基于静态特征组合的图神经网络 Android 恶意软件检测方法

韦昊东[1,2]，万良[1,2]

（1. 贵州大学公共大数据国家重点实验室，贵州 贵阳 550025；2. 贵州大学计算机科学与技术学院，贵州 贵阳 550025）

摘　要： 安卓（Android）是目前移动智能终端使用最广泛的操作系统，但层出不穷的 Android 恶意软件给用户带来重大威胁。一些方法对静态分析提取的特征进行处理，以实现 Android 恶意软件检测，这些方法能够反映软件的一部分属性，但无法捕捉软件潜在恶意行为意图的特征，使得在面对具备逃避能力的 Android 恶意软件时难以取得良好的检测性能。为解决该问题，提出一种基于静态特征组合的图神经网络 Android 恶意软件检测方法。从反编译文件中提取函数调用图，采用 node2vec 构建每个节点的局部结构特征，同时分析每个节点函数，提取操作码并进行分类，使用 Katz 算法计算节点重要程度，并根据 TF-IDF 算法计算图中每个应用程序接口（API）节点对于该 Android 恶意软件以及所属恶意家族的重要系数，将这些特征相结合作为节点特征，对重要节点进行特征自环，以增强节点间的特征差异。在此基础上，设计基于有向图神经网络（DGCN）与图注意力网络（GAT）的分类器 DAg_MAL，该分类器采用 gPool 层，能有效捕获软件行为的关键调用关系，并筛除不重要的节点。实验结果表明，该方法在二分类与多分类任务中都取得了良好的性能表现，总体检测性能优于其他同类方法。

关键词： 安卓；恶意软件检测；静态分析；图神经网络；特征嵌入；有向图

中图分类号： TP391　　**文献标志码：** A　　**DOI：** 10.19678/j.issn.1000-3428.0070073

Android Malware Detection Method Based on Static Feature Combination in Graph Neural Networks

WEI Haodong[1,2], WAN Liang[1,2]

(1. State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, Guizhou, China;
2. College of Computer Science and Technology, Guizhou University, Guiyang 550025, Guizhou, China)

【Abstract】 Android is currently the most widely used operating system for mobile smart terminals; however, the constant emergence of Android malware poses a significant threat to users. Some methods process the features extracted from static analysis to detect Android malware. These methods can reflect some attributes of the software but cannot capture the characteristics of the potential intentions behind malicious behavior; therefore, achieving good detection performance when facing Android malware with evasion capabilities is a challenge. To address this issue, this study proposes an Android malware detection method based on static feature combination in Graph Neural Network (GNN). The function call graph is extracted from the decompiled file. node2vec is used to construct the local structural features of each node, the functions of each node are analyzed, opcodes are extracted and classified, the Katz algorithm is used to calculate node importance, and the importance coefficient of each Application Program Interface (API) node in the graph is calculated for the Android malware and its malicious family according to the TF-IDF algorithm. These features are combined into node features, and feature self-looping is performed on important nodes to enhance the feature differences between nodes. On this basis, a classifier, DAg_MAL, based on a Directed GNN (DGCN) and Graph Attention Network (GAT) is designed. The classifier adopts a gPool layer, which can effectively capture the key call relationships in software behavior and exclude unimportant nodes. Experimental results show that the proposed method achieves good performance in both binary and multi-classification tasks, outperforming other similar methods.

【Key words】 Android; malware detection; static analysis; Graph Neural Network (GNN); feature embedding; digraph

0　引言

智能手机已被广泛应用于人们的日常生活中，但 Android 恶意软件对其安全构成了严重威胁。360 公司发布的移动端威胁报告[1]中指出，在 2023 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 1 699.4 万个，同比 2022 年上半年度（1 079.7 万个）上升了 57.4%，平均每天截获新

基金项目： 国家自然科学基金地区科学基金项目（62262004）。

作者简介： 韦昊东，男，硕士研究生，主研方向为人工智能、信息安全；万良（通信作者），教授、博士、博士生导师。

收稿日期： 2024-07-03　　**修回日期：** 2024-09-21　**E-mail：** lwan@gzu.edu.cn

计算机工程 www.ccied06.com

增手机恶意程序样本约 9.4 万个。大量新的恶意软件涌入市场,有效的检测技术对防御恶意软件威胁至关重要。

Android 恶意软件检测方法根据是否运行软件分为动态分析[2]与静态分析两类。动态分析通过在控制环境(如虚拟机、沙箱)中运行软件,细致观察软件行为和执行流程来判断潜在恶意行为。但该方法无法在短时间内捕获全部动态信息,从而存在漏判或误判,同时,其需要较高资源,提高了检测的成本和复杂性。静态分析无须运行应用程序,通过直接分析源代码或二进制代码,寻找安全隐患和恶意行为模式。尽管静态分析的操作简便,但其依赖代码完整性,难以解析加密或混淆的代码,无法预测程序动态行为,可能导致漏检和误报。随着 Android 系统的不断升级,应用程序接口(API)和权限的数量在不断增加。Android 恶意软件开发者的恶意行为隐蔽能力逐步提升,现有检测模型在面对众多静态特征时,可能会导致特征维度过大、模型复杂度提高、准确率下降等问题。

针对上述问题,本文提出一种节点特征构建的 Android 恶意软件检测方法。通过静态反编译 APK,构建函数调用图,设计算法并用 API、操作码及权限为图中节点构建特征。函数节点不仅包含自身特征信息,还整合了函数调用图中的空间关系,并映射出软件的行为调用流程,这些连接关系直接揭示软件的潜在恶意行为意图。该方法使得相同 API 节点在不同软件中即便包含相同的静态特征信息,空间信息特征也不相同。此外,本文结合有向图卷积网络(DGCN)与图注意力网络(GAT)构建检测模型 DAg_MAL。引入 gPool 层计算节点重要分数,以减少函数调用图的节点数量。

1 相关工作

1.1 针对静态特征的相关工作

APK 分析过程只关注 2 个文件,即 AndroidManifest.xml 文件与源代码文件(class.dex)[3-4]。AndroidManifest.xml 记录了应用的权限、硬件与软件组件等详细信息。源代码文件则包含全部编程代码,涵盖 API 信息、意图和系统事件等。该部分主要关注静态分析得到的权限、操作码和 API 这 3 类特征。

1.1.1 权限

权限是应用软件为了执行特定操作或访问系统资源而需要获得的用户授权。例如,访问用户的精确位置、拍照、使用网络或读取外部存储等都需要相应的权限。这些权限在应用的 AndroidManifest.xml 文件中声明,并在安装或运行时向用户请求。用户需要明确同意这些权限请求,才能确保应用程序的安全和正常运行[5-6]。文献[7]介绍了名为 Kirin 的系统,该系统通过审查应用权限以确保其符合高级安全策略,帮助用户评估新安装应用的安全性。文献[8]指出,细粒度权限可能提高用户判断权限危险性的难度,但通过结合 Kirin 的高级安全策略可以缓解这种副作用。文献[8]还开发了一种警报系统,考虑了应用的请求权限、类别及其在同类应用中的常见权限情况。

