An In-Depth Analysis of the CodeQL Ecosystem

Jean-Charles Noirot Ferrand, Patrick McDaniel
University of Wisconsin–Madison

MADS&P

Overview

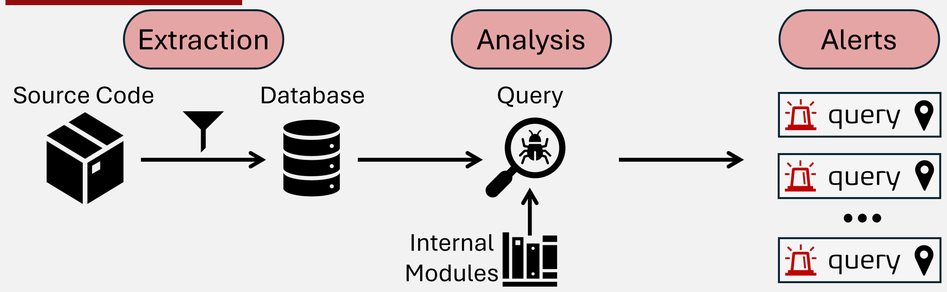
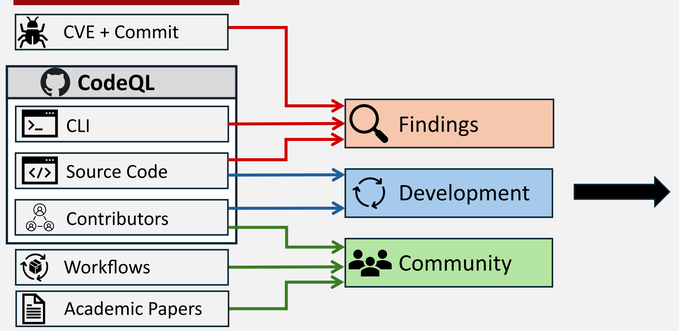


Figure 1: A CodeQL Analysis

- Software relies on **automated vulnerability detection** to improve the security
- **CodeQL** (from GitHub) is the most used static analysis tool
- We measure the full **ecosystem's health** through **three interconnected axes**

Methods



Data Sources

- **CodeQL**: Git and GitHub repository data, changelogs, CLI binaries
- **CVEFixes**: CVEs and corresponding fixing commit
- **Papers**: 1K+ papers mentioning "CodeQL"
- **Users**: 3M+ GitHub workflows histories

Three Axes

Security Queries

- Properties of **security queries**
  - **Accuracy:** Can the query identify CVEs?
  - **Sophistication:** How many dependencies?
  - **Lifecycle:** How does a query evolve?

Community

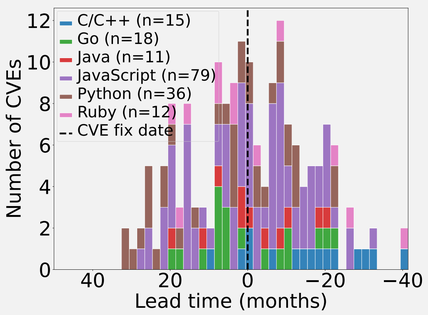
- Impact on academia and open-source software.
  - **Adoption:** Usage as the de facto static analyzer?
  - **In Practice:** How does it shape OSS/Academia?

</> Development

- Characterization of the evolution of the tool
  - **Developers:** Who contributes to the tool?
  - **Implications:** Practical impact on detection?

Measurement

Queries & Properties

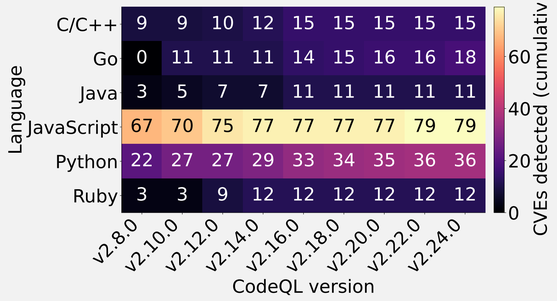


Impact on CVE detection

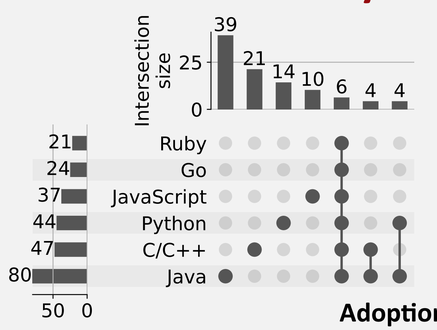
- Lead time to detect CVEs
- Mostly JavaScript/Python
- >50 CVEs could have been found
- **Limitation:** Automated filtering (same files and lines as fix commit)

Accuracy (CVEFixes)

- Low accuracy (comparable to prior studies), **little changes** across versions
- The **evolution** of OSS (languages, frameworks) needs to be reflected



Community & Development

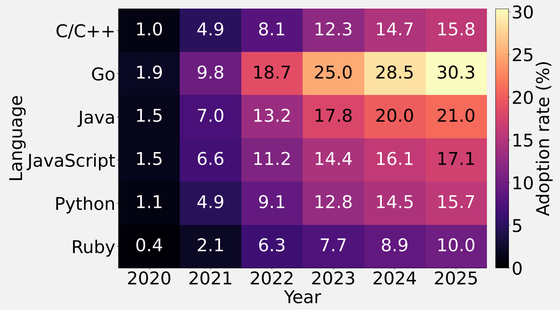


Language Expertise

- Heavy Java/C++ expertise
- Minority of one-language only contributors (42% for Java)
- **6** maintainers on all languages

Adoption Rate

- Study from GitHub Actions workflows histories of 40k+ repositories
- Observed **misalignment** between development and adoption (Go, C/C++)
- Start of adoption in 2022, convergence between 2024-2025



Future Work

Analysis Improvements

- Adoption from **papers** (academia) and **open-source software**
- Characterization of the number of findings (**sensitivity**)

Understanding Queries Development

- Characterization of a security query and its scaffolding **evolution** across time → Addition/Deletion, Scope changes,…
- Improvement of internal library detection (**reachability**)

Security Implication

- Mapping from CodeQL changes to **CVE lead time**
- What matters the most when building **security heuristics** for open-source software program analysis?

Contact

https://jcnf.me
jcnf0
jcnf@cs.wisc.edu