IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154153

# CyberWatch: A Real-Time Intelligent Web Security Monitoring and Attack Detection Platform

[1]Drupad Dhamdhere, [2]Shardul Pharande, [3]Aashish Kondgire, [4]Sakshi Phutane,

[5]Prof. Supriya Jagtap

[1]B.E. Student, [2]B.E. Student, [3]B.E. Student, [4]B.E. Student, [5]Assistant Professor

[1,2,3,4,5]Department of Information Technology,

[1,2,3,4,5]PES Modern College of Engineering, Pune, India

**Abstract:** CyberWatch is a real-time web security monitoring platform built to give defenders immediate visibility into web attacks. Implemented on Spring Boot, it inspects incoming requests and applies modular detectors for SQL injection, brute-force abuse, XSS, command injection, DDoS, CSRF, SSRF, malicious file upload, XXE, and LDAP injection. The implementation also includes log-injection and directory-traversal checks for broader operational coverage. All primary attack categories in project scope were implemented and validated using controlled test traffic. The platform captures contextual evidence, computes a weighted risk score, and streams prioritized alerts to a WebSocket dashboard for faster triage. Experimental results show a strong precision-recall balance with low latency overhead, indicating practical suitability for academic environments and small-to-mid production deployments.

**Keywords:** Web application security, attack detection, Spring Boot, real-time dashboard, cyber threat monitoring.

## 1. INTRODUCTION

Web applications now host identity workflows, payment actions, and sensitive organizational data in one exposed layer. This has made them a primary target for automated exploitation campaigns and repeated probing attempts. In many small and medium deployments, detection is still reactive: logs are inspected after service degradation, and alerting is distributed across disconnected tools. Delayed visibility increases response time and allows adversaries to retry attacks with minimal resistance [1], [2].CyberWatch was developed to close this operational gap by observing threats at request time rather than only during post-incident review. The platform captures what happened, where it happened, and how often it happened, then converts those signals into actionable alerts for security operators.The contributions of this paper are three-fold. First, it presents a modular detector framework that supports multiple web attack categories without tightly coupling all logic into one filter. Second, it introduces an explainable severity model that combines detector confidence with endpoint criticality and request context. Third, it demonstrates that a human-readable dashboard with WebSocket updates can reduce triage effort when compared to raw log-only workflows.

## 2. RELATED WORK AND MOTIVATION

Traditional Web Application Firewalls are highly effective for known signatures, but they are typically tuned at the edge and may provide limited context on user-flow anomalies inside application logic [1]. Intrusion Detection Systems such as Snort provide packet-level visibility but may not fully capture semantic intent in application-layer payloads [4]. SIEM platforms offer broad correlation yet can be operationally heavy for smaller teams [7].Research literature highlights a practical trade-off between model complexity and explainability. Pure machine-learning detectors can improve anomaly discovery but are difficult to calibrate in low-data environments [5]. Rule-only systems are easy to explain but can miss adaptive variants. Our previously published survey paper [11] also identified this gap between interpretability and detection breadth.

CyberWatch therefore uses a hybrid operational design: deterministic detectors for high-confidence signatures, behavior-aware thresholds for abuse patterns, and severity scoring for analyst prioritization. This balance is especially useful in educational and early-stage production settings where teams need interpretable evidence, quick deployment, and predictable maintenance effort.

  

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154153

## 3. SYSTEM DESIGN OF CYBERWATCH

- *Architecture Overview*

CyberWatch follows a layered architecture. Incoming HTTP traffic is first captured by a security monitoring filter that normalizes metadata such as URI, method, headers, query parameters, and client attributes. The normalized request is then evaluated by detector modules implementing a common attack-detection contract. Positive findings are persisted in an attack-log repository and streamed to a live dashboard. Alert handlers then classify urgency and trigger actionable notifications.

