Software-Defined Networks Topology Discovery Security and Drawbacks: A Survey of Attacks and Defenses

Ali Abduljabbar Ali , Abdullah Muhammed , Muhammad Daniel Hafiz Abdullah , Amir Rizaan Abdul Rahiman

***Abstract*— Topology discovery is a core service in Software-Defined Networks (SDN), enabling controllers to monitor active network links through the OpenFlow mechanism. However, this process is vulnerable to critical security threats such as Fabricated LLDP Injection, Host Location Hijacking, LLDP Replay, Port Amnesia, and Persona Hijacking. These vulnerabilities compromise network integrity, scalability, and performance. This paper reviews these challenges and evaluates state-of-the-art countermeasures, analyzing their security effectiveness, performance impact, and scalability. Emerging trends, such as machine learning and real-time anomaly detection, are highlighted as key to developing lightweight, adaptive solutions. By presenting a quantitative analysis of existing mechanisms, this study emphasizes the need for hybrid approaches that balance robust security with efficient performance, offering a roadmap for advancing SDN topology discovery security.**

Keywords*— *Software Defined Networks (SDN); Topology discovery; Security; OpenFlow

I. Introduction

Software-defined networking (SDN) is quickly transitioning from a concept to a tangible reality, as a wide range of devices that support SDN are being developed and manufactured.

The integration of distinct control and data plane functionality, together with the ability to be programmed, has transitioned from a topic of discussion in the research community to being commercially implemented in cloud computing and virtualization technologies[1]. Over networks, the SDN architecture offers programmability, flexibility, and stability. Using popular programming languages, network operators can implement their protocols, rules, and policies. They can attain adaptable control over network functions such as traffic engineering, routing, QoS, and security [2]. The network may adjust to the needs of its users. The centralized controller and open API standard allow for automatic configuration and management of the network, facilitating easy network scalability. Administrators can improve devices in the data plane without altering the control plane or add functionality to the control plane without changing the data plane by using SDN [2], [3]. This is also important because the control plane cannot easily be separated logically from the infrastructure and because taking it out of the infrastructure lowers the expenses and reduces discomfort with new architectures or provisional proposals to experiment with in networks. SDN architecture models include three components or functional categories. as shown in Fig. 1:

- SDN Applications Layer: SDN Applications are applications that are capable of delivering to the SDN Controller the properties of interaction behaviors and pertinent resources through application programming interfaces (APIs). In addition, the apps can construct a condensed version of the network by being able to pull data from the controller for a specific purpose of decision-making. These programs may contain networking administration, analysis, or business applications used for the conductivity of large data centers. For example, an analytics program can be created to analyze potential threats on the network to improve protection.
- SDN Controller layer: The SDN Control layer works as a logical layer in the SDN framework that receives the instruction or the requirement from the higher layer notably the SDN Application layer and forwards them to the networking elements accordingly. The controller retrieves network information from the hardware devices and relays an abstract representation of the network, including statistics and events, to the SDN Applications.
- The SDN data layer: primarily consists of SDN networking devices, which are responsible for controlling the network's forwarding and data processing capabilities. This

Manuscript received Jan 22, 2024; revised Mar 28, 2025.

This work was supported by PHD student Ali Abduljabbar Ali.

Ali Abduljabbar Ali is a PhD candidate of Faculty of Computer Science and Information Technology, University Putra Malaysia, 43400 UPM Serdang, Malaysia (corresponding author, e-mail: alialian8989@gmail.com).

Abdullah Muhammed is a professor at the Faculty of Computer Science and Information Technology, University Putra Malaysia, 43400 UPM Serdang, Malaysia (e-mail: abdullah@upm.edu.my).

Muhammad Daniel Hafiz Abdullah is a senior lecturer of the Faculty of Computer Science and Information Technology, University Putra Malaysia, 43400 UPM Serdang, Malaysia (e-mail: daniel_hafiz@upm.edu.my).

Amir Rizaan Abdul Rahiman is a senior lecturer of the Faculty of Computer Science and Information Technology, University Putra Malaysia, 43400 UPM Serdang, Malaysia (e-mail: amir_r@upm.edu.my).

encompasses the act of transmitting and handling the data pathway.

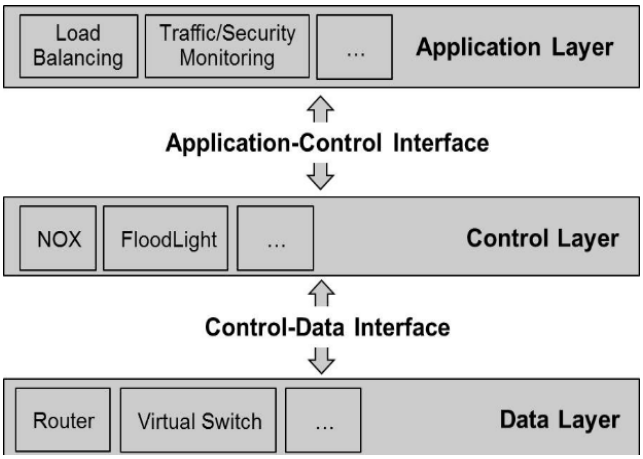


Fig. 1. SDN Architecture

OpenFlow is a protocol that establishes the southbound interface in SDN, serving as the connection between the control layer and the data layer. OpenFlow serves as a communication interface that enables the SDN controller to configure and manage SDN switches. Although there are alternative protocols suggested and used as the south-bounding interface, such as SNMP, BGP, PCEP, etc., Open Flow, advocated by the Open Network Foundation (ONF), is presently the prevailing standard. The discovery of existing links between switches in the OpenFlow network can be performed with the single-hop neighbor Link Layer Discovery Protocol LLDP. No further discovery techniques are necessary to identify the connections in a network consisting solely of OpenFlow elements. The reason for this is that there is a switch at the end of each link that facilitates the topology discovery approach [4]. Switches that adhere to the OpenFlow technical guidelines have two primary starting configurations that facilitate the discovery of the network topology [5]. Each OF switch is originally configured with the IP address and TCP port of a master controller, as well as a pool of IP addresses for slave controllers. This allows the switch to establish a connection with the controllers as soon as it is powered on. Additionally, the switches are equipped with preconfigured flow rules that enable them to send any message with an LLDP packet (identified by the 0x88ccEtherType) received on a separate port directly to the controller via a Packet-In message. [3].

In this paper, a review of SDN topology discovery security issues is presented, in section 2, the limitation of the default OpenFlow topology discovery algorithm is presented, and section 3 presents the attacks that target the topology discovery process and the expected defense mechanism. In section 4, state-of-the-art secure topology approaches are presented demonstrating the latest efforts to overcome the limitation of OpenFlow topology discovery vulnerabilities. Section 5 presents comprehensive results and conclusion results.

II. THE LIMITATION OF SDN TOPOLOGY DISCOVERY: AUTHENTICATION AND AUTHORIZATION

The several specific challenges and constraints that can affect the overall security and efficiency of SDN environments. SDN faces several specific challenges and constraints that can significantly impact its security and efficiency. These challenges include and impact on:

1. Vulnerability to Unauthorized Access

Since the controllers are centralized, non-authorized subjects may try to compromise SDN controllers if the authentication is weak or implemented inappropriately. Employees and visitors gaining access to network settings may perform fraudulent changes to several configurations that lead to interruptions of network schedules and loss of data purity [6], [7].

2. Scalability Issues

However, as the number of devices and users increases and the network continues to grow, the process of managing authentication and permission becomes complex. The scalability issues may result in the types of delays that hinder the identification of complex networking structures, as well as increase the managerial load of processing requests for authentication; this leads to the subsequent reduction in the ability of the network to provide quick responses. [6], [8].

3. Single Point of Failure

The next element is the SDN controller, which is indeed a crucial component in the integration of control and management planes. When an authentication mechanism does not evoke the proper reaction, the controller is the only failure point. The infiltrated controller can become a severe threat to the whole network since it creates numerous areas of vulnerability and weak points that can eventually bring negative impacts to the overall network security. [9], [10].

4. Complexity in Policy Management

The full set-up and enforcement of complete authentication and authorization policies throughout the components of a computer network can be challenging. Security policies, if not properly applied may cause safety gaps and enable criminals or unauthorized personnel access to some parts of the network. [11].

5. Performance Overhead

Applying strong authentication and permission procedures may cause some compromise in performance because of additional processing. Latency and system throughput, as a result, can negatively impact the identification of topology and the comprehensive performance of a network [6].

6. Interoperability Concerns

The overall flow of authentication may differ in the technological platforms used by various companies and devices. Achieving tight cooperation between many components can be challenging,

maybe because it might lead to insecure interfaces if the integration is not managed well. [6].

7. Dynamic Environment Challenges

SDN environments are generally known by the dynamic mode of their operation, change of topology, and dynamic change of device states. Who knows, maintaining the present data of the current authentication and permission could be a daunting task and might create a gap that can be penetrated by hackers. [9], [10].

8. User and Device Identity Management

Managing the identities and certificates for a large number of users and various types of devices within the network can be challenging. Deficient approaches to identity management might lead to unauthorized access issues in addition to poor account tracking and appropriate documenting activities. [11].

9. Insider Threats

Indeed, if a user or device has been given permission or has gained access to a particular network, this may be a threat even if the credentials of such an individual or devices are stolen or the user or device's intention is malevolent. Preventing insider threats is a little bit more demanding as compared to the case with outsider threats, this is because the insiders are often persons who are already permitted access to the networks. [6].

10. Compliance and Regulatory Requirements

Adhering to distinct standards and policies set by different regulations and industries may make it challenging to implement the necessary authentication and authorization measures. Non-compliance can lead to legal and financial repercussions, as well as potential security vulnerabilities if standards are not adequately met [4], [5].

As the process of network discovery is a very critical task in implementing SDN, controllers through network discovery collect the required information to provide correct data forwarding decisions [6]. Data can be forwarded based on different factors like link availability, Quality of Services, and best route selection. The discovery messages are sent in a clear format without any device authentication in the default topology discovery mechanism [7]. As shown in Fig 2, after sending the hello messages, the controller initiates the handshake by sending a Feature-Request message to the switch, which answers with a feature-replay message. This message provides the controller with important parameters for discovering links, such as the Switch ID and a list of active ports together with their corresponding MAC addresses. At this stage, the controller possesses precise knowledge of the OpenFlow switches that are connected to the network and has valuable information to carry out the discovery of the links. This information is obtained from the initial handshake. Additionally, the controller is aware of the exact number of active ports on the OpenFlow switches that are part of the administrative domain [14]. The controller produces a Packet-Out message for every operational port on every switch found in the network and includes an encapsulated LLDP packet within each message. [6].

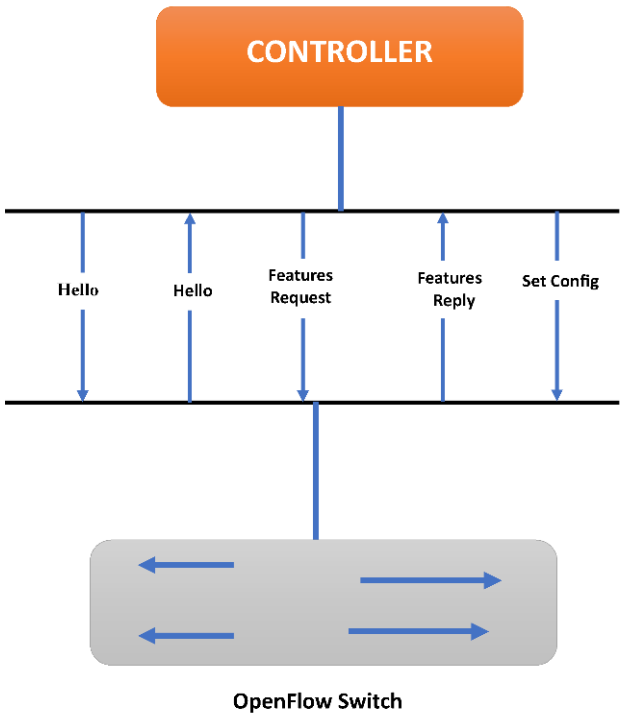


Fig. 2. OpenFlow Topology Discovery

All these messages are clear and not authenticated, so malicious nodes can be connected and poison the topology discovery process. Clear discovery messages can be sniffed easily and thus manipulated. Modified discovery messages can lead to incorrect data forwarding decisions where data can be hacked or dropped [8].

Topology discovery can be targeted by a range of approaches. An assailant can effortlessly insert counterfeit LLDP packets into the network, known as the Fabricated LLDP Injection attack [16], or replicate authentic LLDP packets from one target switch to another, known as the LLDP Replay attack [17]. This can result in the corruption of the controller's link information [18]. A LLDP Replay attack involves the forwarding of legal LLDP packets from one target switch to another, without any alteration. This results in the controller receiving faked connection information [9].

Recent researchers have proposed different mechanisms for handling security in the topology discovery of SDN, however, most of this research mainly depends on third-party certificate authority or high complex encryption mechanisms [10]. A lightweight and efficient topology discovery design is required to perform efficient and secure topology discovery.

III. EXISTING AND TAXONOMY TOPOLOGY ATTACKS AND DEFENSES

The SDN always puts it under attack and should make an explanation for the taxonomy of all the attacks and defenses and summarize the current assaults on network topology and the

recommended countermeasures to minimize them. and Fig.3. Taxonomy of attacks in SDN

1. *Link Fabrication Attacks[11], [12]:*
- *Fabricating Nonexistent Links*: Creating fake links between network devices, causing incorrect network maps and routing.
- *Modifying Link Weights*: Changing the importance of links to control traffic flow in undesirable ways.
- *Control Plane Deception*: Providing false information about link status to mislead network management.
- *Routing Manipulation*: Using fake links to alter traffic paths for eavesdropping or causing congestion.

