基于敏感特征深度域关联的 Android 恶意应用检测方法

姜建国[1,2], 李 松[1,2], 喻 民[1,2], 李 罡[3], 刘 超[1], 李梅梅[1], 黄伟庆[1]

[1] 中国科学院信息工程研究所 北京 中国 100093
[2] 中国科学院大学 网络空间安全学院 北京 中国 100093
[3] 迪肯大学 信息技术学院 吉朗 澳大利亚

摘要 利用机器学习或深度学习算法进行 Android 恶意应用的检测是当前主流方法, 取得了一定的效果。然而, 多数方法仅关注应用的权限和敏感行为等信息, 缺乏对敏感行为协同的深度分析, 导致恶意应用检测准确率低。对敏感行为协同深度分析的挑战主要有两个: 表征敏感特征域关联和基于敏感特征域关联的深层分析与检测。本文提出了一种新的 Android 恶意应用检测模型 GCNDroid, 基于敏感特征域关联关系图描述的应用程序主要敏感行为以及敏感行为之间的域关联关系来有效地检测 Android 恶意应用。首先, 为了筛选出对分类更加敏感的特征, 同时减少图节点的数量, 加速分析, 本文构建了敏感特征字典。接着, 定义类或者包为域, 在同一个域中的敏感特征具有域关联关系。通过敏感特征所在域的相对范围, 构造敏感特征之间不同的域关联权重, 生成敏感特征域关联关系图, 敏感特征域关联关系图可以准确表征特定功能模块中的敏感行为, 以及敏感行为之间的完整关系。然后, 基于敏感特征域关联关系图, 设计基于图卷积神经网络的深度表征, 构建 Android 恶意应用检测模型 GCNDroid。在实践中, GCNDroid 还可以利用新的敏感特征不断更新, 以适应移动应用程序新的敏感行为。最后, 本文对 GCDNroid 进行了系统评估, 召回率、调和平均数、AUC 等重要指标均超过 96%。与传统的机器学习算法(支持向量机和决策树)和深度学习算法(深度神经网络和卷积神经网络)相比, GCNDroid 取得了预期的效果。

关键词 Android 恶意应用; 域关联; 图卷积神经网络; 敏感特征
中图法分类号 TP309.5 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2023.01.01

Android Malware Detection Approach Based on Deep Domain Correlation of Sensitive Features

JIANG Jianguo[1,2], LI Song[1,2], YU Min[1,2], LI Gang[3], LIU Chao[1], LI Meimei[1], HUANG Weiqing[1]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100093, China
[3] School of Information Technology, Deakin University, Geelong, VIC 3216, Australia

Abstract The approaches based on traditional machine learning or deep learning algorithms are popular for Android malware detection, however, the majority of existing approaches focus only on the permissions of applications and sensitive APIs, and still lack in-depth analysis of the coordination of sensitive behaviors, resulting in low accuracy. There are two main challenges to study Android applications based on domain correlation: characterizing sensitive feature domain correlation and deep analysis and detection based on sensitive feature domain correlations. In this paper, we propose a new Android malware detection model called GCNDroid, which is based on the main sensitive behaviors of the application described by the sensitive feature domain correlation graph, and the domain correlation between sensitive behaviors to effectively detect Android malware. First, in order to filter out the features that are more sensitive to classification, and reduce the number of graph nodes to make the analysis faster, a dictionary of sensitive features is constructed in this paper. Then, we define a class or package as a domain, and sensitive features in the same domain have a domain correlation. Through the relative range of the sensitive feature's domain, we construct various domain correlation weights between the sensitive features, and generate the sensitive feature domain correlation graph, which can accurately characterize the sensitive behaviors in a specific functional module and the complete relationship between sensitive behaviors. Then, based on the graph, we design a deep representation with graph convolutional neural network to construct the Android malware detection model GCNDroid. In practice, GCNDroid can also be constantly updated using new features, which can adapt to the new sensitive behaviors of mobile apps. Finally, extensive evaluations of GCNDroid have been done, compared with traditional machine learning algorithms(SVM and Decision Tree) and deep learning algorithms(DNN and CNN) and the

通讯作者: 喻民, 博士, 高级工程师, 硕导, Email: yumin@iie.ac.cn。
本课题得到中国科学院青年创新促进会(No. 2021155)资助。
收稿日期: 2020-08-19; 修改日期: 2020-11-09; 定稿日期: 2022-12-07

results show that GCNDroid achieves high agreement on Android malware detection, in which the recall, f1-score, AUC, etc. all exceed 96%.

Key words android malware; domain correlation; GCN; sensitive features

1 引言

Android 恶意应用发展迅速, 其复杂性和影响范围不断增加, 已经成了网络空间最严峻的威胁之一。一方面, Android 系统的设备持续增长, 预计到 2023 年, Android 智能手机设备的出货量将达到 13 亿[1]; 越来越多的物联网设备也相继搭载 Android 系统, 如冰箱、智能灯泡、智能插座、家用机器人等, 进一步拓展了 Android 设备的范围。另一方面, Android 恶意应用威胁仍十分严峻, 2019 年全年, 360 安全大脑累计拦截恶意程序攻击约 9.5 亿次, 平均每天拦截手机恶意程序攻击约 259.2 万次[2]。因此, Android 恶意应用检测仍面临严峻的挑战, 对 Android 恶意应用检测领域的进一步研究具有十分重要的意义。

现有的研究方法已经在 Android 恶意应用检测方面取得了一定的成功。基于权限的方法[3-7]通常从 AndroidManifest 文件中提取声明的权限、组件等相关信息, 或者从源代码中提取使用的权限作为特征进行分析。敏感应用程序接口(Application programming interface, API)通常映射为应用程序的某些敏感行为。因此, 基于敏感 API 的方法始终具有良好的分类性能。这些方法提取代码中的敏感 API 调用, 基于敏感 API 使用情况和频率来构建统计特征[8-10]; 基于敏感 API 或者操作码的序列构建序列特征[11-15]; 基于敏感 API 之间的调用关系来构造图或子图特征[16-18]。最后, 使用传统的机器学习(例如支持向量机(Support vector machine, SVM)、决策树(Decision tree, DT)[3-4,8-9,16,18])或深度学习(例如深度神经网络(Deep neural networks, DNN)[19-20]、卷积神经网络(Convolutional neural network, CNN)[21-23])算法构建分类模型, 实现 Android 恶意应用检测。但是, 这些方法仍存在一定的局限性, 如没有考虑不同敏感行为在同一类或者包空间中的协同作用。我们定义域来表示类或者包空间, 在同一个域中的敏感特征具有域关联关系。本文拟基于域关联关系对上述局限性进行研究, 主要存在如下挑战:

(1) 对敏感特征域关联表征。Android 恶意应用通常监听系统或者应用的广播信息选择适合的时机, 通过调用一系列敏感的 API 执行相应敏感的行为, 从而实现恶意目的。所以其恶意行为多基于多个敏感特征的关联协同[16]。同时, Android 应用开发者往往将具有特定功能的代码组织到特定的域, 主要体现在将多个相关方法组织到同一个类, 将多个相关类组织到相同的包空间, 实现代码的有效管理[24]。同理, Android 恶意应用开发者通过将恶意负载组织到相同的域插入到良性应用中, 从而实现恶意应用的快速开发与变种[25-28]。通过域来对敏感特征进行关联, 可以进一步表征特定功能模块, 准确刻画 Android 恶意应用的恶意负载。如何表征域关联敏感特征有待进一步研究。

(2) 基于敏感特征域关联的深层分析与检测。构建敏感特征域关联表征有利于刻画 Android 恶意应用特定的恶意负载模式, 提高模型的检测能力。但现有的基于机器学习或者深度学习的检测、分析方法通常基于离散的、欧几里得空间的敏感特征构建检测、分析模型。通过建立权限、敏感 API 等特征的多维向量表示, 用相应机器学习或者深度学习算法提取向量表示中的统计行为或者深层的隐藏模式, 实现最终的决策判定[3-12,16-18]。这些方法往往无法有效分析非欧几里得数据[29-31], 丢失了对检测有用的敏感特征之间的域关联关系这种图形结构的关联信息, 无法准确表示敏感特征和敏感特征之间完整的行为模式, 进而导致检测模型的检测能力受限。

