软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007163] [CSTR: 32375.14.jos.007163]

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

开源软件供应链漏洞威胁智能感知*

王丽敏[1], 吴敬征[1,2], 武延军[1,2], 芮志清[1,3], 罗天悦[1], 屈 晟[1], 杨牧天[1]

[1](中国科学院 软件研究所 智能软件研究中心, 北京 100190)

[2](计算机科学国家重点实验室 (中国科学院 软件研究所), 北京 100190)

[3](中国科学院大学, 北京 100049)

通信作者: 吴敬征, E-mail: jingzheng08@iscas.ac.cn

摘 要: 开源软件的繁荣推动了软件领域的蓬勃发展, 也促使以开源软件为基础的供应链开发模式的形成. 开源软件供应链本质上是个复杂的供应链拓扑网络, 由开源生态的关键元素及其关联关系构成, 其产品全球化等优势有助于提高软件行业的开发效率. 然而, 开源软件供应链也存在依赖关系复杂、传播范围广泛、攻击面暴露扩大等特点, 带来了新的安全风险. 现有的以安全漏洞、威胁情报为基础的安全管理虽然可以实现安全预警、预先防御, 但是由于漏洞威胁信息获取不及时、缺少攻击技术和缓解措施等信息, 严重影响了漏洞处理效率. 针对上述问题, 设计并实现一种针对开源软件供应链的漏洞威胁智能感知方法, 包括两部分: 1) 构建 CTI (网络威胁情报) 知识图谱, 在其构建的过程中使用到相关技术, 可以实现安全情报的实时分析与处理, 尤其提出 SecERNIE 模型以及软件包命名矩阵, 分别缓解漏洞威胁关联挖掘的问题和开源软件别名的问题. 2) 漏洞风险信息推送, 以软件包命名矩阵为基础, 构建软件包过滤规则, 实现开源系统漏洞实时过滤与推送. 通过实验验证所提方法的有效性和可用性. 实验结果显示, 相较于 NVD 等传统漏洞平台, 本方法平均感知时间最高提前 90.03 天; 在操作系统软件覆盖率上提升 74.37%, 并利用 SecERNIE 模型实现 63 492 个 CVE 漏洞与攻击技术实体之间的关联关系映射. 特别地, 针对 openEuler 操作系统, 可追踪的系统软件覆盖率达到 92.76%, 并累计感知 6 239 个安全漏洞; 同时, 还发现 openEuler 中 891 条漏洞与攻击的关联关系, 进而获取到相应的解决方案, 为漏洞处理提供了参考依据. 在真实攻击环境验证 2 种典型的攻击场景, 证明所提方法在漏洞威胁感知方面的良好的效果.

关键词: 开源软件供应链; 漏洞威胁感知; 特征表示; 知识图谱; 风险推送

中图法分类号: TP311

中文引用格式: 王丽敏, 吴敬征, 武延军, 芮志清, 罗天悦, 屈晟, 杨牧天. 开源软件供应链漏洞威胁智能感知. 软件学报. http://www.jos.org.cn/1000-9825/7163.htm

英文引用格式: Wang LM, Wu JZ, Wu YJ, Rui ZQ, Luo TY, Qu S, Yang MT. Intelligent Perception for Vulnerability Threats in Open-source Software Supply Chain. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7163.htm

Intelligent Perception for Vulnerability Threats in Open-source Software Supply Chain

WANG Li-Min[1], WU Jing-Zheng[1,2], WU Yan-Jun[1,2], RUI Zhi-Qing[1,3], LUO Tian-Yue[1], QU Sheng[1], YANG Mu-Tian[1]

[1](Intelligent Software Research Center, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

[2](State Key Laboratory of Computer Science (Institute of Software, Chinese Academy of Sciences), Beijing 100190, China)

[3](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: The prosperity of open-source software has spurred robust growth in the software industry and has also facilitated the formation of a supply chain development model based on open-source software. Essentially, the open-source software supply chain is a complex topology network, composed of key elements of the open-source ecosystem and their interrelations. Its globalized product advantages

* 基金项目: 中国科学院战略性先导科技专项 (XDA0320401); 国家自然科学基金青年项目 (62202457)
收稿时间: 2022-11-29; 修改时间: 2023-05-10, 2023-08-31,2023-12-27; 采用时间: 2023-12-30

contribute to enhancing the development efficiency of the software industry. However, the open-source software supply chain also has characteristics such as intricate dependencies, widespread propagation, and an expanded attack surface, introducing new security risks. Although existing security management based on vulnerabilities and threat intelligence can achieve early warnings and proactive defense, the efficiency of vulnerability handling is severely affected due to delays in obtaining vulnerability threat information, and the lack of attack techniques and mitigation measures. Addressing these issues, a vulnerability threat intelligence sensing method for the open-source software supply chain is designed and implemented, which includes two parts: 1) Construction of the cyber threat intelligence (CTI) knowledge graph. In the process of constructing it, relevant technologies are utilized to achieve real-time analysis and processing of security intelligence. Particularly, the SecERNIE model and the software package naming matrix are introduced to address the challenges of vulnerability threat correlation mining and open-source software alias issues, respectively. 2) Vulnerability risk information push,based on the software package naming matrix, software package filtering rules are established to enable real-time filtering and pushing of vulnerabilities in open-source systems. This study validates the effectiveness and applicability of the proposed method through experiments. Results show that, compared to traditional vulnerability platforms like NVD, the proposed method advances the sensing time by an average of 90.03 days. The coverage rate of operating system software increases by 74.37%, and using the SecERNIE model, the relationships between 63492 CVE vulnerabilities and attack technique entities are mapped. Specifically, for the openEuler operating system, the traceable system software coverage rate reaches 92.76%, with 6239 security vulnerabilities detected. This study also identifies 891 vulnerability-attack correlations in openEuler, obtaining corresponding solutions that serve as a reference for vulnerability handling. Two typical attack scenarios in a real attack environment are verified, demonstrating the efficacy of the proposed method in vulnerability threat perception.

Key words: open-source software supply chain; vulnerability threat perception; feature representation; knowledge graph; risk push

开源软件供应链是一个实际业务系统, 包括开发和运行过程中涉及的各种实体, 如开源软件上游社区、源码包、二进制包、包管理器、存储仓库、社区、基金会等. 这些实体之间按照依赖、组合等关系形成一个供应关系网络, 用于更准确地描述开源软件的构成[1]、降低开源开发成本以及提高创新速度. 与传统的软件供应链相比, 开源软件供应链采用了去中心化的开发模式, 允许全球各地的开发者自由参与. 这种开发模式促进了更广泛的创新和协作. 然而, 随着软件生产模式的变化, 开源软件供应链也同样存在许多安全风险. 例如, 复杂的开源供应链网络增加了对依赖关系的管理难度, 未经审查或未及时更新的依赖关系可能存在安全漏洞, 这影响整个软件供应链的安全性, 导致开源软件供应链安全问题频繁发生. 根据 Sonatype 发布的报告[2], 2021 年针对开源软件的供应链攻击数量增长了 650%. 开源软件供应链安全问题对开源生态造成了严重威胁, 因此, 及时识别开源软件供应链安全风险, 以保障开源生态的安全性和可持续性, 是当前亟需解决的问题. 同时, 当前的安全漏洞应急处理方式耗时极多, 同样增大安全风险被攻击的可能性. 然而, 开源系统一旦发现安全漏洞, 需要安全技术人员、系统维护人员和第三方安全事件应急服务人员协同合作, 确定事件的起因、性质和影响范围, 通过分析安全漏洞形成的原因和机制, 来确定应急处理方案. 在测试方案的可行性和影响后, 才能对风险进行处理. 这种方式极大的延长应急响应时间, 造成风险空窗期, 导致系统被攻击的可能性增大. 以 Dell 公司在 2016 年发布的调研报告[3]为例, 有 49% 的安全防御人员需要额外的时间成本和资源挖掘足够的信息来保障系统的安全. 因此, 漏洞感知后的及时防御和缓解对系统的安全防护同样至关重要.

传统的被动防御方法通常是基于已知威胁和攻击手法的反应性措施. 它依赖于安全漏洞的修复、入侵检测系统、防火墙等传统安全工具, 在被攻击的时候减缓攻击的影响. 然而, 针对开源软件供应链的安全风险, 被动防御方法已经跟不上指数增长的攻击数量和不断迭代的攻击技术. 例如, 被认为是互联网历史上破坏力最惊人的漏洞之一的 Log4Shell 漏洞, ACMS[4]通过对 Check Point Research 关于 Log4Shell 漏洞的攻击数据进行研究和分析, 发现该漏洞受到的攻击呈指数级蔓延. 在漏洞曝光后的 3 天内, 攻击次数从几千次快速增长至 800000 次, 漏洞变体在 24 h 内达到 60 个以上, 影响范围覆盖全球 60% 以上的企业网络. 然而, 传统的被动防御方法主要依赖漏洞数据库和已知漏洞的修复. 因此, 在漏洞曝光之前, 安全团队难以采取任何预防性的措施. 根据 2021 年开源安全与风险分析 (OSSRA) 报告[5]显示, 84% 的代码库至少存在一个漏洞, 每个代码库平均有 158 个漏洞. 可见, 开源软件供应链攻击产生的机会性是非常高的. 同时, 在传统的漏洞库的数据本身也存在缺失和软件别名等数据问题. Forain 等人[6]将不同数据库进行数据质量测评, 发现 NVD 在影响范围方面存在上万条数据缺失, 并且相同的软件可能采用

不同的命名规范, 甚至在不同的上下文中拥有不同的别名, 对于软件的匹配带来了挑战. 因此, 针对开源软件供应链的安全风险, 需要对威胁情报进行主动收集和分析, 采用先进的技术来预测和防范新型攻击.

本文针对开源软件供应链安全风险实时感知及处理的问题, 提出了一种开源软件供应链漏洞威胁智能感知方法, 包括两个部分: CTI 知识图谱构建和漏洞风险信息推送. CTI 知识图谱构建模块首先通过参考公开漏洞库数据结构、开源软件生态平台、社区情报、ATT&CK 框架及 STIX2.1 标准架构设计 CTI 知识图谱本体结构. 之后, 通过将结构化数据按照本体结构进行数据填充, 并通过数据抽取技术将邮件列表、威胁情报和漏洞描述等文本知识整合到知识图谱中, 实现漏洞缺失数据补全并提高漏洞时效性. 针对开源软件采用不同命名规范所造成的别名问题, 该模块还通过分析开源软件内部文件, 以获取有关开源系统中软件上游源信息, 从而提高对开源系统的软件覆盖率. 最后, 通过本文训练的安全领域语义表示模型 SecERNIE, 实现了 CTI 知识图谱子领域之间的实体映射挖掘. 漏洞风险信息推送模块基于动态更新的 CTI 知识图谱数据, 通过制定安全漏洞过滤规则, 实现对开源系统的漏洞风险实时感知. 本文的贡献如下.

(1) 将信息抽取模型应用于安全领域文本信息, 缓解了漏洞库已知漏洞信息缺失问题, 显著提高安全漏洞威胁感知时效性, 有 94.21% 的安全漏洞感知时间早于 NVD 漏洞库.

(2) 构建了安全垂直领域知识增强语言表示模型 SecERNIE, 并基于该模型实现了安全漏洞与攻击技术实体间关联映射挖掘, 有效协助安全管理员漏洞处理及防御的工作.

(3) 构建了节点规模超过 2 千万、关系规模超过 2.8 亿的 CTI 知识图谱, 实现海量安全领域信息实时分析与处理.

(4) 分析开源系统包含的软件内部文件, 构建了开源软件命名矩阵, 解决开源软件别名问题, 提高本文方法软件覆盖率.

本文第 1 节介绍研究背景和相关工作. 第 2 节介绍本文所提出的开源软件供应链漏洞威胁智能感知方法架构及实现细节. 第 3 节对本文所进行的实验进行效果验证, 并通过实例进行实际验证. 最后, 第 4 节对本文进行总结与展望.

1 背 景

开源软件供应链漏洞威胁智能感知是一种开源软件供应链风险分析方法, 旨在智能感知开源软件供应链中的漏洞威胁, 并采取相应的措施进行分析和管理. 该方法主要分为知识图谱构建和信息推送两个主要部分, 不同阶段, 可能会面临多种现实问题. 例如, 在漏洞数据的分析和处理过程中, 可能会发现数据缺失的情况, 或者可能会遇到在所有数据源中未找到所需数据的情况. 因此, 需要借助相关技术来解决这些问题. 本文方法通过使用信息抽取技术解决了数据缺失问题; 通过信息嵌入技术, 使用实体的文本描述挖掘安全漏洞与攻击技术实体之间的关联关系, 实现攻击预测. 通过对开源系统软件进行漏洞过滤, 从而实现漏洞威胁智能信息推送. 本节将对文本中所涉及的背景知识与相关技术进行介绍. 包括开源软件供应链风险分析、漏洞感知技术、信息嵌入技术和威胁关联技术这 4 个方面.

1.1 开源软件供应链风险分析技术

随着开源软件供应链的飞速发展, 越来越多的专家学者对其威胁模型进行分析并给出多类型的威胁渗入点或攻击向量. 纪守领等人[7]从开源软件供应链的不同环节出发, 将其威胁模型分为 3 个类别, 即组件和应用软件开发环节、应用软件分发环节、应用软件使用环节. 他们指出, 这 3 个环节中存在着不同的威胁攻击面.

组件和应用软件开发环节涉及开发环境构建与开发依赖的选择, 这些过程都可能受到攻击者的威胁. 例如, CISA[8]在 2019 年发布的报告中披露, SolarWinds 的软件被植入了 SUNBURST 的后门, 而在此之前, 攻击者已经通过密码猜测等技术攻陷了其云基础设施环境. 相比之下, 因为开发依赖而造成的攻击可能造成更大的影响, 例如 2018 年发现的 unacev2.dll 中的安全漏洞, 由于该组件被 WinRAR 等 220 款软件所使用, 进而传递到多个开源软件中, 使得超过 5 亿的用户面临风险[9], 这类威胁是所有的攻击面中最难被预防且造成影响最为广泛. 应用软件

分发环节是软件开发完成后向用户提供服务的一个重要的环节, 商用软件通常通过应用商店进行分发, 而开源软件则多以镜像的形式发布到开源托管平台, 由用户自行下载或部署. 无论哪种方式, 分发平台都会对软件进行验证、管控等安全检测. 然而, 仍有不少攻击者能够绕过分发平台的检测流程, 将携带威胁的软件分发到平台上. 应用软件使用环节主要涉及用户对软件的下载和运行环境的配置, 这些环节也存在着安全风险. 攻击者可能在下载过程中利用中间人攻击等技术, 诱导用户连接到恶意站点, 或者篡改软件的内容. 例如 2022 年 8 月, CHSF 医院[10]遭遇到的软件劫持事件, 导致多个存储系统、业务软件无法使用, 给大量病人带来危险.

