# 基于多视图表示学习的安卓恶意应用检测方法

赵文翔, 孟昭逸, 熊 焰, 黄文超

中国科学技术大学 计算机科学与技术学院 合肥 中国 230027

**摘要** 安卓操作系统自发布以来一直保持着很高的市场份额, 并且由于安卓应用的数量庞大、功能繁多、行为语义复杂, 攻击者可采取多种手段将其真实攻击意图隐藏在合法功能之中。然而, 现有检测方案往往只能识别有限类型的恶意应用及行为。为了解决这个问题, 本文利用异构信息网络对现有的代表性检测方案进行高度抽象, 并使用多视图表示学习和多视图融合方法对其进行深度挖掘与协同融合, 以充分释放不同方案的检测潜力, 构建更为精确且全面的恶意应用检测系统。为了实现上述目的, 本文提出并实现了一个基于多视图表示学习的安卓恶意应用检测系统 MVFDroid。该系统首先从敏感数据流、可疑控制条件和权限三个视角出发充分观察安卓应用, 从而构建出异构信息网络, 以描述应用行为的执行逻辑以及行为间的关联关系; 然后采用基于视图的游走方式对异构信息网络进行采样, 以生成不同视图下的应用行为表示向量; 最后利用基于多视图融合的安卓恶意应用检测方法, 将表示向量融合后送入深度神经网络(DNN)分类器中, 从不同视角综合判断其目标应用的恶意性。实验表明, 本文提出的方法可有效检测出安卓恶意应用, 其检测的准确率为 96.57%且 F1 值为 95.56%, 均优于当前的代表性检测方案 Drebin、HinDroid 和 MaMaDroid。同时, 实验结果表明, 本文所使用的基于视图融合的表示学习方法可有效应用于安卓恶意应用检测任务, 其效果优于基准方法 DeepWalk、node2vec 和 metapath2vec。

**关键词** Android 恶意应用检测; 异构信息网络; 多视图融合; 图表示学习

**中图法分类号** TP391 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2024.09.04

# Android Malware Detection Based on Multi-view Representation Learning

ZHAO Wenxiang, MENG Zhaoyi, XIONG Yan, HUANG Wenchao

School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China

**Abstract** The Android operating system has maintained a high market share since its release, and because of the large number of Android applications, their many functions, and their complex behavioral semantics, attackers can adopt a variety of means to hide their true attack intent within legitimate functionality. However, existing detection solutions often identify only limited types of malicious applications and behaviors. To solve this problem, we use heterogeneous information networks to highly abstract existing representative detection schemes, and use multi-view representation learning and multi-view fusion methods to deeply mine and collaboratively fuse them, in order to fully unleash the detection potential of different schemes and build a more accurate and comprehensive malicious application detection system. To achieve the above purpose, this paper proposes and implements a multi-view representation learning based Android malicious application detection system MVFDroid. In this system, we first fully observe Android applications from three perspectives: sensitive data flow, suspicious control conditions and permissions, so as to build a heterogeneous information network to describe the execution logic of application behaviors and the association relationships among behaviors; then we adopt a view-based wandering approach to sample the heterogeneous information network to generate application behavior representation vectors under different views; finally, we use a multi-view fusion-based Android malicious application detection method to fuse the representation vectors and feed them into a deep neural network (DNN) classifier to comprehensively determine the maliciousness of its target applications from different perspectives. Experiments show that the proposed method can effectively detect Android malicious applications with an accuracy of 96.57% and an F1 value of 95.56%, both of which are better than the current representative detection scheme Drebin, HinDroid and MaMaDroid. Meanwhile, experimental results show that the view fusion-based representation learning method used in this paper can be effectively applied to the Android malicious application detection task, which outperforms the benchmark methods DeepWalk, node2vec and metapath2vec.

**Key words** Android malware detection; heterogeneous information network; multi-view fusion; graph representation learning

**通讯作者:** 孟昭逸, 博士, 博士后, Email: mzy516@ustc.edu.cn。

本课题得到国家自然科学基金项目(No. 62102385, No. 62372422, No. 62272434, No. 61972369)、安徽省自然科学基金项目(No. 2108085QF262) 和中央高校基本科研业务费专项资金(No. WK2150110024) 资助。

收稿日期: 2022-08-18; 修改日期: 2023-01-06; 定稿日期: 2024-05-30

## 1 引言

安卓平台自发布以来, 在移动操作系统中一直保持很高的市场份额。据权威数据分析公司 IDC 预测[1], 2022 年安卓操作系统市场份额约为 84.1%, 较 2021 年的 83.8%仍有小幅上涨, 是移动操作系统的绝对霸主。随着安卓系统的蓬勃发展, 安卓端应用也被广泛应用于社交、支付、学习、工作等场景, 渗透进日常生活的方方面面。然而, 安卓应用的安全性问题日益凸显。多方权威安全报告显示, 目前安卓生态系统中存在大量恶意应用, 严重侵害正常用户的隐私、财产乃至人身安全, 对维护社会稳定和国家安全也提出了不小的挑战。例如, 360 手机卫士联合 360 数字安全、360 安全大脑发布的《2022 年上半年度中国手机安全状况报告》显示[2], 2022 年上半年度, 360 安全大脑共截获移动端新增恶意程序样本约 1079.7 万个, 同比增长了 180.5%, 平均每天截获新增手机恶意程序样本约 6.0 万个, 其中包括资费消耗、隐私窃取、违法违规、流氓行为和远程控制等一系列恶意应用类型; Synopsis 公司于 2021 年发布的《COVID-19 大流行期间最受欢迎的安卓应用程序的安全分析》报告显示[3], 63%的应用程序存在已知的安全漏洞, 平均每个应用程序有 39 个漏洞, 其中 44%的漏洞被认为是高风险的。因此, 如何检测安卓恶意应用、遏制针对安卓平台的恶意行为已经成为全球范围内的重要议题。越来越多国家和组织已经从立法方面入手, 保护移动端用户的合法权益。例如, 我国近几年相继颁布了《中华人民共和国网络安全法》、《移动智能终端应用软件预置和分发管理暂行规定》等各类法律规范; 欧盟颁布了《通用数据保护条例》保护公民个人信息。

为解决上述问题, 学术界已有大量的研究成果提出, 一些研究已经在工业界得到应用并取得了明显的成效。其中的代表性方法主要包括静态分析、动态分析以及基于安卓特征的机器学习检测方法[4-21]。具体而言, 静态分析旨在不运行目标代码的前提下, 对目标应用进行分析, 以检测相关恶意行为[4-10]; 与此不同的是, 动态分析通过实际运行应用来实时发现恶意行为[11-13]; 此外, 相关机器学习检测方法大多是通过对安卓应用的特征进行提取, 再编码为表示向量的方式送入分类器进行分类检测[18-21]。

虽然现已提出了上述各类研究成果, 但是安卓应用的复杂性使得构建一个较为全面的检测系统存在困难。具体而言, 由于安卓应用的数量庞大、功能繁多、行为语义复杂, 恶意攻击者可采用各种方式将其真实攻击意图隐藏在合法功能之中。现有研究方案已积累了大量的经验性检测数据以识别各类攻击行为: 如信息流分析专注于识别隐私泄露行为[4], 基于既定规则的检测方案专注于识别如虚假 UI[22]、逻辑炸弹[23]、远程控制[24]等行为, 基于机器学习的检测方案专注于识别满足指定特征的恶意行为[18-21], 行为重构方案专注于从所捕获的实时信息中发现潜在的恶意行为[14]。然而, 上述方案均只能识别有限类型的恶意行为, 当面对现阶段层出不穷、语义各异的安卓攻击手段时, 仅依靠部分检测方案难以实现对不同类型恶意应用的精确识别。若尝试通过逐一执行各类检测方案来实现对恶意行为的全面检测, 需耗费巨大的时间和人力成本。

本文提出并实现了一个基于多视图表示学习的安卓恶意应用检测系统 MVFDroid。首先, 通过构建异构信息网络, 对代表性检测方案(包括敏感数据流、可疑控制条件以及权限)所关注的应用行为进行高度抽象和统一建模; 然后, 设计基于组合视图的表示学习方案, 从多个视角对安卓应用行为的语义信息进行深度挖掘; 最后, 利用多视图融合的思想, 对目标安卓应用进行协同表征, 进而实现对安卓恶意应用的精确识别。本研究的主要目的在于充分发挥不同方案的检测潜力, 自动学习和利用不同视角下应用行为语义的一致性和互补性, 综合刻画安卓应用行为的恶意性。

在实现 MVFDroid 的过程中, 一个关键技术挑战是如何准确表征异构信息网络中的节点, 从而最大程度释放网络中的结构化行为语义信息。具体而言, 虽然 MVFDroid 所构建的异构信息网络包含丰富的安卓应用行为信息, 但如何在充分利用异构信息网络结构且不丢失其语义的前提下, 将不同类型的实体和关系表示为机器可读的向量形式, 进而将其充分运用于下游的节点分类任务(即安卓 APK 实体是否存在恶意性的分类问题), 以提高检测的准确度, 是本文需要重点解决的问题。

为了解决上述挑战, 本文以基于视图的游走方式对构建完毕的异构信息网络进行采样, 更大限度提升采样序列的语义信息丰富度, 再利用 NLP 领域中的 word2vec[25]方法学习节点的低维表示向量。本文所采用的游走方式是在 metapath2vec[26]的基础上, 以视图为单位构建不同视角下的元路径组合, 并将每个视图下的元路径组合视为整体, 指导随机游走的采样全过程。值得说明的是, 本文采用的基于视图的随机游走方式相比于其他采样方式[26-28]能够更充分的释放异构信息网络中的结构和语义信息。实验

表明, 本文所使用的采样方法能够有效挖掘异构信息网络蕴含的深层语义。

本文的主要贡献如下:

1) 提出了一种多视图表示学习安卓应用表征方案, 设计并实现了安卓恶意应用检测系统 MVFDroid。通过构建异构信息网络将敏感数据流、可疑控制条件和权限相关代表性检测方案的安卓特征进行静态提取和有机融合, 并通过深度神经网络模型对融合表示结果进行分类, 最终实现对安卓恶意应用的精确识别。

