基于改进人工蜂群算法的 Android 恶意应用检测

徐开勇 肖警续 郭 松 戴乐育 段佳良
(网络空间安全教研室(信息工程大学) 郑州 450001)

摘 要 随着互联网和移动终端的飞速发展,手机中存储着很多重要的信息,要保证这些信息安全不被泄露的一个重要方法就是对手机中的恶意应用进行检测与处理。在对恶意应用进行检测前需要对样本进行特征提取,而如何在众多特征中进行有效的选取是恶意应用检测中一个至关重要的过程。文中针对 Android 平台的应用,参考相关的 Android 恶意检测方法,建立了一个基于改进人工蜂群算法的 Android 恶意应用检测模型,通过对特征进行有效的选择,最终得到使分类结果最优的特征组合,从而提高对 Android 恶意应用检测的检测性能。在静态和动态条件下分别对 Android 应用特征进行提取,通过多种分类算法对恶意应用检测模型进行检验,结果证实提出的基于改进人工蜂群算法的 Android 恶意应用检测方法具有可行性与优越性。

关键词 恶意应用检测,人工蜂群,特征选取,特征优化
中图法分类号 TP311 **文献标识码** A

Android Malicious Application Detection Based on Improved Artificial Bee Colony Algorithms

XU Kai-yong XIAO Jing-xu GUO Song DAI Le-yu DUAN Jia-liang
(Country Network Space Security Teaching and Research Room (Information Engineering University), Zhengzhou 450001, China)

Abstract With the rapid development of the Internet and mobile terminals, there are a lot of important information stored in mobile phones. An important way to ensure that these information is not compromised is to detect and process malicious applications in mobile phones. Before detecting malicious applications, feature extraction is required for samples, and how to effectively select features among many features is a crucial process in malicious application detection. Based on the application of Android platform, this paper established an Android malicious application detection model based on the improved artificial bee colony algorithm. By effectively selecting the features, the feature combination that optimizes the classification results is finally obtained, thereby improving the detection performance of Android malicious application detection. The Android application features are extracted under static and dynamic conditions respectively. The malicious application detection model is tested by various classification algorithms. It is proved that the proposed malicious malicious detection method based on the improved artificial bee colony algorithm has the feasibility and superiority.

Keywords Malicious application detection, Artificial bee colony classification, Feature selection, Feature optimization

1 引言

近年来,随着网络的快速发展,智能手机的普及率也飞速增加,通过智能手机中的各式各样的应用程序人们几乎可以完成每日生活的全部需求,因而手机中的应用程序的安全性引发了广泛关注。针对于智能手机平台,安卓(Android),因为其操作系统的开源性,自发布以来逐步占据了市场的主导地位。截至 2019 年 3 月,根据 statcount[1] 官方显示,Android 的市场份额占比高达 75.33%,第二名的 IOS 只占有 22.4%。由此可见,对于安卓恶意应用的检测与研究是十分必要的。

由于安卓的开源性,用户可以从 Google Play[2] 和第三方 Android 市场(如安卓市场、手机管家、Amazon 等)下载所需的应用程序,应用程序的开发者也可以上传应用程序到第三方市场供用户下载,恶意应用程序因此变得十分活跃。据 360 安全大脑发布的《2018 年 Android 恶意软件专题报告》[3],2018 年全年,360 安全大脑共截获移动端新增恶意软件样本约 434.2 万个,累计监测移动端恶意软件感染量约为 1.1 亿人次,虽同比均呈现出下降趋势,但由于其高基数,显示了移动恶意软件总体进入平稳高发期。全球各大手机杀毒平台虽在不断地完善对于恶意应用检测的准确率,但因为恶意应用的多样性,更迭迅速,仍亟需一种更完善的恶意应用检测方法。

本文对于 Android 应用进行动静态结合的特征提取,静态特征提取权限特征和敏感 api 特征,动态特征通过 DroidBox[10] 与 ApiMonitor[11] 工具,在应用程序在虚拟机中运行时提取相应特征。为证明所提方法的合理性,本文建立一个 Android 恶意应用检测模型并通过选取多种分类算法对恶意应用检测模型进行验证,最终得出本文方法具有较高的恶意

本文高安全等级移动终端项目资助。
徐开勇(1963—),男,硕士,研究员,主要研究方向为信息安全、可信计算,E-mail:345371975@qq.com;肖警续(1994—),男,硕士,主要研究方向为 Android 恶意应用检测,E-mail:345371975@qq.com(通信作者)。

