论文题目：基于集成学习的多特征Android恶意应用程序检测

工程领域：计算机技术

硕 士 生：\*\*\* （签名）

指导教师：\*\*\* （签名）

摘 要

随着大量程序样本的积累，以数据统计为基础的机器学习方法已经成为检测恶意应用的重要方法之一。现有的基于机器学习的检测方法通常以权限或API为特征表示应用行为，这种方法普遍存在特征提取维度过高、特征选取不够全面以及单一分类算法不能有效发挥多类特征在恶意应用检测上所起的不同作用等问题。针对现有机器学习检测方法存在的弊端，本文提出一种新的Android恶意应用检测方法。主要研究内容如下：

(1)针对恶意应用检测时对所有权限与API进行提取，造成特征维度过高，检测效率较低的问题，本文总结了恶意应用最常用的50个权限与39个API，并将其作为特征集。通过对大量恶意应用进行分析后发现，大部分API和权限不能够有效区分恶意应用和正常应用，对所有权限与API进行特征提取，会造成大量冗余数据。因此对本文所总结的特征集进行特征提取，可以减少特征提取维度，提高特征提取效率。

(2)针对单一使用API或权限特征无法充分体现恶意应用程序的特性，造成一定误报率这一问题，本文提出了一种基于多特征的Android恶意应用程序检测方法。在保留API和权限的基础上，增加了组件、图片、界面元素等资源文件特征，最后将这些特征保存到同一个特征向量集合中，并采用机器学习算法构建了一个多特征检测模型。实验结果表明，新增的资源特征容易提取，而且与单一特征检测方法相比，基于多特征的检测方法表现出了出更好的准确率。

(3)应用程序的不同特征在不同机器学习分类算法上的表现会有所差异，单一的机器学习分类算法不能有效的发挥应用程序不同特征在恶意应用检测上所起到的不同作用，针对这一问题，本文提出了一种基于集成学习的多特征Android恶意应用程序检测方法。将提取的权限、API以及资源特征分别用不同的分类算法进行训练并将其结果作为子模型进行集成学习，选取每类特征的最优算法，采用加权投票选择的方式对最优算法中的分类结果进行特征融合。实验结果表明，该集成方法有效弥补了传统检测方法的不足，从而能更加高效地检测Android恶意应用程序。

关 键 词：恶意代码检测；权限；API；资源特征；集成学习

研究类型：理论研究

|  |  |  |  |
| --- | --- | --- | --- |
| **Subject** | **:** | **Multi-feature Android Malicious Code Detection Based on Ensemble Learning** | |
| **Specialty** | **:** | **Computer Technology** | |
| **Name** | **:** | **\*\*\*** | **(Signature)** |
| **Instructor** | **:** | **\*\*\*** | **(Signature)** |

**ABSTRACT**

With the accumulation of a large number of program samples, machine learning methods based on data statistics have become one of the important methods to detect malicious applications. The existing machine learning-based detection methods usually use the authority or API to represent the application behavior. This method is generally characterized by too high feature extraction dimension, insufficient feature selection, and single classification algorithm can not effectively play multiple types of features in malicious application detection. Problems such as different roles. Aiming at the shortcomings of existing machine learning detection methods, this paper proposes a new Android malicious application detection method. The main research contents are as follows:

(1)For the detection of malicious applications, all permissions and APIs are extracted, resulting in high feature dimension and low detection efficiency. This paper summarizes the 50 most commonly used malicious applications and 39 APIs, and uses them as features. set. After analyzing a large number of malicious applications, it is found that most APIs and permissions cannot effectively distinguish between malicious applications and normal applications, and feature extraction of all permissions and APIs will result in a large amount of redundant data. Therefore, feature extraction of the feature set summarized in this paper can reduce the feature extraction dimension and improve feature extraction efficiency.

(2)For the single use API or permission feature can not fully reflect the characteristics of malicious applications, resulting in a certain false positive rate, this paper proposes a multi-feature based Android malicious application detection method. On the basis of preserving the API and permissions, the resource file features such as components, pictures and interface elements are added. Finally, these features are saved in the same feature vector set, and a multi-feature detection model is constructed by machine learning algorithm. The experimental results show that the new resource features are easy to extract, and the multi-feature based detection method shows better accuracy than the single feature detection method.

(3)The different characteristics of the application will be different in different machine learning classification algorithms. A single machine learning classification algorithm can not effectively play the different roles of different features of the application in malicious application detection. A problem, this paper proposes a multi-feature Android malicious application detection method based on integrated learning. The extracted permissions, API and resource features are trained by different classification algorithms and the results are integrated learning as sub-models. The optimal algorithm for each type of feature is selected, and the classification in the optimal algorithm is adopted by weighted voting selection. The result is feature fusion. The experimental results show that the integrated method effectively compensates for the shortcomings of traditional detection methods, so that Android malicious applications can be detected more efficiently.

**Key words:** Malicious code detection; Permissions; API; Resource characteristics; Integrated learning

**Thesis :** Theoretical Research

目 录

[1 绪 论](#_Toc2921_WPSOffice_Level1) [1](#_Toc2921_WPSOffice_Level1)

[1.1 选题背景及意义](#_Toc8603_WPSOffice_Level2) [1](#_Toc8603_WPSOffice_Level2)

[1.2 国内外研究现状](#_Toc4754_WPSOffice_Level2) [3](#_Toc4754_WPSOffice_Level2)

[1.2.1基于特征代码的检测技术](#_Toc4754_WPSOffice_Level3) [3](#_Toc4754_WPSOffice_Level3)

[1.2.2 基于行为分析的检测技术](#_Toc5583_WPSOffice_Level3) [3](#_Toc5583_WPSOffice_Level3)

[1.2.3 恶意代码的挖掘技术](#_Toc8544_WPSOffice_Level3) [7](#_Toc8544_WPSOffice_Level3)

[1.2.4 基于云的检测技术](#_Toc14272_WPSOffice_Level3) [8](#_Toc14272_WPSOffice_Level3)

[1.3 研究内容](#_Toc5583_WPSOffice_Level2) [8](#_Toc5583_WPSOffice_Level2)

[1.4 论文结构安排](#_Toc8544_WPSOffice_Level2) [9](#_Toc8544_WPSOffice_Level2)

[2 Android相关知识概述](#_Toc8603_WPSOffice_Level1) [10](#_Toc8603_WPSOffice_Level1)

[2.1 Android系统介绍](#_Toc14272_WPSOffice_Level2) [10](#_Toc14272_WPSOffice_Level2)

[2.1.1 Android系统框架](#_Toc7830_WPSOffice_Level3) [10](#_Toc7830_WPSOffice_Level3)

[2.1.2 Android软件结构-APK](#_Toc11429_WPSOffice_Level3) [11](#_Toc11429_WPSOffice_Level3)

[2.1.3 Android系统应用程序组件](#_Toc14712_WPSOffice_Level3) [12](#_Toc14712_WPSOffice_Level3)

[2.2 Android安全机制](#_Toc7830_WPSOffice_Level2) [13](#_Toc7830_WPSOffice_Level2)

[2.2.1 进程沙箱隔离机制](#_Toc12781_WPSOffice_Level3) [13](#_Toc12781_WPSOffice_Level3)

[2.2.2 应用程序签名机制](#_Toc24561_WPSOffice_Level3) [14](#_Toc24561_WPSOffice_Level3)

[2.3.3 权限声明机制](#_Toc29488_WPSOffice_Level3) [14](#_Toc29488_WPSOffice_Level3)

[2.3.4 进程通信机制](#_Toc30533_WPSOffice_Level3) [15](#_Toc30533_WPSOffice_Level3)

[2.3 Android平台下恶意应用常用技术分析](#_Toc11429_WPSOffice_Level2) [15](#_Toc11429_WPSOffice_Level2)

[2.3.1 恶意代码载荷技术](#_Toc12411_WPSOffice_Level3) [15](#_Toc12411_WPSOffice_Level3)

[2.3.2 Android平台恶意软件事件监听技术](#_Toc30630_WPSOffice_Level3) [16](#_Toc30630_WPSOffice_Level3)

[2.3.3 恶意代码数据发送技术](#_Toc16435_WPSOffice_Level3) [16](#_Toc16435_WPSOffice_Level3)

[2.4 Android平台恶意软件分类](#_Toc14712_WPSOffice_Level2) [17](#_Toc14712_WPSOffice_Level2)

[2.5 本章小结](#_Toc12781_WPSOffice_Level2) [18](#_Toc12781_WPSOffice_Level2)

[3 Android软件静态特征提取 1](#_Toc4754_WPSOffice_Level1)9

[3.1 恶意代码常用权限的分析与提取 1](#_Toc24561_WPSOffice_Level2)9

[3.1.1 权限特征的分析 1](#_Toc29771_WPSOffice_Level3)9

[3.1.2 权限特征的提取](#_Toc28027_WPSOffice_Level3) [20](#_Toc28027_WPSOffice_Level3)

[3.2 恶意代码常用API分析与提取](#_Toc29488_WPSOffice_Level2) [22](#_Toc29488_WPSOffice_Level2)

[3.2.1 系统API的行为分析](#_Toc20226_WPSOffice_Level3) [23](#_Toc20226_WPSOffice_Level3)

[3.2.2 系统API特征的提取](#_Toc15225_WPSOffice_Level3) [25](#_Toc15225_WPSOffice_Level3)

[3.3资源特征的分析与提取](#_Toc30533_WPSOffice_Level2) [25](#_Toc30533_WPSOffice_Level2)

[3.3.1 图片特征 26](#_Toc20226_WPSOffice_Level3)

[3.3.2 组件特征 27](#_Toc15225_WPSOffice_Level3)

[3.3.3 界面元素特征 28](#_Toc15225_WPSOffice_Level3)

[3.4本章小结 2](#_Toc12411_WPSOffice_Level2)9

[4 基于集成学习的多特征Android恶意应用程序检测模型](#_Toc5583_WPSOffice_Level1) [30](#_Toc5583_WPSOffice_Level1)

[4.1 基于权限特征的检测模块](#_Toc30630_WPSOffice_Level2) [30](#_Toc30630_WPSOffice_Level2)

[4.2基于系统API特征的检测模块](#_Toc16435_WPSOffice_Level2) [32](#_Toc16435_WPSOffice_Level2)

[4.3基于资源特征的检测模块](#_Toc26894_WPSOffice_Level2) [33](#_Toc26894_WPSOffice_Level2)

[4.4基于多特征的检测模块](#_Toc4328_WPSOffice_Level2) [34](#_Toc4328_WPSOffice_Level2)

[4.5基于集成学习的多特征Android恶意应用程序检测模型](#_Toc29771_WPSOffice_Level2) [34](#_Toc29771_WPSOffice_Level2)

[4.5.1机器学习分类算法](#_Toc14691_WPSOffice_Level3) [34](#_Toc14691_WPSOffice_Level3)

[4.5.2集成学习](#_Toc17208_WPSOffice_Level3) [37](#_Toc17208_WPSOffice_Level3)

[4.5.3 结合MLP神经网络的Stacking集成分类技术](#_Toc5815_WPSOffice_Level3) [38](#_Toc5815_WPSOffice_Level3)

[4.5.4 基于支持向量机的Bagging集成分类技术](#_Toc30066_WPSOffice_Level3) [38](#_Toc30066_WPSOffice_Level3)

[4.5.5改进Bagging集成分类技术 3](#_Toc2315_WPSOffice_Level3)9

[4.6 本章小结](#_Toc12994_WPSOffice_Level2) [4](#_Toc12994_WPSOffice_Level2)0

[5 实验设计与结果分析](#_Toc8544_WPSOffice_Level1) [41](#_Toc8544_WPSOffice_Level1)

[5.1 测试环境](#_Toc4394_WPSOffice_Level2) [41](#_Toc4394_WPSOffice_Level2)

[5.2 实验样本介绍](#_Toc28027_WPSOffice_Level2) [41](#_Toc28027_WPSOffice_Level2)

[5.3 实验设计](#_Toc25368_WPSOffice_Level2) [42](#_Toc25368_WPSOffice_Level2)

[5.3.1实验数据集的划分](#_Toc20891_WPSOffice_Level3) [43](#_Toc20891_WPSOffice_Level3)

[5.3.2恶意代码检测分类评价方法](#_Toc9538_WPSOffice_Level3) [43](#_Toc9538_WPSOffice_Level3)

[5.4 实验结果与分析](#_Toc20226_WPSOffice_Level2) [43](#_Toc20226_WPSOffice_Level2)

[5.4.1基于单一特征的检测模块](#_Toc15206_WPSOffice_Level3) [44](#_Toc15206_WPSOffice_Level3)

[5.4.2 基于多特征的检测模型的分类实验](#_Toc24960_WPSOffice_Level3) [47](#_Toc24960_WPSOffice_Level3)

[5.4.3本课题的检测模型与以往方法的对比实验](#_Toc6770_WPSOffice_Level3) [48](#_Toc6770_WPSOffice_Level3)

[5.5 本章小结](#_Toc8120_WPSOffice_Level2) [49](#_Toc8120_WPSOffice_Level2)

[6 总结与展望](#_Toc14272_WPSOffice_Level1) [50](#_Toc14272_WPSOffice_Level1)

[6.1 本文工作总结](#_Toc16427_WPSOffice_Level2) [50](#_Toc16427_WPSOffice_Level2)

[6.2 未来工作展望](#_Toc30526_WPSOffice_Level2) [51](#_Toc30526_WPSOffice_Level2)

[致 谢](#_Toc7830_WPSOffice_Level1) [52](#_Toc7830_WPSOffice_Level1)

[参考文献](#_Toc11429_WPSOffice_Level1) [53](#_Toc11429_WPSOffice_Level1)

[附 录](#_Toc14712_WPSOffice_Level1) [57](#_Toc14712_WPSOffice_Level1)

1 绪 论

## 1.1 选题背景及意义

相较于其他操作系统而言，Android系统的免费性、开发的易学性、开放性、丰富的谷歌应用等特点，使得Android系统迅速的积累了大量人气，被广大消费者与开发者所接受，从而极大地促进了Android系统的发展。2019-2025年中国智能手机操作系统行业市场需求预测及投资未来发展趋势报告，Android系统在智能手机设备领域的市面占据比例高达80%[1]。虽然Android系统具有诸多优势，但是由于其开放性和自身安全机制存在的一些漏洞，会给一些恶意应用可乘之机，从而出现越来越多的信息安全问题，给用户的信息和财产安全造成极大的威胁。

腾讯安全反病毒实验室发布《2018年Q3季度互联网安全报告》数据显示[2]，PC端前三季度共计拦截的恶意应用数量超过了3亿多次，平均每天拦截的恶意应用超过了120万次。随着大量Android软件丰富着公众的学习、生活和工作，越来越多的Android安全问题也都暴露出来，一些恶意应用程序在用户毫无察觉的情况下，读取手机短信、联系人信息等隐私数据，甚至将用户的这些隐私数据发送出去；一些木马甚至会自动安装一些恶意扣费软件来获取用户的银行账户以及一些支付信息，极大的影响了个人的财产安全。由于恶意代码技术的不断发展，使得恶意应用程序的种类与数量也有了较大的增长，目前常见的恶意应用程序分为以下几大类[3]，如表1.1所示。

