计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(2):197−201 [doi: 10.15888/j.cnki.csa.006183]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于多特征和 Stacking 算法的 Android 恶意软件检测方法①

盛　杰, 刘　岳, 尹成语

(重庆大学 弘深学院, 重庆 401331)

摘　要: Android 由于其广泛的普及率使得其平台上的恶意软件数量不断增加, 针对目前大部分方法采用单一特征和单一算法进行检验, 准确率不高的不足, 提出了一种基于多特征与 Stacking 算法的静态检测方法, 该方法能够弥补这两方面的不足. 首先使用多种特征信息组成特征向量, 并且使用 Stacking 集成学习算法组合 Logistic, SVM, k 近邻和 CART 决策树多个基本算法, 再通过训练样本进行学习形成分类器. 实验结果表明, 相对于使用单一特征和单一算法其识别准确率得到提高, 可达 94.05%, 该分类器对测试样本拥有较好的识别性能.

关键词: Android; 恶意软件检测; 集成学习; Stacking 算法; 多特征

引用格式: 盛杰,刘岳,尹成语.基于多特征和 Stacking 算法的 Android 恶意软件检测方法.计算机系统应用,2018,27(2):197–201. http://www.c-s-a.org.cn/1003-3254/6183.html

Detection Method of Android Malware Based on Multi-Feature and Stacking Algorithm

SHENG Jie, LIU Yue, YIN Cheng-Yu

(Hongshen Honors School, Chongqing University, Chongqing 401331, China)

Abstract: As a result of the Android system's popularity, the number of malware on it is increasing rapidly. In this study, a static detection method based on multi-feature and Stacking algorithm is proposed, which can make up the shortcomings of the two aspects, i.e., based on single feature and single algorithm. Firstly, this study uses a variety of feature information to compose the eigenvector, and uses the ensemble learning algorithm of Stacking to combine Logistic, SVM, k-Nearest Neighbor and CART decision trees. Then, classifiers are generated through training samples. The experimental results show that the recognition accuracy is up to 94.05% compared with the single feature and single algorithm, and the classifier has better recognition performance.

Key words: Android; malware detection; ensemble learning; Stacking algorithm; multi-feature

Android 系统随着智能终端设备的发展在全球广泛普及, 根据调查报告显示, 截止 2016 年第四季度, Android 的市场占有率已达到 81.7%[1]. 在其高速发展的同时也带来了严重的问题, 大量恶意 apk 在互联网上流行, 据 360 互联网安全中心统计, 2016 年全年, 360 互联网安全中心累计截获 Android 平台新增恶意程序样本 1403.3 万个, 平均每天新增 3.8 万恶意程序样本[2]. 恶意 apk 的一些行为例如盗取隐私信息、锁屏勒索等, 对用户的利益造成损害, 使得恶意 apk 的检测尤为重要.

目前, Android 平台恶意代码检测已展开广泛的研究, 主要思想方法是直接由桌面平台移植到移动平台, 再根据 Android 平台的一些特点加以改进, 主要技术可划分为静态检测与动态检测[3], 两种方法各有优劣, 本文将研究静态检测.

现今主流方法是在提取出 apk 的各种特征后通过

① 基金项目: 重庆大学国家级大学生创新创业计划 (201610611016)
收稿时间: 2017-05-06; 采用时间: 2017-05-26

一些机器学习算法对已知样本进行学习形成分类器, 再对未知样本进行分类[4]. 权限信息作为 Android 软件的典型静态型特征, 在恶意 apk 检测方面得到广泛的应用. 文献[5]通过对权限进行聚类去冗余得到相互独立并相关性强的特征集合, 再通过改进的朴素贝叶斯算法进行分类; 文献[6]以窃取用户隐私数据为切入点, 提炼出对隐私资源十分敏感的权限组合, 并实现了一种专门针对窃取隐私的恶意应用的检测方法. 但若恶意 apk 制作者故意对权限进行混淆, 仅以权限作为特征拥有明显的不足性, 除了权限特征, apk 的 API 调用与系统调用也是重要的特征.