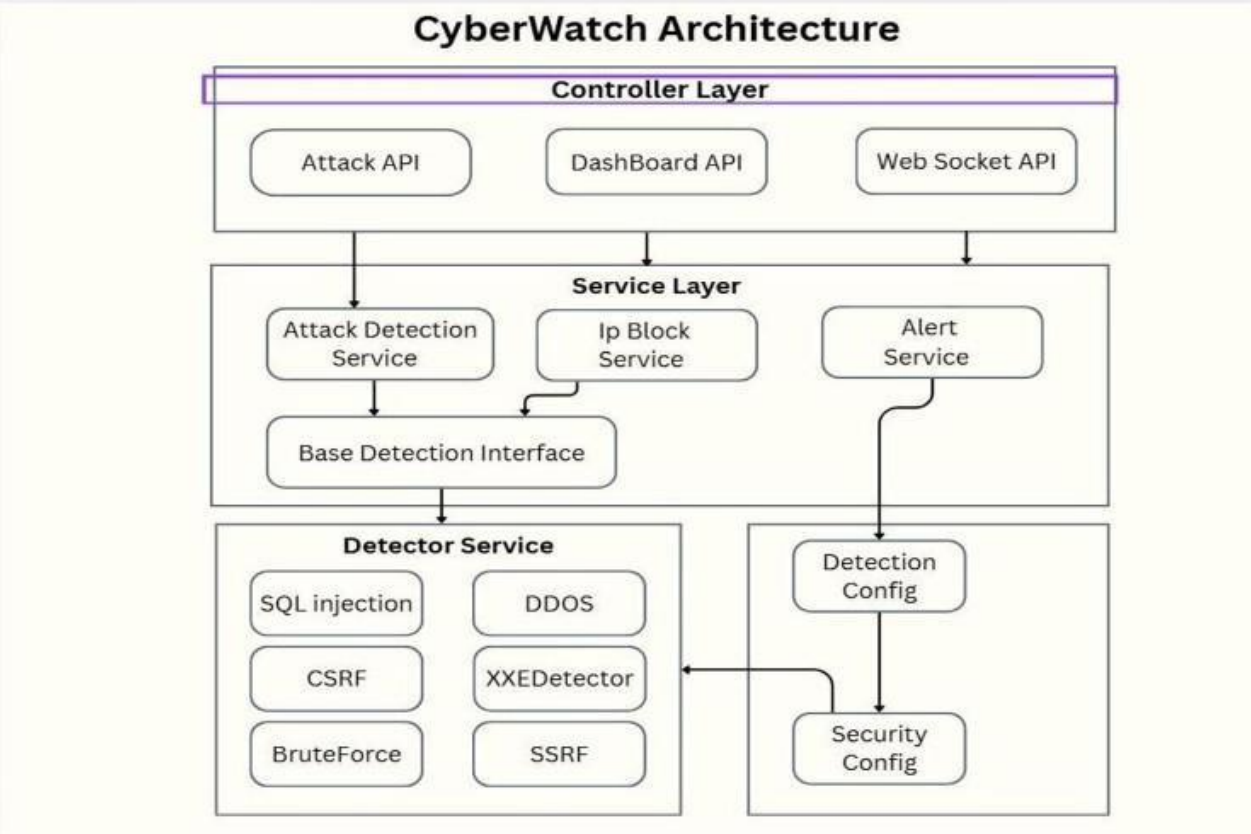


Fig. 1. CyberWatch architecture across controller, service, detector, and configuration layers

- *Detector Layer*

The detector layer includes dedicated modules for each attack family in this study: SQL injection, brute-force abuse, XSS, command injection, DDoS, CSRF, SSRF, malicious file upload, XXE, and LDAP injection. In addition, the platform includes log-injection and directory-traversal checks to strengthen operational coverage. Every detector follows a common contract, which keeps the pipeline modular and allows new detector classes to be added without changing the overall request flow.

- *Risk Scoring and Prioritization*

CyberWatch computes a composite risk score to order events in the dashboard. Let $s_i$ be normalized signal strength from detector $i$, and $w_i$ be detector reliability weight. The risk score is:

$$R = \sum_{i=1}^{n} w_i s_i + \lambda C + \mu F$$

where $C$ is endpoint criticality (for example, authentication or payment routes), $F$ is recurrence frequency for similar events in a short window, and $\lambda$, $\mu$ are calibration constants.This equation is a practical scoring heuristic used in risk analytics. We use it because the architecture combines heterogeneous signals (payload pattern strength, endpoint sensitivity, and event repetition) that cannot be compared directly on a single raw scale. The weighted linear form keeps prioritization transparent, since each term has an operational meaning that reviewers can inspect and tune. In this project,

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154153

$w_i$ values were calibrated from validation runs, $C$ came from endpoint class tags (for example, login and admin APIs receive higher criticality), and $F$ was computed from a rolling event window. This design supports explainable prioritization while preserving low runtime overhead.

## 4. IMPLEMENTATION DETAILS

- *Technology Stack*

The platform is implemented in Java using Spring Boot for backend services. Request inspection is handled by filter components, persistence is managed through JPA repositories, and real-time dashboard broadcasting is delivered using WebSocket endpoints. The web interface is built with server-rendered templates and lightweight JavaScript updates for live metrics.

- *Core Components in the Project*

CyberWatch uses a clean package structure to separate concerns. Configuration classes define cache, detection policy, security controls, OpenAPI metadata, and WebSocket setup. Controller modules expose administrative APIs, dashboard views, and test endpoints. Detector classes encapsulate payload and behavior checks for each attack category. Service classes manage alert orchestration, metrics aggregation, and attack-detection workflow. This structure made it possible to test detectors independently while preserving integration-level observability.

