TopoPoisoning Attacks and Prevention in Hybrid Software-Defined Networks

Pragati Shrivastava∗, Kotaro Kataoka†
Department of Computer Science and Engineering
Indian Institute of Technology Hyderabad
Email: *cs14resch11007@iith.ac.in, †kotaro@cse.iith.ac.in

Abstract—The hybrid software-defined networks (SDN) architectures are beneficial for a smooth transition and less costly SDN deployment. However, legacy switches and SDN switches coexistence brings new challenges of deployment inconsistency management and security. Security is not well studied for hybrid SDN architecture. In this paper, we study the topology poisoning attacks in hybrid SDN for the first time. We propose new attack vectors for link fabrication in hybrid SDN. The new attack is named “multi-hop link fabrication”, in which an adversary successfully injects a fake multi-hop link (MHL) by exploiting the link discovery protocols. We presented the Hybrid-Shield, a link verification framework for hybrid SDN link discovery. Hybrid-Shield introduces a novel verification technique that includes: i) monitoring legacy switch and host generated traffic at MHL and ii) validating the existence of legacy switches contained in an MHL. This paper presents the prototype implementation of Hybrid-Shield over a real SDN controller. The experimental evaluation is performed with the mininet virtual network emulation. Our evaluation shows that Hybrid-Shield is capable of detecting MHL fabrication attacks in real-time with high accuracy. Hybrid-Shield’s performance evaluation shows that it is lightweight at the controller as it causes less overhead and requires no additional functionalities at the SDN controller for deployment.

Index Terms—Software-defined Networks (SDN), Topology Poisoning, OpenFlow, Hybrid SDN

I. INTRODUCTION

Software-Defined Networking (SDN) facilitates agile network management, programmable network devices and supports network innovations by easy deployment. However, the adaptation of SDN is costly in terms of replacing all legacy devices, human efforts, and network downtime [1]. Therefore, the transition from traditional network architecture to SDN could not be straightforward. This transition brings the hybrid SDN deployment, i.e., the coexistence of legacy switches with the SDN switches in the network. Hybrid SDN deployment is less costly and extracts the benefits of centralized SDN and distributed legacy networks together [2] [3]. Therefore, hybrid SDN becomes an easy and low-cost choice for SDN deployments.

Securing the SDN architecture is paramount for error-prone and efficient network functioning. The hybrid SDN deployment requires the new control functionalities, routing, and monitoring strategies [4] [5], which brings new security concerns. Security solutions of SDN do not consider hybrid SDN deployment. For example, the topology poisoning checker TopoGuard [6] and data-plane device configuration debugger Anteater [7] require further consideration of the hybrid SDN deployment.

The SDN controller exchanges the control messages with the SDN switches through the southbound protocol (e.g., OpenFlow) and collects the information about network device connectivity, flow statistics, and traffic. This information collectively provides the abstract view of the network to SDN applications and is utilized for forwarding decisions. An attacker can exploit the communication between the controller and switches and damage the controller’s network view. This tampered network state causes dysfunction of the network applications, such as QoS and Load balancing. These attacks are called Topology Poisoning attacks. Topology Poisoning attacks can trigger more severe attacks like Denial-of-Service (DoS) at the controller. Protection from Topology Poisoning attacks become more challenging in hybrid SDN deployment than the pure SDN. The fundamental challenges for defending against topology poisoning attacks in a hybrid SDN are: 1) The controller cannot control the legacy part of the network, i.e., the controller cannot instruct legacy switches in hybrid SDN. 2) The legacy part of the network is unknown to the controller, i.e., the controller does not have the complete topology view in hybrid SDN. 3) The controller cannot directly monitor the topology changes in the legacy network. The controller can only observe the incoming packets from the SDN switches connected to the legacy switches. The limited control and monitoring over legacy network devices make the defense against topology poisoning more challenging in hybrid SDN.

In this paper, we first study the topology poisoning attacks in hybrid SDN. We discover a new topology poisoning attack in hybrid SDN, named Multi-hop link (MHL) fabrication. The attacker injects a fake MHL by exploiting the SDN controller’s link discovery protocol in hybrid SDN deployment. The injection of fake MHL can severely affect the security of the data plane and control plane both. An attacker can perform the man-in-the-middle (MITM) attack in the data plane and DoS attack at the controller by MHL fabrication attack. The hybrid SDN deployments [8] [9] [10] aim to maximize the benefits of SDN functionalities like monitoring and control; therefore, redirect all the traffic via SDN switches. Thus, the SDN switches become an easy target for attackers to eavesdrop/manipulate the messages. The fake MHL injection triggers the spanning tree calculation and new route deployment at the controller, which can be further exploited to perform DoS at the controller by making the controller’s topology and routing services busy.
Although the topology poisoning attacks are not investigated in hybrid SDN, these are well explored in SDN. Thus, we investigate the existing defense strategies [6] [11] [12] [13] [14] of direct link fabrication attacks in SDN. We found that defenses proposed for SDN could not apply to the MHL fabrication attack due to the differences in direct link and MHL properties. For example, the MHL’s latency threshold generalization is nontrivial, as MHL is not a direct physical connection. It consists of an unknown number of legacy switches and traffic originating from legacy devices. However, the latency-based prediction of a MITM host in the direct link is well-researched [12] [13].

In order to mitigate the MHL fabrication attack in hybrid SDN, we propose Hybrid-Shield, a new prevention framework for a hybrid SDN controller. Hybrid-Shield prevents the attacker from poisoning the controller’s topology view and mitigates the attack by blocking the fake MHLs. Hybrid-Shield cannot prevent the attacker from launching the attack, but it prevents the consequences of an MHL fabrication attack. Consequently, it provides both prevention and mitigation. Hybrid-Shield utilizes the existing SDN functionalities and protocols of the legacy network to prevent MHL fabrication. Hybrid-Shield includes the novel verification strategy based on monitoring the traffic originated by the legacy network in the MHL and validating the existence of legacy switches in MHL. This paper makes the following contributions:

1) We construct a new relay-based link fabrication attack, “MHL fabrication,” to inject a fake MHL in the controller’s topology view. In the proposed MHL fabrication attack, an adversary requires the careful observation of packets to drop the LLDP messages while relaying the BDDP messages.

2) We present the extension of the proposed basic MHL fabrication to evade simple behavior-based defense approaches. In the proposed attacking strategies over the basic MHL fabrication attack, the adversary mimics the legacy switches and hosts traffic to impersonate the MHL’s behavior.

3) We propose a novel defense mechanism for both categories of proposed MHL fabrication attacks, basic and extended ones. We introduce the MHL behavior monitoring and validation of legacy switch’s MAC learning process a unique defense strategy.

4) We propose “Hybrid-Shield,” a prevention framework. We present the prototype implementation of Hybrid-Shield with the Floodlight open-source controller. Hybrid-Shield can be easily extended to any SDN controller as it utilizes the basic SDN functionalities. Hybrid-Shield integration is lightweight at the controller as it majorly requires passive monitoring of topology updates and Packet_In messages.

5) We present the experimental evaluation to show the Hybrid-Shield’s effectiveness as quick detection, i.e., it takes less than a sub-millisecond and high accuracy of MHL verification. We analyze the overhead introduced at the controller with the integration of Hybrid-Shield. Hybrid-Shield imposes negligible overhead.

