Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Exploit-Aware Dependency Risk Modeling in Modern Software Supply Chains

Vinay Vardhan. T
Jain Deemed to Be University
Banglore, india

Thatte Pavan Kumar
Jain Deemed to Be University
Banglore, india

Subhradeep De
Jain Deemed to Be University
Banglore, india

Sasmitha N. K
Jain Deemed to Be University Banglore, india

Dr. Prabhakaran
Jain Deemed to Be University Banglore, india

Abstract - Software supply chain attacks have emerged as a persistent and escalating threat category within modern software ecosystems, driven by increasing reliance on open-source components and deeply nested transitive dependency structures. Industry analyses consistently report year-over-year growth in malicious package publication and third-party breach involvement, reflecting a shift toward dependency-centric exploitation strategies. High-impact incidents such as the SolarWinds compromise and the Log4Shell vulnerability demonstrate how compromise within widely adopted upstream components can propagate systemic risk across thousands of downstream systems. Despite advances in Software Composition Analysis (SCA) tooling, prevailing remediation practices remain largely severity-centric, prioritizing vulnerabilities primarily through static CVSS- based metrics. Prior research, however, indicates that severity alone does not consistently correlate with

real-world exploitation likelihood or structural exposure within dependency graphs. This study examines contemporary supply chain attack trends through industry trend synthesis and selected case evaluation, identifying structural and exploit-intelligence gaps in conventional dependency risk assessment approaches. Building upon these observations, we outline an analytical dependency risk modeling perspective that integrates version-filtered vulnerability data, exploit-awareness signals, and structural graph context. A comparative case analysis demonstrates how contextual weighting can materially influence remediation prioritization relative to severity-only aggregation. The objective is to provide an analytical examination of dependency risk assessment practices rather than to propose a finalized defensive system.

Keywords **- software supply chain, dependency risk, vulnerability prioritization, exploit intelligence, structural modeling, CVSS, EPSS**

I. INTRODUCTION

The contemporary software development paradigm is characterized by extensive reliance on open- source libraries and third-party components distributed through public package ecosystems. Modern applications frequently incorporate large networks of direct and transitive dependencies managed via registries such as npm, Maven Central, and PyPI. While this modular approach accelerates development velocity and promotes ecosystem reuse, it also introduces layered trust dependencies that extend beyond organizational boundaries. As a result, compromise or vulnerability within a single upstream component may propagate across thousands of downstream systems, amplifying systemic exposure in ways that are often opaque to application maintainers.

Over the past decade, software supply chain attacks have transitioned from isolated exploitation events to a sustained and strategically targeted threat class. The SolarWinds Orion incident demonstrated how manipulation of a trusted software distribution channel could facilitate widespread infiltration across governmental and enterprise networks [4]– [5]. Subsequent events such as Log4Shell (CVE- 2021-44228) and Spring4Shell (CVE-2022-22965)
illustrated how vulnerabilities in widely deployed libraries can trigger global remediation efforts, long- term exposure windows, and cascading operational impact [6]. More recently, ecosystem abuses including malicious package injection, dependency confusion, and upstream build-system compromise have reinforced concerns regarding structural fragility within transitive dependency graphs [7]– [8].

Industry analyses consistently indicate measurable growth in supply chain–oriented threat activity. The Sonatype *State of the Software Supply Chain* reports document sustained year-over-year increases in malicious open-source components and dependency-based exploitation patterns [1]. ReversingLabs similarly reports escalation in software supply chain abuse, highlighting adversarial focus on highly adopted packages within large ecosystems [2]. Verizon's Data Breach Investigations Report further identifies increasing third-party software involvement in breach patterns, underscoring the operational significance of dependency-related risk [3]. When aggregated across reporting sources, these analyses suggest a persistent upward trajectory in dependency-centric threat activity.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

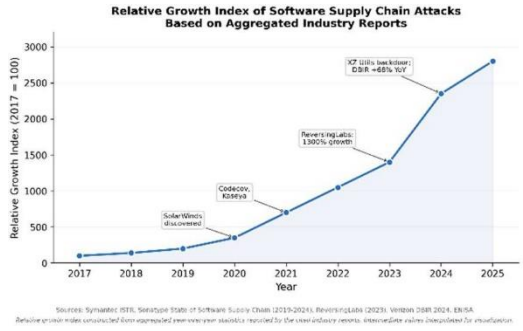

Fig. 1. Relative growth index of software supply chain attacks.

