Using Github to Create a Dataset of Natural Occurring Vulnerabilities

Sofia Reis[1] & Rui Abreu[2]

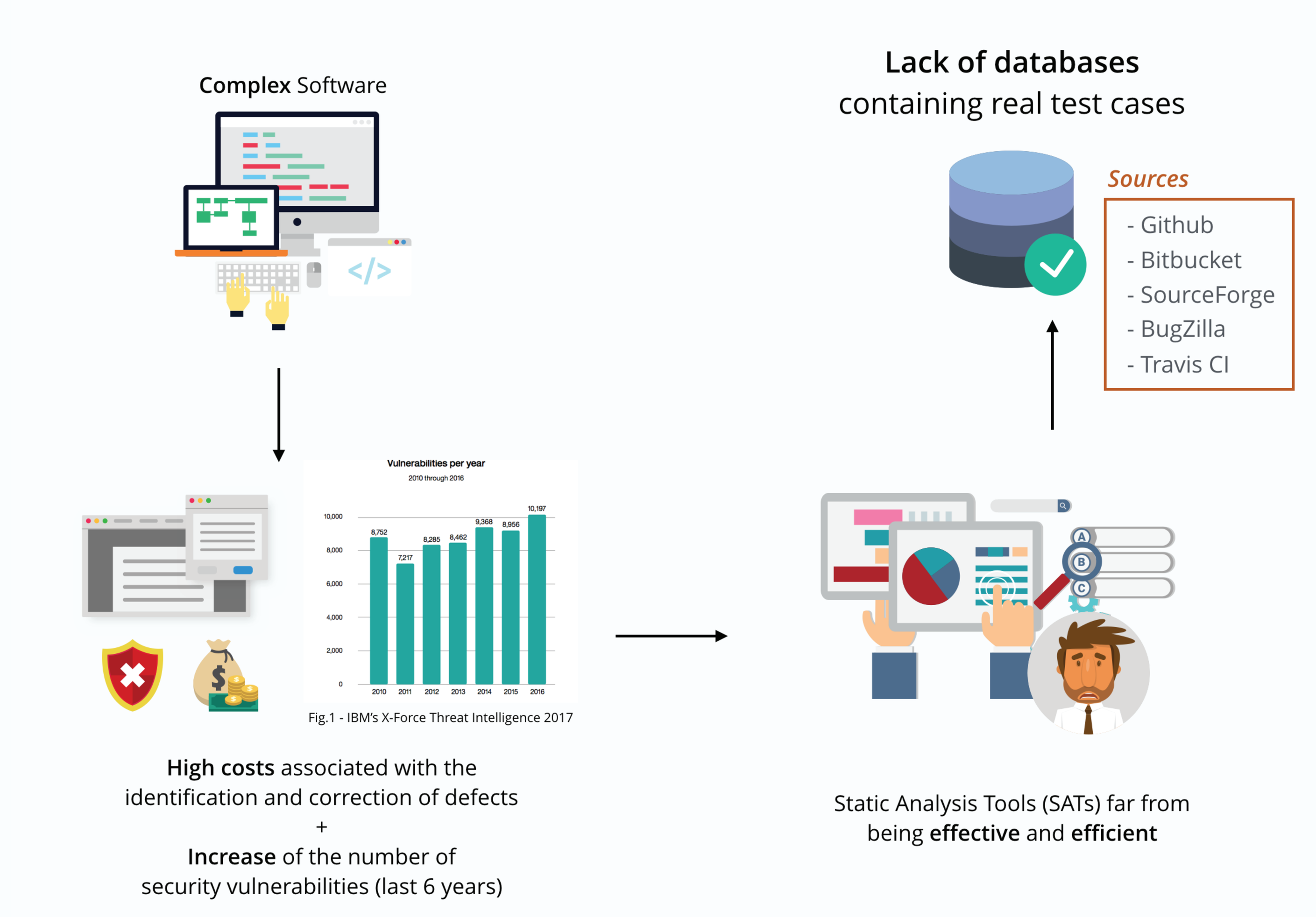
[1] Faculty of Engineering of University of Porto, Portugal

