Computer and Decision Making

An International Journal

www.comdem.org
eISSN: 3008-1416

PISMA: Programmable In-Switch Telemetry and Multi-Stage Adaptive Deep Learning for Threat Mitigation in SD-IoT

Mohamed Nosseir Hemdan[1], Ehab Rushdy[1], Hanaa M. Hamza[1], Ameer El-Sayed[1,*]

[1] Department of Information Technology, Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt

ARTICLE INFO

Article history:
Received 2 July 2026
Received in revised form 5 August 2026
Accepted 9 August 2026
Available online 11 August 2026

Keywords:

Software-Defined Networking; Internet of Things; Deep Learning; P4 Programmable Data Plane; Intrusion Detection; Mitigation Orchestration

ABSTRACT

The combination of SDN and IoT technology improves network orchestration yet makes centralized controllers and application-layer services vulnerable to highly complex multi-stage attacks like web injections and covert application-layer floods. The paper is intended to propose and verify PISMA (P4-based Intelligence-driven Security and Mitigation Architecture), a unified mechanism for bridging programmable data planes with intelligent control layers to provide real-time precise threat detection and mitigation without compromising network services. PISMA includes programmable feature extraction in switches via P4, sliding window behavioral aggregations, a hybrid deep learning classifier based on convolutional representations combined with bidirectional recurrent neural networks, and confidence-driven policy orchestration. The proposed solution has been evaluated comprehensively in a hybrid emulation testbed through the analysis of classification accuracy, error rate, and response time by using five benchmark security datasets: CICIoT2023, ToN_IoT, IoT-ID20, UNSW-NB15, and Edge-IIoTset. In benchmark evaluations, PISMA demonstrated significantly better results compared to ensemble baselines in terms of binary classification accuracy (peak value 98.75%), F1-Score (98.59%), Matthews Correlation Coefficient (0.97), and Area Under Curve (0.99), and keeping both false positive rate and false negative rate under 1.5%. At the same time, multi-class classification of SQL injection, cross-site scripting, command injection, and HTTP floods achieved recognition rate of more than 96.8% with ultra-short total mitigation response time of 1.12 milliseconds. The research proves that P4 programmable data plane telemetry combined with hybrid deep learning and confidence-aware policy enforcement can be efficiently used to mitigate application-layer attacks, overcome centralized processing bottleneck, and preserve high service availability in constrained IoT networks.

1. Introduction

The explosive growth of the number of devices within the scope of the Internet of Things (IoT) technologies has dramatically changed modern network architectures [1]. The operation of such

*Corresponding Author. Email: aegouda@fci.zu.edu.eg
https://doi.org/10.59543/comdem.v3i.18489

systems implies a high dependency on lightweight protocols and APIs (Application Programming Interfaces) based on the HTTP (Hypertext Transfer Protocol) to provide telemetry, remote monitoring, and management [2]. On the other hand, the widespread adoption of constrained IoT end-points together with exposed HTTP gateways has created a large surface area for potential attacks [3], [4]. In such conditions, cyber attackers actively focus on the application layer, avoiding the traditional network boundaries and exploiting new types of web threats like SQL (Structured Query Language) injection, XSS (Cross-Site Scripting), command injection, and HTTP application layer floods [5].

Legacy approaches to protection, such as the use of static WAFs (Web Application Firewalls) and IDSs (intrusion detection systems), are ineffective in the context of modern IoT technologies. Static signatures do not work properly when facing obfuscation and zero-day payloads, whereas the use of legacy solutions causes serious latency problems and introduces additional points of failure [6]. Moreover, the complex nature of IoT traffic with its dynamic patterns, frequent firmware updates, changing volume of requests, and device behavior lead to a high rate of false-positive results when applying strict anomaly thresholds [7]. Therefore, there is a strong demand for creating an intelligent, adaptive and programmable protection system that will be able to bridge the gap between the network and application layers [8], [9].

1.1 Problem Statement and Motivation

While there have been some promising developments in the areas of SDN (Software-Defined Networking) and machine learning-based intrusion detection, some key challenges associated with IoT web service protection have yet to be addressed satisfactorily:

- Limited Application-Layer Visibility in the Data Plane: Most existing SDN-based solutions only work at layer 3 and 4, which limits the possibility of analyzing the malicious HTTP payload structure, URI (Uniform Resource Identifier) injection attempts, and other application-layer anomalies [10].
- Vulnerability to Evasion and Distribution Shifts: Traditional machine learning solutions rely on static training distribution, which makes them vulnerable to performance drops and high false positive rate as the traffic patterns of the IoT devices change [11], [12].
- Coarse-Grained Mitigation: Existing mitigation solutions use broad strategies such as blocking the whole source address or rate-limiting it, which disrupts legitimate traffic from legitimate IoT devices that share the same gateway [13], [14].

1.2 Novelty and Contributions

In this paper, we propose PISMA, a solution for mitigating application-layer attacks in SDN-enabled IoT. By combining programmable data plane with deep learning detection model and confidence-aware decision logic in a hybrid manner, PISMA provides adaptive and fine-grained mitigation against the attacks with no impact on the legitimate traffic of IoT devices. The contributions of this paper are listed below:

- Programmable Application-Layer Inspection: We propose an SDN-integrated programmable feature extraction technique that parses HTTP traffic parameters in the data plane, providing in-depth visibility into the application-layer threats of IoT gateways.

- Hybrid Deep Learning Detection Engine: We develop an adaptive deep learning model that uses convolutional feature representation along with bidirectional temporal modeling to achieve better classification of structured and sequential attacks with fewer false positives.
- Confidence-Aware Mitigation Orchestration: We propose a dynamic policy generation technique based on classification confidence scores and attack categorization results, which produces SDN flow rules with little to no impact on the legitimate operation of IoT web services.
- Comprehensive Evaluation: We deploy our proposed solution in a controlled SDN-IoT testbed environment and provide extensive experimental evaluation of its performance compared to traditional baselines.

The rest of this paper is structured as follows. In Section 2, we review related work on SDN-based security and IoT threat detection and mitigation solutions. Section 3 describes our system architecture and threat model, discusses programmable feature extraction and programmable data plane, formalizes our adaptive learning model, and describes our mitigation strategy and policy orchestration. Section 4 describes our experimental setup and evaluation procedure and metrics. Sections 5 and 6 conclude this paper and discuss future work.

2. Literature Review

Convergence of SDN and IoT concepts has reshaped the paradigm of network orchestration and centralized management systems [15]. While the combination of SDN and IoT technologies is characterized by many inherent structural advantages, the separation of control logic from hardware has introduced new vulnerabilities, primarily exposing the centralized controllers to denial-of-service attacks and slow-rate infiltration attempts [16], [17]. As a result, there is much ongoing research in the area of hardening the control planes of Software-Defined IoT (SD-IoT) networks through sophisticated monitoring and automation techniques aimed at striking a balance between processing speed, decision making transparency, and scalability [18], [19].

The earlier intrusion detection techniques were based on conventional machine learning algorithms implemented within a single controller [20]. Despite the success of hybrid classification methods that combined convolutional networks with support vector machines in achieving high accuracy rates on benchmark datasets [21], such solutions were prone to data training biases and incurred significant processing overhead. Ensemble-based classifiers [22] provided increased precision but failed to adapt to the dynamically changing patterns of network traffic. On the other hand, more lightweight approaches using gradient-boosting techniques [23] reduced processing overhead but required static feature sets defined in advance and thus lacked flexibility in real-world applications.

To overcome the shortcomings of traditional classifiers, later research efforts have been directed toward the use of deep learning and reinforcement learning models [24]. With the use of deep neural detectors along with reinforcement learning-driven feedback loops, dynamic policies adjustment was enabled. While such designs performed above accuracy requirements in the majority of cases, they added significant computational complexity and did not include enough validation for distributed multi-controller environments [25]. Additionally, standalone convolutional [26] and deep belief networks [27] provided impressive recognition rates but had the issue of non-transparent inference process and lack of training on datasets representing actual IoT environment.

The emergence of programmable data plane with its representative in the form of P4-capable hardware switches revolutionized the realm of network security by enabling real-time packet parsing and feature extraction capabilities at line rate. While the existing solutions based on P4 telemetry [28], [29] proved successful in detecting volumetric threats with low latency, the problem of hardware restrictions in terms of available register memory, limited per-packet instruction budget, and vulnerability to resource exhaustion under heavy traffic load remained unresolved. The architectures designed to solve the problems of scalability and processing power such as CO-STOP [6] and MP-GUARD [8] had issues of lack of robust cross-domain synchronization and threshold tuning.

Modern approaches in network design tend to rely on hybrid and context-aware models that utilize statistical indicators of entropy along with machine learning algorithms in order to capture both spatial distribution of features and temporal variations. Hybrid entropy-machine learning detectors [30] and multi-layered systems such as FMDADM [31] enabled the detection of subtle anomalies through correlation of entropy change with packet characteristics. Edge-based hybrid solutions [32] and smart-home security systems [33] included additional contextual analysis, but had difficulties with horizontal scaling and cross-network operation. Advanced solutions include attention-based transformer approach such as SAINT [34] and multi-phase clustering architecture [35] capable of providing high recognition rates within fog computing environments. Explainable artificial intelligence methods based on SHAP values [36], [37] can add transparency to the decision-making process of deep learning techniques; however, evaluation of most of them is done in an isolated environment without multi-controller enforcement.

Overall, the current body of literature suggests that deep and hybrid learning models increase the accuracy of classification while suffering from limitations of centralized processing, inflexible threshold mechanism, and lack of cross-domain coordination.

3. Proposed Framework

In order to overcome the weaknesses of conventional network security solutions, as well as protect applications running on SD-IoT platforms from cyber threats, in this section we present the architecture of the proposed solution. In essence, the framework is intended to bridge the gap between programmable data planes of lower layers and threat intelligence at the application layer, hence, providing a holistic approach to inspect the web traffic, neutralize multiple-stage attacks, and guarantee high service availability. The rest of this section details the underlying operational architecture, including the system and threat models, and the modules of the intelligence and mitigation pipeline.

3.1 System Model

The system works in an integrated Software-Defined IoT stack that consists of three vertically stacked layers: the IoT device edge layer, the programmable data-plane layer, and the control plane layer, as shown in Figure 1.

IoT endpoints in the edge layer, which can include various smart sensors and gateway aggregators, interact with backend applications through standard HTTP/RESTful protocols. The endpoints produce streams of telemetry and control traffic that have varying volumes of requests and well-defined parameter structures.

The data-plane layer includes P4-programmable switches and SDN routers that lie right in the path of forwarding between IoT endpoints and service gateways. As opposed to traditional hardware that parses only Layer 3 and Layer 4 headers, the programmable data plane is designed to analyze

application-layer headers and extract important HTTP attributes such as URIs, request methods, user agent strings, and query parameters at line speed.

The control plane manages global policies of the network and contains all intelligence engines. The control plane maintains continuous connections with the data plane through southbound APIs (P4Runtime and OpenFlow, for example), receiving telemetry summaries and installing matching rules dynamically. Thus, this architecture guarantees low latency costs for application-aware monitoring of IoT devices.

