



ARTICLE

Enhancing Intrusion Prevention Systems through Hybrid AI-Driven Log Analysis: A Framework for Predictive Blocking

Cătălin-Eugen Bucur¹, Georgiana Crihan^{2,*} and Elena-Grațiela Robe-Voinea^{3,*}

¹Faculty of Electronics, Telecommunications and Information Technology, National University of Science and Technology POLITEHNICA Bucharest, Bucharest, Romania

²Department of Communication and Information Technology, Romanian Naval Forces, Constanta, Romania

³Department of Tactics and Naval Armament, Faculty of Marine Engineering, Romanian Naval Academy “Mircea cel Bătrân”, Constanta, Romania

*Corresponding Authors: Georgiana Crihan. Email: georgian.crihan@ugal.ro; Elena-Gratiela Robe-Voinea. Email: elena.robe@anmb.ro

Received: 14 May 2026; Accepted: 09 September 2026; Published: 22 October 2026

ABSTRACT: A modular AI-driven intrusion prevention framework integrating a hybrid machine-learning decision engine, explainable artificial intelligence, and kernel-level enforcement is presented and experimentally validated. The solution decouples attack-specific event detection from decision-making and mitigation, allowing heterogeneous detectors to operate within a shared prevention workflow. Mature detection mechanisms are preserved where appropriate, while a hybrid Random Forest (RF) and Long Short-Term Memory (LSTM) engine analyzes standardized flow-level representations to support behavior-aware blocking decisions. SHAP-based explainability provides feature-level interpretation of AI decisions, whereas extended Berkeley Packet Filter/eXpress Data Path (eBPF/XDP) enables packet filtering at the kernel level. Experimental validation was performed through repeated executions of benign traffic, SSH brute-force attacks, slow SSH brute-force attacks, and port-scanning activity. The results show successful mitigation of the evaluated attack scenarios without false positives during benign operation. Compared with a conventional Fail2Ban deployment, the proposed framework introduces additional computational overhead due to feature extraction, machine-learning inference, explainability, and orchestration services. This overhead represents the cost of a reusable prevention architecture that combines conventional detection, hybrid AI inference, interpretable decision support, and kernel-level enforcement. Overall, the study demonstrates the feasibility of integrating heterogeneous detection mechanisms into a unified AI-driven intrusion prevention pipeline and offers a basis for future extension toward broader classes of cyberattacks.

KEYWORDS: Intrusion prevention systems; machine learning; network security; deep learning; AI; eBPF/XDP

1 Introduction

Maintaining the integrity of enterprise data networks remains a fundamental requirement for modern organizational infrastructures. Although network technologies continue to evolve, many enterprise environments still rely on traditional client-server architectures and TCP/IP-based communication. Consequently, protecting these infrastructures requires security mechanisms capable of operating effectively under increasingly complex network conditions. As network traffic volumes increase and cyberattacks become more diverse and sophisticated, conventional security mechanisms face growing challenges in distinguishing legitimate administrative activity from malicious behavior.

Traditional firewalls and signature-based Intrusion Detection and Prevention Systems (IDS/IPS) remain essential components of enterprise security platforms; however, their rule-based operation may provide limited adaptability to evolving attack strategies and previously unseen behavioral patterns. These challenges have stimulated increasing interest in behavioral analysis and machine-learning-driven security mechanisms for improving the adaptability of intrusion prevention systems. In this context, improving the adaptability and extensibility of intrusion prevention solutions has become an important research direction for protecting modern networked infrastructures [1–3]. In modern enterprise environments, exposed services such as SSH, FTP, and web administration interfaces remain among the primary targets of cyberattacks. Traditional IDS/IPS solutions, particularly rule-based mechanisms, continue to be widely deployed because of their simplicity, predictability, and operational maturity. Their deterministic operation makes them highly effective against well-defined attack patterns; however, adapting to evolving attack strategies and complex behavioral characteristics often requires additional analytical capabilities. Recent studies have shown that machine-learning approaches capable of automatically learning behavioral relationships from network traffic can effectively complement conventional rule-based detection, particularly in scenarios involving previously unseen attacks or patterns that are difficult to express using manually defined rules [4].

Threshold-based protection mechanisms rely on predefined configuration parameters, such as event-count thresholds and observation windows, which provide deterministic, predictable, and operationally mature behavior. These mechanisms remain highly effective against well-defined attack patterns, particularly in authentication-oriented scenarios. However, their static decision logic may offer limited adaptability to adversaries that deliberately distribute malicious activity over extended periods or across multiple sources. Low-and-slow attack strategies and distributed campaigns are specifically designed to remain below fixed detection thresholds, reducing the effectiveness of purely rule-based protection mechanisms [5]. The design philosophy is to decouple attack-specific event detection from decision-making and enforcement. The architectural separation allows dedicated detection elements to operate independently while sharing an AI-driven decision engine, explainability module, and enforcement mechanism, facilitating the extension of the platform toward additional attack categories. Furthermore, regular expression (regex)-based detection is inherently limited to matching predefined textual patterns and cannot directly capture complex behavioral relationships or temporal dependencies across multiple events. While this approach remains highly effective for well-defined log-driven attack scenarios, it enables only a partial view of increasingly diverse attack behaviors. Combining specialized event detectors with a shared behavior-aware inference layer allows deterministic detection to be complemented by AI-based behavioral analysis before mitigation is applied [6]. Building on this design principle, the framework integrates hybrid RF–LSTM inference, SHAP-based explainability, and eBPF/XDP kernel-level enforcement into a unified pipeline.

The principal contributions of this study are summarized as follows:

- A modular and scalable AI-driven intrusion-prevention framework that decouples attack-specific event detection, hybrid decision making, explainability, and kernel-level enforcement, enabling specialized detectors to be integrated without modifying the core decision and enforcement pipeline.
- A hybrid RF–LSTM decision mechanism that combines flow-level discrimination with temporal behavioral analysis using a standardized 78-feature network-flow representation.
- The integration of SHAP-based explainability into an operational prevention workflow, providing interpretable support for AI-driven blocking decisions.
- The implementation and experimental evaluation of AI-driven eBPF/XDP-based kernel-level enforcement.

- An operational evaluation of the proposed framework across benign, authentication-attack, and reconnaissance scenarios, including comparison with a conventional Fail2Ban deployment as a reference baseline.

The remainder of this paper is organized as follows. [Section 2](#) reviews the relevant literature. [Section 3](#) presents the proposed architecture together with the model development and implementation details. [Section 4](#) reports the experimental evaluation and discusses the obtained results and their implications. Finally, [Section 5](#) concludes the paper and outlines future research directions.

2 Related Work

Traditional intrusion detection and prevention systems are commonly classified into signature-based and anomaly-based approaches. Signature-based detection identifies malicious activities by matching observed traffic against predefined attack patterns, providing high accuracy and low false alarm rates for known threats, but exhibiting limited capability against previously unseen or zero-day attacks. Anomaly-based detection identifies deviations from established normal behavior, enabling the detection of unknown threats at the expense of higher false positive rates and computational overhead. To leverage the complementary strengths of both paradigms, hybrid approaches combine signature- and anomaly-based techniques to improve detection robustness and adaptability in complex network environments. In operational deployments, platforms such as Snort, Suricata, and Zeek exemplify different approaches to network monitoring and intrusion detection. Snort and Suricata primarily rely on rule-based inspection for accurate detection of known attack patterns, whereas Zeek emphasizes protocol-aware traffic analysis and event generation. Although these mature solutions provide effective visibility into network activity, their dependence on predefined rules or protocol semantics may limit their ability to identify previously unseen behavioral patterns and complex multi-stage attacks [7,8]. Complementing network-based monitoring, host-based mechanisms such as Fail2Ban [9] provide reactive protection by analyzing service logs and automatically blocking offending IP addresses. While their simplicity, maturity, and ease of deployment have made them widely adopted for mitigating authentication-related attacks, their reliance on application logs and predefined thresholds allows only a partial view of system activity, motivating the integration of more adaptive intelligence-driven detection mechanisms.

These complementary strengths have motivated increasing research into Machine Learning (ML) and Deep Learning (DL) techniques capable of extending conventional rule-based detection through behavior analysis. By learning complex and non-linear relationships from network traffic, these approaches provide additional decision support beyond manually defined rules, thereby improving the adaptability of IDS/IPS solutions in increasingly complex environments [7]. Among supervised machine-learning approaches, Random Forest (RF) remains one of the most widely adopted algorithms because of its robustness, computational efficiency, and strong performance on structured cybersecurity datasets such as CICIDS2017 and UNSW-NB15 [10]. Deep learning approaches have further extended intrusion detection capabilities by modeling temporal dependencies within network traffic. Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) architectures, are specifically designed to process sequential data by maintaining information over multiple observations.

