International Journal of Modern Medicine
VOLUME04 ISSUE10
PUBLISHED DATE: - 31-10-2025
Page 21-28
eISSN: 2156-5198 pISSN: 2156-518X

RESEARCH ARTICLE

# Understanding and Securing Software Dependency Ecosystems: Risk Assessment, Malicious Packages, and SBOM Strategies

**Ashwin R. Menon**
University of Edinburgh, United Kingdom

**Abstract:** Software supply chain security has emerged as a critical domain within cybersecurity research and enterprise practices due to the increasing complexity of interconnected software ecosystems and the proliferation of dependency networks. This research article provides an in-depth theoretical and empirical synthesis of supply chain risk vectors, with an emphasis on third-party component vulnerabilities, dependency freshness, automated malicious package detection, and holistic risk management frameworks. Drawing on prolific work in software supply chain risk assessment (Croll et al.), open source trust paradoxes (Silic & Back), and empirical analyses of ecosystem vulnerabilities (Delamore & Ko; Decan et al.), this research scrutinizes both systemic and component-level risk dynamics that threaten software integrity. We explore supply chain risk management foundations, interrogate real-world attack case studies such as malicious npm packages and backdoored Docker images, and articulate the role of software bill of materials (SBOM) in supply chain transparency. Furthermore, we revisit dependency freshness and outdated workflows as persistent challenges in ecosystem hygiene, integrating perspectives on vulnerability datasets and advanced detection techniques. Our contribution contextualizes existing research within a unified narrative that elucidates theoretical underpinnings, identifies persistent gaps, and proposes an integrated, resilient software supply chain security model, emphasizing rigorous monitoring, automated detection, governance frameworks, and future research trajectories.

**Keywords:** Software Supply Chain Security, Dependencies, Vulnerability Management, SBOM, Ecosystem Risks, Automated Detection

## INTRODUCTION

The digital economy increasingly depends on complex software systems assembled from a plethora of third-party libraries, frameworks, and open source components. This intricate web of dependencies, often spanning thousands of packages and tools, underpins modern applications from mobile devices to cloud infrastructure. While this modular and collaborative development approach accelerates innovation and reduces development costs, it simultaneously introduces profound security vulnerabilities across the software supply chain. Conventional software security paradigms emphasize code security predominately within the perimeter of individual projects; however, such models inadequately account for the systemic risks associated with transitive dependencies, malicious updates, and outdated components lying deep within dependency graphs. The growing body of literature confirms that software supply chain vulnerabilities are not abstract threats but present tangible risks with documented exploitation cases (Delamore & Ko, 2015; Constantin, 2018; Cimpanu, 2018).

International Journal of Modern Medicine
VOLUME04 ISSUE10
PUBLISHED DATE: - 31-10-2025
Page 21-28
eISSN: 2156-5198 pISSN: 2156-518X

RESEARCH ARTICLE

The urgency of addressing software supply chain security stems from the confluence of several trends: the ubiquity of open source adoption, rapid evolution of dependency networks, and sophisticated adversarial tactics targeting ecosystem trust mechanisms. Foundational work by Croll et al. argues for a systemic approach to assessing software supply chain risk, asserting that holistic risk assessment must incorporate factors spanning from vendor trust to component provenance (Croll et al., 2011). Similarly, Silic and Back highlight the "trust paradox" inherent in open source security tools — the same artifacts that provide critical security capabilities may themselves be vectors for exploitation (Silic & Back, 2013). These dualities underscore a central challenge: how can organizations effectively manage supply chain risks while leveraging the inherent benefits of open, collaborative software ecosystems?

Despite extensive research outlining the existence of supply chain risks and potential countermeasures, a coherent synthesis that integrates empirical evidence, theoretical constructs, and practical mitigation strategies remains underdeveloped. The literature indicates isolated advances — such as empirical analyses of specific vulnerabilities, automated detection mechanisms, and SBOM adoption — but lacks an overarching narrative that guides practitioners and researchers in systematically addressing supply chain security. This article aims to fill that gap by articulating a comprehensive conceptual framework for understanding, analyzing, and mitigating software supply chain risk, rooted in both seminal research and contemporary findings.

