Low-Rate Flow Table Overflow Attacks in SDN: Model, Analysis and a Machine Learning Based Mitigation Approach

Dan Tang, Pei Tan, Yudong Yan, Keqin Li, *Fellow, IEEE*, Wei Liang, Zheng Qin, *Associate Member, IEEE*, and Jiliang Zhang

***Abstract*—Software-Defined Networking (SDN) is an emerging network architecture. The decoupled data and control plane provide flexible manageability and programmability to the network. One of the core components in SDN switches to direct traffic forwarding is the flow table, which is usually stored in the Ternary Content Addressable Memory (TCAM) with limited space and high power consumption, making the flow table a potential target for attacks. This paper examines the Low-Rate Flow Table Overflow (LRFTO) attacks, which fill the flow table and render it unavailable by continuously sending attack rules to occupy the space. We propose a quantitative model of LRFTO attacks to describe the attack behavior, analyze its difference from legitimate traffic, and summarize some key aspects and features to distinguish attack rules from legitimate rules. We also propose LRFTO-ADMS, a system that utilizes machine learning-based classification as its core to evict suspicious rules, thereby ensuring flow table availability. Experimental results show that the threat model can adapt correctly to the network environment. The proposed LRFTO-ADMS can evict attack rules with an accuracy of more than 95% and low overhead while protecting the flow table and the legitimate rules from being unavailable.**

***Index Terms*—Attack detection, attack mitigation, flow table overflow, machine learning, software-defined networking.**

Received 5 March 2025; revised 25 July 2025; accepted 7 December 2025. Date of publication 25 March 2026; date of current version 12 May 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62472153, in part by the Open Research Fund of State Key Laboratory of Internet Architecture under Grant HLW2025MS18, and in part by Hunan Provincial Natural Science Foundation of China under Grant 2025JJ50350. Recommended for acceptance by M. Correia. *(Corresponding author: Jiliang Zhang.)*

Dan Tang is with the College of Cyber Science and Technology, Hunan University (HNU), Changsha 410082, China, and also with the State Key Laboratory of Internet Architecture, Tsinghua University, Beijing 100084, China (e-mail: dtang@hnu.edu.cn).

Pei Tan, Yudong Yan, and Zheng Qin are with the College of Cyber Science and Technology, Hunan University (HNU), Changsha 410082, China (e-mail: tanpei@hnu.edu.cn; yudongyan@hnu.edu.cn; zqin@hnu.edu.cn).

Keqin Li is with Hunan University (HNU), Changsha 410082, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Wei Liang is with the School of Computer Science and Engineering, Hunan University of Science and Technology (HNUST), Xiangtan 411201, China (e-mail: wliang@hnust.edu.cn).

Jiliang Zhang is with the College of Semiconductors (College of Integrated Circuits), Hunan University (HNU), Changsha 410082, China (e-mail: zhangjiliang@hnu.edu.cn).

Digital Object Identifier 10.1109/TC.2026.3677483

I. Introduction

With the accelerated deployment of Software-Defined Networking (SDN), network management and innovation are becoming convenient. As a proposed solution for next-generation networking, SDN decouples the *Network Layer* in TCP/IP into the data plane and the control plane separately, simplifying the data forwarding while providing programmability to the control plane [1]. So far, SDN has been widely used in data center networks and Internet of things (IoT) networks [2], and SDN has improved the network performance. For instance, Google's B4 leverages SDN controllers to orchestrate traffic across its global data centers, optimizing bandwidth utilization and reducing latency. Facebook's WAN Fabric employs programmable switches under centralized control for elastic traffic engineering at a global scale. Due to the centralized control logic, SDN features a powerful controller and a simplified data plane, which rely on specific protocols for communication such as OpenFlow [3]. The control plane installs forwarding rules on the data plane via such protocols.

However, the innovative architecture of SDN also faces unique vulnerabilities [4]. Among them, the flow table space limitation is a severe security threat. Most flow tables in SDN switches rely on Ternary Content Addressable Memory (TCAM). Due to physical limitations, TCAM has small capacity and high power consumption [5], and mainstream SDN switches can only support up to several thousand rules without expansion [6]. However, the flow arrival rate in data center networking can reach tens of thousands of flows per second [7]. Once the flow table overflows, newly arrived rules will be stored in software or be denied from entering the flow table, causing the forwarding performance to deteriorate. Moreover, packets that data plane flow tables cannot process will be sent to controllers, which significantly increases the risk of control link flooding and data-to-control saturation attacks [8]. Therefore, it is crucial to ensure the flow table availability.

The intense competitiveness of the flow table space makes it a possible target for resource-consumption attacks. This paper discusses a typical attack that aims at the limited flow table space, called Low-Rate Flow Table Overflow (LRFTO). Attackers send unmatched packets to switches in a slow and stealthy mode, causing continuous new rule installation. With the attack

0018-9340 © 2026 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: HUNAN UNIVERSITY. Downloaded on May 12,2026 at 00:23:01 UTC from IEEE Xplore. Restrictions apply.

growing, the flow table can finally be occupied with malicious attack rules, making it unavailable for legitimate rules. Since LRFTO attacks only need to trigger rule installation, the low-rate attack flow can hide in legitimate traffic.

Unfortunately, most previous methods for mitigating flow table overflow, such as flow scheduling [9], rate limiting [10], and eviction strategy improving [11] focus on the overflow phenomenon itself. Although some global flow table features have been proposed, they did not analyze the fine-grained rule features and the behavioral difference between legitimate and attack flow. At present, learning technologies are widely introduced into network attack detection [12], and the detection granularity is finer [13]. As a result, another type of research considered the fine-grained rule features. However, they focus on intelligent rule eviction [14] to improve the flow table utilization efficiency, but do not consider malicious attack scenarios. To accurately identify such attacks and protect the flow tables on a fine-grained level, we still face these challenges:

- How to extract appropriate features from plenty of flow table data and accurately describe attack behavior?
- How to identify attack rules while protecting the security and reliability of legitimate traffic and flow tables?
- How to implement an efficient and low-overhead real-time mitigation mechanism for LRFTO attacks?

In response to these challenges, we comprehensively study LRFTO attacks in this paper. Given the shortcomings of previous work that only focused on the macroscopic phenomenon of flow table overflow, we propose a quantitative model to describe the attack behavior, proving LRFTO is a very threatening attack. Based on this model, we analyze the network trace, study the difference between legitimate and LRFTO attack rules in detail, and then propose three key aspects to distinguish them. We also proposed LRFTO-ADMS, an online detection and mitigation system for LRFTO attacks. The core of the system is the classification of rules based on machine learning. When the flow table overflows, the system judges the eviction priority of each rule and dynamically deletes the rules to free up the flow table.

We conduct experiments with SDN controllers and switches. Experimental results show that the threat model is in line with reality. The detection accuracy of LRFTO attack rules can be up to 95%. The LRFTO-ADMS can mitigate flow table overflow while protecting legitimate rules from being evicted. In summary, this paper has made the following contributions:

- We propose a quantitative model of flow table overflow attacks to achieve fine-grained feature analysis and classification of rules.
- We adopt various metrics to analyze the behavioral anomalies of the LRFTO attack rules.
- We propose the features to classify LRFTO attack rules.
- We propose LRFTO-ADMS, a system for real-time detection and mitigation of LRFTO attacks.

The paper structure is as follows: Section II introduces the background of flow tables and proposes the LRFTO attack model and its parameters. Section III analyzes the feature differences between legitimate and attack rules. Our design of LRFTO-ADMS is in Section IV, and Section V is the experiment. We discuss some additional issues of LRFTO-ADMS in Section VI, introduce related works in Section VII, and summarize the paper in Section VIII.

II. Background and Threat Model

In this section, we set up an attack threat model and provide a theoretical basis for traffic analysis and attack mitigation.

A. The Timeout Mechanism of Flow Table

In flow tables, each flow entry is considered a *flow* or a *rule*. Rules guide the action of the matched packets according to the `action` field. As the flow table space is limited, the flow table provides a timeout mechanism for rules. One is the *Idle Timeout* (IT), and another is the *Hard Timeout* (HT). IT declares that if a rule has not been matched or used for a specific time, it will be evicted. HT means that a rule will be evicted unconditionally within a period after its installation and usually has $HT \geq IT$. Flow tables can evict inactive rules through timeout and make space for new rule installation.

B. Attack Mechanism

LRFTO is a stealthy resource exhaustion attack that specifically targets flow table-based switching architectures, such as OpenFlow/SDN or P4 programmable switches. The attack's core mechanism relies on installing new flow entries at an extremely low rate until the switch's flow table becomes full and can no longer service legitimate sessions. To evade detection and mimic benign behavior, the adversary periodically (**Attack Period**) injects and installs a small number of new malicious flow entries (**Attack Step**) rather than initiating a sudden burst. Simultaneously, to prevent the installed malicious entries from being evicted for timing out, the attackers periodically (**Attack Period**) send packets matching those installed entries, refreshing their idle timeouts and keeping them in the flow table. By continuously maintaining old entries while gradually adding new ones, the attackers eventually force a table overflow. Even if the attackers stop installing new entries, subsequent legitimate flows that miss the table will be forwarded to the controller, thereby increasing the southbound control plane load. Should the adversary continue to inject new flows or possess an attack whose intensity (**Max Attack Strength**) significantly exceeds the table capacity, both legitimate and malicious packets will inundate the controller, further saturating the control channel. The combined effect is a marked degradation in network throughput and reliability, effectively disrupting the target network's operation.

