ADADOS: ADAPTIVE DOS ATTACK VIA DEEP ADVERSARIAL REINFORCEMENT LEARNING IN SDN

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing defence mechanisms have demonstrated significant effectiveness in mitigating rule-based Denial-of-Service (DoS) like attacks, leveraging predefined signatures and static heuristics to identify and block malicious traffic. However, the emergence of AI-driven techniques presents new challenges to SDN security, potentially compromising the efficacy of existing defence mechanisms. In this paper, we introduce *AdaDoS*, an adaptive attack model that disrupt network operations while evading detection by existing DoS-based detectors through adversarial reinforcement learning (RL). Specifically, AdaDoS models the problem as a competitive game between an attacker, whose goal is to obstruct network traffic without being detected, and a detector, which aims to identify malicious traffic. AdaDoS can solve this game by dynamically adjusting its attack strategy based on feedback from the SDN and the detector. Additionally, recognising that attackers typically have less information than defenders, AdaDoS formulates the DoS-like attack as a partially observed Markov decision process (POMDP), with the attacker having access only to delay information between attacker and victim nodes. We address this challenge with a novel reciprocal learning module, where the student agent, with limited observations, enhances its performance by learning from the teacher agent, who has full observational capabilities in the SDN environment. AdaDoS represents the first application of RL to develop DoS-like attack sequences, capable of adaptively evading both machine learning-based and rule-based DoS-like attack detectors.Experimental results show that AdaDoS achieves attack success rates of 79.0–87.1% across four network topologies, while reducing average available bandwidth to 0.1 Mbps in the simple topology.

1 INTRODUCTION

Denial-of-Service (DoS) attacks disrupt legitimate communications by exhausting network resources. Low-rate DoS (LDoS) attacks instead use intermittent traffic bursts, exploiting protocol or device behaviour while avoiding sustained high-volume traffic. Nevertheless, their regular patterns remain detectable. Recent SDN defence frameworks, including GASF-IPP (Tang et al., 2023b) and FSS-RSR (Tang et al., 2023a), extract traffic features and apply machine learning to identify malicious activity.

Two challenges motivate adaptive DoS-like attack modelling. First, fixed attack patterns limit survival: recurring timing and intensity provide recognisable features for detectors. Second, attackers operate with substantially less information than defenders. SDN controllers can access global traffic and network configuration, whereas attackers may lack topology information, protocol parameters, and internal network measurements. These limitations make it difficult to sustain disruption while responding to changing network conditions.

We propose *AdaDoS*, an adversarial reinforcement learning framework for adaptive DoS-like attacks under partial observation. Its policy separates attack timing from attack shaping: a *decider* determines whether to attack, and a *shaper* selects the rate and duration. The policy is updated using network and detector feedback, with a reward that balances detection avoidance, congestion, and resource consumption.

To address limited observability, AdaDoS introduces reciprocal learning between a teacher with global network observations and a student observing only end-to-end delays. The student learns

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

from the teacher through reward-based policy transfer. When the student performs better, reciprocal updates also allow the teacher to benefit from the student's policy. This mechanism supports policy learning under restricted observations without requiring the deployed student to access controller-level measurements.

Our contributions are threefold:

- We propose AdaDoS, a novel adaptive attack approach leveraging adversarial reinforcement learning to dynamically adjust the attack strategy in response to feedback from SDN environment in real time, enabling it to elude detection by both rule-based and machine learning-based security systems.
- We propose a novel two-stage model that determines the optimal timing and adjusts the intensity of attacks. This model utilises a carefully designed reward function to evade detection and improve the survival ability.
- We design a teacher-student structure based transfer learning framework to enhance the performance of the attacker with partial observation and adaptive to various SDN environment without prior knowledge about network configuration and topology. To further improve the efficacy of this framework, we introduce a novel learning mechanism that facilitates reciprocal learning between the teacher and student agents.
- We conducted extensive experiments to validate the generality, robustness and performance of the AdaDoS compared to previous DoS-like attack. AdaDoS significantly outperforms the baselines with various settings.

2 Related Work

DoS-like attacks in SDN. DoS attacks can exhaust controller resources, communication channels, or forwarding tables (Shang et al., 2017; Ahmad et al., 2015; Wang et al., 2015). LDoS attacks reduce sustained traffic volume by exploiting specific network mechanisms. CrossPath targets shared control and data links (Cao et al., 2019), while LOFT and related attacks exploit flow-table capacity and rule-management behaviour (Cao et al., 2018; Pascoal et al., 2020). AdaDoS focuses on adapting attack decisions to network and detector feedback rather than following a fixed traffic pattern.

Detection and mitigation. SDN defences include flow caching, resource allocation, and routing or rule-management strategies (Wang et al., 2015/12; Yuan et al., 2019; Guo et al., 2017; Zhao et al., 2018; Sattar et al., 2016). Learning-based methods classify traffic using models such as SVMs, neural networks, and KNN (Meti et al., 2017; Ezekiel et al., 2017; Prakash & Priyadarshini, 2018; Phan et al., 2016). For LDoS, anomaly-based approaches use clustering, entropy, or sequence analysis (Li et al., 2021; Tang et al., 2022; Xiang et al., 2011; Wu et al., 2019), whereas feature-based approaches analyse burst patterns and traffic statistics (Xie et al., 2020; Tang et al., 2021; 2023b;a). These methods motivate our evaluation of adaptive policies against traffic-based detectors. An extended review is provided in Appendix A.

3 Methodology

AdaDoS combines a two-stage RL policy with reciprocal learning to address adaptive decision-making and limited observability.

3.1 Threat Model

We consider an SDN equipped with a traffic detector. A malicious host can generate traffic with controlled timing, rate, and duration, but has no network-administration privileges. The attacker knows the victim address and can measure end-to-end delay, while the network topology and configuration are unknown. Bottleneck capacity may be estimated using network measurement tools (Melander et al., 2000).

The objectives are to avoid detection, reduce the bandwidth available to legitimate traffic, and limit resource consumption. Full network measurements are used by the teacher during training; the

deployed student policy uses only delay observations. The training reward additionally uses network and detector feedback, as defined below.

3.2 AdaDoS Framework

AdaDoS interacts with the SDN and its detector to update an attack policy. A two-stage actor separates the decision to attack from the selection of attack rate and duration. Reciprocal learning transfers knowledge from a pre-trained teacher with global observations to a student with a shorter delay-observation window.

End-to-end delay provides an indirect description of network conditions, consistent with the motivation of network tomography (He et al., 2021). The student therefore avoids requiring controller-level observations as policy inputs.

3.3 Adversarial Reinforcement Learning

State. The fully observed state combines traffic statistics, instantaneous network measurements, and recent delays:

$$\mathbf{S} = [\mathbf{s}_{\text{gram}}, \mathbf{s}_{\text{traffic}}, \mathbf{s}_{\text{delay}}], \tag{1}$$

where $\mathbf{s}_{\text{gram}} = [m_1, m_2, m_3]$ contains the GASF statistics defined in Appendix B.2, $\mathbf{s}_{\text{traffic}} = [v_{\text{tcp}}, v_{\text{udp}}, b]$ contains TCP and UDP traffic volumes and available bandwidth, and $\mathbf{s}_{\text{delay}} = [\tau(t-n+1), \ldots, \tau(t)]$ contains the most recent n delay measurements.

Action. Each action comprises an attack decision, rate, and duration:

$$\mathbf{a} = [a_{\text{dec}}, a_{\text{rate}}, a_{\text{dur}}], \qquad a_{\text{dec}} \in \{0, 1\}, \quad a_{\text{rate}}, a_{\text{dur}} \geq 0. \tag{2}$$

The transmitted volume is $v_A = a_{\text{dec}} a_{\text{dur}} a_{\text{rate}}$.

Reward. The reward combines detection avoidance, congestion, and cost:

$$\mathcal{R}(\mathbf{a}, \mathbf{S}, f_d) = \begin{cases} (z - c^2) R_c, & a_{\text{dec}} = 1,\ f_d = 0, \\ -P, & a_{\text{dec}} = 1,\ f_d = 1, \\ -\kappa, & a_{\text{dec}} = 0,\ f_d = 0, \\ -P - \kappa, & a_{\text{dec}} = 0,\ f_d = 1. \end{cases} \tag{3}$$

Here, f_d is the detection flag, R_c scales the congestion reward, P penalises detection, and κ penalises inactivity. The clipped congestion measure z and relative cost c are

$$z = \begin{cases} 1 - b/B_{\max}, & b > B_{\text{th}}, \\ z_0, & b \leq B_{\text{th}}, \end{cases} \qquad c = \frac{a_{\text{dur}} a_{\text{rate}}}{C_{\max}}. \tag{4}$$

$B_{\max}$ denotes link capacity, B_{th} the bandwidth threshold, z_0 the congestion value below that threshold, and $C_{\max}$ the maximum per-attack cost. Detection is penalised regardless of the current attack decision.

Two-stage policy. We use Proximal Policy Optimisation (PPO) (Schulman et al., 2017) with an actor–critic architecture (Mnih et al., 2016). The actor consists of a *decider*, which determines whether to attack, and a *shaper*, which selects the rate and duration when an attack is triggered. The critic estimates the state value. This sequential structure separates timing from traffic shaping (Nachum et al., 2018), as illustrated in Figure 1.

Decoupled training and delayed feedback. The decider is trained using subsequent rewards, whereas the shaper's reward estimate includes only decisions for which $a_{\text{dec}} = 1$. To account for delayed network effects, a first-in-first-out queue stores network measurements and detector feedback before they are used for policy updates.