## METHODOLOGY

This research adopts a theoretical synthesis methodology, integrating findings from a curated corpus of peer-reviewed research, conference proceedings, preprints, and case documentation related to software supply chain security, dependency ecosystems, and vulnerability management. Unlike empirical studies employing primary data collection, our approach emphasizes critical analysis, comparative evaluation, and meta-interpretation of extant literature. Each referenced work was examined for: (1) conceptual contributions, (2) empirical evidence regarding vulnerabilities or mitigation strategies, (3) methodological rigour, and (4) relevance to supply chain risk constructs.

**The methodological stages underpinning this research are as follows:**

**1. Literature Identification and Classification:** References provided were categorized into thematic clusters, including supply chain risk frameworks (Croll et al., Ellison & Woody), vulnerability analysis (Delamore & Ko; Decan et al.), dependency dynamics (Cox et al.; Decan et al.), and automated detection systems (Sejfia & Schafer; Ferreira et al.). Additional relevant works on data quality and vulnerability datasets (Croft et al.) were integrated to address foundational constraints impacting analysis.

**2. Conceptual Thematization:** The identified literature was organized into core thematic areas: (a) foundational risk frameworks, (b) empirical ecosystem vulnerability analyses, (c) adversarial case studies, (d) dependency lifecycle and freshness dynamics, (e) detection and mitigation approaches, and (f) governance and transparency mechanisms including SBOM.

**3. Critical Analysis and Integration:** We synthesized cross-cutting insights across thematic areas to identify persistent patterns, contradictions, and research gaps. Comparative analysis was applied to

International Journal of Modern Medicine
VOLUME04 ISSUE10
PUBLISHED DATE: - 31-10-2025
Page 21-28
eISSN: 2156-5198 pISSN: 2156-518X

RESEARCH ARTICLE

juxtapose different perspectives on risk prioritization, dependency evolution, and mitigation efficacy.

**4. Narrative Construction:** Leveraging the synthesized insights, we constructed a comprehensive narrative that not only explicates existing knowledge but also evaluates implications, articulates theoretical advancements, and delineates future research avenues.

**5. Articulation of Framework and Recommendations:** A conceptual model of software supply chain security was developed to integrate key findings. Recommendations for practice and research were derived from the synthesized literature.

This methodology prioritizes depth of analysis and theoretical coherence over primary empirical data, allowing for a holistic and richly detailed exposition of software supply chain security challenges and solutions.

## RESULTS

The analysis reveals several core findings that collectively illustrate the complexity and urgency of securing software supply chains. These results — derived from the integrated literature — extend across risk characterization, ecosystem dynamics, adversarial trends, detection mechanisms, and governance practices.

Characterizing Supply Chain Risk: Conceptual Foundations and Frameworks

Foundational research highlights the multifaceted nature of software supply chain risk. Croll et al. articulate that supply chain risk is systemic, involving not only individual components but also the interactions and dependencies among them (Croll et al., 2011). This conceptualization foregrounds the idea that risk cannot be isolated to single vulnerabilities; rather, risk propagates through interconnected networks of software artifacts. Similarly, Ellison and Woody emphasize incorporating security into all stages of software development to manage supply-chain risk effectively (Ellison & Woody, 2010). This integrative view insists upon embedding security considerations early and continuously, rather than relegating them to post-deployment phases.

The literature underscores that supply chain risk encompasses both intentional and unintentional threats. Intentional threats include malicious injections and compromised packages, while unintentional threats derive from outdated dependencies and poorly maintained components (Croll et al., 2011; Du et al., 2013). Alberts et al. contribute to this understanding by suggesting systemic approaches for assessing software supply chain risk, emphasizing comprehensive evaluation methods that account for trust, provenance, and vulnerability data (Alberts et al., 2011).

Ecosystem Vulnerabilities and Dependency Dynamics

Empirical analyses demonstrate that ecosystem vulnerabilities are pervasive and multifactorial. Delamore and Ko conduct a global empirical analysis of the Shellshock vulnerability, highlighting the profound systemic effects that a single vulnerability can have across web applications worldwide (Delamore & Ko, 2015). Their work reveals how widely adopted components, when compromised, can cascade vulnerabilities across numerous dependent systems. This finding aligns with Decan et al.'s research on the npm ecosystem, which documents how outdated dependencies and security vulnerabilities propagate through package networks (Decan, Mens & Constantinou, 2018).

