软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(6):2510−2526 [doi: 10.13328/j.cnki.jos.007487] [CSTR: 32375.14.jos.007487]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于异质图匹配网络的恶意软件相似性度量方法*

陈永威[1,2], 谷勇浩[1,2], 谢玉奇[1,2], 吴铁军[3]

[1](北京邮电大学 计算机学院 (国家示范性软件学院), 北京 100876)

[2](智能通信软件与多媒体北京市重点实验室 (北京邮电大学), 北京 100876)

[3](绿盟科技集团股份有限公司, 北京 100089)

通信作者: 谷勇浩, E-mail: guyonghao@bupt.edu.cn

摘　要: 现有静态恶意软件相似性度量方法受到静态免杀技术影响, 模型使用的特征易被混淆或者恶意软件语义未被充分挖掘. 提出一种基于异质图匹配网络的恶意软件相似性度量方法 HGMSim (heterogeneous graph matching network-based similarity) 解决上述问题, 该方法首先利用反汇编工具 IDA Pro 提取恶意软件的函数调用图, 将函数调用图抽象为异质图, 充分挖掘函数调用图中不同类型函数节点和函数调用关系的异质语义. 同时, 为了挖掘不同函数调用图节点之间的隐式邻居语义, 对两个函数调用图中相似的同类型函数节点建立跨图边, 构建异质图匹配网络. 然后, 提出基于局部点图匹配的异质图嵌入方法并实现恶意软件相似性度量, 解决现有方法对不同家族之间图结构高度相似恶意软件难区分的问题. 最后, 通过对比实验验证 HGMSim 在恶意软件相似性度量方面具有最佳的性能表现.

关键词: 恶意软件相似性; 函数调用图; 异质图匹配网络; 跨图交互

中图法分类号: TP311

中文引用格式: 陈永威,谷勇浩,谢玉奇,吴铁军.基于异质图匹配网络的恶意软件相似性度量方法.软件学报,2026,37(6):2510–2526. http://www.jos.org.cn/1000-9825/7487.htm

英文引用格式: Chen YW, Gu YH, Xie YQ, Wu TJ. Malware Similarity Measurement Method Based on Heterogeneous Graph Matching Network. Ruan Jian Xue Bao/Journal of Software, 2026, 37(6): 2510–2526 (in Chinese). http://www.jos.org.cn/1000-9825/7487.htm

Malware Similarity Measurement Method Based on Heterogeneous Graph Matching Network

CHEN Yong-Wei[1,2], GU Yong-Hao[1,2], XIE Yu-Qi[1,2], WU Tie-Jun[3]

[1](School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing 100876, China)

[2](Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia (Beijing University of Posts and Telecommunications), Beijing 100876, China)

[3](Nsfocus Technologies Group Co. Ltd., Beijing 100089, China)

Abstract: Existing static malware similarity measurement methods are affected by static anti-antivirus techniques, and the model features are either easily confused or fail to fully capture malware semantics. This study proposes a malware similarity measurement method called heterogeneous graph matching network-based similarity (HGMSim) to address the above problems. This method first uses the disassembly tool IDA Pro to extract a malware's call graph, which is then abstracted into a heterogeneous graph to effectively capture the heterogeneous semantics of different function node types and their call relationships. Meanwhile, cross-graph edges are established for similar function nodes of the same type in two call graphs to mine the implicit neighbor semantics between nodes in different call graphs,

* 基金项目: CCF-绿盟科技"鲲鹏"科研基金 (CCF-NSFOCUS202213); 工业信息安全感知与评估技术工业和信息化部重点实验室开放课题 (202406); 北京邮电大学数智北邮融创项目 (RCXM-2025-029)
收稿时间: 2024-09-09; 修改时间: 2025-02-12, 2025-05-26; 采用时间: 2025-06-12; jos 在线出版时间: 2025-10-29
CNKI 网络首发时间: 2025-10-31

and a heterogeneous graph matching network is constructed. Then, the study proposes a heterogeneous graph embedding method based on local node graph matching strategy and implements malware similarity measurement to solve the problem of difficulty in distinguishing malware with highly similar graph structures between different families. Finally, experimental results show that HGMSim performs best in malware similarity measurement.

Key words: malware similarity; call graph; heterogeneous graph matching network; cross-graph interaction

1 引 言

恶意软件是网络安全的主要威胁之一, 近年来 Windows 恶意软件数量不断增加. 根据 AV-ATLAS 团队提供的数据, 过去一年发现了近 1 亿个新恶意软件 (数据统计网站: https://portal.av-atlas.org/malware/statistics), 大多数新恶意软件都是已有恶意软件结合多态引擎和代码混淆技术创建的变体, 恶意软件变体在拥有相同恶意功能的前提下有着不同的语法表示形式. 当前反病毒工具大多采用基于签名的方法, 该方法易被混淆、多态或其他类似技术绕过[1−3], 同时实时地为每个变体创建签名费时费力. 恶意软件相似性度量方法可以检测恶意软件变体, 基于相似值聚类恶意软件可以推断恶意软件是否属于同一家族或同一组织开发, 有助于安全专家快速了解威胁来源, 辅助构建攻击者画像. 因此, 如何高效准确地度量 Windows 恶意软件相似性有实际应用价值.

相比动态分析只能度量运行代码的相似性, 静态分析能提供更全面的代码覆盖[4], 因此本文专注于基于静态特征的恶意软件相似性度量方法. 按照输入特征不同, 静态恶意软件相似性度量方法主要包括基于程序序列的方法和基于程序图的方法. 基于程序序列的方法效率较高, 但其相似性度量大多为句法相似, 对二进制代码中的简单变换敏感且易受到混淆机制的影响. 基于程序图的方法, 大多使用函数调用图 CG (call graph) 作为方法的输入, 其相似性度量为结构相似, 相似代码的图结构变化较小, 同基于程序序列的方法相比抗混淆能力更强, 能达到更好的相似性度量效果[5]. 但是, 现有基于程序图的方法忽略了 CG 中不同种类函数节点的语义差异以及软件间 CG 语义关系的挖掘, 为此本文将研究如何利用异质图技术和跨图交互技术对恶意软件 CG 建模, 实现有效地恶意软件相似性度量, 解决如下两方面问题.

恶意软件 CG 语义未充分挖掘: 基于程序图的恶意软件相似性度量方法大多简单使用 CG 作为输入, 并未充分考虑 CG 中丰富的异质语义. 首先, CG 中存在着多种类型节点, 主要包括静态链接库函数、动态导入函数和本地函数. 不同类型函数的编写者不同, 如本地函数由恶意软件开发者编写, 不同类型函数的特征提取方式也存在差异 (详见第 3.1.2 节). 其次, CG 中存在着多种类型调用边, 如本地函数调用敏感动态导入函数, 隐含着该软件的恶意行为. 现有基于程序图的恶意软件相似性度量方法在构图时仅考虑同类函数或者在图表征学习时将不同类型函数同等对待, 忽略了 CG 中丰富的异质语义.

图结构高度相似的不同家族恶意软件难以区分: 首先, 不同家族软件会共享代码, 许多新恶意软件是由旧恶意软件代码片段组装而成[4], 部分家族之间共享基础恶意功能的实现代码, 如获取管理员权限、自启动和重启服务等, 导致其函数调用图结构中存在大量相似的子图结构. 图 1 展示了两个不同家族恶意软件的部分重复子图结构 (SG 表示函数调用图中的子图), 其调用关系和对应函数的汇编代码完全一致. 其次, 恶意软件函数调用图部分结构可能被隐藏, simbot 家族恶意软件 (md5 值: 74aa01eb4412c873e732376168f3b772) 在运行时会解密. text 段以执行恶意功能, 静态反汇编仅能生成加解密相关函数的图结构, 恶意功能相关的图结构由于被加密而缺失. 不同恶意家族使用相同加密方法隐藏恶意图结构时, 会导致它们的函数调用图结构变得相似从而难以区分. 现有基于图的恶意软件相似性度量方法只考虑单个恶意软件图结构特征的挖掘, 难以区分高度相似的不同家族恶意软件, 导致相似性度量性能下降.

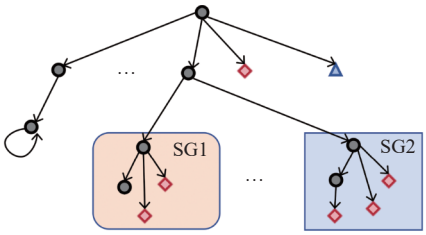
本文提出基于异质图匹配网络的恶意软件相似性度量方法 HGMSim. 首先, 从恶意软件中提取 CG, 为深入挖掘 CG 中不同类型节点的异质语义和 CG 之间的关系语义, 提出异质图匹配网络, 该网络包含两个恶意软件 CG 和 CG 间同类型节点的跨图边. 然后, 提出基于局部点图匹配的异质图嵌入方法, 将构建的异质图匹配网络作为输入, 得到 CG 的嵌入向量, 再利用相似性度量函数得到 CG 间的相似值, 实现恶意软件间的相似性度量. 主要贡献如下.

(1) 构建异质图匹配网络对恶意软件函数调用图之间的关系建模, 提出基于异质图匹配网络的恶意软件相似

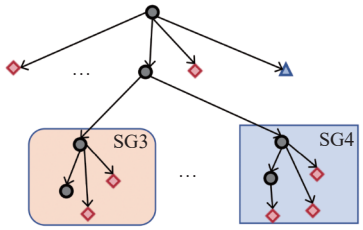
性度量方法 HGMSim.

