SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Design and Implementation of a Novel Machine Learning Framework for DDoS Attack Detection and Mitigation in SDNs for IoT Environments

Jay Chand[1], Akash Sanghi[2]*

[1]Research Scholar, Department of CSE, Invertis University, Bareilly, India, Email: jaychandvbs04@gmail.com
[2]*Department of CSE, Invertis University, Bareilly, India, Email: sanghiakash@gmail.com

Abstract

With the emergence of applications of Internet of Things (IoT) that are based on Software-Defined Networking (SDN), there is a flexible but insecure environment for distributed denial-of-service attacks that can saturate links, switches, controllers, or services. A recently proposed machine-learning framework enables the real-time IoT traffic monitoring, flow-level preprocessing and feature engineering, hybrid threat detection, and adaptive SDN mitigation, and the application of security/privacy controls to detect and mitigate hybrid threats in IoT-enabled SDN. The two-algorithm implementation examines Random Forest and XGBoost, while the framework retains an anomaly component for uncharacterized behaviors. Besides, the paper employs OpenFlow statistics for light-weight monitoring and the controller translates the model decisions into graduated actions, such as rate limiting, flow dropping, rerouting, and isolation of devices, and a reproducible Mininet-based evaluation protocol is defined based on the accuracy, precision, recall, F1-score, ROC-AUC, false-positive rate, detection latency, controller overhead, and network recovery indicators. In the controlled prototype implementation, Random Forest and XGBoost achieved 95.40% and 95.38% accuracy, respectively, with ROC-AUC values of 98.22% and 98.20%. These classifier-level measurements support the two-algorithm implementation, while complete controller and network-level validation remains to be carried out in Mininet/SDN. The design satisfies all objectives in the doctoral synopsis and can be used as a basis for implementation.

Keyword: Distributed denial-of-service; Internet of Things; Software-defined networking; Machine learning; Anomaly detection; OpenFlow; Adaptive mitigation

1. Introduction

The Internet of Things (IoT) has connected resource-constrained sensors, consumer appliances, cameras, industrial controllers, gateways, and cyber-physical systems to the Internet. This creates new operational opportunities but also expands the attack surface available to adversaries. IoT devices can be compromised and coordinated as a botnet to launch distributed denial-of-service (DDoS) traffic at a scale that overwhelms network links, application servers, and control infrastructure. DDoS attacks have the objective of availability degradation rather than unauthorized access to data; the attacker wants to make a legitimate service unavailable by exhausting limited network or computation resources. IoT traffic is highly heterogeneous and low-cost devices often operate with limited update mechanisms, weak credentials, and long deployment lifetimes. While software-defined networking (SDN) is a good basis for protection, it can also be a target of attack due to the separation of the control plane and the forwarding plane and the controller's logically centralized view of the network. OpenFlow-enabled switches can export flow statistics, and the controller can install or modify forwarding rules according to current conditions. This programmability enables rapid security orchestration but also forms a critical dependency on the controller; a DDoS campaign that triggers large volumes of new flows can consume switch table capacity and increase packet-in events, overwhelming controller processing. An effective SDN-IoT defense must be able to detect attacks early enough to protect both the service endpoint and the SDN control plane (Yin et al., 2018; Swami et al., 2023).

Static thresholds and signature-based intrusion detection systems are challenging to maintain. IoT legitimate traffic can vary depending on the particular firmware, users, time of day, mobility, and application needs. Attackers can also vary the packet rates, protocols, source distributions, and temporal patterns to bypass the thresholds.

Machine learning provides means to learn traffic patterns from data and differentiate between legitimate and attack traffic based on multiple features instead of single thresholds. Tree-based models, Support Vector Machines, Neural models, Entropy-based methods, and Reinforcement Learning have been shown to be useful in DDoS detection and mitigation (Kalkan et al., 2018; Harikrishna & Amuthan, 2021; Yungaicela-Naula et al., 2023).

Several gaps are relevant to SDN-enabled IoT deployments. First, high-performance models may incur large numbers of features that incur unnecessary collection and inference overhead at the controller. Second, most detection studies stop at classification and do not close the loop between detection and mitigation. Third, purely supervised models can excel on known classes but are unable to generalize to previously unseen or modified attacks. Fourth, aggressive blocking can damage legitimate IoT traffic if a false positive is immediately translated into a deny rule. The monitoring pipeline can create privacy and integrity issues if raw device-level data is collected without minimization, access control, or secure communications, motivating a system in which monitoring, pre-processing, detection, mitigation, and security governance comprise a unified control loop.

This work proposes the ML-SDIoT-DM framework for DDoS (Distributed Denial of Service) attack detection and mitigation in SDNs (Software Defined Networks) hosting IoT (Internet of Things) environments. Specifically, this work covers the five objectives put forth in the doctoral work's synopsis. They are IoT monitoring in real time, preprocessing and feature engineering to improve feature extraction, machine learning-based anomaly or threat detection, integration of detection into the SDN network control plane, and security and privacy. The synopsis also brings forth lightweight feature engineering, a hybrid supervised/unsupervised detection approach, and adaptive controller-based mitigation. We present the elements of this implementation framework in this paper. Figure 2 presents the overall framework and Figure 3 presents the online decision-making process.

The most novel part is the combination of a lightweight flow-telemetry layer with a two-stage intelligence mechanism and a graduated mitigation policy. The supervised portion detects traffic patterns that are found in the training data while the anomaly component serves as a secondary line of defense if the traffic pattern is different from what is expected. The results are fused to calculate a threat score, which defines whether the controller should continue to transmit, rate-limit, redirect, temporarily isolate, or block a flow. It reduces reliance on a single classifier and avoids treating each uncertain event as a confirmed attack. A security and privacy layer regulates telemetry transport, access rights, model versions, event retention, and feedback.

