Ministry of Education of the Republic of Belarus
Educational institution
Belarusian State University of Informatics and Radioelectronics

UDC 004.056.53

Kondo

Kudakwashe, Ntando

Web Application Vulnerability Testing Technique

Abstract to master degree thesis

Specialty 7-06-0611-02 Information Security

_____

Supervisor:
N.V. Nasonova, Dr., Assoc. Prof.

_____

Minsk 2026

GENERAL DESCRIPTION OF WORK

The Research Aim and Objectives

This research focuses on designing and evaluating a proof-of-concept hybrid analysis technique for post-processing and prioritizing vulnerability findings. The scope includes:

1. Application-layer vulnerabilities aligned with categories represented in the OWASP Benchmark (e.g., injection, XSS, cryptography, path traversal) and the OWASP Top 10.
2. Primary evaluation on Java-based web applications (OWASP Benchmark, WebGoat), with preliminary assessment on a mixed Java/Python application (crAPI) and a PHP application (DVWA).
3. Utilization of specified open-source SAST and DAST tools as data sources.
4. Evaluation using the OWASP Benchmark v1.2 (for controlled metric computation) and manually verified subsets of crAPI, WebGoat, and DVWA (for real-world relevance).

Work Connection with Priority Areas of Scientific Research

The topic of the thesis corresponds to paragraph 6 Ensuring the security of humans, society and the state (means of technical and cryptographic information protection, cryptology and cybersecurity) of the list of priority areas of scientific, scientific-technical and innovative activities for 2021–2025, approved by the Decree of the President of the Republic of Belarus dated 07.05. 2020 No. 156.

Thesis Results Approbation

The main results of the dissertation were reported and discussed at the 61st Scientific conference of Graduate Students, Master Students and Students of BSUIR (Minsk, April 21–25, 2025), and the 62st Scientific conference of Graduate Students, Master Students and Students of BSUIR (Minsk, April 13–17, 2026).

INTRODUCTION

Background and Motivation

Web applications underpin critical digital services across finance, healthcare, government, and e-commerce. The rapid adoption of API-first designs, microservices, and cloud-native architectures has dramatically expanded the attack surface. Industry reports consistently show web application and API attacks as a primary vector for data breaches. For instance, Verizon's 2019 Data Breach Investigations Report found that 62% of security incidents involved web application attacks, and they were the primary vector in 25% of all breaches. This trend underscores the urgent need for effective, automated security testing integrated into modern development pipelines.

Core security testing methodologies–Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Interactive Application Security Testing (IAST) are foundational to secure development. However, each exhibits significant limitations when applied to contemporary web environments. SAST, while good at finding coding flaws early, suffers from high false-positive rates and lacks runtime context, missing environment-specific vulnerabilities. DAST observes runtime behavior but struggles with complex authentication, stateful logic, and comprehensive API coverage, also generating substantial noise; the Veracode 2019 State of Software Security Report noted DAST false-positive rates can reach 30%. IAST improves accuracy by correlating runtime instrumentation with requests but introduces deployment complexity and technology constraints.

In the last decade, academia and industry have increasingly explored hybrid approaches that combine static and dynamic analyses, and more recently, Machine Learning (ML) and anomaly detection to improve prioritization, reduce false positives, and detect novel or obfuscated attacks. For example, hybrid SAST/DAST models like Snyk, Checkmarx integrate code-level and runtime findings to lower mean-time-to-detect (MTTD). Runtime Application Self-Protection (RASP) embeds security controls directly into applications to block attacks in real time. Other advanced approaches leverage Large Language Models (LLMs) to augment traditional SAST scanners (LSAST) or employ Retrieval-Augmented Generation (RAG) to keep detection models current with emerging threats.

However, these existing hybrid frameworks exhibit notable limitations:

- Hybrid SAST/DAST integrations often struggle with tool correlation, data-format mismatches, and overlapping false positives.
- RASP and IAST introduce performance overhead (5–15% latency) and deployment complexity, making them unsuitable for resource-constrained or high-throughput environments.
- ML-based anomaly detection frequently suffers from explainability deficits, dependence on high-quality training data, and computational cost.
- LLM-enhanced approaches, for example, LSAST face challenges with irrelevant knowledge retrieval, privacy trade-offs, and high resource demands.

