分类号：TP393　　密级：公开

UDC：　　学号：111487

东　南　大　学
工 程 硕 士 学 位 论 文

Android 软件恶意行为的智能分析与处理

研究生姓名：　张 扬

导师姓名：　罗军舟 教授

　　李国峰 高工

申请学位级别　工程硕士　　学位授予单位　东南大学

一级学科名称 计算机科学与工程　　论文答辩日期 **2013 年 8 月 30 日**

二级学科名称　计算机应用技术　　学位授予日期　　年　月　日

答辩委员会主席　曹玖新　　评　阅　人

2013 年 8 月 29 日

Intelligent Analysis and Disposal of Malicious Behavior in Android System

A Thesis Submitted to

Southeast University

For the Academic Degree of Master of Engineering

By

Yang Zhang

Supervised by

Professor Luo Junzhou

and

Senior Engineer Li Guofeng

School of Computer Science and Engineering

Southeast University, Nanjing, P. R. China

August 2013

东 南 大 学 学 位 论 文 独 创 性 声 明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。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过的研究成果，也不包含为获得东南大学或其它教育机构的学位或证书而使用过的材料。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作了明确的说明并表示了谢意。

研究生签名：_____日　期：_____

东 南 大 学 学 位 论 文 使 用 授 权 声 明

东南大学、中国科学技术信息研究所、国家图书馆有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权东南大学研究生院办理。

研究生签名：_____导师签名：_____日　期：_____

摘 要

近年来，随着智能终端的普及，Android 系统发展迅速。在 Android 操作系统被业界广泛接受的同时，各种恶意软件层出不穷。这不仅影响了用户的体验，更存在用户隐私数据被盗、资产受到侵害的危险。Android 系统的安全问题不但给随身携带智能终端的用户带来了困扰和忧虑，而且遏制了 Android 市场的进一步发展。因此，对 Android 系统恶意软件检测技术的研究已成为学术界与工业界关注的热点领域。

现有 Android 系统恶意软件检测技术主要分为静态特征码检测和动态行为分析两种方法。虽然静态特征码检测有较高的识别率，但该方法存在不可避免的滞后性，而且需要频繁联网更新病毒库，耗费用户的流量费用。而智能终端上的软件动态行为分析借鉴了传统 PC 机上的恶意软件识别技术，但是现有的研究很少考虑智能终端用户的行为习惯或者诸如权限特性之类的 Android 系统独有特性。

针对上述问题，本论文在综合考虑用户行为习惯和 Android 系统特性的基础上，研究 Android 软件恶意行为的智能分析与处理方案，实现一套 Android 安全防护软件。具体来说，主要工作包括以下几个方面：

1. 研究 Android 系统的安全机制，离线静态分析 2110 个 Android 软件的权限，实现 Apriori 算法挖掘出恶意软件和敏感权限组合的关联，并归纳出需要在软件运行时实时监控的敏感权限。

2. 研究 Android 系统源代码，设计并实现 Android 系统敏感权限的动态实时监控技术，从而能够动态地修改 Android 软件的权限，并实时地检测 Android 系统中软件运行时使用敏感权限的情况，拦截权限的恶意使用。

3. 动态分析 83 个软件在 Android 系统中的行为，结合静态分析与动态分析的结果，抽取用于识别 Android 系统中软件恶意行为的抽象属性特征，在改进朴素贝叶斯决策方法的基础上，研究并设计了 Android 软件恶意行为的智能识别算法。

4. 整合 Android 软件恶意行为的智能分析与处理方案，在扩展 Android 系统权限管理机制的基础上，设计并实现一款 Android 安全防护软件。

本论文在静态分析 Android 权限申请和动态分析 Android 权限使用的基础上，提出恶意行为的智能识别算法，并结合 Android 系统敏感权限的动态实时监控机制，最终实现一款完整、可靠的 Android 安全防护软件，可以有效地保障 Android 系统的安全。

关键词：Android 安全，权限管理机制，关联规则挖掘，动态实时拦截，朴素贝叶斯分类器

Abstract

In recent years, with the popularization of intelligent terminal, the development of Android system is quite rapid. As the Android operating system is widely accepted by the industry, all kinds of malicious software emerge in an endless stream. This not only affects the user experience, but also threats user privacy and property. Security issue of Android system not only brings troubles and worries to smart phone users, but also limits further development of Android market. Therefore, research on Android security has become a hot area in both academic and industrial field.

At present, existing Android malware detection technology is mainly divided into two methods, static scanning and dynamical analysis. Even though software based on static scanning can achieve high detection rate, it will frequectly updates antivirus software, which will bring extra cost. Research on dynamical analysis rarely takes user behavior and unique Android characteristics such as permission into consideration.

In view of the problems above, this thesis studies intelligent analysis and disposal of malicious behavior in Android system and implements Android defending system. Specifically, the main work includes the following:

1. Study the security mechanism of Android system, conduct static analysis of 2110 Android software authorities, use Apriori algorithm for mining associations between malware software and sensitive permissions, sum up sensitive permissions which need to be monitored dynamically.

2. Study Android system source code, design and realize real-time interception mechanism of Android API so as to monitor sensitive permissions in real time and intercept malicious use of sensitive permission.

3. Analyze the behavior of 83 software in the Android system, combine data from static analysis and dynamic analysis, extract abstract attributes for software malicious behavior recognition, and then based on improved naive Bias decision method, design Android software malicious behavior intelligent recognition algorithm.

4. Design and realize functional modules of the intelligent analysis and disposal scheme of malicious behavior, and then through extending the original permission management mechanism, realize Android defending software.

The research on malicious behavior of Android software includes static analysis of Android permission, dynamic interception of Android software's usage of permission and intelligent algorithm of Android software's malicious behavior recognition. Compared with

the existing research, intelligent analysis and disposal of Android software's malicious behavior proposed in this thesis is more reasonable, more efficient, and it is of great significance for improving permission system in Android system.

Keywords: Android Security，Permission Management Mechanism，Association Rules Mining，Dynamic and Real-time Interception，Navie Bayesian Classifier

目 录

图形目录

表格目录

第一章　引言

本章首先分别介绍了 Android 系统和 Android 系统安全问题的相关背景，对现有 Android 系统及其安全框架、Android 系统安全威胁以及 Android 恶意软件检测技术进行调研，指出存在的问题和安全隐患，最后提出本文的研究目标和内容。

1.1 研究背景与意义

Android系统的智能手机在当今世界呈现普及化趋势。根据Gartner的报告[1]，2013年第一季度Android系统市场份额已达到1.5亿台，占全球智能手机市场份额的74.4%，远高于排名第二的iOS系统所占的18.2%，并预测到2015年，Android系统的智能手机将达到5亿台。Android系统的快速发展，给移动服务带来了巨大的变革。Android系统智能手机的功能早已超越了传统手机的通信功能，各种新型应用层出不穷，Android应用商城也呈现了空前的火热场面。Android系统的智能手机已经蜕变成为融合通信、个人业务处理以及娱乐的强大个人终端。

然而，Android系统的普及也带来了巨大的安全挑战[2]。根据《2011年中国手机安全状况报告》中的统计，在2011年，Android木马呈现爆发式增长。与2010年全年共发现的12个木马样本相比，2011年全年，360手机云安全中心捕获新增Android木马样本数为4722个，被感染人数超过498万人次。例如Soundcomber[3]就是Android平台中典型的攻击案例，该木马可以从手机摄像头画面中抽取出隐私数据，并通过上下文感知技术，从语音信息中智能地提取出诸如银行卡卡号和密码等敏感信息。Android上的恶意软件通常偷偷收集智能终端上的隐私数据，然后通过短信、电话或者网络的方式将用户隐私数据传输到远端服务器，从而达到恶意统计、隐私出售、远端控制、恶意收费等目的。

造成Android系统中病毒爆发主要可以归结为以下三个原因：一是操作系统的开源特性。众所周知，Android操作系统是一个开源的操作系统。正因为如此，Android操作系统能够被广泛应用于各种嵌入式平台。然而，这也为黑客们开发病毒提供了便利。每一个电脑爱好者都可以去研究Android操作系统，发现其中的漏洞，并有针对性地开发出病毒软件。二是Android开发的低门槛性。Android系统提供的基于JAVA语言的程序开发套件为软件的开发提供了很大的便利。而另一方面，由于任何人只要交纳25美金后就可以上传自己的应用到Android

Market供别人下载，这为病毒的扩散提供了很大的便利。三是Android系统有缺陷的安全模型。Android软件在安装的时候需要申请相应的权限，这些权限的申请会在软件安装时以列表的形式提供给用户。程序所需权限列表在安装时必须被用户全部赋予，软件安装完成后不可更改，这种粗粒度的特性导致Android系统成百个权限的授权责任被转嫁给没有安全意识的用户，一旦程序被授予了权限，则该权限不会再被改变，并且在该权限使用的时候，Android系统也不会通知用户。这样的框架设计给系统留下了极大的安全隐患。由于病毒必须通过系统敏感权限调用来实现对敏感资源的访问，因此，对系统敏感权限调用进行有效控制是遏制病毒爆发的有效途径。

市面上现有的 Android 安全防护软件，例如 LBE、360 手机卫士等，都对软件的恶意行为进行一定程度的监控，当发现系统 API 调用对敏感资源进行访问的时候，会提示用户，让用户来判断是否允许这些软件进行这一系列的操作。这种方式的确细化了 Android 系统的安全体系，但是授权软件进行敏感操作的责任仍旧被转嫁到没有安全意识的用户头上。而且，当用户正常操作的时候，安全防护软件经常会弹出对话框来提示用户当前有软件在调用某些敏感资源，这时常会打断用户的正常操作。因此，如何智能化地对系统敏感资源的访问进行分析和处理，在兼顾系统安全需求的同时，保障用户体验，是开发 Android 安全防护软件所需要解决的关键问题。

本文在 Android 软件基础框架层代码上对 Android 系统敏感权限调用进行有效地控制，并对软件的行为进行智能分析，从而识别出软件的恶意行为，并最终实现一款 Android 安全防护软件。该软件能够在用户安装时，提供智能安装功能，不但给予用户更为直观的安全提示，而且能够限制软件实施恶意行为的能力；在软件运行时，可以自动地判断软件在调用系统敏感权限时是否属于正常的行为，从而避免了由没有专业背景的用户来判断当前软件的行为是否合理，智能化地保障了 Andriod 系统的安全。

1.2 研究现状

根据 Android 系统日益增长的安全需求，本文对相关研究领域进行了深入的调研和总结，包括：（1）Android 系统及其安全架构；（2）Android 系统面临的安全威胁；（3）Android 系统恶意软件检测技术。

1.2.1　Android 系统及其体系结构

众所周知，Android 系统是由 Google 的 Open Handset Alliance(OHA)团队开发的移动开源平台。正是由于其开源特性，现有市场上对 Android 操作系统的修改版本层出不穷，各式各样的修改版本充斥着 Android 智能终端市场。为了更好的了解 Android 操作系统，首先得对各个版本的 Android 操作系统有所了解。目前，市场上常见的 Android 操作系统项目包括 Google 原生的 AOSP、Cyanogen 团队开发的 CyanogenMod、Roman Birg 所领导开发的 AOKP，以及国内市场上常见的 MIUI、OPDA、腾讯 tita、百度云系统等。

（1）AOSP

AOSP 是“Android Open-Source Project”的缩写。Google 公司每发布一个新版本的 Android 系统，都会将其源代码版本发送至开源社区，所发放出来的操作系统源代码即是 AOSP，这也是最为纯净的 Android 系统。其优点是由 Google 公司官方进行维护与更新，代码的可靠性、安全性与可读性都有着较高水平的保障。

（2）Cyanogen

Cyanogen 是全球最大的第三方操作系统编译团队，该团队所开发覆盖的机型是最广的，几乎所有的热门机型都有相对应的 CyanogenMod(CM)系统可以供用户刷机。基于 AOSP，Cyanogen 对 Android 系统的性能和可靠性方面进行了深度的优化。不但如此，Cyanogen 还提供了一系列原生 Android 操作系统所没有的特性功能，例如，CM 系统支持 FLAC 音频格式、程序可从 SD 外置存储器运行、高速缓存压缩[4]、大量的接入点名单、Wi-Fi 无线网络支持、蓝牙及 USB 网络分享等。

（3）MIUI

MIUI 是小米科技运营的 Android 操作系统项目，它是基于 CM 系统进行二次修改的衍生项目。小米团队每周在其论坛上发布新的版本，并通过空中下载[5]的方式对每一部小米手机进行系统升级。相对于 CM 系统来说，MIUI 的开发重心放在了用户体验与细节功能的改进，自主原创了全套的用户体验设计体系，其首创的各种炫彩功能为 Android 智能终端用户带来了华丽的界面体验。MIUI 也特别侧重于系统空间的纯净，从不集成第三方软件，而且其独特的双系统共存模式更方便用户刷新系统。

（4）OPDA

OPDA 团队成立于 2007 年，是国内最早的 Android 开发团队。旗下更汇集了众多优秀的编程、汉化、开发、破解、资源、制作打包技术团队和资深专家。

OPDA 系统在保证 Android 系统的完善性和安全性的前提下，对其进行了更深层次的优化精简，在国内外都享有一定的声誉。OPDA 系统相较于 CM，更贴近于国人的操作习惯；相较于 MIUI，则加入了定制化的贴心服务；相较于其它众多 Android 系统，又精简了许多不必要的软件程序。

通过以上的调研发现，很多 Android 开源平台在设计理念、系统架构、规模、兼容性、安全性以及性能等方面都有着相应的考虑。同时，Android 系统的开源特性为嵌入式智能终端系统的研究带来了更加开放、灵活与自由的构建方法。

当前 Android 操作系统的典型的体系结构如图 1-1 所示。

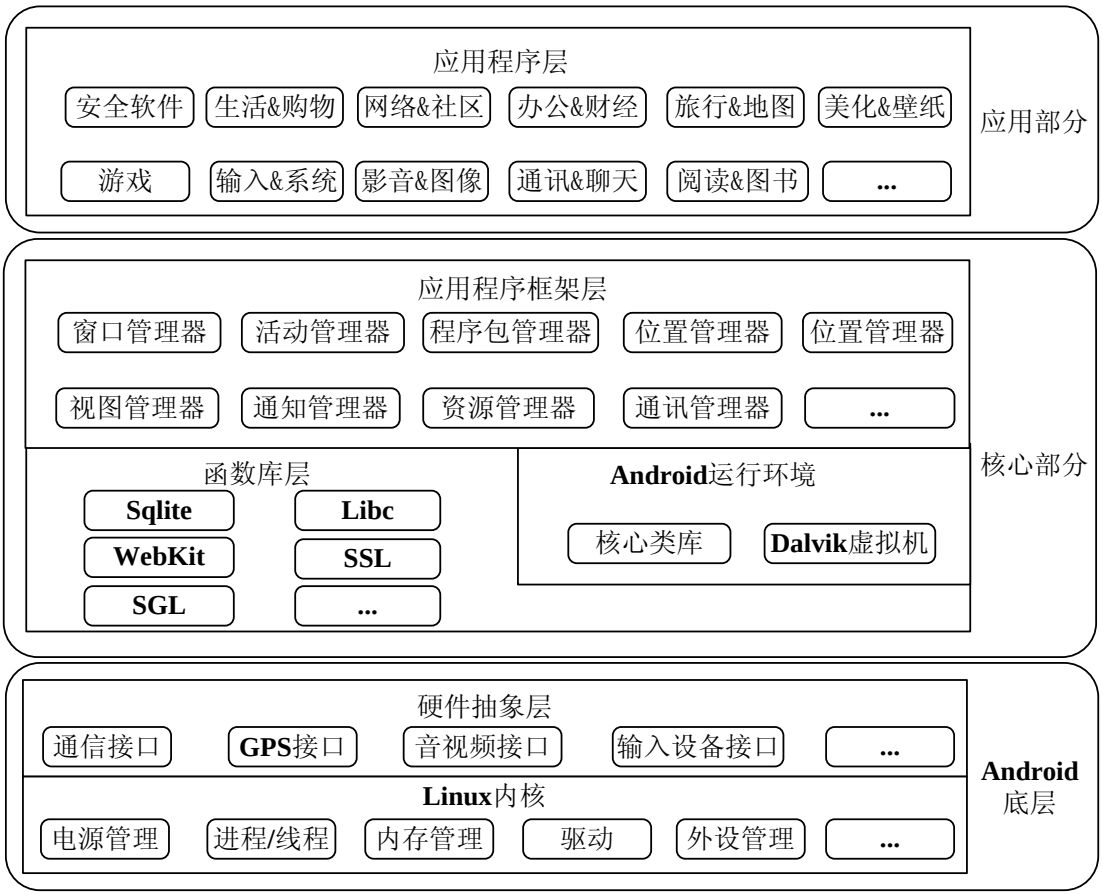

图 1-1 Android 系统安全框架

Android 操作系统的体系结构共分为三个层次：

1）第一层是 Android 底层，由 Linux 内核与硬件抽象层组成。Android 的核心系统服务基于 Linux2.6 内核，但对驱动层和硬件抽象层进行了一些修改，以适应智能手机的嵌入式操作环境。这层的主要功能是对硬件进行管理和抽象。

2）第二层是 Android 核心部分，由 Android 核心库、运行环境和应用程序框架层组成。核心库中包含 C/C++库，其中大部分都是开源的函数库。运行环境由 Java 的核心库和 Dalvik 虚拟机[6]构成。该层为每一个软件提供单独的执行

环境，每一个 Android 软件都在 Dalvik 虚拟机的一个实例中执行，所以不同的软件之间不能互相干扰。应用程序框架层提供了丰富的程序开发接口供开发者使用，负责应用程序层和底层之间的交互。

3）第三层是应用程序层。这层提供直接与用户交互的软件，包括系统软件和第三方软件。

1.2.2　Android 系统面临的安全威胁

随着 Android 操作系统被业界广泛接受并普及的同时，各种恶意软件不断出现。这不仅影响了用户的体验，更存在用户隐私数据被盗、资产受到侵害的危险。

由于移动设备的特点，对于移动设备的攻击与传统针对个人电脑的攻击存在相当大的差异。大部分攻击者的目的都是为了在攻击中受益，所以在传统的攻击中，攻击者一般是通过攻击操作系统的漏洞达到控制受害主机的目的，除了特定的情况外，攻击者很少对电脑中的数据感兴趣。而且控制受害机本身并不能够直接获益，需要通过 DDoS 攻击等间接的方式才能从攻击中受益。但是，对于移动设备，特别是智能手机的攻击则完全不同。智能手机中保存着大量的个人信息，这些信息本身就蕴含着丰富的商业价值。所以，对于智能手机的攻击，目的性更明确，大部分的攻击者都是为了窃取手机中的个人信息。此外，随着移动支付技术的成熟，恶意软件还会威胁到了用户支付账户的安全。

Android 移动终端的安全问题不但给随身携带智能终端的用户带来了困扰和忧虑，而且阻碍了 Android 市场的进一步发展。由于第三方移动应用越来越多地和用户的隐私、商业信息无缝结合，针对移动平台的隐私泄露和隐私攻击成为 Android 系统研究中亟待解决的问题。在目前有关 Android 智能终端安全问题的研究中，用户隐私问题已经引起学术界的广泛关注，研究主要分为两类：

第一类是研究木马程序的恶意行为[7]，通过木马程序，攻击者可以直接执行包括窃取用户隐私信息在内的恶意行为。木马程序特指在实现合法功能的掩护下，偷偷地执行恶意行为的程序。智能终端上的木马程序通常伪装成合法的应用软件，骗取用户下载并安装，然后作为智能终端后台常驻程序，执行窃取用户隐私信息或者破坏系统文件等恶意行为。有些恶意程序会注册一些系统事件响应，例如在收到特定来源的信息时偷偷拨打收费电话或者发送特定的短信。还有些程序通过定时获取的方式，在用户不知情的情况下侵犯用户隐私，例如收集手机的 GPS 信息、手机中存储的 WIFI 信息等等，表 1-1 中列出网秦公司[8]最新发现的恶意程序。这些恶意软件就如同正常软件一样，在安装的时候需要申请相应的权限，这些权限的申请会在软件安装时以列表的形式提供给用户，

程序所需权限列表在安装时必须被用户全部赋予，软件安装完成后不可更改。Android 系统这种粗粒度的权限框架不但给系统带来了极大的安全隐患，而且这种权限框架缺乏应对权限提升攻击[9]的有效防御。权限提升攻击是通过程序间的漏洞，非授权的用户可以获取到隐私数据和设备权限，例如读取联系人的权限等。

表 1-1 网秦公司最新发布的恶意软件

类型	恶意软件别名	恶意行为
隐私窃取	privacy.CallSpy	1.上传短信、通话录音等隐私信息，造成用户隐私泄露。 2.上传隐私过程消耗用户大量流量，造成用户资费消耗。
恶意扣费	remote.UpdtBot	1.后台安装软件，消耗用户流量并对用户手机造成破坏。 2.后台发送短信和拨打电话，对用户的资费造成消耗。
诱骗欺诈	fraud.Fakesms	1.后台联网，根据服务器指令，推送广告，消耗用户流量。 2.后台联网，根据服务器指令，伪装短信推送广告。
流氓行为	payment.TSBlocker	1.根据远程指令后台发送短信，造成用户资费损失。 2.屏蔽特定短信，造成系统破坏。 3.试图卸载用户安全软件，使用户安全受到威胁。
远程控制	remote.BgEngine	1.后台联网，与服务器交互，根据服务器指令向外发送短信，消耗用户资费，造成用户经济损失。
系统破坏	system.KillAV	1.屏蔽运营商短信，使用户收不到运营商发来的短信。 2.破坏安全软件运行，给手机带来安全隐患。 3.后台联网接收服务器指令，对手机进行远程控制。

