Check for updates

Security and Communication Networks

WILEY

RESEARCH ARTICLE OPEN ACCESS

Risk-Bounded Temporal Graph Decision Learning for DDoS Mitigation in Software-Defined Internet of Things Networks

Pranto Halder

Computer Science and Engineering, Faculty of Science and Engineering, Teesta University, Rangpur 5404, Bangladesh

Correspondence: Pranto Halder (prantohalder117@gmail.com)

Received: 23 April 2026 | **Revised:** 29 August 2026 | **Accepted:** 15 September 2026

Academic Editor: Pramita Mishra

Keywords: audit logging | controller-visible telemetry | distributed denial-of-service (DDoS) mitigation | graph neural networks | Internet of Things (IoT) | risk-bounded action selection | shadow-mode policy evaluation | software-defined networking (SDN) | temporal graph learning

ABSTRACT

Distributed denial-of-service (DDoS) attacks remain a major availability threat in software-defined Internet of Things (SDN-IoT) networks because the controller must both recognize abnormal traffic and choose a mitigation action without causing unnecessary service disruption. This paper presents a risk-bounded temporal graph decision framework that converts controller-visible telemetry into a history-aware graph state, scores four candidate actions (ALLOW, RATE_LIMIT, BLOCK, and ISOLATE_SUBNET), filters candidates through predicted benign-loss, controller-cost, and risk constraints, and records decision provenance in a permissioned hash-linked audit log. The archived temporal-transition dataset contains 1282 one-step transitions with an eight-window history and a contiguous source-time split of 910/196/176 train/validation/test samples. The original action-selection replay contains 130 decision steps: the temporal model with the risk selector reaches 0.5077 accuracy and 0.6667 F1-score, compared with 0.9385 accuracy and 0.9344 F1-score for the strongest observable baselines, and it collapses toward RATE_LIMIT in 128/130 decisions. These negative results are analyzed rather than hidden. Inspection of the archived transition pipeline also shows incomplete factual support for the four-action space because the source stress logs record operational status rather than an explicit four-action treatment label. Consequently, the learned temporal policy is retained as shadow-mode feasibility evidence rather than an autonomous controller. Supporting controlled validation adds a timestamped temporal detector split, topology-scale tests, five repeated live/stress runs at a common threshold, benign flash-crowd testing, and audit-overhead measurements. Zero unsafe-action and controller-overload events in the 130-step replay are, therefore, reported only as empirical constraint satisfaction under the evaluated conditions, not as a safety guarantee. The contribution is an auditable, nonleaky, constraint-filtered research architecture and a transparent diagnosis of the remaining adaptive-policy learning problem.

1 | Introduction

The story of modern IoT networking is also a story of concentration. As devices multiply, traffic diversity increases, and service dependencies become more intricate, network operators increasingly rely on software-defined networking (SDN) to keep management tractable. SDN separates the control and data planes, exposes a programmable control interface, and enables policy updates at network scale [1, 2]. In IoT settings, where many end devices are resource-constrained and operational visibility is fragmented, this centralized control model is especially attractive [3–5]. Yet the same concentration that gives SDN its power also gives attackers a leverage point. When the controller is flooded, the entire network feels the pressure.

Recent surveys show how persistent this problem has become. Polat et al. [6] systematically review supervised and deep-learning distributed denial-of-service (DDoS) detection in SDN

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

Copyright © 2026 Pranto Halder. *Security and Communication Networks* published by John Wiley & Sons Ltd.

and show that the most recent work remains dominated by detection accuracy, feature engineering, and benchmark choice. Broader SDN and software-defined Internet of Things (SDN-IoT) reviews likewise cover lightweight statistics, classical machine learning, deep learning, federated learning, controller-centric orchestration, and trust-assisted defense [5–12]. This literature establishes a strong detection baseline against which a more complex decision framework must be judged. Accordingly, this revision separates two questions that were previously blurred: binary attack detection and multiaction mitigation selection. High detection accuracy is useful, but it does not by itself establish that a learned controller can choose among several disruptive actions safely or adaptively.

That shift from recognition to response matters. A mitigation action changes the future. Rate limiting may preserve service while suppressing a burst, or it may unnecessarily throttle benign flows. Blocking may prevent escalation, or it may create collateral damage. In multicontroller or heterogeneous deployments, the consequences can also include control-plane strain, rule churn, and asymmetric policy behavior [1, 13, 14]. Early decision-theoretic work, including Markov decision process formulations for SDN-IoT defense, already pointed to the need to treat mitigation as a sequential control problem rather than a one-shot classification task [15]. However, many current systems still represent state through flat windows of controller statistics and then map those windows directly to labels or fixed actions [8, 16–19].

This paper starts from a simple observation: DDoS behavior is rarely flat. It is relational, concentrated, and temporal. Multiple sources converge on services. Traffic surges change controller conditions over time. The meaning of a mitigation action depends on recent history. A flat feature vector can summarize these effects, but it cannot represent them explicitly. That is the motivation for the proposed framework. We model recent controller-visible activity as a temporal graph, learn action-conditioned predictions for the likely consequence of each candidate mitigation decision, and then enforce only those actions that remain acceptable under explicit safety bounds.

The framework addresses three design gaps, while this revision deliberately narrows the performance claim. First, many efficient SDN-IoT pipelines operate on flat window-level counters and do not explicitly represent source-service-controller relations through time. Second, learned action selection creates a different risk profile from binary detection because disruptive actions can damage benign service or overload the controller. Third, operational accountability requires the controller to preserve what was observed, which actions were considered, which were rejected, and why a final action was selected. The present study, therefore, evaluates temporal graph state, action-conditioned consequence prediction, a constraint-filtered risk selector, and audit-ready decision provenance as an integrated research architecture. It does not claim that the learned temporal policy is currently superior to simpler observable baselines.

The novelty is integrative rather than component-first. The paper does not claim the first use of SDN, machine learning, graph representation learning, reinforcement learning, or audit logging. Its contribution is the explicit coupling of (i) controller-observable temporal graph state, (ii) action-conditioned consequence estimates over ALLOW, RATE_LIMIT, BLOCK, and ISOLATE_SUBNET, (iii) pre-enforcement risk constraints and fallback, and (iv) decision-provenance logging. Just as importantly, the revision treats the failure of the learned policy as a scientific result. The strongest observable baselines remain better in the 130-step action replay, and the learned four-action policy remains shadow-mode only until action support, calibration, and broader independent evaluation are improved.

The remainder of the paper is organized as follows. Section 2 reviews the literature and positions the proposed framework relative to existing SDN-IoT defense work. Section 3 presents the methodology, including temporal graph construction, action-conditioned prediction, and risk-bounded selection. Section 4 describes the experimental setup and evaluation protocol. Section 5 reports the results and discusses both the strengths and the current limitations of the system. Section 6 concludes the paper and outlines future work.

2 | Related Work

2.1 | From SDN Visibility to Deployable SDN-IoT Defense

The literature on SDN-IoT DDoS defense can be read as a progression from visibility to decision-making. Early SDN foundations established why centralized control, programmability, and OpenFlow-based rule installation changed network management so profoundly [1, 2]. Later work turned that programmability into a security opportunity, especially in IoT settings where centralized visibility can compensate for limited endpoint defenses [3–5]. More recent reviews have broadened the picture further by cataloging DDoS defenses across SDN architectures, controller vulnerabilities, IoT-specific attack surfaces, and machine-learning-enabled defense pipelines [6–13, 20].

Within that broader landscape, one large family of studies emphasizes detection and immediate mitigation from controller-visible features. These include entropy-based methods, supervised classifiers, multidimensional feature models, low-rate attack mitigation, and practical experimental pipelines for stateful SDN and IoT settings [17–19, 21–25]. Such systems are attractive because they are efficient and often accurate. But they usually compress recent traffic into flat windows, thereby losing structural information about who is interacting with whom, how that interaction evolves, and how different actions might change what happens next.

A second family emphasizes collaborative or trust-enhanced defense, often with blockchain or distributed coordination layers. This direction includes blockchain-assisted ANN frameworks, collaborative mitigation schemes, federated-learning defense in AIoT, east–west interface security, intrusion detection architectures, and collaborative DDoS mitigation using smart contracts or SDN domains [21, 26–34]. These works contribute accountability, coordination, and integrity, but many of them still leave the controller with a fairly direct mapping from observed features to defense outputs. The decision is made; the infrastructure around it becomes stronger.

A third family begins to treat defense more explicitly as a decision problem. Studies on online learning, hierarchical models, real-time ensemble learning, and controller-tolerant designs move the field closer to operational decision support [5, 14, 16, 35]. The clearest conceptual bridge appears in work that frames DDoS

2037, 2026, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/sec/2116169 by Pranto Halder - Bangladesh Hinari NPL , Wiley Online Library on [06/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

mitigation through decision processes such as MDP-based defense for SDN-IoT [15]. This line of work is important because mitigation is inherently sequential: the current action affects the next state of the network.

2.2 | Why Temporal Graphs Matter

Despite these advances, most practical SDN-IoT DDoS defenses still reason over state as if it were flat. Yet coordinated attacks are not flat. They are structured in space and time. Sources converge, services become focal points, and controller stress evolves as mitigation rules accumulate. A temporal graph is, therefore, a natural state representation for a controller that must reason about relations, history, and consequences rather than about isolated counters alone.

That shift in representation also changes what the model should predict. In detection-driven pipelines, the model usually asks whether current traffic is malicious. In controller-side mitigation, the more operational question is which action will best reduce harm while preserving service and avoiding unsafe control behavior. The proposed work, therefore, differs from much of the existing literature by moving the decision target from attack recognition to action-conditioned consequence estimation. This is a subtle but important change: the controller should not only recognize danger; it should weigh alternatives under bounded risk.