II. TOPOLOGY DISCOVERY IN SDN

A. Overview of SDN and Hybrid SDN

SDN separates the control logic from the forwarding devices (data plane) and puts it in the logically centralized controller (control plane). The SDN controller maintains the global view of network topology and controls the network functionalities. The control plane runs various services to maintain the network states and control the network functionalities. OpenFlow is the most widely utilized communication protocol between the data plane and control plane; hence SDN also refers to OpenFlow [15] network. The SDN controller’s decisions to control and manage the data plane devices and network functionalities depend upon the controller’s topology view. The controller discovers and maintains the data plane topology’s logical view by implementing the various topology discovery mechanisms.

The network where legacy switches coexist with the SDN-enabled switches is called hybrid SDN. In hybrid SDN, the distributed and centralized control works together and the SDN controller can communicate with SDN switches, as shown in Fig. 1.

The controller discovers two types of links in hybrid data plane:

Direct Link: The direct connection between two SDN switches in the data plane is a direct link. For example, link between switch-ports pairs (S1, Port-20) and (S2, Port-11) is a direct link shown in Fig. 1.

Multi-hop Link (MHL): The MHL is a logical indirect link between two SDN switches. An MHL connects two SDN switches across a legacy network that contains an unknown number of layer-2 legacy switches and hosts in physical topology. For example, an in-direct connection between two SDN switches (S1, Port-4) and (S3, Port-6) is an MHL shown in Fig. 1.

B. Topology Discovery at Control Plane

Topology discovery is performed by the SDN controller to maintain the current global topology view. Topology discovery includes the host discovery and link discovery. In the hybrid SDN deployment, the controller discovers the direct links between the SDN switches as well as the external MHLs across legacy switches. The SDN controllers [16] [17] [18]...
implement the OFDP (OpenFlow discovery protocol) that utilizes the LLDP (Link layer discovery protocol) to find the links between the SDN switches in the data plane. The MHLs are discovered using the BDDP (Broadcast domain discovery protocol), a variation to LLDP.

The LLDP and BDDP both pause the security invariants of integrity and freshness and are vulnerable to spoofing and replay attacks. The origin of LLDP and BDDP packets cannot be verified, and an adversary can spoof LLDP packets to inject fake links. However, the TopoGuard proposed a mechanism to compute a controller signed TLV over the DPID and port of source switch using a cryptographic key only known to the controller and add the authenticator TLV to LLDP to protect the integrity of the packets. Further, to provide integrity with freshness, Alharbi et al. proposed that authenticator TLV is updated at every LLDP round, but it needs recording of keys of every LLDP round. Marin et al. proposed the calculation of authenticator TLV over the source switch DPID, port number, and time-stamp using a single cryptographic key, which resolves the issues of maintaining the keys of every LLDP round and provide the freshness invariant. All discussed authentication-based protection mechanisms work on attacks based on spoofed or replayed LLDP packets. However, the authentication mechanisms do not prevent the fake link injection by relaying the link discovery protocol packets.

1) Direct Link Discovery: The controller periodically sends the LLDP messages wrapped in Packet_Out to every switch-port in the data plane. SDN switches forward the LLDP message to the port instructed by the controller. When a switch receives an LLDP packet from another switch-port, it sends the LLDP packet back to the controller wrapped in the Packet_In message, as shown in Fig. 2. This procedure determines how the controller discovers the links between SDN switches. The controller maintains the links’ liveliness and keeps the dynamic link-state in the data plane with the periodic link discovery.

2) Multi-hop Link Discovery: In hybrid SDN networks, the controller cannot directly monitor and control the legacy part of the network. LLDP protocol packets are communicated between switches to share the switch and its neighbors’ information in the legacy network. When a legacy switch receives an LLDP packet from its neighbor switch, it collects the information and discards the LLDP packet. Thus LLDP packet can traverse only one hop in the link layer. Therefore, to discover the MHLs, the SDN controller utilizes BDDP. The BDDP packets contain the broadcast destination MAC address and different protocol numbers than LLDP. The SDN controller periodically sends the BDDP packets wrapped in Packet_Out to every SDN switch-port, and the SDN switch forwards the BDDP packets to the instructed port. When BDDP packets reach the legacy switches, the legacy switches broadcast the BDDP packets as non-LLDP packets, and BDDP packets make the way to the SDN switches. If a BDDP packet is received wrapped in Packet_In from another end SDN switch, then the controller assumes there exists an MHL between the switch at which Packet_Out sent and the switch that sends back the BDDP packet as a Packet_In, as depicted in Fig. 3.

III. THREAT MODEL

This section presents the MHL fabrication attack in hybrid SDN that enables the injection of fake MHL in the controller’s topology view. We first describe the attacker’s capabilities required for the attack. Then the MHL fabrication attacks their categorization based on the attacker’s capabilities, and possible adverse effects of MHL fabrication attacks are discussed.

A. Attacker Model

We consider an adversary compromises a host in the network. The attacker host can connect to two SDN switches and relays the packets between these two victim switches. The attacker hosts have the read privileges and can read packet headers to filter out the LLDP packets.

The attacker does not compromise any network infrastructure, switches, SDN controller, and the control channel between controller and switches is protected using TLS/SSL. We assume the attacker host connects to the SDN switches to perform the attack. In that case, the attacker does not require prior knowledge of topology. Suppose the attacker connects to the legacy switches. Then attacker requires the knowledge of hybrid data plane topology and other parameters to perform a successful injection of fake MHL, discussed in detail in Section III-C.

B. MHL Fabrication Attack

MHL fabrication is performed by a compromised host connected to the two victim SDN switches, as shown in Fig. 4.
The LLDP packets traverse only one hop because the legacy switch reads the LLDP content and discards them. Therefore attacker reads the packet headers and drops the LLDP packets. On the other hand, the legacy switch handles the BDDP packets as regular broadcast packets because the destination address of the BDDP packet is the broadcast address. Thus the attacker reads the packet header and relays the BDDP packets between two victim switches.

The attack scenario is depicted in Fig. 4. An attacker host connects to two victim SDN switches, S1 and S2. Initially, the attacker listens to the incoming packets at both switch-ports (S1, P1) and (S2, P2). The controller periodically generates link discovery protocol packets (LLDP/BDDP) that reach the attacking host via victim switches. Attacker relays BDDP packets between both switches and drops LLDP packets between the victim SDN switches. Whenever the victim switches receive the relayed BDDP packets, switches will forward these packets wrapped in the Packet_In to the controller. It causes the controller to discover an MHL between switchports (S1, P1) and (S2, P2), a fabricated link. It is a basic MHL fabrication attack strategy, and we define the more strong attacking strategies in the attack categorization, which are extensions of basic attack, presented in Table I.

The legacy switches and hosts connected in the legacy network flows in the MHL. The MHL fabrication attack categorization is presented in Table I. The attacker can be a passive host that performs a basic relay-based MHL fabrication attack or an active attacker trying to mimic the legacy network by generating traffic similar to legacy switches or hosts.

Detection of fake MHL becomes harder as the attacking strategy evolves from the passive relay to the active relay, mimicking most MHL’s behavior.

C. Attack Assumptions

An attacking host connects to the SDN switches only; an attacker cannot connect to the legacy switch and become a part of the legacy network. This assumption is reasonable as the attack involving legacy switches requires additional information, limiting the feasibility of an attack involving legacy switches. The required information includes prior knowledge of data plane topology, broadcast domains, and the existing MHL discovered by the controller, which is discussed further.

Feasibility of Assumptions: To perform an MHL fabrication attack by connecting through one or two legacy switches, the attacker requires the topology information to create a fake MHL successfully. We have discussed possible attack scenarios that involve one or two legacy switches and discussed the limitations further.