第二类是研究用户隐私信息的侧信道攻击。攻击者可以通过收集一些非敏感信息来推断出敏感信息。例如 Owusu 等人[10]开发了一套利用加速度传感器信息就能推断出虚拟按键的恶意软件，但是推断结果的准确性受加速度传感器的频率、键的位置，以及键的大小的影响。Owusu 在预处理过的加速度数据流中提取了 46 个特征属性，结合模式识别的知识，用来推断虚拟按键。类似的功能也在一款名叫 ToughLogger[11]的恶意软件中被实现。Marquardt 等人[12]也利用加速度传感器来检测手机的震动数据，从而推断出手机附近的 PC 键盘上正在输入的文本信息。然而手机的加速度传感器的采样频率只有 100Hz 左右，这势必导致推断结果不准确。Marquardt 使用键值对，结合神经网络的方法在原始的加速度数据中还原出文本信息。击键的推断一直在被研究，也能够通过各种各样的属性来进行推断，例如通过击键的声音频率[13]，两个按键间的时间间隔[14]，以及按键的声音[15]来推断用户的键盘输入。Michal Zalewski[16]等人利用屏幕上的手指热残留来推断用户之前所击键的位置。然而当使用普通的热能成像相机

来推断用户的输入的话，攻击者必须在手指热残留保持的 5 到 10 分钟之内拍摄热能图像。Mowery Zalewski 从包括触摸屏材质，身体热能分布以及攻击可拓展性这三个方面来对这种攻击的有效性与合理性进行考量。密码顺序则可以从按键的热残留程度来推断。Zhang 等人[17]和 Aviv 等人[18]利用对触摸屏上的指纹残留来推断用户在智能终端上的按键输入。两者不同的是，Zhang 推断的是用户的文本密码，而 Aviv 推断的是 Android 图案模式的密码。

从恶意软件的传播特性来说，执行恶意行为的木马程序相比较于侧信道攻击，具有大规模爆发的能力与更稳定的商业利益。通过侧信道手段获取用户隐私信息的方法，仍旧处于研究阶段，其可靠性也限制了该技术的大规模应用。因此，如何有效地遏制 Android 软件的恶意行为，是完善 Android 系统安全的主要问题。

1.2.3　Android 系统恶意软件检测技术

现有Android系统恶意软件检测技术主要分为静态特征码检测和动态行为分析两种方法。

静态特征码检测是检测恶意软件最常用、最直接的方法。静态特征码检测根据分析对象的不同，主要分为二进制程序扫描[19]和源代码扫描[20]。二进制程序扫描是通过查找软件编译生成的二进制文件中是否包含恶意行为特征码，从而判断程序是否为恶意软件。源代码扫描，是指通过反编译工具，人工地提取出病毒程序的特征码。以恶意扣费为例，引起扣费行为的发生往往是由于恶意应用以发送短消息的方式，注册了相关的服务。分析Android应用反编译后的源代码，通过查询源代码中是否包含发送短消息相关的API调用，并判断相应的API调用参数，是否为某些特定服务注册码，从而检测Android应用中是否包含恶意扣费等代码。

虽然静态特征码检测在识别已知的恶意软件时是高效准确的，但在检测未知的恶意软件或已知恶意软件的变种时，则显得无能为力。因为静态特征码检测技术依赖病毒库，对于病毒库中不存在的病毒，便无法查杀。因此，面对每天都会产生的各种新型恶意软件及其变种，静态特征码检测存在不可避免的滞后性。为了查杀新型恶意软件，则必须频繁更新病毒库，在智能终端平台上，这可能会给用户造成额外的流量费用。另外，随着近些年Android系统中的恶意软件数量指数型增长，病毒库也变得越发庞大，静态特征码检测的效率将变得难以保证。受制于智能终端上的物理资源和电池续航能力，静态特征码检测将会给用户的正常使用带来较大的影响。

为了简化静态特征码检测的复杂度，Enck等人[21]提出Kirin系统，通过分析Android软件申请的权限来判定软件是否存在恶意行为。Barrera等人[22]采用SOM算法研究Android权限机制，他们通过对1100个Android软件的权限进行分析，证实了不同类别软件在申请权限时候存在很大的差异。由此，Android权限机制也是可以作为检测Android恶意软件的特征属性。但现有的基于Android权限的恶意软件静态检测技术，仅仅凭借经验提取了一些恶意权限的组合，其完整性与可靠性仍旧是值得质疑的。

动态行为分析是对软件运行时所表现出的行为进行抽取与建模，然后通过模式识别算法判断这一系列的行为是否是恶意行为的技术。该方法不依赖于特征库，能够识别出未知的恶意软件。通常，软件动态分析分为模型创建和恶意判定两个步骤完成，如图1-2所示。

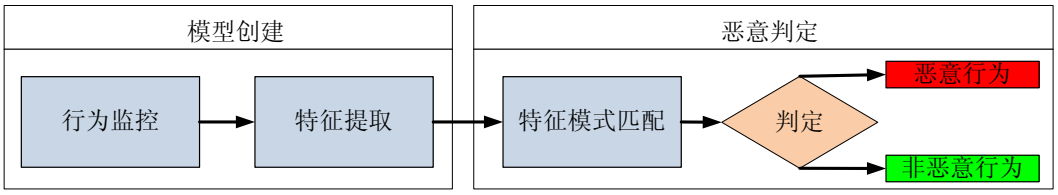

图 1-2 软件动态分析的步骤

模型创建目的是对软件的行为加以抽象描述，从而为恶意判定工作做准备。行为分析分为两个步骤，一是对智能手机上的软件行为进行捕获，通常是对智能手机上的软件运行时的调用进行监控，包括监控软件的系统调用、软件的系统资源消耗等数据。二是对捕获到的数据进行建模，即将这些信息通过形式化的方法表示出来。用特定的模型表示程序行为，是为了在后续的恶意判定过程中，可以依据模型间的比较，做出恶意软件的判定，或是进行变种检测。

恶意判定是通过相关的算法对行为分析抽象出来的模型加以分析，判定这一系列行为是否是恶意行为的过程。在恶意判定过程中，通常的做法是进行特征模型的匹配。比如，对已有的恶意软件集合建立特征库，在对未知的软件行为进行行为分析后，将分析到的特征模型和特征库中的特征模型一一匹配，如果匹配成功，则判定是恶意行为。其中判定用到的各种算法，从最初的二进制序列特征或者正则表达式匹配[23]到现在的模式识别方法和机器学习方法[24]。

动态行为分析对特征属性的抽取与建模有很强的依赖性，所以该方法虽然能够分析出未知的恶意软件，但其分析结果往往存在误报。因此，动态行为分析的研究主要集中在行为监控、特征提取与特征模式匹配算法，从而提高正确率、降低误判率、提高算法执行效率等。

对行为监控的研究，Enck等人[25]设计实现了TaintDroid系统用以标记内存数据的流向，TaintDroid通过修改Dalivk虚拟机，将传统的信息流追踪技术应用于

Android系统中，在软件运行时对系统中敏感信息的传播进行监控，从而达到安全防范的目的。然而TaintDroid因采用传统的动态污点传播技术，使得运行时的程序性能开销明显提升，这对于实时性要求非常高的移动平台而言，过高的性能开销将大大降低其实用性。刘泽衡等人[26]主要利用Android系统的广播机制来对短信接收、电话接听与发送以及网络连接等敏感操作进行监控。这种方法对于Android系统内的不同资源需要设计不同的方法进行监控，而且不能捕获到用户隐私信息的泄漏。

对特征提取的研究，Miettine等人[27]提出了智能手机异常检测中需要监控的一些系统信息，包括操作系统事件响应、操作系统资源使用率等一系列量化数据，并在此基础上提出了一个入侵检测理论模型。Schmidt等人[28]从Linux内核角度分析了Android系统的安全性，并利用网络流量、系统调用以及文件系统日志来检测系统异常。乜聚虎等人[29]提出了基于智能手机行为分析的异常检测方法，并在Android系统上实现了这套系统。可是其方法并不能对异常进行精确定位，即无法找出引起异常的恶意软件，并且在发现系统异常以后，系统不能实时地阻止软件的恶意行为发生。

对特征模式匹配算法的研究，Rieck等人[30]提出使用支持向量机对恶意软件行为进行有监督的学习。首先对大量恶意软件样本行为进行映射，然后用支持向量机对行为进行划分。该算法需要有大量的恶意软件样本做依托，而且对智能终端性能要求较高。Bose等人[31]在智能终端上提出一个基于聚类的智能算法，通过分析短信或彩信的日志文件，学习软件使用短信或彩信的行为，从而检测恶意软件。类似的，Xie等人[32]提出pBMDS系统，通过拦截智能终端上用户的输入，从而建立HMM模型来检测智能终端上的病毒。这两种方法对智能终端病毒的检测仅局限于短信和彩信的恶意发送，并没有对其他诸如打电话、网络或者敏感权限的恶意调用进行检测与拦截。Shabtai等人[33]在Android系统中使用了标准的机器学习算法，包括K均值、逻辑回归、决策树、贝叶斯网络和朴素贝叶斯，分别对恶意软件进行识别。从实验结果看，决策树和朴素贝叶斯算法均能达到很高的准确率。但是，Shabtai的数据集中仅有24个Android软件，其中4个恶意软件是由他自己编写。因此，Shabtai对算法的测试是不够全面的。

1.2.4 研究现状总结

Android系统在近些年得到了迅猛发展。伴随着Android系统的普及，Android系统中的恶意软件呈现火爆地增长。因此，Android系统恶意软件检测技术的研究已经成为了学术界与工业界关注的热点领域。

现有Android系统恶意软件检测技术主要分为静态特征码检测和动态行为分析两种方法。静态特征码检测是检测恶意软件最常用、最直接的方法，而且静态特征码检测对已知的病毒软件能达到较高的识别率，但是该方法存在不可避免的滞后性，而且该方法需要频繁联网更新病毒库，耗费用户的流量费用。随着Android中恶意软件指数级地增长，该方法的检测效率也难以保障。智能终端上的软件动态行为分析借鉴了许多传统PC机上的恶意软件识别技术，能够对未知的恶意软件进行识别。然而现有Android软件行为动态分析的研究侧重于系统运行时的状态，对智能终端用户的行为习惯或者诸如权限特性之类的Android系统独有特性很少考虑，而且对行为监控的方法很少提及。现有研究大多只是在模拟平台上运行，很少能够实现一套完整的Android安全防护软件。

因此，在 Android 系统中，如何结合考虑用户的行为习惯与系统独有特性，从而智能化地对系统敏感资源的访问进行分析和处理，在兼顾系统安全需求的同时，保障用户体验，并设计实现一套智能的 Android 系统安全软件，是当下对 Android 安全研究需要解决的关键问题。

1.3 研究目标和内容

1.3.1 研究目标

本文针对 Android 系统日益增长的安全需求，研究 Android 软件恶意行为智能分析与处理方案。在结合 Android 权限静态分析与敏感权限实时监控的基础上，实现 Android 软件恶意行为识别技术，并最终实现一款智能的 Android 安全防护软件，为 Android 系统用户提供安全的使用环境。

1.3.2 研究内容

为实现上述研究目标，具体研究内容包括以下四点：

（1）Android 系统中软件权限的静态分析

研究 Android 系统的安全机制及其核心的权限管理机制。在此基础上，静态分析 Android 软件在安装时所申请的权限，实现 Apriori 算法挖掘出恶意软件和敏感权限组合的关联，并归纳出需要在软件运行时实时监控的敏感权限。

（2）Android 系统敏感权限的动态实时监控

通过 Google 公布的 Android 系统源码，分析 Android 应用程序框架层的实现流程，提出一套适用于 Android 系统的、对敏感权限进行动态实时监控的机制。该监控机制可以实现三个功能，一是动态修改 Android 软件的权限、二是

检测 Android 软件使用敏感权限的情况、三是动态地对当前系统敏感权限的使用进行阻断或者放行。

（3）Android 软件恶意行为识别技术的研究

通过捕获软件使用敏感权限时的相关状态，提出合适的模型。在此基础上，选用合适的模型匹配算法，对软件的行为进行分析，判断软件当前行为是否合理，并利用软件权限静态分析与实时监控的研究成果，智能化地协助用户做出选择，即在当前时刻是否授权软件使用系统敏感权限。

（4）Android 安全防护软件的设计与实现

基于上述技术与理论成果，在确保实用性和安全性的前提下，设计实现 Android 安全防护软件。综合应用软件权限的静态分析结果、软件行为的实时监控技术和对软件恶意行为识别的研究，保证 Android 系统的安全。

1.4 论文组织结构

论文章节内容安排如下：

第一章概述现有 Android 系统中的安全问题以及相关研究，对现有 Android 系统及其安全框架、Android 系统安全威胁以及 Android 恶意软件检测技术进行调研，指出存在的问题和安全隐患。

第二章分析增强 Android 系统安全的具体需求，提出增强 Android 系统安全的方案框架，设计 Android 安全防护软件，并介绍软件的体系结构和功能模块。

第三章分析 Android 系统的安全机制，并对其中最为核心的权限管理机制进行研究。通过对收集到的 1260 个恶意软件和 Android 市场中 17 个分类的 850 个热门软件所申请的权限进行静态分析，实现 Apriori 算法挖掘出恶意软件和敏感权限组合的关联，然后归纳出需要在软件运行时被监控的 10 种敏感权限。

第四章分析了 Android 系统源代码中短信发送和权限审查的具体实现，归纳出 Android 系统操作的流程。设计并实现了 Android 系统敏感权限的动态实时监控机制，从而捕获静态分析中归纳出的 10 个敏感权限在 Dalvik 虚拟机中的使用情况。

第五章通过对静态分析和实时监控到的数据进行分析，抽取出 5 个用于识别 Android 系统中软件恶意行为的抽象属性特征。基于采集到的 8849 条属性特征数据，构建朴素贝叶斯模型，并根据实际场景，在改进朴素贝叶斯决策算法的基础上，设计了基于朴素贝叶斯分类器的 Android 软件恶意行为智能识别的算法，并验证正确度和误报率。

第六章阐述 Android 安全防护软件各功能模块的具体实现，并给出软件的功能测试和性能测试。

第七章总结论文完成的工作，对本文的理论和实践成果进行总结，并对未来工作进行了展望。

第二章 Android 安全防护软件的总体设计

本章在分析增强 Android 系统安全的具体需求基础上，提出增强 Android 系统安全的方案框架，并阐述 Android 安全防护软件的具体设计。

2.1 需求分析

为了保证本章所设计的 Android 安全防护软件的合理性，对实际应用的需求和同类型 Android 安全防护软件的功能进行调研和分析，确定了本系统的功能需求和非功能需求。

功能需求主要有以下几项：

(1) 获取准备安装的 Android 软件所申请的权限；

(2) 分析并拦截 Android 软件安装时对权限的申请；

(3) 智能安装 Android 软件，提示软件中存在的恶意权限组合；

(4) 安装时一键设置软件中的恶意权限组合的使用为拦截状态；

(5) 直观显示各个软件所拥有的权限；

(6) 用户能够对各个 Android 软件拥有的权限进行设置；

(7) 根据用户的设置，拦截 Android 软件运行时对敏感权限的使用；

(8) 获取 Android 软件运行时候的相关状态，包括当前时刻是否有用户在操作，敏感权限的使用是否存在突发性；

(9) 实时智能分析 Android 软件运行时所使用的敏感权限是否合理；

非功能性需求有以下几项：

(1) Android 安全防护软件界面友好、软件操作流程清晰流畅；

(2) 能快速处理用户的请求，平均相应时间小于等于 5 秒；

(3) 作为安全防护软件，要求软件无故障率达 99.9%。

(4) 能够在常用分辨率的显示屏上正常操作。

2.2 系统方案框架

为了有效地保护 Android 系统安全，本系统主要防范的目标是用户准备安装的 Android 软件和正在 Android 系统中运行的软件。因此，本系统的方案框架如图 2-1 所示。

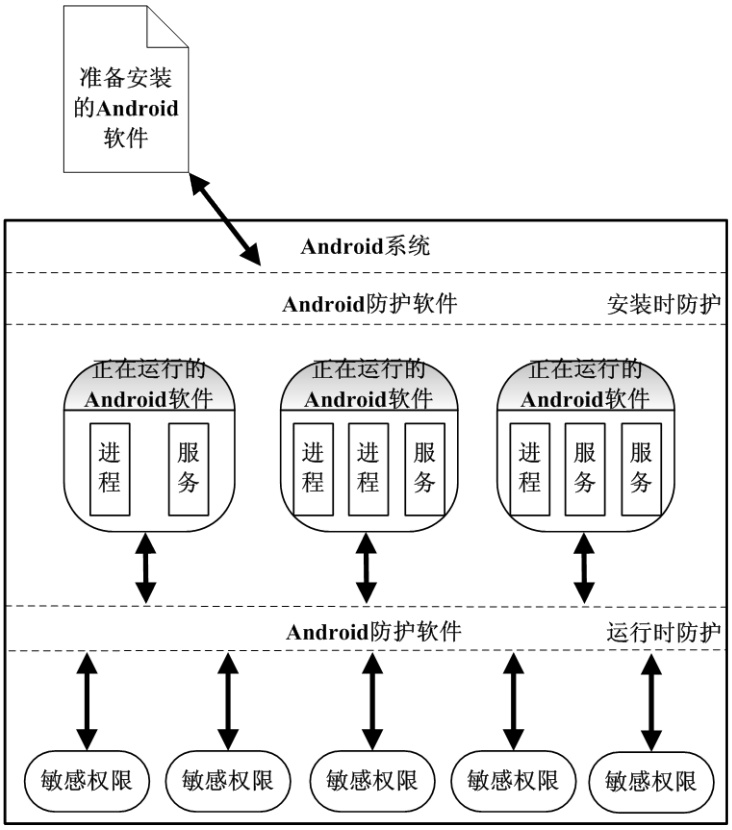

图 2- 1 Android 防护方案设计

从图 2-1 中可以看出，本系统的防护分为两个阶段：第一个阶段防护是在 Android 软件安装的阶段，当 Android 用户准备安装软件的时候，Android 防护软件将会抽取所安装软件申请的权限,然后根据静态分析得出的恶意权限组合，匹配该软件所申请的权限，给予用户安全提示，告诉用户该软件中是否存在恶意权限组合，让用户选择是否继续安装。此时用户可以选择智能安装方式，使得用户在安装了拥有恶意权限组合的软件后，软件会自动设置这些恶意权限组合的使用为拦截状态。第二部分防护是在 Android 软件运行的阶段，当正在运行的 Android 软件使用 Android 系统敏感权限的时候，Android 防护软件将根据用户的设置选择不同的处理方式，包括放行、拦截、提示用户和智能分析。当预先的设置为智能分析的时候，Android 防护软件则会结合软件本身的基础属性以及系统运行环境，对当前使用敏感权限的行为作出识别，最后拦截该软件恶意使用所拥有的敏感权限。

2.3 系统体系结构

系统总体设计框架如图 2-2 所示，主要由三个层次组成：（1）基础数据层；（2）系统功能层；（3）人机交互层。

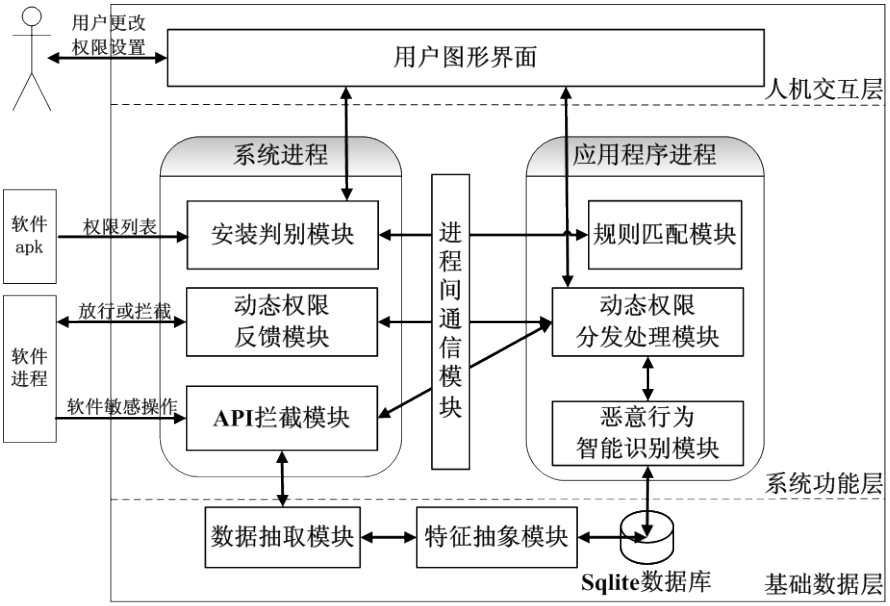

图 2- 2 系统整体设计

2.3.1 基础数据层的设计

Android 安全防护软件基础数据层的主要功能是记录 Android 软件运行时对各敏感权限的使用情况和 Android 系统中各软件的基础属性，然后将这些数据组成软件行为特征序列，为系统功能层的智能判别做好准备。

其核心模块包括：数据抽取模块和特征抽象模块。具体如下：

（1）数据抽取模块

数据抽取模块负责记录 Android 软件运行时对各敏感权限的使用情况和各软件的基础属性，主要包括当前系统时间、软件的属性、软件所申请的权限信息、设备输入硬件当前的使用情况等。

（2）特征抽象模块

特征抽象模块负责将数据抽取模块采集到的数据抽象为行为特征序列，其中包括敏感权限的使用者是否是系统应用、敏感权限使用时是否有用户操作、敏感权限的使用者是否在安装时申请了大量权限、敏感权限的使用者是否在安装时存在恶意权限组合、敏感权限的使用是否存在突发性。

2.3.2 系统功能层的设计

系统功能层的主要功能分为两部分：第一部分是解析准备安装的 Android 软件权限信息，通过匹配敏感权限组合规则集，给予用户安全提示，并且提供用户智能安装功能；第二部分是能够实时监控 Android 系统中软件使用敏感权限的情况，并智能分析权限使用的合理性，拦截权限的恶意使用。

