

doi: 10.12052/gdutxb.230021

基于异构信息网络的Android恶意程序检测方法

殷丹丽, 凌捷

(广东工业大学 计算机学院, 广东 广州 510006)

摘要: 针对传统Android恶意程序检测方法无法解决的伪装及实时检测问题, 提出了一种基于异构信息网络的Android恶意程序检测方法。将Android实体及关系建模为异构信息网络中的节点和边, 设计了元结构注意力网络表示学习模型和增量学习模型。首先使用元结构注意力网络表示学习模型进行训练集节点嵌入, 将节点嵌入及标签输入到深度神经网络中进行训练, 再采用增量表示学习模型学习测试集节点嵌入, 使用top-k算法寻找邻居节点进行聚合, 将待检测节点输入到训练好的深度神经网络中进行检测。实验结果表明, 该方法 F_1 值为97.5%, 准确率为96.7%, 平均检测时间3.7 ms。与现有方法相比, F_1 值和准确率更高, 平均检测时间更短, 表明该方法能够有效应对Android恶意程序伪装, 可以用于实时Android恶意程序检测。

关键词: 安卓; 恶意程序检测; 异构信息网络; 元结构; 深度神经网络

中图分类号: TP309

文献标志码: A

文章编号: 1007-7162(2024)02-0056-09

Android Malware Application Detection Method Based on Heterogeneous Information Network

Yin Dan-li, Ling Jie

(School of Computer Science and Technology, Guangzhou University of Technology, Guangzhou 510006, China)

Abstract: To address the problems of camouflage and real-time detection of the traditional Android malware detection methods, a new Android malware detection method based on heterogeneous information networks is proposed. By modeling the Android entities and relationships nodes and edges, respectively, in a heterogeneous information network, two network representation learning models are designed, including the meta-structure attention network representation learning and the incremental learning models. First, the meta-structure attention network representation learning model is used for intra-sample node embedding, and the embedded nodes and labels are input to a deep neural network for training. Then, the incremental learning model is used for learning the extra-sample node embeddings. The top-k algorithm is used to aggregate neighboring nodes within the heterogeneous information network, and the embedded node to be detected is input to the trained deep neural network for detection. Experimental results show that the F_1 value of the proposed method is 97.5%, the accuracy rate is 96.7%, and the average detection time is 3.7 ms, which are better than the existing methods, demonstrating the effectiveness of the proposed method for dealing with Android malware camouflage and for real-time Android malware detection.

Key words: Android; malware detection; heterogeneous information networks; meta-structure; deep neural networks

Android作为一种用于智能手机的开源可定制系统, 占据智能手机市场的巨大份额, 由于Android操作系统的可扩展性和开放性, 吸引的不只是生产

良性应用程序的开发者, 在巨大经济利益驱动下, 也吸引了大量开发恶意程序的攻击者。用户面临各种恶意程序的威胁, 包括窃取数据、推送垃圾广告等,

收稿日期: 2023-02-15

基金项目: 广东省重点领域研发计划项目(2019B010139002); 广州市科技研发计划项目(202007010004)

作者简介: 殷丹丽(1997-), 女, 硕士研究生, 主要研究方向为网络信息安全, E-mail: 1521506972@qq.com

通信作者: 凌捷(1964-), 男, 教授, 博士, 主要研究方向为网络信息安全, E-mail: jling@gdut.edu.cn

Android恶意程序数量的爆炸式增长和结构日益复杂化亟需新的防御技术来保护用户免受新的威胁。

Android恶意程序检测方法^[1]包括基于特征的静态分析以及基于行为的动态分析,然而,静态分析^[2]无法有效检测带有代码混淆以及热加载的Android恶意程序,动态分析^[3]严重依赖于操作系统的版本和Android虚拟机的类型。为了解决这个问题,许多方法采用基于机器学习技术来进行关键特征提取,再输入分类算法进行检测^[4],然而,Android恶意程序在不断地演变伪装和攻击的类型,这些方法很难检测出这类新型伪装的Android恶意程序。图神经网络(Graph Neural Networks, GNN)可以用于模拟实体之间的关系,异构信息网络^[5]是图形神经网络的一个特例,为了有效地对异构信息网络中的节点(应用程序)进行检测,可以将网络结构转化为向量的形式,来学习网络中节点和关系的语义信息。但异构信息网络是针对静态的网络进行的,它要求所有的节点在模型训练之前都是已知标签的,然而在实际应用中,需要的是实时的Android恶意程序检测,如果每来一个新的节点(新型未知标签应用程序)都要重新构建整个模型,同时也需要对分类器进行重新训练,导致效率过低,限制了技术的实用性及规模。

针对上述问题,本文提出了基于异构信息网络的Android恶意程序检测方法,利用网络表示学习^[6]来挖掘应用程序之间隐藏的丰富语义信息。首先反编译安卓应用程序包(Android Application Package, APK)文件提取Android特征实体,通过将特征实体及关系分别转换为节点和边来建立异构信息网络,引入元结构注意力机制定义不同类型实体之间的关系序列及权重,获取异构信息网络内部节点嵌入,将节点嵌入及其标签输入深度神经网络分类模型^[7]进行训练,对于新型未知标签节点检测,使用top-k算法寻找异构信息网络内的邻居节点,将已经学到的元结构网络表示学习模型拓展到新节点上,这种设计可以基于已建立的异构信息网络结构快速计算嵌入,无需重新运行整个模型或调整整体嵌入,从而显著提高训练效率和模型的可扩展性。获取待检测应用程序节点嵌入后,通过训练好的分类模型将其预测为恶意程序或良性程序,以提高对Android恶意程序检测的精度和效率。