(2) 提出基于局部点图匹配的异质图嵌入方法, 其中区分函数调用图的异质节点解决恶意软件函数调用图语义未充分挖掘的问题, 局部点图匹配解决图结构高度相似的不同家族恶意软件难以区分的问题.

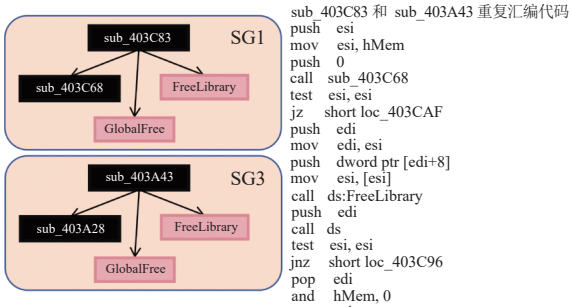
(3) 在多个数据集上实验, 验证所提模型显著优于其他基线方法. 同时, 验证不同反汇编工具生成不同函数调用图对模型带来的影响.



样本 1 局部函数调用图
(md5: eec55f1295d0cce40bbba7be231fdddb)



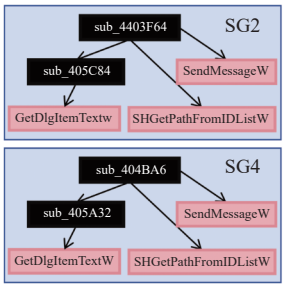
样本 2 局部函数调用图
(md5: 6abf24e3aa3e971e6461bcfea35fe950)



sub_403C83 和 sub_403A43 重复汇编代码

```
push   esi
mov    esi, hMem
push   0
call   sub_403C68
test   esi, esi
jz     short loc_403CAF
push   edi
mov    edi, esi
push   dword ptr [edi+8]
mov    esi, [esi]
call   ds:FreeLibrary
push   edi
call   ds
test   esi, esi
jnz    short loc_403C96
pop    edi
and    hMem, 0
pop    esi
retn 10 h
```

(a) 相似图结构和重复汇编代码示例 1



sub_4403F64 和 sub_404BA6 重复汇编代码

```
...
push   [ebp+lParam]
push   3FBh
call   sub_405A32
push   [ebp+lParam]
push   1
push   467 h
push   [ebp+hWnd]
call   esi ; SendMessageW
cmp    [ebp+arg_4], 2
jnz    short loc_404C05
push   [ebp+lParam]
push   [ebp+pidl]
call   ds:SHGetPathFromIDListW
...
pop    esi
pop    ebp
retn   10 h
```

(b) 相似图结构和重复汇编代码示例 2

图 1 不同家族软件共享代码片段示例

2 相关工作

本节从基于静态分析的恶意软件相似性度量、异质图相似性度量和基于跨图交互的图相似性度量这 3 个方面展开介绍.

2.1 基于静态分析的恶意软件相似性度量

该类方法主要包括基于程序序列的方法和基于程序图的方法.

(1) 基于程序序列

基于程序序列的方法通常使用恶意软件中的二进制序列、字符串信息、导入地址表中的 API 函数以及反汇编后的操作码序列等来计算相似性. Oliver 等人[6]提出了局部敏感哈希方案 TLSH, 该方法计算的哈希值对二进制文件的细微修改不敏感, 能有效用于二进制文件的相似性计算. Adkins 等人[7]考虑到恶意软件开发者经常使用共享代码的技术来提升程序开发质量和效率, 于是将滑动窗口方法应用于二进制文件特征提取, 并通过特征散列来创建可用于相似性度量的指纹. Hsiao 等人[8]将恶意软件转换成恶意样本灰度图, 然后利用多层卷积神经网络获得恶意软件的低维嵌入, 从而进行样本间的相似性度量.

(2) 基于程序图

为了降低混淆等机制对于恶意软件相似性度量的影响, 目前人们倾向于使用对多种语法转换都更加稳健的图结构 (如 CG、控制流图) 作为相似性度量模型的特征输入. Wu 等人[9]通过匹配外部函数名称和本地函数中不同类型指令的数量构建恶意软件中的相似图结构, 利用相似图结构中的重复边占比来计算两个恶意软件的相似度.

Hu 等人[10]提取 CG 并构建函数调用图数据库, 然后使用图匹配技术来近似计算图编辑距离. 上述方法的相似性计算基于图编辑距离或者最大公共子图, 其值的求解均为 NP-complete 问题[11], 即使采用多种加速算法 (如剪枝、启发式计算) 也难以在合理时间内完成对较大恶意软件间的相似性计算.

近年来, 图神经网络通过基于数据驱动的方法将图相似性度量转化为一个学习问题, 训练好的模型可以应用于未见过的图结构, 在相似性度量方面具有更高的效率和准确率[12]. Xu 等人[13]第 1 个把图嵌入和图相似性学习的过程结合起来, 提出了端到端的相似性度量方法 Gemini, 利用控制流图和指令统计属性构建带属性控制流图 ACFG (attributed control flow graph), 使用 struct2vec 模型来将 ACFG 转化成嵌入向量, 采用向量间的相似度衡量函数间的相似性. Puodzius 等人[14]只使用 CG 中的外部节点来构建更简洁的调用图, 作者声称能在保留原有语义的前提下, 提升图相似性度量的效率. ModDiff 模型[15]注意到之前工作中忽略了恶意软件的模块化组成, 将调用图中的函数聚类成更高维的模块表示, 使用比函数粒度更大的模块表示, 能加快计算的效率. COBRA-GCN 模型[16]认为从单一粒度上难以衡量两个二进制文件之间的相似程度, 从指令、函数、调用图这 3 个粒度分别进行嵌入表示, 提升了相似性度量的准确性. Chen 等人[17]认为 CG 中图节点属性使用手工提取的特征会遗漏丰富的汇编指令信息, 于是使用 Asm2Vec 模型通过无监督的方式得到对应函数的特征表示, 然后使用 GraphSAGE 模型对邻域特征进行聚合, 得到该样本的嵌入向量用于相似性度量.

2.2 异质图相似性度量

利用异质图来构建实体之间复杂关系, 能更好地应对恶意软件不断发展的伪装技术. Windows 静态恶意软件相似性度量方法大多基于同质图, 基于异质图的研究还较少. RGCN[18]在扩展了 GCN 的信息传递框架, 引入关系类型权重来区分异质图中不同类型的节点, 从而更好地捕获异质图中的结构信息. HetG[19]利用随机游走扩大邻居采样范围, 对于节点上的多维数据独立设置表示学习方法, 并使用 LSTM 对节点信息聚合, 最后在节点信息传递时结合注意力机制.

2.3 基于跨图交互的图相似性度量

在图相似性度量领域, 人们提出了跨图交互机制, 通过结合对比图的隐式邻居来改进本图的节点嵌入, 通过考虑成对图之间更细粒度的交互, 能有效提高图相似性分析任务的推理精度, 该思想已应用于二进制函数相似性度量方面, 但在恶意软件相似性度量方面的研究还较少. Li 等人[20]认为图结构的细微差异可能会导致语义上非常不同, 传统的方法难以发现两张图之间的微小差异, 在传统图神经网络嵌入的基础上, 加入了跨图匹配机制, 提出了图匹配网络 (graph matching network, GMN) 模型, 该模型在图神经网络更新节点的特征向量时, 不仅聚合了节点的邻居节点信息, 还通过 attention 机制聚合另一张图的所有节点信息, 在实验中取得了最佳的性能表现. graphSim 模型[21]认为传统使用固定维度图级特征向量表征图的嵌入方法难以捕获相似图中的细微差异, 考虑两张图之间点-点之间的交互, 在图神经网络每次信息传递的过程中, 计算两张图所有节点间的相似矩阵, 最后根据获得的多个相似矩阵计算两张图的相似性得分. Ling 等人[22]认为跨层交互 (如点-图) 包含丰富的信息, 提出了一种有效的模型来学习两种图中点-图的交互信息. 现有跨图交互属于全局跨图交互 (聚合另一张图的所有节点信息), 忽略跨图交互过程中不同节点类型的差异, 影响恶意软件相似性度量性能.

3 HGMSim 方法

基于异质图匹配网络的恶意软件相似性度量方法 HGMSim 框架如图 2 所示, 包括异质图匹配网络建模和相似性度量两部分. 在异质图匹配网络建模阶段, 从恶意软件中提取 CG, 为了挖掘 CG 中不同类型节点的异质语义, 进一步区分调用图中节点的函数类型, 将 CG 抽象为异质图. 为了挖掘不同 CG 间的隐式邻居语义, 对两个样本 CG 中相似的同类型函数节点建立跨图边, 构建异质图匹配网络. 在相似性度量阶段, 提出基于局部点图匹配的异质图嵌入方法, 其中函数节点嵌入包括本图异质邻居聚合和局部点图匹配信息聚合. 本图异质邻居聚合解决函数调用图中异质语义未充分挖掘的问题, 局部点图匹配信息聚合解决图结构高度相似的不同家族恶意软件难以区分的问题. 后续章节将分别介绍异质图匹配网络建模和相似性度量.

