计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2021.20210560
58(11):2456-2474,2021

InterDroid:面向概念漂移的可解释性 Android 恶意软件检测方法

张 炳[1,2] 文 峥[1,2] 魏筱瑜[3] 任家东[1,2]
[1](燕山大学信息科学与工程学院 河北秦皇岛 066004)
[2](河北省软件工程重点实验室(燕山大学) 河北秦皇岛 066004)
[3](中国五洲工程设计研究院 北京 100053)
(jdren@ysu.edu.cn)

InterDroid: An Interpretable Android Malware Detection Method for Conceptual Drift

Zhang Bing[1,2], Wen Zheng[1,2], Wei Xiaoyu[3], and Ren Jiadong[1,2]
[1](*School of Information Science and Engineering*, *Yanshan University*, *Qinhuangdao*, *Hebei* 066004)
[2](*Key Laboratory of Software Engineering of Hebei Province*(*Yanshan University*), *Qinhuangdao*, *Hebei* 066004)
[3](*China Wuzhou Engineering Group*, *Beijing* 100053)

Abstract Aiming at the problems in Android malware detection, which are high subjectivity of feature definition, poor interpretability of feature selection process, and lack of temporal instability of training model detection accuracy, an interpretable Android malware detection method for concept drift called InterDroid is proposed. Firstly, four characteristics of the detection model: permission, API package name, intention and Dalvik bytecode are inferred through the high-quality artificial Android malware analysis report. And InterDroid training and comparison algorithm are obtained through automatic machine learning algorithm TPOT (tree-based tipeline optimization tool), thus abandoning the complicated process of model selection and parameter adjustment in traditional methods. After that, the traditional feature wrapper method is improved by integrating the model interpretation algorithm SHAP (shapley additive explanations), and the feature set with high contribution to the classification results is obtained for detection model training. Finally, the existence of concept drift in Android malware detection is proved by the double tests of MWU(Mann-Whitney U) and machine learning model. Based on the JDA(joint distribution adaptation), the accuracy of the detection model for Android malware in the new era is improved. The experimental results show that the feature screened by InterDroid is stable and interpretable. At the same time, the feature-representation transfer module in InterDroid can improve the detection accuracy of Android malware in 2019 and 2020 by 46% and 44%.

Key words Android malware detection; interpretability; concept drift; feature-representation transfer; automated machine learning

摘 要 针对 Android 恶意软件检测存在特征引入过程主观性高、特征选择过程可解释性差、训练模型检测效果不具备时间稳定性的问题,提出了一种面向概念漂移的可解释性 Android 恶意软件检测方法 InterDroid,该方法首先通过高质量的人工 Android 恶意软件分析报告引入权限、API 包名、意图、

收稿日期:2021-06-07;**修回日期**:2021-08-11
基金项目:国家自然科学基金项目(61802332,61807028,61772449);燕山大学博士基金项目(BL18012)
This work was supported by the National Natural Science Foundation of China (61802332, 61807028, 61772449) and the Doctoral Foundation Program of Yanshan University (BL18012).

Dalvik 字节码 4 种特征.并通过自动化机器学习算法 TPOT(tree-based tipeline optimization tool)获得 InterDroid 训练及对比算法,从而摒弃传统方法中繁复的模型选择与参数调整过程.其后,融入模型解释算法 SHAP(shapley additive explanations)改进传统的特征包装方法,从而获得对分类结果具有高贡献度的特征组合用于检测模型训练.最后,通过曼-惠特尼 U(Mann-Whitney U, MWU)与机器学习模型的双重检验证明概念漂移现象在 Android 恶意软件检测中的存在性.并基于联合分布适配(joint distribution adaptation, JDA)算法提高检测模型对新时期 Android 恶意软件的检测准确率.实验表明:InterDroid 筛选出的特征组合具备稳定性与可解释性.同时,InterDroid 中的特征迁移模块可将自身对 2019 年、2020 年新兴 Android 恶意软件的检测准确率分别提高 46%,44%.

关键词　Android 恶意软件检测;可解释性;概念漂移;特征迁移;自动化机器学习

中图法分类号　TP309

2021 年第 1 季度,360 互联网安全中心截获移动端新增恶意程序样本约 206.5 万个,比 2020 年同期增长 426.5%,造成人均经济损失 14 611 元[1].相较于 iOS 操作系统,Android 操作系统占据中国移动端市场 76.91%[2],且 Android 开放平台的应用软件生态,使其更易受到恶意软件威胁.

现有的 Android 恶意软件检测方法分为:基于特征码的检测方法、基于机器学习的静态检测方法、基于机器学习的应用行为检测方法 3 大类.基于机器学习的静态检测方法因其对未知恶意软件检测准确率高、对设备硬件要求低等优点成为主流的 Android 恶意软件检测方法.

基于机器学习的静态检测方法在特征选取上以权限特征为核心,并选取多种其他特征作为辅助.例如:应用程序编程接口(application programming interface, API)、服务、广播、字符串等.在模型训练算法上,决策树、梯度提升树等分类树算法,频繁项集与关联模式挖掘等推荐算法取得较好表现后[3],卷积神经网络等计算机视觉领域算法及 Word2vec 等自然语言处理领域算法也被引入 Android 恶意软件检测,用于处理多模态的恶意软件特征.

但是,传统的机器学习静态检测方法正面临 3 个主要挑战:

1) 应用市场中请求敏感权限的应用比例正在下降[4],部分恶意应用能在不申请新权限的基础上完成攻击.单一的权限特征,或无逻辑引入的特征组合不足以表征恶意软件.

2) 黑盒的机器学习算法获得越来越高准确率的同时,恶意应用检测对模型的可解释性与透明性要求越来越高[5].Android 恶意软件逆向人员需要模型提供决策依据,以促进人工分析或判断模型决策的合理性.

3) Android 系统版本的高频率更新导致以各版本软件开发工具包(software development kit, SDK)为基础开发的 Android 应用均拥有一定市场占有率[6].而由于概念漂移现象,以大量样本为代价训练得到的机器学习模型在对不同时期 Android 恶意软件的检测上表现较差.

为此,本文提出一种面向概念漂移的可解释性 Android 恶意软件检测方法.在特征引入阶段,该方法首先基于 Kharon 数据集[7]以恶意软件人工分析报告中的高频词作为依据,在源代码层次选取权限名、API 包名、意图 3 类特征,并在汇编指令层次选取 Dalvik 字节码特征作为补充.从而多层次地扩充单一的权限特征,并提升特征引入过程逻辑性及特征对恶意软件的表征能力.在特征包装与解释阶段,首先对每种特征分别以 TPOT(tree-based tipeline optimization tool)算法[8-10]筛选机器学习模型进行预训练,其后基于 SHAP(shapley additive explanations)算法[11-13]对每个预训练模型建立解释器,筛选得到对分类结果具有高贡献度的可解释性特征组合.并以此提高模型的透明度,为人工分析与验证提供决策依据.在模型迁移阶段,首先判断已有模型是否需要迁移.一方面利用预训练得到的模型预测不同时期恶意样本,另一方面基于特征包装与解释阶段输出的特征组合采样不同时期 Android 恶意软件,并进行曼-惠特尼 U(Mann-Whitney U, MWU)检验[14].在满足特征迁移条件后,基于联合分布适配(joint distribution adaptation, JDA)算法[15-16]改进已有训练模型.最终以无标注小样本的不同时期软件为代价完成已有训练模型迁移,提升已有检测模型对不同时期恶意软件检测效果.最终,再次使用 TPOT 算法筛选对比模型进行验证.实验结果表明,InterDroid 对同时期 Android 恶意软件检测准确率达 95%,对

不同时期恶意软件能将已有模型检测准确率提升1倍以上.能有效检测同时期恶意软件,提高模型决策依据透明度,并在较低迭代次数下完成模型迁移,从而缓解概念漂移导致的 Android 恶意软件检测准确率下降问题.

综上,本文提出的方法主要有3方面贡献:

1) 基于 Android 恶意软件分析报告提取攻击流程中的高频词,据此引入权限名、API 包名、意图这3类源代码层次特征,以及汇编指令层次的 Dalvik 字节码特征.使初始特征组合在保证低存储开销与较高分析速度的同时能够更好地表征恶意软件,提高引入特征组合的逻辑性与合理性.

2) 改进传统的特征包装方法,使用自动化机器学习 TPOT 算法筛选最佳分类模型集合,将集合中模型与4种特征两两组合训练,并为训练得到的模型建立解释器.InterDroid 筛选出的特征对多数训练样本的分类结果具有高贡献度,并为模型的验证与恶意软件的人工分析提供依据.

3) 将领域自适应方法引入 Android 恶意软件检测.在已有数据和模型的基础上,以少量无标注新时期 Android 软件为代价,即可有效提升现有模型对新恶意样本的检测准确率,进而缓解概念漂移问题.

1 相关工作

Android 系统的沙盒机制,使非定制系统中的应用动态行为监控较为困难[17].目前 Android 恶意应用自动化检测在特征提取阶段仍以静态特征为主,并以权限特征为主线展开.Felt 等人[18]最早建立了 API 到权限的映射,并开发了检测 Android 应用程序恶意代码的 Stowaway;但忽略了用户自定义权限,并且无法建立复杂的 Java 反射调用到权限的映射关系.Bartel 等人[19]将 API 链接到服务的绑定机制,找到应用声明权限与其实际使用权限的不同,填补了 Stowaway 在 Java 反射调用到权限映射上的空白,但其未能研究 Dalvik 字节码层面的特征.Karim 等人[20]将 API 到权限的单向映射完善为可相互追溯的双射,通过关联模式挖掘,在软件开发阶段提醒开发者使用的 API 应申请的最小权限;但其研究局限于开源软件,而实际的 Android 恶意软件检测目标多为编译后的应用程序安装包.Olukoya 等人[21]引入自然语言处理技术,将应用市场中的程序描述作为权限的补充信息.Wang 等人[22]以用户评论替代应用描述进行权限推断,并认为在反映权限的真实使用状况上众包用户评论较之应用描述更具有效性.但以上2种方法均依赖于应用描述与用户评论的真实性.

相较于上述方法,本文基于人工 Android 恶意软件分析报告,从权限特征出发,增加 API 包名、意图等 Android 逆向分析流程中的突破点作为特征,并加入汇编指令层的 Dalvik 字节码作为补充,提高了特征的全面性与客观性.

