

A Survey on Static and Dynamic Analysis Techniques for Software Security

Sagolsem Akash

Department of AI

FET.Jain Deemed-to-be University

Bengaluru-562112

sagolsemakash98@gmail.com

Dr. K S Arvind

Department of CSE

FET.Jain Deemed-to-be University

Bengaluru-562112

Ks.arvind@jainuniversity.ac.in

Abstract—Software security has become a critical concern due to the rapid growth of complex and connected systems. Vulnerabilities in software can lead to severe consequences, including data breaches, financial loss, and system concession. Static and dynamic analysis ways are extensively used to identify security vulnerabilities during different phases of the software development lifecycle. This survey provides a comprehensive review of static and dynamic analysis ways, including their methodologies, tools, advantages, limitations, and operations in software security. Static analysis focuses on analyzing source code without execution, enabling early discovery of vulnerabilities, while dynamic analysis evaluates software behavior during runtime to identify real-world security issues. The paper also explores hybrid approaches that combine both ways to improve discovery accuracy and content. The survey concludes that integrating static and dynamic analysis within secure development practices is essential for building robust and secure software systems.

Keywords—Software Security, Static Analysis, Dynamic Analysis, Vulnerability Discovery, Secure Software Development, SAST, DAST

I. INTRODUCTION

In the modern digital era, software systems are increasingly used in critical disciplines such as healthcare, finance, defence, and cloud computing. As software complexity grows, so does the threat of security vulnerabilities. These vulnerabilities can be exploited by attackers, leading to serious consequences such as data theft, system failures, and financial losses.

To address these challenges, software security testing ways have evolved significantly. Among them, static

and dynamic analyses are two fundamental approaches extensively used in vulnerability discovery. Static analysis examines the source code without executing the program, enabling early discovery of defects. In contrast, dynamic analysis evaluates software during execution, relating vulnerabilities that appear only at runtime.

This survey aims to provide a detailed comparison of these ways and highlight their significance in modern software security practices.

II. BACKGROUND AND OVERVIEW OF SOFTWARE TESTING

Software security analysis ways are broadly classified into:

- 1) Static Analysis (SAST): Static Application Security Testing (SAST) involves analyzing source code, bytecode, or binaries without executing the program. It helps detect vulnerabilities during the development phase.
- 2) Dynamic Analysis (DAST): Dynamic Application Security Testing (DAST) evaluates the software during runtime by simulating attacks and monitoring system behavior.

III. STATIC ANALYSIS TECHNIQUES

Definition: Static analysis is the process of examining source code or binaries without executing the program to detect vulnerabilities and coding errors.

Ways: Data Flow Analysis, Control Flow Analysis, Pattern Matching, Symbolic Execution.

Advantages: Early discovery during development, No need for execution, Scalable for large codebases.

Disadvantages: High false positives, Limited understanding of runtime behavior, Cannot detect context-specific vulnerabilities.

IV. DYNAMIC ANALYSIS TECHNIQUES

Definition: Dynamic analysis evaluates software behavior during execution to detect vulnerabilities that occur in real-world scenarios.

Ways: Penetration Testing, Fuzz Testing, Runtime Monitoring, Behavioral Analysis.

Advantages: Detects runtime vulnerabilities, Real-world testing scenarios, Lower false positives.

Disadvantages: Time-consuming, Requires execution environment, Limited code coverage.

V. COMPARATIVE ANALYSIS

Feature	Static Analysis	Dynamic Analysis
Execution	No execution needed	Requires execution
Discovery Phase	Early (development)	Late (runtime)
Coverage	High (entire code)	Limited (executed paths)
Accuracy	May produce false positives	More accurate results
Speed	Fast	Slower

VI. HYBRID ANALYSIS

Hybrid analysis combines static and dynamic ways to improve discovery accuracy and content. Benefits include reduced false positives, advanced vulnerability discovery, and comprehensive security coverage.

VII. CHALLENGES

1) High False Positives in Static Analysis: Static tools often lack full contextual understanding of runtime conditions, leading to unnecessary alerts.

2) Limited Coverage in Dynamic Analysis: Dynamic analysis can only analyze executed portions of code, leaving untested paths unexplored.

3) Tool Integration Issues: Integrating multiple tools into a unified workflow (DevSecOps) can be challenging due to incompatible formats.

