Security Threats in the Data Plane of Software-Defined Networks

Shang Gao, Zecheng Li, Bin Xiao, and Guiyi Wei

Abstract

SDN has enabled extensive network programmability and speedy network innovations by decoupling the control plane from the data plane. However, the separation of the two planes could also be a potential threat to the whole network. Previous approaches pointed out that attackers can launch various attacks from the data plane against SDN, such as DoS attacks, topology poisoning attacks, and side-channel attacks. To address the security issues, we present a comprehensive study of data plane attacks in SDN, and propose FlowKeeper, a common framework to build a robust data plane against different attacks. FlowKeeper enforces port control of the data plane and reduces the workload of the control plane by filtering out illegal packets. Experimental results show that FlowKeeper could be used to efficiently mitigate different kinds of attacks (i.e., DoS and topology poisoning attacks).

Introduction

Software-defined networking (SDN) is regarded as one of the most promising architectures for next generation computer networks. By separating the ossified network infrastructure into control plane and data plane, SDN allows operators to dynamically control network traffic via controller applications with high programmability. In SDN, the centralized control plane operates like a brain to manage and control network flows, while the data plane works as a body to process each flow based on the decisions of the control plane. The communications between the two planes are supervised by a "southbound" protocol (i.e., OpenFlow [1]). In recent years, OpenFlow networks have been adopted in data centers [2] and 5G networks [3].

While communications between the control plane and data plane enable high programmability, they also open the door for attackers to launch new attacks against OpenFlow networks. Attacks from the data plane have posed great threats to SDN [4]. By simply utilizing several hosts under OpenFlow switches, attackers can dysfunction, disturb the control plane, or learn its behaviors without much information about controller applications. These attacks include the following.

DoS Attacks: The resources of the control and data planes could be targets of denial of service (DoS) attacks (e.g., data-to-control plane saturation attacks [5, 6]). Attackers can jam a switch-controller bandwidth, overload a switch's flow table and memory, and consume the CPU and memory of the controller by flooding table-miss packets (table-miss packets will trigger data-control plane communications).

Topology Poisoning Attacks: The global network view of the control plane could be poisoned by attackers (e.g., network topology poisoning attacks [7]). By forging or relaying Link Layer Discovery Protocol (LLDP) packets, attackers can forge nonexistent links between switches.

Side-Channel Attacks: Attackers can learn the detail of network configurations [8–11] (i.e., some logic of controller applications). These attacks analyze the amount of delay added to timing pings that are specifically crafted to infer the network configurations and control application logic.

The countermeasures against these attacks have also been studied. First, to mitigate DoS attacks, AvantGuard builds a TCP proxy on the data plane as an extension to verify the legitimacy of TCP handshakes [5]. Furthermore, a protocol-independent defense system, FloodGuard, pre-installs proactive flow rules to reduce table-miss packets and forwards table-miss packets to an additional data plane cache [12]. To reduce the cost of hardware modifications, FloodDefender offloads table-miss packets to neighbor switches and filters out attack traffic with two-phase filtering [13]. Second, for topology poisoning attacks, TopoGuard identifies the type of connected device, and dynamically checks the updates [7]. LLDP packets will be dropped when received from a host-connected port. Finally, side-channel attacks can be prevented by normalizing the control plane delay to a configurable default responding time [11].

The architectures of these defense systems vary from hardware extensions to additional specific devices. In this article, we discuss a common defense system architecture against various kinds of attacks in OpenFlow networks. Furthermore, based on the architecture, we introduce FlowKeeper to build a robust data plane against different attacks. FlowKeeper enforces port control of each OpenFlow switch and reduces the workload of the control plane by filtering out illegal packets when DoS or topology poisoning attacks occur.

We conclude our contributions as follows:

- We present a comprehensive review of data plane attacks in OpenFlow networks and discuss the architecture of each countermeasure.
- We propose a common defense system architecture. Based on this architecture, we propose FlowKeeper to mitigate different attacks.

