Hindawi
Security and Communication Networks
Volume 2022, Article ID 1957497, 14 pages
https://doi.org/10.1155/2022/1957497

Research Article

Research on SDN Fingerprint Attack Defense Mechanism Based on Dynamic Disturbance and Information Entropy Detection

Xing Fang, Wenhui Zhang, Jiming Lin, and Yuming Liu

Guilin University of Electronic Technology, Guilin 541004, China

Correspondence should be addressed to Wenhui Zhang; zhangwh@guet.edu.cn

Received 13 January 2022; Revised 4 March 2022; Accepted 5 July 2022; Published 13 August 2022

Academic Editor: Shah Nazir

As an emerging type of network architecture, SDN is widespread used and security issues have also received more and more attention. Fingerprint attacks represent one of the most significant threats to network security. Attackers obtain key fingerprint information of the target network, which lays the foundation for subsequent more threatening attacks. Currently, research on domestic and international SDN fingerprint attacks focuses on how to attack, and less research is being done on how to defend against fingerprinting attacks. This paper proposes a mechanism for defending fingerprint attacks that combines dynamic disturbance and information entropy detection. This mechanism adopts the principle of fingerprint attack, combined with a moving average algorithm, Bloom Filter, and packet delay tool, to confuse opponents by disturbing a small number of packets, simultaneously, combined with the information entropy detection to make real-time processing feedback to the network. The experimental results show that this mechanism works effectively to defend SDNs against fingerprint attacks without affecting the normal network communication.

1. Introduction

Software-Defined Network (SDN) is an emerging network architecture consisting of the application layer, control layer, and infrastructure layer. The decoupling between the control plane and data plane is realized by OpenFlow technology, and a more flexible way to manage network traffic is introduced with high programmability [1]. With the rapid development of SDN technology and its wide application in various fields, more and more attention has been paid to its security issues. Researchers at home and abroad have analyzed and summarized the security threats faced by SDNs from different perspectives. The packet processing speed on the data plane is multiple orders of magnitude faster than that on the software-based control plane due to the characteristics of SDN's 'three layers —three interfaces' architecture, particularly when performing packet forwarding on hardware. As shown in Figure 1, when the host1 communicates with the host 2, the data header is parsed by Step 1 to query the internal flow table of the switch. If a matching rule exists, the corresponding action (Forward or Drop) is executed, otherwise, Step 2 and Step 3 are executed, after the computation and processing of the SDN controller, the corresponding rules are issued and the corresponding actions are performed on the packet. The round-trip delay RTT values of packets in these two cases are greatly different due to the different processing processes of packets with or without matching rules as shown in Figure 2. The attacker can send two identical detection packets, measure their RTT values, and calculate $\Delta RTT = RTT_1 - RTT_2$. If $\Delta RTT \approx 0$, it can be inferred that there are related matching rules in the switch flow table or that the target network is a traditional network type. If $|\Delta RTT| \gg 0$, it can be speculated that the target network is an SDN and the detection packet leads to the installation of flow table rules. Furthermore, the attacker can infer more fingerprint information such as flow matching rules of the target network by carefully constructing detection packets [2–4], thereby exposing the entire SDN to a sea of threats as well as laying the foundation for the attacker to carry out the next more threatening and precise attack, such as Distributed Denial of Service (DDoS) attacks through forged data packets, etc. Panjwani et al. concluded that up to 70% of network attacks were carried out after fingerprint

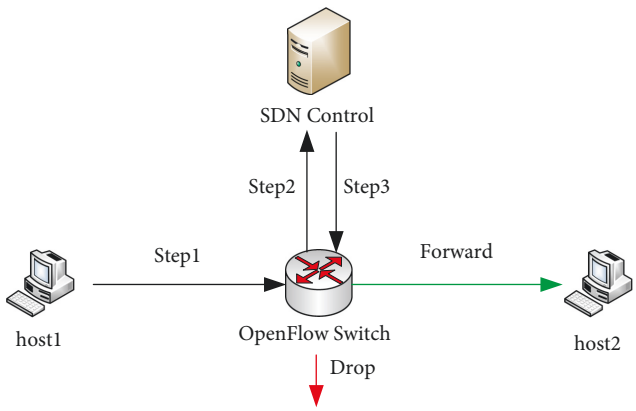


Figure 1: Openflow switch processing data packet process.

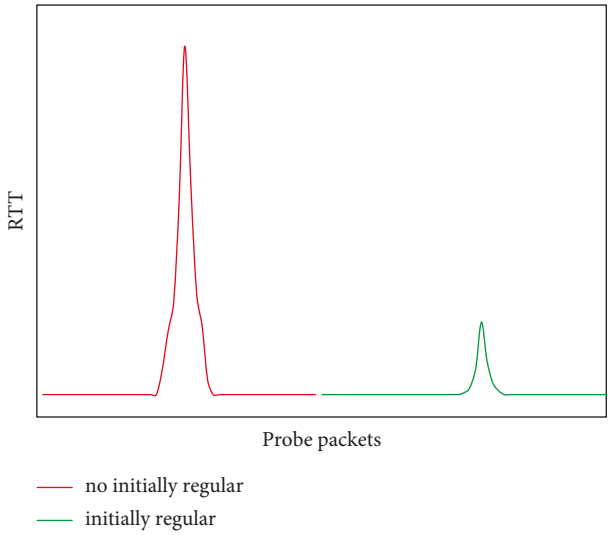


Figure 2: RTT comparison with or without rules.

attacks [5]. SDN fingerprint attacks have become an urgent problem to be solved.

At present, the research on domestic and international SDN fingerprint attacks is mainly about how to attack, and there is only a little literature on the defense research of SDN fingerprint attacks. The defense against SDN fingerprint attacks is mostly through certain active measures, mainly through packet delay forwarding [5–10], moving target defense (MDT) [11–14], and setting honeypots [15, 16] to confuse attackers. The method of packet delayed forwarding takes countermeasures to fingerprint attacks based on the attack principle, which can bemuse the attacker available. Without affecting the network performance, delayed forwarding of a small number of initial packets is an effective defense against fingerprint attacks. However, there is currently no detailed method for its specific implementation, and it is arduous to determine the specific value of the packet delay. MTD continuously changes the attack surface by dynamically changing the infrastructure. Attackers are forced to use tremendous resources to continuously analyze and detect the architecture of this change, which becomes more difficult with time. However, this type of method needs to consider users' normal needs in the network. High-speed address hopping may interrupt the communication in the connection while defending against fingerprint attacks, and the communication between legitimate hosts is seriously affected. Honeypots lure attackers through trap technology and study the characteristics of attackers at the same time. But traditional honeypots have shortcomings such as static configuration and fixed position. Once the attacker finds or bypasses them, the honeypot will immediately fail. One of the major problems faced by the existing honeypot solutions is the overhead of system resources, honeypots need to continue to operate, even if no attack occurs.

The focus of this study is how to use the principle of fingerprint attack to interference with the round-trip delay RTT of the detection packet to confuse the attacker, and how to quickly discover the ongoing fingerprint attack and take measures through real-time monitoring of the entire SDN. Based on this, this paper proposes a fingerprint attack defense mechanism that combines dynamic disturbance and information entropy detection for SDN fingerprint attacks, and designs as well as implements the system SDFADS (Software Defined fingerprint attack defense system). The main research contributions of this paper are as follows:

(1) The fingerprint attack threat faced by the SDN is discussed and its attack principle is analyzed, and a fingerprint attack defense mechanism combining dynamic disturbance and information entropy detection is proposed.

(2) The filtering strategy of the multi-feature data stream is proposed, and Bloom filter technology is used to analyze the data packet to judge whether the current data stream appears for the first time;

(3) This paper analyzes the common characteristics of time-based fingerprint attacks in SDNs and puts forward a dynamic disturbance strategy. It is proposed for the first time to dynamically disturb the round-trip delay RTT of a small number of initial data packets by combining the moving average algorithm and the packet delay tool, to achieve the effect of confusing the detection packet RTT, which has little impact on the network performance while confusing the attacker;

(4) A lightweight information entropy initial detection mechanism based on multi-dimensional features is proposed. The suspected abnormal traffic is inferred by statistically calculating the information entropy of data packets, and the suspicious ports are further processed and abnormal logs are recorded.

