Combination Attacks and Defenses on SDN Topology Discovery

Dezhang Kong, *Graduate Student Member, IEEE*, Yi Shen, Xiang Chen, *Member, IEEE*, Qiumei Cheng, Hongyan Liu, Dong Zhang, *Member, IEEE*, Xuan Liu, *Senior Member, IEEE*, Shuangxi Chen, and Chunming Wu

***Abstract*— The topology discovery service in Software-Defined Networking (SDN) provides the controller with a global view of the substrate network topology, allowing for central management of the entire network. Unfortunately, emerging *topology attacks* can poison the network topology and result in unforeseeable disasters. Although researchers have made great efforts to mitigate this problem, security hazards still exist. In this paper, we propose *Invisible Assailant Attack* (IAA), the first combination topology attack capable of injecting and maintaining fake links even when 12 existing defense strategies are deployed simultaneously. IAA consists of 14 attack phases that apply multiple attack strategies. Attackers skillfully disguise the attack traffic in each phase so that it looks like normal network traffic, and perform these phases in a well-planned sequence, thereby bypassing existing defenses step by step. To mitigate this attack, we propose a *Route Path Verification* (RPV) mechanism that orchestrates multiple defense strategies to identify fake links. According to the experiments, RPV can successfully detect IAA with low overhead: its detection completes within 1 ms while its per-flow storage consumption is only a few KB.**

***Index Terms*— Software-defined networking (SDN), topology discovery, topology attacks and defenses.**

I. Introduction

SOFTWARE-DEFINED networking (SDN) is widely recognized as a revolutionary network paradigm that provides flexible, programmable, and holistic network management [17], [19], [20], [28], [48]. One of the fundamental differences between SDN and conventional networks is that the SDN controller maintains a global view of the entire network. When reacting to network events, the controller can use its global view to make network-wide optimal solutions, which significantly improves the quality of networking services [13], [18]. The topology discovery service plays a key role in establishing this global visibility. It collects metadata (e.g., the device information and link status) from the data plane and uses them to construct a global view of the substrate network topology in the control plane. Many network management applications, such as load balancing, packet forwarding, and network optimization, rely on the global information to accomplish their tasks [7], [35].

Unfortunately, if the topology discovery service provides poisoned topology information, the entire network may be affected or completely exposed to attackers. Given the importance of the topology discovery service, numerous attacks have been proposed [9], [29], [33], [34], [46], [47], [52]. For example, the link fabrication attack [25] is a popular topology attack that injects fake links into the network. It exploits the vulnerabilities of the Link Layer Discovery Protocol (LLDP), which is a crucial protocol for discovering internal links [25], [36], [44]. Attackers can leverage the fake link to cause extensive damage, such as manipulating the load balancing application to create a routing black hole [25] or injecting malicious packets to steal the identities of the network devices [22], [25], [30], [36]. To this end, researchers have revealed massive related vulnerabilities [11], [25], [30], [44] and have developed corresponding defenses to enhance the security of the network topology [11], [21], [22], [25], [27], [30], [31], [38], [43], [44], [45], [49].

However, the topology discovery service is still susceptible to potential threats due to the lack of work that studies combination attacks and their implications for SDN topology discovery. Implementing multiple security strategies to protect a system is pretty common in real situations [14], [15], [26]. In this context, attackers have to conduct combination attacks, i.e., those that combine multiple attack strategies and take complex actions, to invade the target system step by step [24], [42], [51]. However, existing topology attacks are proposed based on the assumption that the controller is directly exposed to attackers [11], [22], [25], [30], [43] or that only a single defense is protecting the controller [36], [44], [50]. These assumptions do not consider the scenarios where multiple defenses are deployed, resulting in relatively simple attack strategies and procedures of existing attacks. To the best of our knowledge, the feasibility of combination

Manuscript received 29 December 2021; revised 21 May 2022 and 29 July 2022; accepted 26 August 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor T. Qiu. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1804705, in part by the Science and Technology Development Funds of China under Grant 2021ZY1025, in part by the Key Research and Development Program of Zhejiang Province under Grant 2021C01036, and in part by the Fundamental Research Funds for the Central Universities (Zhejiang University NGICS Platform). *(Corresponding author: Chunming Wu.)*

Dezhang Kong, Yi Shen, Xiang Chen, Qiumei Cheng, Hongyan Liu, and Chunming Wu are with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310058, China (e-mail: kdz@zju.edu.cn; shenyizju@zju.edu.cn; wasdnsxchen@gmail.com; chengqiumei@zju.edu.cn; hyliu20@zju.edu.cn; wuchunming@zju.edu.cn).

Dong Zhang is with the College of Computer and Data Science, Fuzhou University, Fuzhou 350108, China (e-mail: zhangdong@fzu.edu.cn).

Xuan Liu is with the College of Information Engineering (College of Artificial Intelligence), Yangzhou University, Yangzhou 225127, China, and also with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: yusuf@yzu.edu.cn).

Shuangxi Chen is with the Jiaxing Vocational Technical College, Jiaxing 314036, China, also with the Jiaxing Key Laboratory of Industrial Internet, Jiaxing 314036, China, and also with the Polytechnic Institute, Zhejiang University, Hangzhou 310058, China (e-mail: abelchen@zju.edu.cn).

Digital Object Identifier 10.1109/TNET.2022.3203561

topology attacks and their destructiveness has not been studied.

This paper studies combination attacks and their implications for SDN topology discovery. Specifically, we propose *Invisible Assailant Attack* (IAA), the first combination topology attack that employs multiple attack strategies to inject and maintain fake links in the network. Different from existing attacks that exploit either an unprotected controller or a specific security mechanism, IAA is conducted under the assumption that 12 existing defense strategies are deployed *simultaneously*. IAA combines a link injection strategy with multiple circumvention strategies, and skillfully integrates them into a complete attack sequence that consists of 14 attack phases. In each attack phase, the packets generated by IAA are disguised to appear as the normal packets existing in the network. Besides, the 14 attack phases are conducted in a well-planned sequence to evade the 12 defenses step by step. In this manner, IAA can inject a fake link and ensure that the fake link does not generate any suspicious packets that may expose attackers.

According to our analyses, existing defenses have drawbacks that attackers can exploit to bypass detection. Besides, there is a lack of effective orchestrations among these defenses to compensate for their respective vulnerabilities. As a result, even though these defenses are piled up, IAA can still poison the controller's view. To this end, we propose the *Route Path Verification* (RPV), a security mechanism that orchestrates multiple defense strategies to defend against IAA. Firstly, we observe that normal packets can always be traced back to their source hosts along with the switches they passed, whereas the packets generated by IAA cannot. We refer to this feature as *the integrity of route paths*. Therefore, RPV validates this constraint for each flow reported to the controller. Secondly, attackers can force the controller to discard existing route paths originating from these hosts by forging host migrations [25], [44]. Therefore, RPV employs the host authentication strategy to validate each host strictly. A trustworthy route path must originate from an authenticated host, which ensures the security of the whole verification chain. Thirdly, since many attacks probe network status to determine the optimal attack opportunity [16], [44], RPV monitors the frequency of probing packets (e.g., ICMP echo requests [40]) to help detect potential attacks. The experiment results demonstrate that RPV can detect both IAA and eight existing topology attacks. Besides, RPV has a minimal overhead: it takes less than 1 ms to detect a fake link, and it requires only a few KB of storage to store the metadata for each flow.

The contributions of this paper are as follows:

- This paper is the first study of combination attacks and their implications for SDN topology discovery.
- This paper proposes the first combination topology attack and demonstrates its destructiveness when the 12 defense strategies are deployed simultaneously.
- A security mechanism that orchestrates multiple defense strategies is developed to detect combination topology attacks.
- The experiments demonstrate that the proposed security mechanism is capable of preventing combination attacks with low overhead.

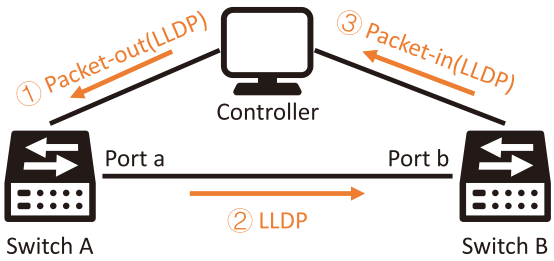


Fig. 1. The link discovery process using LLDP packets.

The rest of this paper is organized as follows: Section II exhibits the background of the topology discovery service in SDN and the motivation to study combination attacks in this area. Section III introduces IAA and related evaluation. In Section IV, a new defense is proposed and evaluated. The discussion of the limitation of this paper is shown in Section V. Section VI exhibits related works. Finally, we conclude our work in Section VII.

II. Background and Motivation

A. Background

In SDN networks, network management applications rely on the correct and timely topology information provided by the controller to realize their functions. To dynamically discover the changes in network topology, the controller deploys three services to track hosts, switches, and internal links, respectively.

The host discovery service is used to track the location of hosts. When a host first sends a packet into the network, the connected edge switch will report this packet to the controller due to the lack of corresponding flow rules. Then the controller records related metadata (e.g., MAC address) to represent this host uniquely and stores the host's location (i.e., the corresponding connected switch port). In this context, if the controller does not find any existing records of this host, it deems that this host is new. If the controller finds that related metadata has already existed, but the stored location does not match the new switch port, it assumes that the host moved to a new switch port and updates related records. In contrast, the switch discovery service is easy to achieve. When a new switch enters the network, it must establish a connection with the controller, by which the controller recognizes the existence and profile of the switch.

The link discovery service uses LLDP packets to monitor the internal links between switches. Figure 1 illustrates the discovery of the link between $Switch\ A$ and $Switch\ B$. The controller periodically sends out a Packet-out message that contains an LLDP packet to $Switch\ A$. This LLDP packet stores the unique ID of $Switch\ A$ and the port number of $port\ a$. When receiving this LLDP packet, $Switch\ A$ sends it out via $port\ a$ according to the instruction of the controller. This packet reaches $port\ b$ of $Switch\ B$ and is immediately reported to the controller via a Packet-in message, which also contains the unique ID of $Switch\ B$ and the port number of the in_port $port\ b$. After receiving the report from $Switch\ B$, the controller deems that there is a link between $Switch\ A$ and $Switch\ B$. The discovery of the reverse link is conducted in a similar manner.

