Android 环境下恶意软件静态检测方法研究

重庆大学硕士学位论文

（学术学位）

学生姓名：张 锐

指导教师：杨吉云副教授

专　　业：计算机应用技术

学科门类：工　学

重庆大学计算机学院

二 O 一四年四月

Research on Malware Detecting based on Static Analysis under Android Environment

A Thesis Submitted to Chongqing University

in Partial Fulfillment of the Requirement for

Master's Degree of Engineering

By

Rui Zhang

Supervised by Ass. Prof. Yang Jiyun

Specialty：Computer Software and Theory

College of Computer Science of

ChongqingUniversity, Chongqing, China

April, 2014

摘　要

近年来，智能手机快速普及，逐渐成为人们日常生活中不可或缺的一部分。在众多的智能手机系统中，Android 系统以其免费开源的特点，迅速占领了智能手机操作系统市场。Android 平台下丰富的软件应用给人们的生活带来了极大的便利。然而，混杂在正常应用软件中的恶意软件数量也以惊人的速度不断增长，Android 平台的安全问题变得越来越严重。因此，如何应对 Android 恶意软件的飞速发展，以较小的开销准确的检测出 Android 恶意软件是一个值得研究的课题。本文以静态检测的方式对 Android 环境下恶意软件检测进行了研究，具体成果如下：

①首先，对智能手机的发展历史进行了回顾，分析了移动恶意软件对于智能手机平台的安全威胁。然后，按照检测时机的不同，对三种主要的恶意软件检测方法（动态检测，静态检测和云计算检测）进行了介绍和优缺点分析。最后，对国内外学术界在 Android 恶意软件检测领域的研究成果以及难点进行了阐述和总结。

②针对低开销初步判断 Android 软件恶意倾向的需求，提出一种从 Android 权限角度判断恶意软件的检测方法。分析 Android 权限机制自身的特点,利用特征预处理知识，从权限之间以及权限和恶意倾向之间的相关性入手，进行特征选取和特征去冗余，形成最具代表性的权限特征集合作为分类依据。实验结果表明，在少量特征情况下，获得了一定的检测准确率。作为初步检测方案，具备一定的实用价值。

③针对静态检测在应对新型 Android 恶意软件时，检测准确率下降的问题，提出一种基于集成概率神经网络的恶意软件检测方法。检测方案以逆向获取的高危 API 作为检测特征，利用概率神经网络用于分类所具备的样本容错性好、追加能力强等特点作为分类器。同时,利用 Bagging 集成技术,形成差异度大的个体概率神经网络进行集成，进一步提高 Android 恶意软件检测的泛化能力。实验结果表面，整体检测准确率较高，新样本学习能力强，应用于新型 Android 恶意软件检测，检测准确率只出现了小幅度下滑，改善了泛化能力。

关键字：Android 恶意软件，特征选取，集成学习，概率神经网络

ABSTRACT

In recent years, with the rapid spread of smart phones, it has been more and more indispensable in people's daily life. Among the many smart phone system, Android quickly occupied the smart phone operating system market with its advantages of free and open source. Explosive growth of the software based on Android system has brought great convenience to people's lives. However, the number of malicious software mixed in normal software is also growing at an alarming rate. Therefore, how to cope with the rapid development of Android malware and detect the Android malware accurately with less overhead is a topic worth studying. This thesis mainly studies by static method to classify the Android software, to achieve the purpose of detecting malicious android software, the main work is as follows:

① Review the development history of smart phones. Analysis the mobile malware threats to the security of smart phones. Three main malware detection methods (dynamic detection method, static detection method and the cloud based detection method) are introduced and analysis of its advantages and disadvantages. Review and summarize the domestic and foreign research achievements in the field of Android malware detection.

②Considering the demand of preliminary detecting malware of android at low cost, a method of detecting malicious tendency is proposed on the basis of permission correlation. Use pre-treatment knowledge of characteristics to select typical features and reduce redundancy on the basis of correlation between each permission and the permissions to malicious tendencies to get the most representative permissions set as the classification basis. The experimental results show that the proposed scheme under the condition of small features, obtain a certain detection accuracy. It has a certain practical value as a preliminary test method.

③Considering the ubiquitous issue of accuracy drop when detecting new android software samples on the basis of old samples, a new malware detection method was proposed based on the probabilistic neural network ensemble. Decompile android code to identify the Dalvik API as the high-risk characteristics through reverse engineering. Probabilistic Neural Network was used as classifier with its features of good fault-tolerance and additional ability. And Bagging integration technology was used to form the difference degree of individual probabilistic neural network, and integrate each

probabilistic neural network to improve the generalization ability on Android malware detection. The experimental results show that the proposed scheme has high detection accuracy with strong samples learning ability. When applied to detect new Android malware, detection accuracy rate appears only a small decline, and generalization ability has been improved.

Keywords: Android Malware, Feature Selection, Ensemble Learning, Probabilistic Neural Network.

目　　录

1 绪　论

本章首先阐述了智能手机恶意软件的研究背景，进而总结了 Android 恶意软件检测的研究现状，在此基础上归纳了本文的主要贡献。最后介绍了本文的组织结构。

1.1 研究背景

本节首先描述了智能手机近年来的发展状况，然后分析智能手机系统领域中，Android 系统所占有的重要地位，最后对 Android 系统上恶意软件的发展现状进行了讨论。

1.1.1 智能手机发展状况

手机作为人类通信史上的一个伟大发明，从它诞生至今，已经过去了十几个年头。而智能手机是最近几年才开始普及使用的新兴事物，它的出现可以说是手机发展历程中一个非常重要的转折点。与传统的功能手机相比，手机的作用开始出现巨大的改变。智能手机不仅可以打电话、发短信，更重要的是智能手机可以实现用户随时随地上网的需求，实现更多智能化的应用。IDC 在 2014 年最新一份报告中指出，智能手机的出货量仅在 2013 年达到 10.004 亿台，比 2012 年的 7.253 亿台同比增长了 38.4%。智能手机在全球手机出货量中的比例继续加大，达到 55.1%，同比增长 9.4%[1]

Android 系统是由 Google 成立的 OHA(Open Handset Alliance，开放手机联盟）开发而成，它的底层基于 Linux 系统，现已广泛应用于智能手机平台。Android 系统自 2007 年发布以来，发展迅速，在智能手机操作系统市场占据了绝对的主导优势，其最大的竞争对手来自高端手机市场的 Apple IOS。Strategy Analytics 最新发布的手机操作系统市场份额统计数据（见表 1.1）显示，2013 年 Android 系统的市场份额达到了新高 78.9%, 在 2012 年 68.8%的基础上有了较大的跃升。而 Applie IOS 的市场份额则呈现下降趋势，从 2012 年的 19.4%下降到 2013 年的 15.5%。

表 1.1 2012-2013 全球手机操作系统市场份额对比

Table.1.1 The Comparison of the Market Share of Global Smartphone Operating System during 2012-2013

手机操作系统	2012 年第 4 季度/%	2012 年全年/%	2013 年第 4 季度/%	2013 年全年/%
Android	70.3	68.8	78.4	78.9
Apple IOS	22.0	19.4	17.6	15.5
Microsoft	2.7	2.7	3.2	3.6
其他	5.0	9.1	0.7	2.0

数据来源：Strategy Analytics 无线智能手机战略服务(WSS)[2]

图 1.1 由表 1.1 中数据生成, 更加直观的反应了 2013 年全球智能手机市场份额对比, 其中 Android,Apple IOS 和 Microsoft 一共占据了 98%的市场份额, 而 Android 以 78.9%位列第一。

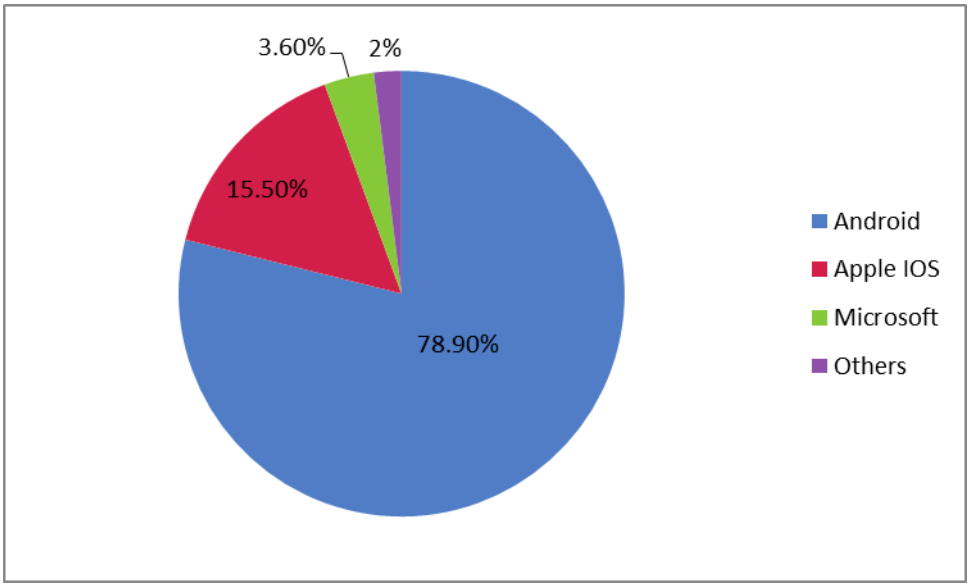

图 1.1 2013 年全球智能手机系统市场份额

Fig.1.1 The Market Share of Global Smartphone Operating System in 2013

以上数据反映了当前智能手机平台的发展现状，其中 Android 系统在智能手机系统中占有重要地位, 在学术领域对 Android 系统进行关注和研究非常具备现实意义。

1.1.2 智能手机与恶意软件

随着 Android 系统的发展壮大，恶意软件随之而来。最早的恶意软件出现于传统的 PC 平台，恶意软件是一类具有恶意倾向的特殊软件，它能够损害系统安全，

打乱日常软件运行。互联网协会总结了恶意软件的特征，并给出定义[3]，“恶意软件是指在未明确提示用户或未经用户许可的情况下，在用户计算机或其他终端上安装运行，侵害用户合法权益的软件”。一般具有如下行为的软件，可以被定义为恶意软件，概括如下表 1.2。

表 1.2 恶意软件主要破坏行为

Table.1.2 The Main Sabotage of Malware

恶意行为	特征描述
强制安装	指未明确提示用户或未经用户许可，在用户计算机或其他终端上安装软件的行为
难以卸载	指未提供通用的卸载方式，或在不受其他软件影响、人为破坏的情况下，卸载后仍然有活动程序的行为
浏览器劫持	指未经用户许可，修改用户浏览器或其他相关设置，迫使用户访问特定网站或导致用户无法正常上网的行为
广告弹出	指未明确提示用户或未经用户许可，利用安装在用户计算机或其他终端上的软件弹出广告的行为。
恶意收集用户信	指未明确提示用户或未经用户许可，恶意收集用户信息的行为。
恶意卸载	指未明确提示用户、未经用户许可，或误导、欺骗用户卸载其他软件的行为。
恶意捆绑	指在软件中捆绑已被认定为恶意软件的行为。
其他	其他侵害用户软件安装、使用和卸载知情权、选择权的恶意行为。

移动平台方面，尽管第一个移动平台恶意软件“Liberty Crack”，直到 2000 年才第一次出现，但移动恶意软件在 2004 到 2006 之间发展迅速[4]。海量的恶意软件在这一时期爆发，并不断进化。这些恶意软件同传统 PC 平台的恶意软件十分相似，包括病毒、蠕虫、木马以及扩展开来的后门程序、恶意广告软件等。截至 2012 年底，共有 46，445 个恶意程序，然而到了 2013 年 7 月，KaspersKy 实验室已经在他们的实验系统中集成了 100,386 个恶意程序样本库，到 2013 年 12 月，移动恶意软件达到 148,778 个[5]。2013 年第一季度，移动安全服务企业网秦发布的全球手机安全报告[6]显示，恶意软件感染的手机达到 1040 万部，与 2012 年同一时期相比，增长了 99.23%。其中，Android 系统的手机在感染对象中达到了 82%的惊人比例。在众多感染特征中，恶意扣费以 26%的感染率位居第一，隐私窃取，远程控制，资费消耗紧随其后，分别为 20%，18%和 14%。以上数据说明，智能手机平台，尤其是 Android 平台上，恶意软件发展非常迅速，它们的恶意行为已经给用户利益带来了严重的损害。

1.2 Android 恶意软件检测研究现状

本节首先对 3 种主流的恶意软件检测方案[7],动态检测、静态检测以及近年来兴

起的基于云计算的检测方案进行综述，分析三种检测方案的基本原理和优缺点。接着梳理和总结了 Android 恶意软件检测的研究成果。最后针对当前 Android 恶意软件检测研究中存在的难点和挑战进行了讨论。

1.2.1 检测方法综述

Android 平台的恶意软件检测，按照检测时机划分，整体上可以划分为动态检测、静态检测以及基于云计算的检测。所谓动态检测，就是在系统运行的情况下去收集相关信息，利用监控软件监控其在运行状态下，是否有联网、获取隐私等行为，判别软件是否有恶意性。静态检测则不需要在实际运行过程去分析程序，而是直接分析软件本身来判断程序中是否有恶意倾向。基于云计算的检测是针对移动设备在电量和计算能力方面的局限性，把检测方案部署到具备海量存储和大量计算能力的云端服务器上，在被检测设备上只保留代理软件采集基本信息，检测结果通过网络返回，从而提高检测表现，节省手机能源。

1 动态检测方法

动态检测的基本数据来源于软件运行时的行为表现，在动态分析时，软件被运行于一个封闭的运行环境中（比如虚拟机）,运行过程中的软件行为在这个环境中被记录下来[8-10]，然后通过一些常用的动态分析方法进行分析。

影响动态检测精确度的关键点有两个。一个是软件封闭运行时的，用户操作情况模拟，与之相关的是自动化测试技术[11]。它是将以人为主要驱动力的测试行为转化为以机器为主要执行者的一种过程。使用自动化测试技术可以更加全面真实的模拟用户使用软件的行为，根据方案设计，去遍历软件功能点，从而模拟用户的日常使用。目前，已经有一些比较成熟的自动化测试工具，比如 Monkey，Monkeyrunner，Robotium，HierarehyViewer 等[12]，这些工具都可用于动态检测中，合理使用这些工具可以帮助真实重现用户软件使用习惯，触发对应的软件行为，为动态分析提供基本依据。影响动态检测方案精确度的第二个关键点是软件行为的终端监控技术。Android 平台一些常用的监控技术包括函数监控, 函数参数分析, 信息流追踪, 指令分析等[8]。此外常用的还有通过分析软件运行时的网络数据包分析[13]和利用 Android SDK 提供的工具 sqlite3 [14]进行数据库分析。

总的来说，动态检测技术检测精确度较高，以软件行为为检测依据，对于未知新出现的恶意软件也能保持较好的检测准确率。但是，动态检测具备检测方案一般较为复杂，对于资源消耗程度较大，实时性不高。检测精确度依赖于自动化测试方案的设计，在自动化测试软件时，很难覆盖到软件全部的执行路线，找出只有特定情况才发生的恶意行为非常困难。

2 静态检测方法

静态检测方案[9,10]，是在不执行软件的情况下，对软件进行分析，通过逆向工

程技术[15]，抽取程序的特征，如权限信息，操作码序列，函数调用，恶意程序片段等，以这些特征作为基础，来还原程序的行为。静态检测的一般流程如下图 1.3，分为样本收集，逆向分解，特征提取，样本学习，检测分析几个阶段。检测方案涉及到机器学习、数据挖掘、模式识别等相关知识，有着扎实的理论的知识作为支撑。

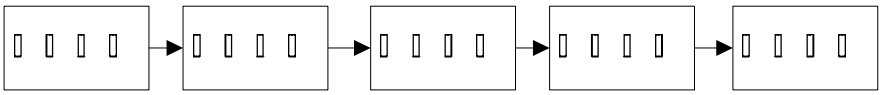

图 1.3 静态检测方案一般流程

Fig.1.3. The General Process of Static Detection Method

与动态检测相比，静态检测只用分析一次就可以查出应用中的可疑恶意行为。静态检测的优点主要是自动化程度高，分析速度快，但是也存在精确度相对较低，对未知恶意软件检测准确率下降的问题。

3 云计算检测方法

随着前沿研究的发展，近年来兴起的云计算也被使用于 Android 恶意软件检测中，形成了一批基于云计算的检测方案[10, 16]。云计算检测方案中，所有大计算量的恶意检测计算都被放到了云端，而信息的采集则是依赖于移动设备上的代理程序。云端服务器存储有丰富的恶意软件鉴别信息，代理程序收集好程序运行时的各类信息后，发送到云端服务器。云端服务器进行综合的静态和动态检测，得出最后的判断结果。然后返回给移动设备。

和传统静态检测方案相比，实时性更强，能够及时获得软件恶意倾向信息，同时又因为把核心计算部分移到了云端，移动设备上只需运行代理软件，保证了合理资源开销。云端强大的存储和计算能力，保证了可以实施较为复杂的静态动态结合方案，提高恶意软件检测准确率。

云计算检测方案为提高恶意软件检测准确率，提供了一个新的思路。不过云计算检测方案对于如何利用云计算的相关理论知识，以及移动设备代理端和云端服务器如何更好的配合等，都还有待继续深入的研究。同时云计算检测方案的核心，云端具体的检测方式的实质也是静态检测和动态检测方案的综合使用，它的发展还是依赖于静态检测和动态检测相关研究理论的不断深入。

1.2.2 智能手机恶意软件检测研究成果

传统的 PC 平台有较多的技术积累，机器学习、数据挖掘等理论知识被广泛应用于恶意软件检测当中[17-21]，因为智能手机系统和传统 PC 系统有一定共通性，在智能手机的恶意软件检测研究初期，研究者们尝试把之前 PC 端积累的恶意软件检

测理论简单的迁移到智能平台上来。

Schmidt 等人[22]利用 Linux 平台的理论进行了尝试，考虑到 Android 系统底层是基于 Linux 实现的，提出利用 readelf 工具静态抽取可执行链接格式文件的函数调用列表, 再运用 PART,Prism 和 Nearest Neighbor Algorithms 等算法进行分类判断。由于没有足够的 Android 恶意软件，实验样本也以 Linux 恶意软件作为了代替。Burguora 等人[23]同样从传统 Linux 恶意软件检测角度入手，利用动态检测的思想，开发了一个叫做 Crowdroid 的客户端去监控 Android 底层的 Linux Kernel 系统调用，然后进行划分聚类，识别恶意软件。