3.4 Reciprocal Learning

Partial observation. The student operates under partial observability and receives only a recent delay window:

$$\mathbf{O}_S = [\tau(t - n' + 1), \ldots, \tau(t)], \qquad \mathbf{O}_S \in \mathbb{R}^{n'}. \tag{5}$$

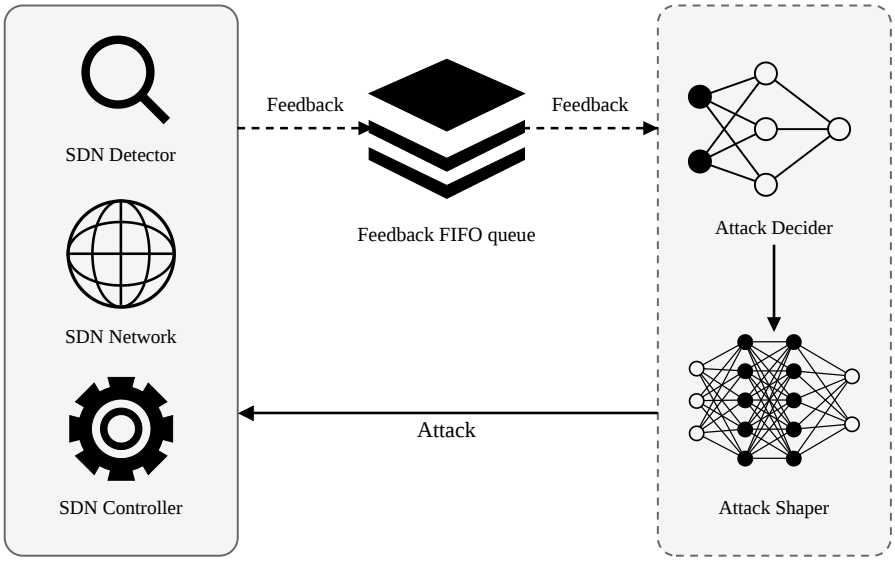


Figure 1: AdaDoS adversarial reinforcement learning framework.

The teacher observes $\mathbf{O}_T = \mathbf{S}$ from Equation 1; the student's delay window n' is typically smaller than the teacher's window n. Both agents use the action space and environmental reward defined above.

Reward-based policy transfer. A pre-trained teacher guides the student through reward shaping (Yin & Pan, 2017; Brys et al., 2015):

$$r_S^{\text{rec}} = r_S - \lambda_r g(\pi_S^{\theta_S}, \pi_T^{\theta_T}, r_S, r_T, \mathbf{O}_T, \mathbf{O}_S), \tag{6}$$

where r_S and r_T are the environmental rewards, $\pi_S^{\theta_S}$ and $\pi_T^{\theta_T}$ are the student and teacher policies, and λ_r controls transfer strength. The transfer function g uses the reward difference and policy KL divergence; its branches are specified in Algorithm 1.

Reciprocal updates. When the teacher obtains a higher reward, the student's reward is adjusted to encourage policy alignment. When the student performs better, its reward is increased and the teacher receives a policy-alignment adjustment. The teacher's environmental reward is scaled by k, allowing continued adaptation during reciprocal learning. Figure 2 summarises this process, and Appendix C provides the complete algorithm and policy-divergence discussion.

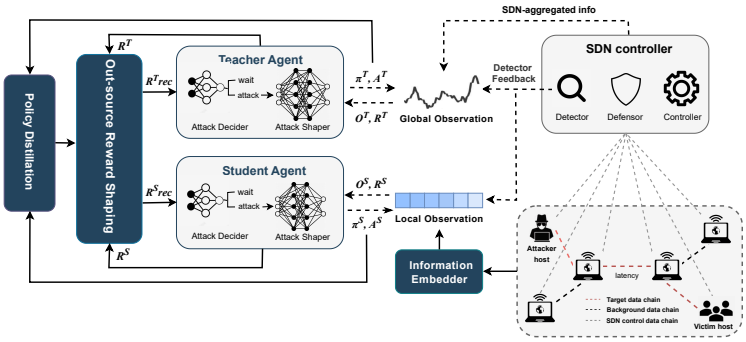


Figure 2: Reciprocal learning between the fully informed teacher and the partially informed student.

4 Theoretical Results

In AdaDoS, the teacher observes the full state $S \in \mathcal{S}$, while the student only observes a local end-to-end delay window $O_S \in \Omega$. Let $\pi_T(a\|S)$ be the teacher's policy and $\pi_S(a\|O_S)$ be the student's policy.

Theorem 1 (Upper Bound on Performance Discrepancy under Partial Observability)**.** *Let $J(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R(S_t, A_t)\right]$ be the expected discounted cumulative return of a policy π. If the expected KL divergence between the teacher's policy and the student's policy across states is bounded by ϵ_{KL}, i.e.,*

$$\mathbb{E}_{S\sim\rho_{\pi_T}}\left[D_{KL}(\pi_T(\cdot\|S)\parallel\pi_S(\cdot\|O_S))\right]\leq\epsilon_{KL}, \tag{7}$$

then the performance gap between the teacher and the student is bounded by the following upper bound.

$$\|J(\pi_T)-J(\pi_S)\|\leq\frac{R_{\max}}{(1-\gamma)^2}\sqrt{2\epsilon_{KL}}, \tag{8}$$

where $R_{\max}=\max_{a,S}\|R(a,S,f_d)\|$, $\gamma\in(0,1)$ is the discount factor, and ρ_{π_T} is the state-visitation distribution under π_T.

The reciprocal learning module modifies the student's environmental reward r_S using a transfer function g based on the reward discrepancy and KL divergence:

$$r^S_{\text{rec}}=r_S-\lambda_r g(\pi_S,\pi_T,r_S,r_T,O_T,O_S). \tag{9}$$

Theorem 2 (Convergence of Reciprocal Learning Policy Distillation Objective)**.** *Assume that the policy family $\pi_\theta(a\|O)$ is parameterized by $\theta\in\Theta$ and the policy optimization uses a convex Surrogate Loss function $\mathcal{L}(\theta)=\mathbb{E}_\rho\left[-r^S_{rec}\log\pi_\theta(a\|O_S)\right]$. Under L-Lipschitz smoothness of $\nabla_\theta\mathcal{L}(\theta)$ and step-size $\alpha_t=\frac{1}{L}$, gradient descent updates on θ, given by $\theta_{t+1}=\theta_t-\alpha_t\nabla_\theta\mathcal{L}(\theta_t)$, converge to a ϵ-stationary point $\left(\|\nabla_\theta\mathcal{L}(\theta_T)\|^2\leq\epsilon\right)$ within $T=O(1/\epsilon)$ iterations.*

The proofs are given in Appendix D.

5 Experiments and Evaluation

5.1 Experimental Setup

In line with most research papers on DoS-like attack and defence, our experimental setup utilises a Ryu controller to emulate the control plane of a Software-Defined Network (SDN) and employs Mininet to simulate the corresponding data plane. This paper replays the WIDE dataset as background traffic for experiments. We also use two evaluation metrics to evaluate the AdaDoS attack performance, namely, **attack success rate** and **available bandwidth**. Both metrics are the average values for each episode. The higher the attack success rate, the better the concealment of the attack; The lower available bandwidth, the higher the network congestion caused by the attack. Additionally, we utilise **attack cost** to measure the cost of the attack.

The complete experimental setup, including hardware and software configurations, hyperparameters, dataset processing, and metric definitions, is provided in Appendix E.

5.2 Attack Performance

The Attack Capability of AdaDoS. To verify the attack capability of AdaDoS, we conducted 10 groups of LDoS attacks with different parameters, and compared their attack success rate and bandwidth with AdaDoS. Each group of LDoS attacks lasts 10000 seconds to ensure that we get enough experimental data. We use LDoS as the baseline for two main reasons: (1) No other work has shown significantly better performance than LDoS in DoS-like attack area, and (2) most recent, high-quality research papers (Tang et al., 2023b; 2021) use LDoS as the baseline for both attack and defence purposes. Given the critical and well-recognised threat posed by LDoS attacks, our comprehensive analysis within this context is sufficient to validate the effectiveness of AdaDoS. As such, comparisons with a broader range of attack types are not necessary for establishing the primary contributions of our work.

According to most recent related works (Tang et al., 2023b; 2021), we use a simple topology as the network topology, as shown in Figure 3. Among them, host 1 sends the WIDE dataset background traffic to host 3, and the malicious host sends the attack traffic to the victim host. Host 2 and host 4 use $iperf$ to measure the real-time TCP bandwidth of bottleneck links.

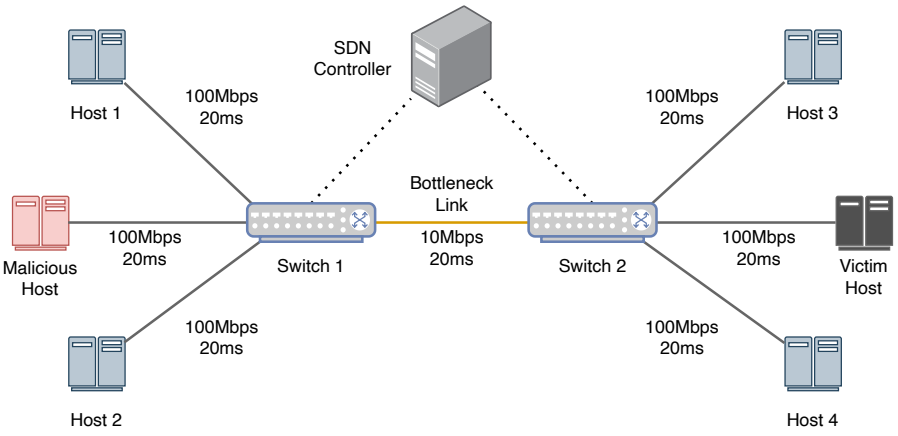


Figure 3: Simple topology

Table 1 presents the performance comparison between LDoS and AdaDoS attack. Compared to LDoS, AdaDoS not only achieves a higher attack success rate but also significantly increases network congestion. When detected by detector, each group of LDoS attacks had a very low attack success rate, whereas AdaDoS skilfully evades the detection and achieves higher bandwidth congestion. Although our algorithm incurs a higher attack cost compared to the baseline, this trade-off is justified in the context of real-world applications. In practice, the availability of additional resources often outweighs the challenge of increasing the attack success rate. Furthermore, it is important to note that our algorithm's cost is substantially lower than that of a normal DoS attack. Thus, the enhanced effectiveness and efficiency of our algorithm make it a superior choice despite the higher cost, as the ultimate goal is to achieve more reliable and impactful attacks while maintaining a reasonable resource expenditure.

