浙江工业大学

硕士学位论文

论文题目：基于 Dalvik 指令特征的 Android 恶意应用检测方法研究

作者姓名 杨益敏

指导教师 陈铁明 教授

学科专业 计算机科学与技术（学术）

所在学院 计算机科学与技术学院

提交日期 2017 年月日

浙江工业大学硕士学位论文

基于 Dalvik 指令特征的 Android 恶意应用检测方法研究

作者姓名：杨益敏

指导教师：陈铁明教授

浙江工业大学计算机科学与技术学院

2017 年 3 月

Dissertation Submitted to Zhejiang University of Technology for the Degree of Master

An Android Malware Detection Method Based on the feature of Dalvik Instruction

Candidate: Yang Yimin

Advisor: Chen Tieming

College of Computer Science and Technology
Zhejiang University of Technology
June 2017

浙江工业大学

学位论文原创性声明

本人郑重声明：所提交的学位论文是本人在导师的指导下，独立进行研究工作所取得的研究成果。除文中已经加以标注引用的内容外，本论文不包含其他个人或集体已经发表或撰写过的研究成果，也不含为获得浙江工业大学或其它教育机构的学位证书而使用过的材料。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人承担本声明的法律责任。

作者签名：　　日期：年月日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权浙江工业大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

作者签名：　　日期：年月日

导师签名：　　日期：年月日

基于 Dalvik 指令特征的 Android 恶意应用检测方法研究

摘要

随着移动互联网与 Android 应用的普及，Android 恶意应用程序正呈现持续的高速增长趋势。为了能够更加有效地检测移动恶意程序，本文结合 Dalvik 指令特征与机器学习等多种算法，提出了一种新颖的轻量级的静态检测方法。

首先，将 Android 应用程序中的 DEX 文件进行反汇编，并用一个事先定义好的指令符号集对汇编后的指令代码进行简化与抽象。然后，再对简化与抽象后的指令符号进行 N-Gram 编码，被 N-Gram 编码的指令符号序列将作为 Android 应用程序的特征，通过机器学习的分类算法训练样本数据集的特征，创建一个恶意代码检测模型。此外，为了进一步优化该恶意代码检测模型的检测效率，还对高维数据进行水平压缩和对海量样本进行垂直压缩。基于公开的 Drebin 数据库，本文提出的方法与专业反病毒软件就检测恶意代码的能力进行对比实验，实验结果表明本文的方法能够有效地进行 Android 恶意代码检测，并获得较高的检测率。

另外，针对 Android 恶意家族分类的问题，本文提出了一种基于 bytecode 生成图的 Android 恶意代码家族分类方法。首先，将 Android 恶意应用程序的二进制 byte code 转化为 256 阶灰度图形式的 bytecode 生成图，然后利用 GIST 算法提取图像中的纹理特征。最后，通过 Random Forest 算法对特征进行分类与建模。基于公开的 Drebin 数据库，本文对最为常见的 14 种 Android 恶意代码家族的样本进行了实验验证，并与 Drebin 方法进行比较，实验结果表明本文的方法可有效进行 Android 恶意代码家族分类，并获得较高的检测率。本文的主要工作和成果如下：

（1）研究并简化了 Dalvik 的官方指令集，用指令符号的方法抽象同一类型的指令操作码，并提出了一种基于 N-Gram 序列的指令符号特征。

（2）针对抽象后的 Dalvik 指令符号的 N-Gram 序列特征，利用 AP 聚类算法对样本进行压缩和利用信息增益方法进行特征筛选，并结合多种分类算法进行模型训练与验证，有效实现了一种 Android 恶意代码的检测模型，与常见反病毒软件进行实验相比较，表明有较高的检测率。

（3）本文提出了一种可快速实现 bytecode 生成图的方法，并深入研究了 DEX 文件结

构与 bytecode 生成图纹理之间的对应关系。

（4）结合 GIST 算法与 Random Forest 分类算法，对 bytecode 生成图的纹理特征进行了提取和建模，进而实现了 Android 平台的恶意代码家族分类，实验表明该方法检测精度高且误报率较低。

（5）设计并实现了 Android 恶意代码智能检测平台，该平台可以满足在线 APK 分析服务的应用场景。

关键字：恶意代码，安卓，Dalvik 指令，N-Gram，机器学习，图像纹理

AN ANDROID MALWARE DETECTION METHOD BASED ON THE FEATURE OF DALVIK INSTRUCTION

ABSTRACT

With the popularity of Mobile Internet and Android applications, the Android malware has a continuous high growth trend. In order to detect Android malware more effectively, a novel lightweight static detection model, which is proposed using the feature of Dalvik Instruction and machine learning techniques in this paper.

Firstly, a symbol set based simplification method is proposed to abstract the OpCode sequence decompiled from Android DEX files. Then, N-Gram is employed to extract features from the simplified Opcode sequence, and a classifier is trained for the malware detection and classification tasks. To improve the efficiency and scalability of the proposed detection model, a compressed procedure is also used to reduce features and select exemplars for the malware sample dataset. Our method is compared against the state-of-the-art anti-virus tools in real-world using Drebin dataset. The experimental results show that our method can get a higher accuracy rate and lower false alarm rate with satisfactory efficiency.

In addition, to solve the problem of malware family classification, this paper proposes an Android malware family classification method based on the image of bytecode. A bytecode file of Android malware is converted to a 256-level grayscale image and texture features are extracted from the image by GIST. The random forest algorithm is applied to classify the extracted features. We have verified the method by the experimental data of 14 kinds of common Android malware families and compared it against the Drebin on the same dataset. The experimental results show that our method has high detection precision and lower false positive rate. The main work of this paper is as follows:

(1) A simple symbol set is introduced to simplify the original Dalvik instructions sequence, where one series of instructions with similar function can be assigned as one symbol. N-Gram technique is employed to handle the symbolic sequence.

(2) In order to achieve high efficiency of detection, a further reduced scheme is proposed to largely cut down the number of N-Gram items and training samples. Specifically, information gain is employed for attribute reduction, and affinity propagation for sample selection. Our method is compared against the state-of-the-art anti-virus tools in real-world and the results show that our method can get a higher accuracy rate.

(3) This paper proposes a fast method of creating the bytecode image and has a research on the relation the structure of DEX files with the texture of bytecode image.

(4) Texture features are extracted from the image by GIST and the random forest algorithm is applied to classify the extracted features. We have compared our method against the Drebin on the same dataset. The experimental results show that our method has high detection precision and lower false positive rate.

(5) An Android malware detection platform is designed and implemented, which is appropriate to use in the analysis service of APKs scenarios.

Key Words：malware, Android, Dalvik instruction, N-Gram, machine learning, image texture

目　　录

第 1 章　绪论

1.1　研究背景

1.1.1　恶意程序高速增长现状

随着移动互联网的快速发展，以及各种移动智能终端变得越来越普及，移动应用程序不管从种类上还是数量上都呈现高速增长的趋势，智能手机现在已经成为全球网民最常用的一种访问互联网的设备。一份来自 Gartner 统计数据表明，仅 2015 年第 4 季度全球智能手机的销售量就已经高达 4 亿多台，在手机操作系统占有率排行榜中，Android 系统占据首位，占有率高达 80.7%[1]。截止到 2016 年 2 月 1 日，仅 Google Play 应用市场上的 Android 应用数量就接近 200 万[2]。于此同时，Android 的应用安全如今也正面临着两大主要安全威胁：应用漏洞与恶意应用。Android 应用程序开发者一般都缺少基本的代码审计与代码安全规范检查的能力，所以往往在软件生命周期中的各个阶段中会产生移动安全漏洞，这些漏洞一旦被恶意的攻击者利用，将会对应用软件的完整性、保密性和可用性造成巨大的损害[3]。一般来说，Android 恶意应用是指那些在热门的 Android 移动应用中插入一段恶意的攻击代码，并将这些恶意应用在安全管理比较薄弱的第三方 Android 应用商店发布与传播[4]。阿里聚安全公布的 2015 移动安全病毒年报表明，18%的 Android 设备曾经感染过病毒，而将近 95%的热门移动应用存在相应的仿冒应用，而且正呈现出恶意应用种类越来越多的趋势[5]。最为常见的恶意移动应用的恶意行为包括个人信息窃取、恶意扣费和短信劫持等，这些行为将严重损害智能手机用户的切身利益，其危害完全不容忽视。

1.1.2　主流的恶意程序检测方法

Android 恶意程序检测方法目前主要分为基于特征代码的检测方法和基于行为的检测方法。基于特征代码的检测方法是通过检测程序文件的二进制代码是否匹配那些已知恶意代码的特征来进行判断程序文件是否为恶意，这个方法具有检测速度快、准确率高等优点，但是缺点也较为明显，无法检测出特征数据库中没有的恶意代码且需要长期维护一个巨大

的病毒特征数据库。目前，国外有一款著名的 Android 恶意代码检测程序 Androguard[6]，此检测程序就是基于特征代码方式进行实现的。

相反，基于行为的检测方法通过匹配待测程序的行为模式与已知恶意程序的行为模式，来判断待测程序是否可能包含恶意代码。此方法虽然误报率较高，但是却可实现对未知恶意代码和未知病毒的检测，对基于特征代码的检测方法起到了很好的弥补作用。基于行为模式的分析又能进一步划分为动态分析法和静态分析法两种[7]。动态分析法是通过“沙盒或虚拟机”等技术来模拟运行待测程序，然后监控与记录待测程序运行时的具体行为特征，具有能够检测采用过代码保护机制的恶意程序的能力，但缺点是整个检测过程计算资源消耗较大、时间用时较久、代码覆盖率偏低。除了上述的重量级的动态分析，采用静态分析法则属于轻量级的方法，该方法通常是通过逆向工程抽取待测程序的特征，例如分析程序中的函数调用、程序的指令调用序列，具备检测速度快、代码覆盖率高等优点。

1.1.3 恶意代码的变种与常见检测方法

虽然恶意代码的数量逐年递增，但是新的变种往往采用在原有恶意代码种类的基础上进行一些变换操作的方法[8]。因为恶意软件的作者在制作生成恶意软件的一系列过程中，往往会采用模块重用技术或者自动化工具来快速编写新的变种。这些很大程度上共用了祖先的相同代码，故有一定内在的关联性、相似性的恶意代码合称为恶意代码家族。对恶意代码的分类研究非常重要，是直接影响恶意代码检测效果的一个核心因素。

在学术领域，已经有不少学者对恶意代码可视化方法在进行了初步研究[9-11]，此类方法的基本思路都是通过将恶意代码的特征转变为视觉图像的特征，再利用常见的图像特征的提取方法和分类方法来间接实现恶意家族分类效果或者恶意代码检测效果。

1.2 国内外相关工作

1.2.1 恶意代码检测的相关工作

2006 年，Miettinen[12]等人提出了一种基于主机的恶意代码检测方法。他们认为移动设备是唯一有权限获得设备安全状态的对象，与其他方式相比较，基于主机收集的信息要更有精确性和可靠性。但是，该方法主要是收集移动设备的内存、CPU、文件 I/O、网络 I/O 等信息，却无法适应恶意移动应用的资料窃取、恶意吸费、恶意扣费等攻击方式。

2010 年，Shabtai, Asaf[13]等人又进一步发展了之前的理论，提出了一种在 Android 系统中构建动态检测环境的方法，将系统的 CPU、网络、内存、电量等信息作为检测对象，引入并比较了机器学习中 NB、BN、k-Means、Logistic Regression 等多种分类算法，实验表明该方法可获得大约平均 80%的准确率。但是，测试集只选用了作者编写的少数几个恶意应用程序，显得过于薄弱而无法适应不断发展的恶意程序检测。

2010 年，Thomas Blasing[14]等人提出了一种 Android 应用程序沙箱的概念，其核心思想是通过 Android 应用沙箱进行静态分析和动态分析。此方法的静态分析是通过获取应用程序的 smali 代码并实现模式匹配，最后将结果输出到日志；而动态分析方法则在 Android 模拟器的 Linux 底层增加一个代理层来实现劫持系统调用，最后利用 Moneky 工具来简单模拟用户输入，并将结果输出到日志。该方法的缺点是构建代理方法过于复杂，而且动态分析方法只分析了单位时间内的系统调用个数，且对 Android 系统相关函数调用没有深入进行研究。

2014 年，Yang Chen[15]等人提出了一种利用隐马尔科夫模型作为 Android 系统的 Intent 对象互信息建模，其中将 80%数据作为训练集和 20%数据作为测试集，最后实验评估环节表能明能达到 70%左右准确率。由于 Intent 对象所携带信息并不够充分，所以这种动态检测方法并不能准确地描述所有的恶意行为，比如部分恶意攻击行为在 Intent 对象交互上无法表现。

另外，恶意代码检测技术在 PC 平台上已经发展的比较充分，比如利用 N-Gram 编码提取汇编指令特征的方法已经获得了很好的效果。例如，文献[16]提出了一种 OpCode N-Gram 的方法。该方法代替之前的 bytecode N-Gram 方法，进一步提高检测精度，并以 x86 平台的汇编指令为实验对象进行了验证。文献[17]提出利用 OpCode 序列频率的表达式来检测和分类恶意代码，并进一步结合了数据挖掘等多种方法，该方法也有较高的检测率。

静态分析方法是从 AndroidManifes.xml 文件、库文件、反编译后的 Java 源文件中进行提取特征与分析特征。例如，文献[18]提出通过反编译 APK 文件得到 Java 源代码，再从 Java 源代码中提取出调用方法名作为特征，最后结合 N-Gram 编码等方法实现恶意软件分类，但是该方法具有不能处理混淆过的 Java 源代码的缺点；文献[19]提出了一种轻量级的 Android 恶意代码检测方法，该方法将 APK 中的申请权限、敏感 API 调用以及网络地址等信息作为静态分析的特征，并通过 5,560 个恶意样本和 123,453 个良性样本作为实验数据。实验表明该方法有较高的准确率，但是该方法中的特征提取完全依赖于反汇编后的 Java 源

代码，若恶意代码对 Java 源码进行混淆将会导致该方法失效；文献[20]提出了一种简化 Dalvik 中间语言（SDIL）用于恶意代码分析，该中间语言结合 MOSS 算法进行了实验测试，实验验证了该分析方法的有效性，但是该方法在恶意代码分析、恶意特征提取时还需要加入大量人工操作的部分，并且该方法提取的特征数量与涵盖的恶意代码种类仍然较少。

1.2.2 恶意代码可视化与家族分类的相关工作

恶意代码的可视化方法在 PC 平台已经有不少学者进行了一定的研究。2011 年，L Nataraj[21]等人提出了一种将 PE 格式的恶意代码转化为纹理图像并进行恶意代码分类的方法，该方法具有无需反汇编和执行代码等多种优点。2012 年，YWu[22]等人提出了一种点图可视化方法并对恶意代码进行了聚类分析研究。2013 年，KS Han[23]等人提出将恶意代码反汇编后提取出 OpCode 部分，并用哈希函数将 OpCode 序列转化为像素点的坐标值信息与 RGB 色彩信息，最后通过相似度函数来计算图像的相似度来对恶意代码家族进行分类。2014 年，M Shaid[24]等人提出用不同的色系代表不同的 API 调用的风险程度，并生成彩色图像来可视化恶意代码的行为模式。实验表明该方法用于识别恶意代码的变种有较高的精准度。另外，国内也有学者进行了相关的可视化研究，韩晓光[25]等人提出将恶意代码转化为灰度图片，然后用灰阶共生矩阵算法提取灰度图片的纹理指纹特征作为恶意代码的指纹，实验表明该方法对恶意代码变种有较好的识别能力。目前除了 PC 平台，Android 移动平台利用可视化方法来检测恶意代码变种的相关研究目前还是空白。