其核心模块包括：安装判别模块、规则匹配模块、API 拦截模块、动态权限分发处理模块、恶意行为智能识别模块、动态权限决策反馈模块和进程间通信模块。具体如下：

（1）安装判别模块

安装判别模块负责解析准备安装的 Android 软件 apk 文件中的权限信息，并且将这部分信息交付给规则匹配模块，然后根据规则匹配模块的匹配结果，在用户安装软件的时候，给出相应的安全提示，告诉用户准备安装的软件中存在的恶意权限组合。

（2）规则匹配模块

规则匹配模块负责匹配安装判别模块发来的权限列表与敏感权限组合规则集，返回准备安装的软件中可能实施恶意行为的敏感权限组合。

（3）API 拦截模块

API 拦截模块负责实时监控 Android 系统中软件的敏感操作并发送给动态权限分发模块，然后提示数据抽取模块采集当前时刻的系统相关信息。

（4）动态权限分发处理模块

动态权限分发处理模块接收 API 拦截模块所发来的敏感权限使用信息，然后分发接收到的敏感操作到对应的处理类中。在对应的处理类中，该模块将根据用户预先对各软件权限使用的设置，进行相应的处理。例如，如果用户设置当前敏感权限的使用者对当前使用的敏感权限为拦截状态，则动态权限分发处理模块将会拦截本次权限的使用；如果用户设置为智能处理状态，则该模块将会调用恶意行为智能识别模块进行分析处理。

（5）恶意行为智能识别模块

恶意行为智能识别模块根据基础数据层的特征抽取模块提供的数据来创建朴素贝叶斯分类器，从而判断当前软件行为是否合理，在将结果反馈给动态权限分发处理模块后，更新朴素贝叶斯分类器，并将数据保存到 Sqlite 数据库中。

（6）动态权限决策反馈模块

动态权限决策反馈模块根据动态权限分发处理模块的最终处理结果，对软件本次敏感权限使用选择放行、拦截或者提示用户这三种处理方式。

（7）进程间通信模块

进程间通信模块负责连接 Android 防护软件中位于系统进程与应用程序进程的相应模块，使得它们能够相互通信，保障 Android 系统的安全。

2.3.3 人机交互层的设计

人机交互层的主要功能分为两部分：第一部分是在用户安装 Android 软件的时候，根据安装判别模块的反馈结果，给予用户安全提示，告诉用户该软件中是否存在敏感权限组合，让用户选择是否继续安装，并提供智能安装功能，限制敏感权限组合中的权限在软件运行时被调用。第二部分是呈现拥有敏感权限的软件，并且用户能够对某个软件所能够使用的权限进行动态设置并保存到注册表中，供动态权限分发处理模块的查询与使用。

2.4 本章小结

本章首先在对实际应用的需求和同类型 Android 安全防护软件的功能进行调研和分析，在此基础上，提出了增强 Android 系统安全的具体需求。接着针对具体需求，提出了增强 Android 系统安全的方案框架，主要包括软件安装阶段防护和软件运行阶段防护。然后根据 Android 系统安全增强方案框架，设计了 Android 安全防护软件，并阐述其三层次的体系结构和各个层次中对应的功能模块。在接下来的第三章将通过静态分析 Android 系统中的软件权限来实现软件安装阶段的防护方案，而第四章和第五章分别从技术和理论两个方面来实现软件运行阶段的防护方案。

第三章 Android 系统中软件权限的静态分析

本章首先详细介绍 Android 系统的安全机制，然后对实验采集的 2110 个软件从权限申请的角度进行静态分析，统计权限申请的情况，并实现 Apriori 算法挖掘出恶意软件和敏感权限组合的关联，最后归纳出需要在软件运行时被监控的敏感权限。

3.1 Android 系统的安全机制

Android 系统引入了诸多安全保护机制，本文将这些安全机制分为三类，如表 3-1 所示，分别是：Linux 机制、Android 特有机制和其它安全机制。

表 3- 1 Android 安全机制

机制	描述	防范目标
Linux 机制		
POSIX UID	每一个应用程序被授予一个不同的 UID	防止应用程序执行过程被其它应用程序所篡改
文件访问控制	应用程序的目录只能对该应用自己所公开	防止应用程序数据文件被其它应用程序所访问
Android 特有的机制		
应用程序权限	每一个应用程序必须在安装的时候申请所需要用到的权限	防范应用程序行使恶意行为
应用程序签名	开发者需要对.apk 文件进行签名，来验证不同应用程序来自相同的开发团体，从而能够使用特定的权限	防止第三方软件获得系统签名所独有的权限
Dalvik 虚拟机	每一个应用程序必须运行在自己的虚拟机中	防止缓存溢出攻击与远程代码执行
其它安全机制		
内存管理单元	应用程序只能运行在自己的内存地址空间中	防止信息泄露、权限提升与服务拒绝
强变量类型安全	在编译和运行的时候，变量内容强行限制在特定的格式中	防止缓存溢出攻击

3.1.1 Linux 内核层的安全机制[34]

Android 系统是基于 Linux 内核的，所以 Android Linux 内核层的安全机制直接影响了 Android 底层安全。在 Android 的内核层，有两种主要的安全机制，分别是：POSIX UID[35]和文件访问控制。内核层安全机制的基本元素是用户。

每一个内核中的对象，例如进程或者文件都属于一个用户，用户又被进一步被划分到某个用户组中。

（1）POSIX UID

进程有独立的地址空间，进程与进程间默认是不能互相访问的，是一种很可靠的保护机制。Android 通过为每个安装在设备上的 Android 程序包文件(apk 文件)分配唯一的 POSIX UID。不同的程序包代码不能够运行在同一个进程中，这样不同的软件则不能相互干扰。如果想让不同软件运行在同一个进程中，这些软件必须使用 Android 的 sharedUserID 特性来共享相同的 UID。为了共享相同的 UID，不同的软件必须显式申请它们使用相同的 UID，并且必须使用同一个私钥来对这些软件进行数字签名。

（2）文件访问控制

Android 系统中放在存储设备上的文件，其访问权限是受制于 Linux 权限机制。与 Linux 权限机制一样，文件的访问权限是用三个三元组<Read, Write, Execute>来表示。第一个三元组被用于表示拥有者对该文件的权限，第二个三元组被用于表示拥有者组别的用户对该文件的访问权限，第三个三元组被用于表示其他用户对该文件的访问权限。然而如果该文件运行起来后所具有的权限与 Linux 文件系统上的权限完全不相关，文件运行起来以后所在的进程则受限于 Android 系统特有的软件权限保护机制。

3.1.2　Android 特有的安全机制

Google 开发团队为 Android 设计了以下四项专用的安全机制：软件权限、组件封装、软件签名和 Dalvik 虚拟机。

（1）软件权限

软件权限设置的目的是限制软件对系统或者其它软件的操控能力。在软件中，所被用到的每条权限都以一条字符串的形式被显式地申请在软件的 manifest.xml 文件中。在 manifest.xml 文件中，权限被分为两类：系统内置权限(users-permission 标签)和用户自定义权限(permission 标签)，如图 3-1 所示。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.yang.graduation"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />
```

```
    <uses-permission xmlns:android="http://schemas.android.com/apk/res/android"
        android:name="android.permission.WRITE_OWNER_DATA" />
    <uses-permission xmlns:android="http://schemas.android.com/apk/res/android"
        android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <permission xmlns:android="http://schemas.android.com/apk/res/android"
        android:label="@+id/textElement"
        android:name="com.yang.graduation.READ_APP_DATA"
        android:permissionGroup="@string/app_name" android:protectionLevel="normal" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/logo"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity android:name=".MainFrame"
...
```

图 3- 1 Android Manifest 示例

Android 系统在应用程序框架层引入了 130 个内置权限[36]，包括拨打电话的权限、设置屏幕方向的权限、使用网络的权限等。特定版本的 Android 系统所定义的内置权限定义在该版本源码的 android.Manifest.permissions 类中。

除了 Android 系统内置的权限外，Android 系统支持自定义权限。例如，如果希望让软件使用其它软件提供的服务，则可以定义自定义权限来实现。定义了自定义权限之后，它们将作为组件定义的一部分被引用。

所有的权限还被赋予了不同的保护等级：

- normal：低风险的权限，不会对系统、用户或其它软件造成危害；
- dangerous：高风险的权限，可能涉及用户隐私数据或者造成危害的功能，软件安装的时候会被显式列出，并请求用户确认；
- signature：当软件所用数字签名与声明此权限的软件所有数字签名相同时，这类权限才会被授予；
- signature-or-system：这类权限是 signature 权限的特例，只能被赋予与系统签名相同的软件。

在软件安装的时候，软件所需要的权限将以列表的形式呈现给用户，package installer 会检测该应用请求的权限，如果安装时权限获取失败，则该软件不允许被安装。通常情况下，程序运行未被授予权限的敏感操作时，会引发一个 SecurityException，在系统 log（通过 Log.d 函数访问）中有相关记录。

Android 权限是限制在进程层面的，即一个软件启动的子进程的权限不允许超越其父进程的权限，即使单独运行某个应用有相应权限，但如果它是由一个没有相应权限的软件调用时，那该权限就会被限制。例如，当遇到软件权限不

足时，即使考虑在内核层实现一个 Linux 可执行程序，然后由 Android 软件调用它去完成某个它没有权限完成的事情，这种方法是行不通的。Android 是通过 Linux 内核层的 POSIX UID 机制来实现限制子进程的权限，因为子进程在构建的时候，被分配了与父进程相同的 UID。

（2）软件签名

Android 中的软件签名是权限保护机制的前提，每一个 Android 软件都必须被签名，默认生成的软件文件是 debug 签名。该签名并非基于权威证书，不会决定某个软件是否允许安装，而是基于自签名证书，用于申请 Android 中的 signature 类型和 signature-or-system 类型的权限。

（3）Dalvik 虚拟机

Dalvik 虚拟机可以支持已转换为.dex（即 Dalvik Executable）格式的 Java 软件的运行。经过优化，它允许在有限的内存中同时运行多个虚拟机的实例，其中每一个 Dalvik 应用作为一个独立的 Linux 进程执行。独立的进程可以防止进程间冲突，也可以防止在虚拟机崩溃的时候所有程序都被关闭。

3.1.3　其它安全机制

Android 运行时的环境，例如硬件设备、编程语言等，也为 Android 系统提供了诸多机制来保障系统的安全。

（1）内存管理单元

内存管理单元是一个硬件组件，能够保证不同的进程被划分到不同的地址空间中运行。该硬件组件是诸多现代操作系统架构设计的前提条件。很多操作系统都采用内存管理单元来确保进程不能读取或修改其它进程的内存（信息闭包）。该硬件组件减小了权限提升攻击的可能性因为一个进程不能通过复写操作系统内存来让其程序运行在特权模式。

（2）强类型安全

强类型安全是编程语言的一种特性，该特性强制变量内容符合一种特定的格式，从而避免了缓存区溢出。如果编程语言缺少强变量类型安全机制会可能遭受缓存区溢出攻击，最终恶意程序将能注入恶意代码。

Android 采用强类型的 Java 语言。比起 C 语言能够在没有类型检测和边界检测的情况下执行代码，Java 编写的程序很难做到代码的随意执行。

Android Interface Definition Language(AIDL)是 Android 特有的 Binder 进程间通信机制所使用的语言，它也是强类型安全的。开发者通过 AIDL 语言定制所需要传送的数据元素类型。该语言确保了用于通信的数据也被保存在进程边界中。

3.1.4 安全机制小结

从以上的分析可以看出，Android 系统从多个层面保障了系统的安全。本章主要从软件权限这个角度来分析 Android 的安全机制。如果本文将所有安全机制综合起来考虑，那本文所讨论的安全模型将会变得异常复杂。况且，软件权限的申请是赋予软件操作其它应用或者系统的能力，而 Linux 内核层结合 Dalvik 虚拟机所提供的程序隔离机制是限制软件的能力的。因此，本文假设 Linux 内核层结合 Dalvik 虚拟机所提供的程序隔离机制运行正常，重点关注 Android 权限的获取。

3.2 软件权限的统计分析

3.2.1 数据来源

本文采集的数据集一共由 2110 个 Android 软件组成：其中 850 个软件来自 2013 年 2 月份的 Android Market[37]，分别是 Android Market 里 11 个软件分类和 6 个游戏分类中，排名前 50 的应用；另外 1260 个软件来自于 Zhou 等人[38]的恶意软件数据集，该 1260 个软件都标注为 Android 系统里的恶意软件。本文假定从 Android 市场收集来各个分类前 50 的应用不是恶意程序，因为这些应用通常吸引很多的用户（所有分类下载量第 50 的软件都超过了 100000 次），一般是由 Android 软件公司或者专业组织所开发与维护。因此，Android 软件开发公司或者专业组织为了自身的声誉以及法律法规的约束，并不会在应用中实施恶意行为。所以，我们将这 850 个软件列入白名单，结合 1260 个恶意程序，来展开对软件的静态分析。数据集中软件的分类如表 3-2 所示。

表 3- 2 数据集概况

	分类名称	分类排第 1 的下载量	分类排第 50 的下载量	本文数据集中的数量
软件	通讯&聊天	128076781 次	1263982 次	50
	网络&社区	40082519 次	768232 次	50
	影音&图像	58514118 次	2862179 次	50
	办公&财经	5216025 次	239081 次	50
	资讯&词典	16774536 次	195351 次	50
	旅行&地图	41130661 次	689291 次	50
	输入法&系统	67953884 次	2794265 次	50
	生活&购物	56545553 次	1469522 次	50
	美化&壁纸	18841581 次	254428 次	50
	阅读&图书	11310028 次	183311 次	50

	其它	1898444 次	100876 次	50
游戏	动作冒险	29624496 次	622803 次	50
	角色扮演	11898707 次	271388 次	50
	射击飞行	25745181 次	417421 次	50
	赛车竞速	2088142 次	373192 次	50
	策略经营	7665185 次	427239 次	50
	棋牌休闲	24404234 次	1267457 次	50
恶意程序	恶意程序	X	X	1260

3.2.2　权限统计方法

本文数据集中的软件都是标准 ZIP 兼容的 Android Package（APK）格式。虽然这些软件都是编译以后的文件，并不是很容易进行源代码分析，但是每个软件所申请的权限被写在了二进制化的 manifest.xml，解压读取该二进制 xml 文件是相对容易的。

本文编写了一个 Android 权限抽取的 shell 脚本，如图 3-2 所示。该脚本调用 Google SDK 自带的 Android Asset Packaging Tool[39]，从而抽取出每个 Android 软件中的 manifest.xml 文件，然后读取 manifest.xml 中<uses-permission>和<permission>节点中的内容。如图 3-3 所示，图中 package 行表示软件的开发包名，用于唯一标识一个软件；uses-permission 行表示软件使用的系统内置权限；permission 行表示软件使用的自定义权限。

```
@echo off
setlocal enabledelayedexpansion
::循环读取全部 apk 文件，文件路径存入 i
for /r %%i in (*.apk) do (
::使用 appt 工具来分析 apk 包中的 manifest.xml 文件
set s=%%i
set s=!s:%~dp0=!
::将路径转为文件名
aapt d permissions "!s!"
)
```

图 3- 2 Android 权限抽取的 shell 脚本

```
package: com.netmite.andme
uses-permission: android.permission.INTERNET
uses-permission: android.permission.VIBRATE
```

