文章编号:1007-5321(2021)04-0095-07　　**DOI**:10.13190/j.jbupt.2020-259

面向 Android 支付破解应用的检测方法

汤永利，　李星宇，　赵宗渠，　李运峰
(河南理工大学 计算机科学与技术学院，焦作 454003)

摘要：Android 破解应用存在侵犯合法软件权益和传播恶意代码的风险. 为有效检测 Android 平台上的支付破解应用,提出一种基于机器学习的检测方法. 针对反汇编的字节码文件构建了支付语义信息调用控制流和支付数据库操作函数集,通过 *n*-gram 和重复代码子块长度统计方法构造相应特征集,最后构建带决策机制的多分类器检测模型以识别 Android 应用中不同的支付破解行为. 实验结果表明,所提检测方法的模型检测精确率为 85.24%,AUC 值为 0.87,与同类方法相比,对支付破解类应用的检测率有显著提高,有效解决了支付破解应用的检测问题.

关　键　词：Android；支付破解；软件安全；特征提取；机器学习
中图分类号：TP309　　**文献标志码**：A

Detection Method for Android Payment Cracked Application

TANG Yong-li，　LI Xing-yu，　ZHAO Zong-qu，　LI Yun-feng
(School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454003, China)

Abstract: Android cracked applications have the risks of infringing on legitimate software rights and spreading malicious code. To detect the payment cracked applications on Android platform, we propose a detection method based on machine learning. Based on the disassembled bytecode file, the call control flow of payment semantic information and the payment database operation function set are constructed. We use a *n*-gram statistical method and a repeated code sub-block length statistical method to construct the corresponding feature set, and build a multi-classifier detection model with a decision-making mechanism to identify different payment cracked behaviors in Android applications. The experimental results show that the detection accuracy rate of this model is 85.24%, and the area under curve (AUC) value is 0.87. Compared with the baseline methods, the detection rate of payment cracked applications is significantly improved, which effectively solves the detection problem of payment cracked applications.

Key words: Android; payment cracked; software security; feature extraction; machine learning

2019 年 Liftoff 发布的统计报告显示,2018-09—2019-08 期间,Android 软件的下载量达到 1 430 亿,新增 Android 应用总数为 iPhone 操作系统(IOS, iphone operating system)的 3 倍,达到 145 万个,但 IOS 的应用盈利为 Android 的 1.5 倍,造成这种收益反差现象的部分原因在于存在许多 Android 支付破解应用. 支付破解应用在侵犯应用权益的同时也成为恶意软件传播的优良载体. 目前,移动端和个人

收稿日期：2020-12-01
基金项目：国家自然科学基金项目(61802117)；河南省高校科技创新团队项目(20IRTSTHN013)；河南理工大学创新型科研团队项目(T2018-1)
作者简介：汤永利(1972—)，男，教授，硕士生导师.
通信作者：赵宗渠(1974—)，男，讲师，硕士生导师，E-mail:zhaozong_qu@hpu.edu.cn.

计算机(PC,personal computer)端对恶意软件的定义基本相同. 作为新型 Android 恶意软件的支付破解应用,有极大的概率存在恶意插件和更新组件. 更新组件会伴随应用的运行下载安装传统的恶意代码文件.