Consequently, security teams remain inundated with fragmented, noisy alerts

from disparate tools, leading to alert fatigue and potentially missing critical vulnerabilities amidst the clutter. This operational burden highlights a clear research gap: while hybrid frameworks represent a step forward, there is a lack of a lightweight, automated technique that effectively synthesizes static and dynamic evidence into a high-confidence, prioritized risk view without introducing significant overhead or complexity.

Problem Statement

Despite the emergence of hybrid testing frameworks, current vulnerability detection for modern, distributed web applications remains insufficiently cohesive and operationally burdensome. Existing hybrid approaches either introduce performance penalties, require complex integration, or fail to adequately correlate and prioritize findings across static and dynamic analysis tools. Specifically, there is a lack of an automated, evidence-driven, and lightweight technique that:

1. Correlates findings from SAST and DAST to suppress false alarms,
2. Leverages machine learning to classify and prioritize true vulnerabilities, and
3. Does so without imposing significant runtime overhead or deployment complexity.

Aim and Objectives

The primary aim of this research is to design and evaluate a hybrid vulnerability testing technique that advances the state-of-the-art by integrating evidence from static and dynamic analysis using a tailored machine-learning model, while explicitly addressing the limitations of existing hybrid frameworks (e.g., performance overhead, correlation complexity, and explainability).

Specific objectives are:

1. To systematically analyze the limitations of existing SAST, DAST, and current hybrid frameworks in detecting vulnerabilities in modern web architectures, focusing on coverage gaps, false-positive patterns, and operational drawbacks.
2. To develop a proof-of-concept hybrid analysis technique that advances prior work by:

a) Implementing a lightweight data-collection pipeline for standardized outputs from selected open-source SAST and DAST tools,

b) Designing a feature-engineering process that transforms raw tool findings into a unified, ML-ready feature set,

c) Building an interpretable machine-learning model for classifying and prioritizing aggregated findings to reduce false positives without the high computational cost or black-box nature of existing ML-hybrids,

d) Introducing a novel correlation logic that fuses corroborating evidence from static and dynamic sources while avoiding the integration overhead of traditional hybrid SAST/DAST platforms.

3. To validate the technique against a standardized benchmark dataset (OWASP Benchmark v1.2) and deliberately vulnerable applications (e.g., OWASP crAPI, WebGoat, DVWA) using precision, recall, and F1-score metrics, comparing

its performance against individual standalone SAST and DAST tools and simple rule-based baselines.

This study seeks to answer the following research questions:

1. To what extent can a machine learning classifier trained on tool-agreement features derived from open-source SAST and DAST outputs improve precision and recall compared to (a) individual standalone tools and (b) simple rule-based baselines, when evaluated on the OWASP Benchmark?
2. Can the ML-based classifier reduce the false-positive rate compared to raw SAST and DAST outputs on the OWASP Benchmark while maintaining recall above 90%, and can it provide interpretable predictions with low computational overhead?
3. How does the technique's performance vary across the vulnerability categories present in the OWASP Benchmark, and does the inclusion of authenticated dynamic analysis improve vulnerability prioritization in a real-world testbed?

Based on the research questions, the following hypotheses are proposed:

1. The hybrid technique will achieve a statistically significantly higher F1-score than any individual standalone SAST or DAST tool and any simple rule-based combination on the OWASP Benchmark test set, as assessed by McNemar's test ($p < 0.05$), while requiring only lightweight post-processing.
2. On the OWASP Benchmark test set, the calibrated classifier will reduce the false-positive rate by at least 15% compared to the raw outputs of individual SAST and DAST tools, while maintaining recall $\geq 90\%$ and providing SHAP-based interpretable predictions with inference times below 0.1 seconds. On an external testbed (crAPI), as preliminary evidence, the classifier will achieve precision@10 $\geq$ 0.70 and reduce high-confidence predictions by at least 80% relative to raw DAST alerts.
3. On the OWASP Benchmark, the technique will exhibit higher precision for vulnerability categories where dynamic analysis provides strong corroborating signals (e.g., SQL injection, command injection) compared to categories where static analysis dominates but suffers from high false-positive rates (e.g., path traversal). On an external testbed, the inclusion of authenticated dynamic analysis will lead to observable improvement in prioritization of access-control vulnerabilities, as assessed qualitatively.