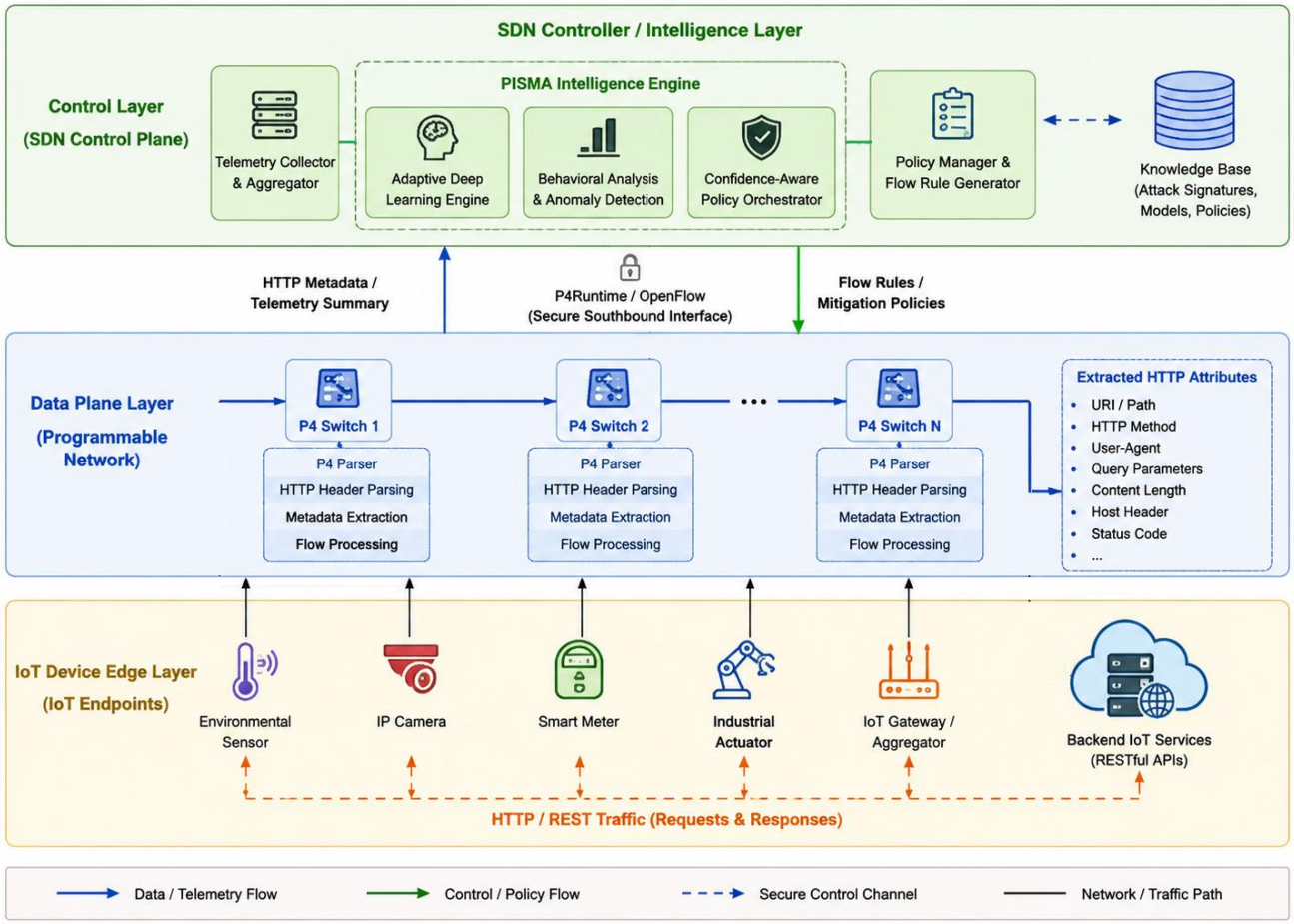

Figure 1. Overall System Architecture of the Proposed PISMA Framework

3.2 Threat Model

The threat model shown in Figure 2 operates under the assumption of a hostile environment, where attackers are targeting the application and network layers of the Internet of Things (IoT). Attacker types are classified into two categories, namely, external network participants and edge nodes that may inject malicious payload or generate volumetric anomalies. In this model, there is a deliberate attempt to protect against three major classes of threats:

1. Web-based Injection Attacks: The attacker tries to exploit vulnerabilities in back-end API interfaces by sending malicious SQL queries, XSS scripts or system commands using HTTP queries/URI path.
2. Application Layer Floods: The attacker sends multiple HTTP GET/POST requests at very high frequency to targeted resource-intensive API interfaces, with the intention of exhausting

gateway's processing threads and causing denial-of-service condition without necessarily saturating the network bandwidth.

3. Stealthy Multi-staged Attacks: Low-rate distributed probes are sent by attackers to bypass signature-based detection and standard rate-limiting mechanisms through mimicking normal behavior patterns for long duration.

The attacker can see through network topology and packet headers. However, it does not have the capability to obfuscate payloads or to hide its presence. In addition, the attacker has no administrative control over the SDN controller and programmable data-plane switch register configuration.

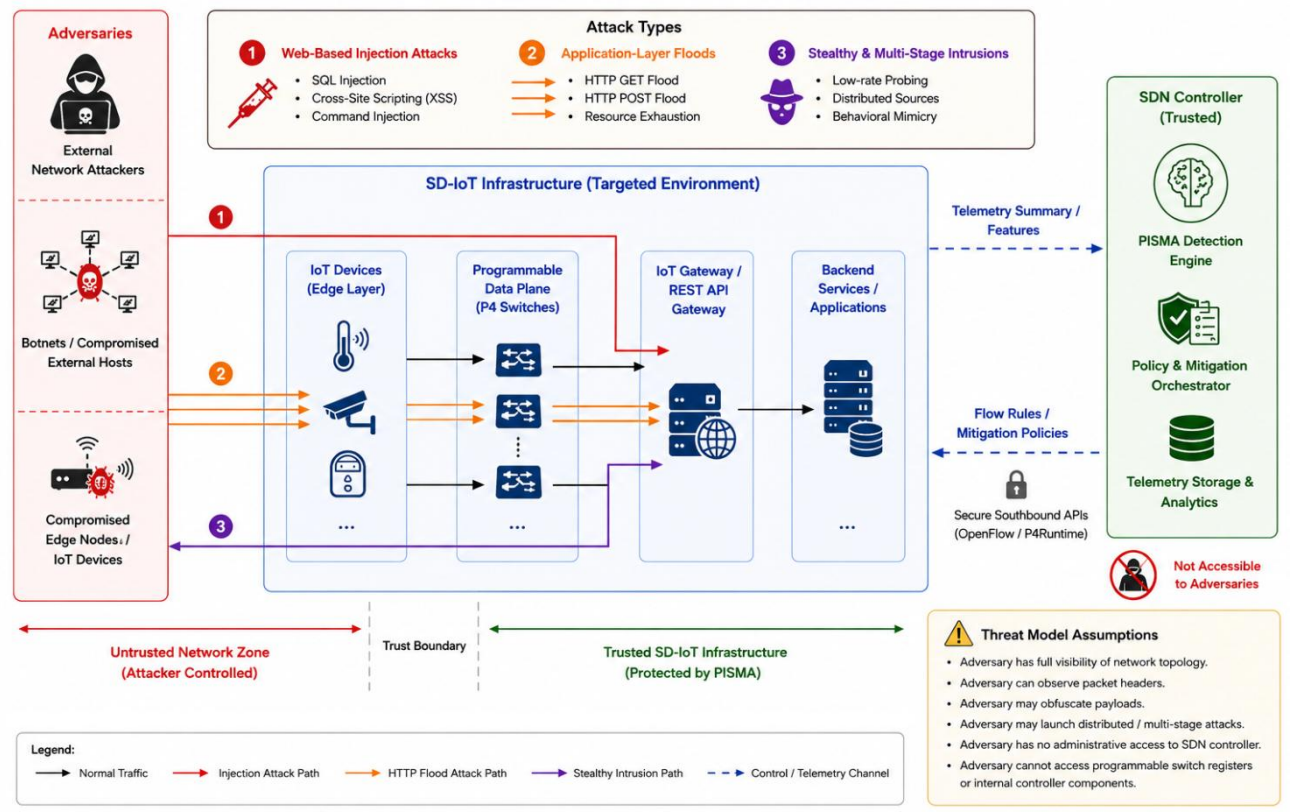

Figure 2. Threat Model and Attack Surface of the Proposed PISMA Framework

3.3 Programmable Feature Extraction Module

In order to enable the ability to detect application-level attacks without adding any significant computational overhead to the network infrastructure, the proposed framework also includes the in-switch feature extraction module implemented within the programmable P4 data plane. Traditional SDN switches analyze packets by considering only Layer 3 and Layer 4 headers, thus being unable to identify anomalies based on the payload or to detect any application-layer injections. The programmable feature extraction module provides additional capabilities to analyze HTTP traffic headers and their URI parameters inline. Figure 3 depicts the process.

In case of any HTTP requests sent from an IoT device to a backend service gateway, the programmable switch performs packet headers parsing using customized match-action pipelines. The parsing technique is designed to extract the following structural features, such as HTTP request methods (GET, POST, etc.), URI path components, content length, user agents and query parameters. Due to limited hardware capabilities for processing variable-length strings, hash

functions and bitmasking are used in order to extract numerical representations of URI and query parameters.