1.2 知识图谱与漏洞感知技术

态势感知是一种能够基于多种环境因素动态洞察风险的能力, 它从多特征的角度来发现、识别和处理威胁. 该概念最早由 Endsley[11]提出, 将时间、空间及环境等要素应用于军事领域未来发展趋势的预测, 并提出了态势感知三级模型, 奠定了态势感知的研究方向. 随着网络安全的迅速发展, 网络安全态势感知逐渐成为专家学者研究的重点, 目前的研究主要集中态势提取、态势评估和态势预测等方面. Bass 等人[12]首先将态势感知应用于安全领域, 并认为通过融合多个传感器的数据形成的态势感知能力是未来入侵检测技术的关键突破点. 他提出的网络安全态势感知定义成为研究安全态势感知的基础. Jajodia 等人[13]最早进行了漏洞态势要素提取相关研究, 将漏洞影响后果及漏洞的利用条件等要素纳入漏洞评估中, 从而获取导致特定攻击目标的漏洞利用路径.

知识图谱是一种用于表示和存储知识的结构化数据模型, 它最早由谷歌提出, 用于增强其搜索引擎功能[14]. 近年来, 知识图谱相关技术得到了快速发展, 并广泛应用于各个领域中. 例如 Zheng 等人[15]将知识图谱作为一种有效的知识管理技术, 通过分析中医诊疗过程、提取中医核心概念构建本体, 并通过深度学习技术构建中医知识图谱, 为中医知识检索、数据管理等功能提供支持. Zhao 等人[16]从军事装备知识出发, 综合考虑军事装备本身的属性和军事装备在作战中应用, 通过知识体系构建、知识抽取、知识融合和知识差异解析等方法构建军事装备知识图谱, 为巡逻反潜涉及的军事装备给出基于图的知识论证. Zhang 等人[17]将知识图谱应用于农业领域, 通过知识获取、知识表示、知识存储等相关技术实现农业知识图谱的构建, 为农业信息咨询应用提供了知识基础. Fu 等人[18]将知识图谱应用于机械加工行业, 提出了一个 EDM (放电匹配) 知识图谱, 用于预测分析设备状态和设备故障特征, 有效缩短设备故障维修时间.

同样的, 知识图谱技术也大量应用于安全领域中, 例如 Zhong 等人[19]通过研究基于网络资产构建网络安全知识图谱的方法, 并在知识图谱构建过程中采用较为新颖的命名实体识别方法, 利用模式识别实现基于知识图谱的网络资产自动化扫描, 为实现自动化、智能化的网络资产普查提供基础, 也为智能渗透测试做出充分的知识准备. 除此以外, 大量安全领域专家及学者发现其特有的以三元组来表示知识的拓扑图结构及相关挖掘、推理等技术在态势感知和决策方面具有独特优势, 因此将其引入到安全领域感知研究中, 如王一琁等人[20]提出基于知识图谱的网络安全态势感知模型 KG-NSSA, 该模型在基于资产的网络安全知识图谱的基础上搭建仿真攻击场景, 并成功实现攻击挖掘、资产攻击态势理解. 安全漏洞垂直领域态势感知是网络安全态势感知的一个垂直子域, 主要目的是实现软件的安全漏洞影响范围挖掘、实现软件漏洞安全态势理解及决策, 王丽敏等人[21]通过构建安全漏洞知识图谱, 挖掘安全漏洞与软件之间的关联关系, 并通过实体抽取、关系推理等方式进行安全漏洞影响范围的补全, 实时评估安全漏洞威胁状态进而实现软件中安全漏洞态势感知.

1.3 文本表示模型

文本表示可以对一段文本的合理性概率进行评估, 对文本分类、信息挖掘等自然语言处理任务有着重要作用. 在安全领域, 文本表示模型被大量的用于威胁发现、关联关系挖掘等困难任务中. 常用的文本语言模型有两种: 统计语言表示模型和神经语言表示模型, 统计语言模型的核心是判断单词在语料库中出现的概率, 这种模型在大量文本的情况下会出现维度灾难. 神经语言模型的出现克服了维度灾难的发生, 通过词向量表示单词及其上下文.

1.3.1 统计语言表示模型

语言表示模型最初是为了计算句子合理性概率而发展起来的. 基于统计学的语言表示模型不需要通过人工指

定维度和训练语义向量, 只需要通过语料信息构建自然语言的数学模型. Firth[22]在 1957 年提出了意义语境理论, 即通过上下文的语言环境来表示一个词, 这是表示模型的起源. Deerwester 等人[23]在 1990 年提出了潜在语义分析 (LSA), 在文本中发现隐含的语义维度, 将词和文档映射到潜在语义空间, 可以有效地解决文档中出现的多个词语具有一类语义的问题, 但是由于 LSA 这种将词语映射到话题上的表示方式很难去应对一词多义的问题, 因此在 1996 年 Lund 等人[24]提出了语义存储模型 (HAL), 认为意义相近的词总是反复共同出现, 可以通过共现矩阵来表示两个词的相关性. Rohde 等人[25]在 HAL 模型的基础上进行系数归一化, 提出了 COALS 模型.

由于句子长度增加会上述模型的计算量呈指数增长, 因此基于马尔可夫假设的 *n*-gram 模型被用于长文本表示中, 它假设一个词的表示与之前的 *n* 个词有关, 通过极大似然估计来计算每个词出现的条件概率. 目前, *n*-gram 模型被广泛应用于自然语言处理相关的任务中, 例如 Raff 等人[26]将 *n*-gram 模型引入恶意软件分类的任务, 将文件视为字节序列进行 *n* 元恶意软件特征提取, 设定不同 *n* 的取值, 输入机器学习算法获取最优解, 从而提高恶意软件分类效果.

1.3.2 神经语言表示模型

语言表示模型是为了计算句子合理性概率而发展起来的, 它可以用于自然语言处理的各种任务中. 在词向量出现之前, 传统的语言表示模型都是用独热编码 (one-hot representation)[27]将文本转化为机器学习算法易于利用的形式, 但是这种方式存在维度灾难和词义丢失的问题. Hinton 等人[28]提出了分布式表示 (distributed reputation) 的方法, 用 *N* 维连续且稠密的实数向量来表示每个词, 降低了特征空间的大小, 但复杂度较高. Mikolov 等人[29,30]提出的表达模型, 利用分层 Softmax 和神经网络训练得到分布式表示的 *N* 维词向量, 保留了词向量之间一定的关联关系, 但不能解决一词多义的问题. Devlin 等人[31]提出了 BERT 模型, 利用双向 Transformers 模型和预训练语言模型实现词的动态表征, 解决了一词多义的问题, 成为自然语言处理任务表现最好的表征模型. Zhang 等人[32]提出了将知识图谱中的先验知识引入 BERT 语言模型, 将结构化实体知识与非结构化文本信息进行融合, 解决了 BERT 知识表示的不足, 但该模型复杂度高且依赖于命名实体识别模型. Sun 等人[33]提出了 ERNIE 模型, 在不改变 BERT 模型架构的前提下, 通过改进知识掩码策略增强语言表示, 实现实体关系、实体属性等知识信息的学习.

1.4 关联关系挖掘技术

关联关系挖掘也叫关系抽取, 作为知识图谱构建过程中文本挖掘和信息抽取的核心任务被广泛运用, 其目的是从结构化或非结构化文本中识别实体之间 (例如安全漏洞与攻击技术之间) 的目标关系. 本文根据是否引入学习技术, 将关联关系挖掘技术分为基于规则的关联关系挖掘方法和基于学习的关联关系挖掘方法, 并在下文中针对这两类方法进行展开介绍.

1.4.1 基于规则的关联关系挖掘技术

基于规则的关联关系挖掘技术也叫基于模板的关联关系挖掘技术, 它需要先定义好关系类别模板[34], 然后用模板匹配文本, 得到文本中的关系类别. 最早的基于规则的关系匹配算法由 Chinchor 等人[35]提出, 其方法是通过定义文本中包含的词性、属性等组合规则, 将其运用到文本关系匹配任务中. Appelt 等人[36]提出的 FASTUS 关系抽取系统, 引入了“宏”的方式, 当目标关系发生变化时, 可以通过修改“宏”的参数修改模板规则, 增加了灵活性和复用能力. 针对文本变化可能会导致匹配规则失效的问题, Yong 等人[37]提出使用基于软匹配关联规则来提高关联关系挖掘的准确性. Brin 等人[38]提出了一种半监督实体关系抽取的方法, 将已有的关系模式作为种子实例, 通过利用规则和关系列表之间的关联, 持续的总结目标关系模式.

基于规则的关联关系挖掘最早应用于消费者关系挖掘、医学等领域中. 例如 Tong 等人[39]将其与客户关系管理相结合, 实现消费精神分析与顾客关系管理. Mallik 等人[40]将其用于肿瘤预测的基因表达和全基因组 DNA 甲基化的综合分析, 使用 Apriori 算法从训练数据中生成关联规则, 挖掘出基因状态与肿瘤形成的关联关系. Srisawat 等人[41]将其应用于股票市场, 通过发现各个股之间的关系来指示个股发展趋势.

1.4.2 基于学习的关联关系挖掘技术

基于学习的关联关系挖掘技术通过大规模语料库训练关系抽取模型[42], 将关联关系挖掘问题转化为分类或

聚类的问题. 2002 年, Collins 等人[43]将卷积树核引入关系挖掘的任务, 通过计算两个语法树的相似性获取实体间存在关系的可能性. Liu 等人[44]提出了一种基于同义词编码的 CNN 结构用于关系提取. Xu 等人[45]提出了一个负采样可调整头尾实体顺序的关系抽取模型, 引入了最短依存路径特征和关系的方向性. Guo 等人[46]通过注意力机制重新分配依存树边的权值, 再将结果通过图卷积神经网络进行分类. 为了降低模型对标注数据的依赖, Chen 等人[47]通过标注传播的方式, 将已标注和未标注的数据集作为图的节点, 将两者距离作为边的权重实现标注函数.

此外, 基于学习的关联关系挖掘技术在生物工程、金融财经、医学、软件工程等领域应用广泛, 并且很多垂直领域的模型被提出. 例如 Peng 等人[48]提出 CST-Miner 方法用于挖掘生物网络中的多实体拓扑关系. Zhou 等人[49]提出一个深度学习模型 BGAJM, 该模型是一种基于 BiGru 和注意力机制的金融信息领域 BiGru 注意力联合模型, 将实体识别和关系提取作为一个整体的学习和训练, 避免了传统实体识别和关系抽取两个不同子任务引起的误差传播问题, 该模型在金融领域的关系抽取任务中取得较好的实际效果.

2 开源软件供应链漏洞威胁智能感知

开源软件供应链具有开放性特点, 虽然有利于提高开发效率和降低开发成本, 但也带来了安全风险. 与商业软件不同的是, 开源软件的风险修复往往依赖于开发人员的主动性和及时性. 如果没有对开源软件进行定期的风险检查修复, 那么产品中的安全风险就会一直存在, 并随着代码量的增长而逐年累积[50]. 现有的安全风险检测方法存在着编程语言范围受限、数据库质量不足以及时间成本高昂等问题. 这些限制使得它们难以满足对快速且准确的安全风险检测的迫切需求. 即便在一些最新发布的漏洞威胁感知系统 (例如 WIC) 中, 仍然存在对漏洞库结构化数据的依赖. 相对于非结构化情报数据, 这种依赖性带来了一定的滞后性, 导致了防御措施在时间上存在过长的空窗期. 因此, 迫切需要在安全领域引入专用语言模型, 以处理非结构化情报数据. 通过这种方式, 可以将具有高时效性的情报数据处理出来的结构化信息应用于感知系统, 实现对漏洞威胁更为深入的理解, 并能够及时推送相关的威胁信息.

本文针对开源软件供应链漏洞威胁实时感知及处理的问题, 提出了一种开源软件供应链漏洞威胁智能感知方法. 该方法的整体框架如图 1 所示, 主要包括两个部分: ① CTI 知识图谱构建, 通过知识图谱相关技术实现安全领域数据动态分析与处理, 为本文方法提供实时且全面的数据基础; ② 漏洞风险信息推送, 实时获取开源系统软件列表与 CTI 知识图谱漏洞威胁动态信息, 实现开源系统漏洞威胁智能感知.

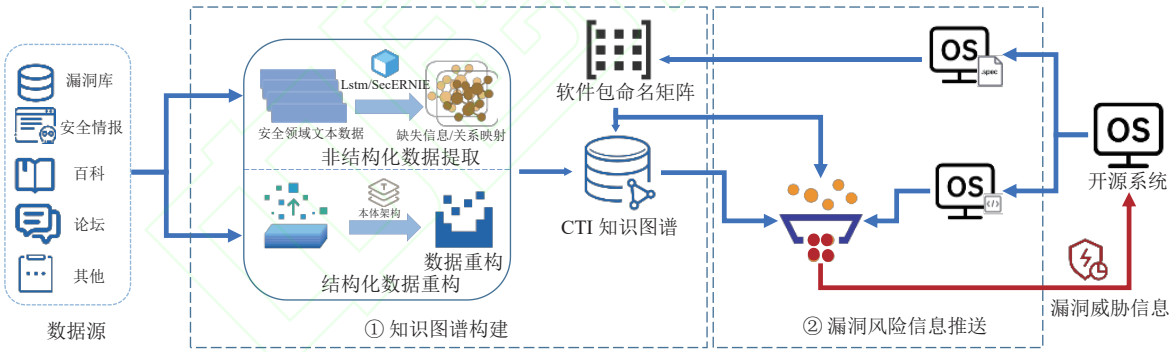

图 1 开源软件供应链漏洞威胁智能感知方法整体框架

2.1 CTI 知识图谱的构建

CTI 知识图谱是智能感知的数据基础, 通过知识图谱相关技术实现安全情报实时处理, 首先, 通过参考公开漏洞库数据结构、ATT&CK 框架及 STIX2.1 标准架构设计 CTI 知识图谱本体结构. 之后, 通过将结构化数据按照本体结构进行数据填充, 并通过数据抽取技术将邮件列表、威胁情报和漏洞描述等文本知识整合到知识图谱中, 实现漏洞缺失数据补全并提高漏洞时效性. 然后, 采集并处理不同操作系统家族的软件包上游源信息构建软件包命

名矩阵, 缓解了开源软件生态中操作系统子领域别名问题. 最后, 通过本文训练的安全领域语义表示模型 SecERNIE, 实现了 CTI 知识图谱子领域之间的实体映射挖掘.

2.1.1 CTI 知识图谱本体构建

本体结构是知识图谱的核心, 也是描述知识图谱概念层次体系的知识模板. 在构建 CTI 知识图谱本体架构的过程中, 可以明确安全类专业术语、梳理不同类型实体之间的关联关系并使其形式化, 实现一定程度的知识复用, 降低知识的冗余. 本文结合 NVD、ExploitDB、CNNVD、CWE 等大型安全相关数据库, NPM、RPM、GitHub 等开源软件生态平台, 分析大量安全公告、威胁情报等社区情报, 参考 ATT&CK 框架及 STIX2.1 标准, 设计形成一个 CTI 知识图谱本体架构. CTI 知识图谱分为漏洞风险和威胁风险两个子图, 其中漏洞风险子图中包含安全漏洞、软件、补丁、许可证等 17 类实体类型; 威胁风险子图中包含了攻击技术、恶意软件、攻击组织等 9 类实体类型, 共包含 41 类实体之间的关系. 其本体中包含的所有实体类型和关联关系如图 2 所示.