(5) The effectiveness of the proposed method is demonstrated by evaluating the RTT distribution after dynamic interference, the detection effect of information entropy, and the change of the number of flow tables in SDN switches.

2. Related Works

Liu et al. [2] estimated the distribution of flow table rules in the switch by constructing a Markov model of the SDN switch and further used the distribution to launch

fingerprint attacks. Patwardhan et al. [5] assumed that the attack will rely on a fixed address to send probe packets, and the structure of probe packets should show similarity from the same source IP address and source port number to numerous different destinations. The controller checks the header and data of the detection packet (deep packet inspection) to identify the attack detection packet, checks the suspicious host, and prevents the attack. Analyzing the time interval between consecutive packets from the same source IP address can also provide the characteristics of fingerprint attacks but might increase the load on the switch. Yu et al. [6] proposed a fine-grained attack scheme by inferring the flow table fingerprint information such as flow table capacity and flow table replacement strategy. Yadong et al. [7] pointed out that the attackers can proactively generate probe packets to trigger the interaction between the controller and the switch about the insertion and deletion of flow table rules, and then the attackers can estimate the internal state of the SDN by measuring the changes in network performance, including flow table capacity and flow table usage situation. And propose a route aggregation scheme based on a packaging optimization algorithm and a multi-level flow table architecture combining TCAM and SRAM. However, route aggregation cannot replace long-term architecture solutions because it cannot solve the fundamental problems of flow table scalability and fingerprint inference attacks. The capacity and update speed of TCAM are inherently rooted in the hardware design of the memory chip and cannot be improved immediately. Zhang et al. [8] proposed a two-stage detection trigger attack strategy, including the detection phase and the trigger phase, which make fingerprint attacks more effective and powerful.

At present, the defense against SDN fingerprint attacks is mostly through a few active measures, such as packet delay forwarding, moving target defense, and honeypot setting.

2.1. Packet Delay Forwarding. Unified delay for all packets is one method to defend against fingerprint attacks, but this approach seriously affects the performance of the entire network, which is job desirable.

Cui H et al. [9] proposed to focus on existing matching rules of data flow, using group table to forward active flow directly, inactive flow forward to specific port utilizing network delay device for delay processing. However, the processing of the new flow is not involved. When the detection packet involves the installation of a new flow rule, the attacker can observe the difference between the round-trip delay and the existing matching rule, and the delay network device is not described in detail and its availability is unknown. Wang and Chen [10] proposed to impose perturbation on the probability of the initial data packet. By defining a new action bucket selection logic for the group table, different data packets can be implemented with different delay operations, and the scrambling strategy can be converted into a data plane executable instruction. Nonetheless, the specific implementation method has not been mentioned and the situation when there are matching rules in the flow table has not been considered, and the parameter settings in the probability model are affected by human factors. In the real SDN, the location of the host will not change frequently, and the host connected to the switch port will not change frequently in the network [17]. Based on this, Hou et al. [18] analyzed the common characteristics of time-based fingerprint attacks in SDNs, and explored a lightweight method to counter fingerprint attacks, taking the source IP and source MAC changes of the host address as potential fingerprint detection behavior, when the host address changes, delay the installation of matching rules in the switch, thereby increasing the difficulty of fingerprint attacks. Notwithstanding, if the attacker only uses one host to perform a fingerprint attack, this method is not applicable. The paper does not explain how to implement the delayed installation of the rules. Yuwen et al. [19] proposed an unpredictable delay strategy that will send packets to the SDN with adjustable probability delay, which aims to confuse the time information received by the attacker and reduce the performance cost as much as possible. However, the method proposed in this paper to increase the delay of data packets by issuing matching rules and the controller itself is open to discussion. Achleitner et al. [20] proposed to delay the attacker's detection packets to invalidate the information they collect to hinder network reconnaissance, and at the same time limit the performance impact on benign network traffic. But a specific value of the delay added to the packet is easily found by the attackers.

The method of packet delay forwarding takes countermeasures to fingerprint attacks based on the attack principle, which can befuddle the attacker efficiently. Without affecting the network performance, the delayed forwarding of a small number of data packets is an effective method to defend against fingerprint attacks. But the specific implementation method of the above literature is not described in detail.

2.2. Mobile Target Defense (MTD). Zhao et al. [21] introduced the idea of moving target defense against and proposed a defense method that uses intermediate fingerprint jumps to display a jump fingerprint to the attacker, which increases the attacker's exploration space and integrates the interaction process of fingerprint attack and defense. Modeled as a signal game, analyze the equilibrium point of the game, and propose the optimal defense strategy. Sengupta et al. [22] pointed out that it can benefit delay the spread of network attacks by hiding the real response and responding with random responses, thereby confusing the attacker. Sharma et al. [23] proposed a flexible random virtual IP multiplexing method FRVM, which enables the host to have multiple random virtual IP addresses that change over time, multiplexed to the real IP address of the host, multiplexed, or demultiplexed use events to dynamically remap all virtual network addresses of the host to increase the attacker's cost. Zhang et al. [24] combined end hop and routing hop, and proposed a two-hop communication based on an SDN. This increases the complication for the attacker to obtain the complete communication data.

Mobile target defense is to dynamically change the environment of cyberspace targets and continuously adjust

the attack surface. Attackers are forced to use massive resources to continuously analyze and detect the structure of this change, and the difficulty increases with time. However, this type of method needs to consider users' normal needs in the network. High-speed address hopping may interrupt the communication in the connection while defending against fingerprint attacks, and the legal communication between legal hosts is seriously affected.

2.3. Set Honeypots. Wang and Wu [11] proposed a new architecture of a hybrid honeypot system, which combines the characteristics of high and low interaction honeypots for network topology simulation and attack traffic migration. The system can simulate a large and real network to capture attackers, and redirect high-level attacks to a high-interaction honeypot to capture attacks and conduct further analysis. Honeynets is a network architecture that uses multiple honeypots to deceive attackers and analyze their malicious behavior. Kyung et al. [12] designed an SDN honeynet (HONEYPROXY) to monitor all internal traffic globally, using a new connection management mechanism that spans different honeypots in the network to support the honeypot transition. By multicasting malicious traffic to relevant honeypots, and choosing responses that do not contain fingerprint indicators, fingerprint attacks are avoided, and data capture capabilities are improved. But the honeynet was built under the condition that the honeypot can capture fingerprint attacks.

Honeypots use trap technology to deceive the attacker while studying the characteristics of the attacker. However, traditional honeypots have the disadvantages of static configuration and fixed location. Once discovered or bypassed by the attacker, the honeypot will immediately become invalid. A major problem faced by some honeypot solutions is the overhead of system resources. Honeypots need to operate continuously even if no attack occurs.

2.4. Other Methods. Khorsandroo and Tosun [13] proposed various methods of defense of different types of detection packets. For ICMP packets, after the controller and the switch complete the handshake process, the controller actively installs rules to make the response time of scanning traffic independent of the network state. However, when the number of hosts throughout the system is significant, this method will make the flow of each switch. There are many useless matching rules in the table. TCP packets use TCP proxy in the data plane to defend. This method requires that the proxy server be effective in defending. Once the proxy server is hijacked by an attacker, the entire network will be in a more dangerous state. For UDP packets, defense is carried out by limiting the forwarding rate of detection packets and filtering methods. The results of UDP detection packet fingerprint attacks are largely dependent on the ICMP-related information received from the target destination, filtering specific types of ICMP message delivery, and limiting ICMP. The response rate makes the UDP scan traffic very slow and greatly reduces its reliability. As long as time permits, the attacker may break through the defense and obtain information about the target. Cusack et al. [14] used machine learning to write a stream processor and used random forests and binary classifiers to use these rich stream records to fingerprint malicious network activities without deep packet inspection. Nonetheless, real-time is extremely paramount in the detection of fingerprint attacks. Malik et al. [15] proposed a control plane-based coordination of various complex threats and attacks. The mechanism consists of a hybrid CUDA-supported DL-driven architecture that utilizes the predictive capabilities of long- and short-term memory (LSTM) and convolutional neural networks (CNN) for efficient and timely detection of multi-vector threats and attacks. Krzysztof et al. [16] introduced a dedicated integrated security framework based on SDN and introduced how to use this method to detect and mitigate scanning activities based on TCP and SYN.