虽然智能手机系统和传统 PC 系统在原理上有一定的相似性，但是智能手机在计算能力、存储能力、电量、移动特性以及软件特征等多个方面和传统 PC 还是存在差异。要想在智能手机平台取得更准确的恶意软件检测率，仅仅对传统的 PC 平台的理论和技术进行迁移还不够。

为此，Chiang[24]提出利用本体论的知识，对智能手机恶意软件样本进行分析。提取出共有的行为特征（如访问文件系统，下载安装程序等），再对这些行为进行描述，形成手机平台特有的行为特征。Enck 等人[25]设计了一种认证机制，形成 APK(Android Package)软件的安全规则，在安装过程中对 APK 文件进行检测，如果 APK 文件不能通过安全规则的认证就阻止安装，从软件进入手机的源头上进行了管制，防止恶意软件进入手机。Shabtai 等人[26]在 Android 平台实现了一个基于行为的检测系统 HIDS(host-based intrusion prevention systems)，动态监控手机设备的各种特征和事件，再利用决策树和回归分析等方法对收集的特征进行评估，将手机的软件分为无害和恶意两类。戴帅夫等人[27]对 Windows Mobile 平台的恶意软件检测展开了研究，提出了一个基于软件行为检测的恶意软件检测方案。以 API 调用序列作为行为特征，通过 API 拦截技术获取调用信息，然后以有穷状态机的方式来表示。同时，把手机恶意软件的行为模式用下推自动机表示，于是对恶意软件的检测就转换成了对有穷自动机和下推自动机的匹配。

1.2.3 Android 恶意软件检测研究的难点和挑战

Android 恶意软件检测研究面临的问题，首先是样本的缺乏。同传统计算机平台的情况不同，Android 应用软件的分发渠道比较单一。用户安装软件主要通过官方应用商店 Googel Play 或者第三方软件中心（比如 360 软件中心）进行下载。个人通过公开渠道获得大量的恶意软件样本十分困难。同时 Android 系统的版本更新十分迅速，不同的版本下，软件的特性有一定的差别，这对于恶意软件检测的准确性存在影响。

其次，传统 PC 的恶意软件检测理论应用到 Android 平台时，如何考虑智能手机有限的资源（存储能力、计算能力、电量等）对于检测方案实施性的影响也是

一个难点。尽可能的在保持检测准确率的情况下，节省检测开销，是一个值得研究的问题。

最后，Android 平台恶意软件发展迅速，各种新型恶意软件频繁显现。面对这一情况，动态检测方式虽然可以根据软件运行时的表现，进行监控判断，检测恶意倾向，但是很难覆盖软件运行所有路径，并且开销较大（需要沙盒，虚拟机等）。而静态检测方案，对于样本库已有特征的恶意软件，可以较好检测，但对于新出现的恶意行为有一定差异的恶意软件，检测准确率呈现下降[28]。如何在保证较小资源开销的情况下，提高恶意软件检测面对新样本的泛化能力，保持准确率不大幅度下滑，也是当前 Android 恶意软件检测所面临的难点和挑战。

1.3 课题研究内容与意义

本论文首先介绍了当前三类主流的 Android 恶意软件检测方法(动态检测、静态检测和基于云计算的检测)，对各自的技术特点和优缺点进行了分析。然后对智能手机恶意软件检测的主要研究成果进行了阐述，总结了当前手机恶意软件检测研究存在的难点和挑战。最后，在对 Android 及其相关安全机制分析的基础上，把 Android 恶意软件静态检测作为了研究的重点，从节省开销形成初步检测结果和提高泛化能力应对新的恶意软件样本两个方面进行了研究讨论。主要成果归纳如下：

(1)分析了 Android 恶意软件的发展形势和危害，对三种主要的恶意软件检测方法：动态检测、静态检测和基于云计算的检测的优缺点进行了分析。梳理和总结了恶意软件检测研究的发展过程。

(2) 提出以 Android 软件的权限信息为特征，利用 Android 系统权限机制的特点，进行特征选取和特征去冗余，形成小规模的软件特征集合。然后改进朴素贝叶斯分类器用于检测软件进行分类，实现了软件恶意倾向的初步判断。

(3)针对 Android 静态检测面对新样本检测准确率下降的问题，提出利用概率神经网络训练速度快、学习能力好，样本追加能力强的特点，进行恶意软件检测，反编译 Android APK 安装包，获取 Dalvik 汇编码中的高风险 API 作为检测特征，结合 Bagging 算法，反复抽取软件样本，形成多个概率神经网络进行学习，形成差异度大的个体概率神经网络。最后的通过集成的多个神经网络投票来决定检测结果，提高了 Android 恶意软件静态检测的泛化能力。

1.4 本文组织结构

本文共分七章，内容如下:

第一章：绪论。介绍课题研究背景，分析智能手机发展状况以及恶意软件的飞速发展给智能手机带来的安全威胁。对主流的检测方法进行了梳理，讨论了

Android 恶意软件检测研究中存在的难点和挑战。

第二章：Android 及其安全机制。首先对 Android 系统进行了综述，分别介绍了 Android 体系结构，应用基本概念以及 Dalvik 虚拟机的相关知识。然后在此基础上，按照 Android 体系由下往上的层次，依次从 Linux 内核，本地库和运行环境，应用程序框架层，对 Android 的安全机制进行了分析。。

第三章：机器学习和神经网络算法简介。第一部分，介绍了机器学习的定义，对机器学习中的特征选取技术和主要分类算法进行了阐述。第二部分，讨论神经网络的基本概念，简要介绍概率神经网络和神经网络集成的相关知识。。

第四章：基于 Android 权限相关性的恶意软件初步检测，提出了一种 Android 恶意软件初步检测的方案。首先对 Android 权限机制进行了分析，在此基础上，对 Android 软件的权限进行特征选取和特征去冗余，形成小规模的权限集合,最后改进朴素贝叶斯分类器用于分类，实现恶意软件的初步检测。

第五章：基于集成概率神经网络的 Android 恶意软件深入检测方案。提出深入分解 Android APK 安装包，以 Dalvik 汇编码中的高危 API 作为软件特征，利用集成概率神经网络泛化能力强、学习能力好、训练时间短的特点进行恶意软件检测，提高对未知样本的恶意软件的检测能力。

第六章：实验和结果分析。介绍了第三章和第四章恶意软件检测方案中所需要用到的技术和工具。收集正常软件和恶意软件样本，分别对两种方案进行实验和对比。分析实验结果，求证所设计方案的实验效果。

第七章：总结和展望。总结全文的研究工作，分析当前研究中存在的不足，对以后的研究方向提出了展望。

2 Android 及其相关安全机制

全面了解 Android 系统的体系结构和安全机制，是理解 Android 恶意软件检测技术的基础。因此, 本章对 Android 系统中与恶意软件检测相关的知识进行了梳理。内容包括，Android 体系结构，Android 应用基本概念以及 Android Dalvik 虚拟机简介。然后，对应于 Android 体系结构，从下往上依次从 Linux 内核、Android 本地库及运行环境、应用程序框架层 3 个方面介绍了 Android 现有的安全机制。

2.1 Android 综述

本节主要对 Android 的基本概念进行综述，包括三个部分，首先介绍 Android 的体系结构(Android Linux 内核、Android 本地库和运行环境、应用程序框架以及 Android 应用程序)。接着对 Android 软件应用的基本概念进行归纳，主要包括 Android 应用运行原理，Android 应用四大组件，组件激活方式以及相关配置文件。最后对 Android 应用运行的基础平台，Dalvik 虚拟机进行介绍。

2.1.1 Android 体系结构

Android 系统采用分层思想构建，主要由 Linux 内核、函数库和运行环境、应用框架和应用程序组件组成，如图 2.1 所示。

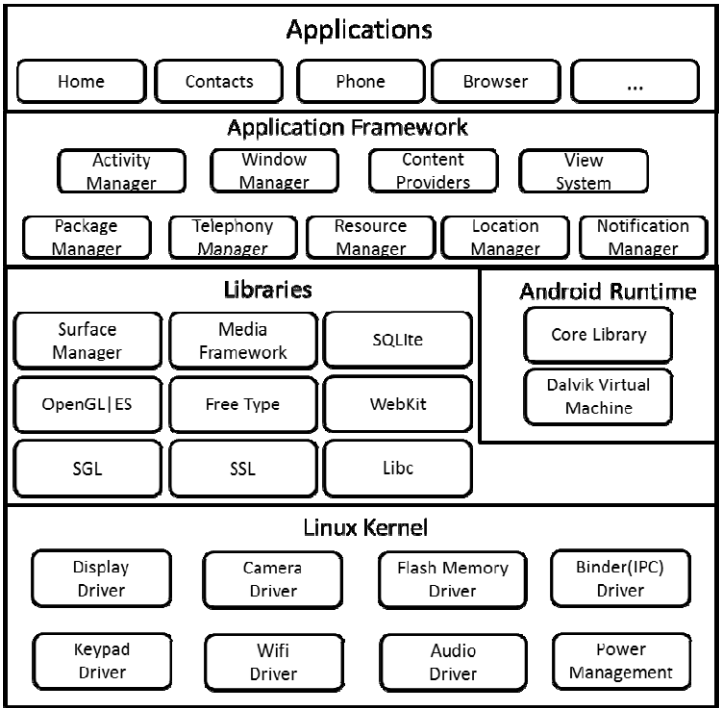

图 2.1 Android 体系结构

Fig. 2.1 The System Structure of Android

第一层、Linux 内核

Android 系统核心服务包括安全控制、内存管理、进程管理、网络协议栈和驱动模型等部分，这些服务都构建在 Linux2.6 内核功能之上。Linux 内核在这个过程中扮演着硬件和软件之间的抽象接口的角色。

第二层、函数库和运行环境

许多被上层 Android 组件使用的 C/C++函数库都包含于这一层。这些函数库通过上层的应用框架层暴露给应用开发者调用，一些典型的函数库如下表 2.1：

表 2.1 Android 典型依赖函数库

Table 2.1 The Typical Function Library of Android

函数库	含义
Surface Manager	执行多个应用程序时候，负责管理显示与存取操作间的互动，另外也负责2D绘图与3D绘图进行显示合成。
Media	Android多媒体的核心部分，基于PacketVideo（即PV）的OpenCORE
SQLite	一个通用的嵌入式数据库
OpenGL|ES	提供对3D的支持
FreeType	位图和矢量字体的功能
WebKit	网络浏览器的核心
SGL	2D图像引擎
SSL	Secure Socket Layer，位于TCP/IP协议与各种应用层协议之间,为数据通讯提供安全
Libc	从BSD继承来的标准C系统函数库，专门为基于embedded linux的设备定制

同时，Android 在这一层还包含了一个核心库和 Android 虚拟机。核心库提供了 Java 编程语言核心的大部分功能。这个 Android 虚拟机叫做 Dalvik,它不同于传统的 Java 虚拟机，被重新设计以满足在同一个设备上同时高效运行多个虚拟系统。实质就是每一个 Android 应用都对应到一个 Dalvik 虚拟机实例，多个 Dalvik 虚拟机实例同时运行，互不干扰。

第三层、应用程序框架层

开发人员通过使用该层提供的框架 API，具有和 Android 自带系统应用一样的完全功能访问权限。这种设计简化了组件的重用，任意应用程序都能封装自己的功能模块。该层开发的功能模块除了被自己使用之外，同时还能被符合规则的其他应用程序使用。应用程序重用的设计使得用户可以通过更换代码模块来很方便的开发自己想要的应用程序。

第四层、应用程序层

Android 的应用程序主要以 Java 编码而成，把 JAVA 程序及相关资源经过编译后，生成一个 APK 安装包供用户安装使用。Android 在这一层本身提供了短信、浏览器、电话、地图等众多的核心应用。此外，软件开发人员还可以使用应用程

序框架层的 API 来开发自己的程序。

2.1.2 Android 应用程序概述

构建一个完整的 Android 应用，一般由多个应用组件组合而成。每个应用组件有着特定的作用，不同应用组件之间又相互联系。下面分别介绍 Android 系统四种基本的应用组件：

①Activities

一个 Activity 代表手机上可以看到的每一页。比如一个地图应用软件，可以有一个 Activity 来显示当前地图，一个 Activity 来显示地图显示设置，另外一个 Acitivty 来接收地图搜索信息，这些 Activity 共同构成了一个完整的应用软件，并且互相之间相互独立。Android 系统中不同应用的 Activity 之间也可以相互关联，比如在发微博的时候，想给文字配上一张图片，那么可以在微博应用中启动图库的 Activity，来找到需要的图片。

②Services

Service 用来执行一些耗时的操作或是远程进程的任务，它运行系统的后台，对于用户是不可见的。比如当用户在使用一个其他的应用程序时，Service 可以在后台播放音乐，或者它可以在后台从网络下载数据而不打断用户当前的软件交互流程。它主要通过前面提到的 Acitivty 来启动或是终止。

③Content providers

Content providers 对应用程序之间共享的数据集提供管理功能。应用程序可以把数据保存在文件系统，SQLlite 数据库中，然后其他的应用可以通过 Content providers 来访问甚至修改这些数据。比如，Android 系统中比较隐私的电话本资料，如果一个应用程序拥有相关权限，那么它就可以通过 Android 系统提供的 content provider 来调用编辑用户的电话本信息。

④Broadcast receivers

Broadcast receiver 组件被用于响应系统范围内的广播通知，系统会发布很多广播通知，比如屏幕关闭、电池电量低或者图片被截屏等。应用程序也可以发布广播信息，比如在手机助手类软件中下载应用时，当软件已经下载完毕，就会发布广播，通知关联的安装程序，数据已经下载完成，可以安装。Broadcast Receiverhe 和 Service 一样没有用户界面，但是它可以在状态栏发出通知表示广播发出。通常，Broadcast Receivers 只是被用于处理一些小型事务，比如启动一些基于某些特定事件的 Services。

四个组件之间的关系，简要概括如图 2.2

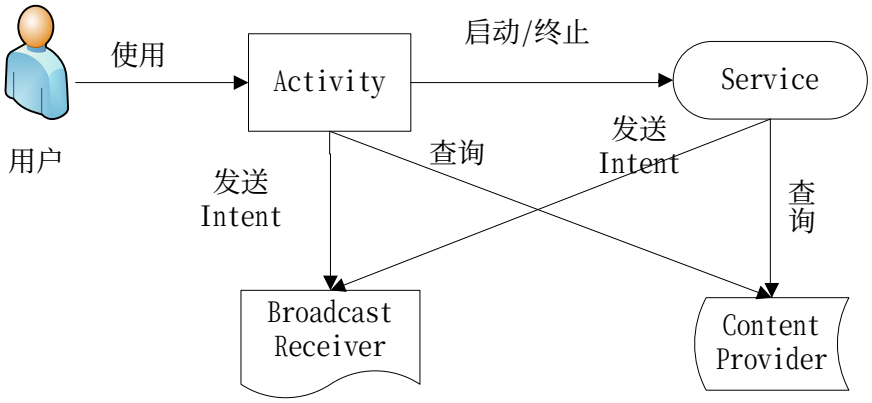

图 2.2 Android 各组件关系图

Fig 2.2 The Relational Graph of Android Components

用户直接使用的组件是 Activity，Activity 里面可以启动或终止 Service，访问 Content Provider 的数据，或者通过 Intent 发送消息给 Broadcast Receiver。后台运行的 Service 同样可以查询 ContentProvider 的数据, 或是给 Broadcast Receiver 发送消息。

2.1.3Android Dalvik 虚拟机

虽然 Android 平台以 Java 作为开发语言, 但 Android 程序却不是运行在标准 Java 虚拟机上的。Google 为 Android 平台专门设计 Dalvik 的虚拟机来运行 Android 程序，Dalvik 虚拟机与传统的 Java 虚拟机有着许多不同点，两者并不兼容，它们显著的不同点主要表现在以下几个方面[14]：

①字节码运行格式不同

Java 虚拟机中运行的是 Class 文件，它是由 Java 程序编译而来。Dalvik 虚拟机中运行的是 Dex(Dalvik Executable)文件。虽然它同样由 Java 程序编译形成字节码，但是 Dex 文件对字节码文件进行了重新的排列，把每个类文件的常量池分解开来，去除重复过后，重新组成一个新的常量池，使得相同的字符串、常量在 Dex 文件中只出现一次。

②可执行文件大小不同

Dalvik 的可执行 Dex 文件，针对手机平台资源有限的情况，去除了文件中的冗余信息，文件的体积更小，减少初始化时文件加载与解析的时间。

③运行时架构不同

Java 虚拟机基于栈结构，Dalvik 虚拟机基于寄存器结构。栈结构使得数据读写时需要反复访问内存，需要更多的指令来完成工作，而寄存器结构，使得数据可

以直接传递，这种访问方式相对而言会快很多。

2.2 Android 安全机制

Android 系统构建于 Linux 内核之上，继承了 Linux 的安全特性，并在此基础上做出了创新，进行了有针对性的改进。本部分从自下而上的顺序，分别从 Linux 内核、本地库和运行环境和应用框架层三个方面, 介绍了 Android 系统的安全机制。

2.2.1 Android Linux 内核安全机制

Android Linux 内核部分继承了 Linux 两个最重要的安全措施，应用程序用户 ID 控制和自主访问控制。

应用程序用户 ID 控制是指在安装 Android 应用时，每个应用都会被分配给一个属于自己的用户 ID，并且为它创建一个沙箱，以防止影响其他程序（或者其他程序影响它）。Android 系统的 uid 对应着 Linux 系统的 uid，Linux 系统通过用户的 uid 实现权限控制，Android 继承了这种能力。一个程序如果想访问系统资源或是其他应用资源，那么它必须在 manifest 文件中声明相关权限或者是共享 uid。

自主访问控制机制则是由文件访问控制来实现，在 Linux 系统下，文件的读写控制由两组属性构成，分别是用户组别（自己，组员，其他）和读写权限（读，写，执行）。通过组合，可以达到多种控制效果。在 Android 系统下，uid 是“root”或是“system”的用户才能完全访问系统文件资源，对于普通应用程序，如果需要访问，则需要申请 Android 权限。

