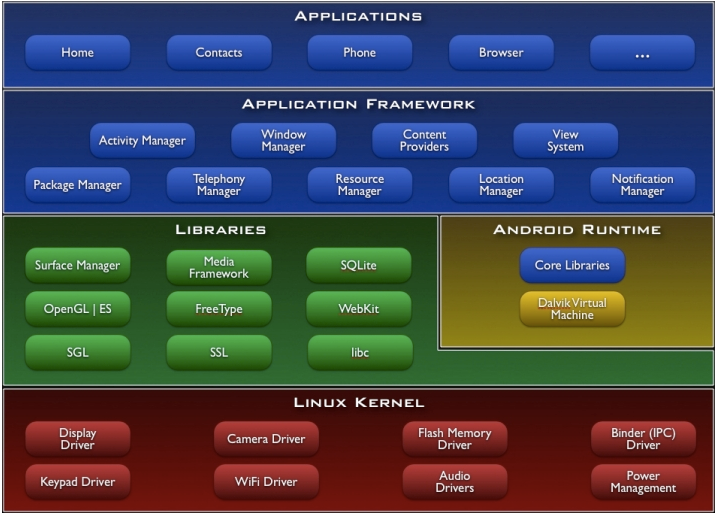
罗维亮等人提出一种检测Android权限滥用的方法DroidPAD[26]，利用权限系统的信息状态实时检测并阻止Android应用程序权限的滥用，尤其是对与短信相关的权限的检测。该方法重点检测与短信权限滥用容易软件，然而更多的程序倾向于使用网络进行恶意操作。

* 1. 相关概念及技术

简要介绍Android的程序架构及安全机制。

* + 1. Android架构

Android系统是一个用于移动设备的功能强大的计算平台，是应用程序的执行环境。Android操作系统具有开放性，吸引了众多的厂商，并且其良好的体验效果受到用户的青睐。但是，由于Android系统的开源性，针对智能设备的恶意软件也层出不穷。为了更好地了解Android系统的安全机制，本节以Android安全性为出发点，简要说明Android系统架构。Android系统架构如图2-1所示，包括应用程序层、应用框架层、系统运行时库层、Linux内核层[27]。



**图2- 1 Android架构图**

1. **应用程序层：**应用程序层提供一系列可以和用户交互的应用程序，包括短信、日历、闹钟、计算器、文件管理、地图、浏览器和联系人管理等，这些应用程序有用户界面并且直接和用户交互。
2. **应用框架层：**应用程序框架层是一个开源的开发平台，提供了大量可以供开发者直接调用的API，是从事Android开发的基础，该层通过对应用程序层提供服务实现其功能。框架层中包含了许多系统和服务，包括包管理器（“Package Manager”）、通知管理器（“Notification Manager”）、活动管理器（“Activity Manager”）、内容提供者（“Content Provider”）、资源管理器（“Resource Manager”）、广播接收器（“Broadcast Receiver”）、视图（“View System”）等。
3. **系统运行时库：**系统运行时库层作为应用程序框架的支撑，为Android系统中的各个组件提供服务，为应用程序框架层提供调用程序库的接口。核心库提供了Java语言API中大多数功能，同时也包含了Android的一心核心API，如android.os、android.net、android.media等。系统运行库层可以分成两部分，分别是系统库和Android运行时，分别介绍如下：

* **系统库**：系统库大部分由C/C++编写，包括SQLite、Webkit、OpenGL等，提供了Java语言API中的功能，供Android各个组件使用，并通过应用程序框架层提供给开发者，是应用程序框架的支撑。Java语言与C/C++的交互是通过JNI（“Java Native Interface”）有机地联系起来的，JNI 提供一系列的接口，允许Java类与使用C/C++编写的应用程序、模块、库进行交互操作。
* **Android运行时**：Android运行时分为核心库和Dalvik虚拟机两部分。Dalvik虚拟机是Android移动设备平台的核心组成部分之一，是Android应用程序Java代码的运行基础，其指令集是基于寄存器的，执行的是专为它设计的一种压缩的Dex字节码文件格式[28]。Android应用程序运行在Dalvik虚拟机里，一个应用程序就是一个单独的进程，并且每一个应用程序都会对应一个独立的Dalvik虚拟机实例[29]，这是Android沙箱机制的实现基础。核心库提供了Java语言API中大多数功能，同时也包含了Android的一心核心API，如android.os、android.net、android.media等。

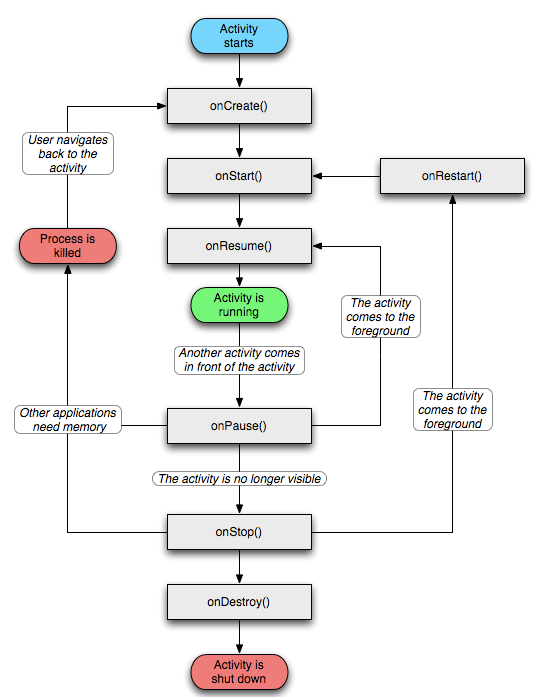
1. **Linux内核：**Android是基于Linux2.6内核，以Linux内核为基础，其核心系统服务如安全性、内存管理、进程管理、网路协议、电源管理、无线通信以及驱动模型都依赖于Linux内核服务实现。

Android系统架构采用分层架构的思想，架构清晰，层析分明，各层有明确的分工，且相互之间不会影响。每层使用下层提供的服务并且向上层提供服务，相互协调完成应用程序的需求的功能。

* + 1. 应用程序组件

应用程序层是与开发人员直接交互的可视化的界面，开发人员在这一层完成Android应用程序的开发。组件是应用程序的组成部分，主要有“Activity”（活动）、“Service”（服务）、“Broadcast Receiver”（广播接收器）、“Content Provider”（内容提供者），每个组件都发挥着自己特定的功能，这些不同的组件的结合组成了一个Android应用程序。Android应用程序在使用组件时，需要在AndroidManifest.xml配置文件中进行注册。

1. **Activity**

应用程序中，一个Activity（活动）通常就是一个单独的用户可见的屏幕，它直接与用户进行交互。开发者可以在Activity添加文本框、按钮、菜单、图片等UI元素。一个应用程序通过多个不同的活动呈现在用户面前。活动的生命周期包括6个过程[30]，如图2-2所示，onCreate()方法在Activity创建时调用，此时Activity还没有获得屏幕焦点，一般在onCreate()方法里完成一些初始化的操作；随后活动会启动，调用onStart()方法，此时活动获得屏幕焦点，（或者活动从后台会到前台是也会调用onStart()方法）；第三个状态是onResume()，此时活动处于运行状态，获得了屏幕焦点，可以和用户交互（或者活动由被覆盖状态回到前台）；onPause()方法是在当前活动呗其他活动覆盖时调用，此时当前活动处于不可执行状态；onStop()方法是在当前活动跳转到新的活动界面或按Home键回到主屏时调用，此时当前活动处于不可见状态，随后调用onStop()进入停滞状态；左后调用onDestory()方法销毁此活动窗口。

**图2- 2 Activity生命周期状态**

Activity之间通过Intent进行通信，Intent是不同组件或相同组件之间相互通信的纽带。Intent包装的是组件相互调用的信息，Intent有两个重要部分：动作和动作对应的数据，Intent负责对动作和动作对应的数据进行描述，Android系统根据Intent的描述，会找到要调用组件。

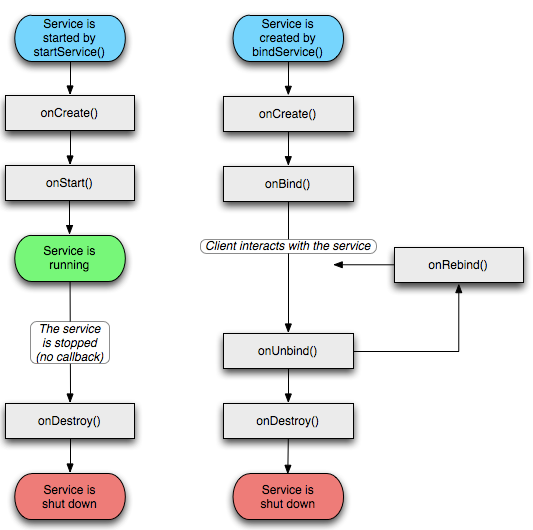
1. **Service**

Service 是一个没有用户界面的长期在后台运行的Android组件，比如后台播放音乐就是用Service来实现的，可以用来开发如监控类程序。Service有两种启动方式，Contex.startService()或Contex.bindService()启动Service[30]，如图2-3所示。Service启动后就会在后台运行。Service是通过通知栏或是发送广播和用户进行交互的。

通过startService()方法启动的Service与调用者没有关系，Service一旦启

动，就会一直在后台运行。停止Service要调用stopService()方法，随后系统会调用onDestory()方法销毁Service。

使用bindService()启动的Service会与调用者绑定，一旦调用者进程停止，Service也会随之关闭。使用此方法启动时, Service首次启动系统先调用Service的onCreate()-->onBind()，如果Service已经启动再次调用不会再触发这2个方法，调用者退出时系统会调用Service的onUnbind()和onDestory()，主动解除绑定可使用Contex.unbindService()，系统依次调用onUnbind()-->onDestory()。



**图2- 3 Service生命周期状态**

1. **BroadcastReceiver**

BroadcastReceiver是用于接收Android系统或应用程序发出的系统级别的广播通知的，例如在收到短信、数据网络可用、电量不足、下载应用程序下载文件完成时，系统就会发送广播通知并做出响应。广播接收器没有用户界面，对用户来说不不可见的，但是BroadcastReceiver可以创建一个状态栏通知，用户可以打开它并启动一个Activity或Serice 来响应它们收到的信息，或者用NotificationManager 来通知用户。BroadcastReceiver也是通过Intent与其他组件进行交互的。

1. **Content Provider**

默认情况下，Android平台应用程序的数据是私有的。为了实现不同应用程序之间交换数据，Android系统了提供了Content Provider这一组件，通过内容提供者就会把应用程序的数据暴露给其他应用。Android通过内容提供者来管理通讯录、图片、视频、等数据，并且可以访问SQLite数据库。其他应用可以通过ContentResolver类从内容提供者中获取或存入数据，实现数据的增、删、改、查。内容提供者通过统一数据访问方式在不同应用程序之间交换数据。

Android应用程序的三大基本组件（Activity、Service、Broadcast Receiver）之间的数据交换和跳转是通过Intent（意图）实现的。Intent的作用是在组件之间传递信息和协调工作，让彼此独立的应用组件整合成一个真正的Android应用程序。Intetn代表着一组系列化的数据，能够在应用程序中传播以实现某些功能。Intent传递数据如图2-4所示。

发送Intent

查

询

发送Intent

Activity

Broadcast Receiver

查

询

Content Provider

发送Intent

**图2- 4 Android组件交互方式**

* 1. Android安全机制

信息安全要考虑私密性、完整性、与可用性等要素，安全威胁通过各种手段，试图利用系统的弱点、缺陷或漏洞，造成破坏信息私密性、完整性与可用性的后果，这些威胁、缺陷与造成的后果组合在一起，就构成了安全威胁。Android拥有一个开放的系统架构，移动计算能力强大，网络互连能力好，Android为了应对来自外界的安全威胁来达到信息安全的目的，就需要强健的安全架构与更为严格的安全规范。Android的系统安全机制贯穿了系统内核、虚拟机、应用程序框架以及应用程序层等体系结构的每个环节，力求保护用户数据、应用程序、设备及无线网络，为应用程序的开发提供了一套安全规范[31]。Android顺其自然地继承了Linux内核安全机制，同时结合移动终端的特点，进行了许多有意的改进和提升。

1. **进程沙箱**

Android系统继承了Linux操作系统基于用户的保护机制，实现应用程序的识别和隔离应用程序资源。Android中的每个应用程序就是一个进程，Android系统在应用程序安装时会为每个应用分配一个独立唯一的用户ID（UID），不同的应用程序就会拥有自己的UID，属于不同的用户，运行在自己独立的进程空间中，互不干扰，也就是将程序置于“沙箱”之中。默认情况下，应用程序没有被分配权限，也只能访问有限的系统资源，并且应用程序之间是不可信的，不能进行通信或交换数据。进程沙箱是在Dalvik虚拟机的基础上实现的一种保护应用程序资源的一种手段。

进程沙箱机制能有效地隔离应用程序，从而保证当某一个应用程序受到恶意软件的攻击时，其他应用程序不会受到来自此应用程序的干扰。因此，恶意软件不能攻击其他应用程序，不会访问其他区应用程序的私有数据或是篡改其他应用程序，从而保护其他应用程序。