基于权限和应用程序接口(API, application programming interface)的特征构造方案被广泛用于恶意软件检测. Peiravian 等[1]使用了权限和 API 相结合的二进制特征检测恶意软件. Song 等[2]使用危险等级较高的权限及其对应的 API 作为特征用于检测恶意软件. Aafer 等[3]通过静态分析的方法提取了 5 种不同类型的 API 特征. 许艳萍等[4]选取 Manifest.xml 中申请的权限作为检测特征. 此类方案对恶意软件的抽象程度较低,部分恶意软件可通过申请与良性应用相同的权限和 API 绕过检测. 基于图的特征构造方案弥补了权限和 API 这类特征构造方法的不足. Fan 等[5]构建了应用中敏感 API 的关系图,并提取 5 种数据作为特征,使用 DAPASA(detecting Android piggybacked apps through sensitive subgraph analysis)方法检测恶意软件. Gascon[6]使用函数调用流图作为检测特征. Zhou 等[7]使用函数包依赖关系图作为检测特征. 此类方案考虑了函数、API 和函数包之间的联系,故检测效果明显优于单独使用权限和 API 的特征构造方案. 基于操作码的特征构造方案对恶意软件的检测效果也优于权限和 API 特征方案. Zhang 等[8]首先计算出操作码的 n-gram 值,之后将其分成规模大小相同的数据片用于训练卷积神经网络(CNN,convolutional neural networks). 经过训练的 CNN 用于检测 PC 端的勒索病毒. Li 等[9]统计出应用中各种操作码的数量,并将统计值用二进制表示,使用规模大小相同的矩阵表示每个样本;之后使用操作码的二进制统计值训练 CNN. 经过训练的 CNN 模型用于检测恶意软件. 此类方案使用概率统计值表示代码的语义信息,将代码逻辑抽象为检测特征,方案难以被恶意变种软件绕过. 上述特征构造方案无法直接检测支付破解应用,主要原因在于:①特征构造范围为整个反汇编代码集,而支付破解修改的是一小部分支付代码,以全体代码集作为特征构造的对象降低了检测模型的拟合度,使其无法用于支付破解检测;②支付破解行为同时改变了支付代码集中函数内部的控制逻辑和函数间的调用逻辑,上述特征构造方案只考虑了一种逻辑关系,无法准确地表示出支付破解行为. 为检测这种新型移动端恶意软件,借鉴图特征和操作码特征的提取方法,提出了一种面向 Android 支付破解应用的检测模型.

1 破解分析

支付程序在整个程序中是一个相对独立的程序块. 汇编文件经反汇编得到汇编代码集,黑色方块为独立的支付代码集. 支付代码框架分为接口调用函数集、第 3 方支付接口函数集和数据库操作函数集 3 部分,如图 1 所示. 其中,接口调用函数集用于构造支付订单和支付请求;第 3 方支付接口函数集用于发送支付订单和请求,并依据返回的支付状态码选择相应的支付接口与数据库操作函数;数据库操作函数集用于修改本地数据库中的道具数据,如应用数据库中储存的金币、积分和游戏应用中用户使用的道具数据,并向用户反馈支付结果等交互信息. 针对这种以第 3 方支付接口为核心的支付逻辑框架存在的破解方式主要包括接口调用逻辑破解(简称“逻辑破解”)和数据库写入函数内容破解(简称“内容破解”).

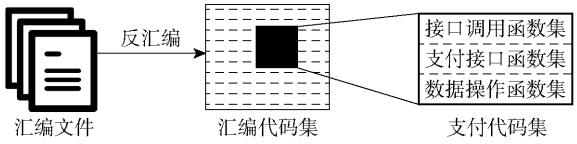


图 1 应用支付函数框架

逻辑破解的修改对象为接口调用函数集与支付接口函数集之间的调用逻辑. 未修改的接口调用函数集会调用支付接口函数集中的成功、失败和异常(取消)3 种支付接口,每种支付接口均对应着不同的数据库操作与信息交互函数. 破解后,接口调用函数与失败和异常 2 种支付接口的调用关系被切断,如图 2 所示. 虽然存在失败与异常支付接口函

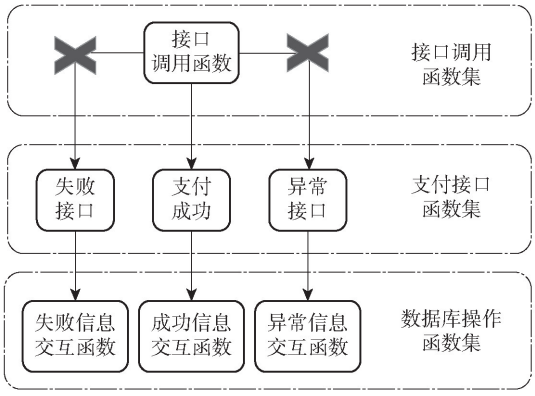


图 2 逻辑破解图

数及其数据库操作函数，但是接口调用函数集不再调用这 2 种支付接口.