Digital Object Identifier:
10.1109/MNET.2018.1700283

Shang Gao, Zecheng Li, and Bin Xiao are with The Hong Kong Polytechnic University; Guiyi Wei is with Zhejiang Gongshang University.

0890-8044/18/$25.00 © 2018 IEEE

Data plane attacks	DoS			Topology poisoning	Side-channel	All attacks
Countermeasures	AvantGuard [5]	FloodGuard [12]	FloodDefender [13]	TopoGuard [7]	Timeout Proxy [11]	FlowKeeper
Controller application		✓	✓	✓		✓
Data plane extension	✓				✓	
Additional device		✓				✓

TABLE 1. Architectures of the countermeasures against data plane attacks.

- We evaluate the effectiveness of FlowKeeper in both hardware and software environments.

Data Plane Attacks and Countermeasures

We first introduce the packet processing mechanism in SDN. Then we present three data plane attack vectors. Finally, we discuss the countermeasures. We summarize the architecture of these countermeasures in Table 1 for convenience.

SDN Workflow

In OpenFlow networks, the control plane directs the data plane to process incoming traffic via two approaches: proactive flow rule installation and reactive flow rule installation. In the proactive approach, the controller pre-installs all flow rules into the flow tables of OpenFlow switches. The switches then process each incoming packet based on these flow rules. In the reactive approach, when a switch receives a packet that cannot match with any existing flow rules (table-miss packet), it will follow the following four steps to process the packet. First, the switch will buffer this packet, encapsulate its header into a packet_in message, and report to the controller.[1] Second, the controller decides the action(s) based on the logic of the controller applications and sends a packet_out message to indicate the reaction to the table-miss packet. Third, the switch will process this packet based on the action field of the packet_out message. Finally, the controller can further install flow rules on the switch to allow it to process the packets of the same flow without consulting the control plane.

The reactive approach allows the controller to manage traffic dynamically and enables a global view of the network for the controller (e.g., collecting network topology information via LLDP packets). For instance, the controller can use a packet_out message to make a switch broadcast LLDP packets. When neighbor switches receive these LLDP packets, they will report them to the controller. The controller then knows the connections between these switches, and can further learn the network topology. The reactive approach has been applied in most SDN applications.

Data Plane Attacks in SDN

Threat 1: DoS attacks: DoS attacks against SDN are regarded as the biggest threat. The data-to-control plane saturation attacks [5] utilize table-miss to flood both the control and data planes. Specifically, an attacker uses several compromised hosts (botnet) to send massive packets to an OpenFlow switch by randomly forging some fields. Since these packets have a very low probability to match with existing flow rules, the switch will regard them as table-miss packets and deliver them to the controller in packet_in messages. These packet_in messages will consume a large amount of switch-controller bandwidth and controller resources (e.g., CPU and memory). Also, the memory of the switch will be exhausted by table-miss packets, and when the controller decides to install flow rules to handle attack traffic, the switch's flow table will be overloaded.

Threat 2: Topology Poisoning Attacks: Topology poisoning attacks forge or relay some control packets (i.e., LLDP) in an OpenFlow network to poison the global information collected by a controller [7]. First, an attacker monitors genuine LLDP packets and records the corresponding LLDP syntax. Second, the attacker can either modify some specific contents of the LLDP packets (e.g., port number) to forge a response to the controller, or repeat them to other compromised hosts to trigger the connected switches response to the controller. As a result, a nonexistent link between two disconnected switches is created. The attacker can further launch DoS attacks (blocking some legal ports of the target switch) or man-in-the-middle attacks (building an LLDP relay channel) based on the topology poisoning attacks.

