SATE V Workshop

CVE-selected Analysis Results

Bertrand STIVALET, NIST
stivalet [at] nist [dot] gov
March 14, 2014

The **SAMATE** Project
http://samate.nist.gov

Outline

- Test sets
- CVE*-based analysis
  - Procedure
  - Observations
- Subset warnings analysis
  - Procedure
  - Observations

*CVE: Common Vulnerabilities and Exposures

Test sets

- PHP language:

Software	Vulnerable version	Fixed version	Lines of code	Participating tools
Wordpress Blogging platform	2	2.2.3	< 25k	**1**

Certain instruments, software, materials, and organizations are identified in this paper to specify the exposition adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the instruments, software, or materials are necessarily the best available for the purpose.

Test sets

- C/C++ language:

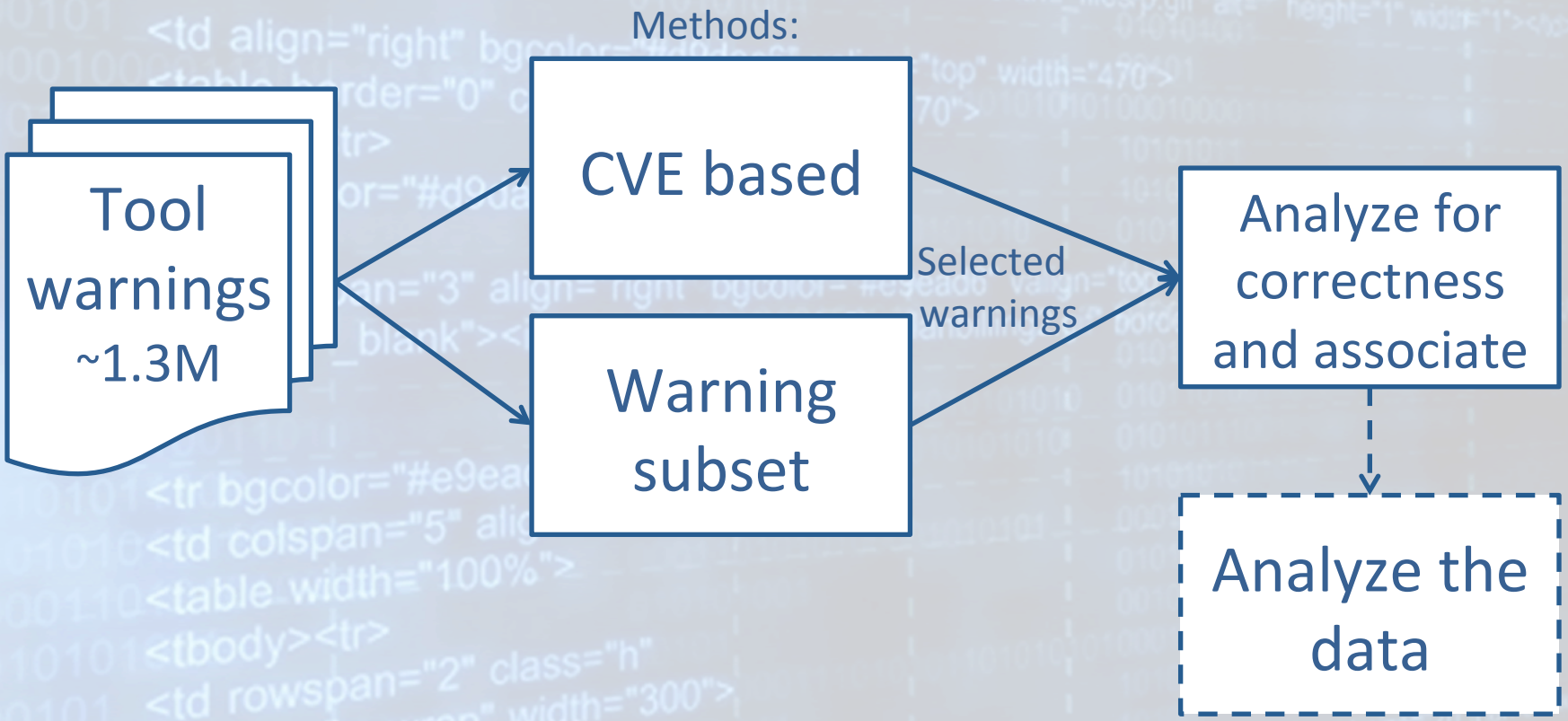
Software	Vulnerable version	Fixed version	Lines of code	Participating tools
Asterisk PBX Platform	10.2.0	10.12.2	> 500k	**8**
Wireshark Traffic Analyzer	1.8.0	1.8.7	> 2M	**9**

