Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 8898949, 16 pages
https://doi.org/10.1155/2020/8898949

Research Article

Behavior Anomaly Detection in SDN Control Plane: A Case Study of Topology Discovery Attacks

Li-Der Chou,[1] **Chien-Chang Liu**,[1] **Meng-Sheng Lai**,[1] **Kai-Cheng Chiu**,[1] **Hsuan-Hao Tu**,[1] **Sen Su**,[2] **Chun-Lin Lai**,[2] **Chia-Kuan Yen**,[2] **and Wei-Hsiang Tsai**[2]

[1]*Department of Computer Science and Information Engineering, National Central University, Taoyuan 32001, Taiwan*
[2]*Information & Communication Research Division, National Chung-Shan Institute of Science and Technology, Taoyuan 32546, Taiwan*

Correspondence should be addressed to Li-Der Chou; cld@csie.ncu.edu.tw

Received 22 July 2020; Revised 13 September 2020; Accepted 2 November 2020; Published 21 November 2020

Academic Editor: Juraj Gazda

Software-defined networking controllers use the OpenFlow discovery protocol (OFDP) to collect network topology status. The OFDP detects the link between switches by generating link layer discovery protocol (LLDP) packets. However, OFDP is not a security protocol. Attackers can use it to perform topology discovery via injection, man-in-the-middle, and flooding attacks to confuse the network topology. This study proposes a correlation-based topology anomaly detection mechanism. Spearman's rank correlation is used to analyze the network traffic between links and measure the round-trip time of each LLDP frame to determine whether a topology discovery via man-in-the-middle attack exists. This study also adds a dynamic authentication key and counting mechanism in the LLDP frame to prevent attackers from using topology discovery via injection attack to generate fake links and topology discovery via flooding attack to cause network routing or switching abnormalities.

1. Introduction

In recent years, with the rapid development of smart devices, mobile devices, and network technologies, the number of network devices has exploded, and numerous network services have emerged. Network services connect many devices to the Internet; thus, users' real-time processing capabilities and diversified service requirements for network services have also increased. However, traditional networks use a decentralized architecture, and network packets are difficult to monitor and manage. Therefore, a software-defined networking (SDN) architecture based on centralized management was proposed [1].

The SDN architecture separates the control layer of the traditional network device from the data layer and manages the routing and forwarding of network packets through one or several network controllers. SDN controllers can adjust network rules for centralized management with the topology connection status of the entire network. SDN integrates the link layer discovery protocol (LLDP) [2], which is called the OpenFlow discovery protocol (OFDP) [3], for topology discovery. The controller generates LLDP packets in a specific format to detect links between switches, and it maintains network topology information to facilitate optimal routing decisions for each service.

However, most SDN controllers perform topology discovery through OFDP without security mechanisms, such as RYU [4], Floodlight [5], OpenDaylight [6], and POX [7]. Therefore, an attacker can easily confuse the current network topology through a topology discovery attack. SDN controllers do not implement an authentication mechanism on OFDP. Any attacker can forge LLDP packets and send them to the switch to perform a topology discovery attack. The OFDP cannot determine whether the SDN controller has generated LLDP data packets; thus, making judgments becomes more difficult for the SDN controller. Attackers can use topology discovery attacks to generate multiple false links. As a result, the SDN controller cannot determine the routing and switching paths of each service or even forward normal traffic through the attacker's computer.

Related research has no complete detection mechanism for topology discovery attacks. The contributions of the present study can be highlighted as follows:

(1) This study designs and implements a topology anomaly detection mechanism based on correlation analysis, which can detect an abnormal LLDP packet

(2) The mechanism generates an authentication key in the LLDP packets and calculates the amount of received LLDP packets of each switch to avoid topology discovery via injection and flooding attacks

(3) For the topology discovery via man-in-the-middle attack, the mechanism uses Spearman's rank correlation coefficients [8] to analyze the network traffic between links and measure the round-trip time (RTT) of each LLDP packet

The remainder of this paper is organized as follows. Section 2 presents the related works. Section 3 describes the proposed methods. Section 4 presents the experimental results. Section 5 concludes the study.

2. Related Works

2.1. SDN. The characteristic of SDN is that physical equipment is used as a basic resource and is only responsible for data processing. SDN abstracts the control components of network packet processing and provides a unified management and control interface for controlling the upper-level components of packets. Network management engineers can use software programs to replan and define network topology, resource allocation, and flow control mechanisms. In addition to key OpenFlow technology based on this architecture, in recent years, many integrated SDN technologies have been discussed and developed, such as Programming Protocol-independent Packet Processors [9], Stratum [10], and ONOS [11].

The SDN architecture is divided into three layers, namely, infrastructure, control, and application layers. The top application layer includes various unique services and applications. The middle control layer is responsible for the configuration of network resources and network management by the SDN controller. The lowest basic equipment layer processes and forwards data based on protocols, such as OpenFlow [12].

2.2. OpenFlow. OpenFlow is a communication protocol that can access the data plane of a switch or router. The SDN controller [13] can communicate with the switch or router through OpenFlow. The network equipment supporting OpenFlow includes the following two components: (1) a medium called the OpenFlow Channel, which can communicate with the SDN controller, and (2) an OpenFlow flow table for recording network packet processing principles.

Flow table is composed of many flow entries. Each entry defines the actions required for each network flow. Flow entry includes fields, such as match and action fields. The match field is used to identify and classify each corresponding flow, including multiple identification fields in the second to fourth layers of the seven layers of the network OSI. The action field defines the actions that must be performed after each packet is matched, including forward to port, drop, and send to controller.

The OpenFlow switch only needs to compare the rules of the flow table to forward the data and communicate with the SDN controller through OpenFlow Packet-In and Packet-Out messages. The OpenFlow switch compares each received packet to the flow table. If a corresponding rule is found, then the packet is modified or forwarded according to the rule. If no corresponding rule can be found, then the received packet is forwarded to the SDN controller, which is called Packet-In. The SDN controller analyzes each packet from the OpenFlow Packet-In and issues rules to the OpenFlow switch for each flow, which is called Packet-Out.

2.3. OFDP. The OpenFlow switch does not have an LLDP mechanism; thus, the SDN controller performs topology discovery through the OFDP. OFDP sends LLDP packets to the switch and disassembles those return packets to obtain the current network topology. The OFDP cycle is different for each controller platform. The topology discovery cycle of Ryu controller is one second [14]. Figure 1 shows the operational flow of the OFDP.

Step 1. The SDN controller inquires the number of ports being used on each switch and generates fresh LLDP packets for each port. The packet includes the switch number and the port number and transmits the LLDP to the designated switch through an OpenFlow Packet-Out message.

Step 2. After receiving the LLDP packet, the switch sends it according to the specified port number.

Step 3. The switch that received the LLDP packet will send it back to the SDN controller through the Packet-In. After the SDN controller disassembles the packet, it can know which of the two switches and of the two ports are connected to each other.

Through the above process, the SDN controller performs the same action for each switch in the network and obtains the topology information of the entire network [15]. To maintain the latest network topology, the SDN controller performs the OFDP regularly.

2.4. LLDP. The LLDP is the second-layer network protocol in the seven-layer architecture of the OSI network [16]. It is an optional protocol in the IEEE 802 LAN protocol. The switch exchanges device and link information with each other through the LLDP protocol; such information includes functions supported by the device, management address, and port status.

Type/length/value (TLV) is the unit that forms the LLDP data unit (LLDPDU). Each TLV represents a message. The switch composes device and link information into different TLVs and encapsulates them into LLDPDUs. The LLDPDU is transmitted through the 802.3 Ethernet frame. Figure 2 shows the LLDP frame [17].

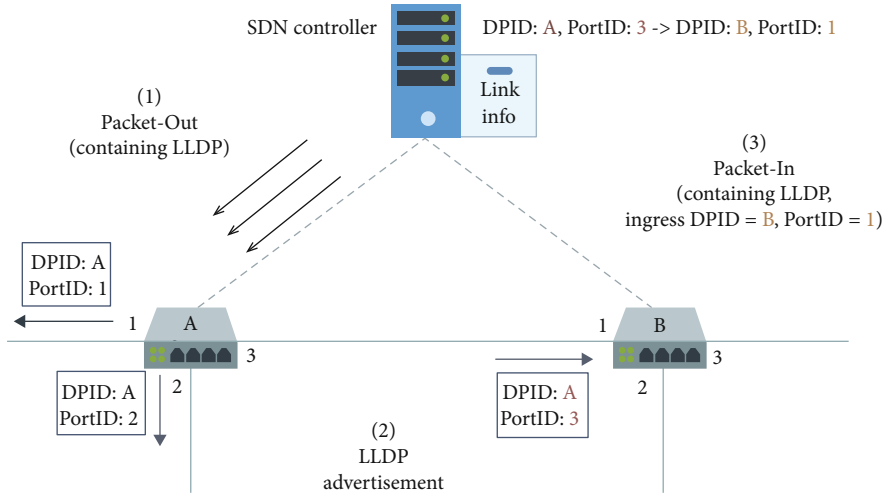

Figure 1: Operational flow of OFDP: encapsulate and send LLDP packets and disassemble them to obtain the current network topology.