AdaDoS Robustness with Various SDN Detectors. To verify the robustness of AdaDoS attacks under various detectors, we deployed different detectors to detect AdaDoS attacks. With the global traffic monitoring capabilities of the SDN controller, we can collect and analyse the data of attack traffic, label the data, and input it into the detector for training. Based on the experimental data of LDoS attacks, we can train an LDoS detector. Based on the traffic data of AdaDoS attacks, we can train an AdaDoS detector to detect AdaDoS attacks under existing strategies. Finally, we mix the LDoS attack data and AdaDoS attack data, or use a mix detector with more comprehensive detection capabilities. This group of experiments is conducted on a simple topology. Similarly, host 1 sends WIDE dataset background traffic to host 3, and malicious host sends attack traffic to the victim host. Host 2 uses $iperf$ tool to measure the TCP bandwidth of bottleneck links between host 2 and host 4.

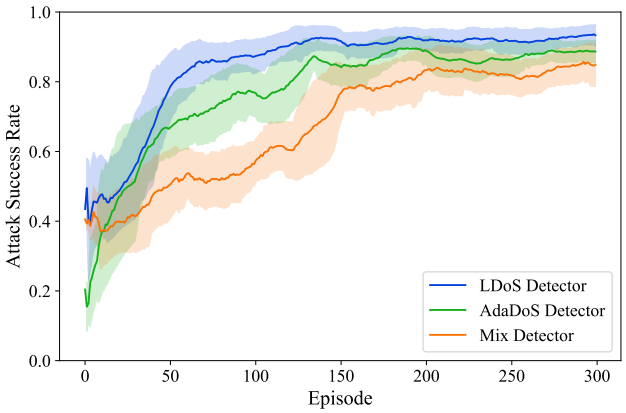


Figure 4: AdaDoS against different detectors

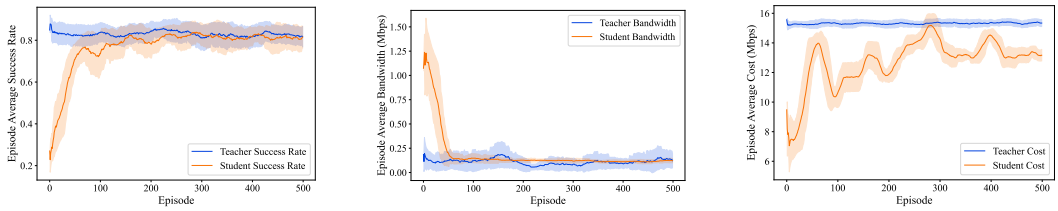
Figure 4 displays the curves for the attack success rates of the AdaDoS attack across three different detectors. It is evident that the AdaDoS attack ultimately achieves a higher success rate under each detector. Initially, a notable observation is the lower success rate of the AdaDoS attack against the AdaDoS detector, which is specifically trained with data from previous AdaDoS attacks. Despite this initial challenge, the AdaDoS attack demonstrates remarkable adaptability; subsequent training sessions significantly enhance its attack strategies. This evolution underscores the AdaDoS attack's sophisticated capability to continually refine and evolve its methods in response to defensive measures.

AdaDoS Robustness with Various Network Topologies. To validate the robustness of AdaDoS against complex network topologies in real-world scenarios, we conducted experiments using three network topologies: Aarnet, Ansnet, and Yorknet. Among three topologies, the Aarnet and Yorknet topologies are large-scale network topologies, while the Ansnet is a medium-scale network topology. The selection of these three topologies is intended to demonstrate the scalability of our method with respect to network topologies with different scale and complexity, and to provide a comprehensive comparison of the performance of LDoS and AdaDoS in different network environments. We respectively executed three sets of LDoS attack experiments in these topologies using various parameters as baselines, and each group of experiments also lasted 10000 seconds. All detailed of these topologies are shown in Appendix.

We directly deployed the attacker model trained on the simple topology shown in Figure 3 on these three complex networks. Table 2 presents the attack performance for LDoS and AdaDoS in various topologies. AdaDoS outperforms LDoS in terms of attack success rate while maintaining a significantly higher bandwidth congestion rate in all three topologies. The trigger rate varies in topologies with different complexity. This indicates that AdaDoS can adapt to diverse network topology and adjust its strategies to achieve optimal attack outcomes.

5.3 Reciprocal Learning Performance

To validate the effectiveness of the reciprocal learning mechanism, we test our reciprocal learning framework performance on the simple topology with WIDE dataset as background traffic, and compare the teacher and student performance within 500 episodes. Figure 5 illustrates the training curves of teacher and student model on three metrics. The teacher model is pre-trained and updated only with reciprocal learning mechanism, while the student model is updated with both reciprocal learning and environment feedback. As shown by the figure 5, the student model achieve about the same attack success rate and bandwidth congestion compared to the teacher, and even outperform the teacher model on the metric of attack average cost. The experiment demonstrates the potential of our reciprocal learning framework, which enable the student attacker with limited observation to reach the similar attack performance of teacher model with a complete observation.



(a) Attack Success Rate (b) Available Bandwidth (c) Attack Cost

Figure 5: Reciprocal Learning Performance

5.4 Noise Disturbance Influence

As our method relies on observed delay information, introducing noise can significantly impact on the AdaDoS's performance. To assess the noise sensitivity of AdaDoS, we added three sets of Gaussian noise to the delay information observed by the agent in the environment of a simple topology, maintaining the same settings as in Section 5.2. We then compared the attack performance with and without noise. Fig 6 displays the attack success rate (ASR) of AdaDoS under three sets of Gaussian noise. The depicted curves show the trained

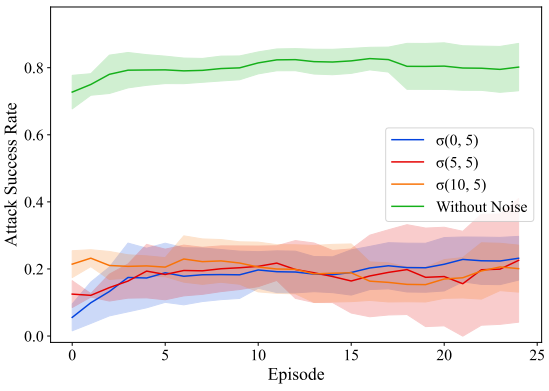


Figure 6: Noise disturbance on AdaDoS

attack agent's ASR under various levels of delay observation noise. The substantial reduction in ASR between experiments with and without noise highlights AdaDoS's vulnerability to delay information disturbance. This finding suggests that adding noise to delay information can be an effective countermeasure against AdaDoS.

Despite this vulnerability, our attack approach remains valuable for demonstrating the importance of accurate delay information in SDN environments and for developing more robust SDN defence mechanisms. It is important to note, however, that adding noise to delay information may have side effects, such as degrading the overall network performance and potentially affecting legitimate traffic. Therefore, while noise addition can serve as a countermeasure, it must be carefully calibrated to mitigate these adverse effects. By understanding AdaDoS's sensitivity to noise, researchers can better appreciate the trade-offs involved in SDN security and work towards more comprehensive solutions that balance attack effectiveness and defence strategies.

Table 1: The attack capability of AdaDoS and the LDoS baselines with different parameters. For the attack success rate and the average bandwidth index, the mean value is indicated on the left of the sign, while the standard deviation is shown on the right. '/' indicates the value is not applicable. The duration random is selected from $\{0.1, 0.15, 0.2\}$, the period random is selected from $\{1.0, 1.5, 2.0\}$, and the attack rate random is selected from $\{15, 20, 25, 30\}$.

Type	Duration (s)	Period (s)	Attack Rate (Mbps)	Trigger rate	Attack Success Rate	Average Bandwidth (Mbps)	Attack Cost (Mbps)
LDoS	0.15	1.0	15	1.00	0.254 ± 0.039	5.3 ± 0.044	2.25
	0.15	1.5	15	1.00	0.257 ± 0.044	5.4 ± 0.122	1.50
	0.20	1.5	15	1.00	0.262 ± 0.044	5.4 ± 0.126	2.00
	0.15	1.0	20	1.00	0.237 ± 0.036	5.3 ± 0.195	3.00
	0.15	1.5	20	1.00	0.255 ± 0.044	5.4 ± 0.148	2.00
	0.20	1.5	20	1.00	0.269 ± 0.045	5.4 ± 0.134	2.67
	0.15	2.0	25	1.00	0.234 ± 0.050	5.6 ± 0.176	1.88
	0.15	2.0	30	1.00	0.235 ± 0.049	5.6 ± 0.172	2.25
	0.10	2.0	30	1.00	0.225 ± 0.045	5.6 ± 0.134	1.50
Double-LDoS	0.15	1.0	15	1.00	0.150 ± 0.447	0.3 ± 0.056	6.63
	0.15	1.5	20	1.00	0.270 ± 0.054	0.1 ± 0.242	5.81
	0.20	1.5	20	1.00	0.282 ± 0.052	0.1 ± 0.367	8.39
	0.15	1.0	20	1.00	0.302 ± 0.074	0.1 ± 0.313	7.17
	0.15	1.5	15	1.00	0.278 ± 0.077	0.1 ± 0.236	7.64
	0.20	1.5	15	1.00	0.285 ± 0.069	0.1 ± 0.212	8.68
	0.20	1.5	25	1.00	0.311 ± 0.063	0.2 ± 0.326	23.24
	0.15	2.0	30	1.00	0.314 ± 0.064	0.1 ± 0.243	10.57
	0.10	2.0	30	1.00	0.294 ± 0.078	0.2 ± 0.371	26.89
	Random	Random	Random	1.00	0.254 ± 0.004	5.2 ± 0.105	/
AdaDoS	0.55 ± 0.21	1.0	9.00 ± 3.92	0.95 ± 0.02	$\mathbf{0.795 \pm 0.062}$	$\mathbf{0.1 \pm 0.126}$	4.78

Table 2: The robustness of AdaDoS in three extensive topology and the LDoS baseline with different parameters. For the AdaDoS rows, the mean value is indicated on the left of the sign, while the standard deviation is shown on the right. '/' indicates the value is not applicable.

