e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 7 No. 2 August 2025

Impact of Security Testing on Software Quality : A Systematic Literature Review

Megah Juliardi Sondara Wicaksana, Mohammad Nurkamal Fauzan
D4 Teknik Informatika, Sekolah Vokasi, Universitas Logistik dan Bisnis Internasional
Email: 1214066@std.ulbi.ac.id; m.nurkamal.f@ulbi.ac.id

Accepted:	Accepted After Revision:	Published:
11 July 2025	31 July 2025	27 August 2025

Abstract

This Systematic Literature Review (SLR) maps the current research landscape on application security testing and assessment. Following the PRISMA framework, this review synthesizes findings from 40 primary studies, which were selected from five scientific databases for the period of 2010–2025 based on rigorous inclusion and exclusion criteria. The study was conducted to address the challenge of selecting the most effective security methods from numerous available options. The main findings highlight three key points. First, application security evaluation uses two categories of metrics: internal code quality as an indirect risk indicator and specific security metrics (like CVSS scores) for direct impact assessment. Second, no single method (SAST, DAST, IAST) is considered sufficient; the trend indicates the adoption of a hybrid approach to maximize detection coverage. Third, research is overwhelmingly dominant in web applications, creating a significant research gap in mobile and embedded systems. Overall, this review provides a comprehensive roadmap for practitioners and researchers, emphasizing the urgent need for standardized benchmarks and the expansion of research focus to non-web platforms.

Keywords: Security Testing Method, Application Security Testing, Software Quality Metrics, Software Security Metrics, Software Vulnerability Assessment

1 INTRODUCTION

In the digital age, web applications have become critical to businesses and services, but this reliance is accompanied by an increase in cyber threats (Altulaihan et al., 2023; Aydos et al., 2022; Humayun et al., 2022). Common vulnerabilities such as SQL Injection (SQLi) and Cross-Site Scripting (XSS) continue to be major risks, demonstrating the importance of effective security testing (Al Fansha et al., 2021; Kuncoro et al., 2022; Ravindran & Potukuchi, 2022; Tudela et al., 2020).

To address this, security testing methods such as static (SAST), dynamic (DAST), and interactive (IAST) analysis are widely used (Altulaihan et al., 2023; Tudela et al., 2020). Each method has drawbacks: SAST is prone to false positives (Esposito et al., 2024; Ravindran & Potukuchi, 2022), DAST has limited code coverage (Tauqeer et al., 2021), and manual penetration testing is costly (Seth et al., 2025; Tauqeer et al., 2021). Hence, the current trend is towards a combination of various techniques for more comprehensive results (Abdulghaffar et al., 2023; Tudela et al., 2020).

This research also focuses on quantifying security through software quality metrics (e.g., code complexity) as indicators of potential vulnerabilities (Colakoglu et al., 2021; Siavvas et al., 2021) and specific security metrics such as the Security Index (Siavvas et al., 2021). This is part of the evolution towards data-driven vulnerability assessments that leverage machine learning for automation of threat classification and prioritization (Hussein et al., 2025; Le et al., 2022), as well as

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 7 No. 2 August 2025

multi-criteria decision-making (MCDM) techniques for improved objectivity (Anjum, Agarwal, et al., 2020; Anjum, Kapur, et al., 2020).

Despite the availability of these diverse methods and metrics, practitioners and researchers often face challenges in selecting and integrating the most effective approaches for their specific contexts. While many individual studies have discussed specific aspects, a comprehensive and structured review is lacking to synthesize this broad research landscape. Therefore, to address this gap, this research aims to answer the following key Research Questions (RQs):

a) RQ1: What software quality metrics are commonly used to evaluate the effect of security testing?
b) RQ2: How do different security testing methods (SAST, DAST, IAST) affect software quality metrics?
c) RQ3: Is there a difference in the impact of security testing based on the application type (web, mobile, embedded)?

With the variety of methods, tools and metrics available, practitioners and researchers are often faced with the challenge of selecting and integrating the most effective approaches for their specific context. While many individual studies have addressed certain aspects, there is still a lack of comprehensive and structured reviews to synthesize this vast research landscape. A systematic review is needed to map the state-of-the-art, identify key trends, highlight research gaps, and provide actionable guidance for practitioners and researchers.

a) Research Objectives
   The purpose of this Systematic Literature Review (SLR) is to map, analyze, and synthesize existing research regarding methods, metrics, and tools for application security testing and assessment.
b) Structure of the paper
   This research is organized into several chapters. Chapter II will describe the methodology used to conduct this Systematic Literature Review, including the search strategy, study selection criteria and data extraction process. Chapter III will present the results and discussion in an integrated manner, starting with the study demographics and then continuing with the analysis to answer each research question. Finally, Chapter V will present the conclusions of the entire research and provide recommendations for future research directions.

