COLOMBO SCIENTIFIC PUBLISHING

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE & INFORMATION SYSTEM
E-ISSN: 2536-7919
P-ISSN: 2536-7900

PAGE NO: 71-76

Securing Software Supply Chains: Threat Modeling, Dependency Vulnerabilities, and Intelligence-Driven Risk Assessment

Aarav Mitchell
Global Institute of Cybersecurity Research, United Kingdom

ARTICLE INFO

Article history:
Submission: November 01, 2025
Accepted: November 17, 2025
Published: November 30, 2025
VOLUME: Vol.10 Issue 11 2025

Keywords:
Software Supply Chain Security, Open-Source Dependencies, Threat Modeling, Vulnerability Assessment, Threat Intelligence, Socio-Technical Frameworks, SBOM

ABSTRACT

The rapid evolution of software development paradigms, combined with integral reliance on open-source components, has precipitated an intricate landscape of supply chain vulnerabilities that challenge traditional security mechanisms. In modern ecosystems, developers routinely integrate third-party libraries and tools, accelerating innovation at the cost of increased exposure to security risks. This research article provides a comprehensive examination of the multifaceted problem of software supply chain security, with a particular focus on threat intelligence mining, dependency vulnerability measurement, socio-technical threat modeling, dynamic compartmentalization techniques, and evaluative frameworks for automated vulnerability reporting tools. Through a synthesis of extant literature, this study articulates the theoretical underpinnings of software supply chain risk, delineates methodological advances in vulnerability impact assessment, and critiques current tools and frameworks designed to detect and mitigate such vulnerabilities. We further elucidate the importance of structured metadata such as Software Bills of Materials (SBOMs) (Shukla, O.) and advocate for the integration of advanced machine intelligence to augment existing security practices. Findings highlight significant gaps in current security measures, including inconsistent reporting by composition analysis tools, challenges in accurate threat prioritization, and socio-technical barriers that impede effective risk mitigation strategies. The study concludes by offering a detailed discussion on future research directions, emphasizing the critical role of collaborative, intelligence-driven approaches to secure global software supply chains.

INTRODUCTION

Software development has transformed from isolated, monolithic practices to highly interconnected ecosystems powered by open-source software (OSS), rapid deployment pipelines, and distributed development communities. This transformation has provided marked improvements in productivity, cost reduction, and collaborative innovation. However, it has simultaneously introduced profound complexities in understanding and defending against security vulnerabilities inherent in supply chain components. These concerns are not trivial: the interdependencies among OSS libraries and modules create expansive attack surfaces that can be exploited by malicious actors, sometimes with devastating consequences. Indeed, empirical evidence shows that vulnerable dependencies, if left unaccounted for, can serve as entry points for systemic compromise (Pashchenko et al., 2018).

The problem of software supply chain security is distinguished by its breadth and depth. It encompasses technical dimensions such as coding flaws and dependency risks, as well as socio-technical dimensions involving human actors, organizational processes, and collaborative incentives (Sabbagh & Kowalski, 2015). The burgeoning popularity of open-source components has amplified the scale of these challenges, as projects often rely on hundreds or even thousands of dependencies, each of which might harbor unreported or undisclosed vulnerabilities. The process of identifying and quantifying which vulnerabilities matter most within the context of a given application remains an open area of research (Pashchenko et al., 2018; Plate, Ponta & Sabetta, 2015).

A further complicating factor in supply chain security is the lack of standardization in documenting software components and their interdependencies. The adoption of structured metadata, particularly Software Bills of Materials (SBOMs), has emerged as a critical enabler for transparency and proactive risk assessment (Shukla, O.). SBOMs allow organizations to map their software dependencies systematically, track known vulnerabilities, and facilitate timely updates, thereby mitigating potential exposure in complex ecosystems.

In this research article, we explore the critical issues associated with securing software supply chains. We begin by reviewing seminal work on threat intelligence mining from code repositories and bug reports (Neil, Mittal & Joshi, 2018), before delving into quantitative assessments of open-source dependency vulnerabilities (Pashchenko et al., 2018). We then examine innovative approaches for automated application compartmentalization (Vasilakis et al., 2018), a technical mitigation strategy, and critically evaluate the performance of software composition analysis tools (Imtiaz, Thorn & Williams, 2021). Additionally, socio-technical frameworks for threat modeling (Sabbagh & Kowalski, 2015) and empirical studies addressing dependency-based attacks (Pfretzschner & Othmane, 2017) are discussed, providing a holistic lens through which to view software supply chain risk. Finally, the role of SBOMs and standardized supply chain documentation (Shukla, O.) is emphasized as a cornerstone for modern secure software ecosystems.