1 方法与原理

基于异构信息网络,本文所提出方法的框架如图1所示。该方法共包含以下几个模块。

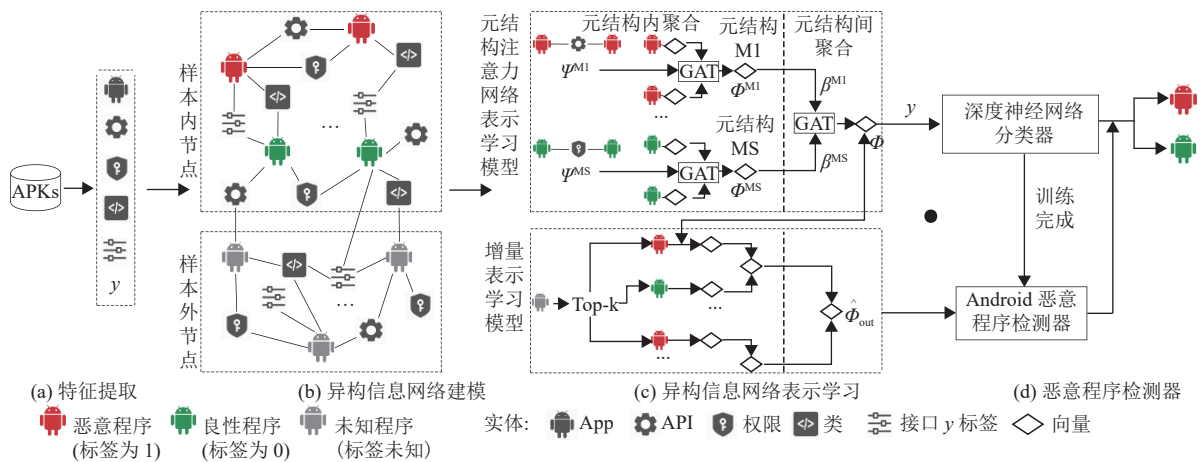


图1 基于异构信息网络的Android恶意程序检测方法框架图

Fig.1 Block diagram of the Android malicious program detection method based on heterogeneous information network

1.1 特征提取

Android应用程序在系统中会被编译打包成APK文件,使用apktool工具将数据集中APK文件反编译得到smali文件,从中进行关键特征实体提取,主要有以下5类。

APP: 每一个Android应用程序都有着唯一标识的ID;

应用程序编程接口(Application Programming

Interface, API): 调用序列可以捕捉程序行为;

权限: 规定Android应用程序可以进行的操作;

类: 描述Android应用程序的行为和状态;

接口: 描述Android应用程序要实现的功能。

进一步考虑各种关系特征,基于领域知识可以对关键特征实体进一步进行关系提取:

由于节点嵌入的基本要求是获得图的结构,因此,需要计算邻接矩阵来反映图中节点的连通性和

邻近性,使用关系矩阵分别存储每个关系,如表1所示,5个矩阵用于表示和存储关系 $R_1 \sim R_5$ 。

表1 关系矩阵定义表
Table 1 Definitions of the relational matrix

关系	矩阵	定义
R_1	A	如果APP $_i$ 有API $_j$,则 $A_{ij}=1$,否则 $A_{ij}=0$
R_2	P	如果APP $_i$ 有权限 j ,则 $P_{ij}=1$,否则 $P_{ij}=0$
R_3	C	如果APP $_i$ 有类 j ,则 $C_{ij}=1$,否则 $C_{ij}=0$
R_4	I	如果APP $_i$ 调用接口 j ,则 $I_{ij}=1$,否则 $I_{ij}=0$
R_5	M	如果类 i 实现接口 j ,则 $M_{ij}=1$,否则 $M_{ij}=0$

1.2 异构信息网络建模

为了使用提取的特征描述应用程序,且各实体类型不同,将实体及关系特征分别以节点和边的形式建模为异构信息网络,以便能够更好地处理不同类型的实体和关系,下面是一些基本定义及实例:

定义1 异构信息网络:给定节点集合 V 、关系集合 E ,节点类型集合 P 、关系类型集合 R ,异构信息网络可以表示为七元组:

$$G = (V, E, P, R, \varphi, \theta, \psi)$$

式中: $\varphi: V \times V \rightarrow R$, $\theta: V \rightarrow P$, $\psi: E \rightarrow R$ 分别为关系映射、节点类型映射和关系类型映射,并且满足 $|P| + |R| > 2$ 。

定义2 网络模式:提取异构信息网络 G 中的节点类型集合作为新的图顶点集,关系集合作为边集合,基于上述定义,构建的异构信息网络的网络模式具有5种实体类型和5种实体类型之间的关系,如图2所示,网络模式图可以表示应用程序的语义和结构信息。

定义3 元结构(MetaStructure, MS):是一种在异构信息网络中指定不同类型实体之间语义关系的方式,可以描述节点之间的复杂关系,如节点之间的相似性、传递性和路径结构,包括元路径(MetaPath, MP)或者元图(MetaGraph, MG)两种形式。其中元路径是用于描述异构信息网络中节点之间关系的一种模式。它是一种有向无环图,其中节点表示数据集中的实体(应用程序实体和特征实体),边表示它们之间的关系。元路径由一个节点序列和一个与序列中的节点类型对应的边类型序列组成,表示了从一个节点类型到另一个节点类型的路径,而元图则是由多个元路径组成的,反映了应用程序间的复杂关系。