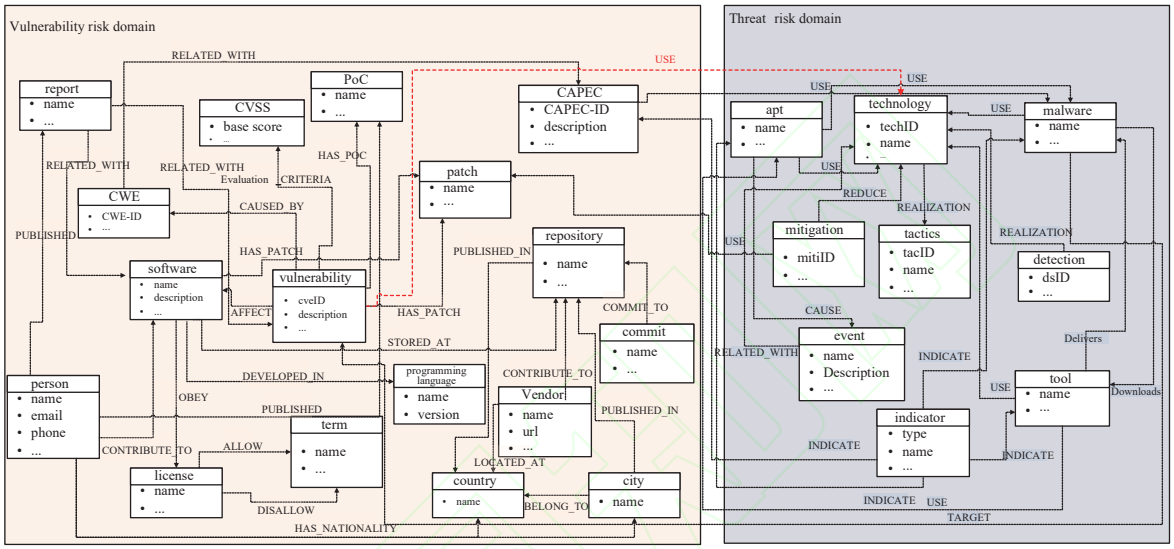

图 2 CTI 知识图谱本体结构

2.1.2 数据填充及补全

在完成知识图谱本体设计后, 本文通过获取开源软件、安全漏洞库、威胁情报、安全公告、厂商通知等不同类型数据来源, 实现 CTI 知识图谱数据融合. 本文以 NVD 漏洞库数据融合为例, 阐述 CTI 知识图谱知识实时填充及补全的过程.

2.1.2.1 安全知识实时填充

安全知识的填充是构建 CTI 知识图谱的关键, 这涉及从多个数据源中收集、整理和标准化安全领域信息, 按照 CTI 知识图谱本体架构将安全知识进行填充. 面对来自不同数据源的异构数据, 信息的融合是确保知识图谱质量和一致性的重要方式. 针对结构化数据的知识填充, 具体实现步骤:

(1) 格式规范化: 为了提高数据一致性和可用性, 从而更好地支持 CTI 知识图谱的构建和管理, 本文将所有数据字段格式规范化, 例如, 日期字段通过时间戳的形式存储; 软件名称一致进行小写化处理等.

(2) 标识符分配: 在构建 CTI 知识图谱中, 涵盖多实体类别, 包括安全漏洞、开源软件、厂商、攻击技术等, 每个实体都需要被赋予唯一的标识符. 这对知识图谱整体管理和查询操作至关重要. 本文采用 URL (统一资源标识符) 来分配标识符. 这是一种全球唯一的标识符体系, 具有标准化和良好的表达能力. 通过为每个实体类别定义一套 URL 模板, 有效地实现实体唯一性. 例如:

- 安全漏洞实体使用“https://ctigraph.com/vulnerability/{CVE 编号}”的 URI, 其中 CVE 编号是漏洞的唯一标识.

● 软件实体使用“https://ctigraph.com/software/{厂商名称/软件名称/版本号}”的 URI, 其中后缀为软件名称与厂商名称和版本号的组合, 以确保唯一性.

● 厂商实体使用“https://ctigraph.com/vendor/{厂商名称}”的 URI, 其中厂商名称是唯一的.

● 攻击技术实体使用“https://ctigraph.com/technique/{攻击技术编号}”的 URI, 以确保唯一性.

(3) 信息聚合: 将来自不同数据源却具有相同标识符的实体数据合并为一个实体, 以安全漏洞为例, 将相同漏洞的漏洞描述、漏洞等级和漏洞类别等信息进行合并.

(4) 数据冲突解决: 若信息聚合是出现数据冲突, 例如不同数据源对安全漏洞评价等级不同, 怎将冲突数据共同存储, 同时标注数据来源.

2.1.2.2 安全漏洞影响范围补全

NVD 等漏洞库中包含大量的漏洞信息, 是 CTI 知识图谱中安全漏洞节点主要数据源之一. 然而, 通过分析发现, 尽管 NVD 数据库被广泛使用, 但是其安全漏洞数据并不完全可靠, 存在漏洞影响范围信息缺失的问题. 具体而言, 漏洞描述中包含的信息没有完全被影响范围列出. 例如, 漏洞 CVE-2019-13377 的描述中写明该漏洞存在于“hostapd”和“wpa_supplicant”两个软件中, 但是该漏洞的影响范围信息中只列出了 hostapd. 上述问题在数据库中并不少见, 根据 Dong 等人[51]的统计, NVD 漏洞库中有 59.82% 的漏洞与漏洞情报或漏洞描述所包含的信息一致.

本文针对漏洞影响范围缺失问题, 采用 BERT+LSTM+CRF 模型, 抽取漏洞描述中软件名称、软件版本以及名称与版本之间的关系. 如图 3 所示, 为该模型对漏洞描述数据抽取的算法流程图. 首先通过 BERT 层处理漏洞描述文本, 获取每个文本中每个单词的词向量. 通过 BERT 层获取的词向量对处理长距离依赖信息的语句有很好的效果. 将 BERT 嵌入后得到的向量输入到 LSTM 模型中, 该核心主要是以下结构: 遗忘门、输入门、输出门以及记忆 Cell. 其中, 输入门与遗忘门两者的共同作用就是舍弃无用的信息, 把有用的信息传入到下一时刻. 对于整个结构的输出, 主要是记忆 Cell 的输出和输出门的输出相乘所得到的. 最后, 通过 CRF 可以得到单词的序列标签得分, 从而抽取到描述中的软件名称、软件版本信息. 为了识别 NVD 等漏洞库影响范围缺失的数据, 本文通过分析漏洞描述的句法结构, 构建软件名称与版本之间的映射规则, 从而抽取软件名称与版本之间的关联关系. 最后通过抽取出的软件名称及版本与漏洞已知影响范围进行匹配, 识别出漏洞库中存在数据缺失的漏洞, 最后将抽取出的信息作为补充信息添加到漏洞影响范围, 补全缺失的信息.

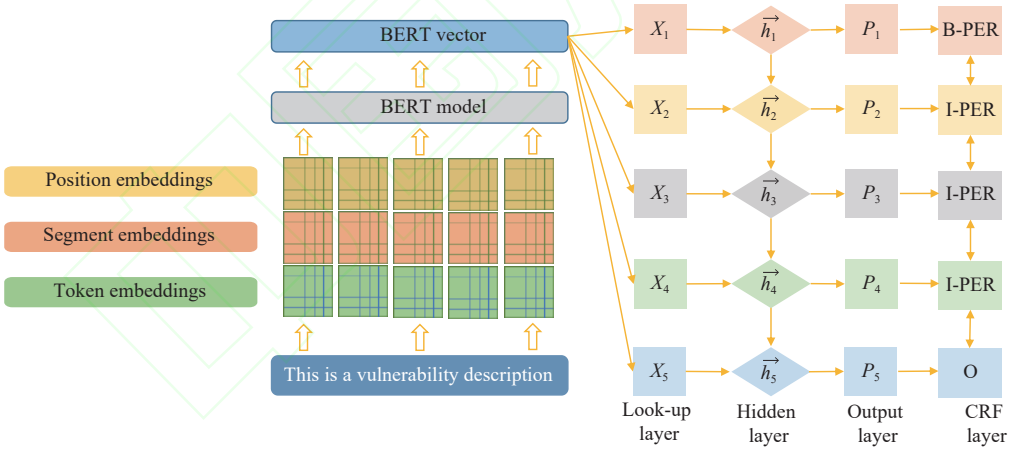

图 3 BERT+LSTM+CRF 数据抽取算法流程图

本文研究发现, 许多漏洞在获得 CVE 编号后并未立即公开披露, 这种现象涉及多种因素. 首先, 漏洞披露是一个复杂的过程, 涉及漏洞报告者、供应商、安全研究团队等多方的协调和合作. 其次, 供应商需要时间来应对漏洞. 一旦漏洞被分配 CVE 编号, 供应商需要评估漏洞的严重性, 设计和测试修复方案, 并准备发布安全补丁, 该过程需要一定的时间来完成. 最后, 一些系统组织需要遵守合规性规定, 这会影响漏洞披露时间. 因此 NVD、CNVD

等漏洞数据库发布的平均时间要晚于厂商漏洞库, 而威胁情报、邮件列表等非结构化文本漏洞数据发布会比厂商漏洞库发布时间更早.

因此, BERT+LSTM+CRF 模型除了解决数据缺失问题以外, 将其应用于威胁情报、安全公告等时效性较高的文本数据可以提高 CTI 知识图谱结构化数据的时效性, 并且该数据可直接应用于漏洞威胁智能感知. 以 Openwall 为例, Openwall 是一个小型的安全项目, 项目研究人员会独立发现安全漏洞而不依赖于其他安全漏洞库或渠道. 同时, 该项目吸引了大量从事与漏洞研究、代码审查或安全工具开发工作的志愿者, 并与 Linux 社区及各个子社区有紧密联系. Openwall 强调及时的漏洞披露, 因为他们认为这有助于提高计算机系统的整体安全性, 因此本文通过该项目邮件列表文本抽取, 相对于 NVD, 可以更早感知到安全漏洞信息. 例如, 安全漏洞 CVE-2023-0045 是一个涉及 Linux 内核的高危漏洞, 其在 NVD 于 2023 年 4 月 25 日发布. 然而, 值得注意的是, 该漏洞在早于 NVD 的时间点, 即 2023 年 2 月 3 日, 已经在 Openwall 邮件列表中正式披露. 本文通过对 Openwall 邮件列表文本信息的提取与分析, 使得 CTI 知识图谱在该漏洞披露当天感知到该漏洞的存在, 并提取其漏洞编号及影响范围信息, 将其整合到知识库中为下游信息过滤及推送任务提供数据基础, 实现了对漏洞信息的高效感知.

2.1.2.3 软件包命名矩阵的构建

开源软件的生态系统通常是分散的, 由不同的开发者或者团队独立维护. 这导致了在不同的项目中存在相同软件可能使用不同的命名规范, 进一步导致了相同的软件在不同上下文中拥有不同的别名. 例如, PostgreSQL 在命令行、脚本或非正式文档中被简称为 Postgres 或 QSql, 而在官方的文档中会被使用完整的 PostgreSQL. 获取尽可能全面的软件名称可以提高下游信息过滤及推送任务的软件覆盖率, 即可追踪到的软件安全情况的比率.

本文采集大量的软件数据源, 获取软件的别名信息并构建软件包命名矩阵来缓解开源软件生态中操作系统子领域别名问题. 在开源软件生态系统中, 开发者会进行大量的开源软件复用并进行软件包名称的更改, 导致相同软件会有不同的名称. 本文将上述被复用的原始开源软件称为上游源软件. 在针对开源软件漏洞的扫描过程中, 由于开源软件包别名问题, 很多软件包不能被追踪到, 增加了扫描系统的漏报率. Shao 等人[52]针对 Linux 操作系统分析, 将使用人数最高的前 50 类 Linux 操作系统分为 Fedora、Arch、Debian 等 6 类系统家族, 针对别名问题, 本文在不同系统的官方代码托管平台分别获取上述 6 类系的统软件包, 并添加 anolisOS 和 openEuler 两个国内系统进行综合分析, 发现在不同操作系统家族的软件包中都存在包含上游源信息的文件. 例如 centOS 和 anolisOS 中的 spec 文件包含 upstream 字段, 该字段表示了软件的上游源软件的名称, upstream 值相同的软件包来源于同一个上游源. 除此以外, Debian 系统中的 control 文件以及 ArchLinux 系统中的 build 文件都包含上述上游源信息. 本文将获取的 8 个操作系统已知软件开源文件信息, 对所有软件的上游源组件信息按照名称进行匹配, 如果匹配成功, 则将组件上游源以及上游源包含的漏洞进行映射关系的构建. 如后文图 4 所示, 漏洞 CVE-2022-41852 影响软件 commons-jxpath, 而 openEuler 发行版中包含的软件名称为 apache-commons-jxpath, 通过本文构建的软件包命名矩阵发现 apache-commons-jxpath 的上游源软件为 commons-jxpath, 因此可以判断出该漏洞同样影响到 apache-commons-jxpath.

2.1.3 漏洞威胁映射关系挖掘

在获知开源软件具有漏洞后再进行修复是一种被动的安全措施, 而开源软件供应链防御需要采取积极主动的方法, 即通过主动收集和分析威胁情报, 自动化地将安全漏洞与 ATT&CK 框架 TTP 进行映射, 使安全管理人员能够专注于漏洞修复, 并主动保护其组织免受外部和内部威胁, 这一方法增强了漏洞处理及防御工作. 同时, 漏洞威胁映射关系是 CTI 知识图谱本体架构中的一个关键关系, 然而这一关系在诸如 NVD 等漏洞库中并未充分涵盖, 因此为了实现漏洞与威胁的关联, 本文提出了一种基于安全领域知识增强语言表示模型 SecERNIE, 并将其应用于漏洞威胁映射关系挖掘任务中.

2.1.3.1 基于安全领域知识增强语言表示模型

信息表示模型应该能够捕获文本数据中隐藏的语言规则, 如词汇含义、语法结构、语义角色等. 目前常用的信息表示模型是以 BERT 为代表的神经网络表示模型, 这类模型可以从纯文本中很好地捕获丰富的语句信息, 并

且可以通过微调以持续提高 NLP 任务的性能. 但是, 神经网络表示模型只能获取语句是否通顺, 不能很好地进行知识的表示.

