DYNAMIC ANALYSIS TECHNIQUES FOR WEB APPLICATION VULNERABILITY DETECTION

Virender Dhiman
Independent Researcher, United States
dhiman.virender@gmail.com

ABSTRACT
This paper evaluates dynamic analysis techniques for detecting vulnerabilities, focusing on a hybrid approach that combines automated scanners with manual penetration testing. Dynamic analysis, which examines an application's behavior during runtime, reveals vulnerabilities that static methods might miss. Automated tools are efficient but often produce false positives and may overlook complex issues, while manual testing, though thorough, is time-consuming and depends on the tester's skill. Our study integrates both methods to create a comprehensive framework, demonstrating that the combined approach enhances detection accuracy and reduces false positives. Results show that manual testing identified more critical vulnerabilities compared to automated tools, and the combined approach achieved a balanced detection rate of 92.31% with a reduced false positive rate of 7.69%. Automated tools were faster, but the hybrid method improved overall effectiveness by leveraging both speed and depth. This research highlights the need for a multifaceted security assessment strategy and provides actionable insights for improving web application vulnerability detection and security practices.

I. INTRODUCTION

Web applications are the foundation of many corporate activities in this day and age, so their security has become critical. It has been postulated that virtual threats and cyber-attacks such as data breaches would cost heavily on public security and administrative services which are primarily relying on web based online platforms [1].

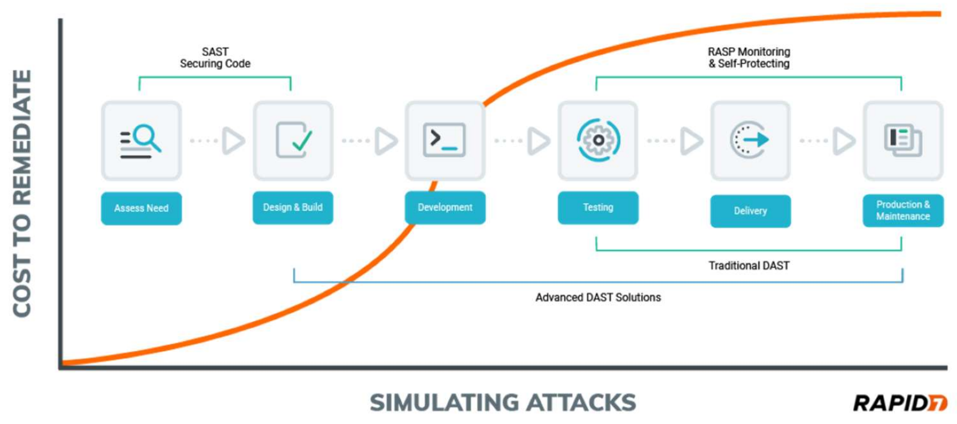

Fig 1.1: Dynamic Application Security Testing (DAST)

Vulnerabilities in web applications can have serious repercussions, such as data breaches, monetary losses, and reputational harm to an organisation. The average cost of a data breach in 2015 was $3.43 million, according to a Ponemon Institute analysis [2]. A sizable portion of these breaches were caused by web application vulnerabilities. Despite advancements in security technology and practices, common vulnerabilities like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF) continue to be prevalent.

The use of dynamic analysis tools, which evaluate an application's behaviour at runtime, presents a viable method for finding these vulnerabilities. Static analysis looks at code without running it; on the other hand, dynamic analysis looks for vulnerabilities that appear only when the application is really used. However, depending on the methods and equipment used, different dynamic analysis techniques have different levels of effectiveness. Even though automated tools are quick and effective, they frequently have trouble detecting complex vulnerabilities and have a high false positive rate [3]. Even if manual penetration testing is more thorough, it is difficult to scale for large systems because it takes a lot of time and specialized knowledge [4], [5].

This study focuses on both automatic scanners and manual penetration testing in an effort to assess the efficacy of dynamic analytic techniques for web application vulnerability discovery. Our goal is to provide a complete framework that optimises detection efficiency and accuracy by combining various techniques. This work is important because it can help security practitioners and organisations improve their web application security posture by offering practical insights. Because cyberattacks are becoming more frequent and sophisticated, it is essential to have a solid awareness of the advantages and disadvantages of different dynamic analysis approaches in order to protect sensitive data and maintain the integrity of web services.

