A Low-Rate DoS Attack Mitigation Scheme Based on Port and Traffic State in SDN

Dan Tang, Rui Dai, Chenguang Zuo, Jingwen Chen, Keqin Li, *Fellow, IEEE*, and Zheng Qin

***Abstract*—Low-rate Denial of Service (DoS) attacks can significantly compromise network availability and are difficult to detect and mitigate due to their stealthy exploitation of flaws in congestion control mechanisms. Software-Defined Networking (SDN) is a revolutionary architecture that decouples network control from packet forwarding, emerging as a promising solution for defending against low-rate DoS attacks. In this paper, we propose Trident, a low-rate DoS attack mitigation scheme based on port and traffic state in SDN. Specifically, we design a multi-step strategy to monitor switch states. First, Trident identifies switches suspected of suffering from low-rate DoS attacks through port state detection. Then, it monitors the traffic state of switches with abnormal port states. Once a switch is identified as suffering from an attack, Trident analyzes the flow information to pinpoint the malicious flow. Finally, Trident issues rules to the switch's flow table to block the malicious flow, effectively mitigating the attack. We prototype Trident on the Mininet platform and conduct experiments using a real-world topology to evaluate its performance. The experiments show that Trident can accurately and robustly detect low-rate DoS attacks, respond quickly to mitigate them, and maintain low overhead.**

***Index Terms*—Attack detection, attack mitigation, low-rate denial of service attacks, software-defined networking.**

I. Introduction

In today's rapidly advancing era of artificial intelligence, network security is becoming increasingly important [1], [2]. Particularly, Denial of Service (DoS) attacks [3], [4], [5] have posed a serious threat to network security. A variant known as low-rate DoS attacks [6], [7] targets the quality of service of the Internet infrastructure by using a small amount of traffic rather than a sustained high volume, effectively degrading or disabling services while remaining less detectable.

In this context, Software-Defined Networking (SDN) offers new avenues for low-rate DoS attack defense [8], [9]. SDN is a revolutionary architecture centered on decoupling network control and packet forwarding [10], [11], [12]. In SDN, control logic is moved to the control plane, while underlying devices (e.g., switches) in the data plane are simplified to function as forwarding units [13], [14], [15], which significantly streamlines network management and provides ideal conditions for efficient traffic monitoring and attack mitigation [16], [17].

Despite researchers' efforts, defending against low-rate DoS attacks remains a challenging work [18], [19]. In existing solutions, the pattern-based methods [20], [21], [22] typically design detection schemes based on the bursty and periodic patterns of low-rate DoS attacks, as the attacker continuously sends short bursts. Another approach is the anomaly-based method [23], [24], [25], which develops strategies by monitoring the network anomalies induced by these attacks. These two methods are currently the most common approaches for detecting low-rate DoS attacks. However, network traffic is highly variable and influenced by numerous factors, such as device types, user behavior, and application protocols, which greatly complicates its patterns. In an increasingly complex network environment, both pattern-based and anomaly-based methods struggle to achieve accurate detection for low-rate DoS attacks. Furthermore, another critical requirement for defending against low-rate DoS attacks is the ability to mitigate them quickly and effectively, but much of the existing research focuses solely on detection, often overlooking mitigation. While some studies have considered mitigation measures, the slow response time remains a significant issue. Additionally, some research has focused only on algorithm design without adequately addressing the practical feasibility of deployment, either by not designing a scheme for actual deployment at all, or these algorithms can perform well on offline datasets, but their effectiveness often falls short in online applications.

In a nutshell, defending against low-rate DoS attacks still faces several challenges. These attacks are difficult to detect accurately, hard to mitigate quickly and effectively, and deploying detection and mitigation strategies remains a significant hurdle. Despite the stealthy nature of this attack, the inherent patterns in its behavior and the network anomalies it triggers, still make it

Received 22 August 2024; revised 31 January 2025; accepted 5 February 2025. Date of publication 11 February 2025; date of current version 9 April 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62472153, in part by the Natural Science Foundation General Project of Chongqing under Grant CSTB2022NSCQ-MSX1378, and in part by YueLuShan Center Industrial Innovation Project under Grant 2023YCII0115. Recommended for acceptance by X. Jia. *(Corresponding author: Rui Dai.)*

Dan Tang is with the College of Computer Science and Electronic Engineering (CSEE), Hunan University (HNU), Changsha 410082, China, and also with the Research Institute of Hunan University in Chongqing, Chongqing 401120, China (e-mail: dtang@hnu.edu.cn).

Rui Dai, Chenguang Zuo, Jingwen Chen, and Zheng Qin are with the College of Computer Science and Electronic Engineering (CSEE), Hunan University (HNU), Changsha 410082, China (e-mail: dairui@hnu.edu.cn; chenguangzuo@hnu.edu.cn; cjw1128@hnu.edu.cn; zqin@hnu.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TC.2025.3541143

feasible to accurately detect this attack. In addition, SDN offers significant flexibility and efficiency to network monitoring and management, greatly lowering the barriers to deploying solutions for low-rate DoS attacks and enabling quick and effective attack mitigation.

Inspired by existing solutions for this attack, we utilize anomalies caused by the attack as indicators for detection and leverage attack patterns to pinpoint and localize the attack flow. To the end, we propose Trident, a scheme based on port and traffic state in SDN. Trident first identifies switches that are suspected of suffering from the attack. Specifically, Trident uses Port Net Factor (PNF) to characterize the state of incoming and outgoing traffic on switch ports. In the normal state, the distribution of PNF tends to follow a normal distribution, whereas under attack, the distribution is abnormal and deviates from the normal distribution. When Trident detects an abnormal port state on a switch, it proceeds to monitor the switch's traffic state. Trident uses XGBoost [26] to analyze the state of traffic flowing through the switch. If its traffic state is abnormal, the switch is considered to have suffered an attack. The two-step detection of port and traffic states can effectively enhance the detection accuracy. Once a switch is detected to have suffered an attack, Trident analyzes the flows in the switch, examining both their periodicity and burstiness to pinpoint the malicious flow. It then issues flow rules to contain the malicious flow, thus effectively mitigating the attack. The main contributions are summarized below.

- We perform port state analysis to identify port anomalies caused by low-rate DoS attacks and develop a port state detection module to monitor switches that may suffer from attacks.
- We perform traffic state analysis to determine traffic anomalies resulting from low-rate DoS attacks and develop a traffic state detection module to pinpoint the switch suffering from attacks from these switches with anomalous ports.
- We develop a mitigation module to localize the malicious flow that can respond quickly to low-rate DoS attacks and filter out the attack traffic.
- We prototype Trident with Mininet platform and conduct experiments with a real-world topology to validate the performance of Trident.

The rest of the paper is as follows: Section II introduces the threat model and design goals of Trident. Section III reviews related works. Section IV provides an overview of Trident, followed by the design details in Section V. Section VI presents the evaluation of Trident's performance. Section VII discusses several limitations of Trident and Section VIII concludes the paper.

II. Threat Model and Design Goals

A. Threat Model

Low-rate DoS attack is typically aimed at the congestion control mechanism, during which the attacker periodically sends traffic pulses to trigger the TCP congestion control mechanism multiple times, with a low average traffic rate, thus reducing the throughput of TCP traffic and degrading the network's quality of service. The low-rate DoS attack can be characterized by a tuple (T, R, L), where T denotes the attack cycle (the time interval between successive traffic pulses), R represents the attack strength (the attack rate during each pulse), and L indicates the attack duration (the length of each traffic pulse).

B. Design Goals

We aim to develop a low-rate DoS attack mitigation scheme that ensures high detection accuracy and rapid response. Specifically, the scheme should achieve the following goals.

Robust and Accurate Detection. The scheme should enable the detection of low-rate DoS attacks in complex network environments. Particularly, it should be able to accurately detect various low-rate DoS attacks with different parameters, e.g., various attack rates, cycles, and durations.

Rapid Response Mitigation. The scheme should be capable of quickly responding to low-rate DoS attacks and mitigating them promptly, e.g., within seconds.

Easy Deployment with Low Resource Overhead. The scheme should be easy to deploy and have a low resource overhead, e.g., low CPU usage and low memory consumption.

III. Related Work

Low-rate DoS attacks use carefully crafted low-rate traffic to exploit flaws in the congestion control mechanism, significantly increasing the difficulty of detection [27]. However, the attack's tendency to trigger congestion, along with its periodic and bursty traffic patterns, provides a basis for detection. Furthermore, the network anomalies induced by the attack serve as valuable clues for identifying and tracing the attack. Here, we broadly categorize existing methods into pattern-based and anomaly-based methods.