[2] IST, University of Lisbon & INESC-ID, Portugal

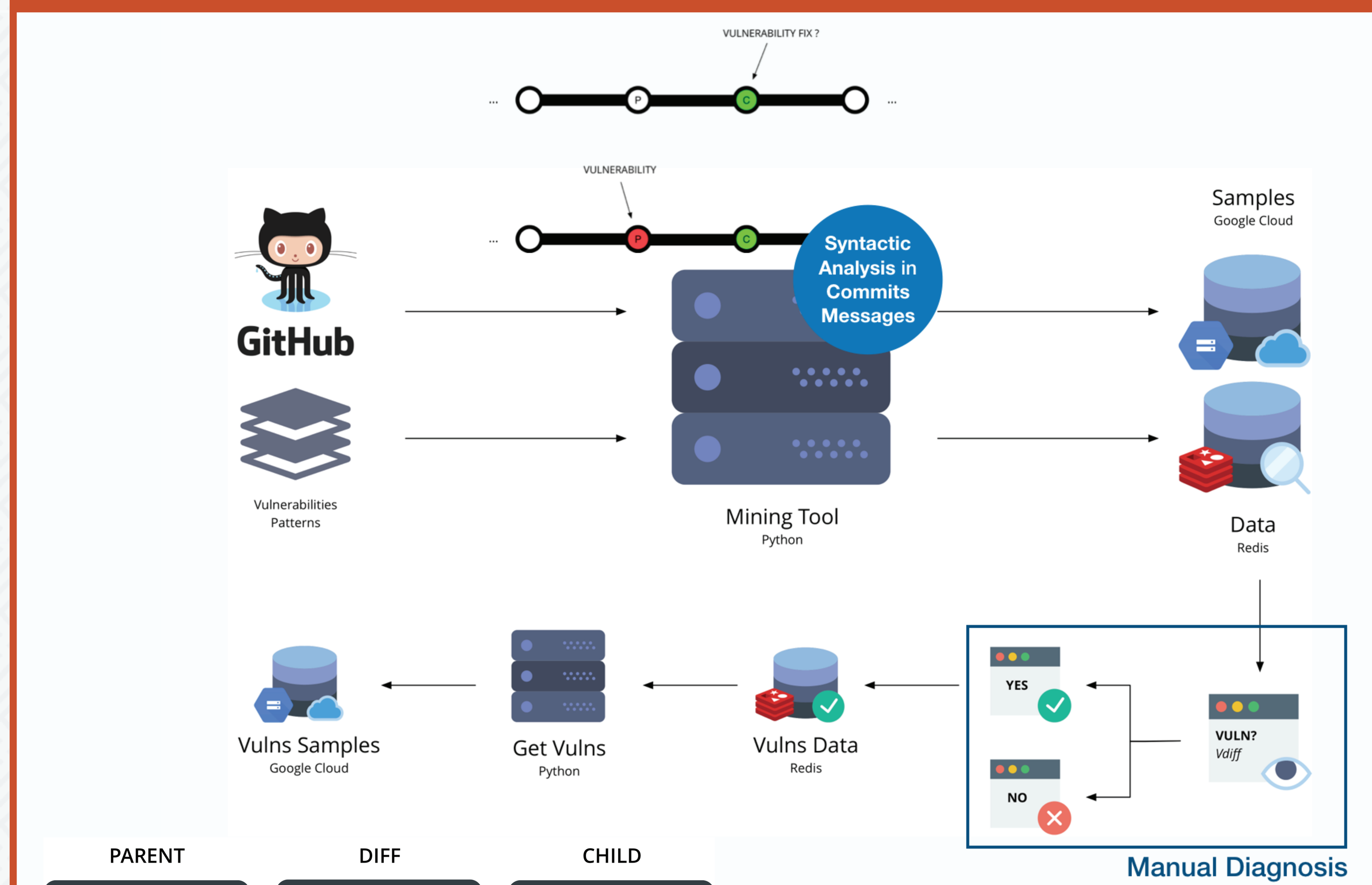
ABSTRACT

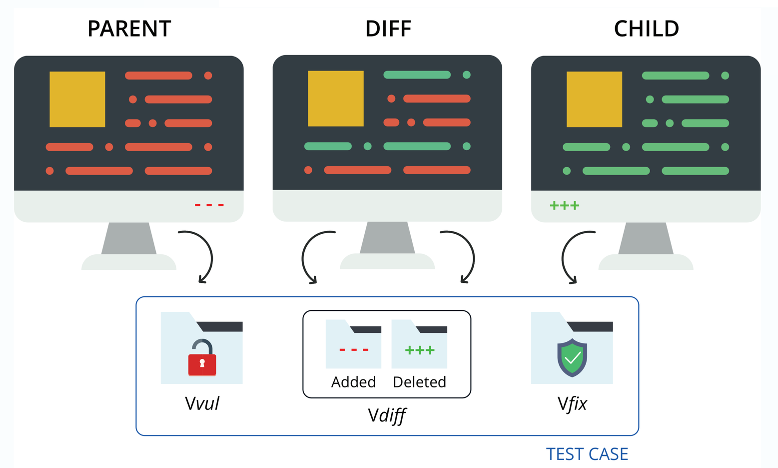
Currently, to satisfy the potential high number of system requirements, complex software is crafted which turns its development cost-intensive and more susceptible to security vulnerabilities. In software security testing, empirical studies typically use artificial faulty programs because of the challenges involved in the extraction or reproduction of real security vulnerabilities. Thus, researchers tend to use databases of hand-seeded vulnerabilities, which may differ inadvertently from real vulnerabilities and thus might lead of misleading assessments of the capabilities of the tools. **Secbench** is a database of security vulnerabilities mined from GitHub which hosts millions of open-source projects carrying a considerable number of security vulnerabilities. We mined **238 repositories** - accounting to more than **1M commits** - for **16 different vulnerability patterns**, yielding a Database with **602 real security vulnerabilities**.

