Understanding Industry Perspectives of Static Application Security Testing (SAST) Evaluation

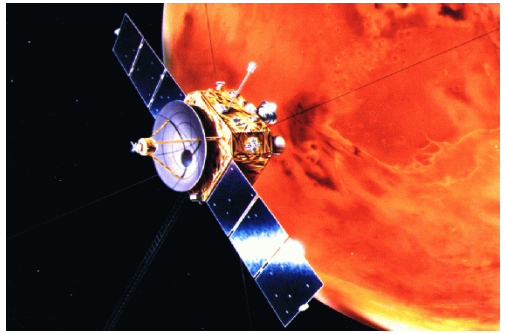
Yuan Li[1], Peisen Yao[1], Kan Yu[2], Chengpeng Wang[3], Yaoyang Ye[1], Song Li[1], Meng Luo[1], Yepang Liu[4], and Kui Ren[1]

Threats to Software Security

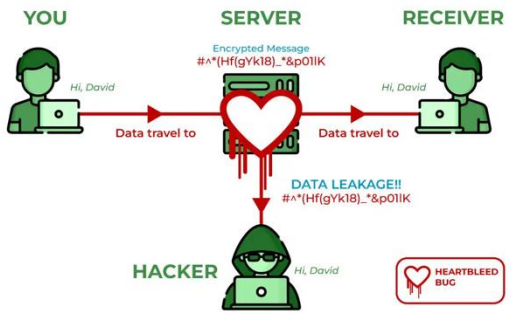
Log4Shell

Millennium

Data Race Problem



Time Control Error

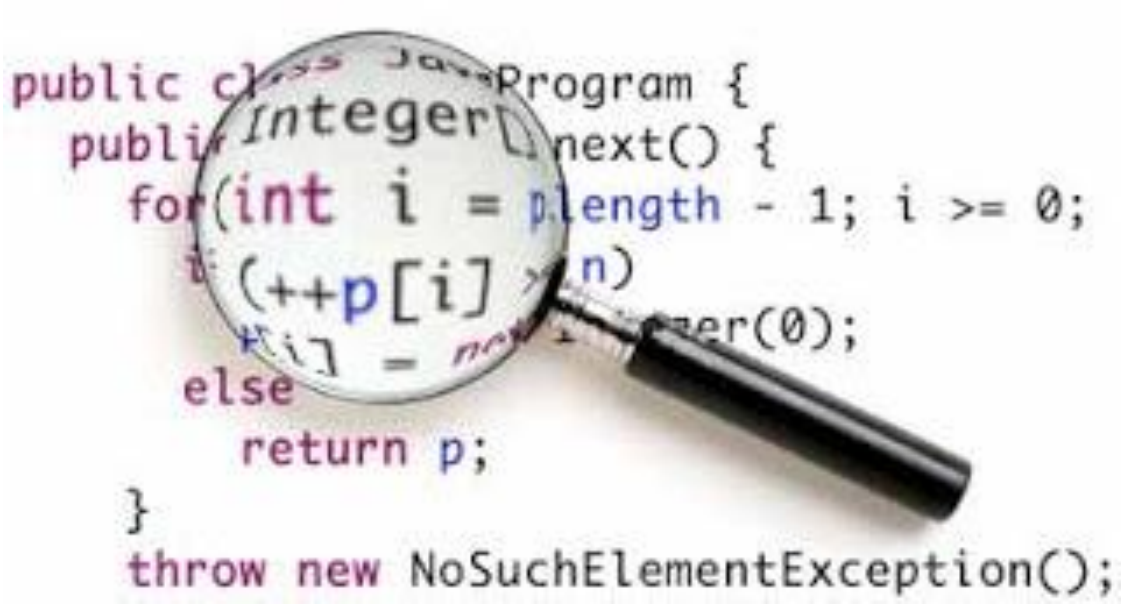


Heartbleed

Defects in Auto-stall System

Static Application Security Testing (SAST)