1. **进程通信**

Android中一个应用程序运行的自己独立的进程沙箱中，Android应用程序由不同的组件组成，一般都会包括Activity和Service，Activity可以运行在同一个进程中，也可以在不同进程中运行，Service通常会在自己独立的进程中运行。不同进程中的组件要相互协作，实现完整的应用功能，就需要通信，以获取数据与服务。不同应用程序的Activity与Service之间通过操作系统交换数据与服务对象的机制就是进程通信。

Android引入Binder机制实现进程通信，进程通信包括Client、Server、Service Manager和Binder Driver四个部分。Binder驱动工作于内核空间，在用户空间工作的是Service、Client、Service Manager[32,33]。Client与Service之间的通信实际上是通过Binder驱动来完成的，Binder就是把这四个部分联合起来的粘合剂。Service Manager进程在Android虚拟机启动之前启动，然后就会打开Binder 驱动，Client和Service在Binder驱动和Service Manager的基础上进行客户端与服务器之间的通信的。Binder驱动提供文件/dev/binder与用户交互，Client、Server和Service Manager 通过open和ioctl文件操作函数与Binder驱动程序进行通信。

1. **签名机制**

签名机制确保Android应用程序的原始性，安装在Android 设备上的应用程序必须具备数字签名，否则将安装失败。应用程序签名就是为应用程序生成私有密钥和公共密钥对，使用私有密钥签署公共密钥证书[34]。签名的数字证书不需要权威机构来认证，既可以由第三方完成也可以由开发者自己签名。并且具有相同数字签名的应用程序之间会建立信任关系，就可以相互授予权限来访问受限的资源和API；如果应用程序之间声明共享用户ID，他们就会运行在同一进程中，它们就可以安全地共享代码和数据资源。通常情况下，由一个开发者开发的应用程序具有相同的数字签名，共享用户ID，运行在相同的进程中，共享代码和资源。但是，这种签名机制并不能有效地防止恶意软件的攻击，因为这种不需要认证的签名具有随意性，恶意软件的作者可以生成任意的数字证书，而不会被追踪到，也不能防止合法的应用程序被篡改。恶意软件攻击者可以在合法应用程序中添加一些恶意代码逻辑，然后再匿名生成新的数字签名，并发布出去。

1. **权限机制**

进程沙箱提供的隔离机制默认情况下，应用程序没有任何权限，不能访问Android系统保护的设备API和资源，确保了系统的安全[35]。Android应用程序为了能够使用系统API和资源，定义权限机制，例如发送短信，就需要在申请“SEND\_SMS”权限。Android系统中定义了一个系统权限集，每一个权限代表一个系统资源或操作，实现对发送短信、访问网络、录音、通话等关键功能的权限控制，以此来控制对设备中的硬件或数据的访问。

Android应用程序把代码、配置文件、资源文件打包成压缩文件，以APK形式存在。APK文件包含AndroidManifest.mxl配置文件，该文件提供Android程序组件的相关信息、程序依赖库以及运行该应用程序所需要的权限等等。文件中使用<uses-permission>标签来声明所要使用的权限，如图2-5所示。

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:baiduadsdk="http://schemas.android.com/apk/res/com.lixue.word"

package="com.ljh.word"

android:versionCode="1"

android:versionName="1.0" >

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

<application

android:icon="@drawable/logo"

android:label="@string/app\_name" >

<activity

android:name="com.baidu.mobads.AppActivity"

android:configChanges="keyboard|keyboardHidden|orientation" />

<activity

android:name="wordroid.activitys.Main"

android:configChanges="orientation|keyboardHidden"

android:label="@string/app\_name" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>

**图2- 5 Manifest.xml配置文件**

这些申请的权限会在应用安装之前向用户展示。权限的授权方式采用一刀切的方式，用户只有全部同意，否则就拒绝安装。应用程序一旦安装，在设备的存续期间，权限不会改变并且不能单独更改任何一个权限。权限机制是应用框架层的一套重要安全机制，程序AndroidManifest.xml文件中申请的权限需要映射到底层的用户和用户组权限，才能够被执行。当Android系统启动时，SystemServer会首先启动PackageManagerService安装包服务，PackageManagerService服务中的parsePackage函数将对AndroidManifest.xml文件进行解析，然后获取应用程序所申请的权限。系统根据所申请的这些权限，授予应用程序相应权限的组ID。同时应用程序在安装时系统已经赋予了它唯一的用户ID，这样一来，拥有用户ID和组ID的应用程序便可正常启动。Android系统权限机制如图2-6所示。

**APK**

**Android框架**

**应用层**

**框架层**

**内核层**

.java

res

Manifest.xml

系统分配用户

ID（uid）

获得相应权限的组ID（gid）

Fork进程

setgid(gid);

setuid(uid);

ParsePackacge()解析

安装

**图2- 6 Android系统权限访问机制**

* 1. Kirin安全规则

Enck等人通过现有的安全需求技术定义Android系统的权限使用的安全规则，即Kirin安全规则[18]，对Kirin安全访问规则的解释如下：

1. 该规则不允许应用程序使用“SET\_DEBUG\_APP”权限，避免程序开启调试模式，关闭反病毒软件。
2. 该规则保证用户通话内容不能通过网络泄露，从而保护用户的通话记录安全。
3. 该规则保证用户通话内容不能通过网络泄露，从而保护用户的通话记录安全。
4. 该规则保证用户详细地理位置不能通过网络泄露，防止用户身份暴漏或被跟踪。
5. 该规则保证用户粗略地理位置不能通过网络泄露，防止用户身份暴漏或被跟踪。
6. 该规则保护用户收到的信息不被恶意应用篡改。
7. 该规则防止恶意应用编辑短信并发送短信，保护用户的隐私和财产安全。
8. 该规则防止恶意应用替换程序的快捷方式，使之指向恶意软件。
9. 该规则保护系统打电话应用不被恶意应用设置默认处理方式，保护用户和系统的安全。

Kirin安全规则中，“A”规则中，与“SET\_DEBUG\_APP”权限相关的API在是“隐藏”的，不能被第三方程序访问，若Android系统源代码被重新编译，第三方程序则可以使用该权限开启调试模式，关闭反病毒软件。规则“F”是保护接收到的短信不被篡改；规则“H”可以防止一个应用程序创建快捷方式时重定向到一个恶意软件；规则“I” 中“SET\_PERFERRED\_APPLICATION”权限在API level 7以上已经被弃用了，即Android2.1版本以后已经不再使用。通过分析可以得到，规则“A”、“F”、“H”、“I”与本文所讨论的隐私数据泄露没有关系，故本文设计的权限组合规则不包含这几项组合。

Su在文献[36]中研究了不同权限在正常软件和恶意软件中的使用率。如表2-2所示。恶意软件的样本集是120个，正常软件的样本集是415个。例如，“CALL\_PHONE”（打电话）权限在35个正常软件中声明，它的使用频率是 8%；然而，“CALL\_PHONE”权限在恶意软件中的使用率却是32.5%，相比之下，恶意软件申请“CALL\_PHONE”权限的比例大大超过了正常软件。该文献还研究了“CALL\_PHONE”（打电话）、“INTERNET”（访问网络）、“GET\_ACCOUNTS”（获得账户）、“INSTALL\_PACKAGES”（安装应用程序包）、“READ\_CONTACTS”（读取短信）、“READ\_SMS”（读短信）、“READ\_PHONE\_STATE”（读取电话状态）、“SEND\_SMS”（发送短信）这七种权限的使用频率。

从表中可以看出，415个正常程序和120个恶意程序中，八种权限的使用情频率。“READ\_CONTACTS”权限的申请率在恶意应用中是69%，相比其他权限的使用频率都较高，可以看出“READ\_CONTACTS”在恶意应用中占着很重要的比例。因此，在进行危险信息泄露检查时，把“READ\_CONTACTS”权限考虑进去是很有必要的，能够大大提高危险信息泄露的检测率。

**表2- 1 正常软件与恶意软件危险权限使用情况**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Permission** | **Malware（120）** | **Percentage** | **Normal（415）** | **Percentage** |
| CALL\_PHONE | 39 | 0.325 | 35 | 0.08 |
| GET\_ACCOUNTS | 8 | 0.07 | 96 | 0.23 |
| INSTALL\_PACKAGES | 16 | 0.13 | 8 | 0.02 |
| INTERNET | 108 | 0.9 | 395 | 0.95 |
| READ\_PHONE\_STATE | 45 | 0.373 | 80 | 0.19 |
| READ\_CONTACTS | 83 | 0.69 | 200 | 0.48 |
| READ\_SMS | 57 | 0.475 | 28 | 0.07 |
| SEND\_SMS | 74 | 0.62 | 31 | 0.07 |

* 1. TaintDroid框架

Enck等人提出TaintDroid扩展Android虚拟机[19]，对隐私数据的处理是在Android系统的中间件层完成的，在这一层实现了对隐私数据的标记、跟踪，利用Dalvik虚拟机解释器多级数据传播实现对这一层实现了对隐私数据的标记、跟踪。当应用程序执行本地方法时，污点标签通过Binder（进程间通信）传递，最后在网络和短信出口进行污点标签的检测。

解释器

用户空间

内核

隐私源或信任程序

数据标记

（1）

**Dalvik VM**

**解释器**

|  |  |
| --- | --- |
|  |  |
|  |  |

着色表

**Binder IPC库**

Binder Hook

（2）

（3）

**Binder Kernel**

**Binder IPC库**

Binder Hook

（4）

**DalvikVM**

**解释器**

|  |  |
| --- | --- |
|  |  |
|  |  |

着色表

（5）

信任库

隐私标记

非信任程序

（7）

（6）

**图2- 7 TaintDroid系统架构图**

TaintDroid通过对位于运行时库层中的Dalvik虚拟机字节码解析部分进行修改实现了着色数据处理，向隐私数据源添加钩子的方法对隐私数据进行着色。TaintDroid的实现时修改了Dalvik虚拟机的数据处理指令源码，通过在内存栈里向隐私数据分配额外的32位空间存储着色表，在隐私数据源提供模块中采用Hook函数调用着色函数进行处理。图2-7是TaintDroid的着色流程。

图2-7描述了TaintDroid的体系结构，TaintDroid起始认为应用程序是可以被信任的，当有应用程序访问了隐私数据时，该数据就会被标记为隐私源（1），着色标记接口调用Dalvik虚拟机提供的本地方法将着色信息存储在着色表里（2），当隐私数据依据数据流规则在Dalvik虚拟机里传播时会被认为是可信任的程序，当被用于IPC传播时，被修改的库（3）对传送的包Parcel着色，在实现时，Parcel包通过内核的IPC机制进行传播（4）并被远程非信任程序接收。被修改的 Binder库（5）从接收到的Parcel包提取着色信息并存储在自己的着色表里，当有隐私数据被调用时（7），Dalvik虚拟机解释器负责对着色信息的处理。形成一个完整的TaintDroid着色流程。

* 1. 小结

本章介绍了针对Android应用程序数据泄露的检测方法的国内外现状以及Android平台的特点,并详细讨论了Android平台的安全访问机制，并且深入研究Kirin安全规则和TaintDroid检测框架，奠定了本文的理论基础和方法框架。

1. 基于扩展权限组合的Android程序安全检测方法

介绍面向Android系统的危险权限组合规则的分析、安全检测方法和框架。

* 1. 扩展权限组合规则

Android是一个基于Linux的移动开源平台，拥有丰富的资源和API，主要功能是打电话；还有其它辅助功能，例如发短信、访问网络、获得用户地理位置等。执行这些动作的前提是要在Android程序的Manifest.xml配置文件中注册相应的权限。由于Android应用程序在开发的过程可以任意地申请权限，获得系统资源，因而造成用户隐私数据的泄露。有些应用程序在开发的过程中，滥用Android系统的权限资源，申请过多非必要的权限，增大用户隐私数据泄露的可能性。研究发现，任何一个权限的单独使用，不会造成数据的泄露；当多个权限组合使用时，就有可能泄露用户的隐私数据。

在Android官方网站定义了一百多种权限，包括很多涉及用户隐私信息的权限。根据研究发现，单独一个权限不会造成用户隐私数据的泄露，但是，多个权限的组合使用就有可能通过网络或者短信的方式泄露用户的隐私数据。

Android系统中一共设置了140多种权限[37]，用于访问系统资源和受保护的API。本文通过分析Android系统中定义的权限，找出与用户隐私信息密切相关的八种危险权限，如表3-1所示。

**表3- 1 与用户隐私数据相关的危险权限**

|  |  |  |
| --- | --- | --- |
| **编号** | **危险权限** | **描述** |
| 1 | READ\_PHONE\_STATE | 允许应用程序访问电话状态 |
| 2 | ACCESS\_COARSE\_LOCATION | 允许应用程序获取粗略位置 |
| 3 | ACCESS\_FINE\_LOCATION | 允许应用程序获取精确位置 |
| 5 | READ\_SMS | 允许应用读取短信息 |
| 6 | RECORD\_AUDIO | 允许应用记录音频信息 |
| 4 | READ\_CONTACTS | 允许应用读取用户的联系人数据 |
| 7 | GET\_ACCOUNTS | 允许用户获取账户信息 |
| 8 | CAMERA | 允许用户启动相机服务 |

在Android中危险信息泄露的途径包括短信和网络，若应用程序含有“SEND\_SMS”、或“INTERNENT”和其他危险权限组合使用，就有可能造成用户危险信息的泄露。