表1.1 恶意软件分类说明

|  |  |
| --- | --- |
| 恶意软件名称 | 说明 |
| 恶意扣费 | 恶意代码通过各种手段诱导用户产生付费行为，使用户的财产安全受到损失。 |
| 窃取隐私 | 恶意代码获取用户的隐私数据、个人信息，并将其发送出去。 |
| 远程控制 | 没有认证或授权的应用程序，接收和执行远程命令，被其所控制。 |
| 恶意传播 | 恶意应用程序通过各种技术手段（下载、复制、感染等）对恶意代码进行传播。 |
| 费用消耗 | 恶意应用通过自动拨打电话，发送短信等方式对用户造成额外费用消耗。 |
| 系统损坏 | 恶意代码可能通过各种方式损坏系统功能，如移动终端，网络服务等。 |
| 恶意欺骗 | 恶意软件通过欺骗的方式，使用户伪造或篡改正常的应用程序。 |
| 流氓行为 | 恶意软件驻留在内存中，产生资源消耗，自动安装，弹出广告等流氓行为。 |

以下是360互联网安全中心的数据统计，从图1.1中的(a)可以看出2012年至2015年新增的恶意应用数量进入一个爆发期，仅2015年新增的恶意应用的数量就达到了0.15亿次，从2015年开始，新增的应用应用数量逐渐减少，到了2018年新增的恶意应用的数量约434.2万个；从图1.1中的(b)可以看出2018年全年，移动端恶意应用感染量约为1.1亿次，相比2017年感染量下降48.6%，相较于2014最高的3.8亿次，感染量减少了约2.7亿次，从整体上看2014年至2018年，恶意应用的感染量在逐年下降。

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

图1.1 2012-2018年移动端恶意程序新增样本数与移动端恶意程序感染量

图1.2中的(a)是2018年移动端恶意应用新增量与感染量的按季度对比情况，从恶意应用的新增量来看，第二季度最高，约141.7万，但与第一季度相差不大，第四季度却只有52.4万次，一整年来看，整体上是逐渐下降的；从恶意应用的感染量来看，最高的为第一季度，约为3437万次，最少的为第四季度，约为2057万次，整体上从第一季度到第四季度也是不断下降的。从图1.2中的(b)中可以看出新增恶意应用类型主要分为为资费消耗、隐私窃取、恶意扣费、流氓行为、远程控制等，其中资费消耗占比最高，达到了63.2%。

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

图1.2 2018年移动端恶意程序新增量与感染量和新增恶意程序类型分布

一方面，由于Android系统的开放与免费性，使得Android系统被广泛使用在我们的日常生活中，Android系统在给我们的生活带来巨大生活便利的同时，相应的也带来了许多的移动安全问题。另一方面，由于Android恶意代码混淆，脱壳，重打包技术的不断发展[4]，出现越来越多新的安全问题，如恶意载荷、应用监听以及注入攻击等。因此，开发快速、高效的Android恶意应用检测工具势在必行，Android应用软件逆向分析与检测技术的发展，能够极大的提高Android系统中恶意应用程序的检测效率，对于移动安全的贡献意义非凡。

## 1.2 国内外研究现状

1.2.1基于特征代码的检测技术

在恶意应用检测中，不同的文献对于特征代码检测的定义会有所差异。广义上，学者们会将对字节码特征检测认定为对应用程序的特征代码检测出；但是也有部分学者会将对特征库的检测认定为对应用程序的特征代码的检测，然而基于特征库的检测方法的特征库需要及时更新，且需要特别大的内存空间，因此我们一般采广义上的定义，对应用程序特征代码的检测是通过判断应用中是否包含字节码特征(如一段代码、应用签名或者字符串)来检测未知应用[5]。如果所提取的特征出现在所检测的未知应用中，则将该应用程序检测为恶意应用程序。

基于特征代码的检测技术是对一些已知的特征(签名、字符串)进行提取，所以该方法具有简单、直接、高效的优点，但它却存在很大的弊端，例如使用签名来作为特征代码，需要特别大的内存来存储签名，而且该方法对签名库以外的软件没有作用。除此之外，如果恶意软件经过一些恶意技术的伪装后，该方法对其不具备任何抵抗能力。目前市面上比较流行检测软件都是都是采用基于签名的方法对软件安装进行检测，如网秦手机卫士、奇虎360等产品。

1.2.2 基于行为分析的检测技术

目前，关于Android应用程序的行为分析的检测技术主要分为静态检测方法[6]和动态检测方法[7]两条技术路线。静态检测主要是检查应用程序的代码和文档，而不执行被检测的应用程序，以此来判断应用程序是否安全；反之如果在检测过程中通过执行应用程序，根据系统资源的使用情况来判别应用程序的恶意性，这种检测方法称为动态检测。下面就这两种方法做一个详细介绍。

（1）静态检测技术

静态检测技术就是指不安装运行运行应用程序，而是直接对应用程序反编译后的各类文件进行检测分析，从而确定应用程序的结构特性，也可以确定恶意代码的一些静态行为特征，甚至还可以分析出恶意代码的敏感API的控制流图。目前静态检测技术种类较多，如基于校验、基于正则表达式比对的检验技术等，但是使用最多的还是下面两种检测技术：

①代码语义分析

代码语义分析[8]，就是将APK文件通过反编译工具进行反编译，得到能够容易理解，易于识别和提取所需特征信息代码的文件，然后分析其行为逻辑，以及实现的功能，从而构建出APK的系统调用序、控制流图等，并通过获得的特征判断应用代码是否为恶意代码。通过分析多态变形技术原理发现，多态变形技术虽然改变了代码语法特征，但是其语义特征却没有改变。基于语义的检测方法正是利用该技术的这一弱点，为更好地对抗加壳技术、多态变形技术提供了可能。现阶段，针对代码语义的分析技术[9-11]已经大量应用于恶意应用程序的检测中来，并且在一些方面已经取得的很大的进步，其中比较有代表性的是Meng等人[10]提出了一种用于恶意软件理解，检测和分类的模型(DSA)。并且开发了一个名为SMART的自动分析框架，它通过检测和总结来自恶意软件系列的语义克隆来学习DSA，然后从学习的DSA中提取语义特征，以根据攻击模式对恶意软件进行分类。

②代码特征分析

基于代码特征的分析技术通过分析应用程序源码，建立程序代码特征与程序性质之间的关系，从而进行恶意行为检测。随着反编译技术的日臻成熟，从打包文件获取程序源码成为可能。学者们对APK进行反编译，并从反编译结果中获取APK的代码特征，如隐私数据流向等[12]，并结合控制流分析、模糊哈希技术[13]、数据流分析[14]等技术，对应用程序内部结构进行详细分析，从而实现对应用程序恶意行为的检测。此外，由于重打包技术[15]是对一个APK文件进行反编译，做出某种修改，再将其重新打包成APK文件的过程。该过程往往是攻击者通过向正常应用中加入恶意负载，从而实现其恶意行为的手段。因此，可以利用启发式扫描[16]技术对重打包样本进行检测。

Arp D等人[17]提出了一种轻量级方法DREBIN，该方法通过各种静态分析手段，将应用程序的特征提取出来，并将提取的特征以统一的格式存入特征向量空间中，使用支持向量机分类算法进行应用程序恶意性判断。Zhuo Ma等人[18]将控制流图与机器学习相结合对恶意应用进行检测。首先，获取应用程序的API信息并且构建控制流图，基于API信息，Zhuo Ma等人创新地构建布尔，频率和时间序列数据集，并且基于这三个数据集，构建了关于API调用，API频率和API序列方面的Android恶意软件检测的三种子模型，最终，将三种子模型构建成一个集合模型。Enck[19]等人最先开始对应用程序的权限特征进行研究，并且制定了Kirin规则，该规则是基于权限安全的，如果应用程序在使用权限时，违反了该规则，则系统会拒绝安装此应用程序。但是由于该规则只考虑了权限信息，仅仅拿权限特征来判别应用程序是否具有恶意性会具有较高的误报率。Kang H等人[20]首先使用创建者信息对恶意软件进行快速检测，除此此外，他们还对应用程序的权限，API调用以及恶意命令等特征进行分析，以此来提高检测准确性。Wu DJ等人[21]提出了一种基于静态特征的检测机制—，该机制考虑APK静态信息，包括敏感权限，通信组件以及可以代表应用程序行为的敏感信息，并对提取的APK静态特征通过算法进行聚类，最后使用分类算法对其进行分类。Do Y等人[22]提出了一种使用加权函数调用图的社区结构的Android恶意软件检测方法，该方法自动将函数调用图划分为社区结构。然后，可以使用这些社区结构的功能来检测恶意软件。通过改进Girvan-Newman算法并使用机器学习分类器而不是子图的相似性比较来减少计算时间。杨欢等人[23]在系统的考虑了应用程序的不同的静态特征的基础上，提出了一种算法—THEA，该算法将提取到的三种不同的应用程序的静态特征，分别训练出三种分类模型，并选取每类特征的最优模型形成一个集成的分类模型进行应用程序的恶意性判断。Wu S等人[24]利用机器学习算法对恶意应用进行检测，提取应用程序的API，并将其作为特征来检测Android恶意软件，机器学习算法采用改进k-最近邻分类模型，最后将提取的API特征利用机器学习算法进一步优化，在准确率几乎不变的情况下，特征传输效率能够提高40%左右。

（2）动态检测技术

动态检测与静态检测技术不同，动态检测技术是指检测者在提前设计好的环境中安装运行应用程序，在这个环境中可以对应用程序产生的一切行为进行监控，例如应用程序消耗的内存情况、权限的申请状况与敏感的调用信息等，然后分析该应用是否会造成用户信息泄露，是否会攻击系统，是否产生流氓行为。动态监测有检测精度高，误报率低等特点，但它的特点也同样明显，动态监测会消耗大量系统资源，实时性不强。动态检测技术通常有以下三种分析方法。

①日志信息分析

日志信息分析是指通过对应用程序进行反编译，然后对反编译的代码进行修改，对各种敏感API的底层代码加入Log输出信息，重打包后，在输出的日志信息中查看应用程序是否会造成用户信息泄露，是否会攻击系统，是否产生流氓行为，从而判别应用程序是否有具有恶意性。日志信息分析需要对反编译后的API进行操作，因此就必须要考虑逆向对抗技术对反编译过程的影响，除此之外还要考虑反汇编技术、签名验证等技术的影响，从而使得检测成本过高，检测过程变得极其复杂。基于人工进行日志信息分析的诸多不便，Sony等人设计了一款自动化日志信息分析工具—ChkBugReport[25]通过处理Log输出的参数信息，尽可能多地提取有用的数据，并将提取的数据保存在本地，生成HTML格式的文件。

②代码行为分析

代码行为分析[26]方法指在是指检测者在提前设计好的环境中安装运行应用程序，在这个环境中可以对应用程序产生的一切行为进行监控，例如应用程序消耗的内存情况、权限的申请状况与敏感的调用信息等，从而识别该应用程序是不是恶意应用程序。应用程序的行为获取可通过以下三种方式：系统调用把用户从底层的硬件编程中解放出来，使应用程序具有可移植性，系统调用通过软中断向内核发出一个明确的请求，极大的提高了系统的安全性；恶意应用要实现某个功能就调用安卓系统自己提供的功能，如果不想调用就必须通过继承实现个性化扩展，因此通过监听应用程序框架层的API调用，也可以检测应用的恶意行为；虚拟机自省是指利用自身创建的虚拟机对其他虚拟机进行检查与监督的行为，这个行为是在一个单独的内存中进行的，因此非常安全，利用虚拟机自省方式可以获取应用程序的行为信息。

Iker Burguera等人[27]设计了一个恶意应用检测系统—，将探测器嵌入在整体框架中，并监视软件对Linux内核系统调用的使用情况，用于收集来自Android软件的真实用户的痕迹，并使用划分聚类算法来判断该软件是否是恶意软件。该方法已经通过使用来自用于测试目的的人工恶意软件和来自未知的真实恶意软件的数据集证明隔离恶意软件并警告用户下载的恶意软件的有效手段，这表明可以避免将检测到的恶意软件传播到更大的社区。

Vidal JM等人[28]提出了一种新颖的模式识别系统，用于通过分析可疑的启动序列来检测Android恶意软件。该方法仅研究在最近安装的应用程序的引导过程期间执行的系统调用，减少了要考虑的信息量。该方法定义了一个模式识别系统，该系统具有三个处理层：监控、分析和决策。首先，为了提取系统调用序列，可能受损的应用程序在安全且隔离的环境中执行。然后，对决策生成的指标进行分析。该级别将序列比对算法与装袋相结合，其允许考虑其最相似的区域对提取的序列之间的相似性进行评分。在决策阶段，实施Wilcoxon签名等级测试，该测试确定新软件是否被标记为合法或恶意。

于洲等人[29]实现了对系统恶意应用的自动检测。通过恶意软件自动执行与行为触发方法的技术，可以对前台用户交互事件与处理器响应事件自动执行并处理，更加全面的检测到恶意应用产生的恶意行为；同时通过动态软件胎记相似性的检测方法，可以对API调用信息进行拦截，提取出应用程序胎记，这一特征对同一家族的恶意代码的检测非常有效；该自动检测系统，可以尽可能地避免因脱壳、多态变形、反调试等技术对静态检测带来的困扰，使恶意软件的检测变得更加高效。

③数据分析

数据分析是指对应用程序的数据信息做好标记，在执行应用程序的过程中，对这些标记数据进行监控，观察应用程序是否出现数据泄露行。是一种高效的系统范围的动态污点跟踪和分析系统[30]，它解决了Android操作系统无法为用户提供对私有数据的充分与可见性，能够同时跟踪多个敏感数据源。在上产生的系统开销很低，只有不到15%，而且其在应用程序上的开销几乎为0。对应用程序常用的30个权限使用进行监控，发现其中有68个应用程序会过度申请用户隐私权限。Mingshen Sun等人[31]为新的Android系统设计了一个名为的多级信息流跟踪系统。采用多级污点分析技术来最小化污点标签存储。因此，污点标签可以存储在处理器寄存器中，以提供有效的污点传播操作。Mingshen Sun等人还定制了ART编译器，以最大限度地提高编译优化的性能，此外还设计了多级隐私实施，以防止敏感数据泄露。与Android 4.4中的传统环境相比，使运行时基准测试的性能提高了约99.7％。

1.2.3 恶意代码的挖掘技术

恶意代码的挖掘技术指的是对Android应用程序进行反编译，然后在反编译的结果中提取出一些能够代表应用程序特点的特征(如系统API调用信息，权限信息等)，然后通过机器学习算法，将提取的特征集合进行训练并形成一个检测模型。

2001年，Schultz提出了一种数据挖掘框架[32]，该框架第一次将数据挖掘与机器学习的方法应用于恶意代码检测当中，该框架可以准确，自动地检测恶意代码，并且对之前从未出现的恶意代码也会有不错的检测结果。数据挖掘框架与传统的恶意应用程序的检测方法不同，它会对数据集进行自动的训练，并形成一个检测模型模型，然后使用此模型对新的恶意应用程序进行检测。与当时传统的基于签名的方法进行比较，数据挖掘框架的检测率提高了一倍不止。此后，越来越多的研究人员开始往恶意代码的挖掘技术这一领域进行研究。