Pattern-based methods. These methods design detection strategies based on the attack's behavior patterns, such as its intent to trigger network congestion [20], [21], as well as traffic behavior such as periodicity [28] and burstiness [22], [29]. Starting from the behavior of attack flow attempting to congest network, Zhang et al. [20] proposed the metric of Congestion Participation Rate (CPR) to portray the intention of flows and introduced a thresholding method to differentiate and filter attack flows with CPR exceeding a threshold. Chen et al. [21] proposed ConQuest to perform fine-grained queue statistics since queue utilization is an important metric that reflects network congestion. ConQuest identifies the flows contributing to the queue backlog, thereby detecting malicious flows with low-rate DoS attacks. Based on the periodicity and burstiness of attacks, Wu et al. [22] applied sequence alignment techniques to calculate the similarity score and compared it against a double-threshold rule to detect the attacks.

Anomaly-based methods. These methods design detection schemes based on the network anomalies caused by the attack, such as time-domain anomalies [23], frequency-domain anomalies [24], [30] and fractal feature anomalies [25] of network traffic. Tang et al. [23] proposed the GASF-IPP framework, which employs a time-domain analysis approach to monitor

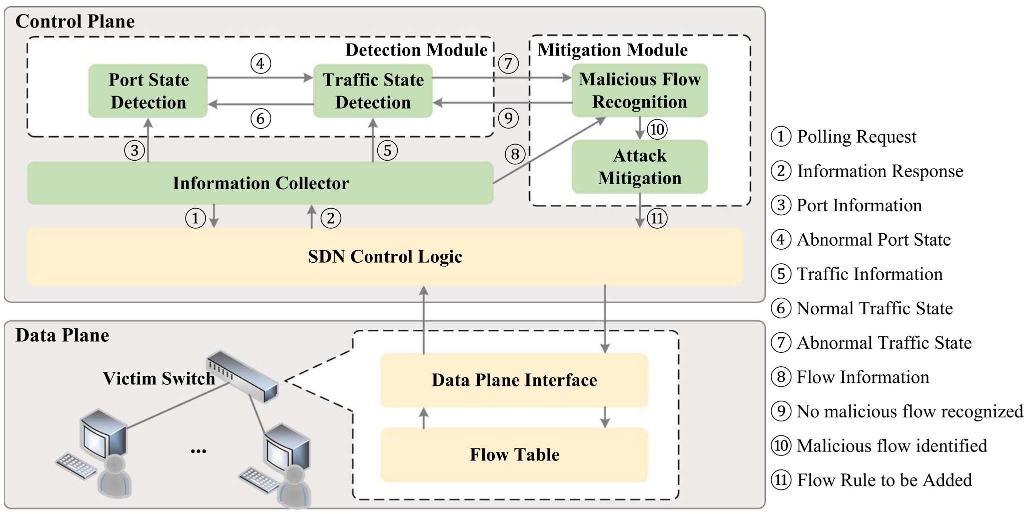


Fig. 1. Overview of Trident.

switch traffic. Upon detecting an attack, the framework identifies the IP and port information, enabling prompt mitigation of the attack. Targeting the power spectrum anomalies in network traffic caused by low-rate DoS attacks, Chen et al. [24] used power spectrum entropy to detect the attacks and applied a queuing algorithm to mitigate them. Since network traffic exhibits multifractal features at small time scales, the sudden onset of attack traffic can disrupt these features, leading to anomalies in traffic characteristics. Wu et al. [25] proposed to monitor changes in the multifractal features to detect the attacks.

However, defending against low-rate DoS attacks is still not an easy work and several challenges remain. (1) Network traffic is highly variable and influenced by various factors, such as user behavior, device types, and application protocols. In today's increasingly complex network environment, both pattern-based and anomaly-based methods struggle to achieve accurate attack detection. (2) Existing methods primarily focus on attack detection, often neglecting the need for effective attack mitigation and faster response times. (3) Most current approaches are designed at the algorithmic level without considering the deployment schemes and costs. Therefore, it is crucial to develop a scheme that offers high detection performance, rapid mitigation responses, and low-cost deployment.

IV. Overview of Trident

In this section, we present our low-rate DoS attack mitigation scheme, Trident[1]. Leveraging the flexibility and convenience of SDN for network management and monitoring, we develop Trident within the control plane to enable real-time monitoring of network traffic across switches and to provide a prompt response to the attack. Fig. 1 shows the overview of Trident, which comprises four major components: information collector module, port state detection module, traffic state detection module, and mitigation module.

[1]The port state detection module, traffic state detection module, and mitigation module collaborate seamlessly, forming a Trident of defense against low-rate DoS attacks.

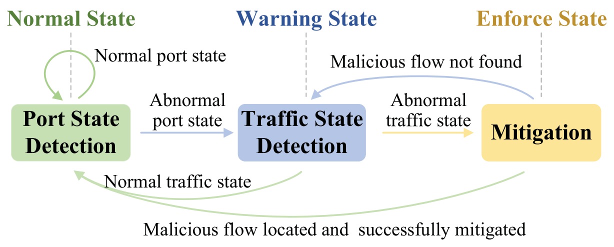


Fig. 2. Workflow of Trident.

Information Collector Module. Information collector module obtains switch information in real time, including port information, traffic information, and flow information. This information serves as the source for decision-making in the port state detection module, traffic state detection module, and mitigation module.

Port State Detection Module. Port state detection module monitors the switch's port state using the port information provided by the information collector module to identify switches potentially suffering from low-rate DoS attacks.

Traffic State Detection Module. Traffic state detection module monitors the switch's traffic state using the traffic information provided by the information collector module to identify those switches suffering from an attack. The two rounds of detection, involving port state and traffic state, greatly enhance detection accuracy.

Mitigation Module. Upon detecting an attack on the switch, mitigation module recognizes the malicious flow and enforces countermeasures to shield the switch from the attack.

Fig. 2 shows the workflow of Trident, consisting of three logical steps: port state detection, traffic state detection, and mitigation, which correspond to three logical states: normal state, warning state, and enforce state. Trident monitors multiple switches in the network in real time, maintaining monitoring information for each one. The three logical states represent the state of each switch as monitored by Trident. Trident initially

TABLE I
ABBREVIATIONS AND NOTATIONS IN THE PAPER

Abbreviations or Notations	Full Name or Explanation
SDN	Software-Defined Networking
DoS	Denial of Service
PNF	Port Net Factor
$PPNF$	Port Packet Net Factor
CPR	Congestion Participation Rate
PI	Polling Interval
XGBoost	Extreme Gradient Boosting
OAC	Optimal Auto-correlation Coefficient
CoV	Coefficient of Variation
$\{record_1, record_2, \ldots, record_{index}, \ldots, record_n\}$	The time series recording switch information
n	The length of time series
$index$	The position in the time series where records should be updated
$num\_records$	The total number of records
$\{v_{in_1}, \ldots, v_{in_n}\}$	The time series recording switch port inflow rates
$\{v_{out_1}, \ldots, v_{out_n}\}$	The time series recording switch port outflow rates
μ, σ	The mean value and standard deviation of PNF
k	The parameter in port state detection
th	Threshold for determining whether a switch port state is abnormal
ub_ave, tb_ave	The average value of UDP or TCP traffic to bandwidth
ub_cov, tb_cov	The coefficient of variation of UDP or TCP traffic
upl_ave	The average size of UDP packets
up_ent, tp_ent	The information entropy of packet rate for UDP or TCP traffic
B	The bandwidth of the bottleneck link
$\{bytes_{udp_1}, \ldots, bytes_{udp_i}, \ldots, bytes_{udp_n}\}$	$Series_{byte_{udp}}$ recording byte rates of the UDP traffic
$\{packets_{udp_1}, \ldots, packets_{udp_i}, \ldots, packets_{udp_n}\}$	$Series_{packet_{udp}}$ recording packet rates of the UDP traffic
$\{bytes_{tcp_1}, \ldots, bytes_{tcp_i}, \ldots, bytes_{tcp_n}\}$	$Series_{byte_{tcp}}$ recording byte rates of the TCP traffic
$\{packets_{tcp_1}, \ldots, packets_{tcp_i}, \ldots, packets_{tcp_n}\}$	$Series_{packet_{tcp}}$ recording packet rates of the TCP traffic
$\{PNF_1, \ldots, PNF_i, \ldots, PNF_n\}$	$Series_{PNF}$ recording PNF
$\{PPNF_1, \ldots, PPNF_i, \ldots, PPNF_n\}$	$Series_{PPNF}$ recording $PPNF$
$dpid$	The identifier information of a switch
$Series_{flowID}$	The time series containing flow information for the switch with an abnormal state
th_{oac}	Threshold for determining whether the optimal auto-correlation coefficient of a flow is abnormal
th_{cov}	Threshold for assessing whether the coefficient of variation of a flow is abnormal

starts in the normal state, assuming the monitoring switch is operating normally. In this state, it performs port state detection using collected port information. If the port state is normal, it remains in the normal state. If an abnormal port state is detected, Trident transitions to the warning state. In the warning state, Trident has already monitored that the switch port state is abnormal, but it cannot conclusively determine whether the switch has really suffered from a low-rate DoS attack. To this, Trident performs traffic state detection on the collected traffic data for further confirmation, and if it detects the switch's traffic state abnormality, it moves to the enforce state. If not, it returns to the normal state. When Trident reaches the enforce state, it has detected both abnormal port and traffic states of the switch, indicating an attack on the switch. Trident then recognizes the malicious flow from the collected flow information to locate the low-rate DoS attack. After identifying the malicious flow, Trident performs mitigation measures to protect the switch. Once the attack is mitigated, Trident reverts to the normal state. If no malicious flow is identified, the detection module may have false alarms, and Trident returns to the warning state to continue monitoring the switch's traffic state.