It should be noted that the mobile target defense method adopted in [21–24] and set honeypots defense method mentioned in [11, 12] are mainly aimed at the detection of the entire SDN architecture and target host information in the network. The data packet delay forwarding defense [9, 10, 17–20]. The authors of [13–16] are mainly aimed at the fingerprint attack of the flow table matching rule. This paper focuses on the fingerprint attack defense against flow table matching rules. Considering that SDN fingerprint attacks are very different from previous attacks against SDN, these solutions cannot prevent the SDN fingerprint attacks studied in this article. Therefore, it is of great significance to propose a defense scheme against SDN fingerprint attacks.

3. System Design

3.1. Problem Statement. Through problem analysis, it can be seen that the attacker mainly judges whether the constructed detection packet triggers the interaction between the data plane and the control plane, observes the change of the round-trip delay RTT of the packet, and infers the fingerprint information of the matching rule in the network. This article proposes a fingerprint attack defense mechanism that combines active interference and passive detection against SDN fingerprint attacks. Due to the "three layers three interfaces" architecture characteristics of SDN, the attackers can infer whether the detected detection packet triggers the interaction between the data plane and the control plane by observing the change of the round-trip delay RTT of the constructed detection packet, and further infer the fingerprint information such as network type and relevant matching rules. Based on the analysis of the introduction, this paper proposes a fingerprint attack defense mechanism combining active interference and passive detection for SDN fingerprint attacks.

3.2. System Model of SDFADF. The system framework of the mechanism proposed in this paper is shown in Figure 3, which is composed of a Monitor, an active interference module, and a passive detection module. Among them, the Monitor monitors and regulates the entire system in real-time. The active interference module is the focus of this

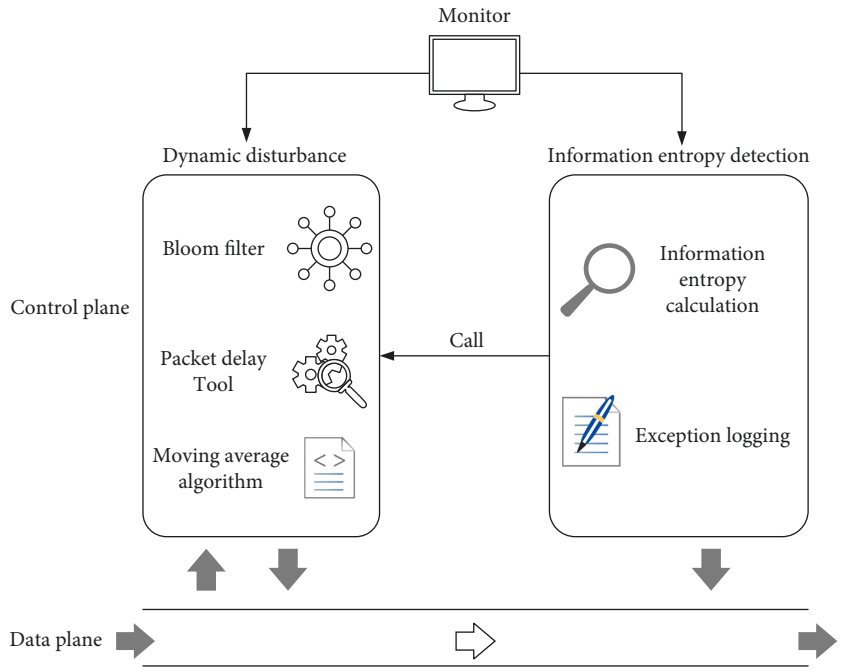


Figure 3: System framework of SDFADS.

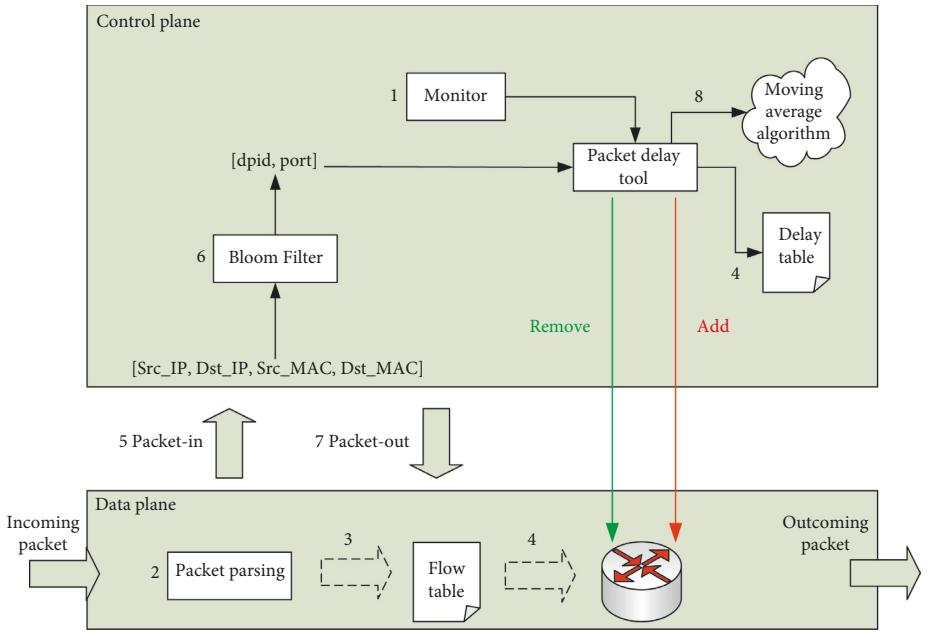


Figure 4: Dynamic disturbance module workflow.

article, and the passive monitoring module is used as an auxiliary module to better adjust the real-time attacks. The detailed introduction of the active interference module and passive detection module is shown in Figure 3.

3.3. Dynamic Disturbance Module. Since the attacker infers the fingerprint information by observing the change in the round-trip delay (RTT) of the detection packet, here we propose for the first time to use the moving average algorithm combined with Bloom Filter and the packet delay tool to dynamically change the RTT of the packet. Disturbance achieves the effect of confusing the RTT of the packet, thereby confusing the attacker.

As shown in Figure 4, active interference is mainly composed of the Monitor module, Bloom Filter module, and disturbance module. The entire workflow is as follows:

(1) As the incoming packet, first perform step 1 to initialize and start the Monitor module to monitor

the entire network in real-time. After the data flow is input to the data plane, perform step 2 packet header parsing to obtain the header field information of the data packet, and then perform step 3 to query the Flow Table for related processing rules that match the data packet.

(2) If the match is successful, perform step 4, update the corresponding counter, and perform the corresponding actions. When the Monitor module detects that the first data packet is successfully matched, execute the (Add) disturbance module, call the packet delay tool, and Delay Table to compare the current switch, and the port number specified by the rule is processed for the delay. The Delay table records the average value of the RTT when the rule is installed in the network environment (tested 100 times). When the Monitor module detects that the number of matches of the flow table rule reaches the set threshold n. That is after the existing n data packets are successfully matched, execute (Remove) to remove the delay processing of the above-mentioned switch and the corresponding port, and subsequent data packets will continue to match the corresponding flow table rule to be forwarded. If there is no corresponding matching rule, encapsulate the data packet and perform step 5, Packet-in to the control plane. After the control plane receives the Packet-in message, it constructs a four-element group [Src_IP, Dst_IP, Src_MAC, Dst_MAC] composed of source IP, destination IP, source MAC, and destination MAC. At this time, step 6 is executed and the four-element group Group input Bloom Filter module determines whether the data packet corresponding to the quadruple appears for the first time, if it appears for the first time, record the switch ID and port number [dpid, port] corresponding to the current data packet for subsequent data The package delay tool is called.