本文在Kirin安全规则的基础上，通过扩展Kirin安全规则，定义基于权限组合的隐私数据泄露规则，如表3-2所示，其中第九条到第十三条是Kirin安全规则所提出的。

**表3- 2 扩展权权限组合队则**

|  |  |  |  |
| --- | --- | --- | --- |
| **规则** | **权限一** | **权限二** | **权限三** |
| 1 | READ\_PHONE\_STATE | INTERNET |  |
| 2 | READ\_PHONE\_STATE | SEND\_SMS |  |
| 3 | READ\_SMS | INTERNET |  |
| 4 | READ\_CONTACTS | SEND\_SMS |  |
| 5 | READ\_CONTACTS | INTERNET |  |
| 6 | GET\_ACCOUNTS | INTERNET |  |
| 7 | GET\_ACCOUNTS | SEND\_SMS |  |
| 8 | CAMERA | INTERNET |  |
| 9 | WRITE\_SMS | SEND\_SMS |  |
| 10 | READ\_PHONE\_STATE | RECORD\_AUDIO | INTERNET |
| 11 | PROCASS\_OUTGOING\_CALLS | RECORD\_AUDIO | INTERNET |
| 12 | ACCESS\_FINE\_LOCATION | RECEIVE\_BOOT\_COMPLETED | INTERNET |
| 13 | ACCESS\_COARSE\_LOCATION | RECEIVE\_BOOT\_COMPLETED | INTERNET |

每种权限涉及用户的隐私数据类型不同，因此用户带来的危险程度也不尽相同。根据权限的描述，每组权限组合的危险如下所示。

1. “READ\_PHONE\_STATE”和“INTERNE”权限组合，是读取手机状态和访问网络组合。该组合通过读取用户手机状态，获得“IMEI”、“ICCID”、“IMSI”、手机号码，通过网络的形式发送。
2. “READ\_PHONE\_STATE”和“SEND\_SMS”权限组合，是读取手机状态和发送短信组合。该组合通过读取用户手机状态，获得“IMEI”、“ICCID”、“IMSI”、手机号码，通过短信的形式发送出去。
3. “READ\_SMS”和“INTERNET”权限组合，是读取短信和访问网络组合。该组合通过读取短信内容，通过网络形式发送出去。
4. “READ\_CONTACTS”和“SEND\_SMS”权限组合，是读取联系人和发送短信组合。该组合通过读取联系人权限，获得用户存储在手机上的联系人信息，通过短信的形式发送出去。
5. “READ\_CONTACTS”和“INTERNET”权限组合，是读取联系人信息和访问网络组合。该组合通过读取联系人权限，获得用户存储在手机上的联系人信息，通过网络的形式发送出去。
6. “GET\_ ACCOUNTS”和“INTERNET”权限组合，是读取账户信息和访问网络组合。该组合获得手机上存储的账户信息（主要是Gmail账户）并通过网络形式发送出去。
7. “GET\_ACCOUNTS”和“SEND\_SMS”权限组合，是读取账户信息和访问网络组合。该组合获得手机上存储的账户信息（主要是Gmail账户）并通过短信形式发送出去。
8. “CAMERA”和“INTERNET”权限组合，是访问摄像头拍照和访问网络组合。该组合利用照相机拍照并把照片上传到网络。
9. “SEND\_SMS”和“WRITE\_SMS”权限组合，是发送短消息和写短消息组合。该组合在编辑好短消息并发送，不仅会泄露用户隐私数据，还会造成扣费现象的发生。
10. “READ\_PHONE\_STATE”、“RECORD\_AUDIO”和“INTERNET”权限组合，是读取手机状态、录音和访问网络组合。该组合能够在用户打电话的情况下记录用户的语音内容，并且能够通过网络形式发送出去。
11. “PROCESS\_OUTGOING\_CALLS”、“RECORD\_AUDIO”和“INTERNET”权限组合，是管理手机中打出的电话、录音和访问网络组合。该组合能够在用户打电话的情况下记录用户的语音内容，并且能够通过网络形式发送出去。
12. “ACCESS\_FINE\_LOCATION”、“RECEIVE\_BOOT\_COMPLETED”和“INTERNE”权限组合，是获得精确地理位置、开机自启动和访问网络组合。该组合在用户开机以后能够自动开启应用程序，获得用户的精确地理位置并且通过网络形式发送出去。
13. “ACCESS\_COARSE\_LOCATION” “RECEIVE\_BOOT\_COMPLETED”和“INTERNE”权限组合，是获得粗略地理位置、开机自启动和访问网络组合。该组合在用户开机以后能够自动开启应用程序，获得用户的粗略地理位置并且通过网络形式发送出去。

以上是基于Kirin安全规则扩展的危险权限组合规则，其中第九条、第十条、第十一条、第十二条、第十三条是本文保留的Kirin规则。其他的规则在Kirin安全规则的基础上扩展得到。

* 1. 基于扩展权限组合的安全检测框架

Android程序若要使用系统资源或API，需要在程序的Manifest.xml配置文件中申请相应的权限。研究表明，权限的组合使用可能造成用户隐私数据的泄露。由于Android程序权限访问机制特殊性，本文提出一种基于权限组合的隐私数据泄露检测方法，为描述该方法应用于检测Android程序数据泄露的若干关键问题，提出了一种安全检测框架，如图3-1所示。

本文扩展了TaintDroid框架，图中虚线框中的阴影部分是本文所做的扩展和修改，包括权限组合规则、权限解析器和数据标记。该框架中，权限组合规则和着色表是核心组成部分。使用该框架检测Android程序的过程如下：

扩展部分

隐私源或信任程序

|  |  |
| --- | --- |
|  |  |
|  |  |

着色表

**Binder IPC库**

**Binder IPC库**

|  |  |
| --- | --- |
|  |  |
|  |  |

着色表

信任库

隐私标记

非信任程序

**应用层**

**中间件**

**Binder Kernel**

**权限解析器**

**与权限相关的隐私数据标记**

**权限组合规则**

**图3- 1 基于权限组合的Android程序安全检测框架**

1. 通过权限解器获取Android程序的Manifest.xml配置<uses-permission>中申请的权限集合。
2. 依据权限组合规则，与权限集合进行比较，获得程序访问权限组合规则；判断规则是否含有“INTERNET”或“SEND\_SMS”权限，若包含其中任何一个权限，则依据规则提取危险权限，设置隐私数据标记与污点检测选项。
3. 隐私数据污点的传播利用着色表，着色标记接口利用Android系统的JNI机制调用本地方法将着色信息存储在着色表里。
4. 当着色信息用于IPC传播时，Binder IPC库对传送的Parcel包着色。Parcel包通过Binder Kernel 的IPC机制进行传播，并被远程非信任程序接收，Binder库从接收到的Parcel包提取着色信息并存储在自己的着色表里。
5. 当有应用程序调用隐私数据时，Dalvik虚拟机在数据出口处负责对着色信息进行处理、检测。

下面对框架实现的一些关键问题进行介绍。

* + 1. 权限解析器

Android应用程序包是以APK为扩展名的压缩包。在安装包的根目录存放AndroidManifest.xml配置文件，该文件声明了应用程序包名、程序基本组件、以及程序所申请的权限集合等基本信息。

Android应用程序在应用层通过Manifest.xml文件注册权限，权限按照图2-2权限访问机制映射到Android系统框架。权限解析器调用parsePackage()函数解析获得应用程序申请的权限集合，调用getComPerm()方法分析应用程序包含的可能存在泄露隐私数据的权限组合。

否

开始

含有“SEND\_SMS”或“INTERNET”权限

检测危险权限

保存危险权限

是

结束

**图3- 2 基于权限组合分析的流程图**

getComPerm()方法静态分析危险权限组合流程如图3-2所示，结果使用HashMap保存，使用HashMap保存结果的目的是为后续的污点添加提供便利。HashMap键值对保存的数据类型分别是危险权限和污点标签。

* + 1. 隐私数据访问与污点标记

Android框架层包含Android系统核心API，应用程序若要访问Android系统提供的资源和服务，需要调用这些API 来实现其功能。同样地，Android权限若要实现其功能，也需要调用系统API。本文研究了与用户隐私数据相关的API。

1. **与手机信息相关**

通过“READ\_PHONE\_STATE”权限可以得到用户手机的许多信息，比如，手机的状态（通话、响铃、摘机）、电话方位、“IMEI”（移动设备国际省份码）、手机号、“ICCID”（SIM卡序列号）、IMSI（唯一用户ID）等。Android框架下TelephonyManager类封装了与手机信息相关的API。本文研究与手机信息相关的API有getSimSerialNumber()（sim卡序列号）、getLine1Number()（获得手机号）、getDeviceId()（获得手机IMEI）、getSubscriberId()（获得唯一用户ID），这些都关系到用户隐私数据。本文将这些数据标记为污点的来源，放置污点标签。

1. **与地理位置相关**

地理位置信息也是关系到用户隐私的又一重要权限，Android系统框架下的LocationManager类提供了一系列的方法来处理与地理位置相关的信息。LocationManager类中的getLatitude()方法可以获得当前的经度，getLongitude()方法可以获得当前的纬度，调用这两个方法即可以获得当前位置。GpsLocationProvider类中的reportLocation()方法会把altitude和longitude作为参数传入方法内部，因此reportLocation()可以用来放置地理位置的污点标签。

1. **与媒体相关**

本文研究的危险权限中与媒体相关的包括“RECORD\_AUDIO”权限和“CAMERA”权限。在Android框架下也对这一方面信息进行了处理，Android框架下的多媒体文件夹下的AudioRecord类对录制音频进行了详细的处理。AudioRecord类中的read()方法是实现录制音频的主要实现方法；对于访问相机，Android 系统使用的API是Camera.open()，open()方法会调用到框架层Camera类中的takePicture()方法进行拍照。对“RECORD\_AUDIO”和“CAMERA”权限数据的污点标签放置在read()方法和open()方法中。

1. **与通信录、短信相关**

Android系统获取联系人和读取短信是通过ContentResolver这一组件实现的，ContentResolver类提供了query()方法来查询联系人信息和存储在手机上的短信内容。因此在该方法中添加相应的污点标记，当应用程序调用该方法时，这些数据会被打上污点的标记。

1. **与账户相关**

Android设备可以存储来自不同提供商的多个账户，这个功能可以使Android手机记住账户信息，只要输入一次账户密码后，就不用再重新输入账号信息就能直接登录应用（例如QQ、微信、淘宝等）。Android框架层下AccountManager类是管理Android设备中的实现类，然而AccountManager中的方法是由AccountManagerService类实现的，获得账户密码的API是getPassword()，获得账户账号信息API的是getAccount()。因此，本文中会把污点标签放置在AccountManagerService.类中。

根据隐私的数据的类型，本文规定危险权限、污点标签与隐私数据接口调用相关的API的对应关系如表3-3所示。

**表3- 3 权限、API和污点标签的对应关系**

|  |  |  |
| --- | --- | --- |
| **权限** | **相应API** | **污点标签** |
| READ\_PHONE\_STATE | getDeviceId() | IMEI |
| READ\_PHONE\_STATE | getSimSerialNumber() | ICCID |
| READ\_PHONE\_STATE | getLine1Number() | Phone Number |
| READ\_PHONE\_STATE | getSubscriberId() | IMSI |
| ACCESS\_FINE\_LOCATION  ACCESS\_COARSE\_LOCATION | reportLocation() | Location |
| RECORD\_AUDIO | read() | Audio |
| CAMERA | takePicture() | Camera |
| READ\_CONTACTS | query() | Contacts |
| READ\_SMS | query() | SMS |
| GET\_ACCOUNTS | getPassword()、getAccounts() | Accounts |

* + 1. 隐私数据着色和传播

隐私数据的着色处理依据TaintDroid开源项目，着色逻辑是TaintDroid的核心模块，着色表是着色逻辑的实现基础。着色表是Android框架层维护的一个数据结构，为隐私数据污点标记提供接口。

程序中数据类型包括基本数据类型如int，double等，扩展数据类型可能是数组和字符串。例如地理位置信息就是基本double型，短信信息则是扩展数据类型字符串。隐私数据污点标记是一个Native接口，通过JNI调用存储在框架层中着色表里的本地方法，着色表解析传入的参数arg[0]和arg[1]，arg[0]和arg[1]分别代表污点的传入的数据类型和预添加的污点标签，然后建立返回指针，获取隐私数据自带的污点。当应用程序调用API时，这些涉及到用户隐私数据的API调用污点标记接口，进行隐私数据着色，设置污染源。

隐私数据的内部传递在Android系统的中间件层完成，利用TaintDroid实现隐私数据在Android系统内部的传递，当有应用程访问隐私数据时，隐私数据源会被标记，并在中间件层实现数据的跟踪。利用Dalvik虚拟机解释器多级数据传播实现对这一层实现了对隐私数据的标记、跟踪。当应用程序执行本地方法时，污点标签通过Binder（进程间通信）传递，当有数据调用网络接口进行数据的传递时，TaintDroid记录数据。

* + 1. 出口边界检查

Android应用程序与外界交换数据的途径包括短信、网络、蓝牙和建立局域FTP连接。由于蓝牙和建立局域FTP连接只能在短距离范围内传输，因此本文确定的隐私数据传播途径为发送短信和访问网络。

1. **短信出口检查**

短信作为手机的一项基本功能，在用户中已普遍使用，第三方应用程序也可利用短信功能在取得短信发送权限以后在后台静默发送短信[38]，这样不仅造成用户隐私的泄露，也给用户带来经济损失。

