软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(7):3188−3205 [doi: 10.13328/j.cnki.jos.006538]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于多重异质图的恶意软件相似性度量方法*

谷勇浩[1,2,3], 王翼翡[1,2], 刘威歆[4], 吴铁军[4], 孟国柱[5,6]

[1](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)

[2](北京邮电大学 计算机学院, 北京 100876)

[3](广东省信息安全技术重点实验室 (中山大学), 广东 广州 510006)

[4](绿盟科技集团股份有限公司, 北京 100089)

[5](信息安全国家重点实验室 (中国科学院 信息工程研究所), 北京 100093)

[6](中国科学院大学 网络空间安全学院, 北京 100049)

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

摘　要: 现有恶意软件相似性度量易受混淆技术影响, 同时缺少恶意软件间复杂关系的表征能力, 提出一种基于多重异质图的恶意软件相似性度量方法 RG-MHPE (API relation graph enhanced multiple heterogeneous ProxEmbed)解决上述问题. 方法首先利用恶意软件动静态特征构建多重异质图, 然后提出基于关系路径的增强型邻近嵌入方法, 解决邻近嵌入无法应用于多重异质图相似性度量的问题. 此外, 从 MSDN 网站的 API 文档中提取知识, 构建 API 关系图, 学习 Windows API 间的相似关系, 有效减缓相似性度量模型老化速度. 最后, 通过对比实验验证所提方法 RG-MHPE 在相似性度量性能和模型抗老化能力等方面表现最好.

关键词: 恶意软件相似性; 多重异质图; 邻近嵌入; API 关系图; 模型老化

中图法分类号: TP311

中文引用格式: 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

英文引用格式: 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). http://www.jos.org.cn/1000-9825/6538.htm

Malware Similarity Measurement Method Based on Multiplex Heterogeneous Graph

GU Yong-Hao[1,2,3], WANG Yi-Fei[1,2], LIU Wei-Xin[4], WU Tie-Jun[4], MENG Guo-Zhu[5,6]

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

[2](School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China)

[3](Guangdong Provincial Key Laboratory of Information Security Technology (Sun Yat-sen University), Guangzhou 510006, China)

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

[5](State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences), Beijing 100093, China)

[6](School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Existing malware similarity measurement methods cannot accommodate code obfuscation technology and lack the ability to model the complex relationships between malware. This study proposes a malware similarity measurement method called API relation graph enhanced multiple heterogeneous proxembed (RG-MHPE) based on multiplex heterogeneous graph to solve the above problems. This

* 基金项目: 北京邮电大学中央高校基本科研业务费行动计划 (2021XD-A11-1); 国家自然科学基金 (U20B2045, U1936216); 广东省信息安全技术重点实验室开放基金 (2020B1212060078)
收稿时间: 2021-03-16; 修改时间: 2021-08-20; 采用时间: 2021-11-16; jos 在线出版时间: 2022-01-28
CNKI 网络首发时间: 2022-11-15

© 中国科学院软件研究所 http://www.jos.org.cn

method first uses the dynamic and static feature of malware to construct the multiplex heterogeneous graph and then proposes an enhanced proximity embedding method based on relational paths to solve the problem that proximity embedding cannot be applied to the similarity measurement of the multiplex heterogeneous graph. In addition, this study extracts knowledge from API documents on the MSDN website, builds an API relation graph, learns the similarity between Windows APIs, and effectively slows down the aging speed of similarity measurement models. Finally, the experimental results show that RG-MHPE has the best performance in similarity measurement performance and model anti-aging ability.

Key words: malware similarity; multiplex heterogeneous graph; proximity embedding; API relation graph; model aging

1 引 言

近年来 Windows 恶意软件数量不断增加, 主要原因是变体技术的兴起与普及, 大多恶意软件都出现过新变体, 新变体采用多态引擎和代码混淆技术创建. 当前防病毒工具多采用基于签名的方法, 为每个变体创建签名费时费力, 而且恶意软件变体可以逃避基于签名的检测. 恶意软件相似性度量方法可以检测恶意软件变体, 同时通过相似样本聚类有助于发现高价值的家族特征, 减少人工标注量. 因此, 如何准确度量 Windows 恶意软件相似性有实际应用价值.

随着软件的不断发展, 软件间关系复杂且相互影响逐渐形成软件生态系统, 恶意软件也不例外, 将恶意软件生态系统建模为恶意软件关系网络可以有效衡量恶意软件间的相似性. 传统恶意软件相似性度量方法主要是对单个恶意软件的二进制代码进行抽象分析[1−11], 并没有考虑恶意软件间的关系且易受到代码混淆技术影响. 异质图技术可以描述不同恶意软件间的复杂关系, 已经用于恶意软件检测, 但是其在恶意软件相似性度量方面还没有被系统研究. 本文将研究如何利用异质图技术对恶意软件关系网络建模, 实现有效地恶意软件相似性度量, 解决如下两方面问题.

(1) 传统异质图相似性度量方法主要是基于元路径的方法, 基于元路径的异质图相似性度量方法[12,13]无法衡量远距离结点的相似性而且需要大量专家经验定义元路径, 邻近嵌入方法 ProxEmbed [14]对于这些问题提出较好的解决思路. 但是恶意软件间的关系复杂, 两种类型结点之间可能存在多种关系 (例如: 两个恶意软件写文件与这两个恶意软件删除文件具有不同的关系语义), 构建的异质图属于多重异质图[15], 邻近嵌入方法没有区分不同类型的关系, 难以学习到恶意软件之间复杂的关系表示.

(2) 为了避免被杀毒软件查杀, 恶意软件演化过程中很多恶意软件会保持相似功能但采用不同实现方式, 导致机器学习模型的性能随时间下降, 该问题被称为模型老化问题[16]. 随着恶意软件版本演进, 恶意软件使用不同 API 实现相似功能, 模型老化问题直接影响恶意软件相似性度量效果.

本文提出基于多重异质图的恶意软件相似性度量方法 RG-MHPE. 首先, 为了避免代码混淆技术的影响, 从 PE 文件头中提取静态特征、从沙箱运行结果中提取动态特征后构建多重异质图, 对恶意软件关系网络建模; 然后, 提出基于关系路径的增强型邻近嵌入方法并实现恶意软件相似性度量. 同时, 为了缓解模型老化, 从 Microsoft developer network (MSDN) 官方网站上爬取 Windows API 参考文档, 构建 API 关系图, 从中找出 Windows API 间的相似关系, 进一步提升恶意软件相似性度量效果.

本文主要贡献如下.

(1) 利用异质图建模恶意软件间关系, 提出基于多重异质图的恶意软件相似性度量方法 RG-MHPE.

(2) 提出基于关系路径的增强型邻近嵌入方法 MHPE (multiple heterogeneous ProxEmbed), 解决邻近嵌入方法无法应用于多重异质图的问题.

(3) 利用 MSDN 官方文档构建 API 关系图并对 API 聚类, 有效缓解 MHPE 方法老化问题, 进一步提升恶意软件相似性度量效果.

(4) 在企业提供数据集上实验, 验证所提模型相对于恶意软件相似性度量方法和异质图相似性度量方法, 具有更好效果.

2 相关工作

本节从恶意软件相似性度量和异质图相似性度量两个方面展开介绍.

© 中国科学院软件研究所 http://www.jos.org.cn

2.1 恶意软件相似性度量

恶意软件相似性度量主要包括基于结构化程序序列的模型、基于结构化程序图的模型和基于动态行为的模型.

(1) 基于结构化程序序列

基于结构化程序序列的模型将二进制代码抽象为字节序列、字符串特征、反汇编后的操作码序列等. 文献 [1] 提出的 CARDINAL 方法将每个函数调用参数的数量拼接在一起, 输入 Bloom 过滤器, 构造的签名可以快速相互比较得到相似性分数. 文献 [2] 提出一种基于语义等价基本块最长公共子序列的模糊匹配方法, 该方法以基本块为元素, 利用最长公共子序列对两条路径进行语义相似性建模. 文献 [3] 对完整文件或单个基本块采用滑动窗口方法, 分别计算两个二进制文件之间或二进制文件中两个函数之间的相似性.

(2) 基于结构化程序图

基于结构化程序图的模型将二进制代码抽象为图模型, 包括控制流图、函数调用图等. 文献 [4] 为控制流程图中每个基本块提取输入-输出对作为其功能 (或标签), 然后执行图匹配方法. 文献 [5] 提出一个控制流图集相似搜索系统, 通过特征向量间的距离来衡量恶意软件相似性. 文献 [6] 提出的 discovRE 方法, 通过提取轻量级语法级别的特征进行预过滤, 提高搜索效率. 文献 [7] 提出简化的指令依赖图 (reductive instruction dependency graph, RIDG), RIDG 用图结构描述指令间的依赖关系, 并间接描述程序间相似性. 上述方法是基于图匹配的方法, 近年来又提出基于图嵌入的相似性度量方法. 文献 [8] 提出一种跨平台二进制漏洞搜索器 Vulseeker, 通过构造标记语义流图, 采用余弦距离度量两个二元函数的相似性. 文献 [9] 提出 Genius 方法, 通过二部图匹配量化属性控制流图的相似性. 文献 [10] 提出 Gemini 方法, 使用图嵌入方式将控制流图转化为向量, 采用向量间余弦相似度衡量恶意样本间的相似性. 文献 [11] 将恶意软件二进制文件表示为图像, 同家族的样本在图像中表现相似.