为了解决上述挑战, 本文提出一种新的 Android 恶意应用检测模型 GCNDroid。涉及敏感特征和敏感特征的域关联关系, 本文基于图来表征应用, 图的节点存储敏感特征信息, 图的边表示敏感特征的域关联。图卷积神经网络(Graph convolutional network, GCN)可以通过获取节点邻域信息的加权平均值来执行图卷积, 实现邻域节点信息的有效关联和节点信息的深层分析, 所以本文最后基于 GCN 实现最终的检测模型。本文将从以下两方面展开介绍: (1)构建敏感特征域关联关系图; (2)基于 GCN 的 Android 恶意应用检测。特别的, 本文构建敏感特征域关联关系图(Sensitive feature domain correlation graph, SCG)来表示敏感行为的域关联关系。首先, 基于敏感 API、硬编码的 action 等信息建立敏感特征字典表示敏感行为集合。提取样本中存在于上述特征字典的敏感特征集合, 根据任意两个敏感特征所在类、包的相对深度为敏感特征之间建立不同级别的边, 并设置相应阈值(这里的相对深度表示包含任意的两个敏感特征的包集合中, 包目录树中深度最深的包与这两个敏

感特征父包的相对路径, 较长的相对路径), 从而建立敏感特征域关联关系图 SCG。图卷积神经网络 GCN 是卷积神经网络 CNN 的扩展, 在图域中, 旨在学习节点特征信息和结构信息的端到端模型, 可以有效处理 SCG 等非欧几里得结构的数据, 实现特征的离散卷积。因此, 我们基于 GCN 关联敏感特征及其具有域关联关系的敏感特征, 提取 SCG 中节点与边深层的隐藏模式, 构建基于 GCN 的图分类模型 GCNDroid, 实现 Android 恶意应用的检测。

经实验分析, GCNDroid 在 Android 恶意应用检测方面展现了其优越性。本文的贡献可分为以下三点:

(1) 本文提出了一种基于敏感特征域关联的特征 SCG, 用来准确表征特定功能模块(包括恶意负载)中的敏感行为, 敏感行为之间的完整关系, 作为后续检测分析的基础。

(2) 本文提出了一种基于敏感特征深度域关联分析的检测模型 GCNDroid, 有效结合了 SCG 中敏感特征与敏感特征之间的关联信息, 弥补了当前检测方法中的不足。

(3) 本文实现了基于 GCNDroid 的 Android 恶意应用检测系统, 实验表明, 该系统可以有效提高 Android 恶意应用的检测能力(召回率、调和平均数、AUC 值均达到 96%以上)。

2 研究背景

该节介绍本文必要的研究背景: 包机制, 敏感特征等。

2.1 包机制

Java 是 Android 开发的官方语言之一, 由 Android Studio 提供支持。Java 或者其他 Android 应用开发语言为了更好地组织类, 提供了包机制, 用于区别类名的命名空间[32]。包的主要作用有: (1) 把功能相似或相关的类或接口组织在同一个包中, 方便类的查找和使用。(2) 如同文件夹一样, 包也采用了树形目录的存储方式。同一个包中的类名字是不同的, 不同的包中的类的名字是可以相同的, 当同时调用两个不同包中相同类名的类时, 应该加上包名加以区别。因此, 包可以避免名字冲突。(3) 包也限定了访问权限, 拥有包访问权限的类才能访问某个包中的类。同一个包中的类可以访问彼此的包私有成员和受保护成员。Android 应用开发使用包这种机制是为了防止命名冲突, 访问控制, 提供搜索和定位类、接口、枚举和注释等。包机制允许开发人员将类(和接口)组合在一起。这些类都将以某种方式相互关联—常常与特定的应用程序或执行一组特定的任务有关。为了方便代码的组织与管理, 程序员通常使用包或类形成特定的域来组织特定的功能。

2.2 敏感特征

Android 为应用开发提供了各式各样的 API, 用于实现多种复杂, 丰富的功能。而 Android 恶意应用往往利用敏感 API 执行恶意行为。例如, 利用用于发送短信的 API android.telephony.smsmsmanager. sendTextMessage 传输隐私数据; 利用 API java.lang.Runtime 运行时调用.exec 触发外部指令, 执行特定敏感字符串命令, 进而实现隐蔽的、恶意的行为; 通过"/system/bin/sh"等命令执行特定的敏感 shell 脚本。由于敏感 API 直接或间接映射 Android 恶意应用的敏感行为, 经常作为恶意应用检测的主要特征进行分析[33-35]。

广播是在 Android 应用程序之间传输信息广泛使用的一种机制[36]。广播接收器 Broadcast Receiver 是对广播进行过滤和响应的组件。当需要发送消息时, 通过调用 SendOrderBroadcast 或 SendStickyBroadcast 方法将操作和类别等筛选信息加载到意图 Intent 对象中。这里, Intent 用于描述在某个 Intent 对象中执行的简单操作(如"查看地图"或"拍摄照片"), 用来启动另一应用中的某个 Activity, 表示指定的一项操作并提供执行该操作所需的一些数据[37]。在这里, Intent 的操作主要用 action 来描述。Intent 发送后, 所有注册的广播接收器将检查注册的 IntentFilter 是否与发送的意图匹配。一旦匹配, 将调用组件中的 OnReceive 方法。Android 恶意应用通常通过活动、服务等组件监听匹配系统或者其他应用的 Intent 的 action 等敏感行为, 选择合适的时机执行恶意负载[9]。

Android 恶意应用开发者往往使用多个敏感特征行为的组合开发恶意负载, 达到恶意目的。同时, 通过包或类机制将相应的敏感特征集合组织到特定的域中。

3 系统架构

本文提出了一种基于敏感特征深度域关联分析的 Android 恶意应用检测方法 GCNDroid。该方法的整体架构如图 1 所示, 主要分为以下三个模块: 预处理模块、特征工程模块、检测模型模块。其中, 预处理模块通过 Androguard[38]等工具将 Android 应用解压缩、反编译为可读的 smali 代码, 用于后续敏感特征字典的构建等。敏感特征字典的构建主要通过分析恶意应用中出现的频率高于良性应用中的硬编码的 API、action 和涉及敏感操作的字符串, 形成恶意

应用敏感特征集, 进而构建敏感特征字典, 用于后续的检测分析。特征工程模块基于上述的敏感特征字典, 提取 smali 代码中存在于字典中的敏感特征, 形成该应用的敏感特征集。结合敏感特征集、敏感特征所在的文件路径, 形成特定的敏感特征树。基于敏感特征集, 初始化域关联关系图节点, 结合敏感特征树, 设定相对深度阈值, 初始化域关联关系图中不同节点的边, 从而构建敏感特征域关联关系图 SCG。在检测模型模块, 基于自定义的权重邻接矩阵构建图卷积模式, 在此基础上, 构建基于 GCN 的分类器。接着, 用上述训练样本的域关联关系图对分类器进行训练、微调等, 形成 Android 恶意应用检测模型 GCNDroid。进入模型的测试阶段, 待检测应用经过预处理、特征工程模块生成相应的域关联关系图, 输入到GCNDroid检测模型中, 实现对待检测应用的恶意性检测。

