Journal of Innovations in Engineering Education
JIEE 2025, Vol. 8, Issue 1.

https://doi.org/10.3126/jiee.v8i1.86281

THAPATHALI CAMPUS
Institute of Engineering
Tribhuvan University

Reinforcement Learning based Malware mitigation in balanced Multicontroller Software Defined Networking enabled Internet of Things Networks

Utkarsha Shukla[a], Binod Sapkota[a,*] and Babu R. Dawadi[b]

[a]*Department of Electronics and Computer Engineering, Thapathali Campus, Tribhuvan University, Kathmandu, Nepal, 44600, Bagmati, Nepal*
[b]*Department of Electronics and Computer Engineering, Pulchowk Campus, Tribhuvan University, Lalitpur, Nepal, 44700, Bagmati, Nepal*

ARTICLE INFO

Article history:

Received 10 November 2025
Revised in 19 December 2025
Accepted 28 December 2025

Keywords:

Internet of Things
Q-Learning
REINFORCE
Reinforcement Learning
Software Defined Networking

Abstract

The rapid growth of Internet of Things (IoT) ecosystems has led to an increase in sophisticated malware threats that require advanced security solutions. This research proposes a novel reinforcement learning framework that combines Q-learning for dynamic load balancing and REINFORCE for malware classification and mitigation in software-defined networking (SDN)-enabled IoT networks and has been applied innovatively. Q-learning optimizes resource allocation between controllers, achieving a high load balance ratio (0.984), low imbalance (0.167), minimal packet loss (1.2%), low latency (0.0125 s), and low jitter (0.0079 s). The REINFORCE algorithm enables probabilistic malware classification with an overall accuracy of 0.93 and assesses device criticality to apply targeted countermeasures. All devices were mitigated through an epidemic modeling approach based on the susceptible-infected-recovered (SIR) model, which ensures comprehensive threat containment through controlled state transitions. Experimental results demonstrate the framework's ability to maintain network stability and responsiveness while neutralizing malware threats. Malware detection reliability was analyzed using average confidence scores and class-wise concept drift analysis, which enhances robustness by ensuring stable detection of static threats while adapting to evolving malware behaviors. The integration of reinforcement learning with SDN and epidemic modeling allows IoT systems to strengthen their cybersecurity defenses without compromising optimal network performance.

©JIEE Thapathali Campus, IOE, TU. All rights reserved

1. Introduction

Load balancing in SDN ensures efficient traffic distribution, improving resource utilization, scalability, and QoS through centralized, real-time control,etc. Reinforcement learning further enhances load balancing by optimizing traffic based on network feedback [1]. The convergence of SDN and IoT introduces significant cybersecurity challenges, particularly malware attacks, which traditional methods struggle to counter and therefore require machine learning-driven mitigation approaches [2].

Balanced multi-controller SDN architectures address these challenges by distributing control tasks across multiple controllers thereby, enhancing fault tolerance,reducing latency, and improving security. Controllers monitor distinct network segments while maintaining synchronization to ensure consistency and redundancy. This setup enables faster anomaly detection and coordinated threat responses. Multi-controller architectures also optimize bandwidth, enforce consistent security, and support critical IoT applications which improves network resilience and performance [3]. In SDN environments, controller-based management necessitates effective load balancing to prevent bottlenecks and sustain Quality of Service (QoS), particularly within dynamic and heterogeneous IoT environments where static strategies often prove inadequate [4]. Consequently,a balanced multi-controller framework enhances system

*Corresponding author:
replybinod@gmail.com (B. Sapkota)

efficiency, reduces latency, and optimizes resource utilization.

RL further strengthens this architecture by enabling real-time, adaptive threat mitigation. When integrated with multi-controller SDN, RL improves scalability and security through dynamic threat detection, control load redistribution, and intelligent decision-making. Controllers can collaboratively detect anomalies, isolate malicious flows, and share learned intelligence to support real-time reconfiguration, thereby increasing resilience and performance against evolving cyber threats [5]. In SDN, managing multiple controllers is vital for scalability, but load imbalances can lead to latency and failures. Static methods fail in dynamic environments, making RL essential for real-time load balancing.

SDN-enabled IoT networks face increasing malware threats, with attackers targeting centralized controllers via Distributed Denial-of-Service (DDoS) attacks and unauthorized access. Static security solutions can't adapt quickly enough to such dynamic threat environment. Balanced multi-controller setups reduce failure risks but introduce coordination challenges. Integrating machine learning enhances adaptive threat detection and improves the overall security and resilience of SDN-enabled IoT networks. This work proposes a novel unified framework that combines Q-learning for dynamic load balancing with the REINFORCE algorithm integrated with the SIR model for adaptive malware mitigation. Q-Learning improves resource utilization and reduces response time by efficiently distributing workloads across controllers, while the REINFORCE-SIR integration enables adaptive, real-time mitigation of malware mitigation.

REINFORCE enables real-time threat detection and response by adjusting security protocols based on learned patterns. Together, these approaches support autonomous and intelligent cybersecurity in evolving IoT environments[6][3].

The proposed framework aims to improve malware mitigation; however, it may face certain limitations, including a dependence on high-quality training data and scalability challenges in large IoT networks. The performance of RL algorithms can also vary with the complexity of the attack and the available computational resources [7]. A key limitation of this work is the reliance on synthetic or offline network traffic datasets, which may not fully capture the complexity and dynamics of real-world IoT malware behavior. Additionally, all experiments were conducted in a virtual environment, which may not entirely reflect the conditions and challenges of live network deployments. The major contributions of our paper are as follows:

- Introduces a real-time Q-Learning approach to distribute traffic across multiple SDN controllers, reducing bottlenecks and enhancing QoS.
- Proposes a reinforcement learning framework using the REINFORCE algorithm, integrated with the SIR (Susceptible-Infected-Recovered) model, to effectively classify and mitigate malware with improved adaptability and responsiveness.
- Implements a balanced multi-controller SDN design that enhances scalability, fault tolerance, and enables coordinated threat detection and isolation.
- Supports real-time, dynamic security policy updates to overcome the limitations of static defenses and ensure adaptability to evolving cybersecurity threats.

This paper is organized into six sections. Section I introduces the problem addressed in the study. Section II presents a comprehensive literature review of relevant works and identifies the related gap. Section III describes the methodology adopted for the implementation of the proposed approach. Section IV reports the experimental results, while Section V provides a discussion and in-depth analysis of the findings. Section VI concludes the work and suggests potential directions for future enhancements.

2. Literature Review

Abha Kumari et al.[8] tackled distributed SDN load balancing using a Constrained Markov Decision Process (CMDP)-based RL framework for dynamic switch migration. Their optimized online Q-learning approach, enhanced with Lagrangian multipliers, improves load imbalance by 5–16%, reduces migration costs by 40–50%, and reduces migration frequency by over 50%.

Singh et al. [9] presented an ML-based DDoS mitigation approach for SDN-enabled IoT, using feature selection to optimize classifiers. KNN with wrapper methods achieved 98.3% accuracy, outperforming SVM, ANN, and NB.

Aslam et al. [10] proposed an adaptive ML-based SDN-enabled DDoS detection and mitigation framework. This multilayered approach uses ensemble voting across SVM, NB, RF, k-NN, and logistic regression, outperforming LEDEM and the Content-Oriented Networking Architecture (CONA) in accuracy, precision, recall, and F1 score, while providing real-time detection and resource optimization.

Wang et al. [11] presented Secure Control and Data

Plane (SECOD), a resource-efficient SDN application-layer algorithm for DDoS detection and mitigation in IoT networks. Validated via simulation and testbed, it maintains over 98% normal traffic throughput with minimal loss. SECOD adapts to IoT traffic without extra hardware but struggles to differentiate bursty legitimate traffic from attacks and faces challenges with sophisticated threats such as IP spoofing.

Zolotukhin et al. [6] proposed an RL-based SDN architecture for real-time attack mitigation, leveraging DQN and PPO within a virtualized OpenFlow environment. The system effectively detects and blocks attacks like SSH brute force, DNS tunneling, and Slowloris DDoS, balancing security and service quality. Enhancement of adaptability and manual effort has been reduced in the work.