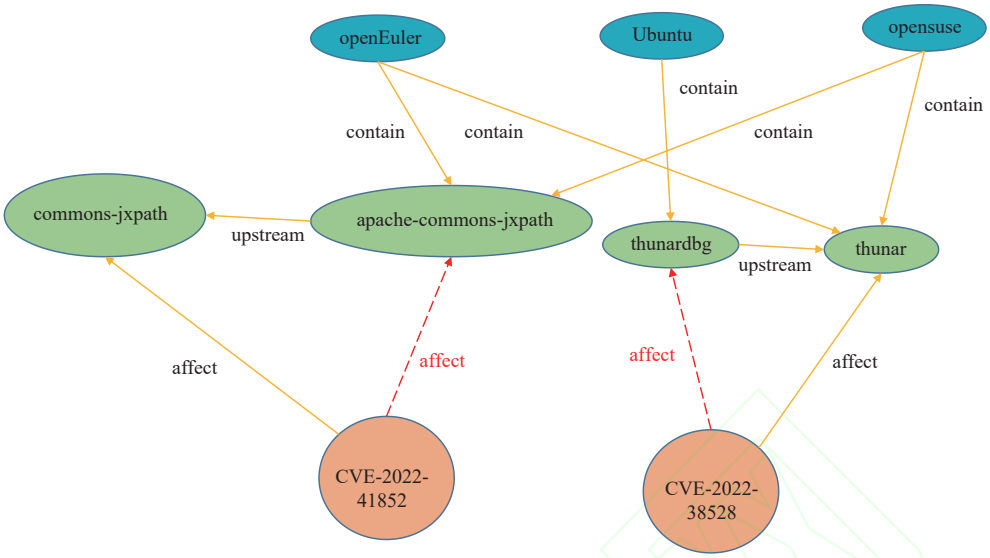

图 4　包命名矩阵匹配示意图

针对 BERT 表示模型缺少知识表示的问题, 本文考虑到 BERT 模型中单词分布从通用领域语料库向安全领域语料库的转移, 往往会产生结果偏差. 因此, 本文在 ERNIE 模型的基础上, 通过大规模安全领域语料库进行预训练, 构建了安全垂直领域知识增强语言表示模型 SecERNIE, 将安全领域知识信息与表示模型融合, 进一步提升模型在安全领域的语义表示能力.

本节介绍 SecERNIE 及其详细实现. SecERNIE 本质上是基于 ERNIE 模型架构, 通过构建安全领域数据集进行模型预训练获得的安全垂直领域语义表示模型, 模型架构如图 5 所示, 由文本嵌入模块和 Transformer 编码器两个模块组成. 其中, 文本嵌入模块将输入序列中的每个单词转换为其对应的词向量表示. 这包括将单词转换为词嵌入 (word embeddings) 和添加位置嵌入 (position embeddings), 以便模型能够捕捉单词的语义和上下文信息. Transformer 编码器是模型的主体, 每个 Transformer 编码器层包括多头自注意力机制 (multi-head self-attention) 和前馈神经网络 (feed forward neural network) 两部分. 通过这些层使得模型在双向上下文中对输入序列进行编码和表示学习. ERNIE 模型在 BERT 模型基础上, 将模型掩码方式改为多层次掩码, 即单词层、短语层和实体层, 通过此方式实现模型语义加强. SecERNIE 沿用该方式将安全领域知识融合到模型表述中, 其与 BERT 的比较如图 6 所示.

在模型构建完成后, 预训练的具体步骤包括: 构建安全领域词汇表、构建预训练样本和预训练语言模型. 其中, 安全领域词汇表包含了预训练模型需要理解和处理的所有可能的词汇和子词单元, 不仅可以解决安全领域与通用领域术语差异问题, 提高模型的语言表达能力, 还能提高模型的效率, 将模型的词汇表限制在领域相关的术语上, 减少无关词汇计算资源与时间. 本文安全领域词汇表构建步骤如下.

(1) 收集安全领域文本数据: 获取已发行的成熟数据集、大型安全类结构化数据库、安全新闻文本等非结构化文本以及开源软件生态数据源这 4 类安全领域数据.

(2) 数据预处理: 针对获取到的安全领域文本数据去除特殊符号、标点符号、数字等信息, 再将文本转化为小写后去除停用词, 减少文本噪音与冗余.

(3) 分词与词频统计: 将预处理后的安全领域文本数据通过 Wordpiece 算法将文本分割为子词单元, 每个子词单元都有一个对应的词频, 本文选择最高的子词单元作为确认词汇.

(4) 分配唯一标识符: 对于每个确认词汇, 使用整数数字作为唯一标识符, 词频越高的词汇数字越小, 将其排序后通过文本文件的形式存储形成安全领域词汇表.

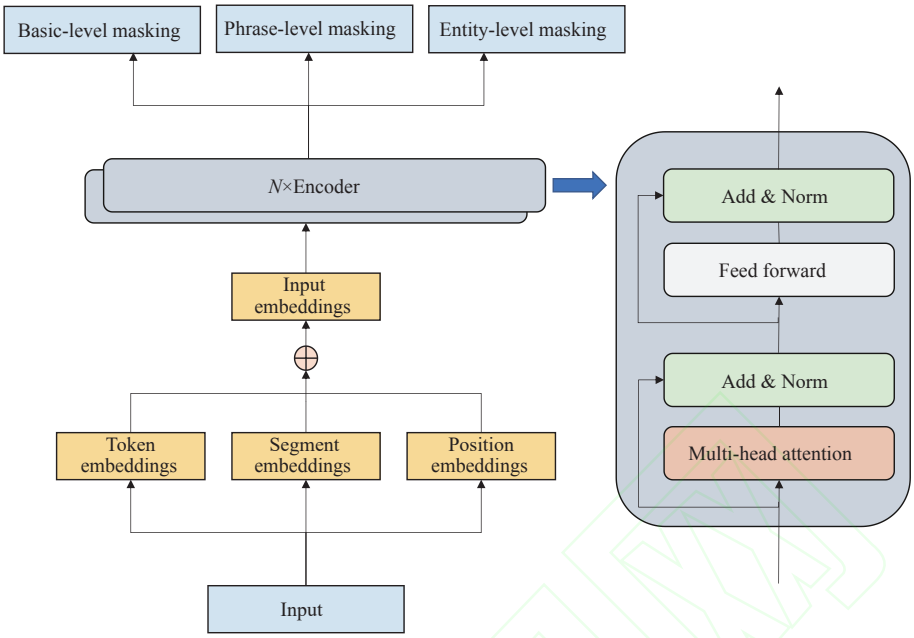

图 5 SecERNIE 模型架构

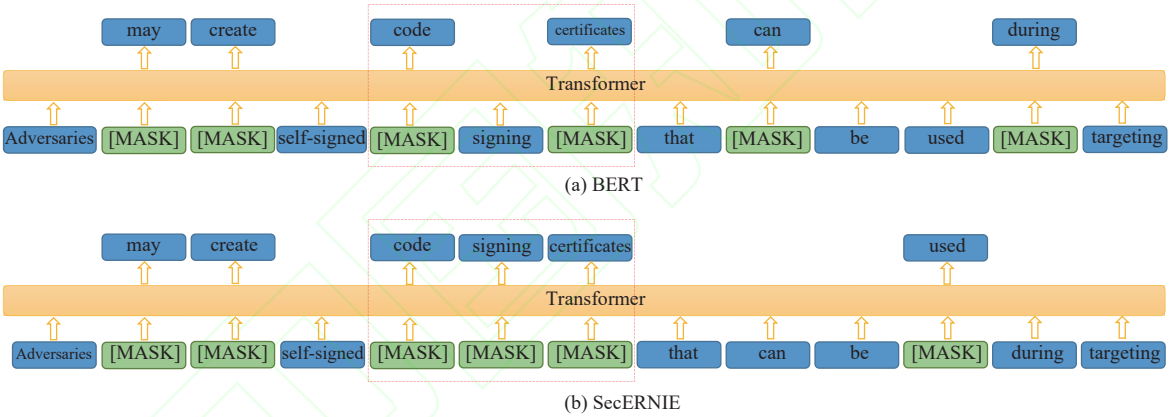

图 6 SecERNIE 与 BERT 掩码策略差异示意图

预训练样本是模型预训练的输入数据, 样本数据来源于构建安全领域词汇表所收集到的大规模文本语料. 本文预训练样本的构建过程如下.

(1) 文本清洗及预处理: 为了使原始文本数据更有效地用于模型训练, 本文将文本数据的字符编码标准化为统一的 UTF-8 编码, 以确保文本的一致性; 将文本中 HTML、特殊字符、停用词等不包含文本信息的标签移除, 提高模型处理效率; 最后将全部文本转化为小写并进行词干提取, 减少词汇多样性.

(2) 创建掩码: 将清洗后样本词汇映射到对应的词汇表并等分为 3 组子样本, 按照不同层级的掩码策略创建掩码. 其中, 单词层子样本随机选择 15% 的领域词汇, 将其替换为特殊的掩码标记 [MASK_Word]; 短语层子样本使用 spaCy 依存分析器获取短语边界信息, 随机选择 15% 的短语将其替换为特殊的掩码标记 [MASK_Phrase]; 实体层子样本通过 Stanford NER 工具进行实体抽取, 随机选择 15% 样本中出现在领域词汇表中的实体 (人员, 位置, 组

织, 产品等), 将其替换为特殊的掩码标记 [MASK_Entity].

模型预训练采用自监督学习方法, 将构建好的训练样本输入到模型中, 通过前向传播生成文本表示、计算损失函数, 将结果用于衡量模型的性能, 反向传播和权重更新用于调整模型参数, 以便使其学习通用的语言表示, 通过 MLM (masked language model) 任务迫使模型在不同层次上学习语言知识, 包括词汇、语法、语义等多个层面. 例如, 对于输入文本序列为“SeaLotus is a suspected [MASK_Entity] based threat group”, 模型的一次训练过程如下: 首先预测“[MASK_Entity]”对应的词汇, 通过交叉熵损失函数度量模型的预测和实际词汇之间的差异, 使用链式法则计算损失函数对模型参数的梯度, 最后用 Adam 算法进行参数优化. 为了校正 ERNIE 模型嵌入分布, 本文在模型接入了一个 flow 变换的模型, 在模型预训练结束后会学习出 flow 变换, 然后可以用 flow 变换将 ERNIE 句子嵌入转换到高斯空间, 从而解决模型嵌入分布不均匀的问题.

SecERNIE 的编码器架构与 transformer 相同, 模型可以根据结构表示为 4 个部分:

$$S = \{PE, MA, AN, FF\} \tag{1}$$

其中, *PE* 为 position embedding 表示位置编码; *MA* 为 multi-head attention 表示多头注意力机制; *AN* 为 add norm 表示残差连接及归一化; *FF* 为 feed forward 表示激活函数.

PE 为输入文本在句子中的位置进行标记, 公式如下所示:

$$PE(pos, 2i) = \sin\left(pos/10000^{2i/d_{\text{mod}}}\right) \tag{2}$$

其中, *pos* 表示字在句子中的位置, *i* 表示词向量的维度. 经过 *PE* 编码, 可以得到和输入维度完全一致的编码数组 X_{POS}, 并将其叠加到原文本词表示中得到新的词嵌入.

MA 可以计算相关性, 其中引入了查询 (query)、键值 (value) 和键 (key) 的概念, 通过查询结果与键进行比较可以得到一个相关性分数, 再与键值相乘得到最终 *MA* 的结果. 公式如下所示:

$$MA(Q, K, V) = Softmax\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V \tag{3}$$

其中, $\sqrt{d_k}$ 是为了将注意力矩阵变成标准正态分布, *Softmax* 进行归一化, 使每个字与其他字的注意力权重之和为 1.

AN 有两个操作, 残差连接和归一化 (layer normalization), 残差连接将上一层输入 *X* 与上一层输出相加; 归一化的作用是把神经网络中隐藏层归一为标准正态分布, 加速模型的收敛.

FF 是前馈网络, 包括两层线性映射及激活函数, 如公式 (4), 其中残差操作与归一化的步骤与 *AN* 相同; *表示矩阵乘法.

$$X_{\text{hidden}} = ReLU(X_{\text{hidden}} * W_1 * W_2) \tag{4}$$

SecERNIE 使用 12 个编码器层, 768 个隐藏单元和 12 个注意力 Header, 在 Transformer 的基础上对掩码策略进行修改, 通过单词、短语、实体层 3 个层次将安全领域知识编码到向量表示中, 在对话文本数据方面采用 ERNIE 自带的对话语料, 没有额外增加新的安全领域对话训练数据.

2.1.3.2 实体间映射关系挖掘

ATT&CK 是一种高级攻击信息模型, 将网络攻击活动的不同战术与特定的技术和程序 (TTP) 相关联. 因此, 自动化地将 CVE 标识的安全漏洞与 ATT&CK 框架 TTP 进行映射, 使安全团队可以更准确地确定哪些漏洞在特定威胁情境下具有更高的紧迫性. 这有助于更有效地分配资源, 集中精力修复最具威胁性的漏洞. 除此以外, 通过理解漏洞可能导致的威胁行为, 组织可以采取措施来降低相关攻击的风险, 提高漏洞威胁的防范能力. 但是, 由 CTI 知识图谱本体架构可知, 其数据结构共分为漏洞风险和威胁风险两个子图, 且两个子图之间存在的关联较少, 缺少安全漏洞与攻击技术、缓解措施之的关联关系. 针对上述问题, Hemberg 等人[53]提出通过已知数据实现 CVE 漏洞、CWE、CAPEC 与攻击技术之间的关联并构建关系图——BRON, 在关系图中可以查询漏洞与攻击技术的间接关联, 然而实际上 CVE 漏洞、CWE、CAPEC 等实体之间都存在一对多的关系, 如图 7 所示为 CVE-2016-1658 漏洞在 BRON 中漏洞到攻击技术之间的关联关系, 最终获取漏洞相关攻击技术 26 个. 经统计, 在 BRON 上有 21 097 个漏洞映射超过 5 个以上的攻击技术, 因此通过 BRON 不能直接获取最可能被攻击的技术, 不利于安全人员进行及时防护.

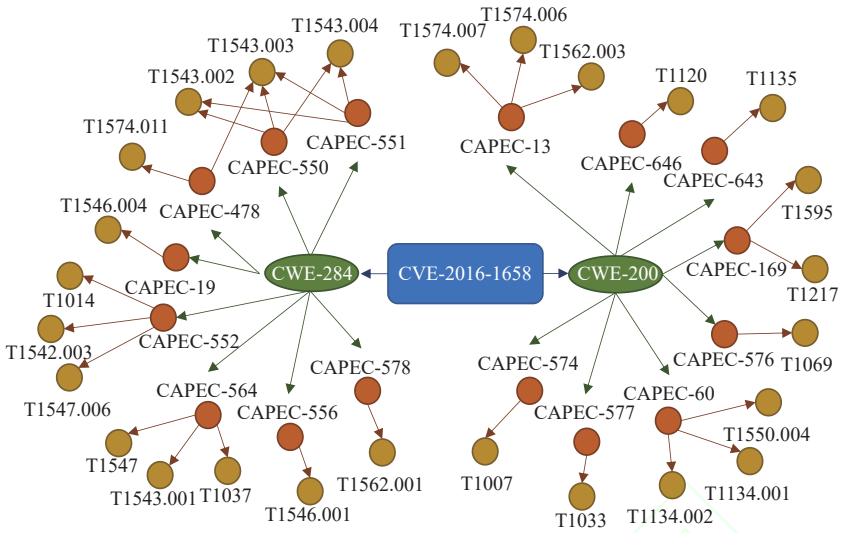

图 7　CVE-2016-1658 在 BRON 中的关联关系

针对上述问题, 为了获取安全漏洞与攻击技术之间映射关系, 本文通过将 SecERNIE 文本表示模型引入到开源软件供应链风险映射挖掘中, 获取最相关的关联映射, 并将该映射关系加入知识图谱中, 实现安全漏洞与攻击技术之间的关联关系挖掘, 图 8 为关联关系挖掘算法流程图.