针对 Android 恶意应用自动化检测中的维数灾难问题,现有研究一方面采用过滤、包装、嵌入[23]等方法降低特征维度,另一方面将高维特征转化为图像,结合计算机视觉改进检测模型.在特征降维方面:Alecakir 等人[24]基于注意力机制与递归神经网络等机器学习模型包装特征;Li 等人[25]采用嵌入方法,通过排序得到恶意与良性样本的差异性特征,并融合决策树剪枝与频繁项集挖掘算法进行特征筛选;Yuan 等人[26]提出权限风险值的思想,通过 Markov 链过滤特征.在高维特征表征为图像方面:Chen 等人[27]将源代码基本块转换成多通道图像并以此训练卷积神经网络模型;Ünver 等人[28]则首先将源代码转换为灰度图像,再选取图像的局部与全局特征.

上述研究虽然有效地解决了高维特征的表征问题,但其特征引入过程具有主观性.例如,同样是重复出现的权限集合,Wang 等人[29]认为应该保留,而文献[25]主张删除,并且特征选择与模型训练过程缺乏可解释性.文献[22]虽然生成了可解释的判别规则,但模型依赖决策树,不能扩展为其他模型.本文融合 SHAP 算法,通过解释特征包装模型输出完成特征选择.在降低特征维度的同时,保证了训练模型的可扩展性与选择特征的可解释性,为人工逆向分析提供指导.

由于 Android 版本与开发库的频繁更新,Android 恶意软件检测存在概念漂移问题[30].Mariconti 等人[31]通过构建 Markov 行为模型链,有效地提升了2年内恶意软件的检测效率.本文最终的特征迁移可与文献[31]相互补充.

2 方法设计

首先,本节介绍 InterDroid 整体研究框架,建立 InterDroid 数学模型.其次,分节介绍研究框架中不同模块及其涉及的方法.由于模型训练模块与实验

数据联系紧密,因此其中包含的数据处理与训练结果将在第 3 节实验评估中详细展开.

2.1　研究框架

本节分为框架总述与框架建模 2 部分介绍 InterDroid 研究框架.其中,框架总述为本文方法各阶段研究目标与研究步骤的简要概括;框架建模为本文方法建立数学模型,细化总体框架.

1) 框架总述

本文研究框架总体上分为特征包装与解释、模型训练、迁移判定、特征迁移 4 个部分.如图 1 所示:

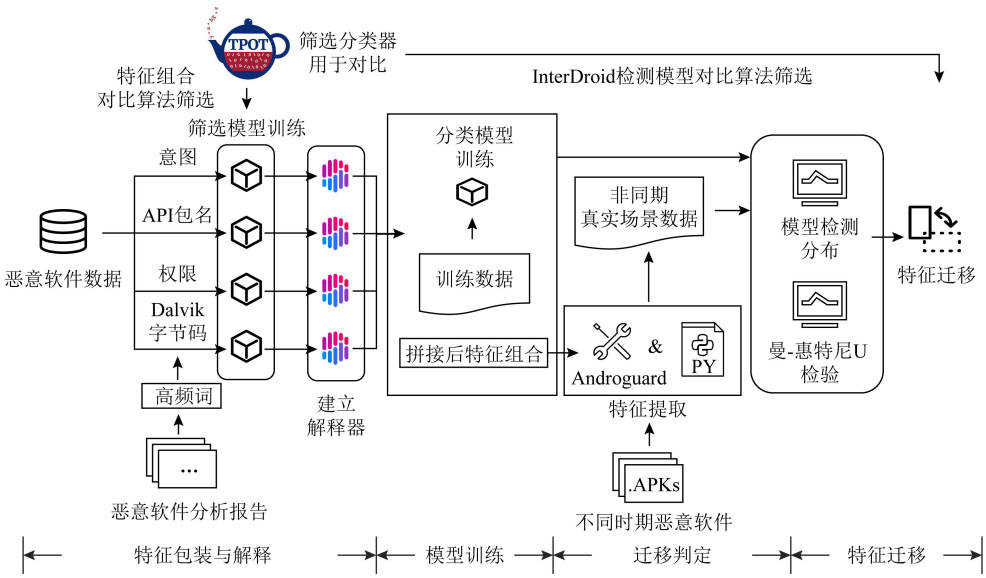

Fig. 1　Illustration of the research framework of InterDroid

图 1　InterDroid 研究框架示意图

① 特征包装与解释.以筛选出可解释的特征组合为目标.首先提取 Android 恶意样本分析报告关键词,获得权限、API 包名、意图、Dalvik 字节码 4 种特征类型.其次,基于 TPOT 算法得到筛选模型集合并与 4 种特征数据两两组合进行训练.基于 SHAP 算法对训练后的筛选模型建立解释器,从而类内排序每类特征包含特征分量的 SHAP 值.筛选出 SHAP 值高,即对分类结果贡献度高且对大多数样本具有显著影响的特征分量,并采用水平拼接方式融合上述特征分量,融合结果即为 InterDroid 筛选出的特征组合.

② 模型训练.以获得 InterDroid 检测模型及不同时期 Android 恶意软件数据为目标.基于筛选出的特征组合,一方面抽取已有 Android 恶意软件数据集中对应数据,训练检测模型;另一方面,通过 Python 整合 Androguard[32],从而提取不同时期恶意软件安装包中对应数据,供迁移判定模块使用.

③ 迁移判定.以判定当前模型是否需要进行特征迁移为目标.一方面,使用不同时期 Android 恶意软件数据测试原有分类检测模型;另一方面对不同时期 Android 恶意软件数据进行同分布检验.若当前模型准确率低于工业界标准且不同时期 Android 恶意软件特征存在概念漂移现象,则对当前分类检测模型进行特征迁移.

④ 特征迁移.以提升现有模型对新兴 Android 恶意软件检测准确率为目标.基于 JDA 算法将当前模型训练数据领域知识迁移到不同时期 Android 恶意软件领域,进而提升 InterDroid 检测模型对新兴 Android 恶意软件的检测准确率.

2) 框架建模

InterDroid 检测模型依据的特征组合 finalfeature 经由特征包装与解释模块中高频词提取、基于 TPOT 的对比算法筛选、基于 SHAP 的特征可解释性包装 3 步获得.

① 高频词提取.基于 Kharon 数据集提取 Android 恶意软件分析报告高频词汇,并将排名靠前的高频词语对应到 Androguard 可提取的静态特征种类.

$$r(R)\rightarrow featureset=\{fn_1,fn_2,\cdots,fn_s\},\quad(1)$$

$$R=\{report_1,report_2,\cdots,report_n\},\quad(2)$$

其中,r 为高频词提取映射,R 为 Android 恶意软件报告集合,$featureset$ 为选取的特征种类集合,fn_i $(1\leqslant i\leqslant s)$ 为特征名,$report_j$ $(1\leqslant j\leqslant n)$ 为第 j 篇 Android 恶意软件分析报告,$report_j$ 中截取片段示例如图 2 所示:

*A task executor MainService$3, that is launched every 180 seconds, sends an **intent** TOR_SERVICE to start the TorService **class**. If Tor is already up, the TorSender **class** is called to send the IMEI of the phone using the service.*

Fig. 2 Illustration of malware analysis report fragment

图 2 恶意软件分析报告片段示意图

② 基于 TPOT 的检测算法筛选.基于 OmniDroid[33] 数据集,投影得到 4 种特征对应的 Android 恶意软件子数据集合.

$$pd_i=\prod_{fn_i}(maldata), \tag{3}$$

其中,$maldata$ 为 OmniDroid 包含的所有 Android 恶意软件数据,pd_i 为 $maldata$ 取 fn_i 特征投影后的子数据集合.

为每种特征数据分别建立 TPOT 算法筛选器,得到 4 种划分数据集上检测表现最佳的分类器集合.

$$\bigcup_{h=1}^{s} tp(pd_h)=\{m_1,m_2,\cdots,m_k,\cdots m_p\}, \tag{4}$$

其中,tp 为 TPOT 算法筛选映射,p 为 tp 筛选得到的分类模型种类总数,m_k 为 tp 筛选得到的分类模型,$1\leqslant k\leqslant p$.

③ 基于 SHAP 的特征可解释性包装.将分类器与子数据集合两两组合进行训练,并为训练得到的模型建立 SHAP 解释器,从而获得对模型分类结果影响最大的 t 个特征.

$$calculate(m_k(pd_h))=shap_h, \tag{5}$$

$$shap_h=\{v_{h1},v_{h2},\cdots,v_{ht}\}, \tag{6}$$

其中,$calculate$ 为检测模型 m_k 的 SHAP 值计算与排序映射,$shap_h$ 为第 h 个子数据集合 pd_h 得到的可解释特征组合,v_{hq} 为该特征组合中第 q 个特征名称,$1\leqslant q\leqslant t$.

融合 4 个特征组合作为 InterDroid 的分类特征,并由此得到 InterDroid 最初分类模型训练数据.

$$finalfeature=\bigcup_{h=1}^{s} shap_h, \tag{7}$$

$$\prod_{finalfeature}(maldata)=\boldsymbol{tdata}, \tag{8}$$

其中,$finalfeature$ 为特征包装与解释模块输出的特征组合,$\boldsymbol{tdata}$ 为模型训练模块的训练数据.

最终,InterDroid 统一建模为

$$InterDrover=\{e,t,j,m\}, \tag{9}$$

其中,e 为基于 Androguard 的 APK 特征提取映射,t 为基于双重分布检验的迁移判定映射,j 为基于 JDA 的特征迁移映射,m 为恶意软件检测分类映射.

对于新兴未知 Android 软件小样本,e 以.APK 格式的安装包作为输入,输出该新时期 Android 软件 $finalfeature$ 对应数据.结合原有数据库采样数据作为 t 的输入,从而判断 2 组数据之间是否需要特征迁移.t 的判断结果与 e 提取到的原始数据共同作为 j 的输入,以获得模型分类数据并输入到 m,从而输出分类结果.

$$e(APK)\rightarrow fdata, \tag{10}$$

$$t(fdata,sam)\rightarrow judge=\{\mathrm{T},\mathrm{F}\}, \tag{11}$$

$$j(fdata,judge)\rightarrow finaldata, \tag{12}$$

$$finaldata=\begin{cases}cdata, & judge=\mathrm{T},\\ fdata, & judge=\mathrm{F},\end{cases} \tag{13}$$

$$m(finaldata)=\{0,1\}, \tag{14}$$

其中,$fdata$ 为基于新兴 Android 软件提取到的 $finalfeature$ 特征数据,sam 为模型训练模块中训练数据采样后的子集,$judge$ 为迁移判定映射的判断结果,$finaldata$ 为本文检测模型输入数据,$cdata$ 为 $fdata$ 特征迁移后的结果.T 表示 $fdata$ 需要特征迁移,F 反之.1 表示未知 Android 软件为恶意软件,0 反之.

