

一种基于 API 调用图和外部知识的恶意代码分类方法

夏冰^{1,2}, 楚世豪^{1,2}, 董玉^{1,2}, 唐崇俊^{1,2}, 单征³

¹ 中原工学院 网络空间安全学院, 郑州 中国 450007

² 河南省网络舆情监测与智能分析重点实验室, 郑州 中国 450001

³ 信息工程大学 嵩山实验室, 郑州 中国 450007

摘要 正确识别恶意代码的类别已成为现代数字防御机制的关键组成部分。针对现有基于应用程序编程接口(Application Programming Interface, API)的分类方法存在调用结构关系缺乏、行为语义表达不足等问题, 提出了一种新颖的基于 API 调用图和外部知识的恶意代码分类方法。方法首先基于二进制恶意代码函数的控制流图(Control Flow Graph, CFG)构造函数内部的 API 调用子图, 并作为节点嵌入函数调用图(Function Call Graph, FCG)中生成恶意代码 API 调用图(API Call Graph, ACG), 以捕捉函数内部 API 的调用路径和依赖关系, 接着通过爬虫技术获取 API 的功能描述、参数及返回值等外部知识以增强 API 节点的语义表达能力, 最后借助双向编码器表示(Bidirectional Encoder Representations from Transformers, BERT)和图卷积网络(Graph Convolutional Networks, GCN), 充分挖掘 API 序列词汇语义和 API 调用图节点间的结构依赖语义, 实现了对恶意代码行为的深度表征。实验结果表明, 所提出的方法在多样化的恶意代码样本集上显著优于现有基线模型, F1 分值达到 97.8%, 同时可解释性实验也验证了该方法清晰地识别出恶意行为的关键特征和路径。

关键词 恶意代码; BERT 预训练模型; 自然语言处理; 图卷积神经网络; API 调用图
中图分类号 TP309.5 DOI 号 10.19363/J.cnki.cn10-1380/tn.2026.05.12

A Malicious Code Classification Method Based on API Call Graphs and External Knowledge

XIA Bing^{1,2}, CHU Shihao^{1,2}, DONG Yu^{1,2}, TANG Chongjun^{1,2}, SHAN Zheng³

¹ School of Cyberspace Security, Zhongyuan University of Technology, Zhengzhou 450007, China

² Henan Key Laboratory on Public Opinion Intelligent Analysis, Zhengzhou 450001, China

³ Songshan Laboratory, Information Engineering University, Zhengzhou 450007, China

Abstract Accurate identification of malware categories has become a critical component of modern digital defense. Existing Application Programming Interface (API)-based classification methods often fail to capture call-structure relationships adequately and provide insufficient behavioral semantic representation. To address these issues, this paper proposes a malware classification method based on API call graphs and external knowledge. First, the Control Flow Graph (CFG) of each binary function is used to construct an intra-function API call subgraph, which is then embedded into the Function Call Graph (FCG) to generate an API Call Graph (ACG) for the malware sample, thereby capturing intra-function API call paths and inter-function dependencies. Second, external knowledge, including API descriptions, parameters, and return values, is collected through web crawling from official documentation to enrich the semantic representation of API nodes. Finally, Bidirectional Encoder Representations from Transformers (BERT) and Graph Convolutional Networks (GCN) are combined to jointly learn the lexical semantics of API descriptions and the structural dependency semantics of API call graphs, enabling deep representation of malware behavior. Experimental results on diverse malware datasets show that the proposed method significantly outperforms existing baselines and achieves an F1-score of 97.8%. Interpretability experiments further demonstrate that the method can clearly identify key features and critical paths associated with malicious behavior.

Key words malware; BERT pre-trained model; natural language processing; graph convolutional networks; API call graph

通讯作者: 夏冰, 博士, 副教授, Email: zztixiabing@sina.com。

本课题得到河南省科技攻关项目(No. 252102210217)和国家自然科学基金项目(No. 62302540)资助。

收稿日期: 2024-08-13; 修改日期: 2024-11-28; 定稿日期: 2026-04-09

1 引言

恶意代码通常是由攻击者有预谋编写的一段程序, 以达到特定破坏性目的, 其行为方式隐蔽、针对性强。开展恶意代码的检测和分类至关重要, 这是因为对病毒、蠕虫、木马和勒索软件等不同类型^[1]恶意代码需要采用不同防御策略。比如, 快速识别并处置响应是防御勒索软件的关键, 一旦勒索软件被执行, 需要快速隔离受影响的系统, 清除恶意代码, 恢复受损数据, 从而最大限度地减少对业务运行的影响。

近年来, 深度学习技术在恶意代码检测领域取得了显著的进展。将静态恶意代码转换成文本序列, 使用嵌入学习技术编码后喂入神经网络模型, 实现对恶意软件特征的学习和分类^[2]。然而该方法在处理恶意代码结构信息方面存在局限。应用程序接口 API 不仅能揭示应用程序行为, 而且 API 间的调用关系更能描述行为事件之间的相互依赖性^[3]。因此, 有学者构建以 API 为节点, API 调用关系作为边的 API 调用图, 基于图结构描述恶意代码行为, 开展恶意代码分类研究。文献[4]和文献[5]利用 API 序列捕获恶意代码行为, 但其仅关注 API 调用的前后顺序, 忽略了 API 调用之间复杂的结构关系。最新文献[6]虽然关注了 API 调用结构关系, 但存在 API 行为语义表达不足的问题。

为增强 API 行为语义, 捕获 API 执行过程中各种事件的结构信息特征, 本文提出一种新颖的恶意代码分类方法。方法基于恶意代码函数调用图 FCG 和控制流图 CFG 构建 API 调用图, 引入外部知识增强 API 调用图节点的语义表达能力, 借助 BERT 预训练模型和 GCN 实现恶意代码分类。实验结果表明, 方法提升了恶意代码的语义表达能力, 对比指标均优于基线模型。消融实验和可解释性分析也证明了该方法在恶意代码识别任务上的有效性。