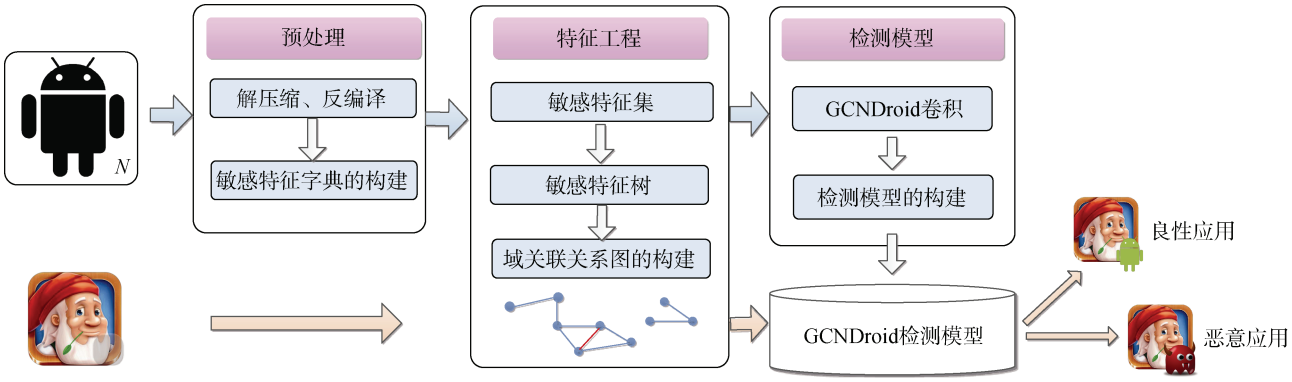


图 1 GCNDroid 检测系统框架图

Figure 1 GCNDroid detection system framework

4 关键技术与方法

本章节主要从三个方面介绍本文的关键技术与方法: 敏感特征字典的构建、域关联关系图的构建和 GCNDroid 检测模型。

4.1 敏感特征字典的构建

4.1.1 敏感特征的选择

为了筛选出对分类更加敏感的特征, 同时减少图节点的数量, 加速分析, 本文进行了敏感特征的选择。首先, 基于开源工作 susi[39]、pscount[40]和 Android developer 官网中定义的 action[37], 建立特征的集合, 对集合中的敏感特征在实验数据集上的频率进行统计分析。接着, 定义上述所有特征的集合为 S_0, $s_i \in S_0$ 为 S_0 中任意的敏感 API、action 或者敏感字符串特征。定义 s_i 在恶意应用样本中的频率为 $f_m(s_i)=\frac{s_i\text{在恶意应用中的数量}}{\text{恶意应用的数量}}$, 在良性应用样本中的频率为 $f_b(s_i)=\frac{s_i\text{在良性应用中的数量}}{\text{良性应用的数量}}$, 在所有样本中的频率为 $f(s_i)=\frac{s_i\text{在所有应用中的数量}}{\text{所有应用的数量}}$。为了降低样本特征向量的维度, 本文从 S_0 选择可用于有效检测的敏感特征, 基于如下条件对特征进行了筛选:

$$S=\left\{s_i \in S_0, \frac{f_m(s_i)+0.001}{f_b(s_i)+0.001}>0.5, f(s_i)>0.001\right\}$$

得到敏感特征集合 S 如表 1 所示。第一栏表示敏感特征的类型, 第 2 栏表示相应的数量, 第 3 栏为该类型敏感特征的示例。最后, 将 S 作为域关联关系图的节点特征字典。

4.1.2 敏感特征的赋值

本文在该部分介绍敏感特征的赋值, 首先将敏感特征字典 S 映射到空间 D 中, 则:

$$D:=S$$

基于 D, 定义 φ, 用来映射样本集 X 与敏感特征向量的关系, 每一个敏感特征对应特征向量的维度的值区间为 $\{0,1\}$。$\varphi(x)$ 表示某一样本 x 的敏感特征向量表示, 则:

$$\varphi: X \to \{0,1\}^{|S|},\ \varphi(x) \leftrightarrow \left(I(x,s)\right)_{s\in S}$$

其中, I 为赋值函数, 当样本 x 包含某一特征 s 时, 其值为 1, 否则为 0。则 I 的定义为:

$$I(x,s)=\begin{cases}1, s\in x\\ 0, s\notin x\end{cases}$$

此外, 考虑到不同的特征对恶意应用检测分类的敏感度不同, 本文基于 TF-IDF 算法[41]引入权重 ω_s, 则样本 x 的特征向量为:

$$\varphi(x)\leftrightarrow\left(I(x,s)*\omega_s\right)_{s\in S}$$

表 1 敏感特征集合表

Table 1 Sensitive feature set

	数量	举例
敏感 API	648	- android/location/LocationManager;->* - android/net/wifi/WifiManager;->*
敏感 action	147	- android.intent.action.MEDIA_EJECT -- android.intent.action.AIRPLANE_MO
敏感字符串	8	- /system/bin/sh - mount -o remount

4.2 域关联关系图的构建

首先, 定义域表示代码中的包空间或类空间。定义域最小关联关系(Domain minimum correlation relationship, DR), 表示任意 2 个或多个敏感特征在同一特定域中, 该域为所述特征的最小域。如图 2, 敏感特征 API0 和敏感特征 API2 同在域 com 和域 com/b 集合中, com/b 为所有域中的最小域, 所以敏感特征 API0 和 API2 在域 com/b 中具有关系 DR。定义任意两个敏感特征 s_i, s_j 的具体关系:

$$\left(s_i,s_j\right)^{DR}\in\{dr_0,dr_1,...,dr_l\}$$

其中, dr_0 表示敏感特征 s_i, s_j 同属于一个类文件; dr_1 表示 s_i, s_j 分别所属的类文件与其最小域相对深度最大值为 1, 且不属于同一个类文件; 以此类推, dr_l 表示敏感特征 s_i, s_j 所属的类文件与其最小域相对深度最大值为 l。l 为 DR 阈值, 可以控制生成域关联关系图边的稠密程度。最后, 敏感特征域关联关系图 SCG 可表示为一种无向加权图 $SCG=(V,E)$。

- $V=\{v_i\,|\,1\leqslant i\leqslant n\}$ 定义为敏感特征节点的集合, 其中, $v_i\in V$ 表示敏感特征节点。
- $E=\{(v,v')\,|\,v\in V,v'\in V\}$ 定义为敏感特征节点边的集合, 其中, $(v,v')\in E$ 表示两个节点 v 和 v' 具有某种 DR 关系。

图 2 表示敏感特征之间的域关联关系示意图, 图中文件夹表示不同的包域, 文件表示不同包域中的类域, 绿色节点表示从类中提取的敏感特征。由图可知, 敏感特征 API1 和 API2 同属于一个类域; 敏感特征 API0 和 API2 的最小域为 com/b, 与敏感特征 API0 和 API2 分别的相对深度最大值为 2, 则 $(API1,API2)^{DR}=dr_0$, $(API0,API2)^{DR}=dr_2$。在图 2 中, 设置 DR 关系阈值 $l=2$, 则 SCG 如图中实线部分所示。其中, 敏感特征之间的边用不同级别的 dr 来表示。同时, 我们将敏感特征自身加上环, 防止后续卷积的过程中丢失孤立点的节点信息。

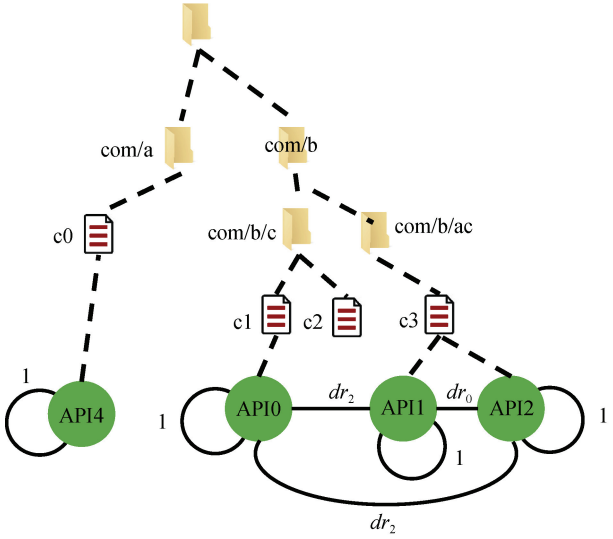


图 2 敏感特征域关联示意图

Figure 2 The sensitive feature domains' correlation

此外, 直接使用 S 中的特征作为 SCG 图的节点易导致不同域中的特征指向同一个高频特征节点, 无法准确刻画特征的功能域。因而, 本文提出特征子图的概念, 用以表示在代码里, 任意方法中敏感特征的集合 v^g, 且对于任意 $s_i\in v^g$, $s_i\in S$。在这里, 我们整理出所有样本中的 v^g 的集合, 并将其作为 SCG 的节点特征。同时, 使用敏感特征的 TF-IDF 值为 v^g 赋值, 其值为: $Value(v^g)=\frac{1}{m}\sum_i^m Value(v_i)$, m 为 v^g 中特征数量。

为了准确刻画应用的行为模式, 本文构建 SCG 来表征应用。进一步, 提出算法 1, 描述如何表示 SCG 的节点和边的特征。同时, 基于工具 DGL[42]生成 SCG 的对象。如算法 1 所示: 算法的输入为样本 x; 参数为构建 SCG 的 DR 阈值 l, 及相应边权重的集合 W (本文将在 GCNDroid 检测模型部分用邻接矩阵作详细介绍); 最终返回 SCG 的对象 g。首先, 用样本的敏感特征集合初始化图 g, 作为图 g 的节点, 并为节点特征赋其 TF-IDF 值(第 1～2 行)。根据样本 x 代码的目录构建敏感特征树, 树中的叶节点为敏感特征(第 4 行)。类比于 $(s_i,s_j)^{DR}=dr_{(l_0-l)}$ 的逆过程, 获取域深度为 l_0-l 域的集合 pac_set。本文的域深度是指某域存在包含敏感特征的类文件, 且该域与类文件的相对深度(第 9 行)。对于任意域深度为 l_0-l 的域,

