基于多特征融合的安卓恶意应用程序检测方法

王 勇, 蔡建宇, 孟 春, 刘振岩, 薛静锋

北京理工大学计算机学院 北京 中国 100081

摘要 安卓恶意应用程序的检测目前存在着检测速度慢、检测率低等问题, 本文针对这些问题提出了一种基于多特征融合的安卓恶意应用程序检测方法。从 Android 恶意应用的恶意行为特点出发, 运用静态分析和动态分析互相结合的方法, 提取出权限和组件、函数 API 调用序列、系统命令、网络请求等多维度特征, 对维度较大的特征种类使用信息增益方法进行特征的筛选, 取出最有用特征。本文还利用半敏感哈希算法的降维和保持相似度的特性, 提出基于 Simhash 算法的特征融合方法, 将原有的大维度的特征降维到相对较小的维度, 并解决了特征的不平衡问题。融合后的特征使用 GBDT 算法和随机森林算法分类, 检测恶意样本。实验对比分析得出本文使用的多种特征融合的方法在可以大大降低分类的训练时间, 提高检测效率。

关键词 Android 恶意应用检测; 特征融合; Simhash 算法; GBDT 算法; 随机森林算法
中图法分类号 TP393 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2018.07.05

Android Malware Detection Based on Multi-feature Fusion

WANG Yong, CAI Jianyu, MENG Chun, LIU Zhenyan, XUE Jingfeng

School of Computer, Beijing Institute of Technology, Beijing 100081, China

Abstract Based on the background and current situation of Android malicious code detection, this paper studies the reasons that cause low efficiency and low accuracy of Android malicious detection. Take the malicious behavior of Android malicious application as a starting point, we use both static analysis method and dynamic analysis method extract the features. Which contains permissions and components, function call sequence, API call sequence, system commands, network requests, etc. And then use the information gain method to filter out the useless features, extracted the most useful features. In this paper, a feature fusion method based on Simhash algorithm is proposed to reduce the original large feature dimension to a relatively small dimension, and the accuracy of the feature classification is ensured while improving the classification efficiency. Then the features are used to classify and detect malicious samples using the GBDT algorithm and the random forest algorithm. Finally, a series of comparative tests have been made. The results show that the proposed method can greatly improve the detection efficiency and the detection efficiency.

Key words Andriod malware detection; feature fusion; Simhash; GBDT; random forest word

1 引言

目前, 国外的智能手机的操作系统在我国的智能手机中所占比重已经超过了 90%。作为智能手机市场中最为流行的操作系统, Android 的使用量 Android 系统受到了大量关注, 其中除了开发者和用户, 还有大量的恶意代码开发者, 因为智能手机中携带着大量的用户隐私, 甚至包括个人身份信息或银行卡等重要信息, 恶意软件也还有可能通过电话、短信等手段进行恶意吸费等流氓行为, 用户的隐私和个人利益受到了极大的威胁。

在安卓市场日趋复杂, 恶意软件种类和数量也在不断攀升的情况下, 如何自动化的, 高效的解决恶意应用的检测问题, 成为安卓安全首要解决的重中之重[1]。在安卓安全问题上, 国内外研究问题最初想到的是增强现有的安卓安全机制, 但由于安卓版本众多, 这种方法无法覆盖所有的用户版本。所以研究人员慢慢将检测第三方应用的安全性变成新的研究重点。通过检测应用程序的安全性来间接提升 Android 系统的安全性[2]来保证手机用户的安全。这

通讯作者: 孟春, 硕士, 助理研究员, Email: mengch@bit.edu.cn。
本课题得到国家重点研发计划资助(NO.2016YFB0801304)。
收稿日期: 2018-03-30; 修改日期: 2018-05-30; 定稿日期: 2018-06-26

种方法在短时间内发展迅速, 并以静态和动态两种检测手段为主要方法[3]。

目前安卓平台下的恶意应用检测面临着两个问题, 第一个是准确率低, 效率低的问题, 安卓的恶意代码检测一般基于特征提取, 而提取的特征种类和维度起着决定性的作用, 特征维度过多会导致检测时间过长, 特征提取少又会导致检测准确率下降, 为了首先提升检测的准确度, 研究人员不得不牺牲效率, 尽量提取出多种的多维度的特征, 所以如何对这些大维度特征筛选和降维, 是一个难点。第二, 现有的安卓检测技术缺少自动化, 批量化的流程, 许多操作需要用到其他工具并人工判断, 使整个过程变得异常复杂。