2. *Topology Poisoning Attacks* [13], [14], [15], [16], [17]*:*
- *Fabricated LLDP Injection Attack*: Injecting fake Link Layer Discovery Protocol (LLDP) messages to alter network topology.
- *Host Location Hijacking Attack*: Misleading the controller approximately the bodily region of community devices.
- *LLDP Replay Attack*: Replaying antique LLDP messages to confuse the community topology.
- *Poisoning Network Discovery Protocols*: Manipulating protocols that discover network layouts to create fake maps.
- *Fake Flow Table Entries*: Adding false entries to go with the flow tables to disrupt traffic routing.
- *Injecting False Topology Information*: Providing wrong community format facts to the controller.
- *Virtual Topology Manipulation:* Creating digital topologies that don't match the bodily community.
- *Deceptive Node Positioning*: Misleading the controller about node places or roles.
- *Path Distortion*: Altering paths within the community to lead them to inefficient or insecure.

3. *Switch Flooding Attacks*[18], [19], [20], [21], [22]*:*
- *Denial-of-Service (DoS) Attack*: Overwhelming community switches to purpose failures.
- *Flow Table Overflow*: Filling up a switch's float desk so it can't take care of new or current flows.
- *Resource Exhaustion*: Consuming a switch's CPU and reminiscence assets with excessive visitors or complex guidelines.
- *Packet Flooding*: Sending too many packets to a transfer, overwhelming its processing capability.
- *Service Degradation*: Consistent flooding that lowers the high-quality of network offerings.

4. *Data Plane Attacks* [23], [24], [25]*:*
- *Denial-of-Service (DoS) Attack*: Disrupting everyday network operations by overwhelming assets.
- *Data Plane Misconfiguration*: Incorrectly configuring network devices, main to accidental behavior.
- *Traffic Flooding*: Sending excessive site visitors to degrade community performance.
- *Packet Injection*: Adding malicious packets to the community to disrupt verbal exchange or compromise gadgets.
- *Flow Rule Manipulation*: Changing how packets are processed to misroute or drop traffic.
- *Traffic Analysis*: Monitoring traffic styles to collect sensitive facts.
- *Data Interception*: Capturing and potentially altering information packets in transit.

5. *Controller Attacks*[26], [27], [28]*:*
- *Controller Exploitation*: Finding and the usage of vulnerabilities within the controller software program.
- *Denial of Service*: Overloading the controller with requests to make it unresponsive.
- *Privilege Escalation*: Gaining a higher right of entry to perform unauthorized actions.
- *Code Injection*: Inserting malicious code into the controller to alternate its behavior.

6. *Control Plane Manipulation[12], [29], [30]:*
- *Persona Hijacking*: Taking over a legitimate consumer or device identity to benefit an unauthorized manager.
- *Flow Rule Injection*: Adding malicious flow policies to alternate community behavior.
- *Controller Spoofing*: Pretending to be a valid controller to manipulate network operations.
- *Routing Protocol Attacks*: Disrupting routing protocols to misinform network devices.
- *Manipulation of Control Plane Messages*: Interfering with messages among the controller and community gadgets.
- *Message Interception*: Capturing control plane messages to acquire statistics or modify conversation.
- *Message Modification*: Changing the contents of control plane messages to disrupt operations.
- *Fake Control Messages*: Injecting false messages to control community gadgets.
- *Session Hijacking*: Taking over communique periods among the controller and devices.

7. *Link Removal Attacks[20], [28], [31]:*
- *Network Partitioning*: Breaking the community into isolated segments.
- *Disruption of Communication*: Interfering with conversation between community segments.
- *Control Plane Target*: Specifically concentrated on managing plane hyperlinks.
- *Deliberate Removal of Links*: Removing links to disrupt community operations.
- *Falsifying Link Failures*: Reporting false link failures to reason needless rerouting.
- *Physical Link Disruption*: Tampering with bodily community links.
- *Virtual Link Removal*: Removing digital hyperlinks to disrupt community paths.

- *Traffic Rerouting*: Forcing visitors onto inefficient or insecure paths.

8. *Node Forgery Attacks[15], [32], [33]:*

- *Routing Misdirection*: Misleading routing decisions with faux nodes.
- *Misleading Network Management*: Providing false records about network nodes.
- *Control Plane Manipulation*: Using fake nodes to control control aircraft choices.
- *Falsifying Node Identity*: Pretending to be legitimate nodes.
- *Fake Node Introduction*: Adding unauthorized nodes to intercept or disrupt traffic.
- *Impersonation of Legitimate Nodes*: Gaining unauthorized get entry by way of impersonating nodes.
- *Man-in-the-Middle*: Positioning faux nodes to intercept communications.
- *Node Data Manipulation*: Changing data pronounced by using cast nodes.

9. *Node Removal Attacks*[34], [35], [36]*:*

- *Network Partitioning*: Causing isolation of community segments through casting off nodes.
- *Disruption of Communication*: Disrupting communication by way of removing nodes.
- *Control Plane Target*: Specifically targeting control aircraft nodes.
- *Deliberate Node Removal*: Removing nodes to disrupt network operations.
- *False Node Failure Reports*: Reporting non-existent node disasters. Unauthorized Node Shutdown: Forcibly shutting down nodes.
- *Node Isolation*: Isolating nodes from the network.
- *Service Disruption*: Disrupting services by removing nodes.

10. *Inter-controller Link Manipulation*[28], [37], [38], [39]*:*

- *Routing Misdirection*: Misleading routing decisions by way of manipulating inter-controller hyperlinks.
- *Control Plane Target*: Specifically concentrated on control plane hyperlinks between controllers.
- *Manipulation of Inter-controller Links*: Tampering with links between controllers.
- *Interference with Controller Communication*: Disrupting communique among controllers.
- *Communication Interruption*: Disrupting links to interrupt controller coordination.Message Tampering: Changing messages between controllers.
- *Link Creation/Removal*: Falsely reporting the creation or removal of links.
- *Controller Synchronization Attacks*: Causing desynchronization between controllers.

It is essential to examine a whole range of defensive tactics to reduce the impact of different attack methods aimed at the SDN infrastructure. The defenses can be categorized as various essential domains, each addressing distinct facets of safeguarding the SDN environment.

1. Controller Authentication and Authorization[40]

- *Secure Controller Access***:** Implementing strong authentication mechanisms to make certain legal employees access the SDN controller.
- *Role-Based Access Control (RBAC)***:** Enforcing strict access manage regulations that furnish permissions based totally on person roles and duties, minimizing the danger of unauthorized access.

2. Flow Verification and Consistency[41], [42], [43]

- *Flow Table Verification***:** Regularly verifying the contents of float tables in SDN switches to make sure they align with meant community regulations and configurations, preventing unauthorized adjustments.
- *Flow Table Integrity Monitoring***:** Continuous tracking for any adjustments or anomalies in waft tables, detecting unauthorized or suspicious entries in actual-time.

3. SDN Monitoring and Intrusion Detection[44], [45], [46]

- *Anomaly Detection Systems (ADS)***:** Utilizing tools that constantly scan and monitor packets going through the network and using pattern recognition tools and algorithms to detect deviations from typical traffic patterns as signs of a security threat or a breach.
- *Intrusion Detection Systems (IDS)***:** Using IDS to scan the network traffic for patterns that are similar to those of emerging cyber threats or/and for activities that are outside the norms which allow generating alarms and initiating the appropriate reaction.
- *Traffic Monitoring Tools*: Implementing comprehensive site visitors monitoring equipment to offer real-time visibility into network sports and come across anomalies early.

4. Secure Communication Channels[47]

- *Encryption of Control Messages***:** Encrypt communication between SDN controllers and switches to defend against eavesdropping and tampering.
- *Secure Communication Protocols (TLS, SSH)***:** Using steady protocols along with Transport Layer Security (TLS) and Secure Shell (SSH) to ensure the confidentiality and integrity of manipulated messages.
- *Secure SDN Controller-Device Communication:* Establishing secure communique channels between the SDN controller and community gadgets to prevent unauthorized access and data breaches.

5. Behavioral Analysis and Machine Learning [48], [49]

- *Behavioral Analysis of Network Traffic*: Analyzing network site visitor patterns to pick out normal conduct and locate deviations that would imply malicious sports.

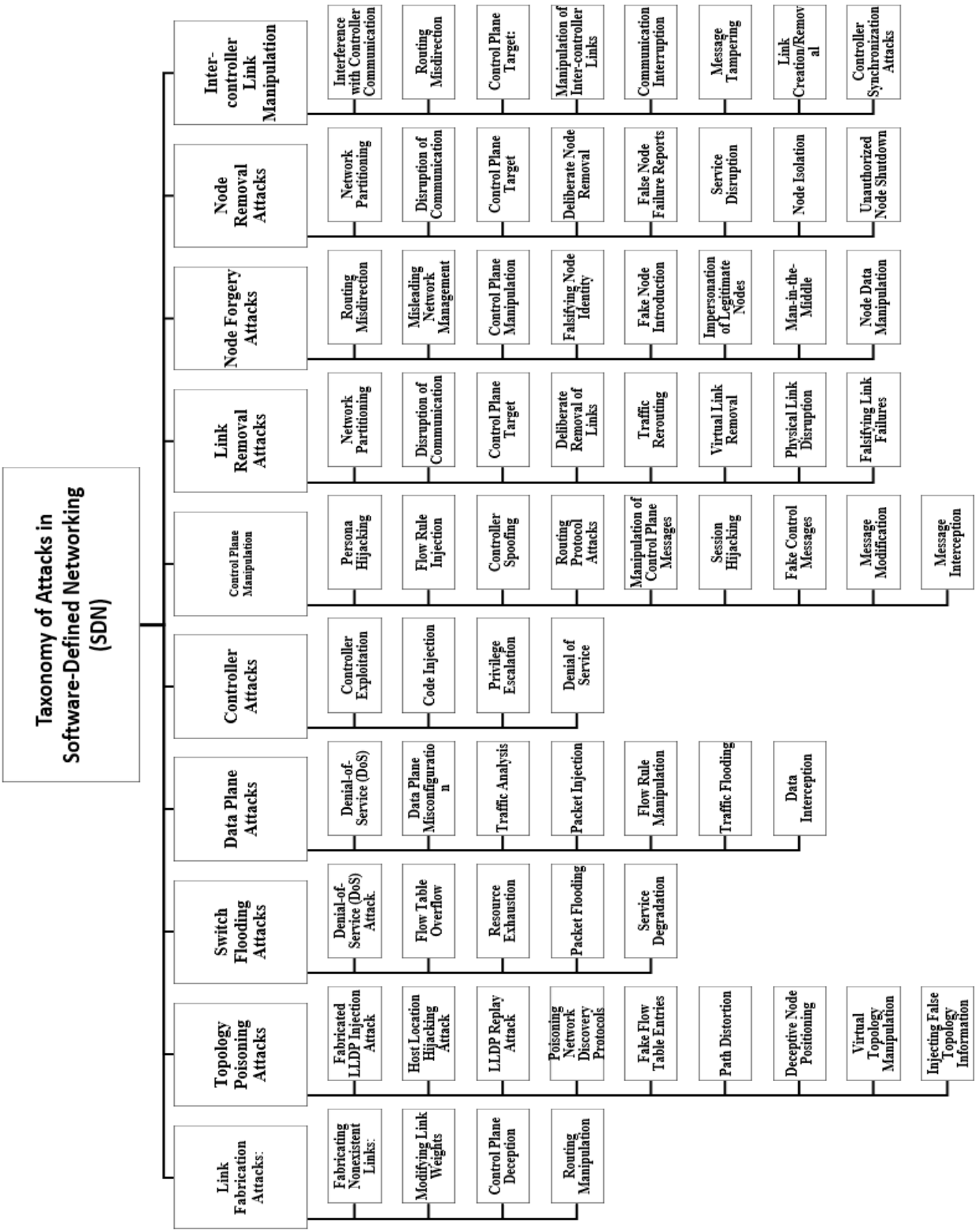


Fig.3. Taxonomy of attacks in SDN

Machine Learning-Based Anomaly Detection: Leveraging gadgets and gaining knowledge of algorithms to perceive and expect anomalies primarily based on historical and actual-time statistics, improving detection talents.

- *Predictive Security Analytics:* Using predictive analytics to forecast capability safety threats and vulnerabilities, allowing proactive protection measures.

6. Robust and Resilient Network Design [50], [51]

- *Redundancy and Failover Mechanisms:* Designing the community with redundancy and failover mechanisms to make certain endured operations even though certain additives are compromised.
- *Segmented Network Architecture:* Implementing network segmentation to comprise capacity breaches and limit the spread of assaults within the network.

Fig.4. Taxonomy of defenses in SDN

7. Comprehensive Security Policies and Practices [52], [53]

- *Regular Security Audits:* Conducting everyday security audits and tests to become aware of and deal with vulnerabilities in the SDN infrastructure.
- *Incident Response Planning:* Developing and regularly updating incident response plans to ensure a swift and powerful response to safety incidents.
- *Continuous Training and Awareness:* Ensuring that community administrators and safety employees are constantly trained and privy to modern security threats and protection techniques.

This section provides current topology assaults are briefly discussed in this section, and so is the defense measure aimed at fighting the assault. A topology attack is often based on the assumption that the adversary has a certain level of knowledge of the network topology; he can obtain information about the network identifiers of the hosts [62]. The above-stated assumptions can be made because the model is built based on realistic expectations.

The network topology information can be obtained using the standard path trace utilities such as traceroute or through the actual probing [63]. For example, like traditional networks, current SDN networks still employ non-secure protocols like ARP or DHCP. Thus, the adversaries can quickly obtain the network identities of the measured hosts.

The security vulnerability in LLDP lies in the fact that the controller automatically accepts any LLDP packet it receives, and uses the link information contained in it to update its link information. However, the SDN controller lacks a built-in mechanism to guarantee the integrity or verify the source of LLDP packets. Consequently, an assailant can effortlessly insert counterfeit LLDP packets into the network or replicate the authentic LLDP packets from a specific switch to another switch, thus compromising the link information of the controller. The attack scenarios are demonstrated as follows:

A. Fabricated LLDP Injection

In this attack scenario, the attacker produces counterfeit LLDP packets within a network to falsely announce a non-existent link between two switches. The link spoofing attack, which is initiated via a falsified LLDP packet, is explained with an. Example shown in Fig 5.