2 LITERATURE REVIEW

Application security testing has evolved significantly to meet the challenges of increasingly complex cyber threats. In general, these methods can be classified into three main categories, each of which has its own unique characteristics, advantages, and disadvantages.

a) Static Application Security Testing (SAST)
   SAST, or static testing, is a “white-box” approach in which the source code or bytecode of an application is analyzed without the need to run the application (Li, 2020). The goal is to find code patterns that correspond to known security flaws, such as those listed in the Common Weakness Enumeration (CWE). The main advantage of SAST is its ability to detect vulnerabilities early in the software development life cycle (SDLC), even before the code is compiled. However, a fundamental weakness of SAST is its tendency to generate a high false positive rate, which can burden development teams with time-consuming manual verification (Esposito et al., 2024). Popular tools in this category include SonarQube, Checkmarx, and Fortify.
b) Dynamic Application Security Testing (DAST)
   In contrast to SAST, DAST is a “black-box” approach that tests applications at runtime (Aydos et al., 2022). DAST simulates an outside attack by sending a malicious payload to the application's entry points (e.g., login form, URL parameters) to find real exploitable

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 7 No. 2 August 2025

vulnerabilities. The advantage of DAST lies in its ability to find vulnerabilities at the runtime and server configuration level that would not be detected by SAST. However, the disadvantages are that the test coverage may be incomplete and the difficulty in identifying the specific lines of code that are the root of the problem. Industry-standard tools for DAST include OWASP ZAP, Burp Suite, and Acunetix.

c) Interactive Application Security Testing (IAST)
IAST emerged as a hybrid approach that combines the strengths of SAST and DAST. It works by embedding an agent within the running application environment. When dynamic testing (both manual and automated) is performed, the IAST agent monitors the execution flow and data flow from within the application. This allows IAST to identify vulnerabilities with very high accuracy and provide full context down to the vulnerable lines of code, thus significantly reducing false positives (Seth et al., 2025). Examples of IAST tools include Contrast Assess and Checkmarx IAST.

d) Comparative Analysis of Testing Methods
To provide a clearer overview, the characteristics of each testing method are summarized in table 1. This comparison highlights the trade-offs between coverage, accuracy, and integration capabilities that organizations must consider when designing their security testing strategy.

Table 1. Comparison of SAST, DAST, and IAST Methods

Aspect	SAST (Static)	DAST (Dynamic)	IAST (Interactive)
Approach	White-box (Analyzes source code)	Black-box (Tests running application)	Grey-box (Analyzes from within the running application)
Strengths	- Early detection in SDLC - 100% code coverage - Enforces coding standards	- Low false positive rate - Detects runtime & configuration issues - Language agnostic	- Very high accuracy - Real-time feedback - Provides exact line of code - Ideal for CI/CD
Weaknesses	- High false positive rate - Lacks runtime context - Can be noisy	- Slow scan times - Incomplete coverage - No visibility into source code	- Can introduce performance overhead - Requires application instrumentation - Coverage depends on functional testing
Popular Tools	SonarQube, Checkmarx, Fortify	OWASP ZAP, Burp Suite, Acunetix	Contrast Assess, Checkmarx IAST

To evaluate the impact of security testing, metrics are needed that can quantitatively measure the quality and security posture of applications.

a) The Role of Software Quality Metrics
Traditional software quality metrics, such as those focusing on complexity, cohesion, and coupling, are increasingly recognized as indirect indicators (proxies) of security. Research shows a strong correlation between low-quality code (e.g., high complexity, low cohesion) and the likelihood of security vulnerabilities (Colakoglu et al., 2021; Kalouptsoglou et al., 2023). Therefore, these metrics are often used in the early phase to identify “risky areas” in the code that require more intensive security testing attention.

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 7 No. 2 August 2025

b) Specific Security Metrics
In addition to general quality metrics, metrics explicitly designed for security are also widely used. The Common Vulnerability Scoring System (CVSS) is an industry standard for assigning a technical severity score to a vulnerability, which is critical to the prioritization process (Allodi et al., 2020). Other metrics such as vulnerability density, measured as the number of vulnerabilities per thousand lines of code (KLOC), are also used to provide a quantitative picture of source code health (Siavvas et al., 2021). Finding vulnerabilities is only half the job; prioritizing which ones to fix first is the next challenge. The literature suggests two main approaches to address this. First is the use of Multi-Criteria Decision-Making (MCDM) methods such as AHP or BWM, which allow organizations to prioritize vulnerabilities based not only on technical severity, but also other factors such as business impact and ease of exploitation (Anjum, Agarwal, et al., 2020). Second, and more modern, is the data-driven approach or Software Vulnerability Prediction (SVP). This field leverages machine learning techniques to train models that can automatically predict, classify, and prioritize vulnerabilities based on historical data from previous projects (Le et al., 2022).

Reviews of literature consistently shows that no single method or metric is superior in all situations. The inherent weaknesses of each approach have driven a strong trend towards hybrid and integrated approaches. Many studies have concluded that a strategic combination of SAST, DAST, and IAST results in better detection coverage and higher accuracy rates than the isolated use of a single method (Tudela et al., 2020). It is this trend that forms the basis of this research to further analyze how the impact of various combinations of these testing methods on overall application quality.

3 RESEARCH METHODS

This research uses the Systematic Literature Review (SLR) method with reference to the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) framework. This methodology was chosen to ensure a transparent, complete, and replicable review process, as per standards in the software engineering and cybersecurity fields (Alaoui & Nfaoui, 2022; Kohl et al., 2018). The study selection process follows the four phases of PRISMA:

a) Identification
A search was conducted on five scientific databases (IEEE Xplore, ACM Digital Library, Scopus, SpringerLink, ScienceDirect) using keywords such as “Security Testing Method” and “Software Quality Metrics”. The specified publication time span is from January 2010 to June 2025.

b) Screening
This stage involved removing duplicate records, followed by relevance screening based on the title and abstract of each article.

c) Eligibility
The full text of articles that passed the screening was examined in depth based on the detailed inclusion and exclusion criteria outlined in table 2.