恶意 apk 不仅拥有静态特征, 还拥有许多运行时的动态特征. 对此, 文献[7]指出了现阶段大部分检测技术的不足和缺陷, 并从安装、激活等特征入手, 研究各种动态恶意行为, 例如路过式下载、软件升级、权限提升; Padriya[8]等人研究了恶意软件的各种行为特征和静态特征, 并探讨了各种检测方法. 针对各种动态特征, 孙润康[9]等人以行为特征为基础实现了一种 Android 软件行为的动态检测框架, 并采用 SVM、朴素贝叶斯等多种基本分类算法验证了其有效性; Luoxu Min[10]等人也提出了基于运行时行为的动态检测方法并且结合了静态分析技术与反编译技术. 可以看出大多方法通常采用朴素贝叶斯, k 近邻等基本算法, 而某一特定算法未必适合此时所选用的特征数据, 因此具有一定局限性. 文献[11]提出了一种充分考虑 Android 应用多类特征的三层混合系综算法, 获取 apk 的各种动态特征与静态特征, 对不同种类特征在不同层面采用不同的分类算法, 对 apk 进行综合评判, 提高了检测准确率, 但其特征的提取过程和算法的实现过程都较为繁琐, 增加了时间成本和计算机资源的消耗.

针对以上方法的局限性, 本文提出了一种基于多特征和 Stacking 算法的 Android 恶意软件静态检测方法. 作者选取权限、API、SO 库作为特征, 这些特征能够明显反映 apk 的恶意倾向, 并且作为静态特征提取较为容易, 缩短特征提取的时间, 将它们进行组合, 在保证特征向量有效的情况下降低计算的复杂度; 在机器学习阶段运用 Stacking 集成学习算法将多个弱学习器组合后形成强学习器提高识别准确率.

1 Android 平台机制简述

Android 系统架构主要分为四个层面: 应用程序层, 应用程序框架层, 系统运行库层, Linux 内核层, 如图 1 所示, Android 程序由 Java 语言编写并经过编译后生成 Dex 可执行文件, 以字节码的形式在 Dalvik 虚拟机中运行. 每一个进程都拥有自己的虚拟环境, 不同进程是相互隔离的, 以此保障系统的安全性. 权限机制是 Android 精心构建的一项安全机制, 若应用程序要进行一些敏感操作和需要某些特权时, 比如访问或使用系统的文件、手机的硬件资源都必须要在 AndroidManifest.xml 配置文件中声明相应的权限[12], 并且在安装时提醒用户程序所使用的权限, 让用户自行判断, Android 官网共列出了所有可使用的 137 种权限[13]. 为了满足 Android 平台下一些软件的复杂功能, 凭借 Java 语言的 JNI 特性 Google 向开发者提供了 Android NDK. 开发者可以使用 C 与 C++编写 SO 动态链接库让 Java 程序直接调用, 这一方法提高了一些程序的运行效率, 并且使软件的跨平台性得到提高. 另一方面复杂的 C/C++代码让破解者难以反编译, 使用它进行 apk 的加密混淆效果更好, 增强了 Android 软件的安全性.

2 检测算法

2.1 检测算法框架

检测算法的整体框架如图 2 所示,

(1) 首先分别提取所有训练样本权限、API、SO 库三种不同特征.

(2) 对获得的所有特征进行卡方检验去除与判断是否为恶意软件无关的特征, 得到强相关性特征集合.

(3) 针对每个 apk, 将其特征中的强相关性组合为特征向量.

(4) 在分类器训练过程中采用 Logistic, SVM, kNN 作为初级分类器, CART 决策树作为次级分类器实现 Stacking 算法, 并通过特征向量数据训练成最终分类器.

(5) 运用测试样本验证算法性能.

2.2 检测算法框架

2.2.1 特征提取

本文的方法使用了 apk 文件中的权限申明信息, API 使用信息, SO 库的使用情况, 各特征提取方式如下:

1) 谷歌 Android 官网共申明了 137 种权限, 使用 SDK 中集成的工具 aapt 可提取 apk 中的权限信息.

