doi：10. 19665/j.issn.1001-2400.2XXX.0X.0

开源软件供应链安全综述——上下游组织视角

张玉清[1,2,3]，贾培养[1,2]，张晓琪[2,3]，王鹤[1,2]，伍高飞[1,2]

（1. 西安电子科技大学 网络与信息安全学院，陕西 西安 710071；
2．中国科学院大学 国家计算机网络入侵防范中心，北京 101408；
3．西安电子科技大学杭州研究院，杭州 311231；）

摘要：随着开源社区的不断壮大，越来越多的企业和组织开始将开源软件应用到其代码库中。开源软件因其低成本、高效率和可扩展等优势，成为越来越多企业和组织的首选。开源软件已经成为软件供应链的重要组成部分，但其安全问题却被普遍忽视。目前针对开源软件供应链攻击及防御策略的研究综述偏向于从整体视角去研究，忽略了基于典型开源开发环境去研究开源软件供应链攻击及防御。因此，本文结合当前开源软件供应链攻击和防御的研究现状，从构建系统、版本控制系统和分发平台等角度出发，分析了典型开源开发环境下，软件供应链攻击随时间的演变，并从开源软件供应链上下游不同组织的视角出发，给出了针对性的防御策略。最后，本文介绍了开源软件供应链攻击及防御未来的研究方向，为防御开源软件供应链攻击提供了新思路。

关键词：开源软件；供应链；构建系统；版本控制系统；分发平台；防御策略；
中图分类号：TP391 **文献标识码**：A **文章编号**：1001-2400(2XXX)0X-0-0

An Overview of Open Source Software Supply Chain Security - An Upstream and Downstream Organizational Perspective

Zhang Yuqing[1,2,3], *JIA Peiyang*[1,2], *Zhang Xiaoqi*[2,3], *Wang He*[1,2], Wu Gaofei[1,2]

(1. School of Cyber Engineering, Xidian University, Xi'an 710126, China;
2. National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, Beijing 101408, China;
3. Hangzhou Research Institute of Xidian University,Hangzhou 311231,China)

Abstract: As the open source community continues to grow, more and more enterprises and organizations are applying open source software to their code base. Open source software has become the preferred choice of more and more enterprises and organizations because of its low cost, high efficiency

收稿日期： **网络出版时间：**
基金项目：国家自然科学基金（U1836210）；海南省重点研发科技项目（ZDYF202012）
作者简介：张玉清（1966—），男，博士，教授，E-mail：zhangyq@nipc.org.cn
贾培养（1997—），男，西安电子科技大学硕士研究生，E-mail：jiapy@nipc.org.cn
张晓琪（1999—），女，西安电子科技大学杭州研究院硕士研究生，E-mail：zhangxq@nipc.org.cn
王鹤（1987—），女，博士，讲师，E-mail：hewang@xidian.edu.cn
伍高飞（1987—），男，博士，讲师，E-mail：wugf@nipc.org.cn
通信作者：王鹤（1987—），女，博士，讲师，E-mail：hewang@xidian.edu.cn
网络出版地址：

and scalability. Open source software has become an important part of the software supply chain, but its security issues are commonly ignored. The current research reviews on open source software supply chain attacks and defense strategies are biased to study from a holistic perspective, neglecting to study open source software supply chain attacks and defense based on typical open source development environments. Therefore, this paper analyzes the evolution of software supply chain attacks over time in a typical open source development environment from the perspectives of build systems, version control systems, and distribution platforms, and gives targeted defense strategies from the perspectives of different organizations upstream and downstream of the open source software supply chain, taking into account the current research status of open source software supply chain attacks and defense. Finally, this paper introduces future research directions for open source software supply chain attacks and defense, which provides new ideas for defending against open source software supply chain attacks.

Key Words: Open source software; Supply chains; build systems; version control systems; distribution platforms; defense strategies

开源软件已成为现代软件供应链的重要组成部分，汽车、医疗、电子产品和工业等传统行业也直接或间接地依赖着开源软件。开源软件允许开发者在全球范围内协作开发，站在巨人的肩膀上开发软件。在应用程序开发过程中所涉及到的库、框架和其他工具等所有元素都可以视为软件供应链的一部分。开源软件供应链模型如图 1 所示，其中开源软件供应链的上游主要包括开源软件、代码和框架等内容，中间环节则是企业或组织进行自动化地软件构建、测试和发布流程（DevOps），下游则是开源软件供应链不同行业的终端消费者。在当今追求高效快速开发迭代的环境下，很少有公司或组织从头开始编写代码以开发软件。事实上，大多数组织都倾向于通过利用开源软件快速构建其应用程序的基本架构，以最大限度地提高开发效率和缩短产品上市时间。Boehmke 等人[1]系统地阐述了开源生态系统已经成为了全球供应链信息共享的基础。Synopsys 网络安全研究中心在 2022 年发布的《开源安全和风险分析报告》中指出，85%的被审开源代码库中包含至少已经过时 4 年的开源代码[2]。

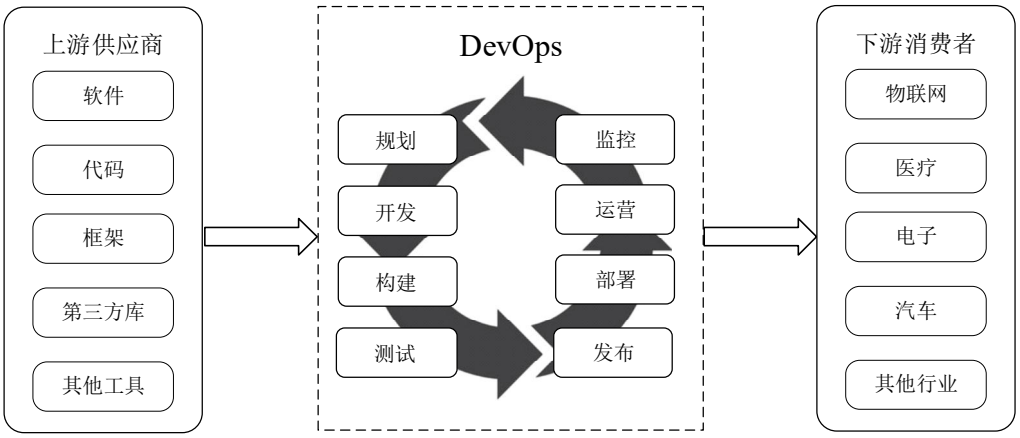

图 1 开源软件供应链模型

随着开源软件供应链规模的不断扩大，越来越多的攻击者利用已经被公开披露的漏洞间接攻击开源软件供应链上下游组织。一旦开源软件供应链上游的一个环节存在漏洞，其下游不计其数的应用程序都有可能受到该漏洞的影响。开源软件供应链易受攻击的原因除了影响面比较大之外，还和开源软件供应链上下游涉及到的众多生态系统的复杂性较高有关。开发者往往只关注自己开发代码的安全性，却很少关注其使用到的第三方开源软件的安全性，甚至很多开发者对其应用程序使用了哪些开源软件也是模糊的，这些都是开源软件供应链易受攻击的重要原因。

2021 年 7 月，被大量托管服务提供商和 IT 行业使用的网络监控和远程管理平台 Kaseya 宣称受到了软件供应链攻击，黑客通过 0day 漏洞绕过身份验证并运行任意命令入侵用于部署各种自动化

IT 任务和软件的 VSA 服务器，然后对多个托管服务提供商及其客户进行勒索软件攻击，这些托管服务提供商为数百甚至数千个下游业务客户提供 IT 外包服务，攻击者向 Kaseya 索要 5000 万美元赎金，约 1500 家企业受到此次攻击的影响。此次事件暴露出了企业对开源软件存在盲目信任的问题，导致大多数企业没有向审查私有软件一样审查开源软件。

防御开源软件供应链攻击已经成为软件供应链上下游组织的当务之急。但是，现有研究很少结合开源软件供应链上下游组织不同的技术特点给出针对性的防御策略。在开源软件安全、开源软件供应链攻击及防御等领域，一些研究人员进行了系统研究。2020 年，何等人[3]从软件供应链安全的定义及发展历程出发，着重研究了软件供应链安全技术和管理，但是对开源软件供应链攻击和防御的研究相对较少。2022 年，纪等人[4]总结了开源软件供应链的关键环节、威胁模型和安全趋势，并通过调研分析从风险识别和加固防御等方面总结了开源软件供应链的现状、挑战和未来的研究方向，但是缺乏对开源软件供应链上下游安全关联的分析。Ladisa 等人[5]在 S&P 安全会议上提出了针对开源软件供应链攻击的一般分类法，采用攻击树的形式，涵盖了 107 个特征向量，与 94 个真实事件相关联，并给出了 33 个缓解措施，其研究侧重于对开源软件供应链攻击进行分类，但缺乏标准化的安全实践。Ohm 等人[6]提供了一个包含恶意软件包的数据集，并分析了 174 个针对开源软件供应链进行真实攻击的恶意软件包，其研究侧重于理论分析，忽视了开源软件供应链上下游不同组织的实际状况。本文发现针对开源软件供应链攻击和防御的研究仍处于初步阶段，现有针对开源软件供应链攻击及防御的文献[3,4,6]大多从整体性的视角出发，对实际开源开发环境下的软件供应链攻击及防御研究较少，使得开源软件供应链上下游组织难以从自身视角出发针对性的防御开源软件供应链攻击。因此，有必要从实际开源开发环境的角度出发，对开源软件供应链攻击及防御的相关文献进行系统梳理和总结。这将为开源软件供应链的上下游组织以及安全研究人员提供宝贵的参考，以便更深入地了解开源软件供应链攻击技术的发展趋势，并制定有效的针对性防御策略。