内容破解的修改对象为数据库操作函数的函数内容. 未修改的数据库操作函数集中的失败、成功和异常 3 类数据库操作函数均使用了不同的函数内容. 破解者在破解时，往往会对失败和异常 2 种数据库的操作函数做修改、删除和替换操作，如图 3 所示. 经过修改的支付失败数据库操作函数和支付异常数据库操作函数与支付成功数据库操作函数的函数内容基本一致.

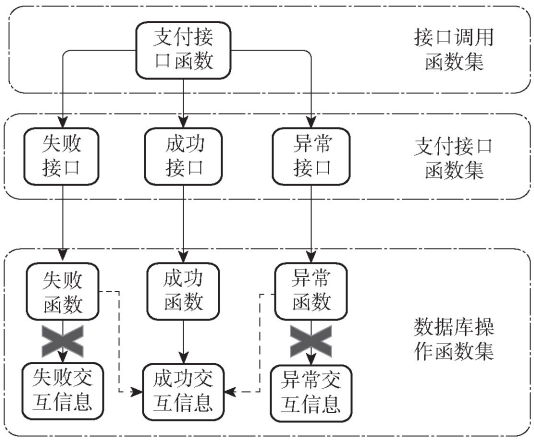


图 3　内容破解图

2　检测模型

逻辑破解的检测特征是一种基于支付调用控制流的操作码 *n*-gram 特征. 内容破解选取重复代码子块的出现频率作为检测特征.

首先从应用文件中提取支付函数集；随后构造支付函数调用控制流和数据库操作汇编代码列表；再从调用控制流和汇编码列表中分别提取操作码特征和重复代码子块特征；最后利用机器学习算法构建检测分类器，用于识别未知的支付破解应用.

图 4 所示为支付破解应用检测模型的工作流程，主要分为 3 个模块.

1）源码提取模块. 使用反汇编工具 Apktools 和 Androgard 分别提取资源文件、调用流列表和控制流文件集，然后以第 3 方支付函数集为中心在调用流列表中向上向下递归查询筛选出支付函数集.

2）特征构造与表示模块. 以支付函数集为基础，分别构造支付函数调用控制流和数据库操作函数汇编代码列表；然后提取调用控制流中的操作码 *n*-gram 值作为特征集合，称为概率特征集，提取数据库操作汇编代码集中的重复代码子块频率统计值作为特征集合，称为频率特征集.

3）模型训练与检测模块. 分别使用 2 种不同的机器学习算法对 2 组特征集中的一部分数据进行模型训练；然后使用 2 个检测模型分别对剩余部分数据进行检测；最后采用决策算法融合 2 组检测值，融合结果作为最终检测结果.

2.1　相关概念与定义

定义 1　调用控制流

调用控制流包含用于描述函数间调用关系的调用流和用于描述函数内部执行流程的函数控制流.

调用流（CG）用有向图 $CG = \langle V, E \rangle$ 来表示. 其中，V 表示图 CG 中所有顶点的集合，每个顶点代表一个函数，每个函数由其所在包名、类名、方法名及其参数组成；E 表示图 CG 中有向边的集合，即 $E \subseteq V \times V$，且 $\forall (u,v) \subseteq V, \exists (u,v) \in E$. 每条有向边代表图 CG 中节点（函数）之间存在的先后调用关系.

控制流（CFG）用有向图 $CFG = \langle V', E' \rangle$ 来表示. 其中，V'表示图 CFG 中所有顶点的集合，每个顶点代表函数中的一条完整语句. E'表示图 CFG 中有向边的集合，即 $E' \subseteq V' \times V'$，且 $\forall (u',v') \subseteq V'$，$\exists (u',v') \in E'$. 每条有向边代表图 CFG 中节点（语句）之间的先后执行顺序.