Threat 3: Side-Channel Attacks: Side-channel attacks utilize the processing time of a control plane to learn network configurations [8–11]. In these attacks, an attacker specifically crafts different kinds of timing probes (e.g., ARP requests for MAC layer and low TTL packets for IP layer) and sends a stream of probes (test stream) and some baseline packets with known effects (e.g., should be reported to the controller before forwarding) to the OpenFlow network. By comparing the response times of the test stream and baseline packets, the attacker can learn whether the network runs OpenFlow [8], the size of switches' flow table [9], whether links contain aggregate flows [10], host communication records [11], network access control configurations [11], and network monitoring policies [11].

Countermeasures Against Data Plane Attacks

DoS Countermeasures: AvantGuard [5] is the first defense system against data-to-control plane saturation attacks. It extends the hardware of OpenFlow switches with a TCP proxy to mitigate TCP-based attacks. The proxy responds with a SYN-ACK packet and forwards the SYN packet to check the existence of the source and destination. AvantGuard will regard the connection as legal only when both source and destination exist. The problem is that AvantGuard can only deal with TCP-based attacks and introduces a long delay for legal SYN packets.

[1] A packet_in message will contain the whole packet when the memory of the OpenFlow switch is full.

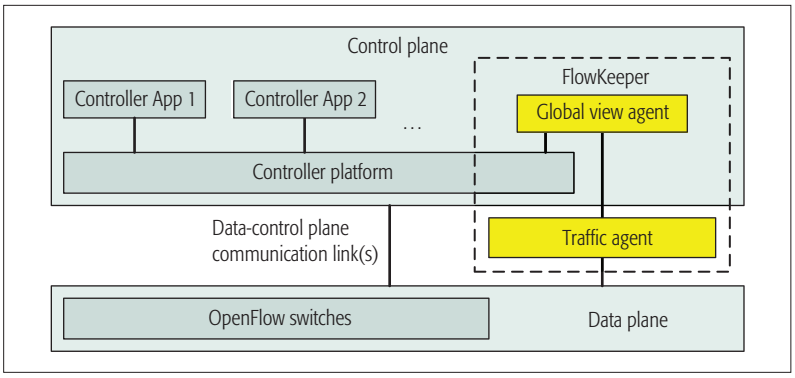

FIGURE 1. Architecture of FlowKeeper.

To mitigate other attack traffic (e.g., UDP and ICMP), FloodGuard [12] pre-installs possible flow rules (proactive flow rules) to handle as much normal traffic as possible, and forwards attack traffic to an additional device (data plane cache) to mitigate attacks. The data plane cache sorts incoming packets based on protocol and reports the head of each protocol queue by round-robin scheduling under a predefined rate. The problem with FloodGuard is a lack of packet filtering. Therefore, it may introduce long delays and a high packet loss rate for some packets (e.g., UDP packets will be affected by UDP-based attacks [13]).

FloodDefender [13] uses existing SDN features to mitigate data-to-control plane saturation attacks without additional devices or hardware modifications. It first saves switch-controller bandwidth by offloading attack traffic to neighbor switches with protecting rules. Furthermore, it uses a two-phase filtering (first filtering out most attack traffic based on frequency and then precisely identifying the rest of the attack packets) to reduce control plane resource consumption. Finally, it separates the flow table into cache region and flow table region to reduce useless flow rules. The biggest problem with FloodDefender is its protecting efforts. Without additional devices, FloodDefender is not as efficient as solutions such as FloodGuard.

Topology Poisoning Countermeasures: The main strategy of topology poisoning attacks is to send LLDP packets through a host connected port. Therefore, TopoGuard [7] aims at identifying the type of neighbor devices connected to OpenFlow switches. LLDP packets from a host connected port will be regarded as illegal and discarded. Specifically, TopoGuard works on the control plane and tracks the type of neighbor devices connected to switches' ports. If a port first receives LLDP packets, TopoGuard will regard the neighbor device as a switch. When packets from a first-hop host are first received, the neighbor device is regarded as a host. Otherwise, TopoGuard continues monitoring incoming traffic. Based on the port property, TopoGuard can avoid topology poisoning attacks by blocking LLDP packets from host connected ports. To verify a topology update, TopoGuard also uses host probes to check the existence of the host in the former location. Since TopoGuard enables a flexible approach to identify the type of connected device dynamically, it may allow attackers to forge a neighbor device transfer from host to switch (first sending Port_Down signals and then LLDP packets).