由于大多数现有的恶意Android应用程序检测方法依赖于手动选择的检测启发式模型。Yang Chao等人描述了一个名为DroidMiner的新的互补系统[33]，为了自动分析恶意应用的程序逻辑，将挖掘的恶意程序逻辑训练成检测模型，然后在其他模型中寻找这些检测模型。在自动学习这些恶意行为模型之后，DroidMiner可以扫描新的Android应用程序，以确定它是否包含恶意模式、诊断与其关系最密切的恶意软件系列以及提供有关为什么该应用程序被视为恶意的证据。Yang Chao等人使用2,466个恶意应用程序来评估DroidMiner，其检测率达到了95.3％，误报率仅为0.4％。

1.2.4 基于云的检测技术

云计算是一种众所周知的体系结构，它通过Internet以按使用付费模式远程提供计算和存储服务，因此可以极大的减少开发与维护成本，从而更加充分的使用资源。由于云计算是一个共享设施并且可以远程访问，因此它容易受到各种攻击，包括需要立即关注的主机和基于网络的攻击，恶意代码部署是当今云模式中威胁的主要原因，防病毒软件无法检测到许多现代恶意软件威胁，这些威胁会对基本云操作造成严重影响。

Shaw S等人提出了一种新的云架构恶意软件检测模型[34]。该模型通过合并多个检测引擎，可以识别恶意软件和不需要的软件。文中遵循DNA序列检测过程，符号检测过程和行为检测过程来检测各种威胁。提出的方法(PMDM)可以部署在VMM上，VMM对于VM和云用户保持完全透明。但是，PMDM可以防止在一个VM(受感染的VM)中运行的恶意代码在VMM的帮助下传播到另一个未受感染的VM。在通过PMDM技术检测到恶意代码后，它会向其他VM发出警告。在本文中，PMDM的原型部分在一个流行的开源云架构-Eucalyptus上实现。

Gupta S等人提出了一种基于即时系统调用序列的云环境中恶意程序执行检测方法来确定云中的恶意程序执行[35]。所提出的技术在涉及的复杂性和所使用的资源方面是有效的，以便证明其可行的部署在云环境中是低成本和平台独立的。所提出的技术也已在所有可用的UNM(新墨西哥大学)数据集上得到验证，并且在用于检测侵入过程的程序范围检测中具有98％的准确性。

## 1.3 研究内容

针对现有的基于机器学习的检测方法通常以权限或API为特征表示程序行为，这种方法存在以下几种问题：(1)对所有权限与API进行提取，会造成特征维度过高，从而降低了检测效率；(2)单一使用API或权限特征无法充分体现恶意应用程序的特性，造成一定误报率；(3)应用程序的不同特征在不同机器学习分类算法上的表现会有所差异，单一的机器学习分类算法不能有效的发挥应用程序不同特征在恶意应用程序检测上所起到的作用。针对机器学习的检测方法存在的不足，本文设计了一种基于集成学习的多特征Android恶意代码检测模型。通过分析正常应用程序和恶意应用程序的特性，本文将权限特征、特征以及资源特征作为特征信息。重点是对这些特征进行处理，得到高效的分类Android应用程序的方法。主要研究内容如下。

(1)本文总结了恶意应用最常用的50个权限与39个API，并将其作为特征集。通过对大量恶意应用程序进行分析后发现，大部分API和权限不能够有效区分恶意应用和正常应用，对所有权限与API进行特征提取，会造成大量冗余数据。因此对本文所总结的特征集进行特征提取，可以减少特征提取维度，提高特征提取效率。

(2)本文提出了一种基于多特征的Android恶意应用程序检测方法。在保留API和权限的基础上，增加了组件、图片、界面元素等资源文件特征，最后将这些特征保存到同一个特征向量集合中，并采用机器学习算法构建了一个多特征检测模型。实验结果表明，新增的资源特征容易提取，而且与单一特征检测方法相比，基于多特征的检测方法表现出了出更好的准确率。

(3)本文提出了一种基于集成学习的多特征Android恶意应用程序检测方法。将提取的权限、API以及资源特征分别用不同的分类算法进行训练并将其结果作为子模型进行集成学习，选取每类特征的最优算法，采用加权投票选择的方式对最优算法中的分类结果进行特征融合。实验结果表明，该集成方法有效弥补了传统检测方法的不足，从而能更加高效地检测Android恶意应用程序。

## 1.4 论文结构安排

第一章绪论。首先对恶意应用程序检测的背景和意义做了简单介绍。接着对国内外恶意应用程序检测技术的发展现状进行了详细的阐述。最后介绍了论文的结构安排。

第二章相关知识介绍。本章首先对系统框架、软件结构与系统四大组件做了介绍，接着介绍了沙箱隔离机制、权限声明机制等安全机制，还对恶意代码常用技术做了详细的介绍，最后对恶意软件的分类进行了描述。

第三章Android软件静态特征的提取。本章重点剖析了权限特征对恶意应用程序检测的意义，特征对恶意应用程序检测的意义和资源特征对恶意应用程序检测的意义，并且对每种特征的提取方法做了详细的介绍。

第四章基于集成学习多特征的恶意应用程序检测模型。本章分别详细阐述了基于各类单一特征的检测模块的构建。最后对集成学习算法进行了简单介绍，并结合本文数据提出了改进Bagging集成学习分类技术。

第五章实验部分。本章首先对本文的实验环境做了简单介绍，接着对实验数据进行了详细的说明，并分别设计了多组对比实验对每一类特征的表现能力进行评估。最后通过与之前其它检测方法的对比，验证了本课题提出的检测方法的有效性。

第六章总结与展望。主要是对本文的研究内容和创新点进行总结归纳，并对未来恶意应用程序的研究方向进行展望。

2 Android相关知识概述

本章首先对系统框架、软件结构与系统四大组件做了简要的概括，接着介绍了沙箱隔离机制、权限声明机制等安全机制，还对恶意代码常用技术做了详细的说明，最后对恶意软件的分类进行了描述。

## 2.1 Android系统介绍

2.1.1 Android系统框架

系统框架由内核层、核心类库层、系统框架层、应用程序层构成，各层都有自己的分工，层层隔离，互不影响。具体如图2.1所示。



图2.1 Android系统架构

内核位于系统框架的最底层，系统的的运行环境是由虚拟机提供的，每个Android应用程序想要运行自己的进程，必须由虚拟机为其专门分配实例。与传统操作系统的内核不同，系统的内核是对传统操作系统的修改和优化过的定制版本，这样可以保证系统与嵌入式环境更加匹配。以内核为基础，系统可以实现资源管理、网络通信、程序运行等核心功能。除此之外，针对移动终端的特点，团队对内核做了一些改进，包括提供了下全新的进程间通信机制—，以及匿名共享内存、USB Gadget驱动以及对设备进行定时控制功能等特性。

核心类库层的系统库提供了C/C++库，使得开发者可以在自己的应用中调用一些C/C++本地库中的接口实现某些功能。运行库包括两个部分：一个是函数库，另外一个是运行环境。函数库主要是为Android系统提服务，它包括C和C++库，当应用程序运行时，函数库为其提供运行时所需要的函数调用。运行环境由两部分组成，一个是虚拟机，每个应用程序想要运行自己的进程，必须由虚拟机为其专门分配实例，另外一个是具有语言核心功能的函数库，它提供应用运行时所需要的各种函数。

系统框架层隐藏在应用程序层之后，实际上系统框架层隐藏与应用程序层息息相关，应用程序所有的功能实现都在系统框架层中有相对应的API接口，这些API接口由JAVA语言实现，当应用程序想要实现具体功能时，直接调用系统框架层提供的API接口就可以。系统框架层主要有Windows界面管理模块、管理生命周期的模块、程序管理模块与网络通信模块等。

应用程序层是Android手机上所有应用程序的总称，它暴露在系统框架的最外面，可以与用户直接产生联系，应用程序层包含两部分应用程序，一部分是谷歌第三方市场提供的，另一部分是系统自带的，这两部分应用程序可以几乎满足用户的所有需求，例如有可以打电话的拨号程序，有运算功能的计算机应用，有视频播放功能的抖音、腾讯视频等播放器。

2.1.2 Android软件结构-APK

应用程序的文件格式是以形式存在的，文件实际上是一种压缩文件，用解压工具对其解压，解压后文件的结构如图2.2所示。这些文件分别用于不同的功能，常用的文件和文件夹如下：



图2.2 Android系统架构

Assets：该目录通常会包含一些多媒体资源(声音、字体、网页等)。

Res：该目录下包含应用程序中用到的图片，界面元素、XML文件等资源。

META\_INF文件：含MANIFEST.MF文件与CERT.RSA文件(证书信息)。

AndroidManifest.xml：该文件是应用程序的配置文件。

Classes.dex：JAVA项目中的类编译后的结果，在虚拟机上运行。

Resources.arsc：存放的诸如XML文件等预先经过虚拟机编译的资源文件。

Libs：存放的是应用程序中打包好Jar文件，在使用时直接导入。

2.1.3 Android系统应用程序组件

Android系统中有五大组件，它们拥有各自的生命周期以及自己独特的作用，Android应用各种功能的实现，其实就是对各个组件的调用，下面对五大组件做一个简单的介绍。

1. Activity：应用程序中，Activity组件可以与用户直接产生联系，它相当于应用程序与外界联系的一个桥梁，Android用户想要操作应用程序，最直接的就是对Activity组件进行操作，每个应用程序都有自己独立组件，相当于有自己独立的与外界联系的窗口，不同Activity间可以进行通信，但是必须要借助于Intent组件，应用程序想要使用Activity组件必须就必须要在配置文件中声明。
2. Service：应用程序与用户直接产生联系的组件，那要是在后台运行程序，就需要用到服务，简单理解就是，用户想要使用应用程序的具体功能，当点击屏幕上可以与用户交互的模块时，就会触发系统的Activity组件，但是Activity组件会把命令传递给下一层，也就是组件，由完成用户的功能需求，组件与Activity组件一样，它们的功能都是封装在自己独自的空间中，符合Android的设计理念，组件的出现，为一些经常需要运行在后台但与用户交互很少的应用程序，提供了完美的解决办法。
3. Content Provider：Android系统的安全机制决定了不同的Android应用程序具有不同的内存空间，除了一些存储在公共存储空间的数据，不同的应用程序间对自己的私有数据是严格保护的，但是如果不同的应用程序间想要交流，而应用程序又对自己的私有数据严格保护，这时候Content Provider的出现完美的解决了这些问题，它提供了不同应用程序间通讯的可能性，应用程序通过Content Provider暴露自己的私有数据，其它应用程序可以通过Content Provider获取自己想要的信息。
4. Broadcast Receivers：不同应用程序间进行通信时，往往一个应用程序发出或接收的指令无法得到及时的反馈，这种情况会造成较大的系统开销，Broadcast Receiver能够很好地解决这种情况，当应用程序进行通信时，Broadcast Receiver接收到别的应用程序发送来的指令信息，会及时进行反馈，发送指令的应用程序接到反馈时便会去执行别的操作，这样可以有效的减少系统开销，使进程间通信更加高效。
5. Intent：在Android系统中，由于各个组件对应着不同的功能需求，它们之间也是单独的个体，当它们想要进行信息交流时，就需要一种媒介，这种媒介在Android中就叫做Intent组件，Intent组件在不同组件之间进行任务沟通，当一个应用程序请求或接收别的应用程序的指令信息时，会先向系统发送Intent请求，系统接收到请求时，便会进行处理，完成相对应的数据信息处理，这样就完成了不同组件之间的沟通交流。Intent与四大组件的关系如图2.3所示。



图2.3 Android系统组件关系

## 2.2 Android安全机制

系统在继承了的安全机制以外，还实现了一套自身安全机制[36]，下面对其进行详细介绍。

2.2.1 进程沙箱隔离机制

系统基于内核，在中，每一个用户对应一个，在Android系统中，用户成为了一个个应用。安装应用时，Android系统会给应用赋予一个UID，且在存续期间，恒定不变。运行于Dalvik虚拟机实例的应用程序会被分配单独的资源和空间，不同的应用程序之间相互隔离。如果希望两个应用共享权限、数据，可以通过设sharedUserID来声明两个应用使用同一个UID，运行于同一进程，并共享其资源和权限。“沙箱机制”的实现方式如图2.4所示。



图2.4 沙箱机制

2.2.2 应用程序签名机制

所有Android应用都具有自我认证机制，在发布应用时必须打包为APK形式且进行签名。其中签名的作用是标识应用程序的作者以此建立信任关系，而不是用来判断该应用程序是否需要被安装。APK签名具有以下优点：(1)应用程序升级：如果两个应用程序包名相同但签名不同，则系统会拒绝安装新版本。(2)应用程序模块化：系统将同一签名的多个应用系统看作一个单独的应用程序，并对其进行模块化处理，因此用户只需改变其中任意一个，其他就会跟着发生变化。(3)代码或数据共享：两个签名相同的应用程序，可以相互公开自己的功能和数据。(4)应用程序的可认定性：开发者的签名可防止应用程序伪造现象的发生。在签名的过程中，若使用私有密钥调试，则其不能发布在Google Play市场中，被称为调试模式应用程序。如果开发者要在Google Play上发布自己的应用程序，那么必须使用私有钥匙方可验证其应用程序的来源。

2.3.3 权限声明机制

权限作为通行证，是软件拥有某种行为的权利。Android通过权限机制保护设备资源以及用户数据，当某应用程序要在某些资源对象上执行操作时，首先必须获得相应的权限。

Android系统在Android.mainfest.permission类中有一套定义好的权限，程序员在开发Android应用时可以根据要访问的资源申请相关权限。并且，Android系统允许用户自定义权限，从而实现资源共享及资源保护。在安装应用程序时，系统首先提示用户授予相关权限，这些权限既可以是系统给定的也可以是用户自定义的，当应用想要访问某些敏感资源时，系统会判断应用是否具有该权限，如果有，则可以完成敏感资源的访问，否则访问失败。所以，Android系统之所以安全是因为任何应用程序在访问敏感资源时都要首先获取相应权限。  
 Android应用程序在AndroidMainfest.xml文件中用<use-persimission>标签来申请权限。在客户端安装应用程序时，提醒用户是否授予该权限。比如，一个用户需要申请一个拨打电话的权限：  
   
 自定义权限使用<permission>标签来完成，以此来限制其他应用程序对本应用某些组件或程序的访问。在安装app时，自定义权限被加入到系统中。程序员自定义权限时可对权限危险级别进行限定，并分为、、和这4个等级。

2.3.4 进程通信机制

在Android系统中，应用程序的功能主要是由Activity和Service完成的。组成应用程序的Activity和Service既可以在同一个进程中运行，也可以在不同的进程中运行。当二者不在同一进程时使用Binder进行通信完成的。Binder实现了不同进程之间的通信功能。其中Binder驱动为进程之间Binder通信的建立，Binder在进程之间的传递等一系列底层支持。而Service Manager负责Service的管理，使得Client能够获取到Service代理对象的服务。Client和Service便是通过Binder驱动程序实现的。Client、Server、Service Manager和Binder驱动程序共同组成了Android系统的Binder，上述4个组件工作原理如图2.5所示。