TABLE I
DRAWBACKS OF EXISTING DEFENSE STRATEGIES

Defense	Mechanism	Drawbacks
Host Authentication [30], [44]	Require new hosts to provide correct private key to pass authentication before entering the network.	Fails when man-in-the-middle attack blocks the authentication traffic [36]
LLDP Authentication [25]	Calculate a hash value for each LLDP packet to detect malicious modification or forged LLDP packets.	Fails if attackers relay LLDP packets without modification [25]
Port-MAC Binding [30]	Detect unauthorized MAC-Port bindings.	Fails if attackers launch Port Probing Attack [44]
MAC-IP Binding [22], [30]	Detect unauthorized MAC-IP bindings.	Fails if attackers launch Port Probing Attack [44]
Jumbo Frames [43]	Send jumbo frames to the target link to expose fake links injected by hosts.	Fails if attackers configure the network interface card to enable larger frames
Latency Detection [44], [45]	Detect fake links by capturing abnormally high latencies.	1) Fails if attackers gradually injecting packets to raise the detection threshold [36] 2) Flaws in accurancy
Link Direction [22]	Require internal links to be bidirectional, thus unidirectional links are regarded as fake links.	Fails if attackers relay LLDP packets that come from opposite directions
Port Connectivity [22], [31]	A switch port is connected to at most one switch.	Fails if attackers inject fake links only on edge ports
Port Property [25]	Require that there is no first-hop host traffic on internal ports and no LLDP packets on edge ports.	Fails if attackers reset the network interface card when sending traffic [44]
Host Migration Condition [25]	Require migrations to meet the pre/post condition.	Fails if attackers launch Port Probing Attack [44]
Anomalous packets [44]	Recognize the frequent Port-Up and Port-Down events as the symbol of Port Amnesia Attack.	Fails if attackers silently relay LLDP packets [43]
SPV [11]	Induce malicious switches to report the probing packets, thereby identifying fake links.	Fails if attackers refuse to report the first few received packets [36]

By exploiting the host discovery service and the link discovery service, attackers are able to inject fake links or usurp the identity of a victim host. For example, attackers can forge LLDP packets to claim a nonexistent link [25] or relay LLDP packets between two disconnected switches via an out-of-band channel to fool the controller into believing a fake link [44]. Besides, they can make the controller believe that a victim host has moved to a new location by forging packets with the victim's addresses [30].

B. Motivation

Inadequate Assumptions of Attacks. Existing studies propose topology attacks based on simplistic assumptions: the network topology is unprotected, or only a single defense is protecting it. These studies do contribute to the discovery of valuable vulnerabilities at the early stage of the research. However, as related studies develop in depth, it is necessary to consider more practical scenarios. Precisely, in practice, administrators always deploy *multiple* defense strategies to protect their systems [8], [14], [15], [26], [32], which forces attackers to conduct combination attacks, i.e., those that combine different attack strategies and perform multiple actions, to invade the target step by step [24], [42], [51]. However, in the area of SDN topology security, none of the existing studies consider such practical cases where multiple defenses are protecting the controller, which results in relatively simple attack strategies and attack forms. As illustrated in Table II, nine attacks assume that the topology is directly exposed to attackers [25], [30], [36], [43], [44]. The other seven attacks assume that only one defense is protecting the network topology [36], [44], [50]. These studies do not consider the practical situation where multiple defenses are deployed. Note that although Marin *et al.* analyze four mainstream defenses, the analysis of each defense is still isolated [36].

TABLE II
ASSUMPTIONS OF EXISTING ATTACKS

Attack	Defenses Deployed
Host Location Hijacking [25]	No defense
Link Fabrication (forge packet) [25]	No defense
Link Fabrication (relay packet) [25]	No defense
Port Amnesia Attack [44]	TopoGuard [25]
Port Probing Attack [44]	TopoGuard [25]
IP Takeover [30]	No defense
Flow Poisoning [30]	No defense
Silent Relay Attack [43]	No defense
Attack 1 to link latency [36]	TopoGuard+ [44]
Attack 2 to link latency [36]	TopoGuard+ [44]
Attack to LLDP freshness [36]	No defense
Attack to low-level bindings [36]	SecureBinder [30]
Reverse loop [36]	No defense
Topology Freezing [36]	No defense
Attack to probing packet [36]	SPV [11]
Attack to LineSweep [36]	SPV [11]

For instance, Marin *et al.* illustrate that forged packets with random addresses can disable the latency-based defense [44] but ignore that these packets will be blocked by SecureBinder [30], which is one of the four defenses they study in the same paper.

The Uncertainty of Topology Security. Existing studies follow a typical research pattern: finding a new vulnerability (attack) and designing a corresponding defense to detect the proposed attack. Therefore, each existing attack has its exclusive "natural predator". In this context, the controller seemingly can patch all the vulnerabilities if it deploys all known defense strategies. However, in actuality, defenses usually have drawbacks that can be exploited by other existing

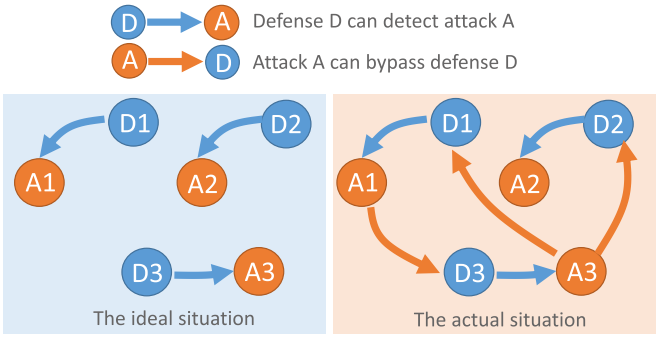


Fig. 2. The uncertainty of the topology security. In the actual situation, although each attack is prevented by a specific defense, the drawbacks of defenses break the ideal defense status and increase the uncertainty of the whole system, which is nonnegligible in a security-conscious system.

attacks, which break the ideal defense status and greatly exacerbates the uncertainty and complexity of the topology security.

Figure 2 shows an example to explain this situation. It depicts three attacks (A1, A2, A3) and three corresponding defenses (D1, D2, D3). In the ideal situation (the left part in Figure 2), the three defenses patch the corresponding vulnerabilities perfectly, so the controller will not suffer from any threat if it deploys the three defenses simultaneously. However, in the actual situation (the right part), defenses D1-D3 have vulnerabilities that A1 and A3 can exploit. Specifically, A1 can bypass D3, and A3 can bypass D1 and D2. In this context, even though A1 and A3 are known attacks, and corresponding defenses have already existed (i.e., D1 and D3), these drawbacks still break the ideal defense status and increase the uncertainty of the network security. For example, the unforeseen inconsistency of the execution order among different defenses and the conflict in the resource occupancy may greatly increase the feasibility of detection evasion in the presence of these vulnerabilities. Such potential risks are nonnegligible when building reliable and secure systems.

Unfortunately, the situation becomes much more complicated when it comes to SDN topology security. Compared to the three attacks and three defenses in Figure 2, there are 16 attacks and 12 defenses related to the SDN topology. According to the analytical results in Table I, the 12 defenses all have drawbacks that some existing attacks can exploit. These drawbacks cause an extremely complicated relationship between existing defenses and attacks, and greatly exacerbate the uncertainty of the topology security. We deem that such potential risks are nonnegligible for all the network management applications that rely on the correct topology information when multiple defenses are deployed. However, to the best of our knowledge, none of the existing studies investigate the security of the network topology in the presence of multiple defenses.

III. Invisible Assailant Attack

This section presents the first combination attack against topology discovery: *Invisible Assailant Attack* (IAA).

A. Attack Model

The network architecture is the OpenFlow-based architecture [37], containing OpenFlow switches and an OpenFlow controller that utilizes LLDP packets to discover internal links. The controller uses independent connections to communicate with OF switches, allowing multiple hosts to connect with one switch port.

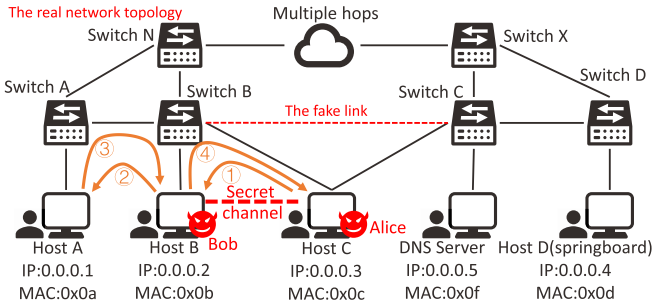
In terms of security, the attack model consists of five basic assumptions. 1) The SDN controller and OpenFlow switches are well-protected. 2) The communication channels between the controller and switches, as well as the management interfaces between the controller and administrators, do not suffer from any security threat. 3) Attackers only compromise hosts and do not have access to switches or the controller. Existing studies also adopted the above assumptions [25], [30], [36], [43], [44]. Besides, to get closer to practical situations, the attack model further limits the capability of the attackers by adding the following two assumptions. 4) It assumes that the 12 defense strategies protect the controller rather than that one defense is deployed at most. 5) The hosts launching attacks are *compromised* instead of innately malicious, which increases the difficulty of the attack but makes the model more in line with actual circumstances. This is because "malicious host" means that attackers can arbitrarily configure the host without considering any consequence. In contrast, "compromised host" means that attackers are external invaders, so they must avoid being noticed by the host's real user when launching attacks. For example, port amnesia attacks require the malicious host to frequently close and restart its network interface card [44], which causes the communication of this host to be interrupted frequently. If this host is a compromised host, its real user will soon find this anomaly, then notice that his machine is compromised.

B. Attack Goal

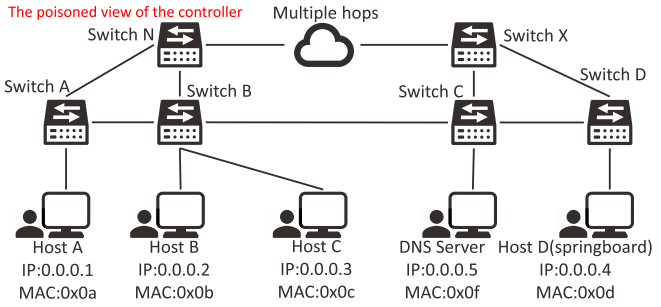
IAA is to inject and maintain a fake internal link even when the 12 defense strategies are deployed. As an example, Figure 3a depicts the actual topology of the network, while Figure 3b depicts the "topology" that IAA wants the controller to believe, i.e., the poisoned view of the controller. In this topology, IAA injects and maintains a fake link between $Switch\ B$ and $Switch\ C$. A fake link enables attackers to further penetrate the network since they are able to sniff, intercept, and modify the normal packets passing through the fake link.

C. Attack Overview

IAA exploits the vulnerability of the link discovery service to inject a fake link, which is similar to existing studies. Differently, IAA is able to hide its attack packets by using the normal packets that exist in the network, thereby bypassing defenses. Figure 4 describes the attack strategies of IAA, which are performed by two compromised hosts. Specifically, one attacker (Alice, who compromised $Host\ C$) injects the fake link, and the other attacker (Bob, who compromised $Host\ B$) assists in maintaining the fake link, i.e., disguises the packets generated by Alice to evade detection.