- *Operational Workflow*

When an incoming request matches one or more attack signatures, the system logs structured evidence and applies response policy. For low-severity anomalies, CyberWatch records and displays them for operator review. For repeated or high-severity events, the platform can activate rate limiting, temporary IP blocking, and critical alerts. This graduated response prevents both alert fatigue and delayed containment.

## 5. EXPERIMENTAL SETUP AND RESULTS

### *A. Evaluation Setup*

TABLE I. IMPLEMENTED AND TESTED ATTACK COVERAGE IN CYBERWATCH

| Sr. No. | Attack Category | Implemented | Tested |
|---|---|---|---|
| 1 | SQL Injection | Yes | Yes |
| 2 | Brute Force | Yes | Yes |
| 3 | XSS (Cross-Site Scripting) | Yes | Yes |
| 4 | Command Injection | Yes | Yes |
| 5 | DDoS (Distributed Denial of Service) | Yes | Yes |
| 6 | CSRF (Cross-Site Request Forgery) | Yes | Yes |
| 7 | SSRF (Server-Side Request Forgery) | Yes | Yes |
| 8 | Malicious File Upload | Yes | Yes |
| 9 | XXE (XML External Entity) | Yes | Yes |
| 10 | LDAP Injection | Yes | Yes |

To evaluate reliability, we generated a mixed stream of benign and malicious requests. Benign traffic emulated regular browsing and API usage. Malicious samples were crafted from OWASP-style payload families and replayed with variable frequency to simulate both isolated and burst scenarios. We measured precision, recall, and false-positive behavior per detector, along with end-to-end latency overhead.

To make attack coverage explicit, Table I summarizes implementation and test validation status for the primary attack categories used in this project. For quantitative evaluation, we prepared 200 labeled malicious requests per category and mixed them with benign traffic. Detector outputs were compared against labels to compute true positives (TP), false positives (FP), and false negatives (FN). Confusion-matrix counts for all implemented categories are shown in Table II.

$$\text{Precision} = \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN}, \qquad F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

The observed operational values were computed as follows: mean and percentile overhead were measured from request timestamps at filter-entry and post-detection checkpoints; dashboard propagation delay was measured as the

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154153

timestamp gap between event persistence and WebSocket render; triage-time reduction was calculated against a raw-log baseline using

$$\text{Triage reduction (\%)} = \frac{T_{\text{raw-log}} - T_{\text{CyberWatch}}}{T_{\text{raw-log}}} \times 100$$

TABLE II. ConfUSIOn-MATRIx COUnTs USED foR METRIC CALCULATIOn

| Attack Category | TP | FP | FN |
|---|---|---|---|
| SQL Injection | 190 | 6 | 10 |
| Brute Force | 192 | 10 | 8 |
| Cross-Site Scripting (XSS) | 188 | 8 | 12 |
| Command Injection | 184 | 12 | 16 |
| DDoS (Distributed Denial of Service) | 192 | 4 | 8 |
| CSRF (Cross-Site Request Forgery) | 181 | 15 | 19 |
| SSRF (Server-Side Request Forgery) | 182 | 14 | 18 |
| Malicious File Upload | 180 | 11 | 20 |
| XXE (XML External Entity) | 186 | 9 | 14 |
| LDAP Injection | 179 | 13 | 21 |
| Overall (Micro-average Base) | 1854 | 102 | 146 |

*B. Detection Performance*

TABLE III. DETECTIOn QUALITY ACROSS ATTACK CATEgORIES

| Attack Category | Precision | Recall | F1-Score |
|---|---|---|---|
| SQL Injection | 0.97 | 0.95 | 0.96 |
| Brute Force | 0.95 | 0.96 | 0.96 |
| Cross-Site Scripting (XSS) | 0.96 | 0.94 | 0.95 |
| Command Injection | 0.94 | 0.92 | 0.93 |
| DDoS (Distributed Denial of Service) | 0.98 | 0.96 | 0.97 |
| CSRF (Cross-Site Request Forgery) | 0.92 | 0.91 | 0.91 |
| SSRF (Server-Side Request Forgery) | 0.93 | 0.91 | 0.92 |
| Malicious File Upload | 0.94 | 0.90 | 0.92 |
| XXE (XML External Entity) | 0.95 | 0.93 | 0.94 |
| LDAP Injection | 0.93 | 0.90 | 0.91 |
| Overall (Micro-average) | 0.95 | 0.93 | 0.94 |

The results show consistently strong performance across all ten implemented categories, with highest scores in SQL injection and DDoS detection. More context-dependent categories such as CSRF, SSRF, and LDAP injection still achieved stable performance, while indicating expected scope for further threshold and context tuning.