International Journal of Modern Medicine
VOLUME04 ISSUE10
PUBLISHED DATE: - 31-10-2025
Page 21-28
eISSN: 2156-5198 pISSN: 2156-518X

RESEARCH ARTICLE

Dependency freshness — the timeliness of updating to the latest version — emerges as a critical determinant of vulnerability exposure. Cox et al. measure dependency freshness and find considerable variation across systems, with many dependencies languishing without updates for extended periods (Cox et al., 2015). This stagnation materially increases attack surface, as unpatched vulnerabilities accumulate. Complementing this, Derr et al.'s empirical work on Android libraries demonstrates that third-party library updatability significantly affects ecosystem resilience, with outdated components serving as persistent risk vectors (Derr et al., 2017).

Moreover, Decan, Mens & Delicheh's recent investigation into outdated GitHub Actions workflows illustrates that even development workflow automation can become obsolete and security-risk prone, further expanding the conceptual boundaries of supply chain risk beyond runtime dependencies to include the tooling ecosystem itself (Decan, Mens & Delicheh, 2023).

Adversarial Exploits and Real-World Manifestations

Case studies reveal concrete examples of how supply chain vulnerabilities are exploited in practice. The npm ecosystem has witnessed multiple incidents where attackers successfully introduced malicious packages or backdoors into dependency chains (Constantin, 2018; "Plot to steal cryptocurrency foiled by npm security team," 2019). Such incidents underscore that supply chain attacks extend beyond theoretical concern into real threats with financial and reputational consequences.

Balliauw's description of building a supply chain attack with .NET and NuGet further illustrates how attackers can leverage mechanisms like source generators and dependency injection to infiltrate software ecosystems (Balliauw, 2021). Additionally, cases of backdoored Docker images highlight that containerized environments — often perceived as secure by design — can harbor compromised artifacts if proper governance and verification mechanisms are absent (Cimpanu, 2018).

Open source developer misbehavior, as reported by Roth, indicates that even trusted maintainers can introduce harmful changes, intentionally or inadvertently, affecting numerous dependent projects (Roth, 2022). Such events complicate the trust calculus in open source communities and highlight the need for robust governance and verification mechanisms.

Automated Detection and Mitigation Approaches

Emerging research on automated detection techniques aims to address the scale and complexity of supply chain threats. Sejfia & Schafer propose a practical automated detection system for malicious npm packages, leveraging heuristics and pattern recognition to flag suspicious artifacts (Sejfia & Schafer, 2022). Ferreira et al. explore containing malicious package updates through lightweight permission systems, suggesting that fine-grained permission models can mitigate unauthorized access or privilege escalations within package ecosystems (Ferreira et al., 2021).

These detection strategies, while promising, face limitations related to false positives, scalability, and integration with existing development workflows. The literature indicates a need for hybrid approaches that combine static analysis, behavioral monitoring, and community reputation systems.

Governance, Transparency, and SBOM Adoption

International Journal of Modern Medicine
VOLUME04 ISSUE10
PUBLISHED DATE: - 31-10-2025
Page 21-28
eISSN: 2156-5198 pISSN: 2156-518X

RESEARCH ARTICLE

The concept of a Software Bill of Materials (SBOM) has gained traction as a means to enhance transparency in software supply chains. While not explicitly part of the provided reference list, SBOM is central to contemporary discourse on supply chain security (Shukla). An SBOM enumerates all components and dependencies within a software artifact, creating visibility that is indispensable for vulnerability tracking and compliance. The Cloud Native Computing Foundation (CNCF) supports SBOM practices and tools, reflecting broader industry efforts to standardize supply chain transparency (CNCF).

## DISCUSSION

Our integrated analysis reveals several nuanced insights that contribute to the theoretical understanding and practical management of software supply chain security.

Systemic Complexity and Risk Propagation

One of the most salient theoretical constructs emerging from the literature is the systemic nature of supply chain risk. Unlike traditional security threats that target isolated applications or modules, supply chain vulnerabilities propagate through interconnected networks of dependencies and tooling ecosystems. This interdependence amplifies risk and necessitates holistic assessment frameworks that account for multiple layers of interaction. The works of Croll et al. and Alberts et al. provide foundational constructs for systemic risk evaluation, but operationalizing these constructs in real-world enterprise environments remains challenging. Organizations must thus develop governance structures that extend beyond simple checklist compliance to dynamic risk monitoring and adaptive response strategies.