2. Literature Review and Research Gap

DDoS defense in SDN has evolved along three main directions: statistical or entropy-based detection, supervised or deep-learning classification, and controller-assisted adaptive mitigation. Statistical approaches are appealing as computationally light and potentially fast in response to abrupt distributional changes in the traffic. Kalkan et al. (2018) proposed JESS, which uses the joint entropy to detect and mitigate the DDoS conditions in SDN. Swami et al. (2023) demonstrated an interquartile-range-based mechanism implemented at the SDN controller, which emphasized the early detection, mitigation, controller CPU utilization, Packet-In messages, and detection time. Such works highlight the benefits of controller-level observation, but also demonstrate why one may need to adapt the rigid statistical assumptions when normal IoT traffic is itself irregular.

Machine-learning approaches attempt to learn the decision boundary directly from traffic features. Doshi et al. (2018) demonstrated that IoT-specific network behaviors and flow-based, protocol-agnostic features can be used to design a low-cost DDoS detection for consumer IoT device. Dasari and Devarakonda (2022) compared several classifiers on CIC-DDoS2019 and demonstrated that the choice of the model is material for the classification performance. Kavitha and Ramalakshmi (2024) specifically studied ML-based DDoS detection and mitigation in SDNs for IoT environments and reported very high classification accuracy in their experimentally chosen scenario, reiterating the potential of the integrated ML-SDN defense. However, one must be careful to apply the accuracy reported by these papers to a particular use case, since the choice of the dataset, balancing of the classes, feature engineering, topology, attack type, and validation paradigm can be significant factors in the outcome.

Meanwhile, the researchers have also worked on making the mitigation process autonomous rather than merely the detection. Yin et al. (2018) designed a software-defined IoT framework in which the SDN control layer can facilitate both DDoS detection and mitigation. Liu et al. (2018) explored deep reinforcement learning for smart mitigation, which relied on the global visibility provided by the SDN controller to learn the mitigation policies. Yungaicela-Naula et al. (2022) integrated the deep-learning detection and the deep-reinforcement-learning prevention for slow-rate DDoS attacks, and the later SDN/NFV framework continued in this vein with reinforcement learning and moving-target defense (Yungaicela-Naula et al., 2023). These works are important in

context of the low-rate DDoS attacks, which can be more difficult to distinguish from normal demand, and a mitigation should be scaled to the confidence level in the detection and the network state.

Harikrishna and Amuthan (2021) proposed an SDN-based DDoS prevention mechanism, which relied on the rival-model penalized self-organizing map. Their work emphasized the value of adaptive learning and topological traffic representation for distinguishing malicious and legitimate flows. Yan et al. (2018) proposed a multi-level mitigation framework for industrial IoT, which demonstrated that defense can be distributed across edge, fog, and cloud layers with the orchestration by SDN to coordinate network behavior. In general, these papers emphasize the need for effective protection by timely telemetry, controller scalability, and mitigation safety, which is aware of heterogeneous IoT behavior.

The research gap addressed by the study is therefore both architectural and algorithmic. The literature typically aims at optimizing a specific component such as classification, entropy detection, reinforcement learning or flow-rule mitigation, whereas an operational IoT defense requires an end-to-end control loop. The proposed framework explicitly maps each of the doctoral objectives to a deployable module including privacy and security controls usually seen as an external concern. It also differentiates known-attack classification and novelty detection and uses a graduated response policy rather than an immediate binary block likely to increase robustness to changing traffic distribution as well as lower the operational cost of false positives.

The primary research gaps in this study include adaptive and evolving distributed denial-of-service (DDoS) detection, processing and scale, software-defined networking (SDN) and Internet of Things (IoT) integration, privacy and security issues, and proactive and reactive approaches.

Table 1. Selected prior approaches and the research gap addressed by the proposed framework

Study	Primary approach	Strength	Gap relevant to this study
Doshi et al. (2018)	IoT-specific ML detection	Low-cost flow-based features	Does not provide a complete SDN mitigation/control architecture.
Kalkan et al. (2018)	Joint entropy in SDN	Statistical detection of unfamiliar attacks	Fixed statistical behavior may be sensitive to legitimate traffic shifts.
Yin et al. (2018)	SD-IoT detection and mitigation	Controller-integrated defense	Limited emphasis on hybrid supervised/unsupervised ML and privacy governance.
Harikrishna & Amuthan (2021)	Self-organizing map prevention	Adaptive traffic mapping	Cloud-oriented design; not explicitly aligned to all IoT monitoring/privacy objectives.
Swami et al. (2023)	IQR-based controller defense	Fast controller-side detection	Statistical rule rather than multi-model learning and confidence-aware response.
Yungaicela-Naula et al. (2023)	DL + RL + NFV/MTD	Autonomous mitigation	Higher architectural complexity; privacy/model-governance not the central focus.
Kavitha & Ramalakshmi (2024)	ML DDoS detection and mitigation in SDN-IoT	Strong integrated ML-SDN focus	Opportunity remains for explicit hybrid novelty detection, graduated response, and security/privacy assurance.

3. Materials and Methods

The overall approach for detecting DDoS attacks in SDNs using IoT environments includes data acquisition, preprocessing, normalization, cleaning of data, traffic generation, splitting the dataset, training, testing, developing a classification model, and classifying, as depicted in Figure 1.