2012 年，北卡罗莱纳州立大学 Y Zhou[26]等人对 Android 恶意家族进行了系统性的研究。他们的研究团队在 2010 年 8 月到 2011 年 10 月期间一共收集了 Android 平台共计 1,200 个恶意样本，并还对 Android 恶意家族的特征类型与演化过程进行了系统性的研究。2014 年，D Arp[27]等人也收集了 5,560 个恶意样本，并提出一个名为 Drebin 的方法。该方法将 APK 中的申请权限、敏感 API 调用以及网络地址等信息作为静态分析的特征，除了能够检测 Android 恶意代码，也能对属于不同家族中的 Android 恶意代码进行分类，该方法具有较高检测率的优点，但是误报率并不理想。

1.3 本文研究内容

本文对 Android 恶意移动应用的静态检测方法进行了深入而详细的研究，主要做了如下多个方面工作：

（1）介绍相关理论知识与关键技术，包括 Android 系统的相关知识、N-Gram 编码原理、常见的聚类算法及 Random Forest 算法、常见的分类算法及 AP 聚类算法、图像纹理特征提取算法等。

（2）提出了一个基于 Dalvik 指令简化的恶意代码分类方法。该方法简化 Dalvik 指令集，用指令符号抽象一类指令的操作码，并提出了一种基于 N-Gram 序列的特征模型。针对抽象的 Dalvik 指令符号的 N-Gram 序列特征，利用 AP 聚类算法进行样本压缩和利用信息增益方法进行特征筛选，并结合分类算法进行模型训练，有效地实现了 Android 恶意代码的检测模型。最后，与常见反病毒软件进行实验相比较，实验表明该方法有较高的检测率

（3）深入研究了 DEX 文件结构与 bytecode 生成图纹理之间的对应关系，提出一种可快速生成 Dalvik bytecode 生成图的方法。

（4）提出了一个基于 bytecode 生成图的恶意家族分类方法，该方法通过结合 GIST 算法与 Random Forest 算法，实现对 bytecode 生成图的纹理特征进行了分类实验，进而实现了 Android 平台的恶意代码家族分类。最后，与 Drebin 方法进行实验相比较，实验表明该方法有较高的检测率。

（5）基于上述工作，设计并实现了 Android 恶意代码智能检测平台，该平台可以满足在线 APK 分析服务的应用场景。智能检测平台分为 Web 交互服务模块和恶意代码检测模块。恶意代码智能检测平台设计并实现完成后，进行了一定的功能测试，鉴定检测平台的正确性、健壮性、安全性。

1.4 论文组织结构

本文的章节安排如下：

第1章　绪论。本章主要介绍了研究背景与国内外相关工作。研究背景包括恶意程序高速增长的现状、一些主流的恶意程序检测方法、恶意代码的变种与常见检测方法。最后说明本文所做的主要研究与论文组织结构安排。

第2章　相关理论知识和关键技术。本章通过对 Android 应用程序包格式、Dalvik 虚拟机与反汇编、N-Gram 编码、聚类算法、分类算法、图像纹理特征提取算法的介绍为第三章中基于 Dalvik 指令简化的恶意代码分类方法和第四章中基于 bytecode 生成图的恶意家族分类方法提供理论知识和技术支持。其中，AP 聚类算法将用于第三章中的大容量样本压缩，纹理特征提取算法将用于第四章中的恶意家族样本的特征提取。

第3章　基于 Dalvik 指令简化的恶意代码分类方法设计。本章提出了一个基于 Dalvik 指令简化的恶意代码分类方法。该方法简化 Dalvik 的官方指令集，用指令符号抽象一类指令的操作码，并提出了一种基于 N-Gram 序列的特征模型。针对抽象的 Dalvik 指令符号的 N-Gram 序列特征，利用 AP 聚类算法进行样本压缩和利用信息增益方法进行特征筛选，并结合分类算法进行模型训练，有效地实现了 Android 恶意代码的检测模型，并与常见反病毒软件进行实验相比较，实验表明该方法有较高的准确率。

第4章　基于 bytecode 生成图的恶意家族分类方法设计。本章首先研究了 DEX 文件结构与 bytecode 生成图纹理之间的对应关系，提出一种可快速生成 Dalvik bytecode 生成图的方法。然后，提出了一个基于 bytecode 生成图的恶意家族分类方法。该方法通过结合 GIST 算法与 Random Forest 算法，对 bytecode 生成图的纹理特征进行了分类实验，进而实现了 Android 平台的恶意代码家族分类，并与 Drebin 方法进行实验相比较，实验表明该方法有较好的分类效果。

第5章　Android 恶意代码智能检测平台的设计与实现。本章设计并实现了一个 Android 恶意代码智能检测平台。Android 恶意代码智能检测平台总体架构主要分为两大功能模块：Web 交互服务模块和恶意代码检测模块。本章首先介绍了整个检测平台的数据流程。然后分别对两大功能模块的八个功能子模块从功能设计和技术实现角度进行了详细的阐述。最后，还对检测平台进行了功能测试，鉴定检测平台的正确性、健壮性、安全性。

第6章　结论与展望。本章对本文内容进行了归纳与总结，并对本文的值得改进的地方、本课题进一步研究方向进行了探讨与展望。

第 2 章　相关理论知识和关键技术

2.1　Android 系统相关知识

2.1.1　Android 应用程序包格式

Android 操作系统是一个基于 Linux 内核的开源式移动操作系统，由 Google 公司成立的 Open Handset Alliance 联盟领导与开发的，主要设计用于如智能手机、平板电脑等触屏式移动设备。

Android 应用程序包又称为 APK (Android application package) ，APK 是 Android 操作系统中用于分发和安装移动应用的一种应用程序包格式，基于 ZIP 文件格式。Android 应用程序的必须先对源代码进行编译，然后再封装成为一个被 Android 操作系统所能识别的文件格式才可以被正常运行。一个标准的 APK 文件内包含被编译的 DEX 文件，文件资源，证书和清单文件。所有的文件中需要重点注意的是 classes.dex 文件，classes.dex 文件封装了可被 Dalvik 虚拟机识别并执行的 Dalvik bytecode。DEX 文件的结构由多个结构体组合而成，包括 dex header、string_ids、method_ids、class_def、type_ids、proto_ids、field_ids、data 等多个部分。

2.1.2　Dalvik 虚拟机与反汇编

2007 年底，Google 正式发布了 Android SDK。Dalvik 虚拟机作为 Android 平台的核心组件之一，具有体积小、占用内存空间小、执行速度快等优点，拥有专属的 DEX 可执行文件格式和指令集代码。与 JAVA 虚拟机(JVM)有所不同，Android 的虚拟机称为 Dalvik 虚拟机（DVM）。Java 虚拟机运行的代码形式是 Java bytecode，而 Dalvik 虚拟机运行的代码形式是 Dalvik bytecode。两者的架构也有所不同，JVM 是基于栈架构，而 DVM 基于寄存器架构。Smali 工具和 baksmali 工具是针对 DEX 执行文件格式的汇编器和反汇编器，在反汇编后 DEX 文件会产生.smali 后缀的代码文件，smali 代码拥有特定的一套格式与语法，它是由一个名叫 JesusFreke 的黑客对 Dalvik bytecode 的解释，而非一种官方标准语言。因为 Dalvik 虚拟机名字来源于冰岛的一个小渔村的名字，JesusFreke 便把 smali 和 baksmali

取自于冰岛语中的“汇编器”和“反编器”。目前 Smali 是在 Google Code 上的一个开源项目。虽然主流的 DEX 可执行文件反汇编工具不少，如 Dedexer、IDA Pro 等，但 smali 工具提供反汇编功能的同时，也提供了打包反汇编代码重新生成 DEX 的功能，因此 smali 工具被广泛地用于 APP 广告注入、汉化、破解和 ROM 定制等多个方面。

apktool 工具是在 smali 工具的基础上进行二次封装和改进的，除了具有对 DEX 文件的汇编和反汇编功能外，还具有对 APK 文件 中已编译成二进制的资源文件进行反编译和重新编译等强大功能。所以，本文直接使用 apktool 工具来反汇编 APK 文件，而不是再独立使用 smali 工具和 baksmali 工具。

2.2 N-Gram 编码原理

N-Gram 是一种基于马尔科夫模型的文法统计模型，该模型通常用于文档分类、信息检索等。N-Gram 是基于一种假设，一个句子中出现的第 N 个词只与前面出现的 N-1 个词相关，而与其它任何词都不相关。所以，一个句子的概率就等于各个词出现概率的乘积，这些词语的概率则可以通过直接从语料中统计词语出现的次数得到。

N-Gram 可以捕获文本中的一些潜在的特征，而这些特征往往具有很好的分类效果，并且很难通过其他方法进行提取得到。N-Gram 可以理解为由一个长度为 N 的滑动窗口滑过一段字符串不断生成一系列子字符串的过程，而且这个滑动窗口每次仅向前滑动一个单位长度，当 N 为 2 时 N-Gram 实现过程如图 2-1 所示。

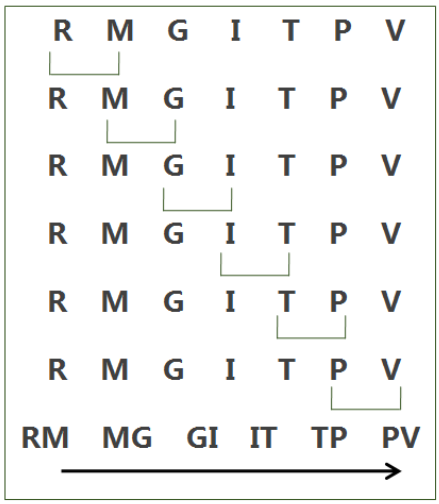

图 2-1 N-Gram 的滑动窗口（N=2）

而对于不同大小的 N，N-Gram 可以生成不同长度与个数的特征子字符串，现在假设有一个指令序列为 RMGITPV， 以 2-Gram（N=2）提取的特征子字符串为[{RM}，{MG}，{GI}，{IT}，{TP}，{PV}]，以 3-Gram（N=3）提取的特征子字符串为[{RMG}，{MGI}，{GIT}，{ITP}，{TPV}]， 以 4-Gram（N=4）提取的特征子字符串为[{RMGI}，{MGIT}，{GITP}，{ITPV}]，具体实现过程如图 2-2 所示。

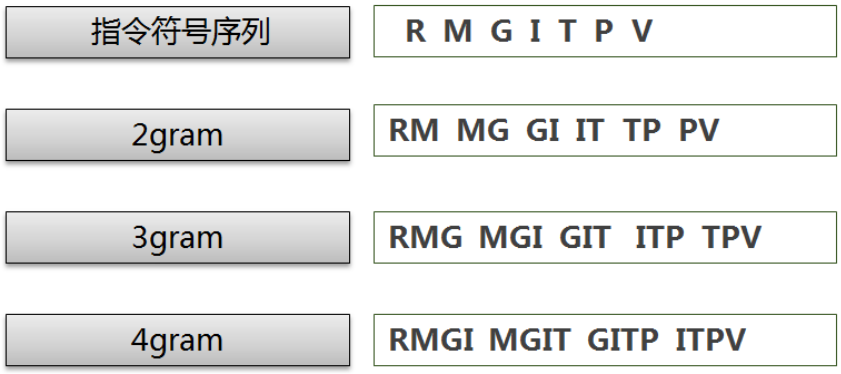

图 2-2　取不同值的 N-Gram

2.3　聚类算法

2.3.1　聚类算法介绍

聚类算法是数据挖掘和机器学习中一项十分重要的算法，在许多领域受到广泛应用，包括统计学，机器学习，数据挖掘，模式识别，电子商务，空间数据库，图像分析以及生物信息等。聚类算法就是一个将数据集划分为由若干个相似实例组成的簇的过程，最终使处于同一个簇中的实例间的相似度最大化，处于不同簇中的实例间的相似度最小化。简而言之，一个簇集合就是由彼此相似度很高的一组实例对象构成，归属于不同簇集合的实例对象通常相似度很低或不相似。

作为一种重要的数据挖掘算法，聚类算法是一种无监督的机器学习算法，主要依据样本间相似性的度量标准将整个数据集自动划分为几个簇，而且聚类中的簇的划分并不是预先定义的，而是根据实际数据的特征按照数据间的相似性由算法自动生成的。聚类算法的输入参数是一组样本数据和一个度量样本间的相似度的标准，输出内容是簇的不同集合。聚类算法的另一个重要作用是对每个簇进行综合描述和初步评价其样本结构，这个结果可以用于进一步深入分析数据集的特性。

聚类算法基本可以分为如下的几种类别：基于划分的聚类方法（Partitioning Methods）、基于层次的聚类方法（Hierarchical Methods）、基于密度的聚类方法（Density-Based Methods）、基于网格的聚类方法（Grid-Based Methods）、基于模型的聚类方法（Model-Based Methods）等。另外，直接导致数据集的聚类效果好坏的因素一般有三个：聚类数目设置的正确性、度量数据点之间相似度的距离函数和聚类算法的优良性[28]。

2.3.2 AP 聚类算法

2007 年，由加拿大多伦大大学的 Frey [29]等人在 Science 杂志上提出的一种全新的半监督聚类算法，称为 AP（Affinity Propagation）聚类算法或吸引子传播算法，该算法近年来在学术界和工业界均获得了极大的关注和应用。与 K-Means 等传统算法相比，AP 聚类的基本思想是通过数据点之间传递消息。AP 聚类算法有如下优点：不需要进行随机选取初值步骤，不需事先指定聚类的个数，对距离矩阵的对称性没要求，对于大规模数据集花费时间代价小。

AP 聚类算法不需要事先指定聚类数目,相反它将所有的数据点都作为潜在的聚类中心,称之为 exemplar。s(i, j)为数据点 i 和 j 的相似度，preference 为数据点 i 的参考度，简写为 p(i)，它反映了初始时选 i 作为聚类中心的程度。AP 聚类执行时在数据点间传递两种消息，分别称为吸引度（responsibility）和归属度（availability）。吸引度(responsibility)是从数据点 i 传递到其候选聚类中心数据点 k 之间的信息，称为点 k 对于点 i 的吸引度值，记为 r(i, k)。吸引度 r(i, k)反映的是数据点 k 通过与其它的数据点 k’比较，作为适合数据点 i 的聚类中心的程度。r(i, k)的计算需要引入数据点 i 对于其它潜在的候选聚类中心点 k’的归属度 a(i, k’)作为参考比较。归属度（availability）是从候选聚类中心数据点 k 传递到数据点 i 的信息，称为数据点 i 对于数据点 k 的归属度值，记为 a(i, k)。归属度 a(i, k)反应的是数据点 i 选择数据点 k 作为其聚类中心的适合程度。同样，a(i, k)的计算需引入作为候选聚类中心的点 k 对其它数据点 i’的吸引度作为参考比较。

2.4 分类算法

2.4.1 分类算法介绍

分类算法是数据挖掘和机器学习中一项十分重要的算法，广泛的应用于医疗诊断、应用卡系统的信用分级、图像模式识别等多个领域。分类的目的是生成一个分类模型，通过该分类模型能把数据库中的数据项映射到某一个类别。分类算法可以用于预测结果，预测的过程是对给定的数据从历史数据集合中自动计算出它的未来趋势。在统计学中常用的预测方法是回归，在数据挖掘中的分类算法与统计学中的回归方法是一对相互联系又有区别的概念，分类算法的输出形态是离散型的类别值，而回归算法的输出形态则是连续型数值。

在进行分类之前，首先要将数据集按照不同比例划分为训练集和测试集两大部分。分类分两步，第一步是根据训练集的特点选择分类算法并构建分类模型，常用的分类算法有决策树分类、贝叶斯分类、k-最近邻分类等；第二步使用已经构建好的分类模型对测试集中的数据进行分类，进一步评估分类模型的准确度等多项指标，最终选择出最优的分类模型。分类算法主要分为以下几大类：