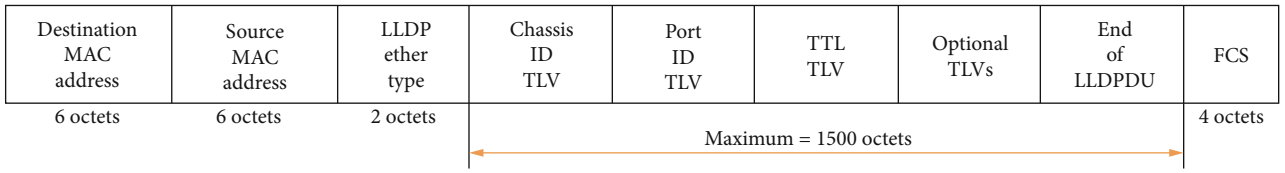

Figure 2: LLDP frame format. The destination MAC address is a fixed set of MAC addresses 01-80-C2-00-00-0E. The source MAC address is the MAC address of the sending LLDP device. The LLDP ether-type code is 88-CC. The chassis ID TLV is the identification code of the switch; port ID TLV is the port number; TTL TLV records the lifetime of this LLDPDU; optional TLV is the selected TLV in the LLDPDU; and end of LLDPDU is used to inform the receiver the body of this LLDP has ended.

The LLDPDU contains the mandatory and optional TLVs. The mandatory and optional fields can add up to 1500 octets. The mandatory TLVs are TLV types 0–3, which are end of LLDPDU, chassis ID, port ID, and time to live. The optional TLVs are TLV types 4–8 and 127, which are port description, system name, system description, system capabilities, management address, and organizationally specific TLVs. TLV types 9–129 are reserved fields and are not currently defined in the standard. Therefore, data types can be defined here to avoid errors in access to network devices using standard LLDPDUs. The timestamp field added to the LLDPDU in this study is also defined in the reserved field.

2.5. Topology Discovery Attacks. The SDN controller collects network topology information through the OFDP. However, the LLDP is not a secure mechanism. Given the lack of authentication messages, attackers can confuse the current network topology status by forging LLDP packets [18]. The three common topology attacks are injection, man-in-the-middle, and flooding attacks [19]. The purpose of the first two is to confuse the current network topology, and that of the third is to consume the computing resources of the SDN controller.

Figure 3(a) shows the topology discovery via injection attack. The attacker fabricates the LLDP packet and fills the chassis ID and port ID in the LLDPDU into the identification codes of other OpenFlow switches on the network. The attacker sends the fabricated LLDP packet to the interface card between the attacker and the switch. After the switch receives the packet, it transmits the packet to the controller through a Packet-In message. The attacker injects a fake link to the topology information because the controller cannot recognize whether the LLDP packet is sent by the controller.

In [20], the optional TLV in the LLDPDU is added to the authentication information. If the fabricated LLDP packet does not contain authentication information, then it will be considered an attack. However, authentication information is a fixed value; thus, the attacker can intercept the LLDP packet sent by the controller to acquire the authentication information and then fabricate the packet containing the authentication field. Thus, in [21], authentication information (KHMAC) is dynamically generated to increase the difficulty for attackers to fabricate the authentication information.

Figure 3(b) shows the topology discovery via man-in-the-middle attack. From the figure, the network has multiple attackers. Instead of spoofing the LLDP packet, Attacker 1 eavesdrops the LLDP packets sent from the SDN controller. Then, Attacker 1 transfers the LLDP packets into other formats, specifically in the pcap format, and transmits it to Attacker 2 who is connected to other switches. After receiving the pcap file, Attacker 2 will convert the format back and send the packet to the switch. After the switch receives the packet, it will transmit the packet to the controller through a Packet-In message. A fake link is added between Attackers 1 and 2. Given that the packet played back is the

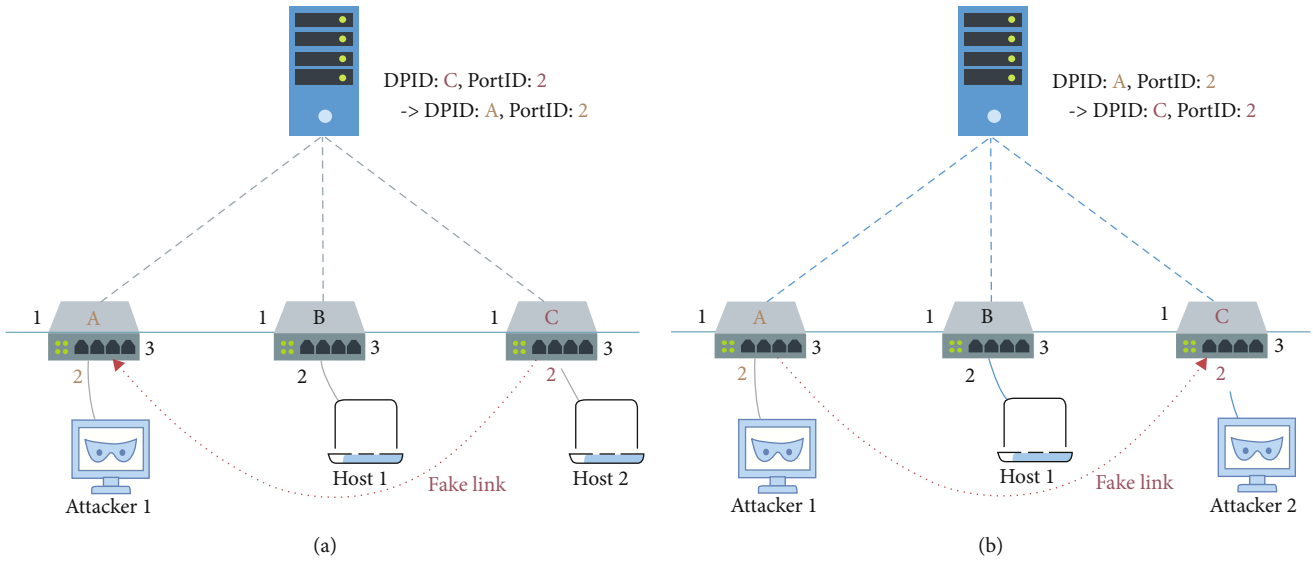

Figure 3: (a) Topology discovery via injection attack. The attacker fabricates the LLDP packet and injects a fake link. (b) Topology discovery via man-in-the-middle attack. Attacker 1 eavesdrops the LLDP packet and transmits it to Attacker 2 to add a fake link between them.

LLDP packet generated by the controller, the authentication field in the LLDPDU cannot be used to avoid the attack.

Literature [22] proposes a statistical analysis of link latencies (SALL) solution to detect topology discovery via man-in-the-middle attack through the latency time of each received LLDP packet. However, if an attacker quickly plays back the LLDP packets, then the latency time will be close to normal, and the attack cannot be detected.

Figure 4 shows the topology discovery via flooding attack. Multiple attackers on the network simultaneously generate multiple LLDP packets and transmit them to the switch. The switch transmits all the received LLDP packets to the controller for analysis, which consumes the computing resources of the SDN controller. Thus, the requests of the normal users cannot be served.

In [23], a secure OpenFlow topology discovery mechanism (sOFDP) is proposed to mitigate the topology discovery via flooding attack by observing the state of the switch. The controller performs OFDP topology discovery only when the connection state of the network changes. Therefore, if the state of the port is unchanged, then even if the attacker sends many LLDP packets, they will be discarded by the switch, and the controller will not be affected. However, the disadvantage of this method is that the attacker can change the state of the port before launching the attack to bypass the detection mechanism. In [24], the secure and lightweight link discovery protocol mechanism (SLDP) is proposed to avoid attacks by creating an eligible port list. Only the LLDP packets received from the port in the list will be accepted. All ports of all switches are on the list when the network starts. However, if an attacker attacks while it is still in the legal list, then the controller cannot avoid being attacked.

Based on the above-mentioned related research, the feature of this study is to propose a behavior anomaly detection mechanism based on correlation analysis and time difference analysis, which is applied to the defense and detection of OpenFlow topology discovery attacks. Compare the four literatures related to the motivations and problems to be solved in this study. The comparison table is shown in Table 1.

2.6. Correlation Coefficient. The correlation coefficient is a statistical technique used to reflect the correlation between two sets of variables. The correlation between the two sets of variables does not fully represent the relationship between them. Common algorithms for calculating the correlation of the variables include Spearman's rank correlation coefficient and Pearson correlation coefficient. Correlation coefficient with absolute values of 0–0.35, 0.36–0.67, and 0.68–1.0 is considered low or weak, modest or moderate, and strong or high correlation, respectively [25].

Spearman's rank correlation coefficient is an algorithm used to calculate the correlation between two sets of variables. It is not affected by outliers in the data, and the calculated data do not need to conform to the normal distribution. However, the data from the two sets of variables must be ordered. The data used to calculate the correlation of traffic between the switches in this study are stored sequentially, and outliers may sometimes exist due to external factors of network users. Thus, this study uses Spearman's rank correlation coefficient. The value of Spearman's rank correlation coefficient is between −1 and 1. A value greater than 0 indicates a positive correlation, and a negative correlation otherwise [26].

3. Proposed Methods

3.1. System Architecture and Design. Figure 5 shows the system architecture in this study, which is divided into three major modules and nine submodules. The three major modules are topology management module, LLDP handling module, and correlation-based topology anomaly detection (CTAD) module. The topology management module and the LLDP handling module detect topology discovery via injection and attacks with verification code information and packet statistical analysis. The CTAD module analyzes the

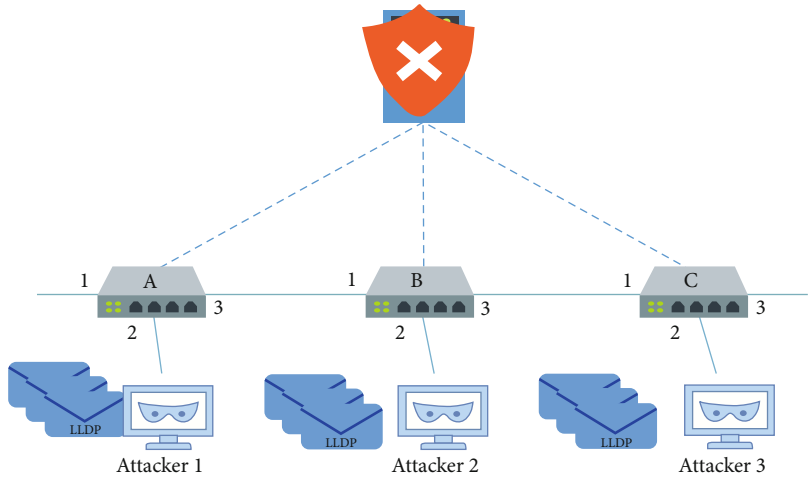

Figure 4: Topology discovery via flooding attack. Multiple attackers simultaneously generate multiple LLDP packets and transmit them to the switch.