INTRODUCTION



EXTRACTING AND ISOLATING VULNERABILITIES FROM GITHUB REPOSITORIES





TEST CASES STRUCTURE - Juliet Test Cases

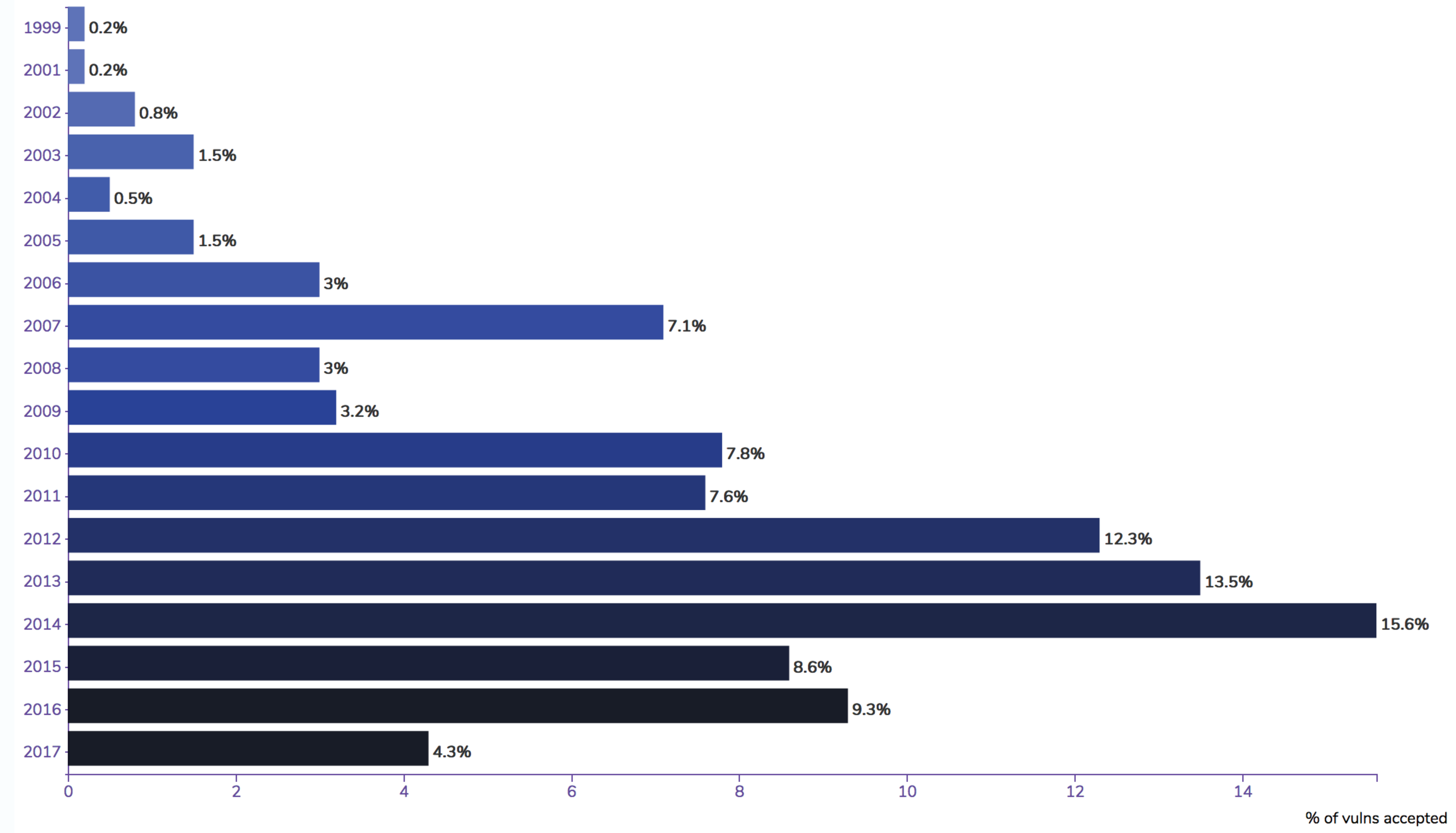
Vvul - Vulnerable source code from the parent commit.

Vfix - Non-vulnerable source code from the commit where the pattern was caught.