Topology	Type	Duration (s)	Period (s)	Attack Rate (Mbps)	Trigger Rate	Attack Success Rate	Average Bandwidth (Mbps)	Attack Cost (Mbps)
Aarnet	LDoS	0.15	1.0	232.5	1.00	0.370 ± 0.051	99.9 ± 1.55	34.91
		0.20	1.5	310.0	1.00	0.372 ± 0.057	100.4 ± 1.66	41.33
		0.10	2.0	465.0	1.00	0.364 ± 0.062	100.5 ± 2.05	23.32
	AdaDoS	0.46 ± 0.19	1.0	229.5 ± 72.6	0.61 ± 0.09	$\mathbf{0.79 \pm 0.091}$	$\mathbf{53.31 \pm 24.95}$	63.08
Ansnet	LDoS	0.15	1.0	75.0	1.00	0.268 ± 0.063	3.0 ± 51.13	11.34
		0.20	1.5	100.0	1.00	0.426 ± 0.103	0.6 ± 0.94	13.31
		0.10	2.0	150.0	1.00	0.295 ± 0.123	0.4 ± 0.96	7.50
	AdaDoS	0.25 ± 0.24	1.0	76.2 ± 18.5	0.59 ± 0.05	$\mathbf{0.871 \pm 0.062}$	$\mathbf{1.129 \pm 0.54}$	22.51
Yorknet	LDoS	0.15	1.0	225.0	1.00	0.560 ± 0.071	71.8 ± 1.72	33.81
		0.20	1.5	300.0	1.00	0.605 ± 0.075	72.3 ± 2.12	40.04
		0.10	2.0	450.0	1.00	0.603 ± 0.072	71.6 ± 2.62	22.52
	AdaDoS	0.50 ± 0.21	1.0	216.8 ± 63.5	0.21 ± 0.09	$\mathbf{0.795 \pm 0.118}$	$\mathbf{26.32 \pm 9.74}$	59.08

6 DEFENCE SCHEME

As previously mentioned, existing rule-based or machine learning-based detectors struggle to identify AdaDoS, as it adapts its attack patterns in response to detector results. The adaptive capabilities of AdaDoS necessitate innovative defence strategies beyond traditional methods. In this section, we discuss possible countermeasures that network administrators can be used to mitigate the attack.

Disturbing Attacker Observaion. One inherent limitation of AdaDoS is its reliance on delay information to orchestrate attacks and it's vulnerability in observation noise, as shown in Fig 6. This dependency offers a strategic point of vulnerability that can be exploited to mitigate the threat. A straightforward defence approach involves deliberately introducing noise into the delay data. This can be achieved by occasionally altering routing paths intentionally or by withholding packages to randomly inject noise into delay information between nodes. Although these tactics can significantly impair the ability of AdaDoS to execute precise attacks, they are not without their drawbacks. Introducing noise or withholding information can disrupt the normal operations of the network, potentially degrading service quality and affecting legitimate network traffic. Therefore, while these strategies offer a viable means of defence against AdaDoS, they also necessitate a careful consideration of the trade-offs between security and network performance.

Adopting Zero-trust Architecture. Adopting a zero-trust architecture is recommended as a fundamental security enhancement. This architecture requires continuous authentication and authorisation of all users, regardless of their position relative to the network perimeter, ensuring rigorous security checks and continuous validation of security configurations before granting or maintaining access to data and network resources. However, overly strict Zero-Trust defenses may inadvertently block innocent traffic; thus, mechanisms like adaptive thresholds, traffic verification feedback loops, and a quarantine mode for further analysis should be integrated to minimize false positives and maintain service availability.

Leveraging Adversarial Trained Detector. A suggested way to boost SDN defense capability is integrating adversarial training into the network's defence mechanisms. This involves training models to recognise and mitigate inputs designed using adversarial attack techniques, potentially enhancing the network's resilience against strategies similar to those used by AdaDoS. However, there is scepticism regarding the effectiveness of this approach. Experiences with generative adversarial networks (GANs) (Goodfellow et al., 2020) have shown that distinguishing between genuine and AI-generated images has become increasingly difficult. This raises concerns about the adaptability of adversarial training, as such models might struggle to differentiate between legitimate network activities and those mimicking AdaDoS patterns.

7 CONCLUSION

This paper introduce AdaDoS, an adaptive adversarial reinforcement learning-based attack framework designed to generate DoS-like malware under partial observation SDN environment. To address the low survival ability problem, AdaDoS employs an innovative deep reinforcement learning model and propose a two-stages structure to dynamically change the attack patterns. To address the limited observation problem from attacker side, AdaDoS propose a reciprocal learning framework with teacher-student structure. Furthermore, we evaluate the performance of AdaDoS against latest detectors via real-world settings, finding that it can bypass these defences. The findings underscore the potential of AdaDoS in the field of attacks in SDN and possible any SDN-like network.

AI USE STATEMENT

During the preparation of this manuscript, we used large language models solely to assist with language editing and improve clarity and readability. These tools were not used to generate research ideas, design the methodology, implement experiments, analyze data, or interpret results. All scientific contributions and findings were developed independently by the authors. The authors reviewed and revised all AI-assisted text to ensure that it accurately reflected the research and take full responsibility for the final content of the paper.

ETHICS STATEMENT

In developing AdaDoS, an advanced adversarial reinforcement learning-based framework for simulating DoS-like attacks in SDN environments, we acknowledge the dual-use nature of this research and the ethical challenges it poses. Our primary objective is to enhance understanding of network vulnerabilities and strengthen defenses against increasingly sophisticated attacks. However, we recognise the potential for misuse of the framework and are committed to addressing this concern responsibly by adhering to ethical principles outlined in relevant resources, including the USENIX Security guidelines.

To mitigate risks, all details regarding the tactics, techniques, and procedures (TTPs) employed in AdaDoS will be shared only with verified entities in the academic and security research communities. Such entities must undergo a rigorous vetting process, including verification of credentials and a formal commitment to ethical use, ensuring that the framework supports defensive efforts without contributing to offensive capabilities. We also emphasise the importance of weighing harms and benefits under both consequentialist and deontological ethics, striving to prevent tangible harms and respect individuals' rights, as recommended by the principles of Beneficence and Respect for Persons.

Furthermore, all experimental activities are conducted exclusively in controlled and isolated environments designed to emulate real-world network configurations without connecting to operational systems. These environments employ virtualized topologies and synthetic traffic while incorporating real-time monitoring and automated safeguards to detect and prevent any anomalous or unintended behaviour. This ensures that research does not inadvertently disrupt real-world systems or compromise the security of external networks.

In addition, we proactively consider the full spectrum of stakeholders potentially impacted by this research, including network operators, end users, and the broader cybersecurity community. By addressing ethical considerations at every stage of the research process and documenting these decisions transparently, we aim to align with global best practices for ethical cybersecurity research. In cases where ethical analyses under different principles lead to conflicting conclusions, we commit to articulating and justifying our decisions clearly, as recommended by USENIX Security guidelines.

Finally, we adhere to all applicable laws and regulations governing cybersecurity research, including disclosure practices for vulnerabilities. Vulnerabilities discovered during this research are disclosed promptly and responsibly to affected parties, in alignment with the principle of minimizing harm and maximizing positive outcomes. This rigorous approach ensures that AdaDoS contributes to advancing the field of network security while upholding the highest ethical standards.

REPRODUCIBILITY STATEMENT

The experimental setup is summarised in Section 5.1. The Appendix E provides detailed configuration and preprocessing information, including Table 3. The network topologies are provided in Appendix F, and the reciprocal learning procedure in Algorithm 1. We plan to release the source code at `https://anonymous.4open.science/r/AdaDoS`.

REFERENCES

AARNet. Australia's academic and research network, 2024. URL `https://www.aarnet.edu.au`. Date of visit: 2024-04-29.

Ijaz Ahmad, Suneth Namal, Mika Ylianttila, and Andrei Gurtov. Security in software defined networks: A survey. *IEEE Communications Surveys & Tutorials*, 17(4):2317–2346, 2015.

Sulaiman Alwabisi, Ridha Ouni, and Kashif Saleem. Using machine learning and software-defined networking to detect and mitigate ddos attacks in fiber-optic networks. *Electronics*, 11(23):4065, 2022.

Tim Brys, Anna Harutyunyan, Matthew E Taylor, and Ann Nowé. Policy transfer using reward shaping. In *Autonomous Agents and Multiagent Systems*, pp. 181–188, 2015.

Jiahao Cao, Mingwei Xu, Qi Li, Kun Sun, Yuan Yang, and Jing Zheng. Disrupting sdn via the data plane: A low-rate flow table overflow attack. In Xiaodong Lin, Ali Ghorbani, Kui Ren, Sencun Zhu, and Aiqing Zhang (eds.), *Security and Privacy in Communication Networks*, pp. 356–376, Cham, 2018. Springer International Publishing. ISBN 978-3-319-78813-5.

Jiahao Cao, Qi Li, Renjie Xie, Kun Sun, Guofei Gu, Mingwei Xu, and Yuan Yang. The CrossPath attack: Disrupting the SDN control channel via shared links. In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 19–36, Santa Clara, CA, August 2019. USENIX Association. ISBN 978-1-939133-06-9. URL https://www.usenix.org/conference/usenixsecurity19/presentation/cao.

Ping Dong, Xiaojiang Du, Hongke Zhang, and Tong Xu. A detection method for a novel ddos attack against sdn controllers by vast new low-traffic flows. In *2016 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2016. doi: 10.1109/ICC.2016.7510992.

Sharon Ezekiel, Dinil Mon Divakaran, and Mohan Gurusamy. Dynamic attack mitigation using sdn. In *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, pp. 1–6, 2017. doi: 10.1109/ATNAC.2017.8215430.

Sean Gillen and Katie Byl. Explicitly encouraging low fractional dimensional trajectories via reinforcement learning. In *Conference on Robot Learning*, pp. 2137–2147. PMLR, 2021.

Kostas Giotis, George Androulidakis, and Vasilis Maglaris. A scalable anomaly detection and mitigation architecture for legacy networks via an openflow middlebox. *Security and Communication Networks*, 9(13):1958–1970, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63:139–144, 2020. ISSN 0001-0782.

Zehua Guo, Ruoyan Liu, Yang Xu, Andrey Gushchin, Anwar Walid, and H. Jonathan Chao. Star: Preventing flow-table overflow in software-defined networks. *Computer Networks*, 125(oct.9): 15–25, 2017.