```
uses-permission: android.permission.SEND_SMS
package: com.qidian.QDReader
permission: com.qidian.QDReader.permission.READ_DB
permission: com.qidian.QDReader.permission.WRITE_DB
uses-permission: android.permission.WRITE_SETTINGS
uses-permission: android.permission.INTERNET
uses-permission: android.permission.WRITE_EXTERNAL_STORAGE
```

图 3- 3 由 shell 脚本读出的 Android 权限示例

由于软件自定义权限由开发者设置，并没有规格化的表示，所以本文对自定义权限不加以分析，只分析系统内置的 130 个权限。

3.2.3 权限统计结果

表 3-3 列举了数据集中 18 个分类的平均权限申请数量。从表中可以看出，恶意程序所平均申请的权限数量为 10.8，大于白名单中软件所申请的平均权限数量 7.1。同样是白名单中的软件，网络&社区类的应用所申请的平均权限数量为 10.2，同样比最小平均权限数量高出了不少，说明不同分类中的软件申请权限的数量也是有差异的。

表 3- 3 各分类软件所申请的权限数

	分类名称	应用程序数量	平均申请的权限数
应用程序	通讯&聊天	50	7.3
	网络&社区	50	10.2
	影音&图像	50	6.8
	办公&财经	50	5.4
	资讯&词典	50	4.5
	旅行&地图	50	6.4
	输入法&系统	50	7.5
	生活&购物	50	4.2
	美化&壁纸	50	4.7
	阅读&图书	50	7.0
	其它	50	8.2
游戏	动作冒险	50	6.3
	角色扮演	50	8.4
	射击飞行	50	5.8

	赛车竞速	50	9.2
	策略经营	50	8.1
	棋牌休闲	50	6.5
恶意程序	恶意程序	1260	10.8

仔细对每个权限分类进行分析以后，图 3-4 给出了前 20 位被频繁申请的权限，其中 y 轴代表每一个在安装时被申请的权限，x 轴代表权限被申请的数量。从图中可以看出，创建网络套接字的 INTERNET 权限被最频繁地申请，它被 2018（95.64%）个软件所申请，其中恶意软件分类中的软件占 1232（97.78%）个，非恶意的软件占 586（68.94%）个。读取通话状态的权限 READ_PHONE_STATE，位列第三，被 1571（74.45%）个软件所申请，其中恶意软件分类中的软件占 1179（93.57%）个，非恶意的软件占 392（46.12%）个。

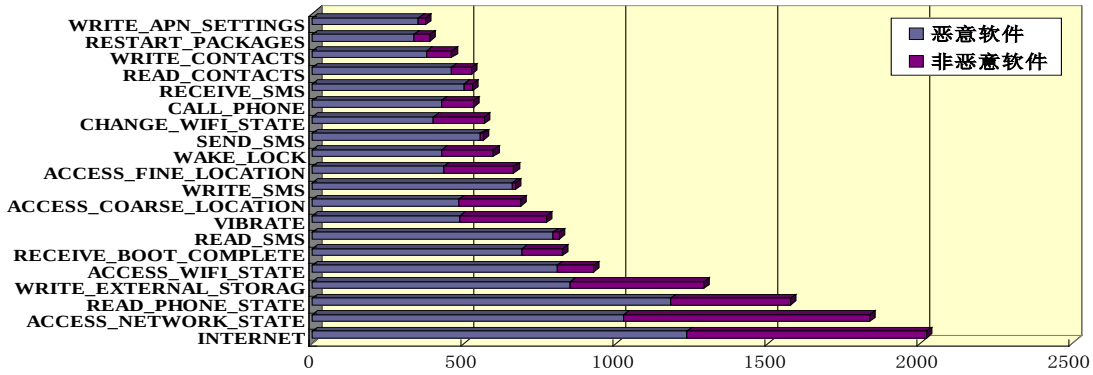

图 3- 4 前 20 位被频繁申请的权限

在对数据集的分析过程中，本文发现在一些软件中，开发者申请了相同的权限两次。这是开发者的问题，因为申请相同的权限多次没有任何的好处。这种重复申请权限的错误不仅仅出现在不出名的软件，例如恶意软件分类中的疯狂打地鼠，它分别申请了 READ_LOGS、WRITE_EXTERNAL_STORAGE 和 RECEIVE_SMS 这些权限两次；也出现在较为受欢迎的软件，如 SOSO 街景地图（下载量为 1882976 次），它申请了 INTERNET 权限两次。这个问题可能是由于程序员个人对权限的错误理解导致的、也可能是开发环境的自动化处理导致的，或者是由于多人开发合作导致重复权限的申请。表 3-4 为重复权限申请数量的汇总。

表 3- 4 数据集中权限重复申请的情况

权限名称	权限重复申请的数量
INTERNET	31
ACCESS_NETWORK_STATE	14
RECEIVE_BOOT_COMPLETED	8

READ_PHONE_STATE	8
READ_CONTACTS	5
ACCESS_FINE_LOCATION	4
READ_LOGS	2
WRITE_EXTERNAL_STORAGE	2
RECEIVE_SMS	2

另外，一些软件会申请 ACCESS_ASSISTED_GPS 和 ACCESS_CELL_ID 这类过时的权限，这些权限在 Android 1.0 系统后就被 ACCESS_FINE_LOCATION 和 ACCESS_COARSE_LOCATION 这两个权限所取代。

同时，本文也发现有些软件会申请 MASTER_CLEAR 这样的权限，该权限能够删除系统所有的配置信息，也就是软格式化（恢复出厂设置）。由于该权限赋予软件软格式化的能力，所以这类权限的使用仅限于与当前 Android 系统拥有相同私钥的软件，并不能够被第三方软件使用。所以即使第三方软件申请了这个权限，在使用的时候进程仍旧会报错。

3.3 基于 Apriori 算法的敏感权限组合挖掘

3.3.1 Android 权限机制在安装时存在的问题

由于 Android 木马通过植入的方式以正常应用程序的形式出现，安全意识不高的用户很可能受到恶意程序的入侵，而一旦安装了恶意程序，用户很难在恶意程序造成伤害之前意识到恶意程序的存在。所以，最好的保护方式是在恶意程序安装时发现可能会导致软件实施恶意行为的威胁并阻止其安装。

当用户在安装 Android 软件的时候，系统会弹出对话框标注该软件所申请的权限列表，如图 3-5 所示，由用户来判断是否授予应用程序这些权限。

图 3- 5 Android 软件安装时显示的权限列表

如果用户仅仅因为 Android 软件申请了单独的某个权限而被判定为恶意软件，这势必导致极高的误报率。因为应用程序申请某项权限是其获得某种能力的途径，而任何一种能力都不应该威胁到系统或者用户数据的安全。如果某项能力本身就会造成安全威胁，那么该项能力就不应该被提供给第三方应用程序，例如 MASTER_CLEAR。所以第三方应用程序所能申请的权限，从单独使用的角度来看，都应该是安全的。

然而，通常 Android 软件会申请多种权限，一旦权限被组合使用，则有可能威胁到系统或者用户数据安全。例如，READ_PHONE_STATE 权限允许程序读取智能终端的相关信息（SIM 卡号、智能终端唯一标识码等），这个权限本身并不会导致用户隐私的泄露，很多应用程序获取这个权限的目的是为了识别出用户所在的省份或者使用的通信服务商，从而为用户提供个性化的服务。但是，如果 Android 软件在拥有获取智能终端相关信息的权限同时，还获得例如 INTERNET 或者 SEND_SMS 权限时，就有可能会造成用户隐私的泄露。另一个例子是 READ_CONTACTS 权限，该权限允许 Android 软件读取用户存储的联系人信息，包括联系人姓名、电话、电子邮件、通信地址、人物关系等。最近特别火爆的 Android 软件“微信”就使用这些信息来识别出使用了“微信”的联系人，从而更为方便用户在移动终端上快速组建出社交圈。然而，如果恶意软件获取到联系人信息以及通信权限，则为群发垃圾短信或者为社会工程学的攻击提供了便利的途径。

虽然有实施恶意行为能力的软件并不一定会实施恶意行为，但是一旦这类软件被安装后将敏感权限组合起来使用，则可能造成更大的破坏。因此，这些具有实施恶意行为能力的敏感权限组合在软件安装阶段就必须被识别出来，并提示用户，引起用户的警觉。

Enck 等人[21]在 CCS09 会议上提出了一个名为 Kirin 的工具，该工具会对预先定义好的、可能进行恶意行为的权限组合做出风险提示。Enck 等人根据经验，预先定义了一些权限组合，包括不能同时拥有获取地理位置、网络和开机启动这三个权限（这会导致恶意软件能够随时跟踪用户的位置）等。然而，Enck 等人的工作存在三个问题：其一，Enck 等人仅仅凭借着个人对 Android 权限的理解定义出一些权限组合，这些权限组合在恶意软件中是否普遍存在？其二，这些权限组合是否也经常存在于非恶意软件中（产生误报率）？其三，这些权限组合是否涵盖了所有威胁用户安全的权限组合？Enck 等人对这三个问题都没有部署相关的恶意软件与非恶意软件进行对比实验。因此，Kirin 工具所预设权限组合的可靠性和全面性是值得质疑的。

针对 Enck 工作的不足，本章采用数据挖掘的方法，在采集的 1260 个 Android 恶意软件和 850 个 Android 非恶意软件的基础上，挖掘出既满足可靠性又满足全面性的权限组合。本章接着根据静态分析的结果，在 Android 软件安装的时候，设计实现了一套权限检测工具，智能化地提示用户规避恶意软件。

3.3.2 敏感权限组合挖掘问题的形式化描述

设 $P=[p_1,p_2,...,p_i,...,p_{130}]^T, p_i \in \{0,1\}$ 为单个 Android 软件申请的权限，其中系统内置的 130 个权限中的每个权限都被标注为一个比特向量位，p_i 表示权限 i 是否在软件安装的时候被申请。设 D 是数据集中所有 Android 软件申请的权限集合。设 X，Y 为某些权限的集合，那么关联规则是形如 $X \Rightarrow Y$ 的蕴含式，其中 $X \subset I$，$Y \subset I$，并且 $X \cap Y = \varnothing$。权限集合 D 中的规则 $X \Rightarrow Y$ 是由支持度 support 和确信度 confidence 来约束的：支持度 support 表示规则出现的频度，即 D 中权限包含 $X \cup Y$ 的百分比；确信度 confidence 表示规则出现的强度，即 D 中包含 X 的权限集同时包含 Y 权限集的百分比。即：

$$support(X \Rightarrow Y) = P(X \cup Y) \tag{3-1}$$

$$confidence(X \Rightarrow Y) = P(Y \mid X) \tag{3-2}$$

定义强规则为同时满足最小支持度阈值 min_support 和最小置信度阈值 min_confidence 的规则。因此，关联规则挖掘问题可以分为两个子问题：

(1) 找出权限数据集中所有大于等于用户指定最小支持度的数据项集，具有最小支持度的数据项集称为频繁项集。

(2) 利用最大数据项集生成所需要的关联规则，根据用户指定的最小确信度确定规则的取舍，最后得到强关联规则。

3.3.3 Apriori 算法原理

Apriori 算法[40]是基于频繁项集的关联规则挖掘算法。其基本思想是利用已知的频繁项集推导其它频繁项集，是一种广度优先算法。对应于关联规则挖掘的两个子问题，Apriori 算法分为以下两步：

(1)链接步骤

链接步骤是为了在现有的频繁集上找出能构建更大一维的频繁集。

为找 k 项频繁集 L_k，首先通过将 $k-1$ 项频繁集 L_{k-1} 与自身连接，产生候选 k 项集的集合，该候选项集合记做 C_k。

(2)剪枝步骤

剪枝步骤是根据支持度和置信度缩减当前频繁集的数目。

C_k 是 L_k 的超集，即 C_k 的成员未必是频繁的，但所有频繁 k 项集都包含在 C_k 中。扫描数据库，确定 C_k 中每个候选的支持度与置信度，从而确定 L_k。然而，C_k 可能很大，这样所涉及的计算量就很大。为压缩 C_k，可以使用 Apriori 性质。任何非频繁的 *k-1* 项集都不是频繁 k 项集的子集。因此，如果候选 k 项集的 *k-1* 项子集不在 L_{k-1} 中，则该候选也不可能在 L_k 中，从而可以直接从 C_k 中删除。

3.3.4　恶意权限组合

本章通过实现 Apriori 算法，对申请数量排前 30 的权限进行挖掘。首先本章构建输入参数 $I=[p_1,p_2,...,p_i,...,p_{30},m]^T$ 表示软件所申请的权限，其中 $p_i \in \{0,1\}$ 表示权限 i 是否在软件安装的时候被申请，$m \in \{0,1\}$ 表示该软件是否是恶意软件数据集中的软件。然后将 I 中 *m=1* 和 *m=0* 的数据条目分开，分别使用 Apriori 算法进行关联规则的挖掘，得到关联规则集 A_m 和 A_b。

Apriori 算法能够根据设定的支持度和置信度值，挖掘出所有的关联规则，但是其中很多的关联规则包含了很长的前置条件，而过长的前置条件只是由于数据挖掘算法本身的搜索条件所到导致的，并不包含通用的、含有特定意义的关联规则，所以本文只选取了前置条件小于等于 3 的敏感权限组合，得到缩减后的关联规则集 B_m 和 B_b。

由于本文并不希望被挖掘出的敏感权限组合既存在于恶意软件中又存在于非恶意软件中，因为这样的敏感权限组合并不能用于判断软件是否是恶意的。所以本文定义恶意软件组合 C_m 为仅存在于 B_m 中，而不存在于 B_b 中，即 $C_m=B_m-B_b$。只有恶意权限组合才能够在误报率较低的情况下识别恶意软件。

经过上述步骤对软件申请的权限进行关联规则挖掘之后，表 3-5 列出部分置信度 85%以上仅存于恶意软件中的敏感权限组合

表 3- 5 恶意权限组合

恶意权限组合	置信度	支持度
INTERNET 与 WRITE_SMS	0.90	0.40
READ_PHONE_STATE 与 WRITE_SMS	0.90	0.39
INTERNET 与 READ_PHONE_STATE 与 READ_SMS	0.88	0.47
ACCESS_WIFI_STATE 与 READ_SMS 与 WRITE_SMS	0.88	0.30

WRITE_SMS 与 RECEIVE_SMS 与 RESTART_PACKAGES	0.88	0.17
READ_PHONE_STATE 与 WRITE_SMS 与 WRITE_CONTACTS	0.87	0.20
WAKE_LOCK 与 WRITE_CONTACTS 与 RESTART_PACKAGES	0.87	0.15
READ_SMS 与 SEND_SMS 与 RECEIVE_SMS	0.86	0.25
READ_SMS 与 RECEIVE_SMS 与 RESTART_PACKAGES	0.85	0.18
WRITE_EXTERNAL_STORAGE 与 READ_SMS 与 WRITE_SMS	0.85	0.26

从表中可以看出，恶意权限组合通常总会伴有通信权限的申请。这些权限不但可以作为上传用户隐私信息的媒介，而且会造成用户话费的损失。例如表 3-5 中第三项，如果软件在安装的时候，既申请了 INTERNET 权限又申请了 READ_PHONE_STATE 和 READ_SMS 权限，那该软件有 88%的概率是恶意软件，而且在恶意软件数据集中，有近一半的软件同时申请了这三个权限。拥有这三个权限的 Android 软件能够在后台偷偷地将用户的设备信息和短信信息上传到指定的服务器，从而达到恶意采集用户隐私的目的。又例如，Android 软件同时申请了 READ_SMS、SEND_SMS 与 RECEIVE_SMS 权限，那该软件也有 88%的概率是恶意软件，而且有近 1/3 的恶意软件同时申请了这三项权限。通常情况下，Android 软件如果同时拥有这三个权限，那该软件可以完全控制用户的短信系统，在屏蔽用户所有短信的情况下，发送任意的短信，从而使得软件能够在用户不知情的情况下，订阅收费的服务。

3.3.5 智能安装方案

图 3- 6 智能安装的实现框架

图 3-6 显示了智能安装方案的实现。该方案作为一个单独的应用程序，通过进程间通信接口和原有的程序安装器进行交互，从而获得应用程序安装包的路径，通过 PackageManager 和 PackageParser 接口得到应用程序申请的权限信息，然后从权限数量和恶意权限组合两方面在 Android 软件安装的时候判别软件在安装后是否可能实施恶意行为，从而提示用户可能会存在的安全威胁。此时用户可以选择智能安装方式，使得用户在安装了带有恶意权限组合的软件后，

这些恶意权限组合被设置为拦截状态。对 Android 系统的用户来说，相比较于逐条查看软件安装时使用的权限列表，该方案更为智能地帮助用户规避恶意软件。

3.4 需要实时监控的敏感权限

虽然智能安装能够防范一部分的恶意软件的安装，但是该方案并不能够要求用户拒绝安装所有被标记为恶意的软件。如果用户仍旧安装了带有恶意权限组合的软件以后，则必须对这些软件在 Dalvik 虚拟机中的执行情况做动态实时地监控。一旦涉及用户的隐私信息或者需要付费的危险操作，则应该有能力及时拦截。

通过以上分析，本章选取恶意软件中涉及付费和用户隐私获取的常用敏感权限，作为动态实时拦截的对象，如表 3-6 所示。

表 3- 6 恶意软件常用的敏感权限

权限名	权限分类	权限说明
CALL_PHONE	需要付费的服务	直接拨打电话号码
SEND_SMS	需要付费的服务	发送短信
INTERNET	需要付费的服务	访问网络
READ_PHONE_STATE	隐私权限	获取手机号、设备识别码
READ_CONTACTS	隐私权限	访问联系人
READ_SMS	隐私权限	读取短信内容
RECEIVE_SMS	隐私权限	拦截收到的短信
PROCESS_OUTGOING_CALLS	隐私权限	读取通话状态
ACCESS_COARSE_LOCATION	隐私权限	读取粗略位置信息
ACCESS_FINE_LOCATION	隐私权限	读取精细位置信息

从表中可以看出，有 3 种权限（android.permission.CALL_PHONE、android.permission.SEND_SMS、android.permission.INTERNET）是会收取用户的费用，这里所说的付费不单单是普通的通信费用（发短信，打电话，数据流量费），还包括使用短信订购大量付费套餐（通过拦截短信接收并分析短信内容能够在用户不知情的情况下实现自动订阅）、拨打高额信息费的电话、网络订阅付费服务、通过话费购买其它 Android 应用等。

另外有7种权限，会威胁到用户的隐私信息，窃取这些信息，再结合上述3种付费权限，则恶意程序提供商就能够很容易对用户的隐私造成危害，例如根据用户的隐私开展诈骗活动，从而给用户带来难以想象的危害。

软件在安装时被赋予了这些权限以后，如果滥用这些权限，则会对 Android 用户造成很大的危害，所以在后续章节文本实时监控了这 10 种权限在 Dalvik

虚拟机中的使用情况，并提出智能算法来识别软件的恶意行为，从而能智能化地对权限的恶意使用进行拦截。

3.5 本章小结

本章在详细介绍 Android 系统所采用的安全机制后，对实验采集的 2110 个软件从权限申请的角度进行静态分析。分析发现，恶意软件往往申请的权限比非恶意软件所用到的权限多。在对权限进一步分析以后，本章通过 Apriori 算法挖掘出恶意软件和敏感权限组合的关联，并阐述了智能安装的实现方案。由于智能安装方案并不能杜绝恶意软件的安装，所以本章接着提取出 10 种恶意软件常用的敏感权限，在本文的第四章将以监控这 10 种敏感权限为目标，研究 Android 系统敏感权限的动态实时监控技术，并在第五章提出智能算法来识别软件恶意行为，给予 Android 用户全方位的防护。

第四章 Android 系统敏感权限的动态实时监控

对 Android 系统的敏感权限进行实时监控是研究 Android 软件恶意行为智能识别的前提条件。本章在调研 Android 平台中相关技术的基础上，分析 Android 系统源代码，给出 Android 系统中短信发送和权限审查操作的具体实现，然后归纳出 Android 系统敏感权限操作的流程，最后根据该流程设计 Android 系统敏感权限的动态实时监控机制。