Kumar et al. [12] proposed an ML-based SDN load balancing framework using 12 key features and clustering over 3.5M+ packets. KMeans, optimized through Elbow and Silhouette methods, outperforms DBSCAN in reducing controller overload and improving resource utilization.

Mishra et al. [13] reviewed SDN-IoT security, spotlighting ML/DL for encryption, authentication, and intrusion detection in hybrid cloud–edge–fog environments. Emphasizing zero-trust and federated learning, they assess 20+ datasets with up to 99.8% accuracy.

Garba et al. [14] proposed an SDN-based DDoS detection framework for smart homes, integrating a 99.57% accurate Decision Tree with OpenFlow-based mitigation and SNORT IDS for controller protection. Real-world testbed validation shows strong accuracy, programmability, and hybrid defense.

Lam Dinh et al. [15] introduced a deep RL-based SDN load balancing framework using Proximal Policy Optimization/Deep Deterministic Policy Gradient (PPO/DDPG) with Control Barrier Functions for safe, QoS-optimized routing. Simulated through Flow and NS3, PPO-CBF outperforms traditional methods in delay, loss, and transfer learning. Strengths include safety assurance, GPU scalability, and dynamic adaptability.

Rostami et al. [16] reviewed 62 studies on QoS-aware load balancing in SDN-IoT, spanning centralized/distributed models and fog–cloud layers, focusing on delay and throughput. Mininet emerges as the dominant simulation tool, highlighting SDN's programmability and scalability.

Khozam et al. [17] proposed **QoSentry**, a DDQN-based DRL framework for DDoS mitigation in SDN, using adaptive countermeasures such as bandwidth throttling and flow redirection. It achieves low latency (<0.25 s), minimal delay (<0.05 s), and high benign throughput (~600 kbps–1.4 Mbps) while preserving legitimate traffic, adapting to varying attack patterns, and scaling to moderate topologies.

Despite notable progress in SDN-enabled IoT load balancing and security using ML and RL techniques, key research gaps persist. Most approaches demonstrate high accuracy and efficiency in simulations but lack real-world validation, limiting their scalability and robustness. Common challenges include computational complexity, hyperparameter sensitivity, reliance on static thresholds, limited attack diversity, and insufficient support for dynamic traffic and protocol variations. Furthermore, few studies address explainability, adversarial resilience, and deployment overhead, highlighting the need for adaptive, scalable, and interpretable solutions validated in real-world SDN-IoT environments.

3. Methodology

3.1. System Design Scenario

The proposed system design integrates a multi-controller SDN with IoT to achieve scalability, resilience, and secure communication. Leveraging MQTT, RabbitMQ, and ZeroMQ, the architecture efficiently handles heterogeneous traffic, while a Q-learning-based load balancing mechanism optimizes controller selection and resource utilization. Embedded security modules enable real-time detection and mitigation of malware, ensuring stable network performance as device density increases.

3.1.1. Network Topology

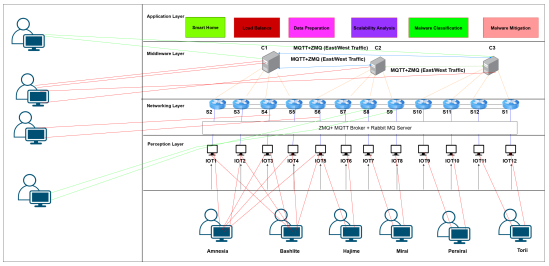

Figure 1: SDN Layered IOT Architecture.

SDN-based IoT architecture shown in Figure 1 adopts a multi-layered design to ensure scalability, security, and efficient communication. The **Perception Layer** consists of IoT devices (IOT1–IOT12) that use Message Queuing Telemetry Transport (MQTT) for lightweight publish/subscribe messaging. The **Networking Layer** includes switches (S1–S12) and supports MQTT, RabbitMQ, and ZeroMQ for reliable, low-latency communication. The **Middleware Layer** comprises controllers

(C1–C3) that coordinate data flow and device management. The **Application Layer** enables functionalities such as smart home automation, load balancing, malware detection, scalability analysis, and mitigation [18].

Security mechanisms address threats from malware families such as Amnesia, Bashlite, Hajime, Mirai, Persirai, and Torii. Controllers interact with MQTT, RabbitMQ, and ZeroMQ to exchange data and commands with IoT devices. MQTT ensures efficient controller-to-controller (east-west) communication, with each controller subscribing to shared topics and using QoS for reliable delivery[19]. ZeroMQ provides brokerless, low-latency peer-to-peer messaging for high-performance demands.

Communication lines connect controllers with brokers, devices with messaging systems, and switches with both, enabling redundancy and load balancing. DDoS simulations include protocol-based flooding of all malwares with triggering CPU/memory spikes.Traffic classification seperates normal from malicious traffic, while mitigation strategies include IP blocking, firmware updates, logging recovery actions .

3.1.2. Block Diagram for load balancing

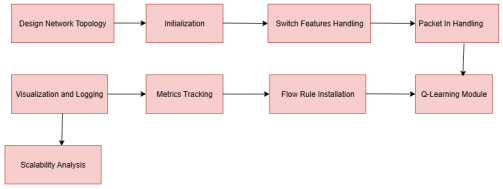

Figure 2: Block Diagram for Load Balancing

The block diagram for load balancing in Figure 2 begins by defining the network topology with switches, controllers, and traffic sources.The Q-table,learning parameters (epsilon,alpha,gamma), and performance metrics are then initialized. When a switch connects, default flow rules are installed. Upon packet arrival, Ethernet and IP details are extracted, and the Q-learning algorithm selects the optimal controller based on the current network state. The Q-table is updated using rewards,following an epsilon-greedy strategy to balance exploration and exploitation. Metrics such as latency, jitter, packet loss, throughput, migration cost, and load efficiency are monitored. Flow rules are adjusted according to Q-learning outcomes, and results are visualized along with the updated Q-table. Scalability analysis is performed to evaluate the performance remains stable despite an increase in number of devices and resource usage in the multi-controller SDN network.

3.2. Load Balancing in SDN using Q-learning

Q-learning is a model-free reinforcement learning algorithm that enables an agent to learn optimal actions in dynamic environments through trial and error. The agent interacts with an environment characterized by states s, actions a, and rewards r, learning a Q-value function $Q(s, a)$ that estimates the expected cumulative reward of taking action a in state s [20]. Action selection follows an ϵ-greedy policy: with probability ϵ the agent explores randomly;otherwise exploits the best-known action $a = \arg\max_a Q(s, a)$. The Q-values are updated iteratively through the Bellman equation in Equation 1:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \tag{1}$$

In Q-learning, $\alpha \in [0, 1]$ is the learning rate controlling Q-value updates, and $\gamma \in [0, 1]$ is the discount factor balancing immediate and future rewards. In distributed SDN-enabled IoT networks, this approach dynamically optimizes switch-to-controller assignments, improving efficiency and stability. The Q-function $Q(s, a)$ estimates the expected utility of action a in state s; after each action, the reward $R(s, a)$ guides learning, while $\max_{a'} Q(s', a')$ captures the best expected future reward [21]. Key performance metrics are described in Equations 2 - 5.

$$J = |L_i - L_{i-1}| \tag{2}$$

where L_i and L_{i-1} are consecutive packet latencies. Load distribution is evaluated using the sum of absolute latency deviations:

$$\sum_{i=1}^{N} \left| L_i - \frac{1}{N} \sum_{k=1}^{N} L_k \right| \tag{3}$$

load imbalance quantified as the standard deviation of controller loads:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (C_i - \mu)^2} \tag{4}$$

where C_i is the load on controller i and μ the average load, and load deviation for each controller:

$$\left| \text{Load}_i - \frac{1}{N} \sum_{j=1}^{N} \text{Load}_j \right| \tag{5}$$

Migration cost (MC) captures the overhead of workload transfers between controllers.