The feasibility of LRFTO comes from three principal factors:

- **Limited hardware resources**: High-performance switching ASICs offer flow table restricted capacities (e.g., in TCAM or SRAM), making them vulnerable to gradual exhaustion by low-rate flow installations.
- **High stealth**: The packet injection rate of attackers closely resembles normal traffic fluctuations, making rate- or burst-based detection mechanisms ineffective.
- **Protocol flexibility**: Adversaries can leverage multiple protocols or custom match fields to camouflage malicious flows and evade fixed-rule inspections.

Authorized licensed use limited to: HUNAN UNIVERSITY. Downloaded on May 12,2026 at 00:23:01 UTC from IEEE Xplore. Restrictions apply.

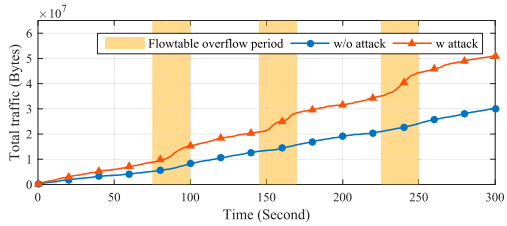


Fig. 1. Total traffic in the control channel.

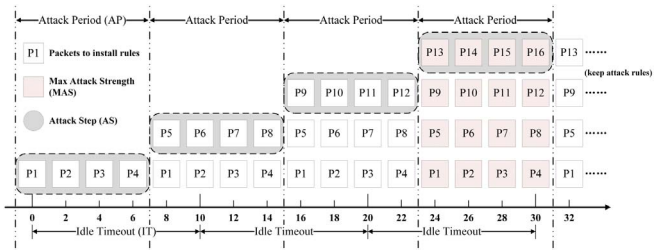


Fig. 2. The low-rate flow table overflow attack model.

To verify LRFTO's effectiveness, we conduct the experiment by replaying the real dataset [15] as normal background traffic and generating attack traffic to simulate the situation, and monitoring the control channel throughput respectively. As shown in Fig. 1, flow table overflow not only rendered the switch's flow table unusable but also caused a substantial surge (about 40%) in control plane traffic.

C. Basic Model Parameters

In this part, we propose a model with three parameters for the LRFTO attack, which is shown in Fig. 2.

- **Attack Period (AP):** To avoid attack rule eviction, the attackers need to repeat the attack packets to keep attack rules active. AP should be shorter than IT, or the attack rules will be deleted. To avoid possible detection methods, AP can be set to random. However, the closer AP is to IT, the lower the attack rate.
- **Attack Step (AS):** To finally overflow the flow table, the attackers have to keep plenty of rules in SDN, and they use AS to make the attack stealthy. When the attack starts gradually, attackers only send part of the rules. However, the attack rule sent in the next AP is more than the rules sent in the previous AP, and the growth of the attack rule number between the adjacent APs is called AS. With this growth, the flow table will finally overflow. In addition, AS can be set to random. AS distinguishes the LRFTO attacks from traditional high-rate attacks. The lower the AS, the longer the attack takes to overflow, but the more concealed the attack is.
- **Max Attack Strength (MAS):** MAS is the total number of attack rules the attackers hold and the maximum amount of rules the attackers can send. The setting of MAS should refer to the flow table size of the SDN switch to ensure that an attack with MAS rules can cause overflow.

TABLE I
VARIABLES AND DESCRIPTION IN THE MODEL

Variable	Description
ΔS	Attack rule growth of adjacent APs.
λ	Average flow arrival rate of each port.
F_u	Rule consumption under normal state.
F_f	Size of free space of flow table.
IT, HT	Idle and Hard timeout.
n	Number of ports of a switch.
S_p	Average size of attack packets.
T_f	Time to overflow.

With the parameters, we set up an attack model of LRFTO and analyze its behavior and cost. The variables are in Table I.

D. Attack Model

Suppose the victim SDN switch contains n input ports, and the arrival of new traffic flows is the same, which is λ unique new flows per second. According to the *idle timeout* mechanism of the switch, the *used flow table* F_u is as Eq.(1):

$$F_u = n \cdot \lambda \cdot IT \tag{1}$$

Then, we define M as the maximum capacity of the flow table, and we can get the average *free space of flow table* F_f under normal state as Eq.(2):

$$F_f = M - n \cdot \lambda \cdot IT \tag{2}$$

From the three parameters of the LRFTO attack, we know there is an AS difference in the number of attack rules sent in two adjacent APs. The *attack speed* (ASp) of the LRFTO attack is defined as Eq.(3), which represents the average increase of attack rule installation speed in flow tables.

$$ASp = \frac{AS}{AP} \tag{3}$$

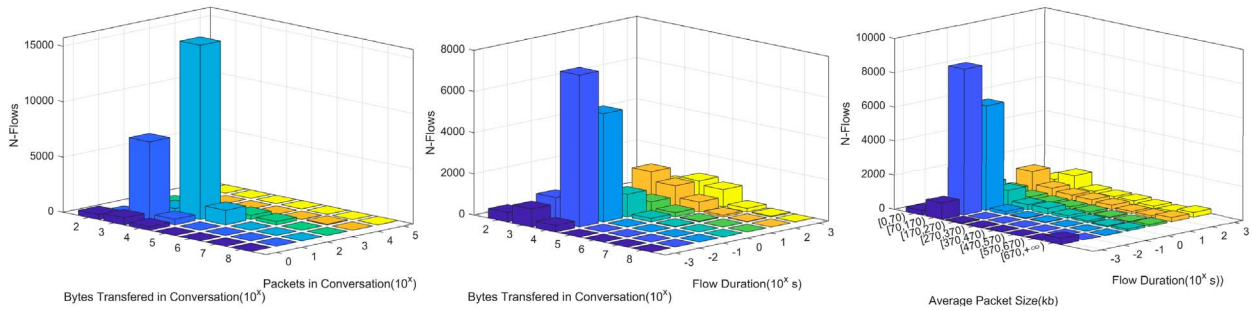
Therefore, in the i^{th} AP, there are $i \cdot ASp \cdot AP$ attack rules in the flow table. Then, it takes a certain time T_f for attackers to overflow the target switch, which is as Eq.(4):

$$T_f = \frac{M - F_u}{ASp} \tag{4}$$

However, an effective LRFTO attack needs to meet certain conditions. The first is MAS. We should ensure that the sum of attack rules is sufficient to overflow the flow table. Moreover, the higher the MAS, the better the attack effect. A larger MAS can not only occupy F_f but also, when existing rules timeout, redundant attack rules can quickly occupy the space, making the flow table unavailable when legitimate rules arrive next time. Second is the *ASp*. Considering HT settings in some SDN systems, it is necessary to keep sufficient *ASp* to overflow the flow table in each HT period [16]. Generally, a successful LRFTO attack should ensure:

$$\begin{cases} MAS \geq M - F_u \\ ASp \geq \dfrac{M - F_u}{HT} \end{cases} \tag{5}$$

Authorized licensed use limited to: HUNAN UNIVERSITY. Downloaded on May 12,2026 at 00:23:01 UTC from IEEE Xplore. Restrictions apply.



(a) Transmitted packets and bytes. (b) Flow duration and transmitted bytes. (c) Flow duration and average packet size.

Fig. 3. Statistics of traffic in each TCP/UDP conversation.

Now, we start to figure out the attack cost. Suppose the attacker sends packets with an average size of S_p to trigger new rule installation. In every adjacent APs, the total transmitted attack data is differenced by ΔS, which is as Eq.(6):

$$\Delta S = AS \cdot S_p \tag{6}$$

Once the flow table overflows, attackers should keep F_f rules in each AP, which means the attack reaches its highest rate. Therefore, the *Max Attack Flow Speed* (MAFS) can reach is as Eq.(7):

$$MAFS = \frac{S_p \cdot (M - F_u)}{AP} \tag{7}$$

Here, we examine the LRFTO attack effect. Suppose a typical flow table size 1500 [17], and legitimate rules occupy one-third of space. Then MAS should be at least 1000. If the AP is 10s (A typical setting for IT), and the AS is 40 unique packets per period, it takes 25 APs (250s) for the LRFTO attack to overflow the flow table. As a low-rate and stealthy attack, the attack flow should be as slow as possible. In python *Scapy*, it is possible to build an incomplete transport layer packet with no extra load using `Ether()/IP()`, which is only 54 bytes in size. Then, MAFS can only reach 53 Kbps, and ΔS is only about 2.1 Kbyte per AP. In conclusion, the LRFTO attack is a long-lasting and highly stealthy threat to the SDN switch and data plane.

III. Traffic Analysis

To conduct proactive rule eviction to mitigate LRFTO attacks, it is necessary to study flow patterns to determine which rule to delete. Based on network trace analysis, we propose the key aspects to classify the network rules to set up machine learning models for attack mitigation.

A. Statistical Analysis of Dataset

Here, we analyse the dataset provided by *tcpreplay* [18], called *bigFlows* [15] (we use *tcpreplay* to generate background traffic with *bigFlows* in experiment). This is a real network capture on a busy private network access point to the outer Internet. The dataset is 5 minutes long and contains 791615 packets from 40686 flows generated by 132 applications. We extracted all 27348 IPv4 TCP and UDP conversations in this trace. The statistical result is shown in Fig. 3.

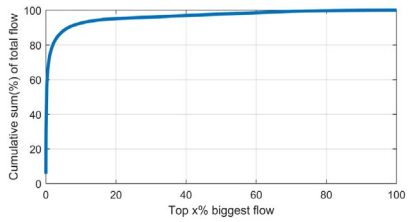


Fig. 4. The CDF distribution of the flows in the dataset.