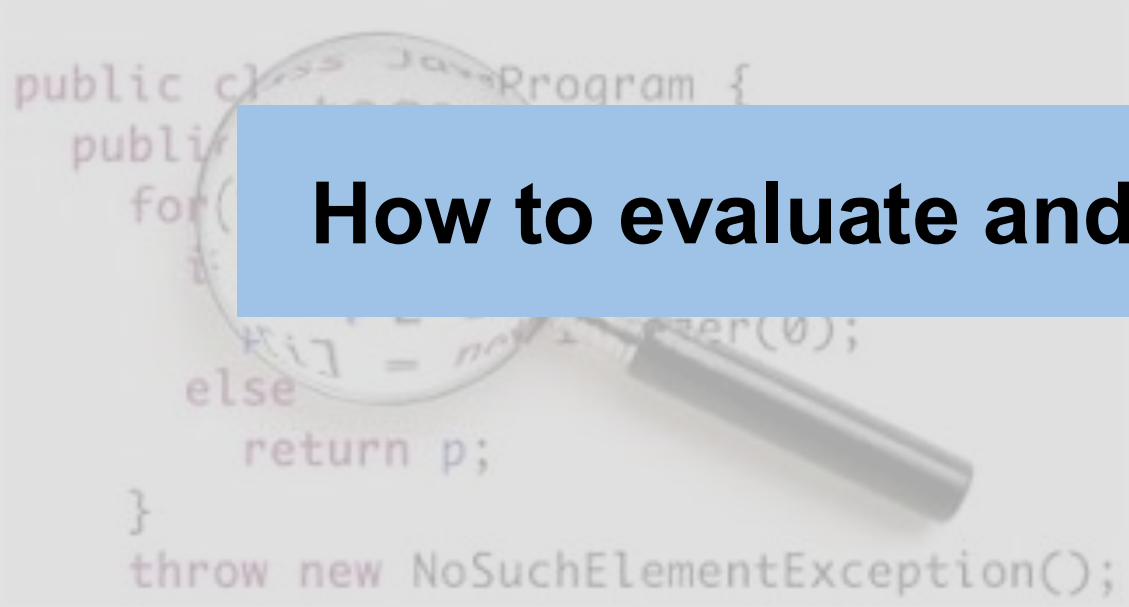
Statically analyze the source code without executing it



coverity · codeql · Checkmarx MAKE SHIFT HAPPEN · sonarqube · Contrast SECURITY · Semgrep · SpotBugs · Fortify by opentext · Sourcebrella PINPOINT · CoBOT a Beidasoft Company

Static Application Security Testing (SAST)

Statically analyze the source code without executing it



How to evaluate and select SAST tools?

coverity
codeql
Contrast SECURITY
SpotBugs
Fortify by opentext
Sourcebrella PINPOINT
CoBOT a Beidasoft Company

Evaluating SAST

Micro-benchmarks

Benchmark	Language	Size
OWASP	Java	2740
Juliet Test Suite	C++, Java	92980
SecuriBench-Micro	Java	96
PointerBench	Java	34
DroidBench	Java	190
PTABen	C/C++	400+

Real-world Benchmarks

Benchmark	Language	Size
DaCapo	Java	8
Defects4J	Java	17
TaintBench	Java	39
ManyBugs	C	9
BugsC++	C/C++	22
SecBench.js	JavaScript	19

Evaluating SAST

Micro-benchmarks	Real-world Benchmarks
Manually crafted or automatically generated	Derive from real-world programs or projects
Assess specific capabilities	✗
✗	Capture real-world complexity

There is still a lack of effective approaches to evaluate SASTs.

Evaluation Dilemma

Practitioners regard current benchmarks either too simple or biased, and tend to trust non-technical factors.

Ami et al., S&P'24

Practitioners find it hard to obtain accurate results from current benchmarks due to their diverse design standards.

Miltenberger et al., AsiaCCS'23

The results of current benchmarks look like "black-box", providing only overall recall rate and false positive rate data.

Expert D from Ant Group

Problem Statement

To bridge the gap between

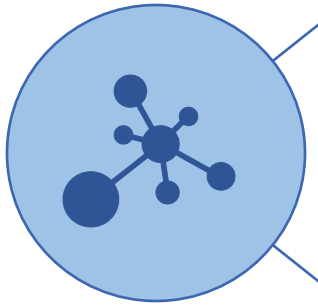
practitioners' expectations

and

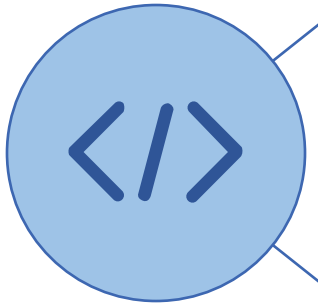
existing benchmarks

on SAST evaluation!

