This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials

Topology Discovery in Software Defined Networks: Threats, Taxonomy, and State-of-the-art

Suleman Khan [1,2], Student Member, IEEE, **Abdullah Gani** [1,2], Senior Member, IEEE, **Ainuddin Wahid Abdul Wahab** [2], Member, IEEE, **Mohsen Guizani** [3], Fellow, IEEE, **Muhammad Khurram Khan** [4], Senior Member, IEEE

[1] Center for Mobile Cloud Computing Research (C4MCCR), University of Malaya, Malaysia
[2] Faculty of Computer Science & Information Technology, University of Malaya, Malaysia.
[3] College of Engineering, University of Idaho, USA.
[4] Center of Excellence in Information Assurance (CoEIA), King Saud University, Saudi Arabia

Corresponding Authors: Suleman khan, Abdullah Gani (sulemankhan1984@yahoo.com, abdullah@um.edu.my)

ABSTRACT

The fundamental role of the Software Defined Networks (SDN) is to decouple the data plane from the control plane; thus providing a logically centralized visibility of the entire network to the controller. This enables the applications to innovate through network programmability. To establish a centralized visibility, a controller is required to discover a network topology of the entire SDN infrastructure. However, discovering a network topology is challenging due to 1) the frequent migration of the virtual machines in the data centers, 2) lack of authentication mechanisms, 3) scarcity of the SDN standards, and 4) integration of security mechanisms for the topology discovery. To this end, in this paper, we present a comprehensive survey of the topology discovery and the associated security implications in SDNs. The paper provides discussions related to the possible threats relevant to each layer of the SDN architecture, highlights the role of the topology discovery in the traditional network and SDN, presents a thematic taxonomy of topology discovery in SDN, and provides insights into the potential threats to the topology discovery along with its state-of-the-art solutions in SDN. Finally, this paper also presents future challenges and research directions in the field of SDN topology discovery.

1. Introduction

This survey focuses on the topology discovery such as the representation of the interconnection between connected peers in Software Defined Networks (SDN). The logically centralized controller collects the topology information from the network devices in the data plane of the SDN architecture. Maintaining a complete and accurate information of the network topology is utmost important and a prerequisite for various network management tasks including monitoring, diagnosing, and resource management. The topology information helps the controller to have an abstract view of the entire network [1], and enables a smooth

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials

and efficient network operation [2]. Moreover, the topology information is crucial for the core controller services in the control plane as well as the topologically dependent services in the application plane.

Since a centralized abstract view of the network topology discovery holds the key for SDN operation, it has drawn much attention of the research community in the past few years [3-5]. Note that, the centralized abstract view is based on a built-in topology discovery mechanism [4] and it strengthens the control capability of the controller over the entire network [6]. Intuitively, this exposes the SDN wherein the controller becomes a single point of failure. It follows that, the security of the topology discovery in protecting the controller from failure is a critical challenge to address [7].

Several threats to the security of the SDN architecture have been identified and discussed in the literature [8-15]. However, the most severe attacks are those affecting the control mechanism in SDN [8]. Once the attack succeeds in controlling the entire network, it can leak out the information from the network or perform other malicious behaviors [16]. In topology discovery, threats must be prevented as early as possible because they pose threat to other services in the application plane [17]. Furthermore, the vulnerabilities found in the network topology would ultimately affect the performance of the topology-dependent services because of their dependencies. While various proposals for topology discovery exist [4, 18, 19], still these proposals are premature in making topology discovery in SDN truly secure and scalable. To this end, in this paper, our aim is to describe the security aspects of the topology discovery in detail. More importantly, we have focused on the topology discovery threats which affect the visibility of the network by exploiting different core functionalities of the controller.

To mark distinction of this study, in the following, brief descriptions of the few existing studies on SDNs are provided. The survey presented in [20] covers a comprehensive information about SDN including definition, benefits, and challenges. It provides insight knowledge about the layered architecture of SDN and explains its role in terms of OF protocol. The survey [21] provides an ample information about the current programmable network architectures used in wired and wireless networks such as SDN, Software-Defined Radio, and Network Functions Virtualization (NFV). The study in [22], surveys the security threats for each layer of the SDN architecture. The state-of-the-art in mitigating the security threats are analyzed. Finally, potential future research directions of SDN security are highlighted. Similarly, the work presented in [23] surveys the security attacks faced by the SDN along with its solutions. The survey analyzes both the security attacks and their corresponding solutions. That is, the network security enhancement based on the SDN framework is discussed for the attack investigation, detection, and prevention. The survey [24] discusses how the Distributed Denial-of-Service (DDoS) attacks can be mitigated through SDN in cloud computing and how SDN can be protected from becoming a victim of DDoS attacks. Furthermore, the role of SDN in a broad perspective is discussed in context with the emerging areas such as the big data, NFV, and information-centric networking. The survey presented in [15] classifies the SDN-based hypervisors with reference to their centralized and distributed architectures. In addition, exhaustive information regarding network attribute abstraction and isolation feature of SDN hypervisors is presented along with the future research directions.

To the best of our knowledge, this is the first comprehensive survey that provides insight about the topology discovery in the SDN architecture. The goal of this survey is to provide critical information about the topology discovery by describing its significance, working function, role in SDN, and security threats. Moreover, the proposed thematic taxonomy will assist in the classification of the topology discovery area into meaningful sub-groups for better and easy comprehension.

The key contributions of this survey are highlighted as follows:

— *Comprehensive background knowledge of SDN:* we provide information regarding the SDN and various threats to the SDN layered architecture.

— *In-depth information regarding topology discovery:* we highlight the importance of the topology discovery and discuss its role in the traditional networks and SDN.

— *Thematic taxonomy:* we devise a comprehensive thematic taxonomy to categorize the topology discovery into different groups i.e., objectives, controller platforms, dependent services, discovery entities, and controller services.

— *Discussion on topology discovery threats:* Classification of topology discovery threats is presented which explains the state-of-the-art security solutions, attack entities, controller vulnerabilities, attack types, and occurrence of the threats.

— *Introduce future research directions*: we provide potential research areas for topology discovery in SDN along with recommendations on possible solutions.

The remainder of this paper is organized as follows: Section 2 provides an overview of the SDN and potential threats to each of its layers. Section 3 describes the importance of the topology discovery and discusses its role in SDN and traditional networks. A thematic taxonomy of topology discovery is presented according to its discovery entities, controller platform, topology-dependent services, and objectives in Section 4. Section 5 provides a classification of topology discovery threats and solutions on the basis of the attack entities, current solutions, and further potential threats. Finally, Section 6 discusses and summarizes the potential future research areas of topology discovery with its possible solutions.

2. Background

2.1 Software Defined Networks (SDN)

The widely known separation of the control plane and the forwarding data plane is shown in Figure 1. This architecture results in numerous benefits, including easy insertion of applications and services, streamlined processes, improved efficiency, reduced complexity, and better user experience [25]. The control plane is controlled by logically centralized controller instead of the conventional control mechanisms present in the Border Gateway Protocol [26] and Open Shortest Path First (OSPF) [27]. The centralized control assists network administrators to dynamically change the network traffic without re-configuring the network devices. For instance, the controller can dynamically change the network flow towards high bandwidth channels while observing high delays on low bandwidth network channels without affecting the network operation [18].

In Figure 1, the SDN architecture is divided into three main layers/planes i.e., infrastructure, control, and application plane [28, 29]. The infrastructure plane consists of all the network devices that communicate and share information with each other [30]. For instance, the OF switches forward the packets towards the destination using rules specified by the controller.

The controller (in the control plane) acts as the brains of the SDN, which manages the entire network through the logically centralized controller [31],[16]. Moreover, the controller has the abstract view of the network topology that assist different applications running on top of the controller in the application plane [32]. The application plane is responsible for implementing essential network services (application, algorithms, protocols, etc.) through the controller [33]. With the given abstract network, the application plane deploys various network applications. These applications include load balancing [33], intrusion detection systems [34], network monitors [35], firewalls [36], and scheduling [37].

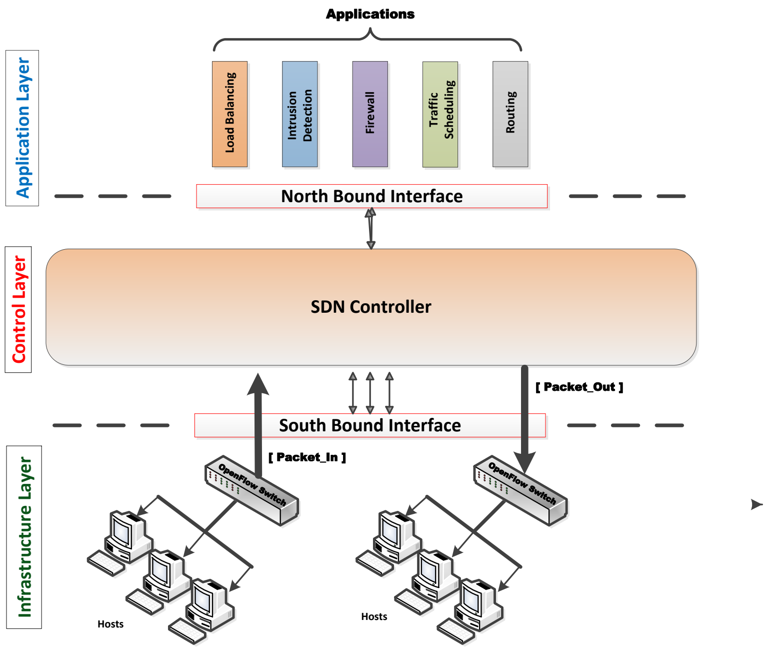

Figure 1: A general SDN layered architecture

The controller has different interfaces in order to communicate with other planes and network devices such as south, north, east, and westbound API's [38]. The most common API's used in SDN are the southbound and the northbound. The southbound API enables communication between the infrastructure plane network devices and the controller. Initially, when a new packet is received by the OF switch from the host, it checks for the matching field between the packet header and the flow rules in the flow table [5, 39]. If the match is not found, a *Packet_In* message is generated by the OF switch and it is sent to the controller on the southbound API. The controller checks the packet header for the necessary information and replies back to the OF switch through a *Packet_Out* message. The *Packet_Out* message contains the specific rules for the respective network flows which are inserted in the flow table of the OF switch. When a similar type of flow (i.e., the same source and destination) arrives at the OF switche, it is forwarded based on the previously inserted flow rule in the flow table [40]. The flow chart of the *Packet_In* message is shown in Figure 2.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials

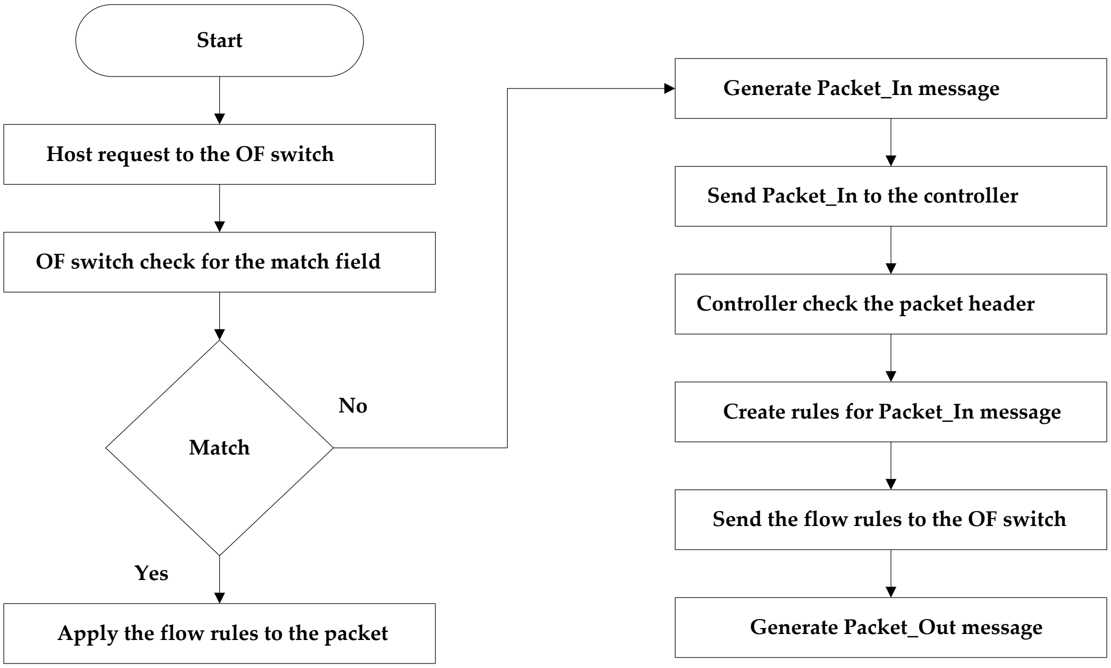
Start
Host request to the OF switch
OF switch check for the match field
Match
No
Yes
Apply the flow rules to the packet
Generate Packet_In message
Send Packet_In to the controller
Controller check the packet header
Create rules for Packet_In message
Send the flow rules to the OF switch
Generate Packet_Out message

Figure 2: A flow chart for *Packet-In* message

Moreover, the controller can modify the packet header information in real-time by modifying source/destination addresses and ports [41]. This characteristic of the controller provides flexibility and reliability to the network. Similarly, the northbound API connects the controller to various network applications that deploy algorithms and protocols to operate the SDN [42]. Unlike, the southbound API, the standard northbound API is not available yet, which presents several security threats [43, 44]. The eastbound and westbound API's manage the distributed controllers in SDN [7]. That is, multiple controllers can be deployed in SDN to manage different parts of the network [45] due to different assigned functions such as load balancing, monitoring and task allocation.