This figure presents a normalized growth index constructed from aggregated year-over-year statistics reported in [1]–[3]. Intermediate values are interpolated for visualization.

Despite the growing visibility of supply chain incidents, prevailing defensive mechanisms remain predominantly vulnerability-centric. Software Composition Analysis (SCA) tools typically enumerate known vulnerabilities present within dependency manifests and prioritize remediation according to static severity metrics such as the Common Vulnerability Scoring System (CVSS). While CVSS provides a standardized severity framework, empirical research suggests that severity alone does not consistently correlate with exploitation likelihood or operational impact [8]– [9]. In practice, high-severity vulnerabilities may remain unexploited, whereas moderate-severity issues can become actively weaponized when strategically advantageous.

Moreover, traditional vulnerability scoring approaches often evaluate components in isolation, without explicit consideration of structural positioning within dependency graphs. Research examining ecosystem "weak links" demonstrates that a relatively small subset of highly central components account for disproportionate downstream reach [9]. Vulnerabilities affecting such components may therefore carry systemic implications exceeding those implied by severity metrics alone. Conversely, deep transitive dependencies with limited reach may present lower practical exposure even when severity appears elevated.

This structural asymmetry suggests that dependency risk assessment requires contextual reasoning beyond static severity enumeration. In addition to graph positioning, exploit-awareness signals— including confirmed in-the-wild exploitation status and probabilistic exploit prediction metrics— provide dynamic indicators of attacker behavior. However, integration of exploit intelligence into dependency-centric evaluation pipelines remains inconsistent across existing SCA implementations [8]. Consequently, a gap persists between ecosystem-level structural insights, vulnerability intelligence research, and practical remediation prioritization mechanisms.

This study examines contemporary software supply chain attack trends and structural risk characteristics through industry synthesis and analytical modeling. The primary objective is not to propose a production-ready defensive platform, but rather to explore how dependency risk evaluation may be refined through integration of version-filtered vulnerability data, exploit-awareness signals, and structural graph context. By examining both macro- level trend indicators and component-level case analysis, the work seeks to contribute an analytical perspective on dependency risk prioritization within modern open-source ecosystems.

The remainder of this paper is organized as follows. Section III reviews related literature on ecosystem risk modeling, malicious package growth, and vulnerability prioritization limitations. Section IV describes the analytical methodology, including dependency graph construction, intelligence enrichment, and contextual risk aggregation. Section
V details the data sources utilized, including vulnerability databases and exploit intelligence feeds. Section VI presents a comparative case analysis examining how exploit-aware contextualization influences remediation prioritization relative to severity-only approaches.

Section VII concludes with discussion and future research directions.

II. LITERATURE REVIEW

The increasing prevalence of software supply chain attacks has prompted extensive academic and industry investigation into the structural and operational characteristics of open-source dependency ecosystems. Existing literature broadly converges on three recurring themes: ecosystem- level structural concentration, malicious package abuse mechanisms, and limitations of vulnerability- centric prioritization models. Together, these strands of research establish the analytical foundation for examining dependency risk beyond static severity metrics.

Empirical studies of large-scale package ecosystems demonstrate that open-source dependency graphs exhibit pronounced structural asymmetry. A relatively small subset of highly central packages serves as hubs upon which substantial portions of the ecosystem depend [9]. This concentration effect implies that compromise or vulnerability within such components may propagate across extensive downstream systems, amplifying systemic exposure. Analyses of npm and similar registries further highlight the depth and breadth of transitive dependencies, illustrating how even indirect components can become operationally significant through cumulative graph reach. These findings underscore the importance of structural context in evaluating dependency risk.

Industry reporting complements academic structural analyses by documenting sustained growth in malicious package publication and dependency- based exploitation campaigns. The Sonatype *State of the Software Supply Chain* reports consistently identify year-over-year increases in malicious open- source components and dependency abuse patterns [2]. ReversingLabs similarly documents escalation in supply chain–oriented attack techniques, including typosquatting, dependency confusion, and targeted compromise of high-impact libraries [2]. Verizon's Data Breach Investigations Report further notes increasing third-party software involvement in breach activity, reinforcing the operational relevance of supply chain exposure [3]. When considered collectively, these sources suggest that adversaries are increasingly leveraging ecosystem scale and automated dependency resolution mechanisms to maximize impact.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