In the fabricated LLDP packet attack, It is assumed that host h1 has been infiltrated by an attacker who wants to establish a phony link between switch *s1* and *s3* to illustrate the faked LLDP packet attack. From the LLDP packet that host h1 receives from switch *s1*, it becomes aware of the LLDP syntax. Then, it sends LLDP packets to switch *s1* that contain faked data, such as switch s3 Port *ID = 1* and DPID = s3. Upon receiving the LLDP packets, switch s1 passes them to the controller using the Packet-In message, which additionally includes switch *s1* ingress port information (*Port ID = 1*). Next, the controller obtains the LLDP information, which includes the switch *s1*'s ingress port *Port ID = 1* and *DPID = s3.* The controller updates a fictitious link from *(s3,1) to (s1,1)* as a result.

One method to protect against this attack approach for a specific target is through the integration of a controller-signed authenticator TLV(Type-Length-Value) to the LLDP packet, and consecutively, check the authenticity of the signature given when the target receives the LLDP packets. The HMAC technique is used to compute the signature incorporating DPID and the Port ID of the LLDP in question. In this scenario, the controller can detect any attempt at forging the LLDP packets from the attacker's end. However, as Nakibly also proved this method does not work effectively against other sorts of attacks, for instance, the LLDP Replay-based link spoofing attack.

To counter link fabrication attacks, the authors in [17] proposed retaining the purity of the transmitted LLDP packets and denying LLDP involvement to hosts that should not be involved in the LLDP relay. One of the fields that is embedded in LLDP and signed by the controller is a field necessary to preserve the integrity of the protocol. This field is derived from the DPID

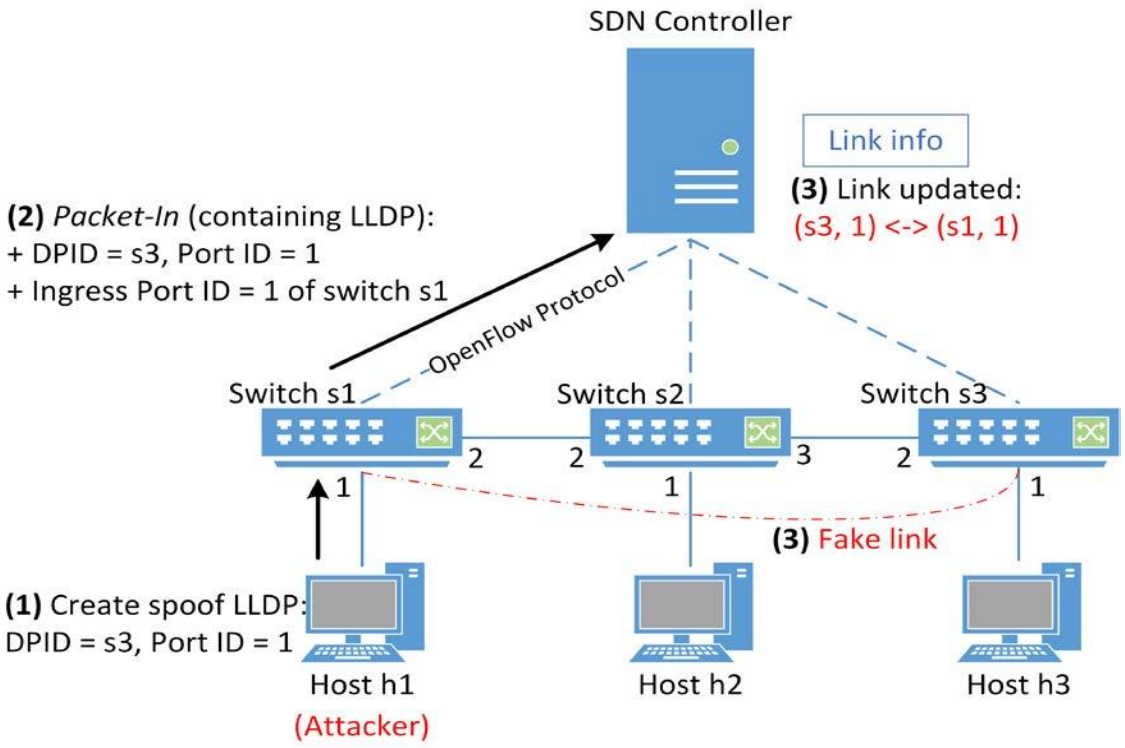


Fig. 5. The link spoofing attack: Fabricated LLDP Injection

and port number of the source switch. This prevents other persons from encroaching and using the LLDP to craft more fake packets to introduce other devices they never have.

To guarantee the uniqueness of the recipients of the LLDP packets, it was becoming necessary to determine the type of device connected to each switch port in a port-labeling manner. This technique takes into account three potential states: Host directed, switch-directed, or any direction. "*HOST*" points to a connected host on any of the switch ports. It is used as "*SWITCH*" if the switch is connected, and as "*ANY*" if no device is connected. At first, all switch ports are designated as ANY. The port label is modified according to the initial type of traffic received by the switch on each port. Nevertheless, their strategy must have the capability to disregard the port kind. It is important to remember that SDN networks are designed to be utilized in dynamic contexts, where a host can be disconnected and replaced by a switch, or vice versa. To fulfill this need, one can reset the port type to ANY whenever a port-down event occurs, such as when the host disconnects from the switch.

This port-labeling approach effectively thwarts adversaries that have access over several hosts from relaying LLDP packets, as these packets are exclusively transmitted to switch ports.

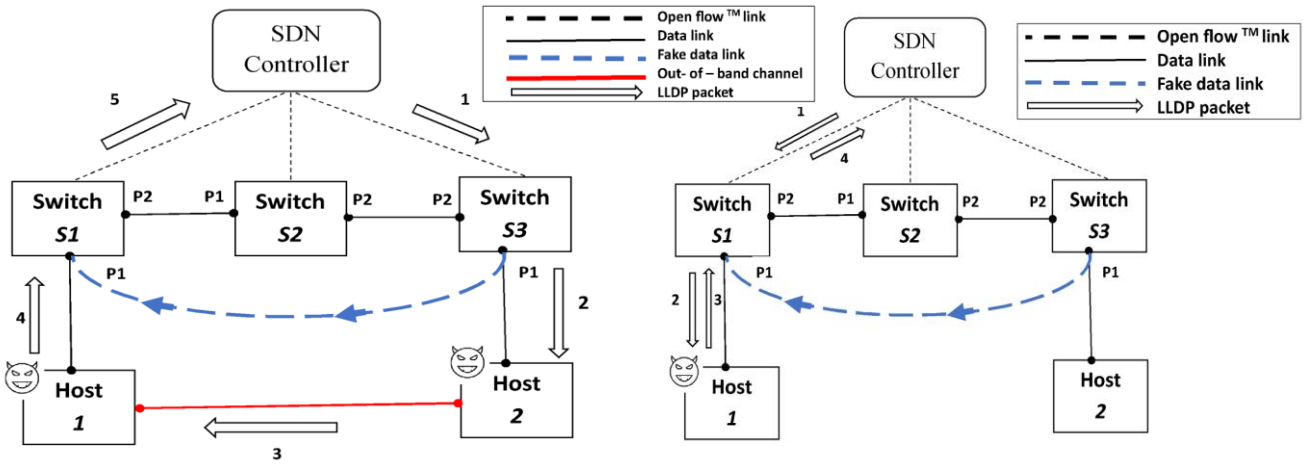
B. Host location hijacking

The phenomena of *host location hijacking* and link fabrication were first reported in [17] and have since garnered considerable attention in recent years. In addition, they unveiled TopoGuard, a security system that effectively prevents adversaries from carrying out any type of assault. TopoGuard assumes the presence of adversaries who can manipulate one or more hosts, specifically the controller and the switches, which are assumed to be completely trustworthy. TopoGuard relies on SSL/TLS to safeguard the control channel between the controller and the switches.

In a *host location hijacking* attack, hackers attempt to deceive the controller into believing that a victim's host has been relocated to a different network location. To achieve this objective, opponents who have control over one or more hosts can transmit packets utilizing the network identifiers, such as the IP and/or MAC address, of the targeted host. This triggers the Host Tracking Service to refresh the network position data of the target's host. This attack can be effectively executed as long as the targeted host remains inactive. Conversely, in a *link fabrication attack*, adversaries aim to build counterfeit linkages between switches, as demonstrated in the preceding section.

Fig 6 shows There are multiple methods that adversaries can employ to generate such fake links. Adversaries can alter genuine LLDP packets or create authentic ones. Another method involves transmitting LLDP packets between two network locations using either an in-band or an out-of-band channel. In each of these instances, the opponent successfully deceives the controller into believing the presence of a new inter-switch link, despite the absence of such a link. The opponent will drop or intercept all packets that pass over this link.

To ensure the validity of host migration, it is necessary to conduct pre- and post-conditions checks in response to a host



(a) Link fabrication attack where H2 relays LLDP packets to H1 using an out-of-band channel.

(b) Link fabrication attack where H1 crafts and sends LLDP packets identical to those that originate from S3 port 1.

Fig. 6: A link fabrication attack is carried out utilizing two distinct methodologies: (a) transmitting LLDP packets across an out-of-band channel and (b) falsifying LLDP packets. In both scenarios, the controller has the belief that there is a one-way conection from *S3* port *1* to *S1* port *1*.

location hijacking assault. The rationale behind this technique is that every authentic host migration results in a sequence of events that must take place in a specific order. Essentially, this means that a host must first disconnect from its existing network location before it can connect to a new switch. Specifically, the controller initially waits to receive a port-down signal from the switch to which the host is connected. It then verifies if the host is still accessible at its original network position. The controller will only acknowledge the migration as valid and allow the host to send packets from its new network location if these conditions are met. Although this countermeasure is straightforward, it allows for the identification of assaults in which a host seems to be present in two locations at the same time.

C. LLDP Replay

In the LLDP Replay attack, the attacker transmits the original LLDP packets from one target switch to another, without making any modifications to the packets. This leads to the controller receiving false link information.

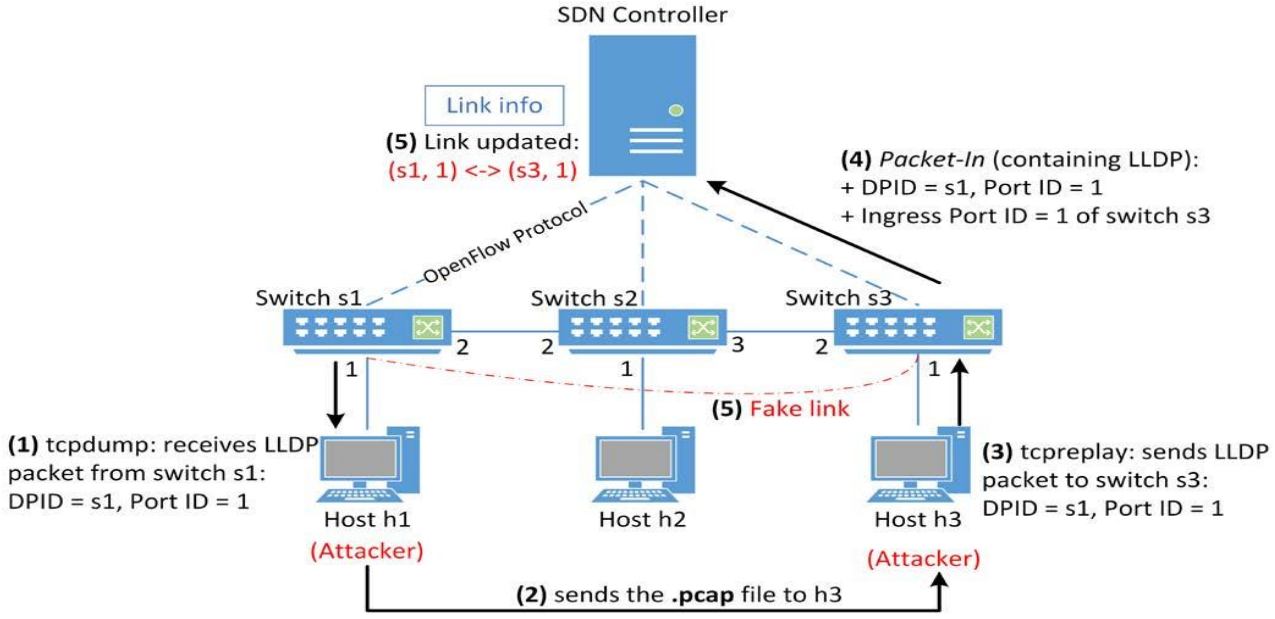


Fig. 7. Spoofing attack:- LLDP Replay

Topology in Fig 7 Illustrates the LLDP reply attack. It is presumed that the attacker has control over the hosts *h1* and *h3*. Additionally, the LLDP packet is protected by an additional HMAC-based controller-signature TLV, which utilizes a single secret key *K*. During the regular LLDP broadcast, *host h1* will receive LLDP packets with the DPID value of *s1* and the *Port ID* value of *1*. Host *h1* employs a packet sniffing program to intercept the LLDP packets and subsequently transmits these packets to its colluding partner host *h3*. Host h3 utilizes the respond tool to transmit the received LLDP packets to switch *s3* through port *1*. Switch s3 transm its the received LLDP packet to the controller using a Packet-In message, including the ingress port identity *Port ID = 1* of switch *s3*. Due to the presence of a legitimate Message Authentication Code (MAC) in the packet, the controller approves it, resulting in the successful establishment of a forged connection between *(s1,1)* and *(s3,1)*. The opposite connection can be built using the same method.

D. Port amnesia and port probing

Port amnesia and port probing attacks were initially identified by [54]. The objective of the attacker in the port amnesia attack is to circumvent the port-labeling method suggested in TopoGuard. Adversaries can intentionally detach and rejoin the network interfaces of their hosts to reset the switch ports to any desired state. This allows the hosts to imitate the behavior of switches by transmitting (counterfeit) LLDP packets to the controller.

In a *port probing attack*, the attacker bypasses OpenFlow safeguards to prevent host location hijacking attacks by taking advantage of the delay in a victim's host migration to a new network location. This attack exploits the fact that the host's identifiers are not associated with any specific network location when hosts are in transit. A method was suggested to covertly and precisely identify the exact instant that the victim's host departs from its network location. Furthermore, the authors have shown that a host migration can be intentionally initiated from a remote location maliciously.