本文主要贡献如下:

(1) 提出一种二进制文件 API 调用图构建技术。通过控制流图提取函数内部的 API 调用子图, 并嵌入到函数调用图中, 建立二进制文件 API 调用图。

(2) 引入外部知识增强 API 行为表示。将 API 的功能描述、参数和返回值等外部知识, 嵌入到 API 调用图节点的特征表示中, 提升了 API 调用图的语义表达能力。

(3) 提出一种恶意代码分类方法。该方法融合词汇语义和结构语义, 实验结果表明在恶意代码家族分类任务中取得优异表现。

2 相关工作

传统的恶意代码分类和识别技术通常依赖于特定的模式或特征码来识别恶意代码^[7]。这种方法在恶意代码采用简单或已知的行为模式时效果很好。然而, 面对复杂的混淆策略^[8], 这些技术往往会捉襟见肘。近年来, 有两组技术显示出了很好的前景。一是基于深度学习的恶意代码分类研究, 该技术借助各种深度学习模型实现恶意代码分类预测, 表现出了有竞争力的结果。二是基于大型语言模型(Large Language Model, LLM)的恶意代码分类策略, 通过与大语言模型进行交互, 完成恶意代码分类等类似任务。

2.1 基于深度学习的恶意代码分类研究

准确描述恶意代码的特征对于静态检测至关重要。其中基于图像可视化的分类技术有助于减轻对专家知识的依赖, 亦可借鉴图像分类领域的先进成果, 成为重要研究方向^[9]。Nataraj 等人^[10]将恶意代码映射为灰度图像, 王博等人^[11]将恶意代码转化为彩色图, 轩勃娜等人^[12]通过坐标注意力机制增强恶意代码图像的特征, 乔延臣等人^[13]将恶意代码视为自然语言描述的文档, 郎大鹏等人^[14]使用操作码序列特征, Chen 等人^[15]利用基本块的 SimHash 值向量。上述方法能全面捕获恶意代码静态特征, 但忽略恶意代码的行为特征。

鉴于 API 调用之间的结构依赖性揭示应用程序行为的关键特征, 不少学者利用 API 描述恶意代码的行为特征, 取得显著成果。D'Angelo 等人^[16]提取代表特定恶意行为的 API 调用子序列, SALEHI 等人^[4]提取恶意代码中的 API 名称及其参数作为分类特征, Zhang 等人^[5]对 API 调用参数进行哈希编码, 文献[17]对 API 调用参数进行敏感度标记, 杨宏宇等人^[18]提出基于 API 调用重组与图像表示的方法, 夏冰等人^[6]提取恶意代码中 API 实体及其调用关系, 构建恶意代码 API 知识图谱实现恶意代码分类。John 等人^[19]利用 API 调用图的中心性测度实现恶意代码检测, Gao 等人^[20]将恶意代码分类问题转化为 API 节点分类问题, Li 等人^[21]构建 API 有向循环图以实现恶意代码的分类。基于 API 行为的分类方法能捕获序列行为和结构特征, 忽略了 API 自身携带的知识语义。

2.2 基于大型语言模型的恶意代码分类研究

随着 GPT 系列模型的兴起, 越来越多的工作尝试将其应用于恶意代码生成和分析中, 利用其强大的上下文语义捕捉能力, 对程序代码(如源代码、汇

编代码和二进制指令)或 API 调用序列进行编码以捕获恶意代码行为模式。Demirkira 等人^[22]利用 BERT 预训练模型捕捉 API 调用之间的序列关系, Yesir 等人^[23]通过优化 API 调用序列, 利用 fastText 和 BERT 模型提升恶意代码分类的性能。文献[24-25]利用 GPT-2 模型对恶意代码反汇编指令序列进行语义特征提取, 提升了恶意软件的检测性能。这些方法的优势在于 GPT 具有强大的长序列特征建模能力, 可以应对复杂的恶意代码结构, 但其在处理图结构数据时存在局限性, 仅适用于序列或文本形式的输入。

有学者将用于代码理解任务的大型语言模型(如 CodeBERT、GraphCodeBERT)引入到恶意代码分析中。这些模型与图结构数据结合, 实现了对恶意代码更精细的特征表示。Hossain 等人^[26]通过对 Java 源代码进行训练, 基于大型语言模型捕捉恶意代码的复杂模式。文献[27]将程序语句、控制流和数据流依赖转化为统一的图表示, 利用代码的语义结构来学习代码表征并生成预训练模型 GraphCodeBERT, 该方法在恶意代码的结构理解和变种检测中取得不错表现。LLM 在恶意代码复杂语义特征表达上具有优势, 但其计算开销较大, 同时大模型微调(Fine-tuning)所需要的大量标注数据, 也使得大语言模型在二进制代码分析领域应用进展缓慢。

2.3 现有方法局限性分析

现有基于 API 调用的恶意代码分类方法在一定程度上能够捕捉恶意代码的行为特征, 但仍存在如下局限性。

API 结构关系缺失。大多数基于 API 调用序列的方法仅关注 API 调用的时序关系, 而忽略了 API 之间的结构关系(如函数间的控制依赖和数据流依赖), 导致模型在处理复杂的嵌套调用和函数间交互时表现不佳。例如, 当恶意代码通过多个函数级调用实现复杂的恶意行为(如信息窃取或权限提升)时, 序列模型无法捕捉这些跨函数的行为路径, 导致分类错误。

API 行为语义表达能力弱。现有的 API 调用图结构化表示通常无法有效表达 API 节点的行为语义, 特别是对 API 调用的上下文信息(如参数类型和返回值)和外部知识(如 API 功能描述)的表达能力有限。这种不足可能导致模型在处理具有相似调用结构的良性软件和恶意代码时, 无法区分其潜在目的和行为差异, 例如, CreateFile 和 ReadFile 等 API 在文件操作中广泛使用, 而现有方法仅考虑其调用频次和先后顺序, 忽视了其特定参数组合所表达的不同意