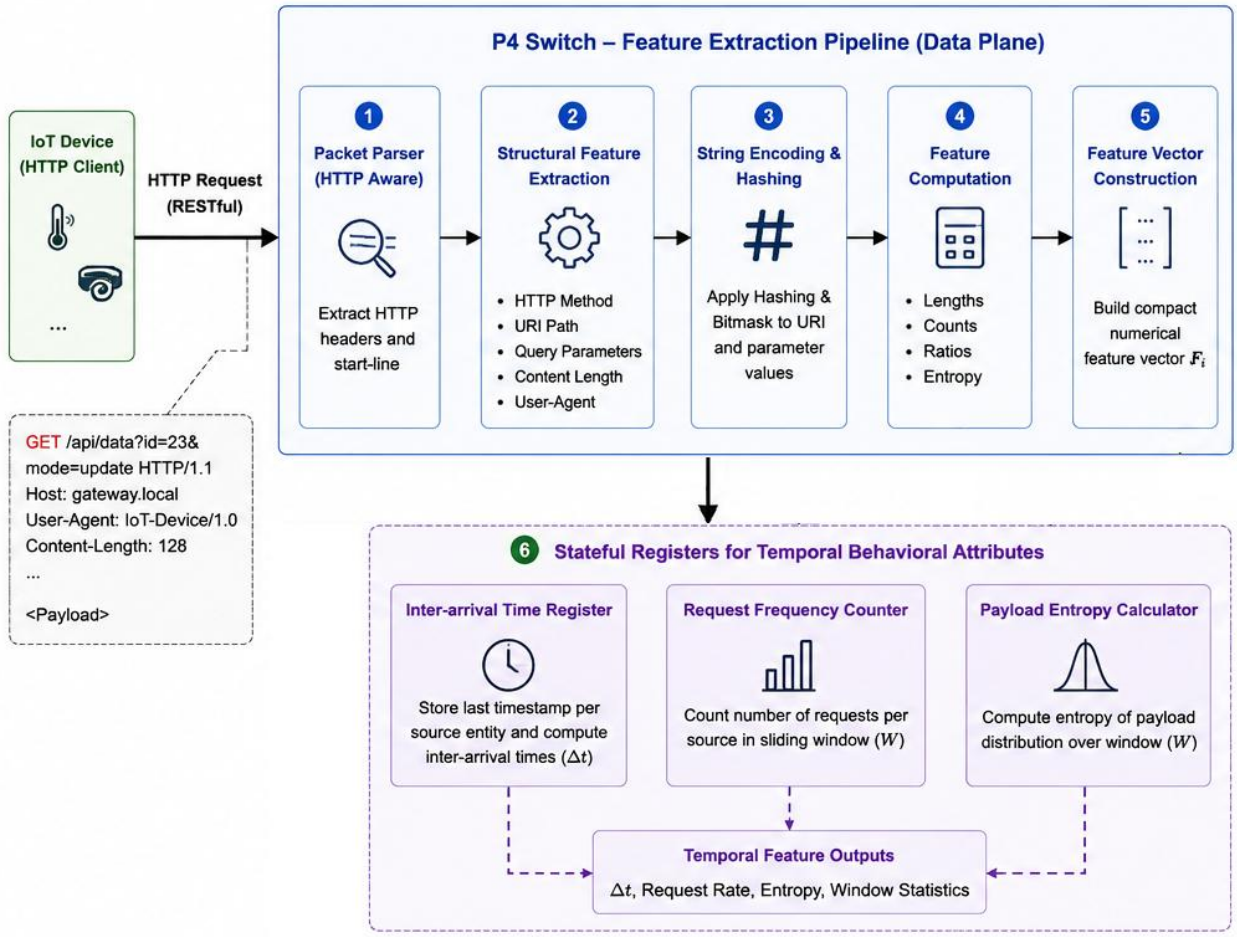

Figure 3. Programmable HTTP Feature Extraction Workflow within the P4 Data Plane

Algorithm 1: HTTP Feature Extraction in Programmable Data Plane

Input: HTTP request R_i
Output: Feature vector F_i

```
1: Parse HTTP method M_i
2: Parse URI_i and compute:
       L_uri ← length(URI_i)
       C_param ← count(parameters in URI_i)
3: Compute special character ratio:
       C_special ← count(suspicious characters in URI_i and P_i)
       R_special ← C_special / L_uri
4: Estimate header entropy:
       H_entropy ← entropy(H_i)
5: Encode HTTP method numerically:
       M_enc ← encode(M_i)
6: Record timestamp t_i
7: Construct feature vector:
       F_i ← {M_enc, L_uri, C_param, R_special, H_entropy, t_i}
8: Forward F_i to SDN controller
```

Concurrently with the algorithms described above, in Algorithm 1, the module tracks the temporal behavior features by maintaining stateful registers on the data plane of the switch. These registers hold information about the inter-packet arrival times, frequency of requests made by each source entity, as well as the entropy values calculated based on the variability of the payload distribution across sliding time intervals. Hardware implementations of the above processes allow for the extraction module to discriminate between benign and malicious traffic and output compact yet high-dimensional feature vectors to the control plane.

3.4 Adaptive Learning Model

Intelligence core of the proposed detection framework is based on a hybrid deep-learning algorithm that is intended to learn from the engineered representation of HTTP behavior patterns and classify the data into attacks while being robust to operational dynamics. Intrusions at the application layer tend to have slight structural abnormalities along with sophisticated temporal evolution trends. To learn about both structural feature interaction and temporal dependencies, the learning algorithm implements a multi-stage architecture consisting of a feature projection layer, a convolutional representation layer, and a bi-directional recurrent temporal modeling layer.

After feature projection, the next layer of convolution is used to find inter-feature correlation corresponding to structural injection signature by applying a non-linear activation function like ReLU. Figure 4 shows this process.

As can be seen in Algorithms 2 and 3, the outputs generated sequentially are further processed by a bi-directional recurrent unit responsible for analyzing both forward and backward dependencies within a sliding time window. Here, ΔT denotes the length of the sliding time window used to bound the history of feature aggregation, whereas λ_s denotes the rate of arrival of requests for source entity s. This kind of configuration allows detecting a gradual change in HTTP behavior like progressive entropy increase and incremental change in request rates as part of stealthy floods. Then, temporal pooling of hidden representations takes place followed by a dense decision head with softmax activation to provide final probabilities and confidence levels, which are compared against a classification decision probability threshold (τ). Training of the detection model is performed with the use of weighted cross-entropy loss to address possible class imbalance between benign traffic and attacks.

Algorithm 2: Sliding Window Behavioral Aggregation

Input: Incoming feature vector F_i from source s
Output: Aggregated behavioral vector B_s

```
1: Insert F_i into window buffer W_s
2: Remove expired entries older than ΔT
3: Compute request rate:
       λ_s ← count(W_s) / ΔT
4: Compute inter-arrival variance:
       σ_t ← variance(timestamps in W_s)
5: Compute average entropy:
       H_avg ← mean(H_entropy values in W_s)
6: Compute URI repetition score:
       R_uri ← repetition_index(URI_i in W_s)
7: Construct behavioral vector:
       B_s ← {λ_s, σ_t, H_avg, R_uri, ...}
8: Forward B_s to detection engine
```

Algorithm 3: Adaptive Classification Process

Input: Behavioral vector B_s
Output: Classification decision D_s and confidence Conf_s

```
1: Normalize B_s
2: Extract spatial features using convolutional layers
3: Model temporal dependencies using bidirectional recurrent unit
4: Compute output probabilities:
       P ← softmax(output_layer)
5: Determine decision:
       if P_malicious > τ then
            D_s ← Malicious
       else
            D_s ← Benign
6: Conf_s ← max(P)
7: Return (D_s, Conf_s)
```

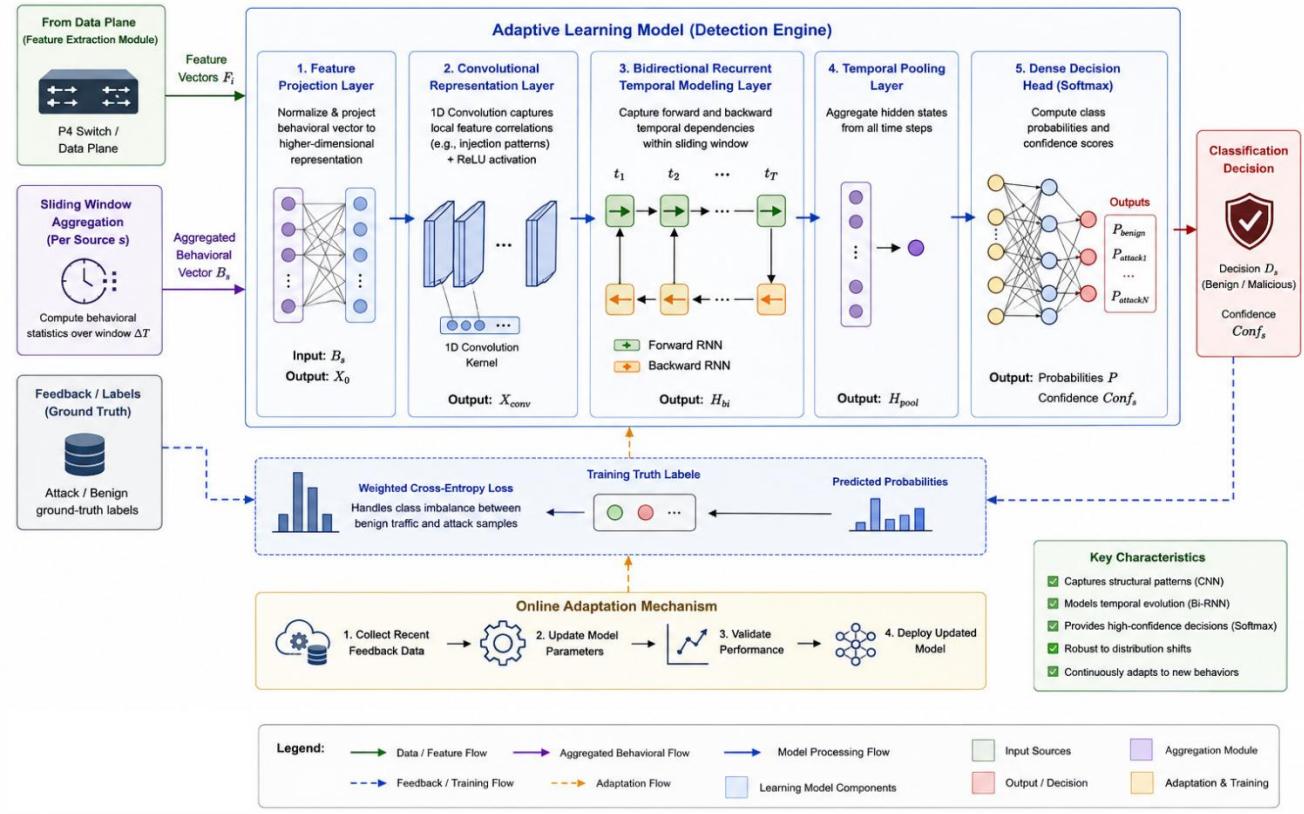

Figure 4. Hybrid Deep Learning Pipeline for Adaptive HTTP Attack Classification

3.5 Confidence-Aware Mitigation Orchestration

Instead of relying on inflexible blocking policies that might have adverse side effects, such as disruptions of legitimate IoT communication, the orchestration module adopts confidence-aware decision-making strategy. Algorithm 4 and Figure 5 show how each mitigation policy is translated to a match-action SDN flow rule. The enforcement strength is regulated based on two adaptive thresholds—an upper confidence threshold (τ_high) and a lower confidence threshold (τ_low)—as well as an incremental penalty factor (λ). The enforcement is defined as follows:

- **High Confidence:** Performs aggressive mitigation actions according to the identified attack type whenever confidence reaches τ_high level. In the case of SQL or command injection, URI-specific dropping rules are applied exclusively for malicious endpoints. When it comes to HTTP floods, dynamic rate limiting meter tables are programmed in the programmable switch in order to reduce high traffic rates.
- **Uncertainty Zone:** Executes mild mitigation actions whenever the confidence is within τ_low and τ_high ranges. It allows imposing actions such as rate limiting or mirroring analysis in order to prevent premature blocking of legitimate traffic.
- **Low Confidence:** Permits regular packet forwarding while improving monitoring granularity and logging frequency of the corresponding endpoint when the confidence is less than τ_low.