(3) 基于动态行为

基于动态行为的模型, 通过动态运行恶意样本获取行为特征, 进行相似性度量. 文献 [17] 关注 API 调用序列的动态分析, 使用 LD 算法衡量 API 序列间的距离作为相似度分数, 实现恶意软件检测. 文献 [18] 执行恶意代码, 统计程序内部汇编指令转移概率, 得到汇编指令的全马尔科夫图, 然后使用图核函数度量两个指令转移图的相似度. 文献 [19] 将恶意软件运行时系统调用序列构建为系统调用依赖图, 使用同类样本的系统调用依赖图构建组关系图 (组关系图描述了不同组系统调用之间的关系), 采用 SaMe-NP 算法衡量未知样本和已知家族之间的相似性. 文献 [20] 提出综合考虑动态指令基本块集合的语义特征和控制流图的结构特征的程序相似性分析方法, 从语义和结构两个维度对恶意程序相似性进行分析, 具有较高的准确度和可靠性. 文献 [21] 提取字符串名命特征、注册表变化特征、API 函数调用序列特征作为恶意代码行为指纹, 通过指纹匹配算法计算不同恶意代码之间的相似性度量, 分析恶意软件是否是已知样本的变种.

2.2 异质图相似性度量

异质图模型已经用在安卓恶意软件检测 [22,23] 和 Windows 恶意软件检测 [24,25], 但是其在恶意软件相似性度量方面还没有被系统的研究.

异质图已经被广泛应用于相似性度量等数据挖掘问题 [26]. 传统异质图相似性度量方法主要基于元路径, 这类方法考虑连接两个结点的元路径种类, 不同元路径表现出不同的语义信息. 文献 [12] 提出 PathSim 方法, 衡量同类结点间基于对称路径的相似性, 文献 [27] 提出有约束的随机游走模型 PCRW, 度量文献异质图中结点的相似性. 文献 [13] 提出 HeteSim 方法, 衡量任意元路径下的相似性. 基于元路径的相似性度量方法仅适用于衡量两个相近结点间的相似性, 而且元路径的选择需要专家经验.

为解决上述问题, 文献 [28] 将深度学习和强化学习结合, 自动发现结点之间体现相似性的路径. 文献 [29] 提出了 HowSim 方法, 通过引入衰减图的概念, 将 SimRank 扩展到异质图场景下, 它具有无元路径的特性, 并同时捕获结构和语义相似性. 文献 [14] 提出邻近嵌入的概念, 将结点间的路径作为结点间的网络结构, 通过将路径嵌入成向量衡量结点间相似性. 文献 [30–32] 认为结点之间的路径只是线性序列, 对于复杂的网络结构其表征能力有限, 分别利用更复杂的结构 (交互路径 [30]、有向无环图 [31]、子图增强路径 [32]) 建模结点间的网络结构. 但是, 这些

© 中国科学院软件研究所 http://www.jos.org.cn

方法都没有解决邻近嵌入无法应用于多重异质图的问题.

3 RG-MHPE 方法

基于多重异质图的恶意软件相似性度量方法 (RG-MHPE) 架构如图 1 所示, 包括异质图建模和相似性度量两部分. 本文首先将恶意软件关系网络建模成多重异质图, 然后在多重异质图中度量恶意软件间的相似性. 在异质图建模阶段, 提取恶意 PE 文件的动静态特征构建多重异质图, 为了防止恶意软件演进过程中使用不同 API 实现相似功能导致模型性能下降, 利用 MSDN 官方文档建模 API 关系图并对 API 聚类, 优化静态特征. 在相似性度量阶段, 为了解决邻近嵌入无法应用到多重异质图的问题, 提出关系路径的概念改进邻近嵌入方法, 然后利用嵌入向量计算两结点之间的相似度. 后续章节将分别介绍异质图建模、相似性度量、API 关系图建模及 API 聚类等.

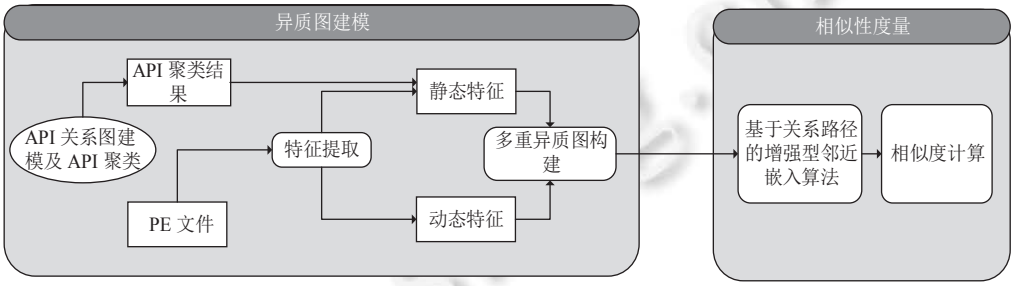


图 1 RG-MHPE 方法框架图

3.1 异质图建模

本节介绍如何从恶意软件中提取合适的动静态特征并构建多重异质图.

3.1.1 异质图定义

为了方便异质图建模工作的介绍, 首先给出异质图定义.

定义 1. 异质图 (heterogeneous graph). 一种包含多种类型实体 (结点) 和关系 (边) 的图, 记为 $G=(V, E)$, 其中 V 和 E 分别表示实体和关系, 用 A 表示实体类型集合, R 表示关系类型集合, 要求 $|A|>1$ 或者 $|R|>1$. 图 G 的网络模式 (network schema) 定义在实体类型 A 和关系类型 R 之上, 记为 $TG=(A, R)$.

图 2(a) 是异质图的一个实例, 图 2(b) 则是图 2(a) 的网络模式. 从网络模式可以看出该异质图包括 4 种类型的实体 (PE 文件、API、文件、域名) 和 3 种类型的关系 (PE 文件调用某个 API, PE 文件访问某域名, PE 文件写某文件).

定义 2. 多重异质图 (multiplex heterogeneous graph). 在异质图 G 中, 至少存在两种类型的结点 $a_i, a_j \in A$, 它们之间的关系类型数量大于 1, $\left|R\left(a_i, a_j\right) \in R\right| > 1$.

图 2(c) 是多重异质图的一个实例, 两相邻结点之间有多种关系 (PE 文件和文件之间即有写的关系, 又有删除的关系), 图 2(d) 是图 2(c) 的网络模式, 相对于图 2(b) 的网络模式, PE 文件和文件之间除了“写”关系, 还多了“删除”关系.

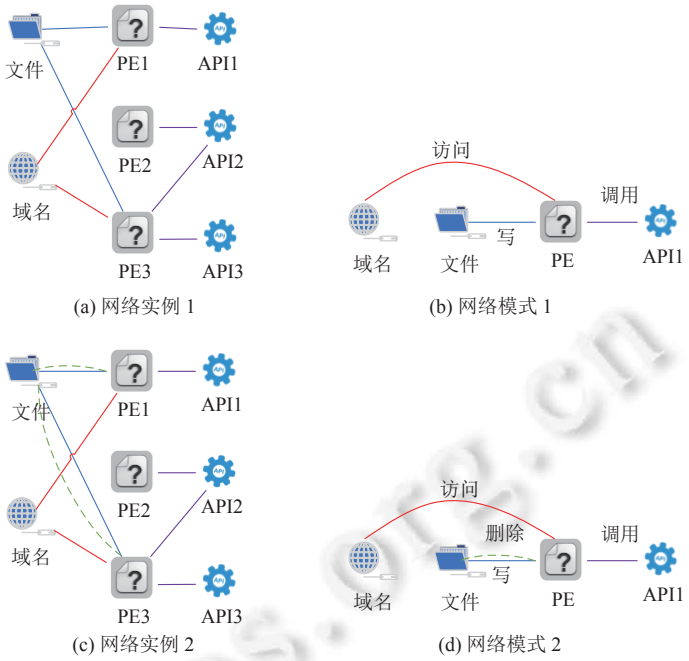
3.1.2 特征提取

为了构造能表达恶意软件之间复杂关系的异质图, 采用静态特征和动态特征相结合的方式进行恶意软件相似性度量.

(1) 静态特征. 为了避免代码混淆对特征提取的影响, 本文从 PE 文件导入表中提取 Windows API[24]调用列表, 利用第 3.3 节提到的 API 关系图从 API 中得到 API 簇作为特征.

(2) 动态特征. 恶意软件动态行为不受代码混淆的影响. 本文将恶意样本上传到 VirusTotal (https://www.virustotal.com/gui/), 从返回的沙箱报告中提取文件活动、网络活动、注册表活动等作为动态特征. 其中, 文件活动包括 PE 文件对文件的读、写、删除、复制、移动、打开, 网络活动包括 PE 文件访问某 IP 或请求某域名, 注册表活动包括 PE 文件设置或删除注册表项.

© 中国科学院软件研究所 http://www.jos.org.cn



(a) 网络实例 1　(b) 网络模式 1

(c) 网络实例 2　(d) 网络模式 2

图 2　PE 文件访问行为异质图示例

3.1.3　多重异质图构建

利用 PE 文件及其动静态特征构建的异质图中包括 8 种类型的实体和 13 种类型的关系, 其中两结点之间可能会有多种类型的关系, 构建的异质图属于多重异质图, 图 3 为构建多重异质图的网络模式.

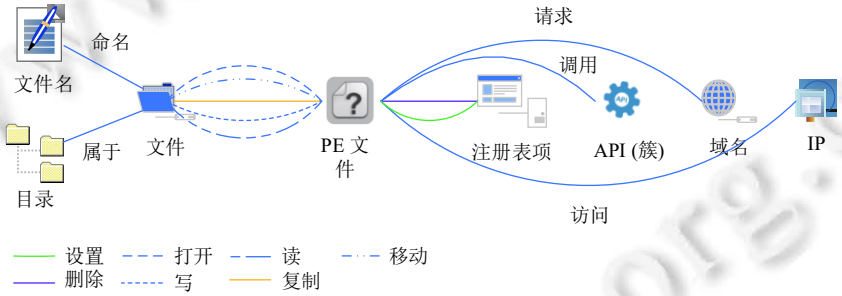


图 3　多重异质图网络模式

实体包括文件名、文件、目录、注册表项、IP、域名、PE 文件、API 簇. 关系包括:

(1) PE 文件调用 API 簇.