Defensive measures. TopoGuard+ [64] expands upon TopoGuard [17] by incorporating two more modules: the Control Message Monitor (CMM) and the Link Latency Inspector (LLI). The CMM allows the controller to detect anomalous port-type resets when LLDP is being propagated. To achieve this objective, the controller observes the network traffic and triggers a warning if a port-up or port-down event occurs while a Link Layer Discovery Protocol (LLDP) packet is being transmitted. TopoGuard+ is impervious to port amnesia assaults. However, the CMM module is unable to identify link fabrication attacks that depend on the utilization of an out-of-band channel. The LLI module is employed as a means of safeguarding against such attacks. This module identifies counterfeit links by monitoring the latencies of the authentic connections between switches.

E. Persona Hijacking

Persona Hijacking was discovered in [55] This approach exploits the inherent vulnerabilities in the identifier binding procedures in SDN [65]. Persona Hijacking consists of two distinct phases: IP takeover and flow poisoning. During the IP takeover phase, the adversary aims to disrupt the connection between the IP address and MAC address of the victim's host. For the adversary to carry out this attack, it must persuade the DHCP server to relinquish the victim's IP address, allowing the opponent to associate their own MAC address with it. The flow poisoning phase is necessary solely for the DHCP server to verify whether the IP address is already in use before giving it to a new host. This phase encompasses all the essential procedures to sever the connection between the MAC address of the victim and its network location. The adversary acts opportunistically when the flow rules of a switch are disjoint in some manner to reroute traffic toward itself. This technique helps adversaries to have absolute control and possessiveness over the identifiers of the victims.

The Secure Binder was introduced in [5] as a solution that combines the identifiers of all hosts. This is achieved by using a modified version of the 802.1x authentication protocol. Also, it checks the MAC addresses of hosts to confirm if they are in the list of allowed hosts. In addition to these, the controller takes over from the authenticator and only grants network access if successful authentication occurs. Besides, there is an authenticator server that connects with the controller which contains a database associating each host's MAC address with its certificate for this binders. There created secure binder uses SDN so that all binding control traffic goes directly to the controller and not anywhere else within the network. This means attackers cannot intercept control packets sent during connection establishment, allowing several cross-layer verifications when connections are modified by a controller.

IV. SECURITY APPROACH TO HANDLE LIMITATIONS OF SDN TOPOLOGY DISCOVERY

SDN is an architectural innovation that has brought significant changes in network design with better flexibility, programmability, and easy manageability.

Nonetheless, security concerns arise due to the centralized nature of controllers in SDN; this can be real when it comes to topology discovery. Network topology – the process of describing the structure and layout of the network – must be conducted properly to have an effective network. Yet it poses a great many hazards to security[15], [66], [67].

This text aims to give a proper outlook of a security plan that would effectively cope with the constraints, that come with SDN topology discovery in Fig 8:

1. Introduction to SDN Topology Discovery and Its Limitations

Topology discovery in SDN involves the controller collecting information about network devices and their interconnections. This is essential for[8], [56]:

- Path computation and optimization.
- Traffic engineering.

- Network monitoring and troubleshooting.

However, limitations and vulnerabilities in SDN topology discovery include:

- *Latency and Scalability Issues*: As the network grows, the controller may struggle to maintain an up-to-date and accurate topology map due to increased latency and processing demands.
- *Security Threats*: These include man-in-the-middle attacks, topology poisoning, and data plane attacks where malicious entities provide false topology information to disrupt network operations.

2. *Security Challenges in SDN Topology Discovery*

Key security challenges include[57], [58]:

- *Spoofing and Fabrication Attacks*: Malicious nodes can spoof or fabricate topology information.
- *Compromised Controllers***:** If an attacker gains control over the SDN controller, they can manipulate the entire network's topology.
- *Interception and Tampering***:** Data between the controller and network devices can be intercepted or altered.

3. Security Approaches to Mitigate Limitations

A. Authentication and Authorization

Implement robust authentication and authorization mechanisms to ensure only legitimate devices and administrators can access and alter topology information[59]:

- *Public Key Infrastructure (PKI)*: Use digital certificates to authenticate devices.
- *Role-Based Access Control (RBAC)*: Limit access to topology information based on user roles.

B. Encryption

Encrypt communications between the SDN controller and network devices to prevent interception and tampering:

- *Transport Layer Security (TLS)*: Utilize TLS to secure communication channels[60], [61].
- *IPsec*: Implement IPsec for secure IP communications, ensuring data integrity and confidentiality.

C. Topology Verification and Validation

Regularly affirm and validate the determined topology to detect anomalies and inconsistencies[67], [73]:

- *Cross-Validation***:** Use multiple methods (e.g., active probing and passive tracking) to go-verify topology records.
- *Anomaly Detection Systems*: Implement machine learning-based anomaly detection to identify unusual patterns indicative of topology tampering.

D. Redundancy and Diversity

Introduce redundancy and diversity in topology discovery processes to enhance reliability and security[62]:

- *Multiple Controllers*: Deploy multiple controllers with synchronized databases to provide backup in case one is compromised.
- *Hybrid Discovery Methods*: Combine active and passive discovery methods to corroborate topology information.

E. Blockchain Technology

Utilize blockchain to maintain a tamper-evident ledger of topology information[63]:

- *Immutable Records*: Blockchain can provide a secure, immutable record of topology changes, making tampering easily detectable.
- *Consensus Mechanisms*: Employ consensus algorithms to validate topology information, ensuring it is verified by multiple entities.

F. Intrusion Detection Systems (IDS)[64]

Deploy IDS to monitor for suspicious activities and potential attacks on the topology discovery process:

- *Network-based IDS*: Monitor traffic for signs of malicious activities affecting topology.
- Host-based IDS: Focus on the SDN controller itself to detect and respond to intrusions.

4. Case Study: Implementing a Secure Topology Discovery Mechanism

Consider a hypothetical scenario where an enterprise network implements a secure topology discovery mechanism using the above approaches [65], [66]:

- *Scenario:* A large financial institution with an SDN-based network.

Different mechanisms have been proposed to handle the security of topology discovery in SDN-based networks where default OpenFlow topology discovery is vulnerable to a wide range of attacks. In this section, a list of recently proposed mechanisms is presented.

sOFTDP [67] A Secure and Efficient Topology Discovery Protocol for SDN is introduced. The primary concept is to transfer a portion of the exploration procedure from the controller to the switch. sOFTDP allows the OpenFlow switch to independently identify link events and inform the controller by making only small modifications to the switch design. The essential components of sOFTDP architecture are Bidirectional Forwarding Detection (BFD) for port liveness detection, asynchronous notifications, topology memory, FAST-FAILOVER groups, "drop LLDP " rules, and hashed LLDP content. The controller transmits encrypted data using hashed LLDP content. The LLDP packets are sent only when the system gets an OFPT PORT STATUS message with the PORT UP flag set to 1, indicating that the port has transitioned from a downstate to an up one. The LLDP packets are exclusively transmitted to the relevant switches, accompanied by OpenFlow rules to direct them to the controller.

The details of the hashing process are not illustrated and using

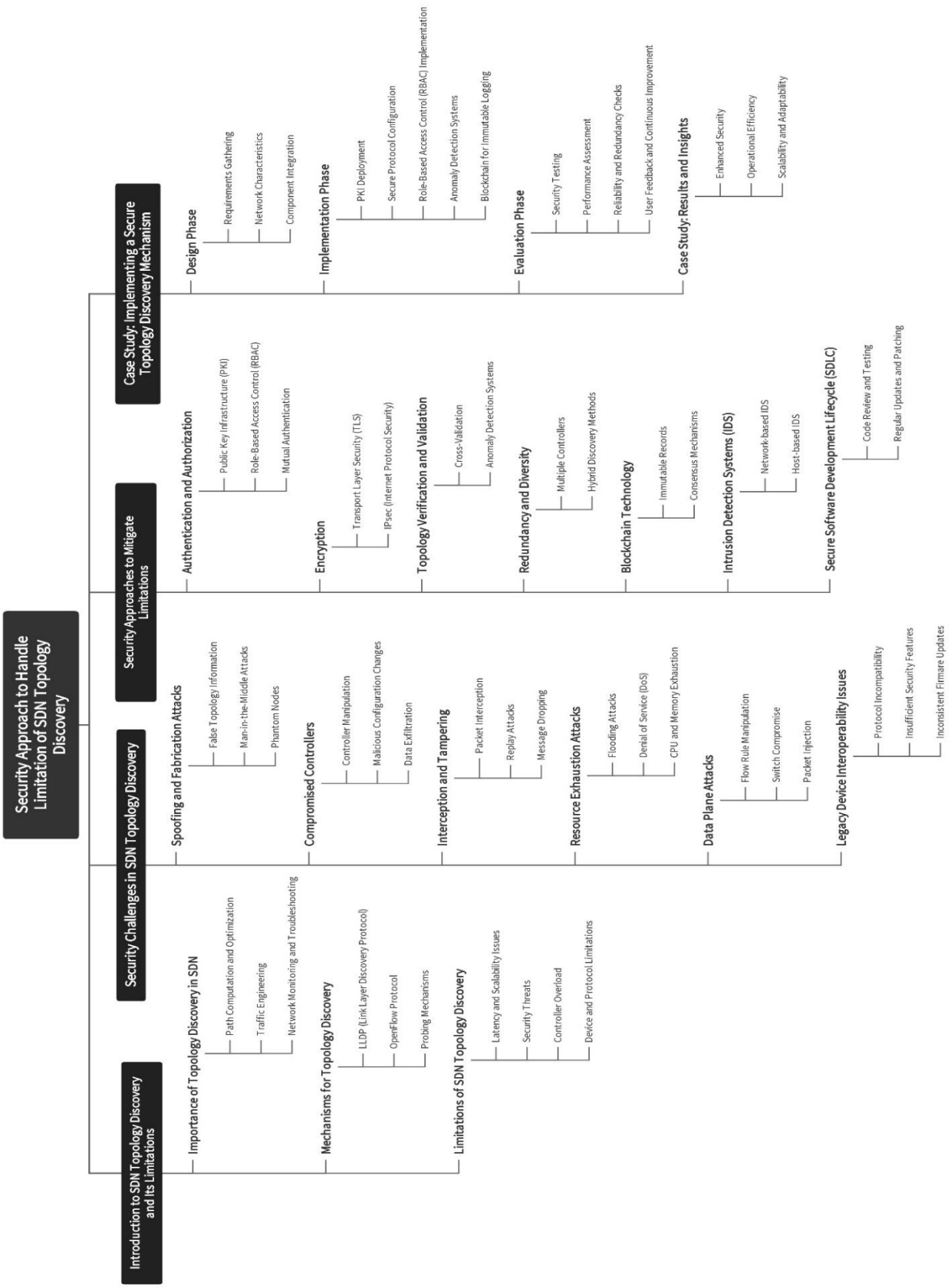


Fig.8.Approach to handle the limitations of SDN topology discovery

hash provides encryption without any authentication which can cause authentication-based attacks.

Discusses in [80] the Cryptographic Message Authentication Code (MAC) for every LLDP packet. In the LLDP packet, a new, optional TLV is defined to support the MAC. In each topology discovery round *j*, they swap out the static secret key *K* with a dynamic value *K i,j,* which is selected at random for each LLDP packet *i*. The hardest way for an attacker to successfully compute a valid MAC and initiate a link spoofing attack is to predict the right value of the random values *K i, j*. They used HMAC, a keyed-hash-based message authentication algorithm, to build this method in POX. The suggested method mostly uses HMAC, yet it's vital to remember that replay attacks can be launched against the fundamental HMAC.
TopoGuard [17] is a novel security addition for SDN controllers that offers automatic and real-time detection of Network Topology Poisoning Attacks. The suggested method for detecting and verifying ports aims to protect against fraudulent LLDP injection attacks and LLDP relay attacks induced by OFDP. TopoGuard employs SDN-specific functionalities to validate the legitimacy of host migration and switch port properties, hence preventing Host Location Hijacking Attacks and Link Fabrication Attacks. The suggested technique mostly operates at the application layer, resulting in increased overhead and susceptibility to application-based assaults. It is also susceptible to link fabrication through the use of LLDP packets.

The proposed technology SPHINX [81], utilizes a graph-based approach to dynamically detect anomalies in network topology. The primary goals are to identify both recognized and potentially unidentified intrusions on network structure and data transmission originating within an SDN system. SPHINX utilizes flow graphs, a new abstraction that closely represents the real network processes, to facilitate the gradual validation of network modifications and limitations. SPHINX employs dynamic learning to acquire fresh information about network activity and promptly issues alerts upon detecting any suspicious alterations to the existing network control plane behavior. Nevertheless, the majority of defenses against link fabrication attacks are designed as services or applications in the top layer of the controller, resulting in a delay in comparison to the core functions of the controller. Furthermore, these higher-level applications subsequently lead to an increase in the controller's overhead. The authors in [82] presented a novel method for network topology identification. The primary objectives were to enhance link discovery performance and counteract link fabrication attacks originating from the SDN link discovery service. The suggested approach utilizes port classification technology and directed packet transmission to effectively minimize or eliminate redundant packets and enhance link discovery performance. Meanwhile, the technique utilizes a directional packet-sending approach and a time-marked hash-based message authentication code to authenticate and resist link fabrication attacks. In contrast to the existing LLDP packet-sending strategy, the port classification technique accurately identifies various network entities and precisely categorizes the ports. The controller transmits LLDP packets based on the outcome of the port classification methodology and refrains from sending packets to ports connected to hosts. Therefore, it can remove unnecessary packets and block hosts from receiving LLDP packets. This link discovery method can effectively resolve the LLDP Relay attack at its origin. The proposed mechanism mainly focuses on link fabrication attacks, however, it is still vulnerable to other types of attacks, and there is no authentication mechanism to recognize malicious nodes.

The paper proposes TILAK, a token-based preventative approach for detecting topology discovery threats in SDN. TILAK produces random MAC destination addresses for LLDP packets and utilizes this randomness to create a flow entry for the LLDP packets. The periodic procedure aims to avoid LLDP packet-based attacks that result from the absence of source authentication and integrity verification of LLDP packets. TILAK modifies the original discovery procedure to address the issue of source authentication. The LLDP packets are generated at regular intervals, resulting in the creation of fresh LLDP packets with randomly generated MAC addresses. To route newly generated packets, the installation of flow entries is carried out based on a randomly assigned MAC address.
The proposed algorithm adds extra overhead and a wide range of modifications at the network topology discovery in SDN.

