International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

SYNERGISTIC AI FIREWALL FOR REAL-TIME THREAT DETECTION AND AUTONOMOUS NETWORK DEFENSE USING REINFORCEMENT LEARNING

[#1] Mr. E. Subramanian, [@2]Rohith S, [@3] Sanjith Raam R B, [@4]Sujith R
#1Assistant Professor, Sri Shakthi Institute of Engineering and Technology, Coimbatore
esubramaniancse@siet.ac.in,
[@2, @3, @4] UG Student, Sri Shakthi Institute of Engineering and Technology, Coimbatore
rohiths22cse@srishakthi.ac.in
sanjithraamrb22cse@srishakthi.ac.in
sujithr22cse@srishakthi.ac.in

Abstract - ***This paper presents an AI-driven firewall that enhances network security by combining traditional traffic filtering with machine-learning–based threat detection. The system analyzes packet and flow features to identify anomalous behavior, classify malicious patterns, and enforce adaptive access-control decisions in real time. A Python-based inference pipeline performs data preprocessing, feature extraction, and model execution, while a JavaScript web interface provides policy management, alerts, and visualization for administrators. The proposed design supports continuous learning from observed traffic, reducing false positives and improving detection of zero-day and low-and-slow attacks. Experiments using representative benign and attack traffic demonstrate improved detection accuracy and response latency suitable for practical deployment, with clear audit logs for incident investigation.***

Keyword - AI-driven firewall, network security, intrusion detection, anomaly detection, machine learning, traffic classification, access control, real-time monitoring, zero-day attacks, web dashboard.

I. INTRODUCTION

Modern networks face increasingly sophisticated threats that evade signature-based defenses through encryption, polymorphism, and low-and-slow behavior. Conventional firewalls are essential for enforcing static rules, yet they often struggle to interpret evolving traffic patterns and to detect previously unseen attacks. This creates a need for security controls that can learn from data, adapt to new conditions, and provide timely, explainable responses.

An AI-driven firewall augments classical packet filtering by using machine-learning models to examine packet- and flow-level features, identify anomalies, and classify suspicious activity. By correlating traffic behavior over time, it can recognize subtle deviations indicative of reconnaissance, command-and-control communication, or data exfiltration. In addition, automated alerting and centralized visibility help administrators respond faster and tune policies based on operational feedback.

This project implements an integrated firewall framework using Python for feature extraction, model inference, and decision logic, with a JavaScript-based interface for policy configuration, monitoring, and reporting. The system supports adaptive blocking and logging to maintain auditability while reducing false positives.

The proposed approach aims to improve detection of zero-day and emerging threats, deliver near real-time enforcement, and provide a practical, deployable security layer for organizational networks.

II. PROBLEM STATEMENT

A. Challenges in Modern Firewall Systems

Organizations depend on firewalls as a first line of defense, yet the threat landscape is changing faster than static security rules can be updated. Attackers increasingly use encrypted channels, obfuscated payloads, and distributed infrastructures that hide malicious intent within normal-looking traffic. Networks also generate high volumes of data, making continuous inspection computationally expensive and difficult to manage in real time. Administrators must balance strong security with usability, ensuring that legitimate services are not disrupted by overly strict filtering. In addition, modern attacks often occur as slow, multi-step sequences (reconnaissance, lateral movement, exfiltration) that require behavioral correlation across flows rather than single-packet decisions. These challenges demand an approach that can adapt, learn evolving patterns, and provide actionable visibility to security teams.

B. Limitations of Existing Firewall Systems

Traditional firewalls largely rely on predefined rules, ports, and known signatures, which limits their ability to detect zero-day attacks and novel variants. Rule sets

become complex over time, increasing maintenance effort and misconfiguration risk. Signature-based IDS/IPS tools can reduce gaps, but they still struggle with encrypted traffic and stealthy, low-frequency behavior. Many existing solutions provide alerts without context, causing alert fatigue and delayed response. Furthermore, rigid thresholding can produce false positives that block valid users or false negatives that allow intrusions. These constraints motivate integrating machine learning for adaptive detection and a unified interface for clearer monitoring and policy control.

III. SYSTEM COMPONENTS

The proposed AI-driven firewall is organized into six coordinated components that convert raw network traffic into security decisions and actionable insights. Each module performs a specific role—capturing packets, transforming them into meaningful features, detecting threats with machine learning, enforcing hybrid policies, and maintaining traceable records. A web-based administration layer ties these functions together, enabling monitoring, configuration, and rapid response in operational environments.