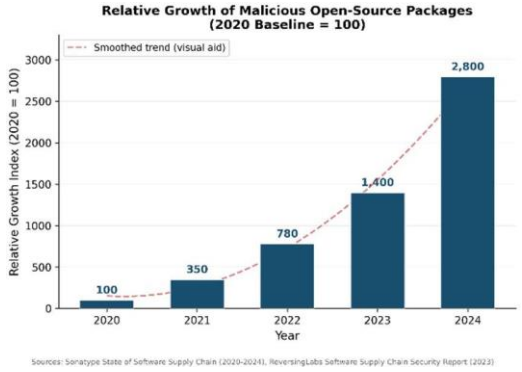

Fig. 2. Growth of malicious open-source packages. *This figure presents a normalized growth index derived from aggregated reporting in [1] and [2], illustrating sustained increases in malicious package publication relative to a defined baseline year.*

Beyond aggregate growth trends, case analyses of high-profile incidents provide insight into attack vectors and propagation dynamics. The SolarWinds compromise demonstrated how tampering within upstream build and distribution channels can enable widespread infiltration across downstream customers [4]–[5]. The Log4Shell vulnerability further illustrated how exploitation of a widely adopted library can produce global remediation efforts and prolonged systemic exposure [6]. Subsequent analyses emphasize that impact magnitude is influenced not only by vulnerability severity but also by dependency centrality and ecosystem reach [8]. These incidents reinforce the structural amplification phenomenon identified in empirical ecosystem studies.

The literature also critically evaluates vulnerability scoring methodologies. The Common Vulnerability Scoring System (CVSS) remains the dominant standardized framework for assessing vulnerability severity; however, multiple studies question its adequacy as a standalone prioritization mechanism [8]–[9]. Empirical observations indicate that CVSS base scores do not consistently correlate with exploitation likelihood in the wild. High-severity vulnerabilities may remain unexploited, whereas lower-severity issues may be rapidly weaponized when strategically advantageous. This disconnect has motivated the development of exploit prediction models and curated exploit intelligence catalogs that incorporate real-world attacker behavior into risk assessment pipelines.

Despite advances in exploit-awareness research, integration of structural graph properties and dynamic intelligence signals into practical dependency risk evaluation remains limited. Many SCA implementations emphasize version-aware vulnerability detection and license compliance but provide limited contextual reasoning regarding transitive depth, centrality, or systemic blast radius [1]–[2]. Consequently, a gap persists between ecosystem-level structural insights and operational remediation prioritization practices.

Taken together, the literature establishes that supply chain attacks are increasing in frequency and sophistication [1]–[3], that structural concentration within dependency graphs amplifies systemic exposure [9], and that reliance on static severity metrics may inadequately capture real-world exploitation dynamics [8]–[9]. These observations motivate a methodological examination of dependency risk that integrates structural modeling and exploit-awareness signals within a transparent analytical framework, as developed in the subsequent section.

III. METHODOLOGY

This study adopts an analytical modeling approach to examine dependency risk within open-source ecosystems by integrating vulnerability intelligence, exploit-awareness signals, and structural graph context. The objective is not to replace standardized vulnerability scoring systems, but to evaluate how contextual enrichment may influence remediation prioritization within transitive dependency structures. The methodology therefore emphasizes transparent aggregation of publicly available intelligence signals while maintaining interpretability of individual contributing factors.

The analytical process begins with dependency extraction from project manifests and associated lock files. Direct and transitive dependencies are modeled as nodes within a directed acyclic graph where edges represent dependency relationships. This structural representation enables computation of graph properties including transitive depth, reachability, and connectivity. Modeling dependencies as graph entities reflects empirical findings that ecosystem exposure is often concentrated among highly central components [9]. By explicitly constructing this graph, the methodology provides structural context for evaluating how vulnerabilities may propagate through dependency hierarchies.

