Android 平台恶意应用程序静态检测方法①

蒋 煦[1], 张长胜[2], 戴大蒙[2], 慕德俊[1]

[1](西北工业大学 自动化学院, 西安 710071)
[2](温州大学 物理与电子信息工程学院, 温州 325000)

摘 要: 本文构建的静态检测系统主要用于检测 Android 平台未知恶意应用程序. 首先, 对待检测应用程序进行预处理, 从AndroidManifest.xml文件中提取权限申请信息作为一类特征属性; 如待检测应用程序存在动态共享库, 则提取从第三方调用的函数名作为另一类特征属性. 对选取的两类特征属性分别选择最优分类算法, 最后根据上述的两个最优分类算法对待检测应用程序的分类结果判定待检测应用程序是否为恶意应用程序. 实验结果表明: 该静态检测系统能够有效地检测出 Android 未知恶意应用程序, 准确率达到 95.4%, 具有良好的应用前景.

关键词: 安卓; 静态检测; 机器学习; 特征属性; 恶意应用

Static Detection of Android Malware

JIANG Xu[1], ZHANG Chang-Sheng[2], DAI Da-Meng[2], MU De-Jun[1]

[1](Department of Automation, Northwestern Polytechnical University, Xi'an 710071, China)
[2](Department of Physics & Electronic Information Engineering, Wenzhou University, Wenzhou 325000, China)

Abstract: The static detection system referred in this article is used to detect unknown malware. First of all, the Android application sample is preprocessed and permission information that the application developer applies for is extracted as a kind of characteristic attribute. If there is shared object file, the name of function called from other libraryis is extracted as another kind of characteristic attribute. Secondly, the two optimal classifiers chose are used to hand two classes of features separately. Finally, the result of comprehensive judgment system depends on the result of classifiers about two types of characteristic attribute. The result of experiment shows that the static detection system can detect the unknown malware efficiently, and the accuracy of the static detection system reaches up to 95.4%.

Key words: Android; static detection; machine learning; characteristic attribute; malware

随着移动智能终端井喷式的增长, 越来越多的人使用智能手机, 智能手机的品牌虽然繁多, 但智能手机的操作系统主要集中在 Android 和 ios 系统. 2015 年 2 月, IDC 发布了 2014 年智能手机出货量数据, 采用 Android 和 ios 操作系统的智能手机出货量占全部智能手机出货量的 96.3%, 其中 Android 操作系统的手机出货量为 10.59 亿部, 市场份额为 81.5%. 因此, 基于 Android 平台的第三方应用软件如同雨后春笋, 数以万计, 随之而来的是恶意应用程序的问题. 在中国, 因为政策的原因, Android 手机用户无法从 Google Play 下载应用程序, 缺少了 Google Play 对应用程序的审查机制. 据 360 互联网安全中心的报告显示, 2014 年全国仅 360 互联网安全中心累计截获 Android 平台新增恶意程序样本 326 万个. 由此可见, Android 平台恶意程序的问题在我国尤为严重. 目前, 国内外典型的 Android 平台恶意代码检测技术主要有基于特征码和基于行为检测这两种方式[1]. 基于特征码的检测方法可以有效地检测出待安装的应用程序是否具备病毒特征库中已存在病毒的特征, 但无法检测出未知的病毒, 相对滞后. 国外著名的 Android 恶意代码检测工具 Androguard 也是基于特征码的方法[2]. 随着恶意病毒及其变种不断增加, 会导致恶意代码特征库需要频繁

① 基金项目:国家自然科学基金(61303224);浙江省科技厅公益项目(2104C31079)
收稿时间:2015-07-10;收到修改稿时间:2015-09-08

© 中国科学院软件研究所 http://www.c-s-a.org.cn

更新, 制约了基于特征码检测方法的发展. 当然, 随着云技术的发展, 可以有效解决样本库内容不断增加的难题.

基于行为的恶意代码检测主要是静态检测和动态检测两种方法, 两种方法各有利弊. 就 Android 平台而言, 静态方法主要是通过反编译应用程序, 获取应用程序组件、权限、API 等信息, 选择部分信息作为特征属性, 与恶意样本库中的样本进行相似度比较, 判断待检测应用程序是否为恶意应用程序. 静态检测方法具有无需运行应用程序、代码覆盖率高、检测时间短等优点, 所以很多杀毒软件都采用这种方法; 缺点是代码混淆、加密等在程序运行时才恢复其真面目的代码, 采用静态检测方法无法检测出[3].