4.1 相关技术

为了更好的对 Android 源代码进行分析，首先本文介绍 Android 系统特有的技术。

4.2.1 Android 组件

Android 一共有有四种组件[41]，而这四种组件是构成 Android 软件的必要元素，它们分别是：Activity、Service、Broadcast Receiver、Content Provider。

（1）Activity

在 Android 软件中，Activity 是一个用户界面的概念，一个软件可以拥有多个 Activity。实际上一个 Activity 就是一个窗口，而里面的内容就是各种 View。通过 setContentView 函数将窗口和内容联系在一起。

（2）Service

实际上，Service 可以就看成没有用户界面而运行在后台的组件，在 Android 中，Service 又分为两种：一种就是 Service 运行在本地进程中，也就是说与软件运行在一个进程之中；还有一种就是 Service 运行在其它的进程中。

（3）Broadcast Rreceiver

在 Android 中，系统或者其它软件在完成某个特定行为以后，通常会发出一个广播信号来通知 Android 系统中的进程，例如在系统收到一条短信息，会发出一个全局的 SMS_RECEIVED 广播。如果其它进程注册了这个广播，则能够捕获接收到的短信，对其进行操作。但务必注意的是，本文所需要捕获的进程敏感行为，例如向外发送短信、向外拨打电话这类操作，Android 系统是不会发出广播的，所以没有办法通过捕获广播的方法来实时抓取这些进程的敏感行为数据。况且，广播机制是一种事后触发的机制，所以即使 Android 系统提供了某些敏感行为的广播，而且假设本文设计的拦截机制能监听到这些敏感行为的广播，也没有办法对软件的恶意行为进行拦截。

（4）Content Provider

Content Provide 是 Android 系统为软件提供的第三方应用数据的访问方案。根据前一章对 Android 安全机制的分析可知，Android 系统有着严格的软件隔离机制。但是作为软件，与其它软件的沟通通常是必须的，很难想象一个短信应用不能访问联系人应用中的数据。所以，Android 系统提供了 ContentProvider 基类，从而对外公布数据。

Android 软件中用到的每一个组件都必须在 manifest.xml 文件中申请。每一个组件在申请的时候都有一个进程选项，可以用来指定该组件所运行的进程，可以让软件的组件运行在一个进程中，或者让多个软件的组件运行在一个进程中从而共享彼此的数据。除了这个方法，Android 专门为软件之间的通信设计了 AIDL 语言，通过该语言，开发者能够很方便地进行 IPC 编程。

4.2.2　AIDL 服务

Android Interface Definition Language（AIDL）是 Android 进程通信接口的描述语言，通过它开发者可以定义进程间的通信接口，从而实现 Android 进程间通信。

AIDL 服务是基于 Android Service 组件之上、能够被多个软件调用、提供数据交换的技术。其数据交换机制与 EJB 所采用的 CORBA、微软提供的 DCOM 和.NET Remoting、简单对象访问协议 SOAP 或者 ZeroC 开发的 ICE 是十分类似的。

进程之间的通信信息，首先会被转换成 AIDL 协议消息，然后发送给 AIDL 服务，AIDL 服务收到协议消息后再转换成相应的对象。由于进程之间的通信信息需要双向转换，所以 Android 采用代理类实现了信息的双向转换。其中代理类由 Android 编译器自动生成，对开发者来说是透明的。

4.2.3　Intent 机制

Intent 是对执行某个操作的一个抽象描述，同时负责提供组件之间相互调用的相关信息传递。例如，在一个短信应用中，当用户点击短信列表中某一项后，希望能够跳出该条短信的具体内容。为了实现这个目的，短信列表需要构造一个 Intent 告诉系统用户要做“查看”动作，以及该动作对象是“某条短信”，然后调用 startActivity(intent)接口将原先构造出的 Intent 传入；系统会根据该 Intent 的描述，在 Manifest.xml 文件中找到满足该 Intent 条件的 Activity，从而启动满足条件的 Activity。

作为一个完整的消息传递机制，Intent 不仅需要指定发送端，还需要指定接收端。对接收端的指定可以是显示的也可以是隐式的。如果没有明确指定目标

组件，那么 Android 系统会使用 Intent 里的（action，data，category）三个属性来寻找和匹配接收端。

4.2 Android 系统源码分析

本文通过对 Android 源代码分析，归纳出 Android 系统进行敏感操作的一般流程。

4.3.1 短信发送流程分析

本节通过分析Android系统源代码，总结Android系统发送短信的流程，分为Android系统短信的发送准备阶段和Android系统短信的实际发送阶段。

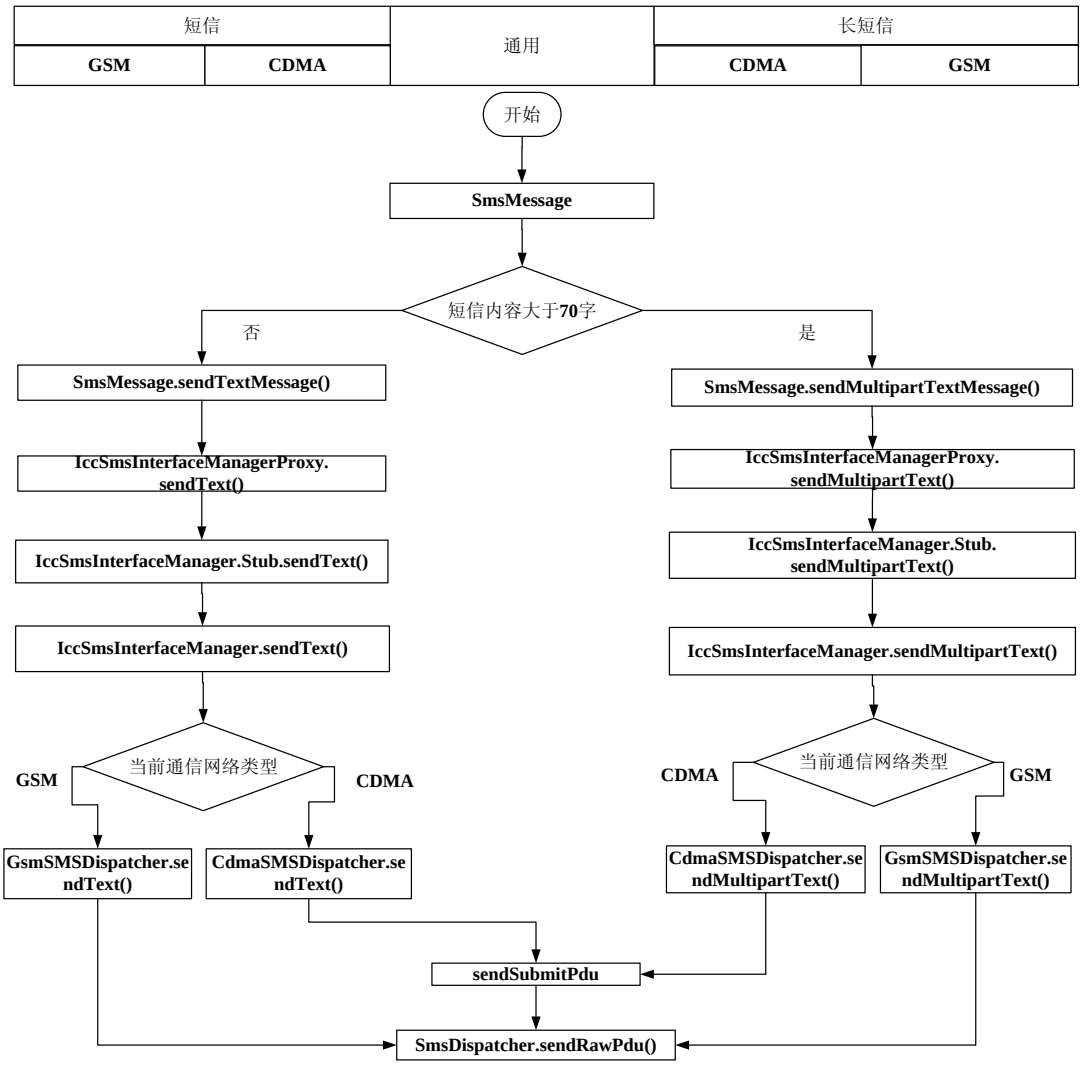

图 4- 1 Android 系统短信的发送准备阶段

图 4-1 是 Android 系统短信发送的元数据准备阶段。程序开发者使用 SDK 中提供的 SmsManager 类发送短信的时候，如果短信内容超过 70 个 Unicode 字符的话，分别调用 sendMultipartTextMessage 和 sendTextMessage。两个函数的基本流程是一样的，只是 sendMultipartTextMessage 在分割短信的基础上，为每个短信分段设置了 SmsHeader。Isms.sendText 通过 Android Binder 机制绑定底层代理服务 IccSmsInterfaceManagerProxy.sendText()，然后由该代理类调用真正提供短信发送功能的服务接口。在绑定底层代理服务的过程中，Android 系统会审查软件是否有绑定特定系统服务的权限，例如绑定短信发送服务，需要有 Android.permission.SEND_SMS 权限。接着 IccSmsInterfaceManager 这个抽象类会根据设备网络类型的不同而调用 SmsDispatcher 中对应的方法，其大致过程是：（1）获取网络编码方式；（2）对每一个分段构造 SmsHeader 对象、SubmitPdu 对象和 SmsTracker 对象，并设置对应的 RadioTechnologyFamily；（3）把消息发送过程记录在 SmsTracker 对象中。

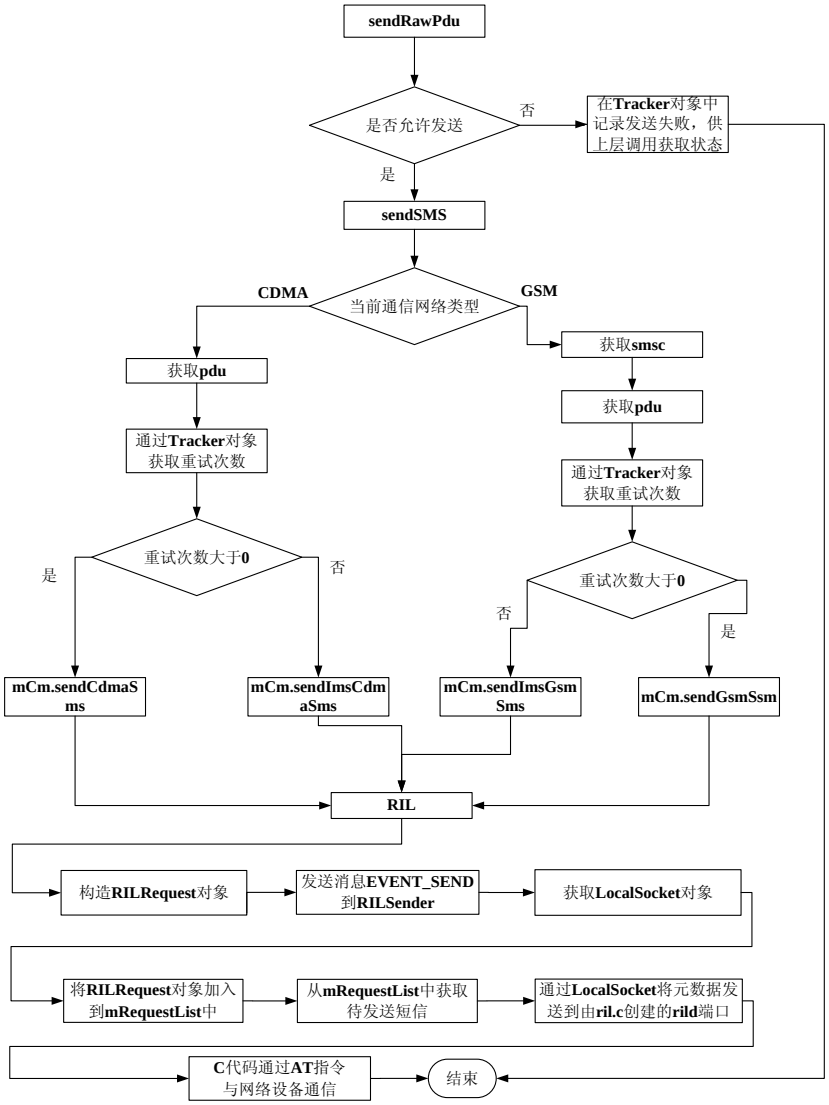

图 4- 2 Android 系统短信的实际发送阶段

图 4-2 是 Android 系统短信的发送阶段。在生成了 RawPdu 以后，系统检查当前网络状况是否能够发送短信，然后就选择合适的网络开始真正发送短信了。由于网络协议的不同，在发送短信之前，GSM 网络需要获取短信提供中心的相关数据（smsc），其它过程与 CDMA 网络相同，最后短信元数据被下发到 Android Radio Interface Layer（RIL）中。

RIL 在 Android 系统中位于应用程序框架层与 Linux Kernel 层之间，从广义上来说，RIL 的设计是为目前智能终端上流行的双 CPU 架构（基带处理器和应用处理器）提供可靠通信。RIL 主要由两个部分组成，其一是 rild 进程，负责向上建立 socket 与应用程序框架层通信；另一个部分是 Vendor RIL，负责向下通过两种方式与基带处理器进行通信，分别是直接通信的 AT 指令通道和用于智能终端联网的数据包传输通道。

当短信元数据进入 RIL 层之前，应用程序框架层会构造 RILRequest 对象，并通知发送 EVENT_SEND 到 RILSender 中，获取返回得到的 LocalSocket 对象，并将 RILRequest 填塞入发送队列，最后通过 LocalSocket 对象发送到 rild 进程中。在 rild 进程接收到应用程序框架层下发的短信之后，通过现有网络设备的 AT 指令，驱动硬件发送短信。

4.3.2 权限审查流程分析

从上述短信发送的源码分析中可以看出，Android 系统中执行某项具体操作的流程是比较复杂的。本文需要对 10 种敏感权限的操作进行监控，如果对这 10 中敏感权限的操作一一分析，分别找到每个操作可被监控的突破口，工作量是比较大的，而且写出的代码也很难满足软件工程中对可维护性的需求。

然而仔细分析上述短信发送的流程可以发现，在应用程序层调用 SDK 提供的接口执行敏感操作的时候，实质上该接口是一个抽象类函数，其具体实现是通过 Android Binder 机制绑定 Android 系统底层服务，由底层服务来完成具体的操作。而在绑定 Android 系统底层服务的时候，需要通过 Android 权限的审查，即 Android 系统会在软件每次执行这些敏感操作的时候，都去审查该应用在安装的时候是否被赋予了相关的权限，而这就可以作为监控系统所有敏感操作的突破口。

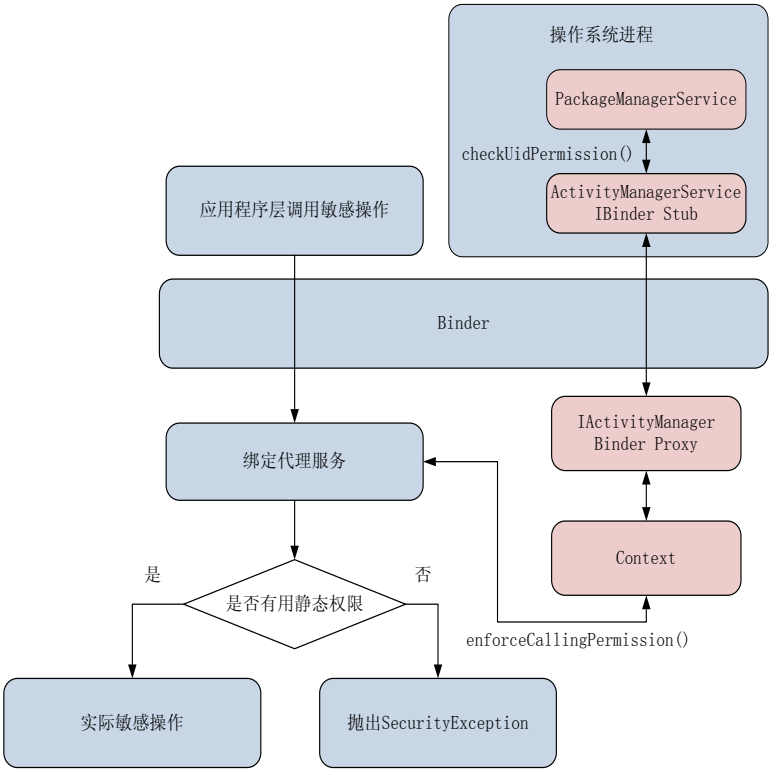

图 4- 3 权限审核流程源码分析

基于该突破口，本文继而调研了 Android 权限审查机制的具体实现，如图 4-3 所示，当应用程序调用敏感操作的时候，例如调用拨打电话，读取联系人等，都会通过 Android Binder 机制将应用程序层的操作绑定到应用程序框架层中对应的代理服务类。在代理服务类中，会调用当前应用程序组件(Context 类型)的 enforceCallingPermission 方法来验证应用层的操作是否有所调用的代理服务类权限。Context.enforceCallingPermission 方法的具体执行流程是通过 aidl 进程间通信机制将所需要查询的应用程序 uid 和应用程序所使用的权限发送给系统进程。系统进程主要由 PackageMangerService 这个类来验证该 uid 对应的应用程序是否在安装的时候被赋予了所要使用的权限。系统进程在启动的时候即将所有已经安装的应用程序在安装过程中申请的权限都读到了内存中，当应用程序在执行敏感操作的时候，系统进程则直接调用 PackageManager.checkUidPermission 方法。该方法返回 PERMISSION_GRANTED 或者 PERMISSION_DENIED 给代理服务类，然后代理服务类根据反馈结果决定是调用敏感操作还是抛出 Android 系统的 SecurityException。

4.3 动态实时监控机制的设计

本文在分析 Android 短信发送源码基础上，发现在每次系统敏感操作执行的过程中，Android 权限管理机制都会对当前操作的发起者进行权限的审查，从而核实该应用在安装的时候是否被赋予了相关的权限。但是，这种软件权限静态审核的思想防御不了程序的伪装欺骗，这些权限在软件安装时被用户点击确认授权之后就不再提醒，并不能动态修改软件的权限，也不能监控权限的使用情况。但这个审查过程可以被利用来实现本章所提出的恶意行为动态实时监控机制，即在 Android 应用程序框架层调用系统敏感权限的时候设置函数拦截，然后将拦截到的权限名称和正在调用的进程名称上传到中心控制软件，通过中心控制软件对该权限的使用作出判断，然后给予反馈。

Android 系统敏感权限的动态实时监控的执行过程如图 4-4 所示，分为五个阶段：

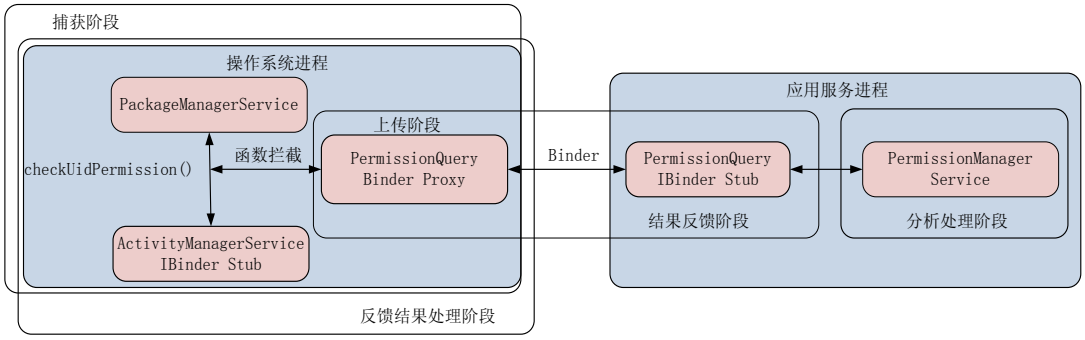

图 4- 4 Android 系统敏感权限的动态实时监控

（1）捕获阶段

捕获阶段通过函数拦截将正在使用敏感权限的进程名称识别出来。主要工作分为两部分：一是实时捕获所有权限使用的记录；二是设置过滤器，将敏感权限的使用过滤出来，并构建成一个二元组形式<packageName, permission>，为中心控制程序的判断做准备。

（2）上传阶段

上传阶段的任务是将 Android 应用程序框架层所捕获、过滤并构建出来的二元组上传到应用程序层的中心控制程序中。这部分采用 Android 的 aidl 进程间通信机制来实现。由于捕获阶段是处于 Android 操作系统进程中，而权限动态判断过程是处于应用程序层的中心控制程序中。为了将这两个程序关联起来，从而使得应用程序层的中心控制程序能够获取到 Android 应用程序框架层中算法的反馈结果，则需要采用进程间通信机制。该阶段在系统进程中需要注册类似于函数回调机制的接口；中心控制程序则负责实现该接口，使得 Android 程序框架层的系统进程能够将数据上传给中心控制程序。

（3）分析处理阶段

分析处理阶段的任务是根据用户在程序界面上面所设置的单个应用进程的监控模式，来对所接收到的特定应用进程的权限使用请求做出反馈。这部分工作首先查找二元组中<packageName, permission>中软件的应用进程名，根据应用进程名，找到中心控制程序中用户对该软件所设置的模式<ALWAYS_ALLOW, ALWAYS_DENIAL, ASK_USER, ARTIFICIAL_INTELLIGENCE>，然后根据不同的反馈模式所处理的结果，生成统一的结果格式，准备传回系统进程。

（4）结果反馈阶段

结果反馈阶段的任务是将 Android 应用程序层的判断结果回传给 Android 应用程序框架层。该阶段同样采用了 Android 的 aidl 进程间通信机制来实现。

（5）反馈结果处理阶段

反馈结果处理阶段的任务是执行中心控制程序的反馈结果。这个阶段所得到的最终结果分为三种形式，分别为 ALLOW、DENY 和 ASK_USER。根据这三种反馈形式，应用程序框架层分别进行放行权限的使用、拦截权限的使用或者询问用户意见这三种处理方式，并且将执行结果存入 Android 自带的 Sqlite 数据库。

4.4 本章小结

本章研究了 Android 系统敏感权限的动态实时监控技术。在调研了 Android 系统组件、AIDL 服务、Intent 机制等相关技术的基础上，分析了 Android 系统的源代码。其中，详细分析了 Android 系统中短信发送和权限审查操作的具体实现，然后归纳出敏感权限操作的流程，并根据该流程实现了 Android 系统敏感权限的动态实时监控。

第五章　基于朴素贝叶斯分类器的恶意行为智能识别

智能识别软件的恶意行为是开发智能的 Android 安全防护软件的一个关键问题。本章将研究使用朴素贝叶斯分类器结合软件属性特征和系统运行环境特征来识别 Android 软件的恶意行为。首先，本章将恶意行为识别问题抽象为分类问题，并抽取用于分类的属性特征；然后，使用朴素贝叶斯分类器识别软件的恶意行为，并分析贝叶斯决策原理，找出导致误报率高的原因；最后，改进贝叶斯决策方法，提出基于朴素贝叶斯分类器的恶意行为智能识别算法，并分析算法在真实数据中的实验结果。

5.1 问题描述

Android 系统中的软件恶意行为识别问题可以认为是通过监控 Android 软件运行时对各敏感权限的使用结合软件本身的基础属性以及系统运行环境，从而判断软件的行为是否为恶意的分类问题。因此，Android 系统中软件恶意行为的识别问题可做如下定义：

给定集合：$Y=\{y_1,y_2,...,y_n\}$ 和 $X=\{x_1,x_2,...,x_n\}$，以及映射规则 $y=f(x)$，对任意 $x_i \in X$ 有且仅有一个 $y_i \in Y$，使得 $y_i=f(x_i)$成立。

其中 Y 为类别集合，其中每一个元素为一个类别，在软件恶意行为识别的问题中，Y 有两个类别，即恶意行为和非恶意行为。X 为特征属性序列集合，其中每一个元素为一类特征属性序列。f 为分类算法，分类算法的任务是基于分类器来构造出分类方法。分类问题通常由于真实环境中存在不确定的信息而不能构造 100%正确的映射规则。一般情况下分类器是通过对输入样本数据的学习从而实现一定概率上正确的分类，因此，所学习生成的分类器并不一定能将每个未知分类项 X_i 准确地映射到类别 Y_i。分类算法的准确率与分类器构造算法、待分类数据属性特性的选取以及训练样本大小等诸多因素有关。

5.2 智能识别方案

5.2.1　方案框架

基于朴素贝叶斯分类器的恶意行为智能识别的方案框架图如图 5-1 所示，主要由五部分构件组成。

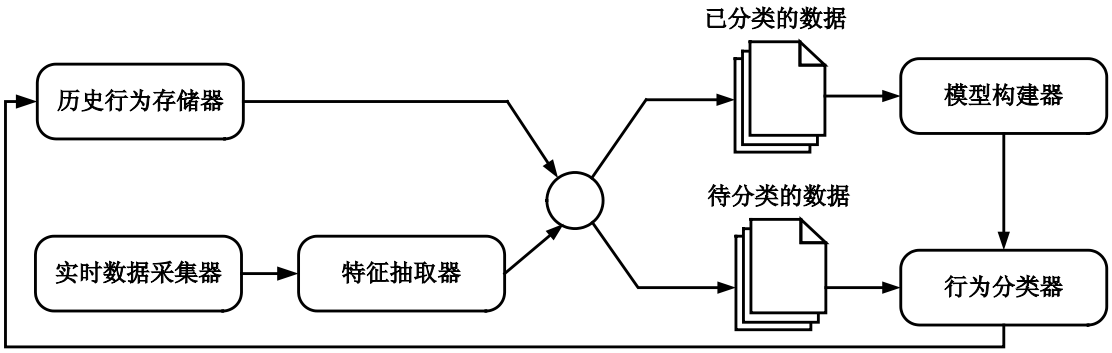

图 5- 1 恶意行为智能识别方案框架

第一个组件是实时数据采集器，负责采集Android软件运行在Dalvik虚拟机时所被监控到的权限申请记录以及各软件的基础属性。

第二个组件是特征抽取器，负责处理数据采集器中所采集到的数据，并从这些数据抽取出能够反映软件恶意行为的特征。

第三个组件是历史行为存储器，用于存储已被分类标记过的特征序列，从而为下一次智能识别提供历史依据。

第四个组件是模型构建器。当数据被标记过之后，被标记过的数据则被放入模型构建器。该组件基于历史行为存储器中已标记过的条目来构建恶意行为检测模型。接着，这些构建好的模型，伴随着未被训练过的数据，作为输入，被送入行为分类器中。

第五个组件是行为分类器，其主要作用是根据模型构建器中所构建出来的模型，对未标记过的软件行为进行分类。在本文中，行为分类器将未标记过的数据分为两类，即恶意行为、非恶意行为。

5.2.2　特征选取与抽象

为了准确地识别出Android软件使用敏感权限是否属于恶意行为，则必须识别出有代表性的属性特征序列来协助判断软件的恶意行为。本文在Android系统的设备上安装了83个软件（其中恶意软件49个、非恶意软件34个），运行了40天，对10个敏感权限的使用进行了详细的记录和分析，共采集了8849条敏感权限的使用记录。同时，本文监控并抽取的系统数据包括：敏感权限使用的时间，用户每次操作智能设备的时间，权限申请的类别，软件安装时所申请的权限，软件是否是系统自带软件。对这些数据分析完以后，本文挑选出5个能够刻画出用于分析Android软件恶意行为的属性特征：

敏感权限的使用者是否是系统应用　系统应用指的是智能终端在安装Android系统时自带的软件，例如短信应用、拨号应用、通讯录、固件管理器、邮件服务等。通常这些软件不会恶意收集用户信息或者恶意耗费用户话费。即使有些Android系统中预装数据采集的软件，也会告知用户，并在获得用户的同意

时进行采集。例如CyanogenMod系统会预装名为CM数据统计的软件，用于统计用户当前使用的CM版本号，当系统第一次被安装的时候启动，并鼓励用户加入该数据采集计划。所以，敏感权限的使用者是否是系统应用可以作为被抽象出来的一个特征。

权限使用时是否有用户操作 通常情况下，用户并不希望在不知情的情况下，软件偷偷地对智能终端进行操作，尤其是涉及敏感数据的操作。所以，权限使用时是否有用户操作需要被抽象出来作为恶意行为识别的一个特征。

敏感权限的使用者是否在安装时申请了大量权限 恶意软件为了能够对智能终端系统更大程度上的控制，通常申请的权限远远大于真实功能所需要的权限。从3.2节对软件申请的权限数量的统计分析可知，通常恶意软件申请的权限数量比热门软件的权限数量多，所以一旦敏感权限的使用者在安装的时候申请了大量的权限，则其对敏感数据的操作会被怀疑成恶意行为。