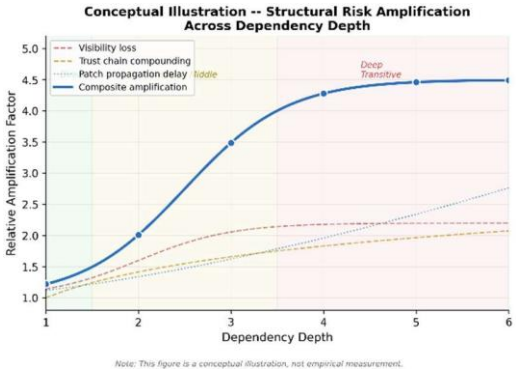

Fig. 3. Conceptual illustration of structural risk amplification across dependency graph depth. *This figure depicts a simplified dependency graph demonstrating how vulnerabilities in structurally central or shallow nodes may influence broader portions of the application surface.*

Following graph construction, vulnerability data applicable to each dependency are retrieved through version-filtered queries against publicly available vulnerability databases. Only vulnerabilities affecting the installed package version are retained, ensuring that patched or non-applicable issues are excluded from analysis. For each relevant vulnerability, standardized severity metrics (CVSS base scores) are recorded alongside associated metadata such as publication date, affected version ranges, and available fixed versions.

To address limitations identified in prior research regarding severity-only prioritization [8], exploit- awareness signals are

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

incorporated into the analytical model. These include curated Known Exploited Vulnerability (KEV) listings indicating confirmed in-the-wild exploitation, as well as probabilistic exploit likelihood estimates derived from the Exploit Prediction Scoring System (EPSS). Incorporating such signals introduces a dynamic behavioral dimension reflecting observed adversarial activity, supplementing static severity metrics. Importantly, exploit indicators are applied as contextual weighting factors rather than as binary overrides, preserving proportional differentiation across vulnerabilities.

Composite vulnerability influence for each dependency is computed through normalized aggregation of severity metrics with exploit-aware adjustments. The aggregation model employs bounded transformations to maintain scores within a defined interval, preventing unbounded inflation while preserving relative differentiation among components. Structural properties are evaluated independently from vulnerability metrics to avoid conflating severity with graph position. This separation enables analysis of how structural context and exploit intelligence jointly influence prioritization outcomes without obscuring underlying data contributions.

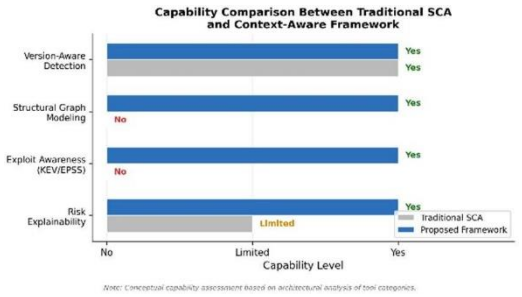

Fig. 4. Conceptual comparison between severity- only and context-enriched dependency risk modeling.
This figure qualitatively contrasts traditional CVSS-centric prioritization approaches with a contextualized model incorporating exploit intelligence and structural graph attributes.

The final stage aggregates indicator contributions into a composite dependency-level score. Individual indicator components remain traceable, allowing identification of primary risk drivers for each dependency. At the project level, aggregated summaries are computed to provide an overall representation of dependency risk distribution. The methodology does not incorporate runtime behavioral telemetry, static code analysis, or binary inspection techniques; its scope is confined to manifest-derived dependency structures combined with publicly accessible vulnerability and exploit intelligence feeds. This constraint ensures reproducibility and aligns the analytical focus specifically with dependency management practices.

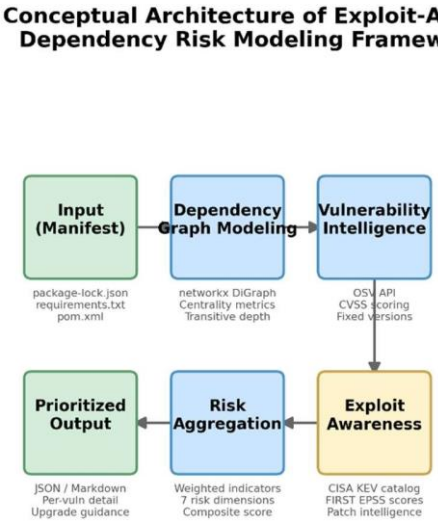

Fig. 5. High-level analytical architecture for context-enriched dependency risk modeling. *This figure presents a conceptual architecture illustrating dependency graph construction, vulnerability ingestion, exploit-awareness enrichment, structural analysis, and composite aggregation.*

By integrating structural modeling and exploit- awareness enrichment within a transparent analytical pipeline, the methodology facilitates examination of how contextual weighting influences remediation prioritization. The subsequent section details the data sources employed and the procedures used to ensure version applicability and reproducibility of intelligence signals.