动态检测是指在虚拟机或手机上执行应用程序, 通过捕捉系统关键数据的变化, 如日志信息、文件访问、数据库等信息, 对这些关键的行为进行分析, 预测应用程序是否为恶意病毒. 该检测方法的优点是解决了代码混淆、加密等需要程序运行才能检测出的问题, 同时也不需要维护恶意代码特征库; 但动态检测的实时性要求比较高, 需要持续地对系统运行状况进行监测, 对手机运行的性能有影响[4].

静态检测与动态检测相比, 优点在于: 静态分析不受一个程序的特定执行过程约束, 适用于程序的所有执行过程; 而动态检测技术只能对符合特定的行为进行检测[5]. 同时, 静态检测能够减少成本, 提高性能.

1 相关工作

目前, Android 平台的恶意代码检测相对已经成熟, 但主要针对特定类型的恶意代码或者更多地依赖与恶意代码特征库样本相似度的比较. 该检测方法在针对未知的恶意应用程序时, 检测的准确率有所下降. 童振飞[6]采用抽取 apk(Android Package)类名和调用函数名作为特征属性, 通过计算与样本相似度的方法对 Android apk 文件进行分类. 实验证明, 该方法可以有效的对 APK 进行分类, 但整个实验的样本取自非恶意应用程序, 设计为从随机的应用程序中识别出一种类型的应用程序, 没有用真实的恶意应用程序作为样本, 同时该检测方法局限于在 JAVA 层, 未考虑动态共享库的加载. 秦中元等[7]设计的 Android 平台恶意软件静态检测的方法不依赖于病毒库的收集和更新, 能够成功检测出隐私窃取与恶意扣费的恶意行为, 但检测方法针对的恶意代码类型局限, 同时检测的样本数量过少. 李子锋等[3]采用静态数据流分析技术, 模拟应用程序的执行过程, 通过跟踪应用程序对常量值的使用, 可以检测出恶意行为, 但是未对本地代码进行安全检测. 李子峰在文中也指出 40%的应用程序存在执行本地命令或加载本地库文件的行为, 加载本地代码中可能存在提权行为. Schmidt[5]抽取应用程序的 Elf 可执行文件(包括恶意和非恶意应用程序)函数调用作为样本, 并运用 Part, Prism 和 Nearest Neighbor Algorithms 等算法对抽取的样本进行分类, 对于选择的特征属性而言, 未必是最优的算法. 杨欢等[8]建立了基于多类特征的 Android 应用恶意行为检测系统, 将静态检测与动态检测相结合, 能够有效的判断未知应用程序是否为恶意病毒. 但该检测系统构建的三层混合系综算法选择的基础分类器都属于强分类器, 而系综算法选择的基础分类器应该为弱分类器[9]. 本文采用杨欢的系综算法对样本进行测试, 发现对特征属性进行系综分类后效果不如单一的强分类器, 说明该分类算法不够完善, 理论上存在缺陷.

基于上述研究, 本文更加专注于动态共享库的分析、特征属性的选择和数据挖掘算法的研究, 构建由应用程序申请的权限分析模块和加载的动态共享库分析模块相结合的全新的 Android 平台恶意应用程序静态检测系统.

2 检测系统架构

Android 的权限机制是 Android 系统在 Linu-x 的权限管理基础上扩展的一种安全保护机制, 这些控制权限包括拨打电话、定位、网络使用等 100 多个内置的权限, 位于 Android 系统的框架层, 它的主要任务是对应用程序可以执行的某些操作进行权限细分和访问控制, 限制应用开发者对系统资源或者用户信息的滥用. 同时 Android 系统不允许动态注册权限, 在安装应用程序之前就必须申请应用程序所需要用到的权限[10]. 由于 Android 应用程序需要在 AndroidManifest.xml 中使用 <uses-permis-sion>标签来标注所申请的权限. 所以本文对应用程序的 AndroidManifest.xml 文件进行扫描, 提取应用程序申请的权限信息作为特征属性进行分析.