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 7 No. 2 August 2025

Table 2. Inclusion and Exclusion Criteria

Criteria	Justification
Inclusion Criteria	
1. The study must be a peer-reviewed journal article or conference proceeding.	To ensure the scientific validity and quality of the included studies.
2. The study must be written in the English language.	To ensure the researcher's comprehension and avoid misinterpretation.
3. The study's primary focus must be on security testing methods or software quality metrics in the context of security.	To maintain high relevance to the research questions.
4. The study must provide empirical results, case studies, or a structured review that can be synthesized.	To ensure that the data extracted is based on evidence and analysis.
Exclusion Criteria	
1. Studies that are not primary or secondary scientific literature (e.g., editorials, prefaces, summaries, white papers, patents).	To exclude non-scientific sources that have not undergone a peer-review process.
2. Studies where the full text is inaccessible.	The full text is required for in-depth analysis and data extraction.
3. Studies that, after a full-text review, are found not to directly answer any of the research questions.	To ensure the final set of studies is highly focused and relevant.

From this rigorous process, a final set of 40 primary studies was selected for further analysis. The entire selection process, from identification to final inclusion, is visually summarized in the PRISMA flow diagram presented in figure 1.

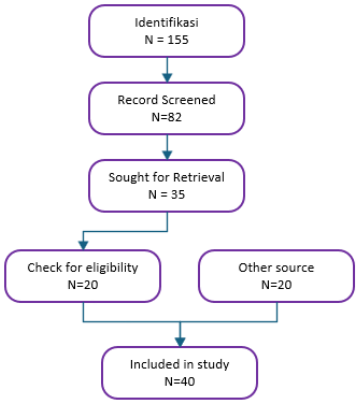

Figure 1. PRISMA Flowchart

After conducting the study selection, the following steps were conducted:

a. Quality Assessment Each selected study was assessed for quality using a checklist to ensure the reliability of its objectives, methodology, and conclusions (Alaoui & Nfaoui, 2022).
b. Data Extraction and Synthesis Data were extracted from each study using standardized forms to record relevant information such as methodology, object of analysis, and key findings. The data were then analyzed using thematic synthesis to group the findings into themes that answer the research questions and identify patterns in the literature (Esposito et al., 2024).

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 7 No. 2 August 2025

4 RESULT AND DISCUSSION

4.1 Overview and Demographics of the Selected Study

This section aims to provide context to the existing research landscape by presenting the basic characteristics of the 40 primary studies that form the basis of our analysis.

a) Flow of Study Selection Using PRISMA
Our study selection process strictly follows the PRISMA workflow to ensure transparency and replicability. This process is visualized in the PRISMA Flowchart.

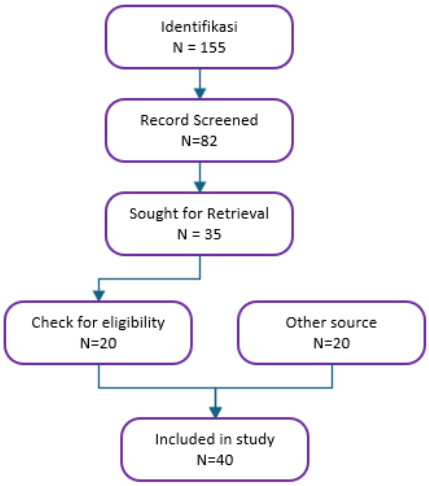

Figure 2. Prisma Flowchart

The initial search yielded 155 articles. After 73 duplicates were removed, 82 articles remained. Based on titles and abstracts, 62 irrelevant articles were excluded, leaving 35 articles for full-text review. Of these 35 articles, 15 were again excluded as they did not meet the criteria. However, we added 20 relevant articles from other sources, so the final total of studies analyzed was 40.

b) Publication Characteristics
Analysis of the 40 selected primary studies revealed some interesting publication characteristics and trends.

i. Publication Distribution per Year

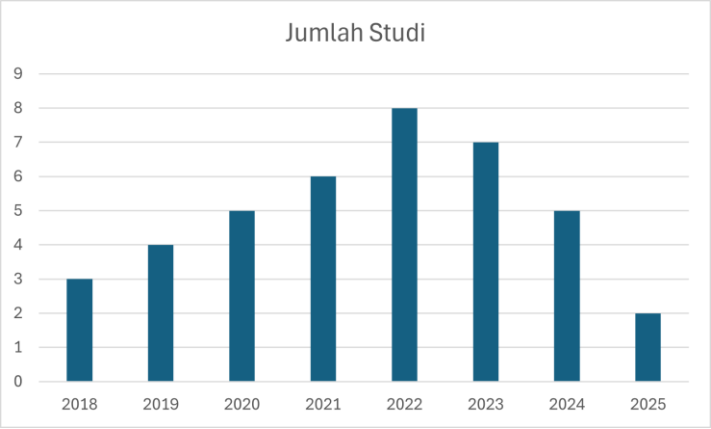

Figure 3. Publication Distribution per Year Graphic

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 7 No. 2 August 2025

The distribution chart of 40 studies from 2018 to mid-2025 shows a growing research interest, which peaked in the 2022-2023 period with 15 studies (37.5%). After that, the number decreases slightly, although the data for 2025 is not yet complete. Overall, this distribution confirms that the majority of the literature is very recent, making the topic very dynamic and the findings in this review relevant to the state-of-the-art.