(2) PE 文件访问 IP.

(3) PE 文件请求域名.

(4) PE 文件读文件.

(5) PE 文件写文件.

(6) PE 文件移动文件.

(7) PE 文件删除文件.

(8) PE 文件打开文件.

(9) PE 文件复制文件.

(10) PE 文件设置注册表项.

© 中国科学院软件研究所 http://www.jos.org.cn

(11) PE 文件删除注册表项.

(12) 文件属于目录.

(13) 文件名命名文件.

需要强调一点: 第 3.3 节 API 关系图建模及聚类过程已经包含 API 与 DLL 之间的关系, 所以图 3 中采用 API 簇作为结点; 但是, 后续实验的部分基准方法中异质图建模方法不采用 API 关系图建模及聚类, 则需要增加 DLL 结点以及 API 属于 DLL 的关系 (详见第 4.2.3 节).

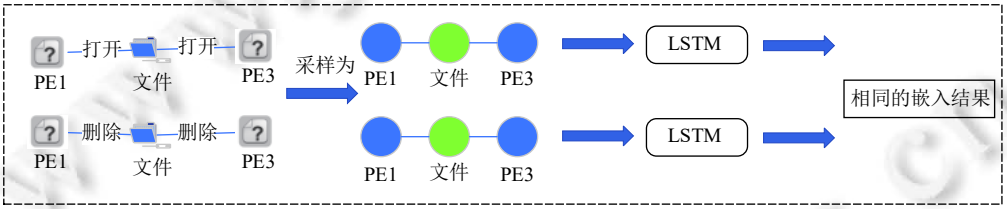
3.2 相似性度量

本节先描述临近嵌入方法的局限性, 然后介绍基于关系路径的增强型邻近嵌入方法.

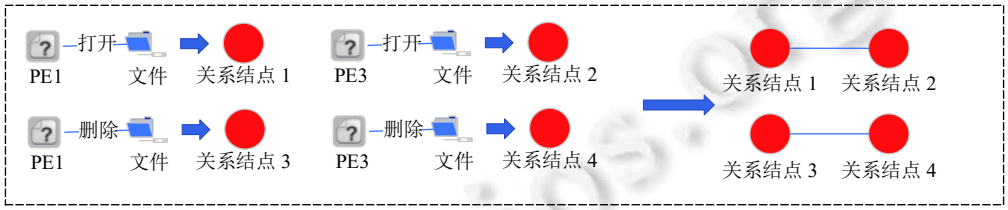
3.2.1 临近嵌入的局限性

异质图相似性度量方法, 是通过构建异质图描述恶意软件关系网络, 进行恶意软件相似性度量. 邻近嵌入方法[14]具有不需要专家经验、可以度量远距离结点相似性的优点, 在异质图相似性度量领域取得一定的成功. 但是在恶意软件关系网络中, 两结点之间可能有多种类型的关系 (图 4 中, PE1 和 PE3 之间既有删除的关系, 又有打开的关系), 这样的异质图属于多重异质图, 邻近嵌入利用结点序列作为两个结点之间的路径, 无法在多重异质图中度量结点间的相似性. 邻近嵌入方法在对异质图路径采样时, 得到 PE1 和 PE3 之间的两条路径 (如图 4(a) 所示). 这两条路径在该方法中, 都将结点序列输入 LSTM 模型得到相同的路径嵌入向量, 嵌入结果忽略了结点间不同关系的语义差异.

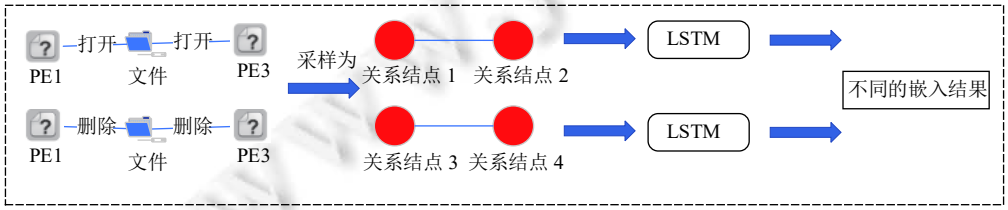
为了解决该问题, 本文提出关系结点的概念, 关系结点在考虑结点自身的同时也考虑了结点间的关系, 将相邻结点和它们之间的关系结合构建关系结点, 多个关系结点序列构成关系路径, 如图 4(b) 所示. 针对图 4 中 PE1 和 PE3 之间不同的关系, 构建各自的关系结点序列 (关系路径) 输入 LSTM 模型, 得到不同的路径嵌入向量, 表征不同的关系语义, 如图 4(c).



(a) 邻近嵌入实例



(b) 关系结点及关系路径



(c) 基于关系路径的邻近嵌入实例

图 4 邻近嵌入与基于关系路径的邻近嵌入

© 中国科学院软件研究所 http://www.jos.org.cn

3.2.2 相关定义

定义 3. 异质图中的路径 (path) 是异质图中结点的序列, 标记为 $v_1 \to v_2 \to \ldots \to v_t$, 其中 $v_i \in V$, t 是路径长度.

定义 4. 异质图中的关系结点是将异质图中两相邻结点以及它们之间某类型的关系结合后形成的结点, 记作 $n = \left(v_i, e_{i,j}^r, v_j\right)$, 其中 $v_i, v_j \in V$, $e_{i,j}^r \in E_r$, E_r 包含所有类型为 $r \in R$.

定义 5. 异质图中的关系路径是异质图中关系结点的序列, 标记为 $n_1 \to n_2 \to \ldots \to n_t$, 其中 $n_i \in N$, N 为所有关系结点的集合, t 是关系路径的长度.

3.2.3 基于关系路径的增强型邻近嵌入方法

利用异质图描述 PE 文件之间语义关系, 将恶意软件的相似性度量问题转化为异质图中结点相似性度量问题. 为了度量第 3.1.3 节多重异质图中结点之间的相似性, 提出基于关系路径的增强型邻近嵌入方法 MHPE (multiple heterogeneous ProxEmbed). 具体步骤: 首先对多重异质图进行关系路径采样, 生成关系路径, 再利用深度学习序列模型 (如: LSTM) 得到两结点之间关系路径的嵌入向量, 然后融合不同关系路径的嵌入向量. 为了表述方法方便, 先将图 3 简化为 4 类实体和 4 种关系, 得到如图 5 所示的基于关系路径的增强型邻近嵌入方法流程, 其中 (1) 表示关系路径采样, (2) 表示单条路径嵌入表示, (3) 表示多路径嵌入融合.

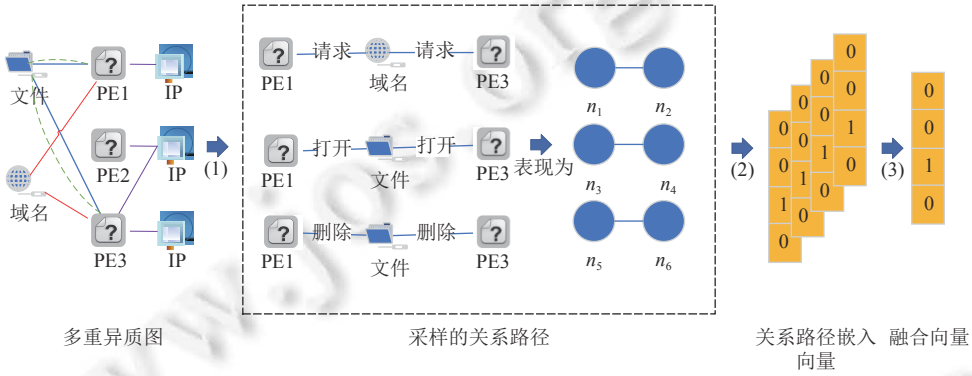


图 5 基于关系路径的增强型邻近嵌入方法

(1) 关系路径采样

对多重异质图 G 进行关系路径采样, 是从 G 中每个结点开始进行 λ 次、长度为 γ 的随机游走采样, 同时对结点序列及结点间不同类型的关系采样, 得到一组关系路径 P.

对于要进行相似性度量的请求结点 q 和目标结点 v, 从中 P 提取结点 q, v 之间的子关系路径, 这些子关系路径对应特定的关系结点序列. 将从结点 q 开始到结点 v 结束的子关系路径集合定义为 $P(q,v)$, 从结点 v 到结点 q 的子关系路径集合定义为 $P(v,q)$, 取 $Q(q,v) = P(q,v) \cup P(v,q)$ 作为两结点之间的关系路径集合.

(2) 单条路径嵌入表示

对结点之间的关系路径 $s \in Q(q,v)$ 进行嵌入表示, 先将关系路径 s 中的关系结点 $\{n_i : i = 1, \ldots, t\}$ 序列输入单层 LSTM, 再利用最大池化层结合各时间段的输出, 得到路径 s 的嵌入表示:

$$h_s = maxPooling(\{a_i : i = 1, \ldots, t\}) \tag{1}$$

其中, a_i 为 LSTM 在时间 i 的输出, 是一个 d 维向量.

关系路径中的每个关系结点 n_i 包含两个相邻结点 v_i, v_j 和对应关系 $e_{i,j}^r$, 每个关系结点的特征是两个相邻结点特征和对应关系特征的拼接, 结点 v_i 包含 4 类特征: 1) 结点类型: 采用 k 维 one-hot 编码, k 是结点类型的数量 $|A|$. 2) 结点的度: 这是一个标量. 3) 结点邻居的分布: 是一个 k 维向量. 该向量每个维度记录结点 v_i 特定类型邻居的数量, 并对这维向量取对数 (对每个维度加 1, 避免某类邻居数量为 0 时无法取对数). 4) 邻居类型熵: 由邻居类型分布计算得到的标量. 对于关系 $e_{i,j}^r$, 选择关系的类型进行 m 维 one-hot 编码作为其特征, m 是关系类型的数量 $|R|$.

