doi:10.11959/j.issn.1000-436x.2017095

基于 APK 签名信息反馈的 Android 恶意应用检测

刘新宇，翁健，张悦，冯丙文，翁嘉思
（暨南大学信息科学技术学院，广东 广州 510632）

摘 要：提出一种新的基于 APK 签名信息反馈的 Android 恶意应用检测方法（SigFeedback）。该方法在 SVM 分类算法的基础上采用启发式规则学习的方式对特征值进行提取，并对检测集中的 APK 签名信息进行验证筛选，实现了启发式反馈，达到更加准确地检测恶意应用的目的。SigFeedback 检测算法具有检测率高、误报率低的特点。最后通过实验显示 SigFeedback 算法具有较高的效率，且能使误报率从 13%降低到 3%。

关键词：误报率；恶意应用；启发式学习；有效性；检测率
中图分类号：TP309.1 **文献标识码**：A

Android malware detection based on APK signature information feedback

LIU Xin-yu, WENG Jian, ZHANG Yue, FENG Bing-wen, WENG Jia-si
(College of Information Science and Technology, Jinan University, Guangzhou 510632, China)

Abstract: A new malware detection method based on APK signature of information feedback (SigFeedback) was proposed. Based on SVM classification algorithm, the method of eigenvalue extraction adoped heuristic rule learning to sig APK information verify screening, and it also implemented the heuristic feedback, from which achieved the purpose of more accurate detection of malicious software. SigFeedback detection algorithm enjoyed the advantage of the high detection rate and low false positive rate. Finally the experiment show that the SigFeedback algorithm has high efficiency, making the rate of false positive from 13% down to 3%.

Key words: false positive rate, malicious application, heuristic learning, effectiveness, detection rate

1 引言

随着智能手机市场的迅速发展，智能手机应用范围更加广泛。目前，智能手机市场中大多以 IOS、Windows Phone 和 Android 操作系统为主，其中，Android 系统受众最广泛。由于 IOS 和 Windows Phone 操作系统平台不开源，存在产品硬件兼容性有限和应用程序控制机制严密的问题。全球最具权威的 IT 顾问咨询公司 Gartner 数据显示，到 2016 年底，Android 拥有超过 80%的销售市场，占据智能手机主导地位的平台。最近，原最大手机厂商 Nokia 也将独立于微软，开始着手 Android 智能手机的生产。

然而，由于开放的安卓应用市场和松散的审核机制，Android 恶意应用泛滥成灾，潜在的隐私安全威胁与日俱增。2015 年 8 月，腾讯在手机安全报告中提出：信息安全问题直接影响了智能终端的发展，并导致了移动恶意代码数量的迅速增长，进而威胁到了用户的个人隐私。另外，Google 在 2015 年也提到：在 Google Play 潜在的恶意应用中，大多数恶意应用因为偷窃用户数据隐私被发现，而其他潜在的恶意应用被发现的数量总数也不到它的一半。此外，在 Google 第三方应用程序中潜在偷窃用户隐私相比 2014 年增长 2 倍，恶意下载量相

收稿日期：2016-12-28；**修回日期**：2017-03-09
基金项目：国家自然科学基金资助项目（No.61133014, No.61272413, No.61373158, No.61472165）；广东省应用型科技研发专项基金资助项目（No.2016B010124009）
Foundation Items: The National Natural Science Foundation of China (No.61133014, No.61272413, No.61373158, No.61472165), Key Program for Guangdong Province Applied Science and Technology R&D Special Funds (No.2016B010124009)

比 2014 年增长 42 倍，权限提升相比 2014 年增长 11 倍。因此，解决 Android 平台的恶意应用检测问题显得格外紧迫。

很多 Android 应用的恶意应用检测都是对 APK 的 Davilk 字节码进行静态分析。在数据流分析框架上，文献[1,2]提出了通过对安卓 APP 的内部组件进行安全审查和利用数据流应用程序接口作为分类特征来检测安卓恶意应用。在动态检测方法中，Cao 等[3]提出了主动检测含蓄性的控制流转换方法，利用已有的工具解决了控制流回调问题并克服了先前未被发现的泄露隐私问题。Qian 等[4]通过加入动态监视器和请求过滤模型实现了行为分析。这些方法都存在正确性验证的问题，因而可能导致较高的误报率。

在协作检测设计上，文伟平等[5]提出了手机客户端和服务器端协作的恶意代码检测方案，但其中得到的 Zero-Day 恶意应用需要人工分析。在检测率验证设计上，出现了大量基于签名算法设计的 Android 恶意应用检测方法。Zheng 等[6]提出了一种基于多级签名的检测方法 DroidAnalytics，通过查表的方式生成签名，并通过判断 2 个应用的相似性来判定是否存在重打包的问题，但检测效率较低。秦中元等[7]提出了多级签名匹配算法的 Android 恶意应用检测方法，通过对每个 APP 进行 API 签名、Method 签名、Class 签名和 APK 签名，最终通过匹配算法寻找相同签名达到检测恶意应用的目的，但操作过程复杂。