2) 从 Androzoo 和 Drebin 数据集中提取共 6310 个真实安卓应用, 其中良性应用 3208 个, 恶意应用 3102 个。在此数据集上的检测结果为 96.57%的准确率和 95.56%的 F1 值。实验结果表明, MVFDroid 在安卓恶意应用的检测方面均优于本文对比的其他代表性方案[29-31]。同时, MVFDroid 所使用的多视图表示学习方法相较于其他图表示学习方法(如, deepwalk[24]、metapath2vec[23]), 能够更全面丰富的描述安卓应用的结构和语义信息。

本文的结构组织如下: 第 2 节介绍当前安卓恶意应用检测的研究现状; 第 3 节总体上介绍本文所提出的 MVFDroid 检测系统的架构; 第 4 节详细叙述 MVFDroid 的实现细节和关键技术手段, 具体包括敏感数据流、可疑控制条件和权限相关特征的静态提取, 异构信息网络及其模式、元路径的设计, 多视图元路径随机游走策略指导的表示学习和基于多视图融合的应用分类模型的建立; 第 5 节评估 MVFDroid 的表征效果和检测性能; 第 6 节讨论目前本系统的局限性和未来工作计划; 第 7 节对本文进行总结。

## 2 研究现状

安卓恶意应用分析的传统方法主要包括静态分析、动态分析和混合分析三大类。静态分析是指在不运行目标应用的前提下, 对应用进行全面的分析, 寻找应用中存在的恶意执行路径。其中, 静态数据流分析是安卓恶意应用检测中的主流分析手段, 旨在通过检测是否存在敏感数据流的执行路径判断应用是否存在隐私泄露。FlowDroid[4]通过对安卓生命周期的精确建模, 并基于 IFDS 框架和污点分析实现了对安卓应用的过程内可疑数据流的检测。Epicc[5]、IC3[6]和 IccTA[7]实现了对安卓组件间通信过程的建模, 为 FlowDroid 补全了过程间分析的能力。Apkcombiner[8]通过将多个 apk 文件反编译后重组的方式达成安卓跨应用恶意数据流检测的目的。此外, ER Catcher[9]提出了一个对流、上下文和线程敏感的静态分析框架, 用于检测安卓应用中的事件竞争缺陷。JuCify[10]通过提取并合并本地代码和字节码的调用图来解决本地代码的分析问题。静态分析最显著的优点是对于目标应用分析的代码覆盖率高, 可在理论上遍历应用中间代码中的全部路径, 并生成对应的分析结果。然而由于反射、混淆、加密等机制的广泛应用, 目标应用的真实行为可能无法被静态分析技术完整还原。

不同于静态分析技术, 动态分析技术是在目标应用运行时追踪、记录、分析其暴露出的行为。该技术的优势在于检测结果具有实时性和真实性, 并且可以成功分析经过混淆、加密的应用代码。TaintDroid[11]提供了一套实时监视隐私数据信息流的方法。通过扩充 JAVA 解释器的执行栈, 给每一个变量都附加一个标签位。当变量被赋值、传递时, 相应变量的标签也被迭代更新。于是当移动应用执行数据传输行为时, 系统可以检查被传输数据的标签位, 以判断该行为是否包含隐私数据以及包含的隐私数据的类型。CopperDroid[12]通过监测移动应用运行时与底层系统之间的系统调用, 还有文件创建、读取, 短信收发、进程间通讯等交互事件, 重构出移动应用的行为。VetDroid[13]基于 TaintDroid, 自动提取软件内部与权限有关的行为, 构建出移动应用的隐私数据权限使用图, 辅助分析人员分析移动应用中存在的恶意行为。然而, 其最为关键的挑战在于应用程序的代码逻辑往往十分复杂(例如, 逻辑炸弹、垃圾代码等), 测试者很难在有限的时间内覆盖到目标应用中的每一块代码段, 这就导致动态信息流分析结果的召回率难以保证。

正因为静态分析和动态分析的优缺点都十分明显且具有较强的互补性, 研究人员尝试采用动静结合的混合分析技术来检测安卓恶意应用。AppAngio[14]通过将 API 层的日志信息和目标应用的控制流图相结合, 进而恢复出安卓应用运行时的完整上下文信息。Jitana[15]框架通过 ADB 实时感知设备中运行的应用并将其 dex 文件动态拉入, 并实现了一个类加载器静态的解析 dex 文件并生成各类静态信息流图, 从而辅助分析人员进行跨应用的恶意信息流检测。DINA[16]基于 Jitana 框架, 将反射和动态代码加载调用补全到控制流图中, 并持续执行增量动态分析, 以检测隐藏在反射和动态代码加载中的恶意意图。FSAFlow[17]尝试将路径跟踪逻辑和污点分析逻辑分离, 优化对代码的静态分析部分, 然后利用有限状态自动机排查出可能违反预定义的安全规则的路径, 并动态运行该路径从而排查代码的恶意性。然而这类混合分析框架依旧存在分析成本高、较为依赖安

全测试人员的判断、无法实现全自动分析的弊端。

随着机器学习相关技术日益成熟, 越来越多的研究人员利用机器学习技术对安卓恶意应用进行检测。AppContext[18]利用静态分析提取出应用中敏感行为的上下文特征, 包括触发敏感行为的事件以及与敏感行为相关的控制因素, 然后通过机器学习的方法, 分辨出包含恶意行为的安卓应用。ICCDetector[19]重点关注安卓的组件间和应用间通信中隐藏的恶意行为, 细致分析了良性应用和恶意应用在使用安卓跨组件通信相关机制时的不同点, 归纳出一组跨组件通信相关的关键特征并送入分类器中训练模型, 并利用该模型进行恶意应用检测。Li 等人[20]开发了一种基于权限使用分析的恶意应用检测系统 SigPID, 该系统通过三级剪枝挖掘应用相关权限数据, 以识别出可以有效区分良性和恶意应用的关键权限特征, 从而利用机器学习分类方法对不同类型的恶意和良性应用进行分类。PermPair[21]关注的是存在潜在风险的重要权限对组合, 以权限对为特征构建权限图, 并通过计算恶意应用权限图的相似度来区分良性和恶意应用。然而, 基于传统机器学习的检测技术通常利用特征工程来提取关键恶意应用特征并应用分类算法, 它们往往严重依赖于特征的选取, 只对某一类符合特定特征的恶意应用敏感。同时, 这类机器学习方法往往采用独热编码方式, 只是特征的堆叠, 无法挖掘语义层面的信息。

近年来, 图表示学习在理论上取得了巨大突破[26-28,32-38], 其在电子商务[39]、社交网络[40]和知识图谱[41]等领域的成功给予安卓恶意应用检测方面很大的启示。异构信息网络作为图网络的一种, 由于其节点和边的异质性, 天然具备丰富的语义和结构信息, 非常适合于建模表征安卓应用的多源异构语义信息, 现已有部分研究人员尝试将其应用于识别安卓恶意应用。HinDroid[31]将安卓应用、相关 API 以及它们之间的丰富关系结构化为异构信息网络, 并基于元路径来描述其语义相关性, 使用多核学习聚合不同元路径的相似度, 再经过分类器来预测恶意应用。HAWK[42]将安卓实体和行为关系构建为异构图, 并利用元结构学习相关的图节点向量, 创建了一个增量学习模型来处理节点的表示学习, 最终将训练出来的向量送入分类器中训练以识别恶意应用。AiDroid[43]除了构造异构信息网络以进行样本内节点表示学习, 还通过 k 阶邻居汇聚的方式, 在一定程度上解决了样本外节点的表示学习问题, 进一步提高了图神经网络在安卓恶意应用检测方面的效率。Dr.HIN[44]通过构建应用、开发者和移动设备间更高层次的语义和社会关系全面描述安卓应用, 引入结构化异构信息网络进行建模, 利用元路径引导的策略学习安卓应用的节点表示, 并首次尝试将解耦表示学习应用于异构信息网络中, 学习隐藏在表示向量中的潜在解释因素, 以检测不断进化的恶意应用。但是, 上述基于异构信息网络的安卓恶意应用检测研究, 其建模方式大都较为简单直接, 尚未充分利用安卓应用的丰富语义信息以及发挥不同方案的检测潜力。

apk2vec[45]设计了一个半监督的图表示学习框架, 以静态分析的过程间数据流图为基础, 将其进一步拆分为 API 依赖图、权限依赖图和污点分析依赖图后利用多视图融合进行图表示学习。Guen 等人[46]广泛采集安卓应用 Manifest 文件、dex 可执行文件以及共享库中的特征, 并利用独热编码的方式分别将几类特征表示为向量形式, 再利用多模态深度学习方法进行降维融合, 实现对安卓恶意应用的检测。虽然以上方案基于多个视图或模态来建模和表征安卓应用行为, 但其并未充分挖掘和利用视图内部特征的潜在关系和语义信息, 这将会影响其对于安卓恶意应用的识别效果。

## 3 MVFDroid 检测系统架构

MVFDroid 由数据收集及特征提取、异构信息网络构建、多视图表示学习和多视图融合应用分类四个模块组成, 本节将具体介绍每个部分的核心功能。

### 3.1 数据收集及特征提取模块

本模块主要用于完成前期的数据收集以及对应的特征提取工作。具体而言, 本文从目前安卓领域内权威的数据集中收集大量来自真实世界的安卓应用作为原始数据; 然后, 实现了一个静态提取器来提取数据集内应用中所有本工作需要使用的特征。

首先, 本模块需要收集大量良性和恶意安卓应用安装包作为原始数据集。具体来说, 本文所使用的安卓应用(包括良性应用和恶意应用)主要来自于 AndroZoo[47]数据集。同时, 为了丰富数据集中的恶意应用种类, 以验证 MVFDroid 对恶意应用检测的精确性和全面性, 辅以部分 Drebin[26]中的恶意应用来扩充恶意数据集。后续模块的工作均基于本模块收集的数据集进行。

在获得原始数据集后, 本模块根据后续构建异构信息网络的实际需要, 来静态提取相关特征。根据对安卓安全领域内代表性检测方案的深入调研, 本文从传统方案的检测优势出发, 以安卓应用的特点为落脚点, 选择从敏感数据流、可疑控制条件和权限