Ting He, Liang Ma, Ananthram Swami, and Don Towsley. *Network tomography: identifiability, measurement design, and network state inference*. Cambridge University Press, 2021.

Shihwen Hsu, Tsengyi Chen, Yungchun Chang, Shuohan Chen, Hanchieh Chao, Tsenyeh Lin, and Weikuan Shih. Design a hash-based control mechanism in vswitch for software-defined networking environment. In *2015 IEEE International Conference on Cluster Computing*, pp. 498–499, Chicago, IL, USA, 2015. IEEE Computer Society.

Zhiyuan Hu, Mingwen Wang, Xueqiang Yan, Yueming Yin, and Zhigang Luo. A comprehensive security architecture for sdn. In *2015 18th International Conference on Intelligence in Next Generation Networks*, pp. 30–37, 2015. doi: 10.1109/ICIN.2015.7073803.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the nineteenth international conference on machine learning*, pp. 267–274, 2002.

Simon Knight, Hung X. Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011. doi: 10.1109/JSAC.2011.111002.

Ezgi Korkmaz. A survey analyzing generalization in deep reinforcement learning. *arXiv preprint arXiv:2401.02349*, 2024.

Xinmeng Li, Kai Zheng, Dan Tang, Zheng Qin, Zhiqing Zheng, and Shihan Zhang. Ldos attack detection based on asnnc-ofa algorithm. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 2021. doi: 10.1109/WCNC49053.2021.9417400.

Bob Melander, Mats Bjorkman, and Per Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In *Globecom'00-IEEE. Global Telecommunications Conference. Conference Record (Cat. No. 00CH37137)*, volume 1, pp. 415–420. IEEE, 2000.

Nisharani Meti, D G Narayan, and V. P. Baligar. Detection of distributed denial of service attacks using machine learning algorithms in software defined networks. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1366–1371, 2017. doi: 10.1109/ICACCI.2017.8126031.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016. URL https://api.semanticscholar.org/CorpusID:6875312.

Mayadah A Mohsin and Ali H Hamad. Performance evaluation of sdn ddos attack detection and mitigation based random forest and k-nearest neighbors machine learning algorithms. *Revue d'Intelligence Artificielle*, 36(2):233, 2022.

Ofir Nachum, Shixiang (Shane) Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/e6384711491713d29bc63fc5eeb5ba4f-Paper.pdf.

Tulio A Pascoal, Iguatemi E Fonseca, and Vivek Nigam. Slow denial-of-service attacks on software defined networks. *Computer Networks*, 173:107223, 2020.

Fabian Pedregosa. Scikit-learn: Machine learning in python, 2011. URL https://cir.nii.ac.jp/crid/1370005891170856713.

Trung V. Phan, Nguyen Khac Bao, and Minho Park. A novel hybrid flow-based handler with ddos attacks in software-defined networking. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pp. 350–357, 2016. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0069.

Aditya Prakash and Rojalina Priyadarshini. An intelligent software defined network controller for preventing distributed denial of service attack. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pp. 585–589. IEEE, 2018.

M Revathi, V V Ramalingam, and B Amutha. A machine learning based detection and mitigation of the ddos attack by using sdn controller framework. *Wireless Personal Communications*, pp. 1–25, 2022. ISSN 0929-6212.

Danish Sattar, Ashraf Matrawy, and Olufemi Adeojo. Adaptive bubble burst (abb): Mitigating ddos attacks in software-defined networks. In *2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks)*, pp. 50–55. IEEE, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Gao Shang, Peng Zhe, Xiao Bin, Hu Aiqun, and Ren Kui. Flooddefender: Protecting data and control plane resources under sdn-aimed dos attacks. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pp. 1–9. IEEE, 2017.

Sajad Shirali-Shahreza and Yashar Ganjali. Efficient implementation of security applications in openflow controller with flexam. In *2013 IEEE 21st annual symposium on high-performance interconnects*, pp. 49–54. IEEE, 2013.

Dan Tang, Yudong Yan, Siqi Zhang, Jingwen Chen, and Zheng Qin. Performance and features: Mitigating the low-rate tcp-targeted dos attack via sdn. *IEEE Journal on Selected Areas in Communications*, 40(1):428–444, 2021.

Dan Tang, Jingwen Chen, Xiyin Wang, Siqi Zhang, and Yudong Yan. A new detection method for ldos attacks based on data mining. *Future Generation Computer Systems*, 128:73–87, 2022.

Dan Tang, Chenjun Gao, Xinmeng Li, Wei Liang, Sheng Xiao, and Qiuwei Yang. A detection and mitigation scheme of ldos attacks via sdn based on the fss-rsr algorithm. *IEEE Transactions on Network Science and Engineering*, 2023a.

Dan Tang, Siyuan Wang, Boru Liu, Wenqiang Jin, and Jiliang Zhang. GASF-IPP: Detection and mitigation of LDoS attack in SDN. *IEEE Transactions on Services Computing*, 16(5):3373–3384, 2023b. doi: 10.1109/TSC.2023.3266757.

Roshni Mary Thomas and Divya James. Ddos detection and denial using third party application in sdn. In *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pp. 3892–3897. IEEE, 2017.

Haopei Wang, Lei Xu, and Guofei Gu. Floodguard: A dos attack prevention extension in software-defined networks. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 239–250. IEEE, 2015.

Mingxin Wang, Huachun Zhou, Jia Chen, and Bo Tong. An approach for protecting the openflow switch from the saturation attack. In *Proceedings of the 2015 4th National Conference on Electrical, Electronics and Computer Engineering*, pp. 729–734. Atlantis Press, 2015/12. ISBN 978-94-6252-150-6. doi: 10.2991/nceece-15.2016.135. URL https://doi.org/10.2991/nceece-15.2016.135.

Zhiguang Wang and Tim Oates. Imaging time-series to improve classification and imputation, 2015.

WIDE Project. Samplepoint-f 2018-01-01 14:00 traffic data. https://mawi.wide.ad.jp/mawi/samplepoint-F/2018/201801011400.html, 2018.

WIDE Project. MAWI working group traffic archive. https://mawi.wide.ad.jp/mawi/, n.d.

Zhijun Wu, Qingbo Pan, Meng Yue, and Liang Liu. Sequence alignment detection of tcp-targeted synchronous low-rate dos attacks. *Computer Networks*, 152:64–77, 2019.

Yang Xiang, Ke Li, and Wanlei Zhou. Low-rate ddos attacks detection and traceback by using new information metrics. *IEEE Transactions on Information Forensics and Security*, 6(2):426–437, 2011. doi: 10.1109/TIFS.2011.2107320.

Shengxu Xie, Changyou Xing, Guomin Zhang, Xianglin Wei, and Guyu Hu. Research on ldos attack detection and defense mechanism in software defined networks. In *Security and Privacy in Social Networks and Big Data: 6th International Symposium, SocialSec 2020, Tianjin, China, September 26–27, 2020, Proceedings 6*, pp. 85–96. Springer, 2020.

Haiyan Yin and Sinno Jialin Pan. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pp. 1640–1646. AAAI Press, 2017.

Bin Yuan, Deqing Zou, Shui Yu, Hai Jin, Weizhong Qiang, and Jinan Shen. Defending against flow table overloading attack in software-defined networks. *Services Computing, IEEE Transactions on*, 12(2):231–246, 2019.

Gongming Zhao, Hongli Xu, Shigang Chen, Liusheng Huang, and Pengzhan Wang. Joint optimization of flow table and group table for default paths in sdns. *IEEE/ACM Transactions on Networking*, PP(4):1–14, 2018.

A Extended Related Work

In this section, we review recent advances in the study of DoS-like attacks and the corresponding defense mechanisms within SDN.

A.1 DoS-like Attacks in SDN

DDoS attacks overwhelm control and data plane resources by generating numerous fake data streams, causing flow table misses in OpenFlow switches and overloading controllers, thereby disrupting legitimate host communications (Shang et al., 2017). Rapid buffer filling due to these streams intensifies control plane bandwidth strain and delays rule installations. Overflow of the forwarding table (the packet forwarding rule repository managed by the controller) (Ahmad et al., 2015) in OpenFlow switches can also cause issues. As numerous new forwarding rules pile up, table saturation ensues, preventing the switch from processing further rules from the controller, culminating in packet discards, error reports, and forwarding latency/dropouts until table memory frees up (Wang et al., 2015).

Although DDoS attacks are renowned for their widespread impact and intense traffic deluge, LDoS attacks demonstrate a more covert and equally destructive approach, achieving their goals by precisely controlling the timing and intensity of the attack. Cao et al. (2019) revealed that exploiting shared links between control and data traffic disrupts SDN control channels, degrading app performance and causing severe network anomalies. Data plane attacks encompass flow table and TCP congestion control based ones. Cao et al. (2018) introduced LOFT, calculating minimum attack rates to induce flow table overflow efficiently. Pascoal et al. (2020) detailed two LDoS tactics: one exhausts TCAM memory by continuously installing uncertain active rules, inhibiting legit rule setup; the other merges slow TCAM with low-rate saturation attacks, using minimal resources to deny service and potentially affecting existing legit flow rules more severely.

A.2 DoS-like Attack Defences in SDN

SDN solutions to address flow table overflow involve caching unmatched flows in OpenFlow switches' cache (Wang et al., 2015/12), QoS-aware strategies (Yuan et al., 2019), and online routing schemes (Guo et al., 2017) that optimise flow table usage. Zhao et al. (2018) optimised group/flow tables using aggregation rules. To safeguard against DDoS, Hsu et al. (2015) suggested hashing for queue allocation and scheduling. Sattar et al. (2016) proposed ABB for dynamic resource allocation under attacks. SDN frameworks like FleXam (Shirali-Shahreza & Ganjali, 2013) and FloodGuard (Wang et al., 2015) provide packet-level control and protection against DoS. Hu et al. (2015) proposed a comprehensive security architecture. Other solutions employ statistical analysis to secure SDN controllers. Giotis et al. (2016) use edge switches to detect anomalies. The method proposed by Dong et al. (2016) classifies flows based on stats to spot low-traffic ones impacting control planes. Thomas & James (2017) leverage third-party apps for filtering and DDoS detection. ML-based approaches train on non-attack flows: Meti et al. (2017) and Ezekiel et al. (2017) use SVM and NN for DDoS detection, while Prakash & Priyadarshini (2018) apply KNN. Phan et al. (2016) combined SVM and SOM in SVMs-SOM for enhanced traffic classification and DDoS prevention.