2.2 Threats to Software Defined Networks planes

The centralized control, network abstraction, and software-based network changes attract malicious users to perform attacks on SDN. Attacks can be on the 1) network devices in the infrastructure plane, 2) control modules in the control plane, 3) network devices in the application plane, or 4) different API's in the SDN [8]. In this section, we discuss and classify different attacks as illustrated in Figure 3. Moreover, we explain attacks performed on various interfaces of SDN. Table 1 illustrates the existing available solutions for each attack in the SDN planes and interfaces.

2.2.1 Attacks on the SDN data plane

There are different ways to maliciously exploit the network devices in the data plane wherein some of these attacks are specific to the SDN while others are inherited from the traditional networks. Each of these attacks is discussed below:

(a) *Malicious OF Switches:* Forwarding network flows through a malicious OF switch allows it to alter the network packets. In this case, the network flows divert and the legitimate traffic is dropped, which interrupts the communication between SDN devices. This can slow down the network traffic and may prevent the legitimate switch from receiving the traffic due to an excessive idle time specified for the flow entries in the flow tables. This can cause the network packets to be dropped [46] or generate numerous *Packet_In* messages to the controller due to mismatch at the OF switch.

(b) *Malicious hosts in the data plane:* Malicious hosts can attack any switch and controller in the SDN by generating forged network packets [47]. In forged network packets, various fields (such as the IP field, the MAC field or other fields), can be modified to hide the identity of the attacker. In addition, a malicious host can generate millions of packets in the form of a Denial-of-Service (DoS) attack to overload the memory of the OF switches [48]. Similarly, for every new forged packet (i.e., unique source IP address), the OF switches generates the *Packet_In* message to the controller which can result in decreasing the performance of the controller [49].

2.2.2 Attacks on the SDN control plane

The attacker is more interested in the control plane due to its significant function such as the network control, network abstraction, and support to various network applications. There are various types of attacks which can be performed by the controller as follows:

(a) *Malicious modules inside the controller:* The integration of the core controller functions creates an initial setup for the SDN. For instance, the topology manager stores information regarding devices such as switches and hosts in the network [50] and uses the Link Layer Discovery Protocol (LLDP) to discover the interconnected links between the OF switches [51]. An attacker can exploit vulnerabilities within these building blocks. As an example, , a recent topological poisoning attack [52] exploits the link discovery module running in the controller by generating fake links between the switches. As a result, the fake links affect the functionalities of the entire network [53].

(b) *Compromised controllers:* The controllers can also be distributed and exchange information from time-to-time to update their states [54]. The module for such distributed communication among controllers can be exploited by the attacker. . For instance, Open daylight controller uses ODL-SDNi app for distributed communication among multiple controllers. The problem arises when one of the controllers is performing maliciously and shares wrong information among the controllers. To identify the malicious controller among a pool of controllers is a challenging task due to isolated functionality of each controller. A malicious controller disseminates incorrect topological updates to another controller to make the network malfunction [55].

(c) *Attack on management consoles:* The management console allows authorized individuals to access the SDN. An attacker can get an unauthorized access to the management console through a password brute-force attack or leaking the password from different sources. Once the attacker gets access to the SDN, attacks can be generated on the controller as well as on different resources of the network. Usually, the access to management console is defined in the policy agent module of the controller. The compromised management console empowers the attacker to create a gateway in launching various other attacks on the SDN.

2.2.3 Attacks on the SDN application plane

The SDN application plane consists of different applications/software [56] for functions such as load balancing, routing, firewall, and intrusion detection. Moreover, these applications/software may be used to monitor the traffic, extract statistical traffic features, apply authentication mechanism to different user domains, and diverts the traffic based on the network etc. The application development in the application plane is considered as a dramatic change to the SDN architecture [57]. A single network infrastructure can be used by multiple applications at the same time to fulfill their requirements. However, this is not possible in the traditional network where the configuration of network device needs update upon using different network applications. A user can easily develop an application module and embed in the application plane [58]. This allows malicious users to affect the entire network. There are various possible attacks in the SDN application plane which are briefly discussed as follows:

(a) *Unauthorized access to applications:* An unauthorized access to these applications can help attackers bypass the security level of the controller [59, 60]. The controller treats all applications as normal network services because of the absence of a trust mechanism between the application and control layers. The unauthorized access to various applications can inform attackers about the operation of the network which further creates a chance to exploit various parts of the network.

(b) *Disclosure of information through the application server:* Once an attacker gains access to the application server, the information of any application that is currently or previously executed in the RAM can be accessed and disclosed. In a traditional network, such kind of attack is called a RAM scraper attack [61]. In SDN, an attacker can scan the RAM processes in the application server to gain access to the application information through the northbound API. An attacker can further identify the rules of the controller for various network flows.

(c) *Modification of user privileges for application execution:* Due to network virtualization, each user can treat the network with its own requirement provided with isolation [15, 62]. Each user is provided with specific rights to execute different applications according to its requirement. However, if the attacker accesses the application server, the user privileges can be changed to produce malfunctioning results [63].

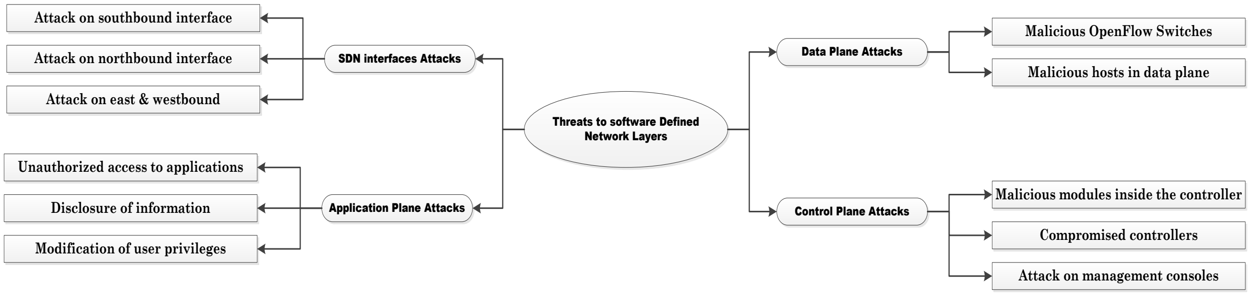
Attack on southbound interface
Attack on northbound interface
Attack on east & westbound
SDN interfaces Attacks
Unauthorized access to applications
Disclosure of information
Modification of user privileges
Application Plane Attacks
Threats to software Defined Network Layers
Data Plane Attacks
Malicious OpenFlow Switches
Malicious hosts in data plane
Control Plane Attacks
Malicious modules inside the controller
Compromised controllers
Attack on management consoles

Figure 3: Classification of attacks on SDN planes

2.2.4 Attacks on SDN interfaces

Note that, the southbound and northbound interfaces are used for the centralized controller environment while the eastbound and westbound interfaces are used in a distributed controller environment. These interfaces are used to send and receive network information which attracts attackers to eavesdrop [47].

(a) *Attacks on the southbound interface:* Mostly, the southbound interface in SDN uses the standard OF protocol [64]. The OF protocol is allows communication between the OF switch and the controller. Each OF switch has to communicate with the controller through *Packet_In* message upon reception of new packets [65]. This makes the southbound interface more suitable for information extraction from the *Packet_In* messages. The attacker can exploit the Transport Layer Security (TLS) vulnerabilities and to take control over the southbound interface [66]. Subsequently, the attacker can create, modify, and delete the flow rules. This causes malfunctioning results in the network due to malicious flow entries in the flow table. Moreover, an attacker can generate forged packets to the OF switches with a unique identity to force OF switch to generate a large number of *Packet_In* messages to overload the bandwidth channel used between the OF switches and the controller [67].

(b) *Attacks on the northbound interface:* The northbound interface is used for communication among the applications of the application plane [68]. Unlike the southbound interface, the northbound interface does not use a standard protocol because of its initial stage of development [43, 44]. The attacker can use the northbound interface to interfere with the communication between the application and the controller. An attacker can get unauthorized access to the northbound interface and may delete some information which can lead to falsified output of the application. Similarly, the attacker can use a malicious application to inform the controller to disconnect other applications leading to a flow rule modification problem. Moreover, the malicious application can send numerous requests to overload the CPU as well as to occupy the available bandwidth of the northbound interface. Note that, proper authentication and encryption mechanism is not standardized for the northbound interface. Various APIs for the northbound interface can increase the security threats because of the built-in vulnerabilities. This decreases the trustworthiness between the controller and various applications.

(c) Attacks on eastbound & westbound interfaces: The eastbound and westbound interfaces are also prone to various attacks. The information updates through these interfaces can be exploited by the attacker with an unauthorized access to the management console. The attacker can take advantage of unencrypted communication of data between controllers for sharing the network information updates [69]. An attacker can also compromise the network by tapping the application to eavesdrop on clear text communications between two controllers.

Table 1: SDN layers/Interfaces possible attacks and existing solutions

SDN Layers/Interfaces	**Possible Attacks**	**Existing Solutions**	**Attack Nature**
Data Plane	Malicious switches	FortNOX [22] , SDNsec [20]	SDN-based attack
	Malicious hosts	VAVE [21], OFGUARD [48], FlowVisor [70]	TN-based attack
Control Plane	Malicious modules	VeriCon [71], FRESCO [10], SPIRIT [72]	SDN-based attack
	Compromised controllers	Fleet [73] , DISCO [54], HyperFlow [74]	SDN-based attack

SDN Layers/Interfaces	Possible Attacks	Existing Solutions	Attack Nature
	Management consoles	Sandbox-based system [75]	SDN-based attack
Application Plane	Unauthorized access	PermOF [76], SDN Rootkits [77]	TN-based attack
	Disclosure of information	Proactive strategies and Randomization [78]	TN-based attack
	Modification of user privileges	OFX [79]	TN-based attack
SDN Interfaces	Southbound interface threats	VeriFlow [80], HSCS architecture [81]	SDN-based attack
	Northbound interface threats	Dynamic Filtering [82]	SDN-based attack
	Eastbound & westbound threats	DRS [83]	SDN-based attack

3. Topology Discovery

3.1 Importance

The topology management is a unique feature of SDN which allows the controller to facilitate the applications in the application plane[84]. For instance, a routing application uses the network topology to route the network traffic to its destination [75, 85, 86]. The controller discovers a topology through [52] a) Host discovery, b) Switch discovery, and c) Inter-connected links between the switches. The controller discovers the host by receiving a *Packet-In* message from the switch. The switches are discovered during the initial handshake process with the controller, and inter-connected links between switches are discovered through the OpenFlow Discovery Protocol (OFDP). However, there are vulnerabilities found in the core applications of the controller which are exploited to initiate topology poisoning attacks [87].

If an attacker poisons the network topology information, its effect will immediately be visible to all its dependent applications [88]. Therefore, it is important to detect a topology poisoning attack at an early stage. Note that, detecting a fake link between the OF switches created by the topology poisoning attack is relatively easy than identifying the source of the attack. Mostly, attackers hide their identity information after they perform the attack [89]. Similarly, in a topology poisoning attack, the attacker creates a fake link between the OF switches by spoofing the LLDP packet to hide his identity [19]. The controller should be aware of the fake links upon their insertion in the network so that the attack can be prevented at an early stage.

3.2 Topology Discovery in Traditional Networks

Topology poisoning attacks are not new to traditional networks. The main aim of a topology poisoning attack is to fabricate the network topology and disturb normal network operations in terms of control and management [90]. If a malicious router advertises its routing information to its neighbors, it will result in a falsified network traffic distribution based on the malicious routing information [91]. For instance, a network using the Routing Information Protocol (RIP) protocol allows each router to send its link with an update information of their topological view to its neighbors [92]. The information includes a destination identifier and a cost metric to the destination. However, the information sent by a malicious can update the neighbor routers link database with the wrong information and can affect the entire routing process [93]. Also, the malicious router may advertise for having the least cost path to a specific destination, thus causing the traffic to be diverted from other sources to a malicious destination [72, 73].

A similar type of attacks can also be performed in the link-state protocol, i.e., OSPF, where every router is bound to send its link update to its neighbors in order to calculate the optimal path depending on the metrics [94]. In OSPF, the link update information sent by the router is called Link State Advertisements (LSA). A malicious router may send a false LSA to its neighbors defining other routers by forging their original information [95]. This will divert the network traffic towards the malicious router which may forward the packet on to a longer path, perform eavesdropping, modify the packet information, and drop some/all the packets in the network flow. Besides the wired networks, topological information can be exploited in the wireless networks as well. For instance, the Optimized Link State Routing (OLSR) is used in mobile ad-hoc networks to discover and disseminate the link-state information throughout the network [96]. This information helps nodes to compute the optimal path to the next node in the network to reach the destination.