实验所用的恶意应用程序来源于参考文献[11], 从中随机抽取 300 个恶意应用程序进行分析, 发现恶意应用程序样本中有 55%含有动态共享库, 而从

© 中国科学院软件研究所 http://www.c-s-a.org.cn

Google play 下载的 300 个非恶意应用程序样本中仅有 39%包含动态共享库, 说明恶意应用程序更倾向于使用动态加载技术. 因此, 本文将动态共享库作为重要研究对象之一.

本文 Android 平台静态检测系统架构如图 1 所示. 首先, 待检测的应用程序通过预处理模块处理, 该模块使用 apktool 工具反编译应用程序, 可以得到 Android 应用程序的源代码、图片、XML 配置、语言资源等文件. 其中配置文件 AndroidManifest.xml 描述了 package 中暴露的组件、权限、启动位置等, 用 Python 语言编写的代码扫描 AndroidManifest.xml, 提取<uses-permission>标签后面的申请权限作为特征属性; 将应用程序的文件类型改为 zip 格式, 对其解压缩, 如果存在动态共享库, 就对其执行 Android NDK 工具包中的 readelf 命令, 得到 ELF(Excutable Linkable Format)文件的存储结构和内容, 提取出来自外部调用、动态共享库或者系统调用的函数名,将这些抽取的函数名作为特征属性. 将上述的两类特征属性作为综合检测模块的输入, 作出应用程序是否为恶意应用程序的判定. 综合检测模块里会对两类特征属性分别进行分类算法的, 如果其中任一特征属性分类结果为恶意应用程序, 则判断该待检测应用程序为恶意应用程序.

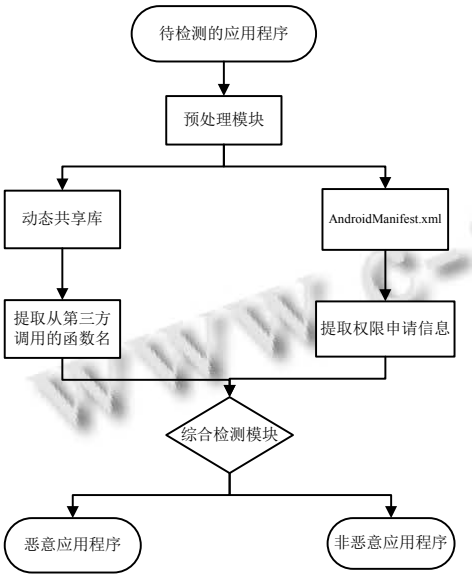

图 1 静态检测系统架构

3 关键技术与算法

3.1 特征属性提取技术

在数据挖掘的研究中, 会通过计算样本距离来比较样本之间的差异, 而样本距离是通过属性来计算的. 不同的属性在样本空间的权重是不一样的, 即关联度不同, 因此有必要进行属性选择. 有些研究者认为应用程序中存在敏感权限的组合即为恶意应用程序, 本文认为这种方法不具备实际应用的价值, 比如应用程序“百度地图”就会在安装时, 申请获得地理位置、Internet 等敏感权限, 虽然恶意应用程序申请这样的敏感权限组合频率更高, 但并不代表“百度地图”是恶意应用程序. 本文从实际出发, 在所有恶意应用程序申请的权限中选取信息增益高的申请权限作为特征属性, 通过机器学习, 去挖掘恶意应用程序申请权限的规律, 用于检测未知的恶意应用程序. 首先使用 apktool 工具反编译从参考文献[11]中随机抽取出的 500 个恶意应用程序和从 Google Play 下载的 500 个非恶意应用程序, 得到 1000 个应用程序的 AndroidManifest.xml 全局配置文件, 扫描这 1000 个全局配置文件, 提取出所有应用程序申请的权限, 使用 Weka 工具采用 InfoGain 算法对提取出的申请权限特征属性进行筛选, 最终选定信息增益超过设定阈值的 95 个申请权限特征属性.