2.3 | Literature Synthesis

Table 1 synthesizes the major strands of related work and highlights the gap that motivates the proposed method.

Table 1 makes three patterns visible. First, there is no shortage of work on what to observe in SDN-IoT defense. Second, there is also a growing body of work on how to detect or coordinate around DDoS events. What remains rarer is a framework that explicitly asks which action should be taken now, given the recent relational state of the network, and which candidate actions should be rejected before they can do harm? That question is where the proposed method is positioned.

2.4 | Comparison and Novelty

The novelty of the proposed framework is, therefore, integrative rather than purely component-wise. It does not claim the first use of controller-visible telemetry, machine learning, distributed security support, or decision-theoretic defense. Instead, it combines four elements that are rarely brought together in a single SDN-IoT mitigation pipeline:

1. A temporal graph state built entirely from controller-observable information;
2. Action-conditioned prediction over multiple mitigation choices;
3. Risk-bounded action filtering before enforcement; and
4. Audit-ready runtime records that preserve decision provenance.

Compared with optimized detection pipelines, the central distinction is the output being learned. Türkoğlu et al. [37] combine feature selection and hyperparameter optimization for SD-VANET DDoS recognition and report 99.35% accuracy, illustrating how careful optimization can make flat-feature detection very strong. That result is not directly comparable to the four-action mitigation task here: the present temporal policy must estimate consequences for candidate actions and is then constrained before enforcement. The lower action-selection performance is, therefore, not presented as justified by complexity; it is presented as a current limitation that the architecture must overcome. The same distinction applies to recent IoT and SDN work using off-policy PPO and GANs [38], double Q-learning with Bi-LSTM and decentralized records [40], vision transformers for IoT anomaly detection [41], and high-accuracy SDN-enabled SD-WAN detection/mitigation [42]. Recent reviews of emerging DDoS intelligence [43] and spatiotemporal policy optimization [39] further motivate sequential state modeling, but they also reinforce the need to separate classifier performance from the evidence required for safe mitigation-action selection.

2.5 | Evidence Gap Addressed by This Revision

The revision, therefore, positions the temporal graph path as a hypothesis about representation and controller decision structure, not as a performance victory. The evidence question is whether relational history, action-conditioned prediction, risk filtering, and audit provenance can be evaluated together without label leakage and without granting a weak learned policy autonomous authority. The answer supported by the present artifact is architectural feasibility with important limitations: the policy is auditable and constraint-filtered, but the 130-step replay remains small, action support is incomplete, calibration is restrictive, and the learned policy does not outperform the strongest observable baseline. These boundaries are carried through the Methodology, Results, Discussion, and Conclusion.

3 | Methodology

3.1 | Problem Formulation and System Overview

The proposed framework treats DDoS mitigation in SDN-IoT as a bounded sequential decision problem. At each decision time (t), the controller observes current telemetry, summarizes recent history as a temporal graph, predicts the expected consequence of each mitigation action, rejects actions that violate runtime bounds, and enforces the safest high-value choice.

The overall decision chain is

$$\text{event}_t \longrightarrow \text{telemetry}_t \longrightarrow \mathcal{H}_t \longrightarrow f_\Theta(\mathcal{H}_t, a) \longrightarrow \widehat{\mathbf{o}}_{t,a} \longrightarrow \mathcal{A}_t^{\text{adm}} \longrightarrow a_t^{\text{final}} \longrightarrow \text{audit}_t. \tag{1}$$

2037, 2026, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/sec/2116169 by Pranto Halder - Bangladesh Hinari NPL , Wiley Online Library on [06/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 1 | Literature synthesis and positioning of the proposed work.

Literature strand	Representative studies	Typical state representation	Typical output	Gap relative to this paper
SDN, controller, and architectural foundations	[1, 2]	Control-plane abstractions, controller models, OpenFlow rules	Architectural guidance	Foundational but not action-conditioned mitigation
Surveys, datasets, and broad SDN-IoT security reviews	[6–12, 36]	Taxonomies, benchmark datasets, controller-visible features, attack categories	Problem framing and comparative review	Broad coverage, but no unified temporal graph decision loop
ML-based DDoS detection and practical mitigation	[16, 18, 19, 22–25, 35]	Flat statistical windows, entropy, engineered features	Detection scores, staged mitigation, experimental defense	Efficient and practical, but structurally shallow
Collaborative and blockchain-assisted SDN defense	[21, 26–34]	Distributed security context, interdomain or multidomain records	Trusted coordination, logging, collaborative defense	Strong on accountability and coordination, weaker on action-conditioned state reasoning
Decision-theoretic, tolerant, and controller-aware defense	[14, 15]	Controller state, decision-process abstractions	Defensive control under uncertainty	Moves toward sequential decision-making, but not temporal graph based
This work	—	Temporal graph state built from controller-visible telemetry and recent action context	Action-conditioned effect prediction + risk-bounded selection	Unifies structural state modeling, bounded decision-making, and audit-ready deployment
Optimization-based ML detection	[37]	Selected flat features + optimized hyperparameters	DDoS class prediction	Strong detection benchmark; does not evaluate four-action consequence selection under runtime bounds
Recent RL/temporal/intelligent detection	[38, 39]	Sequential, temporal, deep, or optimized representations	Detection or adaptive cyber-threat decisions	Useful state/policy comparisons; this paper focuses on constraint-filtered mitigation-action provenance rather than detection alone

2037, 2026, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/sec/2116169 by Pravin Halder - Bangladesh Hinari NPL , Wiley Online Library on [06/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Equation (1) expresses the central logic of the framework. The controller first converts raw observations into a temporal graph history, the decision model then produces action-conditioned outcome predictions, the selector filters those predictions into an admissible action set, and the final enforced action is recorded for traceability.

The mitigation action space is

$$\mathcal{A} = \{\text{ALLOW}, \text{RATE}_{\text{LIMIT}}, \text{BLOCK}, \text{ISOLATE}_{\text{SUBNET}}\}. \quad (2)$$

Equation (2) defines the four candidate actions evaluated at each step. As illustrated conceptually in Figure 1, the controller does not map telemetry directly to one of these actions. It first evaluates the predicted consequence of each action under the current temporal graph state.

3.2 | Controller-Visible Monitoring Model

The monitoring model is intentionally controller-centric. Let packet and byte counters denote the cumulative traffic observed for a traffic entity over time. The controller estimates packet and byte rates over an observation interval as

$$r_i^{(p)}(t) = \frac{P_i(t) - P_i(t - \Delta t)}{\Delta t}, \quad (3)$$

$$r_i^{(b)}(t) = \frac{B_i(t) - B_i(t - \Delta t)}{\Delta t}. \quad (4)$$

Equations (3) and (4) define the basic rate signals from which higher-level features are derived. These features remain deployable because they do not require payload inspection or unavailable future labels.

At the window level, the controller summarizes global context as

$$\mathbf{g}_t = \left[R_t^{(p)}, R_t^{(b)}, N_t^{(\text{flow})}, N_t^{(\text{src})}, H_t^{(\text{src})}, \bar{p}_t^{(\text{attack})}, c_t^{(\text{load})}, c_t^{(\text{rtt})}, a_{t-1}\right]^{\top}, \quad (5)$$

where aggregate rates, flow counts, distinct-source counts, source entropy, attack-score proxies, controller load proxies, response-time proxies, and previous-action context are combined into a single controller-visible vector. The source entropy term in equation (5) is defined as

$$H_t^{(\text{src})} = -\sum_{s \in \mathcal{U}_t} p_t(s) \log p_t(s), \quad (6)$$

where the active-source set contains all currently observed sources in the window and each empirical probability term represents the traffic share contributed by one source. Equations (5) and (6) provide the global context that complements the graph structure in later stages of the model.

3.3 | Temporal Graph-State Construction

For each decision step, the controller constructs a graph snapshot

$$G_t = \left(V_t, E_t, \mathbf{X}_t^{(v)}, \mathbf{X}_t^{(e)}, \mathbf{g}_t\right), \quad (7)$$

where the node set, edge set, node-feature matrix, edge-feature matrix, and global controller context together define the current graph snapshot. The node set contains three semantic roles: source nodes, service nodes, and controller nodes. Source nodes