The effectiveness of Q-learning for load balancing in SDN-enabled IoT networks relies on the quality and diversity of training data. Incomplete or unrepresentative datasets omitt realistic traffic patterns or evolving malware behaviors—can produce policies that fail to generalize, leading to load imbalance and inefficient controller assignments. Ensuring comprehensive, realistic training data is therefore essential for adaptive and reliable network management.

3.2.1. Scalability Analysis

The scalability analysis was conducted in a multi-controller load-balanced environment. The proposed models characterize system behavior under IoT malware attacks as described by Equations 6 - 8:

Resource utilization and mitigation time:

$$\text{CPU/Memory} = \alpha_m\, r\, t + \beta_m, \quad T_{\text{mitigation}} = \gamma\, n \tag{6}$$

where α_m, β_m are malware-specific coefficients, r is the message rate, t is the attack duration, γ is the mitigation coefficient, and n is the number of compromised devices which is given in equation 6.

Network impact:

$$\text{Throughput} = R_0\, e^{-\lambda t}, \quad \text{Latency} = k\, r\, t \tag{7}$$

where R_0 is initial throughput, λ is the decay rate, and k is the latency scaling factor as described in equation 7.

Scalability and control-plane overhead:

$$\begin{aligned} \text{Resource} &= \eta N^2, \\ \text{Throughput} &= \min\left(r, R_{\text{sat}}\right), \\ \text{Overhead} &= \theta N \log S \end{aligned} \tag{8}$$

where N is the number of devices, R_{sat} is the throughput saturation limit, S is the number of switches, and η and θ are scaling coefficients presented in equation 8.

DDoS traffic volume is influenced by the attack rate, payload size, and attack duration,leading to bandwidth saturation under large-scale attacks.

Overall, the impact of attacks increases nonlinearly with network scale, duration, and intensity, highlighting the necessity for efficient mitigation and robust scalability mechanisms.

3.3. Data Preprocessing

In the preprocessing pipeline, missing numeric values are imputed using the mean of the observed entries, ensuring the statistical distribution of the feature remains consistent. For categorical attributes, the most frequent category (mode) is used for imputation as given in Equation 9:

$$\text{mode}(C) = \arg\max_k \text{count}(C = k) \tag{9}$$

This approach preserves dominant class information without introducing bias from rare categories. Categorical labels C are then converted into integer representations through label encoding, enabling compatibility with machine learning algorithms. The dataset is divided into predictor variables $\mathbf{X}$ and the target variable $\mathbf{y}$ for supervised learning tasks.

To retain only the most informative variables, feature selection is applied using Mutual Information (MI), which measures the dependency between a feature X_j and the target Y as given in Equation 10:

$$I(X_j; Y) = \sum_{x \in X_j} \sum_{y \in Y} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right) \tag{10}$$

where $p(x, y)$ denotes the joint probability distribution of X_j and Y, and $p(x)$ and $p(y)$ represent their marginal distributions. A higher MI score indicates a stronger relationship between the feature and the target, making the feature more relevant for prediction. After feature selection,the retained features are standardized using z-score normalization,as shown in Equation 11:

$$X_j' = \frac{X_j - \mu_j}{\sigma_j} \tag{11}$$

This transformation ensures zero mean and unit variance for each feature, thereby improving convergence during gradient-based optimization. The dataset is then split into training, validation, and test subsets using stratified sampling to maintain class proportions p_k across all splits. To address class imbalance, the Synthetic Minority Oversampling Technique (SMOTE) is employed, which generates synthetic samples for minority classes are described in Equations 7 and 8:

$$x_{\text{new}} = x_i + \delta(x_{nn} - x_i), \quad \delta \sim U(0, 1) \tag{12}$$

where x_i is a minority instance, x_{nn} is one of its nearest neighbors, and δ is a random interpolation factor that ensures variability in synthetic points. Additionally, class weights are incorporated into the loss function to penalize misclassification of minority classes proportionally:

$$w_k \propto \frac{1}{n_k + \epsilon}, \tag{13}$$

where n_k denotes the number of samples in class k, and ϵ is a small constant used to prevent division by zero. This ensures balanced learning and improves generalization for imbalanced datasets.

Table 1: Hyperparameter Optimization Results

Parameter	Search Space	Type	Optimal Value
Learning rate (η)	$[10^{-4}, 10^{-2}]$	Log-uniform	0.0005
Exploration (ϵ)	$[0.1, 1.0]$	Uniform	0.3
Reward scaling (r)	$[0.1, 0.99]$	Uniform	0.85
Hidden dims (**h**)	$\begin{cases} \mathcal{D}_1 \in [256, 1024] \\ \mathcal{D}_2 \in [128, 512] \\ \mathcal{D}_3 \in [64, 256] \end{cases}$	Integer	[768,384,192]
Dropout (p)	$[0.1, 0.5]$	Uniform	0.3

3.4. Model Training Using REINFORCE

The proposed REINFORCE-based policy network π_θ maps input features x to action probabilities through a softmax output, as shown in Equation 14:

$$\pi_\theta(a \mid x) = \text{softmax}\Big(W_4 \cdot \sigma\big(W_3 \cdot \sigma\big(W_2 \cdot \sigma(W_1 x + b_1) + b_2\big) + b_3\big)\Big) \tag{14}$$

where W_i, b_i denote the network weights and biases respectively, and σ represents the ReLU activation function. Batch normalization and dropout with a probability of $p = 0.5$ are applied to have generalization and training stability . At each time step t, actions are sampled as $a_t \sim \pi_\theta(\cdot|x_t)$, yielding rewards defined as $r_t = \mathbb{I}(a_t = y_t) \cdot \eta$ where $\mathbb{I}(\cdot)$ is the indicator function and $\eta = 0.9$ is the reward scaling factor. The training objective is the negative log-likelihood weighted by the received rewards, as given in Equation 15:

$$\mathcal{L}(\theta) = -\mathbb{E}_{a_t \sim \pi_\theta}[r_t \log \pi_\theta(a_t|x_t)]. \tag{15}$$

Model parameters are optimized using the AdamW with a learning rate $\alpha = 10^{-4}$ and weight decay coefficient $\lambda = 10^{-4}$. The parameter update rule is expressed in Equation 16:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta) + \lambda \|\theta\|_2, \tag{16}$$

An ϵ-greedy strategy is employed to balance exploration and exploitation, with ϵ initialized to 1.0 and gradually decayed during training.

3.5. Hyperparameter Tuning

Table 1 summarizes the results of the hyperparameter optimization process. Hyperparameters were optimized using Bayesian optimization implemented via `Optuna` [22] over 10 trials,with the objective of maximizing validation accuracy, as defined in Equation 17 and 18:

$$\theta^* = \arg\max_{\theta \in \Theta} \mathcal{A}_{\text{val}}(\theta), \tag{17}$$

where θ denotes the set of tunable hyperparameters. The resulting optimal configuration is given by:

$$\theta^* = \Big\{\eta = 5 \times 10^{-4}, \epsilon = 0.3, r = 0.85, \mathbf{h} = [768,\ 384,\ 192], p = 0.3\Big\} \tag{18}$$

This approach efficiently identified an optimal set of hyperparameters within a limited number of trials, demonstrating the effective of Bayesian optimization for reinforcement learning-based model tuning.