LSTM models are well suited for recognizing behavioral patterns that emerge progressively over time, making them particularly effective for detecting attack scenarios in which temporal context plays an important role. These advantages are accompanied by increased computational complexity, since training sequential neural networks requires substantially greater computational resources than conventional machine-learning models [11]. Moreover, numerous AI methodologies have been investigated for IDS/IPS applications with different objectives and deployment strategies, as summarized in [Table 1](#). Despite these

advances, most existing studies primarily optimize detection performance or individual IDS/IPS components. In contrast, the proposed framework focuses on a modular architecture in which heterogeneous detectors share a common AI inference, explainability, and kernel-level enforcement pipeline. This design facilitates the integration of additional detection capabilities without requiring changes to the core deployment workflow. The pipeline is experimentally evaluated through repeated scenarios and compared with a conventional Fail2Ban deployment that serves as an operational baseline. The study does not aim to provide a comprehensive benchmark against all existing IDS/IPS platforms. Instead, it focuses on comparing individual IDS/IPS implementations, and the present work focuses on the design and experimental evaluation of a modular AI-driven intrusion prevention architecture. A conventional Fail2Ban deployment is adopted as the operational baseline because it represents a mature threshold-based protection mechanism for authentication-oriented attacks and offers a suitable reference for evaluating the AI-driven prevention workflow. Broader qualitative comparisons with representative machine-learning and deep-learning-based IDS/IPS approaches are discussed in [Section 4.2](#).

Table 1: Overview of AI methodologies implemented in IDS/IPS architectures.

Methodologies	References	Use Case
Combined neural network for low-intensity DDoS detection	[5]	A combined approach for detecting low-intensity DDoS attacks that integrates statistical criteria with convolutional and Transformer blocks, an autoencoder for anomaly scoring, and a calibrated regressor for estimating attack intensity, complemented by explainability mechanisms.
AI-XAI models in IoT IDS	[12]	A systematic comparative evaluation of three representative AI models, CNN-BiLSTM, Random Forest, and XGBoost for intrusion detection on the NSL-KDD and UNSW-NB15 datasets applied in heterogeneous IoT environments.
AI-Driven IDS/IPS in 6G networks	[13]	Implementation of machine learning based on blockchain federated Gaussian multi-agent Q-encoder neural networks (BFGMAQENN) in a 6G network, optimized using whale swarm binary wolf optimization (WSBWO).
AI for IDS to secure IoMT	[14]	The proposed IDS hybrid system combines a pre-trained C4.5 classifier and Deep Q-Network reinforcement learning to dynamically adapt to evolving attacks, evaluated across WUSTL-EHMS, ECU-IoHT, DF_IOMT, and CICIOT23.
XAI in IDS	[15]	An intrusion detection system using DT, RF, and SVM, that incorporates the XAI algorithm LIME for better explainability and understanding of the black-box approach to reliable intrusion detection.
AI-Driven Dynamic Firewall Optimization	[16]	Dynamic firewall optimized using RL & LSTM-CNN for both detection and prevention, evaluated on the NSL-KDD and CIC-IDS2017 datasets using a simulated software-defined network environment.

(Continued)

Table 1 (continued)

Methodologies	References	Use Case
XAI techniques with ML models in IDS	[17]	Incorporating XAI techniques such as LIME, SHAP, and ELI5 with multiple ML classifiers in IDS, tested on UNSW-NB15 dataset.
A LIME-based eXplainable AI for IDS	[18]	Local Interpretable Model-Agnostic Explanations (LIME) coupled with ELI5 and Decision Tree algorithms to provide local and global explanations and improve the interpretation of IDS.

3 Methodology

3.1 System Architecture

Fig. 1 illustrates the proposed architecture, which follows a modular event-driven organization composed of four interacting layers. Attack detection, AI inference, kernel-level enforcement, and monitoring are implemented as independent components that communicate through standardized security events. This separation allows new attack-specific detectors without modifying the shared inference, enforcement, or monitoring infrastructure.

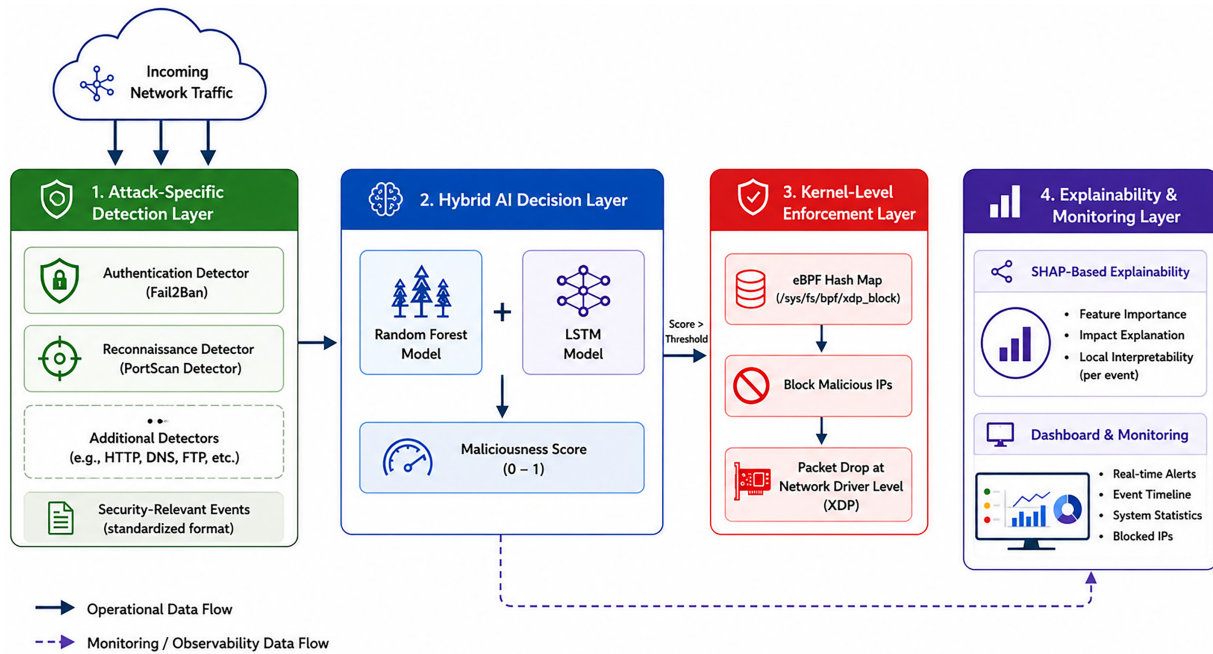


Figure 1: AI-driven intrusion prevention framework overview.

Attack detection is performed by specialized detectors, each designed to identify a particular category of malicious activity. In the current implementation, authentication-related attacks are identified by Fail2Ban through continuous monitoring of authentication logs (e.g., /var/log/auth.log) using configurable parameters such as maxretry and findtime, whereas reconnaissance activities are detected by a dedicated PortScan detector operating directly on network traffic. Each generated event triggers the feature-extraction module,

which constructs a 78-dimensional feature vector compatible with the representation used during model training. The feature ordering and preprocessing pipeline are preserved to ensure that the deployed RF and LSTM models receive input consistent with the data used during training.

The normalized feature vector is evaluated by a hybrid decision engine that combines a Random Forest (RF) classifier with a Long Short-Term Memory (LSTM) network. Static flow characteristics are analyzed by the RF model, whereas temporal relationships between consecutive observations are captured by the LSTM model, improving the detection of low-rate and stealthy attack behavior.

The outputs of both models are fused into a single hybrid attack score, which is compared with a configurable decision threshold exposed through a REST API. Because inference and enforcement operate independently of the event source, the same pipeline can be reused as additional detectors are integrated into the framework. Whenever the hybrid score exceeds the configured threshold, the corresponding source is inserted into the pinned eBPF blacklist map accessed by the XDP program attached to the network interface. Subsequent packets from blocked sources are discarded at the XDP layer before entering the Linux networking stack, reducing unnecessary packet processing. Decision interpretability is provided by the Explainability and Monitoring Layer through SHAP-based feature attribution, enabling the contribution of individual features to each prediction to be examined. Operational visibility is further supported by a Flask-based dashboard that presents model outputs, detector activity, mitigation events, and system status in real time. Because these functions are decoupled from the inference and enforcement path, they support system analysis and monitoring without affecting mitigation decisions.

3.2 RF Component Description

The RF classifier was selected because of its strong performance on structured tabular data, robustness to noisy and heterogeneous features, computational efficiency during inference, and compatibility with SHAP TreeExplainer for prediction interpretation [19,20]. These properties make RF well suited for deployment-oriented intrusion prevention systems, where both timely classification and decision transparency are important [21]. Formally, the Random Forest classifier is an ensemble learning model composed of n decision trees, denoted by $\{h(\mathbf{x}, \Theta_k)\}_{k=1}^n$, where $\mathbf{x}$ represents the input feature vector and Θ_k denotes an independent random vector controlling the construction of the k -th decision tree. The RF prediction score, S_{RF} , is obtained by aggregating the outputs of all constituent trees through a majority voting or probability averaging mechanism.

$$S_{RF} = \frac{1}{n} \sum_{i=1}^n \phi_i(\mathbf{x}) \quad (1)$$

In (1), $\phi_i(\mathbf{x})$ denotes the prediction of the i -th tree. This bagging (bootstrap aggregating) process has been shown to reduce model variance without significantly increasing bias, which accounts for the high robustness to noise and heterogeneity observed in intrusion detection tasks.