Scenario 1: An attacking host connects to a legacy switch and an SDN switch, as depicted in Fig. 5.

Scenario 2: An attacker host connects to two legacy switches, as depicted in Fig. 6

The attacker does not know the topology beyond the victim legacy switches in both the scenarios, as mentioned using a question mark in Fig. 5 and Fig. 6. Practically in scenarios 1 and 2, a fake MHL can only be created when the following three conditions are satisfied at the same time.

| Attack level | Attack strategy | Simple relay | Mimicking switch-traffic | Mimicking host-traffic |
|--------------|----------------|--------------|--------------------------|-----------------------|
| 1. Passive bddp-relay | The attacker relays all packets between two switches and drops the LLDP packets. The attacker relays bddp, mimics switching protocols and fabricates random end-host traffic. The attacker is a relaying host and the attacking host creates an illusion of legacy switches by generating control packets of switching protocols e.g., STP/RSTP. | yes | no | no |
| 2. Active bddp-relay mimics legacy network. | | yes | yes | yes |
| 3. Active bddp-relay mimics layer-2 legacy switch. | | yes | yes | no |

TABLE I
ATTACK CATEGORIZATION WITH DIFFERENT ATTACKER’S CAPABILITIES
1) The end switches, e.g., S1 and S2, are SDN switches in the same broadcast domain in the data plane.

2) The SDN switches S1 and S2 are not physically and virtually the same switch. Otherwise, the attack can create a loop, which is not a fake MHL, in the topology.

3) There is not an existing MHL between S1 and S2. Otherwise, the attack is meaningless.

Therefore, even though the attacker can connect to legacy switches in both scenarios, fake MHL injection is not guaranteed. An attacker can select the victim legacy switches if it knows the layer-2 hybrid SDN topology. However, learning or guessing layer-2 topology information is not feasible for an attacker host. For example:

- The LLDP packets travel up to one hop, the attacker cannot receive the LLDP packets from a neighboring legacy switch, and the attacker fails to learn layer-2 topology.
- STP messages do not contain sufficient topology information, and the STP’s port fast feature does not allow a host to receive STP packets if deployed.

Consequently, the attacker cannot learn the layer-2 topology of the hybrid SDN.

### D. Attack Demonstration

We demonstrate the MHL fabrication attack over the Floodlight [17] and ONOS [19] controller and implements a hybrid SDN topology using Mininet [22] and a mininet host as an attacker. We create a python script to filter the LLDP packets and relays all other messages in both directions. The attacker host connects to two SDN switches and injects a fake MHL by running the relay script. Fig. 7 presents the successful injection of fake MHL in the topology.

### E. Adverse Effects Caused by MHL Fabrication Attack

In the hybrid SDN deployments, the controller cannot directly control and monitor the legacy part of the network, which brings enormous opportunities to attackers to perform DoS and MITM attacks as described further.

The SDN controller maintains the logical topology view, and any event in the topology triggers the spanning tree and shortest path algorithms to calculate and implements new paths. The attacker can exploit these events to make the controller busy. For example, the floodlight controller manages the data plane topology view. The topology view consists of the clusters (i.e., the completely connected component of a network) and archipelagos (i.e., groups of connected clusters/groups of external links connecting the OpenFlow islands). The controller calculates and implements the broadcast tree for every archipelago. The MHL discovery triggers the creation/merging of archipelagos. The attacker can trigger broadcast tree creation and shortest path calculation by frequent fake MHL injection, which indirectly makes the controller busy.

The MHL connects the multiple SDN island via a legacy network in the same broadcast domain. An adversary can increase the broadcast domain by creating non-existing fake MHLs between two SDN islands. This further increases the broadcast traffic at the network that wastes the network bandwidth, and creates congestion.

Existing hybrid SDN implementations [8] [9] [23] proposed the path control mechanisms aim to maximize the benefits of SDN integration. The forwarding paths are selected so that all the host-traffic must pass by at least one SDN switch in the network so that the monitoring and control can be enhanced. Therefore, the SDN switches become the central point for tapping the traffic in hybrid deployments. It makes the SDN switches an easy target for attackers. The MITM host present in the fake MHL can easily sniff and analyze the network flows to launch further attacks in the network.

### IV. HYBRID-SHIELD: A PREVENTION FROM MULTI-HOP LINK FABRICATION ATTACK

This paper proposes Hybrid-Shield, a system for securing hybrid SDN from MHL fabrication attacks. Hybrid-Shield prevents the attacker from poisoning the controller’s topology view and mitigates the attack by blocking the fake MHLs. Hybrid-Shield prevention system includes a dynamic MHL verification method, which verifies every MHL update.

### A. Hybrid-Shield Defense

Our MHL verification strategy is based on: i) Monitoring the traffic originated from a legacy network contained in an MHL, and ii) Validating the presence of legacy switches at MHL. The “traffic monitoring” prevents the passive MHL fabrication, and the “switch learning validation” prevents the active MHL fabrication attack. Both techniques are described further.

To explain the defense mechanism, we first elaborate on the legitimate MHL scenario. In the legitimate MHL scenario, all traffic originated from the legacy network reaches SDN switches, and SDN switches forward these packets as Packet_In to the SDN controller, as shown in Fig. 8. Thus, the SDN controller can monitor the traffic generated by devices present in the legacy network, which is categorized as follows:

1) **Switch-traffic:** The control packets generated by layer-2 legacy switches for the management and forwarding, such as STP/ RSTP, LLDP/CDP packets;

2) **Host-traffic:** The broadcast/unicast traffic generated by hosts connected in the legacy part of the MHL, such as ARP requests, DHCP messages, and DNS;

3) **Controller-traffic:** The controller generated link discovery protocol packets received back from SDN switches, e.g., BDDP.

---

**Fig. 6.** Scenario 2: MHL fabrication attack with two victim legacy switches.

---

**Fig. 7.** The LLDP packets travel up to one hop, the attacker cannot receive the LLDP packets from a neighboring legacy switch, and the attacker fails to learn layer-2 topology. The attacker can select the victim legacy switches if it knows the layer-2 hybrid SDN topology. However, learning or guessing layer-2 topology information is not feasible for an attacker host.
Fig. 7. Floodlight and ONOS controllers web-GUI shows the successful MHL fabrication attack: h3 is the attacking host, h3 creates the fake MHL (external link/indirect link) between SDN switches S1 and S2.

Fig. 8. An MHL between two SDN switches S1, S2.

Fig. 9. The path follows by switch-traffic packets.

3) The latency of the control channel, named switch-to-controller (STC) latency.

The SDN controller observes the traffic originated from the legacy network to detect the attacker. The controller can monitor the possible scenarios:

Case 1: The controller do not receive a Packet_In message that contains the switch-traffic originated from the legacy network;

Case 2: Packet_In message containing switch-traffic and host-traffic is received at the controller;

Case 3: Packet_In message containing switch-traffic is only received at the controller.

If Case 1 happens, then a passive attacker fabricates the MHL, as described in Table I. Case 2 presents all the trades of legitimate MHL, and thus, MHL can be legitimate or fake. The SDN controller further verifies MHL to examine the existence of legacy switches in MHL, as given in section IV-C. If Case 3 happens, then the MHL is verified as fake.