(a) The actual topology of the network. ①⇒②⇒③⇒④ illustrates the communication process between Host C and Host A.



(b) The poisoned view of the controller.

Fig. 3. The illustration of IAA. Note that each switch has an independent link with the controller. For simplification, these links are not exhibited.

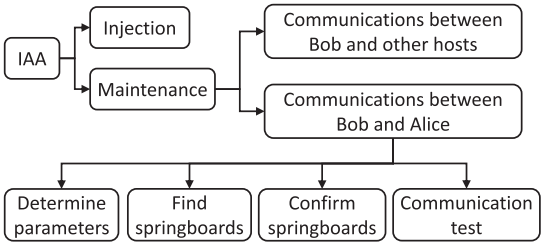


Fig. 4. The attack logic of IAA.

As illustrated in Figure 3, Alice injects a fake link between $Switch\ B$ and $Switch\ C$ by relaying LLDP packets between them, which is also used in previous attacks [25], [43], [44]. However, $Host\ C$ will inevitably send first-hop host packets when communicating with other hosts. If these packets are captured on an internal port by the controller, the related defenses will raise alerts [25]. Since it is scarcely possible for attackers to predict what kinds of packets may expose the fake link, they cannot modify their attack strategies accordingly, which explains why existing attacks fail to maintain fake links for a long time. For example, the ICMP queries generated by the compromised host can expose the fake link [25]. However, since ICMP queries are widely used in normal scenarios, attackers will not drop these packets for no reason if they do not know the concrete defense strategy. Even if attackers recognize that ICMP packets will expose the fake link, the periodically broadcasted ARP packets, as well as other packets that are common in network communications, will also be used to identify fake links. Therefore, it is extremely difficult for attackers to predict which kinds of packets are utilized as the detection metric. However, existing attacks do not solve this problem, which explains why their fake links are easy to detect.

Therefore, the difficulty lies in how to maintain the fake link after the injection. To this end, Bob configures $Host\ B$ to serve as a proxy to relay the communication of $Host\ C$, so $Host\ C$ only needs to communicate with $Host\ B$. As shown in Figure 4, in this context, the communication between Bob and other hosts will not attract any suspicion. In contrast, the communication between Bob and Alice needs to be carefully disguised as normal network traffic that has already existed in the network, thereby evading detection.

To achieve the secret communication between Alice and Bob, IAA needs to conduct four steps. Initially, before launching the attack, two attackers must predetermine critical parameters of the secret communication channel to make them able to distinguish their communication packets from other normal network packets. Secondly, after successfully injecting the fake link, Alice needs to seek a normal host as the springboard, whose packets will be unconsciously utilized to disguise the communication between attackers. Thirdly, Alice informs Bob to confirm whether the found candidate satisfies the requirements of a springboard. Finally, Alice and Bob can practically leverage the involuntary assistance of the springboard to test whether they can disguise their communication as normal packets.

During the whole procedure, IAA also needs to take additional actions to deal with diverse defense strategies. The detailed procedure is presented in Section III-D.

D. Detailed Procedure

The entire attack sequence of IAA is illustrated in Figure 5, which clearly describes how IAA injects and maintains a fake link while bypassing different defense strategies.

Preparation. Before launching IAA, attackers should make some preparations. First, since the communication traffic between Alice and Bob will be disguised as normal network traffic, the attackers must negotiate a way to distinguish these "disguised packets" from other network packets. To this end, at the outset of the attack (phase 1,2 in Figure 5), Alice and Bob determine the parameters (e.g., protocol and socket number) of the disguised packets. Second, as the proxy that relays the communication packets of $Host\ C$, Bob must modify the source addresses of the received packets from $Host\ C$. Specifically, Bob uses $Host\ C$'s addresses as the source addresses when relaying these packets, just as $Host\ C$ has moved to the switch port to which $Host\ B$ is connected. This is because the traffic already addressed to $Host\ C$ may not reach its destination if Bob uses $Host\ B$'s addresses. Therefore, if Bob wants to use $Host\ C$'s addresses, he must bypass related defenses, including the host authentication [30] and the verification of the host migration [25]. To this end, Alice triggers a Port_Down event to make it appear as if $Host\ C$ is disconnected from the current location (phase 3). This action can be achieved by configuring the network interface card on $Host\ C$ to make it close temporarily, which has already been used by the Port Amnesia Attack [44]. Then, Bob deceives the controller into believing that $Host\ C$ has migrated to the new switch port by sending a forged packet whose source addresses are those of $Host\ C$ (phase 4). If the

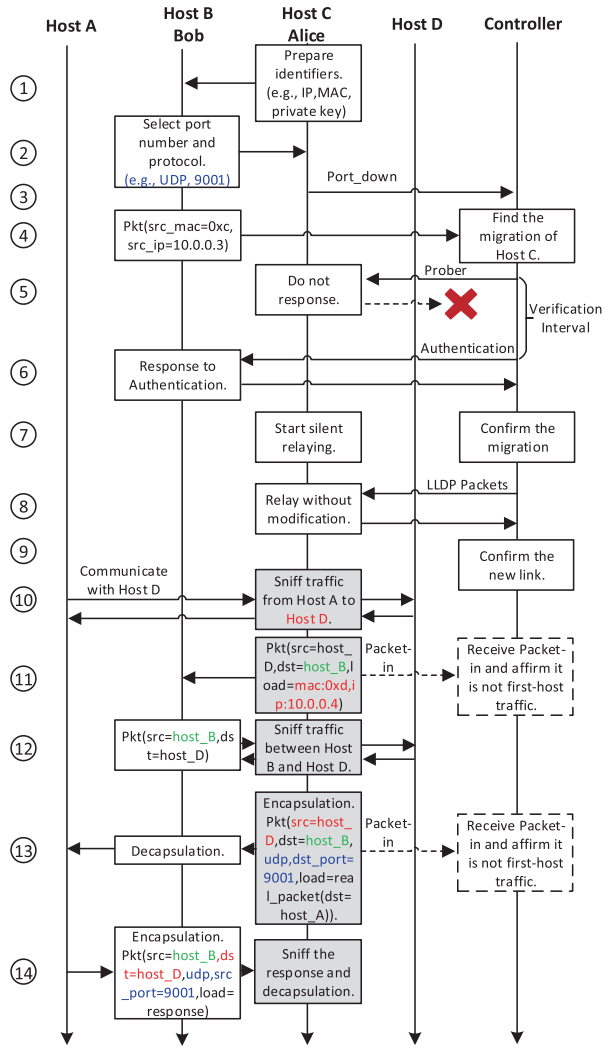


Fig. 5. The timeline of IAA. The shadowed box indicates that a background program is silently relaying packets. The blue font marks the pre-negotiated parameters of the disguised packets. Red text is the springboard and his addresses. Green font means it can be either Alice's address or Bob's address. The dotted-line box indicates that these Packet-In messages will not cause alarm.

controller receives this packet, it may conduct validations, including querying the previous location of $Host\ C$ [25] and authenticating Bob [30]. To evade the former detection, Alice does not answer any queries (phase 5). To evade the latter detection, Alice and Bob can exchange necessary data (e.g., certificates and private keys) in phase 1 to pass the authentication.

Inject a fake link. After the "migration", Alice starts injecting a fake link by relaying LLDP packets (phase 7). The controller may verify the integrity of the LLDP packet in this process (phase 8). However, Alice does not modify LLDP packets when relaying them, thereby evading the detection from the LLDP authentication. After injecting the fake link, Alice should temporarily prevent $Host\ C$ from generating packets before the secret communication channel between Alice and Bob is established. This is because any self-generated packets on the fake link may expose the attack. However, it is impossible to eternally block the normal communication of $Host\ C$, which will soon attract the attention of the real user of $Host\ C$. Therefore, the attackers need to find a way to disguise their communication. Specifically, they should determine (1) how to enable Alice to secretly send packets to Bob and (2) how to ensure that Alice will receive the traffic from Bob.

Maintain the fake link: Traffic from Alice to Bob. When sending packets to Bob, Alice cannot use $Host\ C$'s addresses directly or use forged source addresses that do not exist in the network. These addresses will be reported to the controller due to the lack of related flow rules, thereby immediately exposing the attack. Therefore, IAA proposes a novel method that uses the existing network traffic passing through the route path between Alice and Bob to disguise the communication between the two attackers. In this way, the attacker's communication packets will not trigger `Packet-In` messages, let alone be detected by defenses. To find such traffic, Alice should seek a "springboard", i.e., a host whose communication with $Host\ B$ is routed through the fake link. For example, in Figure 3a, when $Host\ B$ communicates with $Host\ D$, their traffic will be routed through the fake link, so $Host\ D$ is what Alice seeks. To find springboards, Alice sniffs the packets on the fake link and filters those from the opposite side of Bob. For example, in Figure 3a, when $Host\ A$ communicates with $Host\ D$, Alice captures them and finds that $Host\ D$ is on the opposite side of Bob (phase 10). Alice then informs Bob to test whether $Host\ D$ is helpful (phase 11). Note that the springboard is not predetermined. It is found after Alice successfully injects the fake link. Alice uses $Host\ D$'s addresses as the source addresses when informing Bob. After receiving the notification from Alice, Bob sends a probing packet to $Host\ D$, then $Host\ D$ responds to this packet. If Alice captures this bidirectional communication, $Host\ D$ is confirmed as a springboard (phase 12). Afterward, all the packets sent from Alice to Bob will be disguised as those sent from $Host\ D$ to Bob. Specifically, Alice encapsulates the packets generated by $Host\ C$ into disguised packets, whose protocol or the socket number follows those negotiated in phase 1,2. When Bob receives the packets from $Host\ D$ but finds that the protocol or socket number matches the specified protocol or socket number, he will understand that they are from Alice.

Maintain the fake link: Traffic from Bob to Alice. From the perspective of the controller, $Host\ C$ has already moved to $Switch\ B$ in phase 7. Therefore, all the packets addressed to $Host\ C$ will be finally routed to $Switch\ B$. In this context, Alice has to rely on the fake link to receive packets indirectly. Specifically, when sending packets to Alice, Bob uses $Host\ D$ as the destination. These packets will be routed to the fake link and then sniffed by Alice. Similarly, Alice identifies these packets based on their addresses and the specified protocol or socket number. She intercepts these packets and decapsulates the real packets. The decapsulated packets will be delivered to the network stack of $Host\ C$, thereby maintaining its normal communication.