获取该域子目录下, 域深度小于 l_0-l 的子域的集合 subpac_set。在这里, 集合中的子域都存在包含敏感特征的类文件, 且类文件与该域的相对深度小于 l_0-l (第 10～11 行)。然后, 获取每一个子域中与该子域相对深度小于 l_0-l 的所有类文件的敏感特征列表集合 *nodes_set*, 在子域的敏感特征列表之间添加 $dr_{(l_0-l)}$ 类型的边(第 12～13 行)。l 减 1, 提高 dr 的级别, 重复上述过程(第 15 行)。达到预定条件结束循环, 返回 g (第 16～17 行)。

算法1. SCG生成算法

输入: 样本 x

参数: DR阈值 l, 连接权重集合 W

输出: SCG对象 g

```
1   g = SCG()
2   g.addNodes(feature_num)
3   l0 = l
4   pac_tree = InitTree(x)
5   WHILE l > 0 DO
6       IF depth < 1 THEN
7           BREAK
8       END IF
9       pac_set = arcH((s_i, s_j)^DR = dr_(l0-l))
10      FOR pac in pac_set DO
11          subpac_set = getSubPac(pac)
12          nodes_set = getNLSFromTree(pac)
13          g.drawing(nodes_set, w_(l0-l))
14      END FOR
15      l = l - 1
16 END WHILE
17 RETURN g
```

4.3 GCNDroid 检测模型

4.3.1 GCN 的介绍

DNN, CNN 等深度学习模型有效地捕获了欧几里得数据的隐藏模式, 但越来越多的应用程序以图形形式表示数据[43]。受 CNN 的启发, GCN 相关的概念和定义发展迅速。类比于图像卷积, GCN 主要通过图形卷积的方式获取邻居节点的信息。例如, 通过获取节点邻域信息的加权平均来对图形进行卷积, 实现多个节点的有效关联。GCN 相关概念可表示为: 首先, 用 $G=(V,E)$ 表示图形。其中 V 表示节点的集合, E 表示边的集合。$v_i \in V$ 表示一个节点, $e_{ij}=(v_i,v_j)\in E$ 表示节点 v_i 和 v_j 之间的边。节点 v 的邻居节点为 $N(v)=\{u\in V|(v,u)\in E\}$。邻接矩阵 $\boldsymbol{A}$ 用 $n\times n$ 的矩阵表示, 如果存在 $e_{ij}\in E$, 则 $A_{ij}=1$, 否则 $A_{ij}=0$。节点属性用 f 表示, $\boldsymbol{f}\in R^{n\times d}$ 表示节点的特征矩阵, $f_v\in R^d$ 表示节点 v 的特征向量。同时, 边的特征用 $\boldsymbol{f}^e$ 表示, $\boldsymbol{f}^e\in R^{m\times c}$ 表示边的特征矩阵, $\boldsymbol{f}^e_{v,u}\in R^c$ 表示边 $e_{v,u}$ 的特征向量。类比于 CNN, 本文用公式(1), (2)表示GCN模型, 公式(1)为GCN的矩阵表示, 公式(2)为向量表示。其中, $\boldsymbol{H}^{(l)}$ 是第 l 层的激活矩阵, $h_i^{(l)}$ 为节点 v_i 第 l 层的表示, $H^{(0)}=x$。$N(i)$ 为节点 v_i 邻居节点的集合, c_{ij} 为归一化因子, 可用节点 v_i 和 v_j 入度的平方根的乘积表示。σ 为激活函数, 如 $ReLU(.)=\max(0,.)$。$W^{(l)}$ 表示连接权重。

$$H^{(l+1)}=f\left(\boldsymbol{H}^{(l)}+\boldsymbol{A}\right) \tag{1}$$

$$h_i^{(l+1)}=\sigma\left(\sum_{j\in N(i)}\frac{1}{c_{ij}}h_j^{(l)}W^{(l)}\right) \tag{2}$$

4.3.2 基于 GCN 的检测模型

本文使用 $n\times 1$ 的矩阵来表示样本 x, 其中 n 为样本中敏感特征节点的个数。通常情况下, 邻接矩阵 $\boldsymbol{A}$ 表示节点之间的连接状态, 即节点 v_i 和 v_j 存在连接边。此时, $A_{ij}=1$, 否则 $A_{ij}=0$。与社交网络等领域不同, 本文图中的节点基于不同的域进行连接, 用邻接矩阵 $\boldsymbol{A}$ 直接计算, 忽略了节点之间域的相对范围。为此, 我们提出邻接矩阵 $\hat{\boldsymbol{A}}$, $\hat{A}_{ij}=w_A\hat{\boldsymbol{A}}(dr_l)=w_l(1-\log(l))$, $l>0$, w_A 为连接参数。敏感特征域关联关系图的卷积过程数据流如图 3 所示。上面的矩阵为样本 x 的值矩阵, 即多维敏感特征的特征值; 下面的矩阵表示样本 x 基于 SCG 的邻接矩阵 $\hat{\boldsymbol{A}}$。对两个矩阵进行矩阵乘, 实现特征节点的卷积。

接下来, 设计基于 GCN 域关联的检测模型 GCNDroid。检测模型的 3 层网络结构与信息流如公式(3)所示。本文基于公式(3)中的模型进行训练, 其中 $\boldsymbol{W}^0$ 为输入层到隐藏层之间的权重矩阵, $\boldsymbol{W}^1$ 为隐藏层之间的权重矩阵, $\boldsymbol{W}^2$ 为隐藏层到输出层之间的权重矩阵。然后, 选取 softmax 作为激励函数, 用来生成样本 x 的标签类别, 如公式(4)所示。然后, 使用公式(5)中的方法来评估所有标签的交叉熵误差, 其中, y_L 为样本的类别标签。最后, 使用梯度下降算法

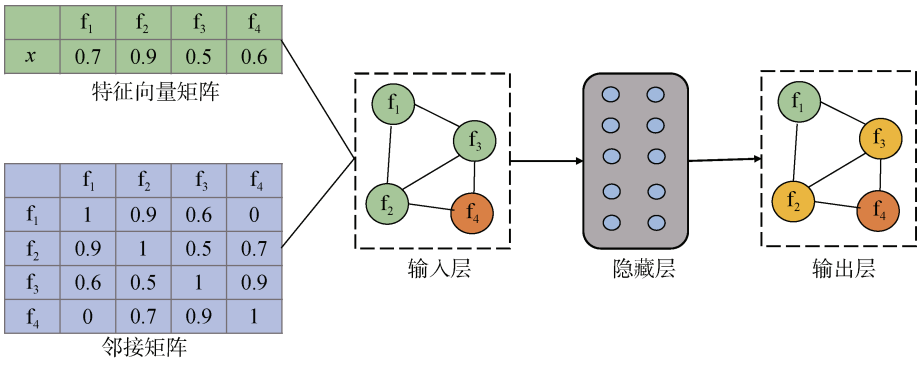


图 3　卷积过程数据流示意图

Figure 3　The information flow of graph convolutional

更新网络中的权重。

$$Z = f\left(x, \hat{\boldsymbol{A}}\right) = softmax \left(b + \left(\hat{\boldsymbol{A}}\, ReLU\left(\hat{\boldsymbol{A}} x \boldsymbol{W}^{0}\right)\boldsymbol{W}^{1}\right)\boldsymbol{W}^{2}\right) \tag{3}$$

$$softmax(x) = \frac{1}{\sum_{t} exp(x)} exp(x) \tag{4}$$

$$\Gamma = -\sum_{l \in y_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf} \tag{5}$$

基于 GCNDroid 的检测模型的构建过程如算法 2 所示。该算法的输入为样本集 X；参数为迭代次数 epoch 和学习率 lr；最终返回训练好的 GCNDroid 模型 model。首先，按照公式(3)，(4)构建分类器 Classifier 并初始化模型 model(第 1 行)。选择损失函数初始化变量 loss_func，同时，选择优化器初始化变量 opt(第 2～3 行)。生成样本集 X 中 x 对应的 SCG 对象 g 并获得相应的样本向量 $\boldsymbol{\varphi}(x)$ 和邻接矩阵 $\hat{\boldsymbol{A}}_{ij}$ (第 6～7 行)。计算 x 的概率值和对应的损失 *loss*(第 8～9 行)。基于 loss，利用梯度下降方法更新模型参数(第 10～12 行)。重复所有训练样本，直到模型收敛或者达到迭代次数的上限 epoch，最终返回 model(第 15 行)。

