FTMaster: A Detection and Mitigation System of Low-Rate Flow Table Overflow Attacks via SDN

Dan Tang, Chenjun Gao, Wei Liang, Jiliang Zhang, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—Software-defined networking (SDN) faces challenges in efficiently forwarding packets across the network due to the limited capacity of flow tables in the switches. Ternary content addressable memory (TCAM) is typically used to store flow tables, but its limited capacity makes it vulnerable to attacks. Specifically, the Low-rate Flow Table Overflow (LFTO) attack is an attack against the flow table capacity limit, which can occupy massive space in the flow table to decrease the forwarding performance of normal flow rules by slowly sending packets that cannot match the flow table. To address this, we propose the FTMaster, a system to monitor, detect and mitigate LFTO attacks based on machine learning. FTMaster monitors and detects the flow table state by analyzing the features of flow tables. Once the LFTO attack is detected, FTMaster will activate the mitigation module to extract and analyze the features of each flow rule, evict attack flows, and ultimately block the attack source, thereby protecting flow tables and normal flows. Experimental results demonstrate that FTMaster enables real-time LFTO attack detection and mitigation, ensuring normal forwarding and availability of flow tables.**

***Index Terms*—Attack detection, attack mitigation, low-rate flow table overflow attacks, software-defined networking.**

I. Introduction

SOFTWARE-DEFINED Networking (SDN) is a centralized control network architecture. It is an implementation of network virtualization that allows the definition and control of networks in software programming. OpenFlow is one of the core technologies of SDN, which enables flexible network traffic control and makes the network more intelligent by separating network devices in the control plane from the data plane.

Manuscript received 17 July 2022; revised 20 December 2022 and 18 April 2023; accepted 20 April 2023. Date of publication 25 April 2023; date of current version 12 December 2023. This work was supported in part by the National Key R&D Program of China under Grant 2020YFB1713400, in part by the National Natural Science Foundation of China under Grants 62122023, in part by the Science and Technology Key Projects of Changsha City under Grant kq2208038, and in part by the Natural Science Foundation of Fujian Province under Grant No. 2021J01544. The associate editor coordinating the review of this article and approving it for publication was K. Xue. *(Corresponding author: Jiliang Zhang.)*

Dan Tang and Chenjun Gao are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: Dtang@hnu.edu.cn; gaochenjun@hnu.edu.cn).

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

Jiliang Zhang is with the College of Semiconductors (College of Integrated Circuits), Hunan University, Changsha, China, and also with the Innovation Institute of Industrial Design and Machine Intelligence Quanzhou-Hunan University, Quanzhou, China. (e-mail: zhangjiliang@hnu.edu.cn).

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

Digital Object Identifier 10.1109/TNSM.2023.3270339

Despite its innovative architecture, SDN is vulnerable to some targeted threats [1], [2], [3], such as those directed at the flow table of the data plane. The flow table stores some flow rules responsible for packet lookup and forwarding, and its limited capacity is a significant challenge for SDN. Flow tables in most SDN switches rely on the ternary content addressable memory (TCAM) with a limited capacity due to its high manufacturing cost. Mainstream SDN switches can only support thousands of rules [4], while data center traffic rates can reach tens of thousands of flows per second. Therefore, flow tables are relatively easy to overflow, which will cause flow tables cannot install legal rules or delete installed legal flow rules, resulting in degraded forwarding performance. In addition, packets that cannot be handled by the flow table will be sent to the controller, which greatly increases the load on the control channel and the risk of data-control saturation attacks. Therefore, it is crucial to maintain network performance to effectively and efficiently manage flow tables.

The low-rate flow table overflow (LFTO) attack [5] is a common attack that targets the flow table capacity limit. LFTO attackers will continuously and periodically send forged packets to SDN switches to install malicious flow rules, invade and occupy the flow table space for normal flow rules and ultimately cause it to overflow. LFTO attacks are periodic and stealthy. Its average attack rate is low and can be easily hidden in normal flows, which makes it a challenge to mitigate it.

Currently, the mainstream countermeasures against flow table overflow are weakly explained and mainly focus on mitigating the macroscopic phenomenon of flow table overflows, such as rate limiting [6], flow scheduling [7] and eviction policy improvement [8]. With the development of anomaly detection [9], [10], some researchers have proposed global or flow-level features from a fine-grained perspective, but they are not comprehensive enough. Moreover, other types of research focus more on the optimization of flow table utilization without considering malicious attack scenarios [11]. Most research for malicious attack scenarios focuses on high-rate attacks and does not consider low-rate attacks with similar attack effects.

In this paper, we propose the FTMaster, a system to detect and mitigate LFTO attacks. FTMaster is composed of three modules that are responsible for monitoring, detecting, and mitigating LFTO attacks, respectively. The monitor module tracks the real-time average count of flow rules and activates the detection module when the count exceeds the detection threshold. The detection module extracts the statistical features of flow tables to judge whether LFTO attacks occur in the flow tables. The mitigation module extracts six features of each flow rule to classify the normal and attack flows, evict the attack

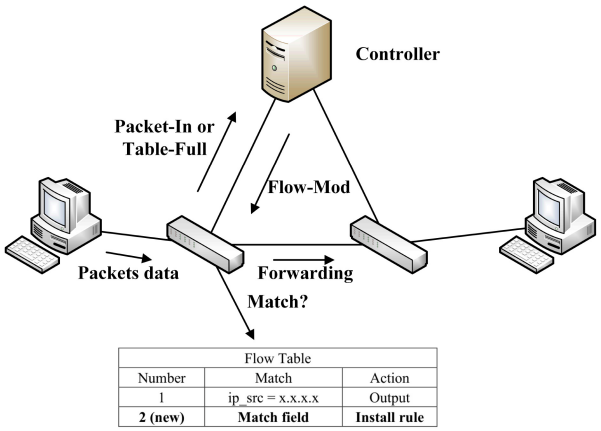


Fig. 1. The workflow of SDN switches.

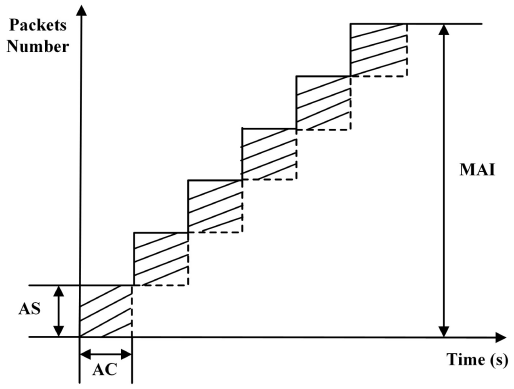


Fig. 2. LFTO attack model.

flows, and ultimately block the attack source. We conducted experiments to evaluate FTMaster's performance in detecting and mitigating LFTO attacks, and the results indicate that it performs well. We summarize our contributions as follows.

- We model the LFTO attack and implement it in SDN.
- We propose three new features of the flow tables for the first time and classify them by combining them with some regular statistical features.
- We evict attack flows by feature extraction and analysis of flow rules, create the blacklist of suspicious attack IPs, and finally block the attack source.
- We propose the FTMaster, a system designed to monitor, detect and mitigate LFTO attacks.

Our paper is structured as follows. Section II provides the background information. Section III is related work, including the SDN data plane security and the mitigation of table overflow attacks. Section IV describes the three modules that comprise FTMaster in detail. Section V is the experiments and evaluation. Section VI is the summary and discussion.

II. Background

A. SDN and Openflow

SDN is based on two fundamental concepts: highly centralized controller management and a decoupled separation of the data plane and the control plane. To achieve the latter, it is necessary to establish a communication interface standard between the controller and the forwarding devices called southbound interface protocols, while the OpenFlow protocol [12] is one of the most commonly used protocols. It works by having network devices process messages according to the OpenFlow flow table, which is created and maintained by the controller. There are some flow entries in each flow table, and each entry is treated as a flow rule, mainly composed of match fields, instructions, etc. When the OpenFlow switch receives a packet, it will be matched with the flow rules. If the matching is successful, the instructions are then executed. The instructions perform actions or pipeline processing, such as forwarding and discarding data packets.