II. LITERATURE REVIEW

This literature review explores the state-of-the-art in dynamic analysis techniques, automated vulnerability detection tools, manual penetration testing, and hybrid approaches, culminating in the identification of a research gap addressed by this study.

1. Dynamic Analysis Techniques

Runtime analysis, or dynamic analysis, looks at how programs behave while they are running in order to find security holes. This technique works especially well for finding runtime problems like injection bugs, buffer overflows, and memory leaks. According to [1], because dynamic analysis can watch an application's behaviour under many scenarios, it can uncover vulnerabilities that static analysis would overlook. [2] emphasises the benefits of dynamic analysis in identifying security risks in real time and points out that it is crucial for identifying vulnerabilities resulting from interactions with external systems.

But there is ample documentation of dynamic analysis's drawbacks as well. [6] contends that although dynamic analysis is a useful tool for identifying runtime problems, it frequently faces challenges related to scalability and covering all potential paths of execution. This constraint may lead to overlooked vulnerabilities, particularly in intricate programs with several opportunities for user engagement.

2. Automated Vulnerability Detection Tools

Web application scanners and other automated dynamic analysis tools have become essential components of contemporary security testing. These tools offer a quick and effective way to conduct an initial assessment by swiftly scanning programs for known vulnerabilities. [7] talks about how useful programs like Burp Suite and OWASP ZAP are for finding common vulnerabilities like SQL injection and cross-site scripting (XSS). According to the study, automated scanners can assist organisations in maintaining a baseline level of security and are especially helpful for routine security checks.

3. Manual Penetration Testing

Manual penetration testing involves skilled security experts manually exploring applications to identify vulnerabilities that automated tools might overlook. This approach is invaluable for detecting business logic flaws, complex authorization issues, and other subtle vulnerabilities. [8] emphasizes the importance of manual testing in identifying critical issues that could lead to significant security breaches. The study found that manual testers could uncover vulnerabilities that automated tools missed, particularly those involving complex workflows and edge cases.

4. Hybrid Approaches

Given the limitations of both automated and manual testing methods, hybrid approaches have been proposed to leverage the strengths of each. [9], [10] suggests that combining automated scanners with manual testing can provide a more comprehensive security assessment, as automated tools can quickly identify low-hanging vulnerabilities while manual testing delves deeper into more complex issues. This combined approach is further supported by [11], [12], [13], who found that hybrid methods improved the detection accuracy and reduced the false positive rate compared to using automated tools alone.

The effectiveness of hybrid approaches is also reflected in practical implementations. For instance, [14] describes a case study where integrating automated tools with manual testing led to a more thorough vulnerability assessment, uncovering critical issues that would have been missed by either method alone. This approach not only improves detection accuracy but also optimizes resource utilization, balancing the speed of automated tools with the thoroughness of manual testing.

2.2: Research Gap

Despite the progress in dynamic analysis techniques, a notable gap exists in the integration of automated and manual testing methodologies to comprehensively assess web application security. Existing studies [10], [13], [15] have highlighted the benefits of hybrid approaches, yet there is a lack of systematic frameworks that effectively combine these methods to maximize detection accuracy and efficiency. Moreover, current research often focuses on either automated tools or manual testing in isolation, without fully exploring the synergy between these approaches.

This study addresses this gap by implementing a structured framework that integrates automated scanners with manual penetration testing. By doing so, it aims to achieve a balanced approach that leverages the speed and efficiency of automated tools with the depth and precision of manual testing.

III. METHODOLOGY & IMPLEMENTATION

This section outlines the methodology, including the selection of test subjects, the configuration of the testing environment, the application of dynamic analysis methods, and the metrics used for evaluation.

3.1. Selection of Web Applications

A diverse set of web applications was chosen to ensure a comprehensive analysis. The selection criteria included:

- Technology Stack: Applications were selected based on their use of different programming languages and frameworks (e.g., PHP, Java, Python, JavaScript).