3.3 LSTM Component Description

The sequence model complements the RF classifier by modeling temporal dependencies that cannot be captured from isolated flow records. This capability is particularly relevant for low-rate and evolving attack patterns, where the temporal ordering of events offers important information for distinguishing malicious from benign activity [22–25]. The final decision score is computed as a weighted hybrid combination of the RF output and the LSTM output. The hybrid score combines the RF probability with the LSTM probability, allowing static flow characteristics and temporal behavior to contribute equally to the final decision. A runtime-configurable decision threshold, stored in `/var/lib/ips-ai/dynamic_config.json`, is used to determine

whether an observed source is classified as malicious. The LSTM component processes the input sequence $\mathbf{x} = (x_1, x_2, \dots, x_T)$ by maintaining a hidden state h_t and a cell state c_t , which act as the long-term memory of the network. At each time step t , the transition is governed by the following gating mechanisms:

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \quad (2)$$

where f_t , i_t , and o_t represent the forget, input, and output gates, respectively, and σ denotes the activation function of the sigmoid. The final output of the LSTM component for the sequence, S_{LSTM} , is derived from the last hidden state h_T . To take advantage of both static and temporal features, the final hybrid classification score S_{final} is calculated as a weighted linear combination of the RF and LSTM outputs:

$$S_{final} = w \cdot S_{RF} + (1 - w) \cdot S_{LSTM} \quad (3)$$

where $w \in [0, 1]$ is a weighting coefficient that balances the contribution of each model. An intrusion is detected if $S_{final} > \tau$, where τ is the dynamic threshold retrieved from the system configuration.

3.4 eBPF/XDP Component Description

The proposed architecture integrates the eBPF/XDP as the kernel-level enforcement mechanism that translates AI-generated security decisions into real-time packet filtering. After the user-space decision engine identifies a malicious source, the corresponding IP address is inserted into the pinned eBPF hash map located at `/sys/fs/bpf/xdp_block`. The XDP program attached to the network interface subsequently checks incoming packets against this map and immediately discards packets originating from blocked sources before they traverse the conventional Linux networking stack. Compared with conventional Netfilter-based frameworks (e.g., iptables or nftables), XDP operates in native driver mode by executing eBPF programs before packets enter the Linux networking stack. The early execution minimizes unnecessary packet processing while keeping packet filtering independent of the user-space decision logic. Accordingly, the AI engine is responsible only for deciding whether a source should be blocked, whereas the XDP program performs deterministic packet filtering within the kernel once that decision has been made [26]. eBPF/XDP was selected because it naturally complements the proposed architecture. It enables a clear separation between decision making and enforcement, allowing different detection modules to share a common kernel-level mitigation mechanism. In addition, because packet filtering is performed before higher-layer protocol processing, the enforcement stage remains lightweight and independent of application-specific services. Moreover, the programmable nature of eBPF enables security policies to be updated dynamically without modifying kernel code, making the mechanism well-suited for intrusion prevention systems [27]. Previous studies have established XDP as a high-performance programmable packet-processing framework capable of achieving substantially higher packet-filtering throughput than traditional Netfilter-based firewall mechanisms under controlled benchmarking conditions [27,28]. Likewise, eBPF has evolved into a mature and secure kernel execution environment whose verification framework enables programmable networking and security functionality while preserving kernel integrity [29]. In the present work, the experimental evaluation includes a dedicated comparison of the eBPF/XDP enforcement mechanism and a conventional Fail2Ban-based configuration in terms of rule installation time, CPU utilization during the enforcement phase, and client-observed transition time. These measurements are presented and discussed in [Section 4.3.3](#).

The objective of this comparison is to validate the practical integration of the enforcement architecture rather than to provide a comprehensive benchmark of packet-processing performance. Consequently, metrics such as maximum forwarding throughput or packet-processing capacity under saturated traffic conditions are outside the scope of the present study and remain part of future work.

3.5 SHAP Component Description

Explainability was incorporated using SHAP to improve the transparency of the AI decision-making process and to support both operational trust and academic interpretability. For each prediction, the system records the most influential features together with their direction and relative contribution. Security analysts can understand not only why a source is considered suspicious but also which network characteristics contributed most to the prediction, enabling them to assess whether model decisions are driven by meaningful network behavior rather than by irrelevant statistical artifacts. SHAP addresses this requirement by providing feature-level attribution for individual predictions. In the present work, representative influential features include packet-length statistics, TCP flag counts, and window-related flow descriptors, which consistently contributed to malicious classifications during experimental evaluation [30,31]. In the proposed system, SHAP analysis is performed using TreeExplainer on the RF classifier. The resulting explanations characterize only the contribution of the RF classifier and are discussed in further [Section 4.3.2](#). The final blocking decision is produced by the hybrid RF–LSTM model according to [Eq. \(3\)](#), meaning that SHAP does not directly explain the complete hybrid decision. Extending explainability to the full hybrid model remains future work.

3.6 Dataset and Model Training

Both the RF and LSTM models were trained on the CICIDS2017 dataset, one of the most widely used benchmarks for intrusion detection research [32]. The dataset contains labeled benign and malicious traffic generated across diverse attack scenarios and normal network activities. Each sample is represented by statistical flow features, including flow duration, bidirectional packet and byte statistics, TCP flag counters, packet-length metrics, and aggregated communication descriptors. This representation maintains a compact input for supervised learning while avoiding the computational overhead of packet-level analysis [33].

The experimental validation presented in this study is based on the implemented deployment scenarios, whereas model training relied on the broader traffic diversity available in CICIDS2017. The reported results should be interpreted as a proof-of-concept validation of the proposed framework rather than as evidence of generalization across diverse deployment environments. Extending the evaluation with additional datasets and real-world deployment scenarios remains part of future work.

3.7 Reproducibility and Model Configuration

This section offers the implementation details necessary to reproduce the implementation, including the dataset composition, preprocessing pipeline, model hyperparameters, evaluation procedure, and deployment configuration. For the training phase, the RF model was developed using traffic records extracted from the Monday-WorkingHours, Tuesday-WorkingHours, Thursday-WorkingHours-Morning-WebAttacks, Thursday-WorkingHours-Afternoon-Infiltration, Friday-WorkingHours-Morning, and Friday-WorkingHours-Afternoon-PortScan subsets of the CICIDS2017 dataset. The LSTM model was trained using data obtained from the Monday-WorkingHours, Thursday-WorkingHours-Morning-WebAttacks, Thursday-WorkingHours-Afternoon-Infiltration, Friday-WorkingHours-Morning, Friday-WorkingHours-Afternoon-PortScan, and Friday-WorkingHours-Afternoon-DDos subsets. The partially different subset composition reflects the resource-constrained model-development configuration

used for the two learning components. Consequently, the RF and LSTM models were exposed to partially different training distributions, which may have influenced their individual learned representations and standalone performance. Therefore, their standalone results should not be interpreted as a strictly controlled RF-vs.-LSTM comparison under identical training conditions.

The final offline evaluation reported in [Section 4.2](#) was conducted on a common evaluation set constructed from the CICIDS2017 subsets shared by both training pipelines, providing consistent evaluation conditions for the RF, LSTM, and Hybrid RF–LSTM classifiers. All three models were evaluated on the same 293,021 samples. For the hybrid model, the RF classifier was applied to the final flow record of each LSTM sequence, ensuring that the RF predictions, LSTM predictions, and corresponding ground-truth labels remained aligned. During preprocessing, the CSV files were imported using Pandas and restricted to the fixed 78 numerical CICFlowMeter features used by the models together with the class label; dataset fields outside this predefined model representation were excluded because they were not part of the numerical input schema used for model training and inference. Infinite values were replaced with NaN values and incomplete records were removed. The retained representation includes flow-duration metrics, bidirectional packet and byte statistics, inter-arrival times, TCP flag counters, packet-length statistics, and connection-level traffic descriptors. The complete list of the 78 features and their fixed input ordering is explicitly defined in the publicly available training implementation referenced in [\[34\]](#). Feature standardization was performed using a StandardScaler generated during the RF preprocessing stage and subsequently reused by the LSTM and runtime inference pipelines. During deployment, the feature-extraction module initializes the fixed-length input vector with a predefined raw value of 0.0 for each feature; runtime descriptors that are available replace the corresponding defaults. The resulting vector is then transformed using the stored StandardScaler before inference. The RF classifier was configured with 100 decision trees ($n_estimators = 100$) and a maximum depth of 20 ($max_depth = 20$). For RF model development, the preprocessed samples were randomly partitioned into 80% for training and 20% for internal testing using $random_state = 42$. The LSTM model used two stacked recurrent layers with 64 and 32 neurons, each followed by dropout (0.2), and a fully connected layer with 16 neurons preceding the sigmoid output neuron used for binary classification. Temporal sequences of length 10 were constructed before model fitting. LSTM training used the Adam optimizer and binary cross-entropy loss for 10 epochs with a batch size of 1024. Keras validation_split = 0.2 was applied to the resulting sequences during model fitting, corresponding to 80% for training and 20% for validation. Thus, the LSTM validation fraction was not drawn from the 80% RF training partition. These model-development partitions are distinct from the common evaluation set described above and used for the final RF, LSTM, and Hybrid RF–LSTM comparison reported in [Section 4.2](#). The hybrid decision score was computed as the arithmetic mean of the probabilistic outputs produced by the RF and LSTM models, assigning equal weights to both classifiers ($w_{RF} = 0.5$ and $w_{LSTM} = 0.5$). Offline evaluation employed a decision threshold of $\tau = 0.50$, whereas deployment used a configurable runtime threshold (τ) that can be adjusted without retraining the models. The source code and implementation artifacts required to reproduce the framework have been publicly released on GitHub and archived on Zenodo [\[34\]](#).