The urgent need for improved supply chain security cannot be overstated. High-profile breaches resulting from compromised dependencies have underscored the real-world impact of overlooked vulnerabilities. For instance, supply chain attacks on widely used software packages have caused cascading failures across sectors, affecting enterprises, governments, and critical infrastructure. These incidents highlight a systemic deficiency in current risk assessment practices, where visibility into third-party risk is often limited and reactive rather than proactive. By synthesizing insights from the literature and presenting an integrated evaluation of existing approaches, this article aims to inform both academic discourse and practical implementation strategies for strengthening software supply chain defenses.

METHODOLOGY

The methodology employed in this research is a qualitative synthesis grounded in comprehensive analysis of peer-reviewed literature, industry frameworks, and empirical studies related to software supply chain security. Rather than generating new empirical data via experimental or survey approaches, this article critically integrates findings from foundational and contemporary sources to construct a detailed comparative assessment of existing models, tools, and conceptual frameworks. This approach is particularly suitable given the interdisciplinary nature of the topic, which spans software engineering, cybersecurity, socio-technical systems, and threat intelligence.

First, we conducted a systematic literature review focusing on several key domains: threat intelligence mining from development artifacts, measurement and prioritization of open-source dependency vulnerabilities, automated containment mechanisms, socio-technical threat modeling frameworks, and the evaluation of software composition analysis tools. Each domain was selected based on its relevance to understanding the multifaceted nature of supply chain risk. Through thematic coding of studies, we identified dominant constructs, recurring patterns, observed limitations, and emergent theoretical perspectives. Additionally, the literature on SBOMs and structured supply chain metadata (Shukla, O.) was analyzed to assess its role in enhancing transparency and risk management.

Second, critical comparative analysis was undertaken to examine how different research contributions approach core challenges such as risk quantification, attack surface reduction, and prioritization of mitigative actions. This involved dissecting the underlying assumptions, data sources, analytical methods, and reported outcomes of each study, paying particular attention to contextual factors that influence applicability in real-world settings.

Third, this article embraces an interpretive stance, wherein findings are not merely summarized but deeply interrogated. Contrasts between approaches are explicated, theoretical implications are explored, and gaps in current practices are identified. For example, discrepancies in vulnerability reporting accuracy among software composition analysis tools (Imtiaz, Thorn & Williams, 2021) are examined not only for their technical implications but also for their potential impact on organizational risk management processes. Moreover, we evaluate how socio-technical elements such as developer behavior and organizational culture intersect with technical safeguards, informed by frameworks proposed by Sabbagh and Kowalski (2015). The integration of SBOM-based visibility (Shukla, O.) is also considered in assessing how systematic documentation can enhance threat mitigation strategies.

By integrating heterogeneous sources and emphasizing interpretive depth, this methodology allows for a robust exploration of software supply chain security that transcends disciplinary boundaries. Rather than being limited to a narrow technical viewpoint, the research encapsulates the complex interplay between technology, people, and processes—an essential vantage point for addressing contemporary supply chain threats effectively.

RESULTS

Through detailed analysis of the literature, several salient findings emerge that collectively illuminate the complex landscape of software supply chain security. These results are organized around key thematic threads identified in the methodology: threat intelligence mining, open-source dependency vulnerabilities, automated mitigation strategies, accuracy and efficacy of vulnerability reporting tools, socio-technical threat modeling, and structured supply chain documentation. Each of these domains reflects critical dimensions of supply chain risk and yields insights with both theoretical and practical significance.

Threat Intelligence from Development Artifacts

Neil, Mittal, and Joshi (2018) demonstrate that code repository issues and bug reports represent rich, yet underutilized, sources of threat intelligence. Their research reveals that unstructured development artifacts frequently contain early indicators of vulnerabilities or emerging attack patterns that conventional scanning tools may overlook. For example, lengthy discussions on public issue trackers often reveal nuanced information about unstable code behavior, exploitation attempts, or latent defects that have yet to be officially cataloged in vulnerability databases. By mining such artifacts, security teams can gain foresight into potential risks before they manifest in production environments. This finding underscores the importance of integrating threat intelligence pipelines with development workflows to proactively surface actionable insights.