鉴于目前各种检测方法存在误报率较高或检测率较低的问题，本文对 Android 平台的恶意应用检测方法进行了研究，在 SVM 分类算法的基础上采用启发式规则学习的方式，进一步提出了基于 APK 签名信息反馈的启发式检测方法。本文方法对启发式检测过程进行了改进，并针对检测集中的 APK 签名信息进行验证筛选，使检测结果的误报率降低到 3%左右，同时检测的准确率提高到 96%。

2　恶意应用检测相关方法

Android 恶意应用检测方法分为静态检测方法和动态检测方法。也有研究者将 Android 检测方法分为基于特征与基于机器学习 2 类。其中，动态检测方法是在无人工干预的前提下，对恶意应用可能存在的恶意行为进行自动化触发。而静态检测方法[8]是对被检测应用进行二进制文件逆向分析，形成汇编程序，通过诸如 dex2jar 和 ApkTool 工具形成 Java 代码或 Smali 指令语言代码，进行特征属性分析，达到检测恶意应用的目的。检测过程通常分为训练阶段和检测阶段。同理，特征集合也相应分为训练集和检测集这 2 部分。

2.1　基于 SVM 分类的检测

定义集合 S[9,10]为包含多种特征集 $S_1, S_2, S_3\cdots$。对于集合 S，定义 *Sizeof*(S)维的向量空间，每一维是 0 或 1，如式(1)所示。

$$\boldsymbol{\varphi}: x \to \{0,1\}^{|S|} \tag{1}$$

对于每一个应用 x 匹配的此类空间，本文构造出向量 $\boldsymbol{\varphi}(x)$，形式化定义如式(2)所示。

$$\boldsymbol{\varphi}(x) \Big| \to I(x,s),\ s \in S \tag{2}$$

对于每一个特征值，应用 x 具有该特征值，则对应的维度值为 1，否则置为 0。形式化定义如式(3)所示。

$$I(x,s) = \begin{cases} 1, s \subset x \\ 0, s \not\subset x \end{cases} \tag{3}$$

由于检测的结果分为良性和恶意 2 类，而 SVM 算法[16]对于二值检测比较适合。本文针对的是 Android 软件检测恰好也是二类分类，且基于 APK 签名信息反馈的 Android 恶意应用检测方法（SigFeedback）能将 SVM 分类算法很好地融入检测模型当中，并通过实验证明了其有效性，故本文采用此算法构造的分类器。对于提取出的特征集 S，SVM 学习算法是通过自动构造检测规则来分辨良性和恶意。用 SVM 这种机器学习方法构成的分类器可以模拟 Android 应用的行为，区分良性与恶意应用。

下面介绍利用 SVM 分类方法来主动分离恶意和良性应用软件的判定过程。SVM 方法基本思想是通过最大边缘化的超平面分层划分来得到 2 类训练集数据，如图 1 所示，w 是|S|维的实数空间上指定方向的超平面空间。

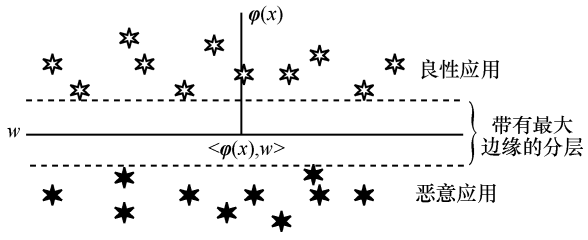

图 1　SVM 设计描述

形式上定义相应的检测函数 f[14,15]，如式(4)所示。

$$f(x)=\langle \boldsymbol{\varphi}(x), w\rangle=\sum_{s\in S}(I(x,s)w_s) \tag{4}$$

其中，w_s 为每个 s 属于 S 时的超平面。对于给定的门槛值 t，如果 $f(x)>t$ 表示为恶意应用，而 $f(x)\leqslant t$ 就表示为良性应用。

2.2 启发式算法

启发式算法[12]是指一个基于直观或经验构造的算法，在可接受的计算时间和空间下给出待解决组合优化问题的每一个实例的一个可行解，该可行解与最优解的偏离程度一般不能被预计。常见启发式方法包括 *A**算法[13]和贪婪最佳优先搜索[14]。

在启发式算法中，*A**算法是个重大突破。因为它引入了启发式估价函数，从无目标搜索转化到有目标的搜索。对于这类搜索问题，关键是寻找到一个最佳估值函数。估值函数表示从当前点出发到目标点的花费。*A**算法是为了达到最高效率，搜索过程将严格沿着最短路径的方式进行。此算法类似于分支线界法，都是在穷举的基础上优化搜索，让每次搜索都更接近目标。

2.3 APK 签名验证方法

为了降低启发式算法的检测误报率，常采用白名单验证方法和 APK 签名验证方法。

白名单验证方法是要求检测者列举出启发式检测过程中检测的每一种检测结果，结果包括区分的判断结果以及每一次检测的恶意应用代号。这种验证方式在一定程度上能消除高误报率的情形。

APK 签名验证方法是通过数字签名来标识应用程序的作者在应用程序之间建立的信任关系。APK 签名包括 APK 信息签名和 APK 对应的文件 MD5 值。其中，APK 信息签名是指最终上传 APK 作者信息的一个签名，APK 签名相当于程序的身份识别码，一般用于程序编译打包之后的身份验证，手机在运行程序之前首先会去验证程序的签名是否合法，只有通过了验证文件才会被运行。