应用检测准确率，证实了本文所提方法的合理性和优越性。

2 相关工作

恶意应用的检测手段在不断发展与进步，检测方法主要分为基于签名的检测方法和基于行为的检测方法。其中传统的基于签名的技术[4-5]已经被普遍使用在各个平台，但该技术只能对特定已记录类别的恶意应用进行检测，对于那些不属于签名库中的未知恶意应用不能进行有效的检测。基于行为的检测方法分为静态检测和动态检测。静态检测的优势在于：在代码层面上对恶意应用进行检测，通过反编译等技术手段解读代码的行为从而对恶意应用进行判别，具有轻量级代码覆盖率高的优点。其缺点是对于一些代码经过特殊处理的恶意应用（如对于代码加密，应用运行后才能解码等）不能进行有效的检测。文献[6-7]分别将权限作为特征对恶意应用进行静态检测，但因为选取的特征单一且代码可能存在绕过代码申请权限的问题，检测的准确率并不理想。基于动态的恶意应用检测方法是在应用程序在独立的模拟器（如沙箱）中运行时对恶意应用特征进行识别与检测，能克服静态检测中遇到的代码混淆和加密等问题，但其缺点是代码覆盖率低，处理大量恶意应用非常耗时。文献[8]提出的 mad4a 方法在静态过程中选取权限作为特征，在动态检测中使用 api 作为特征对安卓恶意应用进行检测，并提出了过度权限的观点以进一步区分恶意应用和正常应用，但该方法并没有对实验结果进行定量分析以及比较。文献[9]提出的 Androdest 方法通过将组件、系统调用、函数调用等作为特征，并将特征分为三层通过混合系统算法对恶意应用进行检测，其成果较单一的静态检测或动态检测有一定提高，但其在静态检测部分仅以申请权限以及使用组件的数量作为特征，忽略了权限之间的差异，并且在使用的分类算法上也可以进行相应的升级，引入近年来兴起的深度学习算法可能会使准确率提升。

大部分 Android 恶意应用检测模型虽然提取了很多不同维度的特征，但都缺少对特征进行有效选取的过程，如 2016 年 Yuan 等提出的 Droiddetector[18] 直接将提取到的全部 192 项权限特征、api 特征和动态特征作为检测依据，并结合深度学习算法进行验证，检测精度达到 96.76%；同年 Saracino 等提出的 MADAM[19] 对 Android 应用进行多个层次的特征提取，包含内核层、应用层、用户和包等 4 个不同级别，并获得对 Android 恶意应用 96%的检测成功率。上述 Android 恶意应用检测方法通过覆盖率较高的特征均得到较满意的检测准确率，但将提取到的所有原始特征作为检测 Android 恶意应用的依据，不但会增加检测过程的计算复杂度，也不能使检测模型达到最优的检测准确率，其检测方法仍有提升的空间。所以，在 Android 恶意应用检测方面对特征进行有效的选取是提升检测性能的重要方法。

针对上述文献提出的方法，可以得出在恶意应用检测过程中特征的选取是一个十分重要的阶段，通过选取有效的特征能够很大程度上提高检测模型的准确性，为解决如何在众多特征中选取出对检测模型最优的特征，本文提出了一种改进的人工蜂群算法对特征进行选取，通过将训练样本的检测准确率作为蜂群的目标进行迭代，从而得出对于检测模型最优的特征组合，以提升对 Android 恶意应用的检测性能。

3 基于改进人工蜂群算法的 Android 恶意应用检测模型

本文构建了一个基于改进人工蜂群算法的 Android 恶意应用的检测模型，模型主要由特征提取、特征优化和应用分类 3 个模块构成。本恶意应用检测系统框图如图 1 所示。

（1）特征提取模块模型在静态特征方面提取权限特征和敏感 api 特征，在动态特征方面利用 DroidBox 和 Apimonitor 工具，对运行在虚拟机中的文件进行监控并提取相应特征。

（2）特征优化模块对于提取的特征信息进行筛选以及优化处理，在众多特征中选取对区分正常应用与恶意应用贡献率大的特征，去除可能会造成混淆的特征。模型通过单个特征对检测结果的贡献度对特征进行帕累托分析，从而对特征进行有效的筛选与过滤。然后通过改进的人工蜂群算法对不同的特征组合进行迭代检验，最终选出使检测结果最优的特征组合。通过对提取特征的筛选与优化，使得检测系统的性能有大幅度的提升。

（3）分类模块通过选取不同的分类算法，通过对样本的训练以及测试，最终得出区分正常与恶意应用的分类结果，从而验证本文所提方法的可行性。

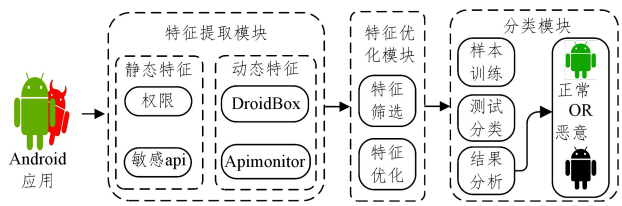

图 1 检测系统框图

4 特征提取方法

4.1 静态特征提取

本文采用动静态结合检测的方法对应用进行特征提取。在静态特征方面，分别利用 Android SDK（Software Development Kit）[12] 自带的可运行程序 aapt 提取样本权限特征以及开源静分析工具 Androguard[13] 中的 androlyze. py 在 python 平台提取敏感 api 特征，权限占用率以及敏感 api 占用率可通过计算得出，静态特征提取的流程图如图 2 所示。

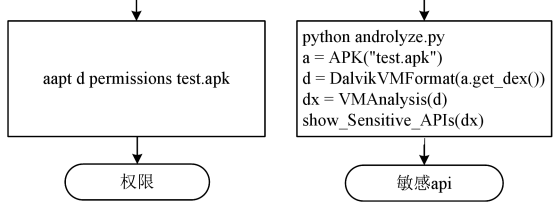

图 2 静态特征提取流程图

（1）Android SDK 是一款安卓软件开发工具包，是软件开发工程师为特定的软件包、软件框架、硬件平台、操作系统等建立应用软件的使用的开发工具集合。aapt 即 Android Asset Packaging Tool，在 SDK 的 build-tools 目录下，该工具可以查看，创建，更新 zip，jar，apk 格式的文档附件。通过指令“aapt d permissions test. apk”可查看 test. apk 文件所申请的权限。

