Vulnerabilities in Open-Source Language Servers

Henry Post
Intro to OS – New York University

Problem and motivating example

- Open-source software (OSS) is everywhere! We use it all the time without knowing
- When performing software development, devs use IDEs (code editors)
  - These code editors use software called “Language Servers” to make coding easier
- Automated code testing (SAST, DAST, SCA) is not always used on OSS
- Because language servers are so critical to software development, they could be vulnerable, and any vulnerabilities would present in many developers’ workflows

Hypothesis

- Language servers are very widely used and contributed to
- Because of this, they may have a large number of unknown security vulnerabilities

Language Server Name	Stars	Forks
ccls	3500	273
clangd	22100	8600
Eclipse JDT Language Serve	1500	380
gopls	7000	2200
intelephense	1500	86
Kotlin Language Server	1300	178
omnisharp-roslyn	1600	419
Phan	5500	372
php-language-server	1100	223
phpactor	1100	119
PHPUnit for VSCode	113	38
PowerShell Editor Services	553	208
Python LSP Server	2600	307
Serenata	62	?
Solargraph	1800	148
Sorbet	3500	481
TypeScript Language Server	1500	158

Related research: Stealthily adding flaws

- This paper outlines (and the author actually does this in a real OSS project) how an attacker could, over the course of multiple different commits, introduce a vulnerability into OSS
- This shows how, while OSS ecosystems are extremely powerful and useful for different reasons, they can still be manipulated and exploited
- [1] Wu, Qiushi and Kangjie Lu. “On the Feasibility of Stealthily Introducing Vulnerabilities in Open-Source Software via Hypocrite Commits.” (2021).

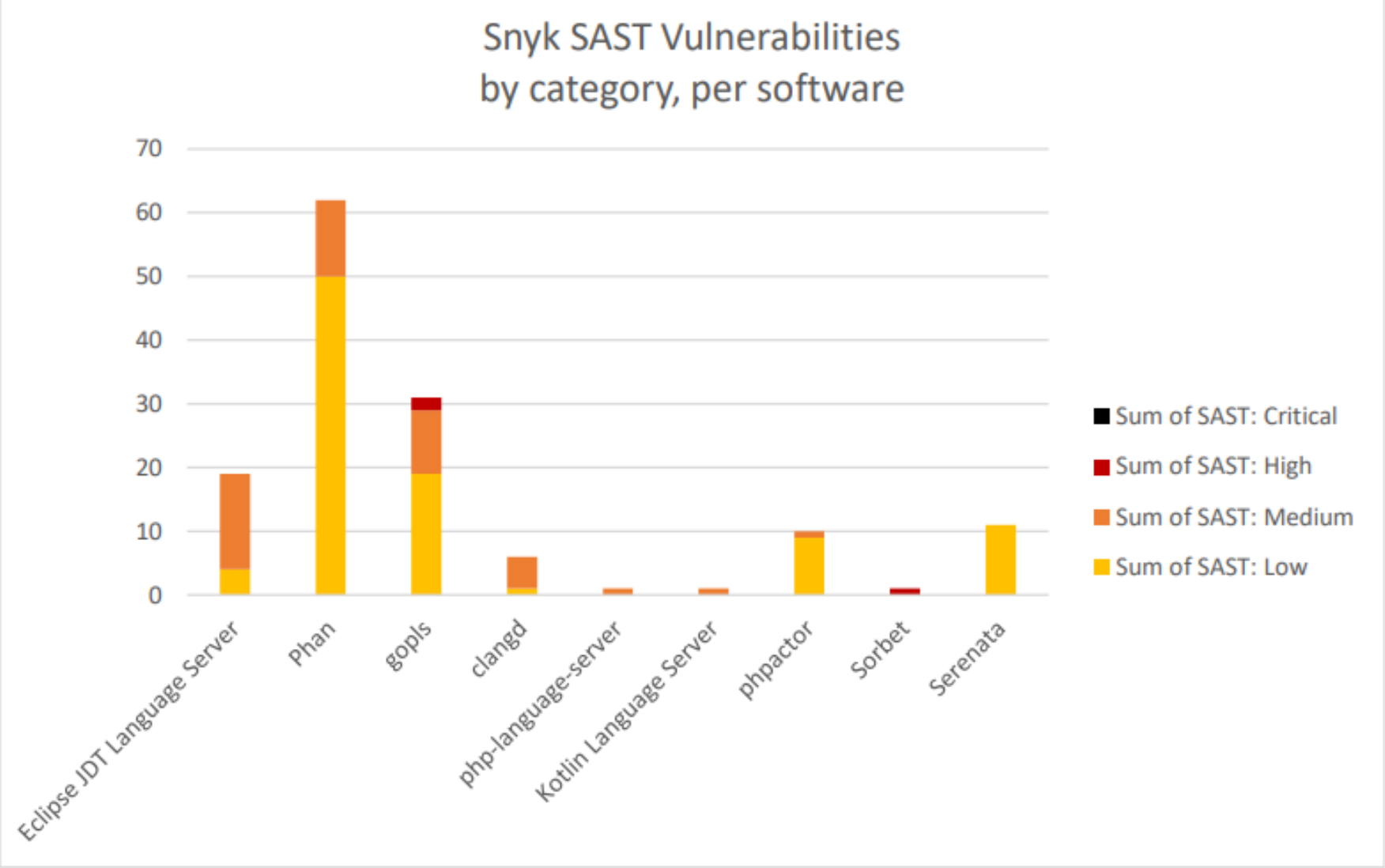
Related research: Vulnerability metadata