2.2 特征包装与解释

本节介绍 InterDroid 检测模型依据的可解释性特征组合筛选过程.依据 2.1 节中建立的数学模型,特征包装与解释模块分为高频词提取、基于 TPOT 的对比算法筛选、基于 SHAP 的特征可解释性包装 3 个部分.但是,由于高频词提取部分具有原理简单、与 Android 恶意软件分析报告语料数据结合紧密的特点,该部分归并到第 3 节实验评估中详细展开.因此,本节包含基于 TPOT 的检测算法筛选、基于 SHAP 的特征可解释性包装 2 部分内容.

1) 基于 TPOT 的检测算法筛选

一方面,针对特征包装与解释模块 4 种不同的特征数据,基于 TPOT 算法筛选适用于不同种类特征的最佳分类器集合,进而得到多个筛选模型对不同种类特征的包装与解释,以便验证 InterDroid 所用特征组合的稳定性.另一方面,针对模型训练模块的输入数据,基于 TPOT 算法筛选适用于 InterDroid 检测模型训练数据的最佳分类器,以对比检测模型的效率与特征迁移效果.

① 自动化机器学习 TPOT 算法

TPOT 以最大化分类精度为目标基于遗传算法自动构建一系列数据转换和机器学习模型,从而自动化机器学习中的模型选择及调参.TPOT 目前支持的分类器主要有贝叶斯、决策树、集成树、SVM、KNN、线性模型、xgboost 等.其整体流程如图 3 所示:

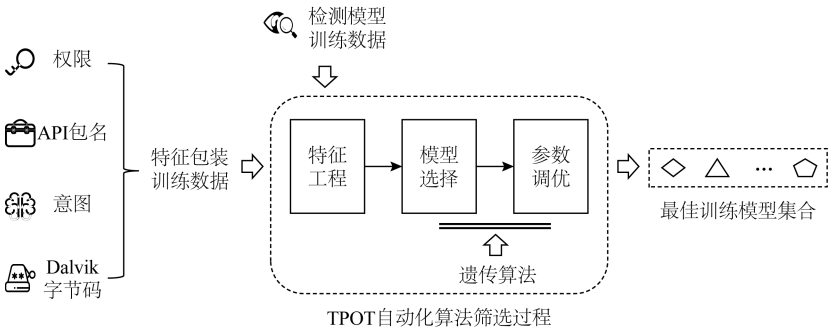

Fig. 3　Illustration of TPOT

图 3　TPOT 示意图

TPOT 对输入的不同种类特征数据或 InterDroid 检测模型训练数据首先尝试进行二值化、正则化等操作,并尝试对输入特征做基于方差或基于 F-值的特征选择操作.之后,面向预处理完成的输入数据随机生成固定数量的管道,根据分类精度对这些管道进行评估.并从现有管道群体中随机选择 3 个管道,移除最低适应度的流水线复制到新的群体中.创建新的群体后,将单点交叉应用于固定百分比的复制管道,其中随机选择 2 条管道相互交换内容.随后,对剩余未受影响的管道进行均匀突变、插入突变与收缩突变.

TPOT 重复迭代以上评估、选择、交叉、变异 4 步,从而分别筛选出用于特征组合对比与 InterDroid 检测模型对比的算法集合.

② 特征组合对比算法筛选

特征组合对比算法筛选步骤整体流程如图 4 所示.基于 OmniDroid 数据集,分别投影权限、API 包名、意图、Dalvik 字节码 4 种特征,得到特征包装训练数据 pd.以 Dalvik 字节码为例,其特征名向量 ***pdname*** 为

("shl-int", "long-to-int", …, "if-gt").

特征包装训练数据中代表 1 个 Android 良性或恶意软件的行切片向量 ***perpd*** 为

(5,3, …,21),

其中,行切片向量中每个分量代表特征名向量中对应分量的出现次数.

分别将权限、API 包名、意图、Dalvik 字节码对应的 4 种特征包装训练数据输入 TPOT 算法,筛选得到针对不同训练数据的最佳管道,并以最佳管道使用的算法作为最佳训练模型 m_k.例如,若 Dalvik 字节码最佳管道为

$\{GradientBoostingClassifier(n\_estimators=50, learning\_rate=0.1), SVC()\}$,

则其最佳训练模型为

$\{GradientBoostingClassifier, SVC\}$.

将所有最佳训练模型添加到最佳训练模型集合,集合内每种训练模型均与 4 种特征包装训练数据两两组合,得到多个特征组合对比模型,且所有对比模型均会作为 SHAP 解释器的输入,用以对比特征包装与解释模块筛选得到的特征组合稳定性.

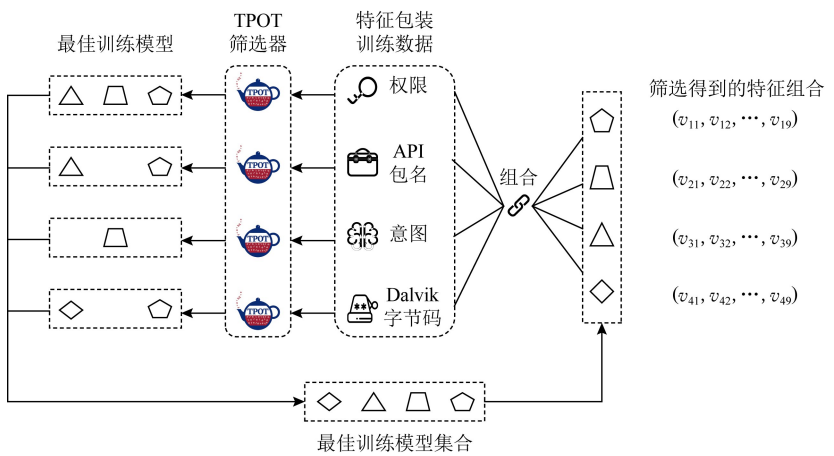

Fig. 4　Illustration of algorithm filtering for comparing the feature set

图 4　特征组合对比算法筛选示意图

特征组合对比算法筛选步骤以 TPOT 算法筛选得到的对比模型替代人工选择的常用对比模型，自动化地完成了不同种类特征数据的训练模型选择与参数调优过程，使选择得到的最佳训练模型具有较高数据相关性，避免了模型训练中参数调整的不确定性.

③ 检测模型对比算法筛选

检测模型对比算法筛选步骤整体流程如图 5 所示：

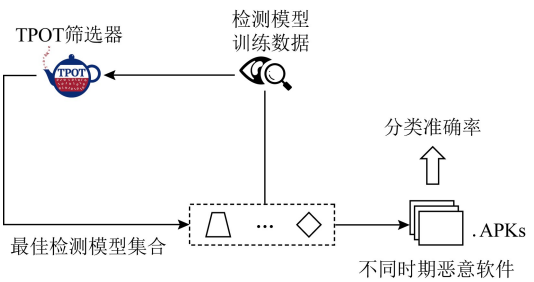

Fig. 5 Illustration of algorithm filtering for detection model

图 5 检测模型对比算法筛选示意图

基于筛选出的特征组合取 OmniDroid 数据集投影，得到 InterDroid 最初检测模型训练数据 ***tdata***.筛选出的特征组合向量为

(“div-float”, …, “android.view.inputmethod”, …, “RECEIVE_BOOT_COMPLETED”, …, “java.util.concurrent.locks”).

最初检测模型训练数据 ***tdata*** 中代表 1 个 Android 恶意软件的行切片向量为

$$(0,\cdots,1,\cdots,21,\cdots,4),$$

其中，行切片向量中每个分量代表特征组合向量中对应分量出现次数.

以 ***tdata*** 作为 TPOT 算法输入，得到面向当前时期 Android 恶意软件特征数据的最佳检测模型集合.测试集合中每个模型对当前时期及不同时期 Android 恶意软件的检测准确率，用以对比 InterDroid 检测模型准确率.

TPOT 算法筛选得到的对比算法均在 InterDroid 检测模型训练数据上具有较高的准确率.较之以往研究中普遍选择的朴素贝叶斯、K 均值等算法，更能突出 InterDroid 在缓解不同时期 Android 恶意软件特征数据的概念漂移问题上具有较佳鲁棒性.

2) 基于 SHAP 的特征可解释性包装

针对黑盒 Android 恶意软件检测模型存在的准确率高但可解释性不足问题，基于 SHAP 算法面向特征包装与解释模块中复杂的特征筛选模型建立解释器，通过 SHAP 事后归因的模型解释方法筛选出对 Android 恶意软件检测结果具有高贡献度的特征.

LIME，DeepLIFT 等可解释方法实质上均可归结为 SHAP 算法.由 SHAP 算法计算得到的 SHAP 值是权限、API 包名、意图、Dalvik 字节码某类特征中某个特征分量在该类特征序列中的平均边际贡献，即 SHAP 算法考虑了同种特征组中不同特征分量之间的协同效应.例如：权限特征组中“RECEIVE_SMS”与“SEND_SMS”分量之间存在相互影响.因此，选择 SHAP 算法为特征包装与解释模块中的解释器建立算法.

特征包装与解释模块流程如图 6 所示：

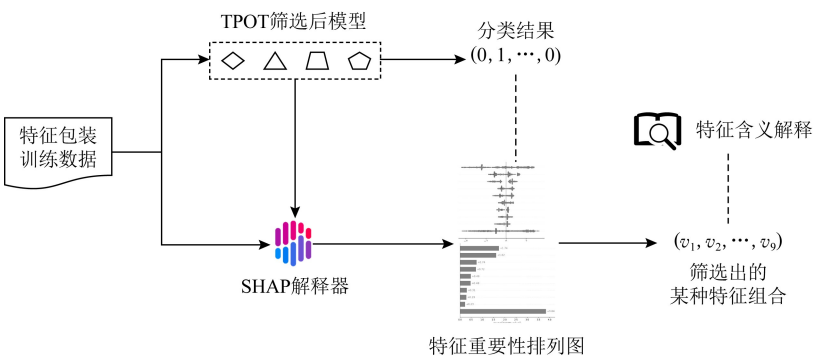

Fig. 6 Illustration of feature wrapper and interpretation module

图 6 特征包装与解释模块示意图

InterDroid 基于特征组合对比算法筛选步骤输出的最佳训练模型集合与特征包装训练数据，通过 SHAP 算法简化每个最佳训练模型 m_k 为事后解释模型 h_k.针对特征包装训练数据 pd 中任意一个代表 Android 良性或恶意软件的行切片向量 ***perpd***，事后解释模型 h_k 满足：

$$h_k(\boldsymbol{perpd})=avg_0+\sum_{q=1}^{Q}shap_q perpd_q=m_k, \quad (15)$$

其中,avg_0 为所有特征包装训练数据 pd 标签值的均值,$shap_q$ 为特征名向量 ***pdname*** 第 q 个分量的 SHAP 值,$perpd_q$ 为行切片向量 ***perpd*** 的第 q 个分量值.