图, 从而影响模型的判别效果。

3 本文恶意代码分类方法

本方法由三部分组成, 恶意代码 API 调用图构建、API 外部知识抽取, 以及基于图卷积网络(GCN)的分类模型。

3.1 构建恶意代码 API 调用图

给定一个二进制恶意代码文件, 利用逆向分析工具提取其 API 调用图 ACG, 记作 $ACG = (V, A)$, 如图 1 所示, 其中 V 是二进制恶意代码文件中所有的节点函数, 节点函数 $v \in V$ 是一个由二进制函数内部 API 调用关系组成的 API 调用子图。 A 为邻接矩阵, 表示节点函数间的结构化关系。每个节点函数标记为 $v = (B, R)$, 其中 B 为 CFG 内含有 API 的所有基本块, R 为基本块之间的依赖关系。

下面给出基于 Ghidra 的恶意代码 API 调用图的具体实现步骤。

(1) 使用 FunctionManagerAPI 遍历二进制文件中的每个函数, 提取函数的基础信息, 包括起始地址、结束地址、调用者和被调用者等, 并存储所有函数节点及其对应的引用关系。

(2) 利用 DecompInterface 类反编译每个目标函数, 获取其对应的 PCode 指令序列。

(3) 解析反编译得到的 PCode 指令, 分析函数内部的 CFG, 提取函数的基本块、分支逻辑及跳转关系。以基本块为节点、控制流为边, 构建二进制函数 CFG(如图 1(a)所示)。

(4) 通过 ReferenceManager 类提取函数间的调用关系, 生成 FCG, 其中节点表示函数, 边表示函数间的直接调用关系。

(5) 遍历二进制函数 CFG, 分析基本块中是否包含 API 调用指令。若存在 API 调用, 则将该基本块节点标记为 True; 否则标记为 False。在 CFG 中移除所有标记为 False 的基本块节点, 并重构 CFG, 得到仅包含 API 调用的精简 CFG, 即二进制函数内部的 API 调用子图(如图 1(c)所示)。

(6) 遍历整个二进制文件的 FCG, 判断每个二进制函数是否包含 API 调用子图。若二进制函数中存在 API 调用子图, 则将该函数节点标记为 True; 否则标记为 False。在 FCG 中移除所有标记为 False 的函数节点, 并重构 FCG, 生成二进制文件级别的 API 调用图(如图 1(b))。

在 CFG 的指导下, 基于基本块间跳转关系生成函数内 API 调用子图, 再将其子图嵌入至恶意代码

二进制文件的 FCG 中, 生成恶意代码二进制文件 ACG。这样, 整个二进制文件就可以用表示操作行为

的 API 调用图 ACG 代替。API 调用图精准地描述了恶意程序中 API 调用行为及其跨函数依赖关系。

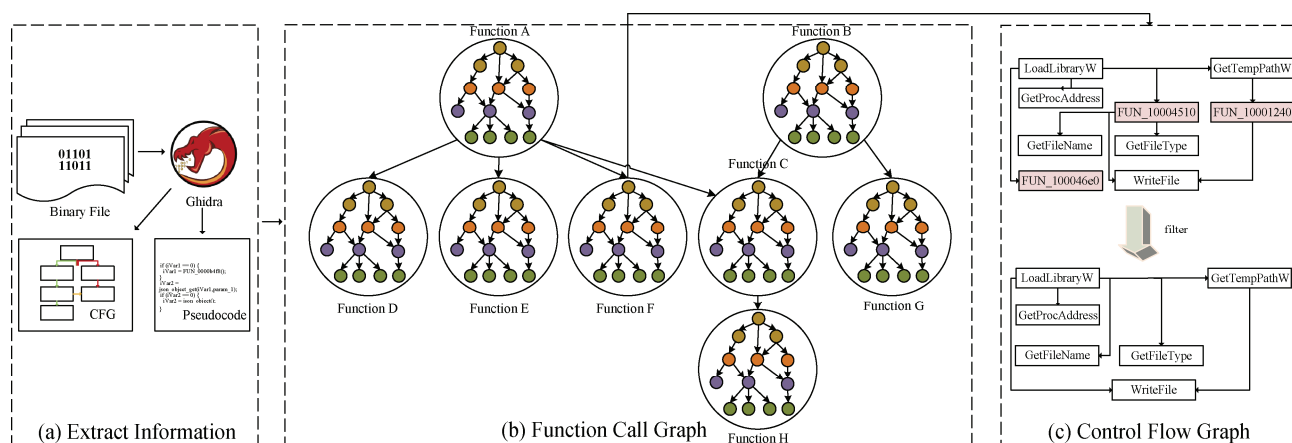


图 1 API 调用图的构建过程

Figure 1 API call graph construction process

由于用户自定义函数在编译过程中通常会被剥离, 并用类似“FUN_10001240”的名称代替, 分析时并不清楚该函数的名称, 不利于分析恶意代码的语义, 因此本文在构建 API 调用关系时过滤掉用户自定义函数。如图 1(c) 中的“GetTempPathW--FUN_10001240--WriteFile”这条调用关系, 在实现过程中删除了“FUN_10001240”用户自定义函数, 将“GetTempPathW 和 WriteFile”直接连接。

3.2 抽取 API 外部知识

API 是操作系统链接库中预定义的一组函数。链接库中的函数通常具有规范的命名, 通过函数名称可以了解该函数的基本功能。本文认为对 API 函数的理解不仅仅局限于其名称, 还应深入到函数背后理解其行为和要实现的功能。例如, 一个文件操作的 API 函数, 通过其名称只能了解到它与文件操作有关; 但如果加入该函数用于创建、读取或修改文件的具体信息, 则能更全面地把握其功能, 从而在分析调用行为时能提取丰富和精确的语义特征, 赋能整个恶意代码的行为理解。

为此, 本文通过自动化爬虫技术, 实现批量从 Microsoft 官方网站获取 API 函数的文本描述、参数和返回值等外部信息。一个名为“DeleteLogFile”的 API 外部知识如表 1 所示。

借助嵌入学习技术, 外部知识为每个 API 函数提供了详细的上下文信息, 使得模型不仅能识别函数的字面意义, 还能理解函数在实际应用中的行为含义。后续实验结果也证实, 外部知识能够有效增强 API 节点的行为语义特征。

表 1 API 外部知识

Table 1 API external knowledge

DeleteLogFile	
Description	Marks a log for deletion. The log is actually deleted when all handles, marshaling areas, and read contexts to the log are closed. If the log is a physical log, its underlying containers are deleted.
Parameters	pszLogFileName, pvReserved
Return value	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero (0).