3.6. Data Postprocessing

The post-processing pipeline integrates classification and mitigation by first applying a confidence threshold to the predicted malware class probabilities **p**, accepting classifications only if $\max(\mathbf{p}_i) \geq \tau = 0.7$. Device-specific criticality scores $c_i \in [1, 5]$ are then used to determine the mitigation intensity. Devices are categorized into three states : **Susceptible (S)**, **Infected (I)**, or **Recovered (R)**. Malware propagates from I to S at rate β, while recovery occurs at rate γ, enabling containment. Recovered devices may revert to the susceptible state S if patching fails. This model supports outbreak severity estimation, urgency assessment, and optimal resource allocation, as expressed by the SIR formulation in Equation 19:

$$\frac{dI}{dt} = \beta S I - \gamma I \tag{19}$$

True positives ($\mathcal{D}_{\text{inf}}$) trigger mitigation actions , while false positives ($\mathcal{D}_{\text{safe}}$) are minimized using the false positive rate in Equation 20):

$$\text{FPR} = \frac{|\mathcal{D}_{\text{mit}} \cap \mathcal{D}_{\text{safe}}|}{|\mathcal{D}_{\text{safe}}|} \tag{20}$$

Mitigation intensity is adjusted according to device criticality, as defined in Equation 21:

$$\text{Intensity}_i = \begin{cases} \text{Aggressive}, & \text{if } c_i \geq 3 \\ \text{Standard}, & \text{otherwise} \end{cases} \quad (21)$$

ensuring stronger countermeasures for high-priority devices. The SIR model further simulates how mitigation actions,captured through the recovery rate γ, influence malware spread. Overall,mitigation optimization combines reinforcement learning based detection with SIR-guided response, adapting actions based on device criticality and detection confidence. High-criticality devices (e.g., security cameras) receive aggressive countermeasures, with actions triggered when the confidence threshold τ is exceeded. Lower thresholds (e.g., 0.7) initiate patching, while higher thresholds (e.g., 0.9) enforce device resets. Cost-aware decision-making balances security against downtime, patching, and maintenance costs.

3.6.1. Mitigation Metrics

The **Severity Score** (S) quantifies the threat level of IoT malware outbreaks by integrating three key factors:

1. **Infection Spread** – Measures initial penetration (analogous to the epidemiological basic reproduction number, R_0), weighted at 50% since early containment is critical.
2. **Persistence Time** – Reflects dwell-time risk, where prolonged infections cause more damage. This factor is weighted at 30%.
3. **Critical Baseline** – A fixed 20-point floor ensuring that any detected outbreak triggers an immediate response. The Severity Score is computed as given in equation 22:

$$S = \min\left(100,\ 50 \cdot \left(\frac{I_0}{N}\right) + 30 \cdot \left(\frac{T}{T_c}\right) + 20\right) \quad (22)$$

where I_0 is the number of initially infected devices, N is the total number of devices, T is the observed duration, and $T_c = 10$ is the critical duration threshold.

4. **Percentage of Final Susceptibles (PFS):**The proportion of devices remains uninfected at the termination of outbreak has been defined in Equation 23.

$$PFS = \left(\frac{S_{\text{final}}}{N}\right) \times 100\% \quad (23)$$

where S_{final} is the number of susceptible devices at the final time step.

5. **Average Infected Speed (AIS):**The mean rate of new infections per time step is computed in Equation 24.

$$AIS = \frac{1}{T-1} \sum_{t=1}^{T-1} |I_{t+1} - I_t| \quad (24)$$

where I_t denotes the number of infected devices at time t, and T is the the total number of simulation time steps.

6. **Average Recovery Speed (ARS):**The mean rate of device recovery per time step,has been defined in Equation 25.

$$ARS = \frac{1}{T-1} \sum_{t=1}^{T-1} (R_{t+1} - R_t) \quad (25)$$

where R_t denotes the number of recovered devices at time t.

PFS is expressed as a percentage (%), while AIS and ARS are measured in devices per step. A PFS value below 10% indicates effective containment, an ARS greater than AIS suggests successful mitigation, and peak AIS values reflect the virulence of the outbreak.

3.7. Malware Behavior Analysis

3.7.1. Average Confidence Score

The **average confidence score** quantifies the model's certainty in its correct classifications. It serves as an indicator of **reliability** (higher confidence implies more assured predictions), supports **threshold-based decision-making** (e.g., triggering actions only when confidence exceeds 70%), highlights **class difficulty** (the relative ease of identifying different malware types), and aids in **error analysis** (low confidence may indicate insufficient features or limited training data). For each class c, the average confidence score is computed in Equation 26.

$$\text{AvgConf}_c = \frac{1}{N_c} \sum_{i=1}^{N_c} P(y_i = c \mid x_i) \quad (26)$$

where N_c denotes the number of correctly classified samples belonging to class c, and $P(y_i = c \mid x_i)$ represents the predicted probability assigned to class c for sample x_i.

3.7.2. Class-Wise Concept Drift Analysis for Robust Malware Detection

Concept drift analysis monitors changes in model prediction confidence for each class over time. By comparing the average probabilities of correct predictions in recent and historical data windows, a drift score is computed, as shown in Equation 27:

$$\text{DriftScore}_i = \max\left(0, \mathbb{E}[P_{\text{new}}(y_i)] - \mathbb{E}[P_{\text{old}}(y_i)]\right) \quad (27)$$

where $P(y_i)$ denotes the predicted probability for class i computed over sliding windows. The concept drift analysis highlights the dependence on training data quality. It detects changes only within known malware classes, as DriftScore_i monitors existing labels (y_i) and cannot identify completely novel threats. Scores which a predefined threshold ($\tau = 0.15$) trigger corrective actions, such as model retraining. While this reactive approach allows some failures before correction, it maintains prediction stability for relatively static classes (e.g., Mirai: DriftScore $\approx$ 0) and enables adaptation to evolving threats (e.g., Bashlite: DriftScore ≈ 0.08). Hence, the balance between adaptability and generalization illustrates the inherent limitations of models trained on static datasets in dynamic, real-world environments.

3.8. Dataset Explanation

3.8.1. IoT Malware Types

The dataset consists of benign and malicious IoT software samples. Benign samples represent normal behavior [23].The malicious categories include Amnesia, which targets digital video recorders (DVRs) to launch botnet-based HTTP/UDP DDoS attacks [24]. Persirai infects IP cameras to conduct UDP flood DDoS attacks [25]. Torii is a stealthy malware that emphasizes on data exfiltration and persistence. Hajime is a botnet that attempts to secures infected devices by closing vulnerable ports [26]. Bashlite targets Linux-based IoT devices to execute large-scale DDoS attacks [27]. Mirai exploits weak or default credentials to launch massive DDoS attacks [28].

3.8.2. Contents of Prepared Dataset

The dataset captures detailed network traffic from a smart home IoT environment which features 100,000 entries with 43 attributes generated within a balanced-load, multi-controller SDN setup. It encompasses both normal and malicious behaviors,enabling robust analysis for anomaly detection and machine learning-based security solutions. Key components include device metadata (e.g., device name, state), traffic characteristics (e.g., message rate, bandwidth, protocol types), and network addressing (e.g., IPs, ports, session parameters). The dataset further incorporates threat intelligence (malware variants, DDoS), flow-level statistical metrics, and traffic volume measurements with temporal features. Each record is labeled with attack and malware categories . The dataset is enriched with derived features from Bot-IoT [29] and IoT-23 [30] datasets,thereby enhancing its diversity and generalizability. The Table 2

Table 2: Malware Dataset Frequency

Malware	Frequency
Benign	50000
Amnesia	8512
Torii	8472
Hajime	8299
Bashlite	8265
Mirai	8250
Persirai	8202

presents the frequency distribution of various malware types from the dataset. The majority of the samples are Benign (50,000), which represents non-malicious instances. The remaining 50,000 samples correspond to malicious instances, with Amnesia being the most frequent (8,512), followed closely by Torii (8,472), Hajime (8,299), Bashlite (8,265), Mirai (8,250), and Persirai (8,202) samples respectively.

4. Results and Discussion

This section presents the results of load balancing in a multi-controller SDN using Q-learning, along with the outcomes of malware classification and mitigation using REINFORCE. The section is accompanied by various types of analysis conducted during the process.

4.1. Results for load balancing in SDN using Q-learning

In the Q-learning based approach for SDN load balancing, an exploration rate $\epsilon = 1.0$ encourages extensive exploration during the learning phase, while a relatively low learning rate $\alpha = 0.1$ ensures stable updates to the Q-values. A high discount factor $\gamma = 0.85$ prioritizes long-term rewards, thereby supporting sustained and effective load balancing performance. The results presented in Table 3 show efficient traffic distribution across controllers, with final loads of 992, 1024, and 1031 packets and minimal deviations (23.67, 8.33, 15.33). A high load balance score (0.984466), low imbalance score (0.16715), and controlled migration cost (1680.30) confirm balanced allocation. System performance remains stable with low packet loss (0.012143), latency (0.012543,s), and jitter (0.007941,s), validating the effectiveness of the proposed approach.