(2)Androguard 是 Google code 上提供的工具，其具有对 apk 以及 dex，odex，arsc 等文件的分析处理功能。androguard 是基于 python 的，它将 apk 文件中的 dex 文件、类、方法等都映射为 python 的对象。简单来说，androguard 提供了 apk 文件的反向工程、恶意软件检测和威胁评估，以及程序行为可视化的功能。Androlyze. py 是 Androguard 中一个强大的静态分析工具，它提供的一个独立的 shell 环境来辅助分析人员执行分析工作。本文在 androlyze. py 下对 test. apk 进行敏感 api 的提取。

(3)此处定义权限占用率 P 和敏感 api 占用率 Q，样本 apk 的申请的权限数量与所有样本申请权限数量的比值为权限占用率，样本 apk 调用的敏感 api 数量与所有样本调用敏感 api 数量的比值为敏感 api 占用率，公式如下：

$$\begin{cases}\varphi=\{apk_i\mid 1\leqslant i\leqslant n\}\\ N=sum1(\varphi)\\ P_i=\dfrac{n_i}{N}\times 100\%\end{cases}\tag{1}$$

$$\begin{cases}\varphi=\{apk_i\mid 1\leqslant i\leqslant n\}\\ M=sum2(\varphi)\\ Q_i=\dfrac{m_i}{M}\times 100\%\end{cases}\tag{2}$$

其中，n 表示样本数量，φ 表示样本集合，n_i 和 m_i 分别代表样本申请权限的数量和样本调用敏感 api 的数量，$sum1$ 和 $sum2$ 函数表示对所有样本的权限和敏感 api 不重复求和。

4.2　动态特征提取

在动态特征方面，分别利用 DroidBox 和 Apimonitor 工具进行特征的提取，通过 Android SDK 中的 SDK Manager 创建 Android 虚拟机，本文创建的虚拟机环境为 4.1.2 版本的 Android 系统，CPU 为 ARM(armeabi-v7a)。动态特征提取的流程图如图 3 所示。

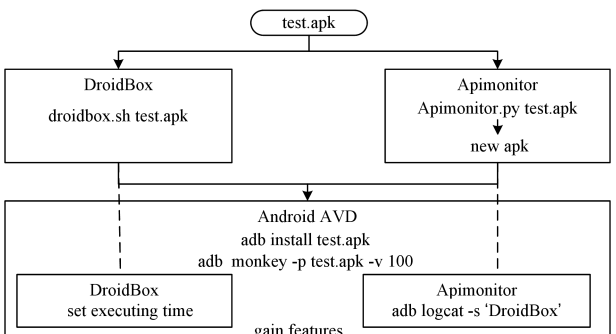

图 3　动态特征提取流程图

(1)DroidBox 是一款分析 Android 的动态分析软件，它的核心技术称作 TaintDroid，从字面上理解就是污染机器。其思想是将所有隐私数据变成污染源，在程序运行的过程中如果你想对污染源进行截取、拼装、加密、传递等操作，那么新生成的数据也会被污染。如果有被污染的数据被泄露出去，那么就发生了隐私泄漏。通过对污染源进行标记即可在程序运行时获取相应特征参数。通过对应用的监控，DroidBox 可以获取以下参数：1)获取 APK 包的哈希值；2)获取网络通信数据，sent data & received data；3)文件读写操作；4)DexClassLoader 加载信息；5)网络通信、文件、以及 SMS 中的信息泄露；6)权限绕过；7)调用 Android API 进行的加密操作；8)Broadcast Receiver 组件信息；9)SMS 短信和电话信息。

(2)Apimonitor[14] 是在 DroidBox 基础上进行改进的一个 python 文件，即 Apimonitor. py，其通过对 apk 文件中的 api 信息进行插桩处理，从而对程序运行过程中调用的敏感 api 进行检测并输出可视化的 log 信息。通过执行命令行“python Apimonitor. py test. apk”，可得出经过插桩后的新 apk 文件，其过程包含反编译 apk 文件，遍历 smali 代码，并在配置文件中将需要监控的 api 进行重新设定，从而达到在虚拟机中运行新的 apk 文件时可获取到需要监控的敏感 api 的调用信息。

(3)adb(Android Debug Bridge)，是一个用于通过电脑与模拟器或者真实设备进行交互的工具，它是一个命令行工具，可以通过在终端窗口输入 adb 指令对虚拟机中的 apk 文件进行安装、运行和卸载等操作。“adb shell monkey”指令是给指定的设备发送压力测试，即相当于模拟用户在虚拟机中对相关 app 进行随机操作，从而达到动态地对 apk 文件进行特征提取。