三个角度切入对安卓应用行为进行综合建模。因此, 本模块基于 FlowDroid[4]实现了一个特征提取器, 在前文收集的数据集中提取出上述三个检测视角的相关特征。具体来说, 本提取器基于静态分析方法, 反编译目标 APK 文件并静态扫描其 AndroidManifest. xml 文件和 dex 可执行文件, 并通过污点分析定位敏感数据流, 支配分析定位可疑控制条件, 自动化的提取后续建模所需的所有特征。具体细节及实现见 4.1 节。

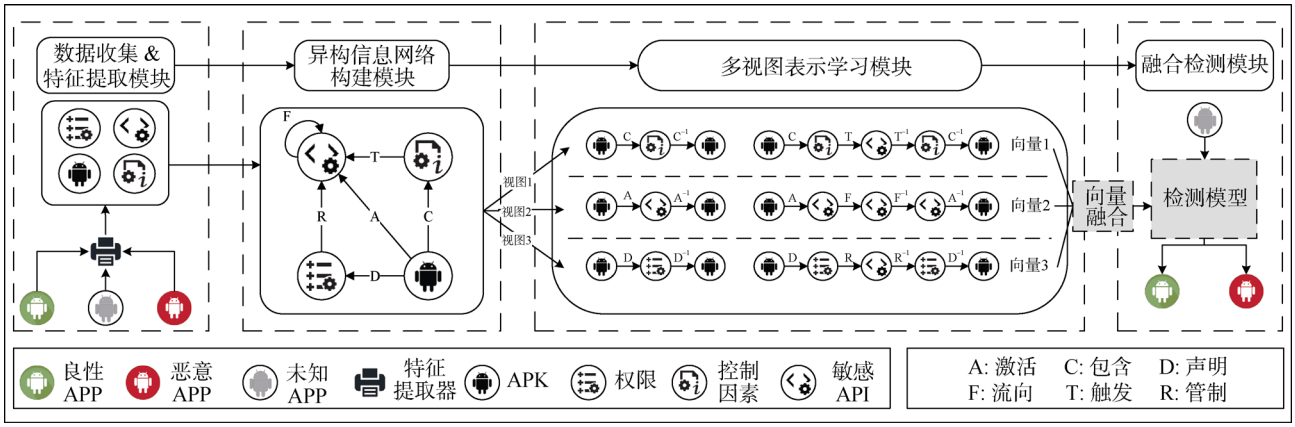


**图 1 MVFDroid 系统架构图**

**Figure 1 System architecture of MVFDroid**

### 3.2 异构信息网络构建模块

本模块根据前述模块所提取出的特征, 构造由敏感数据流、可疑控制条件和权限三个不同的语义视角组成的异构信息网络。具体来说, 为了建模多视图异构信息网络, 本文首先定义相关的实体节点(如权限、敏感 API 等)以及实体间的语义关系(APP-声明-权限, 权限-管理-敏感 API 等), 并基于以上实体及其关系构建异构信息网络模式。然后, 本模块在此模式上为不同的视图设计对应的元路径集合。本模块利用了安卓领域内的先验知识, 构建了一系列拥有明确语义的元路径。至此, 本模块完成本异构信息网络的构建, 为接下来从不同角度综合观察并刻画安卓应用行为的语义信息奠定基础。具体细节及实现见 4.2 节。

### 3.3 多视图表示学习模块

为了解决异构信息网络表示学习这一关键挑战, 本模块通过前述模块构建好的不同视图下的元路径集合来表达实体间的高阶关系, 并进一步扩展异构信息网络表示学习方法 metapath2vec[26], 采用基于视图的方式进行随机游走采样, 将高维图结构表示为一个低维向量, 从而为后续利用机器学习方法来完成恶意应用检测任务创造条件。具体细节及实现见 4.3 节。

### 3.4 多视图融合应用分类模块

本模块对前述模块中生成的、代表每个视图下安卓应用行为语义的向量表示进行融合, 并最终得到一个包含所有视图语义信息的应用节点向量表示, 再将该向量表示送入分类器进行检测, 以判断其是否具有恶意性。具体来说, 经过上述一系列步骤后, 本文成功将安卓应用的敏感数据流、可疑控制条件和权限视图下的相关结构和语义信息分别表示为一个易于训练且含义丰富的低维向量, 它反映了安卓应用在具体视图下的行为特征。鉴于这些视图的路径集合在进行表示学习后会得到不同的节点表征, 现需要一种多视图融合方法来对不同的表示向量进行融合, 从而获得安卓应用节点的最终表示结果。这样的向量融合相当于对安卓应用的不同观察角度得到的信息进行整合, 进一步提高低维向量表示的信息丰富度。本模块采用分权重融合的方式将三个视图的表示向量融合为一个包含丰富结构和语义信息的低维表示, 并将其送入深度神经网络模型, 训练出最终的安卓应用分类器。具体细节及实现见 4.4 节。

## 4 MVFDroid 的详细实现

本节将详细介绍本文提出的安卓恶意应用检测系统 MVFDroid 的详细思路、关键技术和具体实现。

### 4.1 特征选择及提取

基于对安卓应用行为及其代表性检测思路的归纳, 本文从敏感数据流、可疑控制条件和权限这三种对判定安卓应用是否具有恶意性起关键作用的语义信息出发, 对安卓应用及其相关行为进行抽象和建模。

**敏感数据流视图:** 以 FlowDroid 为代表的静态数据流分析是安卓恶意应用检测中非常重要的方法之一, 通过污点分析方法对敏感 API 数据流进行检测,

若应用中存在敏感数据的传输路径, 则说明该应用存在隐私泄露风险。具体来说, 本文依据 FlowDroid 工作中使用的敏感 API 列表, 并将其按照 Source API 和 Sink API 进行分类, 然后抽象出污点分析中的数据流关系 Source API – 流向 – Sink API 作为后续构建本视图的重要关系。

**可疑控制条件视图:** 以 AppContext 为代表的关注控制流相关行为的研究工作发现一些隐藏在控制逻辑下的攻击行为。例如图 2 代码段, 通过判断系统当前时间和上次访问时间来确定是否执行发送短信的逻辑。因此, 掌握控制敏感 API 执行的关键信息特征也是分析安卓应用行为的重要因素。具体来说, 本文选取 AppContext 中提出的控制因素(即其论文中总结的出现在控制条件中的敏感 API 所属的具体分类), 及其控制的具体的敏感 API 为可疑控制条件视图下的特征, 并以此构建该视图下的元路径中的重要关系。

```
01. long last = db.query("LastConnectTIme");
02. long current = System.currentTimeMillis();
03. if(current - last > 43200000){
04.     SmsManager.sendTextMessage();
05.     db.save("LastConnectTime", current);
06. }
```

**图 2　逻辑炸弹代码示例**

**Figure 2　Code example of logic bomb**

**权限视图:** 安卓权限机制是安卓的一种保护机制, 其被引入的初衷是为了保护用户的隐私数据以及受限操作, 然而一般用户对于某个应用所申请的权限往往没有明确的认知, 导致用户错误授予恶意应用敏感权限以致敏感信息泄露, 所以对安卓应用所申请的权限及其对应的敏感 API 的抽象建模也是十分重要的。因此, 为了抽象出权限视图的具体语义, 本文选取了敏感 API 及其使用时必须申请的相应的权限的映射关系, 来构建该视图下的元路径中的重要关系。

综上所述, 本文结合敏感数据流视图下的高危数据流、可疑控制条件视图下的关键控制因素以及安卓权限, 来综合观察安卓应用及其行为, 为后续的异构信息网络建模服务。值得注意的是, 为验证 MVFDroid 的有效性, 本文选择了上述敏感数据流视图、可疑控制条件视图以及权限视图进行后续的建模、融合以及检测工作。实际上, 其可扩展以支持对更多类型检测思路的统一建模。

基于上述观察, 本文通过对敏感数据流、可疑控制条件和权限这三个视图中的关键特征做进一步的细化和整合, 总结需要提取的特征如表 1 所示。需要说明的是, 每一种视图中都包含两类特征, 一种是节点类型的特征(如控制因素、敏感 API), 另一种是关系型特征(如控制因素 – 触发 – 敏感 API)。为了提取相关特征, 本文基于 FlowDroid 实现了一个静态提取器, 自动化的提取上述特征。

**表 1　特征分类表**

**Table 1　Feature classification table**

| | |
|---|---|
| 通用特征 | APK 包名, 组件名, 组件种类, <exported> |
| 敏感数据流相关特征 | Source API, Sink API, Source API–流向–Sink API |
| 可疑控制条件相关特征 | 控制因素, 敏感 API, 控制因素–触发–敏感 API |
| 权限相关特征 | <uses-permission>, <permission>, <permission-Level>, 敏感 API, 权限–管制–敏感 API |

## 4.2　异构信息网络构建

根据特征提取模块所提取出的特征, 本节将详细介绍如何构造以敏感数据流、可疑控制条件和权限三个不同语义视图所构成的异构信息网络。首先, 本节需要介绍异构信息网络和元路径以及表示学习的形式化定义。

### 4.2.1　基本定义

**定义 1.异构信息网络.**异构信息网络是包含实体类型映射 $\phi: \mathsf{V} \to \mathsf{A}$ 和关系类型映射 $\psi: \mathsf{E} \to \mathsf{R}$ 的图结构, 记为 $\mathsf{G}=\{\mathsf{V},\mathsf{E}\}$, 其中 $\mathsf{V}$ 和 $\mathsf{E}$ 分别代表实体集合和关系集合, $\mathsf{A}$ 和 $\mathsf{R}$ 分别表示实体类型集合和关系类型集合, 并规定实体类型数量 $|\mathsf{A}|>1$ 或关系类型数量 $|\mathsf{R}|>1$。同时, 异构信息网络 $\mathsf{G}$ 的网络模式定义为 $\mathsf{T}_{\mathsf{G}}=\{\mathsf{A},\mathsf{R}\}$, 表示为一张以实体类型 $\mathsf{A}$ 和关系类型 $\mathsf{R}$ 连接的网络示意图。

**定义 2.元路径.**一条元路径 $\mathsf{P}$ 被定义为异构信息网络的网络模式 $\mathsf{T}_{\mathsf{G}}=\{\mathsf{A},\mathsf{R}\}$ 中的一条路径, 形如 $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \ldots \xrightarrow{R_L} A_{L+1}$, 定义了实体类型 $A_1$ 和 $A_{L+1}$ 之间的复合关系 $R=R_1 \cdot R_2 \cdot \ldots \cdot R_L$, 其中 $\cdot$ 表示关系复合运算符, $L$ 为元路径 $\mathsf{P}$ 的长度。

