A Systematic Treat Model for Software-Defined Networking

Wenbin Zhang, Zehui Wu*, Qiang Wei, and Huijie Yuan
China National Digital Switching System Engineering and Technological Research Center
Zhengzhou, Henan 450001, China
[e-mail: wenbinzhang@buaa.edu.cn, wuzehui2010@foxmail.com,
prof_weiqiang@163.com, j15290616907@gmail.com]
* Corresponding author: Zehui Wu

Received August 25, 2020; revised January 17, 2021; accepted February 3, 2021; published February 28, 2021

Abstract

Software-Defined Networking (SDN) has three key features: separation of control and forwarding, centralized control, and network programmability. While improving network management flexibility, SDN has many security issues. This paper systemizes the security threats of SDN using spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege (STRIDE) model to understand the current security status of SDN. First, we introduce the network architecture and data flow of SDN. Second, we analyze security threats of the six types given in the STRIDE model, aiming to reveal the vulnerability mechanisms and assess the attack surface. Then, we briefly describe the corresponding defense technologies. Finally, we summarize the work of this paper and discuss the trends of SDN security research.

Keywords: Network Security, Software-Defined Networking, Security Threats, STRIDE Model

This work was supported by the National Key Research and Development Program of China under Grant 2016QY07X1404, 2019QY0501.

http://doi.org/10.3837/tiis.2021.02.011 **ISSN : 1976-7277**

1. Introduction

Software-Defined Networking is a logically centralized network system that can effectively address the shortcomings of traditional networks in terms of reliability, flexibility, and scalability [1]. SDN has three key features [2]: (1) The control logic and the data forwarding process are separated into two planes. (2) The control logic is centralized to the controller, which is equivalent to a network operating system. (3) A developer can use the programming interface provided by the controller to develop a variety of applications to achieve different network functions. Based on these three characteristics, network administrators can perform better network resource allocation, network management and network configuration [3], achieving the efficient management of large-scale communications and allowing more flexible solutions for performing complex tasks [4]. At present, SDN has been used in data centres, local area networks, and cloud computing [5]. Internet giants such as Google and Facebook have carried out research and actual deployment of SDN [6]. Google deployed SDN in the data centre, improving the link utilization rate to almost 100% [7].

Compared with traditional networks, with control logic and forwarding tightly conjoined in one device, the most fundamental feature of SDN is the abstraction of the underlying network equipment, which covers the diversity of the hardware and gives an overall view of the entire network to upper layers. The overall network abstraction makes programmability possible, allowing administrators to manage complex traffic centrally without having to manually configure network flow processing policies for each network device.

However, SDN also has security risks. Centralized control leads to centralized vulnerabilities, and programmability reduces the difficulty of attacks. Kreutz et al. [8] revealed following seven possible vulnerabilities of SDN: the traffic can be forged, switches are unprotected, controllers are vulnerable, applications are untrusted, communications between different planes can be hijacked, administrative stations can be attacked, and forensics and remediation are difficult. This shows that every part of the SDN architecture has the possibility of being attacked.

Through investigating the basic architecture and workflow of SDN, this paper draws on the STRIDE threat model to summarize the security threats faced by SDN in the areas of spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privilege, sorting out the solution ideas and providing a reference for security research on SDN networks. In more detail, the main contributions of this paper are listed below:

(1) A taxonomy that classifies the vulnerabilities and attack methods proposed by reviewed articles in six main categories, covering the possible security risks faced by SDN and constructing a comprehensive threat model of SDN.

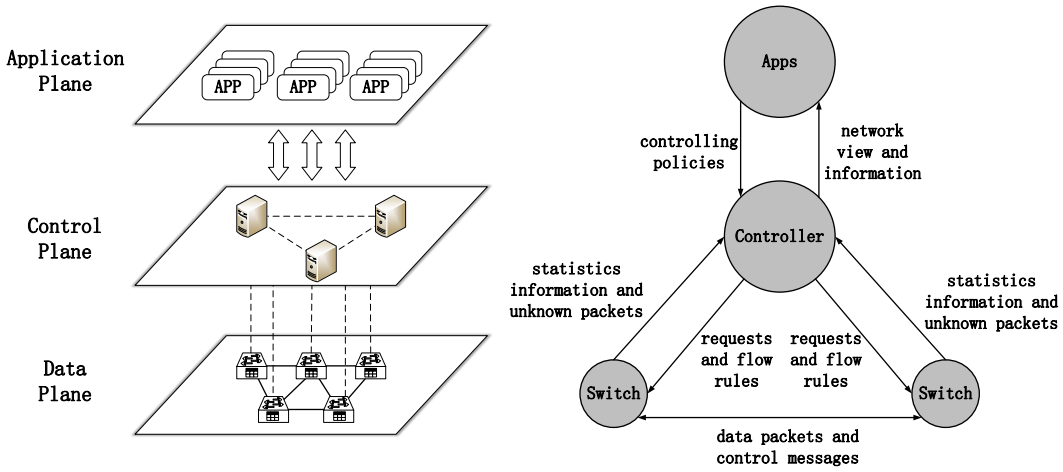
(2) We introduce the mechanism of the vulnerabilities or threats in detail and point out the progressive relationship or complementary relationship between different attack methods, revealing the evolution of attack methods. It also indicates that the complexity of SDN can lead to many unexpected attack points.

(3) A summary of the emerging challenges in SDN security to suggest the directions for future research efforts and proposals.

The paper is organized as follows: In section 2, we analyse the basic architecture and data flow of SDN. In section 3, we summarize the security threats faced by SDN. In section 4, we introduce attack methods of each threat type. In section 5, the main solutions for SDN security issues are discussed. In section 6, we introduce some related work. Finally, we summarize the full paper and provide future research directions for SDN security solutions.

2. SDN Architecture

The Open Networking Foundation [9] divides SDN into three layers, which are, from bottom to top, the data plane, control plane and application plane, as shown in **Fig. 1(a)**. The northbound channel and the southbound channel connect the application plane and the data plane to the control plane respectively, and transmit data bidirectionally. Currently, no standard communication protocol for the northbound channel is constituted, and the most widely used southbound channel communication protocol is OpenFlow [10]. OpenFlow protocol specifies three message types, controller-to-switch, asynchronous, and symmetric to support controllers and switches to deliver management instructions or data packets. Controller-to-switch messages include PACKET_OUT, FLOW_MOD, SET_CONFIG, etc. Asynchronous messages include PACKET_IN, PORT_STATUS, FLOW_REMOVED, etc. Symmetric messages include HELLO, ECHO_REQUEST, ECHO_REPLY, etc.



(a) SDN Architecture (b) SDN Data Flow

Fig. 1. SDN Architecture and Data Flow

The data plane consists of interconnecting forwarding devices, mainly switches. The switches used by SDN support southbound protocols such as OpenFlow, which usually have no self-learning capabilities and are only responsible for data forwarding. Each switch consists of the following components: ports for data packet receiving and sending, flow tables, and a communication interface with the controller. The flow tables store flow rules, according to which the switch processes the received data packets.