(3) 多路径嵌入融合

为了融合结点 q, v 之间不同关系路径的语义, 本文通过最大池化层将 $Q(q,v)$ 中所有关系路径的嵌入表示 h_s 融

© 中国科学院软件研究所 http://www.jos.org.cn

合, 生成整体嵌入向量 $f(q,v)$, 公式如下:

$$f(q,v)=maxPooling(\{h_s\cdot e^{-\alpha|s|}:s\in Q(q,v)\}) \tag{2}$$

为了让更短路径有更大权重, 在使用池化层时加入超参数 α, α 越大长路径的权重越低.

算法 1. MHPE.

输入: 异质图 G, 训练元组 D, 每个结点随机游走次数 λ, 随机游走长度 γ, 嵌入维度 d, 超参数 α,β,μ;
输出: 模型参数 Θ.

1. 初始化关系路径集合 $P=\phi$
2. for all $v\in V$ do
3. for $i=0:\lambda$ do
4. $P\leftarrow P\cup SampleRelationPath(G,v,\gamma)$
5. end for
6. end for
7. $B\leftarrow GenerateBatches(D)$
8. for all batch $b\in B$ do
9. Initialize loss for batch b as $L_b=0$
10. for all $q,u,v\in b$ do
11. $Q(q,v)=P(q,v)\cup P(v,q)$
12. $Q(q,u)=P(q,u)\cup P(u,q)$
13. Compute $f(q,v)$ with $Q(q,v),d,\alpha$ by Eq.2
14. Compute $f(q,u)$ with $Q(q,u),d,\alpha$ by Eq.2
15. $L_b=L_b+l(\pi(q,v),\pi(q,u))$, according to Eq.4
16. end for
17. $L_b=L_b+\mu\Omega(\Theta)$
18. update Θ based on L_b by gradient descent
19. end for
20. return Θ

3.2.4 相似度计算及模型训练

q,v 结点间的相似度 $\pi(q,v)$ 是由前述邻近嵌入向量 $f(q,v)$ 和一个 d 维参数向量 θ 相乘得到:

$$\pi(q,v)=\theta^{\mathrm{T}}f(q,v) \tag{3}$$

为了训练前述相似性度量模型, 从恶意软件关系网络对应的异质图 G 中选择训练元组集合 $D=\{(q_i,v_i,u_i):i=1,\ldots,m\}$, 其中 v_i 相比于 u_i 与 q_i 更相似, 即最大化 $\pi(q_i,v_i)$ 和 $\pi(q_i,u_i)$ 之间的差异, 使用一个排名损失函数:

$$l(\pi(q_i,v_i),\pi(q_i,u_i))=-\log\sigma_\beta(\pi(q_i.v_i)-\pi(q_i,u_i)) \tag{4}$$

其中, $\sigma_\beta(x)=\dfrac{1}{1+e^{-\beta x}}$ 为超参数.

对于训练元组集合 D, 最终的损失函数为:

$$L(\Theta)=\sum_{i=1}^{m}l(\pi(q_i,v_i)-\pi(q_i,u_i))+\mu\Omega(\Theta) \tag{5}$$

其中, Θ 是该模型训练中所有参数, 包括参数向量 θ 和 LSTM 的参数, $\Omega(\cdot)$ 是正则化项, 这里使用 L2 正则化, μ 是正则化的超参数.

训练过程的目标是使公式 (5) 最小化, 如算法 1 所示. 其中, 在第 2–6 行是关系路径采样, 第 7 行是将训练集

© 中国科学院软件研究所 http://www.jos.org.cn

分 batch 训练.

3.3 API 关系图建模及 API 聚类

随着恶意软件版本演进, 恶意软件使用不同 API 实现相似功能, 包括使用不同版本的 API, API 编码方式不同等 (例如: 恶意软件家族 gandcrab 的样本在加载 DLL 文件时, 部分样本使用 LoadLibraryW 函数, 部分样本使用 LoadLibraryA 函数, 两个函数功能相同, 采用的编码方式不同). 为了防止恶意软件演进过程中使用不同 API 实现相似功能导致模型性能下降 (模型老化), 采用 API 关系图建模及聚类方法. 首先提出 API 关系图定义, 并从 MSDN 官方文档中提取信息构建 API 关系图, 然后进行 API 嵌入表示及 API 聚类. API 关系图构建及 API 聚类流程如图 6 所示.

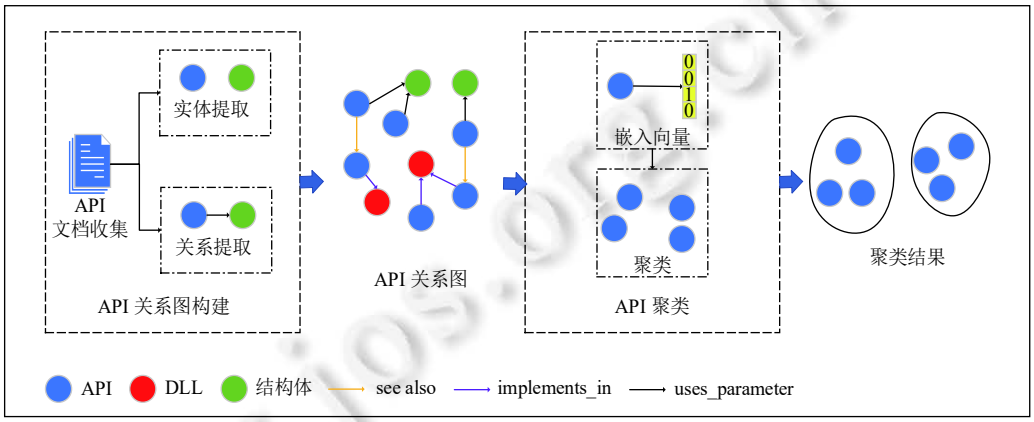


图 6 API 关系图构建及 API 聚类流程

3.3.1 API 关系图定义

定义 6. API 关系图是一个有向图, 表示为 $G=<E, R>$, 其中 E 是所有结点 (又称实体) 的集合, R 是结点间所有边 (又称关系) 的集合, API 关系图包含多种类型的实体和关系.

实体类型: 在本文构建的 API 关系图中, 有 4 种类型实体, 分别是: API (API 函数)、DLL (动态链接库)、头文件、结构体.

关系类型: 本文定义了 5 种类型的关系, 如表 1 所示, 其中包含 API 的多种描述信息, 这 5 种关系可以归纳为 3 类.

表 1 API 关系图中的关系

概括	关系	实体	举例
组织	define_in	API→头文件	CreateFileA define_in fileapi.h
	implements_in	API→DLL	CreateFileA implements_in Kernel32.dll
原型	uses_parameter	API→结构体	CreateFileA uses_parameter SECURITY_ATTRIBUTES
	returns	API→结构体	GetSidSubAuthorityCount returns SID
参考	see also	API→API	WriteFile see also ReadFile

(1) 组织类关系: 描述不同类型实体间的组织结构. 例如, define_in 描述 API 或者结构体在头文件中定义, implements_in 描述 API 在 DLL 中被调用.

(2) 原型类关系: 描述 API 函数的原型 (或声明). 例如, uses_parameter 描述 API 使用某种结构体类型作为参数, returns 描述 API 返回值的类型为结构体.

(3) 参考类关系: 描述两个实体之间的参考关系. 例如, see also 表示一个 API 参考另一个 API 的描述.

© 中国科学院软件研究所 http://www.jos.org.cn

3.3.2 API 关系图构建

API 关系图构建过程包括 API 文档收集、实体提取、关系提取 3 个阶段.

(1) API 文档收集

从 MSDN 下载微软对 Windows API 的描述性文档. Windows API 按照层级描述: 从上层的头文件, 到其中定义的函数和结构体. 每个 API 和结构体在一个 HTML 文件中描述, 包括组织结构、相关参考、函数原型等信息. 每个文档包含结构化文本和非结构化文本, 结构化文本包含组织形式、相关参考、函数原型的简略信息. 非结构化文本描述了函数的功能、要求等信息以及函数原型的详细信息.

(2) 实体提取

API 和结构体文档中描述了 API 需要的动态链接库和被定义的头文件, 可以从该文档中分别提取 API、结构体, DLL 和头文件等实体.

(3) 关系提取

1) 从结构化文本中提取关系: 这类关系可以直接通过解析 HTML 文档得到, 在提取实体过程中, 同时提取 define_in、implements_in、see also 等关系.

2) 从非结构化文本中提取关系: API 的参数和返回值等信息在非结构化文本中提取, 通过对参数的描述性文字进行字符串匹配, 找到 API 和结构体间的 uses_parameter 和 returns 关系.

3.3.3 API 聚类

API 聚类包括 API 嵌入和聚类两个阶段, API 嵌入是将 API 关系图中的 API 转化为表示其语义的向量, 使用 TransE 模型[33]实现关系图中的 API 嵌入, 再利用 K-means 算法实现 API 嵌入向量聚类, 得到 API 簇.

4 实验分析

本节使用企业提供数据集进行相似性度量实验, 从 4 个方面进行模型对比.