因此, 研究一种自动化的, 高效率和高准确率的的检测技术非常有必要。在丰富这一研究领域的同时也具有很强的实用性。本文将 Android 平台的恶意应用检测作为研究重点。在对 Android 系统进行深入分析与总结的基础上, 提出了一种基于多特征融合的安卓恶意应用程序检测技术, 从安卓程序的恶意行为出发, 寻找关键特征, 过滤无用特征。然后本文借鉴现了半敏感哈希算法在降维后仍然保存着相似性的特性, 提出了基于Simhash算法的特征融合方法, 并为小维度样本设置更高的权重, 解决了特征不平衡性, 利用此方法为恶意代码分类的高维特征融合和降维, 进行特征融合与深度处理, 然后使用 GBDT 算法和随机森林算法为恶意样本分类。实验结果表明, 本文提出的方法可以在分类效率和准确率上显著提高。

2 相关研究

目前, 各种分析方法主要集中在静态分析和动态分析两个大方向, 静态分析方法的首次出现是在 2011 年, Étienne Payet 等在研究安卓中的 xml 文件映射问题的时候, 改进了 Julia(Java 字节码静态分析工具)使其适用于 DVM(Dalvik Virtual Machine)字节码的分析[4], 首次将静态分析用于了 Android 程序的分析。 随后, Kui Luo 针对隐私窃取类的恶意代码提出了字节码转换器, 用于将 DVM 字节码转换为 Java 字节码, 并将生成的 Java 字节码输入一种 Java 代码静态分析及切片工具进行分析[5], 这种方法已经比较类似于反编译的方法。

2012 年, Shabtai[6]等人使用动态人工方法, 从自己编写的 Android 恶意应用中提取 API 特征, 然后用分类算法对安卓应用检测和分类。但使用的数据样本少, 没有静态分析, 提取的特征维度和种类都较少。2014 年, Gorla[7]等人从良性应用的 API 为研究点, 提出了一种名为 CHABADA 的安卓恶意程序检测方法。这种方法首先搜集大量的安卓良性应用, 然后利用 LDA(文档主题生成模型)算法, 提取每个应用的主题。然后使用 k-means 算法聚类这些应用的主题, 一共获得了 32 类的应用。对于每一种类别的应用使用通过静态分析方法, 提取出敏感 API 并组成特征向量集, 而对于该种类别的恶意应用往往会调用一些相对异常的敏感 API, 如果发现这些 API 不是良性应用中提取出来的 API, 即说明这个应用是恶意应用的概率比较大。这种方法最后通过机器学习算法 OC-SVM 分类检测恶意应用。2015 年 Avdiienko[8]等人利用静态检测工具 FIOWDROID 提取应用中的敏感信息流。通过将恶意应用与良性应用中的敏感信息流进行对比, 对未知恶意应用进行检测。该方法同样只利用良性样本作为训练数据集来建立检测模型, 这两种方法从良性应用提取特征可以说是另辟蹊径, 最后实验也得出了良好的效果, 但缺点是没有从安卓恶意应用的具体行为考虑, 所以检测准确率和覆盖面都有所欠缺。

2014 年, 杨欢[9]等人提出了一种综合考虑 Android 多类行为特征的三层混合系综算法 THEA(Triple Hybrid Ensemble Algorithm), 用于检测 Android 未知恶意应用。这种方法首先提取了组件、函数调用以及系统调用类特征, 然后, 针对上述 3 类特征设计了三层混合系综算法 THEA, 该算法为每一类特征寻找出最佳的分类器。最后, 搜集现实中的 1000 多个恶意应用和 2000 个非恶意应用, 并对这些应用进行分类检测, 取得比较良好的实验效果。2015 年, 刘阳[1]等人提出了一个利用机器学习算法对 Android 恶意代码进行检测的方案, 该方案首先采集大规模的数据样本, 对 APK 文件采用静态分析, 使用逆向分析技术对 APK 文件做反编译的处理, 提取出大量的特征属性。之后利用随机森林和神经网络这两种分类算法对 Android 应用进行分类。并对比了两种方法的分类效果。结论得出, 随机森林算法在大规模应用样本检测时的表现要优于神经网络算法。该方法提取出了大量了静态特征, 但没有提取动态特征, 并且在特征维度上没有特殊处理, 所以效率上有所欠缺。

3 特征工程

对于恶意代码分析, 最重要的环节为特征的提取和处理上, 该提取哪些特征和如何提取, 是整个分析流程的重中之重, 本文针对安卓恶意应用的行

为特点出发, 提取出了静态特征和动态的大量多种类的特征, 其中静态特征特征包括权限和组件、函数调用和 API 调用特征、文件结构特征、脚本信息特征、字符串信息特征和签名信息特征。动态分析包括系统调用特征、关键路径和数据访问特征、http 请求特征和恶意吸费特征。提取出特征后, 使用信息增益方法筛选特征, 剔除作用较小的特征。

3.1 静态特征提取

静态分析自动化分析反编译后的 apk 文件目录, 提取 apk 文件中的特征, 这些特征包括权限和组件、函数调用和 API 调用特征、文件结构特征、脚本信息特征、字符串信息特征和签名信息特征。静态分析主要使用了反编译技术, 使用反编译工具 Apktool 反编译样本并输出到反编译文件夹, 以供静态特征提取。