敏感权限的使用者是否在安装时存在恶意权限组合 通常恶意软件所申请的多种权限，被组合在一起使用的时候，则有可能威胁到系统或者用户数据的安全。从3.3节对恶意软件组合的挖掘结果可知，当软件申请了恶意权限组合时，该软件是恶意软件的可能性极大，当被赋予了这些恶意权限组合以后，则该软件得到了执行恶意行为的能力。虽然有实施恶意行为能力的软件并不一定会实施恶意行为，但是一旦这类软件调用敏感权限时，则可能造成更大的破坏。因此，敏感权限的使用者是否在安装时存在恶意权限组合需要被考虑作为恶意行为识别的一个特征。

敏感权限的使用是否存在突发性 对实验所采集的8849条敏感权限的使用进行分析以后，发现很多的恶意软件在启动的时候，会伴随大量的隐私权限的使用。例如Gone in 60 seconds这个软件，会在软件启动的时候使用查看通讯录、短信记录、通话记录这些敏感的权限，然后在用户难以反应过来的速度内，将这些信息上传到远端服务器，然后关闭应用进程；另一个软件moonsms在打开的时候，会连续向一个特定号码发送一连串特定代码含义的短信，试图通过短信的方式进行远程控制，然而在数据集中挑选出的34个热门软件里，并没有发现类似现象。所以敏感权限的使用是否呈现突发特征也可以被抽象成恶意行为识别算法中的一个特征。

5.3 基于朴素贝叶斯分类器的恶意行为智能识别算法

5.3.1　朴素贝叶斯分类器

本章使用贝叶斯理论[42]构造用于恶意行为识别的分类器。贝叶斯分类是有监督的学习过程，它通过训练集归纳出贝叶斯分类器，并利用贝叶斯分类器对没有分类的数据进行分类。贝叶斯分类具有如下特点：

(1) 贝叶斯分类选择后验概率最大的类别作为分类结果；

(2) 贝叶斯分类中所有的属性都对分类结果有贡献作用；

(3) 贝叶斯分类对象的属性既可以是离散的也可以是连续的或者混合的。

朴素贝叶斯分类器[43]是贝叶斯分类模型里一种简单、高效而且在实际中被广泛使用的分类器，特别适合检测软件恶意行为这类需要实时判别的应用场合，具有如下优点：

(1) 算法逻辑简单，易于实现；

(2) 算法实施的时间、空间开销小；

(3) 算法性能稳定，对于不同特点的数据其分类性能差别不大，即分类器的健壮性比较好。

朴素贝叶斯分类器是在贝叶斯分类器的基础上，假定特征属性值是互相独立的，即在特征属性间不存在依赖关系。构建朴素贝叶斯分类器的过程可以描述为：

步骤 1 令每个数据样本用一个 n 维特征向量 $X=\{x_1,x_2,...x_n\}$ 分别描述对 n 个属性样本的度量。

步骤 2 给定 m 个类 Y_1、Y_2、…Y_m 和一未知的数据样本 X，分类法将预测 X 属于具有最高后验概率的类。即朴素贝叶斯分类将未知的样本分配给类 Y_i，当且仅当

$$P(Y_i \mid X) > P(Y_j \mid X), j \neq i \tag{5-1}$$

步骤 3 根据贝叶斯公式：

$$P(Y_i \mid X) = \frac{P(X \mid Y_i)P(Y_i)}{P(X)} \tag{5-2}$$

由于 $P(X)$ 对于所有类为常数，分类器得求最大的 $P(X \mid Y_i)P(Y_i)$。分类 Y_i 的先验概率可以用 $P(Y_i)=s_i/s$ 计算，其中 s_i 是分类 Y_i 中的训练样本数，而 s 为

训练样本总数。如果类的先验概率未知，则通常假定这些类是等概率的，即 $P(Y_1)=P(Y_2)=\ldots=P(Y_m)$，此时只需要求最大的 $P(X \mid Y_i)$。

步骤 4 由于朴素贝叶斯分类器假定属性值相互条件独立，即：

$$P(X \mid Y_i)=\prod_{k=1}^{n} p(x_k \mid Y_i) \tag{5-3}$$

由此，概率 $P(X_1 \mid Y_i)$、$P(X_2 \mid Y_i)$、$\ldots P(X_n \mid Y_i)$ 可以由训练样本估值，其中如果 A_k 是离散值属性，则 $P(X_k/C_i)=s_{ik}/s_i$，其中 s_{ik} 是在属性 A_k 上具有值 x_k 的类 Y_i 的样本数，而 s_i 是 Y_i 中的训练样本数。如果 A_k 是连续值属性，则通常假定该属性服从正态分布，即：

$$P(X_k \mid Y_i)=g(x_k,\mu_{C_i},\sigma_{C_i})=\frac{1}{\sqrt{2\pi\sigma_{C_i}}}e^{\frac{\left(x_k-\mu_{C_i}\right)^2}{2\sigma_{C_i}^2}} \tag{5-4}$$

其中，给定类 Y_i 的训练样本属性 A_k 的值，$g(x_k,\mu_{C_i},\sigma_{C_i})$ 是属性 A_k 的正态密度函数，而 μ_{C_i},σ_{C_i} 分别为平均值和标准差。

步骤 5 为了对未知样本 X 进行分类，对每个类 Y_i，计算 $P(X \mid Y_i)P(Y_i)$。样本 X 被分类为 Y_i，当且仅当

$$P(X \mid Y_i)P(Y_i)>P(X \mid Y_j)P(Y_j), j \neq i \tag{5-5}$$

分类中如果出现 $P(X \mid Y_i)$=0，即某个类别中某个特征项没有出现。如果产生这种现象，则分类器质量将大幅降低。为了解决这个问题，本文引入 Laplace 校准，其思想是将所有类别中各属性出现的次数加 1，这样如果训练样本集合充分大，并不会对结果产生影响，而且解决了上述频率为 0 的问题，优化了分类器的质量。

5.3.2 算法描述

基于上述分析，本文将 Android 系统中软件恶意行为的识别问题建模成对软件敏感权限使用进行分类的问题。本章介绍基于朴素贝叶斯分类器的恶意行为智能识别算法，首先给出恶意行为智能识别算法中的相关符号和定义。

定义 1 $X=\{x_1,x_2,x_3,x_4,x_5\}$，表示用于分类的特征属性值。

定义 2 $Y=\{y_{malicious},y_{benign}\}$，表示所有可能出现的分类结果。

恶意行为智能识别的算法的执行过程如图 5-2 所示。

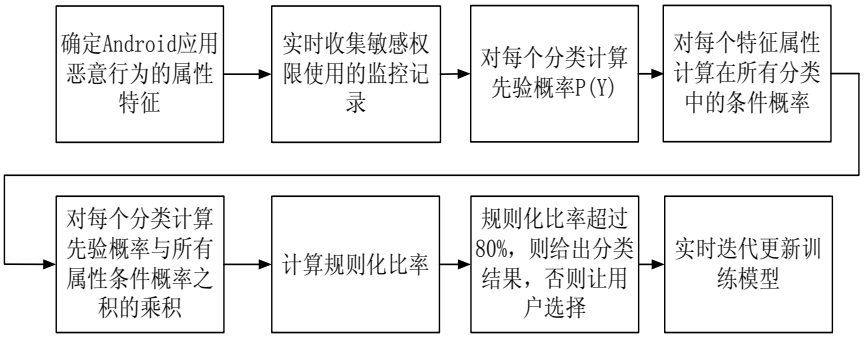

图 5- 2 恶意行为智能识别的算法流程

首先是准备阶段。这个阶段的任务是为基于朴素贝叶斯分类器的软件敏感权限使用的分类做必要的准备，主要工作是确定敏感权限使用行为的特征属性 x，包含确定敏感权限的使用者是否是系统应用、权限使用时是否有用户操作、敏感权限的使用者是否在安装时申请了大量权限、敏感权限的使用者是否在安装时存在恶意权限组合、敏感权限的使用是否存在突发性。收集一系列敏感权限的使用记录，并对每条记录人为地标记分类结果，形成训练样本集合。

然后是分类器训练阶段。这个阶段的任务为生成朴素贝叶斯分类器，即计算出目标函数 *f(x)*。主要工作是计算出贝叶斯公式中条件概率与先验概率的乘积，即计算训练样本中敏感权限被使用时每个特征属性 X_i 在各个结果分类中出现的频率与每个类别划分 Y_i 出现频率的乘积，对出现频率为 0 的安全信任级别分类进行 Laplace 校准，最后将结果记录。其输入是特征属性和训练样本，输出为目标函数 *f(x)*。

接着是应用阶段。这个阶段的任务是使用分类器对待分类项进行分类。实时监控软件使用敏感权限的行为，将行为输入分类器，分类器根据训练得到的目标函数 *f(x)*计算出贝叶斯公式中后验概率的分子部分（分母部分对所有特征属性 *X* 都一致，即可约分掉）。至此，朴素贝叶斯决策方法通过比较各个分类的计算结果，选取其中值最大的类别作为最终分类结果 Y_{result}。本文对贝叶斯模型的基本原理进行了分析，假设各特征属性的均值呈现正态分布，贝叶斯方法决策两分类的问题模型如图 5-3 所示。

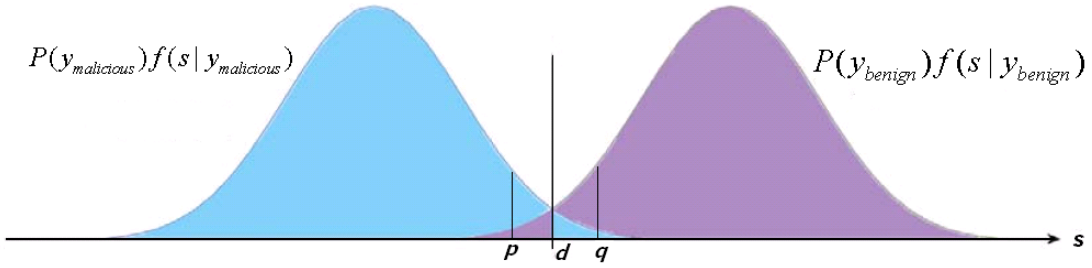

图 5- 3 贝叶斯方法决策两分类问题的模型

其中 s 是分类器特征属性的抽象表示，p 表示恶意行为的标准差，q 表示非恶意行为的标准差。当 $\prod_{i=1}^{5} P(y_{malicious}) f(s_i | y_{malicious}) \le d$ 的时候，贝叶斯方法的决策结果是认为本次敏感权限使用是恶意的；而当 $\prod_{i=1}^{5} P(y_{benign}) f(s_i | y_{benign}) > d$ 的时候，贝叶斯方法的分类结果是认为本次敏感权限使用是非恶意的。但实质上，使用这样的分类方法，当 $p < s < q$ 的时候，该方法的分类结果是错误的，其出错的概率为

$$P_e = P(y_{malicious})\int_{-\infty}^{d} f(s | y_{malicious})ds + P(y_{benign})\int_{d}^{+\infty} f(s | y_{benign})ds \qquad (5\text{-}6)$$

这些分类结果就导致了错误率的原因所在。因此，本文引入了比率规格化参数 φ 来量化分类结果的可靠程度，其计算方法如公式 5-8 所示。当 φ 小于 0.8 的时候，则认为分类结果无效，让用户自行判断本次的敏感权限使用是否是恶意的。

$$\varphi = \begin{cases} \dfrac{P_{benign} - P_{malicious}}{P_{benign}}, P_{benign} \ge P_{malicious} \\ \dfrac{P_{malicious} - P_{benign}}{P_{malicious}}, P_{benign} < P_{malicious} \end{cases} \qquad (5\text{-}7)$$

最后是迭代更新阶段。这个阶段的任务是实时迭代更新训练模型。本文初始时给的模式是基于 8849 条敏感权限使用记录所生成的，但当有新的敏感权限使用记录加入以后，需要对初始模型进行动态更新，从而能增加训练样本的规模，提高算法分类的准确度。

基于上述分析，可构建基于朴素贝叶斯分类器的 Android 软件恶意行为智能识别算法，算法的伪代码如下所示：

算法 5-1 基于朴素贝叶斯的 Android 软件恶意行为智能识别算法

Input: Permission Invoking
Output: Pass or Intercept
1　*Monitor Attribute List*
2　*Collect Examples*
3　*Behaviors←Entities in Examples*
4　Compute $P(y_j)$ and $P(x_k|y_j)$
5　**Foreach** y_j in Y
6　　APP_j*←subset of X has attribute list* $\mathbf{x_j}$;
7　　$P(\mathbf{y_i})$ *←|*APP_i *|/| Examples |*
8　　*Compute* $P(x_k|y_j)$
9　**end foreach**
10　**while** Applying for Permission **do**
11　　*Classify* APP_i
12　　Compute φ
13　　Make decision
14　**return** v_{re};

5.4 实验分析

为了验证本文实现基于朴素贝叶斯分类器的 Android 软件恶意行为智能识别算法，文本在 HTC G7 设备上安装了 83 个软件（其中恶意软件 49 个、非恶意软件 34 个），运行了 40 天，对 10 个敏感权限的使用进行了详细的记录和分析，共采集了 8849 条敏感权限的使用记录作为训练样本，构建基础模型，并采用有监督的学习方式，手动标记这些记录的类别。算法结果在验证之前，首先指定如下符号意义：

$N_{m\to m}$ 为软件恶意行为分类正确的数量

$N_{b\to b}$ 为软件非恶意行为分类正确的数量

$N_{m\to b}$ 为软件恶意行为被分类为非恶意的数量

$N_{b\to m}$ 为软件非恶意行为被分类为恶意的数量

N 为数据集中的样本数量

给出如下定义：

定义 1 正确率(correct rate，记为 P_D)表示数据集中归类正确的概率，即

$$P_D = \frac{N_{m\to m} + N_{b\to b}}{N}$$

定义 2 错误率(error rate，记为 P_E)表示数据集中归类错误的概率，即

$$P_E = \frac{N_{m\to b} + N_{b\to m}}{N}$$

定义 3 命中率(true positive rate，记为 *TPR*)表示 Android 系统中软件的恶意行为被正确归类的概率，即

$$TPR = \frac{N_{m\to m}}{N_{m\to m} + N_{m\to b}}$$

定义 4 误报率(false positive rate，记为 *FPR*)表示 Android 系统中软件的非恶意行为被归类到恶意行为的概率，即

$$FPR = \frac{N_{b\to m}}{N_{b\to b} + N_{b\to m}}$$

对于本文中的样本空间中 8849 条敏感权限的使用记录，本文采用分层 10 折交叉方法进行验证，即取样本中 10%的样本作为测试集，其余样本作为训练集，迭代 10 轮，每轮随机使用不同测试集计算分类器的 P_D 、 P_E 、TPR、FPR。验证结果如图 5-4 所示。

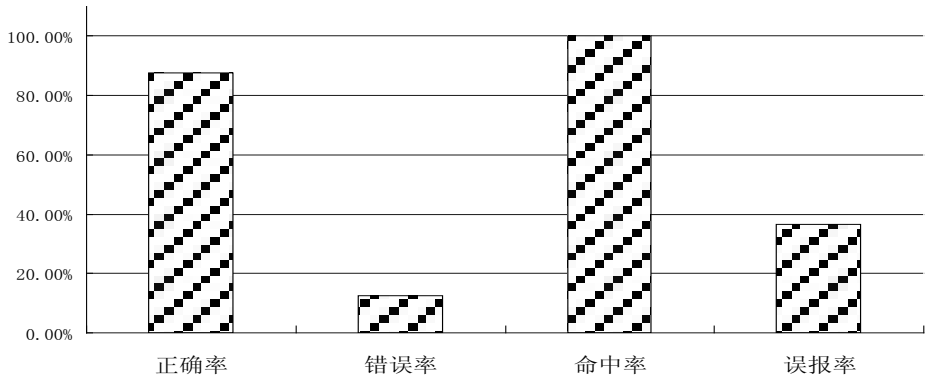

图 5- 4 基于标准朴素贝叶斯模型的恶意行为智能识别的正确率

通过实验结果可以看出，本文实现的基于朴素贝叶斯分类器的 Android 软件恶意行为的分类器的分类正确率是 87.6%、错误率是 12.4%、命中率是 100%、误报率是 36.7%。可见，该初始化的分类器有着极高的命中率和具有较高的正确率以及较低的错误率，但是该分类器的误报率比较高，所以根据 5.2.1 节第三阶段提出的方法，本文将规格化比率这个参数引入分类器的判别，改进现有的朴素贝叶斯算法。当规格化比率小于 80%的时候，分类器不作出决策，而是向用户界面弹出窗口。

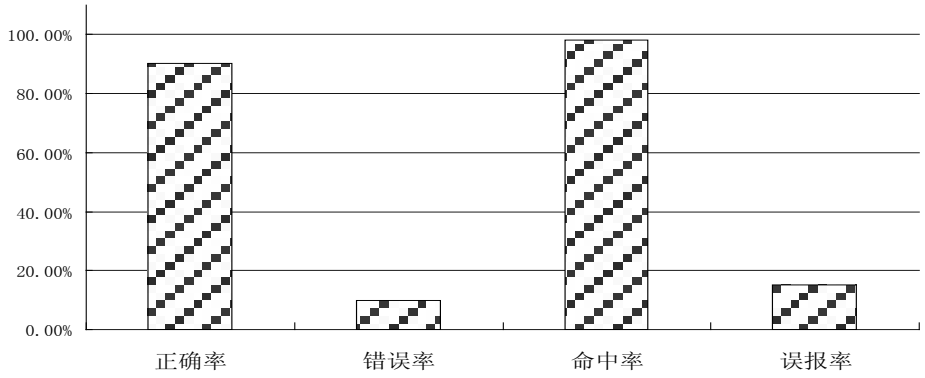

图 5- 5 基于改进的朴素贝叶斯模型的恶意行为智能识别的正确率

假设用户对 Android 软件恶意行为的反馈正确率为 75%，实验结果如图 5-5 所示，改进后的算法正确率为 90.1%、错误率是 9.9%、命中率是 98.2%、误报率是 15.3%，。由于大幅度减少了误报率，这保障了非恶意软件的正常使用。

5.5 本章小结

本章首先描述 Android 系统环境中软件恶意行为识别的问题，接着用朴素贝叶斯模型将该问题建模为分类问题进行求解。通过分析实际场景与实际数据集中所呈现的特性，抽象出了 5 个用于分类的特征属性，然后分析使用贝叶斯模型的理论合理性。在分析 8849 条手动标注分类结果的记录后，发现直接使用朴素贝叶斯分类器的分类结果会导致较高的误报率，然后本章通过引入规则化比率这个参数来改进贝叶斯分类器的决策方式，最后通过验证可以看出，本章改进后的基于朴素贝叶斯的分类器能够成功地在较低误报率的情况下识别 Android 系统中软件的恶意行为。

第六章　Android 安全防护软件的实现与测试

本章首先分别介绍 Android 安全防护软件的基础数据层，系统功能层和人机交互层的实现，最后对系统进行测试与小结。

6.1 系统实现

本文在研究 Android 系统中软件恶意行为的智能分析与处理方案的基础上，实现了一款 Android 安全防护软件。本章基于第二章所设计的体系结构，结合第三、四和五章所提出的静态分析、动态监控和智能识别方法，实现 Android 安全防护软件，包括基础数据层、系统功能层和人机交互层相关模块的具体实现。

6.1.1　基础数据层的实现

基础数据层包括数据抽取模块和特征抽象模块。基础数据层负责采集并确定智能算法抽象出了 5 个特征属性，为系统功能层的恶意行为智能识别模块提供数据。

（1）敏感权限的使用者是否是系统应用

当敏感权限的使用被监控到以后，使用者的uid将会通过进程间通信上传到软件层。Android系统的SDK中提供了PackageInfo和ApplicationInfo类，通过软件的uid可以使用PackageInfo类查询到该软件的PackageName。通过PackageName可以获取到ApplicationInfo对象，接着通过该对象的属性值FLAG_SYSTEM即可查到敏感权限的使用者是否是系统预装的软件。

（2）敏感权限使用时是否有用户操作

本文实现的Android安全防护软件希望能实时对用户操作进行记录，包括用户滑动屏幕的时间，点击智能终端上按键的时间等一系列用户对智能终端操作的时间，从而能够判断在软件使用敏感权限的时候，是否有用户在操作智能终端。通过调研发现，在Android系统的软件层与应用程序框架层并没有提供获取系统全局操作的开发接口，软件只有在本应用组件被激活时，才能获取到所有的用户操作。但当软件不是处于激活状态时，则不再能捕获用户的操作。因此，本文在Android系统的Linux内核层来实现捕获用户全局操作这个特征属性。

首先本文将智能终端是否有用户操作的判断转换为智能终端的硬件输入设备（触摸屏、键盘、轨迹球等）是否被操作。如果想要获知智能终端上的硬件输入设备是否被操作，则必须从硬件输入设备的驱动入手，一层层向上对Android

输入事件处理的运行流程有详细而深入的分析。

不同智能终端中，硬件输入设备各不相同，例如有的设备没有触摸屏、有的设备没有轨迹球。而即使都有触摸屏，触摸屏一般又有电阻式与电容式的区别，有些较大的显示屏甚至采用的是光学触摸屏。虽然硬件设备千差万别，但是这种差异由Linux kernel层的驱动部分来处理，而在Linux kernel层之上，不同型号的设备，事件处理方式都是一样的，其处理方式如图6-1所示。

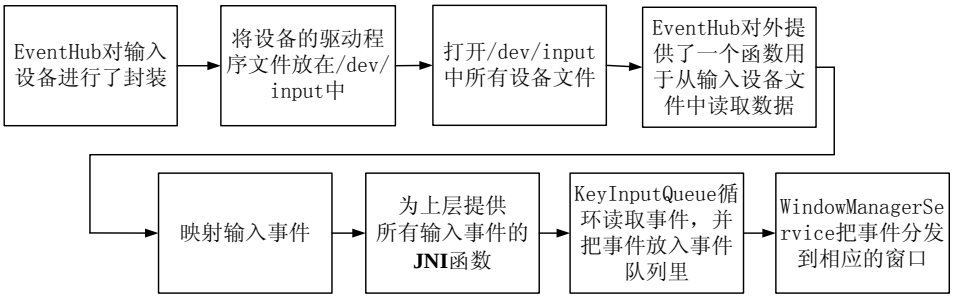

图 6- 1 Android 输入事件处理机制

EventHub是事件的抽象结构，维护着系统输入设备的运行情况。系统在启动的时候，EventHub会通过open_device方法将系统支持的输入设备都增加到这个抽象结构中，并维护一个所有输入设备的文件描述符，另外getEvent方法是对EventHub中的设备文件描述符使用poll操作将驱动层事件写入/dev/input中对应的设备文件中，然后调用map函数将驱动层事件映射为相应的输入值并通过提供JNI函数的方式将输入事件放入队列KeyInputQueue中，最后WindowManagerService负责将事件分发到相应的窗口中。

通过上述分析，本文发现当硬件输入设备被使用的时候，Linux Kernel会通过输入设备的驱动程序将操作更新入/dev/input/目录下面对应的文件中。也就是说，只要循环读取目录下的文件，当有文件更新时，输出更新记录并加上时间戳，即可判断当前时间点上是否有用户操作，监听用户输入操作的程序流程如图6-2所示。

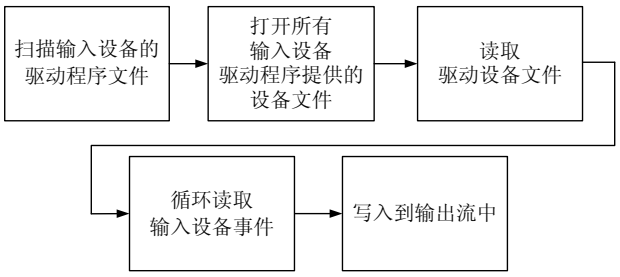

图 6- 2 监听用户输入操作的程序流程

监听用户输入操作的 C 语言核心源代码如图 6-3 所示，其 makefile 文件如图 6-4 所示，然后通过 Android NDK[44]交叉编译，即可生成 Android 系统下的可执行文件。