1) 决策树分类方法。决策树分类方法通过对训练集进行训练，生成一棵的二叉决策树或多叉的决策树。决策树包含三种节点：根节点没有入边，但有零条或多条出边；内部节点只有一条入边和两条或多条出边；叶节点只有一条入边，但没有出边。树的叶节点代表某一个类别值，非叶节点代表某个一般属性的一个测试，测试的输出构成该非叶子节点的多个分支。从根节点到叶节点的一条路径形成一条分类规则，一棵决策树能够方便地转化为若干分类规则，挖掘者可以根据分类规则直观地对未知类别的样本进行预测。不同决策树采用的技术不同，已经有很多成熟而有效的决策树学习算法，如 Random Forest 算法等。
2) 贝叶斯分类方法。贝叶斯分类方法有一个明确的基本概率模型，用以给出某个样本属于某个类别标签的概率。贝叶斯分类方法有两种主要实现：朴素贝叶斯分类器和贝叶斯网络。朴素贝叶斯分类器是基于贝叶斯定理的统计分类方法，它假定属性之间相互独立，但实际数据集中很难保证之一条件。朴素贝叶斯分类器分类速度快且分类准确度高，支持增量学习。贝叶斯网络使用贝叶斯网络描述属性之间的依赖关系。

3) k-最近邻分类方法。k-最近邻分类使用具体的训练实例进行预测，不必维护从数据集中抽象出来的模型。这种基于实例的学习算法需要邻近性度量来确定实例间的相似度或距离，还需要分类函数根据测试实例与其他实例的邻近性返回测试实例的预测类别标签。最近邻分类器基于局部信息进行预测，而决策树分类器则试图找到一个适合整个输入空间的全局模型。由于基于局部分类策略，k-最近邻分类在 k 很小的适合对噪声非常敏感。

4) 神经网络分类方法。神经网络是大量的简单神经元按一定规则连接构成的网络系统，能够模拟人类大脑的结构和功能。它采用某种学习算法从训练样本中学习，将获取的知识存储在网络模型的权值中，模拟人类大脑通过同一个脉冲反复刺激下改变神经元之间神经键连接强度来进行学习。按照各神经元的不同连接方式，神经网络分为前向网络和反馈网络。目前的神经网络模型非常丰富，典型的模型有感知器模型、多层向前传播模型、BP 模型、Hopfield 网络、SOM 自组织网络，等等。

2.4.2 Random Forest 分类算法

Random Forest 算法是机器学习中一种常见的分类算法，它具有较高的准确率，不容易陷入过拟合，具有很好的抗噪声能力，能够处理很高维度的特征数据，所以在生物信息、统计学、计算机等领域有着广泛的应用。该算法以随机的方式建立一个由决策树构成的森林，Random Forest 的每一棵决策树之间并不产生关联关系。在生成 Random Forest 之后，当有一个新的输入样本进入算法时，森林中的每一棵决策树都会分别进行一下判断，判断这个样本应该属于哪一分类，最终以投票的形式决定属于哪一类别。

早在 1995 年，贝尔实验室的 Tin Kam Ho[30]等人提出了 Random Forest 算法思想。1998 年，Tin Kam Ho[31]等人提出了一种用随机切分训练集构造决策树的改进方法。2001 年，Brelman[32]等人在旧算法的基础上提出一种 bagging[33]方法来随机选择特征，并结合使用 CART 方法进行随机生成决策树，使 Random Forest 算法得到了进一步完善。2003 年，Zhou Z[34]等人提出了一种选择性集成的新思想，认为从已有的个体学习器中进行选择之后再集成能达到更好的性能。2004 年，Robnik-Sikonja M[35]等人提出了对特征进行评价并结合特征加权的方式来统计投票结果。2012 年，Kotsiantis[36]等人提出一种集成算法，对 Bagging、Boosting 和 Random Subspace 等算法进行组合。

2.5 图像纹理特征提取算法

2.5.1 图像纹理特征介绍

图像的三大特征分别是色彩、形状和纹理，其中色彩和形状这两个特征在人们看到图像瞬间就有了直观的感受，但是如果想对图像进行更加具体的观察，就必须引入纹理这个特征。纹理特征是局部区域内像素灰度的空间分布特性，几乎任何物体表面的局部区域的色彩和亮度并不是单一的，都存在一定程度灰度变化即纹理特征。

自然界的事物普遍存在纹理特征，不同的物体或不同的图像往往具有不同的纹理特征，纹理特征反映了一种物体表面结构的灰度变化与组织排列，如像素的数目以及这些像素的相互关系。纹理特征帮助人们视觉系统对物体或图像的质地、光滑度、规则性和稀疏度形成一种认识，决定着物体对人们达到的视觉效果。人类很难用语言去精确描述纹理或者两种纹理的差别，但是可以很明确的辨别两种具有完全不同纹理特征的图像或物体。

对于纹理特征的形成有两种解释的方法：随机性方法与结构化方法。随机性方法将纹理特征的形成看作是一个通过统计参数进行二维随机排列而形成的区域，其中较为典型的观点是 Cross 和 Jain[37]等人提出的纹理是一个随机的或周期性的二维图像集合并且在纹理特征中存在某种空间结构。相反，结构化方法认为纹理的形成是由某个局部区域中像素组成的某种模式在不断的周期性或准周期性重复。

2.5.2 纹理特征提取方法介绍

有心理学的研究表明，人类的视觉系统可以做到在很短的时间内提取出一幅图像的表项形式与主要特征，并快速理解整幅场景，这被称为对图像或场景的 GIST。基于人类视觉系统的这类现象，有不少学者进行了大量研究，提出了一些能够提取出快速识别图像内容的方法。在这些方法中，较经典的方法是 Olive 和 Torralba[38]等人所提出 GIST 模型，GIST 模型使用场景的光谱及其他的局部信息来描述场景的整体空间属性，并用五类特征描述场景的空间属性，它们分别是：

1) 自然度（Degree of Naturalness）：自然度一般用于表示自然场景与人造场景的区别，通常人造场景的区域纹理是垂直或水平的，而自然场景的域纹理则是无规则的，自然度的取值取决于场景区域纹理边缘的倾向。

2) 开放度(Degree of Openness)：开放度指场景的空间曲线的弯曲程度，是封闭的还是开放的，如山、城市等场景属于封闭场景，草原、大海等属于开放场景。这种特征适用于现实世界任何类型的物体。

3) 粗糙度（Degree of Roughness）：粗糙度又称复杂度，主要指场景的空间元素的大小，取决于各个元素的尺寸，以及其组成元素之间的相互关系。这种特征适用于现实世界任何类型的物体。

4) 膨胀度（Degree of Expansion）：膨胀度是指空间场景中的平行线的收敛程度，也可引申为空间梯度的深度。这种特征更适用于城市的场景，因为自然界中并行直线的收敛较少，多为不规则的场景

5) 险峻度（Degree of Ruggedness）：险峻度指空间图像相对于水平线的偏移程度，如山峦、悬崖等场景与地平线之间存在倾斜，其倾斜程度越大，则其险峻度越大。这种特征更适用于描述自然界的场景，因为人工制造的场景相比自然场景而言其倾斜程度较小。

五种特征共同组成了 GIST 模型，它们能够对空间场景进行深刻的描述，在图像纹理分析中有十分重要的作用。图像空间结构全局特征抽取是针对以上五个特征对图像进行特征描述，基本思路是利用分块提取图像的全局特征。首先，对原始图像进行滤波处理，然后将滤波处理后的图像划分为 N×N 个互相不重叠的相同大小子区域，最后利用离散傅立叶变换提取各个子区域的特征，以获得全局特征。

2.6 本章小结

本章内容可以概括为五部分，第一部分是 Android 系统相关知识，第二部分是 N-Gram 编码简介，第三部分是聚类算法介绍，第四部分是分类算法介绍，第五部分是图像纹理特征与提取算法介绍。

在第一部分中，通过对 Android 系统相关知识的介绍，通过对 APK 结构与原理、反汇编与 Dalvik 虚拟机的原理阐述，为第 3 章的反汇编成 Smali 文件与 Dalvik 指令简化提供理论依据，为 4 章的 DEX 文件生成为 bytecode 生成图做好了理论准备工作。

在第二部分中，通过对 N-Gram 编码的介绍，为第 3 章的指令符号进行 N-Gram 编码及 N-Gram 编码作为特征做好了理论准备工作。

在第三部分中，介绍了常见的聚类算法，并进一步详细阐述了 AP 聚类算法。该 AP 聚类算法将在第 3 章中体现，用于实现样本压缩、去除噪声数据等功能。

在第四部分中，介绍了常见的分类算法，并进一步详细阐述了 Random Forest 分类算法。该 Random Forest 分类算法将在第 3 章和第 4 章进行分别体现，用于实现训练数据的建模。

在第五部分中，介绍了图像纹理特征与提取算法，为第 4 章的图像纹理特征的提取关键步骤，奠定了相关的理论基础。

第 3 章　基于 Dalvik 指令简化的恶意代码分类方法设计

3.1　系统模型设计

本章提出了一种基于 Dalvik 指令简化的恶意代码分类方法。通过该方法可以快速有效地检测出 Android 移动应用中的恶意程序。该方法的整个系统模型分为两大部分，第一部分是生成恶意代码检测模型，如图 3-1 所示；第二部分是测试恶意代码检测模型，如图 3-2 所示。

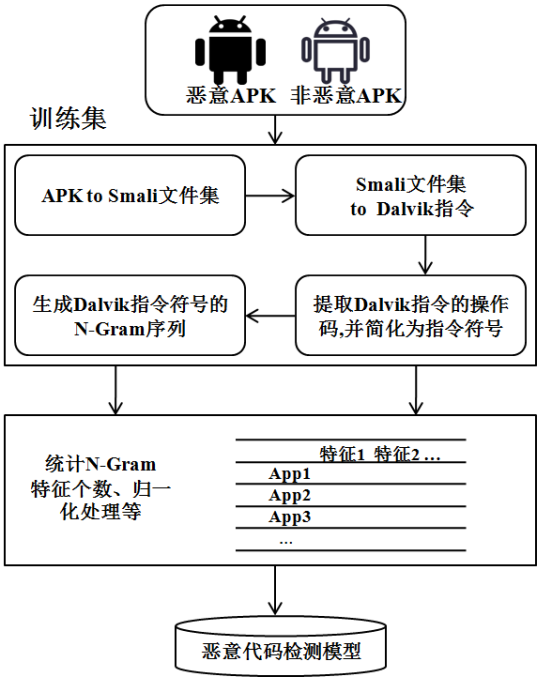

图 3-1　生成恶意代码检测模型

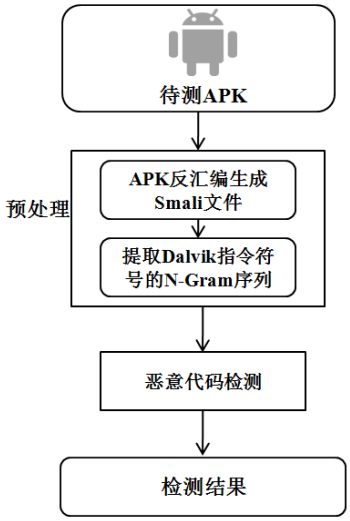

图 3-2　测试恶意代码检测模型

首先，恶意代码检测模型的训练集分为两个基本的子集，一个是恶意移动应用样本集，另一个是良性移动应用样本集。Android 的移动应用以 APK 文件形式存在，它通常都包含一个名为 classes.dex 文件，该 DEX 文件包含了可被 DVM 执行的 Dalvik bytecode。利用工具 apktool[39]实现对 APK 文件反汇编，就反汇编能生成一个 smali 源码的目录，该 smali 目录结构基本对应着 APK 的 Java 源代码的 src 目录。接着，从 smali 文件中提取出 Dalvik opcode 并进行抽象简化为指令符号，再对 Dalvik 指令符号进行 N-Gram 编码，最后对 N-Gram 特征进行统计与归一化处理，并结合机器学习领域中的相关分类算法建立一个恶意代码检测模型。

利用模型检测恶意代码时，先将待测 APK 文件先进行相关的预处理步骤，提取出 Dalvik 指令特征并作同样的抽象简化与 N-Gram 序列化处理，最后通过恶意代码检测模型的测试，就可判断出待测 APK 文件是否为恶意应用。

3.2　Dalvik 指令特征与 N-Gram 编码

文献[40]表明官方标准的 Dalvik 指令有 230 条，分别包括比较指令、跳转指令、判断指令等十几种类型。本章设计了一种指令符号方案：在官方 230 条指令中提取出能够正确反映程序语义的指令操作码，对功能相近或相似的指令进行归类，并将归类后的指令抽象

为一类指令符号，与原有的 Dalvik 的指令集相比，显得更加简化与精要，也便于后续的分析与特征提取。表 3-1 中给出了指令符号集合的定义表，在 Dalvik 指令集中提取出的 107 种代表性指令并分成 10 大功能相近的类别，每类指令抽象为一个特定的指令符号。

表 3-1　指令符号集合的定义表

指令符号	指令含义	数量	代表性指令前缀
C	比较	5	cmpl-float|cmpg-float|cmpl-double|cmpg-double|cmp-long
D	定义	11	const|const/4|const/16|const-wide|const/high|const-string
M	操作	13	move|move-wide|move-object|move-result|move-exception
R	返回	4	return|return-void|return-wide|return-object
L	锁指令	2	monitor-enter|monitor-exit
G	跳转	3	goto|goto/16|goto/32
I	判断	12	if-eq|if-ne|if-lt|if-ge|if-gt|if-le|if-eqz|if-nez|if-ltz|if-gez|if-gtz|if-lez
T	读	21	aget|iget|sget|aget-wide|aget-object|aget-boolea|aget-byte|aget-char
P	写	21	aput|iput|sput|aput-wide|aput-object|aput-boolea|aput-byte|aput-char
V	函数调用	15	invoke-virtual|invoke-super|invoke-direct|invoke-static

N-Gram 编码经常用来处理恶意代码的分析。对上述 Dalvik 指令符号进行 N-Gram 编码可以用于提取 Android 恶意应用的特征。同时，对于高维度的特征，本章可以使用信息增益等算法对特征进行优选，优选出区分度最高的特征集合。表 3-2 中给出了当采用 3-Gram 编码时，APK 集合的特征频率的统计表。

表 3-2　3-Gram 特征频率的统计表

APK 集合	CCC	CCD	CCM	…	VVV
DroidKungFu.apk	0.032016	0.010601	0.019507	..	0.071454
BaseBridge.apk	0.059207	0.011801	0.018162	..	0.047486
Plankton.apk	0.030508	0.006391	0.017376	..	0.053952
…	…	…	…	..	…

3.3 分类算法的评估标准

分类算法是一种最常见的机器学习方法。文献[41]介绍了较为常见的分类算法的评估参数。一般评估分类算法的有效性最基本的指标分别是：真阳性（true positive, TP），表示那些被正确分类为恶意代码的恶意样本数；假阳性（false positive，FP），表示那些被错误分类为恶意代码的良性样本数；真阴性（true negative，TN），表示那些被正确分类为良性代码的良性样本数；假阴性（false negative，FN），表示那些被错误分类为良性代码的恶意样本数。这些基本指标又可以构成如表 3-3 所示的一个混淆矩阵，该矩阵可以用于评估分类算法。

表 3-3　混淆矩阵

	预测恶意	预测非恶意
实际恶意	TP	FN
实际非恶意	FP	TN

由上述的基本指标还构成下面的这些指标：

$$TPR = Recall = \frac{TP}{TP + FN} \tag{3-1}$$

$$FPR = \frac{FP}{FP + TN} \tag{3-2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3-3}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3-4}$$

$$F - Measure = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{3-5}$$

真阳性率（TPR）也称为检测率；假阳性率（FPR）也称为误报率；正确率（Precision）指在预测为恶意样本分类中实际为恶意样本的比例；分类精度（Accuracy）是所有得到正确分类的样本与所有实验样本之比；F-Measure 指标指的是准确率与召回率的调和平均值。ROC 曲线又称为接收者运行特征，它是一种显示 TPR 和 FPR 之间关系的图形化曲线。AUC（Area Under the Curve）是 ROC 曲线下方的面积，当 AUC 为 1 时，表示完全准确，。