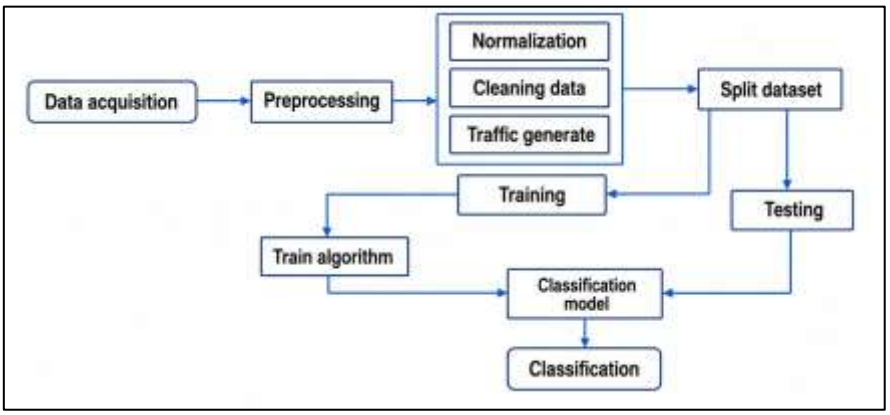

Figure 1. Methodology for DDoS attack in SDNs using IoT environments.

3.1 Research Design and System Architecture

The study was carried out using an iterative design-evaluation framework. The artifact was designed as a security framework that employs machine learning algorithms in the logical control layer of an SDN. The framework consists of a data plane, control plane, applications/intelligence plane, and a cross-cutting security/privacy layer. The data plane comprises IoT devices, gateways, and OpenFlow-enabled switches responsible for switching and forwarding traffic. The control plane consists of an SDN controller such as Ryu, POX, or ONOS that carries out topology discovery, flow statistics requests, asynchronous event notification, and mitigation rule installation. Finally, the application plane entails telemetry processing, feature engineering, ML inference, threat-scoring model, mitigation logic, event logging, and model management components. Figure 2 shows the relationship between the planes.

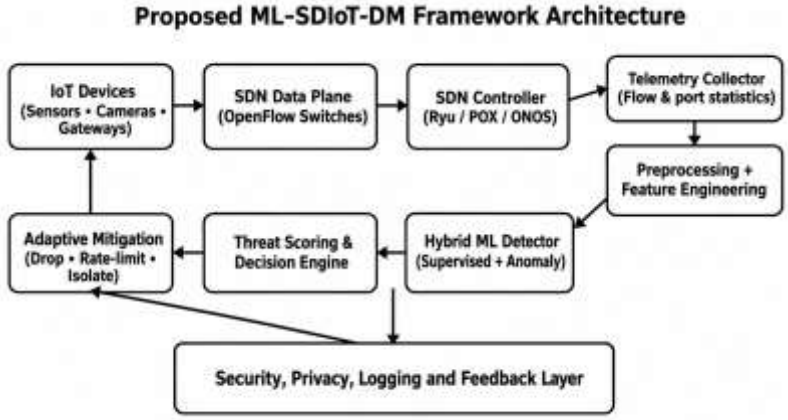

Figure 2. Proposed ML-SDIoT-DM framework architecture.

This layered approach complies with the SDN design principle of keeping the forwarding elements simple while making the control logic programmable. It also enables the ML detector to analyze controller-visible traffic summaries instead of inspecting payloads. Using flow-level telemetry is important in an IoT context due to reduced processing overhead and avoiding the collection of irrelevant application-layer contents. Furthermore, the framework is designed such that inference can run as part of the controller process for small-scale deployments or as a separate security service that has a secure northbound interface to communicate with the controller in larger deployments.

3.2 Objective 1: Real-Time Monitoring of IoT Devices

The telemetry layer enables real-time monitoring by regularly probing the flow and port statistics and inspecting controller events related to new flows. It should be frequent enough to identify a rapidly developing flood but not too aggressive that monitoring traffic itself would create a burden on the controller. Each flow record generated as a result of inspection is condensed into a short summary containing information relevant to any potential DDoS attack, for instance, packet count, byte count, flow lifetime, protocol, source and destination characteristics, inter-arrival characteristics, and new flow rate. For switches supporting exposed port counters, packet-drop and receive/transmit rates may be added to the list. The telemetry collector timestamps each record and attaches it to a switch, a port, and, where available, a logical device identity.

Monitoring can be used to provide both security and generic operational telemetry. A sudden increase of new flows, short-lived connections, unexpected packet-to-byte ratios, or unusually high concentration of traffic to a particular destination could be signs of a flood. At the same time, normal IoT devices activity could feature bursts of traffic. Because of this, the monitoring layer itself does not attempt to distinguish between benign and abusive traffic flows. Instead, it provides normalized telemetry to the next layer, intelligence, which is responsible for analyzing this information. This approach avoids defining a single metric that would trigger an alert on its own.

3.3 Objective 2: Data Preprocessing and Feature Extraction

Raw SDN telemetry can contain missing values, duplicates, categorical protocol fields, highly skewed counters, and imbalanced measurements with different numerical scales. The preprocessing pipeline first filters out invalid or duplicate records and applies consistent handling of missing fields. Categorical values are encoded, while continuous variables are normalized using statistics learned from the training partition (only). Min-max normalization is a good choice when one needs to enforce a fixed bounded scaling, however, robust scaling may be a better option if extreme attack values could overwhelm the normal range. The same transformation objects are kept with the model and used during online inference to avoid training-serving inconsistency.