Table 1: Related research comparison.

Features	Detection of topology discovery via injection attack	Detection of topology discovery via man-in-the-middle attack	Detection of topology discovery via flooding attack
KHMAC [21]	Supported	Supported	Not supported
SALL [22]	Not supported	Supported	Not supported
sOFDP [23]	Not supported	Not supported	Supported
SLDP [24]	Supported	Not supported	Supported
Proposed	Supported	Supported	Supported

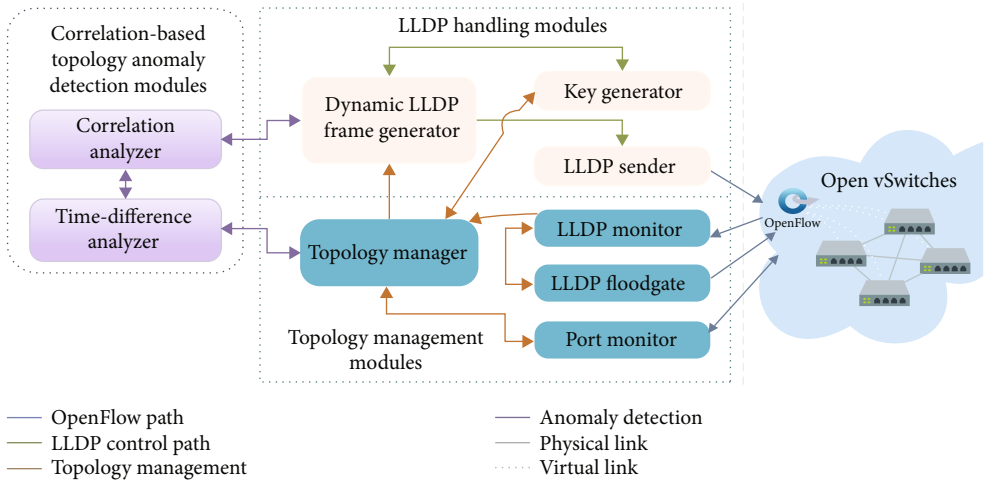

Figure 5: System architecture. It is divided into three major modules and nine submodules.

RTT of LLDP packets and link traffic to detect topology discovery via man-in-the-middle attacks. The nine submodules are topology manager, LLDP monitor, LLDP floodgate, port monitor, dynamic LLDP frame generator, key generator, LLDP sender, correlation analyzer, and time-difference analyzer.

3.1.1. Topology Management Module. This module monitors LLDP packets from switch, records real-time network topology information, and verifies whether a topology discovery was made by injection or flooding attack. This module includes four submodules, namely, topology manager, LLDP monitor, LLDP floodgate, and port monitor.

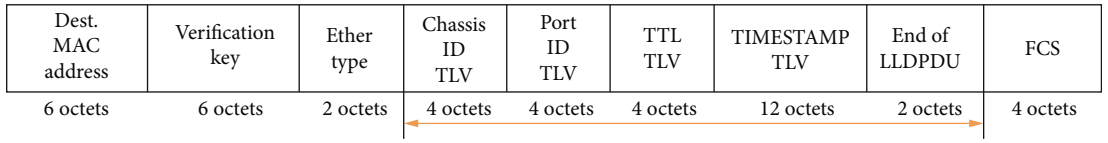

Figure 6: Proposed LLDP packet format. The packet content contains 6 bytes of authentication information (verification key) and 12 bytes of timestamp (TIMESTAMP).

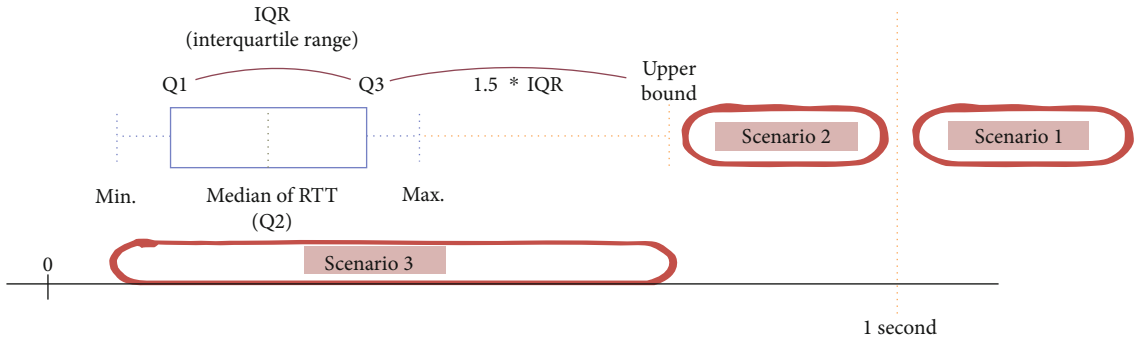

Figure 7: Scenarios of topology discovery via man-in-the-middle attack. Scenario 1: the delay time is greater than one second; scenario 2: the delay time is less than one second, but greater than the upper bound; and scenario 3: the delay time is less than the upper bound.

The topology manager monitors and records the topology status of the entire network by disassembling the LLDP collected by the LLDP monitor. In this manner, the link status between the switches can be obtained.

The LLDP monitor is responsible for parsing whether a Packet-In message received from a switch belongs to the LLDP packet. The LLDP packet is transmitted to the topology manager for further analysis. The LLDP monitor maintains an LLDP packet that receives the quantity table. After each topology discovery period ends, it calculates the number of LLDP packets to determine if any port suffers from topology discovery via flooding attack. Then, this module notifies the LLDP floodgate to block the attacker's network traffic.

The LLDP floodgate controls the flow of LLDP packets on the switch. When the LLDP monitor detects a topology discovery via flooding attack, the LLDP floodgate receives the chassis ID and port ID from the LLDP monitor. This module will perform the flow modification to the related switch through OpenFlow, and the switch will block all LLDP packets from this port.

The port monitor collects the number of switch ports and transmits the information to the topology manager for topology discovery. Moreover, it collects the current network traffic of each port. The correlation analyzer module will notify the port monitor to collect the network traffic of the specified port.

3.1.2. LLDP Handling Module. This module makes LLDP packets in a specific format, generates verification information for the LLDP packets, encapsulates the verification information into the LLDP packets, and sends the LLDP packets to a specific switch. This module includes three submodules, namely, dynamic LLDP frame generator, key generator, and LLDP sender.

The dynamic LLDP frame generator generates LLDP packets of different sizes and encapsulates LLDP verification information into the packets. The dynamic LLDP frame generator receives a list of ports from the topology manager at the beginning of each topology discovery cycle and generates LLDP packets in a specific format according to each port in the list. The packet contains 6 bytes of verification key and 12 bytes of timestamp, as shown in Figure 6. To verify whether a topology discovery via man-in-the-middle attack occurs, the dynamic LLDP frame generator generates 1500 MTU LLDP packets and sends numerous LLDP packets at the specified port to verify the correlation among different link traffic rates.

The key generator generates a set of random MAC addresses at the beginning of each topology discovery cycle to verify whether each LLDP packet returned from the switch is a topology discovery via injection attack. The key generator computes a set of MAC addresses through the current timestamp. The MAC addresses are encapsulated by the dynamic LLDP frame generator into the source MAC address field.

The LLDP sender propagates the LLDP packet encapsulated by the dynamic LLDP frame generator to the specified switch by an OpenFlow Packet-Out message.

3.1.3. CTAD Module. This module measures the time difference of the RTT of each LLDP packet and analyzes the correlation of network traffic on each link to determine a topology discovery via man-in-the-middle attack. This module includes two submodules, namely, correlation and time-difference analyzer.

Figure 7 illustrates that topology discovery via man-in-the-middle attacks can be divided into three scenarios. The first scenario is that the delay for the attacker to replay LLDP packets is greater than one second. In this attack scenario, the

SDN controller adopts a new set of authentication keys for the new topology discovery cycle because the attack delay time is greater than one second. Thus, the attack cannot take effect. The second scenario is that the delay time for the attacker to replay LLDP packets is greater than the upper quartile calculated by the box-and-whisker plot [27] plus 1.5 times interquartile range (IQR), but less than one second. The LLDP packet delay time in this attack scenario is greater than the usual value; thus, it will be judged as abnormal. The third scenario is that the delay time for the attacker to replay LLDP packets is less than the upper bound calculated by the box-and-whisker plot. This type of situation is difficult to detect because the delay is normal. In this study, the correlation analyzer is used to confirm the topology discovery via man-in-the-middle attack.