Flow rules stored in switches may be evicted due to table overflow or timeouts, which indicates that the communication packets between Alice and Bob may still trigger `Packet-In` messages. However, to the best of our knowledge, existing

TABLE III
NOTATION OF ANOMALIES

Symbol	Discription
WPM	Wrong Port-MAC Binding.
WMI	Wrong MAC-IP Binding.
FHT	First-hop Host Traffic on internal ports.
AFSP	Abnormal Frequency of Specific Packets.
AHL	Abnormally High Latency of new internal link.
FHA	Failure of Host Authentication.
AVNC	Anomaly in Victim's Normal Communication (e.g., human-perceivable high latencies).
FPA	Failure of Packet (LLDP) Authentication.
FSHMC	Fail to Satisfy the Host Migration Condition.

defenses raise alerts only if first-hop host packets are captured on internal switch ports [25]. Since the communication packets between Alice and Bob are not first-hop host packets, they will not be considered malicious by existing defenses. Besides, to reduce the risk of exposure, Bob can keep sending probing packets to $Host\ D$, thereby ensuring that related flow rules stay in switches.

E. Analysis

This section makes a detailed analysis of IAA to illustrate how it circumvents existing defense strategies. The analytical results in Table IV demonstrate that each existing attack has at least one detectable anomalous signature, whereas IAA has none of them. The notation of the anomalies in Table IV is shown in Table III.

Bypass authentication. IAA relays rather than crafts LLDP packets, thus bypassing the LLDP authentication easily. Host authentication relies on certificates to verify newly joined hosts. As shown in Figure 5, it works when Bob takes over the identity of $Host\ C$ (phase 6). However, Alice has already sent the necessary data (e.g., the private key and the certificate) to Bob, allowing him to pass the authentication successfully.

Bypass anomaly-based defense strategies. The anomalous packets generated by attackers are widely used to detect corresponding attacks. According to our analysis, 15 existing attacks inevitably generate abnormal packets, while IAA does not have any of them. The results are shown in Table IV. The detailed analysis is illustrated as follows. There is no abnormal packet generated in the attack phases 1-2, during which two attackers just communicate normally. Phases 3-5 show a complete host migration process, which satisfies the pre/post condition of a normal migration [25]. In phase 6, the controller authenticates $Host\ C$ to confirm its new position. Since Alice has provided Bob with the necessary data to pass authentication, the controller will not detect any anomalies. During phases 7-9, Alice silently relays packets to inject a fake link, which does not generate any traffic that may trigger `Packet-In` messages. Besides, Alice can deploy powerful tools to greatly reduce the latency of relaying packets, which is demonstrated in Section III-H. Phase 10 shows the normal communication between $Host\ A$ and $Host\ D$, which will not cause any alerts. In phase 11, Alice generates a packet that uses $Host\ D$'s addresses as the source addresses. This packet might cause a Table_Miss event on the internal port when the controller applies fine-grained definitions of flows. However, to the best of our knowledge, only TopoGuard explicitly verifies the packets captured on internal ports, but it will not treat this packet as malicious. This is because TableGuard only recognizes first-hop host packets on internal links as abnormal, while the packet forged by Alice is not a first-hop host packet. Phase 12 involves a legitimate communication between $Host\ B$ and $Host\ D$, which cannot be considered malicious. In phase 13, Alice generates a packet that uses $Host\ D$'s addresses as the source addresses again. Similar to phase 11, TopoGuard will not treat it as an anomaly. Phase 14 shows a normal communication between $Host\ B$ and $Host\ D$, which does not cause any anomalies. Therefore, the entire process of IAA does not violate the principles of existing defenses.

Bypass proactive defense strategies SPV is developed for detecting fake links injected by malicious switches. It sends forged packets to the suspicious link to trick the malicious switch into sending `Packet-In` messages. However, IAA is carried out by compromised hosts that cannot generate `Packet-In` messages, so SPV cannot detect it. SRD uses jumbo frames larger than 1500 Bytes to detect malicious hosts. This is because the network interface cards of hosts will drop such frames by default. However, attackers can configure the network interface card to support a large MTU (Maximum Transmission Unit), thus making SRD fail.

F. Implementation

Experiment Setup. IAA is conducted in an emulated SDN environment. Mininet 2.2.2 [4] is used for creating topology, and Ryu 4.34 is used as the controller. VMware Workstation Pro 15.5.0 is used in the environment, which is installed on a Windows 10 machine with an Intel Core i7-8700 processor and 16GB of memory. The virtual machines (VMs), on which we install the controller and launch attacks, run Ubuntu 18.04 operating system and have 2GB of memory and 2 CPU cores. The `ifconfig` command is used to configure the attacker's MTU to enable at most 5000-Bytes jumbo frames. `Iperf` [2] is used to test link bandwidth, and the `ping` command is used to record the link Round-Trip Time (RTT).

Implementation of existing defenses. To evaluate the destructiveness of IAA, we deploy eight existing defense strategies on the controller. The other four defense strategies that are obviously unable to detect IAA are abandoned. Specifically, SPV is an abandoned defense because it is developed to detect malicious switches. The LLDP authentication is not deployed either because IAA relays LLDP packets without modification. The Port-MAC binding strategy is not suitable for SDN in which host migration is common, so it is not deployed. Besides, those that rely on specific packets (i.e., frequent Port_Up and Port_Down events) are also given up because IAA does not generate these packets. We simplify some of the eight deployed defenses while ensuring that their detection effects are not affected. Specifically, to simplify the host authentication, hosts need to provide a correct password to enter the network. Besides, static MAC-IP binding strategy

TABLE IV
COMPARISON OF INVISIBLE ASSAILANT ATTACK WITH EXISTING TOPOLOGY ATTACKS ON ANOMALY FEATURES

Attack	WPM	WMI	FHT	AFSP	AHL	FHA	AVNC	FPA	FSHMC
Host Location Hijacking [25]	✔	✔	×	×	×	✔	×	×	✔
Link Fabrication (Forge Packet) [25]	×	×	✔	×	✔	×	×	✔	×
Link Fabrication (Relay Packet) [25]	×	×	✔	×	✔	×	×	×	×
Port Amnesia Attack [44]	×	×	×	✔	✔	×	✔	×	×
Port Probing Attack [44]	✔	✔	×	✔	×	✔	×	×	×
IP Takeover [30]	×	✔	×	✔	×	✔	×	×	×
Flow Poisoning [30]	✔	×	×	×	×	✔	×	×	×
Silent Relay Attack [43]	×	×	×	×	✔	×	✔	×	×
Attack 1 to link latency [36]	✔	✔	×	×	×	×	×	×	×
Attack 2 to link latency [36]	✔	✔	×	×	×	×	×	×	×
Attack to LLDP freshness [36]	×	×	✔	×	×	×	×	✔	×
Attack to low-level bindings [36]	×	×	×	×	×	×	×	×	✔
Reverse loop [36]	×	×	×	×	×	×	×	✔	×
Attack to probing packet [36]	×	×	×	×	✔	×	×	×	×
Attack to LineSweep algorithm [36]	×	×	×	×	✔	×	×	×	×
Invisible Assailant Attack	×	×	×	×	×	×	×	×	×

TABLE V
PERFORMANCES OF EXISTING DEFENSE STRATEGIES WHEN DEFENDING AGAINST IAA

Defense Strategy	Description	Simplification	Result
Host Authentication [30], [44]	Require new host to provide correct private key.	Use password	No alert
MAC-IP Binding [22], [30]	Detect unauthorized MAC-IP binding on edge ports.	Use static binding	No alert
Jumbo Frames [43]	Send jumbo frames to the target link.	None	No alert
Latency Detection [44], [45]	Detect abnormally high latencies of internal links.	None	0.46% alerts
Link Direction [22]	Require internal links to be bidirectiona.l	None	No alert
Port Connectivity [22], [31]	A switch port is connected to at most one switch.	None	No alert
Port Property [25]	Require that there is no first-hop host traffic on internal ports and no LLDP packets on edge ports.	None	No alert
Condition of Host Migration [25], [27]	Require migrations to meet the pre/post condition.	None	No alert

is used as the succedaneum of MAC-IP binding strategy. The deployed defenses, corresponding simplifications, and detection results are shown in Table V.

Conduction of IAA. When disguising the communication between Alice and Bob, only the ICMP, ARP, UDP, and TCP packets are disguised for simplification. This is because the goal of this paper is to demonstrate the feasibility of the attack strategy that attackers can avoid triggering suspicious `Packet-In` messages by disguising their packets as existing network packets. Therefore, it is not the purpose of this paper to disguise all types of packets.

G. Effectiveness Evaluation

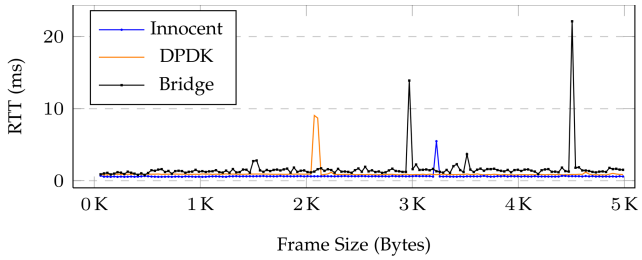
The experiments demonstrate that IAA is successfully conducted under the protection of the eight defense strategies in Table V. The controller accepts the fake link and allows traffic to pass through it. The detection results of the eight defenses are shown in the last column in Table V: seven defenses failed to raise an alert during the attack process. However, the latency-based defense raised alerts infrequently, which uses a straightforward application of the interquartile range (IQR) to find outliers [44]. Specifically, the latency-based defense raised six alerts during the propagation of 1139 LLDP packets. To further investigate the performance of the latency-based defense, three different links are supplemented and evaluated in the network: a fake link forged by Data Plane Development Kit (DPDK) [1], a fake link forged by the network bridge, and a real internal link. The detection results of the latency-based defense in Table VI are surprising: the abnormal latencies of the real internal link account for 0.46% of the total, which is extremely close to the link forged by DPDK (0.53%) and the link forged by network bridge (0.44%). Therefore, the outliers in the link latencies are not accurate enough to prove the existence of fake links.

TABLE VI
EVALUATION RESULTS OF INTERNAL LINKS

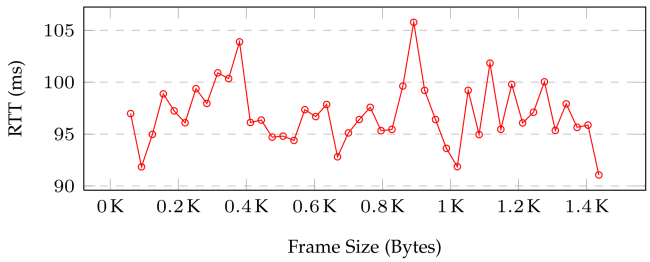
Link Type	DPDK	Bridge	Real link
Anomaly Percentage (%)	0.53	0.44	0.46
Average Latency (ms)	0.98	1.39	0.62
TCP Bandwidth (Mbps)	96.8	95.9	97.8
UDP Bandwidth (Mbps)	97.0	97.1	98.1

H. Stealthiness Evaluation

The performance of the fake link dramatically affects the success of IAA. For example, if the fake link exhibits abnormally high delays or low bandwidth, IAA will be detected easily. Besides, the real user of the compromised host is



(a) Latencies of a real link and two forged links with different frame sizes.