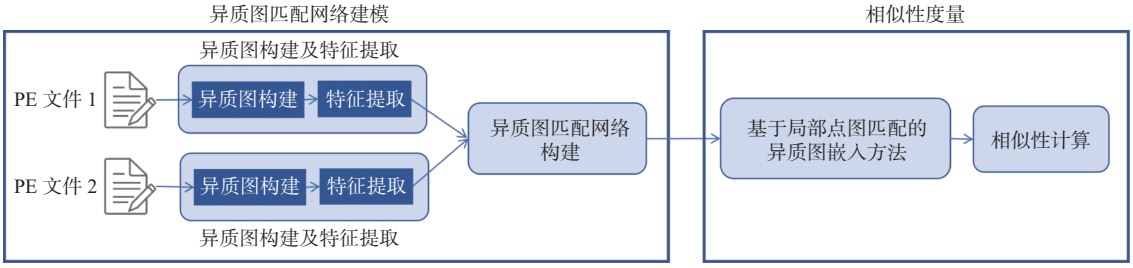


图 2 HGMSim 方法框架图

3.1 异质图匹配网络建模

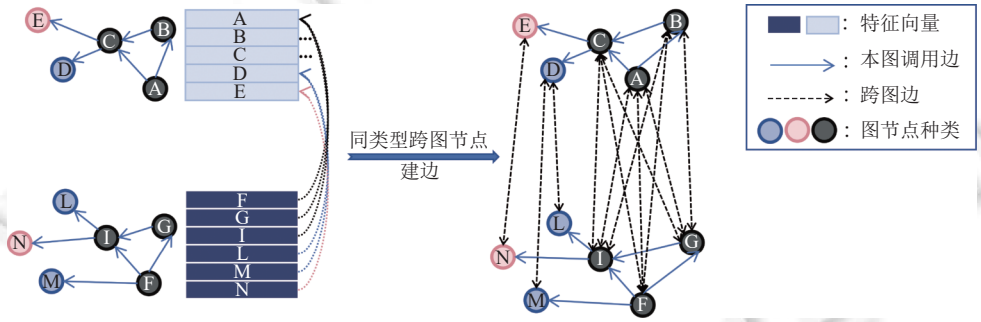
本节介绍如何利用二进制恶意软件生成函数调用图并构建异质图匹配网络.

3.1.1 异质图匹配网络定义

定义 1. 异质图 (heterogeneous graph). 一种包含多种类型实体 (节点) 或关系 (边) 的图, 记为 $HG=\langle V,E\rangle$, 其中 V 表示实体, E 表示关系, 用 A 表示实体类型集合, 用 R 表示关系类型集合, 要求 $|A|>1$ 或者 $|R|>1$.

定义 2. 异质图匹配网络 (heterogeneous graph matching network). 对异质图 HG_1 和异质图 HG_2 中的同类型实体建立跨图边, 构建异质图匹配网络, 记为 $\langle HG_1,E_{\text{cross}},HG_2\rangle$, 其中 E_{cross} 为两张异质图之间建立的跨图边, 要求 $|E_{\text{cross}}|\geqslant 1$.

图 3(a) 是两个异质图的示例, 其图结构中存在着多种类型的图节点, 每个图节点都有对应的特征向量. 图 3(b) 是异质图匹配网络的一个示例, 同异质图相比, 增加了异质图间同类型节点的跨图边.



(a) 异质图示例 (b) 异质图匹配网络示例

图 3 异质图匹配网络生成示例

3.1.2 异质图构建及特征提取

对于每个二进制恶意软件样本提取带属性异质图, 包含异质图构建和特征提取两个部分.

异质图构建: 使用静态反汇编工具 (如 IDA Pro) 从二进制恶意软件中提取函数调用图, 构建如图 4(a) 所示的异质图. 异质图中的节点表示函数, 分为 3 类: (1) 本地函数: 由恶意软件开发者自行编写的函数. (2) 静态链接库函数: 在编译时静态链接到二进制文件中的库函数 (如 C 标准库的“fclose”). (3) 动态导入函数: 在运行或加载时链接的 DLL 函数 (如 Kerel32.dll 中的“GetProcAddress”). 不同类型函数差异较大, 如本地函数通常仅由同一家族中的恶意软件变种共享, 容易使用混淆机制对其结构进行修改[23], 动态导入函数的汇编代码不会出现在恶意软件中. 异质图中的边表示函数间的调用关系, 函数间可能存在着多次调用, 例如图 4 中 lstrlenW 函数被 sub_4014EB 函数调用了两次, 因此本文按照调用次数给边赋权重, 使模型能关注到被调用频次较高的函数节点.

特征提取: 为了避免因函数调用图抽象粒度过粗导致损失汇编代码的语义信息, 同时权衡模型的度量效率, 本文并未使用复杂的表征学习技术获得节点的特征表示, 而是参考 Gemini[13]中提取的基本块特征和 Kim 等人[24]论

文中对二进制代码相似性分析判别能力较高的统计特征, 从函数汇编代码中提取统计信息和函数间的结构信息. 具体提取过程为: 对于本地函数和静态链接库函数节点, 选择了 6 种指令统计特征和 2 种 CG 统计特征作为图节点的特征向量, 提取示例参考图 4(b), 具体特征属性如表 1 所示. 对于动态导入函数节点, 并不能通过静态反汇编获得其汇编代码, 考虑到动态导入函数名称在不同程序中保持一致, 所以利用其函数名生成的 8 维数值编码 (通过哈希算法生成), 作为其图节点的特征表示.

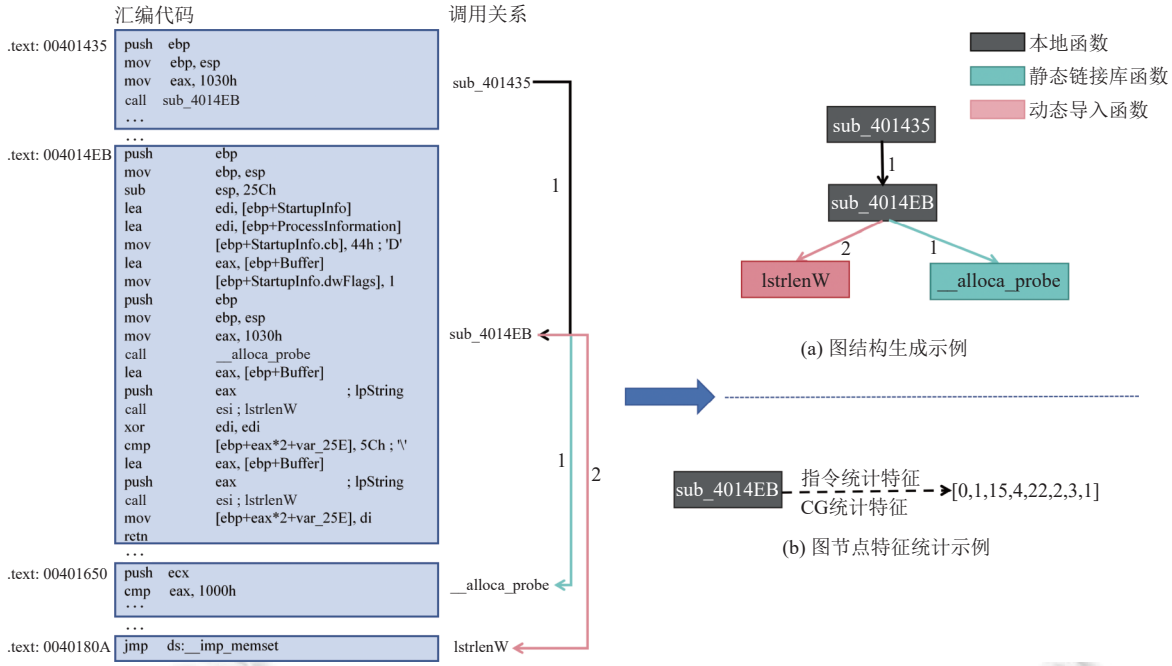


图 4 异质图构建及特征提取示意图

表 1 本地函数和静态链接库函数节点提取特征

类型	属性
指令属性	字符常量数量
	数值常量数量
	转移指令数量
	控制指令数量
	指令总数
	算术指令数量
函数调用图属性	图节点出度
	图节点入度

3.1.3 异质图匹配网络构建

现有恶意软件相似性度量方法中对函数调用图语义的学习仅考虑单图节点间的信息交互, 本文多考虑了捕获恶意软件间丰富的跨图间语义. 基于跨图交互的图相似性度量[22]已经应用于函数相似性度量方面, 但并不适合直接用于恶意软件间的相似性度量, 主要包括两方面原因: (1) 跨图交互方法在每次信息传递过程中, 图中的每个节点都需要与对比图中所有节点间计算相似性, 以便确定隐式邻居, 但恶意软件函数调用图的图节点数量较大 (平均节点数在 500 个左右), 其隐式邻居的计算开销过高. (2) 跨图交互方法用于同质的图结构, 忽略恶意软件函数调用图中不同节点的异质特点. 为了兼顾函数调用图中节点的异质特点以及降低新增的跨图边带来的计算开销, 本文

构建的异质图匹配网络, 是在对单个函数调用图构建的异质图基础上, 仅对两个函数调用图中同类型的函数节点建立跨图边, 降低计算开销同时避免跨图间不同类型函数之间的相互影响, 如图 3(b) 所示.

3.2 相似性度量

本节先详细阐述基于局部点图匹配的异质图嵌入方法, 然后介绍相似性计算和模型训练流程.