MAIN PART

In the first chapter, the theoretical background of web application security testing is reviewed. The chapter analyzes the main testing paradigms, including SAST, DAST, IAST, hybrid and ML-enhanced approaches, compares their strengths and limitations, and identifies the key gap that motivates the research, namely the lack of lightweight and interpretable frameworks for correlating heterogeneous security findings. It also examines datasets, benchmarks, and evaluation testbeds used in this area.

In the second chapter, the methodology for developing the proposed hybrid vulnerability correlation framework is presented. The chapter describes the research design, tool selection, benchmark selection, data normalization, feature engineering, model development, ablation study, SHAP-based interpretability analysis, and comparison with standalone tools and simple rule-based baselines. It also presents the architecture of the proposed framework and explains how raw tool outputs are transformed into prioritized and interpretable vulnerability reports.

In the third chapter, the practical implementation and validation of the developed technique are described. The chapter presents the software environment, project structure, data flow, pipeline implementation, unit and integration testing, and experimental validation on several vulnerable applications, including OWASP crAPI, WebGoat, Juice Shop, and WAVSEP. It also analyzes the model's behavior on different testbeds, evaluates the research hypotheses, and discusses deployment and practical use of the framework.

The fourth chapter discusses the experimental findings within the broader landscape of hybrid vulnerability detection, explained the partial support for two of the three hypotheses through documented tool-coverage constraints and feature-set limitations, and articulated the specific contributions that this research makes to the field. The framework's consistent, interpretable behaviour across diverse testbeds demonstrates that a lightweight, consensus-based ML model can effectively reduce false positives and prioritise vulnerabilities–addressing the gap identified in the literature review between high-overhead LLM approaches and non-learning rule-based frameworks.

CONCLUSION

1. A hybrid web-application vulnerability testing technique was developed. The technique combines binary outputs from SAST and DAST tools by engineering consensus features (num_tools, num_disagree, semgrep_and_zap, spotbugs_and_zap) and applying a calibrated Random Forest classifier with Platt scaling. The model was trained on the OWASP Benchmark v1.2 (2 740 test cases) and deployed as a single-command Docker container [1-A, 2-A].

2. On the OWASP Benchmark, across 10 repeated stratified 80/20 splits, the technique achieved a mean F1-score of 0.808 ± 0.011, significantly outperforming the best standalone tool (Semgrep, 0.785; McNemar's $p = 0.0003$) and the best simple rule-based combination (AND rule, 0.791). The false-positive rate was reduced by 30.5 % relative to FindSecBugs ($0.531 \rightarrow 0.369$) and by 5.9 % relative to Semgrep ($0.392 \rightarrow 0.369$), while mean recall was maintained at 0.912 ± 0.012, exceeding the 90 % threshold specified in Hypothesis ii [2-A].

3. SHAP analysis demonstrated that the model relies primarily on tool-consensus features rather than on any single tool's output. The features num_disagree and num_tools were identified as the most influential predictors, confirming that the technique effectively correlates evidence from multiple, independent sources.

4. The technique was validated on four deliberately vulnerable applications representing four distinct technology stacks and varying tool-coverage conditions:

- On crAPI (Python/Java, two tools, 44 endpoints), the technique reduced the analyst's review burden from 44 raw alerts to 7 high-confidence predictions (84 % reduction) with Precision@10 = 0.50.
- On WebGoat (Java, three tools, 114 endpoints), the technique flagged only the three endpoints where Semgrep, SpotBugs, and ZAP all agreed, achieving perfect precision (1.00) and a 97 % alert reduction.
- On Juice Shop (Node.js, two tools, six reachable endpoints), an authenticated ZAP scan recovered a previously failed pipeline; manual mapping of Semgrep findings raised the model's confidence from 0.675 to 0.743 on four endpoints, and all six flagged endpoints were true positives.
- On WAVSEP (JSP, one tool, 206 endpoints), a solitary DAST signal produced uniform moderate probabilities (0.675), identical to the value observed for DAST-only endpoints on Juice Shop.