**定义 3.异构信息网络表示学习.**给定一个异构信息网络 $\mathsf{G}=\{\mathsf{V},\mathsf{E}\}$, 其表示学习任务被定义为学习一个函数 $f: \mathsf{V} \to \mathbb{R}^d$, 其将每一个节点 $v \in \mathsf{V}$ 映射为一个空间 $\mathbb{R}^d$ 中的 $d$ 维向量, 其中 $d \ll |\mathsf{V}|$, 并且该向量能保留其中的结构和语义信息。

### 4.2.2　网络实体节点及其关系构建

本小节将详细介绍网络实体节点及其关系是如何设计并构建的。具体来说, 根据前文选定的敏感数

据流、可疑控制条件和权限视图以及提取出的对应的特征(见表 1), 本文构建了安卓 APK、权限、控制因素、敏感 API 四类节点, 并定义了以下 6 种关系:

R1: 为描述一个 APK 文件声明某个权限这一行为, 本文生成关系 APK-声明-权限的矩阵 $\boldsymbol{D}$, 其中的元素 $d_{i,j} \in \{0,1\}$ 表示 APK $i$ 中声明了权限 $j$。

R2: 为描述安卓应用中包含的控制因素(即控制敏感 API 的关键控制流信息), 本文生成 APK-包含-控制因素矩阵 $\boldsymbol{C}$, 其中的元素 $c_{i,j} \in \{0,1\}$ 表示 APK $i$ 中包含控制因素 $j$。

R3: 为了描述控制因素所能触发的敏感 API, 本文生成控制因素-触发-敏感 API 矩阵 $\boldsymbol{T}$, 其中的元素 $t_{i,j} \in \{0,1\}$ 表示控制因素 $i$ 能触发敏感 API $j$。

R4: 为了刻画数据流污点分析中最关键的内容, 即 Source API → Sink API(其中, Source API 和 Sink API 都属于敏感 API), 本文生成敏感 API-流向-敏感 API 矩阵 $\boldsymbol{F}$, 其中的元素 $f_{i,j} \in \{0,1\}$ 表示存在一条由敏感 API $i$ 流向敏感 API $j$ 的执行路径。

R5: 为表明安卓应用中存在敏感 API 这一关系, 本文生成 APK-激活-敏感 API 矩阵 $\boldsymbol{A}$, 其中的元素 $a_{i,j} \in \{0,1\}$ 表示 APK $i$ 能激活敏感 API $j$。

R6: 为表明权限及其控制的敏感 API 之间的关系, 本文生成权限-管制-敏感 API 矩阵 $\boldsymbol{R}$, 其中的元素 $r_{i,j} \in \{0,1\}$ .表示权限 $i$ 能管制敏感 API $j$。

根据上文精确刻画的安卓应用及相关实体间的丰富关系(即 R1～R6), 本文使用异构信息网络对其进行建模, 并构建元路径来表示安卓应用及其敏感数据流、可疑控制条件、权限视图下的不同关系, 使得关于安卓应用的丰富结构和语义信息得到充分的表达。本节剩余部分将详细介绍如何利用上述实体和关系特征设计异构信息网络和元路径。

### 4.2.3 网络模式及其元路径构建

本节将基于 4.2.2 节中构建的实体节点及其关系, 来构建异构信息网络的网络模式, 并根据该网络模式以及安卓领域的先验知识来设计每个视图的元路径。

本节首先构建了本文所需的异构信息网络的网络模式。异构信息网络不仅是一种数据关联的网络结构, 也是一种实体类别关联的高度抽象。具体到安卓恶意应用的检测任务, 本文抽象了敏感数据流、可疑控制条件以及权限三种视图共四种实体类型(即安卓 APK、权限、控制因素、敏感 API)以及它们之间的 6 种关系(即 R1-R6), 构建如图 3 所示的异构信息网络的网络模式(Network Schema)。通过本网络模式得到异构信息网络, 使得安卓应用的行为能够以一种多视图的方式得到表达。

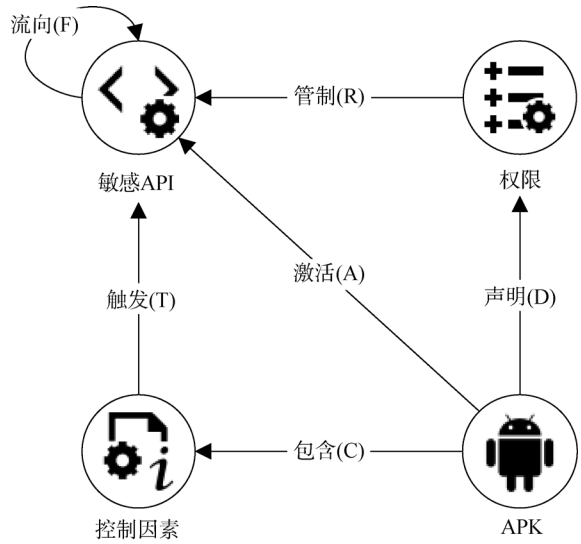


**图 3 MVFDroid 异构信息网络模式图**

**Figure 3 Network schema for HIN in MVFDroid**

为了充分利用异构信息网络中不同实体之间的关系来刻画不同的安卓应用及其行为, 本文通过构建多视图元路径, 综合利用代表性分析方法的优势和安卓本身的特点来描述安卓应用间的关联性。如 4.2.1 定义 2 中对元路径的形式化表述, 元路径描述的是首尾节点间的复合关系, 并且一般以节点对称的结构设计。通过这种方式, 元路径可以用来表达两个同类节点(本文中为安卓应用节点)间在某一特定复合关系下的相似性。

具体来说, 本文根据不同视图的检测视角, 设计了不同的元路径来表征异构信息网络节点和关系间的深层次语义关系。为了构建意义明确的元路径, 本文基于安卓领域的先验知识, 并将其扩展到安卓恶意应用检测的目标上。然后, 如图 4 所示, 本文以三大视图(即敏感数据流、可疑控制条件、权限)和更细粒度的 6 种关系(R1～R6)为基础, 设计了 6 条具有不同含义的元路径(即 MP1～MP6)以描述不同安卓应用实体间的关联性。不同元路径从不同角度描述两个安卓应用间的关系, 例如可疑控制条件视图下的 MP1 表明两个安卓应用包含同样的控制因素; 敏感数据流视图下的 MP4 表明两个安卓应用存在到达同一敏感 API 的数据流; 而权限视图下的 MP6 表明两个安卓应用都通过声明权限的方式调用了同一敏感 API。需要注意的是, 在元路径中, $^{-1}$ 符号表示被动关系。以 MP4 为例, F 代表流向, 指的是由存在一条敏感数据流由某个 Source API 流向某个 Sink API, 那么 $F^{-1}$ 指的就是存在一条被 Sink API 所接收的敏感数据流是由 Source API 所产生的。

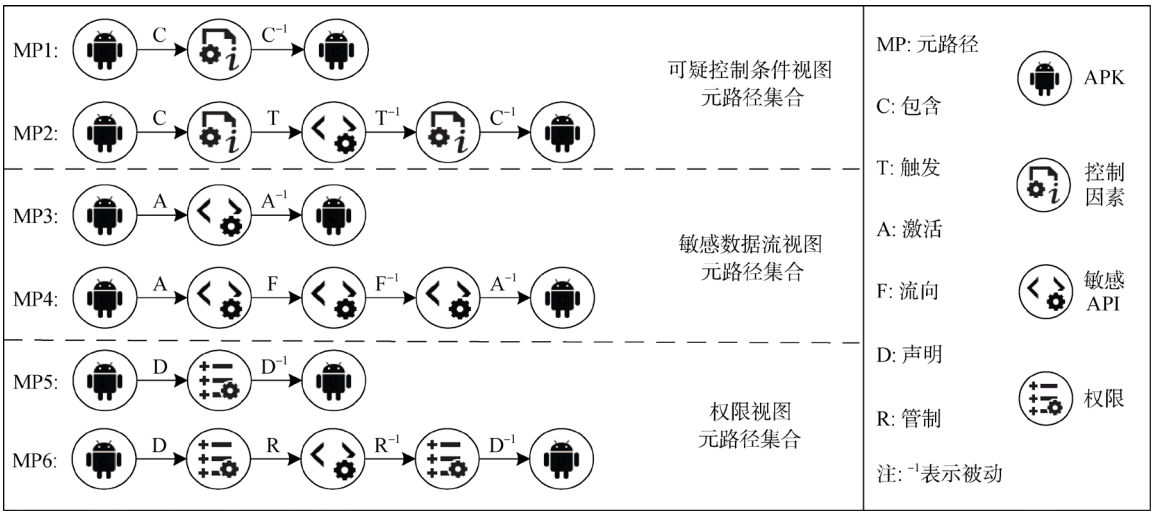


**图 4　MVFDroid 多视图元路径**

**Figure 4　Multi-view metapaths for MVFDroid**

本文构建元路径的目的在于表达出两个同类节点(即安卓应用节点)间在某一特定复合关系下的语义相似性, 进而实现安卓恶意应用检测。具体而言, 本文根据给定元路径在异构信息网络上进行采样(见 4.3.1)得到的序列可反映出哪些安卓应用间存在该元路径中蕴含的语义联系。本文认为在善意应用间采集到的序列与在恶意应用间采集到的序列存在明显差异, 并以此来区分出善意应用和恶意应用(具体见 4.3.2)。

## 4.3　异构信息网络表示学习

为了解决前文提到的异构信息网络表示学习的难点, 本节通过上文设计的不同视图下的元路径来制定异构信息网络中实体间的高阶关系, 并在异构信息网络表示学习方法 metapath2vec 的基础上, 实现基于视图的随机游走策略, 对高维图结构中的每一个视图进行充分采样, 然后通过基于 skip-gram 的 word2vec 方式来学习采样序列中每一个节点的低维向量表示。

### 4.3.1　多视图元路径指导的随机游走

为了衡量异构信息网络实体(如安卓应用)间的关联性, 首先需要对构建完毕的异构信息网络进行表示学习。传统的异构信息网络表示学习[48-50]主要集中在对异构网络中的矩阵(如邻接矩阵)进行矩阵分解, 以生成节点的低维特征。然而, 这样的方法在应用于大规模矩阵时的计算成本非常高, 并且存在统计方面的缺陷[28]。为了减少训练成本, 则需要寻找可扩展的异构信息网络表示学习方法。