3 Android 应用检测方法与算法实现

3.1 检测模型与方法

本文设计的模型是基于启发式学习的自动化检测方法。为了实现检测方法达到高精度化检测率的目的，系统在设计模型上进行了改造。首先对其学习检测阶段产生的恶意 APK 签名信息集合进行提取，然后与待检测集合的 APK 签名信息进行匹配，并进行验证分析，从而筛选出检测集中一部分的恶意应用。在剩余的检测集中，通过加入基于机器学习的 SVM 分类算法进一步筛选出恶意应用。

具体检测系统基本架构如图 2 所示，主要流程包括动态行为监测、特征库构造、规则库获取、恶意应用检测判定、分析数据结果和 APK 签名验证模块。

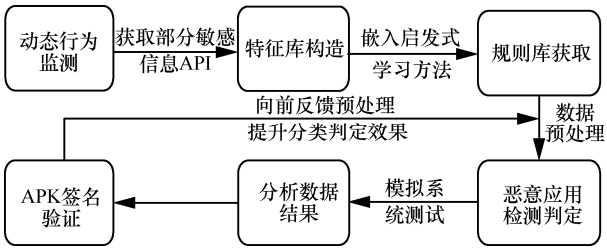

图 2 检测系统基本架构

3.1.1 动态行为监测

每个 APP 都会定义多个监听器，这些监听器会监测调用组件的各种 API 和自定义方法。为了更加准确、快速地获得敏感 API，需要寻找源码中调用这些敏感 API 的入口点，本文采用编写脚本调用监听器的 API 行为变化，最终筛选出敏感 API 接口调用信息和权限请求信息。

3.1.2 特征库构造

特征库主要包括每个 APP 应用的权限请求种类集和 API 调用接口方法集，其中，部分危险权限与敏感性[1,15]如表 1 所示。这些特征反映了恶意应用和良性应用的一些行为，通过这些权限特征值可以分辨出恶意和良性应用。

本文的这些特征库是通过 Python 遍历日志文件，进行字符串匹配，找出敏感行为。通过分析特征集合，选定权重值（如定义 1 所示），在原始数据中获取排名靠前的若干个特征进行矩阵剪裁，构造出恶意应用检测所需的特征集。由于危险权限与敏感 API 对于分辨恶意应用和良性应用更加敏感，其中，隐私类型权限尤为重要，故将权重设置了相应的倍数值。根据权限的影响程度并结合初步实验，将隐私类型权限权重倍数值设置为 20，其他设置为 10。

定义 1 设良性和恶意 APK 总数分别为 N_{b} 和 N_{m}，则权重 *Weight* 如式(5)所示。

$$Weight=\frac{\left|N_{\mathrm{b}}-N_{\mathrm{m}}\right|}{N_{\mathrm{b}}+N_{\mathrm{m}}} \tag{5}$$

表 1　　部分危险权限与敏感 API

权限属性/敏感 API	说明	类型	权重倍数
INTERNET	访问互联网	系统权限	10
READ_PHONE_STATE	访问电话状态	隐私权限	20
ACCESS_WIFI_STATE	访问 Wi-Fi 网络信息	隐私权限	20
READ_SMS	访问手机或 SIM 卡中的 SMS 信息	隐私权限	20
WRITE_SMS	编辑短信或彩信	隐私权限	20
SEND_SMS	发送 SMS 短信	付费/隐私 API	20
RECEIVE_SMS	接收 SMS 短信	隐私权限	20
ACCESS_COARSE_LOCATION	访问 CellID 或 Wi-Fi 获取粗略位置	隐私权限	20
GET_LAST_KNOWN_LOCATION	获取最近位置	隐私 API	10
READ_CONTACTS	访问用户手机所有联系人信息	隐私权限	20
ACCESS_FINEE_LOCATION	访问精度的（GPS）位置	隐私权限	20
CALL_PHONE	直接拨打电话号码	付费	10
CHANGE_WIFI_STATE	改变 Wi-Fi 连接状态	付费	10
WRITE_CONTACTS	写入用户联系人数据	隐私权限	20
WRITE_APN_SETTINGS	写入 API 设置	隐私权限	20
RESTART_PACKAGES	重新启动其他程序	隐私权限	20
PROCESS_OUTGOING_CALLS	允许其他程序监听和控制电话	付费	10
GET_DEVICEID	获取设备号	隐私 API	10
GET_SIM_SERIAL_NUMBER	获取 SIM 卡的序列号	隐私 API	10

3.1.3　规则库获取与恶意应用检测判定

1) 恶意应用检测流程分析

启发式检测过程总体上分为训练和检测 2 个阶段，本文主要是在规则获取过程中融入了启发式学习的过程，实现了新规则的获取，最终使检测率有了较大提高。

启发式学习的规则获取与检测流程如下，详细流程如图 3 所示，其中，训练和检测阶段在图 3 中做了具体划分。

① 选择所有样本集合进行系统输入。

② 对测试集和训练集分别进行特征提取。

③ 进行训练阶段，提取出特征向量，并定义规则，同时也对检测集进行规则化处理。

④ 进行分类检测过程，在构造的特征规则库的基础上，通过分类模型改进算法（见 3.2.1 节）进行特征向量匹配，分类出恶意和良性 APP。其恶意 APP 加入已检测集中，而良性的 APP 加入待检测集中。