5　基于改进人工蜂群算法的特征选取方法

5.1　帕累托分析

帕累托分析法(Pareto analysis)[10] 是制定决策的统计方法，用于从众多任务中选择有限数量的任务以取得显著的整体效果。帕累托分析法使用了帕累托法则，即做 20％的事可以产生整个工作 80％的效果。其原型是 19 世纪意大利经济学家帕累托所创的库存理论。帕累托运用大量的统计资料分析当时的一些社会现象，概括出一种关键的少数和次要的多数的理论，并根据统计数字画成排列图，后人把它称为“帕累托曲线图”。这种排列图把累积百分数在 0～80％之间的因素称为 A 类因素，是主要因素；把累积百分数在 80％～90％之间的因素称为 B 类因素，是次要因素；把累积百分数在 90％～100％之间的因素称为 C 类因素，在这一区域内的因素是最次要因素。

本文在基于改进人工蜂群算法的 Android 恶意应用检测模型对特征进行选取前需对特征进行基于帕累托分析的特征筛选，从而将重要的特征进行组合以有效地筛选，提高 Android 恶意应用检测模型的检测性能。

人工蜂群算法(Artificial Bee Colony Algorithm，ABC)是一个由蜂群行为启发的算法[12]，在 2005 年由 Karaboga 小组为优化代数问题而提出。该算法通过模拟自然界中的蜂群采蜜行为解决数学问题，即寻求问题的近似最优解集合。该算法具有初始参数少、步骤简单等优点。

5.2　特征选择

特征选择是一种组合优化问题，其从 N 个特征集合中选出 M 个特征的子集($M\leqslant N$)，去除冗余或不相关的特征，使得处理后的数据集不仅包含的特征个数更少，而且能够提高分类算法在原有特征集合的分类性能[15]。

按照评价函数与分类算法之间的关系，特征选择可以分为过滤式和封装式。如图 4 所示，封装式的子集评价与分类算法无关，一般根据数据内部特性(如互信息、相关性)得到。封装式的特征选择方法使用特定的分类器算法进行评价，如图 5 所示，产生的特征子集都通过特定的分类器进行评价之

后再进行下一步选择，该方法产生的子集与使用的分类。

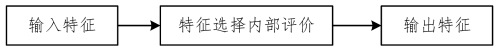

图 4 封装式特征选择

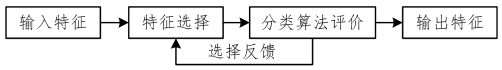

图 5 过滤式特征选择

因为过滤式特征选择的方法不能对所有分类算法都有效，所以本文选取过滤式特征选取办法，即在特征选择过程中通过分类算法参与反馈从而得出最优的特征组合。

5.3 算法描述

最初，人工蜂群算法是用来解决寻找最优解集合问题的，Palanisamy 与 Kanmani 在 2012 年首次将单目标人工蜂群算法用于解决特征选择问题[16]。针对本实验环境，需要对人工蜂群算法进行改进从而适应对恶意应用检测的特征进行最优选取的需求。为描述基于改进人工蜂群算法对 Android 恶意应用特征选取方法，定义以下参数。

定义 1 待选取 Android 恶意应用特征数量 N。N 表示在进行改进人工蜂群算法的特征选取前选取的特征数量总数。

定义 2(雇佣蜂 Employed) Employed 表示雇佣蜂，算法中先通过雇佣蜂对特征组合位置进行初步的侦查，当侦查蜂侦查完毕后雇佣蜂转变成侦查蜂，*nEmployed* 表示雇佣蜂的数量。

定义 3(跟随蜂 OnLooker) OnLooker 表示跟随蜂，算法中跟随蜂在接收雇佣蜂的侦查信息后从中选取优质的特征组合位置作为基准点，并以其为中心在其周围寻找最优特征组合位置。*nOnLooker* 表示跟随蜂的数量。

定义 4(侦查蜂 Scout) Scout 表示侦查蜂，侦查蜂在算法中会在每轮循环的最后进行随机特征组合位置的选取，从而避免局部最优特征组合位置的出现。*nScout* 表示侦查蜂的数量。

定义 5(特征组合位置 Position) Position 是一个 0，1 向量，表示特征组合的选取。其中 0 代表不选取该特征，1 代表选取该特征。

定义 6(特征组合位置矩阵 PopPositon) PopPositon 将每次运算中选取的 Position 组成矩阵，PopPositon 的每一行都为 Position，*nPop* 表示 PopPosition 中 Position 的最大数量，即 PopPosition 最多为 *nPop* 行的矩阵。

定义 7(代价函数 CostFunction) CostFunction 是对 Position 的评价标准，本文选取该 Position 对应的特征组合所得到的 Android 恶意应用检测准确率作为代价函数。

定义 8(特征组合代价矩阵 PopCost) *PopCost* 表示不同 Position 对应的 CostFunction 值，PopCost 的每一行都为一个 CostFunction。

定义 9(最大迭代次数 iter_max) 对算法循环的次数进行限制，即最多进行 *iter_max* 次循环。

定义 10(历史最优特征组合位置 BestSol) *BestSol* 表示在每一次循环过程中最优的 Position 值。

定义 11(特征组合选取概率矩阵 Probability) *Probability* 表示 OnLooker 选取组合特征位置的概率矩阵，即针对 Employed 得到的侦查结果，OnLooker 根据 Probability 在不同 Position 上指派不同数量的 OnLooker，Probability 根据 CostFunction 设置，CostFunction 越高的 Position 具备更大的概率，会派遣更多的 OnLooker 进行以该 Position 为中心的侦查。

定义 12(特征组合开采量矩阵 Mine) Mine 对每个 OnLooker 侦查过的 Position 次数进行记录，设置上限 L，即当某 Position 在 Mine 中的数量达到 L，该 Position 对应的 Employed 自动转换为 Scout。