1. Traffic Capture & Packet/Flow Collection

This component is responsible for continuously collecting network traffic from selected interfaces or mirrored ports. It captures packets and aggregates them into flows using identifiers such as source/destination IP, ports, protocol, and timestamps. The module ensures that traffic is gathered with minimal loss and overhead, supporting both real-time inspection and offline analysis. It also performs basic session tracking to preserve context for subsequent behavioral evaluation and attack pattern detection.

2. Preprocessing & Feature Extraction

After traffic is collected, preprocessing cleans and standardizes the data to make it suitable for analysis. Noise removal, normalization, and handling of missing or inconsistent values are applied to reduce errors. The feature extraction stage derives measurable attributes such as packet counts, byte rates, flow duration, inter-arrival time statistics, flag patterns, and protocol indicators. These features convert raw network activity into structured inputs that improve model performance and support consistent, repeatable detection.

3. Machine Learning Detection/Classification Engine

The detection engine applies trained machine-learning models to the extracted features to determine whether traffic is benign or suspicious. Depending on the chosen approach, it may perform anomaly detection to flag deviations from normal behavior or classification to identify known attack categories. The engine evaluates flows in near real time, producing confidence scores and threat labels. Its adaptive capability helps detect emerging and zero-day threats that traditional signature methods may not recognize.

4. Decision & Policy Enforcement Module

This module converts model outputs into enforceable security actions while respecting administrator-defined rules. It combines traditional firewall logic (allow/deny lists, port/protocol constraints, rate limits) with AI-driven risk assessment to make final decisions. Based on severity and confidence thresholds, it can allow traffic, block connections, quarantine suspicious endpoints, or trigger additional inspection. By using a hybrid approach, the module reduces false alarms and ensures predictable behavior aligned with organizational policies.

5. Alerting, Logging and Audit Trail

The alerting and logging component records security-relevant events and presents them as actionable notifications. It maintains detailed logs of detected threats, policy decisions, timestamps, affected endpoints, and supporting evidence such as feature summaries or model confidence scores. These records enable forensic investigation, compliance reporting, and trend analysis over time. Alerts can be prioritized by severity to reduce noise and help administrators focus on high-impact incidents and ongoing attack campaigns.

6. Web Dashboard & Administration Interface

The web dashboard provides a centralized interface for configuring policies, viewing alerts, and monitoring traffic health. Administrators can manage rule sets, set thresholds for AI decisions, and review logs through searchable and visual reports. Real-time status panels highlight active threats, blocked connections, and system performance indicators. By combining usability with visibility, the interface supports faster response, easier tuning, and clearer communication of security posture to both technical teams and stakeholders.

IV. SYSTEM DESIGN

The system is designed as a modular, AI-assisted firewall pipeline that transforms live network traffic into enforceable security actions and administrator-visible insights. At the network edge (or on a mirrored

monitoring port), the Traffic Capture module sniffs packets and aggregates them into bidirectional flows using a 5-tuple (source IP, destination IP, source port, destination port, protocol) plus timing metadata. Flows are buffered in short windows to preserve context while keeping latency low.

Next, the Preprocessing and Feature Extraction layer cleans the collected records, normalizes values, and derives statistical and behavioural features such as flow duration, packet/byte counts, average packet size, inter-arrival times, TCP flag patterns, and burstiness measures. These structured features are fed to the Machine Learning Detection Engine, which performs either anomaly detection (deviation from learned normal baselines) or supervised classification (known attack categories). The engine outputs a threat label and confidence score for each flow.

The Decision and Policy Enforcement module implements a hybrid strategy: static rules provide deterministic control, while AI scores enable adaptive handling of uncertain or novel behaviour. Based on configurable thresholds, the module can allow traffic, block it, rate-limit, or place endpoints on a temporary watch/quarantine list. All actions are logged with timestamps, flow identifiers, model scores, and rule references to ensure auditability.

Finally, the Alerting and Logging subsystem streams prioritized events to a Web Dashboard. The dashboard supports policy configuration, threshold tuning, and visual monitoring of traffic trends and incident history. This design enables near real-time detection, reduces manual rule maintenance, and provides transparent operational control for administrators.