⑤ 新规则库获取过程，如步骤②所示。

⑥ 利用新规则库重新检测待检测集，重复步骤④和步骤⑤直到检测满足相应精度为止。

下面，重点分析提取新规则，其他步骤属于一般恶意应用检测必要过程，本文不做详细展开。

2) 规则库获取

规则库是特征集合的一种数值表现形式，通过权重和所占数目及其之间的关系确定其值，它反映了特征库的每种特征所体现出分辨良性和恶意 APP 应用的量值。获取规则库可借助于启发式算法，通过学习恶意应用检测所需的特征值，不断挖掘出新的规则库，但这些规则之间却存在一些不可估量的缺陷，缺陷主要表现在规则关联度不大。针对此缺陷，本文利用 SVM 分类算法的优势，设计出启发式学习规则过程如图 3 所示。

提取新规则是通过动静态结合方法来形成规则的过程。首先预设定规则库，对于符合规则的应用软件进行静态规则分析。对于不符合规则的应用软件，通过启发式学习，提出新的规则再进行检测。这些新规则包括匹配度规则、平均值规则和敏感值规则。下面具体分析获取新规则的这些方法。

匹配度规则。对于规则特征向量集的每个维度值，赋予一个权重，构造出特征向量集。通过 Zhang 等[16]的图编辑距离算法进行匹配，如果发现匹配结

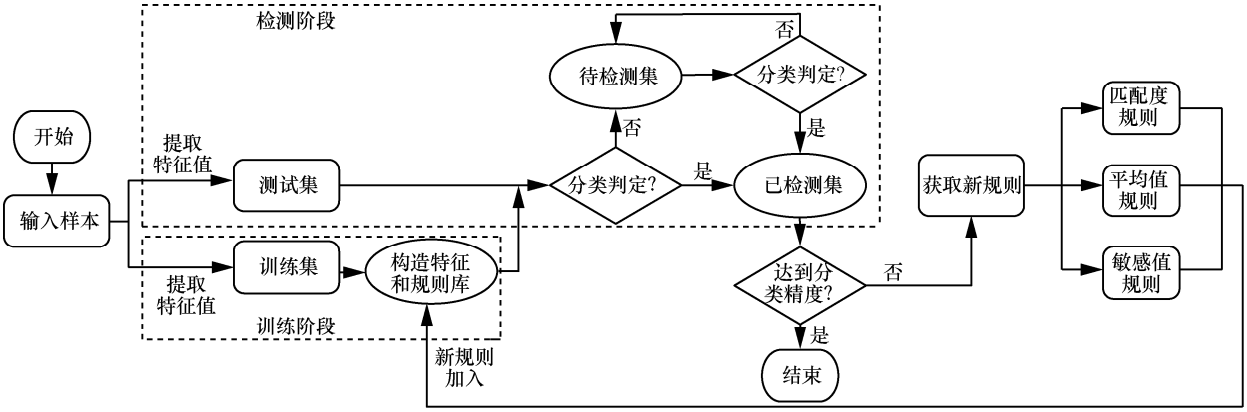

图 3 启发式学习的规则过程

果相似度高，就进行相乘再求平均值的方法获得新规则。具体定义如下，设 2 个向量分别为 $\boldsymbol{\alpha}=(\alpha_1, \alpha_2, \alpha_3,\cdots,\alpha_i,\cdots)$，$\boldsymbol{\beta}=(\beta_1, \beta_2, \beta_3,\cdots, \beta_i,\cdots)$,则产生新的规则 $\boldsymbol{\gamma}=(\gamma_1, \gamma_2, \gamma_3,\cdots, \gamma_i,\cdots)$，其中，$\gamma_i$ 如式(6)所示。

$$\gamma_i = \frac{2\alpha_i\beta_i}{\alpha_i + \beta_i} \tag{6}$$

平均值规则。通过恶意应用库提取出的特征向量集中的某个特征值进行敏感值设定。如果每个恶意 APP 对应的特征向量其某一特征值都非零，将其某一维度特征值求平均，得到新的向量，并加入到新的规则集。具体定义如下，设每个 APP 对应特征向量分别为 $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3, \cdots, \boldsymbol{\lambda}_i, \cdots, \boldsymbol{\lambda}_n$，若第 k 列特征值都不为 0，则 $\lambda_{i,j}$ 如式(7)所示，其中，j 表示列。

$$\lambda_{i,j} = \begin{cases} \dfrac{1}{n}\sum_{i=1}^{n}\lambda_{ij}, j = k \\ \lambda_{ij}, j \neq k \end{cases} \tag{7}$$