Side-Channel Countermeasures: Side-channel attacks utilize the delay of some packets to guess network configurations. Therefore, the authors in [11] introduce a timeout proxy on the data plane as an extension to normalize control plane delay. When the control plane fails to respond within a fixed period of time, the timeout proxy will send a default forwarding instruction to the request. The timeout proxy reduces the response time of some long delay packets to avoid side-channel attacks, but can also reduce network programmability (by changing the processing strategies of these long delay packets into a proactive approach). Also, the predefined response time should be adjusted dynamically with the workload of the control plane.

Building A Robust Data Plane

We introduce a common architecture, FlowKeeper, to build a robust data plane against different data plane attacks. We first present the architecture of FlowKeeper. Then we discuss how to use FlowKeeper against different attacks.

FlowKeeper Architecture

FlowKeeper consists of two modules: the traffic agent and the global view agent, as depicted in Fig. 1. The traffic agent stands between the control plane and data plane. It can replace most data plane extensions to make FlowKeeper compatible with existing OpenFlow devices. By introducing some intelligence into the data plane, the traffic agent processes some traffic to reduce the workload of the controller. The global view agent works on the control plane as a controller application. It provides global network information to the traffic agent and delivers some flows to other controller applications.

The traffic agent module can process some table-miss packets autonomously to offload some workload of the control plane. Instead of reporting table-miss packets to the controller, OpenFlow switches can follow some forwarding rules to forward these packets to the traffic agent. With this traffic information, the traffic agent can identify and filter out illegal packets (against DoS attacks) and delay some packets (against side-channel attacks). Also, the traffic agent can work actively to verify the existence of hosts by sending probing packets (against topology poisoning attacks).

The global view agent module is the bridge between the traffic agent module and other controller applications. It serves three roles. First, the global view agent installs default rules to forward table-miss packets to the traffic agent instead of reporting to the controller. Second, it monitors network status and provides global information to the traffic agent. With this global information, the traffic monitor more precisely identifies illegal packets. Finally, the global view agent also delivers some packets from the traffic agent to other controller applications to update the global view of the control plane.

FlowKeeper Solution

Against DoS Attacks: We first show how to deploy FlowKeeper against DoS attacks. In this scenario, the traffic agent will filter out some attack traffic, and enforce data plane security

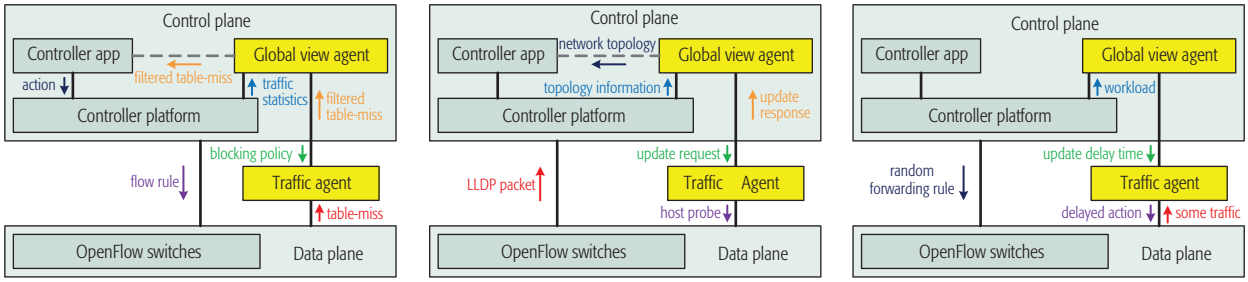

FIGURE 2. FlowKeeper against different kinds of data plane attacks. Dash lines indicate logical connections. a) FlowKeeper against DoS attacks; b) FlowKeeper against topology poisoning attacks; c) FlowKeeper against side-channel attacks.