开始

结束

GSM

输入参数到短信管理器

调用ISms服务

判断手机类型

CDMA

获取pdu

获取smsc

获取pdu

封装Sms Tracker

发送

**图3- 3 短信发送流程示意图**

若隐私数据泄露出口是短信，在短信发送边界设置检测点。短信发送流程是如图3-3所示。从图中可以看出，发送短信的流程最终都会获取pdu对象，最后通过sendRawPdu()方法发送短信。通过研究sendRawPdu()方法，发现该方法的参数有smsc（短信发送途径）、pdu（短信发送编码形式）、sentIntent（短信成功发送时发送广播）、deliveryIntent（短信成功接收时发送广播）、String destAddr（目的地址）。本文在sendRawPdu()方法中添加一个String content（发送内容）参数，从而可以跟踪短信发送的内容和地址。

1. **网络边界检查**

当Android应用程序在Manifest.xml配置文件中申请了“INTERNET”权限时，需要对网络边界进行检查。通过研究TaintDroid开源项目，我们发现，TaintDroid已经对Android系统的网络接口进行了检查，因此在检查网络接口时，本文使用TaintDroid开源项目。

* 1. 小结

本章介绍了危险权限组合规则的分析、安全检测方法的原理。通过对Android程序Manifest.xml配置文件中的权限抽取、隐私数据API调用与污点标记、隐私数据着色跟踪、出口边界检测四个步骤，跟踪隐私数据信息在Dalvik虚拟机的流动。。

1. DroidProtector系统的分析与设计

DroidProtector是一款检测Android设备中隐私数据泄露的Android应用程序，若检测到受测程序发生隐私数据泄露，则向用户发出系统警示。

讨论DroidProtector的需求，并详细讨论DroidProtector的系统架构与设计。

* 1. 需求分析

DroidProtector的主要功能包括应用程序APP解析、权限组合设置、隐私数据访问、出口边界检查、Service服务。DroidProtector的用例模型如图4-1所示。各个用例的详细描述如下。



**图4- 1 DroidProtector的用例模型**

1. **APP解析用例：**该用例负责对应用程序进行解析。其中包括Manifest文件解析、权限规则检测两个用例。

* **Manifest文件解析：**该用例负责解析应用程序的AndroidManifest.xml文件，获得<uses-permission>标签权限集合。
* **权限组合检测：**该用例依据权限组合规则对应用程序申请的权限集合进行检测。

1. **权限组合设置用例：**该用例负责Android应用程序权限的分析。包含Manifest.xml文件解析、权限组合检测、检查泄漏点权限、查看隐私数据权限四个用例。

* **泄漏点权限检查：**该用例用于检查Android应用程序是否包含“INTERNET”和“SEND\_SMS”权限。依据图3-2的流程图，若含有其中任何一个权限，则需要检测危险权限。反之，不需要进行后续检测。
* **隐私数据权限获取：**该用例负责获得Android应用程序包含的与用户隐私数据相关的危险权限，并保存。

1. **隐私数据访问用例：**该用例负责跟踪隐私数据的传播。

* **隐私数据着色：**该用例负责对与危险权限相对应的API着色，标记污点源。
* **动态污点跟踪：**该用例跟踪的是基本数据类型和对象类型的引用，如，int、float、double等。使用数据流作为传播逻辑，跟踪隐私数据在Dalvik虚拟机的数据流动。

1. **出口边界检查**：该用例对隐私数据的出口进行检测。该用例包含短信监控、网络监控两个用例。

* **短信监控：**该用例负责检测隐私数据的短信出口。
* **网络监控：**该用例负责检测隐私数据的网络出口。

1. **Service服务用例**：该用例处于后台一直运行，检测隐私数据的行为，若发现隐私数据泄露行为，则启动广播接收者发送系统广播。该用例包括读取Logcat日志、检测报告两个用例。

* **读取Logcat日志：**该用例开启多线程读取Inter-filter过滤的隐私数据。
* **检测报告：**该用例用于发送系统广播，通知用户隐私数据泄露类型。
  1. 工具的设计与实现

介绍DroidProtector工具的结构设计、工具的实现和实现工具中的关键问题。

* + 1. 系统架构设计

DroidProtector的系统结构如图4-2所示，包括预处理、危险权限组合分析、运行时检测、结果验证四大模块。

1. **预处理：**该组件负责对Android应用程序安装包APK进行分析并开启Service服务。该模块包括三个子模块：Service服务启动模块、Manifest.xml文件解析模块、危险权限获取模块。

* **Service服务启动：**该模块处于后台，持续监听隐私数据的信息流动。若有隐私数据通过数据泄漏点泄露，Service服务会触发Broadcast Receive（广播接收者），发出系统通知告知用户。
* **Manifest.xml文件解析：**该模块通过调用PackageParser.java文件的parsePackage方法，获得应用程序包含的权限集合，并使用ArrayList保存。

隐私数据标记

权限组合规则

检测报告

结果输出

Logcat日志读取

图例

数据流

模块

文档

引用

Logcat

日志

运行时检测

动态污点跟踪

日志记录

隐私数据着色

预处理

Manifest.xml

文件解析

Service服务启动

危险权限组合分析

泄漏点判断

隐私数据权限获取

危险组合检测

**图4- 2 DroidProtector系统架构**

1. **危险权限组合分析：**该组件负责分析应用程序的危险权限组合，包括三个子模块

* **危险组合检测：**该模块依据危险权限组合规则获得应用程序包含的危险权限组合。
* **泄漏点判断：**该模块负责判断应用程序是否含有“INTERNET”和“SEND\_SMS”权限。若检测到其中任何一个权限，则对隐私数据进行着色跟踪；若不存在，则不需要进行后续检测。
* **隐私数据权限获取：**该模块负责获得与隐私数据相关的危险权限，为后续的隐私数据着色做准备。
* **隐私数据标记：**该模块负责对与八种危险权限相对应的API进行标记。

1. **运行时检测：**该组件负责跟踪应用程序隐私数据在Android系统的信息流动，包括三个子模块：泄漏出口判断模块、隐私数据着色模块、日志记录。

* **隐私数据着色：**该模块利用TaintDroid开源项目，通过Android系统API的调用，添加检测污点标签，对隐私数据进行着色处理。
* **动态污点跟踪：**该模块负责跟踪隐私数据在Dalvik虚拟机的数据信息流动，该模块是在TaintDroid项目的基础上完成的。
* **日志记录：**该模块负责Logcat日志的采集。用于分析隐私数据传播时的数据分析，采集的信息包括隐私数据类型以及数据泄漏点。

1. **结果输出：**该组件由两个模块组成：Logcat日志读取模块、危险值统计模块。

* **Logcat日志读取：**该模块负责读取隐私数据流动过程中的日志信息的采集。日志信息包括隐私数据类型以及数据泄漏点的信息。
  + 1. 工具实现

DroidProtector采用Eclipse开发环境，使用Java语言进行开发。android-sdk-windows和ADT是开发工具必不可少的插件。编译环境是Ubuntu 12.04 LTS 操作系统。首先需要在Ubuntu平台上搭建Android开发环境，采用Linux shell 命令编译Android源代码和工具代码到Android系统，生成自定义的Android ROM，把生成的ROM刷入Android设备。

下面讨论DroidProtector工具实现的关键类图。DroidProtector工具检测隐私数据的泄露行为的类图如图4-3所示。

1. **PackageUtil类：**用于描述Android程序包的基本信息。方法如下：

* **getComPerm()：**权限解析器中的 parsePackage()方法对AndroidManifest.xml配置文件中的<uses-permission>标签进行解析，获取权限集合。该方法对获取到的权限集合进行分析，依据危险权限组合规则，动态抽取程序的危险权限组合，使用HashMap保存，HashMap的key和value分别表示危险权限和污点标签。

**DroidService**

-KEY\_DEST：String

-KEY\_DATA：String

-KEY\_APPNAME:：String

-br：BufferReader

-privacyData：List<PrivacyData>

-logEntry：LogEntry

+readLogEntry(LogEntry logEntry)

+get\_processname(int pid)

+get\_dest(String msg)

+get\_data(String msg)

+sendNotifacation(int pid,String dest,

String data,String appname)

+…..()

**LogEntry**

-pid：int

-tag：String

-message：String

+fromLine(string line)

+……()

**LogcatDevice**

-br：BufferReader

-droidService：DroidService

-intetn：Intent

+open()

+startDroidService(Intent intent )

+close()

1

1

1..\*

1..\*

1

1..\*

**PrivacyData**

-Location：int

-Contacts：int

-PhoneNumber：int

-Camera：int

-Sms：int

-IMEI：int

-IMSI：int

-ICCID：int

-Accouts：int

-Audio：int

-packageUtil：PackageUtil

+addDataString(String str, int tag)

+addDataInt(int val, int tag)

+addDataObjectArray(Object[] array,

int tag)

+……()

**PackageUtil**

-pm：PackageManager

-pi：PackageInfo

-application：String

+getComPerm(Context context,String name)

+……()

1..\*

1

**图4- 3 隐私数据检测类图**

1. **PrivacyData类：**遍历HashMap获取污点标签，用含有“0x”的十六进制整形数值tag标记。Java程序有int、String等基本数据类型和扩展数据类型。以下方法用于为隐私数据打上污点标记。

* **AddDataString()：**该方法是native方法，通过JNI联系Java类与C/C++类中相对应的方法。C/C++类中的方法解析传入参数arg[0]和arg[1]，arg[0]和arg[1]分别代表污点的数据类型str和污点标记tag，然后建立返回污点指针，通过更新String类型的隐私数据返回污点标记。
* **addDataInt()：**该方法是native方法，通过JNI联系Java类与C/C++类中相对应的方法。C/C++类中的方法解析传入参数arg[0]和arg[1]，arg[0]和arg[1]分别代表污点数据类型val和污点标记tag，然后建立返回污点指针，通过更新int类型的隐私数据返回污点标记。
* **addDataObjectArray()：**该方法是native方法，通过JNI联系Java类与C/C++类中相对应的方法。C/C++类中的方法解析传入参数arg[0]和arg[1]，arg[0]和arg[1]分别代表污点数据类型array和污点标记tag，然后建立返回污点指针，通过更新ObjectArray类型的隐私数据返回污点标记。

1. **LogEntry类：**该类用于打开并初始化Android日志系统，方法如下：

* **fromLine()：**污点标签tag通过API调用到添加Android框架层，当污点在Android系统传播时，Logcat日志系统queue\_enyrt\_t队列保存日志信息。该方法通过读取输入参数line，若日志信息中存在tag标签，则使用数组保存一条日志记录。

1. **DroidService类：**该类用于实现后台检测服务。服务一旦开启，后台程序一直检测隐私数据的流动。方法如下：

* **readlogEntry()：**若设备的日志处于打开状态，该方法调用br.readLine()方法读入一行数据，调用fromLine()方法返回污点信息。
* **get\_processneme()：**该方法首先调用getRuntimeAppProcesses()方法返回所有运行中的程序进程。若读取到Logcat日志信息含有污点标记，则通过参数pid得到程序进程名。
* **get\_data()：**使用正则表达式匹配Logcat日志，若检测到含有“0x”的数据，则用ArrayList存储污点标签。
* **get\_dest()：**该方法对数据泄露点进行检测。Android框架层中“SSLOutputStream.write”与网络连接相关的信息，从该信息中解析得到网络发送地址；与短信相关的有“GsmSMSDispatcher.sendSMS”和“CdmaSMSDispatcher.sendSMS”，从该信息中解析的得到短信发送地址。
* **detectData()：**该方法获取需要检测的隐私数据的类型，每种隐私数据类型使用十六进制整形数值存储。
* **sendNotification()：**该方法会调用Android系统服务，得到NotificationManager对象。若有程序发生数据泄露，通过bundle对象传递程序进程名、污点标签、数据泄露地址。

1. **LogDevice类：**该类用于读取Android设备Logcat日志信息，每一个 LogcatDevice都包含一个queue对列。方法如下：

* **Open()：**该方法调用并执行底层linux shell 脚本，返回当前程序的Runtime对象，使用br（BufferReader）对象读取Runtime对象字符输入流。该方法用于打开设备的Logcat日志信息。
* **startDroidService()：**传入参数Intent对象封装了DroidServicede 实例，该方法根据inter-filter中的ACTION\_USER\_PRESENT过滤信息，开启Service服务。
* **close()：**用于销毁Runtime对象，释放内存空间。
  + 1. 工具实现关键问题

本节介绍在实现DroidProtector过程中的一些关键问题。包括两个方面。

1. 设置检测选项

Android源代码libcore目录下的system.java文件可以设置系统属性，调用initSystemProperties方法，通过Properties.put()方法把八种危险权限设置成系统属性。

权限解析器首先判断危险权限，然后保存危险权限。Java库中提供的System.getProperty()方法根据保存的危险权限读取系统属性，获取危险权限。获得的危险权限使用HashMap键值对保存，HashMap键值对保存的的数据类型分别是危险权限和污点标签。