图2.5 Android系统Binder机制中的四个组件的关系

## 2.3 Android平台下恶意应用常用技术分析

2.3.1 恶意代码载荷技术

常见的恶意代码载荷技术包括三种，Smali注入技术、APK动态加载技术与ELF文件加解密技术，Smali注入技术就是在对应用程序进行反编译后，在其对应的Smali文件中注入一些指令信息，以此达到更改原来应用程序行为的目的。APK动态加载技术是指应用程序在下载更新过程中，将原来加密并且存储在应用程序中的恶意代码，利用远程操控应用程序，解密并且执行恶意行为。文件加解密技术[37]：为了对抗Android逆向工具，恶意程序利用动态链接库对Android核心部分进行加密，加密的流程是这样：解析ELF→找到字节码→对字节码加密。解密就是解析ELF→找到字节码→字节码解密。详细一点就是通过偏移、个数等信息找到section的头部，然后看是不是我们要找的section(通过名字)。找到后通过sh\_offset(偏移)和sh\_size(大小)，就找到这个section的内容，整体加密。

2.3.2 Android平台恶意软件事件监听技术

Android系统使用监听器与回调两种方式在事件发生时做出响应[38]，监听器是指在应用程序的某个事件发生后，该事件的监听者监听到此信息后，采取相应的措施去处理，在这种情况下，事件的发生者与处理者互不相关。回调是指在应用程序的某个事件发生后，有自己本身去处理该事件，来对事件进行处理，在这种情况下，事件的发生者与处理者是同一个体。而恶意应用程序经常会使用监听器机制监听一些系统发生的事件，从而产生一些恶行为。恶意软件常使用Broadcast Receiver组件来监听系统事件，对于Android系统发出的广播，使用Broadcast Receiver组件进行监听，并采取相对应的措施，在之前的Android系统应用程序组件中介绍了Intent组件，它可以架起了消息传递的桥梁，并且可以在不同组件之间进行任务沟通，当然用它来发送广播也是可以的。系统在配置文件中定义好组件信息，并将接收广播后要执行的操作也放入配置文件中，然后将该其注册到Android系统，这样就能对Android系统广播进行监听。图2.6为Android系统广播机制的工作流程。



图2.6 Android系统广播机制

2.3.3 恶意代码数据发送技术

恶意应用程序将用户的数据信息获取后，为了能够被其所利用，用户的数据信息会发送到服务器，从而被恶意应用非法获利。Android平台下恶意代码发送数据的方法分为SmsManager、Socket与Http发送协议三类：(1)SmsManager：恶意应用程序通过实例化SmsManager类，并且调用信息发送方法，就可以将用户数据发送出去；(2)Socket：恶意应用通过Socket通讯与服务器建立联系，应用程序将数据发送给服务器，服务器接收后会给应用程序一个反馈，以此建立联系，从而将用户数据发送出去。(3)Http发送协议：超文本传输协议发送协议有HttpClient和HttpURLConnection两个接口，它们都有Get与Post两种请求方式。

## 2.4 Android平台恶意软件分类

根据文献[39]可知，对Android恶意应用的分类，不同的文献中有不同的分类方法方法，下面介绍两种分类。按照传统的分类方法，恶意软件可分为如下几类：

**表2.1 按传统方式Android恶意软件分类**

|  |  |  |
| --- | --- | --- |
| 分类 | 恶意软件名称 | 恶意软件行为 |
| 木马类 | Opfake | 伪造[界面](https://www.baidu.com/s?wd=%E7%95%8C%E9%9D%A2&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/ybdesire/article/details/_blank)，逃脱杀毒软件检测 |
| FakeInstaller | FakeInstaller会在安装过程中向付费号码发送短信，它大量存在于流行应用的重打包版本中 |
| Fakeplayer:Spitmo | 被注入到银行应用中，偷盗用户的银行账号密码 |
| FakeDoc | 安装其他应用程序 |
| Iconosys | 窃取用户隐私 |
| 病毒类 | Obad | 蠕虫病毒，自我复制和传播 |
| 后门类 | DroidKungFu | 利用漏洞Root系统，伪装自己 |
| DroidDream | 在用户不知情的情况下下载恶意程序，它通常在用户晚上睡觉的时候发作 |
| Kmin | 盗取信息并传回server端 |
| GinMaster | 会root设备，获取权限 |
| Basebridge | 它会kill掉安全应用，并发送获利短信订阅增值服务 |
| 僵尸类 | Geinimi | 接受远程指令，控制手机端，收集手机信息传回server端 |
| AnserverBot | 接受远程指令 |
| Plankton | 将[手机信息](https://www.baidu.com/s?wd=%E6%89%8B%E6%9C%BA%E4%BF%A1%E6%81%AF&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/ybdesire/article/details/_blank)如IMEI以及用户浏览器浏览历史数据传到远程服务器 |
| 间谍类软件 | GPSSpy | 伪装成其他APP，上传用户的GPS信息 |
| Nickyspy | 记录受害者的电话拨入拨出信息，track用户GPS信息，向其它号码发送短信 |
| 恐吓类软件 | Koler | 显示信息，告诉用户你不交钱就要把你送进监狱 |
| 勒索类软件 | Fakedefender.B | 显示安全报警，说服用户购买服务来删除不存在的malware |
| 广告类软件 | Uapush.APP | 广告软件+偷窃设备信息 |
| 跟踪类软件 | SMSTracker | 窃取用户的行为以及习惯信息 |

Linux内核存在于Android系统的最底，Android系统的内核是对传统操作系统的修改和优化过的定制版本，一些不法分子就会利用Android系统的这个特征，将一些恶意代码部署在应用程序中，从而造成一系列的恶意行为。从另一个角度来看，是否具有恶意代码以及是否会产生恶意行为是判别恶意应用程序的一个重要指标。根据Android恶意应用的特征，对其进行分类，结果如表2.6所示。

**表2.2 根据Android恶意应用的特征来进行分类**

|  |  |
| --- | --- |
| 恶意特性 | 表现形式 |
| 恶意安装 | 重打包 |
| 更新攻击 |
| 诱导下载 |
| 恶意载荷 | 提权攻击 |
| 远程控制 |
| 恶意扣费 |
| 收集信息 |
| 使用权限 | 获取意思数据，位置等 |

## 2.5 本章小结

本章首先对系统进行了简单的介绍，如系统框架、软件结构与统四大组件等，接着对Android安全机制做了介绍，如沙箱隔离机制、权限声明机制等等，还对恶意代码载荷技术、事件监听技术以及恶意代码数据发送技术等恶意代码常用技术做了详细的介绍，最后对Android恶意软件的分类进行了描述。

3 Android软件静态特征提取

传统的Android恶意软件的检测方法主要是应用申请的权限组合或敏感函数调用，而且传统的检测方法普遍存在的问题是特征选取不够全面，导致检测方法只能针对特定的数据集才有效。单一使用API或权限特征无法充分体现恶意应用程序的特性，造成一定误报率，而且对所有权限与API进行提取，会造成特征维度过高，从而降低了检测效率。针对传统检测方法的缺陷，本文首先总结了恶意应用最常用的50个权限与39个API，并将其作为特征集。在保留权限与API特征的基础上，添加了资源文件特征，形成新的多特征集合；同时本章还对每种特征分的提取过程做了详细的介绍。

## 3.1 恶意代码常用权限的分析与提取

3.1.1 权限特征的分析

Android系统中的权限机制是用来协调系统与应用程序的，应用程序只有在配置文件中申请了权限，它才能够进行一些具体操作，然而恶意应用也会常常使用一些敏感权限进行恶意操作，例如定位权限，获取WiFi权限，文件读取权限等。所以本文对恶意应用经常使用的一些权限进行提取与总结，并将其中的一些权限进行组合。表3.1中列出了常见的隐私权限。

**表3.1 常见隐私权限及描述**

|  |  |
| --- | --- |
| 危险权限名称 | 权限描述 |
|  | 短信发送权限 |
|  | 防止卸载 |
|  | 来短信时广播提醒 |
|  | 开启时启动 |
|  | 对系统敏感的读写权限进行设置 |
|  | 杀死进程 |
|  | 系统存储权限 |
|  | 系统任务权限 |
|  | 获取手机状态 |
|  | 锁定时唤醒 |

应用程序通过申请相应的权限来获取系统资源，恶意应用通常使用敏感权限产生恶意行为，在某些情况下，恶意应用单独使用某一个权限不会造成恶意行为，而在使用多个权限的情况下，以一定的方式进行权限组合则会产生恶意行为。例如一个视频播放软件在申请了联系人读取权限的同时还申请了数据发送权限，这样这个视频播放软件便有可能通过网络将用户的联系人信息发送出去。本论文引用文献[40]提出来的一系列恶意权限组合，根据该组合可以向用户提示恶意应用可能存在的恶意行为。部分恶意权限组合如表3.2所示。

**表3.2 部分恶意权限组合**

|  |  |  |
| --- | --- | --- |
|  | 序号 | 恶意权限组合 |
| 恶意扣费 | 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 获取并发送用户短信 | 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 获取位置信息并发送 | 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 获取用户其他隐私信息 | 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |

3.1.2 权限特征的提取

应用程序申请的权限必须在配置文件中声明，当应用程序想要执行具体操作时，如果应用程序在配置文件中申请了该权限，则应用程序可以执行此操作。如果应用程序在配置文件中申请了该权限，但是在实际使用操作过程中却没有用到，这样就造成了过度申请权限的现象，也就是说配置文件中声明的权限不能充表征应用程序的行为特性，为了避免本文提取的权限特征存在过度申请的权限，同时也为了提取到的权限特征能够更加准确的代表应用程序的行为特性，本文提取了应用程序实际使用的权限，对APK进行反编译后的smali文件进行关键字invoke遍历，找到应用程序所有的API调用信息，根据PScout提供的API-权限对于关系，进行信息匹配，这样得到的权限信息就是应用程序实际使用的权限信息，具体的步骤如下。

(1)权限提取流程：

从Android程序反编译代码中提取权限的流程如图3.1所示。



图3.1权限提取流程图

API-权限映射关系：系统使用权限对系统资源进行访问控制，通过权限声明机制，能够有效的保护系统与应用程序的安全，应用程序每一个操作的执行都是首先获得权限申请，然后调用API去执行相应操作，每个权限都与多个API对应，在本文中使用最近发布的版本，该版本几乎覆盖了系统中所有的权限信息，并且记录了每个权限与API对应关系。

反汇编代码：由于本文中需要对应用程序的API信息进行提取，但是对于一个给定的APK，它实际上是一个压缩文件，无法直接从中获取API信息，对APK进行解压后得到的classes.dex文件是在虚拟机上运行的，该文件晦涩难懂，给我们的特征提取造成很大的困难，因此我们需要对APK进行反编译，这样可以得到Java语言编译后中间代码smali文件，这样极大的减少了处理难度。本文使用的反编译工具是APKtool。图3.2是APK文件解压缩后的文件夹。

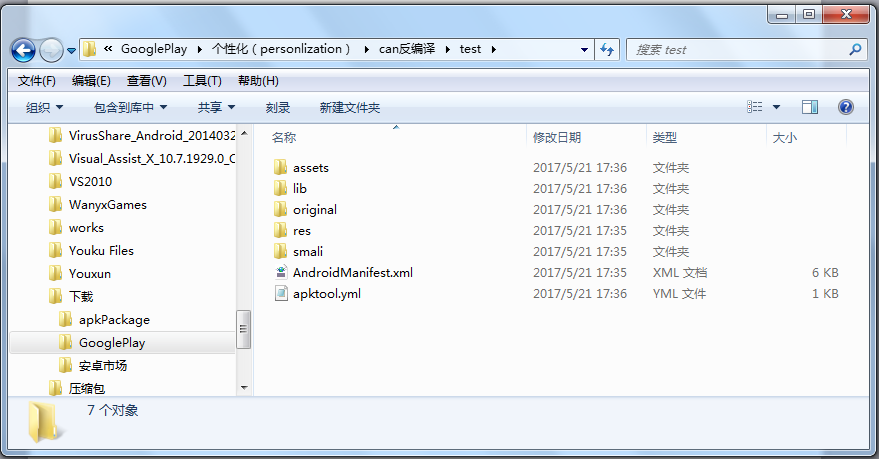


图3.2 APK文件解压缩后的文件夹

API调用信息：应用程序的所有API调用信息都在反编译后的smali文件夹下，对smali文件夹下的所有文件进行深度优先遍历，找到所有smali文件，并对其进行invoke关键字进行查询，将所有匹配到的API信息输入到一个文件中，这个文件就是本文所需要的API调用信息。

信息匹配：应用程序使用的权限信息很难在API调用信息中看出来，为了获取应用程序实际使用的权限，就需要将API调用信息与Pscout中的API与权限的对应关系进行匹配，将Pscout中API与权限的对应关系读入数据库中，然后通过对API调用信息中的每条API信息进行读取，并将每条API信息在数据库中进行查找匹配，找到对应的权限信息，并将查找匹配到的权限信息写入到一个文件中，这个文件中的权限信息就是应用程序实际使用的权限。

(2)权限提取算法

应用使用权限提取：对反编译后的smali文件夹进行遍历，获取每个smali文件的文件，对得到的smali文件进行API信息提取，并将得到的API信息与Pscout中的信息进行匹配，得到应用程序实际使用的的权限信息。算法的伪码如代码3.1所示。

**代码3.1 提取使用权限的算法**

|  |
| --- |
| 输入： filePath //smali文件夹路径  *L*API//敏感API集合  *L*API-permission //API-权限映射表  输出：*L*all-permission //权限报告  function get\_smali\_permission(filePath)  1. fileList=filePath.list(); // 获取该文件夹下的所有文件  2. for i=0 to fileList.Length //对所有的文件进行遍历  3. {  4. *L*smali-API=chackStr(fileList[i],*L*API); //提取fileList[i]文件的敏感API调用信息  5.  *L*permission=getPermission(*L*smali-API, *L*API-permission ); //匹配API调用对应的权限信息  6. *L*all-permission=*L*all-permission∪*L*permission*;* //将得到的权限写入总的API权限报告  7.  *L*smali-API=null;  8. *L*permission=null;  9. }  10. return *L*all-permission; |

## 3.2 恶意代码常用API分析与提取

在第二章中本文对恶意代码的常用技术进行了分析，发现恶意代码有很多共同的行为特征，本文将通过研究提取出恶意代码常用的行为API。

3.2.1 系统API的行为分析

(1)隐私窃取行为

隐私窃取是恶意感染的重灾区，恶意应用程序会将用户的隐私信息泄露出去，从而对用户移动安全造成威胁。用户的隐私信息包括短信信息、位置信息与联系人信息等。恶意应用获取短信信息有SmsMessage.createFromPdu和ContentResolver.query两种方式；获取联系人信息可以在联系人数据库中使用query方法进行查询；获取应用程序的位置信息可以使用Context.getSystemService方法获取，对拍照以及录音数据的获取分别使用MediaStore.action与MediaRecorde.start方法。通过对恶意应用程序对用户隐私窃取的研究，总结获取不同隐私数据的API调用信息，将其API调用提取如表3.3所示。