The OLSR determine and forward the link state information to the neighbor nodes by using *hello* and *topology control* messages. These messages can be falsified to disseminate the wrong information and results in a false topological development. Moreover, Bridge Protocol Data Units (BPDU's) in the Spanning Tree Protocol (STP) [97] can be forged to exploit the information. Such exploitation can be performed by an attacker to make the malicious switch as a root bridge in the network and therefore gain access to the network traffic. Such type of attack is also called an STP mangling [98]. The STP mangling affects the topology of the network in terms of selecting the wrong switch as a root bridge. The root bridge has an easy access to the network traffic that is costly when a malicious switch is selected as a root bridge in the selection process.

3.3 Topology Discovery in SDN

The topology management is a unique characteristic of SDN as compared to traditional networks. Table 2 provides a comparison between a traditional network and an SDN topology discovery. The decoupling of the control plane from the data plane enables the SDN to have a logically centralized control of the network [60, 99]. To achieve the centralized control, a controller (responsible to control the network centrally) should have a global visibility of the complete network [86]. A controller incorporates various core modules that assist in executing various SDN applications [100]. Among the core modules, a topology management is creates a topology of the entire SDN infrastructure [4]. The topology not only facilitates the controller but also assists the application plane service to perform its operation using the network programmability [101]. The network topology is significant to both the control plane and the application plane because it provides an abstract visibility of the entire network devices.

The OF protocol is a standard approach used for communication between the controller and the OF switches on the southbound interface of the SDN [102]. The southbound interface carries requests and replies to both the controller and the OF switches. The updated network topology information is significant to the controller in providing efficient control and management of the network. As a result, the efficient topology discovery is considered to be an important characteristic for the controller. Developing a topology of the network requires switch discovery, host discovery, and interconnected switches' discovery [52]. Each of these discovery mechanisms is briefly explained in Section 4.1.

In the work [4], an efficient topology discovery mechanism is proposed which reduces the topology discovery overhead up to 40 % by minimizing *Packet_Out* messages generated from the controller. A single LLDP packet is sent to each of the OF switches rather than the de-facto standard of sending each LLDP packet to each of the ports of the OF switch. Moreover, a switch broadcasts the LLDP packet to all its active ports which further discovers links between the switches. The work in [18] proposes to represent network topology, find loops, and determine alternative paths at the time of link failure in SDN. An adjacency matrix is used to represent the LLDP packets corresponding to the switches in the network. This helps to find the loops and alternative paths at the time of link failure in SDN. Moreover, the work in [19] presents the security of topology discovery in SDN and shows that how information can be spoofed to generate fake links in the network topology. Finally, it also presents a countermeasure by using the Keyed-Hash Message Authentication Code (HMAC) authentication.

Table 2: Comparison between a traditional network and an SDN topology discovery

Features	Topology Discovery in Traditional Networks	Topology Discovery in Software Defined Networks
Host Discovery	NMAP	*Packet_In* message
Switch Discovery	SNMP	Initial Handshaking process
Link Discovery	Various updates (RIP, OSPF, LSA, OLSR)	LLDP
Control Management	Independent	Controller
Scalability	No. of switches	No. of OF switches
Communication updates	Switch- Switch	Controller-Switch-Controller

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials

4. A Thematic Taxonomy: Topology Discovery

In this section, we provide an in-depth information about the topology discovery of the SDN. We devise a thematic taxonomy of the topology discovery in SDN as illustrated in Figure 4. The thematic taxonomy can be used to establish a conceptual knowledge of the topology discovery [76]. The taxonomy consists of four main categories including (1) Discovery Entities, (2) Controller Platform, (3) Topology-Dependent Services, and (4) Objective. These categories provide a clear understanding of the topology discovery in SDN.

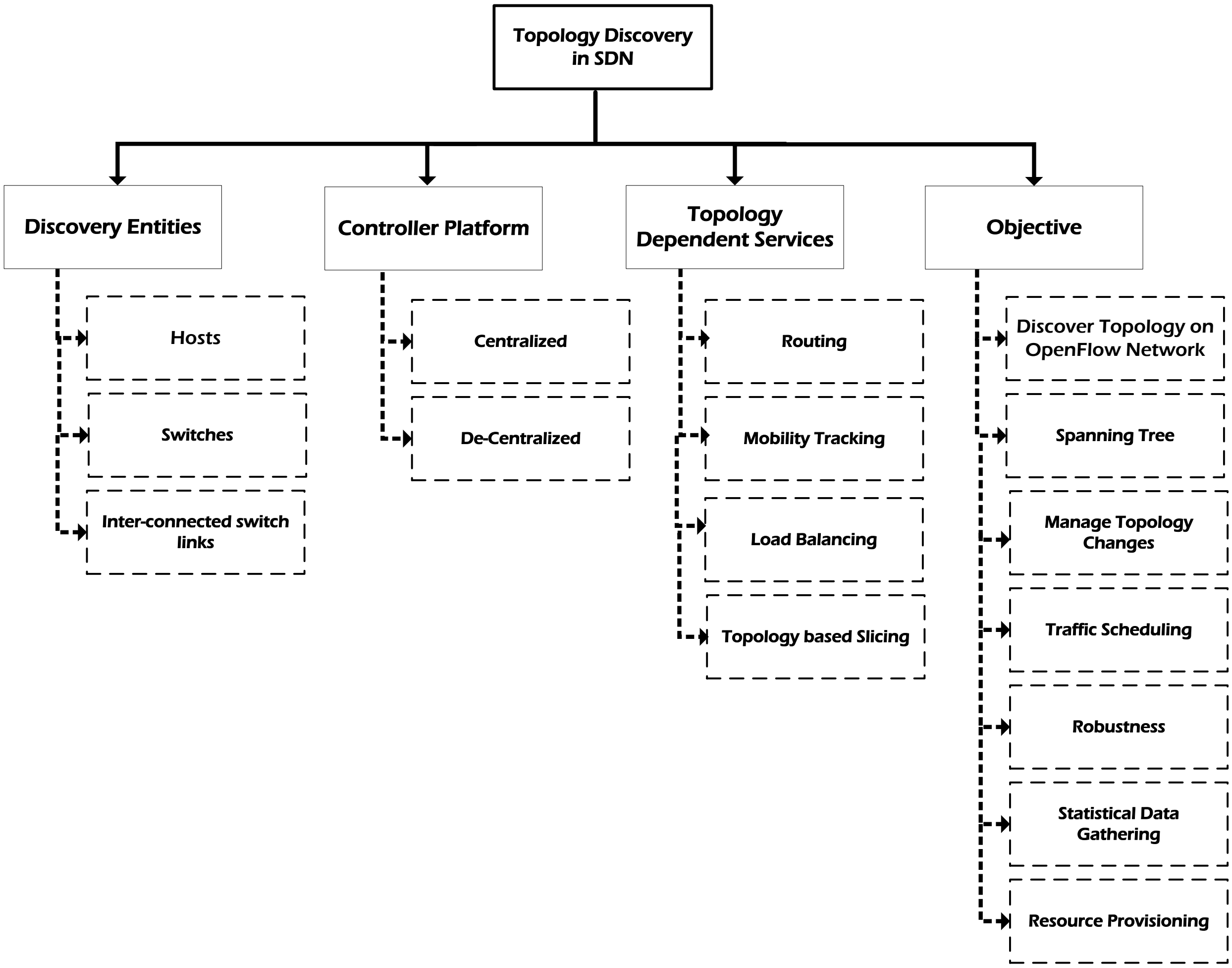

Figure 4: A thematic taxonomy of topology discovery in SDN

4.1 Discovery Entities

The controller has a visibility of the entire network topology. To create a topology of the entire network, the controller has to discover network entities and inter-connected links among them. In particular, the controller has to discover three entities for a complete view of the network topology, i.e., a) Hosts, b) Switches, and c) Inter-connected links between the switches. The hosts are the physical or virtual systems (virtual machines) connected to the switches that are used by users to execute their services. The switches are known as the OpenFlow switches that forward the packets from source to the destination upon receiving flow rules from the controller. The inter-connected links are the physical or the virtual links between the switches which are used to transfer the network packets. Discovery of these entities is significant for the topology management in updating a network topology view of the controller.

4.1.1 Host Discovery

. The host discovery helps the controller in identifying the exact location of the host in the network which allows for traffic monitoring, assisting in traffic routes, and determining the source of the packets [103]. Generally, a host tracking function is available in most of the controllers, which determines the host attached and its respective port of the switch [104]. The host tracking can trace the virtual machine migrations in the data centers, which are difficult if done manually due to their frequent moments. The controller maintains a host profile table for each of the hosts that joins the network.

Similarly, the controller deletes the host profile table when a host leaves the network. To populate the empty host table, the controller uses the *Packet_In* message to generate a host profile table for each of the hosts sent by the OF switch. For example, a host attached to a port of the OF switch generates a request message. This request message is encapsulated by the OF switch in the form of *Packet_In* message and it is then sent to the controller. Based on the *Packet-In* message, a controller identifies the identity of the host.

The host profile is built on the *Packet-In* message which contains information such as a) IP address, b) MAC address and c) Meta information (DPID, port number, and last timestamp). When a host migrates from one switch to another, its port and switch IDs are changed due to its new location. The controller updates the record for the migrated host based on *Packet-In* messages received from another OF switch. The payload information in the *Packet-In* message helps the controller to track the location of the host. Different controllers have different host tracking applications to discover hosts in the network [52]. For instance, the *'hosttracker.cc'* is used in NOX controller, the *'host tracker.py'* is used in Ryu controller, the *'DeviceManagerImpl.java'* is used in the Floodlight controller, and the *'OFMDeviceManager.java'* is used in the OpenIRIS controller.

4.1.2 Switch Discovery

Typically, OF switches communicate with the controller on the arrival of new packets, i.e., *Packet_In* messages. The controller replies with a *Packet_Out* message to insert flow entries in the OF switches. The location of the OF switches is vital to the controller due to its two-way frequent communication. The controller discovers the location of the OF switches in its initial handshake process.

The OF switches are discovered in the initial handshaking process by the controller. Once the OF switch is added to the network, the controller gets the existence and key properties of the OF switch. The controller records the MAC address, the number of ports, etc. to know the information about the OF switch. There is no requirement for a separate protocol to discover the location of the OF switch in SDN.

4.1.3 Inter-connected link between switches

The discovery of the inter-connected link between switches is significant to generate a topology by the controller in SDN. The inter-connected links determine the connectivity between the OF switches that helps the controller and the application plane services to utilize the network according to their requirements. In most of the times (if not all), the OFDP is used to discover the inter-connected links between the OF switches. The OFDP uses LLDP to advertise the capabilities and neighbor information of the nodes in the network [105]. The LLDP is usually used in the Ethernet switch, which actively sends and receives LLDP packets to each of its active ports. The extracted information from the LLDP packet is stored in a Management Information Base (MIB) in the switch.

The collected information from different MIB's of the switches via the SNMP helps to determine the network topology. When the LLDP packet is sent by the switch through its active ports, the Ethernet frame encapsulates the payload of LLDP and set the EtherType field to 0x88cc. The Ethernet frame contains the LLDP Data Unit (LLDPDU) that consists of a Type Length Value (TLV) structure. The TLV contains a switch identifier (chassis ID), Port ID, Time to live value, and other optional values. The OFDP uses a similar format of LLDP packet, however; it operates differently due to its limited API's match-action functionality. Moreover, in the SDN the OF switches does not send, receive, and process the LLDP messages itself but rather created by the controller. The operation of the LLDP packet in the SDN is briefly explained below.

(a) Inter-connection between OF switches: The link discovery using LLDP does not require any other discovery approach because both ends of the link consist the OF switches which support the topology discovery mechanism. The topology discovery determines the initial IP address and the TCP port of the controller which helps the OF switch to establish a connection soon after it is connected to the controller. The

OF switch also has a pre-configured rules, which generates the *Packet_In* message to the controller when it is received on the ports other than the controller. Initially, when an OF switch establishes a connection with a controller, the controller passes a request message to the OF switch such as FEATURE_REQUEST_MESSAGE, wherein the switch responds with a FEATURE_REPLY_MESSAGE. The response includes the switch ID and active ports along with their respective MAC addresses. The controller encapsulates the LLDP packet in a *Packet_Out* message and sends it to each active port of all OF switches in the network. The destination address in the LLDP packet is the multicast MAC address defined in the IEEE 802.1AB standard. The total number of *Packet_Out* messages sent by the controller is equal to the number of active ports in the network, i.e., (Total *Packet_Out* message = Number of active ports of all the switches).

The *Packet_Out* message installs the flow entries in the OF switch in order to route each LLDP packet to its destination port as indicated in the TLV field. The OF switch forwards the received LLDP packet to its corresponding port that is connected to another OF switch. When the neighbor of the OF switch receives the LLDP packet on the port other than the connected controller port, the switch encapsulates the LLDP packet in a *Packet_In* message and forwards it to the controller. The fields in the *Packet_In* message includes the switch ID and the Port ID on which the LLDP packet is received. The controller updates its network topology based on LLDP messages and by default, this process is repeated every 5 seconds. The illustration of this process is shown in Figure 5.