Magnitude and Prioritization of Vulnerable Dependencies

The work of Pashchenko et al. (2018) provides a quantitative lens on the prevalence of vulnerable open-source dependencies. Their study emphasizes that it is not sufficient merely to count vulnerabilities; rather, it is essential to assess which vulnerabilities materially affect the security posture of dependent software. They introduce methodologies to distinguish between critical and peripheral vulnerabilities based on usage context, impact potential, and exposure likelihood. This nuanced prioritization is pivotal for risk mitigation, especially in large projects where resources for remediation are finite. Their findings highlight the need for contextual awareness in vulnerability assessment, pushing beyond simplistic metrics toward more sophisticated evaluative frameworks.

Automated Application Compartmentalization

Vasilakis et al. (2018) present Breakapp, an automated framework designed to compartmentalize application components, thereby reducing the blast radius of potential exploits. The study reveals that flexible isolation of modules can significantly enhance resilience by limiting the extent to which a vulnerability in one component can compromise the entire system. Such architectural approaches are particularly relevant in microservices and modular systems, where fine-grained segmentation can act as an effective barrier against lateral movement by adversaries. This aligns with defense-in-depth principles and reflects the dynamic adaptation of security design to contemporary software structures.

Evaluation of Vulnerability Reporting Tools

Imtiaz, Thorn, and Williams (2021) conduct a comparative study of software composition analysis tools, uncovering considerable variation in vulnerability reporting. Their results indicate that tools often differ in detection capabilities, false positive rates, and coverage of dependency hierarchies. These disparities suggest that reliance on any single tool may produce gaps in visibility or misguided prioritization of risks. Their analysis encourages multi-tool strategies and enhanced calibration of detection thresholds to improve the reliability of reported findings.

Socio-Technical Threat Modeling

The socio-technical framework proposed by Sabbagh and Kowalski (2015) expands threat modeling beyond purely technical vectors to incorporate organizational behavior, interdependencies among stakeholders, and human decision-making processes. This perspective reveals that supply chain security is not solely a technical challenge but also one deeply rooted in social and organizational contexts. For instance, delayed patch application or poor communication between development and security teams can exacerbate exposure to known vulnerabilities, independent of the technical severity of those flaws. This finding reiterates the necessity for integrated governance mechanisms that align incentives across functional domains.

Structured Supply Chain Documentation and SBOMs

The adoption of SBOMs and other structured supply chain documentation (Shukla, O.) emerges as a critical enabler for visibility and proactive defense. SBOMs facilitate comprehensive mapping of dependencies, identification of known vulnerabilities, and alignment with best practice frameworks for supply chain security. By integrating SBOMs with automated vulnerability detection and threat intelligence pipelines, organizations can achieve enhanced situational awareness and more effective risk prioritization.

Collectively, these results paint a comprehensive picture of the current state of software supply chain security. They underscore the importance of leveraging diverse intelligence sources, adopting context-sensitive vulnerability assessment techniques, embracing architectural mitigations, ensuring robust tool ecosystems, acknowledging socio-technical dynamics, and implementing structured supply chain documentation.

DISCUSSION

The results of this comprehensive literature synthesis yield profound insights into the evolving challenge of software supply chain security. They also raise critical questions about the adequacy of current practices and highlight avenues for future research and operational improvement.

Theoretical Implications

From a theoretical standpoint, the findings reinforce the need for interdisciplinary approaches that bridge technical cybersecurity measures with organizational and human factors. The socio-technical perspective offered by Sabbagh and Kowalski (2015) compels us to recognize that vulnerabilities cannot be fully understood through code analysis alone. They are embedded within social processes, communication flows, and institutional structures that influence how risks are identified, communicated, and remediated. The incorporation of structured documentation through SBOMs (Shukla, O.) complements these socio-technical insights by enabling systematic visibility into complex dependency networks.

Moreover, the contextual prioritization of vulnerabilities introduced by Pashchenko et al. (2018) contributes to an emerging theoretical paradigm that emphasizes risk relevance over sheer vulnerability counts. This shift has implications for risk modeling, suggesting a move toward metrics that integrate impact, exposure, and likelihood within a dynamic threat landscape.

Practical Implications

Practically, the observed variability among software composition analysis tools (Imtiaz, Thorn & Williams, 2021) presents a compelling case for ensemble approaches to vulnerability detection. Organizations should consider aggregating outputs from multiple tools, complemented by manual review and threat intelligence inputs mined from development artifacts (Neil, Mittal & Joshi, 2018). Integration with SBOMs (Shukla, O.) allows these insights to be contextualized across the entire software ecosystem, ensuring that critical dependencies are accurately prioritized.

Automated architectural mitigations such as Breakapp (Vasilakis et al., 2018) demonstrate that design principles can materially reduce risk exposure. By compartmentalizing application components, organizations can limit the impact of exploited vulnerabilities, containing threats before they propagate across systems. This operational strategy aligns with zero-trust principles increasingly advocated in enterprise security.