本文采用一种基于视图的随机游走策略来获得实体序列, 再以此为输入, 利用 NLP 领域内的 word2vec 思想获得节点在每个视图下的表示向量, 从而完成表示学习任务[51]。具体来说, 本文借鉴异构信息网络表示学习技术 metapath2vec 的思想, 该技术基于元路径的随机游走和异质 skip-gram 来学习异构信息网络的潜在低维表示, 从而在降维的同时能最大程度的保留不同类型节点间的语义和结构信息。传统的 metapath2vec 根据预先定义的元路径(即 MP1～MP6)在异构图中随机游走遍历, 基于以此方式采样后的序列, 利用 word2vec 思想获得节点在该元路径指导下的向量表示。然而, 使用这种方式得到每个安卓应用节点在六种不同元路径下生成的向量表示则丢失了前期建模时选择的三种基本视图间的区别和联系。具体来说, 4.2.3 节中构建的元路径集合是分属于不同的视图的, 而每个视图下的元路径之间的语义联系更紧密, 如果将这样的路径组合充分利用, 则可以释放并挖掘出更多的语义信息。因此, 本文在沿用 metapath2vec 时进行了一定的改进, 引入了一种不同于传统元路径引导的随机游走方法, 以三种基本视图为主体, 每个视图下的元路径为整体(即可疑控制条件视图下的 MP1～MP2、敏感数据流视图下的 MP3～MP4、权限视图下的 MP5～MP6 各自看成一个整体集合)来指导多视图元路径的随机游走, 然后利用 skip-gram 来学习每个视图的有效节点表示。

具体来说, 给定一个网络模式为 $\mathsf{T_G}=\{\mathsf{A},\mathsf{R}\}$ 的异构信息网络 $\mathsf{G}=\{\mathsf{V},\mathsf{E}\}$, 并规定每个视图下的元路径集合为 $\mathsf{S}=\{\mathsf{P}_k\}_{j=1}^{n}$ (例如, 控制流视图下的元路径集合 $\mathsf{S}=\{MP1,MP2\}$), 其中的每一条元路径的格式都为: $A_1\rightarrow\ldots A_t\rightarrow A_{t+1}\ldots\rightarrow A_l$。本文通过为每一个视图定义一个随机游走器来遍历异构信息网络, 随机游走器首先从集合 $\mathsf{S}$ 中随机的选择一条元路径记为 $\mathsf{P}_k$, 其在步骤 $i$ 处的概率转移方程定义如下:

$$p(v^{i+1}\mid v^i_{A_t},\mathsf{S})=\begin{cases}\dfrac{\lambda}{|\mathsf{S}|}\dfrac{1}{|N_{A_{t+1}}(v^i_{A_t})|} & if\ (v^{i+1},v^i_{A_t})\in\mathsf{E},\phi(v^i_{A_t})=A_{apk},\phi(v^{i+1})=A_{t+1}\\ \dfrac{1}{|N_{A_{t+1}}(v^i_{A_t})|} & if\ (v^{i+1},v^i_{A_t})\in\mathsf{E},\phi(v^i_{A_t})\neq A_{apk},\phi(v^{i+1})=A_{t+1},(A_t,A_{t+1}\in\mathsf{P}_k)\\ 0 & otherwise,\end{cases}\tag{1}$$

其中，$\phi$ 是节点类型映射函数，$N_{A_{t+1}}(v^i_{A_t})$ 表示节点 $v^i_{A_t}$ 的 $A_{t+1}$ 类型的邻居，$A_{apk}$ 是 APK 节点的实体类型，$\lambda$ 是给定集合 $\mathsf{S}$ 中满足由 $A_{apk}\to A_{t+1}$ 开始的元路径的数量。由以上策略生成的游走路径可以保留异构信息网络不同类型节点间的结构和语义关系。因此，通过此方式采样得到的节点序列放入 skip-gram 训练后得到的低维向量表示能有效反映出原异构信息网络的结构和语义关系。

#### 4.3.2 基于 word2vec 的应用节点表示

本小节基于上文的采样序列结果，通过自然语言理解领域中获得序列向量表示的方法获得实体节点的向量表示。具体来说，通过实施上述基于多视图元路径的随机游走策略，本文成功将异构信息网络中实体节点间的关系序列映射到文本语料库中的词语上下文这一概念中(即语料库中的一个句子对应于一条采样路径，一个词语对应于异构信息网络中的一个节点)。因此，可将 word2vec[25]的 skip-gram 模型应用于采样而来的序列上，来最小化观察节点邻居(窗口长度为 $w$)时的损失，从而获得节点的低维向量。skip-gram 的目标函数为:

$$\operatorname{argmin}_Y \sum_{-w\leqslant k\leqslant w,\ j\neq k} -\log p(v_{j+k}\mid Y(v_j)) \tag{2}$$

其中，$Y(v_j)$ 为节点 $v_j$ 的当前向量表示，$p(v_{j+k}\mid Y(v_j))$ 则使用 softmax 函数定义:

$$p(v_{j+k}\mid Y(v_j))=\frac{\exp(Y(v_{j+k})\cdot Y(v_j))}{\sum_{q=1}^{|\mathsf{V}|}\exp(Y(v_q)\cdot Y(v_j))} \tag{3}$$

在实际训练 skip-gram 的过程中，由于考虑到整体的训练性能，本文应用层次化 softmax 技术来计算公式(3)，然后采用随机梯度下降的方法来训练整个 skip-gram。

### 4.4 基于多视图融合的安卓恶意应用分类

经过上述步骤，本文基于三种视图的表示学习后分别得到安卓应用实体节点的三种不同的低维向量表示，它们描述了不同语义下的安卓应用行为，其形式化表示如下: $\mathbf{S}=\{\mathsf{S}_i\}_{i=1}^m$，其中，$m$ 表示视图集合的数量。根据本文的视图设计可知，此处 $m=3$ 且 $\mathbf{S}=\{(MP1,MP2),(MP3,MP4),(MP5,MP6)\}$。

鉴于这些视图的路径集合在进行表示学习后会得到不同的节点表征，现需要一种多视图融合方法来对不同的表示向量进行聚合，从而获得安卓应用节点的最终表示结果。本文以如下方式进行多视图融合: 基于 $m$ 个视图表示学习后可以获得 $m$ 种节点表示，记为 $Y_i(i=1,...,m)$，则合并后的节点表示可以记为: $Y'=\sum_{i=1}^{m}(\alpha_i\times Y_i)$，其中 $\alpha_i(i=1,...,m)$ 是 $Y_i$ 的权重，本文通过实验的方式确定各个视图的权重，详细说明与结果见 5.5 节。

最后，将融合表示向量送入训练好的 DNN 分类模型中完成对安卓恶意应用的检测。具体来说，上文成功将安卓应用的敏感数据流、可疑控制条件和权限视图下的相关结构和语义信息表示为一个易于训练且含义丰富的低维向量，它反映了安卓应用多方面的行为特征。根据该表示向量，本文训练了一个 DNN 分类器，从而以安卓应用是否具有恶意性对其进行分类。

## 5 实验与评估

为验证本文提出的 MVFDroid 安卓恶意应用检测系统的检测能力，本章将尝试回答以下几个问题:

研究问题 1: 与当前的代表性检测方案相比，MVFDroid 在真实世界应用中的检测效果如何？

研究问题 2: 与当前的代表性图表示学习算法相比，MVFDroid 中所使用的图表示学习算法的有效性如何？

研究问题 3: 如何设置合适的实验参数以保证 MVFDroid 的检测效果？

根据以上问题，本文基于真实安卓应用数据集进行了多项实验。下文首先介绍本实验的相关配置、数据集和评价指标，然后基于各类指标设计实验进行客观的评估，以回答提出的问题。

### 5.1 实验设置

**数据集:** 为了客观评价本系统的各方面性能，本文的数据集从 AndroZoo 中收集了 3208 个良性应用和 2190 个恶意样本。同时，为了提升数据集的恶

意样本丰富度, 本文又从 Drebin 提供的恶意应用数据集中随机挑选了 912 个样本作为补充。

**实验配置:** 表 2 展示了本文的实验环境和具体配置。

**表 2 实验配置表**

**Table 2 Experimental configuration table**

| 设备 | 型号 |
|---|---|
| CPU | CPU IntelBroadwellE5-2620v4@2.10GHz |
| 内存容量 | 128G |
| 操作系统 | Ubuntu16.04LTS |
| GPU | GeForce RTX1080Ti(GPU 显存 11.0G) |

**评价指标:** 本文所采用的实验评价指标为准确率、精确率、召回率和 F1 值, 其具体的计算公式如表 3 所示。

**表 3 评价指标表**

**Table 3 Evaluation index table**

| 评价指标 | 计算公式 |
|---|---|
| ACC | $\frac{TP+TN}{TP+TN+FP+FN}$ |
| Precision | $\frac{TP}{TP+FP}$ |
| Recall | $\frac{TP}{TP+FN}$ |
| F1 | $\frac{2\times Precision\times Recall}{Precision+Recall}$ |

其中, TP、TN、FP、FN 分别代表样本中真阳性、真阴性、假阳性、假阴性的样本数。此外, F1 值是对精确率和召回率的一种综合评价指标。

## 5.2 研究问题 1: MVFDroid 检测效果对比实验

为了探究 MVFDroid 在真实世界应用中的检测能力并回答研究问题 1, 本节选取研究界的一些代表性工作作为对比方案。具体选取的方案如下所示。

Drebin[29]: Drebin 是基于机器学习方法进行安卓恶意应用检测任务的代表工作, 它广泛采集了安卓应用的多种特征, 再利用独热的编码方式获得了应用的向量表示。

HinDroid[31]: HinDroid 利用安卓应用和相关 API 间的关系构建异构信息网络, 并利用元路径和多核学习来对安卓应用节点进行表示学习, 最后使用分类器来识别恶意应用。