Defence methods against LDoS attacks in SDN mainly fall into anomaly and feature detection. Anomaly detection, exemplified by ASNNC-OFA (Li et al., 2021), involves clustering traffic characteristics and using outlier analysis to flag abnormalities compared to typical networks. Tang et al. (2022) and Xiang et al. (2011) devised entropy and info-distance metrics to distinguish attack from legitimate traffic, while Wu et al. (2019) applied Smith-Waterman alignment to identify LDoS by comparing detection sequences' pulse attributes against background flows. Feature detection revolves around extracting unique identifiers such as LDoS periodic TCP/UDP pulses. ADAR, proposed by Xie et al. (2020), detects and mitigates LDoS attacks targeting SDN switch buffers. Tang et al. (2021) developed the P&F framework categorising traffic features into performance and attribute types. P&F employs ML to assess if LDoS impacts normal traffic and uses time-frequency analysis to pinpoint attackers and victims. Their GASF-IPP framework analyses traffic with GASF transformations and UDP stream bursts within a sliding window to detect attacks (Tang et al., 2023b). The FSS-RSR-based scheme extracts traffic features, employs FSS-RSR for real-time LDoS detection, monitors traffic fluctuations to locate attacker IPs, and recommends rules to mitigate attacks (Tang et al., 2023a).

B BACKGROUND

In this section, we introduce the latest detectors for Low-rate Denial of Service (LDoS) attacks within SDN environments, and the state-of-the-art in feature engineering for LDoS attack detection

B.1 LDOS ATTACK

LDoS attack is a specially designed network attack pattern that disrupts the availability of target services through periodic bursts of traffic. Unlike traditional sustained high traffic DoS attacks, the key features of LDoS attacks are their low-frequency, high-intensity traffic packets, and the interval time between attack activities. These three parameters - the duration of the attack, the attack rate, and the period between two attacks - collectively determine the effectiveness and concealment of LDoS attacks. A short duration of the attack and a long interval make these attacks harder to detect with standard traffic monitoring tools. Therefore, LDoS attacks can effectively consume the processing power and bandwidth of the target network, especially for critical applications that require stable connections, such as the TCP protocol. Even brief network delays can cause performance problems or service interruptions.

B.2 DETECTOR

In SDN environments, effectively detecting LDoS attacks is crucial due to their subtle nature, which can evade conventional detection systems. Typically, SDN-based security frameworks monitor global data plane traffic through the control plane to identify, filter, and disrupt malicious flows based on distinct traffic patterns. However, the stealthy characteristics of LDoS attacks, which involve sending intermittent bursts of traffic to disrupt service without triggering standard thresholds, demand more sophisticated detection methods.

A promising approach for enhancing the detection of LDoS attacks involves the use of the Gramian Angular Summation Field (GASF), a method originally proposed by Wang & Oates (2015) for processing one-dimensional time series data. Utilising the principles of the Gram matrix, GASF excels in capturing the intricate temporal characteristics of TCP/UDP traffic flows, which is vital for recognising the anomalous patterns indicative of LDoS attacks. The strength of GASF lies in its capability to analyse both the elemental properties and the interrelationships within the data sequences, providing a granular and nuanced analysis that is well-suited for identifying the subtle fluctuations characteristic of DoS-like traffic.

Let $X(t)$ represents the traffic data reading at time t Assuming a sliding window contains l traffic data, as shown in equation 10.

$$W(t) = \{X(t-l+1), X(t-l+2), \ldots, X(t)\}. \tag{10}$$

We subsequently utilise the MinMaxScaler (Pedregosa, 2011) to normalise the stream sequence to a range between -1 and 1. The normalisation formula is provided in Equation 11.

$$X'(t) = \frac{(X(t) - X_{\text{min}}) \times (1 - (-1))}{X_{\text{max}} - X_{\text{min}}} + (-1), \tag{11}$$

where $X(t)$ represents the original data at time t, X_{min} and X_{max} are the minimum and maximum values in the sliding window $W(t)$. This structured approach ensures the data is scaled appropriately, keeping the integrity and relational properties of the original data.

After normalising the stream sequence to a range between -1 and 1 using Equation 11, each data point $X'(t)$ is transformed into its corresponding polar coordinate angle $\phi(t)$. This transformation is accomplished by calculating the arc-cosine of $X'(t)$, as depicted in Equation 12. Calculating the polar angle is pivotal as it retains the numerical relationships among the sequence data points. This preservation is essential for ensuring the structural integrity of the data, which supports further analytical processes where the geometric or angular relationships between data points are crucial.

$$\phi(t) = \arccos(X'(t)). \tag{12}$$

GASF defines a specialised inner product: $\langle X'(t_1), X'(t_2)\rangle = \cos(\phi(t_1)+\phi(t_2))$, which means that the inner product between two stream points is the cosine of the sum of the polar angles converted from the polar coordinates of these two stream points, in the form of:

$$G = \begin{pmatrix} \cos(\phi_1+\phi_1) & \cos(\phi_1+\phi_2) & \cdots & \cos(\phi_1+\phi_n) \\ \cos(\phi_2+\phi_1) & \cos(\phi_2+\phi_2) & \cdots & \cos(\phi_2+\phi_n) \\ \vdots & \vdots & \cdots & \vdots \\ \cos(\phi_n+\phi_1) & \cos(\phi_n+\phi_2) & \cdots & \cos(\phi_n+\phi_n) \end{pmatrix}. \tag{13}$$

Following Tang et al. (2023b), we calculate the first, second, and third moments of the Gram matrix as features of the flow sequence:

$$\begin{aligned} m_1 &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} G_{i,j} \\ m_2 &= \left(\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(G_{i,j}-m_1)^2\right)^{\frac{1}{2}} \\ m_3 &= \left(\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(G_{i,j}-m_1)^3\right)^{\frac{1}{3}}, \end{aligned} \tag{14}$$

where N denotes the number of polar coordinates, and $G_{i,j}$ stands for the elements of matrix G, and m_1, m_2 and m_3 refer to the first, second, and third moments of matrix G, respectively.

Due to the capabilities of the SDN architecture, the controller is capable of aggregating global traffic data from the data plane and collecting traffic across various network protocols, including TCP and UDP streams. This functionality not only allows for the analysis of global traffic characteristics but also enables the extraction of specific features from TCP and UDP streams.

In the initial phases of a DoS-like attack, observable changes occur in the network traffic parameters: the mean of TCP flows decreases, the variance increases, and the coefficient of variation undergoes significant alterations. Due to the connectionless nature of UDP, attackers frequently exploit this protocol to inject malicious traffic into transport layer bottlenecks. As a result, we utilise the coefficient of variation of TCP streams and TCP/UDP traffic volume as discriminate features.

To summarise, the detection system captures the first, second, and third moments of the Gram matrix of global traffic, along with the coefficient of variation of TCP flows, and the number of abnormal peaks in UDP flows, to characterise network traffic. These five features are subsequently inputted into pre-trained machine learning classifiers, including GBDT, KNN, and XGBoost, to ascertain the presence of DoS-like attacks within the network.

C Supplementary Methodological Details

C.1 Additional Assumptions and Framework Rationale

Threat-model assumptions. While the underlying network infrastructure is assumed to be trusted, malicious hosts within the network may aim to degrade the service performance of the entire network. Additionally, sufficient computational resources are accessible for attacker to deploy attack models to dynamically adjust traffic patterns in real-time based on feedback from the network environment.

Adversarial interaction. The AdaDoS framework primarily comprises two components: the attacker and the detector. Both components are designed to outperform each other, forming a game-theoretic zero-sum game. Our method exploits the adversarial nature of this setup to enhance the attacker model with greater concealment and adaptability.

Observation dimensionality. With lower observation dimensionality and noised latency information, the deployed student attacker is expected to get more insensitive and robust to network environment variation according to Gillen & Byl (2021); Korkmaz (2024).

C.2 Decision-Process Formulation and Reward Rationale

Markov decision process. We first formulate the process of network attack as a Markov Decision Process (MDP). With out loss of generality, the MDP of attack can be described as $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma >$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R}$ is the reward function, and $\gamma \in (0, 1)$ is a discount factor.

The state, action, and reward definitions are provided in Section 3.3.

State representation. Inspired by Tang et al. (2023b), we use the gram-like matrix for feature extraction of traffic flow of SDN in a statistic order, and the real-time traffic volume and real-time bandwidth for the instant observation. To capture the trends of link delay and to align observation with the actual environment, the observation of attacker also contains delay of target link within given window size.

Reward rationale. We also use the threshold-based reward in undetected attack reward, encouraging attacker to achieve the target congestion rate, which is represented in z_0 and B_{th}.

Partially observable decision process. The process of network attack under limited observation is formulated as POMDP. Without loss of generality, the POMDP of attack can be described as a 5-tuple, $< \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{R}, \gamma >$, where $\mathcal{O}$ is the observation space, $\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$ is the the state space, action space and reward function defined before in 3.3, $\gamma \in (0, 1)$ is a discount factor in MDP definition.

C.3 Two-Stage Decision and Training Rationale

Two-stage decision mechanism. The attack timing and the attack style is equal important to the survival ability of attacker, since the former avoid the unsuitable timing that should not attack, and the latter improve the concealment of attack. Both of the timing choice and style choice are crucial. In some extreme examples, timing plays a critical role, such as when background traffic is low, a rushed attack is easily detected and the attack host will be blocked. Besides, the concealment built by attack style is based on the right timing choice, as the SDN traffic characteristic is usually variable in different time.

However, the decision of attack timing and the shape of attack style is in different granularity level. The attack timing choice is usually based on long-term planning, while the attack style shaping works mainly on short-term decision. To better generate policy of the long and short term policy, We model this process as a two-stage sequential process, and utilise a sequential decision model to execute this process.