定义4 训练集和测试集:训练集通常指的是用于训练模型的数据集。在训练模型时,使用训练集数据集来拟合模型,调整模型参数,使其能够最好地适

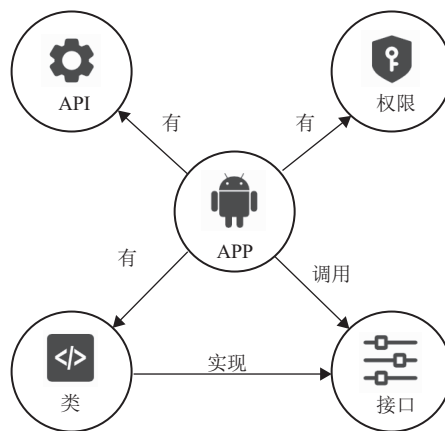


图2 网络模式图

Fig.2 Network model diagram

应训练数据。如果模型在训练集数据集上的表现非常好,可能是因为它过度拟合了训练数据,无法泛化到未见过的数据。测试集通常指的是用于测试模型性能的数据集。在训练模型后,使用测试集数据集来评估模型的泛化能力和性能。如果模型在测试集数据集上的表现与在训练集数据集上的表现类似,说明模型具有很好的泛化能力和性能。

为了方便后续执行GNN模型来进行节点嵌入,需要将异构信息网络转化成只包含应用程序节点的同构图,即将应用实体和其他实体之间的关系合并为应用实体之间的组合连接,利用给定异构信息网络中的元结构来计算应用程序之间的相似度,获得该图的节点邻接矩阵。根据系统安全专业领域知识,并结合上文所提取5类特征,本文方法设计了5个元路径 $MP_1 \sim MP_5$ 和2个元图 $MG_1 \sim MG_2$,如图3所示。 $MP_1 \sim MP_5$ 分别代表通过API、权限、类、接口、类和接口形成2个应用程序之间的路径。 MG_1 表示同时含有 MP_1 和 MP_3 两条元路径, MG_2 表示同时含有 MP_3 和 MP_5 两条元路径。通过元路径和元图的定义可以有效地表示出异构信息网络中2个Android应用程序之间的结构语义,捕捉它们之间丰富的关系。

通过对建模的关系矩阵执行一系列矩阵运算,可以精确计算出图中APP节点的邻接度,对于给定

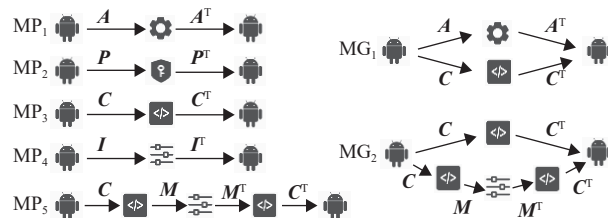


图3 元结构设计图

Fig.3 Diagram of the Metastructure

的元路径 $MP(Z_1, \dots, Z_n)$,邻接矩阵可以通过式(1)计算。

$$\Psi^{MP} = R_{Z_1 Z_2} \cdot R_{Z_2 Z_3} \cdots R_{Z_{n-1} Z_n} \quad (1)$$

式中: $R_{Z_{i-1} Z_i}$ 是实体 Z_{i-1} 和 Z_i 之间的关系矩阵, $i \in [2, n]$ 。例如, MP_1 的邻接矩阵是 $\Psi^{MP_1} = A \cdot A^T$, $\Psi_{ij} > 0$ 表示应用程序 i 与应用程序 j 有关联,数值代表应用程序 i 与应用程序 j 间元路径的数量,对于元图,多个元路径的组合($MP_1, \dots, MP_m$),邻接矩阵计算公式:

$$\Psi^{MG} = \Psi^{MP_1} \odot \dots \odot \Psi^{MP_m} \quad (2)$$

式中: $\odot$ 为哈达玛积,例如 MG_2 的邻接矩阵是 $\Psi^{MG_2} = (C \cdot C^T) \odot (C \cdot M \cdot M^T \cdot C^T)$ 。

通过对每个元结构进行图形建模,原始异构信息网络被转换为多个APP同构图,每个图都属于一个邻接矩阵,给定 Q 个元结构 $\{M_1, \dots, M_Q\}$,则共有 Q 个邻接矩阵的集合,即 $\{\Psi^{M_1}, \dots, \Psi^{M_Q}\}$ 。

1.3 异构信息网络表示学习

传统意义上的网络表示学习是为了将网络中的节点表示为低维稠密的向量形式,分解网络结构转化为邻接矩阵,使得节点能够在向量空间具有推理表达能力,同时也方便作为下游分类任务的输入。然而大规模矩阵的计算成本十分昂贵,传统网络表示学习效率低下,目前的异构信息网络嵌入模型可以挖掘应用程序之间的潜在语义,但没有考虑到每当有新节点需要检测时,需要重新运行整体嵌入,并且下游分类任务也需要重新进行训练,针对实时的Android恶意程序检测效果不佳。为了有效地学习异构信息网络中测试集节点的表示,即节点嵌入后到达的节点,提出针对异构信息网络的可扩展学习方法,面向训练集和测试集节点,设计了元结构注意力网络表示和增量表示2种学习模型^[8]。

