International Journal of Engineering Research and Development
e- ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 22, Issue 2 (February 2026), PP 157-160

A Survey on Runtime Application Self Protect Techniques for Securing Web Applications

[1]Ravi Shankar M, *Department of Computer Science with Cybersecurity, Dr. N.G.P Arts and Science College, India,*
[2]Malathi V, *Associate Professor, Department of Computer Science with Cybersecurity, Dr. N.G.P Arts and Science College, Coimbatore, India.*

Date of Submission: 14-02-2026 Date of acceptance: 28-02-2026

Abstract
With the rapid growth of web applications, security threats suchtt as SQL Injection, Cross-Site Scripting, and logic-based attacks have become increasingly common. Traditional security mechanisms, including network firewalls and Web Applications Firewalls (WAFs), primarily operate outside the application and rely on static rule-based detection. These approaches lack runtime visibility and fail to protect applications from zero-day and logic-level attacks. Runtime Application Self-Protection (RASP) has emerged as a promising solution by embedding security controls directly within the application runtime. This survey paper reviews the concept of RASP, its architecture, detection techniques, enforcement mechanisms, advantages over traditional security systems, challenges, and future research directions.

Keywords: Runtime Application Self-Protection, RASP, Web Security, Application Security, Survey Paper, Intrusion Detection

I. Introduction and Motivation

Web applications form the backbone of modern digital services, supporting online banking, healthcare systems, e-commerce platforms, and cloud-based applications. As these applications handle sensitive data and critical operations, they have become primary targets for cyber attackers. Despite advancements in secure development practices, vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and logic-based attacks remain prevalent.

Traditional security approaches such as network firewalls and WAFs attempt to protect applications by filtering incoming traffic. However, these mechanisms operate externally and lack awareness of application execution flow and internal logic. As a result, attackers can bypass perimeter defences using obfuscation techniques or exploit logical flaws

Runtime Application Self-Protection (RASP) addresses these limitations by integrating security mechanisms directly into the application runtime. By monitoring application behaviour during execution, RASP enables context-aware detection and real-time attack prevention. This survey aims to analyse existing RASP techniques and evaluate their effectiveness in securing modern web applications.

In recent years, the shift toward microservices, cloud-native architectures, and DevOps practices has significantly increased the complexity of web application environments. Applications are no longer monolithic but consist of multiple interconnected services, APIs, and third-party integrations. This expanded attack surface has made traditional perimeter-based security models insufficient.

Attackers increasingly exploit business logic flaws rather than simple input validation errors. Such attacks cannot be detected by signature-based tools, as they require deep understanding of application context. RASP provides this contextual awareness by operating inside the application runtime, enabling precise detection and prevention of malicious behaviour at the exact point of exploitation.

Furthermore, regulatory requirements such as GDPR, HIPAA, and PCI-DSS mandate stronger runtime protection and monitoring mechanisms. RASP aligns well with these requirements by offering real-time visibility, detailed audit logs, and enforcement capabilities that complement secure software development practices

II. Existing Security Mechanisms and Their Limitations

Traditional application security relies heavily on perimeter-based defences.

Network Firewalls

Network firewalls operate at the network and transport layers, filtering traffic based on IP addresses, ports, and protocols. While effective against network-level threats, they cannot analyse application-level payloads or logic.

Web Application Firewalls (WAFs)
WAFs operate at the application layer and inspect HTTP requests to detect common attacks. However, WAFs rely on static rules and signatures, making them vulnerable to false positives and ineffective against zero-day and logic-based attacks. Additionally, WAFs lack visibility into how inputs are processed within the application.
These limitations have motivated the development of in-application security solutions such as RASP.

Comparison of Traditional Security Mechanisms
Traditional security follow a defence-in-depth approach; however, each layer has inherent blind spots. Network firewalls cannot inspect encrypted application payloads, while WAFs operate without execution context. Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) focus on traffic patterns but are ineffective against application logic misuse.
In contrast, RASP provides protection from within the application, enabling it to observe function calls, database queries, and execution paths. This internal visibility allows RASP to detect attacks that bypass external defences, making it a complementary and, in some cases, superior solution to traditional security mechanisms.

III. Runtime Application Self-Protection: Architecture and Classification

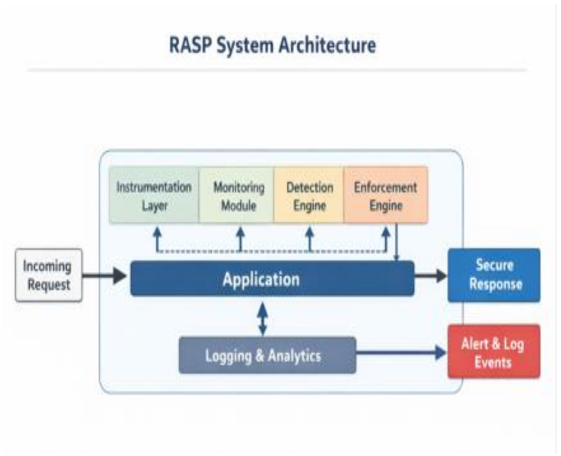
Runtime Application Self-Protection is a security paradigm that embeds monitoring and enforcement capabilities directly within the application.