V. Design Details

In this section, we present the design details of Trident, i.e., the four major components in Trident. The abbreviations and notations in the paper are shown in Table I.

A. Information Collector Module

The information collector module is the cornerstone of Trident's ability to perform attack detection and mitigation. As shown in Fig. 1, information collector is deployed in the control plane, obtaining the switch's information through the SDN control logic interface. Specifically, the information collector sends polling requests to the switch at regular intervals PI. Upon receiving a response from the switch containing the requested information, it analyzes and records the data provided in the response. The response information includes port information, traffic information, and UDP flow information from the flow table. Polling requests should not be too frequent, considering the load they place on the controller and the control channel connecting the controller to the switch. Additionally, since the period of low-rate DoS attacks is typically 1 second or longer, we set the PI to 0.5 seconds to balance detection accuracy and polling load. This allows the scheme to gather information at least twice within a single attack cycle while avoiding too frequent requests and responses.

Information collector continuously polls the switch, and it receives multiple responses containing port and traffic information, which are essentially time-dependent switch state information and flow information, capturing the dynamic changes in port states, traffic patterns, and network flows over time. Information collector records this data as a time series. However, since Trident continuously requests the switch over an indefinite timeframe, the length of the time series grows indefinitely as monitoring hours continue to increase, which will consume controller's memory resources. To optimize resource usage, information collector retains switch information only for the most recent period (i.e., a time series of finite length), continuously updating the recorded information as new poll responses are received. Specifically, it maintains data for each monitored switch, including port state information, traffic state information, and flow information traversing the switch. The time series is denoted as $\{record_1, record_2, \ldots, record_{index}, \ldots, record_n\}$, recording the last n received switch information. We set n to 30, meaning the time series records the switch information for the last 30 polls, corresponding to the switch information within the last 15 seconds, since we have set the polling interval to 0.5 seconds. We use $index$ to indicate the position in the time series where records should be updated each time a polling response is received. The variable $num\_records$ tracks the total number of polling responses received, and $index$ is calculated as the value of $num\_records\%n$, where $\%$ is the modulo operator. Here, $record_{index}$ denotes the most recent entry, while $record_{index+1}$[2] represents the oldest. Note that these time series here are $Series_{byte_{udp}}$, $Series_{packet_{udp}}$, $Series_{byte_{tcp}}$, $Series_{packet_{tcp}}$, $Series_{PNF}$, and $Series_{PPNF}$ in the following sections, which record the port information and traffic information of the switch, respectively.

B. Port State Detection Module

In the detection phase, port state detection module receives and processes port information from the information collector, to determine whether the current port state of the switch is generating anomalies. We define PNF as the inflow-outflow ratio of a switch's port traffic, that is, the ratio of network traffic flowing in all ports to the data flowing out of all ports in RI on a switch. PNF is calculated as shown in Eq. 1, where v_{in} is the rate flowing into the switch port, and v_{out} is the rate flowing out of the switch port. A burst of low-rate DoS attack manifests as a sudden spike in incoming traffic at the port level of a switch within a short timeframe. Since switches can handle only a limited amount of traffic, any excess traffic that cannot be forwarded immediately is typically queued in a buffer. Consequently, when attack traffic arrives, the rate of incoming traffic significantly exceeds the outgoing traffic rate, creating an imbalance. During the silent phase of the attack, the switch processes the accumulated traffic, causing the incoming traffic rate to fall well below the outgoing rate. In contrast, under normal conditions without an attack, the incoming and outgoing traffic rates are generally balanced.

$$PNF = \frac{v_{in}}{v_{out}} \tag{1}$$

By analyzing the switch port data, we observe that under normal conditions, the inflow and outflow of switch port traffic are generally balanced. Fig. 3 shows the distribution of PNF

[2]If $index + 1$ is greater than n, it is the value of the remainder.

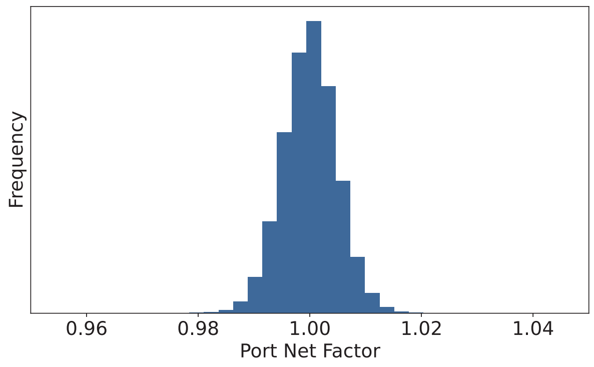


Fig. 3. The distribution of PNF under normal conditions.

under normal conditions, where the port data in the figure is from the Mininet platform. We observe that the PNF under normal conditions is primarily concentrated around 1, and its distribution closely follows a normal distribution. While under low-rate DoS attacks, the switch fails to forward traffic in a timely manner, causing the PNF values to fluctuate significantly. As a result, the distribution of the PNF deviates from the normal distribution.

According to the above analysis, the low-rate DoS attack causes anomalies in the switch port state, i.e., it causes anomalies in the distribution of the PNF values of the switch ports. To quantify and detect these anomalies in port states, we propose an anomaly detection algorithm based on the distribution of PNF values. Specifically, we use the mean value μ and standard deviation value σ of PNF values to measure whether the port state is normal. For switch ports under a normal condition, the PNF values are predominantly distributed around the mean value μ. To evaluate the degree of concentration around this mean value, we calculate the proportion of PNF values within the range $\mu \pm k \cdot \sigma$. It is clear that the proportion of PNF values within the range $\mu \pm k \cdot \sigma$ under normal conditions is greater than that during an attack. The algorithm is shown in Algorithm 1 and its inputs are the parameter k, the threshold th, the time series $Series_{in} = \{v_{in_1}, \ldots, v_{in_n}\}$ and $Series_{out} = \{v_{out_1}, \ldots, v_{out_n}\}$, and the output of the algorithm is the switch port state. If the proportion of PNF values outside the range $\mu \pm k \cdot \sigma$ is greater than th, the switch port state is abnormal, otherwise it is normal.

A key design goal of Trident is to achieve robust and accurate detection, which requires the port state detection algorithm to be adaptable to handle complex and dynamic network environments. The key parameters that determine the output of Algorithm 1 are parameter k and threshold th. Considering that the distribution of PNF values under normal conditions approximately follows a normal distribution, the confidence interval of the normal distribution can be utilized to set k and th, in which we set k to 3 and th to 0.0026. In addition to low-rate DoS attacks, other factors such as different network attacks, and user behavior like flash crowd events can also cause abnormal port states on the switch. To accurately detect the attack, further monitoring of their traffic patterns is necessary.

Algorithm 1: The anomaly detection algorithm based on the distribution of PNF values.

```
Input: The parameter $k$, the threshold $th$, the time
       series $Series_{in} = \{v_{in_1}, ..., v_{in_n}\}$ and
       $Series_{out} = \{v_{out_1}, ..., v_{out_n}\}$
Output: Switch port state
1  $Series_{PNF}$ = [];
2  for $v_{in}$, $v_{out}$ in $Series_{in}$ and $Series_{out}$ do
3      $PNF = \frac{v_{in}}{v_{out}}$;
4      $Series_{PNF}$.append($PNF$);
5  $\mu$ = mean($Series_{PNF}$);
6  $\sigma$ = std($Series_{PNF}$);
7  $cnt$ = 0;
8  for $PNF$ in $Series_{PNF}$ do
9      if $PNF \geq \mu - k \cdot \sigma$ and $PNF \leq \mu + k \cdot \sigma$ then
10         $cnt$ += 1;
11 $Proportion$ = 1 - $cnt/len(Series_{PNF})$;
12 if $Proportion > th$ then
13     Switch port state is abnormal;
14 else
15     Switch port state is normal;
```

C. Traffic State Detection Module

To ensure accurate detection of the attack, traffic state detection is utilized to monitor whether the traffic passing through the switch exhibits characteristics of such attacks. The system transitions from port state detection to traffic state detection only when Trident identifies anomalies in the switch's port state. The traffic state detection module receives and processes traffic data provided by the information collector, which records the traffic information as time series. We employ a combination of feature vectors and classifiers to determine whether the network is currently under an attack. The feature vectors must be carefully selected to be representative, capturing the impact of the attacks on the network traffic. Additionally, they should effectively highlight the differences between the traffic's normal and attack states.