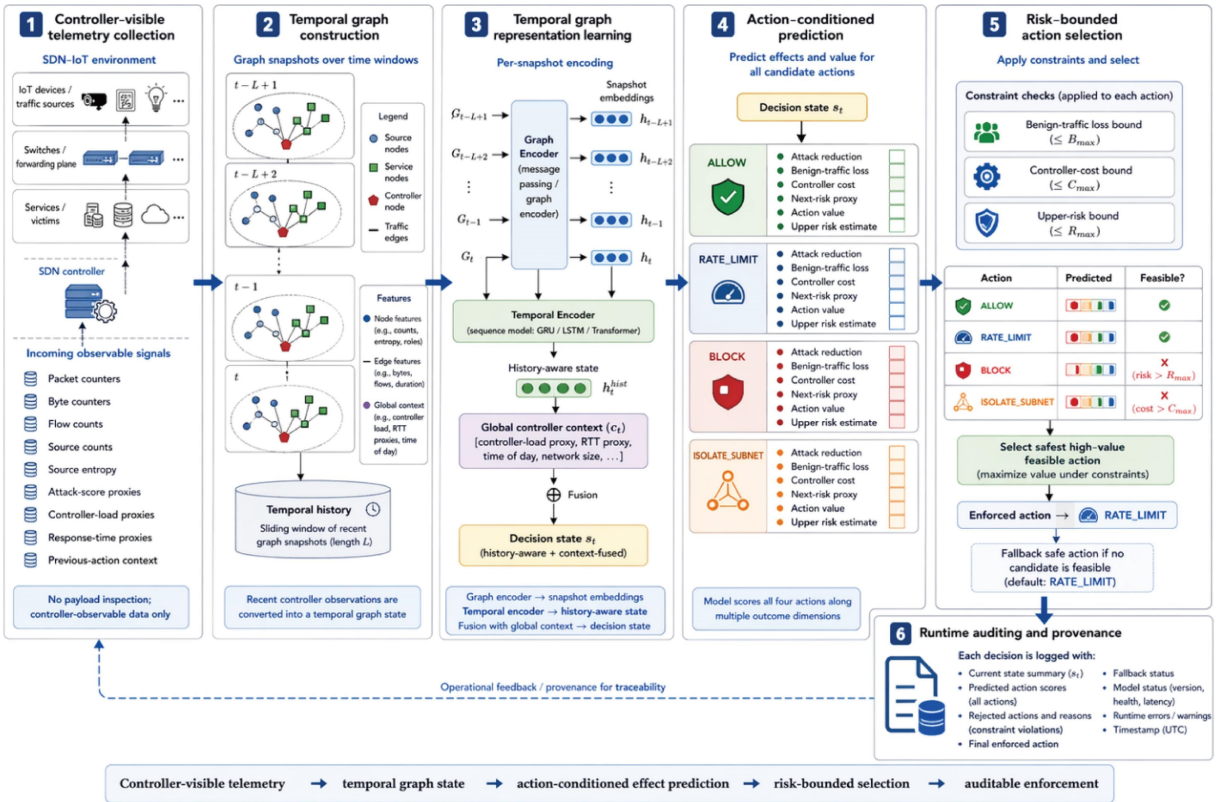


FIGURE 1 | Overall architecture of the proposed risk-bounded temporal graph decision framework, showing controller-visible telemetry collection, temporal graph construction, action-conditioned prediction, risk-bounded selection, and runtime auditing.

2037, 2026, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/sec/2116169 by Pranto Halder - Bangladesh Hinari NPL , Wiley Online Library on [06/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

capture dominant communicating origins, service nodes represent the target service or victim context, and controller nodes summarize control-plane status.

The model operates not on a single snapshot but on a temporal history of the most recent (L) snapshots as follows:

$$\mathcal{H}_t = \{G_{t-L+1}, G_{t-L+2}, \ \dots, G_t\}. \tag{8}$$

Equation (8) represents the temporal history concept. In the archived implementation used for the reported policy experiment, $L = 8$ and the prediction horizon is one step. The transition corpus contains 1282 samples produced from the audit-disabled and audit-enabled stress logs and is divided contiguously within each source-time sequence into 910 training, 196 validation, and 176 test transitions. The runtime graph uses 20-dimensional node features, 6-dimensional edge features, and 12-dimensional global features. The temporal implementation should be read carefully: the eight recent windows are aggregated into a single graph state for the reported experiment; no GRU, LSTM, or other recurrent temporal encoder was used in the archived run. Thus, "temporal" denotes the L-window history embodied in each graph state rather than a separate recurrent network.

The temporal graph-state construction used by the framework is illustrated in Figure 2, where recent controller-visible windows are transformed into graph snapshots and aggregated as a history-aware input for decision learning.

3.4 | Temporal Graph Representation Learning

Each graph snapshot in the history is first mapped to a snapshot embedding by a graph encoder as follows:

$$\mathbf{z}_\tau = f_{\text{graph}}(G_\tau; \Theta_g), \quad \tau \in \{t-L+1, \ \dots, t\}. \tag{9}$$

Equation (9) defines the graph-level encoder. At the node level, the encoder follows a message-passing formulation:

$$\mathbf{h}_{i,\tau}^{(k+1)} = \psi^{(k)}\left(\mathbf{h}_{i,\tau}^{(k)}, \underset{j \in \mathcal{N}(i)}{\text{AGG}} \phi^{(k)}\left(\mathbf{h}_{i,\tau}^{(k)}, \mathbf{h}_{j,\tau}^{(k)}, \mathbf{x}_{ij,\tau}^{(e)}\right)\right), \tag{10}$$

where the hidden state of each node is updated through neighborhood message passing, a learnable message function, a learnable update function, and a permutation-invariant aggregation operator. The graph-level readout is then

$$\mathbf{z}_\tau = \text{READOUT}\left(\left\{\mathbf{h}_{i,\tau}^{(K)} : v_i \in V_\tau\right\}\right). \tag{11}$$

Equation (11) converts node-level information into a graph representation for one snapshot. The temporal encoder then aggregates the snapshot sequence into a history-aware latent state as follows:

$$\mathbf{h}_t = f_{\text{temp}}(\mathbf{z}_{t-L+1}, \ \dots, \mathbf{z}_t; \Theta_t). \tag{12}$$

Finally, the decision state is formed by fusing the temporal graph representation with the controller-global context from equation (5) as follows:

$$\mathbf{s}_t = [\mathbf{h}_t \| \mathbf{g}_t]. \tag{13}$$

Equations (9)–(13) describe the general representation pathway. The archived implementation instantiates the graph encoder with two GraphSAGE layers (20⟶96 and 96⟶96 hidden dimensions), mean neighborhood aggregation, ReLU activation,

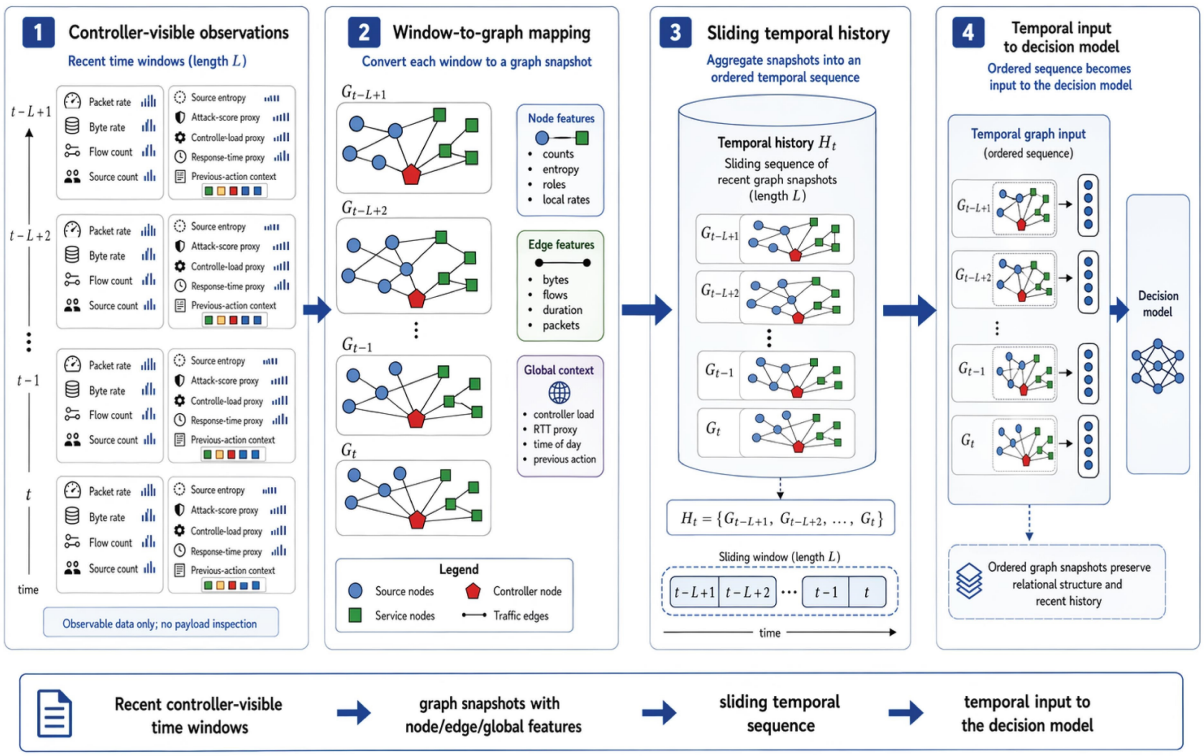


FIGURE 2 | Temporal graph-state construction. Each time window is mapped to a graph containing source, service, and controller nodes with observable node, edge, and global features. A sliding sequence of graph snapshots forms the temporal input to the decision model.

2037, 2026, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/sec/2116169 by Pranto Halder - Bangladesh Hinari NPL , Wiley Online Library on [06/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

dropout $p = 0.15$ between graph layers, and global mean pooling. The pooled graph vector is fused with the 12-dimensional controller-global context and a four-dimensional recent-action context through a 96-dimensional projection. Candidate actions are embedded in 16 dimensions before the action-conditioned output heads. The complete archived action model contains approximately 94,438 trainable parameters. These details are reported so that the conceptual formulation can be mapped to the actual experiment rather than interpreted as an unspecified graph-temporal architecture.

3.5 | Action-Conditioned Effect Prediction

Rather than directly outputting one action label, the model evaluates the likely consequence of each candidate action (a). The action-conditioned head is defined as

$$\hat{\mathbf{o}}_{t,a} = f_{\text{act}}(\mathbf{s}_t, a; \Theta_a), \quad a \in \mathcal{A}, \tag{14}$$

with the predicted outcome vector

$$\hat{\mathbf{o}}_{t,a} = \left[\hat{r}_{t,a}^{(\text{attack})}, \hat{\ell}_{t,a}^{(\text{benign})}, \hat{c}_{t,a}^{(\text{ctrl})}, \hat{\rho}_{t,a}^{(\text{risk})}, \hat{q}_{t,a}, \hat{u}_{t,a}\right], \tag{15}$$

in equation (15), the model predicts attack reduction, benign-traffic loss, controller cost, next-risk proxy, action value, and an upper risk estimate for each candidate action. The raw recommendation is simply the highest-value action,

$$a_t^{\text{raw}} = \arg\max_{a \in \mathcal{A}} \hat{q}_{t,a}, \tag{16}$$

but equation (16) is intentionally not the final enforcement rule. The model's role is to score actions; the selector's role is to reject unsafe ones.