The enhanced Q-learning balancer achieves adaptability through adaptive state representation, real-time optimization, and a moving-window throughput

Table 3: Load Balancing Results

Parameter	Value
Final Load on Controllers	[992, 1024, 1031]
Packet Loss Ratio	0.012143
Load Deviation (C-0)	23.67
Load Deviation (C-1)	8.33
Load Deviation (C-2)	15.33
Packets Sent	3,047
Packets Received	3,047
Packet Loss	37
Migration Cost	1,680.30
Throughput	2.46 pkt/s
Average Latency	0.0125 s
Average Jitter	0.0079 s
Load Balance Ratio	0.984
Load Imbalance Ratio	0.167

tracker. Asynchronous Q-table updates and selective flow changes reduce latency, while a composite reward function balances latency and load. Dynamic tuning (e.g., ϵ decay) ensures stability, combining fast convergence with responsiveness, at the cost of higher initial exploration and stateless flow handling during peak loads. The values in Table 4 are generated from Q-

Table 4: Q-table for balanced action selection

State	Action 0	Action 1	Action 2
0	57.5005	66.4984	66.4998
1	66.3696	57.3805	66.4045
2	66.4041	66.3964	57.3991

learning which demonstrates load balancing, with Q-values across various states and actions being nearly uniform, ensuring no single action dominates decision-making. For instance, in State 0 (57.5005, 66.4984, 66.4998), State 1 (66.3696, 57.3805, 66.4045), and State 2 (66.4041, 66.3964, 57.3991), the values are well-distributed, preventing any controller from becoming overloaded. The learned policy has converged a key indicator of Reinforcement Learning convergence where Q-values stabilize over time. This ensures optimal load allocation and minimal imbalance, confirming that the RL-based approach effectively maintains load equilibrium within the SDN network.

4.2. IOT device simulation

Figure 3 compares communication delays (in ms) of MQTT, RabbitMQ, and ZeroMQ across various IoT devices. **ZeroMQ** shows the lowest latency, making it the most efficient for time-sensitive tasks. **MQTT** follows closely with slightly higher but consistent performance.

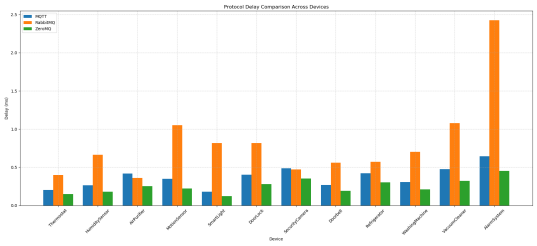

Figure 3: IOT Devices Message Delay Comparison

RabbitMQ, however, exhibits significantly higher delays—peaking at 2.45 ms on devices like the *AlarmSystem* and *MotionSensor*—over three times that of MQTT and ZeroMQ (both under 0.7 ms). These findings highlight **ZeroMQ** and **MQTT** as preferable for latency-critical IoT applications, while **RabbitMQ**'s higher overhead limits its suitability for real-time use.

4.3. Controller to Controller Communication

Table 5: Controller Communication Performance Comparison

Metric	ZeroMQ	MQTT
Throughput (KB/s)	89935.2	6567.15
Average Latency (ms)	1.84	2.51
Min Latency (ms)	0.17	0.43
Max Latency (ms)	11.05	9.41
Median Latency (ms)	0.66	0.998

In Table 5, ZeroMQ outperforms MQTT in controller-to-controller communication due to its decentralized peer-to-peer design, achieving 13.7× higher throughput (89,935 KB/s vs. 6,567 KB/s) and 32% lower latency (1.84 ms vs. 2.51 ms). Its direct messaging ensures efficient load distribution with minimal overhead and consistent performance under high traffic. In contrast, MQTT's broker-based model introduces congestion but provides centralized management, QoS, and device integration, making it ideal for heterogeneous IoT systems prioritizing reliability over speed. Thus, ZeroMQ suits high-speed, scalable coordination, while MQTT is better for structured, reliable communication.

4.4. Results for scalability analysis

The scalability metrics in table 6 show stable system performance, with consistent memory utilization at 33.6% across all test scenarios. Latency decreases from 4.53 millisecond to 1.47 millisecond as message rates increase, indicating efficient load handling. Throughput reaches 85.7 megabitpersecond with 20 devices, maintaining 85.2 megabitpersecond at maximum message rates, demonstrating strong scalability. Mitiga-

Table 6: Scalability Metrics Summary

Devices	Message Rate	CPU Usage	Memory Usage	Latency	Throughput	Mitigation Time
10	200	2.3	20.7	0.9846	83.6031	3.15e-04
20	200	1.2	20.7	1.3005	64.9143	1.96e-05
50	200	2.8	20.7	0.2001	74.1422	1.12e-05
100	200	1.2	20.7	1.4352	35.1569	1.79e-05
200	200	7.1	20.6	4.2276	45.7628	2.57e-05
12	50	1.3	20.6	1.6823	14.2490	1.50e-05
12	100	1.5	20.6	2.8919	50.4402	2.48e-05
12	200	1.0	20.6	2.9376	93.2723	1.14e-05
12	500	1.3	20.6	1.5442	45.1051	2.74e-05
12	1000	1.3	20.6	2.9066	69.7955	1.00e-05

tion times are extremely fast, on the order of 10^{-5} seconds, showcasing the effectiveness of the security measures.

4.5. Results for malware classification using REINFORCE

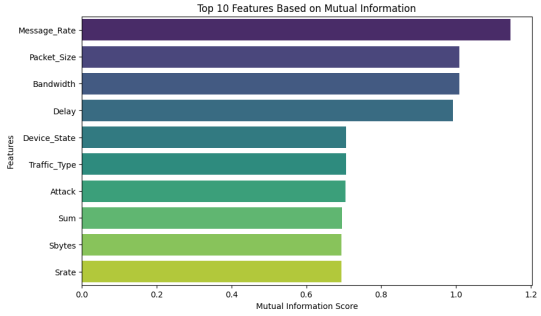

Figure 4: Mutual Information (Feature Engineering)

Figure 4 illustrates the top 10 features selected based on their MI scores, which quantify the dependency between each feature and the target malware class. Features such as Message_Rate, Packet_Size, and Bandwidth exhibit the highest MI scores, indicating their strong relevance for classification. The x-axis represents MI values (ranging from 0.0 to 1.2), while the y-axis ranks features by importance. These selected features, determined through Mutual Information-based feature selection, enhance the REINFORCE algorithm by improving learning efficiency, reducing redundancy, and contributing to the malware classification of SDN-enabled IoT networks. The original training dataset exhibits significant class imbalance in Table 7 with Class 2 dominating at 35,000 samples (50% of the data), while the remaining classes (0, 1, 3, 4, 5, 6) have considerably fewer samples, ranging from 5,741 to 5,958 (approximately 8.2% to 8.5% each). Such imbalance can bias the model towards the majority class during training, potentially degrading performance on minority classes. After applying SMOTE, all classes are balanced with 35,000 samples each in Table 8 representing 14.3% per class. SMOTE synthetically generates minority class samples to match the original majority class size, resulting in an evenly distributed dataset that enables the model to learn equally from all classes and improves its generalization and accuracy on minority classes. The plot in Figure 5 shows training and validation losses decreasing from 0.30 to 0.05 over 200 epochs, indicating successful convergence. The parallel decline without divergence suggests effective learning, proper model design, and strong generalization without overfitting. In figure 6 the reward curve shows a decline from 9300 to 8700 over 200 epochs, indicating initial performance gains

Table 7: Class Distribution Before SMOTE

Class	Samples	Percentage (%)
0	5958	8.5
1	5786	8.3
2	35000	50.0
3	5809	8.3
4	5775	8.2
5	5741	8.2
6	5931	8.5