On this premise, we select three types of features with a total of nine dimensions. The first type is based on the features of attack characteristics. The low-rate DoS attack typically uses UDP traffic to initiate short-time high-speed pulses to achieve the attack effect. Therefore, the UDP traffic will show periodic and obvious changes due to the mixing of attack traffic. Its mean value and dispersion degree are different from the normal situation. The malicious user usually sets the UDP packet size to be small when launching attacks. Based on this, we adopt the average value of UDP traffic to bandwidth, the coefficient of variation of UDP traffic, the average size of UDP packets, and the information entropy of packet rate as the first category of features. The information collector records the most recent UDP traffic flowing through the switch as the time series $\{bytes_{udp_1}, \ldots, bytes_{udp_i}, \ldots, bytes_{udp_n}\}$ and $\{packets_{udp_1}, \ldots, packets_{udp_i}, \ldots, packets_{udp_n}\}$, where $bytes_{udp}$ and $packets_{udp}$ are the byte rate and packet rate of the UDP traffic passing through the switch, recorded at each polling interval. The first type of features are denoted as follows:

$$ub\_ave = \frac{\sum_{i=1}^{n} bytes_{udp_i}}{n \times B} \tag{2}$$

$$ub\_cov = \frac{\sqrt{\frac{1}{n}\sum\limits_{i=1}^{n} (bytes_{udp_i} - mean(bytes_{udp}))^2}}{mean(bytes_{udp})} \tag{3}$$

$$upl\_ave = \frac{\sum_{i=1}^{n} bytes_{udp_i}}{\sum_{i=1}^{n} packets_{udp_i}} \tag{4}$$

$$up\_ent = -\sum_{i=1}^{n} \mathrm{P}(packets_{udp_i}) \log_b \mathrm{P}(packets_{udp_i}) \tag{5}$$

In the above equation, B denotes the bandwidth of the bottleneck link, $mean(bytes_{udp})$ is the mean value of the time series $\{bytes_{udp_1}, \ldots, bytes_{udp_i}, \ldots, bytes_{udp_n}\}$. P($x$) in Eq. (4) denotes the probability mass function of x and b is usually set to 2.

The information collector records the most recent TCP traffic flowing through the switch as the time series $\{bytes_{tcp_1}, \ldots, bytes_{tcp_i}, \ldots, bytes_{tcp_n}\}$ and $\{packets_{tcp_1}, \ldots, packets_{tcp_i}, \ldots, packets_{tcp_n}\}$, where $bytes_{tcp}$ and $packets_{tcp}$ are the byte rate and packet rate of the TCP traffic passing through the switch, recorded at each polling interval. The second category of features is based on the attack effect. The fairness of network resource allocation [31] makes TCP traffic sensitive to low-rate DoS attacks. Due to the nature of the protocol, UDP traffic of short duration and high rate sent by malicious users has the upper hand in the competition for resources and easily wins the right to use the bandwidth and occupies most of it, which increases the TCP packet loss rate. This leads to the triggering of the TCP congestion control mechanism, where TCP senders actively reduce the number of packets sent. Malicious users will send such attack flows for a long time, keeping the TCP connection in congestion control, so TCP transmission rate stays low. The features of the second category are the coefficient of variation of TCP traffic, the average percentage of TCP to total traffic, and the information entropy of TCP packet rate. They are defined as follows:

$$tb\_cov = \frac{\sqrt{\frac{1}{n}\sum\limits_{i=1}^{n} (bytes_{tcp_i} - mean(bytes_{tcp}))^2}}{mean(bytes_{tcp})} \tag{6}$$

$$tb\_ave = \frac{\sum_{i=1}^{n} bytes_{tcp_i}}{n \times B} \tag{7}$$

$$tp\_ent = -\sum_{i=1}^{n} \mathrm{P}(packets_{tcp_i}) \log_b \mathrm{P}(packets_{tcp_i}) \tag{8}$$

In the above equation, $mean(bytes_{tcp})$ is the mean value of the time series $\{bytes_{tcp_1}, \ldots, bytes_{tcp_i}, \ldots, bytes_{tcp_n}\}$.

The third category of features is based on the port information. As described in Section V-B, v_{in} and v_{out} represent the byte rate flowing into and out of the switch port, respectively. Similarly, we define v_{pin} and v_{pout} as the packet rates flowing into and out of the switch port. Based on the port rates recorded in each poll of the information collector, we can get the time series $\{PNF_1, \ldots, PNF_i, \ldots, PNF_n\}$ and $\{PPNF_1, \ldots, PPNF_i, \ldots, PPNF_n\}$, where PNF and $PPNF$ are the inflow-outflow byte and packet rates of a switch's port traffic, calculated as Eq. 1 and Eq. 9, respectively. The features of the third category are the average values of $\{PNF_1, \ldots, PNF_i, \ldots, PNF_n\}$ and $\{PPNF_1, \ldots, PPNF_i, \ldots, PPNF_n\}$, which indicates the state of switch port inflows and outflows for a recent period of time.

$$PPNF = \frac{v_{pin}}{v_{pout}} \tag{9}$$

Machine learning has greatly advanced the fields of software security and cybersecurity [32], [33]. Here, we select the XGBoost [26] classifier for the traffic state detection module because of its superior execution speed and performance. It can process sparse data with high efficiency and supports parallel computation. The optimized data structure and caching of the algorithm make it more efficient. As an ensemble learning algorithm, the underlying idea of XGBoost for classification is to combine multiple weak classifiers to strengthen the final classification result and make the classification result more reliable. The feature vector consisting of the three types of features described above will be fed into the trained classifier to get the prediction result, thus indicating the traffic state of the monitored switch.

The procedure for traffic state detection is shown in Algorithm 2 and its inputs are the XGBoost classifier, $Series_{byte_{udp}}$, $Series_{packet_{udp}}$, $Series_{byte_{tcp}}$, $Series_{packet_{tcp}}$, $Series_{PNF}$, $Series_{PPNF}$, and the bottleneck bandwidth B. We take the samples in normal traffic state as negative samples and the samples under low-rate DoS attack as positive samples. The prediction result of 0 indicates that the switch traffic state is normal, whereas the prediction result of 1 signifies that the switch traffic state is abnormal.

Algorithm 2: The algorithm for traffic state detection.

```
Input: The XGBoost classifier, $Series_{byte_{udp}}$,
       $Series_{packet_{udp}}$, $Series_{byte_{tcp}}$, $Series_{packet_{tcp}}$,
       $Series_{PNF}$, $Series_{PPNF}$, and the bottleneck
       bandwidth $B$
Output: Switch traffic state
1  $ub\_ave$ = cal_ub_ave($Series_{byte_{udp}}$, $B$);
2  $ub\_cov$ = cal_ub_cov($Series_{byte_{udp}}$);
3  $upl\_ave$ = cal_upl_ave($Series_{byte_{udp}}$,
     $Series_{packet_{udp}}$);
4  $up\_ent$ = cal_up_ent($Series_{packet_{udp}}$);
5  $tb\_ave$ = cal_tb_ave($Series_{byte_{tcp}}$, $B$);
6  $tb\_cov$ = cal_tb_cov($Series_{byte_{tcp}}$);
7  $tp\_ent$ = cal_tp_ent($Series_{packet_{tcp}}$);
8  $pnf\_ave$ = cal_pnf_ave($Series_{PNF}$);
9  $ppnf\_ave$ = cal_ppnf_ave($Series_{PPNF}$);
10 $Fea$ = [$ub\_ave$, $ub\_cov$, $upl\_ave$, $up\_ent$, $tb\_ave$,
     $tb\_cov$, $tp\_ent$, $pnf\_ave$, $ppnf\_ave$];
11 $pred$ = XGBoost.predict([$Fea$]);
12 if $pred$ is 0 then
13     Switch traffic state is normal;
14 else
15     Switch traffic state is abnormal;
```

D. Mitigation Module

When the scheme moves to mitigation module, it means that Trident has detected an attack in the network. In the mitigation phase, mitigation module receives and processes flow information from the information collector to identify malicious flow and take mitigation measures. The mitigation module identifies the malicious flow only from udp-type flows since UDP is the most commonly used traffic type for low-rate DoS attacks. Trident uses the ternary ⟨*srcIP*, *dstIP*, *Proto*⟩ as the flow identifier for network flows, where *srcIP* is the source IP address, *dstIP* denotes the destination IP address, and *Proto* refers to the protocol of the network flow.

The periodicity and burstiness of low-rate DoS flow make it show a distinction from the benign flow, which provides clues to identify the malicious flow. Therefore, Trident analyzes the characteristics of network flows in terms of both periodicity and burstiness to recognize the malicious flow of low-rate DoS attacks. Specifically, Trident employs the Optimal Autocorrelation Coefficient (OAC) of UDP flows to indicate the possibility of its periodicity, and the Coefficient of Variation (CoV) of UDP flows to quantify the presence of burstiness. Trident calculates the maximum auto-correlation value for one to six lags, which evaluates the periodicity of the UDP flow over intervals ranging from 0.5 to 3 seconds since the polling interval is 0.5 seconds. Larger or smaller lags are not necessary to be considered. This is because larger attack cycle T will cause the attack flow to fail to achieve the desired attack effect. And a smaller attack cycle makes the attack itself tend to be flooding style and does not fall under the category of low-rate DoS attacks.