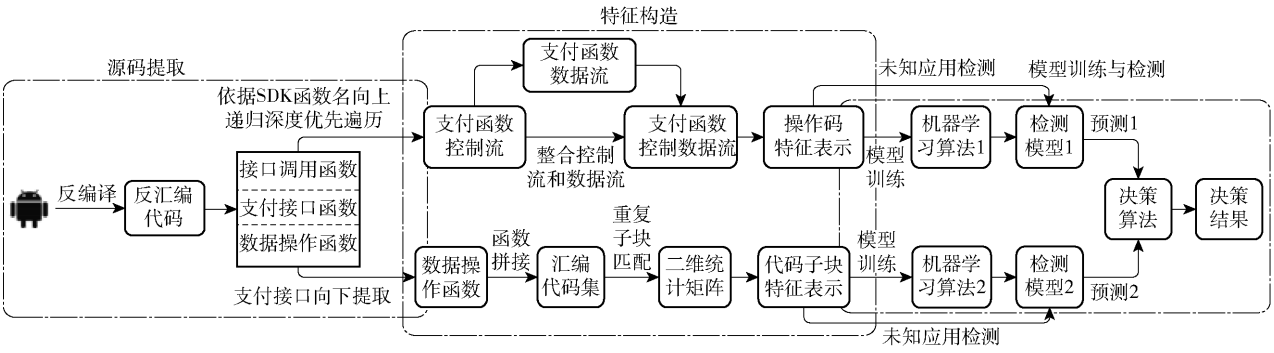


图 4　支付破解应用检测模型

调用控制流(C_CF_G)是一个代码执行列表,每个调用控制流只有一个输入和一个输出,列表中的代码按照执行先后顺序依次排列,不存在执行分支. 以调用流为基础,将控制流嵌入调用流中,生成调用控制流列表 C_CF_G. 在图 CG 中选择一个起点函数 V_1,遍历图 CFG 与之对应的顶点 V_1' 的集合,V_1' 按照执行顺序拼接. 当 V_1' 集合中的语句调用其他函数 V_2 时,将图 CFG 中 V_2' 的集合与调用代码进行拼接,以此形式递归生成调用控制流.

定义 2 重复代码子块

代码块 L 用列表 $L=\{S_1,S_2,\cdots,S_i\}$ 表示,其中 S_i 为一行汇编代码.

重复代码子块用列表 $K=\{s_1,s_2,\cdots,s_j\}$ 表示,其中 $K\subseteq L$,$\text{lenth}(K)<\text{lenth}(L)$ 且 K 在 L 中出现在 2 个以上不同的位置.

定义 3 重复代码子块绝对长度

重复代码子块绝对长度 lenth(K) 为用函数 lenth(K) 计算出重复代码子块 cL 中汇编代码的行数.

定义 4 重复代码子块相对长度

代码块长度 lenth(L) 为用函数 lenth(L) 计算出代码块 L 中汇编代码的行数.

重复代码子块相对长度 rlenth(K) 为用函数 $\text{rlenth}(K)=\text{lenth}(L)-\text{lenth}(K)$ 计算出重复代码子块 K 的相对长度.

2.2 源码提取

源码提取模块首先对应用进行反汇编,提取出 Smali 汇编代码文件集、调用流列表和控制流文件集. 再以第 3 方支付接口为中心,使用在调用流列表中向上向下递归查询调用与被调用函数的方法,定位应用的支付功能和数据库操作功能,提取出所有与支付相关的函数. 调用流列表是一个 $N\times 2$ 的列表,每一行数据表示一组函数调用关系. 以$\langle S_i, D_i\rangle$表示一组调用关系,其中 S_i 为调用函数 ID,D_i 为被调用函数 ID. 图 5 所示为支付代码的提取结果,其中,箭头左侧为所有函数的调用图,箭头右侧为支付函数调用图.

2.3 特征构造与表示

分别从支付函数调用控制流和数据库操作函数汇编代码列表中提取操作码对的概率和重复代码子块频率这 2 组特征.

构造支付函数调用控制流时,首先从支付函数集中提取支付函数调用流;然后提取支付函数调用流中所有函数的函数控制流;最后按照定义 1 中函数调用控制流的构造方式将函数控制流嵌入函数调用流中,生成支付函数调用控制流,如图 4 所示. 假设 $\text{C\_CF\_G}=\{f_1,f_2\cdots,f_i\}$ 为提取到的支付函数调用控制流集合,其中 $f_i=\{c_{i1},c_{i2},\cdots,c_{ij}\}$ 是由 j 条汇编代码构成的汇编码序列. c_{ij} 为一条汇编语句,其结构为$\langle p_{ij},b_{ij}\rangle$. p_{ij} 为语句中的操作码,b_{ij} 为操作码的操作对象. 提取 f_i 中的操作码 p_{ij},构造出 f_i 的操作码序列$(p_{i1},p_{i2},\cdots,p_{ij})$,使用式(1)计算操作码序列 $\text{op\_list}_i=(p_{i1},p_{i2},\cdots,p_{ij})$ 的 n-gram 值.