3.3 基于图卷积网络恶意代码分类网络模型

为了实现恶意代码分类, 本文首先采用 BERT 对每个 API 的详细描述信息进行特征表示(如图 2 所示), 接着利用 GCN 捕捉 API 调用图结构中的局部结构特征与全局语义信息, 最后使用 log_softmax 分类器来处理模型的输出(如图 3 所示)。

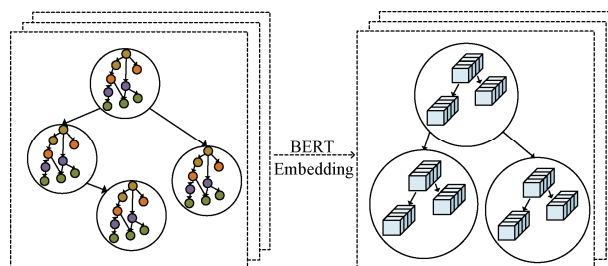


图 2 BERT 嵌入向量过程

Figure 2 BERT embedding vector process

(1) 基于 BERT 的 API 节点学习表示

由于 BERT 模型能够有效捕捉文本中的上下文依赖关系和潜在语义特征, 因此本文采用 BERT 模型对每个 API 的详细描述信息进行特征表示。

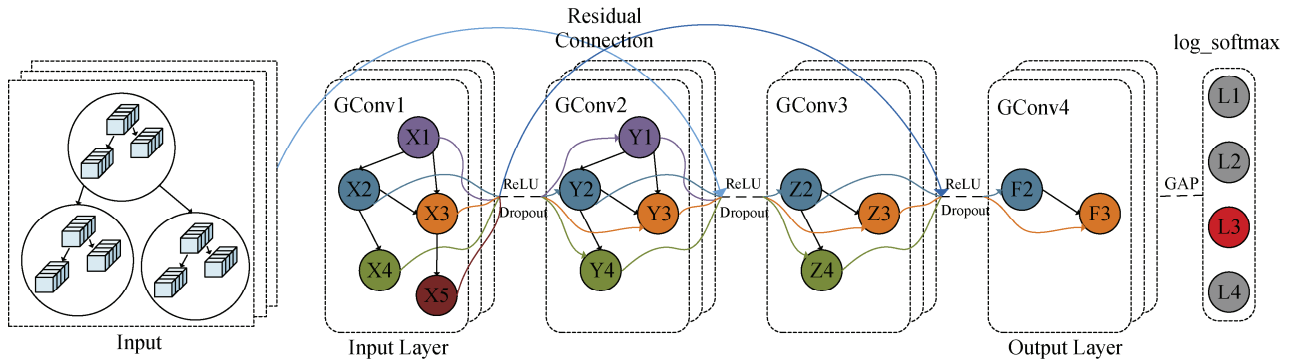


图 3 恶意代码分类模型架构

Figure 3 Malware classification model architecture

BERT 输入以自然语言的形式描述 API 的功能描述、参数说明、返回值等文本内容。文本首先会被分词器处理为一系列子词, 并被转换为模型能够理解的索引。这些索引随后被嵌入到模型的词嵌入层中, 经过多层双向 Transformer 编码器的处理, 每个子词都会生成一个高维的特征向量, 该特征向量表示该子词在上下文中所包含的语义信息和依赖关系。其输出形式是一个多维矩阵, 形状 (l, f) , f 为每个节点的特征维度, 矩阵的每一行对应输入描述文本中某个子词的特征向量。为了得到表示整体 API 描述的全局特征向量, 对输出矩阵中的所有向量进行池化操作。最终得到的全局特征向量是一个定长的一维向量, 形状 (l, f) , 采用该向量表示某个 API 的外部知识的整体语义和上下文关系。

(2) 基于 GCN 的 API 调用图学习表示

图卷积网络(GCN)^[28]能在图节点之间进行特征传递和聚合, 从而捕捉图中复杂的结构信息和节点之间的全局关系, 因此本文采用 GCN 对 API 调用图 ACG 进行结构化建模。

模型的输入数据包括邻接矩阵 E 、节点特征矩阵 X 和特征图类别 Y 。 E 由 $2 \times e$ 维的张量, 用以表明节点间的连接关系, 其中 e 代表 ACG 中边的数量。对于节点特征矩阵 X , 它是一个 $n \times f$ 维的张量, 其中的每一行 X_i 代表一个节点的 f 维特征向量。节点特征矩阵由 BERT 模型对每个 API 结合外部知识的语义表示生成, 特征矩阵大小为 $n \times f$, 其中 n 为图中节点的数量, f 为每个节点的特征维度。矩阵中的每一行表示一个 API 节点的初始特征向量。

GCN 的输出是每个节点的更新特征表示, 大小为 $n \times d'$, 其中 d' 是经过多层卷积后的特征维度。最后, 引入了全局平均池化技术, 将图中所有节点的特征进行聚合。通过对图中每个节点的特征向量进行平均计算, 将大小为 $n \times d'$ 的节点特征矩阵合并成