RASP Architecture
A typical RASP system consists of:

- Instrumental layer
- Monitoring module
- Detection engine
- Enforcement engine
- Logging and analytics module

This architecture ensures complete mediation and runtime visibility.

RASP System Architecture



Classification of RASP Systems
RASP systems can be classified based on their integration approach:

- **Middleware-Based RASP:**
  Integrated at the framework level; lightweight and easy to deploy.
- **Runtime Instrumentation-Based RASP:**
  Uses bytecode or runtime instrumentation for deeper visibility.
- **Agent-Based RASP:**
  Security agents monitor execution alongside the application process.

Each approach offers a trade-off between visibility and performance overhead.
Instrumentation and Runtime Monitoring
RASP systems instrument application runtimes using language-specific hooks, bytecode instrumentation, or middleware interception. This instrumentation enables RASP to monitor sensitive operations such as database queries, file access, authentication logic, and API calls.

Monitoring at runtime ensures that security decisions are made using real execution context rather than assumptions based on request patterns. This against advanced attacks that manipulate application state or exploit logic vulnerabilities.

Deployment Models
RASP solutions can be deployed in various environments including on-premise servers, cloud-based platforms, containerized applications, and serverless architectures. Modern RASP tools are designed to integrate seamlessly with CI/CD pipelines, allowing security controls to be deployed alongside application code.

IV. Detection and Enforcement Techniques in RASP

RASP systems employ multiple detection strategies to identify malicious behaviour.

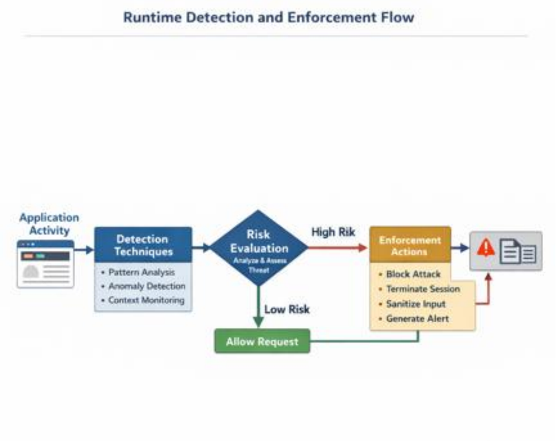
Detection Techniques

- **Pattern-Based Detection:** Uses predefined signatures to detect known attacks.
- **Heuristic-Based Detection:** Identifies anomalies such as abnormal payload size and entropy.
- **Context-Aware Detection:** Analyses execution flow and application logic to detect misuse.

Enforcement Mechanisms
Once an attack is detected, RASO enforces security decisions during execution:

- Blocking malicious requests
- Terminating execution
- Input sanitization
- Logging and alerting

Runtime Detection and Enforcement Flow



Runtime enforcement ensures attacks are blocked before exploitation occurs.

Risk Scoring and verdict Classification
RASP systems typically use a risk-based decision model to classify incoming requests. Detection engines assign scores based on pattern matches, he

5. Challenges, Future Research Directions
Challenges and Limitations
Despite its advantages, RASP faces several challenges:

- Runtime performance overhead
- Integration complexity
- Platform dependency
- Configuration and tunning difficulties

Future Research Directions
Future work in RASP includes:

- Machine learning-based runtime detection
- Cloud-native and microservices-aware RASP
- Integration with SIEM platforms
- Automated policy generation
- Standardization of RASP frameworks

V. Conclusion

This survey reviewed Runtime Application Self-Protection as an effective approach for securing modern web application. By embedding security within the application runtime, RASP overcomes the limitations of traditional perimeter-based defences. Although challenges remain, RASP represents a significant advancement in application security and a promising area for future research.

References

[1]. M. Coates and J. Williams, "Runtime Application Self-Protection (RASP): A New Paradigm," *OWASP Foundation*, 2014.

[2]. J. Williams and J. Manico, "RASP: Protecting Applications at Runtime," *IEEE Security & Privacy Magazine*, vol. 13, no. 6, pp. 80–83, 2015.

[3]. N. Antunes and M. Vieira, "Defending Against Web Application Vulnerabilities," *IEEE Computer*, vol. 45, no. 2, pp. 66–72, 2012.

[4]. G. McGraw, *Software Security: Building Security In*, Addison-Wesley Professional, 2006.

[5]. D. Balzarotti, M. Cova, and G. Vigna, "ClearShot: Eavesdropping on Keyboard Input from Video," *IEEE Symposium on Security and Privacy*, 2008.

[6]. A. Gupta and P. Gupta, "Web Application Security: Threats and Countermeasures," *International Journal of Computer Applications*, vol. 97, no. 1, pp. 15–21, 2014.

[7]. OWASP Foundation, "OWASP Top 10 – Web Application Security Risks," 2021. [Online]. Available: https://owasp.org/www-project-top-ten/

[8]. S. Axelsson, "Intrusion Detection Systems: A Survey and Taxonomy," *Technical Report*, Chalmers University of Technology, 2000.

[9]. M. Almorsy, J. Grundy, and A. S. Ibrahim, "Collaboration-Based Cloud Computing Security Management Framework," *IEEE International Conference on Cloud Computing*, 2011.

[10]. R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," *IEEE Symposium on Security and Privacy*, 2010.

[11]. A. Sabelfeld and A. Myers, "Language-Based Information-Flow Security," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 1, pp. 5–19, 2003