3.6 | Logged-Transition Training Objective

Offline training uses temporal transition samples of the form

$$\mathcal{D}_n = \left(\mathcal{H}_t, a_t, G_{t+1}, \mathbf{y}_{t+1}\right), \tag{17}$$

where the logged action is the action actually observed in the transition and the next-step outcome vector contains the factual consequence signals used for supervision. The scalar factual reward used for value supervision is

$$R_t = \lambda_1 r_{t+1}^{(\text{attack})} - \lambda_2 \ell_{t+1}^{(\text{benign})} - \lambda_3 c_{t+1}^{(\text{ctrl})} - \lambda_4 \rho_{t+1}^{(\text{risk})}, \tag{18}$$

where the reward weights encode the trade-off between suppressing attack traffic and preserving benign service and controller stability.

The logged-action effect loss is

$$\mathcal{L}_{\text{eff}} = \left\| \hat{\mathbf{o}}_{t,a_t}^{\text{fact}} - \mathbf{y}_{t+1} \right\|_2^2, \tag{19}$$

the value regression loss is

$$\mathcal{L}_{\text{val}} = \left(\hat{q}_{t,a_t} - R_t\right)^2, \tag{20}$$

and the total optimization objective is

$$\mathcal{L} = \mathcal{L}_{\text{eff}} + \alpha \mathcal{L}_{\text{val}} + \beta \mathcal{L}_{\text{risk}}. \tag{21}$$

The factual value target used in the archived training run was $q = 2.0 \cdot \text{attack_reduction} - 1.5 \cdot \text{benign_loss} - 0.5 \cdot \text{controller_cost} - 1.0 \cdot \text{next_risk}$, where controller cost was scaled by 1000 during model fitting. The total loss combined mean-squared errors for attack reduction, benign loss, controller cost, next risk, q value, and risk upper bound with weights 1.0, 1.0, 0.5, 0.75, 0.75, and 0.5, respectively. These coefficients were engineering trade-off weights chosen to penalize benign damage and residual risk while still rewarding attack suppression; they were not obtained from a formal utility-elicitation study. This is now treated as a limitation rather than as an optimal reward design. The model was optimized with Adam (learning rate 10^{-3}, weight decay 10^{-5}, batch size 32, and seed 42). The archived result file corresponds to an eight-epoch run in which the checkpoint with the lowest validation loss was retained; there was no patience-based early stopping.

3.7 | Risk-Bounded Action Selection and Audit Logging

The runtime selector defines an admissible set of actions by constraining predicted benign loss, controller cost, and upper risk as follows:

$$\mathcal{A}_t^{\text{adm}} = \left\{ a \in \mathcal{A} \mid \hat{\ell}_{t,a}^{(\text{benign})} \le \tau_\ell, \hat{c}_{t,a}^{(\text{ctrl})} \le \tau_c, \hat{u}_{t,a} \le \tau_u \right\}. \tag{22}$$

Equation (22) formalizes the notion of runtime feasibility. The final enforced action is then

$$a_t^{\text{final}} = \begin{cases} \arg\max_{a \in \mathcal{A}_t^{\text{adm}}} \hat{q}_{t,a}, & \text{if } \mathcal{A}_t^{\text{adm}} \neq \varnothing, \\ a_t^{\text{fb}}, & \text{otherwise}, \end{cases} \tag{23}$$

where the fallback action is the conservative default used when no candidate satisfies the runtime constraints. In the archived experiment, the three selector thresholds were calibrated from validation predictions using the 90th empirical percentile of predicted benign loss, controller cost, and upper risk. The resulting archived thresholds were effectively zero for benign loss and upper risk and approximately 4.48×10^{-15} for the scaled controller-cost prediction. Rather than treating these values as evidence of exceptionally low risk, the revision interprets them as a calibration/scale warning: the learned outputs were compressed near zero, which can make the feasible action set overly restrictive. This behavior is one plausible contributor to conservative action selection, but it is not presented as a proven sole cause. A deployment would require re-estimation on action-complete data, scale checks, and threshold sensitivity analysis before enforcement.

Each decision is also recorded for later inspection. The audit payload can be written as

$$e_t = \left(t, \mathcal{H}_t, \left\{\hat{\mathbf{o}}_{t,a}\right\}_{a \in \mathcal{A}}, \mathcal{A}_t^{\text{adm}}, a_t^{\text{raw}}, a_t^{\text{final}}, \delta_t\right), \tag{24}$$

where the deployment metadata record stores model status, fallback status, selector details, and runtime errors. Figure 3 summarizes how the selector and audit layer close the loop between prediction and enforcement.

3.8 | Implementation, Parameter Choices, and Reproducibility

Table 2 maps the main architectural and training parameters to the archived implementation. The history length of eight windows is a compact context chosen to span short bursts without creating a large transition state, while the 96-dimensional graph representation is a moderate-capacity setting for the small

2037, 2026, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/sec/2116169 by Pranto Halder - Bangladesh Hinari NPL , Wiley Online Library on [06/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

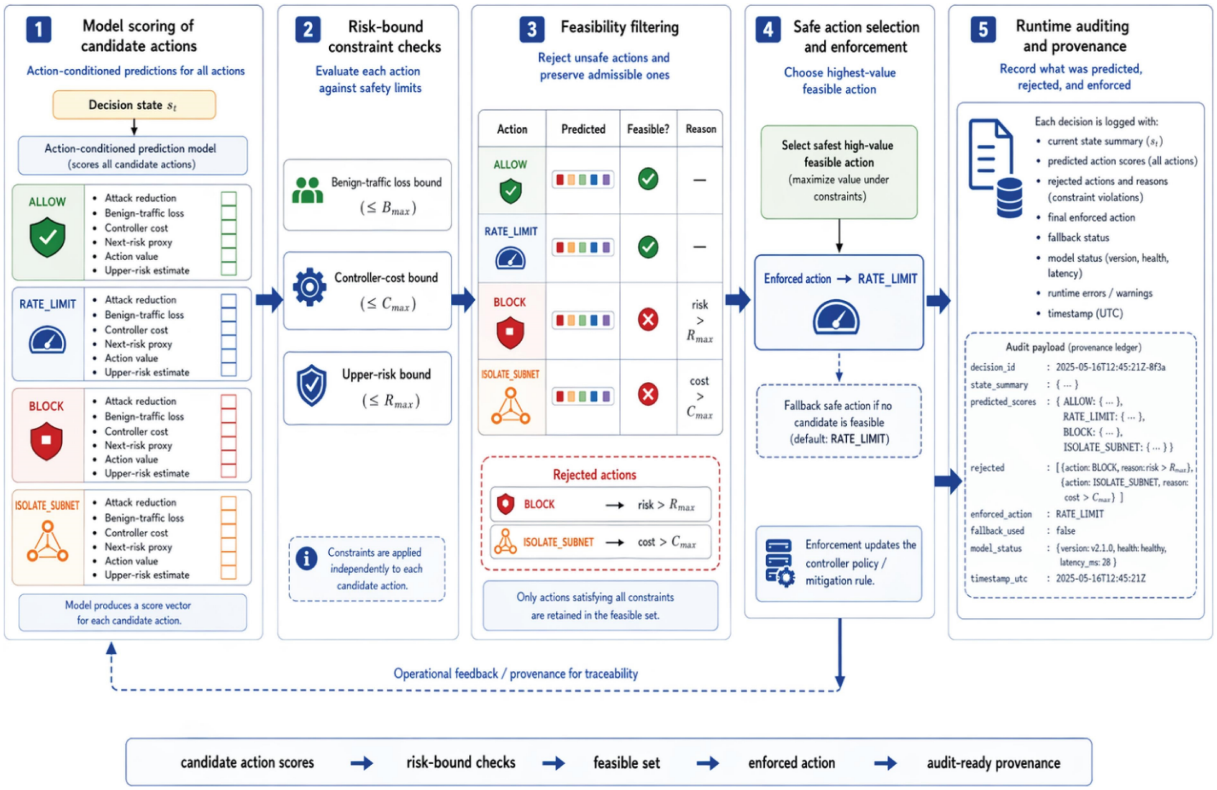


FIGURE 3 | Risk-bounded action selection and audit flow. The model scores all candidate actions, applies benign-loss, controller-cost, and upper-risk constraints, rejects unsafe actions, and records the final enforcement decision with its provenance.

experimental corpus. Because the present corpus is not large enough to support a definitive hyperparameter search, these values are reported as reproducible engineering choices rather than optimal settings. The most important reproducibility constraint is that train, validation, and test transitions are contiguous in source time and that runtime features exclude the offline target true_attack. The operational status field is also excluded from the current-state feature vector and is used only as an offline factual-action surrogate in the transition builder.

A further implementation audit revealed an important action-support limitation. The source stress logs encode status as operational states such as "normal" and "mitigated," whereas the transition builder normalizes string actions only when the text explicitly contains RATE, BLOCK, or ISOLATE. Because the archived factual-action surrogate was derived from the status field, the corpus does not provide reliable factual support for all four actions. In particular, BLOCK and ISOLATE_SUBNET are not supported as explicit logged treatments, and a generic "mitigated" status does not identify whether the underlying action was RATE_LIMIT or another mitigation. Counterfactual heads for unsupported actions therefore cannot be interpreted as if they had been trained on balanced factual outcomes. This finding materially limits the four-action learning claim and is a central part of the failure analysis in Section 5.2.