1.3.1 元结构注意力网络表示学习模型

元结构注意力网络表示学习模型,指的是使用注意力机制来学习不同层次的元结构,将输入的数据(特征向量)转换为低维度的向量表示,以便于计算机对其进行处理和理解。这些向量被称为“表示”或“嵌入”。

对于训练集节点,通过在元结构内部和元结构之间引入2层注意力机制,分别是节点级注意力和语义级注意力,对于给定元结构,通过节点级注意力将元结构内部邻居节点聚合于中心节点,得到每个元结构的特征向量,然后再通过语义级注意力将这些特征向量进行聚合,得到最终的特征向量。

元结构内部节点聚合主要考虑节点自身的特征信息和节点之间的边信息。在同一个元结构上,一个

节点的邻居节点有不同的重要性,并且这些邻居节点的类别也有多种。为了聚合这些邻居节点,对每个训练集应用程序进行 one-hot 编码,形成矩阵 H ,其中矩阵的第 i 行代表第 i 个应用程序的特征向量 h_i ,使用边缘权重感知的 GAT 模型,计算每个节点的权重。对于给定元结构 $M_s (s \in [1, Q])$,其中一对相连节点 i, j 的注意力值计算方式为

$$e_{ij}^{M_s} = \sigma(a_{M_s}^T \cdot [h_i \| h_j]) \quad (3)$$

式中: σ 为sigmoid激活函数, $\|$ 为拼接操作; a_{M_s} 为节点级注意力向量。然后通过softmax操作得到每个特征向量的权值系数:

$$a_{ij}^{M_s} = \text{softmax}(e_{ij}^{M_s}) \quad (4)$$

最终,在这一阶段可以获得所有训练集节点的向量嵌入集合,即 $\{\Phi^{M_1}, \dots, \Phi^{M_Q}\}$,计算方式如式(5)所示。

$$\Phi^{M_s} = \sigma\left(\sum_{j \in N_i^{M_s}} a_{ij}^{M_s} \cdot h_j\right) \quad (5)$$

式中: $N_i^{M_s}$ 为节点 i 在元结构 M_s 上的邻居节点。

每个元结构都是一个单独的语义视图,通过语义级注意力整合不同语义下的嵌入,提高节点嵌入的质量,不同元结构的权重 β^{M_s} 计算方式如式(6)所示。

$$\beta^{M_s} = \text{softmax}(L(\Phi^{M_s})) \quad (6)$$

式中: L 为可以将矩阵映射为数值的神经网络。最终通过矩阵加权,可以获得所有训练集节点的整体嵌入:

$$\Phi = \sum_{s=1}^Q \beta^{M_s} \cdot \Phi^{M_s} \quad (7)$$

1.3.2 增量表示学习模型

为了更高效地嵌入测试集节点,即未包含在训练过程中的未知应用程序,提出增量表示学习模型,准确定位测试集节点与训练集节点之间的潜在连接,通过元结构注意力网络表示学习模型学习到的训练集节点整体嵌入快速表示测试集节点,计算新应用程序节点的关系矩阵 $R_{u_{out}}$; $u \in [1, 5]$,形成节点邻接矩阵的增量段 $\hat{\Psi} = \sum_{u=1}^5 R_u \cdot R_{u_{out}}^T$,形式为 w 行列矩阵, w 表示训练集节点的个数,矩阵的第 w 行数值 $\hat{\Psi}_{v_w}$ 代表测试集节点与训练集节点 v_w 之间元结构的数量, R_u 是训练集节点的关系矩阵,使用top-k算法对 $\hat{\Psi}_{v_w}$ 进行排序,选出数值较大的训练集前 K 个节点 $(v_1, \dots, v_w, \dots, v_K)$, $w \in [1, K]$,得到新应用程序节点嵌入,聚合测试集节点与训练集邻居节点的向量,得到测试集节点嵌入:

$$\hat{\Phi}_{out} = \sum_{s=1}^Q \beta^{M_s} \cdot \sum_{w=1}^K \frac{\hat{\Psi}_{v_w}}{\sum_{w=1}^K \hat{\Psi}_{v_w}} \cdot \Phi^{M_s} \quad (8)$$

基于训练集节点所提出的元结构注意力网络表示学习模型获得训练集节点嵌入,当新节点(例如测试应用程序)到达时,不需要重新运行整体嵌入,这使得下游分类器可用于对新到达节点进行分类而无需重新训练。

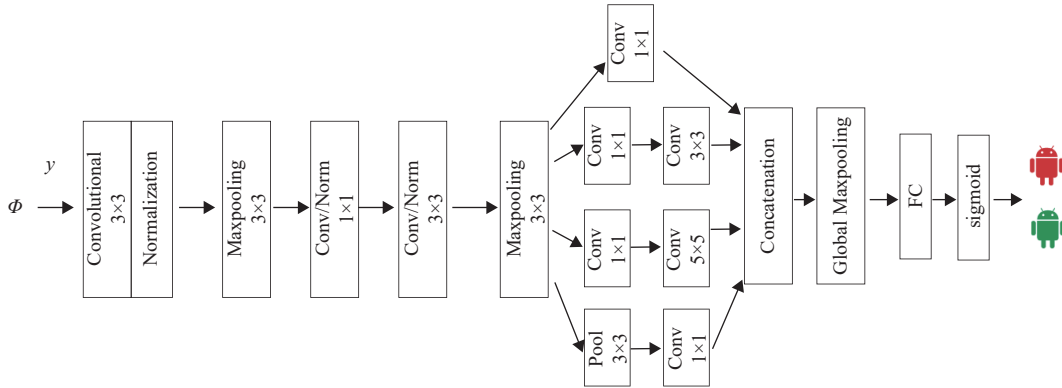


图4 深度神经网络分类器

Fig.4 Deep neural network classifier

在本文设计的深度神经网络中:

- (1) 训练,将训练集节点嵌入 Φ 以及标签 y 作为输入训练一个有监督的深度神经网络;
- (2) 预测,测试集节点将首先通过使用增量表示学习模型嵌入来实时获得其表示,然后由训练模型预测为恶意或良性。

构建异构信息网络的过程,用伪代码描述如算法1所示:

算法1 异构信息网络的构建过程

- 1) 输入:数据集 D
- 2) 数据预处理:
 - (a) 加载数据集 $G=\text{networks.read_edgelist}(D)$;
 - (b) 对数据集进行预处理,并提取每个应用程序节点的特征。
- 3) 构建节点和边的集合:
 - (a) 定义节点类型:


```
node_type=['APP','API','Permission',
            'Class','Interface'];
```
 - (b) 定义边的类型:


```
edge_type=['R1','R2','R3','R4','R5'];
```
 - (c) 创建节点nodes和边edges,构建节点之间的连接关系;
 - (d) 为每个节点和边添加属性,并添加到网络 $G=\text{heterograph}(\text{nodes},\text{edges})$ 中。
- 4) 输出:构建好的异构信息网络 G 。

本文方法的系统实现过程用伪代码描述如算法2所示。

1.4 深度神经网络分类器

深度神经网络可学习分类任务的特征,在保持计算量不变的情况下,通过提高网络的深度和广度优化性能学习到更多的特征,从而提升训练结果。因此,本文方法利用深度神经网络的优势,针对实时Android恶意软件检测设计了深度神经网络分类器,如图4所示。

算法2 基于异构信息网络Android恶意程序检测方法

- 1) 输入:异构信息网络 G ,网络模式 S ,训练集 D_1 及训练集标签 y ,测试集 D_0 ;
- 2) 获得训练集 D_1 嵌入 Φ ;
 - (a) 用式(4)计算得到 $\{\alpha_{v_1}^{M_1}, \dots, \alpha_{v_w}^{M_s}\}$;
 - (b) 用式(5)计算得到 Φ^{M_s} ;
 - (c) 用式(6)计算得到 β^{M_s} ;
 - (d) 用式(7)计算得到计算 Φ
- 3) 用式(8)计算获得测试集 D_0 嵌入 $\hat{\Phi}_{out}$;
- 4) 输入 Φ 和 y 训练深度神经网络分类器;
- 5) 输入 $\hat{\Phi}_{out}$ 到训练完成的深度神经网络分类器;
- 6) 输出:测试集 D_0 的标签 $\hat{y}$ 。

2 实验与评价指标

2.1 实验环境

2.1.1 软硬件配置

实验环境是4个节点的NVIDIA Tesla A100 GPU集群,具体配置如表2所示,使用ApkTool解析Android应用程序。

2.1.2 数据集

本实验数据集由两部分组成,分别为恶意应用数据集和良性应用数据集,如表3所示。

- (1) 恶意应用数据集:实验使用的恶意应用数据集来源于加拿大网络安全研究所的公开数据集

表 2 实验环境与配置
Table 2 Experimental environment and configuration

环境	配置
操作系统	Ubuntu 18.04.5
CPU	Intel(R) Xeon(R) Silver 4 114@ 2.20 GHz
显卡型号	NVIDIA Tesla A100集群
内存	256 G
显存	16 G × 4
开发工具	PyCharm
开发语言	Python 3.7.11

表 3 数据集的组成、编号、种类和数量
Table 3 The composition, id, types and number of datasets

组成	编号	种类	数量/个
恶意应用数据集	49-1	Adware	1 253
	60-1	Banking Malware	2 100
	17-1	SMS Malware	3 904
	36-1	Mobile Riskware	2 546
	39-1	Benign	1 795
良性应用数据集	HC-7-1	Popular	1 340
	HC-4-1	Communication	1 512
	HC-3-1	SNS	1 021
	HC-6-1	EAS	980
	HC-2-1	OA	850
	HC-5-1	Entertainment	9 815

CICMalDroid。表3为从该数据集中选择的恶意应用的种类和数量。CICMalDroid数据集收集了包含广告软件、短信恶意软件、银行恶意软件、风险软件和常规软件5种不同类别共计11 598个应用程序包,如表3所示。为了生成测试集数据,从VirusShare收集4个恶意软件子集(v2018至v2021),每个子集大约5 000个样本。

(2) 良性应用数据集: 良性应用程序来源于 Google Play应用商店, 主要包含流行类、通讯类、社交网络类、教育类、办公类、娱乐类, 共计15 518个APK文件, 如表4所示。同时提取了相同数量的良性应用程序作为测试集, 以匹配测试集。

2.1.3 特征工程

实验总体反编译了27 116个APK, 从中提取特征组成的异构信息网络共63 902个实体, 包含API、权限、类和接口共4类, 其中一部分具体特征信息如表4所示。

