THE SECURITY TOOLS GAP

ACADEMIC EVIDENCE VS. VENDOR CLAIMS

Axeinos | March 2025

FRAMING THE PROBLEM

Security tool vendors promise comprehensive protection and vulnerability detection. Yet, **independent academic research tells a different story** — one of overstated detection rates, hidden false positives, and underestimated implementation complexity.

This white paper presents a critical analysis of the gap between vendor marketing claims and real-world tool performance, drawing on peer-reviewed studies from leading universities and research institutions.

We examine five critical areas where academic research directly contradicts common vendor claims:

- **Benchmark manipulation** and artificial testing environments
- **Actual detection rates** for static analysis tools
- **False positive rates** and their operational impact
- **Coverage limitations** in black-box fuzzing approaches
- **Integration challenges** in a continuous testing environment

This analysis provides security leaders with **objective evidence** to evaluate vendor claims and **shape better-informed security testing strategies**.

INDUSTRY CLAIMS
THE SECURITY TOOLS MARKETING NARRATIVE

Security tool vendors make strong claims about their products' ability to secure modern software environments. These claims often include:

- **High detection accuracy**:
"Find and fix vulnerabilities with real-time feedback and reduce flaws by up to 60% with IDE scans." — Veracode™
- **Minimal false positives**:
"*Checkmarx SAST combines both speed and security to improve developer experience – up to 90% faster with 80% lower false positives."* — Checkmarx™
- **Comprehensive vulnerability coverage**:
“*Our SAST technology identifies hundreds of different types of security issues that are meaningful and relevant—all during development.*” — SonarQube™
- **Maximized code coverage with fuzzing**:
“BeSTORM™ performs across all communication standards (even complex standards, such as SIP), and levels, including network, protocol, file, hardware, DLL and API. BeSTORM delivers an exhaustive search of all possible input combinations to test input implementation for weaknesses.” - Fortra®
- **CI/CD ready**:
“with Black Duck, you can integrate SAST seamlessly into your development and DevOps workflows and toolchains.” — Black Duck Software, Inc™ on Coverity™

However, **independent academic research consistently contradicts these narratives**, revealing substantial gaps between what vendors claim and how tools perform in real-world environments.

ACADEMIC FINDINGS

Multiple academic studies have identified systematic issues with how bench-marking environments are constructed:

- **Artificial Test Cases**:
Synthetic benchmarks, like those in the OWASP Benchmark Project, lack real-world complexity, risking tool overfitting and poor performance on diverse, real-world codebases [1][11].
- **Controlled Variables**:
Benchmarks intentionally exclude realistic factors such as third-party dependencies and complex architectures, resulting in evaluations that dot not reflect true tool effectiveness in practical scenarios [2][4].
- **Selection Bias**:
Benchmarks do not accurately reflect real-world vulnerability distributions. This leads to inflated detection rates for specific vulnerability types and unrealistic assessments of tool performance [11] [12].

REAL-WORLD IMPLICATIONS

These artificial benchmarks create a **fundamental disconnect** between claimed performance and real-world tool effectiveness:

- Detection rates **drop drastically** when applied to complex commercial applications [4][9].
- Effectiveness **decreases significantly** when tested against previously unknown vulnerabilities [5].

STATIC ANALYSIS LIMITATIONS

The most comprehensive study on static analyzer effectiveness, published at ISSTA 2022, found:

- Static analyzers for C/C++ **miss 47%–80%** of real vulnerabilities[1], **87%** for Java[2].
- Combined tools still **leave 30% to 70%** of vulnerabilities undetected[1][2].
- Commercial tools offer minimal to no advantage over open-source alternatives despite higher costs [5][9].

These findings align with earlier research reporting a **detection rates of only 0-21% to 21–49%** for commercial tools when tested against *known* vulnerabilities[3][4].

VULNERABILITY CLASS VARIATIONS

Detection effectiveness varies greatly by vulnerability type:

- Up to 70% detection of incorrect calculations but **less than 20% for improper I/O neutralization** related vulnerabilities[1].
- To date, no systematic academic evaluation measures tool performance against **emerging vulnerability classes** — a gap that warrants further investigation.

[1] [4] ISSTA 2022, P. 7.
[2] [9] LI ET AL., ESEC/FSE 2023, FIDING 6, P. 8.
[3] [5] GOSEVA-POPSTOJANOVA AND PERHINSCHI, 2015, TABLE 3, P. 28.
[4] [12] M. DELAITRE ET AL., NIST SATE V, 2018, TABLE 21, P. 25.