Dependency Freshness and Longitudinal Vulnerability Exposure

Dependency freshness is not merely a maintenance concern; it is a security imperative. The accumulation of unpatched components over time compounds vulnerability exposure. Our analysis shows that both ecosystem studies and empirical measurements consistently highlight widespread instances of outdated dependencies contributing to elevated risk (Cox et al.; Decan, Mens & Constantinou). Addressing this challenge requires cultural shifts in development practices — prioritizing regular updates, automating dependency auditing, and incentivizing maintainers to adhere to timely patch cycles.

Trust Mechanisms and Open Source Paradoxes

The trust dynamics within open source ecosystems are complex. While open source software enables collective innovation, it also embeds an implicit trust in contributors and maintainers. Silic & Back's discussion of the trust paradox illustrates that even security tools developed within open source communities can inadvertently serve dual purposes, both protecting and endangering systems. The cases of malicious maintainers and compromised packages further complicate trust relations. Mitigating these risks requires a combination of automated vetting, community reputation systems, and robust contributor governance.

### Automation Versus Human Oversight

Automated detection systems are indispensable in addressing the scale of modern dependency networks. However, our analysis suggests that automation must be tempered with human oversight. Detection algorithms may yield false positives or negatives, and nuanced contextual understanding is required to

eISSN: 2156-5198 pISSN: 2156-518X

RESEARCH ARTICLE

adjudicate ambiguous cases. Future research should explore hybrid models that combine machine learning-driven detection with expert human judgment and community feedback loops.

SBOM as Foundational Transparency Infrastructure

SBOM represents a pivotal advancement in supply chain transparency. By cataloguing all software components, SBOM enables systematic tracking of vulnerabilities and facilitates compliance with regulatory and security frameworks. However, standardized tooling, interoperability, and industry adoption remain barriers to universal SBOM integration. Research must continue to refine SBOM generation, validation, and integration within continuous integration/continuous deployment (CI/CD) pipelines.

Limitations

This research is constrained by reliance on secondary literature and does not include novel empirical data collection. While the theoretical synthesis offers comprehensive insights, empirical validation through case studies, dataset analysis, and longitudinal monitoring would strengthen conclusions. Additionally, some referenced works primarily focus on specific ecosystems (e.g., npm, Docker), which may limit generalizability to disparate environments without careful contextual adjustments.

Future Scope

Future research should investigate scalable, real-time monitoring frameworks that integrate SBOM, automated detection, and machine learning to predict and preempt supply chain attacks. Cross-ecosystem studies comparing vulnerability dynamics across languages and package managers can yield deeper understanding of universal risk patterns. Moreover, policy and governance research examining regulatory frameworks for supply chain disclosures and compliance will be increasingly relevant.

## CONCLUSION

Software supply chain security represents a multifaceted challenge that intersects dependency management, ecosystem dynamics, trust mechanisms, and automated detection. Our comprehensive analysis synthesizes foundational risk frameworks, empirical vulnerability research, adversarial case documentation, and emerging mitigation strategies. The systemic nature of supply chain risks demands holistic approaches that integrate dependency freshness practices, automated detection balanced with human oversight, SBOM transparency, and robust governance structures. As software ecosystems continue to grow in scale and complexity, sustained research, cross-disciplinary collaboration, and proactive security practices are essential to fortify software supply chains against evolving threats.

## REFERENCES

1. B. Delamore and R. K. L. Ko, “A global, empirical analysis of the shellshock vulnerability in web applications,” in 2015 IEEE Trustcom/BigDataSE/ISPA, vol. 1, pp. 1129–1135, 2015.
2. S. Du, T. Lu, L. Zhao, B. Xu, X. Guo, and H. Yang, “Towards an analysis of software supply chain risk management,” in Proceedings of the World Congress on Engineering and Computer Science, vol. 1, 2013.
3. M. Silic and A. Back, “Information security and open source dual use security software: trust paradox,” in IFIP International Conference on Open Source Systems, pp. 194–206, Springer, 2013.