目前, 大多数针对 Android 平台恶意应用程序的行为研究主要集中在 Java 层, 依赖于控制流程与数据流程的分析, 即如果恶意应用程序将静态信息存在 Java 代码中, 能够有效地检测出来. 但恶意应用程序的开发者为了躲避检测会采用动态加载技术, 则 Java 层的检测方法完全失效[12]. 所以, 本文将动态共享库作为研究的一个重要方面. 如果待检测应用程序中存在动态共享库, 则对其执行 Android NDK 工具中 readelf 命令, 得到 ELF 文件的存储结构和内容, 包含了代码、数据、符号表、字符串等信息. 这些信息按不同的属性, 以段的形式存储, 其中就包括 dynsym 段, 该段包含了在本文件中定义的全局符号以及本文件中引用的外部函数或者变量. 通过对大量恶意和非恶意应用程序中动态共享库进行分析可以发现, 动态共享库中从第三方库调用的函数主要来 libc.so,libstdc++.so, libm.so, liblog.so 等 Android 平台内置的共享库. 因此, 本文将“dynsym”段中从第三方调用的函数名作为研究动态共享库模块的特征属性, 即该函数在符号表中的 Type 属性为“FUNC”和“Ndx”属性为“UND”, 有这两个标签的函数意味着此函数来自外部调用、共享库或者系统调用的函数. 而本地定义的函数不予采纳为特征属性是因为恶意应用程序的开发者为了逃避检测可以

© 中国科学院软件研究所 http://www.c-s-a.org.cn

定义不同的函数名和变量名称, 而代码执行内容却不变.

3.2 分类算法的选择

算法的性能依赖于数据集的特性以及产生差异性的方法等因素, 所以差异性是提高算法性能的一个重要因素[9]. 本文构架的静态检测系统主要依赖于两个检测模块, 一个是关于应用程序申请权限的检测模块, 一个是关于动态共享库中函数调用的检测模块. 由于两个模块的特征属性是不同的类型, 所以为了使检测效果更优, 应该针对不同类型的特征属性选择最适合它的最优的分类算法. THEA[8]采用了系综算法以提高针对不同特征属性的基础分类器性能, 进而提高恶意应用程序的检测率.

DroidMat[13]提取应用程序的 Intent、API 等作为特征属性, 采用了 KNN 和 NB 两种分类算法用于检测恶意应用程序. 本文构建的静态检测系统所选取的分类算法结合前人的研究成果, 选取分类算法差异性较大的 4 种算法, 即 KNN,NB,C4.5,Part 算法, 用 10 折交叉确认的方法对两种特征属性模块进行不同分类算法的训练. 通过对训练集分类结果的对照研究, 选择各自特征属性的最优分类算法. 图 2 为选择最适合该特征属性的最优分类算法的过程.

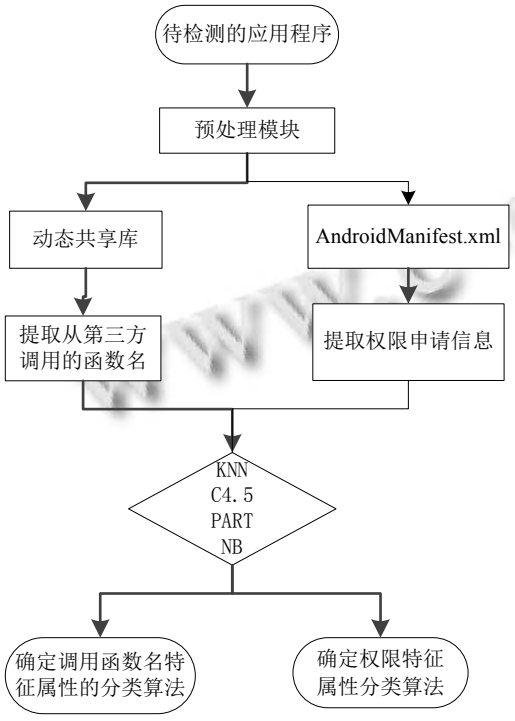

图 2 选择最优分类算法的过程

首先, 创建动态共享库函数调用检测模型的训练集, 该训练集由恶意应用程序和非恶意应用程序构成, 恶意应用程序随机取自于参考文献 [11] 中的 500 个恶意应用程序; 随机下载于 Google Play 商店 500 个非恶意应用程序, 这 1000 个训练样本均含有动态共享库, 即 lib 文件夹下有后缀名为.so 的文件, 同时训练样本均使用目前主流的杀毒软件进一步确认.

其次, 对上述的 1000 个应用程序进行反编译, 利用 4.1 中介绍的特征属性提取方法, 提取 Android Manifest.xml 全局配置文件中权限申请信息作为一类特征属性, 如应用程序中存在动态共享库, 则抽取从第三方库调用的函数名作为另一类特征属性.