2.2.2 Android 本地库及运行环境安全机制

Android 系统在本地库及运行环境层主要依赖于库函数和 Java 自身的安全机制，这里重点介绍 sqllite 的安全机制以及 Java 强制类型安全机制。

SQLite 和大型数据库的实现机制不同, 它是针对嵌入式设备有限资源情况而设计的一种轻量级数据库，使用单一文件来存储数据库的结构和内容。数据库自身不具备用户管理、访问控制和授权机制，完全由操作系统来处理访问控制。系统认可的用户即可拥有对数据库的访问权限, 因此 Android 数据库的安全性还有待提高。

另一方面，Android 在这一层的 Dalvik 虚拟机本质上是 Java 虚拟机的一种，它继承了 Java 自身的安全机制。主要包括编译器对强制类型的支持，自动内存管理，字节码验证机制以及独特的安全加载方式等。

2.2.3 Android 应用程序框架层安全机制

Android 应用层主要的安全机制包括权限检测机制和数字签名机制。应用框架层丰富的 API 为软件开发者提供了对硬件设备完整的控制能力，但是由于对硬件的访问会涉及到高危敏感的个人信息，Android 采取了权限检测机制来提高系统安

全性。开发者在使用 API 开发了软件应用后，需要在软件生成 APK 安装包之前，在 XML 文件中声明 API 用到的权限信息，这样软件才能正常使用，否则软件在使用过程中会因为权限不够而报错。API 和权限申明信息是一个多对多的关系，Android 官方并没有给出 API 对应的权限信息，需要开发者自己根据权限含义和设备使用情况来选择合适的申明信息，这在一定程度上引起了开发者对于权限信息的滥用，出现了为了防止权限不够而多加权限信息的情况。

数字签名机制也是该层重要的安全机制，所有发布的 Android 应用程序都必须有数字签名，数字签名是软件作者身份的重要标识。给软件签名的个人证书文件通过 Keytool 工具生成，一般由个人信息和密码两部分构成，整个过程由软件开发者自己即可完成，不需要数字证书机构签名。数字签名机制的使用，不仅避免了软件应用的鱼龙混杂，以假乱真，同时对于软件应用的升级，应用程序的模块化以及同一签名下应用数据的共享都有重要的意义。

2.3 本章小结

本章首先介绍了一些 Android 系统的基本概念，包括了 Android 体系结构，Android 应用基本概念以及 Android Dalvik 虚拟机的基本知识。然后对应 Android 体系结构，以从下到上的方式分别介绍了 Linux 内核层、本地库及运行环境层和应用程序框架层的 Android 安全机制。以此建立对 Android 系统以及 Android 恶意软件检测技术一个整体的认识，为 Android 恶意软件检测技术的认识和研究奠定了理论基础。

3 Android 恶意软件静态检测相关理论

恶意软件静态检测的实质就对未知的应用软件进行二分类的过程，通过对应用软件的特征进行分析，把它归类到正常软件或者恶意软件。分类之前需要有足够的样本，提取 Android 应用的典型特征，训练分类模型，最终完成对未知应用软件的分类检测。这个过依赖于机器学习等多领域相关知识作为理论依据，本章主要对 Android 恶意软件静态检测相关的一些基本理论进行介绍。

3.1 机器学习概述

机器学习是一门交叉学科，近二十年来开始兴起，涉及概率论，统计学，近似理论，凸分析，复杂性理论等多门学科。它的主要目标是设计一些通过计算机自动学习的算法去分析数据，获得规则，然后用规则来预测未知数据。关于机器学习的定义很多，其中比较经典的有 Tom M. Mitchell 的定义,“计算机程序可以就一些任务 T 和性能测量 P 来从经验 E 进行学习，如果其在 T 上，由 P 测量的性能能够提高经验 E，我们认为这就是机器学习(A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E)”[29]。

机器学习被广泛应用于数据挖掘、计算机视觉、自然语言处理、生物特征识别、医学诊断等方面，其中一部分已经有商业应用。比如，声图文识别领域，连接学习就具备明显的优势，综合型专家系统中用到了分析学习的相关知识，而用于诊断类型的专家系统普遍运用了归纳学习的知识[30]。同时，机器学习在理论研究领域也很活跃，机器学习研讨会在国际上每年召开一次，相关的遗传算法会议和计算机学习理论会议也会定期召开。

3.1.1 特征选择

特征选取是机器学习领域的重要问题，众多的特征之中，有的是冗余的，有的是无效的，正确的选取和舍弃特征决定了整个设计算法的精准和效率。为了提高机器学习预测的准确率，一般为训练准备的样本都拥有大量的属性，选择作为特征的属性越多，训练分类模型的时间就越长，开销就越大。随着大样本和高维的问题不断出现，特征选择变得越来越重要。

另一方面，无效的不相关的特征对于机器学习算法本身的准确率也存在影响。已有的研究结果表明，大多数机器学习算法当不相关的特征增多时，需要的训练样本数会急剧增多[31-34]。Langley 等人[33, 34]在论文中指出最近邻法的样本复杂度随不相关特征数加大呈现指数增加。其他归纳算法也和这个情况类似，比如决策树，

尽管逻辑与概念的样本的复杂度对不相关特征不敏感，但是异或概念的样本的复杂度会随着不相关特征增多而急速增加[34]。而贝叶斯分类器，虽然它对于不相关特征的鲁棒性较强，但是它的性能对冗余特征缺失非常敏感[33]。

基于上述原因，特征选择和机器学习算法的效果密切相关。对于如何判断什么样的特征是冗余特征，什么样的特征是典型特征，研究者们提出了各种评价标准，大致上分为以下四类[35]：

①距离度量

以类内和类之间的距离概念作为样本划分的依据，让同类的样本分布密集，而不同类的样本远离。常用的方法，是从样本集中估算出样本均值，类间离散度和类内离散度矩阵，然后计算距离度量函数。另外，特征空间的两类距离也可以通过 Bhattacharyya 距离和 Chernoff 界限表示[36]。

②不一致度

Almuallim 等人[37]在文献中提出了一种新的特征评估标准。定义样本各特征 $f_1,f_2,\ldots,f_t$ 对应了 $nf_1, nf_2,\ldots,nf_t$ 种取值选择，那么每一种取值组合都为一种模式，最多存在 $nf_1*nf_2*\ldots*nf_t$ 种模式。模式相同的情况下，假如两个样本除类别外其他属性都相同，那么这两个样本在这种模式下不一致。对于一个固定的样本集来说，一个模式的不一致的个数等于所有样本个数同最大类别的样本个数之差，不一致度为全部模式下不一致个数的和在样本总数所占的比例。不一致度的特征选取算法，考虑了不同特征值组合对于不同特征子集的组合形式。所以不一致度应用于高维数据，计算代价很高，主要应用于离散特征。如果是连续特征，还需要首先进行离散化。

③基于信息熵的评估标准

信息熵是信息论中的概念，是信息多少的量化衡量指标。当随机变量 X 是离散时，香农熵定义为：

$$H(x)=-\sum_{t}P(x_i)log_2P(x_i) \tag{3.1}$$

已知变量 y 后 x 的条件信息熵公式如下

$$H(x|y)=-\sum_{j}P(y_j)\sum_{i}P(x_i|y_j)log_2P(x_i|y_j) \tag{3.2}$$

基于信息熵的代表性评估标准有信息增益、最小描述长度、互信息等。变量 x 和变量 y 之间的互信息按以下公式计算

$$MI(x,y)=H(x)-H(x|y)=H(y)-H(x|y)=\sum_{x,y}P(x,y)log_2(\frac{P(xy)}{P(x)P(y)}) \tag{3.3}$$

而变量 x 和变量 y 之间的相关性，可以用如下公式(3,4)计算[36]

$$Sim(x,y)=2[\frac{\mathrm{MI}(x,y)}{H(x)+H(y)}] \tag{3.4}$$

得到的相关度 *Sim(x,y)*取值在[0,1]之间，0 表示两个特征不相关，1 表示两个特征完全相关。

④ReliefF 评估

ReliefF 评估是于 1992 提出的一种特征评价方法[39]，其主要思想是好的特征使得相同类型的样本特征值相近，不同类别的样本特征值相差巨大，以此为特征设置权值并排序，然后通过设定特征权值的阀值和特征子集数目进行特征选取。在此基础上，Marko 等人[40]对 ReliefF 评估进行了深入的研究，从概率统计的角度指出，当样本趋于无线多时。特征的权值满足如下逼近概率:

$$W[i]=P(\mathrm{D}_i|\mathrm{T}_d)\text{-}P(\mathrm{D}_i|\mathrm{T}_s) \tag{3.4}$$

其中$W[i]$表示特征 i 的权值，$P(\mathrm{D}_i|\mathrm{T}_d)$表示不同类的最近邻样本之间的特征 i 取值不同的概率，$P(\mathrm{D}_i|\mathrm{T}_s)$表示同类的最近邻样本之间的特征 i 取值不同的概率。

3.1.2 分类算法

分类学习是一种重要的数据挖掘技术,也是机器学习中一个重要的研究领域。分类的目标通过对样本的学习，生成分类模型，然后利用分类模型预测未知样本的类型。以下是对一些常规的分类算法的介绍。

①决策树分类

决策树实质是一种树形结构的归纳学习算法，它以自顶向下的方式进行递归，和流程图比较类似。一个属性对应其中的一个节点，一种属性选择对应其中一条分支，类标号由最下层的树叶节点表示。一条判断规则可以由从根节点出发，到叶节点为止的一条路径表示。

决策树算法最开始始于 E.B.Hunt，J.Marin 和 P.T.Stone 的概念学习系统，J.Ross Quinlan 对它进行了改进，开发了决策树算法，又叫做 ID3(Interative Dichotomiser，迭代的二分器)。此后，Quinlan 进一步研究 ID3 算法，对缺值处理、剪纸技术和派生规则等方面进行了改进，提出了 C4.5 算法，使之成为了新的分类算法新能比较基准。

决策树的构造过程，就是对样本中蕴含的分类规则的理解和提炼的过程。而影响决策树算法效率的关键在于是否能构造一个精度高、规模小的决策树。通常，决策树的构造分为两部分，决策树生成和决策树减枝。其中决策树生成是在训练样本的基础上提取基本的属性规则，决策树的剪枝则是用新的测试数据，把前一步产生的决策树中影响准确性的分枝剪出，进行校验和修正的过程。

在一般情况下，决策树算法具有分类精度高，模式简单的优点，同时，它对噪声数据具有很好的鲁棒性。但是决策树分类算法难以处理大数据集的样本，因为

算法基于深度优先搜索，对内存需求较大。

②贝叶斯分类

贝叶斯算法是一种基于概率统计的分类算法，它发源于古典数学理论，有着坚实的数学基础。实质是通过已知的先验概率来计算未知的后验概率，然后把样本判定于后验概率最大的类别。贝叶斯分类算法的理论基础是贝叶斯定理，它是由18 世纪一位不信奉国教的英国牧师提出的，并以他的名字命名。

假设随机事件 A 和 B，$P(A)$和 $P(B)$是先验概率，在所有样品中该事件发生的可能性，$P(A|B)$是条件概率，表示在事件 B 已经发生的情况下，A 发生的的概率。那么 P($B|A$)就可以表示知道事件 A 已经发生的情况下，事件 B 会发生的概率，叫做后验概率。后验概率 $P(B|A)$它可以通过贝叶斯定理计算得来。贝叶斯定理如下：

$$P(\mathrm{B}\mid A)=\frac{P(A\mid B)P(B)}{P(\mathrm{A})}$$

利用上面的贝叶斯公式，可以计算事件属于各个类标号的后验概率，从而推测出事件最可能属于哪个类别。最基本的贝叶斯分类法是朴素贝叶斯（也叫简单贝叶斯）分类方法，它的前提是假设事件的各属性之间相互独立，每一个属性对于分类的影响独立于其他属性值。于是假定有 m 个类 $C_1,C_2,...,C_m$,给定元组 X，共有 k 个属性特征，那么对于元组 X，它属于类 C_i的后验概率为

$$P(C_i\mid X)=\frac{P(X\mid C_i)P(C_i)}{P(X)}$$

其中，

$$P(X\mid C_i)=\prod_{k=1}^{n}P(X_k\mid C_i)=P(X_1\mid C_i)\times P(X_2\mid C_i)\times\bullet\bullet\bullet\times P(X_n\mid C_i)$$

可以很容易的从训练样本中获得 $P(X_1\mid C_i),P(X_2\mid C_i),\bullet\bullet\bullet,P(X_n\mid C_i)$ 的值，从而计算出由多个属性组成的元组 X 的先验概率 $P(X\mid C_i)$。

在许多情况下，朴素贝叶斯算法与决策树和神经网络算法实现的分类精度比较相近，同时该算法应用到大型数据库，速度快并且分类准确。但是由于朴素贝叶斯分类假定不同属性值之前互相独立，而实际情况中这一假定常常不成立，因此在实际应用中，分类准确率会出现下滑。

③支持向量机

支持向量机（SVM,Support Vector Machine）是由 Vladimir Vapnik 等于 1992 年第一次提出，属于一般化线性分类器，它发源于统计学理论中的 VC 维理论和结构风险最小原理。本质是寻找学习能力和模型精度之间的最佳折衷，以获得良好的泛化能力。简单地说，支持向量机是利用一个非线性的映射，将原始数据映射到高维层次。那么在新的维度，搜索最优分离超平面，寻找属于两个不同类别的数据之间间隔最大的表面。

SVM 对于非线性小样本及高维模式的识别，具备明显的优势。但是也存在一些不足，比如对大规模训练样本难以实施，因为 SVM 是利用二次规划来计算支持向量，在求解二次规划的过程中，会涉及到 m 阶矩阵的计算（m 为样本个数），如果样本数量巨大，矩阵的存储和计算会消耗大量的内存和计算时间。引起效率的低下。同时，支持向量机分类算法经典的应用是二分类问题，在数据挖掘中的实际应用，往往面对的是多分类的情况。

④后向传播分类

后向传播分类属于神经网络学习算法，是神经网络算法中最为流行的一种，它在上世纪 80 年代提出，并迅速获得了声望。后向传播的过程是对训练数据元组进行迭代的处理，然后将每个元组的实际类标号和预测标号相比较，然后调整权重来反映网络预测误差的偏倚，从而向后传播误差。当权重最终收敛的时候，学习过程停止。

由于神经网络算法的一些固有的优点(如算法可以并行计算，对噪音数据的高承载能力以及应对新样本的泛化能力较好等), 后向传播分类广泛应用到现实之中，包括手写字符识别，病理和实验医学多个方面。关于神经网络的相关概念和知识这里不再展开，将在下一节中做详细介绍。

3.2 神经网络概述

在思维学的理论中认为，人类大脑共有三种基本思维方式：灵感思维，抽象思维和形象思维。其中形象思维是将各类在大脑当中分布存储的信息进行综合，产生一个想法或者是解决问题的思路，这种方式的主要特点有两个，首先是信息是以神经元上的兴奋模式分布存储在网络上的，然后利用神经元之间相互作用的动态过程来完成信息处理。

神经网络就是通过计算机来模拟人类的第二种思维方式,形象思维，利用大量的节点模拟神经元，构成相互之间的复杂关系，每一个节点代表一种特定的输出函数（激励函数），用权重来表示节点之间的关系。不同的节点连接方式，不同的权重和激活函数，形成了不同的人工神经网络。

神经网络的构造思路来源于生物神经网络系统的运作方式, 同各门现代科学技术紧密合作，相互促进。具有一些明显的优点，例如良好的非线性拟合能力，能够映射任意复杂的非线性关系，并且学习规则简单，便于计算机实现。具有很强的鲁棒性，非线性能力和较强的自学能力，具有广阔的应用前景。目前用于数据分类的神经网络模型主要有：LVQ 神经网络, RBF 神经网络, SOM 神经网络和 BP 神经网络。

3.2.1 概率神经网络

概率神经网络(probabilistic neural networks,PNN)于 1989 年由 D.F.Spencht 博士首先提出，是一种用于模式分类的神经网络。它主要基于 Parzen 窗的概率密度函数估计方法和贝叶斯分类规则的基础上发展而来，具备训练简单、结构简洁的特点。概率神经网络的应用十分广泛，尤其在分类问题中，PNN 能够用线性学习算法来完成非线性算法完成的工作，同时还具备非线性算法高精度的特点。网络中的权值直接来源于模式样本的分布，节省了大量训练时间，非常适合一些对训练实时性要求高的应用场景。

目前神经网络分类算法的研究主要集中在以 BP 为代表的神经网络算法上，与 BP 网络相比，PNN 具备以下几个优势[41]：

①样本训练简单，收敛速度快。BP 网络和 PNN 网络的输入输出上是相同的，但是它中间的隐藏层单元的选取没有规则，需要根据经验反复试算得出。而 PNN 网络中隐藏层的层数以及各层的神经元个数都是确定的，使用方便。同时，BP 网络的收敛速度慢，容易陷入局部最优值。PNN 网络的训练数据可以从样本中轻松获得，训练时间几乎就等于样本读取时间，训练过程一步到位，并且不存在局部最优值。

②稳定性高，收敛于贝叶斯优化解。BP 网络的分类规则不透明，无法解释，而 PNN 网络的分类原理是基于贝叶斯最小风险准则，可以最大程度的利用样本先验知识，无论问题的复杂程度有多高，只要具备足够多的训练样本，PNN 就能够获得贝叶斯准则下的最优解。而 BP 神经网络可能会出现在一个局部最优解处中断的问题，无法保证获得全局最优值。

③样本容错能力好，追加能力强。PNN 网络在运行过程中，如果有新的训练样本需要添加或删除一些旧的样品，PNN 网络只需要在模式层单元数量上进行调整，然后从新样本总提取数据，修改输入层到模式层的连接权值。而 BP 网络在对训练样本做出修改后，则需要重新进行训练，所有的连接权值都需要重新赋值，等于是需要重新建立整个网络。