1) Packets and Bytes in Conversation. In Fig. 3(a), most of the conversations are concentrated in the areas where the *Transmitted Packets* (TP) and *Transmitted Bytes* (TB) are both low. About 92.31% of them transmitted less than 100 packets and 10KB, while only 0.29% transmitted over 10k packets. We believe that the network trace is mostly *mice flows*.

2) Bytes and Conversation Duration. In Fig. 3(b), most of the conversations have a short *Duration Time* (DT), and 74.01% conversations lasted no more than 1s. We notice that DT and TB follow a proportional relationship generally. However, 0.72% of conversations lasted for less than 1s but transmitted over 100KB of data. For these high-speed flows, we consider it an "urgent" *elephant flow*, which we will discuss later.

3) Packet Size and Conversation Duration. Most of the conversations in Fig. 3(c) have a small *Average Packet Size* (APS) and a short DT. In 71.00% conversations, APS is less than 370 bytes, and the DT is below 1s. These conversations have little data and can be called *mice flows*. However, some flows lasted for longer with a bigger APS, which means that there may be more data to transmit and more probably to be considered as *elephant flow*.

4) Data Size of Conversation. We calculated the *Cumulative Distribution Function* (CDF) of all the data transmitted by the conversations in this trace, as shown in Fig. 4. We can conclude that only a small number of flows comprise most of the TB in the network transmission. Using this property, we will classify the *elephant flow* and *mice flow*.

B. Mice and Elephant Flow

Before we analyze the necessity to classify legitimate traffic into mice and elephant flows for mitigating LRFTO attacks, we

Authorized licensed use limited to: HUNAN UNIVERSITY. Downloaded on May 12,2026 at 00:23:01 UTC from IEEE Xplore. Restrictions apply.

TABLE II
SPCF, DPCF PROPORTION AND THE BIGGEST FLOW IN SOME TRACES

Trace	SpCF(%)	DpCF(%)	Biggest Flow (packet, byte)
bigFlows	5.86%	11.02%	40.7 K, 19.76 M
LBNL [19]	33.32%	3.70%	173.2 K, 103.16 M
UNI1 [20]	2.20%	59.45%	26.5 K, 32.55 M
UNI2	27.12%	17.73%	3099.9 K, 2656.66 M
WIDE [21]	88.43%	2.87%	108.3 K, 227.43 M

look into two auxiliary concepts, namely *Single-packet Conversation Flow* (SpCF) and *Double-packet Conversation Flow* (DpCF). SpCF means the conversation contains a one-packet flow from the client to the server or from the server to the client. DpCF means a conversation with two packets, one from the client to the server and the other from the server to the client.

In OpenFlow, `Packet-In` and `Flow-Mod` will install a new rule as a new packet arrives. An SpCF will install one rule, and a DpCF will install two. However, as there is only one packet in the flows, once the SpCF or DpCF installs its rule, it will never be used. The rules will have to wait for an IT to be evicted, which causes a waste of the flow table space. Therefore, SpCF and DpCF present an extreme flow table space waste situation. According to real network traces, the proportion of SpCF and DpCF is considerable, as shown in Table II. Except for SpCF and DpCF, the data and packets transmitted by most of the conversations in the network are low, and these are probably all *mice flows*, which may also cause flow table space waste. However, the biggest flow in a network can transmit millions or billions of bytes. As a result, a dataset that labels both minority huge *elephant flows* and plenty of SpCF/DpCF-like *mice flows* as *legitimate* is not consistent with this imbalance in a real network. Plenty of *mice flows* may mislead the machine learning model, making it unable to accurately identify the features of *elephant flow*, which fails to provide sufficient protection for *elephant flow* from being evicted under LRFTO attacks.

To classify *mice* and *elephant flows*, according to previous work [22], we consider both flow urgency and flow size to decide the threshold. We define the flows with top 20% *Transmission Speed* (TS) as *elephant* and the rest 80% as *mice*. We label the flows according to their protocol, source-destination IP and source-destination transport-layer ports tuple `<ip_proto, ip_src, port_src, ip_dst, port_dst>`. TS is calculated with the trace file, and because the flow table does not record the first arrived packet once it is not matched, we do not use TS as a feature in machine learning.

C. Attack Flow Analysis

We analyze the LRFTO attack flow features and discuss the difference between attack and legitimate flow for classification.

1) Flow Duration. Different from the legitimate flow, which has a short DT, the LRFTO attack flow lasts for much longer, aiming to take up a significant amount of space in the flow table continuously. Unless HT clears the rules, the LRFTO attack rules will never expire.

2) Flow Size. The difference between LRFTO attacks and other high-rate attacks lies in their low rate. The APS of the LRFTO flow should be low to install plenty of rules while ensuring a low attack rate. Moreover, during the LRFTO attack, the packets match the attack rule continuously, so even if the TP and TB are small, it is not SpCF or DpCF. Based on this, it is possible to distinguish legitimate *mice flow* and the *mice flow* based LRFTO attack flow.

3) Flow Behavior. In each AP, an attack rule will only be matched once to avoid timeout, and the setting of AP is based on IT. Therefore, the arrival of packets in the LRFTO attack flow will appear more orderly, and the *Average Packet Arrival Interval* (APAI) can be larger than the legitimate flow.

D. Classification of Elephant, Mice and Attack

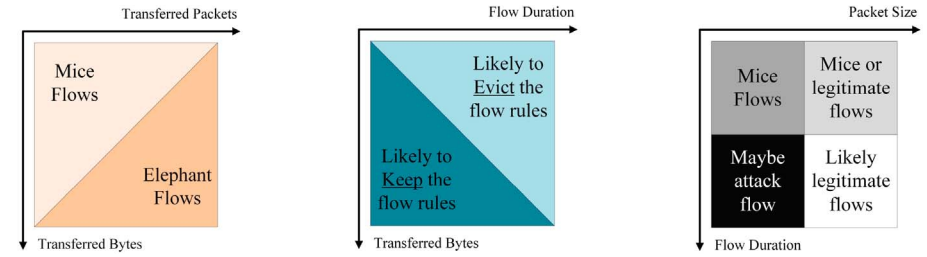
We summarize the features for the rule classification into the following points, which are also shown in Fig. 5.

- **TP and TB: Mice and Elephants**. These two features indicate the amount of data transmitted by the rule, which is the key to distinguishing legitimate traffic into *mice* and *elephant*. Although the label of mice and elephants is predetermined by TS before training, a larger TP and TB indicate that the flow is more likely to be *elephant flow*, as shown in Fig. 5(a).
- **DT and TB: Keep or Evict.** As shown in Fig. 5(b), we illustrate what kind of rule will be kept or evicted in the two-feature dimensions of DT and TB. We believe it is "efficient" if a rule transmits more data in a shorter DT. We should keep such rules in the flow table to avoid eviction, which will cause additional control plane delays to install rules. Conversely, if the DT of the rule is long and the TB is small, it means that the efficiency of this rule is low. When the flow table overflows, the inefficient rule has a higher eviction priority.
- **APS and DT: Rule Classification.** The LRFTO attack flow has a low APS, and the APS of legitimate traffic generated by simple network operations that do not transmit large amounts of data is also low. However, in the rules with low APS, the *mice flows* usually have a short DT, while the DT of LRFTO attack rules is long. When the flow APS is high, it is more likely to be a legitimate rule. In this case, the longer the DT, the more likely the rule is to transmit more data, that is, the *elephant flow*. Based on this, we can get a classification of network rules. Different classes of rules have different priorities when being evicted, as shown in Fig. 5(c).

IV. SYSTEM DESIGN

In this section, we propose the LRFTO attack detection and mitigation system (LRFTO-ADMS) based on machine learning. The system consists of four modules: *Rule Monitoring, Data Processing, Model Setup and Detection, Attack Mitigation*. We analyze the features of the rules and use the labeled data to train a rule classification model offline. Then, it is deployed on the

Authorized licensed use limited to: HUNAN UNIVERSITY. Downloaded on May 12,2026 at 00:23:01 UTC from IEEE Xplore. Restrictions apply.



(a) Mice and elephant flows. (b) Eviction strategy based on efficiency. (c) A possible classfication of flow rules.

Fig. 5. A classification and probable management of network flow rules.

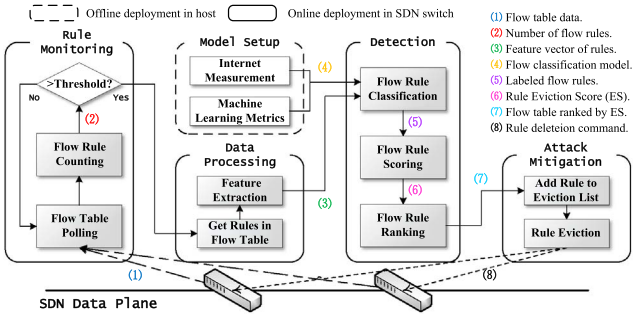


Fig. 6. The overall framework of the proposed LRFTO attack detection and mitigation system (LRFTO-ADMS).

SDN switch for attack detection and mitigation. The workflow of the system is shown in Fig. 6.

A. Design Goals

Before the design of LRFTO-ADMS, We first clarify its design goals based on the LRFTO attack features.