3.4 实验与结果分析

3.4.1 实验数据与实验环境

本章实验部分的恶意移动应用来源于德国哥廷根大学 Drebin 项目 [42]，良性移动应用通过爬虫技术从 Google Play 应用商店[43]下载得到。文献[44]指出 Google Play 应用商店中 52208 个移动应用中只有 2 个为恶意的应用，所以可认为从 Google Play 应用商店中下载的安卓移动应用均为良性。

Weka（Waikato Environment for Knowledge Analysis）是一款著名的开源机器学习软件，它集成了大量的机器学习算法，并且开放 API 等接口。所以本章使用 Weka 工具对特征进行训练、建模和测试。

3.4.2 算法与模型优选

本节首先采用 270 个恶意应用和 330 良性应用作为样本库，并以 2-Gram 编码，3-Gram 编码，4-Gram 编码分别提取不同类型的 Dalvik 指令符号序列，然后再分别使用 Random Forest、SVM、KNN、Naive Bayes 等分类算法对 3 种 N-Gram 编码的序列采用 10 折交叉验证比较，实验结果如表 3-4、表 3-5、表 3-6 所示。

表 3-4　2-Gram 序列的测试结果

分类算法	TPR	FPR	Precision	Recall	F-Measure	AUC
Random Forest	0.915	0.106	0.876	0.915	0.895	0.97
SVM	0.852	0.173	0.801	0.852	0.826	0.84
KNN	0.930	0.136	0.848	0.930	0.887	0.897
Naive Bayes	0.870	0.221	0.763	0.870	0.813	0.888

表 3-5　3-Gram 序列的测试结果

分类算法	TPR	FPR	Precision	Recall	F-Measure	AUC
Random Forest	0.956	0.079	0.908	0.956	0.931	0.982
SVM	0.922	0.13	0.853	0.922	0.886	0.896
KNN	0.952	0.148	0.840	0.952	0.892	0.903

Naive Bayes	0.867	0.17	0.807	0.867	0.836	0.903

表 3-6　4-Gram 序列的测试结果

分类算法	TPR	FPR	Precision	Recall	F-Measure	AUC
Random Forest	0.956	0.103	0.884	0.956	0.918	0.984
SVM	0.944	0.106	0.879	0.944	0.911	0.919
KNN	0.904	0.088	0.894	0.904	0.899	0.909
Naive Bayes	0.9	0.103	0.877	0.9	0.888	0.917

采用 TPR，FPR，AUC，Precision，Recall，F-Measure 等 6 个指标作为实验的评估标准，其中 AUC 是最值得参考的评估标准，因为它可以综合地反映 TPR 和 FPR 两者的关系，若 AUC 值越低代表综合表现越差，反之若 AUC 值越高代表综合表现越优异。另外，FPR 也是一项值得注意的指标，若 FPR 值越高代表误报率越高。分析表 3-4，Random Forest 算法的 AUC 值最高，FPR 值最低即误报率最低，但是 TPR 值略微不如 KNN 算法。分析表 3-5，Random Forest 算法的各项指标都呈现最优。分析表 3-6，Random Forest 算法的 AUC 值最高，它的 FPR 值与 FPR 值最优的 KNN 算法差距也不大。综合分析，在 2-Gram 编码，3-Gram 编码，4-Gram 编码的指令符号序列中，Random Forest 算法是综合表现最优的。

以上述 Random Forest 算法最优的结论为基础，进一步分析与寻找最优的 N-Gram 编码。通过综合比较表 3-4~表 3-6，可以看到 Random Forest 算法 AUC 值按从优到劣的依次顺序为：4-Gram 编码，3-Gram 编码，2-Gram 编码；Random Forest 算法 FPR 值按从优到劣的依次顺序为：3-Gram 编码，4-Gram 编码，2-Gram 编码；2-Gram 编码表现都是最差的，可以不再考虑。4-Gram 编码的 Random Forest 算法 AUC 值比 3-Gram 编码的 Random Forest 算法 AUC 值仅高出了 2%，而 4-Gram 编码的 Random Forest 算法 FPR 值比 3-Gram 编码的 Random Forest 算法 FPR 值却足足高出了 30%。综上所述，3-Gram 编码的 Random Forest 算法是一种最优的组合方案。

接着进一步分析样本集合的数量对实验指标的影响。设计 3 个不同规模的样本集作为对比实验数据，这 3 个样本集分别为 600 个样本、1200 个样本、2400 个样本，按 3-Gram 的 Random Forest 算法，采用 10 折交叉验证测试与评估，最终实验结果如表 3-7 和图 3-3 所示。不难发现，规模大的样本集的所有的评估指标都要优于规模小的样本集，这表明样

本集合的数量对恶意代码检测效果有一定的影响，采用规模越大的样本集，各项指标表现越优异。

表 3-7　样本数量对比表

样本数	TPR	FPR	Precision	AUC
600 samples	0.956	0.086	0.908	0.982
1200 samples	0.980	0.079	0.917	0.992
2400 samples	0.988	0.077	0.921	0.994

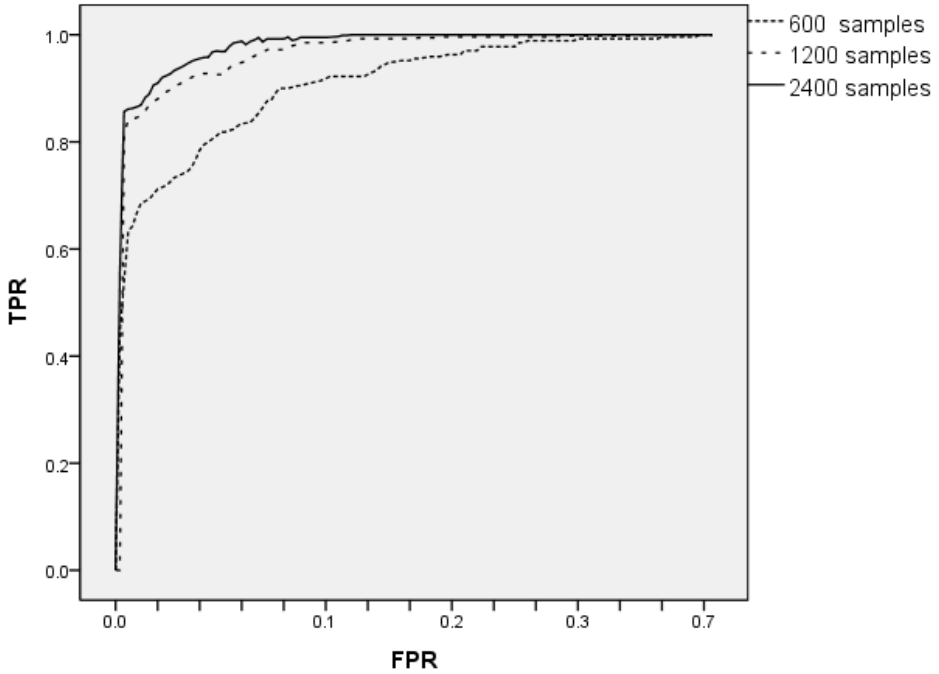

图 3-3 不同样本数量的 ROC 曲线对比图

最近的研究表明，AP 聚类算法能够有效地用于入侵检测领域中选取代表性的训练数据，并获得较好的实验效果 [45]. 恶意代码检测的数据训练过程与入侵检测的数据训练过程十分相似，所以本章选用 AP 聚类算法作为大样本的训练数据的优选，当训练数据集变小，检测模型的性能将会大幅度提高。首先，利用 AP 聚类算法来生成得到训练数据的具有代表性的聚类中心，再利用得到的聚类中心来训练模型。本章随机选择了 2400 个样本作为初始训练数据，然后利用 AP 聚类处理初始训练数据，并通过不断调整 preference 参数，

生成不同数量规模的聚类中心，并分别进行十折交叉验证，结果如表 3-8 所示，在 TPR 和 FPR 保持较小变化的情况下，时间性能得到了较大的提升。

表 3-8　时间成本与检测性能对比表

不同数量的聚类中心	训练时间(秒)	TPR	FPR
2400	2.90	0.988	0.077
2003	2.23	0.982	0.078
1610	1.82	0.981	0.081
1203	1.45	0.978	0.080
801	0.73	0.976	0.085
424	0.33	0.971	0.091

3.4.3　与专业反病毒软件实验比较

根据上一节的实验结论，以实验效果最优的 3-Gram 编码与 Random Forest 算法组合作为基础，通过计算信息增益的方法选出 200 个区分度最高的特征实现特征选择，作为本节检测模型的实现方案。本节实验将分别在 2000 个样本和 4000 个样本两个数据样本集上进行实验对比，并将 60%数据作为训练样本集，40%数据作为测试样本集。作为实验对比，也将测试样本集一并发送到 ThreatBook 威胁分析平台[46]。ThreatBook 平台具有多引擎的杀毒软件的功能，本节采用了其中的 9 款常见的反病毒软件（Avira、Dr.Web、AVG、Kaspersky、ESET、GDATA、Rising、MSE、Avast）给出的检测结果，根据检测结果统计出各反病毒软件的检测率与误报率。

根据表 3-9 显示的本次实验结果分析，大部分反病毒软件对测试样本的检测率超过了 90%，但是也存在几款检测率低于 50%的反病毒软件，据分析这些低检测率的反病毒软件并不适用于安卓应用程序格式的检测。在 2000 个数据样本集上，本节检测模型以 95.3%的检测率在 10 个系统中排名第 3；在 4000 个数据样本集，本节模型以 98.6 %的检测率在 10 个系统中排名第 2，这表明随着训练样本数量的不断增加，本节模型的精确度还能进一步提高。另外必须注意的是，实验所采用的恶意样本早已经对外公开了很长一段时间，大多数反病毒软件的病毒特征库早已经提取并加入了这些恶意样本的特征。但是，在检测那些未公开或未知的恶意样本时，本节的检测模型则更加能体现检测未知恶意代码的优势。

9 款反病毒软件的误报率较低，本节模型的误报率比专业病毒软件要稍微高一点。但是，表 3-10 表明如果训练样本数量能够增加，本节模型的误报率能够进一步下降。

表 3-9　本节检测模型与反病毒软件的检测率(%)比较

样本集	本节模型	AV1	AV2	AV3	AV4	AV5	AV6	AV7	AV8	AV9
2000	95.3	100.0	97.0	94.0	91.5	91.0	66.0	49.5	15.5	11.0
4000	98.6	99.1	92.7	92.3	78.7	90.0	42.9	45.0	7.5	10.2

表 3-10　本节检测模型与反病毒软件的误报率(%)比较

样本集	本节模型	AV1	AV2	AV3	AV4	AV5	AV6	AV7	AV8	AV9
2000	5.0	0.5	0.5	0	0	0	0.5	0.5	0	0
4000	1.4	1.6	0.9	0	0	0	0	1.2	0	0

3.5　本章小结

本章提出了一种 Android 静态检测方法，采用逆向工程将 DEX 文件转化为 Dalvik 指令并对其进行简化抽象，再将抽象后的指令序列进行 N-Gram 编码作为样本训练，通过机器学习的分类算法来创建分类检测模型。在实验环节，对不同分类算法与 N-Gram 序列分别进行了比较，提出了基于 3-Gram 编码和 Random Forest 算法的优选检测方法。最后，采用 2000 个样本与 4000 个样本两组样本集，与 9 款专业的反毒软件进行对比实验，实验结果表明该方法可有效进行 Android 恶意代码检测，且获得较高的检测率。

第 4 章　基于 bytecode 生成图的恶意家族分类方法设计

4.1　系统模型设计

本章的基本思路是将 Android 恶意应用的 Dalvik bytecode 转化为一幅 256 阶灰度图形式的 bytecode 生成图，然后用 GIST 算法提取图像的纹理特征，最后再结合 Random Forest 算法对特征进行分类。该方法的系统模型如图 4-1 所示，将 APK 中的 DEX 文件提取出来，该 DEX 文件包含了 Dalvik bytecode，可以在 Dalvik 虚拟机中执行。然后，将 bytecode 构建成一个合适宽度的像素点矩阵，因为矩阵中每一个像素点为一个字节，一个字节的取值范围从 0x00 至 0xFF，即有 256 种变化刚好对应 256 阶灰度，可以由像素点矩阵可生成 png、jpeg 等格式的 bytecode 生成图。最后，利用 GIST 特征算法提取出该 bytecode 生成图中的纹理特征，得到一个 512 维的特征向量，最后采用 Random Forest 分类算法对 GIST 特征进行分类。

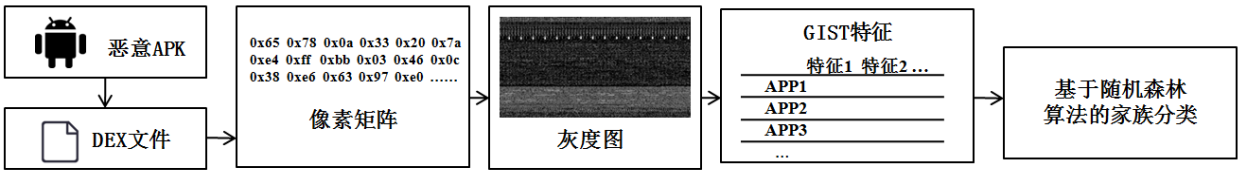

图 4-1　系统模型图

4.2　bytecode 生成图原理

APK 格式的文件通常都包含了一个名为 classes.dex 的文件，Dalvik 虚拟机所能执行的 Dalvik bytecode 包含在该 DEX 文件内。DEX 文件的结构由几部分结构体构成。dex header 是 dex 的文件头，dex 文件属性和其他部分的结构在 dex 文件中的偏移量由 dex 的文件头指定。索引结构区由 string_ids、type_ids、proto_ids、field_ids、method_ids、class_def 等部分构成。真实的数据存放区称为的 data section[47]。

将 dex 文件的 bytecode 构建成一个合适宽度的像素点矩阵，因为矩阵中每一个像素点恰好为一个字节，一个字节的取值范围从 0x00 至 0xFF 即有 256 种变化刚好对应一张灰度图形式的 bytecode 生成图。图 4-2 显示了一个 Android 应用中的 DEX 文件结构与 bytecode 生成图的一一对应关系，bytecode 生成图的纹理特征是对 bytecode 的数据结构与数据的一种综合反映。

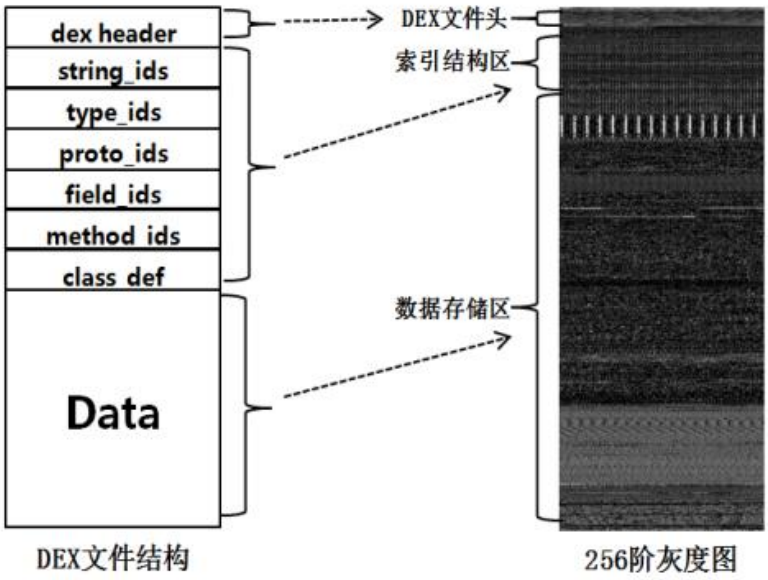

图 4-2　DEX 文件结构与 bytecode 生成图的对应关系

因为恶意软件的作者在制作生成恶意软件的一系列过程中，往往会采用模块重用技术或者自动化工具来快速编写新的变种。这些很大程度上共用了祖先的相同代码，有一定内在的关联性、相似性的恶意代码合称为恶意代码家族。因而同一个家族的恶意代码的 bytecode 生成图的纹理也会表现出一定的相似性和关联性。如图 4-3 所示，DroidKungFu、Plankton、Opfake、Kmin 等 4 个恶意家族分别有 5 个变种，同恶意家族的 bytecode 生成图的相似性要远远大于不同恶意家族的 bytecode 生成图的相似性。如图 4-4 所示， 14 种恶意家族较为有代表性的 bytecode 生成图，可以观察到各恶意家族的 bytecode 生成图有较明显区分的视觉特点。