1.1.2 操作码

操作码是从 APK 的二进制文件中提取的,这些二进制文件包含多个 smali 文件,每个 smali 文件代表一个应用程序的类文件,由一系列操作码序列构成。文献[9]从操作码序列中提取 n-gram 特征并用机器学习模型进行恶意软件分类。文献[10]同样使用 n-gram 特征,但注重频率来训练机器学习模型,其最佳曲线下面积(AUC)值仅为 72.7%。文献[11]通过长短期记忆网络学习操作码序列的上下文语义来识别恶意软件。文献[12]提出了一种将 Android 应用的操作码序列映射为图片颜色矩阵的方法,将操作码转换为 RGB 值,利用卷积神经网络(CNN)模型分析这些图像以识别恶意行为。

1.1.3 API

API 允许应用访问系统服务或功能。API 调用通常通过调用系统库中的函数或方法来实现,这些函数或方法由操作码序列在 APK 的二进制文件中定义和执行,同时需要用户授权相应的权限。文献[13]开发了 HomDroid 系统,使用社交网络的同质性原理深入分析恶意软件 API 调用图中的关联。该系统区分包含敏感 API 和非敏感 API 的社区,通过分析敏感社区与良性社区之间的关系,有效检测隐蔽恶意软件。文献[14]提出 SDAC 方法,利用 FlowDroid 提取 API 调用路径,并通过滑动窗口技术生成 API 语义距离特征集,用于高效检测恶意 Android 应用。

1.2 深度学习检测方法

使用深度学习检测 Android 恶意软件成为目前常用的方法。文献[15]提出一个两阶段框架,将 Android APK 转换为二进制灰度图像,这些图像用作传统 CNN 模型的输入,然后将迁移学习方法用于训练模型,通过冻结预训练模型的初始层来克服过拟合、复杂性和计算成本的问题。文献[16]提出 DL-AMDet 双层架构,首先利用静态分析(如权限

计算机工程 www.ecice06.com

和 API 调用)结合 CNN-BiLSTM 模型进行初步检测;接着通过动态分析(如系统调用),使用深度自动编码器进行异常检测。该方法有效整合了 3.1×10^3 个静态特征和 100 个动态特征。文献[17]提出一种基于改进 DenseNet 网络的 Android 恶意软件检测方法,通过引入 SimAM 注意力模块,将特征整合到 2 个维度,实现对重要特征的关注。文献[18]在特征空间拼接 8 组特征,该设计将不同的特征集分开,而不考虑预处理过程中隐藏的关系。文献[19]通过解析 APK 文件的字节码,静态分析组件行为,追踪敏感信息流的源和交汇点,以检测恶意信息。文献[20]提取函数调用图,然后构造频繁子图,代表恶意软件的常见行为样本属于相同的恶意软件家族,最后开发 FalDroid 工具,自动分类恶意软件。

现有深度学习检测方法对特征信息的处理方式依赖特征统计与特征嵌入,通过神经网络模型挖掘各个独立特征之间更深层的特征信息,但在面对包含海量特征的应用软件时,可能存在计算成本高、性能不足或不够健壮的问题。为此,本文提出节点特征构建的 Android 恶意软件检测方法,对特征统计设计嵌入算法之外,还考虑这些特征与其相关的行为意图之间的关系,使用图神经网络模型捕获软件的行为意图特征信息,以提高下游分类器的性能。

2 本文检测方法设计与实现

本文提出一种 Android 恶意软件检测方法,该方法分为 3 个模块,包含特征提取、特征构建以及分类检测,如图 1 所示。使用静态分析技术反编译由 APK 解压的 class. dex 文件,提取函数与函数调用关系,分析 AndroidManifest. xml 文件和反编译后的. smali 文件,提取权限、API、操作码特征并构建函数调用图,对特征进行嵌入;考虑恶意行为与函数调用的对应关系,使用 node2vec[21] 游走算法得到各个函数在调用图中的空间特征,并结合函数自身特征嵌入到向量空间中;结合 DGCN+GAT 的神经网络分类模型学习软件行为特征,对未知软件进行分类,实现 Android 恶意软件检测。

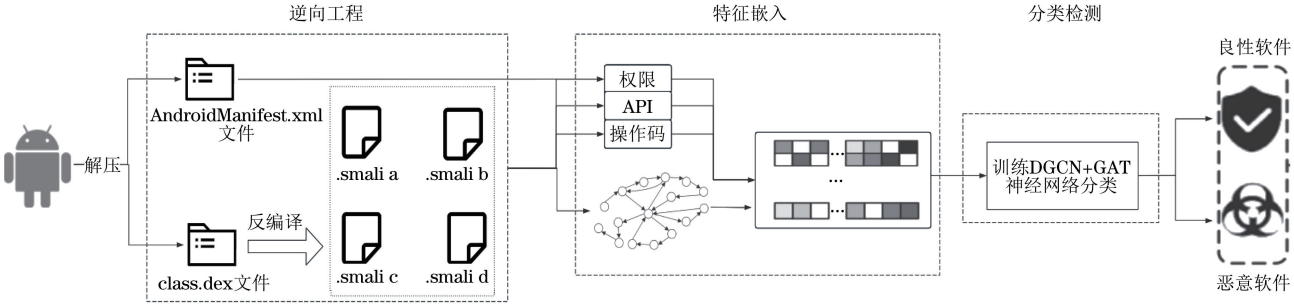

图 1 Android 恶意软件检测方法框架

Fig. 1 Android malware detection method framework

2.1 函数调用图构建

Androguard[4] 是一款开源的 Android 应用程序反编译和分析工具,可在不运行程序的情况下分析静态数据,它提供了强大的 API,能够读取 APK 文件,分析其中的字节码,提取类、方法和指令等信息。通过编写 Python 代码程序,使用 Androguard,可以自动地将 APK 中的. dex 文件反编译为. smali 代码,提取. smali 文件代码内的方法调用关系。

建模函数调用图(FCG)为有向图 $G=(V,E)$,其中每个节点 $v_i\in V$ 代表应用程序中的函数,每条边 $e=(v_i,v_j)\in E$ 表示从 v_i 到 v_j 的调用关系。分析函数之间的调用关系,深入了解应用程序的执行流程,理解各个功能模块之间的依赖关系,从而揭示应用程序的内在逻辑,检测潜在的恶意行为并识别与正常应用程序行为不符的模式,发现可能导致漏洞的代码路径。

对于方法所调用的 API,系统库和第三方库提供了封装好的代码和接口作为 API,使开发者能更方便地实现预设功能开发。API 在功能实现这一层面与函数相同,为此在构图过程中将被调用的 API 也视作一个函数节点。最终,将提取出来的函数节点与调用关系构建为. pkl 格式的有向图。