3.1.1 权限和组件特征

Android 应用程序由一个或多个组件组成, 所有的应用程序都被赋予了最小权限。由于沙盒机制和线程隔离机制, 使得安卓应用的整体行为特点变得相对容易管理, 不至于在没有任何声明的情况下实施恶意行为。权限和组件在一定程度上反映了程序的恶意性倾向。

权限和组件特征的提取主要通过解析 AndroidManifest.xml 文件获得。解析方法为利用 Python 的 xml.dom.minidom 包解析 xml 标签, 获取 uses-permission、activity、services、contentProvider、Intent Receiver 标签信息。对于组件特征, 主要提取 activity、services、contentProvider 标签内信息, 提取方法与权限提取方法类似。特别的, 对于动态注册的广播接收器特征的提取, 由于动态注册的广播接收器需要开发者自己在代码中注册。用这种方法注册的广播接收器, 需要在 smali 文件中查找对应的代码, 匹配提取。

3.1.2 函数调用和 API 调用特征

提取函数调用和 API 调用特征。提取方法如图 1 所示, 主要针对样本的每个 smali 文件, 用正则匹配方式匹配出所有函数调用, 提取出来的函数调用特征建立 N-gram 特征序列。为了更好地区分和描述恶意行为特征, 本文还提取了 API 调用序列, 提取方法为对于每个 smali 文件, 匹配每个官网 API 对应的 smali 代码, 提取出 API 调用特征。

因为每个样本的函数调用非常多, 而许多函数是开发者自己定义的, 和恶意行为并没有多大直接关联, 本文将函数调用和 API 调用两种特征组合, 选取只包含 API 调用的函数调用序列, 组合方法为遍历每种函数调用序列, 若发现其中的某个函数调用也是 API 调用, 则选取该特征, 否则丢弃该特征。这样就提取出了所有带有 API 调用的函数调用序列特征。

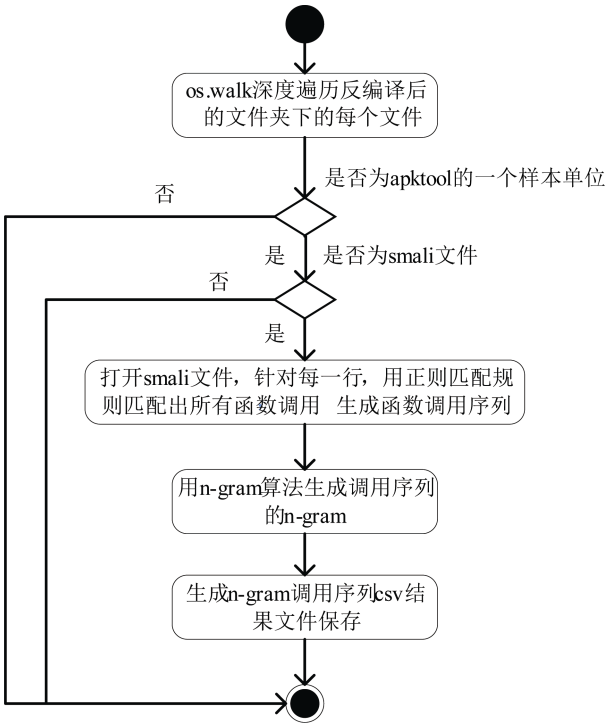

图 1 函数 API 调用序列提取方法

Figure 1 Function API call sequence extraction method

3.1.3 文件结构特征

分析文件结构, 提取文件目录结构特征。有些恶意样本所包含的包名是相同的, 甚至该包内层的文件结构及内容也很类似, 对于这种样本, 我们有理由的相信这他们是“同源”的, 甚至可能出自同样的作者之手, 所以本文提取文件结构特征作为待融合特征。

提取方法为用 python 的 os.walk 方法遍历样本文件夹, 提取出其中的文件作为特征, 因为每个样本下的文件可能太多, 这里只提取代码的文件结构, 即只分析 smali 文件夹结构。提取每个文件时, 除了文件本身作为一种特征, 还将文件的路径也作为一种特征。

3.1.4 脚本信息特征

脚本信息特征。某些安卓应用在运行过程中会执行恶意脚本, 所以脚本信息可以作为一种特征, 提取方法为用 find 命令在样本中查找 js 脚本文件, 将找到的文件名作为特征。

3.1.5 字符串信息特征

字符串信息特征。安卓恶意应用中的字符串可

能包含网络 http 请求信息和一些关键数据, 这些字符串可以是一种很好的特征资源。提取方法为针对每一个样本, 查找出样本里的所有 smali 文件, 然后对于每个 smali 文件用正则匹配匹配出所有字符串信息, 提取出字符串后建立 n-gram 字符序列作为特征。