Feature engineering transforms counters to behaviorally meaningful rates and ratios. Candidate features include packets per second, bytes per second, average packet size, flow duration, packets per flow, bytes per flow, new-flow rate, bidirectional packet ratio, SYN or protocol-specific proportions (where available), unique source count per destination, destination concentration, and inter-arrival statistics. Then, a feature selection stage (lightweight) identifies and eliminates redundant or irrelevant features (e.g., by applying correlation-based filtering, mutual information, recursive feature elimination, or tree-based feature importance) to identify a reduced set of features with maximal overall offline accuracy. The reduced set should be implementable in the controller in real-time given the computational budget. Table 2 defines the feature subsets.

Table 2. Recommended lightweight flow-level feature groups

Feature group	Example variables	Purpose
Volume and rate	packet_count, byte_count, packets/s, bytes/s	Identify sudden load growth and high-rate floods.
Temporal	flow_duration, inter-arrival mean/variance, new-flow rate	Capture short-lived bursts and abnormal timing.
Protocol/flag	protocol type, SYN proportion, ICMP/UDP share	Differentiate protocol-specific attack behavior.
Dispersion/concentration	unique sources per destination, destination share, source entropy proxy	Identify distributed sources converging on a victim.
Bidirectional behavior	forward/reverse packet ratio, response ratio	Detect unbalanced request-heavy flows.
Controller context	Packet-In rate, active-flow count, port drops	Estimate stress on SDN control and forwarding resources.

3.4 Objective 3: Machine-Learning-Based Anomaly or Threat Detection

The detection engine has two stages. The first (supervised) stage compares Random Forest and XGBoost. These two algorithms are retained for implementation because they can handle nonlinear flow-level features and can be evaluated for both predictive performance and real-time use. The models are trained using labeled benign and DDoS flows according to an 80:20 training/testing division and stratified k-fold cross-validation within the training sets to select hyperparameters. Class imbalance should be addressed by class weighting, careful under-sampling or over-sampling, and threshold selection, not just reporting the global accuracy. The preferred supervised model is chosen according to a combination of recall, false positive rate, inference-time, memory requirements, and controller overhead rather than just classification accuracy.

The implementation steps of the two supervised algorithms used in the prototype are presented in Algorithms 1 and 2.

Algorithm 1: Random Forest Model for DDoS Attack Detection

Input: Preprocessed flow-level dataset D with benign and DDoS labels.
Output: Trained Random Forest model M_RF and holdout predictions.

1. Load the flow-level dataset D.
2. Remove the invalid or duplicated records and apply the preprocessing and feature selection steps described in Section 3.3
3. Stratify the data set into a Training set 80% and a Testing set 20%
4. The testing set should remain hidden during the model building process.
5. Build Random Forest model: Number of trees 250, Max depth 16, Class weight Balanced.
6. Apply 5-fold stratified cross-validation on the training set.
7. for each fold k in {1 to 5} do
   7.1 Use four folds for training and one fold for validation.
   7.2 Fit the Random Forest model using the training folds.
   7.3 Generate validation predictions and class probabilities.
   7.4 Compute Accuracy, Precision, Recall, F1-score, ROC-AUC, and False-Positive Rate.
   7.5 Store the fold-level performance.
   end for
8. 8. Train MRF on complete 80% of the training set with chosen settings.
9. Predict labels and probabilities for the 20% of the testing set.
10. Evaluate MRF on the testing set and record prediction time.
11. Save MRF and its preprocessing objects for the SDN-controller-side inference.

Algorithm 2: XGBoost Model for DDoS Attack Detection

Input: Preprocessed flow-level dataset D with benign and DDoS labels.
Output: Trained XGBoost model M_XGB and holdout predictions.

1. Load the flow-level dataset D.
2. Remove invalid or duplicate data records and apply the preprocessing and feature-selection operations described in Section 3.3.
3. Stratify the dataset into Training set = 80% and Testing set = 20%.
4. Keep the testing set unseen during model development.
5. Initialize the XGBoost model: Number of Estimators = 250, Maximum Depth = 5, Learning Rate = 0.08, with subsampling enabled.
6. Apply 5-fold stratified cross-validation on the training set.
7. for each fold k in {1 to 5} do
   7.1 Use four folds for training and one fold for validation.
   7.2 Fit the XGBoost model using the training folds.
   7.3 Generate validation predictions and attack probabilities.
   7.4 Compute Accuracy, Precision, Recall, F1-score, ROC-AUC, and False-Positive Rate.
   7.5 Store the fold-level performance.
   end for
8. Train M_XGB on the complete 80% training set using the selected settings.
9. Predict the labels and probabilities for the 20% testing set.
10. Evaluate M_XGB on the testing set and record prediction time.
11. Save M_XGB and its preprocessing objects for SDN-controller-side inference.

An anomaly detector is the second stage that is trained on either normal or representative traffic to detect outliers. Possible options include Isolation Forest, one-class SVM, or an autoencoder for sufficiently large traffic data sets. The results of the online phase of the supervised classifier are attacking probabilities or the confidence of each class, while the anomaly detector outputs anomaly scores. The rule-based fusion layer combines attack scores with the contextual information, such as the load of the destination or source dispersion, to produce a final threat score. It directly incorporates the synopsis concept through the supervised learning method, which can identify known attacks, while zero-day anomalies are identified using the unsupervised technique.

The model outputs are designed such that uncertainties are explicit. If supervised attack probability is high and anomaly score is high, then immediate countermeasures can be taken. On the other hand, if the evidence for attack is mixed or weak, rate limiting and additional observation can be imposed. This is particularly important in the IoT setting, where a large-scale coordinated attack could look very similar to a normal update procedure from a single IoT device. Finally, note that the framework views ML model outputs as evidence to be evaluated and acted upon, not as actions in and of themselves.