2.2 特征构建

图结构可用于表示各类复杂网络中的特征数据,让图的丰富信息得以保留,同时减少时间和空间损耗。恶意行为通常包含在恶意软件的代码片段中,FCG 提供了对恶意软件行为的全面视图,本文对节点特征进行构建,不仅要考虑单个函数所包含的信息特征,还要考虑模拟执行过程中的调用行为特征,主要选择了 4 个健壮的属性,即调用图、操作码、API 和权限。

www.c-s-a.org.cn 计算机工程

2.2.1 node2vec

node2vec[21]算法在游走时考虑局部结构，与恶意软件在其运行期间某一行为的函数调用模式相契合，因此，可以采用 node2vec 捕获不同函数之间的语义关系，获取每个函数节点的空间特征。

设随机游走的长度为 1，c_i 为游走的第 i 个节点，起始节点 $c_0=u$。游走策略公式如下：

$$P=(C_i=x\mid C_{i-1}=v)=\begin{cases}\dfrac{\pi_{vx}}{Z}, & (v,x)\in E\\ 0, & \text{otherwise}\end{cases} \tag{1}$$

式中：P 为当前节点 v 访问下一个节点 x 的概率；π_{vx} 是节点 v 与节点 x 未归一化转移概率；Z 是归一化常数。式(1)是最简单直接的随机游走方式，是基于边权重 W_{vx} 进行下一节点采样，如 $\pi_{vx}=W_{vx}$ ($W_{vx}=1$)。但是，这种情况没有考虑到网络的结构，不能进行不同形式邻居节点的探查。BFS 和 DFS 分别适合结构等价和同质的网络的极端采样范式，现实中多是 2 种情况的融合。如图 2 所示，如果定义 2 阶的随机游走，根据 p 和 q 引导游走方式：假设刚游走过一条边 (t,v)，现处于节点 v，基于边 (v,x) 决定下一步游走的转移概率 π_{vx}。设未标准化的转移概率 $\pi_{vx}=\alpha_{pq}(t,x)\cdot W_{vx}$，有如下公式：

$$\alpha_{pq}(t,x)=\begin{cases}\dfrac{1}{p}, & d_{t,x}=0\\ 1, & d_{t,x}=1\\ \dfrac{1}{q}, & d_{t,x}=2\end{cases} \tag{2}$$

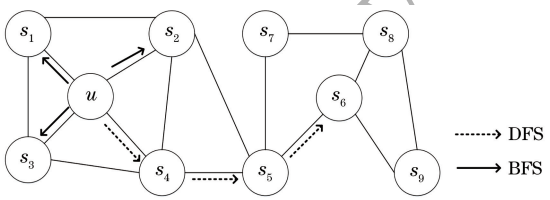

图 2 BFS 与 DFS 的游走示例

Fig. 2 Example of BFS and DFS walk

2.2.2 Dalvik 操作码提取

Dalvik 是谷歌专门为安卓操作系统设计的虚拟机，操作码存在于.dex 文件的代码中。在提取操作码之前，先对内部类和外部类进行区分，其中内部类是从反编译的 APK 的.dex 文件中获得的，可以获得每个内部类的代码源，而外部类来自第三方库，代码通常被混淆或加密，使得提取后的代码难以阅读和理解。对于内部类和外部类之间的差异，根据 Gabor Paller[22]提供的操作码列表，使用长度为 14 的向量来表示每种操作码的特征。对外部类函数，将 API 节点表示为 D-opcodes[0]=1。从.dex 文件中获得函数中详细的代码指令，根据操作码的功能，将其分为 13 种类别，如表 1 所示，根据 Dalvik 指令的操作码获取相应的关键字和 i 值，表示为 D-opcodes[i]=1。

表 1 13 种 Dalvik 操作码

Table 1 13 kinds of Dalvik opcodes

i	Opcode	关键词	作用
1	none	none	无意义
2	01-0D	move	数据操作指令
3	0E-11	return	返回指令
4	12-1C	const	数据定义指令
5	1D-1E	monitor	锁定指令
6	1F/20/22	check	对象操作说明
7	21/23-26	array	阵列操作指令
8	27	throw	异常指令
9	28-2C/32-D	goto/switch/if	跳转指令
10	2D-31	cmpl/cmpg/cmp	比较说明
11	44-6D/F2-F7	iget/iput/sget/sput	字段操作指令
12	6E-72/74-78/F0/F8-FB	invoke	函数调用指令
13	7B-E2	neg/not/int/long/float/double/int/long/float/double	数据类型传输指令

2.2.3 节点重要度

在应用软件运行行为的过程中，将该行为所需的函数间调用与信息传递模式映射到图结构中，可表示为节点之间通过多条路径进行信息传递。通过分析 Katz 算法、度中心算法和接近度中心算法 3 个中心节点算法，可得出结论：每个节点的度值可以直接反映该节点的重要程度。选用 Katz 中心性作为节点重要程度特征，不仅考虑了节点之间的直接连接，还考虑了所有可能的间接路径，能反映节点在行为结构中的重要占比。Katz 算法计算公式如下：

$$K=(\boldsymbol{L}-\alpha\boldsymbol{A})^{-1}-\boldsymbol{L} \tag{3}$$

式中：K 表示节点的重要性分数；$\boldsymbol{L}$ 为单位矩阵，其对角线上的元素都是 1，其余元素为 0；$\boldsymbol{A}$ 为邻接矩阵；α 是衰减因子，其随着路径长度的增加而减小。

构建节点重要程度特征向量 $\boldsymbol{I}_{\text{Node Improtance}}\in\mathbb{R}^3$，对外部类函数节点，使用入度 I，对应用程序初始化函数节点，只使用出度 O。特征表示为：

$$\boldsymbol{I}_{\text{Node Improtance}}=[\text{I},\text{O},\text{Katz Centrality}] \tag{4}$$

2.2.4 权限特征构建

权限决定应用软件是否可以访问某些敏感数据，与恶意行为高度相关。从 Android 6.0 开始，应用软件的功能行为实现与所需权限对于用户而言更具透明度，对于涉及用户隐私或设备安全的敏感权限，即使在 Manifest.xml 文件中声明了它们，也需

计算机工程 www.ecice06.com

要在代码中显式地请求这些权限，在必要时请求用户授权[23]。如图 3 所示，Google 规范化了方法和 API 调用时权限申明的编码格式，使用@RequiresPermission 注解可以验证方法调用方的权限，表明该方法的调用方必须具有 permission.SET_WALLPAPERS 权限。

```
@RequiresPermission(Manifest.permission.SET_WALLPAPER)
public abstract void setWallpaper(Bitmap bitmap) throws IOException;
```

图 3 权限调用示例

Fig.3 Example of permission invocation

本文统计数据集中每个 Manifest 内包含的权限，分析包含的危险权限，排名前 9 的敏感权限如表 2 所示。用 one-hot 编码对其特征构建 $\boldsymbol{F}_{\mathrm{PER}}\in\mathbb{R}^2$，特征向量 $\boldsymbol{F}_{\mathrm{PER}}[0]=1$ 表示该节点为 API，$\boldsymbol{F}_{\mathrm{PER}}[1]=1$ 表示该节点调用了这 9 类敏感权限。