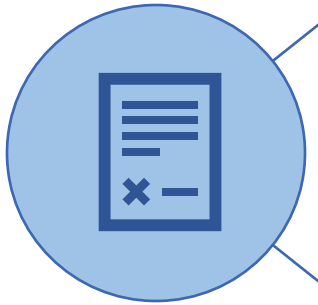
Research Questions



1. Why do practitioners use SAST benchmarks and what are their concerns about SAST **evaluation goals**?

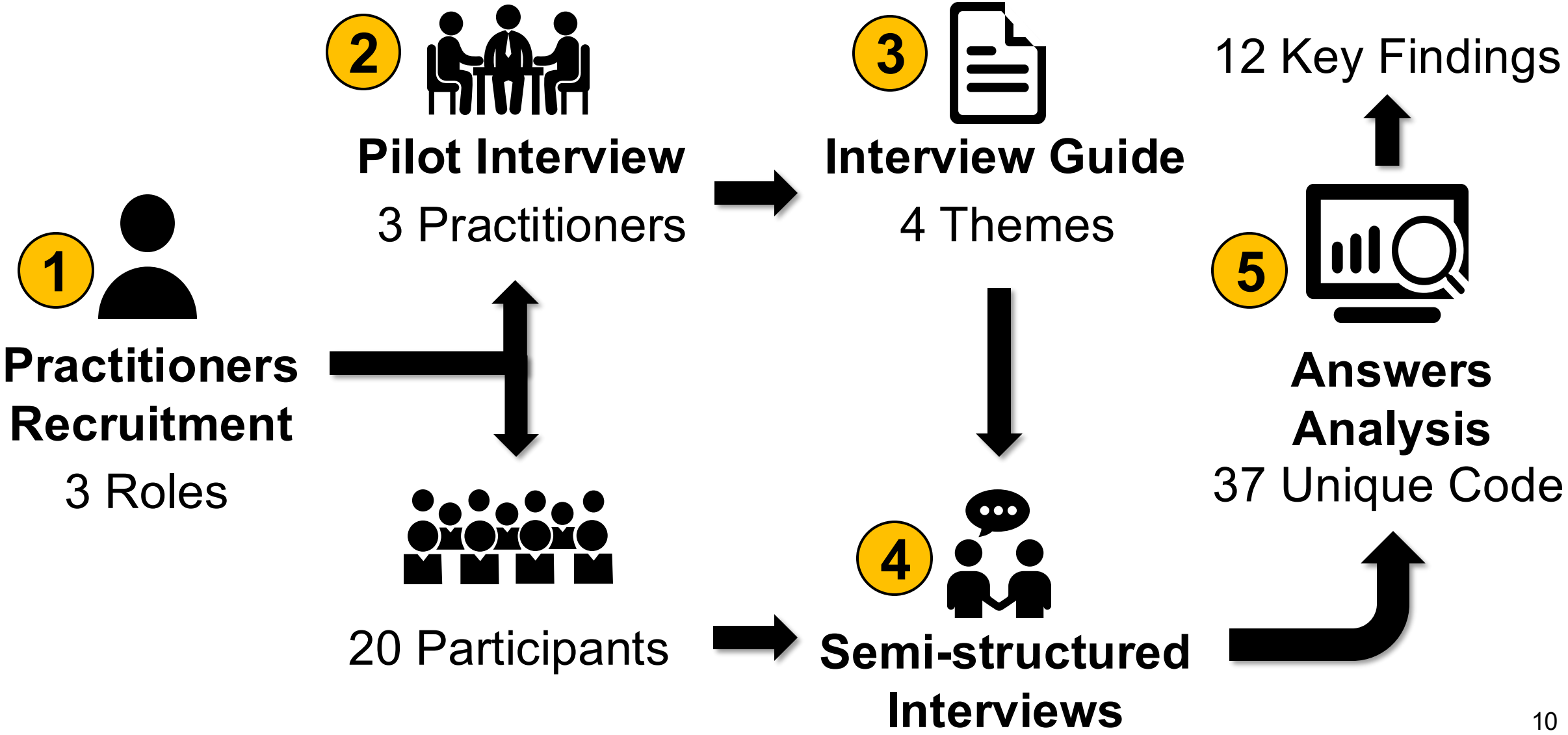


2. What **barriers** hinder the **adoption** of existing SAST benchmarks?



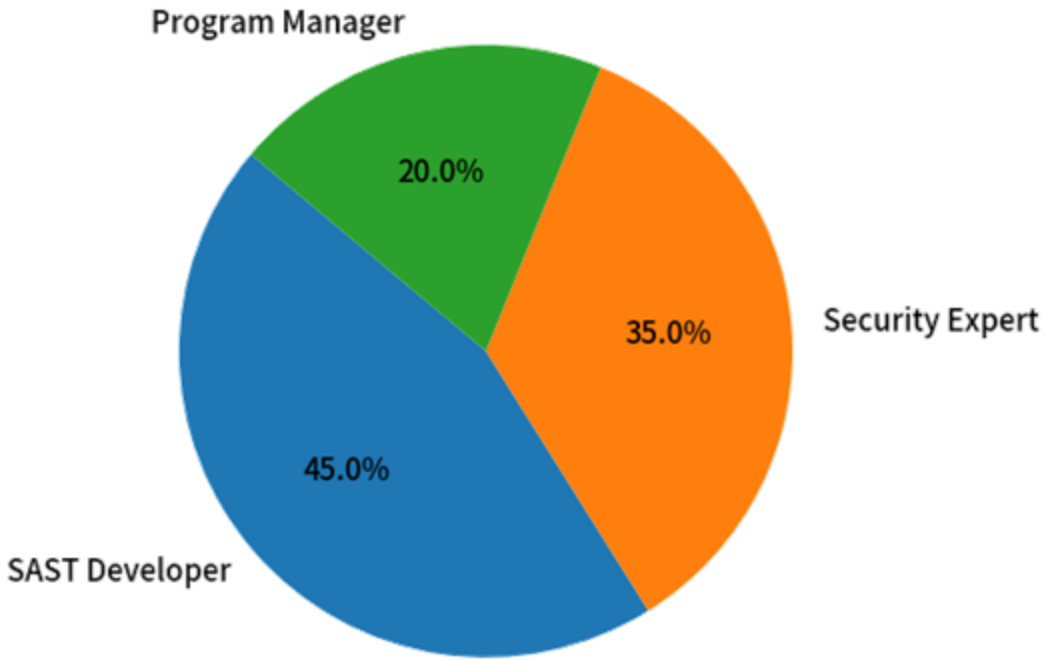
3. How can the **effectiveness** of SAST evaluation be **enhanced**?

