A Study into the Use of Version Locking in Gradle Projects

1. Introduction

- Developers often use open-source code for functionality that is not specific to their project
- Dependency managers, like Gradle, are used to make managing dependencies easier
- Breaking changes in dependency updates can cause projects to break unexpectedly
- Version locking was introduced to Gradle to lock specific dependency versions in place
- We analyzed how version locking is used in Gradle projects

Research Questions

- RQ1: How widespread is the use of version locking?
- RQ2: How widespread is the use of version ranges in projects that also use version locking?
- RQ3: How can we measure the effect of version locking?

2. Methodology

- We sampled over 12,000 Gradle projects and analyzed how many of those contained a lock file.
- We constructed a dataset of 47 projects using version locking
- We analyzed how many of those also used version ranges and what these ranges looked like
- We tried building these projects after deleting the lock files, which disables version locking, to see how often version locking prevents a build to fail
- We re-resolved the dependencies for these projects and compared the new lock file with the original one, to see how often version locking causes newer versions of dependencies are ignored

3. Results

- Only 0.34% of analyzed Gradle projects contained a lock file
- 29.5% – 44.4% of projects using version locking, also used version ranges
- For 18.2% of the projects, version locking prevents the build to fail
- Up to 9.8% of dependencies in projects using version locking, are at risk of containing known vulnerabilities, as they are locked to an older version

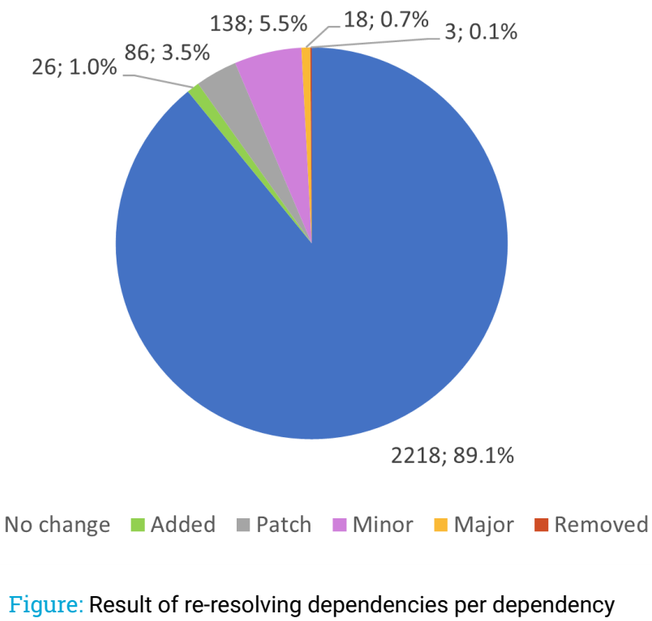

Figure: Result of re-resolving dependencies per dependency

4. Discussion

- The low adoption of version locking in Gradle projects suggests either a lack of awareness on the part of developers or a preference to keeping dependencies up-to-date
- Less than half of projects using version locking, used version ranges. This is surprising as version locking is most useful when permissive declaration strategies are used

5. Conclusion

- We conclude there may be a need for better education about the use and benefits of version locking to developers
- Future work could investigate the security implications further, which might also convince more developers to adopt it, as little information about these risks are currently known.

Author: Joppe Boerop (j.w.boerop@student.tudelft.nl)

Supervisor: Cathrine Paulsen

Responsible Professor: Sebastian Proksch

TU Delft Delft University of Technology