3.2.1 基于局部点图匹配的异质图嵌入方法

该方法包括初始化、信息传递和异质图嵌入这 3 个步骤.

(1) 初始化

首先计算异质图匹配网络 $\langle HG_1, E_{\text{cross}}, HG_2\rangle$ 中的跨图边权重, 在计算 HG_1 到 HG_2 的跨图边权重时, HG_1 为本图, HG_2 为对比图, 计算公式如下所示:

$$\omega_{v,i} = sim(\mu_v, \mu_{i,c}),\ \forall i \in V_r^c,\ r = R(v) \tag{1}$$

$$\alpha_{v,i} = Softmax(\omega_{v,i}) = \frac{e^{\omega_{v,i}}}{\sum_{j \in V_r^c} e^{\omega_{v,j}}},\ r = R(v) \tag{2}$$

其中, $R(v)$ 表示节点 v 的函数类型, V_r^c 表示对比图中类型为 r 的节点集, μ_v 表示本图节点 v 的特征向量, $\mu_{i,c}$ 表示对比图中节点 i 的特征向量, sim 为相似性度量函数 (使用 cosine 相似性度量函数), $\omega_{v,i}$ 表示对比图节点 i 与本图节点 v 的相似程度, $\alpha_{v,i}$ 表示经过 $Softmax$ 归一化后的本图节点 v 和对比图节点 i 之间边的权重. 计算 HG_2 到 HG_1 的跨图边权重同理.

然后初始化 CG 中图节点的嵌入向量, 初始化公式如下:

$$\mu_v^{(0)} = W_0 x_v,\ \forall v \in V \tag{3}$$

其中, W_0 是一个 $d \times p$ 的矩阵, d 为函数特征向量维度 (8 维), p 为嵌入向量维度, x_v 为图节点 v 特征向量 (第 3.1.2 节中提取的特征), V 为异质图匹配网络中的节点集.

(2) 信息传递

模型信息传递过程如图 5 所示, 具体过程如下.

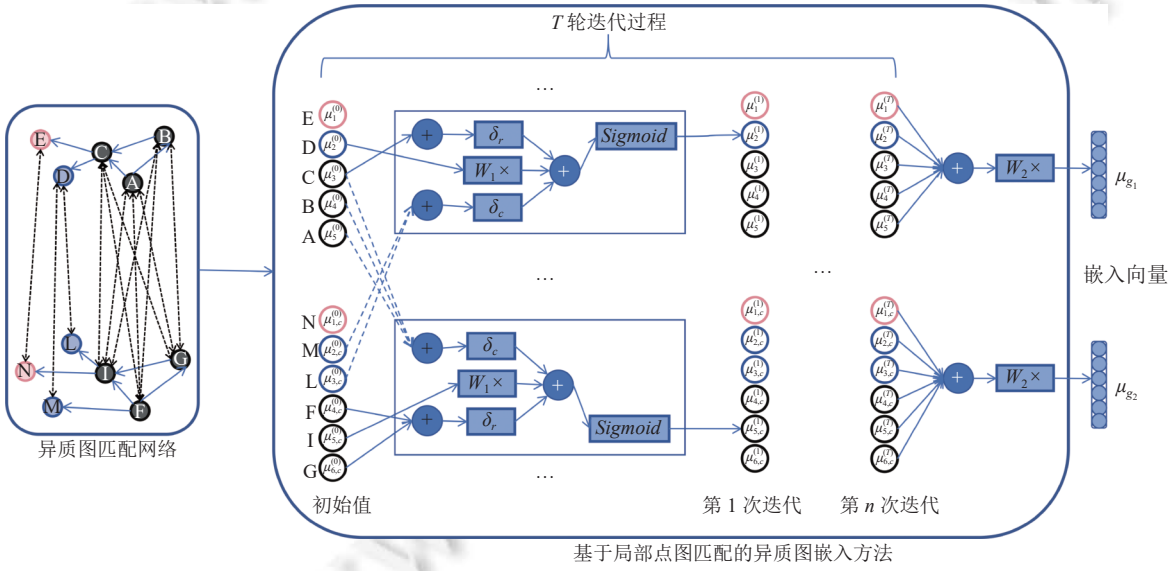


图 5 基于局部点图匹配的异质图嵌入方法

本图异质邻居聚合: 为了使该模型能更好地应用于恶意软件相似性度量场景, 充分挖掘恶意软件函数调用图语义, HGMSim 在信息传递的过程中将不同类型的入边邻居节点区别对待. 使用 $F_{\text{local}}(v,t)$ 表示对节点 v 的第 t 次

聚合:

$$F_{\text{local}}(v,t)=\sum\nolimits_{r\in R}\delta_r\left(\sum\nolimits_{i\in N_r(v)}c_{v,i}\mu_i^{(t)}\right) \tag{4}$$

$$c_{v,i}=\frac{call_{v,i}}{\sum_{u\in N(v)}call_{v,u}} \tag{5}$$

$$\delta_r(l)=P_{1,r}\times ReLU(P_{2,r}\times\ldots\times ReLU(P_{n,r}l)) \tag{6}$$

其中, R 为函数类型的集合, $N_r(v)$ 表示节点 v 类型为 r 的入边邻居节点集, $N(v)$ 表示节点 v 的所有入边邻居节点集, $call_{v,i}$ 表示节点 v 调用节点 i 的次数, $c_{v,i}$ 用来表示节点 v 调用 i 占节点 v 所有入边调用的比例, δ_r 是一个 n 层全连接神经网络 (计算方法见公式 (6)), 其中 $P_{i,r}$ $(i=1,2,\ldots,n)$ 是 $p\times p$ 的矩阵, 对于不同类型的邻居节点使用不同的 n 层全连接神经网络.

局部点图匹配信息聚合: 为了解决图结构高度相似的不同家族恶意软件难以区分的问题, HGMSim 在信息传递中每个节点都增加了跨图间的高相似节点差异信息的传递. 为了更好地寻找对比图中高相似的节点和降低模型的计算开销, HGMSim 提出局部点图匹配信息聚合方法, 即对于本图节点 v, 只使用对比图中和节点 v 同类型的节点生成高相似隐式邻居的局部图嵌入, 计算节点 v 的嵌入和生成的局部图嵌入的差异值作为跨图间的信息传递, 使得模型能捕获高相似图结构间的细微差异, 从而提升模型的相似性度量能力. 使用 $F_{\text{cross}}(v,t)$ 表示对节点 v 的第 t 次跨图交互过程:

$$F_{\text{cross}}(v,t)=\delta_c\left(\mu_v^{(t)}-\sum\nolimits_{i\in V^c_{R(v)}}\alpha_{v,i}\mu_{i,c}^{(t)}\right) \tag{7}$$

其中, $\alpha_{v,i}$ 是异质图匹配网络构建时计算的跨图边权重 (详见公式 (2)), 权重越高表示边对应的节点越相似, δ_c 表示全连接层神经网络 (计算方法同公式 (6)).

综上, 模型的信息传递公式如下所示:

$$\mu_v^{(t+1)}=\sigma\left(W_1\mu_v^{(t)}+F_{\text{local}}(v,t)+F_{\text{cross}}(v,t)\right),\ \forall v\in V \tag{8}$$

其中, W_1 是一个 $p\times p$ 的矩阵, σ 为 *Sigmoid* 激活函数. 根据本图中异质邻居聚合和局部点图匹配信息聚合, 更新函数节点 v 的特征向量.

(3) 异质图嵌入

经过 T 轮迭代后, 图节点的特征向量会包含由图拓扑和所涉及的节点特征决定的 T 跳邻域的信息. 函数调用图的图级嵌入 μ_g 由所有图节点的特征向量聚合形成, 公式如下, 其中 W_2 是一个 $p\times p$ 的矩阵:

$$\mu_g=W_2\left(\sum\nolimits_{v\in V}\mu_v^{(T)}\right) \tag{9}$$

基于局部点图匹配的异质图嵌入算法如算法 1 所示.

算法 1. 基于局部点图匹配的异质图嵌入生成算法.

输入: 异质图 g_1 和对比图 g_2 构成的异质图匹配网络;
输出: 异质图 g_1 和 g_2 的嵌入表示 μ_{g_1}, μ_{g_2}.

1. **for** each node $v_1\in V_1$ **do**
2. Initialize $\mu_{v_1}^{(0)}=W_0x_{v_1}$
3. **end for**
4. **for** each node $v_2\in V_2$ **do**
5. Initialize $\mu_{v_2}^{(0)}=W_0x_{v_2}$
6. **end for**
7. **for** $t=1{:}T$ **do**
8. **for** $v_1\in V_1$ **do**

9. Get $\mu_{v_1}^{(t)}$ using 公式 (8)
10. **end for**
11. **for** $v_2 \in V_2$ **do**
12. Get $\mu_{v_2}^{(t)}$ using 公式 (8)
13. **end for**
14. **end for**
15. Get μ_{g_1} using 公式 (9)
16. Get μ_{g_2} using 公式 (9)
17. **return** μ_{g_1}, μ_{g_2}

对基于局部点图匹配的异质图嵌入方法进行时间复杂度分析, 其中计算跨图边权重的时间复杂度为 $O\left(|V|^2\right)$, $|V|$表示 HG_1 和 HG_2 的节点数, 多层全连接神经网络的时间复杂度为 $O(D|V|F^2)$, D 为全连接神经网络层数, F 为向量嵌入维度, 一次本图异质邻居聚合的时间复杂度为 $O(2(\|A\|_0 F + D|V|F^2))$, 其中 $\|A\|_0$ 为邻接矩阵中非零的个数, 一次局部点图匹配信息聚合的时间复杂度为 $O(2(|V|^2 F + D|V|F^2))$, 综上算法复杂度为 $O(|V|^2 + 2L(|V|^2 F + D|V|F^2 + \|A\|_0 F + D|V|F^2))$, L 为信息传递层数. 其中 $\|A\|_0 \ll |V|^2$, 算法复杂度可以简化为 $O(L(|V|^2 F + D|V|F^2))$.