Through port state detection and traffic state detection, Trident identifies which monitored switches have suffered an attack and passes the information of these switches to the mitigation module for mitigation operations. The procedure for the mitigation module is shown in Algorithm 3 and its inputs include the $dpid$, $Series_{flowID}$, threshold th_{oac}, and threshold th_{cov}. Here, $dpid$ represents the identifier of the switch with an abnormal state, $Series_{flowID}$ contains the flow information of the $dpid$ switch provided by the information collector, th_{oac} is the threshold for determining whether the optimal auto-correlation coefficient of a flow is abnormal, and th_{cov} is the threshold for assessing whether the coefficient of variation of a flow is abnormal. The algorithm first calculates the optimal auto-correlation coefficient and the coefficient of variation for each flow and compares them with the thresholds to determine whether the flow is malicious. If it is recognized as malicious, the flowID will be added to the blacklist. Next, mitigation operations are performed for each flow in the blacklist. FlowMod

Algorithm 3: The algorithm for mitigation module.

```
Input: $dpid$, $Series_{flowID}$, threshold $th_{oac}$, and
       threshold $th_{cov}$
Output: Mitigation measures for low-rate DoS attacks
1  $blacklist$ = [];
2  for $Series_{flowID_i}$ in $Series_{flowID}$ do
3      $OAC_i$ = cal_oac($Series_{flowID_i}$);
4      $CoV_i$ = cal_cov($Series_{flowID_i}$);
5      if $OAC_i > th_{oac}$ and $CoV_i > th_{cov}$ then
6          if $flowID_i$ not in $blacklist$ then
7              $blacklist$.append($flowID_i$);
8  for $flowID_i$ in $blacklist$ do
9      $srcIP$, $dstIP$, $Proto$ = getFlowInfo($flowID_i$);
10     $match$ = {$ipv4\_src$: $srcIP$, $ipv4\_dst$: $dstIP$,
         $ip\_proto$: $Proto$};
11     $priority$ = 65535;
12     $actions$ = [];
13     $msg$ = FlowMod({dpid=$dpid$, match=$match$,
         priority=$priority$, action=$actions$});
14     send_msg($msg$);
```

messages are used to add flow table entries that implement mitigation actions to the switch suffering the low-rate DoS attack, thereby mitigating the attacks. Specifically, Trident obtains $srcIP$, $dstIP$, and $Proto$ of the flow, and constructs a FlowMod message for the flow, including a *match* field, a *priority* field, and an *action* field. The detailed explanation of each field is given below.

Field *match*: This field contains $srcIP$, $dstIP$, and $Proto$ of the flow and is used to locate the malicious flow.

Field *priority*: This field represents the priority of this entry. The value of this field is usually an integer between 0 and 65535. The greater the $priority$, the higher the rank of the rule. To ensure that mitigation entries are matched first, we set $priority$ to a boundary value of 65535.

Field *action*: This field defines the action to be taken on packets that match this traffic entry. We set it to *drop* (*actions* = []) to indicate that all packets are discarded.

VI. Experimental Evaluation

In this section, we develop the Trident prototype and evaluate its performance and overhead.

A. Experiment Setup

Experimental Platform. We conduct experiments and evaluations using Mininet[3], a lightweight simulation platform for SDN development and evaluation. The SDN controller framework utilized in the experiments is Ryu[4], an open source controller that allows developers to write and manage applications in Python.

[3]https://mininet.org/
[4]https://ryu-sdn.org/

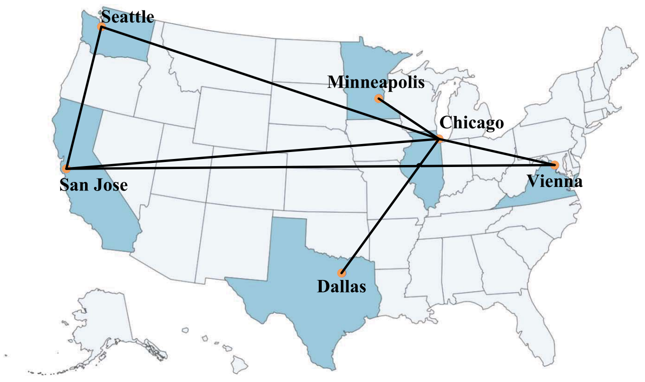


Fig. 4. Network topology.

TABLE II
Properties of Nodes in the Topology

Node Name	Node Type	Node Function
Seattle	City node	Forwarding network traffic
Seattle_host	Local host	Simulating low-rate DoS attackers
San Jose	City node	Forwarding network traffic
Minneapolis	City node	Forwarding network traffic
Minneapolis_host	Local host	Sending benign TCP traffic
Chicago	City node	Forwarding network traffic
Vienna	City node	Forwarding network traffic
Vienna_host	Local host	Receiving network traffic
Dallas	City node	Forwarding network traffic
Dallas_host	Local host	Sending benign UDP traffic

Network Topology. We built the network topology on Mininet based on the real-world topology *NapNet* from Topology Zoo[5] [34]. Fig. 4 shows the topology of *NapNet*, which includes six city nodes: Seattle, San Jose, Minneapolis, Chicago, Vienna, and Dallas. Table II shows the types and functions of the nodes in the topology. In the experiments, we set up four local hosts connected to the Seattle, Minneapolis, Vienna, and Dallas nodes. Each city node serves as an edge switch, with internal connections to local hosts and external links to other city nodes. Local links are the internal connections that connect the local hosts to their respective city nodes, providing a bandwidth of 1000 Mbps. Similarly, city links are the external links that interconnect the city nodes with a bandwidth of 128 Mbps, except for the link connecting Vienna and San Jose, which is the bottleneck link in the network topology and has a reduced bandwidth of 45 Mbps.

Workload Traffic. We employ *iPerf*, a widely used network performance testing tool for measuring and optimizing network bandwidth, to generate benign TCP and UDP traffic. Additionally, we implement a script for the low-rate DoS attack using the Python socket to generate attack traffic. The Minneapolis_host generates benign TCP traffic, transmitting it to the Vienna_host via the city nodes Minneapolis, Chicago, San Jose, and Vienna. Similarly, the Dallas_host generates benign UDP traffic and sends it to the Vienna_host through the city nodes Dallas, Chicago, San Jose, and Vienna. Meanwhile, the Seattle_host

[5]https://topology-zoo.org/dataset.html

TABLE III
THE PARAMETERS OF LOW-RATE DOS ATTACKS IN TESTING DATA

No.	Attack Rate (R)	Attack Cycle (T)	Attack Duration (L)
1	50Mbps	1.5s	0.2s
2	50Mbps	1.5s	0.4s
3	50Mbps	2.5s	0.2s
4	50Mbps	2.5s	0.4s
5	60Mbps	1.5s	0.2s
6	60Mbps	1.5s	0.4s
7	60Mbps	2.5s	0.2s
8	60Mbps	2.5s	0.4s

generates attack traffic, routing it to the Vienna_host via the city nodes Seattle, San Jose, and Vienna. Attack traffic and benign traffic travel through the San Jose city node and converge on the link connecting San Jose and Vienna.

Evaluation Metrics. The detection module, including the port state detection module and the traffic state detection module, analyzes and monitors the time series maintained by the information collector module, which reflects the port state and traffic state of the switch. Therefore, evaluating the detection performance of Trident is essentially assessing its accuracy in classifying the time series maintained by the information collector module. These time series contain the port and traffic state information of the monitored switch in the recent period. By analyzing these time series, Trident can effectively monitor the switch's state in real time. The following metrics are selected to evaluate Trident's detection performance: *Accuracy*, *Recall*, *Precision*, and *F1-Score*.

B. Detection Performance

We evaluate the performance of Trident's detection module, which includes both port state detection and traffic state detection, as a key goal of Trident is to achieve robust and accurate detection for the attack. To evaluate the robustness of Trident, we simulate 8 sets of detection experiments with different attack parameters on the Mininet platform, as shown in Table III. Each set of experiments simulates for 7200 seconds with a polling interval of 0.5 seconds, thus generating a total of 14400 records per set of experiments. In the *NapNet* topology, the link connecting San Jose and Vienna is the bottleneck link. Although the low-rate DoS attack traffic passes through the Seattle, San Jose, and Vienna nodes, the attack traffic rate does not exceed the bandwidth of the links connecting Seattle and San Jose, and the link between Vienna and Vienna_host. Thus only the San Jose node is the switch that has suffered an attack.