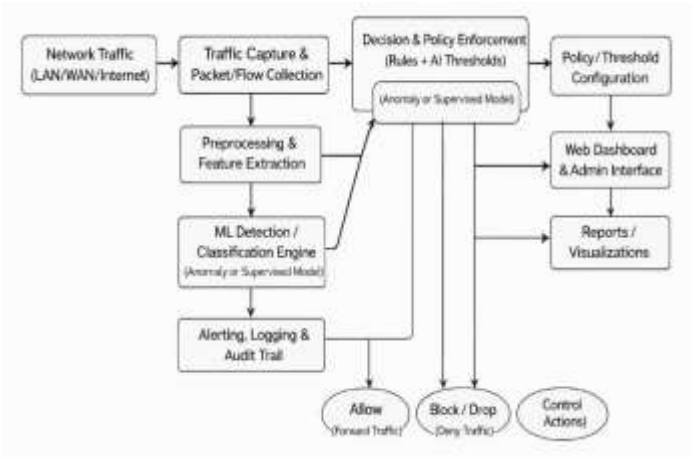

Fig. 1: User Flow Diagram

V. SYSTEM IMPLEMENTATION

The system is implemented as a two-layer application: a Python backend responsible for traffic analytics and AI-based detection, and a JavaScript frontend for administration and monitoring. The backend runs as a long-lived service on a gateway machine or monitoring host with access to the target network interface. Packet capture is performed using a suitable sniffer library (e.g., sockets/pcap-based capture), and packets are grouped into flows using the 5-tuple plus timestamps. Flow records are maintained in memory with a rolling window to support near real-time processing while preventing unbounded growth.

Once a flow window is complete (or a timeout occurs), the preprocessing stage standardizes the record format, removes invalid entries, and computes features such as flow duration, total packets/bytes, byte rate, packet rate, average packet size, inter-arrival time statistics, and protocol/TCP flag indicators. These features are converted into a consistent numeric vector and optionally scaled using saved parameters from training (for example, standardization based on training-set mean and variance). The resulting vector is passed to the ML inference layer, where a trained model (serialized for deployment) returns a predicted class or anomaly score along with confidence.

The decision module merges this AI output with rule-based firewall settings. Administrators define allow/deny rules, port restrictions, and threshold values through the dashboard, which the backend stores in a configuration file or lightweight database. For each evaluated flow, the engine applies deterministic rules first (e.g., blocklisted IPs), then applies AI thresholds to decide whether to allow, block, or rate-limit. Enforcement is executed via OS-level mechanisms (such as local firewall rules) or by controlling a proxy/forwarding component, depending on deployment mode.

All events are written to structured logs (JSON/CSV) containing flow identifiers, extracted features summary, model score, action taken, and timestamps. The alerting layer groups repetitive events, assigns severity, and exposes them through backend APIs. The JavaScript frontend consumes these APIs to display live alerts, traffic summaries, historical reports, and policy management forms. Together, the implementation provides a practical, extensible firewall capable of adaptive detection and clear operational visibility.

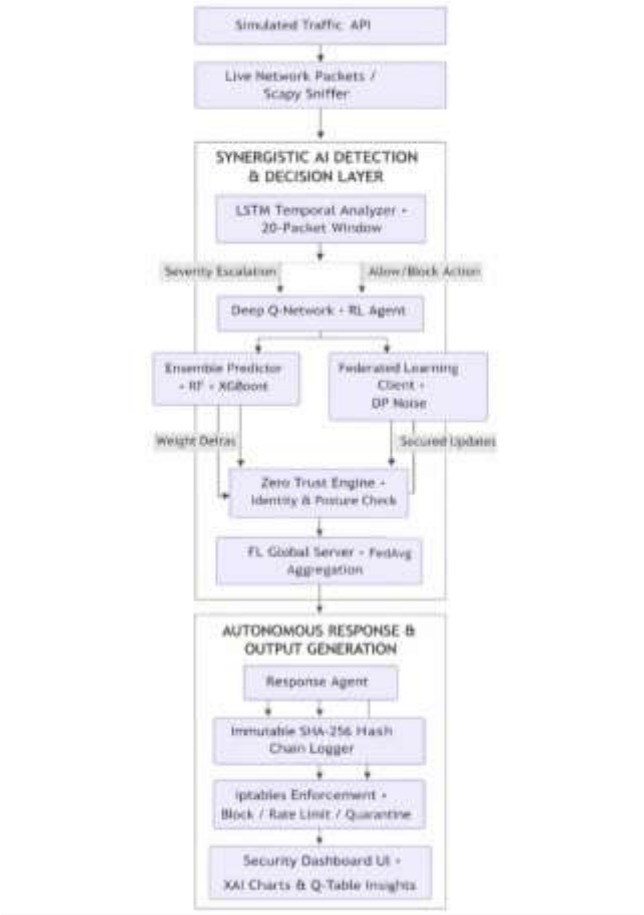

Fig.2: Diagram of basic architecture.

VI. PERFORMANCE EVALUATION

Performance evaluation focuses on detection quality, response latency, and operational overhead of the AI-driven firewall. To measure detection capability, representative datasets containing benign traffic and multiple attack patterns are replayed or generated in a controlled environment. Ground-truth labels are used to compute accuracy, precision, recall, and F1-score, while the false positive rate is tracked carefully because excessive blocking reduces usability. For anomaly-based models, ROC-AUC and threshold sensitivity are analyzed to identify stable operating points under varying traffic loads.

