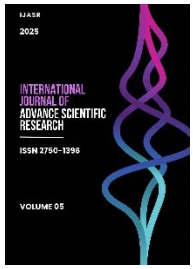
International Journal of Advance Scientific Research
(ISSN – 2750-1396)
VOLUME 05 ISSUE 10 Pages: 155-161
OCLC – 1368736135



Journal Website: http://sciencebring.com/index.php/ijasr

Research Article

Enhancing Software Supply Chain Security: Comprehensive Analysis of Vulnerabilities, Dependency Management, and SBOM Strategies

Submission Date: October 02, 2025, **Accepted Date:** October 15, 2025,
Published Date: October 31, 2025

John H. Whitaker
Department of Computer Science, University of Edinburgh, United Kingdom

Abstract

The rapid proliferation of open-source software and the extensive integration of third-party dependencies in contemporary software ecosystems have transformed the software supply chain into a critical area of cybersecurity concern. The increasing complexity of these ecosystems exposes software products to diverse vulnerabilities, including dependency-based attacks, misconfigurations, and targeted supply chain compromises. This research presents a comprehensive analysis of software supply chain security, focusing on the theoretical and practical challenges of vulnerability assessment, dependency management, and the implementation of Software Bill of Materials (SBOM) solutions. By synthesizing findings from recent empirical studies and historical analyses, the paper elucidates the structural and operational weaknesses in open-source ecosystems, such as npm and Apache, and examines the efficacy of automated vulnerability detection, SBOM generation, and LLM-based vulnerability sourcing. This study adopts a descriptive methodological framework, leveraging theoretical constructs like attack trees, dependency graphs, and supply chain threat modeling to systematically evaluate the current security landscape. Findings indicate that while tools for automated vulnerability scanning and SBOM generation provide measurable improvements in visibility and traceability, they are constrained by limitations in detection accuracy, dependency coverage, and the dynamic nature of software evolution. Additionally, the research highlights the sociotechnical dimensions of software supply chain security, emphasizing the roles of developer practices, community governance, and stakeholder perceptions in mitigating risk. The discussion integrates empirical insights with conceptual analysis to propose strategic and operational recommendations, including standardized SBOM practices, continuous dependency monitoring, and the integration of AI-assisted vulnerability identification within development workflows. This paper

International Journal of Advance Scientific Research
(ISSN – 2750-1396)
VOLUME 05 ISSUE 10 Pages: 155-161
OCLC – 1368736135

contributes to the literature by offering a holistic, theoretically grounded, and practically relevant perspective on securing software supply chains, underscoring the need for multi-layered, proactive, and ecosystem-aware approaches.

KEYWORDS

Software supply chain security, SBOM, open-source vulnerabilities, dependency management, attack trees, vulnerability assessment, LLM vulnerability sourcing

INTRODUCTION

The modern software ecosystem is increasingly characterized by extensive reliance on third-party libraries, packages, and frameworks. Open-source development, in particular, has enabled accelerated innovation, collaboration, and cost efficiency (Haddad & Warner, 2011). However, these benefits are accompanied by heightened risks arising from complex dependency networks, inconsistent update practices, and the lack of comprehensive oversight mechanisms (Cox, 2019). The exponential growth in software interdependencies has transformed supply chain security into a central concern, as exemplified by high-profile vulnerabilities such as Heartbleed (Durumeric et al., 2014) and numerous attacks on package managers like npm and PyPI (Zimmermann et al., 2019; Duan et al., 2020).

A critical challenge lies in the visibility and traceability of dependencies, where nested, transitive relationships obscure potential attack surfaces. Package management systems, while facilitating software distribution, inadvertently amplify the risk of supply chain attacks by automating dependency resolution without rigorous verification or auditing (Cappos et al., 2008). Empirical studies have demonstrated that vulnerabilities can propagate rapidly through dependency chains, often affecting thousands of downstream projects before detection (Ohm et al., 2020). Furthermore, the heterogeneity of programming languages and ecosystems complicates the implementation of standardized security measures, necessitating adaptive and context-specific approaches.

Despite the recognition of these challenges, there exists a notable gap in systematically evaluating the interplay between vulnerability detection mechanisms, dependency management practices, and SBOM implementation. While tools for automated vulnerability scanning and SBOM generation have proliferated, their efficacy in real-world contexts remains variable and under-explored (Dann et al., 2021; Benedetti et al., 2024). Concurrently, emerging approaches leveraging large language models (LLMs) for vulnerability sourcing present novel opportunities and risks, requiring careful empirical and theoretical scrutiny (Ashiwal et al., 2024; Asare et al., 2023).