Table 8: Class Distribution After SMOTE

Class	Samples	Percentage (%)
0	35000	14.3
1	35000	14.3
2	35000	14.3
3	35000	14.3
4	35000	14.3
5	35000	14.3
6	35000	14.3

Table 9: Comparison of Best and Worst Model Performances

Metric	Best Model (Trial 1)	Worst Model (Trial 7)
Validation Accuracy	0.9295	0.9224
Learning Rate (lr)	0.0007	0.0013
Epsilon Reward Rate	0.8661	0.4743
Weight Decay	2.1302	4.4825
Dropout Rate	0.2009	0.1425

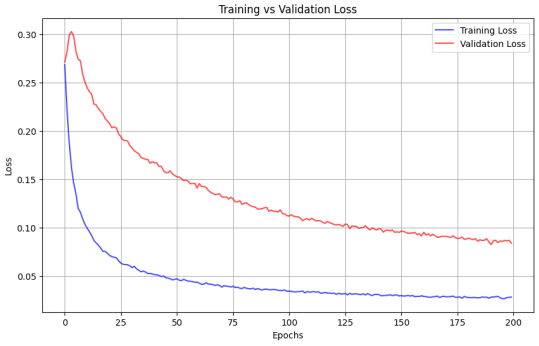

Figure 5: Training Validation loss Plot

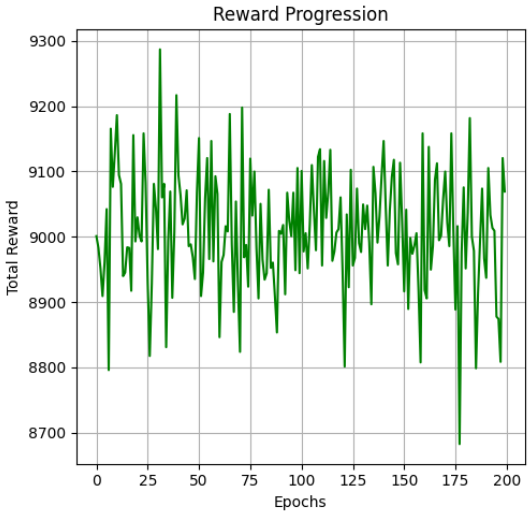

Figure 6: Reward Progression Plot

followed by stabilization. This reflects effective early exploration by the REINFORCE algorithm, with convergence to a stable policy over time. The downward trend highlights the exploration-exploitation trade-off, as reduced exploration limited high-reward discoveries. Overall, the curve indicates successful learning. In Table 9 the best model (Trial 1) achieved **0.71**% higher validation accuracy than the worst (Trial 7) (**92.95**% vs. **92.24**%) due to balanced hyperparameters: a moderate learning rate (**0.0007**), better exploration-exploitation trade-off (ϵ = **0.6963**), and higher reward rate (**0.8661**). Its architecture (**270-499-131**) with moderate regularization (weight decay = **2.1302**, dropout = **0.2009**) supported stable learning, unlike Trial 7's larger layers and high weight decay (**4.4825**), which likely hindered performance. These results highlight the impact of tuning exploration, rewards, and model architecture for optimal convergence. In Figure 7,the confusion matrix reveals

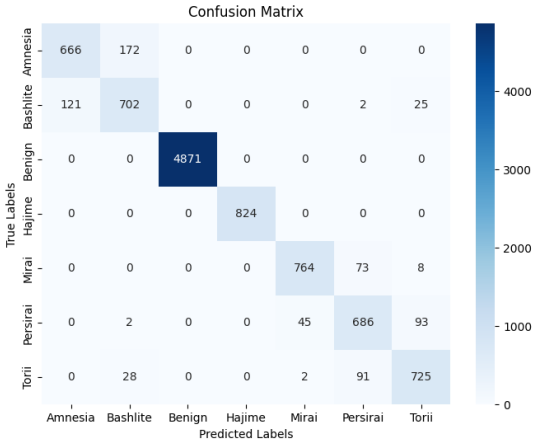

Figure 7: Confusion Matrix

perfect classification for Benign (4,821) and Bashlite (824), with Mirai and Bashlite achieving perfect precision. However, Hajime is frequently misclassified as Bashlite (666 errors), and Persirai performs poorly with only 2 correct out of 47. Torii also shows confusion with Hajime and Persirai. While performance is strong for distinct classes, improvements are needed to better distinguish Persirai and Torii, possibly through enhanced features or model architecture. The classification report in Table 10 shows strong overall performance with 93% accuracy. Classes 2 and 3 achieve perfect scores (F1 = 1.00), while class 4 performs well (F1 = 0.92). Classes 0, 1, 5, and 6 have slightly lower F1-scores (0.80–0.85), indicating minor classification challenges. Macro and weighted averages are high (0.89 and 0.93), with the latter boosted by the large support of well-classified classes like class 2 (support = 4871). The model maintains a good balance between precision and recall across all cat-

Table 10: Malware Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.85	0.79	0.82	838
1	0.78	0.83	0.80	850
2	1.00	1.00	1.00	4871
3	1.00	1.00	1.00	824
4	0.94	0.90	0.92	845
5	0.81	0.83	0.82	826
6	0.85	0.86	0.85	846
Accuracy		0.93		
Macro Avg	0.89	0.89	0.89	9900
Weighted Avg	0.93	0.93	0.93	9900

egories. The ROC-AUC curve plot in Figure 8 evaluates

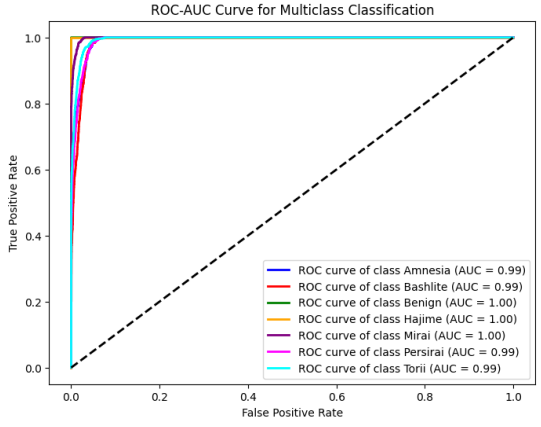

Figure 8: ROC-AUC plot

a multiclass classification model's performance across seven categories: Amnesia, Bashlite, Benign, Hajime, Mirai, Persirai, and Torii. Each curve illustrates the balance between true positive and false positive rates, with AUC scores indicating near-perfect or perfect classification—Amnesia (0.99), Bashlite (0.99), Benign (1.00), Hajime (1.00), Mirai (1.00), Persirai (0.99), and Torii (0.99). The model excels with perfect discrimination (AUC = 1.00) for Benign, Hajime, and Mirai, while other classes also achieve exceptional scores (≥ 0.99), demonstrating strong reliability in distinguishing all categories with minimal errors.

4.6. Results for malware mitigation using REINFORCE

The plot in Figure 9 shows the initial state of IoT devices, all in a **susceptible** (blue) condition, indicating no infections. The y-axis (0–10) likely denotes device IDs or susceptibility, and the flat distribution confirms a uniform, pre-infection baseline. This serves as the

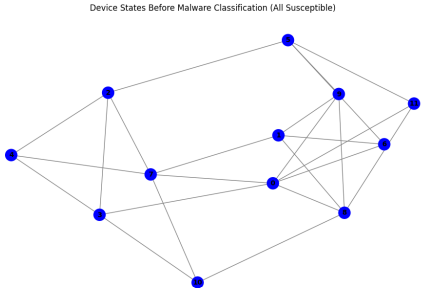

Figure 9: Susceptible Devices

starting point for SIR-based simulations before malware classification and mitigation begin. The plot in Figure

