Carnegie Mellon University
Software Engineering Institute

DEV SEC OPS DAYS

Commonality and Trends in SAST Results

December, 2021

Dr. Chris Near
CyberSagacity Ltd.
www.cybersagacity.io

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Trends in SAST Results: Know Your Tools

Did you know……

- There is less than 1% overlap amongst the defects found by different AST tools
- 80% of context-determined severe defects are dismissed as unimportant by SASTs
- AST-found Java defects are 3 times more likely to have high confidence than C/C++ defects
- Only 5.5% of AST defects are easy to exploit
- Only 6.5% of AST defects are rated high severity by the AST

The Defect Database

10 SAST, 8 Open Source Projects

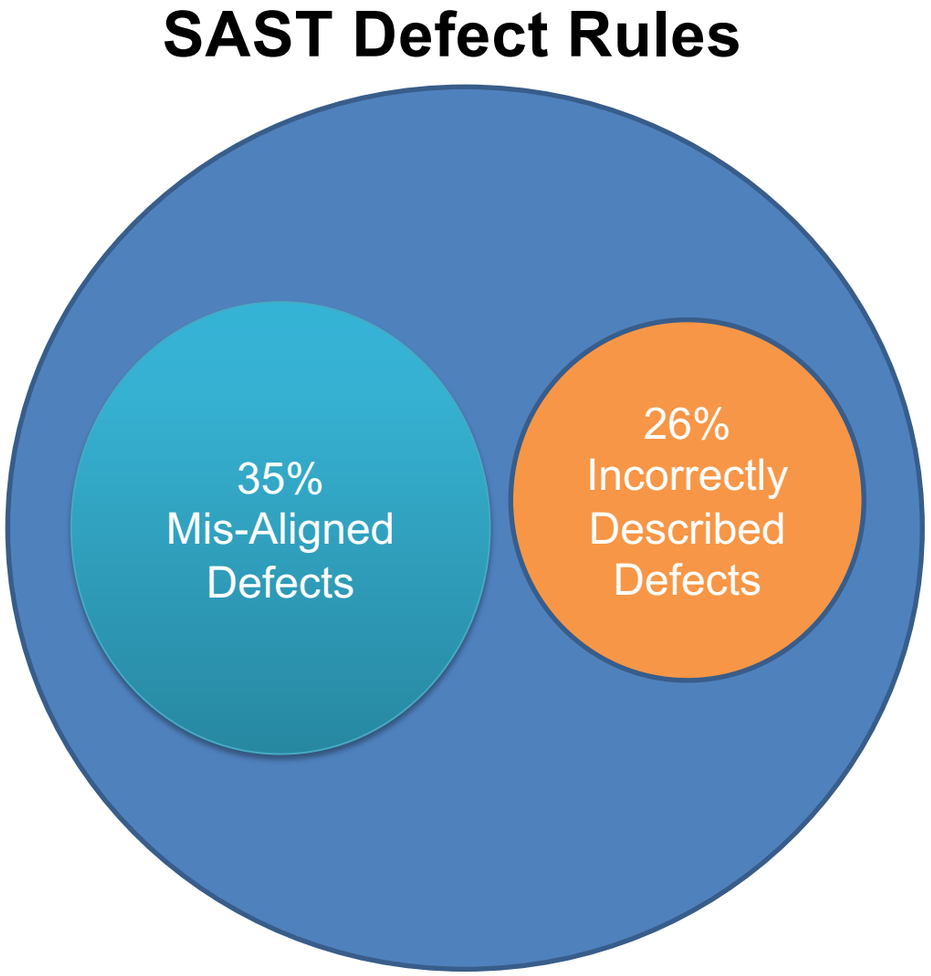
Project	Defect Density (Lines of Code/Defect)
A	6.8
B	7.2
C	14.9
D	1.3
E	8.1
F	32.0
G	37.6
H	24.0

- 7.7 MLOC: Java/Javascript and C/C++ code
- Mature: Well-used, well-maintained, dozens of release cycles, half were > 25 years old
- 684,816 Defects

Defect Normalization

10,102 Defects, 20 SAST, 10 Languages

- Re-alignment based on consistent & accurate mapping to MITRE CWE
- 11 commercial Tools
- Errors, laziness, keywords, effect-not-cause “expanded” coverage
- 9 open source tools: most do not map to CWE
- 415 CWE items covered