policies based on the direction from the global view agent, as depicted in Fig. 2a. The global view agent will first install forwarding rules to the victim switch. Therefore, the table-miss packets are forwarded to the traffic agent via data plane links, and the switch-controller bandwidth can be saved. Second, the traffic agent classifies received flows based on two frequency thresholds, f_{low} and f_{high} (flows are identified based on protocol ID, source/destination MAC, and source/destination IP if applicable). Low-frequency flows (frequency lower than flow) will be filtered out to eliminate packets with forged sources, and high-frequency flows (frequency higher than f_{high}) will be regarded as legal and processed normally (delivering to the controller via the global view agent). Other flows are delivered to the global view agent for further analysis. Third, the global view agent precisely identifies attack traffic based on the statistical traffic information.[2] It can also update the blocking policies to the traffic agent when attack packets with real source addresses are identified (dropping packets from some malicious hosts), and adjust the frequency thresholds based on the classification result. Finally, the filtered packets are then delivered to other controller applications (via the controller platform) and processed based on the logic of these applications. Flow rules can be further installed on OpenFlow switches to process normal traffic when received again.

The frequency thresholds (f_{low} and f_{high}) will be set to 1 and 10 by default. We also allow administrators and the controller to adjust these two values based on their demands. A higher f_{low} can filter out more attack packets but may sacrifice some benign traffic. Similarly, a higher f_{high} will identify less benign traffic but ensures fewer attack packets regarded as benign ones. To set proper thresholds, administrators can use some training data to find the proper f_{low} that can filter out more than 50 percent of attack traffic with less than 0.5 percent false-positives, and f_{high} to identify more than 30 percent of benign traffic with less than 0.5 percent false-positives. The controller can also adjust the thresholds dynamically based on traffic classification result. For instance, when less than 50 percent of attack traffic is filtered out, the controller can increase f_{low} (by 1) to increase the efficiency of filtering. Note that 50 percent and 30 percent are acceptable in filtering, since we also adopt traffic classification after filtering. Other attack and benign traffic can be further identified by traffic classification.

Against Topology Poisoning Attacks: FlowKeeper monitors network topology changes with the global view agent and uses the traffic agent to identify the type of neighbor device connected to each switch, as depicted in Fig. 2b. The global view agent preserves a list of neighbor device type (*host*/*switch*/*any*/*untested*) on all enabled ports of OpenFlow switches. Initially, the devices will be set to "*any*." When LLDP packets are received from an *any*-port, the device type of this port will be set to *switch*; when first-hop host packets are received from an *any*-port, the device type of this port will be set to *host*. *Switch*-to-*any* transfers can also be triggered by receiving Port_Down signals. To avoid attackers forging illegal *host*-to-*switch* transfers, we set the device type to *untested* when receiving Port_Down signals from a *host*-port. After the updates are verified by the traffic agent, the device type is set to *any*. When LLDP packets are received by the global view agent, it can filter out illegal LLDP packets (LLDP packets from a *host*/untested *port) against topology poisoning attacks*.

The traffic agent adopts a probing scheme to verify the existence of the previous disconnected hosts on *untested* port. Specifically, it uses both ICMP echoes and ARP requests as probes to test the existence. When no ICMP replies and ARP responses are received, the updates will be identified as legal. The traffic agent further informs the global view agent with *untested*-to-*any* updates. Otherwise, the previous host still exists, and the updates will be regarded as illegal. The traffic agent will alert the global view agent with the illegal updates.

Against Side-Channel Attacks: In this scenario, FlowKeeper uses a traffic agent to delay some traffic and disrupt the response time against side-channel attacks, as depicted in Fig. 2c. The global view agent randomly installs some temporary forwarding rules to OpenFlow switches to deliver some delay-tolerable traffic to the traffic agent. When the traffic agent receives these packets, it first applies a delay on these received packets and then processes them based on the action from the controller. The delay time can also be adjusted dynamically by the global view agent based on the workload of the controller. Since a delay is applied to some packets, the response time of the probes becomes unreliable to infer network configurations. This delay will not greatly affect benign traffic.