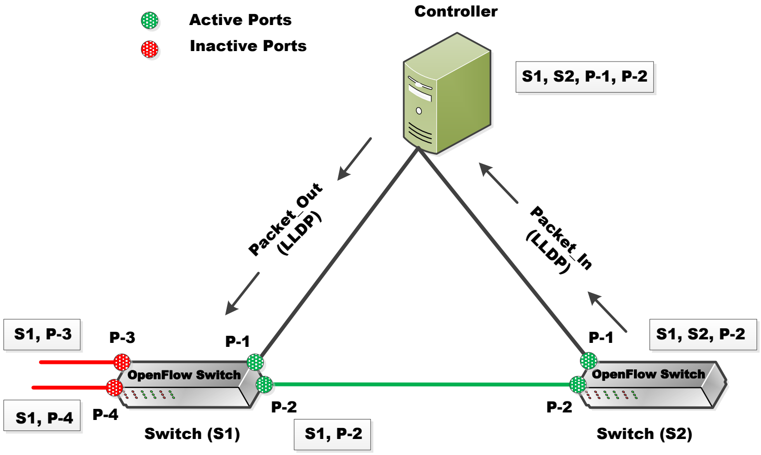

Figure 5: The LLDP process in SDN environment

(b) Inter-connection between the OF switch and the traditional switch: Currently, the adoption of SDN architecture in the current emerging networks integrates OF protocol with the existing traditional network technologies. This requires a new

mechanism to operate in a new network infrastructure without affecting the performance. Similarly, using both traditional and OF switches in data centers will create problems to identify the inter-connected link between these switches. The approach to finding the inter-connected link between the OF switches is not implemented in a hybrid switch infrastructure. The controller needs a mechanism for the handshake with an OF switch to identify its information and capabilities. However, handshake is not performed for the traditional switches. A controller in a hybrid switch infrastructure can identify the inter-connected links between the OF switch and a traditional switch which must be connected to another OF switch. This scenario can be considered as a non-direct connection between two OF switches. Simply, a controller can find the multi-hop connections between the OF switches. The LLDP is a single-hop discovery mechanism and it is not applicable to a multi-hop connection. It requires a new approach for finding non-directed connections among the OF switches. To identify the inter-connection between two OF switches, both the OF switches should be in the same broadcast domain or the controller will not able to associate addresses to the multi-hops among the OF switches. The current Open source controller such as Floodlight and Open Daylight controller have integrated layer 2 topology discovery protocols such as the LLDP and the Broadcast Domain Discovery Protocol (BDDP) to discover multi-hop links between OF switches and traditional switches within a broadcast domain [106].

The BDDP message and the LLDP messages are identical but have different destination MAC address fields. The BDDP message has a broadcast address in its destination field while the LLDP message has a multicast address in its destination field. This feature allows the traditional switch to forward a BDDP message to find multi-hop links between the OF switches within a broadcast domain. The controller sends each BDDP message to each active port of the switch by encapsulating it in the *Packet_Out* message. When the *Packet_Out* message is sent to the OF switch, it installs a flow entry in the flow table indicating that the OF switch has received the message. Then, the OF switch forwards the message to the neighbor switches via a port indicated in the TLV field. If the neighbor switch is a traditional, it examines the destination MAC address and further floods the packet to all its active ports. The port connecting the controller receives the *Packet_In* message that incorporates the Meta data required to identify multi-hop links. The *Packet_In* message contains a BDDP packet which helps the controller to know indirect links between two OF switches such as through multi-hop links.

4.2 Controller Platform

The SDN has an architecture which consists of a single or multiple controllers to control the entire network as illustrated in Figure 6(a) and 6(b). Usually, small data center networks incorporates a single controller while large data centers are distributed and have multiple controllers. This section explains the significance of a topology discovery in single and multiple controller platforms in SDN.

4.2.1 Topology Discovery in Single Controller Platforms

The single controller platform is used for a homogeneous network, which is a network of devices connected within a single physical location. The controller is responsible for discovering the network topology by querying the switches through the LLDP packets as described in Section 4.3.3. The controller communicates with the switches through LLDP packet after a specified time interval (i.e., after every 5 seconds) to identify the links between the OF switches in the network. In discovering the network topology, the position of the controller is crucial. That is, the controller that is closer to the switches will result in a faster transmission of the LLDP packets to the OF switches as well as receiving a quick response from the OF switches.

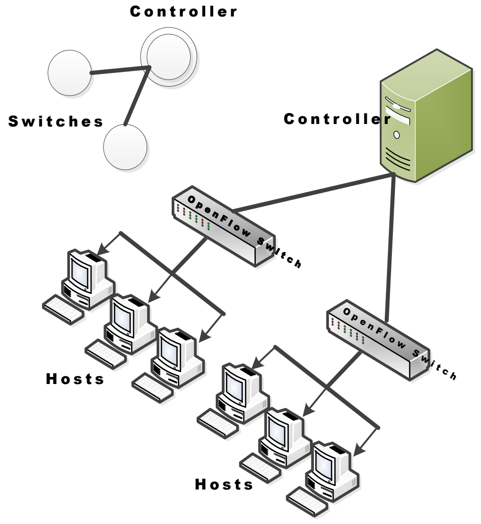

Figure 6(a): Single controller architecture

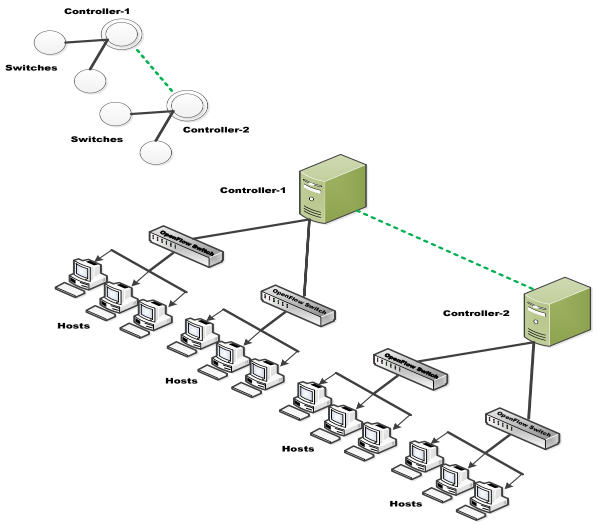

Figure 6(b): Multiple controller architectures

4.2.2 Topology Discovery in Multiple Controller Platforms

Large networks are employed using heterogeneous setting that includes multiple controllers responsible for different portions of the network. All these controllers coordinate through a logically centralized controller. Each controller requires discovery of the network topology of the assigned SDN domain. The topology discovery information is sent to the centralized controller and also to the neighboring controllers for the latest updates. However, sharing topology information among controllers requires a standard procedure which is not available till date.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials

The shared topology information is not verified during the sharing process and might be shared by the malicious controller. This may affect the performance of the other controllers such as routing, load balancing, scheduling and various other services. The importance of topology discovery in multiple controller platforms increases due to the distributed controllers, sharing of topology updates, and instance (virtual machine) migration [101] among different SDN domains.

4.3 Dependent Services

This section discusses the topology-dependent services used in SDN. The logically centralized visibility of the network supports various network applications to efficiently perform tasks and control the network devices. We have explained some of the topology dependent services to highlight the significance of the topology discovery in SDN as shown in Table 3.

4.3.1 Routing

The routing application depends on the controller's abstract view of the topology to provide the visibility of the entire network [107]. For instance, a routing application will require information about the network topology to route the network traffic to its destination on the shortest path [17]. However, falsified topological information may lead the routing application to route its network traffic on to a malicious route.

In the case of a link fabrication attack, an attacker can spoof the LLDP packet with a malicious switch DPID and Port ID to inject a fake link in the network topology. This may affect the existing legitimate shortest path towards the destination. For example, as shown in Figure 7, host 1 requires four hops (switches) to reach host 4. However, during a link fabrication attack, host 1 will send the LLDP packet with DPID-3 and Port ID-1 to switch 1, which further inserts DPID-1 and ingress Port ID-1 in the metadata of the *Packet_In* message and informs the controller that there is a direct link between switch-1 and switch-3. Subsequently, the controller may wrongly update its topology information by assuming a direct link between switch 1 and switch 3. This affects the legitimate shortest path, as the traffic from host 1 can be sent to host 4 through the newly added fake link. As a result, the malicious switch 3 can eavesdrop or modify the traffic before it reaches the destination.

4.3.2 Mobility Tracking

The mobility tracking refers to a mechanism of tracking a mobile node in the network. Mobility tracking is generally associated with the cellular networks [4, 108]. The mobility tracking in SDN is achieved through a mobility management function running on top of the controller [109]. The mobile management function is responsible for monitoring nodes' movements. Mobility tracking depends on the network topology for information on the current and future location of the network nodes. Usually, when a network node (host) changes its position from one switch to another, it changes its IP address and results in a connection break down. However, in SDN, with the help of a mobility management function, the forwarding function informs the controller about the nodes' movement which then re-calculates the forwarding rules and forwards it to the forwarding function to route the IP packets accordingly.

As a result, it continues with the application session and makes the movement of the node without changing the IP address. Node mobility changes the network topology which is updated by the controller based on the information received from the forwarding function. Thus, the node movement should be sent to the controller and mobility tracking function on a timely basis to keep the network topology up to date in the SDN.

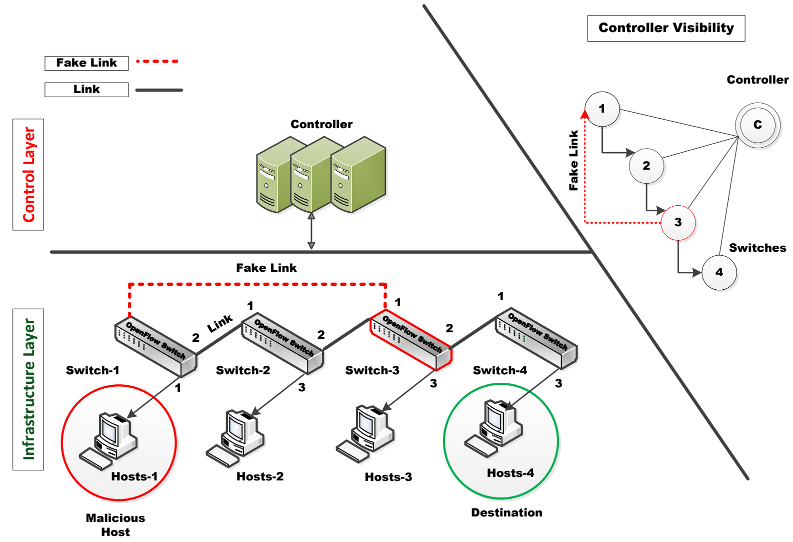

Figure 7: A diagram illustrating link fabrication attacks

4.3.3 Load Balancing

The load balancing is used to improve the utilization of resources and power by distributing the traffic more simply and more efficiently. The load balancer uses a logically centralized control of the SDN to perform traffic load balancing [110]. The dependency of the load balancing on the network topology is significant, which is its selection of the optimal server for the traffic execution. For instance, the load balancing application that is installed on top of the controller requires the location of the servers and the optimal path to access them in the network.

The optimal path is selected by computing augmented bandwidth of the links between the switches. Any modification in the network topology causes the load balancer to re-calculate the bandwidth for the optimal path. Thus, topology discovery plays a vital role in executing the load balancing properly in SDN.

4.3.4 Topology-based Slicing

The topology-based slicing is a mechanism of the Flow-visor in SDN that divides the network topology into different parts/slices [111]. The aim of slicing is to provide a dedicated link to each of the tenants in the multi-tenant environment. Topology-based slicing, also known as port-based slicing, creates different slices based on the switch ports. Each switch part has a subset of full network topology controlled by the Flow-visor [112].

The Flow-visor handles the network traffic on each of the connected links by adding a flow space. The slicing phenomenon reduces the controller load by focusing on specific OF switches of the topology. Therefore, the slicing depends on the locations and ports of the OF switches which are key entities of the network topology discovery. However, any modification in the topology will cause Flow-visor to re-compute the specific slice that is affected by the change in the specific domain.

Table 3: Topology-dependent applications with its effected attacks

Topology-Dependent Applications	Description	Effected attacks
Routing	Route network traffic from source to the destination	Link Fabrication
Mobility Tracking	Determine the location of the host in the network	Host Location Hijacking
Load Balancing	Distribute network traffic among different servers	Link Fabrication
Topology-based Slicing	Divide single network topology into sub-topologies	Link Fabrication

4.4 Objective

The key objectives which are achieved through an efficient topology discovery in SDN are listed as follows:

(a) *Multiple switches:* The topology discovery provides an easy way to identify OF switches in SDN. The OF switches could be in single or multiple management domains, controlled by single or multiple distributed controllers [113]. The identification of the OF switches in the network assists in topology discovery and updating its topology information, respectively.