3.2.2 相似性计算及模型训练

模型最后利用相似性度量函数计算两个嵌入向量 μ_{g_1} 和 μ_{g_2} 的相似性, 来确定两个二进制恶意软件间的相似程度, 本文参考 Gemini[13], 使用简单高效的余弦相似性作为相似性度量函数, 具体公式如下:

$$sim(g_1, g_2) = \cos(\mu_{g_1}, \mu_{g_2}) = \frac{\langle \mu_{g_1}, \mu_{g_2} \rangle}{\| \mu_{g_1} \| \| \mu_{g_2} \|} \tag{10}$$

为了训练相似性度量模型, 从恶意软件函数调用图集中选择训练元组集合 $D = \{(g_i, g_i', g_i'') : i = 1, 2, \ldots, m\}$, 其中 g_i' 与 g_i 相似, g_i'' 与 g_i 不相似. 模型训练时, 为了使相似样本的嵌入向量相似, 同时不相似样本的嵌入向量差异变大, 使用了排名损失函数中的 triplet loss 函数来计算损失, 最大化 $sim(g_i, g_i')$ 和 $sim(g_i, g_i'')$ 之间的差异, triplet loss 函数如下所示:

$$Loss(g_i, g_i', g_i'' = \sigma(sim(g_i, g_i' - sim(g_i, g_i'') + margin) \tag{11}$$

其中, $\sigma(x) = \max(x, 0)$, $margin$ 为一个超参数. 模型训练过程如算法 2 所示.

算法 2. 模型训练流程.

输入: 训练对 $D = \{(g_i, g_i', g_i'') : i = 1, 2, \ldots, m\}$, 迭代次数 T, 嵌入维度 d, 全连接层网络层数 n;
输出: 模型参数 Θ.

1. $B \leftarrow GenerateBatches(D)$
2. **for** each batch $b \in B$ **do**
3. Initialize the loss $L_b = 0$ for the batch b
4. **for** $g, g', g'' \in b$ **do**
5. Get the embedding vector μ_g, μ_g', μ_g'' of g, g', g'' using 算法 1
6. Get $Loss(g, g', g'')$ using 公式 (11)
7. Get $L_b \leftarrow L_b + Loss(g, g', g'')$
8. **end for**
9. Update Θ using Adam according to L_b
10. **end for**
11. **return** Θ

4 实验分析

4.1 实验设置

4.1.1 数据集

绿盟企业数据集: 数据集来自绿盟科技提供的 2012–2021 年间捕获的 2 620 个恶意软件. 利用 VirusTotal 来判断样本是否为恶意软件, VirusTotal 整合了 70 多种防病毒引擎来检测样本, 为避免少数几个引擎的误报, 本文参考文献 [25] 将报毒引擎数小于 5 个的样本视为良性样本, 不参与相似性的计算, 将报毒引擎数大于等于 5 个的样本标记为恶意样本, 恶意样本家族标签通过 VirusTotal 结合恶意样本家族标记工具 AVclass2[26]获得. 筛除良性样本, 最后共 2 500 个恶意软件 (样本 md5 和 HGMSim 源码: https://gitee.com/yt_mm/hgmsim-code) 进行实验, 包含 155 种恶意软件家族.

BODMAS 数据集[27]: 由 Blue Hexagon 发布的包含时间戳和精心策划家族信息的恶意软件数据集, 其采集于 2019 年 8 月–2020 年 9 月, 共包含 57 293 个恶意软件样本, 累计 581 个家族. 不同家族样本数量差异较大, 本文只选择恶意软件数量大于 10 的家族, 同时每个家族最多选择 100 个恶意软件, 最后共 9 802 个恶意软件进行实验, 包含 196 个家族.

利用家族标记获得样本对间的相似性关系, 将同家族样本相似度标记为相似, 不同家族样本相似度标记为不相似. 随机将 70% 样本作为训练请求图, 15% 样本作为测试请求图, 15% 样本作为验证请求图. 对于每个训练请求图 g_i, 随机从训练请求图选取一个相似图 g_i' 构成正样本对和一个不相似图 g_i'' 构成负样本对, 绿盟企业数据集共生成 3 500 个样本对作为训练集. 验证集和测试集同理. BODMAS 数据集共生成 13 722 个样本对作为训练集, 验证集和测试集同理.

4.1.2 基线模型

将所提 HGMSim 方法与基于程序序列、基于程序图、基于异质图和基于跨图交互等的相似性度量方法进行对比.

TLSH[6]: 广泛使用的基于局部敏感 hash 的相似性度量算法, 该方法使用局部敏感 hash 作为二进制样本的签名, 利用 hash 值之间的相似值来衡量两个样本间的相似程度. 数据预处理: 直接使用原始二进制恶意软件. 通过 Python 的 TLSH 库实现该方法.

Siamese_CNNs[8]: 基于恶意软件灰度图的二进制恶意软件相似性度量算法, 该方法将恶意样本转换成灰度图, 利用 CNN 获得恶意样本图片的表示, 通过 Siamese 架构结合全连接层得到样本对的相似值. 数据预处理: 将二进制恶意软件转换成 8 位向量图像, 同时使用双边插值法将向量图像转换成 105×105 的灰度图. 实现的模型结构和原论文保持一致.

Siamese_GNN[17]: 基于图神经网络的恶意软件相似性度量算法, 使用 Asm2Vec[28]模型获得样本函数调用图中各个函数节点的特征向量, 然后使用设计的 GNN 获得恶意样本嵌入, 利用嵌入间相似值来衡量样本间的相似值. 数据预处理: 反汇编恶意软件获得其函数间控制流图, 获得恶意软件的函数调用图和各函数中的汇编指令, 利用生成的 180 万个函数汇编指令, 训练 asm2vec-pytorch (https://github.com/oalieno/asm2vec-pytorch) 提供的 Asm2Vec 模型, 用于函数调用图中节点特征向量的计算. 实现的模型结构和原论文保持一致.

RGCN[18]: 异质图领域的经典模型, 该方法扩展了 GCN 的信息传递框架. 引入关系类型权重来区分异质图中不同类型的节点, 从而更好地捕获异质图中的结构信息. 数据预处理: 反汇编样本获得函数调用图, 统计指令统计信息作为图节点的特征向量, 实现的模型结构和原论文保持一致.

MGMN[22]: 跨图交互领域最新的模型, 该方法考虑跨层交互 (如点-图) 中丰富的信息, 提出多层图匹配网络 (MGMN) 框架, 用于端到端计算任意一对图结构对象之间的图相似度. 数据预处理: 反汇编样本获得函数调用图, 统计指令统计信息作为图节点的特征向量. 使用 MGMN 源码 (https://github.com/ryderling/MGMN) 进行实验.

4.1.3 超参数设置

基线模型的超参数设置尽可能与原论文保持一致, HGMSim 的超参数选择参考 Gemini[13], 具体数值由调参

生成.

TLSH 方法: 无需训练, 无超参数.

Siamese_CNNs: mini-batch size: 6, 学习率: 0.000 06, 训练轮次: 2 000.

Siamese_GNN: 学习率: 0.001, batch size: 64.

RGCN: 学习率: 3E–5, 层数: 2, 嵌入向量维度: 128.

MGMN: 学习率: 0.000 5, 嵌入向量维度: 100, dropout: 0.1.

HGMSim: 学习率: 3E–5, 层数: 2, 全连接网络层数: 2, 嵌入向量维度: 128.

4.1.4 评价指标

实验使用的评价指标包括分类指标和检索评价指标.

分类指标: 由于恶意软件相似性度量可以判断两个样本是否属于同一家族, 考虑到不同相似性度量模型判断样本对是否属于同一家族的阈值并不相同[29], 本文采用 AUC (area under curve) 指标, AUC 被定义为 ROC (receiver operating characteristic) 曲线下的面积, 用于综合衡量模型对于测试集样本在所有可能分类阈值中的性能表现.

检索评价指标:

(1) 平均倒数排名 *MRR* (mean reciprocal rank)

$$MRR = \frac{1}{N}\sum_{p_i \in P}\frac{1}{f_{\min}(rk_{p_i})} \tag{12}$$

对于测试请求图 $P=\{p_1,p_2,\ldots,p_n\}$ 中的每个样本 p_i, 从测试请求图构建与 p_i 家族标记相同的样本集合 $SF_{p_i}=\{sp_1, sp_2,\ldots,sp_m\}$, 并获得 p_i 的预测排名 $rk_{p_i}=\{rank_{sp_1},rank_{sp_2},\ldots,rank_{sp_m}\}$ 和真实排名 $gt_{p_i}=\left\{rank^{gt}_{sp_1},rank^{gt}_{sp_2},\ldots,rank^{gt}_{sp_m}\right\}$, 使用模型计算整个测试请求图中所有样本和 p_i 的相似值, 其中预测排名为 p_i 同家族样本相似度在整个测试请求图样本中的排名情况, 真实排名根据目标样本的家族标记构建, 即来自同一家族的样本之间的排名高于来自不同家族的样本的排名, $f_{\min}$ 用于取出集合中最高的排名.