In the experiments, we take the samples in normal state as negative samples and the samples under low-rate DoS attack as positive samples. *Precision* quantifies Trident's performance in false positives and *Recall* reflects the Trident's ability to detect attacks. *Accuracy* and *F1-Score* indicate the overall performance of Trident.

1) Performance Evaluation for Port State Detection: The port state detection employs the anomaly detection algorithm based on the distribution of PNF values, which is trained using data that does not include any attacks. To obtain training data

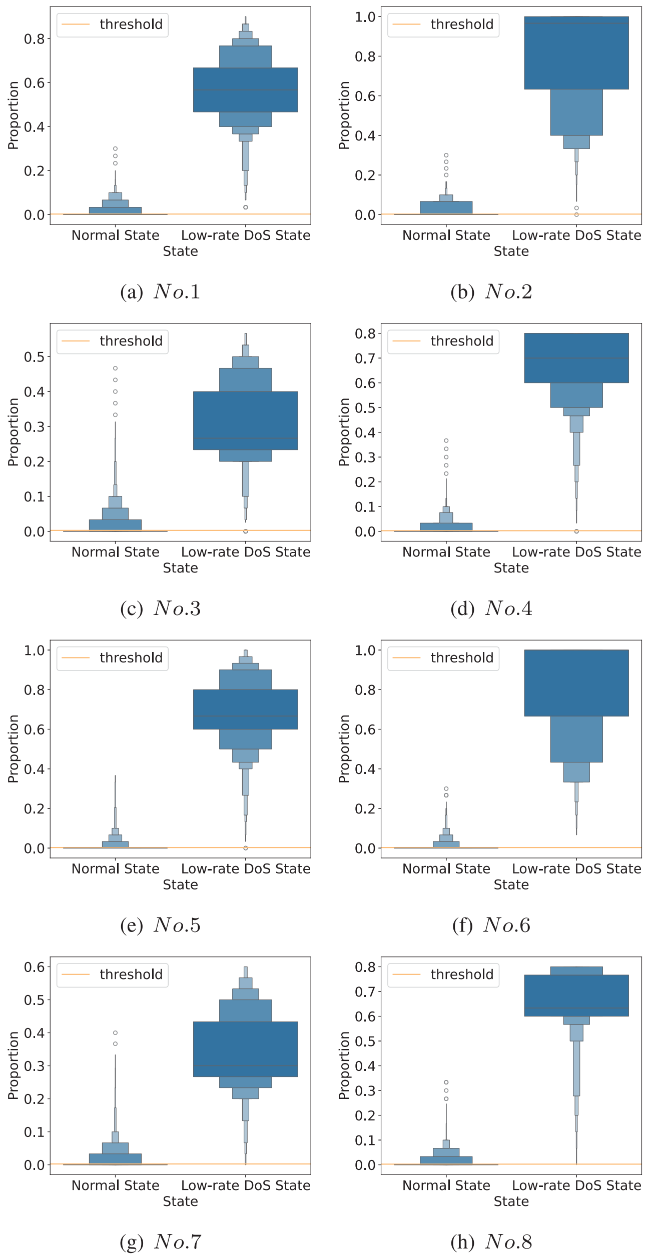


Fig. 5. Distribution of proportions beyond the range of $\mu \pm k \cdot \sigma$.

for this algorithm, we have conducted a 4-hour simulation on the Mininet platform, totaling 14400 seconds. As illustrated in Section V-B, we can set the parameters of the anomaly detection algorithm based on the confidence interval of the normal distribution, where we set the parameters k to 3 and th to 0.0026.

Fig. 5 shows the distribution of proportions beyond the range of $\mu \pm k \cdot \sigma$ for 8 sets of attack experiments in Table III. The box plots represent the proportions under normal and low-rate DoS states, with the yellow solid line indicating the threshold of 0.0026. As seen, for attacks with different parameters, the proportion exceeding the range of $\mu \pm k \cdot \sigma$ is much higher than that in the normal state. When k is set to 3 and th to 0.0026, nearly all low-rate DoS states can be identified, but it also causes some normal states to be classified as attack states,

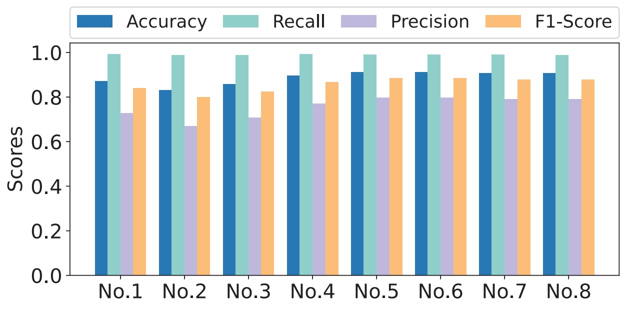
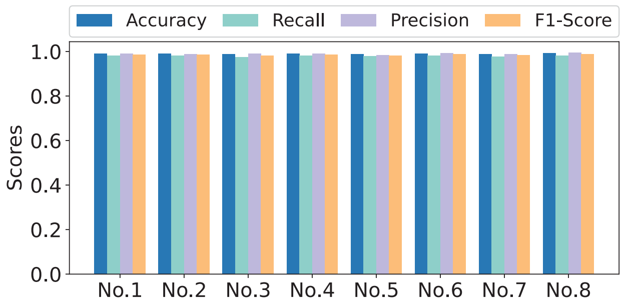


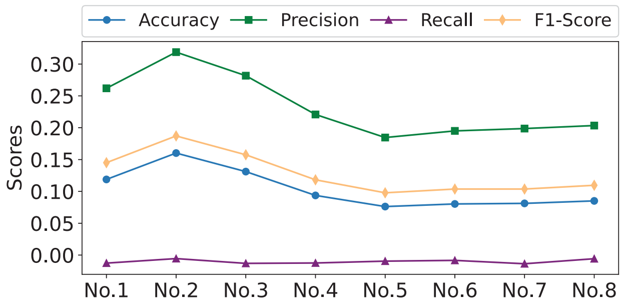
Fig. 6. Port state detection performance.

TABLE IV
THE PARAMETERS OF LOW-RATE DOS ATTACKS IN TRAINING DATA OF TRAFFIC STATE DETECTION

No.	Attack Rate (R)	Attack Cycle (T)	Attack Duration (L)
1	45Mbps	1s	0.1s
2	45Mbps	1s	0.3s
3	45Mbps	2s	0.1s
4	45Mbps	2s	0.3s
5	55Mbps	1s	0.1s
6	55Mbps	1s	0.3s
7	55Mbps	2s	0.1s
8	55Mbps	2s	0.3s



(a) Traffic state detection performance.



(b) Performance gains through traffic state detection.

Fig. 7. The evaluation of Traffic state detection performance.

leading to false positives. As the first round of detection, port state detection aims to monitor low-rate DoS attacks as comprehensively as possible. Therefore, false positives are tolerated, as they will be further confirmed in subsequent traffic state detection.

Fig. 6 shows the performance of port state detection for 8 sets of attack experiments in Table III. The horizontal coordinate indicates the group number and the vertical coordinate is the evaluation scores, including: *Accuracy*, *Recall*, *Precision*, and *F1-Score*. As shown, the *Recall* of port state detection among 8 sets of low-rate DoS attack experiments is all high, exceeding 98%, which indicates that port state detection has a strong ability to detect the attacks. While the *Precision* of port state detection is relatively low, which indicates that the probability of false positives is high. Trident contains two rounds of detection of low-rate DoS attacks. The purpose of port state detection is to monitor the occurrence of attacks as much as possible. Given that traffic state detection will further filter out false positives, some level of false positives in port state detection is tolerable. Due to the high false positive rate in port state detection, the overall *Accuracy* and *F1-Score* are not very high.

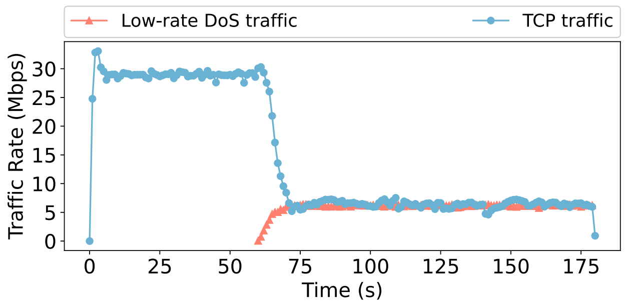
2) Performance Evaluation for Traffic State Detection: The traffic state detection employs the XGBoost to classify the traffic states, which is a supervised learning model. To obtain training data for the XGBoost, we have conducted a 4-hour simulation on the Mininet platform, totaling 14400 seconds, which includes 8 sets of attacks with different parameters. The parameters of attacks in training data of traffic state detection are shown in Table IV.