2) API 调用信息的提取利用 baksmali 对 apk 文件中的 classes.dex 进行反编译得到 smali 文件, 再对所有的 smali 文件进行扫描, 即可获得样本的 API 调用集合.

3) SO 文件存在于 apk 包中 lib 目录下, 直接对 apk 解压后提取.

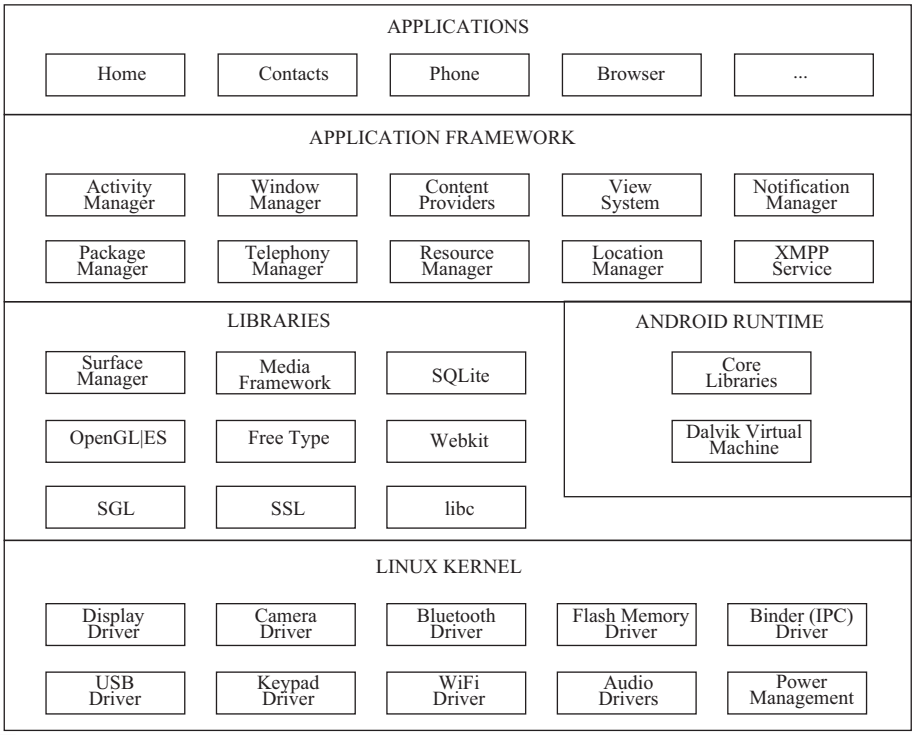

图 1 Android 总体架构

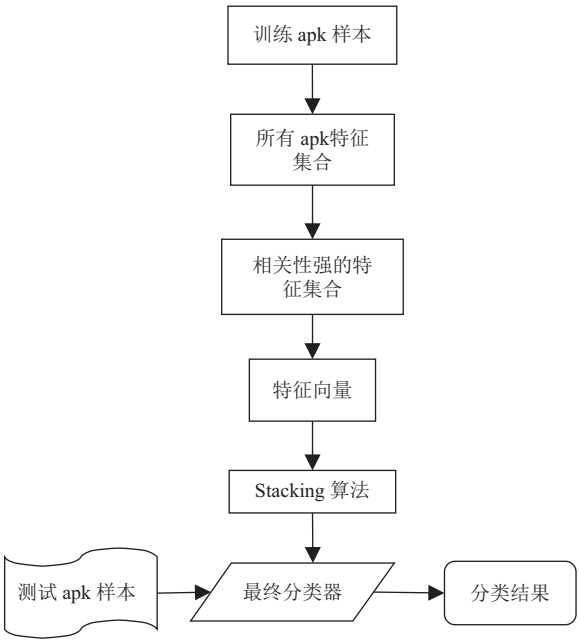

图 2 检测系统框架

2.2.2 特征选取

从 apk 中提取的这些特征并不是都可利用于对恶意软件的识别, 某些特征无论是正常软件还是恶意软件都会频繁出现, 故其中存在着大量的冗余, 必须去除一些与识别恶意软件无关的特征, 降低无关特征对识别的影响和计算的复杂性.