This research addresses these gaps by adopting a multi-dimensional approach that integrates software engineering, security analysis, and empirical evidence to explore the structural and operational vulnerabilities of software supply chains. By combining dependency analysis, attack tree modeling, and SBOM evaluation, the study aims to provide actionable insights into securing

International Journal of Advance Scientific Research
(ISSN – 2750-1396)
VOLUME 05 ISSUE 10 Pages: 155-161
OCLC – 1368736135

software ecosystems against contemporary threats.

Methodology

The methodological framework of this study is predicated on a comprehensive, qualitative, and descriptive analysis of software supply chain security, leveraging both theoretical constructs and empirical findings. The primary objective is to synthesize extant research on dependency vulnerabilities, attack vectors, and SBOM practices, contextualized within modern open-source and proprietary software ecosystems.

The first methodological step involves a systematic examination of dependency structures within software projects. Drawing on prior studies of package management systems, including npm, PyPI, and Apache repositories (Zimmermann et al., 2019; Bavota et al., 2013, 2015), the research constructs detailed dependency graphs that model transitive relationships and potential propagation of vulnerabilities. These graphs are annotated with metadata on package popularity, update frequency, and known security incidents, enabling the identification of critical nodes and high-risk dependencies.

Next, the study applies attack tree theory (Schneier, 1999; Mauw & Oostdijk, 2006) to conceptualize potential exploitation pathways within the software supply chain. Each node in the dependency graph is evaluated against a set of attack vectors, including code injection, tampering with package metadata, malicious updates, and compromised build processes (Ohm et al., 2020; Duan et al., 2020). By integrating attack tree modeling with empirical vulnerability reports, the methodology provides a structured framework for threat assessment and risk prioritization.

The research further investigates the role of SBOMs in mitigating supply chain risks. Drawing on recent analyses of SBOM generation tools and their impact on vulnerability assessment (Benedetti et al., 2024; Balliu et al., 2023), the study evaluates the coverage, accuracy, and practical utility of SBOMs in diverse software ecosystems. This component includes comparative analysis of SBOM formats, automated generation pipelines, and the integration of SBOM data into continuous security monitoring frameworks.

Additionally, the study incorporates insights from LLM-based vulnerability sourcing and automated code analysis (Ashiwal et al., 2024; Asare et al., 2023). These approaches are examined in terms of their ability to identify previously undetected vulnerabilities, supplement traditional scanning tools, and provide actionable intelligence for developers and security practitioners. The methodology emphasizes the contextual limitations of AI-assisted approaches, including false positive rates, model biases, and reliance on unstructured data sources.

Finally, the study synthesizes the findings into a comprehensive risk evaluation framework that links dependency structures, attack pathways, and SBOM utility. The framework enables descriptive, text-based analysis of vulnerabilities, offering both operational and strategic recommendations without reliance on visual aids or mathematical modeling.

Results

International Journal of Advance Scientific Research
(ISSN – 2750-1396)
VOLUME 05 ISSUE 10 Pages: 155-161
OCLC – 1368736135

ISSN-2750-1396

The analysis of dependency networks across multiple software ecosystems reveals a pronounced concentration of risk among a relatively small set of highly interconnected packages. In npm, for example, a significant proportion of vulnerabilities propagate through a limited number of widely used libraries, amplifying the potential impact of targeted attacks (Zimmermann et al., 2019). Apache projects demonstrate a similar pattern, where transitive dependencies account for a majority of known security incidents (Bavota et al., 2013). The descriptive evaluation highlights the complex interdependencies that create cascading vulnerability effects, emphasizing the necessity for rigorous monitoring and proactive mitigation.

Attack tree modeling provides a granular understanding of potential exploitation pathways. Malicious actors can exploit the inherent trust in package managers, inserting compromised packages or malicious updates that propagate silently through dependency chains (Ohm et al., 2020; Duan et al., 2020). Specific attack vectors include typosquatting, metadata manipulation, and introduction of backdoors in minor version updates. The analysis underscores the critical importance of multi-layered security controls, including verification of package provenance, automated anomaly detection, and continuous monitoring of dependency evolution.