The divided two part of actor take action in a sequential style, and update in a decoupled style in their own action space. The divided model works as a hierarchical RL model, which means the action output and expectation evaluation works in a sequential order.

Decoupled training process. As the *decider* model and the *shaper* model determine the timing and the manner of one attack respectively, estimates of their policies' reward expectations should differ. The *decider*'s action affects all subsequent states, and therefore take responsible for all the future results. The calculate of *decider*'s reward expectation should contains all the subsequent rewards. However, as attack *shaper* won't take any action when *decider*'s action a_{dec} is 0, the *shaper*'s reward expectation only contains rewards when *decider*'s action a_{dec} is 1. The Hierarchical update style is paramount for judging the quality of action, as it limits the reward expectation estimate within the scope of action, making the update of the *decider* model and the *shaper* model, more accurate and efficient.

Deferred reward. As the impact of attacks takes time to be felt by the network, the reward should be deferred. With the reward deferred, the model is able to learn the network and detector character precisely with a more accurate feedback, which is much closer real case.

C.4 Policy Transfer and Reciprocal Learning Rationale

Reward-based policy transfer. As the policy is a conditional distribution of observation with parameters, the transfer of policy is equal to the transfer a probability distribution.

Inspired by Yin & Pan (2017) and Brys et al. (2015) and based on reinforcement learning update mechanism, we can consolidate the transfer update on parameter into the rectification of reward and use the method of reward shaping to conduct the transfer process.

The reward rectification is given in Equation 6.

The utilisation of the reward rectification greatly simplify the training process. It also leverage the update module of RL algorithm, improving the stability and efficiency of transfer learning process, contributing to better performance of student model.

Reciprocal learning mechanism. The policy transfer depends on a stable and robust teacher model, whose policy will be distilled to the student model. However, due to the variable environment of SDN, the robustness and consistency of the pre-trained teacher are not guaranteed. We propose the reciprocal learning mechanism to mitigate this problem.

With such a mutual learning paradigm, both of the teacher and student will improve compared to the model before, and the high-quality policy outputted by teacher model will in turn enhance the training of student model. The details of reward rectification is shown in algorithm 1.

Adaptive fine-tune. The robustness and consistency of the pre-trained teacher are also related to the feedback of the SDN environment. To make the teacher model better adapted to a new SDN environment, we propose the Adaptive Fine-tune for the teacher model update. With the adjustment of environment effect on teacher model update, pre-trained model will be able to keep performance and output high-quality policy continuously for the student model.

C.5 Policy Kullback–Leibler Divergence

The A-C model is usually based on stochastic policy model, which is hard to measure diffent policy with few concrete action. To measure difference of the teacher's and student's policy, we introduce the Policy Kullback-Leibler (KL) Divergence, which can directly measure the variance between two distribution, as well as stochastic policy. To better estimate the policy KL divergence in training, We first treat the policy as the joint distribution of state and action, and then convert it as a conditional distribution based on state, which is the actual format of the model outputted policy. We also rewrite the integration process in origin KL divergence as the expectation of the sampling of policy. The policy KL divergence between teacher and student policy is described as follows:

$$\begin{aligned} \mathcal{D}_{\mathrm{KL}}\left(\pi_T||\pi_S\right) &= \int_{\boldsymbol{O}_T, A_T} \pi_T(\boldsymbol{O}_T, A_T) \cdot \ln\left(\frac{\pi_T(\boldsymbol{O}_T, A_T)}{\pi_S(\boldsymbol{O}_T, A_T)}\right) \\ &= E_{(\boldsymbol{O}_T, A_t)\sim\pi_T}\left[\ln\left(\frac{\pi_T(A_T|\boldsymbol{O}_T)}{\pi_S(A_T|\boldsymbol{O}_T)}\right)\right], \end{aligned} \tag{15}$$

where π_S and π_T are the distribution format policy of teacher and student, A_T is the action taken by teacher, $\mathcal{S}_T$ indicates the policy-related teacher state set.

The rewrote policy KL divergence allows us to directly measure the difference of two policy outputted by two different model through policy sampling, and the process of estimation can be conducted during the training process, which proves practical in training.

C.6 The Algorithm of Reciprocal Learning

The Reciprocal Learning algorithm leverages a collaborative learning paradigm between a student and a teacher agent. Both agents operate in their respective environments, E_S for the student and E_T for the teacher. The teacher begins with a pre-trained actor-critic model, while the student learns from scratch. In each iteration, the agents take actions based on their respective policies, observe new states, and receive rewards. The rewards are adjusted via reciprocal learning mechanisms, incorporating a KL-divergence-based regularization to encourage policy alignment. The teacher aids

the student by scaling and modifying the reward function based on performance differences. Both agents update their actor-critic networks using temporal-difference (TD) errors derived from the adjusted rewards with Generalized Advantage Estimation (GAE). This process iteratively improves the student's performance while maintaining collaboration between the agents.

Algorithm 1 Reciprocal learning

Input: Environment of student E_S and Environment of teacher E_T, learning rate of student and teacher critic model α_S, α_T, learning rate of student and teacher actor model $\beta_S, \beta_T \in [0,1]$, transfer rate of student and teacher λ_S, λ_T, reward scale rate $k \in [0,1]$, discount rate $\gamma \in [0,1]$, Pre-trained teacher actor-critic network $\pi_T^{\theta_T}$,$V_T^{\omega_T}$

Output: actor-critic network $\pi_S^{\theta_S}, V_S^{\omega_S}$

Initialise environment E_S, E_T and parameters of student actor-critic network $\pi_S^{\theta_S}, V_S^{\omega_S}$: $\theta_S, \omega_S \in \mathbb{R}^h$

for each iteration **do**
  observe $\boldsymbol{O}_S, \boldsymbol{O}_T$
  while state not terminal **do**
    $a_S \sim \pi_S(\cdot|\boldsymbol{O}_S;\ \theta_S)$
    $a_T \sim \pi_T(\cdot|\boldsymbol{O}_T;\ \theta_T)$
    Take action a_S, update E_S, observe $\boldsymbol{O'}_S$
    $f_d \leftarrow dtc(E_S),\ r_S \leftarrow \mathcal{R}(E_S, f_d)$
    $r_d \leftarrow \mathcal{D}_{\mathrm{KL}}\big(\pi_S^{\theta_S} \,\|\, \pi_T^{\theta_T}\big)$
    Take action a_t, update E_S, observe $\boldsymbol{O'}_T$
    $f'_d \leftarrow dtc(E_T),\ r_T \leftarrow \mathcal{R}(E_T, f'_d)$
    $r'_d \leftarrow \mathcal{D}_{\mathrm{KL}}\big(\pi_T^{\theta_T} \,\|\, \pi_S^{\theta_S}\big)$
    if $r_T \geq r_S$ **then**
      $r_S^{rec} \leftarrow r_S - \lambda_S \cdot (r_T - r_S) \cdot r_d$
      $r_T^{rec} \leftarrow k \cdot r_T$
    else if $r_T < r_S$ **then**
      $r_S^{rec} \leftarrow r_S + \lambda_S \cdot (r_S - r_T)$
      $r_T^{rec} \leftarrow k \cdot r_T - \lambda_T \cdot (r_S - r_T) \cdot r'_d$
    end if
    $\delta_S \leftarrow R_S^{rec} + \gamma \cdot V_S(\boldsymbol{O'}_S; \omega_S) - V_S(\boldsymbol{O}_S; \omega_S)$
    $\delta_T \leftarrow R_T^{rec} + \gamma \cdot V_T(\boldsymbol{O'}_T; \omega_T) - V_T(\boldsymbol{O}_T; \omega_T)$
    $\omega_S \leftarrow \omega_S - \alpha_S \cdot \delta_S \cdot \nabla_{\omega_S} V_S(\boldsymbol{O}_S; \omega_S)$
    $\omega_T \leftarrow \omega_T - \alpha_T \cdot \delta_T \cdot \nabla_{\omega_T} V_T(\boldsymbol{O}_T; \omega_T)$
    $\theta_S \leftarrow \theta_S - \beta_S \cdot \delta_S \cdot \nabla_{\theta_S} \ln \pi_S(\boldsymbol{O}_S; \theta_S)$
    $\theta_T \leftarrow \theta_T - \beta_T \cdot \delta_T \cdot \nabla_{\theta_T} \ln \pi_T(\boldsymbol{O}_T; \theta_T)$
    $E_T \leftarrow E_S$
  end while
end for

D PROOFS

Proof of Theorem 1. From the performance difference Lemma (Kakade & Langford, 2002), the difference in return between π_T and π_S can be written as

$$J(\pi_T) - J(\pi_S) = \frac{1}{1-\gamma} \mathbb{E}_{S \sim \rho_{\pi_T}, A \sim \pi_T(\cdot\|S)} \left[A^{\pi_S}(S, A) \right], \tag{16}$$

where $A^{\pi_S}(S,A) = Q^{\pi_S}(S,A) - V^{\pi_S}(S)$ is the Advantage function under policy π_S. Since $\mathbb{E}_{A \sim \pi_S(\cdot\|O_S)} \left[A^{\pi_S}(S,A) \right] = 0$, we can rewrite the expectation over actions as:

$$\sum_a \pi_T(a\|S) A^{\pi_S}(S,a) = \sum_a \left(\pi_T(a\|S) - \pi_S(a\|O_S) \right) A^{\pi_S}(S,a). \tag{17}$$

Taking absolute values and applying Hölder's inequality:

$$\left\| \sum_a \pi_T(a\|S) A^{\pi_S}(S,a) \right\| \leq \|\pi_T(\cdot\|S) - \pi_S(\cdot\|O_S)\|_1 \cdot \|A^{\pi_S}(S,\cdot)\|_\infty. \tag{18}$$

The advantage function is bounded by the maximum reward range as

$$\|A^{\pi_S}(S,\cdot)\|_\infty \leq \max_a Q^{\pi_S}(S,a) - \min_a Q^{\pi_S}(S,a) \leq \frac{R_{\max}}{1-\gamma} \tag{19}$$

By Pinsker's inequality, the L^1 norm (Total Variation distance) is bounded by the KL divergence as

$$\|\pi_T(\cdot\|S) - \pi_S(\cdot\|O_S)\|_1 \leq \sqrt{2 D_{\text{KL}}(\pi_T(\cdot\|S) \parallel \pi_S(\cdot\|O_S))} \tag{20}$$