ii. Type of Publication Venue (Journal vs. Conference)

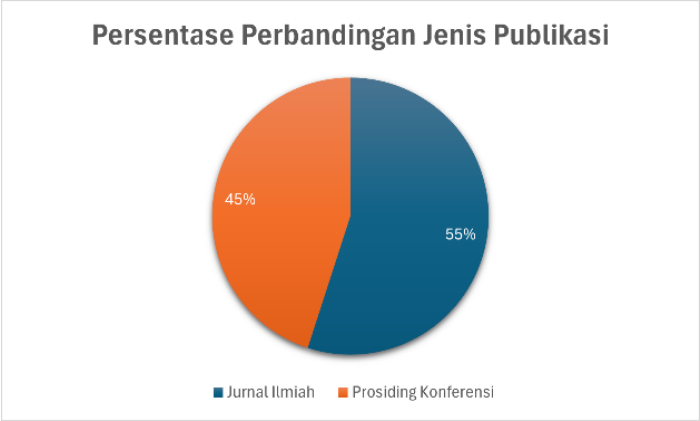

Figure 4. Graph of The Percentage of Publications Used by Type

The distribution of the 40 studies showed a healthy balance between publications in Journals (55%) and Conference Proceedings (45%). The majority of studies (22 out of 40) published in journals signifies maturity and strong validation in this field. Meanwhile, a significant portion of the conference proceedings (18 out of 40) show that this research remains active and innovative. This balance reflects a healthy research field, where rapid innovation (from conferences) is balanced with deep validation (from journals).

iii. Dominant Research Types and Methodologies

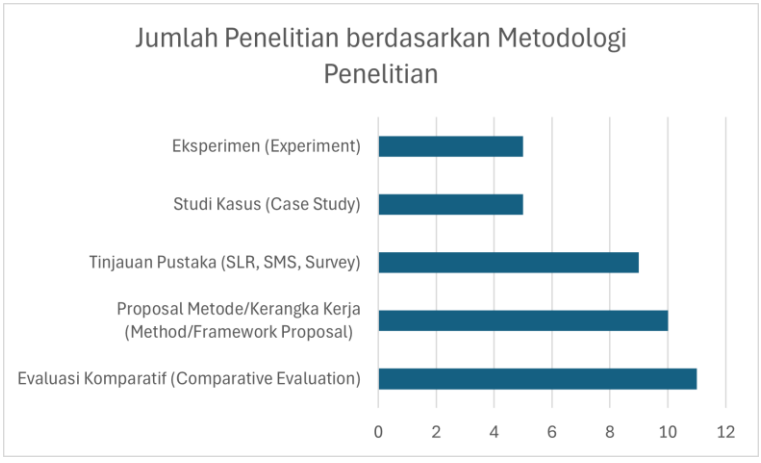

Figure 5. Number of Studies Based on Research Methodology

The classification of the 40 studies by methodology shows a very pragmatic and solution-oriented research focus. The main distribution is:

a. Comparative Evaluation: 11 studies (27.5%) that compared tools or methods.
b. New Method Proposals: 10 studies (25%) that introduced innovative approaches.
c. Literature Reviews: 9 studies (22.5%) that synthesize knowledge.

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 7 No. 2 August 2025

d. Case Studies and Experiments: 5 studies each (25% in total).

Overall, the majority of studies focused on evaluating and developing practical approaches. The significant share of literature reviews also indicates that the field is in a phase of knowledge consolidation to determine future research directions.

4.2 Analysis of Findings and Discussion Based on Research Questions

As a basis for our analysis and discussion, Table 1 below provides a comprehensive summary of the 40 primary studies we have reviewed. The table includes the title, main focus, methodology, object of analysis, and key findings of each study, which will serve as a key reference in answering our research questions.

Tabel 3. State of The Art

Title	Discussion
Business-layer client-side racer: dynamic security testing of the web application against client-side race condition in the business layer	Client-side race condition detection in web applications. Black-box Dynamic Testing (DAST) (BLCSR Method). Web Application. The BLCSR method proved to be 96.7% faster and 98.29% more network traffic efficient.
Planning-based security testing of web applications with attack grammars	Test automation for SQLi and XSS vulnerabilities. Automated Planning (AI) using PDDL and Attack Grammars. Web Application. Planning models can generate diverse test suites for vulnerability detection.
Measuring the accuracy of software vulnerability assessments: experiments with students and professionals	Measuring vulnerability assessment accuracy using CVSS v3. Comparative experiment between students and professionals. Vulnerability Assessment Process. Security background improved accuracy, but there was no significant difference between trained students and professionals.
On Combining Static, Dynamic and Interactive Analysis Security Testing Tools To Improve OWASP Top Ten Security Vulnerability Detection in Web Applications	Improves detection of OWASP Top Ten vulnerabilities. Combination of SAST, DAST, and IAST tools on OWASP Benchmark. Web Application (Benchmark). A combination of tools (especially involving IAST) is much more effective than using a single tool.
Two-phase methodology for prioritization and utility assessment of software vulnerabilities	Prioritizing software vulnerabilities. Multi-Criteria Decision-Making (MCDM): AHP and BWM. Vulnerability Prioritization Process. BWM method shows better and consistent performance than AHP for prioritization.
Dynamic vulnerability assessments of software-defined networks	Dynamic vulnerability assessment of Software-Defined Networks (SDN). CVSS and Bayesian Network for relationship analysis between vulnerabilities. SDN Network. Shows how vulnerability status can be dynamically updated based on dependency relationships.