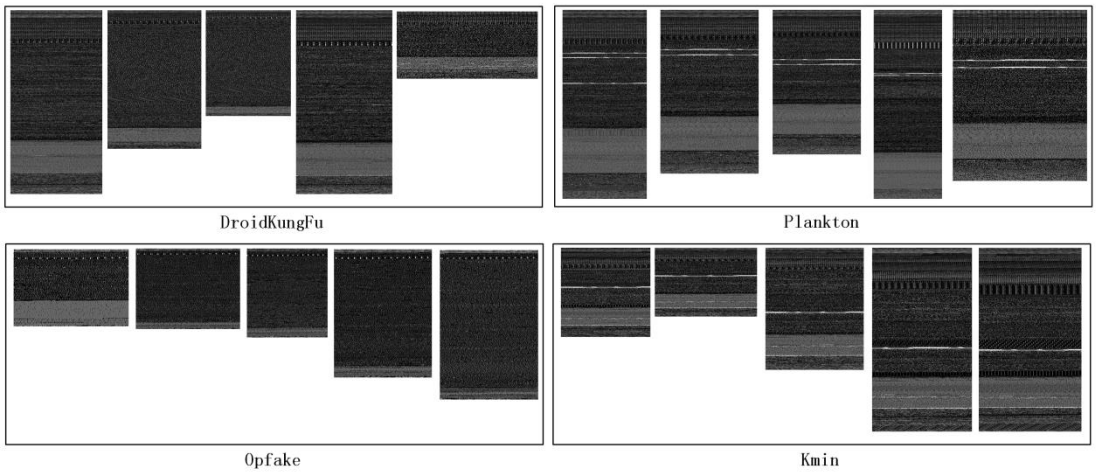

图 4-3　同恶意家族的 bytecode 生成图

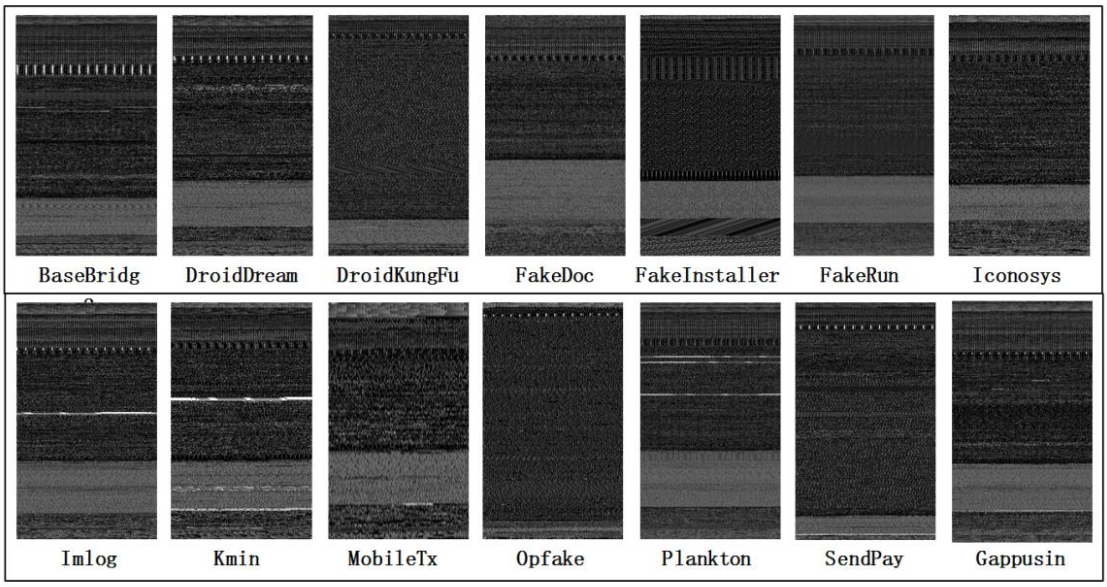

图 4-4　不同家族的 bytecode 生成图

4.3　bytecode 生成图特征的提取

图像的纹理特征提取方法一般包括 GIST 模型、GLCM、Gabor 滤波法等多种方法。本章采用较为经典的 GIST 模型[48],该算法常用于图像识别系统等领域。利用 GIST 模型提取 bytecode 生成图特征的过程：

1.按照 bytecode 生成图的像素点数，计算生成一个宽度与高度相同的像素点矩阵，矩阵的数据点就是生成图的像素点信息。

2. 利用矩阵可以反过来生成一个正方形图像，通过计算将正方形图像划分成 4×4 个互不重叠的均匀子区域。

3.每个子图像进行 8 个方向与 4 个尺度的小波滤波提取图像的纹理特征。最终每一幅图像获得一个 16×8×4=512 维的特征向量。

采用上述 3 个步骤对 Android 恶意代码家族的样本进行了特征提取，并将得到的高维数据用 t-SNE[49]。进行降维与数据可视化。如图 4-5 所示，GIST 模型进行特征提取有较明星的分类效果。

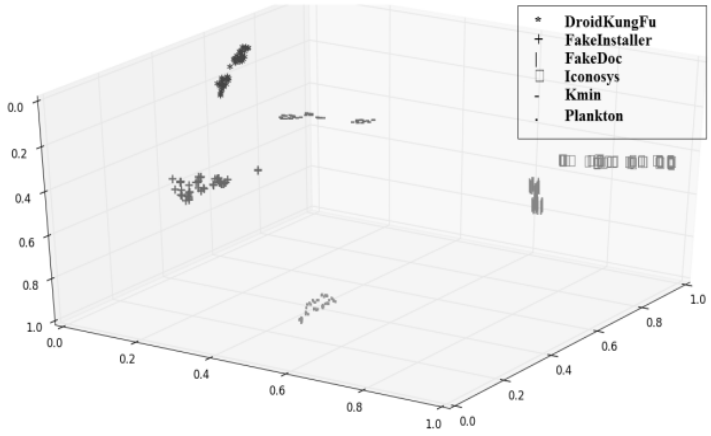

图 4-5　GIST 特征的降维与数据可视化

4.4　实验与结果分析

4.4.1　Random Forest 算法的选用

Random Forest 算法是机器学习中一种常见的分类算法，它具有较高的准确率，不容易陷入过拟合，具有很好的抗噪声能力，能够处理很高维度的特征数据，所以在生物信息、统计学、计算机等领域有着广泛的应用。该算法以随机的方式建立一个由决策树构成的森林，Random Forest 的每一棵决策树之间并不产生关联关系。在生成 Random Forest 之后，当有一个新的输入样本进入算法时，森林中的每一棵决策树都会分别进行一下判断，判断这个样本应该属于哪一分类，最终以投票的形式决定属于哪一类别[50]。

Random Forest 算法非常适用于恶意家族样本集的分类，因为该算法在处理多种类型数据时可以产生高准确度，对那些不平衡的样本集可以平衡误差，所以本章的实验的分类算法选用 Random Forest 算法。

4.4.2 实验数据来源

本章实验部分的恶意移动应用来源于德国哥廷根大学 Drebin 项目**错误!未找到引用源。**。在该样本数据库中选取 14 类常见的 Android 恶意家族作为本章的实验对象，家族名与对应的样本数如表 4-1 所示。

表 4-1 Android 恶意家族与样本数量

家族名称	样本数
FakeInstaller	925
DroidKungFu	667
Plankton	625
Opfake	613
BaseBridge	330
Iconosys	152
Kmin	147
FakeDoc	132
DroidDream	81
MobileTx	69
FakeRun	61
SendPay	59
Gappusin	58
Imlog	43

4.4.3 恶意家族分类实验

对 14 类家族 3962 个恶意样本利用 GIST 模型提取 bytecode 生成图特征，结合使用 Random Forest 算法以 10 折交叉验证方式进行评估。结果如表 4-2 所示，因为 Gappusin 恶意家族由于来源的样本数量太少导致 TPR 值不高，其他恶意家族的 TPR 值都高于 0.8；平

均 TPR 值要高于 90%，平均 FPR 值约为 0.6%，表明该方法对 Android 恶意家族分类有较不错的效果。

表 4-2　Android 恶意家族分类效果表

恶意家族	TPR	FPR	Precision	Recall	F-Measure	AUC
BaseBridge	0.839	0.002	0.965	0.839	0.898	0.984
DroidDream	0.827	0.002	0.882	0.827	0.854	0.983
DroidKungFu	0.934	0.057	0.733	0.934	0.821	0.989
FakeDoc	0.962	0.001	0.977	0.962	0.969	0.995
FakeInstaller	0.969	0.011	0.956	0.969	0.962	0.999
FakeRun	0.951	0	1	0.951	0.975	1
Iconosys	0.961	0.004	0.885	0.961	0.921	0.999
Imlog	0.837	0	1	0.837	0.911	0.984
Kmin	0.98	0.001	0.973	0.98	0.976	1
MobileTx	1	0	0.986	1	0.993	1
Opfake	0.977	0.008	0.951	0.977	0.964	0.999
Plankton	0.974	0.005	0.965	0.974	0.97	0.998
SendPay	0.915	0.001	0.947	0.915	0.931	0.989
Gappusin	0.534	0	1	0.534	0.697	0.954

在相同实验样本集上，本章方法与 Drebin 检测方法进行了实验对比，如图 4-6 所示，两种检测方法在检测 Android 恶意家族的表现上互有优劣，总体平均效果基本接近。对于 Gappusin 家族的检测，两种方法的检测率都不高，但是用本章方法达到的检测效果还略好于 Drebin 检测方法。另外，Drebin 检测方法的平均误报率约为 1%，而本章方法的平均误报率约为 0.6%，仍略优于 Drebin 检测方法。

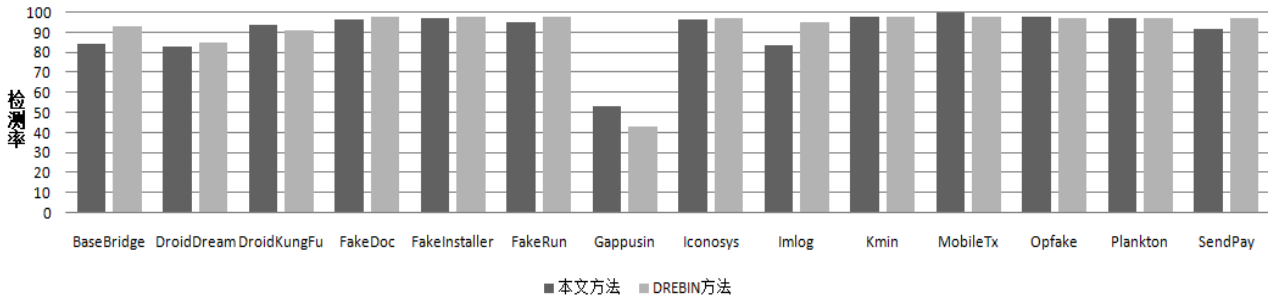

图 4-6　两种检测方法的对比

4.5　本章小结

本章研究了 DEX 文件结构与 bytecode 生成图纹理之间的对应关系，提出了一种基于 bytecode 生成图的 Android 恶意代码家族分类方法，通过将 Android 恶意应用的 Dalvik bytecode 转化为 256 阶灰度图形式的 bytecode，利用 GIST 算法提取图像的纹理特征，并结合 Random Forest 算法对特征进行分类。对常见的 14 种 Android 恶意代码家族的样本数据进行了实验验证，并与 Drebin 检测方法进行了比较，实验结果表明该方法可有效进行 Android 恶意代码家族分类，检测精度较高且误报率较低，效果略优于 Drebin 检测方法。

第 5 章　Android 恶意代码智能检测平台的设计与实现

5.1　恶意代码检测系统的应用场景

目前，市场上出现了很多安卓应用商店，除了官方的 GooglePlay 以外**错误!未找到引用源。**，还有很多第三方的安卓应用商店，国内安卓应用商店中较为著名有腾讯应用宝、360 手机助手、小米应用商店、百度手机助手、91 手机助手、豌豆荚、安智市场、历趣市场、沃商店等十几家[51]，另外还存在不少不知名的安卓应用商店，所以安卓应用在程序发布上较为混乱，呈现一个松散的架构，没有一个统一管理的发布源。

安卓系统由于其开源的特点，攻击者很容易制作含有恶意代码的程序并放到网上的应用商店中，一旦有用户下载并安装了这些含有恶意代码的程序，用户的隐私就可能被恶意程序窃取，甚至造成经济上损失。本章所设计的恶意代码智能检测系统模型所适用的应用场景如图 5-1 与图 5-2 所示。在图 5-1 中，在安卓应用商店的服务器一端部署一个检测系统，在安卓开发人员上传应用程序之后，用户手机下载之前对应用程序进行审核，可以有效的保障用户的安全性。在图 5-2 中，建立了一个提供可疑 APK 文件分析服务的 Web 网站，给有需要的用户或者网络安全人员用于在线上传可疑的安卓应用程序，当服务网站检测完可疑 APK 文件后可生成一份可靠的检测报告供用户参考。该检测报告可以包含 APK 各种详细信息、可能存在的风险漏洞等。

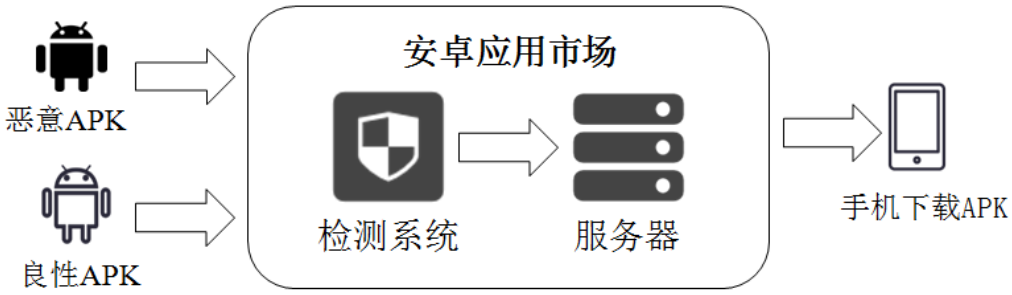

图 5-1　检测系统的应用场景一

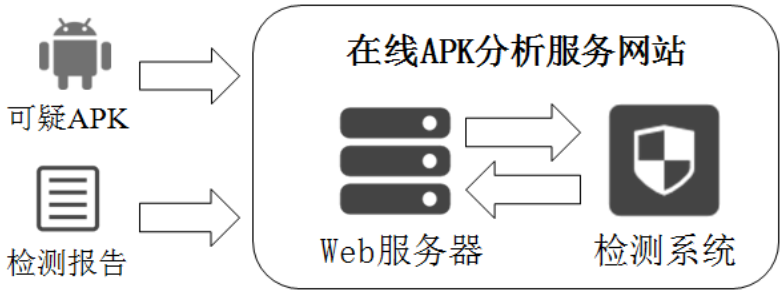

图 5-2　检测系统的应用场景二

5.2　恶意代码智能检测平台的总体架构

本章以在线 APK 分析服务的应用场景为例，设计并实现了一个 Android 恶意代码智能检测平台。

Android 恶意代码智能检测平台总体架构主要分为两大功能模块：Web 交互服务模块和恶意代码检测模块。Web 交互服务模块的主要功能是实现与用户的交互功能，包括可疑 APK 上传、检测报告生成、训练数据的数据可视化展示和日志查询等数个功能子模块。恶意代码检测模块的主要功能是实现恶意代码检测的功能，包括基本信息采集、文件预处理、恶意代码判定、检测模型生成等数个功能子模块。一方面，两大功能模块之间有数据的传递，另一方面，两大功能模块又能各自独立完成一定的功能，满足特定的需求。检测平台的总体架构设计如图 5-3 所示。