- **Reliability.** A complete design is needed to reproduce the theoretical basis fully during detection and mitigation.
- **Protection.** The system should identify and delete attack rules in time to mitigate LRFTO attacks.
- **Versatility.** In addition to LRFTO attack mitigation, the system should support rule eviction under normal conditions. When the rules are too much to support, the system should also identify and evict "inactive" rules.
- **QoS-aware.** Since the LRFTO-ADMS is based on machine learning to classify rules, prediction, model deployment, and rule deletion are all time-consuming. System optimization is an essential requirement.
- **Error Tolerance.** The LRFTO attack has a macroscopic target to overflow the flow tables. Therefore, classification and mitigation do not need to be completely accurate. Microscopic errors, such as few misclassifications, should be allowed and do not affect the system's functions.

B. Rule Monitoring Module

The rule monitoring module is designed to obtain the rules in the flow tables. Since flow tables do not overflow most of the time in normal situations, we ignore the rule legality in this module. Instead, we only count the number of rules to avoid frequent, time-consuming operations such as rule data processing and classification. The module compares the obtained rule count with a preset threshold. Once the rule count exceeds the threshold, it indicates the flow table may be about to overflow. Then, the data processing module is activated to prepare rule feature data for further attack mitigation operations.

In the experimental environment, we use shell scripts with the OpenvSwitch command to process the rule sum. This process is executed every 1s. To ensure the legitimate flows can enter the switch as much as possible and ensure the normal function of the flow table, we set the threshold higher based on experience. It is 90% of the maximum flow table capacity.

C. Data Processing Module

When the rule monitoring module finds that the flow table is about to overflow, we further process the rule data instead of focusing on the rule count. We extract features from each rule using a Python script for machine learning to further classify rules and later evict rules. Based on the theory in Section IV, we extract a rule's five features as the basis for rule classification.

- **Duration Time (DT):** The attack rule's DT can be long, while the legitimate rule's DT can be irregular. If the legitimate rules also have a long DT, the amount of the data transmitted by this rule is more likely to be larger, compared with the *mice flow* based LRFTO attack flow.
- **Transmitted Packets (TP):** TP is the total matched packets of a rule. The greater the TP, the more likely it is an *elephant flow*, otherwise, it may be a *mice flow*.
- **Transmitted Bytes (TB):** TB is the total matched bytes of a rule, which is similar as TP. TP can be used to distinguish *elephants flow* and *mice flow*.
- **Average Packet Arrival Interval (APAI):** APAI indicates the rule activity. The smaller the APAI, the more active the rule, which may be an urgent *elephant flow*. Otherwise, the less active the rule is, the more likely it is an inefficient *mice flow*. The LRFTO attack behavior also makes the APAI of attack rule large. Therefore, APAI is the ratio of TP and DT.
- **Average Packet Size (APS):** APS may be related to *Type of Service* (ToS). A larger APS usually indicates network data transmission, which may be considered as *elephant flow*. In comparison, a smaller APS is more likely to be

Authorized licensed use limited to: HUNAN UNIVERSITY. Downloaded on May 12,2026 at 00:23:01 UTC from IEEE Xplore. Restrictions apply.

simple operations such as authentication and connection establishment, which may be considered as *mice flow*. According to the LRFTO attack behavior, the low-rate attack usually has a low APS to lower the attack rate. APS is the ratio of TB and TP.

Since flow tables keep record of DT, TP, and TB, and APAI and APS can be calculated with the three features. The low computing complexity can guarantee the module efficiency.

D. Model Setup and Detection Module

We set up a rule classification model based on machine learning based on the feature vector <DT, TP, TB, APAI, APS> and the labeled rules. The reason for using machine learning is based on Fig. 5, we know that the rule classification is a complex, multi-feature fusion and linearly inseparable problem. Therefore, due to the ability to solve nonlinear problems, process high-dimensional datasets, and strong generalization capabilities, machine learning is suitable for classifying rules. In this paper, we use different types of machine learning algorithms in the python module library `scikit-learn` [23] to verify the classification performance of rules. We extract the feature vector of each rule as the input of the machine learning model and classify it into labels 0, 1, and 2. Among them, 0 represents *elephant flow*, 1 represents *mice flow*, and 2 represents *attack flow*.

We score and rank the rules to perform further rule eviction and attack mitigation. Firstly, we define that each rule has an *Eviction Score* (ES), a metric that indicates the likelihood of the rule being evicted. The higher the ES, the more likely the rule is to be deleted. To evict the attack rules as much as possible while protecting the legitimate rules, we need to ensure that the ES of the *elephant flow* is the smallest (to avoid downstream Table-Miss penalties), followed by the *mice flow*, and the *attack flow* is the largest (to make room for the legitimate). We use the `predict_proba` function in `scikit-learn`. For the above-mentioned triple-classification problem, it will output a triple-vector P_e, P_m, P_a, representing the probability that the rule belongs to three categories. We calculate the ES with a set of weights $w = \{w_e, w_m, w_a\}$, which is shown in Eq.(8).

$$ES = \sum_{i=0}^{i\in\{e,m,a\}} P_i \cdot w_i \tag{8}$$

In this paper, we set $w = \{-1, 1, 2\}$, and finally, we can get an ES in [-1, 2]. Due to the weight difference, we ensure the ES of *attack flow* is higher than that of *mice flow*. The ES of *mice flow* is also higher than *elephant flow*, which conforms to the priority of rule eviction, which we discussed before. Through this mechanism, when the flow table overflows, it can identify which rules should be evicted most frequently.

E. Attack Mitigation Module

The attack mitigation module consists of two steps. The first step is to determine the rules that need to be deleted. Specifically, it adds those rules to a *eviction list*. The second step is to delete the rules and release the space on the flow table. The eviction list contains a series of rules to be deleted. Specifically, the rules in the eviction list only include the protocol and source-destination tuple in the matching field, that is, <ip_proto, ip_src, port_src, ip_dst, port_dst>, and the list identifies a specific rule based on this tuple. The rules in the eviction list are extracted from the flow table and sorted by ES in the detection module. There are two management modes for the eviction list.

- **Maintenance.** Extracts all the rules with a predicted label 2-*Attack flow* in each IT period to implement dynamic deletion of suspicious rules.
- **Mitigation.** When the number of rules exceeds the threshold, not only are the rules predicted as *attack flow* extracted, but also a certain proportion of the flow table space will be forcibly released, called *Flow Eviction Ratio* (FER). We delete rules according to ES from high to low and finally delete the rules with the top FER% highest ES. If the number of *attack flow* is insufficient, part of the *mice flow* will also be deleted.

Finally, the LRFTO-ADMS can deploy on SDN switches. We can eliminate rules without the overhead of control messages by using the switch command line. Specifically, we use the `ovs-ofctl del-flows <src-dst tuple>` command to remove each rule from the eviction list. The deletion operation can be multi-threading to enhance the mitigation.

V. Experiment

A. Experiment Environment

We set up the experiments on two hosts. We deployed the Python-based controller Ryu version 4.3.0 [24] and the network simulator Mininet version 2.3.0d6 [25] on the virtual machine on different hosts, and they use *remote controller* for communication. To more comprehensively assess the practical performance of our proposed mechanism, we select a partial structure of a real-world topology from Topology Zoo named *Internode* for experiments, with 2 switches (the targets of the attack whose flow tables may overflow due to the LRFTO attack) and 5 hosts (simulating benign users, attackers, and the party receiving the traffic, respectively). There is a bottleneck link between the two switches *S1, S2* of 250Mbps bandwidth and 3.0379ms delay, and other local links are of 1Gbps bandwidth and 0ms delay. The software switch is OpenvSwitch 2.5.9.The controller host CPU is Intel Core i5-4670K, and the Mininet host CPU is Intel Xeon E5-2680v4. Both hosts run Ubuntu 16.04.06 LTS and have a RAM of 16GB.

The controller runs a *layer-4 learning switch* application, of which match fields include IP source, destination and port. According to Section V-E, the number of rules to evict is determined by FER, not the capacity of the flow tables. Therefore, the capacity of the flow table is fixed at 4000, and the *idle timeout* is set to 10s.

B. Attack Implementation and Dataset

1) Attack Implementation. We launched LRFTO attacks with three sets of parameters listed in Table III. The attack is initiated

Authorized licensed use limited to: HUNAN UNIVERSITY. Downloaded on May 12,2026 at 00:23:01 UTC from IEEE Xplore. Restrictions apply.

TABLE III
PARAMETERS OF THE LRFTO ATTACKER

Attack No.	AP(s)	AS(pkt/AP)	MAS(pkt)	S_p(KB)
1	[5, 8]	75	4000	[54, 144]
2	[6, 9]	100	4000	[54, 168]
3	[5, 9.5]	150	4000	[54, 192]

TABLE IV
FEATURES AND IMPORTANCE

Classfier \ Feature	DT	TP	TB	APAI	APS
Decision tree	0.2363	0.1041	0.1545	0.3845	0.1206
AdaBoost	0.1000	0.0800	0.0800	0.3400	0.4000
Random forest	0.2438	0.1251	0.1433	0.3845	0.1033
GBDT	0.2098	0.1303	0.1461	0.2851	0.2288

with feature of *mice flow*. To generate the attack flow, we transmit packets to several destination ports using Scapy*Scapy*.

2) Dataset. To obtain the legitimate rule data, we replayed the *bigFlows.pcap* with *tcpreplay* at the rate of 1000 packets per second between host *h1* and *h5* through link *S1-S2*, and the LRFTO-ADMS polled request on the flow table of *S1* and *S2*. The polling period is the same as IT, avoiding data duplication of inactive rules. We replayed the 5-minute dataset periodically, and the process of obtaining legitimate rule data lasted for 1500s, during which we got 150 flow table records containing 383,693 legitimate rules.