本文利用“软件供应链安全研究”、“开源软件供应链攻击”、“软件供应链安全策略”等和软件供应链攻击及防御相关的关键词在 SCI、EI、IEEE、ACM 等数据库进行了数轮检索，并对相关文献进行了系统地收集和整理，本文共收集到了 80 篇相关文献，其中与开源软件供应链攻击直接相关的文献有 78 篇。本文对这些文献进行了年份统计，如图 2 所示，从图中可以看出针对软件供应链攻击的研究呈现逐年递增的趋势。本文发现现有软件供应链攻击主要集中在构建系统、版本控制系统和软件分发平台三个方面，因此本文把开源软件供应链攻击及防御的相关文献分为三个大的方向：

（1） 基于构建系统的开源软件供应链攻击

基于构建系统的开源软件供应链攻击主要发生在典型开源开发环境中的构建和部署环节。现代软件开发过程中大量采用构建系统将代码进行打包、测试和部署，构建系统已经成为了软件开发过程的中枢系统，例如被广泛使用的 CI/CD 系统。构建系统一旦被攻击将给软件供应链下游带来不可预知的安全隐患。

（2） 基于版本控制系统的开源软件供应链攻击

基于版本控制系统的开源软件供应链攻击多出现在远程代码托管平台（GitHub、GitLab 等）、本地版本控制系统等和版本控制相关的地方。此类攻击主要是通过分析远程代码托管平台中的源代码来获取密钥、凭证信息等敏感信息，进而借助这些数据对目标系统发起开源软件供应链攻击。

（3） 基于软件分发平台的开源软件供应链攻击

基于软件分发平台的开源软件供应链攻击指的是借助于软件包管理器生态系统或对应的软件注册表实施的开源软件供应链攻击。软件分发平台主要功能是帮助开发者存储和下载第三方开源软件包，攻击者往往通过依赖混淆攻击、恶意代码注入等方式发起开源软件供应链攻击。

本文的工作与主要贡献如下：

（1） 基于典型开源开发环境研究了开源软件供应链攻击及其防御。

本文从典型开源开发环境中涉及到的构建系统、版本控制系统和软件分发平台等开源软件供应链关键环节出发，系统归纳整理了相关文献。本文首先对目标环节中出现过的开源软件供应链攻击

事件的技术进展进行了研究，然后总结了该环节主要面临的开源软件供应链攻击类型，最后分析了不同文献中的防御策略。

（2） 基于开源软件供应链上下游不同组织的视角，给出了针对性的防御策略。

本文从开源软件供应链上下游中关键的开发者、构建系统、版本控制系统、软件分发平台以及软件供应链下游的终端用户等视角出发，将其自身技术特点和现有文献的防御策略进行结合，给出了被证实可行的防御策略。

（3） 基于开源软件供应链前沿安全研究，给出了未来可行的研究方向。

通过对开源软件供应链攻击及防御相关文献的研究，本文归纳了当前开源软件供应链迫切需要解决的问题，并结合当前研究热点和前沿方向，给出了未来可行的研究方向。本文认为解决开源软件供应链攻击首先需要列出详细完整的软件物料清单（SBOM），对软件开发生命周期的每一个环节可能出现的问题进行安全预警，同时监控项目中使用到的第三方开源依赖，结合现有主流防御技术为开源软件供应链构建多层防御体系。

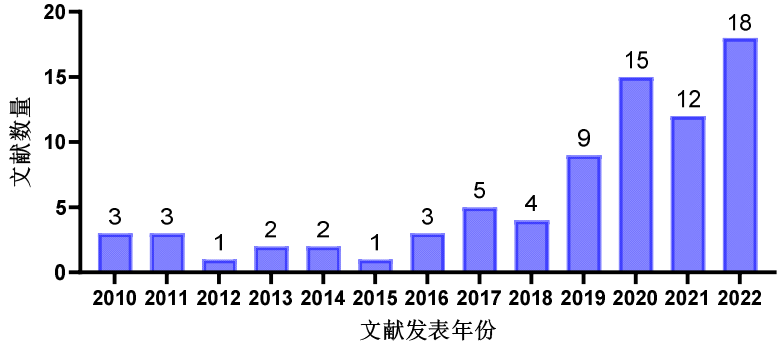

图 2 开源软件供应链攻击及防御的文献发表年份统计

1 开源软件供应链攻击概述

1.1 术语定义

开源软件供应链上下游组织中存在大量的专业术语，为了便于描述，表 1 总结了与开源软件供应链上下游相关的术语定义。

表 1 开源软件供应链上下游术语定义

术语	描述	示例
开源软件（OSS）	开源软件是根据许可发布的计算机软件，在许可授权下可以使用、研究、更改和分发软件及其源代码[7]。	Linux、Mozilla Firefox、Apache web server
开源许可证	开源许可证是一种允许计算机软件或其他产品在定义的条款下使用、修改或共享源代码的许可证[8]。	Apache License 2.0、MIT license
依赖项（也称为库或包）	当某个软件是其他软件使用时所需要的，也就是说其他软件依赖于这个软件，这个软件就成了依赖项。	rand(Rust)、request(Python)、math(JavaScript)
软件组成分析（SCA）	软件组成分析（SCA）能够自动化地识别代码库中的开源软件并评估其安全性、合规性和代码质量。	FlexNet、Debricked、WhiteSource、FOSSA
软件物料清单（SBOM）	软件物料清单（SBOM）是代码库中存在的所有开源和第三方依赖的列表。	SBOM 要包括数据字段、自动化和实践流程。
构建系统	构建系统可以让软件编译、打包和测试的过程自动化。	CI/CD、Maven
分发平台	分发平台主要用于开源软件包的注册和存储。	Npm、Packagist、PyPI
版本控制系统	版本控制系统主要用于跟踪和管理软件中的代码更改。	Git、SVN、CVS

1.2 典型开源开发环境

开源软件供应链的上游和下游是一个相对的概念。开源软件供应链的上游主要是生产类似原材料的软件或项目，其他开发者可以在这些项目中提交代码、测试、编写文档等。项目的维护者可以根据项目的需要选择是否合并其他开发者的代码到其项目代码库中，开源软件供应链上游的项目以这种方式不断地迭代更新，从而形成规模更大、更完善的开源软件。开源软件供应链的下游可以是某个组织为了实现某个功能，便在其项目中应用了上游的代码，形成了自己的应用程序。

开源软件供应链中代码的演变过程是极其复杂的，Kshetri 等人[9]认为对任何供应链的信任都是一个难以衡量的问题，因为软件供应链涉及大量的伙伴和产品。为了更好地理解开源软件供应链，可以从一个典型的代码开发者的视角出发，来探索在实际开发过程中开发者所依赖的开源开发环境和软件供应链之间的关系。典型的开源开发环境模型如图 3 所示。典型开源开发环境中的不同角色的主要任务如下所示：

（1） 开发者

开发者从零开始开发项目时，首先会通过代码编译器编写并测试代码，这个代码编译器可能是开源的，也可能是私有软件。当开发者开发完一个软件版本时，开发者往往会将代码提交到提供远程版本控制功能的代码托管平台。

（2） 构建系统

随着面向对象编程方法的普及，开发者往往是通过面向对象的编程方法进行编程，但是通过高级语言写成的源代码无法直接在计算机上运行，还需要通过构建系统进行构建才可以独立运行。构建系统已经成现代软件构建、自动化部署的中枢系统。

（3） 版本控制系统

版本控制系统既包含本地版本控制系统还包括远程代码托管平台。远程代码托管平台可以帮助开发者记录代码在整个软件开发生命周期内维护的记录，包括软件版本更改历史、作者身份以及其他细节。同时，全球参与相同开源项目的其他开发者也是通过远程代码托管平台进行协作开发。

（4） 软件分发平台

开发者在项目开发过程中如果需要用到第三方开源组件，还需要和类似于 npm 软件注册表等软件分发平台进行交互。软件分发平台通过一个中央代码存储库来进行第三方开源组件的保存和注册。开发者可以将自己开发的开源组件上传到软件分发平台，也可以从软件开发平台上下载其他开发者开发好的开源组件，这极大的缩短了软件开发周期，避免了软件开发中“重复造轮子”的现象。

（5） 终端用户

终端用户处于开源软件供应链的下游，软件供应链中任何上游组织出现的安全问题都可能波及到下游的应用程序。开源软件供应链中终端用户的定义是非常广泛的，普通应用程序的使用者、开源软件包的使用者等软件消费者都可以视为开源软件供应链的终端用户。

图 3 典型开源开发环境模型

1.3 开源软件供应链攻击发展历程

开源软件供应链攻击被大众熟知是因为近几年发生的软件供应链攻击事件，然而，软件供应链攻击并不是一个新的概念。自从软件供应链出现以来，针对软件供应链的攻击便从未停止过。攻击者往往通过寻找开源软件供应链中阻力最小的路径来发起软件供应链攻击。本文回顾了从 1982 年到 2022 年，近 40 年发生过的经典开源软件供应链攻击事件，以便更好地了解开源软件供应链攻击随时间的演变。

1982 年 6 月，俄罗斯西伯利亚管道发生严重爆炸，此次爆炸的强度是二战时期投向日本的核武器强度的 1/7，给俄罗斯的石油经济带来了严重的损失，此次事件后来证实是苏联在建设管道时使

用了美国情报部门提供的被提前预留后门的软件导致的[10]。2011 年，kernel.org 网站被不明身份的攻击者入侵，导致网站暂时下线，该攻击事件后来被证实是因为攻击者通过志愿者的凭证安装了木马并通过记录的密码发起软件供应链攻击[11]。

2013 年 6 月，攻击者利用韩国的文件共享和存储服务 SimDisk 存在漏洞的自动更新机制，攻击了韩国的政府和新闻网站[12]。同年，美国零售巨头 Target 的服务器被黑客入侵，黑客通过将恶意软件放在零售网络的 POS 机上，导致超过 4000 万张信用卡信息被泄露，最终 Target 向消费者赔偿了 1850 万美元[13]。