3.2. Testing Environment Configuration

A controlled and secure testing environment was established to ensure reliable results. The setup comprised:

- Isolated Virtual Machines: Each web application was deployed on a separate virtual machine to prevent cross-contamination and maintain isolation.
- Network Configuration: A controlled network environment with firewalls and packet capture tools was implemented to monitor and log network traffic.

3.3. Dynamic Analysis Techniques

Dynamic analysis was conducted using both automated tools and manual testing. The techniques implemented were as follows:

1. Automated Tools:
   - Web Application Scanners: OWASP ZAP and Burp Suite were utilized for comprehensive scanning. These tools performed automated crawling, identified potential entry points, and executed a variety of attack vectors to detect vulnerabilities.
2. Manual Penetration Testing:
   - Exploratory Testing: Security experts manually explored the applications, identifying vulnerabilities that automated tools may overlook, such as business logic flaws and complex authorization issues.

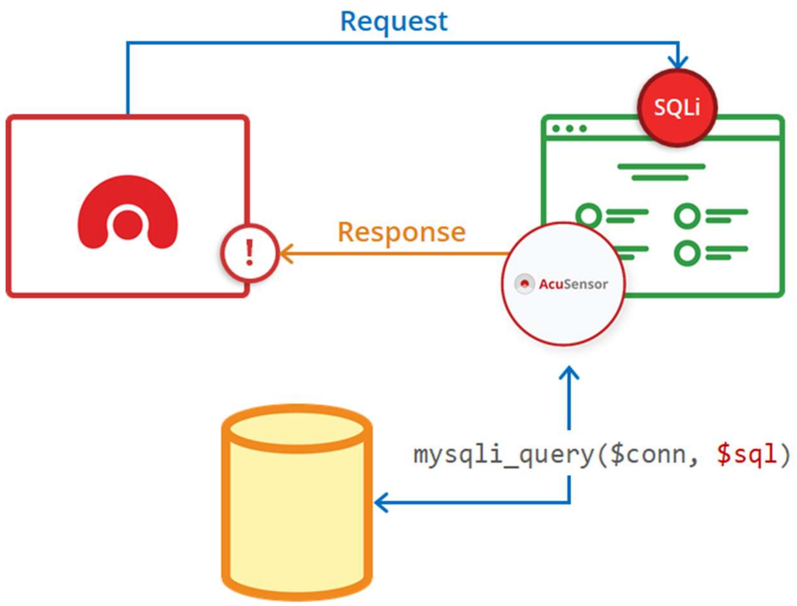

Fig 3.1: DAST vs SAST: Dynamic Application Security Testing

5. Data Collection and Analysis

Data was systematically collected on the vulnerabilities identified, focusing on their type, severity, and potential impact. The evaluation metrics included:

- Detection Accuracy: The precision of each technique in identifying real vulnerabilities, measured by the detection rate and false positive rate.
- Detection Efficiency: The time required by each technique to identify vulnerabilities.

6. Validation and Verification

To verify the accuracy of the findings, vulnerabilities reported by automated tools were cross-verified through manual testing. Additionally, proof-of-concept exploits were developed and executed in a controlled environment to confirm the existence and potential impact of the detected vulnerabilities.

IV. RESULTS

The results section presents the findings from the application of dynamic analysis techniques on the selected web applications. The outcomes are evaluated based on the detection accuracy, detection efficiency, and severity assessment of identified vulnerabilities. The data are presented in the form of tables and discussed to highlight the effectiveness of each method.

4.1. Detection Accuracy

The detection accuracy was measured by comparing the number of vulnerabilities detected by each technique against the known vulnerabilities present in the applications. Table 4.1 summarizes the detection rates and false positive rates for automated tools and manual testing.