一个大小为 $1 \times d'$ 的图级别特征向量, 从而生成统一的图表示, 用于图级别的分类任务。

(3) 恶意代码分类

图 3 为本文恶意代码分类模型架构示意图。

模型嵌入了四层图卷积层(Graph Convolution, GConv)来进行特征的更新和传递。GConv 通过聚合其邻近节点的特征来更新节点的特征, 从而增强节点特征的表达力。与此同时, 模型在第二和第三 GConv 层之间引入残差连接^[29](Residual Connection), 以促进深层网络中信息的流动, 并避免梯度消失或爆炸的问题。

在模型的最终阶段, 运用了全局平均池化技术(Global Average Pooling, GAP), 将图中所有节点的特征通过平均池化合并成一个图级别的特征表示, 为整个图的分类任务做准备。如果考虑大小为 $W \times H$ (宽度和高度) 的特征图 F , 则全局平均池化操作 F_f 计算公式(1)如下所示:

$$F_f = \frac{1}{W \times H} \sum_{i=1}^W \sum_{j=1}^H F(i, j) \quad (1)$$

最终, 使用了 $\log_softmax$ 分类器来处理模型的输出。 $\log_softmax$ 是多分类问题中常用的输出层函数, 它输出每个类别的概率分布的对数形式, 便于接下来的概率解释和计算。

4 实验和分析

4.1 实验数据集与评价指标

本实验使用 VXHeavens^[30]恶意代码数据集, 从中选出 7 个恶意代码家族样本, 如表 2 所示。为了减少由数据量差异带来的偏差, 实验在每个家族中均匀随机选择相同数量的样本。数据集被划分为 80% 用于训练, 10% 用于测试, 10% 用于验证。

为全面评估恶意软件分类的效果, 本文使用准确率(Accuracy)、精确率(Precision)、召回率(Recall)

和 F1 分数(F1-Score)四项关键指标来衡量模型性能。

表 2 恶意代码样本
Table 2 Malware samples

恶意代码家族	总数量
Hupigon	1295
Buzus	1382
Small	1353
OnLineGames	1315
Delf	1311
Agent	1356
Vapsup	1364
合计	9376

4.2 实验参数

模型训练和验证的实验环境配置如表 3 所示。

表 3 实验环境配置
Table 3 Experimental environment configuration

实验环境	配置
操作系统	Windows 10
CPU	Intel(R) Core(TM) i7-7700 CPU 3.60GHz
内存	32GB
GPU	NVIDIA GeForce GTX 1050Ti
编程环境	Python 3.10+PyTorch2.0.0+cu118
逆向工具	Ghidra 10.1.2

如图 4 所示, 使用 GCN 进行训练时, 优化器使用 Adam, 学习率为 0.005。学习率采用指数衰减策略, 其衰减率(γ)设置为 0.95 时效果最佳。

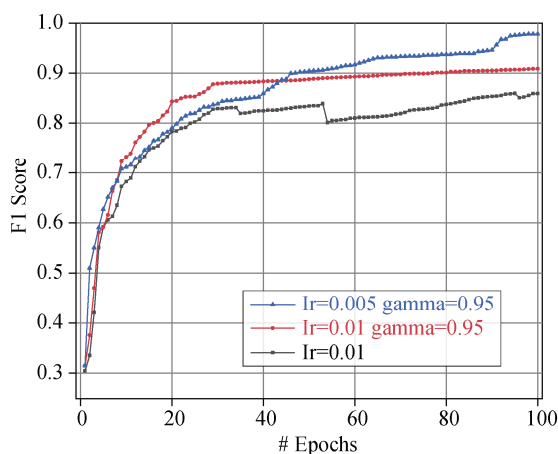


图 4 模型性能随模型参数变化趋势图

Figure 4 Trend chart of model performance changing with model parameters

4.3 实验结果与分析

为验证方法性能, 本文与其他基线模型进行对

比。文献[31]是将操作码(opcode)作为图的节点, 并将操作码之间的关系作为边, 通过将操作码表示为 one-hot 编码形式并运用 GNN 来提取特征以进行分类, 而本文的方法是以 API 调用作为图节点, 以 API 之间的调用关系作为边, 创建图结构并使用 GCN 进行特征学习。

文献[19]、文献[20]和文献[6]都利用了 API 调用作为特征。文献[6]与本文使用同一数据集, 其基于 API 调用关系构造恶意代码 API 知识图谱, 再利用 Word2Vec 对 API 序列编码, 放入 TextCNN 模型中。不同于文献[19]的方法, 文献[20]和本文都采用了节点嵌入技术, 文献[20]使用 Word2Vec 中的 Skip-gram 模式进行节点嵌入。本文的方法独特之处在于节点嵌入阶段, 通过结合 API 和其外部知识并使用 BERT 模型获取其特征向量表示。

文献[16]和文献[22]均利用 API 调用序列作为特征进行恶意代码检测。文献[16]提取 API 调用子序列, 并将其交由贝叶斯模型进行分类。文献[22]在提取 API 调用序列后, 分别使用基于词分片的 BERT 模型和基于字符的 CANINE 模型进行分类。而本文则通过提取 API 调用图来捕捉恶意代码的结构特征实现分类。

对比结果如表 4 所示, 本文方法在恶意代码分类任务上展现出较优的性能。相较于文献[19]的方法, 本方法在分类准确率上提升了 5.6%, 在召回率上提升了 4.4%, 在 F1 Score 上提升了 5.5%。分析原因在于前者是利用图的中心性测度作为输入特征, 而本文将增强型的 API 节点通过 BERT 抽取深层语义信息, 更好地表达恶意代码的行为语义特征。

与文献[20]和文献[6]相比, 本文的各项评价指标均有所提升, 对比结果表明 BERT 模型提供了比 Word2Vec 更深层次的语义表示能力, 引入的外部知识同样增强了 API 节点的行为语义特征。

与文献[16]和文献[22]相比, 本文的各项评价指标均有所提升。对比结果表明, 通过融合结构特征和文本特征, 可以更有效地提高模型的分类性能。此外, 本文在多个数据集上验证了所提方法的有效性。与其他方法相比, 本文的方法在分类性能上同样表现优异, 表明其在不同数据分布下依然能够保持稳定的分类效果, 展现了较强的泛化能力。在 VirusShare 数据集上的 F1 值表现较差, 主要原因在于该数据集是一个类别严重不平衡的多类恶意软件分类数据集。