通过式(15),SHAP 解释器计算得到权限、API 包名、意图、Dalvik 字节码中某类特征序列各个特征分量的 SHAP 值,从而绘制特征密度散点图与特征重要性 SHAP 值图.

特征密度散点图、特征重要性 SHAP 值图均为特征重要性的排列图.其中,特征重要性 SHAP 值图为每个特征 SHAP 值排序;特征密度散点图将所有 Android 良性或恶意软件样本点呈现在图中,一方面可以直观得到哪些特征分量对大部分样本的分类结果有影响,另一方面可以观察到某个特征分量中不同 Android 良性或恶意软件样本数据的数值大小对分类结果的影响.

2.3　迁移判定

不同时期 Android 恶意软件特征数据的概念漂移可能使 InterDroid 对新兴 Android 恶意软件的检测准确率下降.迁移判定模块以模型训练模块中分类模型训练数据 sam、新时期 Android 恶意软件数据 $fdata$ 作为数据支撑,基于机器学习模型检测与 MWU 检验双重同分布检测方法,判定提取到的新兴 Android 恶意软件特征数据是否需要进行特征迁移.

因为 sam,$fdata$ 两类数据具有 3 个特点:1)数据分布状况未知;2)数据来自不同的独立样本;3)数据量在 100 以上.而 KS 检验、t 检验、MWU 检验、威尔科克森符号秩(Wilcoxon signed rank, WSR)检验等常用数据同分布检测方法中:1)当检验的数据符合特定的分布时,KS 检验灵敏度较低.2)t 检验样本需满足正态分布,且其最佳样本容量小于 30.3)WSR 检验与 MWU 相似,均不要求样本服从正态分布且样本容量大于 20 时即可获得较好的检验效果,但 WSR 检验应用于 2 个相关样本.因此,迁移判定模块选择 MWU 检验.同时,使用 InterDroid 所使用的机器学习模型进行二次检验,从而确定概念漂移现象已致使本文模型检测准确率降低.

首先,以 $fdata$ 作为测试数据得到 InterDroid 当前分类检测模型的准确率.若当前准确率低于工业界允许的最低准确率,即 90%,则认为当前检测模型已不能满足新时期 Android 恶意软件检测标准.采用 MWU 检验判定其检测准确率下降的原因是否为新时期 Android 恶意软件特征的概念漂移.

将分类模型训练数据 sam、新时期 Android 恶意软件数据 $fdata$ 混合并编排等级,继而分别求出两样本的秩和.得到 sam,$fdata$ 两样本 MWU 检验统计量为

$$U_1=R_1-\frac{\|sam\|(\|sam\|+1)}{2}, \quad (16)$$

$$U_2=R_2-\frac{\|fdata\|(\|fdata\|+1)}{2}, \quad (17)$$

其中,U_1 为分类模型训练数据样本的 MWU 统计量,$\|sam\|$ 为分类模型训练数据样本的数量,R_1 为分类模型训练数据样本的秩和;U_2 为新兴 Android 恶意软件样本的 MWU 统计量,$\|fdata\|$ 为新兴 Android 恶意软件样本的数量,R_2 为新兴 Android 恶意软件样本的秩和.

若 U_1,U_2 最小值小于显著检验 $U_{0.05}$ 时,则提取到的新兴 Android 恶意软件特征数据需要在检测前进行特征迁移.

具体算法如算法 1 所示.

算法 1. 双重分布检验算法.

输入:小样本新时期 Android 恶意软件数据 $fdata$、与小样本同规模的原有模型训练数据采样 sam;

输出:决定是否需要进行特征迁移的判定结果 $judge$.

① $res=m(fdata)$;
　　/* 使用原有模型检测新时期恶意软件 */
② if $acc(res)>90\%$
　　/* 如果现有模型准确率仍能符合要求 */
③　　return false;
④ end if
⑤ $D=\varnothing$; /* 需删除的特征集合 */
⑥ for $(f_i,s_i)\leftarrow(fdata,sam)$
　　/* 每种特征取 2 类数据 */
⑦　　if $MWU(f_i,s_i)>5\%$ /* 如果该特征在 2 个样本上具有显著性差异 */
⑧　　$D=D\cup f$;
⑨　　end if
⑩ end for
⑪ $temp=m(fdata-D)$;
　　/* 去除有差异的特征训练临时模型 */
⑫ if $|acc(temp)-acc(res)|\leqslant 10\%$
　　/* 两模型准确率不具有明显差距 */

⑬ return true;
⑭ else
⑮ return false;
⑯ end if

2.4 特征迁移

针对不同时期 Android 恶意软件特征概念漂移导致的 Android 恶意软件检测模型对新时期恶意软件检测准确率降低的问题,以有标注的 InterDroid 检测模型训练数据为源域,基于 JDA 算法完成对目标域中无标注新时期 Android 软件良性或恶意的检测判定,从而达到领域自适应的效果,提高 InterDroid 对新兴 Android 恶意软件的检测准确率.

新兴 Android 恶意软件存在 2 个问题:1)权限、意图等特征数据较之早期数据整体不相似或部分不相似.2)短期内难以收集大规模、同分布、带标注的数据集用以提升检测模型准确率.而 JDA 算法具有 2 个优点:1)能够同时处理 InterDroid 原有训练数据中知识需迁移到边缘分布不同的新数据(即整体不相似新数据)、条件分布不同新数据(即类内不相似新数据)这 2 种情况.2)以小样本、无标注新时期 Android 软件即可完成特征迁移.因此,特征迁移模块选择 JDA 算法以提升 InterDroid 检测模型的准确率.

特征迁移模块流程如图 7 所示:

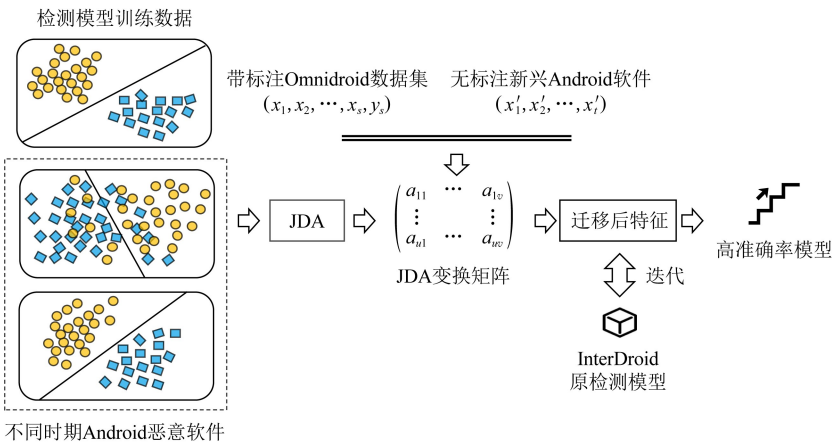

Fig. 7 Illustration of feature-representation transfer

图 7 特征迁移模块示意图

基于不同时期 Android 恶意软件样本集,提取特征包装与解释模块输出的特征组合对应数据.将提取得到的新兴 Android 恶意软件数据与 InterDroid 检测模型训练数据同时作为特征迁移模块的输入.输入数据在特征迁移模块中经过边缘分布适配、条件分布适配 2 次适配后输出变换矩阵.

边缘分布适配应满足:

$$D_1(fdata,sam)=\mathrm{tr}(\boldsymbol{J}_1^{\mathrm{T}}\boldsymbol{C}\boldsymbol{M}_0\boldsymbol{C}^{\mathrm{T}}\boldsymbol{J}_1), \quad (18)$$

其中,D_1 代表边缘分布适配中新兴 Android 恶意软件数据 $fdata$ 与 InterDroid 检测模型训练数据 sam 之间的距离,$\boldsymbol{J}_1$ 为边缘分布适配变换矩阵,$\boldsymbol{C}$ 为 $fdata$ 与 sam 合并后所得矩阵,$\boldsymbol{M}_0$ 为边缘分布适配最大-最小距离(maximum-minimum distance, MMD)矩阵.条件分布适配应满足:

$$D_2(fdata,sam)=\sum_{e=0}^{1}\mathrm{tr}(\boldsymbol{J}_2^{\mathrm{T}}\boldsymbol{C}\boldsymbol{M}_e\boldsymbol{C}^{\mathrm{T}}\boldsymbol{J}_2), \quad (19)$$

其中,D_2 代表条件分布适配中 $fdata$,sam 两类数据之间的距离,$\boldsymbol{J}_2$ 为条件分布适配变换矩阵,$\boldsymbol{M}_e$ 为条件分布适配 MMD 矩阵.继而得到统一的变换矩阵求解方程:

$$\left(X\sum_{e=0}^{1}\boldsymbol{M}_e\boldsymbol{C}^{\mathrm{T}}+\alpha I\right)\boldsymbol{J}=\boldsymbol{C}\boldsymbol{H}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{J}\boldsymbol{\Phi}, \quad (20)$$

其中,$\boldsymbol{\Phi}$ 是拉格朗日乘子,$\boldsymbol{J}$ 为变换矩阵,$\boldsymbol{H}$ 为中心矩阵.经由变换矩阵变换后,新兴 Android 恶意软件数据 new_{fdata},InterDroid 检测模型训练数据 new_{sam} 分别为

$$new_{fdata}=\boldsymbol{J}^{\mathrm{T}}fdata, \quad (21)$$

$$new_{sam}=\boldsymbol{J}^{\mathrm{T}}sam. \quad (22)$$

以变换后 new_{sam} 作为训练集修正 InterDroid 原检测模型,预测变换后 new_{fdata} 数据,即可得到新时期未知 Android 软件良性或恶意的检测结果.

3 实验评估

本节从 4 个方面验证 InterDroid 的创新性与有效性:

1) 验证 InterDroid 选取的特征及机器学习模型的合理性.

2) 验证特征包装与解释模块筛选得到的特征组合的稳定性与可解释性.

3) 验证概念漂移现象在不同时期 Android 恶意软件特征中的存在性.

4) 验证特征迁移能够有效缓解概念漂移问题,提升 InterDroid 对新时期 Android 恶意软件的检测准确率.

本文实验的实验环境配置如下:CPU 为 Intel 10210U,内存为 16 GB 的便携计算机,操作系统版本为 Windows 10 20H2.

3.1　数据集与评估指标

本文数据基础为 Kharon 数据集、OmniDroid 数据集与 2019 年、2020 年流行的 Android 恶意软件源数据.