Monitoring time: We define the monitoring time (MT) as the maximum waiting time for switch-traffic to reach the controller via MHL’s end SDN switch-ports. The time duration for a packet of switch-traffic to reach the controller via MHL’s end SDN switch-ports depends upon the following factors:

1) The latency of MHL;
2) The time interval of control packet generation by legacy switches;
3) The latency of the control channel, named switch-to-controller (STC) latency.

The presence of legacy layer-2 switches is validated by testing the legacy switch’s mac-learning capability. MAC-learning is the process of associating the MAC address to the interface/port of the switch at which the host-generated packet is received. The legacy switch learns and stores the host’s MAC address to port mapping when it receives an Ethernet packet containing an unlearned host’s MAC address in the source MAC address header field. The basic idea is that if the MHL’s end switches receive the host-generated packet originated from the legacy network, the legacy switches in the forwarding path have learned the ports from where the source host is reachable. The scenario is depicted in Fig. 10(i), host
H1 generated packet reaches to SDN switch-port \((S_i, px)\) the ingress switch-port. Switch \(S_i\) forwards it as a Packet_In to the SDN controller. All the intermediate legacy switches \((4,3,1)\) in the forwarding path learn the H1’s MAC address to port mapping. The legacy switches \((4,3,1)\) direct the probe packet to the previously learned path if a packet destined to the H1’s MAC address is forwarded by the same ingress switch-port \((S_i, px)\) from where the controller receives the host-traffic in Packet_In, as shown in Fig. 10 (ii).

To validate the presence of the legacy switch in the MHL, the SDN controller creates a probe packet with the destination address as the MAC address of any host connected in the legacy network. The host MAC addresses are the previously collected source MAC addresses from the Packet_Inss containing the host-traffic generated by a host connected in the legacy network. The controller sends the probe packet in a Packet_Out to the end switch-port of MHL from where the host generated packet was received, i.e., ingress switch-port. If the same probe packet is received back as Packet_In from another end switch-port of MHL, the MHL is verified as fake. However, if there exists a legitimate MHL, as shown in Fig. 10, the probe packet is forwarded to the previously learned path towards host H1 and will not be received back at another end switch-port \((S_j, py)\) of MHL, as shown in Fig. 10 (ii). Thus, MHL is verified as legitimate.

**Claim 1:** The probe packet destined to host MAC connected in the legacy network must not reach back to the other end switch-port of MHL.

**Proof:** The probe packet can be reached back to the other end switch-port only if the legacy switches broadcast the probe packet.

We prove the ‘claim 1’ by contradiction:

Let us assume the probe packet destined to host MAC is broadcasted by a legacy switch contained in an MHL. If the probe packet is forwarded from MHL’s end switch-port \((S_i, px)\), it reaches to the legacy switch 1. Legacy switch 1 has the mapping of host MAC to port ‘p5’; thus, legacy switch 1 forwards the probe packet to previously learned port p5, and probe packet is not broadcasted. Similarly, the probe packet will be forwarded by intermediate legacy switches in the previously learned path towards host H1. Hence, the probe packet is not reached back to the other end switch-port \((S_j, py)\).

**Waiting Time for Probe Packet:** The controller waits for Packet_In containing the probe packet within a timeout greater than the link latency. The relay time of an MHL depends upon the link latency and switch-to-controller latency, as shown in Fig. 11. Thus, the relay timer of an MHL is:

\[ T_r(MHL) = 2 \times \text{Link latency}(MHL) + \text{STC}(MHL’s \ switches). \]

The controller’s waiting time is significantly larger than the maximum time a packet takes to traverse the MHL, as we included twice the link latency.

**D. MHL Verification Process:**

Algorithm 1 presents the MHL verification process. It eliminates the fabricated MHLs created by the passive attacker, and the active attackers are detected using switch learning validation given in Algorithm 2.

**Algorithm 1 MHL verification process.**

**Input:** MHL L1, end switch-port\(((S_i, px), (S_j, py))\).

**Output:** MHL (L1) is legitimate or not (True/False).

1: Monitor end switch-port\(((S_i, px), (S_j, py)))
2: Start timer for switch-traffic \(T_m(L1))
3: if \(T_m(L1)\) timeout then
4: if No Packet_In containing STP’s BPDU received at \(((S_i, px), (S_j, py))\) then
5: return False;
6: end if
7: end if
8: if Packet_In containing host-traffic received then
9: Record host-traffic info (HTI) of L1.
10: Perform switch-learning validation by “Algorithm 2”;
11: return Algorithm 2 result;
12: end if
13: return False;

Algorithm 2 verifies the MHL using the MAC-learning based validation of legacy switch existence in the MHL. The SDN controller constructs a probe packet destined to the Source MAC address from the previously collected host-traffic information. We utilize the ARP reply as a probe to test the MAC-learning process, i.e., a tailor-made ARP reply packet, as depicted in Fig. 12 for switch learning validation. Hybrid-Shield uses host-traffic information, i.e., source MAC and IP addresses as the destination MAC and IP addresses of the ARP reply packet. The controller uses randomly generated MAC and IP addresses as source MAC and IP addresses of the ARP reply packet. The SDN controller sends the probe packet wrapped in Packet_Out at MHL’s end switch-port, i.e., the ingress switch-port of host-traffic and monitor at MHL’s another end switch-port. If the same probe packet is received back as Packet_In at another end switch-port, it indicates that
Algorithm 2 Validation of switch-learning.

**Input:** MHL L1, end switch-port((Si, px), (Sj, py)) Link latency L1, switch-to-controller (STC) latency for Si, Sj Host-traffic info (HTI) of L1.

**Output:** MHL (L1) is legitimate or not (True/False).

1: Create the probe packet using HTI(L1): (probe).
2: Extract ingress switch-port from HTI(L1): SP in.
3: Record other end switch-port of L1: SP o.
4: if SP in == Si then
5:   Sj → SP o;
6: else
7:   Si → SP o;
8: end if
9: Send (probe) as Packet-Out to the SP in, start relay timer;
10: while \( T_r(L1) = 2 \times \text{link latency}(L1) + \text{STC}(S_i) + \text{STC}(S_j) \)
11:   if probe receives back form SP o then
12:     return False;
13: end if
14: end while
15: return True;

an adversary is just relaying the traffic over the MHL, as shown in Fig. 13. If there is an original legacy network, then the probe packet must be forwarded to the previously learned path as shown in Fig. 14 and must not receive back at the MHL’s another end switch-port.

The validation of switch learning assures the presence of legacy switches in the MHL. After the new MHL passes the verification process, the link is included in the SDN controller’s topology view.

Fig. 12. ARP reply probe packet.

**E. Hybrid-Shield’s System Design**

Hybrid-Shield aims to provide real-time, accurate verification of MHLs. Hybrid-Shield utilizes the basic functionalities of SDN. It monitors the traffic at MHL and validates legacy switches in the MHL. Fig. 15 presents the architecture of Hybrid-Shield that includes the switch-port monitor and link verifier modules.

Hybrid-Shield intercepts before MHLs are included in the controller’s topology view. The link verification intercepts the link discovery process, and whenever the link discovery module discovers a new MHL, the link verifier performs the MHL verification process explained earlier in subsection IV-A. The link verifier implements the MHL verification strategy to verify the legitimacy of a newly discovered MHL. The event handler and link discovery modules are part of the SDN controller’s basic functionalities, which are utilized to implement the MHL verification strategy.