Test sets

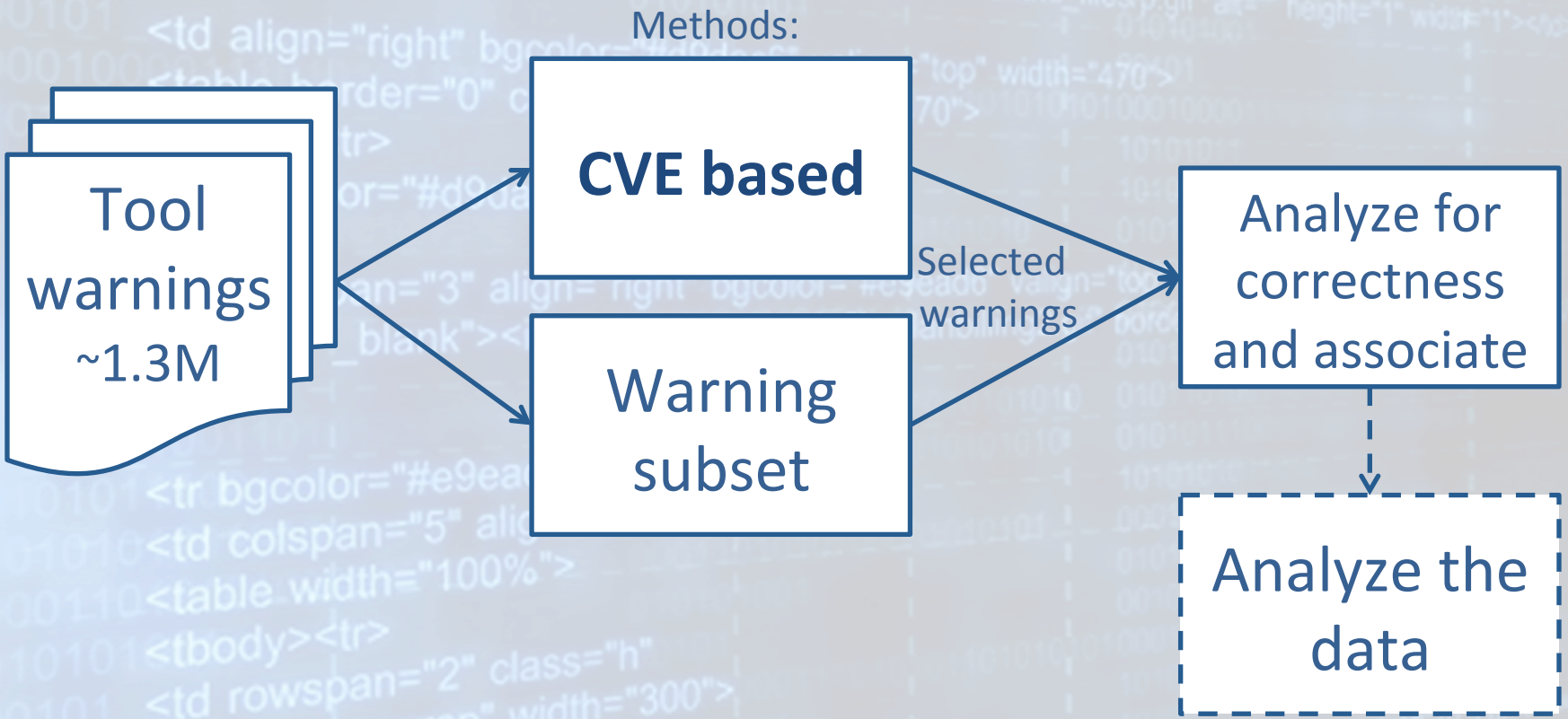
- JAVA language:

Software	Vulnerable version	Fixed version	Lines of code	Participating tools
JSPWiki Wiki	2.5.124	2.5.139	> 60k	**6**
Openfire Groupchat server	3.6.0	3.6.4	> 200k	**6**