(2) 归一化折损累计增益 *NDCG* (normalized discounted cumulative gain)

$$NDCG@k = \sum_{p_i \in P}\frac{DCG_{p_i}@k}{IDCG_{p_i}@k} \tag{13}$$

其中, $DCG_{p_i}@k=\sum_{loc\in rk_{p_i}}\dfrac{2^{f(loc,k)}-1}{\log_2(loc+1)}$, $DCG_{p_i}@k$ 是样本 p_i 返回的前 k 个样本的折扣累计增益, $IDCG_{p_i}@k=\sum_{loc\in gt_{p_i}}\dfrac{2^{f(loc,k)}-1}{\log_2(loc+1)}$, $IDCG_{p_i}@k$ 是样本 p_i 在理想情况下返回的前 k 个样本的折扣累积增益, $f(n,k)=\begin{cases} n, & \text{if } n<k \\ 0, & \text{if } n\geqslant k \end{cases}$.

4.2 模型评估

为了验证模型性能, 论文通过实验回答以下 3 个问题.

RQ1: HGMSim 同基线模型相比, 在多个指标下的性能表现如何? (第 4.2.1 节)

RQ2: HGMSim 异质信息语义挖掘和局部点图匹配方法的有效性如何? (第 4.2.2 节)

RQ3: 不同反汇编工具生成图结构的差异对模型性能的影响如何? (第 4.2.3 节)

实验在配备两个 Intel(R) Xeon(R) Silver 4210R CPU (总共 40 个逻辑核心, 运行频率 2.40 GHz), 128 GB 内存, 1 块 3090 显卡的服务器上进行. 对于需要使用反汇编工具的基线方法, 本文使用其指定的反汇编工具来进行数据预处理, 如果未指定反汇编工具, 便统一使用业界主流的反汇编工具 IDA Pro 进行数据处理.

4.2.1 模型相似性度量性能对比实验

本节将从模型性能和时间复杂度两方面, 将 HGMSim 模型和基线模型进行对比.

模型性能对比结果如表 2 和表 3 所示. 可以看到, HGMSim 在全部指标上都高于基于程序序列的 TLSH 和 Siamese_CNNs 方法, 表明将恶意样本构建成异质图匹配网络, 能大幅度提升模型的相似性度量性能. HGMSim 指标优于 RGCN, 表明增加跨图间的信息传递, 能有效提升模型的检索性能. 相对于基于跨图匹配的 MGMN 方法, HGMSim 在所有指标上都有一定提升, 表明区分图节点的异质性, 有助于模型挖掘恶意软件的语义信息. HGMSim 同基于程序图的 Siamese_GNN 相比, 两者在 AUC 分类指标上十分接近, 且明显高于其他基线模型的性能, 表明

HGMSim 计算出的正样本对之间的相似值有相当大概率大于负样本对间的相似值, 进行样本对分类时的效果更佳. 同时, HGMSim 在检索评价类指标上大幅度优于 Siamese_GNN, 表明局部点图匹配的嵌入方法能有效学习到不同家族间高相似样本的语义差异, 使得在检索出的高相似样本中同家族样本排名更靠前. 对比实验表明, HGMSim 能很好地捕获恶意软件间语义, 有效地度量恶意软件间相似性.

表 2 模型性能对比结果 (绿盟企业数据集)

模型名	*MRR*	*NDCG*@10	AUC
TLSH	0.589	0.400	0.811
Siamese_CNNs	0.404	0.269	0.810
Siamese_GNN	0.251	0.172	**0.920**
RGCN	0.340	0.287	0.889
MGMN	0.568	0.416	0.889
HGMSim	**0.595**	**0.508**	0.919

表 3 模型性能对比结果 (BODMAS 数据集)

模型名	*MRR*	*NDCG*@10	AUC
TLSH	0.465	0.315	0.815
Siamese_CNNs	0.294	0.203	0.824
Siamese_GNN	0.242	0.171	**0.882**
RGCN	0.335	0.268	0.875
MGMN	0.455	0.324	0.870
HGMSim	**0.471**	**0.335**	0.878

对各个模型的预处理、训练、测试耗时进行统计和对比. 预处理耗时是统计模型处理 2 500 个恶意软件的耗时, 训练耗时是统计使用 3 500 对训练样本对模型进行训练的耗时, 测试耗时是统计模型对测试集中 750 对样本进行相似性度量的耗时. 其中, Siamese_CNNs 的预处理耗时包括将样本重采样为图像的时间, Siamese_GNN 的预处理耗时包括 Radare2 提取函数调用图和 Asm2Vec 生成图节点嵌入向量的时间. MGMN、HGMSim 和 RGCN 的预处理耗时包括 IDA Pro 提取函数调用图和生成函数特征向量的时间. 所有模型的耗时对比结果如表 4 所示, 其中 TLSH 为非机器学习方法, 直接使用二进制恶意软件作为模型输入, 所以无训练耗时和预处理耗时. Siamese_CNNs 将二进制恶意软件采样为恶意软件灰度图, 没有使用耗时的反汇编操作和计算开销大的图神经网络, 因此整体的耗时较少. HGMSim 使用基于局部点图匹配的异质图嵌入方法, 虽然其时间复杂度 $O(L(|V|^2F+D|V|F^2))$ 较高, 但和不使用跨图交互的图方法相比, HGMSim 在训练和测试耗时上和 Siamese_GNN 接近, 小幅度高于 RGCN, 这验证了 HGMSim 在二进制恶意软件相似性度量任务中的适用性. 耗时差异不大的原因为: 大多数恶意软件的函数调用图较小 (绿盟数据集中, 图节点数量小于 1 000 的恶意软件占比 84.37%; BODMAS 数据集中, 图节点数量小于 1 000 的恶意软件占比为 84.76%), 同时我们优化了跨图交互的信息传递方式, 仅考虑同类型跨图节点间的信息传递, 从而有效降低了计算复杂度.

表 4 耗时对比结果

模型名	测试耗时 (s)	训练耗时 (h)	预处理耗时 (h)
TLSH	28.8	无	无
Siamese_CNNs	1.47	0.77	1.25
Siamese_GNN	43.1	7.89	28
RGCN	33.2	4.79	9.32
MGMN	40.6	9	9.32
HGMSim	48.7	7	9.32

4.2.2 消融实验

本节展示区分函数节点类型和增添跨图边使用局部点图匹配方法对 HGMSim 模型性能的提升效果. 训练如下模型, 进行消融实验.

HGMSim_zero: (1) HGMSim 模型不区分函数节点类型, 使用 struct2vec 方法聚合本图邻居信息. (2) 不考虑本图和对比图的跨图间信息聚合.

HGMSim_w/o_cross: (1) HGMSim 模型区分函数节点类型, 使用本图异质邻居聚合方法聚合本图邻居信息. (2) 不考虑本图和对比图的跨图间信息聚合.

HGMSim_GMN: (1) HGMSim 模型区分函数节点类型, 使用本图异质邻居聚合方法聚合本图邻居信息. (2) 考虑本图和对比图的跨图间信息聚合, 其信息聚合基于 GMN[20]的方法, 跨图间信息聚合时, 每个图节点都会聚合其

对比图中所有图节点的信息.

HGMSim: (1) HGMSim 模型区分函数节点类型, 使用本图异质邻居聚合方法聚合本图邻居信息. (2) 考虑本图和对比图的跨图间信息聚合, 建模添加跨图边, 其信息聚合基于本文提出的局部点图匹配方法.

消融实验结果如表 5 所示. HGMSim_w/o_cross 模型比 HGMSim_zero 模型在检索类指标上明显提升, 说明区分函数调用图的异质信息, 能显著提高同家族样本在检索出的样本中排名的位置. HGMSim_GMN 模型比 HGMSim_w/o_cross 模型在所有指标上都有提升, 表明增加跨图间的节点交互, 能更好地度量恶意软件间的相似性. HGMSim 模型比 HGMSim_GMN 模型指标高, 体现本文所提局部点图匹配方法的优势. 在充分利用函数调用图中节点异质特性的同时, 仅考虑跨图间同类型函数节点的交互, 降低跨图匹配的计算开销, 同时实现了更好的性能.

表 5 消融实验结果

模型名	*MRR*	*NDCG*@10	AUC
HGMSim_zero	0.365	0.288	0.886
HGMSim_w/o_cross	0.482	0.386	0.879
HGMSim_GMN	0.595	0.465	0.898
HGMSim	**0.595**	**0.508**	**0.919**

为了进一步解释跨图交互是如何提升模型检测性能的, 对 installmonster 家族和 fusioncore 家族的样本进行分析, 其样本 md5 值分别为 eec55f1295d0cce40bbba7be231fdddb 和 6abf24e3aa3e971e6461bcfea35fe950, 这两个样本中接近 70% 的函数是完全相同的, 其函数调用图结构相似程度很高. 未使用跨图交互的模型并未很好地区分该类样本. 使用 PCA (主成分分析) 将图神经网络每层嵌入学习后得到的 CG 函数嵌入向量从 128 维降低到 2 维, 然后使用密度散点图将降维后的 CG 函数嵌入向量进行可视化, 结果如图 6 所示.