For the attacked rule data, we launched LRFTO attacks at host *h2* and *h3* to *h4* to overflow switch *S1* and *S2*. Each attack group lasted for 240s, and we record the flow table every 10s. We got 72 flow table records in the three groups, containing 283,896 LRFTO attack rules. The legitimate and attack rules are labeled by their source IP address.

For legitimate rules containing both *elephant* and *mice flow*, we label the rules with top 20% transmission speed as *elephant*, and the rest 80% as *mice*. Two-thirds of the rules are in the training set, with the test set being the remaining portion.

C. Rule Classification Result

1) Feature Importance. Firstly, we test the effectiveness of the five features for rule classification using four machine learning techniques. We used the `feature_importance` option in `scikit-learn` to obtain the importance score of each feature, and the results are shown in Table IV.

From Table IV, we can see that for different machine learning models, `APAI` is the essential feature caused by the abnormal LRFTO attack behavior. To resist the timeout mechanism and reduce the attack rate, the *attack flow* has a larger `APAI`, while in the legitimate *elephant* and *mice flows*, `APAI` has no regularity. Secondly, `DT` and `APS` are also important, for LRFTO attacks usually require a long duration and a small APS to affect the flow table. Finally, `TP` and `TB` are mainly used to distinguish *elephant* and *mice* in legitimate rules, while LRFTO attacks are launched based on *mice flow*. Therefore, `TP` and `TB` cannot distinguish all three classes, so they are not as critical as the previous three features.

2) Learning Models. We use machine learning to build classification models based on training data. We use six machine learning models for training, as shown in Table V. Among them, there are four kinds of different learning technologies, including `K-Nearest Neighbour`, linear and nonlinear classification `Logistic Regression`, `Random Forest`, neural network-based `MultiLayer Perceptron`, ensemble learning-based `AdaBoost` and `Gradient Boosting Decision Tree`.

3) Evaluation Metrics. We assess the models' classification performance using a variety of metrics. As a triple-classification task, for each category, we use *Recall* and *Error Rate* (Er) for evaluation. *Recall* represents the proportion of the correct classification in each category, and Er is the proportion of misclassification into another category, presenting by Er-XF. Er-XF is the proportion of rules that are misclassified into XF class, where X can be E, M, or A, representing *elephant*, *mice*, and *attack* respectively, and F is short for *flow*. For the overall model performance, we use *Accuracy* and *F1-score* to evaluate. Among them, *Accuracy* is the proportion of all the rules in the three categories that are correctly classified, and *F1-score* represents the balance of *Precision* and *Recall*. We use *Macro-F1* here. In addition, we use *Training Time* (TT) to evaluate the time consumption of the model and *Model Size* (MS) to evaluate the model complexity. TT and MS are tested in Conda version 4.5.11 (Python 3.5) on a Windows 10-20H2 host with Intel Core i5-4670K CPU and 32GB RAM.

4) Results. The classification results of different models are shown in Table V. All the models have a high *Recall* for *elephant flow*, except for Adaboot with 13.93% Er-MF. Therefore, the machine learning models can accurately identify "important and urgent" rules based on the five features. As for *mice flow* and *attack flow*, the behavioral features have similar parts. The TP, TB, and APS may be low, so there are more mutual classification errors between the two types. However, due to APAI, the *mice* and *attack* flow classification can also have an acceptable *Accuracy*, which meets the design goals of *Error Tolerance*, for both *mice* and attack *flow* should be less protected than *elephant* when the flow table overflows.

We found that the TT and MS of different models are quite different. The fastest model LR requires only 5.6s of training, and the MS is less than 1KB. However, this model is too simple to perform classification accurately. In addition, the TT of ensemble learning models is longer, and due to the large `n_estimator`, the MS of RF can reach about 195MB. Considering both *Accuracy* and MS, we choose the GBDT with the highest *Accuracy* and an acceptable MS as the online rule classifier.

D. Offline Rule Eviction Experiment

Before deploying our system on SDN switches, we first extract the flow table under the LRFTO attack with different MAS parameters and analyze the performance of rule eviction offline.

Authorized licensed use limited to: HUNAN UNIVERSITY. Downloaded on May 12,2026 at 00:23:01 UTC from IEEE Xplore. Restrictions apply.

TABLE V
DIFFERENT CLASSIFIERS AND THE RULE CLASSIFICATION RESULTS

Classifier	Elephant Flow(EF)			Mice Flow(MF)			Attack Flow(AF)			Overall			
	Recall	Er-MF	Er-AF	Recall	Er-EF	Er-AF	Recall	Er-EF	Er-MF	Acc	F1	TT(s)	MS(KB)
KNN	98.60%	1.32%	0.09%	96.58%	0.78%	2.64%	90.95%	0.12%	8.93%	0.944	0.94	24.1	41946
LR	98.94%	1.02%	0.04%	95.61%	0.29%	4.10%	86.39%	0.13%	13.49%	0.919	0.92	**5.6**	**1**
RF	99.58%	**0.21%**	0.21%	95.22%	**0.04%**	4.74%	90.62%	0.11%	9.27%	0.937	0.94	67.9	195162
MLP	98.40%	1.60%	**0.00%**	94.28%	2.38%	3.34%	89.92%	1.15%	8.94%	0.928	0.93	12.5	34
AdaBoost	86.05%	13.93%	0.02%	82.13%	0.08%	17.80%	91.21%	**0.10%**	8.70%	0.865	0.87	24.9	64
GBDT	**99.60%**	0.39%	0.01%	**97.51%**	0.05%	**2.44%**	**91.60%**	0.11%	**8.29%**	**0.952**	**0.95**	151.6	1633

Then we discuss the performance of the ES indicator, which is used for rule eviction and deletion online.

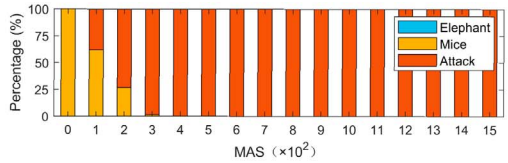
1) Offline Rule Eviction. We launch LRFTO attacks with different MAS in [0, 1500] (AS = 100) to flow tables, and we discuss the FER in [10%, 30%]. Fig. 7 shows the rule component with the highest FER% ES under the attack of MAS strength, which are the rules that should be evicted. The larger the MAS, the more *attack flows* are evicted. When the MAS exceeds 1200, almost all deleted rules are *attack flow*, for the weighted ES makes the *attack flow* get the highest scores. In addition, when the FER is high, more legitimate rules can be deleted along with attack rules. However, we can allow some "inefficient" *mice flows* to be evicted together. We should set a proper FER to get a balance of attack mitigation and legitimate rule eviction. Finally, under different attack situations, the proportion of *elephant flows* being evicted is shallow, which shows that they are well protected.

2) Eviction Score Distribution. We calculated the ES of all test set rules and got the ES *Probability Distribution Function* (PDF) shown in Fig. 8. The ES of *elephant*, *mice*, and *attack flow* are concentrated in three areas around -1, 1, and 2, equivalenting to the ES weighting parameters. Therefore, the ES-based rule ranking mechanism can well reflect the different eviction priorities, providing a theoretical basis for LRFTO-ADMS deployment.

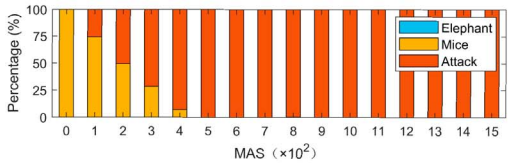
3) Labels and Eviction Score. We further explain why we set three tags (*0-elephant, 1-mice, 2-attack*) instead of two tags (*0-legitimate, 1-attack*) for rule classification from the perspective of ES. Suppose we label the rules with two tags (label both *elephant* and *mice* as *legitimate*). The output of `predict_proba` on each rule will reduce to two dimensions P_n, P_a. Therefore, the ES calculation will be modified as Eq.(9):

$$ES = P_n + 2P_a \tag{9}$$

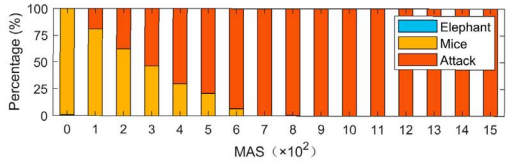
Similarly, we use the models in Table V for the classification with two and three labels, respectively. Then, we calculate the average ES of the *elephant*, *mice*, and *attack* under the two schemes. For the three-label scheme, we calculate the average ES of each class according to Eq.(8). For the two-label scheme, although we calculate ES based on Eq.(9), the rules that initially belonged to *elephant* and *mice* in *legitimate* class are calculated separately with Eq.(8) for ES average. Table VI shows the



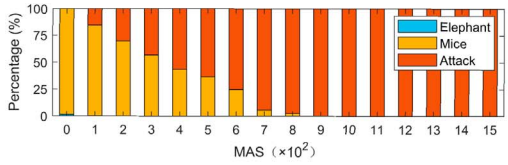
(a) FER=10%.



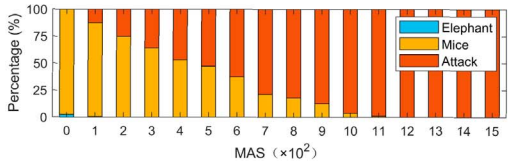
(b) FER=15%.



(c) FER=20%.



(d) FER=25%.



(e) FER=30%.

Fig. 7. Components of the rules in flow table with top FER% ES.

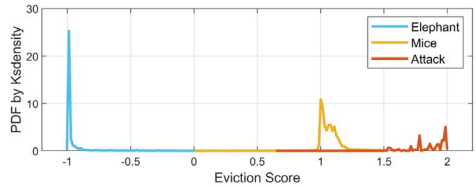


Fig. 8. The PDF curve of ES in three classes of rules.