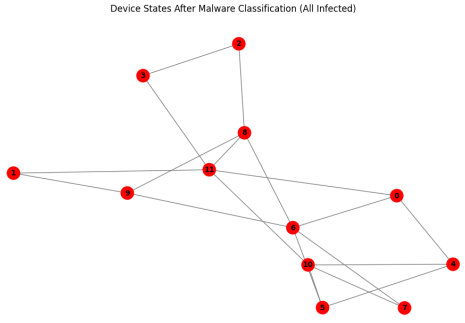

Figure 10: Infected Devices

10 illustrates a post-classification scenario where all IoT devices are marked as **infected** (red), indicating full network compromise. The y-axis (1–7) likely reflects device IDs or infection severity. This uniform infection state represents the worst-case phase in the SIR model, highlighting the urgency for mitigation before recovery is driven by REINFORCE-based policy decisions. The plot in Figure 11 shows a deterministic Suspectible-Infected-Recovered (SIR) model over five time steps,

Table 11: Devices, Malware, and Mitigation Strategies

Malware	Device(s)	Mitigation Strategy
Mirai	HumiditySensor, AirPurifier, MotionSensor, DoorLock, Refrigerator	Reset Devices
Bashlite	Doorbell	Block Malicious Traffic
Hajime	Thermostat, HumiditySensor, AirPurifier, MotionSensor, SmartLight, DoorLock, SecurityCamera, Doorbell, VacuumCleaner, AlarmSystem	Update Firmware
Persirai	SecurityCamera	Isolate Affected Devices
Torii	Refrigerator, VacuumCleaner, AlarmSystem	Block C&C Servers
Amnesia	Thermostat, AlarmSystem	Patch Vulnerabilities

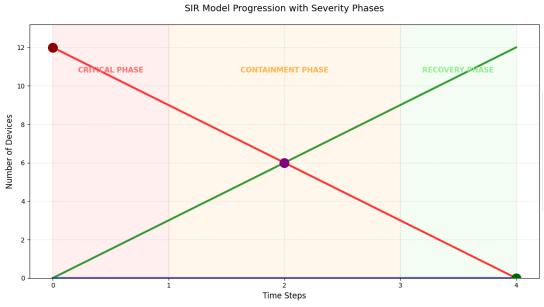

Figure 11: SIR Analysis Plot

where all 12 IoT devices begin infected ($I = 12$, $R = 0$) and recover linearly to a fully recovered state ($I = 0$, $R = 12$). With no susceptible devices ($S = 0$), the closed-system model assumes no new infections. Using $\beta = 0.3$ and $\gamma = 0.1$, the linear recovery reflects ideal mitigation and homogeneous device behavior. Though simplified and lacking stochastic effects, it illustrates optimal mitigation performance, offering a theoretical benchmark for real-world strategies influenced by device diversity and detection imperfections. The mitigation metrics presented in Table **??** demonstrate complete outbreak containment, with 0.00% of devices remaining susceptible and all 12 infected devices transitioning to the recovered state within five steps. The balanced rates of infection and recovery (–2.40 vs. 2.40 devices per step) highlight the effectiveness of the REINFORCE algorithm's countermeasures. A high severity score of 85.0 out of 100 further confirms the robustness of the mitigation strategy, validating the use of dynamic thresholding, risk-based prioritization, and rapid isolation and patching to achieve efficient containment. The mitigation strategies displayed in Table 11 list devices such as the *HumiditySensor*, *AirPurifier*, and *Thermostat*, along with the types of malware affecting them—including Mirai, Bashlite, Hajime, Persirai, Torii, and Amnesia—as well as their respective mitigation strategies. Mirai infections require device resets, Bashlite calls for traffic blocking, and Hajime necessitates firmware updates. Devices infected with Persirai should be isolated, while Torii requires blocking command and control (C&C) servers. Amnesia is mitigated by patching known vulnerabilities. This structured approach ensures that each threat is addressed with a targeted and effective security measure. The plot in Figure 12 shows the final

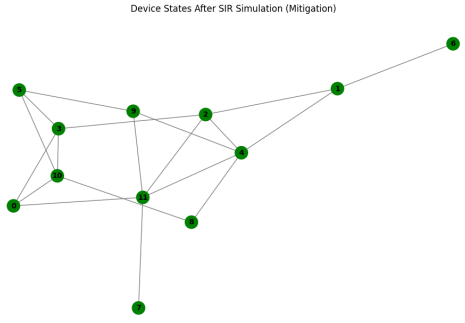

Figure 12: Mitigation Graph

device states after the SIR simulation with mitigation, where all nodes are marked as recovered (green), indicating 100% successful remediation. The values (1–10) likely represent normalized security scores or device IDs. This outcome validates the REINFORCE algorithm's mitigation strategies: high-criticality devices received aggressive responses (e.g., isolation, firmware updates), while standard measures were applied elsewhere. The uniform green color reflects the effectiveness of the dynamic threshold-based classification and criticality-aware response system.

4.7. Results for malware behavior analysis

Figure 13 tracks concept drift scores for malware classes over 200 training iterations, with *Bashlite* showing the highest drift (~ 0.08) but none crossing the 0.15 thresh-

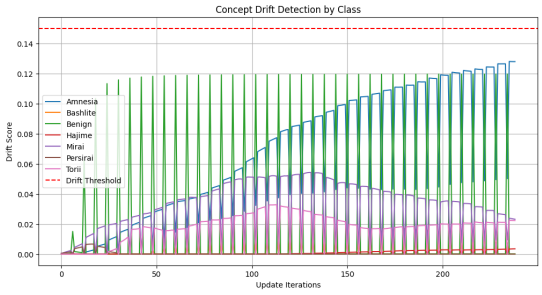

Figure 13: Concept Drift Analysis

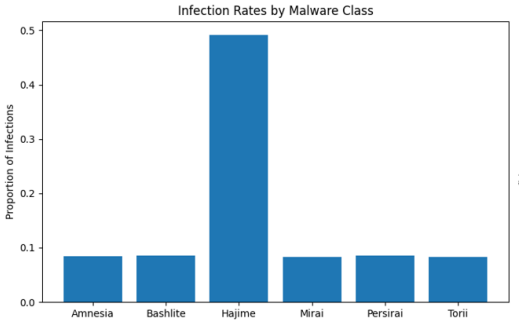

Figure 14: Infection Rate

old. Stable classes (e.g., *Mirai*, *Torii*) indicate consistent model performance, while moderate drift in others (e.g., *Amnesia*, *Hajime*) suggests evolving attack patterns. The system detects shifts but requires no intervention, demonstrating adaptability while highlighting class-specific generalization limits. Table 12 displays

Table 12: Malware Confidence Scores

Malware	Confidence Score
Amnesia	0.88
Torii	0.85
Hajime	1.00
Bashlite	0.82
Mirai	0.99
Persirai	0.95

the average confidence scores of a REINFORCE-based malware detection model across different IoT threats. Hajime (1.0) and Mirai (0.99) achieve near-perfect confidence due to their distinct patterns and specialized neural network heads, while Bashlite (0.82) shows relatively lower confidence, likely due to overlapping features or fewer training samples. These scores, derived from the model's softmax probabilities during evaluation, directly influence mitigation strategies. High-confidence predictions trigger immediate actions such as isolation or firmware updates, whereas lower-confidence detections may require additional verification. Figure 14 reflects the relative prevalence of each malware class in the dataset, with *Mirai* and *Hajime* showing the highest infection rates, aligning with their near-perfect confidence scores (0.99 and 1.0) in the model's predictions. This correlation suggests that the REINFORCE-based model performs exceptionally well for frequently encountered threats, triggering aggressive mitigation strategies such as network isolation or firmware updates. In contrast, *Bashlite*'s moderate infection rate but lower confidence score (0.82) indicates potential challenges in distinguishing its patterns, highlighting opportunities for targeted improvements in feature engineering or adversarial training to enhance detection accuracy for less prevalent or more subtle malware variants.

5. Conclusion and Future Work

This work proposes an intelligent SDN-IoT security framework that combines Q-learning-based load balancing with RL-driven malware detection to counter DDoS attacks. The Q-learning algorithm optimizes traffic distribution for high throughput, low latency, and QoS, while integrated threat detection enables real-time mitigation. The REINFORCE-based policy network detects malware strains and applies mitigation using same algorithm using SIR approach. The scalable, distributed architecture ensures robustness and prevents single points of failure, optimized for resource-constrained IoT environments. Average confidence scoring and drift analysis together improves interpretability and resilience in malware detection models, supporting adaptive retraining and balancing generalization with adaptability to emerging threats.