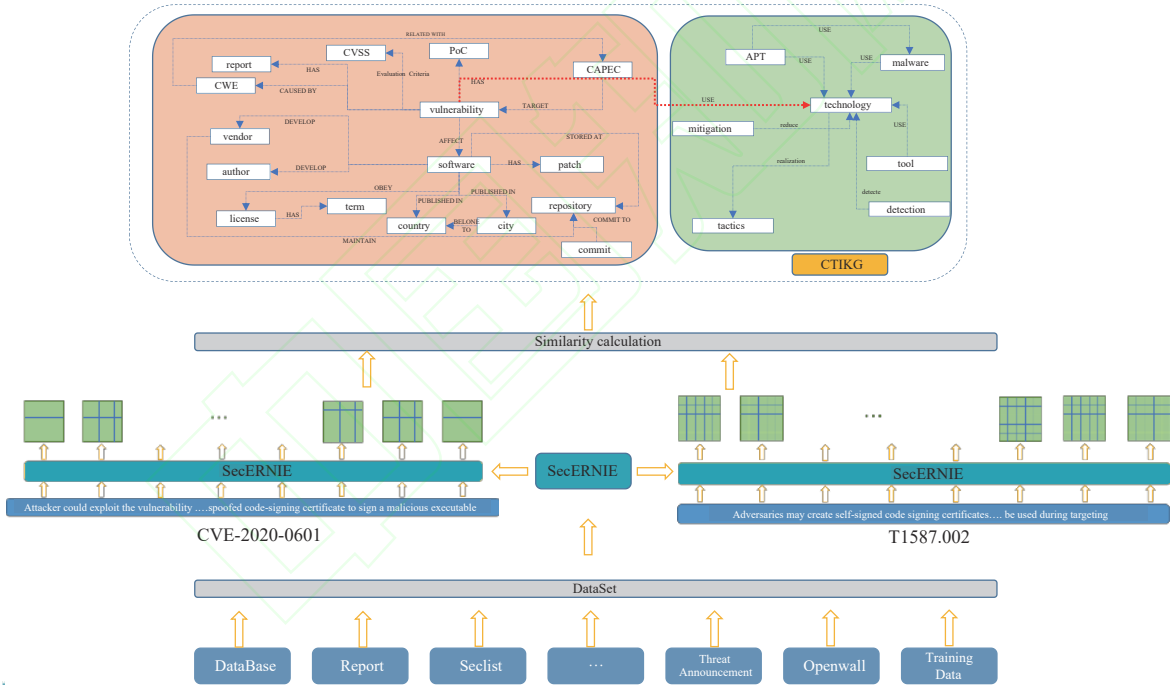

图 8　安全漏洞与攻击映射关系挖掘流程图

由于本文在 ERNIE 模型的基础上进行调整, 而该模型的词向量在空间中的分布不够均匀, 高频词分布更为紧凑, 这种锥形分布导致低频词语义不完整. 因此本文参考 BERT-flow[54]方法, 通过无监督的方式将 SecERNIE 得到的语义空间转换为一个平滑、各向同性的高斯分布.

为了计算 ATT&CK 技术候选向量和 CVE 描述表示向量的相似性/不相似性, 需要选择正确的距离函数. 我们手动标记随机抽样的 30 个 CVE, 使用其相应的 ATT&CK 技术在威胁报告中找到, 提取上下文短语, 并如上所述创建候选向量. 为了选择 ATT&CK 技术与威胁报告中 CVE 描述之间的相似性度量, 我们测量 L2 距离、余弦距离、最大平均差异 (MMD) 和费舍尔距离. 选择性能最佳的距离测量方法来标记 pipline. 进行比较是为了确定每个距离函数对基础数据分布的影响. 例如, 具有相同均值和方差的样本可能具有较小的 L2 距离. 最终本文通过标记 pipline 选择余弦距离, 因为它具有适合此问题的最高相关值.

本文根据上述结论, 按照图 7 进行安全漏洞与攻击映射的挖掘. 以安全漏洞 CVE-2020-0601 为例, 首先提取该漏洞以及所有攻击的描述文本, 将所有描述文本通过预先训练好的 SecERNIE 文本表示模型映射到相同的语义空间. 为了使得漏洞描述和攻击描述高频词分布更加紧凑, 通过流生成学习成表示空间和数据之间的互信息没有变化的可逆空间, 再通过余弦相似度计算与该漏洞最相似的攻击技术, 构建两者之间的关系并用“USE”表示关联映射.

2.2 漏洞风险信息推送

漏洞风险信息推送主要基于 CTI 知识图谱中的漏洞动态信息, 并将其应用于开源系统的漏洞数据过滤. 当安全漏洞发生新增或更新的变化时, 我们将对该漏洞关联的所有软件实体名称与开源系统中的软件名称进行匹配. 若匹配成功, 则判定该漏洞影响到开源系统中的软件. 然而, 在实际工程应用中, 由于数据来源不同, 匹配信息的置信度也不同, 这导致匹配结果存在误报的问题.

为了实现系统软件的可信追踪, 本文设计了一种基于软件包命名矩阵的软件威胁过滤规则, 通过不同的信息来源和包名匹配方式, 对过滤出的数据赋予 0–6 的置信度, 置信度越高表示软件中包含该漏洞的可能性越小, 从而筛选出 openEuler 系统软件包含的漏洞数据. 不同匹配方式对应置信度的方式如表 1 所示.

表 1　基于软件包命名矩阵的软件威胁过滤规则

置信度	匹配方式说明
0	待过滤软件包和版本号与NVD中cpe原始数据中的软件名称及版本信息进行完全匹配
1	待过滤软件名称通过软件包命名矩阵进行匹配, 版本号通过NVD中cpe原始数据版本信息匹配
2	待过滤软件包名和版本号与NVD漏洞描述中抽取出的软件名称和版本信息匹配
3	待过滤软件包名与NVD漏洞描述中抽取的软件名称在软件包命名矩阵中的别名进行匹配, 版本号与漏洞描述中的版本信息匹配
4	待过滤软件包名和版本号与通过安全公告抽取的软件名称和低于公告中修复版本的信息匹配
5	待过滤软件包名与通过安全公告抽取出的软件名称在软件包命名矩阵中的别名进行匹配, 版本号和低于公告中修复版本的信息匹配
6	待过滤软件版本未匹配成功, 包名通过上述任意方式匹配成功, 该等级置信度, 本文认为没有成功匹配, 不进行漏洞推送

本文按照设计好的本体结构, 将结构化数据按照知识模板构建初步的知识图谱. 将非结构化数据通过实体抽取、关系抽取等知识图谱技术进行清洗和知识补充, 构建形成一个大型 CTI 知识图谱, 并在此基础上实现全网数据的实时获取, 使得 CTI 知识图谱完成数据的自动更新, 实现开源软件的安全实时检测.

3 实验分析

本节将对实验数据和环境进行阐述, 首先将对实验准备工作进行说明, 包括实验环境、实验数据, 其次对本文进行实验结果阐述和实例分析.

3.1 实验准备

(1) 实验环境

本文中涉及的实验在配有 Tesla P100 PCIE 12 GB GPU, Intel Xeon Silver 4116 CPU@2.10 GHz CPU 和 64 GB 内存的设备上进行, 运行负载为每分钟处理 400 条情报数据.

(2) 评价指标

针对文本表示模型在完形填空任务中的对比实验评价, 本文使用平均正确率 (mean accuracy, *MA*) 和语义相关性 (semantic relevance, *SR*) 两个指标评估模型在语义方面的性能. 假设 N 为完型填空被替换单词总数, T 表示填空正确的单词数量, A_i 表示模型预测出的第 i 个单词向量, B_i 表示第 i 个目标单词向量. 则正确率 (*MA*)、语义相关性 (*SR*) 的计算方法如下:

$$MA = \frac{T}{N} \tag{5}$$

$$SR = \frac{\sum_{i=0}^{n} \cos(A_i, B_i)}{N} \tag{6}$$

从公式 (5) 可见, 平均正确率表示所有预测正确的单词与全部被替换单词总数的比例, 平均正确率越高表示模型性能越高. 从公式 (6) 可见, 语义相关性通过计算全部预测单词与目标单词的余弦距离求和后取平均值得到, 语义相关性数值越高表示模型预测出的词语与原文词语意思越相符, 模型语义表示能力越强.

针对文本表示模型在关联映射挖掘任务中的对比实验评价, 误报率 (false positive rate, *FPR*)、漏报率 (false negative rate, *FNR*)、查全率 (true positive rate, *TPR*)、查准率 (precision, *P*) 和 *F*1 分数 (*F*1 score, *F*1) 是 5 个常见的评价模型检测准确程度的指标. 假设 *FP* 为漏洞与攻击技术之间不存在关系但是被判定为存在关系的样本数量, *FN* 为漏洞与攻击技术之间存在关系但是被判定为不存在关系的样本数量, *TP* 为漏洞与攻击技术之间存在关系同时也被判定为存在关系的样本数量, *TN* 为漏洞与攻击技术之间不存在关系同时也被判定为不存在关系的样本数量. 则误报率 (*FPR*)、漏报率 (*FNR*)、查全率 (*TPR*)、查准率 (*P*) 和 *F*1 分数 (*F*1) 的计算方法如下:

$$FPR = \frac{FP}{FP + TN} \tag{7}$$

$$FNR = \frac{FN}{TP + FN} \tag{8}$$

$$TPR = \frac{TP}{TP + FN} \tag{9}$$

$$P = \frac{TP}{TP + FP} \tag{10}$$

$$F1 = \frac{2 \times P \times TPR}{P + TPR} \tag{11}$$

从公式 (7) 可见, 误报率表示所有漏洞与攻击技术之间不存在关系的样本中被错误判定为存在关系的样本的比例, 误报率越低则模型检测效果越理想. 从公式 (8) 可见, 漏报率表示所有漏洞与攻击技术之间存在关系的样本中被错误判定为不存在关系的样本的比例, 漏报率越低则模型检测效果越理想. 从公式 (9) 可见, 查全率表示所有漏洞与攻击及时之间存在关系的样本中被正确判定为存在关系的样本的比例, 查全率越高则模型检测效果越理想. 从公式 (10) 可见, 查准率表示所有漏洞与攻击技术之间被判定为存在关系的样本中确实存在关系的样本的比例. 从公式 (11) 可见, *F*1 分数表示查全率和查准率的调和平均值, 其综合考虑了查全率和查准率, *F*1 分数越高则模型检测效果越理想.

针对操作系统软件漏洞过滤过程中软件追踪效果评价, 将软件覆盖率 (software coverage, *SC*) 作为实验评估指标, 假设操作系统中全部软件数量为 *FSN* (full software number), 方法可追踪且感知到软件漏洞情况的数量为 *STN* (software track number), 则软件覆盖率 (*SC*) 的计算方法如下:

$$SC = \frac{STN}{FSN} \tag{12}$$

从公式 (12) 可见, 软件覆盖率表示本文模型中可追踪到软件安全情况的软件样本的比例, 软件覆盖率越高则软件漏洞过滤过程中软件追踪效果越理想.

针对本文方法在开源软件供应链的可用性实验评价, 将误推率 (false send rate, *FSR*)、漏推率 (omit send rate,

OSR) 作为实验评估指标, 假设已知官方平台软件全部漏洞数量为 *FQ* (full quantity), 官方平台接受用户推送漏洞的数量为 *AN* (accept number), 官方平台拒绝用户推送漏洞的数量 *RN* (reject number), 则误推率 (*FSR*)、漏推率 (*OSR*) 的计算方法如下:

$$FSR = \frac{RN}{AN + RN} \tag{13}$$

$$OSR = \frac{AN - RN}{AN} \tag{14}$$

从公式 (13) 可见, 误推率表示官方平台不存在的漏洞被错误的推送到平台的漏洞样本的比例, 误推率越低则方法的漏洞感知效果越理想. 从公式 (14) 可见, 漏推率表示官方平台存在的漏洞但是没有被感知到的漏洞样本比例, 漏推率越低则方法的漏洞感知效果越理想.

(3) 数据准备

本文在预训练 SecERNIE 表示模型以及构建 CTI 知识图谱 2 个模块中使用的数据集包含 4 部分: 已发行的成熟数据集、大型安全类结构化数据库、安全新闻文本等非结构化文本以及开源软件生态数据源. 其中, 已发行的成熟数据集如表 2 所示, 包括 MalwareTextDB 和 CASIE 工具使用的数据集. MalwareTextDB 是由 Lim 等人[55]在 2017 年提出的一个为带注释的恶意软件文本构建的数据库, 该数据库包含了 317 篇 APT 报告, 帮助网络安全人员进行分析工作. Satyapanich 等人[56]提出的用于网络攻击感知与信息提取工具 CASIE 使用的涉及数据破坏、网络钓鱼、勒索病毒、发现和补丁等类型相关的 1 000 篇网络安全事件的新闻文章.

表 2 已发行数据集列表

数据源	数据量	包含信息	数据用途
MalwareTextDB	317	恶意软件、APT事件	SecERNIE训练
CASIE数据集	1 000	攻击者、技术、工具、漏洞、事件	SecERNIE训练

安全类结构化数据库数据集是构建 CTI 知识图谱的核心数据集, 主要通过获取 NVD (美国国家漏洞数据库)、CNVD (国家信息安全漏洞共享平台)、CWE (漏洞弱点类型数据库) 以及 ATT&CK (攻击战术和技术知识库)、CAPEC (攻击模式数据库)、Exloit DB (漏洞利用数据库) 这 6 个安全领域数据库, 涵盖漏洞、攻击、战术、类型、利用、补丁等不同实体相关信息, 提供了全面、权威的数据支撑, 如表 3 所示为各个数据库中获取信息情况.

表 3 安全类数据库列表

数据源	数据量	包含信息	数据用途
CNVD	184 692	CNVD漏洞信息、安全公告、软件、补丁	CTI知识图谱构建
Exloit DB	47 714	漏洞利用、作者、平台类型	CTI知识图谱构建
CAPEC	518	攻击模式	CTI知识图谱构建、SecERNIE训练
ATT&CK	3 023	APT、攻击技术、恶意软件、战术、工具	CTI知识图谱构建、SecERNIE训练
NVD	211 820	CVE漏洞信息、软件、厂商、开发语言	CTI知识图谱构建、SecERNIE训练
CWE	909	漏洞类型、利用条件、威胁级别	CTI知识图谱构建、SecERNIE训练

针对安全非结构化文本的获取, 本文采集了 securelist、seclist、Openwall 等 12 个安全相关的网站, 获取了 73 439 条涉及攻击、修复、漏洞等不同类型的文本数据. 其中, securelist 包含了针对性攻击、移动威胁、垃圾邮件、钓鱼网络等多类型安全威胁或事件信息; seclist 和 Openwall 是安全邮件列表存档, 包含大量的安全漏洞披露信息、补丁提交等信息, 相对于安全网站, 其信息披露时间更早, 因此该数据源信息对于 CTI 知识图谱信息更新的时效性有重要意义, 如表 4 所示为重点非结构化信息数据列表.