```
while(1) {
        pollres = poll(ufds, nfds, -1);
        if(ufds[0].revents & POLLIN) {
            read_notify(device_path, ufds[0].fd, print_flags);}
        for(i = 1; i < nfds; i++) {
            if(ufds[i].revents) {
                if(ufds[i].revents & POLLIN) {
                    res = read(ufds[i].fd, &event, sizeof(event));
                    if(res < (int)sizeof(event)) {
                        fprintf(stderr, "could not get event\n");
                        return 1;}
                    if(get_time) {
                        printf("%ld-%ld: ", event.time.tv_sec, event.time.tv_usec);}
                    if(print_device)
                        printf("%s: ", device_names[i]);
                    printf("%04x %04x %08x", event.type, event.code, event.value);
                    if(sync_rate && event.type == 0 && event.code == 0) {
                        int64_t now= event.time.tv_sec * 1000000L + event.time.tv_usec;
                        if(last_sync_time)
                            printf(" rate %lld", 1000000L / (now - last_sync_time));
                        last_sync_time = now;
                    }
                    printf("%s", newline);
                }
            }
        }
    }
```

图 6- 3 监听用户输入操作的 C 语言核心源代码

```
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)
LOCAL_SRC_FILES := getInputEvent.c
LOCAL_MODULE := getInputEvent
include $( BUILD_EXECUTABLE)
```

图 6- 4 监听用户输入操作的 makefile 文件

仅获得有用户操作的时间戳是不够的，文本需要获取到敏感权限使用时是否有用户操作。由于用户输入监听进程是将操作数据写入数据流中，所以本文设计两个文件用以交替式写入操作数据。如图6-5所示，每当验证敏感权限使用是否合理的时候，用户输入操作监听进程停止当前输出流文件的记录，切换到另一输出流文件用以记录用户新的输入操作。然后打开静止状态下输出流文件，跳转到文件最后一行，该行显示的长整型数据（格林威治时间1970年1月1日0时0分0秒

至今所经过的毫秒数）即为最后一次用户操作的时间戳，然后用当前时间与该时间戳相减，如果差小于1000（即1秒），则认为有本次敏感权限使用时是存在用户操作，即软件并非在偷偷地使用敏感权限。

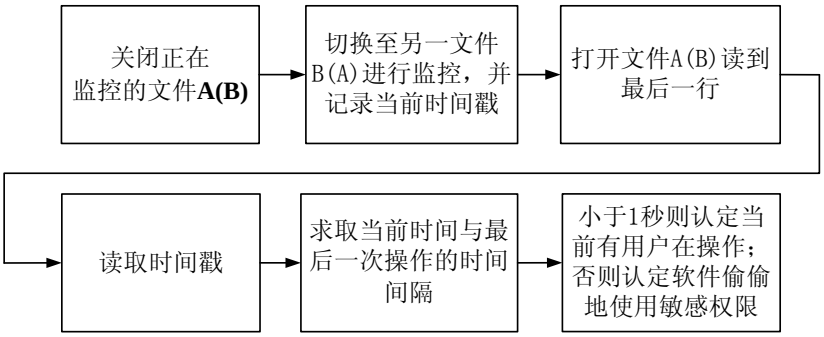

图 6- 5 实际监控流程

本文将数据流切换功能的C语言程序以JNI形式用NDK编译成一个动态链接库，供上层软件调用，其具体代码如图6-6所示，Makefile文件如图6-7所示，加载C语言链接库的过程如图6-8所示。