皮尔森卡方检验是由 Pearson 提出的一种假设检验方法, 可用于独立性检验, 即验证两个变量是否相互独立. 我们选用基本的四格卡方检验, 其公式如下:

$$\chi^2 = \frac{(ad-bc)^2 \times N}{(a+b)(c+d)(a+c)(b+d)} \quad (1)$$

在式 (1) 中 a, b, c, d 分别代表拥有某特征是恶意软件, 拥有某特征非恶意软件, 无某特征是恶意软件和无某特征非恶意软件这四种情况的频数, N 代表总的频数. 针对样本中的所出现的所有特征, 都利用式 (1) 进行计算其卡方值, 卡方值越大则表明该特征与是否为恶意软件有较强相关性, 根据某一阈值即可筛选出合适的特征, 剔除冗余的特征.

2.2.3 特征的组合

针对每一个样本只获取卡方检验后得到的特征集合中的特征, 可将权限的特征向量定义为 $(a_1, a_2,\cdots,a_h)$, 若该样本拥有 k 特征, 则 $a_k = 1$, 否则 $a_k=0$. 同理将 API 特征向量定义为 $(b_1, b_2,\cdots, b_i)$, SO 库信息特征向量定义为 $(c_1, c_2,\cdots, c_j)$, 然后简单组合为 x = $(a_1, a_2,\cdots, a_h, b_1, b_2,\cdots, b_i, c_1, c_2, \ldots, c_j)$, 这种简单组合在一定程度上可降低分类器学习与测试时计算的复杂度.

2.3 分类算法

Stacking 算法是由 Wolpert 于 1992 年提出的一种

集成学习算法, 又称为 Stacked generalization[14–16]. 相比于 Bagging 与 Boosting 在较多情况下采用相同的分类算法训练个体学习器, Stacking 算法结合多个不同的分类算法, 可看做一种特殊的结合策略. Stacking 算法主要分为两层, 将第 0 层的学习器称为初级学习器, 而将第 1 层用于结合的学习器称为次级学习器. 先通过原始的特征数据作为输入训练出多个初级学习器, 在将初级学习器的输出作为特征用于训练出次级学习器, 如图 3 所示.

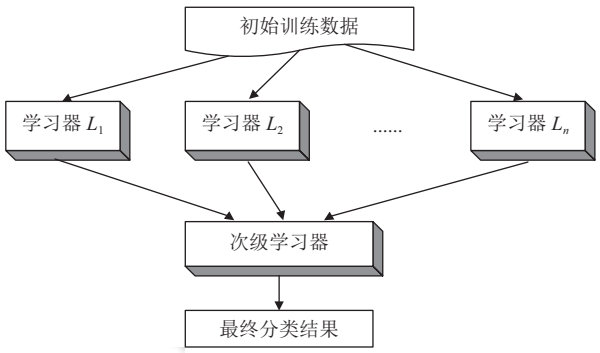

图 3 Stacking 算法结构

具体方法如下:

在初级学习阶段使用 k 折交叉检验方法[17]训练各学习器. 即将初始训练数据集 $T = ((x_1, s_1), (x_2, s_2),\cdots, (x_h, s_h))$ 划分成 k 个大小相似的子数据集 $T_1, T_2,\cdots, T_k$, 将 $T–T_j$ 作为 m 学习算法的训练数据并得到学习器 L_m^j, 然后将 T_j 作为测试数据输入 L_m^j. 对每一个子数据集都用 m 学习算法进行此操作, 最后每一个样本都会得到测试并输出结果 y_{im}. 若共有 n 个学习算法, 训练结束后对每一个样本 x_i, 都会产生 n 个结果, 由它们组成新的特征向量 $y_i = (y_{i1}, y_{i2},\cdots, y_{in})$ 作为次级学习器的训练数据, 标记依然为原标记 s_i.