敏感值规则。启发式学习过程对某一维向量值特别敏感，就生成一个新的规则集，并修改规则权重参数，使此敏感数据项远大于其他数据项。具体如下，设原第 j 列特征值为 a_j，权重参数 $Weigh_j$ 由 1 变为 p_j(本文实验中 p_j 取值为 10)，则此时的特征值如式(8)所示。

$$a_j' = p_j a_j \tag{8}$$

为了获得更高的检测率，设置的规则需要更加灵活，这样必然会导致一定程度上的误报检测率。对于上述现象，本文提出具体的 APK 签名验证方案来降低误报率。

3.1.4 APK 签名检测方案

APK 签名包括作者签名信息和文件 MD5 值 2 个部分。具体过程如图 4 所示，本文将检测集合分为有限多个集合，分别记为 A，B，C，D，…。具体签名检测过程如下。

定义 2 恶意筛选预处理集是指检测结果判定为恶意集中具有不同 APK 签名的恶意应用构成的集合。

1) 通过启发式学习检测方法对部分待检测集合 A 进行分类，得到判定为恶意的 A_1 部分和判定为良性的 A_2 部分，其中，A_1 为恶意筛选预处理集。

2) 提取 A_1 中每个 APK 的签名信息和文件 MD5 值，先利用文件 MD5 值对检测集合 B 进行判定，值相同的 B_1 直接判定为恶意，再对剩余部分 B_2 进行签名信息判定。如果与恶意集合 A 具有相同作者信息签名，就将其分到集合 B_{21} 中，并设定为恶意性权重较大的标识，即将集合 B_{21} 对应的特征向量集中恶意性判定为较敏感的特征列乘以较大倍数，否则分到集合 B_{22} 中。

3) 继续对 B_{21} 与 B_{22} 进行启发式学习检测，得到判定为恶意的 B_{31} 和良性的 B_{32}，其中，B_{31} 和 A_1 加入恶意集合，而 B_1 与 A_1 已确认其文件 MD5 值相同，故 B_1 不加入恶意筛选预处理集中，但这些都归属于恶意检测结果集合中。

3.2 算法分析与实现

本文检测方案主要是通过启发式学习获得更多新规则，并进行了 SVM 分类检测过程，具体算法如 3.2.1 节描述。对其中的检测集部分进行 APK 签名信息反馈，具体算法如 3.2.2 节描述。

3.2.1 启发式学习的分类模型改进算法

启发式 A^*算法定义如下：$F=G+H$，其中，对于每个点，都有自己的 G、H 和 F。G 表示从特定

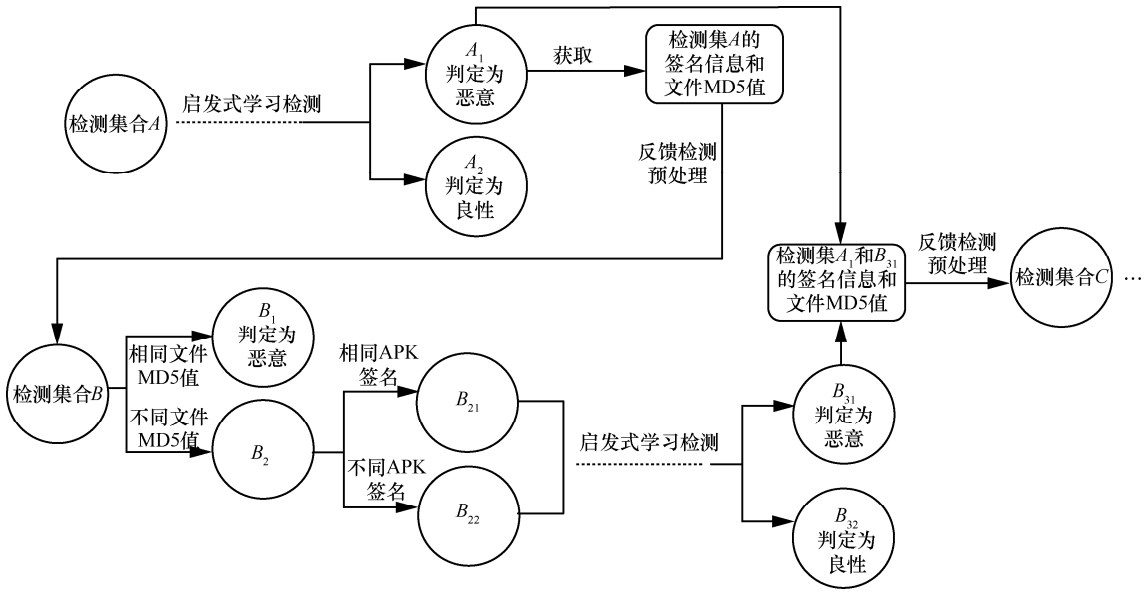

图 4　APK 签名检测过程

的点到起点的距离，H 表示从该点到目标点的估值，那么 F 就是经过该点路径的估值。其中，G 为确定值，F 的大小取决于 H 值。这里 H 定义为曼哈顿距离，即向量每一列的差值的绝对值再求和。

由于恶意应用检测中训练集和检测集数据量大，特征向量集合维度较高，即使某一维特征值很大，也可能与另一种 APK 特征向量差值之和相等。这种情况下，直接进行粗粒度相减的优势不突出，因而采用较细粒度化算法马氏距离[17]来度量，此距离主要通过求 2 个向量之间的协方差的均值来计算。