算法2. 检测模型实现流程

输入: 样本集 X
参数: 迭代次数epoch，学习率lr
输出: 检测模型model

```
1  model = Classifier()
2  loss_func = nn.CrossEntropyLoss()
3  opt = optim.Adam (model.parameters, lr)
4  FOR epo in range (epoch) DO
5      FOR ( x , label) in  X  DO
6          g  =  算法1( x )
7          φ(x),  Â_ij  =  g .getVector()
8          pre =  模型model( φ(x) , Â_ij )
9          loss = loss_func(pre, label)
10         opt.zero_grad()
11         loss.backward()
12         opt.step()
13     END FOR
14 END FOR
15 RETURN model
```

5　实验数据与结果分析

在该部分，本文进行了多组实验进行分析、评估 GCNDroid 的先进性，并按照如下两个方面展开介绍：相关配置，实验结果分析。

5.1　相关配置

5.1.1　数据集

本文用爬虫等工具获取的 20713 个样本，通过 VirusTotal[44]等扫描判定为良性的样本作为实验数据集所用的良性样本；同时，与国内安全厂商合作提供的 5109 个恶意应用样本与 Drebin[9]数据集混合作为实验数据集所用的恶意应用样本，共 10669 个样本。为了规避应用大小对检测分类的影响，本文选择样本大小在 100K 和 500M 之间的样本进行实验，其中良性样本的平均大小为 35.1M，恶意应用样本的平均大小为 29.8M，进行实验的良性样本为 7175 个，恶意应用样本为 7180 个。最后，采用 10 折交叉检验测试方法评估检测的效果。

5.1.2　实验配置

本文使用 Python 实现了基于 GCNDroid 的 Android 恶意应用检测系统。其中，样本的解压缩与反编译基于开源工具 Androguard[38]，敏感特征域关联关系图 SCG 的构建基于开源工具 DGL[42]，SVM 等

机器学习算法的实现基于开源工具 Sklearn[45], GCN、DNN 等深度学习模型的实现基于开源框架 Pytorch[46]。实验的硬件环境: 内存为 32G, 处理器为 Intel(R) Core(TM) i7-7820X CPU @ 3.6GHz 的服务器; 软件环境为 Win10, Python3.6。

5.1.3 评价指标

本文将恶意应用样本定义为正元组, 良性应用样本为负元组。TP 表示分类器将恶意应用样本正确标识为恶意的元组; TN 表示分类器将良性应用样本正确标识为良性的元组; FP 表示分类器将恶意应用样本错误标识为良性的元组; FN 表示分类器将良性应用样本错误标识为恶意的元组。除此之外, M 表示测试样本中恶意应用样本的数量, N 表示测试样本中良性应用样本的数量。测试样本中, 我们按照被判定为恶意的样本的概率进行排序, 序号用 Rank 表示。以恶意应用样本的精确率(precision)、召回率(recall)和调和平均数(f1-score)为例, 本文评价指标的计算公式如下:

$$precision = \frac{TP}{TP + FN}$$

$$recall = \frac{TP}{TP + FN}$$

$$f1 - score = \frac{2 \times recall \times precision}{recall + precision}$$

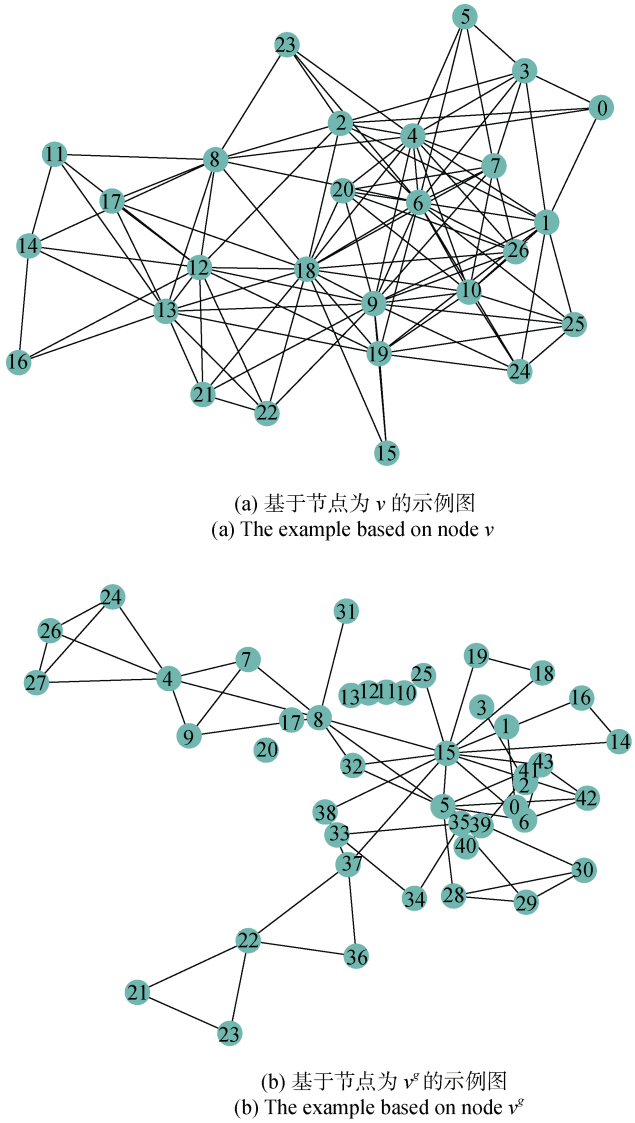
$$auc = \frac{\sum_{i \in malware} rank_i - \frac{M \times (M+1)}{2}}{M \times N}$$

5.2 实验结果分析

5.2.1 SCG 示例分析

本小结分别以敏感特征 v 为节点特征和以敏感子图 v^g 为节点特征对 SCG 进行可视化, 分析两类节点特征对 SCG 的影响。首先, 设置 DR 阈值 $l=1$,随机选择 MD5 为 040809d3f9401854efc96a71873e2f70 样本, 可视化其域关联关系。其 SCG 的示意图如图 4 所示, 其中, 图 4(a)表示基于节点特征为 v 的 SCG, 图 4(b)基于节点特征为 v^g 的 SCG。

由图 4 可知, 同一样本中, 基于节点特征为 v 的 SCG 图比基于节点特征为 v^g 的节点数更少, 主要原因是 v^g 为样本中节点 v 的组合, 一定程度能够表示出特定的敏感特征在多个域中的分布。另一方面, 以边作为比较, 基于节点特征为 v^g 的 SCG 更加稀疏, 多个节点之间的关系亦更加清晰; 中心节点较少, 在卷积的过程中, 特定域内的节点不易受其他域相同节点计算带来的影响, 更适合构建 GCNDroid 的检测模型。



(a) 基于节点为 v 的示例图
(a) The example based on node v

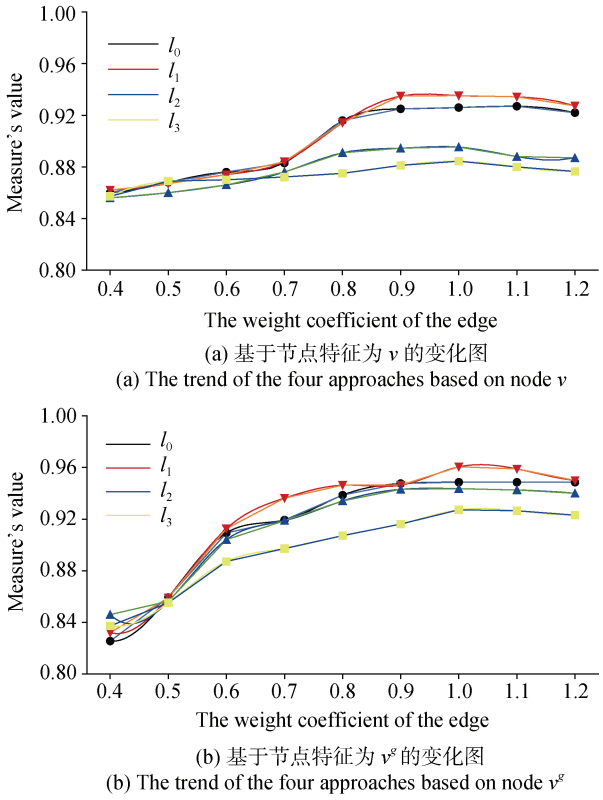
(b) 基于节点为 v^g 的示例图
(b) The example based on node v^g