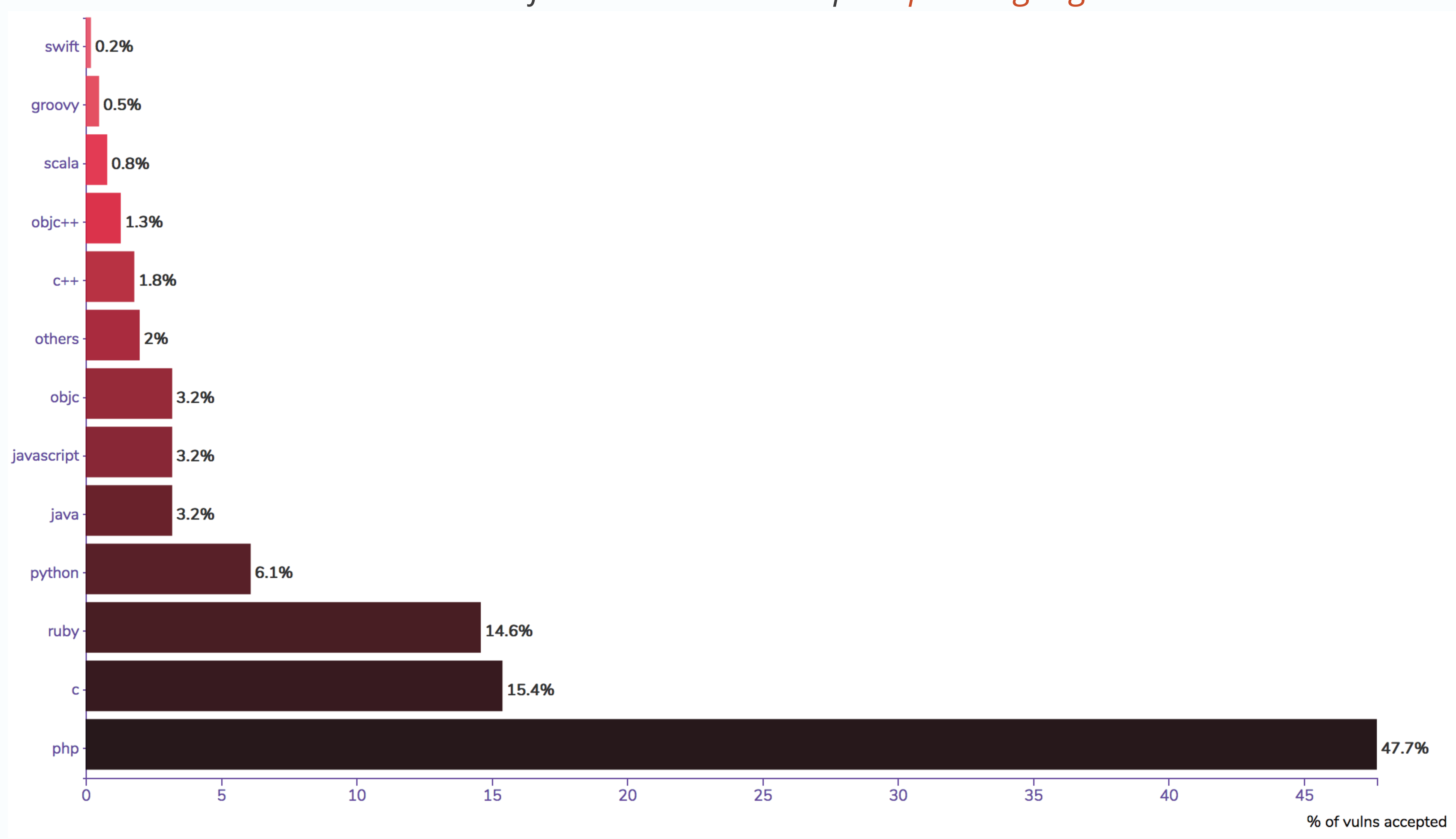
Vdiff - Two folders, containing the added lines to fix the vulnerability and the deleted lines representing the security vulnerability.