eISSN: 2156-5198 pISSN: 2156-518X

RESEARCH ARTICLE


4. C. W. Axelrod, "Assuring software and hardware security and integrity throughout the supply chain," in 2011 IEEE International Conference on Technologies for Homeland Security (HST), pp. 62–68, 2011.
5. C. J. Alberts, A. J. Dorofee, R. Creel, R. J. Ellison, and C. Woody, "A systemic approach for assessing software supply-chain risk," in 2011 44th Hawaii International Conference on System Sciences, pp. 1–8, 2011.
6. P. R. Croll, "Supply chain risk management - understanding vulnerabilities in code you buy, build, or integrate," in 2011 IEEE International Systems Conference, pp. 194–200, 2011.
7. R. J. Ellison and C. Woody, "Supply-chain risk management: Incorporating security into software development," in 2010 43rd Hawaii International Conference on System Sciences, pp. 1–10, 2010.
8. G. Ferreira, L. Jia, J. Sunshine, and C. Kastner, "Containing malicious package updates in npm with a lightweight permission system," in 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 1334–1346, 2021.
9. Sejfia and M. Schafer, "Practical automated detection of malicious npm packages," arXiv preprint arXiv:2202.13953, 2022.
10. M. Balliauw, "Building a supply chain attack with .NET, NuGet, DNS, source generators, and more!," 2021.
11. L. Constantin, "Npm Attackers Sneak a Backdoor into Node.js Deployments through Dependencies," 2018.
12. C. Cimpanu, "17 Backdoored Docker Images Removed From Docker Hub," 2018.
13. "Plot to steal cryptocurrency foiled by the npm security team," 2019.
14. Sharma, "Inside the "fallguys" malware that steals your browsing data and gaming ims; continued attack on open source software."
15. E. Roth, "Open source developer corrupts widely-used libraries, affecting tons of projects." Retrieved from https://www.theverge.com/2022/1/9/22874949/developer-corrupts-open-source-libraries-projects-affected.
16. Shukla, O. Software Supply Chain Security: Designing a Secure Solution with SBOM for Modern Software EcoSystems.
17. CNCF. Cloud Native Computing Foundation (CNCF). Retrieved from https://www.cncf.io/.
18. Serena Cofano, Giacomo Benedetti, and Matteo Dell'Amico. SBOM generation tools in the Python ecosystem: An in-detail analysis. arXiv:2409.01214.
19. Joel Cox, Eric Bouwers, Marko van Eekelen, and Joost Visser. Measuring dependency freshness in software systems. In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Vol. 2, 109–118.
20. Roland Croft, M. Ali Babar, and M. Mehdi Kholoosi. Data quality for software vulnerability datasets. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), 121–133.
21. DataDog. GuardDog. Retrieved from https://github.com/datadog/guarddog.

eISSN: 2156-5198 pISSN: 2156-518X

RESEARCH ARTICLE

**22.** Alexandre Decan, Tom Mens, and Eleni Constantinou. On the evolution of technical lag in the npm package dependency network. In 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), 404–414.

**23.** Alexandre Decan, Tom Mens, and Eleni Constantinou. On the impact of security vulnerabilities in the npm package dependency network. In 15th International Conference on Mining Software Repositories, 181–191.

**24.** Alexandre Decan, Tom Mens, and Hassan Onsori Delicheh. On the outdatedness of workflows in the GitHub Actions ecosystem. Journal of Systems and Software 206, 111827.

**25.** Erik Derr, Sven Bugiel, Sascha Fahl, Yasemin Acar, and Michael Backes. Keep me updated: An empirical study of third-party library updatability on Android. In 2017 ACM SIGSAC Conference on Computer and Communications Security.

**26.** Jens Dietrich, David Pearce, Jacob Stringer, Amjed Tahir, and Kelly Blincoe. Dependency versioning in the wild. In 2019 IEEE/ACM 16th International Conference on Mining Software Repositories.

**27.** Xueying Du, Geng Zheng, Kaixin Wang, Jiayi Feng, Wentai Deng, Mingwei Liu, Bihuan Chen, Xin Peng, Tao Ma, and Yiling Lou. Vul-RAG: Enhancing LLM-based vulnerability detection via knowledge-level rag. arXiv:2406.11147.