1) **Link Discovery Module:** Hybrid-Shield prevention system includes a dynamic MHL verification method, which verifies every MHL update before its actual use. It implements the SDN controller’s link discovery process. It periodically sends the LLDP and BDDP packets to discover the direct and multi-hop links in the data plane. The controller collects link
ly latency values using the TLV fields of LLDP/BDDP packets by utilizing the link-discovery process. Upon MHL discovery, the Link verifier records link latency in the MHL mapping Table I. Whenever an MHL is discovered, the MHL update is received at the Link verifier, and it further performs the MHL validation to prevent fake MHL injection.

2) Switch-port Monitor: The switch-port monitor captures the switch-traffic and host-traffic at the newly discovered MHL’s end switch-ports. It receives Packet_In messages from the controller’s event handler module and forwards the packets received from the end switch-ports of MHLs to the link verifier module.

| Link [NodePortTuple]          | Latency (ms) | Host-traffic Info        |
|-------------------------------|--------------|--------------------------|
| L1{(s1,p1), (s2,p2)}         | x            | h1-mac, h1-IP, in-port(s1,p2) |
| L2{(s3,p3), (s4,p4)}         | y            | h5-mac, h5-IP, in-port(s3,p3) |
| L3{(s5,p5), (s6,p6)}         | z            | h10-mac, h10-IP, in-port(s5,p5) |
| L4{(s7,p7), (s8,p8)}         | a            | h2-mac, h2-IP, in-port(s7,p7) |

3) Link Verifier: The link verifier module verifies a newly discovered MHL by implementing the MHL verification strategy. The link verifier manages an MHL mapping table to store the host-traffic information received at the MHL’s end switch-ports. The mapping table entry contains the source host’s MAC and IP addresses and packet’s ingress switch-port named “in-port”, collected from the host-traffic packets received at the MHL’s end switch-ports, as shown in Table I. The host-traffic information is utilized to construct probe packets for implementing the switch learning validation, as given in section IV. After performing the MHL verification, the link verifier sends the result to the link discovery module, and legitimate MHLs are included in the controller’s topology view.

F. Security Analysis

The link fabrication attack can be possible in two ways, either by forged link discovery protocol packets (LLDP/BDDP) injection or by relaying link discovery protocol packets (LLDP/BDDP). We proposed defense against the relay-based MHL fabrication attacks that is not defended by simply extending the protections of direct link fabrication [6], [12], [24] to the MHL discovery. Similar to the direct link fabrication discussed by Hong et al. [6], the MHL fabrication can also be performed by forged BDDP packets injection. However, the existing authentication-based defenses can be utilized for BDDP to protect the MHL fabrication by forged BDDP packets in the hybrid SDN deployments.

Existing authentication-based defenses [6], [20], [21] for LLDP can handle the attacks performed by generating fake BDDP packets. Hong et al. [6] proposed the idea to add an additional authenticator TLV containing the controller’s signature in the LLDP packet. Alharbi et al. [20] proposed an HMAC based packet authentication. Marin et al. [21] suggested the computation of authenticator TLV over the DPID, port number, and a time-stamp using a cyclic cryptographic key. The authentication mechanisms proposed by Alharbi et al. [20] and Marin et al. [21] provide the integrity protection and freshness to defend against the link fabrication attack by fake LLDP packet injection that can also be implementable for MHL discovery and protect against fake BDDP injection.

Further, to provide total protection from the link fabrication attacks, the defense approaches of direct link fabrication attack and MHL fabrication attack both need to be utilized in the hybrid SDN deployments. Hybrid-Shield can coexist with the other defense mechanisms for direct link fabrication attacks. Hybrid-Shield verifies the newly discovered MHLs, and it does not interfere with the controller’s direct link discovery process. We will further explain how the Hybrid-Shield can coexist with the specific defense systems for direct link fabrication attacks based on either fake LLDP injection or BDDP relay. All the defense approaches [6], [12], [20], [21] using authentication require adding an extra authenticator TLV in the LLDP/BDDP packet that can work as it is to protect against the fake LLDP/BDDP based link fabrication. Other defense mechanisms for LLDP relay-based link fabrication attacks such as TopoGuard [6] and TopoGuard+ [12] maintain and verify the direct links between switches. Hybrid-Shield verifies MHLs and does not intercept with direct link discovery and verification. Thus MHL verification proposed in this paper can work along with direct link verification.

V. Evaluation

We present an evaluation of Hybrid-Shield based on our prototype implementation. In section V-B, the effectiveness of the system is measured in terms of quick detection and accuracy in terms of false alarms generated by the verification algorithm in different topological scenarios. In section V-C, the Hybrid-Shield’s performance is computed as overhead generated by Hybrid-Shield’s integration at the controller. Further, theoretical complexity analysis of Hybrid-Shield’s verification technique is presented.

A. Experimental Setup

We implement the prototype of Hybrid-Shield on top of the floodlight [17] controller. Hybrid-Shield integrates the MHL verification technique in the link discovery module of the floodlight controller. The hybrid SDN data plane topology is implemented in the mininet [22].

Attack Implementation: The attacker is implemented as a mininet host with two Ethernet interfaces. The attacking host connects to two victim SDN switches by two interfaces for performing the MHL fabrication attack. We implement the second type of attacking strategy, “Bddp-relay mimics legacy network” for evaluation (as given in Table I) because the detection of second type attackers covers the other two cases of attacking strategies. The attacker generates fake switch-traffic using a script to generate STP packets and fake host-traffic using the ping tool.

System Setup: Floodlight controller is installed over an Ubuntu system with Intel-Core-i5 CPU, 1.6 GHz, four cores, 16 GB RAM. Mininet emulates the data plane topology over the same system, and legacy switches are implemented using OVSKernelSwitch with standalone mode and STP.
Topologies: To evaluate the Hybrid-Shield, we utilize the four different topologies. The NSFNet and Cost266 topologies create the scenarios of varying the number of hops in an MHL, i.e., the length of MHL. We utilize one backbone topology of five universities supercomputer center NSFNet (Fig. 16), one dense ISP topology Cost266 (Fig. 17). The NSFNet and Cost266 both are studied by the researchers [25][26] for hybrid data plane implementation. We utilize the hybrid variation of both topologies, NSFNet, and Cost266. The other two topologies are simple tree topologies created for generating the small and large number of MHLs in the hybrid-data plane. The third topology is a tree topology with depth=3, fanout=2 named Tree-topo, the fourth large tree topology with depth=3, min degree =2 max degree=9, named Large-topo. The large-topo contains the maximum number of MHLs.

Our purpose of including four different topologies is to cover various topological scenarios of hybrid-SDN to investigate the link verification accuracy. The fourth topology Tree-topo is constructed for performance comparison with a small topology. Note: All topologies are implemented as one broadcast domain controlled by a single SDN controller.

### B. Effectiveness Evaluation

1) Detection of Fake MHL: One attacking host is created in the tree topology, i.e., Tree-topo, to evaluate the accuracy of Hybrid-Shield. It performs the attack by connecting two random switches in the network. Fig. 18 shows the Hybrid-Shield’s controller log of detection of fake MHL.

2) Attack Detection Time: We evaluate the effectiveness in terms of detection time. In this experiment, the number of attacks (i.e., fake MHL injection) is performed per second varies between 1,5,10. The experiment runs 20 times, and the measured detection time is presented in Fig. 19. Hybrid-Shield initially monitors for switch-traffic and host-traffic in the newly discovered link. After switch-traffic and host-traffic are received, further verification takes the time equivalent to link latency. Therefore, the Hybrid-Shield detection process is affected by the reception of the first packet of switching and host traffic from an unverified MHL’s end switch-ports. When the number of simultaneous MHL updates increases, the host-traffic may take some additional time to reach the controller. Therefore, the maximum detection time value is affected by the number of MHL updates/sec, as shown in Fig. 19.
shows that the Hybrid-Shield takes less than 4.5 ms for detecting fake MHL in the hybrid data plane. Hybrid-Shield is capable of quick detection of fabricated MHLs in the hybrid data plane.