综上, 本文方法通过构建 API 调用图表示恶意代码的结构化信息, 引入外部知识丰富 API 节点的语义表达能力, 借助 BERT 和图卷积网络学习表示恶意代码的行为信息是行之有效的。

表 4 实验结果

Table 4 Experimental results

	Dataset	Feature	Embedding	Model	Accuracy	Precision	Recall	F1-Score
文献[31]	McAfee Labs	Opcode	One-hot	GNN	0.69	0.67	0.74	0.71
文献[6]	VXHeaven	知识图谱+API 序列	Word2Vec	TextCNN	0.938	0.941	0.943	0.929
本文	VXHeaven	API 调用图+外部知识	BERT	GCN	0.979	0.981	0.977	0.978
文献[16]	TEKDEFENSE	API 序列	-	Naive Bayes	0.935	0.947	0.938	0.961
本文	TEKDEFENSE	API 调用图+外部知识	BERT	GCN	0.977	0.973	0.972	0.972
文献[19]	Drebin	API 调用图	-	GCN	0.923	0.915	0.933	0.923
文献[20]	Drebin	API 调用图	Skip-gram	GCN	0.969	0.969	0.959	0.960
本文	Drebin	API 调用图+外部知识	BERT	GCN	0.9723	0.971	0.967	0.969
文献[22]	VirusShare	API 序列	-	BERT	0.9531	-	-	0.7275
文献[22]	VirusShare	API 序列	-	CANINE-S	0.9286	-	-	0.7064
本文	VirusShare	API 调用图+外部知识	BERT	GCN	0.9636	0.75	0.7326	0.7412

4.4 消融实验

为识别恶意代码分类预测的影响因素, 本文进行了详细的消融研究, 如图 5 所示。

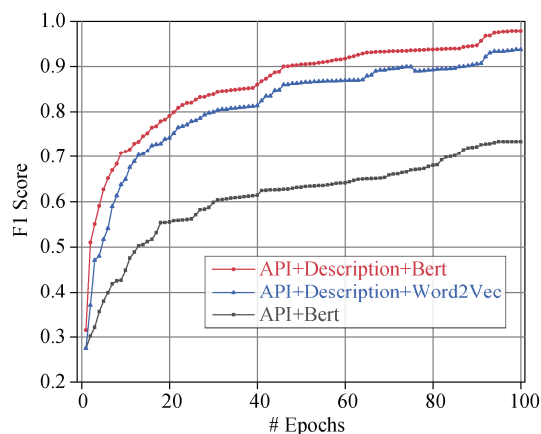


图 5 消融实验

Figure 5 Ablation experiment

在消融实验中, 本文对多个影响因素对恶意代码分类性能的影响进行了深入研究, 以明确各因素在模型性能提升中的作用机制。

首先, 仅将 API 作为节点并利用 BERT 提取特征的方法在 F1 Score 上达到了 74%, 如图 5 中的黑色曲线所示。这是因为 API 名称本身能够表达函数调用的基本语义特征, 但无法涵盖函数的实际行为与上下文信息, 因此仅依靠 API 名称作为特征时, 模型在区分具有相似结构的恶意代码样本时会存在一定局限性。

引入 API 的外部知识后, 模型分类性能显著提

升, F1 Score 攀升至大约 97.8%(如图 5 中红色曲线所示), 可见外部知识的引入极大地增强了 API 节点的语义表达能力, 从而帮助模型更好地区分功能相似但行为不同的 API 调用模式。如两个文件操作 API, CreateFile 和 ReadFile 虽然结构上相似, 但其具体操作意图截然不同。通过引入 API 外部知识, 模型能够准确地识别 ReadFile 涉及的数据读取行为与 CreateFile 涉及的数据创建行为, 从而提升分类的精确度。

最后, 比较 BERT 与 Word2Vec 的嵌入方式可以看出, BERT 的嵌入方法优于 Word2Vec。Word2Vec 基于上下文窗口来学习 API 的词向量, 尽管 Word2Vec 能够在一定程度上捕捉词汇级别的语义特征, 但它无法充分捕捉上下文依赖关系, 也缺乏对长序列和复杂句法结构的建模能力。而 BERT 作为一种双向 Transformer 结构, 能够同时从上下文中捕捉前后词汇的依赖关系, 并生成更深层次的语义表示。例如, 在恶意代码分析任务中, 当 API 功能描述涉及多个参数和条件时, BERT 可以有效区分不同参数组合下的语义差异, 并利用多层 Transformer 网络将这些差异融入最终的嵌入表示中。

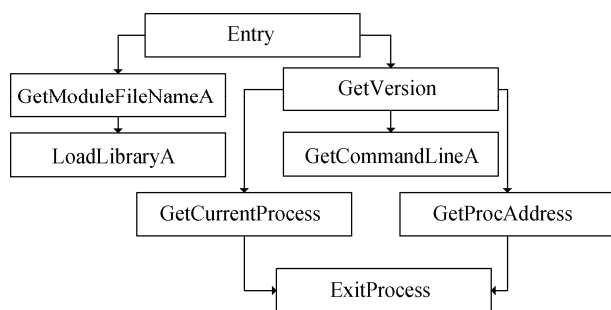
4.5 图模型可解释性分析

为了更好地理解模型的分类预测行为, 本文引入 GNNExplainer^[32]对图神经网络模型进行可解释性分析, 以理解哪些节点和边对模型的决策至关重要, 方便后续优化模型结构, 提高模型泛化能力。