本文所实现的 Stacking 算法的初级学习器采用 Logistic 回归, 支持向量机, k-近邻作为分类算法, CART 决策树作为次级学习器的分类算法. 伪代码描述:

Input: 初始训练集 $T = ((x_1, s_1), (x_2, s_2),\cdots, (x_h, s_h))$, 初级学习算法 Logistic(), SVM(), kNN(), 次级学习算法 CART(), 交叉检验子集数 k

$T_1, T_2,\cdots, T_k = k\_Fold(T, k)$

Meta_T = {}

FOR EACH T_j in $\{T_1, T_2,\cdots, T_j\}$:

LogisticClf[j] = Logistic($T – T_j$)

SVMClf[j] = SVM($T – T_j$)

kNNClf[j] = kNN($T – T_j$)

FOR EACH x_i in T_j:

y_{i1}= LogisticClf[j].predict(x_i)

y_{i2}= SVMClf[j].predict(x_i)

y_{i3}= kNNClf[j].predict(x_i)

Meta_T.append(((y_{i1}, y_{i2}, y_{i3}), s_i))

END FOR

END FOR

CARTClf = CART(Meta_T)

Output: 初级分类器 LogisticClf, SVMClf, kNNClf 与次级分类器 CARTClf

2.4 学习器的测试

在学习器的训练过程中, 每一种分类算法都生成了 k 个分类器, 测试过程中根据投票法决定该算法分类结果. 即使用这 k 个分类器都对测试样本进行分类, 对可能产生的结果进行投票, 本实验中只有 Normal 与 Virus 两种结果, 最后若 Normal 票数更多, 则该分类算法分类结果为 Normal, 否则为 Virus. 将 LogisticClf, SVMClf, KNNClf 的分类结果组合 (y_{i1}, y_{i2}, y_{i3}) 作为新的向量输入次级学习器 CARTClf 进行分类取得最终结果.

3 实验结果

作者首先分别从 VirusShare[18]和安卓市场获取了 2460 份恶意 apk 样本与 2659 份正常 apk 样本, 并将其中 1460 份恶意样本与 1659 份正常样本作为训练样本

在实验中进行反复测试后将三种特征的阈值分别定位 60、800 和 20, 最后统计得到 53 种权限, 238 种 API, 94 种 SO 库, 如表 1 所示.

表 1 去冗余后的特征

特征	取值
权限	BATTERY_STATS RESTART_PACKAGES RECORD_AUDIO VIBRATE……
API	Landroid/graphics/Canvas→rotate Landroid/view/View→isClickable Landroid/view/ViewGroup→draw Landroid/widget/EdgeEffect→onPull……
SO库	libMD5_v1.so libijksdl.so libgetuiext.so librealm-jni.so……

本实验使用 TP 代表正常样本判断正确数, FN 代表正常样本判断错误数, TN 代表恶意样本判断正确

数, FP 代表恶意样本判断错误数. 定义以下指标:

准确率:

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

从图 4 可以看出, 在使用单种特征的情况下, Stacking 算法相比于其它单个算法有所提高. 另外可以看出由于 SO 库并不是所有的 apk 都使用, 所以检测率较低, 在实验中作为辅助特征.

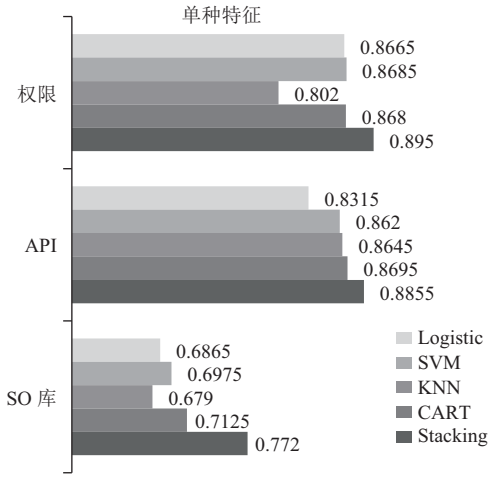

图 4 单种特种使用各方法的准确率

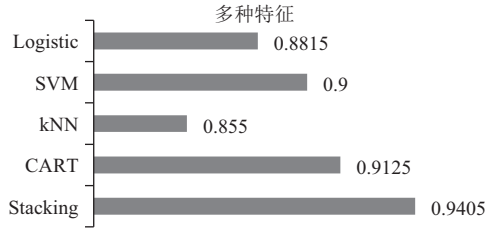

图 5 多种特种使用各方法的准确率

结合图 4 与图 5, 在使用多特征后, 各种检测算法的检测率都有一定提升, 并且 Stacking 算法最为突出, 准确率达到 94.05%, 对恶意软件有较好的识别能力.