4.1 数据集及实验设置

为了验证模型的相似性度量性能, 先从合作企业得到 15 689 个样本报告, 筛去加壳、抗沙箱、良性样本的报告, 得到 747 个恶意软件 (https://github.com/ASUIDH/MHPE/), 时间跨度为 2012–2019 年. 该报告为 JSON 格式, 包括 PE 头和恶意软件沙箱行为等信息, 是将恶意软件上传到 VirusTotal 得到. VirusTotal 集成多种防病毒产品和文件分析工具, 包括 PEiD (PE Identifier)、pefile、Cuckoo Sandbox 等, 可以有效分析恶意软件加壳信息、PE 头信息和沙箱行为.

经统计, 本文使用的 VirusTotal 报告包含 72 个杀毒引擎的标记结果, 根据现状调研, 大部分论文选择 1–5 的整数值作为区分样本是否为恶意的阈值, 即大于等于阈值个数的杀毒引擎标记某样本为恶意, 本文将该样本标记为恶意. 本文目的是进行恶意软件相似性度量, 为了降低样本标签噪声的影响, 本文将大于等于 5 个杀毒引擎标记为恶意的样本, 打上恶意标签; 少于 5 个杀毒引擎标记为恶意的样本认为是良性的, 不参与分析. 恶意软件家族标记通过 AVclass[34]得到, AVclass 是一个利用 VirusTotal 上不同防病毒产品对恶意软件进行家族标记的开源工具. 上述 747 个恶意样本通过 AVclass 得到了 78 个家族, 通过家族信息得到恶意软件相似性关系, 即相同家族的样本相似度标记为相似, 不同家族的样本相似度标记为不相似.

为了验证模型的抗老化能力, 增加时间跨度为 2020 年第 2 季度至 2021 年第 2 季度的恶意样本. 为了保证每个时间段某个家族样本不过少, 同时参与构建异质图样本的动态行为不过少, 因此对所有样本重新筛选, 得到 2019 年及之前的样本 709 个和 2019 年之后的样本 1 581 个, 共 2 290 个恶意软件.

4.2 对比方法及预处理

4.2.1 基准方法

将所提 RG-MHPE 和 MHPE 方法与基于控制流图的恶意软件相似性度量方法[10]、基于邻近嵌入方法[14]以及

© 中国科学院软件研究所 http://www.jos.org.cn

其他相似性度量方法进行对比.

Gemini[10]首先提取恶意代码控制流图, 并将其操作码特征作为控制流图中基本块属性, 通过图神经网络为控制流图中的每个基本块学习嵌入向量, 然后将所有基本块的嵌入向量加权求和得到最终嵌入向量, 最后利用余弦距离计算相似度, 利用排名损失进行优化.

DWR[14]通过 Deepwalk[35]为每个结点学得属性向量, 然后将两结点之间的哈达玛积作为相似性嵌入, 利用排名损失进行优化.

MPP[12]利用元路径作为特征, 提出 PathSim 方法计算两个结点的相似性分数, 然后设置权重参数对所有路径的相似性分数加权求和, 利用排名损失进行优化. 其中, 元路径的生成方法参考文献 [36].

HowSim[29]首先枚举两个结点之间的路径, 并根据已知的真实结果优化衰减图参数, 然后利用衰减图迭代优化相似性矩阵, 得到任意两结点之间的相似性.

ProxEmbed[14]使用两个结点间的路径嵌入向量进行相似性度量, 该方法没有考虑不同类型的关系.

MHPE 考虑不同类型的关系, 对邻近嵌入方法 ProxEmbed 进行改进. 但是该方法仅采用单个 API 作为特征, 并没有利用 API 关系图对 API 进行聚类.

4.2.2 超参数设置

各方法的超参数设置如下.

Gemini 方法: 嵌入维度为 32, 嵌入深度为 2 层, 基本块属性选择 Block+O, 其中包括 6 个块级属性和后代的数量, 共 7 个属性 (基本块属性在 Gemini 是一个有 3 种选择的超参数).

DWR 方法: 嵌入维度 d 为 32, 路径采样为从每个结点采样次数 γ 为 20 次, 采样长度 t 为 80 的路径采样, 窗口大小 w 为 5.

HowSim 方法: 最大路径枚举长度 L 为 3, 误差边界 ϵ 为 0.01, 默认衰减图设置为 $c(r)=0.2,\ r\in R$, 训练样本对 $|\Lambda|$为 1 000 (即利用实验设置中的训练样本进行衰减图优化)

ProEmbed 方法: 嵌入维度 d 为 128, 每个结点采样次数 λ 为 20, 路径采样长度 γ 为 80, α 为 0.5, β 为 1, μ 为 0.000 1.

MHPE 方法: 嵌入维度 d 为 128, 每个结点采样次数 λ 为 20, 路径采样长度 γ 为 80, α 为 0.5, β 为 1, μ 为 0.000 3.

RG-MHPE 方法: 与 MHPE 方法相同.

4.2.3 数据预处理

Gemini 方法: 通过 Python 的二进制分析框架 angr (https://github.com/angr/angr) 生成恶意样本的控制流图和基本块的汇编代码.

DWR、MPP、ProxEmbed、MHPE 等方法: 按照第 3.1.2 节操作, 从 747 个恶意 PE 软件的 JSON 报告中获得样本的动静态特征, 构建异质图. 该异质图中不使用 API 簇作为结点, 而使用 API 作为结点, 同时增加 DLL 结点以及 API 属于 DLL 的关系.

RG-MHPE、HowSim 方法: 利用 API 参考文档构建 API 关系图, 根据肘部法则[37]对 API 聚类形成 500 个 API 簇, 按照第 3.1.2 节操作, 从 747 个恶意 PE 软件 JSON 报告中获得样本的动静态特征, 构建异质图, 构建的异质图包括 39 756 个结点和 199 330 条边.

4.3 评价指标

实验使用评价指标包括:

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

$$NDCG=\frac{DCG}{iDCG} \tag{6}$$

其中, $DCG=\sum_{i=1}^{k}\frac{rel_i}{\log_2(i+1)}$, DCG 考虑针对某样本的相似性排名中每个样本的相似性和排名位置的影响. rel_i 表示第 i 个结果的相似性, $\log_2(i+1)$ 表示第 i 个结果的位置对排名效果造成的影响, 相似性大的结果在排序列表中排名靠前则排名效果好, 相似性大的结果排名靠后则排名效果差. $iDCG$ 为理想状态下排名的 DCG, 结果按照相

© 中国科学院软件研究所 http://www.jos.org.cn

似性从大到小的顺序排序计算 DCG.

(2) 平均精度均值 (mean average precision, MAP):

$$MAP = \frac{1}{m}\sum_{i=1}^{m} AP_i \tag{7}$$

其中, MAP 为所有样本相似性排名 AP 的均值, $AP_i = \frac{1}{N_i}\cdot\sum_{r=1}^{N_i}\frac{r}{p(r)}$, 其中 i 表示第 i 个样本, m 为总样本数, N_i 为第 i 个样本具有相似样本的总数, $P(r)$ 表示结果列表从前往后看, 第 r 个相似结果在列表中的位置.

(3) Jaccard 系数:

$$J(A,B) = \frac{A\cap B}{A\cup B} \tag{8}$$

Jaccard 系数可以用来衡量两个集合 A,B 的相似性, 定义为 A 与 B 交集的大小和 A 与 B 并集大小的比值.

(4) 函数随时间变化曲线下面积 (area under time, AUT)[38]:

$$AUT(f,N) = \frac{1}{N-1}\sum_{k}^{N-1}\frac{|f(x_{k+1})+f(x_k)|}{2}\cdot(1/N) \tag{9}$$

其中, AUT 代表相似性度量函数随时间变化曲线下的面积, 用于比较时间衰减情况下不同度量函数的性能. 其中 $f(x_k)$ 为度量函数 f 在点 x_k 时的指标, 度量函数 f 在本文中为 $NDCG$ 和 MAP, x_k 是第 k 个时间窗口在图中的横坐标, N 为测试时间窗口的数量.

4.4 对比实验

本节设置 4 组实验进行模型对比, 包括模型相似性度量性能实验 (第 4.4.1 节), 模型抗老化能力实验 (第 4.4.2 节)、API 特征稳定性实验 (第 4.4.3 节) 和超参数影响实验 (第 4.4.4 节).

4.4.1 模型相似性度量性能实验

该实验参考文献 [14] 中的实验设置, 将所有 PE 文件结点分为两类, 有同类家族样本的结点为请求结点, 没有同家族样本的结点为非请求结点, 将请求结点中 20% 作为训练请求结点, 80% 作为测试请求结点. 在训练阶段, 对于每个训练请求结点 q_i, 随机选取一个相似节点 v_i 和一个不相似结点 u_i 构成训练元组 (q_i,v_i,u_i), 共选择 1 000 个元组作为训练集. 在测试阶段, 对于每个测试请求结点, 构建了一个真实排名, 该排名是根据其家族标记构建的, 即同家族的样本间相似性排名高于不同家族样本. 针对每个测试请求结点, 预测训练请求结点外的所有结点相似性排名, 并与真实排名比较, 计算排名前 10 结点的 $NDCG$ 和 MAP 来评估方法性能. 为了降低随机性, 重复 3 次实验结果取平均值.

实验结果如表 2 所示, 结果表明本文所提方法的相似性度量结果最好. RG-MHPE 和 MHPE 相似性度量结果好于控制流图方法 Gemini, 证明基于多重异质图的相似性度量方法在恶意软件领域行之有效. RG-MHPE、MHPE 和 ProxEmbed 的相似性度量结果好于 DWR, 再次证明邻近嵌入方法比单纯的结点嵌入效果好. RG-MHPE、MHPE 相似性度量结果好于 MPP, 证明基于多重异质图邻近嵌入的方法好于基于元路径的方法. RG-MHPE 和 MHPE 相似性度量结果好于 HowSim, 证明了多重异质图邻近嵌入的方法在多重异质图场景下能更有效捕获图结构信息, 进行结点相似性度量. RG-MHPE 和 MHPE 相似性度量结果好于 ProxEmbed, 证明在复杂的恶意软件关系网络中, 将恶意软件关系网络建模为多重异质图, 区分不同类型的关系, 有利于提升相似性度量结果的准确性. RG-MHPE 相似性度量结果好于 MHPE, 证明利用 API 关系图并对 API 聚类, 有利于对功能相似但实现方式不同的恶意样本进行相似性度量.