表 2 前 9 类敏感权限

Table 2 Top 9 categories of sensitive permissions

权限名称	说明
READ_PHONE_STATE	访问电话状态
SEND_SMS	发送短信
RECEIVE_SMS	接收短信
WRITE_EXTERNAL_STORAGE	写入外部存储
READ_SMS	读取短信内容
ACCESS_COARSE_LOCATION	获取用户粗略的定位信息
ACCESS_FINE_LOCATION	获取用户精确的定位信息
CALL_PHONE	从非系统拨号器中拨打电话
READ_CONTACTS	访问联系人通讯录信息

2.3 DGCN+GAT 模型

2.3.1 DGCN

图卷积网络(GCN)因在处理图结构数据方面具有出色性能而被广泛使用，但 GCN 存在 2 个缺点：1)仅适用于无向图，对于有向图，其只能采用对称的邻接矩阵，得到半正定的拉普拉斯矩阵，同时失去了有向图的唯一结构；2)在每次卷积操作中仅考虑 1 跳节点特征，但是没有直接相连的节点也可能存在某些关系。文献[24]提出 DGCN，并认为对于有向图，不仅要考虑邻居节点，还要考虑邻近节点。

一阶邻近表示直接相连的顶点，二阶邻近分为入度邻近与出度邻近。如图 4 所示，节点 1 的二阶入度邻近节点为节点 2，2 阶出度邻近节点为节点 3，顶点 1 与顶点 2 有相同的入度顶点，与顶点 3 有相同的出度顶点。对于顶点 1、4，存在$(4,1)\in E$，则$(4,1)$表示为一阶边。对于顶点$(1,2,4)$，存在$(4,1),(4,2)\in E$，则$(1,2)$表示二阶边。

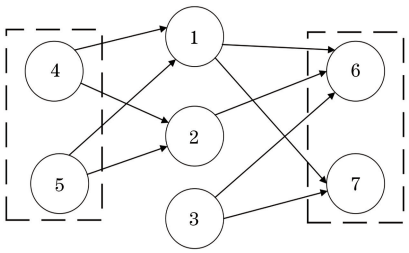

图 4 有向图示例

Fig.4 Example of directed graph

一阶邻近矩阵的表达式为：

$$\boldsymbol{A}_{\mathrm{F}}(i,j)=\boldsymbol{A}^{\mathrm{sym}}(i,j) \tag{5}$$

式中：$\boldsymbol{A}^{\mathrm{sym}}$ 表示邻接矩阵 $\boldsymbol{A}$ 的对称矩阵，相当于转换成无向图的邻接矩阵。通过二阶邻近，将这种转换导致的信息丢失保留下来。

二阶邻近矩阵的表达式为：

$$\boldsymbol{A}_{\mathrm{in}}(i,j)=\sum_k\frac{\boldsymbol{A}_{k,i}\boldsymbol{A}_{k,j}}{\sum_v\boldsymbol{A}_{k,v}} \tag{6}$$

$$\boldsymbol{A}_{\mathrm{out}}(i,j)=\sum_k\frac{\boldsymbol{A}_{i,k}\boldsymbol{A}_{j,i}}{\sum_v\boldsymbol{A}_{v,k}} \tag{7}$$

顶点之间的二阶接近度是通过计算与该顶点对共享邻域节点且相连的所有边的归一化权重总和来确定的，其中，$\boldsymbol{A}_{\mathrm{in}}$ 表示二阶入度邻近矩阵，$\boldsymbol{A}_{\mathrm{out}}$ 表示二阶出度邻近矩阵。以图 4 为例，$\boldsymbol{A}_{\mathrm{in}}(1,2)=\frac{\boldsymbol{A}_{1,4}\boldsymbol{A}_{2,4}}{2}+\frac{\boldsymbol{A}_{1,5}\boldsymbol{A}_{2,5}}{2}=1$。

DGCN 特征融合采用 GCN 的特征融合方式，表达式为：

$$\boldsymbol{H}^{(l+1)}=\sigma(\boldsymbol{A}\cdot\boldsymbol{H}^{(l)}\cdot\boldsymbol{W}^{(l)}) \tag{8}$$

式中：$\boldsymbol{H}^{(l+1)}$ 表示第 $l+1$ 层的节点表示矩阵；σ 是激活函数 ReLU；$\boldsymbol{A}$ 为邻接矩阵；$\boldsymbol{W}^{(l)}$ 是第 l 层的权重矩阵。当 $l=0$ 时，$\boldsymbol{H}^{(l+1)}=\boldsymbol{X}$，$\boldsymbol{X}$ 表示节点特征矩阵。

根据上述公式进行有向图卷积，将 3 个临近矩阵经过同一个滤波器得到的结果进行 Concat，得到一个 $N\times 3C$ 的特征矩阵。DGCN 总体模型结构如图 5 所示。

2.3.2 GAT

受 GCN 的启发，GAT 通过应用共享注意力机制来区分不同相邻节点的重要性，从而学习图网络中节点的有效表示。GAT 能够自适应地聚焦于对当前节点最为重要的邻居节点，精准捕捉图的结构信息和节点间的依赖关系，适用于图网络中以节点特征为主的图预测任务和节点预测任务。

计算机工程 www.ecice06.com

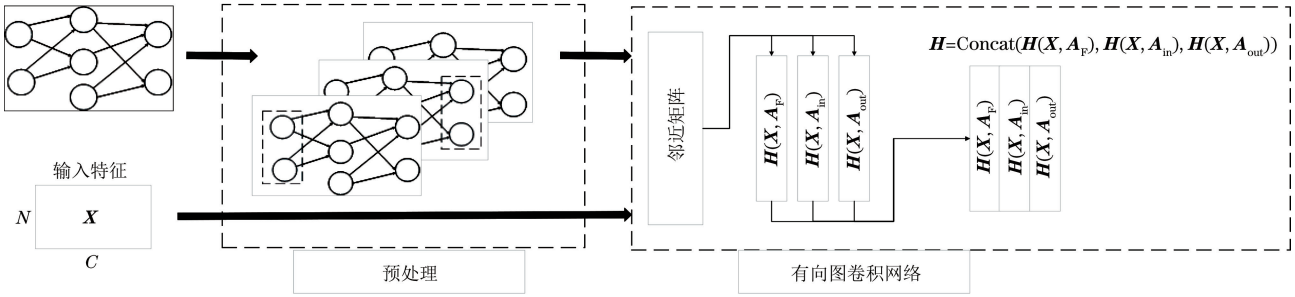

图 5 DGCN 结构

Fig. 5 Structure of DGCN

图注意力机制的运算只在邻居节点上进行。对于节点 i，通过学习得到注意力系数 α_{ij}，从而衡量节点 i 与邻居 j 之间关系的重要性。注意力系数表达式如下：

$$\alpha_{ij}=\frac{\exp(\text{LeakyReLU}(\boldsymbol{a}^{\mathrm{T}}[\boldsymbol{W}\boldsymbol{h}_i\|\boldsymbol{W}\boldsymbol{h}_j]))}{\sum_{k\in N_i}\exp(\text{LeakyReLU}(\boldsymbol{a}^{\mathrm{T}}[\boldsymbol{W}\boldsymbol{h}_i\|\boldsymbol{W}\boldsymbol{h}_k]))} \tag{9}$$