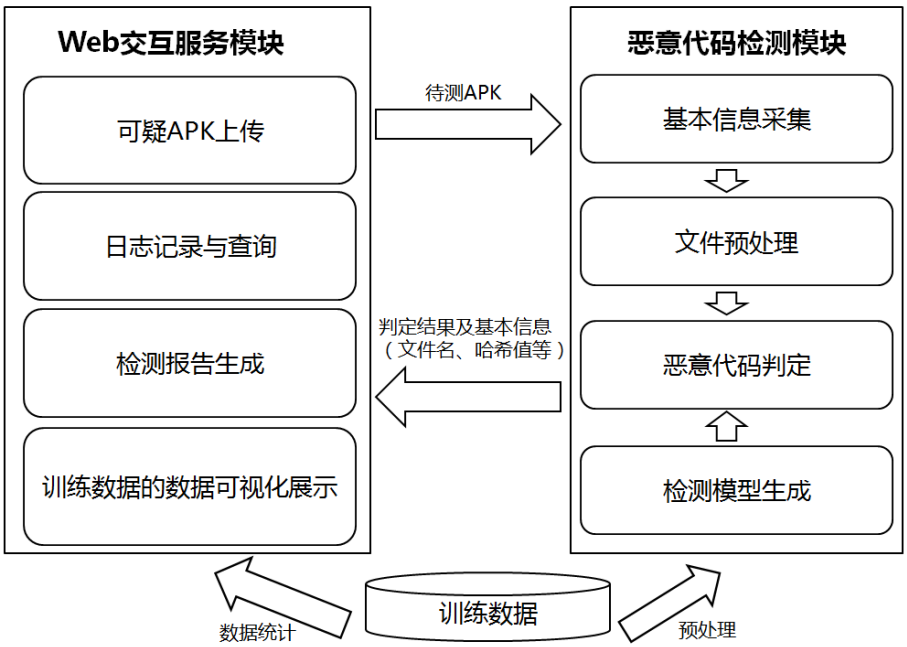

图 5-3　检测平台的总体架构设计图

整个检测平台的数据流程总体上可以分为两大部分，一部分为训练数据的数据流程，另一部分为测试数据的数据流程。两部分的数据流程贯穿了整个架构的所有功能子模块，影响检测平台的有效运行。

5.2.1　训练数据的数据流程

检测平台的训练数据的数据流程按照如下步骤进行：

a）检测模型的生成。将 2400 个训练样本（其中包括 1200 个恶意样本与 1200 个良性样本）进行反汇编并按照本文提到的简化指令分类方案，提取指令符号。结合较优效果的 Random Forest 算法，用 3-Gram 编码指令符号生成了一个恶意代码检测模型 model 文件。该文件将在恶意代码判定的阶段用到。

b）训练数据的数据可视化展示。训练数据分为恶意程序与良性程序，其中恶意程序来源于德国哥廷根大学 Drebin 项目的恶意样本数据库，有 179 种恶意家族分类。而良性程序来自 GooglePlay 应用商店，涉及 24 个门类的移动应用程序。监测平台将这些训练数据进行统计，基于 D3.js 的技术以可视化的方式呈现给用户，让用户对监测平台的数据来源与分布有一个更直观的了解。

5.2.2 测试数据的数据流程

检测平台的测试数据的数据流程按照如下步骤进行：

a）用户上传可疑 APK。当用户对某个 APK 的安全性存在疑问，可通过访问检测平台的 APK 上传模块，通过 HTTP POST 方式上传可疑 APK 到检测平台。另外，平台对上传的文件类型做了限定，用户只能上传 APK 类型的文件。

b）基本信息采集。进入平台后的待测 APK 首先将被基本信息采集模块进行一次基本的信息采集，采集的基本信息包括文件名、文件大小、检测日期、文件的 MD5 值、文件的 SHA-1 值、文件的 SHA-256 值、来源 IP 等多项值。这些值将作为最终检测报告的组成部分呈现给用户。

c）文件预处理。通过基本信息采集后的待测 APK，将进入文件预处理阶段。在这个阶段，待测 APK 首先将被反汇编成为 Smali 格式的文件，对于部分无效格式的 APK 文件在进行反汇编时候如果出现失败的情况，平台将进行出错处理，以检测失败告终。若反汇编成功，将得到的 Smali 文件将按照本文提到的简化指令分类方案，提取指令符号。然后，将指令符号进行 3-Gram 进行编码，生成 3-Gram 的指令符号序列。最后，通过 3-Gram 的特征个数并进行归一化处理等方式，得到一份 ARFF 文件，该文件可以被 Weka 软件识别和调用。

d）恶意代码判定。在上一步的文件预处理阶段，已经得到了待测 APK 的 ARFF 格式文件。另外，平台已经事先通过 2400 个训练样本结合 Random Forest 算法，用同样的 3-Gram 编码生成了一个恶意代码检测模型 model 文件。在本阶段，检测平台将只需要调用 Weka 软件的 API 接口，以待测 APK 的 ARFF 文件、恶意代码检测模型 model 文件和 Random Forest 算法作为参数，可最终得到的一个是否是恶意代码的预测结果。通过本文的实验环节不难发现，该预测结果的准确率较高，所以将该预测结果作为恶意代码判定结果。

e）检测报告生成与日志查询。在这个阶段，检测平台将进行汇总恶意代码判定结果和 APK 基本信息，以检测报告的形式呈现给用户，并提供用户打印与下载等功能。另外，检测平台将把每次的检测报告以日志的形式进行记录备案，提供给管理员用户进行查询与审计。

综上，本章设计并实现了一个 Android 恶意代码智能检测平台，检测平台总体架构主要分为两大部分：Web 交互服务模块和恶意代码检测模块。Web 交互服务模块的主要功能是实现与用户的交互功能，包括可疑 APK 上传、检测报告生成、训练数据的数据可视化

展示和日志查询等数个功能子模块。恶意代码检测模块的主要功能是实现恶意代码检测的功能，包括基本信息采集、文件预处理、恶意代码判定、检测模型生成等数个功能子模块。最后，检测平台的数据流程总体上可以分为两大部分，一部分为训练数据的数据流程，另一部分为测试数据的数据流程，并分步骤对数据流程进行了详细的介绍。

5.3 Web 交互服务模块的设计与实现

Web 交互服务模块的主要功能是实现与用户的交互功能，它又可以分为 4 个子功能模块，分别是可疑 APK 上传子模块、训练数据的数据可视化展示子模块、检测报告生成子模块和日志记录与查询子模块。Web 交互服务模块的架构设计如图 5-4 所示。

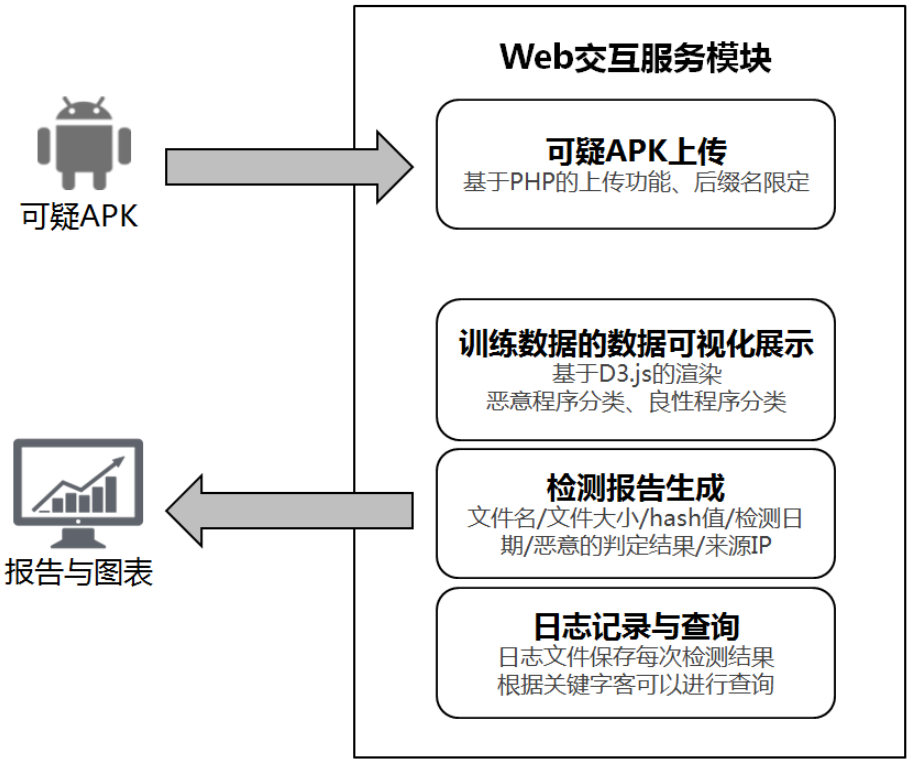

图 5-4 Web 交互服务模块的架构设计图

另外，Web 交互服务模块利用 Apache2.4 作为 Web 服务器软件，并利用 HTML5、CSS3、JavaScript、PHP 等多种 Web 技术，实现 Web 交互的多种功能。

5.3.1 可疑 APK 上传子模块

当用户访问检测平台的 APK 上传模块时候，检测平台会呈现一个文件上传界面；用户可通过界面进行操作，点击上传本地的文件，并通过 HTTP POST 方式将可疑文件上传到检测平台。另外，平台还可以对上传的文件类型与文件大小进行一些限定。后台的 PHP 程序在得到上传调用后，将会把可疑文件存放到平台的指定目录，待进一步处理。该子功能模块的界面如图 5-5 所示。

图 5-5 可疑 APK 上传子模块的界面图

5.3.2 数据可视化展示子模块

训练数据分为恶意程序与良性程序，其中恶意程序来源于德国哥廷根大学 Drebin 项目的恶意样本数据库，有 179 种恶意家族分类，每个分类又有不同数量的样本。同时，良性程序来自 GooglePlay 应用商店，涉及 24 个门类的移动应用程序。

D3.js[52]是一个著名的 JavaScript 库，它可以通过数据来操作文档。D3（Data-Driven Documents）可以通过使用 HTML、SVG 和 CSS 等 Web 技术把数据以鲜活形象地展现出来，使用 D3 就可以让数据可视化的效率和丰富程度产生指数化的增长。监测平台首先将这些训练数据进行统计，然后利用 D3.js 的数据可视化技术以可视化的方式呈现给用户，让用户对监测平台的数据来源与分布有一个更加直观的了解。该子功能模块的界面如图 5-6、图 5-7 所示。

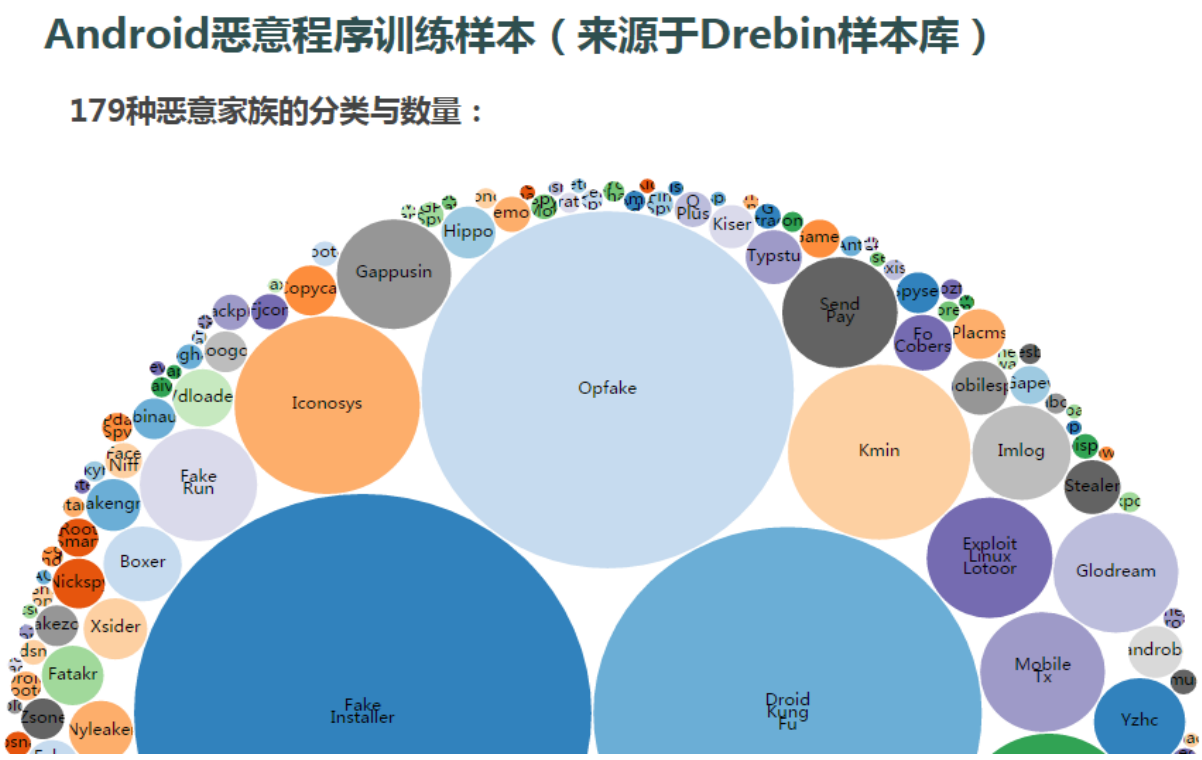

图 5-6　恶意程序训练样本的数据可视化

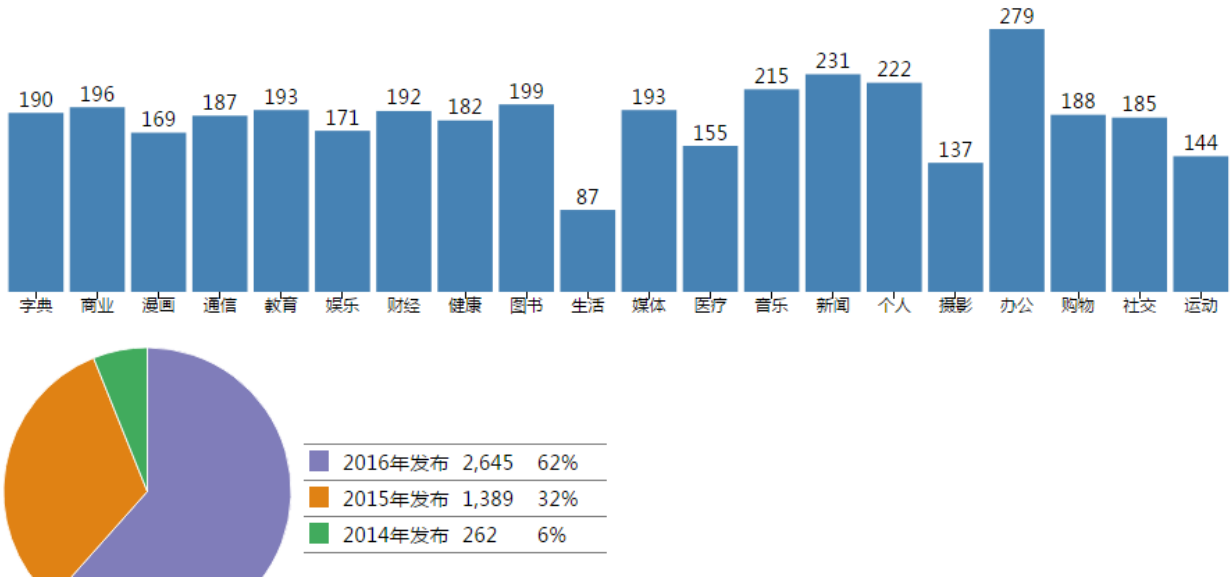

图 5-7　良性程序训练样本的数据可视化

5.3.3 检测报告生成子模块

在完成恶意代码检测后，检测报告生成子模块将汇总恶意代码判定结果和 APK 基本信息，以检测报告的形式呈现给用户。报告中除了有最重要的检测结果一项信息，还将包括文件名、文件大小、检测日期、文件的 MD5 值、文件的 SHA-1 值、文件的 SHA-256 值、来源 IP 等多项信息。用户除了可以查看检测报告，还能对检测报告进行打印、下载。该子功能模块的界面如图 5-8 所示。