(b) *Spanning Tree:* The spanning tree protocol in SDN [114] provides a loop-free topology. It utilizes discovery services to identify a neighbor link detection between OF switches. The spanning tree installs flow entries in the OF switches. However, without having an efficient topology discovery, the spanning tree is unable to select an appropriate path to eliminate a loop from the network.

(c) *Managing Topology changes:* The host migration, isolation of working domains, and insertion/deletion of the network devices in SDN can cause changes in the network topology. The function of topology discovery includes coping up with the change detected in the network. For instance, in a data center environment, virtual machines (hosts) often migrate from one resource to another, which results in a change in topology such as appearance of new switches and ports ID [115]. Consequently, the changes in the network should be sent to the controller to update the topology based on its discovery mechanism.

(d) *Traffic Scheduling:* Often, the optimal path, i.e., the shortest path is selected to route the network traffic from source to the destination. The topology discovery assists the traffic scheduler in finding the optimal path with less propagation delays between different number of hops (switches) [40, 116]. However, incomplete information regarding the network topology may lead to improper traffic scheduling that causes high bandwidth delays and time overhead.

(e) *Robustness:* The ability to tolerate the packet loss depends on the topology. If the topology is timely updated by the controller, the network application runs smoothly without causing any packet loss. The correct topology of the network reduces the overhead of the controller that cause to increase robustness of the SDN without affecting its normal operation.

(f) *Statistical Data Gathering*: The OF switches provide different levels of statistical information to the controller including port statistics, flow statistics, and other counter measurements [117]. The statistical information helps the controller to have an in-depth observation about the flows, network devices and the overall behavior of the network. However, changes in the network topology (due to the insertion of new hosts, flow entries, and inter-connected links) between the OF switches can cause changes in the statistical data previously gathered by the controller. The controller has to update its database information based on the new topology of the network.

(g) *Resource Provisioning:* To operate an elastic data center infrastructure through the SDN architecture, a proper resource provisioning mechanism is required to enable on-demand resources for the applications [118]. The resource provision depends on the network topology in order to understand the allocation and processing of available resources to different applications. The topology discovery information assists the resource provisioning module in selecting the right resources for the right application.

5. Topology Discovery Threats and Solutions

In this section, we provide a comprehensive description of the potential threats to the topological discovery. The threats exploit the vulnerabilities in the controller by performing attacks on the network. Basically, we devise a classification of the topology discovery threats as illustrated in Figure 8. The classification comprises of three categories, such as a) Attack entity, b) Controller vulnerabilities, c) Current solutions, and d) Miscellaneous threats. Each of the categories is explained as follows.

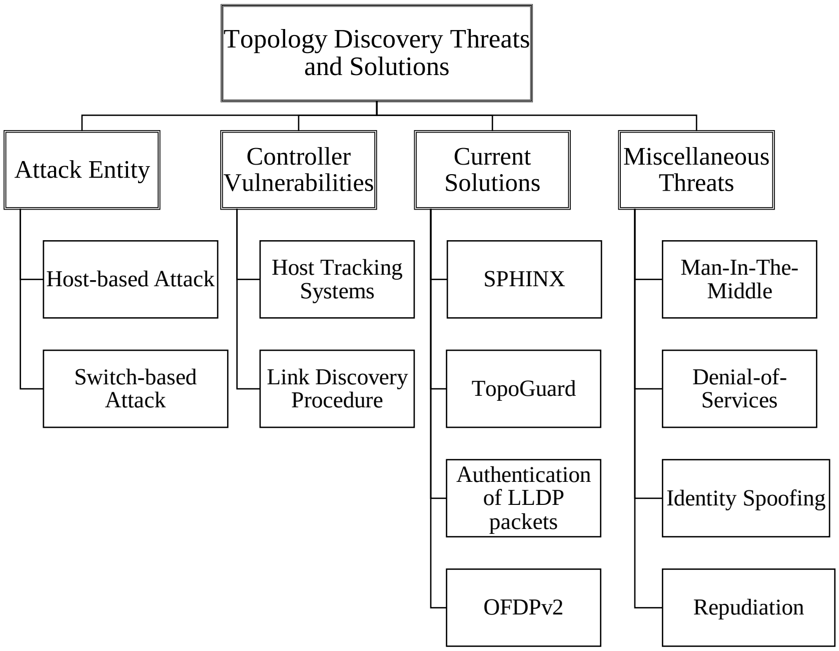

Figure 8: Classification of topology discovery threats and solutions

5.1 Attack Entity

Several security threats from different parts of the SDN architecture can be recognized through literature. In this section, we focus on the topology poisoning attacks. The topology poisoning attack is generated with respect to two entities in the SDN architecture i.e., hosts and the OF switches. These attacks are explained with respect to their working operations as follows.

5.1.1 Host-based Attacks

The topology poisoning attack generated from the host (system connected to the switch) is called host location hijacking attack. In this attack, the attacker impersonates the target host location in the network and starts receiving traffic intended for the target host. The attacker exploits Host Tracking System (HTS) of the controller which lacks an authentication mechanism especially for the host mobility in SDN. The controller uses HTS to record parameters such as joining and mobility of the host in SDN by maintaining a host profile table. The controller uses *Packet-In* message to update the host profile table by monitoring the DPID, ingress Port ID, and other metadata information. However, lack of security consideration in HTS provides an opportunity for the attacker to temper the target host location information by diverting the target host traffic towards itself.

The controller may assume that the target host has moved towards the new location but actually towards the attacker location. The attacker easily hijacks the traffic of the target host by generating a spoofed IP address of the target host using the *Packet_In* message. Upon receiving the *Packet-In* message, the controller updates the host profile table of the target host using its new location, as a result, affecting the topology-dependent applications including routing, load balancing, and various others.

Moreover, the malicious host in SDN can spoof the legitimate LLDP packets and forward it to the OF switch as well. That is, the OF switch can forward the spoofed LLDP packet to all of its active ports, which may reach the controller and can update its link record of the OF switches. The malicious host can also send a legitimate LLDP packet to another OF switch, which may create a fake link between the OF switches. Therefore, the controller may route the traffic on the fake links that actually gets forwarded to the malicious host.

5.1.2 Switch-based Attacks

The topology poisoning attack can also be performed through the malicious OF switches. The malicious OF switches spoofs the LLDP packets by creating fake links in the network. This type of attack is called a link fabrication attack in SDN. The malicious OF switches can affect a large scale of the network due to fake connections with many devices. The topology poisoned through malicious OF switches is difficult to detect due to the minimal clues in reference to the fake link creation in the network. For instance, it does not require any host to create a fake link between the OF switches during a topological poisoning attack.

After receiving the LLDP packet from a single OF switch, the malicious OF switch relays the packets to another OF switch instead of forwarding them to the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials

controller. Subsequently, upon reception of LLDP packets by the new OF switch, the LLDP packets are sent to the controller in the form of *Packet-In* message. This tricks the controller into believing that there exists a link between the malicious OF switch and the legitimate OF switch that forwarded the LLDP packet to the controller. Such fake link injection attracts future possible attacks such as the DoS attack, man-in-the-middle attack and much more.

5.2 Controller Vulnerabilities

This section describes the vulnerabilities in the controller that are used by the attacker to launch an attack.

5.2.1 Host Tracking Systems

The vulnerability discovered in the HTS attracts attackers to hijack the location of the hosts. As stated earlier that the host profile in the controller contains the DPID, ingress Port ID, and other metadata information which exhibits the controller with the location of the host and the connected OF switch. The key exploited vulnerability includes the lack of authentication mechanism that can be used to verify the host updates received by the controller through *Packet-In* message. All information received by the controller is considered as genuine (even if received from a malicious OF switch) and the host profile is updated accordingly.

In previous versions of Floodlight and Open Daylight controllers, an empty shell API *'isEntityAllowed'* is provided, which accepts all updates related to the host locations. The attacker simply spoofs the packet with target host identity and forwards it to the connected OF switch which further send it to the controller in the form of *Packet_In* message. The controller assumes a shift of position of the host and updates the host profile for the target host. The lack of authentication mechanism in HTS makes the controller update the topology with falsified host information and this will affect numerous services, especially routing.

5.2.2 Link Discovery Procedure

The vulnerability in the link discovery procedure can also be exploited by fabricating the false link between the OF switches. Firstly, there is no authentication mechanism for the controller to ensure the origin of the LLDP packet. Secondly, the controller is unable to verify the traversed path used by the LLDP. Addressing these issues is critical in preventing the OF switches from inserting a fake link. Note that, the OF switches receive LLDP packets from each of its ports. This allows the attacker to spoof the LLDP packets to create a fake inter-connected link between the OF switches which is known as link fabrication attack.

This attack can be performed in two ways 1) modification of LLDP packets, and 2) through the LLDP relay. In the case of the LLDP modification,, a fake link is established between the OF switches by spoofing the DPID and the Port ID of legitimate OF switch. This causes the controller to update a new link between legitimate and malicious OF switch. The link fabrication attack generated through the modified LLDP packet is explained with an example shown in Figure 7.

The host attached to switch-1 learns about the LLDP syntax from the receiving LLDP packet from the controller. The switch-1 then forwards it to all of its ports except the controller port. The host-1 sends LLDP packet to the switch-1 with spoofed information including DPID=3 of switch-3 and Port ID=1 of switch 3. The switch-1 inserts the DPID=1, and Port ID=1 in the metadata and and forwards *the Packet-In* message to the controller. The controller checks the *Packet_In* message and perceives a link between switch 3 and switch 1. The controller takes the LLDP source information from the TLV such as (DPID=3, Port ID=1) and the link information from the metadata such as (DPID=1, Port ID=1). Thus, the controller is updated with the wrong related to a fake link. In another type of link fabrication attack, the attacker simply forward one of the legitimate LLDP packets to another OF switch and resulting in a falsified link information received by the controller. The malicious OF switch requires a relay OF switch to forward the LLDP packet to the target OF switch.

The relay OF switch is identified through a connection test. In addition, some controllers such as POX and Floodlight disables the HTS on the internal link switch ports, however, an attacker can still launch the attack by using a tunnel-based LLDP relay attack. The tunnel-based LLDP relay attack is used to launch fake links between multi-hop link ports having OF switches connected to the traditional switches. It is difficult to disable these ports in SDN due to the availability of the hybrid switches in the network. Thus, the link fabrication attack also opens doors to numerous attacks including the DoS and the man-in-the-middle attack.

5.3 Current solutions

We explain the state-of-the-art topological poisoning solutions in this section. However, the literature has very few solutions the topology poisoning attacks. We briefly explain the state-of-the-art solution with reference to their proposed methodology. Table 4 presents the comparison between the proposed solutions using parameters such as techniques, SDN features, attack entity, the problem addressed, and future work. The parameter techniques highlights the key module/application developed by the proposed solution. The parameter SDN features points out which features have been used to model the solution in SDN. The attack entity shows which type of attacks can be detected through the current solution. The parameter problem addressed points out a objective functions which has been addressed to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials

detect the attack. Finally, the parameter future work explains future research directions of the current solution.

5.3.1 SPHINX

In [88], the work has presented several attacks which target the network topology and forwarding devices in SDN. It has been shown that an attack can be launched from malicious hosts and OF switches. A proposed solution as a SPHINX is presented to detect an unknown attack on network topology and the forwarding devices in SDN. The SPHINX provides a real-time and accurate verification solution of the network behavior by a) monitoring all OF messages, b) analyzing features set of the messages, and c) providing a fast validation of the network updates. The SPHINX focuses on four messages, i.e., *Packet_In, stats_reply, features_reply,* and *flow_mod* to get the metadata, detect network topology and forwarding device attacks. First, the SPHINX intercepts the OF messages transferred between the switch and the controller. Then, it builds the incremental flow graphs with new updates and validates the network behavior. These intercepts are important to identify the malicious behavior of the attacker.

After getting the latest update, SPHINX increments and updates its network topology flow graphs and detects malicious behavior based on the tangible changes observed in the network topology and the data plane forwarding. Specifically, IP/MAC address binding, MAC/port binding, and flow statistics of the host are used to provide metadata for assisting SPHINX to detect malicious behaviors in the network topology and the data plane forwarding. The network behavior is validated through the SPHINX policy engine. The policy engine enables administrators to validate the incremental flow graphs. The constraints specified by the administrators is written in the policy language. However, validating the policy itself is not considered in the SPHINX and is left for future work. In [24], a policy-based security is provided for an SDN.

5.3.2 TopoGuard

Hong *et al.* [52] has first time proposed an attack in the SDN architecture that affect the visibility of the controller by providing poisoned network topology view. The attack illegally modify the network visibility by hijacking the host location and inserting a fake link between the OF switches. These attacks disturb the operation of different network applications that run on top of the controller such as packet routing, network virtualization, and mobility tracking. A TopoGuard application is proposed to overcome the problem of the poisoned network topology in SDN.

The TopoGuard application is executed in the OF controller that is composed of three main modules namely, port manager, host prober, and topology update

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials

checker. Each of these modules in the TopoGuard application depends on the *Packet_In* message to investigate and detect the illegal modifications in the network topology. The port manager provides information related to the device type connected to the switch. The device type contains values of *host, switch,* or *any*. The value *any* is the default value for the device type and will change to *host* or *switch* once the packet has been forwarded. The port manager detects the attack and generates an alert to the topology update checker by receiving LLDP packets from the host. The alerts are generated when LLDP packets traverse between internal link ports of the switches rather than hosts.