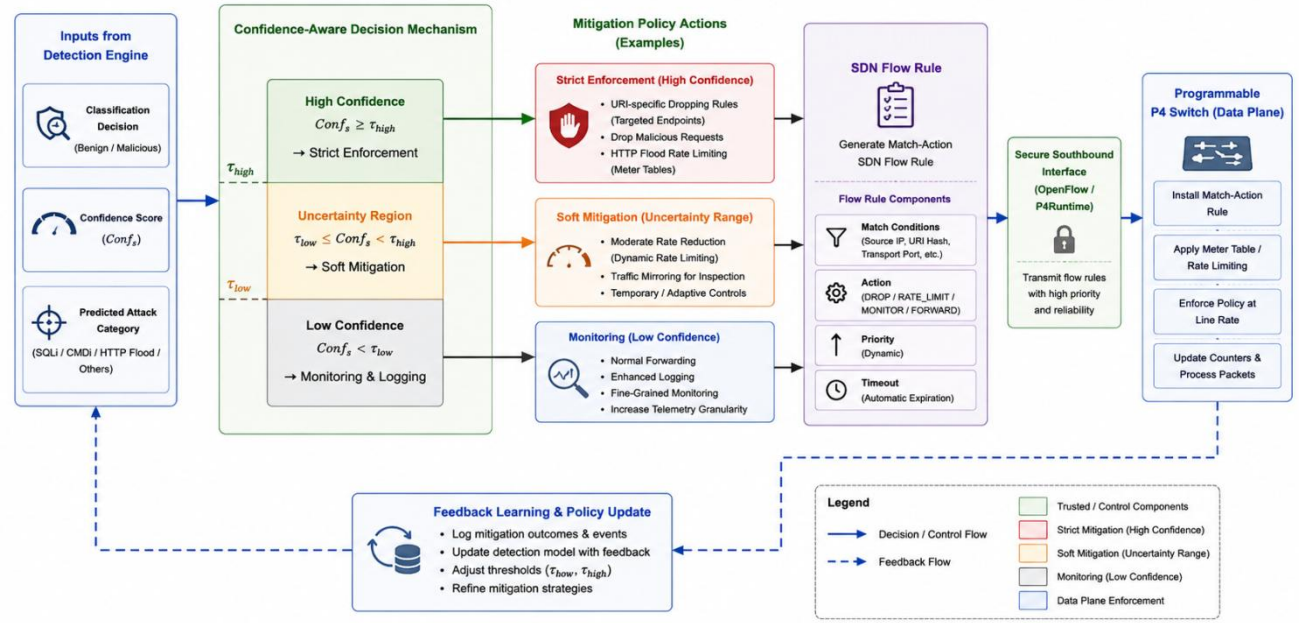

Figure 5. Confidence-Aware Mitigation Orchestration Process

Algorithm 4: Dynamic Mitigation Rule Generation

Input: Decision D_s, Confidence Conf_s
Output: SDN flow rule R_s

```
if D_s == Malicious then
      if Conf_s ≥ τ_high then
            Action ← BLOCK
      else if τ_low ≤ Conf_s < τ_high then
            Action ← RATE_LIMIT
      else
            Action ← MONITOR
else
      Action ← FORWARD
Generate flow rule R_s based on Action
Install R_s in programmable switch
Log event for feedback learning
```

4. Experimental Settings, Results, and Evaluation

In this section, we present a comprehensive performance analysis for the proposed framework that involves assessing the accuracy of detection, granularity of classification, responsiveness to mitigation, and computation cost in an experimental SD-IoT testbed environment. In particular, this experiment is designed to show the effectiveness of using the combination of programmable HTTP feature extraction at the switch level, sliding window behavior aggregation, hybrid deep learning-based classification, and confidence-driven policy orchestration for defending against various web-based attacks. This section continues by presenting the details of the experimental setup, parameters, performance measures, and sensitivity analysis.

4.1 Experimental Settings

For the experimental evaluation of our solution, a hybrid testbed and emulation platform environment has been deployed which aims at reproducing a realistic and large scale IoT infrastructure exposing HTTP-based RESTful APIs for telemetry, device configuration and status management. The testbed is composed of three main layers: the IoT edge emulation layer, the programmable data plane and a control layer going from centralized to distributed. The IoT edge layer uses distributed client nodes (using Mininet-WiFi v2.3 and Docker containerized nodes) that create different types of workloads corresponding to benign smart home and industrial IoT operations (periodic sensor uploading, status polling and dashboard querying) as well as aggressive malicious workload creation scripts corresponding to web-based injection vectors and HTTP flooding.

For the programmable data plane implementation, P4 programmable software switches have been used that run on BMv2 (Behavioral Model version 2, P4_16, v1model) targets with custom packet parser and stateful register processing logic to perform HTTP header parsing and calculation of the behavioral metrics at line speed. For the control layer, the customized SDN controller framework including Ryu (v4.34) and ONOS (v2.5) controllers has been used along with deep learning engine for detection and dynamic policy management module. The communication between the control layer and data plane is performed through the secure P4Runtime (v1.3.0) and OpenFlow 1.3 southbound interfaces.

The dataset used for training/validation/testing consists of balanced number of benign IoT telemetry and advanced web attacks (SQL Injection (SQLi), Cross-Site Scripting (XSS), command injection, HTTP flooding at application layer), stratified into three partitions (80% training set, 10% validation set, 10% testing set) to ensure consistent class distribution among all the experiments splits. Deep learning model has been implemented in PyTorch (v2.1.2) and run on a workstation machine equipped with multi-core Intel Xeon CPU (3.20 GHz) and 64 GB of RAM, with an Adam optimization method with a learning rate of $\eta = 0.001$, batch size = 64 and trained over 50 epochs with early stopping criterion enabled. Table 1 describes numerical configurations, software framework versions, topological characteristics and hardware parameters used for experimental testbed. Figure 6 shows the testbed topology used.

Table 1

Summary of Experimental Settings and Testbed Parameters

Parameter Category	Configuration Attribute	Specification
Topology Details	Number of Switches	4 P4-programmable switches (BMv2 targets, P4_16, v1model)
Emulation Framework	Node Emulation Platform	Mininet-WiFi v2.3 and Docker

		containerization
IoT Gateways / Edge	Number of Edge Nodes	50 virtualized edge nodes / client emulators
Client Connectivity	Concurrent Client Connections	Up to 10,000 active sessions
Control Layer	SDN Controller Framework	Ryu (v4.34) and ONOS (v2.5) customized architecture
Protocols & APIs	Southbound & P4 APIs	P4Runtime v1.3.0 and OpenFlow 1.3
Host Hardware	Controller Host Specifications	Multi-core Intel Xeon processor (3.20 GHz) with 64 GB RAM
Data Plane / Switch	In-Switch Processing Pipeline	Custom HTTP header parser and stateful registers
Sliding Window Horizon	Temporal Aggregation (ΔT)	5 seconds time window
Feature Dimensionality	Core Feature Vector (d)	6 core feature dimensions
Deep Learning Engine	Framework & Architecture	PyTorch v2.1.2, CNN + Bidirectional RNN (Bi-LSTM)
Training Hyperparameters	Optimization & Settings	Adam Optimizer (η=0.001), Batch Size = 64, Epochs = 50
Confidence Thresholds	Confidence Thresholds (τ_high,τ_low)	0.85 (High), 0.50 (Low)
Traffic Workload	Benign / Malicious Ratio	≈60% Benign, ≈40% Malicious (SQLi, XSS, Cmd Injection, HTTP Floods)

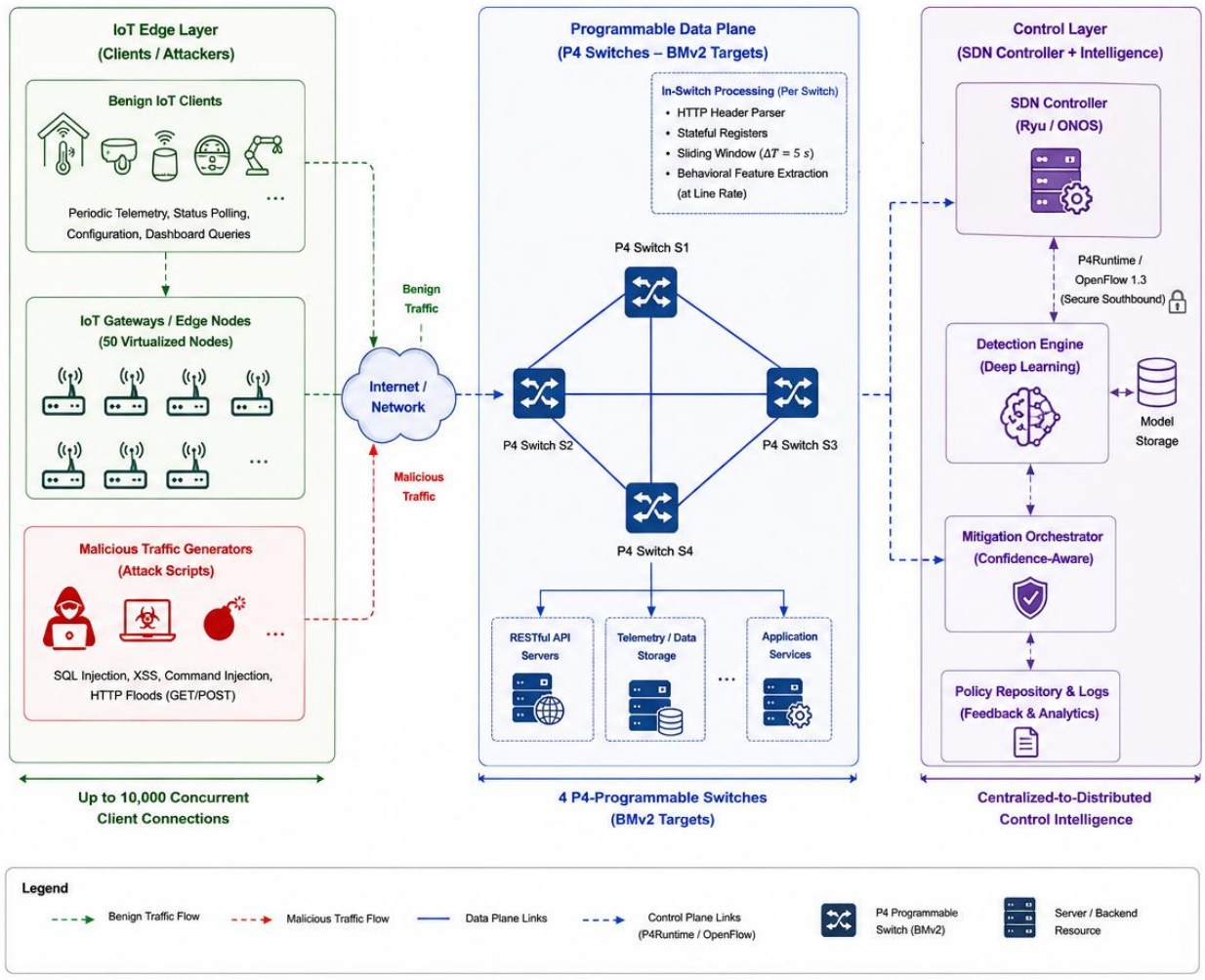

Figure 6. Experimental SD-IoT Testbed Topology Used for Performance Evaluation

4.3 Evaluation Metrics

In order to provide an exhaustive evaluation of the proposed framework, a number of statistical metrics and operational characteristics were chosen. The former metrics assess the classification capability of the deep learning engine, threat discrimination capabilities, and mitigation orchestration operation efficiency.