(b) Latencies of the secret communication channel with different frame sizes.

Fig. 6. Evaluation of fake links and secret channels.

the other potential "threat" that can detect the attacker. For instance, in Figure 3a, if $Host\ C$'s normal communication suffers from a considerable delay or even fails due to IAA, its real user will soon notice the anomaly of his machine. Therefore, there are three critical characteristics that can expose IAA according to our analysis: the latency of the fake link, the bandwidth of the fake link, and the latency of $Host\ C$'s normal communications. To this end, the fake link $<Switch\ B-Host\ C-Switch\ C>$ in Figure 3a is evaluated.

Latencies of fake links. Figure 6a depicts the latencies of three different links: a genuine link, a link forged by DPDK, and a link forged by the network bridge. In general, the latency of the bridge-forged link is higher and exhibits more jitters than the two other links. However, as shown in Table VI, their average latencies are still quite close: 1.39 ms (bridge-forged), 0.62 ms (genuine), and 0.98 ms (DPDK-forged). The differences between the real link and the two fake links are too tiny to identify (0.77 ms between the real link and the bridge-forged link, 0.36 ms between the real link and the DPDK-forged link). Therefore, IAA can achieve the low latency of the fake link, thereby evading the latency-based defense.

In addition, the latency of the real link at different congestion rates is also evaluated, whose results demonstrate that the latency-based defense has error detection performances that cannot be ignored even if no attackers are injecting packets. Figure 7 illustrates the experiment results on a 100Mbps internal link. Specifically, when the congestion rate is below roughly 40% (i.e., less than 40% of bandwidth is consumed), latencies of the real link are predominantly distributed over a short range. When the congestion rate exceeds approximately 50%, the overall distribution gradually broadens from 0 ms to 50 ms. Since extreme congestion is uncommon in real-world scenarios, only the latencies when the congestion rate is less than 20% are counted. In this range, the average latency is 0.707 ms, but those exceeding 1 ms still account for 18.7%,

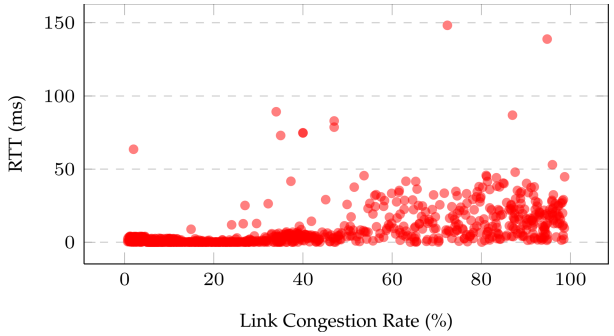


Fig. 7. Link latency at different congestion rate.

and those exceeding 2 ms account for 11.9%. Therefore, if the threshold is too low (e.g., less than 2 ms), many genuine latencies (11.9%) will be considered anomalous. However, if the threshold is set higher, the controller will not detect the existence of fake links. The results in Table VI and Figure 7 indicate that the average latency of fake links (1.39 ms at most) is extremely close to that of the genuine link (0.62 ms in Table VI and 0.707 ms in Figure 7). Therefore, if the threshold is set, for example, to more than 2 ms, the latency-based defense will not notice the malicious link at all. In this context, the latency-based defense cannot become a reliable security mechanism even if no attackers are poisoning it.

Bandwidths of fake links. The bandwidth of the three links is originally set to 100 Mbps. As shown in Table VI, the real link achieves the highest practical bandwidth (about 98 Mbps) in both TCP and UDP tests, while the two fake links are not far behind. The bridge-forged link achieves the lowest TCP bandwidth, but it still reaches nearly 96 Mbps. In terms of the UDP test, both DPDK-forged and bridge-forged links have similar high bandwidths (97 Mbps). Therefore, the differences in bandwidth are too small to be used as a reliable indicator of fake links.

The latencies of communication. The communication of the compromised host can also expose attackers. For example, the user of $Host\ C$ may believe that an intrusion has occurred if he finds that the host cannot access the Internet or that the response time of webpages suddenly increases (this might happen during steps 3-12 in Figure 5). In other words, if attackers can keep the communication latency of the compromised host within a **human-imperceptible** range, they will not be noticed by benign users. Figure 6b illustrates the latencies of the compromised host. The tested path is $<Host\ C-Switch\ B-Host\ B-Switch\ B-Switch\ A-Host\ A>$ in Figure 3a. It is obvious that latencies vary between 90 ms and 110 ms, which is significantly higher than the fake links. This is because the communication of the compromised host relies on Alice and Bob, who introduce extra network stack travels and the en/decapsulation of packets to evade detection. However, 110 ms is still within a human-imperceptible range, which means that the user of the compromised host cannot notice attackers.

I. Further Penetration of the Network

Since attackers are capable of sniffing, intercepting, and modifying the normal traffic passing through the fake link,

IAA enables attackers to cause larger damage to the network. We demonstrate it by launching man-in-the-middle attacks on the fake link. Specifically, Alice sniffs kinds of packets passing through the fake link, such as ARP, ICMP, TCP, and UDP packets. Alice intercepts the ARP responses and modifies them to trick victim hosts into establishing connections with Bob instead of their real destinations. Besides, after we deployed a DNS server in the network, Alice even captured the DNS responses that contained a mapping between IP address and domain name. In this context, Alice can even inject forged DNS responses to trick victims into trusting a malicious website, thereby stealing more useful information such as user accounts and passwords. We deem that there are more practical attack cases by using a fake link, which can cause more serious damage to the network.

IV. Route Path Verification

In this section, we design the Route Path Verification (RPV) mechanism, a new security mechanism that detects potential attackers by verifying the integrity of route paths. The experiment results indicate that this mechanism can effectively detect IAA.

A. Overview

RPV is developed based on a novel observation: when a normal packet is reported to the controller due to Table_Miss, the controller can always trace back to the source host that generated this packet. For example, if a new flow f is reported to the controller by $switch\ n$, the controller can find its upstream hop ($switch\ n-1$) according to the in_port on $switch\ n$. Then, the controller can check whether f really reached $switch\ n-1$ by querying the flow rules stored in it. If the corresponding flow rule exists, the controller then traces to $switch\ n-2$ according to the in_port. This process repeats until the controller traces back to the source host of f. However, this phenomenon does not apply to the malicious packets injected by IAA. For example, the packets in steps 11 and 13 in Figure 5 do not have a complete route path. Even though these packets are not first-hop host packets and may not trigger `Packet-In` messages, they can still expose the attack if the controller tries to trace to the source host.

We refer to this characteristic as *the integrity of route paths*. Based on this observation, RPV, a security mechanism that detects attacks by verifying the integrity of route paths, is developed. RPV collects the topology information and flow information from the data plane, and uses them to reconstruct the route path for each flow on the controller. When a new packet is reported to the controller due to Table_Miss, RPV verifies the integrity of the route path to validate this packet. If the packet passes the verification, RPV records it and the current switch to further construct the route path. When a host leaves the network or migrates to a new switch, RPV will delete all the route paths originating from this host.

To further enhance the effectiveness of RPV, there are two cases that need to be considered. First, the migration of a host can force the controller to discard previous route paths that originate from this host. If malicious hosts create fake migrations, they may bypass the verification of the route path. To this end, RPV authenticates hosts when they enter the network. Only if a host is authenticated can RPV start constructing route paths that originate from this host. Therefore, the verification of route paths and the authentication of the source host collectively make up a complete verification chain. The security of the chain relies on the security of its root, i.e., the authentication of hosts. Since host authentication is demonstrated to be trustable [30], the whole verification chain is secure. Second, if (1) Bob keeps sending probing packets to $Host\ D$ to force related flow rules to stay in switches, and (2) Alice directly uses these packets to disguise the communication between attackers, IAA can significantly reduce the possibility of triggering `Packet-In` messages. Considering that probing packets help not only IAA but also other different types of attacks [16], [44], [54], RPV does not issue flow rules for probing packets when they are reported to the controller. Therefore, the disguised packets generated by IAA will inevitably be reported to the controller and then expose the attackers. Specifically, RPV focuses on four widely-used types of probing packets: ICMP echo requests [40], TCP SYN packets [41], ICMP timestamp requests [40], and ARP requests [39]. Note that stopping generating flow rules for these probing packets does not degrade the network performance. This is because these packets are "single-use" in normal circumstances.

B. Workflow

Since the work on secure identifier binding in SDNs [30] has studied the authentication of hosts, we focus on the verification of route paths. When RPV receives a new `Packet-In` message, its actions fall into the verification and recording phase. In the verification phase, RPV verifies the integrity of the route path of the newly-reported flow. If no anomaly is detected, RPV enters the recording phase and saves the associated metadata for further verification. Algorithm 1 detailedly describes the workflow of RPV. In algorithm 1, msg represents the reported packet that triggers a `Table-Miss` event, $links$ records all internal links in the network, $flows$ is a collection of the recorded flows, and $switches$ represents the existing switches.

Verification Phase: RPV uses the 5-tuple, i.e., the source/destination MAC addresses, the source/destination IP addresses, and the protocol, to identify a flow uniquely. The definition of flow is adjustable in different scenarios, which does not influence the effectiveness of RPV. When a new flow f is reported to the controller as a result of Table_Miss, RPV turns into the verification phase. It extracts the 5-tuple to uniquely identify f, then records switch n and corresponding in_port that receives f (line 1). If this in_port is an internal port (i.e., a switch port that is connected to another switch), RPV searches for the upstream port and the upstream $switch\ n-1$ (lines 3-6). It checks whether f has passed through the upstream port (line 10) by querying whether the upstream port has the record of f's 5-tuple. If so, RPV can confirm that f is not a malicious packet (line 11). RPV does not need to verify the integrity of earlier route paths $(n-2, n-1, \ldots, 0)$, which

Algorithm 1 Route Path Verification

Input: msg, $links$, $flows$, $switches$
Output: mal

```
1: metadata = GetMetadata(msg)
2: if msg.port is INTERNAL then
3:   for link in links do
4:     if msg.port == link.des then
5:       up_port = link.src
6:       upstream_switch = GetSwitch(up_port)
7:       break
8:     end if
9:   end for
10:  if metadata in flows[upstream_switch][up_port]
     then
11:    mal = FALSE
12:  else
13:    mal = TRUE
14:  end if
15: end if
16: if msg.port is EDGE then
17:   hosts = GetHosts(msg.port)
18:   if msg.src_mac in hosts then
19:     if mgs.src_mac is not authenticated then
20:       mal = Authenticate(msg.mac)
21:     else
22:       mal = FALSE
23:     end if
24:   else
25:     mal = TRUE
26:   end if
27: end if
28: if mal == FALSE then
29:   current_switch = GetSwitch(msg.port)
30:   flows[current_switch][msg.port].add(metadata)
31:   out_port = CalculateOutPort(metadata)
32:   if out_port == BROADCAST then
33:     for port in current_switch.ports do
34:       flows[current_switch][port].add(metadata)
35:     end for
36:   else
37:     flows[current_switch][out_port].add(metadata)
38:   end if
39: end if
40: return mal
```

were already verified by the time f arrived at $switch$ $n-1$. If the in_port receiving f is an edge port, i.e., a port that is connected to hosts, RPV pulls the records of the hosts connected to this port and extracts the source MAC address of f (lines 16-17). If this MAC address is found in the records, RPV can affirm that f is a benign flow. If none of these hosts possess this MAC address, RPV authenticates the identity of this host to detect potential impersonator hosts (lines 19-20).