- This paper describes a system for attaching context to an identified vulnerability that affects an OSS component, in order to understand if it is actually exploitable, given the context in which the OSS component is used by a software application.
- This is a useful paper because some vulnerabilities identified by SCA (Software Composition Analysis) tools are actually not exploitable depending on how the OSS components are used
- I didn’t use this paper in my paper, but its methods and conclusions would be useful to reduce SCA noise.
- [3] H. Plate, S. E. Ponta and A. Sabetta, "Impact assessment for vulnerabilities in open-source software libraries," 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), Bremen, Germany, 2015, pp. 411-420, doi: 10.1109/ICSM.2015.7332492.

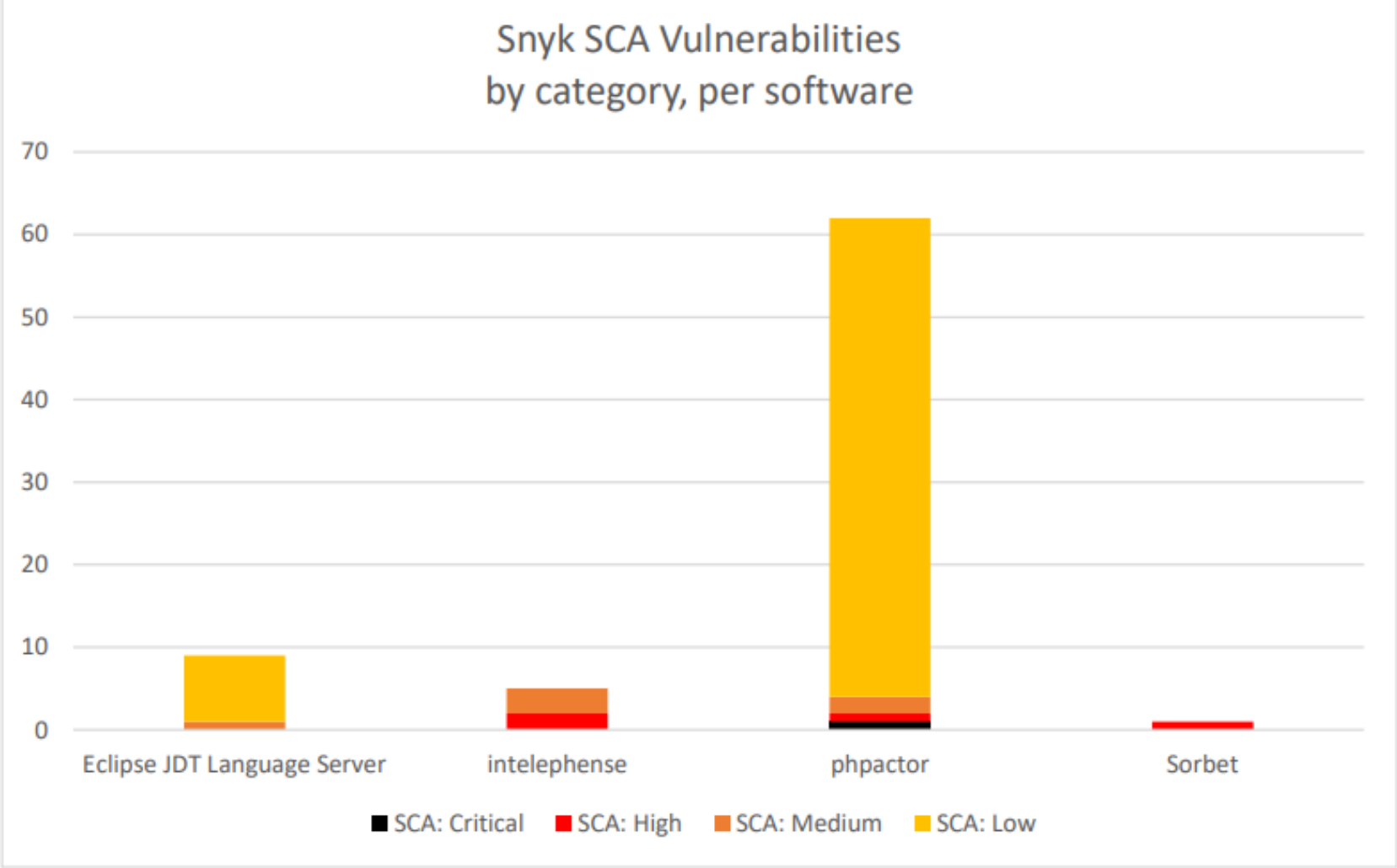
Findings: Overview

- 17 different language servers were chosen, all vary in star count, fork count, and language.
- Almost all had some SAST and SCA vulnerabilities.
- Some had invalid findings, since test code got scanned
- Scanning tools used:
  - Snyk SCA
  - Snyk SAST
  - SonarQube SAST