Analysis procedure for CVE-selected test cases



Analysis procedure for CVE-selected test cases

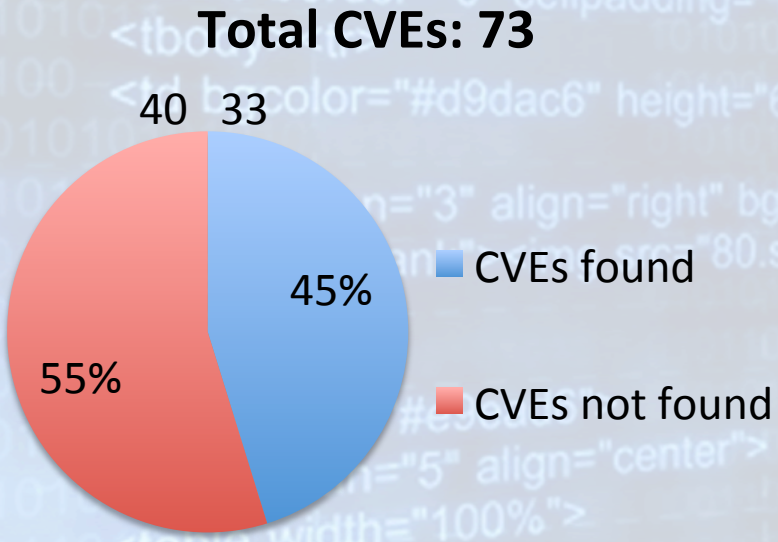


CVE-based analysis

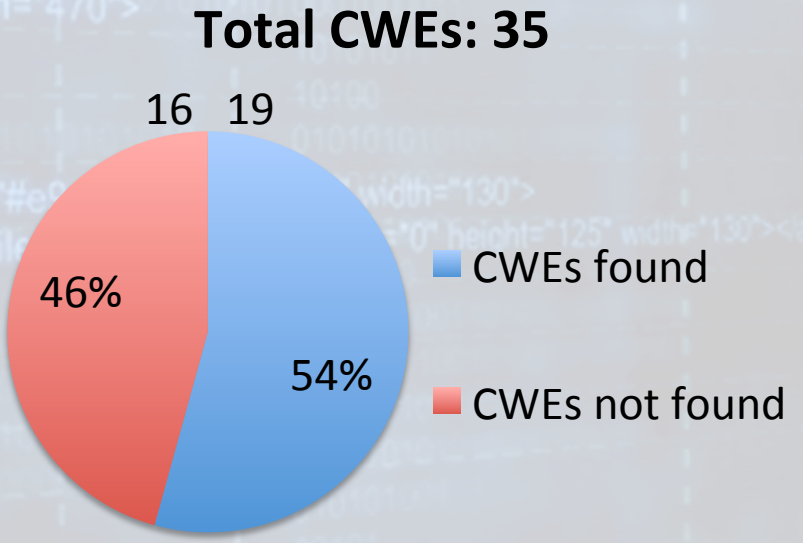
- CVE: Commun Vulnerabilities and Exposures
- Focused on real-life exploitable vulnerabilities
- 73 CVEs in the 5 test cases
  - Identify source, sink or path locations
  - Match to tool warnings

CVE-based observations

- **CVEs** found by tools:



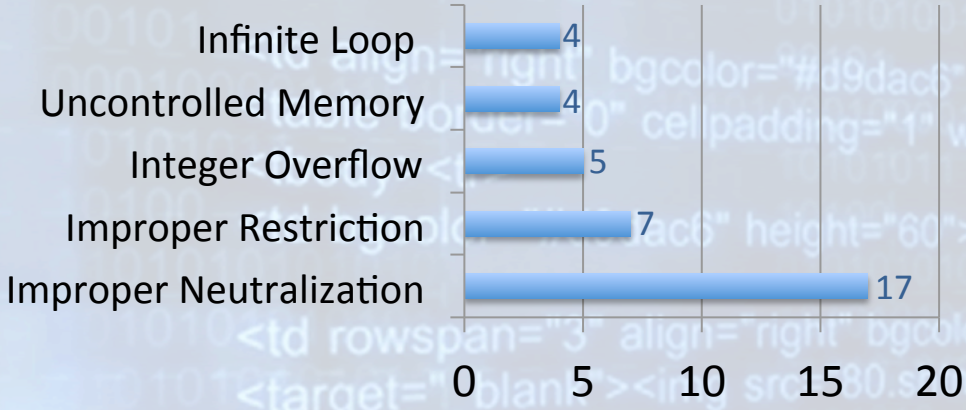
- **CWEs*** found by tools:



*CWE: Common Weakness Enumeration

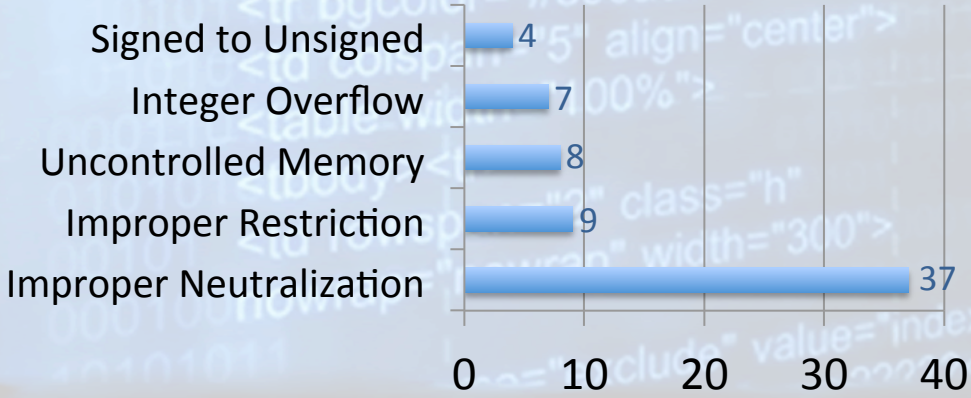
CVE-based observations

Top 5 CWEs present in CVEs



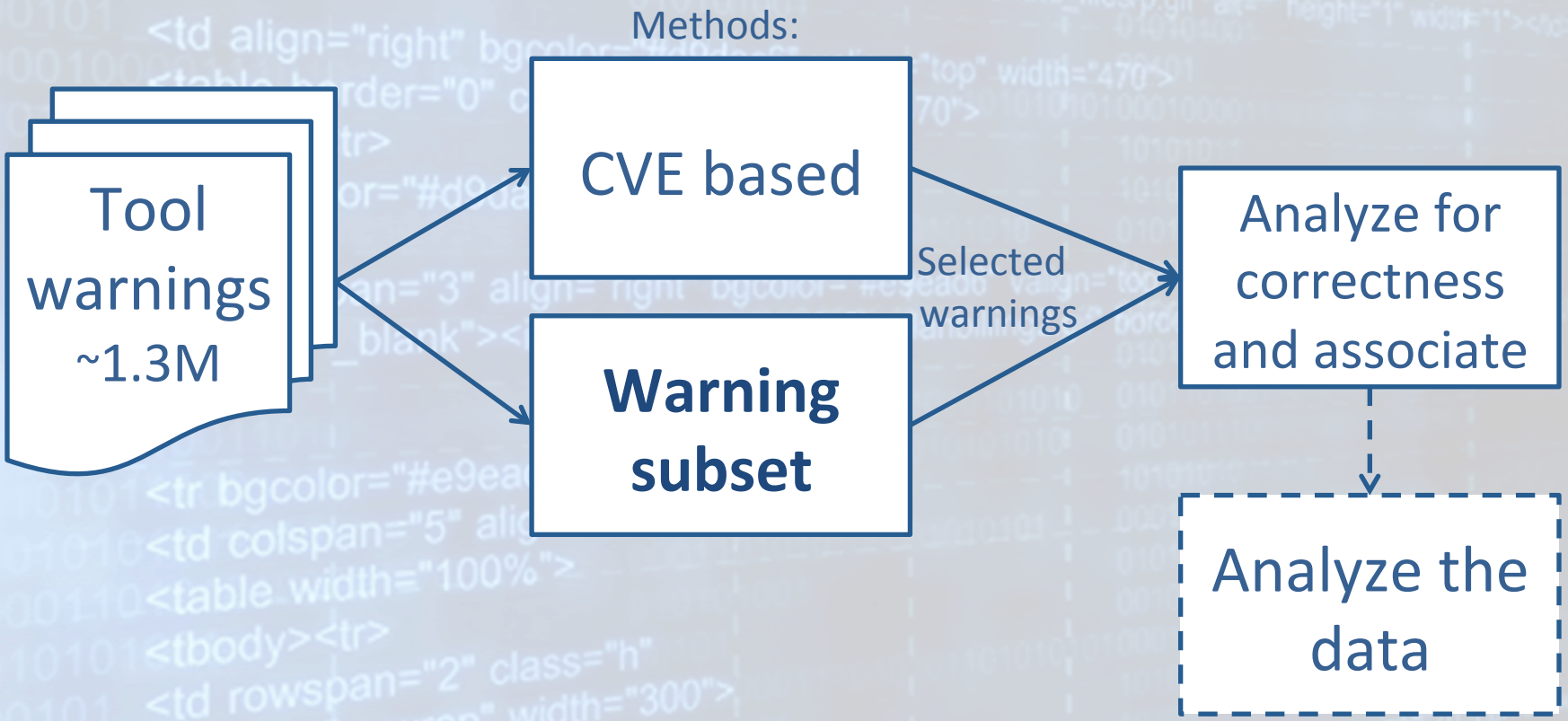
- Top 5 CWEs cover 37 of 73 CVEs

Top 5 CWEs found by tools



- "Infinite loop" weakness has only 1 hit.