3.5 Objective 4: Integration with SDN Network Control Systems

After training, the chosen model and preprocessing objects are loaded by the controller-side detection service. For every monitoring interval, that service receives a feature vector from the controller-side, applies preprocessing, receives supervised and anomaly scores, and reports the resulting threat assessment to the mitigation manager.

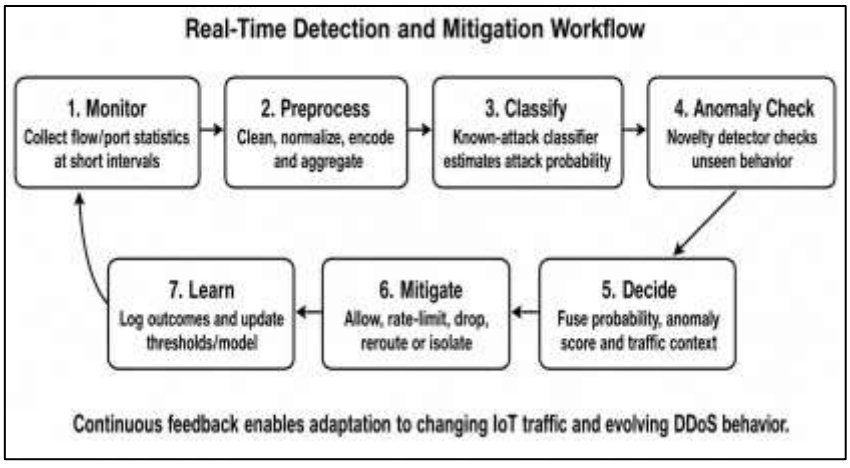

Figure 3. Real-time detection and mitigation workflow.

The latter converts the assessment to OpenFlow actions: depending upon confidence and severity, the controller may continue normal forwarding, apply a meter or queue to rate-limit a source, install a temporary drop rule, or

redirect the suspicious traffic to a scrubbing or honeypot path, or isolate a device or switch port. Figure 3 depicts the resulting closed loop.
Mitigation is intentionally graduated: a temporary rate limit is less disruptive to the network than a permanent block and so is appropriate for less confident events. Higher confidence events with high rate of attack may be mitigated with a short-lived drop rule with carefully chosen idle and hard timeouts. Device isolation is reserved for more persistent malicious behaviors or compromised endpoints. Flow rule duration is important here: denying all traffic to a device will do more harm than good if the attacked host is still under attack, but the deny rule will age out. The controller accordingly tracks what action was taken, its expiry time, what evidence led to it, and what subsequent traffic arrived. If the subsequent traffic is normal, then the mitigation can be relaxed.

Reactive and proactive modes of operation are supported. In reactive mode, mitigation takes place after an attack has been detected; proactive operations make changes to thresholds or sampling rates if there are signs of unusual patterns. The same framework can also use current traffic baselines to adjust thresholds depending on device classes or time windows. Such adaptations should be carefully constrained and audited to prevent an attacker from gradually poisoning the baseline, thus normalizing their attack traffic.

3.6 Objective 5: Security and Privacy Assurance

Security and privacy present what our framework requires. We use authentication and encryption of controller-switch and controller-application communication as much as possible within the platform. We have also put in place that administrative access to the controller and ML service be restricted by role-based access control, which includes the least privilege approach. We use version control for model files, preprocessing parameters, and policy configurations, and also put in place integrity checks to help prevent ML model poisoning and unauthorized changes to the forwarding policies. Privacy is achieved via data minimization, which means we use flow-level metadata for detection and do not include flow content. Telemetry data must be kept only for the time required to achieve training purposes, incident forensics, and validation, with access logged and export restricted. We may apply de-identification of device identifiers to research datasets when we are able to. In Figure 4, we present the security and privacy controls as well as model governance. All of these elements support the protected use of sensitive IoT data at each stage and put in place the required communication and authentication measures.

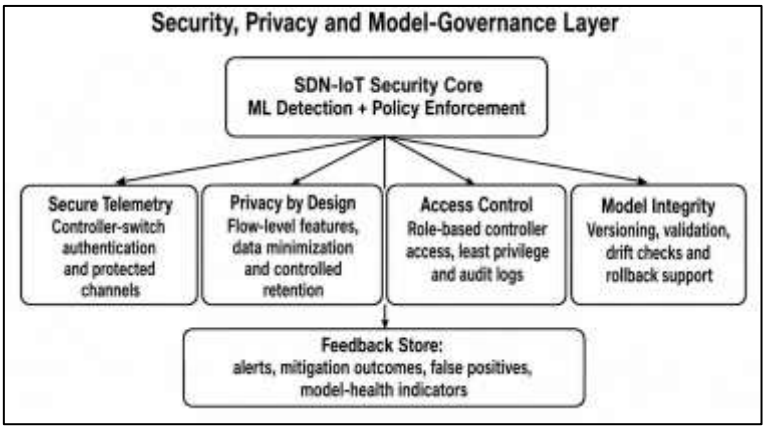

Figure 4. Security, privacy and model-governance layer.