式中：$\boldsymbol{h}_i$ 是节点 i 的特征表示；N_i 是节点 i 的邻居集合；$\boldsymbol{a}$ 是可学习的参数向量；$\|$ 表示向量的连接。节点特征更新的表达式为：

$$\boldsymbol{h}'_i=\sigma\Big(\sum_{j\in N_i}\alpha_{ij}\boldsymbol{W}\boldsymbol{h}_j\Big) \tag{10}$$

式中：$\boldsymbol{h}'_i$ 是更新后的节点 i 的特征表示。

2.3.3 DAg_MAL 模型构建

本文构建 DGCN＋GAT 的图神经网络模型 DAg_MAL，以解决恶意软件 FCG 的分类问题。模型结构如图 6 所示。

对节点特征向量进行归一化后，将处理好的图数据输入到模型中，包含节点特征矩阵 $\boldsymbol{X}$ 和邻接矩阵 $\boldsymbol{A}$。数据首先通过 2 层 DGCN 实现节点特征信息传递。gPool[25] 层在 DGCN 之后应用，以捕获用于图特征选择的重要信息。gPool 计算每个节点的投影分数，并选择得分最高的 $k\%$ 个节点，通过可学习的向量 $\boldsymbol{p}$ 与每个节点特征的点积来估计投影分数：$z=(x_p)/\boldsymbol{p}$，然后选择得分最好的 $k\%$ 个节点：$i_{\text{idx}}=\text{top-}k(z)$。最后，图将更新节点特征矩阵 $\boldsymbol{X}$ 和邻接矩阵 $\boldsymbol{A}$，表达式为：

$$\boldsymbol{X}'=(\boldsymbol{X}\odot\tanh(z))_{\text{idx}} \tag{11}$$

$$\boldsymbol{A}'=\boldsymbol{A}_{\text{idx}} \tag{12}$$

根据 gPool 选择结果删除不重要的节点，只保留重要的节点。第一个 gPool 的输出数据将分别被输入到不同通道中。将输出数据 $(\boldsymbol{X},\boldsymbol{A})$ 输入到 GAT 中，用以学习 FCG 的多维结构信息。GAT 的输出通过第二个 gPool 层，对 gPool 2 分别采用全局最大池化层(GMaP)和全局平均池化层(GMeP)，

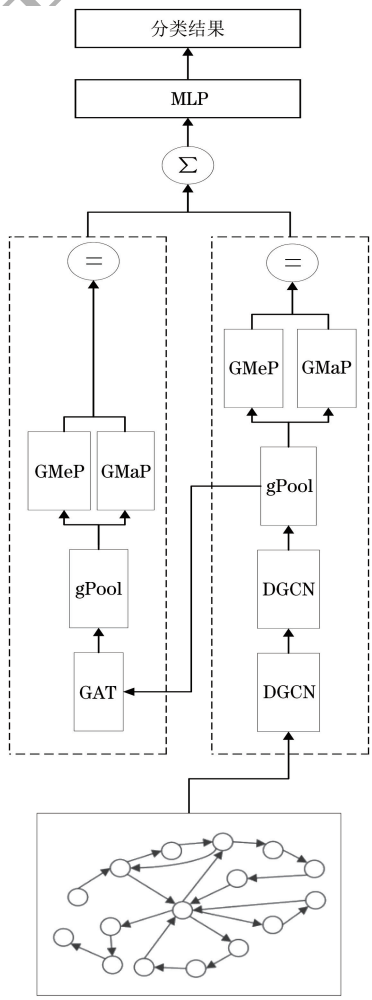

图 6 DAg_MAL 模型结构

Fig. 6 DAg_MAL model structure

从图中提取抽象特征，并将 2 个池化层特征采用 Concat 连接。GMaP 与 GMeP 表达式如下：

$$\text{GMaP}(\boldsymbol{X})_d=\max_{i=1}^{N}\boldsymbol{X}_{i,d} \tag{13}$$

$$\text{GMeP}(\boldsymbol{X})_d=\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{X}_{i,d} \tag{14}$$

式中：$\boldsymbol{X}$ 为节点特征矩阵，尺寸为 $N\times d$，N 是节点个数，d 是每个节点的特征维度。

另一个通道也分别采用 GMaP 和 GMeP 处理，输出数据 $(\boldsymbol{X},\boldsymbol{A})$。2 个池化层特征采用 Concat 连接。将 2 个通道输出的节点特征矩阵进行求和操

计算机工程 www.ecj

作，送入一个 3 层的 MLP 中，MLP 隐藏层采用 ReLU 激活函数。最后，通过一个带有 Softmax 激活的输出层输出概率向量。

3 实验设计与结果分析

3.1 数据集与实验环境

为验证静态特征组合方法检测 Android 恶意软件的能力，考虑到单一数据集中不同恶意软件家族数据的不平衡问题，对实验数据集 Drebin 与 CICdataset2020 进行整合处理，处理后的数据集包含勒索软件(Ransomware)、恐吓软件(Scareware)、短信恶意软件(SMS malware)、风险软件(Riskware)和其他恶意软件(Others)5 个类型，共 3 886 个样本。良性数据通过编写 Python 爬虫代码爬取 Google play 和应用宝收集得到，共 3 665 个良性软件，除去无法反编译或无法从中获得代码资源的软件，能成功反编译并构图的还剩 3 543 个。依照类别，以 7∶3的比例将数据集随机划分用以训练与测试。数据集描述如表 3 所示。

表 3 数据集描述

Table 3 Dataset description 单位：个

分类	类别	数量
二分类	良性	3 543
	恶意	3 886
多分类	勒索	648
	恐吓	837
	短信	769
	风险	857
	其他	775

在本文实验中，硬件环境是：AMD EPYC 7742 64-Core Processor，40 GB 内存，GPU 为 NVIDIA 2080Ti；软件环境是：64 位 Ubuntu 22.04 20 GB，CUDA 11.8，PyTorch 2.1.2，Python 3.9，Androguard 3.3.5。

3.2 实验评价指标

本文使用 4 种性能度量指标，分别为精确率(Precision，P)、召回率(Recall，R)、F1 值(F1，F_1)和准确率(Accuracy，A)。精确率是指模型预测为正样本的实例中真正为正样本的比例；召回率是指所有真正的正样本中被模型正确预测出来的比例；F1 值是精确率和召回率的调和平均数，用于在两者之间进行权衡；准确率是指模型正确预测所有样本(包括正样本和负样本)的比例。这些性能度量通过真阳性(TP，N_{TP})、假阳性(FP，N_{FP})、真阴性(TN，N_{TN})和假阴性(FN，N_{FN})4 个检验指标运算得出，指标定义如表 4 所示。