The control plane is the core of SDN, consisting of at least one controller. Through interacting with the data plane using the southbound interface, the controller obtains a dynamic view of the topology and switch information. And through the open programming interface, the application plane can request the network status from the control plane to implement different network functions. In a multi-controller system, different controllers perform state synchronization and data exchange through the east-west interfaces. Well-known controllers include FloodLight [11], OpenDaylight [12], ONOS [13], Ryu[14], NOX [15], etc.

The application plane consists of applications with different network functions, which is a concentrated expression of the programmability of SDN. Using the network status information obtained from the control plane, the applications may perform routing, load balancing, firewalls, quality of service (QoS) etc, and translate the network management strategy into flow rules, then install them into the switches through the controller.

The data flow between the three planes is shown in **Fig. 1(b)**. Between data plane switches, data flow consists of packets and control information. The controller communicates with switches through the southbound interface to inquire about their packet statistics and installs flow rules. Applications request network information from the controller, generate network policies, and send them to the controller.

3. SDN Security Overview

The layered architecture and centralized control of SDN provide a new attack surface. Each layer of the network has certain vulnerabilities, and an attack on one layer will affect the entire network. Therefore, this paper classifies known attacks according to their effects. **Table 1** summarizes the possible attack methods and impact scope of various threat types.

Table 1. Overview of SDN Threats

Threat type	Attack methods	Attack location	Impact scope
Spoofing	Spoofing as a legitimate switch	Data plane	Data plane Control plane Application plane
	ARP spoofing	Data plane	
	IP hijacking	Data plane	
Tampering	Exploiting controller storage component vulnerability	Control plane	Data plane Control plane Application plane
	Tampering with controller topology data	Data plane	
	Man-in-the-middle attack	Data plane	
	Attacking switch flow table	Data plane	
Repudiation	Deleting log files	Control plane	Data plane Control plane
	Covert communication channel	Data plane	
Information disclosure	SDN network presence probing	Data plane	Data plane Control plane Application plane
	SDN controller type probing	Data plane	
	Network flow rule probing	Data plane	
	Application probing	Data plane	
Denial of Service	PACKET_IN message flooding attack	Data plane	Data plane Control plane Application plane
	Virtual host attack	Data plane	
	Shared-path attack	Data plane	
	Malicious application attacks	Application plane	
Elevation of privilege	Cross application privilege escalation	Application plane	Application plane
	Data Plane Access Control Bypass	Data plane	Data plane

4. SDN Security Threat Analysis

4.1 Spoofing

Spoofing refers to a process whereby an attacker obtains the trust of the target network through deception or other means. It is usually located at the front end of the attack chain and prepares for subsequent attacks. Identification is the basis of network communication. In a network, entities are bound with identifiers, such as IP address and MAC address. These identifiers play an important role in network management and communication, and many network security policies are also implemented based on these identifiers. Therefore, there are many network counterfeiting attacks against identification, including domain name system (DNS) spoofing

[16], address resolution protocol (ARP) spoofing [17], dynamic host configuration protocol (DHCP) forgery, and host location hijacking [18]. There are two reasons why these attacks can succeed: The first is that protocols based on broadcast technology do not perform identity authentication, making it difficult to ensure security. The second is that the identifier can be changed manually and is not unique.

In SDN, spoofing threats also exist. An attacker can impersonate a switch or perform ARP hijacking and IP hijacking to impersonate a legitimate host.

4.1.1 Spoofing as a Legitimate Switch

A switch in SDN uses a Datapath ID (DPID) and MAC address as its identifiers. When connecting to the controller, a switch will firstly send its own DPID to the controller through a HELLO message without authentication. Therefore, any switch can easily create a new connection with the controller using the DPID of another existing switch. Because of the vulnerability of the connection management module, the controller will disconnect from the legitimate switch and accept a counterfeiter, who successfully poses as a legitimate switch in the network [19].

4.1.2 ARP Spoofing

As a connectionless protocol, the ARP protocol has no identity authentication mechanism. Any host can send a forged ARP request or response packet to the network, and a host that receives the ARP packet cannot judge the legitimacy of the sender. Because host communications in SDN networks still follow the TCP/IP protocol, SDN also suffers ARP attacks. An attacker can personate another host by sending an ARP packet with a fake source IP or source MAC [20]. Even worse, the SDN controller maintains the IP-MAC-location mapping of every host by analysing the ARP packets in PACKET_IN messages, so this type of attack affects all topology-based applications.

As shown in **Fig. 2**, the host H1 is an attacker whose MAC address and IP address MAC1 and IP1 respectively. Another host H2 is the victim whose MAC address and IP address are MAC2 and IP2 respectively. H1 sends a fake ARP packet using the IP address of H2. When receiving the ARP packet, switch S1 has no matching flow rules, so it sends the packet encapsulated into a PACKET_IN message to the controller. The controller learns the information of the source IP address from the message and updates the host configuration entry to "MAC1-IP2-Location1", which affects the normal communication of host H2.

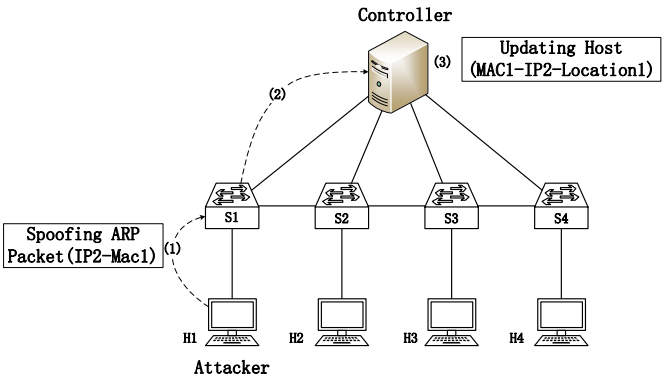


Fig. 2. ARP Spoofing in SDN

4.1.3 IP Hijacking

ARP spoofing attacks can cause great harm to a network, but the impact of the attack is limited in duration, and the attack becomes invalid when the ARP cache entry times out. Another method for counterfeiting legitimate hosts in the network is to hijack the IP of the target host and consolidate the attack effect by preventing the victim from claiming ownership of the IP [21]. The specific steps are as follows:

First, induce the DHCP server to release the IP address of H2 to the available address pool by sending a DHCP_RELEASE message.

Then, request the DHCP server to assign the IP address of H2 to the attacker by releasing numerous DHCP_DISCOVER messages.

Finally, exploit the competition condition of the SDN to prevent the victim from replying to the ARP message that the DHCP server uses to confirm that the IP address is not being used. The effect of this attack will last hours or days until the victim acquires a new DHCP lease after knowing that its IP address was stolen.

Experiments on ONOS 1.5.1 demonstrate that the malicious host can convince the network devices that it is the legitimate owner of the victim's identifiers, allowing it to persistently hold the compromised identifiers.