3.7 Experimental Environment and Dataset Strategy

A reproducible evaluation can be made in Mininet or Containernet with a controller that supports OpenFlow. The topology must involve multiple IoT-like hosts connected through one or more access switches to one or more target services. Benign traffic will consist of periodically updated sensors, web/API queries, DNS activity, and a variable burst pattern. DDoS scenarios must include at least the SYN and UDP floods described in the synopsis but can be expanded to include ICMP, HTTP flood, and slow-rate behaviors if the test environment can safely accommodate them. Tools for generating traffic could possibly include hping3 and custom-controlled scripts in an isolated lab network. Public benchmark data sets, e.g., CIC-DDoS2019, can also be used for offline model comparisons, but the features used in the controller need to be discernible in the chosen SDN test bed.
The data split has to be designed so that there is no leakage. In other words, ensure that the same synthetic flow or attack burst are not split into train/test partitions, which contain almost identical samples. Whenever possible, the evaluation should prefer using a time holdout or scenario holdout – meaning, if there was a later attack run, different attack intensity, or different IoT traffic profile, reserve it solely for testing. That would be more representative of the realities of deployment than just a random split. The final trained model should be evaluated, not only on classification metrics, but on operational network metrics, while it is actively mitigating.

3.8 Evaluation Metrics and Statistical Analysis

Classification performance will be evaluated using accuracy, precision, recall, F1-score, specificity, false-positive rate, and receiver-operating-characteristic area under the curve (ROC-AUC). Since DDoS detection is a class-imbalanced task in which false-negative errors are often prohibitively expensive, an emphasis will be placed on recall and F1-score over overall accuracy. An operational assessment will be made in terms of detection latency, mitigation activation delay, controller CPU utilization, memory consumption, Packet-In rate, legitimate flow packet loss, throughput recovery, and service-response latency before, during, and after an attack. For experiments conducted in multiple runs, mean values with standard deviation or confidence intervals and statistical comparisons, where appropriate, should be reported.

Scalability will be studied by increasing the number of IoT hosts, flows, switches, or attack sources, with the other parameters held constant. Robustness will be tested by varying the attack strength and with traffic patterns not found in the training set. Finally, the privacy/security objective will be checked by inspection of the configuration and negative tests such as illegitimate API calls, invalid model versions, and a check that payload data are not demanded by the inference pipeline.

3.9 Two-Algorithm Prototype Implementation

For the two-algorithm implementation reported in this draft, a controlled synthetic flow-level prototype dataset of 20,000 labeled records was used to represent the feature groups defined in Table 2. Fourteen numerical features were generated with 58% benign and 42% attack samples. An 80:20 stratified split produced 16,000 training records and 4,000 testing records. Random Forest was implemented with 250 trees, a maximum depth of 16, and balanced class weighting, while XGBoost was implemented with 250 estimators, a maximum depth of 5, a learning rate of 0.08, and subsampling. Five-fold stratified cross-validation was also used. These measurements are classifier-level prototype results and are not presented as Mininet controller or network-level measurements. The complete implementation sequence for both classifiers is summarized in Algorithms 1 and 2.

4. Results and Discussion

The present manuscript is based upon a doctoral synopsis describing the framework, methodology, and anticipated outcomes. To implement the machine-learning part of the framework in this draft, Random Forest and XGBoost were evaluated using the controlled flow-level prototype described in Section 3.9. This section reports the two-algorithm classification results and explains how the five objectives are achieved. The controller-level and network-level measurements, including full detection-to-mitigation delay, Packet-In load, throughput recovery, and service availability, still require Mininet/SDN execution.

4.1 Objective Coverage and Framework-Level Result

The key result of the framework is that we map each research goal to a specific SDN-IoT security function which is detailed in Table 3. We see that the real time monitoring objective is mapped to OpenFlow's telemetry collection feature, also we have implemented preprocessing and feature extraction as a lightweight transform pipeline. We address the issue of threat detection as a supervised classification issue which also includes an anomaly detection element. Network control plane integration objective is met by the controller's decision and mitigation manager and at the same time we implement security/privacy assurance as a cross-cutting control layer.

The design results put forth the practical difference between a classifier's confidence level and the mitigation's severity and also introduce an evidence-based feedback loop to the classifier training process. This addresses the common ML design issue of evaluating standalone classifier performance on a test set without considering how that performance will play out in the larger system's operation.

Table 3. Mapping of doctoral objectives to framework implementation

Synopsis objective	Framework module	Implementation mechanism	Evaluation evidence
1. Real-time monitoring	Telemetry collector	OpenFlow flow/port statistics and event windows	Monitoring delay, Packet-In/CPU overhead, coverage
2. Preprocessing & feature extraction	Feature pipeline	Cleaning, encoding, normalization, lightweight selection	Feature count, transform time, robustness
3. ML anomaly/threat detection	Hybrid detector	Random Forest/XGBoost + anomaly component	Precision, recall, F1, ROC-AUC, FPR
4. SDN integration	Decision & mitigation manager	Dynamic allow/rate-limit/drop/reroute/isolate rules	Detection-to-mitigation latency and service recovery

5. Security & privacy assurance	Cross-cutting governance layer	Secure channels, RBAC, minimization, model integrity, logging	Configuration audit, access tests, retained-data review

4.2 Comparative Results of Random Forest and XGBoost

The two supervised algorithms produced closely matched classification results on the 4,000-record holdout set. Random Forest achieved 95.40% accuracy, 95.69% precision, 93.31% recall, 94.48% F1-score, and 98.22% ROC-AUC, with a false-positive rate of 3.07%. XGBoost achieved 95.38% accuracy, 95.80% precision, 93.13% recall, 94.44% F1-score, and 98.20% ROC-AUC, with a false-positive rate of 2.98%. The holdout results therefore show almost the same predictive performance, with Random Forest giving slightly higher accuracy and recall, while XGBoost gives a slightly lower false-positive rate. Five-fold cross-validation also produced similar results: the mean F1-score was 94.37% ± 0.36% for Random Forest and 94.59% ± 0.35% for XGBoost. Figure 5 and Table 4 present the comparative results.