1. 添加污点标签

采用java本地代码为每种数据类型污点的添加提供数据接口。如4-4图

示。Java的基本数据类型包括int、long、short、byte、double等，还包括一些扩展数据类型如数组合字符串。通过图中的Native本地方法，为数据标记提供数据接口。第一个参数代表数据类型，第二个参数代表污点的十六进制标记。

native public static void addString(String str,int tag);

native public static void addObjectArray(Object[] array,int tag);

native public static void addByteArray(byte[] array,int tag);

native public static void addIntArray(int[] array,int tag);

native public static void addShortArray(short[] array,int tag);

native public static void addLongArray(long[] array,int tag);

native public static int addIntArray(int val,int tag);

native public static long addLongArray(long val,int tag);

native public static float addFloatArray(float val,int tag);

native public static double addDoubleArray(double val,int tag);

native public static int getString(String str);

narive public static int getIntArray(int[]; array);

native public static int getFile(int fd);

native public static void log(String str);

……

**图4- 4 污点添加的本地方法**

Android采用分层结构，下层使用C++代码实现，上层使用Java本地代码实现。隐私数据污点标记的实现，需要通过 JNI系统调用，连接Java本地代码和C++代码。图4-4代码的具体实现，需要修改Dalvik虚拟机，在Android系统源代码native/dalvik目录下，以添加string数据类型为例，具体实现如图4-5所示。该方法解析传入参数arg[0]和arg[1]，arg[0]和arg[1]分别代表污点数据类型array和污点标记tag，然后建立返回污点指针，通过更新string类型的隐私数据返回污点标记。

static void Dalvik\_dalvik\_system\_addString(const u4\* args,JValue\* pResylt){

u4 val = args[0];

u4 tag = args[1];

u4\* r1 = (u4\*)&args[2];

u4 r2 = args[3];

\*r1 = (r2 | tag);

RETUEN\_INT(val);

}

**图4- 5 污点添加的具体实现**

此时，对Dalvik虚拟机的修改还不能和Android框架中的API建立联系，需要在API进行污点标记，以“IMEI”数据类型为例，对API标记过程进行描述。

首先，获得Android设备的“IMEI”数据需要调用getDeviceId()方法，该方法在框架层通过Handler来更新UI线程，在该方法中的handleMessage()方法内部调用addString(iImei，IMEI)方法对API进行标记。

* 1. 工具演示

采用一个Android应用程序例子演示DroidProtector的使用。沃邮箱（Android APP）是一款通信社交软件。DroidProtector工具通过对沃邮箱APP的检测，发现隐私数据泄露，验证该工具的有效性。具体使用过程如下。



**图4- 6 沃邮箱首页**

图4-5所示是沃邮箱APP首页，该程序可以登录沃邮箱、qq邮箱、163邮箱、126邮箱。本文认为所有的应用程序是用户信任的。

首先，启动Android系统，在系统功能列表里看到DroidProtector应用程序。打开应用程序后，出现如图4-6所示的界面。点击[Start]按钮启动隐私监控服务，这里，开启的是在后台持续运行的服务，不断检测隐私数据的运行情况。此时，可以退出DroidProtector应用（服务程序仍在后台继续运行），当用户打开其他安装在Android设备上的应用程序时，DroidProtector在后台检测该应用程序的隐私数据流动。



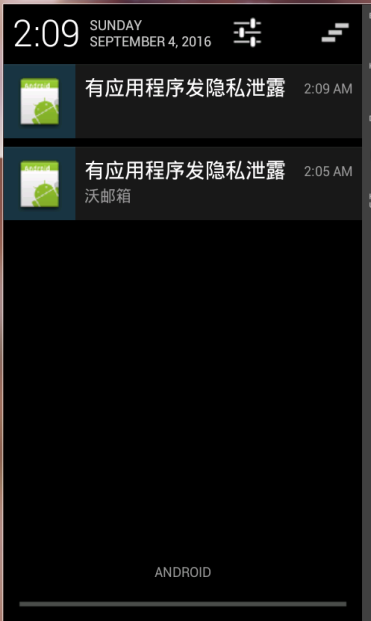
**图4- 7 DroidProtector首页**

其中图4-6的左上角的图案图标是关于工具的一些介绍，点击图案图标按钮，如图4-7所示，包括DroidProtector工具的使用说明、相关术语和工具介绍。点击[About]按钮，介绍DroidProtector工具的开发地点及主要功能；点击[Terminology]按钮，介绍13组危险权限组合及其相应解释；点击[Help]按钮，介绍DroidProtector工具的使用环境的相关配置。



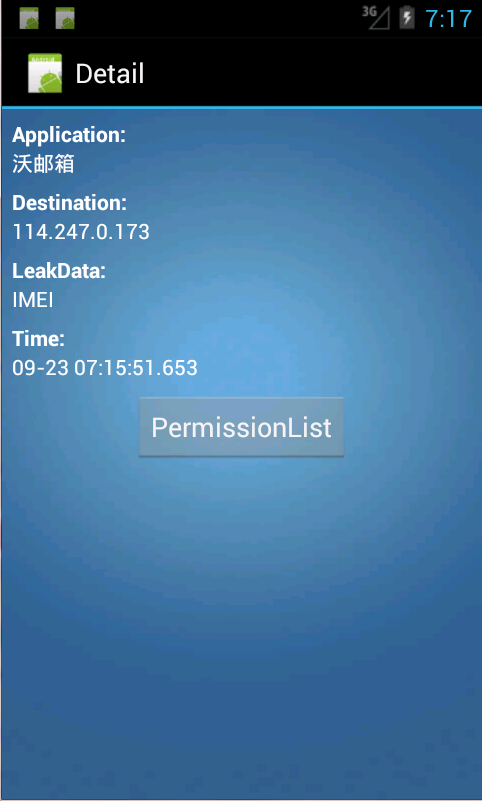
**图4- 8 DroidProtector介绍**

在Android应用功能列表里， 打开沃邮箱APP。DroidProtector开启后台服务实时监测该应用程序的隐私数据流动情况。若有隐私数据发生泄漏，DroidProtector在系统通知栏发出系统通知，包括系统通知名称以及发生隐私数据泄露的应用程序名称，告知用户该应用程序存在隐私数据泄漏，如图4-8所示。



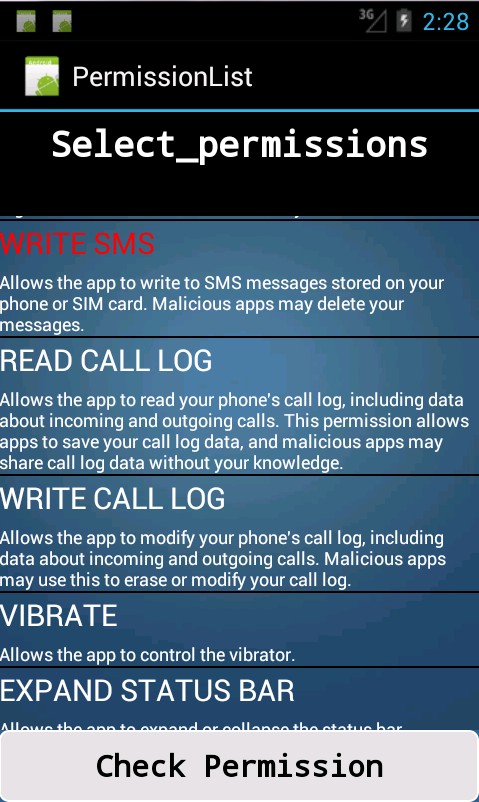
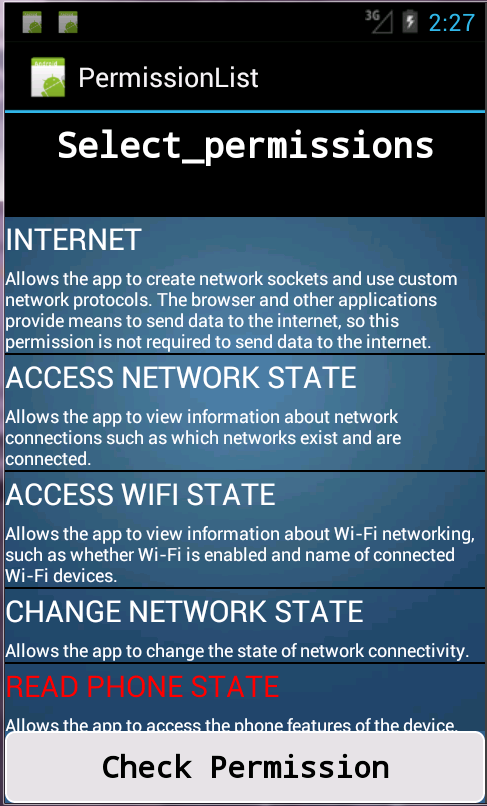
**图4- 9 DroidProtector系统通知**

点击系统通知下拉菜单列表（图4-8右），DroidProtector显示该应用程序泄漏隐私数据的详细信息，如图4-9所示。图4-9显示的有发生隐私数据泄露的程序名称、隐私数据泄露的地址、泄露隐私数据类型以及数据泄露的时间。



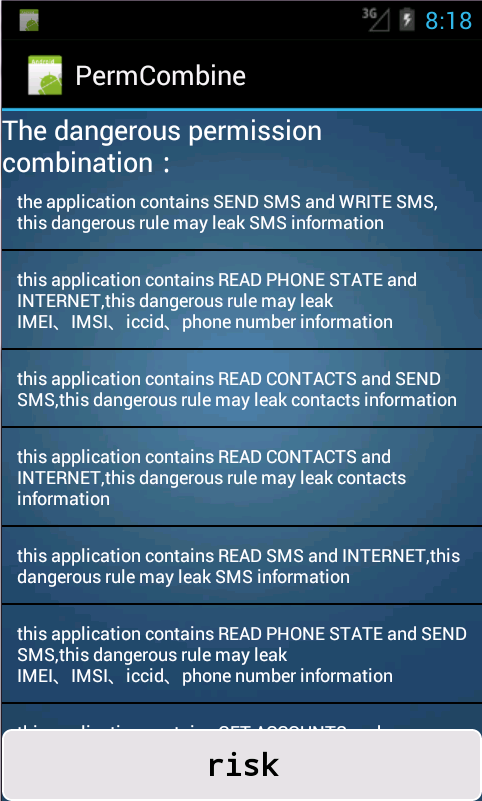
**图4- 10 隐私数据泄露详细信息**

点击图4-8中[PermissionList]按钮，DroidProtector系统把解析到的该应用程序的Manifest.xml配置文件中<uses-permission>标签里的权限信息，通过ListView列表形式展示给用户，并且把涉及到用户隐私数据的危险权限使用红色字体标识。用户就能很直观地查看该应用程申请的权限，如图4-10所示。图4-10中显示的不仅包含该应用程序申请的系统权限，还包含用户自定义的权限。对于一些不明白Android系统中权限作用的用户，DroidProtector在相应权限下方给出权限说明。



**图4- 11 应用程序权限信息**

根据经验得知，任何一个权限都不会造成用户隐私数据的泄露。只有多个权限的组合使用才有可能泄露用户的隐私数据。点击[Check Permission]按钮，DroidProtector根据危险权限组合规则，以列表的形式列出所有可能泄漏隐私数据的危险权限组合，如图4-11所示。



**图4- 12 应用程序危险权限组合**

* 1. 小结

本章详细讨论了基于权限组合的Android应用程序的安全检测支持平台DroidProtector的设计与实现过程。DroidProtector基于Service服务实现隐私数据的泄露检测，若检测到应用程序有泄露隐私数据行为，则发送系统通知警示用户。

1. 实验评估

采用大量的Android应用程序作为实验对象，验证本文提出的基于权限组合Android程序的隐私数据泄露检测方法与工具的有效性，评估所开发的工具的性能。

* 1. 实验环境及配置

实验环境是Android设备，该设备的操作系统版本是Android 4.1.2，CPU是ARM（armeabi-v7a）。检测过程中Android设备需要开启移动网络连接，并且保证待测应用程序的APK安装包在Android设备文件中的“/data/app”目录下。

* 1. 实验对象

本文采用209个应用程序进行实验，同时与TaintDroid项目实验样本集进行了比较，这些样本程序均来自Android市场[39]。应用程序样本名称如表5-1所示。

**表5- 1 实验样本集**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **应用程序名称** | **序号** | **应用程序名称** |
| 1 | cn.apppark.ckj10439668\_044802 | 2 | cn.apps123.shell.redainongyewang\_034806 |
| 3 | cn.etouch.ecalendar\_065939 | 4 | cn.habito.formhabits\_094804 |
| 5 | cn.mama.activity\_054406 | 6 | cn.ppmmt.miliantc\_035219 |
| 7 | com.A17zuoye.mobile.homework\_100339 | 8 | com.android.chrome\_101701 |
| 9 | com.android.dblside\_031718 | 10 | com.anshang.base.CAKIJDYZLFP\_175009 |
| 11 | com.babytree.apps.pregnancy\_021615 | 12 | com.baidu.baidutranslate\_073036 |
| 13 | com.baidu.hao123\_012208 | 14 | com.chinamworld.bocmbci\_1.5.20\_36 |
| 15 | com.chinamworld.main\_120203 | 16 | com.chinat2t17160wc.templte\_054907 |
| 17 | com.chinat2t26183yuneb.templte\_040206 | 18 | com.chuangdong.dongdong\_024504 |
| 19 | com.cxzg.m.bjwpt\_013105 | 20 | com.cxzg.m.hnfsw\_110046 |
| 21 | com.cxzg.m.scjdyd\_013517 | 22 | com.cxzg.m.scmrsc\_104309 |
| 23 | com.cxzg.m.yamsw\_125912 | 24 | com.ddx.tll\_024658 |
| 25 | com.dw.btime\_055203 | 26 | com.evernote\_082902 |
| 27 | com..android.apps.docs.editors.slides\_104951 | 28 | com.google.android.apps.docs\_072947 |
| 29 | com.google.android.marvin.talkback\_011607 | 30 | com.google.android.talk\_014403 |
| 31 | com.google.android.videos\_021505 | 32 | com.haodf.android\_112819 |
| 33 | com.hexin.plat.android\_055917 | 34 | com.hp.android.printservice\_021504 |
| 35 | com.huika.yuejiao\_105951 | 36 | com.ianjia.yyaj\_113303 |
| 37 | com.immomo.momo\_022224 | 38 | com.ionicframework.app993263\_031704 |
| 39 | com.itings.myradio\_045004 | 40 | com.jewelcredit\_124330 |