ACADEMIC MEASUREMENT

False positives represent one of the most significant operational challenges with static application security testing (SAST):

- In 2018, a NIST report documented **false positive rates ranging from 25 to 80%+**[1] for leading commercial static analyzers.
- Combining multiple analyzers to improve detection increased false positives by an **additional 15%**[2] **to 60%**[3], creating diminishing returns.
- High false positive rates are the **main reason developers abandon these tools** [6].

OPERATIONAL IMPACT

The real cost of false positives goes beyond triage and resolution:

- With false positive rates often **far exceeding the recommended 20%** [13], static analysis tools place a significant triage burden on developers and AppSec teams [6].
- Teams develop alert fatigue, **ignoring legitimate findings** [6].

This burden **diverts engineering capacity** from development and real security work and creates a false sense of coverage.

CLAIMS IN SAST ADVANCEMENT

Some vendors claim to offer a new generation of SAST tools with significantly lower false positive rates. However, no rigorous independent evaluation exists to date.

[1] [12] NIST 2018, TABLE 14, P. 30.
[2] [6] ISSTA 2022, SECTION 5.2.
[3] [9] LI ET AL., ESEC/FSE 2023, FINDING 6, P. 8.

COVERAGE CHALLENGES IN PRODUCTION APPLICATIONS

While fuzzing is effective in specific contexts, research highlights limitations in black-box fuzzing:

- Coverage **plateaus early**, leaving large portions untested [6][13][15].
- Coverage **degrades as complexity increases** [9].

TEST CASES GENERATION EFFECTIVENESS

The ability of black-box fuzzers to generate effective test cases varies significantly based on application complexity and input design:

- For complex input validation, fuzzers **rarely achieve significant code coverage** beyond input layers [2].
- Industry-specific protocols and binary formats reduce coverage further [9].
- **State-dependent vulnerabilities are routinely missed** [2][9].

INDEPENDENT EVALUATION

While the technical limitations of black-box fuzzers are well-documented, there is a **lack of recent independent academic assessments of commercial tools**. ProFuzzBench [11] and Magma [6] could help bridge this gap.

INTEGRATION AND OPERATIONAL CHALLENGES
THE HIDDEN COSTS

Security tools often appear “plug-and-play” or close in marketing materials. In reality, achieving effective real-world performance demands custom engineering work, tool fine-tuning, and ongoing adaptation.

SETUP & INTEGRATION COMPLEXITY

Security tools often require more effort than vendors disclose:

- **Custom Test Setup**:
Black-box fuzzers need custom harnesses and input generators for real-world apps [2][6][9].
- **Fine-Tuning SAST**:
Static analyzers require extensive tuning, sometimes taking months [16].
- **CI/CD Compatibility**:
Black-box fuzzers often unsuitable for fast-paced CI/CD due to long runtimes and high resource needs [14] while SAST tools integration depends on careful configuration and tuning [4][5][6][9].

AUTHENTICATION & STATEFUL TESTING LIMITATIONS

- **Authenticated Testing**:
Requires custom scripting for login workflows and session handling [9].
- **State Management**:
Without robust state mechanisms, tools miss multi-step interaction vulnerabilities. Even stateful grammar-based black box fuzzers require extensive configuration for custom protocols and will miss variations in standard ones. [9]

ONGOING MAINTENANCE & ADAPTATION

As applications evolve, both fuzzers and static analyzers **require ongoing tuning** — including revising rules, updating test inputs, and adapting to new frameworks — to maintain effectiveness [3][4][14][17].

BENCHMARK ASSESSMENT FRAMEWORK

Research highlights key questions that security leaders should ask when evaluating security tools to avoid misleading vendor claims

- Were tests conducted on **real-world, complex applications**, not just simplified test cases?
- Has the tool been tested against **previously unknown vulnerabilities**?
- Does the methodology disclose **exact testing parameters and application characteristics**?
- Do published metrics include **false positive rates, false negative rates, coverage, and implementation costs**?
- Are the results **independently verified** by third parties without vendor funding?

ACADEMIC-INFORMED SELECTION CRITERIA

Research suggests several **evidence-based criteria** for selecting and evaluating security tools:

- **Empirical evidence**: Prioritize tools with performance verified by independent evaluation, ideally academic research.
- **Coverage transparency**: Request detailed coverage metrics and detection rates. The number of found vulnerabilities alone says little about depth or reliability.
- **Developer experience**: Evaluate how the tool impacts daily workflows — false positives, feedback latency, and friction in local development.
- **Operational integration**: Assess the engineering effort required to implement, integrate, and maintain the tool across environments, including CI/CD compatibility and infrastructure overhead.
- **Complementary approaches**: No tool is comprehensive — security leaders should build a multi-layered strategy combining multiple tools and techniques to address known gaps.

CONCLUSION
TOWARD EVIDENCE-BASED SECURITY

The academic evidence presented in this paper reveals a **troubling disconnect** between security vendor marketing claims and real-world tool performance.