The records of route paths are deleted when the corresponding source hosts leave the network, but the records of flow rules do not get influenced when related flow rules in switches are evicted. This is because if RPV removes related records synchronously when the flow rules in switches are deleted due to timeouts or table overflow, RPV may mistakenly treat normal paths as malicious paths. However, for probing packets, RPV deletes their route paths when they reach the destination. For instance, if a probing packet p is sent from host a to host z, RPV deletes the route path of p when p is reported by the last switch. Besides, RPV does not issue flow rules for probing packets. In this context, frequent probing packets will be all reported to the controller. Therefore, RPV can dynamically monitor each probing packet on the data plane to detect attacks. Note that this strategy does not influence the network performance since probing packets are single-use in normal scenarios: they are only generated once at the beginning of communications. Therefore, if RPV finds that probing packets are generated at an abnormal rate, it will raise alerts.

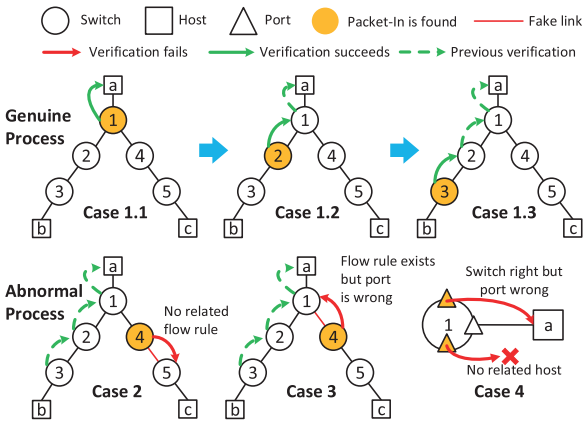


Fig. 8. Practical cases of RPV.

Recording Phase: If the reported flow f passes the verification phase, RPV records its metadata for further verification. Specifically, the 5-tuple, the current switch n, the in_port receiving f, and the out port(s) to which f will be delivered are stored by RPV (lines 28-39). Besides, RPV dynamically monitors the changes in network topology because the establishment of route paths relies on network topology.

Note that RPV is different from SPHINX [22] in the defense strategy. Both of them collect topological and flow metadata from the data plane and reconstruct route paths on the controller. However, their strategies are significantly different when it comes to how to use the collected metadata to detect topology attacks. Specifically, SPHINX validates the IP-MAC-Port binding to prevent host location hijacking attacks. Besides, it requires that a switch port can be connected to at most one switch, and that internal links should be bidirectional (the Section III.A and VIII.A in [22]). In contrast, RPV proposes that different defense strategies should be integrated together to detect covert combination attacks. Specifically, hosts should pass authentication when entering the network, then all flows must pass the verification of route paths to be delivered to their destinations, and probing packets are required to face more rigorous detection than normal packets.

TABLE VII
DETECTION RESULTS

Attack	The Malicious Packets Detected by RPV
Host Location Hijacking [25]	Unauthenticated pkt(smac=victim,sip=victim) on edge ports.
Link Fabrication (Relay Packet) [25]	ARP queries, ICMPv6 packets, and MDNS packets on the fake link.
Port Amnesia Attack [44]	Gratuitous ARP packets on the fake link every time the host is restarted.
Port Probing Attack [44], [50]	Unauthenticated pkt(smac=attacker,sip=victim) on edge ports.
IP Takeover [30]	Dhcp_release(smac=victim,sip=victim), dhcp_discover(smac=randommac) on edge ports.
Flow Poisoning [30]	Unauthenticate icmp(smac=victim) packets on edge ports.
Silent Relay Attack [43]	ARP queries, ICMPv6 packets, and MDNS packets on the fake link.
The attack 1 to latency-based defense [36]	Unauthenticated pkt(smac=randommac) on edge ports.

C. Practical Cases

With the help of the examples in Figure 8, the detection process of RPV is detailedly illustrated. Specifically, case 1 shows the verification of a genuine path, while cases 2-4 depict the verification of four different attacks.

Detect genuine route paths. Case 1 in Figure 8 illustrates the detection of a normal route path. Initially, when $host\ a$ joins the network for the first time, the controller strictly authenticates its identity, which is already conducted in [30]. Case 1.1: When a packet p is sent from $host\ a$ to $host\ b$, it first arrives at $switch\ 1$ and triggers a Table_Miss event. This event is then reported to the controller. RPV extracts the metadata of p and checks the property of the switch port receiving this packet. Since this port on $switch\ 1$ is an edge port, RPV checks whether the host connecting to this port has the correct source address. Since $host\ a$ has already passed the authentication, RPV confirms that p is a normal packet and generates a flow rule to forward p to the next hop (i.e., $switch\ 2$). RPV records the metadata of p, the current switch ($switch\ 1$), the in-port receiving this packet, and the out-port to which p is transmitted. Case 1.2: When this packet arrives at $switch\ 2$, it is reported to the controller again. Different from case 1.1, RPV finds that the switch port receiving this packet is an internal port. Therefore, it checks whether the upstream switch port has recorded this packet. As soon as the controller confirms that p has been recorded by $switch\ 1$, it records the metadata again and instructs $switch\ 2$ to forward this packet to $switch\ 3$. Case 1.3: Similar to case 1.2, this packet triggers the same verification when arriving at $switch\ 3$. Since RPV can finds its records on $switch\ 2$, p is finally forwarded to $host\ b$. Therefore, a normal flow can always be traced back to the source host.

Malicious Route Path. Cases 2-4 illustrate RPV's detection of fake links. In case 2, an attacker injects a fake link between $switch\ 4$ and $switch\ 5$. Then he sends a malicious packet in the name of $host\ c$ to $switch\ 4$, which triggers a `Packet-In` message. RPV finds that the switch port reporting this packet is an internal port, then it verifies whether the upstream port (i.e., the corresponding out-port on $switch\ 5$) has recorded this packet. However, the traffic from $host\ a$ to $host\ c$ did not pass through $switch\ 4$ and $switch\ 5$ because the two switches were not connected before the attack. Therefore, the controller finds no expected records on $switch\ 5$ and affirms that this packet is malicious. This kind of anomaly is generated by IAA when Alice communicates with Bob. Besides, the link fabrication attack [25] and the attack on LLDP freshness [36] can also be detected as long as the malicious host generates packets on fake links. Case 3 is an evolution of case 2. An attacker creates a fake link between $switch\ 4$ and $switch\ 1$, then generates a forged packet on $switch\ 4$ in the name of $host\ a$. Different from case 2, $host\ a$ has sent packets to $host\ b$, which passed through $switch\ 1$ and left relevant records on the controller. However, RPV still notices anomalies due to the fact that the expected out_port on $switch\ 1$ connects to $switch\ 2$ rather than $switch\ 4$. Case 4 describes two attacks that generate malicious packets on an edge port instead of an internal port. Specifically, 1) the attacker creates packets in the name of existing hosts (e.g., $host\ a$) or 2) injects forged packets whose source addresses are nonexistent addresses to impersonate victims [25], [30], [44], [50] or launch saturation attacks [36]. When these packets are reported to the controller, the controller uses authentication-based methods to strictly verify new hosts or host migrations, thereby finding malicious attackers.

D. Evaluation

Implementation. RPV is implemented in the topology discovery service of the Ryu controller. The experimental environment is identical to that described in Section III-G. The attack topology is shown in Figure 3a. Redis 5.0.7 [5] is used to store the collected metadata. We launch IAA and some typical topology attacks to evaluate the performance of RPV. The results are shown in Table VII.

The Detection of IAA. RPV can successfully detect IAA when Alice injects forged packets into the network (steps 11 and 13 in Figure 5). If Alice wants to directly use the responses to the probing packets sent by $Host\ D$ to disguise her communication with Bob, Bob must keep sending probing packets to $Host\ D$ to ensure that relevant flow rules are retained in switches. However, since RPV does not issue flow rules for probing packets, all these probing packets will be reported to the controller. As a result, RPV will find that the frequency of probing packets is abnormally high, then detects the fake link. If Bob does not frequently send probing packets, the disguised packets injected by Alice will be detected due to the incomplete route paths. This is because RPV deletes related records when a probing packet reaches its destination. Therefore, the packets injected by Alice fall into case 2 in Figure 8.

TABLE VIII
TIME OVERHEAD OF RPV

Destination	Impact of Graph Construction	Genuine Flow Detection	Malicious Flow Detection
Unicast Address	1.840ms	0.408ms	0.707ms
Broadcast Address	3.962 ms	0.461ms	0.637ms

Attacks injecting fake links. In addition to IAA, the existing topology attacks that inject fake links are also conducted. They are the link fabrication attack [25], the port amnesia attack [44], and the silent relay attack [43]. The fake link is injected by a malicious host that runs Ubuntu 18.04 operating system. The experiment results demonstrate that RPV is able to detect these attacks by capturing the packets that the malicious host automatically generates. Specifically, these packets include ARP queries, ICMPv6 packets [6], and MDNS packets. Besides, the Gratuitous ARP [23], [39] packets are automatically broadcast when the host is started, so RPV can also detect them. Since these packets contain unauthenticated source addresses, they immediately expose the fake link when RPV captures them. The way to avoid detection is to block such packets. However, to the best of our knowledge, none of the existing attacks explicitly note that attackers should configure the firewall to block these packets.

Attacks impersonating hosts. To evaluate the performance of RPV, several typical topology attacks that impersonate hosts are conducted. Specifically, they are the host location hijacking attack [25], the port probing attack [44], the IP takeover attack [30], and the attack 1 to latency-based defense [36]. When launching the IP takeover attack, the `isc-dhcp-server` [3] is deployed as the DHCP server. RPV detects the host location hijacking attack when attackers inject forged packets that use the victim's addresses. This is because attackers cannot pass RPV's authentication of hosts. Similarly, the port probing attack fails when the attacker claims to be the victim host. Besides, RPV can detect the IP takeover attack, but the result depends on the form of the forged packets. Specifically, when forging DHCP_RELEASE packets, if the malicious host uses its own addresses in the Ethernet frame header, it can avoid the detection of RPV. This is because RPV does not inspect the load of DHCP packets. In contrast, if the attacker directly uses the victim's addresses in the Ethernet frame header, RPV is able to detect it. By the way, the IP takeover attack may rely on the flow poisoning attack, which was also detected by RPV when attackers injected forged ICMP requests. The attack 1 to latency-based defense is also prevented by RPV when the attacker tries to flood the switch with forged packets that have random source addresses.