针对 Android 恶意软件检测模型分类特征引入的随意性问题,基于 Kharon 数据集中不同 Android 恶意家族代表性恶意软件人工分析报告提取高频词,作为 InterDroid 检测模型的特征引入依据.针对 Android 恶意软件检测模型训练数据普遍不平衡的问题,基于 OmniDroid 数据集,预处理后得到良性软件与恶意软件同比例的训练数据.针对不同时期 Android 恶意软件数据概念漂移导致的检测模型准确率大幅降低问题,收集 2019 年、2020 年流行的 Android 恶意软件安装包,供 InterDroid 迁移判定与特征迁移模块使用.

1) Kharon

Kharon 数据集由一系列被完全解析的 Android 恶意软件分析文档组成.Kharon 包含攻击性广告软件(agressive adware, AA)、勒索软件(ransomware, R)、远程控制软件(remote administration tool, RAT)、监控软件(spyware, S)等不同 Android 恶意家族代表性软件样本,其部分样本描述如表 1 所示:

Table 1　Partial Sample Description of Kharon Dataset

表 1　Kharon 数据集部分样本描述

样本名称	家族	描述
MobiDash	AA	Aggressive adware which can wait several weeks before triggering
SimpLocker	R	Ransom, data encryption and phone locking
DroidKungFu1	RAT	Undesired applications installation
SaveMe	S	Remote controlled spyware which can make phone calls and send SMS

本文选取 Kharon 每条记录中的“Details”部分,即人工触发并记录的 Android 恶意软件攻击行为.基于记录集合中提取到的高频词语推定 Android 恶意软件检测模型所使用的特征种类.记录片段示例如 2.1 节中图 2 所示.

2) OmniDroid

OmniDroid 数据集发布于 2018 年,包含 21992 个真实恶意或良性软件样本的 28 种静态、动态与预处理特征数据.OmniDroid 仅给出不同反病毒引擎对该数据集中每条记录对应 Android 软件的检测结果.因此在使用该数据集时,需根据多种反病毒引擎检测结果自行确定良性或恶意软件标准.

基于 OmniDroid 数据集中包含的反病毒引擎检测结果,本文判定 1 条记录来源于良性或恶意软件的标准为:①若记录“VT_positives”字段数值为 0,即无任何反病毒引擎报警,则认为其源于良性软件;②若记录“VT_positives”字段数值超过“VT_engines”字段数值一半,即所采用的反病毒引擎中超过半数报警,则认为其源于恶意软件.抽取 OmniDroid 数据集中符合上述条件的记录,得到恶意与良性软件记录数量比为 1∶1的平衡数据集作为 InterDroid 训练数据支撑.该平衡数据集有恶意与良性软件记录各 5 900 条,其中 80%用于模型训练,20%用于模型测试.

3) 2019 年、2020 年 Android 恶意软件

以开源网站分享的 2019 年、2020 年流行的 Android 恶意软件集合[34-35]为基础,其中 2019 年、2020 年流行的 Android 恶意软件数量分别为 164 个、195 个.保留各年集合中全部满足以下条件的样本:①文件格式为.APK 且无损坏;②源代码 API 级别高于 10;③VirusTotal 包含的反病毒引擎中有半数以上检测其为恶意软件或 VirusTotal 社区人工标注其为恶意软件.最终,得到 2019 年、2020 年流行的 Android 恶意软件安装包个数分别为 151 个、181 个,作为迁移判定模块不同时期 Android 恶意软件样本的数据支撑.以 SHA-1 值为 8aec6f2967b32f03bb1eba93b27d8a70a80d4276 的安装包为例,其 VirusTotal 检测结果如表 2 所示.

4) 评估指标

InterDroid 所述 Android 恶意软件检测模型实质上为二分类模型.由于本文基于 OmniDroid 选取的训练数据为平衡数据,因此准确率(accuracy, *ACC*)是最直观判断能否有效检测 Android 恶意软件的指标.马修斯相关系数(Matthews correlation coefficient, *MCC*)是衡量二分类模型的一个较为

Table 2 Antivirus Test Results of VirusTotal

表 2 VirusTotal 防病毒的检测结果

检测结果	反病毒引擎名称
良性软件	Alibaba, Antiy-AVL, Avira (no cloud), Baidu, BitDefenderTheta, Bkav Pro, ClamAV, Cynet, Cyren, F-Secure, Gridinsoft, Malwarebytes, Jiangmin, K7AntiVirus, Kingsoft, Symantec, ALYac, MaxSecure, NANO-Antivirus, Panda, Qihoo-360, Sangfor Engine Zero, TrendMicro, Rising, Yandex, Zoner, VBA32, VIPRE, SUPERAntiSpyware, TrendMicro-HouseCall, CMC, TACHYON, ViRobot
恶意软件	Ad-Aware, AegisLab, AhnLab-V3, Arcabit, Avast, Avast-Mobile, AVG, BitDefender, BitDefenderFalx, CAT-QuickHeal, Comodo, DrWeb, Emsisoft, eScan, ESET-NOD32, FireEye, Fortinet, GData, Ikarus, K7GW, Kaspersky, McAfee-GW-Edition, Trustlook, MAX, McAfee, Artemis, Microsoft, Sophos, ZoneAlarm by Check Point, Symantec Mobile Insight, Tencent, Zillya

均衡的指标.$F1$ 值($F1$-Measure, $F1$)为 Android 恶意软件检测领域较佳评估指标[36].因此,选取 ACC,$F1$,MCC 三个指标评估 InterDroid,各指标计算为

$$ACC=\frac{TP+TN}{TP+TN+FP+FN}, \tag{23}$$

$$F1=\frac{2\times TP}{2\times TP+FP+FN}, \tag{24}$$

$$MCC=\frac{TP\times TN-FP\times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}, \tag{25}$$

其中,TP 为恶意软件被正确检测的个数,TN 为的良性软件被正确识别的个数,FP 为良性软件被检测为恶意软件的个数,FN 为未能检测到的恶意软件个数.

3.2 特征提取与 TPOT 模型筛选

本节介绍本文实验中使用的数据表与训练模型,即本文实验的数据基础与模型准备.分为高频词提取、Androguard 特征数据提取、TPOT 训练模型筛选 3 部分.

1) 高频词提取

以 Kharon 数据集作为特征种类引入实验的数据支撑,计算其去除冠词、数量词、代词等无意义语料后剩余文本中的高频词汇.无意义语料示例如图 8 所示.绘制词云图并令词频与图中词汇大小成正比,如图 9 所示.

由图 9 中可直观得出:"function""call""class""intent""service""activity""device""code"为报告中高频出现词汇.基于此,本文推断:①因"function""call""class"等词汇均与函数调用相关,故而选取 API 包名作为特征以代表上述词汇.②以汇编指令层 Dalvik 字节码作为特征代表"device""code"等词汇.③在"intent""service""activity"即意图、服务、活动词汇组中,因活动与服务均属于 Android 的四大组件,而意图既可以用于开启活动又可以用于与服务交流,所以选择意图特征代表该词汇组中词汇.

the, is, to, a, and, in, of, also, This, from, has, are, s, which, In, if, used, when, number, data, main, each, your, no, itself, them, our, last, so, like, after, because, executes, try, server, user, its, into, been, any, we…

Fig. 8 Examples of meaningless corpus

图 8 无意义语料示例

Fig. 9 Illustration of wordle

图 9 词云示意图

综上,本文在保留静态检测方法核心权限特征的基础上,融入 API 包名、意图、Dalvik 字节码 3 种特征作为特征种类引入实验结果.

2) Androguard 特征数据提取

通过 Python 整合 Androguard 工具,提取 3.1 节中处理后的 2019 年、2020 年 Android 恶意软件权限、API 包名、意图、Dalvik 字节码 4 种特征数据.其中,在权限特征的提取上:①包含用户自定义权限与 Android API 定义权限.②由于 Android 权限命名规则导致不同权限全称中重复字符占比较高,因此提取特征只保留权限全称中的具体权限名称部分.例如,"android.permission.ACCESS_COARSE_LOCATION""android.permission.CAMERA"提取后分别为"ACCESS_COARSE_LOCATIO""CAMERA".API 包名数据由 Androguard 提取 DEX 文件中类名得到.例如:由 DEX 类名信息中的"Ljava/lang/Object"得到包名"java.lang.Object".本文实验中使用的所有数据表详情如表 3 所示:

Table 3　Detailed Chart of Experimental Data

表 3　实验数据详情表

输入表名	特征种类	维数	数据用途
api	API 包名	184	特征包装与解释模块 不同筛选模型的训练数据
intent	意图	4 185	
opcode	Dalvik 字节码	223	
permission	权限	436	
final_merge	筛选后特征组合	36	本文检测模型训练数据
19_real_malware	筛选后特征组合	36	用于特征迁移的 2019 年 Android 恶意软件数据
real_malware	筛选后特征组合	36	用于特征迁移的 2020 年 Android 恶意软件数据

3) TPOT 训练模型筛选

通过 TPOT 算法筛选得到面向不同输入数据的输出管道,即 2.2 节中多种最佳训练模型.抽取输出管道中出现的训练模型种类,得到 2.2 节中最佳训练模型集合.TPOT 算法筛选结果如表 4 所示.

由表 4 可知:① 基于“api”“intent”“opcode”“permission”数据,计算得到用于 2.2 节中特征组合对比算法筛选步骤的最佳训练模型集合,其包含梯度提升决策树(gradient boosting decision tree, GBDT)、极端随机树(extremely randomized trees, ET)两种训练模型.② 针对“permission”“api”“intent”“opcode”中任一数据,以 GBDT 或 ET 作为训练模型相较于原输出管道在准确率上相差不超过 4%.因此,简化原输出管道为最佳训练模型集合,并将集合中算法模型用于特征包装与解释模块.③ 基于“final_merge”数据,计算得到 GBDT 算法,即 2.2 节检测模型对比算法筛选步骤的输出算法.④ 由于 TPOT 算法限制,输出管道中只包含 sklearn 库中的基础模型,因此将 GBDT, ET 改进后的前沿算法 CatBoost[37] 加入最佳训练模型集合.

Table 4　Results After TPOT Algorithm Filtering

表 4　TPOT 算法筛选结果

输入表名	输出管道	管道准确率/%	选取模型	模型准确率/%
api	*ExtraTreesClassifier*(*CombineDFs*(*CombineDFs*(*input_matrix*, *input_matrix*), *CombineDFs*(*input_matrix*, *input_matrix*)))	97	GradientBoostingClassifier	96
			ExtraTreesClassifier	94
intent	*GradientBoostingClassifier*()	90	GradientBoostingClassifier	90
			ExtraTreesClassifier	87
opcode	*GradientBoostingClassifier*(*GradientBoostingClassifier*(*CombineDFs*(*input_matrix*, *input_matrix*)))	96	GradientBoostingClassifier	96
			ExtraTreesClassifier	92
permission	*ExtraTreesClassifier*(*ExtraTreesClassifier*(*input_matrix*))	97	GradientBoostingClassifier	96
			ExtraTreesClassifier	97
final_merge	*GradientBoostingClassifier*()	98	GradientBoostingClassifier	98

以上,为本文实验提供特征包装、分类检测与对比验证模型.