Fig. 1 shows the workflow of SDN switches. When a host sends the packet data to a switch, the switch will query the flow table for rule matching. If the matching is failed, the flow will be considered a new flow, and a `Packet-In` message will be sent to the controller to request the processing of the packet data. Then, the controller will then send a `Flow-Mod` command to install new flow rules on the switch. However, if the flow table is already full, the switch will send a `Table-Full` signal to the controller, and this packet will be forwarded to the controller for further processing.

B. Timeout Mechanism of Flow Table

To compensate for the limitations in flow table space, the timeout mechanisms for flow rules are provided. Each flow rule can be allocated a fixed valid time by the controller. Old and expired flow rules will be automatically cleaned up to free up more space for new flow rules.

The fixed valid time can be divided into hard timeout and idle timeout. The hard timeout indicates that a rule will be unconditionally deleted sometime after installation. Then idle timeout indicates that if a rule is not matched or used for a certain period, it will be deleted. Due to the inflexibility of hard timeouts, the controller usually sets idle timeouts for flow rules instead of hard timeouts. That is, the hard timeout cannot distinguish whether the flow rules are valid, resulting in ineffective utilization of the flow table space. Besides, setting idle timeouts is also a widely discussed problem. Too low or too high idle timeouts will increase unnecessary costs and overhead. Therefore, it is essential to set a proper idle timeout based on different traffic features. However, the OpenFlow protocol does not provide a feasible solution to calculate the appropriate idle timeout.

C. LFTO Attacks

To provide theoretical preparation for attack detection and mitigation, we introduce the threat model of LFTO attacks. The match field of LFTO attacks is the 5-tuples, including `protocol, port_src, port_dst, ip_src, and ip_dst`. Fig. 2 shows the LFTO attack model. The LFTO attack has three parameters: attack cycle, attack step, and Maximum attack intensity. We explain them below in detail.

Attack cycle (AC): To avoid the switch deleting the attack rule due to timeouts, the AC of the LFTO attack needs to be shorter than the idle timeout. LFTO attackers need to repeat

the attack packet in each AC to ensure that attack rules remain active. For example, if the idle timeout is set to 10 seconds, the AC of the LFTO attack would typically be set to [5, 9.5] seconds. The closer AC is to the idle timeout, the lower the attack rate is.

Attack step (AS): To maintain the attack rules established in the flow table while gradually increasing the number of new attack rules, LFTO attackers need to send more packets than the previous AC each time, and the number of attack rules growing between adjacent ACs is called AS. The lower AS is, the longer the time required for attack overflow, and the more stealthy the attack is.

Maximum Attack Intensity (MAI): To maintain the effectiveness and concealment of the LFTO attack, the upper limit of the number of attack rules needs to be regulated. The setting of MAI refers to the flow table capacity of the SDN switch to ensure that the attack can cause overflow.

We suppose the victim SDN switch contains m input ports, and the arrival rate of new flows is n per second. According to the idle timeout (IT) mechanism of the switch, the flow table space that has been used (FT_u) can be calculated as:

$$FT_u = m \times n \times IT \tag{1}$$

From AS and AC, we can know that the average increase of the speed (AIS) of attack rules installation in flow tables can be calculated as:

$$AIS = \frac{AS}{AC} \tag{2}$$

Then, we can calculate the time for attackers to overflow the target switch, as shown in Eq. (3), where FT_{size} is the size of the flow table.

$$T_{overflow} = \frac{FT_{size} - FT_u}{AIS} \tag{3}$$

To ensure the effectiveness of the LFTO attacks, we need to ensure the settings of MAI and AIS. For MAI, it must ensure malicious flows can overflow the flow table. Besides, when some attack flow rules are evicted by the timeout mechanism, the remaining attack flow rules can still occupy the flow table space. Therefore, MAI should be greater than the remaining capacity of flow tables. For AIS, we need to consider the hard timeout (HT) mechanism of SDN. Therefore, the attackers should overflow the flow table in each period of HT, and then AIS should be set as:

$$AIS \geq \frac{FT_{size} - FT_u}{HT} \tag{4}$$

III. Related Work

A. SDN Data Plane Security

The SDN data plane is composed of OpenFlow switches, making it susceptible to some security threats, which can be divided into three types [13].

1) DoS Attacks: DoS attackers can consume the resources of controllers and switches by flooding the flow table with unmatched packets [14], [15], [16]. Tang et al. proposed a series of methods against the low-rate DoS attacks in SDN, such as P&F [17], GASF-IPP [18], and PeakSAX [19]. Zheng et al. [20] proposed a Distributed DoS mitigation system, Reinforcing Anti-DDoS Actions in Realtime, which can use adaptive correlation analysis to locate attackers.

2) Topology Poisoning Attacks: Topology poisoning attackers can forge links between switches that do not exist by forging or relaying Link Layer Discovery Protocol (LLDP) packets [21]. Marin et al. [22] made a comprehensive analysis of the security of research on topology attacks. Shrivastava and Kataoka [23] introduced a new type of topology poisoning attack via hybrid SDNs, and discussed the countermeasures of the attack detection. Skowyra et al. [24] designed a new system, called TopoGuard+, which can defend *port amnesia* and *port probing* that TopoGuard [25] and Sphinx [26] can not work. Smyth et al. [27] explored a new mechanism for topology poisoning attacks based on programmable switches.

3) Side-Channel Attacks: Side-channel attackers can steal details about the network configuration by reconnaissance or timing technologies. Shoaib et al. [28] randomizes response timing to conceal the original response time from side-channel attackers. Conti et al. [29] used network flows obfuscation to prevent the attackers from collecting the network configuration information. Liu et al. [30] protected SDN switches from reconnaissance attacks by a Markov model.

B. Mitigation of Flow Table Overflow Attacks

So far, there are lots of research on flow table overflow attacks. Except for the LFTO attack proposed in [5], there is a series of similar research on probing the network configuration and triggering the flow table overflow attacks. SDNMap [31] actively probes and monitors the network traffic to precisely reconstruct the detailed flow rules, which could strengthen the LFTO attack. Yu et al. [32] develop an intelligent attack model, which infers the internal configuration and state of the flow table according to its specific cache-like behaviors and then designs attack parameters. The control plane reflection attack proposed in [33] forces the controller to send costly control messages to the SDN hardware switches by exploiting their limited processing power.

Most mitigation strategies for flow table overflow attacks ultimately mitigate attacks by evicting or scheduling flow rules to free up the space of flow tables, regardless of whether they are active or passive.

Researchers generally improve the timeout and traditional eviction mechanisms for the flow rules eviction strategies. Due to the limitation of the fixed timeout mechanism, some adaptive timeout allocation mechanisms have been proposed [34], [35], [36]. Due to the limitations of common eviction algorithms like the Least recently used (LRU) and the First In First Out (FIFO) mechanisms, researchers have proposed some improved eviction algorithms. The dynamic least frequently used (DLFU) mechanism improved the LRU to mitigate the flow table overflow threats [37]. The adaptive least frequently evicted (ALFE) mechanism can identify, protect and give priority to the elephant flows [38]. FTGuard [39] assign different priorities to each flow, and suspected flows with lower priority will be first evicted. SIFT [8] evicts flow rules with a certain probability after the flow table overflows and

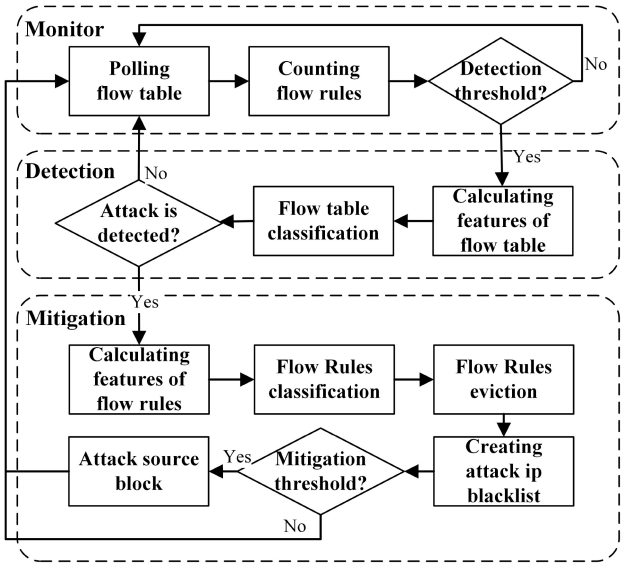


Fig. 3. The workflow of FTMaster.