[2] Different techniques (e.g., machine learning) could be applied to precisely identify attack traffic.

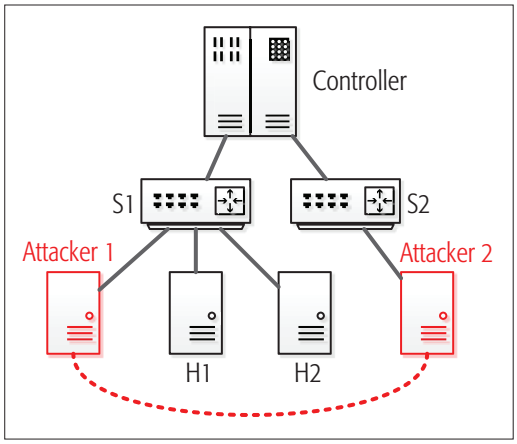

FIGURE 3. Test topology. We do not show traffic agent to simplify the expression.a

EXPERIMENT

We implement FlowKeeper against DoS attacks and topology poisoning attacks, and evaluate its performance in both software and hardware environments.

IMPLEMENTATION AND SETUP

We implement the FlowKeeper system, including the global view agent and the traffic agent modules. The global view agent is implemented as a controller application on the RYU controller in Python, and the traffic agent is implemented on an additional Linux host between the control plane and data plane in C++. To mitigate DoS attacks, the traffic agent sets $f_{low} = 2$ and $f_{high} = 10$ to classify received flows. The global view agent will further use Support Vector Machine (SVM) [14], a supervised learning model, as our classifier to filter out attack traffic from other flows ($2 \leqslant$ flow.frequency $\leqslant 10$). In the design of topology poisoning mitigation, we modify the port property in [7] by adding *untested* type into the global view agent. In the traffic agent, we use *scapy* to send ARP and ICMP probes.

We evaluate the protecting effort of the FlowKeeper system in both software and hardware environments. The RYU controller is installed on a computer equipped with an i7 CPU and 8 GB memory. In the software environment, we use Mininet to create the network with virtual OpenFlow switches, while in the hardware environment, we use Polaris xSwitch X10-24S2Q [15], a commercial OpenFlow switch to build the network. The test topology is depicted in Fig. 3.

EVALUATION

DoS Attacks: We compare the bandwidth consumption under DoS attacks. Attacker 1 will flood S1 with TCP packets (randomly forged source/destination IP and MAC). We measure the available bandwidth between H1 and H2 to show the protecting effort of FlowKeeper. The result is presented in Fig. 4. In the software environment, the bandwidth of an OpenFlow network (without any protecting system) will be quickly exhausted. The network becomes dysfunctional under 450 PPS (packet per second) attack rate, while with the protection of FlowKeeper, the DoS attacks consume only 9 percent of the bandwidth. Though the network still works under 500 PPS in the hardware environment, the DoS attacks consume more than 80 percent of the bandwidth. The network with FlowKeeper preserves 80 percent of the available bandwidth in the hardware environment.

Topology Poisoning Attacks: We show the protecting effort of FlowKeeper under topology poisoning attacks. Attacker 1 and attacker 2 will send forged LLDP packets to forge a link between S1 and S2. We show the verification of LLDP packets on the controller in Fig. 5. Once a link update is detected, the global view agent will verify the legitimacy of received LLDP packets based on the type of connected device. Since these forged LLDP packets are received from a *host*-port, they will be regarded as illegal and ignored.