IV. DATA USED

The analytical evaluation conducted in this study relies exclusively on publicly accessible vulnerability and exploit intelligence sources to ensure transparency and reproducibility. All data were retrieved via documented APIs or official reporting publications, with version applicability explicitly enforced during ingestion.

Vulnerability data were obtained from the National Vulnerability Database (NVD) through its REST API interface. For each dependency evaluated in the case analysis, vulnerabilities were filtered by affected version range to ensure inclusion only when the installed package version satisfied the documented vulnerability constraints. Retrieved metadata included CVE identifiers, CVSS base scores (v3.1 where available), CVSS vector strings, publication dates, modification timestamps, and fixed version information where provided. CVSS values were treated as standardized severity indicators in accordance with established scoring guidelines [8].

Exploit-awareness enrichment was performed using two publicly available intelligence sources. First, the Cybersecurity and Infrastructure Security Agency (CISA) Known Exploited Vulnerabilities (KEV) catalog was used to identify CVEs with confirmed in-the-wild exploitation status. The KEV dataset provides a curated listing of vulnerabilities actively exploited in real-world campaigns and is periodically updated by CISA. Second, probabilistic exploit likelihood estimates were retrieved

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

via the FIRST Exploit Prediction Scoring System (EPSS) API. EPSS values represent data-driven predictions of exploitation probability based on historical attack telemetry and vulnerability characteristics. All exploit-related signals were retrieved with documented retrieval dates to ensure temporal traceability. Because EPSS scores are time-variant, retrieval timestamps are explicitly recorded in the case analysis to preserve interpretive consistency.

Macro-level ecosystem growth indicators presented in Sections III and IV were derived from publicly reported industry analyses. Aggregate year-over- year growth percentages for malicious package publication and supply chain attack trends were synthesized from Sonatype's *State of the Software Supply Chain* reports [1], ReversingLabs threat intelligence analyses [2], and the Verizon Data Breach Investigations Report [3]. Where necessary, reported percentage increases were normalized to a defined baseline year to construct relative growth indices for visualization purposes. These indices are explicitly labeled as normalized representations rather than raw datasets to maintain interpretive transparency.

For case analysis evaluation, selected widely adopted libraries were chosen based on documented historical incidents and ecosystem relevance, including components affected by Log4Shell and Spring4Shell vulnerabilities [6]. For each component, vulnerability inclusion was strictly version-filtered to exclude issues resolved in installed versions. Vulnerabilities marked as withdrawn within the NVD dataset were excluded from computation to avoid inflation of composite indicators. Exploit-awareness enrichment was applied only when corresponding KEV listings or EPSS values were available. In cases where exploit prediction data were temporarily unavailable, the model defaults preserved severity-based weighting without artificial substitution.

All dependency graph modeling was performed using manifest and lock file representations to ensure deterministic extraction of direct and transitive components. Structural properties were computed solely from dependency relationships and did not incorporate runtime telemetry or dynamic execution traces. The analytical scope therefore remains bounded to dependency metadata and publicly accessible intelligence signals, preserving reproducibility and alignment with dependency management workflows.

By grounding the analysis in version-filtered vulnerability data [8], curated exploitation intelligence, and published ecosystem reports [1]– [3], this study maintains traceable data provenance while enabling examination of contextualized dependency risk assessment. The following section presents a comparative evaluation illustrating how integration of exploit-awareness signals influences prioritization outcomes relative to severity-only aggregation approaches.

V. COMPARATIVE ANALYSIS WITH EXISTING RESEARCH

The literature reviewed in Section III identifies two principal limitations in prevailing dependency risk assessment practices: reliance on static severity metrics and limited incorporation of structural or exploit-contextual signals [8]–[9]. To examine the practical implications of these limitations, this section presents a focused comparative analysis contrasting severity-only prioritization with a contextualized, exploit-aware aggregation approach. The objective is not to benchmark against proprietary tools, but to evaluate how integration of exploit intelligence influences remediation ordering within representative dependency sets.

The comparative evaluation considers selected widely deployed libraries with documented historical vulnerabilities, including components associated with the Log4Shell (CVE-2021-44228) and Spring4Shell (CVE-2022-22965) incidents [6]. For each dependency, vulnerability inclusion was strictly constrained to version-applicable CVEs as described in Section V. Two aggregation strategies were then computed. The first reflects a conventional CVSS-based prioritization model in which vulnerability influence is determined solely by normalized severity metrics. The second incorporates exploit-awareness signals, including Known Exploited Vulnerability (KEV) status and probabilistic Exploit Prediction Scoring System (EPSS) values, applied as contextual weighting adjustments.