**表5-1 实验样本集（续）**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **应用程序名称** | **序号** | **应用程序名称** |
| 41 | com.kekeclient\_\_080931 | 42 | com.kingroot.kinguser\_064725 |
| 43 | com.kuaichecaifu.app\_074804 | 44 | com.lanjiejie\_020012 |
| 45 | com.licaidi.finance\_125508 | 46 | com.lydwenhuayichan\_021105 |
| 47 | com.mcs\_102621 | 48 | com.meitu.meipaimv\_010754 |
| 49 | cn.com.cmbc.mbank\_105906 | 50 | com.yxbao.faith\_115106 |
| 51 | cn.idongri.customer\_042529 | 52 | com.zkyc.maqi\_032707 |
| 53 | cn.wps.moffice\_eng\_103406 | 54 | net.easyconn.carman\_103422 |
| 55 | com.android.dazhihui\_113030 | 56 | vStudio.Android.Camera360\_053449 |
| 57 | com.asiainfo.android\_102423 | 58 | cn.com.gf.android\_store\_054305 |
| 59 | com.baidu.browser.apps\_082630 | 60 | cn.wps.moffice\_eng\_065211 |
| 61 | com.chinamworld.main\_052803 | 62 | com.baidu.browser.apps\_060405 |
| 63 | com.chinat2t18101wc.templte\_021507 | 64 | com.chaozh.iReaderFree\_115303 |
| 65 | com.cn21.ecloud\_113027 | 66 | com.gf.client\_025724 |
| 67 | com.cxzg.m.hnlwpt\_063619 | 68 | com.google.android.gm\_052003 |
| 69 | com.cxzg.m.sczsjc\_113809 | 70 | com.guojinbao.app\_034603 |
| 71 | com.duowan.mobile\_120013 | 72 | com.ijinshan.browser\_fast\_032321 |
| 73 | com..android.apps.docs.editors.slides\_070912 | 74 | com.kekeclient\_\_105710 |
| 75 | com.google.android.calendar\_034803 | 76 | com.lemistudio.app.wifiviewer\_091604 |
| 77 | com.google.android.tts\_014712 | 78 | com.miui.video\_020506 |
| 79 | com.hbqh.zscs\_100106 | 80 | com.qiyi.video\_063103 |
| 81 | com.huawei.hidisk\_063911 | 82 | com.sina.weibo\_022403 |
| 83 | com.icbc\_022504 | 84 | com.ss.android.essay.joke\_115104 |
| 85 | com.ishow.dzgyzmhw32\_021707 | 86 | com.tencent.mobileqq\_025708 |
| 87 | com.jxedt\_043823 | 88 | com.tudou.android\_030507 |
| 89 | com.kingsoft\_024504 | 90 | com.wochacha\_055907 |
| 91 | com.lemistudio.app.wifiviewer\_091604 | 92 | com.yd.ydyun30969826.activity\_064845 |
| 93 | com.lydzhangjiajie\_052616 | 94 | com.youdao.note\_063010 |
| 95 | com.meitu.meiyancamera\_024603 | 96 | flipboard.cn\_041204 |
| 97 | com.mobisystems.office\_060904 | 98 | com.babytree.apps.pregnancy\_021615 |
| 99 | com.pplive.androidphone\_080602 | 100 | com.sec.android.app.shealth\_053809 |
| 101 | com.richba.linkwin\_022305 | 102 | cn.apppark.ckj10439668\_044802 |
| 103 | com.sec.android.app.shealth\_053809 | 104 | com.chuangdong.dongdong\_024504 |
| 105 | com.sina.weibo\_124404 | 106 | com.ianjia.yyaj\_113303 |
| 107 | com.storm.smart\_103302 | 108 | com.jxedt\_043823 |
| 109 | com.tencent.mobileqq\_101104 | 110 | com.meitu.meipaimv\_010754 |
| 111 | com.wedding.app\_105609 | 112 | com.sina.weibo\_124404 |
| 113 | com.wochacha\_055907 | 114 | com.google.android.talkback\_011607 |
| 115 | com.xkhouse.fang\_095004 | 116 | com.haodf.android\_112819 |
| 117 | com.youdao.note\_063010 | 118 | com.kingsoft\_024504 |
| 119 | com.yundu.YaLiMaino1102o\_051854 | 120 | com.pingan.lifeinsurance\_122902 |
| 121 | com.zhangxu.chuheridangwu\_043304 | 122 | com.qzone\_110957 |
| 123 | me.chunyu.ChunyuDoctor\_032005 | 124 | com.sdu.didi.psnger\_054503 |
| 125 | pts.PhoneGap.namespace\_myly\_111004 | 126 | com.shuqi.controller\_041004 |
| 127 | cn.boyakids.m\_093902 | 128 | com.smile.gifmaker\_041614 |
| 129 | cn.kuwo.player\_033203 | 130 | com.tencent.mm\_030204 |
| 131 | com.asiainfo.android\_102423 | 132 | com.unionoil.cyb\_023503 |

**表5-1 实验样本集（续）**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **应用程序名称** | **序号** | **应用程序名称** |
| 133 | com.baidu.searchbox\_samsung\_114805 | 134 | com.whatsapp\_011928 |
| 135 | com.eg.android.AlipayGphone\_085202 | 136 | com.xiachufang\_121704 |
| 137 | com.google.android.apps.maps\_023543 | 138 | com.yoloho.dayima\_040952 |
| 139 | com.google.android.youtube\_25y65767 | 140 | com.yuedong.sport\_101709 |
| 141 | com.iflytek.inputmethod\_093704 | 142 | com.yxbao.faith\_115631 |
| 143 | com.jyj.jiaoyijie\_053600 | 144 | fm.qingting.qtradio\_095357 |
| 145 | com.kugou.android\_100803 | 146 | pts.LianShang.yybjp1022\_113905 |
| 147 | com.lyun.user\_101929 | 148 | zte.com.cn.driverMode\_045422 |
| 149 | com.nuomi\_114003 | 150 | cn.habito.formhabits\_094804 |
| 151 | com.sdu.didi.psnger\_054503 | 152 | com.android.dazhihui\_113030 |
| 153 | com.ss.android.article.news\_110104 | 154 | com.baidu.searchbox\_042053 |
| 155 | com.tencent.mm\_113903 | 156 | com.cn21.ecloud\_113027 |
| 157 | com.tixa.industry.search\_172115 | 158 | com.google.android.apps.books\_081824 |
| 159 | com.twl.qichechaoren\_business\_031005 | 160 | com.google.android.tts\_014712 |
| 161 | com.xiaomi.o2o\_032803 | 162 | com.hbqh.zscs\_100106 |
| 163 | com.yoloho.dayima\_114603 | 164 | com.immomo.momo\_061702 |
| 165 | com.zhanyun.ihealth\_022127 | 166 | com.kingroot.kinguser\_064725 |
| 167 | vStudio.Android.Camera360\_053449 | 168 | com.letv.android.client\_055303 |
| 169 | com.google.android.tts\_014712 | 170 | com.nd.android.pandahome2\_064502 |
| 171 | me.chunyu.ChunyuDoctor\_032005 | 172 | com.qq.reader\_014104 |
| 173 | cn.ppmmt.miliantc\_035219 | 174 | com.sohu.newsclient\_050802 |
| 175 | com.hexin.plat.android\_055917 | 176 | com.storm.smart\_103302 |
| 177 | com.immomo.momo\_022224 | 178 | com.tencent.qqpimsecure\_022103 |
| 179 | com.licaidi.finance\_125508 | 180 | com.twl.qichechaoren\_022328 |
| 181 | com.pplive.androidphone\_080602 | 182 | com.wta.NewCloudApp.jiuwei23834\_025508 |
| 183 | com.wuba\_105957 | 184 | com.yiyuan.wangou\_031618 |
| 185 | com.cn21.ecloud\_113027 | 186 | com.youku.phone\_024012 |
| 187 | cn.crzlink.flygift.emoji\_055203 | 188 | fm.qingting.qtradio\_095357 |
| 189 | com.officebuilding\_010604 | 190 | com.google.android.calendar\_034803 |
| 191 | com.qq.reader\_014104 | 192 | com.sec.android.app.shealth\_101406 |
| 193 | com.sds.android.ttpod\_051949 | 194 | cn.apps123.shell.redainongyewang\_034806 |
| 195 | com.sec.android.app.shealth\_101406 | 196 | com.haodf.android\_112819 |
| 197 | com.sinovatech.unicom.ui\_121722 | 198 | com.icbc\_022504 |
| 199 | com.tcza.app\_115802 | 200 | com.kuaichecaifu.app\_074804 |
| 201 | com.tencent.qqlite\_034612 | 202 | com.meitu.meiyancamera\_024603 |
| 203 | com.wetai.mobile.mall.store\_359\_122303 | 204 | com.tencent.qqlite\_034612 |
| 205 | com.wuba\_105957 | 206 | com.asiainfo.android\_102423 |
| 207 | com.ych.car\_030007 | 208 | com.kingsoft\_024504 |
| 209 | com.youloft.calendar\_042838 |  |  |

* 1. 实验过程

按照如下步骤对待测程序进行实验。

1. 利用工具DroidProtector开启监控服务。
2. 在Android设备上运行待测程序。
3. 生成检测实验报告。
   1. 实验结果分析

本文对Android市场上选取的209个应用程序进行了实验研究。实验结果如图5-1所示。209个实验样本集中，7%由于使用第三方的原生库在检测时崩溃导致检测失败，检测到可能发生隐私数据泄露的程序是17%，76%是正常的程序。

**图5- 1 209个APP检测分布**

在209个应用程序中，有15个应用程序由于使用了第三方原生库而运行失败。剩下的194个应用程序中，其中有36个应用程序可能发生隐私数据的泄露。在这36款软件中，18款应用程序向服务器发送一种隐私数据“IMEI”（移动设备国际身份码），其中有3款程序向两个IP地址发送“IMEI”，一款应用程序向三个IP地址发送了“IMEI”；一款程序泄露了“IMSI”（国际移动用户识别码）；4款应用程序同时泄露“IMEI”（移动设备国际身份码）和“ICCID”（SIM卡序列号）；4款应用程序发送地理位置到服务器；5款应用程序泄露用户手机号码；2款程序泄露了用户的联系人信息；还有两款应用

程序分别泄露短信和账户信息。隐私数据泄露详细信息如表5-2所示。

**表5- 2 隐私数据泄露详细信息**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **应用程序包** | **隐私数据类型** | **泄露途径** |
| 1 | com.baidu.appsearch\_112803 | IMEI | 123.125.112.192 |
| 2 | cn.opda.a.phonoalbumshoushou\_010303 | IMEI | 180.97.33.177 |
| 3 | com.duowan.mobile\_034218 | IMEI | 59.78.210.77  180.208.65.100 |
| 4 | com.sohu.sohuvideo\_100004 | IMEI | 2033.156.253.6  115.25.217.49 |
| 5 | sina.mobile.tianqitong\_111005 | IMEI | 123.125.29.147 |
| 6 | tv.pps.mobile\_022738 | IMEI | 123.125.111.108 |
| 7 | com.igoatech.shuashua\_094404 | IMEI | 140.205.155.238 |
| 8 | com.lemistudio.app.wifiviewer\_091604 | IMEI | 123.125.115.123 |
| 9 | com.qzone\_024922 | IMEI | 112.90.140.213  111.30.131.80 |
| 10 | com.shoujiduoduo.ringtone\_061303 | IMEI | 123.125.1144.195  123.125.1144.195 |
| 11 | cn.habito.formhabits\_094804 | IMEI | 121.40.124.50 |
| 12 | com.cn21.ecloud\_113027 | IMEI | 183.56.128.87 |
| 13 | com.lemistudio.app.wifiviewer\_091604 | IMEI | 219.234.85.192  110.173.196.36  123.125.115.123 |
| 14 | com.qq.reader\_014104 | IMEI | 203.156.253.2 |
| 15 | com.sdu.didi.psnger\_054503 | IMEI | 163.177.71.185 |
| 16 | com.youdao.note\_063010 | IMEI | 121.195.178.202  121.195.178.201 |
| 17 | vStudio.Android.Camera360\_053449 | IMEI | 140.205.155.238 |
| 18 | com.trade.eight\_010003 | IMEI | 120.27.131.199 |
| 19 | com.snda.wifilocating\_082102 | IMEI  ICCID | 175.102.131.42 |
| 20 | com.tencent.qqmusic\_021816 | IMEI  ICCID | 117.185.117.170 |
| 21 | com.tmall.wireless\_101934 | IMEI  ICCID | 110.75.231.90 |
| 22 | com.kingroot.kinguser\_064725 | IMEI  ICCID | 182.154.5.162 |
| 23 | com.ifensi.fansheadlines\_124003 | IMSI | 101.201.60.14 |
| 24 | com.iflytek.inputmethod.oem\_101606 | Phone Number | 111.39.232.24 |
| 25 | com.igoatech.zuowen\_051606 | Phone Number | 140.205.155.238 |
| 26 | com.sg.sledog\_042244 | Phone Number | 123.126.50.194  140.205.155.238 |
| 27 | fm.qingting.qtradio\_095357.apk | Phone Number | 140.205.203.93 |
| 28 | com.asiainfo.android\_102423 | Phone Number | 114.247.0.173 |
| 29 | com.moji.mjweather\_045111 | Contacts | 211.152.122.185 |
| 30 | com.taou.maimai\_080502 | Contacts | 223.202.65.166 |
| 31 | com.ximalaya.ting.android\_124202 | Accounts | 121.52.236.26 |
| 32 | com.android.dazhihui\_113030 | SMS | 120.55.151.43  61.164.153.135 |
| 33 | com.zjxfood.activity\_090329 | Location | 140.205.155.238 |
| 34 | com.zjxfood.activity\_090329 | Location | 140.205.155.238 |
| 35 | com.cootek.smartdialer\_081837 | Location | 220.249.245.90 |
| 36 | com.hbqh.zscs\_100106 | Location | 42.62.120.56 |