表 4 指标定义

Table 4 Indicator definition

指标	定义
真阳性	正确标记为恶意的样本数量
真阴性	正确标记为良性的样本数量
假阳性	错误标记为恶意的样本数量
假阴性	错误标记为良性的样本数量

$$P=\frac{N_{TP}}{N_{TP}+N_{FP}} \tag{15}$$

$$R=\frac{N_{TP}}{N_{TP}+N_{FN}} \tag{16}$$

$$F_1=2\times\frac{P\times R}{P+R} \tag{17}$$

$$A=\frac{N_{TP}+N_{TN}}{N_{TP}+N_{FP}+N_{TN}+N_{FN}} \tag{18}$$

对 Android 恶意软件多分类的实验结果指标以表 5 所示混淆矩阵来进行展示。

表 5 混淆矩阵

Table 5 Confusion matrix

实际类别	预测类别					
	Ransomware	Scareware	SMS malware	Riskware	Others	Benign
Ransomware	TP	FN	FN	FN	FN	FN
Scareware	FP	TP	FN	FN	FN	FN
SMS malware	FP	FP	TP	FN	FN	FN
Riskware	FP	FP	FP	TP	FN	FN
Others	FP	FP	FP	FP	TP	FN
Benign	FP	FP	FP	FP	FP	TN

3.3 实验设计

本文在对数据集中应用软件进行节点特征组合构建的过程中，分别对 node2vec 的不同嵌入长度以及 gPool 层保留得分最高的 k%个节点设计组合构建方法，将不同组合方式的结果输入到 DGCN+GAT 模型中，通过交叉实验确定 Android 恶意软件检测二分类的最佳参数，并与不同的图神经网络模型用相同特征组合方式进行二分类训练的结果进行

计算机工程 www.ecice06.com

对比。

使用二分类确定的特征参数对数据集中应用软件构建特征后，对恶意家族软件进行多分类训练与测试实验，并与不同图神经网络模型以及不同文献方法的实验结果进行对比。对多分类实验结果，取各个类别指标加权平均值作为最终实验结果指标。加权平均提供了一个全面的视角，显示了模型在不同大小的类别中的表现，有助于评估模型的整体性能。使用加权平均，是因为它考虑了每个类别的规模，使得大类别对整体评估的影响更大，从而更真实地反映模型的实用性能。

3.4 结果分析

3.4.1 二分类实验与结果分析

使用本文提出的静态特征组合方法，预设 node2vec 嵌入长度分别为 16、32 和 64，构建 Android 应用软件函数调用图数据集。对不同函数调用图数据集，在本文设计的 DAg_MAL 模型中进行实验，实验过程中 gPool 层保留节点数 k 分别取值为 0.7、0.75 和 0.8，实验结果如表 6 所示。

表 6 不同嵌入长度与不同节点数下的检测结果

Table 6 Detection results under different embedding lengths and node numbers

嵌入长度	k	Precision	Recall	F1	Accuracy
16	0.80	0.975	0.973	0.974	0.972
	0.75	0.980	0.972	0.976	0.976
	0.70	0.970	0.966	0.968	0.962
32	0.80	0.976	0.974	0.975	0.974
	0.75	0.978	0.976	0.977	0.976
	0.70	0.970	0.972	0.971	0.971
64	0.80	0.982	0.987	0.985	0.983
	0.75	0.990	0.986	0.988	0.988
	0.70	0.980	0.983	0.981	0.980

由表 6 可知，当嵌入长度为 16 时，模型的整体性能表现相对平稳。在 k 值从 0.8 到 0.7 的变化过程中，虽然各项性能指标有所波动，但差异不大。这表明在较短的嵌入长度下，模型对 k 值的敏感度较低。当嵌入长度增加到 32 时，模型性能有所提升，但提升幅度有限。同样，k 值的变化对性能的影响也不是特别大，但在 $k=0.75$ 时，F1 值和准确率达到最高，显示出在这个嵌入长度下 $k=0.75$ 可能是一个较好的选择。当嵌入长度增加到 64 时，模型性能得到了显著提升，得到了更高的精确率、召回率、F1 值和准确率。尤其是在 $k=0.75$ 时，各项性能指标均达到最高，这表明在这个嵌入长度下，模型能够更有效地学习数据的特征。对于所有的嵌入长度，$k=0.75$ 时模型性能往往较好或达到最佳。在嵌入长度为 16 和 32 时，k 值从 0.8 降低到 0.7，模型性能并没有出现明显下降，甚至在某些情况下还有所提升，如嵌入长度为 32 时，$k=0.75$ 时的 F1 值略高于 $k=0.8$ 时的 F1 值。这表明在这些情况下，模型对 k 值的容错性较高，较长的嵌入长度能够表示更多的特征，同时删除部分无用的节点，减少特征干扰，都有助于提高模型性能。

通过实验，能确定 node2vec 特征维度为 64、模型 DAg_MAL 中 gPool 层保留 75%个节点时，指标结果最好。使用该特征参数与模型参数，将本文模型与不同图神经网络模型以及同类型工作进行对比，结果如表 7 所示，对照组 GCN-2L 是 2 层 GCN 模型，GCN-2L＋GAT-1L 是 2 层 GCN 与 1 层 GAT 的组合。这 2 个对照组均采用本文提出的静态特征组合方法构建的数据集进行训练。文献[26]将应用程序和 Android-API 映射到一个大型异构图中，使用模式构建“应用程序-API”和“API-API”边，将异构图输入到 GCN 模型中。文献[13]通过对 Android 软件进行行为分析得出结论，正常部分与恶意部分高度相关(同质性非常低)。文献[24]将 Android 实体和行为关系建模为一个异构信息网络(HIN)，利用其丰富的语义元结构来指定隐式的高阶关系，并使用 GAT 模型进行训练测试。文献[24]设计了 App-API、App-Permission、Permission-PermissionType、App-so、App-interface 等连接关系，例如，App1 和 App2 之间共享 SEND_SMS 权限，App2 和 App3 之间共享 READ_SMS 权限，那么 App1 和 App3 之间很可能存在隐式连接。

表 7 二分类实验结果对比

Table 7 Comparison of binary classification experimental results

方法	Precision	Recall	F1	Accuracy
GCN-2L	0.928	0.954	0.941	0.949
GCN-2L＋GAT-1L	0.959	0.960	0.960	0.963
文献[26]方法	0.950	0.944	0.947	0.963
文献[13]方法	0.963	0.918	0.963	0.980
文献[24]方法	0.980	0.910	0.947	0.983
本文方法	0.990	0.986	0.985	0.988

从表 7 可以观察到：GCN-2L 作为一个基础图卷积网络模型，表现出一定的性能，但与结合 GAT-1L 的变体相比，其性能仍有待提升，这一对

计算机工程 www.ecice06.com

比凸显了 GAT 在增强图神经网络模型性能方面的显著作用；将文献[26]方法与 GCN-2L+GAT-1L 进行比较，尽管文献[26]方法在整体上表现不俗，但其指标略低于 GCN-2L+GAT-1L，说明本文提出的静态特征构建方法能更好地表达应用软件属性；将文献[13]方法和文献[24]方法与本文方法进行对比，本文方法显著超越了对比方法，精确率高达 0.990，展示本文方法在分类任务中的极高准确性，接近 0.99 的召回率也表明本文模型在识别正样本方面的优势，F1 值和准确率的提升进一步验证了本文方法在 Android 恶意软件检测方面的能力。