Time Overhead. The time overhead of RPV primarily comes from two components: (1) the detection of new flows and (2) the construction of route paths on the controller. A tree topology is used to evaluate the time consumption of RPV. Both the number of the branch and the number of the depth are set to two. The results are illustrated in Table VIII. When a new flow arrives at a switch, RPV detects whether it is a malicious flow. The detection time for a genuine flow is about 0.4 ms. For a malicious flow, the detection takes approximately 0.7 ms.

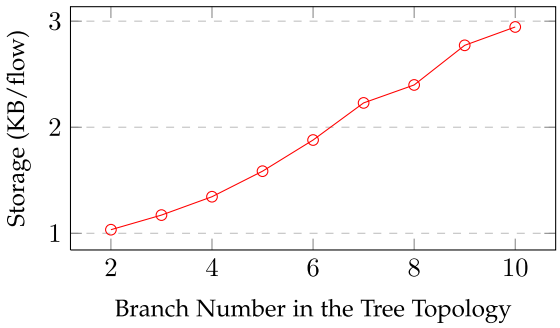


Fig. 9. The average storage consumption varies with the number of branches in the tree topology.

Note that the latency for authentication is not evaluated since it has already been completed in [30]. If the flow is genuine, RPV then stores related metadata to construct the route path on the controller. The construction time for a broadcast flow and for a unicast flow are 3.962 ms and 1.84 ms, respectively. This is because the time RPV takes to store the metadata for a broadcast flow depends on the number of active ports of the switch. The more active ports, the longer it will take to store metadata.

Storage Overhead. To evaluate the storage consumption of RPV, we fix the depth of the tree topology and change the number of branches (from 2 to 10). Figure 9 illustrates the average storage consumption per flow. For the tree topology whose branch number $= 2$ and depth $= 2$, the number of hosts is 2×2.4. Therefore, the number of flows in the network is $4 \times$ (4-1) $= 12$. In this context, the storage space consumed by each flow is about 1.03 KB. If the number of branches reaches 10, then the number of hosts becomes 10×10.100, and the number of flows is $100 \times$ (100-1) $= 9900$. Under this circumstance, the storage consumption for each flow is about 2.94 KB. We speculate that the reason for the growth of average storage consumption with branch number is that the increase of switches introduces more keys in Redis, which consumes more storage space.

CPU Usage. The CPU overhead incurred by RPV is evaluated by using different tree topologies, whose depth is two, and the number of branches ranges from 5 to 10. The psutil tool monitors the process of the Ryu controller to trace its CPU occupation dynamically and records related values every second. The average value of 100 records is used as the final CPU occupation in the current topology. To accurately distinguish the overhead incurred by RPV instead of other components of the Ryu controller, we run the controller twice, in which RPV is/isn't invoked, respectively. The two results are compared to calculate their difference, which is then compared with the original overhead when RPV is not invoked.

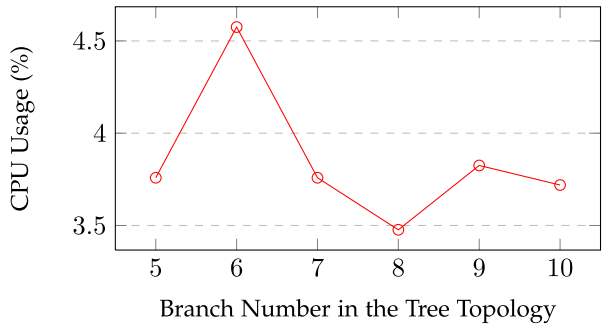


Fig. 10. The CPU usage varies with the number of branches in the tree topology.

The final ratio is regarded as the real overhead incurred by RPV. Figure 10 illustrates the monitor results in different topologies. It is clear that the CPU usage does not increase with the growth of the network scale. This is because RPV incrementally verifies route paths, which will not get influenced by the number of switches. The average CPU usage is only 3.85%. We deem that such value is acceptable and will not put a heavy burden on the CPU.

V. Limitation and Discussion

Attacks. Although IAA is powerful, it does have some limitations. First, the fake link must change the controller's routing strategy, forcing some traffic to pass through it. Only in this way can attackers sniff the packets existing in the network and then use them to disguise the communication between two attackers. As long as packets are passing through the fake link, attackers can always find hosts whose communication with Bob passes through the fake link. Second, the springboards must respond to network probing packets such as ICMP. The firewalls deployed on end hosts or middle boxes may block probing packets. In this context, Bob cannot verify whether the springboard will respond to his queries, let alone force related flow rules to be installed along the route path. Third, there is a time window between step 7 and step 12 in Figure 5, during which the compromised host cannot send out packets. This is because attackers have not found appropriate springboards to disguise their communication, so they must avoid generating suspicious packets on the fake link. However, the failure of communication may cause the alert of $Host\ C$'s user, thereby exposing attackers.

Defenses. Although RPV can detect IAA, it still has some limitations. If attackers use normal packets, e.g., UDP packets, rather than probing packets to disguise their communication, RPV will not detect any anomalies. However, since RPV cannot have an in-depth inspection of all the packets in the data plane, it cannot detect such stealthy malicious behaviors. This limitation also applies to other defenses that are deployed on the controller and rely on the reports from switches to detect attacks in the data plane. **We deem it as an inherent drawback of the SDN architecture.** This is because the controller cannot directly handle all the traffic in the data plane. It has to make decisions based on a small number of sample packets. If attackers can avoid generating `Packet-In` messages, they may be able to circumvent defenses. Therefore, the security of SDN needs to be further studied to deal with potential threats.

VI. Related Work

Topology Attacks. Various attacks have been proposed to poison the visibility of the controller. For example, host location hijacking attacks and link fabrication attacks are the first two types of attacks against topology discovery [25]. The former exploits a vulnerability in the Host Discovery Service to impersonate victim hosts, while the latter takes advantage of a vulnerability in the Link Discovery Service to create fake links. Later, identifier binding attacks [30] reveal the weakness of SDN identifier bindings and demonstrate their values to impersonate victim hosts. Port amnesia attacks [44] propose a novel way to evade the detection of TopoGuard [25] and then exploit the Link Discovery Service again. Port probing attacks use the migration of the victim host to prevent it from reconnecting to the network [44], [50]. Besides, reverse loop attacks uncover a vulnerability in the Floodlight controller that can decrease the network performance [36], and topology freezing attacks show how to use another vulnerability in Floodlight to disable the update of network topology [36]. However, these attacks target either an unprotected controller or a single defense, ignoring the scenarios where multiple defenses are deployed.

Defenses. In response to the increasing threat of topology attacks, a variety of defenses are developed. At first, TopoGuard verifies the integrity of LLDP packets and checks the property of switch ports to detect fake links [25]. Besides, it uses the pre/post-condition of host migration to detect host impersonation attacks [25]. However, as the vulnerabilities in TopoGuard are found, TopoGuard+ is proposed. It uses the delay to distinguish malicious links and monitors certain types of packets to detect malicious hosts [44]. Besides, SPHINX proposes a general framework that verifies certain constraints, such as IP-MAC-port binding, to detect topology attacks [22]. SecureBinder proposes an authentication-based method to verify the identifier bindings of hosts [30]. Considering that malicious switches can also poison the network topology, SPV generates abnormal packets to trick malicious switches into exposing themselves [11]. In addition, there are unnamed countermeasures that protect topology security, such as the latency-based defense [44], [45] or new link discovery protocols for SDN [10], [12], [53]. However, most existing defenses have vulnerabilities that can be exploited by attackers. Besides, these defenses are developed independently for specific situations, which results in the lack of collaboration among them, thereby causing hidden security risks even if they are deployed simultaneously.

VII. Conclusion

This paper has presented the first combination attack, Invisible Assailant Attack, to poison the network topology in SDN. IAA can inject and maintain fake links while bypassing 12 existing defenses. It uses a novel method to disguise the attack packets as normal network packets that already exist in

the network, which greatly increases the difficulty of detection. To mitigate IAA, this paper has proposed a systematical defense framework, RPV, which orchestrates multiple defense strategies to detect potential threats that violate the integrity of route paths. The experiment results of RPV have demonstrated its effectiveness and efficiency when detecting IAA as well as eight typical topology attacks. Since the architecture of SDN determines that the centralized controller has to monitor the network status by sampling packets, stealthy attacks like IAA are likely to emerge continuously. We hope our work can encourage researchers to pay more attention to the security of the network topology and to devise further ways to make SDN more secure.

REFERENCES

[1] *Data Plane Development Kit*. Accessed: Sep. 2019. [Online]. Available: http://core.dpdk.org/download/

[2] *iPerf: The Ultimate Speed Test Tool for TCP, UDP and SCTP*. Accessed: Sep. 2019. [Online]. Available: https://iperf.fr/iperf-download.php

[3] *ISC DHCP Servers*. Accessed: Mar. 2020. [Online]. Available: https://www.isc.org/dhcp/

[4] *Mininet*. Accessed: Aug. 2019. [Online]. Available: http://mininet.org

[5] *Redis Database*. Accessed: Aug. 2019. [Online]. Available: https://redis.io/

[6] S. Deering and A. Conta, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*, document RFC 2463, Dec. 1998.

[7] A. A. Abdelltif, E. Ahmed, A. T. Fong, A. Gani, and M. Imran, "SDN-based load balancing service for cloud servers," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 106–111, Aug. 2018.

[8] A. I. A. Ahmed, S. H. A. Hamid, A. Gani, S. Khan, and M. K. Khan, "Trust and reputation for Internet of Things: Fundamentals, taxonomy, and open research challenges," *J. Netw. Comput. Appl.*, vol. 145, Nov. 2019, Art. no. 102409.

[9] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran, and S. Guizani, "Securing software defined networks: Taxonomy, requirements, and open issues," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 36–44, Apr. 2015.

[10] T. Alharbi, M. Portmann, and F. Pakzad, "The (in)security of topology discovery in software defined networks," in *Proc. IEEE 40th Conf. Local Comput. Netw. (LCN)*, Oct. 2015, pp. 502–505.

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

[12] A. Azzouni, R. Boutaba, N. T. M. Trang, and G. Pujolle, "SOFTDP: Secure and efficient topology discovery protocol for SDN," 2017, *arXiv:1705.04527*.