Methodology

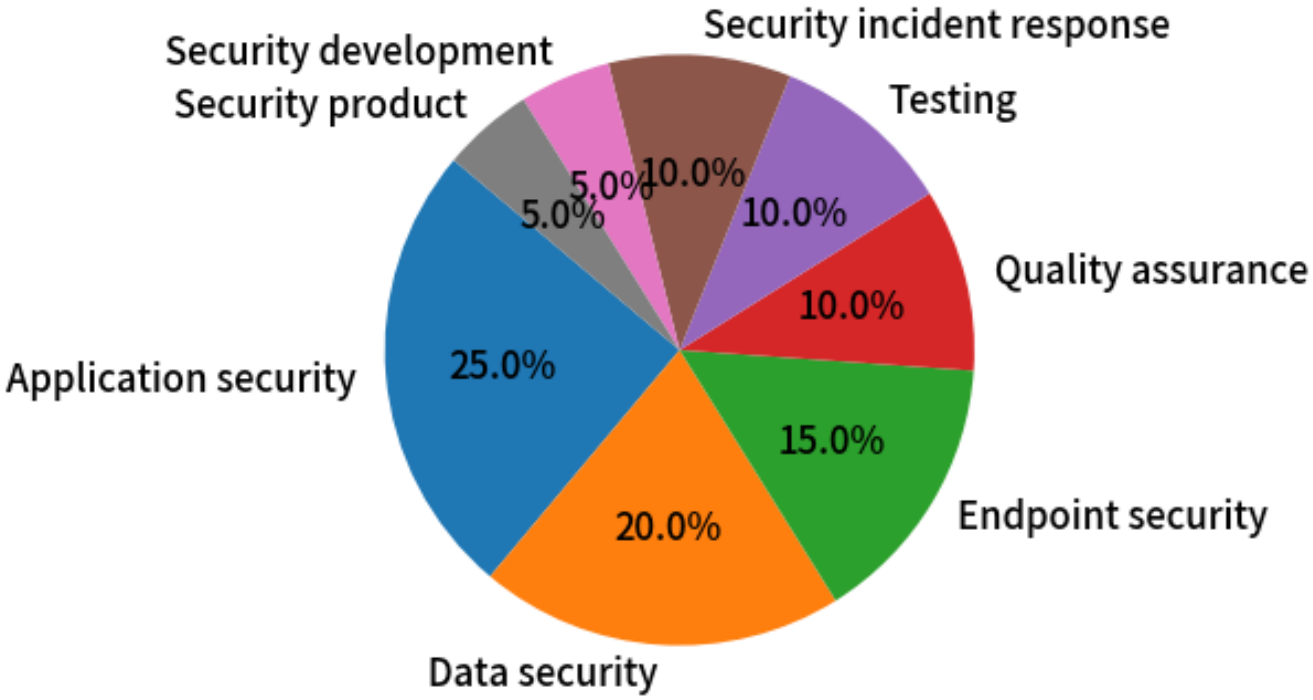


Participants

Role



Domain



蚂蚁集团
ANT GROUP

Interview Guide

Background Information of Participants

Reasons for Using SAST Benchmarks

Obstacles of Using SAST Benchmarks

Suggestions for Improving SAST Benchmarks

Findings and Implications

RQ 1: Role-specific Motivations	
Roles	Motivations
SAST Developer Program Manager Security Expert	Testing and Improvement Marketing and Compliance Comparison and Customization
RQ 2: Benchmark Limitations	
Limitations	Findings
Diagnostic Gaps	Fine-grained Labels and Fault Traces
Revealing Real-world Complexity	Intended Unsound Trade-offs and Deployment Complexities
Insufficient Customizability	Customization Tools and Community Collaboration
RQ 3: Actionable Implications	
Roles	Implications
Researcher	Identifying Intended Unsoundness and Measuring Deployment Robustness Reducing Real Programs and Evaluating Incremental Analysis
Benchmark Builder	Enhancing Quantitative Diversity and Prioritizing Customization Academia-Industry Collaboration and Industry-specific Collaboration