4.2 Tampering

Tampering refers to the illegal modification of data, which can destroy the integrity of the network system. The key data of SDN include the network information data maintained by the controller, the communication data in the northbound and southbound channel and the switch flow table of the data plane [22]. All these data may become a target for attackers to tamper with. For example, controllers such as OpenDayLight and ONOS use network management data storage architecture (NMDA) as the basic storage component. [23] found that a semantic gap problem in NMDA may lead to a risk of attackers tampering with the data in the controller. Spoofing and tampering are often intertwined and interdependent because deceiving network elements and controllers are the main motivations for tampering.

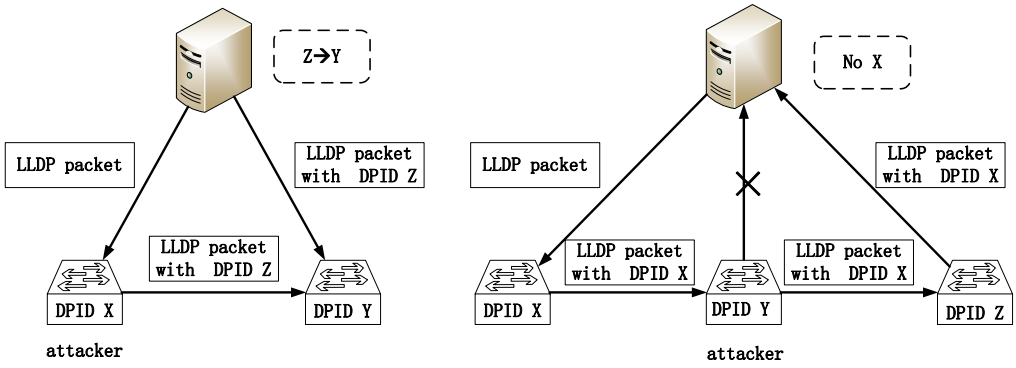
4.2.1 Tampering with Controller Topology Data

In SDN, topology management includes host discovery, switch discovery, and internal link (that is, a link between switches) discovery [18]. If the attacker controls the hosts and switches in the network, he can falsify the topology data in the controller through deception [24, 25].

In the stage of establishing a connection between the switches and the controller, each switch sends its information to controller using FEATURES_REPLY message as a response to the FEATURES_REQUEST message of the controller. In this process, switch discovery is completed. Attackers can use switch spoofing to contaminate the switch data of the controller.

The host tracking service of the controller is responsible for host discovery and tracking host locations. The controller establishes a configuration file for the host by analysing the information in the PACKET_IN message and stores its MAC, IP, VLAN, and location information. The location information of the host is usually the switch port to which the host is connected. When the host location changes, the host tracking service updates the host configuration file. Therefore, the attacker can first obtain an identifier of a legitimate target host by sending ARP probe packets and then use the identifier to create a data packet and send it to the network. When the controller receives the data packet, it will consider that the host has been transferred to a different location and therefore modify the host configuration file.

The attacker tampers with the host data of the controller through this method, which can achieve the purpose of hijacking the data flow of the target host. The controller uses the Link Layer Discovery Protocol (LLDP) to discover links between switches. The attacker has two ways to tamper with the controller topology data through attacking link discovery service. Assuming that the attacker controls a malicious switch, he can construct a fake LLDP packet containing a specific DPID and port field and then broadcast it to an adjacent switch, which allows the controller to construct a link that does not actually exist, as shown in **Fig. 3(a)**. Another method is that the attacker chooses not to report to the controller when receiving the LLDP data packet sent by the neighbour switch but forwards the data packet to the neighbour switch, which can hide his existence in the network, as shown in **Fig. 3(b)**.



(a) Make a fake link (b) Hide the attacker

Fig. 3. Attacking a Link Discovery Service

4.2.2 Man-in-the-Middle Attack

Man-in-the-middle attack is a typical data monitoring and tampering attack. The attacker places herself between two network nodes through deception or another means and can then monitor, tamper, or terminate communication between the nodes without the two nodes being aware of it. The attacker can tamper with the control instructions and data information transmitted in the control channel and data channel [18, 22, 26].

Attackers can implement man-in-the-middle attacks through ARP spoofing. Supposing there are three hosts—hosts H1 and H2 and the attacker—the attacker can execute ARP spoofing attacks on the other two hosts, mapping her MAC addresses to the IP address of host H1 in the ARP cache of host H2 and vice-versa. After that, the attacker can control all communications between hosts H1 and H2, while hosts H1 and H2 still think that there exists a direct connection between them. Attackers can also implement man-in-the-middle attacks by tampering with topology data. When the controller performs link discovery, the attacker relays the LLDP message between switches A and B to make the controller believe that switches A and B are directly connected, as described in the previous subsubsection. The controller ignores the existence of the attacker when planning the data flow path, and the data passing through the link between A and B will be intercepted or tampered with by the attacker.

4.2.3 Attacking the Switch Flow Table

For switches, flow tables and stored flow rules are the most important data. To achieve high search performance, ternary content addressable memory (TCAM) is used to store flow rules. However, the storage space of TCAM is limited and easy to be overflowed. By sending a large number

of random data packets, an attacker can add quite a few invalid flow rules into the flow table in a short time, resulting in the flow rules being overwritten or the flow table overflowing [27-29]. However, a high-rate random data packet flow is easily recognized by the network defence system, so researchers have improved the effectiveness of overflowing the flow table in terms of reducing the attack packet generation rate and improving the attack accuracy. [30] proposed a low-rate flow table overflow attack that first infers exactly which fields of the packet header can trigger the installation of new flow rules and then constructs specific data packets based on the inference results. The efficiency of triggering the flow rule installation can be significantly improved.

[31] studied the logic of how a controller installs flow rules. By probing the cycle and statistical indicators of the controller's network status statistical query, the attacker can send carefully designed data packets at specific time intervals, inducing the controller to send too many FLOW_MOD messages and overflowing the switch flow table. Evaluations on physical OpenFlow Switch (Pica8 P3290) and FloodLight show that the reflection attacks can cause severe effects with acceptable expenses.

4.3 Repudiation

Non-repudiation means that entities in the system cannot deny operations or events that have been performed, so that when malicious behaviour is discovered, it can be accurately traced back to the perpetrator. To achieve undeniability, the basic approach is to keep the audit logs necessary to record every operation of the entities in the system so that it is possible to trace the source after an attack event.

Repudiation is more difficult to prevent in an SDN because the high complexity of the control plane makes it difficult to trace events [32, 33]. For example, when many applications are running simultaneously, determining which application implemented a certain behaviour is difficult. In addition, for the reactive and event-driven controllers, the source of malicious behaviour is difficult to pinpoint precisely [34]. In the data plane, OpenFlow stipulates that the switch reports all changes of the flow table and port status to the controller, but a malicious switch can choose not to report its own status changes. In addition, when multiple controllers manage the network together, it is difficult to determine which one executed a specific instruction in a large number of asynchronous messages. Even if all connections are authenticated, operations cannot be locked to a single participant.