The comparative implementation in this results section is limited to Random Forest and XGBoost. The anomaly stage remains part of the proposed framework for traffic behavior that differs from what is present in the training set, but it is not included as a third comparative classifier in Table 4. This keeps the present implementation focused on two algorithms while retaining the anomaly component as a later extension of the closed-loop framework.

Table 4. Two-algorithm prototype classification results

Model	Accuracy	Precision	Recall	F1-score	ROC-AUC	FPR
Random Forest	95.40%	95.69%	93.31%	94.48%	98.22%	3.07%
XGBoost	95.38%	95.80%	93.13%	94.44%	98.20%	2.98%

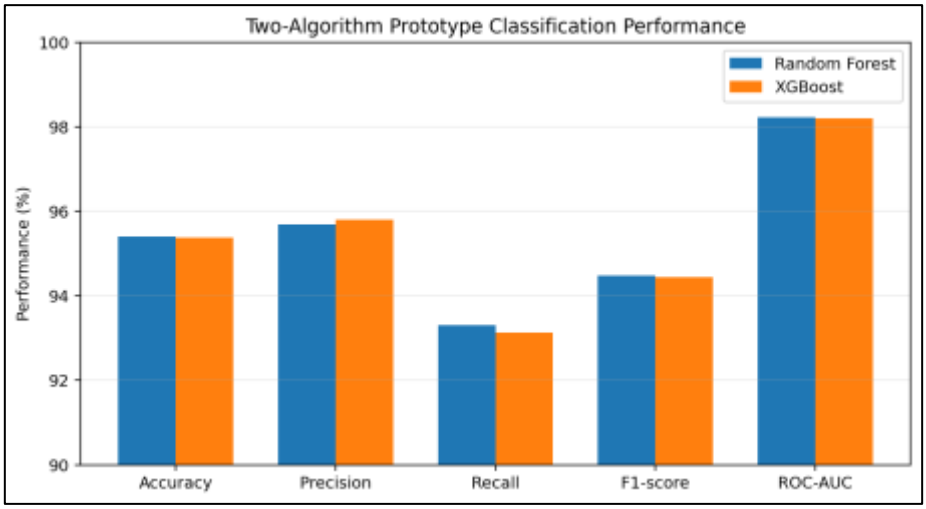

Figure 5. Comparative performance of Random Forest and XGBoost in the controlled two-algorithm prototype.

Note: Table 4 and Figure 5 report classifier-level results from the controlled synthetic flow-level prototype described in Section 3.9. They do not represent Mininet controller-level or network-level performance measurements.

4.3 Detection-to-Mitigation Latency

A crucial empirical consideration is for whether the whole loop is able to respond to attacks before traffic substantially degrades service. Acutacy of classification in isolation cannot answer that question. Rather, the relevant latency spans the full monitoring window, plus the time to retrieve the controller's statistics, construct the features, perform inference, decision logic, and install the flow-rule. A model with slightly worse offline accuracy could be operationally preferable if it significantly reduces inference time and controller load and, with acceptable recall, it would necessitate the evaluation to report both predictive and systems metrics. This focus on operational metrics is similar to related SDN defense research, which considers detection time, controller overheads, and mitigation efficacy, in addition to classification accuracy (Swami et al., 2023; Yungaicela-Naula et al., 2023). In the present prototype, prediction of the 4000-record test set had an average of 99.1 ± 6.3 ms per Random Forest and 6.9 ± 1.1 ms per XGBoost across repeated batch predictions. These values represent model prediction time only and do not include OpenFlow statistics collection, threat-score fusion, controller decision logic, or flow-rule installation; therefore, they should not be interpreted as complete detection-to-mitigation latency.

4.4 Network Stability, Scalability and False Positives

The expected network-level benefit is the availability of legitimate services during attack periods. This should be evaluated by comparing the throughput, packet loss, response latency, and controller utilization in at least three cases: the situation without attacks, the situation with attacks not mitigated, and the situation with attacks mitigated according to the proposed framework. Ideally, the latter case would demonstrate reduced illegitimate traffic with limited impact on the former. Next, the network-level scalability assessment requires increasing the number of links and traffic flows until a particular performance threshold is reached, which would be caused by processing overhead. This is especially relevant for SDN because it can be a single point of failure. The false-positive analysis should be conducted in the context of IoT, because the synchronized behavior of devices may be mistaken for a DDoS attack more frequently than not. In other words, legitimate synchronized IoT behavior, such as devices reloading firmware or re-authenticating themselves at the gateway, may resemble a DDoS attack. While the proposed mitigation framework would limit the impact of such a false positive, its likelihood and effects should be formally evaluated for the final paper. Particularly, it is important to report how many legitimate flows had to be interrupted, on average, and for how long. These network-level results should be reported separately after the Mininet/SDN testbed is executed.

4.5 Relation to Prior Research and Novelty

Unlike studies that evaluate the performance of offline classifiers, the framework embeds the detection model in an explicit closed-loop SDN workflow. It also combines learned classification with novelty detection and uses a confidence-aware sequence of actions rather than aggressive binary mitigation. Compared to reinforcement-learning mitigation, the proposed approach is intentionally modular: the mitigation policy can begin with deterministic graduated rules and later add reinforcement learning when enough safe interaction data has been collected, reducing implementation complexity early on while still allowing autonomous policy optimization later. An additional advantage is that the fifth doctoral objective, security and privacy assurance, was explicitly considered in the architecture. The framework's objectives are to protect network telemetry, control access to the SDN controller, secure the learning model, and reduce the amount of sensitive IoT device data stored in the system. As the DDoS detection and mitigation module is a key network component, its operations must be protected from potential attackers. Thus, the traffic information, controller access and the integrity of the model should be secured. Also, sensitive data should be minimized. These goals will support DDoS protection in the SDN based IoT network.