3.3　SHAP 解释器建立与特征组合稳定性验证

首先,本节给出 SHAP 算法对筛选模型的解释结果,并给出特征包装与解释模块筛选得到的特征组合中每种特征的具体含义,从而验证特征组合的可解释性.其次,对比不同模型得到的特征组合结果,验证特征组合的稳定性.

1) 可解释性验证

通过计算权限、API 包名、意图、Dalvik 字节码 4 种特征中不同特征分量的 SHAP 值,来评估不同特征分量对最终分类效果的贡献度.

以权限特征为例,首先,基于计算所得 SHAP 值,绘制权限特征密度散点图,从而直观地观察不同特征分量如何影响 InterDroid 的分类效果;其次,绘制权限特征重要性 SHAP 值图,从而在量化视角下筛选用于 InterDroid 检测模型的权限特征组合.

图 10 为权限特征密度散点图.其中:纵轴不同行代表权限特征中不同的特征分量,如第 1 行代表“*SEND_SMS*”权限;横轴代表不同分量的 SHAP

值,点灰度由低到高代表特征数值由高到低.例如:第 1 行深灰色样本点对应 SHAP 值为负,浅灰样本点对应 SHAP 值为正,代表"*SEND_SMS*"特征值为正时对分类结果有正向增益,"*SEND_SMS*"特征值为负时对分类结果有负向增益;图中的每个点均代表 1 个用于 InterDroid 权限特征筛选模型训练的 Android 软件.行内样本点的分布数量,代表了该行所对应的特征分量能影响多少样本的分类结果.例如:第 1 行的样本点明显比第 9 行密集,代表在 InterDroid 检测过程中,"*SEND_SMS*"权限比"*RECEIVE*"权限更具有普适性和区分度.

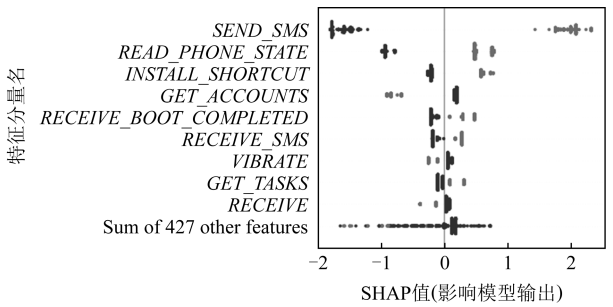

Fig. 10 Permission feature density scatter graph

图 10 权限特征密度散点图

图 11 为权限特征重要性 SHAP 值图.其中,纵轴不同行代表权限特征中不同的特征分量,横轴为该行所代表特征分量的 SHAP 平均绝对值.特征分量的 SHAP 平均绝对值越大代表该特征对分类结果的贡献度越高.

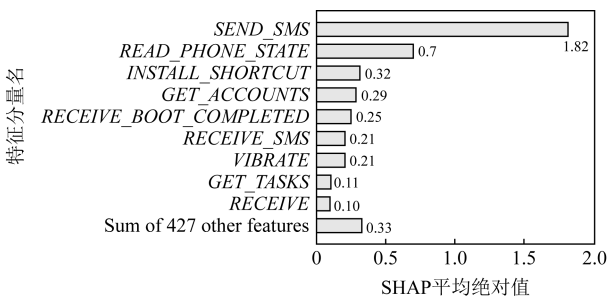

Fig. 11 SHAP value diagram of the importance of permission features

图 11 权限特征重要性 SHAP 值图

由图 10 可知:①使用"*INSTALL_SHORTCUT*""*RECEIVE_BOOT_COMPLETED*""*SEND_SMS*""*READ_PHONE_STATE*""*RECEIVE_SMS*""*GET_TASKS*"等权限,且未使用"*VBRATE*""*RECEIVE*""*GET_ACCOUNTS*"等权限的 Android 应用更易被 InterDroid 检测为恶意软件.②第 1 行至第 9 行样本点的分布数量呈减少趋势,即由"*SEND_SMS*"权限至"*RECEIVE*"权限,对某一 Android 应用是否为恶意软件的区分度逐渐降低.由图 11 可知:第 1 行至第 9 行 SHAP 平均绝对值逐渐降低,即由"*SEND_SMS*"权限至"*RECEIVE*"权限,对模型分类结果的贡献度逐渐降低.

将图 11 中每一行对应到图 10 中,并综合上段结论可得:特征分量的 SHAP 值越高,可检测的 Android 恶意软件范围越大,对 InterDroid 检测结果的贡献度越高.例如:SHAP 值高的特征分量,即图 11 中 SHAP 值最高的"*SEND_SMS*"特征,影响检测数据中大多数 Android 软件的分类结果,即图 10 中"*SEND_SMS*"行样本点分布最密集.据此,计算权限、意图、API 包名、Dalvik 字节码 4 种特征 SHAP 值,以获得不同种类特征中对 InterDroid 检测结果贡献度较高的特征分量集合,结果如表 5 所示:

Table 5 Importance of InterDroid Feature Set

表 5 InterDroid 选取特征组合重要性

特征种类	前 9 分量 SHAP 和	剩余分量 SHAP 和	剩余分量个数	选取分量 SHAP 占比/%
权限	4.01	0.33	427	92
意图	3.51	0.41	4175	90
API 包名	6.62	3.84	175	63
Dalvik 字节码	2.44	0.44	214	85

由表 5 可知,针对权限、意图、Dalvik 字节码 3 类特征,选择每类特征中 SHAP 值排名前 9 的分量即可获得该种特征在 InterDroid 检测结果中 85%以上的贡献度.

针对 API 包名特征,虽然其排名前 9 的特征分量 SHAP 值和在该类特征 SHAP 值总和中所占比重少于 80%.但该类特征中排名第 6,7,8 的特征

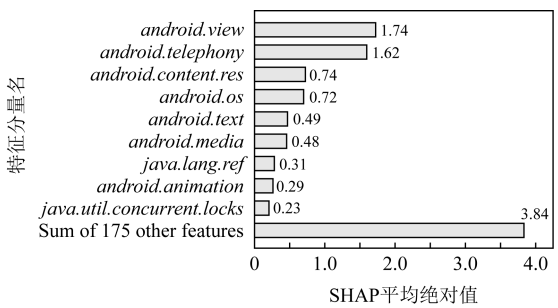

Fig. 12 SHAP value diagram of the importance of API packages features

图 12 API 包名特征重要性 SHAP 值图

分量 SHAP 值分别为 0.31,0.29,0.23,其大小与差距均较小,即排名更靠后的 API 包名特征分量对 InterDroid 检测结果的贡献均较小且差别不大.因此,无需保留排名更加靠后的 API 包名特征分量.API 包名特征重要性 SHAP 值计算结果如图 12 所示.图 12 中横纵轴含义与图 11 相同.

综上,考虑到高维特征提取的时空开销较大,最终选取特征包装与筛选模块中每种特征保留的特征分量数为 9 个.表 6 给出 InterDroid 选用的权限、意图、API 包名 3 类特征中每个特征分量的具体含义.其中,由于 Dalvik 字节码属于汇编指令,其含义较为简单与统一.

Table 6　Meaning of Features Filtered by InterDroid

表 6　InterDroid 筛选得到的特征含义

特征种类	特征分量名	含义
权限	*SEND_SMS*	查看和发送短信
	GET_TASKS	获取运行任务的信息
	READ_PHONE STATE	具备监听电话状态能力
	RECEIVE_BOOT_COMPLETED	开机自启
	RECEIVE_SMS	收接短信并获取内容
	INSTALL_SHORTCUT	添加快捷方式到桌面
	GET_ACCOUNTS	访问账户列表
	VIBRATE	允许程序震动
	RECEIVE	高频的用户自定义权限
意图	*android.provider.Telephony.SMS_RECEIVED'*	阻断短信通知
	android.intent.action.BOOT_COMPLETED	开机自启相关
	android.intent.action.VIEW	显示用户的数据并打开相应活动
	com.google.android.c2dm.intent.RECEIVE	消息推送
	android.intent.action.CREATE_SHORTCUT	创建桌面快捷方式
	android.intent.action.PACKAGE_ADDED	监听新应用安装
	android.intent.action.DATA_SMS_RECEIVED	短信验证注册
	android.intent.action.PHONE_STATE	监听并挂断电话
	android.intent.action.USER_PRESENT	防止程序被用户终止
API 包名	*java.util.concurrent.locks*	提供了锁的相关操作
	android.telephony	涉及通话与数据业务
	android.text	呈现或跟踪文本
	android.os	系统服务、消息传递和进程间通信
	android.content.res	访问应用程序资源、影响应用程序行为
	android.view.inputmethod	生成按键事件、文本
	android.media	管理音频和视频、检测位图中的人脸
	android.animation	提供动画效果
	java.lang.ref	维持对另一对象的引用
Dalvik 字节码	*div-float*	数据运算指令
	xor-int/lit8	
	and-int/2addr	
	rsub-int	
	or-int/lit16	
	rem-long	
	invoke-super/range	方法调用指令
	iput	字段操作指令
	if-nez	跳转指令

2) 稳定性验证

通过将 InterDroid 中 CatBoost 特征筛选算法替换为 TPOT 筛选得到的 GBDT 与 ET 算法,得到 3 组不同的特征筛选结果.以 InterDroid 筛选得到的特征组合为基准,计算 GBDT 算法、ET 算法筛选得到的特征组合与基准算法所得的重合度,来评估

InterDroid 特征包装与解释模块筛选得到的特征组合稳定性[38].对比结果如表 7 所示:

Table 7 Verification of Coincidence Degree of Features

表 7 特征分量重合度验证

筛选算法	特征种类	重合度
GradientBoosting Classifier	权限	7/9
	意图	8/9
	API 包名	6/9
	Dalvik 字节码	4/9
ExtraTreesClassifier	权限	6/9
	意图	6/9
	API 包名	5/9
	Dalvik 字节码	1/9

由表 7 可知:替换 InterDroid 特征包装与解释模块筛选模型算法对权限、意图、API 包名 3 类特征筛选结果造成的扰动较小,权限、意图、API 包名 3 类特征的筛选结果具有稳定性.针对 Dalvik 字节码特征,结合表 6 可知,即使属于同一类指令的 Dalvik 字节码,对不同寄存器的操作也会导致其具体特征分量名称不同.例如:"iget-wide""iget-object""iget-boolean""iget-byte"等不同 Dalvik 字节码同属于字段操作指令,且均用于对所标识的字段执行所标识的对象实例字段操作,将其加载或存储到值寄存器中.因此,当以高检测准确率与低存储消耗为目标,仅选择 SHAP 值前 9 的 Dalvik 字节码类特征分量时,筛选出的 Dalvik 字节码特征稳定性较差.