图 4 SCG 示例图
Figure 4 SCG example

5.2.2 GCNDroid 系统参数

为了分析 DR 阈值 l, 邻接矩阵权重 w_A 和基于敏感特征节点特征 v 和基于敏感子图节点特征 v^g 对检测效果的影响。本文以 AUC 作为调参指标, 选定 DR 阈值 $l \in \{0,1,2,3\}$, $w_A \in \{0.4,0.5,0.6,0.7,0.8,0.9,1.0,1.1,1.2\}$。$l_0$、$l_1$、$l_2$、$l_3$ 分别表示 DR 阈值 $l=\{0,1,2,3\}$ 的方法。在这里, 我们用方法 l_0、l_1、l_2、l_3 随 w_A 变化的指标来评估 l, w_A 对检测的影响, 其效果如图 5 所示。其中图 5(a), 图 5(b)分别表示四种不同 l 的方法基于节点特征为 v 和 v^g 的变化图。

结合图 5(a), 图 5(b)可知, 随着 w_A 的值不断变大, 方法 l_0、l_1、l_2、l_3 相应指标值变大, $w_A=0.8$ 左右时, 两图中的大部分方法的指标趋于稳定, 并在



(a) 基于节点特征为 v 的变化图
(a) The trend of the four approaches based on node v

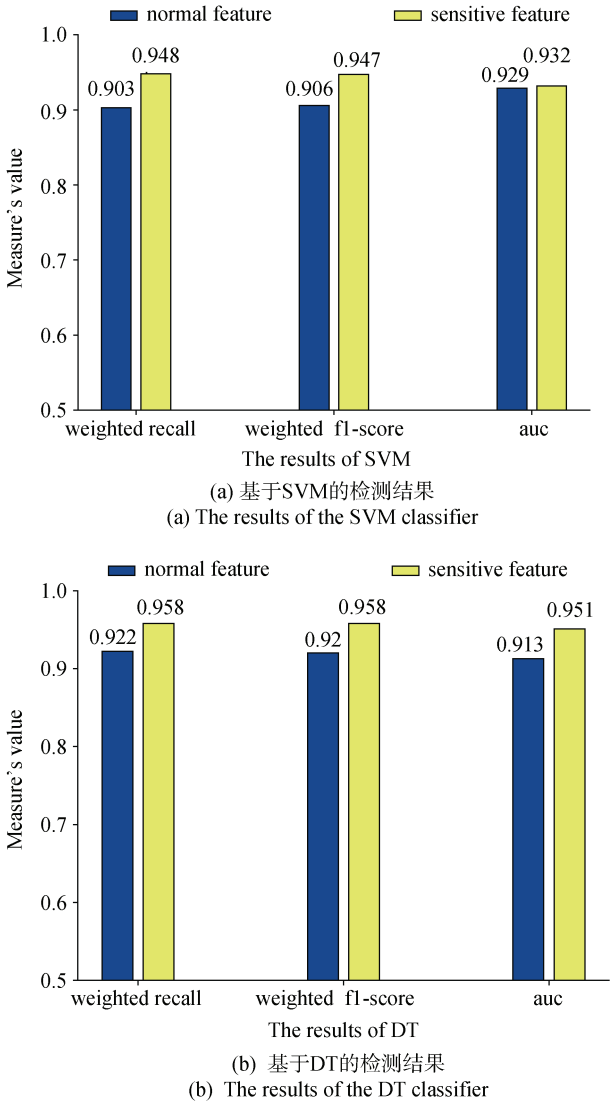
(b) 基于节点特征为 v^g 的变化图
(b) The trend of the four approaches based on node v^g

图 5　四种不同 l 的方法随 w_A 变化图
Figure 5　The trend of the four approaches with w_A

$w_A=1.0$ 时，指标值达到最高，并随着 w_A 数值增大，有下降趋势。两张图中，方法 l_1 整体表现均为最好。从数值上来看，方法 l_1 在图 5(b)中比在图 5(a)变化幅度较大，在 $w_A=1.0$ 时，方法 l_1 在图 5(b)指标最高，即基于敏感子图 v^g 作为 SCG 的节点特征具有最好的分类能力。综上，本文选择 DR 阈值 $l=1$，邻接矩阵权重 $w_A=1.0$，基于敏感子图 v^g 的节点特征实现 GCNDroid 检测系统。

5.2.3　实验对比

本文分析了特征集合 S_0 与敏感特征字典 S 用于分类的效果。使用 SVM，决策树(DT)两种传统的机器学习算法作为分类算法，基于 S_0 和 S 构建了四种检测模型，通过比较权重召回率(weighted recall)、权重调和平均数(weighted f1-score)和 AUC 值来评估上述两类特征。其中，权重召回率和权重调和平均数为恶意和良性应用单个类别对应的召回率和调和平均数的加权平均值。四种检测模型比较的柱状图如图 6 所示。其中图 6(a), (b)分别表示基于 SVM、DT 算法的检测模型。需要说明的是，本文将基于特征 S_0 的方法用正常特征表示(即图中的 normal feature)；基于敏感特征字典 S 的方法用敏感特征表示(即图中的 sensitive feature)。



(a) 基于SVM的检测结果
(a) The results of the SVM classifier

(b) 基于DT的检测结果
(b) The results of the DT classifier

图 6　基于两类特征的检测模型比较
Figure 6　The comparison of the detection models based on two types of features

结合图 6(a)，图 6(b)可知，基于敏感特征的检测方法相较于基于正常特征的方法，在两个通用的检测算法上，其权重召回率、权重调和平均数、AUC 三项指标均有较大提高，且具有较高的指标值。此外，DT 相较于 SVM 在两类特征上三项指标值更高，具有更好的分类效果。综上，筛选之后的敏感特征字典用于分类的效果更好，所以，本文基于敏感特征字典 S 进行后续的检测、分析任务。

为了验证 GCNDroid 方法的先进性，本文选取 Android 恶意应用检测领域中经常使用的机器学习算法 SVM、DT 和深度学习算法 DNN、CNN 进行相关对比实验。其中，对比 SVM 模型基于线性核函数，惩罚系数为 0.025，其他系数采用的是算法接口中默认

的参数[16]; 对比 DT 模型采用的是基尼系数的标准, 设置在所有的选择中选择最好的切分点, 其他系数采用的是算法接口中默认的参数; 类比于 GCNDroid, 构建与 GCNDroid 模型层数接近的对比 DNN 模型和对比 CNN 模型。其中对比 DNN 包括三层 Linear 层, Linear 层与 Linear 层之间加入 Dropout, 元素归零的概率系数为 0.5; 对比 CNN 模型包括两层卷积操作和一个 Linear 层, 卷积操作中池化层核大小为 2。同时, 对比 DNN、对比 CNN 模型采用与 GCNDroid 相同的优化器, 损失函数和学习率, 进行 50 次迭代训练。对比的实验结果如表 2 所示, 其中第 1 栏用于区别不同对比模型基于的敏感特征, 第 2 栏为检测模型基于的算法类别, 第 3～6 栏为恶意、良性应用单个类别对应的召回率和调和平均数值, 第 7、8、9 行分别为分类的权重召回率、权重调和平均数、AUC 值。

由表 2 可知, GCNDroid 在恶意应用和良性应用的上分别的召回率和调和平均数相对较高, 相应指标均在 0.9350 以上; 在良性应用上的两项指标达到了 0.9790 以上, 表明 GCNDroid 能够准确区分恶意应用和良性能用。整体来看, GCNDroid 分类结果的权重召回率、权重调和平均数和 AUC 值达到很高的水平, 分别为 0.9605、0.9642、0.9605, 表明 GCNDroid 具有很高的检测能力。相较于上述 8 种方法, 在恶意应用, 良性应用的召回率、调和平均数以及权重召回率、权重调和平均数、AUC 等指标上均为最高, 且各项指标相对于其他最优的方法均有较大提高。表明 GCNDroid 更加适合恶意应用的检测。此外, 比对发现, 基于敏感子图 v^g 为节点的检测方法相较于基于敏感特征 v 的方法各项指标无明显提升, 在部分模型上甚至有所下降。进一步分析发现, 敏感子图 v^g 虽为多个敏感特征的集合, 在参与计算时没法充分体现子图中单个敏感特征信息, 故而影响后续的检测能力, 本文将这个发现放到未来的研究计划中。