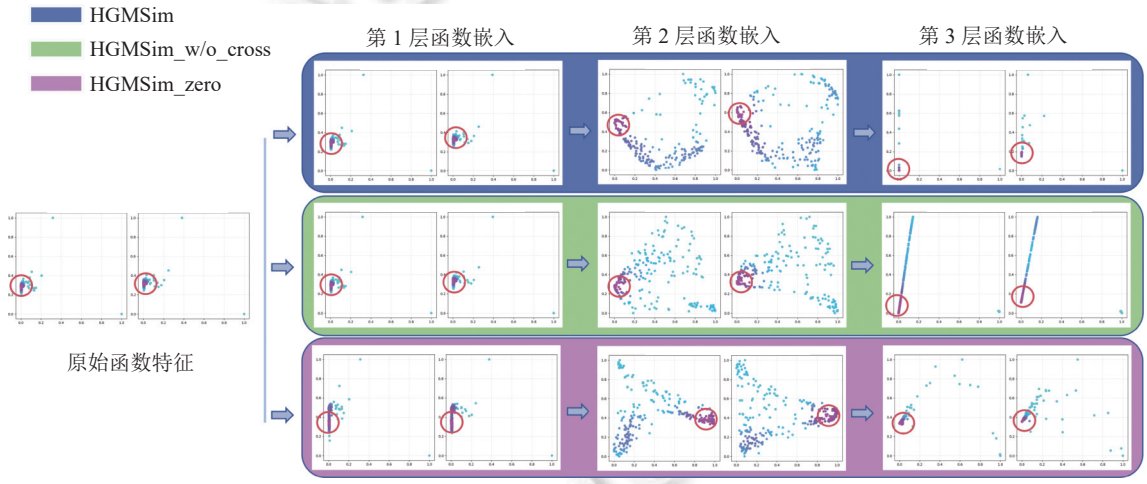


图 6 函数嵌入可视化图

图 6 由多个小图构成, 每个小图右边是 installmonster 家族样本的密度散点图, 左边是 fusioncore 家族样本的密度散点图, 粉色表示聚集节点多, 蓝色表明聚集节点少, 即粉色点聚集的区域代表着样本主要信息. 可以看到, 由于这两个恶意样本 70% 的函数特征向量都是完全相同的, 所以其原始函数特征的粉色区域都聚集在相近的坐标下. 在消融实验中, 未区分异质和跨图语义的 HGMSim_zero 模型 (粉色背景框) 和只区分了本图异质信息的 HGMSim_w/o_cross 模型 (绿色背景框), 由于相同的图结构和图节点占绝大多数, 因此在整个信息传递的过程中, 两个样本可视化图像的粉色点聚集区域位置依旧存在着较高的相似性, 并不能很好地区分这类样本. 对于 HGMSim 方法 (蓝色背景框), 在信息传递的过程中增加了跨图间的局部点图匹配, 使得粉色节点分布差异明显增大, 第 3 次函数

嵌入后, 其聚集位置的纵坐标出现了明显位移, 表明这两个不同家族样本原本相同的函数嵌入向量出现了较大的差异, 从而使模型能区分高相似的不同家族样本对.

4.2.3 不同反汇编工具对比实验

HGMSim 模型的输入依赖于使用 IDA Pro 构建的恶意软件函数调用图, 然而, 不同反汇编工具处理相同的二进制样本时, 生成的函数调用图并不相同. 在二进制代码领域准确地确定函数边界是一项极具挑战性的任务, 反汇编工具通常会使用多种启发式规则来确定函数边界, 因此导致不同工具提取出的图结构存在差异. 现有研究大多默认使用特定的反汇编工具, 而未考虑到使用不同反汇编工具对模型性能的影响. 为此, 本节使用多种主流反汇编工具提取函数调用图并重复实验, 验证使用不同反汇编工具对 HGMSim 性能的影响.

本实验基于以下 3 个原则选择反汇编工具: (1) 能处理 Windows 下 x86 的二进制文件. (2) 反汇编工具开源. (3) 安全行业和研究人员经常使用. 基于这些原则, 最终选择了 4 种工具进行实验, 分别为 IDA Pro (https://hex-rays.com/)、Angr (https://github.com/angr/angr)、RetDec (https://github.com/avast/retdec) 和 Radare2 (https://github.com/radareorg/radare2). 使用这 4 个反汇编工具对恶意软件数据集进行反汇编, 因为 Angr 和 RetDec 在预处理二进制样本时, 会出现部分样本反汇编失败的情况 (如程序无响应, 中间文件异常), 最后使用 2 022 个 4 种反汇编工具均能正确反汇编的恶意软件, 按照之前章节的方法进行模型训练和测试, 实验结果如表 6 和表 7 所示.

表 6 不同反汇编工具生成函数调用图的统计信息对比

工具名	图中平均节点数	图中平均边数	训练耗时
Angr	1 010	1 491	4:38:29
Radare2	329	736	1:12:41
IDA Pro	492	1 139	1:35:39
RetDec	651	831	2:04:25

表 7 HGMSim 模型在不同反汇编工具下的实验结果对比

工具名	*MRR*	*NDCG*@10	AUC
Angr	**0.670**	**0.557**	0.939
Radare2	0.546	0.451	0.938
IDA Pro	0.610	0.502	**0.950**
RetDec	0.567	0.510	0.931

从表 6 中展示的反汇编样本统计信息可以看到, 不同反汇编工具对于同一样本提取的图结构存在着较大差异. 其中 Angr 处理后的样本平均节点数明显大于其他样本的处理结果, 主要原因是其利用符号执行和动态路径生成, 捕获到其他工具无法识别的细节, 而 IDA Pro 和 Radare2 主要使用静态分析, 会生成较为紧凑的图结构. 图结构大小的差异会影响 HGMSim 模型的训练耗时, 训练耗时和图节点个数平方成正比.

从表 7 所展示的结果来看, 使用不同反汇编工具生成的函数调用图, 对 HGMSim 模型在检索评价指标上的影响相对较大, 在分类指标上影响相对较小. 其中 Radare2 的性能同 IDA Pro 和 Angr 相比整体表现较差, 因为 HGMSim 模型的函数调用图构建准确性依赖于反汇编指令恢复、调用边恢复等指标, 二进制反汇编领域的研究表明[30,31], 对 Windows 二进制文件反汇编, 递归工具 Radare2 在指令恢复上表现明显低于商业工具 IDA Pro 和线性工具 Angr. 由此看来, 在二进制反汇编领域表现较好的工具, 能更好地应用于恶意软件相似性度量任务. 虽然 Angr 在检索评价指标上表现最好, 但其存在两方面不足: (1) 通过代码仿真来执行控制流分析, 从而正确地反汇编二进制文件, 样本预处理时间过长. (2) 生成的样本图结构同其他反汇编工具相比明显偏大, 导致模型训练耗时增加. 综合考虑表 6 和表 7 的实验结果, IDA Pro 整体表现最好, 其利用库签名文件结合模式匹配的方法识别静态链接库函数, 能更加精准地识别混杂在一起的本地函数和静态链接库函数, 提升模型的相似性度量能力.

5 有效性威胁

本节讨论所提方法在有效性层面可能面临的潜在挑战与局限性. 重点从反静态分析威胁、数据时效性威胁以及大规模图处理瓶颈这 3 个维度对有效性威胁展开分析, 并针对性地提出应对策略.

反静态分析威胁: 恶意软件使用反静态分析技术 (如 UPX 加壳, RC4 代码加密), 导致静态分析工具无法捕获运行时解密的真实代码, 生成的函数调用图结构不完整, 影响相似性度量的可靠性. 可通过脱壳工具脱掉部分热门壳, 后续将探索动静态混合分析, 补充函数调用图的语义完整性.

数据时效性威胁: 实验使用的两个数据集样本均采集于 2021 年前, 然而恶意软件家族变种迭代速度快, 导致模型对新恶意软件的相似性度量能力可能会明显下降. 计划从 MalwareBazaar 网站持续获得新样本, 并设计增量学习机制, 动态更新模型参数.

大规模图处理瓶颈: 由于 HGMSim 的复杂度较高, 当函数调用图结构过大 (如图节点数量超过 10^4) 时, 难以满足实时检测需求. 同时图结构过大也会对设备内存提出挑战. 后续将尝试对函数调用图结构进行优化, 在保留关键语义的前提下缩减图的大小.

6 总 结

本文提出基于异质图匹配网络的恶意软件相似性度量方法 HGMSim, 该方法充分挖掘恶意软件函数调用图中的异质语义信息, 同时提出局部点图匹配方法来处理不同家族高相似样本难以区分的问题, 提升了模型的相似性度量性能. 模型对比实验表明, 本文所提方法 HGMSim 在分类指标和检索类指标上, 同多种基线相比都有较大提高. 在效率方面, HGMSim 的测试耗时和基于图的基线方法耗时接近. 消融实验结果表明, 本文提出的函数调用图异质语义挖掘方法和函数调用图之间局部点图匹配方法的有效性. 除了有效性威胁中提及的内容, 后续工作还将从以下 2 个方面开展.

(1) 相同的源代码, 由于编译器版本、编译优化选项、编译平台的不同, 在生成的二进制文件层面会产生巨大差异, 该场景下如何有效度量恶意软件的相似性.

(2) 二进制感染型病毒通常只修改合法文件的少量信息就能达到自身目的, 从文件角度看感染前后的二进制文件具有很高的相似性, 该场景下如何区分感染前后的二进制文件.