3.1.6 签名信息特征

安卓的 APK 签名保证了每个应用程序开发商合法 ID, 防止部分开放商可能通过使用相同的 Package Name 来混淆替换已经安装的程序和一些恶意开发者的重打包。所以提取签名的 MD5 码或 SHA1 码作为一种特征。

3.2 动态特征提取

静态分析的优点是快速、高效。但是难以应对代码混淆和多态变形技术。本文采用动态和静态分析结合的方法。动态分析模块主要使用工具 MobSF 来完成, 本文在 virtualbox 上运行安卓系统沙盒并分析样本, 然后使用 MobSF 工具提取了系统调用特征、关键路径和数据访问特征、http 请求特征和恶意吸费特征。

3.2.1 系统调用特征

安卓动态运行过程中也会执行一些系统调用, 这些调用可能包含敏感的数据和 API, 这里将这些系统调用记录并作为特征。

3.2.2 命令执行特征

Android 系统基于 Linux 内核, 同样存在一些敏感路径, 恶意代码可以调用该目录下的系统程序执行命令。如果调用了 chmod、rm 更改文件权限、删除文件等命令, 对操作系统本身将造成极大的威胁。所以命令执行可以作为一种特征。本实验提取出的部分命令执行如表 1 所示。

表 1 应用程序可能执行的部分命令

Table 1 Some commands that the application may execute

命令	描述
chmod	更改文件权限
su/sudo	改变用户或以 root 方式执行
reboot	重启
ps	进程信息
mount	挂载文件
pm install	下载额外包
rm	删除文件
curl	尝试 http 请求
mkdir	创建目录
mv	移动文件或目录

3.2.3 HTTP 请求特征

有的恶意软件会在运行过程中会发送 http 请求包。某些恶意应用会通过 http 请求下载大流量文件造成用户流量损失, 或者通过访问其他网站获取控制命令等信息, 因此可以分析 http 请求并提取特征。

在解析 http 请求提取的属性中, 根据经验, 请求内容、URI 和 Cookie 是恶意特征最重要的体现部位, 所以应该进行更深入的特征提取, 本文提取出 http 请求特征后再对其用正则匹配提取这几种属性的特征。作为一种待融合的特征类。

3.2.4 恶意吸费特征

安卓恶意吸费行为主要通过发短信和打电话来完成。虽然这些行为都会申请特定的权限, 但不是所有申请了发短信和打电话的应用都是恶意应用, 所以记录程序的短信发送、电话拨打行为, 如果这些电话不在运营商之列则是真正的恶意行为。

静态分析的时候, 如果发现程序申请了拨打电话的权限, 是不足以判定该程序就有恶意行为的, 这里将这些行为记录作为一种特征, 对前面的静态特征的权限部分可能产生的误查起到了补偿和相互配合的作用。

3.3 基于信息增益的特征筛选方法

提取出特征之后, 本文先使用信息增益方法筛选特征。信息增益是是一种用数学来度量信息的方法。此方法根据词所归属于类别的信息量来判断特征的有用程度, 即这个词包含的类别的信息量越大, 分类的准确率也就越高。信息增益可以有效反映特征的有用程度, 但信息增益有一个缺点, 就是它会偏袒取值更多的特征, 但实际上并不是取值越多这个特征就越有用。信息增益率在信息增益的基础上做了一些改进, 通过增加一个分裂信息(splitinformation)来惩罚值更多的属性。本文使用了信息增益率来筛选特征。

本文针对前文静态分析和动态分析提取出来的每个特征计算信息增益率后, 进行排序, 然后选取前 k 个特征进行后续的融合; 其中, k 为设定的筛选量。做特征选择, 目的就是选出对分类最有帮助的特征项。但是交给计算机去处理的话, 需要量化。因此如何选出这最有帮助的, 就出现了种种方法。一般来说, 选择特征 k 值在 3000 的时候, 总体效益是很不错的, 再往上涨, 占用空间增大, 但是结果增长并不明显。

4 基于 Simhash 的高维特征融合方法

使用信息增益方法提取出的特征在维度上有所

降低, 但类别还是太多, 并且维度上的差别还是偏大, 本文提出了一种基于Simhash算法的特征融合降维方法, 将特征再次降维的同时融合了所有特征。

4.1 局部敏感哈希算法和 Simhash 算法

LSH 是 Locality Sensitive Hashing 的缩写, 也翻译为局部敏感哈希, 是用来处理海量文本数据的相似度度量的, 因为从文本中提取的特征维度会很高, 需要给特征降维, 而数据的维数在某种程度上能反映其信息量, 一般来说维数越多, 其反映的信息量就越大, 就能越准确地判定两个东西的相似性, 所以, 降维就在某种程度上造成的信息的丢失, LSH 算法基于一个假设, 如果两个文本在原有的数据空间是相似的, 在经过哈希之后他们的相似性仍然存在, 这也是本文使用这种方法来做特征降维的合理性解释。