CONCLUSION

Attacks from the data plane pose a great threat to SDN. In this article, we first systematically review three data plane attacks (i.e., DoS attacks, topology poisoning attacks, and side-channel attacks) and existing countermeasures. We further present a common design, FlowKeeper, against these three attacks. We implement FlowKeeper to defend against DoS attacks and topology poisoning attacks. Experimental results show that

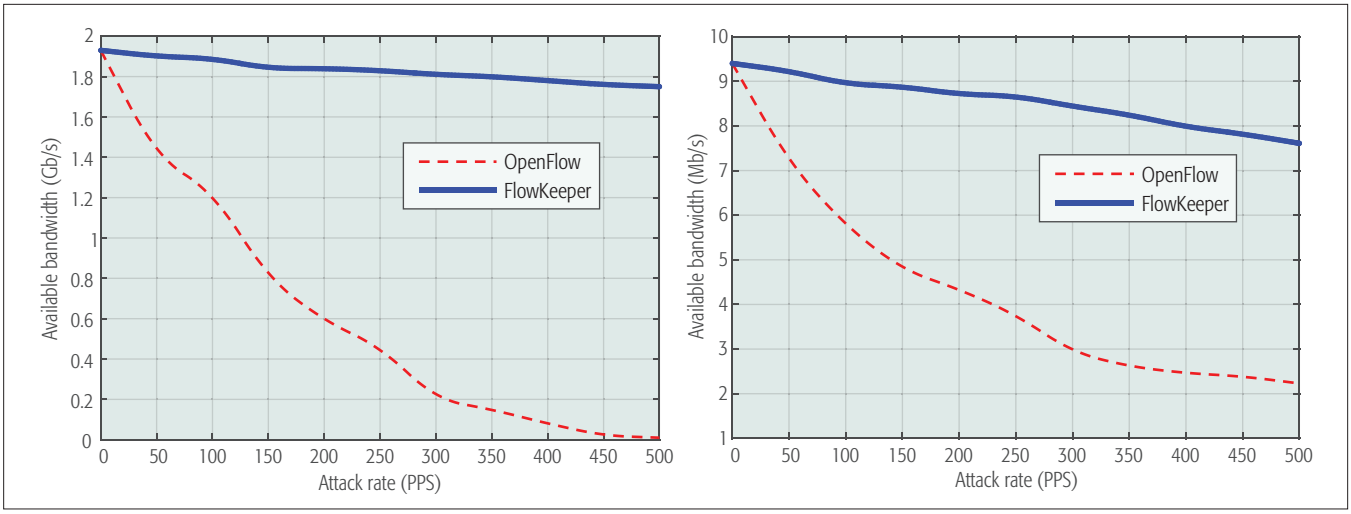

FIGURE 4. Available bandwidth under DoS attacks: a) sofware environment; b) hardware environment.

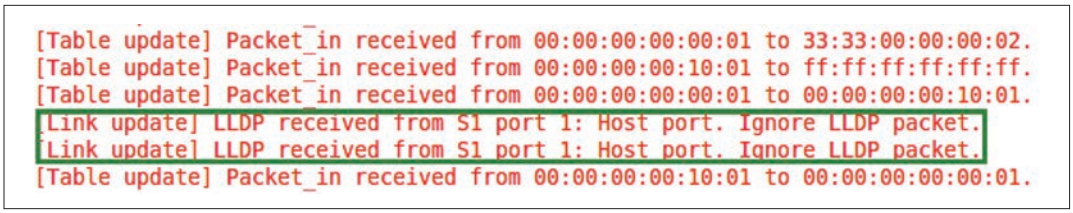

FIGURE 5. Ignore illegal LLDP packets to avoid topology poisoning attacks.

FlowKeeper preserves more than 80 percent bandwidth under DoS attacks and can avoid illegal topology updates by filtering out forged LLDP packets.

Acknowledgment

This article is partially supported by NSFC grants 61772446 and NSFC 61472365.