3) False Alarms: Hybrid-Shield’s verification accuracy is examined using three different topologies, NSFNet, Tree-topo, and Cost266. In this experiment, the legacy switches and SDN switches all start simultaneously in mininet emulation. The false positive rate refers to the percentage of legitimate MHLs verified as fake links, i.e., false verification. We measure the false alarms generated by Hybrid-Shield in the three topologies containing MHLs of different lengths. The random hosts connected to legacy switches generate host-traffic in hybrid topology. The Hybrid-Shield monitors and collects incoming traffic at the MHL’s end switch-ports for the period of monitoring time $T_m$ for switch-traffic, and the waiting time for host-traffic is ten seconds. If any switch-traffic and host-traffic is not received before the timeouts, then a legitimate link is verified as fake. In the case of a hybrid SDN data plane, legacy switches calculate the spanning tree using STP. The discovery of an MHL before the STP convergence can also trigger the false alarm, further discussed in detail.

STP convergence in the legacy part of the MHL coincides with the controller’s probe generation period and can trigger the false positive scenarios. During the verification process of an MHL, if the spanning tree changes in the legacy part of the MHL, then the probe packet may traverse back to the MHL’s another end switch-port and generate a relay-like scenario that triggers a false positive verification at Hybrid-Shield. Due to changes in the layer-2 topology, the legacy switches temporarily lowers the MAC learning time. Suppose the learned MAC addresses are aged at the time when probe packets traverse the legacy network. In this case, probe packets destined to previously learned host MAC addresses are broadcasted, triggering the relay-like scenario and causes false positive verification.

In our implementation of topology, legacy switches and SDN switches start simultaneously in the mininet. Therefore, there is the possibility of coinciding the STP convergence with the discovery and verification of MHL that could cause the false positive cases.

Fig. 20 compares the false positive rate for five runs of Hybrid-Shield’s verification over all three topologies. The Cost266 topology contains the MHLs of various lengths (from 2 hops to 8 hops), and Cost266 is the most complex topology. The false positive rate is stable at around 2.5% for Cost266, which shows the Hybrid-Shield verification technique pauses a stable false alarm rate in a complex topology. We observe that in all the experiments, the false alarm rate is lower than 6.5%. Hence, Hybrid-Shield demonstrates high accuracy in various dynamic scenarios of the hybrid data plane. Table IV presents the experimental results of the scenarios where fake MHLs coexist with the legitimate links in all three topologies. The 20 runs of the experiment are performed and the average false positive rate is presented with the 95% confidence interval. The false negative rate is the percentage of fabricated links verified as legitimate. The attacker fabricates the link by relaying packets between two victim SDN switches. Therefore, the probe packet must be received back to the other end of the link. Hence verification of the fabricated link does not fail. The experimental results show that the Hybrid-Shield successfully detects the fake links as the false negative rate is zero.

C. Performance Evaluation

The Hybrid-Shield verification framework is integrated into the controller’s link discovery module that introduces additional processing costs. Hybrid-Shield’s monitoring function introduces extra processing delay for packet inspection. The link verification process causes a delay in the construction of the probe packet. We evaluate the overhead produced by Hybrid-Shield with different network topology scales, as presented in Table III. The processing cost is evaluated at small and large-scale topologies. Tree-topo is used for small topology, and Large-topo is used for large topology. Further, we measure the CPU usage of the controller with and without Hybrid-Shield integration to evaluate the overhead caused by Hybrid-Shield.

1) Computation Overhead: The Hybrid-Shield framework inspects Packet_In messages input from MHL’s end switch-ports. The MHL verification introduces some delay in the link discovery process. However, the processing overhead is caused only one time per MHL verification. We measure the packet processing and probing delay to quantify the overhead introduced by the Hybrid-Shield verification process at the controller, as shown in Table V and Table VI.

Hybrid-Shield performs probing when an MHL update arrives. The probing includes the construction of the probe packet and writing the Packet_Out message. The number of MHL updates in the Large topology is 36, much larger than four updates in small topology. The average probing delay

| Topology | Avg. false +ve rate | Avg. false -ve rate |
|----------|---------------------|---------------------|
| NSFNet   | 1.6                 | 0.0                 |
| Cost266  | 2.8                 | 0.0                 |
| Tree-topo| 1.7                 | 0.0                 |

Table IV: Comparison of False Positive and False Negative Rate of Hybrid-Shield’s Verification.
is 51 $\mu$s, which accounts for 12.6% of the total Packet\_In processing time (Total PT). The monitoring function causes the average packet inspection delay of 25$\mu$s that accounts for 6.2% of total controller processing triggered by Packet\_In. Hence, we conclude that the Hybrid-Shield is lightweight to integrate into the SDN controller and scalable to manage large networks.

2) CPU Usage: We evaluated the CPU usage to measure the overhead caused by Hybrid-Shield integration at the controller. The experiment is performed with the small and large topologies and measured the CPU usages in two scenarios, with and without Hybrid-Shield used. To measure the CPU performance, we used the top command on the Linux shell. Table VII presented the average CPU usage of the controller with and without Hybrid-Shield.

3) Complexity Analysis: Following terms are used for complexity analysis:
- $n$ is the number of Packet\_Ins in the time period $T_c$(MHL).
- $m$ is the number of MHLs under verification.

Space Complexity: Hybrid-Shield manages the MHLs details and corresponding host-traffic information, only one entry of host-traffic information per link during the verification process. Thus, the memory requirement is bounded by $O(m)$. Hybrid-Shield writes two flow rules on both end switches of fake MHLs to block it, requiring $O(1)$ memory at the switches.

Time Complexity: Hybrid-Shield inspects the packets in the period of $T_c$ to find that the probe packet is relayed back or not. Thus, for a single MHL fabrication attack, the detection time is bounded by $O(n)$.

VI. LIMITATIONS AND DISCUSSION

This section presents the security analysis of Hybrid-Shield’s defense. We discuss how an attacker can learn the defense mechanism of Hybrid-Shield. If the attacker knows the defense, how can it breach the Hybrid-Shield’s protection?

We first discuss the difficulty for an attacker to learn the Hybrid-Shield’s defense mechanism.

1. Hybrid-Shield’s defense based on monitoring incoming packets from MHL and collecting the data from host generated packets is a passive activity that does not reveal any information to learn the defense’s presence.

2. Hybrid-Shield’s switch learning validation approach involves probing to test the MAC-learning process of layer-2 legacy switches. The probing packets are not directed to the attacking host’s MAC and IP addresses and indistinguishable from regular traffic. Further, we used the unicast ARP reply as a probe packet to test the MAC-learning process, not a host liveness probing, such as ICMP ping, TCP SYN, ARP ping. Thus, the controller’s probe packet is not easy to identify even if the attacker actively senses the incoming traffic at MHL.