2016 年，知名的 BT 下载软件 Transmission 的客户端被黑客攻击，通过合法推进下载的客户端软件被恶意软件替换，导致 Transmission 为受感染的程序提供服务，导致其 Linux 和 Mac 版本的安装文件被安装上了木马[14]。2017 年，著名系统清理软件 CCleaner 被攻击者入侵，攻击者通过 TeamViewer 账户侵入了 Piriform 的网络系统，找到了 CCleaner 的分发服务器并发布了一个含有恶意代码的版本，从而发起了软件供应链攻击[15]。同年，乌克兰的一家金融软件制造商 ME Doc 的软件更新基础设施遭到网络攻击，受到该攻击波及的有 12000 多个系统和 64 个国家/地区，损失约 100 亿美元[16]。

2018 年，攻击者通过破坏 PDF 编辑器应用程序供应商和其软件供应商之间的共享设备，来发起多层的软件供应链攻击，使得 PDF 编辑器应用程序的合法版本成为恶意软件的载体，攻击者的目标是在受害者的机器上安装加密货币旷工软件[17]。同年，有攻击者提前将恶意的软件包发布到 Python 语言的包管理器软件注册表中，一旦有开发者使用 PyPI 下载并安装这个含有恶意代码的软件包，这个恶意的软件包会监控用户的支付行为，一旦用户通过加密货币的方式进行支付时，其支付信息将路由到攻击者指定的地址[18]。

2019 年，黑客使用自己的代码签名证书对华硕实时更新的工具进行了数字签名，并将其推送到华硕官方的下载服务器中，导致恶意更新被推送到默认安装该软件的华硕电脑中，影响多达 50 万台计算机[19]。同年，有攻击者企图借助 npm 包管理器将恶意软件包放入加密货币 Agama 的构建链中，以窃取目标应用程序中的钱包种子或登录密码[20]。

2020 年，该年发生的软件供应链攻击事件更是层出不穷。2020 年 3 月，一个名为 Octopus Scanner 的恶意软件攻击了托管在 GitHub 上的 26 个开源项目，攻击者将 GitHub 当做恶意软件交付中转站，将恶意后门代码安插到 NetBeans 项目中，以获取高度敏感信息的访问权限，例如生产环境、数据库密码和其他关键数据[21]。2020 年 11 月，攻击者为了传播其恶意软件，首先窃取了两家不同公司的数字证书，然后将真实的数字证书安装到恶意的浏览器插件中供用户下载，进而攻击韩国的政府和银行网站[22]。2020 年 12 月，系统管理工具开发商 SolarWinds 发生了严重的数据泄露事件。SolarWinds 是一家拥有超过 30 万名客户且服务于美国联邦政府的软件供应商。黑客通过木马化的 SolarWinds Orion 软件侵入了公司的网络，并对文件的源码进行了篡改，植入了后门代码，这直接导致了目标机构的网络设备账号、管理权限、数据库密码等高度敏感信息泄露，此次软件供应链攻击事件导致北美及欧洲近 200 家敏感机构受害[23]。

2021 年，一名研究人员将恶意软件上传到包括 PyPI、npm 和 RubyGems 在内的开源软件存储库中，然后这些恶意软件会自动分发到软件供应链下游公司的应用程序中。此次软件供应链攻击成功的攻击了包括微软、Apple、PayPal、Netflix、Tesla 等 35 家公司的内部系统[24]。2021 年 9 月，SushiSwap 的 MISO 加密货币平台受到开源软件供应链攻击，导致 300 万美元的损失。攻击者通过将恶意代码提交到 Sushi 的私人 GitHub 存储库“miso-studio”，攻击者通过更改公司拍卖网站的前端页面，将真实的钱包地址替换为攻击者指定的钱包地址，导致拍卖结束后价值约 300 万美元的以太坊代币转移到攻击者的钱包中[25]。

2022 年，一名开源开发者因不满公司未付费便使用其开源项目，便在其最新开源版本中加入了无限循环，导致使用到该开源组件的软件供应链下游应用程序出现一系列乱码。已知该开发者开发的开源项目 colors.js 在 npm 存储库中的下载量超过 33 亿次，并且拥有超过 19000 个项目依赖它。

此次攻击事件导致包括亚马逊、FaceBook 在内的多家科技公司受到影响[26]。

开源软件供应链攻击技术随时间的发展如图4所示，回顾开源软件供应链攻击技术的发展历程，可以明显看出开源软件供应链攻击随时间的演变主要经历了以下几个阶段：

（1） 早期阶段

早期的开源软件供应链攻击主要通过恶意软件感染软件包来实现。攻击者会将恶意代码注入到第三方开源软件包中，一旦该软件包被下载使用就会感染目标用户的主机。此类攻击容易被发现，影响面相对有限。

（2） 伪装和混淆阶段

这一阶段的开源软件供应链攻击通过伪装和混淆的技术手段，将恶意代码伪装成为正常代码并将其植入到第三方开源软件包中，从而发起供应链攻击，此类攻击很难被简单的代码审计或静态分析识别出来，因此相对早期阶段更加难以被发现。

（3） 后门和控制阶段

攻击者借助自己预留在开源软件中的软件后门和控制模块发起开源软件供应链攻击，一旦软件被使用，攻击者就可以直接控制用户的主机并窃取相关数据。由于后门和控制模块只在某些特殊的条件下才能被激活，一般的软件测试难以覆盖所有的情况，需要全面的审计才可能被发现，因此此类攻击发现难度较高。

（4） 加密货币挖矿攻击阶段

近几年，通过将开源软件植入到加密货币的挖矿模块从而实施软件供应链攻击以获取巨额利益的攻击事件越来越多，此类攻击隐蔽性较高，仅仅通过消耗系统资源进行挖矿，较少有其他恶意攻击行为，艰难被普通的检测机制识别，用户不易察觉，为攻击者赚取暴利提供了条件。

综上所述，开源软件供应链攻击的形式随时间不断地演变，攻击手法也越来越隐蔽，给开源软件供应链安全带来了很大挑战。虽然开源软件供应链攻击的形式各不相同，但所有开源软件供应链攻击都围绕着软件开发人员、构建系统、版本控制系统和软件分发平台中相对未设防的领域发起攻击。因此，本文将从典型开源开发环境出发对构建系统、版本控制系统和软件分发平台中的开源软件供应链攻击进行研究。

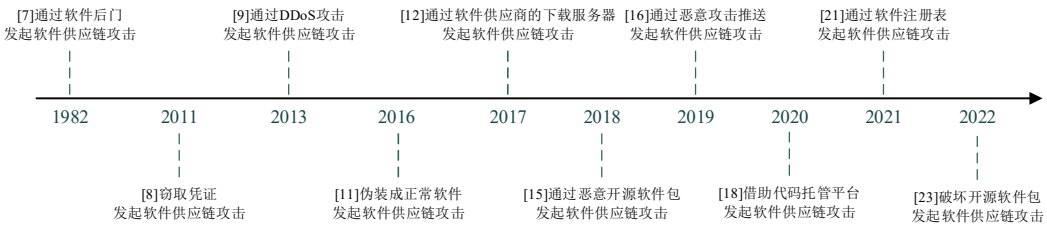

图 4　开源软件供应链攻击技术发展

2　基于构建系统的开源软件供应链攻击

软件构建系统在开源软件供应链中扮演着重要的角色，软件构建系统可以帮助开发者自动化地构建、测试并将代码部署到生产环境中。一旦软件构建系统受到攻击，所有使用该构建系统构建的应用程序将受到影响，保护软件构建系统并确保它们不会成为攻击者攻击的对象十分重要。CI/CD 系统是广泛使用的构建和部署自动化工具，也是软件开发过程中的中枢系统。CI/CD 系统在典型开源开发环境中的位置如图 5 所示。CI/CD 系统产生的更改会导致开源软件供应链下游的应用程序产生连锁反应，因此其也成为了攻击者的首选目标。

2017 年，Lipke 等人[27]通过将软件供应链和 Docker 结合，使用威胁建模过程来评估其安全性。其研究首先讨论了基于机密性、完整性和可用性的软件供应链安全策略。其研究认为构建服务器在处理敏感源代码时，应使用带防火墙的网络进行隔离并且将真实的用户数据与其他组件进行物理隔

离，因为这些源代码可能存在被攻击者攻击的风险，其成果给软件构建系统防御开源软件供应链攻击提供了思路。

2020 年，Ohm 等人[28]为了更好地检测恶意开源软件包，其研究首先对已经发生的开源软件供应链攻击进行了案例研究，然后基于发现提出了一个第三方依赖关系动态分析框架 Buildwatch。通过该框架对 CI/CD 管道安装的过程进行动态分析，其分析结果表明恶意开源软件包相比正常版本引入了大量的新工件。其研究可以更好地帮助开发者判断其依赖项更新是否存在恶意行为。随着万物互联技术的快速发展，容器技术的使用也在逐渐增加，容器技术可以改变应用程序的构建和部署方式。与普通的虚拟机相比，容器技术可以让测试、更新变得更加简洁，因此通过容器发起开源软件供应链攻击也变成了一种可能。Jiang 等人[29]从文件系统隔离、进程和通信隔离、设备管理和主机资源约束等方面对 Docker 容器的安全问题进行了研究，认为和传统虚拟机相比其安全性仍有待提高。Brady 等人[30]认为使用容器容易受到恶意软件的攻击并且缺乏有效量化的工具，现有的措施实施起来可能存在风险，通过病毒扫描和动态分析可以有效地检测 Docker 容器中的恶意漏洞。