2.2 实验评价指标

由于本文提出的恶意程序检测方法的目的是检测出恶意应用样本, 因此将恶意程序样本作为正样

表 4 具体API、权限、类、接口特征信息
Table 4 Specific API, Permissions, Classes, Interfaces

特征	功能	实体名
API	发送短信和拨打电话	SmsManager.sendMessage()
	文件读写	FileWriter()
	网络通信	URLConnection()
	定位	Geocoder.getLocation()
	系统操作	ActivityManager.getRunningTasks()
权限	发送短信	SEND_SMS
	读取联系人列表	READ_CONTACTS
	获取GPS定位信息	ACCESS_FINE_LOCATION
	录制音频	RECORD_AUDIO
类	使用相机拍照或录像	CAMERA
	网络通信	Service.onBind()
	广播消息接收	BroadcastReceiver.onReceive()
	管理数据	ContentProvider.insert()
	用户界面	startActivity()
	加载指定的URL地址	WebView.loadUrl()
接口	获取电话信息	TelephonyManager()
	获取定位信息	LocationManager()
	调用摄像头	Camera.open();
	管理蓝牙连接	getBondedDevices()
	录制音视频	startRecording()

本, 良性程序作为负样本, 定义实验评价指标如表5所示。

表 5 实验评价指标
Table 5 Experimental evaluation metrics

真实	预测	
	恶意程序样本	良性程序样本
恶意程序样本	真阳性 (TP)	假阴性 (FN)
良性程序样本	假阳性 (FP)	真阴性 (TN)

准确率(Accuracy): 指被正确分类的样本数目占测试集所有样本数目的比例, 计算公式为

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

精确率(Precision): 指在被分类的恶意应用的样本中, 被正确分类的恶意应用所占比例, 计算公式为

$$Precision = \frac{TP}{TP + FP}$$

召回率(Recall): 指在所有恶意应用样本中, 被正确检测为恶意应用样本所占比例, 计算公式为

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

F₁值(F₁-score): 综合考虑了精确率和召回率, 是对精确率和召回率的加权调和, 计算公式为

$$F_1\text{-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.3 实验对比模型

为了评估本文方法在异构信息网络中的性能表现,通过查阅相关文献,采用近几年安卓恶意程序检测领域与本文研究方法和实验条件相似的最新研究成果作为实验对比方法,这些方法已经被广泛使用并且已经被证明是有效的,因此可以作为比较的基准,具体如下。

Node2vec^[9]:是一种基于图嵌入的方法,能够在保留图形结构的同时,将图形中的节点映射到低维向量空间中,从而减少计算和存储成本,但参数设置不合理时会影响模型效果。

Metapath2Vec^[10]:是一种基于元路径的图嵌入算法,能够捕获节点的语义信息从而提高恶意程序检测的准确性,但难以处理动态图数据。

GCN^[11]:是一种图卷积神经网络模型,能够捕获全局信息,提高恶意程序检测的准确性,具有一定可扩展性,但可能存在过拟合,难以处理大规模图数据。

GAT^[12]:是一种基于注意力机制的图神经网络,能够捕获局部信息,自适应学习不同节点权重,可以处理异质性图数据,例如安卓恶意程序检测中多种类型的节点和边,但对于稀疏图数据效果不佳。

RS-GCN:是一种在GCN模型基础上将异构信息网络转化为同构图的方法。

RS-GAT:是一种在GAT模型基础上将异构信息网络转为同构图的方法。

Drebin^[13]:是一种基于特征工程的安卓恶意程序检测方法,能够对已知的恶意程序进行高效的检测和分类,可以在不访问源代码的情况下进行检测,但在检测未知的恶意程序方面表现不佳,无法捕捉动态行为。

HinDroid^[14]:是一种基于图神经网络的安卓恶意程序检测方法,能够自动学习特征之间的关系和重要性,同时可以处理异构图的数据结构,能够检测新型恶意程序,但需要大量的计算资源和时间,检测时需要运行在安卓设备或模拟器上,存在安全风险和资源开销。

DroidEvolver^[15]:是一种基于遗传算法的安卓恶意程序检测方法,使用遗传算法进行特征选择和优化,能够有效检测恶意程序,其检测准确率和召回率较高,但需要大量的计算资源和时间来进行特征选择和优化,需要耗费较长的时间。

3 实验结果分析

3.1 实验效果

(1) 训练集检测效果:实验选择20%,40%,60%,80%的样本应用程序来训练模型,并对剩余进行测试,实验模型数据为各个模型在跑样本应用程序的实验结果。表6和表7分别表示各对比模型与本文异构信息网络模型的 F_1 值和ACC分数。一般来说元结构注意力网络表示学习模型能实现与Drebin、HinDroid和DroidEvolver等主流恶意软件检测器相近的准确性,保留了异构网络中的语义元结构信息,从而使得RS-GCN和RS-GAT的准确率比原生GCN和GAT提高了5%以上。

表 6 训练集恶意程序检测 F_1 值
Table 6 F_1 -Score of malware application detection in training samples