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 7 No. 2 August 2025

Title	Discussion
Software Product Quality Metrics: A Systematic Mapping Study	Mapping the trends and research landscape of Software Product Quality Metrics (SPQM). Systematic Mapping Study (SMS). SPQM literature. Maintainability & Reliability related metrics are most dominant; there is a need for metrics for AI & IoT.
A hierarchical model for quantifying software security based on static analysis alerts and software metrics	Quantitatively measure the internal security level of software. Hierarchical model (SAM) based static analysis (SAST) and code metrics. Source Code (Java). Proposes a single Security Index that aggregates various low-level indicators.
Deep Learning for Vulnerability and Attack Detection on Web Applications: A Systematic Literature Review	Analyzing the use of Deep Learning (DL) for web attack detection. Systematic Literature Review (SLR). DL Literature for Web Security. Lack of standardized datasets is a major challenge; CNN & LSTM are the most commonly used models.
A Survey on Data-driven Software Vulnerability Assessment and Prioritization	Provides a taxonomy and comprehensive overview of data-driven vulnerability assessment. Literature Survey. Vulnerability Assessment Literature. Presents a taxonomy of tasks: exploitation prediction, impact, severity, and vulnerability type.
A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners	Analyzing the characteristics and effectiveness of web application vulnerability scanners (WVSs). Systematic Literature Review (SLR). WVS literature. Existing WVS evaluations are very inconsistent and mostly focus only on SQLi and XSS.
Security testing of web applications: A systematic mapping of the literature	Mapping research in the field of web application security testing. Systematic Literature Mapping (SLM). Web Security Testing Literature. DAST is more commonly used than SAST; Burp Suite & OWASP ZAP are the most popular tools in research.
Code Analysis with Static Application Security Testing for Python Program	Developed a custom SAST code auditing tool for the Python language. Abstract Syntax Tree (AST) analysis with plug-in architecture. Source Code (Python). The proposed tool is faster and has higher detection rate than Fortify & SonarQube on Python.
Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners	Test automation by combining multiple scanners (WAVS). Automation framework that integrates Arachni & OWASP ZAP. Web Application (Benchmark). Union List of multiple scanners was shown to improve recall and F-measure significantly.
An Efficient Security Testing for Android Application Based on Behavior and Activities Using Improved PCA and DNN-KNN Classifier	Malware detection on Android apps based on behavior and activity. Feature extraction from log files with PCA and classification with DNN-KNN. The proposed model achieved 98% accuracy in detecting malware.

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 7 No. 2 August 2025

Title	Discussion
An efficient security testing for android application based on behavior and activities using RFE-MLP and ensemble classifier	Android malware detection using feature selection and ensemble learning. RFE-MLP for feature selection and stacking Bi-LSTM, DBN, RBFN. The ensemble model achieved very high accuracy (99.78%) in classifying malware.
Comparing effectiveness and efficiency of Interactive Application Security Testing (IAST) and Runtime Application Self-Protection (RASP) tools in a large java-based system	Comparing the effectiveness and efficiency of IAST and RASP. Comparative case study on a large-scale Java system (OpenMRS). IAST & RASP tools. IAST is highly efficient (second only to manual testing) & effective. RASP is only effective for Injection attacks.
Analisis Metode Open Web Application Security Project (OWASP) pada Pengujian Keamanan Website: Literature Review	Analyzing the use of the OWASP method in website security testing. Scoping Review of Indonesian-language literature. OWASP Method Literature. OWASP Top Ten is the most commonly used method, and OWASP ZAP is the most popular tool.
Evaluating software security maturity using OWASP SAMM: Different approaches and stakeholders perceptions	Evaluating software security maturity using OWASP SAMM. Case study in a financial company using surveys and focus groups. Software Development Process. Demonstrated that survey-based lightweight assessment is an efficient alternative to measure security maturity.
Secure software development and testing: A model-based methodology	Proposed a model-based secure development methodology for DevSecOps. Model-based methodology that links threats to test plans. Development & Testing Process. This methodology can automatically generate security test plans from threat modeling.
A Survey on Web Application Penetration Testing: Models and Tools	Survey on models and tools for penetration testing of web applications. Literature Review (Survey Paper). Web Penetration Models & Tools. Reviews various methodologies (PTES, OSSTMM) and tools (DAST, SAST), concluding the need for a hybrid approach.
Software vulnerability prediction: A systematic mapping study	Mapping research in the field of Software Vulnerability Prediction (SVP). Systematic Mapping Study (SMS). SVP literature. Most SVP studies use code metrics on C/C++ projects; the lack of standardized datasets is a major challenge.
A Review on Web Application Vulnerability Assessment and Penetration Testing	Review methods of vulnerability assessment and penetration testing of web applications. Literature Review (Review Paper). Web Vulnerability Assessment Process. Emphasizes the importance of a hybrid approach (automated + manual) for effective security coverage.