4 | Experimental Setup and Evaluation Protocol

4.1 | Evaluation Design

The evaluation was designed to answer two questions. First, does the proposed temporal graph decision framework behave sensibly under replayed DDoS decision steps when compared with simpler baselines? Second, does the broader system remain operationally practical under live-stress conditions and multi-controller deployment settings?

Three evidence layers are distinguished in this revision. First, the learned temporal policy is trained from 1282 logged one-step transitions with $L = 8$, but the archived multiaction decision replay used for Table 3 contains 130 decision steps. Those 130 steps evaluate mitigation-action selection and should not be treated as a large generalization test. Second, the original live-stress log contains 650 binary attack/benign events and evaluates detector/runtime behavior rather than the same four-action classification task. Third, additional controlled validation was performed to test temporal splitting, topology scale, repeated live/stress behavior at a common threshold, benign flash crowds, and audit overhead. These evidence layers use different labels and answer different questions; their accuracy and F1 values are therefore not interchangeable.

4.2 | Compared Systems

Five systems were compared in replay mode:

1. An observable heuristic baseline;
2. An ML-window baseline;
3. A temporal model without bounded selection;
4. A global observable selector; and
5. The full temporal model with risk selector.

This comparison makes it possible to distinguish three effects: the contribution of the temporal model itself, the contribution of

2037, 2026, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/sec/2116169 by Pranto Halder - Bangladesh Hinari NPL , Wiley Online Library on [06/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 2 | Reproducible implementation and parameter settings for the archived temporal graph policy experiment.

Parameter	Archived setting	Interpretation
Temporal history/horizon	$L = 8$ windows; horizon = 1 step	Short-history engineering choice; not claimed optimal
Transition split	910/196/176 train/validation/test	Contiguous by source time
Graph dimensions	20 node, 6 edge, 12 global features	Controller-observable graph state
Graph encoder	2 × GraphSAGE, hidden = 96, ReLU, mean aggregation	Relational source/service/controller representation
Pooling/action context	Global mean pool; 4-D recent-action context; 16-D action embedding	Graph-level decision state + candidate action
Dropout/parameters	0.15; ∼94,438 trainable parameters	Archived implementation
Optimizer	Adam, lr = 10^-3, weight decay = 10^-5, batch = 32, seed = 42	Reproducible training configuration
Training	Archived run = 8 epochs; best validation-loss checkpoint	No patience-based early stopping
Reward coefficients	+2.0 attack reduction; −1.5 benign loss; −0.5 controller cost; −1.0 next risk	Engineering utility weights; sensitivity remains future work
Loss weights	1.0/1.0/0.5/0.75/0.75/0.5	Attack/benign/controller/next-risk/q/risk-upper MSE
Risk calibration	90th percentile of validation predictions	Archived output scale produced near-zero bounds; flagged as a calibration limitation

TABLE 3 | Replay-based comparison of mitigation systems over 130 decision steps.

System	Accuracy	Precision	Recall	F1	Total reward	Benign-damage rate	Collapse score
Observable heuristic baseline	0.9385	0.9828	0.8906	0.9344	112.5	0.0077	0.5538
ML-window baseline	0.9385	0.9828	0.8906	0.9344	112.5	0.0077	0.5538
Temporal model without selector	0.4923	0.4923	1.0000	0.6598	29.0	0.5077	1.0000
Global observable selector	0.5077	0.0000	0.0000	0.0000	0.0	0.0000	1.0000
Temporal model with risk selector	0.5077	0.5000	1.0000	0.6667	32.0	0.4923	0.9846

the selector, and the residual strength of a strong observable baseline.

For live/stress testing, the strongest directly reproduced runtime comparator is the existing ML-window detector with the same controller path, evaluated with audit disabled and enabled. The archived material does not contain a matched live replay of the separate observable heuristic or a four-action temporal-policy controller under identical traffic. The revision therefore does not claim a live superiority result over those unavailable baselines. The repeated common-threshold experiment in Section 5.3 is used only to test runtime stability and audit overhead under controlled conditions.

4.3 | Metrics

Replay-mode evaluation reports classification/action-selection metrics and policy diagnostics. Because several diagnostics are nonstandard, Table 4 defines them explicitly. "Unsafe-action rate" is an empirical evaluator metric and not a formal safety theorem; "calibration coverage" denotes availability of the required calibrated risk quantities for an evaluated decision and must not be confused with statistical coverage of a conformal prediction interval. For a four-action set A, the number of rejected actions at step i is $|A| - |F_i|$, where F_i is the feasible set after risk filtering.

Live-stress evaluation reports binary detector accuracy, precision, recall, F1-score, false-positive rate (FPR), true-positive rate (TPR), mitigation-trigger success, and latency summaries. These are detector/runtime measures, whereas the replay accuracy/F1 in Table 3 refer to the action-selection comparison. The revision uses task-qualified wording whenever these values are discussed (Table 5).

4.4 | Additional Controlled Validation Protocol

To reduce dependence on the 130-step policy replay, supporting controlled experiments were added for components that can be evaluated independently of the four-action policy. A simulator-generated timestamped detector dataset contains 240 windows (120 benign and 120 attack) from six scenarios, split into 144/48/

2037, 2026, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/sec/2116169 by Pranto Halder - Bangladesh Hinari NPL , Wiley Online Library on [06/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 4 | Definitions of non-standard policy and selector metrics.

Metric	Definition	Interpretation
Unsafe-action rate	N_unsafe/N	Fraction of evaluated policy proposals/final actions that violate the experiment's admissibility/safety reference condition
Benign-damage rate	N_benign,mitigated/N_benign	Fraction of benign steps receiving disruptive mitigation
Controller-overload rate	N_cost > bound/N	Fraction of decisions exceeding the configured controller-cost bound
Collapse score	max_a N (a)/N	Dominant-action frequency; 1.0 denotes a constant policy
Calibration coverage	N_with_required_risk_quantities/N	Availability of calibrated selector quantities, not a probabilistic safety guarantee
Avg. rejected actions	mean_i (4−|F_i|)	How restrictive the risk selector is
Avg. feasible actions	mean_i|F_i|	Average candidate set remaining after filtering

TABLE 5 | Failure analysis of the archived temporal graph action-selection experiment.

Diagnostic	Archived evidence	Implication
Weak graph discrimination	Node-level ROC-AUC 0.5821; graph-level ROC-AUC 0.5000	Relational representation in the archived graph corpus is weak
Small action evaluation	130 action decisions	High uncertainty; insufficient for broad generalization claims
Incomplete factual action support	Source logs store "normal"/"mitigated" status, not explicit four-action treatment	BLOCK/ISOLATE_SUBNET and generic mitigation effects are under-supported or ambiguous
Restrictive calibration	90th-percentile validation calibration produced near-zero archived bounds	Can shrink feasible action diversity; scale mismatch needs correction
Dominant-policy output	RATE_LIMIT 128/130 after selector	The learned policy is effectively non-adaptive in the archived replay
Baseline strength	Observable heuristic and ML-window F1 = 0.9344	Simple observable state is already highly discriminative for the evaluated scenarios

48 train/validation/test windows with threshold selection on validation only. The untouched test split contains TN/FP/FN/TP = 22/2/0/24. Three controlled topology-scale scenarios vary from one switch/18 endpoints to six switches/210 endpoints. Five repeated live/stress runs per audit setting use the same detection threshold of 0.4000. A separate flash-crowd test measures false mitigation on benign surges, and an 800-record audit benchmark measures logging throughput and latency. These additions increase operational coverage but do not turn the simulator into third-party field evidence.

4.5 | Controller Placement, Controller Count, and Deployment Boundary

Controller placement and controller count can materially change telemetry delay, rule-installation delay, control-plane load, and failover behavior. The retained artifact contains heterogeneous latency measurements for primary, backup, and secondary controller roles and controlled topology-scale tests, but it does not isolate controller-to-switch hop distance or compare otherwise-identical one-controller, two-controller, and multi-controller deployments. The present revision, therefore, treats placement and controller-count effects as unresolved deployment variables. Results are reported only as indirect evidence of topology/controller heterogeneity, and no causal claim is made that a particular controller placement or controller count is optimal. A full placement study requires matched topologies in which hop count, controller proximity, switch assignment, and controller population are varied independently.

5 | Results and Discussion

5.1 | Replay-Based Comparative Results

Table 3 reports the replay-based comparison. Two observations stand out immediately. First, the strongest performance still comes from the observable heuristic baseline and the ML-window baseline, which both achieved 0.9385 accuracy, 0.9344 F1-score, and 112.5 total reward over 130 steps. Second, the temporal model with a risk selector performs slightly better than the same temporal model without the selector but remains well below the strongest baseline.

Table 3 shows only a small numerical change when the risk selector is added: F1 increases from 0.6598 to 0.6667 and total reward from 29.0 to 32.0. Because this comparison is based on a single archived 130-step replay and no repeated-seed significance test is available for the policy selector, the revision no longer characterizes this change as statistically reliable improvement. The correct conclusion is narrower: the selector changes the action output while the underlying temporal model remains far below the strongest observable baselines.