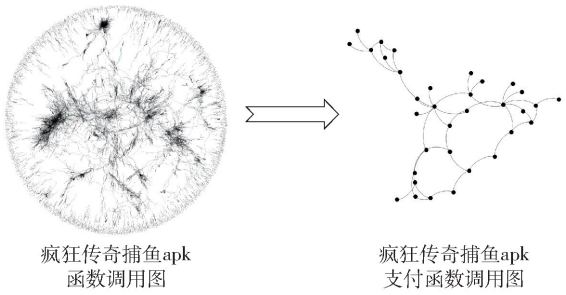


图 5 支付函数提取结果

$$P(p_{i1},p_{i2},\cdots,p_{ij})=P(p_{i1})P(p_{i2}|p_{i1})\cdots P(p_{ij}|p_{ij-1}) \tag{1}$$

$P(p_{ij}|p_{ij-1})$ 为操作码 p_{ij-1} 出现时操作码对 $\{p_{ij-1},p_{ij}\}$ 出现的概率,有

$$P(p_{ij}|p_{ij-1})=\frac{\text{count}(\{p_{ij-1},p_{ij}\})}{\text{count}(p_{ij-1})} \tag{2}$$

通过式(1)、式(2)计算得到一个调用控制流中操作码对的 2-gram 概率列表:$l_i=\{\{p_{i1},p_{i2}\}:g_{i12},\cdots,\{p_{ik},p_{ik+1}\}:g_{ikk+1}\}$. 一个 Android 应用程序包中存在多个操作码对列表:$\text{list}=\{l_1,l_2,\cdots,l_i\}$. 对所有概率列表做平均处理,生成整个支付代码的概率列表 $\text{List}=\{\{p_1,p_2\}:g_{12},\cdots,\{p_k,p_{k+1}\}:g_{kk+1}\}$,作为检测特征.

除了概率特征,还构造了用于检测内容破解的重复代码子块频率特征. 提取频率特征前首先需要构造数据库操作汇编代码集$(C_1,C_2,\cdots,C_k)$. 数据库操作函数集是在第 3 方支付接口函数之后执行的函数集合. 以第 3 方支付接口函数集中的函数名称和函数路径为起点,在函数调用列表中按照广度优先的方式向下遍历 2 层,得到数据库操作函数集. 向下遍历 2 层的原因为数据库操作函数集的破解范围主要集中在数据库操作函数集执行开始的部分,若以深度优先的方式向下遍历出整个数据库操作函数集,函数集中后半部分不在破解范围内的函数会

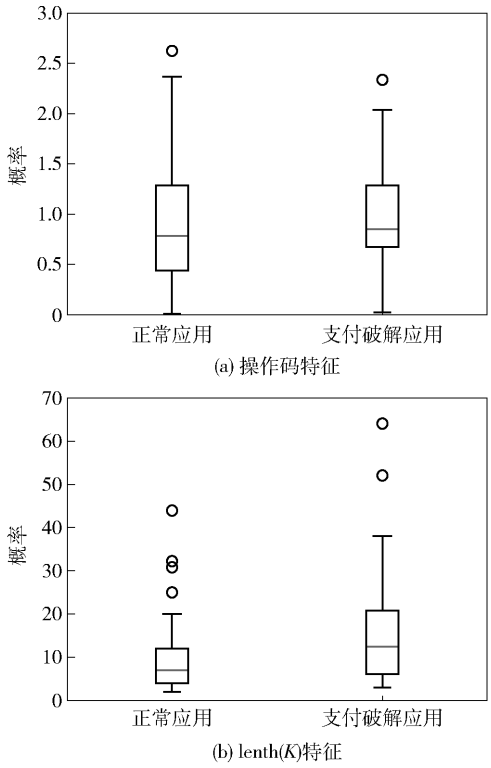
弱化模型的检测效果.