(3) Perform step 7, Packet-out issues the corresponding forwarding rule of the data packet to the Flow table; the Monitor module detects that the first packet matches the rule and executes the (Add) disturbance module, and at the same time, executes step 8 to call the packet delay. The tool and the moving average algorithm delay processing the [dpid, port] obtained by the Bloom Filter in step 6. As the data packet forwarding is performed, the packet processing speed on the data plane is several orders of magnitude faster than the software-based control plane. Through analysis, it can be seen that the main time difference is the packet in between the data plane and the control plane, and the controller's calculation and processing of the data and the time required for Packet-out, so the delay added to the first packet in this article is the time difference between the start of Packet-in and the end of Packet-out. Since the entire network may have more than one OpenFlow Switch, and communication may pass through multiple switches, the moving average algorithm MA (Moving Average) is used for optimization to reduce the volatility of the measurement time difference.

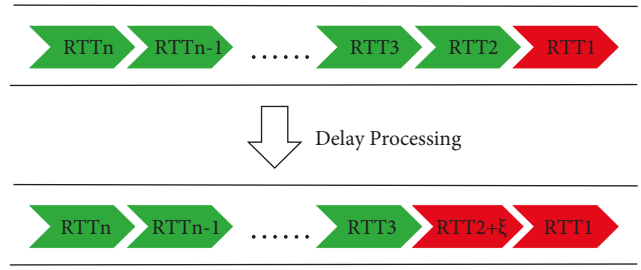


Figure 5: RTT comparison of packets before and after processing.

(4) When the Monitor module detects that the number of matches of the flow table rule reaches the set threshold n, that is, the existing n data packets are successfully forwarded, execute the (Remove) operation to remove the delay processing of [dpid, port] obtained by the Bloom Filter in step 6. Subsequent data packets will continue to match the corresponding flow table rules to be forwarded smoothly.

At this point, the entire active interference is completed, and the effect of delaying forwarding processing on subsequent n data packets of the first data packet is realized. The specific execution Algorithm 1 of the above process is as follows: Among them, Flow is the incoming flow, n is the set number of data packets that need to be delayed, and the Delay table records the average RTT during the installation of this rule in the network environment (100 tests), R and R′ are Matching rules.

The data packet delay tool used in this paper is Traffic Control (TC), which is used for the flow control of the Linux kernel. It mainly realizes the flow control by establishing a queue at the output port. This paper mainly uses TC to delay and forward data packets at the output network card.

The Bloom Filter in the algorithm is also called the Bloom filter, which is used to quickly determine whether an element is in a set. It is composed of an extremely long binary vector (bit vector) and multiple Hash functions. Map the element to a unique point in the bit vector, and set the value of the point to 1. When you need to determine whether an element is not in the set, you only need to check whether the value of the point corresponding to the element is 1.

The p-order MA model in the algorithm is defined as follows:

$$X(t) = a_0 + e_t + a_1 e_{t-1} + \cdots + a_p e_{t-p}. \quad (1)$$

When $a_0 = 0$ in formula (1), it is called a centralized MA (p) model; the highest price parameter a_p is not 0. The random interference item sequence e_t is a zero-mean white noise sequence. The model used in this mechanism is the centralized MA (p) model, where a_p is the processing time of

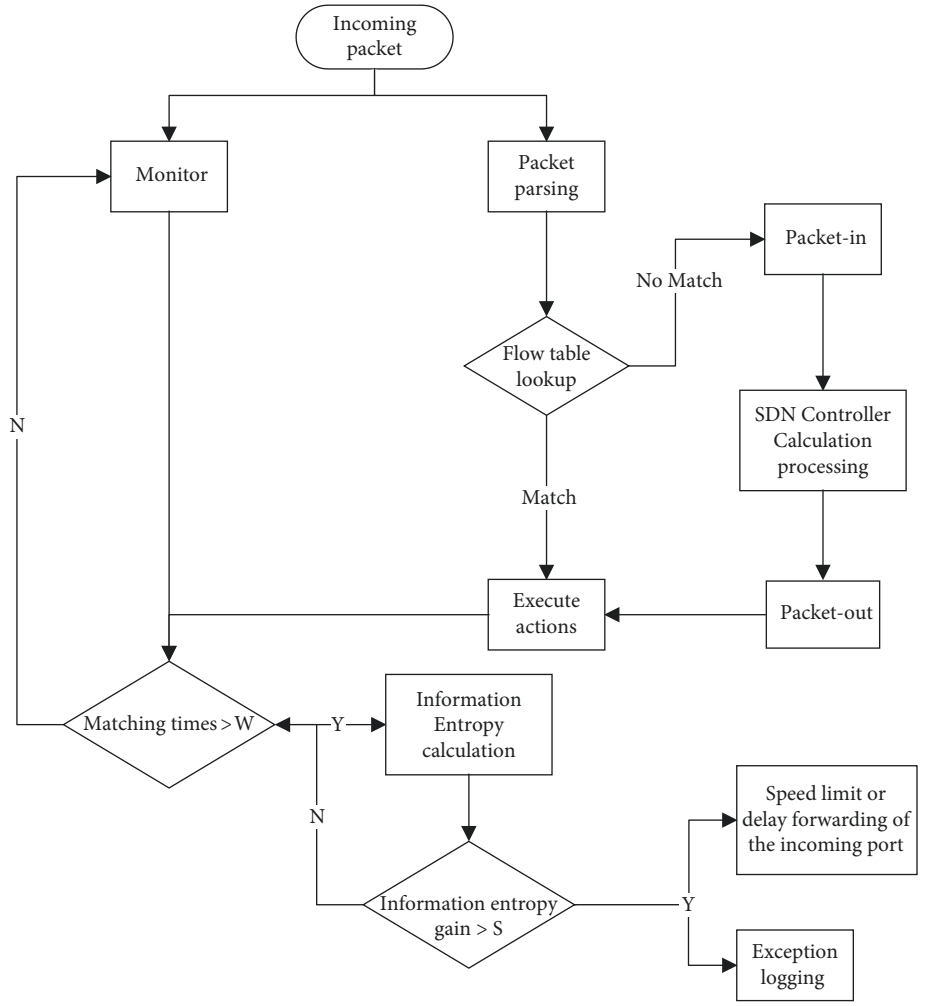


Figure 6: Flowchart of the information entropy detection module.

each switch, so the delay value T added to subsequent data packets after MA processing is calculated as follows:

$$T = t_1\theta_1 + t_2\theta_2 + \cdots + t_n\theta_n. \tag{2}$$

Among them, t_n is the time required for the data packet to go from the previous switch to the nth switch and calculation processing, and θ_n is the transmission weight of the nth switch in the entire communication process and meets:

$$\theta_1 + \theta_2 + \cdots + \theta_n = 1. \tag{3}$$

Furthermore, through the model of each switch in this experimental environment is the same, θ_n is the reciprocal of the number of switches, therefore formula (2) can be transformed into

$$T = \sum_{p=1}^{n} \frac{t_p}{n}, \tag{4}$$

where n is the number of packets passing through the switch.

In the MA model constructed in this paper, the processing delay of the switch through which the detection packet passes is approximately regarded as the RTT difference between the first packet and the subsequent packet, the time required for the packet to pass through each switch is recorded, and then the final required delay value is calculated according to the weight.