MaMaDroid[30]: MaMaDroid 基于静态分析提取出 APK 中的 API 调用图, 然后遍历图中所有可执行路径得到完整的 API 序列, 再通过马尔科夫链对序列进行建模, 得到对应 APK 的特征向量, 最后利用分类器识别恶意应用。值得一提的是, 在得到 API 序列时, 由于 API 的种类非常多, 无法精确到每一个具体的 API, 此工作便将 API 进行了归类, 即把 API 抽象成对应的包或者家族以降低复杂度。本实验在使用该模型时采取的方法是将其 API 序列抽象成包名处理的。

由于 Drebin 能代表大量使用传统机器学习方式来表示特征和训练向量的工作, HinDroid 能代表同样使用构建异构信息网络的方法来实现恶意应用检测的工作, MaMaDroid 能体现使用其他基于机器学习方法的恶意应用检测工作, 同时以上工作本身也具有较好的检测效果, 本实验以 5.1 节中收集的真实世界应用为数据集, 分别使用以上三个检测方案和最佳实验参数训练的 MVFDroid(具体参数选择见 5.4 节)来对应用进行检测, 以探究 MVFDroid 在真实世界中的检测能力。具体检测结果如表 4 所示。

**表 4 不同方案对恶意应用检测效果对比**

**Table 4 Comparison of malicious application detection effects for different schemes** (%)

| 方法 | ACC | Precision | Recall | F1 |
|---|---|---|---|---|
| Drebin | 94.56 | 95.33 | 90.94 | 93.08 |
| HinDroid | 93.88 | 95.35 | 93.18 | 94.25 |
| MaMaDroid | 90.09 | 88.29 | 87.89 | 88.09 |
| MVFDroid | **96.57** | **96.32** | **94.82** | **95.56** |

实验结果显示, MVFDroid 的整体检测效果均优于其他三种方法。具体而言, MVFDroid 检测的准确率达到 96.57%、F1 值达到 95.56%, 而 Drebin 检测的准确率为 94.56%、F1 值为 93.06%, HinDroid 检测准确率为 93.88%、F1 值为 94.25%, MaMaDroid 检测准确率 90.09%、F1 值为 88.09%。该实验结果说明, 与基于独热编码或其他降维方法所产生的向量比较, 基于图表示学习方法得到的表示向量, 其对于语义层面的信息的挖掘能力更强, 这在很大程度上影响了最终的表示向量的语义丰富程度, 同时对于具体的相关下游任务(本实验中是安卓恶意应用检测)的敏感性也更高; 与同样构建异构信息网络的方法相比, 异构信息网络构建的全面性也对最终的检测结果有一定的影响, 以 HinDroid 为代表的一些方法在建图时所选择的实体节点存在类型单一或不够深入的问题, 但它们都可以作为本工作的一种补充视图以加强本工作对安卓应用的表征能力。

## 5.3 研究问题 2: MVFDroid 图表示学习算法的有效性验证实验

MVFDroid 检测系统的检测效果很大程度上取

决于系统所采用的图表示学习算法的表示能力。因此, 为了客观评价图表示学习算法的有效性并回答研究问题 2, 本部分首先设计实验验证基于视图的表示学习的有效性, 然后选取具有代表性的相关图表示学习算法作为参考基准, 横向评价本文使用的表示学习算法的表示能力。更具体的说, 本部分实验首先利用“视图融合的表征结果明显优于基于单个视图或单个元路径的表征结果”证明本文使用图表示学习算法的有效性, 然后利用“图表示学习方法性能对比”实验结果来说明图表示学习算法所具备的良好表示能力。

### 5.3.1 不同视图及其元路径的表征能力对比实验

为了验证本文 4.3 节使用的基于多视图的表示学习方法的可行性和有效性, 本部分基于 5.1 节中收集的数据集开展实验来具体的评估不同元路径、不同视图的表征能力。具体来说, 本实验首先利用 metapath2vec 的方法对每一条单独的元路径进行表征, 获得安卓应用实体节点的一个潜在的低维向量表示; 然后利用本文使用的基于视图的表征方法获得一个同样维度的安卓应用实体节点表示向量; 最后再利用本文 4.4 节中使用的多视图融合方法获得视图融合的安卓应用实体节点表示向量。这些表示向量最终都被送入训练好的 DNN 模型中进行二分类任务, 以判断每一种表征方式的表征能力如何。如前文所述, 本文共有 6 条不同的元路径(即 MP1-MP6), 以及 3 种不同的视图(即可疑控制条件、敏感数据流和权限)。本实验的实验结果如表 5 所示。

**表 5 不同元路径的检测结果**

**Table 5 Detection results of different metapaths** (%)

| 路径名 | ACC | Precision | Recall | F1 |
|---|---|---|---|---|
| MP1 | 90.40 | 90.74 | 92.70 | 91.71 |
| MP2 | 91.21 | 91.67 | 93.68 | 92.66 |
| 可疑控制条件视图 | 93.00 | 93.86 | 94.11 | 93.99 |
| MP3 | 93.40 | 94.76 | 89.52 | 92.07 |
| MP4 | 90.18 | 89.29 | 88.61 | 88.95 |
| 敏感数据流视图 | 94.56 | 94.31 | 90.10 | 92.15 |
| MP5 | 88.57 | 88.15 | 86.33 | 87.23 |
| MP6 | 89.16 | 87.07 | 88.09 | 87.57 |
| 权限视图 | 89.84 | 87.66 | 90.57 | 89.09 |
| 视图融合 | **96.57** | **96.32** | **94.82** | **95.56** |

从表 5 中可以看出, 不同的元路径或视图在表征安卓应用的恶意性中表现出了不同的性能, 这也间接说明每一种元路径或视图间所描述的语义信息也是不同的。除此之外, 本实验还可以得出以下结论: (1)基于视图的表征结果总体上都优于其内部单独的元路径的表征结果。其中, 可疑控制条件视图和权限视图的表征结果从总体上看均明显优于每一个元路径的单独表征; 敏感数据流视图的表征结果虽然从精确率上看略低于 MP3 单独的表征, 但有了 MP4 的语义补充后, 从其准确率、召回率和 F1 值上来看还是优于 MP3 单独表征的; (2)视图融合的表征结果明显优于基于单个视图的表征结果。虽然从单个视图的表征结果来看, 权限视图的整体表现稍差于其他两个视图, 但经过融合后由于表示向量的整体语义更为丰富, 其表示能力也得到了进一步的提升。

上述实验结果可证明本文所使用的表示学习方法具有可行性和有效性, 能够较为充分的表达安卓应用的恶意性相关的语义信息, 同时不同视图间的语义可形成互补效应, 进而能够有效完成安卓恶意应用检测任务。

### 5.3.2 图表示学习算法表征能力对比实验

本部分设计实验对本图表示学习方法的表征和检测能力和效果进行横向对比考察。具体来说, 本实验选取目前图表示学习领域具有代表性的几个工作, 基于前文构建的异构信息网络, 以及相关的元路径, 对图中的节点进行表示学习, 再将得到的表示向量送入 5.3.1 节中使用的训练好的 DNN 模型分类检测, 从而与 MVFDroid 的检测效果进行横向对比, 评价本系统采用的表示学习方法的性能。本实验的实验结果如表 6 所示。

**表 6 不同图表示学习算法效果对比**

**Table 6 Comparison of different graph representation learning algorithms effects** (%)

| 方法 | ACC | Precision | Recall | F1 |
|---|---|---|---|---|
| DeepWalk | 88.41 | 86.93 | 82.78 | 84.80 |
| node2vec | 90.28 | 91.71 | 83.10 | 87.19 |
| metapath2vec | 93.40 | 94.76 | 89.52 | 92.07 |
| MVFDroid 的图表示学习算法 | **96.57** | **96.32** | **94.82** | **95.56** |

如前所述, 为了评估 MVFDroid 系统的性能, 本文选择了领域内具有代表性的图表示学习工作作为参考基准进行对比, 下面将简要介绍一下相关方法及其参数配置:

DeepWalk[27]: DeepWalk 是基于完全随机游走采样的表示学习方法, 该方法最早针对同构网络提出, 是最为经典也是最为简单的一种图表示学习算法。

node2vec[28]: 该工作是对 DeepWalk 算法的改进, 利用权重来调节随机游走的概率, 形成一种综合考

虑 DFS 领域和 BFS 领域的、有偏的随机游走策略。

metapath2vec[26]: 该工作是基于元路径随机游走采样的表示学习算法, 不同于前述方法, 该方法是针对异构信息网络特点提出的一种表示学习方法, 在异构信息网络表示学习任务中具有代表性。

对于上述所有算法, 本文在进行实验时都采用了与 MVFDroid 一致的超参数(见 5.4 节), 其中 node2vec 特有的超参数 $p$ 和 $q$ 的设置本文选择了偏向于 DFS 的游走策略, 具体为 $p$=1, $q$=0.5。另外, 对于 metapath2vec 的评估, 本实验直接选取了 MP3 的结果。具体来说, 由于 MP3 直接刻画了安卓应用和敏感API的关系, 而敏感API为本文构建的异构信息网络的核心节点, 并且所有视图中都存在敏感 API 这一识别恶意应用的关键特征; 另外, 如表 5 所示, 由于在实际实验时 MP3 的实验结果确实是所有以单独元路径游走得到序列(即 MP1～MP6)中效果最好的路径。因此, 本文认为选取 MP3 作为 metapath2vec 的元路径是最合适的。

如表 6 所示, MVFDroid 所采用的表示学习方法, 各项评价指标上都优于其他表示学习方法, 这表明本表示学习方法具有相当的表示能力。本文认为, MVFDroid 所采用的表示学习方法之所以能有更好的检测效果, 正是因为基于视图的随机游走方式能更充分、更深层次的挖掘和释放异构信息网络中蕴含的丰富信息。

### 5.4 研究问题 3: 超参数和融合权重选择实验

为了探究研究问题 3, 本部分将系统实现过程中所有需要人为确定的超参数列举出来, 并采用实验的方法选择最佳参数。

具体来说, 本节针对 4.3 节图表示学习任务中涉及的相关超参数和 4.4 节的向量融合权重进行实验, 从而判断并选择最佳实验参数。

基于 5.1 节中收集的数据集, 本实验首先对能够影响安卓恶意应用检测效果的超参数(即每个节点的游走次数 $r$、游走长度 $l$、窗口大小 $w$、表示向量维度 $d$)进行选择实验。需要注意的是, 由于多视图的融合需要建立在各个视图生成的表征向量基础上进行, 因此本实验对于多视图表示学习的超参数选择, 考虑从每个视图各自的表征结果的好坏为评价指标来选取最佳参数; 又由于本文对于每个视图元路径集合的设计在结构上区别不大, 本实验最终考虑选取其中的一个视图做相关超参数选择实验, 并假定该超参数可以适应于所有的三个视图。最终本实验选取敏感数据流视图作为实验对象, 其实验结果如图 5 所示。