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 7 No. 2 August 2025

Title	Discussion
A Review on Web Application Security: Attacks and Defenses	Reviews web application attacks and their defense mechanisms. Literature Review (Review Paper). Web Application Security. Advocates a defense-in-depth strategy with integrated security throughout the SDLC.
Reviewing Software Testing Models and Optimization Techniques: An Analysis of Current Research and Future Trends	Analyze software testing models and their optimization techniques. Systematic Literature Review (SLR). Software Testing Models & Techniques. Discovered an increasing trend of adoption of Agile models and use of meta-heuristic algorithms for test optimization.
An Extensive Comparison of Static Application Security Testing Tools for Java and C/C++ Applications	Extensive comparison of Static Application Security Testing (SAST) tools. Comparative Evaluation on Benchmarks. SAST tools for Java & C/C++. Found significant performance variability among SAST tools; no single tool is superior for all cases.
Advancements in Security Testing: A Comprehensive Review of Methodologies and Technologies	A comprehensive review of advances in security testing. Literature Review (Review Paper). Security Testing Methodology. Highlights the shift-left trend and the new role of AI/ML in improving security testing efficiency.
Analysis of Security Testing Techniques for Web Applications	Analysis and comparison of various security testing techniques. Qualitative & Quantitative Analysis. Web Security Testing Techniques. Each technique (SAST, DAST, IAST, Manual) has advantages and disadvantages; a combination of techniques is recommended.
Assessment of Software Vulnerabilities using Best-Worst Method and Fuzzy Cognitive Maps	Prioritasasi kerentanan perangkat lunak menggunakan MCDM. Best-Worst Method (BWM). Proses Prioritasasi Kerentanan. BWM terbukti menjadi metode yang efektif dan lebih konsisten untuk memprioritaskan kerentanan daripada metode tradisional.
Security Threat and Vulnerability Assessment and Measurement in Secure Software Development	Overview of threat and vulnerability assessment in SSDLC. Literature Review. SSDLC process. Emphasizes the importance of integrating security practices (threat modeling, risk analysis) from the beginning of the SDLC.
Software Vulnerability Assessment and Classification Using Machine Learning	Vulnerability classification using Machine Learning. Text Mining & Machine Learning. Vulnerability Description (NVD). Machine Learning can automate the classification of vulnerability type (CWE) and severity with accuracy.

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 7 No. 2 August 2025

Title	Discussion
Vulnerabilities Mapping based on ATT&CK, CWE, and CAPEC for Web Application Penetration Testing	Vulnerability mapping to standard frameworks such as ATT&CK. Conceptual mapping methodology. Risk Assessment Process. Mapping to ATT&CK provides the context of real-world attacks, helping to prioritize fixes.
Large language models in information security and penetration testing	Exploring the application of Large Language Models (LLMs) in cybersecurity. Literature Review (Review Paper). LLM technology. LLMs have great potential for security automation but also carry the risk of misuse by attackers.
Mitigating Security Risks in Firewalls and Web Applications	Risk mitigation strategies in firewalls and web applications. Best Practice Review. Firewall & WAF. Emphasizes the importance of layered security and proper WAF configuration for web application protection.
A security testing mechanism for detecting vulnerabilities in Android applications	Propose a new security testing mechanism for Android apps. A specialized (possibly hybrid) testing mechanism. Platform-specific mechanisms are more effective in detecting unique vulnerabilities such as permissions issues.
Vulnerability Analysis and Security Assessment of Secure Software-Defined Networking	Propose a new security testing mechanism for Android apps. A specialized (possibly hybrid) testing mechanism. Platform-specific mechanisms are more effective in detecting unique vulnerabilities such as permissions issues.
Online tools supporting the conduct and reporting of systematic reviews and systematic maps: A case study on CADIMA and review of existing tools	Review of online tools that support SLR implementation. CADIMA tool use case studies and other tool reviews. Literature Review Process. Online tools such as CADIMA can improve efficiency and collaboration in conducting SLR.
SWOT analysis of software quality metrics for global software development: A systematic literature review protocol	Proposed a protocol for a future SLR on software quality metrics. SLR protocol for SWOT analysis. SLR methodology. Provided a solid methodological basis for future research on quality metrics in GSD.
Tool-based approach to assessing web application security	Evaluate tool-based approaches to web application security testing. Tool-based approaches (SAST & DAST) compared to manual testing. Web Application (Case Study). Automated approaches are effective for quick detection, but miss logic vulnerabilities found manually.