一个数据库操作汇编代码集为(C_1, C_2, …, C_k), C_j 是一条汇编代码,则汇编代码集可映射为一个 $N \times N$ 矩阵. 由于代码块与其自身做匹配,最终生成的 N 维矩阵为对称矩阵. 在统计字符串频率时为避免重复统计,数据统计只选用矩阵对角线的上三角或下三角作为统计对象. 矩阵中数字 1 出现的频率表示重复代码子块的个数. 以每个数字 1 为起点向对角线的斜上方进行查询,查询到的最大非 0 数字表示当前重复代码子块的长度. 矩阵的对角线是整个矩阵中最长的重复代码子块即参与匹配的代码块本身. 在频率统计时,对角线数据不在统计范围之内. 当重复代码子块的绝对长度 lenth(K)为 1 时,说明重复的汇编代码语句只有 1 条. 在 Smali 汇编语言中存在一些可单独使用的汇编指令,此种机制导致矩阵中出现大量 lenth(K)值为 1 的数据. 为提高模型拟合度,在频率统计时 lenth(K)值为 1 的数据不在统计范围内.

矩阵的统计信息包括重复代码子块绝对长度 lenth(K)频率统计信息、重复代码子块相对长度 rlenth(K)频率信息和矩阵整体信息 3 个方面. 其中 lenth(K)频率统计信息为指相同绝对长度的重复代码子块的数量;rlenth(K)频率信息为相对长度相同的重复代码子块的数量;矩阵整体信息是指矩阵的维度,也就是用于比较的代码块长度.

图 6(a)为提取的支付函数调用控制流操作码特征分布线箱图,示出了检测模型中包含的 22 组操作码对出现概率值加和结果的分布情况. 图中上下 2 条线表示数据的覆盖范围,矩形方框表示大部分数据的分布范围,矩形中的实线是全体数据的中位值. 相比于正常应用,支付破解应用统计值的覆盖范围更小,统计值要高于中位值. 出现此种现象的原因在于支付破解应用缺少对支付失败和支付异常接口函数的调用. 缺少的调用方式使整体操作码对的概率值降低,使部分操作码对的概率值升高. 所以在最终的分布中出现了统计值覆盖范围变小,中位数小于大多数统计值的情况.

图 6(b)为最大重复代码子块绝对长度 lenth(K)特征的分布线箱图. 相比正常应用,支付破解应用的最大重复代码子块绝对长度 lenth(K)的覆盖范围更广,超过半数以上的数据高于中位值. 此种数据分布清楚地说明在支付破解应用中存在大量重复代码子块.



(a) 操作码特征

(b) lenth(K)特征

图 6　特征分布线箱

2.4　模型训练与检测

使用 XGBoost(extreme gradient boosting)分类器训练操作码对概率特征,生成检测模型 1,检测结果表示应用中是否存在逻辑破解行为. 使用随机森林分类器训练重复代码子块长度频率特征,生成检测模型 2,检测结果表示应用中是否存在内容破解行为. 决策机制是将检测结果 1 和检测结果 2 进行融合,融合结果表示应用是否存在支付破解行为. 详细算法与实验见 4.3 节.

3　实验结果及分析

3.1　数据集与分类器

当前互联网中没有用于支付破解应用检测工作的公开数据集,所用数据集的样本共有 300 个,其中 168 个为从官方网站和应用市场下载游戏类的正常应用,97 个为从互联网和第 3 方软件市场下载的支付破解应用,35 个为制作样本. 97 个下载的支付破解应用在侵犯正版应用数字版权权益的同时,部分软件存在发送垃圾广告、权限滥用和泄露用户隐私信息等恶意行为.

XGBoost 是一种面向基础的 GradientBoosting 算法的优化版,具有高效的运行效率、灵活性和可移植性. 使用 Sklearn 库训练操作码对概率特征,主要参

数 n_estimators 设定为20,其他默认. 随机森林是将多棵决策树集成的一种算法. 使用 Sklearn 库实现对重复子块长度频率特征的训练检测,主要参数 max_depth 设定为10,n_estimators 设定为20.