[13] K. Benzekki, A. E. Fergougui, and A. E. Elalaoui, "Software-defined networking (SDN): A survey," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5803–5833, Dec. 2016.

[14] A. Branitskiy and I. Kotenko, "Network attack detection based on combination of neural, immune and neuro-fuzzy classifiers," in *Proc. IEEE 18th Int. Conf. Comput. Sci. Eng.*, Oct. 2015, pp. 152–159.

[15] A. Branitskiy and I. Kotenko, "Hybridization of computational intelligence methods for attack detection in computer networks," *J. Comput. Sci.*, vol. 23, pp. 145–156, Nov. 2017.

[16] J. Cao, M. Xu, Q. Li, K. Sun, Y. Yang, and J. Zheng, "Disrupting SDN via the data plane: A low-rate flow table overflow attack," in *Security and Privacy in Communication Networks*. Cham, Switzerland: Springer, 2018, pp. 356–376.

[17] R. Chaudhary, G. S. Aujla, N. Kumar, and J. J. P. C. Rodrigues, "Optimized big data management across multi-cloud data centers: Software-defined-network-based analysis," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 118–126, Feb. 2018.

[18] X. Chen *et al.*, "MTP: Avoiding control plane overload with measurement task placement," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, May 2021, pp. 1–10.

[19] X. Chen, Q. Huang, D. Zhang, H. Zhou, and C. Wu, "ApproSync: Approximate state synchronization for programmable networks," in *Proc. IEEE 28th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2020, pp. 1–12.

[20] X. Chen *et al.*, "SPEED: Resource-efficient and high-performance deployment for data plane programs," in *Proc. IEEE 28th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2020, pp. 1–12.

[21] S. Deng, X. Gao, Z. Lu, and X. Gao, "Packet injection attack and its defense in software-defined networks," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 3, pp. 695–705, Mar. 2018.

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

[23] K. R. Fall and W. R. Stevens, *TCP/IP illustrated: The Protocols*, vol. 1. Reading, MA, USA: Addison-Wesley, 2011.

[24] J. Gu, C. Li, D. Lei, and Q. Li, "Combination attack of Android applications analysis scheme based on privacy leak," in *Proc. 4th Int. Conf. Cloud Comput. Intell. Syst. (CCIS)*, Aug. 2016, pp. 62–66.

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

[26] S. Hosseini and B. M. H. Zade, "New hybrid method for attack detection using combination of evolutionary algorithms, SVM, and ANN," *Comput. Netw.*, vol. 173, May 2020, Art. no. 107168.

[27] X. Huang, P. Shi, Y. Liu, and F. Xu, "Towards trusted and efficient SDN topology discovery: A lightweight topology verification scheme," *Comput. Netw.*, vol. 170, Apr. 2020, Art. no. 107119.

[28] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[29] S. Jero, X. Bu, C. Nita-Rotaru, H. Okhravi, R. Skowyra, and S. Fahmy, "Beads: Automated attack discovery in OpenFlow-based SDN systems," in *Proc. Int. Symp. Res. Attacks, Intrusions, Defenses*. Cham, Switzerland: Springer, 2017, pp. 311–333.

[30] S. Jero, W. Koch, R. Skowyra, H. Okhravi, C. Nita-Rotaru, and D. Bigelow, "Identifier binding attacks and defenses in software-defined networks," in *Proc. USENIX Secur. Symp.* Berkeley, CA, USA: USENIX Association, 2017, pp. 415–432.

[31] N. Kaur, A. K. Singh, N. Kumar, and S. Srivastava, "Performance impact of topology poisoning attack in SDN and its countermeasure," in *Proc. 10th Int. Conf. Secur. Inf. Netw.*, Oct. 2017, pp. 179–184.

[32] M. K. Khan, "Fingerprint biometric-based self-authentication and deniable authentication schemes for the electronic world," *IETE Tech. Rev.*, vol. 26, no. 3, pp. 191–195, 2009.

[33] D. Kreutz, F. M. V. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 55–60.

[34] S. Lee, C. Yoon, C. Lee, S. Shin, V. Yegneswaran, and P. Porras, "DELTA: A security assessment framework for software-defined networks," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2017, pp. 1–15.

[35] Q. Li, X. Zou, Q. Huang, J. Zheng, and P. P. C. Lee, "Dynamic packet forwarding verification in SDN," *IEEE Trans. Depend. Sec. Comput.*, vol. 16, no. 6, pp. 915–929, Nov./Dec. 2019.

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

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

[38] T.-H. Nguyen and M. Yoo, "Analysis of link discovery service attacks in SDN controller," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, 2017, pp. 259–261.

[39] D. C. Plummer, *An Ethernet Address Resolution Protocol to 48.bit Ethernet Address for Transmission on Ethernet Hardware*, document RFC 826, Nov. 1982.

[40] J. Postel, *Internet Control Message Protocol*, document RFC 792, Sep. 1981.

[41] J. Postel, *Transmission Control Protocol*, document RFC 793, Sep. 1981.

[42] Y. Ryoya, S. Shiota, N. Ono, and H. Kiya, "Improving replay attack detection by combination of spatial and spectral features," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, Nov. 2019, pp. 833–837.

[43] P. Shrivastava, A. Agarwal, and K. Kataoka, "Detection of topology poisoning by silent relay attacker in SDN," in *Proc. Int. Conf. Mobile Comput. Netw.*, Oct. 2018, pp. 792–794.

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

[45] D. Smyth, S. McSweeney, D. O'Shea, and V. Cionca, "Detecting link fabrication attacks in software-defined networks," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2017, pp. 1–8.

[46] K. Thimmaraju, L. Schiff, and S. Schmid, "Outsmarting network security with SDN teleportation," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroSP)*, Apr. 2017, pp. 563–578.

[47] B. E. Ujcich *et al.*, "Cross-app poisoning in software-defined networking," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 648–663.
[48] T. Wang, F. Liu, J. Guo, and H. Xu, "Dynamic SDN controller assignment in data center networks: Stable matching with transfers," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2016, pp. 1–9.
[49] X. Wang, N. Gao, L. Zhang, Z. Liu, and L. Wang, "Novel MITM attacks on security protocols in SDN: A feasibility study," in *Proc. Int. Conf. Inf. Commun. Secur.* Cham, Switzerland: Springer, 2016, pp. 455–465.
[50] S. Xiang, H. Zhu, L. Xiao, and W. Xie, "Modeling and verifying topoguard in OpenFlow-based software defined networks," in *Proc. Int. Symp. Theor. Aspects Softw. Eng. (TASE)*, Aug. 2018, pp. 84–91.
[51] M. Yaibuates and R. Chaisricharoen, "A combination of ICMP and ARP for DHCP malicious attack identification," in *Proc. Joint Int. Conf. Digit. Arts, Media Technol. ECTI Northern Sect. Conf. Electr., Electron., Comput. Telecommun. Eng. (ECTI DAMT NCON)*, Mar. 2020, pp. 15–19.
[52] M. Zhang, G. Li, L. Xu, J. Bi, G. Gu, and J. Bai, "Control plane reflection attacks in SDNs: New attacks and countermeasures," in *Proc. Int. Symp. Res. Attacks, Intrusions, Defenses*. Cham, Switzerland: Springer, 2018, pp. 161–183.
[53] X. Zhao, L. Yao, and G. Wu, "ESLD: An efficient and secure link discovery scheme for software-defined networking," *Int. J. Commun. Syst.*, vol. 31, no. 10, Jul. 2018, Art. no. e3552.
[54] 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," *Secur. Commun. Netw.*, vol. 2018, pp. 1–15, Jan. 2018.

Dezhang Kong (Graduate Student Member, IEEE) received the B.E. degree in information security from the Huazhong University of Science and Technology in 2018. He is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Zhejiang University, China. His research interests include network security, software-defined networks, and programmable networks.

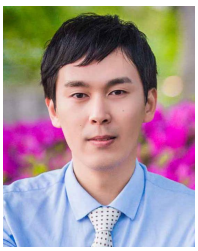
Yi Shen received the B.E. degree in software engineering from Shandong University in 2018. He is currently pursuing the Ph.D. degree in network space security with the College of Computer Science and Technology, Zhejiang University. His research interests include software-defined networking and the security of SDN itself.

Xiang Chen (Member, IEEE) received the B.Eng. and M.Eng. degrees from Fuzhou University in 2019 and 2022, respectively. He is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Zhejiang University, China. He has published papers in IEEE INFOCOM and IEEE ICNP. His research interests include programmable networks and network security. He received the Best Paper Award from IEEE/ACM IWQoS 2021, and the Best Paper Candidate from IEEE INFOCOM 2021.

Qiumei Cheng received the Ph.D. degree from the College of Computer Science and Technology, Zhejiang University, in 2021. Her research interests include network security, deep learning, and graph data analytics.

Hongyan Liu received the B.Eng. degree from the College of Mathematics and Computer Science, Fuzhou University. He is currently pursuing the M.Eng. degree with the College of Computer Science, Zhejiang University, China. He has published papers in IEEE INFOCOM, IEEE ICNP, and IEEE/ACM IWQoS. His current research interests include network measurement and programmable networks.

Dong Zhang (Member, IEEE) received the B.S. and Ph.D. degrees from Zhejiang University, China, in 2005 and 2010, respectively. He visited Alabama University, USA, as a Visiting Scholar, from 2018 to 2019. He is currently a Professor with the College of Computer Science and Big Data, Fuzhou University, China. His research interests include software-defined networking, network virtualization, and internet QoS.



Xuan Liu (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from Southeast University, China. He is currently a Lecturer and a Master's supervisor with the College of Information Engineering (College of Artificial Intelligence), Yangzhou University, China, and a Post-Doctoral Researcher with the School of Computer Science and Engineering, Southeast University, China. His main research interests include future content networking (future internet architecture, DAN, ICN, CDN, VLC, and V2R). He served(s) as a TPC Member for ACM MobiCom, IEEE INFOCOM, IEEE ICC, IEEE Globecom, IEEE WCNC, IFIP/IEEE IM, IEEE NOMS, IEEE PIMRC, IEEE MSN, IEEE VTC, APNOMS, and CollaborateCom. He is also serving as an Editorial Board Member for *Computer Communications*, *TELS*, *IET Networks*, and *IET Smart Cities*. Besides, he served as a reviewer for more than 200 reputable conferences/journal papers.

Shuangxi Chen, photograph and biography not available at the time of publication.

Chunming Wu received the Ph.D. degree in computer science from Zhejiang University in 1995. He is currently a Professor with the College of Computer Science and Technology, Zhejiang University. He is also the Associate Director of the Research Institute of Computer System Architecture and Network Security, Zhejiang University, and the Director of the NGNT Laboratory. His research interests include software-defined networking, reconfigurable networks, proactive network defense, network security, network virtualization, the architecture of next-generation internet, and intelligent networks.