Authorized licensed use limited to: HUNAN UNIVERSITY. Downloaded on May 12,2026 at 00:23:01 UTC from IEEE Xplore. Restrictions apply.

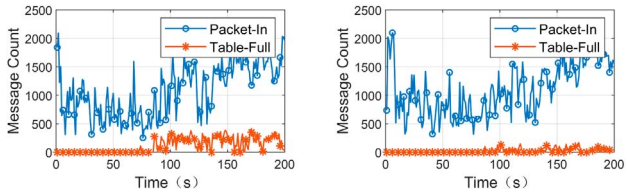
TABLE VI
THE COMPARISON ON CLASSIFICATION: TWO LABELS OR THREE LABELS

Classifier	2 Labels ($ES = P_n + 2P_a$)					3 Labels ($ES = -P_e + P_m + 2P_a$)				
	Accuracy	EA(EF)	EA(MF)	EA(AF)	EA(MF)-EA(EF)	Accuracy	EA(EF)	EA(MF)	EA(AF)	EA(MF)-EA(EF)
KNN	0.949	1.0043	1.0392	1.9053	0.0349	0.944	-0.9365	1.0218	1.9031	1.9583
LR	0.916	1.0266	1.2273	1.8263	0.2007	0.919	-0.5407	1.2494	1.6841	1.7901
RF	0.937	1.0046	1.0677	1.9047	0.0631	0.937	-0.9755	1.0650	1.9031	2.0405
MLP	0.946	1.0048	1.0957	1.9013	0.0909	0.928	-0.7855	1.0384	1.8736	1.8239
AdaBoost	0.899	1.3821	1.4384	1.5744	0.0563	0.865	0.2486	1.0817	1.2451	0.8331
GBDT	0.953	1.0066	1.0843	1.9070	0.0777	0.952	-0.9683	1.0797	1.9051	2.0480

result. In Table VI, EA is short for *ES Average*, and EF, MF, and AF also short for *elephant*, *mice*, and *attack* flow.

From Table VI, we can find that the rule classification *accuracy* under the 2-label scheme is slightly improved compared with the 3-label scheme, which shows that the classification is easier with fewer labels. However, we can further conclude in Section IV-B that the 2-label scheme can easily mislead the machine learning model and lacks protection against the *elephant flow* eviction. When *elephant* and *mice* are both trained as *legitimate*, the ES gap between them is extremely small and is even less than 0.1 points. Therefore, when performing rule eviction based on ES sorting, the probability of *elephant* being ranked in front of *mice* is significantly increased. However, when using the 3-label scheme, the ES gap between *elephant* and *mice* can reach up to 2 points, which provides sufficient protection on *elephant flow*.

We notice that the average ES of *mice flow* has little gap under the 2-label and 3-label schemes. This phenomenon shows that the 2-label makes machine learning models focus on the majority *mice flow*, but fail to notice a few urgent *elephant flow*. Based on the above facts, we believe that the LRFTO-ADMS proposed in this paper can also execute intelligent rule eviction in a non-attack state.



(a) Without Mitigation. (b) With Mitigation.

Fig. 9. The number of packet-in and table-full messages.

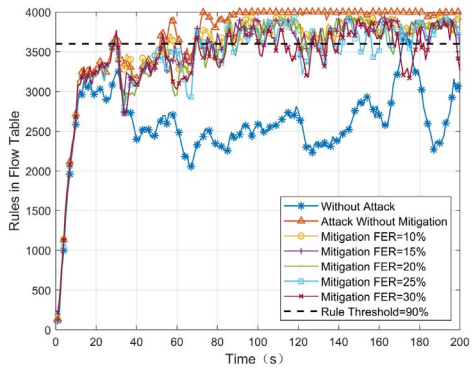


Fig. 10. Number of rules in flow table (AP $\in [5, 9]$, AS $= 150$, MAS $=$ 2000).

E. Online Attack Mitigation

In this part, we deploy the LRFTO-ADMS on the SDN switch to assess the performance of real-time attack mitigation. We will evaluate the performance from five aspects.

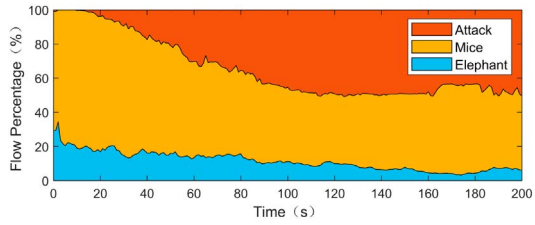
1) Control Message. In Fig. 9, we can see that the LRFTO-ADMS deployment has little effect on `Packet-In` because LFRTO attacks do not generate flooding attack packets. However, the `Table-Full` decreases sharply with LRFTO-ADMS, indicating no long-term flow table overflow. In addition, `Packet-In` messages increase when `Table-Full` appears due to flow table space grabbing by legitimate rules and attack rules.

2) Number of Rules. We set different FER to mitigate the LRFTO attack. The number of rules in the flow table is shown in Fig. 10. When the attack accumulates enough MAS, the flow table always overflows without LRFTO-ADMS, but it doesn't with LRFTO-ADMS. The number of rules stays near the threshold. The number of rules decreases and fluctuates much more significantly with higher FER. However, the rule sum is still higher than in the non-attack situation. This is because the *Error Tolerance* design goal allows more mice flow to enter the flow table when it is not overflowing.

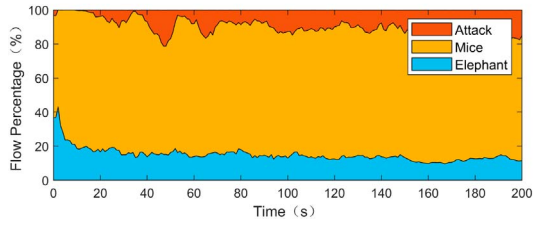
3) Flow Table Composition. We extracted the real-time flow table to verify the protection effect provided by LRFTO-ADMS. As shown in Fig. 11, under LRFTO attacks, legitimate rules are squeezed by the *attack flow*, and their proportion is gradually reduced. After deploying LRFTO-ADMS, the rule component in flow tables is more reasonable. For *attack flows* that are evicted in time, their proportion has dropped significantly, and most of the remaining rules are legitimate flows, indicating that the flow table can be effectively safeguarded by the suggested system. In addition, the proportion of *elephant flow* did not decrease significantly with mitigation, indicating that the LRFTO-ADMS can also protect urgent, effective rules from being evicted.

4) Resource Consumption. We assessed LRFTO-ADMS's system cost, and the results are shown in Fig. 12. Although the

Authorized licensed use limited to: HUNAN UNIVERSITY. Downloaded on May 12,2026 at 00:23:01 UTC from IEEE Xplore. Restrictions apply.

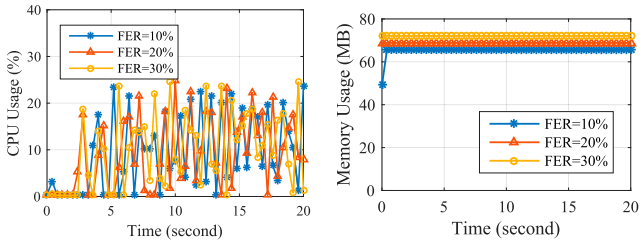


(a) Without Mitigation.



(b) With Mitigation.

Fig. 11. Proportion of predicted label of rules in the flow table (AP $\in [5, 9]$, AS $= 150$, MAS $= 2000$, FER $= 25\%$).



(a) CPU Usage. (b) Memory Usage.

Fig. 12. The resource consumption of LRFTO-ADMS.

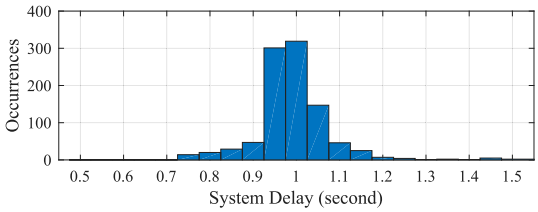


Fig. 13. The system delay of LRFTO-ADMS.

FER setting ranges, the difference in CPU usage was insignificant. When deleting rules, the CPU overhead is 1%-25%, and when monitoring the rule sum, the overhead is about 0.3%. In addition, under different FER settings, the memory usage of LRFTO-ADMS is the same(0.4%), which is 66MB.

5) System Delay. We document the system delay (SD) of 970 attack mitigation operations performed by LRFTO-ADMS, where SD refers to the total time required for flow table feature extraction, sorting, and deletion after activation of the "Data Processing and Attack Detection Module". The distribution of SD is shown in Fig. 13. As illustrated, most attack mitigation operations can be completed within 1.2s, with an average SD of 0.991s.

F. Comparison

We conduct comparative experiments with LRFTO-ADMS against several state-of-the-art approaches: Random Idle Timeout (RIT), Idle and Hard Timeout (IaHT), the native OpenvSwitch Eviction Mechanism (EVICT), Selective Defense for TCAM (SIFT) [11], Small-flow Analysis and Import-flow Analysis (SAIA) [26], and TableGuard [27]. For all the approaches, the attacker launches the LRFTO attack using the parameters in Table III for 200s, with background traffic identical to that in our earlier tests. For LRFTO-ADMS, we set the FER at 20% and define 'flow table overflow' as occupancy above 98% of the capacity. The results are shown in Table VII.