`Table-Full` error arrives. FTS [40] can improve the bad performance caused by flow table overflow. This kind of strategy is low-complexity and easy to implement. However, many attack flows usually are allowed to occupy the flow table space for a certain period and making normal flows inaccessible.

Another kind of strategy advocate pre-detecting potential attacks [41], [42] and actively deleting or blocking malicious flow rules to protect flow tables. Soylu et al. [43] used network function virtualization (NFV) to dynamically filter LFTO attackers. Xu et al. [6] proposed a mitigation strategy by using the token bucket to limit the LFTO attack rate and avoid flow table overflow. Xie et al. [44] designed a prediction and deletion strategy to detect and mitigate LFTO attacks. This kind of strategy effectively avoids the impact of LFTO attacks by identifying and evicting malicious flow rules or limiting the rate of malicious flows. However, most of them require massive real-time data collection, which causes higher overhead and complexity.

IV. The Detection and Mitigation Method

A. System Overview

We propose a system called FTMaster to monitor, detect and mitigate LFTO attacks. The workflow of FTMaster is shown in Fig. 3. FTMaster consists of three modules as follows. The monitor module polls flow tables and count the average number of flow rules in the latest sequence of flow tables. If the count exceeds the detection threshold, the detection module will be started. The detection module first calculates the features of flow tables and makes the flow table classification. If LFTO attacks occur in flow tables, the mitigation module will calculate the features of each flow rule, make flow rule classification, and evict the attack flow rules. At the same time, a blacklist of suspicious source IPs will be created. If the times of the same source IP is deleted exceeds the mitigation threshold, the attack source will be blocked.

B. Monitor Module

The monitor module polls the flow table, counts the average number of flow rules in flow tables in the latest monitor window, and decides whether to activate the detection module. The monitor window always contains the latest flow tables in 10 seconds by sliding back one sampling interval (1 second). The 10 seconds is equal to the idle timeout. We use a detection threshold to judge whether there is a risk of flow table overflow. When the mean in the latest monitor window exceeds the detection threshold, the detection module will be activated. The detection threshold cannot be too high or too low. A high threshold will make the detection module and the mitigation module start too late, which may cause an avoidable overflow. A low threshold will make the detection module start when there is sufficient space in the flow table, which may cause unnecessary overhead. The setting of the detection threshold is discussed in Section V. In a word, the monitor module can monitor the number of flow rules and start the detection module only when there is a risk of flow table overflow, which can avoid the extra cost caused by processing massive real-time flow table data.

C. Detection Module

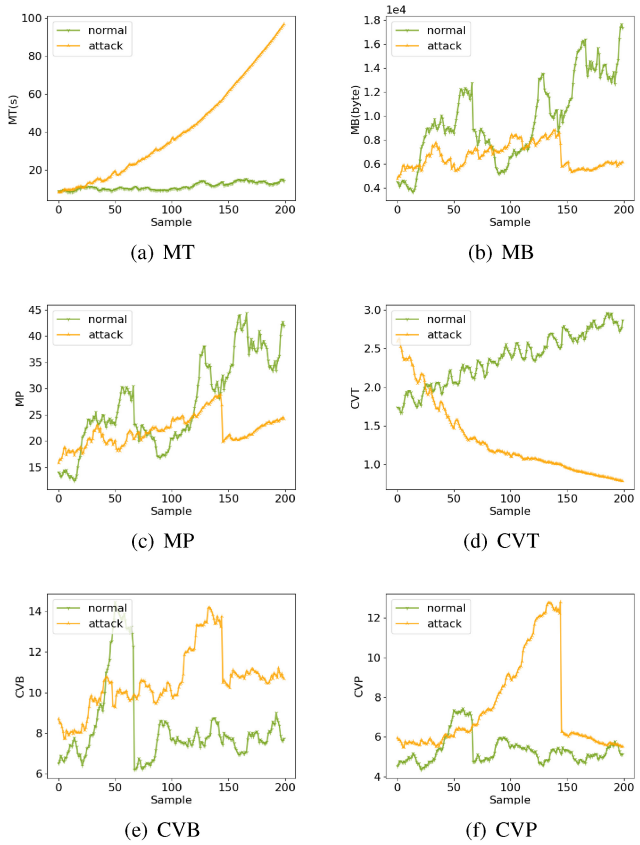
In the detection module, we need to extract the features of the flow table and make a classification to judge whether LFTO attacks occur.

1) Feature Extraction: By analyzing some statistical features of flow tables, we can find differences between flow tables under normal networks and LFTO attacks. We collect the time, bytes, and packets of flow rules in flow tables. Time is the duration of time that a flow occupies the flow table space. Bytes is the cumulative matched bytes number of a flow. Packets is the cumulative matched packet number of a flow. According to the principle of LFTO attacks, we know that most malicious packets are small without any efficient data, but they may constantly occupy the flow table since the attack will be periodically launched. Therefore, the Mean of Time, Bytes, and Packets (MT, MB, MP), and the Coefficient of Variation (CV) of Time, Bytes, and Packets (CVT, CVB, CVP) of all flow rules in a flow table will show different distribution as shown in Fig. 4. The mean can reflect the central tendency of the series. The CV can reflect the degree of variation of the series compared to the mean value.

Besides, we focus on the degree of disorder of source-destination IPs and ports and propose the following three features.

Disorder Degree of IP (DDoIP): Due to the limited performance and number of ports available to a single attacker, attacks can be launched by using spoofed IPs. It may lead to a more disordered distribution of source-destination tuples in flow tables. We use the Shannon entropy of source IPs to quantify the degree of disorder of source IPs, which is calculated as shown in Eq. (5):

$$DDoIP = H(ip_{src}) = -\sum_{ip_{src} \in iplist} p(ip_{src}) \log_2 p(ip_{src}) \quad (5)$$



(a) MT (b) MB (c) MP (d) CVT (e) CVB (f) CVP

Fig. 4. The feature distribution of flow tables.

where ip_{src} means source IPs, $H(ip_{src})$ is Shannon entropy of source IPs, *iplist* means a list of source IPs.

Disorder Degree of Port (DDoP): Similarly, the distribution of source-destination ports of flows during LFTO attacks will be more disordered than normal. We consider the joint entropy and conditional entropy of the source-destination ports together. The calculations are shown in Eqs. (6) and (7):

$$H(X, Y) = -\sum_{x\in X}\sum_{y\in Y} p(x, y)\log_2 p(x, y) \tag{6}$$

$$H(Y|X) = -\sum_{x\in X}\sum_{y\in Y} p(x, y)\log_2 p(y|x) \tag{7}$$

where $H(X, Y)$ is the joint entropy required to describe a pair of random variables X and Y on average, $H(Y|X)$ is the conditional entropy and represents the uncertainty of Y when the X is known. DDoP combines the above two entropies of source-destination ports and is calculated as Eq. (8):

$$DDoP = a \times H(port_{src}, port_{dst}) + (1 - a) \times H(port_{dst}|port_{src}) \tag{8}$$

where $port_{src}$ and $port_{dst}$ means source and destination ports. a is the balance factor, and the default is 0.5.

Coefficient of Disorder Degree (CDD): We consider the disorder degree of IPs and ports together and quantify it by CDD calculated as Eq. (9):

$$CDD = b \times H(ip_{src}) + (1 - b) \times H(port_{dst}|port_{src}) \tag{9}$$

where b is the balance factor, and the default is 0.5.

Fig. 5 shows the difference between DDoIP, DDoP, and CDD in the normal and attacked stages. We can see that the three features are all higher at the early stage of attacks than those at the normal stage. However, the malicious flow rules will gradually occupy the most space in the flow table, and the disorder degree of IPs and ports will decrease. Therefore, the three features gradually decrease at the later stage of attacks.