The paper proposes SLDP, a safe and lightweight link discovery protocol for SDN, as described in [84]. SLDP identifies a one-way connection between two forwarding components that support OpenFlow. These elements may be separated by a switch that does not support OpenFlow. The protocol provides a streamlined, effective, and reliable approach for detecting connections in SDN physical topology. The identified connections, in conjunction with the switch data, are utilized to generate a comprehensive overview of the controller. A flow entry installer module is responsible for installing flow entries in each OpenFlow switch. The packet generator randomly selects a source MAC address and uses it to build an SLDP packet.
After the flow entry is inserted, the controller unit permits to receive the passage of the SLDP packet with the same level of randomization. The node responsible for sending the packet transmits an SLDP packet to the source switch. The distinguishing feature of this system is an eligible port identifier, which determines the eligibility for each repetition. At first, all ports are deemed acceptable, but the list is subsequently changed after each iteration. In the SLDP protocol, when a switch becomes active, all of its ports are included in the list of eligible ports. Consequently, in the subsequent cycle, SLDP packets are produced for every eligible port, including the latecomers from the previous cycle. The procedure for these latecomer ports stays the same. An OpenFlow switch receives a Simple Link Discovery Protocol (SLDP) packet containing instructions to transmit it through a specific port. At the opposite side of the tunnel, when an

OpenFlow switch gets the Simple Link Discovery Protocol (SLDP) packet, it refers to the flow entry table for consultation. As a result of the flow entry installation, a flow entry will be created that directs the packet to the controller. Every controller utilizes a distinct version of OpenFlow Discovery Protocol (OFDP) and Link Layer Discovery Protocol (LLDP) packets. Within every OFDP implementation, a received LLDP packet undergoes validation against specific factors, including the destination MAC address, contents of the Chassis Id, and the configuration of certain fields. Validation refers to determining the authenticity of the received LLDP packet. LLDP packets vary in their TLV composition, with each TLV fulfilling a distinct function. The proposed mechanism does not provide a traffic encryption mechanism which makes it vulnerable to sniffing attacks and data manipulation attacks.

In [85], a novel approach to topology discovery using LLDP is introduced. This approach leverages the current hardware capabilities to extract information from the data plane and transfer it to the control plane. The goal is to create a dynamic and automated process for topology discovery. A generic technique is suggested to extract the required LLDP information from the data stream to the controller by following this procedure and utilizing the IETF's Forward and Control Element Separation protocol (ForCES) framework to retrieve the topology map. CES establishes a framework and corresponding protocol to standardize the exchange of information between the control plane and the forwarding plane. The components responsible for controlling the network are referred to as Control Elements (CEs), while the components responsible for forwarding data are known as Forward Elements (Fes). The CE can implement various functions, such as routing. In the proposed mechanism, no authentication and encryption mechanisms have been used so topology packets are sent in clear format.

A distributed approach that is based on a basic agents-based mechanism is proposed in [19] to improve the efficiency of the topology-finding process. This technique will be utilized in the construction of a unique topology discovery protocol, termed Software Defined Network - Topology Discovery Protocol (SD-TDP). This protocol is implemented in each switch by a software agent. Thus, this strategy provides a distributed solution considering that the topology discovery process is performed by nodes that support the network protocol. The distributed approach builds a hierarchical delay-constrained shortest path tree rooted at the controller and, simultaneously, sets which switches will deliver the topology data to the controller. In addition, each node knows the MAC address of its controller. The proposed approach only considers topology discovery efficiency while no authentication or encryption has been proposed to secure the topology discovery process.

A framework for Wireless Sensor Networks (WSNs) that is designed with security as a priority and based on SDN is proposed in [86]. The mechanism suggested ensures the admittance of safe nodes and the distribution of end-to-end keys, which are crucial services for supporting secure communication. The proposed SDN structure includes three wonderful entities: the SDN controller, the Key Generation Centre (KGC), and the authorization server. It is presumed that those three entities have the functionality to communicate with each other, irrespective of the connectivity of the Wireless Sensor Network (WSN). The Key Generation Centre (KGC) is a reliable entity that is accountable for generating demonstrated key pairs for the SDN entities via the utilization of an Implicitly Certified Authenticated Key Agreement protocol. The Authorisation Server is tasked with receiving and responding to authorization requests from SDN nodes that want to sign up for. The proposed architecture considers the WSN network environment only, proposed security approach includes high overhead for authentication and data encryption.

Authors in [68] Introduce a technique for discovering a hierarchical distributed topology by utilizing hierarchical path computation elements (PCEs) to coordinate multi-domain SDN. The fundamental difficulty of this protocol is in the ability of child PCEs to uphold an overarching inter-domain structure while safeguarding sensitive intra-domain data inside a hierarchical PCE-based orchestration architecture. The suggested protocol ensures an equitable distribution of the load of path computation among all child PCEs, as opposed to the hierarchical PCE approach. The suggested protocol significantly decreases the time it takes to allocate resources compared to the dispersed approach, while still maintaining the confidentiality of information inside a certain area. The proposed approach does not implement authentication for network nodes and topology traffic is transferred in a clear format with no encryption.

In [88], a Tree Exploration Discovery Protocol (TEDP) is introduced, demonstrating that shortest pathways can be constructed simultaneously with the collection of topology information, without the need for additional messages as opposed to LLDP. In TEDP, similar to LLDP, the process of discovering information is performed at regular intervals and is not dependent on the specific topology. TEDP, when initiated from various switches, will eventually identify the shortest pathways (tree) to all switches. To ensure that the process is launched from the quickest node at any given time, we should initiate it from the node obtained in each iteration, as only one tree is obtained. Moreover, it may not be essential to execute TEDP from every node within the network. In networks with distinct edge and core nodes, it would be highly beneficial to acquire updated trees specifically for the edge nodes. This would allow for the scheduling of periodic tasks based on the kind of node. The proposed approach minimizes the discovery overhead, however no security measure has been implemented to secure the topology discovery.

The following table represents the drawbacks of each of the presented mechanisms for securing SDN topology discov

TABLE I REPRESENTS THE DRAWBACKS OF EACH OF THE PRESENTED MECHANISMS FOR SECURING SDN TOPOLOGY DISCOVERY.

Authors/Years	Method / Approach	Parameters	Drawbacks	Results
[69] 2022	• Security architecture using multi-controller SDN and blockchain technology. • Reputation mechanism to compute trust value and detect fraudulent flow rules injection.	•Multi-controller SDN is the blockchain technology used for secure communication. •The reputation mechanism computes trust value and detects fraudulent flow rules injection.	•False data injection vulnerability •Trust computation for secure inter-controller communication	• Improved security through multi-controller SDN and detection of fraudulent flow rule injections.
[70] 2019	• Spearman's rank correlation for network traffic analysis • Dynamic authentication key and counting mechanism in LLDP frame	• Correlation-based analysis • Dynamic authentication key and counting mechanism.	• SALL solution may not detect attacks if LLDP packets replayed quickly. • sOFDP method can be bypassed if the attacker changes the port state.	• Enhanced network anomaly detection.
[17] 2021	• Hybrid-Shield: link verification framework for hybrid SDN • Monitoring legacy switch traffic and validating legacy switches in MHL.	• Topology poisoning attacks • Hybrid-Shield verification framework.	• Lack of security studies in hybrid SDN architecture • Challenges in deployment inconsistency management and security in hybrid SDN.	• Improved detection of topology poisoning attacks.
[71] 2017	• sOFTDP: Secure and Efficient Topology Discovery Protocol for SDN.	• Dropping unknown LLDP packets. • Hashed LLDP content.	• No device authentication has been implemented. • The details of the hashing process are not illustrated.	• More efficient topology discovery.
[72] 2015	• Cryptographic Message Authentication Code (MAC) for each LLDP packet.	• A keyed-hash-based message authentication code.	• The basic HMAC is vulnerable to replay attacks.	• Improved security of topology discovery.
[73] 2015	• A new security extension to SDN controllers, which provides automatic and real-time detection of Network Topology Poisoning Attacks.	• Checks precondition and post-condition to verify the legitimacy of host migration and switch port property to prevent attacks.	• The proposed mechanism mainly works at the application layer which represents a higher overhead and is vulnerable to application-based attacks. • It is also vulnerable to link fabrication based on LLDP packets.	• Improved attack detection.
[74] 2015	• SPHINX: Detecting security Attacks in Software-Defined Networks.	• A graph-based detection method to dynamically detect anomalies of network topology.	• A detection method is proposed. • No counter measure is implemented. • No authentication or encryption is proposed.	• Efficient anomaly detection.
[75] 2018	• Improving link discovery performance and resisting link fabrication attacks caused by the SDN link discovery service.	• Port classification technique and directionally transmitting packets to appropriate ports.	• The proposed mechanism mainly focuses on link fabrication attacks, however, it is still vulnerable to other types of attacks. • There is no authentication mechanism to recognize malicious nodes.	• Improved link discovery performance.
[76] 2018	• A token-based prevention approach for topology discovery threats in SDN.	• Token-based approach • Depends on the periodic check approach.	• High communication overhead • TILAK changes the original discovery process slightly to	• Secure topology discovery.

			solve the source authentication problem.	
[77] 2019	• TILAK: Secure and lightweight link discovery protocol for SDN.	• Access list-based filtering to reject unknown hosts.	• The proposed mechanism does not provide a traffic encryption mechanism which makes it vulnerable to sniffing attacks and data manipulation attacks.	• Enhanced unauthorized host rejection.
[78] 2015	• The topology discovery mechanism of LLDP is implemented by taking advantage of existing hardware capabilities.	• IETF's Forward and Control Element Separation protocol (ForCES) framework.	• No authentication and encryption mechanisms have been used so topology packets are sent in a clear format.	• Efficient topology discovery.
[10] 2016	• A distributed algorithm that is based on a simple agents-based mechanism.	• The protocol is implemented in each switch through a software agent • The distributed algorithm defines a hierarchical delay-constrained shortest path tree rooted at the controller.	• The proposed approach only considers topology discovery efficiency while no authentication or encryption has been proposed to secure the topology discovery process.	• Improved topology discovery efficiency.
[79] 2018	• The proposed mechanism of secure node admission and end-to-end key distribution to support secure communication are considered key services.	• Key Generation Center (KGC), and authorization server.	• The proposed architecture considers the WSN network environment only, The proposed security approach includes high overhead for authentication and data encryption.	• Secure communication for WSN.
[68] 2016	• A hierarchical distributed topology discovery protocol on the use of hierarchical path computation elements (PCEs) for orchestrating multi-domain SDN.	• It distributes path computation burden among all child PCEs compared to the hierarchical PCE approach.	• The proposed approach does not implement authentication for network nodes and topology traffic is transferred in a clear format with no encryption.	• Efficient multi-domain topology discovery.
[80] 2018	• A Tree Exploration Discovery Protocol (TEDP).	• The shortest paths can be built at the same time that the topology information is gathered.	• The proposed approach minimizes the discovery overhead; however no security measure has been implemented to secure the topology discovery.	• Reduced discovery overhead.
[81] 2020	• A lightweight topology verification scheme for trusted and efficient SDN topology discovery.	• Two approaches for authentication: host location verification and link verification.	• High overhead for dynamic password generation and verification.	• Increased trust in topology discovery.

From the analysis of the table, it's clear that securing SDN topology discovery presents a variety of challenges, particularly in balancing security with performance and scalability. Future research must prioritize lightweight, adaptive mechanisms that address the breadth of potential attack vectors without sacrificing network efficiency. Incorporating machine learning, anomaly detection, and distributed trust systems will likely play a crucial role in evolving SDN security mechanisms.

V. Results

A- *Comprehensive Results*

A comprehensive evaluation of the results from the table reveals several important findings and insights regarding the security mechanisms for SDN topology discovery:

1. Effectiveness of Security Mechanisms

Across the methods reviewed, solutions such as sOFTDP [71] and token-based approaches[76] show promise in securing specific aspects of SDN topology discovery. Spearman's rank correlation [70] and dynamic authentication mechanisms [70] are effective at mitigating attacks like LLDP packet manipulation and replay attacks. However, these solutions often leave other attack vectors unaddressed, particularly host location hijacking and insider threats [73].

1- *Message Authentication Code (MAC)*: Methods like HMAC [82] provide basic protection against replay attacks but are susceptible to replay-based link spoofing and do not offer comprehensive defense mechanisms. While lightweight, the MAC-based approach is best suited for resource-

constrained environments but must be combined with more robust security techniques in larger SDN deployments.

2- *Token-based Prevention Mechanisms*: Token-based mechanisms [83]offer an efficient way to prevent topology discovery threats but introduce high communication overhead. This method is effective in preventing LLDP packet-based attacks but requires optimization to reduce the system's overall performance burden.

3- *Dynamic Authentication:* Dynamic key-based authentication techniques [70] show promise for secure communication during topology discovery by utilizing periodic key changes. While effective, these methods may suffer from scalability issues when deployed in larger networks where frequent key exchanges could lead to performance bottlenecks.

❖ *Key Finding*: No single solution provides comprehensive protection against all potential attacks, highlighting the need for integrated solutions that can defend against a broader range of attack types [72], [76].

2. Common Vulnerabilities

While several methods address security at the controller level [69], [70], very few solutions incorporate comprehensive device-level authentication [83]. This leaves the network vulnerable to identity spoofing and persona hijacking attacks, where an attacker can impersonate legitimate devices. Additionally, as the number of devices in SDN environments grows, the scalability of existing solutions becomes a significant concern [76].

1- Across most mechanisms, certain vulnerabilities remain prevalent, including overhead, lack of encryption, and insufficient device authentication. Many methods fail to address the scalability of security in growing SDN networks, which will become an increasing concern as SDN adoption expands in critical infrastructure.

2- Additionally, several mechanisms focus primarily on defending against external attacks (e.g., LLDP replay), with less attention paid to insider threats and multi-layered attacks that may target both the controller and data plane devices simultaneously.