Defect Commonality

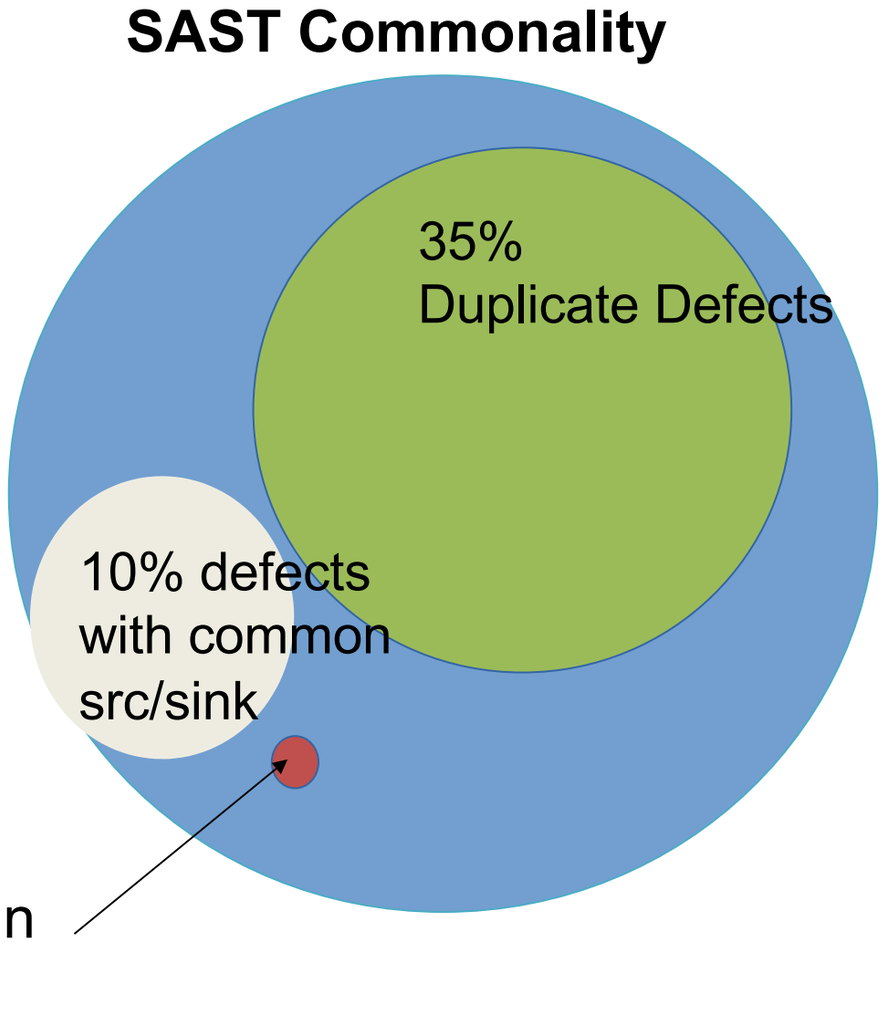
The Process

- Defect Normalization
- De-Duplication
- Common Sink-Source Location
- Parent-Child Relationships
- Cause-Effect Relationships

Defect Commonality

The Result: 6,705 common out of 684,816

- Common Source-Sink: 10%
- Parent-Child: 3x increase in commonality
- Cause-Effect: 12% increase in commonality



Results manually checked

False Positive Trends

The Method

	High	Medium	Low	No Confidence
Java 1	71.4	2.3	16.5	8.2
Java 2	60	1.9	20.8	16.4
Java 3	45.1	4.5	43.4	2.3
Java 4	76.1	0.9	18.5	2.8
C/C++ 1	6.4	13.4	59.5	19.3
C/C++ 2	9.7	6.4	77.4	4.9
C/C++ 3	23	12.3	53.3	9.2
C/C++ 4	16.4	6.5	62.7	13.4
Java Total	61.4	2.6	25.8	7.8
C Total	21.7	12.1	54.6	9.5
All Total	46.8	6.1	36.4	8.4

- Find relative probability that defect type is true positive
- Rule Specific e.g., narrowness of rule
- Analytic study: code properties that affect the math, path, and data value validity
- High confidence →>50% true positive
- No confidence →> 95% false positive

False Positive Trends

Java/Javascript is accurate;C/C++ is not

	High	Medium	Low	No Confidence
Java 1	71.4	2.3	16.5	8.2
Java 2	60	1.9	20.8	16.4
Java 3	45.1	4.5	43.4	2.3
Java 4	76.1	0.9	18.5	2.8
C/C++ 1	6.4	13.4	59.5	19.3
C/C++ 2	9.7	6.4	77.4	4.9
C/C++ 3	23	12.3	53.3	9.2
C/C++ 4	16.4	6.5	62.7	13.4
Java Total	61.4	2.6	25.8	7.8
C Total	21.7	12.1	54.6	9.5
All Total	46.8	6.1	36.4	8.4