To prove the effectiveness of the nine features, we analyze the importance score of the nine features based on the XGBoost [45] in Fig. 6. We can notice that DDoIP obtains the highest importance score, and the importance score of all features is evenly distributed.

2) Classification: We choose XGBoost as the classifier, which is an efficient, flexible, and portable optimized distributed gradient enhancement library. The reason for selecting XGBoost is discussed in Section V.

D. Mitigation Module

In the mitigation module, we need to find the attack flow rules, delete them to prevent the flow table from overflowing, and finally block the attack source to mitigate the LFTO attack fully. Therefore, we extract six features of each flow rule to determine whether they are attack flow rules.

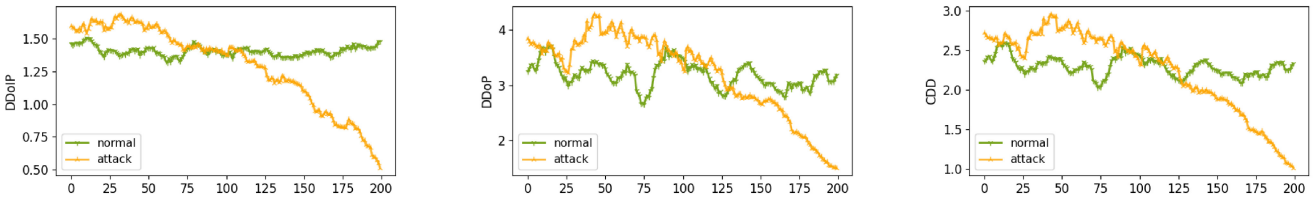
1) Feature Selection: The principle of identifying flow rules is to use various features of flow rules to distinguish the behavioral differences between different network flows. According to the analysis in Section II, LFTO attacks are stealthy, periodic, and usually launched by small packets. Therefore, normal flows and LFTO attack flows can be distinguished by the following six features.

Duration Time (DT): DT represents the duration of time that a flow occupies the flow table space. Attack flows would have a longer DT to occupy the flow table as long as possible. However, some active flows may also have a long DT. Therefore, the long DT cannot distinguish attack flows from some active flows. We need to combine other features below to make the classification.

Number of Bytes (NB): NB represents the cumulative matched bytes number of a flow rule.

Number of Packets (NP): NP represents the cumulative matched packet number of a flow rule. NP and NB can both indicate how much data is transferred. Low NP and NB indicate the amount of transmission data is small. At the same time, if the DT is long, it is more consistent with the features of attack flows.

Mean Packet Size (MPS). In normal network flows, there are large packets generated by fragmentation during data transmission and small packets generated by simple service response. However, an LFTO attacker only needs to periodically trigger a flow rule match to occupy the flow table space without delivering valid data, so the attack is usually launched by small packets. Therefore, the mean packet size of normal flow rules is more widely distributed, while the mean packet size of attack flow rules is concentrated in a smaller range to increase the concealment. MPS is calculated as $MPS = \frac{NB}{NP}$ ($MPS = 0$ if $NP = 0$).



(a) DDoIP (b) DDoP (c) CDD

Fig. 5. The feature distribution of flow tables.

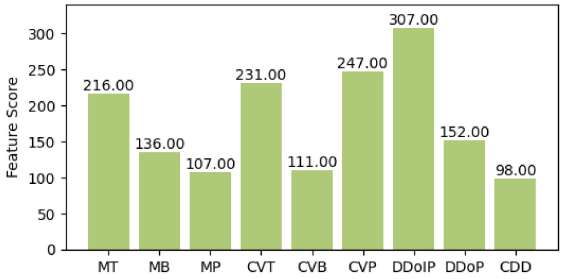


Fig. 6. Detection feature score of XGBoost.

Mean Packet Interarrival Time (MPIT): Due to the concealment of LFTO attacks, the longer attack cycle makes the MPIT of attack flow rules longer than that of normal flow rules. The calculation is $MPIT = \frac{DT}{NP}$ ($MPIT = DT$ if $NP = 0$).

Mean Transmission Speed (MTS): Due to the cyclicality of LFTO attacks and the small size of the packets delivered, the MTS in each attack flow may be extremely low as the DT increases. The calculation is $MTS = \frac{NB}{DT}$.

The six features are visualized in Fig. 7. We can not only focus on the effectiveness of a single feature but need to combine the six features to classify the flow rules. We can notice that the features of normal flow rules are more widely distributed. We also analyze the importance score of the six features based on XGBoost in Fig. 8. We can notice that DT, NP, MPS, and MPIT show vital importance. Although the importance score of NB and MTS is relatively low, Fig. 7 shows that they can produce an effect when combined with the other features.

2) The Mitigation Method: The mitigation module is activated after the LFTO attack is detected, which consists of three steps:

- Classify the flow rules and add the flow rules that need to be removed from the eviction list.
- Remove the rules in the eviction list and free up the flow table.
- Block the attack source.

At this point, FTMaster extracts six features of each flow rule and classifies the normal and attack flow rules using the XGBoost trained on the flow rules collected in the same network environment. The flow rules considered attack flow rules will be added to the eviction list of flow rules.

In addition, the mitigation module needs to protect elephant flows [46]. Elephant flows are essential to the whole network since they can transmit about 80% of the flows. Their MPIT is relatively large, their DT is long, and their NB is high. To prevent the mitigation module from accidentally deleting elephant flows, we filter them. If the NB and MPS of a flow rule are greater than those of 90% of all flow rules, it will be treated as an elephant flow, and its label will be set to 0 directly to protect elephant flows.

After confirming the eviction list, we use the switch command line to remove the rules, thus avoiding the overhead caused by control messages. For each rule in the eviction list, we execute `ovs-ofctl del-flows <src-dst tuple>` command to remove the rule. However, if we only constantly remove the attack flows, the LFTO attack can also constantly affect the flow tables and consume the resource, which is not a suitable mitigation strategy. Therefore, we need to further locate the source of attack flows and block it to fully mitigate the LFTO attack. While FTMaster constantly removes attack flows, it creates a blacklist of suspicious source IPs, and counts the number of times that each `ip_src` is deleted at the same time. If the counts of an `ip_src` exceed the mitigation threshold (succinctly set as 50% of the size of the flow table), we execute `add-flows <src-dst tuple> action = drop` to block the attack source.

V. Evaluation and Discussion

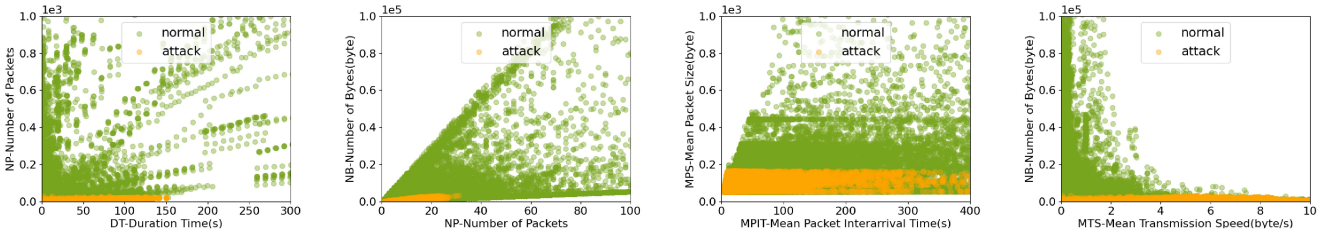
A. Experiment Setup

We deployed the SDN controller Ryu version 4.3.0 and the network simulator Mininet version 2.3.0d6 on Ubuntu 16.04.06. The software switches in Mininet is OpenvSwitch 2.5.9. The controller runs the Layer 4 learning switch application, where the matching fields include `protocol, port_src, port_dst, ip_src, and ip_dst`.