4 Results and Discussion

4.1 Experimental Environment and Infrastructure

The proposed AI-driven hybrid intrusion prevention system was implemented and evaluated in a controlled virtualized environment using VMware Workstation. All experimental nodes executed Ubuntu 24.04.4 LTS with Linux kernel version 6.17.0-19-generic and Python 3.12.3. The software environment included NumPy, Pandas, scikit-learn, TensorFlow/Keras, SHAP, Flask, Joblib, Psutil, Requests, Jinja2, Matplotlib, together with YAML-based configuration support. These components provided the functionality

required for the different layers of the framework, including machine-learning inference, sequence modeling, explainability, runtime monitoring, web-based visualization, and model deployment. The experimental infrastructure was designed to provide a reproducible execution environment while preserving a consistent state throughout the evaluation campaign. The modular organization of the framework allows different event-generation mechanisms to operate with the same AI decision engine and the same kernel-level mitigation stage, making the architecture independent of a particular detection element and facilitating future integration of additional detectors. To support independent verification, the public repository includes the implementation, configuration files, AI decision logs, enforcement records, monitoring data, and complete experimental artifacts collected during the evaluation campaign. Infrastructure-specific elements, such as virtual-machine provisioning and network topology, may require adaptation in other deployment environments. The AI-IPS node used the ens192 network interface driven by the VMware vmxnet3 virtual network driver (version 1.9.0.0-k-NAPI). The virtual machine operated within an isolated VMware network (172.32.32.0/20), where the AI-IPS host was assigned the address 172.32.32.7 and the gateway address 172.32.32.1.

The eBPF/XDP program was attached to the ens192 interface in native driver mode, as verified using `bpftool net`. The XDP program and its associated blacklist map were pinned under `/sys/fs/bpf/` as `/sys/fs/bpf/xdp_prog` and `/sys/fs/bpf/xdp_block`, while the compatibility symlink `/sys/fs/bpf/blacklist` referenced the same pinned map. The primary AI-IPS node was configured with eight virtual CPU cores (Intel Xeon Gold 6226R @ 2.90 GHz, x86_64 architecture), a 25 GB root filesystem, and 3.8 GB of swap space. During development, the virtual machine was initially provisioned with 16 GB of RAM; however, offline LSTM training resulted in memory exhaustion and swap-related interruptions. Increasing the available memory to 24 GB enabled stable model training without further memory-related failures. This observation illustrates an important practical aspect of AI-enabled security systems: the computational resources required for offline model training should not be confused with the considerably lower resource requirements associated with online inference during deployment. The experimental campaign employed three virtual machines: an attacker generating benign and malicious traffic, an AI-IPS node implementing the proposed framework, and a reference node running a conventional Fail2Ban deployment. Separating traffic generation from the evaluated systems prevented measurement interference and ensured that the collected metrics reflected only the behavior of the defense mechanisms. To improve the statistical reliability of the evaluation, each experimental configuration was executed independently 30 times under identical operating conditions. The campaign included four representative scenarios: benign traffic (656 s), SSH brute-force attacks (559 s), slow SSH brute-force attacks (647 s), and port-scanning activities (60 s). Considering both the proposed framework and the reference baseline, the complete evaluation comprised 240 independent experimental executions. Before each repetition, the experimental environment was restored to a predefined initial state by resetting temporary logs, clearing the kernel blacklist map, removing residual blocking entries, and restarting the required monitoring services. This procedure ensured that every execution started from identical operating conditions and prevented residual state from influencing subsequent measurements. The experimental environment enabled a comparative evaluation of defensive behavior, mitigation effectiveness, and runtime resource consumption.

The analysis considered mitigation latency, system-level and application-level CPU and memory utilization, together with the deployment characteristics of XDP-based kernel enforcement. The evaluation was intentionally conducted in a controlled virtualized environment to maximize reproducibility and allow detailed observation of the interaction between AI-driven decision, explainability mechanisms, and kernel-level packet enforcement. Although the campaign comprised 240 independent executions, the evaluation

represents a proof-of-concept validation conducted in a controlled virtualized environment. Larger and more heterogeneous deployments remain the subject of future work.

4.2 Model Evaluation

The offline classification performance of the proposed RF, LSTM, and Hybrid RF–LSTM models was evaluated using a common evaluation set derived from the CICIDS2017 dataset. To ensure a fair comparison, all models were assessed on the same 293,021 labeled flow samples, where benign traffic was assigned to class 0, and all attack categories were aggregated into the malicious class labeled 1. Standard binary classification metrics, including Accuracy, Precision, Recall, F1-score, ROC-AUC, and confusion matrices, were used for evaluation. Following common practice for probabilistic classifiers, the offline assessment employed a decision threshold of $\tau = 0.50$. The evaluation characterizes the intrinsic classification capability of the models, whereas the operational experiments presented in Section 4.3 employ a lower deployment threshold ($\tau = 0.01$) to prioritize early attack interception within the AI-driven intrusion prevention system. As summarized in Table 2, the Hybrid RF–LSTM achieved the highest Accuracy (99.33%), Precision (97.71%), F1-score (96.95%), and ROC-AUC (99.33%) on the common CICIDS2017 evaluation set, whereas the standalone LSTM achieved the highest Recall (97.29%). The RF classifier also demonstrated consistently strong performance, providing a balanced combination of Precision (96.11%) and Recall (96.37%). Overall, at the conventional offline decision threshold, the hybrid RF–LSTM model provided the most balanced trade-off between precision and recall on the common evaluation set, supporting its selection as the inference engine of the proposed framework. The corresponding confusion matrices for the RF, LSTM, and Hybrid RF–LSTM models are presented in Fig. 2.

Table 2: Performance evaluation of the models on the common CICIDS2017 evaluation set (293,021 samples, $\tau = 0.50$).

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
RF	99.16%	96.11%	96.37%	96.24%	98.15%
LSTM	97.42%	82.62%	97.29%	89.36%	98.83%
Hybrid RF-LSTM	99.33%	97.71%	96.21%	96.95%	99.33%

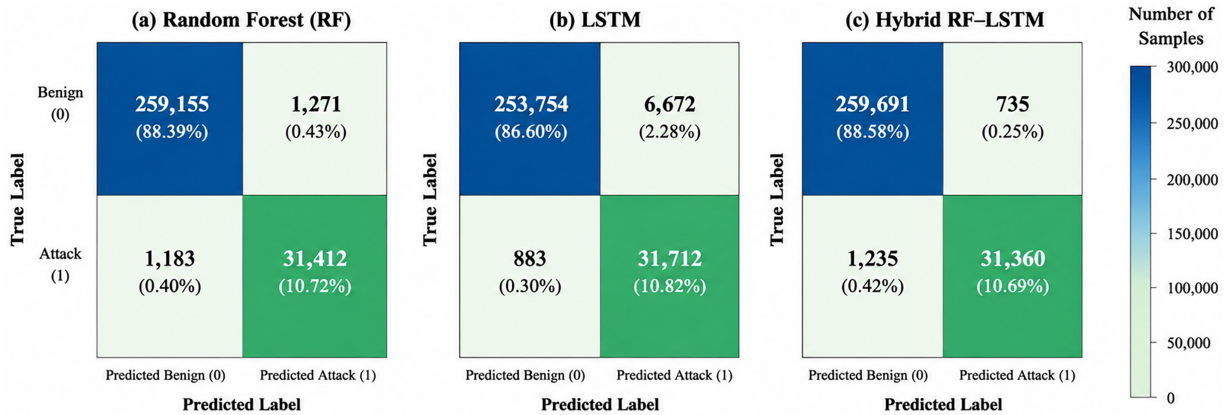


Figure 2: Confusion matrices of the evaluated models on the common CICIDS2017 evaluation set (293,021 Samples).

The confusion matrices provide additional insight into the classification behavior of the evaluated models. The RF classifier correctly identified 259,155 benign and 31,412 malicious samples while producing

1271 false positives and 1183 false negatives. The standalone LSTM achieved the highest Recall by correctly identifying 31,712 malicious samples; however, this improvement was accompanied by 6672 false positives. In contrast, the Hybrid RF–LSTM correctly classified 259,691 benign and 31,360 malicious samples while reducing the number of false positives to only 735, representing an approximately 42% reduction compared with the standalone RF classifier, while maintaining a comparable Recall. This reduction in false-positive decisions directly contributes to the higher Precision, F1-score, and ROC-AUC observed for the Hybrid model, supporting its selection as the inference engine of the architecture.

Table 3 places the proposed framework in the context of representative CICIDS2017-based machine-learning intrusion detection studies. Although several reported approaches achieve classification accuracies comparable to, or in some cases slightly higher than, those obtained by the framework, most focus exclusively on offline intrusion detection. For comparison purposes, Table 3 reports the performance of the best-performing configuration identified in the present study, namely the Hybrid RF–LSTM model, whereas the detailed evaluation of the RF, LSTM, and Hybrid RF–LSTM models is provided in Table 2. The studies summarized in Table 3 primarily evaluate offline intrusion detection performance. This work focuses on the implementation and operational validation of a modular intrusion prevention architecture that combines AI inference, explainability, and automated kernel-level enforcement. The comparison presented in Table 3 should be understood as situating the classification abilities of the proposed AI components within a broader context, rather than implying a direct equivalence between fully developed intrusion prevention systems. The performance values reported in Table 2 were obtained using the conventional decision threshold of ($\tau = 0.50$), which is commonly adopted for offline evaluation of probabilistic classifiers. However, the implementation is intended for operational intrusion prevention, where timely attack interception may be preferred over maximizing offline classification metrics.

Table 3: Representative CICIDS2017-based machine-learning intrusion detection and prevention approaches.