Findings and Implications

RQ 1: Role-specific Motivations	
Roles	Motivations
SAST Developer Program Manager Security Expert	Testing and Improvement Marketing and Compliance Comparison and Customization
RQ 2: Benchmark Limitations	
Limitations	Findings
Diagnostic Gaps	Fine-grained Labels and Fault Traces
Revealing Real-world Complexity	Intended Unsound Trade-offs and Deployment Complexities
Insufficient Customizability	Customization Tools and Community Collaboration
RQ 3: Actionable Implications	
Roles	Implications
Researcher	Identifying Intended Unsoundness and Measuring Deployment Robustness Reducing Real Programs and Evaluating Incremental Analysis
Benchmark Builder	Enhancing Quantitative Diversity and Prioritizing Customization Academia-Industry Collaboration and Industry-specific Collaboration

Why Use Benchmarks?

SAST Developers

"I used OWASP to improve Java container handling." (P6)

- ✓ Verify functionality
- ✓ Improve tool coverage

Program Managers

"The tool I promote must meet industry standards like CWE Top 25." (P12)

- ✓ Tool marketing
- ✓ Ensure compliance

Security Experts

"I rely on benchmarks to help me select the best tool I need." (P14)

- ✓ Compare tools
- ✓ Assess customizability

Findings and Implications

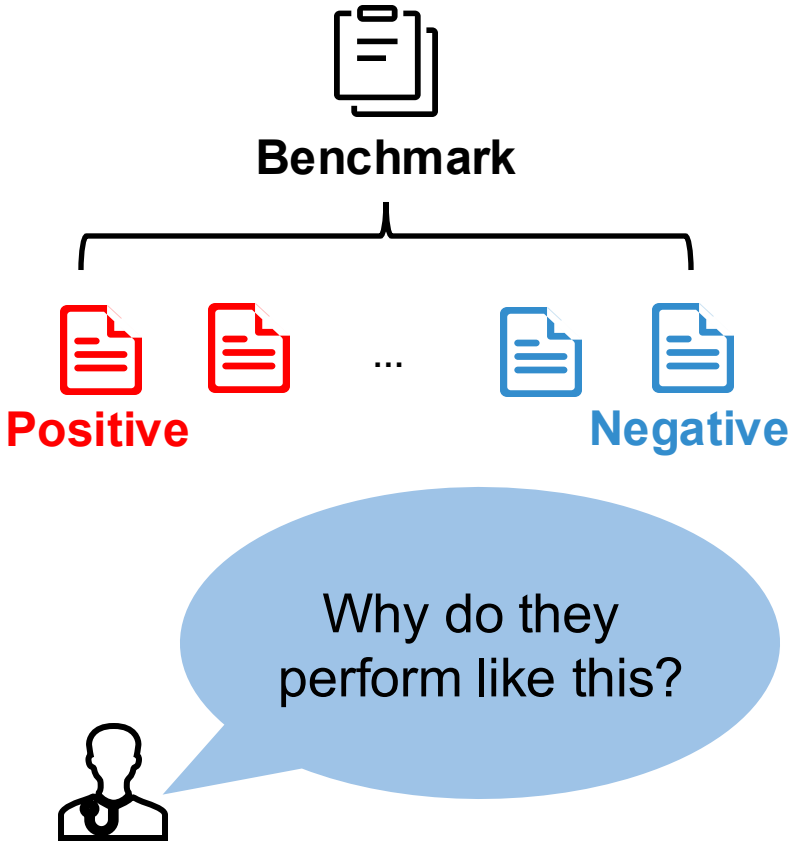
RQ 1: Role-specific Motivations	
Roles	Motivations
SAST Developer Program Manager Security Expert	Testing and Improvement Marketing and Compliance Comparison and Customization
RQ 2: Benchmark Limitations	
Limitations	Findings
Diagnostic Gaps	Fine-grained Labels and Fault Traces
Revealing Real-world Complexity	Intended Unsound Trade-offs and Deployment Complexities
Insufficient Customizability	Customization Tools and Community Collaboration
RQ 3: Actionable Implications	
Roles	Implications
Researcher	Identifying Intended Unsoundness and Measuring Deployment Robustness Reducing Real Programs and Evaluating Incremental Analysis
Benchmark Builder	Enhancing Quantitative Diversity and Prioritizing Customization Academia-Industry Collaboration and Industry-specific Collaboration

Problems

Solutions

Diagnostic Gaps

Tow-sided cases are too simple to explain the results.