3.4.2 多分类实验与结果分析

为进一步验证本文方法的有效性，将其分别在 GCN-2L、GCN-2L+GAT-1L、DAg_MAL 这 3 个模型上进行多分类训练与测试，测试结果如图 7、表 8 所示。

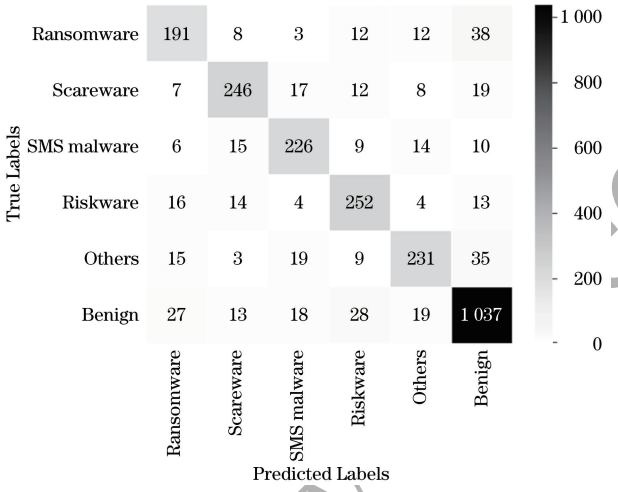

图 7 GCN-2L 多分类混淆矩阵

Fig.7 GCN-2L multi-classification confusion matrix

表 8 GCN-2L 多分类指标

Table 8 GCN-2L multiple-classification indicators

类别	Precision	Recall	F1	Accuracy
Ransomware	0.729	0.724	0.726	0.945
Scareware	0.823	0.796	0.809	0.956
SMS malware	0.788	0.807	0.797	0.956
Riskware	0.782	0.832	0.805	0.953
Others	0.802	0.740	0.770	0.947
Benign	0.902	0.908	0.904	0.916
Weighted Average	0.841	0.842	0.841	0.935

图 7 和表 8 为模型 GCN-2L 的多分类结果。GCN-2L 在 Benign 类上精确率达到 0.902，召回率为 0.908，表明 GCN-2L 在预测正样本时准确性极高，能够有效地识别出真正的正样本，但在 Ransomware 类上精确率为 0.729，召回率为 0.724。混淆矩阵显示，模型在 Benign 和 Others 类别间存在较多的误分类情况，说明模型复杂度较低，同时可能这 2 个类别在特征表达上存在相似性。

图 8 和表 9 为模型 GCN-2L+GAT-1L 的多分类结果。模型在 Benign 和 SMS malware 类上精确率和召回率均超过 0.9，在 Ransomware 和 Others 类上性能相对较低，在混淆矩阵中能看到，这两类被误分到 Benign 类。

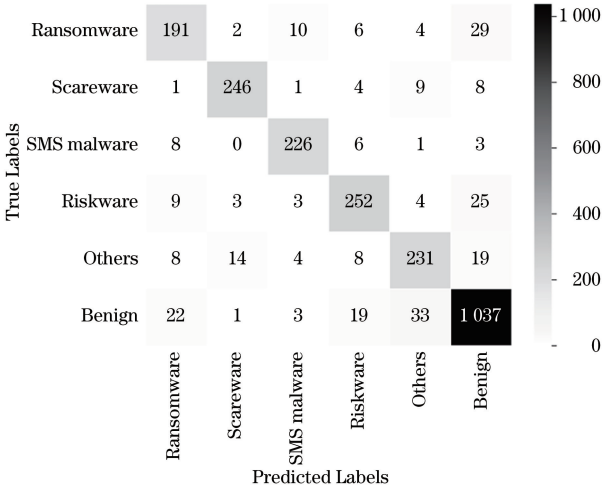

图 8 GCN-2L+GAT-1L 的多分类混淆矩阵

Fig.8 GCN-2L+GAT-1L multi-classification confusion matrix

表 9 GCN-2L+GAT-1L 的多分类指标

Table 9 GCN-2L+GAT-1L multi-classification indicators

类别	Precision	Recall	F1	Accuracy
Ransomware	0.799	0.789	0.794	0.960
Scareware	0.925	0.914	0.920	0.980
SMS malware	0.915	0.926	0.921	0.984
Riskware	0.854	0.851	0.853	0.964
Others	0.819	0.813	0.816	0.958
Benign	0.894	0.930	0.928	0.934
Weighted Average	0.873	0.894	0.894	0.953

图 9 和表 10 为 DAg_MAL 的多分类结果。模型在多个类别上展现出卓越的准确率以及非常高的召回率和精确率，表明其能高效识别和正确分类正样本，F1 值普遍超过 0.95，尤其是在 SMS malware 类上达到了 0.972，显示模型在精确率和召回率间取得了良好平衡。从混淆矩阵可以看出，模型在 Others 和 Benign 类中存在误分类，可能是由于这两类拥有其他类别部分重要功能的行为特征。

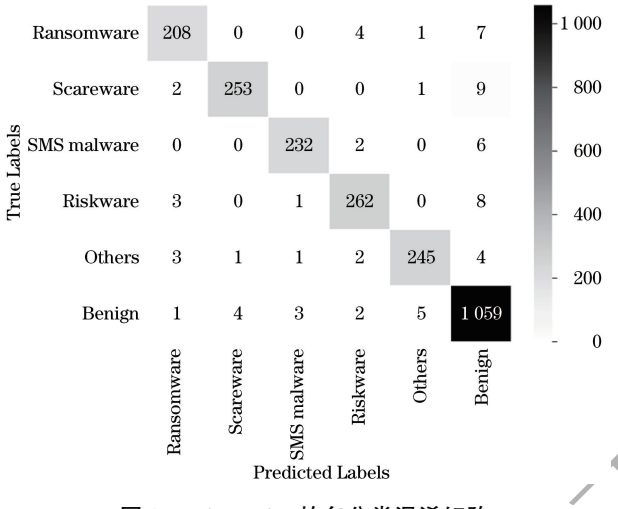

图 9 DAg_MAL 的多分类混淆矩阵

Fig. 9 DAg_MAL multi-classification confusion matrix

表 10 DAg_MAL 的多分类指标

Table 10 DAg_MAL multi-classification indicators

类别	Precision	Recall	F1	Accuracy
Ransomware	0.956	0.946	0.952	0.991
Scareware	0.981	0.955	0.968	0.992
SMS malware	0.979	0.967	0.972	0.994
Riskware	0.963	0.956	0.960	0.990
Others	0.972	0.957	0.964	0.992
Benign	0.969	0.986	0.977	0.980
Weighted Average	0.970	0.970	0.970	0.986

为进一步验证本文方法的有效性，将其与其他方法进行对比，结果如表 11 所示。

表 11 不同方法的多分类实验结果对比

Table 11 Comparison of multi-classification experimental results using different methods