图 5-8 检测报告生成子模块的界面图

5.3.4 日志记录与查询子模块

日志记录与查询子模块将会把每次检测完的检测报告以日志的形式进行记录备案，提供给管理员用户进行查询与审计。该子功能模块的界面如图 5-9 所示。该功能模块还有以文件名作为关键字可进行检索的功能，如图 5-10 所示，以“m2.apk”为关键字进行检索得到三条记录，它们分别是不同检测日期生成的记录。

文件名	文件大小	MD5	检测日期	来源IP	检测结果
m2.apk	11.56640625 Kb	336696ad7c105ce710d2192f14f1a698	2017-03-07 06:18:02pm	127.0.0.1	恶意APP
m10.apk	11.513671875 Kb	511dfef35d75062406a6d20368e0eaeb	2017-03-07 06:22:16pm	127.0.0.1	恶意APP
g4.apk	21.6435546875 Kb	3b765fcbd3ca65ffd08cb5e54d3a3b87	2017-03-07 06:35:05pm	127.0.0.1	非恶意APP
g6.apk	15.751953125 Kb	0026de5f8e18eb0b41a1d3c18c940e07	2017-03-07 07:28:50pm	127.0.0.1	非恶意APP
m2.apk	11.56640625 Kb	336696ad7c105ce710d2192f14f1a698	2017-03-09 12:37:29am	127.0.0.1	恶意APP

Search apk name: 提交

图 5-9　日志记录的界面图

文件名	文件大小	MD5	检测日期	来源IP	检测结果
m2.apk	11.56640625 Kb	336696ad7c105ce710d2192f14f1a698	2017-03-07 06:18:02pm	127.0.0.1	恶意APP
m2.apk	11.56640625 Kb	336696ad7c105ce710d2192f14f1a698	2017-03-09 12:37:29am	127.0.0.1	恶意APP
m2.apk	11.56640625 Kb	336696ad7c105ce710d2192f14f1a698	2017-03-09 03:34:49pm	127.0.0.1	恶意APP

Search apk name: m2.apk 提交

图 5-10　日志查询的界面图

5.4　恶意代码检测模块的设计与实现

恶意代码检测模块的主要功能是实现恶意代码检测的功能，它又可以分为 4 个子功能模块，分别是基本信息采集子模块、文件预处理子模块、恶意代码判定子模块、检测模型生成子模块。恶意代码检测模块的架构设计如图 5-11 所示。

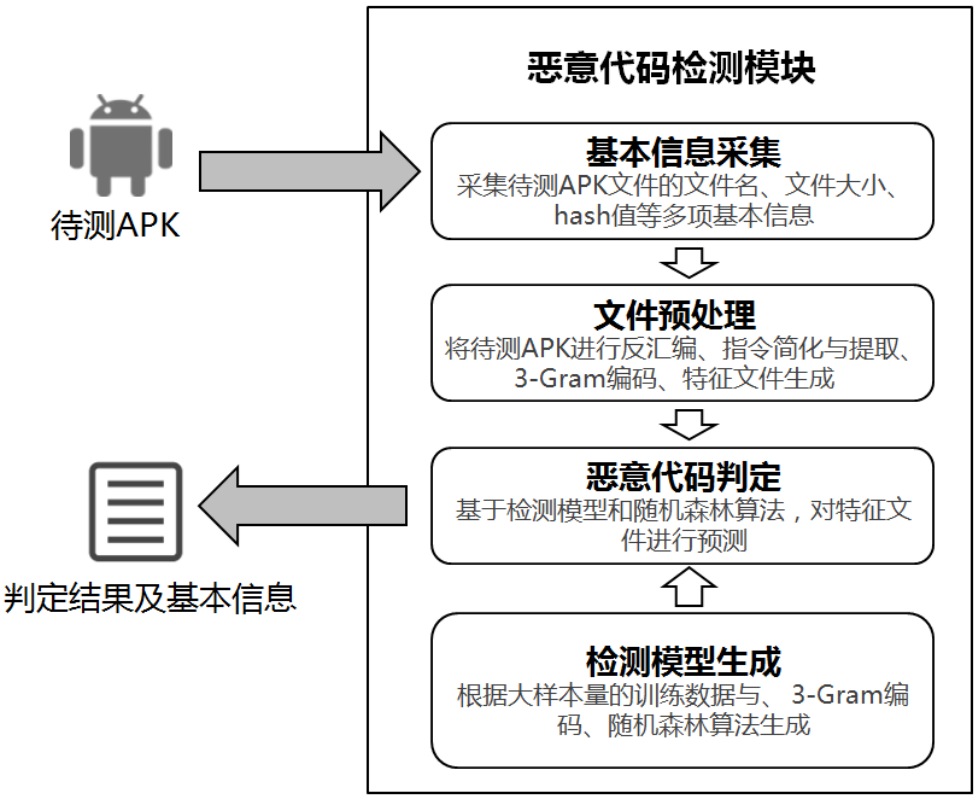

图 5-11　恶意代码检测模块的架构设计图

另外，恶意代码检测模块利用 C++、Java、PHP 等多种技术，通过 C++编程实现基本信息采集、文件预处理的整个过程，最后编写 Java 程序调用数据挖掘软件 Weka 的 API 实现检测模型的生成与恶意代码判定。

5.4.1　基本信息采集子模块

基本信息采集子模块采集待测 APK 的基本信息，采集的基本信息包括文件名、文件大小、检测日期、文件的 MD5 值、文件的 SHA-1 值、文件的 SHA-256 值、来源 IP 等多项信息。其中，文件名是待测 APK 的全名，文件大小则按照 KB 为基本单位。待测文件的 MD5 值、SHA-1 值和 SHA-256 值需要分别调用 PHP 的 md5_file 函数、sha1_file 函数、hash_file 函数来获得文件的散列值。来源 IP 是可疑代码上传者访问检测平台时候显示的 IP 地址，这也将作为一个参考信息进行采集并备案。检测日期是检测平台开始对 APK 文件进行检测时的具体日期及时间，时区按照东八区设置。以上所有这些信息将作为最终检测

报告的组成部分，与恶意代码判定结果一起呈现给用户。如图 5-12 所示，是该模块的关键代码。

```
$app_name=$_FILES["files"]["name"];//获得文件名
$app_size=($_FILES["files"]["size"] / 1024) . " Kb";//获得文件大小
$app_md5=md5_file("upload/" . $_FILES["files"]["name"]);//md5值
$app_sha1=sha1_file("upload/" . $_FILES["files"]["name"]);//sha1值
$app_sha256=hash_file("sha256","upload/" . $_FILES["files"]["name"]);//sha256值
$app_ip=$_SERVER['REMOTE_ADDR'];//来源IP
date_default_timezone_set("Asia/Shanghai");//设置时区
$app_time=date("Y-m-d h:i:sa"); //检测日期与时间
```

图 5-12 基本信息采集子模块的关键代码

5.4.2 文件预处理子模块

文件预处理子模块直接使用 apktool 工具将待测 APK 被反汇编成为 Smali 格式的文件。对于部分无效格式的 APK 文件在进行反汇编时候如果出现反汇编失败，平台将按照出错的处理流程，以恶意代码检测失败告终，并告知用户。若反汇编 Smali 文件成功，将得到的 Smali 文件将按照本文第 3 章提到的简化指令分类方案，对 Smali 文件提取指令符号。然后，将指令符号进行 3-Gram 进行编码，生成 3-Gram 的指令符号序列。最后，通过统计 3-Gram 的特征个数并进行归一化处理等多种方式，得到一份 ARFF 格式的特征文件，该文件格式由于可以被 Weka 软件识别和调用，所以可用于恶意代码判定子模块。如图 5-13 所示，是一份典型的 ARFF 格式的特征文件。

```
@attribute VTP numeric
@attribute VTV numeric
@attribute VPM numeric
@attribute VPR numeric
@attribute VPG numeric
@attribute VPI numeric
@attribute VPT numeric
@attribute VPP numeric
@attribute VPV numeric
@attribute VVM numeric
@attribute VVR numeric
@attribute VVG numeric
@attribute VVI numeric
@attribute VVT numeric
@attribute VVP numeric
@attribute VVV numeric
@attribute class {malware,normal}

@data
0.007246,0.001812,0.000000,0.000000,0.000000,0.001812,0.000000,0.005435,0.001812,0.000000,0.001812,
```

图 5-13 ARFF 格式的特征文件

5.4.3 恶意代码判定子模块

恶意代码判定子模块将调用 Weka 软件的 API 接口，以 ARFF 格式的特征文件、恶意代码检测模型 model 文件和 Random Forest 算法作为基本参数，通过计算可得到的一个预测结果，该预测结果预测 ARFF 格式的特征文件是否是恶意特征，进而表明生成该 ARFF 格式的特征文件的待测 APK 是否是恶意程序。通过本文的第三章的实验环节不难发现，该方法在 4000 个样本情况下可达到 98.6%的检测率，与常见杀毒软件相比较也有较高的排名，所以将该预测结果直接作为恶意代码判定结果是可行的。如图 5-14 所示，是该模块预测结果的关键代码。

```
package weka_malware;

import weka.gui.SimpleCLIPanel;

public class SimpleCLI {
    public static void main(String[] args) throws Exception {
    SimpleCLIPanel simple =new SimpleCLIPanel();
    simple.main(null);
    simple.runCommand("java weka.classifiers.trees.RandomTree -p 9 -l D:/web4paper/1.model"
            + "-T D:/web4paper/test.arff >  D:/web4paper/result.txt");
    }
}
```

图 5-14 恶意代码判定子模块的关键代码

5.4.4 检测模型生成子模块

检测模型生成子模块将 2400 个训练样本（其中包括 1200 个恶意样本与 1200 个良性样本）进行反汇编并按照本文提到的简化指令分类方案，提取指令符号。结合较优效果的 Random Forest 算法与 3-Gram 编码的指令符号，生成了一个恶意代码检测模型。该模型可以以 model 文件保存，并将在恶意代码判定的阶段用到。如图 5-15 所示，是该模块生成模型的关键代码。

```
package weka_malware;

import weka.gui.SimpleCLIPanel;

public class SimpleCLI {
    public static void main(String[] args) throws Exception {
    SimpleCLIPanel simple =new SimpleCLIPanel();
    simple.main(null);
    simple.runCommand("java weka.classifiers.trees.RandomTree -t  "
            + "D:/web4paper/2cy1200.arff -d D:/web4paper/1.model");
    }

}
```

图 5-15　检测模型生成子模块的关键代码

5.5　恶意代码智能检测平台的功能测试

恶意代码智能检测平台设计并实现完成后，需要进行一定的功能测试，鉴定检测平台的正确性、健壮性、安全性。由于需要测试的项目较多，本节就以异常文件处理功能的测试和恶意代码检测功能的测试为例，进行重点介绍。

5.5.1　异常文件处理功能的测试

恶意代码智能检测平台只能对 Android 的移动应用 APK 文件进行检测，所以必须判断用户上传的文件格式是否正确。对于用户上传的有问题格式的安卓应用，平台能够进行有效处理，提升平台的健壮性。

首先，通过在文件上传界面试图上传非 apk 后缀名的文件，可以看到上传界面返回一个“无效的文件类型，请上传 apk 后缀名的文件”的警告信息，并及时阻止了用户上传非法格式的文件。阻止与警告信息的界面如图 5-16 所示。

图 5-16　阻止与警告信息的界面图

然后，上传一个文件内容并非 Android 应用程序，但是后缀名是 apk 的文件。该文件虽然可以通过上传模块上传到检测平台，但是在文件预处理子模块进行反汇编时候会出错。检测平台及时停止了进一步的检测，并返回了一份带有检测出错标识的报告，如图 5-17 所示。

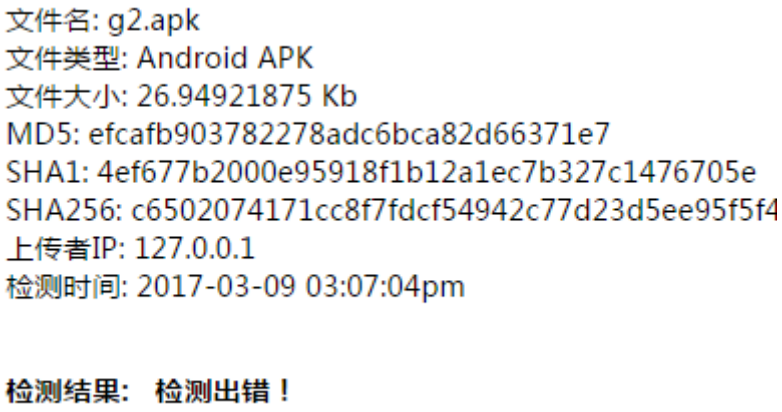

图 5-17　带检测出错标识的报告

5.5.2　恶意代码检测功能的测试

恶意代码检测功能的测试分为两部分，第一部分是上传多个恶意的程序文件，观察检测平台能否正确识别上传的移动应用为恶意程序；第二部分是上传多个良性的程序文件，观察检测平台能否正确识别上传的移动应用为良性程序。

首先，随机选择 10 个恶意程序文件作为测试样本，重新命名为 m1.apk 至 m10.apk，然后分别上传到检测平台，检测平台能够正确识别出恶意程序并进行日志记录，结果如图 5-18 所示。

m1.apk	5.265625 Kb	860a3cd4cad692d8628d8cc84eb07519	2017-03-09 03:34:41pm	127.0.0.1	恶意APP
m2.apk	11.56640625 Kb	336696ad7c105ce710d2192f14f1a698	2017-03-09 03:34:49pm	127.0.0.1	恶意APP
m3.apk	11.515625 Kb	de963efa14d28c34a51e2935f6244e95	2017-03-09 03:35:11pm	127.0.0.1	恶意APP
m4.apk	11.541015625 Kb	2a09671e0b3d848a2ac8152838eda683	2017-03-09 03:35:26pm	127.0.0.1	恶意APP
m5.apk	11.5263671875 Kb	760305a489308b17480402e506d54add	2017-03-09 03:35:42pm	127.0.0.1	恶意APP
m6.apk	11.330078125 Kb	7cba9ab0dc8c25ba8c19a1d5567b5b73	2017-03-09 03:35:54pm	127.0.0.1	恶意APP
m7.apk	11.515625 Kb	5254f1aa8d3d8c5c3698b29d3c4d3f3d	2017-03-09 03:36:05pm	127.0.0.1	恶意APP
m8.apk	11.5029296875 Kb	ee5afb10fdddeba452e37fc24144e582	2017-03-09 03:36:19pm	127.0.0.1	恶意APP
m9.apk	9.9013671875 Kb	84fa2ea103a7e3475adcf1d81c33e54d	2017-03-09 03:36:31pm	127.0.0.1	恶意APP
m10.apk	11.513671875 Kb	511dfef35d75062406a6d20368e0eaeb	2017-03-09 03:36:44pm	127.0.0.1	恶意APP