The information they need to diagnose with:

- Fine-grained features
  - Data structure, sensitivity level
- Fault interpretation references
  - Bug trace, PoC

"I find it hard to identify and reproduce the bugs without any PoC Support." (P16)

Enhancing Quantitative Diversity

Many benchmarks rely on simple **syntactic** metrics.

⇩

Semantic complexity evaluation contributes to diagnosis.

⇩

"A priori" approach: Control semantic features during test program generation

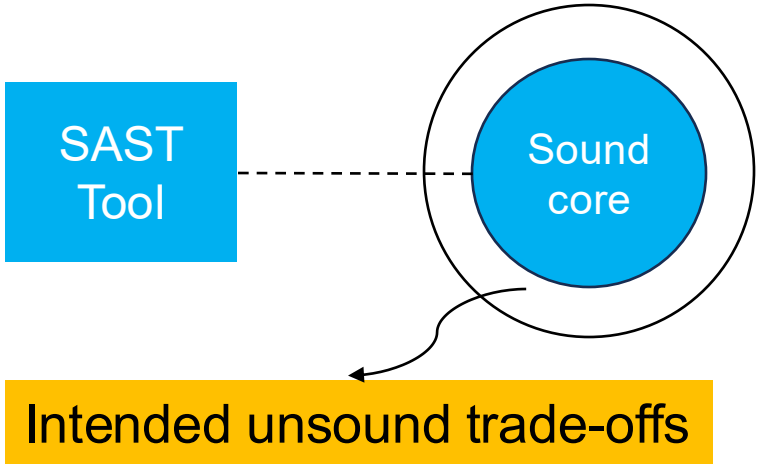
Metric Type	Example	Limitations
Syntactic	Lines of code, Function count	Cannot measure analysis difficulty
Semantic	Calling context scale, Loop nesting	Requires pointer analysis, biased

Solution!

"We can control n when considering a taint-style bug where the sink hides behind the n-th iteration of a loop." (P1)

Revealing Real-World Complexity

Benchmarks do not perform well in reflecting realistic intricacies.



Unsoundness	CSA	Infer
Loop unroll times	√	
Call depth	√	√
Callback level		√

GNU Operating System

Bazel

Maven™

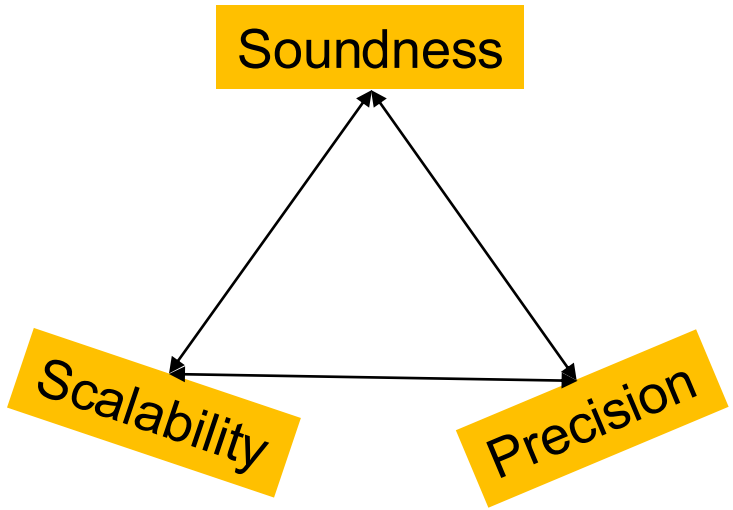
LLVM

"We need to evaluate the capability of diverse deployment contexts, which is essential but missing widely." (P13)

Identifying Intended Unsoundness

Intended unsoundness is prevalent in industry-strength SAST tools.

Source	Description
Limited loop unrolling	Bugs in n-th iteration may be missed
Bounded pointer analysis	Fixed time/memory budget analysis
Priority-driven analysis	Focus on methods like to generate taint



- ✓ Improves performance
- ✓ Faster but potentially incomplete
- ✓ Better scalability

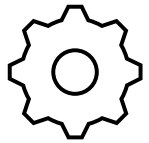
⇨

"I would like to have benchmarks that can reveal 'tricky' trade-offs so that I can understand and may adjust the heuristics." (P13)

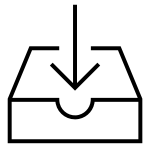
Measuring Deployment Robustness

Benchmarks must assess SASTs' ability to handle diverse contexts.