表 2 相关检测方法比较表

Table 2 The comparison of related detection approaches

	检测模型	恶意应用		良性应用		weighted recall	weighted f1-score	AUC
		recall	f1-score	recall	f1-score			
基于敏感特征 v	SVM	0.8956	0.9095	0.9691	0.9630	0.9324	0.9362	0.9323
	DT	0.9357	0.9300	0.9681	0.9705	0.9519	0.9503	0.9518
	DNN	0.9002	0.9087	0.9662	0.9624	0.9332	0.9356	0.9331
	CNN	0.9362	0.9412	0.9778	0.9756	0.9570	0.9584	0.9569
基于敏感子图 v^g	SVM	0.7837	0.8616	0.9852	0.9494	0.8844	0.9055	0.8844
	DT	0.9169	0.9207	0.9687	0.9671	0.9428	0.9439	0.9428
	DNN	0.9196	0.9386	0.9833	0.9750	0.9514	0.9568	0.9514
	CNN	0.9287	0.9358	0.9765	0.9734	0.9526	0.9546	0.9526
	GCNDroid	**0.9363**	**0.9493**	**0.9848**	**0.9792**	**0.9605**	**0.9642**	**0.9605**

5.2.4 性能分析

GCNDroid 在系统实现时并没有生成样本的 SCG。与直接提取敏感特征, 基于机器学习算法的检测模型类似, GCNDroid 整合了敏感特征集合、敏感特征树、域关联关系图的构建等过程, 最终只生成了敏感特征向量及对应的权重邻接矩阵。相对于直接从 smali 代码中提取敏感特征, GCNDroid 多了构建权重邻接矩阵查找树的过程。本文统计了所有样本通过 GCNDroid 特征工程的过程, 平均每个样本耗时 5.6s, 而直接提取敏感特征向量的平均每个样本的耗时为 5.1s, 差距很小。此外, 生成的邻接矩阵可以作为恶意应用的有效签名, 有利于实现快速高效的签名比对。此外, 检测模型的效率为各个开源算法的差异, 本文将不做过多介绍。

5.2.5 讨论

SCG 是基于 smali 代码进行构建的, 具有一定的可扩展性。如对于加壳、反射等代码可通过动态执行、脱壳技术或者 DroidRA[47]等技术获取.dex 文件、smali 代码片段, 构建相应的 SCG 子图, 可直接并入 SCG 中, 用于进一步分析。同时, 构造的 SCG 关联了恶意负载的多个敏感行为, 使得构造有效的人工智能对抗样本更加困难, 一定程度规避了对抗样本对检测的影响, 检测模型具有检测一般对抗样本的能力。

受到编程语言的限制, 对敏感特征进行域关联的方法仅仅适用于 Android 等具有包特性平台的恶意应用检测。同时, GCNDroid 无法分析 Android 应用中原生代码(native code)。我们将在未来的工作中

进一步的研究, 结合 smali 代码和原生代码, 对 Android 恶意应用进行完整的分析。

6 相关工作

该部分将从特征工程和检测模型两个方面展开介绍。

6.1 特征工程

Android 恶意应用检测方法从特征工程角度划分, 主要包括: 基于权限[3-7]、敏感 API 等统计特征[8-10], 基于操作码和 API 的序列特征[11-12], 基于方法或 API 之间的调用关系[16-18, 45-46]等特征的检测方法。与 GCNDroid 较类似的方法为基于方法和 API 之间的调用关系的方法: Zhang M 等人[48]提出 DroidSIFT 工具, 用来对抗代码转换攻击, 实现恶意代码检测和家族分类。他们构建了上下文加权的 API 依赖图作为特征集, 提出一种新的图相似性度量方法, 以准确匹配应用行为, 同时容忍轻微的实施差异。Sun 等人[17]揭示了恶意应用家族在运行时的相似性。基于静态结构信息和动态行为检测恶意应用。其中, 静态信息包括在 AndroidManifest 文件中的组件信息以及其静态的逻辑关系, 在程序加载前生成; 动态行为签名, 包括运行时候的敏感 API 的调用和系统可疑进程的调用。使用编辑距离来计算待检测应用与各家族样本的相似度。Fan M 等人[18]提出 FalDroid, 实现 Android 恶意家族分类。FalDroid 构造了一种新的基于图的特征来表征应用程序。通过聚类和子图的相似性分析, 提出了一种敏感 API 子图权重自适应的匹配算法来强化特征。由此可见, 上述特征缺乏对敏感特征的域关联分析, 且节点除了敏感特征, 仍有大量的类名、方法名, 需要映射更大的空间图, 更加复杂。

6.2 检测模型

SVM, DT 等[3-4,8-9,16,18,46]传统机器学习, DNN、CNN[17-19]等深度学习算法被广泛用于构建 Android 恶意应用检测模型。其中, GCNDroid 与基于 CNN 的检测模型最类似: Huang 等人[21]在不提取预选特征的前提下减少特征工程的人力, 基于 CNN 实现了 Android 恶意应用检测系统 AndroiD。该系统将 classes.dex 的字节码从 Android 存档文件转换为 rgb 颜色代码, 并将其存储为具有固定大小的彩色图像, 然后将彩色图像输入到卷积神经网络 CNN 中以进行自动特征提取和训练。Wang 等人[22]提出基于自编码器 DAE 和卷积神经网络 CNN 的深度检测模型。利用 DAE 重构高纬度特征, 基于多层 CNN 检测恶意应用。Cui 等人[23]提出了一种利用 CNN 和智能算法来推进恶意代码检测的方法。其中, CNN 用于识别和分类从恶意代码的可执行文件转换而来的灰度图像。然后, 采用非主导排序遗传算法 II(NSGA-II)来处理恶意软件家族的数据不平衡问题。综上, CNN 将敏感特征节点看作图像的像素, 卷积中的每个点的邻居是固定且有序的, 并不具有实际的连接意义。而 GCN 可以通过获取节点邻域信息的加权平均值来执行图卷积, 邻居节点和连接权重都具有现实意义。

7 总结

本文提出了一种基于敏感元素域关联的特征 SCG, 用来准确表征特定功能模块(包括恶意负载)中敏感行为, 敏感行为之间的完整关系, 作为后续检测分析的基础; 提出了一种基于敏感域关联分析的检测模型 GCNDroid, 有效结合了 SCG 中敏感特征与敏感特征之间的结构信息, 弥补了当前检测方法中的不足; 实现了基于 GCNDroid 的 Android 恶意应用检测系统, 实验表明, 该系统可以有效提高模型的检测能力。本文从样本的行为画像展开的分析, 对家族分析等也具有重要的启发和指导意义。此外, 本文也存在一定的不足, 如无法有效表示敏感子图中敏感特征的信息和原生代码问题, 我们将在未来的工作中完善此类不足。

参考文献

[1] Smartphone Challenges Continue in 2019. IDC. https://www.idc.com/getdoc.jsp?containerId=prUS45487719. Sept. 2019.

[2] 2019 年 Android 恶意软件专题报告. 360 烽火实验室. https://blogs.360.cn/post/review_android_malware_of_2019.html. Febr. 2020.

[3] Wang Y, Zheng J, Sun C, et al. Quantitative Security Risk Assessment of Android Permissions and Applications[C]. *The 27th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy XXVII - Volume 7964*, 2013: 226-241.

[4] Merlo A, Georgiu G C. RiskInDroid: Machine Learning-Based Risk Analysis on Android[C]. *IFIP International Conference on ICT Systems Security and Privacy Protectio*, 2017:538-552.

[5] Sanz B, Borja, Laorden C, et al. Puma: Permission usage to detect malware in android[C]. *International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions*, 2013.

[6] Sanz B, Santos I, Laorden C, et al. MAMA: Manifest Analysis for Malware Detection in Android[J]. *Cybernetics and Systems*, 2013, 44(6/7): 469-488.

[7] Wang W, Wang X, Feng D W, 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.

[8] Karbab E B, Debbabi M, Derhab A, et al. Android Malware Detec-

tion Using Deep Learning on API Method Sequences[EB/OL]. 2017: arXiv: 1712.08996. https://arxiv.org/abs/1712.08996.

[9] Arp D, Spreitzenbarth M, Hübner M, et al. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket[C]. *Network and Distributed System Security Symposium*. 2014.

[10] Jiang J G, Li S, Yu M, et al. MRDroid: A multi-act classification model for android malware risk assessment[C]. *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems*, 2018: 64-72.

[11] Suarez-Tangil G, Tapiador J E, Peris-Lopez P, et al. D Endroid: A Text Mining Approach to Analyzing and Classifying Code Structures in Android Malware Families[J]. *Expert Systems With Applications*, 2014, 41(4): 1104-1117.

[12] Li Y P, Jang J, Hu X, et al. Android Malware Clustering through Malicious Payload Mining[M]. Research in Attacks, Intrusions, and Defenses. Cham: Springer International Publishing, 2017: 192-214.