3.2 评价指标

用于性能比较的指标包括分类精确率、检测准确率、检测召回率、F1 得分、ROC(receiver operating characteristic)曲线和 ROC 曲线下面积的 AUC(area under curve)值.

ROC 曲线是受试者工作特征曲线/接收器操作特性曲线,是反映敏感性和特异性连续变量的综合指标,是用构图法揭示敏感性和特异性的相互关系. AUC 值是 ROC 曲线与下坐标轴围成的面积. AUC 的取值范围为[0,1]. 当 AUC 取[0,0.5)时,所构造的检测模型无法检测未知样本;当 AUC 取 0.5 时,所构造的检测模型为随机检测模型;当 AUC 取(0.5,1]时,所构造的检测模型可以有效地检测出未知样本,AUC 值越高,模型检测的能力越强.

3.3 实验设计

按照构造的特征提取方法,最终的特征集包括正常应用特征集和支付破解应用特征集. 在模型训练前,需要对特征数据集进行标签标定和数据拆分,将正常应用特征集标定为数值0. 由于破解方式包括2类,使用2种标签对支付破解应用进行标定. 将逻辑破解的样本数据集标定为数值1,内容破解的样本数据集标定为数值2. 最后再结合机器学习分类算法构建检测分类器,用于检测未知的支付破解应用.

在训练检测模型时,使用整体训练集80%的样本特征及其标签作为训练数据. 分别使用样本数据中的概率特征集和频率特征集训练2种不同的机器学习分类算法,生成2种不同的检测模型.

分别在2个模型中对未用于训练的20%的样本特征数据进行检测,再采用决策机制算法融合2种检测结果. “检测模型1检测结果为1”和“检测模型2检测结果为2”有任何一种情况成立时就认定该应用存在支付破解行为,即2种检测结果做“或”运算,一种为真即决策结果为真. 假设一个未知应用的检测结果为(x,y),若 $x==1||y==2$ 则应用为支付破解应用;若 $x!=1\&\&y!=2$ 则为正常应用.

3.4 实验结果与理论分析

表1所示为在100次随机实验后所提特征构造方法与文献[2,8-9]的特征构造方法各项评价指标的平均值. 所提方法对正常应用的平均检测召回率、F1 得分和检测准确率分别为91.80%,85.64%,81.78%,对支付破解应用的平均检测召回率、F1 得分和检测准确率分别为82.37%,84.88%,87.94%. 模型整体平均检测精确率为85.24%. 文献[2,8-9]中各项评价指标的平均值均远低于所提特征构造方法的指标平均值. 同类方法对破解应用的误报率高出所提方法误报率的2倍.

表1 所提方法与相关方法的检测效果对比 %

特征类型		所提方法	文献[2]	文献[8]	文献[9]
召回率	正常应用	91.80	26.92	38.72	22.66
	破解应用	82.37	73.88	64.50	69.54
F1 得分	正常应用	85.64	32.24	40.02	27.37
	破解应用	84.88	65.47	62.13	62.75
准确率	正常应用	81.78	42.84	42.67	33.78
	破解应用	87.94	53.68	57.74	56.46
精确率		85.24	55.57	51.37	50.91

图7所示为所提方法与文献[2,8-9]中的 ROC 曲线图及其对应的 AUC 值. 所提方法和文献[2,8-9]中的 AUC 值分别为0.87,0.55,0.50,0.46. 所提方法的 AUC 值远高于0.5,表明所提方法构造的特征所训练的检测模型为强检测模型,可以有效地检测出未知支付破解应用;文献[2,8-9]中的 AUC 值接近0.5,表明构造的特征所训练的检测模型为随机检测模型,无法有效检测出未知的支付破解应用.