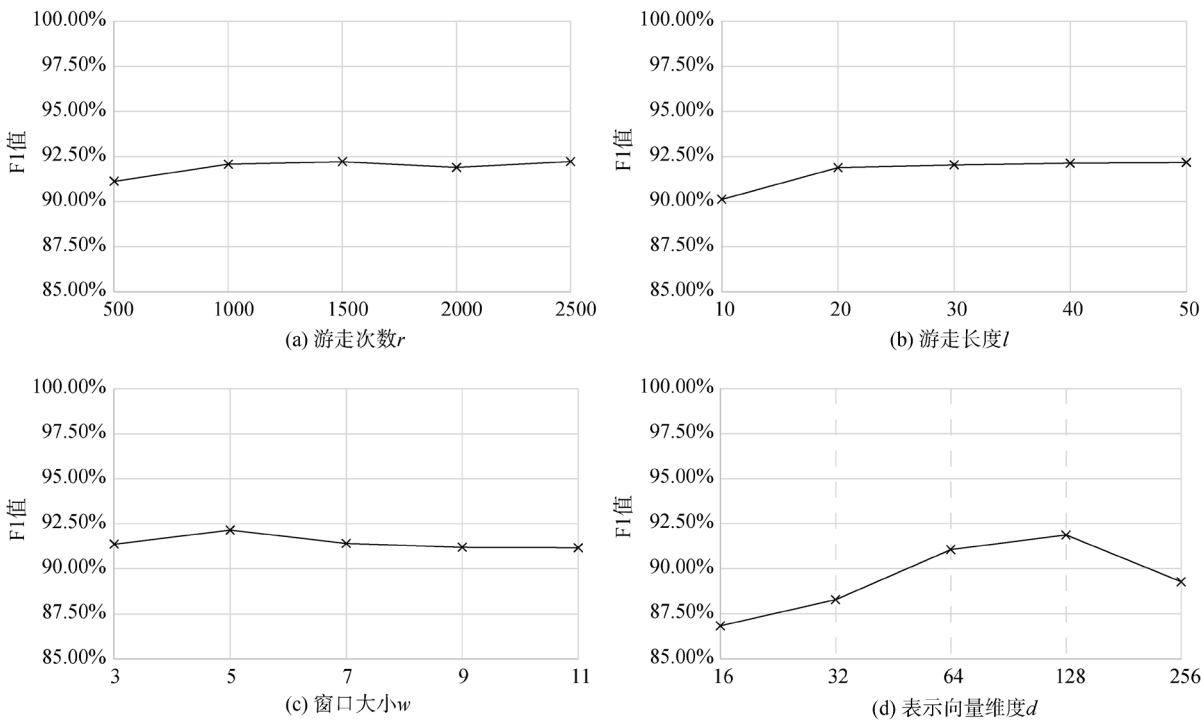


**图 5 超参数选择实验结果**

**Figure 5 Experimental results of hyperparameters selection**

由实验结果可知, 当 $r$ =1000, $l$ =30, $w$ =5 和 $d$ =128 时, 安卓恶意应用检测效果最佳。值得说明的是, 如图 5 中(a)和(b)中曲线, 在参数值达到某一阈值后, 从曲线可以看出后续的 F1 值变化很小, 因

此, 本文在兼顾实验效率的前提下, 选择了综合性能最佳的参数, 以(b)中的游走长度为例, 事实上当游走长度达到 20 及以上时, 实验 F1 值已几乎不变, 而随着游走长度变大, 整个系统的实际运行时间会显著增加, 为了平衡二者以达到最优的实验水平, 本文选择 30 作为最佳实验参数。

此外, 在 4.4 节中提到本文通过给每个视图分配不同的权重来对向量进行融合操作, 然而如何分配权重从而能够获得最佳恶意应用检测效果则需要通过实验进一步确定。具体实验结果如表 7 所示。

**表 7　融合权重参数选择实验结果**

**Table 7　Experimental results of fusion weight parameters selection**

| 各视图权重分配 | | | ACC(%) | F1(%) |
|---|---|---|---|---|
| 可疑控制条件 $\alpha_1$ | 敏感数据流 $\alpha_2$ | 权限 $\alpha_3$ | | |
| 1 | 2 | 3 | 93.53 | 91.62 |
| 1 | 3 | 2 | 94.75 | 93.87 |
| 2 | 1 | 3 | 93.94 | 91.99 |
| 2 | 3 | 1 | **96.57** | **95.56** |
| 3 | 2 | 1 | 95.76 | 94.68 |
| 3 | 1 | 2 | 94.23 | 92.34 |
| 1 | 1 | 1 | 95.15 | 94.37 |

由实验结果可知, 在权重分配比为 2∶3∶1 时, 安卓恶意应用分类效果达到最佳。具体来说, 本实验设置了如上表所示的权重比参数, 以大致判断如何分配权重可以更好的权衡三个视图对于本文的恶意应用检测任务的重要性, 从而分配一个较为合适的权重。从实验结果来看, 给敏感数据流视图分配大权重、权限视图分配最小权重时可以获得最好的检测结果, 这也侧面说明敏感数据流视图对结果的影响是最大的, 同时, 虽然权限视图的重要性相对较小, 但其视图对于综合判断应用行为的恶意性有很好的补充作用。值得注意的是, 本实验所采用的实验方法以及权重的设置仍具有进一步改进的空间, 详见第 6 节的讨论。

## 6　讨论

本文所实现的 MVFDroid 仍存在以下局限性:

(1)不同于 HAWK[42]、AiDroid[43]等工作, 本文为充分挖掘不同检测方案的关联性和互补性, 选取的相关特征(如敏感数据流、控制敏感 API 的控制因素等), 相较于签名、Manifest 文件中的字段等特征, 其客观上的提取需要耗费较大的时间成本。因此, 在数据集体量方面小于上述的相关工作。未来拟对本文的静态提取器做进一步的优化, 加快特征提取速度, 从而帮助快速扩展数据集。

(2)本文目前实现的恶意应用检测系统是基于静态网络的, 这导致系统对于实时性较强的真实应用审计任务的实用性降低。具体来说, 静态网络对于数据集中的节点(即样本内节点)经过统一处理后生成对应的表示向量, 但之后如果数据集进行了扩充, 静态网络获取新样本(即样本外节点)的方式为重新进行一次上述步骤。这对于真实审计场景来说时间成本和算力成本消耗都很大。未来拟在 MVFDroid 的基础上, 设计并实现一个在线学习模块, 提高 MVFDroid 的实用性。

(3)为证明 MVFDroid 的有效性, 本文选取敏感数据流、可疑控制条件和权限三个视图作为观察对象。实际上, MVFDroid 可支持添加更多的视图以丰富观察视角, 提高系统检测的准确性, 并进一步提高检测准确率。例如, 安卓跨组件通信机制也是安卓的一个重要特点, 针对该机制可以实现类似于应用间的串谋攻击等隐蔽性更强的恶意行为。因此, 如何将更多的观察视图融合到目前的检测体系中, 形成更加全面的检测系统, 也是未来的一个重要改进方向。

(4)多视图融合权重的选择。目前, 本系统在多视图融合时是通过专家经验和人工实验结合, 对权重比逐步的筛选以确定各视图权重。然而, 这种方法的局限性较大, 往往也不能最为精确的确定最佳权重比。为了解决这一问题, 后续拟引入注意力机制[52]动态的分配权重, 更为精确地确定该参数。

## 7　结论

当前安卓恶意应用泛滥, 且朝着攻击行为愈发多样且隐蔽的方向发展。为了实现对安卓恶意应用的精确检测, 本文基于异构信息网络、图表示学习和多视图融合技术, 设计并实现了安卓恶意应用检测系统 MVFDroid。本系统通过对传统检测方法的检测优势进行总结与凝练, 形成了基于可疑控制条件、敏感数据流和权限的三种不同的针对安卓恶意行为的观察角度, 并通过基于多视图的表示学习方式对其中的语义信息进行深度挖掘, 形成了一种较为立体的安卓恶意应用检测方法。该系统对恶意应用检测的准确度达到 96.57%、F1 值达到 95.56%, 均优于当前的代表性检测方案 Drebin、HinDroid 和 MaMaDroid。

**致　谢**: 本文受国家自然科学基金项目(No.62102385,

No.62372422, No.62272434, No.61972369)、安徽省自然科学基金项目(No.2108085QF262) 和中央高校基本科研业务费专项资金(WK2150110024) 资助。

## 参考文献


[1] Smartphone Market Share. https://www.idc.com/promo/smartphone-market-share. Jan. 2022.

[2] 2022 年上半年度中国手机安全状况报告. https://pop.shouji.360.cn/safe_report/Mobile-Security-Report-202206.pdf. Oct. 2022.

[3] Peril in a Pandemic: The State of Mobile Application Security. https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/rp-peril-in-pandemic.pdf. Oct. 2022.

[4] Arzt S, Rasthofer S, Fritz C, et al. FlowDroid: Precise Context, Flow, Field, Object-Sensitive and Lifecycle-Aware Taint Analysis for Android Apps[C]. *The 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2014: 259-269.

[5] Octeau D, McDaniel P, Jha S, et al. Effective Inter-Component Communication Mapping in Android with Epicc: An Essential Step towards Holistic Security Analysis[C]. *The 22nd USENIX conference on Security*, 2013: 543-558.

[6] Octeau D, Luchaup D, Dering M, et al. Composite Constant Propagation: Application to Android Inter-Component Communication Analysis[C]. *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015: 77-88.

[7] Li L, Bartel A, Bissyandé T F, et al. IccTA: Detecting Inter-Component Privacy Leaks in Android Apps[C]. *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015: 280-291.

[8] Li L, Bartel A, Bissyandé T F, et al. Apkcombiner: Combining multiple android apps to support inter-app analysis[C]. *IFIP International Information Security and Privacy Conference*, 2015: 513-527.