Study	Dataset	Model	Accuracy (%)	Explainability	Active Prevention
[35]	CICIDS2017	Hierarchical DT + Rules-Based IDS	99.0+	No	No
[36]	CICIDS2017	CNN-LSTM	99.64	No	No
[37]	CICIDS2017	Random Forest	99.80	No	No
Proposed Framework	CICIDS2017	RF + LSTM + SHAP + eBPF/XDP	99.33	Yes	Yes

To examine this trade-off, all three models were evaluated using both the operational deployment threshold ($\tau = 0.01$) and the conventional offline threshold ($\tau = 0.50$). As expected, reducing the decision threshold increases Recall at the expense of Precision by allowing a larger number of flows to be classified as malicious. As summarized in Table 4, the conventional offline threshold ($\tau = 0.50$) provides the highest overall classification performance, whereas the lower operational threshold ($\tau = 0.01$) increases Recall from 96.21% to 98.33%, enabling earlier identification of malicious traffic while increasing the number of false-positive decisions. Consequently, the threshold adopted during deployment should be interpreted as an operational design choice that prioritizes early attack interception rather than as the threshold maximizing offline classification performance.

Table 4: Threshold sensitivity analysis on the common evaluation set.

Threshold	Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
0.01	RF	97.93	86.48	96.47	91.21
0.01	LSTM	94.32	66.52	98.44	79.39
0.01	Hybrid RF-LSTM	95.15	70.11	98.33	81.85
0.50	RF	99.16	96.11	96.37	96.24
0.50	LSTM	97.42	82.62	97.29	89.36
0.50	Hybrid RF-LSTM	99.33	97.71	96.21	96.95

4.3 Operational Evaluation of the Proposed Framework

The operational evaluation validates the modular AI-driven intrusion prevention framework under four representative scenarios: benign traffic, SSH brute-force attacks, slow SSH brute-force attacks, and port-scanning activity. Rather than evaluating a single attack detector, the experiments assess a unified deployment architecture in which heterogeneous attack-specific detectors generate standardized security events that are processed through a common AI-driven decision, explainability, and kernel-level enforcement pipeline. In the current implementation, authentication attacks are handled by the widely adopted Fail2Ban authentication detector, whereas reconnaissance activity is identified by a dedicated port-scanning detector. Despite relying on different detection mechanisms, both detectors interface with the same hybrid RF-LSTM inference engine, SHAP explainability module, and eBPF/XDP enforcement subsystem. This architectural separation enables new attack-specific detectors to be integrated without modifying the AI decision engine or the underlying enforcement mechanism. The following sections evaluate the operational behavior, explainability, runtime characteristics, and architectural properties of the framework under representative attack scenarios.

4.3.1 Modular Detection and AI-Driven Mitigation

The architecture was evaluated under four representative deployment scenarios comprising benign traffic, SSH brute-force attacks, slow SSH brute-force attacks, and port-scanning activity. Fig. 3 compares operational mitigation latency measured over 30 independent executions for each evaluated scenario. The reported values correspond to the elapsed time between attack initiation and the first successful mitigation event, while the error bars represent one standard deviation computed across the repeated experimental runs.

Authentication attacks exhibited stable operational behavior across repeated executions, with mean mitigation latencies of 8.01 s for SSH brute-force attacks and 58.72 s for slow SSH brute-force attacks. The longer response in the slow brute-force scenario reflects the reduced rate at which authentication events accumulate before the detector generates a security event. For the same attack scenarios, the reference Fail2Ban deployment achieved lower mitigation latency because blocking decisions were applied immediately after the detector threshold was reached. This difference reflects the additional processing stages of the proposed framework, including feature extraction, hybrid inference, explainability generation, and eBPF/XDP-based enforcement. The dedicated reconnaissance detector identified port-scanning activity with a mean mitigation latency of 5.20 s, although greater variability was observed because detection depended on the temporal distribution of scanning packets during individual executions.

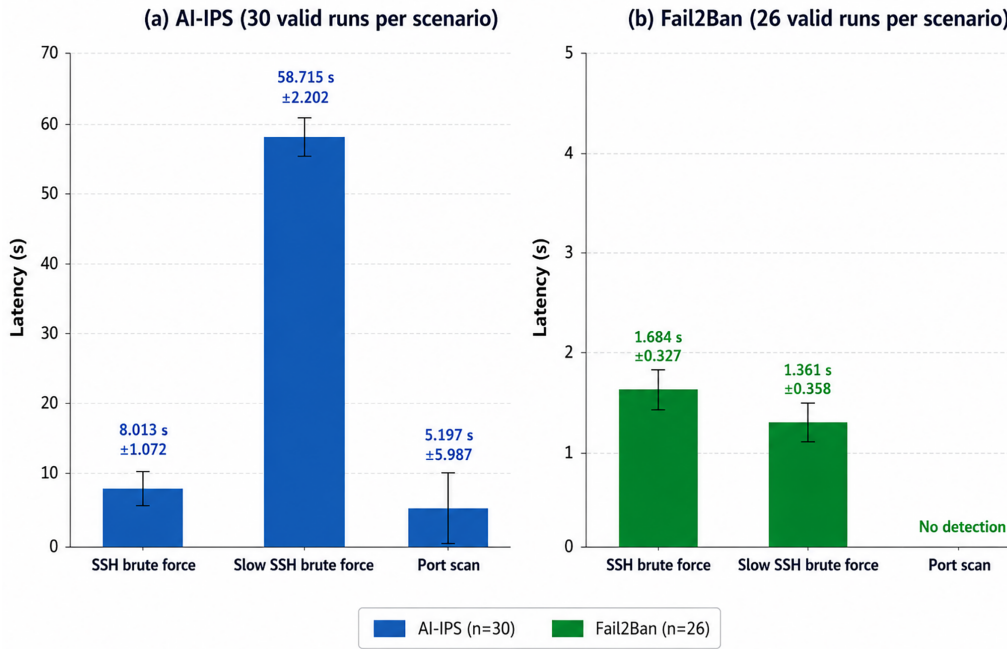


Figure 3: Operational mitigation latency comparison.

4.3.2 Explainability of Operational AI Decisions

Unlike conventional intrusion prevention systems, this system supports decision transparency during operation through the integration of SHAP-based explanations into the AI inference pipeline. Consequently, each mitigation decision can be accompanied by an explanation of the feature contributions that influenced the RF component of the hybrid classifier, allowing operators to inspect the rationale behind individual blocking decisions. The explainability analysis combines complementary global and local perspectives. Global SHAP analysis summarizes the behavior of the deployed RF model across the operational campaign, whereas local explanations displayed through the dashboard provide feature-level interpretation for individual AI decisions. During runtime, the AI engine computes local SHAP explanations for each inference request and exposes the most influential RF features to the dashboard, enabling the operator to inspect the rationale behind individual decisions. Fig. 4a presents the global SHAP summary obtained from the feature vectors collected during the experiments using the deployed RF model and the same preprocessing pipeline adopted during inference. The analysis included SSH brute-force, slow SSH brute-force, and port-scanning scenarios, comprising a total of 178 recorded feature vectors. The resulting importance pattern was stable across all evaluated scenarios, with PSH Flag Count, Subflow Fwd Bytes, Packet Length Mean, Total Length of Fwd Packets, and Flow Bytes/s emerging as the most influential features. Complementing this global perspective, Fig. 4b illustrates the dashboard, where instance-level SHAP explanations provide feature-level contributions for an individual AI decision.

The integration of explainability directly into the deployment workflow extends the implementation beyond conventional rule-based intrusion prevention systems by enabling security analysts to associate mitigation actions with interpretable AI decisions. This capability facilitates operational validation, supports forensic analysis, and provides a transparent interface for future detector modules integrated into the same AI-driven decision pipeline.