**表3.3 隐私窃取类行为API**

|  |  |
| --- | --- |
| 方法名 | 说明 |
|  | 读取短信 |
|  | 读取短信 |
|  | 读取联系人 |
|  | 定位 |
|  | 录音 |
|  | 拍照 |

(2)恶意隐藏行为

为了不让检测软件检测到自己，恶意应用程序经常使用一些技术手段来对躲避检测，恶意应用常用的技术手段包括对桌面应用程序的图标进行隐藏，以及在后台下载时隐藏下载行为，不让用户知晓。对桌面应用程序的图标进行隐藏是通过执行PackageManager.setComponentEnabledSetting方法对MainActivity组件禁用实现的；恶意应用隐藏下载行为是将原来加密并且存储在应用程序中的恶意代码，通过远程操控应用程序，解密并且执行恶意行为。恶意隐藏行为的API如表3.4所示。

**表3.4 恶意隐藏行为API**

|  |  |
| --- | --- |
| 方法名 | 说明 |
|  | 应用图标隐藏 |
|  | 动态加载 |
|  | 内容下载 |

(3)流氓行为

常见的恶意应用的流氓行有屏幕锁定、诱导安装以及勒索用户。屏幕锁定是指当用户打开一个界面时，用户无法退出这个界面，只能在后台关闭应用程序，才能关闭界面。诱导安装是指应用程序会通过短信或者其它途径向用户发送链接，当用户点击链接进入界面，就会自动安装一些应用程序，勒索用户是指，恶意应用会将用户的SD卡上的内容进行加密，然后复制一份后并将原来的SD卡上的数据删除，以此来要挟用户。表3.5是一些流氓行为所用到的API。

**表3.5 流氓行为API**

|  |  |
| --- | --- |
| 方法名 | 说明 |
|  | 发送通知 |
|  | 屏蔽锁屏代码 |
|  | 应用安装 |
|  | 打开网页 |
|  | 访问和删除SD卡信息 |

(4)数据发送行为

恶意应用程序将用户的数据信息获取后，为了能够被其所利用，用户的数据信息会发送到服务器，从而被恶意应用非法获利。Android平台下恶意代码发送数据的方法分为SmsManager、Email、Bluetooth、Socket与Http发送协议五种。在SmsManage数据发送中，应用程序通过实例化SmsManager类，并且调用信息发送方法，就可以将用户数据发送出去。Email数据发送有两种，一种是使用系统邮件发送，另一种是使用JavaMail邮件接口，使用系统邮件发送极容易引起用户的注意，使用JavaMail邮件接口的sendMessage方法便可以发送数据。Bluetooth数据发送中，使用connect便可以与外界建立联系，从而将数据发送出去。恶意应用通过Socket通讯与服务器建立联系，应用程序将数据发送给服务器，服务器接收后会给应用程序一个反馈，以此建立联系，从而将用户数据发送出去。超文本传输协议发送协议有HttpClient和HttpURLConnection两个接口，它们都有Get与Post两种请求方式。表3.6是部分数据发送行为的API。

**表3.6 数据发送行为API**

|  |  |
| --- | --- |
| 方法名 | 说明 |
|  | 发送短信 |
|  | 发送邮件 |
|  | 建立蓝牙通信 |
|  | 连接请求 |
|  | 建立通信连接 |
|  | 中建立通信 |

(5)其他恶意行为

除了上述提到的一些恶意行为外，恶意应用还会产生一些其他的恶意行为，例如在用户不知情的情况下，静默消耗手机流量；杀死应用程序进程，并强制退出；获取手机状态，例如手机型号、手机号码等。表3.7是部分其他恶意行为的API。

**表3.7 其他相关恶意行为和流量检测API**

|  |  |
| --- | --- |
| 方法名 | 说明 |
|  | 杀死进程 |
|  | 关闭应用 |
|  | 获取手机卡状态 |
|  | 删除短信 |
|  | 卸载应用 |
|  | 应用接收的所有字节流量 |
|  | 应用发送的所有字节流量 |

3.2.2 系统API特征的提取

在提取API调用前，首先，将APK文件利用APKTool反编译，生成smali格式的反汇编代码。由于APKTool在对apk进行反编译后，在反编译文件中包含一个smali文件夹，该文件夹下是APKTool为每个类生成的.smali文件，因此，通过遍历smali文件夹即可获取到apk的API调用信息。

由于Smali格式的代码调用类方法的基础指令为“invoke”指令。如Android.os.Parcel→obtain()方法调用对应的samli代码为invoke-static{}，Landroid/os/Parcel;→obtain() Landroid/os/Parcel。因此，本文通过迭代遍历smali文件夹下的每一个.samli文件，然后逐行扫描该文件，以“invoke-”作为待匹配字符串，提取所有以“invoke-”开头的行，然后进一步处理，获取被调函数的原型信息，即函数名、参数类型与返回类型三部分，并将该信息写入特定的api(i).txt文档中，即可完成API特征的提取。

## 3.3资源特征的分析与提取

由于本文中需要对应用程序的资源特征进行提取，但是对于一个给定的APK，它实际上是一个压缩文件，无法直接从中获取资源信息，因此我们需要对APK进行反编译，这样可以得到能够用语言直接处理的资源文件，极大的减少了处理难度。本文使用的反编译工具是。对反编译后的文件进行扫描，可以得到实验想要的数据。其中在文件中可以提取到所需要的图片资源，在文件夹下可以得到界面元素特征，在配置文件中可以得到组件信息。

3.3.1图片特征

APK经过反编译后的res文件存放着Android应用的资源信息，而图片信息则存放该文件下的drawable文件中，像素越高图片越清晰，在本文中根据像素密度把图片分为四类：像素密度最低的图片ldpi、像素密度中等的图片mdpi、像素密度高的图片hdpi和像素密度最高的图片xhdpi。统计所有出现在恶意应用与正常应用中不同像素密度图片的数量，如图3.3~图3.7所示。其中X轴表示不同像素密度的图片数量，Y轴表示包括小于对应X轴的图片数量占全部应用的百分比。通过分析，我们得出以下结论：

(1)不同像素密度的图片会在正常应用与恶意应用的分布中有明显的差异。

(2)正常应用更倾向于使用像素密度较高的图片，而恶意应用恰好相反。

(3)正常应用使用较多的xdpi像素的图片，但恶意应用几乎很少使用。

(4)正常应用中图片总数与恶意应用也有明显的差别，正常应用使用图片总数超过50张的个数占总数的85%，而恶意应用却连10%都不到。

因此可以把不同像素的图片数量当做区分软件是否具有恶意性的重要指标。



图3.3 ldpi图片数量对比 图3.4 mdpi图片数量对比



图3.5 hdpi图片数量对比 图3.6 xdpi图片数量对比

图3.7图片总数对比 图3.8 activity数量对比

针对以上结论，本文对一些恶意样本进行分析发现：恶意应用常常使用特别低像素图片的目的是为了能够将自身的企图掩盖起来，例如：恶意应用会故意将界面中的按键开关伪装成一个清晰度很低的图像，在使用者没有看到这个图像的情况下，无意的点击此按键开关，恶意的行为就会发生。这就促使恶意应用过多的使用清晰度较低的图像，故而mdpi与ldpi图像被选择的概率变得非常大；而清晰度较高的图像则不能很好的掩盖恶意应用的企图，故而xhdpi与hdpi图像被选择的概率会很小。正常应用使用图片的情况正好与之相反，正常应用没有必要掩盖自身的目的，相反的因为其自身功能的需求，正常应用会使用像素较高的图像，除此以外，由于正常应用相对于恶意应用会使用到更多的图片，因此在图片总数上也要比恶意应用多。

3.3.2组件特征

Android系统中应用程序的各个功能的实现离不开组件，其中不同的组件拥有各自的生命周期以及自己独特的作用，Android应用程序通过对Android四大组件的调用来完成一系列的功能。例如，Activity组件可以与Android用户直接产生联系，它相当于应用程序与外界联系的一个桥梁，Android用户想要操作应用程序，最直接的就是对Activity组件进行操作；service组件存在于系统后台，它的工作是服务于Android系统，为系统功能的实现提供保障。通过对大量样本分析后发现，恶意应用程序经常的在我们没有意识到的情况下使用一些组件进行恶意操作，所以会造成在正常应用程序与恶意应用程序对组件的使用频率上存在不同。因此，本文中分别提取应用程序的四大组件的频次，并将其作为特征来构建分类器。

Android系统中的组件信息声明在AndriodManifest.xml文件中，每个组件都是以名称(、、和content Provider)为关键字进行声明，每个组件的具体功能在各个组件的代码块中进行描述，当应用程序多次使用同一组件，必须在配置文件中进行多次声明，这样应用程序将会根据配置文件中的组件声明来整体进行调度。考虑到同一组件在配置文件中会进行多次声明，且不同代码块中组件的功能也不相同，因此在对组件特征进行提取时，将组件出现的频次也作为影响因素，对AndriodManifest.xml文件进行遍历，分别以<关键字为索引进行查找，将每个组件出现的次数存储在一个特征向量中，用表示，其中是界面组件的频次，是广播触发器出现的频次，是服务组件组件出现的频次，是内容提供器出现的频次。上图3.8为的数量在正常一样与恶意应用之间的一个对比。

3.3.3界面元素特征

应用程序中的界面元素包括按钮、提示信息、文本输入框等元素，这些界面元素可以与用户直接产生联系，例如当用户进行支付功能的时候，需要输入银行卡号与密码，这样就需要用到文本输入框与按钮这些界面元素，当支付完成时，也会收到提示信息这个界面元素。我们对应用程序的界面元素做了与图片相同的累计分布统计，统计所有出现在恶意应用与正常应用中的界面元素的数量，如图3.9~图3.11所示。其中X轴代表的界面元素个数，Y轴代表小于对应X轴的应用个数占全部应用的百分比。通过分析，我们得到以下结论：

1. 只有20%的恶意应用使用了元素，但却有63%的正常应用使用了元素。
2. 差不多有60%的恶意应用至少使用1个元素，而有85%的恶意应用会使用到10个以内的元素，使用10个以上的元素的正常应用占所有正常应用的85%。
3. 差不多有62%的恶意应用没有使用界面元素，而有85%以上的正常应用使用了界面元素。



图3.9 EditText界面元素对比 图3.10 TextView界面元素对比



图3.11 Button界面元素对比

针对以上结论，本文对一些恶意样本进行分析发现：通常，在Android提供的开发模板中，不会给出、等界面元素，这些界面元素需要用户自己定义。由于恶意应用的特点，它一般不会使用和这些界面元素，同时的数量也要比正常应用少。恶意应用很少使用出、等界面元素的目的是为了能够将自身的企图掩盖起来，例如：正常应用在进行一些隐私操作后，会弹出提示信息，告知用户，但是在恶意应用中，为了隐藏其恶意行为，恶意应用会将此界面提示信息删除，从而不被用户发现。因此，恶意应用很少使用、这两种提示信息类型的界面元素；正常应用使用界面元素的情况正好与之相反，正常应用没有必要掩盖自身的目的，相反的因为其自身功能的需求，所以相对应的界面元素回会比恶意应用要多。

## 3.4本章小结

本章首先对权限特征进行分析，总结了恶意应用常用的权限，并通过API-权限对应关系提取出了应用程序实际使用的权限特征。接着对恶意应用常用API行为进行总结，并对API特征提取过程做了详细介绍。最后对应用程序的资源特征进行提取，分析了Android应用程序的图片，界面元素以及组件等这些资源特征，发现这些特征在正常应用与恶意应用中有明显的差异。

4 基于集成学习的多特征Android恶意应用程序检测模型

针对应用程序的不同特征在不同机器学习分类算法上的表现会有所差异，单一的机器学习分类算法不能有效的发挥应用程序不同特征在恶意应用程序检测上所起到的作用这一问题，本章介绍了基于集成学习的多特征Android恶意应用程序检测模型，根据上文提取的三类静态特征，形成了三个检测模块，分别是基于权限特征的检测模块，基于系统API特征的检测模块和基于资源文件的检测模块，最后对每种检测模块的预测结果进行集成学习，生成了本课题提出的集成模型[41]，该模型弥补了单一特征的不足[42]，充分发挥了不同分类模块的优势。

## 4.1 基于权限特征的检测模块

基于权限特征的检测模块的建立，首先需要对获取的权限特征进行选择，然后对机器学习分类模型进行选择，根据选取的特征在不同分类器上的表现，选取性能最优的分类器，最后构建基于权限特征的检测模块。在特征选取时，要选取在区分正常与恶意应用效果最好的特征集合，尽量避免机器学习过拟合现象的发生。本课题中的数据集包括为668个正常应用和891个恶意应用，根据各种分类识别算法在不同特征集上的表现，选择分类能力最强的权限特征集合以及性能最好的分类算法。

(1)信息增益

由于权限与API特征提取过程中会产生大量数据，为了对所有权限与API特征在正负样本集上的区分能力进行排序，进而选择分类能力最强的全权限与API特征。本文采用信息增益的概念[43]，来研究每项特征对于判断一个应用程序为恶意软件的可能性的影响大小。信息增益公式定义如下：

|  |  |
| --- | --- |
|  | (4.1) |

其中表示类别，表示恶意样本，表示正常样本，代表包含特征，代表不包含特征，表示含有特征的样本数占样本总数的比例，表示不含有特征的样本数占样本总数的比例，表示恶意样本或正常样本占总样本的比例，代表出现特征的时，类别出现的概率。通过上述的计算可以分别获得每一个特征的信息增益值，信息增益值越大，则说明该权限对判别应用是否具有恶意性越重要。在本实验数据集上的出现频率差排名最高的top15权限如图4.1所示。

图4.1恶意与正常应用频率排名前15的权限

在正常样本与恶意样本出现频率差别最高的top15权限可以发现，正常样本中申请的较多的权限为ACCESS\_NETWORK\_STATE，READ\_SMS，INTERNET等权限，使用率都在94%以上，而恶意样本中申请的较多的权限为READ\_PHONE\_STATE，INTERNET，SEND\_SMS等，使用率最高能到达80%。因为当今社会互联网高速发展，Android应用于网络交互式必不可少的，所以INTERNET这个权限会被大量使用，通过对比发现恶意应用汇更倾向于获取用户隐私信息，以及一些状态信息。

**表4.1 信息增益值top10的权限**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Permission | IG | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 |
| READ\_SMS+INTERNET | 0.557440329 |  | | | | | |
| READ\_SMS | 0.550417761 |
| WRITE\_EXTERNAL\_STORAGE | 0.388434692 |
| ACCESS\_NETWORK\_STATE | 0.313277235 |
| GET\_TASKS | 0.273819998 |
| SEND\_SMS | 0.233864265 |
| READ\_PHONE\_STATE+SEND\_SMS | 0.180446079 |
| READ\_PHONE\_STATE | 0.108837668 |
| ACCESS\_COARSE\_LOCATION+INTERNET | 0.105841103 |
| ACCESS\_COARSE\_LOCATION | 0.101263882 |