3.2.2 神经网络集成

虽然神经网络已经在很多应用领域取得了成功，但是它的使用效果过分依赖使用者的经验，缺少相关的理论知识指导。尽管 Hornik 等人[42]证明了，基于单隐层的前馈神经网络可以以任意精度逼近任意函数，但怎么能找到合适的网络配置和训练却是一个 NP 问题[43, 44]。现实应用下，一般需要反复尝试，找到合适的神经网络模型和参数配置，才能达到理想的应用效果。对于没有神经计算经验的普通技术人员，这种没有工程化的神经网络计算方法，很难保证计算效果。

针对以上问题，Hansen 等人[45]于 1990 年首先提出了神经网络集成的概念。研

究了多个神经网络处理相同的问题，每个神经网络的输出结果集成形成综合结果，证明了神经网络集成可以大大提高神经网络的泛化能力，为神经网络的工程化应用提供了解决思路。同年，Schapire 等人[46]以一个构造性方法从理论上对神经网络集成进行了证明，进一步坚实了理论基础。

基于对神经网络包含的巨大前景和潜力的认同，随后的许多研究者在该领域进行了研究。Krogh 等人[47]于 1995 年对神经网络集成进行了深入分析，指出神经网络集成的泛化误差等于构成集成的个体神经网络的平均泛化误差和各个神经网络之间的平均差异度之差，为如何提高集成神经网络的泛化能力指明了方向。1996 年，Sollich 等人[48]为集成神经网络下了一个被广泛接受的定义，即“神经网络集成是用有限个神经网络对同一个问题进行学习，集成在某输入示例下的输出由构成集成的各神经网络在该示例下的输出共同决定”。

对神经网络进行集成分为两个步骤，第一步生成个体网络，第二步根据一定规则对各个个体网络的输出结果进行合成生成最后的输出结果。在个体网络生成方面，目前最主要的技术有 Bagging[49]和 Boosting[50]两类。Bagging 算法由 Breiman 提出，其主要过程是从原始训练样本中随机的选择若干样本组成一组，然后反复抽取多次，形成若干个训练样本组。在抽取过程中，训练样本允许重复。然后用这些训练样本组训练对应的神经网络学习机，最后集成这些学习机。Bagging 算法通过反复抽取训练样本组成多个训练集的方式，增加了个体神经网络之间的差异性，从而提高神经网络的泛化能力。而 Boosting 算法最早由 Schapire 等[47]提出，Freund[51]在此基础上进行了改进。其核心思想是对训练集中的每一个样本都赋予权重，初始的时候在没有先验知识的情况下，每个样本的权重平均分配，然后每次循环过后提高错误样本的权重，在下一次训练中集中力量对这些错误样本进行判断。

Bagging 和 Boosting 两种神经网络集成方法的主要不同点在于，Bagging 的训练样本抽取是随机进行的。前一轮的训练结构对后一轮训练没有影响，而 Boosting 后面训练样本的选择要受到前面个体网络学习效果的影响。所以 Bagging 的单个预测个体的生成可以并行进行，而 Boosting 只能顺序进行，这使得两者在运行效率上存在差异。

3.3 本章小结

本章首先介绍了机器学习的基本概念，然后对机器学习中的特征选择技术和主要分类算法（决策树分类、贝叶斯分类、支持向量机、后向传播分类）分别进行了介绍。随后介绍了神经网络的基本概念，对其中的概率神经网络和神经网络集成进行了简要的论述。Android 恶意软件静态检测研究的核心，在于针对 Android

系统原理，抽取合适的 Android 软件特征，然后合理利用机器学习、神经网络等相关理论知识进行分析，设计合适的分类算法对 Android 应用软件进行分类。本章论述的相关算法理论是后面的 Android 恶意软件初步检测方案和 Android 恶意软件深入检测方案的理论基础。

4 基于权限相关性的 Android 恶意软件检测算法

Android 软件权限信息通过解压 APK 安装包可以轻松提取, 而权限信息在一定程度上反应了软件自身特征[52]，因此对 Android 权限进行分析可以初步判断软件恶意倾向。本章首先对 Android 权限机制进行了分析，指出 Android 权限和恶意软件之间的关联特点，然后针对 Android 权限机制，设计特征选取方案来节省开销，最后利用改进的朴素贝叶斯分类器对 Android 软件是否有恶意倾向进行检测。

4.1 Android 权限机制分析

权限机制是 Andriod 安全策略的核心，它的实现是通过要求软件开发者在 manifest 文件中声明一系列软件在运行过程中需要用到的权限信息, 用户在安装软件时，同意接受所有权限申请，软件才能正常安装使用。通过这种方式 Android 系统限制了应用软件访问声明之外的或者一些危险的系统资源。Android 系统一共定义了 134 种权限信息[53],并对每种权限的含义做了说明。根据权限的危险级别，又划分为了 4 大类：normal、dangerous、signature 和 signatureorSystem。它们的含义如下表 4.1

表 4.1 Android 四类权限机制含义

Table 4.1 The Explanation of Four Android Permission Systems

类别	含义
normal	默认值。给予申请权限的应用程序访问应用层功能的权利，不会或是很少会给系统、用户或是其他应用程序带来风险。应用程序在安装的时候系统会自动授予这类权限，而不再要求用户批准（当然用户可以在安装之间检测这些权限）
dangerous	较高风险的权限。给予申请权限的应用程序访问私人用户数据或是控制设备的权利，这些操作可能会对用户的利益产生负面影响。因为这个类型的使用权限存在潜在风险，系统不会自动批准这些权限，而是会把权限申请信息显示给用户，让用户来决定是否同意这些权限。
signature	签名权限。只有当申请权限的应用程序的数字签名同声明权限的应用程序的数字签名相同时，系统会自动授予权限，不再询问用户是否允许。
signatureOrSystem	签名或者系统权限。系统只把此类权限授予 Android 系统固件中的应用程序或是同权限声明者有相同数字证书的应用程序。这类权限被用于某些特殊情况，比如厂商在系统固件中内置了一些应用程序，而这些应用程序的功能需要暴露出来共享，以供调用。

虽然 Android 系统声明的较多的权限信息，但是实际上软件开发者真正使用的并不多，只占很少一部分[54]。2012 年，YAJIN ZHOU 等人[55]对正常软件和恶意软件对权限使用情况进行了分析，发现访问网络、读手机状态、访问网络状态、写 SD 卡等权限在恶意软件和正常软件中都广泛使用，但恶意软件倾向于使用短信有关的权限（62.7%）、开机自启动权限（54.6%）、更改 WIFI 状态的权限（31.6%），而良性程序很少使用这些权限。

从以上研究可以看出，不同 Android 权限在使用频率上存在较大差异，并且正常软件和恶意软件在权限使用和类别倾向上也存在较大的不同。而 Android 权限机制中软件的权限信息蕴含了软件固有的特征，并且不同软件之间，正常软件和恶意软件之间在权限信息上是存在差异的，那么从权限的角度去判断一个软件是否是恶意软件具备了理论上的可行性。

4.2 权限特征选取

要快速获得准确的恶意软件分类效果，合适的特征选择非常关键，而 Android 软件的权限信息中存在大量与恶意软件鉴别无关的信息，并且部分权限存在关联度高和信息冗余的情况。那么采取措施对 Android 权限特征进行特征选取就成为非常必要的一部分工作。

4.2.1 基于卡方检验的权限特征过滤

本文选择卡方检验的方法来获得各个权限信息同是否能够区分恶意倾向的相关性，然后根据相关性过滤掉低相关的部分权限特征。

卡方检验是现代统计学的创始人之一 K.pearson 于 1990 年提出的一种具有广泛用途的假设检测方法，分为成组比较和个别比较两类，通过卡方检验公式可以快速获得两个成分的相关性。虽然卡方检验的设计思想中对于样本特征只统计样本中的有无特征，不考虑出现计次数，在次数敏感的应用场景中，相关性的计算会受到影响，存在“低频缺陷”问题。但在 Android 软件的权限声明信息中每条权限信息只会出现一次或者不出现，恰好避开了“低频缺陷”问题，所有该应用场景适用于卡方检验。

卡方检验中基本的四格卡方检验公式如下：

$$\chi^2=\frac{(ad-bc)^2\cdot N}{(a+b)(c+d)(a+c)(b+d)} \tag{4.1}$$

式中 *a,b,c,d* 代表两种构成所形成的四种比较，N 表示总的频数，即 a、b、c、d 之和。本文把是否是恶意软件与是否具备该权限作为两种构成，两两组合，对测试样本统计可得 a、b、c、d。应用上面的公式在样本数据上对 Android 系统 134 个权限分别计算获得其卡方值 χ_i^2，以此作为衡量该权限是否和软件具备恶意软件

倾向弱相关的依据，通过该方式可以初步过滤一些对鉴别恶意软件弱相关的权限特征，即那部分权限与否与该软件是否是恶意软件相关性很低的权限特征。

4.2.2 基于聚类的权限特征去冗余

直观的看，理想的权限特征，应该是和分类结果相关性高，和其他特征不相关或者弱相关。在朴素贝叶斯算法中对于特征属性还存在假设，一个属性对于分类的影响独立于其他属性。观察 Android 权限特征集合，可以发现权限之间存在明显的相关性，很多权限在样本中成组出现或者缺失。比如 SUBSCRIBED_FEEDS_READ,SUBSCRIBED_FEEDS_WRITE 这两个对 RSS 订阅读写的权限就具备明显的强相关性，在几乎所有的样本信息中一旦出现其中的一个，另外一个一定存在。那么把这种强相关的权限同时作为权限特征，就会对软件分类结构造成干扰。为了提高权限特征的代表性，减少分类开销，就还需要在特征集合中去除这部分信息冗余的权限信息。在第 3 章对机器学习中特征选择技术的介绍中，我们知道常规的考察离散特征的相关性有很多方法，比如类间距离、不一致度、信息熵等。其中信息熵是信息理论中的一个重要概念，它能从全局上把握特征之间相关性重要程度。熵的概念最早起源于物理学，用于度量一个热力学系统的无序程度。在信息论里面，熵是对不确定性的测量，主要采用数值形式表达随机变量取值的不确定性程度，目的是刻划信息含量的多少。熵越高，则能传输越多的信息，熵越低，则意味着传输的信息越少。本文设计方案中正常软件和恶意软件都是利用相同的权限信息作为检测依据，因此选用信息熵来衡量特征之间的相关度符合信息熵全局性应用环境。变量 x 的信息熵 H(x)以及已知变量 y 后 x 的条件信息熵公式如下：

$$H(x) = -\sum_{t} P(x_i) log_2 P(x_i) \tag{4.2}$$

$$H(x|y) = -\sum_{j} P(y_j) \sum_{i} P(x_i|y_j) log_2 P(x_i|y_j) \tag{4.3}$$

变量 x 和 y 之间的互信息 MI(x,y)可按以下公式(4.4)计算：

$$MI(x,y) = H(x) - H(x|y) = H(y) - H(x|y) = \sum_{x,y} P(x,y) log_2 (\frac{P(xy)}{P(x)P(y)}) \tag{4.4}$$

采用公式(4.5)来衡量特征x和特征y之间的相关性

$$Sim(x,y) = 2[\frac{\mathrm{MI}(x,y)}{H(x)+H(y)}] \tag{4.5}$$

得到的相关度 *Sim(x,y)*取值在[0,1]之间，0 表示两个特征不相关，1 表示两个特征完全相关。根据公式（4.5）可以计算 Android 权限特征之间的相关性，

然后以此作为依据对 Android 权限特征简单的聚类。把相关性高的权限特征聚合在一起形成一个权限特征簇。算法如下：

输入：权限特征集合T

输出：去冗余后权限特征集合T'

步骤：

①取出 T 中第一个权限作为权限簇 S_1。

②查看集合 T 中其他任意权限与当前权限簇的相关度距离 $Sim(x,y)$,若 $Sim(x,y)$大于阀值 η，则把该权限加入当前权限簇。

③判断集合 T 中是否还有权限没有加入任意权限簇 $\boldsymbol{S}(S_1,S_2,S_3...S_k)$,如果有则建立新的权限簇 S_i，返回②

④遍历权限簇集合 S，如果 S_i 中权限类别超过 3 个，则选取属性权重 W 最大的特征和该簇中与其相关性 $Sim(x,y)$最小的特征共同作为特征权限加入 T' ；如果类别个数小于 3 个，则选取属性权重最大的权限加入 T'。

至此，通过以上两个步骤，权限特征的预处理过程结束。同恶意软件鉴别弱相关的权限特征和冗余性高的权限特征都被去除，形成的新的权限特征集合T'，属性维数较少，并且权限之间相关性低，同恶意软件分类决策相关性高。

4.3 改进的朴素贝叶斯 Android 软件分类

朴素贝叶斯分类的基础是古典数学理论，有着坚实的数学基础，以及稳定的分类效率。同时它所需估计的参数少，对缺失数据敏感性低，算法比较简单。基本思想是通过训练样本获得先验概率，根据先验概率和样本数据信息获得确定事件的后验概率，最后把事件归于后验概率最大的类别。假定有 m 个类 $C_1,C_2,...,C_m$,给定元组 X,那么元组 $\boldsymbol{X}(X_1,X_2,....X_k)$属于类 C_i 的后验假设为

$$P(C_i \mid X) = \frac{P(X \mid C)\ P(C)}{P(X)} \tag{4.6}$$

$P(X)$对于所有类为常数,那么在比较后验概率时只需考虑 $P(X|C_i)P(C_i)$最大即可。而朴素贝叶斯的前提是假设不同属性对于分类的影响相互独立，这样

$$P(X \mid C_i) = \prod_{k=1}^{n} P(X_k \mid C_i) \tag{4.7}$$

在实际应用中，这样的假设往往是不成立的。但是，在经过本文前述的特征选取工作处理后，得到的特征集合T',则较为满足特征属性相互独立，特征代表分类

效果明显的特点，适用朴素贝叶斯分类的应用场景。

结合 4.1 节中对权限机制的论述，进一步考虑，不同权限对于恶意软件决策影响是不同的，为了提高分类的准确率，需要对不同的权限属性做出区分,而标准朴素贝叶斯分类算法中，没有考虑特征间的差异性。本文在 2.1 节中已经利用卡方检验得到了各个权限特征对于分类结果的卡方值 χ_{i}^{2}。以此作为基础数据，把权重的影响因子 β 引入朴素贝叶斯分类后验概率计算中，公式(4.7)变形为:

$$P(C_i \mid X)=\frac{P(C_i)\prod_{\mathrm{k}=1}^{\mathrm{m}} P(X_{\mathrm{k}} \mid C_i)\beta_{\mathrm{k}}}{P(X)} \tag{4.8}$$

其中 $\beta_{\mathrm{k}}=\frac{\chi_{\mathrm{k}}^{2}}{\sum_{i=1}^{m}\chi_{\mathrm{i}}^{2}}$，表示该权限在所有的分类影响中所占的比重。利用公式（4.8）得到待检测软件分别属于正常软件和恶意软件的后验概率，通过比较从而得到最后的检测结果。

4.4 本章小结

本章首先对 Android 安全策略的核心，权限机制进行了介绍和分析，然后针对 Android 权限信息的特点，提出一套从权限相关性角度初步检测恶意软件的方案。对于 Android 权限特征冗余的问题, 提出采用卡方检验计算各权限属性对于分类结果的影响大小，去除冗余权限特征，再对权限属性聚类，提取代表性权限特征，进一步减少冗余。最后，引入不同权限特征的权重，改进朴素贝叶斯算法，进行软件分类。

通过对 Android 权限特征的预处理，减少了恶意软件鉴别的运算开销，引入的权限特征权重，改善了软件分类的准确性。同时因为 Android 软件的权限信息获得相对容易，方案实施起来较为简单，具备一定的恶意软件检测效果，可以作为一种 Android 恶意软件初步检测方案。具体的方案实施细节在第 6 章再作详细的介绍。

5 基于集成概率神经网络的 Android 恶意软件检测算法

5.1 引言

上一章介绍了利用较易获取的 Android 软件权限信息进行恶意软件初步检测，虽然可以获得一定的检测准确率，但是由于 Android 软件权限自身的一些特点，比如开发者往往倾向于申请比软件实际需要更多的权限信息[56, 57]，同时权限申请集合只是 Android 软件一个较为表层的信息，还不足以完全体现 Android 软件的特征，对于 Android 恶意软件的检测，还需要更加深入软件内部进行分析。另一方面，基于特征的检测方法的实质是利用已有的特征规律去匹配待检测样本，这种方法适用于检测已知的 Android 恶意软件，对于新出现的在软件特征上存在一些差异的恶意软件，检测准确率会出现下降的情况。

针对以上两个问题，本章提出通过逆向解析 Android 软件安装包，获取代码级的高危 API 作为软件特征，反复抽取高危 API 特征形成多组样本组向量。同时引入概率神经网络，利用其鲁棒性强，自学能力好等特点作为分类器，对恶意软件检测这一问题进行多个概率神经网络的集成学习，待测样本的判断结果由集成的概率神经网络共同决定，从而提高 Android 恶意软件检测的泛化能力，缓解检测未知新样本的恶意软件时出现的检测准确率下降的问题。

本章主要分为三个部分，第一部分介绍 Android 高危 API 的提取，以及这个过程中需要的 Dalvik 语言背景知识，。第二部分介绍如何利用集成概率神经网络对 Android 恶意软件进行检测，形成检测方案。最后，对本章的内容进行了梳理和总结。

5.2 Android 高危 API 提取

Android API 位于 Android 体系结构中从下往上的第三层，是一些预先定义好的函数，目的是提供软件开发者在不需要理解内部实现的情况下，实现对底层软件或者硬件的控制。 Android API 就是一套可供调用的系统接口，它为上层丰富的软件应用提供了框架支撑。上层的应用软件利用 API 实现了各种各样的软件行为，而其中就包含一些访问电话本、发送短信、获得网络等高危行为。软件中包含这些高危行为的 API 较好的反应了 Android 应用的真实行为，可以作为衡量一个软件是否有恶意倾向的可靠依据。

5.2.1 Dalvik 汇编语言基础

在 2.1.3 节中，对 Android 程序运行的基础，Dalvik 虚拟机进行了简要的介绍，Dalvik 虚拟机专门设计了一套指令集，并且制定了自己的指令格式与调用规范。