Analysis procedure for CVE-selected test cases



Warning subset analysis

- Helps understand what weaknesses are found by tools in real-world software
- 30 warnings from each tool report statistically selected
  => **900 warnings**!
- Analyzers:
  - Aure
  - Bertrand
  - Charles
  - Kamillia
  - Paul
  - Sean (volunteer)
  - Vadim
  - Yan

Decision process

Correctness categories:

- True **security** weakness: a weakness relevant to security
- True **quality** weakness: poor code quality
- True but **insignificant** claim.
- Not a weakness – **false**: invalid conclusion about the code
- Weakness status unknown

Wordpress precision analysis

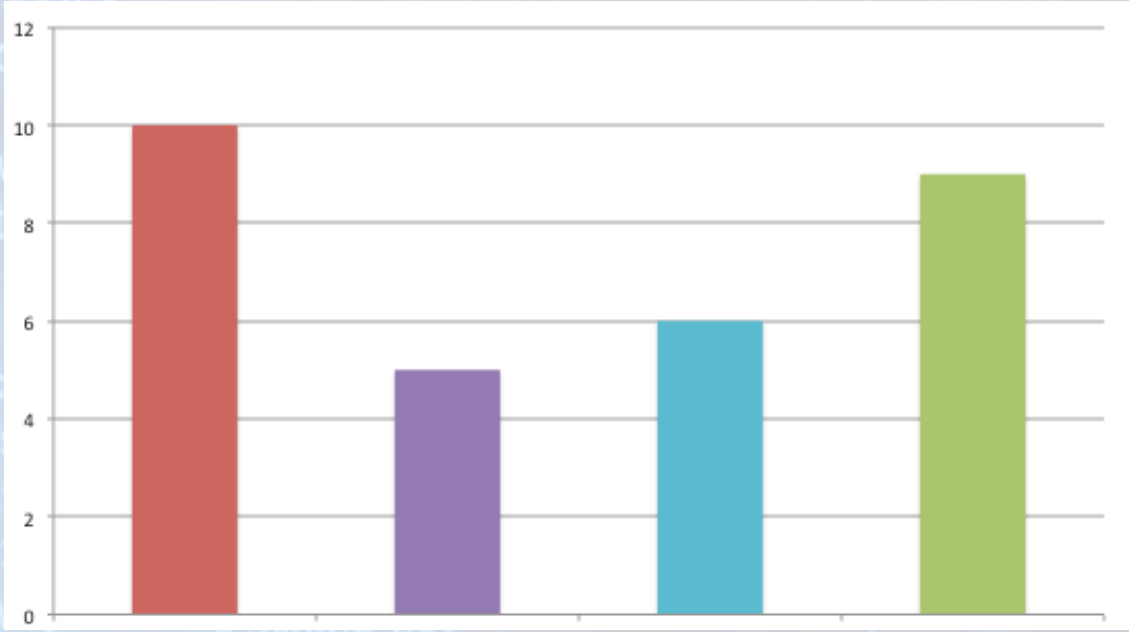
Precision = (#Useful Warnings) / (#All Warnings)

= (q + s) / (f + i + q + s)

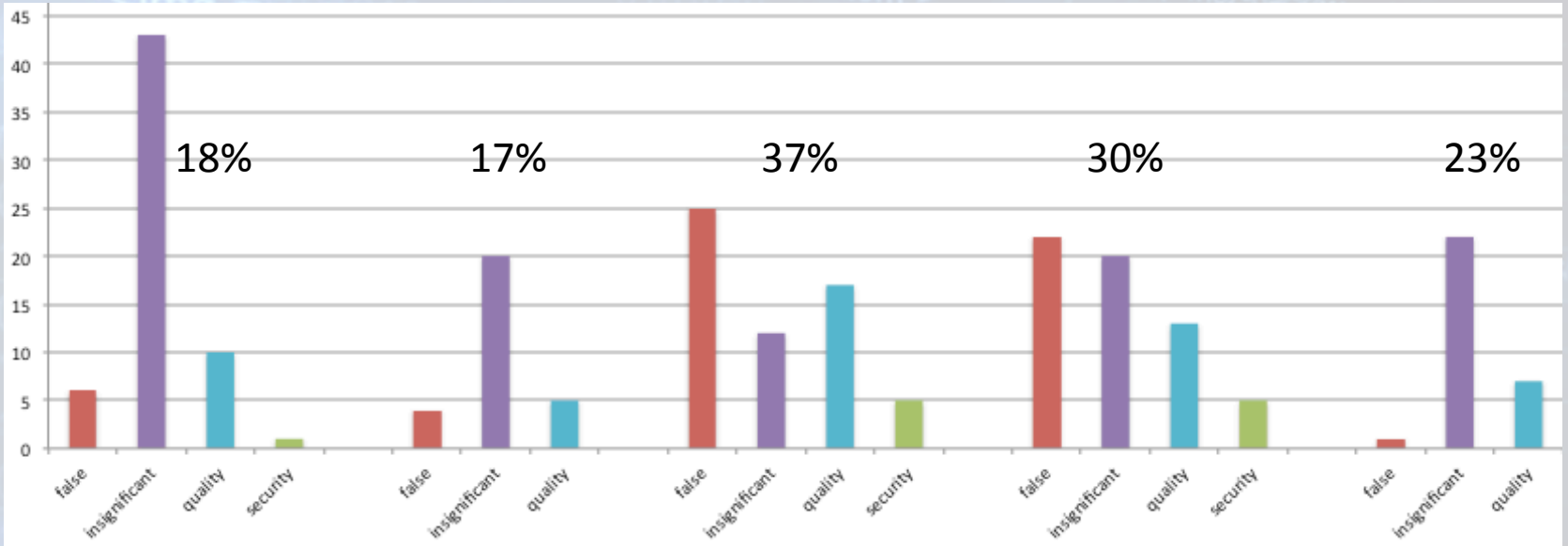
= (6 + 9) / (10 + 5 + 6 + 9)