Evaluation of SBOM generation tools demonstrates measurable improvements in visibility and traceability of dependencies. Tools provide comprehensive inventories of package relationships, version histories, and known vulnerabilities, facilitating rapid identification of high-risk nodes (Benedetti et al., 2024). However, descriptive analysis reveals significant limitations, including incomplete coverage of nested dependencies, delays in vulnerability updates, and inconsistencies in SBOM formats (Balliu et al., 2023). These challenges reduce the operational effectiveness of SBOMs as standalone security measures, highlighting the necessity for integration with broader monitoring and auditing frameworks.

The incorporation of LLM-based vulnerability sourcing offers complementary advantages. Large language models can process unstructured data sources, including code repositories, commit histories, and vulnerability disclosures, to identify potential weaknesses not captured by traditional scanners (Ashiwal et al., 2024). Empirical comparisons suggest that while LLMs can detect subtle patterns indicative of security risks, their performance varies by language, codebase size, and dataset quality (Asare et al., 2023). The descriptive assessment emphasizes that LLMs should be deployed as augmentative tools rather than replacements for established security practices.

Discussion

The findings indicate that software supply chain security is a multi-dimensional problem that cannot be effectively addressed through isolated interventions. Dependency analysis reveals structural vulnerabilities inherent in ecosystem design, suggesting that high-risk packages should be subject to enhanced scrutiny and automated monitoring. Attack tree modeling reinforces the need for a proactive, risk-based approach, prioritizing intervention at critical nodes where exploitation could result in widespread impact.

International Journal of Advance Scientific Research
(ISSN – 2750-1396)
VOLUME 05 ISSUE 10 Pages: 155-161
OCLC – 1368736135

ISSN-2750-1396

The interplay between these structural insights and operational tools, such as SBOMs and automated vulnerability scanners, defines the contemporary security landscape.

SBOM implementation, while beneficial, is constrained by limitations in accuracy, completeness, and timeliness. Practical deployment requires standardization across ecosystems, continuous integration with build and deployment pipelines, and mechanisms to address the dynamic evolution of dependencies (Benedetti et al., 2024; Balliu et al., 2023). The descriptive evaluation highlights that SBOMs alone do not prevent attacks but provide essential situational awareness that informs risk mitigation strategies.

LLM-based vulnerability sourcing introduces both opportunities and challenges. The ability to extract actionable intelligence from unstructured data enhances early detection capabilities and supports informed decision-making (Ashiwal et al., 2024; Asare et al., 2023). However, reliance on AI models introduces new uncertainties, including the propagation of false positives, biases, and potential adversarial exploitation of model outputs. Integrating LLMs within a comprehensive risk management framework mitigates these limitations and enables synergistic interaction with traditional scanning and auditing tools.

Limitations of the current research include the reliance on descriptive rather than quantitative modeling, the inherent variability of software ecosystems, and the evolving nature of supply chain threats. Future research should focus on longitudinal studies of dependency evolution, empirical validation of LLM-assisted vulnerability detection, and the development of standardized SBOM practices across diverse programming languages and ecosystems. Additionally, sociotechnical factors, including developer practices, governance structures, and community norms, warrant closer investigation to understand their influence on supply chain security outcomes (Balayn et al., 2024).

Conclusion

This research underscores the critical importance of securing modern software supply chains, highlighting the interplay between dependency management, vulnerability assessment, and SBOM implementation. The findings demonstrate that software ecosystems are inherently complex and prone to cascading risks, necessitating multi-layered, proactive approaches. SBOMs, automated vulnerability scanners, and LLM-based tools provide valuable mechanisms for risk mitigation, but their efficacy depends on comprehensive integration, standardization, and continuous monitoring. Attack tree modeling and dependency analysis offer theoretical and practical frameworks for identifying high-risk nodes and prioritizing interventions.

Ultimately, securing software supply chains requires a holistic understanding of structural vulnerabilities, operational practices, and emerging technological tools. Multi-dimensional strategies that combine technical, procedural, and sociotechnical measures provide the most effective defense against contemporary threats. By advancing the theoretical and empirical understanding of software supply chain security, this study contributes actionable insights for developers, organizations, and policymakers seeking to safeguard critical software infrastructure.