Build System



Compiler

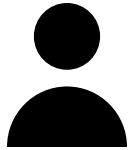


Dependency Configuration

"Analyzing Maven-built projects succeeds, how about compatibility with alternatives?" (P4)	"Benchmarks lack comprehensive coverage of compiler/IR compatibility." (P5)	"Benchmarks must evaluate robustness in diverse dependency scenarios." (P7)
✓ GNU Make, Ant ✓ Bazel, Maven	✓ Clang vs IR	✓ Third-party libs ✓ Environment variables

Insufficient Customizability

There is few flexible tools or flatforms to assist customization.

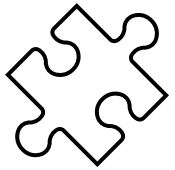


"I want to customize the benchmark and update it, but no tools have been provided." (P17)

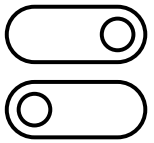
"I have the trouble to customize the benchmark, but online solutions are mostly either close-source or outdated." (P18)

Benchmark is there but not suitable!

Customization and Collaboration



Subsets



Variants

Contribution

"OWASP lacks customization to run only relevant cases (e.g., taint flows)." (P18)

"Mutating benchmarks ensures tools adapt to new OS versions." (P5)

"Practitioners rely on community forums for benchmark troubleshooting." (P20)

✓ Feature-specific

✓ Case templates

✓ Forums, conference

✓ Honors, awards

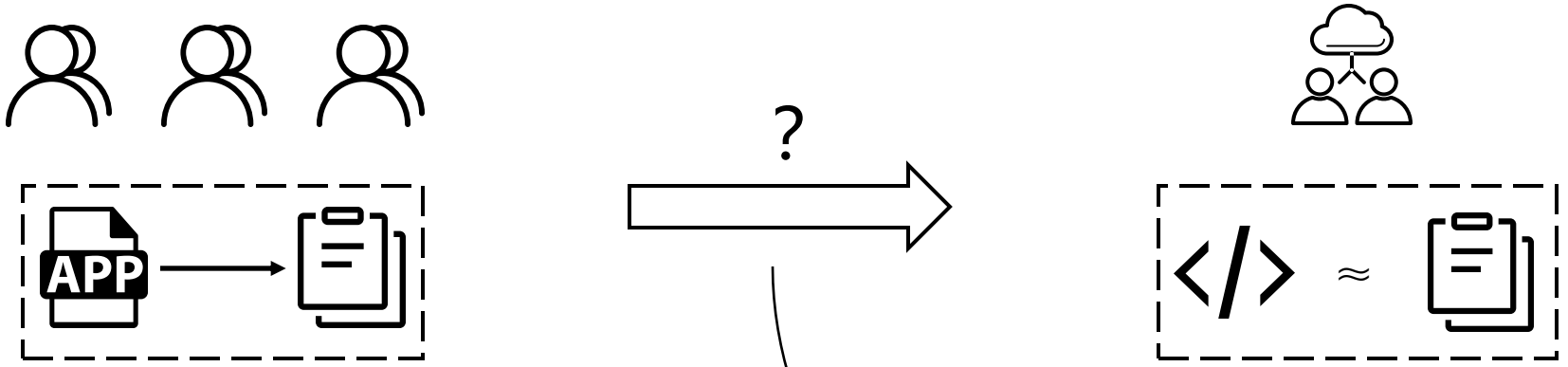
Tools/Plug-ins

Findings and Implications

RQ 1: Role-specific Motivations	
Roles	Motivations
SAST Developer Program Manager Security Expert	Testing and Improvement Marketing and Compliance Comparison and Customization
RQ 2: Benchmark Limitations	
Limitations	Findings
Diagnostic Gaps	Fine-grained Labels and Fault Traces
Revealing Real-world Complexity	Intended Unsound Trade-offs and Deployment Complexities
Insufficient Customizability	Customization Tools and Community Collaboration
RQ 3: Actionable Implications	
Roles	Implications
Researcher	Identifying Intended Unsoundness and Measuring Deployment Robustness Reducing Real Programs and Evaluating Incremental Analysis
Benchmark Builder	Enhancing Quantitative Diversity and Prioritizing Customization Academia-Industry Collaboration and Industry-specific Collaboration

Reducing Real Programs

It is hard to reduce large programs while preserving root causes.