通过分析表5-2，发现泄露“IMEI”的程序最多，其次是“ICCID”和“Location”。表5-3中列出了泄露每种隐私数据的程序数目及百分比，表中22款泄露“IMEI”的应用程序中，包括泄露“ICCID”的4款应用程序，也就是说22款程序中存在4款应用程序泄露了“IMEI”和“ICCID”两种隐私数据。

**表5- 3 隐私数据泄露详细信息**

|  |  |  |
| --- | --- | --- |
| **泄露隐私数据类型** | **程序数目** | **占总量百分比（194）** |
| IMEI | 22 | 11.0% |
| IMSI | 1 | 0.47% |
| ICCID | 4 | 2.0% |
| Phone Number | 5 | 2.4% |
| SMS | 1 | 0.47% |
| Contacts | 2 | 0.94% |
| Accounts | 1 | 0.47% |
| Location | 4 | 2.0% |

随后本文对36 款应用程序中含有的危险权限组合规则做了统计。权限组合规则1，泄露的隐私数据类型主要包括“IMEI”、“IMSI”、“ICCID”、“Phone Number”，共有28个应用程序违反组合规则1，存在数据泄露行为。组合规则2、组合规则3、组合规则5和组合规则8的数据泄露出口是短信，在样本程序中，没有检测到由短信出口发生隐私数据泄露的应用程序。组合规则3、组合规则5、组合规则6、组合规则12和组合规则13分别是由网络出口引起的隐私数据泄露。统计结果如表5-4所示。

**表5- 4 权限组合规则检测到的隐私数据泄露**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **规则** | **隐私数据类型** | **泄露途径** | **数目** | **百分比（194）** |
| 1 | IMEI、IMSI、ICCID、Phone Number | INTERNET | 28 | 14.43% |
| 2 | IMEI、IMSI、ICCID、Phone Number | SEND\_SMS | 0 | 0% |
| 3 | SMS | INTERNET | 1 | 0.52% |
| 4 | Contacts | SEND\_SMS | 0 | 0% |
| 5 | Contacts | INTERNET | 2 | 1.03% |
| 6 | Accounts | INTERNET | 1 | 0.52% |
| 7 | Accounts | SEND\_SMS | 0 | 0% |
| 8 | Camera | INTERNET | 0 | 0% |
| 9 | SMS | SEND\_SMS | 0 | 0% |
| 10 | Audio | INTERNET | 0 | 0% |
| 11 | Audio | INTERNET | 0 | 0% |
| 12 | Location | INTERNET | 4 | 2.06% |
| 13 | Location | INTERNET | 4 | 2.06% |

组合规则1是本文新提出来的权限组合规则，从检测结果来看，绝大多数的应用程序通过组合规则1泄露手机信息，泄露的信息包括“IMEI”、“IMSI”、“ICCID”、“Phone Number”。本文检测到有28款应用通过网络形式泄露“IMEI”、“IMSI”、“ICCID”、“Phone Number”，其中包括4款程序同时泄露“IMEI”和“ICCID”。组合规则3、组合规则5和组合规则6也是本文新提出来的组合规则，这些组合规则也同样检测到数据泄露。如表5-5所示。

**表5- 5 新增规则检测到的数据泄露**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **规则** | **隐私数据类型** | **泄露途径** | **数目** | **百分比（194）** |
| 1 | IMEI、IMSI、ICCID、Phone Number | INTERNET | 28 | 14.43% |
| 3 | SMS | INTERNET | 1 | 0.52% |
| 5 | Contacts | INTERNET | 2 | 1.03% |
| 6 | Accounts | INTERNET | 1 | 0.52% |

组合规则1检测到28款程序通过网络泄露“IMEI”（移动设备国际身份码）。本文对这28款应用程序的权限组合分布做了统计。结果如图5-2所示，横轴表示13组权限组合规则，纵轴表示在28款软件中每组权限组合使用频率。

从图中可以看出，虽然这28 款应用程序只是通过网络形式泄露手机信息。但是通过分析发现，在这些应用程序中，不仅存在组合规则1，还存在其他危险权限组合，其中组合规则5、组合规则12、和组合规则13所占的比例比较高。说明这些程序还存在泄露用户其他隐私数据的可能。

**图5- 2 通过规则1泄露数据的应用程序所申请的危险权限组合**

本文对检测到发生隐私泄露的36款应用程序，进行了整体的统计，每组权限组合的使用频率如图5-3所示，横轴表示13组权限组合规则，纵轴表示在36款软件中每组权限组合使用频率。

从图中可以看出组合规则1的使用率依然居高不下，在95%以上。另外还有组合规则5、组合规则12、和组合规则13这三组权限也都在60%左右。

**图5- 3 36款应用程序危险权限组合使用分布**

在194个应用程序中没有发现使用短信泄露隐私数据的应用程序。不过，我们通过实验例子验证短信也是应用程序泄露隐私数据的途径。

为了更清晰地对比隐私数据泄露和危险权限组合之间存在一定的关系，本文又分析了100个没有发生数据泄露的应用程序权限组合的分布，如图5-4所示，横轴表示13组权限组合规则，纵轴表示在100款正常软件中每组权限组合使用频率。

**图5- 4 100个正常软件危险权限组合使用分布**

通过分析发现，除了组合规则1 的使用率在70%左右之外，其他大部分在40%以下，其中组合规则6、组合规则12、和组合规则13这三组权限组合的使用率也大大下降了。

另外，通过数据研究发现，无论是发生数据泄露的应用程序还是没有发生数据泄露的应用程序。组合规3和5的使用频率比较少。出现这种情况的原因是，在互联网高速发展的现在，网络已经成为人们社交网络的第一主流工具，基本上很少有人使用短息的方式联系。

TaintDroid项目对Android市场的应用程序进行了长达两年的跟踪对比，表5-6列出了TaintDroid的实验样本集。表5-6中列出了从2010年到2016年的应用程序名称，“🗹”表示当时存在的（版本有升级）。

**表5- 6 TaintDroid样本集对比**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **应用程序** | **2010** | **2012** | **2016** |
| 1 | 3001 Wisdonm | 🗹 | 🗹 |  |
| 2 | Antivirus Freee | 🗹 | 🗹 | 🗹 |
| 3 | Astrid | 🗹 | 🗹 | 🗹 |
| 4 | Barcode Scanner | 🗹 | 🗹 | 🗹 |
| 5 | Blackjachk | 🗹 | 🗹 | 🗹 |
| 6 | Cestos Full | 🗹 | 🗹 | 🗹 |
| 7 | The coupons App | 🗹 | 🗹 |  |
| 8 | The directory for | 🗹 | 🗹 |  |
| 9 | Evernote | 🗹 | 🗹 | 🗹 |
| 10 | Hearts (Free) | 🗹 | 🗹 | 🗹 |
| 11 | Horoscope | 🗹 | 🗹 | 🗹 |
| 12 | iXmat Barcode Scanner | 🗹 | 🗹 | 🗹 |
| 13 | Myspace | 🗹 | 🗹 |  |
| 14 | Solitare Free | 🗹 | 🗹 | 🗹 |
| 15 | Traffic Jam Free | 🗹 | 🗹 | 🗹 |
| 16 | Trapster | 🗹 | 🗹 | 🗹 |
| 17 | The Weather | 🗹 | 🗹 | 🗹 |
| 18 | Yellow Pages | 🗹 | 🗹 |  |
| 19 | Bump | 🗹 | 🗹 | 🗹 |
| 20 | Layar | 🗹 | 🗹 | 🗹 |
| 21 | Babble Book | 🗹 |  |  |
| 22 | BBC News listen&tweet | 🗹 |  |  |
| 23 | Children’s ABC Animals | 🗹 |  |  |
| 24 | Knocking Live Video Beta | 🗹 |  |  |
| 25 | Mabilo Ringtones | 🗹 |  |  |
| 26 | Manga Browser | 🗹 |  |  |
| 27 | Movies and showtimes | 🗹 |  |  |
| 28 | Pro Basketball Scores | 🗹 |  |  |
| 29 | Slide:Spongebob | 🗹 |  |  |

本文依据权限组合规则，对2016年存在于Android市场的15款应用程序的权限分布做了统计，并和TaintDroid实验结果进行对比，如表5-7所示，表中列出了可能存在隐私数据泄露的应用程序（由于样本有升级，结果可能不准确），结果如表5-7所示。

**表5- 7 2012年与2016年样本泄露数据对比**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **应用程序** | **Location** | **Phone state** | **Contacts** | **Camera** | **Audio** |
| Antivirus Freee |  |  |  |  |  |
| Barcode Scanner |  |  |  |  |  |
| Blackjachk |  |  |  |  |  |
| Cestos Full |  |  |  |  |  |
| Evernote |  |  |  |  |  |
| Horoscope |  |  |  |  |  |
| Solitare Free |  |  |  |  |  |
| Traffic Jam Free |  |  |  |  |  |
| Bump |  |  |  |  |  |

注：2012 TaintDroid 样本，代表潜在威胁

2016 DroidProtector study 样本，代表潜在威胁

表5-7中是可能泄露用户隐私数据的应用程序，采用本文的检测方法和技术，对表中的九款应用程序进行安全检测，结果发现有两款应用程序泄露用户隐私数据，结果见表5-8。

**表5- 8 2016年泄露隐私数据的程序**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **应用程序** | **数据泄露类型** | **泄露途径** |
| 1 | Horoscope | IMEI | 212.234.169.241（法国） |
| 2 | Solitare Free | IMEI | 104.118.134.239（美国） |

* 1. 性能评估

本文使用Java性能评估工具CaffeineMark3.0对DroidProtector系统进行评估，CaffeineMark是用来评估不同平台上Java程序的执行速度[40]，本文对这三种不同系统进行了性能测试，结果如图5-5所示。

**图5- 5 三个系统性能测试的结果**

通过比较Android原系统、TaintDroid系统和DroidProtector系统，可以看出，DroidProtector性能损失的原因是由于在DroidProtector系统中增加权限组合判断逻辑模块。表5-9所示的是三个系统总体性能开销对比。

**表5- 9 三个系统性能对比**

|  |  |  |
| --- | --- | --- |
|  | **性能总分** | **相对性能开销** |
| **Android** | 1620 | 100% |
| **TaintDroid** | 1403 | 86.6% |
| **DroidProtector** | 1152 | 71.5% |

DroidProtector系统在启动系统应用程序和第三方应用程序的时间响应上和TaintDroid差别不太大，但是与Android原系统还是存在一定的时间差。DroidProtector系统加载系统应用比加载第三方应用程序性能较好。表5-10所示的是Android原系统、TaintDroid和DroidProtector加载系统应用和第三方应用程序的时间开销之比。

**表5- 10 三个系统时间开销比**

|  |  |  |
| --- | --- | --- |
|  | **加载第三方程序相对时间** | **加载系统程序相对时间** |
| **Android** | 1 | 1 |
| **TaintDroid** | 138% | 122% |
| **DroidProtector** | 145% | 126% |

本文提出的基于权限组合的Android程序隐私数据泄露检测方法和DroidProtector工具，结合了Android程序权限信息和API系统调用，相对于Kirin安全规则判定方法具有更高的准确性。但是，由于DroidProtector添加了权限组合判断逻辑和API调用的污点标记，在加载程序的时间上比Android原系统和TaintDroid系统时间开销大。

* 1. 方法的局限性

DroidProtector的设计是在TaintDroid开源项目的基础上完成的，由于TaintDroid具有一定的局限性，DroidProtector不可避免的在实验方案上会存在局限性，TaintDroid的局限体现在一下几点：

**（1）方法的局限性**

考虑到Android系统的性能开销，TaintDroid仅跟踪数据流（即显示流），而不跟踪控制流（隐私流），TaintDroid能够识别泄露危险信息的应用程序。但是，TaintDroid对真正的恶意程序并不能发挥作用，因为，跟踪控制流需要对第三方应用程序进行静态分析，由于第三方应用程序的源代码不可用，因而，基于控制流的静态分析难以实现。