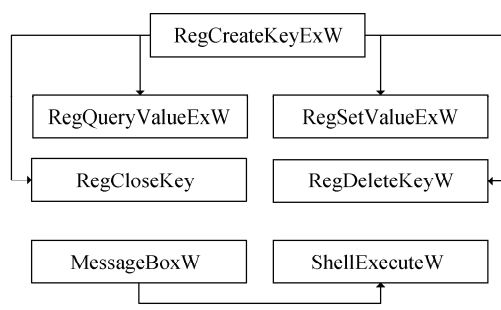
将恶意软件分类模型和待解释的 API 调用图输入到 GNNExplainer, 模型首先构建边掩码和特征掩

码, 并通过训练不断优化这两个掩码, 从而生成一个局部可解释模型, 以逼近原始 GCN 模型在特定预测上的行为; 随后分析获得高权重的边和特征, 识别对模型预测至关重要的图结构部分; 最后生成可视化结果, 展示哪些子图或路径对模型预测起到了决定性作用。

以 OnLineGames 恶意代码家族的实验结果为



(a) Explain For OnlineGames



(b) Explain For Vapup

图 6 模型可解释性

Figure 6 Model interpretability

图 6(b)是 Vapsup 类别(一种广告软件程序)的可解释性分析部分结果, RegCreateKeyExW、RegQueryValueExW、RegSetValueExW 和 RegDeleteKeyW 等函数用于创建、查询、写入和删除系统注册表。广告软件通常会通过修改注册表来改变浏览器设置或系统配置(如修改默认搜索引擎或主页)。从 MessageBoxW 到 ShellExecuteW 这条关键执行路径也是 Vapsup 这类恶意代码的常见特征, 用于显示广告弹窗等。

上述可解释性分析表明, 本文所提方法是行之有效的, 具有一定的实际应用价值。

5 总结与未来工作

本文利用恶意代码的函数调用图(FCG)和控制流图(CFG)等相关信息构建 API 调用图, 并通过自动化爬虫技术获取 API 函数的详细功能描述、参数和返回值等信息, 以增强 API 节点的语义表示能力, 进而构建了一个基于 API 调用图与外部知识的恶意代码分类模型。实验结果表明, 该模型能够有效表征恶意代码的行为语义特征, 并提升恶意代码分类性能。

实验中发现, 部分恶意代码采用了复杂的混淆技术, 并产生了众多变体。由于数据集的限制, 本文无法将所有恶意代码样本纳入深度学习模型进行训练。未来, 拟采用启发式分析方法和大语言模型交互方式, 检测新出现或未知的恶意代码。此外, 本文还发现大量恶意代码在编译过程中剥离了符号信息,

例(如图 6 所示), 图 6(a)对其分类结果影响权重较高。人工分析表明, LoadLibraryA 是操作系统用于加载动态链接库的 API 函数, GetCurrentProcess、GetProcAddress 和 ExitProcess 均与进程或线程操作相关。这些 API 函数所形成的调用路径, 正是 OnLineGames 用于注入恶意代码到游戏进程或其他应用程序中一种典型手段。

这对利用 API 名称来获取行为信息造成了困难。因此, 利用深度学习对剥离符号信息的二进制软件进行语义理解恢复其函数名称, 将是下一步研究工作重点。

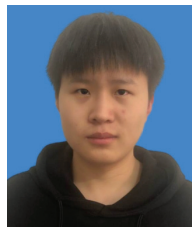
参考文献

- [1] Wang Z W, Liu G Q, Han X H, et al. Survey on Machine-Learning-Based Malware Identification Research[J]. *Journal of Chinese Computer Systems*, 2022, 43(12): 2628-2637.
(王志文, 刘广起, 韩晓晖, 等. 基于机器学习的恶意软件识别研究综述[J]. *小型微型计算机系统*, 2022, 43(12): 2628-2637.)
- [2] He Q D. Research on malicious code classification method based on knowledge graph. Zhengzhou: Zhongyuan University of Technology, 2023.
(何取东. 基于知识图谱的恶意代码分类方法研究[D]. 郑州: 中原工学院, 2023.)
- [3] Saxe J, Berlin K. Deep Neural Network Based Malware Detection Using Two Dimensional Binary Program Features[C]. *2015 10th International Conference on Malicious and Unwanted Software*, 2015: 11-20.
- [4] Salehi Z, Ghiasi M, Sami A. A Miner for Malware Detection Based on API Function Calls and Their Arguments[C]. *The 16th CSI International Symposium on Artificial Intelligence and Signal Processing*, 2012: 563-568.
- [5] Zhang Z Q, Qi P P, Wang W. Dynamic Malware Analysis with Feature Engineering and Feature Learning[J]. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, 34(1): 1210-1217.
- [6] XIA Bing, HE Qudong, LIU Wenbo. Malware Classification Method Based on Knowledge Graph Enhancement[J/OL]. *Journal of*