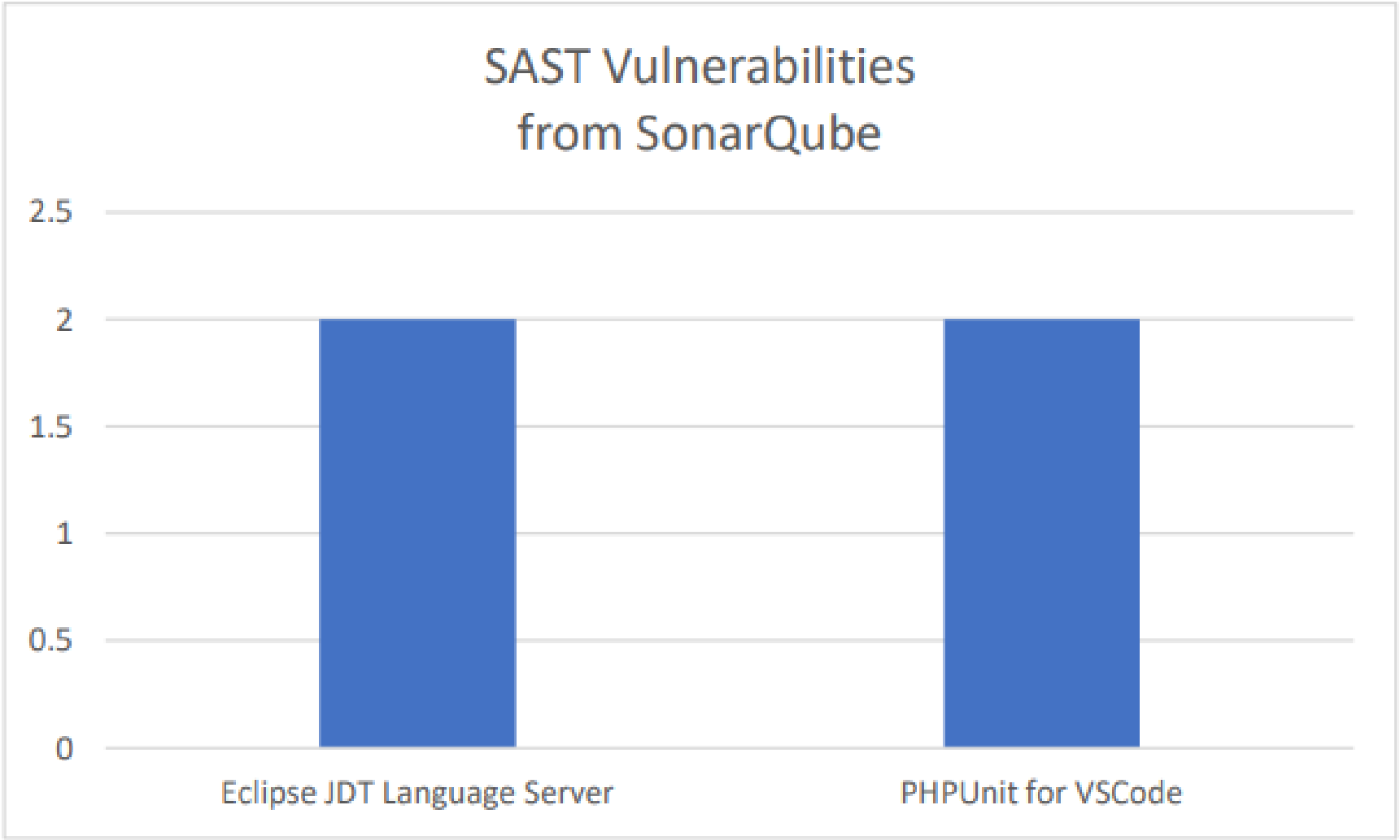
Findings: Snyk SAST



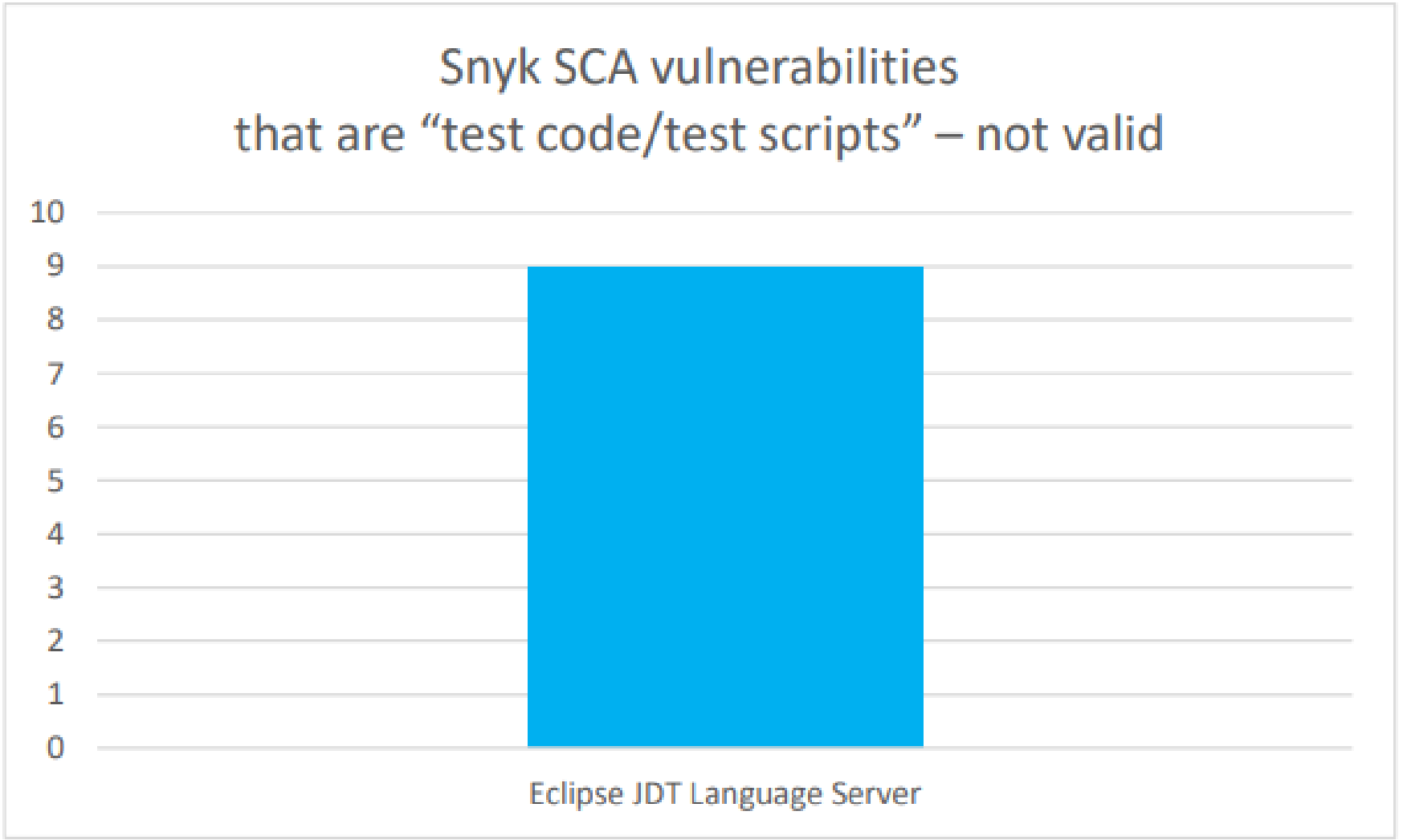
Findings: Snyk SCA



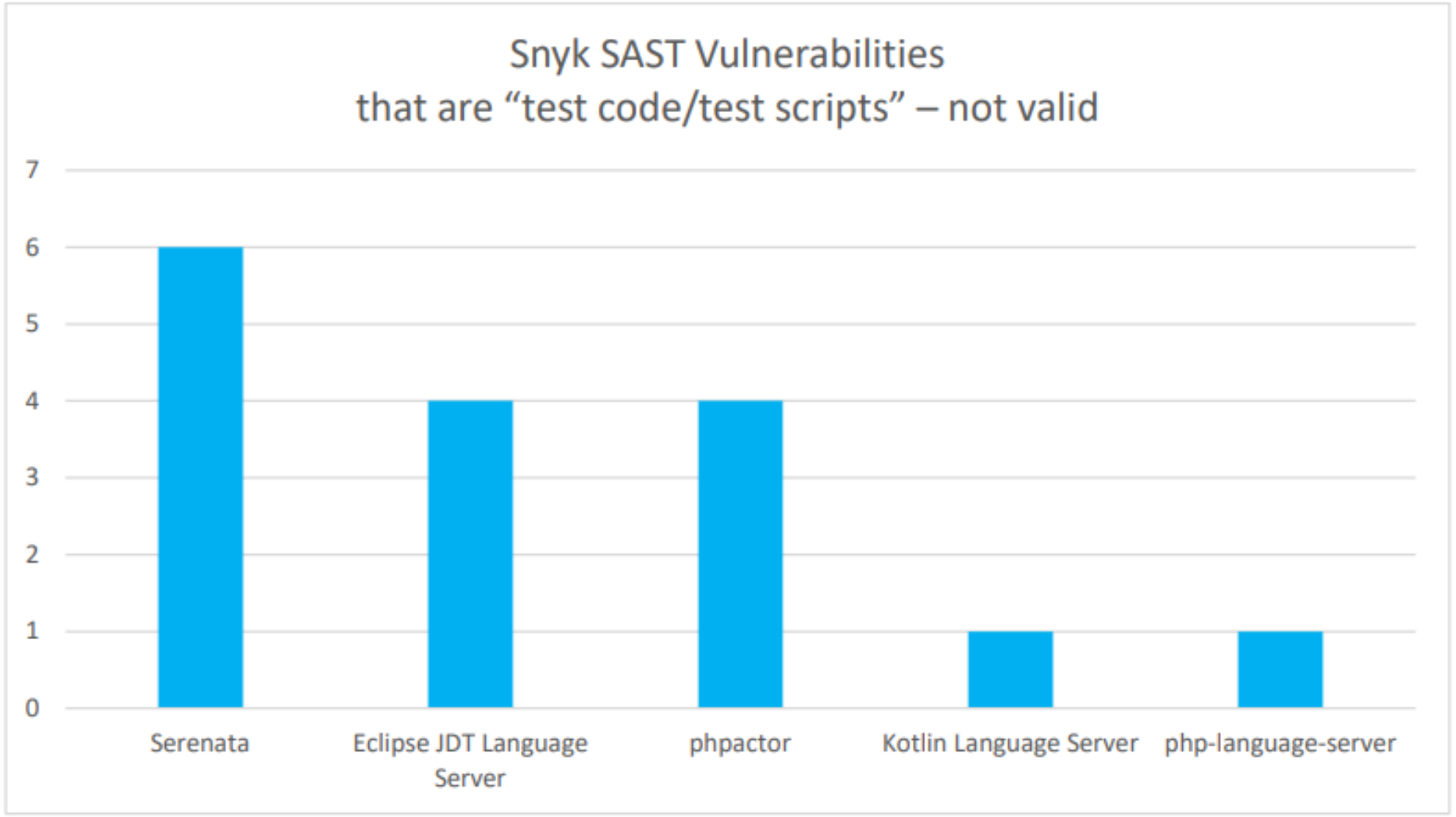
Findings: SonarQube SAST Vulnerabilities



Findings: Test code that shouldn’t be scanned: Snyk SCA



Findings: Test code that shouldn’t be scanned: Snyk SAST



Findings: Conclusions

- Many open-source projects (not just language servers) might have numerous high-severity vulnerabilities that their authors have no idea about
  - 9 projects have valid Snyk SAST vulnerabilities, 4 projects w/ valid Snyk SCA vulnerabilities, and 2 projects with SonarQube SAST vulnerabilities
- Those “test code” findings waste time – We need to automate their removal!
  - 11.2% SAST and 11.6% SCA findings only showed up because of test code getting scanned

Next Steps

- Automate my work and expand this to 1000’s of repositories on GitHub?
  - Can be done easily with shell scripts and the Snyk REST API
- An automated system for evaluating the security posture of any open-source software project
- A “software security health” scorecard system that can be used to give a score to open-source software projects
- An AI-powered system that can evaluate open-source software projects for potential security issues based on code patterns and open-source contributor behaviors
- Examine how various projects use security checks within their CI/CD pipelines
- A thorough review and comparison of various SAST and SCA tools, to determine which ones offer the most comprehensive and accurate results