If the adversary somehow learns the defense, it can bypass the defense when the active MHL fabrication attack is launched. The active attacker can generate the host-traffic with spoofed source MAC and IP addresses and keep the record of host-generated traffic to identify the probe packets. Hybrid-Shield may generate the probe packets destined to the same spoofed MAC and IP addresses. Then the attacker can bypass the defense by dropping all the packets destined to the spoofed MAC and IP addresses of previously generated host-traffic. However, this mechanism works when the attacker targets the SDN switches. In the case of legacy switches, probe packets destined to other host’s MAC and IP addresses cannot be filtered out. Furthermore, MHL fabrication attack and defense while attackers target legacy switches can be challenging to explore.

VII. RELATED WORK

In a traditional network, topology poisoning is easy to perform, such as injection of fake BGP packets and STP mangling attacks [27]. SDN avoids many attacks in distributed routing protocols of the traditional network by providing centralized control over all data plane functionalities, such as mitigate the routing problems in BGP [28][29]. However, the centralized SDN architecture is vulnerable to topology poisoning attacks known as link fabrication and host location hijacking. The attacker exploits the communication between the controller and data plane devices to introduce false topology information. In this section, we discuss the various link fabrication attacks in SDN and existing countermeasures. Further, we explore
whether the existing countermeasures of link fabrication in SDN works in hybrid-SDN or not? Furthermore, we present the various protocols proposed for secure link discovery and investigate their feasibility against the MHL fabrication attacks.

The link fabrication attacks are performed in two ways: i) by injecting fake packets of topology discovery protocol and ii) by relaying the legitimate discovery protocol packets. The attacker poisoned the controller’s view by inserting the false link information. These fake links mislead the applications that forward the data plane traffic and implement effective optimizing services, such as QoS, Shortest Path, and Slicing. Researchers broadly utilized the following defense techniques to mitigate the link fabrication attacks:

- Authentication of control packets, such as LLDP authentication.
- Behavior analysis of network elements, such as link latency, port type.

The LLDP authentication fails against the relay-based link fabrication attacks. Therefore most of the proposed defenses are based upon behavior analysis. Huang et al. presented TopoGuard [6], a security extension for topology discovery. TopoGuard studied how an attacker host can easily inject the fake link by generating the fake LLDP packets or relaying the legitimate LLDP packets. Topoguard’s defense uses the behavioral property that is the type of device connected to the switch port, and they classify the device type by HOST, SWITCH, or ANY. If host-generated traffic (DNS/ARP/ICMP) is received at a switch port which is generated from an unknown source MAC address, TopoGuard declares the port as HOST type. Topoguard’s approach defines an anomaly that "a Host type switch port cannot participate in LLDP propagation." However, in the case of an MHL, the end switch ports are connected to legacy switches that can forward the host-generated packets and link discovery and other control messages to the MHL. Therefore, from an MHL’s end switch-ports from where the host-generated packets are received containing an unknown source MAC address, the BDDP and other control messages can also be received. Hence, the port property cannot determine an anomaly for the fake MHL. Moreover, an MHL’s end switch-ports cannot be labeled as "HOST" by receiving the unknown host-generated traffic.

Sphinx [11] uses the incremental flow graph to find anomalous behavior in the data plane. Sphinx does not target a specific attack, but it observes the differences between the controller’s network state and flow metadata. It then detects the attack at network topology discovery and data plane forwarding. Sphinx’s flow graphs can perceive incorrect messages as normal behavior if most messages are altered. Thus Sphinx cannot defend against relay-based link fabrication attacks. Alimohammadifar et al. [24] proposed SPV (stealthy probing-based verification) based detection against link fabrication attacks. SPV works on the assumption that the compromised switch in the fake link cannot distinguish the stealthy probing packets. The SPV fails if an adversary relays all the packets and behaves like the original link.

The authors of TopoGuard+ [13] addressed a new link fabrication attack named Port Amnesia. Topoguard+ distinguishes between the legitimate and fake links by inspecting inter-switch link latency and defend against link fabrication by an attacker relaying the LLDP using an out-of-bound channel. Huang et al. [30] proposed TrustTopo, a new topology verification scheme. TrustTopo uses dynamic password generation and password timeouts for link verification in SDN. The password timeouts are computed based on the weighted average of the LLDP delay of direct links. Smyth et al. [31] also proposed the defense based on statistical testing of link latency threshold. The latency measurement [12], [13], [31] and dynamic passwords with LLDP timeouts [30] are no-win as the variable length of the MHL can contain many switches in between. Therefore we could not bound the maximum link latency or LLDP delay-based timeouts.

Furthermore, Marin et al. [21] analyze the existing countermeasures of topology poisoning attacks, such as TopoGuard, TopoGuard+, SPV, and Securebinder, and find the new vulnerabilities and attacks. Hauser et al. [32] proposed the P4-MACsec to protect the SDN data plane. They implemented the MACsec authentication mechanism for LLDP frames, P4-MACsec protects against the fake LLDP injection. However, the authentication-based mechanisms could not defend the LLDP relay-based attack. Hence defenses against direct link fabrication are insignificant for the MHL fabrication attack, and none of these solutions [12], [31], [13], [30] studied the link fabrication attacks in hybrid SDN deployment.

Link Discovery Protocols: All of the existing security frameworks against topology poisoning consider the OFDP. Researchers proposed some other link discovery protocols, OFDPv2 [33], softdp [34], SLDP [35], and LADP [36], to provide more efficient and secure link discovery in SDN. These discovery protocols explore security against link fabrication attacks and flooding attacks. The OFDP is a widely implemented protocol, and switching to other discovery protocols requires changes in existing architecture and standards. However, only the SLDP [35] considers discovering the unidirectional links containing legacy switch (multi-hop/external link) among all these discovery protocols. The SLDP uses the random source MAC address in the discovery packets to mitigate fake discovery packet injection. The SLDP does not mitigate the relay-based link fabrication attacks where the attacker hosts are connected to two SDN switches (e.g., port amnesia, silent-relay). Thus, even the SLDP implementation could not mitigate the MHL fabrication attack in hybrid SDN.

Furthermore, Alvarez-Horcajo et al. [37] proposed a link discovery protocol that discovers the entire hybrid topology in WSN. However, HDDP is not practical in the wired scenario because it requires modification over legacy switches to support HDPD.

VIII. CONCLUSION AND FUTURE WORK

This paper introduces the MHL fabrication attack in the hybrid SDN. We proposed a prevention framework, “Hybrid-Shield,” which includes the novel link verification mechanism. Hybrid-Shield prevents the injection of fabricated MHLs in the controller’s topology view and rescues the data plane and control plane from the attack’s severe effects. We evaluate
the prototype implementation of the Hybrid-Shield framework in the real SDN controller. The quick detection and high accuracy demonstrate the effectiveness of the Hybrid-Shield verification technique. Hybrid-Shield is lightweight as it introduces less overhead at the controller. Hybrid-Shield can be easily deployed at any SDN controller as it uses only basic functionalities of SDN, i.e., monitoring of Packet_In messages and Packet_Out for probing.

Securing the hybrid SDN deployment needs further exploration in the direction of the control plane and data plane both. Further analysis of hybrid SDN architecture to discover the vulnerabilities that can introduce the network threat is most important.

REFERENCES

[1] S. Vissicchio, L. Vanbever, and O. Bonaventure, “Opportunities and research challenges of hybrid software defined networks,” ACM SIGCOMM Computer Communication Review, vol. 44, no. 2, pp. 70–75, 2014.

[2] X. Huang, S. Cheng, K. Cao, P. Cong, T. Wei, and S. Hu, “A survey of deployment solutions and optimization strategies for hybrid sdn networks,” IEEE Communications Surveys & Tutorials, vol. 21, no. 2, pp. 1483–1507, 2018.