As shown in Figure 5, we can observe the changes before and after the packet RTT delay processing. Given that a single packet is delayed, this has little impact on the overall performance of the network. It can be seen from the above that as a consequence of the interaction between the data plane and the control plane, RTT1 will be much larger than RTT_2,...RTT_n, which can be approximated as $\mathrm{RTT}_2 \approx \cdots \mathrm{RTT}_n$. After processing, make

$$\mathrm{RTT}_2 + \xi \approx \mathrm{RTT}_1. \tag{5}$$

That is the processing process of the 8th and 19th lines of the above algorithm. The ξ in formula (5) is a value that dynamically changes under the action of the moving

```
Input: Flow, n, Delay table
Output: Flow of the 2 to n + 1 packets is delayed
(1)  Monitor ()
(2)  while TRUE do:
(3)      packeti ← Flow
(4)      P = Parser (packet1)
(5)      if R in (Flow table with P satisfied):
(6)          R.action (packeti)
(7)          counters + 1
(8)      Add (dpid, port) with Delay table
(9)      while counters = = n:
(10)         Remove (dpid, port)
(11)   else:
(12)       Packet-in, t1 ← nowtime
(13)       V = [Src_IP, Dst_IP, Src_MAC, Dst_MAC]
(14)       if Bloom Filter (V) = = 1:
(15)           [dpid, port]
(16)       Packet-out, t2 ← nowtime
(17)       R'.action(packeti)
(18)       counters + 1
(19)       Add (dpid, port) with MA (t2 − t1)
(20)       while counters = = n
(21)           Remove (dpid, port)
(22)       end while
(23) end while
```

Algorithm 1: Active jamming algorithm.

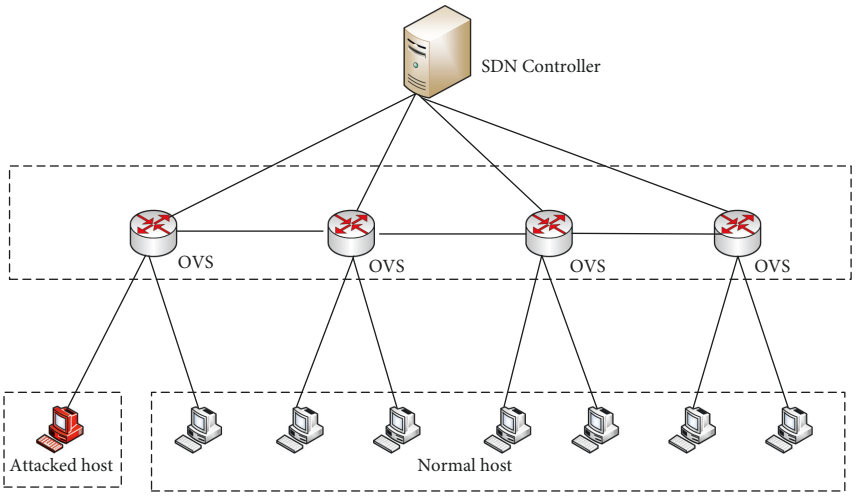


Figure 7: Experimental topology.

Table 1: Comparison of information entropy gain standard deviation with different thresholds.

W	Standard deviation
70	2.037137
80	2.057096
90	2.016084
100	1.966543

Table 2: Experiment host-related information.

Host	IP	MAC
Attacked host	10.0.0.1	00 : 00 : 00 : 00 : 00 : 01
Normal hosts	10.0.0.2	00 : 00 : 00 : 00 : 00 : 02
	10.0.0.3	00 : 00 : 00 : 00 : 00 : 03
...	...	...
	10.0.0.8	00 : 00 : 00 : 00 : 00 : 08

average algorithm according to the number of switches that the forwarding process passes through, it can be considered that $\xi = T$.

3.4. Information Entropy Detection Module. OpenFlow is currently the most influential protocol in the SDN framework [25]. The matching field that can be used as a forwarding rule

Table 3: ICMP probe packet parameter settings.

Attacker IP	Destination host IP	Number of packets
10.0.0.1	10.0.0.2	$n1$
	10.0.0.4	$n2$
	10.0.0.6	$n3$

Table 4: TCP/UDP probe packet parameter settings.

Attacker IP	Destination host IP	Source port/destination port	Number of packets
10.0.0.1	10.0.0.4	80/80–80 + $n4$	$n4$
10.0.0.1	10.0.0.6	80/80–80 + $n5$	$n5$

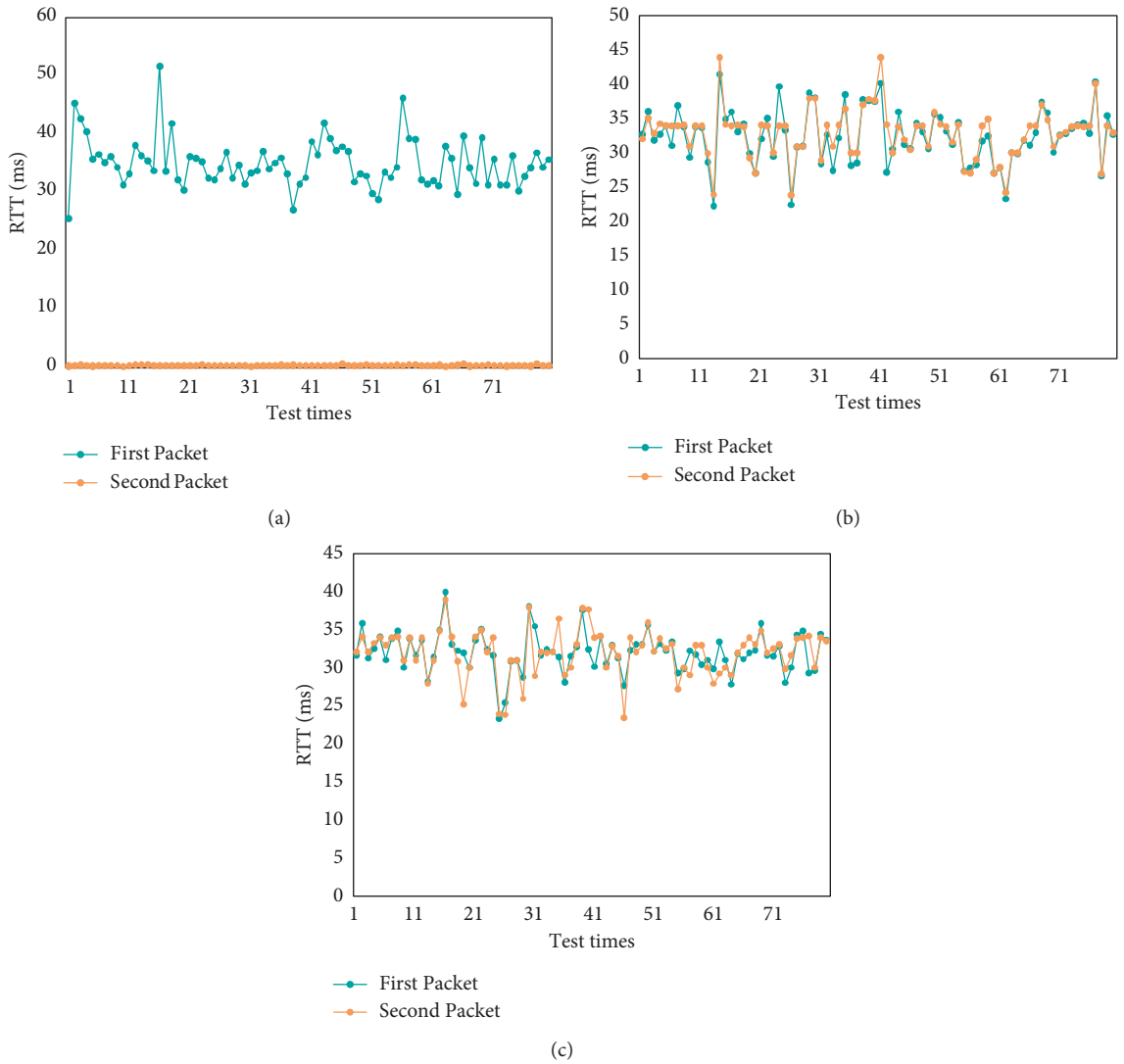


Figure 8: RTT distribution of probe packets. (a) RTT distribution without interference. (b) RTT distribution after interference processing (no initial rules). (c) RTT distribution after interference processing (initially regular).