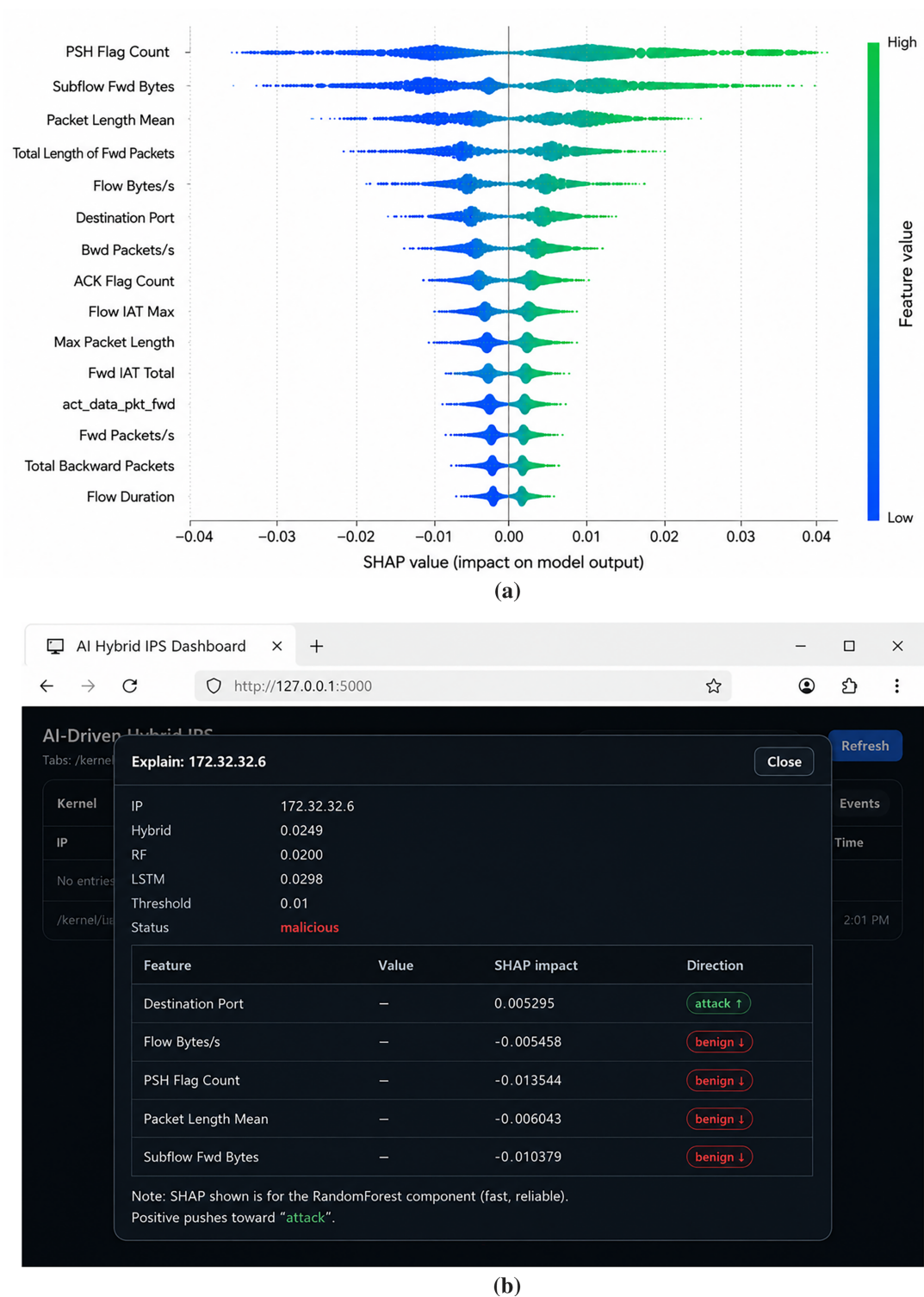


Figure 4: SHAP-based explainability in the proposed AI-driven IPS: (a) global SHAP summary during the operational experiments; (b) runtime dashboard showing local SHAP contributions for an individual AI decision.

4.3.3 Runtime Resource and Enforcement Characteristics

Beyond detection performance and decision explainability, the evaluation also investigated the run-time characteristics of the framework. The analysis considered computational resource utilization during representative scenarios together with the behavior of the kernel-level enforcement mechanism. These measurements characterize the practical overhead associated with the modular AI-driven architecture rather than the performance of an individual detector. CPU utilization was evaluated throughout the complete experimental campaign to quantify the computational overhead introduced by the AI-IPS framework. The measurements were collected over 30 independent executions of each experimental scenario and are reported as mean values with the corresponding standard deviations. To provide a comprehensive view of the deployment cost, CPU utilization was analyzed from two complementary perspectives: (i) the protected system as a whole and (ii) the intrusion-prevention solution itself. For both perspectives, average and peak CPU utilization were considered. The comparative system-level and solution-level CPU utilization measurements for the AI-IPS framework and the reference Fail2Ban deployment are presented in Fig. 5.

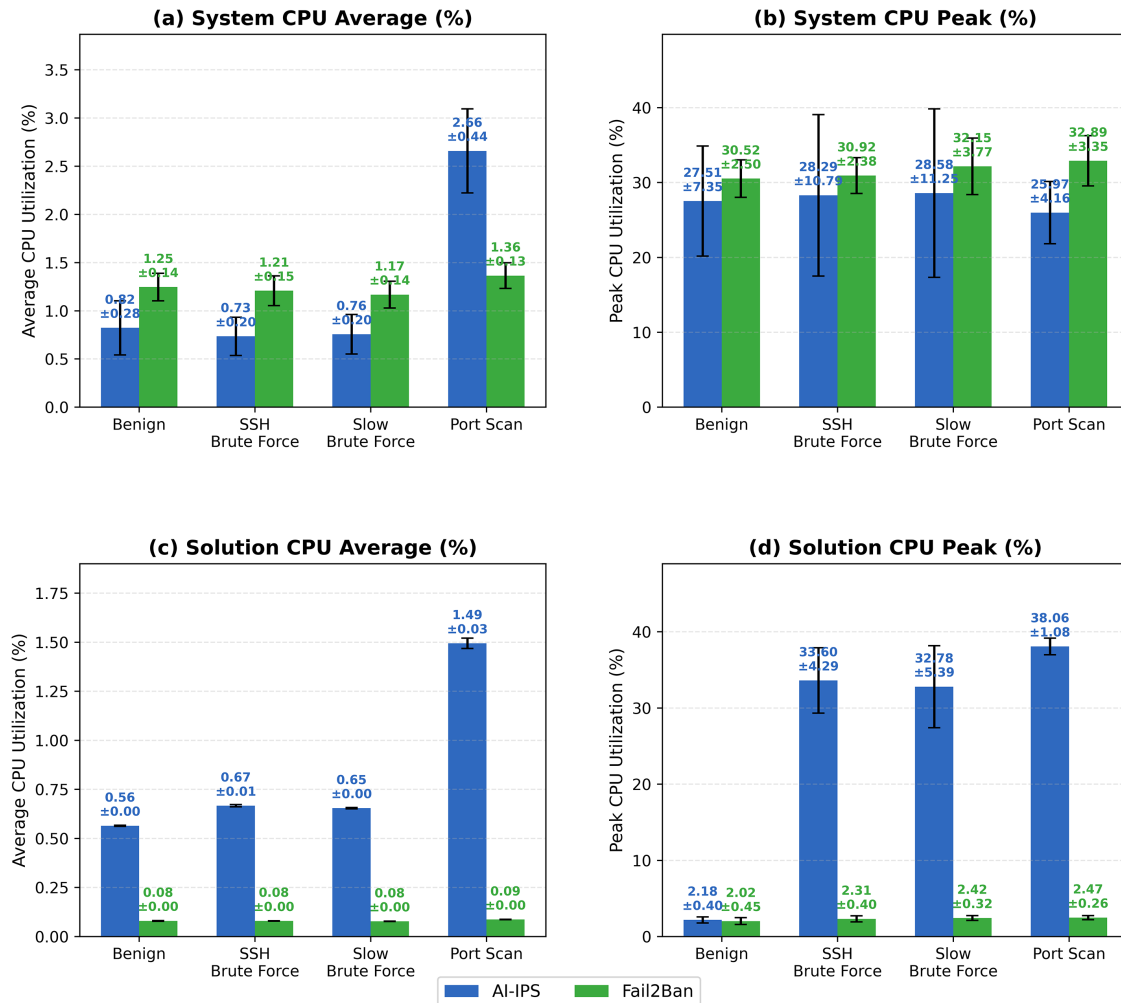


Figure 5: Comparative CPU utilization of the AI-IPS system and the reference Fail2Ban deployment across the evaluated experimental scenarios.

The figure reports the average and peak CPU utilization measured for both the protected system and the intrusion-prevention solution over 30 independent experimental repetitions. Error bars indicate one standard deviation. As shown in Fig. 5a,b, the average system-level CPU utilization remained below 3% for all evaluated scenarios, although the AI-IPS framework consistently required higher processing resources than the reference Fail2Ban deployment. The largest increase was observed during the port-scanning scenario, reflecting the additional processing associated with the dedicated reconnaissance detector and the AI-based decision pipeline. The solution-level measurements presented in Fig. 5c,d further isolate the computational contribution of the intrusion-prevention components.

Unlike the lightweight Fail2Ban implementation, the proposed framework performs feature extraction, hybrid RF-LSTM inference, SHAP explanation generation, REST-based communication between architectural modules, and management of the eBPF/XDP enforcement infrastructure. The observed increase in CPU utilization reflects the additional processing stages introduced by the architecture. Memory utilization was monitored throughout the entire experimental campaign to quantify the runtime memory requirements of the AI-IPS framework relative to the reference Fail2Ban deployment. Measurements were collected over 30 independent executions of each experimental scenario and are reported as mean values together with the corresponding standard deviations. Fig. 6a presents the average memory consumption, whereas Fig. 6b summarizes the variability observed across repeated experimental runs.