- Zhengzhou University(Natural Science Edition), 1-8[2024-04-13]. (夏冰, 何取东, 刘文博, 基于知识图谱增强的恶意代码分类方法[J/OL]. 郑州大学学报(理学版), 1-8[2024-04-13].)
- [7] Wolsey A. The State-of-the-Art in AI-Based Malware Detection Techniques: A Review[EB/OL]. 2022: arXiv: 2210.11239. <https://arxiv.org/abs/2210.11239>.
- [8] Zhang C W, Zhou S, Chain B M. Hybrid Epidemics—A Case Study on Computer Worm Conficker[J]. *PLoS ONE*, 2015, 10(5): e0127478.
- [9] Qian L P, Wang D W. A Survey on Image Visualization Approaches-Based Malware Classification Techniques[J]. *Journal of Cyber Security*, 2024, 9(5): 139-161. (钱丽萍, 王大伟. 基于图像可视化的恶意软件分类技术综述[J]. 信息安全学报, 2024, 9(5): 139-161.)
- [10] Nataraj L, Karthikeyan S, Jacob G, et al. Malware Images: Visualization and Automatic Classification[C]. *The 8th International Symposium on Visualization for Cyber Security*, 2011: 1-7.
- [11] Wang B, Cai H H, Su Y. Classification of Malicious Code Variants Based on VGGNet[J]. *Journal of Computer Applications*, 2020, 40(1): 162-167. (王博, 蔡弘昊, 苏旸. 基于 VGGNet 的恶意代码变种分类[J]. 计算机应用, 2020, 40(1): 162-167.)
- [12] Xuan B N, Li J, Song Y F, et al. Malicious Code Classification Method Based on Improved MobileNetV2[J]. *Journal of Computer Applications*, 2023, 43(7): 2217-2225. (轩勃娜, 李进, 宋亚飞, 等. 基于改进 MobileNetV2 的恶意代码分类方法[J]. 计算机应用, 2023, 43(7): 2217-2225.)
- [13] Qiao Y C, Jiang Q S, Zhan L. Malware Classification Method Based OH Word Vector of Assembly Instruction and CNN[J]. *Netinfo Security*, 2019, 19(4): 20-28.
- [14] Lang D P, Ding W, Jiang H C, et al. Malicious Code Classification Algorithm Based on Multi-Feature Fusion[J]. *Journal of Computer Applications*, 2019, 39(8): 2333-2338. (郎大鹏, 丁巍, 姜昊辰, 等. 基于多特征融合的恶意代码分类算法[J]. 计算机应用, 2019, 39(8): 2333-2338.)
- [15] Chen J R. A Malware Classification Method Based on Basic Block and CNN[C]. *Neural Information Processing*, 2020: 275-283.
- [16] D'Angelo G, Ficco M, Palmieri F. Association Rule-Based Malware Classification Using Common Subsequences of API Calls[J]. *Applied Soft Computing*, 2021, 105: 107234.
- [17] Chen X H, Hao Z Y, Li L, et al. CruParamer: Learning on Parameter-Augmented API Sequences for Malware Detection[J]. *IEEE Transactions on Information Forensics and Security*, 2022, 17: 788-803.
- [18] Yang H Y, Zhang Y P, Zhang L, et al. Malware Detection and Classification Based on API Block Reconstruction and Image Representation[J]. *Journal of Cyber Security*, 2024, 9(5): 110-126. (杨宏宇, 张宇沛, 张良, 等. 基于 API 分组重构与图像表示的恶意软件检测分类[J]. 信息安全学报, 2024, 9(5): 110-126.)
- [19] John T S, Thomas T, Emmanuel S. Graph Convolutional Networks for Android Malware Detection with System Call Graphs[C]. *2020 Third ISEA Conference on Security and Privacy*, 2020: 162-170.
- [20] Gao H, Cheng S Y, Zhang W M. GDroid: Android Malware Detection and Classification with Graph Convolutional Network[J]. *Computers & Security*, 2021, 106: 102264.
- [21] Li S X, Zhou Q G, Zhou R, et al. Intelligent Malware Detection Based on Graph Convolutional Network[J]. *The Journal of Supercomputing*, 2022, 78(3): 4182-4198.
- [22] Demirkıran F, Çayır A, Ünal U, et al. An Ensemble of Pre-Trained Transformer Models for Imbalanced Multiclass Malware Classification[J]. *Computers & Security*, 2022, 121: 102846.
- [23] Yesir S, Sogukpinar I. Malware Detection and Classification Using fastText and BERT[C]. *2021 9th International Symposium on Digital Forensics and Security*, 2021: 1-6.
- [24] Demirci D, Şahin N, Şırlancis M, et al. Static Malware Detection Using Stacked BiLSTM and GPT-2[J]. *IEEE Access*, 2022, 10: 58488-58502.
- [25] Şahin N. Malware detection using transformers-based model GPT-2[D]. Middle East Technical University, 2021.
- [26] Hossain A A, Pk M K, Zhang J J, et al. Malicious Code Detection Using LLM[C]. *NAECON 2024 - IEEE National Aerospace and Electronics Conference*, 2024: 414-416.
- [27] Guo D Y, Ren S, Lu S, et al. GraphCodeBERT: Pre-Training Code Representations with Data Flow[EB/OL]. 2020: arXiv: 2009.08366. <https://arxiv.org/abs/2009.08366>.
- [28] Kipf T N, Welling M. Semi-Supervised Classification with Graph Convolutional Networks[EB/OL]. 2016: arXiv: 1609.02907. <https://arxiv.org/abs/1609.02907>.
- [29] He K M, Zhang X Y, Ren S Q, et al. Deep Residual Learning for Image Recognition[C]. *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016: 770-778.
- [30] VxHeaven Dataset[EB/OL] <https://1vx.ug/archive/VxHeaven/>.
- [31] McLaughlin N, Martinez del Rincon J, Kang B, et al. Deep Android Malware Detection[C]. *The Seventh ACM on Conference on Data and Application Security and Privacy*, 2017: 301-308.
- [32] Ying R, Bourgeois D, You J X, et al. GNNExplainer: Generating Explanations for Graph Neural Networks[J]. *Advances in Neural Information Processing Systems*, 2019, 32: 9240-9251.



夏冰 CCF 杰出会员。于 2023 年在信息工程大学软件工程专业获得博士学位。现任中原工学院网络空间安全学院副教授。研究领域为网络安全、逆向工程。研究兴趣包括：网络安全、逆向工程。Email: zztixiabing@sina.com



楚世豪 于 2020 年在中原工学院计算机学院网络工程专业获得学士学位。现在中原工学院网络空间安全学院电子信息专业攻读硕士学位。研究领域为二进制安全、人工智能。研究兴趣包括：二进制安全、LLM。Email: 2570946521@qq.com



董玉 于 2022 年在中原工学院信息安全专业获得学士学位。现在中原工学院电子信息专业攻读硕士学位。研究领域为二进制安全、网络安全。研究兴趣包括: 二进制安全、网络安全。Email: 3510365061@qq.com



唐崇俊 于 2022 年在中原工学院计算机学院网络工程专业获得学士学位。现在中原工学院网络空间安全学院电子信息专业攻读硕士学位。研究领域为二进制安全、网络安全。研究兴趣包括: 二进制安全、网络安全。Email: 572421318@qq.com