方法	Precision	Recall	F1	Accuracy
GCN-2L	0.841	0.842	0.841	0.935
GCN-2L+GAT-1L	0.873	0.894	0.894	0.953
文献[26]方法	0.863	0.860	0.861	0.959
文献[13]方法	0.905	0.880	0.892	0.966
文献[24]方法	0.920	0.888	0.903	0.967
本文方法	0.970	0.970	0.970	0.986

从表 11 可以看到，本文方法在多分类任务中取得了 0.986 的加权平均准确率，显著优于其他方法。本文方法与 GCN-2L+GAT-1L 对比，说明在相同的特征处理方法下，使用 DGCN 有向图能更好地捕捉样本内节点间的空间关系。

4 结束语

本文提出一种基于图神经网络的 Android 恶意软件检测方法。该方法通过现有反编译工具 Androguard 对 APK 文件进行操作，提取 APK 的 FCG 与特征相关信息；使用本文设计的特征嵌入方法，使得每个节点包含图局部特征和节点自身特征；在输入模型之前，根据本文设计的节点差异算法对特征进行操作；最后，训练与测试 DAg_MAL 模型，对实验参数进行调优。实验结果表明，本文方法在二分类实验中准确率为 0.988，在多分类任务中也取得了良好效果。在未来工作中，将不再局限于分析已有恶意软件特征，尝试挖掘恶意软件的恶意行为，在 FCG 中提取恶意软件的恶意行为子图集，提高 Android 恶意软件检测水平。

参考文献

[1] 360 互联网安全中心. 2023 年上半年度中国手机安全状况报告[EB/OL]. [2024-05-05]. https://pop.shouji.360.cn/safe_report/Mobile-Security-Report-202306.pdf.
360 Internet Security Center. China mobile phone security status report in the first half of 2023[EB/OL]. [2024-05-05]. https://pop.shouji.360.cn/safe_report/Mobile-Security-Report-202306.pdf. (in Chinese)

[2] OR-MEIR O, NISSIM N, ELOVICI Y, et al. Dynamic malware analysis in the modern era—a state of the art survey[J]. ACM Computing Surveys, 2020, 52(5): 1-48.

[3] LEI T, QIN Z, WANG Z B, et al. EveDroid: event-aware Android malware detection against model degrading for IoT devices[J]. IEEE Internet of Things Journal, 2019, 6(4): 6668-6680.

[4] 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.

[5] TAM K, FEIZOLLAH A, ANUAR N B, et al. The evolution of Android malware and Android analysis techniques[J]. ACM Computing Surveys, 2017, 49(4): 1-41.

[6] ODUSAMI M, ABAYOMI-ALLI O, MISRA S, et al. Android malware detection: a survey[EB/OL]. [2024-05-05]. https://www.researchgate.net/publication/328495819_Android_Malware_Detection_A_Survey.

[7] ENCK W, ONGTANG M, MCDANIEL P. On lightweight mobile phone application certification[C]//Proceedings of the 16th ACM Conference on Computer and Communications Security. New York, USA: ACM Press, 2009: 235-245.

[8] SARMA B P, LI N H, GATES C, et al. Android permissions: a perspective combining risks and benefits[C]//Proceedings of the 17th ACM Symposium on Access Control Models and Technologies. New York, USA: ACM Press, 2012: 13-22.

[9] JEROME Q, ALLIX K, STATE R, et al. Using opcode-sequences to detect malicious Android applications[C]//Proceedings of the IEEE International Conference on Communications (ICC). Washington D. C., USA: IEEE Press, 2014: 914-919.

[10] CANFORA G, DE LORENZO A, MEDVET E, et al. Effectiveness of opcode ngrams for detection of multi family Android malware[C]//Proceedings of the 10th International Conference on Availability, Reliability and Security. Washington D. C., USA: IEEE Press, 2015: 333-340.

[11] YAN J P, QI Y, RAO Q F. LSTM-based hierarchical denoising network for Android malware detection[J]. Security and Communication Networks, 2018, 2018(1):

计算机工程 www.ecice.com

5249190.
[12] XIAO X S, YANG S. An image-inspired and CNN-based Android malware detection approach[C]//Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). Washington D.C., USA: IEEE Press, 2020: 1259-1261.
[13] WU Y M, ZOU D Q, YANG W, et al. HomDroid: detecting Android covert malware by social-network homophily analysis [C]// Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis. New York, USA: ACM Press, 2021: 216-229.
[14] XU J Y, LI Y J, DENG R H, et al. SDAC: a slow-aging solution for Android malware detection using semantic distance based API clustering[J]. IEEE Transactions on Dependable and Secure Computing, 2022, 19(2): 1149-1163.
[15] ACHARYA S, RAWAT U, BHATNAGAR R. A low computational cost method for mobile malware detection using transfer learning and familial classification using topic modelling[J]. Applied Computational Intelligence and Soft Computing, 2022, 2022(1): 4119500.
[16] NASSER A R, HASAN A M, HUMAIDI A J. DL-AMDet: deep learning-based malware detector for Android [J]. Intelligent Systems with Applications, 2024, 21: 200318.
[17] 张雪芹, 王逸璇, 赵敏. 基于深度学习的 Android 恶意软件动态检测[J]. 计算机工程与设计, 2024, 45(1): 10-16.
ZHANG X Q, WANG Y X, ZHAO M. Android malware dynamic detection method based on deep learning [J]. Computer Engineering and Design, 2024, 45(1): 10-16. (in Chinese)
[18] ARP D, SPREITZENBARTH M, HÜBNER M, et al. Drebin: effective and explainable detection of Android malware in your pocket[C]//Proceedings of 2014 Network and Distributed System Security Symposium. New York, USA: ACM Press, 2014: 23-26.
[19] 褚堃, 万良, 马丹, 等. 深度可分离卷积在 Android 恶意软件分类的应用研究[J]. 计算机应用研究, 2022, 39(5): 1534-1540.
CHU K, WAN L, MA D, et al. Research on application of depthwise separable convolution in Android malware classification[J]. Application Research of Computers, 2022, 39(5): 1534-1540. (in Chinese)
[20] 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.
[21] GROVER A, LESKOVEC J. node2vec: scalable feature learning for networks [EB/OL]. [2024-05-05]. https://arxiv.org/abs/1607.00653.
[22] Dalvik opcodes[EB/OL]. [2024-05-05]. http://pallergabor.uw.hu/androidblog/dalvik_opcodes.html.
[23] Android developer. Improve code inspection with annotations [EB/OL]. [2024-05-05]. https://developer.android.com/studio/write/annotations.
[24] 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.
[25] CANGEA C, VELIČKOVIĆ P, JOVANOVIĆ N, et al. Towards sparse hierarchical graph classifiers [EB/OL]. [2024-05-05]. https://arxiv.org/abs/1811.01287.
[26] GAO H, CHENG S Y, ZHANG W M. GDroid: Android malware detection and classification with graph convolutional network[J]. Computers & Security, 2021, 106: 102264.

文字编辑 吴云芳
栏目编辑 赖玉玲

计算机工程 www.ecice06.com