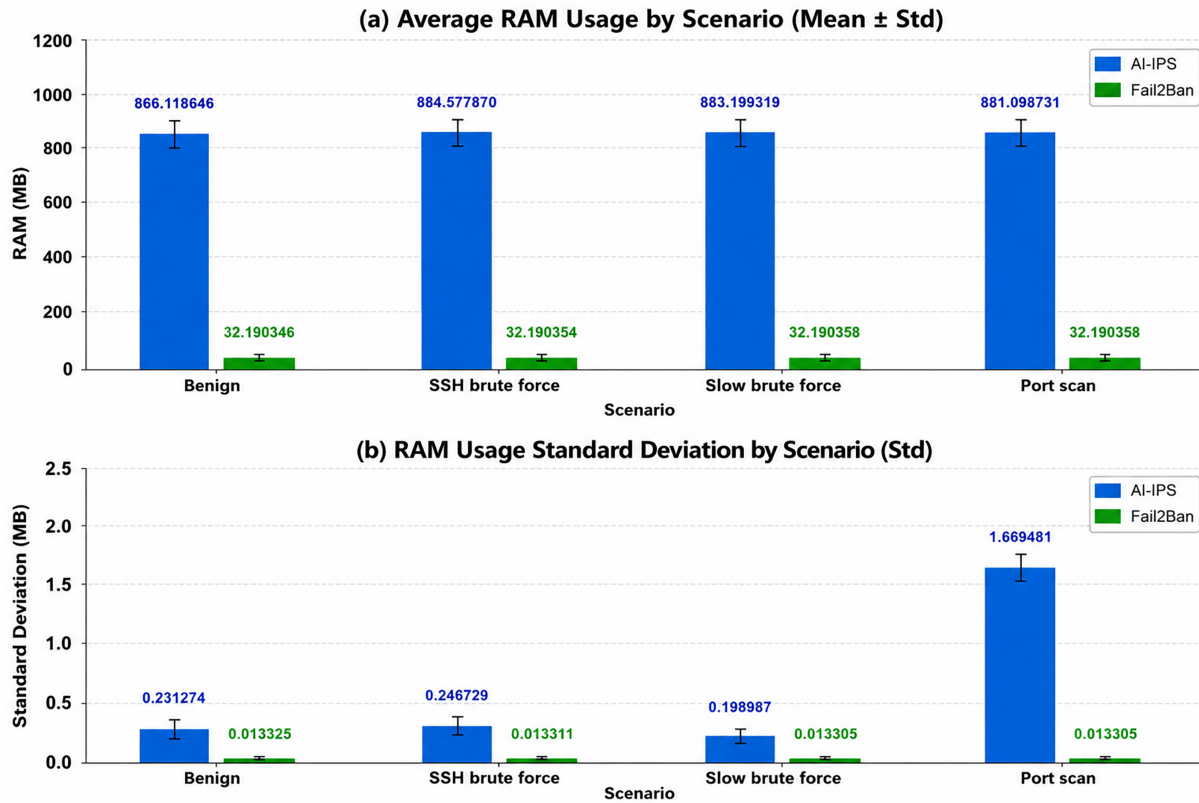


Figure 6: Comparative memory utilization of the AI-IPS framework and the reference Fail2Ban deployment across the evaluated experimental scenarios. (a) Mean memory consumption measured over 30 independent executions of each scenario. (b) Standard deviation of the measured memory consumption across the same experimental repetitions.

As shown in Fig. 6a, the AI-IPS system maintained a nearly constant memory footprint throughout the experimental campaign, with average memory utilization ranging from approximately 866 MB during benign operation to approximately 885 MB under attack conditions. The relatively small differences between scenarios indicate that runtime memory requirements are primarily determined by the software architecture rather than by the characteristics of the attacks evaluated. In contrast, the reference Fail2Ban deployment maintained an almost constant memory footprint of approximately 32 MB in all scenarios, reflecting its lightweight log-monitoring architecture. Fig. 6b characterizes the repeatability of the memory measurements by presenting the standard deviation computed from the 30 independent executions of each scenario. For both AI-IPS and Fail2Ban, the standard deviations remained low, indicating highly consistent memory utilization across repeated experiments. Although the port-scanning scenario produced the largest variation for AI-IPS, the observed standard deviation (1.67 MB) represents less than 0.2% of the average memory footprint, demonstrating that the increase reflects normal runtime variability rather than unstable memory behavior or progressive memory growth. The memory footprint is dominated by the reusable AI infrastructure rather than by the specific detector or attack scenario.

To further characterize the runtime behavior of the architecture, a dedicated benchmark was performed to evaluate the kernel-level enforcement mechanism independently of attack detection. The benchmark compared the eBPF/XDP implementation with a conventional Fail2Ban deployment, focusing exclusively on the enforcement stage. The results are summarized in Table 5.

Table 5: Benchmark comparison of the proposed eBPF/XDP enforcement mechanism and conventional Fail2Ban enforcement.

Metric	eBPF/XDP	Fail2Ban
Rule installation time (ms)	33	103
Client-observed transition (ms)	62	63
Mean CPU utilization (%)	3.55	5.94
Peak CPU utilization (%)	29.70	16.10
Client behavior after enforcement	Timeout (DROP)	Connection refused (REJECT)

The benchmark demonstrates that updating the eBPF blacklist required approximately one third of the rule installation time observed for the reference Fail2Ban deployment (33 vs. 103 ms). Although the client-observed transition between the last connection and the first blocked attempt remained nearly identical (62 vs. 63 ms), the kernel-level implementation exhibited lower mean CPU utilization than Fail2Ban during the enforcement phase (3.55% vs. 5.94%). However, eBPF/XDP also exhibited a higher observed peak CPU utilization (29.70% vs. 16.10%), indicating a trade-off between lower average utilization and higher short-duration CPU peaks rather than a uniform computational advantage. The two mechanisms also differed in their observable network behavior: eBPF/XDP silently discarded packets at the XDP layer, resulting in connection timeouts, whereas Fail2Ban rejected subsequent connections through firewall rules, producing explicit connection refusal responses. Taken together, the CPU, memory, and enforcement measurements indicate that the implementation introduces additional runtime overhead as a consequence of integrating machine-learning inference, explainability, and kernel-level mitigation within a common pipeline. This overhead represents the cost of providing a reusable AI-driven infrastructure capable of integrating heterogeneous detectors while maintaining a unified decision and enforcement architecture.

4.3.4 Discussion

The evaluation indicates that the framework can process standardized security events generated by different attack-specific detectors within a common AI-driven inference and enforcement workflow. Authentication attacks detected through Fail2Ban and reconnaissance activity identified by the dedicated port-scanning detector were both handled using the same feature-extraction, hybrid RF–LSTM inference, explainability, and kernel-level enforcement pipeline. These results suggest that separating attack-specific detection from the subsequent processing stages provides a practical basis for extending the framework with additional detectors while preserving a consistent operational workflow. The experimental results also illustrate the trade-off associated with this design. Compared with the reference Fail2Ban deployment, the proposed framework requires additional computational resources because feature extraction, AI inference, explainability, dashboard services, and kernel-level orchestration are performed in addition to attack detection. This overhead reflects the cost of the reusable infrastructure rather than the behavior of an individual detector. In return, the framework supports a common decision and enforcement pipeline that can be shared by different detection mechanisms. Another aspect highlighted by the experiments is the integration of explainability into the deployment workflow. The SHAP analysis complements the Random Forest component by exposing the most influential features associated with individual blocking decisions, providing additional context for interpreting AI-generated predictions without modifying the mitigation process. The present evaluation was conducted in a controlled virtualized environment using representative authentication and reconnaissance scenarios. Although the results support the feasibility of the proposed workflow, additional validation under larger-scale deployments, more diverse traffic conditions, and broader attack categories would provide a more comprehensive assessment of its applicability.

5 Conclusions

This paper presented a modular AI-driven intrusion prevention framework integrating attack-specific detection, hybrid RF–LSTM inference, SHAP-based explainability, and eBPF/XDP kernel-level enforcement within a reusable deployment architecture. The proposed architecture demonstrates that authentication-oriented and reconnaissance detectors can operate within a shared AI-driven decision and enforcement workflow while remaining independent of the underlying detection mechanism. Experimental evaluation demonstrated the practical feasibility of the architecture under representative authentication and reconnaissance scenarios. The experimental results showed that the framework integrated heterogeneous event sources into a common deployment pipeline while providing interpretable AI-driven decisions, automated mitigation, and predictable runtime behavior. The experiments also characterized the computational cost associated with the additional AI inference, explainability, and orchestration components, providing a transparent assessment of the architectural trade-offs introduced by the proposed design. Unlike many studies that primarily report offline classification performance, this work emphasizes the implementation and experimental validation of a modular intrusion prevention architecture integrating hybrid AI inference, explainability, and kernel-level enforcement. The principal contribution lies in the proposed architecture, while the hybrid RF–LSTM model serves as the decision-making component within this broader framework. The current implementation represents a proof-of-concept validation conducted in a controlled virtualized environment using representative attack scenarios. Future work will focus on extending the framework with additional attack-specific detectors, evaluating larger and more heterogeneous deployment environments, investigating adaptive decision strategies, and exploring explainability techniques applicable to the complete hybrid RF–LSTM decision process.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: conceptualization, Cătălin-Eugen Bucur; methodology, Cătălin-Eugen Bucur; software, Cătălin-Eugen Bucur; validation, Cătălin-Eugen Bucur; formal analysis, Cătălin-Eugen Bucur, Elena-Gratiela Robe-Voinea; investigation, Cătălin-Eugen Bucur; resources, Cătălin-Eugen Bucur; data curation, Cătălin-Eugen Bucur; writing—original draft preparation, Cătălin-Eugen Bucur; writing—review and editing, Georgiana Crihan, Elena-Gratiela Robe-Voinea; visualization, Cătălin-Eugen Bucur, Georgiana Crihan, Elena-Gratiela Robe-Voinea; project administration, Cătălin-Eugen Bucur. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The implementation artifacts associated with this study have been publicly released and archived to support reproducibility. The repository includes the source code, RF and LSTM training scripts, trained models, preprocessing artifacts, feature-order metadata, configuration files, eBPF/XDP enforcement code, Fail2Ban integration modules, attack-scenario documentation, reproducibility documentation, and raw experimental artifacts collected during the evaluation campaign. The released artifacts cover both the AI-IPS framework and the Fail2Ban baseline across the benign traffic, SSH brute-force, slow brute-force, and port-scanning scenarios evaluated in this work. Available at: Zenodo Archive (DOI): [34], GitHub Repository: [38].