Simhash 是 LSH 的一种, 是 google 用来处理海量文本去重的算法, 可以计算两个文本的相似度, 将数据降维到 hash 数字, 使计算量变小, 速度加快, 适合在特征深度处理阶段处理维度高的样本。

4.2 基于 Simhash 的特征融合方法

本文基于 Simhash 算法, 提出了一种特征降维和融合方法。Simhash 分为两个部分, 文本哈希和相似度计算, 本文利用 Simhash 的文本哈希算法部分来进行特征映射和降维, 对上述提取的特征进行特征元素的映射, 映射到低维特征空间, 从而得到最终融合后的特征向量。其中特征深度处理和融合算法如下:

(1) 输入:原始特征向量集, 包括静态和动态特征向量和特征向量的权重, 组成特征/权重对(feature, weight), 其中权重通过统计次数确定。

(2)将特征/权重对(feature, weight) 映射到哈希/权重对(hash(feature), weight), 即对于每个特征(feature)映射到哈希特征(hash(feature)), 生成图中的(hash, weight)对, 此时假设 hash 生成的位数 bits_count=6(如图2所示); 为了解决特征的不平衡性, 防止大维度特征将小维度特征淹没, 对于所有特征, 其特征权重计算公式如公式 1。其中 times(feature)为根据每个特征出现的频次, k 为需要调节的参数, $\frac{\max FeatureDimension}{FeatureDimension}$ 为特征维度最高的特征为度的比例, 而开根号是为了降低这种比例, 因为如果特征只有 20 维, 这里的权重比达到了上百, 开根号可以减少这种比例, 不至于Simhash计算时太过于考虑小维度特征。这个公式的主要思想为: 针对小特征维度, 为其设置更高的权重。

$$w = k * times_{feature} * \sqrt{\frac{\max FeatureDimension}{FeatureDimension}} \quad (1)$$

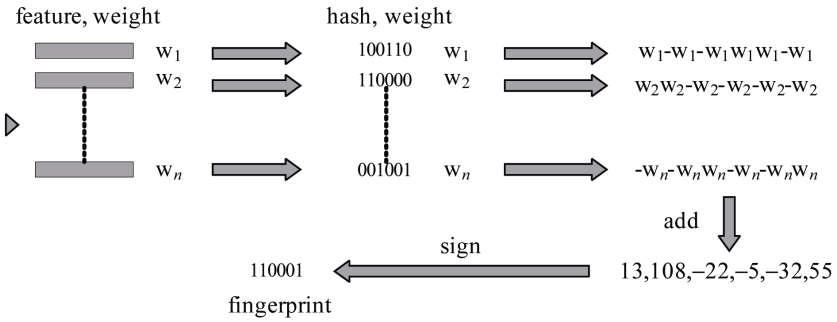

图 2 Simhash 特征融合方法

Figure 2 Simhash feature fusion method

(3) 对哈希/权重对(hash(feature), weight)进行位的纵向累加, 如果该位是 1, 则+weight, 如果是 0, 则-weight, 最后生成 bits_count 个数字, 如图所示是[13, 108, –22, –5, –32, 55], 这里产生的值和 hash 函数所用的算法相关。

(4) 对最后的数字进行 0 和 1 的映射, [13, 108, –22, –5, –32, 55]->110001, 根据获得的数字序列按照正 1 负 0 的原则得到特征深度处理和融合后的特征向量。

使用此特征融合方法测试特征, 设现有三个样本, 样本分别有特征1~20, 特征及权重表如表2所示, 其中样本1与样本2的特征十分相似, 前十个特征的权重都基本。

当hash位数为20时, 三个样本的特征hash值为: 01101010001001101101, 01101000011000101101, 11110110111110100010, 可以看到样本 1 和样本 2 的哈希值非常相似, 他们的海明距离(不同位数)为 3, 而样本 1 与样本 3 的海民距离为 13, 当 hash 位数位 10 位时, 即特征从 20 维降维到了 10 维, 这时三个样本的哈希值为: 1001101101, 1000101101, 1110100010,