我们将 Dalvik 指令集组成的代码称为 Dalvik 汇编语言。Dalvik 汇编语言主要由指令集与字节码组成。

Dalvik 字节码只有两种类型，基本类型与引用类型。Dalvik 使用这两种类型来表示 Java 语言的全部类型，除了对象与数组属于引用对象外，其他的 Java 类型都是基本类型。类型描述符如下表 5.1

表 5.1 类型描述符

Table 5.1 The Description Symbol of Types

语法	含义
V	Void,只用于返回值类型
Z	boolean
B	byte
S	short
C	char
I	int
J	long
F	float
D	Double
L	Java 类类型
[	数组类型

Dalvik 虚拟机是基于寄存器的，而每个寄存器都是 32 位大小，对于小于或等于 32 位长度的类型来说，一个寄存器就可以存放该类型的值。而像 J、D 等 64 位的类型，它们的值是使用相邻两个寄存器来存储。

Dalvik 方法的表现形式比类名要复杂一些，Dalvik 使用方法名、类型参数与返回值来详细描述一个方法。这样一方面有助于 Dalvik 虚拟机在运行时从方法表中快速的找到正确的方法，另一方面，Dalvik 虚拟机也可以使用它们来做一些静态分析，比如 Dalvik 字节码的验证与优化。具体的方法格式如下：

Method(I[[IILjava/lang/String;[Ljava/lang/Object;)Ljava/lang/String;

按照上面的描述，将其转换成 Java 形式的格式应该为：

String method(int,int[][],int,String,Object[])

Dalvik 字段与方法相似，只是字段没有方法签名域中的参数与返回值，取而代之的是字段的类型。同样 Dalvik 虚拟机定位字段与字节码静态分析时会用到它。字段的格式如下：

Lpackage/name/ObjectName;->FieldName:Ljava/lang/String;

字段由类型（Lpackage/name/ObjectName;）、字段名（FieldName）与字段类型（Ljava/lang/String;）组成。其中字段名与字段类型用冒号“：”隔开。

Dalvik 的指令集在调用格式上模仿了 C 语言的调用约定。包含的指令一共有十四种，具体如下表 5.2

表 5.2 Dalvik 指令列表

Table 5.2 Dalvik Instruction List

名称	助记符	示例	含义
空操作指令	nop	nop	对齐代码，无意义
数据操作指	move	move vA,vB	将vB寄存器的值赋给vA寄存器
返回指令	return	Return-void	函数从一个void方法返回
数据定义指	const	const/4 vA,#+B	将数字符号扩展为32位后赋给
锁指令	monitor-enter,monitor-	monitor-enter vAA	为指定的对象获取锁
实际操作指令	check-cast,new-instance等	new-instance vAA, type@BBBB	构造一个指定类型的新实例，并将对象引用赋值给vAA寄存
数组操作指令	array-length,new-array等	array-length vA,vB	获取vB寄存器中数组的长度赋值给vA寄存器
异常指令	throw	thorw vAA	抛出vAA寄存器中指定类型的
跳转指令	goto	goto+AA	无条件跳转到指定偏倚处
比较指令	cmpkind	cmpkind	比较vBB和vCC寄存器，结果
字段操作指	iinstanceop,sstati	iinstanceop vA,vB	操作普通字段vA和vB
方法调用指	invoke	invoke-virtual	调用实例的虚方法
数据转换指	unop	unop vA,vB	把一种类型的数值vA转换成vB
数据运算指令	binop	binop vAA,vBB,vCC	vBB寄存器与vCC寄存器进行运算，结果保存到vAA寄存器

5.2.2 逆向提取高危 API

Android 软件安装包是 APK 格式的文件，它的实质是 ZIP 格式的压缩文件。解压缩之后，APK 文件的结构如下图 5.1

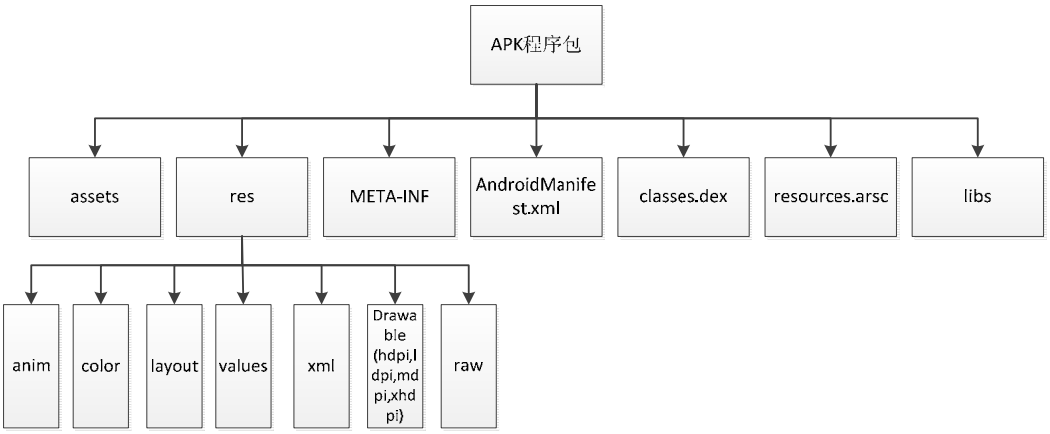

图 5.1 APK 文件结构图

Fig 5.1 APK File Structure

其中 assets 目录中存放的是一些配置文件，这些文件的内容在程序运行过程中可以通过相关 API 获得。Res 是资源文件夹，存放软件的图片、字符串等，里面包含的 anim 文件夹存放的是预置的动画对象。META-INF 目录下存放的是签名信息，用来保证 apk 包的完整性和系统的安全。AndroidManifest.xml 是程序全局配置文件，里面包含应用的名字、版本、权限、引用的库文件等等信息。resources.arsc 是编译后的二进制资源文件。libs 如果存在的话,存放的是 ndk 编译出来的 so 库。

classes.dex 是比较重要的一部分，它是 Dalvik 虚拟机上执行的字节码，它由应用程序的所有 Java 源码编译生成 class 文件，然后由 class 文件转换而成。它包含了所有的源码信息，通过对 classes.dex 进行逆向解析，即可获得以 smali 为后缀的文件，通过在 5.1.1 节中介绍的 Dalvik 汇编语言基础，可以识别提取源代码中用到的 API 方法。其中涉及到网络、通信、本地信息、地理位置、系统安全等行为的 API 方法，即是高危 API。查阅 Android 官网的 API 信息[58]，统计其中具备典型高危行为的 API，共计 41 个。如下表 5.3

表 5.3 Android 高危 API

Table 5.3 Android High-Risk API

包类	方法	作用
adroid.app.PendingIntent	send	延时触发
[illegible]	getAccounts	获取谷歌账号
android.app.ActivityManager	killBackgroundProcess	杀进程
android.app.AlarmManager	Set	延时触发
android.content.BroadcastReceiver	abortBroadcast	拦截短信
android.content.ContentResolver	query	遍历

android.content.ContentResolver	update	篡改
android.content.ContentResolver	insert	插入
android.content.ContentResolver	delete	删除
android.content.Intent	setDataAndType	安装
android.content.Intent	startActivity	邮件
android.content.Intent	getAction	监听广播
android.content.pm.PacakageManager	getInstallerPackageName	获取软件信息
android.content.pm.PackageManager	removePackageFromPrefe	卸载软件
android.content.pm.PackageManager	getInstalledPackages	获取软件信息
android.content.pm.PackageManager	getInstalledApplications	获取软件信息
android.database.sqlite.SQLiteDatabase	execSQL	数据库相关
android.location.LocationManager	getLastKnownLocation	获取地理位置
android.media.MediaRecorder	MediaRecorder	录音
android.telephoney.gsm.SmsManager	sendDataMessage	发送彩信
android.telephoney.gsm.SmsManager	sendMultipartTextMessage	发送短彩信
android.telephoney.gsm.SmsManager	sendTextMessage	发送彩信
android.telephoney.SmsManager	sendDataMessage	发送彩信
android.telephoney.SmsManager	sendMultipartTextMessage	发送短彩信
android.telephoney.SmsManager	sendTextMessage	发送短信
android.telephony.gsm.SmsMessage	getDisplayOriginatingAddress	读取短信
android.telephony.gsm.SmsMessage	getDisplayMessageBody	读取短信
android.telephony.PhoneStateListener	onCallStateChanged	监听手机状态
android.telephony.TelephonyManager	getLine1Number	获取手机号
android.telephony.TelephonyManager	getDeviceId	获取 IMEI
android.telephony.TelephonyManager	getSubscriberId	获取 IMSI
android.telephony.TelephonyManager	getCellLocation	获取地理位置
dalvik.system.DexClassLoader	loadClass	动态加载
dalvik.system.PathClassLoader	loadClass	动态加载
java.lang.Runtime	exec	执行脚本
java.net.HttpURLConnection	connect	联网
java.net.URLConnection	connect	联网
javax.crypto.Cipher	getInstance	加密解密
javax.crypto.Cipher	Init	加密解密
javax.crypto.Cipher	doFinal	加密解密
org.apache.http.impl.client	DefaultHttpClient	联网

值得注意的是，随着 Android 系统的发展，越来越多的软件嵌入了广告插件，支付插件，安全插件，辅助插件等快速增强其功能。这类插件普遍具备联网和信息获取等功能，调用了高危 API。如果直接纳入恶意特征统计，会干扰对正常软件恶意倾向的判断，所以在真正分析样本之前，需要排除一些安全插件的影响。为此，建立白名单机制在查找高危 API 过程中，跳过白名单中 smali 文件，去除插件对软件鉴别的影响。下面是收集的安全的插件白名单，如下表 5.4

表 5.4 Android 安全插件白名单

Table 5.4 The White List of Android Security Plug-In

包名	来源
backport.android	google code上的一个蓝牙项目
cn.com.android	安致论坛插件
com.admob	移动的广告插件
com.adwhirl	广告插件
com.eoemobile	易联致远插件
com.facebook	facebook插件
com.flurry	统计插件
com.google	google的各种插件
com.greystripe	移动广告插件
com.mappn	机锋插件
com.millennialmedia	广告插件
com.mobclick	友盟统计插件
com.mobclix	移动广告插件
com.netqin	网秦插件
com.openfeint	社交游戏平台插件
com.papaya	通用游戏引擎
com.paypal	paypal插件
com.scoreloop	社交游戏平台插件
com.socialin	adturns.com网站的广告插件
com.tapjoy	游戏插件，含有广告
com.tencent	QQ插件
com.umpay	联动优势支付插件
com.wooboo	广告插件
engine.scoreloop	游戏平台
net.youmi	广告插件
org.meteroid	Java ME的Android插件

去除插件干扰后，在剩下的 Smali 文件中搜索高危 API，对每个 Android 软件

统计结果，如果没有该项高危 API,则用 0 表示，如果有就记录次数，最终形成 Android 软件的高危特征向量，整个流程如下图 5.2。

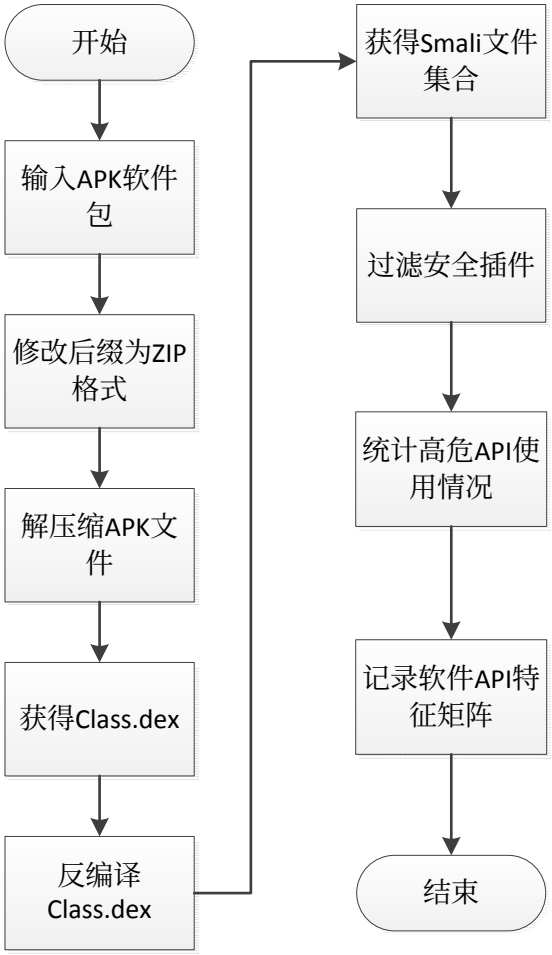

图 5.2 Android 软件高危特征提取过程

Fig 5.2 The Fetch Process of Android High-Risk Feature

在收集的 Android 正常软件和恶意软件样本上批量进行上面的操作，最后对获得特征向量进行汇总，形成特征矩阵。通过辨别 Android 正常软件和恶意软件在高危 API 上使用情况的不同，体现其软件行为的差异性，从而鉴别 Android 恶意软件。

5.3 基于集成概率神经网络的 Android 恶意软件分类

在章节 3.2 中, 我们对概率神经网络以及神经网络集成技术进行了基本的介绍。概率神经网络具备神经网络固有的学习能力好，样本数据适应性强，模拟人脑功能具备智能性等优点，同时它同传统的研究较多的 BP 神经网络相比，又具备训练迅速、网络结构清晰、调节参数少、分类能力强等优点。在面对训练样本更新的

情况时，概率神经网络只需要增加或是减少相应的模式层单元，不用重新建立整个网络，这种特性使它能够很好的用于应对不断更新扩大的 Android 软件样本，加强学习能力，跟随 Android 软件发展趋势。

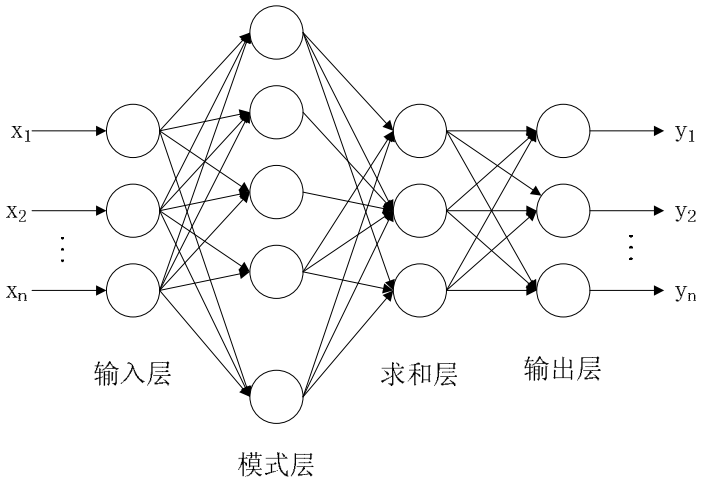

图 5.3 概率神经网络结构

Fig 5.3 The Structure of Probabilistic Neural Network

概率神经网络的基本结构由四层组成，分别是输入层、模式层、求和层和输出层，如图 5.3。输入层的神经元个数等于样本特征的维数，它接受来自训练样本的高危 API 特征向量。模式层计算输入 API 向量同训练集中各个样本之间的匹配关系，模式层的神经元个数等于 Android 训练样本总和，该层每个模式单元的输出为

$$f(X,W_i)=\exp[-\frac{(X-W)^{\mathrm{T}}(X-W_i)}{2\delta^2}] \qquad (5.1)$$

式中，W_i 为输入层到模式层连接的权值；δ 为平滑因子，它对分类起着至关重要的作用。第 3 层是求和层，求和层神经元的的个数等于分类类别个数，这里只需要把软件分为正常软件和恶意软件两类，所以个数为 2。在这一层中，模式层分别按照公式(5.1)计算获得正常软件和恶意软件类别的概率密度函数获得输出，然后累加。求和层的每一个神经元只与属于自己类的模式层单元相连，和模式层其他神经元没有关系。第 4 层，输出层的作用是对求和层的结果进行归一化处理，最后输出各类的概率估计。输出层是由简单的阀值辨别器构成，在正常软件和恶意软件的估计概率密度中选择一个具有最大后验概率密度的神经元作为整个神经元的输出。

在集成神经网络领域，分为 Bagging 算法和 Boosting 算法两类，Baging 算法的训练过程主要是重复抽样，形成多个训练样本集，而 Boosting 算法的主要思想是在每轮训练中加强对上一轮训练中错误的学习，进行多轮训练。这是两种不同的

算法思想, Bagging 算法可以并行计算, 和 Boosting 算法需要串行计算, 因此 Bagging 算法效率相对较好。

为了提高 Android 恶意软件检测泛化能力和分类准确率，同时考虑检测算法的运行效率，本文把概率神经网络和集成学习两种思想结合在一起，以 Bagging 的方式在上一节获得的 Android 软件高危 API 特征集合上进行重复抽样，获得多个特征样本，然后分别用概率神经网络进行训练，得到概率神经网络分类器。最后经过对测试样本的预测获得分类结果，利用多个概率神经网络投票决定最终输出。完整的恶意软件鉴别流程如下图 5.4。