此外, 从表 2 看出使用控制流图进行相似性度量的 Gemini 结果好于 DWR、MPP、HowSim、ProxEmbed, 说明静态特征粒度更细、语义更丰富的控制流图, 在恶意软件检测领域实用效果较好. 本文为了避免代码混淆的影响, 没有使用控制流图, 只采用从 PE 文件头中导入表提取的导入 API 作为静态特征, 静态语义不够丰富, 导致表征学习能力相对有限的上述 4 个模型效果差于 Gemini, 而本文所提方法有更出色的表征学习能力, 相似性度量效果好于 Gemini.

本文所提方法针对大部分家族样本相似性度量效果较好, 但是在少部分家族样本相似性度量上相对于部分其

© 中国科学院软件研究所 http://www.jos.org.cn

他度量方法效果没有明显提升. 例如: (1) sivis 家族的部分样本会表现出非常强的特异性, 它们会读写大量的文件, 且这些文件基本都没有其他家族的样本进行读写操作. 在这种场景下, 仅依靠 PathSim 这类基于元路径进行统计的方法, 就能得到良好的相似性度量结果, 使用我们的方法进行表征学习效果并没有提升. (2) parite 家族和 agentb 家族的样本在某些动态行为方面表现很相似, 例如文件读写、网络行为. 导致我们的方法会得到这两个家族较高的相似性度量结果, 无法对这两个家族样本进行区分.

表 2 不同方法的相似性度量效果对比

模型	*MAP*	*NDCG*
Gemini	0.665	0.731
DWR	0.450	0.546
MPP	0.604	0.672
HowSim	0.633	0.702
ProxEmbed	0.591	0.671
MHPE	0.672	0.741
RG-MHPE	**0.686**	**0.753**

4.4.2 模型抗老化能力实验

为了验证 API 关系图建模及 API 聚类在模型抗老化方面的能力, 本实验使用第 4.1 节提到的 2 290 个样本. 将这 2 290 个样本按时间分为 10 组, 2019 年之前的样本为 1 组, 2019–2021 年的样本按季度分为 9 组 (2020 年第 1 季度没有样本数据), 其中 2019 年之前有 535 个样本, 2019 年及 2019 年之后共有 1 755 个样本. 然后, 对比 API 关系图建模使用前后的相似性度量结果的变化趋势. 实验使用 2019 年之前的 80% 样本进行训练, 2019 年之前剩余的 20% 样本和 2019 年至 2021 年 9 个季度的样本分别进行测试.

实验结果如图 7 所示, 相似性度量模型在未使用 API 簇时老化速度相对明显, 加入 API 关系图对 API 聚类后, 模型老化速度得到缓解, 证明利用 API 关系图对 API 进行聚类后, 模型抗老化能力好于未对 API 进行聚类. 其中, 2019 年第 1 季度实验结果较差, 因为 2019 年第 1 季度大多数样本为 MyWebSearch 样本, 这些样本在 2019 年之前并未出现. 随着 2019 年其他季度 MyWebSearch 比例逐渐减少, 相似性度量指标回升. 从 2020 年第 2 季度开始, 模型整体又处于下降趋势, RG-MHPE 比 MHPE 相对稳定. 由于缺少 2020 年第 1 季度的样本数据, 因此 *MAP* 和 *NDCG* 相似性度量的 *AUT* 指标各分成两个时间段分别计算, 最终模型的 *AUT* 指标是两个时间段 *AUT* 数值的和. *AUT* 对比结果如表 3 所示, RG-MHPE 的抗老化能力好于 MHPE.

4.4.3 API 特征稳定性实验

本实验是验证 API 关系图建模及 API 聚类对 API 特征随时间变化稳定性的影响.

首先根据同一家族恶意软件出现的时间顺序对其进行排序, 分为 5 组. 保证前面组的恶意软件出现时间要严格先于后面组, 我们称最早一组恶意软件为组 0, 然后使用 Jaccard 系数计算组 0 的恶意软件和其他组恶意软件 (组 1–组 4) 之间的 API 稳定性分数.

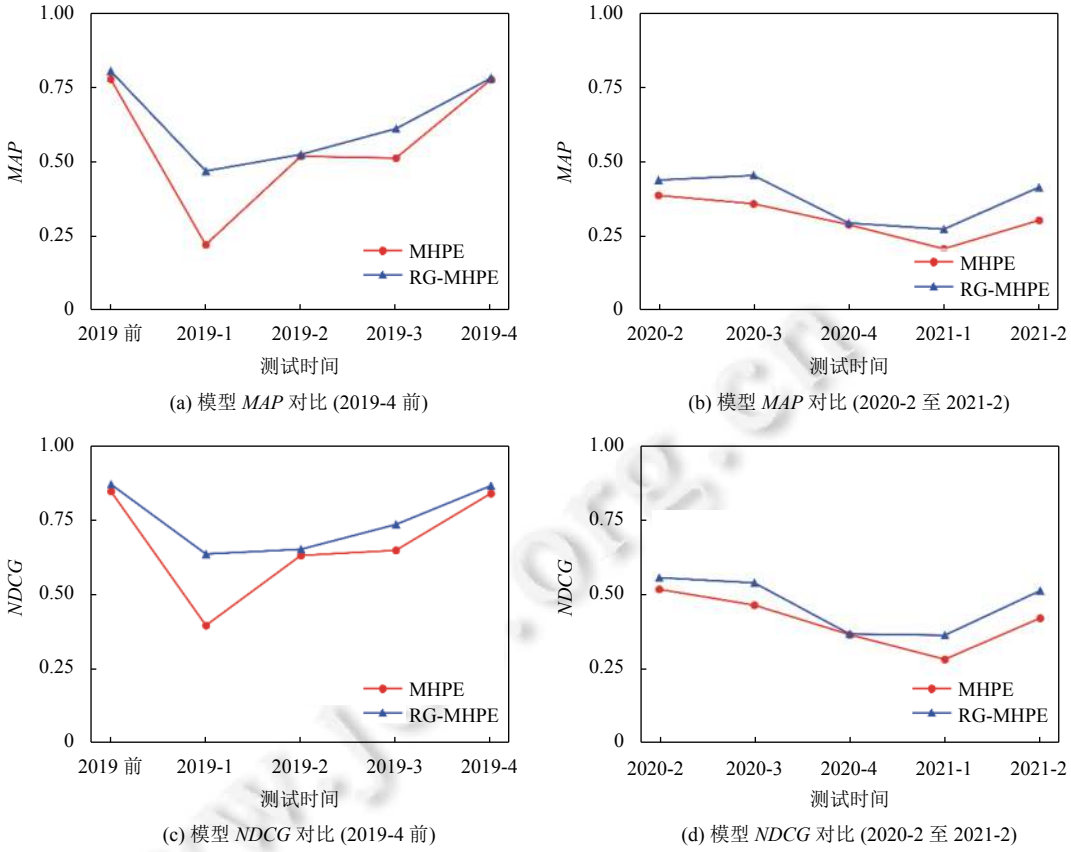
图 8 展示了 3 个恶意软件家族对 API 特征稳定性的实现效果. 图 8(a) 和图 8(b) 的结果表明, API 簇的稳定性要好于单独的 API, 其稳定性分数在 0.5 之上, 但是单独 API 的稳定性分数可能会下降到 0.3 以下. 图 8(c) 的结果表明, API 簇的稳定性随时间下降速度慢于单独的 API. 在恶意软件演化过程中, 采用 API 关系图建模可以有效提升 API 特征的稳定性.

4.4.4 超参数影响

本实验验证超参数对模型结果的影响, 改变模型的超参数 α, β, d, 观察相似性度量结果的变化.

实验结果如图 9 所示, 路径长度削减参数 $\alpha \in [0.1, 1]$ 时模型表现最佳, 如果 α 太大, 模型学习效果将变得很差. 损失函数削减参数 β=1 时模型表现最佳, 表明相似性度量模型的损失函数不需要进行太多或太少的削减. 嵌入维度 d=32 时模型表现最好, 这样的嵌入维度既能保证模型学到恶意软件之间的相似关系, 又不会产生过拟合.