2022 年，Benedetti 等人[31]对 Github Action 工作流进行了安全研究，以了解其对开源软件供应链产生的安全影响。其研究实现了一个工具 GHAST 并在 50 个开源项目中进行了实验。同时概述了 Github Action 工作流的安全状况，并修复了发现的安全漏洞。Bajpai 等人[32]认为在设计安全开发工作流和使用 CI/CD 管道时应使用包括静态应用程序安全测试（SAST）和动态应用程序安全测试（DAST）来代替手动安全审查，通过软件组成分析（SCA）来识别易受攻击的开源软件包，在引入开源软件包的时候执行严格的安全检查，在运营软件时，严格管控与 CI/CD 管道相关的访问控制权限。为了防止攻击者通过开发者凭据进行配置修改和注入恶意指令，应该使用人工监督批准的方式对配置文件和源代码访问控制权限进行限制。

随着 DevOps 技术的快速发展，现代软件开发过程中大量使用包括 CI/CD 系统在内的自动化构建系统，但是这些系统往往具有访问敏感代码权限并且还可以将软件部署到生产环境中，因此软件构建系统在开源软件供应链中的安全问题值得广泛关注。在基于构建系统的开源软件供应链攻击中，攻击者往往通过破坏 CI/CD 管道、对软件构建工具注入恶意代码、修改 CI/CD 配置文件等方式来发起开源软件供应链攻击。

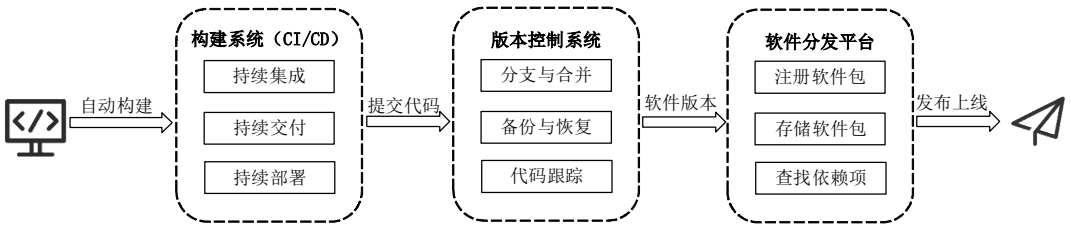

图 5 CI/CD 系统在典型开源开发环境中的位置

3 基于版本控制系统的开源软件供应链攻击

软件版本控制系统是开源软件供应链的重要组成部分，软件版本控制系统允许来自不同地区的开发者在同一个项目上进行协作开发。版本控制系统通过唯一的版本标识符来标记一个文件或一组文件的更改，让所有的开发者在正确的版本中协作。同时，版本控制系统允许开发者回滚到历史的任一版本以减少项目上线出错带来的影响。版本控制系统凭借其历史可追溯、分支与合并、高效率等特性，成为了现代软件开发者不可或缺的工具。常用的分布式版本控制系统的工作原理如图 6 所示。Deepa 等人[33]分析了各种软件版本控制系统的重要性，其研究认为包括 Git 在内的所有版本控制系统都存在一系列的缺陷，通过对比当前市场上主流的版本控制工具，提出了一种新的工具来解决当前版本控制系统存在的安全问题。

软件版本控制系统在开源软件供应链中面临的安全威胁主要是敏感信息泄露。因为软件版本控

制系统中主要存放的是应用程序的源代码，源代码中极有可能含有密钥、密码或凭证等信息。Saha 等人[34]为了更加准确地检测代码托管平台中源代码的敏感信息泄露问题，将正则表达式和机器学习的方法相结合，可以有效的检测源代码中含有的 RSA 私钥、API 密钥、通用密码。Mouw 等人[35]为了研究代码托管平台上的密码泄露滥用问题，故意将 token 泄露到 GitHub、Bitbucket、GitLab 和 SourceForge 等代码托管平台，然后通过监控平台来确定泄露的密码是如何被发现和滥用的。其研究结果表明，仅仅依靠代码托管平台提供的密码扫描功能是不够的。将 git-secrets[36]等工具与 git 结合使用，可以有效的防止将密码提交到托管平台。Farinella 等人[37]为了研究 Git 密码泄露误报率高的问题，首先研究传统检测方法的缺点，然后针对性的给出了一种解决策略。该策略通过收集 URL 和维护者的姓名来识别相关的所有的 Git 代码存储库，然后根据目标优先顺序进行内容检查，该策略可以使得组织生产力不变的同时增加了快速识别密码泄露的机会，但是该策略缺乏对生产环境中的方法进行分析。

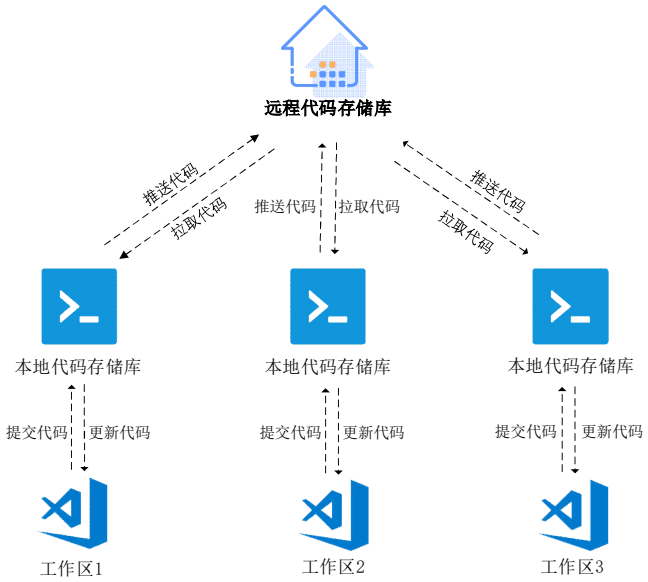

图 6 分布式版本控制系统的工作原理

DevOps 是一种确保软件开发过程中开发和运营之间沟通和协作的实践集合，应用 DevOps 的团队能够极大的提高沟通效率并缩短软件开发周期，但 DevOps 实践过程中也存在秘密泄露的问题。McNierney 等人[38]研究了开发者和 DevOps 团队在基于云的环境中开发时使用的秘密类型，以及这些秘密如何被攻击者所利用，其研究发现如果在 DevSecOps 部署的过程通过自动化工具和行业标准可以防止敏感信息泄露。

软件工件是软件开发过程中的副产品，代码托管平台中很大一部分源代码来自于软件工件。保护软件工件中的秘密就是间接地保护开源软件供应链安全。Basak 等人[39]通过研究互联网工件中的秘密传播问题，其研究认为当前的软件工件严重依赖于使用机密进行授权和验证，机密泄露问题正在逐渐增加。该研究总结出了针对软件工件的秘密管理实践，其研究表明使用本地环境变量和外部秘密管理服务是安全存储秘密的有效做法。

软件版本控制系统中源代码的机密泄露还和开发者意外的提交有关。Krause 等人[40]使用一种混合方法对 109 名开发者进行了访谈，该研究发现 30.3%的参与者曾经遭遇过秘密泄露事件，开发者在面对这些事件时显得准备不足。其研究认为要想避免版本控制系统中源代码的机密泄露问题开发者应该将秘密外部化，尽量避免将秘密提交到公开的源代码存储库中，一旦发生了秘密泄露，应当尽快更新或撤销泄露的秘密。对于代码托管平台，其研究认为应当改进开发指南来更好地帮助开发者应对秘密泄露问题，同时应当提供并拓展秘密扫描功能。

现有针对版本控制系统中源代码机密泄露的扫描方法主要侧重于检测具有一定格式的秘密，但是却难以识别用于身份验证的文本密码。Feng 等人[41]介绍了 PassFinder 方法，这种方法利用机器学

习中的深度神经网络来探索文本密码的特征和语义，通过这种方法对 GitHub 开源代码存储库撒花姑娘的密码泄露进行了大规模的实证研究，发现在 GitHub 中密码泄露是非常广泛的，其研究表明秘密安全对开源生态系统是十分重要的。

基于版本控制系统的开源软件供应链攻击不是直接攻击版本控制系统，而是从版本控制系统中的项目源代码中获取密钥或凭证信息，进而通过这些密钥针对目标应用程序发起攻击。GitGuardian[42]是面向 DevOps 时代的代码安全平台，它主要的功能是在软件开发全生命周期对秘密进行自动检测和修复。2021 年，GitGuardian 发布了《Github 平台秘密泄露状态报告》显示[43]，在 Github 平台上泄露的秘密主要是授权访问服务、API 密钥、用户名和密码或安全证书等数字身份验证凭据。这些秘密的泄露将使攻击者拥有合法访问的身份，令目标公司极易受到开源软件供应链攻击。因此，开发者在使用代码控制平台时应尽量通过 gitignore 文件忽略掉含有秘密的文件，并开启代码控制平台的密钥扫描提醒功能。公司的秘密管理系统往往不会在软件开发全生命周期生效，开发者尽量不要上传密钥到代码存储库或在明文消息系统传输未加密的秘密，同时还应限制 API 的访问权限。

4 基于软件分发平台的开源软件供应链攻击

开源软件分发平台能够帮助开发者存储和下载第三方开源软件以及开源软件的依赖项。在实际开发过程中，开发者往往借助软件包管理器来和开源软件分发平台进行交互。越来越多的攻击者以软件包管理器为跳板进而攻击开源软件供应链下游的应用程序。不同软件包管理器生态系统中包依赖网络存在的安全问题受到越来越多的学者关注。Decan 等人[44]使用 libraries.io 数据集[45]对七个软件包管理器生态系统中包依赖网络的演变进行了实证分析，提出了增长性、变化性、可恢复性等指标，并使用这些指标来分析目标包管理生态系统包依赖网络的演变。其研究发现依赖网络随着时间的推移，软件包的数量在不断增长并且所有软件包管理器生态系统中大多数软件包都是连接的。在生态系统脆弱性方面，热门发现大多数软件包有较少的直接依赖，但却有大量的传递依赖，其研究认为依赖管理工具需要明确考虑到传递依赖带来的影响。