图 5-18　恶意程序测试的日志记录

首先，随机选择 10 个良性程序文件作为测试样本，重新命名为 g1.apk 至 g10.apk，然后分别上传到检测平台，检测平台能够正确识别出良性程序并进行日志记录，结果如图 5-19 所示。

g1.apk	30.0009765625 Kb	e6c9ea98c63ec715f5dd38d7f3d620d9	2017-03-09 03:53:51pm	127.0.0.1	非恶意APP
g2.apk	26.94921875 Kb	efcafb903782278adc6bca82d66371e7	2017-03-09 03:54:06pm	127.0.0.1	非恶意APP
g3.apk	25.5107421875 Kb	25c2ebf3721e8b7b13a0acd691eb5817	2017-03-09 03:54:17pm	127.0.0.1	非恶意APP
g4.apk	21.6435546875 Kb	3b765fcbd3ca65ffd08cb5e54d3a3b87	2017-03-09 03:54:36pm	127.0.0.1	非恶意APP
g5.apk	10.5234375 Kb	6ae53d2aa0278e8f6f7ee33a5ff18067	2017-03-09 03:54:50pm	127.0.0.1	非恶意APP
g6.apk	15.751953125 Kb	0026de5f8e18eb0b41a1d3c18c940e07	2017-03-09 03:55:02pm	127.0.0.1	非恶意APP
g7.apk	22.732421875 Kb	93786634bff6ce7a239e448a0c4c3305	2017-03-09 03:55:15pm	127.0.0.1	非恶意APP
g8.apk	13.7314453125 Kb	bc628cf81468c09f1df8437d026a9a6e	2017-03-09 03:55:29pm	127.0.0.1	非恶意APP
g9.apk	10.6962890625 Kb	2fea001c0b15e549d501eaa24a978fb9	2017-03-09 03:55:43pm	127.0.0.1	非恶意APP
g10.apk	28.083984375 Kb	6b204e61a7a9142e31b4a66da229d3fe	2017-03-09 03:55:56pm	127.0.0.1	非恶意APP

图 5-19　良性程序测试的日志记录

5.6　本章小结

本章从在线 APK 分析服务的应用场景为例，设计并实现了一个 Android 恶意代码智能检测平台。Android 恶意代码智能检测平台总体架构主要分为两大功能模块：Web 交互服务模块和恶意代码检测模块。Web 交互服务模块的主要功能是实现与用户的交互功能，包括可疑 APK 上传、检测报告生成、训练数据的数据可视化展示和日志查询等数个功能子模块。恶意代码检测模块的主要功能是实现恶意代码检测的功能，包括基本信息采集、文

件预处理、恶意代码判定、检测模型生成等数个功能子模块。首先，先介绍了整个检测平台的数据流程，包括训练数据的数据流程和测试数据的数据流程，并分步骤对数据流程进行了详细的介绍。然后，分别对两大功能模块的八个功能子模块从功能设计和技术实现角度进行了详细的阐述。最后，还对检测平台进行了功能测试，鉴定检测平台的正确性、健壮性、安全性，并以异常文件处理功能的测试和恶意代码检测功能的测试为例，进行了重点介绍。

第 6 章　结论与展望

6.1　结论

本文对 Android 恶意移动应用的静态检测方法进行了深入而详细的研究，主要做了如下多个方面工作：

（1）介绍相关理论知识与关键技术，包括 Android 系统的相关知识、N-Gram 编码原理、常见的聚类算法及 Random Forest 算法、常见的分类算法及 AP 聚类算法、图像纹理特征提取算法等。

（2）提出了一个基于 Dalvik 指令简化的恶意代码分类方法。该方法简化 Dalvik 指令集，用指令符号抽象一类指令的操作码，并提出了一种基于 N-Gram 序列的特征模型。针对抽象的 Dalvik 指令符号的 N-Gram 序列特征，利用 AP 聚类算法进行样本压缩和利用信息增益方法进行特征筛选，并结合分类算法进行模型训练，有效地实现了 Android 恶意代码的检测模型。最后与常见反病毒软件进行实验相比较，实验表明该方法有较高的检测率。

（3）深入研究了 DEX 文件结构与 bytecode 生成图纹理之间的对应关系，提出一种可快速生成 bytecode 生成图的方法。

（4）提出了一个基于 bytecode 生成图的恶意家族分类方法，该方法通过结合 GIST 算法与 Random Forest 算法，实现对 bytecode 生成图的纹理特征进行了分类实验，进而实现了 Android 平台的恶意代码家族分类。最后，与 Drebin 方法进行实验相比较，实验表明该方法有较高的检测率。

（5）基于上述工作，设计并实现了 Android 恶意代码智能检测平台，该平台可以满足在线 APK 分析服务的应用场景。智能检测平台分为 Web 交互服务模块和恶意代码检测模块。恶意代码智能检测平台设计并实现完成后，进行了一定的功能测试，鉴定检测平台的正确性、健壮性、安全性。

6.2 展望

虽然本文做了不少研究工作与贡献，但是本文提到的几个方法还有许多值得改进和尝试的地方：

（1）本文实验选用的恶意代码样本存在时间已经比较久了，如果能够选用更多更新的恶意代码样本进行实验，获得的实验效果将更具有代表性和说服力。

（2）除了将 APK 文件反汇编为 Smali 代码，也可以尝试将 APK 文件反汇编成其他形式的代码。例如，使用 Soot 工具可以将 APK 文件反汇编成 jimple 代码[53]，再提取 jimple 代码的指令进行类似 N-Gram 编码方式进行特征提取。

（3）除了本文提及的静态特征提取方法，还可以使用其他提取特征的方法，加入不同的静态特征、动态特征进行实验比较，甚至采用多类特征混合的方法形成一个新的模型。

（4）深度学习是现在机器学习研究中的一个新领域，可以代替本文采用的 Random Forest 算法、支持向量机等分类算法进行研究，存在更好分类效果的可能性。

（5）针对 Android 平台的特性，研究其他类型的恶意代码可视化方法和图像纹理特征的提取方法，并比较不同的可视化方法与纹理特征提取方法，寻找一个更好分类效果的模型。

（6）进一步改进恶意代码智能检测平台。除了检测恶意代码的功能以外，还可以增加对恶意代码种类进行判断的功能。采用的训练样本可以进一步扩大，并对检测模型进行定期更新，使模型一直保持最佳的状态。

参考文献

[1] Gartner. Gartner Says Worldwide Smartphone Sales Grew 9.7 Percent in Fourth Quarter of 2015[EB/OL]. http://www.gartner.com/newsroom/id/3215217, 2016-02-18/2016-04-01..

[2] AppBrain. Android Statistics: Google Play stats[EB/OL]. http://www.appbrain.com/stats, 2016-02-01/2016-03-01.

[3] 张玉清, 方喆君, 王凯,等. Android安全漏洞挖掘技术综述[J]. 计算机研究与发展, 2015, 52(10):2167-2177.

[4] 张玉清, 王凯, 杨欢,等. Android安全综述[J]. 计算机研究与发展, 2014, 51(7):1385-1396.

[5] 阿里巴巴. 2015年安全报告[EB/OL]. 2016[2016-03-01].http://jaq.alibaba.com.

[6] Google Code. Androguard[EB/OL].[2016-03-01].http://code.google.com/p/androguard

[7] 卿斯汉. Android安全研究进展[J]. 软件学报, 2016(01):45-71.

[8] Fossi M, Egan G, Haley K, et al. Symantec internet security threat report trends for 2010[J]. Volume, 2011, 16: 20.

[9] ContiG, DeanE, SindaM, et al. Visual reverse engineering of binary and data files[J].Journal of General Physiology, 2008, 115(5):637.

[10] Anderson B, Storlie C, Lane T. Improving malware classification: bridging the static/dynamic gap[C]. Pro-ceedings of the 5th ACM workshop on Security and ar-tificial intelligence. ACM, 2012: 3-14.

[11] Nataraj L, Yegneswaran V, Porras P, et al. A comparative assessment of malware classification using binary texture analysis and dynamic analysis[C]. Proceedings of the 4th ACM workshop on Security and artificial intelligence. ACM, 2011: 21-30.

[12] Miettinen M, Halonen P, et al. Host-based intrusion detection for advanced mobile devices[A]. Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on[C]. Washington D.C.: IEEE, 2006, 72-76.

[13] Shabtai A, Kanonov U, et al. Andromaly: a behavioral malware detection framework for android devices[J]. Journal of Intelligent Information Systems. 2012, 38(1): 161-190.

[14] Blasing T, Batyuk L, et al. An android application sandbox system for suspicious software detection[A]. Malicious and unwanted software (MALWARE), 2010 5th international conference on[C]. Washington

D.C.: IEEE, 2010, 55-62.

[15] Chen Y, Ghorbanzadeh M, et al. A hidden Markov model detection of malicious Android applications at runtime[A]. Wireless and Optical Communication Conference (WOCC), 2014 23rd[C]. Washington D.C.: IEEE, 2014, 1-6.

[16] Shabtai A, Moskovitch R, Feher C, et al. Detecting unknown malicious code by applying classification techniques on opcode patterns[J]. Security Informatics, 2012, 1(1): 1.

[17] 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(9):64-82

[18] Dhaya R, Poongodi M. Source Code Analysis for Software Vulnerabilities in Android based Mobile Devices[J]. International Journal of Computer Applications, 2014, 93(17):10-14

[19] Arp D, Gascon H, Rieck K, et al. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket[C]// Network and Distributed System Security Symposium. 2014.

[20] Dong Hang, He Nengqiang, Hu ge, et al. Malware detection method of android application based on simplification instructions[J]. The Journal of China Universities of Posts and Telecommunications, 2014, 21: 94-100.

[21] Nataraj L, Karthikeyan S, Jacob G, et al. Malware images: visualization and automatic classification[C]. Proceedings of the 8th international symposium on visualization for cyber security. ACM, 2011: 4.

[22] Wu Y, Yap R. Experiments with malware visualization[M]. Detection of Intrusions and Malware, and Vulnerability Assessment. Springer Berlin Heidelberg, 2012: 123-133.

[23] Han K, Lim J, Im E. Malware analysis method using visualization of binary files[C]. Proceedings of the 2013 Research in Adaptive and Convergent Systems. ACM, 2013: 317-321.

[24] Shaid M, Zainudeen S, Maarof M. Malware behavior image for malware variant identification[C]. Biometrics and Security Technologies (ISBAST), 2014 International Symposium on. IEEE, 2014: 238-243.

[25] 韩晓光, 曲武, 姚宣霞, 等. 基于纹理指纹的恶意代码变种检测方法研究[J]. 通信学报, 2014, 35(8): 125-136.

[26] Zhou Y, Jiang X. Dissecting android malware: Characterization and evolution[C]. Security and Privacy (SP), 2012 IEEE Symposium on. IEEE, 2012: 95-109.

[27] Arp D, Spreitzenbarth M, Hubner M, et al. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket[C]. NDSS. 2014.

[28] 金建国. 聚类方法综述[J]. 计算机科学, 2014, 41(S2):288-293.

[29] Frey B J, Dueck D. Clustering by passing messages between data points[J]. science, 2007, 315(5814): 972-976.

[30] Ho T K. Random decision forests[A]. Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on[C]. IEEE, 1995, 278-282.

[31] Ho T K. The random subspace method for constructing decision forests[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 1998, 20(8): 832-844.

[32] Breiman L. Random forests[J]. Machine learning. 2001, 45(1): 5-32.

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

[34] Zhou Z, Tang W. Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing[M], Berlin:Springer, 2003.

[35] Robnik M. Machine Learning: ECML 2004[M], Berlin:Springer, 2004.

[36] Kotsiantis S, Kanellopoulos D. Combining bagging: boosting and random subspace ensembles for regression problems[J]. International Journal Of Innovative Computing Information And Control. 2012, 8(6): 3953-3961.

[37] pattern analysis and machine intelligence 1983 IEEE

[38] Oliva A, Torralba A. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope[J]. International Journal of Computer Vision, 2001, 42(3):145-175.

[39] GitHub. Apktool: A tool for reverse engineering Android apk files[EB/OL].[2016-03-01].https://github.com/iBotPeaches/Apktool/

[40] Gabor Paller. Dalvik opcodes[EB/OL].[2016-03-01].http://pallergabor.uw.hu/androidblog/dalvik_opcodes.html

[41] HAN J.Data Mining[M]. San Francisco: Morgan Kaufmann, 2011.

[42] DREBIN. The Drebin Dataset[EB/OL].[2016-03-01].https://www.sec.cs.tu-bs.de/~danarp/drebin/index.html

[43] GooglePlay [EB/OL].[2016-03-01].https://play.google.com

[44] Grace M, Zhou Y, Zhang Q, et al. RiskRanker: scalable and accurate zero-day android malware detection[C]// Proceedings of the 10th international conference on Mobile systems, applications, and services. ACM, 2012:281-294

[45] Chen T, Zhang X, Jin S, et al. Efficient classification using parallel and scalable compressed model and its application on intrusion detection[J]. Expert Systems with Applications, 2014, 41(13):5972–5983.

[46] ThreatBook [EB/OL].[2016-03-01].https://x.threatbook.cn/

[47] Elenkov N. Android Security Internals: An In-Depth Guide to Android's Security Architecture[M]. No Starch Press, 2014.

[48] Oliva A, Torralba A. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope[J]. International Journal of Computer Vision, 2001, 42(3):145-175.

[49] t-SNE

[50] HAN J, KAMBER M, PEI J. Data Mini；ng[M]. San Francisco:Morgan Kaufmann, 2011.

[51] http://www.ard9.com/zxdt/2672260.html

[52] D3.js

[53] Dexpler: converting Android Dalvik bytecode to Jimple for static analysis with Soot

致谢

岁月如梭，几乎是一转眼之间，三年的硕士研究生的学习生涯即将到此画上句号。这三年的学习与生活经历对我来说将是人生中最重要、最宝贵的时光，收获是巨大而丰富的。在这学位论文即将完成、毕业即将来临之际，我要对我的老师们、同学们、亲人和朋友们表示衷心的感谢，感谢你们长久以来的关怀与帮助，使我能够不断成长与蜕变，成为更好的自己。

由衷感谢我的导师陈铁明教授，在攻读硕士研究生期间，循循善诱，对我的学术研究起到了很大的指导与帮助的作用，使我能够很顺利地完成自己的论文。同时，陈老师也引导我们学会了勤于思考，勤于实践，理论结合实践，学术结合技术，认真做事做学问，不断提高解决问题的能力，总之感觉受益匪浅。

同时，感谢同一实验组的江颉老师、陈波老师的指导与帮助，他们渊博的知识、出色的理论功底、高深的科研水平，令我们印象深刻，为之钦佩。在此衷心祝愿各位老师身体健康，阖家欢乐！

最后，衷心感谢几位室友、各位同学陪伴我研究生生涯三年，感到十分有幸一起学习、生活，在此过程中共同成长，愿各位身体健康，万事如意，心想事成！

攻读学位期间参加的科研项目和成果

参加的科研项目

[1] 国家自然科学基金：U1509214，

[2] 浙江省信息安全重点实验室开放课题：KF201603

录用和发表的论文

[1] 陈铁明, 杨益敏, 陈波. Maldetect:基于 Dalvik 指令抽象的 Android 恶意代码检测系统[J]. 计算机研究与发展, 2016, 53(10):2299-2306.

[2] 杨益敏, 陈铁明. 基于字节码图像的 Android 恶意代码家族分类方法[J]. 网络与信息安全学报, 2016, 2(6):38-43.