The correlation analyzer analyzes the correlation of network traffic on each link. When the fake link produced by the topology discovery via man-in-the-middle attack does not exist, the LLDP packets replayed by the attacker to establish this link must pass through other links in the network topology. If the SDN controller sends numerous LLDP packets to a fake link, then the traffic on this fake link and other links to the destination switch will increase simultaneously. The correlation analyzer sends numerous LLDP packets to the source of the new link through the dynamic LLDP frame generator and collects the network traffic of each port from the destination switch through the port monitor. In this manner, the correlation analyzer can calculate whether the link traffic to the destination switch is correlated. If the traffic to the destination switch of the new link is highly correlated with the traffic of other links on the destination switch after the Spearman's correlation analysis (correlation is greater than 0.7), then the topology manager will be notified to delete this added link.

The time-difference analyzer extracts the timestamp in the LLDP packet when the LLDP packet is received to obtain the RTT of the LLDP packet. If any new link is generated, then this module will filter out the historical RTT record of the link associated with this new link and calculate the upper bound of the RTT. If the RTT is greater than the upper bound, then it will be considered a topology discovery via man-in-the-middle attack.

3.2. System Operation Process and Mechanism. This section explains the operation process of the proposed topology management module, LLDP handling module, and CTAD module. Table 2 presents the variables and definitions of topology discovery and CTAD mechanism.

3.2.1. CTAD Operation Process. This section describes the OpenFlow topology discovery process before and after starting the CTAD mechanism and the operation process of the CTAD mechanism. The OpenFlow topology discovery process is divided into two phases. After Phase 1 is completed, the topology manager has the current network state T (Figure 8), and Phase 2 starts the CTAD mechanism (Figure 9). The process of the Phase 1 of OpenFlow topology discovery is described as follows.

Table 2: Variables and definitions of topology discovery and CTAD mechanism.

Variable	Definition
n	nth time slot of topology discovery
Key^n	Verification key for time slot n
Key^{lldp}	Verification key extracted from the received LLDP packet
Switch_i	ith switch
Switch_l	lth switch ($l \neq i$)
Port_j^i	jth port of the ith switch
Port_m^l	mth port of the lth switch ($l \neq i$)
timeout	Signal that informs the topology manager that the cycle of topology discovery has ended
P_i	List of ports of the ith switch
P_l	List of ports of the lth switch ($l \neq i$)
$\text{Link}_{(l,m)}^{(i,j)}$	Link between the jth port of ith switch and the mth port of lth switch
lldp_j^i	LLDP packet sent from the jth port of the ith switch
T	Topology information of the entire SDN
$N_{i,j}^{\text{lldp}}$	Number of LLDP packets received from Port_j^i
N^{lldp}	List of $N_{i,j}^{\text{lldp}}$
key_element	String containing number 0–9 and character A–F
sPath_l^i	Dijkstra's shortest path between the ith switch and lth switch
Skip_j^i	Tag representing if Port_j^i bypass the LLDP flooding check
$\text{Rtt}(i, j, n)$	RTT of lldp_j^i at time slot n
$Rtt(s)$	List of all RTT records in last s second(s)
$\text{IRQ}_{T(s)}$	IQR of $\text{RTT}(s)$ in the last s second(s)
$Q_{T(s)}^3$	The third quartile of $\text{RTT}(s)$ in the last s second(s)
send_rate	The rate of dynamic LLDP frame generator generating LLDP packets
Rate_j^i	The traffic rate of Port_j^i
Rate_m^l	The traffic rate of Port_m^l ($l \neq i$)
R_j^i	The list of traffic data of Port_j^i
R_m^l	The list of traffic data of Port_m^l ($l \neq i$)
c	The correlation coefficient between R_j^i and R_m^l
duration	The execution time of function lldp_gen_random (Port_j^i, *duration*)

Step 1. The topology manager collects the port list of each OpenFlow Switch_i on the network through port monitor.

Step 2. The port monitor requests the port list P_i for each Switch_i through OpenFlow. After receiving the request, the switch returns the port list to the port monitor and then sends it to the topology manager.

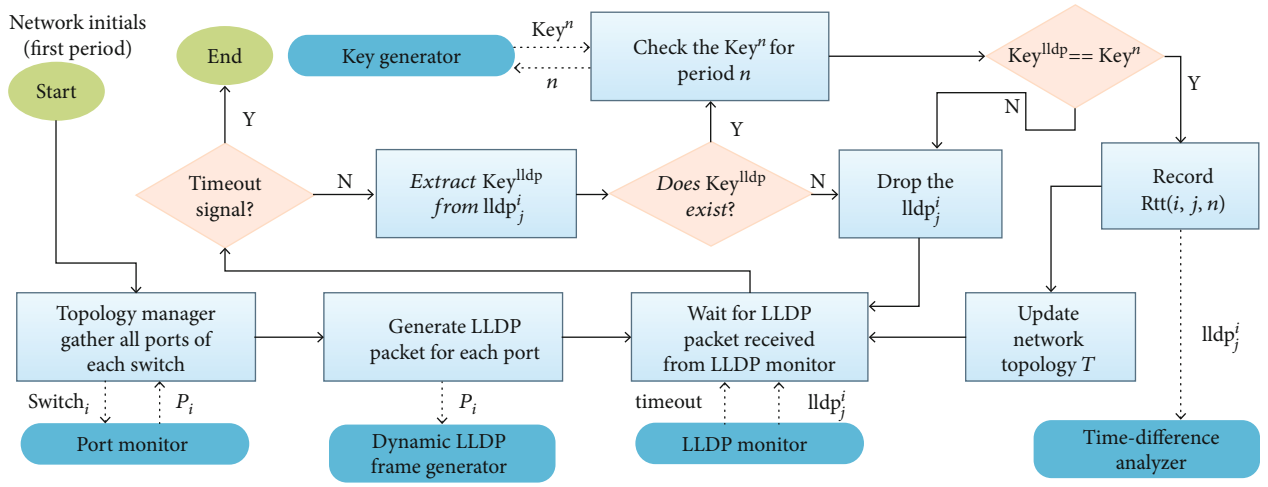

FIGURE 8: Phase 1 of the OpenFlow topology discovery.

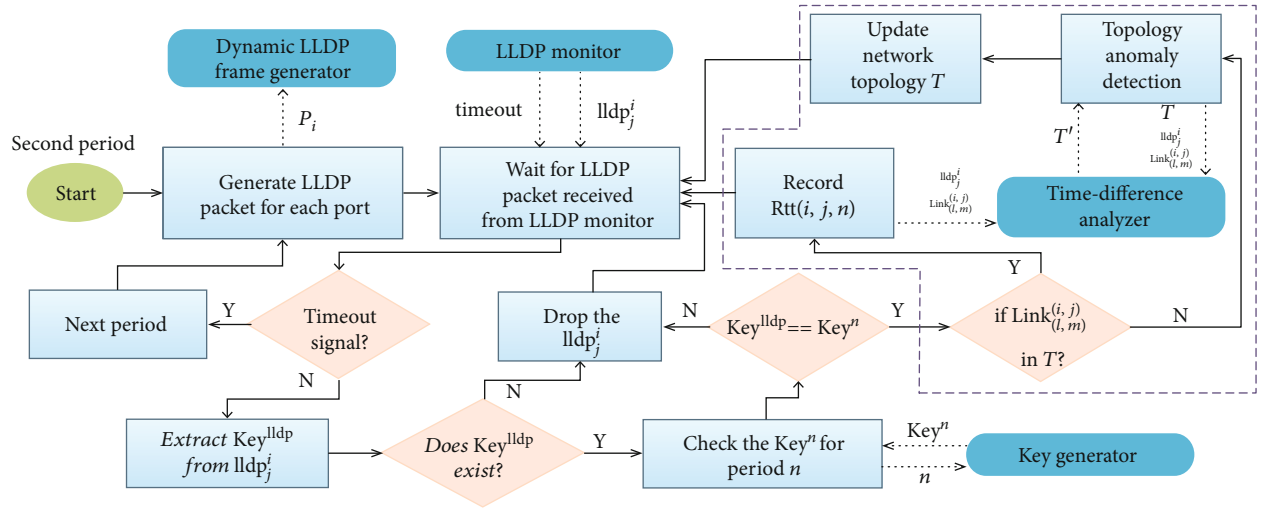

FIGURE 9: Phase 2 of the OpenFlow topology discovery.

Step 3. The topology manager starts topology discovery by sending the port list P_i to the dynamic LLDP frame generator and calling the function lldp_gen_fix() to send LLDP packets to each port.

Step 4. The dynamic LLDP frame generator generates LLDP packets lldp_j^i for the ports in port list P_i and request Key^n of this topology discovery period n by calling the function gen_key(). The dynamic LLDP frame generator encapsulates Key^n and the current timestamp into this lldp_j^i and then uses OpenFlow Packet-Out message to send lldp_j^i to Switch_i through the LLDP sender.

Step 5. The LLDP monitor monitors whether the SDN controller has received an LLDP packet lldp_j^i and sends it to the topology manager.

Step 6. The topology manager unpacks and analyzes the received lldp_j^i. First, the LLDP authentication key is extracted. If an authentication key is present, then the LLDP conforms to the packet format generated by the dynamic LLDP frame generator. Otherwise, it represents a fake LLDP packet, which will be discarded.

Step 7. If Key^{lldp} exists in lldp_j^i, then the topology manager obtains the verification Key^n of the current topology discovery cycle n from key manager and compares it with Key^{lldp}. If the comparison result is different, then lldp_j^i is discarded. Otherwise, the topology manager stores the link information $\text{Link}_{(l,m)}^{(i,j)}$ recorded in lldp_j^i to the topology information T and sends lldp_j^i to the time-difference analyzer to calculate $\text{Rtt}(i, j, n)$.

Step 8. Steps 5–7 will continue until the end of this topology discovery cycle. When the LLDP monitor finds that the topology discovery cycle has ended, it sends a timeout to the topology manager to enter the Phase 2 of OpenFlow topology discovery.