采用马氏距离是为了提高搜索出的分类向量差别，有利于相似度较高特征向量集进行匹配分析，从而提高其精度。具体算法如算法 1 描述，本文采用以下标注。

Q_d：代表优先队列。

Q_d .push(P)：把元素 P 插入 Q_d 的队尾。

Q_d .pop()：从 Q_d 队头弹出一个元素。

算法 1　启发式学习的分类模型改进

输入

$\boldsymbol{G}$：训练阶段的特征向量集

$\boldsymbol{V}_t$：测试样本的特征向量集

输出　$\boldsymbol{V}_t$ 的分类值为 1 或 0

def　$maxnum$=200, ep = 0.005;

wc 为未初始化的权重系数;

while $v_k \leftarrow K$ 元组 $(v, maxnum)$，其中，$v \in \boldsymbol{G}$

Q_d .push(max $\sqrt{v_k v_k}$);

初始化 $\boldsymbol{G}$ 的列/列维协方差矩阵 $\boldsymbol{S}$，且 $\boldsymbol{G}$ 满足其训练 APK 个数大于特征值个数;

for all　$v_i \in \boldsymbol{G}$ do

　　for all　$v_j \in \boldsymbol{G}$ do

$$d_i = wc_i v_i;$$

$$d_j = wc_i v_j;$$

$$S[i][j] = (v_i - d_i)(v_j - d_j);$$

　　end for

end for

for　$(\boldsymbol{G} : \boldsymbol{x})$　do

for　$(\boldsymbol{V}_t : \boldsymbol{y})$　do

$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^{\mathrm{T}} \boldsymbol{S}^{-1} (\boldsymbol{x} - \boldsymbol{y})}$;

　　if　$d(\boldsymbol{x}, \boldsymbol{y})$ < Q_d 中的任意元素

更新优先队列 Q_d ;

else if　fabs($d(\boldsymbol{x}, \boldsymbol{y}) - Q_d$) $\leqslant ep$

　　结束算法;

　　end if

　　end for

end for

最终利用 Q_d 决定 $\boldsymbol{V}_t$ 的分类;

3.2.2　APK 签名检测方案的实现

此方案通过 APK 签名过程，对检测集合进行反馈预处理，筛选掉一部分确认的恶意应用。并加大判定为恶意应用的相同作者信息的 APK 特征集的权重比例。具体方案如算法 2 描述，本文采用以

下标注。

Q_a：表示从特征集合 S 中恶意和良性样本中选出比例为 a 的样本作为测试集合。

Q_{1-a}：表示从特征集合 S 中恶意和良性样本中选出比例为 $1-a$ 的样本作为训练集合。

Q_m：检测为恶意的集合。

Q_b：检测为良性的集合。

算法 2 APK 签名检测

输入

S：所有 APK 特征集

a：训练集所占比例

输出 检测结果中恶意和良性应用个数 N_{m} 和 N_{b}

def ep=0.000 4, $dim=|S_i|$;

for all $S_i \in random(Q_a)$, $S_j \in random(Q_{1-a})$

对 S_i 进行学习训练,得到结果保存到 *train_out* 集合中;

对测试集合 S_i 通过训练,利用分类算法得到检测结果保存到 *test_out* 中;

for k = 0 to *size* (*test_out*)

if *test_out* (k) == 0 then

Q_m.push (*test_out* (k));

else Q_b.push (*test_out* (k));

end if

end for

利用 Q_m 的 MD5 值筛选检测集 S_i,得到 $S_i = S_i'$;

$Q_m = Q_m \cup (S_i - S_i')$;

对于 S_i 进行 APK 签名信息匹配,对于具有相同签名信息的 APK, 将其特征向量每列都乘以 10;

APK_Sig_Detect (S_i, a);

end for

return N_m=*size*(Q_m), N_{b}=*size*(Q_b);

4 实验分析与评估

4.1 实验环境

本文提出的基于启发式学习的自动化检测方法在 Windows 平台上实现。所有实验均在内存为 8 GB，处理器为 Intel Core i5 3.2 GHz 的机器上完成。批量文件处理采用 Python 脚本，使用的开发平台为 Python2.7.9、Code::Blocks 13.12 和 Matlab R2012b。分类规则库仅考虑权限请求和 API 调用构造的向量集合，本实验对比 SVM 分类算法和 SigFeedback 算法均在此环境下完成。

4.2 实验数据来源

在实验分类器评估检测实验室中，实验数据源总数达 2 621 个，APK 总容量达 20 GB。其中，样本中恶意应用 1 214 个，收集于 VirusShare，这是恶意应用库公认平台；良性应用 1 407 个，均来源于中国安卓市场，通过 Python 脚本实现网页爬虫获得。