为了可以更清晰的看出在正常应用与恶意应用对权限的申请差异，文中用信息增益的算法求出信息增益值top10的权限，信息增益值越大，则说明该权限对判别应用是否具有恶意性越重要。如表4.1所示，信息增益值较高的权限是大部分是对用户敏感数据的操作，这表明恶意应用会更多的使用对隐私获取、数据发送的权限。该实验结果证明，恶意应用常常申请一些高危权限来进行恶意操作。

(2)权限特征和分类算法的选择

基于权限特征的检测模块的建立，首先需要对获取的权限特征进行选择，在第三章中分析了恶意应用常用的权限以及权限组合，上面也分析了在本文的数据集上信息增益top10的权限组合，然后对机器学习分类模型进行选择，由于同一数据在不同的分类模型上的表现会不相同，所以选择最优的模型就显得尤为重要。本文最后使用的权限特征为信息增益值最大的前40维特征。机器学习分类算法选取了朴素贝叶斯(NB)，支持向量机(VM)，逻辑回归(LR)，K近邻(KNN)，随机森林(RF)。对这几个机器学习分类器的分类结果进行分析，我们最终选择随机森林算法来构建基于权限特征的检测模块。对于每一个Android应用程序，通过权限提取算法提取权限特征，然后将每个应用出现提取的权限表示成一个二进制向量，如果在应用程序中调用了第个权限，则该向量为，若没有则用0表示，建立基于权限特征的检测模块。

## 4.2基于系统API特征的检测模块

基于API特征的检测模块的建立与基于权限的检测模块的建立相同，首先需要对获取的API特征进行选择，对第三章提取的API特征，根据信息增益值的大小进行排序，选取区分正常与恶意应用效果最好的API特征。然后根据选取的API特征选择性能最优的机器学习分类模型进行分类，最后构建基于API特征的检测模块。

(1)系统API信息增益值排序

系统API特征的信息增益值的大小同样是公式(4.1)给出的公式进行计算。本节对API的信息增益的值大小进行计算。表4.2列出了系统API信息增益值top10的排序。从下表中可以看出，信息增益值较高的API包括DexClassLoader;→loadClass(动态加载APK执行)，SmsManager;→sendTextMessage(发送信息行为)，Socket;→connect(建立网络连接)等，这些API对区分未知应用是否为恶意应用有很好的效果。通过对比发现恶意应用更倾向于使用获取用户隐私信息，以及数据的发送等这些API。

(2)API特征和分类算法的选择

基于API特征的检测模块的建立，首先需要对获取的API特征进行选择，在第三章中分析了恶意应用常用的行为进行分类，提取其常用的API，上面也分析了在本文的数据集上信息增益top10的API，然后对机器学习分类模型进行选择，由于同一数据在不同的分类模型上的表现会不相同，所以选择最优的模型就显得尤为重要。机器学习分类算法的选取与基于权限特征的检测模型选取的算法一致。对这几个机器学习分类器的分类结果进行分析，我们最终选择信息增益值最大的前25维特征，机器学习算法选择随机森林算法来构建基于API特征的检测模块。实验的具体设计和结果分析在第五章的实验部分会详细给出。对于每一个Android应用程序，通过反编译APK文件得到smali文件，然后通过深度优先遍历算法遍历smail文件来提取系统API，将提取出的系统API表示成一个二进制向量，如果在程序的smali文件中含有该API，则该向量数量加1，若没有则用0表示，建立基于权限特征的检测模块。

**表4.2 信息增益值top10的系统API**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| API | IG | 0.1 | 0.2 | 0.3 | 0.4 |
| NotificationManager;→notify | 0.375887656 |  | | | |
| getOutputStream | 0.338335032 |
| DexClassLoader;→loadClass | 0.322963273 |
| URLConnection;→getInputStream | 0.289700224 |
| getLatitude | 0.273782359 |
| ContentResolver;→query | 0.250222695 |
| SmsManager;→sendTextMessage | 0.247682708 |
| HttpClient;→execute | 0.225202612 |
| getLastKnownLocation | 0.107937792 |
| Socket;→connect | 0.095978688 |

## 4.3基于资源特征的检测模块

在第三章中对资源文件的特征提取后发现，(1)在对图片特征进行分析，我们得出以下结论：不同像素密度的图片会在正常应用与恶意应用的分布中有明显的差异，因此可以把不同像素的图片数量以及图片总数当做区分软件是否具有恶意性的重要指标。(2)在对界面元素特征进行分析发现，Button、EditText、TextView界面元素数量对于区分正常与恶意应用有着重要的作用。(3)在对组件特征进行分析，恶意应用常常会在用户不知道的情况下调用系统组件，所以组件的调用频次在恶意应用于正常应用之间会表现出一定的差异性。

基于资源特征的检测模块建立，首先对第三章提出的图片、界面元素、组件这些特征进行提取，统计每个样本程序中各个资源特征出现的频度，并存储在特征向量中(vl，v2，v3，v4)，其中v1代表图片特征，v2代表组件特征，v3代表界面元素特征，v4代表XML文件数量与申请的permission个数，然后根据选取的资源特征选择适合的机器学习分类模型进行分类，选取性能最优的分类器，最后构建基于资源特征的检测模块。本节实验所用的资源特征的数据集与权限特征和API特征的数据集是一样的，处理方式也一样。我们最终支持向量机算法来构建基于资源特征的检测模块。

## 4.4基于多特征的检测模块

针对单一使用API、权限或者资源特征无法充分体现恶意应用程序的特性，造成一定误报率这一问题，本文提出了一种基于多特征的Android恶意应用程序检测方法。基于多特征的检测模块的建立，首先将上述提取的权限、API以及资源特征放入同一个向量空间，形成一个新的79维的一个多特征的向量，并将此向量放入适合的机器学习分类模型进行分类，选取性能最优的分类器。具体技术路线图如图4.2所示。



图4.2 多特征检测模型的技术路线

## 4.5基于集成学习的多特征Android恶意应用程序检测模型

4.5.1机器学习分类算法

分类简单的理解就是将一个未知的样本分成已知的类别，主要分为两部分来进行，第一部分，利用已知的样本，进行特征提取，并进行特征训练，建立模型。第二部分，通过第一部分已建立好的模型对未知的样本进行分类。故而好的模型是分类的关键，本文用到的分类算法如下：

(1)朴素贝叶斯分类算法

朴素贝叶斯[44]是计算复杂度较低的分类算法，该方法基于贝叶斯定理，并默认特征之间是相互独立。如式(4.2)所示，算法根据计算结果选择概率最大的分类：

|  |  |
| --- | --- |
|  | (4.2) |

其中表示待分类项，表示分类类别，表示条件概率。本文对判断一个应用是否具有恶意性，先分别求出该应用属于正常应用与恶意应用的条件概率，如果该应用在恶意应用的条件概率超过利50%，则该应用属于恶意应用，反之。数学形式描述如下：设为一个待分类项，其中为的特征属性，集合为类别集合，分别计算，，，。若则。

(2)支持向量机

SVM的原理[45]是找出一个符合条件的最优分类超平面，在确保分类性能的情况下，使该超平面两边的空白区域最大。

SVM的最优权值和最优偏置为：

|  |  |
| --- | --- |
|  | (4.3) |

式中，为支持向量，，，，。选用径向基核函数，RBF核函数为：

|  |  |
| --- | --- |
|  | (4.4) |

相当于核函数将之前的分类函数映射成：

|  |  |
| --- | --- |
|  | (4.5) |

式中，为支持向量，为未知的输入向量；。

(3)逻辑回归

逻辑回归是一种经典的分类模型[46]，不仅可以解决二分类，还可以解决多分类的问题，决策函数即可以是线性函数，也可以是非线性的函数。线性逻辑回归为：

|  |  |
| --- | --- |
|  | (4.6) |

通过扩展将包含进来，可以得到逻辑回归的非线性决策边界：

|  |  |
| --- | --- |
|  | (4.7) |

其中：表示权值向量；表示输入；表示偏执。

判别软件是否具有恶意性是一个二分类问题，恶意软件的分类可以表示成以为参数的二项分布：

|  |  |
| --- | --- |
|  | (4.8) |

而恶意软件预测模型是在特征向量与之间建立函数关系，可以表示成对的概率建模问题：

|  |  |
| --- | --- |
|  | (4.9) |
|  | (4.10) |

其中：表示输出为正常软件，表示输出为恶意软件。

对于给定条数据的训练集，可以应用极大似然估计模型参数，其对数似然函数为：

|  |  |
| --- | --- |
|  | (4.11) |

此时，问题变成了以对数似然函数为目标的最优问题，对求极大值，得到的估计值。

1. 多层感知机(MLP)神经网络

多层感知机神经网络是一种采用前向传播算法的网络，MLP中每层有多个神经元节点，相邻层节点之间互相关联。在MLP中基本结构包含输入层、隐藏层以及输出层。在输入层输入数据后与隐藏节点的权值以及偏置值进行线性组合，之后通过一个非线性函数也就是激活函数得到输出值。MLP属于有监督分类算法，结构如图4.3所示，其中，圆形代表每层的神经元节点，节点之间的连线为权重。样本数据输入神经网络之后，会作出如下几点操作：

①将输入数据、每层神经元节点的输出节点归一化处理。

②随机初始化参数。如权重w、学习率lr、以及迭代次数epoch。

③对各层节点的输出值通过激活函数非线性处理后得到输出值。

④将输出层节点的值和真实值进行比较，得到误差值E（q），如果误差小于设置的阈值，则得到最优参数模型；否则，利用反向传播思想，返回步骤（3）继续计算直到误差满足要求或者达到设置的迭代次数停止对神经网络的训练。



图4.3 多层感知器模型中神经元之间的连接

4.5.2集成学习

为了发挥不同的分类算法在处理不同数据上的优势，集成学习的思想应运而生。集成学习可以有效的提高系统学习的泛化能力，它本身不是一个单独的机器学习算法，而是通过构建并结合多个机器学习分类器来完成学习任务。集成学习的目的就是想要结合不同子模型的长处，已达到互补短处的目的。简单来说，如果子模型的错误不同且互补，那么正确融合多个子模型就可以得到一个更好的模型。一般来说，一个好的融合的子模型应该符合以下两个假设：(1)准确性：大部分子模型要能对最终输出有所帮助，而不是帮倒忙，应该至少比随机猜测更准确。(2)多样性：子模型间需要有所不同，长处和短处各异，(高度相关的)同质化的子模型无法互补。

从图4.4中，我们可以对集成学习的思想做一个总结：对于训练集数据，我们通过训练若干个子学习器，并且将训练得到的子学习器通过一定的结合策略，就可以最终形成一个强学习器。也就是说，集成学习主要有两个部分，一个是如何通过训练得到若干个子学习器，另一个是选择哪种种结合策略，可以将这些子学习器集合成一个强学习器。



图4.4 集成学习技术路线图

针对应用程序的不同特征在不同机器学习分类算法上的表现会有所差异，单一的机器学习分类算法不能有效的发挥应用程序不同特征在恶意应用程序检测上所起到的作用这一问题，本文提出了一种基于集成学习的多特征Android恶意应用程序检测方法。将提取的权限、API以及资源特征分别用不同的分类算法进行训练并将其结果作为子模型进行集成学习，选取每类特征的最优算法，采用加权投票选择的方式对最优算法中的分类结果进行特征融合。实验结果表明，该集成方法有效弥补了传统检测方法的不足，从而能更加高效地检测Android恶意应用程序。集成学习主要分为三大类：Boosting、Bagging、Stacking。下面就本文用到的集成学习技术进行介绍。

4.5.3 结合MLP神经网络的Stacking集成分类技术

Stacking是一种表示学习，表示学习指的是模型从原始数据中自动抽取有效特征的过程，比如深度学习就是一种表示学习的方法。原始数据可能是杂乱无规律的。在stacking中，通过第一层的多个学习器后，有效的特征被学习出来。从这个角度来看，stacking的第一层就是特征抽取的过程。stacking的学习能力主要来自于对于特征的表示学习，这和神经网络的思路是一致的。

stacking需要宽度，深度学习需要深度，结合MLP神经网络的Stacking集成分类技术可以有效的结合两者的优势。Stacking中的第一层可以等价于神经网络中的前n-1层，而stacking中的最终分类层可以类比于神经网络中最后的输出层。在本文中stacking的第一层使用的是MLP多层感知机神经网络，将前文中提取的应用程序的权限、API、资源特征使用MLP进行特征提取，分别输出一个256的特征向量，并将得到的特征向量最为stacking的第二层的输入。

因为第二层的特征来自于对于第一层数据的学习，那么第二层数据中的特征中不该包括原始特征，以降低过拟合的风险。为了降低过拟合的问题，第二层分类器应该是较为简单的分类器，广义线性如逻辑回归与SVM是一个不错的选择，本文选取SVM为次级学习器。在特征提取的过程中，我们已经使用了复杂的非线性变换，因此在输出层不需要复杂的分类器。这一点可以对比神经网络的激活函数或者输出层，都是很简单的函数，一点原因就是不需要复杂函数并能控制复杂度。因此，stacking的输出层不需要过分复杂的函数。图4.5是结合MLP神经网络的Stacking集成分类技术的技术路线图。



图4.5 结合MLP神经网络的Stacking集成分类技术的技术路线图

4.5.4 基于支持向量机的Bagging集成分类技术

Bagging算法的弱学习器之间没有依赖关系，可以并行生成。基于支持向量机的Bagging集成分类技术的基本流程：对第三章提取的三类特征，分别使用SVM进行训练，得到一个基分类器，再将这些基分类器进行结合。在对预测输出进行结合时，基于支持向量机的Bagging集成分类技术对分类结果采用简单投票法。图4.6是基于支持向量机的Bagging集成分类技术的技术路线图。



图4.6 基于支持向量机的Bagging集成分类技术的技术路线图

4.5.5改进Bagging集成分类技术

传统的Bagging算法选取相同的分类器，但在不同的训练集上进行训练。Bagging技术的重要核心是给出一个训练集和一个弱分类器，能够利用此弱分类器来对给定的训练样本进行归类。而在本文中考虑单个弱分类器算法准确率不高，而且不同机器学习算法在权限、API、资源特征上的分类效果不同，因此文中将提取的权限、API以及资源特征分别用上文所提到的检测模块进行训练，选取针对每类特征的最优算法，并将每类算法中的最优分类结果采用投票选择的方式进行特征融合[47]，简单投票算法由于只对结果进行投票选择，没有其他附加信息，因此这种方法不能体现性能高的分类器的优势。加权投票法是一种很直观的方法，其给分类性能高的分类器赋予一个高的权值。因此改进Bagging集成分类技术采用加权投票法对恶意应用进行判别。