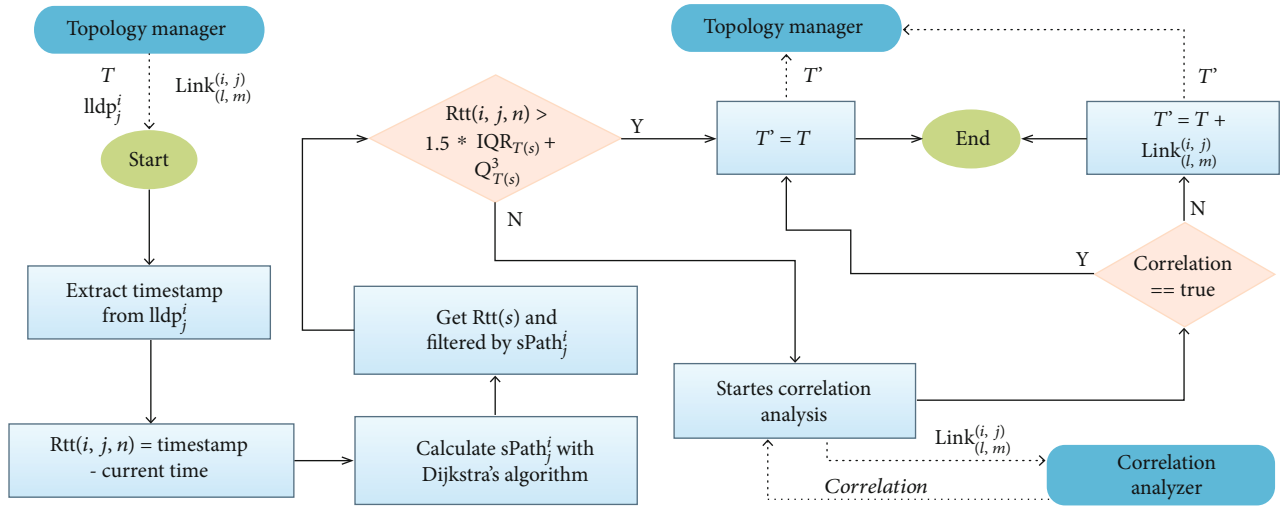

Figure 10: Time difference analysis.

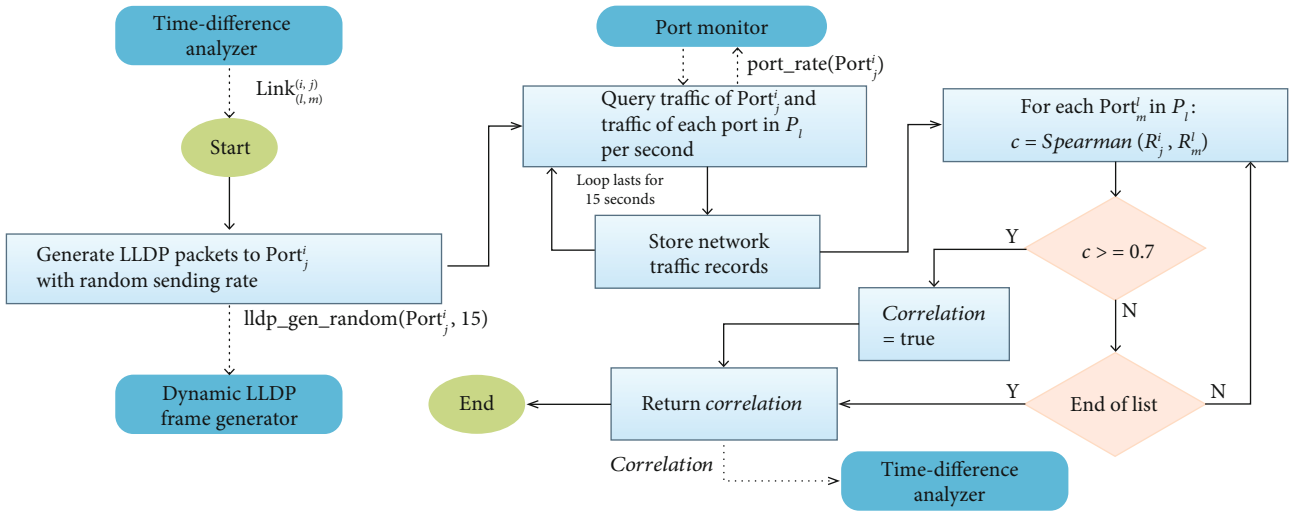

Figure 11: Correlation analysis.

Step 9. The LLDP monitor monitors the number of LLDP packets $N_{i,j}^{\text{lldp}}$ received on each port of each switch and records them in the statistical list N^{lldp}. The LLDP monitor will confirm the number of each $N_{i,j}^{\text{lldp}}$ in N^{lldp}. If the device connected to Port$_j^i$ sends numerous LLDP packets, function block_port() is called to the LLDP floodgate module to stop sending the LLDP packets received by Port$_j^i$ to the SDN controller.

After the Phase 1 of topology discovery is completed, the topology manager has the current topology T, and Phase 2 will start the CTAD mechanism. The process of the Phase 2 of OpenFlow topology discovery is described as follows.

Step 1. The topology manager starts the topology discovery and generates LLDP packets to the port list P_i of each Switch$_i$ through the dynamic LLDP frame generator.

Step 2. The LLDP monitor monitors whether the SDN controller has received an LLDP packet lldp$_j^i$ and sends it to the topology manager. After the topology discovery cycle ends, whether the device connected to Port$_j^i$ sends numerous LLDP packets will be confirmed.

Step 3. The topology manager unpacks and analyzes the received lldp$_j^i$. First, the LLDP authentication key is extracted. If a key exists, then the LLDP conforms to the packet format generated by the dynamic LLDP frame generator. Otherwise, it represents a fake LLDP packet, which will be discarded.

Step 4. If a Key$^{\text{lldp}}$ exists in lldp$_j^i$, then the topology manager acquires the verification Key^n of the current topology discovery cycle n from the key manager and compares it with Key$^{\text{lldp}}$. If the comparison result is different, then lldp$_j^i$ is

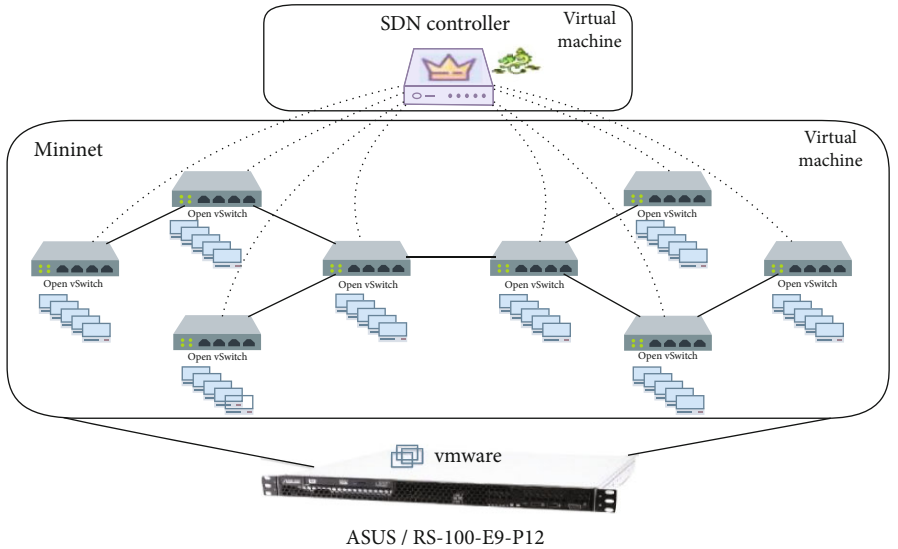

Figure 12: System implementation topology, including 8 Open vSwitches and 40 virtual computers (host).

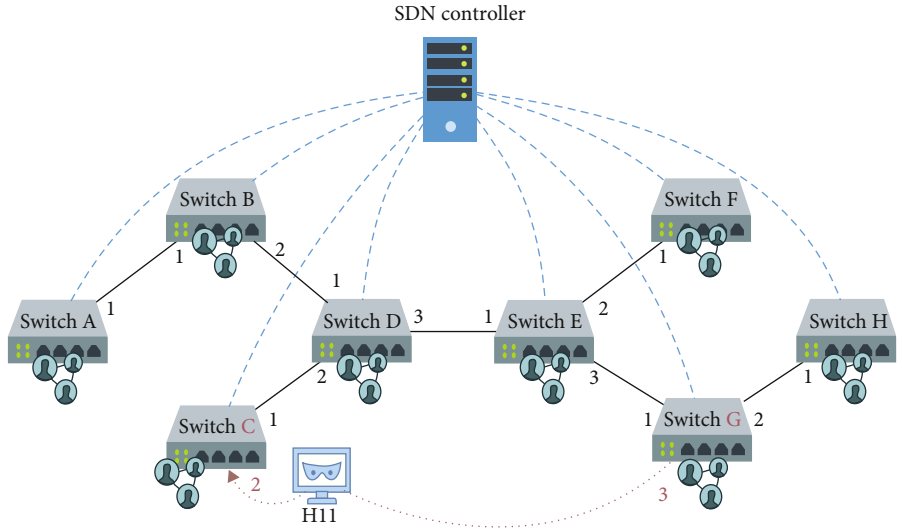

Figure 13: Topology discovery via injection attack. The attacker attempts to forge a (DPID: G, portID: 3→DPID: C, portID: 2) link.

discarded. Otherwise, lldp_j^i is converted to link information $\text{Link}_{(l,m)}^{(i,j)}$ and compared with the current topology information T. If $\text{Link}_{(l,m)}^{(i,j)}$ exists in T, then lldp_j^i is sent to the time-difference analyzer to calculate $\text{Rtt}(i, j, n)$; otherwise, the CTAD mechanism for topology discovery attack analysis will begin.