is continuously increasing as the OpenFlow version continues to update. In OpenFlow 1.0, there were only 12 types. In OpenFlow 1.5, the supported matching fields have been increased to 44 [26]. Attackers unknow the specific matching rules when attacking SDN fingerprints, which is necessary to construct various data packets for detection [27]. This article focuses on information entropy gain detection for some commonly used matching fields [28–31], such as source IP, destination IP, source MAC, destination MAC, TCP/UDP source port, and TCP/UDP destination port. The other

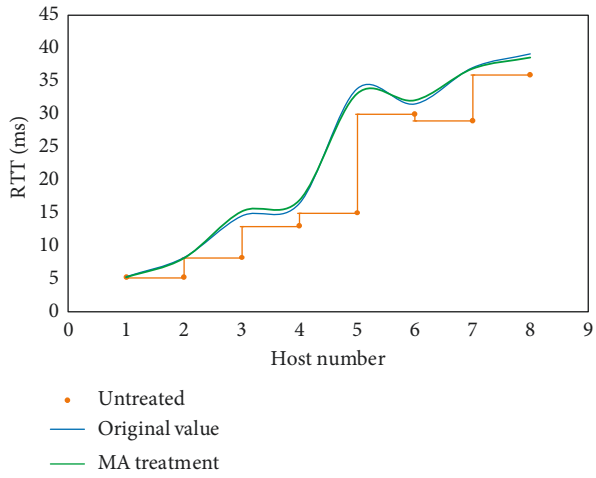


Figure 9: Comparison of delay values before and after MA processing.

matching condition detection methods are the same, so it is no longer covered her repeatedly. The passive detection workflow chart is shown in Figure 6.

The Monitor module monitors the entire network in real-time. If there is the corresponding matching rule after the data stream input is parsed in the packet header, the matching actions will be executed, otherwise, the Packet-in, controller calculation processing, and Packet-out will be installed and the corresponding matching rules will be installed. Next, the number of matches of the matching rule is judged. If the threshold is not reached, monitor monitoring will be continued. Otherwise, the information entropy calculation will be triggered, and the threshold will be judged after the information entropy of each matching condition is obtained. The ingress port corresponding to the matching condition is processed for speed limit or delayed forwarding, and the exception log is recorded for the administrator to view.

The definition formula of information entropy (empirical entropy) is as follows:

$$H(X) = -\sum_{x,\chi} p(x)\log\ p(x). \tag{6}$$

As a multi-characteristic data stream, to solve the information gain [32], the concept of conditional entropy is introduced here. Conditional entropy $H(Y|X)$ represents the uncertainty of the condition of the random variable Y under the condition of the known random variable X, which satisfies the following formula:

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

The information gain infoGain satisfies the following formula:

$$\text{infoGain}(X|Y) = H(X) - H(Y|X). \tag{8}$$

The threshold W (window) mentioned above affects the detection and calculation of the entropy value of the data packet in a short time. If the threshold W is too small, the experimental results are not convincing. If the threshold W is too large, the entropy value of each feature of the data packet will unchange significantly. The selection of W is related to the experimental environment. After testing and calculating in this experimental environment, $W = 80$ is the best choice, which is shown in Table 1.

For the threshold S, in this experimental environment,

$$S = \left(\log\ H(X)_W + \text{infoGain}_{\max}\right) * 0.5, \tag{9}$$

where $\log\ H(X)_W$ is the empirical entropy of the data set under window W and $\text{infoGain}_{\max}$ is the maximum information entropy gain of the data set.

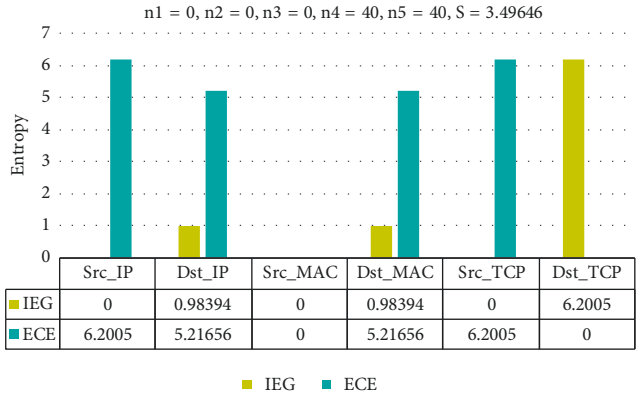
4. Simulation Environment

4.1. Simulation Setup. In this section, we will design experiments to test the defense effect of this mechanism and perform related simulation experiments under the experimental topology shown in Figure 7. This article uses Mininet to build an SDN, which consists of 4 SDN switches and 8 hosts. The deployment controller host system is Ubuntu18.06, the controller uses Python-based open-source Ryu4.3, and Scapy is used to construct the required data packets. The relevant information of the host is shown in Table 2. One host is the attacker and the other seven hosts are the normal network users.

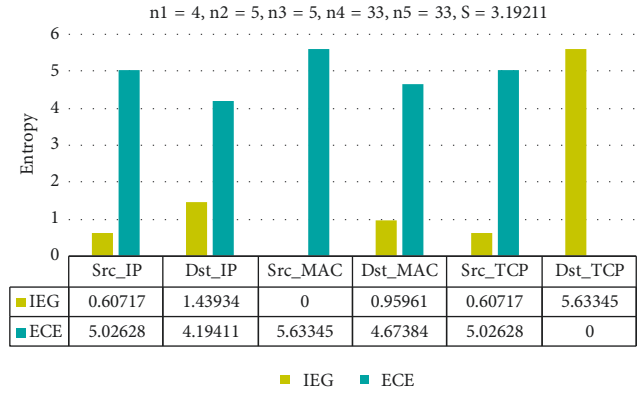
In the evaluation of the passive detection strategy, we constructed a detection data packet under $W = 80$ (window) for testing. The parameter settings of the detection packet are shown in Tables 3 and 4:

4.2. Comparison Analysis. In order to evaluate the effect of the defense mechanism proposed in this paper, we conducted several groups of experiments. First, let the attacker construct and send the detection data packet when the dynamic disturbance mechanism is not started, send two detection packets continuously for the destination address of each detection, and record the round-trip delay RTT value of the two detection packets, then start the dynamic disturbance mechanism under the initial irregular and regular conditions, and repeat the above steps. Suppose that the number of data packets that need to be delayed forwarded in this experiment is $n = 1$, that is, the second data packet is delayed forwarded.

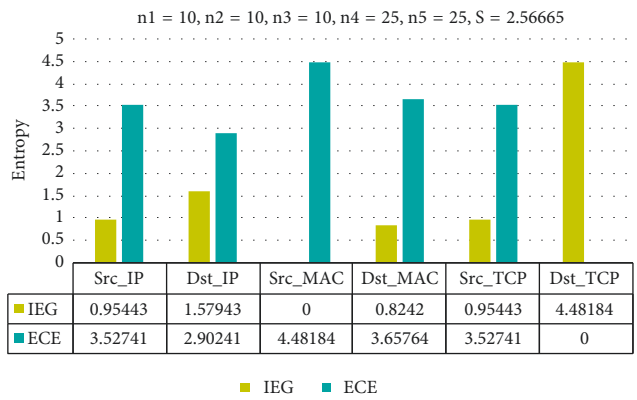
As shown in Figure 8(a), the RTT values of two consecutive detection ICMP packets of the attacker to the normal host are counted 80 times, when without interference processing is performed. There is a significant difference among the RTT value of the first packet and the second packet. It can be inferred that the detection packet causes the interaction between the control layer and the data layer. After the interference is processed on the detection packets, two consecutive ICMP detection data packets are sent again. The statistical results are shown in Figures 8(b) and 8(c). The RTT values of the detection packets tend to be uniformly distributed. Therefore, it can effectively confuse the RTT values of the first two detection packets and effectively bemuse the attackers after deploying the dynamic perturbation strategy.