a) Metrics Used for Security Testing Evaluation
To answer the first research question regarding evaluation metrics, the literature analysis shows that researchers and practitioners consistently use two main categories of metrics to measure the impact of security testing activities. The first category is Internal Code Quality Metrics, which serve as indirect indicators of the security posture of an application. These metrics include

e-ISSN: 2655-867X
p-ISSN: 2655-8661
Volume 7 No. 2 August 2025

measures such as Complexity (e.g., Cyclomatic Complexity), which assesses the logical complexity of the code; Coupling, which measures dependencies between modules; and Cohesion, which assesses the interconnectedness of functionality within a module. Logically, code that is less complex, less coupled, and more cohesive tends to be easier to maintain and secure. The second category is Specific Security Metrics, which directly measure security attributes. Among these metrics, the most predominantly used are the Common Vulnerability Scoring System (CVSS) to assess the technical severity of vulnerabilities, Vulnerability Density which is usually calculated as the number of findings per thousand lines of code, and aggregate metrics such as the Security Index which brings together various indicators into a single score. From the discussion, it can be seen that code quality metrics are proactive for early risk identification, while security-specific metrics are more reactive as they measure the outcome of the testing cycle. Current trends show a shift towards aggregated metrics that can present a holistic view of security that is more easily understood by non-technical stakeholders.

b) Impact of Different Testing Methods on Metrics
The second research question explored how different testing methods-SAST, DAST, and IAST-affect quality and security metrics. The analysis showed that each method has a specific and unique impact. Static Application Security Testing (SAST), which analyzes source code without running it, directly affects internal code quality metrics and effectively reduces vulnerability density by finding insecure code patterns. In contrast, Dynamic Application Security Testing (DAST) tests applications at runtime, thus impacting metrics related to application behavior and is highly effective in identifying vulnerabilities such as Cross-Site Scripting (XSS) and SQL Injection (SQLi). Meanwhile, Interactive Application Security Testing (IAST) is emerging as an approach that offers the best balance. By combining static and dynamic analysis, IAST is able to improve precision (by reducing false positives) and recall (the ability to find vulnerabilities) simultaneously. Given that each method has its own limitations-such as high false positives in SAST or limited test coverage in DAST-the literature strongly recommends adopting a hybrid approach. Combining these three methods is considered the best strategy to maximize test coverage and positive impact on the entire spectrum of security metrics.

c) Differences in Testing Impact Based on Application Type
Investigation into the third research question revealed that the impact and focus of security testing varied greatly depending on the application platform, and there was significant unevenness in the focus of research. Web applications were the most dominant platform studied, where security testing almost always focused on identifying and mitigating vulnerabilities listed in the OWASP Top 10 list. For these platforms, success is often measured by a decrease in the number of such critical findings. For mobile applications, the focus shifts to more specific areas, such as analyzing excessive application permissions, user privacy issues, and testing impact on device performance, such as battery and memory consumption. On the other hand, for embedded systems and networks, where research areas are still very limited, security is often measured through operational metrics such as reliability, availability, and system integrity. This confirms that the concept of "security" itself is contextual and has different meanings across platforms. More importantly, the lack of studies on IoT and embedded platforms highlights a significant and urgent research gap to fill.

d) Implications of Findings and Future Research Directions
The findings of this systematic review offer a number of important implications. For industry practitioners, the key guidance is to adopt a hybrid approach to testing, as no single tool is perfect. It is highly recommended to consider IAST to improve efficiency, as well as integrating security practices early on in the software development lifecycle through DevSecOps or Shift-Left. Additionally, practitioners are encouraged to actively use metrics for continuous security monitoring, rather than simply as a tool to find vulnerabilities. For academic researchers, this review highlights some promising future research directions. There is an urgent need to develop

benchmarks and standardized datasets that can be used to fairly compare different tools. Further research is also urgently needed to explore security on non-web platforms, such as IoT and embedded systems. In addition, the utilization of advanced technologies such as AI and Large Language Models (LLM) for intelligent automation in security analysis as well as the development of more contextualized vulnerability prioritization models (considering business impact) are areas of great potential to be explored.

e) Research Limitation
It is important to recognize that this study has some limitations that are natural in a systematic literature review. Firstly, there is a potential publication bias, where the review is limited to studies that have been formally published and are available in English, thus possibly overlooking negative results or relevant studies from other regions. Secondly, there is the possibility of selection bias, where the keywords used may not have captured all relevant studies due to differences in terminology. Lastly, although PRISMA's rigorous methodology has been applied, the process of data interpretation and synthesis is not completely free from researcher subjectivity, which may affect the final results.

5 CONCLUSION

This Systematic Literature Review (SLR) of 40 studies maps the application security assessment landscape with three key findings:

a) Evaluation Metrics
Security is measured using two types of metrics, namely internal code quality metrics (such as complexity) as an indicator of risk, and specific security metrics (such as CVSS) to measure direct impact.

b) Testing Methods
Each method (SAST, DAST, IAST) has a unique impact. Trends show that hybrid approaches, especially those involving IAST, are the most effective for accuracy.

c) Application Focus
Research is predominantly on web applications, while there is a significant research gap on mobile and embedded platforms, where security is also about reliability and availability.

Therefore, developer organizations are advised to integrate a combination of SAST, DAST, and IAST in their SDLC stages.

REFERENCES

Abdulghaffar, K., Elmrabit, N., & Yousefi, M. (2023). Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners. *Computers*, *12*(11). https://doi.org/10.3390/computers12110235

Al Fansha, D., Yusril, M., Setyawan, H., & Fauzan, M. N. (2021). Load Test pada Microservice yang menerapkan CQRS dan Event Sourcing. In *Jurnal Buana Informatika* (Vol. 12, Issue 2).