2037, 2026, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/sec/2116169 by Pranto Halder - Bangladesh Hinari NPL , Wiley Online Library on [06/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

5.2 | Safety and Action-Selection Behavior

The full temporal model with the risk selector produced zero observed unsafe-action events and zero observed controller-overload events in the 130-step replay, with the selector quantities available for all evaluated steps. These results are empirical constraint-satisfaction observations only. With zero observed violations in $N = 130$ trials, a one-sided 95% exact binomial upper confidence bound on the underlying violation probability is approximately 2.28%; therefore, zero observed violations cannot be interpreted as a general safety guarantee.

Action diversity is the dominant failure. The temporal model with the selector chose RATE_LIMIT in 128 of 130 steps, ALLOW twice, and never selected BLOCK or ISOLATE_SUBNET; collapse score = 0.9846 and benign-damage rate = 0.4923. The selector rejected an average of 2.1846 of the four candidate actions, leaving 1.8154 feasible actions per step on average. More importantly, inspection of the training pipeline revealed a factual-support problem: the archived stress logs record operational status ("normal" or "mitigated") rather than an explicit four-action treatment label, while the transition builder derives the factual action from that status field. The four-action counterfactual model is, therefore, not trained from balanced factual observations of ALLOW, RATE_LIMIT, BLOCK, and ISOLATE_SUBNET. This data-support defect, together with weak graph discrimination, reward design, and near-zero calibrated bounds, provides a concrete explanation for why the policy can collapse. Correcting the action logger and retraining on action-complete transitions is required before autonomous enforcement is scientifically defensible.

The interpretation is, therefore, deliberately conservative. The architecture can filter and record action proposals, but the archived learned policy does not demonstrate effective four-action adaptation. Its current value is as a shadow-mode policy-evaluation scaffold that exposes feasibility, rejection, fallback, and audit information while the stronger observable baseline retains control authority.

The resulting action-selection distribution across replay decisions is presented in Figure 4, highlighting the observed behavior of the evaluated mitigation policies.

5.3 | Live-Stress Performance

Table 6 reports the best binary detector operating point separately for audit-disabled and audit-enabled live-stress logs. The best audit-disabled setting reaches 0.9092 accuracy and 0.9002 F1, while the best audit-enabled setting reaches 0.9231 accuracy and 0.9141 F1. These thresholds differ (0.3049 versus 0.3208), so the difference must not be attributed causally to audit logging.

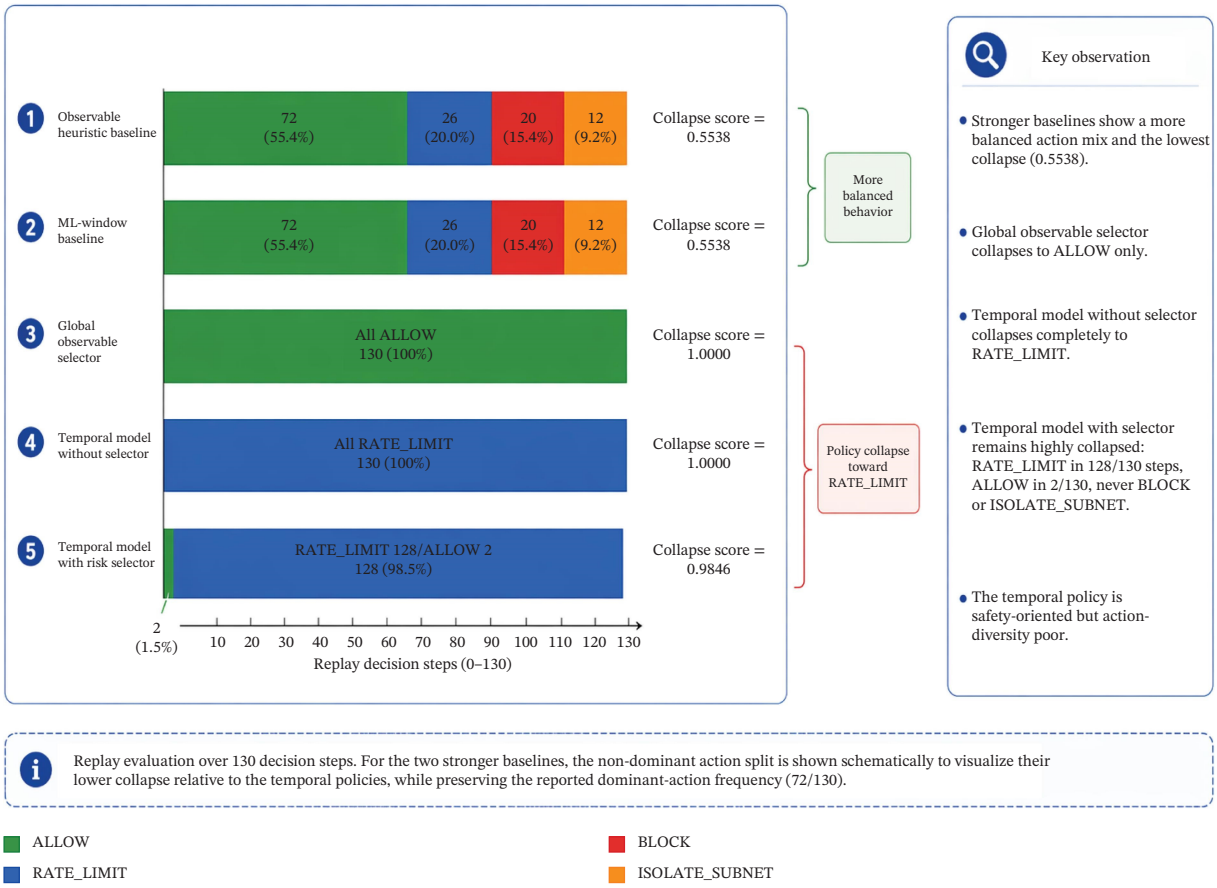


FIGURE 4 | Action-distribution comparison over 130 replay decisions. The full temporal model with risk selector selects RATE_LIMIT 128 times and ALLOW twice, with no BLOCK or ISOLATE_SUBNET selections; its collapse score is max_a N (a)/N = 128/130 = 0.9846. The figure is diagnostic evidence of policy collapse, not evidence of autonomous deployment readiness.

2037, 2026, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/sec/2116169 by Pranto Halder - Bangladesh Hinari NPL , Wiley Online Library on [06/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Audit logging should primarily affect runtime overhead and provenance, not predictive discrimination. The independently optimized rows are retained for completeness, and Table 7 adds five repeated controlled runs at a common threshold of 0.4000.

Table 6 also illustrates the expected trade-off between aggressiveness and restraint. When the operating point is tuned to minimize FPR below approximately 2%, the model becomes much more conservative, recall drops sharply, and the F1-score falls accordingly. This behavior is not unexpected; it simply reveals the tension between avoiding false alarms and maintaining detection sensitivity under stress.

The broader 650-event live-stress summary uses audit-disabled and audit-enabled terminology. In the archived run, the corresponding TPR/FPR values were 0.7934/0.0319 and 0.7770/0.0377. These binary detector values should not be compared directly with the four-action accuracy/F1 in Table 3. The audit mechanism in this implementation is a permissioned hash-linked audit log; it does not implement a public blockchain consensus protocol. References to "with blockchain" and "no blockchain" have, therefore, been removed from the runtime-results discussion.

5.4 | Latency and Deployment Practicality

For controller-side defense, a mitigation method that is accurate but too slow can still fail in practice. Table 8, therefore, reports the multicontroller latency summary. The primary controller recorded mean latency of 77.53 ms without audit and 81.55 ms with audit. The backup controller recorded 82.57 ms without audit and 75.24 ms with audit. The secondary controller showed the heaviest overhead, with mean latency rising from 101.45 ms without audit to 143.40 ms with audit (Table 9).

Table 8 demonstrates substantial controller heterogeneity rather than universal real-time suitability. The secondary controller reaches P99 = 579.40 ms with audit enabled, which is large enough to matter for fast DDoS response and should not be averaged away. The archived experiments were executed on Ubuntu 25.10/Linux with an Intel Core i7-1165G7 (4 cores/8 threads, 2.80 GHz) and approximately 7 GiB RAM. Retained process traces from the baseline environment show roughly 3.01% mean Python CPU usage and about 217 MB resident memory; the audit service trace is much smaller, but these process-level measurements do not isolate graph construction, feature extraction, inference, controller communication, or rule installation. Those stage-wise timings were not instrumented separately in the archived policy run and are, therefore, reported as a reproducibility limitation rather than reconstructed retrospectively.

5.5 | Additional Controlled Validation and Topology-Scale Evidence

The controlled timestamped detector split provides an additional leakage-aware internal check: at a validation-selected threshold

TABLE 6 | Live-stress summary at the best operating points.

Setting	Selector rule	Threshold	Accuracy	Precision	Recall	F1	FPR
No audit	Best by F1	0.3049	0.9092	0.9301	0.8721	0.9002	0.0580
With audit	Best by F1	0.3208	0.9231	0.9603	0.8721	0.9141	0.0319
No audit	Best FPR $\leq 2\%$	0.7669	0.6923	0.9487	0.3648	0.5273	0.0174
With audit	Best FPR $\leq 2\%$	0.7669	0.7000	0.9741	0.3705	0.5368	0.0087

TABLE 7 | Five repeated controlled live/stress runs per audit setting at the same detection threshold.

Setting	Runs	Common threshold	Precision	Recall	F1	Accuracy	FPR	P99 latency (ms)
Audit disabled	5	0.4000	1.0000	0.9519	0.9753	0.9759	0.0000	147.50
Audit enabled	5	0.4000	0.9964	0.9926	0.9944	0.9944	0.0037	171.92

Note: Differences are descriptive and are not interpreted as an audit-caused change in predictive accuracy.