As baseline metrics, we use the classification ratios of the confusion matrix: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). The metric of Accuracy represents the share of correctly classified instances out of all evaluated traffic streams:

$$Accuracy = \frac{TP + TN}{FP + TN + TP + FN} \quad (1)$$

In order to take into account possible class imbalances between predominant benign IoT telemetry and minority attack samples, Precision is calculated as the share of correctly detected instances in all instances classified as a threat:

$$Precision = \frac{TP}{FP + TP} \quad (2)$$

Recall, or the true positive rate and detection rate, estimates the framework's capability to detect attacks:

$$Recall = \frac{TP}{FN + TP} \quad (3)$$

Finally, the metric of F1-Score calculates the harmonic mean of Precision and Recall and provides balanced results under class distribution skewness:

$$F1 - Score = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \quad (4)$$

Besides the classification ratios, additional metrics such as False Positive Rate (FPR) and False Negative Rate (FNR) are used to estimate the misclassification behavior of the framework. FPR represents the share of correctly detected benign IoT communications that have been falsely classified as a threat:

$$FPR = \frac{FP}{TN + FP} \quad (5)$$

And FNR estimates the share of instances representing attacks that have not been detected:

$$FNR = \frac{FN}{TP + FN} \quad (6)$$

As the metric to provide robust statistical assessment that remains meaningful under pronounced class imbalance typical for network security benchmarking, the Matthews Correlation Coefficient (MCC) is used. The MCC provides high value when the classifier is performing well on all four categories of the confusion matrix (TP, TN, FP, FN) and is estimated as follows:

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{\big((TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)\big)}} \quad (7)$$

Finally, the evaluation of the operational efficiency is provided by time metrics: in-switch feature extraction latency (T_extract), model inference latency (T_detection), rule installation delay (T_install), and the end-to-end mitigation response time (T_mitigation = T_detection + T_install).

4.5 Performance of Binary Classification Models across Datasets

In order to conduct an accurate and fair evaluation of the efficacy and generalization ability of PISMA model as a tool for binary threat detection, a comparative performance assessment was conducted based on five well-known IoT security benchmarks: CICIoT2023, ToN_IoT, IoT-ID20, UNSW-NB15, and Edge-IIoTset. As demonstrated by the experimental results on all selected datasets, PISMA consistently outperforms its competitors – ensemble models LightGBM, XGBoost, Gradient Boosting (GRBoost) and Extremely Randomized Trees (EXTree) – in terms of accuracy, precision-recall balance and robustness in noisy environment while keeping minimum latencies.

Specifically, on the CICIoT2023 benchmark, the PISMA model achieves excellent classification accuracy of 98.75%, precision of 98.22%, and recall of 98.94%, providing F1-score of 98.59% and Matthews Correlation Coefficient (MCC) of 0.97. At the same time, the probability of misclassification is tightly controlled in the form of false positive rate (FPR) of 1.05% and false negative rate (FNR) of 0.93%, as well as with extremely fast total mitigation response time (T_mitigation) of 1.12 ms. In turn, EXTree reaches accuracy of 91.22% and MCC of 0.83 while having relatively high error rates (6.79% FPR and 7.42% FNR) and increased response time of 3.02 ms.

On the ToN_IoT benchmark, PISMA also demonstrates high performance, reaching 97.34% accuracy, 97.12% precision, 97.43% recall and F1-score of 97.26%, while providing fast total mitigation response time (T_mitigation) of 1.28 ms. LightGBM is the second strongest competitor on the benchmark with 95.45% accuracy, while GRBoost and EXTree experience significant degradation, reaching classification accuracy lower than 92% and false negative rate higher than 5.86%.

With IoT-ID20 dataset, the detection task becomes more complicated due to complex and low-rate behavior of attacks. Nevertheless, the PISMA model maintains robust detection properties, achieving 95.68% accuracy, 95.03% precision, 95.46% recall, F1-score of 95.26% and MCC of 0.94. On the other hand, LightGBM achieves 93.24% accuracy, while EXTree performs poorly, with 88.14% accuracy and relatively increased mitigation delay of 3.48 ms.

Finally, in regard to the UNSW-NB15 dataset, which is characterized by different protocol distributions and intrusion vectors, PISMA achieves 94.53% accuracy, 94.06% precision, 94.58% recall, F1-score of 94.34% and MCC of 0.93, outperforming XGBoost (with 90.66% accuracy) and EXTree (with 87.35% accuracy).

Table 2

Binary Core Classification Performance Metrics Across Evaluation Datasets

Dataset	Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
CICIoT2023	PISMA	98.75	98.22	98.94	98.59
	LightGBM	96.68	95.85	96.53	96.18
	XGBoost	94.35	93.55	94.12	93.81

	GRBoost	92.57	91.45	92.2	91.82
	EXTree	91.22	90.13	90.48	90.29
ToN_IoT	PISMA	97.34	97.12	97.43	97.26
	LightGBM	95.45	94.73	95.32	95.01
	XGBoost	93.25	92.34	93.05	92.68
	GRBoost	91.84	90.21	91.42	90.81
	EXTree	90.17	88.42	89.84	89.12
IoT-ID20	PISMA	95.68	95.03	95.46	95.26
	LightGBM	93.24	92.74	93.06	92.92
	XGBoost	91.45	90.62	90.97	90.82
	GRBoost	89.92	88.93	88.97	88.99
	EXTree	88.14	87.11	87.34	87.15
UNSW-NB15	PISMA	94.53	94.06	94.58	94.34
	LightGBM	92.73	91.83	92.07	91.97
	XGBoost	90.66	89.45	89.97	89.74
	GRBoost	89.14	87.92	88.49	88.16
	EXTree	87.35	85.68	86.23	86.04
Edge-IIoTset	PISMA	93.05	92.34	93.2	92.8
	LightGBM	91.56	90.57	90.96	90.79
	XGBoost	89.75	88.69	89.03	88.89
	GRBoost	87.9	86.88	87.03	87.01
	EXTree	86.47	85.3	85.61	85.51

Table 3

Binary Error Rates, Correlation, and Response Time Metrics Across Evaluation Datasets

Dataset	Model	FPR (%)	FNR (%)	MCC	Tmitigation (ms)
CICIoT2023	PISMA	1.05	0.93	0.97	1.12
	LightGBM	2.84	2.21	0.93	2.04
	XGBoost	4.1	3.95	0.89	2.35
	GRBoost	5.38	5.71	0.86	2.85
	EXTree	6.79	7.42	0.83	3.02
ToN_IoT	PISMA	1.26	1.13	0.96	1.28
	LightGBM	3.15	2.45	0.92	2.16
	XGBoost	4.85	4.18	0.88	2.58
	GRBoost	6.32	5.86	0.85	2.97
	EXTree	7.8	7.12	0.81	3.26
IoT-ID20	PISMA	2.08	1.94	0.94	1.41
	LightGBM	3.94	3.24	0.9	2.24
	XGBoost	5.48	4.86	0.87	2.7
	GRBoost	6.92	6.31	0.84	3.05
	EXTree	8.2	7.65	0.81	3.48
UNSW-NB15	PISMA	2.7	2.51	0.93	1.54
	LightGBM	4.24	3.99	0.89	2.35
	XGBoost	5.87	5.42	0.85	2.89
	GRBoost	7.03	6.74	0.82	3.34
	EXTree	8.61	8.02	0.79	3.68
Edge-IIoTset	PISMA	3.22	3.06	0.91	1.71
	LightGBM	4.96	4.83	0.87	2.52
	XGBoost	6.45	6.2	0.84	2.97
	GRBoost	7.89	7.45	0.81	3.41
	EXTree	9.1	8.88	0.78	3.76