如前所述，多分类器加权投票算法能够最大程度地纠正单个分类器的初始分类错误。首先，计算各分类器对分类结果的正确率，然后再把各分类器的正确率相加得到总正确率的值，将分类器正确率与总正确率相比，得到单个分类器的权值，公式如下(其中表示分类器的个数):

|  |  |
| --- | --- |
|  | (4.12) |

然后将计算得到的权值与各分类器对各个应用分类的概率相乘，得到，将各结果的相加，得到，然后将与0.5进行比较，得到集成分类结果，具体公式如下所示(其中1表示“正常应用”，0表示“恶意应用”):

|  |  |
| --- | --- |
|  | (4.13) |

基于改进Bagging集成分类技术具体技术路线图如图4.7所示。



图4.7 基于改进Bagging集成分类技术的技术路线图

## 4.6 本章小结

本章详细介绍了基于集成学习多特征的Android恶意应用程序检测模型。分别详细阐述了基于各类特征的检测模块的构建。最后对集成学习算法进行了简单介绍，并结合本文数据提出了结合MLP神经网络的Stacking集成分类技术、基于支持向量机的Bagging集成学习分类技术与改进Bagging集成学习分类技术。通过多个检测模型的分析比较，生成了本课题的基于集成学习的多特征Android恶意应用程序检测模型。

5 实验设计与结果分析

## 5.1 测试环境

本文实验是在个人PC上进行的，处理器为Inter(R)Core(TM)i5-4210U，操作系统为Windows10(64位)，内存为4GB，硬盘500GB，使用的程序语言为JAVA与Python，使用的软件为Eclipse与PyCharm。

## 5.2 实验样本介绍

在本文中数据集包括正常应用数据集与恶意应用数据集。本文的恶意应用数据集来自Drebin，该数据集一共有5510个，覆盖了177个恶意家族，其中有的家族最多有925个恶意应用，而最少的家族仅仅只有1个恶意应用。这些恶意家族几乎涵盖了2.4节中介绍的所有恶意应用的分类。本文恶意应用数据集随机从中抽取了891个恶意数据。恶意应用数据集详细构成如表5.1所示：

**表5.1 恶意样本数据集**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 恶意代码种类  (大类) | 子类 | 数量 | 百分比 | 5% | 10% | 15% | 20% | 25% | 30% |
| 木马类(Trojan) | FakeInstaller | 148 | 16.63% |  | | | | | |
| Opfake | 98 | 11.02% |
| Iconosys | 24 | 2.73% |
| FakeDoc | 21 | 2.37% |
| 后门类(Backdoor) | Kmin | 24 | 2.64% |
| Basebridge | 53 | 5.93% |
| DroidKungFu | 107 | 11.99% |
| GinMaster | 54 | 6.10% |
| DroidDream | 13 | 1.46% |
| 僵尸类(Zombie) | Plankton | 100 | 11.24% |
| Geinimi | 15 | 1.65% |
| Adrd | 15 | 1.64% |
| 其他 |  | 219 | 24.62% |

除了Android恶意应用程序外，为了验证本文方法的有效性，在实验中还需要加入正常应用。本次实验从GooglePlay官网上下载668个正常APK，共24个种类，其中有工具软件，社交软件，旅游软件以及游戏软件等。然后经各种反病毒软件检测，确保不存在恶意软件。正常应用程序的具体类别和每个类别中包含的样本数量如表5.2所示。

**表5.2 Android正常应用程序数据集列**

|  |  |  |  |
| --- | --- | --- | --- |
| 良性应用程序类别 | 包含的样本数量 | 良性应用程序类别 | 包含的样本数量 |
| 财务(finance) | 6 | 社交(social) | 12 |
| 动漫(comic) | 12 | 摄影(photograph) | 54 |
| 个性化(personalization) | 12 | 生活时尚(lifestyle) | 12 |
| 工具(tools) | 48 | 体育(sport) | 12 |
| 购物(shopping) | 12 | 通讯(communication) | 44 |
| 健身与健康(health&fitness) | 32 | 图书与工具书(book) | 12 |
| 交通运输(transportation) | 12 | 小部件(wallpaper) | 50 |
| 教育(education) | 50 | 效率(productivity) | 40 |
| 旅与本地出行(travel&local) | 46 | 闻杂志(news&magazines) | 46 |
| 媒体与视频(media&video) | 42 | 医务(medical) | 12 |
| 软件与演示(libraries&demo) | 46 | 乐与音频(music&audio) | 32 |
| 商务(business) | 12 | 游戏(game) | 12 |

## 5.3 实验设计

针对本课题中提出的基于多特征的Android恶意代码的检测模型，进行了多组实验来验证本文提出的检测模型的性能，按照上一章介绍的恶意代码检测模型，本实验将有七种机器学习分类器进行学习：

(1)基于权限分类器PC。

(2)基于API分类器AC。

(3)基于资源的分类器RC。

(4)基于多特征的分类器MC。

(5)基于MLP神经网络的Stacking集成学习分类器Sta-MLP。

(6)基于支持向量机的Bagging集成学习分类器Bag-SVM。

(7)基于改进的Bagging集成学习分类器Bag-IMP。

针对以上七种分类模型，本文最终选择其中表现最好的分类将应用到恶意应用的检测中。理论上，集成后的分类模型会更加精确，性能更加优越；改进集成分类技术可以充分发挥各个分类模型的优势，可以避免简单投票带来的弊端。基于MLP神经网络的Stacking集成学习分类器可以结合深层模型与浅层模型的优势，该模型有较高的分类准确率，并且能解决过拟合的问题。

5.3.1实验数据集的划分

在本课题中正常样本的分类有24类，恶意样本几乎覆盖了所有的恶意家族，因此选择数据集的时候要尽可能的均匀的在每一类中进行选择，构成训练数据集和测试数据集。有效的划分数据集，可以使得训练出来的结果更具代表性。本实验中训练数据与测试数据的抽按照9:1的比例分布，表5.3给出了本实验数据的具体划分情况。

**表5.3 实验训练集与测试集划分**

|  |  |
| --- | --- |
| 样本分类 | 数据集样本数量 |
| 训练数据集 | 1403 |
| 测试数据集 | 156 |

5.3.2恶意代码检测分类评价方法

在本实验中，将Android正常应用定义为正样本，将恶意应用定义为负样本。对于分类结果本文采用混淆矩阵来计算Accuracy，Precision，FNR等评价指标。其中表示正常应用被正确分类为正常应用的数量。表示正常应用被误报是恶意应用的数量。表示恶意应用被正确检测为恶意应用的数量。表示恶意应用被误认为正常应用的数量。对应评价指标的计算公式如下所示：

|  |  |
| --- | --- |
|  | (5.1) |
|  | (5.2) |
|  | (5.3) |
|  | (5.4) |

## 5.4 实验结果与分析

本节包含了对不同特征的分类性能的验证实验，具体包括以下几部分实验：

(1)基于单一特征的检测实验，包括基于权限特征的检测结果、基于API特征的检测结果与基于资源特征的检测结果。

(2)基于多特征检测模型的分类实验，包括基于多特征的检测结果、基于MLP神经网络的Stacking集成学习分类的检测结果、支持向量机的Bagging集成学习分类器的检测结果与基于改进的Bagging集成学习分类器的检测结果。

5.4.1基于单一特征的检测模块

(1)权限特征和分类算法的选择实验

基于权限特征的检测模块的建立，首先需要对获取的权限特征进行选择，在第三章中分析了恶意应用常用的权限以及权限组合，第四章也对权限特征的信息增益值进行了排序，我们按照排序结果从低到高的顺序每5个一组进行分类，分别使用朴素贝叶斯，支持向量机，逻辑回归，K近邻，随机森林进行实验。图5.1，5.2，5.3对权限特征在不同特征维度与不同分类算法的准确率，漏报率与误报率表现进行了详细地统计。



图5.1 权限特征在不同特征维度与不同分类算法的准确率

图5.2 权限特征在不同特征维度与不同分类算法的漏报率



图5.3 权限特征在不同特征维度与不同分类算法的误报率

从上面三副图中可以看出，当权限特征维数在40维左右的时候(图中的虚线处)，对于不同分类算法的各项指标都趋于稳定且有一个好的表现。因此，我们选择信息增益值top40的权限特征来建立基于权限特征的检测模块。对于分类算法的选择，从图5.1中可以看出，随机森林算法在各种特征维度上的准确率都比其它分类算法的效果好。从图5.2中可以看出，虽然朴素贝叶斯的漏报率是最低的，但别的评价指标表现的却很差。从图5.3中可以看出随机森林在误报率上的表现也是最低的。因此本文选择信息增益值最大的前40维权限特征与随机森林机器学习算法建立基于权限特征的检测模块。

(2)API特征和分类算法的选择实验

基于API特征的检测模块的建立，与基于权限特征的检测模块实验设计相同。图5.4，5.5，5.6对API特征在不同特征维度与不同分类算法的准确率，漏报率与误报率表现进行了详细地统计。



图5.4 API特征在不同特征维度与不同分类算法的准确率



图5.5 API特征在不同特征维度与不同分类算法的漏报率



图5.6 API特征在不同特征维度与不同分类算法的误报率

从这三幅图中可以看出，当权限特征维数在25维左右的时候(图中的虚线处)，对于不同分类算法的准确率最高，漏报率与误报率最低。所以我们选择信息增益值前25维的权限特征来建立基于API特征的检测模块。对于机器学习分类算法的选择，从图5.4中可以看出，支持向量机与随机森林算法在各种特征维度上的准确率都比其它分类算法的效果好。从图5.5中可以看出，虽然KNN算法的漏报率最低，但同样的KNN的准确率也低。从图5.6中可以看出支持向量机要比随机森林算法的误报率更低，由于误报率指标在评价恶意代码检测模型性能时更具有代表性，因此本文选择信息增益值最大的前25维API特征与支持向量机机器学习算法建立基于API特征的检测模块。

(3)资源特征分类算法的选择实验

基于资源特征的检测模块的建立，首先需要对获取的资源特征进行选择，对第三章提取的图片、组件以及界面元素等特征，分别使用朴素贝叶斯，支持向量机，逻辑回归，K近邻，随机森林进行实验。图5.7-5.10对资源特征在不同分类算法的准确率、精确度、漏报率与误报率表现进行了详细地统计。

|  |  |
| --- | --- |
|  |  |
| 图5.7 资源特征在不同分类算法的准确率 | 图5.8 资源特征在不同分类算法精确度 |
|  |  |
| 图5.9 资源特征在不同分类算法的漏报率 | 图5.10 资源特征在不同分类算法的误报率 |

从图5.7-5.10中可以看出，当使用随机森林作为资源特征的分类器时，其准确率最好，达到了97.44%，漏报率与误报率也是最小的，分别达到了3.51%与2.03%，精确度虽然不是最高的，但是其精确度也没有太差，达到了96.49%，虽然逻辑回归的精确度最高，但是其别的指标却比随机森林差太多。因此本文选择随机森林机器学习算法建立基于资源特征的检测模块。

5.4.2 基于多特征的检测模型的分类实验

在这一小节中将本文实验设计中提到的七种实验模型在各项指标上进行了对比。各个模型的实验性能指标如表5.4所示，表5.4首先列举了基于权限特征模块、基于API特征模块、基于资源模块与基于多特征的最优检测结果，表格的后三行是本文提出的基于集成学习的多特征融合的实验结果。本实验从准确率，精确度，漏检率和误报率四个评价指标对本文所提出的七种检测模型在本文的实验数据集上做了评估。从准确率上来看，各个模型的准确率也都有不错的表现，其中准确率最高可以达到98.72%。从精确度来看，本文设计的七种实验检测模型均能够对Android恶意应用有不错的检测精确度，最高的精确度达到了98.25%。从漏检率来看，本文提出的七种检测模型的漏检率也都比较低，最低的能达到1.75%，说明本文提出的方法可以有极大的避免将正常应用程序识别为恶意应用的可能性。从误报率来看，各个模型的误报率都比较低，大部分在2%左右，最好的可以达到1%，说明本文提出的方法将恶意应用判别成正常应用的概率几乎不存在，可以有效避免恶意应用的威胁。

实验结果表明，(1)基于多特征的检测模块相比较单纯使用权限、API或资源文件特征进行判断，各项检测结果都有明显的提高，而且性能还比较稳定，没有出现漏报率与误报率相差较大的现象。(2)基于改进的Bagging多特征融合模型在精确度，准确率、召回率等检测指标都有了明显提高，其中准确率为98.72%，精确度为98.25%，漏报率为1.75%，误报率为1.01%，相较于MC与Bag-SVM算法，Bag-IMP算法的漏检率有一个很大的提升，说明本文提出的Bag-IMP算法的有效性。

**表5.4 基于多特征模型的分类实验结果**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模型 | 精确度 | 准确率 | 漏检率 | 误报率 |
| PC | 96.30% | 95.51% | 8.77% | 2.00% |
| AC | 98.18% | 97.74% | 5.26% | 1.00% |
| RC | 96.49% | 97.44% | 3.51% | 2.03% |
| MC | 98.25% | 98.07% | 3.59% | 1.01% |
| Sta-MLP | 96.55% | 97.44% | 2.88% | 1.97% |
| Bag-SVM | 96.43% | 96.79% | 5.26% | 2.02% |
| Bag-IMP | 98.25% | 98.72% | 1.75% | 1.01% |

5.4.3本课题的检测模型与以往方法的对比实验

为了验证本文提出的基于集成学习的多特征恶意应用程序的检测方法的有效性，将本文提出的Bag-IMP与其他工作进行对比，对比结果如表5.5所示。本文使用的数据集从Drebin获取，与文献[17]和文献[20]使用的数据集相同，因此将本文实验结果与其做比较会有说服力；文献[21]DroidMat的数据集是从contagio网站获取的，Wu等人[24]和杨欢等人[23]从VirusShare网站获取的恶意样本集，由于本文数据集与其数据集不同，所以本文将DroidMat、Wu以及Wei等提到的恶意检测方法的实验结果作为参考。从下表中可以看出Bag-IMP综合了各个方法所提取的特征，使用的机器学习算法也较为全面，本文提出的方法在各项性能上呈现了更好的效果。

**表5.5 不同方法的检测性能参照对比**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 方法 | 选取特征 | 使用的方法 | 正常数据 | 恶意数据 | 精确度 | 误报率 |
| Bag-IMP | 权限、API、组件、界面元素 | NB、KNN、SVM、RF | 668 | 891 | 98.25% | 1.75% |
| Drebin[17] | 权限、API、系统调用、Intent | SVM | 123,453 | 5,560 | 94% | 6.5% |
| Wu[24] | 权限、API、组件 | K-mean/EM、LR、NB | 1,160 | 1,050 | 97.7% | 2.4% |
| DroidMat[21] | 权限、API、Intent | K-mean/EM、kNN/NB | 1,500 | 238 | 96.7% | 12.6% |
| Kang[20] | 签名、权限、API、恶意命令 | NB | 51,179 | 4,554 | 91.5% | 0.2% |
| 杨欢[23] | 组件、系统调用、函数调用 | 三层混合系综算法 | 2,000 | 1,126 | 94.24% | 6% |

从以上对比结果可以看出，本文提出的基于集成学习的多特征恶意应用程序的检测方法比其他模型的检测精确度都要好，其中最多的精确度可以提升6.75%，而漏报率虽然不是最低的，但是效果已经非常好了。本模型能够充分利用多特征对Android恶意代码检测所起的作用，能够充分使用不同机器学习算法对某类特征的最优效果。结果验证了本文所提出的检测方法的有效性，该方法能够更加精准的识别恶意应用，更全面地表征应用的特性。