综上,InterDroid 特征包装与解释模块筛选得到的权限、意图、API 包名 3 类特征具备较好的可解释性与稳定性.由于 Dalvik 字节码特征本质为汇编指令,具有含义简要、种类繁多等特点,故筛选得到的 Dalvik 字节码特征含义较为简单,稳定性较差.

3.4 特征迁移

首先,本节介绍双重分布检验模块对 3.1 节中 2019 年、2020 年流行的 Android 恶意软件样本的检验结果,并据此验证 2018—2020 年间 Android 恶意软件特征存在概念漂移现象.其次,通过横纵向对比不同算法在不同时期 Android 恶意软件数据上的表现,验证特征迁移对 InterDroid 检测模型准确率提升的有效性.

1) 特征漂移存在性验证

首先,通过 MWU 检验比较 final_merge 与 19_real_malware,real_malware 两表数据分布,即比较 2018 年的 InterDroid 训练数据与 2019 年、2020 年流行 Android 恶意软件对应数据分布是否相同,从而评估 2018—2020 年间 Android 恶意软件特征是否存在概念漂移现象.MWU 检验结果如表 8 所示.

由表 8 可得:①相较于 2018 年,2019 年、2020 年 Android 恶意软件特征分布具有较大改变,不同分布特征占比高,存在概念漂移现象.②2019 年、2020 年均存在少量与 2018 年同分布的特征,其原因可能为部分攻击者采用代码重用、代码改进等方式快速构造 Android 恶意软件.③"*GET_TASKS*""*VIBRATE*""*java.util.concurrent.locks*"三个特征在 2018—2020 年间保持同一分布,具备时间稳定性.④上述 3 个具备时间稳定性的特征存在于 InterDroid 筛选得到的特征组合中,进一步验证了 InterDroid 在缓解概念漂移问题上的有效性.

Table 8 MWU Test Results

表 8 MWU 检验结果

对比年份	不同分布占比	同分布特征分量
2018 年,2019 年	32/36	*GET_TASKS*,*VIBRATE*,*java.util.concurrent.locks*,*android.media*
2018 年,2020 年	29/36	*if-nez*,*android.intent.action.CREATE_SHORTCUT*,*GET_TASKS*,*android.os*,*RECEIVE_BOOT_COMPLETED*,*RECEIVE_SMS*,*VIBRATE*,*java.util.concurrent.locks*

其次,通过 19_real_malware,real_malware 两表数据测试 InterDroid 检测模型,即测试 InterDroid 对 2019 年、2020 年流行的 Android 恶意软件的检测准确率,从而评估概念漂移现象是否降低了本文模型检测准确率.结果表明,相较于 2018 年同时期测试数据,本文检测模型对 2019 年 Android 恶意软件检测准确率下降 62%,对 2020 年 Android 恶意软件检测准确率下降 53%.其中,2020 年下降幅度稍低,与表 7 中 2020 年数据中同分布特征相较于 2019 年较多的表述相一致,侧面验证了概念漂移现象对 InterDroid 检测准确率的负面影响.算法评估结果将在本节下文特征迁移有效性验证部分详细说明.

2) 特征迁移有效性验证

首先,通过纵向比较 InterDroid 与 TPOT 筛选得到的最优算法 GBDT、原 JDA 伪分类器构造算法 KNN、普遍使用于二分类问题的支持向量机(support

vector machines, SVM)等算法在 2018 年 Android 恶意软件检测问题上的表现,来评估 InterDroid 对同时期 Android 恶意软件的检测效果.其次,选取在纵向比较中与 InterDroid 表现相近的算法,通过横向对比 InterDroid 与上述算法在特征迁移前后对 2019 年、2020 年 Android 恶意软件的检测准确率,从而评估 InterDroid 在缓解概念漂移问题上的优越性.结果如表 9、表 10 所示:

Table 9　Results of 2018 Android Malware Detection

表 9　2018 年 Android 恶意软件检测效果

算法	*ACC*/%	*MCC*	*F*1
InterDroid	96	0.89	0.96
GradientBoosting Classifier	98	0.93	0.98
KNN	93	0.80	0.93
SVM	85	0.66	0.86

Table 10　Results of Feature-Representation Transfer

表 10　特征迁移效果

算法	数据	迁移前 *ACC*/%	迁移后 *ACC*/%
InterDroid	19_real_malware	34	80
	real_malware	43	87
GradientBoosting Classifier	19_real_malware	17	60
	real_malware	22	61
KNN	19_real_malware	20	28
	real_malware	8	29

由表 9、表 10 可知:①针对同时期 Android 恶意软件检测问题,InterDroid 表现明显优于 SVM;*MCC* 值优于 KNN;虽然与 TPOT 筛选得到的最优算法相较略差,但 *ACC* 与 *F*1 值仅相差 0.02,*MCC* 值仅相差 0.03.②针对不同时期 Android 恶意软件检测问题,InterDroid 在特征迁移后检测准确率分别提升 46%,44%,提升后准确率达 80%,87%,明显优于纵向比较中表现相近的 GBDT 与 KNN.

因此,不同时期 Android 恶意软件特征存在概念漂移问题,但存在部分特征具备时间稳定性,其原因可能为代码重用、改进等 Android 恶意软件构造方式.而 InterDroid 能在保证对同时期 Android 恶意软件检测效果的同时有效缓解概念漂移问题.

以特征迁移前 InterDroid、火绒安全[39]、腾讯电脑管家[40]、金山毒霸[41]、360 安全卫士[42]、瑞星杀毒软件[43]等检测工具为基准,以 *ACC* 为评估指标,基于 2019 年、2020 年真实 Android 恶意软件数据,进一步验证 InterDroid 能够有效缓解概念漂移问题,稳定检测 Android 恶意软件.检测结果如图 13 所示,InterDroid$_0$ 即特征迁移前 InterDroid.

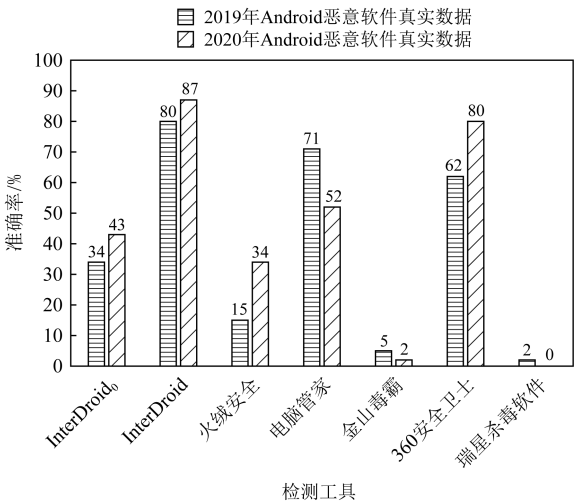

Fig. 13　Comparison of *ACC* of detection tools

图 13　检测工具准确率对比图

由图 13 可知,由于受概念漂移问题影响,针对 2019 年、2020 年真实 Android 恶意软件,火绒安全、金山毒霸、瑞星杀毒软件及特征迁移前 InterDroid 对新兴 Android 恶意软件的检测准确率均较低.而特征迁移后的 InterDroid、电脑管家、360 安全卫士对新兴 Android 恶意软件仍能保持较高的检测准确率,有效缓解了概念漂移问题,且 InterDroid 的检测准确率最优.

综上,特征迁移能够有效缓解概念漂移问题,提升 InterDroid 对新时期 Android 恶意软件的检测准确率.

4　结论与未来工作

本文提出了面向概念漂移的可解释性 Android 恶意软件检测方法.首先,该方法以人工 Android 恶意软件分析报告中高频词汇为指导,引入权限、API 包名、意图、Dalvik 字节码 4 种特征.同时,融合自动化机器学习 TPOT 算法以弱化模型选取及参数调整过程中的不确定性,从而获得 CatBoost 作为 InterDroid 检测算法及 GBDT、ET 作为对比验证算法.其次,通过 SHAP 模型解释算法改进特征包装过程,得到高贡献度特征与其含义,为逆向分析人员验证模型与人工分析提供依据.最终,以本文检测模型检验及 MWU 检验验证不同时期 Android 恶意软件特征存在的概念漂移问题,并基于 JDA 算法提升已有模型对新兴 Android 恶意软件检测准确率.

实验结果表明,针对同时期 Android 恶意软件,InterDroid 检测准确率为 96%.针对新兴 Android 恶意软件,特征迁移后 InterDroid 检测准确率可分别提升 46%,44%.且 InterDroid 筛选出的特征组合包含 2018—2020 年分布稳定的 Android 恶意软件特征.

在未来的工作中,我们将引入自然语言处理领域中的注意力机制,进一步减少特征选取阶段的人工干预.同时,扩展本文检测方法,从而查杀攻击行为不由软件本身发起的具有多层验证机制的 Android 恶意软件或诱导型 Android 恶意软件.