From Table VII, the most striking advantage of LRFTO-ADMS is its ability to eject attack flows: With our method deployed, the average relative attack flow share in the table falls most to 2.05%, a reduction of 93.4% from 30.97% in the unmitigated case. Moreover, LRFTO-ADMS yields the best Maximum relative attack flow occupancy and average relative elephant flow occupancy, demonstrating both strong mitigation and interpretability. Under LRFTO-ADMS, the average flow table occupancy drops to 85.44%, and the overflow duration is reduced to just 19s, outperforming the no-defense baseline, RIT, IaHT, and EVICT. The average attack flow lifetime is the shortest at 10.364s, representing a 92.5% decrease compared to no defense. LRFTO-ADMS also reduces the data-plane Table Miss packet count by 8.88% and the byte volume by 24.1%, indicating lower control plane overhead and a lighter SDN controller load.

Although EVICT and SAIA achieve metrics close to LRFTO-ADMS, EVICT expels rules based on the least recently used (LRU) if no priority principle is specified. LRU cannot distinguish between attack and legitimate flows at a feature level, nor can it adapt to new flow types, resulting in higher attack flow occupancy than LRFTO-ADMS. SAIA, though, performs well in attack flow lifetime, employs control plane polling that significantly burdens the control channel. Control plane methods such as SIFT, SAIA, and TableGuard increase Table Miss traffic because they must store and continuously update every rule in the controller, sending heavy update streams over the control link and risking controller saturation. These results demonstrate that embedding real-time, high-volume flow table logic directly into the data plane is both more secure and more performant than control plane defenses. And LRFTO-ADMS delivers superior and efficient mitigation of LRFTO attacks while optimizing SDN flow table utilization.

VI. Discussion and Limitations

Security of LRFTO-ADMS. The LRFTO-ADMS proposed in this paper is deployed on the switch, relying on the command line to extract and delete rules. Therefore, the network administrator must give the LRFTO-ADMS a high authority (root in OpenvSwitch). This high authority provides LRFTO-ADMS considerable security guarantees on the simplified SDN data plane, as regular users may find it challenging to approach. However, the rule prediction and deletion operations can be time-consuming, and the extracted flow table is a snapshot.

Authorized licensed use limited to: HUNAN UNIVERSITY. Downloaded on May 12,2026 at 00:23:01 UTC from IEEE Xplore. Restrictions apply.

TABLE VII
COMPARISON OF ATTACK MITIGATION PERFORMANCE

Approach	Deployment location	Average relative AF	MAS relative AF	Average relative EF	Average FT occupancy	Overflow duration	Average AF lifetime	Table Miss (PKTs)	Table Miss (Bytes)
no defense	None	30.97%	55.01%	16.63%	91.17%	100s	138.094s	70710	8739077
RIT	DP	30.56%	57.07%	17.45%	88.53%	94s	72.259s	64449	7593298
IaHT	DP	17.24%	30.19%	19.07%	88.08%	66s	29.260s	62911	6640204
EVICT	DP	10.63%	33.86%	21.51%	92.15%	120s	17.853s	52508	4987485
SIFT	CP	20.39%	60.03%	17.76%	87.47%	72s	38.094s	91959	18257439
SAIA	CP	3.34%	11.81%	22.23%	88.44%	73s	12.422s	81856	8768635
TableGuard	CP	18.25%	37.20%	17.78%	86.13%	23s	24.119s	163102	30528637
LRFTO–ADMS	DP	2.05%	11.24%	22.68%	85.44%	19s	10.364s	64428	6630058

Therefore, if an attacker launches flooding DoS attacks or deliberately causes network delay (e.g., *Reduce of Quality* (RoQ) attack [28], [29]) and its variants [30]. The reliability of the LRFTO-ADMS may be reduced.

Limitations. The LRFTO-ADMS still has some shortcomings. The first is the performance. The whole prediction and deletion operations on rules usually take a few seconds. When there are a lot of rules that need to be removed and the attack rate is high, there may be insufficient rule deletion and defense delay. The second is early detection. For example, the LRFTO attack only sends one packet in its first AP. The installed *attack rule* is a SpCF, which may easily cause error classification as *mice flow* according to the models. Most of the Er-MFs under the *Attack Flow* column in Table V fall into this situation. Therefore, the ability of early LRFTO attack detection still needs to be improved.

VII. RELATED WORK

SDN Data Plane threats. As reported in the literature [31], the SDN data plane faces three primary security threats. First is Denial-of-Service. Attackers can launch a DoS attack through a data-to-control plane saturation mode [32] or a shared link [33] between the control and data plane. The second is topology poisoning [34]. Attackers can interfere with control message delivery (e.g., LLDP) to poison SDN global information [35] or inject harmful messages into the data plane [36]. The third is the data leaking attacks. It has been proved that attackers can infer SDN configurations (e.g., timeout, delay) through reconnaissance methods [37], which puts SDN at risk of information leakage.

Rule Eviction in SDN. In addition to the IT and HT, researchers have proposed some intelligent rule eviction schemes to use the limited flow table space efficiently. Some methods aim to improve traditional eviction algorithms (e.g., LRU, FIFO). The *Dynamic Least Frequently Used* (DLFU) [38] improves the LRU with the status of SDN switches by control messages. The *Learning-To-Rank* (LtR) based eviction methods [39] rank malicious traffic according to eviction priority and remove malicious traffic with high priority, which improves the interpretability of the mitigation problem. Another approach is to transform the rule eviction into a knapsack problem, then optimize an objective function to minimize flow table occupancy [14]. Finally, machine learning efforts have been made to use flow tables effectively, and technologies such as Random Forest and hidden Markov model [40] are mentioned.

As far as we are aware, *STEREOS* [41] is the work that is most comparable to ours, in which the relationship between rule features and machine learning is studied in detail. However, *STEREOS* only focuses on effectively using flow tables and rule eviction, but not on mitigating malicious overflow attacks. However, the LRFTO-ADMS proposed in this paper can simultaneously implement rule eviction and attack mitigation.

Mitigate Flow Table Overflow. Flow table overflow attack mitigation techniques currently available fall into two categories, namely *passive defense* and *active defense*.

Typically, passive protection schedules or evicts malicious rules rather than actively deleting them or stopping the attack at its source. Such methods can use `Table-Full` as a signal to initiate defense. Typical examples are peer support [9], selective defense (SIFT) [11] and priority scheduling [42]. Since these techniques only require a few attack detection metrics, they are easy to comprehend, put into practice and deploy. However, many attack rules can be installed into flow tables under passive safeguards, and the attack mitigation may not be sufficient.

Active protection strategies include identifying possible attacks beforehand and removing or blocking attack rules proactively. Typical methods include rule classification and deletion [17], flow rate limiting [10] and adaptive routing management [43]. These approaches could be considered as early attack detection, and the mitigation can be sufficient. These techniques, nevertheless, typically necessitate processing substantial amounts of rule-related data in real-time (e.g., all of the rules in flow tables), which raises mitigation latency and system overhead.

VIII. CONCLUSION

This study examined LRFTO, one of the low-rate attacks that target overflowing the flow table on the SDN data plane. We first proposed three parameters of the LRFTO attack model: AP, AS, and MAS, and established the time and cost model, proving that the LRFTO attack can overflow the flow table at a shallow attack rate. Then, we analyzed some traffic traces. Based on the analysis of Internet measurement, we divided the rules in flow

Authorized licensed use limited to: HUNAN UNIVERSITY. Downloaded on May 12,2026 at 00:23:01 UTC from IEEE Xplore. Restrictions apply.

tables into three categories: *elephant*, *mice*, and *attack flow*. We proposed five features for rule classification by discussing the distinctions between the LRFTO attack rule and the valid rule from the perspective of traffic features and behavior. We also proposed LRFTO-ADMS, which performs dynamic monitoring and deletion of rules to mitigate LRFTO attacks. According to experimental findings, our proposed LRFTO-ADMS can be deployed on SDN switches with low system overhead to effectively mitigate LRFTO attacks and can be applied to legitimate traffic to protect the security of data transmission and flow tables.

In future work, we will further improve LRFTO-ADMS, including but not limited to new rule classification methods and features, new rule management mechanisms, and system performance improvements with more sophisticated and efficient ML models. We also will discuss the security of LRFTO-ADMS more comprehensively to ensure its availability under other vulnerabilities or attacks. Finally, we will also discuss the new deployment modes of LRFTO-ADMS and study the feasibility and performance of its deployment based on OpenFlow improvement and embedded deployment.

REFERENCES

[1] D. Tang, X. Wang, P. Tan, Z. Qin, K. Li, and J. Zhang, "DNSGreen: A comprehensive defense system against bounce-style DNS DDOS attacks with p4," *IEEE Trans. Comput.*, vol. 75, no. 1, pp. 247–260, Jan. 2026.

[2] R. U. Mageswari et al., "Addressing security challenges in industry 4.0: Ava-ma approach for strengthening SDN-IOT network security," *Comput. Secur.*, vol. 144, 2024, Art. no. 103907.

[3] N. McKeown et al., "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[4] 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, pp. 1–32, 2024.

[5] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[6] P. Zhang, H. Wang, C. Hu, and C. Lin, "On denial of service attacks in software defined networks," *IEEE Netw.*, vol. 30, no. 6, pp. 28–33, Nov. 2016.

[7] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. Internet Measur.*, 2010, pp. 267–280.

[8] M. Zhang, G. Li, L. Xu, J. Bi, G. Gu, and J. Bai, Control plane reflection attacks in SDNs: New attacks and countermeasures," in *Proc. Int. Symp. Res. in Attacks, Intrusions, Defenses*. Berlin, Germany: Springer, 2018, pp. 161–183.