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Aslan Ö, Aktuğ SS, Ozkan-Okay M, Yilmaz AA, Akin E. A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. *Electronics*. 2023;12(6):1333. doi:10.3390/electronics12061333.
2. Owoko W. Survey on evolving threats in TCP/IP header attacks: emerging trends and future directions. *World J Adv Eng Technol Sci*. 2024;11(2):454–75. doi:10.30574/wjaets.2024.11.2.0127.
3. Xu D, Gondal I, Yi X, Susnjak T, Watters P, McIntosh TR. The erosion of cybersecurity zero-trust principles through generative AI: a survey on the challenges and future directions. *J Cybersecur Priv*. 2025;5(4):87. doi:10.3390/jcp5040087.
4. Mao J, Yang X, Hu B, Lu Y, Yin G. Intrusion detection system based on multi-level feature extraction and inductive network. *Electronics*. 2025;14(1):189. doi:10.3390/electronics14010189.
5. Vladov S, Mulesa O, Vysotska V, Horvat P, Paziura N, Kolobylna O, et al. Method for detecting low-intensity DDoS attacks based on a combined neural network and its application in law enforcement activities. *Data*. 2025;10(11):173. doi:10.3390/data10110173.
6. Yuan J, Wang C, Zhou H, Zhang Y, Wang Y. DLogParser: an efficient dynamic log parser with multiple grouping criteria. *Appl Sci*. 2026;16(2):811. doi:10.3390/app16020811.
7. Feng Y, Sakurai K. Network intrusion detection: evolution from conventional approaches to LLM collaboration and emerging risks. *arXiv:2510.23313*. 2025. doi:10.48550/arxiv.2510.23313.
8. Wen-Udeoji SC, Mueyba MK, Mohammadi A. Comparative evaluation of machine learning and signature-based NIDS for multi-class and binary threat detection. In: *Data science, AI and applications*. Berlin/Heidelberg, Germany: Springer; 2025. p. 238–51. doi:10.1007/978-3-032-11352-8_17.
9. Fail2Ban Developers. Fail2Ban documentation. 2026 [cited 2026 Jan 1]. Available from: <https://fail2ban.readthedocs.io/en/latest/>.
10. Ali ML, Thakur K, Schmeelk S, Debello J, Dragos D. Deep learning vs. machine learning for intrusion detection in computer networks: a comparative study. *Appl Sci*. 2025;15(4):1903. doi:10.3390/app15041903.
11. Shiri FM, Perumal T, Mustapha N, Mohamed R. A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU. *arXiv:2305.17473*. 2023. doi:10.48550/arXiv.2305.17473.

12. Joshi VR, Assa-Agyei K, Al-Hadhrami T, Qasem SN. Hybrid AI intrusion detection: balancing accuracy and efficiency. *Sensors*. 2025;25(24):7564. doi:10.3390/s25247564.
13. Chinnasamy P, Yarramsetti S, Ayyasamy RK, Rajesh E, Vijayasaro V, Pandey D, et al. AI-Driven intrusion detection and prevention systems to safeguard 6G networks from cyber threats. *Sci Rep*. 2025;15(1):37901. doi:10.1038/s41598-025-21648-5.
14. Saeed H, Naseer M, Rasool A, Alsirhani A, Alserhani F, Alwakid GN, et al. A novel adaptive hybrid intrusion detection system with lightweight optimization for enhanced security in internet of medical things. *Sci Rep*. 2025;16(1):2097. doi:10.1038/s41598-025-31897-z.
15. Patil S, Varadarajan V, Mazhar SM, Sahibzada A, Ahmed N, Sinha O, et al. Explainable artificial intelligence for intrusion detection system. *Electronics*. 2022;11(19):3079. doi:10.3390/electronics11193079.
16. Ahmad T. AI-driven dynamic firewall optimization using reinforcement learning for anomaly detection and prevention. *arXiv:2506.05356*. 2025.
17. Mohale VZ, Obagbuwa IC. Evaluating machine learning-based intrusion detection systems with explainable AI: enhancing transparency and interpretability. *Front Comput Sci*. 2025;7:e4150. doi:10.3389/fcomp.2025.1520741.
18. Muhammad AE, Yow KC, Baćanin-Džakula N, Khan MA. L-xaids: a LIME-based eXplainable AI framework for intrusion detection systems. *Clust Comput*. 2025;28(10):654. doi:10.1007/s10586-025-05326-9.
19. Khraisat A, Talukder MA, Uddin MA, Alazab A. RF-FedAvg: federated learning-based random forest model for intrusion detection in wireless sensor networks. *Clust Comput*. 2025;28(13):873. doi:10.1007/s10586-025-05591-8.
20. Khan N, Ahmad K, Al Tamimi A, Alani MM, Bermak A, Khalil I. Explainable AI-based intrusion detection systems for industry 5.0 and adversarial XAI: a systematic review. *Information*. 2025;16(12):1036. doi:10.3390/info16121036.
21. Ahmed U, Jiangbin Z, Almogren A, Sadiq M, Rehman AU, Sadiq MT, et al. Hybrid bagging and boosting with SHAP based feature selection for enhanced predictive modeling in intrusion detection systems. *Sci Rep*. 2024;14(1):30532. doi:10.1038/s41598-024-81151-1.
22. Zhang Y, Muniyandi RC, Qamar F. A review of deep learning applications in intrusion detection systems: overcoming challenges in spatiotemporal feature extraction and data imbalance. *Appl Sci*. 2025;15(3):1552. doi:10.3390/app15031552.
23. Waqas A, Khan SD, Ullah Z, Ullah M, Ullah H. Comparative analysis of deep learning models for intrusion detection in IoT networks. *Computers*. 2025;14(7):283. doi:10.3390/computers14070283.
24. Susilo B, Muis A, Sari RF. Intelligent intrusion detection system against various attacks based on a hybrid deep learning algorithm. *Sensors*. 2025;25(2):580. doi:10.3390/s25020580.
25. Qin Z, Luo Q, Nong X, Chen X, Zhang H, Wong CUI. MAS-LSTM: a multi-agent LSTM-based approach for scalable anomaly detection in IIoT networks. *Processes*. 2025;13(3):753. doi:10.3390/pr13030753.
26. Vieira MAM, Castanho MS, Pacifico RDG, Santos ERS, Câmara Júnior EPM, Vieira LFM. Fast packet processing with eBPF and XDP: concepts, code, challenges, and applications. *ACM Comput Surv*. 2020;53(1):16:1–36. doi:10.1145/3371038.
27. Høiland-Jørgensen T, Brouer JD, Borkmann D, Fastabend J, Herbert T, Ahern D, et al. The eXpress data path: fast programmable packet processing in the operating system kernel. In: *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*; 2018 Dec 4–7; Crete, Greece. p. 54–66. doi:10.1145/3281411.3281443.
28. Scholz D, Raumer D, Emmerich P, Kurtz A, Lesiak K, Carle G. Performance implications of packet filtering with Linux eBPF. In: *Proceedings of the 2018 30th International Teletraffic Congress (ITC 30)*; 2018 Sep 3–7; Vienna, Austria. p. 209–17. doi:10.1109/ITC30.2018.00039.
29. Gbadamosi B, Leonardi L, Pulls T, Høiland-Jørgensen T, Ferlin-Reiter S, Sorce S, et al. The eBPF runtime in the Linux Kernel. *arXiv:2410.00026*. 2024. doi:10.48550/arxiv.2410.00026.
30. Lundberg SM, Lee SI. A unified approach to interpreting model predictions. In: *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)*; 2017 Dec 4–9; Long Beach, CA, USA.
31. Bucur CE, Crihan G, Rădoi A, Robe-Voinea EG, Morosan IN. Enhancing network traffic monitoring through eXplainable artificial intelligence methodologies. *Telecom*. 2026;7(2):34. doi:10.3390/telecom7020034.

32. Sharafaldin I, Habibi Lashkari A, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP); 2018 Jan 22–24; Madeira, Portugal. p. 108–16.
33. Mondragon JC, Branco P, Jourdan GV, Gutierrez-Rodriguez AE, Biswal RR. Advanced IDS: a comparative study of datasets and machine learning algorithms for network flow-based intrusion detection systems. *Appl Intell.* 2025;55(7):608. doi:10.1007/s10489-025-06422-4.
34. Bucur CE. Hybrid-AI-IPS. Zenodo; 2026 [cited 2026 July 2]. Available from: <https://zenodo.org/doi/10.5281/zenodo.21138711>.
35. Ahmim A, Maglaras LA, Ferrag MA, Derdour M, Janicke H. A novel hierarchical intrusion detection system based on decision tree and rules-based models. In: Proceedings of the 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS); 2019 May 29–31; Santorini, Greece. p. 228–33. doi:10.1109/DCOSS.2019.00057.
36. Halbouni A, Gunawan TS, Habaebi MH, Halbouni M, Kartiwi M, Ahmad R. CNN-LSTM: hybrid deep neural network for network intrusion detection system. *IEEE Access.* 2022;10(11):99837–49. doi:10.1109/ACCESS.2022.3206425.
37. Hamidou ST, Mehdi A. Enhancing IDS performance through a comparative analysis of Random Forest, XGBoost, and Deep Neural Networks. *Mach Learn Appl.* 2025;22(12):100738. doi:10.1016/j.mlwa.2025.100738.
38. Bucur CE. Hybrid-AI-IPS. [cited 2026 June 14]. Available from: <https://github.com/Eugen548/Hybrid-AI-IPS>.