基于改进人工蜂群算法的特征选取方法的流程图如图 6 所示。

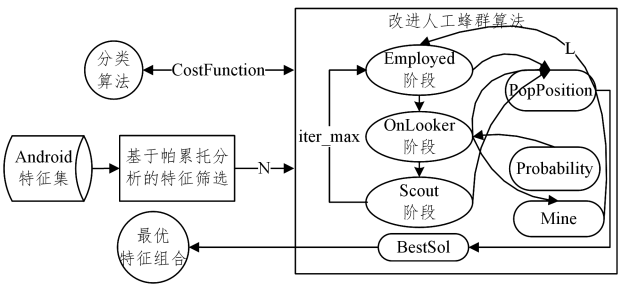

图 6 基于改进人工蜂群算法的特征选取方法流程图

从图 6 中可以看出，改进的人工蜂群算法是一个基于帕累托分析筛选后得到的 N 个特征选取最优特征组合的优化过程，该过程主要分为 5 个阶段，即初始阶段、循环过程中的雇佣蜂选取阶段、跟随蜂选取阶段、侦查蜂选取阶段以及判别模块，其伪代码如算法 1 所示。

算法 1 基于改进人工蜂群算法的特征组合选取伪代码

```
输入:特征数量 N
输出:最优组合特征
1. Initialization Phase;
2. while(i<iter_max){
3.    Employed Bees Phase;
4.    Onlooker Bees Phase;
5.    Scout Bees Phase;
6.    Memorize the best solution;
7. }
8. Judge Phase;
9. Get best Freatures.
```

在初始阶段，首先对于接收到的 N 个特征进行初始化 PopPosition，即随机生成长度为 N 的 nPop 个 0，1 矩阵。同时计算每个 Position 对应的 CostFunction，生成对应的 PopCost，以及选取其中最优的 CostFunction 记录到第 0 代的 BestSol。

雇佣蜂进行侦查特征组合是循环的第一环节，在此阶段 Employed 遍历所有 Position，根据每一个 Position 随机生成一个 NewPosition，在 Position 和 NewPosition 之间进行贪婪选择，选取 CostFunction 更大的特征组合进行更新，如果选择了 NewPosition，则更新 PopPosition 和 PopCost，否则在 Mine 中第 Position 位上增加 1。

在跟随蜂阶段，对 Probability 进行更新，OnLooker 根据 Probability 执行轮盘赌选择新的 NewPosition。轮盘赌选择[22]又称为比例选择算子，其基本思想是个体被选中的概率与其适应度函数值成正比，即对应 Probability 概率更大的

Position 分配更多的 OnLooker。

在侦查蜂阶段，根据阈值抛弃某些 Position 并选取 New-Position，即遍历 PopPosition，查找对应 Mine 中值大于 L 的 Position，对这些 Position 进行更新，并重新计 CostFunction，并将 Mine 中对应的值归零。其伪代码如算法 2 所示。

算法 2　侦查蜂阶段

```
1. for i in range(1,nPop):
2.    if Mine[i] >= L:
3.       PopPosition[i]=rand([VarMin,VarMax],[1,nVar])
4.       PopCost[i]=CostFunction(PopPosition[i])
5. Mine[i]=0
```

为确保改进的人工蜂群对每个特征进行无遗漏的遍历，增加判别阶段，判别阶段的流程图如图 7 所示。即将所有 Position 都未取值为 1 的特征进行默认选定，其余特征不变重新执行新的改进人工蜂群算法。

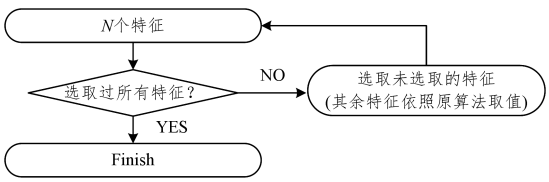

图 7　判别模块

6　实验设计与结果分析

6.1　实验环境

本文使用 Python 语言实现了基于改进的人工蜂群算法的 Android 恶意应用检测系统，本系统对于 3376 个恶意应用和 1015 个正常应用进行检测，实验在内存为 4 GB，处理器为 AMD A10-9630P RADEON R5，10 COMPUTE ORES 4C+6G 2.60 GHz 的机器上完成。静态特征提取以及整体检测分类过程在 windows 下完成，动态特征提取过程在 linux 系统中的虚拟机中进行，虚拟机环境为 4.1.2 版本的 Android 系统，CPU 为 ARM(armeabi-v7a)。

6.2　数据处理

6.2.1　基于帕累托分析的特征筛选

对样本进行权限特征提取后共得到 490 种特征，其中恶意应用所申请的特征种类有 127 个，非恶意应用申请的特征有 479 个，考虑到很多特征是应用自定义设置的权限特征，大部分特征仅被少量应用所调用，若将这些权限全部作为特征，分类结果非但不会更优相反可能会产生较大的误差，并且会造成不必要的运算量，所以需要对这些权限特征进行去除。同理对于选取的 50 种敏感 api 特征与动态过程中提取到的 25 种特征(其中 DroidBox 类特征 13 项，APIMonitor 类特征 12 项)，也需要对出现次数不多的特征进行去除。分别对每组中的特征 t_i 按如图 8 进行特征筛选。