4 结语

本文提出了一种多特征结合与 Stacking 算法组合多个基本分类算法的恶意软件检测方法. 从 apk 中获得多种权限, 并去除其中的冗余特征, 再将它们组合为一个特征向量; 选择 Logistic、SVM 和 kNN 分别作为 Stacking 算法的初级学习算法, 将经过它们分类后的输出结果组合为特征向量输入使用 CART 算法的次级学习器进行判断. 实验结果表明本文提出的分类方法具有较强的检测能力. 该方法的不足之处在于相比其它简单的算法, 检测速率稍低, 下一步的研究将改善分类的各环节, 提高分类的速度.

参考文献

1 Gartner. Gartner says worldwide sales of smartphones grew 7 percent in the fourth quarter of 2016. http://www.gartner.com/newsroom/id/36098. [2017-02-15].

2 360 互联网安全中心. 2016 年中国手机安全状况报告. http://zt.360.cn/1101061855.php?dtid=1101061451&did490260073. [2017-02-06].

3 卿斯汉. Android 安全研究进展. 软件学报, 2016, 27(1): 45–71. [doi: 10.13328/j.cnki.jos.004914]

4 Amos B, Turner H, White J. Applying machine learning classifiers to dynamic: Android malware detection at scale. 2013 9th International Wireless Communications and Mobile Computing Conference. Sardinia, Italy. 2013. 1666–1671.

5 张锐, 杨吉云. 基于权限相关性的 Android 恶意软件检测. 计 算 机 应 用, 2014, 34(5): 1322 –1325. [doi: 10.11772/j.issn.1001-9081.2014.05.1322]

6 黄海根, 曾云科. 基于权限组合的 Android 窃取隐私恶意应用检测方法. 计算机应用与软件, 2016, 33(9): 320–323, 333.

7 边悦, 戴航, 慕德俊. Android 恶意软件特征研究. 计算机技术与发展, 2014, 24(11): 178–181.

8 Padriya N, Mistry N. Review of behavior malware analysis for android. International Journal of Engineering and Innovative Technology (IJEIT), 2013, 2(7): 230–232.

9 孙润康, 彭国军, 李晶雯, 等. 基于行为的 Android 恶意软件判定方法及其有效性. 计算机应用, 2016, 36(4): 973–978. [doi: 10.11772/j.issn.1001-9081.2016.04.0973]

10 Min LX, Cao QH. Runtime-based behavior dynamic analysis system for android malware detection. Advanced Materials Research, 2013, 756-759: 2220 –2225. [doi: 10.4028/www.scientific.net/AMR.756-759]

11 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. 计算机学报, 2014, 37(1): 15–27.

12 Enck W, Ongtang M, McDaniel P. Understanding android security. IEEE Security & Privacy, 2009, 7(1): 50–57.

13 Google. Manifest permission. https://developer.android.com/reference/android/Manifest.permission.html. [2017-02-10].

14 Wolpert DH. Stacked generalization. Neural Networks, 1992, 5(2): 241–259. [doi: 10.1016/S0893-6080(05)80023-1]

15 Zhou ZH. Ensemble Methods: Foundations and Algorithms. Boca Raton, FL: Chapman and Hall/CRC, 2012.

16 周星, 丁立新, 万润泽, 等. 分类器集成算法研究. 武汉大学学报 (理学版), 2015, 61(6): 503–508.

17 李航. 统计学习方法. 北京: 清华大学出版社, 2012.

18 VirusShare.com. Because sharing is caring. http://virusshare.com/support. [2017-02-01].