International Journal of Advance Scientific Research
(ISSN – 2750-1396)
VOLUME 05 ISSUE 10 Pages: 155-161
OCLC – 1368736135

ISSN-2750-1396

References

1. Haddad, I., & Warner, B. (2011). Understanding the open source development model. Linux Journal.
2. Cox, R. (2019). Our software dependency problem. Unpublished essay, available online in January: https://research.swtch.com/deps.pdf
3. Cappos, J., Samuel, J., Baker, S., & Hartman, J. (2008). Package management security.
4. Duan, R., Alrawi, O., Kasturi, R. P., Elder, R., Saltaformaggio, B., & Lee, W. (2020). Towards measuring supply chain attacks on package managers for interpreted languages. arXiv preprint arXiv:2002.01139
5. Zimmermann, M., Staicu, C.-A., Tenny, C., & Pradel, M. (2019). Small world with high risks: A study of security threats in the npm ecosystem. In 28th USENIX Security Symposium (USENIX Security 19), Santa Clara, CA, pp. 995–1010. USENIX Association.
6. Dann, A., Plate, H., Hermann, B., Ponta, S. E., & Bodden, E. (2021). Identifying challenges for OSS vulnerability scanners-a study & test suite. IEEE Transactions on Software Engineering.
7. Durumeric, Z., Li, F., Kasten, J., Amann, J., Beekman, J., Payer, M., Weaver, N., Adrian, D., Paxson, V., Bailey, M., & Halderman, J. A. (2014). The matter of heartbleed. In Proceedings of the 2014 Conference on Internet Measurement Conference, IMC '14, New York, NY, USA, p. 475–488. Association for Computing Machinery.
8. Ohm, M., Plate, H., Sykosch, A., & Meier, M. (2020). Backstabber's knife collection: A review of open source software supply chain attacks.
9. Schneier, B. (1999). Attack trees. Dr. Dobb's journal, 24(12), 21–29.
10. Mauw, S., & Oostdijk, M. (2006). Foundations of attack trees. 3935, 186–198.
11. Shukla, O. Software Supply Chain Security: Designing a Secure Solution with SBOM for Modern Software EcoSystems.
12. Cloud Security Alliance. (2024). Global Security Database (GSD). Retrieved from https://github.com/cloudsecurityalliance/gsd-database
13. Asare, O., Nagappan, M., & Asokan, N. (2023). Is GitHub's copilot as bad as humans at introducing vulnerabilities in code? Empirical Software Engineering, 28(6), 129.
14. Ashiwal, V., Finster, S., & Dawoud, A. (2024). LLM-based vulnerability sourcing from unstructured data. In 2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 634–641. IEEE.
15. Balayn, A., Corti, L., Rancourt, F., Casati, F., & Gadiraju, U. (2024). Understanding stakeholders' perceptions and needs across the LLM supply chain. arXiv preprint arXiv:2405.16311. Retrieved from https://arxiv.org/abs/2405.16311
16. Balliu, M., Baudry, B., Bobadilla, S., Ekstedt, M., Monperrus, M., Ron, J., Sharma, A., Skoglund, G., Soto-Valero, C., & Wittlinger, M. (2023). Challenges of producing software bill of materials for Java. IEEE Security & Privacy, 21(6), 12–23.
17. Barr-Smith, F., Blazytko, T., Baker, R., & Martinovic, I. (2022). Exorcist: Automated differential analysis to detect compromises in closed-source software supply chains. In Proceedings of the 2022 ACM Workshop on

International Journal of Advance Scientific Research
(ISSN – 2750-1396)
VOLUME 05 ISSUE 10 Pages: 155-161
OCLC – 1368736135

Software Supply Chain Offensive Research and Ecosystem Defenses, 51–61.

18. Bavota, G., Canfora, G., Di Penta, M., Oliveto, R., & Panichella, S. (2013). The evolution of project inter-dependencies in a software ecosystem: The case of Apache. In 2013 IEEE International Conference on Software Maintenance, 280–289. IEEE.
19. Bavota, G., Canfora, G., Di Penta, M., Oliveto, R., & Panichella, S. (2015). How the Apache community upgrades dependencies: An evolutionary study. Empirical Software Engineering, 20, 1275–1317.
20. Benedetti, G., Cofano, S., Brighente, A., & Conti, M. (2024). The impact of SBOM generators on vulnerability assessment in Python: A comparison and a novel approach. arXiv:2409.06390. Retrieved from https://arxiv.org/abs/2409.06390