Organizations that rely on vendor benchmarks without critical evaluation risk **false confidence** and **misallocated security resources** — creating blind spots that attackers can exploit and leaving security teams to answer for failures they cannot prevent.

Security must move beyond vendor narratives. An effective security strategy starts with evidence-based evaluation and independent verification of tool performance.

By understanding the limitations documented in peer-reviewed research, security leaders can:

- Set **realistic expectations** for security tool capabilities.
- Allocate resources to **address actual gaps**, not just those vendors choose to highlight.
- Implement **complementary approaches** that go beyond a single tool or technique.

The path forward requires a **shift to evidence-based security decision-making**, with a focus on **independent, verifiable performance data**, not just marketing claims.

REFERENCES

1. OWASP Benchmark Projet: https://owasp.org/www-project-benchmark/
2. Jason Bau, Elie Bursztein, Divij Gupta, and John C. Mitchell. 2010. *State of the Art: Automated Black-Box Web Application Vulnerability Testing*. In Proceedings of the 2010 IEEE Symposium on Security and Privacy (SP '10). IEEE, 332–345.
3. Andrew Austin and Laurie Williams. 2011. *One Technique is Not Enough: A Comparison of Vulnerability Discovery Techniques*. In Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement (ESEM '11). IEEE, 97–106.
4. Stephan Lipp, Sebastian Banescu, and Alexander Pretschner. 2022. *An Empirical Study on the Effectiveness of Static C Code Analyzers for Vulnerability Detection*. In Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '22), pages 544–555.
5. Goseva-Popstojanova, K., & Perhinschi, A. (2015). *On the Capability of Static Code Analysis to Detect Security Vulnerabilities*. Information and Software Technology, 68, 18–33.
6. Johnson, B., Song, Y., Murphy-Hill, E., & Bowdidge, R. (2013). *Why Don't Software Developers Use Static Analysis Tools to Find Bugs?* Proceedings of the 35th International Conference on Software Engineering (ICSE).
7. Hazimeh, A., Herrera, A., & Payer, M. (2020). *Magma: A Ground-Truth Fuzzing Benchmark*. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 4(3), 1–29.
8. Cristian Daniele, Seyed Behnam Andarzian, and Erik Poll. 2024. *Fuzzers for Stateful Systems: Survey and Research Directions*. ACM Computing Surveys 56, 9, Article 222 (2024)
9. Kaixuan Li, Sen Chen, Lingling Fan, Ruitao Feng, Han Liu, Chengwei Liu, Yang Liu, and Yixiang Chen. 2023. *Comparison and Evaluation on Static Application Security Testing (SAST) Tools for Java*. In Proceedings of the 31st Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2023).
10. Roberto Natella and Van-Thuan Pham. 2021. *ProFuzzBench: A Benchmark for Stateful Protocol Fuzzing. In Proceedings of the 30th SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '21).*
11. Kayla Afanador and Cynthia Irvine. 2020. *Representativeness in the Benchmark for Vulnerability Analysis Tools (B-VAT)*. In Proceedings of the 13th USENIX Workshop on Cyber Security Experimentation and Test (CSET '20).
12. Aurelien M. Delaitre et al. 2018. *SATE V Report: Ten Years of Static Analysis Tool Expositions*. NIST Special Publication 500-326. National Institute of Standards and Technology, Gaithersburg, MD.

13. Maria Christakis and Christian Bird. 2016. *What Developers Want and Need from Program Analysis: An Empirical Study*. In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (Singapore, Singapore) (ASE 2016). Association for Computing Machinery, New York, NY, USA, 332–343
14. Klooster, T., Turkmen, F., Broenink, G., Ten Hove, R., & Böhme, M. (2023). Continuous Fuzzing: A Study of the Effectiveness and Scalability of Fuzzing in CI/CD Pipelines. In Proceedings of the 2023 IEEE/ACM International Workshop on Search-Based and Fuzz Testing (SBFT) (pp. 25–32).
15. Böhme, M., & Falk, B. (2020). *Fuzzing: On the Exponential Cost of Vulnerability Discovery*. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020) (pp. 713–724).
16. Nasif Imtiaz, Akond Rahman, Effat Farhana, and Laurie Williams. 2019. *Challenges with Responding to Static Analysis Tool Alerts*. In Proceedings of the 16th International Conference on Mining Software Repositories (MSR '19). IEEE Press, 245–249.
17. David Pérez-Palacín, Riccardo Cabassi, Raffaela Mirandola, and Catia Trubiani. 2021. *Continuous Static Analysis for Security: An Evaluation for Real-World Usage*. In Proceedings of the 15th European Conference on Software Architecture (ECSA '21). Association for Computing Machinery, New York, NY, USA, 151–166.