- Java/JavaScript results are much more accurate than C/C++
- Java results are either highly confident or not – no middle ground
- Considerable quantities of no confidence results are present
- One SAST had 20-25% no confidence rates
- Results verified with historical defect trends

Ease of Exploit Trends

The Method

	Easy	Medium	Hard	No Confidence
Java 1	0.8	35.6	54	8.2
Java 2	0.6	9.5	72.6	16.4
Java 3	14.3	14.2	64.6	2.3
Java 4	0.7	7.7	87.3	2.8
C/C++ 1	5.3	22.1	51.9	19.3
C/C++ 2	10.7	2.5	80	4.9
C/C++ 3	6.8	13.2	68.5	9.2
C/C++ 4	10	9.5	66.2	13.4
Java Total	4.7	17.7	67.5	7.8
C Total	6.9	13.1	68.3	9.5
All Total	5.5	16.1	67.8	8.4

- Expand the CAPEC attack associations with CWEs
- Direct vs indirect attacks
- Easy defects mapped to easy-to-moderate CAPEC
- Half of defects have no associated CAPEC
- Verified with MITRE CVE Database

Ease of Exploit Trends

SAST focuses on hard-to-exploit defects

	Easy	Medium	Hard	No Confidence
Java 1	0.8	35.6	54	8.2
Java 2	0.6	9.5	72.6	16.4
Java 3	14.3	14.2	64.6	2.3
Java 4	0.7	7.7	87.3	2.8
C/C++ 1	5.3	22.1	51.9	19.3
C/C++ 2	10.7	2.5	80	4.9
C/C++ 3	6.8	13.2	68.5	9.2
C/C++ 4	10	9.5	66.2	13.4
Java Total	4.7	17.7	67.5	7.8
C Total	6.9	13.1	68.3	9.5
All Total	5.5	16.1	67.8	8.4

- Few defects are easy to exploit
- Most defects are difficult to exploit with no associated common attack patterns (CAPEC)
- One SAST was considerably better than all others in finding easy-to-exploit defects

Severity of Consequence Trends

The Method

	High	Medium	Low	No Confidence
Java 1	0.2	40	50.3	8.2
Java 2	0.2	13.8	68.6	16.4
Java 3	14.3	10.4	68.4	2.3
Java 4	0.3	5.9	89.4	2.8
C/C++ 1	5.1	20.1	54	19.3
C/C++ 2	11.4	30.8	51.3	4.9
C/C++ 3	10.4	21.9	56.2	9.2
C/C++ 4	13.4	5.0	67.2	13.4
Java Total	4.3	18.7	66.9	7.8
C Total	10.2	22.2	55.9	9.5
All Total	6.5	20.0	62.9	8.4

- Consequence list extension
- Analytic study of likelihood of each consequence for each defect
- Breachable vs. non-breachable defects
- Verified with MITRE CVE Database

Severity of Consequence Trends

More severe C/C++ defects than Java defects

	High	Medium	Low	No Confidence
Java 1	0.2	40	50.3	8.2
Java 2	0.2	13.8	68.6	16.4
Java 3	14.3	10.4	68.4	2.3
Java 4	0.3	5.9	89.4	2.8
C/C++ 1	5.1	20.1	54	19.3
C/C++ 2	11.4	30.8	51.3	4.9
C/C++ 3	10.4	21.9	56.2	9.2
C/C++ 4	13.4	5.0	67.2	13.4
Java Total	4.3	18.7	66.9	7.8
C Total	10.2	22.2	55.9	9.5
All Total	6.5	20.0	62.9	8.4

- Most defects have low probability of severe consequences
- C/C++ defects have higher severity than Java/Javascript
- SASTs determined 35% were severe
- Big difference between probability and possibility of defect presence

Trend Implications

Use lots of tools!

- To get adequate coverage, a project needs to use multiple tools - both commercial and open source
- Increased tool usage adds to triage burden, especially for C/C++ projects
- Ease of exploitation should be used as a prioritization parameter
- Determine probability, not possibility

What is next?

AST Management Tools

- CyberSagacity has an application to determine these trends for numerous ASTs
    - We combine AST trend factors in a very-narrowing triage process
    - We will apply AST trend analysis to defect rules for tool comparisons/trait analysis
- **OPPORTUNITY: Trial use of our trends/triage application. See www.cybersagacity.io for more information.**