Step 5. Steps 1–4 are performed for each topology discovery cycle n.

The CTAD mechanism is divided into two phases. The first phase is the time difference analysis (Figure 10), and the second phase is the correlation analysis (Figure 11). This mechanism analyzes whether a topology discovery via man-in-the-middle attack exists. The process of the time difference analysis is described as follows.

Step 1. The topology manager triggers the time-difference analyzer and passes the current topology information T, the added link information $\text{Link}_{(l,m)}^{(i,j)}$, and the LLDP packet lldp_j^i that generates this link.

Step 2. After the time-difference analyzer receives the LLDP packet lldp_j^i from the topology manager, it reads the timestamp to obtain $\text{Rtt}(i, j, n)$ of this lldp_j^i.

Step 3. For Switch_i and Switch_l on the link information $\text{Link}_{(l,m)}^{(i,j)}$, the shortest distance sPath_l^i is calculated by Dijkstra's algorithm.

Step 4. All $\text{Rtt}(s)$ are filtered through sPath_l^i, and only the RTTs related to sPath_l^i are retained.

Table 3: Network topology.

Link	Source DPID	Source port ID	Destination DPID	Destination port ID
1	A	1	B	1
2	B	1	A	1
3	B	2	D	1
4	C	1	D	2
5	D	1	B	2
6	D	2	C	1
7	D	3	E	1
8	E	1	D	3
9	E	2	F	1
10	E	3	G	1
11	F	1	E	2
12	G	1	E	3
13	G	2	H	1
14	H	1	G	2

Step 5. The filtered Rtt(s) is calculated through the box-and-whisker plot to obtain the third quartile $Q^3_{T(s)}$ and the interquartile distance $\mathrm{IRQ}_{T(s)}$. If $\mathrm{Rtt}(i, j, n)$ of lldp^i_j is greater than $Q^3_{T(s)}$ plus $\mathrm{IRQ}_{T(s)}$, then this $\mathrm{Rtt}(i, j, n)$ is the outlier, and the process will skip Step 8; otherwise, this $\mathrm{Rtt}(i, j, n)$ is a normal value. To avoid man-in-the-middle attacks by replaying the topology of LLDP packets at a high rate, it must be verified through correlation analysis.

Step 6. $\mathrm{Link}^{(i,j)}_{(l,m)}$ is passed to the correlation analyzer for analysis.

Step 7. If the correlation reported by the correlation analyzer is true, then the next step is performed; otherwise, the process will skip to Step 9.

Step 8. $\mathrm{Link}^{(i,j)}_{(l,m)}$ is rejected to be added in topology information T.

Step 9. $\mathrm{Link}^{(i,j)}_{(l,m)}$ is added in the topology information T' to be returned to the topology manager.

The correlation analysis is described as follows.

Step 1. The time-difference analyzer triggers the correlation analyzer and passes the link information $\mathrm{Link}^{(i,j)}_{(l,m)}$ for analysis.

Step 2. The correlation analyzer calls the function lldp_gen_random() to trigger dynamic LLDP frame generator to send numerous random LLDP packets to Port^i_j.

Step 3. The dynamic LLDP frame generator sends random rate LLDP packets to Port^i_j.

Step 4. Skip^i_j is set as true to prevent Port^i_j from receiving a large number of LLDP packets from the dynamic LLDP frame generator and returns to the SDN controller.

Step 5. A countdown timer is set for the duration of the correlation analyzer.

Step 6. Send_rate is set to 10–100 Mbps.

Step 7. A large number of 1500 MTU-sized LLDP packets lldp^i_j are created and sent to Port^i_j at the transmission rate of send_rate.

Step 8. Steps 6 and 7 are repeated until the countdown timer ends.

Step 9. The dynamic LLDP frame generator transmits a large number of LLDP packets. At the same time, the correlation analyzer collects traffic rate for all ports P_l and Port^i_j on the destination switch Switch_l of $\mathrm{Link}^{(i,j)}_{(l,m)}$ and triggers the port monitor to collect data by calling port_rate() [28].

Step 10. The port monitor collects the traffic rate Rate^l_m from each port Port^l_m of all P_l and Port^i_j and obtains the aforementioned rate sets R^i_j and R^l_m.

Step 11. The correlation analyzer performs Spearman's rank correlation coefficient analysis on R^i_j collected on source port and R^l_m on all target ports. The analysis result is the correlation coefficient c. If multiple correlation coefficients are greater than 0.7, then $\mathrm{Link}^{(i,j)}_{(l,m)}$ is generated by the topology discovery via man-in-the-middle attack.

Step 12. The correlation analyzer reports the results of correlation analysis to the time-difference analyzer. If the correlation coefficients c is greater than 0.7, then the return correlation value is true and false otherwise.

4. Results and Discussion

The experimental platform uses the Open vSwitch to simulate a realistic OpenFlow switch. The Open vSwitch is built on a virtual machine to implement the overall OpenFlow switch environment through a Mininet simulator. The underlying architecture uses the server ASUS/RS100-E9-PI2 as the hypervisor, as shown in Figure 12. The SDN controller uses the Ryu controller provided by the Ryu SDN Framework Community, which communicates with the Open vSwitch on another virtual machine through OpenFlow. In this experiment, two attackers exist. Among the 40 virtual computers simulated by Mininet, 2 are used as attackers. LLDP generator [29], TCPdump [30], and TCPreplay [31] are utilized for topology discovery attacks. In this study, corresponding scenarios are designed for topology discovery via injection, flooding, and man-in-the-middle attacks.

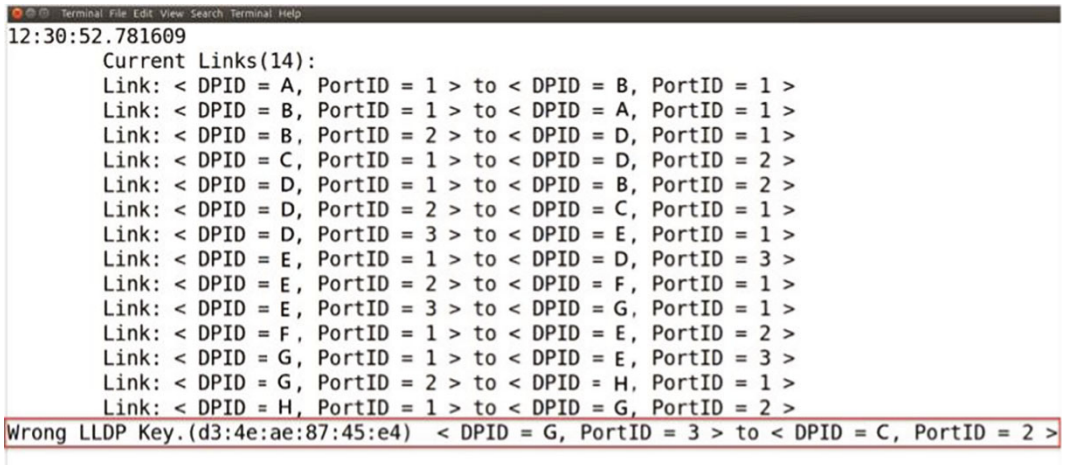

Figure 14: Topology discovery via injection attack. The fabricated LLDP packet has a wrong key.

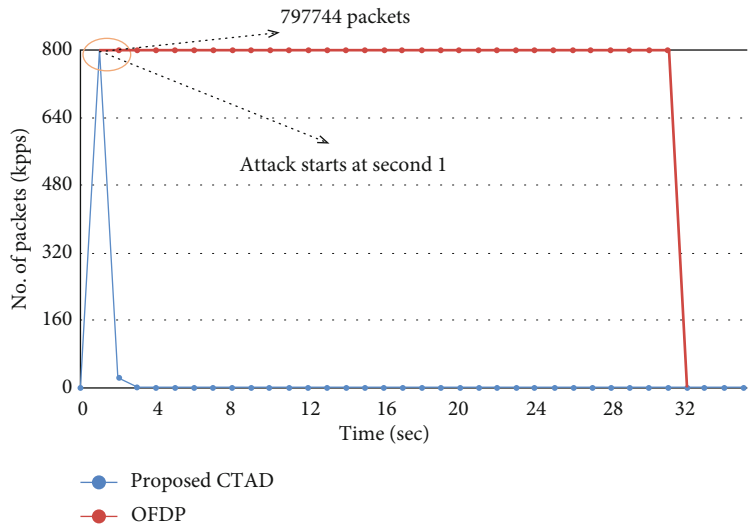

Figure 15: Topology discovery via flooding attack. The proposed mechanism can reduce the amount of attack packets.

4.1. Topology Discovery via Injection Attack Scenario. To simulate the topology discovery via injection attack, the LLDP packet is forged, the chassis ID and port ID are filled in the LLDPDU into Switch G and Port 3, and the fake LLDP packet is sent to Port 2 of Switch C. Switch C, which received the packet, is then transmitted to the SDN controller through a Packet-In message, and an attempt is made to forge a (DPID: G, portID: 3→DPID: C, portID: 2) link, as shown in Figure 13. The topology is shown in Table 3.

However, when the topology manager extracts the verification key from the LLDP packet that does not match the verification key of the current topology discovery period, this scenario indicates that the LLDP packet is fabricated by the attacker. Therefore, the fabricated LLDP packet will be discarded, as shown in Figure 14.

