A Contextualised Risk-Based Framework for Maven Dependency Updates

David Rodrigues Almeida[1], Isabel Azevedo[1]
[1] *ISEP — Instituto Superior de Engenharia do Porto, Portugal*
1181210@isep.ipp.pt | ifp@isep.ipp.pt

Keywords: Maven, Dependency Management, Software Supply Chain Security, Reachability Analysis, Risk-Based Framework.

Context. Modern software systems rely heavily on dependencies distributed through Maven Central, and vulnerabilities frequently reside in transitive rather than direct dependencies [1]. Tools such as OWASP Dependency-Check are effective at flagging these known vulnerabilities, yet a large share of the flagged code is never actually executed by the depending application [2].

Problem. Because such tools operate without this context, developers face a binary choice whenever a vulnerability is reported: upgrade immediately and risk breaking the build — a risk that materialises in a substantial share of Maven releases [3] — or postpone the update and accept the security exposure.

Objective. This work designs, implements and evaluates a Contextualised Risk-Based Framework (CRBF) that supports Maven dependency-update decisions by combining exploitability, reachability, ecosystem stability and compatibility signals into a single risk score, and by selecting an optimal remediation plan under a configurable effort budget.

Method. CRBF is implemented as a Maven plugin with a hexagonal architecture. For every dependency, known vulnerabilities are retrieved from the OSV database and enriched with EPSS exploitation scores; a static call graph built with WALA determines whether the vulnerable code is reachable from the project's own classes; binary compatibility between the current and candidate fix versions is assessed with JApiCmp; and ecosystem adoption and maintenance indicators are retrieved from the Goblin Weaver dependency graph [4]. These signals are aggregated into a contextual risk score, and a Z3 SMT solver selects the subset of upgrades that maximises risk reduction within the effort budget, following the constraint-based approach to dependency reasoning explored in prior work [5].

Results. CRBF has been implemented and exercised end-to-end on individual Maven projects. A comparative evaluation against OSV-Scanner, with and without call-graph analysis, across a broader sample of open-source Maven projects is currently under way, and its outcomes will be presented at the symposium.

Conclusions. By turning severity, exploitability, reachability, stability and compatibility into a single, effort-bounded recommendation, CRBF extends the balance between security and update cost explored in prior dependency-update research [6] and offers developers actionable guidance, addressing the low update rates that non-contextual vulnerability reports have been shown to produce [7].

References

[1] C. Nachuma, M. M. Hossan, A. K. Turzo, and M. Zibran, “Decoding Dependency Risks: A Quantitative Study of Vulnerabilities in the Maven Ecosystem,” in Proc. IEEE/ACM Int. Conf. Mining Software Repositories (MSR), 2025, pp. 270–274, doi: 10.1109/MSR66628.2025.00048.

[2] M. Keshani, S. Vos, and S. Proksch, “On the relation of method popularity to breaking changes in the Maven ecosystem,” Journal of Systems and Software, vol. 203, p. 111738, 2023, doi: 10.1016/j.jss.2023.111738.

[3] S. Raemaekers, A. van Deursen, and J. Visser, “Semantic versioning and impact of breaking changes in the Maven repository,” Journal of Systems and Software, vol. 129, pp. 140–158, 2017, doi: 10.1016/j.jss.2016.04.008.

[4] D. Jaime, J. El Haddad, and P. Poizat, “Goblin: A Framework for Enriching and Querying the Maven Central Dependency Graph,” in Proc. IEEE/ACM Int. Conf. Mining Software Repositories (MSR), 2024, pp. 37–41.

[5] A. G. Márquez, Á. J. Varela-Vaca, M. T. Gómez López, J. A. Galindo, and D. Benavides, “Depex: A software for analysing and reasoning about vulnerabilities in software project dependencies,” SoftwareX, vol. 30, p. 102152, 2025, doi: 10.1016/j.softx.2025.102152.

[6] D. Jaime, P. Poizat, J. El Haddad, and T. Degueule, “Balancing the Quality and Cost of Updating Dependencies,” 2025. [Online]. Available: https://hal.science/hal-04684254v1

[7] K. Huang et al., “Characterizing usages, updates and risks of third-party libraries in Java projects,” Empirical Software Engineering, vol. 27, no. 4, 2022.