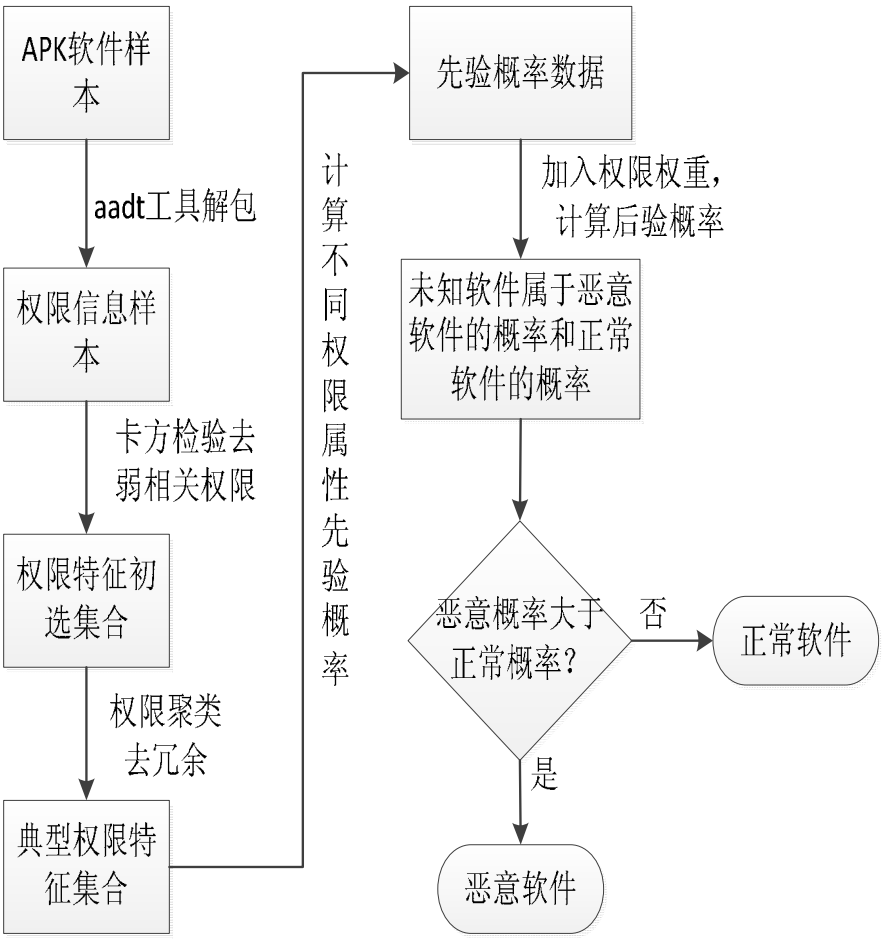

图 5.4 基于集成概率神经网络的 Android 恶意软件分类

Fig 5.4 The Classification of Android Malware Based on Integration Probabilistic Neural Network

同前面的高危 API 特征提取结合起来，整个基于集成概率神经网络的 Android 恶意软件深入检测方案描述如下：

输入：Android 软件训练样本集 *N*，待测试样本 *m*，样本重复抽样次数 *n*

输出：测试样本 m 的类别（正常，恶意）

算法：

(1) for $I = 1$ to N

(2) 从训练样本集 N,抽取一个 APK 软件包，修改文件后缀为 zip 格式。

(3) 解压缩 APK 软件包，获得 Class.dex 文件。

(4) 反编译 Class.dex。获得 Smali 文件集合。

(5) 过滤安全插件的 Smali 文件。

(6) 查找高危 API 使用，并记录。

(7) 加入 Android 软件样本高危 API 特征矩阵 T_o

(8) end for

(9) 形成 Android 软件训练样本集 N 对应的特征矩阵 T_o

(10) 从特征矩阵 T 中有放回的抽取多行特征值，形成 n 份样本，创建 API 特征样本集 T_{io}

(11) 在每个 API 特征样本集 T_i 上用概率神经网络进行训练，形成分类器。

(12) 对待测试 Android 样本 m 在 n 个概率神经网络分类器上进行预测。

(13) 分类信息投票集成，预测类型统计数量最多的一类即为 Android 软件鉴别结果。

5.4 本章小结

本章深入研究了集成概率神经网络在 Android 恶意软件检测中的应用，具体的阐述了如何利用逆向技术，反编译 APK 软件包，提取其中的高危 API 调用作为软件特征，从而贴近软件的真实行为。然后利用概率神经网络鲁棒性强、学习能力好、训练简单的特点作为分类器进行分类鉴别。同时针对 Android 恶意软件静态检测中普遍出现的检测软件特征有一定差异的新样本时出现的检测准确率下降的问题，提出利用 Bagging 集成学习技术对多个概率神经网络进行集成，提高 Android 恶意软件静态检测的泛化能力。最终形成一套基于集成概率神经网络的 Android 恶意软件深入检测方案，获得较好的检测效果，同时在应对新出现的 Android 恶意软件时，检测准确率不会出现大幅降低。

6 实验和结果分析

本章首先简要介绍了实验相关工具, 然后对前两章提出的基于权限相关性的检测方案和基于集成概率神经网络的检测方案进行了实验验证和结果分析。实验环境为宏基笔记本 5552G(AMD Phenom(tm)II N970 四核处理器，4G 内存，500G 机械硬盘)，Matlab 2011b 平台。

6.1 实验相关工具介绍

6.1.1 Python 脚本语言

Python 是一种面向对象、解释型的编程语言。它于 1989 年由 Guido van Rossum 发明，1991 年发布了第一个公开发行版本。Python 语言的语法简单，又有丰富的类库，可以很容易地使用其他语言的模块，因此有胶水语言的称号。

本文使用 Python3.3.2 版本对 APK 文件进行逆向分解，提取其中的权限特征和高危 API 特征, 其中反编译 Dex 文件过程, 通过 Python 调用 Java 中的 baksmali Jar 包实现。

6.1.2 Baksmali 工具包

Baksmali 是使用最广泛的反编译工具，它是 Google Code 上的一个开源项目，第一个版本于 2011 年 11 月发布，并持续更新当中。它以 Java Jar 包的形式提供使用，提供对 APK 中 class.dex 文件解析功能。通过命令行对 baksmali 的调用，可以把 Dalvik 虚拟机上执行的 classes.dex 文件，反编译成 smali 格式的字节码。

本文主要使用 baksmali 解析 classes.dex 文件，获取 Android 软件对应的 smali 文件包，然后根据 smali 语法在文件包中查找统计，高危 API 是否出现以及出现的次数。

6.2 基于权限相关性检测方案的实验及结果分析

基于权限相关性的 Android 恶意软件检测方案，在收集的 Android 正常软件和恶意软件样本上进行了实验，以正常软件预测准确率、恶意软件漏检率以及软件预测总体准确率三个指标对实验结果进行衡量，并同 Chun-Ying Huang 等人[59]基于所有权限信息以及附加软件信息的检测方法的实验效果进行了对比。

6.2.1 实验设计

本方案首先从国外的一个共性病毒库 VirusShare[60]收集了 11080 个恶意 Android 软件样本, 然后从网上下载了截至 2013 年打包的 1247 个国内常用 Android 软件集合，组成软件样本库。

根据第 4 章中对 Android 权限相关性的恶意软件初步检测方案描述，首先利用 Android-SDK 工具包下的 aadt(Android Asset Packaging Tool)工具，解包 APK 格式软件，抽取其中的权限信息，aapt 命令运用如下图 6.1，得到的权限信息。随机选择了 1000 个 Android 恶意软件和 1000 个 Android 正常软件作为 2013 年实验样本，利用 Python 编制脚本批量完成所有样本的权限提取，并写入到 txt 文件中备用。

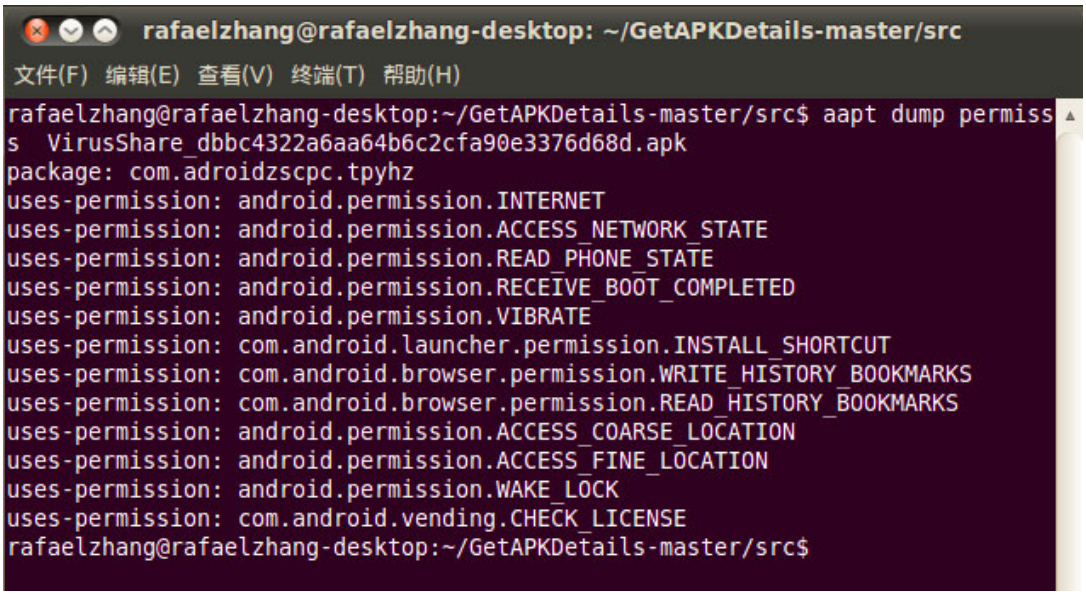

图 6.1 aapt 工具提取 Android 安装包权限信息

Fig 6.1 The Extraction of Android APK Permission by aapt

然后开始特征预处理工作，包括抽取关键权限特征和去除冗余权限属性。利用卡方检验公式，分别计算 134 个权限信息对于鉴别软件是否是恶意软件的影响。计算后卡方统计量分布如下图 6.2

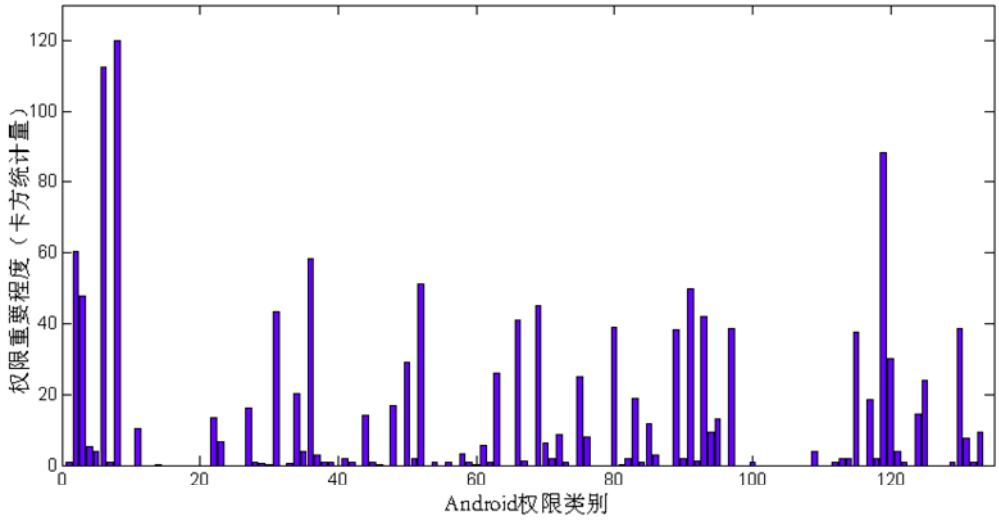

图 6.2 Android 权限重要程度分布

Fig. 6.2 The Distribution Diagram of the Importance of Android Permissions

从上图可以看出，Android 的 134 个权限对于鉴别软件是否是恶意软件的影响程度存在较大差异。图中相当部分权限出现了没有对应蓝条的情况，是因为这部分权限在 1000 个正常软件样本和 1000 个恶意软件样本中都没有出现。这也印证了文献[51]中指出的 Android 系统声明的权限信息有很大部分很少使用的情况。因此在考虑权限特征选取的时候，我们就可以去除对这部分极少使用的权限特征的考虑，同时对于一些经常出现，但是与软件是否是恶意程序相关程度极低，无论是否恶意软件都经常出现的权限特征，也可以排除考虑。这样权限特征得到了初步的降维。

接着是针对剩余权限之间的部分权限相关性过高的问题，存在冗余的问题做出处理，也即是通过聚类的方式去冗余。依据章节 4.2.2 中的简单聚类算法，其中确定合适的权限聚类的参数，权限相关度阀值 η 对于特征选取非常关键。η 过大，聚类过度，形成的权限簇无法在权限含义上进行理解，η 过小，聚类不够，达不到简化特征的目的。经过反复试验，当 η=0.4 时，得到以下成组的强相关权限簇，如表 6.1：

表 6.1 Android 强相关权限簇

Table 6.1 Strong Correlation Permission Clusters of Android

序号	权限簇
1	ACCESS_COARSE_LOCATION,ACCESS_FINE_LOCATION
2	ACCESS_NETWORK_STATE,ACCESS_WIFI_STATE
3	BLUETOOTH，BLUETOOTH_ADMIN
4	CLEAR_APP_CACHE,GET_PACKAGE_SIZE
5	AUTHENTICATE_ACCOUNTS,GET_ACCOUNTS,MANAGE_ACCOUNTS
6	READ_SMS，RECEIVE_SMS，SEND_SMS
7	GET_ACCOUNTS,MANAGE_ACCOUNTS,USE_CREDENTIALS
8	SUBSCRIBED_FEEDS_READ,SUBSCRIBED_FEEDS_WRITE
9	READ_SYNC_SETTINGS,WRITE_SYNC_SETTINGS

以 Android 权限含义作为背景知识理解，上表 6.1 分组结果恰好把功能接近相关性高的权限归为一组，同时又没有过渡聚类，遗漏重要权限特征，符合预期。经过两轮特征预处理。利用 4.2.2 节的聚类算法处理过后，权限特征从 134 个减少

到 68 个。

接下来，进行样本学习，从软件样本库中，随机选出 500 个正常软件和 500 个恶意软件作为学习样本得出训练数据，计算贝叶斯先验概率。最后利用带权重的后验概率公式(4.8)分别计算其属于正常软件和恶意软件可能性的参数值, 通过比较得到软件检测结果。整个实验流程如图 6.3:

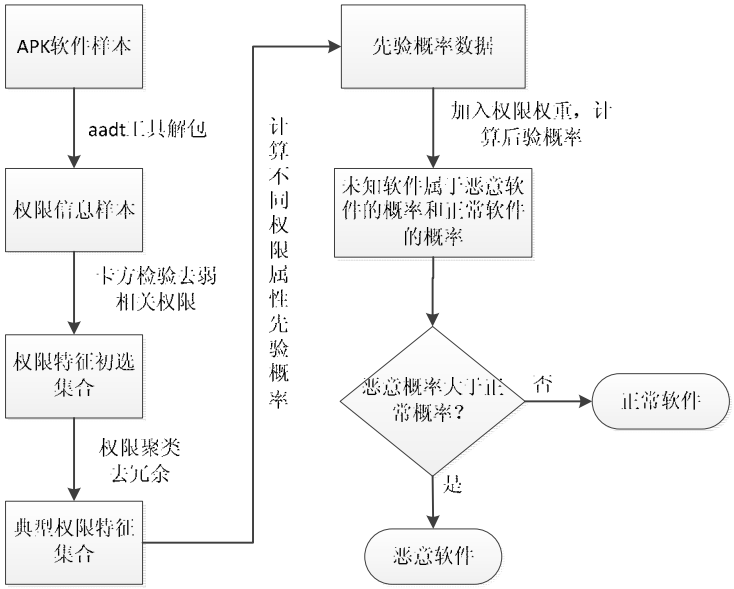

图 6.3 基于权限相关性的恶意软件初步检测流程图

Fig. 6.3 The Flowchart of Preliminary Detection Based on Correlation of Permissions

最后是测试环节，本文以正常软件 50 个和恶意软件 50 个组合在一起为一组，在样本中随机抽取 6 组数据参与测试。另一方面对权限特征不进行过滤处理，保留 134 个全部权限对测试样本直接进行预测，记录两种方式的实验结果。为了量化实验结果, 现定义一组相关指标,从各方面衡量实验效果。设 TP 表示正常软件预测准确数量，FP 表示正常软件预测错误数量，TN 表示恶意软件预测准确数量；FN 表示恶意软件预测错误数量。

那么定义 $TPR=\frac{TP}{TP+FN}$，表示正常软件在预测为正常的软件中所占比例。$FNR=\frac{FN}{TN+FN}$，表示恶意软件漏检率。$ACC=\frac{TP+TN}{TP+TN+FP+FN}$，表示所有软件预测准确的比例。本文方案的检测结果将通过上述指标进行量化验证。

6.2.2 结果分析

统计实验中 6 组测试样本在权限特征抽取集合上的检测结果，如下表 6.2：

表 6.2 基于权限相关性的初步检测方案预测结果

Table 6.2 The Predicting Outcomes of Preliminary Detection Based on Correlation of Permissions

序号	TP	FP	TN	FN	TPR/%	FNR/%	ACC/%
1	41	9	43	7	85.41	14.00	84.00
2	40	10	43	7	85.10	14.00	83.00
3	43	7	42	8	84.31	16.00	85.00
4	39	11	44	6	86.67	12.00	83.00
5	42	8	43	7	85.71	14.00	85.00
6	43	7	41	9	82.69	18.00	84.00

典型权限特征集合上的 6 组实验结果平均计算, 总体预测准确率 ACC 为 84%, 恶意软件漏检率为 FNR14.67%，正常软件判断准确率 TPR 为 84.99%。6 组测试样本在所有权限特征集合上的检测结果如下表 6.3。平均计算总体预测准确率 ACC 为 84%，恶意软件漏检率为 FNR14.33%，正常软件判断准确率 TPR 为 85.22%。

表 6.3 基于全部权限的初步检测方案预测结果

Table 6.3 The Predicting Outcomes of Preliminary Detection Based on All Permissions

序号	TP	FP	TN	FN	TPR/%	FNR/%	ACC/%
1	42	8	43	7	85.71	14.00	85.00
2	40	10	44	6	86.96	12.00	84.00
3	42	8	42	8	84.00	16.00	84.00
4	39	11	44	6	86.67	12.00	83.00
5	41	9	42	8	83.67	16.00	85.00
6	43	7	42	8	84.31	16.00	84.00

而全文献[59]同样是以 Android 权限信息为核心特征来初步检测恶意软件，不同在于它把所有权限信息都纳入了特征考虑，同时还获取了部分比较容易获的得软件特点（比如 ELF 文件个数，SO 文件个数等）一起作为特征。为了验证本方案的实验效果，本文与它的实验结果进行了比对，文献[59]中用到的软件特征一共有 296 个，本方案用到的特征只有 68 个，文献[59]中的方案通过大量实验，统计出 Android 恶意软件中有 81%的样本能够被预测出来,而本文中的恶意软件漏检率 FNR 平均为 14.67%, 即是有 85.33%的恶意软件能够检测出来, 检测效果和文献[59]相当，并有一定提升。具体对比如下表 6.4：

表 6.4 基于权限的初步检测方案检测效果对比

Table 6.4 The Detection Effect of Preliminary Detection Based on Correlation of Permissions