© 中国科学院软件研究所 http://www.jos.org.cn



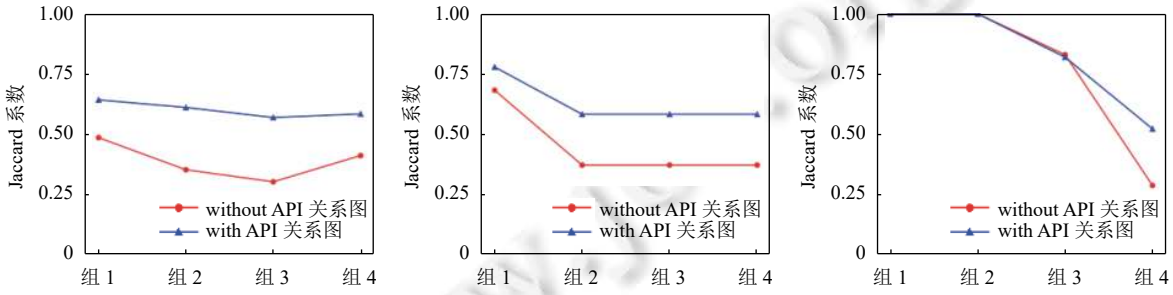
(a) 模型 *MAP* 对比 (2019-4 前)　(b) 模型 *MAP* 对比 (2020-2 至 2021-2)

(c) 模型 *NDCG* 对比 (2019-4 前)　(d) 模型 *NDCG* 对比 (2020-2 至 2021-2)

图 7　利用 API 关系图对 API 聚类前后, 模型老化速度对比

表 3　利用 API 关系图对 API 聚类前后, *AUT* 指标对比

模型	AUT_{MAP}	AUT_{NDCG}
MHPE	0.414	0.520
RG-MHPE	**0.477**	**0.581**



(a) Gandcrab　(b) Spigot　(c) Multiplug

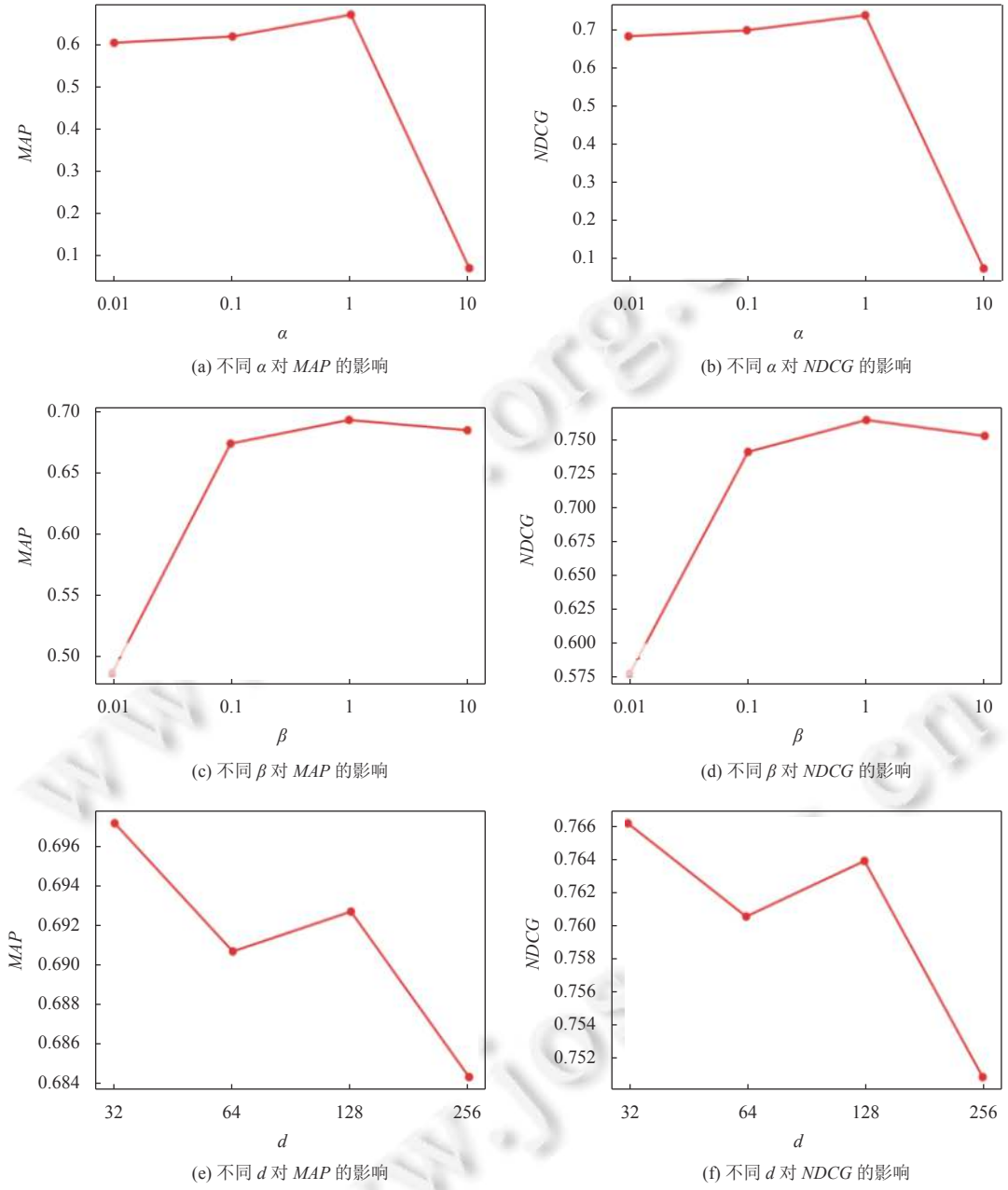
图 8　利用 API 关系图对 API 聚类前后, 不同家族 API 的 Jaccard 系数

5 结　论

本文将异质图邻近嵌入方法用于恶意软件相似性度量, 同时根据恶意软件关系网络复杂的特点, 提出多重异

© 中国科学院软件研究所 http://www.jos.org.cn

质图的邻近嵌入方法提升原始邻近嵌入方法的性能. 此外, 为了提升相似性度量模型的抗老化能力, 利用 MSDN 官方文档构建 API 关系图并对 API 聚类, 优化 API 特征. 实验结果表明: 相比现有方法, 本文所提模型 RG-MHPE 在相似性度量性能和模型抗老化能力等方面表现更好. 后续工作将从以下 3 个方面开展.



(a) 不同 α 对 *MAP* 的影响　(b) 不同 α 对 *NDCG* 的影响

(c) 不同 β 对 *MAP* 的影响　(d) 不同 β 对 *NDCG* 的影响

(e) 不同 d 对 *MAP* 的影响　(f) 不同 d 对 *NDCG* 的影响

图 9　RG-MHPE 下不同超参数取值 (α, β, d) 对相似度的影响

(1) 本文使用恶意软件动静态特征构建恶意软件关系网络, 所用动态特征是从 VirusTotal APIv2 版本 (https://developers.virustotal.com/v2.0/reference#file-behaviour) 的沙箱报告中提取的, 该报告是用 cuckoo sandbox 执行恶意样本的结果. 但是动态分析会受样本运行时间和环境的影响, 带来路径覆盖和动态特征不能完全表现等问题[39]. 为了减少上述问题对模型性能及适用范围的影响, 后续工作中一方面会尝试选择 VirusTotal 其他版本

© 中国科学院软件研究所 http://www.jos.org.cn

API, 得到更多版本的沙箱报告, 另一方面是对开源沙箱进行优化或自研发沙箱, 同时可采用较好的脱壳工具增加可用样本量.

(2) 本文利用 MSDN 官方文档构建 API 关系图并对 API 进行聚类, 增强模型的抗老化能力. 但是, Windows API 发生过多次版本演进, 本文并未考虑所有版本的 API (如: 仅考虑 C 语言的 Win32 API 并提取其中的关系, 没考虑 C++和.NET 语言的 API). 同时, 本文仅在 API 参考文档中提取了 5 种基本关系, 其他关系对模型的影响有待进一步研究.

(3) 基于关系路径的增强型邻近嵌入方法对未知恶意软件家族样本相似性度量表现不理想, 此外, 模型在训练过程中是随机选择 1 000 个训练元组, 当数据量增大、恶意软件家族种类增多时, 训练元组的随机选择可能导致结果不太稳定. 因此, 如何在不增加复杂度的基础上提升模型的稳定性, 有待进一步研究.

References:

[1] Jones L, Sellers A, Carlisle M. CARDINAL: Similarity analysis to defeat malware compiler variations. In: Proc. of the 11th Int'l Conf. on Malicious and Unwanted Software (MALWARE). Fajardo: IEEE, 2016. 1–8. [doi: 10.1109/MALWARE.2016.7888728]

[2] Luo LN, Ming J, Wu DH, Liu P, Zhu SC. Semantics-based obfuscation-resilient binary code similarity comparison with applications to software and algorithm plagiarism detection. IEEE Trans. on Software Engineering, 2017, 43(12): 1157–1177. [doi: 10.1109/TSE.2017.2655046]

[3] 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]

[4] Pewny J, Garmany B, Gawlik R, Rossow C, Holz T. Cross-architecture bug search in binary executables. In: Proc. of the 2015 IEEE Symp. on Security and Privacy. San Jose: IEEE, 2015. 709–724. [doi: 10.1109/SP.2015.49]

[5] Cesare S, Xiang Y. Malware variant detection using similarity search over sets of control flow graphs. In: Proc. of the 10th IEEE Int'l Conf. on Trust, Security and Privacy in Computing and Communications. Changsha: IEEE, 2011. 181–189. [doi: 10.1109/TrustCom.2011.26]

[6] Eschweiler S, Yakdan K, Gerhards-Padilla E. discovRE: Efficient cross-architecture identification of bugs in binary code. In: Proc. of the 23rd Annual Network and Distributed System Security Symp. San Diego: The Internet Society, 2016. 21–24. [doi: 10.14722/ndss.2016.23185]

[7] Zhang XC, Pang JM, Liu XN. Common program similarity metric method for anti-obfuscation. IEEE Access, 2018, 6: 47557–47565. [doi: 10.1109/ACCESS.2018.2867531]

[8] Gao J, Yang X, Fu Y, Jiang Y, Sun JG. VulSeeker: A semantic learning based vulnerability seeker for cross-platform binary. In: Proc. of the 33rd IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). Montpellier: IEEE, 2018. 896–899. [doi: 10.1145/3238147.3240480]

[9] Feng Q, Zhou RD, Xu CC, Cheng Y, Testa B, Yin H. Scalable graph-based bug search for firmware images. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. New York: Association for Computing Machinery, 2016. 480–491. [doi: 10.1145/2976749.2978370]

[10] 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. New York: Association for Computing Machinery, 2017. 363–376. [doi: 10.1145/3133956.3134018]

[11] Kumar N, Meenpal T. Texture-based malware family classification. In: Proc. of the 10th Int'l Conf. on Computing, Communication and Networking Technologies (ICCCNT). Kanpur: IEEE, 2019. 1–6. [doi: 10.1109/ICCCNT45670.2019.8944659]

[12] Sun YZ, Han JW, Yan XF, Yu PS, Wu TY. PathSim: Meta path-based top-K similarity search in heterogeneous information networks. Proc. of the VLDB Endowment, 2011, 4(11): 992–1003. [doi: 10.14778/3402707.3402736]

[13] Shi C, Kong XN, Huang Y, Yu PS, Wu B. HeteSim: A general framework for relevance measure in heterogeneous networks. IEEE Trans. on Knowledge and Data Engineering, 2014, 26(10): 2479–2492. [doi: 10.1109/TKDE.2013.2297920]

[14] Liu ZM, Zheng VW, Zhao Z, Zhu FW, Chang KCC, Wu MH, Ying J. Semantic proximity search on heterogeneous graph by proximity embedding. In: Proc. of the 31st AAAI Conf. on Artificial Intelligence. San Francisco: AAAI Press, 2017. 154–160.