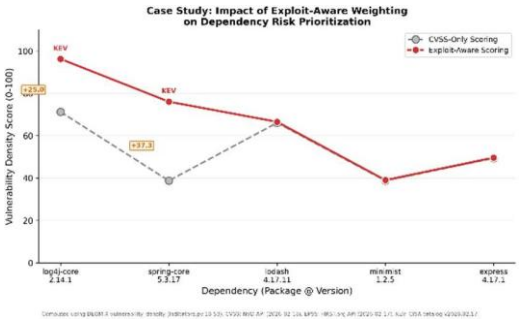

Fig. 6. Comparative dependency risk scores: severity-only vs. exploit-aware aggregation. *This figure illustrates composite vulnerability risk scores under two prioritization strategies. Data sources include NVD (CVSS), CISA KEV catalog, and FIRST EPSS API, retrieved February 2026.*

The results demonstrate that incorporation of exploit-awareness signals produces measurable prioritization shifts for dependencies associated with confirmed in-the-wild exploitation or elevated exploit probability. For example, vulnerabilities listed in the KEV catalog exhibit amplified composite influence under exploit-aware aggregation relative to severity-only scoring. Conversely, vulnerabilities with moderate or high CVSS scores but low EPSS probabilities exhibit comparatively smaller adjustments. These findings are consistent with prior research indicating that CVSS severity does not necessarily correspond to real-world exploitation likelihood [8]–[9].

Importantly, the comparative differences do not invalidate severity metrics; rather, they highlight the contextual dimension absent from static scoring frameworks. In severity-only models, two vulnerabilities with identical CVSS scores receive equivalent prioritization regardless of exploitation activity. Under contextualized aggregation, confirmed exploitation status and probabilistic signals introduce differentiation reflective of adversarial behavior. This distinction aligns with literature

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

emphasizing the importance of behavioral intelligence in vulnerability management [8].

The analysis also reinforces structural considerations discussed in Section IV. Dependencies positioned near the root of the graph or exhibiting broader reach may exert greater systemic influence than deeply nested components, even when severity metrics are comparable. While the present evaluation does not incorporate full-scale ecosystem sampling, it illustrates how structural and exploit-aware signals can jointly influence remediation ordering within representative dependency subsets. Such contextual reasoning addresses the structural-context gap identified in existing SCA implementations [1]–[2].

Several limitations must be acknowledged. The case analysis is intentionally focused and does not constitute a statistically representative survey of the broader ecosystem. Exploit prediction scores such as EPSS are time-variant and probabilistic, potentially shifting across retrieval intervals. Additionally, the analytical framework remains bounded to dependency metadata and publicly accessible intelligence feeds, excluding runtime monitoring and code-level validation techniques. Consequently, the findings should be interpreted as illustrative of prioritization sensitivity rather than as definitive predictive assessments.

Nevertheless, the comparative evaluation demonstrates that contextual enrichment— specifically, integration of exploit-awareness signals—can materially influence remediation prioritization relative to severity-only approaches. These findings support the literature's broader argument that dependency risk assessment benefits from incorporation of dynamic intelligence and structural reasoning alongside standardized severity metrics. The subsequent section concludes the study by summarizing analytical insights and identifying directions for further research.

VI. CONCLUSION AND FUTURE WORK

Software supply chain security has emerged as a central concern within contemporary software engineering due to the pervasive integration of open- source dependencies and deeply nested transitive relationships. As demonstrated through industry reporting and documented incidents, adversaries increasingly exploit structural characteristics of dependency ecosystems to maximize impact [1]–[3], [6]. High-profile events such as SolarWinds and Log4Shell underscore how compromise or vulnerability within upstream components can propagate systemic exposure across downstream environments [4]–[6]. These observations reinforce the need for contextualized approaches to dependency risk evaluation.

This study examined prevailing dependency risk assessment practices through a structured review of ecosystem research and vulnerability prioritization literature. Existing research identifies structural concentration within dependency graphs [9] and highlights limitations of severity-only remediation strategies [8]. In response to these observations, the present work articulated an analytical framework integrating version-filtered vulnerability data, exploit-awareness signals, and structural graph context. Rather than proposing a replacement for standardized severity metrics, the study evaluated how contextual enrichment influences remediation prioritization outcomes.