Another repudiation threat is that the operation trace can be eliminated by tampering with the log file in the controller resource database. [35] developed a malicious Trojan program for the OpenDayLight controller. This program can use the Java reflection mechanism to delete objects added during its installation process and replace the OpenDayLight service registration centre. Ultimately, controller services can be used by the program to install malicious flow rules while hiding the existence of the program from the controller.

On the data plane, [36] proposed a remote transmission scheme that can carry out covert communication between two malicious switches. The two switches connect to the controller using an ahead negotiated DPID, and each switch judges whether the other switch is accepted by the controller or not based on the controller's response, thereby achieving single-bit data transmission, which can be extended to the transmission of data frames. Because the malicious switches do not seem to have performed any abnormal operations, this covert channel and communication behaviour is difficult to discover and track. Somewhat differently, [37] proposed a two-step worm-hole attack to build a covert channel between two switches. A fake link is firstly built using the method of [18], and then packets are delivered using a well-selected relay host. Without inserting any actions and tags to the packets in the attacking

process, the invisibility is improved.

4.4 Information Disclosure

In SDN, there are two common types of information disclosure. On the one hand, user traffic can be monitored and intercepted, resulting in critical data being obtained by the attacker. On the other hand, the attacker can investigate the target network through network detection in preparation for a larger-scale attack (usually a DoS).

In terms of data interception, OpenFlow faces the same vulnerabilities as traditional networks. If the communication is not encrypted, an attacker can sniff passwords and confidential data through man-in-the-middle attacks. This problem can be solved using authentication and encryption. Although the OpenFlow protocol supports transport layer security (TLS) protocol, it is rarely adopted due to its low performance [38]. Because we have introduced how attackers implement man-in-the-middle attack in section 4.2, we mainly talk about network detection here.

Network detection for an SDN includes SDN network presence detection, controller type detection, flow rule detection, and application detection.

4.4.1 SDN Network Presence Probing

When attackers carry out an attack, they first need to understand the type of the target network. In an SDN, the time needed for a switch to process different data packets varies. If the data packet is unfamiliar, the switch needs to request the flow rule from the controller and install it. If a flow rule already exists, the switch directly processes the data packet according to the flow rule. Assume that the response time when there is no matching rule for the data packet is T1, and the response time when there is an entry is T2; T1 will be greater than T2. There is no such obvious difference in traditional networks. If the difference between T1 and T2 can be distinguished with high accuracy, attacker can determine whether the network is an SDN network. Based on this principle, [39] developed SDN-Scanner, a scanning tool to recognize remote networks and determine the existence of SDN networks.

4.4.2 SDN Controller Type Probing

Determining the type of controller that manages the target network is an important step in launching an effective attack on SDN network. Attackers can use the known vulnerabilities of such controllers to carry out specific attacks. [40] proposed two techniques for fingerprinting SDN networks, which can detect the type of network controller in the data plane.

(1) Controller fingerprinting based on time analysis

Controller fingerprinting based on time analysis infers the controller used by the SDN network by measuring time parameters, such as the timeout value of the flow table entry and the time to process data packet by the controller.

Different controllers will install flow rules with different idle timeout and hard timeout, indicating the time before an entry without packet matching is deleted from the switch and the time before an entry will be deleted in any case. Attackers can identify the controller by firstly inferring the timeout values of flow rules and then comparing them with the known timeout values of existing SDN controllers. Different controllers use different tools, libraries and frameworks for design and implementation, with different execution speeds, and thus spend distinct time processing packets and replying to data plane. Therefore, the controller can be determined by estimating the packet processing time. Similar to the method based on the flow rules timeout, the attacker first measures the response time of the target controller and then

compares it with the known processing time values to identify it.
(2) Controller fingerprinting based on packet analysis

In SDN network, the controller processes data packets to discover the network topology and guide the data forwarding process of switches. For example, the controller uses LLDP packets to build the topology of the entire network. The contents of the LLDP packets used by different controllers are different, so the controller can be distinguished by examining the contents of the LLDP packets.

4.4.3 Network Flow Rule Probing

In SDN network, a flow describes the data transmitted from one network node to another node. Each flow has a corresponding rule to describe the forwarding action that the forwarding device should take for the data belonging to the flow. SDN uses flow rules to formulate performance and security policies such as load balancing and access controls. Therefore, if the composition of the flow rules in the network is known, the flow rules can be triggered or evaded to carry out specific attacks. In some known attacks [30, 31], flow rule detection is taken as the preparatory stage of the attack.

Round-trip time (RTT) is widely used to prob flow rules. Compared with packets with flow rules matching, those without matching flow rules will be processed by the controller and have a longer forwarding delay, and the load of the controller will increase. Therefore, the flow rule configuration can be inferred by constructing a probe packet and analysing the difference of the RTTs. By changing the packet field and comparing the RTTs before and after the change, whether a changed field triggered the installation of flow rules can be determined. If the changed field triggers a flow rule installation, controller is probably sensitive to this header field. Further, [41] uses two streams, the timing packet stream and test packet stream, that cooperate with each other to infer the values of the matching fields. To improve the detection accuracy, [42] developed an explicit inference algorithm to detect the cache size, policy and state of the target flow table in a more fine-grained way and implement intelligent DoS attacks according to the inferred parameters, causing greater damage with less cost.

4.4.4 Application Probing

In SDN networks, applications provide multiform network management functions, such as load balancing, firewalls, intrusion detection, and security forensics. For cyber attackers, it is of great significance to know which applications are running on the network, especially for understanding the existence of defensive programs.

Different SDN applications call different controller application programming interfaces (APIs) to construct their control logic, and different APIs have different behaviors, resulting in different control traffic patterns. Therefore, applications can be fingerprinted by analyzing the control traffic. Since deep learning can automatically extract features from massive data, it can be used to extract the control flow characteristics of different applications to identify them. [43] trained 10 binary classifiers for 10 applications, each of which could be used to determine if a captured packet sequence fits the control flow mode of a specific application. The model training process is shown in **Fig. 4**:

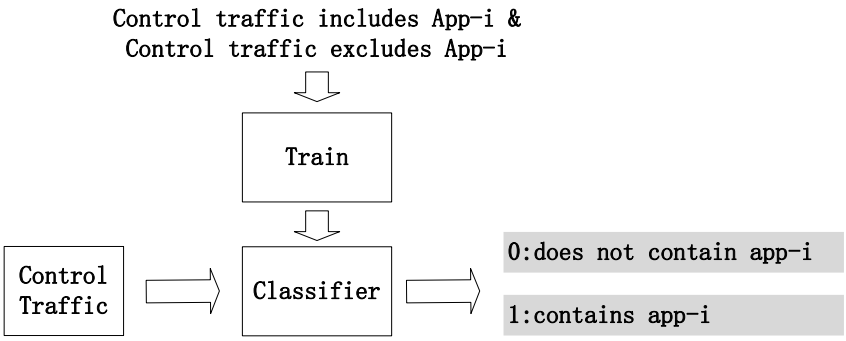


Fig. 4. Training the Application Classifier

Experiments on a real SDN testbed with five commercial hardware SDN switches (Edgecore AS4610-54T) and a Floodlight controller show that an attacker can precisely fingerprint SDN applications.