表 2 测试样本特征

Table 2 Test sample characteristics

样本	(特征, 权重)
样本 1	[('1', 2), ('2', 2),('3', 2),('4', 2),('5', 2),('6', 2),('7', 2),('8', 2),('9', 2),('10', 0),('11', 0),('12', 0),('13', 0),('14', 0),('15', 0),('16', 0),('17', 0),('18', 0),('19', 0),('20', 0)]
样本 2	[('1', 3), ('2', 4),('3', 2),('4', 2),('5', 2),('6', 2),('7', 2),('8', 2),('9', 2),('10', 0),('11', 0),('12', 0),('13', 0),('14', 0),('15', 0),('16', 0),('17', 0),('18', 0),('19', 0),('20', 0)]
样本 3	[('1', 0), ('2', 0),('3', 0),('4', 0),('5', 0),('6', 0),('7', 0),('8', 0),('9', 0),('10', 2),('11', 2),('12', 2),('13', 2),('14', 2),('15', 2),('16', 2),('17', 2),('18', 2),('19', 2),('20', 2)]

此时样本 1 与样本 2 的海明距离只有 1, 样本 1 与样本 3 的海明距离为 7。在 hash 位数位 6 位时, 三个样本的 hash 值为 101101, 101101, 100010, 可以看到此时样本 1 与样本 2 的哈希值完全相同了, 从 20 位降维到了 6 位时, 其不同的部分信息丢失了。但整体上的相似还是可以表达出来。

5 实验与分析

经过前面对实现 Android 应用恶意代码检测的研究分析后, 本章从网上收集到了大量安卓恶意样本, 从这些样本中提取特征并分类检测, 并做了不同的实验以验证本文提出的方法的有效性。

5.1 实验数据与环境说明

本文的样本搜集渠道为, 恶意样本从 http://contagiominidump.blogspot.jp 搜集, 共 1500 个恶意样本, 所有样本的解压密码需要给作者发送邮件获取。良性样本从 https://www.secsilo.com 搜集, 同样搜集了 1500 个良性样本。将恶意样本和良性样本记录样本名并随机分开, 其中将 80%数据用做训练集样本, 剩余 20%的良性样本和恶意样本组成测试集样本。

本实验的实验环境为 Macbook Unix 内核, 2.7 GHz Intel Core i5 CPU, 内存为 8 GB 1867 MHz DDR3。

5.2 实验结果与对比分析

本实验将 1500 个恶意样本和 1500 个良性样本随机分开, 80%数据用做训练集样本, 剩余 20%的良性样本和恶意样本组成测试集, 首先对样本提取特征, 本文提取的所有特征及特征维度如表 3 所示。

首先为了验证信息增益的方法的有效性, 分别在使用信息增益方法和在不使用信息增益方法的情况下实验, 直接使用 Simhash 特征融合, 并输入 GBDT 算法, 分类时间在 80s 左右, 分类的准确率为 93%。

表 3 本实验提取的特征及维度

Table 3 The features and dimensions extracted in the experiment

特征	维度
权限特征	180
组件特征	4000
函数调用和 API 调用	200000
文件结构	500000
脚本信息	5000
字符串信息	800000
签名信息	3000
系统调用	120
命令执行	20
HTTP 请求	5000
恶意吸费	2500

在使用信息增益特征筛选时, 针对于每一种特征, 如果特征维度大于 3000 则使用信息增益方法筛选, 并将筛选出来的特征在 GBDT 分类算法下, 在不使用信息增益和 Simhash 降维融合方法、仅使用信息增益方法, 仅使用 Simhash 降维融合方法和同时使用两种方法时做分类对比。如表 4 所示, 在两种方法都不使用时, 分类时间大概在 10 分钟左右, 分类的准确率为 95%; 只使用信息增益方法时, 分类时间为 1 分钟左右, 准确率有所提高, 只使用 Simhash 特征融合时, 分类时间只需要 2s, 但准确率只有 94%, 在两者都使用时, 时间和准确率达到了最优。从表的数据可以得出结论如下结论:

表 4 分类时间和准确率

Table 4 Classification time and accuracy

特征	时间	准确率
不使用信息增益和 Simhash 降维	600s	95%
仅使用信息增益特征筛选	58s	98%
仅使用 Simhash 降维融合	2s	94%
使用信息增益和 Simhash 降维融合方法	2s	97%

(1) 在不使用信息增益提取特征时, 由于从样本中提取出的字符串、函数和 API 调用序列等特征维度过大, 而像权限, 命令执行这种特征维度较小, 大维度的特征会将小维度的特征淹没, 小维度特征就不起作用了。而且特征维度过多, 所以分类时间过慢;

(2)只使用信息增益方法时, 过滤了无用特征, 在一定程度上平衡了特征集, 检测准确率有所提高,

但分类时间没有达到最快。

(3)直接使用 Simhash 特征融合方法时面临着和结论 1 一样的问题, 即在 128 位的哈希位数中, 大维度特征的比重占用太多, 小维度特征所占比重太少, 使小维度特征不起作用, 这时候的分类时间很快, 但分类准确率并没有达到最高。

(4) 使用信息增益方法和 Simhash 特征融合方法组合后, 为小维度特征设置更高的权重。这样即没有丢失大维度特征的信息, 也没有淹没小维度的特征, 分类时间和效率都达到了最高。