We used the actual network topology from the topology zoo dataset [47], called Napnet. As shown in Fig. 9, each city corresponds to a switch, and each switch connects to a host. There are six switches and seven links in total. We try to preserve the topological characteristics to adapt to reality. We set the bandwidth based on the dataset, and calculate the link distance and delay based on the city's latitude and longitude [48]. We set different flow table sizes to verify the universality of the method. The idle timeout is set as 10 seconds.

B. Dataset

1) Traffic Configuration (Background Traffic): We use the tcpreplay [49] to replay the dataset IMC-10 data center network trace [50] to simulate SDN network flows. We choose pk1 of univ1 as normal background flows. Since it is the network traffic collected around ten years ago, we think the



(a) Feature DT and NP (b) Feature NP and NB (c) Feature MPIT and MPS (d) Feature MTS and NB

Fig. 7. The distribution and relationship of the features of flow rules.

TABLE I
BACKGROUND TRAFFIC SETTINGS

No.	Table Size	RS (pps) h1 to h3	RS (pps) h2 to h1	RS (pps) h5 to h6
1	2K	400	200	400
2	3K	600	400	600
3	4K	800	800	800

TABLE II
PARAMETERS OF THE LFTO ATTACK

No.	AC(s)	AS(pkt/AC)	MAI(pkt))	Packet Size(Kb)
1	[5 , 8]	50	2500	[0,112]
2	[6 , 9]	75	2500	[0,112]
3	[6 , 9.5]	100	2500	[0,112]

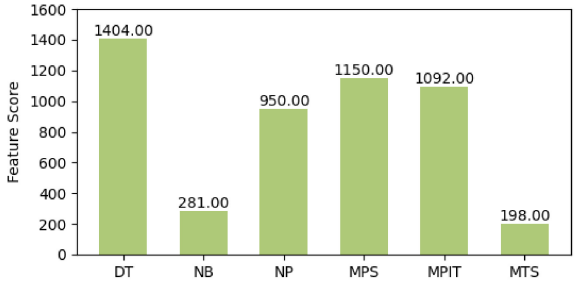


Fig. 8. Mitigation feature score of XGBoost.

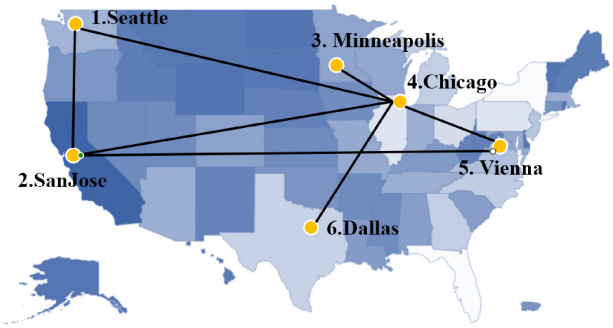


Fig. 9. Experiment Topology.

2K, 3K, and 4K flow table sizes can better reflect the hardware capacity at that time [51]. Table I shows the settings of background traffic, where RS means the replay speed (*packets per second* (pps)).

Attack Implementation: We implemented the LFTO attacks with Python scapy [52] and simulated three sets of LFTO attacks. The parameters are shown in Table II. The AS of LFTO attacks should not be too high. The higher the AS is, the weaker the concealment of LFTO attacks is. Therefore, we set the AC longer if the AS is higher to ensure the concealment of LFTO attacks. The packet size is set according to the packet size range in the background traffic dataset.

2) Datasets Generation: By replaying background traffic and simulating LFTO attacks, each switch is guaranteed to have at least one normal traffic and one attack traffic pass through.

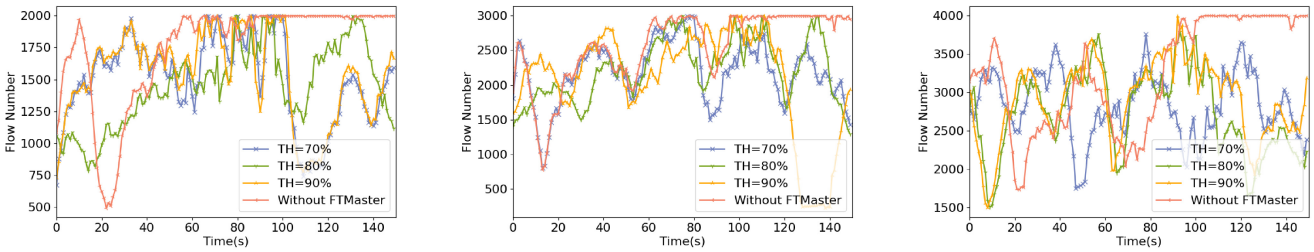
Detection Dataset: We simulated three sets of attacks with different parameters under flow table sizes of 2K, 3K, and 4K, respectively, and collected the flow tables of each switch every 1 second. Each set lasts 400 seconds, and the attacks are launched at 200th seconds. We collected 21600 flow tables, including 10,781 normal flow tables and 10,819 attack flow tables as the detection dataset (training set : test set $= 3 : 1$). We set the labels of flow tables according to whether attack flow rules exist in them.

Mitigation Dataset: Similarly, we simulated three sets of attacks with different parameters under flow table sizes of 2K, 3K, and 4K, respectively, and collected the flow tables of each switch every 10 seconds. The sampling interval of 10 seconds is equal to the idle timeout to avoid duplicate records from inactive flow rules. Each set lasts 200 seconds, and the attacks are launched at 50th seconds. We collected a total of 1583634 flow rules including 862131 normal flow rules and 721503 attack flow rules as the mitigation dataset (training set : test set $= 3 : 1$) and labeled them according to different source IPs for the training of the mitigation classification model.

C. Offline Evaluation of FTMaster

We experiment and evaluate the detection and mitigation effects of FTMaster offline. The following metrics are used for evaluation.

- *Accuracy (Acc):* The proportion of samples that are correctly classified, calculated as $Acc = \frac{TP+TN}{TP+TN+FP+FN}$.
- *False positive rate (FPR):* The proportion of normal samples incorrectly classified among all normal samples, calculated as $FPR = \frac{FP}{FP+TN}$.
- *False negative rate (FNR):* The proportion of attack samples incorrectly judged among all attack samples, calculated as $FNR = \frac{FN}{FN+TP}$.



(a) Table Size 2k (b) Table Size 3k (c) Table Size 4k

Fig. 10. Flow numbers in flow tables with different detection thresholds under different table sizes.

TABLE III
COMPARISON OF CLASSIFICATION PERFORMANCE OF DETECTION MODULE

Classifier	Acc	FPR	FNR	F1-Score
NGBoost [53]	94.405%	3.304%	7.877%	94.285%
XGBoost	**99.333%**	0.037%	1.294%	**99.330%**
CatBoost [54]	98.870%	**0%**	2.256%	98.859%
AdaBoost	98.462%	1.225%	1.849%	98.460%
GBDT	99.092%	0.186%	1.627%	99.087%
RF	99.315%	0.111%	**1.257%**	99.312%

TABLE IV
COMPARISON OF CLASSIFICATION PERFORMANCE OF MITIGATION MODULE

Classifier	Acc	FPR	FNR	F1-Score
NGBoost	89.368%	7.657%	14.186%	88.031%
XGBoost	**92.479%**	**5.020%**	**10.509%**	**91.556%**
CatBoost	89.883%	6.940%	13.913%	88.576%
AdaBoost	89.469%	8.885%	12.497%	83.333%
GBDT	90.537%	7.410%	11.905%	89.455%
RF	90.081%	8.037%	12.168%	88.973%

- *F1-Score:* The harmonized average of Precision and Recall, calculated as $F1-Score = \frac{2\times Precision\times Recall}{Precision+Recall}$. Precision means the proportion of true attack samples among samples classified as attack samples, calculated as $Precision = \frac{TP}{TP+FP}$. Recall means the proportion of attack samples correctly classified among all attack samples, calculated as $Recall = \frac{TP}{TP+FN}$.

1) Offline Evaluation of Detection Module: The detection module aims to achieve the detection of attacked flow tables with low FPR for normal flow tables and early detection. We test the detection performance of various classifiers based on ensemble learning. Table III shows the results. We can see that most classifiers can achieve an accuracy of 98% or higher and a low FPR approaching 0. However, XGBoost has the highest accuracy and F1-Score, and the FPR and FNR are relatively low. According to the above analysis, we choose XGBoost as the detection classifier.