方法	训练比例			
	20%	40%	60%	80%
Node2Vec	0.841 2	0.837 6	0.854 4	0.860 2
Metapath2Vec	0.923 2	0.932 2	0.932 6	0.939 6
GCN	0.865 2	0.867 6	0.872 2	0.876 2
GAT	0.843 6	0.863 4	0.875 4	0.880 2
RS-GCN	0.921 4	0.951 2	0.951 6	0.956 2
RS-GAT	0.950 6	0.963 2	0.965 4	0.966 2
Drebin	0.931 4	0.938 6	0.944 6	0.947 6
HinDroid	0.964 2	0.966 8	0.968 2	0.974 6
DroidEvolver	0.941 4	0.951 6	0.946 4	0.950 8
本文方法	0.965 6	0.972 0	0.973 2	0.975 7

表 7 训练集恶意程序检测的准确率
Table 7 Accuracy of malware application detection in training samples

方法	训练比例			
	20%	40%	60%	80%
Node2Vec	0.825 2	0.838 6	0.840 3	0.859 1
Metapath2Vec	0.919 8	0.932 2	0.933 6	0.938 6
GCN	0.855 6	0.866 2	0.863 1	0.869 0
GAT	0.846 2	0.864 4	0.876 0	0.883 2
RS-GCN	0.919 8	0.949 0	0.952 7	0.954 6
RS-GAT	0.947 6	0.961 8	0.965 4	0.969 8
Drebin	0.929 4	0.935 4	0.940 6	0.944 5
HinDroid	0.949 0	0.949 1	0.952 0	0.966 2
DroidEvolver	0.932 8	0.950 4	0.955 6	0.962 4
本文方法	0.947 9	0.955 9	0.961 9	0.967 2

(2) 测试集恶意程序检测效果:表8和表9显示了当采用不同的方法时的 F_1 值和假阳性率。

表 8 测试集恶意程序检测的 F_1 值
Table 8 F_1 -score detected by out-of-sample malware applications

方法	F_1 值			
	v2018	v2019	v2020	v2021
Node2Vec	0.810 4	0.795 8	0.812 6	0.823 6
Metapath2Vec	0.932 2	0.919 7	0.921 5	0.934 7
GCN	0.870 6	0.836 8	0.849 8	0.869 4
GAT	0.843 4	0.863 1	0.854 3	0.865 1
RS-GCN	0.954 5	0.947 8	0.947 6	0.965 4
RS-GAT	0.934 3	0.942 5	0.956 3	0.943 2
Drebin	0.772 6	0.785 5	0.827 8	0.943 1
HinDroid	0.923 4	0.929 7	0.927 5	0.923 4
DroidEvolver	0.845 6	0.854 3	0.896 4	0.965 4
本文方法	0.980 5	0.973 8	0.967 3	0.967 8

表 9 测试集恶意程序检测的FP值
Table 9 FP values detected by out-of-sample malware applications

方法	FP值			
	v2018	v2019	v2020	v2021
Node2Vec	0.084 6	0.819 2	0.078 2	0.077 4
Metapath2Vec	0.041 8	0.057 6	0.059 2	0.065 6
GCN	0.035 8	0.042 8	0.041 4	0.036 6
GAT	0.040 8	0.038 6	0.040 2	0.033 5
RS-GCN	0.011 5	0.013 2	0.012 6	0.008 6
RS-GAT	0.017 3	0.014 5	0.010 6	0.012 6
Drebin	0.058 3	0.054 6	0.044 2	0.014 2
HinDroid	0.017 6	0.025 2	0.015 6	0.006 2
DroidEvolver	0.039 2	0.037 6	0.025 5	0.010 8
本文方法	0.004 9	0.006 8	0.007 8	0.008 1

如实验结果所示,本文提出的异构信息网络的检测方案表现出了最佳的检测效果。重新训练有着不可忽略的时间开销,相比较之下本文方案是一个折中但有竞争力的解决方案。RS-GAT、RS-GCN的检测能力优于Metapath2Vec和GAT、GCN,这种性能的提升证明了在嵌入模型中丰富的语义元结构能带来更强的迁移能力。Hindroid、Drebin在不同数据集上表现的效果并不稳定,说明其泛化迁移能力有限,可能该系列模型的训练依赖大型训练样本,因此在某些特定的数据集上精度较低。DroidEvolver基于特征工程根据测试集在线更新模型从而提高分类精度,然而与语义丰富的方法相比,单纯依赖显性特征在本质上是存在缺陷的。考虑到市场上应用程序的类型和数量的爆炸式增长,故很难仅仅通过显式特征提取将传统的机器学习模型应用到可靠的恶意软件检

测中。相比之下,本文提出的检测模型能够挖掘应用程序之间的高阶关系,因此具有很强的通用性,即无论数据集的类型和大小都具有很高的效率。

3.2 检测效率

在本实验中,将基于异构信息网络检测模型的时间效率与其他可统计模型进行比较。重新计算Drebin、DroidEvolver、Metapath2Vec与异构信息网络的应用检测执行时间。为了简单起见,实验中在计算整体时间时,排除了应用元数据提取时间,只计算APP检测时所消耗的时间,因为实验中所有方法都使用的相同特征提取过程。

如图 5 所示,Drebin、DroidEvolver、Metapath2Vec方案平均检测每个应用程序的时间分别为7.7,18.1,89.6 ms,本文方法的检测时间比其他方案短得多,平均3.7 ms就能检测出一个样本应用程序,这毫秒级的检测能力说明了其在大规模实时恶意软件检测场景中的适用性,加速主要来自于增量学习的方案设计,可以充分利用以前的学习信息,而无需重新运行整个模型。此外,本文方法只需选择固定数量的邻居节点来重新校准嵌入,因此时间消耗仅随测试集的数量线性增加。