[15] Cen YK, Zou X, Zhang JW, Yang HX, Zhou JR, Tang J. Representation learning for attributed multiplex heterogeneous network. In: Proc. of the 25th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. New York: Association for Computing

© 中国科学院软件研究所 http://www.jos.org.cn

Machinery, 2019. 1358–1368. [doi: 10.1145/3292500.3330964]

[16] Zhang XH, Zhang Y, Zhong M, Ding DZ, Cao YZ, Zhang YK, Zhang M, Yang M. Enhancing state-of-the-art classifiers with API semantics to detect evolved android malware. In: Proc. of the 2020 ACM SIGSAC Conf. on Computer and Communications Security. New York: Association for Computing Machinery, 2020. 757–770. [doi: 10.1145/3372297.3417291]

[17] Alkhateeb EMS. Dynamic malware detection using API similarity. In: Proc. of the 2017 IEEE Int'l Conf. on Computer and Information Technology (CIT). Helsinki: IEEE, 2017. 297–301. [doi: 10.1109/CIT.2017.14]

[18] Anderson B, Quist D, Neil J, Storlie C, Lane T. Graph-based malware detection using dynamic analysis. Journal in Computer Virology, 2011, 7(4): 247–258. [doi: 10.1007/s11416-011-0152-x]

[19] Nikolopoulos SD, Polenakis I. A graph-based model for malware detection and classification using system-call groups. Journal of Computer Virology and Hacking Techniques, 2017, 13(1): 29–46. [doi: 10.1007/s11416-016-0267-1]

[20] Ren YC, Xiao D. Similarity analysis of malicious programs based on two dimensional characteristics of programs. Computer Engineering and Applications, 2021, 57(1): 118–125 (in Chinese with English abstract). [doi: 10.3778/j.issn.1002-8331.2004-0259]

[21] Zheng RF, Fang Y, Liu L. Homology analysis of malicious code based on dynamic-behavior fingerprint. Journal of Sichuan University (Natural Science Edition), 2016, 53(4): 793–798 (in Chinese with English abstract).

[22] Gu YH, Li LX, Zhang Y. Robust Android malware detection based on attributed heterogenous graph embedding. In: Xu GQ, Liang KT, Su CH, eds. Frontiers in Cyber Security (FCS). Singapore: Springer, 2020. 432–446, 2020.

[23] Ye YF, Hou SF, Chen LW, Lei JW, Wan WQ, Wang JB, Xiong Q, Shao FD. Out-of-sample node representation learning for heterogeneous graph in real-time Android malware detection. In: Proc. of the 28th Int'l Joint Conf. on Artificial Intelligence Main Track. Macao: IJCAI, 2019. 4150–4156. [doi: 10.24963/ijcai.2019/576]

[24] Fan YJ, Hou SF, Zhang YM, Ye YF, Abdulhayoglu M. Gotcha-sly malware!: Scorpion a metagraph2vec based malware detection system. In: Proc. of the 24th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. New York: Association for Computing Machinery, 2018. 253–262. [doi: 10.1145/3219819.3219862]

[25] Yin SN, Kang HS, Chen ZG, Kim SR. A malware detection system based on heterogeneous information network. In: Proc. of the 2018 Conf. on Research in Adaptive and Convergent Systems. New York: Association for Computing Machinery, 2018. 154–159. [doi: 10.1145/3264746.3264784]

[26] Shi C, Sun YZ, Yu PS. Research status and future development of heterogeneous information networks. Communications of the CCF, 2017, 13(11): 35–40 (in Chinese with English abstract).

[27] Lao N, Cohen WW. Fast query execution for retrieval models based on path-constrained random walks. In: Proc. of the 16th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: Association for Computing Machinery, 2010. 881–888. [doi: 10.1145/1835804.1835916]

[28] Yang C, Liu MX, He F, Zhang XK, Peng J, Han JW. Similarity modeling on heterogeneous networks via automatic path discovery. In: Proc. of the Joint European Conf. on Machine Learning and Knowledge Discovery in Databases. Ghent: Springer, 2018. 37–54. [doi: 10.1007/978-3-030-10928-8_3]

[29] Wang Y, Wang Z, Zhao ZY, Li ZJ, Jian X, Xin H, Chen L, Song JC, Chen ZH, Zhao M. Effective similarity search on heterogeneous networks: A meta-path free approach. IEEE Trans. on Knowledge and Data Engineering, 2020, 34(7): 3225–3240. [doi: 10.1109/TKDE.2020.3019488]

[30] Liu ZM, Zheng VW, Zhao Z, Li Z, Yang HX, Wu MH, Ying J. Interactive paths embedding for semantic proximity search on heterogeneous graphs. In: Proc. of the 24th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. New York: Association for Computing Machinery, 2018. 1860–1869. [doi: 10.1145/3219819.3219953]

[31] Liu ZM, Zheng V, Zhao Z, Zhu FW, Chang K, Wu MH, Yiang J. Distance-aware DAG embedding for proximity search on heterogeneous graphs. In: Proc. of the 32nd AAAI Conf. on Artificial Intelligence. New Orleans: AAAI, 2018. 2355–2362.

[32] Liu ZM, Zheng VW, Zhao Z, Yang HX, Chang KCC, Wu MH, Ying J. Subgraph-augmented path embedding for semantic user search on heterogeneous social network. In: Proc. of the 2018 World Wide Web Conf. Republic and Canton of Geneva: Int'l World Wide Web Conferences Steering Committee, 2018. 1613–1622. [doi: 10.1145/3178876.3186073]

[33] Bordes A, Usunier N, Garcia-Durán A, Weston J, Yakhnenko O. Translating embeddings for modeling multi-relational data. In: Proc. of the 26th Int'l Conf. on Neural Information Processing Systems. Red Hook: Curran Associates Inc., 2013. 2787–2795.

[34] Sebastián M, Rivera R, Kotzias P, Caballero J. Avclass: A tool for massive malware labeling. In: Proc. of the 19th Int'l Symp. on Research in Attacks, Intrusions, and Defenses. Paris: Springer, 2016. 230–253. [doi: 10.1007/978-3-319-45719-2_11]

[35] Perozzi B, Al-Rfou R, Skiena S. DeepWalk: Online learning of social representations. In: Proc. of the 20th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: Association for Computing Machinery, 2014. 701–710. [doi: 10.1145/2623330.2623

© 中国科学院软件研究所 http://www.jos.org.cn

732]
[36] Fang Y, Lin WQ, Zheng VW, Wu M, Chang KCC, Li XL. Semantic proximity search on graphs with metagraph-based learning. In: Proc. of the 32nd IEEE Int'l Conf. on Data Engineering (ICDE). Helsinki: IEEE, 2016. 277–288. [doi: 10.1109/ICDE.2016.7498247]
[37] Syakur MA, Khotimah BK, Rochman EMS, Satoto BD. Integration K-means clustering method and elbow method for identification of the best customer profile cluster. IOP Conf. Series: Materials Science and Engineering, 2018, 336(1): 012017.
[38] Pendlebury F, Pierazzi F, Jordaney R, Kinder J, Cavallaro L. TESSERACT: Eliminating experimental bias in malware classification across space and time. In: Proc. of the 28th USENIX Security Symp. Santa Clara: USENIX Association, 2019. 729–746.
[39] Moser A, Kruegel C, Kirda E. Exploring multiple execution paths for malware analysis. In: Proc. of the 2007 IEEE Symp. on Security and Privacy. Berkeley: IEEE, 2007. 231–245. [doi: 10.1109/SP.2007.17]

附中文参考文献:

[20] 任益辰, 肖达. 基于程序双维度特征的恶意程序相似性分析. 计算机工程与应用, 2021, 57(1): 118–125. [doi: 10.3778/j.issn.1002-8331.2004-0259]
[21] 郑荣锋, 方勇, 刘亮. 基于动态行为指纹的恶意代码同源性分析. 四川大学学报(自然科学版), 2016, 53(4): 793–798.
[26] 石川, 孙怡舟, 菲利普·俞. 异质信息网络的研究现状和未来发展. 中国计算机学会通讯, 2017, 13(11): 35–40.

谷勇浩(1980－), 男, 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.

吴铁军(1978－), 男, 研究员, CCF 专业会员, 主要研究领域为高级威胁检测, 威胁线索推理.

王翼翡(1996－), 男, 硕士, 主要研究领域为网络安全, 深度学习.

孟国柱(1987－), 男, 博士, 副研究员, CCF 高级会员, 主要研究领域为移动安全, 人工智能安全.

刘威歆(1987－), 男, 博士, 研究员, 主要研究领域为网络安全, 威胁情报, 人工智能.

© 中国科学院软件研究所 http://www.jos.org.cn