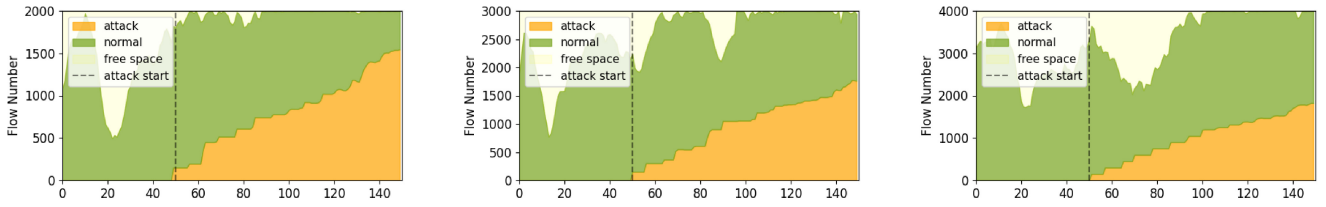
2) Offline Evaluation of Mitigation Module: The mitigation module aims to identify the attack flow rules and delete them. We can easily accept some attack flow rules occupying flow tables space, but we cannot accept that even normal flow rules are misclassified. Therefore, a low FPR is very important. Similarly, we compare the classification performance on normal and attack flow rules of each classifier. As shown in Table IV, XGBoost has the highest accuracy and F1-Score, and the lowest FPR and FNR. Although the FNRs of classifiers are relatively high, FTMaster will remove all the flow rules classified as attack flow rules. Therefore, the remaining attack flow rules are less threatening to the flow table, even if they are hidden in the flow table. Finally, we choose XGBoost, which has the best overall performance, as the classifier for the mitigation module.

D. Online Evaluation of FTMaster

In this section, we deploy FTMaster on SDN switches and evaluate its effectiveness and performance under different flow table sizes. We focus on the deployment effect of Node 4 in our topology. The parameters of background traffic are set as Table I. The parameters of attack traffic are the mix of the three groups in Table II.

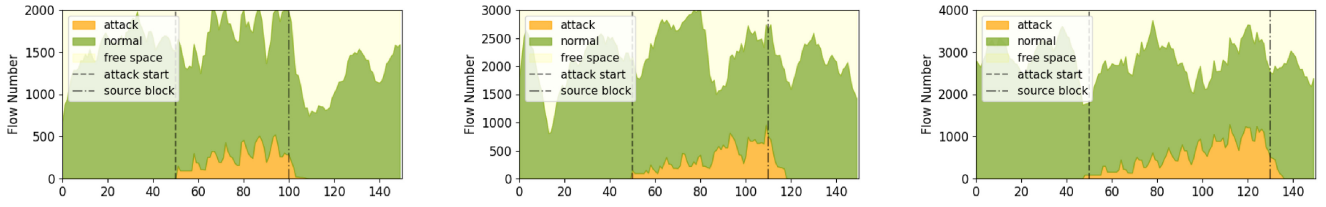
1) Detection and Mitigation Effect: First, we test the detection and mitigation effect with different detection thresholds and compare the number of flow rules with and without FTMaster under different flow table sizes. We set the detection threshold to 70%, 80%, and 90%.

We show the results in Fig. 10. According to our setting, the number of normal flow rules is about 500 to 2000 under the 2K flow table, 1000 to 2500 under the 3K flow table, and 1500 to 3500 under the 4K flow table. At the 50th second, the LFTO attack is started and the flow table constantly overflows after the 100th second when the FTMaster is not deployed. Especially, Fig. 10(a) shows that when the FTMaster is not deployed, the flow table of OpenvSwitch is easy to overflow even in a normal situation since the flow table space is too limited. Although the flow table still overflows several times when FTMaster is deployed with different thresholds, the attack source is blocked at about the 100th to 120th seconds and then the flow processing will return to a normal state. Fig. 10(b) shows that when the threshold is 70%, the flow table only overflows shortly and then remains in a normal state constantly. In Fig. 10(c), the 4K table size provides relatively sufficient space for flow rules, and the flow table overflow needs more time, while FTMaster can delete the attack flow rules firmly and block the attack source before the flow table overflows. According to the above analysis, we finally set the detection



(a) Table Size 2k (b) Table Size 3k (c) Table Size 4k

Fig. 11. Number of normal and attack flow rules in the flow table without FTMaster.



(a) Table Size 2k (b) Table Size 3k (c) Table Size 4k

Fig. 12. Number of normal and attack flow rules in the flow table with FTMaster.

threshold as 70% because of its relatively sound effects under 3K and 4K table sizes.

Second, we evaluate the effectiveness of the flow rule eviction under different flow table sizes, that is, whether the evicted flow rules are attack flow rules. We collect the contents of the switch flow table for LFTO attacks every second, extract normal and attack flow rules in it, and calculate their proportions, respectively.

Fig. 11 shows the number of normal and attack flow rules in the flow table space without FTMaster. In the first 50 seconds, the normal flow rules can occupy more than 80% of the space in flow tables. That is, the burden of data processing in flow tables is heavy even in a normal network. We can see that after the LFTO attack starts, the flow table constantly overflows after the 100th second, and there are approximately 1500 attack flow rules at the 150th second, which have occupied a large part of space for normal flow rules.

Fig. 12 shows the number of normal and attack flow rules in the flow table space with FTMaster. We can see that attack flow rules occupies below 25% of the flow table space continuously at the early stage, ensuring the forwarding and processing of most normal flows. At the later stage, the attack source can be blocked, and the flow table returns to a normal state after FTMaster has deleted the remaining attack flow rules. This indicates that FTMaster can accurately identify most attack flow rules, evict them while ensuring the installation and forwarding of normal flows, and then finally block the attack source.

2) Real-Time Performance: We evaluate the time consumption of the detection module and mitigation module. Since FTMaster needs to process the real-time flow table data, the time consumption is partly related to the flow table size. The larger the flow table size, the more the flow rules and the more time it takes for FTMaster to process data. Table V shows that it only takes 0.03 to 0.06 seconds for the detection module, and the time consumption is mainly concentrated in the mitigation module. The maximum time of mitigation under the 4K flow table reaches approximately 4 seconds because of the deletion of a large number of flow rules. However, it is only an isolated case. The mitigation time ranges from 0.2 to 0.6 seconds in most cases. Therefore, FTMaster has good real-time performance, and each detection and mitigation can be completed in 0.3 to 0.7 seconds on average.

TABLE V
THE TIME CONSUMPTION OF FTMASTER

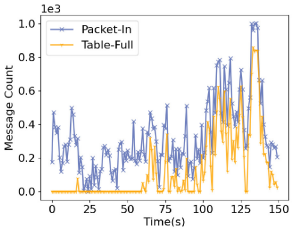
Table size	Module	mean(s)	maximum(s)	minimum(s)	medium(s)
2K	Detection	0.0446	0.0643	0.0349	0.0434
	Mitigaiton	0.2879	1.2117	0.0475	0.2018
	overall	0.3326	1.2550	0.0835	0.2479
3K	Detection	0.0489	0.0756	0.0375	0.0485
	Mitigaiton	0.3625	1.4879	0.0667	0.2439
	overall	0.4114	1.5291	0.1130	0.2948
4K	Detection	0.0537	0.0643	0.0392	0.0513
	Mitigaiton	0.6391	4.2267	0.0817	0.4029
	overall	0.6924	4.2881	0.1244	0.4630

3) Control Link Protection: The aim of protecting the flow table is not just about its availability, but also about reducing controller resource consumption. Sufficient flow table space can ensure that more traffic is processed on the data plane without forwarding it to the controller for further processing. Therefore, we focus on the `Packet-In` messages and `Table-Full` messages received by the controller when FTMaster is deployed and not deployed under different flow table sizes. Figs. 13–15 show the results. When FTMaster is not deployed, the `Packet-In` messages will increase, which indicates that certain flows cannot install flow rules in the data plane and perform matching directly. In addition, a `Table-Full` error occurs after the attack starts, which indicates that the flow table is overflowing at this point.

