CS846 Project Proposal: Tracking Dependency Updates & Security: Do Bots Make a Difference?

Jaffer Iqbal
j2iqbal@uwaterloo.ca
University of Waterloo
Waterloo, Canada

Eimaan Saqib
e2saqib@uwaterloo.ca
University of Waterloo
Waterloo, Canada

ACM Reference Format:
Jaffer Iqbal and Eimaan Saqib. 2025. CS846 Project Proposal: Tracking Dependency Updates & Security: Do Bots Make a Difference?. In . ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Modern software projects heavily use open-source dependencies. But managing these dependencies in large projects comes with risks that include security risks, technical debt, compliance issues, etc. Vulnerabilities in dependencies are a significant cause of software breaches. For example, the 2021 Log4j vulnerability (CVE-2021-44228) exposed millions of systems globally and demonstrated the high risk of unpatched vulnerabilities in the software ecosystem [7]. A 2024 report by Synopsys found that 84% of open-source codebases still contained at least one open-source vulnerability, while 74% included high-risk vulnerabilities, marking a 54% increase from 2022 [2, p. 6]. This highlights the ongoing need for improved dependency management and maintenance practices.

To address these challenges, automated dependency management tools (often referred to as *dependency management bots*) have been introduced to automate the process of updating dependencies. Several tools provide automated dependency management services, including GitHub's Dependabot [4], Renovate Bot [6], Snyk [9], and Depfu [1], which have been developed to help maintain dependency hygiene, security, and software updates. While these tools are used across open-source projects, their effectiveness in real-world ecosystems such as Maven Central (the largest repository of Java libraries) remains unclear. To make informed investment and adoption decisions, organizations need empirical evidence of how these tools impact software security, maintenance practices, and dependency stability within large-scale open-source projects.

This study seeks to bridge this gap by evaluating the effectiveness of Dependabot, Renovate Bot, Snyk, and Depfu's dependency management practices within the Maven ecosystem.

2 DATASET AND TOOLS

This study will use the **Goblin framework**, a Neo4J-based dependency graph of Maven Central artifacts, that has millions of nodes that represent libraries and their releases. We will use Goblin's **Weaver** tool to compute metrics such as **dependency freshness** (time since the latest version of a dependency was adopted) and **vulnerability exposure windows** (days between a vulnerability's disclosure and the adoption of its patched version).

Our study focuses on analyzing the impact of automated dependency management tools, specifically **GitHub Dependabot**, **Renovate Bot**, **Snyk** and **Depfu**. To identify the relevant open sourced projects that use these tools, we will use the GitHub API and get configuration files and CI/CD workflows (see Section 3). We will use Python libraries like `Pandas`, `Scipy`, and `Plotly` to perform statistical analysis and visualize trends in dependency management behavior.

3 RESEARCH QUESTIONS

(1) **RQ-1:** How often do projects update their dependencies, and what factors influence this frequency (e.g., project size, popularity, type)?
(2) **RQ-2:** What is the average time taken to patch vulnerabilities in dependencies, and how does this vary across projects?
(3) **RQ-3:** Does the adoption of dependency management bots correlate with reduced dependency update latency and vulnerability exposure windows?

4 METHODOLOGY

This project will have the following three phases:

Phase 1: We will establish baseline trends by analyzing Maven projects stratified by size (number of dependencies or lines of code), type (libraries, applications, or frameworks – inferred from metadata and repository descriptions), and popularity (dependents, GitHub stars), etc. The Goblin dependency graph will be used to calculate dependency update frequency and vulnerability patching times. Statistical analysis will be used to identify relationship between project attributes and dependency management behavior.

Phase 2: We will evaluate the impact of the dependency management tools as case studies. To identify open-source projects utilizing these tools, we will first query the GitHub API [3] to locate repositories containing relevant configuration files, CI/CD workflow definitions, or automated pull request activity. For instance, in the case of Dependabot, we first look for repositories containing the `dependabot.yml` configuration file. Repositories with a `dependabot.yml` file are assumed to use GitHub's Dependabot to automate dependency updates and management, as highlighted in [5]. Once a repository is confirmed to have this file, we will extract its `Group ID` and `Artifact ID` and use them to query the Maven Central Repository [8] to verify whether the project exists

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

as a publicly available artifact. Finally, with this information, we will leverage the Neo4J Maven Central dependency graph and execute Cypher queries to filter and analyze the relevant artifacts and release data within the dependency graph.

Phase 3: We will combine the results of the statistical analysis to generate visualizations, insights, and inferences. We will produce a final report to summarize the findings.

4.1 Milestones

- **Week 1-2:** Data extraction from Maven Central and GitHub API integration
- **Week 3-4:** Statistical analysis of baseline trends
- **Week 5-6:** Case study data collection and hypothesis testing
- **Week 7:** Gathering insights and preparing final report

5 THREATS TO VALIDITY

Repositories containing a `.yml` configuration file (like `dependabot.yml`) may not actively use the dependency management bot. This can occur due to several reasons, such as mis-configured settings, disabled automation, archived or inactive repositories, or a lack of dependencies matching the update criteria. To mitigate this, we can verify active bot usage by checking for pull requests authored by the bot, analyzing commit histories for dependency updates, and querying repository settings to confirm if the bot is enabled. This ensures that only repositories with demonstrable bot activity are considered.

There is also a risk of selection bias as projects using dependency management bots may inherently prioritize security and better dependency management as compared to other projects, which could skew results. To mitigate this, we will try to use control groups of similar project size, popularity, and domain. Moreover, causal ambiguity will need to be acknowledged (correlation does not equal causation). If time permits, we will try to perform a longitudinal study to isolate tool impact.

REFERENCES

[1] Depfu. [n. d.]. Depfu. https://depfu.com/.

[2] Synopsys Black Duck. 2024. *Open Source Security and Risk Analysis Report.* Technical Report. Black Duck by Synopsys. 18 pages. https://www.blackduck.com/resources/analyst-reports/open-source-security-risk-analysis.html Accessed: 2024-02-21.

[3] GitHub. [n. d.]. Github REST Api Documentation. https://docs.github.com/en/rest?apiVersion=2022-11-28

[4] Github. [n. d.]. GitHub's Dependabot. https://github.com/dependabot.

[5] Runzhi He, Hao He, Yuxia Zhang, and Minghui Zhou. 2023. Automating dependency updates in practice: An exploratory study on github dependabot. *IEEE Transactions on Software Engineering* 49, 8 (2023), 4004–4022.

[6] Michael Kriese. [n. d.]. Renovate Bot. https://github.com/renovatebot.

[7] Red Hat Customer Portal. [n. d.]. 2021 Log4j vulnerability (CVE-2021- 44228. https://access.redhat.com/security/cve/cve-2021-44228.

[8] Sonatype. [n. d.]. Maven Central Repository Search. https://central.sonatype.com/?smo=true.

[9] Synk. [n. d.]. Synk Bot. https://github.com/snyk-bot.