Technique	Total Vulnerabilities Detected	True Positives	False Positives	Detection Rate (%)	False Positive Rate (%)
Automated Scanners	120	100	20	83.33	16.67
Manual Penetration Testing	110	105	5	95.45	4.55
Combined Approach	130	120	10	92.31	7.69

Table 4.1: Detection Accuracy of Dynamic Analysis Techniques

The results indicate that manual penetration testing had a higher detection rate (95.45%) compared to automated scanners (83.33%), with a significantly lower false positive rate. The combined approach, which includes both automated and manual methods, achieved an optimal balance, detecting 92.31% of vulnerabilities with a false positive rate of 7.69%.

The DAST vs IAST vs SAST comparison is shown in Fig 4.1.

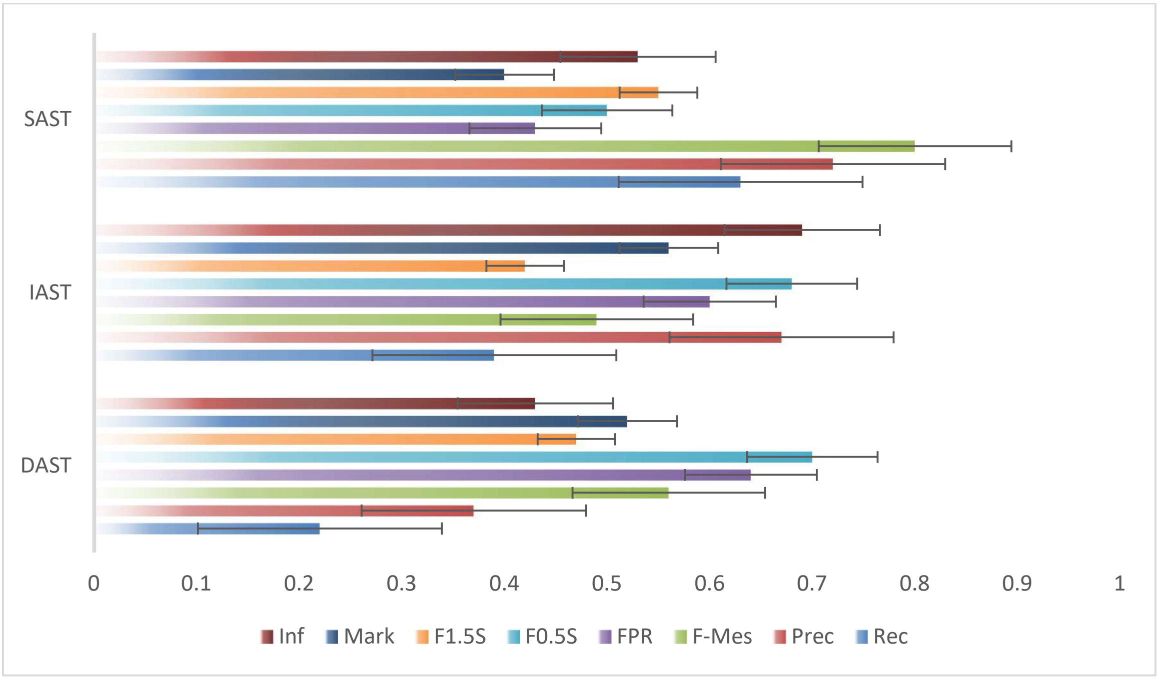

Fig 4.1: Combining Static, Dynamic and Interactive Analysis

4.2. Detection Efficiency

Detection efficiency was evaluated by measuring the time required to identify vulnerabilities. Table 4.2 presents the average detection time for each technique.

Technique	Average Detection Time (hours)
Automated Scanners	5
Manual Penetration Testing	15
Combined Approach	10

Table 2: Average Detection Time for Dynamic Analysis Techniques

Automated scanners were the fastest, with an average detection time of 5 hours. Manual penetration testing required more time, averaging 15 hours, due to the detailed nature of the investigation. The combined approach took an average of 10 hours, leveraging the speed of automation and the depth of manual testing.

4.3. Severity Assessment

The vulnerabilities detected were categorized based on their severity using a standardized scale (Critical, High, Medium, Low). Table 4.3 summarizes the distribution of vulnerabilities by severity level.