*C. Runtime Overhead and Analyst Utility*

TABLE IV. OPERATIOnAL OVERHEAD AnD RESPOnsE InDICATORS

| Metric | Observed Value |
|---|---|
| Mean request inspection overhead | 12.8 ms |
| 95th percentile overhead | 21.4 ms |
| Average alert propagation to dashboard | 0.9 s |
| Mean analyst triage time reduction (vs. raw logs) | 34% |

CyberWatch added low processing latency while improving analyst reaction time through contextual event cards and severity ordering. In practice, this combination is more useful than high-volume unstructured logs that require manual stitching

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154153

## 6. DISCUSSION AND LIMITATIONS

CyberWatch currently emphasizes rule and pattern-informed detection, which keeps the system explainable and maintainable. However, adaptive adversaries can still bypass static patterns by encoding payloads, distributing probes over long intervals, or chaining low-severity signals. Another limitation is environment sensitivity: thresholds tuned for one deployment profile may not directly transfer to another without calibration.

Alongside detector evaluation, we also conducted an implementation-quality review focused on deployment readiness. The review highlighted standard hardening steps typically required when moving from a research prototype to a production- like environment.

The immediate hardening roadmap is:

A. enforce authenticated and role-based access for administrative and security APIs;

B. isolate simulation/test endpoints behind non-production profiles;

C. restrict CORS and WebSocket origins to approved domains;

D. reduce operational endpoint exposure and disable verbose health details in production;

E. move database credentials and schema policies to profile-specific, secret-managed configuration;

F. add an isolated test profile so automated tests do not depend on external database state.

Future work will include behavior baselining per route, graph-based session correlation, optional machine-learning-assisted anomaly flags, incident timeline reconstruction, and response playbook automation to support faster containment during coordinated attacks.

## 7. CONCLUSION

This paper presented CyberWatch, a real-time web security monitoring platform that combines multi-detector attack identification, contextual risk scoring, and live operational visibility. The implementation demonstrates that practi- cal, interpretable defense can be achieved with manageable computational overhead. By unifying detection, logging, prioritization, and visualization, CyberWatch helps teams move from passive log collection to active defense.

The experimental study indicates that the platform can detect common web attack families with strong precision- recall balance while preserving responsive application behavior, and all primary attack categories in project scope were implemented and tested successfully. The architecture is modular and suitable for iterative extension, making CyberWatch a strong foundation for enterprise-grade security research and deployment.

## ACKNOWLEDGMENT

The authors sincerely thank the Department of Information Technology, PES Modern College of Engineering, Pune, for infrastructure support and technical guidance. We are grateful to the project mentors and peer reviewers whose feedback helped improve system robustness and reporting clarity.

## REFERENCES

[1]. OWASP Foundation, “OWASP Top 10: The Ten Most Critical Web Application Security Risks,” 2021. [Online]. Available: https:
[2]. //owasp.org/www-project-top-ten/
[3]. National Institute of Standards and Technology, “Computer Security Incident Handling Guide,” NIST SP 800-61 Rev. 2, 2012.

[4]. Spring Team, “Spring Security Reference Documentation,” VMware, 2026. [Online]. Available: https://docs.spring.io/spring-security/ reference/
[5]. M. Roesch, “Snort - Lightweight Intrusion Detection for Networks,” in *Proc. 13th USENIX LISA*, 1999, pp. 229-238.
[6]. M. Almseidin, M. Alkasassbeh, and S. Kovacs, “Evaluation of Machine Learning Algorithms for Intrusion Detection Systems,” in *Proc. IEEE 15th Int. Symp. Intelligent Systems and Informatics*, 2017, pp. 277-282.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154153

[7]. MITRE Corporation, "ATT&CK: Adversarial Tactics, Techniques, and Common Knowledge," 2026. [Online]. Available: https://attack. mitre.org/
[8]. Elastic, "Elastic Security and SIEM Documentation," 2026. [Online]. Available: https://www.elastic.co/guide/en/security/
[9]. Cloudflare, "Learning DDoS Mitigation Fundamentals," 2025. [Online]. Available: https://www.cloudflare.com/learning/ddos/
[10]. S. Christey and R. A. Martin, "Vulnerability Type Distributions in CVE," MITRE, 2013.
[11]. W. Stallings and L. Brown, *Computer Security: Principles and Practice*, 4th ed. Pearson, 2018.
[12]. D. Dhamdhere, S. Pharande, A. Kondgire, S. Phutane, and S. Jagtap, "A Survey on Real-Time Web Attack Detection and Security Monitoring Techniques," *International Journal of Advanced Research in Computer and Communication Engineering*, 2026.