TABLE 8 | Multicontroller latency summary.

Controller	Setting	Mean latency (ms)	P95 (ms)	P99 (ms)
ctrl_primary	No audit	77.53	85.71	99.48
ctrl_primary	With audit	81.55	102.24	123.97
ctrl_backup	No audit	82.57	100.69	114.21
ctrl_backup	With audit	75.24	83.38	101.27
ctrl_secondary	No audit	101.45	157.88	273.81
ctrl_secondary	With audit	143.40	306.68	579.40

2037, 2026, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/sec/2116169 by Pranto Halder - Bangladesh Hinari NPL , Wiley Online Library on [06/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 9 | Controlled microbenchmark of the permissioned hash-linked audit log.

Audit benchmark item	Controlled result
Records appended	800
Mean write latency	8.4516 ms
P50/P90/P99 write latency	8.2072/9.9938/13.0228 ms
Full-chain verification time	17.2219 ms
Append throughput	118.20 records/s
Average storage	352.59 bytes/record
Hash-chain validation	Passed in the controlled benchmark
Audit terminology	Permissioned hash-linked JSONL audit log (not public blockchain consensus)

TABLE 10 | Controlled topology-scale detector/runtime results.

Topology	Scale	F1	Observed FPR	Mitigation success	P90 latency (ms)
Small smarthome	1 switch/18 endpoints	0.8119	0.0000 (0 FP)	0.6833	97.83
Medium industrial	3 switches/74 endpoints	0.9167	0.1818 (12 FP/66 benign)	1.0000	195.79
Larger multiswitch	6 switches/210 endpoints	0.8136	0.4583 (33 FP/72 benign)	1.0000	296.32

Note: Topology size changes several factors simultaneously; the table is not an isolated controller-placement experiment.

of 0.1316, the 48-window test set produces TN/FP/FN/TP = 22/2/0/24, accuracy = 0.9583, precision = 0.9231, recall = 1.0000, F1 = 0.9600, ROC-AUC = 0.9965, and PR-AUC = 0.9968. These timestamps are simulator-generated elapsed-experiment timestamps and should not be interpreted as a historical field trace. The topology-scale results in Table 10 reveal a different limitation: the same detector behavior changes materially as the controlled topology grows, particularly in false positives and latency. This supports topology-aware calibration and also shows why the available topology tests cannot substitute for a proper controller-placement experiment.

Benign-surge stress testing also exposes a practical safety limitation. In the controlled flash-crowd scenario, 4 of 72 benign events were falsely mitigated, corresponding to a benign-damage rate of 0.0556, while the normal-benign and reconnect-storm scenarios produced no false mitigations in the retained runs. This finding argues for temporal smoothing, per-topology calibration, or human approval before disruptive actions in real deployments.

5.6 | Discussion

Taken together, the evidence supports a bounded systems claim rather than a claim of solved adaptive mitigation. The architecture separates observable telemetry, action scoring, constraint filtering, fallback, and audit provenance. The learned temporal policy, however, remains weak: its graph representation is only weakly discriminative, its logged factual action support is incomplete, its calibrated bounds are restrictive, and its 130-step action evaluation is too small for broad generalization. The strongest observable baselines therefore remain the appropriate enforcement reference in the current artifact, while the temporal policy remains shadow mode.

The negative policy result remains scientifically useful because it identifies where complexity fails to translate into decision quality. The architecture demonstrates feasibility of an auditable and constraint-filtered mitigation loop, not a proof that the architectural or policy-learning problem is solved. Future evidence must come from explicit four-action logging, action-support-aware offline learning, calibrated threshold sensitivity, repeated policy training, matched controller-placement/controller-count experiments, and independent timestamped traffic before the temporal graph path can be considered for autonomous enforcement.

6 | Conclusions and Future Work

This paper presented and critically re-evaluated a risk-bounded temporal graph decision framework for DDoS mitigation in SDN-IoT. The framework changes the controller abstraction from one-shot attack labeling to history-aware candidate-action scoring, constraint filtering, fallback, and auditable provenance. The revision also makes the evidence boundary explicit: the current learned temporal policy is a research component evaluated in shadow mode, not a production-ready autonomous mitigation controller.

The evidence is mixed by design. In the 130-step action replay, the learned temporal policy remains far below the observable baselines and collapses toward RATE_LIMIT. Zero observed unsafe-action and controller-overload events are only empirical results for that finite replay. Pipeline inspection further shows that the archived transition logs do not contain an explicit, balanced four-action treatment record, which limits factual support for counterfactual action learning. On the positive side, supporting controlled experiments show strong binary detection on a timestamped temporal split, reproducible topology sensitivity, stable repeated live/stress operation at a common threshold, measurable benign flash-crowd false mitigation, and bounded local audit overhead. These findings support the

2037, 2026, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/sec/2116169 by Pranto Halder - Bangladesh Hinari NPL , Wiley Online Library on [06/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

architecture as an auditable experimental foundation while leaving effective adaptive-policy learning unresolved.

The immediate next step is to correct the data-generation contract: every enforced action must be logged explicitly as ALLOW, RATE_LIMIT, BLOCK, or ISOLATE_SUBNET together with its preaction state and postaction outcome. That action-complete corpus can support balanced or support-aware sampling, conservative offline learning, uncertainty-aware selection, threshold and reward sensitivity analysis, per-action precision/recall, and repeated-seed significance testing. Deployment evaluation should independently vary controller-to-switch distance and hop count, controller population, topology size, rule-installation delay, and failover. Stage-wise timing should separately instrument feature extraction, graph construction, model inference, controller communication, rule installation, and audit append. Until those experiments are completed, the temporal graph policy should remain shadow-mode only.

In short, the study argues that accountable SDN-IoT mitigation should behave less like a one-step classifier and more like a well-instrumented controller: it should represent recent context, compare alternatives, enforce explicit constraints, retain a conservative fallback, and leave behind an auditable explanation of what it decided and why.

Author Contributions

Pranto Halder: conceptualization, methodology, software, formal analysis, investigation, validation, visualization, writing–original draft, and writing–review and editing.

Funding

This research received no external funding.

Conflicts of Interest

The author declares no conflict of interest.

Data Availability Statement

The data, code, and other supporting information used to generate the findings reported in this study are available from the author upon reasonable request.

References

1. Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A Survey on Software Defined Networking With Multiple Controllers," *Journal of Network and Computer Applications* 103 (2018): 101–118, https://doi.org/10.1016/j.jnca.2017.11.015.

2. N. McKeown, T. Anderson, H. Balakrishnan, et al., "Openflow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review* 38, no. 2 (2008): 69–74, https://doi.org/10.1145/1355734.1355746.

3. C. Singh and A. K. Jain, "A Comprehensive Survey on DDoS Attacks Detection & Mitigation in SDN-IoT Network," *E-Prime-Advances in Electrical Engineering, Electronics and Energy* 8 (2024): 100543, https://doi.org/10.1016/j.prime.2024.100543.

4. A. K. Jain, H. Shukla, and D. Goel, "A Comprehensive Survey on DDoS Detection, Mitigation, and Defense Strategies in Software-Defined Networks," *Cluster Computing* 27, no. 9 (2024): 13129–13164, https://doi.org/10.1007/s10586-024-04596-z.

5. A. H. Janabi, T. Kanakis, and M. Johnson, "Survey: Intrusion Detection System in Software-Defined Networking," *IEEE Access* 12 (2024): 164097–164120, https://doi.org/10.1109/ACCESS.2024.3493384.

6. O. Polat, Ö. Durmuş, F. Doğan, et al., "Supervised and Deep Learning Techniques for DDoS Detection in Software-Defined Network Architectures: A Systematic Review," *Engineering Science and Technology, an International Journal* 75 (2026): 102290, https://doi.org/10.1016/j.jestch.2026.102290.

7. W. Wang, Y. Liu, Q. Meng, and Z. Chen, "DDoS Attack Detection and Defense Techniques in Software Defined Networks: A Survey," *Computer Science Review* 60 (2026): 100921, https://doi.org/10.1016/j.cosrev.2026.100921.

8. M. Sinha, "A Comprehensive Survey of DDoS Attack Defense Systems for Different SDN Architectures," *Computer Networks* 272 (2025): 111711, https://doi.org/10.1016/j.comnet.2025.111711.

9. S. Batool, M. Aslam, E. Akpokodje, and S. F. Jilani, "A Comprehensive Review of DDoS Detection and Mitigation in SDN Environments: Machine Learning, Deep Learning, and Federated Learning Perspectives," *Electronics* 14, no. 21 (2025): 4222, https://doi.org/10.3390/electronics14214222.

10. D. Kalambe, D. Sharma, P. Kadam, and S. Surati, "A Comprehensive Plane-Wise Review of DDoS Attacks in SDN: Leveraging Detection and Mitigation Through Machine Learning and Deep Learning," *Journal of Network and Computer Applications* 235 (2025): 104081, https://doi.org/10.1016/j.jnca.2024.104081.

11. T. E. Ali, Y.-W. Chong, and S. Manickam, "Machine Learning Techniques to Detect a DDoS Attack in SDN: a Systematic Review," *Applied Sciences* 13, no. 5 (2023): 3183, https://doi.org/10.3390/app13053183.

12. S. Kaur, K. Kumar, N. Aggarwal, and G. Singh, "A Comprehensive Survey of DDoS Defense Solutions in SDN: Taxonomy, Research Challenges, and Future Directions," *Computers & Security* 110 (2021): 102423, https://doi.org/10.1016/j.cose.2021.102423.