RESULTS

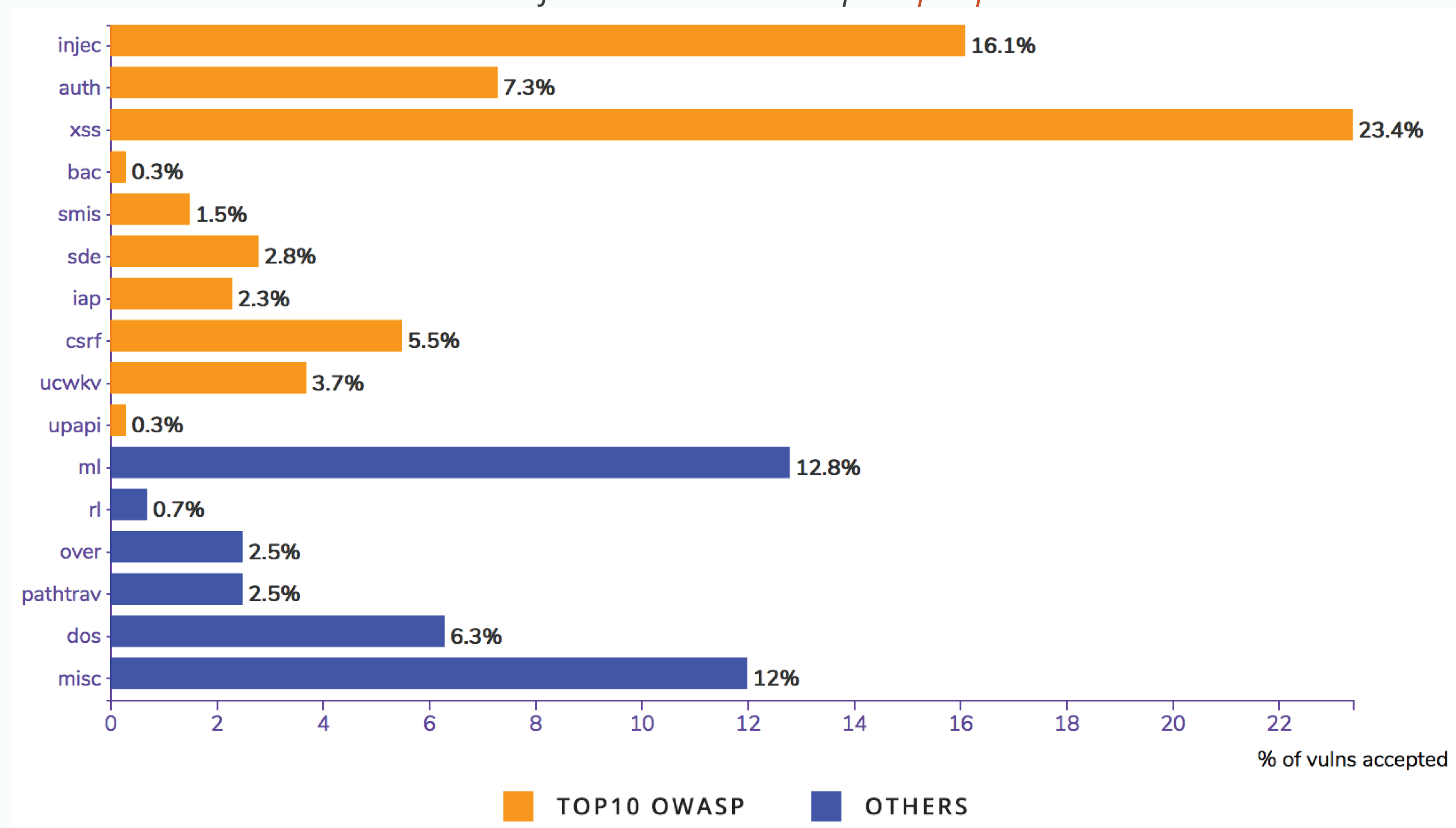
Number of vulnerabilities accepted per year



Number of vulnerabilities accepted per language



Number of vulnerabilities accepted per pattern



RESEARCH QUESTIONS

RQ1: Is there enough information available on open-source repositories to create a database of software security vulnerabilities?
A1: There are enough vulnerabilities available on open-source repositories to create a database of real security vulnerabilities.