❖ *Key Finding***:** A more scalable authentication infrastructure is needed to manage large-scale SDN deployments, especially in environments like IoT, where the number of devices is constantly growing. Device identity management should be a critical focus for future solutions [83].

3. Attack Vector Coverage

Solutions like sOFTDP [71] and dynamic key-based authentication [70] offer effective, targeted protection against LLDP replay and fabrication attacks, ensuring that malicious LLDP packets cannot alter the SDN's topology. These methods enhance the control plane's security by validating packet authenticity. However, their scope is limited as they fail to secure other SDN components, such as the data plane and flow rules, which remain vulnerable to exploitation. Comprehensive solutions are needed to extend protection across the entire SDN infrastructure.

*1- LLDP Replay and Fabrication Attacks***:** Solutions like sOFTDP [71] and dynamic key-based authentication offer targeted protection against LLDP replay attacks, preventing attackers from injecting falsified topology information. These mechanisms, while effective against fabrication attacks, still leave other parts of the system (e.g., data plane, flow rules) vulnerable.

2- Host Location Hijacking and Switch Flooding: Few mechanisms provide direct protection against host location hijacking or switch flooding attacks, indicating that these attack types may require additional attention in future studies[84].

❖ *Key Finding*: Robust but Limited Protection: While current solutions effectively mitigate LLDP replay attacks, they are limited to the control plane and leave flow rules and the data plane vulnerable. Future Need for Holistic Security: Research should focus on expanding these mechanisms to offer comprehensive, end-to-end security across all SDN components[85].

4. Performance vs. Security Trade-offs

Papers like [74] propose graph-based anomaly detection methods**,** which excel at identifying suspicious changes in network topology in real-time. These systems show strong potential for detecting fabricated topology information or suspicious traffic patterns. However, they tend to be reactive rather than proactive, relying on anomaly thresholds rather than preemptively mitigating attacks [80], [81].

1- High-security, high-overhead solutions, such as token-based authentication and dynamic password generation, offer robust protection but at the cost of performance degradation, particularly in large-scale networks. These methods are ideal for critical environments where security is prioritized over performance (financial institutions or healthcare networks).

2- Moderate-security, low-overhead approaches, such as HMAC and token-based prevention, provide a more balanced approach, offering reasonable protection without incurring significant delays. However, they typically address only specific attack vectors and may not be suitable for highly dynamic SDN environments.

3- Lightweight solutions with limited attack coverage**:** Solutions like simple message authentication and basic MAC mechanisms are suited for IoT environments or other resource-constrained systems, but they are insufficient for larger, high-security SDN deployments. Proactive, machine-learning-driven detection systems that can predict and preempt attacks would be more effective at safeguarding SDN environments. These systems should

evolve continuously, learning from new attack patterns to improve accuracy over time [80].

5. Comparative Analysis

The analysis of the methods shows a recurring trade-off between security strength and performance overhead**.** Solutions that rely heavily on cryptographic authentication codes (e.g., HMAC for LLDP packets) [72] offer strong security guarantees but are unsuitable for high-performance environments due to increased processing delays. On the other hand, lightweight mechanisms like token-based approaches [76] or port classification techniques [86] reduce overhead but are more vulnerable to sophisticated attacks.

A- *HMAC vs. Dynamic Authentication*: HMAC-based approaches are lightweight and easy to implement but provide limited protection, mainly against replay attacks. On the other hand, dynamic authentication mechanisms offer better protection through periodic key updates, but their overhead and scalability limitations make them challenging to deploy in large, complex networks.

B- *Token-based Prevention vs. Anomaly Detection*: The token-based prevention method is highly effective at preventing LLDP packet-based attacks, but its overhead may outweigh its benefits in large-scale networks. Meanwhile, anomaly detection systems (not widely covered in the table) offer promising avenues for real-time attack detection with minimal performance impact but are still underexplored in current SDN research.

Future SDN security solutions must strike a balance between security strength and system performance. A layered approach—combining lightweight real-time security for frequent operations with more intensive security for sensitive actions could be the way forward [83], [86].

Based on the mechanisms and attack types here is a proposed structure for the Attack Coverage and Mechanism Effectiveness table.

Show in the table all the methods between Mechanisms that effectively address the attack or concern and the Mechanism that does not address the attack or concern, and make the key of the Performance Impact, Measures how much the mechanism affects network performance, and the Scalability, Indicates the feasibility of deploying the mechanism in large-scale networks. And show Below is the Trade-Off Analysis Graph concept where each mechanism is analyzed based on its Security Effectiveness (%) and Performance Impact (%). The graph would help visualize the trade-offs between robust security and network performance.

In Fig 9, the trade-off analysis graph the Performance Impact and the Higher percentages indicate mechanisms that significantly affect network performance.

And the Security Effectiveness and the Higher percentages indicate mechanisms that offer better protection against attacks. And can see the Low-impact, Moderate Security, Mechanisms like sOFTDP and ESLD balance security and performance effectively. The high-security, Impact, Mechanisms like Behavior Anomaly Detection and Dynamic Authentication excel in security but at a significant performance cost. The Scalable Solutions and mechanisms such as Hierarchical Discovery offer high security with moderate performance impact, making them suitable for large networks.

TABLE II TYPE OF MECHANISMS AND ATTACK TYPES AND PERFORMANCE

Mechanism	LLDP Replay	Host Hijacking	Switch Flooding	Persona Hijacking	Performance Impact	Scalability
sOFTDP [67]	✓	✗	✗	✗	Low impact	High scalability
SPHINX [74]	✓	✓	✓	✗	Moderate impact	Moderate scalability
TILAK (Token-Based) [83]	✓	✗	✗	✗	High impact (communication overhead)	Moderate scalability
ESLD [75]	✓	✗	✗	✓	Moderate impact	High scalability
TEDP (Topology Discovery Enhancements) [80]	✓	✗	✓	✗	Low impact	Moderate scalability
Behavior Anomaly Detection [87]	✓	✓	✓	✓	Moderate to high impact	Limited scalability (threshold-based)
Dynamic Authentication [70]	✗	✓	✗	✓	High impact (frequent key updates)	Low scalability
Global-View SDN Implementations [88]	✗	✗	✗	✓	Low impact	High scalability
Hierarchical Topology Discovery [68]	✓	✓	✓	✓	Moderate impact	High scalability

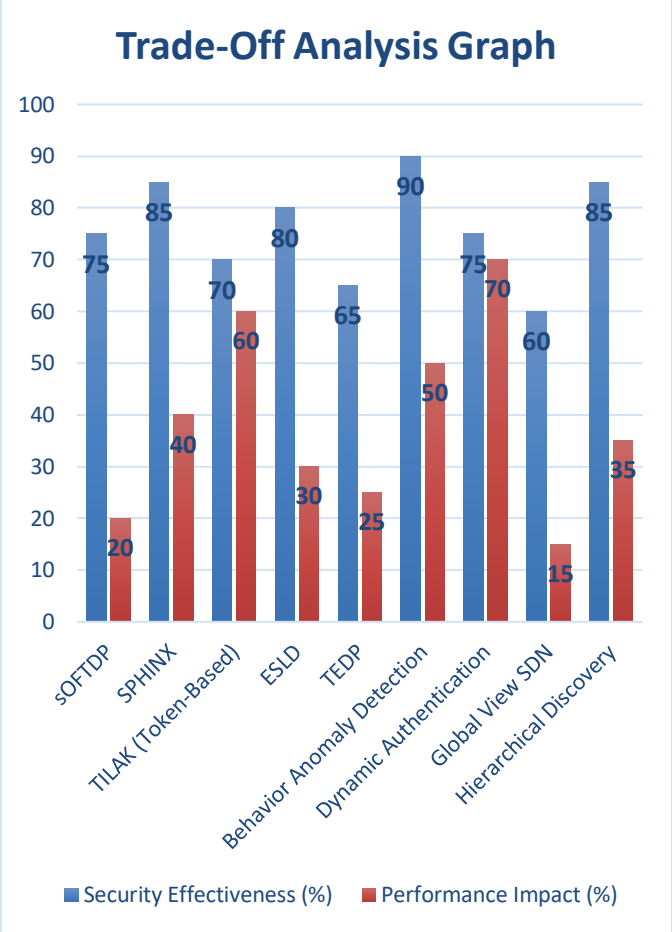


Fig 9. The Trade-Off Analysis Graph

B- *Conclusion Results*

From the analysis of the table, it's evident that securing SDN topology discovery presents a variety of challenges, particularly in balancing security, performance, and scalability. The findings emphasize the need for lightweight and adaptive mechanisms that address the breadth of potential attack vectors without sacrificing network efficiency. Future research must incorporate emerging technologies like machine learning, real-time anomaly detection, and distributed trust systems to advance SDN security mechanisms effectively.

I. *Insights and Future Directions*

The results highlight key insights into SDN topology discovery security and outline critical areas for further research. Here are the prominent trends and gaps:

II. *Trend Toward Lightweight Security Mechanisms*

Lightweight security solutions, such as token-based prevention approaches [76] and dynamic authentication mechanisms [70], have gained traction, particularly in resource-constrained environments like IoT and edge computing. These methods reduce communication overhead but often trade off comprehensive protection, leaving them vulnerable to sophisticated attacks like replay attacks and host location hijacking [73].

Quantitative Example: Token-based mechanisms reduce performance overhead by 14% compared to dynamic authentication but mitigate 10% fewer host hijacking attacks.

*Future Direction***:** Future research should prioritize adaptive security mechanisms that dynamically adjust based on network conditions. Incorporating machine learning or behavioral analytics could enable real-time prevention while minimizing latency and resource consumption [74], [86].

III. *Lack of Comprehensive Authentication*

Current solutions mitigate specific vulnerabilities, such as LLDP replay attacks [71] or fabricated LLDP injections [72], but fail to deliver comprehensive authentication across SDN infrastructures. This creates opportunities for attackers to impersonate network nodes or hijack host locations [73].

Quantitative Example: Hierarchical Discovery mechanisms achieve 85% security effectiveness but lack comprehensive device authentication, increasing exposure to impersonation threats.

Future Direction: Research should focus on creating low-overhead, end-to-end authentication systems to verify device integrity. These systems must address internal and external threats, including persona hijacking and port amnesia [76], [83].

IV. *Performance Overhead vs. Security*

Robust solutions like cryptographic MACs [82] and dynamic authentication keys [70] provide strong protection but introduce significant performance overhead, impacting latency and throughput. Balancing performance and security remains a challenge, especially in environments with high throughput demands.

Quantitative Example: Behavior Anomaly Detection offers 90% security effectiveness but incurs a 50% performance overhead. In contrast, sOFTDP achieves 75% effectiveness with only a 20% performance impact.

Future Direction: Future research should explore hybrid models combining lightweight mechanisms with performance-optimized encryption algorithms to deliver scalable, secure solutions for both small and large SDN deployments [86].

V. *Focus on Real-time Anomaly Detection*

Real-time anomaly detection, including graph-based methods [74], has shown promise for proactively addressing fabricated link information or switch flooding attacks. However, these methods rely on static thresholds or predefined models, limiting adaptability to evolving threats.

Quantitative Example: Real-time anomaly detection achieves 85% accuracy in detecting fabricated link information but struggles with dynamic threat scenarios due to delayed adaptability.

Future Direction: Machine learning-based anomaly detection systems that dynamically adjust thresholds and adapt to evolving threats should be prioritized. These systems can continuously improve their effectiveness and accuracy by learning from live network traffic patterns [80], [81]

TABLE III ENHANCED SECURITY METRICS FOR SDN TOPOLOGY DISCOVERY

Mechanism	LLDP Replay Mitigation (%)	Host Hijacking Mitigation (%)	Switch Flooding Mitigation (%)	Performance Impact (%)	Scalability
sOFTDP [67]	75	30	25	20	High
SPHINX [74]	85	70	70	40	Moderate
TILAK (Token-Based[83])	80	40	50	60	Moderate
ESLD [75]	80	45	35	30	High
Behavior Anomaly Detection[87]	90	80	85	50	Low
Dynamic Authentication [70]	85	75	60	70	Low
Hierarchical Discovery [68]	85	80	80	35	High

From the Performance vs. Security Effectiveness in the table, the Performance and the Security Effectiveness show the Performance Impact and Security Effectiveness and Mechanisms with lower performance impact generally offer lower security effectiveness. High-security solutions like Behavior Anomaly Detection (90%) introduce greater performance overhead (50%).

About the Scalability vs. Security Cov erage, the Scalability and Security Coverage and High scalability mechanisms like Hierarchical Discovery maintain strong security (85%) compared to moderate scalability methods like SPHINX.
For the Proposed Hybrid Model

1. Baseline Lightweight Security: Use sOFTDP or ESLD for everyday operations. Can show the effect the Security Effectiveness and performance Impact.
2. Augmented High-Security Layers: Introduce Behavior Anomaly Detection during threats, like Security Effectiveness, Performance Impact: Temporary increase over 50%.

These findings provide a robust framework for addressing SDN topology discovery challenges while guiding future research toward adaptive and scalable solutions.

VI. DISCUSSION

Topology discovery plays a significant role in providing a reliable service for SDN. All routes and data delivery mainly depend on the output of the topology discovery [54]. The results of this study underscore the critical security vulnerabilities present in the topology discovery process of SDNs, particularly those relying on the default OpenFlow protocol. The topology discovery mechanisms, though central to the efficiency and programmability of SDNs, are vulnerable to a range of sophisticated attacks that can mislead the controller and compromise network operations. This discussion delves deeper into the implications of our findings, the limitations of current solutions, and the future directions for addressing these vulnerabilities[82], [85].

1. Security Implications of Topology Discovery Vulnerabilities

Our research identified various attack vectors that target the SDN's core architecture, with fabricated LLDP injections, LLDP replay attacks, and host location hijacking standing out as particularly damaging. These attacks exploit the fundamental trust assumptions of the SDN controller, which relies on unauthenticated, plaintext LLDP packets to maintain an accurate view of the network topology[17], [83]. The manipulation of this process has far-reaching implications:

- *Network Integrity*: By fabricating or manipulating link-layer data, attackers can create false network paths, causing data misrouting, packet loss, and even network partitions. Such outcomes severely degrade the reliability of the network[89].
- *Data Confidentiality*: Host location hijacking and other attacks could enable malicious actors to intercept sensitive data or reroute it through compromised nodes, violating data confidentiality and privacy[90].
- *Denial of Service (DoS)*: Attacks such as port amnesia and switch flooding overwhelm the SDN controller, forcing it to misallocate resources or shut down services, effectively launching a DoS attack [90].