在实际软件开发过程中，有很多开源软件项目是存在多个不同软件生态系统中的，跨生态系统软件包一旦出现问题，可能对其他生态系统产生严重影响。因此，跨生态系统开源软件包的安全问题是开源软件供应链安全研究中的重要课题。Constantinou 等人[46]对 12 个跨生态系统软件包的存在和特征进行了研究，发现跨生态系统的软件包往往能够支持某些生态系统的新版本，这样的软件包随着时间的演变会影响大量其他软件包。

开源并不意味着绝对的免费，许多开发者认为通过软件分发平台下载的开源软件可以随意使用其源代码，这是一个极大的误区。开发者在应用开源软件的时候应该遵循开源许可证。开源许可证可以允许开发者在一定的条款下使用、分发和修改其源代码、设计或算法等。Makari 等人[47]提出了一个能够帮助开发者评估软件包和依赖项许可一致性的工具，通过该工具研究了 npm 和 RubyGems 软件包生态系统中许可证的演变，发现 7.3%的 npm 包和 13.9%的 RubyGems 的包直接或间接依赖不兼容的许可证。

开源软件包管理器生态系统中的安全问题是当前研究的热点，学者们对包管理器生态系统中的恶意软件包问题和漏洞传播问题比较感兴趣。Vaidya 等人[48]从攻击策略、载体和目标等方面对针对软件包管理器生态系统的安全攻击进行了介绍，并对 npm 和 PyPI 生态系统进行了深入的分析，其研究认为通过完全自动化地方式来检测恶意软件包可行性不高，但是评估外部依赖风险的工具和度量标准却能够很大程度的预防攻击。Alfadel [49]首先研究了软件包中漏洞的生命周期，以此来评估软件生态系统对漏洞的反应。同时，还研究了现有软件生态系统针对漏洞的有效性，其研究表明软件包的漏洞一般需要较长的时间才能被发现，很多依赖外部软件包的应用程序仍然收到公开漏洞的影响，即使这些漏洞已经公布很久。Kestilä[50]对 npm 生态系统和 PyPI 生态系统进行了研究，发现在生态系统中存在着大量传递依赖关系、琐碎的包、更新延迟等问题。这些问题使得开源软件供应链面临恶意代码或漏洞注入的风险，Kestilä 认为要想降低安全风险，需要在软件开发生命周期中使用

最佳实践、OWASP 模型[51]等安全开发手段，可以使用免费或商业的软件组成分析工具。Dam 等人[52]对软件代码存储库和包管理器中漏洞的频率和变化进行了概述，同时还探讨了将公共漏洞和暴露（CVE）中的数据映射到开源库的方法，并给出了流行的编程语言中 CVE 条目的频率和分布。

应用程序中使用过时的依赖项，这种现象是开源软件生态系统中较为常见的技术滞后问题。开源软件包管理器生态系统中的技术滞后问题是生态系统安全漏洞的主要来源。开源软件包管理器生态系统中大量使用了过时的软件包版本，虽然这样不会带来兼容问题，但却可能导致漏洞等问题。Zerouali 等人[53]提出了一个技术滞后的通用模型，并用这个模型量化了目标组件集合相对于理想部署的滞后情况。同时，其研究利用提出的模型实证研究了 npm、Debian 和 Docker 生态系统中滞后问题的演变。Stringer 等人[54]以 libraries.io 数据集中 14 个包管理器的开源项目数据作为数据基础，对主要软件包管理器的技术滞后问题进行了大规模的实证研究。其研究发现当前大多数固定版本的声明和很多灵活版本声明都已经过时，并且固定版本声明一般不会被定期更新，这很容易就会产生滞后问题。要想消除软件包管理器中普遍存在的技术滞后问题，可以使用基于语义版本控制的声明范围。

为了获取目标系统的访问权限，很多攻击者在软件供应链的节点中安装木马或后门来实现其篡改软件的目的。Levy[55]认为相较于专有软件，开源软件在分发过程中容易收到攻击，因为在这个过程中可以用少量的投入获取巨大的利益，现有开源供应商应该采用数字签名等技术来规避这种风险。

基于软件分发平台的开源软件供应链攻击，往往和软件包管理器生态系统存在的安全问题有关。软件包管器生态系统存在的技术滞后问题、漏洞传播问题、传递依赖漏洞问题、开源许可证滥用问题等都使得开源软件供应链攻击成为可能。

5 开源软件供应链攻击防御策略

开源软件供应链中任何组件的安全风险，都可能给软件供应链下游的应用程序带来不可预知的风险。开源软件供应链攻击事件带来的安全影响正在与日俱增，如何高效的防御开源软件供应链攻击已经成为开源软件供应链上下游不同规模组织必须考虑的问题。本文以典型开源开发环境中构建系统、版本控制系统、软件分发平台、开发者和终端用户的视角出发，给出了针对性的开源软件供应链攻击防御策略。

5.1 开发者视角

开发者是与应用程序源代码距离最近的人员，开发者应当遵循安全的代码设计原则，尽量编写正确且没有漏洞的源代码。Ellison 等人[56]认为降低软件供应链风险最有效的方法是将安全性纳入软件开发过程，并使用模糊测试等技术来寻找高风险的漏洞。开发者需要接收软件设计、编码和测试实践等方面的安全测试。开发管理团队还应防止内部开发者有意或无意的在应用程序生产代码中注入恶意代码。Eggers[58]认为降低网络供应链风险的关键是建立完整准确的软件物料清单，通过缩小软件物料清单可以有效的减少软件供应链的攻击面。Faruk 等人[59]认为确保软件供应链安全最常见的方法是通过质量保证（QA）、软件开发生命周期和软件供应链管理系统等的密切合作。产品在投入生产之前，应当由独立的第三方进行安全评估，以防止最终产品中出现安全问题。

开发者在对源代码进行测试的时候应该执行静态或动态的安全漏洞扫描来检测可能发生的漏洞。在面对可能包含漏洞的第三方开源组件时，应定期更新开源组件到最新版本以防止可能出现的漏洞问题。使用软件组成分析工具自动化地对第三方开源组件进行公开漏洞扫描，确保当前应用程序所依赖的第三方开源组件没有公开漏洞。列出应用程序中使用到的软件物料清单，通过自动化地漏洞扫描服务对列表上的组件进行持续的监控。自动化地提取软件物料清单可以节省大量人工劳动，特别是当软件规模很大或更新迭代频繁的时候。手动维护物料清单的成本会非常高，而自动化提取可以有效降低这部分成本。

开发者还应注意开发环境的安全性，安装在开发机器上的所有工具、系统等都需经过漏洞扫描和数据权限的批准。Germonprez 等人[60]认为参与开源社区时一定要重视开源许可证的合规问题。开

发者在注重安全性的同时，还应考虑其开源许可证是否合法。

开发者借助软件包管理器下载依赖项时往往会引入大量的运行时依赖，这些运行时依赖的安全问题却经常被开发者所忽略。Dey 等人[61]使用线性回归和随机森林等机器学习模型对 12999 个 NPM 软件包进行了研究，其研究表明上游运行时依赖项的下载量和数量与下游运行时依赖项的数量存在着正比关系，因此开发者在安装依赖项时应当重视运行时依赖项的安全问题。

5.2 构建系统视角

开源软件供应链的构建系统应当有严格的安全性和完整性控制，以防止攻击者通过受损的构建系统来注入恶意代码、修改配置文件等方式劫持系统进而发起软件供应链攻击。Ellison 等人[62]认为随着恶意软件复杂性的增加，软件承包商必须具备管理供应链风险的能力。Alberts 等人[63]发现软件供应链的风险是相互依存的，软件供应链风险评估必须在系统体系环境下进行，因此构建系统还应基于系统体系环境进行风险评估。列出软件从开发到部署所涉及到的所有构建工具清单，并确保这些构建工具的安全性。Sabbagh 等人[64]在研究软件供应链威胁建模的社会技术框架时发现内部员工或未经授权的个人存在将恶意代码插入软件源代码的可能性。Buchicchio 等人[65]在研究已知木马源攻击技术时，认为最简单、最有效的软件供应链保护措施是实施安全开发生命周期（SDL）策略。全面了解构建系统在不同阶段所拥有的软件代码读写权限，以防止未经授权的高敏感数据的修改或访问。对应用程序中的数据权限进行分级管理，访问高敏感的数据必须经过一定的授权。恶意代码注入可能发生在构建系统的任何环节，需要在软件开发生命周期的每一个阶段进行有效的监控，以防止恶意代码注入影响到生产环境下的代码。Heinbockel 等人[66]研究发现在应用程序生产和部署阶段应用网络弹性缓解措施可以减少整个攻击的数量。因此，构建系统可以在合适的场景应用网络弹性缓解措施以减少软件供应链攻击。

5.3 版本控制系统视角

版本控制系统环节最大的安全问题是密钥或凭证信息的泄露，攻击者通过扫描大量的开源代码仓库，从源代码中找到被开发者不经意间上传的密钥等信息，进而通过这些密钥发起开源软件供应链攻击。这些密钥的泄露将目标应用程序暴露在攻击者面前，因此版本控制系统最重要的就是要避免密钥等高度敏感信息的泄露。