本文通过获取大量开源软件信息实现开源软件供应链数据基础, 包括 GitHub、PYPI、NPM、RPM 等共 14 个开源软件生态平台及软件源、Ubuntu 等超过 50 类操作系统包含的软件包列表, 获取了仓库、软件、开发者、许可证、开发语言、国籍等大量开源软件供应链相关信息, 如表 5 所示为重点开源软件数据列表.

表 4　非结构化安全类文本重点数据列表

数据源	数据量	包含信息	数据用途
securelist	2429	漏洞、事件、威胁、APT、恶意软件	SecERNIE训练
seclist	26264	邮件、补丁、作者、国家、漏洞等	CTI知识图谱构建、SecERNIE训练
Openwall	28251	邮件、补丁、作者、国家、漏洞等	CTI知识图谱构建、SecERNIE训练
threatpost	11435	漏洞、恶意软件、作者、威胁情报	SecERNIE训练

表 5　开源软件数据列表

数据源	数据量	包含信息	数据用途
GitHub	456441	仓库、软件、开发者、国籍、许可证、开发语言	SecERNIE训练
PYPI	2549132	软件、许可证、开发者、开发语言	CTI知识图谱构建、SecERNIE训练
OpenHarmony	28251	仓库、软件、开发者、国籍、许可证	CTI知识图谱构建、SecERNIE训练
OS	103982	软件、开发者、厂商	SecERNIE训练

3.2　实验方法与结果分析

为了评估开源软件供应链漏洞威胁智能感知方法的效果, 本文首通过 CTI 知识图谱构建关键过程进行数据时效性、准确率、覆盖率等指标分别对关键步骤进行评估, 最后, 以 openEuler 中包含的开源软件为实验对象测试文本方法的可用性.

3.2.1　CTI 知识图谱构建模块效果评估

3.2.1.1　CTI 知识图谱数据及时效性评估

本文构建 CTI 知识图谱主要目的是实现开源软件供应链风险态势感知, 因此 CTI 知识图谱是实现风险挖掘的数据基础. 本文通过自顶向下的方式构建知识图谱, 通过参考 STIX 2.1、ATT&CK 等多种数据标准及数据结构, 设计并实现如图 4 所示本体架构, 其中包含 26 类实体及 41 类实体之间的关系.

通过部署爬虫系统, 实时获取并分析第 3.1 节中提及相关数据源, 根据本体结构进行数据融合, 实现 CTI 知识图谱的构建, 其中包含关联关系数量超过 2.8 亿、实体数量超过 2 千万, 重点覆盖了 NPM、PYPI 和 GitHub 这 3 类开源软件生态超过 200 万开源软件实体, 其数量分布如图 9 所示.

本文将涉及 Linux 操作系统的 51590 个安全漏洞进行测试. 首先分别提取 ArchLinux、Gentoo、Ubuntu、Redhat 这 4 个大型 Linux 操作系统厂商和 NVD、CNVD 两大漏洞数据库发布漏洞时间, 将其与 CTI 知识图谱漏洞感知时间进行对比统计, 统计结果如表 6 所示.

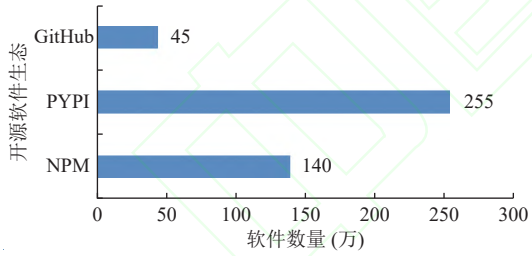

图 9　开源软件生态数量分布

表 6　漏洞风险感知时间对比统计

漏洞平台	平均提前时间 (天)	提前获取漏洞占比 (%)
NVD	90.03	94.21
CNVD	66.97	91.46
ArchLinux	35.23	63.5
Gentoo	67.10	95.33
Ubuntu	54.11	47.81
Redhat	1.81	6.27

通过表 6 中不同数据源的漏洞风险感知时间对比统计可知, Redhat 系统漏洞发布时间远远早于其他操作系统厂商以及 NVD 等大型漏洞库漏洞发布时间, 而 CTI 知识图谱相较于 Redhat 系统漏洞库平均早 1.8 天, 占总漏洞数量的 6.27%, 由此证明 CTI 知识图谱可以实现更早的安全漏洞风险感知.

3.2.1.2　信息抽取模型效果评估

本文方法针对信息缺失问题, 采用 BERT+LSTM+CRF 模型作为文本抽取模型, 为了验证该模型效果, 本文选取 CVErizer 与 NER-ES 作为实体、关系抽取任务的基线进行对比实验. CVErizer 与 NER-ES 两个模型都可以针对安全

漏洞描述进行软件名称及版本的抽取, 其中 NER-ES 通过 ES 专家规则还可以实现软件名称与版本之间关系的抽取.

为训练神经网络并评价抽取效果, 本文首先标注了 3 800 条数据集, 以 6:2:2 的比例将数据集划分为训练集、验证集和测试集, 使用训练集对模型进行训练. 使用测试集对训练完毕的模型的效果进行测试. 为了实现软件名称及版本的匹配, 本文提出一种基于规则的关系映射方法, 首先分析了 NVD 漏洞描述文本的结构特点, 然后根据这些结构特点设计了一种映射规则, 通过这种方法, 可以有效地从漏洞描述中提取出软件名称、版本之间的关系, 并生成结构化的数据. 如表 7 所示为本方法与不同模型在软件版本、软件名称及其之间关联关系抽取任务中的效果对比. 从实验结果中可见, 本方法在实体及关系抽取任务上均优于 CVErizer, 虽然软件版本抽取准确率稍低于 NER-ES, 但是关系抽取的效果显著优于该模型. 将本方法应用于 NVD 漏洞库的全部漏洞描述, 并将抽取结果与 NVD 发布的 CPE 对比, 发现 NVD 数据库中存在 18 396 条漏洞数据存在信息缺失. 本文随机抽取 800 条输出结果进行验证, 有 762 条漏洞数据存在影响范围信息缺失问题, 因此该模型识别 NVD 信息缺失问题的准确率达到 95.25%.

表 7 BERT+LSTM+CRF 模型抽取查准率 (*P*) 统计 (%)

工具或方法	软件版本	软件名称	名称版本关系
CVErizer	72.3	79.6	—
NER-ES	97.1	94.5	52.9
本方法	96.6	95.7	84.7

3.2.1.3 实体间映射关系挖掘效果评估

为了验证 SecERNIE 模型在安全领域方面的语义提升效果, 本文通过构建安全领域完型填空测试, 将威胁情报、漏洞描述、攻击描述等文本, 共构建测试样本 277 条. 对于测试样本, 本文将其中一个或多个出现在安全领域词汇表中的单词替换为掩码标记, 共替换安全领域词汇 314 个. 将测试样本分别输入到 SecERNIE 与 ERNIE 两个模型中. 对于每个模型使用平均正确率及填充内容与上下文的语义相关性得分 2 个指标进行评估, 实验结果如表 8 所示, 加粗数值为最佳结果.

表 8 模型语义表示能力对比统计

表示模型	平均正确率*MA* (%)	语义相关性 *SR*
ERNIE	60.3	0.59
SecERNIE	**79.7**	**0.87**

通过完形填空测试可知, SecERNIE 表现出更高的平均正确率, 这表明其在安全领悟任务中的语义理解能力明显优于 ERNIE. 此外, SecERNIE 语义相关性得分也明显高于 ERNIE.

表 9 呈现了两个模型在完型填空任务中的 3 个案例. 在案例 1 和案例 2 中, ERNIE 模型虽可以进行实体类型推理, 但由于缺乏领域相关知识, 未能准确填充实体名称. SecERNIE 模型则可以根据上下文正确填写实体. 在案例 3 中, ERNIE 模型未能生成实体名称, 而 SecERNIE 模型虽未完全匹配实体名称, 但能够正确地预测出语义信息.

表 9 完型填空测试案例

No.	安全领域文本	SecERNIE预测结果	ERNIE预测结果	目标答案
1	____ is a suspected Vietnam-based threat group that has been active since at least 2014.The group has targeted multiple private sector industries as well as foreign governments, dissidents, and journalists with a strong focus on Southeast Asian countries like Vietnam, the Philippines, Laos, and Cambodia.They have extensively used strategic web compromises to compromise victims.	sealotus	organizations of Vietnam	sealotus
2	The BluStar component in Mitel InAttend before 2.5 SP3 and CMG before 8.4 SP3 Suite Servers has a default password, which could allow remote attackers to gain unauthorized access and execute arbitrary scripts with potential impacts to the ____, integrity and ____ of the system.	confidentiality、availability	stability、security	confidentiality、availability
3	Adversaries may establish persistence and/or ____ by executing malicious content triggered by AppInit DLLs loaded into processes.	escalate privileges	_	elevate privileges

针对知识图谱中安全漏洞与攻击技术之间的映射关系挖掘任务, 本文选取 BERT 和 ERNIE 作为语言表示模型基线, 与 SecERNIE 模型进行对比实验. 本文标注了 4 972 条漏洞及对应的攻击技术数据对, 并将其作为基线任务的测试集, 如表 10 所示为 SecERNIE 与 BERT、ERNIE 在数据集上的对比实验结果. 从对比数据中可见, SecERNIE 在数据集上的表现明显优于其他 2 个方法, 其在数据集上的 *F*1 值达到了 78.43%.

表 10　SecERNIE 与 BERT 和 ERNIE 进行对比实验的结果 (%)

表示模型	误报率*FPR*	漏报率*FNR*	查全率*TPR*	查准率*P*	*F*1分数
BERT	28.15	34.23	63.2	66.1	64.62
ERNIE	35.7	61.42	54.7	62.0	58.12
SecERNIE	19.25	19.3	81.23	75.81	**78.43**

通过本文方法, 共挖掘出安全漏洞与攻击之间的关联关系 63 492 条、安全漏洞与缓解措施之间的间接关联关系 190 526 条、覆盖开源软件 933 432 个, 并在此基础上实现信息自动检测与更新.

3.2.2　漏洞风险信息推送模块效果评估

针对开源软件生态中操作系统子领域软件包别名问题, 本文将 NVD 中所有安全漏洞涉及的 753 809 个软件作为软件匹配基础数据集, 并将 CTI 知识图谱中包含的 103 982 个 Linux 系统软件使用软件名称及版本在基础数据集中查询, 严格匹配到的系统软件数量为 12 478 个, 占比全部系统软件的 12%. 为了提升操作系统软件覆盖率, 本文获取了 8 个不同系统官方代码托管平台软件包文件, 提取了软件包中包含上游源信息的 spec 等文件, 构建了软件包命名矩阵, 涉及操作系统相关软件超过 4.7 万个, 补充了大量的系统软件别名信息. 将上述匹配失败的 91 504 个操作系统软件包通过软件包命名矩阵查询别名后进行上述方法的二次匹配, 成功匹配到 77 331 个软件包, 操作系统的软件覆盖率从 12% 提升到 86.37%.

3.2.3　实验方法可用性评估

为了评估本方法在开源软件供应链的实际运用效果 (可用性), 本文自 2022 年 1 月开始, 将本方法应用于 openEuler 系统的 Gitee 官方开源托管平台, 对其系统发行版包含的 6 726 个开源软件进行漏洞威胁感知, 并将感知结果以 ISSUE 的形式推送给系统官方, 最后通过分析系统官方对 ISSUE 的处理情况来进行可用性评估.

本文通过第 2.2 节制定的过滤规则对 openEuler 系统软件的覆盖率进行分析, 针对 openEuler 中包含的软件进行风险挖掘的结果如图 10 所示, 本实验对 openEuler 发行版操作系统累计追踪 6 239 个软件包的安全情况, 占 openEuler 软件总数的 92.76%, 其中 2 282 个软件包存在漏洞, 占比 36.58%, 3 957 个软件包没有漏洞, 占比 63.42%.

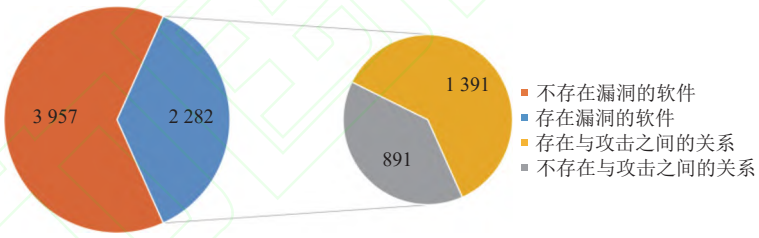

图 10　openEuler 开源软件漏洞威胁关系挖掘结果

为了评估本方法推送漏洞的精确度, 本文将 openEuler 系统的 ISSUE 作为数据集, 进行实验结果统计和分析. 本文将 ISSUE 提交者分为本方法以和其他所有用户, 分别统计两类提交者在官方平台的提交情况, 并进行对比. 截至 2023 年 5 月, opneEuler 系统的 Gitee 官方开源托管平台共发布 8 488 个漏洞 ISSUE, 具体提交情况如表 11 所示. 从表 11 中可以看出, 本方法在漏洞威胁感知任务的误推率和漏推率均低于其他用户. 此外, 本文还统计了漏洞 ISSUE 标签状态为 FIXED 的数量, 发现平台对漏洞的整体修复占比为 66.85%.

本文利用构建的 CTI 知识图谱, 获取了 openEuler 系统软件所涉及的安全漏洞以及漏洞对应的攻击技术, 并将攻击技术关联的缓解措施补充到推送的漏洞信息中. 经统计, 共有 891 个漏洞与攻击技术相关联, 并给出相应的缓解措施, 其中有 784 个漏洞被官方平台修复, 占比为 87.99%, 高于系统平均修复比例. 由此可见, 本文方法可以有效提升平台对漏洞的修复效率.

表 11 openEuler 官方平台 ISSUE 提交对比统计

ISSUE提交者	接受数量 (个)	拒绝数量 (个)	被修复漏洞数量 (个)	误推率 *FSR* (%)	漏推率 *OSR* (%)
本方法	6212	525	3828	7.79	26.81
其他用户	2276	237	1846	9.43	73.19

3.3 实例分析

通过对 openEuler 操作系统中包含的软件进行实验, 在其包含的安全漏洞中挖掘出 891 个安全漏洞与攻击技术的关联关系, 进而获取安全漏洞与缓解措施的间接关联. 为验证其推理得到的漏洞威胁及缓解方法的有效性, 本节从上述漏洞中随机选取两个安全漏洞进行防御前后的可达性实验.

3.3.1 CVE-2022-27774 攻防示例

漏洞 CVE-2022-27774 是 curl 软件的一个“传递型”漏洞, 影响了 181 个版本. 该漏洞源于 libcurl 组件, 自 4.9.0 版本引入后一直存在. 正常情况下, curl 可向同一主机名间的重定向 URL 传递凭据, 不会将凭据传递给其他主机. 如图 11(a) 所示, 当 curl 使用-L 参数递归访问重定向后的 URL 时, 该漏洞导致凭据泄露到其他主机, 如图 11(b) 所示, 用户请求 server1 的凭据会被发送到 server2, 造成凭据泄露.