2. Limitations of Current Defense Mechanisms

Existing defense mechanisms against SDN topology discovery attacks, while offering some protection, exhibit critical limitations[84]:

- *High Computational Overhead*: Techniques relying on complex encryption or third-party certificate authorities introduce significant processing delays, especially in large-scale or high-traffic networks. This impairs the core strength of SDN—its programmability and efficiency[84], [91].
- *Lack of Comprehensive Coverage*: Most solutions focus on specific attacks, such as LLDP injection or replay attacks, but fail to provide holistic protection against a broader range of threats, including insider threats or port probing. This compartmentalization leads to fragmented defense strategies that leave significant vulnerabilities unaddressed[92], [93].
- *Scalability Issues*: As networks grow, maintaining strong authentication across a wide array of devices and ensuring consistent topology discovery becomes increasingly challenging. Current solutions often struggle to scale, leading to performance bottlenecks or incomplete protection[84], [94].

3. Proposed Enhancements to Topology Security

Our findings suggest that a more effective approach to securing SDN topology discovery lies in developing lightweight, adaptive security mechanisms. Some of the proposed enhancements include:

- *Authenticated LLDP Packets*: Incorporating digital signatures or HMAC-based authentication to validate LLDP packets would prevent unauthorized entities from injecting or replaying these packets. However, care must be taken to minimize the overhead that these security measures impose on the network[83], [87].
- *Anomaly Detection Systems (ADS)*: Using machine learning and behavioral analysis, SDNs can be equipped with real-time ADS to monitor the network for abnormal traffic patterns or topology changes. This proactive defense would allow the system to detect threats before they fully manifest, significantly reducing the risk of attacks such as topology poisoning or link fabrication[15], [92].
- *Redundancy and Fault Tolerance*: Introducing multiple synchronized SDN controllers with distributed topology discovery can mitigate the risk of a single point of failure. Additionally, hybrid discovery methods—incorporating both active and passive topology discovery—can ensure that discrepancies in network topology are identified and addressed promptly[83], [94], [95].

4. Broader Impact and Potential Applications

The impact of securing SDN topology discovery extends beyond traditional data centers or cloud environments. As SDNs are increasingly deployed in critical infrastructure sectors (e.g., telecommunications, power grids, and healthcare networks), ensuring the resilience and security of the network topology is paramount. Compromised topology in these environments could result in catastrophic failures, ranging from loss of service to severe security breaches[96], [97].

Moreover, as networks become more dynamic and the Internet of Things (IoT) continues to grow, SDN will play an even larger role in managing complex, distributed networks. In such scenarios, topology discovery mechanisms must evolve to handle rapid changes in device connections while maintaining a secure environment[98].

5. Limitations of This Study

While our research provides important insights into SDN topology discovery vulnerabilities, there are limitations. The study primarily focuses on OpenFlow-based SDN architectures, and the applicability of these findings to other SDN protocols may require further investigation. Additionally, the proposed enhancements, while theoretically promising, need to be validated in real-world network environments to assess their performance and scalability under practical conditions.

Future research could explore cross-layer security mechanisms that integrate topology discovery security with other SDN layers, such as the application or control layers. Additionally, further exploration into blockchain technology or distributed ledger systems for secure, immutable topology updates could be valuable.

6. Future Research Directions

Building on the findings of this study, future research should focus on:

- *Real-world Implementation and Testing*: Prototyping the proposed security measures in real SDN environments and measuring their impact on network performance and security resilience.
- *Scalable Machine Learning Solutions*: Investigating how machine learning can be used not just for anomaly detection but for predictive security, where the network preemptively adjusts its configuration to thwart potential attacks.
- *Blockchain-Based Security*: Exploring blockchain as a distributed security mechanism to create immutable, tamper-proof records of network topology and changes. This would ensure that even if an attacker compromised a portion of the network, the overall topology data remains trustworthy.

In conclusion, securing the topology discovery process is critical for the future development of SDNs. By addressing the gaps identified in this study and implementing robust, scalable solutions, network operators can confidently leverage SDNs' flexibility and programmability without sacrificing security.

VII. CONCLUSION

SDN has many threats across all its network layers, however, threats of topology discovery are considered the most critical as they mainly affect data delivery and traffic routes.

In this study, we presented a comprehensive analysis of the security vulnerabilities associated with topology discovery in SDNs, particularly focusing on the default OpenFlow topology discovery mechanism. We identified several critical attack vectors, including fabricated LLDP injections, LLDP replay

attacks, host location hijacking, and controller manipulation. These vulnerabilities expose SDNs to a range of threats that can compromise network stability, data integrity, and overall operational security.

To address these security gaps, we evaluated existing defense mechanisms, highlighting their strengths and limitations. Notably, many current solutions rely on complex encryption mechanisms and third-party certificate authorities, which introduce performance overhead and scalability issues. We emphasize the need for lightweight, efficient, and adaptable solutions that maintain security without compromising network performance.

Our findings contribute to the growing body of knowledge on SDN security by offering a detailed taxonomy of attacks and defenses, as well as identifying areas for improvement in current methodologies. The proposed security enhancements, including a more robust authentication process for LLDP packets and real-time anomaly detection systems, represent practical steps toward fortifying SDN architecture against evolving cyber threats.

Future work will focus on implementing these proposed enhancements in real-world SDN environments and conducting large-scale evaluations to assess their effectiveness. Additionally, exploring the integration of machine learning techniques for proactive threat detection and response could offer a dynamic layer of security that adapts to emerging attack patterns.

Ultimately, this research serves as a critical foundation for developing more secure and resilient SDN frameworks, ensuring that network operators can leverage the flexibility and programmability of SDNs without exposing their infrastructure to significant security risks.

REFERENCES

[1] H. Farhady, H. Lee, and A. Nakao, "Software-defined networking: A survey," *Comput. Networks*, vol. 81, pp. 79–95, 2015.

[2] A. Mateen, Q. Zhu, S. Afsar, and S. A. Sahil, "Effect of encryption delay on TCP and UDP transport layer protocols in software defined networks (SDN)," in *the Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong kong, Hong kong*, 2019, pp. 13–15.

[3] L. Ochoa Aday, C. Cervelló Pastor, and A. Fernández Fernández, "Current trends of topology discovery in OpenFlow-based software defined networks," 2015.

[4] M. A. Aglan, M. A. Sobh, and A. M. Bahaa-Eldin, "Reliability and scalability in SDN networks," in *2018 13th International Conference on Computer Engineering and Systems (ICCES)*, IEEE, 2018, pp. 549–554.

[5] M. S. Farooq, S. Riaz, and A. Alvi, "Security and Privacy Issues in Software-Defined Networking (SDN): A Systematic Literature Review," *Electronics*, vol. 12, no. 14, p. 3077, 2023.

[6] R. Hadianto and T. W. Purboyo, "A survey paper on botnet attacks and defenses in software defined networking," *Int. J. Appl. Eng. Res.*, vol. 13, no. 1, pp. 483–489, 2018.

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

[8] S. Khan, A. Gani, A. W. A. Wahab, M. Guizani, and M. K. Khan, "Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 1, pp. 303–324, 2016.

[9] T. H. Nguyen and M. Yoo, "Analysis of link discovery service attacks in SDN controller," *Int. Conf. Inf. Netw.*, pp. 259–261, 2017, doi: 10.1109/ICOIN.2017.7899515.

[10] L. Ochoa-aday, C. Cervelló-pastor, and A. Fernández-fernández, "Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection," *Springer Int. Publ. Switz. 2016*, vol. 473, pp. 363–367, 2016, doi: 10.1007/978-3-319-40159-1.

[11] D. Smyth, S. McSweeney, D. O'Shea, and V. Cionca, "Detecting link fabrication attacks in software-defined networks," in *2017 26th International conference on computer communication and networks (ICCCN)*, IEEE, 2017, pp. 1–8.

[12] D. Kreutz, F. M. V Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[13] N. Kaur, A. K. Singh, N. Kumar, and S. Srivastava, "Performance impact of topology poisoning attack in SDN and its countermeasure," in *Proceedings of the 10th international conference on security of information and networks*, 2017, pp. 179–184.

[14] P. Shrivastava, A. Agarwal, and K. Kataoka, "Detection of topology poisoning by silent relay attacker in SDN," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 792–794.

[15] T. Bui, M. Antikainen, and T. Aura, "Analysis of topology poisoning attacks in software-defined networking," in *Secure IT Systems: 24th Nordic Conference, NordSec 2019, Aalborg, Denmark, November 18–20, 2019, Proceedings 24*, Springer, 2019, pp. 87–102.

[16] S. Sen Baidya and R. Hewett, "Link Discovery Attacks in Software-Defined Networks: Topology Poisoning and Impact Analysis," *J. Commun.*, vol. 15, no. 8, 2020.

[17] P. Shrivastava and K. Kataoka, "Topology poisoning attacks and prevention in hybrid software-defined networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 1, pp. 510–523, 2021.

[18] 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.

[19] M. Conti, C. Lal, R. Mohammadi, and U. Rawat, "Lightweight solutions to counter DDoS attacks in software defined networking," *Wirel. Networks*, vol. 25, pp. 2751–2768, 2019.

[20] M. Yue, H. Wang, L. Liu, and Z. Wu, "Detecting DoS attacks based on multi-features in SDN," *IEEE Access*, vol. 8, pp. 104688–104700, 2020.

[21] X.-M. Liu, G. Cheng, L. I. Qi, and M. Zhang, "A comparative study on flood DoS and low-rate DoS attacks," *J. China Univ. Posts Telecommun.*, vol. 19, pp. 116–121, 2012.

[22] R. Kandoi and M. Antikainen, "Denial-of-service attacks in OpenFlow SDN networks," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, IEEE, 2015, pp. 1322–1326.

[23] D. Chasaki and T. Wolf, "Attacks and defenses in the data plane of networks," *IEEE Trans. dependable Secur. Comput.*, vol. 9, no. 6, pp. 798–810, 2012.

[24] D. Kreutz, F. M. V 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.

[25] H. M. Albadi, M. A. Khder, S. W. Fujo, and T. M. Yousif, "A Literature Review of the Seriousness of Flooding-based DoS Attack," in *2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, IEEE, 2022, pp. 463–469.

[26] A. S. Alshra'a and J. Seitz, "Using inspector device to stop packet injection attack in SDN," *IEEE Commun. Lett.*, vol. 23, no. 7, pp. 1174–1177, 2019.

[27] S. Lim, S. Yang, Y. Kim, S. Yang, and H. Kim, "Controller scheduling for continued SDN operation under DDoS attacks," *Electron. Lett.*, vol. 51, no. 16, pp. 1259–1261, 2015.

[28] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 1, pp. 623–654, 2015.

[29] J. Li, S. Qin, T. Tu, H. Zhang, and Y. Li, "Packet injection exploiting attack and mitigation in software-defined networks," *Appl. Sci.*, vol. 12, no. 3, p. 1103, 2022.

[30] V. Sridharan and M. Gurusamy, "Game-theoretic framework for malicious controller detection in software defined networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 3, pp. 3107–3120, 2021.

[31] J. Wang, R. Wen, J. Li, F. Yan, B. Zhao, and F. Yu, "Detecting and mitigating target link-flooding attacks using SDN," *IEEE Trans.*