Upon migration to a new location, the host probe module is responsible for checking whether the host's previous location is unreachable. This is achieved by sending the probe packets (i.e., ICMP echo packets) the host's previous location . If the echo replies are received by the host probe, it will inform the topology update checker to hold the update of a new host location due to a host location hijacking attack. Similarly, topology update checker is also responsible for checking and verifying the information of the host migration and new link discovery in the network topology. Once the host migration is detected, the topology update checker collects the host's previous location from the host probe and then updates its topology discovery of the network.

For the link discovery, topology update checker checks cryptographic hash value for the integrity of the LLDP packet. After the integrity check, the device type is checked from which the LLDP packet is generated. If the device found has a *host* entity, the topology update checker considers it as an attack and holds the update of the new discovery link in the network topology. Therefore, TopoGuard enables a real-time detection of the topology poisoning attacks in SDN.

5.3.3 Authentication of LLDP Packets

In [19], a countermeasure based mechanism is proposed to overcome the security problem presented in the OFDP. The OFDP lacks an authentication of LLDP packets that might risk the packets to be forged. The proposed method uses a cryptographic Message Authentication Code (MAC) in each of the LLDP packets in order to authenticate the packet's integrity. The HMAC is used to compute the MAC code. The uniqueness of the HMAC in authenticating the LLDP packet is the use of a dynamic key instead of the static key. In each round of the topology discovery, a dynamic key is used for each LLDP packet which makes difficulty for the adversaries to speculate the key. Guessing the key is critical in order to compute the MAC value and launch a successful fake link attack.

The key is selected randomly for security and it is difficult for the attacker to guess the key especially when the key is generated with an entropy measurement. Moreover, the controller can detect any attempt made by the attacker for guessing the key. The controller keeps track of each of the keys generated for every packet and verifies the authenticity of the received LLDP packet. The Chassis ID and the Port ID are combined to provide a necessary identifier while hashing is performed through a MD5 hashing function. The HMAC value is inserted in the optional TLV field of the LLDP packet which shows that the OFDP having HMAC can detect any fabrication of the LLDP packets generated by the attacker. The proposed method using HMAC values in the LLDP packet creates 8% of the CPU overhead which is lower than identifying fabricated links in the network.

5.3.4 OFDPv2

In [119], a simple and practical modification is performed on the existing topology discovery approach for reducing the control load and to increase the efficiency of the controller. The proposed approach modifies the de-facto standard of the topology discovery by introducing OFDPv2-A and OFDPv2-B. The two new versions have the same functionality of OFDP with significantly lesser number of control messages used for link identification between the OF switches. The reduction of control messages significantly decreases the controller load.

In OFDPv2-A, a specified rule is inserted in the flow table of every switch. This is to direct the OF switch to create a copy of the received LLDP packet and forward it to all of its active ports. . The forwarded message has a modified MAC address for each port. The LLDP packets are limited to the number of the available OF switches, however, the unique LLDP packet in OFDP is sent to the active ports of the switches that cause to increase the workload on the controller due to handling a large amount of the LLDP packets.

In addition, *Packet-In* event handler in the controller is changed to know the source MAC address of the Ethernet frame in the place of Port ID TLV field of the LLDP payload. The OFDPv2-B operates similar to OFDPv2-A but it has no rules to handle the LLDP packets from the controller. An action list is added to each of the *Packet_Out* messages to inform the OF switch about forwarding the packets. The action list contains the forwarding logic similar to the OFDPv2-A. The key advantage of the OFDPv2-B is the minimum use of the OF switch memory. However, OFDPv2-B has a disadvantage of increased size the of *Packet_Out* messages due to the insertion of an action list for each switch. The experiments results proven that OFDPv2-A and OFDPv2-B reduce 40% of the CPU overload and control traffic overhead as compared to the de-facto standard such as OFDP.

Table 4: A general description of state-of-the-art topology discovery

Classification	Techniques	SDN Features	Attack Entity	Problem Addressed	Future Work
SPHINX [88]	SPHINX controller application	Incremental flow graphs with metadata information	Host-based attack	To detect suspicious changes observe in network topology and data plane	Sphinx will consider flow rule aggregation, Proactive OpenFlow environment, and Mixed networks in the future.
TopoGuard [52]	TopoGuard application	Extension of OF controller by designing topology update checker	Host-based attack, Switch-based attack, Controller-based attack	Detection of network topology poisoning attacks	Design a new security framework which detects more vulnerabilities in SDN
Authentication of LLDP packets [19]	Hash Message Authentication Code	Using HMAC inside LLDP packet in every topology discovery round	Host-based attack	Provides authentication and packet integrity for LLDP packets	Check the impact of the proposed solution in another area rather than routing.
OFDPv2 [119]	OpenFlow discovery protocol	Modify de-facto standard of topology discovery, i.e. OFDP	Controller-based attack	Reduce the control messages used to identify the links between switches	The discovery of hosts in SDN network

5.4 Miscellaneous Threats

With a successful topology poisoning attack, the chance for other security threats also increase. The threats such as man-in-the-middle, denial of service, identity spoofing, and repudiation are explained in the subsequent sections and their side effects on the topology discovery are presented in Table 5.

5.4.1 Man-in-the-middle

The man-in-the-middle attack is performed in SDN in various ways [120]. One of which is to inject a fake link in the network topology. The attacker eavesdrop the traffic from source to destination by using a false update by the controller. The fake link may force the controller to divert traffic to the attacker. It might affect the confidentiality as well as the integrity of the network traffic passing through the fake link created between the OF switches.

To mitigate the man-in-the-middle attack in SDN a proposed solution [52] is presented. The proposed solution used device types such as (switch or host) to detect the spoofed LLDP packet, i.e., generated from the host rather than switches. Usually, no host participates in the legitimate LLDP propagation process. The LLDP packets traverse between the switches to determine a link between each other. The proposed solution determines whether the LLDP packet is generated from the host. In that case, the packet is considered as spoofed and further propagation of the packet in the network is stopped. The host devices are easily detected through the normal network traffic such as the TCP and the UDP. Once the device type is detected as a host, then any information regarding its topology update will not be considered as legitimate. Thus, the solution effectively prevents the man-in-the-middle attack at its early stage by finding the malicious host in SDN.

5.4.2 Denial of service

The controller uses the spanning tree algorithm to remove redundant ports after each topology update. However, an attacker can use the same feature to shut down the normal OF switch ports after injecting the fake links in the topology. This causes burden on the other links connected to the target OF switch and results in a DoS attack [24]. A legitimate link can be removed by sending a fake LLDP packet by the attacker to the OF switch having lower DPID. The attacker announces a link with a target switch as well. However, if the DPID of the selected OF switch by the attacker is lower than the target switch connected with another switch, then the port of the target switch connected to another switch is removed. This causes overhead on the selected OF switch which can cause a DoS attack due to an increase in the workload in the network traffic. The availability of the OF switch is affected through due to numerous flow rules in the flow table generated through as a result of a DoS attack.

The literature has various solutions to mitigate the DoS attacks in SDN. In [48], a connection migration tool is proposed to reduce data-control plane interaction that detects dynamic flow changes in the network traffic. In [70], a seamless primary controller backup is proposed to defend the centralized network operating system from failure. Another DoS prevention mechanism is proposed in [121], which uses proactive flow rules to preserve network policy enforcement. It uses packet migration mechanism to defend the controller from overloading its memory due to numerous *Packet_In* messages. However, the solution fails to find the real source of the attack.

5.4.3 Identity spoofing

When a malicious host injects spoofed LLDP packets, the controller updates the false information in the host profile system. The controller thinks that the host has changed its position and the information is sent to its new position from where the LLDP is received, i.e., a malicious host. The target host remains at its position however; the malicious host pretends to be a legitimate (target) host. The controller passes the information to the malicious host hence, affecting the confidentiality and integrity of the data by modifying it and forwarding it to further destinations.

To overwhelm the spoofed identity problem in SDN, a work in [52] proposes a solution based on the pre-condition and post-condition of the host migration. In the pre-condition, once the host migrates from its position, it has to inform SDN controller about its previous port_shutdown. In the post-condition, the controller confirms that host is not reachable by sending ping messages to its previous location. Thus, the controller can effectively track the real location of the host and can determine the spoofed identity of the malicious host.

5.4.4 Repudiation

The lack of a proper authentication mechanism of LLDP packet in the controller may cause repudiation attacks. The repudiation attacker creates a fake link through injected spoofed LLDP packet and then denies it to generate by him. The attacker inserts the spoofed DPID and Port ID of the victim OF switch and forwards the packet to the controller by showing it has been come from another OF switch. The spoofed LLDP packet loses its confidentiality and integrity by modifying its original value in the packet field.

The work presented in [88] builds an updated flow graph based on metadata of the *Packet_In* and FEATURES_REPLY messages to detect the fake links generated through spoofed LLDP packets. The MAC-IP binding mechanism of the proposed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials

solution is built by using the policy language engine that assists the controller to detect the fake links upon observing the deviation from the bindings.

Table 5: A side effect of threats on topology discovery

Threats	Possible Reason	Affect	Confidentiality	Integrity	Availability
Man-in-the-middle	Injected fake link	Change the shortest path	Yes	Yes	No
Denial of services	Removing legitimate link	Increase a workload	No	No	Yes
Identity spoofing	Spoofing host identity	Illegal information exploitation	Yes	Yes	No
Repudiation	Spoofed LLDP packet	Hiding the attacker identity	Yes	Yes	No

6 Future Challenges and Directions

In this section, future research challenges and directions of topology discovery in SDN are presented. The research on topology discovery is still at its early stages. Therefore, ample opportunities exist for future work to mitigate the challenges in topology discovery. The following future directions will help academicians, industrialists, SDN vendors, and network specialists to explore novel solutions in making the topology discovery secure and sustainable in the SDN. The descriptions with possible solutions for each future direction are given in Table 6.

6.1 Multiple SDN Domains

Practically, the SDNs are created by the network operators in the enterprise according to their network requirements. Mostly, the enterprise have different domains which are controlled by each controller resulting in a multiple SDN domains environments. However, small-scale data center network may require a single SDN domain while a large-scale data center network (carrier networks) may require several SDN domains that are controlled by the logically centralized controller. The division of SDN domains varies on the requirements that includes physical locations, traffic monitoring, load balancing, and various others.

However, interconnecting multiple SDN domains and sharing the network topology updates in the topology discovery can be a very challenging task. These interconnections require a standard protocol to efficiently share and secure the control information between the SDN devices. Moreover, the standard protocol must be able to consider various important aspects of the topology discovery including for instance a) how the network topologies in various SDN domains are connected, b) how one controller communicates with its neighbor controller, c) what will be the form of information format to share among the controllers, d) how to get the controller addresses, and e) which policies and procedures have to be adopted for the communication. The work presented in [23] proposed AutoSlice virtualized layer for the SDN architecture to separate multiple SDN domains on the shared network

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials

physical resources. This could enable efficient sharing of topology discovery information among the controllers placed in multiple SDN domains.

6.2 Topology Discovery through OF switches

Currently, the topology discovery function is executed by the controller and the SDN infrastructure is a single point of failure. The applications in these topologies can be affected by the malicious attacks on the controller. One way to overwhelm this issue is to shift the responsibility of a topology discovery to the OF switches. This reduces the liability on the controller and protects the topology discovery function at the time the controller is attacked. The OF switches can send the LLDP packets to their ports after a specified time interval to determine the links between their neighbor OF switches. The LLDP packet should contain the switch ID and an output port number to identify the origin of the LLDP packet. The OF switch should update the controller on every new link detected to inform the controller for the latest updates. Therefore, the topology discovery cost will be independent of the number of controllers in the SDN. However, discovering a network topology for each controller domain is costly in terms of the LLDP messages used in communication between controllers and the OF switches, network bandwidth consumption, and the time overhead.

The above complications can be minimized through the dependable and simpler topology discovery mechanisms in SDN. For instance, the use of the OF switch-based topology discovery can decrease the controller cost linearly because a single discovery mechanism will work for all the controllers in the network. Moreover, the OF switches can increase the priority values for the LLDP packets in the flow tables to transfer the packets on time even in the heavily loaded network links. This will assist the controller to have more consideration towards the other core functionalities.

6.3 Identification of fake links

The injection of a fake link in the network topology will critically damage the controller visibility and affect the network services to produce false results. To determine whether the link is fake or legitimate, the controller has to be intelligent enough to decide the legitimacy of the link in network topology within a specified time. However, currently, a proper mechanism is lacking essential features to distinguish between legitimate and fake links.

A potential solution to this issue is to access the history information generated by the OF switch to identify any involvement of a malicious activity. Another solution is to check the traffic flow on the newly inserted link as most of the fake links are created to overload the resources i.e., OF switches by flooding the link with packets. Also, selecting the optimal feature of the network traffic plays a vital part in the detection of a fake link. Therefore, utilizing a machine learning

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials

technique can make the management of the topology discovery more secure with minimum risks.