最后, 使用 Weka 工具对两类特征属性进行分类算法的学习, 从 KNN,PART,C4.5,NB 中选出针对该特征属性的最优分类算法. 因为 Weka 工具存储数据采用 ARFF(Attribute-Relati-onFile Format)格式, 所以实验需要用 python 语言编写的程序将以上两类特征属性按照相应的要求编写成后缀名为 arff 的文件.

最优分类算法的选择取决于分类算法学习的结果, 主要依赖于Accuracy、TP Rate、FP rate、Precision等相关指标参数. Accuracy反映了分类器对整个训练集正确判断的能力, 即将恶意应用程序及非恶意应用程序都判断正确的能力; TP Rate为某类被正确分类的训练样本占该类样本的比例, 同时等同于Recall; FP Rate是某类被误判为另一类的数量占该类的比例; Precision为某类判断正确的数量占结果判定为该类的比例; F-Measure是综合Precision和Recall二者指标的评估指标, 用于综合反映整体的指标; ROC指标反映了分类器分类的效果, 一般认为越接近0.5, 模型准确率越低, 最好状态接近1,完全正确的模型面积为1. 以下是相关参数指标的计算公式, 如果是针对非恶意应用程序而言, t_pos 为非恶意应用程序被正确分类为非恶意应用程序的数量, t_neg 为恶意应用程序被正确分类为恶意应用程序的数量, f_pos 为恶意应用程序被错误分为非恶意应用程序的数量, f_neg 为非恶意应用程序被错误分为恶意应用程序的数量.

$$Accuracy=\frac{t_pos+t_neg}{t_pos+t_neg+f_pos+f_neg}$$

$$TP\ \text{Rate}=\frac{t_pos}{t_pos+f_neg}$$

$$FP\ \text{Rate}=\frac{f_pos}{t_neg+f_pos}$$

© 中国科学院软件研究所 http://www.c-s-a.org.cn

$$\text{Recall}=\frac{t_pos}{t_pos+f_neg}$$

$$\text{Precision}=\frac{t_pos}{t_pos+f_pos}$$

$$F-measure=\frac{2\times recall\times precision}{recall+precision}$$

3.3 综合检测模块的建立

综合检测模块从待检测应用程序中抽取的两类特征属性通过各自最优分类算法的检测, 如果两个特征属性检测模块判断结果中任一分类器判定待检测应用程序为恶意应用程序, 则判定待检测应用程序为恶意应用软件. 综合检测模型如图 3 虚线框内所示.

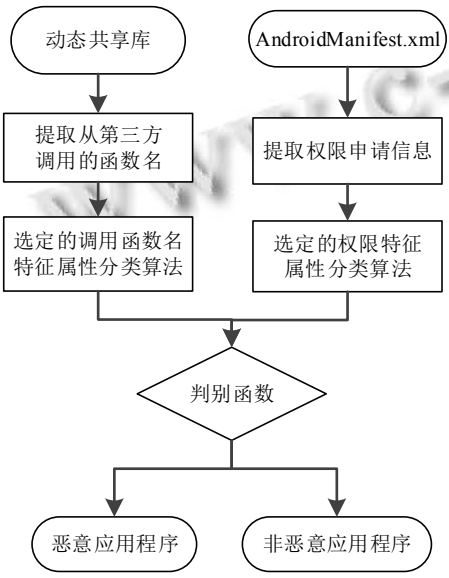

图 3 综合检测模块示意图

4 实验结果

4.1 实验环境

本文检测系统使用 Java 和 python 语言实现, 自动化地检测基于 Android 平台的第三方应用程序. 其中, 两类特征属性的提取和判断部分的函数由 Python 语言编写的代码实现, 最优分类算法使用 WEKA 工具的 jar 包实现. 随机选取参考文献中[11]500 个恶意应用程序和随机下载于 Google Play 的 500 个非恶意应用程序作为测试集进行测试, 测试集和训练集无交集. 所有实验在内存为 4G, 处理器为 Intel(R)Core (TM) i5-3337U 1.80GHz 的机器上完成.