Fig. 7(a) shows the performance of traffic state detection for 8 sets of attack experiments in Table III. The horizontal coordinate indicates the group number and the vertical coordinate is the evaluation scores, including: *Accuracy*, *Recall*, *Precision*, and *F1-Score*. Compared to the low *Precision* of port state detection, traffic state detection has a higher *Precision*, along with high *Accuracy*, *Recall*, and *F1-Score*. The traffic state detection has high evaluation metrics for attacks with different parameters, indicating that Trident achieves accurate and robust detection of the attacks. Fig. 7(b) shows the performance gains from traffic state detection compared to port state detection. The blue line represents the gain in *Accuracy*, the green line shows the gain in *Precision*, the purple line indicates the gain in *Recall*, and the yellow line is the gain in *F1-Score*. As seen, except for *Recall*, the other three metrics show significant improvement, with *Precision* experiencing the greatest increase. This indicates that traffic state detection has significantly reduced false positives and improved detection accuracy. Compared to the *Recall* of port state detection, the *Recall* of traffic state detection is slightly lower, as traffic state detection further filters out the attacks reported by port state detection, which may have led to some false negatives. Overall, traffic state detection has significantly enhanced detection performance, and Trident has achieved accurate and robust detection for the attack.

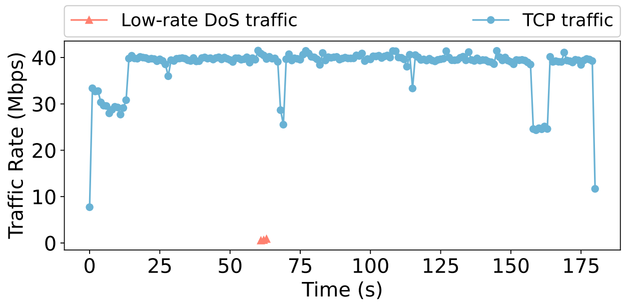
As shown in Table V, we compare Trident with other schemes for defending against low-rate DoS attacks. According to their reported results, HGBPLDT [29] has the highest *Accuracy*, *Precision*, and *F1-Score*, at 96.214%, 97.413%, and 96.901%, respectively, while ERT-EDR [9] achieves the highest *Recall* at 97.91%. Compared to these three mechanisms, Trident achieves a better detection performance across all metrics, proving that it is an effective solution for detecting low-rate DoS attacks.

TABLE V
DETECTION PERFORMANCE COMPARISON WITH OTHER SCHEMES

Scheme	*Accuracy*	*Precision*	*Recall*	*F1-Score*
ERT-EDR [9]	95.25%	–	97.91%	95.57%
GASF-IPP [23]	93.57%	95.39%	93.22%	94.29%
HGBPLDT [29]	96.214%	97.413%	96.394%	96.901%
Trident	98.98%	99.00%	98.00%	98.50%



(a) Network traffic under low-rate DoS attacks when Trident is disabled.



(b) Network traffic under low-rate DoS attacks when Trident is enabled.

Fig. 8. The schematic of network traffic over time.

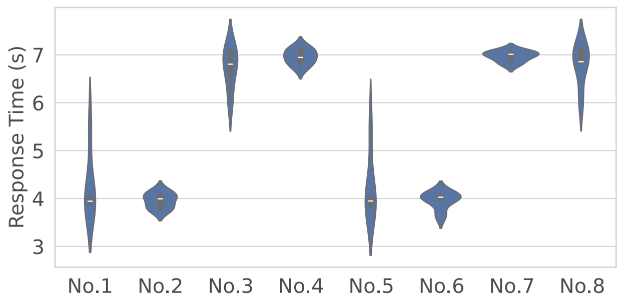
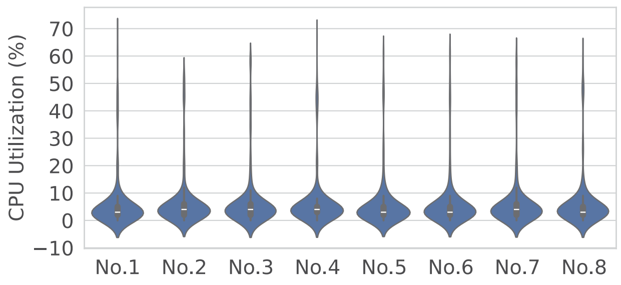
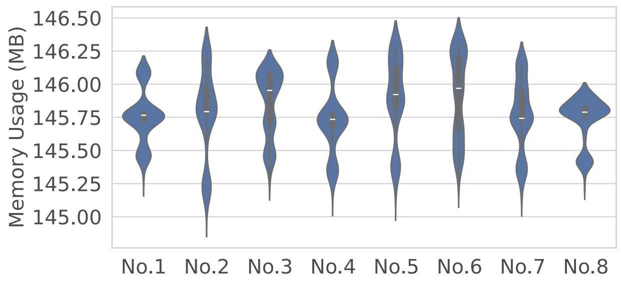


Fig. 9. Response time of Trident.



(a) CPU utilization.



(b) Memory usage.

Fig. 10. The overhead of Trident.

C. Mitigation Performance

Fig. 8 shows the schematic of network traffic over time, where Fig 8(a) shows the network traffic when Trident is disabled while Fig 8(b) shows the network traffic when Trident is enabled. We simulate traffic for about 180 seconds, with the attack launched around the 60th second. As observed, when Trident is disabled, the attack can cause a severe degradation of the TCP traffic rate. However, when Trident is enabled, the attack traffic is mitigated within a short period of time, and the TCP traffic is almost unaffected, with no reduction in the rate.

The faster Trident responds, the less damage the attack inflicts on the switch. The response time here is the duration between the switch receiving the attack traffic and Trident successfully blocking it. To evaluate Trident's response time, we capture the network traffic in the SanJose node using tcpdump and save it as a pcap file. When Trident is enabled, it blocks the traffic of the attack. We record the duration for which the attack flow persists in the switch, that is, the length of time the attack flow appears in the pcap file, as Trident's response time to the attack. The faster Trident responds, the shorter the duration the attack flow remains in the switch, and the less damage the switch experiences. We simulate 8 sets of experiments according to the parameters in Table III to evaluate the response time of Trident. Fig. 9 shows the response time of Trident. As observed, Trident's response time ranges from 3 to 8 seconds, with an average response time of approximately 5.48 seconds across these 8 experiments. Furthermore, Trident's response time remains relatively stable despite changes in the parameters of the attack, demonstrating its robustness.

D. Overhead

A key goal of Trident is to ensure easy deployment with a low resource overhead, meaning it should not consume excessive controller resources during runtime. To evaluate Trident's overhead, we simulate 8 sets of experiments based on the parameters in Table III. Fig. 10 shows Trident's resource consumption, including CPU utilization and memory usage, where Fig. 10(a) depicts Trident's CPU utilization and Fig. 10(b) illustrates its memory usage. As observed, Trident's CPU usage remains below 10%, with an average of approximately 6%, demonstrating that it does not consume excessive controller processing resources. Additionally, Trident's memory usage hovers around 145MB, further confirming that it does not impose a heavy load on the controller's memory.

VII. Discussion

Limitations. We analyze Trident's limitations, focusing on aspects such as processing efficiency and resource overhead.

- Processing Efficiency. Trident is deployed in the control plane and relies on polling data plane switches for information. This process inevitably increases Trident's response latency and reduces its processing efficiency. Furthermore, Trident is limited to processing the polled flow table information from the data plane, which restricts its ability to perform fine-grained analysis and operations.
- Resource Overhead. The information collector module of Trident polls the switch flow table information for analysis by the detection and mitigation modules. As a result, the volume of flow table information directly impacts the amount of data to be recorded by the information collector, and a large volume of flow table information may cause Trident to consume more controller resources. Additionally, the continuous polling of flow table information by the information collector module inevitably increases the bandwidth consumption of the control channel.

Future Direction. The recently emerging programmable data plane [35] presents a promising opportunity to deploy security mechanisms directly in the data plane. By implementing the scheme against low-rate DoS attacks in data plane switches, it becomes possible to perform per-packet processing of network traffic, significantly enhancing processing efficiency while eliminating the consumption of control channel bandwidth.

VIII. Conclusion

In this paper, we propose Trident, a low-rate DoS attack mitigation scheme based on port and traffic state in SDN. Trident first identifies switches suspected of suffering from low-rate DoS attacks through port state detection, then monitors the switch's traffic state using XGBoost. Once Trident detects that a switch is under attack, it analyzes the flow information to identify the malicious flow and issues rules to the switch's flow table to block the attack traffic. The experiments show that Trident can accurately and robustly detect low-rate DoS attacks, respond quickly to mitigate them, and maintain low overhead.

References

[1] W. Zhou et al., "The security of using large language models: A survey with emphasis on chatgpt," *IEEE/CAA J. Automatica Sinica*, vol. 12, pp. 1–26, 2024.

[2] X. Feng, X. Zhu, Q.-L. Han, W. Zhou, S. Wen, and Y. Xiang, "Detecting vulnerability on IOT device firmware: A survey," *IEEE/CAA J. Automatica Sinica*, vol. 10, no. 1, pp. 25–41, 2023.

[3] H. Zhou and G. Gu, "Cerberus: Enabling efficient and effective in-network monitoring on programmable switches," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2024, pp. 4424–4439.