[3] K. Caria, T. Das, A. Jukan, and M. Hoffmann, “Divide and conquer: Partitioning osf networks with sdn,” in 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). IEEE, 2015, pp. 467–474.

[4] H. Xu, X.-Y. Li, L. Huang, H. Deng, H. Huang, and H. Wang, “Incremental deployment and throughput maximization routing for a hybrid sdn,” IEEE/ACM Transactions on Networking (TON), vol. 25, no. 3, pp. 1861–1875, 2017.

[5] S.-H. Tseng, A. Tang, G. L. Choudhury, and S. Tse, “Routing stability in hybrid software-defined networks,” IEEE/ACM Transactions on Networking (TON), vol. 27, no. 2, pp. 790–804, 2019.

[6] S. Hong, L. Xu, H. Wang, and G. Gu, “Poisoning network visibility in software-defined networks: New attacks and countermeasures.” in NDSS, vol. 15, 2015, pp. 8–11.

[7] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. Godfrey, and S. T. King, “Debugging the data plane with anteater,” in ACM SIGCOMM Computer Communication Review, vol. 41, no. 4. ACM, 2011, pp. 290–301.

[8] D. Levin, M. Canini, S. Schmid, F. Schaffert, and A. Feldmann, “Panopticon: Reaping the benefits of incremental (SDN) deployment in enterprise networks,” in 2014 {USENIX} Annual Technical Conference ({USENIX} {ATC} ’14), 2014, pp. 333–345.

[9] C. Jin, C. Lumezanu, Q. Xu, H. Mekky, Z.-L. Zhang, and G. Jiang, “Magneto: Unified fine-grained path control in legacy and openflow hybrid networks,” in Proceedings of the Symposium on SDN Research, 2017, pp. 75–87.

[10] S. Vissicchio, L. Vanbever, and J. Rexford, “Sweet little lies: Fake topologies for flexible routing,” in proceedings of the 13th ACM Workshop on Hot Topics in Networks, 2014, pp. 1–7.

[11] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, “Sphinx: Detecting security attacks in software-defined-networks.” in NDSS, vol. 15, 2015, pp. 8–11.

[12] R. Skowrya, L. Xu, G. Gu, V. Dethia, T. Hobson, H. Okhravi, and J. Landry, “Effective topology tampering attacks and defenses in software-defined networks,” in 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2018, pp. 374–385.

[13] P. Shrivastava, A. Agarwal, and K. Kataoka, “Poster: Detection of topology poisoning by silent relay attacker in sdn,” in Proceedings of the 24th Annual International Conference on Mobile Computing and Networking. ACM, 2018, pp. 792–794.

[14] S. Gao, Z. Li, B. Xiao, and G. Wei, “Security threats in the data plane of software-defined networks,” IEEE network, vol. 32, no. 4, pp. 108–113, 2018.

[15] Openflow protocol. [Online]. Available: https://www.opennetworking.org/ openflow.

[16] Pox:sdn controller. [Online]. Available: www.noxrepo.org/pox/about-pox.

[17] Floodlight openflow controller. [Online]. Available: http://www.projectfloodlight.org/floodlight/

[18] Opendaylight: Towards a model-driven sdn controller architecture. [Online]. Available: https://github.com/opendaylight/controller

[19] Onos - a new carrier-grade sdn network operating system designed for high availability, performance, scale-out. [Online]. Available: https://www.opennetworking.org/onos/

[20] T. Alkharbi, M. Portmann, and F. Pakzad, “The (in) security of topology discovery in software defined networks,” in 2015 IEEE 40th Conference on Local Computer Networks (LCN). IEEE, 2015, pp. 502–505.

[21] E. Marin, N. Bucciol, and M. Conti, “An in-depth look into sdn topology discovery mechanisms: Novel attacks and practical countermeasures,” in Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 1101–1114.

[22] Mininet, a network in a laptop: rapid prototyping for software-defined networks. [Online]. Available: http://mininet.org/

[23] C. Jin, C. Lumezanu, Q. Xu, Z.-L. Zhang, and G. Jiang, “Telekinesis: Controlling legacy switch routing with openflow in hybrid networks,” in Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, 2015, pp. 1–7.

[24] A. Alimohammadiar, S. Majumdar, T. Madi, Y. Jarraya, M. Pourzandi, L. Wang, and M. Debbabi, “Stealthy probing-based verification (spv): An active approach to defending software defined networks against topology poisoning attacks,” in European Symposium on Research in Computer Security. Springer, 2018, pp. 463–484.

[25] C.-Y. Cha, K. Xi, M. Luo, and H. J. Chao, “Congestion-aware single link failure recovery in hybrid sdn networks,” in 2015 IEEE Conference on Computer Communications (INFOCOM). IEEE, 2015, pp. 1086–1094.

[26] M. Caria, A. Jukan, and M. Hoffmann, “Sdn partitioning: A centralized control plane for distributed routing protocols,” IEEE Transactions on Network and Service Management, vol. 13, no. 3, pp. 381–392, 2016.

[27] A. Barbir, S. Murphy, and Y. Yang, “Generic threats to routing protocols,” 2006.

[28] V. Kotronis, X. Dimitropoulos, and B. Ager, “Outsourcing the routing control logic: better internet routing based on sdn principles,” in Proceedings of the 11th ACM Workshop on Hot Topics in Networks. ACM, 2012, pp. 55–60.

[29] S. T. Ali, V. Sivaraman, A. Radford, and S. Jha, “A survey of securing networks using software defined networking,” IEEE transactions on reliability, vol. 64, no. 3, pp. 1086–1097, 2015.

[30] X. Huang, P. Shi, Y. Liu, and F. Xu, “Towards trusted and efficient sdn topology discovery: A lightweight topology verification scheme,” Computer Networks, vol. 170, p. 107119, 2020.

[31] D. Smyth, S. McSweeney, D. O’Shea, and V. Cionca, “Detecting link fabrication attacks in software-defined networks,” in 2017 26th International Conference on Computer Communication and Networks (ICCCN). IEEE, 2017, pp. 1–8.

[32] F. Hauser, M. Schmidt, M. Habele, and M. Menth, “P4-macsec: Dynamic topology monitoring and data layer protection with macsec on p4-based sdn,” IEEE Access, vol. 8, pp. 58,845–58,858, 2020.

[33] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, “Efficient topology discovery in software defined networks,” in 2014 8th International Conference on Signal Processing and Communication Systems (ICSPCS). IEEE, 2014, pp. 1–8.

[34] A. Azzouni, R. Boutaba, N. T. M. Trang, and G. Pujolle, “Sofdpy: Secure and efficient openflow topology discovery protocol,” in NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2018, pp. 1–7.

[35] A. Nehra, M. Tripathi, M. S. Gaur, R. B. Battula, and C. Lal, “Sldp: A secure and lightweight link discovery protocol for software defined networking,” Computer Networks, vol. 150, pp. 102–116, 2019.

[36] Y. Jia, L. Xu, Y. Yang, and X. Zhang, “Lightweight automatic discovery protocol for openflow-based software defined networking,” IEEE Communications Letters, 2019.

[37] J. Alvarez-Horcajo, E. Rojas, I. Martinez-Yelmo, M. Savi, and D. Lopez-Pajares, “Hddp: Hybrid domain discovery protocol for heterogeneous devices in sdn,” IEEE Communications Letters, 2020.