Input: large, “real” program

Output: small, “same” program

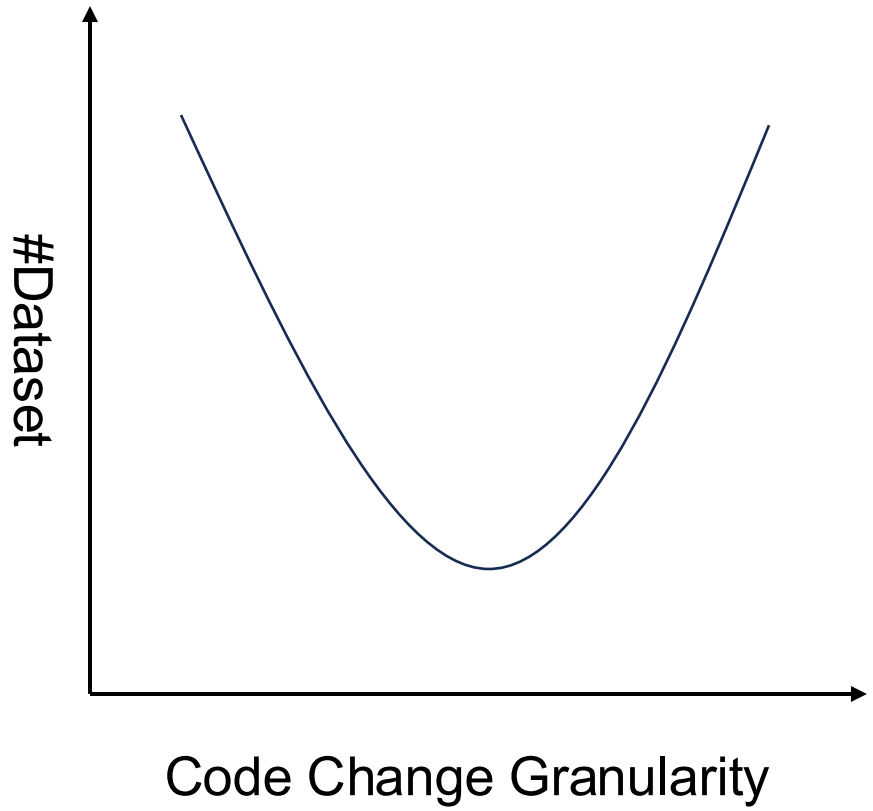
“Minimal proof-of-concept programs enable quick algorithm/SAST validation.” (P13)

Delta debugging ✗

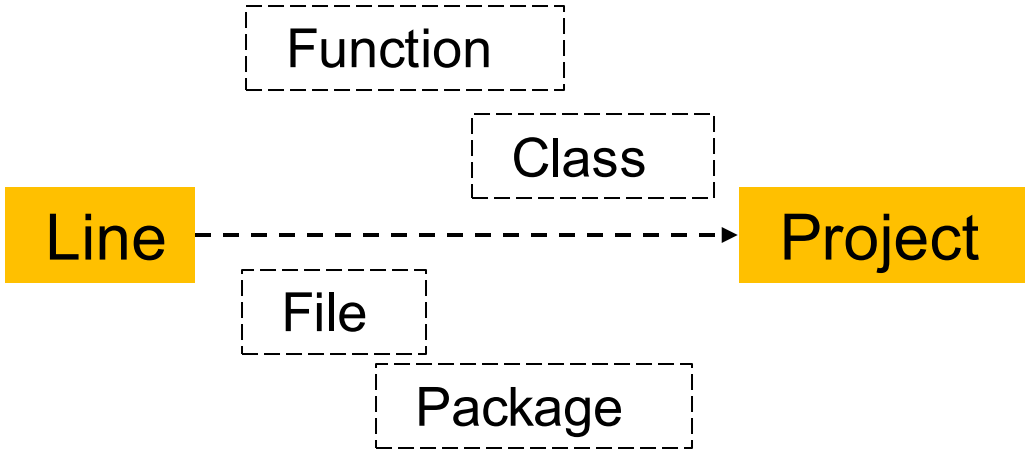
\+

SAST-specific outputs ✓

Evaluating Incremental Analysis



"IDE real-time analysis requires incremental checks for multi-line, function-level, or even sub-project changes." (P13)



Findings and Implications

RQ 1: Role-specific Motivations	
Roles	Motivations
SAST Developer Program Manager Security Expert	Testing and Improvement Marketing and Compliance Comparison and Customization
RQ 2: Benchmark Limitations	
Limitations	Findings
Diagnostic Gaps	Fine-grained Labels and Fault Traces
Revealing Real-world Complexity	Intended Unsound Trade-offs and Deployment Complexities
Insufficient Customizability	Customization Tools and Community Collaboration
RQ 3: Actionable Implications	
Roles	Implications
Researcher	Identifying Intended Unsoundness and Measuring Deployment Robustness Reducing Real Programs and Evaluating Incremental Analysis
Benchmark Builder	Enhancing Quantitative Diversity and Prioritizing Customization Academia-Industry Collaboration and Industry-specific Collaboration

Conclusion

1. Conduct **qualitative study** on industrial SAST evaluation
2. Provide insights into **practitioners' perceptions**
3. Uncover **deficiencies** in current practice
4. Reveal **directions** for further evaluation

Thank you!

Our Paper

xAST Benchmark

蚂蚁集团
ANT GROUP