4) Scalability Challenges: Large-scale systems face performance and resource limitations during both static and dynamic analysis.

VIII. FUTURE TRENDS

Artificial intelligence and machine learning are becoming important in software security testing. AI-based systems can learn from large code datasets and identify patterns that traditional rule-based tools may miss. Models such as CodeBERT and GPT-based systems can help understand code semantics and suggest possible fixes.

DevSecOps integration is another major trend. Instead of performing security testing only at the end of development, organizations are shifting security left by integrating testing into continuous integration and continuous deployment pipelines. This helps identify vulnerabilities earlier and reduces remediation cost.

Cloud security testing is becoming more important because many applications are deployed on cloud platforms. Security testing now includes cloud configuration, identity and access management, container security, serverless functions, and infrastructure-as-code analysis.

Automated penetration testing is also growing. Intelligent tools can simulate attacks, generate payloads, test APIs, and continuously monitor applications for weaknesses. While human expertise is still necessary, automation can improve speed and consistency.

IX. EMERGING RESEARCH DIRECTIONS

- 1) AI-Driven Vulnerability Discovery: Leveraging ML and DL (e.g., CodeBERT, GPT) to identify complex and unknown vulnerabilities by understanding code semantics.
- 2) Automated Penetration Testing: Using intelligent tools and reinforcement learning to simulate real-world attacks continuously.
- 3) Blockchain-Based Security: Using decentralized, tamper-evident ledgers for secure data storage, identity management, and audit trails.

X. DISCUSSION

Static and dynamic analysis techniques play complementary roles in software security. Static analysis is useful for early detection and broad code coverage, while dynamic analysis is useful for validating real runtime behavior. When used together, they provide a more complete understanding of application security.

For academic and industrial projects, the best approach is to integrate security testing throughout the development process. Developers should use static analysis during coding, testers should use dynamic analysis during testing, and security teams should review high-risk findings. This layered approach reduces the chance that vulnerabilities will reach production.

Security testing should not be treated as a one-time activity. Applications change frequently due to new features, dependency updates, configuration changes, and deployment modifications. Continuous testing is therefore necessary to maintain security over time.

XI. CONCLUSION

This survey presented a detailed overview of static and dynamic analysis techniques for software security. Static analysis helps identify vulnerabilities early by examining code without execution, while dynamic analysis detects runtime issues by testing the application during execution. Each technique has advantages and limitations, and neither can fully replace the other.

A hybrid approach provides better coverage, improved accuracy, and stronger vulnerability prioritization. As software systems become more complex, integrating static and dynamic analysis into DevSecOps pipelines will become increasingly important. Future improvements in AI-based testing, automated penetration testing, and cloud security analysis will further strengthen secure software development practices.

Overall, secure software development requires a combination of automated tools, human expertise, continuous testing, and developer awareness. By using static and dynamic analysis effectively, organizations can reduce security risks and build more trustworthy software systems.

REFERENCES

- 1) M. Pistoia et al., 'A Survey of Static Analysis Styles,' IBM Systems Journal, 2007.
- 2) A. Damodaran et al., 'Static vs Dynamic Malware Analysis,' IEEE, 2022.
- 3) M. Shaon et al., 'Software Vulnerability Discovery,' Electronics, 2025.
- 4) A. Kaur, 'Static Code Analysis Tools Study,' ScienceDirect, 2020.
- 5) T. Minh Thanh, 'Static and Dynamic Analysis Overview,' 2024.
- 6) StackHawk, 'Dynamic Analysis Companion,' 2025.
- 7) VFunction, 'Static vs Dynamic Law Analysis,' 2025.
- 8) Kiuwan, 'Static vs Dynamic Testing Companion.'
- 9) IEEE, 'Software Security Testing Standards.'
- 10) G. Myers, 'Software Testing,' Wiley, 2011.
- 11) R. Pressman, 'Software Engineering,' 2014.
- 12) B. Beizer, 'Software Testing Ways.'
- 13) ISO/IEC 29119 Software Testing Standard.
- 14) OWASP, 'Application Security Testing Guide,' 2023.
- 15) NIST, 'Secure Software Development Framework,' 2022.

- 16) Research on Hybrid Malware Analysis, 2026.
- 17) Check on Malware Analysis Ways.
- 18) Dynamic Testing Generalities, IEEE, 2020.