5. Across all four external testbeds, the technique never introduced additional false positives beyond those already present in the input tools. When tool coverage was sparse, the model correctly defaulted to conservative, low-confidence predictions, confirming the graceful degradation predicted by the SHAP analysis.

6. The framework was distributed as a publicly available Docker container (https://github.com/Ntandolis431/vuln-correlation-framework) that includes the trained model, the inference engine, the SHAP explainer, and all Python dependencies. No local installation of libraries or security tools is required; a single command transforms raw scan results into a ranked, explainable vulnerability report in seconds.

7. The partial support for Hypotheses ii and iii was honestly documented. The 15 % FPR reduction target was met against FindSecBugs but not against Semgrep because the latter already possessed a moderate FPR that left limited room for improvement under the recall constraint. On crAPI, Precision@10 fell short of the hypothesized 0.70 due to the inherent discriminative limits of a two-tool feature set. The external-validation component of Hypothesis iii was qualitatively supported but could not be quantitatively confirmed due to incomplete ground-truth data on access-control vulnerabilities.

8. The principal limitations of the study–language coverage of the input SAST tools, availability of ground truth for external testbeds, and the training dataset being restricted to synthetic Java test cases–define directions for future work. Extending the tool-integration layer, adding native JavaScript and Python analyzers, and evaluating the technique on production applications with audited ground truth are recommended next steps.

The developed technique addresses the gap identified in the literature between high-overhead LLM-based approaches and non-learning rule-based frameworks. It provides a practical, interpretable, and reproducible method for reducing false positives in multi-tool application security testing.

Scientific and Practical Value of the Work

The scientific contribution of this thesis is the empirical demonstration that tool consensus – measured simply by counting how many independent SAST and DAST tools flag the same endpoint – is the strongest available signal for distinguishing true vulnerabilities from false positives within the developed framework. SHAP analysis and ablation experiments consistently identify the engineered consensus features as the dominant predictors, and this pattern holds across all four external testbeds. The thesis further documents a consistent, conservative degradation curve in model confidence as tool coverage varies: SAST-only findings receive a low vulnerability probability, DAST-only findings a moderate probability, and multi-tool agreement the highest probability. This quantified relationship between tool consensus and prediction confidence was reproduced across applications in four different technology stacks and represents a finding not previously reported in this form. To the best of the author's knowledge, the work also presents the first published machine-learning-based vulnerability

prioritization results on the OWASP crAPI testbed, establishing a baseline for future research on that application.

The practical value of the work lies in delivering a lightweight, interpretable, and deployable tool that directly addresses the problem of alert fatigue in application security. On two testbeds, the framework reduced the number of alerts requiring manual analyst review by 84 % and 97 % respectively, relative to the raw union of all tool outputs, without introducing additional false positives. The complete pipeline is containerized and publicly available as a single-command Docker container, requiring no local Python installation or dependency management. Per-endpoint SHAP waterfall plots expose each feature's contribution to the model's prediction, enabling analysts to inspect the reasoning behind individual flagged endpoints and supporting human oversight of automated triage decisions. All results are scoped to the testbeds and ground truths evaluated; generalization to production codebases is reserved for future work involving real-world application corpora and formal user evaluation.

LIST OF OWN PUBLICATIONS

1-A. Kondo K. N. Web application vulnerability testing framework / Kondo K. N., Nasonova N. // Technical Means of Information Protection : Materials of the XXIII International Scientific and Technical Conference, Republic of Belarus, Minsk, April 8, 2025. – Minsk, 2025. – P. 20-22.

2-A. Kondo K. N. Efficient Web-Vulnerability Detection Technique Using Hybrid SAST-DAST Analysis and Machine Learning / Kondo K. N., Nasonova N. // Information Security : Proceedings of the 62nd Scientific Conference of Graduate Students, Master Students and Students of BSUIR, Minsk, April 13–17, 2026 / Belarusian State University of Informatics and Radioelectronics. – Minsk, 2026. – P. 40-42.