= 50%

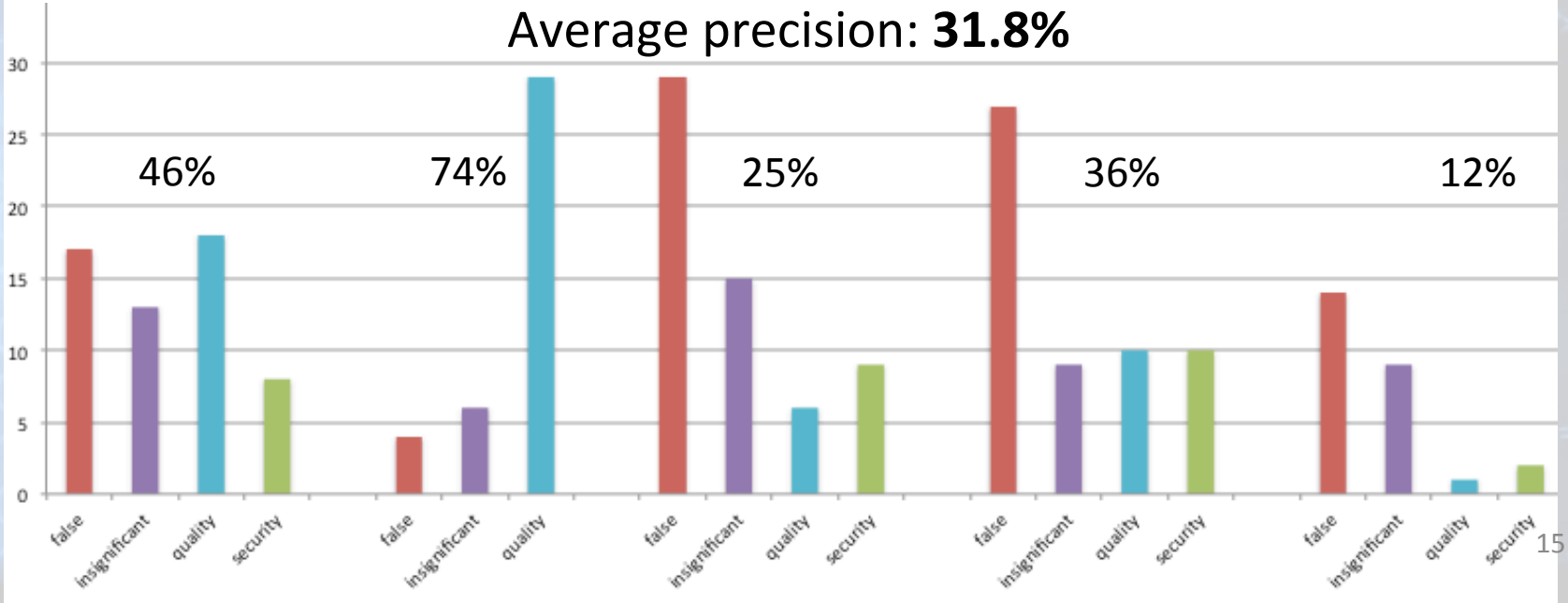
f: false
i: insignificant
q: quality
s: security