4.5 Denial of Service

DoS attacks are the most common threats to SDN, and attackers can implement DoS attack through either southbound channel or northbound channel.

4.5.1 PACKET_IN Message Flooding Attack

DoS attacks to SDN usually originate from the data plane and affect the availability of controllers by populating the switch flow table or filling the southbound channel between the controller and data plane [36, 41, 44]. A DoS attack against an SDN controller is a serious security threat, which may cause the controller to be unavailable and jeopardize network stability. PACKET_IN message flooding attack is a basic DoS attack method [23, 24].

An attacker can control a host to generate many data packets with all or some fields set to random values. The probability of these abnormal data packets having matching flow rules is very low, so a number of PACKET_IN messages will be generated and sent to the controller. If the packet buffer of the switch is filled up, the PACKET_IN message will contain the entire packet. This means that an amplified traffic is generated by the attacker to saturate the data plane and control plane [45, 46].

4.5.2 Virtual Host Attack

Attackers can use data-plane ARP spoofing attacks to make every other host in the network believe the existence of an actually non-existent host with a fake IP address IPvir. After this, when the victims receive packets with source IP address IPvir, they will make responses as usual. These responses will be sent to the controller using PACKET_IN messages because of the table-miss. Since the controller does not know the location of the virtual host, it will instruct the switches to flood the packets, causing network congestion.

4.5.3 Shared-path Attack

In large SDN networks, shared links exist between the control links and the data links. Therefore, an attacker can generate a data flow through these shared links to interfere with the transmission of control flows, achieving the effect of a DoS.

The network shown in **Fig. 5** has five switches {s1, s2, s3, s4, s5}. The hosts h1 and h3 communicate with each other through the data path h1→s2→s3→s4→h3, and the control path between s2 and the controller is s2→s3→s5→c. The link between s2 and s3 is a shared path. Assuming that host h1 is an attacker, it will send a well-crafted TCP-Targeted-DoS data stream to h3. Since the ports of s2 and s3 and the corresponding packet queue are also used by the control paths of s2 and s1, the malicious data stream can significantly delay or discard the control message between the SDN controller and the two switches.

Because there are only a few links in an SDN that transmit control data, it is necessary to determine the target shared path before attacking. [47] proposes a shared path identification method that can find the target path by traversing and testing each path. The main characteristic of this attack method is concealment, because the attacker only sends a relatively small data flow to the shared links, and the control flow is disturbed before the data flow reaches the controller, so it is difficult for the controller to detect the attack. The experiments on FloodLight and five hardware switches (AS4610-54T) show that the attack can significantly disrupt various network functionalities in SDN, including ARP Proxy, Learning Switch, Reactive Routing, etc.

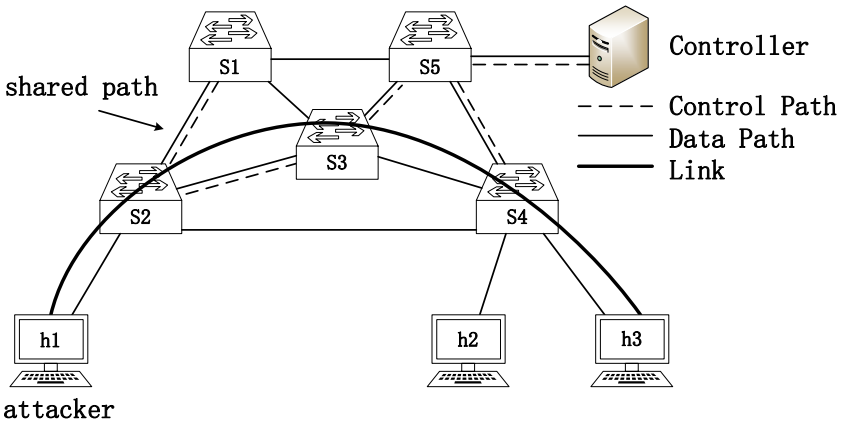


Fig. 5. Cross-Path Attack

4.5.4 Malicious Application Attacks

Most SDN controllers do not inspect applications before executing them, so malicious SDN applications can be deployed on the controller to carry out attacks, as shown in **Fig. 6**.

When the Floodlight controller receives a PACKET_IN message, it will deliver the message to each application registering to handle this message, in a certain order rather by broadcasting simultaneously. Therefore, if an application that receives the message earlier tampers with the message, the subsequent applications will process the error message and cause abnormal behaviour [48]. In addition, a malicious application can generate a FLOW_MOD message such that the buffer ID is set as the ID of a buffered packet in the switch and the *group_all* action contains many buckets of the *output:controller* action. When a switch receives the message, many copies of the buffered packets will be sent to the controller, congesting the southbound channel [49].

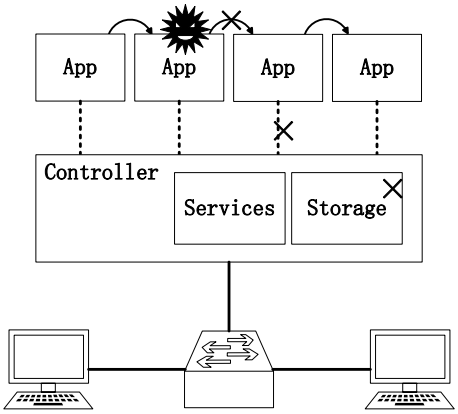


Fig. 6. Malicious Application Attacks

In addition to the above attack method, malicious applications can also unregister other SDN applications from the list of message listeners or abuse northbound APIs to terminate other legitimate applications, such as intrusion detection systems and firewalls. Since the controller has no restrictions on applications generating control messages and modifying the data stored in the internal memory, an application can generate malicious control messages arbitrarily or modify the network link information stored in the internal memory [22].

4.6 Elevation of Privilege

Access control is an important method used to protect system resources, architecture, and information by assigning different privileges to different entities. An SDN uses role-based access control (RBAC) to manage the permissions of applications.

In SDN networks, applications implement specific network functions by accessing or modifying the data storage of the control plane and interacting with the control plane through messages. Therefore, the control plane state is a "shared" state for each application. Although RBAC can restrict application access to shared data through role-based permission assignment, the lack of a good isolation mechanism for the state of the shared control plane and the lack of supervision of the internal information flow of the control plane will lead to the confusion of permissions between applications. For example, a host tracking application has write access to the host information database but has no write access to the flow rules. A routing application has read access to the host information database and both write and read access to the flow rules. If an attacker controls a host tracking application, he can modify the host database to change a host location to a malicious one. When the routing application detects this change, it will recalculate the flow path and then rewrite the flow rules. In this case, the host tracking application indirectly obtains permission to modify the flow rules through the routing application [50]. Besides, on the data plane, the hosts can utilize inappropriate responsibility assigning of the applications to bypass access control [51].