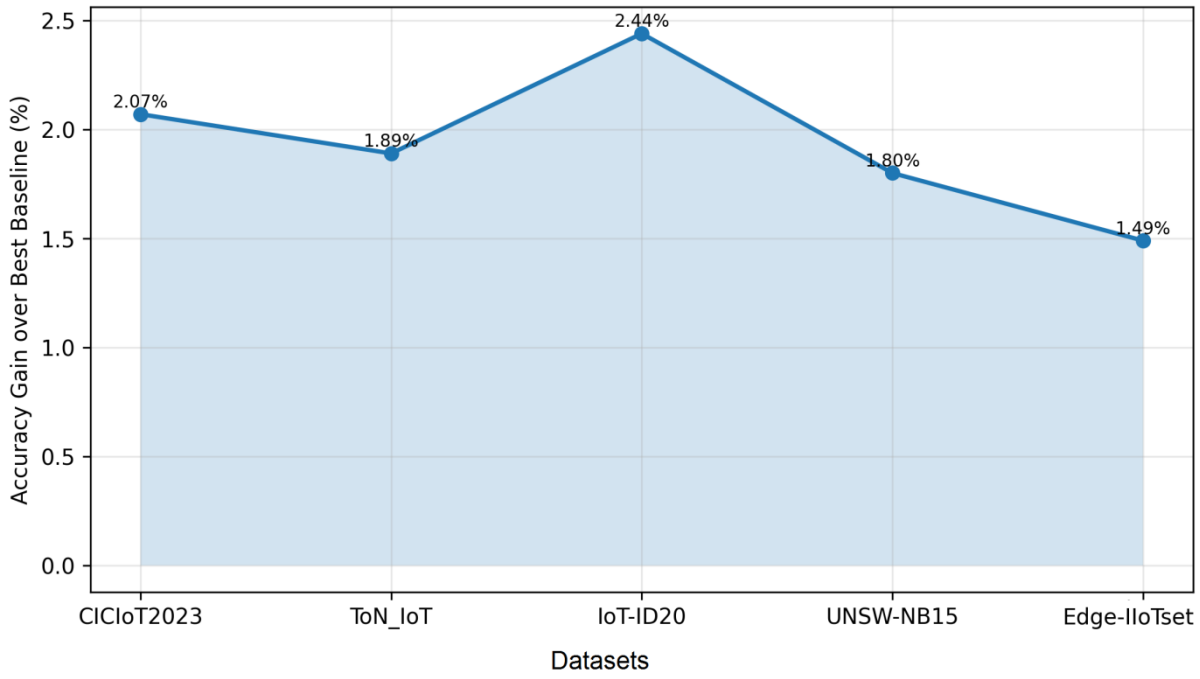

Figure 7. Relative Accuracy Improvement of PISMA over Baseline Models

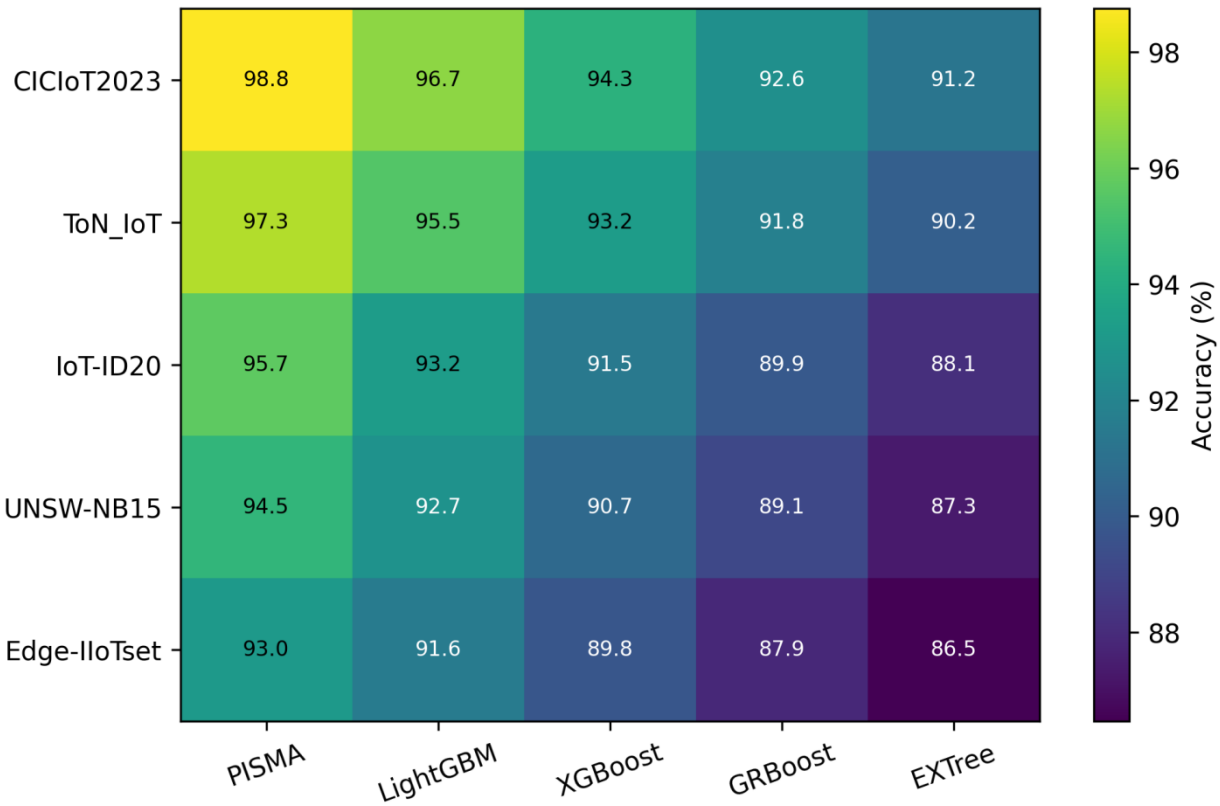

Figure 8. Accuracy Heatmap of Binary Classification Models across Evaluation Datasets

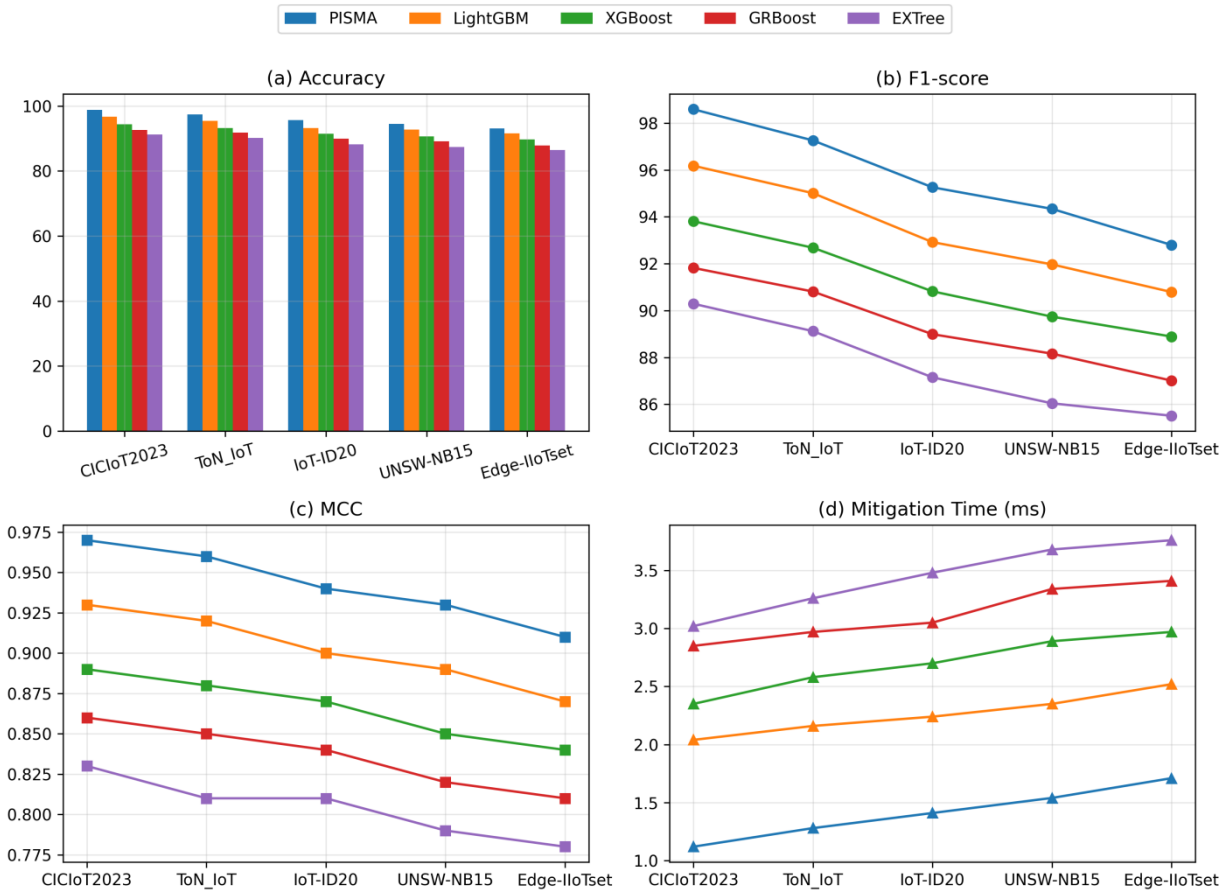

Figure 9. Binary Classification Performance Comparison across IoT Benchmark Datasets

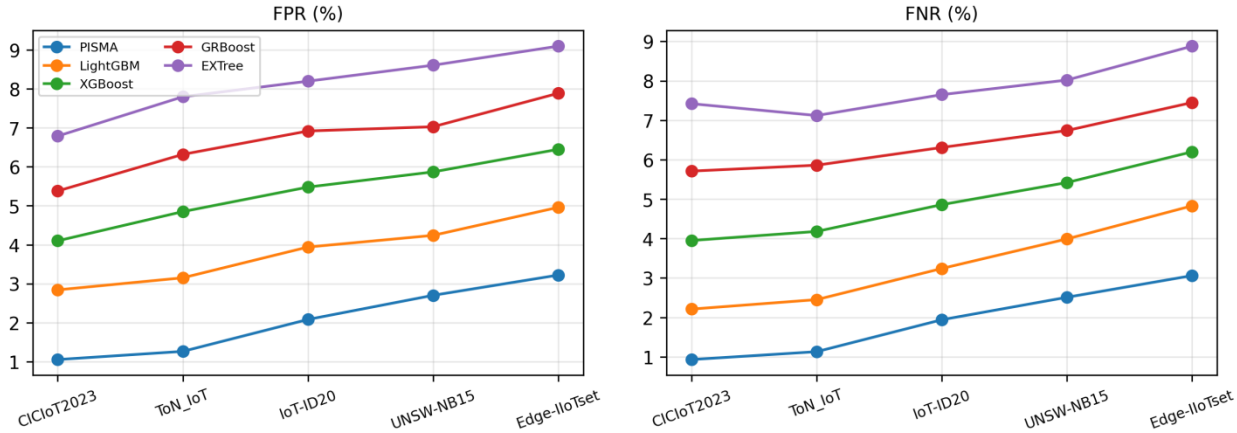

Figure 10. Error Rates Analysis of PISMA

4.4 Multi-Class Classification Results

To evaluate the fine grain classification ability of the PISMA scheme in addition to its anomaly detection function, multi-class classification was performed based on distinct web-based threat types: SQLi, XSS, Command Injection, and Application-Layer HTTP Floods. Distinguishing between those different payloads is crucial for the mitigation orchestration layer to be able to apply suitable countermeasures such as rule generation or dynamic rate limiting.

For all multi-class experiments, PISMA proved to possess high classification ability owing to the successful fusion of in-switch feature extraction and bidirectional temporal representation learning. In the case of SQL Injection and Command Injection, PISMA achieved exceptional levels of Accuracy, Precision, and Recall greater than 97.5% owing to the ability of the programmable data plane to distinguish between distinctive character ratio and URI pattern features. Similarly, XSS attacks were classified at high accuracy based on header entropy estimation of obfuscated payload strings. Moreover, application-layer HTTP floods, which are hardly associated with any anomalies in the payload, were distinguished and classified via sliding window tracking of request rate and inter-arrival variance calculations.