**（2）实现方案的局限性**

Android使用了Java的Apahe Harmony实现方案，并进行一些自定义修改。该实现方案纳入了对PlatformAddress类的支持，PlatformAddress类包含一个原生地址，并由DirectBuffer对象使用。文件和网络 IO API包含了读和写的“直接”变体，这些变体使用了来自DirectBuffer的原生地址。TaintDroid目前并不跟踪DirectBuffer对象上的污点标记，因为该数据存储在不透明的原生数据结构中。目前，TaintDroid能在读或写的“直接”变体被使用时进行记录，单这种情形发生的频率极低。对于通过原生地址运行的sun.misc.Unsafe类，在实现方案上也存在类似的局限。

**（3）污点来源的局限性**

虽然TaintDroid能够有效地跟踪危险信息，但当被跟踪的信息包含配置标识符时，它会产生误报，例如，IMSI数值字符串由移动国家代码（MCC）、移动网络代码（MNC）和移动站识别码（MSIN）组成，所有这些代码都会一同收到污染。Android在传送其他数据时广泛使用MCC和MNC作为配置参数。如果IMSI被视为收到污染，会导致同一包中的所有信息都受到污染，最终造成受污染数据激增。因此，对于包含配置参数的污点来源，对包内的个别变量进行污染更为合适。然而，由TaintDroid的结果分析来看，消息级别的污点跟踪对绝大多数污点来源都是行之有效的。

* 1. 小结

采用209个应用程序对本文开发的安全检测工具进行实验评估，验证面向Android应用程序的检测方法和检测工具DroidProtector的有效性。实验结果表明，DroidProtector 检测工具能够有效预防Android程序隐私数据的泄露。

1. 工作总结与展望

近年来随着移动互联网的蓬勃发展，Android设备作为人们生活的必需品，为人们的工作和生活带来很多便捷。但是，用户对于手机信息的保护意识比较薄弱，没有意识到使用Android应用程序面临着的隐私数据泄露风险。正是由于用户对Android系统的数据保护意识弱，市场上才会出现窃取用户隐私数据的应用程序，并且数量也在不断增加。

本文研究Android权限访问机制，提出一种基于权限组合的Android应用程序隐私数据检测方法，并开发相应支持工具，本文的主要工作有以下几点：

1. 提出一组扩展的访问权限组合规则，扩展了Kirin安全规则。
2. 提出一种基于权限组合的Android程序安全检测方法。该方法自动解析程序权限，依据权限组合规则，添加隐私数据标记，跟踪数据在Android系统的信息流动。
3. 基于上述方法，开发了支持工具DroidProtector。该工具若检测到应用程序存在隐私数据泄露行为，发出系统通知，警示用户。
4. 通过Android程序验证本文所提方法和工具的有效性。对Android程序的实验表明，该工具可以有效检测隐私数据的泄露。

本文提出的基于权限组合的Android应用程序安全检测方法，目前还存在一定的不足，未来工作研究的主要方面有：

1. 当前的权限组合只是对与用户数据关联度比较高的隐私数据类型进行检测，由于Android系统上权限集可能发生变化，将来可能需要扩大危险权限组合的范围。
2. DroidProtector只检测应用程序隐私数据的泄露，并没有在数据泄露后对数据进行跟踪，也没有考虑应用程序是否有明确声明使用相关权限。因此，只要发生隐私数据泄露，本文就认为该应用程序存在数据泄露行为，并没有跟踪数据泄露后是用于合法用途还是非法用途。以后的工作将跟踪隐私数据泄露后的行为。
3. 由于DroidProtector是基于TaintDroid项目，因而不可避免地存在TaintDroid项目的局限性。隐私数据在传播时，不同粒度转换、赋值、传参、加密等操作都会存在隐私着色爆炸，这导致在数据检测时存在一定的误报率。TaintDroid在网络出口和短信出口处有脱色操作，这会导致漏报，研究发现，已经有针对TaintDroid项目脱色的研究，本文将进一步关注，及时改正。
4. 进一步研究针对隐私数据泄露的防御方案。

参考文献

1. Shabtai A, Elovici Y. Applying Behavioral Detection on Android-Based Devices[M]//Mobile Wireless Middleware, Operating Systems, and Applications. Springer Berlin Heidelberg, 2010: 235-249.
2. 网秦2014年 Q3全球手机安全报告[BE/OL], http://info.it.hc360.com/2014/11/14172679500.shtml.
3. Investigates C R. Keep Your Phone Safe: How to Protect Yourself from Wireless Threats[J]. Consumer Reports, 2013, 78: 6-18.
4. TrustGo. BSide Vegas: Your droid has no clothes [EB/OL]. [2014-01-08] . http://blog.trustlook.com/.
5. 张玉清, 王凯, 杨欢, 等. Android 安全综述[J]. 计算机研究与发展, 2015, 51(7): 1385-1396.
6. 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.
7. Lange M, Liebergeld S, Lackorzynski A, et al. L4Android: a generic operating system framework for secure smartphones[C]//Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. ACM, 2011: 39-50.
8. Peng H, Gates C, Sarma B, et al. Using probabilistic generative models for ranking risks of android apps[C]//Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012: 241-252.
9. Huan Y, Yuqing Z, Yupu H. Android malware detection method based on permission sequential pattern mining algorithm[J]. Journal on Communications, 2013, 34(1): 106-115.
10. Shabtai A, Fledel Y, Elovici Y. Securing Android-powered mobile devices using SELinux[J]. IEEE Security & Privacy, 2010, 3(8): 36-44.
11. Android Developer. Application components [EB/OL].[201312 03].https://developer.android.com/guide/components/index.html?hl=zh-cn.
12. Jiang Y Z X. Detecting passive content leaks and pollution in android applications[C]//Proceedings of the 20th Network and Distributed System Security Symposium (NDSS). 2013:1-16.
13. Miller B P, Fredriksen L, So B. An empirical study of the reliability of UNIX utilities[J]. Communications of the ACM, 1990, 33(12): 32-44.
14. Chan P P F, Hui L C K, Yiu S M. Droidchecker: analyzing android applications for capability leak[C]//Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks. ACM, 2012: 125-136.
15. Lu L, Li Z, Wu Z, et al. Chex: statically vetting android apps for component hijacking vulnerabilities[C]//Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012: 229-240.
16. Kim S H, Han D, Lee D H. Predictability of Android OpenSSL's pseudo random number generator[C]//Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013: 659-668.
17. Fahl S, Harbach M, Muders T, et al. Why Eve and Mallory love Android: An analysis of Android SSL (in) security[C]//Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012: 50-61.
18. Enck W, Ongtang M, McDaniel P. On Lightweight Mobile Phone Application Certification[C]//Proceedings of the 16th ACM conference on Computer and communications security. ACM, 2009: 235-245.
19. Enck W, Gilbert P, Han S, et al. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones[J]. ACM Transactions on Computer Systems (TOCS), 2014, 32(2): 5-33
20. Chin E, Felt A P, Greenwood K, et al. Analyzing Inter-Application Communication in Android[C]//Proceedings of the 9th international conference on Mobile systems, applications, and services. ACM, 2011: 239-252.
21. Chan P P F, Hui L C K, Yiu S M. Droidchecker: Analyzing Android Applications for Capability Leak[C]//Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks. ACM, 2012: 125-136.
22. Bugiel S, Davi L, Dmitrienko A, et al. Xmandroid: A New Android Evolution to Mitigate Privilege Escalation Attacks[J]. Technische Universität Darmstadt, Technical Report TR-2011-04, 2011:1-17
23. Somarriba O, Zurutuza U, Uribeetxeberria R, et al. Detection and Visualization of Android Malware Behavior[J]. Journal of Electrical and Computer Engineering, 2016, 17 (2016):1-17.
24. 李 挺 ,董 航,袁春阳,杜跃进,徐国爱. 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.
25. 魏松杰, 杨铃. 基于分层 API 调用的 Android 恶意代码静态描述方法[J]. 计算机科学, 2015, 1(179): 155-158.
26. Luo W, Xu S, Jiang X. Real-Time Detection and Prevention of Android SMS Permission Abuses[C]//Proceedings of the first international workshop on Security in embedded systems and smartphones. ACM, 2013: 11-18.
27. What is Android?[EB/OL].

http://api.apkbus.com/guide/basics/what-is-android.html.

1. Bergeron J, Debbabi M, Desharnais J, et al. Static Detection of Malicious Code in Executable Programs[J]. Int. J. of Req. Eng, 2001, 2001(184-189): 79-87.
2. Shabtai A, Elovici Y. Applying behavioral detection on android-based devices[C]// Proceedings of International Conference on Mobile Wireless Middleware, Operating Systems, and Applications. Springer Berlin Heidelberg, 2010: 235-249.
3. Application Fundamentals[BE/OL].

http://api.apkbus.com/guide/topics/fundamentals.html.

1. Google Inc． Android security[EB/OL].[2013-06-26].

https://source.android.com/security/index.html.

1. Enck W, Ongtang M, McDaniel P D. Understanding Android Security[J]. IEEE security & privacy, 2009, 7(1): 50-57.
2. 陈萍, 吴波, 张以皓等. Linux IPC 机制安全性的研究及改进[J]. 解放军理工大学学报: 自然科学版, 2004, 5(3): 17-19.
3. 吴倩, 赵晨啸, 郭莹, Android 安全解析机制与应用实战[M]. 机械工业出版社, 2013:19.
4. Lee S. Assessment of Malicious Applications Using Permissions and Enhanced User Interfaces on Android[C]// Proceedings of Intelligence and Security Informatics. IEEE, 2013: 270-270.
5. Su M Y, Chang W C. Permission-Based Malware Detection Mechanisms for Smart Phones[C]// Proceedings of the International Conference on Information Networking. IEEE, 2014: 449-452.
6. Developers [EB/OL].

file:///G:/android-sdk-windows/docs/reference/android/Manifest.permission.html.

1. Hamandi K, Elhajj I H, Chehab A, et al. Android SMS Botnet: A New Perspective[C]//Proceedings of the 10th ACM international symposium on Mobility management and wireless access. ACM, 2012: 125-130.
2. 安卓市场[EB/OL]. http://apk.hiapk.com/.
3. Kundu T K, Paul K. Improving Android performance and energy efficiency[C]// /Proceedings of the 24th Internatioal Conference on VLSI Design. IEEE, 2011: 256-261.

作者简历及在学研究成果

1. 作者入学前简历

|  |  |  |
| --- | --- | --- |
| 起止年月 | 学习或工作单位 | 备注 |
| 2010年9月至2014年6月 | 在河北经贸大学计算机科学与技术攻读 |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. 在学期间从事的科研工作

* 孙昌爱，李伟芳. 基于权限组合的Android应用程序安全检测工具（DroidProtector V1.0），计算机软件著作权，登记号：2016RS352225；登记日期：2016.12.04.

1. 在学期间所获的科研奖励
2. 在学期间发表的论文

独创性说明

本人郑重声明：所呈交的论文是我个人在导师指导下进行的研究工作及取得研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写的研究成果，也不包含为获得北京科技大学或其他教育机构的学位或证书所使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

签名：\_\_\_\_\_\_\_\_\_\_\_ 日期：\_\_\_\_\_\_\_\_\_\_\_\_

关于论文使用授权的说明

本人完全了解北京科技大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

**（保密的论文在解密后应遵循此规定）**

签名：\_\_\_\_\_\_\_\_\_\_\_ 导师签名：\_\_\_\_\_\_\_\_\_\_\_ 日期：\_\_\_\_\_\_\_\_\_\_\_\_

学位论文数据集

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **关键词\*** | **密级\*** | **中图分类号\*** | **UDC** | **论文资助** |
| Android应用程序，权限组合，程序分析，程序安全 | 公开 | TP311 |  |  |
| **学位授予单位名称\*** | | **学位授予单位代码\*** | **学位类别\***  **工学** | **学位级别\***  **硕士** |
| 北京科技大学 | | 10008 | 工学 | 硕士 |
| **论文题名\*** | | **并列题名** | | **论文语种\*** |
| 基于权限组合的Android应用程序安全检测方法与工具研究 | |  | | 中文 |
| **作者姓名\*** | 李伟芳 | | **学号\*** | S20140930 |
| **培养单位名称\*** | | **培养单位代码\*** | **培养单位地址** | **邮编** |
| 北京科技大学 | | 10008 | 北京市海淀区学院路30号 | 100083 |
| **学科专业\*** | | **研究方向\*** | **学制\*** | **学位授予年\*** |
| 软件工程 | | Android | 2.5年 | 2017 |
| **论文提交日期\*** | 2016年12月21日 | | | |
| **导师姓名\*** | 孙昌爱 | | **职称\*** | 教授 |
| **评阅人** | **答辩委员会主席\*** | | **答辩委员会成员** | |
| 王昭顺  刘吉强 | 王昭顺 | | 朱岩  刘宏岚 | |
| **电子版论文提交格式** 文本（pdf） 图像（ ） 视频（ ） 音频（ ） 多媒体（ ） 其他（ ） **推荐格式：**application/msword；application/pdf | | | | |
| **电子版论文出版（发布）者** | | **电子版论文出版（发布）地** | | **权限声明** |
|  | |  | |  |
| **论文总页数\*** | 57 | | | |
| 共33项，其中带\*为必填数据，为22项。 | | | | |