With the development of cloud computing, network as a service (NaaS) and other service applications are becoming increasingly extensive, and the deployment of SDN on commercial platforms has also developed. SDN uses a controller to allocate network resources to external customers [52]. The traffic among users is invisible and has a different quality of service. The SDN simplifies the division and distribution of network resources. However, sharing resources among different users will bring new privilege management issues.

5. SDN Security Defenses

As mentioned in the previous sections, SDN is faced with both traditional attack threats and threats specific to its own architecture. In response to these threats, researchers have also published relevant defence measures. In this section, we describe several defence methods against the abovementioned security threats for SDN.

To address the threat of spoofing, [21] proposed an application, SecureBinder, to prevent identifier attacks. SecureBinder can validate identifier binding changes and prevent independent binding across layers, utilizing the data and control separation, centralized controlling, and programmability of SDN. IEEE 802.1x is used to provide a root of trust for network identifiers for strong identity authentication and is able to prevent spoofing attacks on all network layers.

In response to the tampering threat, [23] proposed a method based on hierarchical protection to mitigate attacks against controller storage components. [53] proposed a route aggregation algorithm and a multi-level flow table structure, aiming to mitigate the vulnerability to flow table overflow. [24] proposed a defence framework, SPHINX, against network topology attacks and controller DoS attacks. SPHINX builds an abstract flow graph to detect attacks by dynamically monitoring network behaviour and incrementally verifying network constraints and updates. When finding suspicious behaviour of damaging the topology or data forwarding, it will raise alerts.

In response to repudiation threats, it is a very important consideration to introduce unique IDs for controllers and applications and to notify other network members of the operation of peer devices. The activities of faulty controllers and applications must be carefully recorded and monitored to provide the ability to properly track or discover suspicious behaviour. [44] and [36] proposed defence methods against covert channels. By installing a unique TLS certificate for switches and installing the switch DPID whitelist and the respective public key certificates of switches on the controller, the controller can recognize each switch using the certificates.

In response to the threat of information disclosure, Jafarian et al. [54] proposed a method of dynamically changing IP addresses to hide the host and system resources to resist the spread of worms and the scanning behaviour of attackers and to implement a defence framework, OF-RHM (OpenFlow random host mutation), based on the OpenFlow protocol. However, this method is just an improvement of the traditional network defence method and is not based on the characteristics of SDN. In this regard, Kampanaskis et al. [55] introduced the idea of mobile target defence to propose methods for SDN to resist topology discovery, network scanning, device information hiding, and randomization of hosts and routes. For example, the corresponding strategy is configured by the controller to randomly respond to spy packets so that the attacker cannot obtain accurate information. The defender can also use the OpenFlow protocol to complete the mapping and management of virtual addresses to achieve dynamic changes in controller and switch addresses to increase the attacker's detection range.

In response to DoS threats, [56] proposed a new link-flooding attacks (LFA) defence system, LFADefender, which uses the network view provided by the controller to detect and solve LFA. [40] proposed FLOODGUARD, which dynamically derives active flow rules to maintain the execution of network policies by reasoning about the runtime logic of the controller and applications, and temporarily buffers the flooding data packets and then sends them to the controller at a limited rate to prevent the controller from being overburdened. A defence scheme for packet injection attacks, TopoGuard, has also been proposed [18]. This defence scheme can effectively protect the network topology and at the same time has only a small impact on the normal operation of the OpenFlow controller. [46] proposed a lightweight

expansion module, PacketChecker, based on the SDN controller, which can effectively detect and mitigate the proliferation of forged data packets. [31] proposed a novel and effective defence framework, SWGuard, for control plane reflection attacks. SWGuard uses the host-application pair (HAP) as a new monitoring granularity to deviations of detect downlink message and make optimization if the channel is congested. For malicious applications, two defence mechanisms are (i) static or dynamic analysis and (ii) permission checking [48]. [57] proposed a DDoS defense system, POSEIDON, which can perform a wide range of DDoS defenses, especially dynamic defense.

In response to the threat of elevation of privilege, [49] proposed a defence framework, provSDN. Through tracking the information flow using the data provenance model, provSDN monitors the network to prevent against application privilege elevation attacks.

6. Related Work

Ortiz, Tiago V. et al [4] introduced the architecture, main components, and security threats of SDN, and reviewed the literature about security of SDN. They mainly classified the experimental environment of each literature, including the controller type and hardware or software environment. However, they did not note many technological details. Hori, Y. et al. [1] discussed the threats and countermeasures of the OpenFlow network. The security threats and risks of the OpenFlow network were clarified. On this basis, they discussed the methods of risk assessment and preventive measures for each safety risk. Another survey paper, [6], summarized the situation of SDN security and introduced the main attack methods according to the process of the network attack, including target network detection, forgery and deception to achieve network access, DoS attacks and information leakage. Yoon, Changhoon, et al. [22] evaluated the security requirements of the control plane, data plane and control channel from the three perspectives of confidentiality, integrity and availability and introduced specific attack types against these three components.

Our work was inspired by these early studies, and we integrate these works and newly published papers using the STRIDE model, providing another perspective on the security threats of SDN. Although SDN is a layered architecture, all levels are interrelated and influence each other. And most attack methods can impact all three layers from one layer. So, instead of independently analyzing the security of each layer of SDN, we examine the threats faced by the overall network architecture, creating a more systematic view of the SDN security.

7. Conclusion

As a next-generation network architecture, SDN has broad prospects, although their security issues have received increasing attention. This paper discusses the security threats and defence methods of SDN systems. First, the basic architecture of SDN is introduced. Spoofing, tampering, repudiation, information disclosure, DoS and elevation of privilege, the current threats faced by SDN are analysed in detail.

As seen in the previous discussion, security weaknesses exist at all layers of SDN, and attacks against each layer can affect the entire network. Many ingenious attacks are variants of traditional attacks, which are improved according to the architecture of SDN.

In terms of defence, many studies choose to extend the controller security functions or return some of the decision-making functions to the data plane. The main problem is that most defences are passive and unable to protect networks from new attack patterns.

Attack and defence methods are struggling, and there is much room for research on both sides. Future research on SDN security will focus on the following aspects:

(1) Increasing controller security. As the core of the SDN architecture, the controller cannot only focus on the implementation of control logic but must have sufficient security mechanisms to ensure the robustness of the network. In the future, security controllers with stronger access control and authentication mechanisms will be important research directions.

(2) Automated SDN security assessment. SDN security assessment is an important part of security research, and there is an increasing demand for comprehensive security assessment of SDN components and the interactions between them. The development of automated security assessment tools can effectively reveal potential threats, which is important for improving defences.

(3) Northbound interface standardization. The application layer and the control layer communicate through the northbound interface. However, there is currently no standardized protocol for the SDN northbound interface. Different controllers have different northbound interfaces, and the application permission control is insufficient. This allows attackers to use malicious programs to attack the network. Therefore, designing a safe northbound protocol is a priority of future work.