致谢 本工作得到了北京邮电大学超算平台的支持.

References

[1] Ugarte-Pedrero X, Graziano M, Balzarotti D. A close look at a daily dataset of malware samples. ACM Trans. on Privacy and Security (TOPS), 2019, 22(1): 6. [doi: 10.1145/3291061]

[2] Sihwail R, Omar K, Ariffin KAZ. A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis. Int'l Journal on Advanced Science, Engineering and Information Technology, 2018, 8(4-2): 1662–1671. [doi: 10.18517/ijaseit.8.4-2.6827]

[3] Gu YH, Wang YF, Liu WX, Wu TJ, Meng GZ. Malware similarity measurement method based on multiplex heterogeneous graph. Ruan Jian Xue Bao/Journal of Software, 2023, 34(7): 3188–3205 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6538.htm [doi: 10.13328/j.cnki.jos.006538]

[4] Deng XY, Mirkovic J. Polymorphic malware behavior through network trace analysis. In: Proc. of the 14th Int'l Conf. on Communication Systems & Networks (COMSNETS). Bangalore: IEEE, 2022. 138–146. [doi: 10.1109/COMSNETS53615.2022.9668396]

[5] Haq IU, Caballero J. A survey of binary code similarity. ACM Computing Surveys (CSUR), 2021, 54(3): 51. [doi: 10.1145/3446371]

[6] Oliver J, Cheng C, Chen YG. TLSH—A locality sensitive hash. In: Proc. of the 4th Cybercrime and Trustworthy Computing Workshop. Sydney: IEEE, 2013. 7–13. [doi: 10.1109/CTC.2013.9]

[7] Adkins F, Jones L, Carlisle M, Upchurch J. Heuristic malware detection via basic block comparison. In: Proc. of the 8th Int'l Conf. on Malicious and Unwanted Software: "The Americas" (MALWARE). Fajardo: IEEE, 2013. 11–18. [doi: 10.1109/MALWARE.2013.6703680]

[8] Hsiao SC, Kao DY, Liu ZY, Tso R. Malware image classification using one-shot learning with Siamese networks. Procedia Computer Science, 2019, 159: 1863–1871. [doi: 10.1016/j.procs.2019.09.358]

[9] Wu LF, Xu M, Xu J, Zheng N, Zhang HP. A novel malware variants detection method based on function-call graph. In: Proc. of the 2013 IEEE Conf. Anthology. IEEE, 2013. 1–5. [doi: 10.1109/ANTHOLOGY.2013.6784887]

[10] Hu X, Chiueh TC, Shin KG. Large-scale malware indexing using function-call graphs. In: Proc. of the 16th ACM Conf. on Computer and Communications Security. Chicago: ACM, 2009. 611–620. [doi: 10.1145/1653662.1653736]

[11] Zeng ZP, Tung AKH, Wang JY, Feng JH, Zhou LZ. Comparing stars: On approximating graph edit distance. Proc. of the VLDB Endowment, 2009, 2(1): 25–36. [doi: 10.14778/1687627.1687631]

[12] Tan WH, Gao X, Li YY, Wen GQ, Cao P, Yang JZ, Li WP, Zaiane OR. Exploring attention mechanism for graph similarity learning. Knowledge-based Systems, 2023, 276: 110739. [doi: 10.1016/J.KNOSYS.2023.110739]

[13] Xu XJ, Liu C, Feng Q, Yin H, Song L, Song D. Neural network-based graph embedding for cross-platform binary code similarity detection. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. Dallas: ACM, 2017. 363–376. [doi: 10.1145/3133956.3134018]

[14] Puodzius C, Zendra O, Heuser A, Noureddine L. Accurate and robust malware analysis through similarity of external calls dependency graphs (ECDG). In: Proc. of the 16th Int'l Conf. on Availability, Reliability and Security. Vienna: ACM, 2021. 57. [doi: 10.1145/3465481.3470115]

[15] Sun HQ, Shu H, Kang F, Guang Y. ModDiff: Modularity similarity-based malware homologation detection. Electronics, 2023, 12(10): 2258. [doi: 10.3390/electronics12102258]

[16] Wang M, Interrante-Grant A, Whelan R, Leek T. COBRA-GCN: Contrastive learning to optimize binary representation analysis with graph convolutional networks. In: Proc. of the 19th Int'l Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment. Cagliari: Springer, 2022. 53–74. [doi: 10.1007/978-3-031-09484-2_4]

[17] Chen YH, Chen JL, Deng RF. Similarity-based malware classification using graph neural networks. Applied Sciences, 2022, 12(21): 10837. [doi: 10.3390/app122110837]

[18] Schlichtkrull M, Kipf TN, Bloem P, van den Berg R, Titov I, Welling M. Modeling relational data with graph convolutional networks. In: Proc. of the 15th Int'l Conf. on the Semantic Web. Heraklion: Springer, 2018. 593–607. [doi: 10.1007/978-3-319-93417-4_38]

[19] Zhang CX, Song DJ, Huang C, Swami A, Chawla NV. Heterogeneous graph neural network. In: Proc. of the 25th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. Anchorage: ACM, 2019. 793–803. [doi: 10.1145/3292500.3330961]

[20] Li YJ, Gu CJ, Dullien T, Vinyals O, Kohli P. Graph matching networks for learning the similarity of graph structured objects. In: Proc. of the 36th Int'l Conf. on Machine Learning. Long Beach: PMLR, 2019. 3835–3845.

[21] Bai YS, Ding H, Gu K, Sun YZ, Wang W. Learning-based efficient graph similarity computation via multi-scale convolutional set matching. In: Proc. of the 34th AAAI Conf. on Artificial Intelligence. New York: AAAI, 2020. 3219–3226. [doi: 10.1609/aaai.v34i04.5720]

[22] Ling X, Wu LF, Wang SZ, Ma TF, Xu FL, Liu AX, Wu CM, Ji SL. Multilevel graph matching networks for deep graph similarity learning. IEEE Trans. on Neural Networks and Learning Systems, 2023, 34(2): 799–813. [doi: 10.1109/TNNLS.2021.3102234]

[23] Zhao S, Ma XB, Zou W, Bai B. DeepCG: Classifying metamorphic malware through deep learning of call graphs. In: Proc. of the 15th EAI Int'l Conf. on Security and Privacy in Communication Networks. Orlando: Springer, 2019. 171–190. [doi: 10.1007/978-3-030-37228-6_9]

[24] Kim D, Kim E, Cha SK, Son S, Kim Y. Revisiting binary code similarity analysis using interpretable feature engineering and lessons learned. IEEE Trans. on Software Engineering, 2023, 49(4): 1661–1682. [doi: 10.1109/TSE.2022.3187689]

[25] Zhu SF, Shi JJ, Yang LM, Qin BQ, Zhang ZY, Song LH, Wang G. Measuring and modeling the label dynamics of online anti-malware engines. In: Proc. of the 29th USENIX Security Symp. USENIX, 2020. 2361–2378.

[26] Sebastián S, Caballero J. AVclass2: Massive malware tag extraction from AV labels. In: Proc. of the 36th Annual Computer Security Applications Conf. Austin: ACM, 2020. 42–53. [doi: 10.1145/3427228.3427261]

[27] Yang LM, Ciptadi A, Laziuk I, Ahmadzadeh A, Wang G. BODMAS: An open dataset for learning based temporal analysis of PE malware. In: Proc. of the 2021 IEEE Security and Privacy Workshops (SPW). San Francisco: IEEE, 2021. 78–84. [doi: 10.1109/SPW53761.2021.00020]

[28] Ding SHH, Fung BCM, Charland P. Asm2Vec: Boosting static representation robustness for binary clone search against code obfuscation and compiler optimization. In: Proc. of the 2019 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2019. 472–489. [doi: 10.1109/SP.2019.00003]

[29] Oliver J, Forman S, Cheng C. Using randomization to attack similarity digests. In: Proc. of the 2014 Int'l Conf. on Applications and Techniques in Information Security. Melbourne: Springer, 2014. 199–210. [doi: 10.1007/978-3-662-45670-5_19]

[30] Andriesse D, Chen X, van der Veen V, Slowinska A, Bos H. An in-depth analysis of disassembly on full-scale x86/x64 binaries. In: Proc. of the 25th USENIX Security Symp. Washington: USENIX, 2016. 583–600.

[31] Pang CB, Yu RT, Chen YH, Koskinen E, Portokalidis G, Mao B, Xu J. SoK: All you ever wanted to know about x86/x64 binary disassembly but were afraid to ask. In: Proc. of the 2021 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2021. 833–851. [doi: 10.1109/SP40001.2021.00012]

附中文参考文献

[3] 谷勇浩, 王翼翡, 刘威歆, 吴铁军, 孟国柱. 基于多重异质图的恶意软件相似性度量方法. 软件学报, 2023, 34(7): 3188–3205. http://www.jos.org.cn/1000-9825/6538.htm [doi: 10.13328/j.cnki.jos.006538]

作者简介

陈永威, 硕士, 主要研究领域为图神经网络, 恶意软件相似性度量.
谷勇浩, 博士, 讲师, CCF 专业会员, 主要研究领域为网络安全, 异常行为检测.
谢玉奇, 硕士, 主要研究领域为大语言模型, 恶意代码检测.
吴铁军, 研究员, CCF 高级会员, 主要研究领域为高级威胁检测, 威胁线索推理.