# of MVulns	# of MRepositories	% of MRepositories
> 0	150	63.03%
$= 0$	88	36.97%
Total	238	100%

Table 1: Mined Vulnerabilities Distribution

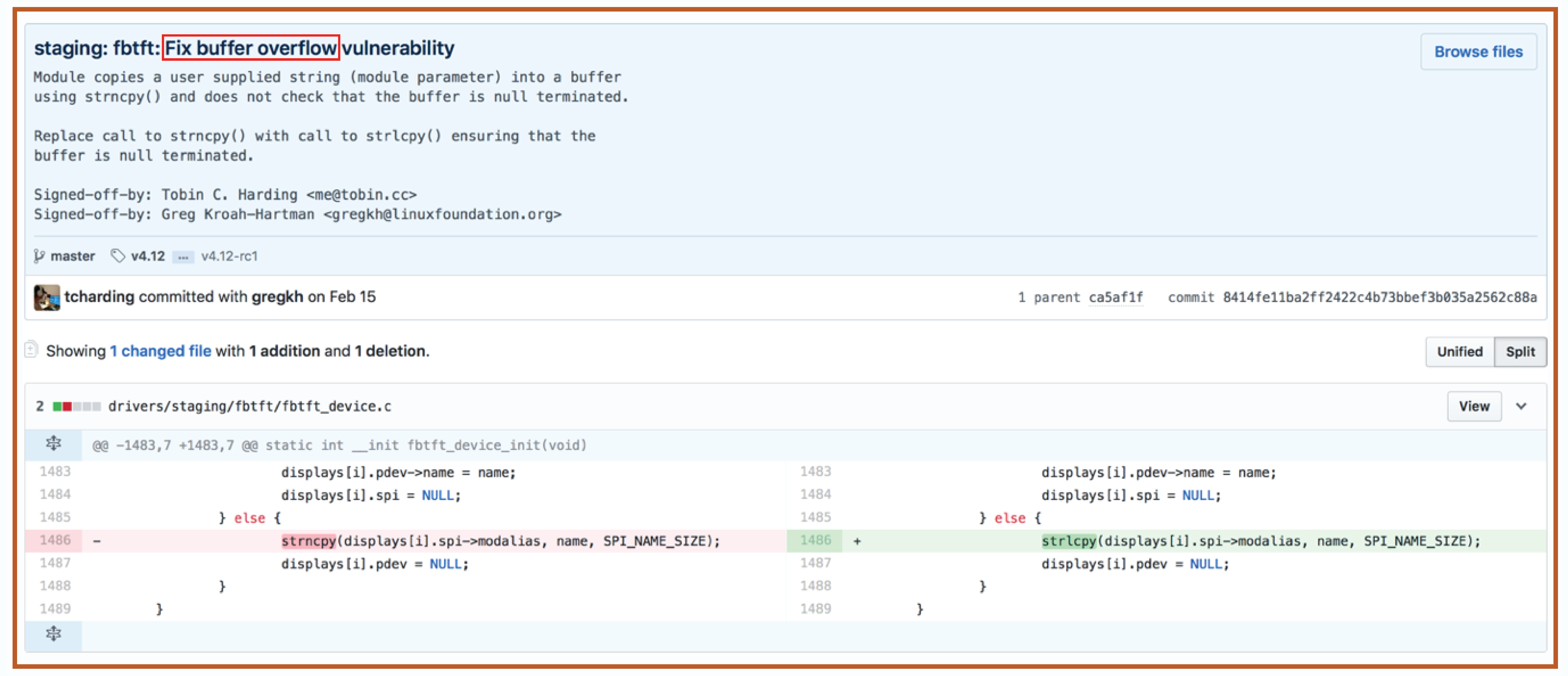
# of RVulns	# of VRepositories	% of VRepositories
> 0	79	52.67%
$= 0$	71	47.33%
Total	150	100%

Table 1: Accepted/Real Vulnerabilities Distribution

RQ2: What are the most prevalent security patterns on open-source repositories?
A2: The most prevalent security patterns are Injection, Cross-Site Scripting and Memory Leaks.

HOW ARE VULNERABILITIES IDENTIFIED?

AUTOMATED IN COMMITS MESSAGES AND **MANUALLY** IN SOURCE CODE



Example 1 - Buffer Overflow

Example 2 - Two Resource Leaks

CONCLUSIONS & FUTURE WORK

The importance of this database is the potential to help researchers and practitioners alike improve and evaluate software security testing techniques. We have demonstrated that there is enough information on open-source repositories to create a database of real security vulnerability for different languages and patterns. And thus, we can contribute to considerably reduce the lack of real security vulnerabilities databases. This methodology has proven itself as being very valuable since we were able to collect a considerable number of security vulnerabilities from a small group of repositories **238 from 63M**. However, there is still much work to do in order to improve not only in the mining tool but also in the evaluation and identification process which can be costly and time-consuming. As future work, we plan to extend the number of security vulnerabilities, patterns, and languages support. We will continue to study and collect patterns from GitHub repositories and possibly extend the study to other source code hosting websites (e.g., bitbucket, svn, etc). We will also explore natural processing languages, in order to introduce semantics and, hopefully, decrease the percentage of garbage associated with the mining process.