[9] B. Yuan, D. Zou, S. Yu, H. Jin, W. Qiang, and J. Shen, "Defending against flow table overloading attack in software-defined networks," *IEEE Trans. Services Comput.*, vol. 12, no. 2, pp. 231–246, Mar./Apr. 2016.

[10] T. Xu, D. Gao, P. Dong, C. H. Foh, and H. Zhang, "Mitigating the table-overflow attack in software-defined networking," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 4, pp. 1086–1097, Dec. 2017.

[11] T. A. Pascoal, I. E. Fonseca, and V. Nigam, "Slow denial-of-service attacks on software defined networks," *Comput. Networks*, vol. 173, 2020, Art. no. 107223.

[12] 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, Jan. 2022.

[13] R. Dai, D. Tang, Z. Qin, K. Chen, K. Li, and J. Zhang, "Detecting congestion-related attacks via fine-grained queue diagnosis," *IEEE Trans. Cogn. Commun. Netw.*, vol. 12, pp. 1255–1268, 2026.

[14] S. Shirali-Shahreza and Y. Ganjali, "Delayed installation and expedited eviction: An alternative approach to reduce flow table occupancy in SDN switches," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1547–1561, Aug. 2018.

[15] Tcpreplay, "bigflows.pcap," Accessed: Jul. 22, 2025. [Online]. Available: http://tcpreplay.appneta.com/wiki/captures.html

[16] J. Cao, M. Xu, Q. Li, K. Sun, Y. Yang, and J. Zheng, "Disrupting SDN via the data plane: A low-rate flow table overflow attack," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.*, 2017, pp. 356–376.

[17] W. Zhijun, X. Qing, W. Jingjie, Y. Meng, and L. Liang, "Low-rate DDoS attack detection based on factorization machine in software defined network," *IEEE Access*, vol. 8, pp. 17404–17418, 2020.

[18] Tcpreplay, "tcpreplay," Accessed: Jul. 22, 2025. [Online]. Available: https://github.com/appneta/tcpreplay/

[19] "lbl-internal.20050107-1323.port026.dump.anon," Accessed: Jul. 22, 2025. [Online]. Available: ftp://ftp.bro-ids.org/enterprise-traces/hdr-traces05/

[20] "Data set for IMC 2010 data center measurement," Accessed: Jul. 22, 2025. [Online]. Available: https://pages.cs.wisc.edu/~tbenson/IMC10_Data.html

[21] "Wide project: Traffic of sample point-f (fragment of the first minute)," Accessed: Jul. 22, 2025. [Online]. Available: http://mawi.wide.ad.jp/mawi/samplepoint-F/2020/202010121400.html

[22] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 254–265.

[23] "scikit-learn," Accessed: Jul. 22, 2025. [Online]. Available: https://scikit-learn.org/stable/

[24] "Ryu SDN controller," Accessed: Jul. 22, 2025. [Online]. Available: https://github.com/faucetsdn/ryu

[25] "Mininet," Accessed: Jul. 22, 2025. [Online]. Available: https://github.com/mininet/mininet/

[26] S. Xie, C. Xing, G. Zhang, and J. Zhao, "A table overflow LDoS attack defending mechanism in software-defined networks," *Secur. Commun. Networks*, vol. 2021, no. 1, 2021, Art. no. 6667922.

[27] D. Kong, C. Wu, Y. Shen, X. Chen, H. Liu, and D. Zhang, "TableGuard: A novel security mechanism against flow table overflow attacks in SDN," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*. Piscataway, NJ, USA: IEEE Press, 2022, pp. 4167–4172.

[28] D. Tang et al., "MWD-CFM: Detection and mitigation of DDoS attack against SDN flow tables," *IEEE Trans. Netw.*, vol. 34, pp. 4269–4282, 2026.

[29] 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.*, vol. 22, no. 3, pp. 2855–2872, May/Jun. 2025.

[30] D. Tang, R. Dai, C. Zuo, J. Chen, K. Li, and Z. Qin, "A low-rate dos attack mitigation scheme based on port and traffic state in SDN," *IEEE Trans. Comput.*, vol. 74, no. 5, pp. 1758–1770, May 2025.

[31] S. Gao, Z. Li, B. Xiao, and G. Wei, "Security threats in the data plane of software-defined networks," *IEEE Netw.*, vol. 32, no. 4, pp. 108–113, Jul./Aug. 2018.

[32] 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.

[33] 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.

[34] S. Deng, X. Qing, X. Li, X. Gao, and X. Gao, "SDN application backdoor: Disrupting the service via poisoning the topology," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2023, pp. 1–10.

[35] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: New attacks and countermeasures," *NDSS*, vol. 15, pp. 8–11, Feb. 2015.

[36] F. Xiao, J. Zhang, J. Huang, G. Gu, D. Wu, and P. Liu, "Unexpected data dependency creation and chaining: A new attack to SDN," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2020, pp. 1512–1526.

[37] A. Patwardhan, D. Jayarama, N. Limaye, S. Vidhale, Z. Parekh, and K. Harfoush, "SDN security: Information disclosure and flow table overflow attacks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM),* 2019, pp. 1–6.

[38] H. Luo, W. Li, Y. Qian, and L. Dou, "Mitigating SDN flow table overflow," in *Proc. IEEE 42nd Annu. Computer Softw. IEEE Int. Symp. Spread Spectr. Tech. Appl. Conf. (COMPSAC)*, 2018, vol. 1, pp. 821–822.

Authorized licensed use limited to: HUNAN UNIVERSITY. Downloaded on May 12,2026 at 00:23:01 UTC from IEEE Xplore. Restrictions apply.

[39] D. Tang, Y. Yan, C. Gao, W. Liang, and W. Jin, "LtRFT: Mitigate the low-rate data plane DDoS attack with learning-to-rank enabled flow tables," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 3143–3157, 2023.
[40] G. Huang and H. Y. Youn, "Proactive eviction of flow entry for SDN based on hidden Markov model," *Front. Comput. Sci.*, vol. 14, no. 4, pp. 1–10, 2020.
[41] H. Yang, G. F. Riley, and D. M. Blough, "STEREOS: Smart table entry eviction for openflow switches," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 377–388, Feb. 2019.
[42] M. Zhang, J. Bi, J. Bai, Z. Dong, Y. Li, and Z. Li, "FTGuard: A priority-aware strategy against the flow table overflow attack in SDN," in *Proc. SIGCOMM Posters Demos*, 2017, pp. 141–143.
[43] Z. Guo, R. Liu, Y. Xu, A. Gushchin, A. Walid, and H. J. Chao, "Star: Preventing flow-table overflow in software-defined networks," *Comput. Networks*, vol. 125, pp. 15–25, Oct. 2017.



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 Cyber Science and Technology, Hunan University (HNU), Changsha, China. His research interests include computer network security, computer information security, and architecture of future internet.



Pei Tan received the B.E. degree in computer science and technology from Hunan University, Changsha, China, in 2024. She is currently working toward the Ph.D. degree with the College of Cyber Science and Technology, Hunan University. Her research interests include programmable network and network security.



Yudong Yan received master's degree in computer science from the College of Computer Science and Electronic Engineering (CSEE), Hunan University (HNU), Changsha, China, in 2023. He is currently working as a Software Engineer with the Software Technology Center Asia (STCA), Microsoft China Company Ltd. His research interests include software-defined networking and network attack detection.



Keqin Li (Fellow, IEEE) is a SUNY Distinguished Professor with the Computer Science, State University of New York and also a National Distinguished Professor with Hunan University, Changsha, China. His research interests include cloud computing, high-performance computing, computer networking, and machine learning. He is listed in Scilit Top Cited Scholars (2023–2025) and is among the top 0.02% out of over 20 million scholars worldwide based on top-cited publications in the last ten years. He is an AAAS Fellow and an AIIA Fellow. He is a member of the European Academy of Sciences and Arts and the Academia Europaea (Academician of the Academy of Europe).



Wei Liang received the Ph.D. degree in computer science and technology from Hunan University, Changsha, China, in 2013. From 2014 to 2016, he was a Postdoctoral Scholar with Lehigh University, Bethlehem, PA, USA. He is currently a Professor with the School of Computer Science and Engineering, Hunan University of Science and Technology. His research interests include blockchain security technology, network security protection, embedded system and hardware IP protection, fog computing, and security management in wireless sensor networks.



Zheng Qin (Associate Member, IEEE) received the Ph.D. degree in computer software and theory from Chongqing University, China, in 2001. He is a Professor with the Computer Science and Technology, Hunan University, Changsha, China. He is a member of China Computer Federation (CCF) and ACM. He has accumulated rich experience in products development and application services, such as in the area of financial, medical, military and education sectors. His research interests include computer network and information security, cloud computing, big data processing, and software engineering.



Jiliang Zhang received the Ph.D. degree in computer science and technology from Hunan University, Changsha, China, in 2015. He is currently a Full Professor with the College of Integrated Circuits, Hunan University. He is the Vice Dean of the College of Integrated Circuits, Hunan University, the Director of Chip Security Institute of Hunan University, and the Secretary-General of CCF Fault-Tolerant Computing Professional Committee. His research interests include hardware security, integrated circuit design, and intelligent system. He has been the Program Committee Member for a number of well-known conferences such as DAC, ASP-DAC, GLSVLSI, and FPT. He is a Senior Member of CCF.

Authorized licensed use limited to: HUNAN UNIVERSITY. Downloaded on May 12,2026 at 00:23:01 UTC from IEEE Xplore. Restrictions apply.