Hensley 等人[67]认为攻击者在获得一定的访问权限后，可能会破坏身份验证机制，以访问托管在云上的敏感信息。开发者在将源代码上传到远程代码托管平台之前，首先要手动检查相关配置文件中是否存在可能被上传的密钥信息，如果存在则忽略掉这些文件的上传，上传到远程代码托管平台之后，尽量使用目标平台的密钥扫描功能进行扫描，以防止深层次源代码中包含的密钥信息。在平时开发时，开发者还应尽量避免在聊天工具中直接传输高度敏感的密钥信息，以防止密钥不经意间的泄露。Reed 等人[68]认为软件全生命周期都应防止攻击者对关键硬件或固件组件的篡改，因此远程版本控制系统应采取多种防篡改设备防止远程代码被恶意修改。软件开发生命周期全流程安全预警可以在软件开发早期就检测和预警安全漏洞,不必等到软件部署和交付之后才发现问题,这可以最大限度地减少因为安全问题而需要回滚版本或补丁的情况。同时，软件开发生命周期全流程安全预警还可以及时发现可能的供应链攻击,比如恶意代码注入、后门植入等,在代码合并和组件引入过程中就可以监测和报警,减少这类攻击的成功几率。

Yeboah-Ofori 等人[69]使用贝叶斯信念网络来检测网络物理系统中的网络供应链攻击，其研究结果表明贝叶斯信念网络可以用于 CSC 领域发生的网络攻击检测。版本控制系统可以借助类似的技术对其远程代码库中的源代码进行检测。

实时检测上传到远程代码托管平台的恶意工件是软件供应链安全研究的难点。Vu 等人[70]使用源代码库来检测软件包恶意代码注入问题，其研究提出的方法可以捕获将恶意代码注入合法软件包的行为，远程代码托管平台可以借助类似技术对其平台下的软件包进行分析，以防止恶意代码注入。Martínez 等人[71]认为应对网络供应链攻击需要加强身份验证，可以使用类似单点登录（SSO）的方式对用户的身份进行验证。

5.4 软件分发平台视角

软件分发平台最重要的是防止出现依赖混淆攻击或恶意依赖项。依赖混淆攻击指的是在应用程序中多个依赖项出现冲突时，目标系统不知道该使用哪一个依赖项，便会向公开库中寻找依赖关系，攻击者此时便会发动依赖混淆攻击。Herr[72]认为现代软件产品包含了大量的依赖项，对这些依赖项进行跟踪是非常重要的技术成就。Zhang 等人[73]发现越来越多的攻击者把软件开发者和供应商当做攻击目标，以寻求对其源代码进行修改，在构建和更新过程中安装恶意软件或后门。Duan 等人[74]运用元数据、静态和动态分析等技术研究了软件包管理器生态系统中的注册表滥用问题，其研究发现了 339 个恶意软件包，其中有的恶意软件包下载次数甚至超过 100000 次。Zahan 等人[75]认为软件包在多年内没有更新或维护者在数月内没有更新等行为是软件供应链存在安全漏洞的信号。软件分发平台应对上传到公共存储库的第三方开源组件进行软件组成分析，以确保上传软件包的安全性。软件分发平台还应实时监控公共存储库中的软件包的最新版本是否受公共漏洞库中的漏洞影响，发现受漏洞影响的软件包应及时通知开发者更新到不受影响的版本。第三方开源组件被广泛使用,但其安全风险较高,无法自主控制,依赖监控可以实时知晓这些组件的安全状态,及时采取措施防范风险。开源依赖智能监控技术可以实现依赖漏洞通知,监控工具可以订阅各开源项目的漏洞通知,一旦依赖项目有安全漏洞,可以主动通知,使软件项目及时知晓并采取行动,这对软件安全至关重要。当普通用户下载了受漏洞影响的软件包时，软件分发平台可以通过包管理器等工具发出预警信息。Benthall 等人[76]演示了评估整个软件项目生态系统的方法，因此软件分发平台还可以考虑根据自身生态系统的技术结构来设计风险评估模型。Forte 等人[77]认为软件分发模型已经从传统的物理模型转化为基于互联网的分布式数字模型，攻击者可以通过侵入软件包的分发站点，让不知情的用户去安装恶意软件包。

区块链技术凭借其透明、去中心化等特性被广泛应用在诸多领域，区块链也可以成为实现软件供应链安全的解决思路。Sani 等人[78]提出一种基于区块链的身份管理系统来减轻供应链攻击，通过使用椭圆曲线数字签名算法来构造身份，并使用安全伪随机函数来导出组件。研究结果表明基于区块链的身份管理系统对供应链攻击具有一定的弹性。软件分发平台可以基于这种区块链身份管理的思想来管理数量庞大的开源组件。

借助机器学习等人工智能技术来检测软件供应链攻击已经成为了一种趋势。Ohm 等人[79]基于集群的特征代码片段进行无监督生成签名，然后用签名去扫描整个 npm 注册表中未发现的软件包，通过这个方法发现了 6 个恶意软件包。Vasilakis 等人[80]将主动学习和再生技术（ALR）相结合，用于推断组件在特定领域的行为模型，其研究方法可以消除特定的供应链漏洞。Masum 等人[81]使用量子支持向量机和量子神经网络等 QML 方法来检测软件供应链攻击，虽然该方法与传统的方法相比有更高的计算时间，但却给检测软件供应链攻击提供了新思路。软件分发平台可以将自身技术结构与机器学习技术相结合，检测软件注册表中存在的恶意软件包，以减少人工工作量。

5.5 终端用户视角

终端用户是开源软件供应链的下游用户，也是最容易收到开源软件供应链攻击影响的组织。终端用户预防开源软件供应链攻击的最佳手段是对目标应用程序进行全面的信息掌握。对应用程序权限进行合理的分类管理，以防止不可控的越权攻击。同时，定期更新应用程序到最新版本，以防止应用程序受到公开漏洞的影响。指定针对软件供应链攻击的响应流程，确保软件供应链攻击一旦发生能够快速进行处理，以减少可能带来的不利影响。

Liu 等人[82]描述了在硬件中预留后门从而发起软件供应链攻击的可能性，并提出了基于嵌入式可重构逻辑实现 ASIC 设计混淆的方式来预防针对软件供应链的程序监视木马攻击。终端用户还可以考虑将基于软件的安全解决方案迁移到基于硬件的安全解决方案。Pandey[83]等人认为要想减轻全球化供应链中的网络安全风险应该为普通用户设置严格的密码和账户管理政策，以防止内部人员泄密或滥用数据。

在没有软件供应商参与的情况下进行软件供应链攻击检测对终端用户来说是非常重要的。

Wang[84]等人将流量异常检测和软件包标记技术相结合，可以在没有任何攻击先验知识的情况下检测出软件供应链攻击。研究结果表明，在生产环境下基于攻击信息流的检测非常准确。Ludvigsen 等人[85]在研究软件供应链攻击相关的法律法规时，发现现有的法律法规在防止供应链攻击方面存在很多不足。Kshetri[86]在研究软件供应链攻击中的经济学问题时发现仅靠法规是难以消除软件供应链中存在的违法行为的，应该对供应链的所有合作伙伴进行全面的调查，以确保上下游厂商都能遵守特定的网络安全标准。因此，终端用户在使用应用程序之前可以通过各种技术手段对软件供应链攻击进行检测。

在典型开源开发环境下，开源软件供应链上下游不同组织防御开源软件供应链攻击的策略如表 2 所示。

表 2 开源软件供应链上下游不同组织防御策略

视角	攻击方式	防御策略
开发者视角	1. 在生产代码中注入恶意代码 2. 不执行安全漏洞扫描 3. 使用不合规的开源许可证 4. 预留木马后门	1. 遵循安全代码设计原则 2. 建立完整准确的软件物料清单 3. 执行静态或动态的安全漏洞扫描 4. 避免开源许可证违规
构建系统视角	1. 修改配置文件 2. 未经授权对数据进行访问或修改 3. 在部署阶段修改源代码	1. 列出开发到部署所有构建工具清单 2. 实施安全开发生命周期策略 3. 基于系统体系环境进行风险评估 4. 对数据权限进行分级管理
版本控制系统视角	1. 密钥或凭证信息窃取 2. 破坏身份验证机制 3. 源代码恶意篡改	1. 手动检查要上传的配置文件 2. 使用目标平台的密钥扫描功能 3. 避免直接传输敏感密钥 4. 采用多种防篡改设备
软件分发平台视角	1. 依赖混淆攻击 2. 上传恶意软件包诱导下载 3. 侵入软件分发平台 4. 利用含有公开漏洞的软件包	1. 跟踪监控第三方依赖项 2. 设计风险评估模型 3. 应用软件供应链攻击检测技术 4. 避免注册表滥用
终端用户视角	1. 越权攻击 2. 预留后门攻击 3. 利用恶意软件版本发起攻击	1. 定期更新应用程序到最新版本 2. 设置严格的密码以及账户管理政策 3. 对应用程序进行全面的信息掌握 4. 对应用程序权限进行分类

6 未来研究方向

基于上述研究，本文总结了开源软件供应链攻击及防御未来可行的研究方向及发展趋势，并讨论了在这些研究方向中如何软件物料清单应用新技术。开源软件供应链攻击及防御未来研究方向及其细分领域如表 3 所示。

（1） 软件物料清单（SBOM）智能提取技术

软件物料清单指的是构成目标软件的组件、依赖关系以及其他相关文件的清单。软件物料清单不仅包括开源软件还包括专有软件。一份完整详实的软件物料清单能够帮助组织降低开发成本、减少安全风险和法律风险。开源软件供应链上下游的组织都或多或少的依赖于开源软件，因此上下游的组织都应拥有属于自己的软件物料清单。当目标组件发现安全缺陷时，软件物料清单可以帮助相关人员快速定位受漏洞影响的软件范围，并进行合理的威胁评估。软件物料清单还可以帮助开发者规避潜在的许可证法律风险。结合机器学习、软件组成分析等方法智能分析提取软件物料清单是研