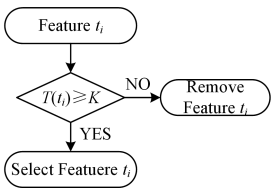

图 8　特征筛选流程图

其中，$T(t_i)$ 表示 t_i 在应用中出现的数量。K 为本实验设定的特征阈值，即选取出现次数大于 k 的特征，将出现次数小于 k 的特征筛选出去。根据帕累托分析选取 k 的取值，即将权限特征和敏感 api 特征按照出现的次数 t_i 进行降序排列，在第 n 个特征处对全部 N 个特征进行切块($0\leqslant n\leqslant N$)，分别计算前后两个部分对分类结果的贡献度的占比 $Proportion$，如式(3)所示。

$$Proportion=\frac{P_{1\sim n}}{P_{1\sim n}+P_{n+1\sim N}}\times 100\% \tag{3}$$

其中，$P_{1\sim n}$ 表示前 n 个特征对恶意应用检测的准确率(检测准确率的计算如式(9)所示)，$P_{n+1\sim N}$ 表示其余部分对恶意应用检测的准确率，分类算法在整个检测系统中保持统一。当贡献度达到 80%以上时，保留此时的 n 值，最终在保留的所有 n 之中选取最小的 n，将第 n 项对应的 k 值作为特征筛选流程图中的 k 值，其流程图如图 9 所示。

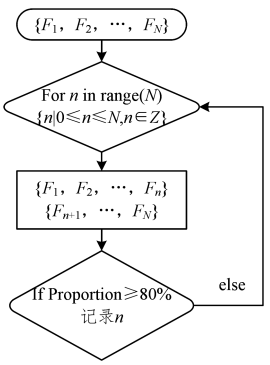

图 9　k 值选取流程图

本文选取的 k 以及筛选后选取特征的数量如表 1 所列。

表 1　k 值及筛选后特征数量

	k 值	原特征数量	筛选后特征数量
权限特征	1079	490	27
敏感 api 特征	879	50	25
动态特征	625	25	13

由表 1 可见经过基于帕累托分析的筛选得到权限特征 27 项，敏感 api 特征 25 项，动态特征 13 项，加上第 2 节介绍的权限占用率和敏感 api 占用率这 2 项特征，在经过特征筛选阶段我们得到 67 项特征。在这一阶段筛选掉接近 90%的多余特征，能有效增强检测模型的性能。

6.2.2　基于改进人工蜂群算法的特征选取

将筛选后得到的 67 项特征通过改进的人工蜂群算法进行恶意应用检测模型的特征选取，在改进的人工蜂群算法上分别使用 SVM、朴素贝叶斯、K 近邻和 BP 神经网络 4 种分类算法。实验中改进的人工蜂群算法的参数设置如下：最大迭代次数为 30，单次保存的蜜源数量为 100，跟随蜂数量为 100，初始特征数为 67。

经过基于改进人工蜂群算法的特征选取，针对不同分类算法得到了不同的特征组合，得到的特征组合数量与目标函数(即改进人工蜂群算法中的检测准确率)的值如表 2 所列。

表 2　特征选取数量与检测准确率

	特征数量	检测准确率
SVM	28	0.9705
朴素贝叶斯	25	0.9500
K 近邻	26	0.9710
BP 神经网络	31	0.9727

从表 2 中数据可见，通过基于改进的人工蜂群算法阶段，

针对不同的分类算法选取得到不同数量的特征。另外通过对 BestSol 矩阵的分析可以得出在进行人工蜂群算法的循环中，5 次循环已经使得结果稳定，之后的循环过程不再产生最优的解，因此可以适当减少最大迭代次数以减少运算的时间复杂度。

6.3 分类模型评估

分类器的性能及分类结果的准确性可由一系列参数评定，下面给出一些评定参数的定义。

定义 13 混淆矩阵（也称误差矩阵）是表示精度评价的一种标准格式。本文采用 2 行 2 列的矩阵表示混淆矩阵，如表 3 所列。

表 3 混淆矩阵

预测↓	正常应用	恶意应用
正常应用	TP	FP
恶意应用	FN	TN

其中，TP(True Positive)为正常应用被预测为正常应用的数量，FN(False Negative)为正常应用被预测为恶意应用的数量，FP(False Positive)为恶意应用被预测为正常应用的数量，TN(True Negative)为恶意应用被预测为恶意应用的数量。由混合矩阵可计算出相应的评估参数，具体如下：

TPR(True Positive Rate)表示在所有正常应用中被预测为正常应用的比例，*TPR* 的值越高表示分类效果越好。敏感度(Sensitivity)可用 TPR 表示。

$$Sensitivity=TPR=\frac{TP}{TP+FN} \tag{4}$$

FPR(False Positive Rate)表示在所有恶意应用中被误预测为正常样本的比例，*FPR* 的值越低表示分类结果越好。

$$FPR=\frac{FP}{FP+TN} \tag{5}$$

精度(Precision)表示在所有预测为正常应用中正常应用所占的比例，*Precision* 的值越高表示分类效果越好。

$$Precision=\frac{TP}{TP+FP} \tag{6}$$

ACC(Accuracy)表示在所有应用中被正确预测的比例，*ACC* 的值越高表示分类效果越好。

$$ACC=\frac{TP+TN}{TP+TN+FP+FN} \tag{7}$$

定义 14 ROC 曲线(Receiver Operating Characteristic Curve)又称为感受性曲线，是根据一系列不同的二分类方式(分界值或决定阈)，以真阳性率 TPR(灵敏度)为纵坐标，假阳性率 FPR(特异度)为横坐标绘制的曲线。曲线上的各点表示在不同情况下对同一信号刺激的反应，能较容易地查出任意界限值时对类别的识别能力。ROC 曲线越靠近左上角，实验的准确性就越高。最靠近左上角的 ROC 曲线的点是错误最少的最好阈值。