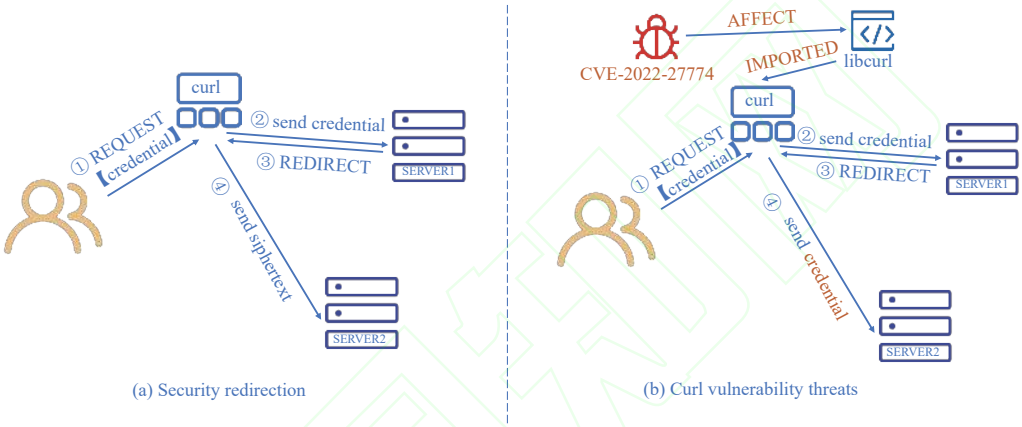

图 11 CVE-2022-27774 攻击示意图

为验证漏洞攻击威胁感知的可达性, 本文将上述漏洞描述与 CTI 图谱中所有“攻击”类型节点描述通过 SecERNIE 映射到同一个向量空间, 并通过余弦距离其相似度, 匹配到最相似的攻击类型为 T1040 (网络嗅探). 针对上述漏洞, 网络嗅探攻击捕获的数据应包括用户凭据. 为验证推理得到的攻击类型是否可达, 本文首先搭建了漏洞攻击实验平台, 即在 server1 上搭建 http 服务器, server2 上搭建 ftp 服务器, 其中 server1 的作用是将请求重定向到 server2, 并在 server2 上开启网络嗅探. 随后在用户终端上使用 curl 命令携带用户凭据访问 server1, 发现 server2 捕获了用户凭据, 如图 12 所示, 表明漏洞攻击可达性验证成功.

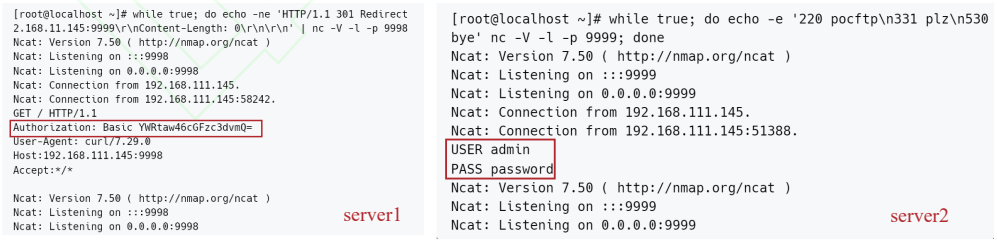

图 12 CVE-2022-27774 攻击结果

为验证漏洞缓解措施的有效性, 如图 13 所示为组件 curl 在知识图谱中的风险映射关系, 通过该关系, 本文可以进一步获取与其相关的 T1040 攻击类型节点对应的所有缓解措施 M1032 (multi-factor authentication)、M1018

(user account management) 和 M1041 (encrypt sensitive information), 进一步经实验人员分析, 选择 M1041 来缓解该漏洞, 即通过对流量进行监控, 确保流量中不存在相应的明文信息.

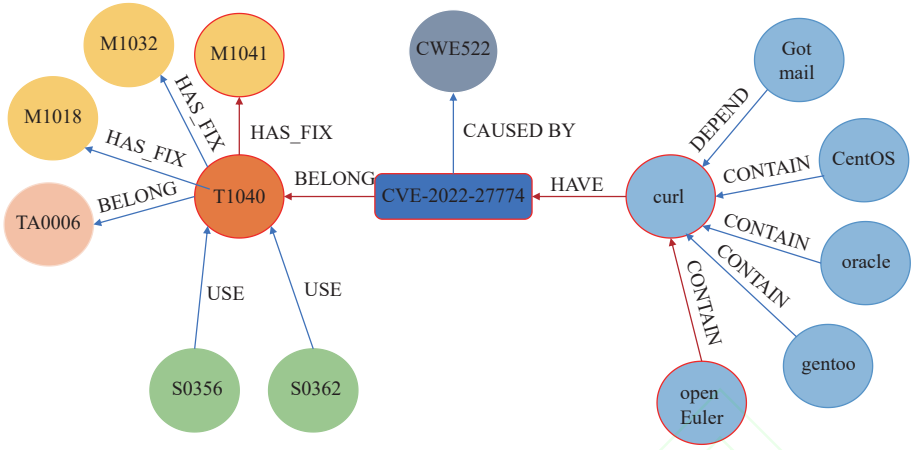

图 13 风险映射关系

以 M1041 节点相应的缓解措施为基础, 设计环境中具体的防御方案: 设置中间人代理按照指定规则监控拦截实时通信流量. 如果发送给 server1 的请求包含用户凭据、请求头中 curl 版本为受影响的版本, 且 server1 的响应为重定向到其他主机, 则拦截该响应. 通过图 14 可以看出, 未进行防御时, 重定向后的其他主机可以获取到用户凭据. 当使用方案后, 重定向后的其他主机没有获得用户凭据, 防御成功.

图 14 CVE-2022-27774 防御结果

3.3.2 CVE-2021-22555 攻防示例

为进一步验证针对内核级漏洞攻击和防御的效果, 本文选择漏洞 CVE-2021-22555 做进一步实验. 该漏洞是 Linux kernel 中包含的一个提权漏洞, 隐藏在内核模块 Netfilter 中长达 15 年之久. 当内核编译选项 CONFIG_USER_NS、CONFIG_NET_NS 开启的情况下, 由于 x_tables.c 中的 xt_compat_target_from_user 函数中由于 8 字节对齐操作产生的 4 字节溢出, 使得 setsockopt 函数处理 IPT_SO_SET_REPLACE 选项时位长转换错误, 引发了堆溢出漏洞, 攻击示意图如图 15 所示.

为验证漏洞攻击威胁的可达性, 将该漏洞描述信息与 CTI 图谱中所有“攻击”类型节点描述通过 SecERNIE 映射到同一个向量空间, 匹配到最可能的攻击类型为 T1067(权限提升), 攻击者可利用软件漏洞以实现权限提升. 为验证推理得到的攻击类型是否可达, 搭建受影响版本内核的攻击实验环境, 并运行公开的漏洞 PoC, 通过命名空间初始化、内存破坏、安全机制绕过、内核代码执行等步骤成功实现权限提升, 攻击结果如图 16 所示.

为验证漏洞缓解措施的有效性, 获取该节点对应的缓解措施 M1051 (Update Software), 即通过对系统采用补丁管理来定期更新软件. 针对漏洞产生的根本原因为 xt_compat_target_from_user 函数 8 字节对齐操作错误, 所以应用补丁将 8 字节对齐操作取消以达成漏洞修复. 由图 17 可以看出在执行对应的缓解措施后, 再次提权攻击时在安全绕过机制的步骤攻击失败, 表明该缓解措施有效.

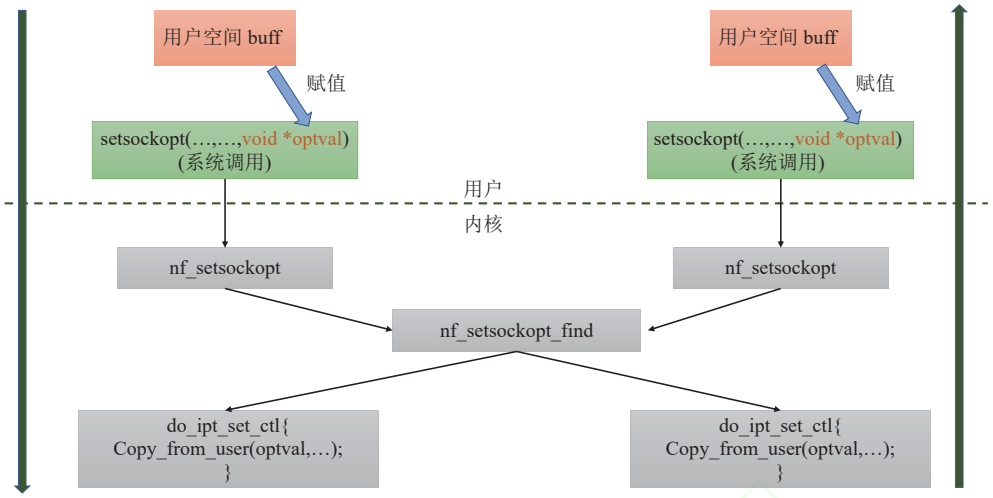

图 15 CVE-2021-22555 攻击示意图

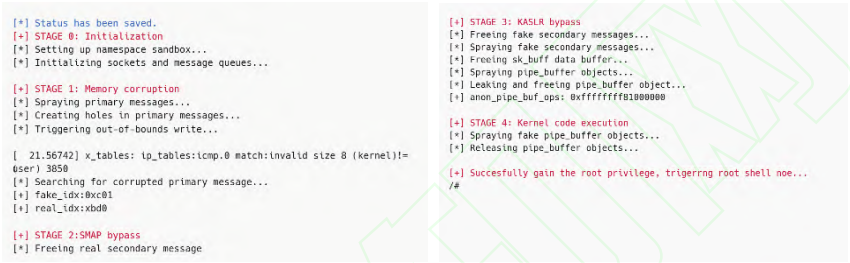

图 16 CVE-2021-22555 攻击结果

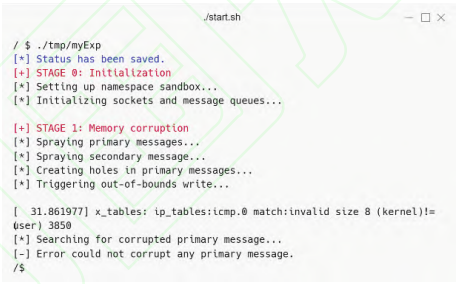

图 17 CVE-2021-22555 防御结果

因此, 通过安全漏洞与攻击技术、攻击技术与缓解措施之间的映射关系, 可对软件进行真实可用的攻击威胁感知和缓解措施推荐, 表明本方法的有效性.

4 总 结

本文针对开源软件供应链的安全问题, 提出了一个漏洞威胁智能感知的方法, 该方法分为构建 CTI 知识图谱与漏洞风险信息推送两部分. 其中构建 CTI 知识图谱是实现安全情报处理的数据基础, 参考公开漏洞库数据结构、开源软件生态平台、社区情报、ATT&CK 框架及 STIX2.1 标准架构设计 CTI 知识图谱本体结构. 之后, 通过将结构化数据按照本体结构进行数据填充, 并通过数据抽取技术将邮件列表、威胁情报和漏洞描述等文本知识整合到知识图谱中, 实现漏洞缺失数据补全并提高漏洞时效性. 针对开源软件别名问题, 该模块还通过分析开源软件内部

文件, 以获取有关开源系统中软件上游源信息, 从而提高对开源系统的软件覆盖率. 最后, 通过本文训练的安全领域语义表示模型 SecERNIE, 实现了 CTI 知识图谱子领域之间的实体映射挖掘. 漏洞风险信息推送模块基于动态更新的 CTI 知识图谱数据, 通过制定安全漏洞过滤规则, 实现对开源系统的漏洞风险实时感知.

本文方法相较于 NVD 等传统漏洞平台, 平均感知时间最高提前 90.03 天; 在操作系统软件覆盖率上提升了 74.37%, 并利用 SecERNIE 模型实现了 63 492 个 CVE 漏洞与攻击技术实体之间的关联关系映射. 特别地, 针对 openEuler 操作系统, 可追踪的系统软件覆盖率达到 92.76%, 并累计感知 6 239 个安全漏洞; 同时, 本文还发现了 openEuler 中 891 条漏洞与攻击的关联关系, 进而获取到相应的解决方案, 为漏洞处理提供了参考依据. 本文在真实攻击环境验证了 2 种典型的攻击场景, 证明本文方法在漏洞威胁感知方面的良好的效果, 为开源软件供应链安全提供了坚实的基础.

本文提出的一种漏洞威胁映射关系挖掘方法, 仍有改进空间. 一是本文通过余弦相似度来计算两个实体之间的相关性, 这种方法较为简单, 只考虑了方向的差异, 忽略了向量的大小和环境的影响. 二是本文无法准确地推荐最合适的防御措施, 因为一个攻击类型可能对应多个缓解措施, 而本文只能根据 CTI 知识图谱给出所有可能的措施, 但是没有办法判断哪个环节措施对于目前漏洞环境是最有效的, 因此如何提高映射关系挖掘的精度和防御措施推荐的有效性将是未来研究的重点.

致谢 本文得到了源图重大基础设施的支持.