究软件物料清单智能提取的新思路。

（2） 软件开发生命周期全流程安全预警技术

完整的软件开发生命周期包括计划、需求分析、软件设计、软件构建、文档撰写、测试、部署、运维等多个阶段。软件开发生命周期的任一环节都有可能出现安全问题，进而影响到整个软件供应链的安全。如何将安全预警集成到软件发开生命周期的每一个环节是一个极具挑战的任务。在软件设计和规划阶段就应详细评估安全风险、将安全要求作为软件的功能要求，软件开发时应注重对编码人员的安全教育，在测试阶段通过静态测试或动态测试的方式对软件进行详细的安全测试，部署时对配置文件和部署环境进行安全评估，运营时进行实时监控以尽早发现安全威胁并准备好应急预案。因此，通过智能化、自动化等方式在软件开发全生命周期内集成安全预警具有重要意义。

（3） 第三方开源依赖智能监控技术

开源软件供应链面临的最大安全风险是存在安全问题的第三方开源软件包，基于第三方开源依赖发起开源软件供应链攻击的案例层出不穷。第三方开源依赖智能监控技术通过分析项目的依赖关系图和使用到的第三方开源库，构建项目的软件成品清单(SBOM)。同时，定期对第三方开源库的更新、安全问题及漏洞公告进行跟踪，并对项目 SBOM 中的对应组件版本进行评估，发现潜在的安全风险并提出升级提示。第三方开源依赖智能监控技术还可以通过构建第三方开源库安全风险知识图谱，实现对风险点的智能标注、关联发现及攻击方式推测。这有助于安全研究人员进行深入分析，并提出针对性的修复建议。

（4） 开源软件供应链多层防御技术

保护开源软件供应链的安全需要软件供应链上下游不同组织的密切配合。无论是开发者、构建系统、软件分发平台、版本控制系统都面临着软件供应链攻击的威胁，不同组织的应对方式也各有不同。一个全方位、多层次且具有较高适应性的防御框架是开源软件供应链迫切需要的。开源软件供应链多层防御技术是指在开源软件供应链关键环节引入多种安全防御策略。多层防御技术在供应链的每个环节都引入必要的安全控制，比如代码审计、漏洞扫描、依赖监控、发布包校验等,通过在多个环节实现安全防护，可以进一步降低供应链攻击和其他安全事故对最终用户的影响，尤其是对依赖开源软件的用户来说，供应链多层防御是很重要的安全保证。

表 3　开源软件供应链攻击及防御未来研究方向及其细分领域

未来研究方向	研究方向细分
软件物料清单（SBOM）智能提取技术	1. 智能软件组成分析技术 2. 软件物料清单风险管理技术 3. 软件物料清单可视化与评估
软件开发生命周期全流程安全预警技术	1. 软件开发生命周期安全指标评估 2. 软件开发生命周期模糊测试技术 3. 软件开发生命周期动态预警技术
开源依赖智能监控技术	1. 开源威胁情报自动分析技术 2. 开源软件生态系统漏洞传播分析 3. 开源依赖威胁识别框架设计
开源软件供应链多层防御技术	1. 开源软件供应链防御策略研究 2. 开源软件供应链风险管理技术 3. 基于硬件的软件供应链防御技术

7　结束语

近年来开源软件供应链攻击事件层出不穷，开源软件供应链上下游组织面临着恶意代码注入、

预留木马后门、许可证违规、密钥凭证泄露、依赖混淆攻击、漏洞传播等安全风险。开源软件供应链攻击随时间的演变及针对性地防御策略已经成为了当前研究的热点。本文归纳整理了软件供应链攻击及防御研究领域的相关文献，详细介绍了开源软件供应链攻击的技术演变以及对应的防御策略。为了方便读者阅读，本文首先给出了开源软件供应链中一些专业术语定义，然后详细回顾了近年来发生的经典开源软件供应链攻击事件。本文立足于经典开源开发环境，系统分析了基于构建系统、版本控制系统、软件分发平台的开源软件供应链攻击及防御策略。最后本文从软件供应链上下游不同组织的视角出发，给出了针对性的防御策略并对未来的研究方向进行了展望。

参考文献：

[1] Boehmke B C, Hazen B T. The future of supply chain information systems: The open source ecosystem[J]. Global Journal of Flexible Systems Management, 2017, 18(2): 163-168.

[2] Synopsys. 2022 Open Source Security and Risk Analysis (OSSRA) Report. [EB/OL]. [2023-02-01] https://www.synopsys.com/zh-cn/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html.

[3] 何熙巽, 张玉清, 刘奇旭. 软件供应链安全综述[J]. Journal of Cyber Security 信息安全学报, 2020, 5(1).
He Xixun, Zhang Yuqing, Liu Qixu. A review of software supply chain security [J]. Journal of Cyber Security Information Security Journal, 2020, 5(1).

[4] 纪守领, 王琴应, 陈安莹, 等. 开源软件供应链安全研究综述[J]. 软件学报, 2022: 0-0.
Ji Shou-Ling, Wang Qin-Ying, Chen An-Ying, et al. A review of open source software supply chain security research[J]. Journal of Software, 2022: 0-0.

[5] Ladisa P, Plate H, Martinez M, et al. SoK: Taxonomy of Attacks on Open-Source Software Supply Chains[C]//2023 IEEE Symposium on Security and Privacy (SP). IEEE Computer Society, 2022: 167-184.

[6] Ohm M, Plate H, Sykosch A, et al. Backstabber's knife collection: A review of open source software supply chain attacks[C]//International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, Cham, 2020: 23-43.

[7] Wikipedia. Wikipedia on Open Source Software Definition. https://en.wikipedia.org/wiki/Open-source_software.

[8] Wikipedia. Open Source License Definition. https://en.wikipedia.org/wiki/Open-source_license.

[9] Kshetri N, Voas J. Supply chain trust[J]. IT Professional, 2019, 21(2): 6-10.

[10] Unredacted. Update: Agent Farewell and the Siberian Pipeline Explosion. [EB/OL]. [2023-02-01] https://unredacted.com/2013/04/26/agent-farewell-and-the-siberian-pipeline-explosion/.

[11] OWAIS SULTAN. The cracking of kernel.org. https://www.hackread.com/kernel-org-linux-foundation-hacker-caught/.

[12] Oscar Celestino Angelo Abendan ll. Trend Micro Investigates June 25 Cyber Attacks in South Korea. https://www.trendmicro.com/vinfo/us/threat-encyclopedia/web-attack/124/trend-micro-investigates-june-25-cyber-attacks-in-south-korea.

[13] WOODROW HARTZOG AND DANIEL J. SOLOVE. We Still Haven't Learned the Major Lesson of the 2013 Target Hack. [EB/OL]. [2023-02-01] https://slate.com/technology/2022/04/breached-excerpt-hartzog-solove-target.html.

[14] Thomas Reed. Transmission hijacked again to spread malware. [EB/OL]. [2023-02-01] https://www.malwarebytes.com/blog/news/2016/09/transmission-hijacked-again-to-spread-malware.

[15] Edmund Brumaghin、 Warren Mercer、 Craig Williams. CCleanup: A Vast Number of Machines at Risk. [EB/OL]. [2023-02-01] https://blog.talosintelligence.com/avast-distributes-malware/.

[16] Wikipedia. 2017 Ukraine ransomware attacks. https://en.wikipedia.org/wiki/2017_Ukraine_ransomware_attacks.

[17] Microsoft. Attack inception: Compromised supply chain within a supply chain poses new risks. [EB/OL]. [2023-02-01] https://www.microsoft.com/en-us/security/blog/2018/07/26/attack-inception-compromised-supply-chain-within-a-supply-chain-poses-new-risks/.

[18] DAN GOODIN. Two new supply-chain attacks come to light in less than a week. [EB/OL]. [2023-02-01] https://arstechnica.com/information-technology/2018/10/two-new-supply-chain-attacks-come-to-light-in-less-than-a-week.

[19] Zack Whittaker. Hackers dropped a secret backdoor in Asus' update software. [EB/OL]. [2023-02-02] https://techcrunch.com/2019/03/25/asus-update-backdoor/.

[20] npm Blog (Archive). Plot to steal cryptocurrency foiled by the npm security team. [EB/OL]. [2023-02-03] https://blog.npmjs.org/post/185397814280/plot-to-steal-cryptocurrency-foiled-by-the-npm.html.

[21] Sergiu Gatlan. New Octopus Scanner malware spreads via GitHub supply chain attack. [EB/OL]. [2023-03-01] https://www.bleepingcomputer.com/news/security/new-octopus-scanner-malware-spreads-via-github-supply-chain-attack/.

[22] Anton Cherepanov. Lazarus supply-chain attack in South Korea. [EB/OL]. [2023-02-04] https://www.welivesecurity.com/2020/11/16/lazarus-supply-chain-attack-south-korea/.

[23] Jake Williams. What You Need to Know About the SolarWinds Supply-Chain Attack. [EB/OL]. [2023-02-05] https://www.sans.org/blog/what-you-need-to-know-about-the-solarwinds-supply-chain-attack/.

[24] Ax Sharma. Researcher hacks over 35 tech firms in novel supply chain attack. [EB/OL]. [2023-02-06] https://www.bleepingcomputer.com/news/security/researcher-hacks-over-35-tech-firms-in-novel-supply-chain-attack/.

[25] Ax Sharma. $3 Million Cryptocurrency Heist Stemmed From a Malicious GitHub Commit. [EB/OL]. [2023-02-07] https://blog.sonatype.com/3-million-cryptocurrency-heist-malicious-github-commit.

[26] Ax Sharma. npm Libraries 'colors' and 'faker' Sabotaged in Protest by their Maintainer—What to do Now? https://blog.sonatype.com/npm-libraries-colors-and-faker-sabotaged-in-protest-by-their-maintainer-what-to-do-now

[27] Lipke S. Building a secure software supply chain using docker[D]. Master's thesis. Hochschule der Medien, Stuttgart, Germany, 2017.

[28] Ohm M, Sykosch A, Meier M. Towards detection of software supply chain attacks by forensic artifacts[C]//Proceedings of the 15th international conference on availability, reliability and security. 2020: 1-6.