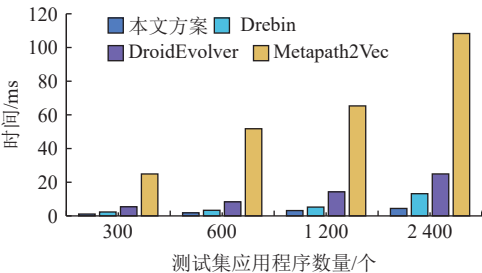


图 5 测试集APP检测时间消耗情况
Fig.5 Consumption of the detection time of out-of-sample APP

4 结语

面对无处不在的Android应用程序和日益复杂的恶意软件,本文提出了一种基于异构信息网络的Android恶意应用程序检测框架,用于快速、渐进地学习和识别新的恶意Android应用程序。异构信息网络将嵌入模型与图注意力网络相结合,从而使得分类器能够较容易识别恶意Android应用程序。利用网络表示学习和异构信息网络挖掘Android应用程序之间隐藏的丰富语义信息,通过将特征实体以及关系转化为节点和边来建立的异构信息网络,捕捉实体之间的隐式高阶关系。提出并设计了元结构注意力网络表示和增量表示2种学习模型,以融合邻居节

点嵌入任意元结构,并精确定位新应用程序和现有应用程序之间的接近度,通过增量表示学习模型,异构信息网络可以有效对新的Android应用程序执行恶意程序检测。实验表明,本文方法在准确性和时间效率方面优于其他模型。未来,我们计划将异构信息网络与移动智能设备集成起来,在支撑模型衰减的情况下,提出更先进的恶意程序检测机制。

参考文献:

- [1] ZHANG G, LI Y, BAO X, *et al.* TSDroid: a novel android malware detection framework based on temporal & spatial metrics in IoMT [J]. *ACM Transactions on Sensor Networks*, 2023, 19(3): 1-23.
- [2] AMI A S, KAFLE K, MORAN K, *et al.* Systematic mutation-based evaluation of the soundness of security-focused android static analysis techniques [J]. *ACM Transactions on Privacy and Security*, 2021, 24(3): 1-37.
- [3] HU C C, JENG T H, Chen Y M. Dynamic android malware analysis with de-identification of personal identifiable information[C]//2020 the 3rd International Conference on Computing and Big Data(ICCBD) . New York: Association for Computing Machinery, 2020: 30-36.
- [4] 郑珏, 欧毓毅. 基于卷积神经网络与多特征融合恶意代码分类方法[J]. *计算机应用研究*, 2022, 39(1): 240-244.
ZHENG J, OU Y Y. Malware classification method based on convolutional neural network and multi-feature fusion [J]. *Application Research of Computers*, 2022, 39(1): 240-244.
- [5] REN X X, ZHAO L R, WANG K, *et al.* Android malware detection based on heterogeneous information network with cross-layer features[C]// 2022 19th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP) . Piscataway: IEEE, 2022: 1-4.
- [6] LI B T, PI D. Network representation learning: a systematic literature review [J]. *Neural Computing and Applications*, 2020, 32(21): 16647-16679.
- [7] 黄剑航, 王振友. 基于特征融合的深度学习目标检测算法研究[J]. *广东工业大学学报*, 2021, 38(4): 52-58.
HUANG J H, WANG Z Y. A research on deep learning object detection algorithm based on feature fusion [J]. *Journal of Guangdong University of Technology*, 2021, 38(4): 52-58.
- [8] 凌捷, 殷丹丽, 罗玉. 一种基于异构图注意力网络的安卓恶意程序检测方法和装置: CN202210983464.3[P]. 2022-8-14.
- [9] GROVER A, LESKOVEC J. node2vec: scalable feature learning for networks[C]//Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD) . New York: Association for Computing Machinery, 2016: 855-864.
- [10] ZHANG D, YIN J, ZHU X, *et al.* Metagraph2vec: complex semantic path augmented heterogeneous network embedding[C]//Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference(PAKDD) . Berlin: Springer, 2018: 196-208.
- [11] ROMERO H J, LI C, WANG P, *et al.* ACE-GCN: a fast data-driven FPGA accelerator for GCN embedding [J]. *ACM Transactions on Reconfigurable Technology and Systems*, 2021, 14(4): 1-23.
- [12] WANG X, JI H Y, SHI C, *et al.* Heterogeneous graph attention network[C]//The 2019 World Wide Web Conference (WWW) . New York: Association for Computing Machinery, 2019: 2022-2032.
- [13] DAOUDI N, ALLIX K, BISSYANDÉ T F, *et al.* A deep dive inside drebin: an explorative analysis beyond android malware detection scores [J]. *ACM Transactions on Privacy and Security*, 2022, 25(2): 1-28.
- [14] WANG S, PHILIP S Y. Heterogeneous graph matching networks: Application to unknown malware detection[C]//2019 IEEE International Conference on Big Data. California: IEEE Computer Society Press, 2019: 5401-5408.
- [15] XU K, LI Y, DENG R, *et al.* Droidevolver: self-evolving android malware detection system[C]//2019 IEEE European Symposium on Security and Privacy. California: IEEE Computer Society Press, 2019: 47-62.

(责任编辑: 杨耀辉 英文审核: 费伦科)