Acknowledgement

We thank the reviewers for their constructive comments. This work was supported by the National Key Research and Development Program of China under Grant 2016QY07X1404, 2019QY0501.

References

[1] Y. Hori, S. Mizoguchi, R. Miyazaki, A. Yamada, Y. Feng, A. Kubota, and K. Sajurai, "A Comprehensive Security Analysis Checksheet for OpenFlow Networks," in *Proc. of the International Conference on Broad-Band Wireless Computing, Communication and Applications*, pp. 231-242, Oct. 2016. Article (CrossRef Link)

[2] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, Jan. 2015. Article (CrossRef Link)

[3] S. Jero, X. Bu, C. Nitarotaru, H. Okhravi, R. Skowyra, and S. Fahmy, "BEADS: Automated Attack Discovery in OpenFlow-Based SDN Systems," in *Proc. of International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 311-333, Oct. 2017. Article (CrossRef Link)

[4] V. Tiago, B. Kimura, J. Ueyama, and V. Rosset, "Experimental Security Analysis of Controller Software in SDNs: A Review," *ArXiv Preprint* ArXiv:1906.09546, 2019. Article (CrossRef Link)

[5] W. Mengmeng, L. Jianwei, and M. Jian, "Software Defined Networking: Security Model, Threats and Mechanism," *Journal of Software*, vol. 27, no. 4, pp. 205-228, Apr. 2016. Article (CrossRef Link)

[6] W. Zehui, W. Qiang, and W. QingXian, "Survey for Attack and Defense Approaches of OpenFlow-Enabled Software Defined Network," *Computer Science*, vol. 44, pp. 121-132, 2017. Article (CrossRef Link)

[7] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, and A. Singh, "B4: Experience with a globally-deployed software defined WAN," in *Proc. of the Conference on SIGCOMM*, vol. 43, no. 4, pp. 3-14, Aug. 2013. Article (CrossRef Link)

[8] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards Secure and Dependable Software-Defined Networks," in *Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, pp. 55-60, Aug. 2013. Article (CrossRef Link)

[9] Open Networking Foundation. [Online] Available: https://www.opennetworking.org
[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulka, and L. Peterson, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM communication Review*, vol. 38, no. 2, pp. 69-74, Mar. 2008. Article (CrossRef Link)
[11] Floodlight. [Online] Available: http://Floodlight.openflowhub.org
[12] Linux Foundation, "OpenDaylight". Article (CrossRef Link)
[13] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: towards an open, distributed SDN OS," in *Proc. of the 3rd Workshop on Hot Topics in Software Defined Networking*, pp. 1-6, Aug. 2014. Article (CrossRef Link)
[14] Ryu. [Online] Available: http://osrg.github.com/ryu
[15] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105-110, July 2008. Article (CrossRef Link)
[16] U. Steinhoff, A. Wiesmaier, and R. Araújo, "The state of the art in DNS spoofing," *ACNS*, 2006. Article (CrossRef Link)
[17] J. King and K. Lauerman, "ARP poisoning (man-in-the-middle) attack and mitigation techniques," Oct. 2020. Article (CrossRef Link)
[18] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures," in *Proc. of Network and Distributed System Security Symposium*, 2015. Article (CrossRef Link)
[19] Pickett G, "Abusing Software Defined Networks," Article (CrossRef Link)
[20] D. Smyth, V. Cionca, S. Mcsweeney, and D. O'Shea, "Exploiting Pitfalls in Software-Defined Networking Implementation," in *Proc. of International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pp. 1-8, July 2016. Article (CrossRef Link)
[21] S. Jero, W. Koch, R. Skowyra, H. Okhravi, C. N. Rotaru, and D. Bigelow, "Identifier Binding Attacks and Defenses in Software-Defined Networks," in *Proc. of the 26th USENIX Security Symposium (USENIX Security 17)*, pp. 415-432, 2017. Aritcle (CrossRef Link)
[22] C. Yoon, S. Lee, H. Kang, S. Shin, V. Yegnesqaran, P. Porras, and G. Gu, "Flow Wars: Systemizing the Attack Surface and Defenses in Software-Defined Networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3514-3530, 2017. Article (CrossRef Link)
[23] V. H. Dixit, A. Doupe, Y. Shoshitaishvili, Z. Zhao, and G. J. Ahn, "AIM-SDN: Attacking Information Mismanagement in SDN-Datastores," in *Proc. of ACM SIGSAC Conference on Computer and Communications Security*, pp. 664-676, 2018. Article (CrossRef Link)
[24] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: Detecting Security Attacks in Software-Defined Networks," in *Proc. of Network and Distributed System Security Symposium*, 2015. Article (CrossRef Link)
[25] S. Shin, P. Porras, V. Yegneswara, M. Fong, G. Gu, and M. Tyson, "FRESCO: Modular Composable Security Services for Software-Defined Networks," in *Proc. of the 20th Annual Network Distributed System Security Symposium*, 2013. Aritcle (CrossRef Link)
[26] D. Smyth, V. Cionca, S. Mcsweeney, and D. O'Shea, "Exploiting Pitfalls in Software-Defined Networking Implementation," in *Proc. of International Conference On Cyber Security And Protection Of Digital Services (Cyber Security)*, pp. 1-8, 2016. Article (CrossRef Link)
[27] H. Wang, L. Xu, and G. Gu, "FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks," in *Proc. of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 239-250, 2015. Article (CrossRef Link)
[28] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: Detecting Security Attacks in Software-Defined Networks," in *Proc. of Network and Distributed System Security Symposium*, 2015. Article (CrossRef Link)
[29] R. Kloti, V. Kotronis, and P. Smith, "OpenFlow: A Security Analysis," in *Proc. of the 21st IEEE International Conference on Network Protocols (ICNP)*, pp. 1-6, 2013. Article (CrossRef Link)