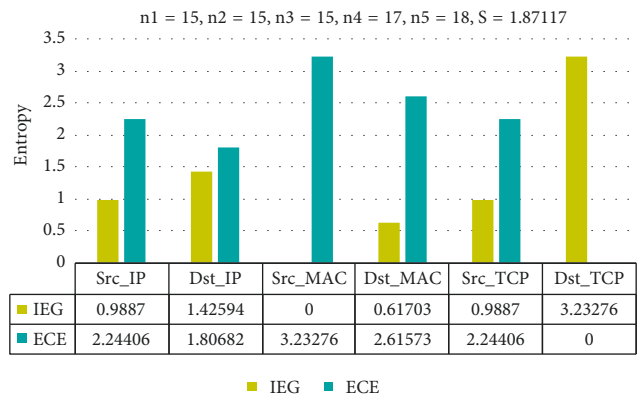
(a)



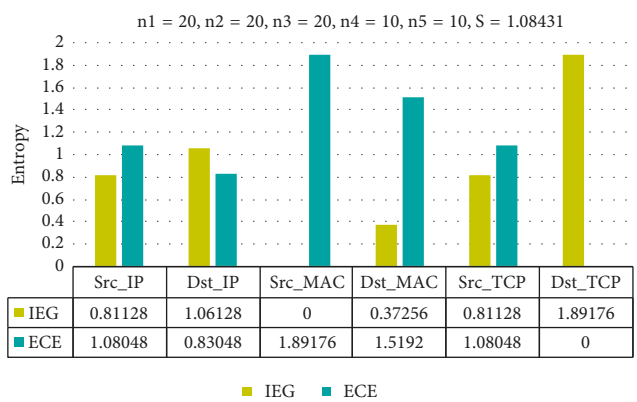
(b)



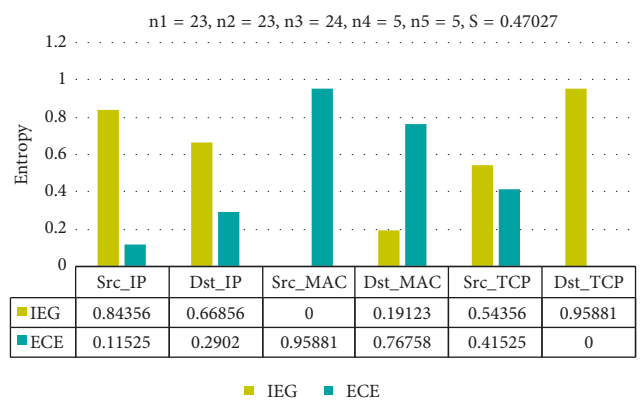
(c)



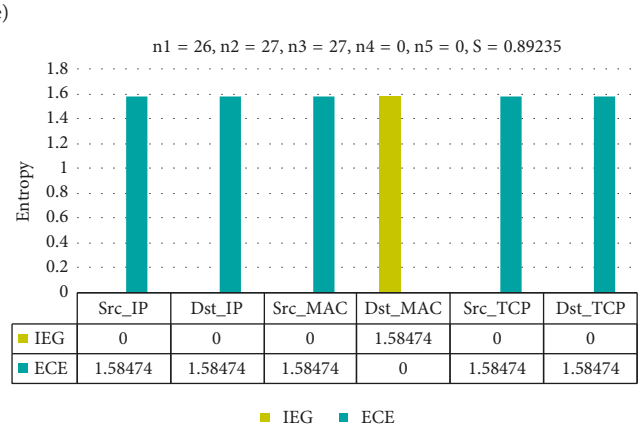
(d)



(e)



(f)



(g)

Figure 10: Entropy detection results under different detection packet settings.

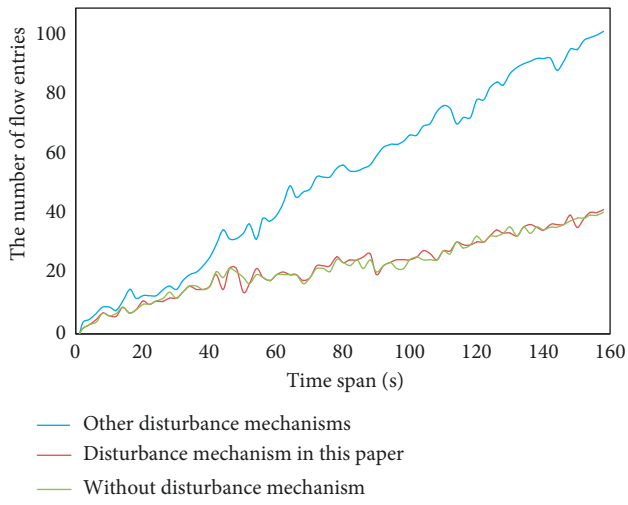


Figure 11: Variation diagram of flow entries with time under different disturbance strategies.

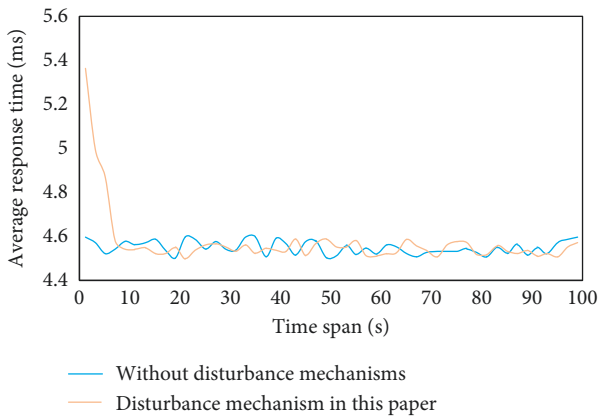


Figure 12: Average response time comparison.

The comparison of delay values before and after MA processing is shown in Figure 9. It can be observed that before MA processing, the parameter settings in the traditional probability model are affected by human factors. The delay value added each time when communicating with a fixed destination host is fixed, which is easier to be recognized by the attackers. Therefore, this paper proposes to use the moving average algorithm to fit the delay value when passing through multiple switches. Through experimental comparison, it can be realized that the delay value after MA processing tends to dynamic transformation, which is closer to the RTT value of the actual first packet, and the effect of confusion is more effective and not accessible to be recognized by the attacker.

Figures 10(a)–10(g) are the results of information entropy detection of different types and different numbers of detection packets under window W, where IEG is Information entropy gain and ECE is Empirical conditional entropy. According to formula (9), the threshold S of different types and different numbers of detection packets under the window W can be obtained to further determine which feature may be fingerprinted by the attacker as the detection condition, and at the same time, the information entropy gain is the minimum feature corresponding to the switch and the port number is subject to the speed limit or delayed forwarding processing, and abnormal log storage is recorded.

4.3. Security Analysis. The general interference mechanism implements different delay operations on different packets by defining a new action bucket selection logic for the group table, which will lead to an enormous increase in the entries of the flow table. In order to measure the impact of different strategies on switch flow table entries, we tested the changes of flow table entries over time under different defense strategies. The experimental results are shown in Figure 11. As the test time passes, the number of flow table entries under other disturbance mechanisms increases significantly compared with that under the production of the undisturbed machine, and the measurement peak is about 120% higher than the normal value, which may further lead to flow table overflow. The disturbance mechanism proposed in this paper does not involve the installation of the flow table, consequently, the changing trend of flow table entries tends to be the same as that under the production of an undisturbed machine, and the influence of convection meter capacity is similarly limited.

In order to comprehensively measure the impact of the defense strategy proposed in this paper on the overall network performance, we test the average response time of data packets before and after the experiment. The experimental results are shown in Figure 12. It can be observed that when the defense mechanism is enabled, the initial average response delay is slightly higher than that when the defense mechanism is not enabled due to the delayed processing of the second packet. However, as the test time passes, the average response time of all subsequent packets tends to be delayed under normal conditions. Therefore, the negative of the mechanism proposed in this paper on the overall network performance is comparatively limited and can meet the performance requirements of the normal network while meeting the requirements of defenders.

5. Conclusion

As the first step of the network attack, the fingerprint attack is one of the most serious threats to network security. Attackers obtain the key fingerprint information of the target network to lay the foundation for subsequent more threatening attacks. In this paper, a fingerprint attack defense mechanism combining active interference and passive detection is proposed for SDN fingerprint attacks. This mechanism uses the principle of fingerprint attack to confuse the attackers by dynamically disturbing a small number of data packets. At the same time, it makes real-time processing feedback to the network combined with the information entropy detection mechanism. The evaluation results of the countermeasure demonstrate its effectiveness in defense of fingerprinting attacks in SDNs with minor overheads, preventing the leakage of network control

information. In future work, we are concerned to introduce the related technologies of network fingerprint information hiding to make the optimization of this scheme more effective, which will be the content of the next optimization research.