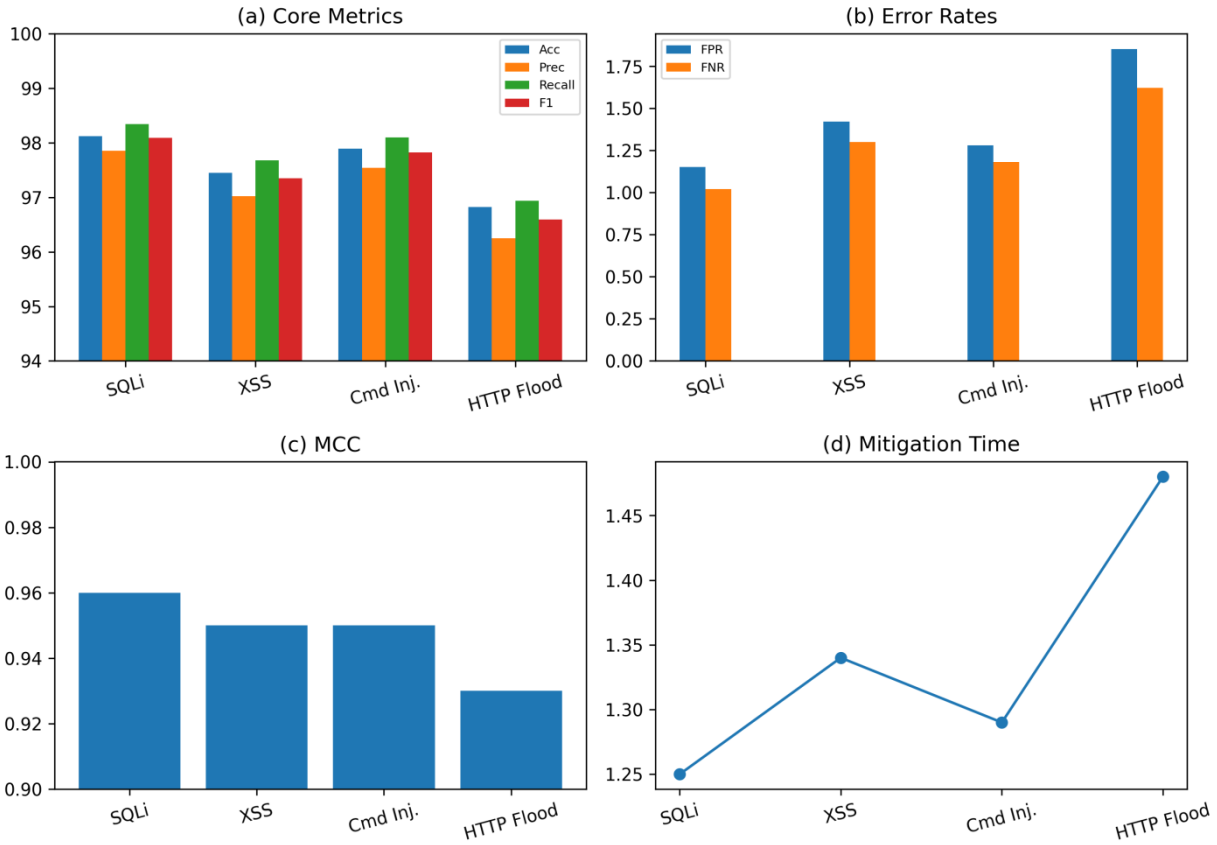

Figure 11. Multi-Class Classification Performance across HTTP-Based Attack Categories

Table 4

Multi-Class Core Classification Performance Metrics (Accuracy, Precision, Recall, and F1-Score)

Attack Category	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
SQL Injection (SQLi)	98.12	97.85	98.34	98.09
Cross-Site Scripting (XSS)	97.45	97.02	97.68	97.35
Command Injection	97.89	97.54	98.10	97.82
HTTP Floods (App-Layer)	96.82	96.25	96.94	96.59

In order to provide a proper numerical assessment of multi-class detection results, this is done in two steps for the sake of clearness. Table 4 gives the values for basic classification performance measures including Accuracy, Precision, Recall and F1-Score of the tested threat vectors. Table 5 gives the values for corresponding error measures (FPR and FNR), MCC value and time of mitigation responses (T_mitigation). Thus, the efficiency of the framework and low tendency of misclassifications are shown in relation to different types of attacks.

Table 5
Multi-Class Error Rates, Correlation, and Response Time Metrics (FPR, FNR, MCC, and T_mitigation)

Attack Category	FPR (%)	FNR (%)	MCC	T_mitigation (ms)
SQL Injection (SQLi)	1.15	1.02	0.96	1.25
Cross-Site Scripting (XSS)	1.42	1.30	0.95	1.34
Command Injection	1.28	1.18	0.95	1.29
HTTP Floods (App-Layer)	1.85	1.62	0.93	1.48

4.5 Ablation Study and Sensitivity Analysis

To understand the effect of individual components of architecture and to perform stability analysis with varying parameters, an extensive ablation study and sensitivity analysis was carried out. In this ablation study, functional aspects of PISMA framework are evaluated through selective disabling to understand their contribution to classification accuracy and response time. Disabling of in-switch programmable feature extraction module leads to a sharp decrease in binary classification accuracy to 64.85% because of the presence of legacy Layer 3 and Layer 4 headers which completely mask payload based application layer injection patterns. Similarly, disabling of sliding window based behavioral aggregation decreases the performance in detecting stealthy and low rate HTTP floods, with a recall rate for volumetric anomaly detection decreasing by over 22%. Furthermore, disabling of confidence aware thresholding mechanism and strict binary blocking for all identified cases increase legitimate packet drop rate by 38.5%, proving that confidence scoring is imperative to avoid service disruption.

Table 6
Ablation Study Performance Comparison Under Component Removal

Configuration Variant	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	T_mitigation (ms)
Full PISMA	98.75	98.22	98.94	98.59	1.12
Without Programmable Feature Extraction	64.85	62.10	65.40	63.71	2.85
Without Sliding-Window Aggregation	76.20	74.80	75.90	75.34	2.10
Without Confidence-Aware Thresholding	85.40	83.15	86.20	84.65	1.45

Alongside the results of the ablation study, a sensitivity analysis was performed to investigate the influence of changing the sliding window time horizon (ΔT) and confidence thresholds (τ_low and τ_high). As can be seen from Figure 12, by looking at various window lengths from 1 second up to 15 seconds, it becomes clear that a middle-length window with a horizon of 5 seconds offers an ideal balance between the stability of temporal features and the threat detection delay. Short windows often create statistical noise because of not enough data collected, while long windows cause unnecessary delays. Moreover, changing the value of the high confidence threshold between 0.75 and 0.95, it is possible to conclude that a value of 0.85 allows keeping the false positive rate under 1.5%. Table 6 shows the results of ablation study in numbers.

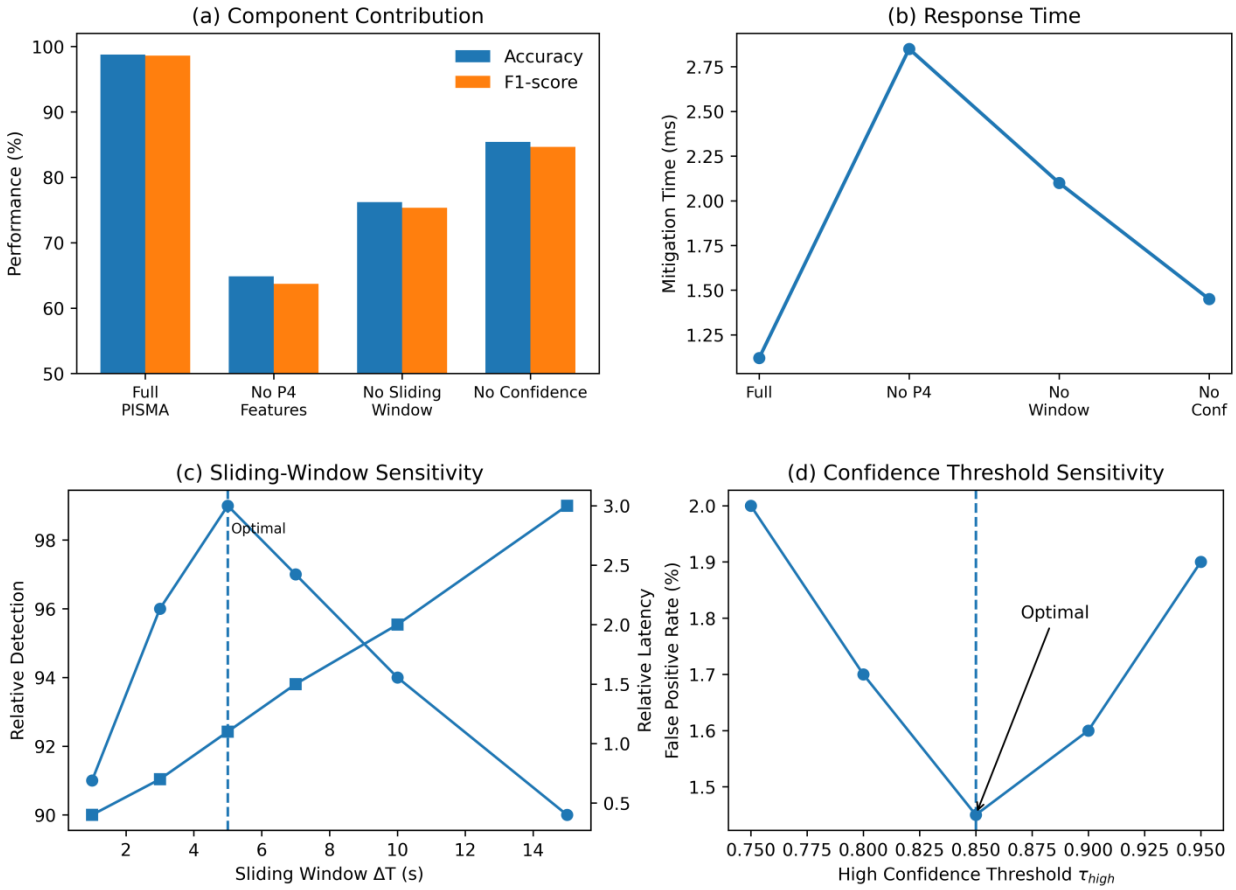

Figure 12. Ablation Study and Sensitivity Analysis of PISMA

5. PISMA Limitations

While the proposed PISMA framework offers high accuracy, low latency, and robust multi-class threat mitigation in SD-IoT environments, several operational limitations and deployment challenges should be noted:

1. Encrypted HTTPS and TLS Traffic Inspection: The current in-switch feature extraction module is tailored specifically for plaintext HTTP-based RESTful services; nevertheless, modern IoT ecosystems increasingly employ encrypted HTTPS and TLS transport protocols. Encryption precludes deep packet inspection at line rate without specialized cryptography offloading, TLS proxy support, and hardware-assisted metadata processing.
2. Hardware Capabilities of Programmable Switches: Implementation of complex behavioral analysis, feature aggregation, and stateful tracking on the P4 data plane faces significant

hardware limitations. Programmable switches have limited SRAM and TCAM match-action memory capabilities and thus cannot accommodate the state table scale and history length needed for advanced anomaly detection.

3. Scalability in Hyper-Scale SD-IoT Domains: The testbed setup used in the experiment mainly concerns local or small-scale SDN control domains. Application of PISMA in the hyper-scale SD-IoT domain poses serious cross-domain scalability issues, which are associated with distributed synchronization of multiple controllers and additional overhead of east-west traffic leading to the control-plane bottleneck in case of cascading distributed attacks.