6.4 Frequent migration

Mostly, in medium and large data center networks, the SDN architecture is implemented for different purposes. The topology discovery is more sophisticated in these data center networks due to the frequent migration of the virtual instances in a virtualized network environment for numerous cloud users [71, 122]. This overloads the controller which requires to frequently update the network instances in order to have a clear and fair network visibility of the network. This opens the opportunity for the malicious node to connect with other network nodes and create fake links.

An intelligent mechanism based on statistical probability is required to track the network nodes behavior to assist in determining the malicious activities that affect the topology discovery mechanism. One way is to use the entropy measurement technique to determine the uncertainty in SDN after the attack [24]. For instance, the attacker injects the fake links in the network which creates uncertainty in the network due to incorrect network topology. The entropy can be used to determine the locations where the fake links are inserted by calculating the uncertainty in the network. It can support forensics mechanism in reaching the real source of the attack [74, 123]. As a result, the topology discovery will be performed efficiently by focusing on the visibility rather than the security parameters.

6.5 Topology discovery information safety

The internal state information of the controller is recorded in the Network Information Base (NIB). The NIB is a separate module in the controller that stores the critical states of the controller. These states can be used to regenerate the events at a specific time as required. Similarly, the topology discovery information is saved in the NIB module of the controller. Nowadays, the controller has become a key focus of attacks due to its core management functions and logically centralized control. The controller can be attacked through various channels to produce a false output. The decision of the malicious controller cannot be trusted and can lead to an incorrect decision. Similarly, during the attack on the controller, the NIB states can be affected, which might destroy the topology states stored in the NIB. As a result, the controller in the next iteration of the topology discovery updates its record without having the information from the previous topology discovery iteration. This may cause the controller to update with the malicious information injected by the attacker after exploiting the records of topology discovery state in the NIB.

To circumvent the aforementioned issue, a controller should forward a copy of its topology discovery states to its neighbor controller. The neighbor controller can re-generate the topology discovery states whenever the topology discovery states are affected by an attack. Alternatively, having a strong authentication mechanism will prevent the attackers from exploiting the core management modules of the controller. The work presented in [54] proposed an extendable control plane, i.e., DISCO in order to deliver end-to-end network services using a distributed controller environment. It enables highly manageable control channels for sharing aggregated network information among the controllers. Thus, it can traverse topology discovery information among the controllers in controlled environment.

6.6 Controller upgrade

The controller must be periodically upgraded by adding features, fixing bugs to improve its performance. This is important due to frequent change in the network infrastructure in the dynamic virtualized environment. Currently, the SDN lacks the effective techniques to assist the controller in upgrading without affecting the current operation of the network. In existing controller up gradation techniques, the controller is restarted or the old states of the controller are recorded and then replayed in the upgraded controller to recover its previous states. Similarly, the situation is same for the topology discovery states. Upon upgrading the controller for its new assignments, the previous topology discovery states are lost. This incurs overhead of re-executing the topology discovery right from the start to acquire the network visibility of the network.

One of the possible solutions is to save the topology discovery state in the neighbor controller [15]. After the controller is upgraded, the operation of the topology discovery is resumed from the last recorded status. However, when the network topology changes during the upgrades, the records will be inadequate in the respective topology. To minimize this issue, the controller should be upgraded at the time when the chance for the topological change in the network is minimum.

Table 6: A description of future challenges and directions of topology discovery with its possible solutions

Future Directions	**Description**	**Possible Solutions**
Multiple SDN domain	The SDN controllers controlling different domains create complication in sharing topology discovery information	— Standard protocol
Topology discovery through OF switches	Reduce less burden on the controller due to topology discovery	— Use OF switches for Topology discovery
Identification of fake links	To know about the status of the link	— Check OF switch history record — Verify traffic flow on the link
Frequent migration	Topology discovery mechanism becomes sophisticated due to frequent migration of instances	— Statistical probability — Entropy measurement
Safety of topology discovery information	The topology discovery states can be exploited by the attackers	— Strong authentication mechanism — Redundant topology discovery states
Upgradation of the controller	The topology discovery should be consistent at the time of the controller up gradation	— Redundant topology discovery states

7 Conclusion

The network visibility at the logically centralized controller is a unique characteristic of SDN. Despite the network visibility of the controller, network topologies are poisoned by attackers through the exploitation of vulnerabilities found in the controller functions. This can happen due to lack of security measures in the design of SDN. A substantial work is in development to build a secure and sustainable method to discover the network topologies by the SDN controller. However, research on the area of topology discovery security is still in its early stages. Efficient secure topology discovery mechanisms remain a distant goal for SDN in the future.

To meet the network visibility requirements, this work presented a comprehensive outline of the topology discovery and its implications towards a secure SDN. We explained the SDN layered architecture by discussing the security threats in each of the planes. In addition, we devised a thematic taxonomy of the topology discovery by reporting the discovery entities, controller platforms, topology-dependent services, and objectives. Comprehensive information is provided related to topology discovery threats by classifying them into four main categories including attack entities, current solutions, and miscellaneous threats.

Distinguished features of the current solutions have been explained along with their working mechanisms. Attack entities used to perform topological attacks are highlighted and discussed. Moreover, vulnerabilities found in the controller functions that can be exploited in poisoning the network are explained. Various types of topology poisoning attacks are presented which we believe may open ventures for further research in this field.

Acknowledgement

This work is fully funded by Bright Spark Unit, University of Malaya, Malaysia and partially funded by Malaysian Ministry of Higher Education under the University of Malaya High Impact Research Grant UM.C/625/1/HIR/MOE/FCSIT/03 and RP012C-13AFR. The authors also extend their sincere appreciations to the Deanship of Scientific Research at King Saud University for funding this Prolific Research Group (PRG-1436-16).

References

[1] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, *et al.*, "NOX: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review,* vol. 38, pp. 105-110, 2008.

[2] L. Vanbever, J. Reich, T. Benson, N. Foster, and J. Rexford, "HotSwap: correct and efficient controller upgrades for software-defined networks," in *Proceedings of the*

second ACM SIGCOMM workshop on Hot topics in software defined networking, 2013, pp. 133-138.

[3] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *Communications magazine, IEEE,* vol. 51, pp. 136-141, 2013.

[4] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, "Efficient topology discovery in OpenFlow-based Software Defined Networks," *Computer Communications,* 2015.

[5] M. Banikazemi, D. Olshefski, A. Shaikh, J. Tracey, and G. Wang, "Meridian: an SDN platform for cloud network services," *Communications Magazine, IEEE,* vol. 51, pp. 120-127, 2013.

[6] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing software defined networks," in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, 2013, pp. 1-13.

[7] S. Sezer, S. Scott-Hayward, P.-K. Chouhan, B. Fraser, D. Lake, J. Finnegan*, et al.*, "Are we ready for SDN? Implementation challenges for software-defined networks," *Communications Magazine, IEEE,* vol. 51, pp. 36-43, 2013.

[8] D. Kreutz, F. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 2013, pp. 55-60.

[9] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Recent Advances in Intrusion Detection*, 2011, pp. 161-180.

[10] S. Shin, P. A. Porras, V. Yegneswaran, M. W. Fong, G. Gu, and M. Tyson, "FRESCO: Modular Composable Security Services for Software-Defined Networks," in *NDSS*, 2013.

[11] S. Shin and G. Gu, "Attacking software-defined networks: A first feasibility study," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 2013, pp. 165-166.

[12] S. Lim, J.-I. Ha, H. Kim, Y. Kim, and S. Yang, "A SDN-oriented DDoS blocking scheme for botnet-based attacks," in *Ubiquitous and Future Networks (ICUFN), 2014 Sixth International Conf on*, 2014, pp. 63-68.

[13] Q. Yan and F. Yu, "Distributed denial of service attacks in software-defined networking with cloud computing," *Communications Magazine, IEEE,* vol. 53, pp. 52-59, 2015.

[14] A. Akhunzada, A. Gani, N. B. Anuar, A. Abdelaziz, M. K. Khan, A. Hayat*, et al.*, "Secure and dependable software defined networks," *Journal of Network and Computer Applications,* 2015.

[15] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on Network Virtualization Hypervisors for Software Defined Networking," *IEEE Communications Surveys & Tutorials,* vol. 18, pp. 655-685, 2016.

[16] Z. Shu, J. Wan, D. Li, J. Lin, A. V. Vasilakos, and M. Imran, "Security in Software-Defined Networking: Threats and Countermeasures," *Mobile Networks and Applications,* pp. 1-13.

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

[18] A. K. Saha, K. Sambyo, and C. Bhunia, "Topology Discovery, Loop Finding and Alternative Path Solution in POX Controller," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2016.

[19] T. Alharbi, M. Portmann, and F. Pakzad, "The (In) Security of Topology Discovery in Software Defined Networks," in *Local Computer Networks (LCN), 2015 IEEE 40th Conference on*, 2015, pp. 502-505.

[20] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A Survey on Software-Defined Networking," *IEEE Communications Surveys & Tutorials,* vol. 17, pp. 27-51, 2015.

[21] D. F. Macedo, D. Guedes, L. F. M. Vieira, M. A. M. Vieira, and M. Nogueira, "Programmable Networks&#x2014;From Software-Defined Radio to Software-Defined Networking," *IEEE Communications Surveys & Tutorials,* vol. 17, pp. 1102-1125, 2015.

[22] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in Software Defined Networks: A Survey," *IEEE Communications Surveys & Tutorials,* vol. 17, pp. 2317-2346, 2015.

[23] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A Survey of Security in Software Defined Networks," *IEEE Communications Surveys & Tutorials,* vol. 18, pp. 623-654, 2016.

[24] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges," *IEEE Communications Surveys & Tutorials,* vol. 18, pp. 602-622, 2016.

[25] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *Communications Surveys & Tutorials, IEEE,* vol. 16, pp. 1617-1634, 2014.

[26] J. W. Stewart III, *BGP4: inter-domain routing in the Internet*: Addison-Wesley Longman Publishing Co., Inc., 1998.

[27] J. T. Moy, *OSPF: anatomy of an Internet routing protocol*: Addison-Wesley Professional, 1998.

[28] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE,* vol. 103, pp. 14-76, 2015.

[29] K. Suleman, G. Abdullah, W. A. W. Ainuddin, A. Ahmed, and A. B. Mustapha, "FML: A novel Forensics Management Layer for Software Defined Networks," presented at the 6th International Conference on Cloud System and Big data Engineering, Confluence-2016, 14-15 Jan, 2016, Amity University, , Noida, UP India, 2016.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials

[30] B. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *Communications Surveys & Tutorials, IEEE,* vol. 16, pp. 1617-1634, 2014.

[31] S. Civanlar, E. Lokman, B. Kaytaz, and A. Murat Tekalp, "Distributed management of service-enabled flow-paths across multiple SDN domains," in *Networks and Communications (EuCNC), 2015 European Conference on*, 2015, pp. 360-364.

[32] L. Cui, F. R. Yu, and Q. Yan, "When big data meets software-defined networking: SDN for big data and big data for SDN," *Network, IEEE,* vol. 30, pp. 58-65, 2016.

[33] T. Zou, H. Xie, and H. Yin, "Supporting software defined networking with application layer traffic optimization," ed: Google Patents, 2013.

[34] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "NICE: Network intrusion detection and countermeasure selection in virtual network systems," *Dependable and Secure Computing, IEEE Transactions on,* vol. 10, pp. 198-211, 2013.

[35] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, "Payless: A low cost network monitoring framework for software defined networks," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, 2014, pp. 1-9.

[36] H. Hu, W. Han, G.-J. Ahn, and Z. Zhao, "FLOWGUARD: building robust firewalls for software-defined networks," in *Proceedings of the third workshop on Hot topics in software defined networking*, 2014, pp. 97-102.

[37] L. E. Li, Z. M. Mao, and J. Rexford, "Toward software-defined cellular networks," in *Software Defined Networking (EWSDN), 2012 European Workshop on*, 2012, pp. 7-12.

[38] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *Communications Surveys & Tutorials, IEEE,* vol. 16, pp. 1955-1980, 2014.

[39] M. Moshref, A. Bhargava, A. Gupta, M. Yu, and R. Govindan, "Flow-level state transition as a new switch primitive for SDN," in *Proceedings of the third workshop on Hot topics in software defined networking*, 2014, pp. 61-66.

[40] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Computer Networks,* vol. 71, pp. 1-30, 2014.

[41] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review,* vol. 44, pp. 87-98, 2014.

[42] W. Zhou, L. Li, M. Luo, and W. Chou, "REST API design patterns for SDN northbound API," in *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*, 2014, pp. 358-365.

[43] P. Sharma, S. Banerjee, S. Tandel, R. Aguiar, R. Amorim, and D. Pinheiro, "Enhancing network management frameworks with SDN-like control," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, 2013, pp. 688-691.

[44] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future internet," *Computer Networks,* vol. 75, pp. 453-471, 2014.

[45] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," *ACM SIGCOMM Computer Communication Review,* vol. 43, pp. 7-12, 2013.