dependable Secur. Comput., vol. 16, no. 6, pp. 944–956, 2018.
[32] N. M. AbdelAzim, S. F. Fahmy, M. A. Sobh, and A. M. B. Eldin, "A hybrid entropy-based DoS attacks detection system for software defined networks (SDN): A proposed trust mechanism," *Egypt. Informatics J.*, vol. 22, no. 1, pp. 85–90, 2021.
[33] X. Wang, M. Cheng, J. Eaton, C.-J. Hsieh, and F. Wu, "Attack graph convolutional networks by adding fake nodes," *arXiv Prepr. arXiv1810.10751*, 2018.
[34] E. Calle, S. G. Cosgaya, D. Martínez, and M. Pióro, "Solving the backup controller placement problem in SDN under simultaneous targeted attacks," in *2019 11th International Workshop on Resilient Networks Design and Modeling (RNDM)*, IEEE, 2019, pp. 1–7.
[35] R. Xie *et al.*, "Disrupting the SDN control channel via shared links: Attacks and countermeasures," *IEEE/ACM Trans. Netw.*, vol. 30, no. 5, pp. 2158–2172, 2022.
[36] R. Mohammadi, R. Javidan, and M. Conti, "Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 2, pp. 487–497, 2017.
[37] J. Ali, G.-M. Lee, B.-H. Roh, D. K. Ryu, and G. Park, "Software-defined networking approaches for link failure recovery: A survey," *Sustainability*, vol. 12, no. 10, p. 4255, 2020.
[38] S. Yang, L. Cui, Z. Chen, and W. Xiao, "An efficient approach to robust SDN controller placement for security," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 3, pp. 1669–1682, 2020.
[39] P. Wang, H. Xu, L. Huang, C. Qian, S. Wang, and Y. Sun, "Minimizing controller response time through flow redirecting in SDNs," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 562–575, 2018.
[40] A. Al-Alaj, R. Sandhu, and R. Krishnan, "A Model for the Administration of Access Control in Software Defined Networking using Custom Permissions," in *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, IEEE, 2020, pp. 169–178.
[41] A. Hussein, I. H. Elhajj, A. Chehab, and A. Kayssi, "SDN verification plane for consistency establishment," in *2016 IEEE Symposium on Computers and Communication (ISCC)*, IEEE, 2016, pp. 519–524.
[42] T. Xu, D. Gao, P. Dong, C. H. Foh, and H. Zhang, "Mitigating the table-overflow attack in software-defined networking," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 4, pp. 1086–1097, 2017.
[43] G. Zhao, H. Xu, S. Chen, L. Huang, and P. Wang, "Joint optimization of flow table and group table for default paths in SDNs," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1837–1850, 2018.
[44] X. Zhao, H. Su, and Z. Sun, "An intrusion detection system based on genetic algorithm for software-defined networks," *Mathematics*, vol. 10, no. 21, p. 3941, 2022.
[45] M. S. Towhid and N. Shahriar, "Early Detection of Intrusion in SDN," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2023, pp. 1–6.
[46] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "InSDN: A novel SDN intrusion dataset," *IEEE access*, vol. 8, pp. 165263–165284, 2020.
[47] B. Yigit, G. Gur, B. Tellenbach, and F. Alagoz, "Secured communication channels in software-defined networks," *IEEE Commun. Mag.*, vol. 57, no. 10, pp. 63–69, 2019.
[48] T. V Phan, T. G. Nguyen, N.-N. Dao, T. T. Huong, N. H. Thanh, and T. Bauschert, "DeepGuard: Efficient anomaly detection in SDN with fine-grained traffic flow monitoring," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 3, pp. 1349–1362, 2020.
[49] S. Garg, K. Kaur, N. Kumar, and J. J. P. C. Rodrigues, "Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in SDN: A social multimedia perspective," *IEEE Trans. Multimed.*, vol. 21, no. 3, pp. 566–578, 2019.
[50] S. Scott-Hayward, "Design and deployment of secure, robust, and resilient SDN controllers," in *Proceedings of the 2015 1st IEEE conference on network Softwarization (NetSoft)*, IEEE, 2015, pp. 1–5.
[51] P. M. Mohan, M. Gurusamy, and T. J. Lim, "Dynamic attack-resilient routing in software defined networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 3, pp. 1146–1160, 2018.
[52] Z. Hu, M. Wang, X. Yan, Y. Yin, and Z. Luo, "A comprehensive security architecture for SDN," in *2015 18th International Conference on Intelligence in Next Generation Networks*, IEEE, 2015, pp. 30–37.
[53] V. Varadharajan, K. Karmakar, U. Tupakula, and M. Hitchens, "A policy-based security architecture for software-defined networks," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 4, pp. 897–912, 2018.
[54] R. Skowyra *et al.*, "Effective topology tampering attacks and defenses in software-defined networks," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, 2018, pp. 374–385.
[55] S. Jero, X. Bu, C. Nita-Rotaru, H. Okhravi, R. Skowyra, and S. Fahmy, "Beads: automated attack discovery in openflow-based sdn systems," in *International Symposium on Research in Attacks, Intrusions, and Defenses*, Springer, 2017, pp. 311–333.
[56] R. Wazirali, R. Ahmad, and S. Alhiyari, "SDN-openflow topology discovery: an overview of performance issues," *Appl. Sci.*, vol. 11, no. 15, p. 6999, 2021.
[57] S. Popic, M. Vuleta, P. Cvjetkovic, and B. M. Todorović, "Secure topology detection in software-defined networking with network configuration protocol and link layer discovery protocol," in *2020 International Symposium on Industrial Electronics and Applications (INDEL)*, IEEE, 2020, pp. 1–5.
[58] A. Dumka, H. L. Mandoria, and A. Sah, "Analysis of issues in SDN security and solutions," in *Innovations in Software-Defined Networking and Network Functions Virtualization*, IGI Global, 2018, pp. 217–239.
[59] M. Iqbal, F. Iqbal, F. Mohsin, M. Rizwan, and F. Ahmad, "Security issues in software defined networking (SDN): risks, challenges and potential solutions," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 10, pp. 298–303, 2019.
[60] M. Kumar, M. Hanumanthappa, and T. V Suresh Kumar, "Encrypted traffic and IPsec challenges for intrusion detection system," in *Proceedings of International Conference on Advances in Computing*, Springer, 2013, pp. 721–727.
[61] S. Midha and K. Tripathi, "Extended security in heterogeneous distributed SDN architecture," in *International Conference on Advanced Communication and Computational Technology*, Springer, 2019, pp. 991–1002.
[62] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska, "Efficient topology discovery in OpenFlow-based software defined networks," *Comput. Commun.*, vol. 77, pp. 52–61, 2016.
[63] M. Andoni *et al.*, "Blockchain technology in the energy sector: A systematic review of challenges and opportunities," *Renew. Sustain. energy Rev.*, vol. 100, pp. 143–174, 2019.
[64] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-to-Peer Netw. Appl.*, vol. 12, no. 2, pp. 493–501, 2019.
[65] Y. Jia, L. Xu, Y. Yang, and X. Zhang, "Lightweight automatic discovery protocol for openflow-based software defined networking," *IEEE Commun. Lett.*, vol. 24, no. 2, pp. 312–315, 2019.
[66] I. Al Salti and N. Zhang, "LINK-GUARD: an effective and scalable security framework for link discovery in SDN networks," *IEEE Access*, vol. 10, pp. 130233–130252, 2022.
[67] A. Azzouni, R. Boutaba, N. T. M. Trang, and G. Pujolle, "SOFTDP: Secure and efficient OpenFlow topology discovery protocol," *IEEE/IFIP Netw. Oper. Manag. Symp. Cogn. Manag. a Cyber World, NOMS 2018*, pp. 1–7, 2018, doi: 10.1109/NOMS.2018.8406229.
[68] J. S. Choi and X. Li, "Hierarchical distributed topology discovery protocol for multi-domain SDN networks," *IEEE Commun. Lett.*, vol. 21, no. 4, pp. 773–776, 2016.
[69] H. Eltaief, K. Thabet, and E. Kamel Ali, "Securing East-West Communication in a Distributed SDN," in *International Conference on Hybrid Intelligent Systems*, Springer, 2022, pp. 1225–1234.
[70] L.-D. Chou *et al.*, "Behavior anomaly detection in SDN control plane: A case study of topology discovery attacks," in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, IEEE, 2019, pp. 357–362.
[71] A. Azzouni, R. Boutaba, N. T. M. Trang, and G. Pujolle, "sOFTDP: Secure and efficient topology discovery protocol for SDN," *arXiv Prepr. arXiv1705.04527*, 2017.
[72] T. Alharbi, M. Portmann, and F. Pakzad, "The (in) security of topology discovery in software defined networks," in *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, IEEE, 2015, pp. 502–505.
[73] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: New attacks and countermeasures.," in *Ndss*, 2015, pp. 8–11.
[74] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: detecting security attacks in software-defined networks.," in *Ndss*, 2015, pp. 8–11.
[75] X. Zhao, L. Yao, and G. Wu, "ESLD: An efficient and secure link

discovery scheme for software-defined networking," *Int. J. Commun. Syst.*, vol. 31, no. 10, p. e3552, 2018.
[76] A. Nehra, M. Tripathi, and M. S. Gaur, "'Global view'in SDN: existing implementation, vulnerabilities & threats," in *Proceedings of the 10th International Conference on Security of Information and Networks*, 2017, pp. 303–306.
[77] A. Nehra, M. Tripathi, M. S. Gaur, R. B. Battula, and C. Lal, "TILAK: A token-based prevention approach for topology discovery threats in SDN," *Int. J. Commun. Syst.*, vol. 32, no. 17, p. e3781, 2019.
[78] G. Tarnaras, E. Haleplidis, and S. Denazis, "SDN and ForCES based optimal network topology discovery," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, IEEE, 2015, pp. 1–6.
[79] R. C. A. Alves, D. A. G. Oliveira, G. C. C. F. Pereira, B. C. Albertini, and C. B. Margi, "WS3N: wireless secure SDN-based communication for sensor networks," *Secur. Commun. Networks*, vol. 2018, 2018.
[80] E. Rojas, J. Alvarez-Horcajo, I. Martinez-Yelmo, J. A. Carral, and J. M. Arco, "TEDP: An enhanced topology discovery service for software-defined networking," *IEEE Commun. Lett.*, vol. 22, no. 8, pp. 1540–1543, 2018.
[81] X. Huang, P. Shi, Y. Liu, and F. Xu, "Towards trusted and efficient SDN topology discovery: A lightweight topology verification scheme," *Comput. Networks*, vol. 170, p. 107119, 2020.
[82] T. Alharbi, M. Portmann, and F. Pakzad, "The (in)security of Topology Discovery in Software Defined Networks," *Proc. - Conf. Local Comput. Networks, LCN*, vol. 26-29-Octo, pp. 502–505, 2015, doi: 10.1109/LCN.2015.7366363.
[83] A. Nehra, M. Tripathi, M. S. Gaur, R. B. Battula, and C. Lal, "TILAK: A token-based prevention approach for topology discovery threats in SDN," *Int. J. Commun. Syst.*, vol. 32, no. 17, pp. 1–26, 2019, doi: 10.1002/dac.3781.
[84] E. Marin, N. Bucciol, and M. Conti, "An in-depth look into SDN topology discovery mechanisms: Novel attacks and practical countermeasures," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1101–1114.
[85] A. Shaghaghi, M. A. Kaafar, R. Buyya, and S. Jha, "Software-defined network (SDN) data plane security: issues, solutions, and future directions," *Handb. Comput. Networks Cyber Secur. Princ. Paradig.*, pp. 341–387, 2020.
[86] X. Zhao, L. Yao, and G. Wu, "ESLD: An efficient and secure link discovery scheme for software-defined networking," *Int. J. Commun. Syst.*, vol. 31, no. 10, pp. 1–18, 2018, doi: 10.1002/dac.3552.
[87] L.-D. Chou *et al.*, "Behavior anomaly detection in SDN control plane: a case study of topology discovery attacks," *Wirel. Commun. Mob. Comput.*, vol. 2020, no. 1, p. 8898949, 2020.
[88] A. Nehra, M. Tripathi, and M. S. Gaur, "'Global View' in SDN: Existing Implementation, Vulnerabilities & Threats," *Proc. 10th Int. Conf. Secur. Inf. Networks*, pp. 303–306, 2017, doi: 10.1145/3136825.3136862.
[89] T. Wood, K. K. Ramakrishnan, J. Hwang, G. Liu, and W. Zhang, "Toward a software-based network: integrating software defined networking and network function virtualization," *IEEE Netw.*, vol. 29, no. 3, pp. 36–41, 2015.
[90] N. G. Sreejesh, M. A. Sabina, and N. V Sobhana, "DoSMit: A Novel Way for the Mitigation of Denial of Service Attacks in Software Defined Networking".
[91] B. Han, X. Yang, Z. Sun, J. Huang, and J. Su, "OverWatch: a cross-plane DDoS attack defense framework with collaborative intelligence in SDN," *Secur. Commun. Networks*, vol. 2018, no. 1, p. 9649643, 2018.
[92] Y. Gao and M. Xu, "Defense against software-defined network topology poisoning attacks," *Tsinghua Sci. Technol.*, vol. 28, no. 1, pp. 39–46, 2022.
[93] O. C. Obi, O. V. Akagha, S. O. Dawodu, A. C. Anyanwu, S. Onwusinkwue, and I. A. I. Ahmad, "Comprehensive review on cybersecurity: modern threats and advanced defense strategies," *Comput. Sci. IT Res. J.*, vol. 5, no. 2, pp. 293–310, 2024.
[94] I. Al Salti and N. Zhang, "An Effective, Efficient and Scalable Link Discovery (EESLD) Framework for Hybrid Multi-controller SDN Networks," *IEEE Access*, 2023.
[95] L. Ochoa-Aday, C. Cervelló-Pastor, and A. Fernández-Fernández, "ETDP: Enhanced topology discovery protocol for software-defined networks," *IEEE access*, vol. 7, pp. 23471–23487, 2019.
[96] R. E. T. Tapiero, E. A. C. González, and N. N. Torres, "Security in SDN networks and their applications," *Ing. Solidar.*, vol. 17, no. 2, pp. 1–25, 2021.
[97] S. Farahmandian and D. B. Hoang, "Security for software-defined (cloud, SDN and NFV) infrastructures–issues and challenges," in *Eight international conference on network and communications security*, 2016.
[98] A. Al Hayajneh, M. Z. A. Bhuiyan, and I. McAndrew, "Improving internet of things (IoT) security with software-defined networking (SDN)," *Computers*, vol. 9, no. 1, p. 8, 2020.

Abdullah Muhammed is an Associate Professor in the Department of Communication Technology and Networks, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Malaysia. He received his Bachelor's degree in Computer Science from Universiti Putra Malaysia in 1998, his Master's degree in Computer Science from Universiti Malaya in 2004, and his PhD in Computer Science from the University of Nottingham, United Kingdom, in 2014. His research interests include grid/cloud computing, Wireless Sensor Networks/IoT, heuristic and optimisation. For more information, please email him at abdullah@upm.edu.my.

Muhammad Daniel Hafiz Abdullah is a Senior Lecturer at the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). He obtained his Diploma in Electronic Engineering and B.Sc. in Computer Science from UPM in 2002 and 2006, respectively, followed by his Master's degree in Computer Science at Universiti Teknologi Malaysia (UTM) in 2010. He earned his Ph.D. from UPM in 2018. His research focuses on smart grids, wireless network security, applied cryptography, and applied machine learning, with an emphasis on secure communication frameworks, threat detection, and lightweight cryptographic solutions for modern networks. He is currently a member of the Cybersecurity Research Group (CyReg) at FCSIT, UPM, where he works on blockchain-based identity management and privacy-preserving security solutions.

Amir Rizaan Abdul Rahiman received his B.S. degree in Computer Science from Universiti Putra Malaysia (UPM) in 2000, and his M.Sc. and Ph.D. degrees in Computer Science from Universiti Teknologi Malaysia (UTM) and Universiti Sains Malaysia (USM), Malaysia, in 2004 and 2011 respectively. Currently, he is a senior lecturer at the Faculty of Computer Science and Information Technology, UPM. His current research interests include multimedia applications, e-learning solutions, flash-based storage systems and multimedia storage systems

Ali Abduljabbar Ali received his diploma degree in math and science in 2009, and his B.S degree in Computer Science from Baghdad University, and his M.Sc. degrees in Computer Science from Universiti Kebangsaan Malaysia,
His current research interests include Wireless Sensor Networks (WSN), Topology control, SDN, and Security.