[9] Salehnamadi N, Alshayban A, Ahmed I, et al. ER Catcher: A Static Analysis Framework for Accurate and Scalable Event-Race Detection in Android[C]. *2020 35th IEEE/ACM International Conference on Automated Software Engineering*, 2020: 324-335.

[10] Samhi J, Gao J, Daoudi N, et al. JuCify: A Step towards Android Code Unification for Enhanced Static Analysis[C]. *2022 IEEE/ACM 44th International Conference on Software Engineering*, 2022: 1232-1244.

[11] W. Enck, P. Gilbert, S. Han, et al. Taintdroid: an informationflow tracking system for realtime privacy monitoring on smartphones[C]. *ACM Transactions on Computer Systems*, 2014: 5.

[12] K. Tam, S. J. Khan, A. Fattori, et al. Copperdroid: Automatic reconstruction of android malware behaviors[C]. *NDSS*, 2015.

[13] Y. Zhang, M. Yang, B. Xu, et al. Vetting undesirable behaviors in android apps with permission use analysis[C]. *The 2013 ACM SIGSAC conference on Computer & communications security*, 2013: 611-622.

[14] Meng Z Y, Xiong Y, Huang W C, et al. AppAngio: Revealing Contextual Information of Android App Behaviors by API-Level Audit Logs[J]. *IEEE Transactions on Information Forensics and Security*, 2021, 16: 1912-1927.

[15] Tsutano Y, Bachala S, Srisa-an W, et al. Jitana: A Modern Hybrid Program Analysis Framework for Android Platforms[J]. *Journal of Computer Languages*, 2019, 52: 55-71.

[16] Alhanahnah M, Yan Q B, Bagheri H, et al. DINA: Detecting Hidden Android Inter-App Communication in Dynamic Loaded Code[J]. *IEEE Transactions on Information Forensics and Security*, 2020, 15: 2782-2797.

[17] Yang Z, Yuan Z H, Jin S Y, et al. FSAFlow: Lightweight and Fast Dynamic Path Tracking and Control for Privacy Protection on Android Using Hybrid Analysis with State-Reduction Strategy[C]. *2022 IEEE Symposium on Security and Privacy*, 2022: 2114-2129.

[18] Yang W, Xiao X S, Andow B, et al. AppContext: Differentiating Malicious and Benign Mobile App Behaviors Using Context[C]. *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015: 303-313.

[19] Xu K, Li Y J, Deng R H. ICCDetector: ICC-Based Malware Detection on Android[J]. *IEEE Transactions on Information Forensics and Security*, 2016, 11(6): 1252-1264.

[20] Li J, Sun L C, Yan Q B, et al. Significant Permission Identification for Machine-Learning-Based Android Malware Detection[J]. *IEEE Transactions on Industrial Informatics*, 2018, 14(7): 3216-3225.

[21] Arora A, Peddoju S K, Conti M. PermPair: Android Malware Detection Using Permission Pairs[J]. *IEEE Transactions on Information Forensics and Security*, 2020, 15: 1968-1982.

[22] Huang J J, Zhang X Y, Tan L, et al. AsDroid: Detecting Stealthy Behaviors in Android Applications by User Interface and Program Behavior Contradiction[C]. *The 36th International Conference on Software Engineering*, 2014: 1036-1046.

[23] Fratantonio Y, Bianchi A, Robertson W, et al. TriggerScope: Towards Detecting Logic Bombs in Android Applications[C]. *2016 IEEE Symposium on Security and Privacy*, 2016: 377-396.

[24] Yang W, Prasad M, Xie T. EnMobile: Entity-Based Characterization and Analysis of Mobile Malware[C]. *2018 IEEE/ACM 40th International Conference on Software Engineering*, 2018: 384-394.

[25] Mikolov T, Sutskever I, Chen K, et al. Distributed Representations of Words and Phrases and Their Compositionality[C]. *The 26th International Conference on Neural Information Processing Systems - Volume 2*, 2013: 3111-3119.

[26] Dong Y X, Chawla N V, Swami A. Metapath2vec: Scalable Representation Learning for Heterogeneous Networks[C]. *The 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017: 135-144.

[27] Perozzi B, Al-Rfou R, Skiena S. DeepWalk: Online Learning of Social Representations[C]. *The 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014: 701-710.

[28] Grover A, Leskovec J. Node2vec: Scalable Feature Learning for Networks[J]. *KDD*: *Proceedings International Conference on Knowledge Discovery & Data Mining*, 2016, 2016: 855-864.

[29] Arp D, Spreitzenbarth M, Hübner M, et al. Drebin: Effective and Explainable Detection of Android Malware in Your Pocket[J]. *21st Annual Network and Distributed System Security Symposium*, *NDSS 2014*, 2014: 23-26.

[30] Onwuzurike L, Mariconti E, Andriotis P, et al. MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models (Extended Version)[J]. *ACM Transactions on Privacy and Security*, 2019, 22(2): 14.

[31] Hou S, Ye Y, Song Y, et al. Hindroid: An intelligent android malware detection system based on structured heterogeneous information network[C]. *The 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017: 1507-1515.

[32] Tang J, Qu M, Wang M Z, et al. LINE: Large-Scale Information Network Embedding[C]. *The 24th International Conference on World Wide Web*, 2015: 1067-1077.

[33] Zhang D K, Yin J, Zhu X Q, et al. MetaGraph2Vec: Complex Semantic Path Augmented Heterogeneous Network Embedding[EB/OL]. 2018: 1803.02533. http://arxiv.org/abs/1803.02533v1.

[34] Hamilton W L, Ying R, Leskovec J. Inductive Representation Learning on Large Graphs[C]. *The 31st International Conference on Neural Information Processing Systems*, 2017: 1025-1035.

[35] Veličković P, Cucurull G, Casanova A, et al. Graph Attention Networks[EB/OL]. 2017: 1710.10903. http://arxiv.org/abs/1710.10903v3.

[36] Wang X, Ji H Y, Shi C, et al. Heterogeneous Graph Attention Network[C]. *WWW '19*: *The World Wide Web Conference*, 2019: 2022-2032.

[37] Xu L C, Wang J, He L F, et al. MixSp: A Framework for Embedding Heterogeneous Information Networks with Arbitrary Number of Node and Edge Types[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2021, 33(6): 2627-2639.

[38] Li J, Chen C, Tong H, et al. Multi-layered network embedding[C]. *The 2018 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics*, 2018: 684-692.

[39] Cen Y K, Zou X, Zhang J W, et al. Representation Learning for Attributed Multiplex Heterogeneous Network[C]. *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019: 1358-1368.

[40] Fan W Q, Ma Y, Li Q, et al. Graph Neural Networks for Social Recommendation[C]. *WWW '19*: *The World Wide Web Conference*, 2019: 417-426.

[41] Wang H W, Zhang F Z, Zhang M D, et al. Knowledge-Aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems[C]. *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019: 968-977.

[42] Hei Y M, Yang R Y, Peng H, et al. Hawk: Rapid Android Malware Detection through Heterogeneous Graph Attention Networks[J]. *IEEE Transactions on Neural Networks and Learning Systems*, 2024, 35(4): 4703-4717.

[43] Ye Y F, Hou S F, Chen L W, et al. Out-of-Sample Node Representation Learning for Heterogeneous Graph in Real-Time Android Malware Detection[C]. *The 28th International Joint Conference on Artificial Intelligence*, 2019: 4150-4156.

[44] Hou S, Fan Y, Ju M, et al. Disentangled representation learning in heterogeneous information network for large-scale android malware detection in the covid-19 era and beyond[C]. *35th AAAI Conference on Artificial Intelligence*, 2021.

[45] Narayanan A, Soh C, Chen L H, et al. Apk2vec: Semi-Supervised Multi-View Representation Learning for Profiling Android Applications[C]. *2018 IEEE International Conference on Data Mining*, 2018: 357-366.

[46] Kim T, Kang B, Rho M, et al. A Multimodal Deep Learning Method for Android Malware Detection Using Various Features[J]. *IEEE Transactions on Information Forensics and Security*, 2019, 14(3): 773-788.

[47] Allix K, Bissyandé T F, Klein J, et al. AndroZoo: Collecting Millions of Android Apps for the Research Community[C]. *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories*, 2016: 468-471.

[48] Hoff P D, Raftery A E, Handcock M S. Latent Space Approaches to Social Network Analysis[J]. *Journal of the American Statistical Association*, 2002, 97(460): 1090-1098.

[49] Sun Y Z, Han J W. Mining Heterogeneous Information Networks: Principles and Methodologies[J]. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 2012, 3(2): 1-159.

[50] Yan S C, Xu D, Zhang B Y, et al. Graph Embedding and Extensions: A General Framework for Dimensionality Reduction[J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007, 29(1): 40-51.

[51] Ye Y, Hou S, Chen L, et al. Icsd: An automatic system for insecure code snippet detection in stack overflow over heterogeneous information network[C]. *The 34th Annual Computer Security Applications Conference*, 2018: 542-552.

[52] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. *Advances in neural information processing systems*, 2017, 30.

**赵文翔** 于 2020 年在合肥工业大学大学工业工程专业获得学士学位。现在中国科学技术大学计算机技术专业攻读硕士学位。研究领域为安卓安全。研究兴趣包括: 静态分析、安卓恶意应用及行为检测等。Email: zhaowx98@mail.ustc.edu.cn

**孟昭逸** 于 2019 年在中国科学技术大学信息安全专业获得博士学位。现任中国科学技术大学计算机科学与技术学院博士后。研究领域为安卓安全、软件形式化验证。研究兴趣包括: 移动安全、软件分析、模型检测等。Email: mzy516@ustc.edu.cn

**熊焰** 于 1990 年在中国科学技术大学计算机系获得博士学位。现任中国科学技术大学计算机科学与技术学院教授。研究领域为计算机网络与信息安全、移动计算与移动网络、分布式处理等。研究兴趣包括: 形式化验证、移动安全、可信计算等。Email: yxiong@ustc.edu.cn

**黄文超** 于 2011 年在中国科学技术大学计算机科学与技术系获得博士学位。现任中国科学技术大学计算机科学与技术学院副教授。研究领域为移动计算、网络与系统安全自动化验证技术、安卓安全等。研究兴趣包括: 形式化验证、移动安全、可信计算等。Email: huangwc@ustc.edu.cn