References

[1] N. McKeown *et al.*, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Commun. Review*, vol. 38, 2008, pp. 69–74.
[2] S. Jain *et al.*, "B4: Experience with a Globally-Deployed Software Defined WAN," *Proc. ACM SIGCOMM Computer Commun. Review*, vol. 43, 2013, pp. 3–14.
[3] R. Trivisonno *et al.*, "SDN-based 5G Mobile Networks: Architecture, Functions, Procedures and Backward Compatibility," *Trans. Emerging Telecommun. Technologies*, vol. 26, 2015, pp. 82–92.
[4] W. Yu "DSSS-Based Flow Marking Technique for Invisible Traceback," *Proc. IEEE Symposium on Security and Privacy (Oakland)*, 2007.
[5] S. Shin *et al.*, "AVANT-GUARD: Scalable and Vigilant Switch Flow Management in Software-Defined Networks," *Proc. ACM Conf. Computer & Communications Security (CCS)*, 2013.
[6] P. Zhang *et al.*, "On Denial of Service Attacks in Software Defined Networks," *IEEE Network*, vol. 30, 2016, pp. 28–33.
[7] S. Hong *et al.*, "Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures," *Proc. Network and Distributed System Security (NDSS)*, 2015.
[8] S. Shin and G. Gu, "Attacking Software-Defined Networks: A First Feasibility Study," *Proc. ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, 2013.
[9] J. Leng, "An Inference Attack Model for Flow Table Capacity and Usage: Exploiting the Vulnerability of Flow Table Overflow in Software-defined Network," arXiv preprint arXiv:1504.03095, 2015.
[10] R. Kloti, V. Kotronis, and P. Smith, "OpenFlow: A Security Analysis," *Proc. IEEE Int'l. Conf. Network Protocols (ICNP)*, 2013.
[11] J. Sonchack *et al.*, "Timing-Based Reconnaissance and Defense in Software-Defined Networks," *Proc. ACM Annual Conference on Computer Security Applications (ACSAC)*, 2016.

FlowKeeper enforces port control of the data plane and reduces the workload of the control plane by filtering out illegal packets. Experimental results show that FlowKeeper could be used to efficiently mitigate different kinds of attacks (i.e., DoS and topology poisoning attacks).

[12] H. Wang, L. Xu, and G. Gu, "FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks," *Proc. IEEE/IFIP Dependable Systems and Networks (DSN)*, 2015.
[13] S. Gao *et al.*, "FloodDefender: Protecting Data and Control Plane Resources under SDN-aimed DoS Attacks," *Proc. IEEE Int'l. Conf. Computer Communications (INFOCOM)*, 2017.
[14] V. N. Vapnik and V. Vapnik, *Statistical Learning Theory*, Wiley New York, 1998.
[15] Polaris xSwitch, "Polaris X10-24S," http://www.polarisdn.com/en/product/html/?80.html, 2017.

Biographies

Gao Shang received his B.S. degree from Hangzhou Dianzi University, China, in 2010, and his M.E. degree from Southeast University, China, in 2014. He is currently a Ph.D. candidate in the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His research interests include information security, network security, and software-defined networks.

Zecheng Li received his B.Sc. degree from Southeast University, Nanjing, China, in 2017. He is currently a Ph.D. candidate in Department of Computing, The Hong Kong Polytechnic University. His research interests include network security, SDN security, and NFV security.

Bin Xiao received the B.Sc and M.Sc degrees in electronics engineering from Fudan University, China, and a Ph.D. degree in computer science from the University of Texas at Dallas, USA. He is currently an associate professor in the Department of Computing, The Hong Kong Polytechnic University. His research interests include distributed wireless systems, network security, and software-defined networks (SDN). He has published more than 100 technical papers in top international journals and conferences.

Guiyi Wei received the Ph.D. degree from Zhejiang University in 2006, where he was advised by Cheung Kong Chair Professor Y. Zheng. He is currently a professor with the School of Computer Science and Information Engineering, Zhejiang Gongshang University. His research interests include wireless networks, mobile computing, cloud computing, social networks, and network security.