## 5.5 本章小结

作为论文的实验部分，本章对本文提出的基于集成学习的多特征的恶意应用程序的检测模型做了充分的分析与验证。首先本章对本文的实验环境以及实验数据进行了详细的介绍，接着对本文提出的单一特征的检测模块的分类算法以及特征维数的选择做了详细的实验，最后对本文提出的多特征的检测结果做了详细对比。从实验的结果中可以看出，本文所提出的检测方法的高效性，并且与相关检测方法相比，本文检测方法呈现了一个好的性能。

6 总结与展望

## 6.1 本文工作总结

由于智能移动终端的普及，手机已经成为了我们生活中必不可少的部分，人们可以在手机上进行交流，娱乐，学习等，极大的方便了我们的生活。相较于其他操作系统而言，Android系统的免费性、开发的易学性与开放性等特点，使得Android系统迅速积累了大量人气，从而极大地促进了Android系统的发展。虽然Android系统具有诸多优势，但是由于其开放性和自身安全机制存在的一些漏洞，会给一些恶意应用可乘之机，给用户的信息和财产安全造成极大的威胁。因此，开发快速、高效的Android恶意应用检测工具势在必行。本文针对恶意应用程序的检测，做了以下研究：

本文首先对系统进行了简单的介绍，如Android系统框架、APK软件结构与Android系统四大组件等，然后对Android安全机制进行介绍，还对恶意代码载荷技术、事件监听技术以及恶意代码数据发送技术等恶意代码常用技术做了详细的介绍，最后对Android恶意软件的分类进行了描述。

其次，本文在对Android系统的权限特征、系统API特征以及资源特征等三类特征进行提取之后，重点剖析了基于权限级别的恶意代码行为，基于API级别的恶意代码行为和资源特征对Android恶意应用检测的意义，并分别介绍了三种特征的提取方法。在基于权限级别的恶意代码行为中，Android恶意应用经常会定义某些具有危险性的权限来进行恶意操作；在基于API级别的恶意代码行为中，本文发现恶意应用有很多具有共性的行为特征；通过对Android应用程序的图片、界面元素以及组件等这些资源特征进行分析，结果表明这些特征在正常应用与恶意应用中有明显的差异。

然后对本文提出的恶意应用程序的多特征检测模型做了详细介绍。首先分对各类特征的检测模块的构建做了具体阐明。接着对集成学习算法进行了简单介绍，并结合本文数据提出了改进Bagging集成学习分类技术、结合MLP神经网络的Stacking集成分类技术。最后通过多个检测模型的融合生成了本课题的基于集成学习多特征的Android恶意应用程序检测模型。

最后对本文提出的恶意应用程序的检测模型做了充分的分析与验证。针对Android恶意应用检测，本文提出了多种新的检测模型，并且分别设计了多组实验，通过实验对每一类特征的表征能力进行分析。而且将本文提出的检测模型与已有检测方法进行实验对比分析，通过实验对比的结果显示，本文提出的检测模型更加有效，效果更佳。

## 6.2 未来工作展望

本文提出了基于集成学习多特征的Android恶意应用程序新的检测方法，并且实验结果达到预期效果。本文研究并提出的这种新方案比目前已有的相关方案虽然具有一定的进步意义，然而，本文研究的方案尚不完善，也存在需要进一步改进的地方，在接下来的工作中，拟打算从下面几个方面着重开展未来的研究工作。

(1)针对恶意应用的检测，传统技术侧重于对单一特征进行检测分析，没有充分利用多特征对Android恶意代码进行检测所起的作用，没有充分采用各种机器学习算法对某类特征的检测建立的最优模型。本文虽然解决了这个问题，但是本文的特征选择也只选择了三类特征，而且在集成学习过程选用的分类模型还不够全面。因此，挖取更重要的特征，以及将不同特征在不同分类模型的最优表现结果进行融合会更有意义。

(2)本文对恶意应用的检测模型是针对的是所有恶意家族的恶意应用，没有具体到是那一类型的恶意应用，因此在后期工作中如果我们可以判别出此应用是恶意应用并且分析出此应用具有那种恶意行为，这也是非常有意义的。

(3)恶意应用程序检测方法最终目的是为用户找到安装或即将安装的应用是否是恶意性的。作为用户，当然希望这个检测方法尽可能操作简便，占内存小，运行时间短，正确率高。因此，研究更加高效简洁的安卓恶意应用检测方法仍需要学者不懈的创新努力。

致 谢

不知不觉已经写到致谢。回想起初次成为研究生的欣喜，到如今，我的研究生生涯已经快要结束了。而我仿佛还停留在研一日复一日的上课学习中，停留在研二每天在实验室看论文，做实验中，停留在小论文和大论文的撰写时。在即将离开校园生活之前，借此深切感谢一直指导我的老师，陪伴我的舍友们和鼓励我的小伙伴，谢谢你们一直在我身边。

首先，我衷心的感谢敬爱的老师\*\*\*老师，感谢老师对我的指导，栽培，给我认识Android软件安全世界的机会。在研究生三年时间里，\*老师对科研的认真和对学生的关心与教诲深深地影响了我。由于我是跨专业入学，刚开始专业知识可能会比较欠缺，是\*老师悉心的教导，严格的要求，给了我非常多的专业性的建议，让我终生受益。\*老师为人热情随和，治学严谨细心，无论是在学习还是在生活上都如春分细雨般鼓励着我。感谢\*老师在日常琐碎的生活中教会我踏实认真的道理。感谢\*\*\*严格的要求我，使我可以全面地了解到Android静态分析和动态监测中常见的技术和工具，拓宽了眼界，增长了见识。感谢\*老师让我明白，学业是一个人的硬实力，个人素养是一个人的软实力。老师，谢谢您。

其次，非常感谢\*\*师姐、\*\*师妹、\*\*\*师妹和402实验室的\*\*\*、\*\*\*、\*\*\*等同学。感谢大家在Android逆向学习和安卓恶意应用程序分析方面的探讨学习。感谢师姐师妹们对我的支持和鼓励，因为有她们，我想成为更好地自己，想要不断的学习进步。感谢实验室的同学，让我有一个良好的学习氛围，感谢他们的持之以恒，让我逐渐进步。感谢大家的互相帮助和互相鼓励，让过去的每天有所收获，未来的每一天有所期待，今天有所坚持。感谢西安科技大学计算机学院2016级全班同学陪我度过这段有苦有甜有意义的时光。

感谢我的女朋友，感谢她总是站在我身边，总是默默爱护我，给予我最大的关怀及感动。感谢她无条件的理解和不问理由的支持。谢谢你一直在我身边。

最后，感谢各位审阅本论文的老师。感谢出席本次论文答辩的老师。感谢你们对我的指导。

参考文献

|  |  |
| --- | --- |
| [1] | 2019-2025年中国智能手机操作系统行业市场需求预测及投资未来发展趋势报告[R].中国报告大厅，2018. |
| [2] | 2018年Q3季度互联网安全报告[R].腾讯安全反病毒实验室发布，2018. |
| [3] | Song J, Han C, Wang K, et al. An integrated static detection and analysis framework for android[J]. Pervasive and Mobile Computing, 2016, 32: 15-25. |
| [4] | 高玉新, 张怡, 唐勇, et al. 恶意代码反分析与分析综述[J]. 小型微型计算机系统, 2015, 36(10):2322-2326. |
| [5] | 王菲飞. 基于Android平台的手机恶意代码检测与防护技术研究[D]. 北京交通大学, 2012. |
| [6] | 张家旺, 李燕伟. 基于机器学习算法的 Android 恶意程序检测系统[J]. 计算机应用研究, 2017, 6: 1-6. |
| [7] | Android系统恶意代码检测技术研究[D]. 哈尔滨工业大学, 2014. |
| [8] | Dong H , Liu Y , Li C Z , et al. Semantic-Based Sensitive Behavior Analysis Method for Android[J]. Journal of University of Electronic Science & Technology of China, 2017, 46(2):434-440. |
| [9] | Sun M , Li X , Lui J C S , et al. Monet: A User-oriented Behavior-based Malware Variants Detection System for Android[J]. IEEE Transactions on Information Forensics & Security, 2016, PP(99):1-1. |
| [10] | Meng G, Xue Y, Xu Z, et al. Semantic modelling of android malware for effective malware comprehension, detection, and classification[C]//Proceedings of the 25th International Symposium on Software Testing and Analysis. ACM, 2016: 306-317. |
| [11] | 王蕊, 冯登国, 杨轶, et al. 基于语义的恶意代码行为特征提取及检测方法[J]. 软件学报, 2012, 23(2):378-393. |
| [12] | 王欢, 来欢, 李国栋,等. 采用函数调用关系的注入型Android恶意应用检测[J]. 西安交通大学学报, 2015, 49(10):84-89. |
| [13] | Zhou W, Zhou Y, Jiang X, et al. Detecting repackaged smartphone applications in third-party android marketplaces[C]//Proceedings of the second ACM conference on Data and Application Security and Privacy. ACM, 2012: 317-326. |
| [14] | Masud M M, Al-Khateeb T M, Hamle K W, et al. Cloud-based malware detection for evolving data streams.[J]. Acm Transactions on Management Information Systems, 2015, 2(3):16. |
| [15] | Rastogi S, Bhushan K, Gupta B B. Android Applications Repackaging Detection Techniques for Smartphone Devices [J]. Procedia Computer Science, 2016, 78:26-32. |
| [16] | Zhang B, Li Q, Ma Y. Research on dynamic heuristic scanning technique and the application of the malicious code detection model[J]. Information Processing Letters, 2017, 117: 19-24. |
| [17] | Arp D, Spreitzenbarth M, Hubner M, et al. Drebin: Effective and explainable detection of android malware in your pocket[C]//Ndss. 2014, 14: 23-26. |
| [18] | Ma Z , Ge H , Liu Y , et al. A Combination Method for Android Malware Detection Based on Control Flow Graphs and Machine Learning Algorithms[J]. IEEE Access, 2019, PP(99):1-1. |
| [19] | Enck W, Ongtang M, McDaniel P. On lightweight mobile phone application certification. In: Proc. of the 16th ACM Conf. On Computer and Communications Security (CCS 2009). 2009. 235-245. |
| [20] | Kang H , Jang J W , Mohaisen A , et al. Detecting and Classifying Android Malware Using Static Analysis along with Creator Information[J]. International Journal of Distributed Sensor Networks, 2015, 2015:1-9. |
| [21] | Wu D J, Mao C H, Wei T E, et al. Droidmat: Android malware detection through manifest and api calls tracing[C]//2012 Seventh Asia Joint Conference on Information Security. IEEE, 2012: 62-69. |
| [22] | Du Y, Wang J, Li Q. An android malware detection approach using community structures of weighted function call graphs[J]. IEEE Access, 2017, 5: 17478-17486. |
| [23] | 杨欢, 张玉清, 胡予濮, 等. 基于多类特征的 Android 应用恶意行为检测系统[J]. 计算机学报, 2014, 37(1): 15-27. |
| [24] | Wu S, Wang P, Li X, et al. Effective detection of android malware based on the usage of data flow APIs and machine learning[J]. Information and software technology, 2016, 75: 17-25. |
| [25] | ChkBugReport[DB/OL]. https://github.com/sonyxperiadev/ChkBugReport/wiki. |
| [26] | |  | | --- | | 张笑地.[Android平台动态恶意行为检测系统的设计与实现](http://www.wanfangdata.com.cn/details/javascript:void(0);)[D].电子科技大学,2017. | |
| [27] | Burguera I, Zurutuza U, Nadjm-Tehrani S. Crowdroid: behavior-based malware detection system for android[C]//Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. ACM, 2011: 15-26. |
| [28] | Vidal J M, Monge M A S, Villalba L J G. A novel pattern recognition system for detecting Android malware by analyzing suspicious boot sequences[J]. Knowledge-Based Systems, 2018, 150: 198-217. |
| [29] | 于洲. Android平台恶意软件的动态检测技术研究[D].电子科技大学,2015. |
| [30] | float. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones[J]. Acm Transactions on Computer Systems, 2014, 32(2):1-29. |
| [31] | Sun M, Wei T, Lui J. Taintart: A practical multi-level information-flow tracking system for android runtime[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016: 331-342. |
| [32] | Schultz M G, Eskin E, Zadok F, et al. Data mining methods for detection of new malicious executables[C]//Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001. IEEE, 2001: 38-49. |
| [33] | Yang C, Xu Z, Gu G, et al. Droidminer: Automated mining and characterization of fine-grained malicious behaviors in android applications[C]//European symposium on research in computer security. Springer, Cham, 2014: 163-182. |
| [34] | Shaw S, Gupta M K, Chakraborty S. Cloud Based Malware Detection Technique[C]//Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications. Springer, Singapore, 2017: 485-495. |
| [35] | Gupta S , Kumar P . An Immediate System Call Sequence Based Approach for Detecting Malicious Program Executions in Cloud Environment.[J]. Wireless Personal Communications, 2015, 81(1):1-21. |
| [36] | 陈伟, 杨秋辉, 程雪梅. 基于安全域的Android系统内核安全增强机制研究[J]. 计算机科学, 2018, 45(2):203-208. |
| [37] | 吴恋, 马敏耀, 黄一峰, et al. 基于AdaBoost算法的Linux病毒检测研究[J]. 计算机工程, 2018, 491(08):167-172+179. |
| [38] | Android平台恶意短信行为的预防和检测技术研究与实现[D]. 北京邮电大学, 2015. |
| [39] | 卿斯汉. Android安全研究进展[J]. 软件学报, 2016, 27(1):45-71. |
| [40] | 黄梅根, 曾云科. 基于权限组合的Android窃取隐私恶意应用检测方法[J]. 计算机应用与软件, 2016, 33(9):320-323. |
| [41] | 一种基于集成学习的Android用户隐私保护方案的研究与实现[D]. 电子科技大学, 2015. |
| [42] | Schapire R E. The Strength of Weak Learnability[J]. 1990, 5(2):197-227. |
| [43] | Wang W , Wang X , Feng D , et al. Exploring Permission-Induced Risk in Android Applications for Malicious Application Detection[J]. IEEE Transactions on Information Forensics and Security, 2014, 9(11):1869-1882. |
| [44] | Moore A W , Zuev D . Internet traffic classification using bayesian analysis techniques[J]. ACM SIGMETRICS Performance Evaluation Review, 2005, 33(1):50. |
| [45] | Cortes C, Vapnik V. Support-Vector Networks[J]. 1995, 20(3):273-297. |
| [46] | Dong G , Taslimitehrani V . Pattern-Aided Regression Modeling and Prediction Model Analysis[J]. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(9):2452-2465. |
| [47] | 周志华. 机器学习: Machine learning[M]. 清华大学出版社, 2016:185-19 |

附 录

硕士期间发表的论文:

|  |  |
| --- | --- |
| [1] | \*\*\*, \*\*\*, \*\*. Android Malware Detection Based on Multi-Features[C]// 2018 the 8th International Conference on Communication and Network Security , 2018: 69-73. |

硕士期间参与的项目：

[1] 《Android应用程序恶意隐私信息泄露行为的检测方法研究》