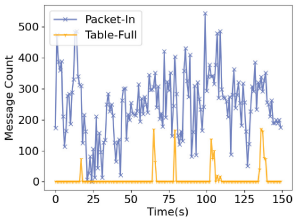
TABLE VI
COMPARED RESULTS UNDER LFTO ATTACKS

Method	Depoly Plane	Average Relative[1] Attack Flows Proportion	Average Flow Table Usage	Earliest Overflow Time (s)	Total Overflow Time (s)	Table-Full Count	CPU	Memory (MB)
FTMaster	data	17.58%	69.70%	113	2	93	2.38%	230
RIT	data	31.82%	83.98%	114	49	13258	0.50%	19
EVICT	data	29.30%	88.58%	105	46	NA[2]	0.44%	0
SIFT	control	31.67%	25.58%	0	0	0	1.25%	21

[1] "Relative" means the proportion among all flow rules in the flow table.
[2] "EVICT" will not send Table-Full to controller.

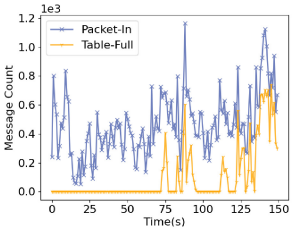


(a) Without FTMaster

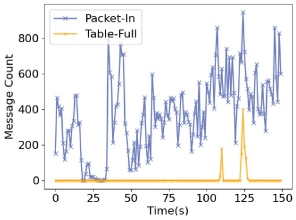


(b) With FTMaster

Fig. 13. Comparison of message counts with and without FTMaster (Table Size = 2000).

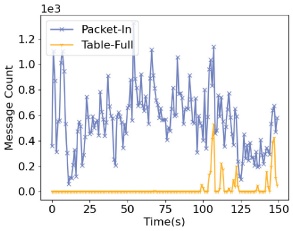


(a) Without FTMaster

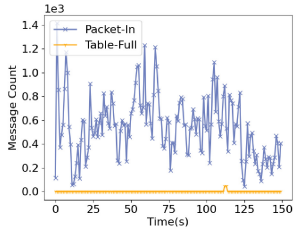


(b) With FTMaster

Fig. 14. Comparison of message counts with and without FTMaster (Table Size = 3000).



(a) Without FTMaster

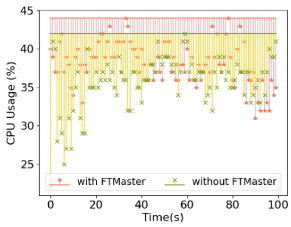


(b) With FTMaster

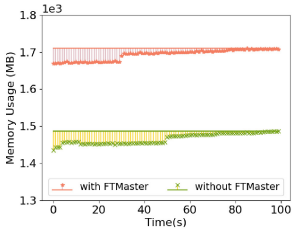
Fig. 15. Comparison of message counts with and without FTMaster (Table Size = 4000).

However, when FTMaster is deployed, the flow table hardly overflows, then the controller hardly receives `Table-Full` error. Also, the frequency of `Packet-In` request will be decreased compared to that without FTMaster.

4) System Overhead: We evaluate the overhead of FTMaster, including CPU usage and memory usage. Fig. 14(a) indicates that the average CPU usage increases only about 2.38% when deploying FTMaster, which indicates that FTMaster does not occupy much system space. Besides, the



(a) CPU Usage



(b) Memory Usage

Fig. 16. Comparison of system overhead with and without FTMaster (Table Size = 4000).

memory usage is about 230MB and needs to be further reduced.

5) Compared Analysis: We compare the mitigation effect of FTMaster with that of some other mitigation methods, including *random idle timeout* (RIT) where the idle timeout is a random number range from 5 to 15 seconds, *the native Overflow policy in OpenFlow switches* (EVICT) [55] that delete a flow instead of the default policy REFUSE that refuse to add a flow, and *Selective Defense for TCAM* (SIFT) [8]. The compared results are shown in Table VI. The simulation lasts 150 seconds, and the LFTO attack starts at the 50th second. The flow table size is 4K. From Table VI, we can see that the average proportion of relative attack flows is only 17.58%, which is the lowest among the four methods, indicating that FTMaster can evict the attack flows more effectively. Besides, the number of `Table-Full` is very low, indicating that FTMaster ensures that the flow table hardly overflows. Although the flow table does not overflow when SIFT is deployed, we can see that the average flow table usage is only 25.58%. We think the reason is that SIFT needs the controller to store massive flow data, which causes the degradation of network performance. Many flows cannot be processed in time and forwarded to the controller. We also notice that the CPU and memory usage of RIT, EVICT, and SIFT are all lower than FTMaster. Therefore, we need to reduce the system overhead of FTMaster in future work.

VI. CONCLUSION AND DISCUSSION

This paper focuses on the LFTO attack targeted at the limited capacity of flow tables based on the TCAM in SDN, analyzes its principle and impact, and designs the FTMaster, a

machine learning-based system against LFTO attacks in SDN. FTMaster tracks the real-time average number of flow rules, detects the LFTO attacks by collecting the statistical features of flow tables and distinguishing the states of flow tables, and finally mitigates the LFTO attacks by evicting the attack flows and blocking the attack source. Experiments are conducted based on the Ryu SDN controller and Mininet, and the results indicate that FTMaster can effectively detect and mitigate LFTO attacks under different flow table sizes.

However, FTMaster still has some limitations. The FNR of attack flow rules needs to be further lower, by which more space can be freed up for normal flow rules. Besides, the system overhead needs to be further reduced. In future work, we would like to discuss more scenarios to enhance the usability of the system in other malicious attacks and vulnerability scenarios. Moreover, we would like to evaluate the effectiveness of FTMaster and find its vulnerabilities and limitations in a real-world network environment.

REFERENCES

[1] Y. Maleh, Y. Qasmaoui, K. El Gholami, Y. Sadqi, and S. Mounir, "A comprehensive survey on SDN security: Threats, mitigations, and future directions," *J. Rel. Intell. Environ.*, vol. 2022, pp. 1–39, Feb. 2022.

[2] R. Deb and S. Roy, "A comprehensive survey of vulnerability and information security in SDN," *Comput. Netw.*, vol. 206, Art. no. 108802, Apr. 2022.

[3] A. Liatifis, P. Sarigiannidis, V. Argyriou, and T. Lagkas, "Advancing SDN from openflow to P4: A survey," *ACM Comput. Surveys*, vol. 55, no. 9, pp. 1–37, 2023.

[4] G. Lu et al., "ServerSwitch: A programmable and high performance platform for data center networks," in *Proc. 8th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2011, pp. 1–14.

[5] 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. Security Privacy Commun. Syst.*, 2017, pp. 356–376.

[6] 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 Manage.*, vol. 14, no. 4, pp. 1086–1097, Dec. 2017.

[7] J. Pei, P. Hong, K. Xue, and D. Li, "Resource aware routing for service function chains in SDN and NFV-enabled network," *IEEE Trans. Services Comput.*, vol. 14, no. 4, pp. 985–997, Jul./Aug. 2021.

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

[9] K. Xie et al., "On-line anomaly detection with high accuracy," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1222–1235, Jun. 2018.

[10] W. Liang, K.-C. Li, J. Long, X. Kui, and A. Y. Zomaya, "An industrial network intrusion detection algorithm based on multifeature data clustering optimization model," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 2063–2071, Mar. 2020.

[11] 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.

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

[13] 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.

[14] X. Huang, K. Xue, Y. Xing, D. Hu, R. Li, and Q. Sun, "An efficient scheme to defend data-to-control-plane saturation attacks in software-defined networking," *J. Comput. Sci. Technol.*, vol. 37, no. 4, pp. 839–851, 2022.

[15] 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, Nov./Dec. 2022.

[16] G. Shang, P. Zhe, X. Bin, H. Aiqun, and R. Kui, "FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

[17] 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.

[18] 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.*, early access, Apr. 13, 2023, doi: 10.1109/TSC.2023.3266757.