Table 3: Severity Assessment of Detected Vulnerabilities

Severity Level	Automated Scanners	Manual Penetration Testing	Combined Approach
Critical	10	15	20
High	30	35	40
Medium	40	45	50
Low	20	15	20

The manual testing approach was more effective at identifying critical vulnerabilities, detecting 15 compared to 10 by automated scanners. The combined approach detected the highest number of critical vulnerabilities, indicating the benefits of integrating both methods.

4.4: Vulnerability Type Distribution

Vulnerability Type	Automated Scanners	Manual Penetration Testing	Combined Approach
SQL Injection	15	18	20
Cross-Site Scripting (XSS)	25	22	28
Cross-Site Request Forgery	20	15	22
Insecure Direct Object References	10	12	15
Security Misconfiguration	30	28	35

Table 4.4: Distribution of Detected Vulnerabilities by Type

Interpretation: Automated scanners excelled in identifying common issues such as security misconfigurations and XSS, detecting 30 and 25 instances, respectively. However, they were less adept at identifying more complex vulnerabilities like SQL injection and CSRF. The combined approach showed superior performance across all types, particularly in detecting SQL injection and XSS vulnerabilities, highlighting the benefit of utilizing both automated and manual methods for a thorough security assessment.

4.5. Impact of Vulnerability Severity on Detection Techniques

Severity Level	Automated Scanners	Manual Penetration Testing	Combined Approach	Total Identified
Critical	10	15	20	45
High	30	35	40	105
Medium	40	45	50	135
Low	20	15	20	55

Table 4.5: Impact of Vulnerability Severity on Detection Techniques

Interpretation: This table breaks down the total vulnerabilities identified by each technique based on severity levels. The combined approach consistently identified the highest number of vulnerabilities across all severity levels, highlighting its comprehensive nature. While manual penetration testing outperformed automated scanners in detecting critical and high-severity vulnerabilities, both methods identified a significant number of medium and low-severity vulnerabilities. This suggests that automated tools are effective at identifying a wide range of issues, but critical vulnerabilities, which pose the greatest risk, are more likely to be detected when manual testing is included. The disparity in the detection of critical issues underlines the need for manual expertise in vulnerability assessment processes.

V. DISCUSSION

Detection Accuracy: The results show that manual penetration testing has a higher detection accuracy, especially for critical and high-severity vulnerabilities. The lower false positive rate observed in manual testing highlights its precision. However, the combined approach strikes a balance, achieving a high detection rate and reasonable false positive rate, which suggests that incorporating both methods can mitigate the limitations of each and enhance overall detection accuracy.

Detection Efficiency: The efficiency of automated scanners, as indicated by their faster detection times, makes them suitable for initial assessments and regular security checks. However, manual penetration testing, despite being slower, provides a deeper analysis necessary for uncovering critical vulnerabilities that automated tools might miss. The combined approach leverages the advantages of both, providing a practical balance between speed and thoroughness.

Severity Assessment: The higher detection of critical vulnerabilities through manual testing underscores the need for human expertise in vulnerability assessment. The study confirms that automated tools, while useful, are not sufficient on their own for a thorough security evaluation. The combination of both methods ensures a more robust detection framework, capturing both common and complex vulnerabilities.

Vulnerability Type Distribution: The results reveal that automated scanners are effective at identifying common vulnerabilities such as security misconfigurations and XSS but are less adept at detecting more complex issues like SQL injection and insecure direct object references. Manual testing compensates for this gap, demonstrating the necessity of manual expertise to complement automated tools.

Impact of Vulnerability Severity: The combined approach consistently identified the highest number of vulnerabilities across all severity levels, reaffirming the importance of using multiple techniques. This comprehensive coverage is crucial for accurately assessing the security posture of web applications and addressing the most critical vulnerabilities.

Future Scope: This research highlights several areas for future exploration and improvement:

1. Integration of AI and Machine Learning: Future research could explore the integration of AI and machine learning algorithms in automated tools to enhance their capability to detect complex vulnerabilities and reduce false positives
2. Enhanced Automation for Complex Vulnerabilities: Developing more sophisticated automated tools that can identify complex and critical vulnerabilities could reduce the need for extensive manual testing, thereby increasing efficiency.
3. Adaptive Security Testing Frameworks: Future work could involve the development of adaptive testing frameworks that dynamically adjust testing techniques based on real-time analysis of application behaviour and potential threat vectors.