Allodi, L., Cremonini, M., Massacci, F., & Shim, W. (2020). Measuring the accuracy of software vulnerability assessments: experiments with students and professionals. *Empirical Software Engineering*, *25*(2), 1063–1094. https://doi.org/10.1007/s10664-019-09797-4

Altulaihan, E. A., Alismail, A., & Frikha, M. (2023). A Survey on Web Application Penetration Testing. In *Electronics (Switzerland)* (Vol. 12, Issue 5). MDPI. https://doi.org/10.3390/electronics12051229

Anjum, M., Agarwal, V., Kapur, P. K., & Khatri, S. K. (2020). Two-phase methodology for prioritization and utility assessment of software vulnerabilities. *International Journal of System Assurance Engineering and Management*, *11*, 289–300. https://doi.org/10.1007/s13198-020-00957-0

Anjum, M., Kapur, P. K., Agarwal, V., & Khatri, S. K. (2020). Assessment of software vulnerabilities using best-worst method and two-way analysis. *International Journal of Mathematical, Engineering and Management Sciences*, *5*(2), 328–342. https://doi.org/10.33889/IJMEMS.2020.5.2.027

Aydos, M., Aldan, Ç., Coşkun, E., & Soydan, A. (2022). Security testing of web applications: A systematic mapping of the literature. In *Journal of King Saud University - Computer and Information Sciences* (Vol. 34, Issue 9, pp. 6775–6792). King Saud bin Abdulaziz University. https://doi.org/10.1016/j.jksuci.2021.09.018

Colakoglu, F. N., Yazici, A., & Mishra, A. (2021). Software Product Quality Metrics: A Systematic Mapping Study. *IEEE Access*, *9*, 44647–44670. https://doi.org/10.1109/ACCESS.2021.3054730

Esposito, M., Falaschi, V., Tor, R. ", Rome, V. ", & Falessi, D. (2024). An Extensive Comparison of Static Application Security Testing Tools. In *Proceedings of The 28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024)* (Vol. 1). https://doi.org/10.48550/arXiv.2403.09219

Humayun, M., Jhanjhi, N. Z., Almufareh, M. F., & Khalil, M. I. (2022). Security Threat and Vulnerability Assessment and Measurement in Secure Software Development. *Computers, Materials and Continua*, *71*(2), 5039–5059. https://doi.org/10.32604/cmc.2022.019289

Hussein, A., Azmi, A., & Abas, H. (2025). Software Vulnerability Assessment and Classification Using Machine Learning, Deep Learning and Feature Selection Techniques. *Informatica (Slovenia)*, *49*(17), 95–104. https://doi.org/10.31449/inf.v49i17.5992

Kalouptsoglou, I., Siavvas, M., Ampatzoglou, A., Kehagias, D., & Chatzigeorgiou, A. (2023). Software vulnerability prediction: A systematic mapping study. In *Information and Software Technology* (Vol. 164). Elsevier B.V. https://doi.org/10.1016/j.infsof.2023.107303

Kuncoro, A. W., Informatika, J., Rahma, F., & Jurusan Informatika, M. E. (2022). *Analisis Metode Open Web Application Security Project (OWASP) pada Pengujian Keamanan Website: Literature Review*. https://www.sciencedirect.com

Le, T. H. M., Chen, H., & Babar, M. A. (2022). A Survey on Data-driven Software Vulnerability Assessment and Prioritization. *ACM Computing Surveys*, *55*(5). https://doi.org/10.1145/3529757

Li, J. (2020). Vulnerabilities mapping based on OWASP-SANS: A survey for static application security testing (SAST). *Annals of Emerging Technologies in Computing*, *4*(3), 1–8. https://doi.org/10.33166/AETiC.2020.03.001

Ravindran, U., & Potukuchi, R. V. (2022). A Review on Web Application Vulnerability Assessment and Penetration Testing. *Review of Computer Engineering Studies*, *9*(1), 1–22. https://doi.org/10.18280/rces.090101

Seth, A., Bhattacharya, S., Elder, S., Zahan, N., & Williams, L. (2025). Comparing effectiveness and efficiency of Interactive Application Security Testing (IAST) and Runtime Application Self-Protection (RASP) tools in a large java-based system. *Empirical Software Engineering*, *30*(3), 67. https://doi.org/10.1007/s10664-025-10621-5

Siavvas, M., Kehagias, D., Tzovaras, D., & Gelenbe, E. (2021). A hierarchical model for quantifying software security based on static analysis alerts and software metrics. *Software Quality Journal*, *29*(2), 431–507. https://doi.org/10.1007/s11219-021-09555-0

Tauqeer, O. Bin, Jan, S., Khadidos, A. O., Khadidos, A. O., Khan, F. Q., & Khattak, S. (2021). Analysis of security testing techniques. *Intelligent Automation and Soft Computing*, *29*(1), 291–306. https://doi.org/10.32604/iasc.2021.017260

Tudela, F. M., Higuera, J. R. B., Higuera, J. B., Montalvo, J. A. S., & Argyros, M. I. (2020). On combining static, dynamic and interactive analysis security testing tools to improve owasp top ten security vulnerability detection in web applications. *Applied Sciences (Switzerland)*, *10*(24), 1–26. https://doi.org/10.3390/app10249119