[19] D. Tang, Z. Zheng, X. Wang, S. Xiao, and Q. Yang, "PeakSAX: Real-time monitoring and mitigation system for LDoS attack in SDN," *IEEE Trans. Netw. Service Manage.*, early access, Nov. 16, 2022, doi: 10.1109/TNSM.2022.3222846.

[20] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Yau, and J. Wu, "Realtime DDOS defense using COTS SDN switches via adaptive correlation analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, pp. 1838–1853, 2018.

[21] A. Alimohammadifar et al., "Stealthy probing-based verification (SPV): An active approach to defending software defined networks against topology poisoning attacks," in *Proc. Eur. Symp. Res. Comput. Security*, 2018, pp. 463–484.

[22] E. Marin, N. Bucciol, and M. Conti, "An in-depth look into SDN topology discovery mechanisms: Novel attacks and practical countermeasures," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2019, pp. 1101–1114.

[23] P. Shrivastava and K. Kataoka, "Topology poisoning attacks and prevention in hybrid software-defined networks," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 1, pp. 510–523, Mar. 2022.

[24] R. Skowyra et al., "Effective topology tampering attacks and defenses in software-defined networks," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, 2018, pp. 374–385.

[25] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: New attacks and countermeasures," in *Proc. Network Distrib. Syst. Security Symp. (NDSS)*, vol. 15, 2015, pp. 8–11.

[26] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: Detecting security attacks in software-defined networks," in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, vol. 15, 2015, pp. 8–11.

[27] D. Smyth, S. Scott-Hayward, V. Cionca, S. McSweeney, and D. O'Shea, "SECAP switch—Defeating topology poisoning attacks using P4 data planes," *J. Netw. Syst. Manage.*, vol. 31, no. 1, p. 28, 2023.

[28] F. Shoaib, Y.-W. Chow, and E. Vlahu-Gjorgievska, "Preventing timing side-channel attacks in software-defined networks," in *Proc. IEEE Asia–Pacific Conf. Comput. Sci. Data Eng. (CSDE)*, 2021, pp. 1–6.

[29] M. Conti, F. De Gaspari, and L. V. Mancini, "A novel stealthy attack to gather SDN configuration-information," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 2, pp. 328–340, Apr.–Jun. 2020.

[30] S. Liu, M. K. Reiter, and V. Sekar, "Flow reconnaissance via timing attacks on SDN switches," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2017, pp. 196–206.

[31] S. Achleitner, T. La Porta, T. Jaeger, and P. McDaniel, "Adversarial network forensics in software defined networking," in *Proc. Symp. SDN Res.*, 2017, pp. 8–20.

[32] M. Yu, T. Xie, T. He, P. McDaniel, and Q. K. Burke, "Flow table security in SDN: Adversarial reconnaissance and intelligent attacks," *IEEE/ACM Trans. Netw.*, vol. 29, no. 6, pp. 2793–2806, Dec. 2021.

[33] 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. Attacks, Intrusions, Defenses*, 2018, pp. 161–183.

[34] B. Isyaku, M. B. Kamat, K. B. A. Bakar, M. S. M. Zahid, and F. A. Ghaleb, "IHTA: Dynamic idle-hard timeout allocation algorithm based OpenFlow switch," in *Proc. IEEE 10th Symp. Comput. Appl. Ind. Electron. (ISCAIE)*, 2020, pp. 170–175.

[35] X. Li and Y. Huang, "A flow table with two-stage timeout mechanism for SDN switches," in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Commun. IEEE 17th Int. Conf. Smart City IEEE 5th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, 2019, pp. 1804–1809.

[36] S. K. Noh, M. Kang, and M. Park, "Protection against flow table overflow attack in software defined networks," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, 2021, pp. 486–490.

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

[38] T. Pan, X. Guo, C. Zhang, W. Meng, and B. Liu, "ALFE: A replacement policy to cache elephant flows in the presence of mice flooding," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2012, pp. 2961–2965.

[39] 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.

[40] S. Qiao, C. Hu, X. Guan, and J. Zou, "Taming the flow table overflow in OpenFlow switch," in *Proc. ACM SIGCOMM Conf.*, 2016, pp. 591–592.
[41] M. Yue, H. Wang, L. Liu, and Z. Wu, "Detecting DoS attacks based on multi-features in SDN," *IEEE Access*, vol. 8, pp. 104688–104700, 2020.
[42] D. Tang, D. Zhang, Z. Qin, Q. Yang, and S. Xiao, "SFTO-Guard: Real-time detection and mitigation system for slow-rate flow table overflow attacks," *J. Netw. Comput. Appl.*, vol. 213, Apr. 2023, Art. no. 103597.
[43] M. Soylu, L. Guillen, S. Izumi, T. Abe, and T. Suganuma, "NFV-Guard: Mitigating flow table-overflow attacks in SDN using NFV," in *Proc. IEEE 7th Int. Conf. Netw. Softwarization (NetSoft)*, 2021, pp. 263–267.
[44] S. Xie, C. Xing, G. Zhang, and J. Zhao, "A table overflow LDoS attack defending mechanism in software-defined networks," *Security Commun. Netw.*, vol. 2021, Jan. 2021, Art. no. 6667922.
[45] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2016, pp. 785–794.
[46] L. Guo and I. Matta, "The war between mice and elephants," in *Proc. 9th Int. Conf. Netw. Protocols*, 2001, pp. 180–188.
[47] "The Internet topology zoo." 2011. [Online]. Available: http://www.topology-zoo.org/dataset.html
[48] "Graphml to MiniNet." 2023. [Online]. Available: https://github.com/yyd19981117/Graphml-To-Mininet
[49] "Tcpreplay." 2023. [Online]. Available: https://tcpreplay.appneta.com/
[50] "Data set for IMC 2010 data center measurement." 2023. [Online]. Available: http://pages.cs.wisc.edu/tbenson/IMC10 Data.html
[51] 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. 2020.
[52] "python scapy," 2023. [Online]. Available: https://scapy.net/
[53] T. Duan et al., "NGBoost: Natural gradient boosting for probabilistic prediction," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 2690–2700.
[54] "Catboost." 2023. [Online]. Available: https://yandex.com/dev/catboost/
[55] "Openvswitch." 2023. [Online]. Available: http://www.openvswitch.org/support/dist-docs/ovs-vswitchd.conf.db.5.txt

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

Chenjun Gao received the B.A. degree from the Wuhan University of Technology, China, in June 2021. She is a currently pursuing the Postgraduate degree with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. Her current research interests are network attack detection and mitigation.

Wei Liang received the Ph.D. degree in computer science and technology from Hunan University, China, in 2013. He was a Postdoctoral Scholar with Lehigh University, Bethlehem, PA, USA, from 2014 to 2016. He is currently a Professor with the School of Computer Science and Engineering, Hunan University of Science and Technology. He has authored or coauthored more than 140 journal and conference papers, such as IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, and IEEE INTERNET OF THINGS JOURNAL. His research interests include blockchain security technology, network security protection, and security management in wireless sensor networks.

Jiliang Zhang (Senior Member, IEEE) 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 and the Secretary-General of CCF Fault-Tolerant Computing Professional Committee. He has authored more than 80 technical papers in leading journals and conferences. His current research interests include hardware security, integrated circuit design, and intelligent system. He was the recipient of CCF Integrated Circuit Early Career Award and the winner of Excellent Youth Fund of the NSFC. He was a CCF Distinguished Speaker. He has been the Program Committee Member for a number of well-known conferences, such as DAC, ASP-DAC, GLSVLSI, and FPT.

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. He has authored or coauthored over 850 journal articles, book chapters, and refereed conference papers. He holds over 70 patents announced or authorized by the Chinese National Intellectual Property Administration. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, heterogeneous computing systems, computer networking, and machine learning. He has received several best paper awards. He is among the world's top five 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. He has chaired many international conferences. He is currently an Associate Editor of the *ACM Computing Surveys* and the *CCF Transactions on High Performance Computing*. He is an AAIA Fellow and also a Member of Academia Europaea (Academician of the Academy of Europe).