4.2. Topology Discovery via Flooding Attack Scenario. TCPreplay is used to generate numerous LLDP packets and conduct a 30-second attack for the simulation of topology discovery via flooding attack. The results show that compared with OFDP, after each topology discovery cycle ends, the LLDP monitor calculates the number of LLDP packets received on each port. Once it finds that a device connected to any port sends numerous fake LLDP packets, the port number will be sent to the LLDP floodgate for blocking. Thus, the topology discovery via flooding attack can be blocked, as shown in Figure 15.

4.3. Topology Discovery via Man-in-the-Middle Attack Scenario. In this scenario, a simulated attacker launches a topology discovery via man-in-the-middle attack, and the time difference and correlation analysis of the proposed CTAD mechanism will be discussed.

4.3.1. Latency of LLDP. This experiment compares the proposed CTAD mechanism with KHMAC [21] and OFDP [3]. Figure 16(a) shows the average RTT of LLDP packets under the network scale of 20 switches using CTAD,

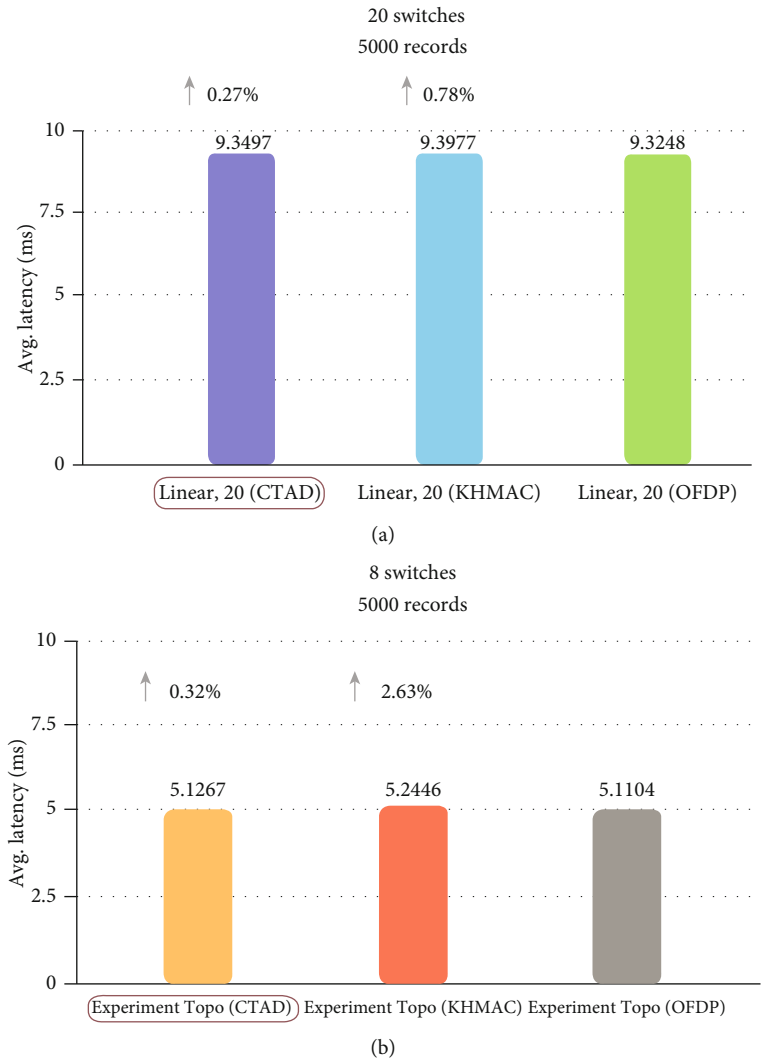

Figure 16: RTT of LLDP packets: (a) 20 switches; (b) 8 switches.

KHMAC, and OFDP. The CTAD adds authentication information and timestamps to the LLDP packets. The KHMAC only adds authentication information. Both will increase the overhead of the LLDP RTT due to the increase in LLDP length. Figure 16(b) shows the average RTT of LLDP packets under the network scale of 8 switches using CTAD, KHMAC, and OFDP. The proposed CTAD mechanism only increases the time by 0.32% compared with OFDP, whereas KHMAC increases the time by 2.63%.

4.3.2. Distribution Analysis of Latency. This experiment analyzes the RTT of LLDP packets. In this experiment, the SDN controller collects the data returned by the 5, 8, 31, and 127 virtual switches. Figure 17(a) illustrates that for the network scale of only 5 switches, the overall RTT distribution of LLDP packets is normally distributed. The distribution of the RTT of LLDP packets determines which outlier calculation method to be used for the time difference analysis. Common methods include the Z-score [33] and the box-and-whisker plot. The Z-score assumes that the data conforms to the normal distribution; thus, 99.7% of the data will be within the range of the maternal mean plus or minus 3 maternal standard deviations. Therefore, the data greater than the maternal mean of 3 maternal standard deviations will be regarded as outliers. The box-and-whisker plot does not assume that the data must conform to the normal distribution. It calculates the maximum, median, first to third quartiles, and IQR of the data to obtain the upper and lower bounds. The data outside the boundary will be regarded as outliers.

Figure 17(a) illustrates that the Z-score and box-and-whisker plot can be used. Figures 17(b)–7(d) indicate that the distribution of the data gradually becomes skewed and does not conform to the normal distribution; thus, it is not suitable to use the Z-score. Therefore, the time difference analysis proposed in this study uses the box-and-whisker plot to determine outliers.

4.4. CTAD Detection Rate. To simulate the topology discovery via man-in-the-middle attack, Attacker 1 (H11) taps and records the LLDP packet from the SDN controller at Port 2 of Switch C and then encapsulates the LLDP packet and sends it to Attacker 2 of Port 3 of Switch G (H31). Attacker

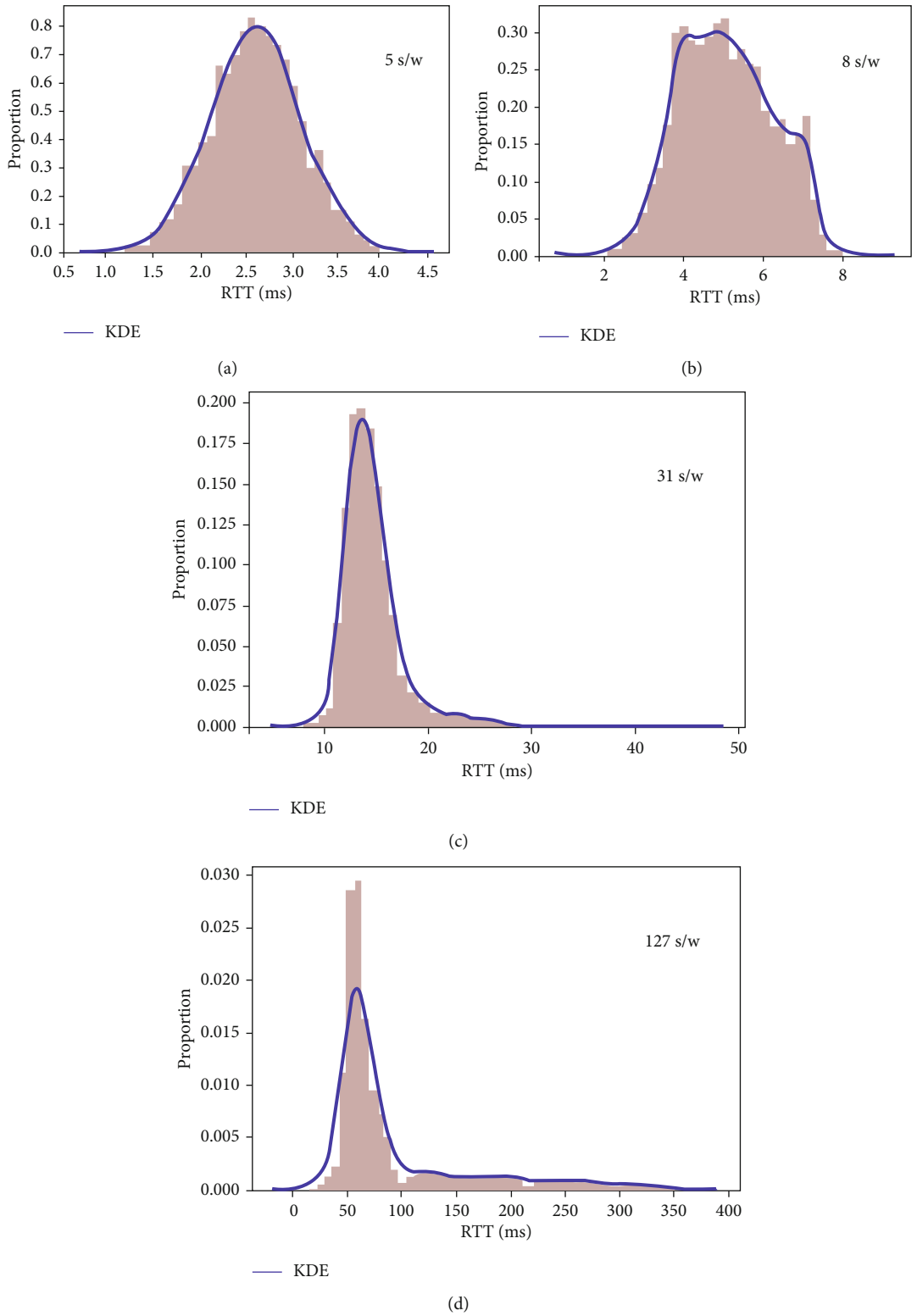

Figure 17: Overall RTT distribution: (a) 5 switches; (b) 8 switches; (c) 31 switches; (d) 127 switches.