13. J. Arevalo-Herrera, J. Camargo Mendoza, J. I. Martínez Torre, T. Zona-Ortiz, and J. M. Ramirez, "Assessing SDN Controller Vulnerabilities: a Survey on Attack Typologies, Detection Mechanisms, Controller Selection, and Dataset Application in Machine Learning," *Wireless Personal Communications* 140, no. 1 (2025): 739–775, https://doi.org/10.1007/s11277-025-11748-w.

14. N. Sanoussi, K. Chetioui, G. Orhanou, and S. El Hajji, "ITC: Intrusion Tolerant Controller for Multicontroller SDN Architecture," *Computers & Security* 132 (2023): 103351, https://doi.org/10.1016/j.cose.2023.103351.

15. J. Zheng and A. S. Namin, "Defending SDN-Based IoT Networks Against DDoS Attacks Using Markov Decision Process," in *Proceedings of the 2018 IEEE International Conference on Big Data* (Seattle, WA: IEEE, 2018), 4589–4592, https://doi.org/10.1109/BigData.2018.8622064.

16. S. Kaur, K. Kumar, and N. Aggarwal, "Enhancing DDoS Defense in SDN Using Hierarchical Machine Learning Models," *Journal of Network and Computer Applications* 239 (2025): 104168, https://doi.org/10.1016/j.jnca.2025.104168.

17. A. Pakmehr, A. Aßmuth, N. Taheri, and A. Ghaffari, "DDoS Attack Detection Techniques in IoT Networks: A Survey," *Cluster Computing* 27, no. 10 (2024): 14637–14668, https://doi.org/10.1007/s10586-024-04662-6.

18. A. I. Hassan, E. Abd El Reheem, and S. K. Guirguis, "An Entropy and Machine Learning Based Approach for DDoS Attacks Detection in Software Defined Networks," *Scientific Reports* 14, no. 1 (2024): 18159, https://doi.org/10.1038/s41598-024-67984-w.

19. M. W. Nadeem, H. G. Goh, V. Ponnusamy, et al., "DDoS Detection in SDN Using Machine Learning Techniques," *Computers, Materials & Continua* 71, no. 1 (2022): 771–789, https://doi.org/10.32604/cmc.2022.021669.

20. Y. Su, D. Xiong, K. Qian, and Y. Wang, "A Comprehensive Survey of Distributed Denial of Service Detection and Mitigation Technologies in Software-Defined Network," *Electronics* 13, no. 4 (2024): 807, https://doi.org/10.3390/electronics13040807.

21. N. Indrason and G. Saha, "Exploring Blockchain-Driven Security in SDN-Based IoT Networks," *Journal of Network and Computer Applications* 224 (2024): 103838, https://doi.org/10.1016/j.jnca.2024.103838.

2037, 2026, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/sec/2116169 by Pranto Halder - Bangladesh Hinari NPL , Wiley Online Library on [06/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

22. K. Wang, Y. Fu, X. Duan, and T. Liu, "Detection and Mitigation of DDoS Attacks Based on Multi-Dimensional Characteristics in SDN," *Scientific Reports* 14, no. 1 (2024): 16421, https://doi.org/10.1038/s41598-024-66907-z.

23. S. Wang, J. F. Balarezo, K. G. Chavez, et al., "Detecting Flooding DDoS Attacks in Software Defined Networks Using Supervised Learning Techniques," *Engineering Science and Technology, an International Journal* 35 (2022): 101176, https://doi.org/10.1016/j.jestch.2022.101176.

24. J. A. Pérez-Díaz, I. A. Valdovinos, K.-K. R. Choo, and D. Zhu, "A Flexible SDN-Based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks Using Machine Learning," *IEEE Access* 8 (2020): 155859–155872, https://doi.org/10.1109/ACCESS.2020.3019330.

25. J. Galeano-Brajones, M. Carmona-Murillo, F. J. Rodriguez, and F. Luna-Valero, "Detection and Mitigation of DoS and DDoS Attacks in IoT-Based Stateful SDN: An Experimental Approach," *Sensors* 20, no. 3 (2020): 816, https://doi.org/10.3390/s20030816.

26. F. S. Alsubaei, M. Alshaikh, R. Amin, and S. Mehmood, "An Efficient Blockchain-Based ANN Framework for SDN-IoT Cyberdefense," *PeerJ Computer Science* 12 (2026): e3528, https://doi.org/10.7717/peerj-cs.3528.

27. S. Garg, S. Goyal, and A. Bhandari, "A Lightweight Blockchain Based Scalable and Collaborative Mitigation Framework Against New Flow DDoS Attacks in SDN Enabled Autonomous Systems," *Scientific Reports* 15, no. 1 (2025): 36002, https://doi.org/10.1038/s41598-025-19989-2.

28. J. Ma and W. Su, "Collaborative DDoS Defense for SDN-Based AIoT With Autoencoder-Enhanced Federated Learning," *Information Fusion* 117 (2025): 102820, https://doi.org/10.1016/j.inffus.2024.102820.

29. S. Peng, J. Tian, X. Zheng, S. Chen, and Z. Shu, "DDoS Defense Strategy Based on Blockchain and Unsupervised Learning Techniques in SDN," *Future Internet* 17, no. 8 (2025): 367, https://doi.org/10.3390/fi17080367.

30. H. Alrashede, F. Eassa, A. Marish Ali, H. Aljihani, and F. Albalwy, "Enhancing East-West Interface Security in Heterogeneous SDN via Blockchain," *PeerJ Computer Science* 11 (2025): e2914, https://doi.org/10.7717/peerj-cs.2914.

31. H. Alrashede, F. Eassa, A. Marish Ali, F. Albalwy, and H. Aljihani, "A Blockchain-Based Security Framework for East-West Interface of SDN," *Electronics* 13, no. 19 (2024): 3799, https://doi.org/10.3390/electronics13193799.

32. C. Núñez-Gómez, M. del C. Carrión, B. Caminero, and F. M. Delicado, "S-HIDRA: A Blockchain and SDN Domain-Based Architecture to Orchestrate Fog Computing Environments," *Computer Networks* 221 (2023): 109512, https://doi.org/10.1016/j.comnet.2022.109512.

33. S. Jiang, L. Yang, X. Gao, et al., "BSD-Guard: A Collaborative Blockchain-Based Approach for Detection and Mitigation of SDN-Targeted DDoS Attacks," *Security and Communication Networks* 2022 (2022): 1608689–16, https://doi.org/10.1155/2022/1608689.

34. Z. A. El Houda, A. S. Hafid, and L. Khoukhi, "Cochain-SC: an Intra- and Inter-Domain DDoS Mitigation Scheme Based on Blockchain Using SDN and Smart Contract," *IEEE Access* 7 (2019): 98893–98907, https://doi.org/10.1109/ACCESS.2019.2930715.

35. S. A. Madoune, S. Senouci, D. D. Jiang, et al., "A Novel Approach for Real-Time DDoS Detection in SDN Using Dimensionality Reduction and Ensemble Learning," *Journal of Information Security and Applications* 94 (2025): 104195, https://doi.org/10.1016/j.jisa.2025.104195.

36. T. L. Yasarathna and N.-A. Le-Khac, "ASEADOS-SDN-IoT: A Novel SDN-IoT Network Intrusion Detection Dataset and Framework," *Internet of Things* 36 (2026): 101891, https://doi.org/10.1016/j.iot.2026.101891.

37. M. Türkoğlu, H. Polat, C. Koçak, and O. Polat, "Recognition of DDoS Attacks on SD-VANET Based on Combination of Hyperparameter Optimization and Feature Selection," *Expert Systems with Applications* 203 (2022): 117500, https://doi.org/10.1016/j.eswa.2022.117500.

38. J. Yang, V. Govindarajan, L. Y. Por, et al., "DDoS Attack Detection in Consumer IoT-Based Healthcare Systems Using Improved Off-Policy Proximal Policy Optimization and Generative Adversarial Network," *IEEE Transactions on Consumer Electronics* 71, no. 4 (2025): 11662–11675, https://doi.org/10.1109/TCE.2025.3605098.

39. J. Yang and L. Y. Por, "Intelligent Spatiotemporal Cyber Threat Detection in Smart Cities Using Trust Region Policy Optimization and Online Data Augmentation," *Reliability Engineering & System Safety* 277, no. Part 2 (2027): 113053, https://doi.org/10.1016/j.ress.2026.113053.

40. J. Yang, E. Rezvani, M. A. Khan, et al., "QB-AutoIDS: A Blockchain-Based Decentralized Autonomous Cyberattack Detection System for Consumer Electronics Using Hybrid Double Q-Learning and Bi-LSTM," *IEEE Transactions on Consumer Electronics* 71, no. 4 (2025): 11487–11496, https://doi.org/10.1109/TCE.2025.3600429.

41. L. Sana, M. M. Nazir, J. Yang, et al., "Securing the IoT Cyber Environment: Enhancing Intrusion Anomaly Detection With Vision Transformers," *IEEE Access* 12 (2024): 82443–82468, https://doi.org/10.1109/ACCESS.2024.3404778.

42. M. Musa, T. F. Ang, Y.-L. Chen, et al., "Distributed Denial of Service Detection and Mitigation in Software-Defined Networking-Enabled Software-Defined Wide Area Networks," *PLoS One* 21, no. 5 (2026): e0346673, https://doi.org/10.1371/journal.pone.0346673.

43. J. Yang, M. A. Khan, N. Niu, et al., "Large Language Models for DDoS Attack Detection: A Systematic Literature Review," *IEEE Internet of Things Journal* (2026): 1, https://doi.org/10.1109/JIOT.2026.3712446.

2037, 2026, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1155/sec/2116169 by Pranto Halder - Bangladesh Hinari NPL , Wiley Online Library on [06/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License