[29] Wenhao J, Zheng L. Vulnerability analysis and security research of Docker container[C]//2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE). IEEE, 2020: 354-357.

[30] Brady K, Moon S, Nguyen T, et al. Docker container security in cloud computing[C]//2020 10th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2020: 0975-0980.

[31] Benedetti G, Verderame L, Merlo A. Automatic Security Assessment of GitHub Actions Workflows[C]//Proceedings of the 2022 ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses. 2022: 37-45.

[32] Bajpai P, Lewis A. Secure Development Workflows in CI/CD Pipelines[C]//2022 IEEE Secure Development Conference (SecDev). IEEE, 2022: 65-66.

[33] Deepa N, Prabadevi B, Krithika L B, et al. An analysis on version control systems[C]//2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE). IEEE, 2020: 1-9.

[34] Saha A. Detecting Secrets in Source Code: Using Machine Learning to Reduce False Positives[D]. The University of Utah, 2019.

[35] Mouw M, Cox M, Mijnen F. Profiling abuse of exposed secrets in public repositories[J]. 2020.

[36] git-secrets [EB/OL]. [2022-12-21] https://github.com/awslabs/git-secrets.

[37] Farinella C, Ahmed A, Watterson C. Git Leaks: Boosting Detection Effectiveness Through Endpoint Visibility[C]//2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE, 2021: 701-709.

[38] McNierney S F. Securing DevOps Environments in the Cloud[D]. Utica College, 2021.

[39] Basak S K, Neil L, Reaves B, et al. What are the Practices for Secret Management in Software Artifacts?[C]//2022 IEEE Secure Development Conference (SecDev). IEEE, 2022: 69-76.

[40] Krause A, Klemmer J H, Huaman N, et al. Committed by Accident: Studying Prevention and Remediation Strategies Against Secret Leakage in Source Code Repositories[J]. arXiv preprint arXiv:2211.06213, 2022.

[41] Feng R, Yan Z, Peng S, et al. Automated Detection of Password Leakage from Public GitHub Repositories[C]//International Conference on Software Engineering (ICSE'22). 2022.

[42] GitGuardian [EB/OL]. [2022-12-22] https://www.gitguardian.com/

[43] GitGuardian. The state of Secrets Sprawl on GitHub. [EB/OL]. [2023-02-01] https://www.gitguardian.com/state-of-secrets-sprawl-on-github-2021#map

[44] Decan A, Mens T, Grosjean P. An empirical comparison of dependency network evolution in seven software packaging ecosystems[J]. Empirical Software Engineering, 2019, 24(1): 381-416.

[45] Libraries.io. Libraries.io dataset. [EB/OL]. [2023-02-02] https://libraries.io/

[46] Constantinou E, Decan A, Mens T. Breaking the borders: an investigation of cross-ecosystem software packages[J]. arXiv preprint arXiv:1812.04868, 2018.

[47] Makari I S, Zerouali A, De Roover C. Prevalence and Evolution of License Violations in npm and RubyGems Dependency Networks[C]//International Conference on Software and Software Reuse. Springer, Cham, 2022: 85-100.

[48] Vaidya R K, De Carli L, Davidson D, et al. Security issues in language-based sofware ecosystems[J]. arXiv preprint arXiv:1903.02613, 2019.

[49] Alfadel M. Assessing and Enhancing the Security of Software Packages[D]. Concordia University, 2021.

[50] Kestilä R. ACKNOWLEDGING THE RISKS OF OPEN SOURCE DEPENDENCIES TO SOFTWARE SUPPLY CHAIN SECURITY[J]. 2022.

[51] Wikipedia. What is OWASP. [EB/OL]. [2023-02-01] https://en.wikipedia.org/wiki/OWASP

[52] Dam T, Neumaier S. Challenges of mapping Vulnerabilities and Exposures to Open-Source Packages[J]. arXiv preprint arXiv:2206.14527, 2022.

[53] Zerouali A. A measurement framework for analyzing technical lag in open-source software ecosystems[D]. PhD thesis, University of Mons, 2019.

[54] Stringer J, Tahir A, Blincoe K, et al. Technical lag of dependencies in major package managers[C]//2020 27th Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2020: 228-237.

[55] Levy E. Poisoning the software supply chain[J]. IEEE Security & Privacy, 2003, 1(3): 70-73.

[56] Ellison R J, Woody C. Supply-chain risk management: Incorporating security into software development[C]//2010 43rd Hawaii International Conference on System Sciences. IEEE, 2010: 1-10.

[57] Ellison R J, Goodenough J B, Weinstock C B, et al. Evaluating and mitigating software supply chain security risks[R]. Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, 2010.

[58] Eggers S. A novel approach for analyzing the nuclear supply chain cyber-attack surface[J]. Nuclear Engineering and Technology, 2021, 53(3): 879-887.

[59] Faruk M J H, Tasnim M, Shahriar H, et al. Investigating Novel Approaches to Defend Software Supply Chain Attacks[C]//2022 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, 2022: 283-288.

[60] Germonprez M, Young B, Warner B, et al. Mitigation in Corporate Management of Open Source Community Participation: Protection and Compliance in an Open Source Supply Chain 5[C]//7th Pre-ICIS International Research Workshop on Information Technology Project Management (IRWITPM 2012). 89.

[61] Dey T, Mockus A. Are software dependency supply chain metrics useful in predicting change of popularity of npm packages?[C]//Proceedings of the 14th international conference on predictive models and data analytics in software engineering. 2018: 66-69.

[62] Ellison R J, Alberts C, Creel R, et al. Software supply chain risk management: From products to systems of systems[R]. Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, 2010.

[63] Alberts C J, Dorofee A J, Creel R, et al. A systemic approach for assessing software supply-chain risk[C]//2011 44th Hawaii International Conference on System Sciences. IEEE, 2011: 1-8.

[64] Al Sabbagh B, Kowalski S. A socio-technical framework for threat modeling a software supply chain[J]. IEEE Security & Privacy, 2015, 13(4): 30-39.

[65] Buchicchio E, Grilli L, Capobianco E, et al. Invisible Supply Chain Attacks Based on Trojan Source[J]. Computer, 2022, 55(10): 18-25.

[66] Heinbockel W J, Laderman E R, Serrao G J. Supply chain attacks and resiliency mitigations[J]. The MITRE Corporation, 2017: 1-30.

[67] Hensley B. Identity is the new perimeter in the fight against supply chain attacks[J]. Network Security, 2021, 2021(7): 7-9.

[68] Reed M, Miller J F, Popick P. Supply chain attack patterns: Framework and Catalog[J]. Office of the Deputy Assistant

Secretary of Defense for Systems Engineering: Washington, DC, USA, 2014.

[69] Yeboah-Ofori A, Islam S, Brimicombe A. Detecting cyber supply chain attacks on cyber physical systems using Bayesian belief network[C]//2019 International Conference on Cyber Security and Internet of Things (ICSIoT). IEEE, 2019: 37-42.

[70] Vu D L, Pashchenko I, Massacci F, et al. Towards using source code repositories to identify software supply chain attacks[C]//Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020: 2093-2095.

[71] Martínez J, Durán J M. Software supply chain attacks, a threat to global cybersecurity: SolarWinds' case study[J]. International Journal of Safety and Security Engineering, vol, 2021, 11(5): 537-545.

[72] Herr T. Breaking Trust–Shades of Crisis Across an Insecure Software Supply Chain[J]. 2021.

[73] Zhang H, Nakamura T, Sakurai K. Security and trust issues on digital supply chain[C]//2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech). IEEE, 2019: 338-343.

[74] Duan R, Alrawi O, Kasturi R P, et al. Towards measuring supply chain attacks on package managers for interpreted languages[J]. arXiv preprint arXiv:2002.01139, 2020.

[75] Zahan N, Zimmermann T, Godefroid P, et al. What are weak links in the NPM supply chain?[C]//Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice. 2022: 331-340.

[76] Benthall S, Pinney T, Herz J C, et al. An ecological approach to software supply chain risk management[C]//15th Python in Science Conference. 2016.

[77] Forte D, Perez R, Kim Y, et al. Supply-chain security for cyberinfrastructure [Guest editors' introduction][J]. Computer, 2016, 49(8): 12-16.

[78] Sani A S, Yuan D, Meng K, et al. Idenx: A Blockchain-based Identity Management System for Supply Chain Attacks Mitigation in Smart Grids[C]//2020 IEEE Power & Energy Society General Meeting (PESGM). IEEE, 2020: 1-5.

[79] Ohm M, Kempf L, Boes F, et al. Supporting the Detection of Software Supply Chain Attacks through Unsupervised Signature Generation[J]. arXiv preprint arXiv:2011.02235, 2020.

[80] Vasilakis N, Benetopoulos A, Handa S, et al. Supply-chain vulnerability elimination via active learning and regeneration[C]//Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. 2021: 1755-1770.

[81] Masum M, Nazim M, Faruk M J H, et al. Quantum Machine Learning for Software Supply Chain Attacks: How Far Can We Go?[J]. arXiv preprint arXiv:2204.02784, 2022.

[82] Liu B, Wang B. Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks[C]//2014 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2014: 1-6.

[83] Pandey S, Singh R K, Gunasekaran A, et al. Cyber security risks in globalized supply chains: conceptual framework[J]. Journal of Global Operations and Strategic Sourcing, 2020.

[84] Wang X. On the Feasibility of Detecting Software Supply Chain Attacks[C]//MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM). IEEE, 2021: 458-463.

[85] Ludvigsen K R, Nagaraja S, Daly A. Preventing or Mitigating Adversarial Supply Chain Attacks: A Legal Analysis[C]//Proceedings of the 2022 ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses. 2022: 25-34.

[86] Kshetri N. Economics of supply chain cyberattacks[J]. IT professional, 2022, 24(3): 96-100.