ABSTRACT
Developers are known to keep third-party dependencies of their projects outdated even if some of them are affected by known vulnerabilities. In this study we aim to understand why they do so. For this, we conducted 25 semi-structured interviews with developers of both large and small-medium enterprises located in nine countries. All interviews were transcribed, coded, and analyzed according to applied thematic analysis. The results of the study reveal important aspects of developers’ practices that should be considered by security researchers and dependency tool developers to improve the security of the dependency management process.

KEYWORDS
Dependency management, security, vulnerable dependencies, qualitative study, interviews

1 RESEARCH PROBLEM AND MOTIVATION
Components with known vulnerabilities (#9 from OWASP Top 10 list of Web Application Security Risks) are the most frequent cause of severe security breaches: according to the Snyk report, known vulnerable components were the root cause of 24% of severe security breaches, like the Equifax breach due to an outdated Apache Struts library, the Panama Papers data leak due to an old unpatched version of Drupal, and the Ubuntu forum breach due to an outdated Forumrunner add-on. Still, developers often keep third-party components used in their projects outdated.

There is a strong temptation to fingerprint FOSS developers for lack of care. However, a more careful analysis reveals more nuances in these broad findings. For example, a study of the Android ecosystem [3] argued that many libraries were vulnerable and could be easily updated. A later study by the same group [5] showed that the original claim was too optimistic: the ‘easy’ update would instead create breaking changes in around 50% dependent projects. Similarly, an initial study on the Maven ecosystem [7] argued that many libraries included vulnerable dependencies. A later study [9] showed that several of those vulnerabilities were in test/development libraries and thus not shipped with the product and, therefore, irrelevant. Developers may, therefore, not be entirely irrational in not always updating their libraries.

Hence, understanding the developers’ decision-making strategies while selecting and/or updating dependencies of their projects is important for both security researchers and dependency tool developers, so they can design appropriate methodologies and tools to improve the security of the dependency management process.

2 BACKGROUND
Quantitative empirical studies of software dependencies (e.g., [7, 9]) mainly focus on the techniques, and therefore, facilitate the ways how developers perform dependency management. However, they provide limited insights on the developers’ motivations while managing software dependencies, such as why developers adopt new dependencies or update/not update the already used ones. Instead, we are interested in understanding the developers’ reasoning while selecting and updating dependencies.

On the other hand, qualitative dependency studies (e.g., [1, 2]) suggest that dependency issues might affect developers’ decisions. However, the studies focus mainly on functionality issues and do not investigate the influence of security concerns.

The qualitative studies of technologies and tools for automating the software engineering process (e.g., [6, 11]) provide interesting insights into the software developers’ experience, but do not consider the dependency analysis tools, and therefore, do not study how developers can use them to discover and mitigate security issues introduced by software dependencies.

The studies on information needs (e.g., [8, 10]) provide useful insights on developers’ decision-making strategies, however, the currently available studies do not show how the developers’ actions and decisions change due to security concerns and the presence of the issues introduced by software dependencies.

3 APPROACH
To find the incentives of developers’ motivations for (not) updating dependencies of their projects, we interviewed developers of 25 different companies located in 9 countries and analyzed their strategies for (i) selecting new dependencies, (ii) updating currently used dependencies, (iii) using automatic dependency management tools, and (iv) manually managing dependencies.
We summarize our findings of the facts the developers reported us. Small and medium-sized enterprise (SME), and user group (UG) widely known, and adoption of the fixed dependency version does not require significant efforts. Current dependency analysis tools (if used) only facilitate the form sensitive dependency management tasks (e.g., updates) manually. Security vulnerabilities, updates, and low-priority alerts. Developers complain that dependency tools produce many false-positive and low-priority alerts.

Unfixed vulnerabilities. The interviewed developers suggested the following actions when a vulnerability is discovered in a dependency, but no newer version fixes it:
- assess whether this vulnerability impacts them since maybe they may not use that particular functionality;
- wait for the fix or a community workaround;
- adapt own project, i.e., disable vulnerable functionality or rollback to a previously safe version of the library;
- maintain own fork of a dependency project (possibly fixing and making a pull request to the dependency project).

5 CONCLUSIONS

We present a qualitative study of developers’ decision-making practices in the presence of security concerns for selecting new dependencies, updating the already used ones, usage of dependency analysis tools, and coping with vulnerable dependencies that do not have a fixed version.

Our study provides important insights and suggestions for security researchers on how to design better approaches for secure dependency management and dependency tool developers on how to improve the dependency analysis tools.

ACKNOWLEDGMENTS

This research has partly funded by the EU under the H2020 Programs H2020-EU.2.1.1-CyberSec4Europe (Grant No. 830929) and the NeCS: European Network for Cyber Security (Grant No. 675320).

REFERENCES