ROC 曲线的面积就是 AUC(Area Under the Curve)。AUC 用于衡量二分类问题机器学习算法的性能，*AUC* 越接近 1，则表示实验的诊断价值越佳。

6.4 实验结果

为证明本文提出的基于改进的人工蜂群算法的 Android 恶意应用检测模型的有效性，选取 5.1 节介绍的 Sensitivity，Precision，ACC 和 AUC 等评估参数对检测模型进行评估。随机选取半数的总样本作为特征筛选与优化阶段的集合，剩余部分作为分类模型使用的集合，在分类模型阶段选取半数样本作为训练集，其余为测试集。为增强实验结果的准确性，本文对分类器进行 5 次测试，取 5 次评估参数的平均值作为每个分类模型的评估参数，评估结果如表 4 所列。

表 4 分类模型评估

Test Set	*Sensitivity*	*Precision*	*ACC*	*AUC*
SVM	0.9649	0.9621	0.9742	0.9878
朴素贝叶斯	0.9486	0.9433	0.9523	0.9810
K 近邻	0.9669	0.9599	0.9703	0.9881
BP 神经网络	0.9700	0.9687	0.9732	0.9877

由表 4 可见，将经过改进的人工蜂群算法选取的特征应用到 Android 恶意应用检测模型后，检测模型具有良好的检测效果，不同分类算法均达到 95%以上的检测准确率，其中选取 SVM 算法时检测准确率最高，可以达到 97.42%。由表中评估参数也可看出，选取不同种类的分类算法可得到不同的检测准确率。本文所使用的改进的人工蜂群算法是在固定某一分类算法后，通过对特征的高质量选取，达到提升 Android 恶意应用检测模型的检测准确率的效果。

为验证本文方法的有效性，分别选取全部特征、随机选取与表 2 中等数量的特征与本文通过改进的人工蜂群算法选取作为恶意应用检测的特征进行对比实验。为使对比结果明显，本文选取检测准确率作为比较参数并通过柱状图进行比较，对比结果如图 10 所示，其中横坐标为选取不同的分类算法，纵坐标为对恶意应用检测的检测准确率。

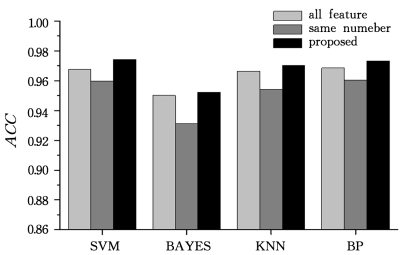

图 10 选取不同特征时 ACC 值

从图 10 可以看出，当选取本文所使用的改进的人工蜂群算法得到的特征作为恶意应用检测的特征时，模型获得的检测准确率最高，选取全部的特征时检测准确率次之，随机选取与改进人工蜂群算法提取到的等数量特征时的检测准确率最低，本文方法能在分类算法不变的基础上将检测准确率提升 1%左右，因此本文改进的人工蜂群算法提取的特征是有效可行的，能有限减少运算复杂度，即将特征降维的前提下有效地提升 Android 恶意应用检测模型的检测准确率。同时，本文改进的人工蜂群算法能有效适用于各种分类算法，由于过滤式封装的性质，可以把任意分类算法放入改进的人工蜂群算法中，从而不同的分类算法都能加入人工蜂群算法之中，以选取出适用于不同分类算法的特征组合，从而提升 Android 恶意应用检测的准确率。

结束语 综上所述，本文提出了一个基于改进的人工蜂群算法的 Android 恶意应用检测方法，采用动静态相结合的方式对安卓应用进行特征提取，该方法能对提取出的 Android 应用的特征进行有效的筛选，并通过改进的人工蜂群算将特征进行最优组合，使其适用于分类算法，使 Android 恶意检测效果达到更优。本文选取 3376 个恶意样本以及 1015 个

正常样本对本检测系统进行检验，通过选取不同的分类算以及对比实验证实了本文所提出的方法具有可行性与优越性。本文所提方法具有以下优点：

(1)在特征选取方面，通过改进的人工蜂群算法，通过将检测准确率作为价值函数对不同的特征组合进行迭代，从而选出最优的特征组合，提升恶意应用检测模型的检测准确率。

(2)本文提出的方法具有可扩展性，因为所使用的改进的人工蜂群算法对特征的选取基于过滤式封装，即更改不同的分类算法时仍然可以适用。

本文所提出的模型也存在一些需要完善的方面：

(1)在本文提出的改进的人工蜂群算法只以检测准确性为价值函数，当特征更多时可能会使选取的特征组合数量过多，可以增设特征数量为另一个价值函数，即用更少的特征数量得到相对较高的检测准确性，如相关文献[17]。

(2)本文中的特征全部为 0，1 类型，当有些特征不能转换为此类型时需要对改进的人工蜂群算法进行部分完善。

(3)本文选取的恶意应用相对较老，应增大样本库以使实验结果更具有普遍性。

参 考 文 献

[1] Mobile Operating System Market Share Worldwide[OL]. http://gs. statcounter. com/os-market-share.