[46] A. Kamisiński and C. Fung, "FlowMon: Detecting Malicious Switches in Software-Defined Networks," in *Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense*, 2015, pp. 39-45.

[47] M. Antikainen, T. Aura, and M. Särelä, "Spook in Your Network: Attacking an SDN with a Compromised OpenFlow Switch," in *Secure IT Systems*, ed: Springer, 2014, pp. 229-244.

[48] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 413-424.

[49] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On Controller Performance in Software-Defined Networks," *Hot-ICE,* vol. 12, pp. 1-6, 2012.

[50] D. Syrivelis, G. Parisis, D. Trossen, P. Flegkas, V. Sourlas, T. Korakis*, et al.*, "Pursuing a software defined information-centric network," in *Software Defined Networking (EWSDN), 2012 European Workshop on*, 2012, pp. 103-108.

[51] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takács, and P. Sköldström, "Scalable fault management for OpenFlow," in *Communications (ICC), 2012 IEEE International Conference on*, 2012, pp. 6606-6610.

[52] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures," in *NDSS*, 2015.

[53] F. Klaedtke, G. O. Karame, R. Bifulco, and H. Cui, "Towards an access control scheme for accessing flows in SDN," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, 2015, pp. 1-6.

[54] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed multi-domain sdn controllers," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, 2014, pp. 1-4.

[55] H. Li, P. Li, S. Guo, and A. Nayak, "Byzantine-resilient secure software-defined networks with multiple controllers in cloud," *Cloud Computing, IEEE Transactions on,* vol. 2, pp. 436-447, 2014.

[56] W. Braun and M. Menth, "Software-Defined Networking using OpenFlow: Protocols, applications and architectural design choices," *Future Internet,* vol. 6, pp. 302-336, 2014.

[57] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir, "Application-awareness in SDN," in *ACM SIGCOMM Computer Communication Review*, 2013, pp. 487-488.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials

[58] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, and P. Tran-Gia, "Sdn-based application-aware networking on the example of youtube video streaming," in *Software Defined Networks (EWSDN), 2013 Second European Workshop on*, 2013, pp. 87-92.

[59] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN security: A survey," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN For*, 2013, pp. 1-7.

[60] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software defined networks," *IEEE Network,* vol. 30, pp. 52-58, 2016.

[61] H. Saini, Y. S. Rao, and T. Panda, "Cyber-crimes and their impacts: A review," *International Journal of Engineering Research and Applications,* vol. 2, pp. 202-209, 2012.

[62] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *Internet Computing, IEEE,* vol. 17, pp. 20-27, 2013.

[63] Z. Hu, M. Wang, X. Yan, Y. Yin, and Z. Luo, "A comprehensive security architecture for SDN," in *Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on*, 2015, pp. 30-37.

[64] T. D. Nadeau and K. Gray, *SDN: software defined networks*: " O'Reilly Media, Inc.", 2013.

[65] M. Monaco, O. Michel, and E. Keller, "Applying operating system principles to SDN controller design," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, 2013, p. 2.

[66] J. M. Dover, "A denial of service attack against the Open Floodlight SDN controller," ed: Dover Networks LCC, 2013.

[67] H. Wang, L. Xu, and G. Gu, "Of-guard: A dos attack prevention extension in software-defined networks," *The Open Network Summit (ONS),* 2014.

[68] S. Cho, S. Chung, W. Lee, I. Joe, J. Park, S. Lee*, et al.*, "An Software Defined Networking Architecture Design Based on Topic Learning-Enabled Data Distribution Service Middleware," *Advanced Science Letters,* vol. 21, pp. 461-464, 2015.

[69] C. Dixon, D. Olshefski, V. Jain, C. Decusatis, W. Felter, J. Carter*, et al.*, "Software defined networking to support the software defined environment," *IBM Journal of Research and Development,* vol. 58, pp. 3: 1-3: 14, 2014.

[70] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient OpenFlow-based networking," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, 2012, pp. 933-939.

[71] H. Qi, M. Shiraz, A. Gani, M. Whaiduzzaman, and S. Khan, "Sierpinski triangle based data center architecture in cloud computing," *The Journal of Supercomputing,* vol. 69, pp. 887-907, 2014.

[72] S. Khan, M. Shiraz, A. W. Abdul Wahab, A. Gani, Q. Han, and Z. Bin Abdul Rahman, "A comprehensive review on adaptability of network forensics frameworks for mobile cloud computing," *The Scientific World Journal,* vol. 2014, 2014.

[73] S. Khan, A. Gani, A. W. A. Wahab, M. Shiraz, and I. Ahmad, "Network forensics: Review, taxonomy, and open challenges," *Journal of Network and Computer Applications,* vol. 66, pp. 214-235, 2016.

[74] S. Khan, E. Ahmad, M. Shiraz, A. Gani, A. W. A. Wahab, and M. A. Bagiwa, "Forensic challenges in mobile cloud computing," in *Computer, Communications, and Control Technology (I4CT), 2014 International Conference on*, 2014, pp. 343-347.

[75] O. A. Mahdi, A. W. A. Wahab, M. Y. I. Idris, A. A. Znaid, Y. R. B. Al-Mayouf, and S. Khan, "WDARS: A Weighted Data Aggregation Routing Strategy with Minimum Link Cost in Event-Driven WSNs."

[76] B. A. Thomas, N. Idris, A. Al-Hnaiyyan, R. Binti Mahmud, A. Abdelaziz, S. Khan*, et al.*, "Towards Knowledge Modeling and Manipulation Technologies: A Survey," *International Journal of Information Management,* 2016.

[77] C. Röpke and T. Holz, "SDN Rootkits: Subverting Network Operating Systems of Software-Defined Networks," in *Research in Attacks, Intrusions, and Defenses*, ed: Springer, 2015, pp. 339-356.

[78] D. He, S. Chan, and M. Guizani, "Securing software defined wireless networks," *Communications Magazine, IEEE,* vol. 54, pp. 20-25, 2016.

[79] J. Sonchack, A. J. Aviv, E. Keller, and J. M. Smith, "Enabling Practical Software-defined Networking Security Applications with OFX," 2016.

[80] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: Verifying network-wide invariants in real time," in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, 2013, pp. 15-27.

[81] M. Furukawa, K. Kuroda, T. Ogawa, and N. Miyaho, "Highly secure communication service architecture using SDN switch," in *Information and Telecommunication Technologies (APSITT), 2015 10th Asia-Pacific Symposium on*, 2015, pp. 1-3.

[82] A. Martin, "Dynamic filtering for sdn api calls across a security boundary," ed: Google Patents, 2014.

[83] H.-z. WANG, P. ZHANG, L. XIONG, X. LIU, and C.-c. HU, "A secure and high-performance multi-controller architecture for software defined networks," *Frontiers,* vol. 1.

[84] S. Shenker, M. Casado, T. Koponen, and N. McKeown, "The future of networking, and the past of protocols," *Open Networking Summit,* vol. 20, 2011.

[85] 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*, 2012, pp. 55-60.

[86] C. Staff, "A purpose-built global network: Google's move to SDN," *Communications of the ACM,* vol. 59, pp. 46-54, 2016.

[87] A. Zaalouk, R. Khondoker, R. Marx, and K. Bayarou, "OrchSec: An orchestrator-based architecture for enhancing network-security using Network Monitoring and SDN Control functions," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, 2014, pp. 1-9.

[88] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: Detecting Security Attacks in Software-Defined Networks," in *NDSS*, 2015.

[89] S. Hwang and K. Kim, "Middlebox Driven Security Threats in Software Defined Network."

[90] S. Jajodia, S. Noel, and B. O'Berry, "Topological analysis of network attack vulnerability," in *Managing Cyber Threats*, ed: Springer, 2005, pp. 247-266.

[91] V. N. Padmanabhan and D. R. Simon, "Secure traceroute to detect faulty or malicious routing," *ACM SIGCOMM Computer Communication Review,* vol. 33, pp. 77-82, 2003.

[92] D. Pei, D. Massey, and L. Zhang, "Detection of invalid routing announcements in rip protocol," in *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*, 2003, pp. 1450-1455.

[93] A. T. Mizrak, Y.-C. Cheng, K. Marzullo, and S. Savage, "Detecting and isolating malicious routers," *Dependable and Secure Computing, IEEE Transactions on,* vol. 3, pp. 230-244, 2006.

[94] D. Katz, K. Kompella, and D. Yeung, "Traffic engineering (TE) extensions to OSPF version 2," RFC 3630, September2003.

[95] J. Moy, P. Pillay-Esnault, and A. Lindem, "Graceful OSPF restart," *RFC3623, November,* 2003.

[96] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler*, et al.*, "Optimized link state routing protocol (OLSR)," 2003.

[97] J. Choudhary, "Distributed BPDU processing for spanning tree protocols," ed: Google Patents, 2010.

[98] A. Ornaghi and M. Valleri, "Man in the middle attacks," in *Blackhat Conference Europe*, 2003.

[99] R. Kloti, V. Kotronis, and P. Smith, "OpenFlow: A security analysis," in *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, 2013, pp. 1-6.

[100] J. C. Mogul, A. AuYoung, S. Banerjee, L. Popa, J. Lee, J. Mudigonda*, et al.*, "Corybantic: towards the modular composition of SDN control programs," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, 2013, p. 1.

[101] M. Aslan and A. Matrawy, "On the Impact of Network State Collection on the Performance of SDN Applications," 2016.

[102] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, *et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review,* vol. 38, pp. 69-74, 2008.

[103] C. Scott, A. Wundsam, B. Raghavan, A. Panda, A. Or, J. Lai*, et al.*, "Troubleshooting blackbox SDN control software with minimal causal sequences," in *ACM SIGCOMM Computer Communication Review*, 2014, pp. 395-406.

[104] P. Goncalves, A. Martins, D. Corujo, and R. Aguiar, "A fail-safe SDN bridging platform for cloud networks," in *Telecommunications Network Strategy and Planning Symposium (Networks), 2014 16th International*, 2014, pp. 1-6.

[105] J. Hollander, *A Link Layer Discovery Protocol Fuzzer*: Citeseer, 2007.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/COMST.2016.2597193, IEEE Communications Surveys & Tutorials

[106] L. Ochoa Aday, C. Cervelló Pastor, and A. Fernández Fernández, "Current Trends of Topology Discovery in OpenFlow-based Software Defined Networks," 2015.

[107] M. Karakus and A. Durresi, "A Scalable Inter-AS QoS Routing Architecture in Software Defined Network (SDN)," in *Advanced Information Networking and Applications (AINA), 2015 IEEE 29th International Conference on*, 2015, pp. 148-154.

[108] F. Yu and V. Leung, "Mobility-based predictive call admission control and bandwidth reservation in wireless cellular networks," *Computer Networks,* vol. 38, pp. 577-589, 2002.

[109] M. Karimzadeh, L. Valtulina, and G. Karagiannis, "Applying sdn/openflow in virtualized lte to support distributed mobility management (dmm)," 2014.

[110] S. Namal, I. Ahmad, A. Gurtov, and M. Ylianttila, "Sdn based inter-technology load balancing leveraged by flow admission control," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, 2013, pp. 1-5.

[111] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown*, et al.*, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep,* pp. 1-13, 2009.

[112] W. You, K. Qian, X. He, Y. Qian, and L. Tao, "Towards security in virtualization of SDN," in *Proceedings of the International Conference on Computer Communications and Networks Security, ser. ICCCNS*, 2014.

[113] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 7-12.

[114] T. Nelson, A. D. Ferguson, M. J. Scheer, and S. Krishnamurthi, "Tierless programming and reasoning for software-defined networks," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, 2014, pp. 519-531.

[115] A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, and R. Martinez, "Experimental Seamless Virtual Machine Migration Using a SDN IT and Network Orchestrator," 2015.

[116] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide*, et al.*, "ONOS: towards an open, distributed SDN OS," in *Proceedings of the third workshop on Hot topics in software defined networking*, 2014, pp. 1-6.

[117] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore, "OFLOPS: An open framework for OpenFlow switch evaluation," in *Passive and Active Measurement*, 2012, pp. 85-95.

[118] K. Bakshi, "Considerations for software defined networking (SDN): Approaches and use cases," in *Aerospace Conference, 2013 IEEE*, 2013, pp. 1-9.

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

[120] S. Namal, I. Ahmad, A. Gurtov, and M. Ylianttila, "Enabling secure mobility with openflow," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, 2013, pp. 1-5.

[121] H. Wang, L. Xu, and G. Gu, "FloodGuard: a dos attack prevention extension in software-defined networks," in *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*, 2015, pp. 239-250.

[122] A. Gani, G. M. Nayeem, M. Shiraz, M. Sookhak, M. Whaiduzzaman, and S. Khan, "A review on interworking and mobility techniques for seamless connectivity in mobile cloud computing," *Journal of Network and Computer Applications,* vol. 43, pp. 84-102, 2014.

[123] S. Khan, A. Gani, A. W. A. Wahab, and M. A. Bagiwa, "SIDNFF: Source identification network forensics framework for cloud computing," in *Consumer Electronics-Taiwan (ICCE-TW), 2015 IEEE International Conference on*, 2015, pp. 418-419.