The comparative case analysis demonstrated that incorporation of exploit-awareness signals, including confirmed exploitation status and probabilistic exploit prediction metrics, can materially alter dependency prioritization relative to severity-only aggregation. These findings align with prior research suggesting that static severity measures do not consistently reflect real-world exploitation dynamics [8]. When considered alongside structural context, contextual enrichment provides a more differentiated representation of dependency risk within transitive ecosystems.

Several limitations constrain the scope of the present analysis. First, the framework relies exclusively on publicly available vulnerability and exploit intelligence feeds and does not incorporate runtime telemetry, behavioral monitoring, or code-level inspection techniques. Second, exploit prediction metrics such as EPSS are probabilistic and time- variant, potentially influencing prioritization outcomes across retrieval intervals. Third, the case evaluation is illustrative rather than statistically representative of the broader ecosystem. Consequently, the findings should be interpreted as analytical sensitivity insights rather than comprehensive predictive validation.

Future research may extend this work along several dimensions. Integration of runtime telemetry or dynamic code analysis could enhance contextual fidelity beyond manifest-derived dependency modeling. Longitudinal ecosystem sampling may provide empirical validation of contextualized prioritization effectiveness across larger datasets. Additionally, formal sensitivity analysis of weighting strategies and cross-ecosystem normalization techniques may strengthen robustness and reproducibility in dependency risk modeling. Further exploration of propagation dynamics within highly centralized ecosystems may also clarify how structural amplification interacts with exploit activity at scale.

Future research may extend this work along several dimensions. Integration of runtime telemetry or dynamic code analysis could enhance contextual fidelity beyond manifest- derived dependency modeling. Longitudinal ecosystem sampling may provide empirical validation of contextualized prioritization effectiveness across larger datasets. Additionally, formal sensitivity analysis of weighting strategies and cross-ecosystem normalization techniques may strengthen robustness and reproducibility in dependency risk modeling. Further exploration of propagation dynamics within highly centralized ecosystems may also clarify how structural amplification interacts with exploit activity at scale.

To support transparency and reproducibility, an open-source implementation of the analytical framework described in this study has been developed and made publicly available. The implementation operationalizes version-filtered vulnerability analysis, structural graph modeling, exploit-awareness integration, simulation capabilities, and explainability reporting within a unified analytical tool. The repository is available at: https://github.com/Fopuu10/DEQM.git. The released

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

codebase is intended as a research artifact to facilitate further experimentation, benchmarking, and methodological extension rather than as a finalized production security system.

In summary, this study contributes an analytical examination of dependency risk assessment practices within modern software supply chains. By synthesizing structural modeling and exploit-awareness signals within a transparent aggregation framework, the work highlights how contextual reasoning can refine remediation prioritization beyond severity- centric approaches. Continued research integrating ecosystem structure, vulnerability intelligence, and operational telemetry will be essential for advancing dependency risk management methodologies in increasingly interconnected software environments.

REFERENCES

[1] Sonatype, *State of the Software Supply Chain Report 2023*, Sonatype, 2023.

[2] ReversingLabs, *Software Supply Chain Security Report 2023*, ReversingLabs, 2023.

[3] Verizon, *2023 Data Breach Investigations Report*, Verizon Business, 2023.

[4] Cybersecurity and Infrastructure Security Agency (CISA), "Advanced Persistent Threat Compromise of Government Agencies, Critical Infrastructure, and Private Sector Organizations," Alert AA20-352A, Dec. 2020.

[5] D. E. Sanger, N. Perlroth, and J. Barnes, "Russian hackers broke into federal agencies, U.S. officials say," *The New York Times*, Dec. 13, 2020.

[6] National Institute of Standards and Technology (NIST), "National Vulnerability Database (NVD)," [Online]. Available: https://nvd.nist.gov. Accessed: Feb. 2026.

[7] A. Alexopoulos, A. Kiayias, R. Talviste, and T. Zacharias, "A taxonomy of software supply chain attacks," in *Proc. IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2022, pp. 98–109.

[8] P. Allodi and F. Massacci, "Comparing vulnerability severity and exploits using case- control studies," *ACM Transactions on Information and System Security*, vol. 17, no. 1, Art. 1, 2014.

[9] L. Decan, T. Mens, and M. Claes, "On the topology of package dependency networks: A comparison of three programming language ecosystems," in *Proc. IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2016, pp. 526–530