参 考 文 献

[1] 360 Internet Security Center. China mobile phone security report for the first quarter of 2021 [R/OL]. 2021 [2021-05-04]. https://rs-beijing.oss.yunpan.360.cn/Object.getFile/360report/MjAyMeW5tOesrOS4gOWto+W6puS4reWbveaJi+acuuWuieWFqOeKtuWGteaKpeWRii5wZGY= (in Chinese)
(360 互联网安全中心. 2021 年第一季度中国手机安全报告 [R/OL]. 2021 [2021-05-04]. https://rs-beijing.oss.yunpan.360.cn/Object.getFile/360report/MjAyMeW5tOesrOS4gOWto+W6puS4reWbveaJi+acuuWuieWFqOeKtuWGteaKpeWRii5wZGY=)

[2] Statcounter. Mobile operating system market share China [EB/OL]. 2021 [2021-05-04]. https://gs.statcounter.com/os-market-share/mobile/china

[3] Zhang Bing, Ren Jiadong, Wang Ning. Network security risk assessment method: A review [J]. Journal of Yanshan University, 2020, 44(3): 290-305 (in Chinese)
(张炳, 任家东, 王苧. 网络安全风险评估分析方法研究综述 [J]. 燕山大学学报, 2020, 44(3): 290-305)

[4] Peruma A, Palmerino J, Krutz D E. Investigating user perception and comprehension of Android permission models [C] //Proc of the 5th Int Conf on Mobile Software Engineering and Systems. Piscataway, NJ: IEEE, 2018: 56-66

[5] Molnar C. Interpretable Machine Learning [M]. Morrisville, NC: Lulu Press, 2019: 11-22

[6] Google. Distribution dashboard [EB/OL]. 2021 [2021-05-04]. https://developer.android.com/about/dashboards

[7] Kiss N, Lalande J F, Leslous M, et al. Kharon dataset: Android malware under a microscope [C] //The {LASER} Workshop: Learning from Authoritative Security Experiment Results ({LASER} 2016). Berkeley, CA: USENIX Association, 2016: 1-12

[8] Le T Trang, Fu Weixuan, Moore J H. Scaling tree-based automated machine learning to biomedical big data with a feature set selector [J]. Bioinformatics, 2020, 36(1): 250-256

[9] Olson R S, Urbanowicz R J, Andrews P C, et al. Automating biomedical data science through tree-based pipeline optimization [C] //Proc of European Conf on the Applications of Evolutionary Computation. Berlin: Springer, 2016: 123-137

[10] Olson R S, Bartley N, Urbanowicz R J, et al. Evaluation of a tree-based pipeline optimization tool for automating data science [C] //Proc of the Genetic and Evolutionary Computation Conf. New York: ACM, 2016: 485-492

[11] Lundberg S, Lee S I. Aunified approach to interpreting model predictions [C] //Proc of the 31st Annual Conf on Neural Information Processing Systems. Cambridge, MA: MIT Press, 2017

[12] Lundberg S M, Erion G, Chen H, et al. From local explanations to global understanding with explainable AI for trees [J]. Nature Machine Intelligence, 2020, 2(1): 56-67

[13] Lundberg S M, Nair B, Vavilala M S, et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery [J]. Nature Biomedical Engineering, 2018, 2(10): 749-760

[14] Nachar N. The Mann-Whitney U: A test for assessing whether two independent samples come from the same distribution [J]. Tutorials in Quantitative Methods for Psychology, 2008, 4(1): 13-20

[15] Long Mingsheng, Wang Jianmin, Ding Guiguang, et al. Transfer feature learning with joint distribution adaptation [C] //Proc of the IEEE Int Conf on Computer Vision. Piscataway, NJ: IEEE, 2013: 2200-2207

[16] Long Mingsheng, Zhu Han, Wang Jianming, et al. Deep transfer learning with joint adaptation networks [C] // Proc of Int Conf on Machine Learning. Sydney, NSW, Australia: PMLR, 2017: 2208-2217

[17] Zhang Yuqing, Wang Kai, Yang Huan, et al. Survey of Android OS security [J]. Journal of Computer Research and Development, 2014, 51(7): 1385-1396 (in Chinese)
(张玉清, 王凯, 杨欢, 等. Android 安全综述[J]. 计算机研究与发展, 2014, 51(7): 1385-1396)

[18] Felt A P, Chin E, Hanna S, et al. Android permissions demystified [C] //Proc of the 18th ACM Conf on Computer and Communications Security. New York: ACM, 2011: 627-638

[19] Bartel A, Klein J, Le Traon Y, et al. Automatically securing permission-based software by reducing the attack surface: An application to Android [C] //Proc of the 27th IEEE/ACM Int Conf on Automated Software Engineering. Piscataway, NJ: IEEE, 2012: 274-277

[20] Karim M Y, Kagdi H, Di Penta M. Mining Android apps to recommend permissions [C] //Proc of 2016 IEEE 23rd Int Conf on Software Analysis, Evolution, and Reengineering (SANER). Piscataway, NJ: IEEE, 2016, 1: 427-437

[21] Olukoya O, Mackenzie L, Omoronyia I. Security-oriented view of app behaviour using textual descriptions and user-granted permission requests [J]. Computers & Security, 2020, 89: 101685

[22] Wang Run, Wang Zhibo, Tang Benxiao, et al. Smartpi: Understanding permission implications of Android apps from user reviews [J]. IEEE Transactions on Mobile Computing, 2019, 19(12): 2933-2945

[23] Ilham S, Abderrahim G, Abdelhakim B A. Permission based malware detection in Android devices [C] //Proc of the 3rd Int Conf on Smart City Applications. New York: ACM, 2018: 1-6

[24] Alecakir H, Can B, Sen S. Attention: There is an inconsistency between Android permissions and application metadata! [J]. International Journal of Information Security, 2021, 2021(3): 1-19

[25] Li Jin, Sun Lichao, Yan Qiben, et al. Significant permission identification for machine-learning-based Android malware detection [J]. IEEE Transactions on Industrial Informatics, 2018, 14(7): 3216-3225

[26] Yuan Hongli, Tang Yongchuan. MADFU: An improved malicious application detection method based on features uncertainty [J]. Entropy, 2020, 22(7): No.792

[27] Chen Hui, Li Zhengqiang, Jiang Qingshan, et al. A hierarchical approach for Android malware detection using authorization-sensitive features [J]. Electronics, 2021, 10(4): No.432

[28] Ünver H M, Bakour K. Android malware detection based on image-based features and machine learning techniques [J]. SN Applied Sciences, 2020, 2(7): 1-15

[29] Wang Wei, Wang Xing, Feng Dawei, et al. Exploring permission-induced risk in Android applications for malicious application detection [J]. IEEE Transactions on Information Forensics and Security, 2014, 9(11): 1869-1882

[30] Fan Ming, Liu Jian, Liu Jun, et al. Review of Android malware detection methods [J]. SCIENTIA SINICA Informationis, 2020, 50(8): 1148-1177 (in Chinese)
(范铭, 刘烃, 刘均, 等. 安卓恶意软件检测方法综述[J]. 中国科学: 信息科学, 2020, 50(8): 1148-1177)

[31] Onwuzurike L, Mariconti E, Andriotis P, et al. MaMaDroid: Detecting Android malware by building Markov chains of behavioral models [J]. ACM Transaction on Information and System Security, 2019, 22(2): 14.1-14.34

[32] Desnos A. Androguard [CP/OL]. 2011 (2020-11-24) [2021-04-07]. https://github.com/androguard/androguard

[33] Martín A, Lara-Cabrera R, Camacho D. Android malware detection through hybrid features fusion and ensemble classifiers: The AndroPyTool framework and the OmniDroid dataset [J]. Information Fusion, 2019, 52: 128-142

[34] Anonymous. AndroidMalware_2019 [DB/OL]. (2020-01-02) [2021-04-07]. https://github.com/sk3ptre/AndroidMalware_2019#readme

[35] Anonymous. AndroidMalware_2020 [DB/OL]. (2021-01-07) [2021-04-07]. https://github.com/sk3ptre/AndroidMalware_2020

[36] Lei Cen, Gates C S, Luo Si, et al. A probabilistic discriminative model for Android malware detection with decompiled source code [J]. IEEE Transactions on Dependable and Secure Computing, 2014, 12(4): 400-412

[37] Dorogush A V, Ershov V, Gulin A. CatBoost: Gradient boosting with categorical features support [J]. arXiv preprint arXiv:1810.11363, 2018

[38] Zhang Bing, Huang Guoyan, He Haitao, et al. Approach to mine influential functions based on software execution sequence [J]. IET Software, 2017, 11(2): 48-54

[39] Beijing Huorong Network Technology Co. Ltd. Huorong Security [CP/OL]. Beijing: Beijing Huorong Network Technology Co. Ltd., 2021 (2021-07-27) [2021-07-27]. https://www.huorong.cn/person5.html (in Chinese)
(北京火绒网络科技有限公司. 火绒安全[CP/OL]. 北京: 北京火绒网络科技有限公司. 2021 (2021-07-27) [2021-07-27]. https://www.huorong.cn/person5.html)

[40] Shenzhen Tencent Computer System Co. Ltd. Tencent PC Manager [CP/OL]. Shenzhen: Shenzhen Tencent Computer System Co. Ltd., 2021 (2021-02-04) [2021-07-27]. https://guanjia.qq.com/ (in Chinese)
(深圳市腾讯计算机系统有限公司. 腾讯电脑管家[CP/OL]. 深圳: 深圳市腾讯计算机系统有限公司. 2021 (2021-02-04) [2021-07-27]. https://guanjia.qq.com/)

[41] Beijing Cheetah Network Technology Co. Ltd. Jinshan Duba [CP/OL]. Beijing: Beijing Cheetah Network Technology Co. Ltd., 2021 (2021-06-24) [2021-07-27]. https://www.ijinshan.com/ (in Chinese)
(北京猎豹网络科技有限公司.金山毒霸[CP/OL]. 北京: 北京猎豹网络科技有限公司. 2021 (2021-06-24) [2021-07-27]. https://www.ijinshan.com/)

[42] 360 Internet Security Center. 360 Security Guard [CP/OL]. Beijing: 360 Internet Security Center. 2019 (2019-06-20) [2021-07-27]. https://weishi.360.cn/?source=homepage/ (in Chinese)
(360 互联网安全中心. 360 安全卫士[CP/OL]. 北京: 360 互联网安全中心. 2019 (2019-06-20) [2021-07-27]. https://weishi.360.cn/?source=homepage/)

[43] Beijing Rising Net an Technology Co. Ltd. Rising antivirus software [CP/OL]. Beijing: Beijing Rising Net an Technology Co. Ltd., 2021 (2021-07-26) [2021-07-27]. http://pc.rising.com.cn/ (in Chinese)

(北京瑞星网安技术股份有限公司. 瑞星杀毒软件[CP/OL]. 北京: 北京瑞星网安技术股份有限公司. 2021 (2021-07-26) [2021-07-27]. http://pc.rising.com.cn/)

Zhang Bing, born in 1989. PhD, associate professor. Member of CCF and ACM. His main research interests include data mining, machine learning, and software security.

张 炳, 1989 年生. 博士, 副教授. CCF, ACM 会员. 主要研究方向为数据挖掘、机器学习、软件安全.

Wen Zheng, born in 1998. Master candidate. Student member of ACM. His main research interest is software security.

文 峥, 1998 年生. 硕士研究生. ACM 学生会员. 主要研究方向为软件安全.

Wei Xiaoyu, born in 1984. Master. Senior engineer. Her main research interests include intelligent guidance system, integration test, comprehensive test verification and digital network support.

魏筱瑜, 1984 年生. 硕士, 高级工程师. 主要研究方向为智能化制导系统、集成测试、综合试验验证、数字化网络支撑.

Ren Jiadong, born in 1967. PhD, professor. Senior member of CCF, member of IEEE and ACM. His main research interests include data mining, temporal data modeling, and software security.

任家东, 1967 年生. 博士, 教授. CCF 高级会员, IEEE 和 ACM 会员. 主要研究方向为数据挖掘、时态数据建模、软件安全.