本实验还测试了 Simhash 分类时的不同哈希桶位数下的准确率和分类时间, 其中准确率如图 3 所示。

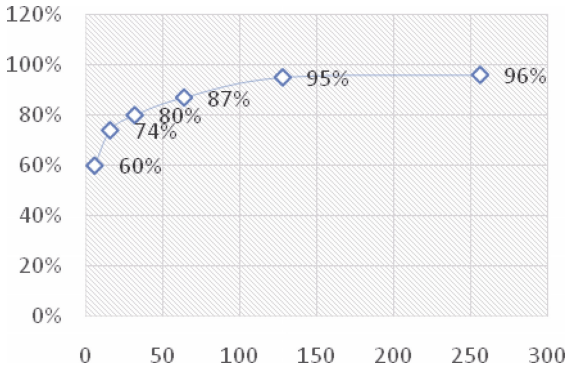

图 3 不同 Simhash 哈希桶位数的分类准确率

Figure 3 Classification accuracy of different Simhash hash buckets

在哈希桶大小为 6 位时, 即为将 3000 维度的特征降维到了 6 位的特征, 无疑失去了大量的准确率, 这时候准确度在 55%左右, 位数大小为 256 位时, 准确率能达到 96%。从数据上可以看出, 特征融合降维后分类的准确率会有所降低, 但随着融合的哈希桶位数的升高, 分类的准确率不断上升, 最后无限逼近不做降维融合的准确率。根据实验可以得出结论, 当哈希位数为 128 位左右时, 本文提出的特征融合的方法可以达到效率和准确度都不错的效果。

虽然在准确率上有所下降, 但在分类时间上, 使用此方法能大大降低, 图 4 表示了不同位数大小下的分类时间, 其中桶大小为 6 位时, 只需 0.5 秒左右就能完成分类, 而哈希位数上升时, 分类的时间缓慢上升, 但在哈希位数为 256 位时, 分类时间也只达到了 3.5 秒, 所以此特征融合方法在分类效率上能有比较大的提升。

本文提出的特征融合算法, 在解决特征的不平衡性上做了特殊的处理, 根据公式(1), 这里的 featureDimension 即为表 3 的特征维度, maxfeatureDimension 为 3000, 本实验特别针对使用公式 1 和直接使用频次作为权重的两种情况, 在哈希桶为 128 位时做实验, 实验数据如表 5 所示。实验数据证明本文提出的公式具有良好的准确率提升效果。

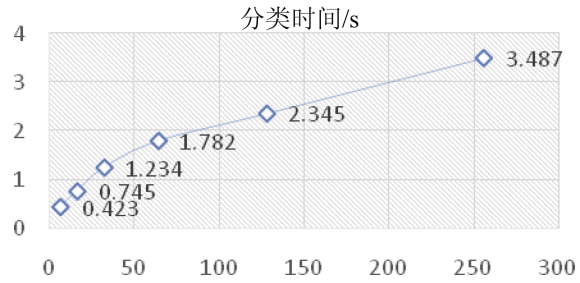

图 4 不同 Simhash 哈希桶位数的分类时间性能

Figure 4 Classification time performance of different Simhash hash buckets

表 5 分类时间和准确率

Table 5 Classification time and accuracy

特征	准确率
使用本文提出的权重计算方法	97.5%
直接使用频次作为权重	95.3%

本实验还针对两种不同的分类算法做比较, 如表 6 所示。分别对 6、16、32、64、128 位的 Simhash 特征融合位数做比较, 可以看出, 在特征维度较低时, GBDT 的分类准确率比随机森林要高, 而随机森林的分类时间比 GBDT 短。随着位数增加, 两种分类算法的准确度都大幅提升。在分类时间上, GBDT 增加的幅度比随机森林要大, 随机森林的时间虽然有所增加, 但一直保持在 0.2s 左右。

表 6 GBDT 和随机森林分类时间和准确率

Table 6 GBDT and random forest classification time and accuracy

算法和 Simhash 特征融合位数	时间	准确率
GBDT, 6 位	0.42275s	60%
随机森林, 6 位	0.109556s	57%
GBDT, 16 位	0.73423s	74%
随机森林, 16 位	0.19311s	82%
GBDT, 32 位	1.25644s	80%
随机森林, 32 位	0.20233s	78%
GBDT, 64 位	1.76351s	87%
随机森林, 64 位	0.21122s	84%
GBDT, 128 位	2.13213s	95%
随机森林, 128 位	0.240123	90%

实验可以得出结论, 首先因为特征融合部分 Simhash 算法已经将所有特征归一化, 所以两种特征出现的过拟合现象都不明显, 由于随机森林的并行

化特性, 在效率上一直优于 GBDT, 而且对于不同的哈希位数, 随机森林的分类时间都很短。而 GBDT 在大维度特征上时间消耗会比较多。但虽然在时间上随机森林取得了比较大的优势, 在准确率和性能消耗上随机森林并没有 GBDT 表现优异。