4.2 算法的选择

对选取的两类特征属性分别进行 KNN、PART、C4.5、NB 机器算法的学习, 得到表 1、表 2 的结果, 表 1 是申请权限信息作为特征属性模块的分类算法学习结果, 表 2 是调用函数名作为特征属性模块的分类算法学习结果, 表 1 和表 2 中的 Class1 是指 500 个恶意应用程序集合, Class0 是指 500 个非恶意应用程序集合. 分类结果的参考指标有很多, 但不同的应用环境, 关注的参考指标侧重点不同. 本文选择特征属性最优分类算法的策略: 在 Accurracy、F-Measure、ROC 参考指标数值接近的情况下, 主要依据恶意应用程序的识别率(TP Rate)和非恶意应用程序的误识率(FP Rate)来选择最优算法,恶意应用程序识别率越高, 非恶意应用程序的误识率越低, 越能说明该算法能够有效地检测出恶意应用程序. 理由是本检测系统的目的是为了检测出恶意应用程序, 恶意应用程序的识别率可以看出该算法是否能够有效的检测出恶意应用程序, 非恶意应用程序的误识率可以看出该算法将恶意应用程序判定为非恶意应用程序的情况, 之所以看中这个指标是因为检测恶意应用程序的算法将恶意应用程序判定为非恶意应用程序带来的后果远超过非恶意应用程序被划归为恶意应用程序.

通过比较, 发现 KNN 算法对于两个特征属性检测模块都是最优分类算法, KNN 算法的各项参考指标都优于其他分类算法. ROC 面积指标也充分印证了对于这两类特征属性模块而言, KNN 算法分类效果最佳. 所以, 本文的两个特征属性检测模块均选取 KNN 算法作为分类算法. 从表 1、表 2 的对比, 也可以看出动态共享库中函数调用特征属性的分类结果明显优于申请权限特征属性的分类结果, 说明恶意应用程序的开发者利用动态加载的技术十分普遍.

表 1 申请权限特征属性分类结果

Algorithm	Class	TP Rate	FP Rate	Precision	Recall	F-measure	ROC	Accuracy(%)
KNN	1	0.939	0.063	0.939	0.939	0.939	0.96	93.78
	0	0.937	0.061	0.937	0.937	0.937	0.96	
C4.5	1	0.939	0.127	0.885	0.939	0.923	0.879	90.68

© 中国科学院软件研究所 http://www.c-s-a.org.cn

	0	0.873	0.061	0.932	0.873	0.923	0.879	
PART	1	0.878	0.076	0.923	0.878	0.9	0.945	90.06
	0	0.924	0.122	0.88	0.924	0.901	0.945	
NB	1	0.976	0.165	0.86	0.976	0.914	0.955	90.68
	0	0.835	0.024	0.971	0.835	0.898	0.955	

表 2 函数调用特征属

Algorithm	Class	TP Rate	FP Rate	Precision	Recall	F-measure	ROC	Accuracy(%)
KNN	1	0.981	0	1	0.981	0.99	0.983	98.93
	0	1	0.019	0.976	1	0.988		
C4.5	1	0.962	0.098	0.927	0.962	0.944	0.93	93.61
	0	0.902	0.038	0.949	0.902	0.925		
PART	1	0.962	0.049	0.962	0.962	0.962	0.95	95.74
	0	0.951	0.038	0.951	0.951	0.951		
NB	1	0.981	0.341	0.788	0.981	0.874	0.88	84.04
	0	0.659	0.019	0.964	0.659	0.783		

4.3 实验结果

将测试集的 1000 个应用程序(并非所有应用程序都包含动态共享库)通过本文架构的静态检测系统, 得到的混淆矩阵如表 3, 准确率为 95.4%. 恶意应用程序的识别率 95.8%, 恶意程序的误识率 4.2%. 测试结果充分证明了该模型能够有效检测出恶意应用程序. 通过表 1、表 2、表 3 的对比, 可以发现检测系统的 Accuracy 指标相对于单一的检测应用程序申请权限 KNN 算法分类结果的 Accuracy 指标有所提高, 但相对于单一的动态共享库函数调用特征属性的 KNN 算法分类结果而言, Accurancy 指标有所下降. 这是因为所有的应用程序都有 AndroidManifest.xml 全局配置文件, 表 1 的训练集的元素可以是任一应用程序; 表 2 的训练集是从包含动态共享库的应用程序中随机抽取的, 而最后测试集是随机抽取的 1000 个恶意和非恶意应用程序, 即有些应用程序不含动态共享库. 所以本检测系统将两个分类器集成于一体, 随机抽取应用程序作为测试集, 更具有实际应用的价值.