Future work involves integrating advanced Deep Reinforcement Learning (DRL) techniques to enhance controller collaboration, with performance compared to Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Fuzzy Logic. The framework will be extended to detect Man-in-the-Middle (MITM) attacks and ransomware, employing mitigation strategies like dynamic key rotation and meta-RL. Adversarial RL using Generative Adversarial Network (GAN) attacks will be explored to improve robustness. Evaluation using above methods will benchmark detection accuracy, false positives, and response time against signature and anomaly-based methods.

6. Acknowledgement

The authors declare that they have not received any financial support for this work.

References

[1] Alhilali A H, Montazerolghaem A. Artificial intelligence based load balancing in sdn: A comprehensive survey[J/OL]. arXiv preprint arXiv:2308.02149, 2023. https://doi.org/10.48550/arXiv.2308.02149.

[2] Negera W G, Schwenker F, Debelee T G, et al. Review of botnet attack detection in sdn-enabled iot using machine learning[J/OL]. Sensors, 2022, 22(24): 9837. https://pmc.ncbi.nlm.nih.gov/articles/PMC9787631/. DOI: 10.3390/s22249837.

[3] Gueriani A, Kheddar H, Mazari A C. Deep reinforcement learning for intrusion detection in iot: A survey[J/OL]. arXiv preprint arXiv:2405.20038v1, 2024. https://arxiv.org/abs/2405.20038v1.

[4] Prabakaran S, Ramar R. Software defined network: Load balancing algorithm design and analysis[J/OL]. International Arab Journal of Information Technology, 2021, 18(3): 123-130. https://doi.org/10.34028/iajit/18/3/7.

[5] Khorrami M, Azizi M A, Bakhshi A A. Deep-learning-based approach for iot attack and malware detection[J/OL]. Electronics, 2024, 14(18): 8505. https://www.mdpi.com/2076-3417/14/18/8505. DOI: 10.3390/electronics14188505.

[6] Zolotukhin M, Kumar S, Hamalainen T D. Reinforcement learning for attack mitigation in sdn-enabled networks[C/OL]// Proceedings of the IEEE/IFIP Network and Service Management Conference (NetSoft). 2020. https://doi.org/10.1109/NetSoft48620.2020.9165383.

[7] Saha R, Chakraborty P, Kumar S, et al. Multi-agent reinforcement learning framework in sdn-iot for transient load detection and prevention[J/OL]. Mathematics, 2020, 9(3): 44. https://www.mdpi.com/2227-7080/9/3/44. DOI: 10.3390/math9030044.

[8] Kumari A, Roy A, Sairam A S. Optimizing sdn controller load balancing using online reinforcement learning[J/OL]. IEEE Access, 2024, 12: 131591-131604. https://doi.org/10.1109/ACCESS.2024.3459952.

[9] Singh K, Kumar P D B, Sunil Kumar V P S, et al. Mitigation of cyber attacks in sdn-based iot systems using machine learning techniques[J/OL]. International Journal of Intelligent Systems and Applications in Engineering (IJISAE), 2023 (ISSN:2147-67992147): 482-492. https://ijisae.org/index.php/IJISAE/article/view/4149/2790.

[10] Aslam M, Ye D, Tariq A, et al. Adaptive machine learning-based distributed denial-of-service attacks detection and mitigation system for sdn-enabled iot[J/OL]. Sensors, 2022, 22(7): 2697. https://www.mdpi.com/1424-8220/22/7/2697.

[11] Wang S, Gomez K, Sithamparanathan K, et al. Mitigating ddos attacks in sdn-based iot networks leveraging secure control and data plane algorithm[J/OL]. Applied Sciences, 2021, 11 (3): 929. https://www.mdpi.com/2076-3417/11/3/929. DOI: 10.3390/app11030929.

[12] Kumar A, Anand D. Load balancing for software defined network using machine learning[J/OL]. Turkish Journal of Computer and Mathematics Education, 2021, 12(2): 527-535. https://turcomat.org/index.php/turkbilmat/article/view/1582.

[13] Mishra S R, Shanmugam B, Yeo K C, et al. SDN-Enabled IoT Security Frameworks—A Review of Existing Challenges[J/OL]. Technologies, 2025, 13(3): 121. https://www.mdpi.com/2227-7080/13/3/121. DOI: 10.3390/technologies13030121.

[14] Garba U H, Toosi A N, Pasha M F, et al. SDN-based Detection and Mitigation of DDoS Attacks on Smart Homes[J/OL]. Computer Communications, 2024, 221: 29-41. https://doi.org/10.1016/j.comcom.2024.04.001.

[15] Dinh L, Quang P T A, Leguay J. Safe load balancing in software-defined-networking[J/OL]. Computer Communications, 2024. https://arxiv.org/abs/2410.16846.

[16] Rostami M, Goli-Bidgoli S. An overview of qos-aware load balancing techniques in sdn-based iot networks[J/OL]. Journal of Cloud Computing: Advances, Systems and Applications, 2024, 13(89). https://doi.org/10.1186/s13677-024-00651-7.

[17] Khozam S, Blanc G, Tixeuil S, et al. Qosentry: A reinforcement learning framework for qos-preserving ddos mitigation in software-defined networks[J/OL]. Journal of Network and Systems Management, 2025, 33(97): 1-31. DOI: 10.1007/s10922-025-09971-8.

[18] Jmal R, Ghabri W, Guesmi R, et al. Distributed blockchain-sdn secure iot system based on ann to mitigate ddos attacks[J/OL]. Appl. Sci., 2023, 13(8): 4953. DOI: 10.3390/app13084953.

[19] Tamri R, Rakrak S. Sdn-mqtt for a smart home[J/OL]. E3S Web of Conferences, 2021, 229: 01031. https://doi.org/10.1051/e3sconf/202122901031.

[20] Simplilearn. What is q-learning? a comprehensive guide[EB/OL]. 2024. https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-q-learning.

[21] GeeksforGeeks. Q-learning in python[EB/OL]. 2024. https://www.geeksforgeeks.org/q-learning-in-python/.

[22] Akiba T, Sano S, Yanase T, et al. Optuna: A next-generation hyperparameter optimization framework[C]// Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2019.

[23] Asam M, Khan S H, Akbar A, et al. Iot malware detection architecture using a novel channel boosted and squeezed cnn[J/OL]. Sci Rep, 2022, 12(15498). https://pmc.ncbi.nlm.nih.gov/articles/PMC9477830/. DOI: 10.1038/s41598-022-18936-9.

[24] News I T. Amnesia: Yet another iot botnet targets global dvrs[J/OL]. IoT Tech News, 2022. https://iottechnews.com/news/amnesia-yet-another-iot-botnet-targets-global-dvrs/.

[25] MDL. The persirai botnet[J/OL]. Westoahu Cybersecurity, 2017. https://westoahu.hawaii.edu/cyber/regional/gce-us-news/the-persirai-botnet/.

[26] Radware. The rise of the botnets: Mirai & hajime[J/OL]. Radware, 2017. https://www.radware.com/security/ddos-threats-attacks/ddos-attack-types/rise-of-botnets-mirai-hajime/.

[27] Micro T. Bashlite iot malware updated with mining and backdoor commands, targets wemo devices[J/OL]. Trend Micro, 2019. https://www.trendmicro.com/en_in/research/19/d/bashlite-iot-malware-updated-with-mining-and-backdoor-commands-targets-wemo-devices.html.

[28] Networks A. Five most famous ddos attacks and then some[Z]. 2024.

[29] Venkateswaran V. Bot-iot: A dataset for botnet attacks in iot networks[EB/OL]. 2020. https://www.kaggle.com/datasets/vigneshvenkateswaran/bot-iot.

[30] Anwar E A. Iot-23 preprocessed data[EB/OL]. 2023. https://www.kaggle.com/datasets/engraqeel/iot23preprocesseddata.