[2] Google Play[OL]. https://play. google. com.

[3] 《2018 年 Android 恶意软件专题报告》[OL]. https://www. anquanke. com/post/id/171110.

[4] QIN Z Y, WANG Z Y, WU F B, et al. Android malware detection based on multi-level signature matching[J]. Application Research of Computers, 2016, 33(3): 891-895.

[5] NING Z, SHAO D C, et al. Android Static Analysis Based on Signature and Data Flow Pattern Mining[J]. Computer Science, 2017, 44(S2).

[6] YANG H, ZHANG Y Q, HU Y P, et al. Android malware detection method based on permission sequential pattern mining algorithm[J]. Journal on Communications, 2017, 34(S1): 106-115.

[7] NAVARRO L C, NAVARRO A K W, GRÉGIO A, et al. Leveraging Ontologies and Machine-learning Techniques for Malware Analysis into Android Permissions Ecosystems[J]. Computers & Security, 2018, 78: 429-453.

[8] KABAKUS A T, DOGRU I A. An in-depth analysis of Android malware using hybrid techniques[J]. Digital Investigation, 2018, 24(3): 25-33.

[9] YANG H, ZHANG Y Q, HU Y P, et al. Malware Behavior Detection System of Android Applications Based on Multi-Class Features[J]. Chinese Journal of Computers, 2014, 37(1): 15-27.

[10] 马建光，姜巍. 大数据的概念，特征及其应用[J]. 国防科技，2013(2): 10-17.

[11] ENCK W, GILBERT P, HAN S, et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones[J]. ACM Transactions on Computer Systems (TOCS), 2014, 32(2): 5.

[12] Android SDK[OL]. https://android-sdk. en. softonic. com.

[13] Androlyze. py 的使用方法[OL]. https://blog. csdn. net/u013107656/article/details/51790153.

[14] APImonitor-DroidBox 的原理分析[OL]. https://code. google. com/p/droidbox/wiki/APIMonitor.

[15] AKBARI R, HEDAYATZADEH R, ZIARATI K, et al. A multi-objective artificial bee colony algorithm[J]. Swarm and Evolutionary Computation, 2012, 2: 39-52.

[16] PALANISAMY S, KANMANI S. Artificial bee colony approach for optimizing feature selection[J]. International Journal of Computer Science Issues (IJCSI), 2012, 9(3): 432.

[17] HANCER E, XUE B, ZHANG M, et al. Paretofront feature selection based on artificial bee colony optimization[J]. Information Sciences, 2018, 422: 462-479.

[18] YUAN Z, LU Y, XUE Y. Droiddetector: android malware characterization and detection using deep learning[J]. 清华大学学报自然科学版(英文版), 2016, 21(1): 114-123.

[19] SARACINO A, SGANDURRA D, DINI G, et al. MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention[C]// IEEE Transactions on Dependable and Secure Computing. 2016.

[20] HANCER E, XUE B, ZHANG M, et al. Pareto front feature selection based on artificial bee colony optimization[J]. Information Sciences, 2018, 422: 462-479.

(上接第 408 页)

[3] 宋永进. 4G 无线通信系统的网络安全分析 [J]. 网络安全技术与应用，2017(9): 101-120.

[4] 张焕国，韩文报，来学嘉，等. 网络空间安全综述[J]. 中国科学：信息科学，2016, 46(2): 125-164.

[5] 乔恩·爱德尼，威廉·阿尔保. 无线局域网安全实务-WPA 与 802.11i[M]. 北京：人民邮电出版社，2006: 37-45.

[6] 冯栋柱，杨登. 基于 VLAN 技术在高校校园网建设中的应用[J]. 网络与通信，2010(26): 131-134.

[7] 谭润芳. 无线网络安全性探讨[J]. 信息科技，2008, 37(6): 24-26.

[8] 沈芳阳. 基于 IEEE802.11 系列标准的无线局域网安全性研究[D]. 广州：广东工业大学，2004.

[9] 吴贤平. 基于 802.1x 的校园网用户身份认证设计与实现[J]. 制造业自动化，2012, 34(5): 47-49.

[10] 李丹，闫晓弟，耶健，等. 基于开放源码软件 Freeradius 的无线网络认证系统实现[J]. 中国现代教育装备，2012(17): 65-67.

[11] 王志军，张帆. 虚拟网络技术在计算机网络信息安全中的应用研究 [J]. 科技创新与应用，2017, 4403: 93.

[12] 文峰. 计算机网络病毒与网络安全维护探究 [J]. 计算机光盘软件与应用，2015.

[13] 胡俊. 计算机网络安全技术在网络安全维护中的应用研究[J]. 科技风，2014(15): 54.

[14] 李文琴，汪大清. VLAN 技术在组建校园网中的应用[J]. 计算机与现代化，2007(5): 76-78.

[15] 商阳. VLAN 技术及其在校园网中的应用[J]. 科技信息，2007(33): 391.