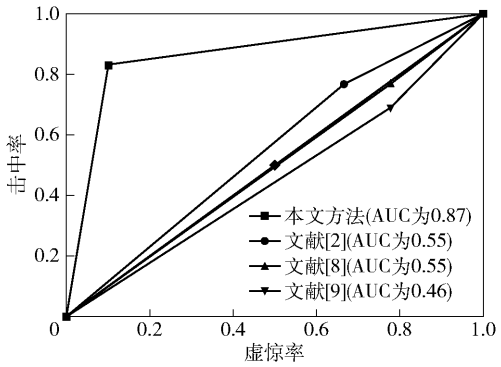


图7 ROC 曲线对比

Peiravian 等[1-9]提出的特征构造方法主要用于检测传统的恶意软件或者对传统的恶意软件家族进行分类. 从评价指标可以看出,用传统方法训练的检测模型为随机检测模型,无法准确检测出未知的支付破解 Android 应用. 破解应用与正常应用之间

的 API 调用情况基本一致，文献[2]中所构造的 API 特征无法表示出应用中存在的支付破解行为. 文献[8-9]中构造的操作码特征无法同时表示出函数间的调用行为、函数内部的执行流程和代码块中出现的大面积代码重复行为.

通过对实验数据和特征进行分析，对召回率贡献度排名前 3 的特征为最大重复代码子块相对长度、代码块长度和重复代码子块相对长度. 3 组特征对存在内容破解行为的应用召回率在 70% 以上. 上述 3 组特征表明，在一个长度较短的代码块中出现了大量的重复代码子块，支付破解应用中存在大面积代码重复的特殊现象.

4　结束语

提出一种基于支付函数调用控制流的操作码概率和重复代码子块频率相结合的特征构造方法，能够准确表示出支付接口函数的调用逻辑和数据库操作函数的函数内容. 分别量化的支付函数调用控制流 *n*-gram 概率特征和数据库操作函数集的重复代码子块长度频率特征准确地表示出 Android 应用中存在的支付破解行为，并基于机器学习实现了带决策机制多分类器的检测模型，有效地识别出支付破解应用. 实验结果表明，检测模型的检测精确率为 85.24%，AUC 值为 0.87，可显著提高对支付破解类应用的检测率，为支付破解应用的检测提供了一种有效的方案.

参考文献：

[1] Peiravian N, Zhu Xingquan. Machine learning for Android malware detection using permission and API calls[C]//2013 IEEE 25th International Conference on Tools with Artificial Intelligence. Herndon: IEEE Press, 2013: 300-305.

[2] Chan P P K, Song Wenkai. Static detection of Android malware by using permissions and API calls[C]//2014 International Conference on Machine Learning and Cybernetics. Lanzhou: IEEE Press, 2014: 82-87.

[3] Aafer Y, Du W, Yin H. DroidAPIMiner: mining API-level features for robust malware detection in Android[C]// International Conference on Security and Privacy in Communication System. Cham: Springer, 2013: 86-103.

[4] 许艳萍，伍淳华，侯美佳，等. 基于改进朴素贝叶斯的 Android 恶意应用检测技术[J]. 北京邮电大学学报，2016，39(2)：43-47.
Xu Yanping, Wu Chunhua, Hou Meijia, et al. Android malware detection technology based on improved naive Bayesian[J]. Journal of Beijing University of Posts and Telecommunications, 2016, 39(2): 43-47.

[5] Fan Ming, Liu Jun, Wang Wei, et al. DAPASA: detecting Android piggybacked apps through sensitive subgraph analysis[J]. IEEE Transactions on Information Forensics and Security, 2017, 12(8): 1772-1785.

[6] Gascon H, Yamaguchi F, Arp D, et al. Structural detection of Android malware using embedded call graphs[C]// Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security. Berlin: ACM, 2013: 45-54.

[7] Zhou Wu, Zhou Yajin, Grace M, et al. Fast, scalable detection of "Piggybacked" mobile applications[C]// Proceedings of the Third ACM Conference on Data and Application Security and Privacy - CODASPY '13. San Antonio: ACM Press, 2013: 185-196.

[8] Zhang Bin, Xiao Wentao, Xiao Xi, et al. Ransomware classification using patch-based CNN and self-attention network on embedded *N*-grams of opcodes[J]. Future Generation Computer Systems, 2020, 110: 708-720.

[9] Li Dan, Zhao Lichao, Cheng Qingfeng, et al. Opcode sequence analysis of Android malware by a convolutional neural network[J]. Concurrency and Computation: Practice and Experience, 2020, 32(18): e5308.