[30] J. Cao, M. Xu, and Q. Li, "Disrupting SDN via the Data Plane: A Low-Rate Flow Table Overflow Attack," in *Proc. of International Conference on Security and Privacy in Communication Systems*, pp. 356-376, 2017. Article (CrossRef Link)
[31] M. Zhang, G. Li, L. Xu, J. Bi, G. Gu, and J. Bai, "Control Plane Reflection Attacks in SDNs: New Attacks and Countermeasures," in *Proc. of International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 161-183, Sep. 2018. Article (CrossRef Link)
[32] S. Lee, C. Yoon, C. Lee, S. Shin, V. Yegneswaran, and P. Porras, "DELTA: A Security Assessment Framework for Software-Defined Networks," in *Proc. of Network and Distributed System Security Symposium*, 2017. Article (CrossRef Link)
[33] L. Xu, J. Huang, S. Hong, J. Zhang, and G. Gu, "Attacking the Brain: Races in the SDN Control Plane," in *Proc. of the 26th USENIX Security Symposium*, pp. 451-468, Aug. 2017. Article (CrossRef Link)
[34] H. Wang, G. Yang, P. Chinprutthiwong, L. Xu, Y. Zhang, and G. Gu, "Towards Fine-Grained Network Security Forensics and Diagnosis in the SDN Era," in *Proc. of the 25th ACM Conference on Computer and Communications Security*, pp. 3-16, 2018. Article (CrossRef Link)
[35] H. Röpke and T. Holz, "SDN Rootkits: Subverting Network Operating Systems of Software-Defined Networks," in *Proc. of International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 339-356, 2015. Article (CrossRef Link)
[36] R. Krosche, K. Thimmaraju, L. Schiff, and S. Schmid, "I DPID It My Way! A Covert Timing Channel in Software-Defined Networks," in *Proc. of IFIP Networking Conference (IFIP Networking) and Workshops*, pp. 217-225, 2018. Article (CrossRef Link)
[37] J. Hua, Z. Zhou, and S. Zhong, "Flow Misleading: Worm-Hole Attack in Software-Defined Networking via Building In-Band Covert Channel," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1029-1043, 2020. Article (CrossRef Link)
[38] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in Software Defined Networks: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2317-2346, 2015. Article (CrossRef Link)
[39] S. Shin and G. Guofei, "Attacking Software-Defined Networks: A First Feasibility Study," in *Proc. of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, pp. 165-166, 2013. Article (CrossRef Link)
[40] A. Azzouni, O. Braham, T. M. Nguyen, G. Pujolle, and R. Boutaba, "Fingerprinting OpenFlow Controllers: The First Step to Attack an SDN Control Plane," in *Proc. of IEEE Global Communications Conference (GLOBECOM)*, pp. 1-6, 2016. Article (CrossRef Link)
[41] J. Sonchack, A. J. Aviv, and E. Keller, "Timing SDN Control Planes to Infer Network Configurations," in *Proc. of ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, pp. 19-22, 2016. Article (CrossRef Link)
[42] M. Yu, T. He, P. Mcdaniel, and Q. K. Burke, "Flow Table Security in SDN: Adversarial Reconnaissance and Intelligent Attacks," in *Proc. of IEEE Conference on Computer Communications*, pp. 1519-1528, 2020. Article (CrossRef Link)
[43] J. Cao, Z. Yang, K. Sun, Q. Li, M. Xu, and P. Han, "Fingerprinting SDN Applications via Encrypted Control Traffic," in *Proc. of the 22nd International Symposium on Research in Attacks, Intrusions and Defenses*, pp. 501-515, 2019. Article (CrossRef Link)
[44] K. Thimmaraju, L. Schiff, and S. Schmid, "Outsmarting Network Security with SDN Teleportation," in *Proc. of IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 563-578, Apr. 2017. Article (CrossRef Link)
[45] H. Wang, L. Xu, and G. Gu, "FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks," in *Proc. of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 239-250, 2015. Article (CrossRef Link)
[46] 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, Oct. 2017. Article (CrossRef Link)

[47] J. Cao, Q. Li, R. Xie, K. Sun, G. Gu, M. Xu, and Y. Yang, "The CrossPath Attack: Disrupting the SDN Control Channel via Shared Links," in *Proc. of the 28th USENIX Security Symposium*, pp. 19-36, Aug. 2019. Article (CrossRef Link)

[48] S. Lee, C. Yoon, and S. Shin, "The Smaller, the Shrewder: A Simple Malicious Application Can Kill an Entire SDN Environment," in *Proc. of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, pp. 23-28, 2016. Article (CrossRef Link)

[49] C. Jiahao, X. Renjie, K. Sun, Q. Li, G. Gu, and M. Xu, "When Match Fields Do Not Need to Match: Buffered Packet Hijacking in SDN," in *Proc. of the Network and Distributed System Security Symposium*, Feb. 2020. Article (CrossRef Link)

[50] B. E. Ujcich, S. Jero, A. Edmundson, Q. Wang, R. Skowyra, J. Landry, A. Bates, W. H. Sanders, C. Nita-Rotaru, and H. Okhravi, "Cross-App Poisoning in Software-Defined Networking," in *Proc. of the 25th ACM Conference on Computer and Communications Security*, pp. 648-663, Oct. 2018. Article (CrossRef Link)

[51] B. E. Ujcich, S. Jero, R. Skowyra, S. R. Gomez, A. Bates, W. H. Sanders, and H. Okhravi, "Automated Discovery of Cross-Plane Event-Based Vulnerabilities in Software-Defined Networking," in *Proc. of Network and Distributed System Security Symposium*, 2020. Article (CrossRef Link)

[52] Q. Duan, "End-to-End Service Delivery with QoS Guarantee in Software Defined Networks," *Transactions on Networks and Communications*, vol. 6, no. 2, 2018. Article (CrossRef Link)

[53] Y. Zhou, K. Chen, J. Zhang, J. Leng, and Y. Tang, "Exploiting the Vulnerability of Flow Table Overflow in Software-Defined Network: Attack Model, Evaluation, and Defense," *Security and Communication Networks*, vol. 2018, pp. 1-15, Jan. 2018. Article (CrossRef Link)

[54] J. H. Jafarian, E. Alshaer, and Q. Duan, "Openflow Random Host Mutation: Transparent Moving Target Defense Using Software Defined Networking," in *Proc.of the 1st Workshop on Hot Topics in Software Defined Networks*, pp. 127-132, Aug. 2012. Article (CrossRef Link)

[55] P. Kampanakis, H. Perros, and T. Beyene, "SDN-Based Solutions for Moving Target Defense Network Protection," *in Proc. of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 1-6, June 2014. Article (CrossRef Link)

[56] J. Wang, R. Wen, J. Li, F. Yan, B. Zhao, and F. Yu, "Detecting and Mitigating Target Link-Flooding Attacks Using SDN," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 6, pp. 944-956, Apr. 2018. Article (CrossRef Link)

[57] M. Zhang, G. Li, S. Wang, C. Liu, A. Chen, H. Hu, G. Gu, Q. Li, M. Xu, and J. Wu, "Poseidon: Mitigating Volumetric DDoS Attacks with Programmable Switches," in *Proc. of Network and Distributed System Security Symposium*, Jan. 2020. Article (CrossRef Link)

Wenbin Zhang is a M.A. degree candidate of The China National Digital Switching System Engineering and Technological Research Center. His research interests include Cyberspace Security and Software-Defined Networking (SDN).

Zehui Wu received Ph.D degree from The China National Digital Switching System Engineering and Technological Research Center. He is currently a lecturer at China National Digital Switching System Engineering and Technological Research Center. His research interests include Software Vulnerability and Software-Defined Networking (SDN).
‘

Qiang Wei is a professor and doctoral tutor at China National Digital Switching System Engineering and Technological Research Center, Zhengzhou, China. His research interests include Software Vulnerability and Cyberspace Security.

Huijie Yuan is a M.A. degree candidate of The China National Digital Switching System Engineering and Technological Research Center. His research interests include Software Vulnerability and Cyberspace Security.