6. Conclusion and Future Work

This paper proposes a novel PISMA framework for securing application-layer services in Software-Defined Internet of Things (SD-IoT) systems. Utilizing in-switch feature extraction based on P4 programming language, sliding-window behavioral aggregation, hybrid deep learning classifier (CNN + bidirectional RNN), and confidence-driven policy orchestration, PISMA enables efficient combination of high-speed forwarding and intelligent attack mitigation. Five benchmark datasets – CICIoT2023, ToN_IoT, IoT-ID20, UNSW-NB15, and Edge-IIoTset – were used in extensive experiments and revealed that the proposed PISMA framework outperforms conventional ensemble algorithms in terms of binary classification accuracy (98.75%), MCC (0.97), and mitigation latency (1.12 ms). Multi-class evaluations and ablation studies confirmed the importance of each architecture component for precise attack classification and reduction of false-positive service disruptions. Further research directions include extending the programmable data-plane parsing logic for secure handling of encrypted traffic telemetry based on TLS metadata inspection and zero-knowledge traffic proofs. Additionally, multi-controller synchronization protocols are planned to be incorporated for enhanced cross-domain threat intelligence sharing, and the deep learning engine will be modified for federated learning frameworks allowing decentralized privacy-preserving model training.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Ameer El-Sayed and Ehab Rushdy; methodology, Ameer El-Sayed and Mohamed Hemdan; software, Ameer El-Sayed and Mohamed Hemdan; validation, Ameer El-Sayed, Mohamed Hemdan, and Hanaa Hamza; formal analysis, Ehab Rushdy; investigation, Ameer El-Sayed and Hanaa Hamza; resources, Ameer El-Sayed and Hanaa Hamza; data curation, Ameer El-Sayed; writing—original draft preparation, Ameer El-Sayed and Mohamed Hemdan; writing—review and editing, Ameer El-Sayed, Ehab Rushdy, and Hanaa Hamza.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflicts of interest.
Acknowledgement: This research was not funded by any grant.
Ethical approval: Not applicable.

References

[1] W. Wu, H. Fouzi, S. Sidi-Mohammed, and S. Ying, "Improving cyber-attack detection in Internet of Medical Things using ensemble deep learning methods," *Cluster Computing,* vol. 28, p. 891, 2025.

[2] A. El-Sayed, W. Said, A. Tolba, Y. Alginahi, and A. A. Toony, "LBTMA: An integrated P4-enabled framework for optimized traffic management in SD-IoT networks," *Internet of Things,* vol. 28, p. 101432, 2024.

[3] M. Maazalahi and S. Hosseini, "A Novel Hybrid Method Using Grey Wolf Algorithm and Genetic Algorithm for IoT Botnet DDoS Attacks Detection," *International Journal of Computational Intelligence Systems,* vol. 18, p. 61, 2025.

[4] A. El-Sayed, H. Ramadan, E. R. Mohamed, and O. M. Elkomy, "SFARP: a multi-layered real-time security framework for hybrid ARP and DDoS attack defense in SD-IoT networks," *Scientific Reports,* vol. 15, p. 43479, 2025.

[5] A. Alyanbaawi, A. El-Sayed, N. Salah, W. Said, M. Elmezain, and O. Elkomy, "MC-LBTO: secure and resilient state-aware multi-controller framework with adaptive load balancing for SD-IoT performance optimization," *Scientific Reports,* vol. 15, p. 44660, 2025.

[6] A. El-Sayed, A. A. Toony, F. Alqahtani, Y. Alginahi, and W. Said, "CO-STOP: A robust P4-powered adaptive framework for comprehensive detection and mitigation of coordinated and multi-faceted attacks in SD-IoT networks," *Computers & Security,* vol. 151, p. 104349, 2025.

[7] A. Hassan, M. M. Khashaba, E. R. Mohamed, and A. El-Sayed, "FlowPrint-P4: Lightweight Behavioral Anomaly Detection for DDoS Mitigation in Programmable Networks," *International Journal of Computers and Informatics (Zagazig University),* vol. 11, pp. 16-35, 2026.

[8] A. El-Sayed, W. Said, A. Tolba, Y. Alginahi, and A. A. Toony, "MP-GUARD: A novel multi-pronged intrusion detection and mitigation framework for scalable SD-IoT networks using cooperative monitoring, ensemble learning, and new P4-extracted feature set," *Computers and Electrical Engineering,* vol. 118, p. 109484, 2024.

[9] D. A. Sivasakthi, A. Sathiyaraj, and R. Devendiran, "HybridRobustNet: enhancing detection of hybrid attacks in IoT networks through advanced learning approach," *Cluster Computing,* vol. 27, pp. 5005-5019, 2024.

[10] A. El-Sayed, M. N. Hemdan, N. Abdelkarim, E. Rushdy, and H. M. Hamza, "AMCS: Adaptive Multi-Controller SDN Security with Stateful Traffic Intelligence for Fast and Accurate Multi-Vector Attack Detection," *International Journal of Advanced Computer Science and Applications,* vol. 17, 2026.

[11] R. F. Kapourchali, R. Mohammadi, and M. Nassiri, "P4httpGuard: detection and prevention of slow-rate DDoS attacks using machine learning techniques in P4 switch," *Cluster Computing,* vol. 27, pp. 8047-8064, 2024.

[12] W. I. Khedr, A. E. Gouda, and E. R. Mohamed, "P4-HLDMC: A novel framework for DDoS and ARP attack detection and mitigation in SD-IoT networks using machine learning, stateful P4, and distributed multi-controller architecture," *Mathematics,* vol. 11, p. 3552, 2023.

[13] A. El-Sayed, A. A. Toony, A. Tolba, F. Alqahtani, Y. Alginahi, and W. Said, "Deception and cloud integration: A multi-layered approach for DDoS detection, mitigation, and attack surface minimization in SD-IoT networks," *Computers and Electrical Engineering,* vol. 126, p. 110543, 2025.

[14] K. M. Hosny, A. E.-S. Gouda, and E. R. Mohamed, "New detection mechanism for distributed denial of service attacks in software defined networks," *International Journal of Sociotechnology and Knowledge Development (IJSKD),* vol. 12, pp. 1-30, 2020.

[15] P. Zhang, Y. Yu, L. Tan, S. He, J. Wang, and A. El-Sayed, "Cascading Failure Dynamics and Edge-Intelligent Defense in Space-Air-Ground Integrated Networks for Internet of Things," *Computers, Materials and Continua,* vol. 88, 2026.

[16] V. Hnamte, A. A. Najar, C. Laldinsanga, J. Hussain, and L. Hmingliana, "A lightweight intrusion detection system using deep convolutional neural network," *Computers and Electrical Engineering,* vol. 127, p. 110561, 2025.

[17] T. Alasali and O. Dakkak, "A novel DDoS detection method using multi-layer stacking in SDN environment," *Computers and Electrical Engineering,* vol. 120, p. 109769, 2024.

[18] F. Wahab, S. Ma, X. Liu, Y. Zhao, A. Shah, and B. Ali, "A ranked filter-based three-way clustering strategy for intrusion detection in highly secure IoT networks," *Computers and Electrical Engineering,* vol. 127, p. 110514, 2025.

[19] H. Ramadan, A. El-Sayed, E. R. Mohamed, and O. M. Elkomy, "P4-TAP: A Data Plane Framework for Traffic Aggregation and Prioritization in Time-Sensitive IoT Networks," *International Journal of Computers and Informatics (Zagazig University),* vol. 9, pp. 55-73, 2025.

[20] N. Salah, A. El-Sayed, and O. M. Elkomy, "MAFAF: Mobility-Aware Flow Anchoring and Dynamic State Migration in Edge SDN Using P4," *International Journal of Computers and Informatics (Zagazig University),* vol. 9, pp. 74-93, 2025.

[21] H. S. Ilango, M. Ma, and R. Su, "A FeedForward–Convolutional Neural Network to Detect Low-Rate DoS in IoT," *Engineering Applications of Artificial Intelligence,* vol. 114, p. 105059, 2022.

[22] M. Aslam, D. Ye, A. Tariq, M. Asad, M. Hanif, D. Ndzi*, et al.*, "Adaptive machine learning based distributed denial-of-services attacks detection and mitigation system for SDN-enabled IoT," *Sensors,* vol. 22, p. 2697, 2022.

[23] P. Chauhan and M. Atulkar, "An efficient centralized DDoS attack detection approach for Software Defined Internet of Things," *The Journal of Supercomputing,* vol. 79, pp. 10386-10422, 2023.

[24] N. M. Yungaicela-Naula, C. Vargas-Rosales, J. A. Pérez-Díaz, and D. F. Carrera, "A flexible SDN-based framework for slow-rate DDoS attack mitigation by using deep reinforcement learning," *Journal of Network and Computer Applications,* vol. 205, p. 103444, 2022.

[25] M. Cherian and S. L. Varma, "Secure SDN–IoT framework for DDoS attack detection using deep learning and counter based approach," *Journal of Network and Systems Management,* vol. 31, p. 54, 2023.

[26] A. A. Najar and S. M. Naik, "Cyber-secure SDN: A CNN-based approach for efficient detection and mitigation of DDoS attacks," *Computers & Security,* vol. 139, p. 103716, 2024.

[27] M. Revathi and S. K. Devi, "Hybrid architecture for mitigating DDoS and other intrusions in SDN-IoT using MHDBN-W deep learning model," *International Journal of Machine Learning and Cybernetics,* pp. 1-22, 2024.

[28] F. Musumeci, A. C. Fidanci, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-enabled DDoS attacks detection in P4 programmable networks," *Journal of Network and Systems Management,* vol. 30, pp. 1-27, 2022.

[29] R. F. Kapourchali, R. Mohammadi, and M. Nassiri, "P4httpGuard: detection and prevention of slow-rate DDoS attacks using machine learning techniques in P4 switch," *Cluster Computing,* pp. 1-18, 2024.

[30] Z. Long and W. Jinsong, "A hybrid method of entropy and SSAE-SVM based DDoS detection and mitigation mechanism in SDN," *Computers & Security,* vol. 115, p. 102604, 2022.

[31] W. I. Khedr, A. E. Gouda, and E. R. Mohamed, "FMDADM: A multi-layer DDoS attack detection and mitigation framework using machine learning for stateful SDN-based IoT networks," *Ieee Access,* vol. 11, pp. 28934-28954, 2023.

[32] J. Ma, W. Su, Y. Li, Y. Yuan, and Z. Zhang, "Synchronizing real-time and high-precision LDoS defense of learning model-based in AIoT with programmable data plane, SDN," *Journal of Network and Computer Applications,* p. 103916, 2024.

[33] U. H. Garba, A. N. Toosi, M. F. Pasha, and S. Khan, "SDN-based detection and mitigation of DDoS attacks on smart homes," *Computer Communications,* vol. 221, pp. 29-41, 2024.

[34] G. Kirubavathi, I. Sumathi, J. Mahalakshmi, and D. Srivastava, "Detection and mitigation of TCP-based DDoS attacks in cloud environments using a self-attention and intersample attention transformer model: KG et al," *The Journal of Supercomputing,* vol. 81, p. 474, 2025.

[35] P. Chaudhary, A. Singh, and B. Gupta, "Dynamic multiphase DDoS attack identification and mitigation framework to secure SDN-based fog-empowered consumer IoT Networks," *Computers and Electrical Engineering,* vol. 123, p. 110226, 2025.

[36] Z. Ullah, F. Arif, Q. M. U. Haq, N. A. Khan, I. U. Din, A. Almogren*, et al.*, "Hybrid CNN-LSTM Model for DDoS Attack Detection in Internet of Things-based Healthcare Industry 5.0," *IEEE Internet of Things Journal,* 2025.

[37] A. El-Sayed, A. Tolba, N. Alalwan, Y. Alginahi, and W. Said, "CATOR: A confidence-adaptive dual-plane in-switch orchestrated resilience framework for multi-vector DDoS defense in software-defined IoT," *Computers and Electrical Engineering,* vol. 130, p. 110894, 2026.