[4] C. Misa, R. Durairajan, A. Gupta, R. Rejaie, and W. Willinger, "Leveraging prefix structure to detect volumetric DDOS attack signatures with programmable switches," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2024, pp. 4535–4553.

[5] Z. Zhao, Z. Liu, H. Chen, F. Zhang, Z. Song, and Z. Li, "Effective ddos mitigation via ml-driven in-network traffic shaping," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 4, pp. 4271–4289, 2024.

[6] M. Yue, J. Li, Z. Wu, and M. Wang, "High-potency models of LDOS attack against cubic + red," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4950–4965, 2021.

[7] R. Xie et al., "Disrupting the sdn control channel via shared links: Attacks and countermeasures," *IEEE/ACM Trans. Netw.*, vol. 30, no. 5, pp. 2158–2172, 2022.

[8] Z. Liu, J. Yu, B. Yan, and G. Wang, "A deep 1-d CNN and bidirectional LSTM ensemble model with arbitration mechanism for LDDOS attack detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 6, no. 6, pp. 1396–1410, 2022.

[9] B. Liu, D. Tang, J. Chen, W. Liang, Y. Liu, and Q. Yang, "ERT-EDR: Online defense framework for TCP-targeted LDOS attacks in SDN," *Expert Syst. Appl.*, vol. 254, 2024, Art. no. 124356.

[10] V. Hnamte, A. A. Najar, H. Nhung-Nguyen, J. Hussain, and M. N. Sugali, "DDOS attack detection and mitigation using deep neural network in sdn environment," *Comput. Secur.*, vol. 138, 2024, Art. no. 103661.

[11] J. K. Chahal, A. Bhandari, and S. Behal, "Ddos attacks & defense mechanisms in SDN-enabled cloud: Taxonomy, review and research challenges," *Comput. Sci. Rev.*, vol. 53, 2024, Art. no. 100644.

[12] D. Tang, R. Dai, Y. Yan, K. Li, W. Liang, and Z. Qin, "When sdn meets low-rate threats: A survey of attacks and countermeasures in programmable networks," *ACM Comput. Surv.*, vol. 57, no. 4, Dec. 2024.

[13] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Commun. Surveys Tut.*, vol. 16, no. 3, pp. 1617–1634, 2014.

[14] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, Apr. 2014.

[15] A. Liatifis, P. Sarigiannidis, V. Argyriou, and T. Lagkas, "Advancing sdn from openflow to p4: A survey," *ACM Comput. Surv.*, vol. 55, no. 9, Jan. 2023.

[16] A. A. Najar and S. M. Naik, "Cyber-secure SDN: A CNN-based approach for efficient detection and mitigation of DDOS attacks," *Comput. Secur.*, vol. 139, 2024, Art. no. 103716.

[17] K. Wang, Y. Fu, X. Duan, T. Liu, and J. Xu, "Abnormal traffic detection system in SDN based on deep learning hybrid models," *Comput. Commun.*, vol. 216, pp. 183–194, 2024.

[18] D. Tang, Y. Yan, S. Zhang, J. Chen, and Z. Qin, "Performance and features: Mitigating the low-rate tcp-targeted DOS attack via SDN," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 428–444, 2022.

[19] M. Yue, Q. Yan, Z. Lu, and Z. Wu, "Ccs: A cross-plane collaboration strategy to defend against LDOS attacks in SDN," *IEEE Trans. Netw. Service Manage.*, vol. 21, no. 3, pp. 3522–3536, 2024.

[20] C. Zhang, Z. Cai, W. Chen, X. Luo, and J. Yin, "Flow level detection and filtering of low-rate DDOS," *Comput. Networks*, vol. 56, no. 15, pp. 3417–3431, 2012.

[21] X. Chen et al., "Fine-grained queue measurement in the data plane," in *Proc. 15th Int. Conf. Emerg. Netw. Experiments Technologies (CoNEXT)*. New York, NY, USA: ACM, 2019, pp. 15–29.

[22] Z. Wu, Q. Pan, M. Yue, and L. Liu, "Sequence alignment detection of TCP-targeted synchronous low-rate DOS attacks," *Comput. Networks*, vol. 152, pp. 64–77, 2019.

[23] D. Tang, S. Wang, B. Liu, W. Jin, and J. Zhang, "GASF-IPP: Detection and mitigation of ldos attack in SDN," *IEEE Trans. Services Comput.*, vol. 16, no. 5, pp. 3373–3384, 2023.

[24] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Power spectrum entropy based detection and mitigation of low-rate DOS attacks," *Comput. Networks*, vol. 136, pp. 80–94, 2018.

[25] Z. Wu, L. Zhang, and M. Yue, "Low-rate DOS attacks detection based on network multifractal," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 5, pp. 559–567, 2016.

[26] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*. New York, NY, USA: ACM, 2016, pp. 785–794.

[27] D. Tang, X. Wang, X. Li, P. Vijayakumar, and N. Kumar, "AKN-FGD: Adaptive kohonen network based fine-grained detection of LDOS

attacks," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 273–287, 2023.

[28] S. Xie, C. Xing, G. Zhang, X. Wei, and G. Hu, "Research on LDOS attack detection and defense mechanism in software defined networks," in *Security and Privacy in Social Networks and Big Data*, Y. Xiang, Z. Liu, and J. Li, Eds. Singapore: Springer Singapore, 2020, pp. 85–96.

[29] D. Tang, S. Zhang, Y. Yan, J. Chen, and Z. Qin, "Real-time detection and mitigation of LDOS attacks in the SDN using the HGB-FP algorithm," *IEEE Trans. Services Comput.*, vol. 15, no. 6, pp. 3471–3484, 2022.

[30] C. Fu, Q. Li, M. Shen, and K. Xu, "Frequency domain feature based robust malicious traffic detection," *IEEE/ACM Trans. Netw.*, vol. 31, no. 1, pp. 452–467, 2023.

[31] T. Cui, L. L. H. Andrew, M. Zukerman, and L. Tan, "Improving the fairness of fast TCP to new flows," *Commun. Lett. IEEE*, vol. 10, no. 5, pp. 414–416, 2006.

[32] H. Liu, J. Lu, X. Wang, C. Wang, R. Jia, and M. Li, "Fedup: Bridging fairness and efficiency in cross-silo federated learning," *IEEE Trans. Services Comput.*, vol. 17, no. 6, pp. 3672–3684, 2024.

[33] X. Chen et al., "Android HIV: A study of repackaging malware for evading machine-learning detection," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 987–1001, 2020.

[34] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, 2011.

[35] D. Tang, B. Liu, K. Li, S. Xiao, W. Liang, and J. Zhang, "Pluto: A robust LDOS attack defense system executing at line speed," *IEEE Trans. Dependable Secure Comput.*, early access, Dec. 25, 2024, pp. 1–18, doi: 10.1109/TDSC.2024.3522104.

Dan Tang received the Ph.D. degree from Huazhong University of Science and Technology, in 2014. He is an Associate Professor with the College of Computer Science and Electronic Engineering (CSEE), Hunan University (HNU), Changsha, China. His research interests include the areas of computer network security, computer information security, and architecture of future Internet.

Rui Dai received B.S. degree in computer science and technology from Hunan Normal University, in 2018 and M.S. degrees in computer science and technology in 2021 from Hunan University, where he is currently working toward the Ph.D. degree with the College of Computer Science and Electronic Engineering. His research interests include intrusion detection, AI for networks, and programmable networking.

Chenguang Zuo received the B.E. degree in computer science and technology from Hunan University, in 2023. He is currently working toward the master's degree with the College of Computer Science and Electronic Engineering (CSEE), Hunan University (HNU), Changsha, China. His research interests include programmable data plane and network security.

Jingwen Chen entered Hunan University, China, in 2015. She is currently working toward the master's degree with the College of Computer Science and Electronic Engineering (CSEE), Hunan University (HNU), Changsha, China. Her research direction is network information security.

Keqin Li (Fellow, IEEE) is a SUNY Distinguished Professor of Computer Science with the State University of New York. He is also a National Distinguished Professor with Hunan University, China. His research interests include fog computing and mobile edge computing, computer networking, and machine learning. He has authored or co-authored over 850 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is among the world's top 5 most influential scientists in parallel and distributed computing in terms of both single-year impact and career-long impact based on a composite indicator of Scopus citation database. Currently, he is an Associate Editor of the ACM Computing Surveys and the CCF Transactions on High Performance Computing. He is an AAIA Fellow. He is also a member of Academia Europaea (Academician of the Academy of Europe).

Zheng Qin received the Ph.D. degree in computer software and theory from Chongqin University, China, in 2001. He is a Professor of computer science and technology with Hunan University, China. He is a member of China Computer Federation (CCF) and ACM respectively. His main interests are computer network and information security, cloud computing, big data processing and software engineering. He has accumulated rich experience in products development and application services, such as in the area of financial, medical, military, and education sectors.