2 (H31) changes back to the normal packet type and attempts to forge a (DPID: C, portID: 2→DPID: G, portID: 3) link, as shown in Figure 18.

To verify the topology discovery via man-in-the-middle attacks with high-speed replay, replay LLDP is generated with different delay times and compare them with SALL [22] and KHMAC [21]. The results show that the correlation analyzer can detect a topology discovery via man-in-the-middle attack that replays LLDP packets at high speeds as low as 40 ms (Figure 19).

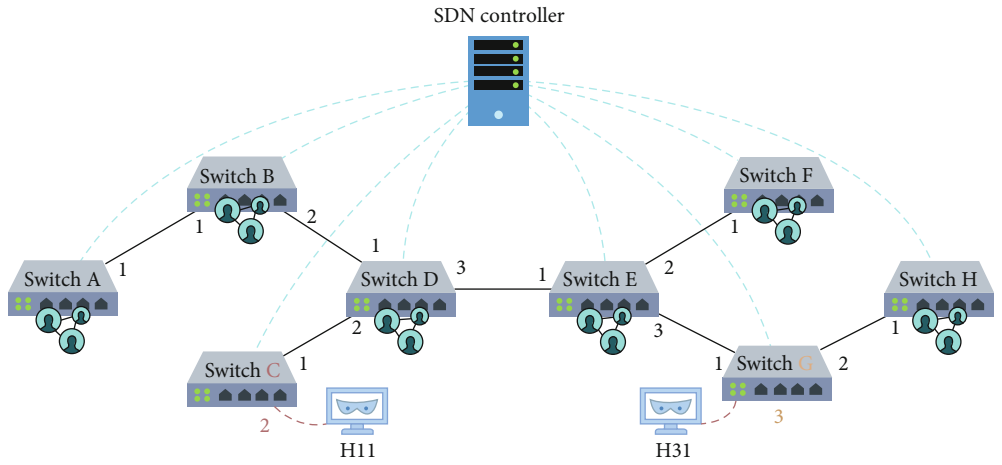

Figure 18: Topology discovery via man-in-the-middle attack. The attacker attempts to forge a (DPID: C, portID: 2→DPID: G, portID: 3) link.

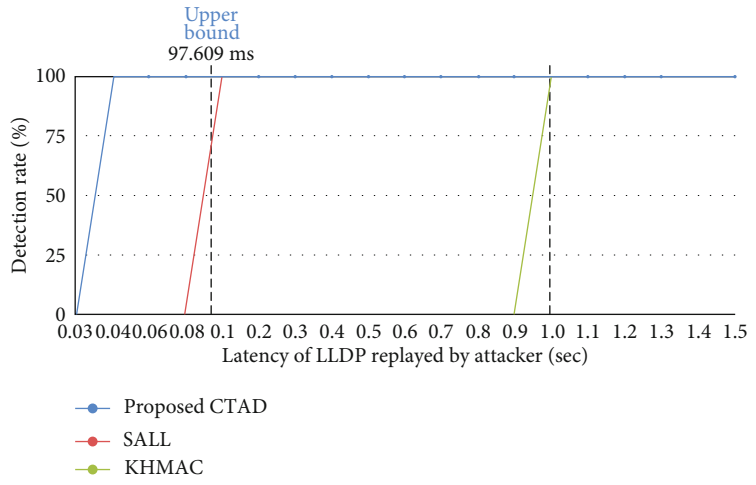

Figure 19: Comparison of topology discovery via man-in-the-middle attack detection rates.

5. Conclusions

This study adds verification information and timestamps to LLDP packets, detects each LLDP packet that is returned to the SDN controller, and successfully detects topology discovery via injection and man-in-the-middle attacks. It also calculates the amount of LLDP packets received on each port during each topology discovery cycle, which reduces the effect of topology discovery via flood attacks on the SDN controller. Spearman's rank correlation is used to analyze the high-speed and highly automated topology discovery via man-in-the-middle attacks and maintain correct topology information of the SDN controller.

The results show that compared with the KHMAC detection mechanism that uses only authentication information and the SALL detection mechanism that uses only timestamps, the proposed mechanism has a wider scope in detecting topology discovery via man-in-the-middle attacks.

Data Availability

The data used to support the findings of this study are included within the article.

Disclosure

This paper is a revised version of a paper originally presented at the 10th International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, October 2019.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

Acknowledgments

The work described in this paper was supported by the National Chung-Shan Institute of Science & Technology under Grant NCSIST-ADV-V101 (108) and the Ministry of Science and Technology of the Republic of China under Grant 108-2221-E-008-033-MY3.

References

[1] "Software-defined networking – Wikipedia," https://en.wikipedia.org/wiki/Software-defined_networking.

[2] "Link layer discovery protocol - Wikipedia," https://en.wikipedia.org/wiki/Link_Layer_Discovery_Protocol.

[3] "OpenFlowDiscoveryProtocol - GENI: geni," https://groups.geni.net/geni/wiki/OpenFlowDiscoveryProtocol.

[4] Ryu SDN frameworkhttps://ryu-sdn.org/.

[5] "Project Floodlight - GitHub," https://github.com/floodlight.

[6] "Home - OpenDaylight," https://www.opendaylight.org/.

[7] "GitHub - brandonheller/riplpox: RipL-POX (Ripcord-Lite for POX): a simple network controller for OpenFlow-based data centers," https://github.com/brandonheller/riplpox.

[8] M. J. Allen and W. M. Yen, *Introduction to Measurement Theory*, Waveland Press, Long Grove, IL, 2001.

[9] "P4, [Online]," https://p4.org/.

[10] "Stratum – open networking foundation," https://www.opennetworking.org/stratum/.

[11] "Open network operating system (ONOS) SDN controller for SDN_NFV solutions," https://onosproject.org/.

[12] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[13] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of SDN/OpenFlow controllers," in *9th Central & Eastern European Software Engineering Conference in Russia*, pp. 24-25, Moscow, Russia, 2013.

[14] A. Azzouni, O. Braham, T. M. T. Nguyen, G. Pujolle, and R. Boutaba, "Fingerprinting OpenFlow Controllerscontrollers: The the First first Step step to Attack attack an SDN Control control Planeplane," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 4–8, Washington, DC, USA, 2016.

[15] H. Zhang, Z. Cai, Q. Liu, Q. Xiao, Y. Li, and C. F. Cheang, "A survey on security-aware measurement in SDN," *Security and Communication Networks*, vol. 2018, Article ID 2459154, 14 pages, 2018.

[16] H. Zimmermann, "OSI reference Model–The ISO model of architecture for open systems interconnection," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, 1980.

[17] V. Z. Attar and P. Chandwadkar, "Network discovery protocol lldp and lldp-med," *International Journal of Computer Applications*, vol. 1, no. 9, pp. 99–103, 2010.

[18] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 151-152, Hong Kong, China, August 2013.

[19] T. H. Nguyen and M. Yoo, "Analysis of link discovery service attacks in SDN controller," in *2017 International Conference on Information Networking (ICOIN)*, pp. 11–13, Da Nang, Vietnam, January 2017.

[20] S. Hong, L. Xu, H. Wang, and G. Gu, *Poisoning network visibility in software-defined networks: new attacks and countermeasures*, National Down Syndrome Society, 2015.

[21] T. Alharbi, M. Portmann, and F. Pakzad, "The (in) security of topology discovery in software defined networks," in *Proceedings of 2015 IEEE 40th Conference on Local Computer Networks (LCN)*, pp. 26–29, Clearwater Beach, FL, USA, Oct. 2015.

[22] D. Smyth, S. McSweeney, D. O'Shea, and V. Cionca, "Detecting link fabrication attacks in software-defined networks," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–8, Vancouver, BC, Canada, August 2017.

[23] A. Azzouni, N. T. M. Trang, R. Boutaba, and G. Pujolle, "sOFTDP: secure and efficient topology discovery protocol for SDN," 2017, http://arxiv.org/abs/1705.04527.

[24] A. Nehra, M. Tripathi, M. S. Gaur, R. B. Battula, and C. Lal, "SLDP: a secure and lightweight link discovery protocol for software defined networking," *Computer Networks*, vol. 150, pp. 102–116, 2019.

[25] R. Taylor, "Interpretation of the correlation coefficient: a basic review," *Journal of Diagnostic Medical Sonography*, vol. 6, no. 1, pp. 35–39, 2016.

[26] L. Myers and M. J. Sirois, "Spearman correlation coefficients, differences between," *Encyclopedia of Statistical Sciences Sonography*, vol. 12, 2006.

[27] J. Laurikkala, M. Juhola, E. Kentala, N. Lavrac, S. Miksch, and B. Kavsek, "Informal identification of outliers in medical data," in *Fifth international workshop on intelligent data analysis in medicine and pharmacology*, Berlin, Germany, August 2000.

[28] D. J. Hamad, K. G. Yalda, and I. T. Okumus, "Getting traffic statistics from network devices in an SDN environment using OpenFlow," in *Proceedings of 2015 Information Technology and Systems (ITaS)*, pp. 951–956, Olympic Village, Sochi, Russia, 2015.

[29] A. Nicolae, L. Gheorghe, M. Carabas, N. Tapus, and C. L. Duta, "LLDP packet generator," in *2015 14th RoEduNet International Conference - Networking in Education and Research (RoEduNet NER)*, pp. 24–26, Craiova, Romania, September 2015.

[30] "GitHub - the-tcpdump-group/tcpdump: the TCPdump network dissector," https://github.com/the-tcpdump-group/tcpdump.

[31] "GitHub - appneta/tcpreplay:Pcap editing and replay tools for ∗NIX and Windows - Users please download source from," https://github.com/brandonheller/riplpox.