4.3 实验步骤

Android 恶意应用检测实验分为以下几个步骤。

1) 通过 Windows 的 shell 脚本，利用 ApkTool 工具实现批量解压 APK，得到相应的解压包。

2) 批量删除每个解压包的部分文件，并保留 Manifest 文件和.RSA 文件。

3) 对于所有 Manifest 文件集，提取并统计出各类权限和组件 API 数目，并对权重值调整进行模拟。

4) 利用 SVM 分类器训练和测试特征集。

5) 对于判定为恶性的应用，通过跟踪每个应用的特征向量寻找到对应的 APK 文件名，从而找到 APK 签名信息，主要包括 APK 的 MD5 值和.RSA 文件中的开发者公钥、所采用的加密算法等信息，并对待检测集中的 APK 签名信息进行筛选预处理。

4.4 检测评估标准

本文将非恶意应用定义为良性元组，恶意应用定义为恶意元组[18,19]。真阳性（TP, true positive）是指分类器将恶意应用正确判定为恶意应用的元组；真阴性（TN, true negative）指分类器将良性应用正确判定为信任应用的元组；假阳性（FN, false negative）指分类器将良性应用错误判定为恶意应用的元组；假阴性（FP, false positive）指分类器将恶意应用错误判定为良性应用的元组。由此，得到以下常用的检测评价指标。

1) 检测率，也称为真阳性率，具体计算如式(9)所示。

$$TPR = \frac{\sum TP}{\sum TP + \sum FN} \tag{9}$$

表示所有恶意样本中正确分类为恶意应用的比例。

2) 准确率（ACC, accuracy），也称为分类精度，具体计算如式(10)所示。

$$ACC = \frac{\sum TP + \sum TN}{\sum TP + \sum FP + \sum TN + \sum FN} \tag{10}$$

表示所有正确分类的样本与所有样本之比。

3) 误报率（FPR, false positive rate），也称为假阳性率，具体计算如式(11)所示。

$$FPR = \frac{\sum FP}{\sum FP + \sum TN} \tag{11}$$

表示所有分类为恶意应用中错误分类为恶意应用的比例。

4.5 算法检测性能分析

本文对比分析 SVM、启发式学习的 SVM（简称 H-SVM）和 SigFeedback 算法，检测性能的受试者工作特征曲线（ROC）如图 5 所示。采用 SVM 分类算法构成的向量集合和 H-SVM 方法得到的检测率都在 90%左右，而利用 SigFeedback 算法检测率达到 95%。通过对比检测性能的 ROC 和检测率结果，表明 SigFeedback 算法在检测恶意应用方面有一定的优势。

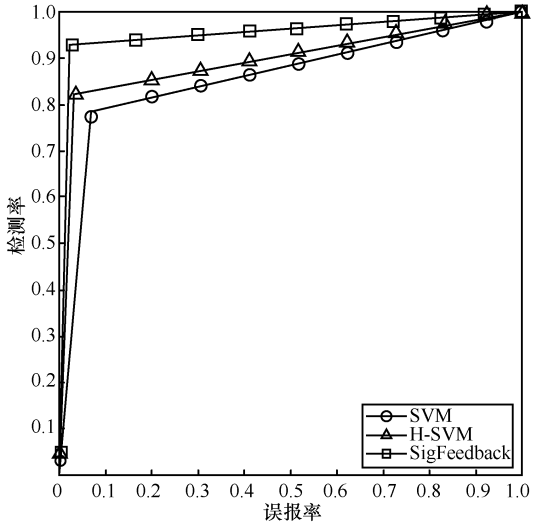

图 5　检测性能的 ROC

4.6 实验数据分析

在本实验中，其核心检测实验过程分为训练和检测 2 个阶段。构造的特征向量中特征值个数为 1 496。本文训练集为随机抽出的 60%的恶意向量组和 60%的良性向量组，测试集为样本库剩余 40%的恶意和良性向量组。因而测试集合中恶意 APK 个数为 486，良性为 563。

通过实验，SigFeedback 检测方法检测出良性测试应用中恶意应用 20 个，良性应用 543 个；恶意测试应用中恶意应用 470 个，良性应用 16 个。其评估标准包括检测率（*TPR*）、准确率（*ACC*）和误报率（*FPR*）。而 SVM 分类方法测试同样的数据集，测得良性测试应用中恶意应用 49 个，良性应用 514 个，测得恶意性测试应用中良性应用 67 个，恶意应用 419 个。

对比分析 SVM 分类和 SigFeedback 检测方法，其检测评估参数对比分析，具体实验对比数据结果如表 2 所示。由表 2 得出，加入 APK 签名验证的启发式自动化检测方法后，检测率和准确率分别提高了 5.15%和 7.62%，误报率降低了 10.5%。

表 2　恶意应用检测的准确率及误报率分析

算法	总测试 APK/个	检测率	准确率	误报率
SVM 分类	1 049	91.29%	88.94%	13.79%
SigFeedback	1 049	96.44%	96.56%	3.29%

5 结束语

本文提出一种基于启发式学习的检测方法进行 Android 应用的规则提取，通过 APK 签名信息验证方式对其进行筛选预处理，从而提高检测率的真实性。在一定程度上解决了检测误报率较高的缺陷，也是检测方式上的一种改良，但检测系统也存在一些缺陷，需要后续改进。

下一步研究工作主要包括如下 3 个方面。

1) 在机器学习过程中，使动态检测方法能够分辨出在恶意应用检测过程中对应的规则。

2) 利用基于语义特征，诸如利用数据依赖图和控制流图分类安卓应用，并将此方法应用到研究对抗不同恶意应用或未知恶意应用中。

3) 对于分类可以考虑依据 APP 的威胁度进行评定，区分恶意性等级，针对恶意级别较低的 APP 考虑手动分析，级别高的直接判定为恶意。

参考文献：

[1] WEI F, ROY S, OU X, et al. Amandroid: a precise and general inter-component data flow analysis framework for security vetting of Android apps[C]// ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014:1329-1341.