Data Availability

The data used to support the finding of this study are included within the article and in the following link (https://github.com/nwxshmfx/sdndata).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially funded by the Guangxi Foundation 2022 (Grant no. GXNSFAA035629), National Natural Science Foundation of China (Grant no. 61966007), Key Laboratory of Cognitive Radio and Information Processing, Ministry of Education, China (Grant No. CRKL180201), Guangxi Cooperative Innovation Center of Cloud Computing and Big Data, Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems (Grant No. 1716), and Guangxi Key Lab of Wireless Wideband Communication and Signal Processing (Grant No. GXKL06200116).

References

[1] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[2] S. Liu, M. K. Reiter, and V. Sekar, "Flow Reconnaissance via Timing Attacks on SDN switches," in *Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 196–206, IEEE, Atlanta, GA, USA, June 2017.

[3] A. Azzouni, O. Braham, and T. M. T. Nguyen, "Fingerprinting OpenFlow Controllers: The First Step to Attack an SDN Control plane," in *Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, Washington, DC, USA, December 2017.

[4] J. Sonchack, A. J. Aviv, and E. Keller, "Timing SDN Control Planes to Infer Network configurations," in *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, pp. 19–22, ACM, NewYork, NY, USA, March 2016.

[5] S. Panjwani, S. Tan, and K. M. Jarrin, "An Experimental Evaluation to Determine if Port Scans Are Precursors to an attack," in *Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)*, pp. 602–611, IEEE, Yokohama, Japan, July 2005.

[6] M. Yu, T. He, and P. McDaniel, "Flow Table Security in SDN: Adversarial Reconnaissance and Intelligent attacks," in *Proceedings of the IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 1519–1528, IEEE, Toronto, ON, Canada, July 2020.

[7] Z. Yadong, C. Kaiyue, and Z. Junjie, "Exploiting the Vulnerability of Flow Table Overflow in Software-Defined Network: Attack Model, Evaluation, and Defense," *Security and Communication Networks*, vol. 2018, Article ID 4760632, 15 pages, 2018.

[8] M. Zhang, G. Li, L. Xu et al., "Control Plane Reflection Attacks and Defenses in Software-Defined Networks," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 623–636, 2021.

[9] H. Cui, G. O. Karame, and F. Klaedtke, "On the fingerprinting of software-defined networks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2160–2173, 2017.

[10] T. Wang and H. Chen, "A lightweight SDN fingerprint attack defense mechanism based on probabilistic scrambling and controller dynamic scheduling strategies," *Security and Communication Networks*, vol. 2021, Article ID 6688489, 23 pages, 2021.

[11] H. Wang and B. Wu, "SDN-Based Hybrid Honeypot for Attack capture," in *Proceedings of the 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pp. 1602–1606, IEEE, Chengdu, China, March 2019.

[12] S. Kyung, W. Han, and N. Tiwari, "HoneyProxy: Design and Implementation of Next-Generation Honeynet via SDN," in *Proceedings of the 2017 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–9, IEEE, Las Vegas, NV, USA, October 2017.

[13] S. Khorsandroo and A. S. Tosun, "Time Inference Attacks on Software-Defined Networks: Challenges and countermeasures," in *Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 342–349, IEEE, San Francisco, CA, USA, July 2018.

[14] G. Cusack, O. Michel, and E. Keller, "Machine Learning-Based Detection of Ransomware Using SDN," in *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, pp. 1–6, ACM, NewYork, NY, USA, March 2018.

[15] J. Malik, A. Akhunzada, I. Bibi, M. Imran, A. Musaddiq, and S. W. Kim, "Hybrid Deep Learning: An Efficient Reconnaissance and Surveillance Detection Mechanism in SDN," *IEEE Access*, vol. 8, Article ID 134695, 2020.

[16] C. Krzysztof, G. Marcin, M. Wojciech, N. Piotr, and P. Ż, "SDN-Based Mitigation of Scanning Attacks for the 5G Internet of Radio Light System," in *Proceedings of the Availability, Reliability and Security*, ACM, NewYork, NY, USA, August 2018.

[17] S. Deng, X. Gao, Z. Lu, and X Gao, "Packet Injection Attack and Its Defense in Software-Defined Networks," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 695–705, 2018.

[18] J. Hou, M. Zhang, Z. Zhang, W. Shi, B. Qin, and B. Liang, "On the fine-grained fingerprinting threat to software-defined networks," *Future Generation Computer Systems*, vol. 107, pp. 485–497, 2020.

[19] H. Yuwen, L. Zhang, and Z. Wang, "Probability-based Delay Scheme for Resisting SDN scanning," in *Proceedings of the 2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pp. 1096–1101, IEEE, Chengdu, China, October 2016.

[20] S. Achleitner, T. F. La Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha, "Deceiving Network Reconnaissance Using SDN-Based Virtual Topologies," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1098–1112, 2017.

[21] Z. Zhao, F. Liu, and D. Gong, "An SDN-Based Fingerprint Hopping Method to Prevent Fingerprinting attacks," *Security and Communication Networks*, vol. 2017, Article ID 1560594, 12 pages, 2017.

[22] S. Sengupta, A. Chowdhary, A. Sabur, A. Alshamrani, D. Huang, and S. Kambhampati, "A Survey of Moving Target Defenses for Network Security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1909–1941, 2020.

[23] D. P. Sharma, D. S. Kim, and S. Yoon, "FRVM: flexible random virtual IP multiplexing in software-defined networks," in *Proceedings of the 2018 17th IEEE international conference on trust, security and privacy In Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*, pp. 579–587, IEEE, New York, NY, USA, August 2018.

[24] C. Zhang, G. U. Xuehui, and C. Meng, "Anti-sniffing attack method based on software-defined network," *Journal of Computer Applications*, vol. 38, no. 11, pp. 3258–3262, 2018.

[25] Q. Y. Zuo, M. Chen, G. S. Zhao, C. Y. Xing, G. M. Zhang, and P. C. Jiang, "Research on OpenFlow-Based SDN Technologies," *Journal of Software*, vol. 24, no. 5, pp. 1078–1097, 2013.

[26] O. F. S. Specification, "Version 1.5. 0," *Open Networking Foundation*, 2014.

[27] H. Zhang, Z. Cai, and Q. Liu, "A Survey on Security-Aware Measurement in SDN," *Security and Communication Networks*, vol. 2018, Article ID 2459154, 12 pages, 2018.

[28] K. Kalkan, L. Altay, G. Gur, and F Alagoz, "JESS: Joint Entropy-Based DDoS Defense Scheme in SDN," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2358–2372, 2018.

[29] P. Kumar, M. Tripathi, A. Nehra, M. Conti, and C. Lal, "SAFETY: Early Detection and Mitigation of TCP SYN Flood Utilizing Entropy in SDN," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1545–1559, 2018.

[30] Z. Chen, F. Jiang, and Y. Cheng, "XGBoost Classifier for DDoS Attack Detection and Analysis in SDN-Based cloud," in *Proceedings of the 2018 IEEE International Conference on Big Data and Smart Computing (Bigcomp)*, pp. 251–256, IEEE, Shanghai, China, January 2018.

[31] T. A. Tang, L. Mhamdi, and D. McLernon, "Deep Learning Approach for Network Intrusion Detection in Software Defined networking," in *Proceedings of the 2016 International Conference on Wireless Networks and mobile Communications (WINCOM)*, pp. 258–263, IEEE, Fez, Morocco, October 2016.

[32] G. T. Landi, M. Paternostro, and A. Belenchia, "Informational steady states and conditional entropy production in continuously monitored systems," *PRX Quantum*, vol. 3, no. 1, Article ID 010303, 2022.