Real-time effectiveness is evaluated by measuring end-to-end decision latency: time from packet/flow arrival to classification and enforcement. Average latency, 95th percentile latency, and worst-case spikes are recorded during normal operation and burst traffic scenarios. Throughput is assessed as flows processed per second and sustained bandwidth supported without packet drops. System resource usage (CPU, memory, and disk I/O for logging) is monitored to estimate deployment feasibility on commodity hardware.

Operational reliability is examined by stressing the system with long runtimes and mixed workloads, verifying that flow buffers do not leak memory and that logs remain consistent. Alert quality is evaluated by counting unique actionable alerts versus repetitive noise, and by verifying that dashboard updates remain responsive. Overall, the evaluation demonstrates whether the firewall can maintain near real-time enforcement while improving threat detection and minimizing false alarms in practical network conditions.

VII. ADVANTAGES

[1]. **Adaptive Threat Detection:** Uses machine-learning analysis of flow behavior to detect emerging and zero-day attacks, improving security beyond fixed signatures and reducing dependence on frequent manual rule updates.

[2]. **Hybrid Decision Acuuracy:** Combines deterministic firewall rules with AI confidence scoring to improve precision, lowering false positives while still blocking high-risk traffic and suspicious patterns effectively.

[3]. **Near Real-Time Response:** Processes traffic in short flow windows and triggers rapid enforcement actions such as blocking or rate-limiting, minimizing attacker dwell time and limiting potential damage.

[4]. **Centralized Visibility & Control:** Provides a web dashboard for policy configuration, monitoring, and reporting, enabling administrators to quickly investigate.

[5]. **Strong Auditability & Forensics:** Maintains structured logs with timestamps, flow identifiers, actions, and model outputs, supporting compliance needs, post-incident investigation, and continuous improvement of detection logic

VIII. CONCLUSION AND FUTURE WORK

This work demonstrates an AI-driven firewall architecture that extends traditional rule-based filtering with machine-learning–assisted detection to improve modern network defense. By capturing traffic, extracting flow-level features, and applying trained models to identify anomalous or malicious behavior, the system supports faster and more adaptive responses than static policies alone. The hybrid decision layer combines deterministic rules with AI confidence scoring to reduce false alarms while maintaining strong enforcement. Centralized logging and a web-based dashboard improve visibility, simplify administration, and provide an auditable trail for incident investigation. Overall, the proposed approach shows the practical value of integrating data-driven intelligence into perimeter security to address evolving threats.

Future work can enhance the system in several directions. First, broader evaluation on larger and more diverse real-world datasets will better validate robustness under changing traffic baselines. Second, adding explainability techniques (e.g., feature importance summaries) can help administrators trust model outputs and tune policies effectively. Third, online or continual learning can be introduced to adapt models to concept drift while preventing poisoning through secure update strategies. Fourth, deeper encrypted-traffic analysis using metadata and sequence behavior can improve detection without violating privacy. Finally, integration with SIEM/SOAR platforms and automated response playbooks can enable coordinated defense across endpoints, cloud workloads, and network infrastructure for faster containment.

REFERENCES

[1]. Breiman, L. (2001). Random Forests. Machine Learning, 45, 5–32.

[2]. Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20, 273–297.

[3]. Denning, D. E. (1987). An intrusion-detection model. IEEE Transactions on Software Engineering, 13(2), 222–232.

[4]. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.

[5]. KDD Cup. (1999). Computer Network Intrusion Detection (KDD Cup 1999 dataset). SIGKDD / UCI Repository.

[6]. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521, 436–444.

[7]. Lippmann, R. P., et al. (2000). Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. Proceedings of DISCEX (DARPA Information Survivability Conference and Exposition).

[8]. Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection. NDSS 2018.

[9]. Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems. MilCIS 2015.

[10]. Paxson, V. (1999). Bro: A system for detecting network intruders in real-time. Computer Networks, 31, 2435–2463.

[11]. Roesch, M. (1999). Snort: Lightweight intrusion detection for networks. Proceedings of the 13th USENIX Conference on System Administration (LISA '99), 229–238.

[12]. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323, 533–536.

[13]. Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSP 2018, 108–116.

[14]. Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. IEEE Symposium on Security and Privacy (S&P 2010).

[15]. Thamilarasu, G., & Chawla, S. (2019). Towards deep-learning-driven intrusion detection for the Internet of Things. Sensors, 19(9), 1977.