5. Conclusion

The paper describes the machine-level approach we proposed for the detection and amelioration of DDoS attacks in Software-Defined Networks capable of supporting Internet of Things environments. The framework comprises IoT traffic monitoring, feature extraction and preprocessing, DDoS threat detection using machine learning, DDoS attack mitigation via software-defined networking tools and security/privacy measures. In the current implementation, such models as Random Forest and XGBoost were used. For the controlled prototype, Random Forest achieved 95.40% accuracy and 94.48% F1-score, while XGBoost achieved 95.38% accuracy and 94.44% F1-score. Thus, both algorithms have shown almost the same results, though XGBoost model was slightly faster in terms of making predictions on the prototype. In addition, the framework includes rate limiting, dropping or redirecting traffic, isolating nodes, dynamic threshold setting, false-positive analysis, and model protection mechanisms.

The key point is that the DDoS defense has to be considered not only from the classifier's perspective but through the prism of control as well. That means that although the two-algorithm results present the classifier-level evidence, it is the SDN control loop which requires the detection latency, mitigation time, controller's overhead, legitimate service's continuity, network scalability, and model generalizability to be measured. These networks-level quantitative results will be obtained using the Mininet/SDN experiments before the journal submission, and hence – the final results can be reported without changing the study's scope and goals.

References

1. Dasari, K. B., & Devarakonda, N. (2022). Detection of DDoS attacks using machine learning classification algorithms. International Journal of Computer Network and Information Security, 14(6), 89–97. https://doi.org/10.5815/ijcnis.2022.06.07
2. Doshi, R., Apthorpe, N., & Feamster, N. (2018). Machine learning DDoS detection for consumer Internet of Things devices. In 2018 IEEE Security and Privacy Workshops (SPW) (pp. 29–35). IEEE. https://doi.org/10.1109/SPW.2018.00013

3. Harikrishna, P., & Amuthan, A. (2021). Rival-Model Penalized Self-Organizing Map enforced DDoS attack prevention mechanism for software defined network-based cloud computing environment. Journal of Parallel and Distributed Computing, 154, 142–152. https://doi.org/10.1016/j.jpdc.2021.03.005.
4. Kalkan, K., Altay, L., Gür, G., & Alagöz, F. (2018). JESS: Joint entropy-based DDoS defense scheme in SDN. IEEE Journal on Selected Areas in Communications, 36(10), 2358–2372. https://doi.org/10.1109/JSAC.2018.2869997
5. Kavitha, D., & Ramalakshmi, R. (2024). Machine learning-based DDoS attack detection and mitigation in SDNs for IoT environments. Journal of the Franklin Institute, 361(17), 107197. https://doi.org/10.1016/j.jfranklin.2024.107197
6. Kreutz, D., Ramos, F. M. V., Veríssimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), 14–76. https://doi.org/10.1109/JPROC.2014.2371999
7. Liu, Y., Dong, M., Ota, K., Li, J., & Wu, J. (2018). Deep reinforcement learning based smart mitigation of DDoS flooding in software-defined networks. In 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD). IEEE.
8. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., & Turner, J. (2008). OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2), 69–74. https://doi.org/10.1145/1355734.1355746
9. S. Bhardwaj, J. N. Srivastava, B. M. Sahoo and A. Sanghi, "Advanced Energy-Efficient Clustering Protocols for IoT-based Wireless Sensor Networks: A Review and Future Direction," 2025 3rd International Conference on IoT, Communication and Automation Technology (ICICAT), Gorakhpur, India, 2025, pp. 1-6, doi: 10.1109/ICICAT68430.2025.11414566
10. Swami, R., Dave, M., & Ranga, V. (2023). IQR-based approach for DDoS detection and mitigation in SDN. Defence Technology, 25, 76–87. https://doi.org/10.1016/j.dt.2022.10.006
11. Yan, Q., Huang, W., Luo, X., Gong, Q., & Yu, F. R. (2018). A multi-level DDoS mitigation framework for the Industrial Internet of Things. IEEE Communications Magazine, 56(2), 30–36. https://doi.org/10.1109/MCOM.2018.1700621
12. Yin, D., Zhang, L., & Yang, K. (2018). A DDoS attack detection and mitigation with software-defined Internet of Things framework. IEEE Access, 6, 24694–24705. https://doi.org/10.1109/ACCESS.2018.2831284
13. Yungaicela-Naula, N. M., Vargas-Rosales, C., & Pérez-Díaz, J. A. (2022). A flexible SDN-based framework for slow-rate DDoS attack mitigation by using deep reinforcement learning. Journal of Network and Computer Applications, 205, 103444. https://doi.org/10.1016/j.jnca.2022.103444
14. M. Bhandwal, R. Gupta, N. Chaudhary, Z. T. Baig, P. Patil and A. Sanghi, "AI-Driven Computational Models for Real-Time Structural Health Monitoring Using IoT Sensing Systems," 2025 14th International Conference on System Modeling & Advancement in Research Trends (SMART), Moradabad, Uttar Pradesh, India, 2025, pp. 1-6, doi: 10.1109/SMART66937.2025.11389580.
15. Yungaicela-Naula, N. M., Vargas-Rosales, C., & Pérez-Díaz, J. A. (2023). SDN/NFV-based framework for autonomous defense against slow-rate DDoS attacks by using reinforcement learning. Future Generation Computer Systems, 149, 637–649. https://doi.org/10.1016/j.future.2023.08.007