By substitute the results from equations equation 20 and equation 19 into the expectation over ρ_{π_T}, we obtain

$$\|J(\pi_T) - J(\pi_S)\| \leq \frac{1}{1-\gamma} \cdot \frac{R_{\max}}{1-\gamma} \mathbb{E}_{S\sim\rho_{\pi_T}} \left[\sqrt{2 D_{\text{KL}}(\pi_T(\cdot\|S) \parallel \pi_S(\cdot\|O_S))} \right] \tag{21}$$

Applying Jensen's Inequality ($\mathbb{E}[\sqrt{X}] \leq \sqrt{\mathbb{E}[X]}$):

$$\|J(\pi_T) - J(\pi_S)\| \leq \frac{R_{\max}}{(1-\gamma)^2} \sqrt{2 \mathbb{E}_{S\sim\rho_{\pi_T}} \left[D_{\text{KL}}(\pi_T(\cdot\|S) \parallel \pi_S(\cdot\|O_S)) \right]} \leq \frac{R_{\max}}{(1-\gamma)^2} \sqrt{2\epsilon_{\text{KL}}}, \tag{22}$$

which completes the proof of the Theorem □

Proof of Theorem 2. Since $\mathcal{L}(\theta)$ is L-smooth, for any θ_{t+1} and θ_t, then we have

$$\mathcal{L}(\theta_{t+1}) \leq \mathcal{L}(\theta_t) + \langle \nabla_\theta \mathcal{L}(\theta_t), \theta_{t+1} - \theta_t \rangle + \frac{L}{2} \|\theta_{t+1} - \theta_t\|^2 \tag{23}$$

By substituting the update rule $\theta_{t+1} - \theta_t = -\alpha_t \nabla_\theta \mathcal{L}(\theta_t)$ into the smoothness inequality, we obtain

$$\mathcal{L}(\theta_{t+1}) \leq \mathcal{L}(\theta_t) - \alpha_t \|\nabla_\theta \mathcal{L}(\theta_t)\|^2 + \frac{L\alpha_t^2}{2} \|\nabla_\theta \mathcal{L}(\theta_t)\|^2 \tag{24}$$

By setting $\alpha_t = \frac{1}{L}$, we obtain

$$\mathcal{L}(\theta_{t+1}) \leq \mathcal{L}(\theta_t) - \frac{1}{L} \|\nabla_\theta \mathcal{L}(\theta_t)\|^2 + \frac{1}{2L} \|\nabla_\theta \mathcal{L}(\theta_t)\|^2, \tag{25}$$

$$\leq \mathcal{L}(\theta_t) - \frac{1}{2L} \|\nabla_\theta \mathcal{L}(\theta_t)\|^2 \tag{26}$$

Rearranging terms will result in

$$\|\nabla_\theta \mathcal{L}(\theta_t)\|^2 \leq 2L \left(\mathcal{L}(\theta_t) - \mathcal{L}(\theta_{t+1}) \right) \tag{27}$$

Summing this inequality from $t = 0$ to $T - 1$, will results in

$$\sum_{t=0}^{T-1} \|\nabla_\theta \mathcal{L}(\theta_t)\|^2 \leq 2L \left(\mathcal{L}(\theta_0) - \mathcal{L}(\theta_T) \right) \tag{28}$$

Since $\mathcal{L}(\theta) \geq \mathcal{L}^*$ (bounded below by minimum possible loss) as

$$\sum_{t=0}^{T-1} \|\nabla_\theta \mathcal{L}(\theta_t)\|^2 \leq 2L \left(\mathcal{L}(\theta_0) - \mathcal{L}^* \right) \tag{29}$$

Dividing both sides by T, we obtain

$$\min_{t\in\{0,\ldots,T-1\}} \|\nabla_\theta \mathcal{L}(\theta_t)\|^2 \leq \frac{1}{T} \sum_{t=0}^{T-1} \|\nabla_\theta \mathcal{L}(\theta_t)\|^2 \leq \frac{2L(\mathcal{L}(\theta_0) - \mathcal{L}^*)}{T} \tag{30}$$

Setting $\frac{2L(\mathcal{L}(\theta_0)-\mathcal{L}^*)}{T} \leq \epsilon$ yields the following bound.

$$T \geq \frac{2L(\mathcal{L}(\theta_0) - \mathcal{L}^*)}{\epsilon} = O\left(\frac{1}{\epsilon}\right), \tag{31}$$

which completes the proof of this Theorem. □

E EXPERIMENTAL SETUP DETAILS

E.1 HARDWARE AND SOFTWARE CONFIGURATION

The hardware configuration for these experiments included an Intel Core i5-13600K CPU with 32GB of RAM and an NVIDIA GeForce RTX 4070 GPU equipped with 12GB of graphics memory. The experiments were performed on a software platform running Ubuntu 22.04.

E.2 SDN CONFIGURATIONS AND HYPER-PARAMETER SETTINGS

The experimental platform described in Section 5.1, widely recognised and adopted in SDN security research (Revathi et al., 2022; Mohsin & Hamad, 2022; Alwabisi et al., 2022), is the standard in the field.

Utilising the OpenFlow protocol, the Ryu controller periodically issues event requests to extract traffic information from data layer switches, encompassing global, TCP, and UDP traffic metrics. In our tests, the Ryu controller was configured to obtain traffic data from the data layer at a minimum interval of 0.5 seconds. We leveraged this rapid cycle to collect real-time network traffic data, which was instrumental in constructing a comprehensive dataset for our detection system.

We utilise $iperf$ for real-time bandwidth assessments on bottleneck links. $Iperf$, a widely recognised tool, measures TCP and UDP bandwidth performance dynamically. Additionally, we employ Ping to evaluate link latency and to approximate network congestion levels. Ping, a ubiquitous network diagnostic tool, operates by sending ICMP echo requests to measure the round-trip time (RTT) and packet loss across various computers, functioning autonomously without the need for user interaction or specialised configuration on the target system.

Table 3 presents the key hyperparameters employed in the experiment.

Table 3: Hyperparameter settings

Parameter Setting	Value
Episode simulation time (s)	100
Controller sample interval (s)	0.5
Decision time slot (s)	1
Reward delay time (s)	0.5
Discount rate γ	0.95
Clipped congestion rate z_0	1
Bandwidth thresh B_{th} (Mbps)	0.1*B_{max}
Living penalty κ	5
Detected penalty p	100
Congestion reward R_c	80

E.3 DATASET AND PREPROCESSING

The WIDE dataset is a traffic data repository maintained by the MAWI working group of the WIDE project in Japan (WIDE Project, n.d.). This traffic data repository collects real traffic data from the backbone network and provides traffic tracking data from seven different sampling points over different time periods (hours to tens of hours) since 2006. These traffic tracking data are collected by the Tcpdump tool and include protocols such as TCP, UDP, and ICMP.

Based on the traffic data collected at sampling point F on January 1, 2018 from the WIDE dataset (WIDE Project, 2018), we processed and generated a Pacp format file. Specifically, we filtered out packets other than the transport layer protocol in the original dataset, retaining only data using TCP, UDP, and ICMP protocols, and modified the source and destination IP addresses of the packets using the Scapy library.

E.4 METRIC DEFINITIONS

Four performance metrics are recorded as LDoS attack's parameters and AdaDoS attack's strategy details, including **duration**, **period**, **attack rate** and **trigger rate**. The duration means the duration

of package sending per attack. The attack rate defines the package sent per seconds in current attack. The period decides the frequency to launch an attack. The trigger rate represents the average possibility for attacker to launch an attack.

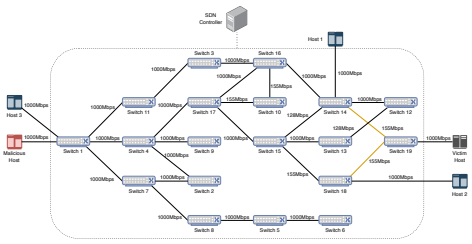
The attack success rate is calculated by equation 32 and bandwidth is measured by $iperf$. For LDoS, attack cost refers to the total number of transmitted data per attack cycle. For AdaDoS, attack cost is the average total number of transmitted data per episode.

$$\text{Attack success rate} = \frac{\text{No. undetected attacks}}{\text{No. attacks}}. \tag{32}$$

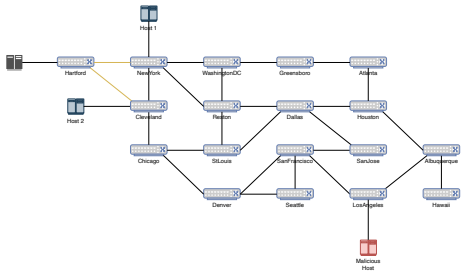
Note that the attack success rate and available bandwidth are the two most significant indicators of attack performance, as the primary objective of all SDN attack algorithms is to maximise effectiveness. Although attack cost is also a consideration for our algorithm, it is of lower priority. In real-world practice, acquiring additional resources is typically easier than achieving a higher success rate.

F Network Topology

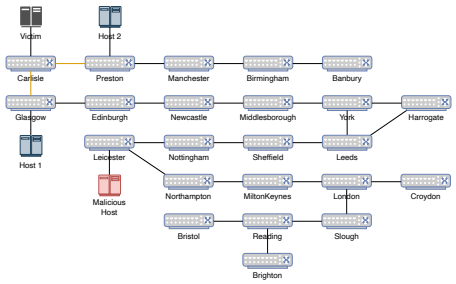
Aarnet is provided by AARNet (2024), as depicted in Figure 7a. Hosts 1 and 2 deliver background traffic from the WIDE dataset to the victim host, while host 3 measures real-time TCP bandwidth with $iperf$. The malicious host delivers attack traffic to the victim host, aiming to congest the link between them. Similar to Aarnet, background traffic from dataset are replayed on the Ansnet and Yorknet from the Internet Topology Zoo (Knight et al., 2011), whose topologies are shown in Figure 7b and Figure 7c, and experiments are conducted on them in the same way.



(a) Aarnet



(b) Ansnet



(c) Yorknet

Figure 7: Extra network topologies