```
JNIEXPORT jint JNICALL Java_com_seu_cose_CatchService_runCommand
(JNIEnv *env, jclass class, jstring command)
    {
        const char *commandString;
        commandString = (*env)->GetStringUTFChars(env, command, 0);
        int exitcode = 0;
        if(strcmp(commandString, "a") == 0)
        {
            exitcode = 1;
            system("getInputEvent > /sdcard/Download/event_a.txt");
        }
        else if(strcmp(commandString, "b") == 0)
        {
            exitcode = 2;
            system("getInputEvent > /sdcard/Download/event_b.txt");
        }
        (*env)->ReleaseStringUTFChars(env, command, commandString);
        return (jint)exitcode;
    }
```

图 6- 6 实际监控流程的 C 语言代码

```
LOCAL_PATH:= $(call my-dir)
```

```
include $(CLEAR_VARS)
LOCAL_SRC_FILES := CatchService.c
LOCAL_MODULE := CatchService
include $(BUILD_SHARED_LIBRARY)
```

图 6- 7 实际监控流程的 makefile 文件

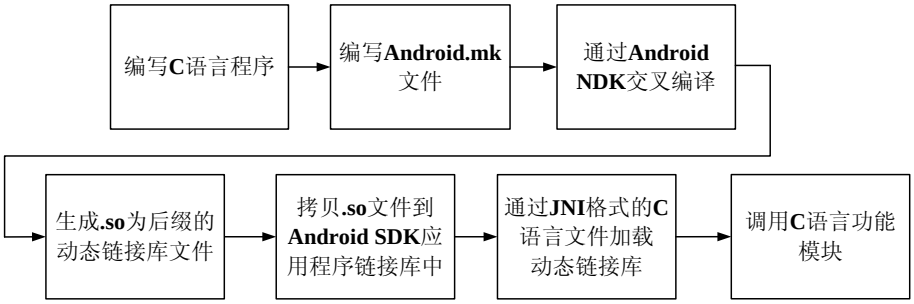

图 6- 8 加载 C 语言链接库的过程

（3）敏感权限的使用者是否在安装时申请了大量权限

该特征属性是一个静态属性，当软件被安装以后，所拥有的权限即被确定，并保存在该软件反射出的PackageInfo对象的requestedPermission字符串数组中。当该软件第一次使用敏感权限时，本文通过第四章所述的Android系统敏感权限的动态实时监控机制能获取到<packageName, permission>这个二元组，通过敏感权限使用者的packageName获取到PackageInfo类，接着读出requestedPermission这个字符串数组的长度即为该软件申请的权限数。将该数值与3.2节所求得的热门软件的平均权限申请数作对比，超过150%平均值，则认为该软件在安装时申请了大量权限。

（4）敏感权限的使用者是否在安装时存在恶意权限组合

该特征属性是在软件被安装的时候被提取。在软件安装的时候，通过PackageManager和PackageParser接口得到被安装应用程序申请的权限信息，然后匹配当前权限列表与3.3节预先挖掘出的敏感权限组合。如果当前权限列表违反了由敏感权限组合制定出的安全规则，则判定当前敏感权限的使用者在安装时存在敏感权限组合。

（5）敏感权限的使用是否存在突发性

当敏感权限的使用被监控到并获取到<packageName, permission>这个二元组以后，进入 lastApply 这个 arrayList 中查询上一次该软件使用权限的时间。lastApply 的数据结构如图 6-9 所示，当没有查询到该队列中的存在现有 uid 的软件名后，则新建一个节点；否则比较当前时间与上一次权限使用的时间，差在 3 秒钟之内则认为本次权限的使用存在突发性。

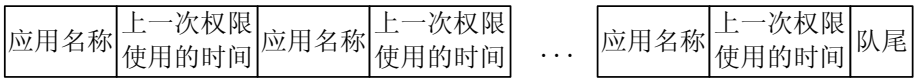

图 6- 9 lastApply 的数据结构

（6）特征属性的组合与存储

当敏感权限的使用被监控以后，通过上述 5 种方法分别获取到智能识别的 5 种特征属性，然后使用智能算法来识别本次对敏感权限的使用是否合理。以上数据将会被保存在 Android 自带 Sqlite 数据库的 PermissionUsageHistory 表中，其表结构定义如表 6-1 所示。

表 6- 1 PermissionUsageHistory 表结构定义

名称	字段名	数据类型	长度
软件包名	PackageName	Vchar	50
是否是系统软件	SystemSoftware	Bit	X
是否有用户操作	UserOperation	Bit	X
是否有突发性	PermissionPeak	Bit	X
是否权限数大于均值	AveragePermission	Bit	X
是否存在敏感权限组合	MaliciousCombination	Bit	X
识别结果	Result	Int	X

6.1.2　系统功能层的实现

系统功能层的具体实现分为两个阶段：第一阶段实现由安装判别模块解析出准备安装的 Android 软件的权限信息，然后通过进程间通信模块发送给规则匹配模块，规则匹配模块匹配当前权限列表和 Apriori 算法挖掘出来的软件敏感权限组合规则集，然后将匹配结果返回给安装判别模块，由该模块给予用户安全提示，并且提供用户智能安装功能；第二阶段实现是由 API 拦截模块实时捕获系统中软件的敏感操作，通过进程间调用模块将二元组<uid, permission>发送至动态权限分发处理模块，然后该模块根据用户对软件预先设置好的权限模式做相应处理。如果用户选择智能分析，则调用恶意行为智能识别模块获取对当前软件行为的识别结果，然后再通过进程间通信模块将结果回传给动态权限反馈模块，最后由该模块根据反馈结果，对当前软件进程的敏感操作执行决策。

（1）安装判别模块

API 拦截模块位于 Android 系统应用程序框架层，其功能是抽取准备安装软件所申请的权限，接着将权限列表发送给应用程序进程的规则匹配模块，然后转发规则匹配模块的返回结果，提供给人机交互层，从而在软件安装的时候给予用户安全提示。其具体实现步骤如下：

步骤 1 读取准备安装的软件的 manifest.xml 文件；

步骤 2 解析 xml 文件中的<uses-permission>字段；

步骤 3 提取<uses-permission>字段中的 android.name 所对应的字符串；

步骤 4 创建 ArrayList 对象用于存放软件所申请的权限；

步骤 5 Intent intent=new Intent("android.os.StaticMatch")意图对象；

步骤 6 创建 StaticMatch.aidl 文件放入 frameworks/base/core/java/android/os 目录；

步骤 7 编写接口函数 staticMatch (ArrayList permissionApplied)；

步骤 8 新建一个 ServiceConnection 对象；

步骤 9 重写 onServiceConnected 方法；

步骤 10 通过 bindService 方法对服务进行绑定，其方法为 bindService(intent, serviceConnection, Context.BIND_AUTO_CREATE);

步骤 11 通过反射出的 aidl 类的对象，以远程调用的参数形式传递准备安装软件的权限列表；

步骤 12 发送规则匹配模块的反馈结果用户界面。

（2）规则匹配模块

规则匹配模块位于 Android 应用程序层，其核心功能是匹配当前权限列表和 Apriori 算法挖掘出来的敏感权限组合，然后将匹配结果返回给安装判别模块，其实现步骤如下：

步骤 1 拷贝 frameworks/base/core/java/android/os 目录下的 StaticMatch.aidl 文件到 Android 应用程序工程目录下；

步骤 2 等待 ADT 自动生成 aidl 的 Java 接口文件；

步骤 3 创建服务类继承 Service 类；

步骤 4 在该服务类中编写 StaticMatch.Stub 内部类；

步骤 5 重写 aidl 的 onBind 方法，返回 aidl 类对象；

步骤 6 在内部类中实现 aidl 定义的接口函数 StaticMatch，根据敏感权限组合规则，逐条扫描接收到的 ArrayList 权限列表；

步骤 7 如果存在违反规则的权限组合，则返回名称；否则返回 null。

（3）API 拦截模块

API 拦截模块位于 Android 系统应用程序框架层，其核心功能是实时捕获系统中软件的敏感操作并发送给动态权限分发模块，然后提示数据抽取模块采集当前时刻的系统相关信息。当软件调用系统权限操作时，该模块被触发。其实现步骤如下：

步骤 1 获取被调用的权限和软件 uid；

步骤 2 筛选出敏感权限调用；

步骤 3 通过 uid 查找当前调用敏感权限的包名；

步骤 4 创建 Intent intent=new Intent("android.os.PermissionQuery")意图对象；

步骤 5 创建 PermissionQuery.aidl 文件放入 frameworks/base/core/java/android/os 目录下；

步骤 6 编写接口函数 permissionQuery(String packageName, String permission)；

步骤 7 新建一个 ServiceConnection 对象；

步骤 8 重写 onServiceConnected 方法；

步骤 9 通过 bindService 方法对服务进行绑定，其方法为 bindService(intent, serviceConnection, Context.BIND_AUTO_CREATE);

步骤 10 通过反射出的 aidl 类的对象，以远程调用的参数形式传递二元组。

（4）动态权限分发处理模块

动态权限分发处理模块位于 Android 应用程序层，其核心功能是分发接收到的敏感操作到对应的处理类中。其实现步骤如下：

步骤 1 拷贝 frameworks/base/core/java/android/os 目录下的 PermissionQuery.aidl 文件到 Android 应用程序工程目录下；

步骤 2 等待 ADT 自动生成 aidl 的 Java 接口文件；

步骤 3 创建服务类继承 Service 类；

步骤 4 在该服务类中编写内部类 aidl 的 PermissionQuery.Stub 类；

步骤 5 重写 aidl 的 onBind 方法，返回 aidl 类对象；

步骤 6 在内部类中实现 aidl 定义的接口函数 permissionQuery，根据接收到的二元组<packageName, permission>按权限类别分发到对应的处理类中；

步骤 7 查看用户对该包名的软件使用该权限的设置；

步骤 8 如果是 ALWAYS_ALLOW, ALWAYS_DENIAL, ASK_USER 这三种用户设置模式，则直接返回相对应的代码给动态权限反馈模块；

步骤 9 如果用户设置的模式是 ARTIFICIAL_INTELLIGENCE，则调用恶意行为智能识别模块获取反馈代码。

（5）恶意行为智能识别模块

恶意行为智能识别模块位于 Android 应用程序层，其核心功能是根据基础数据层的特征抽取模块提供的数据来建立朴素贝叶斯模型，从而判断当前软件行为是否合理，然后将结果反馈给动态权限分发处理模块，并更新朴素贝叶斯模型，然后将数据保存到 Sqlite 数据库中。

步骤 1 如果模型尚未构建，则从 Sqlite 数据库中读取历史数据；否则从内存中读出模型数据；

步骤 2 计算数据中恶意程序和非恶意程序的先验概率；

步骤 3 计算各属性分类的条件概率；

步骤 4 计算上述步骤的两者乘积，并计算规格化比率；

步骤 5 如果规格化比率超过 0.8，则判定乘积大的类别为当前软件行为的类别；

步骤 6 否则反馈 ASK_USER 标签，通过用户选择来恶意行为智能识别模块的误报率；

步骤 7 调用 Android 系统内部的 Sqlite 数据库，将反馈结果存到 PermissionUsageHistory 表中对应的记录内；

步骤 8 更新并保存当前模型数据。

6.1.3 人机交互层的实现

人机交互层主要提供用户操作界面。在 Android 软件中，用户操作界面的逻辑部分通常都是由 Activity 的派生类实现。具体的界面布局是由 XML 文件定义生成的。Activity 的派生类在创建的时候，定义绑定的 XML 文件，从而让界面布局与界面逻辑结合在一起。

在本系统中，主要由以下人机交互界面组成：Android 安全防护软件主界面、通信权限管理界面、隐私权限管理界面、具体权限设置界面、弹窗选择界面、更新过的程序安装界面。界面之间的跳转如图 6-10 所示。

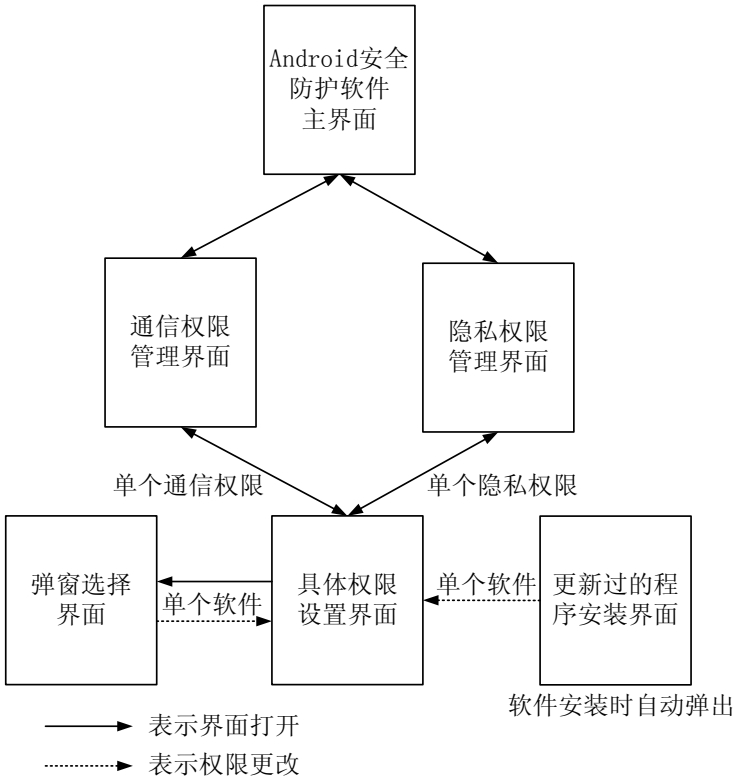

图 6- 10 软件界面跳转流程

6.2 系统测试

本节中将描述对本文实现的 Android 安全防护软件进行功能测试。首先描述测试环境，然后分别对该系统的各个主要功能，以使用软件发短信权限为例，逐个进行测试。

6.2.1　测试环境

硬件信息：HTC G7(CPU：ARMv7 Processor rev 2 (v71) 1024MHz，内存 512M，外部存储 8G)

系统版本：Cyanogenmod-7 (Android 2.3.7)

内核版本：2.6.37.6-cyanogenmod

6.2.2　功能测试

功能测试的任务是验证系统功能是否符合系统的设计。Android 安全防护软件主要测试通信权限管理、显示拥有特定权限的软件、设置软件拥有的权限、拦截软件使用敏感权限、敏感权限管理、智能安装功能等，分别验证每个功能的正确性、完整性及界面与设计的一致性。每一功能测试后，通过查看 Android SDK 提供的 Dalvik 虚拟机调试监控服务 DDMS（Dalvik Debug Monitor Service）所记录的日志信息进一步确认功能执行过程中是否产生了错误。

（1）主界面

图 6- 11 Android 安全防护软件的主界面

Android 安全防护软件的主界面如图 6-11 所示，主要有两个按钮，分别对应通信权限的管理和隐私权限的管理。

（2）通信权限管理

图 6- 12 通信权限管理界面

在通信权限管理界面上，主要有三个按钮，如图 6-12 所示，点选可以分别进入管理拨打电话、发送短信以及连接网络这三种通信权限的软件。点击通信权限管理界面上的发送短信权限管理后，进入权限管理页面。

图 6- 13 拥有短信发送权限的软件

图 6-13 罗列了 Android 系统中一部分拥有短信发送权限的软件，从图中可以看出，系统自带的拨号器等绿色打钩标记的软件被设置成总是允许使用短信发送的权限；设置存储等红色打叉的软件被设置成总是不允许使用短信发送的权限；ADW 主界面等带有问号的软件被设置成每当该软件使用短信发送的权限时，需要询问用户，让用户自行判断当前是否给予软件权限的使用，默认时间是 5 秒钟，不选择则自动选择拒绝本次权限的使用；信息等软件被设置成 AI 智能判断，当软件被设置成这种模式，则系统会根据软件在 Dalvik 虚拟机中运行的情况，自行给出判断。

（3）权限动态设置与实时监控

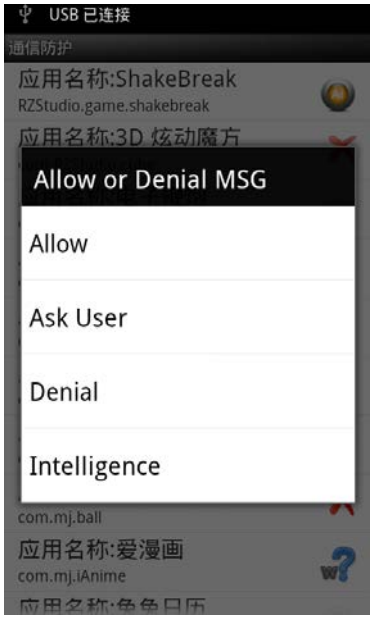

图 6- 14 模式选择对话框

长按其中某个软件，系统会弹出模式选择对话框，如图 6-14，以供用户选择对应软件的权限管理模式。

图 6- 15 允许发送短信的测试

本节使用 Android 系统自带的短信软件做测试，当该软件被设置成总是允许使用发送短信权限的时候，图 6-15 显示用户发送短信至 10086，能够正常收到 10086 的自动回复，证明原短信的确被发送出去了。

图 6- 16 拦截短信发送的测试

当该软件被设置成总是不允许使用发送短信权限的时候，图 6-16 显示用户发送短信至 10086，该条短信旁边出现了表示短信未发送的红色小三角，并且没有收到 10086 的自动回复，证明原短信的确没有被发送出去。

图 6- 17 用户询问模式的测试

当该软件被设置成用户询问模式时，图 6-17 显示用户发送短信至 10086 时的情境，本文设计实现的软件系统会弹出一个全局响应的对话框，让用户选择是否发送短信，当用户点击“Yes”的时候，本系统会通过本次权限使用的请求，授权软件发送短信，否则拦截本次发送短信的权限使用请求。

（4）软件恶意行为的智能识别

图 6- 18 智能识别模式的测试

当该软件被设置成 AI 智能判断模式时，为了验证 AI 智能判断的可靠性，本文另外设计了一个在后台每隔 1 分钟向 10086 发一次短信的测试程序 ZSendMSG，图 6-18 显示当系统短信应用与 ZSendMSG 都被设置成 AI 智能判断模式的时候，通过系统短信应用发送的短信能够收到 10086 的自动回复，而通过 ZSendMSG 发送短信的时候，不能收到 10086 的自动回复（如果设置成总是允许的模式，是可以收到 10086 的回复的），这证明了系统短信应用能够将短信发出，而 ZSendMSG 不能获得发送短信的权限，所以不能将短信发出。

（3）隐私权限管理

隐私权限管理的执行逻辑与通信权限管理是一样的，只是两者操作的对象不同，在隐私权限管理界面上，可以管理隐私权限的使用，实现 Android 系统隐私权限的实时更改。

图 6- 19 隐私权限管理界面

隐私权限管理界面如图 6-19 所示，分别点选可以进入管理拥有终端设备数据访问权限（手机号和设备识别码）的软件、拥有联系人访问权限的软件、拥有拦截短信权限的软件、拥有短信内容读取权限的软件、拥有通话状态读取权限的软件、拥有位置信息读取权限的软件（读取粗略位置和精细位置这两个权限统一管理）。分别点选进入各个权限的管理界面以后，隐私权限管理布局以及操作方式与通信权限管理完全一致，相比较于其它安全软件复杂的操作，用户能够很容易上手本系统对各个敏感权限的管理操作。

（4）智能安装功能

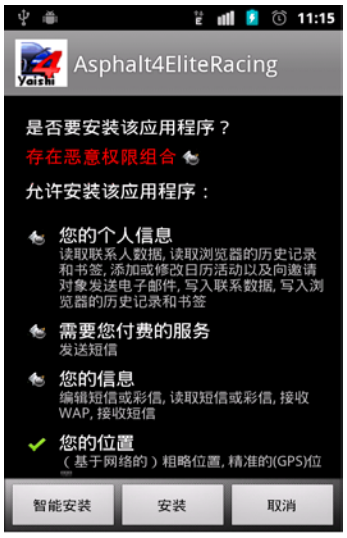

图 6- 20 软件安装时的安全提示

当用户安装软件的时候，智能安装功能能够识别出软件中所携带的恶意软件组合，并将敏感权限组合标识出来，从而给予用户安全提示，如图 6-20 所示。

图 6- 21 智能安装后的自动拦截设置

软件在安装的时候还提供智能安装按钮，当用户点击了智能安装，则该软件安装以后，违反安全规则的敏感权限组合在权限动态设置中，将被自动标注为拦截状态，如图 6-21 所示。

6.2.3　性能测试

Android 平台中响应时间超时 5 秒钟就会弹出应用程序无响应（Application Not Responding，记为 ANR）的消息，提示用户是否继续等待或者关闭。软件的性能测试要求各操作的响应时间不超过 5 秒，不能产生 ANR 异常。本文在 HTC G7 和 Android 模拟器上重复开启了 10 次求取平均启动时间、随机点击了 20 次按钮求取平均响应速度、使用 PerfMon V1.1 和 Memory Status 软件辅助检测包括 CPU 和内存在内的资源占有率，得到的性能测试结果如表 6-2 所示，其中(F)表示 Android 安全防护软件未启动时的状态；(T)表示 Android 安全防护软件已打开以后的状态。

表 6- 2 性能测试结果

性能参数 / 测试平台	软件平均启动时间	软件平均响应速度	稳定性	资源占有率	
				CPU(%)	内存(Mb)
HTC G7(F)	X	X	无 ANR	1~15	240.7
HTC G7(T)	430ms	1130ms	无 ANR	1~58	249.4
Android 模拟器(F)	X	X	无 ANR	20~73	68.2
Android 模拟器(T)	1270ms	2280ms	无 ANR	20~99	75.4

6.3 本章小结

本章分别阐述了 Android 安全防护软件基础数据层，系统功能层和人机交互层的实现。在系统数据层，分别介绍了 5 种用于软件恶意行为识别的软件特征属性的捕获方法。在系统功能层，给出各功能模块的具体实现流程。在人机交互层，给出了软件界面之间的跳转流程。最后对系统各项功能进行测试，验证了系统的可用性。

第七章　总结与展望

本章对 Android 系统中软件恶意行为智能分析与处理的研究进行总结，在归纳论文研究工作的基础上，指出论文中实现的 Android 软件权限的静态分析、Android 系统敏感权限的动态实时监控技术以及基于朴素贝叶斯分类器的软件恶意行为智能识别算法的特点和不足，并对未来进一步研究和开发工作进行展望。

7.1 研究成果总结

论文的研究工作包括以下几个方面：

(1) 概述现有 Android 系统中的安全问题以及相关研究，对现有 Android 系统及其安全框架、Android 系统安全威胁以及 Android 恶意软件检测技术进行调研，指出存在的问题和安全隐患。

(2) 分析了 Android 系统的安全机制，对其中最为核心的权限管理机制进行研究。通过对收集到的 1260 个恶意软件和 Android 市场中 17 个分类中的 850 个热门软件所申请的权限进行静态分析，使用 Apriori 算法挖掘出恶意软件和敏感权限组合的关联，然后归纳出需要在软件运行时被监控的 10 种敏感权限。

(3) 在调研 Android 系统相关技术的基础上，分析了 Android 系统的源代码。其中，详细分析了 Android 系统中短信发送和权限审查操作的具体实现，归纳出 Android 系统对敏感权限操作的流程，并根据该流程设计了 Android 系统敏感权限的动态实时监控机制。

(4) 通过静态分析和动态监控，抽取出 5 个用于识别 Android 系统中软件恶意行为的抽象属性特征。基于采集到的 8849 条属性特征数据，构建朴素贝叶斯分类器，并根据实际场景，在改进朴素贝叶斯决策方法的基础上，设计了基于朴素贝叶斯分类器的 Android 软件恶意行为智能识别的算法，并验证其有着较高的正确度和较低的误报率。

(5) 整合 Android 软件恶意行为的智能分析与处理方案，实现一款 Android 安全防护软件，智能地保障 Android 系统用户的安全。

7.2 未来工作展望

Android 系统作为移动互联网应用终端的热门系统，逐年增高的市场占有率令 Android 程序开发者为之疯狂。层出不穷的 Android 应用对 Android 系统的安

全性提出了严峻的挑战。Android 系统的安全问题抑或可能成为 Android 系统进一步普及的瓶颈，是 Android 系统研究最为重要的问题之一，值得研究者的关注。本论文通过对 Android 安全机制中最为重要权限模型进行了研究，解决了 Android 软件权限在安装时没有安全提示、在安装以后不能更改的问题，并为安全意识不高的用户提供了智能识别软件恶意行为的解决方案。本文从权限管理的角度完善 Android 系统的安全机制，但是论文设计并实现的完善方案仍然存在一些不足。因此，未来的工作可以从以下几个方面开展：

(1) 在现有恶意软件数据集的基础上，分析抽象出其他用于识别软件恶意行为的属性特征，并收集更多恶意软件在 Dalvik 虚拟机中使用权限的记录，来完善本文所设计的 Android 安全软件中的初始模型。

(2) 在对软件权限静态抽取分析时，进一步深入考虑各个类别对于软件权限的影响；在使用关联规则挖掘的时候，考虑将软件类别单独抽象为一层，从而引入多层关键规则挖掘的思想，进一步提高恶意权限组合的识别准确度。

(3) 进一步分析 Android 系统，发现其它可用于实时拦截 Android 系统敏感权限使用的方法，以及考虑使用其它模式识别算法来分类软件的行为，如支持向量机或神经网络算法等。

参考文献

[1] Gartner November Report[EB/OL]. http://www.gartner.com/newsroom/id/2482816, 2013
[2] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, C. Glezer. Google Android: a comprehensive security assessment[J]. IEEE Security and Privacy Magazine, 2010, 8(2): 35-44
[3] K. Zhang, X. Zhou, M. Intwala, A. Kapadia, X. Wang. Soundcomber: a stealthy and context-aware sound Trojan for smartphones[C]. In Proceedings of the 18th Network & Distributed System Security Symposium (NDSS), San Diego, CA, February 6–9, 2011, 17–33
[4] 高速缓存压缩技术[EB/OL]. https://wiki.edubuntu.org/Compcache, 2013
[5] 空中下载技术[EB/OL]. http://en.wikipedia.org/wiki/Over-the-air_programming, 2013
[6] Dalvik Virtual Machine[EB/OL]. http://en.wikipedia.org/wiki/Dalvik, 2013
[7] A. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner. A survey of mobile malware in the wild[C]. In Proceedings of ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM), Chicago, Illinois, USA, October 17, 2011, 3-14
[8] NQMobileInc[EB/OL]. http://weishi.nq.com, 2013
[9] L. Davi, A. Dmitrienko, A. Sadeghi, and M. Winandy. Privilege escalation attacks on android[C]. In Proceedings of International Conference on Information Security (ISC), Boca Raton, Florida, October 25-28, 2010, 346-360
[10] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang. Accessory: keystroke inference using accelerometers on smartphones[C]. In Proceedings of The Thirteenth Workshop on Mobile Computing Systems and Applications (HotMobile). San Diego, California, USA, February 28-29, 2012, 73-81
[11] L. Cai and H. Chen. TouchLogger: Inferring keystrokes on touch screen from smartphone motion[C]. In Proceedings of the 6th USENIX Workshop on Hot Topics in Security (HotSec), San Francisco, California, USA, August 8–12, 2011, 37-42
[12] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp)iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers[C]. In Proceedings of ACM conference on Computer and Communications Security (CCS), Chicago, Illinois, USA, October 17–21, 2011, 551-562
[13] D. Asonovand and R. Agrawal. Keyboard acoustic emanations[C]. In Proceedings of IEEE Symposium on Security and Privacy (S&P), Oakland, California, USA, May 9-12, 2004, 3-11
[14] D. F. Kune and Y. Kim. Timing attacks on pin input devices[C]. In Proceedings of ACM conference on Computer and Communications Security (CCS), Chicago, Illinois, USA, October 4–8, 2010, 678-680
[15] L. Zhuang, F. Zhou, and J. D. Tygar. Keyboard acoustic emanations revisited. [C] In Proceedings of ACM conference on Computer and Communications Security (CCS), Alexandria, Virginia, USA, November 7–11, 2005, 373-382
[16] M. Zalewski. Cracking safes with thermal imaging[EB/OL]. http://lcamtuf.coredump.cx/tsafe/, 2012
[17] Y. Zhang, P. Xia, J. Luo, Z. Ling, B. Liu, and X. Fu. Fingerprint attack against touch-enabled devices[C]. In Proceedings of ACM workshop on Security and Privacy in

Smartphones and Mobile devices (SPSM), Raleigh, North Carolina, USA, October 19, 2012, 57-68
[18] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith. Smudge attacks on smartphone touch screens[C]. In Proceedings of Workshop on Offensive Technology (WOOT), Washington, DC, August 9, 2010, 63-72
[19] B. John. A binary analysis of resultatives[C]. In Proceedings of Texas Linguistics Society Conference (TLSC), Stanford, California, USA, 1997, 43-58
[20] D. Binkley, Source code analysis: A Road Map[C]. In Proceedings of Future of Software Engineering(FSE), Minneapolis, Minnesota, USA, May 23-25, 2007, 104-119
[21] W. Enck, M. Ongtang, and P. McDaniel. On lightweight mobile phone application certification[C]. In Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS), Chicago, Illinois, USA, November 9-13, 2009, 235-245
[22] D. Barrera, H. Kayacik, C. Paul, and A. Somayaji. A methodology for empirical analysis of permission-based security models and its application to android[C]. In Proceedings of the 17th ACM conference on Computer and Communications Security (CCS), Chicago, Illinois, USA, October 4-8, 2010, 73-84
[23] 赖英旭, 刘增辉. 基于关联规则的未知病毒检测方法研究[J], 计算机工程与应用, 2008, 44(7): 137-139
[24] Y. Hu and L. Chen. Unknown malicious executables detection based on run-time behavior[C]. In Proceedings of International Fuzzy Systems and Knowledge Discovery Conference (FSKD), Jinan, Shandong, RPC, October 18-20, 2008, 391-395
[25] W. Enck, and P. Gilbert. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones[C]. In Proceedings of USENIX Symposium on Operating Systems Design and Implementation (OSDI), Berkeley, California, USA, 2010, 24-38
[26] 刘泽衡. 基于 Android 智能手机的安全检测系统的研究与实现[D]. 哈尔滨工业大学硕士学位论文, 2011
[27] M. Miettinen, P. Halonen, and K. Hatonen. Host-based intrusion detection for advanced mobile devices[C]. In Proceedings of Advanced Information Networking and Applications (AINA), Vienna, Austria, April 18-20, 2006, 72-76
[28] A. Schmidt, H. Schmidt, J. Clausen, K. Ali, O. Kiraz, A. Camtepe and S. Albayrak. Enhancing security of Linux-based Android devices[C]. In Proceedings of the 15th International Linux System Technology Conference, Hamburg, Germany, October 7-10, 2008, 174-189
[29] 乜聚虎. 智能手机异常检测技术研究与实现[D]. 中国科学技术大学硕士学位论文, 2011
[30] K. Rieck, T. Holz, C. Willems, P. Dussel, and P. Laskov. Learning and classification of malware behavior[C]. In Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), July 10-11, Paris, France, 2008, 108-125
[31] A. Bose and K. Shin. Proactive security for mobile messaging networks[C]. In Proceedings of the 5th ACM workshop on Wireless security (WiSe), Los Angeles, California, USA, September 29, 2006, 95-104
[32] L. Xie, X. Zhang, J. Seifert, and S. Zhu. pBMDS: A behavior-based malware detection system for cellphone devices[C]. In Proceedings of the third ACM conference on Wireless network security (WiSec), Hoboken, New Jersey, USA, March 22–24, 2010, 37-48

[33] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, Y. Weiss. “Andromaly”: a behavioral malware detection framework for Android devices[J]. Journal of Intelligent Information Systems. 2012, 38(1): 161-190
[34] C. Wright, C. Cowan, S. Smalley, J. Morris, and G. Kroah-Hartman. Linux security modules: general security support for the Linux kernel[C]. In Proceedings of the 11th USENIX Security Symposium, San Francisco, California, USA, 2002, 213-226
[35] POSIX UID[EB/OL]. http://en.wikipedia.org/wiki/POSIX, 2013
[36] A. Schmidt; H. Schmidt; L. Batyuk; J. Clausen, S. Camtepe; S. Albayrak; and C. Yildizli. Smartphone malware evolution revisited: Android next target[C]. In Proceedings of the 4th Malicious and Unwanted Software (MALWARE), Montreal, Quebec, Canada, October 13-14, 2009, 1-7
[37] Android Market [EB/OL]. http://www.android.com/apps, 2013
[38] Y. Zhou, and X. Jiang. Dissecting Android malware: characterization and evolution[C]. In Proceedings of IEEE Symposium on Security and Privacy (S&P), San Francisco, California, USA, May 20-23, 2012, 95-109
[39] G. Chang, C. Tan, G. Li, and Chuan Zhu, Developing mobile applications on the Android platform[J]. Mobile Multimedia Processing: Fundamentals, Methods, and Applications, 2010, 4(1): 264-286
[40] Apriori Algorithm[EB/OL]. http://en.wikipedia.org/wiki/Apriori_algorithm, 2013
[41] M. Gargenta. Learning Android[M]. O’Reilly Media Press, 2011
[42] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky. Bayesian model averaging: a tutorial[J]. Statistical Science, 1999, 14(1): 382-417
[43] Naïve Bayes Classifier[EB/OL]. https://en.wikipedia.org/wiki/Naive_Bayes_classifier, 2013
[44] P. Publishing. Android NDK Beginner's Guide[M]. Packt Publishing Press, 2012

作者简历

张扬，男，1987 年 10 月生，上海人。2006 年，从上海市新中高级中学毕业，考入东南大学临床医学院。2007 年 9 月，在东南大学转系至软件学院。2011 年 9 月，保送东南大学计算机学院计算机应用技术专业攻读硕士学位至今，专业为计算机应用技术，研究方向为网络安全。在攻读硕士学位期间，参与了多项国家和部、省级科研项目，在国际学术会议上发表和录用学术论文 2 篇。

姓　名	张扬	性　别	男	出生年月	1987.10
E-mail	yangzhang@seu.edu.cn				
通信地址	东南大学九龙湖校区计算机楼 331 室，211189				
联系电话	13585129211				

教育背景

2011.9 至今	**东南大学**计算机科学与工程学院计算机应用技术专业**硕士**研究生 研究方向：网络安全
2007.9 至 2011.6	**东南大学**软件学院**本科**生
2006.9 至 2007.6	**东南大学**临床医学院**本科**生

研究经历

硕士生阶段

从事网络安全研究：

- 国家科技支撑计划课题，"区域医疗卫生海量信息处理、安全与隐私保护等关键技术研究"(2010BAI88B03)，2009 年 10 月-2014 年 09 月
- 国家自然科学基金青年基金项目，"基于主动流量分析的匿名通信追踪技术研究"(60903162)，2010 年 01 月-2012 年 12 月
- 计算机网络和信息集成教育部重点实验室开放课题，"匿名 Web 流量检测关键技术研究"(K939201028)，2011 年 01 月-2012 年 12 月
- 国家自然科学基金面上项目，"基于侧信道攻击的匿名通信流量识别、分析和追踪技术研究"(61272054)，2013 年 01 月-2016 年 12 月

获得奖励

2012 年 Google 荣誉学生奖学金

2012 年远景能源奖学金

2011 年东南大学计算机科学与工程学院研究生会优秀部长称号

攻读硕士学位期间发表的论文

1. Yang Zhang, Peng Xia, Junzhou Luo, Zhen Ling，Benyuan Liu，Xinwen Fu: Fingerprint Attack against Touch-enabled Devices[C], in Proceedings of Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM), Raleigh, North Carolina, USA, October 19, 2012, 57-68 (EI)

2. Zhen Ling, Junzhou Luo, Yang Zhang, Ming Yang, Xinwen Fu, Wei Yu: A novel network delay based side-channel attack: Modeling and defense[C], in Proceeding of the 31rd Annual IEEE International Conference on Computer Communications (INFOCOM), Orlando, Florida, USA, March 25-30, 2012, 2390-2398 (EI)

攻读硕士学位期间参与的科研项目

1、国家科技支撑计划课题，"区域医疗卫生海量信息处理、安全与隐私保护等关键技术研究"(2010BAI88B03)，2009 年 10 月-2014 年 09 月，主要成员，从事基于 WebSphere 和 ICE 中间件的多源异构网络数据的动态抽取与融合。

2、国家自然科学基金青年基金项目，"基于主动流量分析的匿名通信追踪技术研究"(60903162)，2010 年 01 月-2012 年 12 月，主要成员，从事 web 站点的指纹攻击。

3、计算机网络和信息集成教育部重点实验室开放课题，"匿名 Web 流量检测关键技术研究"(K939201028)，2011 年 01 月-2012 年 12 月，主要成员，从事 web 站点的指纹攻击。

4、国家自然科学基金面上项目，"基于侧信道攻击的匿名通信流量识别、分析和追踪技术研究"(61272054)，2013 年 01 月-2016 年 12 月，主要成员，从事智能终端上基于指纹的密码攻击。

致谢

硕士研究生的学习即将随着论文的答辩结束，在这段重要的人生旅途中，是我的老师和同学们伴随我一路走来，教导和帮助我不断前进。

在这里我首先要对我的指导老师罗军舟教授表示最崇高的谢意。罗老师是国内计算机网络领域顶尖科学家。在我硕士研究生时间里，罗老师一直非常关心我，从学习到研究状况，从论文的选题到开题、从具体的研究到实现，罗老师都给予了我精心的指导，罗老师指引我真正地走进科研殿堂。在师从罗老师的这段时间里，罗老师以他视野开阔、治学严谨、诲人不倦的良师风范，勤奋务实的工作态度和谦逊随和、以诚待人的优秀品质令我敬仰万分。在我以后的学习和工作中，我将始终铭记罗老师的教导，一步一个脚印，踏踏实实地学习和工作。

同时，我要衷心地感谢杨明老师。杨老师思维敏捷，认真负责，在我研究生期间，在学习和生活上给予了我极大的帮助与指导，而且杨老师在网络安全方面深厚的学术造诣深深地影响了我对这个领域研究的理解。仍然记得每次杨老师与大家讨论学术观点时，总能提出令人耳目一新的好想法；仍然记得每次我碰到困难的时候，杨老师总能帮我分析清眼前的形势；仍然记得杨老师帮我垫付车费、关心我们每一个人的学习、经常给我鼓励等一系列感人的小事。

然后，我还要感谢实验室的曹玖新老师、宋爱波老师、李伟老师、刘波老师、东方老师和吴巍炜老师。感谢大家平时对我的学习和研究工作的关心和帮助。

接着，我还要感谢网络安全与无线组的吴文甲、何高峰、王维、顾晓丹、龚奇源、顾梁、高亢、傅煜川、宋天宝等师兄，在我刚进入实验室时，他们在工作和生活上给予我很多的帮助和关怀。在此，我要特别感谢在我进入硕士实验室以后，对我帮助很大的凌振师兄。凌振师兄在学术研究中取得了骄人的成绩，他在自身进步的同时，还能指导我、帮助我进行科学研究。如果说罗老师与杨老师是指引我进入科研领域的引路人，那凌振师兄则是这条道路上的催化剂，加速我科研思维的形成。另外，也同样感谢我的同学张泽西、肖鹏、舒晴、倪玮等和学弟黄斐乔、徐昊等，两年的时光给大家留下了永远的美好回忆，这里也祝愿大家学业有成。

最后，我要感谢我的家人和朋友们，无论我面对怎样的困境，他们永远都默默的支持我，鼓励我，给予了我无私的关爱和帮助。

张扬

2013年8月于东南大学九龙湖校区计算机学院楼