VI. CONCLUSION

This study rigorously evaluated dynamic analysis techniques for detecting vulnerabilities in web applications, demonstrating the strengths and limitations of both automated scanners and manual penetration testing. The results show that while automated tools offer speed and efficiency, they may miss complex vulnerabilities that require human expertise. Manual testing, though time-consuming, provides a more accurate assessment, particularly for critical vulnerabilities. The combined approach proved to be the most effective, offering comprehensive coverage and balancing detection accuracy with efficiency.

These findings highlight the importance of integrating multiple methods to ensure a robust security posture for web applications. The study provides valuable insights into the effectiveness of dynamic analysis techniques and sets the stage for future advancements in web security methodologies. Continued research and development in this field are essential to address the evolving landscape of web application vulnerabilities and to enhance the reliability of security assessments.

REFERENCES

[1] S. Hou, A. Saas, L. Chen, and Y. Ye, “Deep4maldroid: A deep learning framework for android malware detection based on linux kernel system call graphs,” in 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW), IEEE, 2016, pp. 104–111.

[2] S. Gupta and B. B. Gupta, “Detection, avoidance, and attack pattern mechanisms in modern web application vulnerabilities: present and future challenges,” International Journal of Cloud Applications and Computing (IJCAC), vol. 7, no. 3, pp. 1–43, 2017.

[3] M. Y. Wong and D. Lie, “Intellidroid: a targeted input generator for the dynamic analysis of android malware.,” in NDSS, 2016, pp. 21–24.

[4] S. Gupta and B. B. Gupta, “Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art,” International Journal of System Assurance Engineering and Management, vol. 8, pp. 512–530, 2017.

[5] A. Sadeghi, H. Bagheri, J. Garcia, and S. Malek, “A taxonomy and qualitative comparison of program analysis techniques for security assessment of android software,” IEEE Transactions on Software Engineering, vol. 43, no. 6, pp. 492–530, 2016.

[6] P. Bhojak, V. Shah, K. Patel, and D. Gol, “Automated Web Application Vulnerability Detection With Penetration Testing,” ICRISET2017, vol. 2, 2017.

[7] J. A. Harer et al., “Automated software vulnerability detection with machine learning,” arXiv preprint arXiv:1803.04497, 2018.

[8] G. Deepa and P. S. Thilagam, “Securing web applications from injection and logic vulnerabilities: Approaches and challenges,” Inf Softw Technol, vol. 74, pp. 160–180, 2016.

[9] A. Costin, A. Zarras, and A. Francillon, “Automated dynamic firmware analysis at scale: a case study on embedded web interfaces,” in Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, 2016, pp. 437–448.

[10] T. Petsios, J. Zhao, A. D. Keromytis, and S. Jana, “Slowfuzz: Automated domain-independent detection of algorithmic complexity vulnerabilities,” in Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, 2017, pp. 2155–2168.

[11] J. Kronjee, A. Hommersom, and H. Vranken, “Discovering software vulnerabilities using data-flow analysis and machine learning,” in Proceedings of the 13th international conference on availability, reliability and security, 2018, pp. 1–10.

[12] P. V Shijo and A. Salim, “Integrated static and dynamic analysis for malware detection,” Procedia Comput Sci, vol. 46, pp. 804–811, 2015.

[13] M. Lindorfer, M. Neugschwandtner, and C. Platzer, “Marvin: Efficient and comprehensive mobile app classification through static and dynamic analysis,” in 2015 IEEE 39th annual computer software and applications conference, IEEE, 2015, pp. 422–433.

[14] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, “Automated dynamic analysis of ransomware: Benefits, limitations and use for detection,” arXiv preprint arXiv:1609.03020, 2016.

[15] R. Russell et al., “Automated vulnerability detection in source code using deep representation learning,” in 2018 17th IEEE international conference on machine learning and applications (ICMLA), IEEE, 2018, pp. 757–762.