LIMITATIONS

Despite the depth of insights generated, this study has limitations. First, it is inherently constrained by its reliance on secondary literature rather than primary empirical data. While the synthesis draws from reputable studies, the absence of direct experimentation or case studies may limit the applicability of some conclusions to specific organizational contexts. Second, the heterogeneity of methodologies across the analyzed sources presents challenges for direct comparison of findings. Variations in scope, definition of terms, and analytical frameworks require cautious interpretation. Finally, while SBOM adoption is promoted as a best practice (Shukla, O.), the literature indicates variability in its implementation and standardization, suggesting potential barriers to consistent effectiveness.

Future Scope

Future research must prioritize longitudinal, empirical investigations that validate theoretical models against real-world supply chain incidents. This includes the development of standardized benchmarks to evaluate vulnerability reporting tools, threat intelligence mining techniques, architectural mitigation frameworks, and SBOM integration strategies (Shukla, O.). Additionally, the integration of advanced machine intelligence offers promising opportunities to predict emerging vulnerabilities, automate contextual impact assessment, and enhance socio-technical threat modeling capabilities.

Collaborative ecosystems that facilitate real-time sharing of threat intelligence between open-source communities, commercial entities, and regulatory bodies can further strengthen supply chain defenses. The establishment of shared standards for SBOMs and interoperability of security data formats will be pivotal in enabling such cooperation (Shukla, O.).

CONCLUSION

Securing software supply chains represents a complex and multifaceted challenge that demands holistic approaches. The synthesis presented in this article illustrates that vulnerabilities in open-source dependencies, when combined with socio-technical factors, can undermine the integrity of software ecosystems. Threat intelligence mining from development artifacts, contextual prioritization of dependencies, automated compartmentalization, robust vulnerability reporting tools, socio-technical threat modeling, and structured supply chain documentation via SBOMs (Shukla, O.) each contribute valuable perspectives and mechanisms for risk mitigation.

However, to transcend existing limitations, future efforts must integrate these components into cohesive strategies that are adaptive, data-driven, and informed by both technical and human considerations. Only through such comprehensive frameworks can the software industry hope to defend against the escalating sophistication of supply chain threats in an increasingly interconnected digital world.

REFERENCES

1. L. Neil, S. Mittal, and A. Joshi, “Mining threat intelligence about open-source projects and libraries from code repository issues and bug reports,” in 2018 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 7–12, 2018.
2. I. Pashchenko, H. Plate, S. E. Ponta, A. Sabetta, and F. Massacci, “Vulnerable open source dependencies: Counting those that matter,” in Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM ’18, (New York, NY, USA), Association for Computing Machinery, 2018.
3. N. Vasilakis, B. Karel, N. Roessler, N. Dautenhahn, A. DeHon, and J. M. Smith, “Breakapp: Automated, flexible application compartmentalization,” in NDSS, 2018.
4. N. Imtiaz, S. Thorn, and L. Williams, A Comparative Study of Vulnerability Reporting by Software Composition Analysis Tools. New York, NY, USA: Association for Computing Machinery, 2021.
5. H. Assal and S. Chiasson, “Security in the software development lifecycle,” in Fourteenth symposium on usable privacy and security (SOUPS 2018), pp. 281–296, 2018.
6. S. Benthall, “Assessing software supply chain risk using public data,” in 2017 IEEE 28th Annual Software Technology Conference (STC), pp. 1–5, 2017.
7. B. Pfretzschner and L. ben Othmane, “Identification of dependency-based attacks on node.js,” in Proceedings of the 12th International Conference on Availability, Reliability and Security, ARES ’17, (New York, NY, USA), Association for Computing Machinery, 2017.
8. B. A. Sabbagh and S. Kowalski, “A socio-technical framework for threat modeling a software supply chain,” IEEE Security Privacy, vol. 13, no. 4, pp. 30–39, 2015.

9. S. Zhang, X. Zhang, X. Ou, L. Chen, N. Edwards, and J. Jin, "Assessing attack surface with component-based package dependency," in International Conference on Network and System Security, pp. 405–417, Springer, 2015.

10. H. Plate, S. E. Ponta, and A. Sabetta, "Impact assessment for vulnerabilities in open-source software libraries," in 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 411–420, 2015.

11. Shukla, O. Software Supply Chain Security: Designing a Secure Solution with SBOM for Modern Software EcoSystems.

12. Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. ACM Transactions on Intelligent Systems and Technology 15, 3 (2024), 1–45.

13. Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al.