Asterisk + Wireshark precision analysis



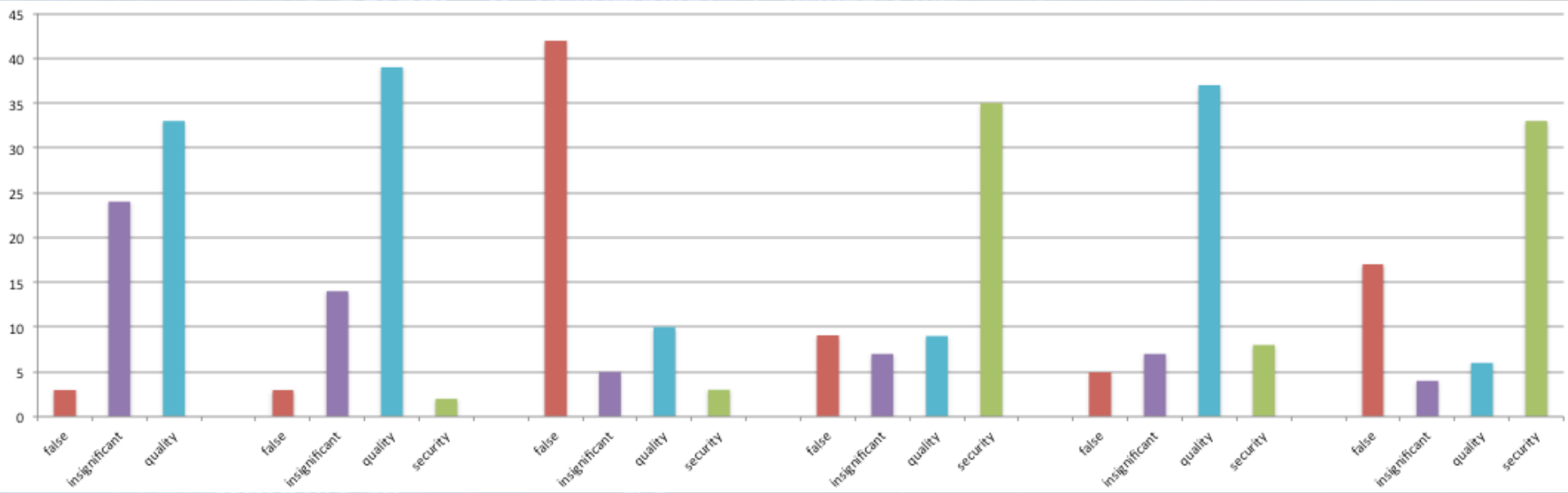
Average precision: **31.8%**



JSPWiki + Openfire precision analysis

Java Testcases

Average precision: **60.8%**



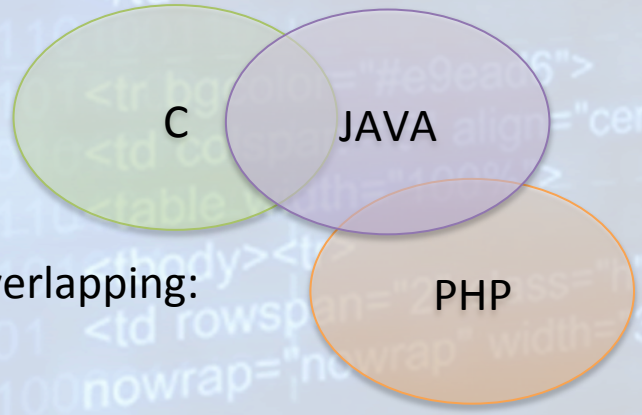
55% 71% 22% 73% 79% 65%

Top 5 CWE groups

- Top CWE groups reported in **security** and **quality** weaknesses

CWE GROUP - C	OCCURENCE
Buffer operation	82
Code quality	69
Denial of Service	46
Design and implementation	38
Memory leak	38

CWE GROUP - JAVA	OCCURENCE
Input validation	165
Code quality	146
API	109
Design and implementation	90
Concurrency	83

