

Tracking Dependencies and Security Risks in the Maven Architecture using Neo4J and Goblin Weaver

...

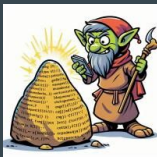
By Daniel Pang, Ahmed El Shatshat

The Problem ?

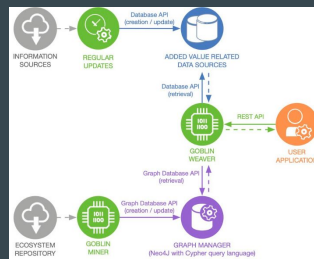
- Software security will always and forever be more and more important
- How can we leverage dependency management to improve security?
- Can we delve into Maven's history to see changes in dependency structure result in improved responses to security concerns?

The Data and Tools 🔧

- The Neo4J database of the enriched Maven Central dependency graph
 - This is needed for the CVEs and Freshness
- The Goblin Framework
- Simply, what is provided by the MSR 2025 Mining Challenge



<https://github.com/Goblin-Ecosystem>



<https://2025.msrconf.org/track/msr-2025-mining-challenge#Call-for-Mining-Challenge-Papers>

Research Questions 🔍

- 🏠 RQ1: Has Maven trended towards software architecture that has fewer dependencies over the years?
- ⌚ RQ2: Has the trend in software dependencies resulted in faster software lifecycles?
- 🗝️ RQ3: Has the frequency of software lifecycles affected security risks in Maven?

How to do the Work

- The RQs lend themselves to a natural order
- As such the rough schedule of milestones follow the same order
 - a. Identification of appropriate artifacts that have potential to answer our research questions (large changes in dependency, security adjacency)
 - b. Extraction of relevant artifacts, and assessment that they are indeed relevant to our research
 - c. Analysis of artifacts and trends that can be found therein
 - d. Analysis of security risks of artifacts that have had large dependency changes
 - e. Compilation of research to form conclusions, writing of paper

Threats to Validity

- Difficult to prove a reduction in dependencies is a result of a refactoring or change in software architecture
- Different CVEs will naturally have different complexities
 - Inherent variance in priority, difficulty, etc. influencing response time

Problems that can be Mitigated

- No or few artifacts with notable trends in dependencies
 - Alternative approach: compare artifacts by number of dependencies in general
- Similar approach for RQ2 and RQ3
 - If there are no trends that can be seen from a high level, can compare artifacts more granularly
 - 2. Do artifacts with fewer dependencies have faster software life cycles?
 - 3. Do artifacts with faster software life cycles have faster responses to CVEs?