[2] 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 & Software Technology, 2016, 75(C):17-25.

[3] CAO Y, FRATANTONIO Y, BIANCHI A, et al. EdgeMiner: Automatically detecting implicit control flow transitions through the Android framework[C]// Network and Distributed System Security Symposium. 2015.

[4] QIAN Q, CAI J, XIE M, et al. Malicious behavior analysis for Android applications[J]. International Journal of Network Security,2016, 18(1):182-192.

[5] 文伟平, 梅瑞, 宁戈, 等. Android 恶意软件检测技术分析和应用研究[J]. 通信学报,2014, 35(8):78-86.

WEN W P, MEI R, NING G, et al. Malware detection technology analysis and applied research of Android platform[J]. Journal on Communications, 2014, 35(8):78-86.

[6] ZHENG M, SUN M, LUI J C S. Droid analytics: a signature based analytic system to collect, extract, analyze and associate Android malware[C]// IEEE International Conference on Trust, Security and Privacy in Computing and Communications. IEEE Computer Society, 2013:163-171.

[7] 秦中元, 王志远, 吴伏宝, 等. 基于多级签名匹配算法的 Android 恶意应用检测[J]. 计算机应用研究, 2016, 33(3):891-895.
QIN Z Y, WANG Z Y, WU F B, et al. Android malware detection base on muti-level signature matching[J]. Application Research of Computer, 2016, 33(3):891-895.

[8] 卿斯汉. Android 安全研究进展[J]. 软件学报,2016, 27(1): 45-71.
QING S H. Research progress on Android security[J]. Journal of Software, 2016, 27(1):45-71.

[9] ARP D, SPREITZENBARTH M, HUBNER M, et al. DREBIN: effective and explainable detection of Android malware in your pocket[C]//Network and Distributed System Security Symposium. 2014.

[10] FAN R E, CHANG K W, HSIEH C J, et al. LIBLINEAR: a library for large linear classification[J]. Journal of Machine Learning Research, 2008, 9(9):1871-1874.

[11] JOACHIMS T. Text categorization with support vector machines: learning with many relevant features[C]//European Conference on Machine Learning. Berlin Heidelberg, 1998: 137-142.

[12] 苗夺谦, 胡桂荣. 知识约简的一种启发式算法[J]. 计算机研究与发展, 1999, 36(6):681-684.
MIAO D Q, HU G R. A heuristic algorithm for deduction of knowledge [J]. Journal of Computer Research & Development, 1999, 36(6):681-684.

[13] HART P E, NILSSON N J, RAPHAEL B. A formal basis for the heuristic determination of minimum cost paths[J]. IEEE Transactions on Systems Science & Cybernetics, 1968, 4(2):100-107.

[14] PEARL J. Heuristics: intelligent search strategies for computer problem solving[M]. Addison-Wesley Pub. Co, 1984.

[15] 杨文. 基于支持向量机的 Android 恶意软件方法研究[D]. 南京: 南京理工大学, 2015.
YANG W. Research of malware detection on Android based on support vector machine[D]. Nanjing: Nanjing University of Science and Technology, 2015.

[16] ZHANG M, DUAN Y, YIN H, et al. Semantics-aware Android malware classification using weighted contextual API dependency Graphs[C]//Computer and Communications Security. ACM, 2014: 1105-1116.

[17] MAESSCHALCK R D, JOUAN-RIMBAUD D, MASSART D L. The Mahalanobis distance[J]. Chemometrics & Intelligent Laboratory Systems, 2000, 50(1):1-18.

[18] 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):12-27.
YANG H, ZHANG Y Q, HU Y P, et al. A malware behavior detection system of Android applications based on multi-class features[J]. Chinese Journal of Computers, 2014, 37(1):12-27.

[19] HANG A, DE LUCA A, HUSSMANN H. I know what you did last week! do you?: dynamic security questions for fallback authentication on smartphones[C]//Conference on Human Factors in Computing Systems. 2015:1383-1392.

作者简介：

刘新宇（1990-），男，湖南衡阳人，暨南大学硕士生，主要研究方向为智能移动端安全与网络安全。

翁健（1976-），男，广东茂名人，博士，暨南大学教授、博士生导师，主要研究方向为密码学与信息安全。

张悦（1990-），男，陕西榆林人，暨南大学博士生，主要研究方向为信息安全与智能移动端安全。

冯丙文（1985-），男，山东东营人，博士，暨南大学讲师，主要研究方向为多媒体安全与数字取证。

翁嘉思（1994-），女，广东汕尾人，暨南大学硕士生，主要研究方向为密码学与云安全。