6 结论

对于安卓恶意应用的检测, 现有的方法存在着检测效率低, 检测率低的问题, 针对这些问题, 本文提出了一种基于多特征融合的安卓恶意应用程序检测方法。检测方法首先对安卓应用样本进行反编译, 获得反编译文件; 然后从反编译文件中提取出大量特征, 提取出的特征使用信息增益方法筛选特征。在不同特征的融合方面, 本文提出了基于Simhash的特征融合方法, 将高维特征映射到低维特征空间, 从而得到融合后的特征向量, 最后利用不同的机器学习分类算法对样本进行分类并得到了比较好的效率和准确率。本文的主要工作总结如下:

(1) 针对安卓恶意行为, 提取出了大量的静态和动态特征, 在提取静态特征的函数调用特征时, 将函数调用中没有 API 调用的部分剔除, 组合了函数调用特征和 API 调用特征, 减少了自定义函数的干扰。使用大量的多维度特征分类, 首先保证了分类的准确度。

(2)从安卓应用中提取的特征通常不同种类的维度相差太大, 这与不同特征的不同来源相关, 无法通过重采样等方法解决, 为了解决大维度特征将小维度特征淹没的问题, 首先在进行特征融合之前, 先对大维度特征进行特征筛选, 筛选时, 针对每个特征计算信息增益, 继而获得信息增益率, 根据信息增益率进行从大到小的特征排序, 选取前 3000 个特征。

(3) 本文借鉴 Google 处理海量文本相似度的原理, 利用 Simhash 方法的局部敏感特性, 提出一种适用于恶意代码分类的高维特征融合方法, 进行特征融合与深度处理, 用 Simhash 融合特征时, 对于特征维度较少的特征, 为其设置较高的权重, 这样就使得小维度的特征获得了更多的哈希位数的占比, 达到了选取了最有用的特征并高效融合处理的目的, 并且平衡了各种特征的贡献。实验数据得出本方法获得了较好的时间性能和分类准确率的提升。

在Simhash的权重问题上, 目前是针对所有特征做权重上的整体调整, 没有具体为每类特征调节权重, 之后考虑加入其他获取权重的方法。另外, 本文提取出来的特征可以用作聚类, 这也是今后需要进一步研究的工作。

致 谢 本文受国家重点研发计划项目(2016YFB0801304)资助。

参考文献

[1] 刘阳. 应用随机森林与神经网络算法检测与分析 Android 应用恶意代码[D]. 北京交通大学,2015.

[2] 刘晓明. 基于 KNN 算法的 Android 应用异常检测技术研究[D]. 北京交通大学,2016.

[3] 蒋志伟. 基于 Ether 的恶意软件动态分析[D]. 山东大学,2013.

[4] Étienne Payet, Fausto Spoto. Static Analysis of Android Programs[J]. *Computer Science*, 2011, V7: 439-445

[5] Kui Luo. Using static analysis on Android applications to identify private information leaks[D], Kansas: Kansas State University, 2011

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

[7] Gorla A, Tavecchia I, Gross F, et al. Checking app behavior against app descriptions[C] //Proceedings of the 36th International Conference on Software Engineering. ACM, 2014: 1025-1035.

[8] Avdiienko V, Kuznetsov K, Gorla A, et al. Mining apps for abnormal usage of sensitive data[C] //2015 International Conference of Software Engineering(ICSE). 2015

[9] 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: 15-27.

王勇 于2003年在北京理工大学计算机应用技术专业获得工学博士学位。现任北京理工大学计算机学院副教授。研究领域为网络空间安全。研究兴趣包括: 恶意代码分析、访问控制。Email: wangyong@bit.edu.cn

蔡建宇 于2017年在北京理工大学软件工程专业获得工程硕士学位。研究领域为网络空间安全。研究兴趣包括: 安卓恶意程序检测。Email: 625473164@qq.com

孟春 于 2002 年在北京理工大学获得硕士学位。现任北京理工大学计算机学院助理研究员。研究领域为网络空间安全。研究兴趣包括: 工控网络安全、身份认证与授权管理。Email: mengch@bit.edu.cn

薛静锋 于 2003 年在北京理工大学计算机应用技术专业获得工学博士学位。现任北京理工大学计算机学院教授。研究领域为网络空间安全。研究兴趣包括: 软件安全、漏洞分析。Email: xuejf@bit.edu.cn

刘振岩 于 2016 年在中国科学院计算技术研究所计算机科学与技术专业获得工学博士学位。现任北京理工大学计算机学院讲师。研究领域为网络空间安全。研究兴趣包括: 大数据安全、人工智能。Email: zhenyanliu@bit.edu.cn