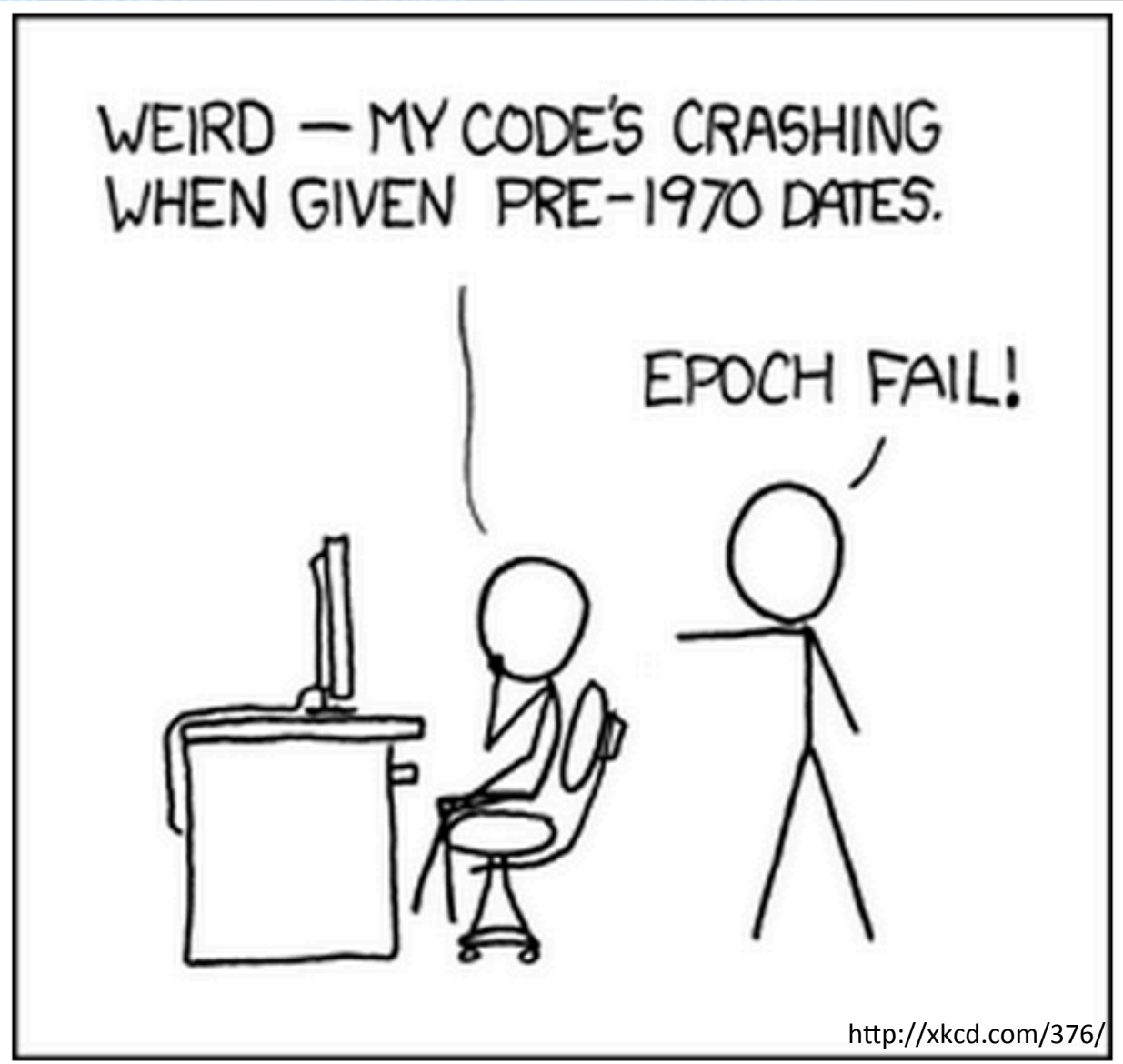


Overlapping:

CWE GROUP - PHP	OCCURENCE
Web	13
Input validation	6
Confidentiality	5
Dynamic code	3
API	2

Summary

- Finalize our results analysis
- Precision varies a lot by tool and weakness category
- Some CWEs are still difficult to find



http://xkcd.com/376/

Selection procedure

- We selected 30 warnings from each tool report (except one report, which had fewer than 30 warnings) using the following procedure. Here, a warning class is a (weakness name, severity), e.g., (Buffer Underrun, 1) pair. For one of the tools we could not use weakness names to identify warning classes, so we used a (CWE id, severity) pair instead.

- The procedure is similar to that used in previous SATEs. The main difference is that some tool reports had more than 30 warning classes with severities 1 through 4, so we chose a random subset of warning classes.

- While more warnings are needed, repeat for severity S (where S is from 1 through 4):
  -- Randomly select one warning from each warning class (or a randomly selected subset of warning classes if there are more warning classes left than warnings needed) identified by a warning name (or by CWE id) with severity S.

Selection procedure

- While more warnings are needed, repeat:

  -- Randomly select 3 of the remaining warnings (or all remaining warnings if there are less than 3 left) from each warning class with severity 1,

  -- Randomly select 2 of the remaining warnings (or all remaining warnings if there are less than 2 left) from each warning class with severity 2,

  -- Randomly select 1 of the remaining warnings from each warning class (if it still has any warnings left) with severity 3.

- If more warnings are still needed, select warnings from warning class with severity 4, then select warnings from warning class with severity 5.

If a tool did not assign severity, we assigned severity based on weakness names and our understanding of their relevance to security.

We excluded from the selection process the warnings that refer exclusively to test code, parser generator code, or external header files.