表 3 本文检测系统的混淆矩阵

预测为↓	恶意程序	非恶意程序
恶意程序	479	25
非恶意程序	21	475

4.4 与其他检测系统比较

目前针对 Android 平台恶意应用软件的检测研究成果有许多, 但由于实验采集的样本不同, 样本中包含动态共享库的应用程序所占比例不同, 直接会导致不同的检测结果; 而且大部分的研究成果都只给出思路和方法, 未公布源码; 另外, 由于本文的恶意应用程序的实验样本均取自于 2013 年之前的, 对于主流的杀毒软件而言, 其 MD5 值早已在其病毒的特征库中. 所以, 本文未与其他的检测系统进行横向比较. 但就其它研究者的研究内容看, 检测系统的架构组成复杂, 当然特征属性选择越多, 静态检测与动态检测相结合等手段会提高恶意应用程序的检测率, 但相对于本文的检测系统需要消耗更多的成本, 并且从其他研究者公布的研究成果看, 恶意应用程序的检出率一般在 94%左右. 综上所述, 本文的静态检测系统在不影响检测率的情况下, 具有高效、轻便的特点, 对资源的依赖更少.

5 结语

本文静态检测系统采取应用程序申请权限分析模块与应用程序加载的动态共享库分析模块相结合的方法, 分别采用适合于模块特征属性的机器算法, 能够有效地检测出 Android 平台未知恶意应用程序, 具有高效、轻便的特点. 但仍然存在需要在后面的研究工作中完善的地方:

① 本检测系统的第一步工作就是对应用程序进行反编译, 软件开发人员为了防止自己的应用程序被反编译, 会采取加壳等手段, 本检测系统未考虑这部分, 所以会出现部分应用程序无法检测.

② 静态检测和动态检测各有利弊, 存在互补的

© 中国科学院软件研究所 http://www.c-s-a.org.cn

关系. 下一步要重点研究动态检测, 将静态检测判断为恶意应用程序作为动态检测的重点, 将动态检测系统集成到本文检测系统中.

参考文献

1 李华,刘智,覃征,等.基于行为分析和特征码的恶意代码检测技术.计算机应用研究,2011,28(3):1127–1129.

2 王菲飞.基于 Android 平台的手机恶意代码检测与防护技术研究[硕士学位论文].北京:北京交通大学,2012.

3 李子锋,程绍银,蒋凡.一种 Android 应用程序恶意行为的静态检测方法.计算机系统应用,2013,22(7):148–151.

4 Enck W, Gilbert P, Chun B, et al. TaintDroid: An information-flow tracking system for real time privacy monitoring on Smartphones. Proc. of the 9th USENIX. Vancouve. BC, Canada. 2010. 1–6.

5 Schmidt AD, Bye R, Schmidt HG, et al. Static analysis of executables for collaborative malware detection on Android. Proc. of the 9th IEEE International Conference on Communications. Dresden, Germany. 2009. 631–635.

6 童振飞.Android 恶意软件静态检测方案的研究[硕士学位论文].南京:南京邮电大学,2013.

7 秦中元,徐毓青,梁彪,等.一种 Android 平台恶意软件静态检测方法.东南大学学报,2013,43(6):1162–1167.

8 杨欢,张玉清,胡玉璞,等.基于多类特征的 Android 应用恶意行为检测系统.计算机学报,2014,37(1):20–26.

9 王秀霞.分类器的选择性集成及其差异性研究[学位论文].兰州:兰州理工大学,2013.

10 刘智伟,孙其博.基于权限管理的 Android 应用行为检测.信息网络安全,2014,(6):72–76.

11 Zhou YJ, Jiang XX. Dissecting Android malware: Characterization and evolution. Proc. of the 33rd IEEE Symposium on Security and Privacy. Oakland, USA. 2012. 95–109.

12 文伟平,梅瑞,宁戈,等.Android 恶意软件检测技术分析和应用研究.通信学报,2014,35(8):78–85.

13 Wu DJ, Mao CH, Wei TE, et al. DroidMat: Android malware detection through manifest and API calls tracing. Proc. of the 7th Asia Joint Conference on Information Security. Tokyo, Japan. 2012. 62–69.

© 中国科学院软件研究所 http://www.c-s-a.org.cn