References:

[1] Liang GY, Wu YJ, Wu JZ, Zhao C. Open source software supply chain for reliability assurance of operating systems. Ruan Jian Xue Bao/Journal of Software, 2020, 31(10): 3056–3073 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6070.htm [doi: 10.13328/j.cnki.jos.006070]

[2] Sonatype. 9th annual state of the software supply chain. 2021. https://www.sonatype.com/resources/state-of-the-software-supply-chain-2021

[3] Dell. Dell data security survey finds that a lack of security knowledge limits business initiatives. 2016. https://www.businesswire.com/news/home/20160308005494/en/Dell-Data-Security-Survey-Finds-that-a-Lack-of-Security-Knowledge-Limits-Business-Initiatives

[4] ACSM. The numbers behind a cyber pandemic. 2021. https://australiancybersecuritymagazine.com.au/the-numbers-behind-a-cyber-pandemic/

[5] Synopsys. Open source security and risk analysis report. 2021. http://www.synopsys.com/software-integrity/resources/analyst-reports/opensource-security-risk-analysis.html

[6] Forain I, De Oliveira Albuquerque R, De Sousa Júnior RT. Towards system security: What a comparison of national vulnerability databases reveals. In: Proc. of the 17th Iberian Conf. on Information Systems and Technologies. Madrid: IEEE, 2022. 1–6. [doi: 10.23919/CISTI54924.2022.9820232]

[7] Ji SL, Wang QY, Chen AY, Zhao BB, Ye T, Zhang XH, Wu JZ, Li J, Yin JW, Wu YJ. Survey on open-source software supply chain security. Ruan Jian Xue Bao/Journal of Software, 2023, 34(3): 1330–1364 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6717.htm [doi: 10.13328/j.cnki.jos.006717]

[8] CISA. Advanced persistent threat compromise of government agencies, critical infrastructure, and private sector organizations. 2021. https://www.cisa.gov/news-events/cybersecurity-advisories/aa20-352a

[9] Extracting a 19 year old code execution from WinRAR. 2019. https://research.checkpoint.com/2019/extracting-code-execution-from-winrar/

[10] Cyberattaque Chsf: Communique De Presse. 2022. https://www.ght-idfsud.fr/portail/actualites/contenus-44-59.html

[11] Endsley MR. Toward a theory of situation awareness in dynamic systems. Human Factors, 1995, 37(1), 32–64. [doi: 10.1518/001872095779049543]

[12] Bass T. Multisensor data fusion for next generation distributed intrusion detection systems. In: Proc. of the 1999 RIS National Symp. on Sensor and Data Fusion. The Johns Hopkins University Applied Physics Laboratory, 1999. 24–27. [doi: 10.13140/RG.2.2.20357.96482/1]

[13] Jajodia S, Noel S, O'Berry B. Topological analysis of network attack vulnerability. In: Kumar V, Srivastava J, Lazarevic A, eds. Managing Cyber Threats: Issues, Approaches, and Challenges. New York: Springer, 2005. 247–266. [doi: 10.1007/0-387-24230-9_9]

[14] Liu Q, Li Y, Duan H, Liu Y, Qin ZG. Knowledge graph construction techniques. Journal of Computer Research and Development, 2016,

53(3): 582–600 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2016.20148228]

[15] Zheng ZQ, Liu YG, Zhang Y, Wen CB. TCMKG: A deep learning based traditional Chinese medicine knowledge graph platform. In: Proc. of the 2020 IEEE Int'l Conf. on Knowledge Graph (ICKG). Nanjing: IEEE, 2020. 560–564. [doi: 10.1109/ICBK50248.2020.00084]

[16] Zhao QS, Hu WT, Ding HH. Study on military equipment knowledge construction based on knowledge graph. In: Proc. of the 8th Int'l Conf. on Big Data and Information Analytics (BigDIA). Guiyang: IEEE, 2022. 336–341. [doi: 10.1109/BigDIA56350.2022.9874117]

[17] Zhang F, Wu JZ, Nie YL, Jiang LH, Zhou AL, Xie NF. Research of knowledge graph technology and its applications in agricultural information consultation field. In: Proc. of the 39th IEEE Int'l Performance Computing and Communications Conf. (IPCCC). Austin: IEEE, 2020. 1–4. [doi: 10.1109/IPCCC50635.2020.9391515]

[18] Fu LJ, Bai Y, Zhong ZY. Constructing a vertical knowledge graph for non-traditional machining industry. In: Proc. of the 15th IEEE Int'l Conf. on Networking, Sensing and Control (ICNSC). Zhuhai: IEEE, 2018. 1–5. [doi: 10.1109/ICNSC.2018.8361341]

[19] Zhong XF, Zhang Y, Liu JJ, Yang GZ, Zhou SC, Wang P. Research on automated cyber asset scanning tools based on cybersecurity knowledge graph. In: Proc. of the 7th Int'l Conf. on Computer and Communications (ICCC). Chengdu: IEEE, 2021. 2046–2049. [doi: 10.1109/ICCC54389.2021.9674377]

[20] Wang YX. Research and implementation of NSSA technology based on knowledge graph [MS. Thesis]. Chengdu: University of Electronic Science and Technology of China, 2020 (in Chinese with English abstract). [doi: 10.27005/d.cnki.gdzku.2020.003048]

[21] Wang LM. Research on construction of vulnerability knowledge graph and vulnerability situation awareness [MS. Thesis]. Beijing: University of Chinese Academy of Sciences (School of Artificial Intelligence), 2020 (in Chinese with English abstract). [doi: 10.27824/d.cnki.gzkdx.2020.000080]

[22] Firth JR. Applications of general linguistics. Trans. of the Philological Society, 1957, 56(1): 1–14. [doi: 10.1111/j.1467-968X.1957.tb00568.x]

[23] Deerwester SC, Dumais ST, Landauer TK, Harshman R. Indexing by latent semantic analysis. Journal of the American Society for Information Science, 1990, 41(6): 391–407. [doi: 10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9]

[24] Lund K, Burgess C. Producing high-dimensional semantic spaces from lexical co-occurrence. Behavior Research Methods, Instruments, & Computers, 1996, 28(2): 203–208. [doi: 10.3758/BF03204766]

[25] Rohde DLT, Gonnerman LM, Plaut DC. An improved model of semantic similarity based on lexical co-occurrence. In: Modeling Word Meaning Using Lexical Co-occurrence. Rohde, 2005.

[26] Raff E, Zak R, Cox R, Sylvester J, Yacci P, Ward R, Tracy A, McLean M, Nicholas C. An investigation of byte *n*-gram features for malware classification. Journal of Computer Virology and Hacking Techniques, 2018, 14(1): 1–20. [doi: 10.1007/s11416-016-0283-1]

[27] Hancock JT, Khoshgoftaar TM. Survey on categorical data for neural networks. Journal of Big Data, 2020, 7(1): 28. [doi: 10.1186/s40537-020-00305-w]

[28] Hinton GE. Learning distributed representations of concepts. In: Morris RGM, ed. Parallel Distributed Processing: Implications for Psychology and Neurobiology. Oxford: Clarendon Press/Oxford University Press, 1989.

[29] Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. In: Proc. of the 26th Int'l Conf. on Neural Information Processing Systems. Lake Tahoe: Curran Associates Inc., 2013. 3111–3119.

[30] Mikolov T, Chen K, Corrado GS, Dean J. Efficient estimation of word representations in vector space. In: Proc. of the 2013 Int'l Conf. on Learning Representations. Scottsdale: ICLR, 2013. 1–12.

[31] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional Transformers for language understanding. In: Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1 (Long and Short Papers). Minneapolis: Association for Computational Linguistics, 2018. 4171–4186. [doi: 10.18653/v1/N19-1423]

[32] Zhang ZY, Han X, Liu ZY, Jiang X, Sun MS, Liu Q. ERNIE: Enhanced language representation with informative entities. In: Proc. of the 57th Annual Meeting of the Association for Computational Linguistics. Florence: Association for Computational Linguistics, 2019. 1441–1451. [doi: 10.18653/v1/P19-1139]

[33] Sun Y, Wang SH, Li YK, Feng SK, Chen XY, Zhang H, Tian X, Zhu DX, Tian H, Wu H. ERNIE: Enhanced representation through knowledge integration. arXiv:1904.09223, 2019

[34] Chen DG. Research on entity relationship extraction algorithm based on deep learning [MS. Thesis]. Chengdu: University of Electronic Science and Technology of China, 2021 (in Chinese with English abstract).

[35] Chinchor N, Marsh E. Appendix D: MUC-7 information extraction task definition (version 5.1). 1998. https://www-nlpir.nist.gov/related_projects/muc/proceedings/ie_task.html

[36] Appelt DE, Hobbs JR, Bear J, Israel D, Kameyama M, Martin D, Myers K, Tyson M. SRI Int'l FASTUS system: MUC-6 test results and analysis. In: Proc. of the 6th Message Understanding Conf. (MUC-6). Columbia, 1995. 237–248.

[37] Nahm UY, Mooney RJ. Using soft-matching mined rules to improve information extraction. In: Proc. of the 2004 AAAI Workshop on Adaptive Text Extraction and Mining (ATEM 2004). San Jose, 2004. 1–6.

[38] Brin S. Extracting patterns and relations from the World Wide Web. In: Proc. of the 1999 World Wide Web and Databases. Valencia: Springer, 1999. 172–183. [doi: 10.1007/10704656_11]

[39] Tong HQ, Lu XC. Consumption psychoanalysis and customer relationship management based on association rules mining. In: Proc. of the 2009 WRI World Congress on Computer Science and Information Engineering. Los Angeles: IEEE, 2009. 384–388. [doi: 10.1109/CSIE.2009.897]

[40] Mallik S, Mukhopadhyay A, Maulik U, Bandyopadhyay S. Integrated analysis of gene expression and genome-wide DNA methylation for tumor prediction: An association rule mining-based approach. In: Proc. of the 2013 IEEE Symp. on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB). Singapore: IEEE, 2013. 120–127. [doi: 10.1109/CIBCB.2013.6595397]

[41] Srisawat A. An application of association rule mining based on stock market. In: Proc. of the the 3rd Int'l Conf. on Data Mining and Intelligent Information Technology Applications. Macao: IEEE, 2011. 259–262.

[42] Zhang Y. Research on the distance supervise relation extraction based on deep learning [MS. Thesis]. Beijing: Beijing Jiaotong University, 2021 (in Chinese with English abstract). [doi: 10.26944/d.cnki.gbfju.2021.002618]

[43] Collins M, Duffy NP. Convolution kernels for natural language. In: Proc. of the 14th Int'l Conf. on Neural Information Processing Systems: Natural and Synthetic. Vancouver: MIT Press, 2001. 625–632.

[44] Liu CY, Sun WB, Chao WH, Che WX. Convolution neural network for relation extraction. In: Proc. of the 9th Int'l Conf. on Advanced Data Mining and Applications. Hangzhou: Springer, 2013. 231–242. [doi: 10.1007/978-3-642-53917-6_21]

[45] Xu K, Feng YS, Huang SF, Zhao DY. Semantic relation classification via convolutional neural networks with simple negative sampling. In: Proc. of the 2015 Conf. on Empirical Methods in Natural Language Processing. Lisbon: Association for Computational Linguistics, 2015. 536–540. [doi: 10.18653/v1/D15-1062]

[46] Guo ZJ, Zhang Y, Lu W. Attention guided graph convolutional networks for relation extraction. In: Proc. of the 57th Annual Meeting of the Association for Computational Linguistics. Florence: Association for Computational Linguistics, 2019. 241–251. [doi: 10.18653/v1/P19-1024]

[47] Chen JX, Ji DH, Tan CL, Niu ZY. Relation extraction using label propagation based semi-supervised learning. In: Proc. of the 21st Int'l Conf. on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics. Sydney: Association for Computational Linguistics, 2006. 129–136. [doi: 10.3115/1220175.1220192]

[48] Peng JJ, Zhu LJ, Wang YD, Chen J. Mining relationships among multiple entities in biological networks. IEEE/ACM Trans. on Computational Biology and Bioinformatics, 2020, 17(3): 769–776. [doi: 10.1109/TCBB.2019.2904965]

[49] Zhou ZY, Zhang HY. Research on entity relationship extraction in financial and economic field based on deep learning. In: Proc. of the 4th IEEE Int'l Conf. on Computer and Communications (ICCC). Chengdu: IEEE, 2018. 2430–2435. [doi: 10.1109/CompComm.2018.8780966]

[50] Qi Y, Liu JF, Li N. The analysis of security risk in open source software supply chain. Journal of Information Security Research, 2021, 7(9): 790–794 (in Chinese with English abstract). [doi: 10.3969/j.issn.2096-1057.2021.09.001]

[51] Dong Y, Guo WB, Chen YQ, Xing XY, Zhang YQ, Wang G. Towards the detection of inconsistencies in public security vulnerability reports. In: Proc. of the 28th USENIX Conf. on Security Symp. Santa Clara: USENIX Association, 2019. 869–885.

[52] Shao YJ, Wu YJ, Yang MT, Luo TY, Wu JZ. A large-scale study on vulnerabilities in Linux using vtopia. In: Proc. of the 21st IEEE Int'l Conf. on Software Quality, Reliability and Security Companion (QRS-C). Hainan: IEEE, 2021. 1–10. [doi: 10.1109/QRS-C55045.2021.00157]

[53] Hemberg E, Kelly J, Shlapentokh-Rothman M, Reinstadler B, Xu K, Rutar N, O'Reilly UM. Linking threat tactics, techniques, and patterns with defensive weaknesses, vulnerabilities and affected platform configurations for cyber hunting. arXiv:2010.00533, 2020.

[54] Li BH, Zhou H, He JX, Wang MX, Yang YM, Li L. On the sentence embeddings from pre-trained language models. In: Proc. of the 2020 Conf. on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2020. 9119–9130. [doi: 10.18653/v1/2020.emnlp-main.733]

[55] Lim SK, Muis AO, Lu W, Ong CH. MalwareTextDB: A database for annotated malware articles. In: Proc. of the 55th Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers). Vancouver: Association for Computational Linguistics, 2017. 1557–1567. [doi: 10.18653/v1/P17-1143]

[56] Satyapanich T, Ferraro F, Finin T. CASIE: Extracting cybersecurity event information from text. In: Proc. of the 34th AAAI Conf. on Artificial Intelligence. New York: AAAI Press, 2020. 8749–8757. [DOI: 10.1609/aaai.v34i05.6401]

附中文参考文献:

[1] 梁冠宇, 武延军, 吴敬征, 赵琛. 面向操作系统可靠性保障的开源软件供应链. 软件学报, 2020, 31(10): 3056–3073. http://www.jos.org.cn/1000-9825/6070.htm [doi: 10.13328/j.cnki.jos.006070]

[7] 纪守领, 王琴应, 陈安莹, 赵彬彬, 叶童, 张旭鸿, 吴敬征, 李昀, 尹建伟, 武延军. 开源软件供应链安全研究综述. 软件学报, 2023, 34(3): 1330–1364. http://www.jos.org.cn/1000-9825/6717.htm [doi: 10.13328/j.cnki.jos.006717]

[14] 刘峤, 李杨, 段宏, 刘瑶, 秦志光. 知识图谱构建技术综述. 计算机研究与发展, 2016, 53(3): 582–600. [doi: 10.7544/issn1000-1239.2016.20148228]

[20] 王一琁. 基于知识图谱的网络安全态势感知技术研究与实现 [硕士学位论文]. 成都: 电子科技大学, 2020. [doi: 10.27005/d.cnki.gdzku.2020.003048]

[21] 王丽敏. 漏洞知识图谱的构建及漏洞态势感知技术研究 [硕士学位论文]. 北京: 中国科学院大学 (中国科学院大学人工智能学院), 2020. [doi: 10.27824/d.cnki.gzkdx.2020.000080]

[34] 陈德岗. 基于深度学习的实体关系抽取算法研究 [硕士学位论文]. 成都: 电子科技大学, 2021.

[42] 张娅. 基于深度学习的远程监督关系抽取算法研究 [硕士学位论文]. 北京: 北京交通大学, 2021. [doi: 10.26944/d.cnki.gbfju.2021.002618]

[50] 齐越, 刘金芳, 李宁. 开源软件供应链安全风险分析. 信息安全研究, 2021, 7(9): 790–794. [doi: 10.3969/j.issn.2096-1057.2021.09.001]

王丽敏(1994－), 女, 工程师, 主要研究领域为系统安全, 知识图谱, 软件供应链安全.

罗天悦(1990－), 男, 工程师, CCF 专业会员, 主要研究领域为操作系统安全分析, 代码漏洞挖掘, 人工智能安全.

吴敬征(1983－), 男, 博士, 研究员, 博士生导师, CCF 高级会员, 主要研究领域为系统安全, 漏洞挖掘, 操作系统安全.

屈晟(1989－), 男, 工程师, CCF 专业会员, 主要研究领域为开源软件供应链, 开源操作系统.

武延军(1979－), 男, 博士, 研究员, 博士生导师, CCF 杰出会员, 主要研究领域为操作系统, 机器学习, 系统软件, 系统安全.

杨牧天(1990－) , 男, 工程师, CCF 专业会员, 主要研究领域为系统安全, 漏洞挖掘, 软件供应链安全.

芮志清(1997－), 男, 博士生, 主要研究领域为系统安全, 物联网安全, 软件供应链安全.