方案	特征数/个	恶意软件检测比例/%
文献[59]	296	81
全部权限特征方案	134	85.67
本文方案	68	85.33

（恶意软件检测比例和漏检率之和为 1）

以上实验结果表明，本文基于权限相关性的 Android 恶意软件初步检测方案在引入权重因子区分权限对于鉴别恶意软件的影响程度之后，检测准确率有了一定的提升。而基于特征选取的检测方案和全部权限特征的检测方案相比，所需特征数进一步下降，节省了开销，准确率仍然保持在了十分接近的水平。

本文方案和文献[59]相比，检测效果保持在了相同的水平段，并有一定的提高，而所需特征更少，证明了检测方案的可行性。然而 Android 开发者普遍倾向于申请比软件真正需要的更多的软件权限信息[56, 57]，所以以 Android 权限信息作为 Android 软件的特征，还不能完全代表 Android 软件的行为特点。那么从权限的角度进行检测恶意软件的思路，应该主要目标是定位于提供一个初步的检测判断，如果要进一步提高 Android 恶意软件检测的准确率，还有待后续的深入检测。

6.3 基于集成概率神经网络方案的实验及结果分析

基于集成概率神经网络的检测方案同样以正常软件预测准确率、恶意软件漏检率以及软件预测总体准确率三个指标进行衡量。在之前所收集的 Android 软件样本的基础上，加入 2014 年最新的 Android 正常软件和恶意软件组成新的样本库，测试检测准确率。

6.3.1 实验设计

深入检测方案沿用之前的初步检测方案中使用的 VirusShare 和 2013 年 Android 常用软件打包作为训练样本。第一步仍然是样本特征的提取，这里我们以 5.1 节中定义的 41 个 Android 高危 API 特征作为提取对象，用 Python 语言编写脚本批量解压 APK 文件，获得其中的 classes.dex 文件，修改名称为各自的软件名。主要 python 代码如下图 6.4

```
files = os.listdir(root)
for filename in files:
    li = os.path.splitext(filename)
    if li[1] == ".apk":
        newname = "%s\%s"%(root,li[0]+".zip")
```

```
newDexname = "%s\%s"%(root,li[0]+".dex")
newDexname = "".join(newDexname.split())
newname = "".join(newname.split())
os.rename("%s\%s"%(root,filename),newname)
#去除错误的 APK 安装包
try:
    zipFile = zipfile.ZipFile(newname)
except:
    continue
zipFile = zipfile.ZipFile(newname)
for file in zipFile.namelist():
    if file== "classes.dex":
        zipFile.extract(file,root)
        os.rename("%s\%s"%(root,"classes.dex"),newDexname)
zipFile.close()
```

经过这一步骤，获得了 Android 软件的 dex 字节码文件，然后就是对 Dex 文件进行逆向编译，这里用到 6.1.2 节中提到的 Baksmali 工具。其基本的命令行用法是“java -jar baksmali-2.0b4.jar –o /输出路径 文件名”,用 Python OS 包中的命令行指令循环调用，即可获得 Android 软件的 Smali 文件集合，如下图 6.4

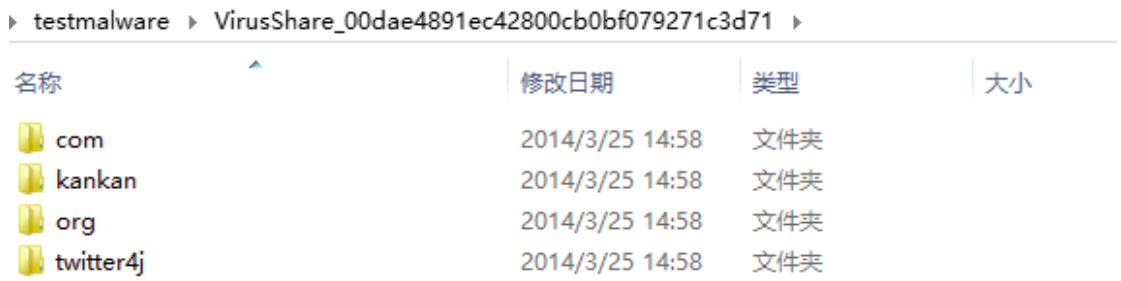

图 6.4 Android 软件逆向 Smali 文件集合示例

Fig. 6.4 The Example of Android Inverse Smali Documents

这里文件夹路径和代码的包名路径是一样的，值得注意的是，在搜索其中的 smali 文件之前，还需要去除 Android 插件对于软件特性的干扰。在 5.1.2 节中罗列了收集的 Android 插件白名单，在实验过程中，对于这部分插件对应的文件路径下的 smali 文件直接忽略，不计入特征数量统计之中。最后形成的 Android 恶意软件高危特征如下图 6.5

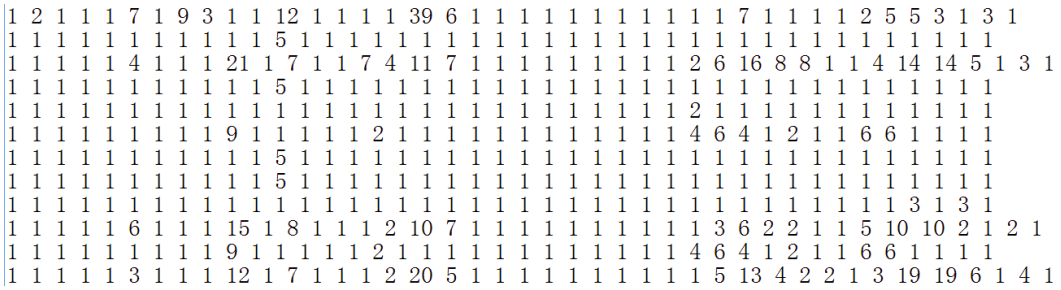

图 6.5 Android 软件高危 API 特征样本示例

Fig. 6.5 The Example of Android High-Risk API Features

到此，特征提取工作结束。下面是利用获得的高危 API 特征进行概率神经网络的训练。利用 matlab 提供的 newpnn()函数创建概率神经网络，Sim()函数进行网络预测。其中有一个较为关键的参数 spread 扩展系数，默认值为 1.0。spread 的值越大，计算越困难，输出的结果越光滑。应该保证 spread 的值足够大，从而使得概率神经网络对于输入的高危特征向量都产生响应。本实验中，spread 取值 1.5 时，实验效果达到较为理想的情况。核心代码如下：

```
%% 将期望转换为类别
t_train=ind2vec(t_train);
t_train_temp=Train(:,41)';
%% 使用 newwpnn()函数建立 pnn，spread 选取为 1.5,并统计训练时间
Spread=1.5;
tic
net=newpnn(p_train,t_train,Spread)
toc
%% Sim 函数预测未知数据并效果展示
Y=sim(net,p_test);
Yc=vec2ind(Y);
figure(1)
stem(1:length(Yc),Yc,'b^')
hold on
stem(1:length(Yc),t_test,'r*')
title('Android 恶意软件预测效果')
xlabel('预测样本标号')
ylabel('检测结果')
set(gca,'Ytick',[1:2])
```

以上是单个概率神经网络的实验预测，为了在应对新的Android样本时，维持较好的准确率，本文在此基础上，对多个概率神经网络进行了 bagging 集成，训练多个概率神经网络模型来同时预测，待检测 Android 软件由多个概率神经网络预测结果投票选出，具体算法依照 5.3 节中的伪代码实现。训练样本沿用上一节中的500个正常软件和500个恶意软件，默认进行十次可重复样本抽样，形成十个同样大小的训练子样本，训练对应的概率神经网络参与集成。由于本文是以 Bagging 方式进行的集成，多个概率神经网络的训练过程是可以并行进行节省开销。而 Matlab 平台本身就能支持并行计算，所以在代码中开启并行功能，进行并行计算，核心代码如下：

```
%初始化 Matlab 并行计算环境
CoreNum=4;
%设定机器 CPU 核心数量，实验机器为四核 AMD Phenom(tm)II N970 处理器，所以 CoreNum=4
if matlabpool('size')<=0
%判断并行计算环境是否已然启动
    matlabpool('open','local',CoreNum);
%若尚未启动，则启动并行环境
else
    disp('并行环境已经启动');
end
%并行训练十个 PNN 网络,并记录时间
n = 10;
net = cell(n,1)
tic
parfor x= 1:n;
    net{i} = newpnn(p_train(i),t_train(i),Spread)
end
toc
```

实验所需的训练样本沿用之前采集的 13 年 Android 软件训练样本（1000 正常，1000 恶意）。整个实验方案的完整流程如下图 6.6:

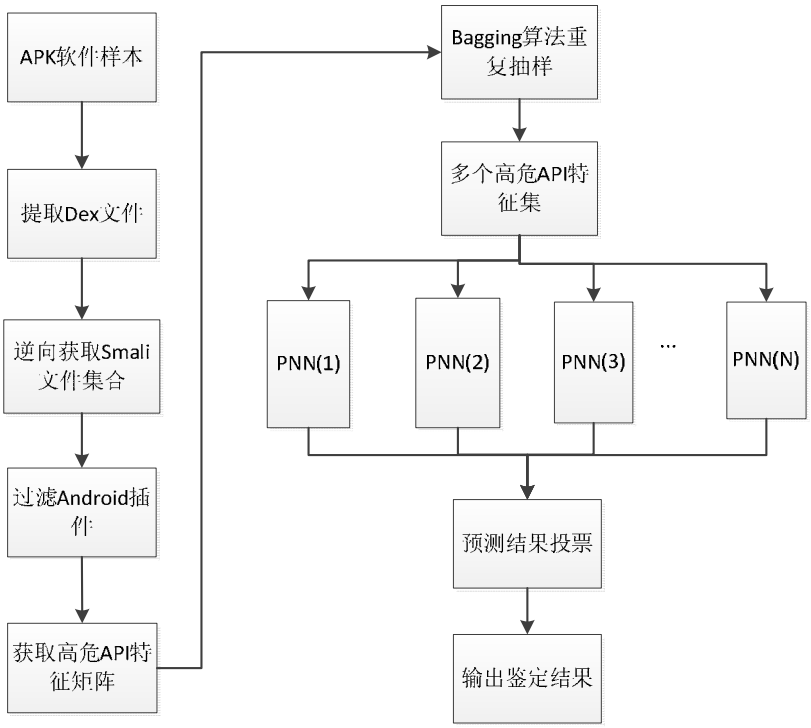

图 6.6 Android 恶意软件深入检测方案流程

Fig. 6.6 The Process of Android Malware Deeply Detection Scheme

预测阶段，首先从软件样本中随机抽取 50 个恶意软件，50 个正常软件组合为一组，共计六组，进行预测，统计基础实验效果。

接着，为了验证集成概率神经网络在检测新样本时，对泛化能力的影响。从 360 手机助手中下载最新的 300 个装机必备的正常软件，同时重新从 VirusShare 下载了 2014 年最新的 300 个恶意软件集合在一起，作为最新的 2014 年 Android 软件样本。为了区别，实验中后续统一以之前 2013 年的 Android 软件样本为旧样本，2014 年 Android 软件样本为新样本。

然后，以 50 个正常软件和 50 个恶意软件为一组，从中随机选出 4 组数据参与预测。先用在旧样本上训练的单独 PNN 分类器进行预测，再用同样在旧样本集合上训练而成的集成 PNN 分类器进行预测。对比预测效果，分别观察在旧样本作为训练集检测 14 年新样本时，检测准确率是否下降。

最后，把 2014 年的 600 个新样本加入到之前 2013 年旧样本中，重新训练形成新的集成概率神经网络分类器，对上一步从新样本选出的 4 组数据进行预测，统计预测结果，验证泛化能力是否有改善。

实验完毕，统计以上各种情况下的平均检测指标，集中对比分析。

6.3.2 结果分析

基于集成概率神经网络的 Android 恶意软件检测方案，在 2013 年旧样本上，

检测旧样本时，基础实验结果如下图 6.7

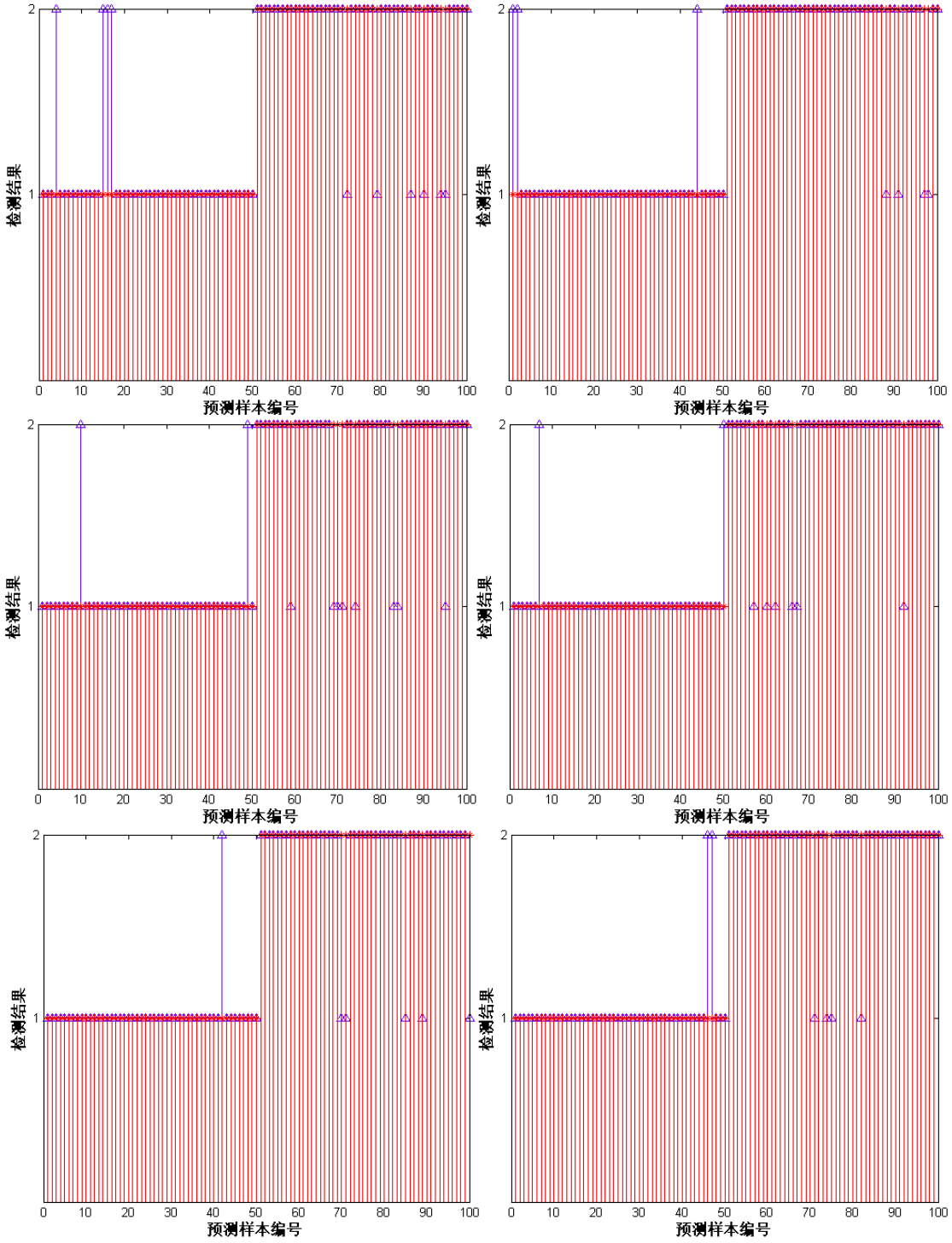

图 6.7 基于集成概率神经网络检测方案基础实验结果

Fig. 6.7 The Outcome of Detection Scheme Based on Integration Probabilistic Neural Network

上图中，纵坐标中的 1 代表恶意软件，2 代表正常软件。*代表实际值，△代表预测值。从图上可以看出整体检测效果较为理想，特别是对于前半部分的恶意

软件，出现漏检的情况比较少见。统计 6 组实验数据，得出总体预测准确率 ACC 为92.17%，恶意软件漏检率FNR为4.67%。正常软件判断准确率TPR为95.05%。三项关键指标相比于之前的初步检测方案都有了大幅度的提高，基本检测效果维持在了一个较高的水准。

应对 2014 年新样本时的，统计三种不同方案的平均指标，如下表 6.5：

表 6.5 新样本检测指标对比

Table 6.5 The Comparison of New Sample Detection Index

方案编号	分类器	学习对象	训练时间/ms	TPR/%	FNR/%	ACC/%
1	单独 PNN	旧样本	97.75	93.07	6.50	90.50
2	集成 PNN	旧样本	222.31	94.13	5.50	91.00
3	集成 PNN	新旧混合	237.59	94.68	5.00	91.75

从表中数据可以看出，三种不同的方案在检测 2014 年新样本时，相对于预测旧样本的基础检测效果（TPR:95.05%，FNR:4.67%,ACC:92.17%），检测准确率都出现了不同程度的下降。方案 2 和方案 1 相比，分类器为多个集成的概率神经网络，检测准确率更高，证明了以 Bagging 方式对概率神经网络进行集成，可以缓解检测新样本时，检测准确率下降的问题。同时训练时间上，由于 Bagging 方式支持并行计算，十个概率神经网络集成的训练时间并不需用单独 PNN 训练时间的十倍，只是约 2.27 倍，算法效率较为理想。如果在更多处理核心的设备上运行，并行效果还会进一步提高。

方案3和方案2相比，在训练样本上，加入了对2014年新样本的学习，检测准确率进一步提高，逼近预测旧样本时的基础检测效果，达到了较高的检测水平。而基于概率神经网络对于新样本的良好追加能力，训练样本在原有 1000 个扩大到 1600 个的情况下，训练时间只是出现了极小幅度的提高。