[13] Ali-Gombe A, Ahmed I, Richard G G, et al. OpSeq: Android Malware Fingerprinting[C]. *The 5th Program Protection and Reverse Engineering Workshop*, 2015: 1-12.

[14] Hanna S, Huang L, Wu E, et al. Juxtapp: A Scalable System for Detecting Code Reuse among Android Applications[C]. *The 9th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2012: 62-81.

[15] Li Y P, Jang J, Hu X, et al. Android Malware Clustering through Malicious Payload Mining[EB/OL]. 2017: arXiv: 1707.04795. https://arxiv.org/abs/1707.04795.

[16] Fan M, Liu J, Luo X P, et al. Frequent subgraph based familial classification of android malware[C]. *2016 IEEE 27th International Symposium on Software Reliability Engineering*, 2016: 24-35.

[17] Sun M S, Li X L, Lui J C S, et al. Monet: A User-Oriented Behavior-Based Malware Variants Detection System for Android[J]. *IEEE Transactions on Information Forensics and Security*, 2017, 12(5): 1103-1112.

[18] Fan M, Liu J, Luo X P, et al. Android Malware Familial Classification and Representative Sample Selection via Frequent Subgraph Analysis[J]. *IEEE Transactions on Information Forensics and Security*, 2018, 13(8): 1890-1905.

[19] Zhang J M, Zou F T, Zhu J R, et al. Android malware detection based on deep learning[C]. *2018 IEEE 4th International Conference on Computer and Communications*, 2019: 2190-2194.

[20] Xu L,Zhang D, Jayasena N, et al. Hadm: Hybrid analysis for detection of malware[C]. *Proceedings of SAI Intelligent Systems Conference*. Springer, Cham, 2016.

[21] Huang T H D, Kao H Y, Communication N A B T, et al. R2-D2: ColoR-inspired convolutional NeuRal network (CNN)-based AndroiD malware detections[C]. *2018 IEEE International Conference on Big Data*, 2019: 2633-2642.

[22] Wang W, Zhao M X, Wang J G. Effective Android Malware Detection with a Hybrid Model Based on Deep Autoencoder and Convolutional Neural Network[J]. *Journal of Ambient Intelligence and Humanized Computing*, 2019, 10(8): 3035-3043.

[23] Cui Z, Du L, Wang P, et al. Malicious code detection based on CNNs and multi-objective algorithm[J]. *Journal of Parallel and Distributed Computing*, 2019, 129: 50-58.

[24] Li M H, Wang P, Wang W, et al. Large-scale third-party library detection in android markets[C]. *IEEE Transactions on Software Engineering*, 2018: 981-1003.

[25] Chen K, Wang P, Lee Y, et al. Finding Unknown Malice in 10 Seconds: Mass Vetting for New Threats at the Google-Play Scale[C]. *The 24th USENIX Conference on Security Symposium*, 2015: 659-674.

[26] Zhou Y J, Jiang X X, Processing C A, et al. Dissecting android malware: Characterization and evolution[C]. *2012 IEEE Symposium on Security and Privacy*, 2012: 95-109.

[27] Zhou W, Zhou Y J, Jiang X X, et al. Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces[C]. *The second ACM conference on Data and Application Security and Privacy*, 2012: 317-326.

[28] Zhou W, Zhou Y J, Grace M, et al. Fast, Scalable Detection of "Piggybacked" Mobile Applications[C]. *The third ACM conference on Data and application security and privacy*, 2013: 185-196.

[29] Jiang J G, Chen J M, Gu T B, et al. Warder: Online Insider Threat Detection System Using Multi-Feature Modeling and Graph-Based Correlation[C]. *MILCOM 2019 - 2019 IEEE Military Communications Conference*, 2019: 1-6.

[30] Jiang J G, Chen J M, Gu T B, et al. Anomaly Detection with Graph Convolutional Networks for Insider Threat and Fraud Detection[C]. *MILCOM 2019 - 2019 IEEE Military Communications Conference*, 2019: 109-114.

[31] Kipf T N, Welling M. Semi-Supervised Classification with Graph Convolutional Networks[EB/OL]. 2016: arXiv: 1609.02907. https://arxiv.org/abs/1609.02907.

[32] Java package. Wikipedia. https://en.wikipedia.org/wiki/Java_package. June 2020.

[33] Zhu Z Y, Dumitraş T. FeatureSmith: Automatically Engineering Features for Malware Detection by Mining the Security Literature[C]. *The 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016: 767-778.

[34] Singh A K, Jaidhar C D, Kumara M A A. Experimental Analysis of Android Malware Detection Based on Combinations of Permissions and API-Calls[J]. *Journal of Computer Virology and Hacking Techniques*, 2019, 15(3): 209-218.

[35] 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 systems*, 2013: 86-103.

[36] BroadcastReceiver | Android Developers. Google. https://developer.android.com/reference/android/content/BroadcastReceiver/. July 2019.

[37] Intents common. Android developer. https://developer.android.com/guide/components/intents-common. May 2020.

[38] Androguard is a full python tool to play with Android files. Androguard. https://github.com/androguard/androguard/. May 2020.

[39] Rasthofer S, Arzt S, Bodden E. A Machine-Learning Approach for Classifying and Categorizing Android Sources and Sinks[J]. 2014: 14: 1125.

[40] Au K W Y, Zhou Y F, Huang Z, et al. PScout: Analyzing the Android Permission Specification[C]. *The 2012 ACM conference on Computer and communications security*, 2012: 217-228.

[41] Huang C H, Yin J, Hou F. A Text Similarity Measurement Combining Word Semantic Information with TF-IDF Method[J]. *Chinese Journal of Computers*, 2011, 34(5): 856-864.
[42] Overview of DGL. DGL Team. https://docs.dgl.ai/. May 2020.
[43] Wu Z H, Pan S R, Chen F W, et al. A Comprehensive Survey on Graph Neural Networks[J]. *IEEE Transactions on Neural Networks and Learning Systems*, 2021, 32(1): 4-24.
[44] VirusTotal. VirusTotal Community. https://www.virustotal.com/gui/home/upload. Dece. 2019.
[45] Scikit-learn Machine Learning in Python. scikit-learn. https://scikit-learn.org/stable/. May 2020.
[46] Pytorch. Pytorch. https://pytorch.org/. Mar. 2020.
[47] Li L, Bissyandé T F, Octeau D, et al. DroidRA: Taming Reflection to Support Whole-Program Analysis of Android Apps[C]. *The 25th International Symposium on Software Testing and Analysis*, 2016: 318-329.
[48] Zhang M, Duan Y, Yin H, et al. Semantics-Aware Android Malware Classification Using Weighted Contextual API Dependency Graphs[C]. *The 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014: 1105-1116.
[49] Fan M, Luo X P, Liu J, et al. Graph Embedding Based Familial Analysis of Android Malware Using Unsupervised Learning[C]. *The 41st International Conference on Software Engineering*, 2019: 771-782.

姜建国 于 2003 年在四川大学信息安全与应用密码专业获得博士学位, 现任中国科学院信息工程研究所研究员。研究领域为信息安全、保密技术等。Email: jiangjianguo@iie.ac.cn

李松 于 2015 年在河南大学网络工程专业获得学士学位, 现在中国科学院信息工程研究所网络空间安全专业攻读博士学位。研究领域为信息安全、保密技术等。研究兴趣包括: 恶意代码检测、机器学习等。Email: lisong9057@iie.ac.cn

喻民 于 2020 年在中国科学院大学获得博士学位。研究领域为数据安全、保密技术等。研究兴趣包括: 恶意文档检测、文档内容安全、深度伪造检测等。Email: yumin@iie.ac.cn

李罡(Gang Li) 于 2005 年在澳大利亚迪肯大学获得博士学位, 目前担任迪肯大学信息技术学院教授, 研究领域为数据挖掘, 数据隐私保护等。Email: gang.li@deakin.edu.au

刘超 现任中国科学院信息工程研究所正高级工程师。研究领域为移动互联网安全、网络安全测评等。Email: liuchao@iie.ac. cn

李梅梅 于 2007 年在北京大学信息与信号处理专业获得硕士学位, 现任中国科学院信息工程研究所高级工程师。研究领域为信息安全、数据分析等。Email: limeimei@iie.ac.cn

黄伟庆 现任中国科学院信息工程研究所正高级工程师。研究领域为电磁信息安全、信号处理、数据安全等。Email: huangweiqing@iie.ac.cn