横向对比 3 个方案和原有旧样本的基础检测效果，验证应对新样本检测时的泛华能力。如下图 6.8

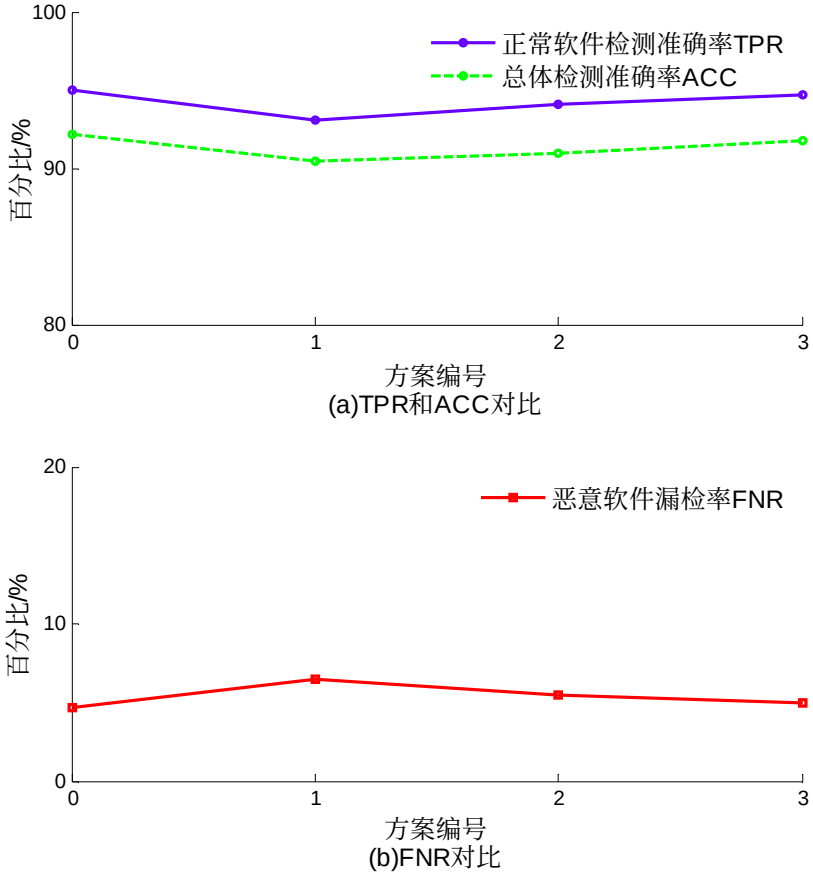

图 6.8 泛化能力关键指标对比

Fig. 6.8 The Comparison of Key Index for Generalization Ability

检测新样本时，检测指标出现了一定的下滑。但是利用多个 PNN 集成检测时候，检测效果有了一定提高。而加入最新样本进行学习后，开销的时间小幅增加，但是检测效果进一步提升，泛化能力得到进一步改善。证明了本文提出的基于集成概率神经网络的 Android 恶意软件静态检测方案可以在很小额外开销的情况下，及时追踪 Android 恶意软件发展趋势，提高应对新型 Android 恶意软件检测时的泛化能力。

6.4 本章小结

本章主要对前面提出的基于 Android 权限相关性的恶意软件初步检测方案和基于集成概率神经网络的 Android 恶意软件深入检测方案进行实验验证。用到的实验样本来自于 VirusShare 共享数据库、360 手机助手以及 2013 年常用 Android 软件打包集合，根据时间的先后，把样本分为了新样本和旧样本两部分。最终的实验结果说明：

1、基于权限的 Android 恶意软件的初步检测方案中对于权限的处理方法，和

Android 系统权限模块的自身设计背景相符合。与同样定位初步检测的文献[59]相比，需求的特征数由文献[59]的 296 个下降到 68 个，效率上得到了改善，同时能够检测出85.33%的Android恶意软件，相对于文献[59]的81%，检测精度有了一定的提高。

2、基于集成概率神经网络的 Android 恶意软件深入检测方案，以逆向编译获取的高危 API 特征作为软件特征，贴近软件的真实行为。利用概率神经网络对软件特征集合进行学习，训练迅速，可以在开销很小的情况下及时追踪 Android 恶意软件发展趋势，更新分类模型。应用于当前样本，总体预测准确率 ACC 为 92.17%，恶意软件漏检率仅为 FNR 为 4.67%，面对新样本时，总体预测准确率 ACC 为 91.75%，恶意软件漏检率为 5.00%，保持在了较高的水准。

7 总结和展望

本部分主要对全文的工作进行梳理和总结, 并对文中所设计方案存在的问题以及今后的改进方向进行阐述。

7.1 本文工作总结

Android 系统是当前最为流行的智能手机系统，随着它的快速发展和普及，Android 系统上丰富的软件应用给人们的生活带来了巨大的便利。同时受巨大的经济利益驱使使得 Android 平台成了恶意软件泛滥的地方，对于 Android 恶意软件检测和防范成为了亟需解决的问题。

而 Android 作为新兴事物，对于它自身特点的研究还并不多，如何利用传统领域相关理论知识同 Android 平台的背景相结合是一个值得研究的方向。对于 Android 恶意软件检测，本文针对 Android 系统的特点提出了基于权限的 Android 恶意初步检测方案和基于高危 API 调用的深入检测方案。

初步检测方案以较易获得的 Android 权限作为特征，通过分析 Android 权限的特点，去除冗余权限信息，然后利用不同权限对于鉴别软件的影响程度作为权重，改进朴素贝叶斯分类器进行分类, 以形成一个初步的 Android 恶意软件初步检测判断。而深入检测方案，则是以逆向解析 Android APK 安装包，获取高危 API 调用作为检测特征，把机器学习中集成学习方法同概率神经网络知识相结合形成多个神经网络分类器，综合预测 Android 软件恶意倾向。

以 Virusshare，360 手机助手以及网上收集软件作为样本来源，进行了实验。实验结果表明初步检测方案能够检测出 85.33%的恶意软件，总体预测准确率为 84%。深入检测方案能够检测出 95.33%的恶意软件，总体预测准确率为 92.17%，在面对新的检测样本时，及时追踪样本变化趋势，恶意软件检测率和总体预测准确率分别只下降了 0.33 和 0.42 个百分点。

7.2 未来工作展望

本论文提出的 Android 初步检测方案和深入检测方案，实验效果达到预期，具备了一定的恶意软件检测能力。但是要投入实际的运用仍然有许多工作要做。主要包括以下两个方面：

1、Android 软件的特征信息很多，本文仅仅是选用了具有代表性 Android 权限信息以及高危 API 调用作为软件特征。而其他一些软件内在信息比如敏感字符串、文件大小、ELF 文件等以及 Android 软件类别、软件商城评分等外在信息，没有纳

入考虑。如果对这部分相关特征所蕴含的信息进行融合和分析，对于鉴别 Android 恶意软件，尤其是初步检测应该会有一定的提高和改进。

2、本文主要对静态检测方案的一些现有问题进行了研究，而要获得更好的检测效果，单纯的静态检测或者动态检测还不够。静态检测和动态检测方案需要合理的结合起来，形成全面的综合检测方案，这是后续继续研究和改进的一个方向。

致　　谢

首先，感谢我的导师，杨吉云副教授。在我论文选题、实验和撰写整个过程中，都得到了杨老师悉心的关怀和无私的帮助。杨老师果断干练的作风和宽厚善良的处事方式，渊博的学识和严谨的治学态度，永远值得我学习和效仿。

感谢我的父母，感谢他们对我生活无微不至的关怀和照顾。他们的鼓励和支持，给予了我面对困难，战胜困难的勇气。

感谢研究生三年所有传授我知识的老师们，您们的谆谆教导，牢记于心，受益终身。

感谢身边的朋友，欧阳源游，王小恒。是你们陪我一起在操场跑圈，在挥汗如雨中体会坚持的意义。

感谢 3201 实验室与我一起度过无数春夏的同学们。那些日子我们一起调程序、做实验、写论文，这些时光都将是最美好的回忆。

最后，衷心感谢那些默默支持过我的朋友，在这里请接收我最真挚的谢意。

时光冲冲，三年的学习生活，转瞬即过。回首这段只叹起短暂匆促的学习生活，里面充满的苦乐酸甜，而这些回忆中的每一点每一滴都会留在我心间，成为宝贵的人生经历，转化为今后治学路上的信心和力量。

张 锐

二 O 一四年四月于重庆

参考文献

[1] IDC.Worldwide Smartphone Shipments Top One Billion Units for the First Time[EB/OL]. http://www.idc.com/getdoc.jsp?containerId=prUS24645514.2014-01-27/2014-03-10.

[2] Mawston N.Android Captures 79 Percent Share of Global Smartphone Shipments in2013[EB/OL].http://www.prnewswire.com/news-releases/strategy-analytics-android-captures-79-percent-share-of-global-smartphone-shipments-in-2013-242563381.html. 2014-01-19/2014-03-10.

[3] 中国互联网协会. 恶意软件定义 [EB/OL]. http://www.isc.org.cn/zxzx/xhdt/listinfo-1390.html.2006-11-22/2014-03-10.

[4] La Polla M, Martinelli F, Sgandurra D. A survey on security for mobile devices [J]. Communications Surveys & Tutorials, IEEE, 2013, 15(1): 446-471.

[5] Kaspersky Lab. IT Threat Evolution [EB/OL]. http://www.securelist.com/en/analysis/204792312/IT_Threat_Evolution_Q3_2013.2013-11-14/ 2014-03-10.

[6] 网秦.2013 第一季度全球手机安全报告[EB/OL]. http://cn.nq.com/neirong/2013Q1.pdf.2013-07-01/2014-03-10.

[7] Mohite S. A Survey on mobile malware: War without end [J]. International Journal of Computer Science and Business Informatics, 2014, 9(1):23-35.

[8] Egele M, Scholte T, Kirda E, et al. A survey on automated dynamic malware-analysis techniques and tools [J]. ACM Computing Surveys (CSUR), 2012, 44(2): 6.

[9] Yan Q, Li Y, Li T, et al. Insights into malware detection and prevention on mobile phones [M]. Security Technology. Springer. 2009: 242-249.

[10] Chandramohan M, Tan H B K. Detection of mobile malware in the wild [J]. Computer, 2012, 45(9): 65-71.

[11] 陆登. 自动化测试在大型软件系统的应用与研究[D]; 浙江大学, 2010.

[12] 谢红霞, 吴红梅. 基于 Android 的自动化测试的设计与实现[J]. 计算机时代, 2012, 5(2): 20-22.

[13] 文志. Google Android 程序设计指南[M]. 电子工业出版社, 2009.

[14] 岩靳, 尚朗. Google Android 开发入门与实战[M]. 人民邮电出版社, 2009.

[15] 丰生强. Android 软件安全与逆向分析[M].人民邮电出版社，2013.

[16] Portokalidis G, Homburg P, Anagnostakis K, et al. Paranoid Android: versatile protection for smartphones[C]. proceedings of the Proceedings of the 26th Annual Computer Security Applications Conference, 2010: 347-356.

[17] Schultz M G, Eskin E, Zadok E, et al. Data mining methods for detection of new malicious executables[C]. proceedings of the Security and Privacy, 2001 S&P 2001 Proceedings 2001 IEEE Symposium on, 2001:38-49.

[18] Devesa J, Santos I, Cantero X, et al. Automatic Behaviour-based Analysis and Classification System for Malware Detection[C]. proceedings of the ICEIS (2), 2010: 395-399.

[19] Santos I, Nieves J, Bringas P G. Semi-supervised learning for unknown malware detection; proceedings of the International Symposium on Distributed Computing and Artificial Intelligence, 2011: 415-422.

[20] Santos I, Laorden C, Bringas P G. Collective Classification for Unknown Malware Detection[C]. proceedings of the SECRYPT, 2011:251-256.

[21] Santos I, Brezo F, Ugarte-Pedrero X, et al. Opcode sequences as representation of executables for data-mining-based unknown malware detection [J]. Information Sciences, 2013, 231 :64-82.

[22] Schmidt A-D, Bye R, Schmidt H-G, et al. Static analysis of executables for collaborative malware detection on android; proceedings of the Communications[C]. 2009 ICC'09 IEEE International Conference on, 2009:1-5.

[23] Burguera I, Zurutuza U, Nadjm-Tehrani S. Crowdroid: behavior-based malware detection system for android[C]. proceedings of the Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, 2011:15-26.

[24] Chiang H-S, Tsaur W. Mobile malware behavioral analysis and preventive strategy using ontology[C]. proceedings of the Social Computing (SocialCom), 2010 IEEE Second International Conference on, 2010:1080-1085.

[25] Enck W, Ongtang M, McDaniel P. On lightweight mobile phone application certification[C]. proceedings of the Proceedings of the 16th ACM conference on Computer and communications security, 2009:235-245.

[26] Shabtai A. Malware Detection on Mobile Devices[C]. proceedings of the Mobile Data Management, 2010:289-290.

[27] Dai S, Liu Y, Wang T, et al. Behavior-based malware detection on mobile phone[C].proceedings of the Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on, 2010:1-4.

[28] Spinellis D. Reliable identification of bounded-length viruses is NP-complete [J]. Information Theory, IEEE Transactions on, 2003, 49(1): 280-284.

[29] Mitchell T M. Machine learning[J]. Burr Ridge, IL: McGraw Hill, 1997, 45(3).

[30] 王珏, 周志华, 周傲英.机器学习及其应用[M]. 清华大学出版社有限公司, 2006.

[31] Xing E P, Jordan M I, Karp R M. Feature selection for high-dimensional genomic microarray data[C]. proceedings of the ICML, F, 2001:601-608.

[32] Jain A, Zongker D. Feature selection: Evaluation, application, and small sample performance [J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1997, 19(2): 153-158.

[33] Langley P, Iba W. Average-case analysis of a nearest neighbor algorithm[C] proceedings of the IJCAI, 1993:889-894.

[34] Langley P. Selection of relevant features in machine learning [M]. Defense Technical Information Center, 1994.

[35] 张丽新, 王家钦, 赵雁南, et al. 机器学习中的特征选择 [J]. 计算机科学,2004,31(11):180-184.

[36] Liu H, Motoda H. Feature selection for knowledge discovery and data mining [M]. Springer, 1998.

[37] Almuallim H, Dietterich T G. Learning with Many Irrelevant Features[C].proceedings of the AAAI, 1991:547-552.

[38] Yu L, Liu H. Efficient feature selection via analysis of relevance and redundancy [J]. The Journal of Machine Learning Research, 2004, 5(8):1205-1224.

[39] Kira K, Rendell L A. The feature selection problem: Traditional methods and a new algorithm[C]. proceedings of the AAAI,1992:129-134.

[40] Robnik-Šikonja M, Kononenko I. Comprehensible interpretation of relief's estimates[C]. proceedings of the Machine Learning: Proceedings of the Eighteenth International Conference on Machine Learning (ICML'2001), Williamstown, MA, USA San Francisco: Morgan Kaufmann, F, 2001:430-440.

[41] 王小川. Matlab 神经网络 43 个经典案例分析.北京航空航天大学出版社.2013

[42] Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators [J]. Neural networks, 1989, 2(5): 359-366.

[43] Baum E B, Haussler D. What size net gives valid generalization [J]. Neural computation, 1989, 1(1): 151-160.

[44] Judd S. Learning in networks is hard[C].proceedings of the Proceedings of the First International Conference on Neural Networks, 1987:685-692.

[45] Hansen L K, Salamon P. Neural network ensembles [J]. IEEE transactions on pattern analysis and machine intelligence, 1990, 12(993-1001).

[46] Kearns M J, Schapire R E. Efficient distribution-free learning of probabilistic concepts[C].proceedings of the Foundations of Computer Science, 1990 Proceedings, 31st Annual Symposium on, 1990:382-391.

[47] Krogh A, Vedelsby J. Neural network ensembles, cross validation, and active learning [J]. Advances in neural information processing systems, 1995, 231-238.

[48] Krogh P S A. Learning with ensembles: How over-fitting can be useful [J]. Advances in neural information processing systems, 1996, 8(190).

[49] Breiman L. Bagging predictors [J]. Machine learning, 1996, 24(2): 123-140.

[50] Schapire R E. The strength of weak learnability [J]. Machine learning, 1990, 5(2): 197-227.

[51] Freund Y. Boosting a weak learning algorithm by majority [J]. Information and computation, 1995, 121(2): 256-285.

[52]Fang Z, Han W, Li Y. Permission Based Android Security: Issues and Countermeasures [J]. Computers & Security, 2014

[53] Google. Manifest permission[EB/OL]. http:// Android.com/reference/Android/Manifest.permission.html.2014-03-10/2014-03-10.

[54] Barrera D, Kayacik H G, van Oorschot P C, et al. A methodology for empirical analysis of permission-based security models and its application to android[C].proceedings of the Proceedings of the 17th ACM conference on Computer and communications security, F, 2010:73-84.

[55] Zhou Y, Jiang X. Dissecting android malware: Characterization and evolution. Security and Privacy[C]. proceedings of the IEEE Symposium on, 2012:1063-1069.

[56] Johnson R, Wang Z, Gagnon C, et al. Analysis of Android Applications' Permissions[C]. proceedings of the Software Security and Reliability Companion (SERE-C), 2012 IEEE Sixth International Conference on, 2012:45-46.

[57] Felt A P, Chin E, Hanna S, et al. Android permissions demystified[C]; proceedings of the Proceedings of the 18th ACM conference on Computer and communications security, F, 2011:627-638.

[58] Google. android API [EB/OL]. 2014, http:// http://developer.android.com/reference/packages.html.2014-03-10/2014-03-10.

[59] Huang C-Y, Tsai Y-T, Hsu C-H. Performance Evaluation on Permission-Based Detection for Android Malware [M]. Advances in Intelligent Systems and Applications-Volume 2. Springer. 2013: 111-120.

[60] VIRUSSHARE.COM. Because Sharing is Carring[DB/OL]. http://virusshare.com/support.2014-03-10/2014-03-10.

附　　录

A. 作者在攻读学位期间发表的论文目录

[1] 张锐，杨吉云. 基于权限相关性的 Android 恶意软件检测研究.计算机应用. [已发表]

[2] Jiyun Yang, Rui Zhang, Di Xiao. Analysis and Improvement of an Efficient and Secure Key Agreement Protocol. The 9th International Conference on Computational Intelligence and Security. [Published]

B. 作者在攻读学位期间取得的科研成果目录

[1] 明祥公司物流管理集成系统。项目编号：1042012920120114

[2] 重庆市城投路桥管理有限公司数字化管理中心软件开发。
项目编号：1042012920120861